Computer Algebra
Independent Integration Tests

Summer 2023 edition

4-Trig-functions/4.1-Sine/81-4.1.9-trig^m-a+b-sin^n+c-sin^-2-n-^p

Nasser M. Abbasi

September 5, 2023 Compiled on September 5, 2023 at 11:22pm

Contents

CHAPTER 1

INTRODUCTION

This report gives the result of running the computer algebra independent integration test.The download section in on the main webpage contains links to download the problems in plain text format used for all CAS systems. The number of integrals in this report is [19]. This is test number [81].

1.1 Listing of CAS systems tested

The following are the CAS systems tested:

1. Mathematica 13.3.1 (August 16, 2023) on windows 10.
2. Rubi 4.16.1 (Dec 19, 2018) on Mathematica 13.3 on windows 10
3. Maple 2023.1 (July, 12, 2023) on windows 10.
4. Maxima 5.47 (June 1, 2023) using Lisp SBCL 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
5. FriCAS 1.3.9 (July 8, 2023) based on sbcl 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
6. Giac/Xcas 1.9.0-57 (June 26, 2023) on Linux via sagemath 10.1 (Aug 20, 2023).
7. Sympy 1.12 (May 10, 2023) Using Python 3.11.3 on Linux.
8. Mupad using Matlab 2021a with Symbolic Math Toolbox Version 8.7 on windows 10.

Maxima and Fricas and Giac are called using Sagemath. This was done using Sagemath `integrate` command by changing the name of the algorithm to use the different CAS systems.

Sympy was run directly in Python not via sagemath.

1.2 Results

Important note: A number of problems in this test suite have no antiderivative in closed form. This means the antiderivative of these integrals can not be expressed in terms of elementary, special functions or `Hypergeometric2F1` functions. `RootSum` and `RootOf` are not allowed. If a CAS returns the above integral unevaluated within the time limit, then the result is counted as passed and assigned an A grade.

However, if CAS times out, then it is assigned an F grade even if the integral is not integrable, as this implies CAS could not determine that the integral is not integrable in the time limit.

If a CAS returns an antiderivative to such an integral, it is assigned an A grade automatically and this special result is listed in the introduction section of each individual test report to make it easy to identify as this can be important result to investigate.

The results given in in the table below reflects the above.

System	% solved	% Failed
Rubi	100.00 (19)	0.00 (0)
Mathematica	100.00 (19)	0.00 (0)
Maple	100.00 (19)	0.00 (0)
Mupad	100.00 (19)	0.00 (0)
Fricas	94.74 (18)	5.26 (1)
Giac	47.37 (9)	52.63 (10)
Sympy	31.58 (6)	68.42 (13)
Maxima	26.32 (5)	73.68 (14)

Table 1.1: Percentage solved for each CAS

The table below gives additional break down of the grading of quality of the antiderivatives generated by each CAS. The grading is given using the letters A,B,C and F with A being the best quality. The grading is accomplished by comparing the antiderivative generated with the optimal antiderivatives included in the test suite. The following table describes the meaning of these grades.

grade	description
A	Integral was solved and antiderivative is optimal in quality and leaf size.
B	Integral was solved and antiderivative is optimal in quality but leaf size is larger than twice the optimal antiderivatives leaf size.
C	Integral was solved and antiderivative is non-optimal in quality. This can be due to one or more of the following reasons 1. antiderivative contains a hypergeometric function and the optimal antiderivative does not. 2. antiderivative contains a special function and the optimal antiderivative does not. 3. antiderivative contains the imaginary unit and the optimal antiderivative does not.
F	Integral was not solved. Either the integral was returned unevaluated within the time limit, or it timed out, or CAS hanged or crashed or an exception was raised.

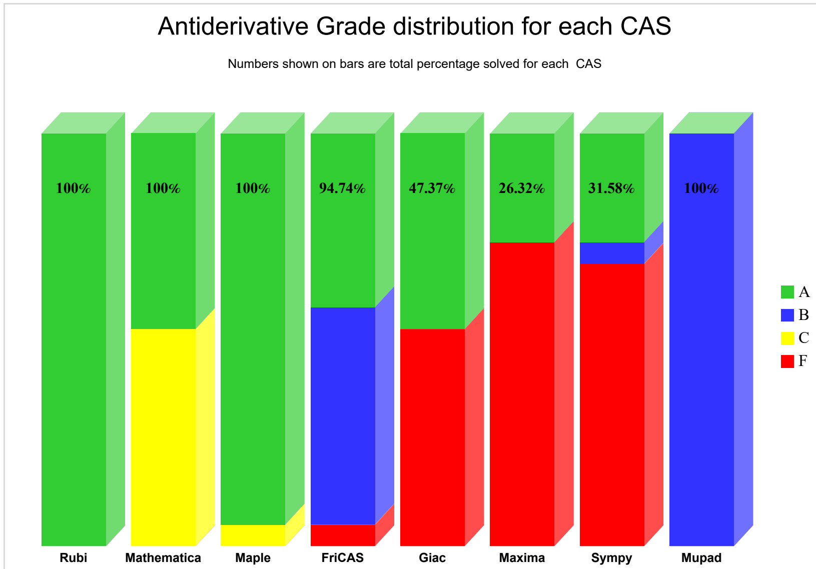
Table 1.2: Description of grading applied to integration result

Grading is implemented for all CAS systems. Based on the above, the following table summarizes the grading for this test suite.

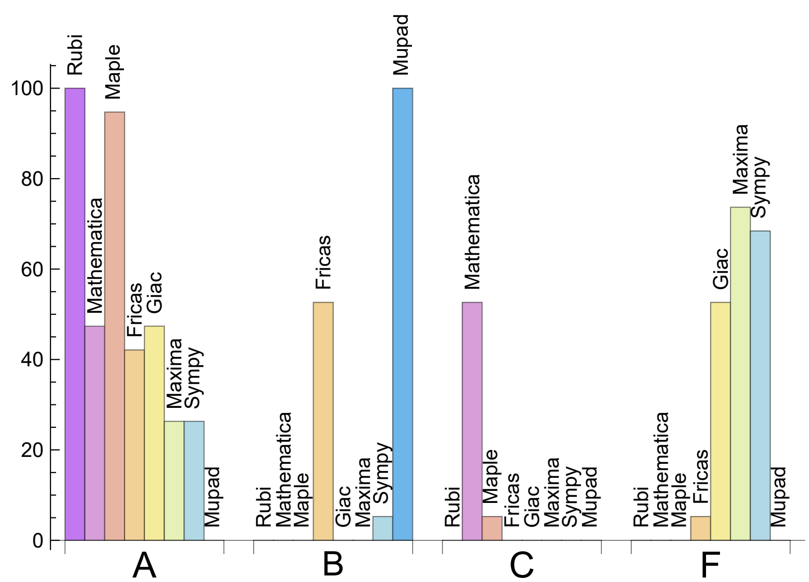
System	% A grade	% B grade	% C grade	% F grade
Rubi	100.000	0.000	0.000	0.000
Maple	94.737	0.000	5.263	0.000
Mathematica	47.368	0.000	52.632	0.000
Giac	47.368	0.000	0.000	52.632
Fricas	42.105	52.632	0.000	5.263
Maxima	26.316	0.000	0.000	73.684
Sympy	26.316	5.263	0.000	68.421
Mupad	0.000	100.000	0.000	0.000

Table 1.3: Antiderivative Grade distribution of each CAS

The following is a Bar chart illustration of the data in the above table.



The figure below compares the grades of the CAS systems.



The following table shows the distribution of the different types of failures for each CAS. There are 3 types failures. The first is when CAS returns the input within the time limit, which means it could not solve it. This is the typical failure and given as **F**.

The second failure is due to time out. CAS could not solve the integral within the 3 minutes time limit which is assigned. This is assigned **F(-1)**.

The third is due to an exception generated, indicated as **F(-2)**. This most likely indicates an interface problem between sagemath and the CAS (applicable only to FriCAS, Maxima and Giac) or it could be an indication of an internal error in the CAS itself. This type of error requires more investigation to determine the cause.

System	Number failed	Percentage normal failure	Percentage timeout failure	Percentage exception failure
Rubi	0	0.00	0.00	0.00
Mathematica	0	0.00	0.00	0.00
Maple	0	0.00	0.00	0.00
Mupad	0	0.00	0.00	0.00
Fricas	1	0.00	100.00	0.00
Giac	10	0.00	100.00	0.00
Sympy	13	46.15	53.85	0.00
Maxima	14	71.43	0.00	28.57

Table 1.4: Failure statistics for each CAS

1.3 Time and leaf size Performance

The table below summarizes the performance of each CAS system in terms of time used and leaf size of results.

Mean size is the average leaf size produced by the CAS (before any normalization). The Normalized mean is relative to the mean size of the optimal anti-derivative given in the input files.

For example, if CAS has **Normalized mean** of 3, then the mean size of its leaf size is 3 times as large as the mean size of the optimal leaf size.

Median size is value of leaf size where half the values are larger than this and half are smaller (before any normalization). i.e. The Middle value.

Similarly the **Normalized median** is relative to the median leaf size of the optimal.

For example, if a CAS has Normalized median of 1.2, then its median is 1.2 as large as the median leaf size of the optimal.

System	Mean time (sec)
Maxima	0.28
Sympy	0.33
Giac	0.33
Mathematica	0.83
Rubi	0.96
Maple	1.89
Mupad	18.30
Fricas	19.66

Table 1.5: Time performance for each CAS

System	Mean size	Normalized mean	Median size	Normalized median
Maxima	10.60	0.75	13.00	0.71
Sympy	25.17	1.09	13.50	0.71
Giac	75.78	1.01	17.00	1.00
Rubi	170.47	1.00	221.00	1.00
Maple	186.84	1.02	217.00	1.03
Mathematica	208.26	1.08	233.00	1.05
Fricas	3264.33	12.37	1107.50	5.13
Mupad	9923.79	35.03	5048.00	22.34

Table 1.6: Leaf size performance for each CAS

1.4 Performance based on number of rules Rubi used

This section shows how each CAS performed based on the number of rules Rubi needed to solve the same integral. One diagram is given for each CAS.

On the y axis is the percentage solved which Rubi itself needed the number of rules given the x axis. These plots show that as more rules are needed then most CAS system percentage of solving decreases which indicates the integral is becoming more complicated to solve.

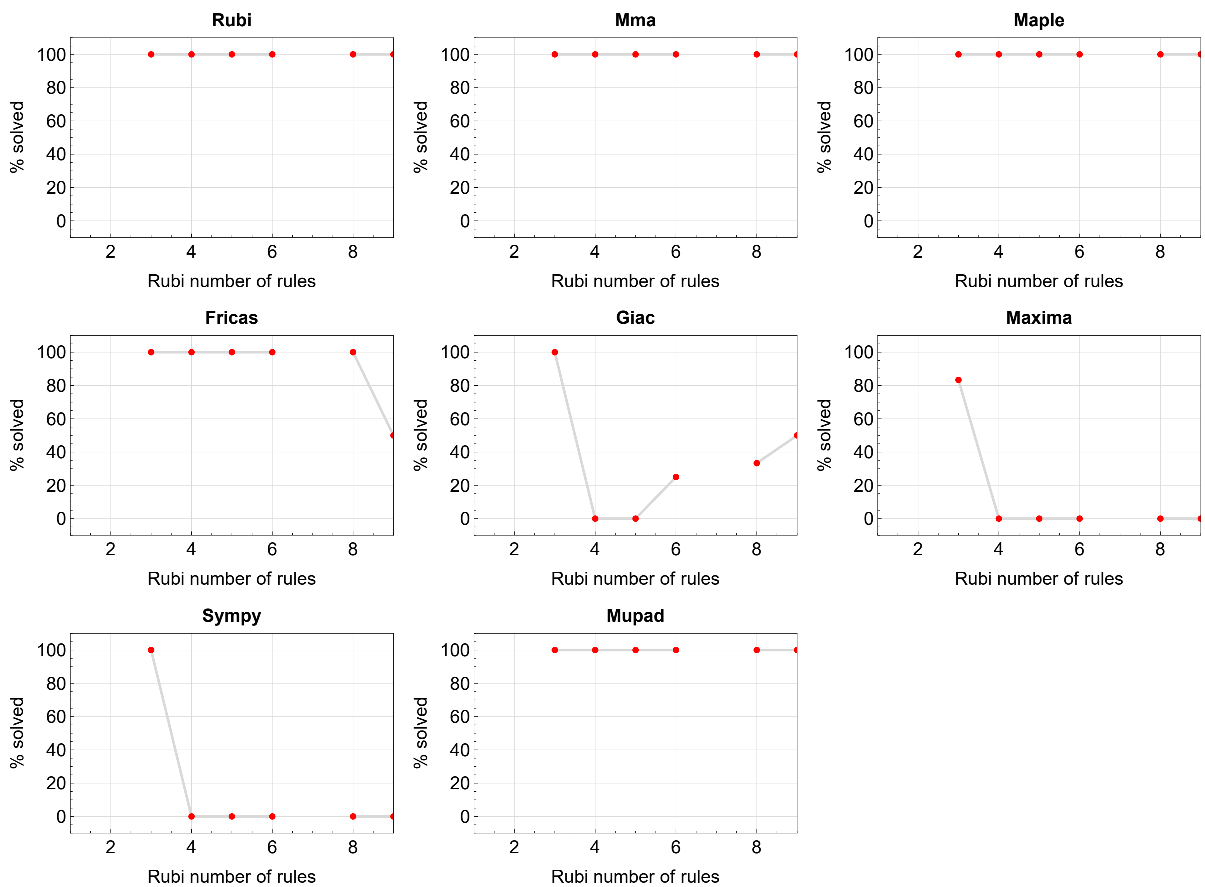


Figure 1.1: Solving statistics per number of Rubi rules used

1.5 Performance based on number of steps Rubi used

This section shows how each CAS performed based on the number of steps Rubi needed to solve the same integral. Note that the number of steps Rubi needed can be much higher than the number of rules, as the same rule could be used more than once.

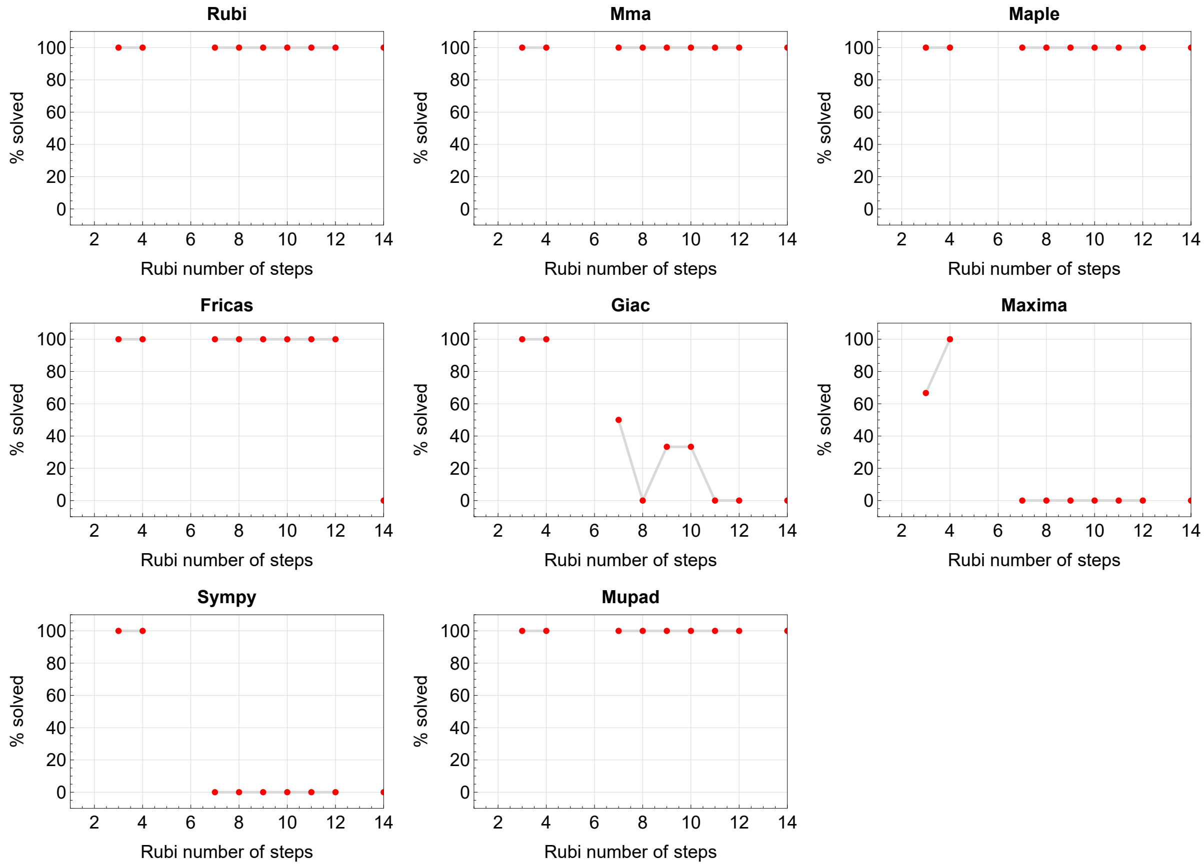


Figure 1.2: Solving statistics per number of Rubi steps used

The above diagram show that the precentage of solved intergals decreases for most CAS systems as the number of steps increases. As expected, for integrals that required less steps by Rubi, CAS systems had more success which indicates the integral was not as hard to solve. As Rubi needed more steps to solve the integral, the solved percentage decreased for most CAS systems which indicates the integral is becoming harder to solve.

1.6 Solved integrals histogram based on leaf size of result

The following shows the distribution of solved integrals for each CAS system based on leaf size of the antiderivatives produced by each CAS. It shows that most integrals solved produced leaf size less than about 100 to 150. The bin size used is 40.

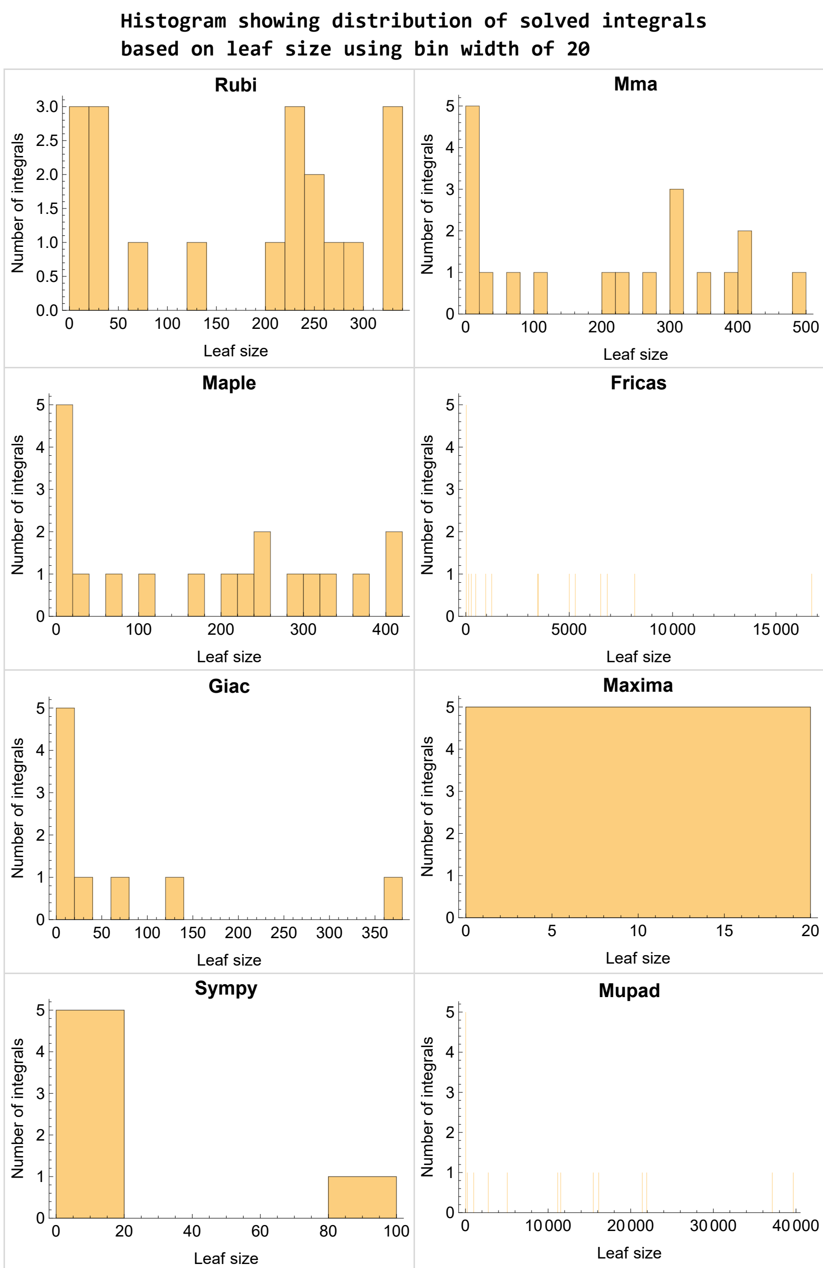


Figure 1.3: Solved integrals based on leaf size distribution

1.7 Solved integrals histogram based on CPU time used

The following shows the distribution of solved integrals for each CAS system based on CPU time used in seconds. The bin size used is 0.1 second.

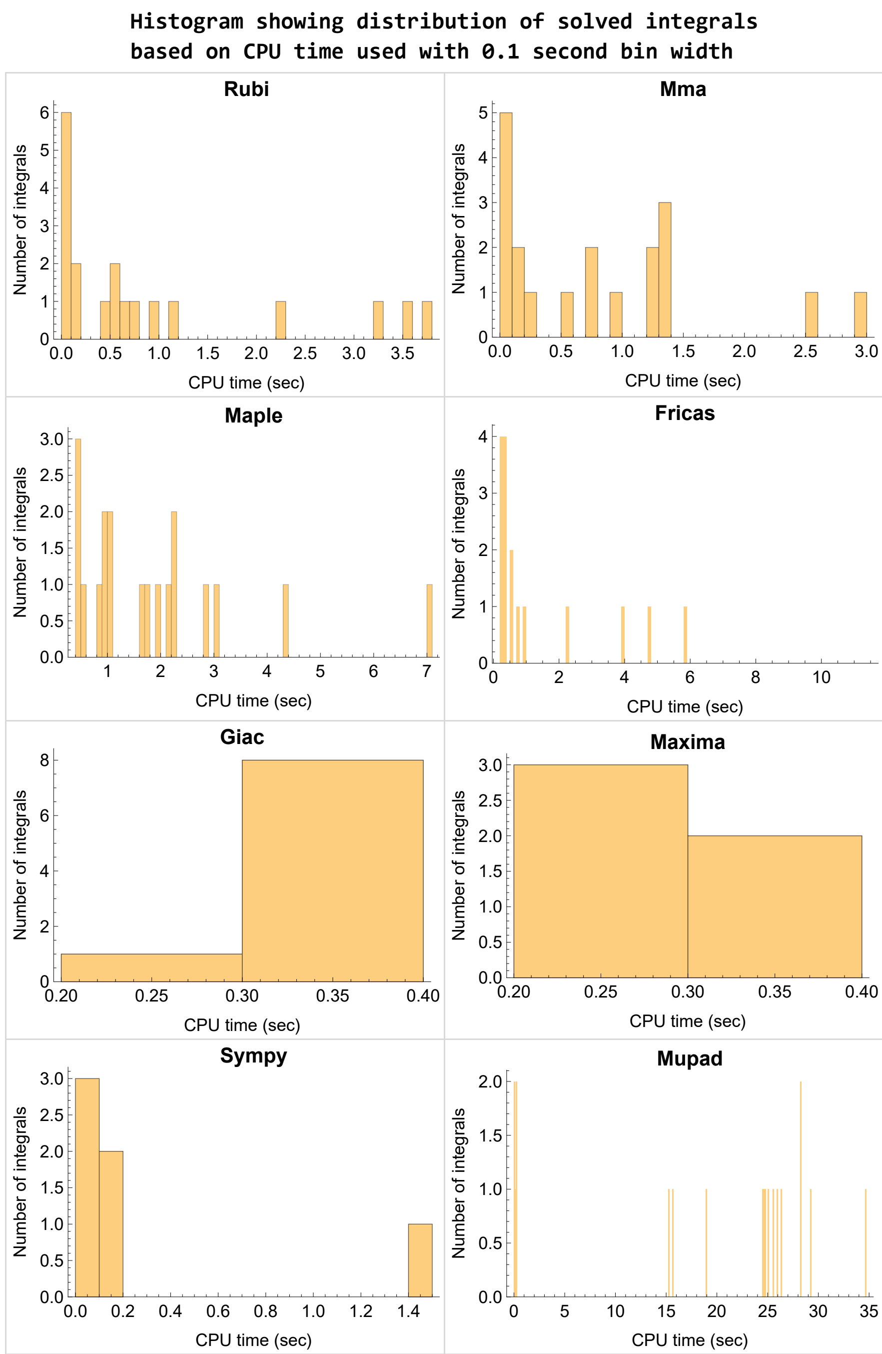


Figure 1.4: Solved integrals histogram based on CPU time used

1.8 Leaf size vs. CPU time used

The following shows the relation between the CPU time used to solve an integral and the leaf size of the antiderivative.

The result for Fricas, Maxima and Giac is shifted more to the right than the other CAS system due to the use of sagemath to call them, which causes an initial slight delay in the timing to start the integration due to overhead of starting a new process each time. This should also be taken into account when looking at the timing of these three CAS systems. Direct calls not using sagemath would result in faster timings, but current implementation uses sagemath as this makes testing much easier to do.

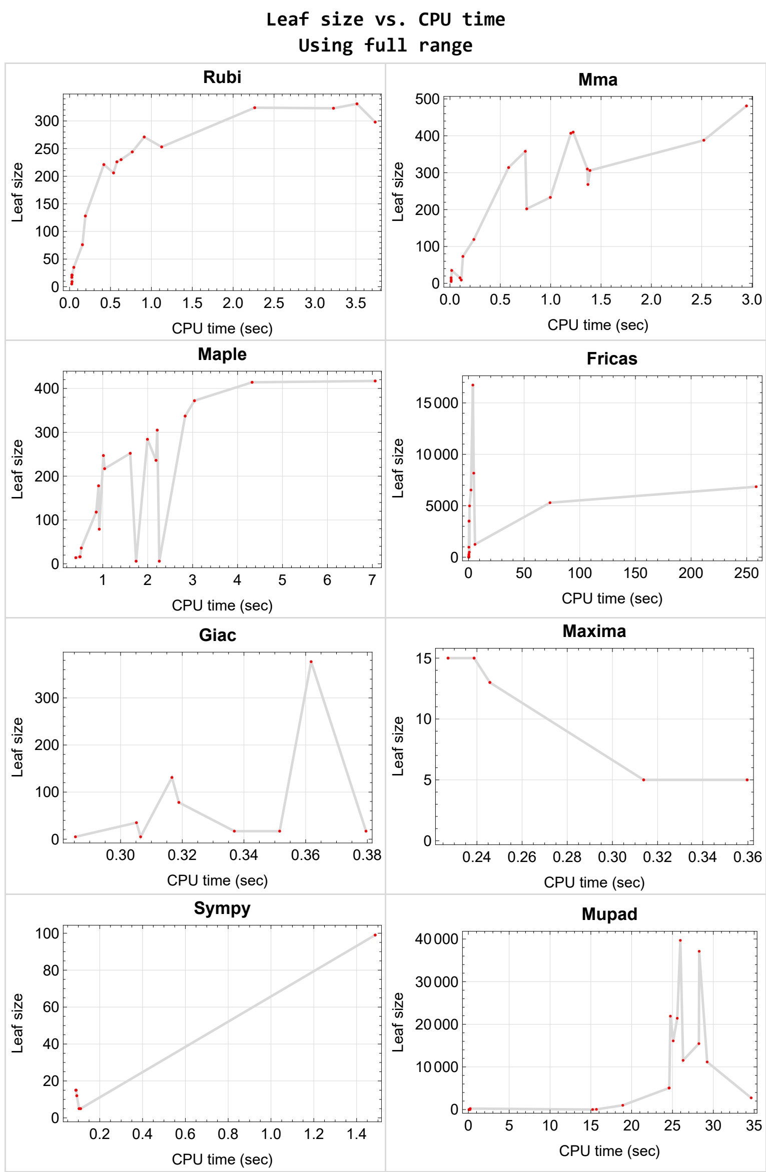


Figure 1.5: Leaf size vs. CPU time. Full range

1.9 list of integrals with no known antiderivative

{}

1.10 List of integrals solved by CAS but has no known antiderivative

Rubi {}

Mathematica {}

Maple {}

Maxima {}

Fricas {}

Sympy {}

Giac {}

Mupad {}

1.11 list of integrals solved by CAS but failed verification

The following are integrals solved by CAS but the verification phase failed to verify the anti-derivative produced is correct. This does not necessarily mean that the anti-derivative is wrong as additional methods of verification might be needed, or more time is needed (3 minutes time limit was used). These integrals are listed here to make it possible to do further investigation to determine why the result could not be verified.

Rubi {}

Mathematica {}

Maple {}

Maxima Verification phase not currently implemented.

Fricas Verification phase not currently implemented.

Sympy Verification phase not currently implemented.

Giac Verification phase not currently implemented.

Mupad Verification phase not currently implemented.

1.12 Timing

The command `AbsoluteTiming[]` was used in Mathematica to obtain the elapsed time for each integrate call. In Maple, the command `Usage` was used as in the following example

```
cpu_time := Usage(assign ('result_of_int',int(expr,x)),output='realtime'
```

For all other CAS systems, the elapsed time to complete each integral was found by taking the difference between the time after the call completed from the time before the call was made. This was done using Python's `time.time()` call.

All elapsed times shown are in seconds. A time limit of 3 CPU minutes was used for each integral. If the integrate command did not complete within this time limit, the integral was aborted and considered to have failed and assigned an F grade. The time used by failed integrals due to time out was not counted in the final statistics.

1.13 Verification

A verification phase was applied on the result of integration for `Rubi` and `Mathematica`.

Future version of this report will implement verification for the other CAS systems. For the integrals whose result was not run through a verification phase, it is assumed that the antiderivative was correct.

Verification phase also had 3 minutes time out. An integral whose result was not verified could still be correct, but further investigation is needed on those integrals. These integrals were marked in the summary table below and also in each integral separate section so they are easy to identify and locate.

1.14 Important notes about some of the results

Important note about Maxima results

Since tests were run in a batch mode, and using an automated script, then any integral where Maxima needed an interactive response from the user to answer a question during the evaluation of the integral will fail.

The exception raised is `ValueError`. Therefore Maxima results is lower than what would result if Maxima was run directly and each question was answered correctly.

The percentage of such failures were not counted for each test file, but for an example, for the `Timofeev` test file, there were about 14 such integrals out of total 705, or about 2 percent. This percentage can be higher or lower depending on the specific input test file.

Such integrals can be identified by looking at the output of the integration in each section for Maxima. The exception message will indicate the cause of error.

Maxima integrate was run using SageMath with the following settings set by default

```
'besselexpand : true'
'display2d : false'
'domain : complex'
'keepfloat : true'
'load(to_poly_solve)'
'load(simplify_sum)'
'load(abs_integrate)' 'load(diag)'
```

SageMath automatic loading of Maxima `abs_integrate` was found to cause some problems. So the following code was added to disable this effect.

```
from sage.interfaces.maxima_lib import maxima_lib
maxima_lib.set('extra_definite_integration_methods', '[]')
maxima_lib.set('extra_integration_methods', '[]')
```

See `https://ask.sagemath.org/question/43088/integrate-results-that-are-different-from-using-maxima/` for reference.

Important note about FriCAS result

There were few integrals which failed due to SageMath interface and not because FriCAS system could not do the integration.

These will fail With error `Exception raised: NotImplementedError`.

The number of such cases seems to be very small. About 1 or 2 percent of all integrals. These can be identified by looking at the exception message given in the result.

Important note about finding leaf size of antiderivative

For Mathematica, Rubi, and Maple, the builtin system function `LeafSize` was used to find the leaf size of each antiderivative.

The other CAS systems (SageMath and Sympy) do not have special builtin function for this purpose at this time. Therefore the leaf size for Fricas and Sympy antiderivative was determined using the following function, thanks to user `slelievre` at `https://ask.sagemath.org/question/57123/could-we-have-a-leaf_count-function-in-base-sagemath/`

```
def tree_size(expr):
    r"""
    Return the tree size of this expression.
    """
    if expr not in SR:
        # deal with lists, tuples, vectors
        return 1 + sum(tree_size(a) for a in expr)
    expr = SR(expr)
```

```
    x, aa = expr.operator(), expr.operands()
    if x is None:
        return 1
    else:
        return 1 + sum(tree_size(a) for a in aa)
```

For Sympy, which was called directly from Python, the following code was used to obtain the leafsize of its result

```
try:
  # 1.7 is a fudge factor since it is low side from actual leaf count
  leafCount = round(1.7*count_ops(anti))

  except Exception as ee:
        leafCount =1
```

Important note about Mupad results

Matlab's symbolic toolbox does not have a leaf count function to measure the size of the antiderivative. Maple was used to determine the leaf size of Mupad output by post processing Mupad result.

Currently no grading of the antiderivative for Mupad is implemented. If it can integrate the problem, it was assigned a B grade automatically as a placeholder. In the future, when grading function is implemented for Mupad, the tests will be rerun again.

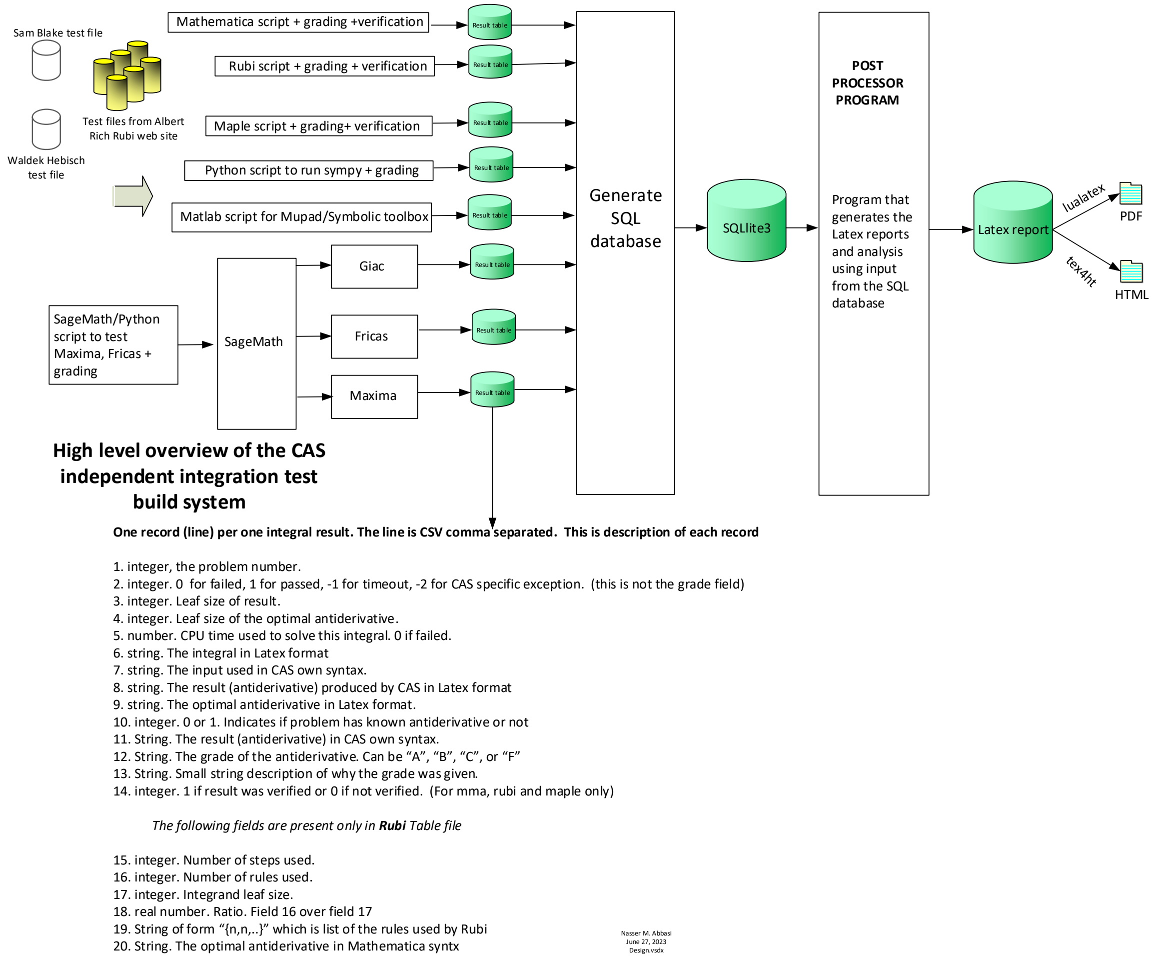
The following is an example of using Matlab's symbolic toolbox (Mupad) to solve an integral

```
integrand = evalin(symengine,'cos(x)*sin(x)')
the_variable = evalin(symengine,'x')
anti = int(integrand,the_variable)
```

Which gives `sin(x)^2/2`

1.15 Design of the test system

The following diagram gives a high level view of the current test build system.



CHAPTER 2

DETAILED SUMMARY TABLES OF RESULTS

2.1 List of integrals sorted by grade for each CAS

Rubi

A grade $\{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 \}$

B grade $\{ \}$

C grade $\{ \}$

F normal fail $\{ \}$

F(-1) timedout fail $\{ \}$

F(-2) exception fail $\{ \}$

Mma

A grade $\{ 9, 11, 12, 14, 15, 16, 17, 18, 19 \}$

B grade $\{ \}$

C grade $\{ 1, 2, 3, 4, 5, 6, 7, 8, 10, 13 \}$

F normal fail $\{ \}$

F(-1) timedout fail $\{ \}$

F(-2) exception fail $\{ \}$

Maple

A grade $\{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19 \}$

B grade $\{ \}$

C grade $\{ 10 \}$

F normal fail $\{ \}$

F(-1) timedout fail $\{ \}$

F(-2) exception fail $\{ \}$

Fricas

A grade $\{ 9, 11, 12, 15, 16, 17, 18, 19 \}$

B grade $\{ 1, 2, 3, 4, 5, 6, 7, 10, 13, 14 \}$

C grade $\{ \}$

F normal fail $\{ \}$

F(-1) timedout fail $\{ 8 \}$

F(-2) exception fail $\{ \}$

Maxima

A grade $\{ 15, 16, 17, 18, 19 \}$

B grade $\{ \}$

C grade $\{ \}$

F normal fail $\{ 1, 2, 3, 4, 5, 6, 7, 8, 10, 13 \}$

F(-1) timedout fail $\{ \}$

F(-2) exception fail $\{ 9, 11, 12, 14 \}$

Giac

A grade $\{ 9, 11, 12, 14, 15, 16, 17, 18, 19 \}$

B grade $\{ \}$

C grade $\{ \}$

F normal fail $\{ \}$

F(-1) timedout fail $\{ 1, 2, 3, 4, 5, 6, 7, 8, 10, 13 \}$

F(-2) exception fail $\{ \}$

Mupad

A grade $\{ \}$

B grade $\{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 \}$

C grade $\{ \}$

F normal fail $\{ \}$

F(-1) timedout fail $\{ \}$

F(-2) exception fail $\{ \}$

Sympy

A grade $\{ 15, 16, 17, 18, 19 \}$

B grade $\{ 11 \}$

C grade $\{ \}$

F normal fail $\{ 6, 7, 8, 12, 13, 14 \}$

F(-1) timedout fail $\{ 1, 2, 3, 4, 5, 9, 10 \}$

F(-2) exception fail $\{ \}$

2.2 Detailed conclusion table per each integral for all CAS systems

Detailed conclusion table per each integral is given by the table below. The elapsed time is in seconds. For failed result it is given as **F(-1)** if the failure was due to timeout. It is given as **F(-2)** if the failure was due to an exception being raised, which could indicate a bug in the system. If the failure was due to integral not being evaluated within the time limit, then it is given as **F**.

In this table, the column `N.S.` means **normalized size** and is defined as $\frac{\text{antiderivative leaf size}}{\text{optimal antiderivative leaf size}}$. To make the table fit the page, the name `Mathematica` was abbreviated to `MMA`.

Problem 1	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F(-1)	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	323	323	410	372	0	8169	0	0	39682
N.S.	1	1.00	1.27	1.15	0.00	25.29	0.00	0.00	122.85
time (sec)	N/A	3.227	1.223	3.041	0.000	4.768	0.000	0.000	25.962

Problem 2	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F(-1)	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	298	298	358	305	0	6531	0	0	21407
N.S.	1	1.00	1.20	1.02	0.00	21.92	0.00	0.00	71.84
time (sec)	N/A	3.736	0.748	2.213	0.000	2.255	0.000	0.000	25.584

Problem 3	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F(-1)	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	253	253	310	252	0	4985	0	0	15461
N.S.	1	1.00	1.23	1.00	0.00	19.70	0.00	0.00	61.11
time (sec)	N/A	1.126	1.364	1.612	0.000	0.981	0.000	0.000	28.232

Problem 4	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F(-1)	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	226	226	268	217	0	3519	0	0	5048
N.S.	1	1.00	1.19	0.96	0.00	15.57	0.00	0.00	22.34
time (sec)	N/A	0.579	1.370	1.042	0.000	0.572	0.000	0.000	24.568

Problem 5	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F(-1)	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	221	221	233	247	0	3495	0	0	5064
N.S.	1	1.00	1.05	1.12	0.00	15.81	0.00	0.00	22.91
time (sec)	N/A	0.419	0.997	1.015	0.000	0.508	0.000	0.000	24.608

Problem 6	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	244	244	306	284	0	5296	0	0	11540
N.S.	1	1.00	1.25	1.16	0.00	21.70	0.00	0.00	47.30
time (sec)	N/A	0.767	1.390	1.995	0.000	73.246	0.000	0.000	26.300

Problem 7	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	271	271	388	337	0	6851	0	0	16102
N.S.	1	1.00	1.43	1.24	0.00	25.28	0.00	0.00	59.42
time (sec)	N/A	0.913	2.521	2.834	0.000	258.462	0.000	0.000	25.087

Problem 8	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	F(-1)	F	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	331	331	481	414	0	0	0	0	21909
N.S.	1	1.00	1.45	1.25	0.00	0.00	0.00	0.00	66.19
time (sec)	N/A	3.514	2.945	4.324	0.000	0.000	0.000	0.000	24.743

Problem 9	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	F(-2)	A	F(-1)	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	76	76	73	79	0	276	0	78	229
N.S.	1	1.00	0.96	1.04	0.00	3.63	0.00	1.03	3.01
time (sec)	N/A	0.157	0.129	0.923	0.000	0.383	0.000	0.319	0.234

Problem 10	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	C	F	B	F(-1)	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	230	230	314	178	0	971	0	0	11164
N.S.	1	1.00	1.37	0.77	0.00	4.22	0.00	0.00	48.54
time (sec)	N/A	0.630	0.582	0.908	0.000	0.388	0.000	0.000	29.244

Problem 11	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	F(-2)	A	B	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	35	35	35	36	0	139	99	35	47
N.S.	1	1.00	1.00	1.03	0.00	3.97	2.83	1.00	1.34
time (sec)	N/A	0.052	0.017	0.523	0.000	0.348	1.487	0.305	15.685

Problem 12	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	F(-2)	A	F	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	128	128	119	118	0	482	0	131	1001
N.S.	1	1.00	0.93	0.92	0.00	3.77	0.00	1.02	7.82
time (sec)	N/A	0.193	0.237	0.855	0.000	0.789	0.000	0.317	18.912

Problem 13	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	C	A	F	B	F	F(-1)	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	324	324	407	417	0	16739	0	0	37118
N.S.	1	1.00	1.26	1.29	0.00	51.66	0.00	0.00	114.56
time (sec)	N/A	2.263	1.200	7.063	0.000	3.952	0.000	0.000	28.270

Problem 14	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	F(-2)	B	F	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	206	206	202	236	0	1244	0	377	2743
N.S.	1	1.00	0.98	1.15	0.00	6.04	0.00	1.83	13.32
time (sec)	N/A	0.538	0.762	2.186	0.000	5.874	0.000	0.362	34.636

Problem 15	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	A	A	A	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	21	21	15	16	15	17	15	17	9
N.S.	1	1.00	0.71	0.76	0.71	0.81	0.71	0.81	0.43
time (sec)	N/A	0.030	0.011	0.497	0.227	0.268	0.088	0.337	0.201

Problem 16	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	A	A	A	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	17	17	9	14	13	17	12	17	9
N.S.	1	1.00	0.53	0.82	0.76	1.00	0.71	1.00	0.53
time (sec)	N/A	0.028	0.112	0.401	0.246	0.280	0.093	0.380	15.220

Problem 17	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	A	A	A	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	21	21	15	16	15	17	15	17	9
N.S.	1	1.00	0.71	0.76	0.71	0.81	0.71	0.81	0.43
time (sec)	N/A	0.028	0.100	0.490	0.239	0.306	0.090	0.352	0.117

Problem 18	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	A	A	A	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	9	9	9	6	5	5	5	5	5
N.S.	1	1.00	1.00	0.67	0.56	0.56	0.56	0.56	0.56
time (sec)	N/A	0.030	0.012	2.259	0.314	0.284	0.110	0.306	0.083

Problem 19	Optimal	Rubi	MMA	Maple	Maxima	Fricas	Sympy	Giac	Mupad
grade	N/A	A	A	A	A	A	A	A	B
verified	N/A	Yes	Yes	Yes	TBD	TBD	TBD	TBD	TBD
size	5	5	5	6	5	5	5	5	5
N.S.	1	1.00	1.00	1.20	1.00	1.00	1.00	1.00	1.00
time (sec)	N/A	0.027	0.012	1.743	0.360	0.270	0.103	0.285	0.073

2.3 Detailed conclusion table specific for Rubi results

The following table is specific to Rubi only. It gives additional statistics for each integral. the column **steps** is the number of steps used by Rubi to obtain the antiderivative. The **rules** column is the number of unique rules used. The **integrand size** column is the leaf size of the integrand. Finally the ratio $\frac{\text{number of rules}}{\text{integrand size}}$ is also given. The larger this ratio is, the harder the integral is to solve. In this test file, problem number [8] had the largest ratio of [.4737000000000000010]

Table 2.1: Rubi specific breakdown of results for each integral

#	grade	number of steps used	number of unique rules	normalized antiderivative leaf size	integrand leaf size	$\frac{\text{number of rules}}{\text{integrand leaf size}}$
1	A	12	8	1.00	19	0.421
2	A	10	6	1.00	19	0.316
3	A	9	5	1.00	19	0.263
4	A	8	4	1.00	17	0.235
5	A	7	4	1.00	14	0.286
6	A	10	6	1.00	17	0.353
7	A	12	8	1.00	19	0.421
8	A	14	9	1.00	19	0.474
9	A	7	6	1.00	19	0.316
10	A	9	5	1.00	19	0.263
11	A	3	3	1.00	17	0.176
12	A	9	8	1.00	17	0.471
13	A	11	6	1.00	19	0.316
14	A	10	9	1.00	19	0.474
15	A	4	3	1.00	13	0.231
16	A	4	3	1.00	15	0.200
17	A	4	3	1.00	15	0.200
18	A	3	3	1.00	15	0.200
19	A	3	3	1.00	15	0.200

CHAPTER 3

LISTING OF INTEGRALS

3.1 $\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 323

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{x}{2c}+\frac{(b^2-ac)\,x}{c^3}-\frac{\sqrt{2}\left(b^3-2abc-\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c^3\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{\sqrt{2}\left(b^3-2abc+\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c^3\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} + \frac{b\cos(x)}{c^2}-\frac{\cos(x)\sin(x)}{2c}$$

```
[Out] 1/2*x/c+(-a*c+b^2)*x/c^3+b*cos(x)/c^2-1/2*cos(x)*sin(x)/c-arctan(1/2*(2*c+(
b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/
2))^(1/2))*2^(1/2)*(b^3-2*a*b*c+(-2*a^2*c^2+4*a*b^2*c-b^4)/(-4*a*c+b^2)^(1/
2))/c^3/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2)-arctan(1/2*(2*c+(b+(-4*a
*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/
2))*2^(1/2)*(b^3-2*a*b*c+(2*a^2*c^2-4*a*b^2*c+b^4)/(-4*a*c+b^2)^(1/2))/c^3/
(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2)
```

Rubi [A] (verified)

Time = 3.23 (sec) , antiderivative size = 323, normalized size of antiderivative = 1.00, number of steps used = 12, number of rules used = 8, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.421, Rules used = {3337, 2718, 2715, 8, 3373, 2739, 632, 210}

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$
$$= -\frac{\sqrt{2}\left(-\frac{2a^2c^2-4ab^2c+b^4}{\sqrt{b^2-4ac}}-2abc+b^3\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c^3\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{\sqrt{2}\left(\frac{2a^2c^2-4ab^2c+b^4}{\sqrt{b^2-4ac}}-2abc+b^3\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c^3\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{x(b^2-ac)}{c^3}+\frac{b\cos(x)}{c^2}+\frac{x}{2c}-\frac{\sin(x)\cos(x)}{2c}$$

```
[In] Int[Sin[x]^4/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] x/(2*c) + ((b^2 - a*c)*x)/c^3 - (Sqrt[2]*(b^3 - 2*a*b*c - (b^4 - 4*a*b^2*c
+ 2*a^2*c^2)/Sqrt[b^2 - 4*a*c])*ArcTan[(2*c + (b - Sqrt[b^2 - 4*a*c])*Tan[x
/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]])])/(c^3*Sqrt[b^
2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]]) - (Sqrt[2]*(b^3 - 2*a*b*c + (b^4 -
4*a*b^2*c + 2*a^2*c^2)/Sqrt[b^2 - 4*a*c])*ArcTan[(2*c + (b + Sqrt[b^2 - 4*a
*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])])/(c
^3*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]]) + (b*Cos[x])/c^2 - (Cos[x
]*Sin[x])/(2*c)
```

Rule 8

```
Int[a_, x_Symbol] :> Simp[a*x, x] /; FreeQ[a, x]
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2715

```
Int[((b_.)*sin[(c_.) + (d_.)*(x_)])^(n_), x_Symbol] :> Simp[(-b)*Cos[c + d*
x]*((b*Sin[c + d*x])^(n - 1)/(d*n)), x] + Dist[b^2*((n - 1)/n), Int[(b*Sin[
c + d*x])^(n - 2), x], x] /; FreeQ[{b, c, d}, x] && GtQ[n, 1] && IntegerQ[2
*n]
```

Rule 2718

```
Int[sin[(c_.) + (d_.)*(x_)], x_Symbol] :> Simp[-Cos[c + d*x]/d, x] /; FreeQ
[{c, d}, x]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3337

```
Int[sin[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_), x_Symbol] :> Int[ExpandTr
ig[sin[d + e*x]^m*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n))^p, x], x] /
; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && NeQ[b^2 - 4*a*c, 0] && Integ
ersQ[m, n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rubi steps

$$\text{integral} = \int \left(\frac{b^2-ac}{c^3}-\frac{b\sin(x)}{c^2}+\frac{\sin^2(x)}{c}+\frac{-ab^2\left(1-\frac{ac}{b^2}\right)-b^3\left(1-\frac{2ac}{b^2}\right)\sin(x)}{c^3\left(a+b\sin(x)+c\sin^2(x)\right)}\right)dx$$

$$=\frac{\left(b^2-ac\right)x}{c^3}+\frac{\int\frac{-ab^2\left(1-\frac{ac}{b^2}\right)-b^3\left(1-\frac{2ac}{b^2}\right)\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{c^3}-\frac{b\int\sin(x)\,dx}{c^2}+\frac{\int\sin^2(x)\,dx}{c}$$

$$= \frac{(b^2-ac)\,x}{c^3} + \frac{b\cos(x)}{c^2} - \frac{\cos(x)\sin(x)}{2c} + \frac{\int 1\,dx}{2c}$$
$$- \frac{\left(b^3-2abc-\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c^3}$$
$$- \frac{\left(b^3-2abc+\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c^3}$$

$$= \frac{x}{2c} + \frac{(b^2-ac)\,x}{c^3} + \frac{b\cos(x)}{c^2} - \frac{\cos(x)\sin(x)}{2c}$$
$$- \frac{\left(2\left(b^3-2abc-\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{c^3}$$
$$- \frac{\left(2\left(b^3-2abc+\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{c^3}$$

$$= \frac{x}{2c} + \frac{(b^2-ac)\,x}{c^3} + \frac{b\cos(x)}{c^2} - \frac{\cos(x)\sin(x)}{2c}$$
$$+ \frac{\left(4\left(b^3-2abc-\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx, x, 4c+2\left(b-\sqrt{b^2-4ac}\right.\right.}{c^3}$$
$$+ \frac{\left(4\left(b^3-2abc+\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx, x, 4c+2\left(b+\sqrt{b^2-4ac}\right)\text{t}\right.}{c^3}$$

$$= \frac{x}{2c} + \frac{(b^2-ac)\,x}{c^3} - \frac{\sqrt{2}\left(b^3-2abc-\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c^3\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}$$
$$- \frac{\sqrt{2}\left(b^3-2abc+\frac{b^4-4ab^2c+2a^2c^2}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c^3\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$
$$+ \frac{b\cos(x)}{c^2} - \frac{\cos(x)\sin(x)}{2c}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 1.22 (sec) , antiderivative size = 410, normalized size of antiderivative = 1.27

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$
$$= \frac{4b^2x+2c(-2a+c)x-\frac{4\left(ib^4-4iab^2c+2ia^2c^2+b^3\sqrt{-b^2+4ac}-2abc\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}-\frac{4\left(-ib^4+4ia\right.}{}}{4c^3}$$

```
[In] Integrate[Sin[x]^4/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (4*b^2*x + 2*c*(-2*a + c)*x - (4*(I*b^4 - (4*I)*a*b^2*c + (2*I)*a^2*c^2 + b
^3*Sqrt[-b^2 + 4*a*c] - 2*a*b*c*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b - I*Sq
rt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2
 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^
2 + 4*a*c]]) - (4*((-I)*b^4 + (4*I)*a*b^2*c - (2*I)*a^2*c^2 + b^3*Sqrt[-b^2
 + 4*a*c] - 2*a*b*c*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b + I*Sqrt[-b^2 + 4*
a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]])]
)/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]])
 + 4*b*c*Cos[x] - c^2*Sin[2*x])/(4*c^3)
```

Maple [A] (verified)

Time = 3.04 (sec) , antiderivative size = 372, normalized size of antiderivative = 1.15

method	result
default	$\frac{2a\left(-\frac{2\left(-3\sqrt{-4ac+b^2}\,abc+\sqrt{-4ac+b^2}\,b^3+4a^2c^2-5a\,b^2c+b^4\right)\arctan\left(\frac{-2a\tan\left(\frac{x}{2}\right)+\sqrt{-4ac+b^2}-b}{\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}\right)}{\left(8ac-2b^2\right)\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}+\frac{2\left(3\sqrt{-4ac+b^2}\,abc-\sqrt{-4ac+b^2}\,b^3+4a^2\right.}{\left(8ac-2b^2\right)}\right.}{c^3}$
risch	Expression too large to display

```
[In] int(sin(x)^4/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 2/c^3*a*(-2*(-3*(-4*a*c+b^2)^(1/2)*a*b*c+(-4*a*c+b^2)^(1/2)*b^3+4*a^2*c^2-5
*a*b^2*c+b^4)/(8*a*c-2*b^2)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2
)*arctan((-2*a*tan(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4*a*c-2*b^2+2*b*(-4*a*c+b^
2)^(1/2)+4*a^2)^(1/2))+2*(3*(-4*a*c+b^2)^(1/2)*a*b*c-(-4*a*c+b^2)^(1/2)*b^3
+4*a^2*c^2-5*a*b^2*c+b^4)/(8*a*c-2*b^2)/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)
+4*a^2)^(1/2)*arctan((2*a*tan(1/2*x)+b+(-4*a*c+b^2)^(1/2))/(4*a*c-2*b^2-2*b
*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)))-2/c^3*((-1/2*c^2*tan(1/2*x)^3-b*c*tan(1/
```

```
2*x)^2+1/2*c^2*tan(1/2*x)-b*c)/(1+tan(1/2*x)^2)^2+1/2*(2*a*c-2*b^2-c^2)*arc
tan(tan(1/2*x)))
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 8169 vs. 2(285) = 570.

Time = 4.77 (sec) , antiderivative size = 8169, normalized size of antiderivative = 25.29

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(sin(x)^4/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
[Out] Too large to include
```

Sympy [F(-1)]

Timed out.

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(sin(x)**4/(a+b*sin(x)+c*sin(x)**2),x)
[Out] Timed out
```

Maxima [F]

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\sin(x)^4}{c\sin(x)^2+b\sin(x)+a}\,dx$$

```
[In] integrate(sin(x)^4/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
[Out] 1/4*(4*c^3*integrate(-2*(2*(b^4 - 2*a*b^2*c)*cos(3*x)^2 + 4*(2*a^2*b^2 - a^
2*c^2 - (2*a^3 - a*b^2)*c)*cos(2*x)^2 + 2*(b^4 - 2*a*b^2*c)*cos(x)^2 + 2*(b
^4 - 2*a*b^2*c)*sin(3*x)^2 + 2*(4*a*b^3 - 2*a*b*c^2 - (6*a^2*b - b^3)*c)*co
s(x)*sin(2*x) + 4*(2*a^2*b^2 - a^2*c^2 - (2*a^3 - a*b^2)*c)*sin(2*x)^2 + 2*
(b^4 - 2*a*b^2*c)*sin(x)^2 - (2*(a*b^2*c - a^2*c^2)*cos(2*x) + (b^3*c - 2*a
*b*c^2)*sin(3*x) - (b^3*c - 2*a*b*c^2)*sin(x))*cos(4*x) - 2*(2*(b^4 - 2*a*b
^2*c)*cos(x) + (4*a*b^3 - 2*a*b*c^2 - (6*a^2*b - b^3)*c)*sin(2*x))*cos(3*x)
 - 2*(a*b^2*c - a^2*c^2 + (4*a*b^3 - 2*a*b*c^2 - (6*a^2*b - b^3)*c)*sin(x))
*cos(2*x) + ((b^3*c - 2*a*b*c^2)*cos(3*x) - (b^3*c - 2*a*b*c^2)*cos(x) - 2*
(a*b^2*c - a^2*c^2)*sin(2*x))*sin(4*x) - (b^3*c - 2*a*b*c^2 - 2*(4*a*b^3 -
2*a*b*c^2 - (6*a^2*b - b^3)*c)*cos(2*x) + 4*(b^4 - 2*a*b^2*c)*sin(x))*sin(3
```

```
*x) + (b^3*c - 2*a*b*c^2)*sin(x))/(c^5*cos(4*x)^2 + 4*b^2*c^3*cos(3*x)^2 +
4*b^2*c^3*cos(x)^2 + c^5*sin(4*x)^2 + 4*b^2*c^3*sin(3*x)^2 + 4*b^2*c^3*sin(
x)^2 + 4*b*c^4*sin(x) + c^5 + 4*(4*a^2*c^3 + 4*a*c^4 + c^5)*cos(2*x)^2 + 8*
(2*a*b*c^3 + b*c^4)*cos(x)*sin(2*x) + 4*(4*a^2*c^3 + 4*a*c^4 + c^5)*sin(2*x
)^2 - 2*(2*b*c^4*sin(3*x) - 2*b*c^4*sin(x) - c^5 + 2*(2*a*c^4 + c^5)*cos(2*
x))*cos(4*x) - 8*(b^2*c^3*cos(x) + (2*a*b*c^3 + b*c^4)*sin(2*x))*cos(3*x) -
 4*(2*a*c^4 + c^5 + 2*(2*a*b*c^3 + b*c^4)*sin(x))*cos(2*x) + 4*(b*c^4*cos(3
*x) - b*c^4*cos(x) - (2*a*c^4 + c^5)*sin(2*x))*sin(4*x) - 4*(2*b^2*c^3*sin(
x) + b*c^4 - 2*(2*a*b*c^3 + b*c^4)*cos(2*x))*sin(3*x)), x) + 4*b*c*cos(x) -
 c^2*sin(2*x) + 2*(2*b^2 - 2*a*c + c^2)*x)/c^3
```

Giac [F(-1)]

Timed out.

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx=\text{Timed out}$$

```
[In] integrate(sin(x)^4/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 25.96 (sec) , antiderivative size = 39682, normalized size of antiderivative = 122.85

$$\int \frac{\sin^4(x)}{a+b\sin(x)+c\sin^2(x)}\,dx=\text{Too large to display}$$

```
[In] int(sin(x)^4/(a + c*sin(x)^2 + b*sin(x)),x)
```

```
[Out] ((2*b)/c^2 - tan(x/2)/c + tan(x/2)^3/c + (2*b*tan(x/2)^2)/c^2)/(2*tan(x/2)^
2 + tan(x/2)^4 + 1) - atan(((((2048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 -
 64*a^7*c^7 + 12*a^8*c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*
a^2*b^4*c^8 - 119*a^2*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b
^2*c^9 + 290*a^3*b^4*c^7 + 397*a^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c
^8 - 491*a^4*b^4*c^6 - 148*a^4*b^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 +
102*a^5*b^4*c^5 - 60*a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a
^7*b^2*c^5))/c^8 - (-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c
 - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2
*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^
3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*
c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(
-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^
10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c
^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2)*((2048*(4*a*b^3*
```

c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2*b*c^12 + 44*a^3*b
*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a^2*b^3*c^10 - 16*a
^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c^7 + 12*a^3*b^7*c^
5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^8 - (((2048*(12*a*
b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13 + 48*a^4*b*c^12 -
68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*(256*a^2*c^15 + 57
6*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 + 68*a*b^4*c^12 - 8
*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^3*b^2*c^12 + 8*a^3
*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^
4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3
)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2
 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a
^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) -
10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2
))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*
c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2)
- (2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a*b^4*c^
11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8 - 176
*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44*a^5*b
^2*c^9))/c^8 + (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 + 256*a^3*b*c^1
2 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a^2*b^5*c^9 -
336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8)*(-(a^2*b^8 - b^10
 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^
2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b
^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*
c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4
*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*
(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c
^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 -
8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13 + 184*a
^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 + 48*a*b^
4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11 + 56*a^
2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 1817*a^3*
b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*a^4*b^4
*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6*b^2*c^
7))/c^8) + (2048*tan(x/2)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a
*b^11*c^2 + 64*a^3*b*c^10 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 +
112*a^7*b*c^6 - 48*a^2*b^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^
2*b^9*c^3 + 364*a^3*b^3*c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^
9*c^2 - 72*a^4*b^3*c^7 + 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c
^6 + 336*a^5*b^5*c^4 - 352*a^6*b^3*c^5))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5
 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a
*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*
a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^
(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^

```
3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b
^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6
 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^
7)))^(1/2) + (2048*(16*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7
 + 84*a^5*b^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^
4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 33
3*a^4*b^5*c^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a
^6*b^3*c^4 - 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3))/c^8 + (2048*tan(x/2)*(32*a
*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*
a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a
^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^
6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5
 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2
- 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32
*a^8*b^2*c^3))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*
c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^
2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c
^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a
*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*
(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c
^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*
c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2)*1i + ((2048*(16
*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*a
^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 - 2
8*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a^
4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*b
^5*c^2 + 180*a^7*b^3*c^3))/c^8 - ((2048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c
^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 -
 28*a^2*b^4*c^8 - 119*a^2*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a
^3*b^2*c^9 + 290*a^3*b^4*c^7 + 397*a^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b
^2*c^8 - 491*a^4*b^4*c^6 - 148*a^4*b^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^
7 + 102*a^5*b^4*c^5 - 60*a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 1
00*a^7*b^2*c^5))/c^8 + (-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4
*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52
*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^
2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(
4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c
^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^
2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b
^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2)*((2048*(4*a*
b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2*b*c^12 + 44*a
^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a^2*b^3*c^10 -
16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c^7 + 12*a^3*b^
7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^8 - ((2048*(32
*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a*b^4*c^11 - 14*a*b
```

```
^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8 - 176*a^3*b^2*c^
11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44*a^5*b^2*c^9))/c^
8 + ((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13 +
 48*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*(
256*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 +
68*a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^3
*b^2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*(-(a^2*b^8 - b^10 + 8*a
^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*
(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4
 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^
2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c -
 b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a
*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b
^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*
b^2*c^7)))^(1/2) - (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 + 256*a^3*b
*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a^2*b^5*c^
9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8)*(-(a^2*b^8 -
b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c
+ a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a
^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(
4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*
(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^
5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b
^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^
6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13 + 1
84*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 + 48*
a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11 + 5
6*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 1817*
a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*a^4
*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6*b^
2*c^7))/c^8) + (2048*tan(x/2)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 -
16*a*b^11*c^2 + 64*a^3*b*c^10 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^
7 + 112*a^7*b*c^6 - 48*a^2*b^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 16
0*a^2*b^9*c^3 + 364*a^3*b^3*c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^
3*b^9*c^2 - 72*a^4*b^3*c^7 + 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b
^3*c^6 + 336*a^5*b^5*c^4 - 352*a^6*b^3*c^5))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5
*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-
(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 +
 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)
^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b
^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c
 - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6
*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^
2*c^7)))^(1/2) + (2048*tan(x/2)*(32*a*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a
^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 3
```

```
2*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c
^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2
 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6
+ 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 -
392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3))/c^8)*(-(a^2*b^8 - b^10
+ 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2
*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^
2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c
 - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*
a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(
-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^
8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8
*a^3*b^2*c^7)))^(1/2)*1i)/((4096*(16*a^6*b^6 - 4*a^8*b^4 - 4*a^7*c^5 + 15*a
^8*c^4 - 14*a^9*c^3 - 48*a^7*b^4*c + 4*a^9*b^2*c + 4*a^6*b^2*c^4 + 16*a^6*b
^4*c^2 - 32*a^7*b^2*c^3 + 44*a^8*b^2*c^2))/c^8 + (((2048*(44*a^5*c^9 - 16*a
^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5
+ 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a^
2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*c^7 + 397*a^3*b^6*c^5 + 62*a^3*b^
8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6 - 148*a^4*b^6*c^4 + 8*a^4*b^8*c^2
 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 13
6*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 - (-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a
^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b
^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^
4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2)
- 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/
2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)
^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a
*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(
1/2)*((2048*(4*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*
a^2*b*c^12 + 44*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 6
3*a^2*b^3*c^10 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^
5*c^7 + 12*a^3*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))
/c^8 - (((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^
13 + 48*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/
2)*(256*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^1
4 + 68*a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264
*a^3*b^2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*(-(a^2*b^8 - b^10 +
 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*
b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2
*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c
- b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a
*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-
(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8
 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*
a^3*b^2*c^7)))^(1/2) - (2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*
```

a^6*c^10 + 2*a*b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8 - 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44*a^5*b^2*c^9))/c^8 + (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 + 256*a^3*b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a^2*b^5*c^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13 + 184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 + 48*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11 + 56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 1817*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*a^4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6*b^2*c^7))/c^8) + (2048*tan(x/2)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^10 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 + 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 352*a^6*b^3*c^5))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*(16*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3))/c^8 + (2048*tan(x/2)*(32*a*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b

^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2)
 - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1
/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3
)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*
a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^
(1/2) - ((2048*(16*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7 + 8
4*a^5*b^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^4 +
16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 333*a^
4*b^5*c^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a^6*b
^3*c^4 - 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3))/c^8 - ((2048*(44*a^5*c^9 - 16*
a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5
 + 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a
^2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*c^7 + 397*a^3*b^6*c^5 + 62*a^3*b
^8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6 - 148*a^4*b^6*c^4 + 8*a^4*b^8*c^
2 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 1
36*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 + (-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*
a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c -
b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b
^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2)
 - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1
/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3
)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*
a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^
(1/2)*((2048*(4*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16
*a^2*b*c^12 + 44*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 -
63*a^2*b^3*c^10 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b
^5*c^7 + 12*a^3*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7)
)/c^8 - ((2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a
*b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^
8 - 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 4
4*a^5*b^2*c^9))/c^8 + ((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14
 + 80*a^3*b*c^13 + 48*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8
+ (2048*tan(x/2)*(256*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12
- 64*a*b^2*c^14 + 68*a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*
b^4*c^11 - 264*a^3*b^2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*(-(a^
2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^
3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^
3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b
c^3(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^
4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2)
+ 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4
*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^
2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) - (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b
^7*c^8 + 256*a^3*b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^1
1 + 128*a^2*b^5*c^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/

c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13 + 184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 + 48*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11 + 56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 1817*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*a^4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6*b^2*c^7))/c^8) + (2048*tan(x/2)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^10 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 + 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 352*a^6*b^3*c^5))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(32*a*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3))/c^8)*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 52*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (4096*tan(x/2)*(32*a^5*b^7 - 16*a^7*b^5 - 16*a^6*b*c^5 - 128*a^6*b^5*c + 60*a^7*b*c^4 - 48*a^8*b*c^3 + 32*a^8*b^3*c - 16*a^9*b*c^2 + 8*a^5*b^3*c^4 + 32*a^5*b^5*c^2 - 96*a^6*b^3*c^3 + 144*a^7*b^3*c^2))/c^8))*(-(a^2*b^8 - b^10 + 8*a^5*c^5 + 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) - 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) - 5

2*a^2*b^6*c^2 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + 33*a^4*b^4*c^2 - 38*a^5*b
^2*c^3 + 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-
(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*
c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a
^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*
b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2)*2i - atan((
(((2048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4
*a*b^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2*b^6*c^
6 - 128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*c^7 + 3
97*a^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6 - 148*a
^4*b^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*a^5*b^6
*c^3 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 - ((2048*(4
*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2*b*c^12 + 4
4*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a^2*b^3*c^10
 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c^7 + 12*a^3
*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^8 - (((2048
*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13 + 48*a^4*b*
c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*(256*a^2*c^
15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 + 68*a*b^4*c
^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^3*b^2*c^12
+ 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*
a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c -
b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b
^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2)
 - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1
/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3
)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*
a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^
(1/2) - (2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a*
b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8
 - 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44
*a^5*b^2*c^9))/c^8 + (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 + 256*a^3
*b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a^2*b^5*
c^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8)*((b^10 - a^
2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c
 + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*
a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-
(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2
*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b
^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 +
b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c
^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13 +
184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 + 48
*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11 +
56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 1817

*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*a^
4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6*b
^2*c^7))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2
)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c
^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12
*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^
2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*
c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 3
2*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 3
2*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(8*a*
b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^10 - 176
*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b^3*c^9
- 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*c^8 + 2
12*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 + 1008*a
^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 352*a^6*
b^3*c^5))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^
2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*
c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 1
2*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b
^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a
*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 +
32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 -
32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*(16*a^2*b^11
- 12*a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*a^6*b*c^6 -
52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*
c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a^4*b^7*c^2
- 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*b^5*c^2 + 1
80*a^7*b^3*c^3))/c^8 + (2048*tan(x/2)*(32*a*b^12 - 32*a^3*b^10 + 4*a^5*b^8
+ 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c
^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 24*a^6*b^6*c - 64*a^2
*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*b^6*c^4 + 1152*a^3*b
^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*b^6*c^3 + 504*a^5*b^
2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^2*c^5 + 812*a^6*b^4*
c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3))/c^8)*((b^10 - a^2
*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c
+ a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a
^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(
4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*
(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^
5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b
^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^
6 - 8*a^3*b^2*c^7)))^(1/2)*1i + ((2048*(16*a^2*b^11 - 12*a^4*b^9 - 144*a^3*
b^9*c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*
b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 +
56*a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600

```
*a^5*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3))/c^8 -
((2048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4*
a*b^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2*b^6*c^6
 - 128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*c^7 + 39
7*a^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6 - 148*a^
4*b^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*a^5*b^6*
c^3 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 + ((2048*(4*
a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2*b*c^12 + 44
*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a^2*b^3*c^10
- 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c^7 + 12*a^3*
b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^8 - (((2048*
(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13 + 48*a^4*b*c
^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*(256*a^2*c^1
5 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 + 68*a*b^4*c^
12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^3*b^2*c^12 +
 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a
^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b
^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^
4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2)
- 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/
2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)
^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a
*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(
1/2) + (2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a*b
^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8
- 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44*
a^5*b^2*c^9))/c^8 - (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 + 256*a^3*
b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a^2*b^5*c
^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8)*((b^10 - a^2
*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c
+ a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a
^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(
4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*
(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^
5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b
^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^
6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13 + 1
84*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 + 48*
a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11 + 5
6*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 1817*
a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*a^4
*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6*b^
2*c^7))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)
^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^
2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*
```

```
a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2
)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c
 - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32
*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32
*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(8*a*b
^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^10 - 176*
a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b^3*c^9 -
 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*c^8 + 21
2*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 + 1008*a^
4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 352*a^6*b
^3*c^5))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2
)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c
^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12
*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^
2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*
c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 3
2*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 3
2*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(32*a
*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*
a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a
^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^
6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5
 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2
- 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32
*a^8*b^2*c^3))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c
 - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2
*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^
3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*
c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(
-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^
10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c
^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2)*1i)/((4096*(16*a
^6*b^6 - 4*a^8*b^4 - 4*a^7*c^5 + 15*a^8*c^4 - 14*a^9*c^3 - 48*a^7*b^4*c + 4
*a^9*b^2*c + 4*a^6*b^2*c^4 + 16*a^6*b^4*c^2 - 32*a^7*b^2*c^3 + 44*a^8*b^2*c
^2))/c^8 + (((2048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*
a^8*c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119
*a^2*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*
b^4*c^7 + 397*a^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*
c^6 - 148*a^4*b^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 -
 60*a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8
- ((2048*(4*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2
*b*c^12 + 44*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a
^2*b^3*c^10 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c
^7 + 12*a^3*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^
8 - (((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13
```

```
+ 48*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*
(256*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 +
 68*a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^
3*b^2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*((b^10 - a^2*b^8 - 8*a
^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*
(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4
 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^
2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c -
 b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a
*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b
^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*
b^2*c^7)))^(1/2) - (2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*
c^10 + 2*a*b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*
a^2*b^6*c^8 - 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b
^4*c^8 + 44*a^5*b^2*c^9))/c^8 + (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^
8 + 256*a^3*b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 1
28*a^2*b^5*c^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8)*
((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 1
0*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^
4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a
^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) +
3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1
/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16
*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8
+ a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*
a^2*c^13 + 184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^
2*c^12 + 48*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*
b^2*c^11 + 56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*
c^10 - 1817*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^
9 - 1140*a^4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6
+ 256*a^6*b^2*c^7))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(
4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 5
2*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b
^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-
(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*
c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a
^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*
b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan
(x/2)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b
*c^10 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a
^2*b^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*
b^3*c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c
^7 + 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4
 - 352*a^6*b^3*c^5))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-
(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) +
```

```
52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*
b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(
-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3
*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*
a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a
*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*(1
6*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*
a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 -
28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a
^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*
b^5*c^2 + 180*a^7*b^3*c^3))/c^8 + (2048*tan(x/2)*(32*a*b^12 - 32*a^3*b^10 +
 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12*a^9*c^4
+ 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 24*a^6*b^6
*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*b^6*c^4 +
 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*b^6*c^3 +
 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^2*c^5 + 8
12*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3))/c^8)*(
(b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10
*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4
*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^
3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3
*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/
2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*
a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 +
 a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) - ((2048*(16*a^2*b^11 - 12*a^4*b^9 -
144*a^3*b^9*c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 -
 60*a^8*b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b
^7*c^3 + 56*a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c
^5 - 600*a^5*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3)
)/c^8 - ((2048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*
c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2
*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*
c^7 + 397*a^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6
- 148*a^4*b^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*
a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 + ((
2048*(4*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2*b*c
^12 + 44*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a^2*b
^3*c^10 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c^7 +
 12*a^3*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^8 -
(((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13 + 48
*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*(256
*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 + 68*
a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^3*b^
2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c
^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4
```

```
*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 3
3*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3
)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2
)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c -
 b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c
^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*
c^7)))^(1/2) + (2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10
 + 2*a*b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*
b^6*c^8 - 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c
^8 + 44*a^5*b^2*c^9))/c^8 - (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 +
256*a^3*b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a
^2*b^5*c^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8)*((b^
10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^
3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^
3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b
*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^
4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2)
+ 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4
*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^
2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*
c^13 + 184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^
12 + 48*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*
c^11 + 56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10
 - 1817*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 -
1140*a^4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 25
6*a^6*b^2*c^7))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a*
c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a^
2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*c
^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*a
*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*
(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c
^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4*
c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/2
)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^1
0 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b
^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*
c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 +
 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 3
52*a^6*b^3*c^5))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*(-(4*a
*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2) + 52*a
^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^5*b^2*
c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c*(-(4*
a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2
*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*
c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10*a*b^4
```

*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (2048*tan(x/
2)*(32*a*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c
^6 + 56*a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c
 + 256*a^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^
3*b^4*c^6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4
*b^4*c^5 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*
b^6*c^2 - 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*
c^2 - 32*a^8*b^2*c^3))/c^8)*((b^10 - a^2*b^8 - 8*a^5*c^5 - 8*a^6*c^4 - b^7*
(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-(4*a*c - b^2)^3)^(1/2)
+ 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 - 33*a^4*b^4*c^2 + 38*a^
5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a^3*b^3*c
*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b
^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(1
6*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6*c^6 - 8*a*b^2*c^9 + 10
*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^2*c^7)))^(1/2) + (4096*
tan(x/2)*(32*a^5*b^7 - 16*a^7*b^5 - 16*a^6*b*c^5 - 128*a^6*b^5*c + 60*a^7*b
*c^4 - 48*a^8*b*c^3 + 32*a^8*b^3*c - 16*a^9*b*c^2 + 8*a^5*b^3*c^4 + 32*a^5*
b^5*c^2 - 96*a^6*b^3*c^3 + 144*a^7*b^3*c^2))/c^8))*((b^10 - a^2*b^8 - 8*a^5
*c^5 - 8*a^6*c^4 - b^7*(-(4*a*c - b^2)^3)^(1/2) + 10*a^3*b^6*c + a^2*b^5*(-
(4*a*c - b^2)^3)^(1/2) + 52*a^2*b^6*c^2 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 -
 33*a^4*b^4*c^2 + 38*a^5*b^2*c^3 - 12*a*b^8*c + 4*a^3*b*c^3*(-(4*a*c - b^2)
^3)^(1/2) - 4*a^3*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 3*a^4*b*c^2*(-(4*a*c - b
^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c
 - b^2)^3)^(1/2))/(2*(16*a^2*c^10 + 32*a^3*c^9 + 16*a^4*c^8 + b^4*c^8 - b^6
*c^6 - 8*a*b^2*c^9 + 10*a*b^4*c^7 - 32*a^2*b^2*c^8 + a^2*b^4*c^6 - 8*a^3*b^
2*c^7)))^(1/2)*2i - (atan(((((2048*(16*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*
c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^
4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*
a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5
*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3))/c^8 + (((2
048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4*a*b
^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2*b^6*c^6 -
128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*c^7 + 397*a
^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6 - 148*a^4*b
^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*a^5*b^6*c^3
 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 + (2048*tan(x/2
)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^1
0 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b
^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*
c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 +
 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 3
52*a^6*b^3*c^5))/c^8 - (((2048*(4*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 -
 12*a*b^9*c^5 - 16*a^2*b*c^12 + 44*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9
 + 12*a^6*b*c^8 - 63*a^2*b^3*c^10 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b
^3*c^9 - 104*a^3*b^5*c^7 + 12*a^3*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6

```
 + 48*a^5*b^3*c^7))/c^8 - (((2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 +
256*a^3*b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a
^2*b^5*c^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8 - (20
48*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a*b^4*c^11 -
14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8 - 176*a^3*
b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44*a^5*b^2*c^
9))/c^8 + (((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b
*c^13 + 48*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan
(x/2)*(256*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*
c^14 + 68*a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 -
264*a^3*b^2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*(b^2*2i - a*c*2i
 + c^2*1i))/(2*c^3))*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3) + (2048*tan(x/2)*(
128*a^3*c^12 - 64*a^2*c^13 + 184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72
*a^7*c^8 + 16*a*b^2*c^12 + 48*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b
^10*c^4 - 224*a^2*b^2*c^11 + 56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*
c^5 - 286*a^3*b^2*c^10 - 1817*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4
 + 1502*a^4*b^2*c^9 - 1140*a^4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8
- 220*a^5*b^4*c^6 + 256*a^6*b^2*c^7))/c^8)*(b^2*2i - a*c*2i + c^2*1i))/(2*c
^3))*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3) + (2048*tan(x/2)*(32*a*b^12 - 32*a
^3*b^10 + 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12
*a^9*c^4 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 2
4*a^6*b^6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*
b^6*c^4 + 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*
b^6*c^3 + 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^
2*c^5 + 812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3
))/c^8)*(b^2*2i - a*c*2i + c^2*1i)*1i)/(2*c^3) + (((2048*(16*a^2*b^11 - 12*
a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7 + 84*a^5*b^7*c + 97*a^6*b*c^6 - 52*a
^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 -
 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 333*a^4*b^5*c^4 + 452*a^4*b^7*c^2 - 321
*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a^6*b^3*c^4 - 192*a^6*b^5*c^2 + 180*a^
7*b^3*c^3))/c^8 - (((2048*(44*a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^
7 + 12*a^8*c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*a^2*b^4*c^
8 - 119*a^2*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b^2*c^9 + 2
90*a^3*b^4*c^7 + 397*a^3*b^6*c^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c^8 - 491*a
^4*b^4*c^6 - 148*a^4*b^6*c^4 + 8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 + 102*a^5*b^
4*c^5 - 60*a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a^7*b^2*c^5
))/c^8 + (2048*tan(x/2)*(8*a*b^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b
^11*c^2 + 64*a^3*b*c^10 - 176*a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 11
2*a^7*b*c^6 - 48*a^2*b^3*c^9 - 192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*
b^9*c^3 + 364*a^3*b^3*c^8 + 212*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*
c^2 - 72*a^4*b^3*c^7 + 1008*a^4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6
 + 336*a^5*b^5*c^4 - 352*a^6*b^3*c^5))/c^8 + (((2048*(4*a*b^3*c^11 + 13*a*b
^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2*b*c^12 + 44*a^3*b*c^11 + 4*a^4
*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a^2*b^3*c^10 - 16*a^2*b^5*c^8 +
76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c^7 + 12*a^3*b^7*c^5 - 56*a^4*b^
```

3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^8 - (((2048*(32*a^3*c^13 + 64*a
^4*c^12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a*b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*
b^2*c^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8 - 176*a^3*b^2*c^11 - 46*a^3*b^4*
c^9 + 60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44*a^5*b^2*c^9))/c^8 - (2048*tan(x/
2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 + 256*a^3*b*c^12 + 320*a^4*b*c^11 + 128*a^
5*b*c^10 - 192*a^2*b^3*c^11 + 128*a^2*b^5*c^9 - 336*a^3*b^3*c^10 + 16*a^3*b
^5*c^8 - 96*a^4*b^3*c^9))/c^8 + (((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64
*a^2*b*c^14 + 80*a^3*b*c^13 + 48*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*
c^11))/c^8 + (2048*tan(x/2)*(256*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 9
6*a^5*c^12 - 64*a*b^2*c^14 + 68*a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^1
3 + 72*a^2*b^4*c^11 - 264*a^3*b^2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))
/c^8)*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3))*(b^2*2i - a*c*2i + c^2*1i))/(2*c
^3) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13 + 184*a^4*c^11 - 296*a^5*c
^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 + 48*a*b^4*c^10 + a*b^6*c^8
- 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11 + 56*a^2*b^4*c^9 + 732*a^2
*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 1817*a^3*b^4*c^8 + 440*a^3*b
^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*a^4*b^4*c^7 + 72*a^4*b^6*c
^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6*b^2*c^7))/c^8)*(b^2*2i -
a*c*2i + c^2*1i))/(2*c^3))*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3) + (2048*tan(
x/2)*(32*a*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7
*c^6 + 56*a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10
*c + 256*a^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*
a^3*b^4*c^6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a
^4*b^4*c^5 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^
5*b^6*c^2 - 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^
4*c^2 - 32*a^8*b^2*c^3))/c^8)*(b^2*2i - a*c*2i + c^2*1i)*1i)/(2*c^3))/((409
6*(16*a^6*b^6 - 4*a^8*b^4 - 4*a^7*c^5 + 15*a^8*c^4 - 14*a^9*c^3 - 48*a^7*b^
4*c + 4*a^9*b^2*c + 4*a^6*b^2*c^4 + 16*a^6*b^4*c^2 - 32*a^7*b^2*c^3 + 44*a^
8*b^2*c^2))/c^8 + (4096*tan(x/2)*(32*a^5*b^7 - 16*a^7*b^5 - 16*a^6*b*c^5 -
128*a^6*b^5*c + 60*a^7*b*c^4 - 48*a^8*b*c^3 + 32*a^8*b^3*c - 16*a^9*b*c^2 +
8*a^5*b^3*c^4 + 32*a^5*b^5*c^2 - 96*a^6*b^3*c^3 + 144*a^7*b^3*c^2))/c^8 +
(((2048*(16*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*c - 28*a^5*b*c^7 + 84*a^5*b
^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^2*b^7*c^4 + 16*a^2*
b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*c^6 + 333*a^4*b^5*c
^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3 + 328*a^6*b^3*c^4
- 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3))/c^8 + (((2048*(44*a^5*c^9 - 16*a^4*c^
10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4*a*b^6*c^7 + 15*a*b^8*c^5 + 14*
a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2*b^6*c^6 - 128*a^2*b^8*c^4 - 8*a^2*b^1
0*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*c^7 + 397*a^3*b^6*c^5 + 62*a^3*b^8*c^3
- 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6 - 148*a^4*b^6*c^4 + 8*a^4*b^8*c^2 + 22
1*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*a^5*b^6*c^3 + 68*a^6*b^2*c^6 + 136*a^6
*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 + (2048*tan(x/2)*(8*a*b^5*c^8 + 28*a*b^7*c
^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^10 - 176*a^4*b*c^9 - 32*a^5*
b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b^3*c^9 - 192*a^2*b^5*c^7 -
112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*c^8 + 212*a^3*b^5*c^6 - 592

*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 + 1008*a^4*b^5*c^5 - 128*a^4
*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 352*a^6*b^3*c^5))/c^8 - (((2
048*(4*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^9*c^5 - 16*a^2*b*c^
12 + 44*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^6*b*c^8 - 63*a^2*b^
3*c^10 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 - 104*a^3*b^5*c^7 +
12*a^3*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^5*b^3*c^7))/c^8 - (
((2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c^8 + 256*a^3*b*c^12 + 320*a^4*b*
c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 + 128*a^2*b^5*c^9 - 336*a^3*b^3*c^
10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8 - (2048*(32*a^3*c^13 + 64*a^4*c^
12 - 16*a^5*c^11 - 48*a^6*c^10 + 2*a*b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c
^12 + 96*a^2*b^4*c^10 + 8*a^2*b^6*c^8 - 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 +
60*a^4*b^2*c^10 - 8*a^4*b^4*c^8 + 44*a^5*b^2*c^9))/c^8 + (((2048*(12*a*b^5
*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13 + 48*a^4*b*c^12 - 68*
a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*(256*a^2*c^15 + 576*a
^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 + 68*a*b^4*c^12 - 8*a*
b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^3*b^2*c^12 + 8*a^3*b^
4*c^10 - 56*a^4*b^2*c^11))/c^8)*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3))*(b^2*2
i - a*c*2i + c^2*1i))/(2*c^3) + (2048*tan(x/2)*(128*a^3*c^12 - 64*a^2*c^13
+ 184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8 + 16*a*b^2*c^12 +
48*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4 - 224*a^2*b^2*c^11
+ 56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 286*a^3*b^2*c^10 - 18
17*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*a^4*b^2*c^9 - 1140*
a^4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^5*b^4*c^6 + 256*a^6
*b^2*c^7))/c^8)*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3))*(b^2*2i - a*c*2i + c^2
*1i))/(2*c^3) + (2048*tan(x/2)*(32*a*b^12 - 32*a^3*b^10 + 4*a^5*b^8 + 16*a^
5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12*a^9*c^4 + 8*a*b^8*c^4 + 32
*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 24*a^6*b^6*c - 64*a^2*b^6*c^
5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*b^6*c^4 + 1152*a^3*b^8*c^2
- 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*b^6*c^3 + 504*a^5*b^2*c^6 +
1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^2*c^5 + 812*a^6*b^4*c^3 - 3
92*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3))/c^8)*(b^2*2i - a*c*2i +
c^2*1i))/(2*c^3) - (((2048*(16*a^2*b^11 - 12*a^4*b^9 - 144*a^3*b^9*c - 28*a
^5*b*c^7 + 84*a^5*b^7*c + 97*a^6*b*c^6 - 52*a^7*b*c^5 - 60*a^8*b*c^4 + 4*a^
2*b^7*c^4 + 16*a^2*b^9*c^2 - 28*a^3*b^5*c^5 - 128*a^3*b^7*c^3 + 56*a^4*b^3*
c^6 + 333*a^4*b^5*c^4 + 452*a^4*b^7*c^2 - 321*a^5*b^3*c^5 - 600*a^5*b^5*c^3
+ 328*a^6*b^3*c^4 - 192*a^6*b^5*c^2 + 180*a^7*b^3*c^3))/c^8 - (((2048*(44*
a^5*c^9 - 16*a^4*c^10 - 4*a^6*c^8 - 64*a^7*c^7 + 12*a^8*c^6 + 4*a*b^6*c^7 +
15*a*b^8*c^5 + 14*a*b^10*c^3 - 28*a^2*b^4*c^8 - 119*a^2*b^6*c^6 - 128*a^2*
b^8*c^4 - 8*a^2*b^10*c^2 + 52*a^3*b^2*c^9 + 290*a^3*b^4*c^7 + 397*a^3*b^6*c
^5 + 62*a^3*b^8*c^3 - 227*a^4*b^2*c^8 - 491*a^4*b^4*c^6 - 148*a^4*b^6*c^4 +
8*a^4*b^8*c^2 + 221*a^5*b^2*c^7 + 102*a^5*b^4*c^5 - 60*a^5*b^6*c^3 + 68*a^
6*b^2*c^6 + 136*a^6*b^4*c^4 - 100*a^7*b^2*c^5))/c^8 + (2048*tan(x/2)*(8*a*b
^5*c^8 + 28*a*b^7*c^6 + 16*a*b^9*c^4 - 16*a*b^11*c^2 + 64*a^3*b*c^10 - 176*
a^4*b*c^9 - 32*a^5*b*c^8 + 128*a^6*b*c^7 + 112*a^7*b*c^6 - 48*a^2*b^3*c^9 -
192*a^2*b^5*c^7 - 112*a^2*b^7*c^5 + 160*a^2*b^9*c^3 + 364*a^3*b^3*c^8 + 21

```
2*a^3*b^5*c^6 - 592*a^3*b^7*c^4 + 16*a^3*b^9*c^2 - 72*a^4*b^3*c^7 + 1008*a^
4*b^5*c^5 - 128*a^4*b^7*c^3 - 720*a^5*b^3*c^6 + 336*a^5*b^5*c^4 - 352*a^6*b
^3*c^5))/c^8 + (((2048*(4*a*b^3*c^11 + 13*a*b^5*c^9 + 4*a*b^7*c^7 - 12*a*b^
9*c^5 - 16*a^2*b*c^12 + 44*a^3*b*c^11 + 4*a^4*b*c^10 + 80*a^5*b*c^9 + 12*a^
6*b*c^8 - 63*a^2*b^3*c^10 - 16*a^2*b^5*c^8 + 76*a^2*b^7*c^6 - a^3*b^3*c^9 -
 104*a^3*b^5*c^7 + 12*a^3*b^7*c^5 - 56*a^4*b^3*c^8 - 60*a^4*b^5*c^6 + 48*a^
5*b^3*c^7))/c^8 - (((2048*(32*a^3*c^13 + 64*a^4*c^12 - 16*a^5*c^11 - 48*a^6
*c^10 + 2*a*b^4*c^11 - 14*a*b^6*c^9 - 16*a^2*b^2*c^12 + 96*a^2*b^4*c^10 + 8
*a^2*b^6*c^8 - 176*a^3*b^2*c^11 - 46*a^3*b^4*c^9 + 60*a^4*b^2*c^10 - 8*a^4*
b^4*c^8 + 44*a^5*b^2*c^9))/c^8 - (2048*tan(x/2)*(32*a*b^5*c^10 - 16*a*b^7*c
^8 + 256*a^3*b*c^12 + 320*a^4*b*c^11 + 128*a^5*b*c^10 - 192*a^2*b^3*c^11 +
128*a^2*b^5*c^9 - 336*a^3*b^3*c^10 + 16*a^3*b^5*c^8 - 96*a^4*b^3*c^9))/c^8
+ (((2048*(12*a*b^5*c^11 - 16*a*b^3*c^13 + 64*a^2*b*c^14 + 80*a^3*b*c^13 +
48*a^4*b*c^12 - 68*a^2*b^3*c^12 - 12*a^3*b^3*c^11))/c^8 + (2048*tan(x/2)*(2
56*a^2*c^15 + 576*a^3*c^14 + 416*a^4*c^13 + 96*a^5*c^12 - 64*a*b^2*c^14 + 6
8*a*b^4*c^12 - 8*a*b^6*c^10 - 416*a^2*b^2*c^13 + 72*a^2*b^4*c^11 - 264*a^3*
b^2*c^12 + 8*a^3*b^4*c^10 - 56*a^4*b^2*c^11))/c^8)*(b^2*2i - a*c*2i + c^2*1
i))/(2*c^3))*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3) + (2048*tan(x/2)*(128*a^3*
c^12 - 64*a^2*c^13 + 184*a^4*c^11 - 296*a^5*c^10 - 352*a^6*c^9 - 72*a^7*c^8
 + 16*a*b^2*c^12 + 48*a*b^4*c^10 + a*b^6*c^8 - 92*a*b^8*c^6 + 8*a*b^10*c^4
- 224*a^2*b^2*c^11 + 56*a^2*b^4*c^9 + 732*a^2*b^6*c^7 - 88*a^2*b^8*c^5 - 28
6*a^3*b^2*c^10 - 1817*a^3*b^4*c^8 + 440*a^3*b^6*c^6 - 8*a^3*b^8*c^4 + 1502*
a^4*b^2*c^9 - 1140*a^4*b^4*c^7 + 72*a^4*b^6*c^5 + 1208*a^5*b^2*c^8 - 220*a^
5*b^4*c^6 + 256*a^6*b^2*c^7))/c^8)*(b^2*2i - a*c*2i + c^2*1i))/(2*c^3))*(b^
2*2i - a*c*2i + c^2*1i))/(2*c^3) + (2048*tan(x/2)*(32*a*b^12 - 32*a^3*b^10
+ 4*a^5*b^8 + 16*a^5*c^8 - 48*a^6*c^7 + 2*a^7*c^6 + 56*a^8*c^5 + 12*a^9*c^4
 + 8*a*b^8*c^4 + 32*a*b^10*c^2 - 320*a^2*b^10*c + 256*a^4*b^8*c - 24*a^6*b^
6*c - 64*a^2*b^6*c^5 - 288*a^2*b^8*c^3 + 160*a^3*b^4*c^6 + 888*a^3*b^6*c^4
+ 1152*a^3*b^8*c^2 - 128*a^4*b^2*c^7 - 1104*a^4*b^4*c^5 - 1824*a^4*b^6*c^3
+ 504*a^5*b^2*c^6 + 1249*a^5*b^4*c^4 - 700*a^5*b^6*c^2 - 292*a^6*b^2*c^5 +
812*a^6*b^4*c^3 - 392*a^7*b^2*c^4 + 44*a^7*b^4*c^2 - 32*a^8*b^2*c^3))/c^8)*
(b^2*2i - a*c*2i + c^2*1i))/(2*c^3)))*(b^2*2i - a*c*2i + c^2*1i)*1i)/c^3
```

3.2 $\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 298

$$\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{bx}{c^2}+\frac{\sqrt{2}b\left(b-\frac{ac}{b}-\frac{b^2}{\sqrt{b^2-4ac}}+\frac{3ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c^2\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}+\frac{\sqrt{2}b\left(b-\frac{ac}{b}+\frac{b^2}{\sqrt{b^2-4ac}}-\frac{3ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c^2\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}-\frac{\cos(x)}{c}$$

```
[Out] -b*x/c^2-cos(x)/c+b*arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1
/2)/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(b-a*c/b-b^2/(-4*a*
c+b^2)^(1/2)+3*a*c/(-4*a*c+b^2)^(1/2))/c^2/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1
/2))^(1/2)+b*arctan(1/2*(2*c+(b+(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^
2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(b-a*c/b+b^2/(-4*a*c+b^2)^
(1/2)-3*a*c/(-4*a*c+b^2)^(1/2))/c^2/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1
/2)
```

Rubi [A] (verified)

Time = 3.74 (sec) , antiderivative size = 298, normalized size of antiderivative = 1.00, number of steps used = 10, number of rules used = 6, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.316, Rules used = {3337, 2718, 3373, 2739, 632, 210}

$$\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\sqrt{2}b\left(-\frac{b^2}{\sqrt{b^2-4ac}}+\frac{3ac}{\sqrt{b^2-4ac}}-\frac{ac}{b}+b\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c^2\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{\sqrt{2}b\left(\frac{b^2}{\sqrt{b^2-4ac}}-\frac{3ac}{\sqrt{b^2-4ac}}-\frac{ac}{b}+b\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c^2\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{bx}{c^2} - \frac{\cos(x)}{c}$$

```
[In] Int[Sin[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -((b*x)/c^2) + (Sqrt[2]*b*(b - (a*c)/b - b^2/Sqrt[b^2 - 4*a*c] + (3*a*c)/Sq
rt[b^2 - 4*a*c])*ArcTan[(2*c + (b - Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*S
qrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]])])/(c^2*Sqrt[b^2 - 2*c*(a + c)
 - b*Sqrt[b^2 - 4*a*c]]) + (Sqrt[2]*b*(b - (a*c)/b + b^2/Sqrt[b^2 - 4*a*c]
- (3*a*c)/Sqrt[b^2 - 4*a*c])*ArcTan[(2*c + (b + Sqrt[b^2 - 4*a*c])*Tan[x/2]
)/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])])/(c^2*Sqrt[b^2 -
 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]]) - Cos[x]/c
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2718

```
Int[sin[(c_.) + (d_.)*(x_)], x_Symbol] :> Simp[-Cos[c + d*x]/d, x] /; FreeQ
[{c, d}, x]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
```

```
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3337

```
Int[sin[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_), x_Symbol] :> Int[ExpandTr
ig[sin[d + e*x]^m*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n))^p, x], x] /
; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && NeQ[b^2 - 4*a*c, 0] && Integ
ersQ[m, n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rubi steps

$$\text{integral} = \int \left(-\frac{b}{c^2}+\frac{\sin(x)}{c}+\frac{ab+b^2\left(1-\frac{ac}{b^2}\right)\sin(x)}{c^2\left(a+b\sin(x)+c\sin^2(x)\right)}\right)dx$$

$$= -\frac{bx}{c^2}+\frac{\int \frac{ab+b^2\left(1-\frac{ac}{b^2}\right)\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{c^2}+\frac{\int \sin(x)\,dx}{c}$$

$$= -\frac{bx}{c^2}-\frac{\cos(x)}{c}+\frac{\left(b^2-ac+\frac{b^3}{\sqrt{b^2-4ac}}-\frac{3abc}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c^2}$$
$$+\frac{\left(b^2-ac-\frac{b^3}{\sqrt{b^2-4ac}}+\frac{3abc}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c^2}$$

$$= -\frac{bx}{c^2}-\frac{\cos(x)}{c}$$
$$+\frac{\left(2\left(b^2-ac+\frac{b^3}{\sqrt{b^2-4ac}}-\frac{3abc}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)}{c^2}$$
$$+\frac{\left(2\left(b^2-ac-\frac{b^3}{\sqrt{b^2-4ac}}+\frac{3abc}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)}{c^2}$$

$$= -\frac{bx}{c^2} - \frac{\cos(x)}{c}$$

$$-\frac{\left(4\left(b^2-ac+\frac{b^3}{\sqrt{b^2-4ac}}-\frac{3abc}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int\frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}dx,x,4c+2\left(b+\sqrt{b^2-4ac}\right.\right.}{c^2}$$

$$-\frac{\left(4\left(b^2-ac-\frac{b^3}{\sqrt{b^2-4ac}}+\frac{3abc}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int\frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}dx,x,4c+2\left(b-\sqrt{b^2-\right.\right.}{c^2}$$

$$= -\frac{bx}{c^2} + \frac{\sqrt{2}\left(b^2-ac-\frac{b^3}{\sqrt{b^2-4ac}}+\frac{3abc}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c^2\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}$$

$$+\frac{\sqrt{2}\left(b^2-ac+\frac{b^3}{\sqrt{b^2-4ac}}-\frac{3abc}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c^2\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} - \frac{\cos(x)}{c}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 0.75 (sec) , antiderivative size = 358, normalized size of antiderivative = 1.20

$$\int\frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}dx$$

$$= \frac{-bx+\frac{\left(ib^3-3iabc+b^2\sqrt{-b^2+4ac}-ac\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}+\frac{\left(-ib^3+3iabc+b^2\sqrt{-b^2+4ac}-ac\sqrt{-b^2+4ac}\right)a}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+}}}{c^2}$$

```
[In] Integrate[Sin[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (-(b*x) + ((I*b^3 - (3*I)*a*b*c + b^2*Sqrt[-b^2 + 4*a*c] - a*c*Sqrt[-b^2 +
4*a*c])*ArcTan[(2*c + (b - I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^
2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b
^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]]) + (((-I)*b^3 + (3*I)*a*b*c + b^
2*Sqrt[-b^2 + 4*a*c] - a*c*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b + I*Sqrt[-b
^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*
a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4
*a*c]]) - c*Cos[x])/c^2
```

Maple [A] (verified)

Time = 2.21 (sec) , antiderivative size = 305, normalized size of antiderivative = 1.02

method	result
default	$\frac{2a\left(\frac{2\left(-2\sqrt{-4ac+b^2}\,ac+\sqrt{-4ac+b^2}\,b^2+4bca-b^3\right)\arctan\left(\frac{2a\tan\left(\frac{x}{2}\right)+b+\sqrt{-4ac+b^2}}{\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}\right)}{\left(8ac-2b^2\right)\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}-\frac{2\left(2\sqrt{-4ac+b^2}\,ac-\sqrt{-4ac+b^2}\,b^2+4bca-b^3\right)\arctan \cdots}{\left(8ac-2b^2\right)\sqrt{4ac-2b^2+2b\sqrt{-\cdots}}}\right.}{c^2}$
risch	Expression too large to display

```
[In] int(sin(x)^3/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 2/c^2*a*(2*(-2*(-4*a*c+b^2)^(1/2)*a*c+(-4*a*c+b^2)^(1/2)*b^2+4*b*c*a-b^3)/(
8*a*c-2*b^2)/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((2*a*t
an(1/2*x)+b+(-4*a*c+b^2)^(1/2))/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^
(1/2))-2*(2*(-4*a*c+b^2)^(1/2)*a*c-(-4*a*c+b^2)^(1/2)*b^2+4*b*c*a-b^3)/(8*a
*c-2*b^2)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((-2*a*tan
(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1
/2)))-2/c^2*(c/(1+tan(1/2*x)^2)+b*arctan(tan(1/2*x)))
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 6531 vs. $2(261) = 522$.

Time = 2.25 (sec) , antiderivative size = 6531, normalized size of antiderivative = 21.92

$$\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(sin(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] Too large to include
```

Sympy [F(-1)]

Timed out.

$$\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(sin(x)**3/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Timed out
```

Maxima [F]

$$\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx=\int \frac{\sin{(x)}^3}{c\sin{(x)}^2+b\sin{(x)}+a}\,dx$$

```
[In] integrate(sin(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] -(c^2*integrate(-2*(2*(b^3 - a*b*c)*cos(3*x)^2 + 4*(2*a^2*b + a*b*c)*cos(2*
x)^2 + 2*(b^3 - a*b*c)*cos(x)^2 + 2*(b^3 - a*b*c)*sin(3*x)^2 + 2*(4*a*b^2 -
 a*c^2 - (2*a^2 - b^2)*c)*cos(x)*sin(2*x) + 4*(2*a^2*b + a*b*c)*sin(2*x)^2
+ 2*(b^3 - a*b*c)*sin(x)^2 - (2*a*b*c*cos(2*x) + (b^2*c - a*c^2)*sin(3*x) -
 (b^2*c - a*c^2)*sin(x))*cos(4*x) - 2*(2*(b^3 - a*b*c)*cos(x) + (4*a*b^2 -
a*c^2 - (2*a^2 - b^2)*c)*sin(2*x))*cos(3*x) - 2*(a*b*c + (4*a*b^2 - a*c^2 -
 (2*a^2 - b^2)*c)*sin(x))*cos(2*x) - (2*a*b*c*sin(2*x) - (b^2*c - a*c^2)*co
s(3*x) + (b^2*c - a*c^2)*cos(x))*sin(4*x) - (b^2*c - a*c^2 - 2*(4*a*b^2 - a
*c^2 - (2*a^2 - b^2)*c)*cos(2*x) + 4*(b^3 - a*b*c)*sin(x))*sin(3*x) + (b^2*
c - a*c^2)*sin(x))/(c^4*cos(4*x)^2 + 4*b^2*c^2*cos(3*x)^2 + 4*b^2*c^2*cos(x
)^2 + c^4*sin(4*x)^2 + 4*b^2*c^2*sin(3*x)^2 + 4*b^2*c^2*sin(x)^2 + 4*b*c^3*
sin(x) + c^4 + 4*(4*a^2*c^2 + 4*a*c^3 + c^4)*cos(2*x)^2 + 8*(2*a*b*c^2 + b*
c^3)*cos(x)*sin(2*x) + 4*(4*a^2*c^2 + 4*a*c^3 + c^4)*sin(2*x)^2 - 2*(2*b*c^
3*sin(3*x) - 2*b*c^3*sin(x) - c^4 + 2*(2*a*c^3 + c^4)*cos(2*x))*cos(4*x) -
8*(b^2*c^2*cos(x) + (2*a*b*c^2 + b*c^3)*sin(2*x))*cos(3*x) - 4*(2*a*c^3 + c
^4 + 2*(2*a*b*c^2 + b*c^3)*sin(x))*cos(2*x) + 4*(b*c^3*cos(3*x) - b*c^3*cos
(x) - (2*a*c^3 + c^4)*sin(2*x))*sin(4*x) - 4*(2*b^2*c^2*sin(x) + b*c^3 - 2*
(2*a*b*c^2 + b*c^3)*cos(2*x))*sin(3*x)), x) + b*x + c*cos(x))/c^2
```

Giac [F(-1)]

Timed out.

$$\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx=\text{Timed out}$$

```
[In] integrate(sin(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 25.58 (sec) , antiderivative size = 21407, normalized size of antiderivative = 71.84

$$\int \frac{\sin^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(sin(x)^3/(a + c*sin(x)^2 + b*sin(x)),x)
```

```
[Out] - 2/(c*(tan(x/2)^2 + 1)) - atan((((8192*(4*a^2*b^7 - 3*a^4*b^5 - 20*a^3*b^5
*c + 9*a^5*b^3*c + 20*a^4*b^3*c^2))/c^4 + ((8192*(4*a*b^7*c^2 - 2*a^2*b^7*c
 + 2*a^4*b^5*c + 12*a^5*b*c^4 + 8*a^6*b*c^3 - 24*a^2*b^5*c^3 + 32*a^3*b^3*c
^4 + 10*a^3*b^5*c^2 - 10*a^4*b^3*c^3 - 10*a^5*b^3*c^2))/c^4 + ((b^8 - a^2*b
^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a
^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*
b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(
4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8
+ 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5
- 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((8192*(3*a*b^7*c^3
 - 4*a*b^5*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^5 + 16*a^2*b^3*c^6 - 13*a^2*b^5*c
^4 - 3*a^3*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 + ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*
a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b
^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*
c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/
2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a
^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 +
a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((8192*(3*a*b^5*c^6 + 16*a^3*b*c^8 - 4
*a^4*b*c^7 - 8*a^5*b*c^6 - 16*a^2*b^3*c^7 - 2*a^2*b^5*c^5 + 9*a^3*b^3*c^6 +
 2*a^4*b^3*c^5))/c^4 + ((8192*(3*a*b^5*c^7 - 4*a*b^3*c^9 + 16*a^2*b*c^10 +
20*a^3*b*c^9 + 12*a^4*b*c^8 - 17*a^2*b^3*c^8 - 3*a^3*b^3*c^7))/c^4 + (8192*
tan(x/2)*(64*a^2*c^11 + 144*a^3*c^10 + 104*a^4*c^9 + 24*a^5*c^8 - 16*a*b^2*
c^10 + 17*a*b^4*c^8 - 2*a*b^6*c^6 - 104*a^2*b^2*c^9 + 18*a^2*b^4*c^7 - 66*a
^3*b^2*c^8 + 2*a^3*b^4*c^6 - 14*a^4*b^2*c^7))/c^4)*((b^8 - a^2*b^6 + 8*a^4*
c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4
*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 1
0*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2
)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^
7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^
2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(32*a^3*c^9 +
 48*a^4*c^8 + 16*a^5*c^7 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40*a^2*b^2*c^8 + 28*
a^2*b^4*c^6 - 60*a^3*b^2*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2*c^6))/c^4) - (819
2*tan(x/2)*(16*a^4*c^7 + 24*a^5*c^6 + 10*a^6*c^5 + 16*a*b^4*c^6 - 24*a*b^6*
c^4 + 2*a*b^8*c^2 - 64*a^2*b^2*c^7 + 144*a^2*b^4*c^5 - 18*a^2*b^6*c^3 - 200
*a^3*b^2*c^6 + 75*a^3*b^4*c^4 - 2*a^3*b^6*c^2 - 142*a^4*b^2*c^5 + 14*a^4*b^
4*c^3 - 27*a^5*b^2*c^4))/c^4) - (8192*tan(x/2)*(8*a^5*c^5 + 4*a^6*c^4 - 8*a
```

```
*b^6*c^3 - 4*a^3*b^6*c + 40*a^2*b^4*c^4 - 28*a^2*b^6*c^2 - 32*a^3*b^2*c^5 +
 60*a^3*b^4*c^3 - 56*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 + 4*a*b^
8*c))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)
^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 -
38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^
3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3
)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a
*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(
1/2) + (8192*tan(x/2)*(8*a*b^8 - 8*a^3*b^6 + a^5*b^4 + a^7*c^2 - 48*a^2*b^6
*c + 32*a^4*b^4*c - 2*a^6*b^2*c + 72*a^3*b^4*c^2 - 16*a^4*b^2*c^3 - 16*a^5*
b^2*c^2))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2
)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^
2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b
^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^
2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 -
 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)
))^(1/2)*1i + ((8192*(4*a^2*b^7 - 3*a^4*b^5 - 20*a^3*b^5*c + 9*a^5*b^3*c +
20*a^4*b^3*c^2))/c^4 - ((8192*(4*a*b^7*c^2 - 2*a^2*b^7*c + 2*a^4*b^5*c + 12
*a^5*b*c^4 + 8*a^6*b*c^3 - 24*a^2*b^5*c^3 + 32*a^3*b^3*c^4 + 10*a^3*b^5*c^2
 - 10*a^4*b^3*c^3 - 10*a^5*b^3*c^2))/c^4 + ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*
a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b
^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*
c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/
2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a
^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 +
a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*(((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c
^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3
)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3
*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) +
2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^
6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b
^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((8192*(3*a*b^5*c^6 + 16*a^3*b*c^8 - 4*a^4*
b*c^7 - 8*a^5*b*c^6 - 16*a^2*b^3*c^7 - 2*a^2*b^5*c^5 + 9*a^3*b^3*c^6 + 2*a^
4*b^3*c^5))/c^4 - ((8192*(3*a*b^5*c^7 - 4*a*b^3*c^9 + 16*a^2*b*c^10 + 20*a^
3*b*c^9 + 12*a^4*b*c^8 - 17*a^2*b^3*c^8 - 3*a^3*b^3*c^7))/c^4 + (8192*tan(x
/2)*(64*a^2*c^11 + 144*a^3*c^10 + 104*a^4*c^9 + 24*a^5*c^8 - 16*a*b^2*c^10
+ 17*a*b^4*c^8 - 2*a*b^6*c^6 - 104*a^2*b^2*c^9 + 18*a^2*b^4*c^7 - 66*a^3*b^
2*c^8 + 2*a^3*b^4*c^6 - 14*a^4*b^2*c^7))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 +
 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c
- b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b
^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^
(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 1
6*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6
 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(32*a^3*c^9 + 48*a
^4*c^8 + 16*a^5*c^7 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40*a^2*b^2*c^8 + 28*a^2*b
```

^4*c^6 - 60*a^3*b^2*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2*c^6))/c^4) - (8192*(3*
a*b^7*c^3 - 4*a*b^5*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^5 + 16*a^2*b^3*c^6 - 13*
a^2*b^5*c^4 - 3*a^3*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 + (8192*tan(x/2)*(16*a^4*
c^7 + 24*a^5*c^6 + 10*a^6*c^5 + 16*a*b^4*c^6 - 24*a*b^6*c^4 + 2*a*b^8*c^2 -
64*a^2*b^2*c^7 + 144*a^2*b^4*c^5 - 18*a^2*b^6*c^3 - 200*a^3*b^2*c^6 + 75*a
^3*b^4*c^4 - 2*a^3*b^6*c^2 - 142*a^4*b^2*c^5 + 14*a^4*b^4*c^3 - 27*a^5*b^2*
c^4))/c^4) - (8192*tan(x/2)*(8*a^5*c^5 + 4*a^6*c^4 - 8*a*b^6*c^3 - 4*a^3*b^
6*c + 40*a^2*b^4*c^4 - 28*a^2*b^6*c^2 - 32*a^3*b^2*c^5 + 60*a^3*b^4*c^3 - 5
6*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 + 4*a*b^8*c))/c^4)*((b^8 -
a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*
c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18
*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*
c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2
*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4
*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/
2)*(8*a*b^8 - 8*a^3*b^6 + a^5*b^4 + a^7*c^2 - 48*a^2*b^6*c + 32*a^4*b^4*c -
2*a^6*b^2*c + 72*a^3*b^4*c^2 - 16*a^4*b^2*c^3 - 16*a^5*b^2*c^2))/c^4)*((b^
8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*
b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3
- 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*
b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16
*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a
*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*1i)/(((819
2*(4*a^2*b^7 - 3*a^4*b^5 - 20*a^3*b^5*c + 9*a^5*b^3*c + 20*a^4*b^3*c^2))/c^
4 - ((8192*(4*a*b^7*c^2 - 2*a^2*b^7*c + 2*a^4*b^5*c + 12*a^5*b*c^4 + 8*a^6*
b*c^3 - 24*a^2*b^5*c^3 + 32*a^3*b^3*c^4 + 10*a^3*b^5*c^2 - 10*a^4*b^3*c^3 -
10*a^5*b^3*c^2))/c^4 + ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*
a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a
^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(
4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4
*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 -
b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3
*b^2*c^5)))^(1/2)*(((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c -
b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^
4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c
- b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c
- b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c
^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*
c^5)))^(1/2)*((8192*(3*a*b^5*c^6 + 16*a^3*b*c^8 - 4*a^4*b*c^7 - 8*a^5*b*c^6
- 16*a^2*b^3*c^7 - 2*a^2*b^5*c^5 + 9*a^3*b^3*c^6 + 2*a^4*b^3*c^5))/c^4 - (
(8192*(3*a*b^5*c^7 - 4*a*b^3*c^9 + 16*a^2*b*c^10 + 20*a^3*b*c^9 + 12*a^4*b*
c^8 - 17*a^2*b^3*c^8 - 3*a^3*b^3*c^7))/c^4 + (8192*tan(x/2)*(64*a^2*c^11 +
144*a^3*c^10 + 104*a^4*c^9 + 24*a^5*c^8 - 16*a*b^2*c^10 + 17*a*b^4*c^8 - 2*
a*b^6*c^6 - 104*a^2*b^2*c^9 + 18*a^2*b^4*c^7 - 66*a^3*b^2*c^8 + 2*a^3*b^4*c
^6 - 14*a^4*b^2*c^7))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-

```
(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 3
3*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*
(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(
-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6
 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*
a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(32*a^3*c^9 + 48*a^4*c^8 + 16*a^5*c^7
 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40*a^2*b^2*c^8 + 28*a^2*b^4*c^6 - 60*a^3*b^2
*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2*c^6))/c^4) - (8192*(3*a*b^7*c^3 - 4*a*b^5
*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^5 + 16*a^2*b^3*c^6 - 13*a^2*b^5*c^4 - 3*a^3
*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 + (8192*tan(x/2)*(16*a^4*c^7 + 24*a^5*c^6 +
10*a^6*c^5 + 16*a*b^4*c^6 - 24*a*b^6*c^4 + 2*a*b^8*c^2 - 64*a^2*b^2*c^7 + 1
44*a^2*b^4*c^5 - 18*a^2*b^6*c^3 - 200*a^3*b^2*c^6 + 75*a^3*b^4*c^4 - 2*a^3*
b^6*c^2 - 142*a^4*b^2*c^5 + 14*a^4*b^4*c^3 - 27*a^5*b^2*c^4))/c^4) - (8192*
tan(x/2)*(8*a^5*c^5 + 4*a^6*c^4 - 8*a*b^6*c^3 - 4*a^3*b^6*c + 40*a^2*b^4*c^
4 - 28*a^2*b^6*c^2 - 32*a^3*b^2*c^5 + 60*a^3*b^4*c^3 - 56*a^4*b^2*c^4 + 20*
a^4*b^4*c^2 - 16*a^5*b^2*c^3 + 4*a*b^8*c))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4
 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*
c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a
*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3
)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 +
 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c
^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(8*a*b^8 - 8*a^3
*b^6 + a^5*b^4 + a^7*c^2 - 48*a^2*b^6*c + 32*a^4*b^4*c - 2*a^6*b^2*c + 72*a
^3*b^4*c^2 - 16*a^4*b^2*c^3 - 16*a^5*b^2*c^2))/c^4)*((b^8 - a^2*b^6 + 8*a^4
*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(
4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 -
10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^
2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c
^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b
^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) - ((8192*(4*a^2*b^7 - 3*a^4*b
^5 - 20*a^3*b^5*c + 9*a^5*b^3*c + 20*a^4*b^3*c^2))/c^4 + ((8192*(4*a*b^7*c^
2 - 2*a^2*b^7*c + 2*a^4*b^5*c + 12*a^5*b*c^4 + 8*a^6*b*c^3 - 24*a^2*b^5*c^3
 + 32*a^3*b^3*c^4 + 10*a^3*b^5*c^2 - 10*a^4*b^3*c^3 - 10*a^5*b^3*c^2))/c^4
+ ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) +
8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^
2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2)
- 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/
(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7
+ 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((81
92*(3*a*b^7*c^3 - 4*a*b^5*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^5 + 16*a^2*b^3*c^6
 - 13*a^2*b^5*c^4 - 3*a^3*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 + ((b^8 - a^2*b^6 +
 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b
^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*
c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*
c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32
```

*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32
*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((8192*(3*a*b^5*c^6 + 1
6*a^3*b*c^8 - 4*a^4*b*c^7 - 8*a^5*b*c^6 - 16*a^2*b^3*c^7 - 2*a^2*b^5*c^5 +
9*a^3*b^3*c^6 + 2*a^4*b^3*c^5))/c^4 + ((8192*(3*a*b^5*c^7 - 4*a*b^3*c^9 + 1
6*a^2*b*c^10 + 20*a^3*b*c^9 + 12*a^4*b*c^8 - 17*a^2*b^3*c^8 - 3*a^3*b^3*c^7
))/c^4 + (8192*tan(x/2)*(64*a^2*c^11 + 144*a^3*c^10 + 104*a^4*c^9 + 24*a^5*
c^8 - 16*a*b^2*c^10 + 17*a*b^4*c^8 - 2*a*b^6*c^6 - 104*a^2*b^2*c^9 + 18*a^2
*b^4*c^7 - 66*a^3*b^2*c^8 + 2*a^3*b^4*c^6 - 14*a^4*b^2*c^7))/c^4)*((b^8 - a
^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c
 - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*
a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c
*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*
c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*
c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2
)*(32*a^3*c^9 + 48*a^4*c^8 + 16*a^5*c^7 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40*a^
2*b^2*c^8 + 28*a^2*b^4*c^6 - 60*a^3*b^2*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2*c^
6))/c^4) - (8192*tan(x/2)*(16*a^4*c^7 + 24*a^5*c^6 + 10*a^6*c^5 + 16*a*b^4*
c^6 - 24*a*b^6*c^4 + 2*a*b^8*c^2 - 64*a^2*b^2*c^7 + 144*a^2*b^4*c^5 - 18*a^
2*b^6*c^3 - 200*a^3*b^2*c^6 + 75*a^3*b^4*c^4 - 2*a^3*b^6*c^2 - 142*a^4*b^2*
c^5 + 14*a^4*b^4*c^3 - 27*a^5*b^2*c^4))/c^4) - (8192*tan(x/2)*(8*a^5*c^5 +
4*a^6*c^4 - 8*a*b^6*c^3 - 4*a^3*b^6*c + 40*a^2*b^4*c^4 - 28*a^2*b^6*c^2 - 3
2*a^3*b^2*c^5 + 60*a^3*b^4*c^3 - 56*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b
^2*c^3 + 4*a*b^8*c))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5*(-(
4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33
*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c^2*(
-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*c*(-
(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6
- b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a
^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(8*a*b^8 - 8*a^3*b^6 + a^5*b^4 + a^7*c
^2 - 48*a^2*b^6*c + 32*a^4*b^4*c - 2*a^6*b^2*c + 72*a^3*b^4*c^2 - 16*a^4*b^
2*c^3 - 16*a^5*b^2*c^2))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 + b^5
*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c - b^2)^3)^(1/2)
+ 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c + 3*a^2*b*c
^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) + 2*a^3*b*
c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*
c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 -
 8*a^3*b^2*c^5)))^(1/2) + (16384*(a^7*b - 4*a^5*b^3))/c^4 + (16384*tan(x/2)
*(4*a^6*b^2 - 8*a^4*b^4 + 8*a^5*b^2*c))/c^4))*((b^8 - a^2*b^6 + 8*a^4*c^4 +
 8*a^5*c^3 + b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c - a^2*b^3*(-(4*a*c
- b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b
^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^
(1/2) + 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 1
6*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6
 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*2i - atan((((8192*(4*a^2*b^7 - 3*a^
4*b^5 - 20*a^3*b^5*c + 9*a^5*b^3*c + 20*a^4*b^3*c^2))/c^4 + ((8192*(4*a*b^7

```
*c^2 - 2*a^2*b^7*c + 2*a^4*b^5*c + 12*a^5*b*c^4 + 8*a^6*b*c^3 - 24*a^2*b^5*
c^3 + 32*a^3*b^3*c^4 + 10*a^3*b^5*c^2 - 10*a^4*b^3*c^3 - 10*a^5*b^3*c^2))/c
^4 + ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2)
 + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3
*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/
2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2
))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c
^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*(
(8192*(3*a*b^7*c^3 - 4*a*b^5*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^5 + 16*a^2*b^3*
c^6 - 13*a^2*b^5*c^4 - 3*a^3*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 + ((b^8 - a^2*b^
6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^
2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b
^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4
*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 +
 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 -
 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((8192*(3*a*b^5*c^6
+ 16*a^3*b*c^8 - 4*a^4*b*c^7 - 8*a^5*b*c^6 - 16*a^2*b^3*c^7 - 2*a^2*b^5*c^5
 + 9*a^3*b^3*c^6 + 2*a^4*b^3*c^5))/c^4 + ((8192*(3*a*b^5*c^7 - 4*a*b^3*c^9
+ 16*a^2*b*c^10 + 20*a^3*b*c^9 + 12*a^4*b*c^8 - 17*a^2*b^3*c^8 - 3*a^3*b^3*
c^7))/c^4 + (8192*tan(x/2)*(64*a^2*c^11 + 144*a^3*c^10 + 104*a^4*c^9 + 24*a
^5*c^8 - 16*a*b^2*c^10 + 17*a*b^4*c^8 - 2*a*b^6*c^6 - 104*a^2*b^2*c^9 + 18*
a^2*b^4*c^7 - 66*a^3*b^2*c^8 + 2*a^3*b^4*c^6 - 14*a^4*b^2*c^7))/c^4)*((b^8
- a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^
4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 -
18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^
3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a
^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b
^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(
x/2)*(32*a^3*c^9 + 48*a^4*c^8 + 16*a^5*c^7 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40
*a^2*b^2*c^8 + 28*a^2*b^4*c^6 - 60*a^3*b^2*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2
*c^6))/c^4) - (8192*tan(x/2)*(16*a^4*c^7 + 24*a^5*c^6 + 10*a^6*c^5 + 16*a*b
^4*c^6 - 24*a*b^6*c^4 + 2*a*b^8*c^2 - 64*a^2*b^2*c^7 + 144*a^2*b^4*c^5 - 18
*a^2*b^6*c^3 - 200*a^3*b^2*c^6 + 75*a^3*b^4*c^4 - 2*a^3*b^6*c^2 - 142*a^4*b
^2*c^5 + 14*a^4*b^4*c^3 - 27*a^5*b^2*c^4))/c^4) - (8192*tan(x/2)*(8*a^5*c^5
 + 4*a^6*c^4 - 8*a*b^6*c^3 - 4*a^3*b^6*c + 40*a^2*b^4*c^4 - 28*a^2*b^6*c^2
- 32*a^3*b^2*c^5 + 60*a^3*b^4*c^3 - 56*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^
5*b^2*c^3 + 4*a*b^8*c))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*
(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) +
 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^
2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c
*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c
^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 -
8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(8*a*b^8 - 8*a^3*b^6 + a^5*b^4 + a^
7*c^2 - 48*a^2*b^6*c + 32*a^4*b^4*c - 2*a^6*b^2*c + 72*a^3*b^4*c^2 - 16*a^4
*b^2*c^3 - 16*a^5*b^2*c^2))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 -
```

```
b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/
2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*
b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3
*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b
^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^
4 - 8*a^3*b^2*c^5)))^(1/2)*1i + ((8192*(4*a^2*b^7 - 3*a^4*b^5 - 20*a^3*b^5*
c + 9*a^5*b^3*c + 20*a^4*b^3*c^2))/c^4 - ((8192*(4*a*b^7*c^2 - 2*a^2*b^7*c
+ 2*a^4*b^5*c + 12*a^5*b*c^4 + 8*a^6*b*c^3 - 24*a^2*b^5*c^3 + 32*a^3*b^3*c^
4 + 10*a^3*b^5*c^2 - 10*a^4*b^3*c^3 - 10*a^5*b^3*c^2))/c^4 + ((b^8 - a^2*b^
6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^
2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b
^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4
*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 +
 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 -
 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*(((b^8 - a^2*b^6 + 8
*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3
*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^
2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c
- b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a
^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a
^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((8192*(3*a*b^5*c^6 + 16*
a^3*b*c^8 - 4*a^4*b*c^7 - 8*a^5*b*c^6 - 16*a^2*b^3*c^7 - 2*a^2*b^5*c^5 + 9*
a^3*b^3*c^6 + 2*a^4*b^3*c^5))/c^4 - ((8192*(3*a*b^5*c^7 - 4*a*b^3*c^9 + 16*
a^2*b*c^10 + 20*a^3*b*c^9 + 12*a^4*b*c^8 - 17*a^2*b^3*c^8 - 3*a^3*b^3*c^7))
/c^4 + (8192*tan(x/2)*(64*a^2*c^11 + 144*a^3*c^10 + 104*a^4*c^9 + 24*a^5*c^
8 - 16*a*b^2*c^10 + 17*a*b^4*c^8 - 2*a*b^6*c^6 - 104*a^2*b^2*c^9 + 18*a^2*b
^4*c^7 - 66*a^3*b^2*c^8 + 2*a^3*b^4*c^6 - 14*a^4*b^2*c^7))/c^4)*((b^8 - a^2
*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c +
 a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^
4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(
-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^
8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^
5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*
(32*a^3*c^9 + 48*a^4*c^8 + 16*a^5*c^7 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40*a^2*
b^2*c^8 + 28*a^2*b^4*c^6 - 60*a^3*b^2*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2*c^6)
)/c^4) - (8192*(3*a*b^7*c^3 - 4*a*b^5*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^5 + 16
*a^2*b^3*c^6 - 13*a^2*b^5*c^4 - 3*a^3*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 + (8192
*tan(x/2)*(16*a^4*c^7 + 24*a^5*c^6 + 10*a^6*c^5 + 16*a*b^4*c^6 - 24*a*b^6*c
^4 + 2*a*b^8*c^2 - 64*a^2*b^2*c^7 + 144*a^2*b^4*c^5 - 18*a^2*b^6*c^3 - 200*
a^3*b^2*c^6 + 75*a^3*b^4*c^4 - 2*a^3*b^6*c^2 - 142*a^4*b^2*c^5 + 14*a^4*b^4
*c^3 - 27*a^5*b^2*c^4))/c^4) - (8192*tan(x/2)*(8*a^5*c^5 + 4*a^6*c^4 - 8*a*
b^6*c^3 - 4*a^3*b^6*c + 40*a^2*b^4*c^4 - 28*a^2*b^6*c^2 - 32*a^3*b^2*c^5 +
60*a^3*b^4*c^3 - 56*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 + 4*a*b^8
*c))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^
(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 3
```

```
8*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3
)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)
^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*
b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1
/2) + (8192*tan(x/2)*(8*a*b^8 - 8*a^3*b^6 + a^5*b^4 + a^7*c^2 - 48*a^2*b^6*
c + 32*a^4*b^4*c - 2*a^6*b^2*c + 72*a^3*b^4*c^2 - 16*a^4*b^2*c^3 - 16*a^5*b
^2*c^2))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)
^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2
 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^
2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2
)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 -
8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5))
)^(1/2)*1i)/(((8192*(4*a^2*b^7 - 3*a^4*b^5 - 20*a^3*b^5*c + 9*a^5*b^3*c + 2
0*a^4*b^3*c^2))/c^4 - ((8192*(4*a*b^7*c^2 - 2*a^2*b^7*c + 2*a^4*b^5*c + 12*
a^5*b*c^4 + 8*a^6*b*c^3 - 24*a^2*b^5*c^3 + 32*a^3*b^3*c^4 + 10*a^3*b^5*c^2
- 10*a^4*b^3*c^3 - 10*a^5*b^3*c^2))/c^4 + ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a
^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^
2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c
 - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2
) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^
4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a
^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*(((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^
3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)
^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*
a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2
*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6
 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^
4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((8192*(3*a*b^5*c^6 + 16*a^3*b*c^8 - 4*a^4*b
*c^7 - 8*a^5*b*c^6 - 16*a^2*b^3*c^7 - 2*a^2*b^5*c^5 + 9*a^3*b^3*c^6 + 2*a^4
*b^3*c^5))/c^4 - ((8192*(3*a*b^5*c^7 - 4*a*b^3*c^9 + 16*a^2*b*c^10 + 20*a^3
*b*c^9 + 12*a^4*b*c^8 - 17*a^2*b^3*c^8 - 3*a^3*b^3*c^7))/c^4 + (8192*tan(x/
2)*(64*a^2*c^11 + 144*a^3*c^10 + 104*a^4*c^9 + 24*a^5*c^8 - 16*a*b^2*c^10 +
 17*a*b^4*c^8 - 2*a*b^6*c^6 - 104*a^2*b^2*c^9 + 18*a^2*b^4*c^7 - 66*a^3*b^2
*c^8 + 2*a^3*b^4*c^6 - 14*a^4*b^2*c^7))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 +
8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c -
 b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^
6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(
1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16
*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6
+ a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(32*a^3*c^9 + 48*a^
4*c^8 + 16*a^5*c^7 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40*a^2*b^2*c^8 + 28*a^2*b^
4*c^6 - 60*a^3*b^2*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2*c^6))/c^4) - (8192*(3*a
*b^7*c^3 - 4*a*b^5*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^5 + 16*a^2*b^3*c^6 - 13*a
^2*b^5*c^4 - 3*a^3*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 + (8192*tan(x/2)*(16*a^4*c
^7 + 24*a^5*c^6 + 10*a^6*c^5 + 16*a*b^4*c^6 - 24*a*b^6*c^4 + 2*a*b^8*c^2 -
```

```
64*a^2*b^2*c^7 + 144*a^2*b^4*c^5 - 18*a^2*b^6*c^3 - 200*a^3*b^2*c^6 + 75*a^
3*b^4*c^4 - 2*a^3*b^6*c^2 - 142*a^4*b^2*c^5 + 14*a^4*b^4*c^3 - 27*a^5*b^2*c
^4))/c^4) - (8192*tan(x/2)*(8*a^5*c^5 + 4*a^6*c^4 - 8*a*b^6*c^3 - 4*a^3*b^6
*c + 40*a^2*b^4*c^4 - 28*a^2*b^6*c^2 - 32*a^3*b^2*c^5 + 60*a^3*b^4*c^3 - 56
*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 + 4*a*b^8*c))/c^4)*((b^8 - a
^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c
 + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*
a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c
*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*
c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*
c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2
)*(8*a*b^8 - 8*a^3*b^6 + a^5*b^4 + a^7*c^2 - 48*a^2*b^6*c + 32*a^4*b^4*c -
2*a^6*b^2*c + 72*a^3*b^4*c^2 - 16*a^4*b^2*c^3 - 16*a^5*b^2*c^2))/c^4)*((b^8
 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b
^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 -
 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b
^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*
a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*
b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) - ((8192*(4
*a^2*b^7 - 3*a^4*b^5 - 20*a^3*b^5*c + 9*a^5*b^3*c + 20*a^4*b^3*c^2))/c^4 +
((8192*(4*a*b^7*c^2 - 2*a^2*b^7*c + 2*a^4*b^5*c + 12*a^5*b*c^4 + 8*a^6*b*c^
3 - 24*a^2*b^5*c^3 + 32*a^3*b^3*c^4 + 10*a^3*b^5*c^2 - 10*a^4*b^3*c^3 - 10*
a^5*b^3*c^2))/c^4 + ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c
- b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b
^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*
c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c
 - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*
c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2
*c^5)))^(1/2)*((8192*(3*a*b^7*c^3 - 4*a*b^5*c^5 + 20*a^4*b*c^6 + 9*a^5*b*c^
5 + 16*a^2*b^3*c^6 - 13*a^2*b^5*c^4 - 3*a^3*b^5*c^3 + 9*a^4*b^3*c^4))/c^4 +
 ((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8
*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2
*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) +
 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(
2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 +
 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*((819
2*(3*a*b^5*c^6 + 16*a^3*b*c^8 - 4*a^4*b*c^7 - 8*a^5*b*c^6 - 16*a^2*b^3*c^7
- 2*a^2*b^5*c^5 + 9*a^3*b^3*c^6 + 2*a^4*b^3*c^5))/c^4 + ((8192*(3*a*b^5*c^7
 - 4*a*b^3*c^9 + 16*a^2*b*c^10 + 20*a^3*b*c^9 + 12*a^4*b*c^8 - 17*a^2*b^3*c
^8 - 3*a^3*b^3*c^7))/c^4 + (8192*tan(x/2)*(64*a^2*c^11 + 144*a^3*c^10 + 104
*a^4*c^9 + 24*a^5*c^8 - 16*a*b^2*c^10 + 17*a*b^4*c^8 - 2*a*b^6*c^6 - 104*a^
2*b^2*c^9 + 18*a^2*b^4*c^7 - 66*a^3*b^2*c^8 + 2*a^3*b^4*c^6 - 14*a^4*b^2*c^
7))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(
1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38
*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)
```

^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(32*a^3*c^9 + 48*a^4*c^8 + 16*a^5*c^7 + 8*a*b^4*c^7 - 4*a*b^6*c^5 - 40*a^2*b^2*c^8 + 28*a^2*b^4*c^6 - 60*a^3*b^2*c^7 + 4*a^3*b^4*c^5 - 20*a^4*b^2*c^6))/c^4) - (8192*tan(x/2)*(16*a^4*c^7 + 24*a^5*c^6 + 10*a^6*c^5 + 16*a*b^4*c^6 - 24*a*b^6*c^4 + 2*a*b^8*c^2 - 64*a^2*b^2*c^7 + 144*a^2*b^4*c^5 - 18*a^2*b^6*c^3 - 200*a^3*b^2*c^6 + 75*a^3*b^4*c^4 - 2*a^3*b^6*c^2 - 142*a^4*b^2*c^5 + 14*a^4*b^4*c^3 - 27*a^5*b^2*c^4))/c^4) - (8192*tan(x/2)*(8*a^5*c^5 + 4*a^6*c^4 - 8*a*b^6*c^3 - 4*a^3*b^6*c + 40*a^2*b^4*c^4 - 28*a^2*b^6*c^2 - 32*a^3*b^2*c^5 + 60*a^3*b^4*c^3 - 56*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 + 4*a*b^8*c))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (8192*tan(x/2)*(8*a*b^8 - 8*a^3*b^6 + a^5*b^4 + a^7*c^2 - 48*a^2*b^6*c + 32*a^4*b^4*c - 2*a^6*b^2*c + 72*a^3*b^4*c^2 - 16*a^4*b^2*c^3 - 16*a^5*b^2*c^2))/c^4)*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2) + (16384*(a^7*b - 4*a^5*b^3))/c^4 + (16384*tan(x/2)*(4*a^6*b^2 - 8*a^4*b^4 + 8*a^5*b^2*c))/c^4))*((b^8 - a^2*b^6 + 8*a^4*c^4 + 8*a^5*c^3 - b^5*(-(4*a*c - b^2)^3)^(1/2) + 8*a^3*b^4*c + a^2*b^3*(-(4*a*c - b^2)^3)^(1/2) + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 18*a^4*b^2*c^2 - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2) - 2*a^3*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^8 + 32*a^3*c^7 + 16*a^4*c^6 + b^4*c^6 - b^6*c^4 - 8*a*b^2*c^7 + 10*a*b^4*c^5 - 32*a^2*b^2*c^6 + a^2*b^4*c^4 - 8*a^3*b^2*c^5)))^(1/2)*2i - (2*b*atan((16384*a*b^9*tan(x/2))/(16384*a*b^9 + 16384*a^3*b^7 - 32768*a^5*b^5 - 131072*a^2*b^7*c - 98304*a^4*b^5*c + 131072*a^6*b^3*c + 16384*a^7*b*c^2 + 262144*a^3*b^5*c^2 + 131072*a^5*b^3*c^2) + (16384*a^7*b*tan(x/2))/(16384*a^7*b + 262144*a^3*b^5 + 131072*a^5*b^3 + (16384*a*b^9)/c^2 - (131072*a^2*b^7)/c - (98304*a^4*b^5)/c + (131072*a^6*b^3)/c + (16384*a^3*b^7)/c^2 - (32768*a^5*b^5)/c^2) - (131072*a^2*b^7*tan(x/2))/(131072*a^6*b^3 - 98304*a^4*b^5 - 131072*a^2*b^7 + 262144*a^3*b^5*c + 131072*a^5*b^3*c + (16384*a*b^9)/c + (16384*a^3*b^7)/c - (32768*a^5*b^5)/c + 16384*a^7*b*c) - (98304*a^4*b^5*tan(x/2))/(131072*a^6*b^3 - 98304*a^4*b^5 - 131072*a^2*b^7 + 262144*a^3*b^5*c + 131072*a^5*b^3*c + (16384*a*b^9)/c + (16384*a^3*b^7)/c - (32768*a^5*b^5)/c + 16384*a^7*b*c) + (131072*a^6*b^3*tan(x/2))/(131072*a^6*b^3 - 98304*a^4*b^5 - 131072*a^2*b^7 + 262144*a^3*b^5*c + 131072*a^5*b^3*c + (16384*a*b^9)/c + (163

```
84*a^3*b^7)/c - (32768*a^5*b^5)/c + 16384*a^7*b*c) + (16384*a^3*b^7*tan(x/2
))/(16384*a*b^9 + 16384*a^3*b^7 - 32768*a^5*b^5 - 131072*a^2*b^7*c - 98304*
a^4*b^5*c + 131072*a^6*b^3*c + 16384*a^7*b*c^2 + 262144*a^3*b^5*c^2 + 13107
2*a^5*b^3*c^2) - (32768*a^5*b^5*tan(x/2))/(16384*a*b^9 + 16384*a^3*b^7 - 32
768*a^5*b^5 - 131072*a^2*b^7*c - 98304*a^4*b^5*c + 131072*a^6*b^3*c + 16384
*a^7*b*c^2 + 262144*a^3*b^5*c^2 + 131072*a^5*b^3*c^2) + (262144*a^3*b^5*tan
(x/2))/(16384*a^7*b + 262144*a^3*b^5 + 131072*a^5*b^3 + (16384*a*b^9)/c^2 -
 (131072*a^2*b^7)/c - (98304*a^4*b^5)/c + (131072*a^6*b^3)/c + (16384*a^3*b
^7)/c^2 - (32768*a^5*b^5)/c^2) + (131072*a^5*b^3*tan(x/2))/(16384*a^7*b + 2
62144*a^3*b^5 + 131072*a^5*b^3 + (16384*a*b^9)/c^2 - (131072*a^2*b^7)/c - (
98304*a^4*b^5)/c + (131072*a^6*b^3)/c + (16384*a^3*b^7)/c^2 - (32768*a^5*b^
5)/c^2)))/c^2
```

3.3 $\int \frac{\sin^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 253

$$\int \frac{\sin^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx=\frac{x}{c}-\frac{\sqrt{2}\left(b-\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}-\frac{\sqrt{2}\left(b+\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$

```
[Out] x/c-arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+
c)-b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(b+(2*a*c-b^2)/(-4*a*c+b^2)^(1/2))/
c/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2)-arctan(1/2*(2*c+(b+(-4*a*c+b^2
)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2))*2^
(1/2)*(b+(-2*a*c+b^2)/(-4*a*c+b^2)^(1/2))/c/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(
1/2))^(1/2)
```

Rubi [A] (verified)

Time = 1.13 (sec) , antiderivative size = 253, normalized size of antiderivative = 1.00, number of steps used = 9, number of rules used = 5, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.263, Rules used

$= \{3337, 3373, 2739, 632, 210\}$

$$\int \frac{\sin^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\sqrt{2}\left(b-\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{\sqrt{2}\left(\frac{b^2-2ac}{\sqrt{b^2-4ac}}+b\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{x}{c}$$

```
[In] Int[Sin[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] x/c - (Sqrt[2]*(b - (b^2 - 2*a*c)/Sqrt[b^2 - 4*a*c])*ArcTan[(2*c + (b - Sqr
t[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*
a*c]])])/(c*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]]) - (Sqrt[2]*(b +
(b^2 - 2*a*c)/Sqrt[b^2 - 4*a*c])*ArcTan[(2*c + (b + Sqrt[b^2 - 4*a*c])*Tan[
x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])])/(c*Sqrt[b^2
 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3337

```
Int[sin[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_), x_Symbol] :> Int[ExpandTr
ig[sin[d + e*x]^m*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n))^p, x], x] /
; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && NeQ[b^2 - 4*a*c, 0] && Integ
ersQ[m, n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rubi steps

$$\text{integral} = \int \left(\frac{1}{c} + \frac{-a-b\sin(x)}{c\left(a+b\sin(x)+c\sin^2(x)\right)}\right) dx$$

$$= \frac{x}{c} + \frac{\int \frac{-a-b\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{c}$$

$$= \frac{x}{c} - \frac{\left(b-\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c} - \frac{\left(b+\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c}$$

$$= \frac{x}{c} - \frac{\left(2\left(b-\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{c} - \frac{\left(2\left(b+\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{c}$$

$$= \frac{x}{c} + \frac{\left(4\left(b-\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx, x, 4c+2\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{c} + \frac{\left(4\left(b+\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx, x, 4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{c}$$

$$= \frac{x}{c} - \frac{\sqrt{2}\left(b-\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{\sqrt{2}\left(b+\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 1.36 (sec) , antiderivative size = 310, normalized size of antiderivative = 1.23

$$\int \frac{\sin^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{x-\frac{\left(ib^2-2iac+b\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}-\frac{\left(-ib^2+2iac+b\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b+i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}}{c}$$

```
[In] Integrate[Sin[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (x - ((I*b^2 - (2*I)*a*c + b*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b - I*Sqrt[
-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 +
4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 +
 4*a*c]]) - (((-I)*b^2 + (2*I)*a*c + b*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b
 + I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sq
rt[-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) + I*b*S
qrt[-b^2 + 4*a*c]]))/c
```

Maple [A] (verified)

Time = 1.61 (sec) , antiderivative size = 252, normalized size of antiderivative = 1.00

method	result
default	$\frac{2a\left(\frac{2\left(-b\sqrt{-4ac+b^2}-4ac+b^2\right)\arctan\left(\frac{2a\tan\left(\frac{x}{2}\right)+b+\sqrt{-4ac+b^2}}{\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}\right)}{\left(8ac-2b^2\right)\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}-\frac{2\left(b\sqrt{-4ac+b^2}-4ac+b^2\right)\arctan\left(\frac{-2a\tan\left(\frac{x}{2}\right)+\sqrt{-4ac+b^2}-b}{\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}\right)}{\left(8ac-2b^2\right)\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}\right)}{c}$
risch	Expression too large to display

```
[In] int(sin(x)^2/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 2/c*a*(2*(-b*(-4*a*c+b^2)^(1/2)-4*a*c+b^2)/(8*a*c-2*b^2)/(4*a*c-2*b^2-2*b*(
-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((2*a*tan(1/2*x)+b+(-4*a*c+b^2)^(1/2))
/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2))-2*(b*(-4*a*c+b^2)^(1/2)-
4*a*c+b^2)/(8*a*c-2*b^2)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*a
rctan((-2*a*tan(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^
(1/2)+4*a^2)^(1/2)))+2/c*arctan(tan(1/2*x))
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 4985 vs. 2(219) = 438.

Time = 0.98 (sec) , antiderivative size = 4985, normalized size of antiderivative = 19.70

$$\int \frac{\sin^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(sin(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] 1/4*(sqrt(2)*c*sqrt((a^2*b^2 - b^4 - 2*a^2*c^2 - 2*(a^3 - 2*a*b^2)*c + (4*a
*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)*s
qrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4
*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^
2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b
^6)*c^4)))/(4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2*b^
2 - b^4)*c^2))*log(8*a^3*b*c^2 + 2*(4*a^3*c^5 + (8*a^4 - a^2*b^2)*c^4 + 2*(
2*a^5 - 3*a^3*b^2)*c^3 - (a^4*b^2 - a^2*b^4)*c^2)*sqrt(-(a^4*b^2 - 2*a^2*b^
4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*
c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 -
3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4))*sin(x) + 4*(a^
4*b - a^2*b^3)*c - sqrt(2)*((8*a^2*c^7 + 6*(4*a^3 - a*b^2)*c^6 + (24*a^4 -
22*a^2*b^2 + b^4)*c^5 + 2*(4*a^5 - 9*a^3*b^2 + 4*a*b^4)*c^4 - (2*a^4*b^2 -
3*a^2*b^4 + b^6)*c^3)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*
(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7
 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (
a^4*b^2 - 2*a^2*b^4 + b^6)*c^4))*cos(x) - (8*a^2*b^2*c^3 + 2*(2*a^3*b^2 - 3
*a*b^4)*c^2 - (a^2*b^4 - b^6)*c)*cos(x))*sqrt((a^2*b^2 - b^4 - 2*a^2*c^2 -
2*(a^3 - 2*a*b^2)*c + (4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^
3 - (a^2*b^2 - b^4)*c^2)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 +
 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*
c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5
- (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4)))/(4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^
3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)) + 2*(a^4*b^2 - a^2*b^4 + 2*a^3*b^2
*c)*sin(x)) - sqrt(2)*c*sqrt((a^2*b^2 - b^4 - 2*a^2*c^2 - 2*(a^3 - 2*a*b^2)
*c - (4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^
4)*c^2)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b
^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 -
11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^
2*b^4 + b^6)*c^4)))/(4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3
- (a^2*b^2 - b^4)*c^2))*log(8*a^3*b*c^2 - 2*(4*a^3*c^5 + (8*a^4 - a^2*b^2)*
c^4 + 2*(2*a^5 - 3*a^3*b^2)*c^3 - (a^4*b^2 - a^2*b^4)*c^2)*sqrt(-(a^4*b^2 -
 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^
2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 +
4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4))*sin(x
```

```
) + 4*(a^4*b - a^2*b^3)*c - sqrt(2)*((8*a^2*c^7 + 6*(4*a^3 - a*b^2)*c^6 + (
24*a^4 - 22*a^2*b^2 + b^4)*c^5 + 2*(4*a^5 - 9*a^3*b^2 + 4*a*b^4)*c^4 - (2*a
^4*b^2 - 3*a^2*b^4 + b^6)*c^3)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2
*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a
*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4
)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4))*cos(x) + (8*a^2*b^2*c^3 + 2*(2*a^
3*b^2 - 3*a*b^4)*c^2 - (a^2*b^4 - b^6)*c)*cos(x))*sqrt((a^2*b^2 - b^4 - 2*a
^2*c^2 - 2*(a^3 - 2*a*b^2)*c - (4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*
a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*
b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3
 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*
b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4)))/(4*a*c^5 + (8*a^2 - b^2)*c^4
 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)) + 2*(a^4*b^2 - a^2*b^4 +
2*a^3*b^2*c)*sin(x)) + sqrt(2)*c*sqrt((a^2*b^2 - b^4 - 2*a^2*c^2 - 2*(a^3 -
 2*a*b^2)*c - (4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2
*b^2 - b^4)*c^2)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*
b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*
(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b
^2 - 2*a^2*b^4 + b^6)*c^4)))/(4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*
b^2)*c^3 - (a^2*b^2 - b^4)*c^2))*log(-8*a^3*b*c^2 + 2*(4*a^3*c^5 + (8*a^4 -
 a^2*b^2)*c^4 + 2*(2*a^5 - 3*a^3*b^2)*c^3 - (a^4*b^2 - a^2*b^4)*c^2)*sqrt(-
(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^
9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b
^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c
^4))*sin(x) - 4*(a^4*b - a^2*b^3)*c - sqrt(2)*((8*a^2*c^7 + 6*(4*a^3 - a*b^
2)*c^6 + (24*a^4 - 22*a^2*b^2 + b^4)*c^5 + 2*(4*a^5 - 9*a^3*b^2 + 4*a*b^4)*
c^4 - (2*a^4*b^2 - 3*a^2*b^4 + b^6)*c^3)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 +
 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*
(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2
 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4))*cos(x) + (8*a^2*b^2*c^3
 + 2*(2*a^3*b^2 - 3*a*b^4)*c^2 - (a^2*b^4 - b^6)*c)*cos(x))*sqrt((a^2*b^2 -
 b^4 - 2*a^2*c^2 - 2*(a^3 - 2*a*b^2)*c - (4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(
2*a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^
6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 +
12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*
b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4)))/(4*a*c^5 + (8*a^2 -
 b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)) - 2*(a^4*b^2 -
a^2*b^4 + 2*a^3*b^2*c)*sin(x)) - sqrt(2)*c*sqrt((a^2*b^2 - b^4 - 2*a^2*c^2
 - 2*(a^3 - 2*a*b^2)*c + (4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*
c^3 - (a^2*b^2 - b^4)*c^2)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2
 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2
)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^
5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4)))/(4*a*c^5 + (8*a^2 - b^2)*c^4 + 2*(2*
a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2))*log(-8*a^3*b*c^2 - 2*(4*a^3*c^5
+ (8*a^4 - a^2*b^2)*c^4 + 2*(2*a^5 - 3*a^3*b^2)*c^3 - (a^4*b^2 - a^2*b^4)*c
```

```
^2)*sqrt(-(a^4*b^2 - 2*a^2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*
c)/(4*a*c^9 + (16*a^2 - b^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a
^2*b^2 + b^4)*c^6 + 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^
4 + b^6)*c^4))*sin(x) - 4*(a^4*b - a^2*b^3)*c - sqrt(2)*((8*a^2*c^7 + 6*(4*
a^3 - a*b^2)*c^6 + (24*a^4 - 22*a^2*b^2 + b^4)*c^5 + 2*(4*a^5 - 9*a^3*b^2 +
 4*a*b^4)*c^4 - (2*a^4*b^2 - 3*a^2*b^4 + b^6)*c^3)*sqrt(-(a^4*b^2 - 2*a^2*b
^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b^2)
*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^5 -
 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4))*cos(x) - (8*a
^2*b^2*c^3 + 2*(2*a^3*b^2 - 3*a*b^4)*c^2 - (a^2*b^4 - b^6)*c)*cos(x))*sqrt(
(a^2*b^2 - b^4 - 2*a^2*c^2 - 2*(a^3 - 2*a*b^2)*c + (4*a*c^5 + (8*a^2 - b^2)
*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)*sqrt(-(a^4*b^2 - 2*a^
2*b^4 + b^6 + 4*a^2*b^2*c^2 + 4*(a^3*b^2 - a*b^4)*c)/(4*a*c^9 + (16*a^2 - b
^2)*c^8 + 12*(2*a^3 - a*b^2)*c^7 + 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^6 + 4*(a^
5 - 3*a^3*b^2 + 2*a*b^4)*c^5 - (a^4*b^2 - 2*a^2*b^4 + b^6)*c^4)))/(4*a*c^5
+ (8*a^2 - b^2)*c^4 + 2*(2*a^3 - 3*a*b^2)*c^3 - (a^2*b^2 - b^4)*c^2)) - 2*(
a^4*b^2 - a^2*b^4 + 2*a^3*b^2*c)*sin(x)) + 4*x)/c
```

Sympy [F(-1)]

Timed out.

$$\int \frac{\sin^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(sin(x)**2/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Timed out
```

Maxima [F]

$$\int \frac{\sin^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\sin\left(x\right)^2}{c\sin\left(x\right)^2+b\sin\left(x\right)+a}\,dx$$

```
[In] integrate(sin(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] -(c*integrate(2*(2*b^2*cos(3*x)^2 + 2*b^2*cos(x)^2 + 2*b^2*sin(3*x)^2 + 2*b
^2*sin(x)^2 + 4*(2*a^2 + a*c)*cos(2*x)^2 + 2*(4*a*b + b*c)*cos(x)*sin(2*x)
+ 4*(2*a^2 + a*c)*sin(2*x)^2 + b*c*sin(x) - (2*a*c*cos(2*x) + b*c*sin(3*x)
- b*c*sin(x))*cos(4*x) - 2*(2*b^2*cos(x) + (4*a*b + b*c)*sin(2*x))*cos(3*x)
 - 2*(a*c + (4*a*b + b*c)*sin(x))*cos(2*x) + (b*c*cos(3*x) - b*c*cos(x) - 2
*a*c*sin(2*x))*sin(4*x) - (4*b^2*sin(x) + b*c - 2*(4*a*b + b*c)*cos(2*x))*s
in(3*x))/(c^3*cos(4*x)^2 + 4*b^2*c*cos(3*x)^2 + 4*b^2*c*cos(x)^2 + c^3*sin(
4*x)^2 + 4*b^2*c*sin(3*x)^2 + 4*b^2*c*sin(x)^2 + 4*b*c^2*sin(x) + c^3 + 4*(
```

```
4*a^2*c + 4*a*c^2 + c^3)*cos(2*x)^2 + 8*(2*a*b*c + b*c^2)*cos(x)*sin(2*x) +
 4*(4*a^2*c + 4*a*c^2 + c^3)*sin(2*x)^2 - 2*(2*b*c^2*sin(3*x) - 2*b*c^2*sin
(x) - c^3 + 2*(2*a*c^2 + c^3)*cos(2*x))*cos(4*x) - 8*(b^2*c*cos(x) + (2*a*b
*c + b*c^2)*sin(2*x))*cos(3*x) - 4*(2*a*c^2 + c^3 + 2*(2*a*b*c + b*c^2)*sin
(x))*cos(2*x) + 4*(b*c^2*cos(3*x) - b*c^2*cos(x) - (2*a*c^2 + c^3)*sin(2*x)
)*sin(4*x) - 4*(2*b^2*c*sin(x) + b*c^2 - 2*(2*a*b*c + b*c^2)*cos(2*x))*sin(
3*x)), x) - x)/c
```

Giac [F(-1)]

Timed out.

$$\int \frac{\sin^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(sin(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 28.23 (sec) , antiderivative size = 15461, normalized size of antiderivative = 61.11

$$\int \frac{\sin^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(sin(x)^2/(a + c*sin(x)^2 + b*sin(x)),x)
```

```
[Out] (2*atan((147456*a^5*tan(x/2))/(16384*a*b^4 + 393216*a^4*c + 147456*a^5 - 22
9376*a^3*b^2 + 262144*a^3*c^2 - 131072*a^2*b^2*c + (32768*a^2*b^4)/c - (327
68*a^4*b^2)/c) + (393216*a^4*tan(x/2))/(262144*a^3*c + 393216*a^4 - 131072*
a^2*b^2 + (147456*a^5)/c + (16384*a*b^4)/c - (229376*a^3*b^2)/c + (32768*a^
2*b^4)/c^2 - (32768*a^4*b^2)/c^2) + (16384*a*b^4*tan(x/2))/(16384*a*b^4 + 3
93216*a^4*c + 147456*a^5 - 229376*a^3*b^2 + 262144*a^3*c^2 - 131072*a^2*b^2
*c + (32768*a^2*b^4)/c - (32768*a^4*b^2)/c) + (262144*a^3*c*tan(x/2))/(2621
44*a^3*c + 393216*a^4 - 131072*a^2*b^2 + (147456*a^5)/c + (16384*a*b^4)/c -
 (229376*a^3*b^2)/c + (32768*a^2*b^4)/c^2 - (32768*a^4*b^2)/c^2) - (229376*
a^3*b^2*tan(x/2))/(16384*a*b^4 + 393216*a^4*c + 147456*a^5 - 229376*a^3*b^2
 + 262144*a^3*c^2 - 131072*a^2*b^2*c + (32768*a^2*b^4)/c - (32768*a^4*b^2)/
c) - (131072*a^2*b^2*tan(x/2))/(262144*a^3*c + 393216*a^4 - 131072*a^2*b^2
+ (147456*a^5)/c + (16384*a*b^4)/c - (229376*a^3*b^2)/c + (32768*a^2*b^4)/c
^2 - (32768*a^4*b^2)/c^2) + (32768*a^2*b^4*tan(x/2))/(147456*a^5*c + 32768*
a^2*b^4 - 32768*a^4*b^2 + 262144*a^3*c^3 + 393216*a^4*c^2 - 229376*a^3*b^2*
c - 131072*a^2*b^2*c^2 + 16384*a*b^4*c) - (32768*a^4*b^2*tan(x/2))/(147456*
a^5*c + 32768*a^2*b^4 - 32768*a^4*b^2 + 262144*a^3*c^3 + 393216*a^4*c^2 - 2
29376*a^3*b^2*c - 131072*a^2*b^2*c^2 + 16384*a*b^4*c)))/c - atan((((b^6 - a
```

```
^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*
a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-
(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4
- b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a
^3*b^2*c^3)))^(1/2)*(tan(x/2)*(65536*a*b^4 + 131072*a^4*c + 24576*a^5 - 655
36*a^3*b^2 + 131072*a^3*c^2 - 262144*a^2*b^2*c) + ((b^6 - a^2*b^4 - 8*a^3*c
^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1
/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)
^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*
b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1
/2)*(tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 - 65536*a*b^3*c^2 + 262144*a^2*b
*c^3 - 196608*a^2*b^3*c + 196608*a^3*b*c^2 + 131072*a^4*b*c) + 24576*a^5*c
+ 8192*a^2*b^4 - 8192*a^4*b^2 - 131072*a^3*c^3 - 131072*a^4*c^2 + ((b^6 - a
^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*
a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-
(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4
- b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a
^3*b^2*c^3)))^(1/2)*(tan(x/2)*(16384*a^3*b^4 - 16384*a*b^6 + 524288*a^2*c^5
 + 1179648*a^3*c^4 + 786432*a^4*c^3 + 147456*a^5*c^2 - 131072*a*b^2*c^4 + 1
96608*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1048576*a^2*b^2*c^3
- 491520*a^3*b^2*c^2) + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*
a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2
*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 3
2*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 3
2*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(((b^6 - a^2*b^4 - 8*a
^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3
)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2
)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 -
8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3))
)^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 1966
08*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968
*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2
- 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^
6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3
*c^3) + tan(x/2)*(32768*a*b^5*c^2 - 65536*a*b^3*c^4 + 262144*a^2*b*c^5 + 26
2144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2)
 + 98304*a^4*c^4 + 98304*a^5*c^3 - 24576*a*b^4*c^3 + 98304*a^2*b^2*c^4 + 24
576*a^2*b^4*c^2 - 122880*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) - 32768*a*b^3*c^3
 + 131072*a^2*b*c^4 + 65536*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 -
 106496*a^2*b^3*c^2 + 24576*a*b^5*c) - 8192*a^3*b^2*c + 163840*a^2*b^2*c^2
- 32768*a*b^4*c) - 24576*a^4*b + 32768*a^2*b^3 - 98304*a^3*b*c)*1i + ((b^6
- a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-
(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c
*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c
^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 -
```

8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(65536*a*b^4 + 131072*a^4*c + 24576*a^5 - 65536*a^3*b^2 + 131072*a^3*c^2 - 262144*a^2*b^2*c) + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(8192*a^4*b^2 - 24576*a^5*c - 8192*a^2*b^4 - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 - 65536*a*b^3*c^2 + 262144*a^2*b*c^3 - 196608*a^2*b^3*c + 196608*a^3*b*c^2 + 131072*a^4*b*c) + 131072*a^3*c^3 + 131072*a^4*c^2 + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(16384*a^3*b^4 - 16384*a*b^6 + 524288*a^2*c^5 + 1179648*a^3*c^4 + 786432*a^4*c^3 + 147456*a^5*c^2 - 131072*a*b^2*c^4 + 196608*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1048576*a^2*b^2*c^3 - 491520*a^3*b^2*c^2) - ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(32768*a*b^5*c^2 - 65536*a*b^3*c^4 + 262144*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) - ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) + 98304*a^4*c^4 + 98304*a^5*c^3 - 24576*a*b^4*c^3 + 98304*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 122880*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) - 32768*a*b^3*c^3 + 131072*a^2*b*c^4 + 65536*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 24576*a*b^5*c) + 8192*a^3*b^2*c - 163840*a^2*b^2*c^2 + 32768*a*b^4*c) - 24576*a^4*b + 32768*a^2*b^3 - 98304*a^3*b*c)*1i)/(65536*a^4 - ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(65536*a*b^4 + 131072*a^4*c + 24576*a^5 - 65536*a^3*b^2 + 131072*a^3*c^2 - 262144*a^2*b^2*c) + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*

(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^
4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8
*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 - 65536*a*b^3*
c^2 + 262144*a^2*b*c^3 - 196608*a^2*b^3*c + 196608*a^3*b*c^2 + 131072*a^4*b
*c) + 24576*a^5*c + 8192*a^2*b^4 - 8192*a^4*b^2 - 131072*a^3*c^3 - 131072*a
^4*c^2 + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(
1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*
b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*
a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 +
 a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(16384*a^3*b^4 - 16384*a*b^
6 + 524288*a^2*c^5 + 1179648*a^3*c^4 + 786432*a^4*c^3 + 147456*a^5*c^2 - 13
1072*a*b^2*c^4 + 196608*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 10
48576*a^2*b^2*c^3 - 491520*a^3*b^2*c^2) + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a
^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*
a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/
(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5
 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(((b
^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b
*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*
b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^
4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2
 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851
968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*
a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 +
16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3
 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c
^4 - 24576*a^3*b^3*c^3) + tan(x/2)*(32768*a*b^5*c^2 - 65536*a*b^3*c^4 + 262
144*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 -
32768*a^3*b^3*c^2) + 98304*a^4*c^4 + 98304*a^5*c^3 - 24576*a*b^4*c^3 + 9830
4*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 122880*a^3*b^2*c^3 - 24576*a^4*b^2*c^2)
 - 32768*a*b^3*c^3 + 131072*a^2*b*c^4 + 65536*a^3*b*c^3 - 24576*a^3*b^3*c +
 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 24576*a*b^5*c) - 8192*a^3*b^2*c + 1
63840*a^2*b^2*c^2 - 32768*a*b^4*c) - 24576*a^4*b + 32768*a^2*b^3 - 98304*a^
3*b*c) + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(
1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*
b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*
a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 +
 a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(65536*a*b^4 + 131072*a^4*c
 + 24576*a^5 - 65536*a^3*b^2 + 131072*a^3*c^2 - 262144*a^2*b^2*c) + ((b^6 -
 a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(
4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*
(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^
4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8
*a^3*b^2*c^3)))^(1/2)*(8192*a^4*b^2 - 24576*a^5*c - 8192*a^2*b^4 - tan(x/2)
*(32768*a*b^5 - 32768*a^3*b^3 - 65536*a*b^3*c^2 + 262144*a^2*b*c^3 - 196608

*a^2*b^3*c + 196608*a^3*b*c^2 + 131072*a^4*b*c) + 131072*a^3*c^3 + 131072*a
^4*c^2 + ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(
1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*
b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*
a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 +
 a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(16384*a^3*b^4 - 16384*a*b^
6 + 524288*a^2*c^5 + 1179648*a^3*c^4 + 786432*a^4*c^3 + 147456*a^5*c^2 - 13
1072*a*b^2*c^4 + 196608*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 10
48576*a^2*b^2*c^3 - 491520*a^3*b^2*c^2) - ((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a
^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*
a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/
(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5
 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan
(x/2)*(32768*a*b^5*c^2 - 65536*a*b^3*c^4 + 262144*a^2*b*c^5 + 262144*a^3*b*
c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) - ((b^6 -
a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4
*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(
-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4
 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*
a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a
^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6
*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384
*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 13
1072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 -
24576*a^3*b^3*c^3) + 98304*a^4*c^4 + 98304*a^5*c^3 - 24576*a*b^4*c^3 + 9830
4*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 122880*a^3*b^2*c^3 - 24576*a^4*b^2*c^2)
 - 32768*a*b^3*c^3 + 131072*a^2*b*c^4 + 65536*a^3*b*c^3 - 24576*a^3*b^3*c +
 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 24576*a*b^5*c) + 8192*a^3*b^2*c - 1
63840*a^2*b^2*c^2 + 32768*a*b^4*c) - 24576*a^4*b + 32768*a^2*b^3 - 98304*a^
3*b*c) + 131072*a^3*b*tan(x/2)))*((b^6 - a^2*b^4 - 8*a^3*c^3 - 8*a^4*c^2 -
b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) + 6*a^3*b^2*c
 + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^
2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^
4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*2i - atan(((-
(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2
b(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*
a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 +
b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c
^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(65536*a*b^4 + 131072*a^4*c + 24576*a
^5 - 65536*a^3*b^2 + 131072*a^3*c^2 - 262144*a^2*b^2*c) - 24576*a^4*b + (-(
a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*
b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a
*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b
^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^
2 - 8*a^3*b^2*c^3)))^(1/2)*((-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*

```
(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 1
8*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^
6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^
3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(16384*
a^3*b^4 - 16384*a*b^6 + 524288*a^2*c^5 + 1179648*a^3*c^4 + 786432*a^4*c^3 +
 147456*a^5*c^2 - 131072*a*b^2*c^4 + 196608*a*b^4*c^2 + 131072*a^2*b^4*c -
98304*a^4*b^2*c - 1048576*a^2*b^2*c^3 - 491520*a^3*b^2*c^2) + (-(a^2*b^4 -
b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c
 - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*
a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b
^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*
b^2*c^3)))^(1/2)*(tan(x/2)*(32768*a*b^5*c^2 - 65536*a*b^3*c^4 + 262144*a^2*
b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^
3*b^3*c^2) + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)
^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 +
 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5
+ 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*
c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179
648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a
*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 5406
72*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5
+ 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 -
 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) + 98304*a^4*c^4 + 98304*a^5*c^3 -
24576*a*b^4*c^3 + 98304*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 122880*a^3*b^2*c^
3 - 24576*a^4*b^2*c^2) - 32768*a*b^3*c^3 + 131072*a^2*b*c^4 + 65536*a^3*b*c
^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 24576*a*b^5*c
) + tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 - 65536*a*b^3*c^2 + 262144*a^2*b*
c^3 - 196608*a^2*b^3*c + 196608*a^3*b*c^2 + 131072*a^4*b*c) + 24576*a^5*c +
 8192*a^2*b^4 - 8192*a^4*b^2 - 131072*a^3*c^3 - 131072*a^4*c^2 - 8192*a^3*b
^2*c + 163840*a^2*b^2*c^2 - 32768*a*b^4*c) + 32768*a^2*b^3 - 98304*a^3*b*c)
*1i + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/
2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^
4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^
4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a
^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(65536*a*b^4 + 131072*a^4*c +
 24576*a^5 - 65536*a^3*b^2 + 131072*a^3*c^2 - 262144*a^2*b^2*c) - 24576*a^4
*b + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2
) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4
*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4
*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^
2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*((-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^
2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b
^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(1
6*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*
a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)
```

*(16384*a^3*b^4 - 16384*a*b^6 + 524288*a^2*c^5 + 1179648*a^3*c^4 + 786432*a
^4*c^3 + 147456*a^5*c^2 - 131072*a*b^2*c^4 + 196608*a*b^4*c^2 + 131072*a^2*
b^4*c - 98304*a^4*b^2*c - 1048576*a^2*b^2*c^3 - 491520*a^3*b^2*c^2) - (-(a^
2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*
(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b
c(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4
*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2
- 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(32768*a*b^5*c^2 - 65536*a*b^3*c^4 + 262
144*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 -
32768*a^3*b^3*c^2) - (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*
c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b
^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*
a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*
a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^
7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 +
139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^
3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*
b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4
*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) + 98304*a^4*c^4 + 98304*a^
5*c^3 - 24576*a*b^4*c^3 + 98304*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 122880*a^
3*b^2*c^3 - 24576*a^4*b^2*c^2) - 32768*a*b^3*c^3 + 131072*a^2*b*c^4 + 65536
*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 24576
*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 - 65536*a*b^3*c^2 + 26214
4*a^2*b*c^3 - 196608*a^2*b^3*c + 196608*a^3*b*c^2 + 131072*a^4*b*c) - 24576
*a^5*c - 8192*a^2*b^4 + 8192*a^4*b^2 + 131072*a^3*c^3 + 131072*a^4*c^2 + 81
92*a^3*b^2*c - 163840*a^2*b^2*c^2 + 32768*a*b^4*c) + 32768*a^2*b^3 - 98304*
a^3*b*c)*1i)/(65536*a^4 - (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-
(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*
a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6
+ 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3
- 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(65536*a*
b^4 + 131072*a^4*c + 24576*a^5 - 65536*a^3*b^2 + 131072*a^3*c^2 - 262144*a^
2*b^2*c) - 24576*a^4*b + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(
4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a
^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 +
32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 -
32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*((-(a^2*b^4 - b^6 +
8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2
)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c -
b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2
- 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^
3)))^(1/2)*(tan(x/2)*(16384*a^3*b^4 - 16384*a*b^6 + 524288*a^2*c^5 + 117964
8*a^3*c^4 + 786432*a^4*c^3 + 147456*a^5*c^2 - 131072*a*b^2*c^4 + 196608*a*b
^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1048576*a^2*b^2*c^3 - 491520*
a^3*b^2*c^2) + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^

```
2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2
 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^
5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^
2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(32768*a*b^5*c^2 - 6
5536*a*b^3*c^4 + 262144*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 1
96608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a
^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*
a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/
(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5
+ 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan
(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c^4 -
 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2*c^5
 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4
*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163840*a
^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) + 9830
4*a^4*c^4 + 98304*a^5*c^3 - 24576*a*b^4*c^3 + 98304*a^2*b^2*c^4 + 24576*a^2
*b^4*c^2 - 122880*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) - 32768*a*b^3*c^3 + 1310
72*a^2*b*c^4 + 65536*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496
*a^2*b^3*c^2 + 24576*a*b^5*c) + tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 - 655
36*a*b^3*c^2 + 262144*a^2*b*c^3 - 196608*a^2*b^3*c + 196608*a^3*b*c^2 + 131
072*a^4*b*c) + 24576*a^5*c + 8192*a^2*b^4 - 8192*a^4*b^2 - 131072*a^3*c^3 -
 131072*a^4*c^2 - 8192*a^3*b^2*c + 163840*a^2*b^2*c^2 - 32768*a*b^4*c) + 32
768*a^2*b^3 - 98304*a^3*b*c) + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b
^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c
- 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2
*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4
*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(655
36*a*b^4 + 131072*a^4*c + 24576*a^5 - 65536*a^3*b^2 + 131072*a^3*c^2 - 2621
44*a^2*b^2*c) - 24576*a^4*b + (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^
3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c -
 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*
c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*
c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*((-(a^2*b^4 - b
^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c
- b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a
*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^
6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b
^2*c^3)))^(1/2)*(tan(x/2)*(16384*a^3*b^4 - 16384*a*b^6 + 524288*a^2*c^5 + 1
179648*a^3*c^4 + 786432*a^4*c^3 + 147456*a^5*c^2 - 131072*a*b^2*c^4 + 19660
8*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1048576*a^2*b^2*c^3 - 49
1520*a^3*b^2*c^2) - (-(a^2*b^4 - b^6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c
 - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^
2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a
^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a
^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)*(tan(x/2)*(32768*a*b^5*c^
```

```
2 - 65536*a*b^3*c^4 + 262144*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^
3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) - (-(a^2*b^4 - b^6 + 8*a^3*c^3
+ 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c - b^2)^3)^(1/2)
 - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1
/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6*c^2 - 8*a*b^2
*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^2*c^3)))^(1/2)
*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*
c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^
2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 11468
8*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163
840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) +
 98304*a^4*c^4 + 98304*a^5*c^3 - 24576*a*b^4*c^3 + 98304*a^2*b^2*c^4 + 2457
6*a^2*b^4*c^2 - 122880*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) - 32768*a*b^3*c^3 +
 131072*a^2*b*c^4 + 65536*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 1
06496*a^2*b^3*c^2 + 24576*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3
- 65536*a*b^3*c^2 + 262144*a^2*b*c^3 - 196608*a^2*b^3*c + 196608*a^3*b*c^2
+ 131072*a^4*b*c) - 24576*a^5*c - 8192*a^2*b^4 + 8192*a^4*b^2 + 131072*a^3*
c^3 + 131072*a^4*c^2 + 8192*a^3*b^2*c - 163840*a^2*b^2*c^2 + 32768*a*b^4*c)
 + 32768*a^2*b^3 - 98304*a^3*b*c) + 131072*a^3*b*tan(x/2)))*(-(a^2*b^4 - b^
6 + 8*a^3*c^3 + 8*a^4*c^2 - b^3*(-(4*a*c - b^2)^3)^(1/2) + a^2*b*(-(4*a*c -
 b^2)^3)^(1/2) - 6*a^3*b^2*c - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*
c - b^2)^3)^(1/2))/(2*(16*a^2*c^6 + 32*a^3*c^5 + 16*a^4*c^4 + b^4*c^4 - b^6
*c^2 - 8*a*b^2*c^5 + 10*a*b^4*c^3 - 32*a^2*b^2*c^4 + a^2*b^4*c^2 - 8*a^3*b^
2*c^3)))^(1/2)*2i
```

3.4 $\int \frac{\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 17, antiderivative size = 226

$$\int \frac{\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\sqrt{2}\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} + \frac{\sqrt{2}\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$

```
[Out] arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)-b
*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(1-b/(-4*a*c+b^2)^(1/2))/(b^2-2*c*(a+c)
-b*(-4*a*c+b^2)^(1/2))^(1/2)+arctan(1/2*(2*c+(b+(-4*a*c+b^2)^(1/2))*tan(1/2
*x))*2^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(1+b/(-4*a
*c+b^2)^(1/2))/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2)
```

Rubi [A] (verified)

Time = 0.58 (sec) , antiderivative size = 226, normalized size of antiderivative = 1.00, number of steps used = 8, number of rules used = 4, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.235, Rules used = {3337, 2739, 632, 210}

$$\int \frac{\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\sqrt{2}\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{\sqrt{2}\left(\frac{b}{\sqrt{b^2-4ac}}+1\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}$$

```
[In] Int[Sin[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (Sqrt[2]*(1 - b/Sqrt[b^2 - 4*a*c])*ArcTan[(2*c + (b - Sqrt[b^2 - 4*a*c])*Ta
n[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]])])/Sqrt[b^2
- 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]] + (Sqrt[2]*(1 + b/Sqrt[b^2 - 4*a*c])*A
rcTan[(2*c + (b + Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a +
 c) + b*Sqrt[b^2 - 4*a*c]])])/Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]]
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3337

```
Int[sin[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_), x_Symbol] :> Int[ExpandTr
ig[sin[d + e*x]^m*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n))^p, x], x] /
```

```
; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && NeQ[b^2 - 4*a*c, 0] && Integ
ersQ[m, n, p]
```

Rubi steps

$$\text{integral} = \int \left(\frac{1-\frac{b}{\sqrt{b^2-4ac}}}{b-\sqrt{b^2-4ac}+2c\sin(x)}+\frac{1+\frac{b}{\sqrt{b^2-4ac}}}{b+\sqrt{b^2-4ac}+2c\sin(x)}\right)dx$$

$$=\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx+\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx$$

$$=\left(2\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)+\left(2\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)$$

$$=-\left(\left(4\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx,x,4c+2\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)\right)-\left(4\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx,x,4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)$$

$$=\frac{\sqrt{2}\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}+\frac{\sqrt{2}\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 1.37 (sec) , antiderivative size = 268, normalized size of antiderivative = 1.19

$$\int \frac{\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\frac{\left(ib+\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}+\frac{\left(-ib+\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b+i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}}{\sqrt{-\frac{b^2}{2}+2ac}}$$

```
[In] Integrate[Sin[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (((I*b + Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b - I*Sqrt[-b^2 + 4*a*c])*Tan[x
/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]])])/Sqrt[b^2
- 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]] + (((-I)*b + Sqrt[-b^2 + 4*a*c])*Ar
cTan[(2*c + (b + I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a
 + c) + I*b*Sqrt[-b^2 + 4*a*c]])])/Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 +
 4*a*c]])/Sqrt[-1/2*b^2 + 2*a*c]
```

Maple [A] (verified)

Time = 1.04 (sec) , antiderivative size = 217, normalized size of antiderivative = 0.96

method	result
default	$4a\left(\frac{2\sqrt{-4ac+b^2}\ \arctan\left(\frac{-2a\tan\left(\frac{x}{2}\right)+\sqrt{-4ac+b^2}-b}{\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}\right)}{(8ac-2b^2)\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}+\frac{2\sqrt{-4ac+b^2}\ \arctan\left(\frac{2a\tan\left(\frac{x}{2}\right)+b+\sqrt{-4ac+b^2}}{\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}\right)}{(8ac-2b^2)\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}\right)$
risch	$-\frac{i\left(\sum_{\_R=\operatorname{RootOf}\left(\left(16a^4c^2-8a^3b^2c+32a^3c^3+a^2b^4-32a^2b^2c^2+16a^2c^4+10a\,b^4c-8a\,b^2c^3-b^6+b^4c^2\right)\_Z^4+\left(32a^3c-8a^2b^2+32a^2c^2-24a\,b^2c+4b^4\right)\_\right.}\right.}{}$

```
[In] int(sin(x)/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 4*a*(2*(-4*a*c+b^2)^(1/2)/(8*a*c-2*b^2)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)
+4*a^2)^(1/2)*arctan((-2*a*tan(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4*a*c-2*b^2+2*
b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2))+2*(-4*a*c+b^2)^(1/2)/(8*a*c-2*b^2)/(4*a*
c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((2*a*tan(1/2*x)+b+(-4*a*
c+b^2)^(1/2))/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)))
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 3519 vs. 2(192) = 384.

Time = 0.57 (sec) , antiderivative size = 3519, normalized size of antiderivative = 15.57

$$\int \frac{\sin(x)}{a + b\sin(x) + c\sin^2(x)} \, dx = \text{Too large to display}$$

```
[In] integrate(sin(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] -1/4*sqrt(2)*sqrt(-(2*a^2 - b^2 + 2*a*c - (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2
 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 -
4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b
^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3
 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))*log(2*a*b^2*sin(x) + 4*a*b*c
 + 2*(a^3*b^2 - a*b^4 - 4*a^2*c^3 - (8*a^3 - a*b^2)*c^2 - 2*(2*a^4 - 3*a^2*
b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4
- 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^
3*b^2 + 2*a*b^4)*c))*sin(x) - sqrt(2)*((a^3*b^3 - a*b^5 + 4*a*b*c^4 + (4*a^
2*b - b^3)*c^3 - (4*a^3*b + 5*a*b^3)*c^2 - (4*a^4*b - 5*a^2*b^3 - b^5)*c)*s
qrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a
^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2
*a*b^4)*c))*cos(x) + (a*b^3 - 4*a*b*c^2 - (4*a^2*b - b^3)*c)*cos(x))*sqrt(-
(2*a^2 - b^2 + 2*a*c - (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*
a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 -
 b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(
a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^
2 - 2*(2*a^3 - 3*a*b^2)*c))) + 1/4*sqrt(2)*sqrt(-(2*a^2 - b^2 + 2*a*c + (a^
2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2
/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*
b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)
*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)
)*log(2*a*b^2*sin(x) + 4*a*b*c - 2*(a^3*b^2 - a*b^4 - 4*a^2*c^3 - (8*a^3 -
a*b^2)*c^2 - 2*(2*a^4 - 3*a^2*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 -
 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*
b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*sin(x) - sqrt(2)*((a^3*b
^3 - a*b^5 + 4*a*b*c^4 + (4*a^2*b - b^3)*c^3 - (4*a^3*b + 5*a*b^3)*c^2 - (4
*a^4*b - 5*a^2*b^3 - b^5)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5
- (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4
)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*cos(x) - (a*b^3 - 4*a*b*c^2 - (4*
a^2*b - b^3)*c)*cos(x))*sqrt(-(2*a^2 - b^2 + 2*a*c + (a^2*b^2 - b^4 - 4*a*c
^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b
^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4
 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^
4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))) - 1/4*sqrt(2)*sq
```

```
rt(-(2*a^2 - b^2 + 2*a*c + (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2
*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a
^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 -
 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2
)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))*log(-2*a*b^2*sin(x) - 4*a*b*c + 2*(a^3*b^2
- a*b^4 - 4*a^2*c^3 - (8*a^3 - a*b^2)*c^2 - 2*(2*a^4 - 3*a^2*b^2)*c)*sqrt(b
^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 -
a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^
4)*c))*sin(x) - sqrt(2)*((a^3*b^3 - a*b^5 + 4*a*b*c^4 + (4*a^2*b - b^3)*c^3
 - (4*a^3*b + 5*a*b^3)*c^2 - (4*a^4*b - 5*a^2*b^3 - b^5)*c)*sqrt(b^2/(a^4*b
^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^
3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*co
s(x) - (a*b^3 - 4*a*b*c^2 - (4*a^2*b - b^3)*c)*cos(x))*sqrt(-(2*a^2 - b^2 +
 2*a*c + (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)
*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12
*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^
2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 -
 3*a*b^2)*c))) + 1/4*sqrt(2)*sqrt(-(2*a^2 - b^2 + 2*a*c - (a^2*b^2 - b^4 -
4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*
a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(
8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2
 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))*log(-2*a*b^2
*sin(x) - 4*a*b*c - 2*(a^3*b^2 - a*b^4 - 4*a^2*c^3 - (8*a^3 - a*b^2)*c^2 -
2*(2*a^4 - 3*a^2*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (1
6*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^
2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*sin(x) - sqrt(2)*((a^3*b^3 - a*b^5 +
4*a*b*c^4 + (4*a^2*b - b^3)*c^3 - (4*a^3*b + 5*a*b^3)*c^2 - (4*a^4*b - 5*a^
2*b^3 - b^5)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b
^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^
5 - 3*a^3*b^2 + 2*a*b^4)*c))*cos(x) + (a*b^3 - 4*a*b*c^2 - (4*a^2*b - b^3)*
c)*cos(x))*sqrt(-(2*a^2 - b^2 + 2*a*c - (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 -
 b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*
a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2
 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 -
 (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)))
```

Sympy [F(-1)]

Timed out.

$$\int \frac{\sin(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(sin(x)/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Timed out
```

Maxima [F]

$$\int \frac{\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\sin{(x)}}{c\sin{(x)}^2+b\sin{(x)}+a}\,dx$$

```
[In] integrate(sin(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
[Out] integrate(sin(x)/(c*sin(x)^2 + b*sin(x) + a), x)
```

Giac [F(-1)]

Timed out.

$$\int \frac{\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(sin(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 24.57 (sec) , antiderivative size = 5048, normalized size of antiderivative = 22.34

$$\int \frac{\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(sin(x)/(a + c*sin(x)^2 + b*sin(x)),x)
[Out] atan(-((((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^
2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b
^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(
tan(x/2)*(256*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 - 64*a*b^2*c) - 32*a*b^3 + (
(8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b
^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 -
8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*
(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*
c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*
c) + 128*a^2*b*c) - tan(x/2)*(128*a^2*c - 64*a*b^2 + 64*a^3) + 32*a^2*b)*((
8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^
2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8
*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*1i - (((8*a
^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c
)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*
b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(256
```

*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 - 64*a*b^2*c) - 32*a*b^3 - ((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 128*a^2*b*c) + tan(x/2)*(128*a^2*c - 64*a*b^2 + 64*a^3) - 32*a^2*b)*((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*1i)/((((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(256*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 - 64*a*b^2*c) - 32*a*b^3 + ((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 128*a^2*b*c) - tan(x/2)*(128*a^2*c - 64*a*b^2 + 64*a^3) + 32*a^2*b)*((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2) - 128*a^2*tan(x/2) + (((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(256*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 - 64*a*b^2*c) - 32*a*b^3 - ((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 128*a^2*b*c) + tan(x/2)*(128*a^2*c - 64*a*b^2 + 64*a^3) - 32*a^2*b)*((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)))*((8*a^3*c + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*2i + atan(-((((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(256*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 - 64*a*b^2*c) - 32*a*b^3 + ((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b

```
^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 +
 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 +
128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 128*a^2*b*c) - tan(x/2)*(128*a^
2*c - 64*a*b^2 + 64*a^3) + 32*a^2*b)*((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2)
 + b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4
+ 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^
2*c^2 + 10*a*b^4*c)))^(1/2)*1i - (((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) +
b^4 - 2*a^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 3
2*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c
^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(256*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 -
64*a*b^2*c) - 32*a*b^3 - ((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a
^2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3
 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a
*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3
+ 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2
 - 32*a*b^3*c - 128*a^3*b*c) + 128*a^2*b*c) + tan(x/2)*(128*a^2*c - 64*a*b^
2 + 64*a^3) - 32*a^2*b)*((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^
2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3
+ 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*
b^4*c)))^(1/2)*1i)/((((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b
^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 1
6*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4
*c)))^(1/2)*(tan(x/2)*(256*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 - 64*a*b^2*c) -
 32*a*b^3 + ((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^
2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2
 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/
2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2
 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c
 - 128*a^3*b*c) + 128*a^2*b*c) - tan(x/2)*(128*a^2*c - 64*a*b^2 + 64*a^3) +
 32*a^2*b)*((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2
*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2
+ b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2
) - 128*a^2*tan(x/2) + (((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^
2*b^2 + 8*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3
+ 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*
b^4*c)))^(1/2)*(tan(x/2)*(256*a^3*c - 64*a^2*b^2 + 256*a^2*c^2 - 64*a*b^2*c
) - 32*a*b^3 - ((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8
*a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*
c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^
(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*
c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^
3*c - 128*a^3*b*c) + 128*a^2*b*c) + tan(x/2)*(128*a^2*c - 64*a*b^2 + 64*a^3
) - 32*a^2*b)*((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*
a^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c
^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(
```

```
1/2)))*((8*a^3*c - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 - 2*a^2*b^2 + 8*a^2*c^2
 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^
4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*2i
```

3.5 $\int \frac{1}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 14, antiderivative size = 221

$$\int \frac{1}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{2\sqrt{2}c\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{2\sqrt{2}c\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$

```
[Out] 2*c*arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+
c)-b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)/(-4*a*c+b^2)^(1/2)/(b^2-2*c*(a+c)-b
*(-4*a*c+b^2)^(1/2))^(1/2)-2*c*arctan(1/2*(2*c+(b+(-4*a*c+b^2)^(1/2))*tan(1
/2*x))*2^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)/(-4*a*c+
b^2)^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2)
```

Rubi [A] (verified)

Time = 0.42 (sec) , antiderivative size = 221, normalized size of antiderivative = 1.00, number of steps used = 7, number of rules used = 4, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.286, Rules used = {3329, 2739, 632, 210}

$$\int \frac{1}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{2\sqrt{2}c\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{\sqrt{b^2-4ac}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{2\sqrt{2}c\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{\sqrt{b^2-4ac}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}$$

```
[In] Int[(a + b*Sin[x] + c*Sin[x]^2)^(-1),x]
```

```
[Out] (2*Sqrt[2]*c*ArcTan[(2*c + (b - Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[
b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]])])/(Sqrt[b^2 - 4*a*c]*Sqrt[b^2 - 2
*c*(a + c) - b*Sqrt[b^2 - 4*a*c]]) - (2*Sqrt[2]*c*ArcTan[(2*c + (b + Sqrt[b
^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c
]])])/(Sqrt[b^2 - 4*a*c]*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3329

```
Int[((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n_.) + (c_.)*sin[(d_.) + (e_.)*
(x_)]^(n2_.))^(-1), x_Symbol] :> Module[{q = Rt[b^2 - 4*a*c, 2]}, Dist[2*(c
/q), Int[1/(b - q + 2*c*Sin[d + e*x]^n), x], x] - Dist[2*(c/q), Int[1/(b +
```

```
q + 2*c*Sin[d + e*x]^n), x], x]] /; FreeQ[{a, b, c, d, e, n}, x] && EqQ[n2,
 2*n] && NeQ[b^2 - 4*a*c, 0]
```

Rubi steps

$$\text{integral} = \frac{(2c)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{\sqrt{b^2-4ac}} - \frac{(2c)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{\sqrt{b^2-4ac}}$$

$$= \frac{(4c)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{\sqrt{b^2-4ac}} - \frac{(4c)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{\sqrt{b^2-4ac}}$$

$$= -\frac{(8c)\text{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx, x, 4c+2\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{\sqrt{b^2-4ac}} + \frac{(8c)\text{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx, x, 4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{\sqrt{b^2-4ac}}$$

$$= \frac{2\sqrt{2}c\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{2\sqrt{2}c\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 1.00 (sec) , antiderivative size = 233, normalized size of antiderivative = 1.05

$$\int \frac{1}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{2ic\left(\frac{\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}} - \frac{\arctan\left(\frac{2c+\left(b+i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}}$$

```
[In] Integrate[(a + b*Sin[x] + c*Sin[x]^2)^(-1),x]
[Out] ((-2*I)*c*(ArcTan[(2*c + (b - I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt
[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]])]/Sqrt[b^2 - 2*c*(a + c) - I*b
```

```
*Sqrt[-b^2 + 4*a*c]] - ArcTan[(2*c + (b + I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(
Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]])]/Sqrt[b^2 - 2*c*(
a + c) + I*b*Sqrt[-b^2 + 4*a*c]]))/Sqrt[-1/2*b^2 + 2*a*c]
```

Maple [A] (verified)

Time = 1.02 (sec) , antiderivative size = 247, normalized size of antiderivative = 1.12

method	result
default	$2a\left(-\frac{\left(b\sqrt{-4ac+b^{2}}+4ac-b^{2}\right)\arctan\left(\frac{-2a\tan\left(\frac{x}{2}\right)+\sqrt{-4ac+b^{2}}-b}{\sqrt{4ac-2b^{2}+2b\sqrt{-4ac+b^{2}}+4a^{2}}}\right)}{a\left(4ac-b^{2}\right)\sqrt{4ac-2b^{2}+2b\sqrt{-4ac+b^{2}}+4a^{2}}}+\frac{\left(-b\sqrt{-4ac+b^{2}}+4ac-b^{2}\right)\arctan\left(\frac{2a\tan\left(\frac{x}{2}\right)+b+\sqrt{-4a}}{\sqrt{4ac-2b^{2}-2b\sqrt{-4ac+}}}\right.}{a\left(4ac-b^{2}\right)\sqrt{4ac-2b^{2}-2b\sqrt{-4ac+b^{2}}+4a^{2}}}\right.$
risch	$\sum_{\_R=\mathrm{RootOf}\left(\left(16a^{4}c^{2}-8a^{3}b^{2}c+32a^{3}c^{3}+a^{2}b^{4}-32a^{2}b^{2}c^{2}+16a^{2}c^{4}+10a\,b^{4}c-8a\,b^{2}c^{3}-b^{6}+b^{4}c^{2}\right)\_Z^{4}+\left(8a^{2}c^{2}-6a\,b^{2}c+8a\,c^{3}+b^{4}-2b^{2}c^{2}\right)\right.}$

```
[In] int(1/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 2*a*(-(b*(-4*a*c+b^2)^(1/2)+4*a*c-b^2)/a/(4*a*c-b^2)/(4*a*c-2*b^2+2*b*(-4*a
*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((-2*a*tan(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4
*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2))+(-b*(-4*a*c+b^2)^(1/2)+4*a*
c-b^2)/a/(4*a*c-b^2)/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arcta
n((2*a*tan(1/2*x)+b+(-4*a*c+b^2)^(1/2))/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)
+4*a^2)^(1/2)))
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 3495 vs. $2(187) = 374$.

Time = 0.51 (sec) , antiderivative size = 3495, normalized size of antiderivative = 15.81

$$\int \frac{1}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(1/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] -1/4*sqrt(2)*sqrt(-(b^2 - 2*a*c - 2*c^2 + (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2
 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 -
4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b
^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3
 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))*log(2*b^2*c*sin(x) + 4*b*c^2
 + 2*(4*a*c^4 + (8*a^2 - b^2)*c^3 + 2*(2*a^3 - 3*a*b^2)*c^2 - (a^2*b^2 - b^
4)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 -
12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*
b^2 + 2*a*b^4)*c))*sin(x) - sqrt(2)*((a^2*b^4 - b^6 + 8*a*c^5 + 2*(12*a^2 -
 b^2)*c^4 + 6*(4*a^3 - 3*a*b^2)*c^3 + (8*a^4 - 22*a^2*b^2 + 3*b^4)*c^2 - 2*
```

```
(3*a^3*b^2 - 4*a*b^4)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (1
6*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^
2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*cos(x) - (b^4 - 4*a*b^2*c)*cos(x))*sq
rt(-(b^2 - 2*a*c - 2*c^2 + (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2
*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a
^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 -
 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2
)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))) + 1/4*sqrt(2)*sqrt(-(b^2 - 2*a*c - 2*c^2 -
 (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt
(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3
- a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*
b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2
)*c))*log(2*b^2*c*sin(x) + 4*b*c^2 - 2*(4*a*c^4 + (8*a^2 - b^2)*c^3 + 2*(2*
a^3 - 3*a*b^2)*c^2 - (a^2*b^2 - b^4)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6
 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^
2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*sin(x) - sqrt(2)*((a^2
*b^4 - b^6 + 8*a*c^5 + 2*(12*a^2 - b^2)*c^4 + 6*(4*a^3 - 3*a*b^2)*c^3 + (8*
a^4 - 22*a^2*b^2 + 3*b^4)*c^2 - 2*(3*a^3*b^2 - 4*a*b^4)*c)*sqrt(b^2/(a^4*b^
2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3
 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*cos
(x) + (b^4 - 4*a*b^2*c)*cos(x))*sqrt(-(b^2 - 2*a*c - 2*c^2 - (a^2*b^2 - b^4
 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 -
 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 -
2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*
b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))) - 1/4*sq
rt(2)*sqrt(-(b^2 - 2*a*c - 2*c^2 - (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)
*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5
 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^
4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a
^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))*log(-2*b^2*c*sin(x) - 4*b*c^2 + 2*(
4*a*c^4 + (8*a^2 - b^2)*c^3 + 2*(2*a^3 - 3*a*b^2)*c^2 - (a^2*b^2 - b^4)*c)*
sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*
a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 +
2*a*b^4)*c))*sin(x) - sqrt(2)*((a^2*b^4 - b^6 + 8*a*c^5 + 2*(12*a^2 - b^2)*
c^4 + 6*(4*a^3 - 3*a*b^2)*c^3 + (8*a^4 - 22*a^2*b^2 + 3*b^4)*c^2 - 2*(3*a^3
*b^2 - 4*a*b^4)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2
- b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*
(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*cos(x) + (b^4 - 4*a*b^2*c)*cos(x))*sqrt(-(b
^2 - 2*a*c - 2*c^2 - (a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^
3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b
^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^
5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2
- 2*(2*a^3 - 3*a*b^2)*c))) + 1/4*sqrt(2)*sqrt(-(b^2 - 2*a*c - 2*c^2 + (a^2*
b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(
a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^
```

```
2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c
)))/(a^2*b^2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c))*
log(-2*b^2*c*sin(x) - 4*b*c^2 - 2*(4*a*c^4 + (8*a^2 - b^2)*c^3 + 2*(2*a^3 -
 3*a*b^2)*c^2 - (a^2*b^2 - b^4)*c)*sqrt(b^2/(a^4*b^2 - 2*a^2*b^4 + b^6 - 4*
a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2
 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*sin(x) - sqrt(2)*((a^2*b^4
- b^6 + 8*a*c^5 + 2*(12*a^2 - b^2)*c^4 + 6*(4*a^3 - 3*a*b^2)*c^3 + (8*a^4 -
 22*a^2*b^2 + 3*b^4)*c^2 - 2*(3*a^3*b^2 - 4*a*b^4)*c)*sqrt(b^2/(a^4*b^2 - 2
*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*
(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c))*cos(x) -
 (b^4 - 4*a*b^2*c)*cos(x))*sqrt(-(b^2 - 2*a*c - 2*c^2 + (a^2*b^2 - b^4 - 4*
a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)*sqrt(b^2/(a^4*b^2 - 2*a^
2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^3 - 2*(8*
a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)))/(a^2*b^2 -
 b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)))
```

Sympy [F(-1)]

Timed out.

$$\int \frac{1}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(1/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Timed out
```

Maxima [F]

$$\int \frac{1}{a + b\sin(x) + c\sin^2(x)}\,dx = \int \frac{1}{c\sin(x)^2 + b\sin(x) + a}\,dx$$

```
[In] integrate(1/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] integrate(1/(c*sin(x)^2 + b*sin(x) + a), x)
```

Giac [F(-1)]

Timed out.

$$\int \frac{1}{a+b\sin(x)+c\sin^2(x)}\,dx=\text{Timed out}$$

```
[In] integrate(1/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 24.61 (sec) , antiderivative size = 5064, normalized size of antiderivative = 22.91

$$\int \frac{1}{a+b\sin(x)+c\sin^2(x)}\,dx=\text{Too large to display}$$

```
[In] int(1/(a + c*sin(x)^2 + b*sin(x)),x)
```

```
[Out] atan(((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2
 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^
4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*((
-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*
b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 -
 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)
*(64*a*b^3 - 256*a^2*b*c) - 128*a^3*c + (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(
1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*
c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^
2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^
2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3
 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 32*a^2*b^2 - 128*a^2*c^2 + 3
2*a*b^2*c) + tan(x/2)*(128*a*c^2 - 32*a*b^2 + 64*a^2*c) + 32*a*b*c)*1i + (-
(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b
^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 -
8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*
(128*a*c^2 - 32*a*b^2 + 64*a^2*c) - (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2)
 + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4
+ 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^
2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(64*a*b^3 - 256*a^2*b*c) - 128*a^3*c
- (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6
*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^
2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x
/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*
b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^
3*b*c) + 32*a^2*b^2 - 128*a^2*c^2 + 32*a*b^2*c) + 32*a*b*c)*1i)/(64*a*c - (
```

```
-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*
b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 -
 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*((-(8*a*c
^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/
(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^
2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(64*a*
b^3 - 256*a^2*b*c) - 128*a^3*c + (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) +
b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 3
2*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c
^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512
*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*
a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 32*a^2*b^2 - 128*a^2*c^2 + 32*a*b^2
*c) + tan(x/2)*(128*a*c^2 - 32*a*b^2 + 64*a^2*c) + 32*a*b*c) + (-(8*a*c^3 +
 b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(
a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^
3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(128*a*c^2
 - 32*a*b^2 + 64*a^2*c) - (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8
*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c
^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10
*a*b^4*c)))^(1/2)*(tan(x/2)*(64*a*b^3 - 256*a^2*b*c) - 128*a^3*c - (-(8*a*c
^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/
(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^
2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*
b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 -
576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 3
2*a^2*b^2 - 128*a^2*c^2 + 32*a*b^2*c) + 32*a*b*c)))*(-(8*a*c^3 + b*(-(4*a*c
 - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b
^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b
^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*2i + atan(((-(8*a*c^3 - b*(-(4*
a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4
- b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^
3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b
^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 +
 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c
 - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(64*a*b^3 - 256*a^2*b*c)
- 128*a^3*c + (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2
*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*
c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^
(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*
c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^
3*c - 128*a^3*b*c) + 32*a^2*b^2 - 128*a^2*c^2 + 32*a*b^2*c) + tan(x/2)*(128
*a*c^2 - 32*a*b^2 + 64*a^2*c) + 32*a*b*c)*1i + (-(8*a*c^3 - b*(-(4*a*c - b^
2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 +
16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c
- 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(128*a*c^2 - 32*a*b^2 + 64
```

```
*a^2*c) - (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2
*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2
+ b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2
)*(tan(x/2)*(64*a*b^3 - 256*a^2*b*c) - 128*a^3*c - (-(8*a*c^3 - b*(-(4*a*c
- b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^
6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^
2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c
- 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) +
 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 32*a^2*b^2 - 128*
a^2*c^2 + 32*a*b^2*c) + 32*a*b*c)*1i)/(64*a*c - (-(8*a*c^3 - b*(-(4*a*c - b
^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 +
 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c
 - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^
(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2
*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a
^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(64*a*b^3 - 256*a^2*b*c) - 128*a
^3*c + (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^
2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b
^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(
tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b^2 + 512*a^2*c^3 + 768*a^3*c^2 - 1
28*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^3 + 128*a^2*b*c^2 - 32*a*b^3*c - 1
28*a^3*b*c) + 32*a^2*b^2 - 128*a^2*c^2 + 32*a*b^2*c) + tan(x/2)*(128*a*c^2
- 32*a*b^2 + 64*a^2*c) + 32*a*b*c) + (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2
) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4
 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b
^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(128*a*c^2 - 32*a*b^2 + 64*a^2*c) -
(-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a
*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2
- 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2
)*(64*a*b^3 - 256*a^2*b*c) - 128*a^3*c - (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^
(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2
*c^4 + 32*a^3*c^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a
^2*b^2*c^2 + 10*a*b^4*c)))^(1/2)*(tan(x/2)*(96*a*b^4 + 256*a^4*c - 64*a^3*b
^2 + 512*a^2*c^3 + 768*a^3*c^2 - 128*a*b^2*c^2 - 576*a^2*b^2*c) + 32*a^2*b^
3 + 128*a^2*b*c^2 - 32*a*b^3*c - 128*a^3*b*c) + 32*a^2*b^2 - 128*a^2*c^2 +
32*a*b^2*c) + 32*a*b*c)))*(-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8
*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(a^2*b^4 - b^6 + 16*a^2*c^4 + 32*a^3*c
^3 + 16*a^4*c^2 + b^4*c^2 - 8*a*b^2*c^3 - 8*a^3*b^2*c - 32*a^2*b^2*c^2 + 10
*a*b^4*c)))^(1/2)*2i
```

3.6 $\int \frac{\csc(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 17, antiderivative size = 244

$$\int \frac{\csc(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\sqrt{2}c\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{a\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{\sqrt{2}c\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{a\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} - \frac{\operatorname{arctanh}(\cos(x))}{a}$$

```
[Out] -arctanh(cos(x))/a-c*arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(
1/2)/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(1+b/(-4*a*c+b^2)^
(1/2))/a/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2)-c*arctan(1/2*(2*c+(b+(-
4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^
(1/2))*2^(1/2)*(1-b/(-4*a*c+b^2)^(1/2))/a/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/
2))^(1/2)
```

Rubi [A] (verified)

Time = 0.77 (sec) , antiderivative size = 244, normalized size of antiderivative = 1.00, number of steps used = 10, number of rules used = 6, $\frac{\text{number of rules}}{\text{integrand size}} = 0.353$, Rules used = {3337, 3855, 3373, 2739, 632, 210}

$$\int \frac{\csc(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\sqrt{2}c\left(\frac{b}{\sqrt{b^2-4ac}}+1\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{a\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{\sqrt{2}c\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{a\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{\operatorname{arctanh}(\cos(x))}{a}$$

```
[In] Int[Csc[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -((Sqrt[2]*c*(1 + b/Sqrt[b^2 - 4*a*c])*ArcTan[(2*c + (b - Sqrt[b^2 - 4*a*c]
)*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]])])/(a*Sq
rt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]])) - (Sqrt[2]*c*(1 - b/Sqrt[b^2
- 4*a*c])*ArcTan[(2*c + (b + Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2
 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])])/(a*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt
[b^2 - 4*a*c]]) - ArcTanh[Cos[x]]/a
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3337

```
Int[sin[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_), x_Symbol] :> Int[ExpandTr
ig[sin[d + e*x]^m*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n))^p, x], x] /
; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && NeQ[b^2 - 4*a*c, 0] && Integ
ersQ[m, n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3855

```
Int[csc[(c_.) + (d_.)*(x_)], x_Symbol] :> Simp[-ArcTanh[Cos[c + d*x]]/d, x]
 /; FreeQ[{c, d}, x]
```

Rubi steps

$$\begin{aligned}
\text{integral} &= \int \left(\frac{\csc(x)}{a}+\frac{-b-c\sin(x)}{a\left(a+b\sin(x)+c\sin^2(x)\right)}\right)dx\\
&= \frac{\int \csc(x)\,dx}{a}+\frac{\int \frac{-b-c\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{a}\\
&= -\frac{\operatorname{arctanh}(\cos(x))}{a}-\frac{\left(c\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{a}\\
&\quad-\frac{\left(c\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{a}\\
&= -\frac{\operatorname{arctanh}(\cos(x))}{a}\\
&\quad-\frac{\left(2c\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)}{a}\\
&\quad-\frac{\left(2c\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)}{a}
\end{aligned}$$

$$= -\frac{\operatorname{arctanh}(\cos(x))}{a} + \frac{\left(4c\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}dx, x, 4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{a} + \frac{\left(4c\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}dx, x, 4c+2\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{a}$$

$$= -\frac{\sqrt{2}c\left(1+\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{a\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{\sqrt{2}c\left(1-\frac{b}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{a\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} - \frac{\operatorname{arctanh}(\cos(x))}{a}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 1.39 (sec) , antiderivative size = 306, normalized size of antiderivative = 1.25

$$\int \frac{\csc(x)}{a+b\sin(x)+c\sin^2(x)}dx = -\frac{\frac{c\left(-ib+\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}} + \frac{c\left(ib+\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b+i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}} + \log\left(\cos\left(\right.\right.}{a}$$

```
[In] Integrate[Csc[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -(((c*((-I)*b + Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b - I*Sqrt[-b^2 + 4*a*c]
)*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]])])/(S
qrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]]) + (
c*(I*b + Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b + I*Sqrt[-b^2 + 4*a*c])*Tan[x
/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]])])/(Sqrt[-1/
2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]]) + Log[Cos[
x/2]] - Log[Sin[x/2]])/a)
```

Maple [A] (verified)

Time = 2.00 (sec) , antiderivative size = 284, normalized size of antiderivative = 1.16

method	result
default	$-\frac{2\left(2\sqrt{-4ac+b^2}\,ac-\sqrt{-4ac+b^2}\,b^2-4bca+b^3\right)\arctan\left(\frac{-2a\tan\left(\frac{x}{2}\right)+\sqrt{-4ac+b^2}-b}{\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}\right)}{a\left(4ac-b^2\right)\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}+\frac{2\left(-2\sqrt{-4ac+b^2}\,ac+\sqrt{-4ac+b^2}\,b^2-4bca-\cdots\right.}{a\left(4ac-b^2\right)\sqrt{4ac-2b\cdots}}$
risch	Expression too large to display

```
[In] int(csc(x)/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] -2*(2*(-4*a*c+b^2)^(1/2)*a*c-(-4*a*c+b^2)^(1/2)*b^2-4*b*c*a+b^3)/a/(4*a*c-b
^2)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((-2*a*tan(1/2*x
)+(-4*a*c+b^2)^(1/2)-b)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2))+2
*(-2*(-4*a*c+b^2)^(1/2)*a*c+(-4*a*c+b^2)^(1/2)*b^2-4*b*c*a+b^3)/a/(4*a*c-b^
2)/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((2*a*tan(1/2*x)+
b+(-4*a*c+b^2)^(1/2))/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2))+1/a
*ln(tan(1/2*x))
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 5296 vs. 2(208) = 416.

Time = 73.25 (sec) , antiderivative size = 5296, normalized size of antiderivative = 21.70

$$\int \frac{\csc(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(csc(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] Too large to include
```

Sympy [F]

$$\int \frac{\csc(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\csc{(x)}}{a+b\sin{(x)}+c\sin^2{(x)}}\,dx$$

```
[In] integrate(csc(x)/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Integral(csc(x)/(a + b*sin(x) + c*sin(x)**2), x)
```

Maxima [F]

$$\int \frac{\csc(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \int \frac{\csc(x)}{c\sin(x)^2 + b\sin(x) + a}\,dx$$

```
[In] integrate(csc(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] -1/2*(2*a*integrate(2*(2*b*c*cos(3*x)^2 + 2*b*c*cos(x)^2 + 2*b*c*sin(3*x)^2
 + 2*b*c*sin(x)^2 + 4*(2*a*b + b*c)*cos(2*x)^2 + 2*(2*b^2 + 2*a*c + c^2)*co
s(x)*sin(2*x) + 4*(2*a*b + b*c)*sin(2*x)^2 + c^2*sin(x) - (2*b*c*cos(2*x) +
 c^2*sin(3*x) - c^2*sin(x))*cos(4*x) - 2*(2*b*c*cos(x) + (2*b^2 + 2*a*c + c
^2)*sin(2*x))*cos(3*x) - 2*(b*c + (2*b^2 + 2*a*c + c^2)*sin(x))*cos(2*x) +
(c^2*cos(3*x) - c^2*cos(x) - 2*b*c*sin(2*x))*sin(4*x) - (4*b*c*sin(x) + c^2
 - 2*(2*b^2 + 2*a*c + c^2)*cos(2*x))*sin(3*x))/(a*c^2*cos(4*x)^2 + 4*a*b^2*
cos(3*x)^2 + 4*a*b^2*cos(x)^2 + a*c^2*sin(4*x)^2 + 4*a*b^2*sin(3*x)^2 + 4*a
*b^2*sin(x)^2 + 4*a*b*c*sin(x) + a*c^2 + 4*(4*a^3 + 4*a^2*c + a*c^2)*cos(2*
x)^2 + 8*(2*a^2*b + a*b*c)*cos(x)*sin(2*x) + 4*(4*a^3 + 4*a^2*c + a*c^2)*si
n(2*x)^2 - 2*(2*a*b*c*sin(3*x) - 2*a*b*c*sin(x) - a*c^2 + 2*(2*a^2*c + a*c^
2)*cos(2*x))*cos(4*x) - 8*(a*b^2*cos(x) + (2*a^2*b + a*b*c)*sin(2*x))*cos(3
*x) - 4*(2*a^2*c + a*c^2 + 2*(2*a^2*b + a*b*c)*sin(x))*cos(2*x) + 4*(a*b*c*
cos(3*x) - a*b*c*cos(x) - (2*a^2*c + a*c^2)*sin(2*x))*sin(4*x) - 4*(2*a*b^2
*sin(x) + a*b*c - 2*(2*a^2*b + a*b*c)*cos(2*x))*sin(3*x)), x) + log(cos(x)^
2 + sin(x)^2 + 2*cos(x) + 1) - log(cos(x)^2 + sin(x)^2 - 2*cos(x) + 1))/a
```

Giac [F(-1)]

Timed out.

$$\int \frac{\csc(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(csc(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 26.30 (sec) , antiderivative size = 11540, normalized size of antiderivative = 47.30

$$\int \frac{\csc(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(1/(sin(x)*(a + c*sin(x)^2 + b*sin(x))),x)
```

```
[Out] atan((((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^
2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4
```

```
*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 +
 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3
*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(
4*a*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1
/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^
4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a
^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(((8*a^2*c
^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3
 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-
(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 1
6*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a
^4*b^2*c^2)))^(1/2)*(((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)
^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^
2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6
 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*
b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^6*c - 51
2*a*b^6 + 544*a^3*b^4 - 64*a^5*b^2 + 6144*a^3*c^4 + 12288*a^4*c^3 + 6400*a^
5*c^2 + 512*a*b^4*c^2 + 4608*a^2*b^4*c - 3776*a^4*b^2*c - 3584*a^2*b^2*c^3
 - 13312*a^3*b^2*c^2) - 128*a^2*b^5 + 96*a^4*b^3 - 512*a^3*b*c^3 + 800*a^3*b
^3*c - 1152*a^4*b*c^2 + 128*a^2*b^3*c^2 - 384*a^5*b*c) - tan(x/2)*(256*a^5*
c - 512*b^6 + 416*a^2*b^4 - 64*a^4*b^2 + 3072*a^2*c^4 + 5632*a^3*c^3 + 2816
*a^4*c^2 + 512*b^4*c^2 - 2816*a*b^2*c^3 - 2368*a^3*b^2*c - 8576*a^2*b^2*c^2
 + 3840*a*b^4*c) + 256*a*b^5 - 128*a^3*b^3 - 256*a*b^3*c^2 + 1024*a^2*b*c^3
 - 1568*a^2*b^3*c + 2176*a^3*b*c^2 + 512*a^4*b*c) - 128*b^5 + tan(x/2)*(96*
a*b^4 - 1536*a*c^4 - 256*b^4*c - 1024*a^2*c^3 + 448*a^3*c^2 + 256*b^2*c^3 +
 1408*a*b^2*c^2 - 512*a^2*b^2*c) + 32*a^2*b^3 + 128*b^3*c^2 - 1312*a^2*b*c^
2 - 640*a*b*c^3 + 864*a*b^3*c - 128*a^3*b*c) + tan(x/2)*(640*a*c^3 + 32*b^4
 + 768*c^4 + 64*a^2*c^2 - 256*b^2*c^2 - 128*a*b^2*c) + 128*b*c^3 - 96*b^3*c
 + 320*a*b*c^2)*1i + ((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)
^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^
2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6
 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*
b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(640*a*c^3 + 32
*b^4 + 768*c^4 + 64*a^2*c^2 - 256*b^2*c^2 - 128*a*b^2*c) - ((8*a^2*c^4 - b^
6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^
2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c
 - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c
^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*
c^2)))^(1/2)*(((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2)
 + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 +
 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a
^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2
 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^5*c - 512*b^6 +
 416*a^2*b^4 - 64*a^4*b^2 + 3072*a^2*c^4 + 5632*a^3*c^3 + 2816*a^4*c^2 + 51
2*b^4*c^2 - 2816*a*b^2*c^3 - 2368*a^3*b^2*c - 8576*a^2*b^2*c^2 + 3840*a*b^4
```

*c) + ((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^
2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4
*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 +
 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3
*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^6*c - 512*a*b^6 + 544*a
^3*b^4 - 64*a^5*b^2 + 6144*a^3*c^4 + 12288*a^4*c^3 + 6400*a^5*c^2 + 512*a*b
^4*c^2 + 4608*a^2*b^4*c - 3776*a^4*b^2*c - 3584*a^2*b^2*c^3 - 13312*a^3*b^2
*c^2) - 128*a^2*b^5 + 96*a^4*b^3 - 512*a^3*b*c^3 + 800*a^3*b^3*c - 1152*a^4
*b*c^2 + 128*a^2*b^3*c^2 - 384*a^5*b*c) - 256*a*b^5 + 128*a^3*b^3 + 256*a*b
^3*c^2 - 1024*a^2*b*c^3 + 1568*a^2*b^3*c - 2176*a^3*b*c^2 - 512*a^4*b*c) -
128*b^5 + tan(x/2)*(96*a*b^4 - 1536*a*c^4 - 256*b^4*c - 1024*a^2*c^3 + 448*
a^3*c^2 + 256*b^2*c^3 + 1408*a*b^2*c^2 - 512*a^2*b^2*c) + 32*a^2*b^3 + 128*
b^3*c^2 - 1312*a^2*b*c^2 - 640*a*b*c^3 + 864*a*b^3*c - 128*a^3*b*c) + 128*b
*c^3 - 96*b^3*c + 320*a*b*c^2)*1i)/(((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(
4*a*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1
/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^
4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a
^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*
(640*a*c^3 + 32*b^4 + 768*c^4 + 64*a^2*c^2 - 256*b^2*c^2 - 128*a*b^2*c) - (
(8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a
*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*
a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5
*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^
3 - 32*a^4*b^2*c^2)))^(1/2)*(((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c -
 b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 1
8*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 -
 a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*
c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^
5*c - 512*b^6 + 416*a^2*b^4 - 64*a^4*b^2 + 3072*a^2*c^4 + 5632*a^3*c^3 + 28
16*a^4*c^2 + 512*b^4*c^2 - 2816*a*b^2*c^3 - 2368*a^3*b^2*c - 8576*a^2*b^2*c
^2 + 3840*a*b^4*c) + ((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)
^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^
2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6
 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*
b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^6*c - 51
2*a*b^6 + 544*a^3*b^4 - 64*a^5*b^2 + 6144*a^3*c^4 + 12288*a^4*c^3 + 6400*a^
5*c^2 + 512*a*b^4*c^2 + 4608*a^2*b^4*c - 3776*a^4*b^2*c - 3584*a^2*b^2*c^3
- 13312*a^3*b^2*c^2) - 128*a^2*b^5 + 96*a^4*b^3 - 512*a^3*b*c^3 + 800*a^3*b
^3*c - 1152*a^4*b*c^2 + 128*a^2*b^3*c^2 - 384*a^5*b*c) - 256*a*b^5 + 128*a^
3*b^3 + 256*a*b^3*c^2 - 1024*a^2*b*c^3 + 1568*a^2*b^3*c - 2176*a^3*b*c^2 -
512*a^4*b*c) - 128*b^5 + tan(x/2)*(96*a*b^4 - 1536*a*c^4 - 256*b^4*c - 1024
*a^2*c^3 + 448*a^3*c^2 + 256*b^2*c^3 + 1408*a*b^2*c^2 - 512*a^2*b^2*c) + 32
*a^2*b^3 + 128*b^3*c^2 - 1312*a^2*b*c^2 - 640*a*b*c^3 + 864*a*b^3*c - 128*a
^3*b*c) + 128*b*c^3 - 96*b^3*c + 320*a*b*c^2) - ((8*a^2*c^4 - b^6 + 8*a^3*c
^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c

```
- b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(
1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3
*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2
)*(((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^2 -
 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c
+ 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32
*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^
2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a
*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2)
 - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b
^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*
b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(((8*a^2*c^4
- b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) + b^4*c^2 - 6*a*b^2*c^3 +
b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2 + 8*a*b^4*c + 2*a*b*c*(-(4*
a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a
^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*
b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^6*c - 512*a*b^6 + 544*a^3*b^4 - 64*a^5*b^
2 + 6144*a^3*c^4 + 12288*a^4*c^3 + 6400*a^5*c^2 + 512*a*b^4*c^2 + 4608*a^2*
b^4*c - 3776*a^4*b^2*c - 3584*a^2*b^2*c^3 - 13312*a^3*b^2*c^2) - 128*a^2*b^
5 + 96*a^4*b^3 - 512*a^3*b*c^3 + 800*a^3*b^3*c - 1152*a^4*b*c^2 + 128*a^2*b
^3*c^2 - 384*a^5*b*c) - tan(x/2)*(256*a^5*c - 512*b^6 + 416*a^2*b^4 - 64*a^
4*b^2 + 3072*a^2*c^4 + 5632*a^3*c^3 + 2816*a^4*c^2 + 512*b^4*c^2 - 2816*a*b
^2*c^3 - 2368*a^3*b^2*c - 8576*a^2*b^2*c^2 + 3840*a*b^4*c) + 256*a*b^5 - 12
8*a^3*b^3 - 256*a*b^3*c^2 + 1024*a^2*b*c^3 - 1568*a^2*b^3*c + 2176*a^3*b*c^
2 + 512*a^4*b*c) - 128*b^5 + tan(x/2)*(96*a*b^4 - 1536*a*c^4 - 256*b^4*c -
1024*a^2*c^3 + 448*a^3*c^2 + 256*b^2*c^3 + 1408*a*b^2*c^2 - 512*a^2*b^2*c)
+ 32*a^2*b^3 + 128*b^3*c^2 - 1312*a^2*b*c^2 - 640*a*b*c^3 + 864*a*b^3*c - 1
28*a^3*b*c) + tan(x/2)*(640*a*c^3 + 32*b^4 + 768*c^4 + 64*a^2*c^2 - 256*b^2
*c^2 - 128*a*b^2*c) + 128*b*c^3 - 96*b^3*c + 320*a*b*c^2) + 256*c^3*tan(x/2
) + 64*b*c^2))*((8*a^2*c^4 - b^6 + 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2)
 + b^4*c^2 - 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 18*a^2*b^2*c^2
+ 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*
a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^
2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*2i + atan(((-(b^6 - 8*a^2*c^4 -
 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(
-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b
^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2
+ 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2
)))^(1/2)*((-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) -
b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8
*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4
*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 -
 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*((-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 -
b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2
)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))
```

/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*
c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(25
6*a*b^5 - tan(x/2)*(256*a^5*c - 512*b^6 + 416*a^2*b^4 - 64*a^4*b^2 + 3072*a
^2*c^4 + 5632*a^3*c^3 + 2816*a^4*c^2 + 512*b^4*c^2 - 2816*a*b^2*c^3 - 2368*
a^3*b^2*c - 8576*a^2*b^2*c^2 + 3840*a*b^4*c) + (-(b^6 - 8*a^2*c^4 - 8*a^3*c
^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c
- b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(
1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3
*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2
)*(tan(x/2)*(256*a^6*c - 512*a*b^6 + 544*a^3*b^4 - 64*a^5*b^2 + 6144*a^3*c^
4 + 12288*a^4*c^3 + 6400*a^5*c^2 + 512*a*b^4*c^2 + 4608*a^2*b^4*c - 3776*a^
4*b^2*c - 3584*a^2*b^2*c^3 - 13312*a^3*b^2*c^2) - 128*a^2*b^5 + 96*a^4*b^3
- 512*a^3*b*c^3 + 800*a^3*b^3*c - 1152*a^4*b*c^2 + 128*a^2*b^3*c^2 - 384*a^
5*b*c) - 128*a^3*b^3 - 256*a*b^3*c^2 + 1024*a^2*b*c^3 - 1568*a^2*b^3*c + 21
76*a^3*b*c^2 + 512*a^4*b*c) - 128*b^5 + tan(x/2)*(96*a*b^4 - 1536*a*c^4 - 2
56*b^4*c - 1024*a^2*c^3 + 448*a^3*c^2 + 256*b^2*c^3 + 1408*a*b^2*c^2 - 512*
a^2*b^2*c) + 32*a^2*b^3 + 128*b^3*c^2 - 1312*a^2*b*c^2 - 640*a*b*c^3 + 864*
a*b^3*c - 128*a^3*b*c) + tan(x/2)*(640*a*c^3 + 32*b^4 + 768*c^4 + 64*a^2*c^
2 - 256*b^2*c^2 - 128*a*b^2*c) + 128*b*c^3 - 96*b^3*c + 320*a*b*c^2)*1i + (
-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*
a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2
*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^
5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c
^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(640*a*c^3 + 32*b^4 + 768*c^4 + 64*a
^2*c^2 - 256*b^2*c^2 - 128*a*b^2*c) - (-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*
(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)
^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*
(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c -
8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*((-(b^6
- 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2
*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*
c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3
+ 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 -
32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^5*c - 512*b^6 + 416*a^2*b^4 - 64*a
^4*b^2 + 3072*a^2*c^4 + 5632*a^3*c^3 + 2816*a^4*c^2 + 512*b^4*c^2 - 2816*a*
b^2*c^3 - 2368*a^3*b^2*c - 8576*a^2*b^2*c^2 + 3840*a*b^4*c) - 256*a*b^5 + (
-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*
a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2
*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^
5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c
^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^6*c - 512*a*b^6 + 544*a^3*b^4
- 64*a^5*b^2 + 6144*a^3*c^4 + 12288*a^4*c^3 + 6400*a^5*c^2 + 512*a*b^4*c^2
+ 4608*a^2*b^4*c - 3776*a^4*b^2*c - 3584*a^2*b^2*c^3 - 13312*a^3*b^2*c^2)
- 128*a^2*b^5 + 96*a^4*b^3 - 512*a^3*b*c^3 + 800*a^3*b^3*c - 1152*a^4*b*c^2
+ 128*a^2*b^3*c^2 - 384*a^5*b*c) + 128*a^3*b^3 + 256*a*b^3*c^2 - 1024*a^2*

```
b*c^3 + 1568*a^2*b^3*c - 2176*a^3*b*c^2 - 512*a^4*b*c) - 128*b^5 + tan(x/2)
*(96*a*b^4 - 1536*a*c^4 - 256*b^4*c - 1024*a^2*c^3 + 448*a^3*c^2 + 256*b^2*
c^3 + 1408*a*b^2*c^2 - 512*a^2*b^2*c) + 32*a^2*b^3 + 128*b^3*c^2 - 1312*a^2
*b*c^2 - 640*a*b*c^3 + 864*a*b^3*c - 128*a^3*b*c) + 128*b*c^3 - 96*b^3*c +
320*a*b*c^2)*1i)/(256*c^3*tan(x/2) + 64*b*c^2 - (-(b^6 - 8*a^2*c^4 - 8*a^3*
c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c
 - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^
(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^
3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/
2)*((-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2
 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*
c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 +
32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*
b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*((-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(
4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1
/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^
4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a
^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(256*a*b^5
 - tan(x/2)*(256*a^5*c - 512*b^6 + 416*a^2*b^4 - 64*a^4*b^2 + 3072*a^2*c^4
+ 5632*a^3*c^3 + 2816*a^4*c^2 + 512*b^4*c^2 - 2816*a*b^2*c^3 - 2368*a^3*b^2
*c - 8576*a^2*b^2*c^2 + 3840*a*b^4*c) + (-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^
3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^
3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(
2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c
 - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*(tan(
x/2)*(256*a^6*c - 512*a*b^6 + 544*a^3*b^4 - 64*a^5*b^2 + 6144*a^3*c^4 + 122
88*a^4*c^3 + 6400*a^5*c^2 + 512*a*b^4*c^2 + 4608*a^2*b^4*c - 3776*a^4*b^2*c
 - 3584*a^2*b^2*c^3 - 13312*a^3*b^2*c^2) - 128*a^2*b^5 + 96*a^4*b^3 - 512*a
^3*b*c^3 + 800*a^3*b^3*c - 1152*a^4*b*c^2 + 128*a^2*b^3*c^2 - 384*a^5*b*c)
 - 128*a^3*b^3 - 256*a*b^3*c^2 + 1024*a^2*b*c^3 - 1568*a^2*b^3*c + 2176*a^3*
b*c^2 + 512*a^4*b*c) - 128*b^5 + tan(x/2)*(96*a*b^4 - 1536*a*c^4 - 256*b^4*
c - 1024*a^2*c^3 + 448*a^3*c^2 + 256*b^2*c^3 + 1408*a*b^2*c^2 - 512*a^2*b^2
*c) + 32*a^2*b^3 + 128*b^3*c^2 - 1312*a^2*b*c^2 - 640*a*b*c^3 + 864*a*b^3*c
 - 128*a^3*b*c) + tan(x/2)*(640*a*c^3 + 32*b^4 + 768*c^4 + 64*a^2*c^2 - 256
*b^2*c^2 - 128*a*b^2*c) + 128*b*c^3 - 96*b^3*c + 320*a*b*c^2) + (-(b^6 - 8*
a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3
+ b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(
4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16
*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^
4*b^2*c^2)))^(1/2)*(tan(x/2)*(640*a*c^3 + 32*b^4 + 768*c^4 + 64*a^2*c^2 - 2
56*b^2*c^2 - 128*a*b^2*c) - (-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c -
 b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 1
8*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 -
 a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*
c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*((-(b^6 - 8*a^2*c
```

```
^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3 + b*c
^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(4*a*c
 - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16*a^6*
c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^4*b^2
*c^2)))^(1/2)*(tan(x/2)*(256*a^5*c - 512*b^6 + 416*a^2*b^4 - 64*a^4*b^2 + 3
072*a^2*c^4 + 5632*a^3*c^3 + 2816*a^4*c^2 + 512*b^4*c^2 - 2816*a*b^2*c^3 -
2368*a^3*b^2*c - 8576*a^2*b^2*c^2 + 3840*a*b^4*c) - 256*a*b^5 + (-(b^6 - 8*
a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^2 + 6*a*b^2*c^3
+ b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4*c + 2*a*b*c*(-(
4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 + 32*a^5*c^3 + 16
*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3*b^2*c^3 - 32*a^
4*b^2*c^2)))^(1/2)*(tan(x/2)*(256*a^6*c - 512*a*b^6 + 544*a^3*b^4 - 64*a^5*
b^2 + 6144*a^3*c^4 + 12288*a^4*c^3 + 6400*a^5*c^2 + 512*a*b^4*c^2 + 4608*a^
2*b^4*c - 3776*a^4*b^2*c - 3584*a^2*b^2*c^3 - 13312*a^3*b^2*c^2) - 128*a^2*
b^5 + 96*a^4*b^3 - 512*a^3*b*c^3 + 800*a^3*b^3*c - 1152*a^4*b*c^2 + 128*a^2
*b^3*c^2 - 384*a^5*b*c) + 128*a^3*b^3 + 256*a*b^3*c^2 - 1024*a^2*b*c^3 + 15
68*a^2*b^3*c - 2176*a^3*b*c^2 - 512*a^4*b*c) - 128*b^5 + tan(x/2)*(96*a*b^4
 - 1536*a*c^4 - 256*b^4*c - 1024*a^2*c^3 + 448*a^3*c^2 + 256*b^2*c^3 + 1408
*a*b^2*c^2 - 512*a^2*b^2*c) + 32*a^2*b^3 + 128*b^3*c^2 - 1312*a^2*b*c^2 - 6
40*a*b*c^3 + 864*a*b^3*c - 128*a^3*b*c) + 128*b*c^3 - 96*b^3*c + 320*a*b*c^
2)))*(-(b^6 - 8*a^2*c^4 - 8*a^3*c^3 - b^3*(-(4*a*c - b^2)^3)^(1/2) - b^4*c^
2 + 6*a*b^2*c^3 + b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 18*a^2*b^2*c^2 - 8*a*b^4
*c + 2*a*b*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^4*b^4 - a^2*b^6 + 16*a^4*c^4 +
 32*a^5*c^3 + 16*a^6*c^2 + 10*a^3*b^4*c - 8*a^5*b^2*c + a^2*b^4*c^2 - 8*a^3
*b^2*c^3 - 32*a^4*b^2*c^2)))^(1/2)*2i + log(tan(x/2))/a
```

3.7 $\int \frac{\csc^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 271

$$\int \frac{\csc^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\sqrt{2}bc\left(1+\frac{b^2-2ac}{b\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{a^2\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} + \frac{\sqrt{2}bc\left(1-\frac{b^2-2ac}{b\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{a^2\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} + \frac{b\,\text{arctanh}(\cos(x))}{a^2} - \frac{\cot(x)}{a}$$

```
[Out] b*arctanh(cos(x))/a^2-cot(x)/a+b*c*arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*t
an(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(1+(
-2*a*c+b^2)/b/(-4*a*c+b^2)^(1/2))/a^2/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^
(1/2)+b*c*arctan(1/2*(2*c+(b+(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2
*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(1+(2*a*c-b^2)/b/(-4*a*c+b^2)
^(1/2))/a^2/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2)
```

Rubi [A] (verified)

Time = 0.91 (sec) , antiderivative size = 271, normalized size of antiderivative = 1.00, number of steps used = 12, number of rules used = 8, $\frac{\text{number of rules}}{\text{integrand size}} = 0.421$, Rules used = {3337, 3855, 3852, 8, 3373, 2739, 632, 210}

$$\int \frac{\csc^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\sqrt{2}bc\left(\frac{b^2-2ac}{b\sqrt{b^2-4ac}}+1\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{a^2\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{\sqrt{2}bc\left(1-\frac{b^2-2ac}{b\sqrt{b^2-4ac}}\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{a^2\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{b\operatorname{arctanh}(\cos(x))}{a^2} - \frac{\cot(x)}{a}$$

```
[In] Int[Csc[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (Sqrt[2]*b*c*(1 + (b^2 - 2*a*c)/(b*Sqrt[b^2 - 4*a*c]))*ArcTan[(2*c + (b - S
qrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 -
4*a*c]])])/(a^2*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]]) + (Sqrt[2]*b
*c*(1 - (b^2 - 2*a*c)/(b*Sqrt[b^2 - 4*a*c]))*ArcTan[(2*c + (b + Sqrt[b^2 -
4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])])
/(a^2*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]]) + (b*ArcTanh[Cos[x]])/
a^2 - Cot[x]/a
```

Rule 8

```
Int[a_, x_Symbol] :> Simp[a*x, x] /; FreeQ[a, x]
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
```

```
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3337

```
Int[sin[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_), x_Symbol] :> Int[ExpandTr
ig[sin[d + e*x]^m*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n))^p, x], x] /
; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && NeQ[b^2 - 4*a*c, 0] && Integ
ersQ[m, n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3852

```
Int[csc[(c_.) + (d_.)*(x_)]^(n_), x_Symbol] :> Dist[-d^(-1), Subst[Int[Expa
ndIntegrand[(1 + x^2)^(n/2 - 1), x], x], x, Cot[c + d*x]], x] /; FreeQ[{c,
d}, x] && IGtQ[n/2, 0]
```

Rule 3855

```
Int[csc[(c_.) + (d_.)*(x_)], x_Symbol] :> Simp[-ArcTanh[Cos[c + d*x]]/d, x]
 /; FreeQ[{c, d}, x]
```

Rubi steps

$$
\begin{aligned}
\text{integral} &= \int \left(-\frac{b\csc(x)}{a^2}+\frac{\csc^2(x)}{a}+\frac{b^2\left(1-\frac{ac}{b^2}\right)+bc\sin(x)}{a^2\left(a+b\sin(x)+c\sin^2(x)\right)}\right)dx\\
&= \frac{\int \frac{b^2\left(1-\frac{ac}{b^2}\right)+bc\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{a^2}+\frac{\int \csc^2(x)\,dx}{a}-\frac{b\int \csc(x)\,dx}{a^2}\\
&= \frac{b\text{arctanh}(\cos(x))}{a^2}-\frac{\text{Subst}\left(\int 1\,dx,x,\cot(x)\right)}{a}\\
&\quad+\frac{\left(c\left(b-\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{a^2}\\
&\quad+\frac{\left(c\left(b+\frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{a^2}
\end{aligned}
$$

$$= \frac{b\operatorname{arctanh}(\cos(x))}{a^2} - \frac{\cot(x)}{a} + \frac{\left(2c\left(b - \frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{a^2} + \frac{\left(2c\left(b + \frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{a^2}$$

$$= \frac{b\operatorname{arctanh}(\cos(x))}{a^2} - \frac{\cot(x)}{a} - \frac{\left(4c\left(b - \frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx, x, 4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{a^2} - \frac{\left(4c\left(b + \frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\right)\operatorname{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx, x, 4c+2\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{a^2}$$

$$= \frac{\sqrt{2}c\left(b + \frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{a^2\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} + \frac{\sqrt{2}c\left(b - \frac{b^2-2ac}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{a^2\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} + \frac{b\operatorname{arctanh}(\cos(x))}{a^2} - \frac{\cot(x)}{a}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 2.52 (sec) , antiderivative size = 388, normalized size of antiderivative = 1.43

$$\int \frac{\csc^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$

$$= \frac{\csc^2(x)(-2a-c+c\cos(2x)-2b\sin(x))\left(-\frac{2c\left(-ib^2+2iac+b\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}} + \frac{2ic\left(-b\right.}{}\right.}{4a^2\left(c+b\csc\right.}$$

```
[In] Integrate[Csc[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (Csc[x]^2*(-2*a - c + c*Cos[2*x] - 2*b*Sin[x])*((-2*c*((-I)*b^2 + (2*I)*a*c
 + b*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b - I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])
/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^
```

```
2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]]) + ((2*I)*c*(-b
^2 + 2*a*c + I*b*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b + I*Sqrt[-b^2 + 4*a*c
])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]])])/(
Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]]) +
a*Cot[x/2] - 2*b*Log[Cos[x/2]] + 2*b*Log[Sin[x/2]] - a*Tan[x/2]))/(4*a^2*(c
 + b*Csc[x] + a*Csc[x]^2))
```

Maple [A] (verified)

Time = 2.83 (sec) , antiderivative size = 337, normalized size of antiderivative = 1.24

method	result
default	$\frac{\tan\left(\frac{x}{2}\right)}{2a}+\frac{\frac{2\left(3\sqrt{-4ac+b^2}\,abc-\sqrt{-4ac+b^2}\,b^3-4a^2c^2+5a\,b^2c-b^4\right)\arctan\left(\frac{2a\tan\left(\frac{x}{2}\right)+b+\sqrt{-4ac+b^2}}{\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}\right)}{a\left(4ac-b^2\right)\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}-\frac{2\left(-3\sqrt{-4ac+b^2}\,abc+\sqrt{-4ac+b^2}\,b \cdots}{a\left(4ac \cdots\right.}}{a}$
risch	Expression too large to display

```
[In] int(csc(x)^2/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 1/2*tan(1/2*x)/a+2/a*((3*(-4*a*c+b^2)^(1/2)*a*b*c-(-4*a*c+b^2)^(1/2)*b^3-4*
a^2*c^2+5*a*b^2*c-b^4)/a/(4*a*c-b^2)/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*
a^2)^(1/2)*arctan((2*a*tan(1/2*x)+b+(-4*a*c+b^2)^(1/2))/(4*a*c-2*b^2-2*b*(-
4*a*c+b^2)^(1/2)+4*a^2)^(1/2))-(-3*(-4*a*c+b^2)^(1/2)*a*b*c+(-4*a*c+b^2)^(1
/2)*b^3-4*a^2*c^2+5*a*b^2*c-b^4)/a/(4*a*c-b^2)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2
)^(1/2)+4*a^2)^(1/2)*arctan((-2*a*tan(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4*a*c-2
*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)))-1/2/a/tan(1/2*x)-1/a^2*b*ln(tan(
1/2*x))
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 6851 vs. 2(237) = 474.

Time = 258.46 (sec) , antiderivative size = 6851, normalized size of antiderivative = 25.28

$$\int \frac{\csc^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(csc(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] Too large to include
```

Sympy [F]

$$\int \frac{\csc^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx=\int \frac{\csc^2{(x)}}{a+b\sin{(x)}+c\sin^2{(x)}}\,dx$$

```
[In] integrate(csc(x)**2/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Integral(csc(x)**2/(a + b*sin(x) + c*sin(x)**2), x)
```

Maxima [F]

$$\int \frac{\csc^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx=\int \frac{\csc{(x)}^2}{c\sin{(x)}^2+b\sin{(x)}+a}\,dx$$

```
[In] integrate(csc(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] 1/2*(2*(a^2*cos(2*x)^2 + a^2*sin(2*x)^2 - 2*a^2*cos(2*x) + a^2)*integrate(2
*(2*b^2*c*cos(3*x)^2 + 2*b^2*c*cos(x)^2 + 2*b^2*c*sin(3*x)^2 + 2*b^2*c*sin(
x)^2 + b*c^2*sin(x) + 4*(2*a*b^2 - a*c^2 - (2*a^2 - b^2)*c)*cos(2*x)^2 + 2*
(2*b^3 + b*c^2)*cos(x)*sin(2*x) + 4*(2*a*b^2 - a*c^2 - (2*a^2 - b^2)*c)*sin
(2*x)^2 - (b*c^2*sin(3*x) - b*c^2*sin(x) + 2*(b^2*c - a*c^2)*cos(2*x))*cos(
4*x) - 2*(2*b^2*c*cos(x) + (2*b^3 + b*c^2)*sin(2*x))*cos(3*x) - 2*(b^2*c -
a*c^2 + (2*b^3 + b*c^2)*sin(x))*cos(2*x) + (b*c^2*cos(3*x) - b*c^2*cos(x) -
 2*(b^2*c - a*c^2)*sin(2*x))*sin(4*x) - (4*b^2*c*sin(x) + b*c^2 - 2*(2*b^3
+ b*c^2)*cos(2*x))*sin(3*x))/(a^2*c^2*cos(4*x)^2 + 4*a^2*b^2*cos(3*x)^2 + 4
*a^2*b^2*cos(x)^2 + a^2*c^2*sin(4*x)^2 + 4*a^2*b^2*sin(3*x)^2 + 4*a^2*b^2*s
in(x)^2 + 4*a^2*b*c*sin(x) + a^2*c^2 + 4*(4*a^4 + 4*a^3*c + a^2*c^2)*cos(2*
x)^2 + 8*(2*a^3*b + a^2*b*c)*cos(x)*sin(2*x) + 4*(4*a^4 + 4*a^3*c + a^2*c^2
)*sin(2*x)^2 - 2*(2*a^2*b*c*sin(3*x) - 2*a^2*b*c*sin(x) - a^2*c^2 + 2*(2*a^
3*c + a^2*c^2)*cos(2*x))*cos(4*x) - 8*(a^2*b^2*cos(x) + (2*a^3*b + a^2*b*c)
*sin(2*x))*cos(3*x) - 4*(2*a^3*c + a^2*c^2 + 2*(2*a^3*b + a^2*b*c)*sin(x))*
cos(2*x) + 4*(a^2*b*c*cos(3*x) - a^2*b*c*cos(x) - (2*a^3*c + a^2*c^2)*sin(2
*x))*sin(4*x) - 4*(2*a^2*b^2*sin(x) + a^2*b*c - 2*(2*a^3*b + a^2*b*c)*cos(2
*x))*sin(3*x)), x) + (b*cos(2*x)^2 + b*sin(2*x)^2 - 2*b*cos(2*x) + b)*log(c
os(x)^2 + sin(x)^2 + 2*cos(x) + 1) - (b*cos(2*x)^2 + b*sin(2*x)^2 - 2*b*cos
(2*x) + b)*log(cos(x)^2 + sin(x)^2 - 2*cos(x) + 1) - 4*a*sin(2*x))/(a^2*cos
(2*x)^2 + a^2*sin(2*x)^2 - 2*a^2*cos(2*x) + a^2)
```

Giac [F(-1)]

Timed out.

$$\int \frac{\csc^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(csc(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 25.09 (sec) , antiderivative size = 16102, normalized size of antiderivative = 59.42

$$\int \frac{\csc^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(1/(sin(x)^2*(a + c*sin(x)^2 + b*sin(x))),x)
[Out] tan(x/2)/(2*a) - 1/(2*a*tan(x/2)) - atan(((((-(b^8 + 8*a^3*c^5 + 8*a^4*c^4
+ b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 3
3*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^
6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(
1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c
^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8
*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*((32*(4*a^5*b^4 - 8*a^3*b^6 + 16*a^5
*c^4 + 20*a^6*c^3 + 4*a^7*c^2 + 53*a^4*b^4*c - 17*a^6*b^2*c + 8*a^3*b^4*c^2
 - 36*a^4*b^2*c^3 - 89*a^5*b^2*c^2))/a^3 - ((32*(4*a^5*b^5 - 3*a^7*b^3 + 16
*a^6*b*c^3 - 25*a^6*b^3*c + 36*a^7*b*c^2 - 4*a^5*b^3*c^2 + 12*a^8*b*c))/a^3
 - (32*tan(x/2)*(8*a^9*c - 16*a^4*b^6 + 17*a^6*b^4 - 2*a^8*b^2 + 192*a^6*c^
4 + 384*a^7*c^3 + 200*a^8*c^2 + 144*a^5*b^4*c - 118*a^7*b^2*c + 16*a^4*b^4*
c^2 - 112*a^5*b^2*c^3 - 416*a^6*b^2*c^2))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c
^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4
+ 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a
*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3
)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^
6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2
- 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2)*(13*a^4*b^5 - 16*a
^2*b^7 - 2*a^6*b^3 + 128*a^3*b^5*c + 128*a^4*b*c^4 + 240*a^5*b*c^3 - 78*a^5
*b^3*c + 104*a^6*b*c^2 + 16*a^2*b^5*c^2 - 96*a^3*b^3*c^3 - 316*a^4*b^3*c^2
+ 8*a^7*b*c))/a^3) + (32*(a^3*b^5 - 4*a*b^7 + 4*a*b^5*c^2 + 31*a^2*b^5*c +
28*a^3*b*c^4 + 35*a^4*b*c^3 - 5*a^4*b^3*c + 4*a^5*b*c^2 - 24*a^2*b^3*c^3 -
68*a^3*b^3*c^2))/a^3 + (32*tan(x/2)*(3*a^2*b^6 + 80*a^3*c^5 + 80*a^4*c^4 +
2*a^5*c^3 + 16*a*b^4*c^3 - 18*a^3*b^4*c - 88*a^2*b^2*c^4 + 116*a^2*b^4*c^2
- 224*a^3*b^2*c^3 + 23*a^4*b^2*c^2 - 16*a*b^6*c))/a^3)*(-(b^8 + 8*a^3*c^5 +
```

```
 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*
b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2
) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c
- b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6
 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*
b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*(3*b^6*c + 4*a^2*c^
5 + a^3*c^4 - 4*b^4*c^3 + 12*a*b^2*c^4 - 15*a*b^4*c^2 + 14*a^2*b^2*c^3))/a^
3 + (32*tan(x/2)*(8*b^5*c^2 - 8*b^3*c^4 - b^7 - 32*a*b^3*c^3 + 12*a^2*b*c^4
 + 2*a^3*b*c^3 - 9*a^2*b^3*c^2 + 16*a*b*c^5 + 6*a*b^5*c))/a^3)*(-(b^8 + 8*a
^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 -
 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)
^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(
-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 -
 a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*
c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*1i + ((32*(3*b^6*
c + 4*a^2*c^5 + a^3*c^4 - 4*b^4*c^3 + 12*a*b^2*c^4 - 15*a*b^4*c^2 + 14*a^2*
b^2*c^3))/a^3 - ((32*(a^3*b^5 - 4*a*b^7 + 4*a*b^5*c^2 + 31*a^2*b^5*c + 28*a
^3*b*c^4 + 35*a^4*b*c^3 - 5*a^4*b^3*c + 4*a^5*b*c^2 - 24*a^2*b^3*c^3 - 68*a
^3*b^3*c^2))/a^3 - (-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^
(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^
2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*
c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c
 - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*
c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2
*c^2)))^(1/2)*((32*(4*a^5*b^4 - 8*a^3*b^6 + 16*a^5*c^4 + 20*a^6*c^3 + 4*a^7
*c^2 + 53*a^4*b^4*c - 17*a^6*b^2*c + 8*a^3*b^4*c^2 - 36*a^4*b^2*c^3 - 89*a^
5*b^2*c^2))/a^3 + ((32*(4*a^5*b^5 - 3*a^7*b^3 + 16*a^6*b*c^3 - 25*a^6*b^3*c
 + 36*a^7*b*c^2 - 4*a^5*b^3*c^2 + 12*a^8*b*c))/a^3 - (32*tan(x/2)*(8*a^9*c
- 16*a^4*b^6 + 17*a^6*b^4 - 2*a^8*b^2 + 192*a^6*c^4 + 384*a^7*c^3 + 200*a^8
*c^2 + 144*a^5*b^4*c - 118*a^7*b^2*c + 16*a^4*b^4*c^2 - 112*a^5*b^2*c^3 - 4
16*a^6*b^2*c^2))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^
3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3
*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4
*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*
a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a
^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*
b^2*c^2)))^(1/2) + (32*tan(x/2)*(13*a^4*b^5 - 16*a^2*b^7 - 2*a^6*b^3 + 128*
a^3*b^5*c + 128*a^4*b*c^4 + 240*a^5*b*c^3 - 78*a^5*b^3*c + 104*a^6*b*c^2 +
16*a^2*b^5*c^2 - 96*a^3*b^3*c^3 - 316*a^4*b^3*c^2 + 8*a^7*b*c))/a^3) + (32*
tan(x/2)*(3*a^2*b^6 + 80*a^3*c^5 + 80*a^4*c^4 + 2*a^5*c^3 + 16*a*b^4*c^3 -
18*a^3*b^4*c - 88*a^2*b^2*c^4 + 116*a^2*b^4*c^2 - 224*a^3*b^2*c^3 + 23*a^4*
b^2*c^2 - 16*a*b^6*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c -
 b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 -
38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^
2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c
```

```
*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3
+ 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 3
2*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2)*(8*b^5*c^2 - 8*b^3*c^4 - b^7 - 32*a*b
^3*c^3 + 12*a^2*b*c^4 + 2*a^3*b*c^3 - 9*a^2*b^3*c^2 + 16*a*b*c^5 + 6*a*b^5*
c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^
6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^
3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3
)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)
^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a
^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1
/2)*1i)/((((-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) -
b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 -
b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)
^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^
3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10
*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^
(1/2)*((32*(4*a^5*b^4 - 8*a^3*b^6 + 16*a^5*c^4 + 20*a^6*c^3 + 4*a^7*c^2 + 5
3*a^4*b^4*c - 17*a^6*b^2*c + 8*a^3*b^4*c^2 - 36*a^4*b^2*c^3 - 89*a^5*b^2*c^
2))/a^3 - ((32*(4*a^5*b^5 - 3*a^7*b^3 + 16*a^6*b*c^3 - 25*a^6*b^3*c + 36*a^
7*b*c^2 - 4*a^5*b^3*c^2 + 12*a^8*b*c))/a^3 - (32*tan(x/2)*(8*a^9*c - 16*a^4
*b^6 + 17*a^6*b^4 - 2*a^8*b^2 + 192*a^6*c^4 + 384*a^7*c^3 + 200*a^8*c^2 + 1
44*a^5*b^4*c - 118*a^7*b^2*c + 16*a^4*b^4*c^2 - 112*a^5*b^2*c^3 - 416*a^6*b
^2*c^2))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2)
 - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3
 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b
^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^
2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 +
 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)
))^(1/2) + (32*tan(x/2)*(13*a^4*b^5 - 16*a^2*b^7 - 2*a^6*b^3 + 128*a^3*b^5*
c + 128*a^4*b*c^4 + 240*a^5*b*c^3 - 78*a^5*b^3*c + 104*a^6*b*c^2 + 16*a^2*b
^5*c^2 - 96*a^3*b^3*c^3 - 316*a^4*b^3*c^2 + 8*a^7*b*c))/a^3) + (32*(a^3*b^5
 - 4*a*b^7 + 4*a*b^5*c^2 + 31*a^2*b^5*c + 28*a^3*b*c^4 + 35*a^4*b*c^3 - 5*a
^4*b^3*c + 4*a^5*b*c^2 - 24*a^2*b^3*c^3 - 68*a^3*b^3*c^2))/a^3 + (32*tan(x/
2)*(3*a^2*b^6 + 80*a^3*c^5 + 80*a^4*c^4 + 2*a^5*c^3 + 16*a*b^4*c^3 - 18*a^3
*b^4*c - 88*a^2*b^2*c^4 + 116*a^2*b^4*c^2 - 224*a^3*b^2*c^3 + 23*a^4*b^2*c^
2 - 16*a*b^6*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^
3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3
*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4
*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*
a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a
^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*
b^2*c^2)))^(1/2) + (32*(3*b^6*c + 4*a^2*c^5 + a^3*c^4 - 4*b^4*c^3 + 12*a*b^
2*c^4 - 15*a*b^4*c^2 + 14*a^2*b^2*c^3))/a^3 + (32*tan(x/2)*(8*b^5*c^2 - 8*b
^3*c^4 - b^7 - 32*a*b^3*c^3 + 12*a^2*b*c^4 + 2*a^3*b*c^3 - 9*a^2*b^3*c^2 +
16*a*b*c^5 + 6*a*b^5*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c
```

```
 - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2
- 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*
c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3
*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^
3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 -
 32*a^6*b^2*c^2)))^(1/2) - ((32*(3*b^6*c + 4*a^2*c^5 + a^3*c^4 - 4*b^4*c^3
+ 12*a*b^2*c^4 - 15*a*b^4*c^2 + 14*a^2*b^2*c^3))/a^3 - ((32*(a^3*b^5 - 4*a*
b^7 + 4*a*b^5*c^2 + 31*a^2*b^5*c + 28*a^3*b*c^4 + 35*a^4*b*c^3 - 5*a^4*b^3*
c + 4*a^5*b*c^2 - 24*a^2*b^3*c^3 - 68*a^3*b^3*c^2))/a^3 - (-(b^8 + 8*a^3*c^
5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a
^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(
1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a
*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*
b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a
^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*((32*(4*a^5*b^4 - 8*a^
3*b^6 + 16*a^5*c^4 + 20*a^6*c^3 + 4*a^7*c^2 + 53*a^4*b^4*c - 17*a^6*b^2*c +
 8*a^3*b^4*c^2 - 36*a^4*b^2*c^3 - 89*a^5*b^2*c^2))/a^3 + ((32*(4*a^5*b^5 -
3*a^7*b^3 + 16*a^6*b*c^3 - 25*a^6*b^3*c + 36*a^7*b*c^2 - 4*a^5*b^3*c^2 + 12
*a^8*b*c))/a^3 - (32*tan(x/2)*(8*a^9*c - 16*a^4*b^6 + 17*a^6*b^4 - 2*a^8*b^
2 + 192*a^6*c^4 + 384*a^7*c^3 + 200*a^8*c^2 + 144*a^5*b^4*c - 118*a^7*b^2*c
 + 16*a^4*b^4*c^2 - 112*a^5*b^2*c^3 - 416*a^6*b^2*c^2))/a^3)*(-(b^8 + 8*a^3
*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 1
8*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3
)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(
4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a
^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c
+ a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2)*(13*
a^4*b^5 - 16*a^2*b^7 - 2*a^6*b^3 + 128*a^3*b^5*c + 128*a^4*b*c^4 + 240*a^5*
b*c^3 - 78*a^5*b^3*c + 104*a^6*b*c^2 + 16*a^2*b^5*c^2 - 96*a^3*b^3*c^3 - 31
6*a^4*b^3*c^2 + 8*a^7*b*c))/a^3) + (32*tan(x/2)*(3*a^2*b^6 + 80*a^3*c^5 + 8
0*a^4*c^4 + 2*a^5*c^3 + 16*a*b^4*c^3 - 18*a^3*b^4*c - 88*a^2*b^2*c^4 + 116*
a^2*b^4*c^2 - 224*a^3*b^2*c^3 + 23*a^4*b^2*c^2 - 16*a*b^6*c))/a^3)*(-(b^8 +
 8*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c
^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c -
b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c
^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b
^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*
b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2
)*(8*b^5*c^2 - 8*b^3*c^4 - b^7 - 32*a*b^3*c^3 + 12*a^2*b*c^4 + 2*a^3*b*c^3
- 9*a^2*b^3*c^2 + 16*a*b*c^5 + 6*a*b^5*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*
c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4
 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*
a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^2)^
3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a
^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2
```

 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (64*(4*b*c^5 - b^3*c^3 + a*b*c
^4))/a^3 + (64*tan(x/2)*(8*c^6 - 4*b^2*c^4))/a^3))*(-(b^8 + 8*a^3*c^5 + 8*a
^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*
c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) -
10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 2*a*b*c^3*(-(4*a*c - b^
2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 1
6*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*
c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*2i - atan(((((-(b^8 + 8*a^3*c
^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*
a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^
(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*
a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4
*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c +
a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*((32*(4*a^5*b^4 - 8*a
^3*b^6 + 16*a^5*c^4 + 20*a^6*c^3 + 4*a^7*c^2 + 53*a^4*b^4*c - 17*a^6*b^2*c
+ 8*a^3*b^4*c^2 - 36*a^4*b^2*c^3 - 89*a^5*b^2*c^2))/a^3 - ((32*(4*a^5*b^5 -
 3*a^7*b^3 + 16*a^6*b*c^3 - 25*a^6*b^3*c + 36*a^7*b*c^2 - 4*a^5*b^3*c^2 + 1
2*a^8*b*c))/a^3 - (32*tan(x/2)*(8*a^9*c - 16*a^4*b^6 + 17*a^6*b^4 - 2*a^8*b
^2 + 192*a^6*c^4 + 384*a^7*c^3 + 200*a^8*c^2 + 144*a^5*b^4*c - 118*a^7*b^2*
c + 16*a^4*b^4*c^2 - 112*a^5*b^2*c^3 - 416*a^6*b^2*c^2))/a^3)*(-(b^8 + 8*a^
3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 -
18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^
3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-
(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 -
a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c
 + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2)*(13
*a^4*b^5 - 16*a^2*b^7 - 2*a^6*b^3 + 128*a^3*b^5*c + 128*a^4*b*c^4 + 240*a^5
*b*c^3 - 78*a^5*b^3*c + 104*a^6*b*c^2 + 16*a^2*b^5*c^2 - 96*a^3*b^3*c^3 - 3
16*a^4*b^3*c^2 + 8*a^7*b*c))/a^3) + (32*(a^3*b^5 - 4*a*b^7 + 4*a*b^5*c^2 +
31*a^2*b^5*c + 28*a^3*b*c^4 + 35*a^4*b*c^3 - 5*a^4*b^3*c + 4*a^5*b*c^2 - 24
*a^2*b^3*c^3 - 68*a^3*b^3*c^2))/a^3 + (32*tan(x/2)*(3*a^2*b^6 + 80*a^3*c^5
+ 80*a^4*c^4 + 2*a^5*c^3 + 16*a*b^4*c^3 - 18*a^3*b^4*c - 88*a^2*b^2*c^4 + 1
16*a^2*b^4*c^2 - 224*a^3*b^2*c^3 + 23*a^4*b^2*c^2 - 16*a*b^6*c))/a^3)*(-(b^
8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^
4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c
 - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*
b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^
6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a
^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*(3*b
^6*c + 4*a^2*c^5 + a^3*c^4 - 4*b^4*c^3 + 12*a*b^2*c^4 - 15*a*b^4*c^2 + 14*a
^2*b^2*c^3))/a^3 + (32*tan(x/2)*(8*b^5*c^2 - 8*b^3*c^4 - b^7 - 32*a*b^3*c^3
 + 12*a^2*b*c^4 + 2*a^3*b*c^3 - 9*a^2*b^3*c^2 + 16*a*b*c^5 + 6*a*b^5*c))/a^
3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2
+ 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*
(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2

) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2)
)/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4
*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*1i
 + ((32*(3*b^6*c + 4*a^2*c^5 + a^3*c^4 - 4*b^4*c^3 + 12*a*b^2*c^4 - 15*a*b^
4*c^2 + 14*a^2*b^2*c^3))/a^3 - ((32*(a^3*b^5 - 4*a*b^7 + 4*a*b^5*c^2 + 31*a
^2*b^5*c + 28*a^3*b*c^4 + 35*a^4*b*c^3 - 5*a^4*b^3*c + 4*a^5*b*c^2 - 24*a^2
*b^3*c^3 - 68*a^3*b^3*c^2))/a^3 - (-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4
*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*
c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^
2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a
*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^
7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c
^3 - 32*a^6*b^2*c^2)))^(1/2)*((32*(4*a^5*b^4 - 8*a^3*b^6 + 16*a^5*c^4 + 20*
a^6*c^3 + 4*a^7*c^2 + 53*a^4*b^4*c - 17*a^6*b^2*c + 8*a^3*b^4*c^2 - 36*a^4*
b^2*c^3 - 89*a^5*b^2*c^2))/a^3 + ((32*(4*a^5*b^5 - 3*a^7*b^3 + 16*a^6*b*c^3
 - 25*a^6*b^3*c + 36*a^7*b*c^2 - 4*a^5*b^3*c^2 + 12*a^8*b*c))/a^3 - (32*tan
(x/2)*(8*a^9*c - 16*a^4*b^6 + 17*a^6*b^4 - 2*a^8*b^2 + 192*a^6*c^4 + 384*a^
7*c^3 + 200*a^8*c^2 + 144*a^5*b^4*c - 118*a^7*b^2*c + 16*a^4*b^4*c^2 - 112*
a^5*b^2*c^3 - 416*a^6*b^2*c^2))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(
-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b
^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3
*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) +
4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32
*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^
2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2)*(13*a^4*b^5 - 16*a^2*b^7 - 2
*a^6*b^3 + 128*a^3*b^5*c + 128*a^4*b*c^4 + 240*a^5*b*c^3 - 78*a^5*b^3*c + 1
04*a^6*b*c^2 + 16*a^2*b^5*c^2 - 96*a^3*b^3*c^3 - 316*a^4*b^3*c^2 + 8*a^7*b*
c))/a^3) + (32*tan(x/2)*(3*a^2*b^6 + 80*a^3*c^5 + 80*a^4*c^4 + 2*a^5*c^3 +
16*a*b^4*c^3 - 18*a^3*b^4*c - 88*a^2*b^2*c^4 + 116*a^2*b^4*c^2 - 224*a^3*b^
2*c^3 + 23*a^4*b^2*c^2 - 16*a*b^6*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 -
 b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33
*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6
*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1
/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^
4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*
a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2)*(8*b^5*c^2 - 8*b^3*c^4
 - b^7 - 32*a*b^3*c^3 + 12*a^2*b*c^4 + 2*a^3*b*c^3 - 9*a^2*b^3*c^2 + 16*a*b
*c^5 + 6*a*b^5*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2
)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a
^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-
(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(
4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16
*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^
6*b^2*c^2)))^(1/2)*1i)/((((-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b
^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38

*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*
(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(
-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 +
16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*
a^6*b^2*c^2)))^(1/2)*((32*(4*a^5*b^4 - 8*a^3*b^6 + 16*a^5*c^4 + 20*a^6*c^3
+ 4*a^7*c^2 + 53*a^4*b^4*c - 17*a^6*b^2*c + 8*a^3*b^4*c^2 - 36*a^4*b^2*c^3
- 89*a^5*b^2*c^2))/a^3 - ((32*(4*a^5*b^5 - 3*a^7*b^3 + 16*a^6*b*c^3 - 25*a^
6*b^3*c + 36*a^7*b*c^2 - 4*a^5*b^3*c^2 + 12*a^8*b*c))/a^3 - (32*tan(x/2)*(8
*a^9*c - 16*a^4*b^6 + 17*a^6*b^4 - 2*a^8*b^2 + 192*a^6*c^4 + 384*a^7*c^3 +
200*a^8*c^2 + 144*a^5*b^4*c - 118*a^7*b^2*c + 16*a^4*b^4*c^2 - 112*a^5*b^2*
c^3 - 416*a^6*b^2*c^2))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c
- b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 -
38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c
^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*
c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3
+ 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 -
32*a^6*b^2*c^2)))^(1/2) + (32*tan(x/2)*(13*a^4*b^5 - 16*a^2*b^7 - 2*a^6*b^3
+ 128*a^3*b^5*c + 128*a^4*b*c^4 + 240*a^5*b*c^3 - 78*a^5*b^3*c + 104*a^6*b
*c^2 + 16*a^2*b^5*c^2 - 96*a^3*b^3*c^3 - 316*a^4*b^3*c^2 + 8*a^7*b*c))/a^3)
+ (32*(a^3*b^5 - 4*a*b^7 + 4*a*b^5*c^2 + 31*a^2*b^5*c + 28*a^3*b*c^4 + 35*
a^4*b*c^3 - 5*a^4*b^3*c + 4*a^5*b*c^2 - 24*a^2*b^3*c^3 - 68*a^3*b^3*c^2))/a
^3 + (32*tan(x/2)*(3*a^2*b^6 + 80*a^3*c^5 + 80*a^4*c^4 + 2*a^5*c^3 + 16*a*b
^4*c^3 - 18*a^3*b^4*c - 88*a^2*b^2*c^4 + 116*a^2*b^4*c^2 - 224*a^3*b^2*c^3
+ 23*a^4*b^2*c^2 - 16*a*b^6*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(
-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b
^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3
*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) +
4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32
*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^
2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (32*(3*b^6*c + 4*a^2*c^5 + a^3*c^4 - 4*b^
4*c^3 + 12*a*b^2*c^4 - 15*a*b^4*c^2 + 14*a^2*b^2*c^3))/a^3 + (32*tan(x/2)*(
8*b^5*c^2 - 8*b^3*c^4 - b^7 - 32*a*b^3*c^3 + 12*a^2*b*c^4 + 2*a^3*b*c^3 - 9
*a^2*b^3*c^2 + 16*a*b*c^5 + 6*a*b^5*c))/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4
- b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 +
33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b
^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^
(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*
c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 -
8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) - ((32*(3*b^6*c + 4*a^2*c^5 + a^3*c
^4 - 4*b^4*c^3 + 12*a*b^2*c^4 - 15*a*b^4*c^2 + 14*a^2*b^2*c^3))/a^3 - ((32*
(a^3*b^5 - 4*a*b^7 + 4*a*b^5*c^2 + 31*a^2*b^5*c + 28*a^3*b*c^4 + 35*a^4*b*c
^3 - 5*a^4*b^3*c + 4*a^5*b*c^2 - 24*a^2*b^3*c^3 - 68*a^3*b^3*c^2))/a^3 - (-
(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a
*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*
a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2

```
*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*
(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c -
8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*((32*(4
*a^5*b^4 - 8*a^3*b^6 + 16*a^5*c^4 + 20*a^6*c^3 + 4*a^7*c^2 + 53*a^4*b^4*c -
 17*a^6*b^2*c + 8*a^3*b^4*c^2 - 36*a^4*b^2*c^3 - 89*a^5*b^2*c^2))/a^3 + ((3
2*(4*a^5*b^5 - 3*a^7*b^3 + 16*a^6*b*c^3 - 25*a^6*b^3*c + 36*a^7*b*c^2 - 4*a
^5*b^3*c^2 + 12*a^8*b*c))/a^3 - (32*tan(x/2)*(8*a^9*c - 16*a^4*b^6 + 17*a^6
*b^4 - 2*a^8*b^2 + 192*a^6*c^4 + 384*a^7*c^3 + 200*a^8*c^2 + 144*a^5*b^4*c
- 118*a^7*b^2*c + 16*a^4*b^4*c^2 - 112*a^5*b^2*c^3 - 416*a^6*b^2*c^2))/a^3)
*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 +
8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-
(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2)
- 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/
(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c
 - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (3
2*tan(x/2)*(13*a^4*b^5 - 16*a^2*b^7 - 2*a^6*b^3 + 128*a^3*b^5*c + 128*a^4*b
*c^4 + 240*a^5*b*c^3 - 78*a^5*b^3*c + 104*a^6*b*c^2 + 16*a^2*b^5*c^2 - 96*a
^3*b^3*c^3 - 316*a^4*b^3*c^2 + 8*a^7*b*c))/a^3) + (32*tan(x/2)*(3*a^2*b^6 +
 80*a^3*c^5 + 80*a^4*c^4 + 2*a^5*c^3 + 16*a*b^4*c^3 - 18*a^3*b^4*c - 88*a^2
*b^2*c^4 + 116*a^2*b^4*c^2 - 224*a^3*b^2*c^3 + 23*a^4*b^2*c^2 - 16*a*b^6*c)
)/a^3)*(-(b^8 + 8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*
c^2 + 8*a*b^4*c^3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*
c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^
(1/2) - 2*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(
1/2))/(2*(a^6*b^4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5
*b^4*c - 8*a^7*b^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2
) + (32*tan(x/2)*(8*b^5*c^2 - 8*b^3*c^4 - b^7 - 32*a*b^3*c^3 + 12*a^2*b*c^4
 + 2*a^3*b*c^3 - 9*a^2*b^3*c^2 + 16*a*b*c^5 + 6*a*b^5*c))/a^3)*(-(b^8 + 8*a
^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^3 -
 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b^2)
^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^3*(
-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^4 -
 a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b^2*
c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2) + (64*(4*b*c^5 -
b^3*c^3 + a*b*c^4))/a^3 + (64*tan(x/2)*(8*c^6 - 4*b^2*c^4))/a^3))*(-(b^8 +
8*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - b^6*c^2 + 8*a*b^4*c^
3 - 18*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + b^3*c^2*(-(4*a*c - b
^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 2*a*b*c^
3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^6*b^
4 - a^4*b^6 + 16*a^6*c^4 + 32*a^7*c^3 + 16*a^8*c^2 + 10*a^5*b^4*c - 8*a^7*b
^2*c + a^4*b^4*c^2 - 8*a^5*b^2*c^3 - 32*a^6*b^2*c^2)))^(1/2)*2i - (b*log(ta
n(x/2)))/a^2
```

3.8 $\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 331

$$\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\sqrt{2}c\left(b^3-3abc+\sqrt{b^2-4ac}(b^2-ac)\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{a^3\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} + \frac{\sqrt{2}c\left(b^3-3abc-\sqrt{b^2-4ac}(b^2-ac)\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{a^3\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} - \frac{\operatorname{arctanh}(\cos(x))}{2a} - \frac{(b^2-ac)\operatorname{arctanh}(\cos(x))}{a^3} + \frac{b\cot(x)}{a^2} - \frac{\cot(x)\csc(x)}{2a}$$

```
[Out] -1/2*arctanh(cos(x))/a-(-a*c+b^2)*arctanh(cos(x))/a^3+b*cot(x)/a^2-1/2*cot(
x)*csc(x)/a-c*arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b
^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(b^3-3*a*b*c+(-a*c+b^2)*(
-4*a*c+b^2)^(1/2))/a^3/(-4*a*c+b^2)^(1/2)/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/
2))^(1/2)+c*arctan(1/2*(2*c+(b+(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2
-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(b^3-3*a*b*c-(-a*c+b^2)*(-4
*a*c+b^2)^(1/2))/a^3/(-4*a*c+b^2)^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2)
)^(1/2)
```

Rubi [A] (verified)

Time = 3.51 (sec) , antiderivative size = 331, normalized size of antiderivative = 1.00, number of steps used = 14, number of rules used = 9, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.474, Rules used = {3337, 3855, 3852, 8, 3853, 3373, 2739, 632, 210}

$$\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\sqrt{2}c\left(\sqrt{b^2-4ac}(b^2-ac)-3abc+b^3\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{a^3\sqrt{b^2-4ac}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{\sqrt{2}c\left(-\sqrt{b^2-4ac}(b^2-ac)-3abc+b^3\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{a^3\sqrt{b^2-4ac}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{(b^2-ac)\,\text{arctanh}(\cos(x))}{a^3} + \frac{b\cot(x)}{a^2} - \frac{\text{arctanh}(\cos(x))}{2a} - \frac{\cot(x)\csc(x)}{2a}$$

```
[In] Int[Csc[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -((Sqrt[2]*c*(b^3 - 3*a*b*c + Sqrt[b^2 - 4*a*c]*(b^2 - a*c))*ArcTan[(2*c +
(b - Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[
b^2 - 4*a*c]])])/(a^3*Sqrt[b^2 - 4*a*c]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2
 - 4*a*c]])) + (Sqrt[2]*c*(b^3 - 3*a*b*c - Sqrt[b^2 - 4*a*c]*(b^2 - a*c))*A
rcTan[(2*c + (b + Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a +
 c) + b*Sqrt[b^2 - 4*a*c]])])/(a^3*Sqrt[b^2 - 4*a*c]*Sqrt[b^2 - 2*c*(a + c)
 + b*Sqrt[b^2 - 4*a*c]]) - ArcTanh[Cos[x]]/(2*a) - ((b^2 - a*c)*ArcTanh[Cos
[x]])/a^3 + (b*Cot[x])/a^2 - (Cot[x]*Csc[x])/(2*a)
```

Rule 8

```
Int[a_, x_Symbol] :> Simp[a*x, x] /; FreeQ[a, x]
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3337

```
Int[sin[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_), x_Symbol] :> Int[ExpandTr
ig[sin[d + e*x]^m*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n))^p, x], x] /
; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && NeQ[b^2 - 4*a*c, 0] && Integ
ersQ[m, n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3852

```
Int[csc[(c_.) + (d_.)*(x_)]^(n_), x_Symbol] :> Dist[-d^(-1), Subst[Int[Expa
ndIntegrand[(1 + x^2)^(n/2 - 1), x], x], x, Cot[c + d*x]], x] /; FreeQ[{c,
d}, x] && IGtQ[n/2, 0]
```

Rule 3853

```
Int[(csc[(c_.) + (d_.)*(x_)]*(b_.))^(n_), x_Symbol] :> Simp[(-b)*Cos[c + d*
x]*((b*Csc[c + d*x])^(n - 1)/(d*(n - 1))), x] + Dist[b^2*((n - 2)/(n - 1)),
 Int[(b*Csc[c + d*x])^(n - 2), x], x] /; FreeQ[{b, c, d}, x] && GtQ[n, 1] &
& IntegerQ[2*n]
```

Rule 3855

```
Int[csc[(c_.) + (d_.)*(x_)], x_Symbol] :> Simp[-ArcTanh[Cos[c + d*x]]/d, x]
 /; FreeQ[{c, d}, x]
```

Rubi steps

integral

$$= \int \left(\frac{(b^2-ac)\csc(x)}{a^3} - \frac{b\csc^2(x)}{a^2} + \frac{\csc^3(x)}{a} + \frac{-b^3\left(1-\frac{2ac}{b^2}\right) - b^2c\left(1-\frac{ac}{b^2}\right)\sin(x)}{a^3\left(a+b\sin(x)+c\sin^2(x)\right)} \right) dx$$

$$= \frac{\int \frac{-b^3\left(1-\frac{2ac}{b^2}\right)-b^2c\left(1-\frac{ac}{b^2}\right)\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{a^3} + \frac{\int \csc^3(x)\,dx}{a} - \frac{b\int \csc^2(x)\,dx}{a^2} + \frac{(b^2-ac)\int \csc(x)\,dx}{a^3}$$

$$= -\frac{(b^2-ac)\operatorname{arctanh}(\cos(x))}{a^3} - \frac{\cot(x)\csc(x)}{2a} + \frac{\int \csc(x)\,dx}{2a} + \frac{b\text{Subst}(\int 1\,dx, x, \cot(x))}{a^2}$$
$$+ \frac{\left(c\left(b^3-3abc-\sqrt{b^2-4ac}(b^2-ac)\right)\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{a^3\sqrt{b^2-4ac}}$$
$$- \frac{\left(c\left(b^3-3abc+\sqrt{b^2-4ac}(b^2-ac)\right)\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{a^3\sqrt{b^2-4ac}}$$

$$= -\frac{\operatorname{arctanh}(\cos(x))}{2a} - \frac{(b^2-ac)\operatorname{arctanh}(\cos(x))}{a^3} + \frac{b\cot(x)}{a^2} - \frac{\cot(x)\csc(x)}{2a}$$
$$+ \frac{\left(2c\left(b^3-3abc-\sqrt{b^2-4ac}(b^2-ac)\right)\right)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{a^3\sqrt{b^2-4ac}}$$
$$- \frac{\left(2c\left(b^3-3abc+\sqrt{b^2-4ac}(b^2-ac)\right)\right)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{a^3\sqrt{b^2-4ac}}$$

$$= -\frac{\operatorname{arctanh}(\cos(x))}{2a} - \frac{(b^2-ac)\operatorname{arctanh}(\cos(x))}{a^3} + \frac{b\cot(x)}{a^2} - \frac{\cot(x)\csc(x)}{2a}$$
$$- \frac{\left(4c\left(b^3-3abc-\sqrt{b^2-4ac}(b^2-ac)\right)\right)\text{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx, x, 4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\right.}{a^3\sqrt{b^2-4ac}}$$
$$+ \frac{\left(4c\left(b^3-3abc+\sqrt{b^2-4ac}(b^2-ac)\right)\right)\text{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx, x, 4c+2\left(b-\sqrt{b^2-4ac}\right)\right.}{a^3\sqrt{b^2-4ac}}$$

$$= -\frac{\sqrt{2}c\left(b^3-3abc+\sqrt{b^2-4ac}(b^2-ac)\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{a^3\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}$$
$$+ \frac{\sqrt{2}c\left(b^3-3abc-\sqrt{b^2-4ac}(b^2-ac)\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{a^3\sqrt{b^2-4ac}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}$$
$$- \frac{\operatorname{arctanh}(\cos(x))}{2a} - \frac{(b^2-ac)\operatorname{arctanh}(\cos(x))}{a^3} + \frac{b\cot(x)}{a^2} - \frac{\cot(x)\csc(x)}{2a}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 2.94 (sec) , antiderivative size = 481, normalized size of antiderivative = 1.45

$$\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$
$$= \frac{\csc^2(x)(-2a-c+c\cos(2x)-2b\sin(x))\left(\frac{8c\left(-ib^3+3iabc+b^2\sqrt{-b^2+4ac}-ac\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right.}{}$$

```
[In] Integrate[Csc[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (Csc[x]^2*(-2*a - c + c*Cos[2*x] - 2*b*Sin[x])*((8*c*((-I)*b^3 + (3*I)*a*b*
c + b^2*Sqrt[-b^2 + 4*a*c] - a*c*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b - I*S
qrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b^
2 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[-b
^2 + 4*a*c]]) + (8*c*(I*b^3 - (3*I)*a*b*c + b^2*Sqrt[-b^2 + 4*a*c] - a*c*Sq
rt[-b^2 + 4*a*c])*ArcTan[(2*c + (b + I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[
2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a
*c]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]]) - 4*a*b*Cot[x/2] + a^
2*Csc[x/2]^2 + 4*(a^2 + 2*b^2 - 2*a*c)*Log[Cos[x/2]] - 4*(a^2 + 2*b^2 - 2*a
*c)*Log[Sin[x/2]] - a^2*Sec[x/2]^2 + 4*a*b*Tan[x/2]))/(16*a^3*(c + b*Csc[x]
 + a*Csc[x]^2))
```

Maple [A] (verified)

Time = 4.32 (sec) , antiderivative size = 414, normalized size of antiderivative = 1.25

method	result
default	$\frac{\frac{a\left(\tan^2\left(\frac{x}{2}\right)\right)}{2}-2b\tan\left(\frac{x}{2}\right)}{4a^2}+\frac{-\frac{2\left(-2\sqrt{-4ac+b^2}\,a^2c^2+4\sqrt{-4ac+b^2}\,b^2ca-\sqrt{-4ac+b^2}\,b^4+8a^2bc^2-6ab^3c+b^5\right)\arctan\left(\frac{-2a\tan\left(\frac{x}{2}\right)+\sqrt{-4ac+b^2}}{\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}}}\right.}{a\left(4ac-b^2\right)\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}}{}$
risch	Expression too large to display

```
[In] int(csc(x)^3/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 1/4/a^2*(1/2*a*tan(1/2*x)^2-2*b*tan(1/2*x))+2/a^2*(-(-2*(-4*a*c+b^2)^(1/2)*
a^2*c^2+4*(-4*a*c+b^2)^(1/2)*b^2*c*a-(-4*a*c+b^2)^(1/2)*b^4+8*a^2*b*c^2-6*a
*b^3*c+b^5)/a/(4*a*c-b^2)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*
arctan((-2*a*tan(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4*a*c-2*b^2+2*b*(-4*a*c+b^2)
^(1/2)+4*a^2)^(1/2))+(2*(-4*a*c+b^2)^(1/2)*a^2*c^2-4*(-4*a*c+b^2)^(1/2)*b^2
```

```
*c*a+(-4*a*c+b^2)^(1/2)*b^4+8*a^2*b*c^2-6*a*b^3*c+b^5)/a/(4*a*c-b^2)/(4*a*c
-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((2*a*tan(1/2*x)+b+(-4*a*c
+b^2)^(1/2))/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)))-1/8/a/tan(1
/2*x)^2+1/4/a^3*(2*a^2-4*a*c+4*b^2)*ln(tan(1/2*x))+1/2/a^2*b/tan(1/2*x)
```

Fricas [F(-1)]

Timed out.

$$\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(csc(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
[Out] Timed out
```

Sympy [F]

$$\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\csc^3{(x)}}{a+b\sin{(x)}+c\sin^2{(x)}}\,dx$$

```
[In] integrate(csc(x)**3/(a+b*sin(x)+c*sin(x)**2),x)
[Out] Integral(csc(x)**3/(a + b*sin(x) + c*sin(x)**2), x)
```

Maxima [F]

$$\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\csc{(x)}^3}{c\sin{(x)}^2+b\sin{(x)}+a}\,dx$$

```
[In] integrate(csc(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
[Out] -1/4*(8*a^2*cos(2*x)*cos(x) + 8*a^2*sin(3*x)*sin(2*x) - 4*a^2*cos(x) - 4*(a
^2*cos(3*x) + a^2*cos(x) - 2*a*b*sin(2*x))*cos(4*x) + 4*(2*a^2*cos(2*x) - a
^2)*cos(3*x) - 4*(a^3*cos(4*x)^2 + 4*a^3*cos(2*x)^2 + a^3*sin(4*x)^2 - 4*a^
3*sin(4*x)*sin(2*x) + 4*a^3*sin(2*x)^2 - 4*a^3*cos(2*x) + a^3 - 2*(2*a^3*co
s(2*x) - a^3)*cos(4*x))*integrate(-2*(2*(b^3*c - a*b*c^2)*cos(3*x)^2 + 4*(2
*a*b^3 - 2*a*b*c^2 - (4*a^2*b - b^3)*c)*cos(2*x)^2 + 2*(b^3*c - a*b*c^2)*co
s(x)^2 + 2*(b^3*c - a*b*c^2)*sin(3*x)^2 + 2*(2*b^4 - 2*a*b^2*c - a*c^3 - (2
*a^2 - b^2)*c^2)*cos(x)*sin(2*x) + 4*(2*a*b^3 - 2*a*b*c^2 - (4*a^2*b - b^3)
*c)*sin(2*x)^2 + 2*(b^3*c - a*b*c^2)*sin(x)^2 - (2*(b^3*c - 2*a*b*c^2)*cos(
2*x) + (b^2*c^2 - a*c^3)*sin(3*x) - (b^2*c^2 - a*c^3)*sin(x))*cos(4*x) - 2*
(2*(b^3*c - a*b*c^2)*cos(x) + (2*b^4 - 2*a*b^2*c - a*c^3 - (2*a^2 - b^2)*c^
```

```
2)*sin(2*x))*cos(3*x) - 2*(b^3*c - 2*a*b*c^2 + (2*b^4 - 2*a*b^2*c - a*c^3 -
 (2*a^2 - b^2)*c^2)*sin(x))*cos(2*x) + ((b^2*c^2 - a*c^3)*cos(3*x) - (b^2*c
^2 - a*c^3)*cos(x) - 2*(b^3*c - 2*a*b*c^2)*sin(2*x))*sin(4*x) - (b^2*c^2 -
a*c^3 - 2*(2*b^4 - 2*a*b^2*c - a*c^3 - (2*a^2 - b^2)*c^2)*cos(2*x) + 4*(b^3
*c - a*b*c^2)*sin(x))*sin(3*x) + (b^2*c^2 - a*c^3)*sin(x))/(a^3*c^2*cos(4*x
)^2 + 4*a^3*b^2*cos(3*x)^2 + 4*a^3*b^2*cos(x)^2 + a^3*c^2*sin(4*x)^2 + 4*a^
3*b^2*sin(3*x)^2 + 4*a^3*b^2*sin(x)^2 + 4*a^3*b*c*sin(x) + a^3*c^2 + 4*(4*a
^5 + 4*a^4*c + a^3*c^2)*cos(2*x)^2 + 8*(2*a^4*b + a^3*b*c)*cos(x)*sin(2*x)
+ 4*(4*a^5 + 4*a^4*c + a^3*c^2)*sin(2*x)^2 - 2*(2*a^3*b*c*sin(3*x) - 2*a^3*
b*c*sin(x) - a^3*c^2 + 2*(2*a^4*c + a^3*c^2)*cos(2*x))*cos(4*x) - 8*(a^3*b^
2*cos(x) + (2*a^4*b + a^3*b*c)*sin(2*x))*cos(3*x) - 4*(2*a^4*c + a^3*c^2 +
2*(2*a^4*b + a^3*b*c)*sin(x))*cos(2*x) + 4*(a^3*b*c*cos(3*x) - a^3*b*c*cos(
x) - (2*a^4*c + a^3*c^2)*sin(2*x))*sin(4*x) - 4*(2*a^3*b^2*sin(x) + a^3*b*c
 - 2*(2*a^4*b + a^3*b*c)*cos(2*x))*sin(3*x)), x) + ((a^2 + 2*b^2 - 2*a*c)*c
os(4*x)^2 + 4*(a^2 + 2*b^2 - 2*a*c)*cos(2*x)^2 + (a^2 + 2*b^2 - 2*a*c)*sin(
4*x)^2 - 4*(a^2 + 2*b^2 - 2*a*c)*sin(4*x)*sin(2*x) + 4*(a^2 + 2*b^2 - 2*a*c
)*sin(2*x)^2 + a^2 + 2*b^2 - 2*a*c + 2*(a^2 + 2*b^2 - 2*a*c - 2*(a^2 + 2*b^
2 - 2*a*c)*cos(2*x))*cos(4*x) - 4*(a^2 + 2*b^2 - 2*a*c)*cos(2*x))*log(cos(x
)^2 + sin(x)^2 + 2*cos(x) + 1) - ((a^2 + 2*b^2 - 2*a*c)*cos(4*x)^2 + 4*(a^2
 + 2*b^2 - 2*a*c)*cos(2*x)^2 + (a^2 + 2*b^2 - 2*a*c)*sin(4*x)^2 - 4*(a^2 +
2*b^2 - 2*a*c)*sin(4*x)*sin(2*x) + 4*(a^2 + 2*b^2 - 2*a*c)*sin(2*x)^2 + a^2
 + 2*b^2 - 2*a*c + 2*(a^2 + 2*b^2 - 2*a*c - 2*(a^2 + 2*b^2 - 2*a*c)*cos(2*x
))*cos(4*x) - 4*(a^2 + 2*b^2 - 2*a*c)*cos(2*x))*log(cos(x)^2 + sin(x)^2 - 2
*cos(x) + 1) - 4*(2*a*b*cos(2*x) + a^2*sin(3*x) + a^2*sin(x) - 2*a*b)*sin(4
*x) + 8*(a^2*sin(x) - a*b)*sin(2*x))/(a^3*cos(4*x)^2 + 4*a^3*cos(2*x)^2 + a
^3*sin(4*x)^2 - 4*a^3*sin(4*x)*sin(2*x) + 4*a^3*sin(2*x)^2 - 4*a^3*cos(2*x)
 + a^3 - 2*(2*a^3*cos(2*x) - a^3)*cos(4*x))
```

Giac [F(-1)]

Timed out.

$$\int \frac{\csc^3(x)}{a + b\sin(x) + c\sin^2(x)}\, dx = \text{Timed out}$$

```
[In] integrate(csc(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 24.74 (sec) , antiderivative size = 21909, normalized size of antiderivative = 66.19

$$\int \frac{\csc^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(1/(sin(x)^3*(a + c*sin(x)^2 + b*sin(x))),x)
```

```
[Out] atan(-(((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*
c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*
a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*
c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(
1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2
)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16
*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*
c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*(((8*a^4*c^6 - b^10 + 8*a^5*c
^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4
 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*
c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^
(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^
3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b
^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2
 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^
2)))^(1/2)*(((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) +
 b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5
+ 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a
*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)
^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c
- b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6
 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6
*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*((16*(4*a^7*b^5 - 16*a^5
*b^7 + 3*a^9*b^3 + 122*a^6*b^5*c + 96*a^7*b*c^4 + 160*a^8*b*c^3 - 17*a^8*b^
3*c + 4*a^9*b*c^2 + 16*a^5*b^5*c^2 - 88*a^6*b^3*c^3 - 272*a^7*b^3*c^2 - 12*
a^10*b*c))/a^6 + ((16*(8*a^8*b^5 - 6*a^10*b^3 + 32*a^9*b*c^3 - 50*a^9*b^3*c
 + 72*a^10*b*c^2 - 8*a^8*b^3*c^2 + 24*a^11*b*c))/a^6 - (16*tan(x/2)*(16*a^1
2*c - 32*a^7*b^6 + 34*a^9*b^4 - 4*a^11*b^2 + 384*a^9*c^4 + 768*a^10*c^3 + 4
00*a^11*c^2 + 288*a^8*b^4*c - 236*a^10*b^2*c + 32*a^7*b^4*c^2 - 224*a^8*b^2
*c^3 - 832*a^9*b^2*c^2))/a^6)*((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c
 - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2
 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^
2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c
^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2
*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*
(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c -
```

8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*
tan(x/2)*(8*a^11*c - 32*a^4*b^8 + 18*a^6*b^6 + 5*a^8*b^4 - 2*a^10*b^2 - 192
*a^7*c^5 - 288*a^8*c^4 - 48*a^9*c^3 + 56*a^10*c^2 + 288*a^5*b^6*c - 118*a^7
*b^4*c - 34*a^9*b^2*c + 32*a^4*b^6*c^2 - 224*a^5*b^4*c^3 + 432*a^6*b^2*c^4
- 864*a^6*b^4*c^2 + 968*a^7*b^2*c^3 + 196*a^8*b^2*c^2))/a^6) + (16*(8*a^2*b
^9 + 2*a^4*b^7 - a^6*b^5 - 78*a^3*b^7*c + 104*a^5*b*c^5 - 18*a^5*b^5*c + 11
4*a^6*b*c^4 - 36*a^7*b*c^3 + 6*a^7*b^3*c - 8*a^8*b*c^2 - 8*a^2*b^7*c^2 + 64
*a^3*b^5*c^3 - 152*a^4*b^3*c^4 + 256*a^4*b^5*c^2 - 318*a^5*b^3*c^3 + 49*a^6
*b^3*c^2))/a^6 + (16*tan(x/2)*(2*a^3*b^8 - 4*a^5*b^6 + 96*a^5*c^6 + 96*a^6*
c^5 + 20*a^7*c^4 + 16*a^8*c^3 + 32*a^2*b^8*c - 24*a^4*b^6*c + 28*a^6*b^4*c
- 32*a^2*b^6*c^3 + 224*a^3*b^4*c^4 - 288*a^3*b^6*c^2 - 400*a^4*b^2*c^5 + 82
4*a^4*b^4*c^3 - 768*a^5*b^2*c^4 + 92*a^5*b^4*c^2 - 116*a^6*b^2*c^3 - 52*a^7
*b^2*c^2))/a^6) + (16*(6*b^9*c - 8*b^7*c^3 + 48*a*b^5*c^4 - 48*a*b^7*c^2 +
3*a^2*b^7*c + 48*a^3*b*c^6 + 26*a^4*b*c^5 - 21*a^5*b*c^4 - 80*a^2*b^3*c^5 +
122*a^2*b^5*c^3 - 108*a^3*b^3*c^4 - 21*a^3*b^5*c^2 + 42*a^4*b^3*c^3))/a^6
- (16*tan(x/2)*(2*b^10 + a^2*b^8 - 48*a^3*c^7 - 24*a^4*c^6 + 12*a^5*c^5 + 2
*a^6*c^4 + 16*b^6*c^4 - 16*b^8*c^2 - 80*a*b^4*c^5 + 112*a*b^6*c^3 - 8*a^3*b
^6*c + 96*a^2*b^2*c^6 - 232*a^2*b^4*c^4 + 48*a^2*b^6*c^2 + 152*a^3*b^2*c^5
- 24*a^3*b^4*c^3 - 36*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 - 18*a*
b^8*c))/a^6)*1i - ((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(
1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^
2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2)
+ 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c
- b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(
4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a
^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c
+ a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*(((8*a^4*c^6 - b^1
0 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*
a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2
*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c
- b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*
a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(
-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 +
16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32
*a^8*b^2*c^2)))^(1/2)*((16*(8*a^2*b^9 + 2*a^4*b^7 - a^6*b^5 - 78*a^3*b^7*c
+ 104*a^5*b*c^5 - 18*a^5*b^5*c + 114*a^6*b*c^4 - 36*a^7*b*c^3 + 6*a^7*b^3*c
- 8*a^8*b*c^2 - 8*a^2*b^7*c^2 + 64*a^3*b^5*c^3 - 152*a^4*b^3*c^4 + 256*a^4
*b^5*c^2 - 318*a^5*b^3*c^3 + 49*a^6*b^3*c^2))/a^6 - ((8*a^4*c^6 - b^10 + 8*
a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^
4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 +
b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2
)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c -
b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*
c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^1
0*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b

```
^2*c^2)))^(1/2)*((16*(4*a^7*b^5 - 16*a^5*b^7 + 3*a^9*b^3 + 122*a^6*b^5*c +
96*a^7*b*c^4 + 160*a^8*b*c^3 - 17*a^8*b^3*c + 4*a^9*b*c^2 + 16*a^5*b^5*c^2
- 88*a^6*b^3*c^3 - 272*a^7*b^3*c^2 - 12*a^10*b*c))/a^6 - ((16*(8*a^8*b^5 -
6*a^10*b^3 + 32*a^9*b*c^3 - 50*a^9*b^3*c + 72*a^10*b*c^2 - 8*a^8*b^3*c^2 +
24*a^11*b*c))/a^6 - (16*tan(x/2)*(16*a^12*c - 32*a^7*b^6 + 34*a^9*b^4 - 4*a
^11*b^2 + 384*a^9*c^4 + 768*a^10*c^3 + 400*a^11*c^2 + 288*a^8*b^4*c - 236*a
^10*b^2*c + 32*a^7*b^4*c^2 - 224*a^8*b^2*c^3 - 832*a^9*b^2*c^2))/a^6)*((8*a
^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b
^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3
- 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*
c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3
*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) +
 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 3
2*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*
b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*tan(x/2)*(8*a^11*c - 32*a^4*b^8 + 1
8*a^6*b^6 + 5*a^8*b^4 - 2*a^10*b^2 - 192*a^7*c^5 - 288*a^8*c^4 - 48*a^9*c^3
 + 56*a^10*c^2 + 288*a^5*b^6*c - 118*a^7*b^4*c - 34*a^9*b^2*c + 32*a^4*b^6*
c^2 - 224*a^5*b^4*c^3 + 432*a^6*b^2*c^4 - 864*a^6*b^4*c^2 + 968*a^7*b^2*c^3
 + 196*a^8*b^2*c^2))/a^6) + (16*tan(x/2)*(2*a^3*b^8 - 4*a^5*b^6 + 96*a^5*c^
6 + 96*a^6*c^5 + 20*a^7*c^4 + 16*a^8*c^3 + 32*a^2*b^8*c - 24*a^4*b^6*c + 28
*a^6*b^4*c - 32*a^2*b^6*c^3 + 224*a^3*b^4*c^4 - 288*a^3*b^6*c^2 - 400*a^4*b
^2*c^5 + 824*a^4*b^4*c^3 - 768*a^5*b^2*c^4 + 92*a^5*b^4*c^2 - 116*a^6*b^2*c
^3 - 52*a^7*b^2*c^2))/a^6) - (16*(6*b^9*c - 8*b^7*c^3 + 48*a*b^5*c^4 - 48*a
*b^7*c^2 + 3*a^2*b^7*c + 48*a^3*b*c^6 + 26*a^4*b*c^5 - 21*a^5*b*c^4 - 80*a^
2*b^3*c^5 + 122*a^2*b^5*c^3 - 108*a^3*b^3*c^4 - 21*a^3*b^5*c^2 + 42*a^4*b^3
*c^3))/a^6 + (16*tan(x/2)*(2*b^10 + a^2*b^8 - 48*a^3*c^7 - 24*a^4*c^6 + 12*
a^5*c^5 + 2*a^6*c^4 + 16*b^6*c^4 - 16*b^8*c^2 - 80*a*b^4*c^5 + 112*a*b^6*c^
3 - 8*a^3*b^6*c + 96*a^2*b^2*c^6 - 232*a^2*b^4*c^4 + 48*a^2*b^6*c^2 + 152*a
^3*b^2*c^5 - 24*a^3*b^4*c^3 - 36*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*
c^3 - 18*a*b^8*c))/a^6)*1i)/(((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c
- b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2
- 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2
)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^
4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*
b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(
a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c -
8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*(((8*a^
4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^
6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 -
 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c
^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*
b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) +
6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32
*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b
^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*(((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4
```

```
*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6
*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c
- b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2
*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10
*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))
/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4
*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*((
16*(4*a^7*b^5 - 16*a^5*b^7 + 3*a^9*b^3 + 122*a^6*b^5*c + 96*a^7*b*c^4 + 160
*a^8*b*c^3 - 17*a^8*b^3*c + 4*a^9*b*c^2 + 16*a^5*b^5*c^2 - 88*a^6*b^3*c^3 -
 272*a^7*b^3*c^2 - 12*a^10*b*c))/a^6 + ((16*(8*a^8*b^5 - 6*a^10*b^3 + 32*a^
9*b*c^3 - 50*a^9*b^3*c + 72*a^10*b*c^2 - 8*a^8*b^3*c^2 + 24*a^11*b*c))/a^6
- (16*tan(x/2)*(16*a^12*c - 32*a^7*b^6 + 34*a^9*b^4 - 4*a^11*b^2 + 384*a^9*
c^4 + 768*a^10*c^3 + 400*a^11*c^2 + 288*a^8*b^4*c - 236*a^10*b^2*c + 32*a^7
*b^4*c^2 - 224*a^8*b^2*c^3 - 832*a^9*b^2*c^2))/a^6)*((8*a^4*c^6 - b^10 + 8*
a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^
4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 +
 b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2
)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c -
b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*
c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^1
0*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b
^2*c^2)))^(1/2) + (16*tan(x/2)*(8*a^11*c - 32*a^4*b^8 + 18*a^6*b^6 + 5*a^8*
b^4 - 2*a^10*b^2 - 192*a^7*c^5 - 288*a^8*c^4 - 48*a^9*c^3 + 56*a^10*c^2 + 2
88*a^5*b^6*c - 118*a^7*b^4*c - 34*a^9*b^2*c + 32*a^4*b^6*c^2 - 224*a^5*b^4*
c^3 + 432*a^6*b^2*c^4 - 864*a^6*b^4*c^2 + 968*a^7*b^2*c^3 + 196*a^8*b^2*c^2
))/a^6) + (16*(8*a^2*b^9 + 2*a^4*b^7 - a^6*b^5 - 78*a^3*b^7*c + 104*a^5*b*c
^5 - 18*a^5*b^5*c + 114*a^6*b*c^4 - 36*a^7*b*c^3 + 6*a^7*b^3*c - 8*a^8*b*c^
2 - 8*a^2*b^7*c^2 + 64*a^3*b^5*c^3 - 152*a^4*b^3*c^4 + 256*a^4*b^5*c^2 - 31
8*a^5*b^3*c^3 + 49*a^6*b^3*c^2))/a^6 + (16*tan(x/2)*(2*a^3*b^8 - 4*a^5*b^6
+ 96*a^5*c^6 + 96*a^6*c^5 + 20*a^7*c^4 + 16*a^8*c^3 + 32*a^2*b^8*c - 24*a^4
*b^6*c + 28*a^6*b^4*c - 32*a^2*b^6*c^3 + 224*a^3*b^4*c^4 - 288*a^3*b^6*c^2
- 400*a^4*b^2*c^5 + 824*a^4*b^4*c^3 - 768*a^5*b^2*c^4 + 92*a^5*b^4*c^2 - 11
6*a^6*b^2*c^3 - 52*a^7*b^2*c^2))/a^6) + (16*(6*b^9*c - 8*b^7*c^3 + 48*a*b^5
*c^4 - 48*a*b^7*c^2 + 3*a^2*b^7*c + 48*a^3*b*c^6 + 26*a^4*b*c^5 - 21*a^5*b*
c^4 - 80*a^2*b^3*c^5 + 122*a^2*b^5*c^3 - 108*a^3*b^3*c^4 - 21*a^3*b^5*c^2 +
 42*a^4*b^3*c^3))/a^6 - (16*tan(x/2)*(2*b^10 + a^2*b^8 - 48*a^3*c^7 - 24*a^
4*c^6 + 12*a^5*c^5 + 2*a^6*c^4 + 16*b^6*c^4 - 16*b^8*c^2 - 80*a*b^4*c^5 + 1
12*a*b^6*c^3 - 8*a^3*b^6*c + 96*a^2*b^2*c^6 - 232*a^2*b^4*c^4 + 48*a^2*b^6*
c^2 + 152*a^3*b^2*c^5 - 24*a^3*b^4*c^3 - 36*a^4*b^2*c^4 + 20*a^4*b^4*c^2 -
16*a^5*b^2*c^3 - 18*a*b^8*c))/a^6) + ((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(
-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*
b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a
*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*
a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) -
 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/
```

```
2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*
b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)
*(((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 -
 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b
^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4
*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2)
+ 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^
(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*
c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 -
 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*((16*(8*a^2*b^9 + 2*a^4*b^7 - a^6*
b^5 - 78*a^3*b^7*c + 104*a^5*b*c^5 - 18*a^5*b^5*c + 114*a^6*b*c^4 - 36*a^7*
b*c^3 + 6*a^7*b^3*c - 8*a^8*b*c^2 - 8*a^2*b^7*c^2 + 64*a^3*b^5*c^3 - 152*a^
4*b^3*c^4 + 256*a^4*b^5*c^2 - 318*a^5*b^3*c^3 + 49*a^6*b^3*c^2))/a^6 - ((8*
a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a*
b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^3
 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^3
*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^
3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2)
+ 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 +
32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7
*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*((16*(4*a^7*b^5 - 16*a^5*b^7 + 3*a^9*b^3
 + 122*a^6*b^5*c + 96*a^7*b*c^4 + 160*a^8*b*c^3 - 17*a^8*b^3*c + 4*a^9*b*c^
2 + 16*a^5*b^5*c^2 - 88*a^6*b^3*c^3 - 272*a^7*b^3*c^2 - 12*a^10*b*c))/a^6 -
 ((16*(8*a^8*b^5 - 6*a^10*b^3 + 32*a^9*b*c^3 - 50*a^9*b^3*c + 72*a^10*b*c^2
 - 8*a^8*b^3*c^2 + 24*a^11*b*c))/a^6 - (16*tan(x/2)*(16*a^12*c - 32*a^7*b^6
 + 34*a^9*b^4 - 4*a^11*b^2 + 384*a^9*c^4 + 768*a^10*c^3 + 400*a^11*c^2 + 28
8*a^8*b^4*c - 236*a^10*b^2*c + 32*a^7*b^4*c^2 - 224*a^8*b^2*c^3 - 832*a^9*b
^2*c^2))/a^6)*((8*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2)
 + b^8*c^2 - 10*a*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^
5 + 96*a^3*b^4*c^3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12
*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^
2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*
c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b
^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a
^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*tan(x/2)*(8*a^11
*c - 32*a^4*b^8 + 18*a^6*b^6 + 5*a^8*b^4 - 2*a^10*b^2 - 192*a^7*c^5 - 288*a
^8*c^4 - 48*a^9*c^3 + 56*a^10*c^2 + 288*a^5*b^6*c - 118*a^7*b^4*c - 34*a^9*
b^2*c + 32*a^4*b^6*c^2 - 224*a^5*b^4*c^3 + 432*a^6*b^2*c^4 - 864*a^6*b^4*c^
2 + 968*a^7*b^2*c^3 + 196*a^8*b^2*c^2))/a^6) + (16*tan(x/2)*(2*a^3*b^8 - 4*
a^5*b^6 + 96*a^5*c^6 + 96*a^6*c^5 + 20*a^7*c^4 + 16*a^8*c^3 + 32*a^2*b^8*c
- 24*a^4*b^6*c + 28*a^6*b^4*c - 32*a^2*b^6*c^3 + 224*a^3*b^4*c^4 - 288*a^3*
b^6*c^2 - 400*a^4*b^2*c^5 + 824*a^4*b^4*c^3 - 768*a^5*b^2*c^4 + 92*a^5*b^4*
c^2 - 116*a^6*b^2*c^3 - 52*a^7*b^2*c^2))/a^6) - (16*(6*b^9*c - 8*b^7*c^3 +
48*a*b^5*c^4 - 48*a*b^7*c^2 + 3*a^2*b^7*c + 48*a^3*b*c^6 + 26*a^4*b*c^5 - 2
1*a^5*b*c^4 - 80*a^2*b^3*c^5 + 122*a^2*b^5*c^3 - 108*a^3*b^3*c^4 - 21*a^3*b
```

```
^5*c^2 + 42*a^4*b^3*c^3))/a^6 + (16*tan(x/2)*(2*b^10 + a^2*b^8 - 48*a^3*c^7
 - 24*a^4*c^6 + 12*a^5*c^5 + 2*a^6*c^4 + 16*b^6*c^4 - 16*b^8*c^2 - 80*a*b^4
*c^5 + 112*a*b^6*c^3 - 8*a^3*b^6*c + 96*a^2*b^2*c^6 - 232*a^2*b^4*c^4 + 48*
a^2*b^6*c^2 + 152*a^3*b^2*c^5 - 24*a^3*b^4*c^3 - 36*a^4*b^2*c^4 + 20*a^4*b^
4*c^2 - 16*a^5*b^2*c^3 - 18*a*b^8*c))/a^6) - (32*(8*b^3*c^6 - 2*b^5*c^4 + 6
*a*b^3*c^5 + 2*a^3*b*c^5 - a^2*b^3*c^4 - 8*a*b*c^7))/a^6 - (32*tan(x/2)*(4*
a^3*c^6 + 16*b^2*c^7 - 8*b^4*c^5 + 16*a*b^2*c^6 - 4*a^2*b^2*c^5))/a^6))*((8
*a^4*c^6 - b^10 + 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) + b^8*c^2 - 10*a
*b^6*c^3 + 33*a^2*b^4*c^4 - 52*a^2*b^6*c^2 - 38*a^3*b^2*c^5 + 96*a^3*b^4*c^
3 - 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) + 12*a*b^8*c - 4*a*b^
3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a
^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2)
 + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 +
 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^
7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*2i - atan(-(((((16*(4*a^7*b^5 - 16*a^5*
b^7 + 3*a^9*b^3 + 122*a^6*b^5*c + 96*a^7*b*c^4 + 160*a^8*b*c^3 - 17*a^8*b^3
*c + 4*a^9*b*c^2 + 16*a^5*b^5*c^2 - 88*a^6*b^3*c^3 - 272*a^7*b^3*c^2 - 12*a
^10*b*c))/a^6 + ((16*(8*a^8*b^5 - 6*a^10*b^3 + 32*a^9*b*c^3 - 50*a^9*b^3*c
+ 72*a^10*b*c^2 - 8*a^8*b^3*c^2 + 24*a^11*b*c))/a^6 - (16*tan(x/2)*(16*a^12
*c - 32*a^7*b^6 + 34*a^9*b^4 - 4*a^11*b^2 + 384*a^9*c^4 + 768*a^10*c^3 + 40
0*a^11*c^2 + 288*a^8*b^4*c - 236*a^10*b^2*c + 32*a^7*b^4*c^2 - 224*a^8*b^2*
c^3 - 832*a^9*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c
 - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2
 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^
2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c
^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2
*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*
(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c -
 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*
tan(x/2)*(8*a^11*c - 32*a^4*b^8 + 18*a^6*b^6 + 5*a^8*b^4 - 2*a^10*b^2 - 192
*a^7*c^5 - 288*a^8*c^4 - 48*a^9*c^3 + 56*a^10*c^2 + 288*a^5*b^6*c - 118*a^7
*b^4*c - 34*a^9*b^2*c + 32*a^4*b^6*c^2 - 224*a^5*b^4*c^3 + 432*a^6*b^2*c^4
- 864*a^6*b^4*c^2 + 968*a^7*b^2*c^3 + 196*a^8*b^2*c^2))/a^6)*(-(b^10 - 8*a^
4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 -
 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4
*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4
*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(
-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5
*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^
3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3
- 32*a^8*b^2*c^2)))^(1/2) + (16*(8*a^2*b^9 + 2*a^4*b^7 - a^6*b^5 - 78*a^3*b
^7*c + 104*a^5*b*c^5 - 18*a^5*b^5*c + 114*a^6*b*c^4 - 36*a^7*b*c^3 + 6*a^7*
b^3*c - 8*a^8*b*c^2 - 8*a^2*b^7*c^2 + 64*a^3*b^5*c^3 - 152*a^4*b^3*c^4 + 25
6*a^4*b^5*c^2 - 318*a^5*b^3*c^3 + 49*a^6*b^3*c^2))/a^6 + (16*tan(x/2)*(2*a^
3*b^8 - 4*a^5*b^6 + 96*a^5*c^6 + 96*a^6*c^5 + 20*a^7*c^4 + 16*a^8*c^3 + 32*
```

a^2*b^8*c - 24*a^4*b^6*c + 28*a^6*b^4*c - 32*a^2*b^6*c^3 + 224*a^3*b^4*c^4
- 288*a^3*b^6*c^2 - 400*a^4*b^2*c^5 + 824*a^4*b^4*c^3 - 768*a^5*b^2*c^4 + 9
2*a^5*b^4*c^2 - 116*a^6*b^2*c^3 - 52*a^7*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6
 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a
^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*
c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c
- b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a
*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-
(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 1
6*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*
a^8*b^2*c^2)))^(1/2) + (16*(6*b^9*c - 8*b^7*c^3 + 48*a*b^5*c^4 - 48*a*b^7*c
^2 + 3*a^2*b^7*c + 48*a^3*b*c^6 + 26*a^4*b*c^5 - 21*a^5*b*c^4 - 80*a^2*b^3*
c^5 + 122*a^2*b^5*c^3 - 108*a^3*b^3*c^4 - 21*a^3*b^5*c^2 + 42*a^4*b^3*c^3))
/a^6 - (16*tan(x/2)*(2*b^10 + a^2*b^8 - 48*a^3*c^7 - 24*a^4*c^6 + 12*a^5*c^
5 + 2*a^6*c^4 + 16*b^6*c^4 - 16*b^8*c^2 - 80*a*b^4*c^5 + 112*a*b^6*c^3 - 8*
a^3*b^6*c + 96*a^2*b^2*c^6 - 232*a^2*b^4*c^4 + 48*a^2*b^6*c^2 + 152*a^3*b^2
*c^5 - 24*a^3*b^4*c^3 - 36*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 -
18*a*b^8*c))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^
(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b
^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2)
 - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c
 - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-
(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 -
a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*
c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*1i - (((16*(8*a^2
*b^9 + 2*a^4*b^7 - a^6*b^5 - 78*a^3*b^7*c + 104*a^5*b*c^5 - 18*a^5*b^5*c +
114*a^6*b*c^4 - 36*a^7*b*c^3 + 6*a^7*b^3*c - 8*a^8*b*c^2 - 8*a^2*b^7*c^2 +
64*a^3*b^5*c^3 - 152*a^4*b^3*c^4 + 256*a^4*b^5*c^2 - 318*a^5*b^3*c^3 + 49*a
^6*b^3*c^2))/a^6 - ((16*(4*a^7*b^5 - 16*a^5*b^7 + 3*a^9*b^3 + 122*a^6*b^5*c
 + 96*a^7*b*c^4 + 160*a^8*b*c^3 - 17*a^8*b^3*c + 4*a^9*b*c^2 + 16*a^5*b^5*c
^2 - 88*a^6*b^3*c^3 - 272*a^7*b^3*c^2 - 12*a^10*b*c))/a^6 - ((16*(8*a^8*b^5
 - 6*a^10*b^3 + 32*a^9*b*c^3 - 50*a^9*b^3*c + 72*a^10*b*c^2 - 8*a^8*b^3*c^2
 + 24*a^11*b*c))/a^6 - (16*tan(x/2)*(16*a^12*c - 32*a^7*b^6 + 34*a^9*b^4 -
4*a^11*b^2 + 384*a^9*c^4 + 768*a^10*c^3 + 400*a^11*c^2 + 288*a^8*b^4*c - 23
6*a^10*b^2*c + 32*a^7*b^4*c^2 - 224*a^8*b^2*c^3 - 832*a^9*b^2*c^2))/a^6)*(-
(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10
*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*
c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*
b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4
*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/
2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4
 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*
a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*tan(x/2)*(8*a^11*c - 32*a^4*b^8
 + 18*a^6*b^6 + 5*a^8*b^4 - 2*a^10*b^2 - 192*a^7*c^5 - 288*a^8*c^4 - 48*a^9
*c^3 + 56*a^10*c^2 + 288*a^5*b^6*c - 118*a^7*b^4*c - 34*a^9*b^2*c + 32*a^4*

b^6*c^2 - 224*a^5*b^4*c^3 + 432*a^6*b^2*c^4 - 864*a^6*b^4*c^2 + 968*a^7*b^2
*c^3 + 196*a^8*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*
c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^
2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b
^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*
c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^
2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2
*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c
- 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16
tan(x/2)(2*a^3*b^8 - 4*a^5*b^6 + 96*a^5*c^6 + 96*a^6*c^5 + 20*a^7*c^4 + 1
6*a^8*c^3 + 32*a^2*b^8*c - 24*a^4*b^6*c + 28*a^6*b^4*c - 32*a^2*b^6*c^3 + 2
24*a^3*b^4*c^4 - 288*a^3*b^6*c^2 - 400*a^4*b^2*c^5 + 824*a^4*b^4*c^3 - 768*
a^5*b^2*c^4 + 92*a^5*b^4*c^2 - 116*a^6*b^2*c^3 - 52*a^7*b^2*c^2))/a^6)*(-(b
^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a
*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^
3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^
3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a
^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2)
 + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 +
 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^
7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) - (16*(6*b^9*c - 8*b^7*c^3 + 48*a*b^5*c
^4 - 48*a*b^7*c^2 + 3*a^2*b^7*c + 48*a^3*b*c^6 + 26*a^4*b*c^5 - 21*a^5*b*c^
4 - 80*a^2*b^3*c^5 + 122*a^2*b^5*c^3 - 108*a^3*b^3*c^4 - 21*a^3*b^5*c^2 + 4
2*a^4*b^3*c^3))/a^6 + (16*tan(x/2)*(2*b^10 + a^2*b^8 - 48*a^3*c^7 - 24*a^4*
c^6 + 12*a^5*c^5 + 2*a^6*c^4 + 16*b^6*c^4 - 16*b^8*c^2 - 80*a*b^4*c^5 + 112
*a*b^6*c^3 - 8*a^3*b^6*c + 96*a^2*b^2*c^6 - 232*a^2*b^4*c^4 + 48*a^2*b^6*c^
2 + 152*a^3*b^2*c^5 - 24*a^3*b^4*c^3 - 36*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16
*a^5*b^2*c^3 - 18*a*b^8*c))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4
*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6
*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c
- b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2
*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10
*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))
/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4
*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2)*1i
)/((32*(8*b^3*c^6 - 2*b^5*c^4 + 6*a*b^3*c^5 + 2*a^3*b*c^5 - a^2*b^3*c^4 - 8
*a*b*c^7))/a^6 - ((((16*(4*a^7*b^5 - 16*a^5*b^7 + 3*a^9*b^3 + 122*a^6*b^5*c
 + 96*a^7*b*c^4 + 160*a^8*b*c^3 - 17*a^8*b^3*c + 4*a^9*b*c^2 + 16*a^5*b^5*c
^2 - 88*a^6*b^3*c^3 - 272*a^7*b^3*c^2 - 12*a^10*b*c))/a^6 + ((16*(8*a^8*b^5
 - 6*a^10*b^3 + 32*a^9*b*c^3 - 50*a^9*b^3*c + 72*a^10*b*c^2 - 8*a^8*b^3*c^2
 + 24*a^11*b*c))/a^6 - (16*tan(x/2)*(16*a^12*c - 32*a^7*b^6 + 34*a^9*b^4 -
4*a^11*b^2 + 384*a^9*c^4 + 768*a^10*c^3 + 400*a^11*c^2 + 288*a^8*b^4*c - 23
6*a^10*b^2*c + 32*a^7*b^4*c^2 - 224*a^8*b^2*c^3 - 832*a^9*b^2*c^2))/a^6)*(-
(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10
*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*

c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*
b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4
*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/
2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4
 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*
a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*tan(x/2)*(8*a^11*c - 32*a^4*b^8
 + 18*a^6*b^6 + 5*a^8*b^4 - 2*a^10*b^2 - 192*a^7*c^5 - 288*a^8*c^4 - 48*a^9
*c^3 + 56*a^10*c^2 + 288*a^5*b^6*c - 118*a^7*b^4*c - 34*a^9*b^2*c + 32*a^4*
b^6*c^2 - 224*a^5*b^4*c^3 + 432*a^6*b^2*c^4 - 864*a^6*b^4*c^2 + 968*a^7*b^2
*c^3 + 196*a^8*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*
c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^
2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b
^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*
c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^
2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2
*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c
 - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16
*(8*a^2*b^9 + 2*a^4*b^7 - a^6*b^5 - 78*a^3*b^7*c + 104*a^5*b*c^5 - 18*a^5*b
^5*c + 114*a^6*b*c^4 - 36*a^7*b*c^3 + 6*a^7*b^3*c - 8*a^8*b*c^2 - 8*a^2*b^7
*c^2 + 64*a^3*b^5*c^3 - 152*a^4*b^3*c^4 + 256*a^4*b^5*c^2 - 318*a^5*b^3*c^3
 + 49*a^6*b^3*c^2))/a^6 + (16*tan(x/2)*(2*a^3*b^8 - 4*a^5*b^6 + 96*a^5*c^6
 + 96*a^6*c^5 + 20*a^7*c^4 + 16*a^8*c^3 + 32*a^2*b^8*c - 24*a^4*b^6*c + 28*a
^6*b^4*c - 32*a^2*b^6*c^3 + 224*a^3*b^4*c^4 - 288*a^3*b^6*c^2 - 400*a^4*b^2
*c^5 + 824*a^4*b^4*c^3 - 768*a^5*b^2*c^4 + 92*a^5*b^4*c^2 - 116*a^6*b^2*c^3
 - 52*a^7*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b
^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 3
8*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3
)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(
-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3
c^2(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8
*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a
^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*(6*b
^9*c - 8*b^7*c^3 + 48*a*b^5*c^4 - 48*a*b^7*c^2 + 3*a^2*b^7*c + 48*a^3*b*c^6
 + 26*a^4*b*c^5 - 21*a^5*b*c^4 - 80*a^2*b^3*c^5 + 122*a^2*b^5*c^3 - 108*a^3
*b^3*c^4 - 21*a^3*b^5*c^2 + 42*a^4*b^3*c^3))/a^6 - (16*tan(x/2)*(2*b^10 + a
^2*b^8 - 48*a^3*c^7 - 24*a^4*c^6 + 12*a^5*c^5 + 2*a^6*c^4 + 16*b^6*c^4 - 16
*b^8*c^2 - 80*a*b^4*c^5 + 112*a*b^6*c^3 - 8*a^3*b^6*c + 96*a^2*b^2*c^6 - 23
2*a^2*b^4*c^4 + 48*a^2*b^6*c^2 + 152*a^3*b^2*c^5 - 24*a^3*b^4*c^3 - 36*a^4*
b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 - 18*a*b^8*c))/a^6)*(-(b^10 - 8*a
^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3
 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^
4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(
4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*
(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^
5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c

^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) - (((16*(8*a^2*b^9 + 2*a^4*b^7 - a^6*b^5 - 78*a^3*b^7*c + 104*a^5*b*c^5 - 18*a^5*b^5*c + 114*a^6*b*c^4 - 36*a^7*b*c^3 + 6*a^7*b^3*c - 8*a^8*b*c^2 - 8*a^2*b^7*c^2 + 64*a^3*b^5*c^3 - 152*a^4*b^3*c^4 + 256*a^4*b^5*c^2 - 318*a^5*b^3*c^3 + 49*a^6*b^3*c^2))/a^6 - ((16*(4*a^7*b^5 - 16*a^5*b^7 + 3*a^9*b^3 + 122*a^6*b^5*c + 96*a^7*b*c^4 + 160*a^8*b*c^3 - 17*a^8*b^3*c + 4*a^9*b*c^2 + 16*a^5*b^5*c^2 - 88*a^6*b^3*c^3 - 272*a^7*b^3*c^2 - 12*a^10*b*c))/a^6 - ((16*(8*a^8*b^5 - 6*a^10*b^3 + 32*a^9*b*c^3 - 50*a^9*b^3*c + 72*a^10*b*c^2 - 8*a^8*b^3*c^2 + 24*a^11*b*c))/a^6 - (16*tan(x/2)*(16*a^12*c - 32*a^7*b^6 + 34*a^9*b^4 - 4*a^11*b^2 + 384*a^9*c^4 + 768*a^10*c^3 + 400*a^11*c^2 + 288*a^8*b^4*c - 236*a^10*b^2*c + 32*a^7*b^4*c^2 - 224*a^8*b^2*c^3 - 832*a^9*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*tan(x/2)*(8*a^11*c - 32*a^4*b^8 + 18*a^6*b^6 + 5*a^8*b^4 - 2*a^10*b^2 - 192*a^7*c^5 - 288*a^8*c^4 - 48*a^9*c^3 + 56*a^10*c^2 + 288*a^5*b^6*c - 118*a^7*b^4*c - 34*a^9*b^2*c + 32*a^4*b^6*c^2 - 224*a^5*b^4*c^3 + 432*a^6*b^2*c^4 - 864*a^6*b^4*c^2 + 968*a^7*b^2*c^3 + 196*a^8*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (16*tan(x/2)*(2*a^3*b^8 - 4*a^5*b^6 + 96*a^5*c^6 + 96*a^6*c^5 + 20*a^7*c^4 + 16*a^8*c^3 + 32*a^2*b^8*c - 24*a^4*b^6*c + 28*a^6*b^4*c - 32*a^2*b^6*c^3 + 224*a^3*b^4*c^4 - 288*a^3*b^6*c^2 - 400*a^4*b^2*c^5 + 824*a^4*b^4*c^3 - 768*a^5*b^2*c^4 + 92*a^5*b^4*c^2 - 116*a^6*b^2*c^3 - 52*a^7*b^2*c^2))/a^6)*(-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) - (16*(6*b^9*c - 8*b^7*c^3 + 48*a*b^5*c^4 - 48*a*b^7*c^2 + 3*a^2*b^7*c + 48*a^3*b*c^6 + 26*a^4*b*c^5 - 21*a^5*b*c^4 - 80*a^2*b^3*c^5 + 122*a^2*b^5*c^3 - 108*a^3*b^3*c^4 - 21*a^3*b^5*c^2 + 42*a^4*b^3*c^3))/a^6 + (16*tan(x/2)

```
*(2*b^10 + a^2*b^8 - 48*a^3*c^7 - 24*a^4*c^6 + 12*a^5*c^5 + 2*a^6*c^4 + 16*
b^6*c^4 - 16*b^8*c^2 - 80*a*b^4*c^5 + 112*a*b^6*c^3 - 8*a^3*b^6*c + 96*a^2*
b^2*c^6 - 232*a^2*b^4*c^4 + 48*a^2*b^6*c^2 + 152*a^3*b^2*c^5 - 24*a^3*b^4*c
^3 - 36*a^4*b^2*c^4 + 20*a^4*b^4*c^2 - 16*a^5*b^2*c^3 - 18*a*b^8*c))/a^6)*(
-(b^10 - 8*a^4*c^6 - 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 1
0*a*b^6*c^3 - 33*a^2*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4
*c^3 + 66*a^4*b^2*c^4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a
*b^3*c^3*(-(4*a*c - b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) +
4*a^3*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1
/2) + 6*a*b^5*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^
4 + 32*a^9*c^3 + 16*a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8
*a^7*b^2*c^3 - 32*a^8*b^2*c^2)))^(1/2) + (32*tan(x/2)*(4*a^3*c^6 + 16*b^2*c
^7 - 8*b^4*c^5 + 16*a*b^2*c^6 - 4*a^2*b^2*c^5))/a^6))*(-(b^10 - 8*a^4*c^6 -
 8*a^5*c^5 - b^7*(-(4*a*c - b^2)^3)^(1/2) - b^8*c^2 + 10*a*b^6*c^3 - 33*a^2
*b^4*c^4 + 52*a^2*b^6*c^2 + 38*a^3*b^2*c^5 - 96*a^3*b^4*c^3 + 66*a^4*b^2*c^
4 + b^5*c^2*(-(4*a*c - b^2)^3)^(1/2) - 12*a*b^8*c - 4*a*b^3*c^3*(-(4*a*c -
b^2)^3)^(1/2) + 3*a^2*b*c^4*(-(4*a*c - b^2)^3)^(1/2) + 4*a^3*b*c^3*(-(4*a*c
 - b^2)^3)^(1/2) - 10*a^2*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b^5*c*(-(4
*a*c - b^2)^3)^(1/2))/(2*(a^8*b^4 - a^6*b^6 + 16*a^8*c^4 + 32*a^9*c^3 + 16*
a^10*c^2 + 10*a^7*b^4*c - 8*a^9*b^2*c + a^6*b^4*c^2 - 8*a^7*b^2*c^3 - 32*a^
8*b^2*c^2)))^(1/2)*2i + tan(x/2)^2/(8*a) + (log(tan(x/2))*(a^2 - 2*a*c + 2*
b^2))/(2*a^3) - (b*tan(x/2))/(2*a^2) - (a/2 - 2*b*tan(x/2))/(4*a^2*tan(x/2)
^2)
```

3.9 $\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 76

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{(b^2-2c(a+c))\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{c^2\sqrt{b^2-4ac}} + \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2c^2} - \frac{\sin(x)}{c}$$

```
[Out] 1/2*b*ln(a+b*sin(x)+c*sin(x)^2)/c^2-sin(x)/c+(b^2-2*c*(a+c))*arctanh((b+2*c
*sin(x))/(-4*a*c+b^2)^(1/2))/c^2/(-4*a*c+b^2)^(1/2)
```

Rubi [A] (verified)

Time = 0.16 (sec) , antiderivative size = 76, normalized size of antiderivative = 1.00, number of steps used = 7, number of rules used = 6, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.316, Rules used = {3339, 1671, 648, 632, 212, 642}

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{(b^2-2c(a+c))\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{c^2\sqrt{b^2-4ac}} + \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2c^2} - \frac{\sin(x)}{c}$$

```
[In] Int[Cos[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] ((b^2 - 2*c*(a + c))*ArcTanh[(b + 2*c*Sin[x])/Sqrt[b^2 - 4*a*c]])/(c^2*Sqrt
[b^2 - 4*a*c]) + (b*Log[a + b*Sin[x] + c*Sin[x]^2])/(2*c^2) - Sin[x]/c
```

Rule 212

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(1/(Rt[a, 2]*Rt[-b, 2]))*
ArcTanh[Rt[-b, 2]*(x/Rt[a, 2])], x] /; FreeQ[{a, b}, x] && NegQ[a/b] && (Gt
Q[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 642

```
Int[((d_) + (e_.)*(x_))/((a_.) + (b_.)*(x_) + (c_.)*(x_)^2), x_Symbol] :> S
imp[d*(Log[RemoveContent[a + b*x + c*x^2, x]]/b), x] /; FreeQ[{a, b, c, d,
e}, x] && EqQ[2*c*d - b*e, 0]
```

Rule 648

```
Int[((d_.) + (e_.)*(x_))/((a_) + (b_.)*(x_) + (c_.)*(x_)^2), x_Symbol] :> D
ist[(2*c*d - b*e)/(2*c), Int[1/(a + b*x + c*x^2), x], x] + Dist[e/(2*c), In
t[(b + 2*c*x)/(a + b*x + c*x^2), x], x] /; FreeQ[{a, b, c, d, e}, x] && NeQ
[2*c*d - b*e, 0] && NeQ[b^2 - 4*a*c, 0] &&  !NiceSqrtQ[b^2 - 4*a*c]
```

Rule 1671

```
Int[(Pq_)*((a_) + (b_.)*(x_) + (c_.)*(x_)^2)^(p_.), x_Symbol] :> Int[Expand
Integrand[Pq*(a + b*x + c*x^2)^p, x], x] /; FreeQ[{a, b, c}, x] && PolyQ[Pq
, x] && IGtQ[p, -2]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}\text{integral} &= \text{Subst}\left(\int \frac{1-x^2}{a+bx+cx^2}\,dx, x, \sin(x)\right)\\ &= \text{Subst}\left(\int \left(-\frac{1}{c}+\frac{a+c+bx}{c\,(a+bx+cx^2)}\right)\,dx, x, \sin(x)\right)\end{aligned}$$

$$= -\frac{\sin(x)}{c} + \frac{\text{Subst}\left(\int \frac{a+c+bx}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{c}$$

$$= -\frac{\sin(x)}{c} + \frac{b\text{Subst}\left(\int \frac{b+2cx}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{2c^2} - \frac{(b^2-2c(a+c))\,\text{Subst}\left(\int \frac{1}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{2c^2}$$

$$= \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2c^2} - \frac{\sin(x)}{c} + \frac{(b^2-2c(a+c))\,\text{Subst}\left(\int \frac{1}{b^2-4ac-x^2}\,dx, x, b+2c\sin(x)\right)}{c^2}$$

$$= \frac{(b^2-2c(a+c))\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{c^2\sqrt{b^2-4ac}} + \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2c^2} - \frac{\sin(x)}{c}$$

Mathematica [A] (verified)

Time = 0.13 (sec) , antiderivative size = 73, normalized size of antiderivative = 0.96

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\frac{2(b^2-2c(a+c))\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}} + b\log\left(a+b\sin(x)+c\sin^2(x)\right) - 2c\sin(x)}{2c^2}$$

```
[In] Integrate[Cos[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
[Out] ((2*(b^2 - 2*c*(a + c))*ArcTanh[(b + 2*c*Sin[x])/Sqrt[b^2 - 4*a*c]])/Sqrt[b
^2 - 4*a*c] + b*Log[a + b*Sin[x] + c*Sin[x]^2] - 2*c*Sin[x])/(2*c^2)
```

Maple [A] (verified)

Time = 0.92 (sec) , antiderivative size = 79, normalized size of antiderivative = 1.04

method	result	size
derivativedivides	$-\frac{\sin(x)}{c}+\frac{\frac{b\ln\left(a+b\sin(x)+c\left(\sin^2(x)\right)\right)}{2c}+\frac{2\left(a+c-\frac{b^2}{2c}\right)\arctan\left(\frac{b+2\sin(x)c}{\sqrt{4ac-b^2}}\right)}{\sqrt{4ac-b^2}}}{c}$	79
default	$-\frac{\sin(x)}{c}+\frac{\frac{b\ln\left(a+b\sin(x)+c\left(\sin^2(x)\right)\right)}{2c}+\frac{2\left(a+c-\frac{b^2}{2c}\right)\arctan\left(\frac{b+2\sin(x)c}{\sqrt{4ac-b^2}}\right)}{\sqrt{4ac-b^2}}}{c}$	79
risch	Expression too large to display	1072

```
[In] int(cos(x)^3/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
[Out] -sin(x)/c+1/c*(1/2*b/c*ln(a+b*sin(x)+c*sin(x)^2)+2*(a+c-1/2*b^2/c)/(4*a*c-b
^2)^(1/2)*arctan((b+2*sin(x)*c)/(4*a*c-b^2)^(1/2)))
```

Fricas [A] (verification not implemented)

none

Time = 0.38 (sec) , antiderivative size = 276, normalized size of antiderivative = 3.63

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$
$$= \left[-\frac{(b^2-2\,ac-2\,c^2)\sqrt{b^2-4\,ac}\log\left(-\frac{2\,c^2\cos(x)^2-2\,bc\sin(x)-b^2+2\,ac-2\,c^2+\sqrt{b^2-4\,ac}(2\,c\sin(x)+b)}{c\cos(x)^2-b\sin(x)-a-c}\right)-(b^3-4\,abc)\log}{2\,(b^2c^2-4\,ac^3)}\right.$$

```
[In] integrate(cos(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] [-1/2*((b^2 - 2*a*c - 2*c^2)*sqrt(b^2 - 4*a*c)*log(-(2*c^2*cos(x)^2 - 2*b*c
*sin(x) - b^2 + 2*a*c - 2*c^2 + sqrt(b^2 - 4*a*c)*(2*c*sin(x) + b))/(c*cos(
x)^2 - b*sin(x) - a - c)) - (b^3 - 4*a*b*c)*log(-c*cos(x)^2 + b*sin(x) + a
+ c) + 2*(b^2*c - 4*a*c^2)*sin(x))/(b^2*c^2 - 4*a*c^3), 1/2*(2*(b^2 - 2*a*c
 - 2*c^2)*sqrt(-b^2 + 4*a*c)*arctan(-sqrt(-b^2 + 4*a*c)*(2*c*sin(x) + b)/(b
^2 - 4*a*c)) + (b^3 - 4*a*b*c)*log(-c*cos(x)^2 + b*sin(x) + a + c) - 2*(b^2
*c - 4*a*c^2)*sin(x))/(b^2*c^2 - 4*a*c^3)]
```

Sympy [F(-1)]

Timed out.

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(cos(x)**3/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Timed out
```

Maxima [F(-2)]

Exception generated.

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Exception raised: ValueError}$$

```
[In] integrate(cos(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] Exception raised: ValueError >> Computation failed since Maxima requested a
dditional constraints; using the 'assume' command before evaluation *may* h
elp (example of legal syntax is 'assume(4*a*c-b^2>0)', see `assume?` for mo
re deta
```

Giac [A] (verification not implemented)

none

Time = 0.32 (sec) , antiderivative size = 78, normalized size of antiderivative = 1.03

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{b\log\left(c\sin\left(x\right)^2+b\sin\left(x\right)+a\right)}{2\,c^2} - \frac{\sin\left(x\right)}{c} - \frac{\left(b^2-2\,ac-2\,c^2\right)\arctan\left(\frac{2\,c\sin(x)+b}{\sqrt{-b^2+4\,ac}}\right)}{\sqrt{-b^2+4\,ac}c^2}$$

```
[In] integrate(cos(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] 1/2*b*log(c*sin(x)^2 + b*sin(x) + a)/c^2 - sin(x)/c - (b^2 - 2*a*c - 2*c^2)
*arctan((2*c*sin(x) + b)/sqrt(-b^2 + 4*a*c))/(sqrt(-b^2 + 4*a*c)*c^2)
```

Mupad [B] (verification not implemented)

Time = 0.23 (sec) , antiderivative size = 229, normalized size of antiderivative = 3.01

$$\int \frac{\cos^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{2\,\mathrm{atan}\left(\frac{b}{\sqrt{4\,a\,c-b^2}}+\frac{2\,c\,\sin(x)}{\sqrt{4\,a\,c-b^2}}\right)}{\sqrt{4\,a\,c-b^2}} - \frac{\sin\left(x\right)}{c} - \frac{b^3\,\ln\left(c\sin\left(x\right)^2+b\,\sin\left(x\right)+a\right)}{2\,\left(4\,a\,c^3-b^2\,c^2\right)} - \frac{b^2\,\mathrm{atan}\left(\frac{b}{\sqrt{4\,a\,c-b^2}}+\frac{2\,c\,\sin(x)}{\sqrt{4\,a\,c-b^2}}\right)}{c^2\,\sqrt{4\,a\,c-b^2}} + \frac{2\,a\,\mathrm{atan}\left(\frac{b}{\sqrt{4\,a\,c-b^2}}+\frac{2\,c\,\sin(x)}{\sqrt{4\,a\,c-b^2}}\right)}{c\,\sqrt{4\,a\,c-b^2}} + \frac{2\,a\,b\,c\,\ln\left(c\sin\left(x\right)^2+b\,\sin\left(x\right)+a\right)}{4\,a\,c^3-b^2\,c^2}$$

```
[In] int(cos(x)^3/(a + c*sin(x)^2 + b*sin(x)),x)
```

```
[Out] (2*atan(b/(4*a*c - b^2)^(1/2) + (2*c*sin(x))/(4*a*c - b^2)^(1/2)))/(4*a*c -
 b^2)^(1/2) - sin(x)/c - (b^3*log(a + c*sin(x)^2 + b*sin(x)))/(2*(4*a*c^3 -
 b^2*c^2)) - (b^2*atan(b/(4*a*c - b^2)^(1/2) + (2*c*sin(x))/(4*a*c - b^2)^(
1/2)))/(c^2*(4*a*c - b^2)^(1/2)) + (2*a*atan(b/(4*a*c - b^2)^(1/2) + (2*c*s
in(x))/(4*a*c - b^2)^(1/2)))/(c*(4*a*c - b^2)^(1/2)) + (2*a*b*c*log(a + c*s
in(x)^2 + b*sin(x)))/(4*a*c^3 - b^2*c^2)
```

3.10 $\int \frac{\cos^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 230

$$\int \frac{\cos^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{x}{c}-\frac{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-4ac}}+\frac{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-4ac}}$$

```
[Out] -x/c-arctan(1/2*(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a
+c)-b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2
))^(1/2)/c/(-4*a*c+b^2)^(1/2)+arctan(1/2*(2*c+(b+(-4*a*c+b^2)^(1/2))*tan(1/
2*x))*2^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2))*2^(1/2)*(b^2-2*c*
(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2)/c/(-4*a*c+b^2)^(1/2)
```

Rubi [A] (verified)

Time = 0.63 (sec) , antiderivative size = 230, normalized size of antiderivative = 1.00, number of steps used = 9, number of rules used = 5, $\frac{\text{number of rules}}{\text{integrand size}} = 0.263$, Rules used

$= \{3347, 3373, 2739, 632, 210\}$

$$\int \frac{\cos^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$
$$= -\frac{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c\sqrt{b^2-4ac}}$$
$$+\frac{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{c\sqrt{b^2-4ac}}-\frac{x}{c}$$

```
[In] Int[Cos[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -(x/c) - (Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt[b^2 - 4*a*c]]*ArcTan[(2*c
 + (b - Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sq
rt[b^2 - 4*a*c]])])/(c*Sqrt[b^2 - 4*a*c]) + (Sqrt[2]*Sqrt[b^2 - 2*c*(a + c)
 + b*Sqrt[b^2 - 4*a*c]]*ArcTan[(2*c + (b + Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sq
rt[2]*Sqrt[b^2 - 2*c*(a + c) + b*Sqrt[b^2 - 4*a*c]])])/(c*Sqrt[b^2 - 4*a*c]
)
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3347

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_.), x_Symbol] :> Int[ExpandT
rig[(1 - sin[d + e*x]^2)^(m/2)*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n)
)^p, x], x] /; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && IntegerQ[m/2] &
& NeQ[b^2 - 4*a*c, 0] && IntegersQ[n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rubi steps

$$\begin{aligned}
\text{integral} &= \int \left(-\frac{1}{c}+\frac{a\left(1+\frac{c}{a}\right)+b\sin(x)}{c\left(a+b\sin(x)+c\sin^2(x)\right)}\right)dx\\
&= -\frac{x}{c}+\frac{\int \frac{a\left(1+\frac{c}{a}\right)+b\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{c}\\
&= -\frac{x}{c}+\frac{\left(b-\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c}+\frac{\left(b+\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{c}\\
&= -\frac{x}{c}+\frac{\left(2\left(b-\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)}{c}\\
&\quad+\frac{\left(2\left(b+\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx,x,\tan\left(\frac{x}{2}\right)\right)}{c}\\
&= -\frac{x}{c}\\
&\quad-\frac{\left(4\left(b-\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx,x,4c+2\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{c}\\
&\quad-\frac{\left(4\left(b+\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx,x,4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{c}\\
&= -\frac{x}{c}-\frac{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-4ac}}\\
&\quad+\frac{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{c\sqrt{b^2-4ac}}
\end{aligned}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 0.58 (sec) , antiderivative size = 314, normalized size of antiderivative = 1.37

$$\int \frac{\cos^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{-x+\frac{\left(ib^2-2ic(a+c)+b\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}+\frac{\left(-ib^2+2ic(a+c)+b\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b+i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}}{c}$$

```
[In] Integrate[Cos[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (-x + ((I*b^2 - (2*I)*c*(a + c) + b*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b -
I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt[
-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt
[-b^2 + 4*a*c]]) + (((-I)*b^2 + (2*I)*c*(a + c) + b*Sqrt[-b^2 + 4*a*c])*Arc
Tan[(2*c + (b + I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a
+ c) + I*b*Sqrt[-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*Sqrt[b^2 - 2*c*(a
 + c) + I*b*Sqrt[-b^2 + 4*a*c]]))/c
```

Maple [C] (verified)

Result contains higher order function than in optimal. Order 9 vs. order 3.

Time = 0.91 (sec) , antiderivative size = 178, normalized size of antiderivative = 0.77

method	result
risch	$-\frac{x}{c}+\left(\sum_{\_R=\operatorname{RootOf}\left((16a^2c^4-8a\,b^2c^3+b^4c^2)\_Z^4+(8a^2c^2-6a\,b^2c+8a\,c^3+b^4-2b^2c^2)\_Z^2+a^2+2ac-b^2+c^2\right)}\_R\ln\left(\mathrm{e}^{ix}+\left(\right.\right.\right.$
default	$\frac{2a\left(-\frac{\left(-\sqrt{-4ac+b^2}\,ba+\sqrt{-4ac+b^2}\,bc+4a^2c-a\,b^2+4a\,c^2-b^2c\right)\arctan\left(\frac{-2a\tan\left(\frac{x}{2}\right)+\sqrt{-4ac+b^2}-b}{\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}\right)}{a\left(4ac-b^2\right)\sqrt{4ac-2b^2+2b\sqrt{-4ac+b^2}+4a^2}}+\frac{\left(\sqrt{-4ac+b^2}\,ba-\sqrt{-4ac+b^2}\,bc+4a^2c-\right.}{a\left(4ac-b^2\right.}\right.}{c}$

```
[In] int(cos(x)^2/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] -x/c+sum(_R*ln(exp(I*x)+(8*c^3/b*a-2*b*c^2)*_R^3+(4*I/b*c^2*a-I*b*c)*_R^2+(
2*a*c/b-b+2*c^2/b)*_R+I/b*a+I/b*c),_R=RootOf((16*a^2*c^4-8*a*b^2*c^3+b^4*c^
2)*_Z^4+(8*a^2*c^2-6*a*b^2*c+8*a*c^3+b^4-2*b^2*c^2)*_Z^2+a^2+2*a*c-b^2+c^2)
)
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 971 vs. $2(196) = 392$.

Time = 0.39 (sec) , antiderivative size = 971, normalized size of antiderivative = 4.22

$$\int \frac{\cos^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx$$
$$= \frac{\sqrt{2}c\sqrt{-\frac{b^2-2\,ac-2\,c^2+(b^2c^2-4\,ac^3)\sqrt{\frac{b^2}{b^2c^4-4\,ac^5}}}{b^2c^2-4\,ac^3}}\log\left(\sqrt{2}(b^2c^3-4\,ac^4)\sqrt{\frac{b^2}{b^2c^4-4\,ac^5}}\sqrt{-\frac{b^2-2\,ac-2\,c^2+(b^2c^2-4\,ac^3)\sqrt{\frac{b^2}{b^2c^4-4\cdots}}}{b^2c^2-4\,ac^3}}\cdots\right.}{\cdots}$$

```
[In] integrate(cos(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] 1/4*(sqrt(2)*c*sqrt(-(b^2 - 2*a*c - 2*c^2 + (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b
^2*c^4 - 4*a*c^5)))/(b^2*c^2 - 4*a*c^3))*log(sqrt(2)*(b^2*c^3 - 4*a*c^4)*sq
rt(b^2/(b^2*c^4 - 4*a*c^5))*sqrt(-(b^2 - 2*a*c - 2*c^2 + (b^2*c^2 - 4*a*c^3
)*sqrt(b^2/(b^2*c^4 - 4*a*c^5)))/(b^2*c^2 - 4*a*c^3))*cos(x) + b^2*sin(x) +
 (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b^2*c^4 - 4*a*c^5))*sin(x) + 2*b*c) - sqrt(2
)*c*sqrt(-(b^2 - 2*a*c - 2*c^2 + (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b^2*c^4 - 4*
a*c^5)))/(b^2*c^2 - 4*a*c^3))*log(sqrt(2)*(b^2*c^3 - 4*a*c^4)*sqrt(b^2/(b^2
*c^4 - 4*a*c^5))*sqrt(-(b^2 - 2*a*c - 2*c^2 + (b^2*c^2 - 4*a*c^3)*sqrt(b^2/
(b^2*c^4 - 4*a*c^5)))/(b^2*c^2 - 4*a*c^3))*cos(x) - b^2*sin(x) - (b^2*c^2 -
 4*a*c^3)*sqrt(b^2/(b^2*c^4 - 4*a*c^5))*sin(x) - 2*b*c) - sqrt(2)*c*sqrt(-(
b^2 - 2*a*c - 2*c^2 - (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b^2*c^4 - 4*a*c^5)))/(b
^2*c^2 - 4*a*c^3))*log(sqrt(2)*(b^2*c^3 - 4*a*c^4)*sqrt(b^2/(b^2*c^4 - 4*a*
c^5))*sqrt(-(b^2 - 2*a*c - 2*c^2 - (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b^2*c^4 -
4*a*c^5)))/(b^2*c^2 - 4*a*c^3))*cos(x) + b^2*sin(x) - (b^2*c^2 - 4*a*c^3)*s
qrt(b^2/(b^2*c^4 - 4*a*c^5))*sin(x) + 2*b*c) + sqrt(2)*c*sqrt(-(b^2 - 2*a*c
 - 2*c^2 - (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b^2*c^4 - 4*a*c^5)))/(b^2*c^2 - 4*
a*c^3))*log(sqrt(2)*(b^2*c^3 - 4*a*c^4)*sqrt(b^2/(b^2*c^4 - 4*a*c^5))*sqrt(
-(b^2 - 2*a*c - 2*c^2 - (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b^2*c^4 - 4*a*c^5)))/
(b^2*c^2 - 4*a*c^3))*cos(x) - b^2*sin(x) + (b^2*c^2 - 4*a*c^3)*sqrt(b^2/(b^
2*c^4 - 4*a*c^5))*sin(x) - 2*b*c) - 4*x)/c
```

Sympy [F(-1)]

Timed out.

$$\int \frac{\cos^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(cos(x)**2/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Timed out
```

Maxima [F]

$$\int \frac{\cos^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\cos(x)^2}{c\sin(x)^2+b\sin(x)+a}\,dx$$

```
[In] integrate(cos(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] (c*integrate(2*(2*b^2*cos(3*x)^2 + 2*b^2*cos(x)^2 + 2*b^2*sin(3*x)^2 + 2*b^
2*sin(x)^2 + 4*(2*a^2 + 3*a*c + c^2)*cos(2*x)^2 + 2*(4*a*b + 3*b*c)*cos(x)*
sin(2*x) + 4*(2*a^2 + 3*a*c + c^2)*sin(2*x)^2 + b*c*sin(x) - (b*c*sin(3*x)
- b*c*sin(x) + 2*(a*c + c^2)*cos(2*x))*cos(4*x) - 2*(2*b^2*cos(x) + (4*a*b
+ 3*b*c)*sin(2*x))*cos(3*x) - 2*(a*c + c^2 + (4*a*b + 3*b*c)*sin(x))*cos(2*
x) + (b*c*cos(3*x) - b*c*cos(x) - 2*(a*c + c^2)*sin(2*x))*sin(4*x) - (4*b^2
*sin(x) + b*c - 2*(4*a*b + 3*b*c)*cos(2*x))*sin(3*x))/(c^3*cos(4*x)^2 + 4*b
^2*c*cos(3*x)^2 + 4*b^2*c*cos(x)^2 + c^3*sin(4*x)^2 + 4*b^2*c*sin(3*x)^2 +
4*b^2*c*sin(x)^2 + 4*b*c^2*sin(x) + c^3 + 4*(4*a^2*c + 4*a*c^2 + c^3)*cos(2
*x)^2 + 8*(2*a*b*c + b*c^2)*cos(x)*sin(2*x) + 4*(4*a^2*c + 4*a*c^2 + c^3)*s
in(2*x)^2 - 2*(2*b*c^2*sin(3*x) - 2*b*c^2*sin(x) - c^3 + 2*(2*a*c^2 + c^3)*
cos(2*x))*cos(4*x) - 8*(b^2*c*cos(x) + (2*a*b*c + b*c^2)*sin(2*x))*cos(3*x)
 - 4*(2*a*c^2 + c^3 + 2*(2*a*b*c + b*c^2)*sin(x))*cos(2*x) + 4*(b*c^2*cos(3
*x) - b*c^2*cos(x) - (2*a*c^2 + c^3)*sin(2*x))*sin(4*x) - 4*(2*b^2*c*sin(x)
 + b*c^2 - 2*(2*a*b*c + b*c^2)*cos(2*x))*sin(3*x)), x) - x)/c
```

Giac [F(-1)]

Timed out.

$$\int \frac{\cos^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Timed out}$$

```
[In] integrate(cos(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 29.24 (sec) , antiderivative size = 11164, normalized size of antiderivative = 48.54

$$\int \frac{\cos^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(cos(x)^2/(a + c*sin(x)^2 + b*sin(x)),x)
```

```
[Out] atan(((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2
 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(81
```

920*a*b^4 + 139264*a*c^4 + 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2 + 42598
4*a^2*c^3 + 458752*a^3*c^2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) - 24576*a
^4*b + 32768*a^2*b^3 + (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^
2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1
/2)*((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2
- 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 +
b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(1
6*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 + b*(-(4*a*c - b^2)^
3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^
2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 8519
68*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a
*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 1
6384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3
+ 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^
4 - 24576*a^3*b^3*c^3) - tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 1310
72*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 3
2768*a^3*b^3*c^2) + 131072*a^2*c^6 + 163840*a^3*c^5 - 65536*a^4*c^4 - 98304
*a^5*c^3 - 32768*a*b^2*c^5 + 32768*a*b^4*c^3 - 172032*a^2*b^2*c^4 - 24576*a
^2*b^4*c^2 + 114688*a^3*b^2*c^3 + 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c
^6 - 16384*a*b^6 + 16384*a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 95027
2*a^4*c^3 + 147456*a^5*c^2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a
^2*b^4*c - 98304*a^4*b^2*c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57
344*a*b^3*c^3 + 139264*a^2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 737
28*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/
2)*(32768*a*b^5 - 32768*a^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 2293
76*a^3*b*c^2 - 32768*a*b*c^4 + 131072*a^4*b*c) + 32768*a*c^5 - 24576*a^5*c
- 8192*a^2*b^4 + 8192*a^4*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4
*c^2 - 57344*a*b^2*c^3 + 16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4
*c) + 8192*a^2*b*c^2 + 32768*a*b*c^3 - 24576*a*b^3*c - 49152*a^3*b*c)*1i -
(-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a
*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(24576*a^4*b - tan(
x/2)*(81920*a*b^4 + 139264*a*c^4 + 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2
+ 425984*a^2*c^3 + 458752*a^3*c^2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) -
32768*a^2*b^3 + (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2
- 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(3
2768*a*c^5 - (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*
b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*
a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*
c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 + b*(-(4*a*c
- b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4
+ b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^
6 + 851968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 -
16384*a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2
*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*
b^5*c^3 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^

2*b^3*c^4 - 24576*a^3*b^3*c^3) + tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 131072*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) - 131072*a^2*c^6 - 163840*a^3*c^5 + 65536*a^4*c^4 + 98304*a^5*c^3 + 32768*a*b^2*c^5 - 32768*a*b^4*c^3 + 172032*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 114688*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c^6 - 16384*a*b^6 + 16384*a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 950272*a^4*c^3 + 147456*a^5*c^2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57344*a*b^3*c^3 + 139264*a^2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 229376*a^3*b*c^2 - 32768*a*b*c^4 + 131072*a^4*b*c) - 24576*a^5*c - 8192*a^2*b^4 + 8192*a^4*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4*c^2 - 57344*a*b^2*c^3 + 16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4*c) - 8192*a^2*b*c^2 - 32768*a*b*c^3 + 24576*a*b^3*c + 49152*a^3*b*c)*1i)/((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(81920*a*b^4 + 139264*a*c^4 + 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2 + 425984*a^2*c^3 + 458752*a^3*c^2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) - 24576*a^4*b + 32768*a^2*b^3 + (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) - tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 131072*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) + 131072*a^2*c^6 + 163840*a^3*c^5 - 65536*a^4*c^4 - 98304*a^5*c^3 - 32768*a*b^2*c^5 + 32768*a*b^4*c^3 - 172032*a^2*b^2*c^4 - 24576*a^2*b^4*c^2 + 114688*a^3*b^2*c^3 + 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c^6 - 16384*a*b^6 + 16384*a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 950272*a^4*c^3 + 147456*a^5*c^2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57344*a*b^3*c^3 + 139264*a^2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 229376*a^3*b*c^2 - 32768*a*b*c^4 + 131072*a^4*b*c) + 32768*a*c^5 - 24576*a^5*c - 8192*a^2*b^4 + 8192*a^4*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4*c^2 - 57344*a*b^2*c^3 + 16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4*c) + 8192*a

^2*b*c^2 + 32768*a*b*c^3 - 24576*a*b^3*c - 49152*a^3*b*c) + (-(8*a*c^3 + b*
(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*
a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(24576*a^4*b - tan(x/2)*(81920*a*b
^4 + 139264*a*c^4 + 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2 + 425984*a^2*c
^3 + 458752*a^3*c^2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) - 32768*a^2*b^3
+ (-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6
*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(32768*a*c^5 - (-
(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b
^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 + b*(-(4*
a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c
^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 + b*(-(4*a*c - b^2)^3)^(1/2
) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a
*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*
c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^
2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^
3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 13107
2*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 245
76*a^3*b^3*c^3) + tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 131072*a^2*
b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^
3*b^3*c^2) - 131072*a^2*c^6 - 163840*a^3*c^5 + 65536*a^4*c^4 + 98304*a^5*c^
3 + 32768*a*b^2*c^5 - 32768*a*b^4*c^3 + 172032*a^2*b^2*c^4 + 24576*a^2*b^4*
c^2 - 114688*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c^6 - 16
384*a*b^6 + 16384*a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 950272*a^4*c
^3 + 147456*a^5*c^2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a^2*b^4*
c - 98304*a^4*b^2*c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57344*a*b
^3*c^3 + 139264*a^2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*
b*c^2 - 106496*a^2*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/2)*(327
68*a*b^5 - 32768*a^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 229376*a^3*
b*c^2 - 32768*a*b*c^4 + 131072*a^4*b*c) - 24576*a^5*c - 8192*a^2*b^4 + 8192
*a^4*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4*c^2 - 57344*a*b^2*c^
3 + 16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4*c) - 8192*a^2*b*c^2
- 32768*a*b*c^3 + 24576*a*b^3*c + 49152*a^3*b*c) + 49152*a*c^3 + 147456*a^3
*c + 49152*a^4 + 2*tan(x/2)*(32768*a^3*b - 32768*a*b^3 + 32768*a*b*c^2 + 65
536*a^2*b*c) - 49152*a^2*b^2 + 147456*a^2*c^2 - 49152*a*b^2*c))*(-(8*a*c^3
+ b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*
(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*2i + atan(((-(8*a*c^3 - b*(-(4
*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*
c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(81920*a*b^4 + 139264*a*c^4
+ 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2 + 425984*a^2*c^3 + 458752*a^3*c^
2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) - 24576*a^4*b + 32768*a^2*b^3 + (-
(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b
^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*
a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c
^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2
) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a

```
*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^
2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*
(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c
^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2
*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688
*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 1638
40*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) -
tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 131072*a^2*b*c^5 + 262144*a^3
*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) + 13107
2*a^2*c^6 + 163840*a^3*c^5 - 65536*a^4*c^4 - 98304*a^5*c^3 - 32768*a*b^2*c^
5 + 32768*a*b^4*c^3 - 172032*a^2*b^2*c^4 - 24576*a^2*b^4*c^2 + 114688*a^3*b
^2*c^3 + 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c^6 - 16384*a*b^6 + 16384*
a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 950272*a^4*c^3 + 147456*a^5*c^
2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*
c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57344*a*b^3*c^3 + 139264*a^
2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2
*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a
^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 229376*a^3*b*c^2 - 32768*a*b*
c^4 + 131072*a^4*b*c) + 32768*a*c^5 - 24576*a^5*c - 8192*a^2*b^4 + 8192*a^4
*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4*c^2 - 57344*a*b^2*c^3 +
16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4*c) + 8192*a^2*b*c^2 + 32
768*a*b*c^3 - 24576*a*b^3*c - 49152*a^3*b*c)*1i - (-(8*a*c^3 - b*(-(4*a*c -
 b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 +
b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(24576*a^4*b - tan(x/2)*(81920*a*b^4 + 13926
4*a*c^4 + 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2 + 425984*a^2*c^3 + 45875
2*a^3*c^2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) - 32768*a^2*b^3 + (-(8*a*c
^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/
(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(32768*a*c^5 - (-(8*a*c^3 -
 b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(
16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)
^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c
^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 +
8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3))
)^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 1966
08*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968
*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2
- 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^
6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3
*c^3) + tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 131072*a^2*b*c^5 + 26
2144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2)
 - 131072*a^2*c^6 - 163840*a^3*c^5 + 65536*a^4*c^4 + 98304*a^5*c^3 + 32768*
a*b^2*c^5 - 32768*a*b^4*c^3 + 172032*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 1146
88*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c^6 - 16384*a*b^6
+ 16384*a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 950272*a^4*c^3 + 14745
6*a^5*c^2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*
```

a^4*b^2*c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57344*a*b^3*c^3 + 139264*a^2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 229376*a^3*b*c^2 - 32768*a*b*c^4 + 131072*a^4*b*c) - 24576*a^5*c - 8192*a^2*b^4 + 8192*a^4*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4*c^2 - 57344*a*b^2*c^3 + 16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4*c) - 8192*a^2*b*c^2 - 32768*a*b*c^3 + 24576*a*b^3*c + 49152*a^3*b*c)*1i)/((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(81920*a*b^4 + 139264*a*c^4 + 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2 + 425984*a^2*c^3 + 458752*a^3*c^2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) - 24576*a^4*b + 32768*a^2*b^3 + (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) - tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 131072*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) + 131072*a^2*c^6 + 163840*a^3*c^5 - 65536*a^4*c^4 - 98304*a^5*c^3 - 32768*a*b^2*c^5 + 32768*a*b^4*c^3 - 172032*a^2*b^2*c^4 - 24576*a^2*b^4*c^2 + 114688*a^3*b^2*c^3 + 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c^6 - 16384*a*b^6 + 16384*a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 950272*a^4*c^3 + 147456*a^5*c^2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57344*a*b^3*c^3 + 139264*a^2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 229376*a^3*b*c^2 - 32768*a*b*c^4 + 131072*a^4*b*c) + 32768*a*c^5 - 24576*a^5*c - 8192*a^2*b^4 + 8192*a^4*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4*c^2 - 57344*a*b^2*c^3 + 16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4*c) + 8192*a^2*b*c^2 + 32768*a*b*c^3 - 24576*a*b^3*c - 49152*a^3*b*c) + (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(24576*a^4*b - tan(x/2)*(81920*a*b^4 + 139264*a*c^4 + 196608*a^4*c + 24576*a^5 - 98304*a^3*b^2 + 425984*a^2*c^3 + 458752*a^3*c^2 - 212992*a*b^2*c^2 - 327680*a^2*b^2*c) - 32768*a^2*b^3 + (-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(32768*a*c^5 - (-(8*a*c^3 - b*(-(4*a*c - b

^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*((-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*(tan(x/2)*(524288*a^2*c^7 + 1179648*a^3*c^6 + 851968*a^4*c^5 + 196608*a^5*c^4 - 131072*a*b^2*c^6 + 139264*a*b^4*c^4 - 16384*a*b^6*c^2 - 851968*a^2*b^2*c^5 + 147456*a^2*b^4*c^3 - 540672*a^3*b^2*c^4 + 16384*a^3*b^4*c^2 - 114688*a^4*b^2*c^3) - 32768*a*b^3*c^5 + 24576*a*b^5*c^3 + 131072*a^2*b*c^6 + 163840*a^3*b*c^5 + 98304*a^4*b*c^4 - 139264*a^2*b^3*c^4 - 24576*a^3*b^3*c^3) + tan(x/2)*(32768*a*b^5*c^2 - 32768*a*b^3*c^4 + 131072*a^2*b*c^5 + 262144*a^3*b*c^4 + 131072*a^4*b*c^3 - 196608*a^2*b^3*c^3 - 32768*a^3*b^3*c^2) - 131072*a^2*c^6 - 163840*a^3*c^5 + 65536*a^4*c^4 + 98304*a^5*c^3 + 32768*a*b^2*c^5 - 32768*a*b^4*c^3 + 172032*a^2*b^2*c^4 + 24576*a^2*b^4*c^2 - 114688*a^3*b^2*c^3 - 24576*a^4*b^2*c^2) + tan(x/2)*(131072*a*c^6 - 16384*a*b^6 + 16384*a^3*b^4 + 983040*a^2*c^5 + 1654784*a^3*c^4 + 950272*a^4*c^3 + 147456*a^5*c^2 - 344064*a*b^2*c^4 + 229376*a*b^4*c^2 + 131072*a^2*b^4*c - 98304*a^4*b^2*c - 1228800*a^2*b^2*c^3 - 540672*a^3*b^2*c^2) - 57344*a*b^3*c^3 + 139264*a^2*b*c^4 + 114688*a^3*b*c^3 - 24576*a^3*b^3*c + 73728*a^4*b*c^2 - 106496*a^2*b^3*c^2 + 32768*a*b*c^5 + 24576*a*b^5*c) - tan(x/2)*(32768*a*b^5 - 32768*a^3*b^3 + 65536*a^2*b*c^3 - 196608*a^2*b^3*c + 229376*a^3*b*c^2 - 32768*a*b*c^4 + 131072*a^4*b*c) - 24576*a^5*c - 8192*a^2*b^4 + 8192*a^4*b^2 + 172032*a^2*c^4 + 221184*a^3*c^3 + 57344*a^4*c^2 - 57344*a*b^2*c^3 + 16384*a^3*b^2*c - 147456*a^2*b^2*c^2 + 24576*a*b^4*c) - 8192*a^2*b*c^2 - 32768*a*b*c^3 + 24576*a*b^3*c + 49152*a^3*b*c) + 49152*a*c^3 + 147456*a^3*c + 49152*a^4 + 2*tan(x/2)*(32768*a^3*b - 32768*a*b^3 + 32768*a*b*c^2 + 65536*a^2*b*c) - 49152*a^2*b^2 + 147456*a^2*c^2 - 49152*a*b^2*c))*(-(8*a*c^3 - b*(-(4*a*c - b^2)^3)^(1/2) + b^4 + 8*a^2*c^2 - 2*b^2*c^2 - 6*a*b^2*c)/(2*(16*a^2*c^4 + b^4*c^2 - 8*a*b^2*c^3)))^(1/2)*2i - (2*atan((196608*a^4*tan(x/2))/(16384*a*c^3 - 32768*a^3*c + 196608*a^4 + 98304*a^2*b^2 - 65536*a^2*c^2 + (147456*a^5)/c - (16384*a*b^4)/c - (196608*a^3*b^2)/c + (32768*a^2*b^4)/c^2 - (32768*a^4*b^2)/c^2) - (147456*a^5*tan(x/2))/(16384*a*b^4 - 16384*a*c^4 - 196608*a^4*c - 147456*a^5 + 196608*a^3*b^2 + 65536*a^2*c^3 + 32768*a^3*c^2 - 98304*a^2*b^2*c - (32768*a^2*b^4)/c + (32768*a^4*b^2)/c) + (32768*a^2*b^4*tan(x/2))/(16384*a*c^5 + 147456*a^5*c + 32768*a^2*b^4 - 32768*a^4*b^2 - 65536*a^2*c^4 - 32768*a^3*c^3 + 196608*a^4*c^2 - 196608*a^3*b^2*c + 98304*a^2*b^2*c^2 - 16384*a*b^4*c) - (32768*a^4*b^2*tan(x/2))/(16384*a*c^5 + 147456*a^5*c + 32768*a^2*b^4 - 32768*a^4*b^2 - 65536*a^2*c^4 - 32768*a^3*c^3 + 196608*a^4*c^2 - 196608*a^3*b^2*c + 98304*a^2*b^2*c^2 - 16384*a*b^4*c) + (16384*a*b^4*tan(x/2))/(16384*a*b^4 - 16384*a*c^4 - 196608*a^4*c - 147456*a^5 + 196608*a^3*b^2 + 65536*a^2*c^3 + 32768*a^3*c^2 - 98304*a^2*b^2*c - (32768*a^2*b^4)/c + (32768*a^4*b^2)/c) + (16384*a*c^3*tan(x/2))/(16384*a*c^3 - 32768*a^3*c + 196608*a^4 + 98304*a^2*b^2 - 65536*a^2*c^2 + (147456*a^5)/c - (16384*a*b^4)/c - (196608*a^3*b^2)/c + (32768*a^2*b^4)/c^2 - (32768*a^4*b^2)/c^2) - (32768*a^3*c*tan(x/2))/(16384*a*c^3 - 32768*a^3*c + 196608*a^4 + 98304*a^2*b^2 - 65536*a

```
^2*c^2 + (147456*a^5)/c - (16384*a*b^4)/c - (196608*a^3*b^2)/c + (32768*a^2
*b^4)/c^2 - (32768*a^4*b^2)/c^2) + (196608*a^3*b^2*tan(x/2))/(16384*a*b^4 -
 16384*a*c^4 - 196608*a^4*c - 147456*a^5 + 196608*a^3*b^2 + 65536*a^2*c^3 +
 32768*a^3*c^2 - 98304*a^2*b^2*c - (32768*a^2*b^4)/c + (32768*a^4*b^2)/c) +
 (98304*a^2*b^2*tan(x/2))/(16384*a*c^3 - 32768*a^3*c + 196608*a^4 + 98304*a
^2*b^2 - 65536*a^2*c^2 + (147456*a^5)/c - (16384*a*b^4)/c - (196608*a^3*b^2
)/c + (32768*a^2*b^4)/c^2 - (32768*a^4*b^2)/c^2) - (65536*a^2*c^2*tan(x/2))
/(16384*a*c^3 - 32768*a^3*c + 196608*a^4 + 98304*a^2*b^2 - 65536*a^2*c^2 +
(147456*a^5)/c - (16384*a*b^4)/c - (196608*a^3*b^2)/c + (32768*a^2*b^4)/c^2
 - (32768*a^4*b^2)/c^2)))/c
```

3.11 $\int \frac{\cos(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 17, antiderivative size = 35

$$\int \frac{\cos(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{2\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}}$$

```
[Out] -2*arctanh((b+2*c*sin(x))/(-4*a*c+b^2)^(1/2))/(-4*a*c+b^2)^(1/2)
```

Rubi [A] (verified)

Time = 0.05 (sec) , antiderivative size = 35, normalized size of antiderivative = 1.00, number of steps used = 3, number of rules used = 3, $\frac{\text{number of rules}}{\text{integrand size}} = 0.176$, Rules used = {3339, 632, 212}

$$\int \frac{\cos(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{2\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}}$$

```
[In] Int[Cos[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] (-2*ArcTanh[(b + 2*c*Sin[x])/Sqrt[b^2 - 4*a*c]])/Sqrt[b^2 - 4*a*c]
```

Rule 212

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(1/(Rt[a, 2]*Rt[-b, 2]))*
ArcTanh[Rt[-b, 2]*(x/Rt[a, 2])], x] /; FreeQ[{a, b}, x] && NegQ[a/b] && (Gt
Q[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
```

```
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}
\text{integral} &= \text{Subst}\left(\int \frac{1}{a+bx+cx^2}\,dx, x, \sin(x)\right)\\
&= -\left(2\text{Subst}\left(\int \frac{1}{b^2-4ac-x^2}\,dx, x, b+2c\sin(x)\right)\right)\\
&= -\frac{2\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}}
\end{aligned}$$

Mathematica [A] (verified)

Time = 0.02 (sec) , antiderivative size = 35, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{2\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}}$$

```
[In] Integrate[Cos[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
[Out] (-2*ArcTanh[(b + 2*c*Sin[x])/Sqrt[b^2 - 4*a*c]])/Sqrt[b^2 - 4*a*c]
```

Maple [A] (verified)

Time = 0.52 (sec) , antiderivative size = 36, normalized size of antiderivative = 1.03

method	result	size
derivativedivides	$\frac{2\arctan\left(\frac{b+2\sin(x)c}{\sqrt{4ac-b^2}}\right)}{\sqrt{4ac-b^2}}$	36
default	$\frac{2\arctan\left(\frac{b+2\sin(x)c}{\sqrt{4ac-b^2}}\right)}{\sqrt{4ac-b^2}}$	36
risch	$-\frac{\ln\left(e^{2ix}+\frac{i\left(b\sqrt{-4ac+b^2}-4ac+b^2\right)e^{ix}}{c\sqrt{-4ac+b^2}}-1\right)}{\sqrt{-4ac+b^2}}+\frac{\ln\left(e^{2ix}+\frac{i\left(b\sqrt{-4ac+b^2}+4ac-b^2\right)e^{ix}}{c\sqrt{-4ac+b^2}}-1\right)}{\sqrt{-4ac+b^2}}$	125

```
[In] int(cos(x)/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
[Out] 2/(4*a*c-b^2)^(1/2)*arctan((b+2*sin(x)*c)/(4*a*c-b^2)^(1/2))
```

Fricas [A] (verification not implemented)

none

Time = 0.35 (sec) , antiderivative size = 139, normalized size of antiderivative = 3.97

$$\int \frac{\cos(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \left[\frac{\log\left(-\frac{2\,c^2\cos(x)^2-2\,bc\sin(x)-b^2+2\,ac-2\,c^2+\sqrt{b^2-4\,ac}(2\,c\sin(x)+b)}{c\cos(x)^2-b\sin(x)-a-c}\right)}{\sqrt{b^2-4\,ac}}, -\frac{2\sqrt{-b^2+4\,ac}\arctan\left(-\frac{\sqrt{-b^2+4\,ac}(2\,c\sin(x)+b)}{b^2-4\,ac}\right)}{b^2-4\,ac}\right]$$

```
[In] integrate(cos(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
[Out] [log(-(2*c^2*cos(x)^2 - 2*b*c*sin(x) - b^2 + 2*a*c - 2*c^2 + sqrt(b^2 - 4*a
*c)*(2*c*sin(x) + b))/(c*cos(x)^2 - b*sin(x) - a - c))/sqrt(b^2 - 4*a*c), -
2*sqrt(-b^2 + 4*a*c)*arctan(-sqrt(-b^2 + 4*a*c)*(2*c*sin(x) + b)/(b^2 - 4*a
*c))/(b^2 - 4*a*c)]
```

Sympy [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 99 vs. 2(36) = 72.

Time = 1.49 (sec) , antiderivative size = 99, normalized size of antiderivative = 2.83

$$\int \frac{\cos(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \begin{cases} \frac{\log\left(\frac{a}{b}+\sin(x)\right)}{b} & \text{for } c=0 \\ -\frac{2}{b+2c\sin(x)} & \text{for } a=\frac{b^2}{4c} \\ \frac{\log\left(\frac{b}{2c}+\sin(x)-\frac{\sqrt{-4ac+b^2}}{2c}\right)}{\sqrt{-4ac+b^2}}-\frac{\log\left(\frac{b}{2c}+\sin(x)+\frac{\sqrt{-4ac+b^2}}{2c}\right)}{\sqrt{-4ac+b^2}} & \text{otherwise} \end{cases}$$

```
[In] integrate(cos(x)/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Piecewise((log(a/b + sin(x))/b, Eq(c, 0)), (-2/(b + 2*c*sin(x)), Eq(a, b**2
/(4*c))), (log(b/(2*c) + sin(x) - sqrt(-4*a*c + b**2)/(2*c))/sqrt(-4*a*c +
b**2) - log(b/(2*c) + sin(x) + sqrt(-4*a*c + b**2)/(2*c))/sqrt(-4*a*c + b**
2), True))
```

Maxima [F(-2)]

Exception generated.

$$\int \frac{\cos(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Exception raised: ValueError}$$

```
[In] integrate(cos(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] Exception raised: ValueError >> Computation failed since Maxima requested a
dditional constraints; using the 'assume' command before evaluation *may* h
elp (example of legal syntax is 'assume(4*a*c-b^2>0)', see `assume?` for mo
re deta
```

Giac [A] (verification not implemented)

none

Time = 0.31 (sec) , antiderivative size = 35, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \frac{2\arctan\left(\frac{2c\sin(x)+b}{\sqrt{-b^2+4ac}}\right)}{\sqrt{-b^2 + 4ac}}$$

```
[In] integrate(cos(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] 2*arctan((2*c*sin(x) + b)/sqrt(-b^2 + 4*a*c))/sqrt(-b^2 + 4*a*c)
```

Mupad [B] (verification not implemented)

Time = 15.68 (sec) , antiderivative size = 47, normalized size of antiderivative = 1.34

$$\int \frac{\cos(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \frac{2\,\mathrm{atan}\left(\frac{b}{\sqrt{4\,a\,c-b^2}} + \frac{2\,c\,\sin(x)}{\sqrt{4\,a\,c-b^2}}\right)}{\sqrt{4\,a\,c - b^2}}$$

```
[In] int(cos(x)/(a + c*sin(x)^2 + b*sin(x)),x)
```

```
[Out] (2*atan(b/(4*a*c - b^2)^(1/2) + (2*c*sin(x))/(4*a*c - b^2)^(1/2)))/(4*a*c -
 b^2)^(1/2)
```

3.12 $\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 17, antiderivative size = 128

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{(b^2-2ac-2c^2)\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{(a-b+c)(a+b+c)\sqrt{b^2-4ac}} - \frac{\log(1-\sin(x))}{2(a+b+c)} + \frac{\log(1+\sin(x))}{2(a-b+c)} - \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2(a-b+c)(a+b+c)}$$

```
[Out] -1/2*ln(1-sin(x))/(a+b+c)+1/2*ln(1+sin(x))/(a-b+c)-1/2*b*ln(a+b*sin(x)+c*si
n(x)^2)/(a-b+c)/(a+b+c)+(-2*a*c+b^2-2*c^2)*arctanh((b+2*c*sin(x))/(-4*a*c+b
^2)^(1/2))/(a-b+c)/(a+b+c)/(-4*a*c+b^2)^(1/2)
```

Rubi [A] (verified)

Time = 0.19 (sec) , antiderivative size = 128, normalized size of antiderivative = 1.00, number of steps used = 9, number of rules used = 8, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.471, Rules used = {3339, 995, 648, 632, 212, 642, 647, 31}

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{(-2ac+b^2-2c^2)\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{(a-b+c)(a+b+c)\sqrt{b^2-4ac}} - \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2(a-b+c)(a+b+c)} - \frac{\log(1-\sin(x))}{2(a+b+c)} + \frac{\log(\sin(x)+1)}{2(a-b+c)}$$

```
[In] Int[Sec[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] ((b^2 - 2*a*c - 2*c^2)*ArcTanh[(b + 2*c*Sin[x])/Sqrt[b^2 - 4*a*c]])/((a - b
 + c)*(a + b + c)*Sqrt[b^2 - 4*a*c]) - Log[1 - Sin[x]]/(2*(a + b + c)) + Lo
```

```
g[1 + Sin[x]]/(2*(a - b + c)) - (b*Log[a + b*Sin[x] + c*Sin[x]^2])/(2*(a -
b + c)*(a + b + c))
```

Rule 31

```
Int[((a_) + (b_.)*(x_))^(-1), x_Symbol] :> Simp[Log[RemoveContent[a + b*x,
x]]/b, x] /; FreeQ[{a, b}, x]
```

Rule 212

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(1/(Rt[a, 2]*Rt[-b, 2]))*
ArcTanh[Rt[-b, 2]*(x/Rt[a, 2])], x] /; FreeQ[{a, b}, x] && NegQ[a/b] && (Gt
Q[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 642

```
Int[((d_) + (e_.)*(x_))/((a_.) + (b_.)*(x_) + (c_.)*(x_)^2), x_Symbol] :> S
imp[d*(Log[RemoveContent[a + b*x + c*x^2, x]]/b), x] /; FreeQ[{a, b, c, d,
e}, x] && EqQ[2*c*d - b*e, 0]
```

Rule 647

```
Int[((d_) + (e_.)*(x_))/((a_) + (c_.)*(x_)^2), x_Symbol] :> With[{q = Rt[(-
a)*c, 2]}, Dist[e/2 + c*(d/(2*q)), Int[1/(-q + c*x), x], x] + Dist[e/2 - c*
(d/(2*q)), Int[1/(q + c*x), x], x]] /; FreeQ[{a, c, d, e}, x] && NiceSqrtQ[
(-a)*c]
```

Rule 648

```
Int[((d_.) + (e_.)*(x_))/((a_) + (b_.)*(x_) + (c_.)*(x_)^2), x_Symbol] :> D
ist[(2*c*d - b*e)/(2*c), Int[1/(a + b*x + c*x^2), x], x] + Dist[e/(2*c), In
t[(b + 2*c*x)/(a + b*x + c*x^2), x], x] /; FreeQ[{a, b, c, d, e}, x] && NeQ
[2*c*d - b*e, 0] && NeQ[b^2 - 4*a*c, 0] &&  !NiceSqrtQ[b^2 - 4*a*c]
```

Rule 995

```
Int[1/(((a_) + (b_.)*(x_) + (c_.)*(x_)^2)*((d_) + (f_.)*(x_)^2)), x_Symbol]
 :> With[{q = c^2*d^2 + b^2*d*f - 2*a*c*d*f + a^2*f^2}, Dist[1/q, Int[(c^2*
d + b^2*f - a*c*f + b*c*f*x)/(a + b*x + c*x^2), x], x] - Dist[1/q, Int[(c*d
*f - a*f^2 + b*f^2*x)/(d + f*x^2), x], x] /; NeQ[q, 0]] /; FreeQ[{a, b, c,
d, f}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}
\text{integral} &= \text{Subst}\left(\int \frac{1}{(1-x^2)(a+bx+cx^2)}\,dx, x, \sin(x)\right)\\
&= -\frac{\text{Subst}\left(\int \frac{-a-c+bx}{1-x^2}\,dx, x, \sin(x)\right)}{(a-b+c)(a+b+c)} + \frac{\text{Subst}\left(\int \frac{-b^2+ac+c^2-bcx}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{(a-b+c)(a+b+c)}\\
&= -\frac{\text{Subst}\left(\int \frac{1}{-1-x}\,dx, x, \sin(x)\right)}{2(a-b+c)} + \frac{\text{Subst}\left(\int \frac{1}{1-x}\,dx, x, \sin(x)\right)}{2(a+b+c)}\\
&\quad - \frac{b\text{Subst}\left(\int \frac{b+2cx}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{2(a-b+c)(a+b+c)}\\
&\quad - \frac{(b^2-2c(a+c))\,\text{Subst}\left(\int \frac{1}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{2(a-b+c)(a+b+c)}\\
&= -\frac{\log(1-\sin(x))}{2(a+b+c)} + \frac{\log(1+\sin(x))}{2(a-b+c)} - \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2(a-b+c)(a+b+c)}\\
&\quad + \frac{(b^2-2c(a+c))\,\text{Subst}\left(\int \frac{1}{b^2-4ac-x^2}\,dx, x, b+2c\sin(x)\right)}{(a-b+c)(a+b+c)}\\
&= \frac{(b^2-2c(a+c))\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{(a-b+c)(a+b+c)\sqrt{b^2-4ac}} - \frac{\log(1-\sin(x))}{2(a+b+c)}\\
&\quad + \frac{\log(1+\sin(x))}{2(a-b+c)} - \frac{b\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2(a-b+c)(a+b+c)}
\end{aligned}$$

Mathematica [A] (verified)

Time = 0.24 (sec) , antiderivative size = 119, normalized size of antiderivative = 0.93

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{(-2b^2+4c(a+c))\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)+\sqrt{b^2-4ac}((a-b+c)\log(1-\sin(x))-(a+b+c)\log(1+\sin}{2(a-b+c)(a+b+c)\sqrt{b^2-4ac}}$$

```
[In] Integrate[Sec[x]/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -1/2*((-2*b^2 + 4*c*(a + c))*ArcTanh[(b + 2*c*Sin[x])/Sqrt[b^2 - 4*a*c]] +
Sqrt[b^2 - 4*a*c]*((a - b + c)*Log[1 - Sin[x]] - (a + b + c)*Log[1 + Sin[x]
] + b*Log[a + b*Sin[x] + c*Sin[x]^2]))/((a - b + c)*(a + b + c)*Sqrt[b^2 -
4*a*c])
```

Maple [A] (verified)

Time = 0.86 (sec) , antiderivative size = 118, normalized size of antiderivative = 0.92

method	result	size
default	$\frac{\ln(1+\sin(x))}{2a-2b+2c}+\frac{-\frac{b\ln\left(a+b\sin(x)+c\left(\sin^2(x)\right)\right)}{2}+\frac{2\left(ac-\frac{1}{2}b^2+c^2\right)\arctan\left(\frac{b+2\sin(x)c}{\sqrt{4ac-b^2}}\right)}{\sqrt{4ac-b^2}}}{(a-b+c)(a+b+c)}-\frac{\ln(\sin(x)-1)}{2a+2b+2c}$	118
risch	Expression too large to display	1369

```
[In] int(sec(x)/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 1/(2*a-2*b+2*c)*ln(1+sin(x))+1/(a-b+c)/(a+b+c)*(-1/2*b*ln(a+b*sin(x)+c*sin(
x)^2)+2*(a*c-1/2*b^2+c^2)/(4*a*c-b^2)^(1/2)*arctan((b+2*sin(x)*c)/(4*a*c-b^
2)^(1/2)))-1/(2*a+2*b+2*c)*ln(sin(x)-1)
```

Fricas [A] (verification not implemented)

none

Time = 0.79 (sec) , antiderivative size = 482, normalized size of antiderivative = 3.77

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$
$$=\left[-\frac{(b^2-2\,ac-2\,c^2)\sqrt{b^2-4\,ac}\log\left(-\frac{2\,c^2\cos(x)^2-2\,bc\sin(x)-b^2+2\,ac-2\,c^2+\sqrt{b^2-4\,ac}(2\,c\sin(x)+b)}{c\cos(x)^2-b\sin(x)-a-c}\right)+(b^3-4\,abc)\log}{2\,(}\right.$$

```
[In] integrate(sec(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] [-1/2*((b^2 - 2*a*c - 2*c^2)*sqrt(b^2 - 4*a*c)*log(-(2*c^2*cos(x)^2 - 2*b*c
*sin(x) - b^2 + 2*a*c - 2*c^2 + sqrt(b^2 - 4*a*c)*(2*c*sin(x) + b))/(c*cos(
x)^2 - b*sin(x) - a - c)) + (b^3 - 4*a*b*c)*log(-c*cos(x)^2 + b*sin(x) + a
+ c) - (a*b^2 + b^3 - 4*a*c^2 - (4*a^2 + 4*a*b - b^2)*c)*log(sin(x) + 1) +
(a*b^2 - b^3 - 4*a*c^2 - (4*a^2 - 4*a*b - b^2)*c)*log(-sin(x) + 1))/(a^2*b^
2 - b^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c), 1/2*(2*(b^2
 - 2*a*c - 2*c^2)*sqrt(-b^2 + 4*a*c)*arctan(-sqrt(-b^2 + 4*a*c)*(2*c*sin(x)
 + b)/(b^2 - 4*a*c)) - (b^3 - 4*a*b*c)*log(-c*cos(x)^2 + b*sin(x) + a + c)
+ (a*b^2 + b^3 - 4*a*c^2 - (4*a^2 + 4*a*b - b^2)*c)*log(sin(x) + 1) - (a*b^
2 - b^3 - 4*a*c^2 - (4*a^2 - 4*a*b - b^2)*c)*log(-sin(x) + 1))/(a^2*b^2 - b
^4 - 4*a*c^3 - (8*a^2 - b^2)*c^2 - 2*(2*a^3 - 3*a*b^2)*c)]
```

Sympy [F]

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$

```
[In] integrate(sec(x)/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Integral(sec(x)/(a + b*sin(x) + c*sin(x)**2), x)
```

Maxima [F(-2)]

Exception generated.

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Exception raised: ValueError}$$

```
[In] integrate(sec(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] Exception raised: ValueError >> Computation failed since Maxima requested a
dditional constraints; using the 'assume' command before evaluation *may* h
elp (example of legal syntax is 'assume(4*a*c-b^2>0)', see `assume?` for mo
re deta
```

Giac [A] (verification not implemented)

none

Time = 0.32 (sec) , antiderivative size = 131, normalized size of antiderivative = 1.02

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{b\log\left(c\sin(x)^2+b\sin(x)+a\right)}{2\,(a^2-b^2+2\,ac+c^2)} - \frac{(b^2-2\,ac-2\,c^2)\arctan\left(\frac{2\,c\sin(x)+b}{\sqrt{-b^2+4\,ac}}\right)}{(a^2-b^2+2\,ac+c^2)\sqrt{-b^2+4\,ac}} + \frac{\log(\sin(x)+1)}{2\,(a-b+c)} - \frac{\log(-\sin(x)+1)}{2\,(a+b+c)}$$

```
[In] integrate(sec(x)/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] -1/2*b*log(c*sin(x)^2 + b*sin(x) + a)/(a^2 - b^2 + 2*a*c + c^2) - (b^2 - 2*
a*c - 2*c^2)*arctan((2*c*sin(x) + b)/sqrt(-b^2 + 4*a*c))/((a^2 - b^2 + 2*a*
c + c^2)*sqrt(-b^2 + 4*a*c)) + 1/2*log(sin(x) + 1)/(a - b + c) - 1/2*log(-s
in(x) + 1)/(a + b + c)
```

Mupad [B] (verification not implemented)

Time = 18.91 (sec) , antiderivative size = 1001, normalized size of antiderivative = 7.82

$$\int \frac{\sec(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{\ln\left(\sin\left(x\right)+1\right)}{2\ \left(a-b+c\right)}-\frac{\ln\left(\sin\left(x\right)-1\right)}{2\ \left(a+b+c\right)}$$
$$+\frac{\ln\left(4\,c^3\,\sin\left(x\right)+b\,c^2+\frac{\left(a\,\left(4\,b\,c-2\,c\,\sqrt{b^2-4\,a\,c}\right)-b^3+b^2\,\sqrt{b^2-4\,a\,c}-2\,c^2\,\sqrt{b^2-4\,a\,c}\right)\,\left(8\,a\,c^3+\sin(x)\,\left(-3\,b^3\,c+12\,b\,c^3+12\,a\,b\,c^2\right)+4\cdots\right.}{\cdots}\right.}{\cdots}$$
$$+\frac{\ln\left(4\,c^3\,\sin\left(x\right)+b\,c^2+\frac{\left(a\,\left(4\,b\,c+2\,c\,\sqrt{b^2-4\,a\,c}\right)-b^3-b^2\,\sqrt{b^2-4\,a\,c}+2\,c^2\,\sqrt{b^2-4\,a\,c}\right)\,\left(8\,a\,c^3+\sin(x)\,\left(-3\,b^3\,c+12\,b\,c^3+12\,a\,b\,c^2\right)+4\cdots\right.}{\cdots}\right.}{\cdots}$$

```
[In] int(1/(cos(x)*(a + c*sin(x)^2 + b*sin(x))),x)
```

```
[Out] log(sin(x) + 1)/(2*(a - b + c)) - log(sin(x) - 1)/(2*(a + b + c)) + (log(4*
c^3*sin(x) + b*c^2 + ((a*(4*b*c - 2*c*(b^2 - 4*a*c)^(1/2)) - b^3 + b^2*(b^2
 - 4*a*c)^(1/2) - 2*c^2*(b^2 - 4*a*c)^(1/2))*(8*a*c^3 + sin(x)*(12*b*c^3 -
3*b^3*c + 12*a*b*c^2) + 4*c^4 + 4*a^2*c^2 + 3*b^2*c^2 - ((a*(4*b*c - 2*c*(b
^2 - 4*a*c)^(1/2)) - b^3 + b^2*(b^2 - 4*a*c)^(1/2) - 2*c^2*(b^2 - 4*a*c)^(1
/2))*(sin(x)*(8*a*c^4 + 6*b^4*c + 8*c^5 - 8*a^2*c^3 - 8*a^3*c^2 - 6*b^2*c^3
 - 20*a*b^2*c^2 + 2*a^2*b^2*c) + 4*b*c^4 + 4*b^3*c^2 - 28*a^2*b*c^2 - 24*a*
b*c^3 + 8*a*b^3*c))/(b^2*(12*a*c + 2*a^2 - 2*b^2 + 2*c^2) - 4*a*c*(4*a*c +
2*a^2 + 2*c^2)) - a*b^2*c))/(b^2*(12*a*c + 2*a^2 - 2*b^2 + 2*c^2) - 4*a*c*(
4*a*c + 2*a^2 + 2*c^2)))*(a*(4*b*c - 2*c*(b^2 - 4*a*c)^(1/2)) - b^3 + b^2*(
b^2 - 4*a*c)^(1/2) - 2*c^2*(b^2 - 4*a*c)^(1/2)))/(b^2*(12*a*c + 2*a^2 - 2*b
^2 + 2*c^2) - 4*a*c*(4*a*c + 2*a^2 + 2*c^2)) + (log(4*c^3*sin(x) + b*c^2 +
((a*(4*b*c + 2*c*(b^2 - 4*a*c)^(1/2)) - b^3 - b^2*(b^2 - 4*a*c)^(1/2) + 2*c
^2*(b^2 - 4*a*c)^(1/2))*(8*a*c^3 + sin(x)*(12*b*c^3 - 3*b^3*c + 12*a*b*c^2)
 + 4*c^4 + 4*a^2*c^2 + 3*b^2*c^2 - ((a*(4*b*c + 2*c*(b^2 - 4*a*c)^(1/2)) -
b^3 - b^2*(b^2 - 4*a*c)^(1/2) + 2*c^2*(b^2 - 4*a*c)^(1/2))*(sin(x)*(8*a*c^4
 + 6*b^4*c + 8*c^5 - 8*a^2*c^3 - 8*a^3*c^2 - 6*b^2*c^3 - 20*a*b^2*c^2 + 2*a
^2*b^2*c) + 4*b*c^4 + 4*b^3*c^2 - 28*a^2*b*c^2 - 24*a*b*c^3 + 8*a*b^3*c))/(
b^2*(12*a*c + 2*a^2 - 2*b^2 + 2*c^2) - 4*a*c*(4*a*c + 2*a^2 + 2*c^2)) - a*b
^2*c))/(b^2*(12*a*c + 2*a^2 - 2*b^2 + 2*c^2) - 4*a*c*(4*a*c + 2*a^2 + 2*c^2
)))*(a*(4*b*c + 2*c*(b^2 - 4*a*c)^(1/2)) - b^3 - b^2*(b^2 - 4*a*c)^(1/2) +
2*c^2*(b^2 - 4*a*c)^(1/2)))/(b^2*(12*a*c + 2*a^2 - 2*b^2 + 2*c^2) - 4*a*c*(
4*a*c + 2*a^2 + 2*c^2))
```

3.13 $\int \frac{\sec^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 324

$$\int \frac{\sec^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\sqrt{2}bc\left(1+\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{(a-b+c)(a+b+c)\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{\sqrt{2}bc\left(1-\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{(a-b+c)(a+b+c)\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} + \frac{\cos(x)}{2(a+b+c)(1-\sin(x))} - \frac{\cos(x)}{2(a-b+c)(1+\sin(x))}$$

```
[Out] 1/2*cos(x)/(a+b+c)/(1-sin(x))-1/2*cos(x)/(a-b+c)/(1+sin(x))-b*c*arctan(1/2*
(2*c+(b-(-4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)-b*(-4*a*c+b^
2)^(1/2))^(1/2))*2^(1/2)*(1+(b^2-2*c*(a+c))/b/(-4*a*c+b^2)^(1/2))/(a-b+c)/(
a+b+c)/(b^2-2*c*(a+c)-b*(-4*a*c+b^2)^(1/2))^(1/2)-b*c*arctan(1/2*(2*c+(b+(-
4*a*c+b^2)^(1/2))*tan(1/2*x))*2^(1/2)/(b^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^
(1/2))*2^(1/2)*(1+(-b^2+2*c*(a+c))/b/(-4*a*c+b^2)^(1/2))/(a-b+c)/(a+b+c)/(b
^2-2*c*(a+c)+b*(-4*a*c+b^2)^(1/2))^(1/2)
```

Rubi [A] (verified)

Time = 2.26 (sec) , antiderivative size = 324, normalized size of antiderivative = 1.00, number of steps used = 11, number of rules used = 6, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.316, Rules used = {3347, 2727, 3373, 2739, 632, 210}

$$\int \frac{\sec^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\sqrt{2}bc\left(\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}+1\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(b-\sqrt{b^2-4ac}\right)+2c}{\sqrt{2}\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{(a-b+c)(a+b+c)\sqrt{-b\sqrt{b^2-4ac}-2c(a+c)+b^2}} - \frac{\sqrt{2}bc\left(1-\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}\right)\arctan\left(\frac{\tan\left(\frac{x}{2}\right)\left(\sqrt{b^2-4ac}+b\right)+2c}{\sqrt{2}\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}}\right)}{(a-b+c)(a+b+c)\sqrt{b\sqrt{b^2-4ac}-2c(a+c)+b^2}} + \frac{\cos(x)}{2(1-\sin(x))(a+b+c)} - \frac{\cos(x)}{2(\sin(x)+1)(a-b+c)}$$

```
[In] Int[Sec[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -((Sqrt[2]*b*c*(1 + (b^2 - 2*c*(a + c))/(b*Sqrt[b^2 - 4*a*c]))*ArcTan[(2*c
+ (b - Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - b*Sqr
t[b^2 - 4*a*c]])])/((a - b + c)*(a + b + c)*Sqrt[b^2 - 2*c*(a + c) - b*Sqrt
[b^2 - 4*a*c]])) - (Sqrt[2]*b*c*(1 - (b^2 - 2*c*(a + c))/(b*Sqrt[b^2 - 4*a*
c]))*ArcTan[(2*c + (b + Sqrt[b^2 - 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*
c*(a + c) + b*Sqrt[b^2 - 4*a*c]])])/((a - b + c)*(a + b + c)*Sqrt[b^2 - 2*c
*(a + c) + b*Sqrt[b^2 - 4*a*c]]) + Cos[x]/(2*(a + b + c)*(1 - Sin[x])) - Co
s[x]/(2*(a - b + c)*(1 + Sin[x]))
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 2727

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> Simp[-Cos[c +
 d*x]/(d*(b + a*Sin[c + d*x])), x] /; FreeQ[{a, b, c, d}, x] && EqQ[a^2 - b
^2, 0]
```

Rule 2739

```
Int[((a_) + (b_.)*sin[(c_.) + (d_.)*(x_)])^(-1), x_Symbol] :> With[{e = Fre
eFactors[Tan[(c + d*x)/2], x]}, Dist[2*(e/d), Subst[Int[1/(a + 2*b*e*x + a*
e^2*x^2), x], x, Tan[(c + d*x)/2]/e], x]] /; FreeQ[{a, b, c, d}, x] && NeQ[
a^2 - b^2, 0]
```

Rule 3347

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*sin[(d_.) + (e_.)*(x_)]^(n
_.) + (c_.)*sin[(d_.) + (e_.)*(x_)]^(n2_.))^(p_.), x_Symbol] :> Int[ExpandT
rig[(1 - sin[d + e*x]^2)^(m/2)*(a + b*sin[d + e*x]^n + c*sin[d + e*x]^(2*n)
)^p, x], x] /; FreeQ[{a, b, c, d, e}, x] && EqQ[n2, 2*n] && IntegerQ[m/2] &
& NeQ[b^2 - 4*a*c, 0] && IntegersQ[n, p]
```

Rule 3373

```
Int[((A_) + (B_.)*sin[(d_.) + (e_.)*(x_)])/((a_.) + (b_.)*sin[(d_.) + (e_.)
*(x_)] + (c_.)*sin[(d_.) + (e_.)*(x_)]^2), x_Symbol] :> Module[{q = Rt[b^2
- 4*a*c, 2]}, Dist[B + (b*B - 2*A*c)/q, Int[1/(b + q + 2*c*Sin[d + e*x]), x
], x] + Dist[B - (b*B - 2*A*c)/q, Int[1/(b - q + 2*c*Sin[d + e*x]), x], x]]
 /; FreeQ[{a, b, c, d, e, A, B}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rubi steps

$$
\begin{aligned}
\text{integral} &= \int \left(-\frac{1}{2(a+b+c)(-1+\sin(x))} + \frac{1}{2(a-b+c)(1+\sin(x))} + \frac{-b^2\left(1-\frac{c(a+c)}{b^2}\right)-bc\sin(x)}{(a-b+c)(a+b+c)\left(a+b\sin(x)+c\sin^2(x)\right)} \right) dx \\
&= \frac{\int \frac{1}{1+\sin(x)}\,dx}{2(a-b+c)} - \frac{\int \frac{1}{-1+\sin(x)}\,dx}{2(a+b+c)} + \frac{\int \frac{-b^2\left(1-\frac{c(a+c)}{b^2}\right)-bc\sin(x)}{a+b\sin(x)+c\sin^2(x)}\,dx}{(a-b+c)(a+b+c)} \\
&= \frac{\cos(x)}{2(a+b+c)(1-\sin(x))} - \frac{\cos(x)}{2(a-b+c)(1+\sin(x))} \\
&\quad - \frac{\left(c\left(b+\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\right)\int \frac{1}{b-\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{(a-b+c)(a+b+c)} \\
&\quad - \frac{\left(bc\left(1-\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}\right)\right)\int \frac{1}{b+\sqrt{b^2-4ac}+2c\sin(x)}\,dx}{(a-b+c)(a+b+c)}
\end{aligned}
$$

$$= \frac{\cos(x)}{2(a+b+c)(1-\sin(x))} - \frac{\cos(x)}{2(a-b+c)(1+\sin(x))} - \frac{\left(2c\left(b+\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b-\sqrt{b^2-4ac}+4cx+\left(b-\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{(a-b+c)(a+b+c)} - \frac{\left(2bc\left(1-\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{b+\sqrt{b^2-4ac}+4cx+\left(b+\sqrt{b^2-4ac}\right)x^2}\,dx, x, \tan\left(\frac{x}{2}\right)\right)}{(a-b+c)(a+b+c)}$$

$$= \frac{\cos(x)}{2(a+b+c)(1-\sin(x))} - \frac{\cos(x)}{2(a-b+c)(1+\sin(x))} + \frac{\left(4c\left(b+\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{-8\left(b^2-2c(a+c)-b\sqrt{b^2-4ac}\right)-x^2}\,dx, x, 4c+2\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{(a-b+c)(a+b+c)} + \frac{\left(4bc\left(1-\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}\right)\right)\text{Subst}\left(\int \frac{1}{4\left(4c^2-\left(b+\sqrt{b^2-4ac}\right)^2\right)-x^2}\,dx, x, 4c+2\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)\right)}{(a-b+c)(a+b+c)}$$

$$= -\frac{\sqrt{2}c\left(b+\frac{b^2-2c(a+c)}{\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b-\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}}\right)}{(a-b+c)(a+b+c)\sqrt{b^2-2c(a+c)-b\sqrt{b^2-4ac}}} - \frac{\sqrt{2}bc\left(1-\frac{b^2-2c(a+c)}{b\sqrt{b^2-4ac}}\right)\arctan\left(\frac{2c+\left(b+\sqrt{b^2-4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}}\right)}{(a-b+c)(a+b+c)\sqrt{b^2-2c(a+c)+b\sqrt{b^2-4ac}}} + \frac{\cos(x)}{2(a+b+c)(1-\sin(x))} - \frac{\cos(x)}{2(a-b+c)(1+\sin(x))}$$

Mathematica [C] (verified)

Result contains complex when optimal does not.

Time = 1.20 (sec) , antiderivative size = 407, normalized size of antiderivative = 1.26

$$\int \frac{\sec^2(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$
$$= -\frac{c\left(-ib^2+2ic(a+c)+b\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b-i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\,(a^2-b^2+2ac+c^2)\sqrt{b^2-2c(a+c)-ib\sqrt{-b^2+4ac}}}$$
$$-\frac{c\left(ib^2-2ic(a+c)+b\sqrt{-b^2+4ac}\right)\arctan\left(\frac{2c+\left(b+i\sqrt{-b^2+4ac}\right)\tan\left(\frac{x}{2}\right)}{\sqrt{2}\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}\right)}{\sqrt{-\frac{b^2}{2}+2ac}\,(a^2-b^2+2ac+c^2)\sqrt{b^2-2c(a+c)+ib\sqrt{-b^2+4ac}}}$$
$$+\frac{\sin\left(\frac{x}{2}\right)}{(a+b+c)\left(\cos\left(\frac{x}{2}\right)-\sin\left(\frac{x}{2}\right)\right)}+\frac{\sin\left(\frac{x}{2}\right)}{(a-b+c)\left(\cos\left(\frac{x}{2}\right)+\sin\left(\frac{x}{2}\right)\right)}$$

```
[In] Integrate[Sec[x]^2/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -((c*((-I)*b^2 + (2*I)*c*(a + c) + b*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b -
 I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) - I*b*Sqrt
[-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^2 + 2*a*c]*(a^2 - b^2 + 2*a*c + c^2)*Sqrt[b
^2 - 2*c*(a + c) - I*b*Sqrt[-b^2 + 4*a*c]])) - (c*(I*b^2 - (2*I)*c*(a + c)
+ b*Sqrt[-b^2 + 4*a*c])*ArcTan[(2*c + (b + I*Sqrt[-b^2 + 4*a*c])*Tan[x/2])/
(Sqrt[2]*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2 + 4*a*c]])])/(Sqrt[-1/2*b^2
 + 2*a*c]*(a^2 - b^2 + 2*a*c + c^2)*Sqrt[b^2 - 2*c*(a + c) + I*b*Sqrt[-b^2
+ 4*a*c]]) + Sin[x/2]/((a + b + c)*(Cos[x/2] - Sin[x/2])) + Sin[x/2]/((a -
b + c)*(Cos[x/2] + Sin[x/2]))
```

Maple [A] (verified)

Time = 7.06 (sec) , antiderivative size = 417, normalized size of antiderivative = 1.29

method	result
default	$\frac{2a\left(\frac{\left(-3\sqrt{-4ac+b^2}\,abc+\sqrt{-4ac+b^2}\,b^3-\sqrt{-4ac+b^2}\,b\,c^2+4a^2c^2-5a\,b^2c+4a\,c^3+b^4-b^2c^2\right)\arctan\left(\frac{2a\tan\left(\frac{x}{2}\right)+b+\sqrt{-4ac+b^2}}{\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}\right)}{a\left(4ac-b^2\right)\sqrt{4ac-2b^2-2b\sqrt{-4ac+b^2}+4a^2}}-\frac{\left(3\sqrt{-4ac+b\right.}{}}{}\right.}{(a-b+c)(a+b+c)}$
risch	Expression too large to display

```
[In] int(sec(x)^2/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 2/(a-b+c)/(a+b+c)*a*((-3*(-4*a*c+b^2)^(1/2)*a*b*c+(-4*a*c+b^2)^(1/2)*b^3-(-
4*a*c+b^2)^(1/2)*b*c^2+4*a^2*c^2-5*a*b^2*c+4*a*c^3+b^4-b^2*c^2)/a/(4*a*c-b^
2)/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((2*a*tan(1/2*x)+
b+(-4*a*c+b^2)^(1/2))/(4*a*c-2*b^2-2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2))-(3*
(-4*a*c+b^2)^(1/2)*a*b*c-(-4*a*c+b^2)^(1/2)*b^3+(-4*a*c+b^2)^(1/2)*b*c^2+4*
a^2*c^2-5*a*b^2*c+4*a*c^3+b^4-b^2*c^2)/a/(4*a*c-b^2)/(4*a*c-2*b^2+2*b*(-4*a
*c+b^2)^(1/2)+4*a^2)^(1/2)*arctan((-2*a*tan(1/2*x)+(-4*a*c+b^2)^(1/2)-b)/(4
*a*c-2*b^2+2*b*(-4*a*c+b^2)^(1/2)+4*a^2)^(1/2)))-2/(2*a+2*b+2*c)/(tan(1/2*x
)-1)-2/(2*a-2*b+2*c)/(tan(1/2*x)+1)
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 16739 vs. $2(282) = 564$.

Time = 3.95 (sec) , antiderivative size = 16739, normalized size of antiderivative = 51.66

$$\int \frac{\sec^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(sec(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] Too large to include
```

Sympy [F]

$$\int \frac{\sec^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \int \frac{\sec^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx$$

```
[In] integrate(sec(x)**2/(a+b*sin(x)+c*sin(x)**2),x)
```

```
[Out] Integral(sec(x)**2/(a + b*sin(x) + c*sin(x)**2), x)
```

Maxima [F]

$$\int \frac{\sec^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \int \frac{\sec(x)^2}{c\sin(x)^2 + b\sin(x) + a}\,dx$$

```
[In] integrate(sec(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
```

```
[Out] -(2*b*cos(2*x)*cos(x) + 2*b*cos(x) + ((a^2 - b^2 + 2*a*c + c^2)*cos(2*x)^2
+ (a^2 - b^2 + 2*a*c + c^2)*sin(2*x)^2 + a^2 - b^2 + 2*a*c + c^2 + 2*(a^2 -
 b^2 + 2*a*c + c^2)*cos(2*x))*integrate(2*(2*b^2*c*cos(3*x)^2 + 2*b^2*c*cos
(x)^2 + 2*b^2*c*sin(3*x)^2 + 2*b^2*c*sin(x)^2 + b*c^2*sin(x) + 4*(2*a*b^2 -
```

```
3*a*c^2 - c^3 - (2*a^2 - b^2)*c)*cos(2*x)^2 + 2*(2*b^3 - b*c^2)*cos(x)*sin
(2*x) + 4*(2*a*b^2 - 3*a*c^2 - c^3 - (2*a^2 - b^2)*c)*sin(2*x)^2 - (b*c^2*s
in(3*x) - b*c^2*sin(x) + 2*(b^2*c - a*c^2 - c^3)*cos(2*x))*cos(4*x) - 2*(2*
b^2*c*cos(x) + (2*b^3 - b*c^2)*sin(2*x))*cos(3*x) - 2*(b^2*c - a*c^2 - c^3
+ (2*b^3 - b*c^2)*sin(x))*cos(2*x) + (b*c^2*cos(3*x) - b*c^2*cos(x) - 2*(b^
2*c - a*c^2 - c^3)*sin(2*x))*sin(4*x) - (4*b^2*c*sin(x) + b*c^2 - 2*(2*b^3
- b*c^2)*cos(2*x))*sin(3*x))/(2*a*c^3 + c^4 + (a^2 - b^2)*c^2 + (2*a*c^3 +
c^4 + (a^2 - b^2)*c^2)*cos(4*x)^2 + 4*(a^2*b^2 - b^4 + 2*a*b^2*c + b^2*c^2)
*cos(3*x)^2 + 4*(4*a^4 - 4*a^2*b^2 + 6*a*c^3 + c^4 + (13*a^2 - b^2)*c^2 + 4
*(3*a^3 - a*b^2)*c)*cos(2*x)^2 + 4*(a^2*b^2 - b^4 + 2*a*b^2*c + b^2*c^2)*co
s(x)^2 + (2*a*c^3 + c^4 + (a^2 - b^2)*c^2)*sin(4*x)^2 + 4*(a^2*b^2 - b^4 +
2*a*b^2*c + b^2*c^2)*sin(3*x)^2 + 8*(2*a^3*b - 2*a*b^3 + 4*a*b*c^2 + b*c^3
+ (5*a^2*b - b^3)*c)*cos(x)*sin(2*x) + 4*(4*a^4 - 4*a^2*b^2 + 6*a*c^3 + c^4
 + (13*a^2 - b^2)*c^2 + 4*(3*a^3 - a*b^2)*c)*sin(2*x)^2 + 4*(a^2*b^2 - b^4
+ 2*a*b^2*c + b^2*c^2)*sin(x)^2 + 2*(2*a*c^3 + c^4 + (a^2 - b^2)*c^2 - 2*(4
*a*c^3 + c^4 + (5*a^2 - b^2)*c^2 + 2*(a^3 - a*b^2)*c)*cos(2*x) - 2*(2*a*b*c
^2 + b*c^3 + (a^2*b - b^3)*c)*sin(3*x) + 2*(2*a*b*c^2 + b*c^3 + (a^2*b - b^
3)*c)*sin(x))*cos(4*x) - 8*((a^2*b^2 - b^4 + 2*a*b^2*c + b^2*c^2)*cos(x) +
(2*a^3*b - 2*a*b^3 + 4*a*b*c^2 + b*c^3 + (5*a^2*b - b^3)*c)*sin(2*x))*cos(3
*x) - 4*(4*a*c^3 + c^4 + (5*a^2 - b^2)*c^2 + 2*(a^3 - a*b^2)*c + 2*(2*a^3*b
 - 2*a*b^3 + 4*a*b*c^2 + b*c^3 + (5*a^2*b - b^3)*c)*sin(x))*cos(2*x) + 4*((
2*a*b*c^2 + b*c^3 + (a^2*b - b^3)*c)*cos(3*x) - (2*a*b*c^2 + b*c^3 + (a^2*b
 - b^3)*c)*cos(x) - (4*a*c^3 + c^4 + (5*a^2 - b^2)*c^2 + 2*(a^3 - a*b^2)*c)
*sin(2*x))*sin(4*x) - 4*(2*a*b*c^2 + b*c^3 + (a^2*b - b^3)*c - 2*(2*a^3*b -
 2*a*b^3 + 4*a*b*c^2 + b*c^3 + (5*a^2*b - b^3)*c)*cos(2*x) + 2*(a^2*b^2 - b
^4 + 2*a*b^2*c + b^2*c^2)*sin(x))*sin(3*x) + 4*(2*a*b*c^2 + b*c^3 + (a^2*b
- b^3)*c)*sin(x)), x) + 2*(b*sin(x) - a - c)*sin(2*x))/((a^2 - b^2 + 2*a*c
+ c^2)*cos(2*x)^2 + (a^2 - b^2 + 2*a*c + c^2)*sin(2*x)^2 + a^2 - b^2 + 2*a*
c + c^2 + 2*(a^2 - b^2 + 2*a*c + c^2)*cos(2*x))
```

Giac [F(-1)]

Timed out.

$$\int \frac{\sec^2(x)}{a + b\sin(x) + c\sin^2(x)}\, dx = \text{Timed out}$$

```
[In] integrate(sec(x)^2/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] Timed out
```

Mupad [B] (verification not implemented)

Time = 28.27 (sec) , antiderivative size = 37118, normalized size of antiderivative = 114.56

$$\int \frac{\sec^2(x)}{a + b\sin(x) + c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(1/(cos(x)^2*(a + c*sin(x)^2 + b*sin(x))),x)
```

```
[Out] atan(((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c
 - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*
b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^
2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^
2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a
*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4
 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a
^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*
b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2
*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4
*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2
 + 14*a*b^8*c)))^(1/2)*((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*
c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18
*a*b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c
^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) -
 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b
^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3
*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 +
240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8
*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*
a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2
*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^
3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(64*a*b^13 - 256*a^3*b^1
1 + 384*a^5*b^9 - 256*a^7*b^7 + 64*a^9*b^5 - 128*a*b^3*c^10 + 576*a*b^5*c^8
 - 1024*a*b^7*c^6 + 896*a*b^9*c^4 - 384*a*b^11*c^2 + 512*a^2*b*c^11 - 896*a
^2*b^11*c + 4608*a^3*b*c^10 + 18432*a^4*b*c^9 + 3072*a^4*b^9*c + 43008*a^5*
b*c^8 + 64512*a^6*b*c^7 - 3840*a^6*b^7*c + 64512*a^7*b*c^6 + 43008*a^8*b*c^
5 + 2048*a^8*b^5*c + 18432*a^9*b*c^4 + 4608*a^10*b*c^3 - 384*a^10*b^3*c + 5
12*a^11*b*c^2 - 3456*a^2*b^3*c^9 + 8192*a^2*b^5*c^7 - 8960*a^2*b^7*c^5 + 46
08*a^2*b^9*c^3 - 20992*a^3*b^3*c^8 + 34048*a^3*b^5*c^6 - 23808*a^3*b^7*c^4
+ 6400*a^3*b^9*c^2 - 60928*a^4*b^3*c^7 + 67584*a^4*b^5*c^5 - 28160*a^4*b^7*
c^3 - 102144*a^5*b^3*c^6 + 73600*a^5*b^5*c^4 - 15872*a^5*b^7*c^2 - 105728*a
^6*b^3*c^5 + 45056*a^6*b^5*c^3 - 68096*a^7*b^3*c^4 + 14592*a^7*b^5*c^2 - 26
112*a^8*b^3*c^3 - 5248*a^9*b^3*c^2) + (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^
3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 -
3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2)
```

```
- 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b
^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^
3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*
b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 +
320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 +
 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*
a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2
 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 -
312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(256*a^14*
c - 96*a*b^14 + 544*a^3*b^12 - 1280*a^5*b^10 + 1600*a^7*b^8 - 1120*a^9*b^6
+ 416*a^11*b^4 - 64*a^13*b^2 + 512*a^2*c^13 + 5888*a^3*c^12 + 30976*a^4*c^1
1 + 98560*a^5*c^10 + 211200*a^6*c^9 + 321024*a^7*c^8 + 354816*a^8*c^7 + 287
232*a^9*c^6 + 168960*a^10*c^5 + 70400*a^11*c^4 + 19712*a^12*c^3 + 3328*a^13
*c^2 - 128*a*b^2*c^12 + 736*a*b^4*c^10 - 1760*a*b^6*c^8 + 2240*a*b^8*c^6 -
1600*a*b^10*c^4 + 608*a*b^12*c^2 + 1536*a^2*b^12*c - 7616*a^4*b^10*c + 1536
0*a^6*b^8*c - 16000*a^8*b^6*c + 8960*a^10*b^4*c - 2496*a^12*b^2*c - 4416*a^
2*b^2*c^11 + 14080*a^2*b^4*c^9 - 22400*a^2*b^6*c^7 + 19200*a^2*b^8*c^5 - 85
12*a^2*b^10*c^3 - 35904*a^3*b^2*c^10 + 84000*a^3*b^4*c^8 - 96000*a^3*b^6*c^
6 + 54720*a^3*b^8*c^4 - 13248*a^3*b^10*c^2 - 145600*a^4*b^2*c^9 + 256000*a^
4*b^4*c^7 - 206720*a^4*b^6*c^5 + 72960*a^4*b^8*c^3 - 360000*a^5*b^2*c^8 + 4
68160*a^5*b^4*c^6 - 254400*a^5*b^6*c^4 + 48960*a^5*b^8*c^2 - 590976*a^6*b^2
*c^7 + 548352*a^6*b^4*c^5 - 184960*a^6*b^6*c^3 - 669312*a^7*b^2*c^6 + 41888
0*a^7*b^4*c^4 - 76800*a^7*b^6*c^2 - 528768*a^8*b^2*c^5 + 204800*a^8*b^4*c^3
 - 288000*a^9*b^2*c^4 + 60000*a^9*b^4*c^2 - 104000*a^10*b^2*c^3 - 22848*a^1
1*b^2*c^2) - 32*a^2*b^13 + 160*a^4*b^11 - 320*a^6*b^9 + 320*a^8*b^7 - 160*a
^10*b^5 + 32*a^12*b^3 - 32*a*b^3*c^11 + 160*a*b^5*c^9 - 320*a*b^7*c^7 + 320
*a*b^9*c^5 - 160*a*b^11*c^3 + 128*a^2*b*c^12 + 1152*a^3*b*c^11 + 288*a^3*b^
11*c + 4480*a^4*b*c^10 + 9600*a^5*b*c^9 - 1600*a^5*b^9*c + 11520*a^6*b*c^8
+ 5376*a^7*b*c^7 + 2880*a^7*b^7*c - 5376*a^8*b*c^6 - 11520*a^9*b*c^5 - 2400
*a^9*b^5*c - 9600*a^10*b*c^4 - 4480*a^11*b*c^3 + 928*a^11*b^3*c - 1152*a^12
*b*c^2 - 928*a^2*b^3*c^10 + 2400*a^2*b^5*c^8 - 2880*a^2*b^7*c^6 + 1600*a^2*
b^9*c^4 - 288*a^2*b^11*c^2 - 5600*a^3*b^3*c^9 + 9600*a^3*b^5*c^7 - 6720*a^3
*b^7*c^5 + 1280*a^3*b^9*c^3 - 15200*a^4*b^3*c^8 + 16000*a^4*b^5*c^6 - 4160*
a^4*b^7*c^4 - 1280*a^4*b^9*c^2 - 20800*a^5*b^3*c^7 + 8640*a^5*b^5*c^5 + 416
0*a^5*b^7*c^3 - 10304*a^6*b^3*c^6 - 8640*a^6*b^5*c^4 + 6720*a^6*b^7*c^2 + 1
0304*a^7*b^3*c^5 - 16000*a^7*b^5*c^3 + 20800*a^8*b^3*c^4 - 9600*a^8*b^5*c^2
 + 15200*a^9*b^3*c^3 + 5600*a^10*b^3*c^2 + 32*a*b^13*c - 128*a^13*b*c) + 32
*a^2*b^12 - 128*a^4*b^10 + 192*a^6*b^8 - 128*a^8*b^6 + 32*a^10*b^4 + 128*a^
2*c^12 + 1280*a^3*c^11 + 5760*a^4*c^10 + 15360*a^5*c^9 + 26880*a^6*c^8 + 32
256*a^7*c^7 + 26880*a^8*c^6 + 15360*a^9*c^5 + 5760*a^10*c^4 + 1280*a^11*c^3
 + 128*a^12*c^2 - 32*a*b^2*c^11 + 128*a*b^4*c^9 - 192*a*b^6*c^7 + 128*a*b^8
*c^5 - 32*a*b^10*c^3 - 416*a^3*b^10*c + 1408*a^5*b^8*c - 1728*a^7*b^6*c + 8
96*a^9*b^4*c - 160*a^11*b^2*c - 832*a^2*b^2*c^10 + 1824*a^2*b^4*c^8 - 1792*
a^2*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^10*c^2 - 5664*a^3*b^2*c^9 + 8960*
a^3*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*b^8*c^3 - 19200*a^4*b^2*c^8 + 226
```

56*a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*a^4*b^8*c^2 - 38976*a^5*b^2*c^7 + 33792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 51072*a^6*b^2*c^6 + 31168*a^6*b^4*c^4 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5 + 17664*a^7*b^4*c^3 - 25344*a^8*b^2*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*c^3 - 1856*a^10*b^2*c^2) + tan(x/2)*(32*a*b^12 + 128*a*c^12 - 96*a^3*b^10 + 96*a^5*b^8 - 32*a^7*b^6 + 1088*a^2*c^11 + 4096*a^3*c^10 + 8960*a^4*c^9 + 12544*a^5*c^8 + 11648*a^6*c^7 + 7168*a^7*c^6 + 2816*a^8*c^5 + 640*a^9*c^4 + 64*a^10*c^3 - 544*a*b^2*c^10 + 992*a*b^4*c^8 - 1024*a*b^6*c^6 + 640*a*b^8*c^4 - 224*a*b^10*c^2 - 384*a^2*b^10*c + 960*a^4*b^8*c - 768*a^6*b^6*c + 192*a^8*b^4*c - 3968*a^2*b^2*c^9 + 6144*a^2*b^4*c^7 - 5120*a^2*b^6*c^5 + 2240*a^2*b^8*c^3 - 12672*a^3*b^2*c^8 + 16032*a^3*b^4*c^6 - 9760*a^3*b^6*c^4 + 2400*a^3*b^8*c^2 - 23168*a^4*b^2*c^7 + 22720*a^4*b^4*c^5 - 8960*a^4*b^6*c^3 - 26560*a^5*b^2*c^6 + 18720*a^5*b^4*c^4 - 4032*a^5*b^6*c^2 - 19584*a^6*b^2*c^5 + 8832*a^6*b^4*c^3 - 9088*a^7*b^2*c^4 + 2144*a^7*b^4*c^2 - 2432*a^8*b^2*c^3 - 288*a^9*b^2*c^2) - 160*a*b^3*c^9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 + 160*a*b^9*c^3 + 384*a^2*b*c^10 + 1792*a^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b*c^8 + 6720*a^5*b*c^7 - 96*a^5*b^7*c + 6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32*a^7*b^5*c + 1152*a^8*b*c^4 + 160*a^9*b*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b^5*c^6 - 1440*a^2*b^7*c^4 + 352*a^2*b^9*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b^5*c^5 - 1888*a^3*b^7*c^3 - 9440*a^4*b^3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b^7*c^2 - 9440*a^5*b^3*c^5 + 3072*a^5*b^5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b^5*c^2 - 1504*a^7*b^3*c^3 - 160*a^8*b^3*c^2 + 32*a*b*c^11 - 32*a*b^11*c)*1i + (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(32*a*b^12 + 128*a*c^12 - 96*a^3*b^10 + 96*a^5*b^8 - 32*a^7*b^6 + 1088*a^2*c^11 + 4096*a^3*c^10 + 8960*a^4*c^9 + 12544*a^5*c^8 + 11648*a^6*c^7 + 7168*a^7*c^6 + 2816*a^8*c^5 + 640*a^9*c^4 + 64*a^10*c^3 - 544*a*b^2*c^10 + 992*a*b^4*c^8 - 1024*a*b^6*c^6 + 640*a*b^8*c^4 - 224*a*b^10*c^2 - 384*a^2*b^10*c + 960*a^4*b^8*c - 768*a^6*b^6*c + 192*a^8*b^4*c - 3968*a^2*b^2*c^9 + 6144*a^2*b^4*c^7 - 5120*a^2*b^6*c^5 + 2240*a^2*b^8*c^3 - 12672*a^3*b^2*c^8 + 16032*a^3*b^4*c^6 - 9760*a^3*b^6*c^4 + 2400*a^3*b^8*c^2 - 23168*a^4*b^2*c^7 + 22720*a^4*b^4*c^5 - 8960*a^4*b^6*c^3 - 26560*a^5*b^2*c^6 + 18720*a^5*b^4*c^4 - 4032*a^5*b^6*c^2 - 19584*a^6*b^2*c^5 + 8832*a^6*b^4*c^3 - 9088*a^7*b^2*c^4 + 2144*a^7*b^4*c^2 - 2432*a^8*b^2*c^3 - 288*a^9*b^2*c^2) - (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a

*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b
^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^
3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3
)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c
^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^
4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 -
 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c
^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 +
 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(
tan(x/2)*(64*a*b^13 - 256*a^3*b^11 + 384*a^5*b^9 - 256*a^7*b^7 + 64*a^9*b^5
 - 128*a*b^3*c^10 + 576*a*b^5*c^8 - 1024*a*b^7*c^6 + 896*a*b^9*c^4 - 384*a*
b^11*c^2 + 512*a^2*b*c^11 - 896*a^2*b^11*c + 4608*a^3*b*c^10 + 18432*a^4*b*
c^9 + 3072*a^4*b^9*c + 43008*a^5*b*c^8 + 64512*a^6*b*c^7 - 3840*a^6*b^7*c +
 64512*a^7*b*c^6 + 43008*a^8*b*c^5 + 2048*a^8*b^5*c + 18432*a^9*b*c^4 + 460
8*a^10*b*c^3 - 384*a^10*b^3*c + 512*a^11*b*c^2 - 3456*a^2*b^3*c^9 + 8192*a^
2*b^5*c^7 - 8960*a^2*b^7*c^5 + 4608*a^2*b^9*c^3 - 20992*a^3*b^3*c^8 + 34048
*a^3*b^5*c^6 - 23808*a^3*b^7*c^4 + 6400*a^3*b^9*c^2 - 60928*a^4*b^3*c^7 + 6
7584*a^4*b^5*c^5 - 28160*a^4*b^7*c^3 - 102144*a^5*b^3*c^6 + 73600*a^5*b^5*c
^4 - 15872*a^5*b^7*c^2 - 105728*a^6*b^3*c^5 + 45056*a^6*b^5*c^3 - 68096*a^7
*b^3*c^4 + 14592*a^7*b^5*c^2 - 26112*a^8*b^3*c^3 - 5248*a^9*b^3*c^2) - (-(8
*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)
^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 +
3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3
*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-
(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(
4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*
c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 1
6*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 -
36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 +
 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448
*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^
8*c)))^(1/2)*(tan(x/2)*(256*a^14*c - 96*a*b^14 + 544*a^3*b^12 - 1280*a^5*b^
10 + 1600*a^7*b^8 - 1120*a^9*b^6 + 416*a^11*b^4 - 64*a^13*b^2 + 512*a^2*c^1
3 + 5888*a^3*c^12 + 30976*a^4*c^11 + 98560*a^5*c^10 + 211200*a^6*c^9 + 3210
24*a^7*c^8 + 354816*a^8*c^7 + 287232*a^9*c^6 + 168960*a^10*c^5 + 70400*a^11
*c^4 + 19712*a^12*c^3 + 3328*a^13*c^2 - 128*a*b^2*c^12 + 736*a*b^4*c^10 - 1
760*a*b^6*c^8 + 2240*a*b^8*c^6 - 1600*a*b^10*c^4 + 608*a*b^12*c^2 + 1536*a^
2*b^12*c - 7616*a^4*b^10*c + 15360*a^6*b^8*c - 16000*a^8*b^6*c + 8960*a^10*
b^4*c - 2496*a^12*b^2*c - 4416*a^2*b^2*c^11 + 14080*a^2*b^4*c^9 - 22400*a^2
*b^6*c^7 + 19200*a^2*b^8*c^5 - 8512*a^2*b^10*c^3 - 35904*a^3*b^2*c^10 + 840
00*a^3*b^4*c^8 - 96000*a^3*b^6*c^6 + 54720*a^3*b^8*c^4 - 13248*a^3*b^10*c^2
 - 145600*a^4*b^2*c^9 + 256000*a^4*b^4*c^7 - 206720*a^4*b^6*c^5 + 72960*a^4
*b^8*c^3 - 360000*a^5*b^2*c^8 + 468160*a^5*b^4*c^6 - 254400*a^5*b^6*c^4 + 4
8960*a^5*b^8*c^2 - 590976*a^6*b^2*c^7 + 548352*a^6*b^4*c^5 - 184960*a^6*b^6
*c^3 - 669312*a^7*b^2*c^6 + 418880*a^7*b^4*c^4 - 76800*a^7*b^6*c^2 - 528768

```
*a^8*b^2*c^5 + 204800*a^8*b^4*c^3 - 288000*a^9*b^2*c^4 + 60000*a^9*b^4*c^2
- 104000*a^10*b^2*c^3 - 22848*a^11*b^2*c^2) - 32*a^2*b^13 + 160*a^4*b^11 -
320*a^6*b^9 + 320*a^8*b^7 - 160*a^10*b^5 + 32*a^12*b^3 - 32*a*b^3*c^11 + 16
0*a*b^5*c^9 - 320*a*b^7*c^7 + 320*a*b^9*c^5 - 160*a*b^11*c^3 + 128*a^2*b*c^
12 + 1152*a^3*b*c^11 + 288*a^3*b^11*c + 4480*a^4*b*c^10 + 9600*a^5*b*c^9 -
1600*a^5*b^9*c + 11520*a^6*b*c^8 + 5376*a^7*b*c^7 + 2880*a^7*b^7*c - 5376*a
^8*b*c^6 - 11520*a^9*b*c^5 - 2400*a^9*b^5*c - 9600*a^10*b*c^4 - 4480*a^11*b
*c^3 + 928*a^11*b^3*c - 1152*a^12*b*c^2 - 928*a^2*b^3*c^10 + 2400*a^2*b^5*c
^8 - 2880*a^2*b^7*c^6 + 1600*a^2*b^9*c^4 - 288*a^2*b^11*c^2 - 5600*a^3*b^3*
c^9 + 9600*a^3*b^5*c^7 - 6720*a^3*b^7*c^5 + 1280*a^3*b^9*c^3 - 15200*a^4*b^
3*c^8 + 16000*a^4*b^5*c^6 - 4160*a^4*b^7*c^4 - 1280*a^4*b^9*c^2 - 20800*a^5
*b^3*c^7 + 8640*a^5*b^5*c^5 + 4160*a^5*b^7*c^3 - 10304*a^6*b^3*c^6 - 8640*a
^6*b^5*c^4 + 6720*a^6*b^7*c^2 + 10304*a^7*b^3*c^5 - 16000*a^7*b^5*c^3 + 208
00*a^8*b^3*c^4 - 9600*a^8*b^5*c^2 + 15200*a^9*b^3*c^3 + 5600*a^10*b^3*c^2 +
 32*a*b^13*c - 128*a^13*b*c) + 32*a^2*b^12 - 128*a^4*b^10 + 192*a^6*b^8 - 1
28*a^8*b^6 + 32*a^10*b^4 + 128*a^2*c^12 + 1280*a^3*c^11 + 5760*a^4*c^10 + 1
5360*a^5*c^9 + 26880*a^6*c^8 + 32256*a^7*c^7 + 26880*a^8*c^6 + 15360*a^9*c^
5 + 5760*a^10*c^4 + 1280*a^11*c^3 + 128*a^12*c^2 - 32*a*b^2*c^11 + 128*a*b^
4*c^9 - 192*a*b^6*c^7 + 128*a*b^8*c^5 - 32*a*b^10*c^3 - 416*a^3*b^10*c + 14
08*a^5*b^8*c - 1728*a^7*b^6*c + 896*a^9*b^4*c - 160*a^11*b^2*c - 832*a^2*b^
2*c^10 + 1824*a^2*b^4*c^8 - 1792*a^2*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^
10*c^2 - 5664*a^3*b^2*c^9 + 8960*a^3*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*
b^8*c^3 - 19200*a^4*b^2*c^8 + 22656*a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*
a^4*b^8*c^2 - 38976*a^5*b^2*c^7 + 33792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 5
1072*a^6*b^2*c^6 + 31168*a^6*b^4*c^4 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5
 + 17664*a^7*b^4*c^3 - 25344*a^8*b^2*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*
c^3 - 1856*a^10*b^2*c^2) - 160*a*b^3*c^9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 +
160*a*b^9*c^3 + 384*a^2*b*c^10 + 1792*a^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b
*c^8 + 6720*a^5*b*c^7 - 96*a^5*b^7*c + 6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32
*a^7*b^5*c + 1152*a^8*b*c^4 + 160*a^9*b*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b
^5*c^6 - 1440*a^2*b^7*c^4 + 352*a^2*b^9*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b
^5*c^5 - 1888*a^3*b^7*c^3 - 9440*a^4*b^3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b
^7*c^2 - 9440*a^5*b^3*c^5 + 3072*a^5*b^5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b
^5*c^2 - 1504*a^7*b^3*c^3 - 160*a^8*b^3*c^2 + 32*a*b*c^11 - 32*a*b^11*c)*1i
)/((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c -
b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4
*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 -
 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b
*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^
3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 +
16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*
c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4
*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^
2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^
3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 +
```

```
14*a*b^8*c)))^(1/2)*((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4
 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*
b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4
+ 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10
*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)
^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^
4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240
*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*
b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7
*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^
5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 -
 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(64*a*b^13 - 256*a^3*b^11 +
 384*a^5*b^9 - 256*a^7*b^7 + 64*a^9*b^5 - 128*a*b^3*c^10 + 576*a*b^5*c^8 -
1024*a*b^7*c^6 + 896*a*b^9*c^4 - 384*a*b^11*c^2 + 512*a^2*b*c^11 - 896*a^2*
b^11*c + 4608*a^3*b*c^10 + 18432*a^4*b*c^9 + 3072*a^4*b^9*c + 43008*a^5*b*c
^8 + 64512*a^6*b*c^7 - 3840*a^6*b^7*c + 64512*a^7*b*c^6 + 43008*a^8*b*c^5 +
 2048*a^8*b^5*c + 18432*a^9*b*c^4 + 4608*a^10*b*c^3 - 384*a^10*b^3*c + 512*
a^11*b*c^2 - 3456*a^2*b^3*c^9 + 8192*a^2*b^5*c^7 - 8960*a^2*b^7*c^5 + 4608*
a^2*b^9*c^3 - 20992*a^3*b^3*c^8 + 34048*a^3*b^5*c^6 - 23808*a^3*b^7*c^4 + 6
400*a^3*b^9*c^2 - 60928*a^4*b^3*c^7 + 67584*a^4*b^5*c^5 - 28160*a^4*b^7*c^3
 - 102144*a^5*b^3*c^6 + 73600*a^5*b^5*c^4 - 15872*a^5*b^7*c^2 - 105728*a^6*
b^3*c^5 + 45056*a^6*b^5*c^3 - 68096*a^7*b^3*c^4 + 14592*a^7*b^5*c^2 - 26112
*a^8*b^3*c^3 - 5248*a^9*b^3*c^2) + (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c
^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b
^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 5
4*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)
^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(
-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8
 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320
*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*
b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5
*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 -
312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312
*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(256*a^14*c -
 96*a*b^14 + 544*a^3*b^12 - 1280*a^5*b^10 + 1600*a^7*b^8 - 1120*a^9*b^6 + 4
16*a^11*b^4 - 64*a^13*b^2 + 512*a^2*c^13 + 5888*a^3*c^12 + 30976*a^4*c^11 +
 98560*a^5*c^10 + 211200*a^6*c^9 + 321024*a^7*c^8 + 354816*a^8*c^7 + 287232
*a^9*c^6 + 168960*a^10*c^5 + 70400*a^11*c^4 + 19712*a^12*c^3 + 3328*a^13*c^
2 - 128*a*b^2*c^12 + 736*a*b^4*c^10 - 1760*a*b^6*c^8 + 2240*a*b^8*c^6 - 160
0*a*b^10*c^4 + 608*a*b^12*c^2 + 1536*a^2*b^12*c - 7616*a^4*b^10*c + 15360*a
^6*b^8*c - 16000*a^8*b^6*c + 8960*a^10*b^4*c - 2496*a^12*b^2*c - 4416*a^2*b
^2*c^11 + 14080*a^2*b^4*c^9 - 22400*a^2*b^6*c^7 + 19200*a^2*b^8*c^5 - 8512*
a^2*b^10*c^3 - 35904*a^3*b^2*c^10 + 84000*a^3*b^4*c^8 - 96000*a^3*b^6*c^6 +
 54720*a^3*b^8*c^4 - 13248*a^3*b^10*c^2 - 145600*a^4*b^2*c^9 + 256000*a^4*b
^4*c^7 - 206720*a^4*b^6*c^5 + 72960*a^4*b^8*c^3 - 360000*a^5*b^2*c^8 + 4681
```

60*a^5*b^4*c^6 - 254400*a^5*b^6*c^4 + 48960*a^5*b^8*c^2 - 590976*a^6*b^2*c^
7 + 548352*a^6*b^4*c^5 - 184960*a^6*b^6*c^3 - 669312*a^7*b^2*c^6 + 418880*a
^7*b^4*c^4 - 76800*a^7*b^6*c^2 - 528768*a^8*b^2*c^5 + 204800*a^8*b^4*c^3 -
288000*a^9*b^2*c^4 + 60000*a^9*b^4*c^2 - 104000*a^10*b^2*c^3 - 22848*a^11*b
^2*c^2) - 32*a^2*b^13 + 160*a^4*b^11 - 320*a^6*b^9 + 320*a^8*b^7 - 160*a^10
*b^5 + 32*a^12*b^3 - 32*a*b^3*c^11 + 160*a*b^5*c^9 - 320*a*b^7*c^7 + 320*a*
b^9*c^5 - 160*a*b^11*c^3 + 128*a^2*b*c^12 + 1152*a^3*b*c^11 + 288*a^3*b^11*
c + 4480*a^4*b*c^10 + 9600*a^5*b*c^9 - 1600*a^5*b^9*c + 11520*a^6*b*c^8 + 5
376*a^7*b*c^7 + 2880*a^7*b^7*c - 5376*a^8*b*c^6 - 11520*a^9*b*c^5 - 2400*a^
9*b^5*c - 9600*a^10*b*c^4 - 4480*a^11*b*c^3 + 928*a^11*b^3*c - 1152*a^12*b*
c^2 - 928*a^2*b^3*c^10 + 2400*a^2*b^5*c^8 - 2880*a^2*b^7*c^6 + 1600*a^2*b^9
*c^4 - 288*a^2*b^11*c^2 - 5600*a^3*b^3*c^9 + 9600*a^3*b^5*c^7 - 6720*a^3*b^
7*c^5 + 1280*a^3*b^9*c^3 - 15200*a^4*b^3*c^8 + 16000*a^4*b^5*c^6 - 4160*a^4
*b^7*c^4 - 1280*a^4*b^9*c^2 - 20800*a^5*b^3*c^7 + 8640*a^5*b^5*c^5 + 4160*a
^5*b^7*c^3 - 10304*a^6*b^3*c^6 - 8640*a^6*b^5*c^4 + 6720*a^6*b^7*c^2 + 1030
4*a^7*b^3*c^5 - 16000*a^7*b^5*c^3 + 20800*a^8*b^3*c^4 - 9600*a^8*b^5*c^2 +
15200*a^9*b^3*c^3 + 5600*a^10*b^3*c^2 + 32*a*b^13*c - 128*a^13*b*c) + 32*a^
2*b^12 - 128*a^4*b^10 + 192*a^6*b^8 - 128*a^8*b^6 + 32*a^10*b^4 + 128*a^2*c
^12 + 1280*a^3*c^11 + 5760*a^4*c^10 + 15360*a^5*c^9 + 26880*a^6*c^8 + 32256
*a^7*c^7 + 26880*a^8*c^6 + 15360*a^9*c^5 + 5760*a^10*c^4 + 1280*a^11*c^3 +
128*a^12*c^2 - 32*a*b^2*c^11 + 128*a*b^4*c^9 - 192*a*b^6*c^7 + 128*a*b^8*c^
5 - 32*a*b^10*c^3 - 416*a^3*b^10*c + 1408*a^5*b^8*c - 1728*a^7*b^6*c + 896*
a^9*b^4*c - 160*a^11*b^2*c - 832*a^2*b^2*c^10 + 1824*a^2*b^4*c^8 - 1792*a^2
*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^10*c^2 - 5664*a^3*b^2*c^9 + 8960*a^3
*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*b^8*c^3 - 19200*a^4*b^2*c^8 + 22656*
a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*a^4*b^8*c^2 - 38976*a^5*b^2*c^7 + 33
792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 51072*a^6*b^2*c^6 + 31168*a^6*b^4*c^4
 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5 + 17664*a^7*b^4*c^3 - 25344*a^8*b^2
*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*c^3 - 1856*a^10*b^2*c^2) + tan(x/2)*
(32*a*b^12 + 128*a*c^12 - 96*a^3*b^10 + 96*a^5*b^8 - 32*a^7*b^6 + 1088*a^2*
c^11 + 4096*a^3*c^10 + 8960*a^4*c^9 + 12544*a^5*c^8 + 11648*a^6*c^7 + 7168*
a^7*c^6 + 2816*a^8*c^5 + 640*a^9*c^4 + 64*a^10*c^3 - 544*a*b^2*c^10 + 992*a
*b^4*c^8 - 1024*a*b^6*c^6 + 640*a*b^8*c^4 - 224*a*b^10*c^2 - 384*a^2*b^10*c
 + 960*a^4*b^8*c - 768*a^6*b^6*c + 192*a^8*b^4*c - 3968*a^2*b^2*c^9 + 6144*
a^2*b^4*c^7 - 5120*a^2*b^6*c^5 + 2240*a^2*b^8*c^3 - 12672*a^3*b^2*c^8 + 160
32*a^3*b^4*c^6 - 9760*a^3*b^6*c^4 + 2400*a^3*b^8*c^2 - 23168*a^4*b^2*c^7 +
22720*a^4*b^4*c^5 - 8960*a^4*b^6*c^3 - 26560*a^5*b^2*c^6 + 18720*a^5*b^4*c^
4 - 4032*a^5*b^6*c^2 - 19584*a^6*b^2*c^5 + 8832*a^6*b^4*c^3 - 9088*a^7*b^2*
c^4 + 2144*a^7*b^4*c^2 - 2432*a^8*b^2*c^3 - 288*a^9*b^2*c^2) - 160*a*b^3*c^
9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 + 160*a*b^9*c^3 + 384*a^2*b*c^10 + 1792*a
^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b*c^8 + 6720*a^5*b*c^7 - 96*a^5*b^7*c +
6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32*a^7*b^5*c + 1152*a^8*b*c^4 + 160*a^9*b
*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b^5*c^6 - 1440*a^2*b^7*c^4 + 352*a^2*b^9
*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b^5*c^5 - 1888*a^3*b^7*c^3 - 9440*a^4*b^
3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b^7*c^2 - 9440*a^5*b^3*c^5 + 3072*a^5*b^

5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b^5*c^2 - 1504*a^7*b^3*c^3 - 160*a^8*b^3
*c^2 + 32*a*b*c^11 - 32*a*b^11*c) - 2*tan(x/2)*(192*a*b^5*c^6 - 192*a*b^3*c
^8 - 64*a*b^7*c^4 + 384*a^2*b*c^9 + 960*a^3*b*c^8 + 1280*a^4*b*c^7 + 960*a^
5*b*c^6 + 384*a^6*b*c^5 + 64*a^7*b*c^4 - 768*a^2*b^3*c^7 + 384*a^2*b^5*c^5
- 1152*a^3*b^3*c^6 + 192*a^3*b^5*c^4 - 768*a^4*b^3*c^5 - 192*a^5*b^3*c^4 +
64*a*b*c^10) - (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 + b^5
*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^
5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a
^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6
*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1
/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6
+ a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c
^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^
7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c
- 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 26
0*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^
6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(32*a*b^12 + 128*a*c^12 - 96*a^3*
b^10 + 96*a^5*b^8 - 32*a^7*b^6 + 1088*a^2*c^11 + 4096*a^3*c^10 + 8960*a^4*c
^9 + 12544*a^5*c^8 + 11648*a^6*c^7 + 7168*a^7*c^6 + 2816*a^8*c^5 + 640*a^9*
c^4 + 64*a^10*c^3 - 544*a*b^2*c^10 + 992*a*b^4*c^8 - 1024*a*b^6*c^6 + 640*a
*b^8*c^4 - 224*a*b^10*c^2 - 384*a^2*b^10*c + 960*a^4*b^8*c - 768*a^6*b^6*c
+ 192*a^8*b^4*c - 3968*a^2*b^2*c^9 + 6144*a^2*b^4*c^7 - 5120*a^2*b^6*c^5 +
2240*a^2*b^8*c^3 - 12672*a^3*b^2*c^8 + 16032*a^3*b^4*c^6 - 9760*a^3*b^6*c^4
+ 2400*a^3*b^8*c^2 - 23168*a^4*b^2*c^7 + 22720*a^4*b^4*c^5 - 8960*a^4*b^6*
c^3 - 26560*a^5*b^2*c^6 + 18720*a^5*b^4*c^4 - 4032*a^5*b^6*c^2 - 19584*a^6*
b^2*c^5 + 8832*a^6*b^4*c^3 - 9088*a^7*b^2*c^4 + 2144*a^7*b^4*c^2 - 2432*a^8
*b^2*c^3 - 288*a^9*b^2*c^2) - (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 +
8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^
2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2
*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(
1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a
*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^
10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*
c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c
^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*
c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a
^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*
b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(64*a*b^13 - 256*a
^3*b^11 + 384*a^5*b^9 - 256*a^7*b^7 + 64*a^9*b^5 - 128*a*b^3*c^10 + 576*a*b
^5*c^8 - 1024*a*b^7*c^6 + 896*a*b^9*c^4 - 384*a*b^11*c^2 + 512*a^2*b*c^11 -
896*a^2*b^11*c + 4608*a^3*b*c^10 + 18432*a^4*b*c^9 + 3072*a^4*b^9*c + 4300
8*a^5*b*c^8 + 64512*a^6*b*c^7 - 3840*a^6*b^7*c + 64512*a^7*b*c^6 + 43008*a^
8*b*c^5 + 2048*a^8*b^5*c + 18432*a^9*b*c^4 + 4608*a^10*b*c^3 - 384*a^10*b^3
*c + 512*a^11*b*c^2 - 3456*a^2*b^3*c^9 + 8192*a^2*b^5*c^7 - 8960*a^2*b^7*c^
5 + 4608*a^2*b^9*c^3 - 20992*a^3*b^3*c^8 + 34048*a^3*b^5*c^6 - 23808*a^3*b^

7*c^4 + 6400*a^3*b^9*c^2 - 60928*a^4*b^3*c^7 + 67584*a^4*b^5*c^5 - 28160*a^
4*b^7*c^3 - 102144*a^5*b^3*c^6 + 73600*a^5*b^5*c^4 - 15872*a^5*b^7*c^2 - 10
5728*a^6*b^3*c^5 + 45056*a^6*b^5*c^3 - 68096*a^7*b^3*c^4 + 14592*a^7*b^5*c^
2 - 26112*a^8*b^3*c^3 - 5248*a^9*b^3*c^2) - (-(8*a*c^7 + b^8 + 24*a^2*c^6 +
 24*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*
c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 + 3*b*c^4*(-(4*a*c - b^2)^3)^
(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 - 3*b^3*c^2*(-(4*a
*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) + 6*
a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(
3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*
c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6
*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c
 + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b
^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*
c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(256
*a^14*c - 96*a*b^14 + 544*a^3*b^12 - 1280*a^5*b^10 + 1600*a^7*b^8 - 1120*a^
9*b^6 + 416*a^11*b^4 - 64*a^13*b^2 + 512*a^2*c^13 + 5888*a^3*c^12 + 30976*a
^4*c^11 + 98560*a^5*c^10 + 211200*a^6*c^9 + 321024*a^7*c^8 + 354816*a^8*c^7
 + 287232*a^9*c^6 + 168960*a^10*c^5 + 70400*a^11*c^4 + 19712*a^12*c^3 + 332
8*a^13*c^2 - 128*a*b^2*c^12 + 736*a*b^4*c^10 - 1760*a*b^6*c^8 + 2240*a*b^8*
c^6 - 1600*a*b^10*c^4 + 608*a*b^12*c^2 + 1536*a^2*b^12*c - 7616*a^4*b^10*c
+ 15360*a^6*b^8*c - 16000*a^8*b^6*c + 8960*a^10*b^4*c - 2496*a^12*b^2*c - 4
416*a^2*b^2*c^11 + 14080*a^2*b^4*c^9 - 22400*a^2*b^6*c^7 + 19200*a^2*b^8*c^
5 - 8512*a^2*b^10*c^3 - 35904*a^3*b^2*c^10 + 84000*a^3*b^4*c^8 - 96000*a^3*
b^6*c^6 + 54720*a^3*b^8*c^4 - 13248*a^3*b^10*c^2 - 145600*a^4*b^2*c^9 + 256
000*a^4*b^4*c^7 - 206720*a^4*b^6*c^5 + 72960*a^4*b^8*c^3 - 360000*a^5*b^2*c
^8 + 468160*a^5*b^4*c^6 - 254400*a^5*b^6*c^4 + 48960*a^5*b^8*c^2 - 590976*a
^6*b^2*c^7 + 548352*a^6*b^4*c^5 - 184960*a^6*b^6*c^3 - 669312*a^7*b^2*c^6 +
 418880*a^7*b^4*c^4 - 76800*a^7*b^6*c^2 - 528768*a^8*b^2*c^5 + 204800*a^8*b
^4*c^3 - 288000*a^9*b^2*c^4 + 60000*a^9*b^4*c^2 - 104000*a^10*b^2*c^3 - 228
48*a^11*b^2*c^2) - 32*a^2*b^13 + 160*a^4*b^11 - 320*a^6*b^9 + 320*a^8*b^7 -
 160*a^10*b^5 + 32*a^12*b^3 - 32*a*b^3*c^11 + 160*a*b^5*c^9 - 320*a*b^7*c^7
 + 320*a*b^9*c^5 - 160*a*b^11*c^3 + 128*a^2*b*c^12 + 1152*a^3*b*c^11 + 288*
a^3*b^11*c + 4480*a^4*b*c^10 + 9600*a^5*b*c^9 - 1600*a^5*b^9*c + 11520*a^6*
b*c^8 + 5376*a^7*b*c^7 + 2880*a^7*b^7*c - 5376*a^8*b*c^6 - 11520*a^9*b*c^5
- 2400*a^9*b^5*c - 9600*a^10*b*c^4 - 4480*a^11*b*c^3 + 928*a^11*b^3*c - 115
2*a^12*b*c^2 - 928*a^2*b^3*c^10 + 2400*a^2*b^5*c^8 - 2880*a^2*b^7*c^6 + 160
0*a^2*b^9*c^4 - 288*a^2*b^11*c^2 - 5600*a^3*b^3*c^9 + 9600*a^3*b^5*c^7 - 67
20*a^3*b^7*c^5 + 1280*a^3*b^9*c^3 - 15200*a^4*b^3*c^8 + 16000*a^4*b^5*c^6 -
 4160*a^4*b^7*c^4 - 1280*a^4*b^9*c^2 - 20800*a^5*b^3*c^7 + 8640*a^5*b^5*c^5
 + 4160*a^5*b^7*c^3 - 10304*a^6*b^3*c^6 - 8640*a^6*b^5*c^4 + 6720*a^6*b^7*c
^2 + 10304*a^7*b^3*c^5 - 16000*a^7*b^5*c^3 + 20800*a^8*b^3*c^4 - 9600*a^8*b
^5*c^2 + 15200*a^9*b^3*c^3 + 5600*a^10*b^3*c^2 + 32*a*b^13*c - 128*a^13*b*c
) + 32*a^2*b^12 - 128*a^4*b^10 + 192*a^6*b^8 - 128*a^8*b^6 + 32*a^10*b^4 +
128*a^2*c^12 + 1280*a^3*c^11 + 5760*a^4*c^10 + 15360*a^5*c^9 + 26880*a^6*c^

8 + 32256*a^7*c^7 + 26880*a^8*c^6 + 15360*a^9*c^5 + 5760*a^10*c^4 + 1280*a^
11*c^3 + 128*a^12*c^2 - 32*a*b^2*c^11 + 128*a*b^4*c^9 - 192*a*b^6*c^7 + 128
*a*b^8*c^5 - 32*a*b^10*c^3 - 416*a^3*b^10*c + 1408*a^5*b^8*c - 1728*a^7*b^6
*c + 896*a^9*b^4*c - 160*a^11*b^2*c - 832*a^2*b^2*c^10 + 1824*a^2*b^4*c^8 -
1792*a^2*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^10*c^2 - 5664*a^3*b^2*c^9 +
8960*a^3*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*b^8*c^3 - 19200*a^4*b^2*c^8
+ 22656*a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*a^4*b^8*c^2 - 38976*a^5*b^2
*c^7 + 33792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 51072*a^6*b^2*c^6 + 31168*a^
6*b^4*c^4 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5 + 17664*a^7*b^4*c^3 - 2534
4*a^8*b^2*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*c^3 - 1856*a^10*b^2*c^2) -
160*a*b^3*c^9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 + 160*a*b^9*c^3 + 384*a^2*b*c
^10 + 1792*a^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b*c^8 + 6720*a^5*b*c^7 - 96*
a^5*b^7*c + 6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32*a^7*b^5*c + 1152*a^8*b*c^4
+ 160*a^9*b*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b^5*c^6 - 1440*a^2*b^7*c^4 +
352*a^2*b^9*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b^5*c^5 - 1888*a^3*b^7*c^3 -
9440*a^4*b^3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b^7*c^2 - 9440*a^5*b^3*c^5 +
3072*a^5*b^5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b^5*c^2 - 1504*a^7*b^3*c^3 -
160*a^8*b^3*c^2 + 32*a*b*c^11 - 32*a*b^11*c) + 64*a*c^11 + 448*a^2*c^10 +
1344*a^3*c^9 + 2240*a^4*c^8 + 2240*a^5*c^7 + 1344*a^6*c^6 + 448*a^7*c^5 + 6
4*a^8*c^4 - 256*a*b^2*c^9 + 384*a*b^4*c^7 - 256*a*b^6*c^5 + 64*a*b^8*c^3 -
1344*a^2*b^2*c^8 + 1344*a^2*b^4*c^6 - 448*a^2*b^6*c^4 - 2880*a^3*b^2*c^7 +
1728*a^3*b^4*c^5 - 192*a^3*b^6*c^3 - 3200*a^4*b^2*c^6 + 960*a^4*b^4*c^4 - 1
920*a^5*b^2*c^5 + 192*a^5*b^4*c^3 - 576*a^6*b^2*c^4 - 64*a^7*b^2*c^3))*(-(8
*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 + b^5*(-(4*a*c - b^2)^3)
^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 +
3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3
*b^2*c^3 - 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c + 3*a^2*b*c^2*(-
(4*a*c - b^2)^3)^(1/2) + 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) - 4*a*b^3*c*(-(
4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*
c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 1
6*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 -
36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 +
159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448
*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^
8*c)))^(1/2)*2i + atan(((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*
c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18
*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c
^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) -
10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b
^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3
*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 +
240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8
*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*
a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2
*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^

3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 2
4*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^
4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1
/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c
 - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*
b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*
a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^
6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c
^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c +
 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6
*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^
2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(64*a*
b^13 - 256*a^3*b^11 + 384*a^5*b^9 - 256*a^7*b^7 + 64*a^9*b^5 - 128*a*b^3*c^
10 + 576*a*b^5*c^8 - 1024*a*b^7*c^6 + 896*a*b^9*c^4 - 384*a*b^11*c^2 + 512*
a^2*b*c^11 - 896*a^2*b^11*c + 4608*a^3*b*c^10 + 18432*a^4*b*c^9 + 3072*a^4*
b^9*c + 43008*a^5*b*c^8 + 64512*a^6*b*c^7 - 3840*a^6*b^7*c + 64512*a^7*b*c^
6 + 43008*a^8*b*c^5 + 2048*a^8*b^5*c + 18432*a^9*b*c^4 + 4608*a^10*b*c^3 -
384*a^10*b^3*c + 512*a^11*b*c^2 - 3456*a^2*b^3*c^9 + 8192*a^2*b^5*c^7 - 896
0*a^2*b^7*c^5 + 4608*a^2*b^9*c^3 - 20992*a^3*b^3*c^8 + 34048*a^3*b^5*c^6 -
23808*a^3*b^7*c^4 + 6400*a^3*b^9*c^2 - 60928*a^4*b^3*c^7 + 67584*a^4*b^5*c^
5 - 28160*a^4*b^7*c^3 - 102144*a^5*b^3*c^6 + 73600*a^5*b^5*c^4 - 15872*a^5*
b^7*c^2 - 105728*a^6*b^3*c^5 + 45056*a^6*b^5*c^3 - 68096*a^7*b^3*c^4 + 1459
2*a^7*b^5*c^2 - 26112*a^8*b^3*c^3 - 5248*a^9*b^3*c^2) + (-(8*a*c^7 + b^8 +
24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*
c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*
c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^
3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3
)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)
^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^
7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4
*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 -
36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^
4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 +
159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(t
an(x/2)*(256*a^14*c - 96*a*b^14 + 544*a^3*b^12 - 1280*a^5*b^10 + 1600*a^7*b
^8 - 1120*a^9*b^6 + 416*a^11*b^4 - 64*a^13*b^2 + 512*a^2*c^13 + 5888*a^3*c^
12 + 30976*a^4*c^11 + 98560*a^5*c^10 + 211200*a^6*c^9 + 321024*a^7*c^8 + 35
4816*a^8*c^7 + 287232*a^9*c^6 + 168960*a^10*c^5 + 70400*a^11*c^4 + 19712*a^
12*c^3 + 3328*a^13*c^2 - 128*a*b^2*c^12 + 736*a*b^4*c^10 - 1760*a*b^6*c^8 +
 2240*a*b^8*c^6 - 1600*a*b^10*c^4 + 608*a*b^12*c^2 + 1536*a^2*b^12*c - 7616
*a^4*b^10*c + 15360*a^6*b^8*c - 16000*a^8*b^6*c + 8960*a^10*b^4*c - 2496*a^
12*b^2*c - 4416*a^2*b^2*c^11 + 14080*a^2*b^4*c^9 - 22400*a^2*b^6*c^7 + 1920
0*a^2*b^8*c^5 - 8512*a^2*b^10*c^3 - 35904*a^3*b^2*c^10 + 84000*a^3*b^4*c^8
- 96000*a^3*b^6*c^6 + 54720*a^3*b^8*c^4 - 13248*a^3*b^10*c^2 - 145600*a^4*b
^2*c^9 + 256000*a^4*b^4*c^7 - 206720*a^4*b^6*c^5 + 72960*a^4*b^8*c^3 - 3600

00*a^5*b^2*c^8 + 468160*a^5*b^4*c^6 - 254400*a^5*b^6*c^4 + 48960*a^5*b^8*c^
2 - 590976*a^6*b^2*c^7 + 548352*a^6*b^4*c^5 - 184960*a^6*b^6*c^3 - 669312*a
^7*b^2*c^6 + 418880*a^7*b^4*c^4 - 76800*a^7*b^6*c^2 - 528768*a^8*b^2*c^5 +
204800*a^8*b^4*c^3 - 288000*a^9*b^2*c^4 + 60000*a^9*b^4*c^2 - 104000*a^10*b
^2*c^3 - 22848*a^11*b^2*c^2) - 32*a^2*b^13 + 160*a^4*b^11 - 320*a^6*b^9 + 3
20*a^8*b^7 - 160*a^10*b^5 + 32*a^12*b^3 - 32*a*b^3*c^11 + 160*a*b^5*c^9 - 3
20*a*b^7*c^7 + 320*a*b^9*c^5 - 160*a*b^11*c^3 + 128*a^2*b*c^12 + 1152*a^3*b
*c^11 + 288*a^3*b^11*c + 4480*a^4*b*c^10 + 9600*a^5*b*c^9 - 1600*a^5*b^9*c
+ 11520*a^6*b*c^8 + 5376*a^7*b*c^7 + 2880*a^7*b^7*c - 5376*a^8*b*c^6 - 1152
0*a^9*b*c^5 - 2400*a^9*b^5*c - 9600*a^10*b*c^4 - 4480*a^11*b*c^3 + 928*a^11
*b^3*c - 1152*a^12*b*c^2 - 928*a^2*b^3*c^10 + 2400*a^2*b^5*c^8 - 2880*a^2*b
^7*c^6 + 1600*a^2*b^9*c^4 - 288*a^2*b^11*c^2 - 5600*a^3*b^3*c^9 + 9600*a^3*
b^5*c^7 - 6720*a^3*b^7*c^5 + 1280*a^3*b^9*c^3 - 15200*a^4*b^3*c^8 + 16000*a
^4*b^5*c^6 - 4160*a^4*b^7*c^4 - 1280*a^4*b^9*c^2 - 20800*a^5*b^3*c^7 + 8640
*a^5*b^5*c^5 + 4160*a^5*b^7*c^3 - 10304*a^6*b^3*c^6 - 8640*a^6*b^5*c^4 + 67
20*a^6*b^7*c^2 + 10304*a^7*b^3*c^5 - 16000*a^7*b^5*c^3 + 20800*a^8*b^3*c^4
- 9600*a^8*b^5*c^2 + 15200*a^9*b^3*c^3 + 5600*a^10*b^3*c^2 + 32*a*b^13*c -
128*a^13*b*c) + 32*a^2*b^12 - 128*a^4*b^10 + 192*a^6*b^8 - 128*a^8*b^6 + 32
*a^10*b^4 + 128*a^2*c^12 + 1280*a^3*c^11 + 5760*a^4*c^10 + 15360*a^5*c^9 +
26880*a^6*c^8 + 32256*a^7*c^7 + 26880*a^8*c^6 + 15360*a^9*c^5 + 5760*a^10*c
^4 + 1280*a^11*c^3 + 128*a^12*c^2 - 32*a*b^2*c^11 + 128*a*b^4*c^9 - 192*a*b
^6*c^7 + 128*a*b^8*c^5 - 32*a*b^10*c^3 - 416*a^3*b^10*c + 1408*a^5*b^8*c -
1728*a^7*b^6*c + 896*a^9*b^4*c - 160*a^11*b^2*c - 832*a^2*b^2*c^10 + 1824*a
^2*b^4*c^8 - 1792*a^2*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^10*c^2 - 5664*a
^3*b^2*c^9 + 8960*a^3*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*b^8*c^3 - 19200
*a^4*b^2*c^8 + 22656*a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*a^4*b^8*c^2 - 3
8976*a^5*b^2*c^7 + 33792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 51072*a^6*b^2*c^
6 + 31168*a^6*b^4*c^4 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5 + 17664*a^7*b^
4*c^3 - 25344*a^8*b^2*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*c^3 - 1856*a^10
*b^2*c^2) + tan(x/2)*(32*a*b^12 + 128*a*c^12 - 96*a^3*b^10 + 96*a^5*b^8 - 3
2*a^7*b^6 + 1088*a^2*c^11 + 4096*a^3*c^10 + 8960*a^4*c^9 + 12544*a^5*c^8 +
11648*a^6*c^7 + 7168*a^7*c^6 + 2816*a^8*c^5 + 640*a^9*c^4 + 64*a^10*c^3 - 5
44*a*b^2*c^10 + 992*a*b^4*c^8 - 1024*a*b^6*c^6 + 640*a*b^8*c^4 - 224*a*b^10
*c^2 - 384*a^2*b^10*c + 960*a^4*b^8*c - 768*a^6*b^6*c + 192*a^8*b^4*c - 396
8*a^2*b^2*c^9 + 6144*a^2*b^4*c^7 - 5120*a^2*b^6*c^5 + 2240*a^2*b^8*c^3 - 12
672*a^3*b^2*c^8 + 16032*a^3*b^4*c^6 - 9760*a^3*b^6*c^4 + 2400*a^3*b^8*c^2 -
23168*a^4*b^2*c^7 + 22720*a^4*b^4*c^5 - 8960*a^4*b^6*c^3 - 26560*a^5*b^2*c
^6 + 18720*a^5*b^4*c^4 - 4032*a^5*b^6*c^2 - 19584*a^6*b^2*c^5 + 8832*a^6*b^
4*c^3 - 9088*a^7*b^2*c^4 + 2144*a^7*b^4*c^2 - 2432*a^8*b^2*c^3 - 288*a^9*b^
2*c^2) - 160*a*b^3*c^9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 + 160*a*b^9*c^3 + 38
4*a^2*b*c^10 + 1792*a^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b*c^8 + 6720*a^5*b*
c^7 - 96*a^5*b^7*c + 6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32*a^7*b^5*c + 1152*
a^8*b*c^4 + 160*a^9*b*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b^5*c^6 - 1440*a^2*
b^7*c^4 + 352*a^2*b^9*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b^5*c^5 - 1888*a^3*
b^7*c^3 - 9440*a^4*b^3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b^7*c^2 - 9440*a^5*

```
b^3*c^5 + 3072*a^5*b^5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b^5*c^2 - 1504*a^7*
b^3*c^3 - 160*a^8*b^3*c^2 + 32*a*b*c^11 - 32*a*b^11*c)*1i + (-(8*a*c^7 + b^
8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*
b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(
4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 +
3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^
2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2
)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^
3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 +
 b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^
3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^
4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^
4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2
)*(tan(x/2)*(32*a*b^12 + 128*a*c^12 - 96*a^3*b^10 + 96*a^5*b^8 - 32*a^7*b^6
 + 1088*a^2*c^11 + 4096*a^3*c^10 + 8960*a^4*c^9 + 12544*a^5*c^8 + 11648*a^6
*c^7 + 7168*a^7*c^6 + 2816*a^8*c^5 + 640*a^9*c^4 + 64*a^10*c^3 - 544*a*b^2*
c^10 + 992*a*b^4*c^8 - 1024*a*b^6*c^6 + 640*a*b^8*c^4 - 224*a*b^10*c^2 - 38
4*a^2*b^10*c + 960*a^4*b^8*c - 768*a^6*b^6*c + 192*a^8*b^4*c - 3968*a^2*b^2
*c^9 + 6144*a^2*b^4*c^7 - 5120*a^2*b^6*c^5 + 2240*a^2*b^8*c^3 - 12672*a^3*b
^2*c^8 + 16032*a^3*b^4*c^6 - 9760*a^3*b^6*c^4 + 2400*a^3*b^8*c^2 - 23168*a^
4*b^2*c^7 + 22720*a^4*b^4*c^5 - 8960*a^4*b^6*c^3 - 26560*a^5*b^2*c^6 + 1872
0*a^5*b^4*c^4 - 4032*a^5*b^6*c^2 - 19584*a^6*b^2*c^5 + 8832*a^6*b^4*c^3 - 9
088*a^7*b^2*c^4 + 2144*a^7*b^4*c^2 - 2432*a^8*b^2*c^3 - 288*a^9*b^2*c^2) -
(-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2
)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^
3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38
*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^
2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c
*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*
a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3
 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^
5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c
^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 -
 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*
a*b^8*c)))^(1/2)*(tan(x/2)*(64*a*b^13 - 256*a^3*b^11 + 384*a^5*b^9 - 256*a^
7*b^7 + 64*a^9*b^5 - 128*a*b^3*c^10 + 576*a*b^5*c^8 - 1024*a*b^7*c^6 + 896*
a*b^9*c^4 - 384*a*b^11*c^2 + 512*a^2*b*c^11 - 896*a^2*b^11*c + 4608*a^3*b*c
^10 + 18432*a^4*b*c^9 + 3072*a^4*b^9*c + 43008*a^5*b*c^8 + 64512*a^6*b*c^7
 - 3840*a^6*b^7*c + 64512*a^7*b*c^6 + 43008*a^8*b*c^5 + 2048*a^8*b^5*c + 184
32*a^9*b*c^4 + 4608*a^10*b*c^3 - 384*a^10*b^3*c + 512*a^11*b*c^2 - 3456*a^2
*b^3*c^9 + 8192*a^2*b^5*c^7 - 8960*a^2*b^7*c^5 + 4608*a^2*b^9*c^3 - 20992*a
^3*b^3*c^8 + 34048*a^3*b^5*c^6 - 23808*a^3*b^7*c^4 + 6400*a^3*b^9*c^2 - 609
28*a^4*b^3*c^7 + 67584*a^4*b^5*c^5 - 28160*a^4*b^7*c^3 - 102144*a^5*b^3*c^6
 + 73600*a^5*b^5*c^4 - 15872*a^5*b^7*c^2 - 105728*a^6*b^3*c^5 + 45056*a^6*b
^5*c^3 - 68096*a^7*b^3*c^4 + 14592*a^7*b^5*c^2 - 26112*a^8*b^3*c^3 - 5248*a
```

^9*b^3*c^2) - (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*
(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5
 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^
2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*
c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/
2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 +
 a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^
4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7
 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c
- 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260
*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6
*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(256*a^14*c - 96*a*b^14 + 544*a^3*
b^12 - 1280*a^5*b^10 + 1600*a^7*b^8 - 1120*a^9*b^6 + 416*a^11*b^4 - 64*a^13
*b^2 + 512*a^2*c^13 + 5888*a^3*c^12 + 30976*a^4*c^11 + 98560*a^5*c^10 + 211
200*a^6*c^9 + 321024*a^7*c^8 + 354816*a^8*c^7 + 287232*a^9*c^6 + 168960*a^1
0*c^5 + 70400*a^11*c^4 + 19712*a^12*c^3 + 3328*a^13*c^2 - 128*a*b^2*c^12 +
736*a*b^4*c^10 - 1760*a*b^6*c^8 + 2240*a*b^8*c^6 - 1600*a*b^10*c^4 + 608*a*
b^12*c^2 + 1536*a^2*b^12*c - 7616*a^4*b^10*c + 15360*a^6*b^8*c - 16000*a^8*
b^6*c + 8960*a^10*b^4*c - 2496*a^12*b^2*c - 4416*a^2*b^2*c^11 + 14080*a^2*b
^4*c^9 - 22400*a^2*b^6*c^7 + 19200*a^2*b^8*c^5 - 8512*a^2*b^10*c^3 - 35904*
a^3*b^2*c^10 + 84000*a^3*b^4*c^8 - 96000*a^3*b^6*c^6 + 54720*a^3*b^8*c^4 -
13248*a^3*b^10*c^2 - 145600*a^4*b^2*c^9 + 256000*a^4*b^4*c^7 - 206720*a^4*b
^6*c^5 + 72960*a^4*b^8*c^3 - 360000*a^5*b^2*c^8 + 468160*a^5*b^4*c^6 - 2544
00*a^5*b^6*c^4 + 48960*a^5*b^8*c^2 - 590976*a^6*b^2*c^7 + 548352*a^6*b^4*c^
5 - 184960*a^6*b^6*c^3 - 669312*a^7*b^2*c^6 + 418880*a^7*b^4*c^4 - 76800*a^
7*b^6*c^2 - 528768*a^8*b^2*c^5 + 204800*a^8*b^4*c^3 - 288000*a^9*b^2*c^4 +
60000*a^9*b^4*c^2 - 104000*a^10*b^2*c^3 - 22848*a^11*b^2*c^2) - 32*a^2*b^13
 + 160*a^4*b^11 - 320*a^6*b^9 + 320*a^8*b^7 - 160*a^10*b^5 + 32*a^12*b^3 -
32*a*b^3*c^11 + 160*a*b^5*c^9 - 320*a*b^7*c^7 + 320*a*b^9*c^5 - 160*a*b^11*
c^3 + 128*a^2*b*c^12 + 1152*a^3*b*c^11 + 288*a^3*b^11*c + 4480*a^4*b*c^10 +
 9600*a^5*b*c^9 - 1600*a^5*b^9*c + 11520*a^6*b*c^8 + 5376*a^7*b*c^7 + 2880*
a^7*b^7*c - 5376*a^8*b*c^6 - 11520*a^9*b*c^5 - 2400*a^9*b^5*c - 9600*a^10*b
*c^4 - 4480*a^11*b*c^3 + 928*a^11*b^3*c - 1152*a^12*b*c^2 - 928*a^2*b^3*c^1
0 + 2400*a^2*b^5*c^8 - 2880*a^2*b^7*c^6 + 1600*a^2*b^9*c^4 - 288*a^2*b^11*c
^2 - 5600*a^3*b^3*c^9 + 9600*a^3*b^5*c^7 - 6720*a^3*b^7*c^5 + 1280*a^3*b^9*
c^3 - 15200*a^4*b^3*c^8 + 16000*a^4*b^5*c^6 - 4160*a^4*b^7*c^4 - 1280*a^4*b
^9*c^2 - 20800*a^5*b^3*c^7 + 8640*a^5*b^5*c^5 + 4160*a^5*b^7*c^3 - 10304*a^
6*b^3*c^6 - 8640*a^6*b^5*c^4 + 6720*a^6*b^7*c^2 + 10304*a^7*b^3*c^5 - 16000
*a^7*b^5*c^3 + 20800*a^8*b^3*c^4 - 9600*a^8*b^5*c^2 + 15200*a^9*b^3*c^3 + 5
600*a^10*b^3*c^2 + 32*a*b^13*c - 128*a^13*b*c) + 32*a^2*b^12 - 128*a^4*b^10
 + 192*a^6*b^8 - 128*a^8*b^6 + 32*a^10*b^4 + 128*a^2*c^12 + 1280*a^3*c^11 +
 5760*a^4*c^10 + 15360*a^5*c^9 + 26880*a^6*c^8 + 32256*a^7*c^7 + 26880*a^8*
c^6 + 15360*a^9*c^5 + 5760*a^10*c^4 + 1280*a^11*c^3 + 128*a^12*c^2 - 32*a*b
^2*c^11 + 128*a*b^4*c^9 - 192*a*b^6*c^7 + 128*a*b^8*c^5 - 32*a*b^10*c^3 - 4
16*a^3*b^10*c + 1408*a^5*b^8*c - 1728*a^7*b^6*c + 896*a^9*b^4*c - 160*a^11*

b^2*c - 832*a^2*b^2*c^10 + 1824*a^2*b^4*c^8 - 1792*a^2*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^10*c^2 - 5664*a^3*b^2*c^9 + 8960*a^3*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*b^8*c^3 - 19200*a^4*b^2*c^8 + 22656*a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*a^4*b^8*c^2 - 38976*a^5*b^2*c^7 + 33792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 51072*a^6*b^2*c^6 + 31168*a^6*b^4*c^4 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5 + 17664*a^7*b^4*c^3 - 25344*a^8*b^2*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*c^3 - 1856*a^10*b^2*c^2) - 160*a*b^3*c^9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 + 160*a*b^9*c^3 + 384*a^2*b*c^10 + 1792*a^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b*c^8 + 6720*a^5*b*c^7 - 96*a^5*b^7*c + 6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32*a^7*b^5*c + 1152*a^8*b*c^4 + 160*a^9*b*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b^5*c^6 - 1440*a^2*b^7*c^4 + 352*a^2*b^9*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b^5*c^5 - 1888*a^3*b^7*c^3 - 9440*a^4*b^3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b^7*c^2 - 9440*a^5*b^3*c^5 + 3072*a^5*b^5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b^5*c^2 - 1504*a^7*b^3*c^3 - 160*a^8*b^3*c^2 + 32*a*b*c^11 - 32*a*b^11*c)*1i)/((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*((-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(64*a*b^13 - 256*a^3*b^11 + 384*a^5*b^9 - 256*a^7*b^7 + 64*a^9*b^5 - 128*a*b^3*c^10 + 576*a*b^5*c^8 - 1024*a*b^7*c^6 + 896*a*b^9*c^4 - 384*a*b^11*c^2 + 512*a^2*b*c^11 - 896*a^2*b^11*c + 4608*a^3*b*c^10 + 18432*a^4*b*c^9 + 3072*a^4*b^9*c + 43008*a^5*b*c^8 + 64512*a^6*b*c^7 - 3840*a^6*b^7*c + 64512*a^7*b*c^6 + 43008*a^8*b*c^5 + 2048*a^8*b^5*c + 18432*a^9*b*c^4 + 4608*a^10*b*c^3 - 384*a^10*b^3*c + 512*a^11*b*c^2 - 3456*a^2*b^3*c^9 + 8192*a^2*b^5*c^7 - 8960*a^2*b^7*c^5 + 4608*a^2*b^9*c^3 - 20992*a^3*b^3*c^8 + 34048*a^3*b^5*c^6 - 23808*a^3*b^7*c^4 + 6400*a^3*b^9*c^2 - 60928*a^4*b^3*c^7 + 67584*a^4*b^5*c^5 -

28160*a^4*b^7*c^3 - 102144*a^5*b^3*c^6 + 73600*a^5*b^5*c^4 - 15872*a^5*b^7*c^2 - 105728*a^6*b^3*c^5 + 45056*a^6*b^5*c^3 - 68096*a^7*b^3*c^4 + 14592*a^7*b^5*c^2 - 26112*a^8*b^3*c^3 - 5248*a^9*b^3*c^2) + (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(256*a^14*c - 96*a*b^14 + 544*a^3*b^12 - 1280*a^5*b^10 + 1600*a^7*b^8 - 1120*a^9*b^6 + 416*a^11*b^4 - 64*a^13*b^2 + 512*a^2*c^13 + 5888*a^3*c^12 + 30976*a^4*c^11 + 98560*a^5*c^10 + 211200*a^6*c^9 + 321024*a^7*c^8 + 354816*a^8*c^7 + 287232*a^9*c^6 + 168960*a^10*c^5 + 70400*a^11*c^4 + 19712*a^12*c^3 + 3328*a^13*c^2 - 128*a*b^2*c^12 + 736*a*b^4*c^10 - 1760*a*b^6*c^8 + 2240*a*b^8*c^6 - 1600*a*b^10*c^4 + 608*a*b^12*c^2 + 1536*a^2*b^12*c - 7616*a^4*b^10*c + 15360*a^6*b^8*c - 16000*a^8*b^6*c + 8960*a^10*b^4*c - 2496*a^12*b^2*c - 4416*a^2*b^2*c^11 + 14080*a^2*b^4*c^9 - 22400*a^2*b^6*c^7 + 19200*a^2*b^8*c^5 - 8512*a^2*b^10*c^3 - 35904*a^3*b^2*c^10 + 84000*a^3*b^4*c^8 - 96000*a^3*b^6*c^6 + 54720*a^3*b^8*c^4 - 13248*a^3*b^10*c^2 - 145600*a^4*b^2*c^9 + 256000*a^4*b^4*c^7 - 206720*a^4*b^6*c^5 + 72960*a^4*b^8*c^3 - 360000*a^5*b^2*c^8 + 468160*a^5*b^4*c^6 - 254400*a^5*b^6*c^4 + 48960*a^5*b^8*c^2 - 590976*a^6*b^2*c^7 + 548352*a^6*b^4*c^5 - 184960*a^6*b^6*c^3 - 669312*a^7*b^2*c^6 + 418880*a^7*b^4*c^4 - 76800*a^7*b^6*c^2 - 528768*a^8*b^2*c^5 + 204800*a^8*b^4*c^3 - 288000*a^9*b^2*c^4 + 60000*a^9*b^4*c^2 - 104000*a^10*b^2*c^3 - 22848*a^11*b^2*c^2) - 32*a^2*b^13 + 160*a^4*b^11 - 320*a^6*b^9 + 320*a^8*b^7 - 160*a^10*b^5 + 32*a^12*b^3 - 32*a*b^3*c^11 + 160*a*b^5*c^9 - 320*a*b^7*c^7 + 320*a*b^9*c^5 - 160*a*b^11*c^3 + 128*a^2*b*c^12 + 1152*a^3*b*c^11 + 288*a^3*b^11*c + 4480*a^4*b*c^10 + 9600*a^5*b*c^9 - 1600*a^5*b^9*c + 11520*a^6*b*c^8 + 5376*a^7*b*c^7 + 2880*a^7*b^7*c - 5376*a^8*b*c^6 - 11520*a^9*b*c^5 - 2400*a^9*b^5*c - 9600*a^10*b*c^4 - 4480*a^11*b*c^3 + 928*a^11*b^3*c - 1152*a^12*b*c^2 - 928*a^2*b^3*c^10 + 2400*a^2*b^5*c^8 - 2880*a^2*b^7*c^6 + 1600*a^2*b^9*c^4 - 288*a^2*b^11*c^2 - 5600*a^3*b^3*c^9 + 9600*a^3*b^5*c^7 - 6720*a^3*b^7*c^5 + 1280*a^3*b^9*c^3 - 15200*a^4*b^3*c^8 + 16000*a^4*b^5*c^6 - 4160*a^4*b^7*c^4 - 1280*a^4*b^9*c^2 - 20800*a^5*b^3*c^7 + 8640*a^5*b^5*c^5 + 4160*a^5*b^7*c^3 - 10304*a^6*b^3*c^6 - 8640*a^6*b^5*c^4 + 6720*a^6*b^7*c^2 + 10304*a^7*b^3*c^5 - 16000*a^7*b^5*c^3 + 20800*a^8*b^3*c^4 - 9600*a^8*b^5*c^2 + 15200*a^9*b^3*c^3 + 5600*a^10*b^3*c^2 + 32*a*b^13*c - 128*a^13*b*c) + 32*a^2*b^12 - 128*a^4*b^10 + 192*a^6*b^8 - 128*a^8*b^6 + 32*a^10*b^4 + 128*a^2*c^12 + 1280*a^3*c^11 + 5760*a^4*c^10 + 15360*a^5*c^9 + 26880*a^6*c^8 + 32256*a^7*c^7 + 26880*a^8*c^6 + 15360*a^9*c^5 + 5760*a^10*c^4

+ 1280*a^11*c^3 + 128*a^12*c^2 - 32*a*b^2*c^11 + 128*a*b^4*c^9 - 192*a*b^6*
c^7 + 128*a*b^8*c^5 - 32*a*b^10*c^3 - 416*a^3*b^10*c + 1408*a^5*b^8*c - 172
8*a^7*b^6*c + 896*a^9*b^4*c - 160*a^11*b^2*c - 832*a^2*b^2*c^10 + 1824*a^2*
b^4*c^8 - 1792*a^2*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^10*c^2 - 5664*a^3*
b^2*c^9 + 8960*a^3*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*b^8*c^3 - 19200*a^
4*b^2*c^8 + 22656*a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*a^4*b^8*c^2 - 3897
6*a^5*b^2*c^7 + 33792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 51072*a^6*b^2*c^6 +
31168*a^6*b^4*c^4 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5 + 17664*a^7*b^4*c
^3 - 25344*a^8*b^2*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*c^3 - 1856*a^10*b^
2*c^2) + tan(x/2)*(32*a*b^12 + 128*a*c^12 - 96*a^3*b^10 + 96*a^5*b^8 - 32*a
^7*b^6 + 1088*a^2*c^11 + 4096*a^3*c^10 + 8960*a^4*c^9 + 12544*a^5*c^8 + 116
48*a^6*c^7 + 7168*a^7*c^6 + 2816*a^8*c^5 + 640*a^9*c^4 + 64*a^10*c^3 - 544*
a*b^2*c^10 + 992*a*b^4*c^8 - 1024*a*b^6*c^6 + 640*a*b^8*c^4 - 224*a*b^10*c^
2 - 384*a^2*b^10*c + 960*a^4*b^8*c - 768*a^6*b^6*c + 192*a^8*b^4*c - 3968*a
^2*b^2*c^9 + 6144*a^2*b^4*c^7 - 5120*a^2*b^6*c^5 + 2240*a^2*b^8*c^3 - 12672
*a^3*b^2*c^8 + 16032*a^3*b^4*c^6 - 9760*a^3*b^6*c^4 + 2400*a^3*b^8*c^2 - 23
168*a^4*b^2*c^7 + 22720*a^4*b^4*c^5 - 8960*a^4*b^6*c^3 - 26560*a^5*b^2*c^6
+ 18720*a^5*b^4*c^4 - 4032*a^5*b^6*c^2 - 19584*a^6*b^2*c^5 + 8832*a^6*b^4*c
^3 - 9088*a^7*b^2*c^4 + 2144*a^7*b^4*c^2 - 2432*a^8*b^2*c^3 - 288*a^9*b^2*c
^2) - 160*a*b^3*c^9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 + 160*a*b^9*c^3 + 384*a
^2*b*c^10 + 1792*a^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b*c^8 + 6720*a^5*b*c^7
- 96*a^5*b^7*c + 6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32*a^7*b^5*c + 1152*a^8
*b*c^4 + 160*a^9*b*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b^5*c^6 - 1440*a^2*b^7
*c^4 + 352*a^2*b^9*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b^5*c^5 - 1888*a^3*b^7
*c^3 - 9440*a^4*b^3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b^7*c^2 - 9440*a^5*b^3
*c^5 + 3072*a^5*b^5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b^5*c^2 - 1504*a^7*b^3
*c^3 - 160*a^8*b^3*c^2 + 32*a*b*c^11 - 32*a*b^11*c) - 2*tan(x/2)*(192*a*b^5
*c^6 - 192*a*b^3*c^8 - 64*a*b^7*c^4 + 384*a^2*b*c^9 + 960*a^3*b*c^8 + 1280*
a^4*b*c^7 + 960*a^5*b*c^6 + 384*a^6*b*c^5 + 64*a^7*b*c^4 - 768*a^2*b^3*c^7
+ 384*a^2*b^5*c^5 - 1152*a^3*b^3*c^6 + 192*a^3*b^5*c^4 - 768*a^4*b^3*c^5 -
192*a^5*b^3*c^4 + 64*a*b*c^10) - (-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5
+ 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6
*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*
a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3
)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(
4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 -
b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a
^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^
8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b
^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 31
2*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a
^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*(32*a*b^12 + 12
8*a*c^12 - 96*a^3*b^10 + 96*a^5*b^8 - 32*a^7*b^6 + 1088*a^2*c^11 + 4096*a^3
*c^10 + 8960*a^4*c^9 + 12544*a^5*c^8 + 11648*a^6*c^7 + 7168*a^7*c^6 + 2816*
a^8*c^5 + 640*a^9*c^4 + 64*a^10*c^3 - 544*a*b^2*c^10 + 992*a*b^4*c^8 - 1024

*a*b^6*c^6 + 640*a*b^8*c^4 - 224*a*b^10*c^2 - 384*a^2*b^10*c + 960*a^4*b^8*
c - 768*a^6*b^6*c + 192*a^8*b^4*c - 3968*a^2*b^2*c^9 + 6144*a^2*b^4*c^7 - 5
120*a^2*b^6*c^5 + 2240*a^2*b^8*c^3 - 12672*a^3*b^2*c^8 + 16032*a^3*b^4*c^6
- 9760*a^3*b^6*c^4 + 2400*a^3*b^8*c^2 - 23168*a^4*b^2*c^7 + 22720*a^4*b^4*c
^5 - 8960*a^4*b^6*c^3 - 26560*a^5*b^2*c^6 + 18720*a^5*b^4*c^4 - 4032*a^5*b^
6*c^2 - 19584*a^6*b^2*c^5 + 8832*a^6*b^4*c^3 - 9088*a^7*b^2*c^4 + 2144*a^7*
b^4*c^2 - 2432*a^8*b^2*c^3 - 288*a^9*b^2*c^2) - (-(8*a*c^7 + b^8 + 24*a^2*c
^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*
b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)
^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-
(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2)
- 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/
(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*
a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3
*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b
^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a
^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*
b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1/2)*(tan(x/2)*
(64*a*b^13 - 256*a^3*b^11 + 384*a^5*b^9 - 256*a^7*b^7 + 64*a^9*b^5 - 128*a*
b^3*c^10 + 576*a*b^5*c^8 - 1024*a*b^7*c^6 + 896*a*b^9*c^4 - 384*a*b^11*c^2
+ 512*a^2*b*c^11 - 896*a^2*b^11*c + 4608*a^3*b*c^10 + 18432*a^4*b*c^9 + 307
2*a^4*b^9*c + 43008*a^5*b*c^8 + 64512*a^6*b*c^7 - 3840*a^6*b^7*c + 64512*a^
7*b*c^6 + 43008*a^8*b*c^5 + 2048*a^8*b^5*c + 18432*a^9*b*c^4 + 4608*a^10*b*
c^3 - 384*a^10*b^3*c + 512*a^11*b*c^2 - 3456*a^2*b^3*c^9 + 8192*a^2*b^5*c^7
- 8960*a^2*b^7*c^5 + 4608*a^2*b^9*c^3 - 20992*a^3*b^3*c^8 + 34048*a^3*b^5*
c^6 - 23808*a^3*b^7*c^4 + 6400*a^3*b^9*c^2 - 60928*a^4*b^3*c^7 + 67584*a^4*
b^5*c^5 - 28160*a^4*b^7*c^3 - 102144*a^5*b^3*c^6 + 73600*a^5*b^5*c^4 - 1587
2*a^5*b^7*c^2 - 105728*a^6*b^3*c^5 + 45056*a^6*b^5*c^3 - 68096*a^7*b^3*c^4
+ 14592*a^7*b^5*c^2 - 26112*a^8*b^3*c^3 - 5248*a^9*b^3*c^2) - (-(8*a*c^7 +
b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*(-(4*a*c - b^2)^3)^(1/2) -
2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5 + 24*a*b^4*c^3 - 3*b*c^4*(
-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^2*b^4*c^2 - 38*a^3*b^2*c^3
+ 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*c - 3*a^2*b*c^2*(-(4*a*c -
b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/2) + 4*a*b^3*c*(-(4*a*c - b
^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 + a^6*b^4 + 16*a^2*c^8 + 96*
a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^4 + 96*a^7*c^3 + 16*a^8*c^2
+ b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7 + 30*a*b^4*c^5 - 36*a*b^6*
c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c - 96*a^2*b^2*c^6 + 159*a^2*
b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260*a^3*b^4*c^3 - 448*a^4*b^2*
c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6*b^2*c^2 + 14*a*b^8*c)))^(1
/2)*(tan(x/2)*(256*a^14*c - 96*a*b^14 + 544*a^3*b^12 - 1280*a^5*b^10 + 1600
*a^7*b^8 - 1120*a^9*b^6 + 416*a^11*b^4 - 64*a^13*b^2 + 512*a^2*c^13 + 5888*
a^3*c^12 + 30976*a^4*c^11 + 98560*a^5*c^10 + 211200*a^6*c^9 + 321024*a^7*c^
8 + 354816*a^8*c^7 + 287232*a^9*c^6 + 168960*a^10*c^5 + 70400*a^11*c^4 + 19
712*a^12*c^3 + 3328*a^13*c^2 - 128*a*b^2*c^12 + 736*a*b^4*c^10 - 1760*a*b^6

```
*c^8 + 2240*a*b^8*c^6 - 1600*a*b^10*c^4 + 608*a*b^12*c^2 + 1536*a^2*b^12*c
- 7616*a^4*b^10*c + 15360*a^6*b^8*c - 16000*a^8*b^6*c + 8960*a^10*b^4*c - 2
496*a^12*b^2*c - 4416*a^2*b^2*c^11 + 14080*a^2*b^4*c^9 - 22400*a^2*b^6*c^7
+ 19200*a^2*b^8*c^5 - 8512*a^2*b^10*c^3 - 35904*a^3*b^2*c^10 + 84000*a^3*b^
4*c^8 - 96000*a^3*b^6*c^6 + 54720*a^3*b^8*c^4 - 13248*a^3*b^10*c^2 - 145600
*a^4*b^2*c^9 + 256000*a^4*b^4*c^7 - 206720*a^4*b^6*c^5 + 72960*a^4*b^8*c^3
- 360000*a^5*b^2*c^8 + 468160*a^5*b^4*c^6 - 254400*a^5*b^6*c^4 + 48960*a^5*
b^8*c^2 - 590976*a^6*b^2*c^7 + 548352*a^6*b^4*c^5 - 184960*a^6*b^6*c^3 - 66
9312*a^7*b^2*c^6 + 418880*a^7*b^4*c^4 - 76800*a^7*b^6*c^2 - 528768*a^8*b^2*
c^5 + 204800*a^8*b^4*c^3 - 288000*a^9*b^2*c^4 + 60000*a^9*b^4*c^2 - 104000*
a^10*b^2*c^3 - 22848*a^11*b^2*c^2) - 32*a^2*b^13 + 160*a^4*b^11 - 320*a^6*b
^9 + 320*a^8*b^7 - 160*a^10*b^5 + 32*a^12*b^3 - 32*a*b^3*c^11 + 160*a*b^5*c
^9 - 320*a*b^7*c^7 + 320*a*b^9*c^5 - 160*a*b^11*c^3 + 128*a^2*b*c^12 + 1152
*a^3*b*c^11 + 288*a^3*b^11*c + 4480*a^4*b*c^10 + 9600*a^5*b*c^9 - 1600*a^5*
b^9*c + 11520*a^6*b*c^8 + 5376*a^7*b*c^7 + 2880*a^7*b^7*c - 5376*a^8*b*c^6
- 11520*a^9*b*c^5 - 2400*a^9*b^5*c - 9600*a^10*b*c^4 - 4480*a^11*b*c^3 + 92
8*a^11*b^3*c - 1152*a^12*b*c^2 - 928*a^2*b^3*c^10 + 2400*a^2*b^5*c^8 - 2880
*a^2*b^7*c^6 + 1600*a^2*b^9*c^4 - 288*a^2*b^11*c^2 - 5600*a^3*b^3*c^9 + 960
0*a^3*b^5*c^7 - 6720*a^3*b^7*c^5 + 1280*a^3*b^9*c^3 - 15200*a^4*b^3*c^8 + 1
6000*a^4*b^5*c^6 - 4160*a^4*b^7*c^4 - 1280*a^4*b^9*c^2 - 20800*a^5*b^3*c^7
+ 8640*a^5*b^5*c^5 + 4160*a^5*b^7*c^3 - 10304*a^6*b^3*c^6 - 8640*a^6*b^5*c^
4 + 6720*a^6*b^7*c^2 + 10304*a^7*b^3*c^5 - 16000*a^7*b^5*c^3 + 20800*a^8*b^
3*c^4 - 9600*a^8*b^5*c^2 + 15200*a^9*b^3*c^3 + 5600*a^10*b^3*c^2 + 32*a*b^1
3*c - 128*a^13*b*c) + 32*a^2*b^12 - 128*a^4*b^10 + 192*a^6*b^8 - 128*a^8*b^
6 + 32*a^10*b^4 + 128*a^2*c^12 + 1280*a^3*c^11 + 5760*a^4*c^10 + 15360*a^5*
c^9 + 26880*a^6*c^8 + 32256*a^7*c^7 + 26880*a^8*c^6 + 15360*a^9*c^5 + 5760*
a^10*c^4 + 1280*a^11*c^3 + 128*a^12*c^2 - 32*a*b^2*c^11 + 128*a*b^4*c^9 - 1
92*a*b^6*c^7 + 128*a*b^8*c^5 - 32*a*b^10*c^3 - 416*a^3*b^10*c + 1408*a^5*b^
8*c - 1728*a^7*b^6*c + 896*a^9*b^4*c - 160*a^11*b^2*c - 832*a^2*b^2*c^10 +
1824*a^2*b^4*c^8 - 1792*a^2*b^6*c^6 + 832*a^2*b^8*c^4 - 192*a^2*b^10*c^2 -
5664*a^3*b^2*c^9 + 8960*a^3*b^4*c^7 - 6464*a^3*b^6*c^5 + 2304*a^3*b^8*c^3 -
 19200*a^4*b^2*c^8 + 22656*a^4*b^4*c^6 - 11904*a^4*b^6*c^4 + 2816*a^4*b^8*c
^2 - 38976*a^5*b^2*c^7 + 33792*a^5*b^4*c^5 - 12096*a^5*b^6*c^3 - 51072*a^6*
b^2*c^6 + 31168*a^6*b^4*c^4 - 6656*a^6*b^6*c^2 - 44352*a^7*b^2*c^5 + 17664*
a^7*b^4*c^3 - 25344*a^8*b^2*c^4 + 5760*a^8*b^4*c^2 - 9120*a^9*b^2*c^3 - 185
6*a^10*b^2*c^2) - 160*a*b^3*c^9 + 320*a*b^5*c^7 - 320*a*b^7*c^5 + 160*a*b^9
*c^3 + 384*a^2*b*c^10 + 1792*a^3*b*c^9 + 96*a^3*b^9*c + 4480*a^4*b*c^8 + 67
20*a^5*b*c^7 - 96*a^5*b^7*c + 6272*a^6*b*c^6 + 3584*a^7*b*c^5 + 32*a^7*b^5*
c + 1152*a^8*b*c^4 + 160*a^9*b*c^3 - 1504*a^2*b^3*c^8 + 2208*a^2*b^5*c^6 -
1440*a^2*b^7*c^4 + 352*a^2*b^9*c^2 - 5280*a^3*b^3*c^7 + 5280*a^3*b^5*c^5 -
1888*a^3*b^7*c^3 - 9440*a^4*b^3*c^6 + 5824*a^4*b^5*c^4 - 864*a^4*b^7*c^2 -
9440*a^5*b^3*c^5 + 3072*a^5*b^5*c^3 - 5280*a^6*b^3*c^4 + 672*a^6*b^5*c^2 -
1504*a^7*b^3*c^3 - 160*a^8*b^3*c^2 + 32*a*b*c^11 - 32*a*b^11*c) + 64*a*c^11
 + 448*a^2*c^10 + 1344*a^3*c^9 + 2240*a^4*c^8 + 2240*a^5*c^7 + 1344*a^6*c^6
 + 448*a^7*c^5 + 64*a^8*c^4 - 256*a*b^2*c^9 + 384*a*b^4*c^7 - 256*a*b^6*c^5
```

```
 + 64*a*b^8*c^3 - 1344*a^2*b^2*c^8 + 1344*a^2*b^4*c^6 - 448*a^2*b^6*c^4 - 2
880*a^3*b^2*c^7 + 1728*a^3*b^4*c^5 - 192*a^3*b^6*c^3 - 3200*a^4*b^2*c^6 + 9
60*a^4*b^4*c^4 - 1920*a^5*b^2*c^5 + 192*a^5*b^4*c^3 - 576*a^6*b^2*c^4 - 64*
a^7*b^2*c^3))*(-(8*a*c^7 + b^8 + 24*a^2*c^6 + 24*a^3*c^5 + 8*a^4*c^4 - b^5*
(-(4*a*c - b^2)^3)^(1/2) - 2*b^2*c^6 + 3*b^4*c^4 - 3*b^6*c^2 - 18*a*b^2*c^5
 + 24*a*b^4*c^3 - 3*b*c^4*(-(4*a*c - b^2)^3)^(1/2) - 54*a^2*b^2*c^4 + 33*a^
2*b^4*c^2 - 38*a^3*b^2*c^3 + 3*b^3*c^2*(-(4*a*c - b^2)^3)^(1/2) - 10*a*b^6*
c - 3*a^2*b*c^2*(-(4*a*c - b^2)^3)^(1/2) - 6*a*b*c^3*(-(4*a*c - b^2)^3)^(1/
2) + 4*a*b^3*c*(-(4*a*c - b^2)^3)^(1/2))/(2*(3*a^2*b^8 - b^10 - 3*a^4*b^6 +
 a^6*b^4 + 16*a^2*c^8 + 96*a^3*c^7 + 240*a^4*c^6 + 320*a^5*c^5 + 240*a^6*c^
4 + 96*a^7*c^3 + 16*a^8*c^2 + b^4*c^6 - 3*b^6*c^4 + 3*b^8*c^2 - 8*a*b^2*c^7
 + 30*a*b^4*c^5 - 36*a*b^6*c^3 - 36*a^3*b^6*c + 30*a^5*b^4*c - 8*a^7*b^2*c
- 96*a^2*b^2*c^6 + 159*a^2*b^4*c^4 - 82*a^2*b^6*c^2 - 312*a^3*b^2*c^5 + 260
*a^3*b^4*c^3 - 448*a^4*b^2*c^4 + 159*a^4*b^4*c^2 - 312*a^5*b^2*c^3 - 96*a^6
*b^2*c^2 + 14*a*b^8*c)))^(1/2)*2i + ((2*b)/(2*a*c + a^2 - b^2 + c^2) - (2*t
an(x/2)*(a + c))/(2*a*c + a^2 - b^2 + c^2))/(tan(x/2)^2 - 1)
```

3.14 $\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$

Optimal result

Integrand size = 19, antiderivative size = 206

$$\begin{aligned}\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx =& -\frac{(b^4+2c^2(a+c)^2-2b^2c(2a+c))\operatorname{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}\,(a^2-b^2+2ac+c^2)^2} \\ &- \frac{(a+2b+3c)\log(1-\sin(x))}{4(a+b+c)^2} \\ &+ \frac{(a-2b+3c)\log(1+\sin(x))}{4(a-b+c)^2} \\ &+ \frac{b(b^2-2c(a+c))\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2\left(a^2-b^2+2ac+c^2\right)^2} \\ &- \frac{\sec^2(x)(b-(a+c)\sin(x))}{2(a-b+c)(a+b+c)}\end{aligned}$$

```
[Out] -1/4*(a+2*b+3*c)*ln(1-sin(x))/(a+b+c)^2+1/4*(a-2*b+3*c)*ln(1+sin(x))/(a-b+c
)^2+1/2*b*(b^2-2*c*(a+c))*ln(a+b*sin(x)+c*sin(x)^2)/(a^2+2*a*c-b^2+c^2)^2-1
/2*sec(x)^2*(b-(a+c)*sin(x))/(a-b+c)/(a+b+c)-(b^4+2*c^2*(a+c)^2-2*b^2*c*(2*
a+c))*arctanh((b+2*c*sin(x))/(-4*a*c+b^2)^(1/2))/(a^2+2*a*c-b^2+c^2)^2/(-4*
a*c+b^2)^(1/2)
```

Rubi [A] (verified)

Time = 0.54 (sec) , antiderivative size = 206, normalized size of antiderivative = 1.00, number of steps used = 10, number of rules used = 9, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.474, Rules used = {3339, 990, 1088, 648, 632, 212, 642, 647, 31}

$$\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = -\frac{\left(-2b^2c(2a+c)+2c^2(a+c)^2+b^4\right)\operatorname{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}\,\left(a^2+2ac-b^2+c^2\right)^2} + \frac{b(b^2-2c(a+c))\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2\left(a^2+2ac-b^2+c^2\right)^2} - \frac{(a+2b+3c)\log(1-\sin(x))}{4(a+b+c)^2} + \frac{(a-2b+3c)\log(\sin(x)+1)}{4(a-b+c)^2} - \frac{\sec^2(x)(b-(a+c)\sin(x))}{2(a-b+c)(a+b+c)}$$

```
[In] Int[Sec[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] -(((b^4 + 2*c^2*(a + c)^2 - 2*b^2*c*(2*a + c))*ArcTanh[(b + 2*c*Sin[x])/Sqr
t[b^2 - 4*a*c]])/(Sqrt[b^2 - 4*a*c]*(a^2 - b^2 + 2*a*c + c^2)^2)) - ((a + 2
*b + 3*c)*Log[1 - Sin[x]])/(4*(a + b + c)^2) + ((a - 2*b + 3*c)*Log[1 + Sin
[x]])/(4*(a - b + c)^2) + (b*(b^2 - 2*c*(a + c))*Log[a + b*Sin[x] + c*Sin[x
]^2])/(2*(a^2 - b^2 + 2*a*c + c^2)^2) - (Sec[x]^2*(b - (a + c)*Sin[x]))/(2*
(a - b + c)*(a + b + c))
```

Rule 31

```
Int[((a_) + (b_.)*(x_))^(-1), x_Symbol] :> Simp[Log[RemoveContent[a + b*x,
x]]/b, x] /; FreeQ[{a, b}, x]
```

Rule 212

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(1/(Rt[a, 2]*Rt[-b, 2]))*
ArcTanh[Rt[-b, 2]*(x/Rt[a, 2])], x] /; FreeQ[{a, b}, x] && NegQ[a/b] && (Gt
Q[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 642

```
Int[((d_) + (e_.)*(x_))/((a_.) + (b_.)*(x_) + (c_.)*(x_)^2), x_Symbol] :> S
imp[d*(Log[RemoveContent[a + b*x + c*x^2, x]]/b), x] /; FreeQ[{a, b, c, d,
```

```
e}, x] && EqQ[2*c*d - b*e, 0]
```

Rule 647

```
Int[((d_) + (e_.)*(x_))/((a_) + (c_.)*(x_)^2), x_Symbol] :> With[{q = Rt[(-
a)*c, 2]}, Dist[e/2 + c*(d/(2*q)), Int[1/(-q + c*x), x], x] + Dist[e/2 - c*
(d/(2*q)), Int[1/(q + c*x), x], x]] /; FreeQ[{a, c, d, e}, x] && NiceSqrtQ[
(-a)*c]
```

Rule 648

```
Int[((d_.) + (e_.)*(x_))/((a_) + (b_.)*(x_) + (c_.)*(x_)^2), x_Symbol] :> D
ist[(2*c*d - b*e)/(2*c), Int[1/(a + b*x + c*x^2), x], x] + Dist[e/(2*c), In
t[(b + 2*c*x)/(a + b*x + c*x^2), x], x] /; FreeQ[{a, b, c, d, e}, x] && NeQ
[2*c*d - b*e, 0] && NeQ[b^2 - 4*a*c, 0] &&  !NiceSqrtQ[b^2 - 4*a*c]
```

Rule 990

```
Int[((a_.) + (c_.)*(x_)^2)^(p_)*((d_.) + (e_.)*(x_) + (f_.)*(x_)^2)^(q_), x
_Symbol] :> Simp[(2*a*c^2*e + c*(2*c^2*d - c*(2*a*f))*x)*(a + c*x^2)^(p + 1
)*((d + e*x + f*x^2)^(q + 1)/((-4*a*c)*(a*c*e^2 + (c*d - a*f)^2)*(p + 1))),
 x] - Dist[1/((-4*a*c)*(a*c*e^2 + (c*d - a*f)^2)*(p + 1)), Int[(a + c*x^2)^
(p + 1)*(d + e*x + f*x^2)^q*Simp[2*c*((c*d - a*f)^2 - ((-a)*e)*(c*e))*(p +
1) - (2*c^2*d - c*(2*a*f))*(a*f*(p + 1) - c*d*(p + 2)) - e*(-2*a*c^2*e)*(p
+ q + 2) + (2*f*(2*a*c^2*e)*(p + q + 2) - (2*c^2*d - c*(2*a*f))*((-c)*e*(2*
p + q + 4)))*x + c*f*(2*c^2*d - c*(2*a*f))*(2*p + 2*q + 5)*x^2, x], x], x]
/; FreeQ[{a, c, d, e, f, q}, x] && NeQ[e^2 - 4*d*f, 0] && LtQ[p, -1] && NeQ
[a*c*e^2 + (c*d - a*f)^2, 0] &&  !( !IntegerQ[p] && ILtQ[q, -1]) &&  !IGtQ[
q, 0]
```

Rule 1088

```
Int[((A_.) + (B_.)*(x_) + (C_.)*(x_)^2)/(((a_) + (b_.)*(x_) + (c_.)*(x_)^2)
*((d_) + (f_.)*(x_)^2)), x_Symbol] :> With[{q = c^2*d^2 + b^2*d*f - 2*a*c*d
*f + a^2*f^2}, Dist[1/q, Int[(A*c^2*d - a*c*C*d + A*b^2*f - a*b*B*f - a*A*c
*f + a^2*C*f + c*(B*c*d - b*C*d + A*b*f - a*B*f)*x)/(a + b*x + c*x^2), x],
x] + Dist[1/q, Int[(c*C*d^2 + b*B*d*f - A*c*d*f - a*C*d*f + a*A*f^2 - f*(B*
c*d - b*C*d + A*b*f - a*B*f)*x)/(d + f*x^2), x], x] /; NeQ[q, 0]] /; FreeQ[
{a, b, c, d, f, A, B, C}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
```

```
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\text{integral} = \text{Subst}\left(\int \frac{1}{(1-x^2)^2\,(a+bx+cx^2)}\,dx, x, \sin(x)\right)$$

$$= -\frac{\sec^2(x)(b-(a+c)\sin(x))}{2(a-b+c)(a+b+c)} + \frac{\text{Subst}\left(\int \frac{2(a^2-2b^2+3ac+2c^2)+2b(a-c)x+2c(a+c)x^2}{(1-x^2)(a+bx+cx^2)}\,dx, x, \sin(x)\right)}{4(a-b+c)(a+b+c)}$$

$$= -\frac{\sec^2(x)(b-(a+c)\sin(x))}{2(a-b+c)(a+b+c)}$$
$$+ \frac{\text{Subst}\Big(\int \frac{-2b^2(a-c)+2ac(a+c)+2c^2(a+c)+2a(a^2-2b^2+3ac+2c^2)+2c(a^2-2b^2+3ac+2c^2)+(2ab(a-c)+2b(a-c)c-2bc(a+c)-2b}{1-x^2}}{4(a-b+c)^2(a+b+c)^2}$$
$$+ \frac{\text{Subst}\Big(\int \frac{2ab^2(a-c)-2a^2c(a+c)-2ac^2(a+c)-2b^2(a^2-2b^2+3ac+2c^2)+2ac(a^2-2b^2+3ac+2c^2)+2c^2(a^2-2b^2+3ac+2c^2)+c(2ab(}{a+bx+cx^2}}{4(a-b+c)^2(a+b+c)^2}$$

$$= -\frac{\sec^2(x)(b-(a+c)\sin(x))}{2(a-b+c)(a+b+c)} - \frac{(a-2b+3c)\text{Subst}\left(\int \frac{1}{-1-x}\,dx, x, \sin(x)\right)}{4(a-b+c)^2}$$
$$+ \frac{(a+2b+3c)\text{Subst}\left(\int \frac{1}{1-x}\,dx, x, \sin(x)\right)}{4(a+b+c)^2}$$
$$+ \frac{(b(b^2-2c(a+c)))\,\text{Subst}\left(\int \frac{b+2cx}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{2(a-b+c)^2(a+b+c)^2}$$
$$+ \frac{(b^4+2c^2(a+c)^2-2b^2c(2a+c))\,\text{Subst}\left(\int \frac{1}{a+bx+cx^2}\,dx, x, \sin(x)\right)}{2(a-b+c)^2(a+b+c)^2}$$

$$= -\frac{(a+2b+3c)\log(1-\sin(x))}{4(a+b+c)^2} + \frac{(a-2b+3c)\log(1+\sin(x))}{4(a-b+c)^2}$$
$$+ \frac{b(b^2-2c(a+c))\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2(a-b+c)^2(a+b+c)^2} - \frac{\sec^2(x)(b-(a+c)\sin(x))}{2(a-b+c)(a+b+c)}$$
$$- \frac{(b^4+2c^2(a+c)^2-2b^2c(2a+c))\,\text{Subst}\left(\int \frac{1}{b^2-4ac-x^2}\,dx, x, b+2c\sin(x)\right)}{(a-b+c)^2(a+b+c)^2}$$

$$= -\frac{(b^4+2c^2(a+c)^2-2b^2c(2a+c))\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{(a-b+c)^2(a+b+c)^2\sqrt{b^2-4ac}}$$
$$- \frac{(a+2b+3c)\log(1-\sin(x))}{4(a+b+c)^2} + \frac{(a-2b+3c)\log(1+\sin(x))}{4(a-b+c)^2}$$
$$+ \frac{b(b^2-2c(a+c))\log\left(a+b\sin(x)+c\sin^2(x)\right)}{2(a-b+c)^2(a+b+c)^2} - \frac{\sec^2(x)(b-(a+c)\sin(x))}{2(a-b+c)(a+b+c)}$$

Mathematica [A] (verified)

Time = 0.76 (sec) , antiderivative size = 202, normalized size of antiderivative = 0.98

$$\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \frac{1}{4}\left(-\frac{4(b^4+2c^2(a+c)^2-2b^2c(2a+c))\,\text{arctanh}\left(\frac{b+2c\sin(x)}{\sqrt{b^2-4ac}}\right)}{\sqrt{b^2-4ac}\,(a^2-b^2+2ac+c^2)^2} - \frac{(a+2b+3c)\log(1-\sin(x))}{(a+b+c)^2} + \frac{(a-2b+3c)\log(1+\sin(x))}{(a-b+c)^2} + \frac{2b(b^2-2c(a+c))\log\left(a+b\sin(x)+c\sin^2(x)\right)}{(a^2-b^2+2ac+c^2)^2} - \frac{1}{(a+b+c)(-1+\sin(x))} - \frac{1}{(a-b+c)(1+\sin(x))}\right)$$

```
[In] Integrate[Sec[x]^3/(a + b*Sin[x] + c*Sin[x]^2),x]
```

```
[Out] ((-4*(b^4 + 2*c^2*(a + c)^2 - 2*b^2*c*(2*a + c))*ArcTanh[(b + 2*c*Sin[x])/S
qrt[b^2 - 4*a*c]])/(Sqrt[b^2 - 4*a*c]*(a^2 - b^2 + 2*a*c + c^2)^2) - ((a +
2*b + 3*c)*Log[1 - Sin[x]])/(a + b + c)^2 + ((a - 2*b + 3*c)*Log[1 + Sin[x]
])/(a - b + c)^2 + (2*b*(b^2 - 2*c*(a + c))*Log[a + b*Sin[x] + c*Sin[x]^2])
/(a^2 - b^2 + 2*a*c + c^2)^2 - 1/((a + b + c)*(-1 + Sin[x])) - 1/((a - b +
c)*(1 + Sin[x])))/4
```

Maple [A] (verified)

Time = 2.19 (sec) , antiderivative size = 236, normalized size of antiderivative = 1.15

method	result
default	$\frac{\frac{\left(-2ab\,c^2+b^3c-2b\,c^3\right)\ln\left(a+b\sin(x)+c\left(\sin^2(x)\right)\right)}{2c}+\frac{2\left(a^2c^2-3a\,b^2c+2a\,c^3+b^4-2b^2c^2+c^4-\frac{\left(-2ab\,c^2+b^3c-2b\,c^3\right)b}{2c}\right)\arctan\left(\frac{b+2\sin(x)c}{\sqrt{4ac-b^2}}\right)}{\sqrt{4ac-b^2}}}{(a-b+c)^2(a+b+c)^2} -$
risch	Expression too large to display

```
[In] int(sec(x)^3/(a+b*sin(x)+c*sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] 1/(a-b+c)^2/(a+b+c)^2*(1/2*(-2*a*b*c^2+b^3*c-2*b*c^3)/c*ln(a+b*sin(x)+c*sin
(x)^2)+2*(a^2*c^2-3*a*b^2*c+2*a*c^3+b^4-2*b^2*c^2+c^4-1/2*(-2*a*b*c^2+b^3*c
-2*b*c^3)*b/c)/(4*a*c-b^2)^(1/2)*arctan((b+2*sin(x)*c)/(4*a*c-b^2)^(1/2)))-
1/(4*a-4*b+4*c)/(1+sin(x))+1/4*(a-2*b+3*c)*ln(1+sin(x))/(a-b+c)^2-1/(4*a+4*
b+4*c)/(sin(x)-1)+1/4/(a+b+c)^2*(-a-2*b-3*c)*ln(sin(x)-1)
```

Fricas [B] (verification not implemented)

Leaf count of result is larger than twice the leaf count of optimal. 603 vs. 2(195) = 390.

Time = 5.87 (sec) , antiderivative size = 1244, normalized size of antiderivative = 6.04

$$\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] integrate(sec(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="fricas")
```

```
[Out] [-1/4*(2*a^2*b^3 - 2*b^5 - 8*a*b*c^3 - 2*(b^4 - 4*a*b^2*c + 4*a*c^3 + 2*c^4
 + 2*(a^2 - b^2)*c^2)*sqrt(b^2 - 4*a*c)*cos(x)^2*log(-(2*c^2*cos(x)^2 - 2*b
*c*sin(x) - b^2 + 2*a*c - 2*c^2 + sqrt(b^2 - 4*a*c)*(2*c*sin(x) + b))/(c*co
s(x)^2 - b*sin(x) - a - c)) - 2*(b^5 - 6*a*b^3*c + 8*a*b*c^3 + 2*(4*a^2*b -
 b^3)*c^2)*cos(x)^2*log(-c*cos(x)^2 + b*sin(x) + a + c) - (a^3*b^2 - 3*a*b^
4 - 2*b^5 - 12*a*c^4 - (28*a^2 + 16*a*b - 3*b^2)*c^3 - (20*a^3 + 16*a^2*b -
 11*a*b^2 - 4*b^3)*c^2 - (4*a^4 - 17*a^2*b^2 - 12*a*b^3 + b^4)*c)*cos(x)^2*
log(sin(x) + 1) + (a^3*b^2 - 3*a*b^4 + 2*b^5 - 12*a*c^4 - (28*a^2 - 16*a*b
- 3*b^2)*c^3 - (20*a^3 - 16*a^2*b - 11*a*b^2 + 4*b^3)*c^2 - (4*a^4 - 17*a^2
*b^2 + 12*a*b^3 + b^4)*c)*cos(x)^2*log(-sin(x) + 1) - 2*(8*a^2*b - b^3)*c^2
 - 4*(2*a^3*b - 3*a*b^3)*c - 2*(a^3*b^2 - a*b^4 - 4*a*c^4 - (12*a^2 - b^2)*
c^3 - (12*a^3 - 7*a*b^2)*c^2 - (4*a^4 - 7*a^2*b^2 + b^4)*c)*sin(x))/((a^4*b
^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^3 - a*b^2)*c^
3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*a*b^4)*c)*cos
(x)^2), -1/4*(2*a^2*b^3 - 2*b^5 - 8*a*b*c^3 + 4*(b^4 - 4*a*b^2*c + 4*a*c^3
+ 2*c^4 + 2*(a^2 - b^2)*c^2)*sqrt(-b^2 + 4*a*c)*arctan(-sqrt(-b^2 + 4*a*c)*
(2*c*sin(x) + b)/(b^2 - 4*a*c))*cos(x)^2 - 2*(b^5 - 6*a*b^3*c + 8*a*b*c^3 +
 2*(4*a^2*b - b^3)*c^2)*cos(x)^2*log(-c*cos(x)^2 + b*sin(x) + a + c) - (a^3
*b^2 - 3*a*b^4 - 2*b^5 - 12*a*c^4 - (28*a^2 + 16*a*b - 3*b^2)*c^3 - (20*a^3
 + 16*a^2*b - 11*a*b^2 - 4*b^3)*c^2 - (4*a^4 - 17*a^2*b^2 - 12*a*b^3 + b^4)
*c)*cos(x)^2*log(sin(x) + 1) + (a^3*b^2 - 3*a*b^4 + 2*b^5 - 12*a*c^4 - (28*
a^2 - 16*a*b - 3*b^2)*c^3 - (20*a^3 - 16*a^2*b - 11*a*b^2 + 4*b^3)*c^2 - (4
*a^4 - 17*a^2*b^2 + 12*a*b^3 + b^4)*c)*cos(x)^2*log(-sin(x) + 1) - 2*(8*a^2
*b - b^3)*c^2 - 4*(2*a^3*b - 3*a*b^3)*c - 2*(a^3*b^2 - a*b^4 - 4*a*c^4 - (1
2*a^2 - b^2)*c^3 - (12*a^3 - 7*a*b^2)*c^2 - (4*a^4 - 7*a^2*b^2 + b^4)*c)*si
n(x))/((a^4*b^2 - 2*a^2*b^4 + b^6 - 4*a*c^5 - (16*a^2 - b^2)*c^4 - 12*(2*a^
3 - a*b^2)*c^3 - 2*(8*a^4 - 11*a^2*b^2 + b^4)*c^2 - 4*(a^5 - 3*a^3*b^2 + 2*
a*b^4)*c)*cos(x)^2)]
```

Sympy [F]

$$\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx$$

```
[In] integrate(sec(x)**3/(a+b*sin(x)+c*sin(x)**2),x)
[Out] Integral(sec(x)**3/(a + b*sin(x) + c*sin(x)**2), x)
```

Maxima [F(-2)]

Exception generated.

$$\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Exception raised: ValueError}$$

```
[In] integrate(sec(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="maxima")
[Out] Exception raised: ValueError >> Computation failed since Maxima requested a
dditional constraints; using the 'assume' command before evaluation *may* h
elp (example of legal syntax is 'assume(4*a*c-b^2>0)', see `assume?` for mo
re deta
```

Giac [A] (verification not implemented)

none

Time = 0.36 (sec) , antiderivative size = 377, normalized size of antiderivative = 1.83

$$\begin{aligned}\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx &= \frac{(b^3-2abc-2bc^2)\log\left(c\sin(x)^2+b\sin(x)+a\right)}{2\left(a^4-2a^2b^2+b^4+4a^3c-4ab^2c+6a^2c^2-2b^2c^2+4ac^3+c^4\right)} \\ &\quad+\frac{(a-2b+3c)\log(\sin(x)+1)}{4\left(a^2-2ab+b^2+2ac-2bc+c^2\right)}-\frac{(a+2b+3c)\log(-\sin(x)+1)}{4\left(a^2+2ab+b^2+2ac+2bc+c^2\right)} \\ &\quad+\frac{\left(b^4-4ab^2c+2a^2c^2-2b^2c^2+4ac^3+2c^4\right)\arctan\left(\frac{2c\sin(x)+b}{\sqrt{-b^2+4ac}}\right)}{\left(a^4-2a^2b^2+b^4+4a^3c-4ab^2c+6a^2c^2-2b^2c^2+4ac^3+c^4\right)\sqrt{-b^2+4ac}} \\ &\quad+\frac{a^2b-b^3+2abc+bc^2-\left(a^3-ab^2+3a^2c-b^2c+3ac^2+c^3\right)\sin(x)}{2(a+b+c)^2(a-b+c)^2(\sin(x)+1)(\sin(x)-1)}\end{aligned}$$

```
[In] integrate(sec(x)^3/(a+b*sin(x)+c*sin(x)^2),x, algorithm="giac")
```

```
[Out] 1/2*(b^3 - 2*a*b*c - 2*b*c^2)*log(c*sin(x)^2 + b*sin(x) + a)/(a^4 - 2*a^2*b
^2 + b^4 + 4*a^3*c - 4*a*b^2*c + 6*a^2*c^2 - 2*b^2*c^2 + 4*a*c^3 + c^4) + 1
/4*(a - 2*b + 3*c)*log(sin(x) + 1)/(a^2 - 2*a*b + b^2 + 2*a*c - 2*b*c + c^2
) - 1/4*(a + 2*b + 3*c)*log(-sin(x) + 1)/(a^2 + 2*a*b + b^2 + 2*a*c + 2*b*c
 + c^2) + (b^4 - 4*a*b^2*c + 2*a^2*c^2 - 2*b^2*c^2 + 4*a*c^3 + 2*c^4)*arcta
n((2*c*sin(x) + b)/sqrt(-b^2 + 4*a*c))/((a^4 - 2*a^2*b^2 + b^4 + 4*a^3*c -
4*a*b^2*c + 6*a^2*c^2 - 2*b^2*c^2 + 4*a*c^3 + c^4)*sqrt(-b^2 + 4*a*c)) + 1/
2*(a^2*b - b^3 + 2*a*b*c + b*c^2 - (a^3 - a*b^2 + 3*a^2*c - b^2*c + 3*a*c^2
 + c^3)*sin(x))/((a + b + c)^2*(a - b + c)^2*(sin(x) + 1)*(sin(x) - 1))
```

Mupad [B] (verification not implemented)

Time = 34.64 (sec) , antiderivative size = 2743, normalized size of antiderivative = 13.32

$$\int \frac{\sec^3(x)}{a+b\sin(x)+c\sin^2(x)}\,dx = \text{Too large to display}$$

```
[In] int(1/(cos(x)^3*(a + c*sin(x)^2 + b*sin(x))),x)
```

```
[Out] log(sin(x) + 1)*(1/(4*(a - b + c)) - (b/4 - c/2)/(a - b + c)^2) - (b/(2*(2*
a*c + a^2 - b^2 + c^2)) - (sin(x)*(a + c))/(2*(2*a*c + a^2 - b^2 + c^2)))/c
os(x)^2 - log(sin(x) - 1)*((b/4 + c/2)/(a + b + c)^2 + 1/(4*(a + b + c))) +
 (log((c^4*(4*a*c + a^2 - 4*b^2 + 3*c^2))/(4*(2*a*c + a^2 - b^2 + c^2)^2) -
 (((((c*(a*b^4 + 28*a*c^4 + 4*a^4*c - 5*b^4*c + 8*c^5 - a^3*b^2 + 36*a^2*c^
3 + 20*a^3*c^2 + 5*b^2*c^3 - 3*a*b^2*c^2 - 9*a^2*b^2*c))/(2*(2*a*c + a^2 -
b^2 + c^2)) + (b*c*sin(x)*(36*a*c^3 + 4*a^3*c + 3*b^4 + 16*c^4 - a^2*b^2 +
24*a^2*c^2 - 13*b^2*c^2 - 18*a*b^2*c))/(2*a*c + a^2 - b^2 + c^2) - (2*c*((b
^4*(b^2 - 4*a*c)^(1/2))/2 - b^5/2 + c^4*(b^2 - 4*a*c)^(1/2) + b^3*c^2 + 2*a
*c^3*(b^2 - 4*a*c)^(1/2) - 4*a^2*b*c^2 + a^2*c^2*(b^2 - 4*a*c)^(1/2) - b^2*
c^2*(b^2 - 4*a*c)^(1/2) - 4*a*b*c^3 + 3*a*b^3*c - 2*a*b^2*c*(b^2 - 4*a*c)^(
1/2))*(3*b^4*sin(x) + 4*c^4*sin(x) + 4*a*b^3 + 2*b*c^3 + 2*b^3*c + 4*a*c^3*
sin(x) - 4*a^3*c*sin(x) + a^2*b^2*sin(x) - 4*a^2*c^2*sin(x) - 3*b^2*c^2*sin
(x) - 12*a*b*c^2 - 14*a^2*b*c - 10*a*b^2*c*sin(x)))/((4*a*c - b^2)*(2*a*c +
 a^2 - b^2 + c^2)^2))*((b^4*(b^2 - 4*a*c)^(1/2))/2 - b^5/2 + c^4*(b^2 - 4*a
*c)^(1/2) + b^3*c^2 + 2*a*c^3*(b^2 - 4*a*c)^(1/2) - 4*a^2*b*c^2 + a^2*c^2*(
b^2 - 4*a*c)^(1/2) - b^2*c^2*(b^2 - 4*a*c)^(1/2) - 4*a*b*c^3 + 3*a*b^3*c -
2*a*b^2*c*(b^2 - 4*a*c)^(1/2)))/((4*a*c - b^2)*(2*a*c + a^2 - b^2 + c^2)^2)
 - (b*c*(2*a*b^4 - 20*a*c^4 + 3*a^4*c - 6*b^4*c + 7*c^5 - a^3*b^2 - 26*a^2*
c^3 + 4*a^3*c^2 + 23*a*b^2*c^2 - 6*a^2*b^2*c))/(4*(2*a*c + a^2 - b^2 + c^2)
^2) + (c*sin(x)*(64*a*c^5 + 26*c^6 + a^2*b^4 + 52*a^2*c^4 + 16*a^3*c^3 + 2*
a^4*c^2 - 18*b^2*c^4 + 9*b^4*c^2 - 32*a*b^2*c^3 - 4*a^3*b^2*c - 2*a^2*b^2*c
^2 - 2*a*b^4*c))/(4*(2*a*c + a^2 - b^2 + c^2)^2))*((b^4*(b^2 - 4*a*c)^(1/2)
)/2 - b^5/2 + c^4*(b^2 - 4*a*c)^(1/2) + b^3*c^2 + 2*a*c^3*(b^2 - 4*a*c)^(1/
2) - 4*a^2*b*c^2 + a^2*c^2*(b^2 - 4*a*c)^(1/2) - b^2*c^2*(b^2 - 4*a*c)^(1/2
) - 4*a*b*c^3 + 3*a*b^3*c - 2*a*b^2*c*(b^2 - 4*a*c)^(1/2)))/((4*a*c - b^2)*
```

```
(2*a*c + a^2 - b^2 + c^2)^2) - (b*c^5*sin(x))/(2*a*c + a^2 - b^2 + c^2)^2)*
(b^3*(3*a*c + c^2) - b^2*(c^2*(b^2 - 4*a*c)^(1/2) + 2*a*c*(b^2 - 4*a*c)^(1/
2)) - b*(4*a*c^3 + 4*a^2*c^2) - b^5/2 + (b^4*(b^2 - 4*a*c)^(1/2))/2 + c^4*(
b^2 - 4*a*c)^(1/2) + 2*a*c^3*(b^2 - 4*a*c)^(1/2) + a^2*c^2*(b^2 - 4*a*c)^(1
/2)))/(4*a*c^5 + 4*a^5*c - b^6 + 2*a^2*b^4 - a^4*b^2 + 16*a^2*c^4 + 24*a^3*
c^3 + 16*a^4*c^2 - b^2*c^4 + 2*b^4*c^2 - 12*a*b^2*c^3 - 12*a^3*b^2*c - 22*a
^2*b^2*c^2 + 8*a*b^4*c) - (log((c^4*(4*a*c + a^2 - 4*b^2 + 3*c^2))/(4*(2*a*
c + a^2 - b^2 + c^2)^2) - (((b*c*(2*a*b^4 - 20*a*c^4 + 3*a^4*c - 6*b^4*c +
7*c^5 - a^3*b^2 - 26*a^2*c^3 + 4*a^3*c^2 + 23*a*b^2*c^2 - 6*a^2*b^2*c))/(4*
(2*a*c + a^2 - b^2 + c^2)^2) + (((c*(a*b^4 + 28*a*c^4 + 4*a^4*c - 5*b^4*c +
 8*c^5 - a^3*b^2 + 36*a^2*c^3 + 20*a^3*c^2 + 5*b^2*c^3 - 3*a*b^2*c^2 - 9*a^
2*b^2*c))/(2*(2*a*c + a^2 - b^2 + c^2)) + (b*c*sin(x)*(36*a*c^3 + 4*a^3*c +
 3*b^4 + 16*c^4 - a^2*b^2 + 24*a^2*c^2 - 13*b^2*c^2 - 18*a*b^2*c))/(2*a*c +
 a^2 - b^2 + c^2) + (2*c*(b^5/2 + (b^4*(b^2 - 4*a*c)^(1/2))/2 + c^4*(b^2 -
4*a*c)^(1/2) - b^3*c^2 + 2*a*c^3*(b^2 - 4*a*c)^(1/2) + 4*a^2*b*c^2 + a^2*c^
2*(b^2 - 4*a*c)^(1/2) - b^2*c^2*(b^2 - 4*a*c)^(1/2) + 4*a*b*c^3 - 3*a*b^3*c
 - 2*a*b^2*c*(b^2 - 4*a*c)^(1/2))*(3*b^4*sin(x) + 4*c^4*sin(x) + 4*a*b^3 +
2*b*c^3 + 2*b^3*c + 4*a*c^3*sin(x) - 4*a^3*c*sin(x) + a^2*b^2*sin(x) - 4*a^
2*c^2*sin(x) - 3*b^2*c^2*sin(x) - 12*a*b*c^2 - 14*a^2*b*c - 10*a*b^2*c*sin(
x)))/((4*a*c - b^2)*(2*a*c + a^2 - b^2 + c^2)^2))*(b^5/2 + (b^4*(b^2 - 4*a*
c)^(1/2))/2 + c^4*(b^2 - 4*a*c)^(1/2) - b^3*c^2 + 2*a*c^3*(b^2 - 4*a*c)^(1/
2) + 4*a^2*b*c^2 + a^2*c^2*(b^2 - 4*a*c)^(1/2) - b^2*c^2*(b^2 - 4*a*c)^(1/2
) + 4*a*b*c^3 - 3*a*b^3*c - 2*a*b^2*c*(b^2 - 4*a*c)^(1/2)))/((4*a*c - b^2)*
(2*a*c + a^2 - b^2 + c^2)^2) - (c*sin(x)*(64*a*c^5 + 26*c^6 + a^2*b^4 + 52*
a^2*c^4 + 16*a^3*c^3 + 2*a^4*c^2 - 18*b^2*c^4 + 9*b^4*c^2 - 32*a*b^2*c^3 -
4*a^3*b^2*c - 2*a^2*b^2*c^2 - 2*a*b^4*c))/(4*(2*a*c + a^2 - b^2 + c^2)^2))*
(b^5/2 + (b^4*(b^2 - 4*a*c)^(1/2))/2 + c^4*(b^2 - 4*a*c)^(1/2) - b^3*c^2 +
2*a*c^3*(b^2 - 4*a*c)^(1/2) + 4*a^2*b*c^2 + a^2*c^2*(b^2 - 4*a*c)^(1/2) - b
^2*c^2*(b^2 - 4*a*c)^(1/2) + 4*a*b*c^3 - 3*a*b^3*c - 2*a*b^2*c*(b^2 - 4*a*c
)^(1/2)))/((4*a*c - b^2)*(2*a*c + a^2 - b^2 + c^2)^2) - (b*c^5*sin(x))/(2*a
*c + a^2 - b^2 + c^2)^2)*(b*(4*a*c^3 + 4*a^2*c^2) - b^3*(3*a*c + c^2) - b^2
*(c^2*(b^2 - 4*a*c)^(1/2) + 2*a*c*(b^2 - 4*a*c)^(1/2)) + b^5/2 + (b^4*(b^2
- 4*a*c)^(1/2))/2 + c^4*(b^2 - 4*a*c)^(1/2) + 2*a*c^3*(b^2 - 4*a*c)^(1/2) +
 a^2*c^2*(b^2 - 4*a*c)^(1/2)))/(4*a*c^5 + 4*a^5*c - b^6 + 2*a^2*b^4 - a^4*b
^2 + 16*a^2*c^4 + 24*a^3*c^3 + 16*a^4*c^2 - b^2*c^4 + 2*b^4*c^2 - 12*a*b^2*
c^3 - 12*a^3*b^2*c - 22*a^2*b^2*c^2 + 8*a*b^4*c)
```

3.15 $\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx$

Optimal result

Integrand size = 13, antiderivative size = 21

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx=\frac{1}{5}\log(2-\sin(x))-\frac{1}{5}\log(3+\sin(x))$$

```
[Out] 1/5*ln(2-sin(x))-1/5*ln(3+sin(x))
```

Rubi [A] (verified)

Time = 0.03 (sec) , antiderivative size = 21, normalized size of antiderivative = 1.00, number of steps used = 4, number of rules used = 3, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.231, Rules used = {3339, 630, 31}

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx=\frac{1}{5}\log(2-\sin(x))-\frac{1}{5}\log(\sin(x)+3)$$

```
[In] Int[Cos[x]/(-6 + Sin[x] + Sin[x]^2),x]
```

```
[Out] Log[2 - Sin[x]]/5 - Log[3 + Sin[x]]/5
```

Rule 31

```
Int[((a_) + (b_.)*(x_))^(-1), x_Symbol] :> Simp[Log[RemoveContent[a + b*x,
x]]/b, x] /; FreeQ[{a, b}, x]
```

Rule 630

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> With[{q = Rt[b^2
 - 4*a*c, 2]}, Dist[c/q, Int[1/Simp[b/2 - q/2 + c*x, x], x], x] - Dist[c/q,
 Int[1/Simp[b/2 + q/2 + c*x, x], x], x]] /; FreeQ[{a, b, c}, x] && NeQ[b^2
```

```
- 4*a*c, 0] && PosQ[b^2 - 4*a*c] && PerfectSquareQ[b^2 - 4*a*c]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}
\text{integral} &= \text{Subst}\left(\int \frac{1}{-6+x+x^2}\,dx, x, \sin(x)\right)\\
&= \frac{1}{5}\text{Subst}\left(\int \frac{1}{-2+x}\,dx, x, \sin(x)\right) - \frac{1}{5}\text{Subst}\left(\int \frac{1}{3+x}\,dx, x, \sin(x)\right)\\
&= \frac{1}{5}\log(2-\sin(x)) - \frac{1}{5}\log(3+\sin(x))
\end{aligned}$$

Mathematica [A] (verified)

Time = 0.01 (sec) , antiderivative size = 15, normalized size of antiderivative = 0.71

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx = -\frac{2}{5}\text{arctanh}\left(\frac{1}{5}(1+2\sin(x))\right)$$

```
[In] Integrate[Cos[x]/(-6 + Sin[x] + Sin[x]^2),x]
[Out] (-2*ArcTanh[(1 + 2*Sin[x])/5])/5
```

Maple [A] (verified)

Time = 0.50 (sec) , antiderivative size = 16, normalized size of antiderivative = 0.76

method	result	size
derivativedivides	$\frac{\ln(\sin(x)-2)}{5}-\frac{\ln(3+\sin(x))}{5}$	16
default	$\frac{\ln(\sin(x)-2)}{5}-\frac{\ln(3+\sin(x))}{5}$	16
parallelrisch	$\ln\left(\frac{3}{\left(\frac{486+162\sin(x)}{\cos(x)+1}\right)^{\frac{1}{5}}}\right)+\ln\left(\left(-\frac{\sin(x)-2}{\cos(x)+1}\right)^{\frac{1}{5}}\right)$	35
norman	$\frac{\ln\left(\tan^2\left(\frac{x}{2}\right)-\tan\left(\frac{x}{2}\right)+1\right)}{5}-\frac{\ln\left(3\left(\tan^2\left(\frac{x}{2}\right)\right)+2\tan\left(\frac{x}{2}\right)+3\right)}{5}$	38
risch	$\frac{\ln\left(-4i\mathrm{e}^{ix}+\mathrm{e}^{2ix}-1\right)}{5}-\frac{\ln\left(6i\mathrm{e}^{ix}+\mathrm{e}^{2ix}-1\right)}{5}$	38

```
[In] int(cos(x)/(-6+sin(x)+sin(x)^2),x,method=_RETURNVERBOSE)
[Out] 1/5*ln(sin(x)-2)-1/5*ln(3+sin(x))
```

Fricas [A] (verification not implemented)

none

Time = 0.27 (sec) , antiderivative size = 17, normalized size of antiderivative = 0.81

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx = -\frac{1}{5}\log\left(\sin\left(x\right)+3\right)+\frac{1}{5}\log\left(-\frac{1}{2}\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(-6+sin(x)+sin(x)^2),x, algorithm="fricas")
[Out] -1/5*log(sin(x) + 3) + 1/5*log(-1/2*sin(x) + 1)
```

Sympy [A] (verification not implemented)

Time = 0.09 (sec) , antiderivative size = 15, normalized size of antiderivative = 0.71

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx = \frac{\log\left(\sin\left(x\right)-2\right)}{5}-\frac{\log\left(\sin\left(x\right)+3\right)}{5}$$

```
[In] integrate(cos(x)/(-6+sin(x)+sin(x)**2),x)
[Out] log(sin(x) - 2)/5 - log(sin(x) + 3)/5
```

Maxima [A] (verification not implemented)

none

Time = 0.23 (sec) , antiderivative size = 15, normalized size of antiderivative = 0.71

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx = -\frac{1}{5}\,\log\left(\sin\left(x\right)+3\right)+\frac{1}{5}\,\log\left(\sin\left(x\right)-2\right)$$

```
[In] integrate(cos(x)/(-6+sin(x)+sin(x)^2),x, algorithm="maxima")
[Out] -1/5*log(sin(x) + 3) + 1/5*log(sin(x) - 2)
```

Giac [A] (verification not implemented)

none

Time = 0.34 (sec) , antiderivative size = 17, normalized size of antiderivative = 0.81

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx = -\frac{1}{5}\,\log\left(\sin\left(x\right)+3\right)+\frac{1}{5}\,\log\left(-\sin\left(x\right)+2\right)$$

```
[In] integrate(cos(x)/(-6+sin(x)+sin(x)^2),x, algorithm="giac")
[Out] -1/5*log(sin(x) + 3) + 1/5*log(-sin(x) + 2)
```

Mupad [B] (verification not implemented)

Time = 0.20 (sec) , antiderivative size = 9, normalized size of antiderivative = 0.43

$$\int \frac{\cos(x)}{-6+\sin(x)+\sin^2(x)}\,dx = -\frac{2\,\mathrm{atanh}\left(\frac{2\,\sin(x)}{5}+\frac{1}{5}\right)}{5}$$

```
[In] int(cos(x)/(sin(x) + sin(x)^2 - 6),x)
[Out] -(2*atanh((2*sin(x))/5 + 1/5))/5
```

3.16 $\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx$

Optimal result

Integrand size = 15, antiderivative size = 17

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = -\log(1-\sin(x))+\log(2-\sin(x))$$

```
[Out] -ln(1-sin(x))+ln(2-sin(x))
```

Rubi [A] (verified)

Time = 0.03 (sec) , antiderivative size = 17, normalized size of antiderivative = 1.00, number of steps used = 4, number of rules used = 3, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.200, Rules used = {3339, 630, 31}

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = \log(2-\sin(x))-\log(1-\sin(x))$$

```
[In] Int[Cos[x]/(2 - 3*Sin[x] + Sin[x]^2),x]
```

```
[Out] -Log[1 - Sin[x]] + Log[2 - Sin[x]]
```

Rule 31

```
Int[((a_) + (b_.)*(x_))^(-1), x_Symbol] :> Simp[Log[RemoveContent[a + b*x,
x]]/b, x] /; FreeQ[{a, b}, x]
```

Rule 630

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> With[{q = Rt[b^2
 - 4*a*c, 2]}, Dist[c/q, Int[1/Simp[b/2 - q/2 + c*x, x], x], x] - Dist[c/q,
 Int[1/Simp[b/2 + q/2 + c*x, x], x], x]] /; FreeQ[{a, b, c}, x] && NeQ[b^2
- 4*a*c, 0] && PosQ[b^2 - 4*a*c] && PerfectSquareQ[b^2 - 4*a*c]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}\text{integral} &= \text{Subst}\left(\int \frac{1}{2-3x+x^2}\,dx, x, \sin(x)\right)\\ &= \text{Subst}\left(\int \frac{1}{-2+x}\,dx, x, \sin(x)\right) - \text{Subst}\left(\int \frac{1}{-1+x}\,dx, x, \sin(x)\right)\\ &= -\log(1-\sin(x)) + \log(2-\sin(x))\end{aligned}$$

Mathematica [A] (verified)

Time = 0.11 (sec) , antiderivative size = 9, normalized size of antiderivative = 0.53

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = 2\text{arctanh}(3-2\sin(x))$$

```
[In] Integrate[Cos[x]/(2 - 3*Sin[x] + Sin[x]^2),x]
[Out] 2*ArcTanh[3 - 2*Sin[x]]
```

Maple [A] (verified)

Time = 0.40 (sec) , antiderivative size = 14, normalized size of antiderivative = 0.82

method	result	size
derivativedivides	$-\ln(\sin(x)-1)+\ln(\sin(x)-2)$	14
default	$-\ln(\sin(x)-1)+\ln(\sin(x)-2)$	14
norman	$-2\ln\left(\tan\left(\frac{x}{2}\right)-1\right)+\ln\left(\tan^2\left(\frac{x}{2}\right)-\tan\left(\frac{x}{2}\right)+1\right)$	26
parallelrisch	$-2\ln(-\cot(x)+\csc(x)-1)+\ln\left(\frac{2-\sin(x)}{\cos(x)+1}\right)$	27
risch	$-2\ln(e^{ix}-i)+\ln(-4ie^{ix}+e^{2ix}-1)$	29

```
[In] int(cos(x)/(2-3*sin(x)+sin(x)^2),x,method=_RETURNVERBOSE)
[Out] -ln(sin(x)-1)+ln(sin(x)-2)
```

Fricas [A] (verification not implemented)

none

Time = 0.28 (sec) , antiderivative size = 17, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = \log\left(-\frac{1}{2}\sin\left(x\right)+1\right) - \log\left(-\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(2-3*sin(x)+sin(x)^2),x, algorithm="fricas")
[Out] log(-1/2*sin(x) + 1) - log(-sin(x) + 1)
```

Sympy [A] (verification not implemented)

Time = 0.09 (sec) , antiderivative size = 12, normalized size of antiderivative = 0.71

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = \log\left(\sin\left(x\right)-2\right) - \log\left(\sin\left(x\right)-1\right)$$

```
[In] integrate(cos(x)/(2-3*sin(x)+sin(x)**2),x)
[Out] log(sin(x) - 2) - log(sin(x) - 1)
```

Maxima [A] (verification not implemented)

none

Time = 0.25 (sec) , antiderivative size = 13, normalized size of antiderivative = 0.76

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = -\log\left(\sin\left(x\right)-1\right) + \log\left(\sin\left(x\right)-2\right)$$

```
[In] integrate(cos(x)/(2-3*sin(x)+sin(x)^2),x, algorithm="maxima")
[Out] -log(sin(x) - 1) + log(sin(x) - 2)
```

Giac [A] (verification not implemented)

none

Time = 0.38 (sec) , antiderivative size = 17, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = \log\left(-\sin\left(x\right)+2\right) - \log\left(-\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(2-3*sin(x)+sin(x)^2),x, algorithm="giac")
[Out] log(-sin(x) + 2) - log(-sin(x) + 1)
```

Mupad [B] (verification not implemented)

Time = 15.22 (sec) , antiderivative size = 9, normalized size of antiderivative = 0.53

$$\int \frac{\cos(x)}{2-3\sin(x)+\sin^2(x)}\,dx = -2\,\mathrm{atanh}(2\,\sin{(x)}-3)$$

```
[In] int(cos(x)/(sin(x)^2 - 3*sin(x) + 2),x)
[Out] -2*atanh(2*sin(x) - 3)
```

3.17 $\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx$

Optimal result

Integrand size = 15, antiderivative size = 21

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx = \frac{1}{6}\log(1-\sin(x)) - \frac{1}{6}\log(5+\sin(x))$$

```
[Out] 1/6*ln(1-sin(x))-1/6*ln(5+sin(x))
```

Rubi [A] (verified)

Time = 0.03 (sec) , antiderivative size = 21, normalized size of antiderivative = 1.00, number of steps used = 4, number of rules used = 3, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.200, Rules used = {3339, 630, 31}

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx = \frac{1}{6}\log(1-\sin(x)) - \frac{1}{6}\log(\sin(x)+5)$$

```
[In] Int[Cos[x]/(-5 + 4*Sin[x] + Sin[x]^2),x]
```

```
[Out] Log[1 - Sin[x]]/6 - Log[5 + Sin[x]]/6
```

Rule 31

```
Int[((a_) + (b_.)*(x_))^(-1), x_Symbol] :> Simp[Log[RemoveContent[a + b*x,
x]]/b, x] /; FreeQ[{a, b}, x]
```

Rule 630

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> With[{q = Rt[b^2
 - 4*a*c, 2]}, Dist[c/q, Int[1/Simp[b/2 - q/2 + c*x, x], x], x] - Dist[c/q,
 Int[1/Simp[b/2 + q/2 + c*x, x], x], x]] /; FreeQ[{a, b, c}, x] && NeQ[b^2
```

```
- 4*a*c, 0] && PosQ[b^2 - 4*a*c] && PerfectSquareQ[b^2 - 4*a*c]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}\text{integral} &= \text{Subst}\left(\int \frac{1}{-5+4x+x^2}\,dx, x, \sin(x)\right)\\ &= \frac{1}{6}\text{Subst}\left(\int \frac{1}{-1+x}\,dx, x, \sin(x)\right) - \frac{1}{6}\text{Subst}\left(\int \frac{1}{5+x}\,dx, x, \sin(x)\right)\\ &= \frac{1}{6}\log(1-\sin(x)) - \frac{1}{6}\log(5+\sin(x))\end{aligned}$$

Mathematica [A] (verified)

Time = 0.10 (sec) , antiderivative size = 15, normalized size of antiderivative = 0.71

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx = -\frac{1}{3}\text{arctanh}\left(\frac{1}{6}(4+2\sin(x))\right)$$

```
[In] Integrate[Cos[x]/(-5 + 4*Sin[x] + Sin[x]^2),x]
[Out] -1/3*ArcTanh[(4 + 2*Sin[x])/6]
```

Maple [A] (verified)

Time = 0.49 (sec) , antiderivative size = 16, normalized size of antiderivative = 0.76

method	result	size
derivativedivides	$-\frac{\ln(5+\sin(x))}{6}+\frac{\ln(\sin(x)-1)}{6}$	16
default	$-\frac{\ln(5+\sin(x))}{6}+\frac{\ln(\sin(x)-1)}{6}$	16
norman	$\frac{\ln(\tan(\frac{x}{2})-1)}{3}-\frac{\ln(5(\tan^2(\frac{x}{2}))+2\tan(\frac{x}{2})+5)}{6}$	30
risch	$\frac{\ln(e^{ix}-i)}{3}-\frac{\ln(10ie^{ix}+e^{2ix}-1)}{6}$	31
parallelrisch	$\ln\left(\frac{(-\cot(x)+\csc(x)-1)^{\frac{1}{3}}160^{\frac{1}{6}}}{2}\right)+\ln\left(\frac{1}{\left(\frac{5+\sin(x)}{\cos(x)+1}\right)^{\frac{1}{6}}}\right)$	32

```
[In] int(cos(x)/(-5+4*sin(x)+sin(x)^2),x,method=_RETURNVERBOSE)
[Out] -1/6*ln(5+sin(x))+1/6*ln(sin(x)-1)
```

Fricas [A] (verification not implemented)

none

Time = 0.31 (sec) , antiderivative size = 17, normalized size of antiderivative = 0.81

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx=-\frac{1}{6}\log\left(\sin\left(x\right)+5\right)+\frac{1}{6}\log\left(-\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(-5+4*sin(x)+sin(x)^2),x, algorithm="fricas")
[Out] -1/6*log(sin(x) + 5) + 1/6*log(-sin(x) + 1)
```

Sympy [A] (verification not implemented)

Time = 0.09 (sec) , antiderivative size = 15, normalized size of antiderivative = 0.71

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx=\frac{\log\left(\sin\left(x\right)-1\right)}{6}-\frac{\log\left(\sin\left(x\right)+5\right)}{6}$$

```
[In] integrate(cos(x)/(-5+4*sin(x)+sin(x)**2),x)
[Out] log(sin(x) - 1)/6 - log(sin(x) + 5)/6
```

Maxima [A] (verification not implemented)

none

Time = 0.24 (sec) , antiderivative size = 15, normalized size of antiderivative = 0.71

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx = -\frac{1}{6}\,\log\left(\sin\left(x\right)+5\right)+\frac{1}{6}\,\log\left(\sin\left(x\right)-1\right)$$

```
[In] integrate(cos(x)/(-5+4*sin(x)+sin(x)^2),x, algorithm="maxima")
[Out] -1/6*log(sin(x) + 5) + 1/6*log(sin(x) - 1)
```

Giac [A] (verification not implemented)

none

Time = 0.35 (sec) , antiderivative size = 17, normalized size of antiderivative = 0.81

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx = -\frac{1}{6}\,\log\left(\sin\left(x\right)+5\right)+\frac{1}{6}\,\log\left(-\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(-5+4*sin(x)+sin(x)^2),x, algorithm="giac")
[Out] -1/6*log(sin(x) + 5) + 1/6*log(-sin(x) + 1)
```

Mupad [B] (verification not implemented)

Time = 0.12 (sec) , antiderivative size = 9, normalized size of antiderivative = 0.43

$$\int \frac{\cos(x)}{-5+4\sin(x)+\sin^2(x)}\,dx = -\frac{\mathrm{atanh}\left(\frac{\sin(x)}{3}+\frac{2}{3}\right)}{3}$$

```
[In] int(cos(x)/(4*sin(x) + sin(x)^2 - 5),x)
[Out] -atanh(sin(x)/3 + 2/3)/3
```

3.18 $\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx$

Optimal result

Integrand size = 15, antiderivative size = 9

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = -\arctan(3-\sin(x))$$

```
[Out] arctan(-3+sin(x))
```

Rubi [A] (verified)

Time = 0.03 (sec) , antiderivative size = 9, normalized size of antiderivative = 1.00, number of steps used = 3, number of rules used = 3, $\frac{\text{number of rules}}{\text{integrand size}} = 0.200$, Rules used = {3339, 632, 210}

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = -\arctan(3-\sin(x))$$

```
[In] Int[Cos[x]/(10 - 6*Sin[x] + Sin[x]^2),x]
```

```
[Out] -ArcTan[3 - Sin[x]]
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 632

```
Int[((a_.) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> Dist[-2, Subst[I
nt[1/Simp[b^2 - 4*a*c - x^2, x], x], x, b + 2*c*x], x] /; FreeQ[{a, b, c},
x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}\text{integral} &= \text{Subst}\left(\int \frac{1}{10-6x+x^2}\,dx, x, \sin(x)\right)\\ &= -\left(2\text{Subst}\left(\int \frac{1}{-4-x^2}\,dx, x, -6+2\sin(x)\right)\right)\\ &= -\arctan(3-\sin(x))\end{aligned}$$

Mathematica [A] (verified)

Time = 0.01 (sec) , antiderivative size = 9, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = -\arctan(3-\sin(x))$$

```
[In] Integrate[Cos[x]/(10 - 6*Sin[x] + Sin[x]^2),x]
[Out] -ArcTan[3 - Sin[x]]
```

Maple [A] (verified)

Time = 2.26 (sec) , antiderivative size = 6, normalized size of antiderivative = 0.67

method	result	size
derivativedivides	$\arctan\left(-3+\sin\left(x\right)\right)$	6
default	$\arctan\left(-3+\sin\left(x\right)\right)$	6
risch	$-\frac{i\ln\left(e^{2ix}+(2-6i)e^{ix}-1\right)}{2}+\frac{i\ln\left(e^{2ix}+(-2-6i)e^{ix}-1\right)}{2}$	42
parallelrisch	$\frac{i\left(\ln\left(\frac{(-3-i)(\sin(x)-3+i)}{5\cos(x)+5}\right)-\ln\left(\frac{10+(-3+i)\sin(x)}{5\cos(x)+5}\right)\right)}{2}$	44

```
[In] int(cos(x)/(10-6*sin(x)+sin(x)^2),x,method=_RETURNVERBOSE)
[Out] arctan(-3+sin(x))
```

Fricas [A] (verification not implemented)

none

Time = 0.28 (sec) , antiderivative size = 5, normalized size of antiderivative = 0.56

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = \arctan\left(\sin\left(x\right)-3\right)$$

```
[In] integrate(cos(x)/(10-6*sin(x)+sin(x)^2),x, algorithm="fricas")
[Out] arctan(sin(x) - 3)
```

Sympy [A] (verification not implemented)

Time = 0.11 (sec) , antiderivative size = 5, normalized size of antiderivative = 0.56

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = \operatorname{atan}\left(\sin\left(x\right)-3\right)$$

```
[In] integrate(cos(x)/(10-6*sin(x)+sin(x)**2),x)
[Out] atan(sin(x) - 3)
```

Maxima [A] (verification not implemented)

none

Time = 0.31 (sec) , antiderivative size = 5, normalized size of antiderivative = 0.56

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = \arctan\left(\sin\left(x\right)-3\right)$$

```
[In] integrate(cos(x)/(10-6*sin(x)+sin(x)^2),x, algorithm="maxima")
[Out] arctan(sin(x) - 3)
```

Giac [A] (verification not implemented)

none

Time = 0.31 (sec) , antiderivative size = 5, normalized size of antiderivative = 0.56

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = \arctan\left(\sin\left(x\right)-3\right)$$

```
[In] integrate(cos(x)/(10-6*sin(x)+sin(x)^2),x, algorithm="giac")
[Out] arctan(sin(x) - 3)
```

Mupad [B] (verification not implemented)

Time = 0.08 (sec) , antiderivative size = 5, normalized size of antiderivative = 0.56

$$\int \frac{\cos(x)}{10-6\sin(x)+\sin^2(x)}\,dx = \operatorname{atan}(\sin{(x)}-3)$$

```
[In] int(cos(x)/(sin(x)^2 - 6*sin(x) + 10),x)
[Out] atan(sin(x) - 3)
```

3.19 $\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx$

Optimal result

Integrand size = 15, antiderivative size = 5

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \arctan(1+\sin(x))$$

```
[Out] arctan(1+sin(x))
```

Rubi [A] (verified)

Time = 0.03 (sec) , antiderivative size = 5, normalized size of antiderivative = 1.00, number of steps used = 3, number of rules used = 3, $\frac{\text{number of rules}}{\text{integrand size}}$ = 0.200, Rules used = {3339, 631, 210}

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \arctan(\sin(x)+1)$$

```
[In] Int[Cos[x]/(2 + 2*Sin[x] + Sin[x]^2),x]
```

```
[Out] ArcTan[1 + Sin[x]]
```

Rule 210

```
Int[((a_) + (b_.)*(x_)^2)^(-1), x_Symbol] :> Simp[(-(Rt[-a, 2]*Rt[-b, 2])^(
-1))*ArcTan[Rt[-b, 2]*(x/Rt[-a, 2])], x] /; FreeQ[{a, b}, x] && PosQ[a/b] &
& (LtQ[a, 0] || LtQ[b, 0])
```

Rule 631

```
Int[((a_) + (b_.)*(x_) + (c_.)*(x_)^2)^(-1), x_Symbol] :> With[{q = 1 - 4*S
implify[a*(c/b^2)]}, Dist[-2/b, Subst[Int[1/(q - x^2), x], x, 1 + 2*c*(x/b)
], x] /; RationalQ[q] && (EqQ[q^2, 1] ||  !RationalQ[b^2 - 4*a*c])] /; Free
```

```
Q[{a, b, c}, x] && NeQ[b^2 - 4*a*c, 0]
```

Rule 3339

```
Int[cos[(d_.) + (e_.)*(x_)]^(m_.)*((a_.) + (b_.)*((f_.)*sin[(d_.) + (e_.)*(
x_)])^(n_.) + (c_.)*((f_.)*sin[(d_.) + (e_.)*(x_)])^(n2_.))^(p_.), x_Symbol
] :> Module[{g = FreeFactors[Sin[d + e*x], x]}, Dist[g/e, Subst[Int[(1 - g^
2*x^2)^((m - 1)/2)*(a + b*(f*g*x)^n + c*(f*g*x)^(2*n))^p, x], x, Sin[d + e*
x]/g], x]] /; FreeQ[{a, b, c, d, e, f, n, p}, x] && EqQ[n2, 2*n] && Integer
Q[(m - 1)/2]
```

Rubi steps

$$\begin{aligned}\text{integral} &= \text{Subst}\left(\int \frac{1}{2+2x+x^2}\,dx, x, \sin(x)\right)\\ &= -\text{Subst}\left(\int \frac{1}{-1-x^2}\,dx, x, 1+\sin(x)\right)\\ &= \arctan(1+\sin(x))\end{aligned}$$

Mathematica [A] (verified)

Time = 0.01 (sec) , antiderivative size = 5, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \arctan(1+\sin(x))$$

```
[In] Integrate[Cos[x]/(2 + 2*Sin[x] + Sin[x]^2),x]
[Out] ArcTan[1 + Sin[x]]
```

Maple [A] (verified)

Time = 1.74 (sec) , antiderivative size = 6, normalized size of antiderivative = 1.20

method	result	size
derivativedivides	$\arctan\left(1+\sin\left(x\right)\right)$	6
default	$\arctan\left(1+\sin\left(x\right)\right)$	6
parallelrisch	$\frac{i\left(\ln\left(\frac{(1-i)(\sin(x)+1+i)}{\cos(x)+1}\right)-\ln\left(\frac{2+(1+i)\sin(x)}{\cos(x)+1}\right)\right)}{2}$	40
risch	$-\frac{i\ln\left(e^{2ix}+(2+2i)e^{ix}-1\right)}{2}+\frac{i\ln\left(e^{2ix}+(-2+2i)e^{ix}-1\right)}{2}$	42

```
[In] int(cos(x)/(2+2*sin(x)+sin(x)^2),x,method=_RETURNVERBOSE)
```

```
[Out] arctan(1+sin(x))
```

Fricas [A] (verification not implemented)

none

Time = 0.27 (sec) , antiderivative size = 5, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \arctan\left(\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(2+2*sin(x)+sin(x)^2),x, algorithm="fricas")
[Out] arctan(sin(x) + 1)
```

Sympy [A] (verification not implemented)

Time = 0.10 (sec) , antiderivative size = 5, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \operatorname{atan}\left(\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(2+2*sin(x)+sin(x)**2),x)
[Out] atan(sin(x) + 1)
```

Maxima [A] (verification not implemented)

none

Time = 0.36 (sec) , antiderivative size = 5, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \arctan\left(\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(2+2*sin(x)+sin(x)^2),x, algorithm="maxima")
[Out] arctan(sin(x) + 1)
```

Giac [A] (verification not implemented)

none

Time = 0.29 (sec) , antiderivative size = 5, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \arctan\left(\sin\left(x\right)+1\right)$$

```
[In] integrate(cos(x)/(2+2*sin(x)+sin(x)^2),x, algorithm="giac")
[Out] arctan(sin(x) + 1)
```

Mupad [B] (verification not implemented)

Time = 0.07 (sec) , antiderivative size = 5, normalized size of antiderivative = 1.00

$$\int \frac{\cos(x)}{2+2\sin(x)+\sin^2(x)}\,dx = \mathrm{atan}(\sin\left(x\right)+1)$$

```
[In] int(cos(x)/(2*sin(x) + sin(x)^2 + 2),x)
[Out] atan(sin(x) + 1)
```

CHAPTER 4

APPENDIX

4.1 Listing of Grading functions

The following are the current version of the grading functions used for grading the quality of the antiderivative with reference to the optimal antiderivative included in the test suite.

There is a version for Maple and for Mathematica/Rubi. There is a version for grading Sympy and version for use with Sagemath.

The following are links to the current source code.

The following are the listings of source code of the grading functions.

Mathematica and Rubi grading function

```
(* Original version thanks to Albert Rich emailed on 03/21/2017 *)
(* ::Package:: *)

(* Nasser: April 7,2022. add second output which gives reason for the grade *)
(*                       Small rewrite of logic in main function to make it*)
(*                       match Maple's logic. No change in functionality otherwise*)

(* ::Subsection:: *)
(*GradeAntiderivative[result,optimal]*)


(* ::Text:: *)
(*If result and optimal are mathematical expressions, *)
```

```
(*          GradeAntiderivative[result,optimal] returns*)
(*  "F" if the result fails to integrate an expression that*)
(*      is integrable*)
(*  "C" if result involves higher level functions than necessary*)
(*  "B" if result is more than twice the size of the optimal*)
(*      antiderivative*)
(*  "A" if result can be considered optimal*)


GradeAntiderivative[result_,optimal_] := Module[{expnResult,expnOptimal,leafCountResult,leafCou
    expnResult  = ExpnType[result];
    expnOptimal = ExpnType[optimal];
    leafCountResult = LeafCount[result];
    leafCountOptimal = LeafCount[optimal];

    (*Print["expnResult=",expnResult," expnOptimal=",expnOptimal];*)
    If[expnResult<=expnOptimal,
        If[Not[FreeQ[result,Complex]], (*result contains complex*)
            If[Not[FreeQ[optimal,Complex]], (*optimal contains complex*)
                If[leafCountResult<=2*leafCountOptimal,
                    finalresult={"A",""}
                    ,(*ELSE*)
                    finalresult={"B","Both result and optimal contain complex but leaf count is
                  ]
            ,(*ELSE*)
                finalresult={"C","Result contains complex when optimal does not."}
            ]
        ,(*ELSE*)(*result does not contains complex*)
            If[leafCountResult<=2*leafCountOptimal,
               finalresult={"A",""}
            ,(*ELSE*)
               finalresult={"B","Leaf count is larger than twice the leaf count of optimal. $"<
              ]
        ]
    ,(*ELSE*) (*expnResult>expnOptimal*)
        If[FreeQ[result,Integrate] && FreeQ[result,Int],
            finalresult={"C","Result contains higher order function than in optimal. Order "<>T
            ,
            finalresult={"F","Contains unresolved integral."}
        ]
    ];

    finalresult
]

(* ::Text:: *)
(*The following summarizes the type number assigned an *)
```

```
(*expression based on the functions it involves*)
(*1 = rational function*)
(*2 = algebraic function*)
(*3 = elementary function*)
(*4 = special function*)
(*5 = hyperpergeometric function*)
(*6 = appell function*)
(*7 = rootsum function*)
(*8 = integrate function*)
(*9 = unknown function*)


ExpnType[expn_] :=
  If[AtomQ[expn],
    1,
  If[ListQ[expn],
    Max[Map[ExpnType,expn]],
  If[Head[expn]===Power,
    If[IntegerQ[expn[[2]]],
      ExpnType[expn[[1]]],
    If[Head[expn[[2]]]===Rational,
      If[IntegerQ[expn[[1]]] || Head[expn[[1]]]===Rational,
        1,
      Max[ExpnType[expn[[1]]],2]],
    Max[ExpnType[expn[[1]]],ExpnType[expn[[2]]],3]]],
  If[Head[expn]===Plus || Head[expn]===Times,
    Max[ExpnType[First[expn]],ExpnType[Rest[expn]]],
  If[ElementaryFunctionQ[Head[expn]],
    Max[3,ExpnType[expn[[1]]]],
  If[SpecialFunctionQ[Head[expn]],
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],4]],
  If[HypergeometricFunctionQ[Head[expn]],
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],5]],
  If[AppellFunctionQ[Head[expn]],
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],6]],
  If[Head[expn]===RootSum,
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],7]],
  If[Head[expn]===Integrate || Head[expn]===Int,
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],8]],
  9]]]]]]]]]]]


ElementaryFunctionQ[func_] :=
  MemberQ[{
  Exp,Log,
  Sin,Cos,Tan,Cot,Sec,Csc,
  ArcSin,ArcCos,ArcTan,ArcCot,ArcSec,ArcCsc,
```

```
  Sinh,Cosh,Tanh,Coth,Sech,Csch,
  ArcSinh,ArcCosh,ArcTanh,ArcCoth,ArcSech,ArcCsch
},func]


SpecialFunctionQ[func_] :=
  MemberQ[{
  Erf, Erfc, Erfi,
  FresnelS, FresnelC,
  ExpIntegralE, ExpIntegralEi, LogIntegral,
  SinIntegral, CosIntegral, SinhIntegral, CoshIntegral,
  Gamma, LogGamma, PolyGamma,
  Zeta, PolyLog, ProductLog,
  EllipticF, EllipticE, EllipticPi
},func]


HypergeometricFunctionQ[func_] :=
  MemberQ[{Hypergeometric1F1,Hypergeometric2F1,HypergeometricPFQ},func]


AppellFunctionQ[func_] :=
  MemberQ[{AppellF1},func]
```

Maple grading function

```
# File: GradeAntiderivative.mpl
# Original version thanks to Albert Rich emailed on 03/21/2017

#Nasser 03/22/2017  Use Maple leaf count instead since buildin
#Nasser 03/23/2017  missing 'ln' for ElementaryFunctionQ added
#Nasser 03/24/2017  corrected the check for complex result
#Nasser 10/27/2017  check for leafsize and do not call ExpnType()
#                   if leaf size is "too large". Set at 500,000
#Nasser 12/22/2019  Added debug flag, added 'dilog' to special functions
#                   see problem 156, file Apostol_Problems
#Nasser 4/07/2022   add second output which gives reason for the grade

GradeAntiderivative := proc(result,optimal)
local leaf_count_result,
        leaf_count_optimal,
        ExpnType_result,
        ExpnType_optimal,
        debug:=false;

        leaf_count_result:=leafcount(result);
```

```
        #do NOT call ExpnType() if leaf size is too large. Recursion problem
        if leaf_count_result > 500000 then
            return "B","result has leaf size over 500,000. Avoiding possible recursion issues
        fi;

        leaf_count_optimal := leafcount(optimal);
        ExpnType_result  := ExpnType(result);
        ExpnType_optimal := ExpnType(optimal);

        if debug then
            print("ExpnType_result",ExpnType_result," ExpnType_optimal=",ExpnType_optimal);
        fi;

# If result and optimal are mathematical expressions,
#  GradeAntiderivative[result,optimal] returns
#   "F" if the result fails to integrate an expression that
#       is integrable
#   "C" if result involves higher level functions than necessary
#   "B" if result is more than twice the size of the optimal
#       antiderivative
#   "A" if result can be considered optimal

    #This check below actually is not needed, since I only
    #call this grading only for passed integrals. i.e. I check
    #for "F" before calling this. But no harm of keeping it here.
    #just in case.


    if not type(result,freeof('int')) then
        return "F","Result contains unresolved integral";
    fi;


    if ExpnType_result<=ExpnType_optimal then
        if debug then
             print("ExpnType_result<=ExpnType_optimal");
        fi;
        if is_contains_complex(result) then
            if is_contains_complex(optimal) then
                if debug then
                        print("both result and optimal complex");
                fi;
                if leaf_count_result<=2*leaf_count_optimal then
                    return "A"," ";
                else
                   return "B",cat("Both result and optimal contain complex but leaf count of
                                convert(leaf_count_result,string)," vs. $2 (",
```

```
                                    convert(leaf_count_optimal,string)," ) = ",convert(2*leaf_co
                end if
            else #result contains complex but optimal is not
                if debug then
                        print("result contains complex but optimal is not");
                fi;
                return "C","Result contains complex when optimal does not.";
            fi;
        else # result do not contain complex
             # this assumes optimal do not as well. No check is needed here.
            if debug then
                   print("result do not contain complex, this assumes optimal do not as well")
            fi;
            if leaf_count_result<=2*leaf_count_optimal then
                if debug then
                    print("leaf_count_result<=2*leaf_count_optimal");
                fi;
                return "A"," ";
            else
                if debug then
                    print("leaf_count_result>2*leaf_count_optimal");
                fi;
                return "B",cat("Leaf count of result is larger than twice the leaf count of opt
                                    convert(leaf_count_result,string),"$ vs. $2(",
                                    convert(leaf_count_optimal,string),")=",convert(2*leaf_count
            fi;
        fi;
    else #ExpnType(result) > ExpnType(optimal)
        if debug then
            print("ExpnType(result) > ExpnType(optimal)");
        fi;
        return "C",cat("Result contains higher order function than in optimal. Order ",
                        convert(ExpnType_result,string)," vs. order ",
                        convert(ExpnType_optimal,string),".");
    fi;

end proc:

#
# is_contains_complex(result)
# takes expressions and returns true if it contains "I" else false
#
#Nasser 032417
is_contains_complex:= proc(expression)
  return (has(expression,I));
end proc:
```

```
# The following summarizes the type number assigned an expression
# based on the functions it involves
# 1 = rational function
# 2 = algebraic function
# 3 = elementary function
# 4 = special function
# 5 = hyperpergeometric function
# 6 = appell function
# 7 = rootsum function
# 8 = integrate function
# 9 = unknown function

ExpnType := proc(expn)
  if type(expn,'atomic') then
    1
  elif type(expn,'list') then
    apply(max,map(ExpnType,expn))
  elif type(expn,'sqrt') then
    if type(op(1,expn),'rational') then
       1
    else
       max(2,ExpnType(op(1,expn)))
    end if
  elif type(expn,'`^`') then
    if type(op(2,expn),'integer') then
      ExpnType(op(1,expn))
    elif type(op(2,expn),'rational') then
      if type(op(1,expn),'rational') then
         1
      else
         max(2,ExpnType(op(1,expn)))
      end if
    else
         max(3,ExpnType(op(1,expn)),ExpnType(op(2,expn)))
    end if
  elif type(expn,'`+`') or type(expn,'`*`') then
    max(ExpnType(op(1,expn)),max(ExpnType(rest(expn))))
  elif ElementaryFunctionQ(op(0,expn)) then
    max(3,ExpnType(op(1,expn)))
  elif SpecialFunctionQ(op(0,expn)) then
    max(4,apply(max,map(ExpnType,[op(expn)])))
  elif HypergeometricFunctionQ(op(0,expn)) then
    max(5,apply(max,map(ExpnType,[op(expn)])))
  elif AppellFunctionQ(op(0,expn)) then
    max(6,apply(max,map(ExpnType,[op(expn)])))
  elif op(0,expn)='int' then
    max(8,apply(max,map(ExpnType,[op(expn)]))) else
```

```
  9
  end if
end proc:


ElementaryFunctionQ := proc(func)
  member(func,[
        exp,log,ln,
        sin,cos,tan,cot,sec,csc,
        arcsin,arccos,arctan,arccot,arcsec,arccsc,
        sinh,cosh,tanh,coth,sech,csch,
        arcsinh,arccosh,arctanh,arccoth,arcsech,arccsch])
end proc:

SpecialFunctionQ := proc(func)
  member(func,[
        erf,erfc,erfi,
        FresnelS,FresnelC,
        Ei,Ei,Li,Si,Ci,Shi,Chi,
        GAMMA,lnGAMMA,Psi,Zeta,polylog,dilog,LambertW,
        EllipticF,EllipticE,EllipticPi])
end proc:

HypergeometricFunctionQ := proc(func)
  member(func,[Hypergeometric1F1,hypergeom,HypergeometricPFQ])
end proc:

AppellFunctionQ := proc(func)
  member(func,[AppellF1])
end proc:

# u is a sum or product.  rest(u) returns all but the
# first term or factor of u.
rest := proc(u) local v;
  if nops(u)=2 then
     op(2,u)
  else
     apply(op(0,u),op(2..nops(u),u))
  end if
end proc:

#leafcount(u) returns the number of nodes in u.
#Nasser 3/23/17  Replaced by build-in leafCount from package in Maple
leafcount := proc(u)
   MmaTranslator[Mma][LeafCount](u);
end proc:
```

Sympy grading function

```
#Dec 24, 2019. Nasser M. Abbasi:
#             Port of original Maple grading function by
#             Albert Rich to use with Sympy/Python
#Dec 27, 2019  Nasser. Added `RootSum`. See problem 177, Timofeev file
#             added 'exp_polar'
from sympy import *

def leaf_count(expr):
    #sympy do not have leaf count function. This is approximation
    return round(1.7*count_ops(expr))

def is_sqrt(expr):
    if isinstance(expr,Pow):
        if expr.args[1] == Rational(1,2):
            return True
        else:
            return False
    else:
        return False

def is_elementary_function(func):
    return func in [exp,log,ln,sin,cos,tan,cot,sec,csc,
            asin,acos,atan,acot,asec,acsc,sinh,cosh,tanh,coth,sech,csch,
            asinh,acosh,atanh,acoth,asech,acsch
        ]

def is_special_function(func):
    return func in [ erf,erfc,erfi,
             fresnels,fresnelc,Ei,Ei,Li,Si,Ci,Shi,Chi,
             gamma,loggamma,digamma,zeta,polylog,LambertW,
             elliptic_f,elliptic_e,elliptic_pi,exp_polar
         ]

def is_hypergeometric_function(func):
    return func in [hyper]

def is_appell_function(func):
    return func in [appellf1]

def is_atom(expn):
    try:
        if expn.isAtom or isinstance(expn,int) or isinstance(expn,float):
          return True
        else:
          return False
```

```
    except AttributeError as error:
        return False

def expnType(expn):
    debug=False
    if debug:
        print("expn=",expn,"type(expn)=",type(expn))

    if is_atom(expn):
       return 1
    elif isinstance(expn,list):
        return max(map(expnType, expn))    #apply(max,map(ExpnType,expn))
    elif  is_sqrt(expn):
        if isinstance(expn.args[0],Rational): #type(op(1,expn),'rational')
            return 1
        else:
            return max(2,expnType(expn.args[0]))  #max(2,ExpnType(op(1,expn)))
    elif isinstance(expn,Pow):   #type(expn,'`^`')
        if isinstance(expn.args[1],Integer):  #type(op(2,expn),'integer')
            return expnType(expn.args[0])   #ExpnType(op(1,expn))
        elif isinstance(expn.args[1],Rational):  #type(op(2,expn),'rational')
            if isinstance(expn.args[0],Rational): #type(op(1,expn),'rational')
                return 1
            else:
                return max(2,expnType(expn.args[0]))  #max(2,ExpnType(op(1,expn)))
        else:
            return max(3,expnType(expn.args[0]),expnType(expn.args[1])) #max(3,ExpnType(op(1,expn)),ExpnTy
    elif isinstance(expn,Add) or isinstance(expn,Mul): #type(expn,'`+`') or type(expn,'`*`')
        m1 = expnType(expn.args[0])
        m2 = expnType(list(expn.args[1:]))
        return max(m1,m2)  #max(ExpnType(op(1,expn)),max(ExpnType(rest(expn))))
    elif is_elementary_function(expn.func):  #ElementaryFunctionQ(op(0,expn))
        return max(3,expnType(expn.args[0]))  #max(3,ExpnType(op(1,expn)))
    elif is_special_function(expn.func): #SpecialFunctionQ(op(0,expn))
        m1 = max(map(expnType, list(expn.args)))
        return max(4,m1)   #max(4,apply(max,map(ExpnType,[op(expn)])))
    elif is_hypergeometric_function(expn.func): #HypergeometricFunctionQ(op(0,expn))
        m1 = max(map(expnType, list(expn.args)))
        return max(5,m1)   #max(5,apply(max,map(ExpnType,[op(expn)])))
    elif is_appell_function(expn.func):
        m1 = max(map(expnType, list(expn.args)))
        return max(6,m1)   #max(5,apply(max,map(ExpnType,[op(expn)])))
    elif isinstance(expn,RootSum):
        m1 = max(map(expnType, list(expn.args))) #Apply[Max,Append[Map[ExpnType,Apply[List,expn]],7]],
        return max(7,m1)
    elif str(expn).find("Integral") != -1:
```

```
        m1 = max(map(expnType, list(expn.args)))
        return max(8,m1)    #max(5,apply(max,map(ExpnType,[op(expn)])))
    else:
        return 9

#main function
def grade_antiderivative(result,optimal):

    #print ("Enter grade_antiderivative for sagemath")
    #print("Enter grade_antiderivative, result=",result," optimal=",optimal)

    leaf_count_result  = leaf_count(result)
    leaf_count_optimal = leaf_count(optimal)

    #print("leaf_count_result=",leaf_count_result)
    #print("leaf_count_optimal=",leaf_count_optimal)

    expnType_result  = expnType(result)
    expnType_optimal = expnType(optimal)

    if str(result).find("Integral") != -1:
        grade = "F"
        grade_annotation =""
    else:
        if expnType_result <= expnType_optimal:
            if result.has(I):
                if optimal.has(I): #both result and optimal complex
                    if leaf_count_result <= 2*leaf_count_optimal:
                        grade = "A"
                        grade_annotation =""
                    else:
                        grade = "B"
                        grade_annotation ="Both result and optimal contain complex but leaf count of result is large
                else: #result contains complex but optimal is not
                    grade = "C"
                    grade_annotation ="Result contains complex when optimal does not."
            else: # result do not contain complex, this assumes optimal do not as well
                if leaf_count_result <= 2*leaf_count_optimal:
                    grade = "A"
                    grade_annotation =""
                else:
                    grade = "B"
                    grade_annotation ="Leaf count of result is larger than twice the leaf count of optimal. "+str(le
        else:
            grade = "C"
            grade_annotation ="Result contains higher order function than in optimal. Order "+str(ExpnType_
```

```
    #print("Before returning. grade=",grade, " grade_annotation=",grade_annotation)

    return grade, grade_annotation
```

SageMath grading function

```
#Dec 24, 2019. Nasser: Ported original Maple grading function by
#               Albert Rich to use with Sagemath. This is used to
#               grade Fricas, Giac and Maxima results.
#Dec 24, 2019. Nasser: Added 'exp_integral_e' and 'sng', 'sin_integral'
#               'arctan2','floor','abs','log_integral'
#June 4, 2022  Made default grade_annotation "none" instead of "" due
#               issue later when reading the file.
#July 14, 2022. Added ellipticF. This is until they fix sagemath, then remove it.

from sage.all import *
from sage.symbolic.operators import add_vararg, mul_vararg

debug=False;

def tree_size(expr):
    r"""
    Return the tree size of this expression.
    """
    #print("Enter tree_size, expr is ",expr)

    if expr not in SR:
        # deal with lists, tuples, vectors
        return 1 + sum(tree_size(a) for a in expr)
    expr = SR(expr)
    x, aa = expr.operator(), expr.operands()
    if x is None:
        return 1
    else:
        return 1 + sum(tree_size(a) for a in aa)

def is_sqrt(expr):
    if expr.operator() == operator.pow:   #isinstance(expr,Pow):
        if expr.operands()[1]==1/2: #expr.args[1] == Rational(1,2):
            if debug: print ("expr is sqrt")
            return True
        else:
            return False
    else:
        return False
```

```
def is_elementary_function(func):
    #debug=False
    m = func.name() in ['exp','log','ln',
            'sin','cos','tan','cot','sec','csc',
            'arcsin','arccos','arctan','arccot','arcsec','arccsc',
            'sinh','cosh','tanh','coth','sech','csch',
            'arcsinh','arccosh','arctanh','arccoth','arcsech','arccsch','sgn',
        'arctan2','floor','abs'
        ]
    if debug:
        if m:
            print ("func ", func , " is elementary_function")
        else:
            print ("func ", func , " is NOT elementary_function")


    return m

def is_special_function(func):
    #debug=False
    if debug:
        print ("type(func)=", type(func))

    m= func.name() in ['erf','erfc','erfi','fresnel_sin','fresnel_cos','Ei',
            'Ei','Li','Si','sin_integral','Ci','cos_integral','Shi','sinh_integral'
            'Chi','cosh_integral','gamma','log_gamma','psi,zeta',
            'polylog','lambert_w','elliptic_f','elliptic_e','ellipticF',
            'elliptic_pi','exp_integral_e','log_integral']

    if debug:
        print ("m=",m)
        if m:
            print ("func ", func ," is special_function")
        else:
            print ("func ", func ," is NOT special_function")


    return m


def is_hypergeometric_function(func):
    return func.name() in ['hypergeometric','hypergeometric_M','hypergeometric_U']

def is_appell_function(func):
    return func.name() in ['hypergeometric']    #[appellf1] can't find this in sagemath
```

```
def is_atom(expn):

    #debug=False
    if debug:
        print ("Enter is_atom, expn=",expn)

    if not hasattr(expn, 'parent'):
        return False


    #thanks to answer at https://ask.sagemath.org/question/49179/what-is-sagemath-equivalent-to-atomic-t
    try:
        if expn.parent() is SR:
            return expn.operator() is None
        if expn.parent() in (ZZ, QQ, AA, QQbar):
            return expn in expn.parent() # Should always return True
        if hasattr(expn.parent(),"base_ring") and hasattr(expn.parent(),"gens"):
            return expn in expn.parent().base_ring() or expn in expn.parent().gens()

        return False

    except AttributeError as error:
        print("Exception,AttributeError in  is_atom")
        print ("cought exception" , type(error).__name__ )
        return False


def expnType(expn):

    if debug:
        print (">>>>>Enter expnType, expn=", expn)
        print (">>>>>is_atom(expn)=", is_atom(expn))

    if is_atom(expn):
        return 1
    elif type(expn)==list:   #isinstance(expn,list):
        return max(map(expnType, expn))    #apply(max,map(ExpnType,expn))
    elif  is_sqrt(expn):
        if  type(expn.operands()[0])==Rational:  #type(isinstance(expn.args[0],Rational):
            return 1
        else:
            return max(2,expnType(expn.operands()[0]))   #max(2,expnType(expn.args[0]))
    elif expn.operator() == operator.pow:    #isinstance(expn,Pow)
        if type(expn.operands()[1])==Integer:   #isinstance(expn.args[1],Integer)
            return expnType(expn.operands()[0])    #expnType(expn.args[0])
        elif type(expn.operands()[1])==Rational:   #isinstance(expn.args[1],Rational)
            if type(expn.operands()[0])==Rational:  #isinstance(expn.args[0],Rational)
```

```
            return 1
         else:
            return max(2,expnType(expn.operands()[0]))  #max(2,expnType(expn.args[0]))
      else:
         return max(3,expnType(expn.operands()[0]),expnType(expn.operands()[1])) #max(3,expnType(expn.a
   elif expn.operator() == add_vararg or expn.operator() == mul_vararg: #isinstance(expn,Add) or isinsta
      m1 = expnType(expn.operands()[0]) #expnType(expn.args[0])
      m2 = expnType(expn.operands()[1:]) #expnType(list(expn.args[1:]))
      return max(m1,m2)  #max(ExpnType(op(1,expn)),max(ExpnType(rest(expn))))
   elif is_elementary_function(expn.operator()):  #is_elementary_function(expn.func)
      return max(3,expnType(expn.operands()[0]))
   elif is_special_function(expn.operator()): #is_special_function(expn.func)
      m1 = max(map(expnType, expn.operands()))      #max(map(expnType, list(expn.args)))
      return max(4,m1)   #max(4,m1)
   elif is_hypergeometric_function(expn.operator()): #is_hypergeometric_function(expn.func)
      m1 = max(map(expnType, expn.operands()))       #max(map(expnType, list(expn.args)))
      return max(5,m1)   #max(5,m1)
   elif is_appell_function(expn.operator()):
      m1 = max(map(expnType, expn.operands()))       #max(map(expnType, list(expn.args)))
      return max(6,m1)   #max(6,m1)
   elif str(expn).find("Integral") != -1: #this will never happen, since it
            #is checked before calling the grading function that is passed.
            #but kept it here.
      m1 = max(map(expnType, expn.operands()))       #max(map(expnType, list(expn.args)))
      return max(8,m1)   #max(5,apply(max,map(ExpnType,[op(expn)])))
   else:
      return 9

#main function
def grade_antiderivative(result,optimal):


   if debug:
      print ("Enter grade_antiderivative for sagemath")
      print("Enter grade_antiderivative, result=",result)
      print("Enter grade_antiderivative, optimal=",optimal)
      print("type(anti)=",type(result))
      print("type(optimal)=",type(optimal))

   leaf_count_result  = tree_size(result) #leaf_count(result)
   leaf_count_optimal = tree_size(optimal) #leaf_count(optimal)

   #if debug: print ("leaf_count_result=", leaf_count_result, "leaf_count_optimal=",leaf_count_optimal)


   expnType_result  = expnType(result)
   expnType_optimal = expnType(optimal)
```

```
if debug: print ("expnType_result=", expnType_result, "expnType_optimal=",expnType_optimal)

if expnType_result <= expnType_optimal:
    if result.has(I):
        if optimal.has(I): #both result and optimal complex
            if leaf_count_result <= 2*leaf_count_optimal:
                grade = "A"
                grade_annotation ="none"
            else:
                grade = "B"
                grade_annotation ="Both result and optimal contain complex but leaf count of result is larger th
        else: #result contains complex but optimal is not
            grade = "C"
            grade_annotation ="Result contains complex when optimal does not."
    else: # result do not contain complex, this assumes optimal do not as well
        if leaf_count_result <= 2*leaf_count_optimal:
            grade = "A"
            grade_annotation ="none"
        else:
            grade = "B"
            grade_annotation ="Leaf count of result is larger than twice the leaf count of optimal. "+str(leaf_c
else:
    grade = "C"
    grade_annotation ="Result contains higher order function than in optimal. Order "+str(expnType_resul


print("Before returning. grade=",grade, " grade_annotation=",grade_annotation)

return grade, grade_annotation
```