

advection PDE in 1D using Ada

Nasser M. Abbasi

Summer 2010

Compiled on September 26, 2025 at 4:17am

1 introduction

To help learn a little more about Ada, I wrote ¹ the following small program which solves the advection PDE $u_t + au_x = 0$ with initial condition $u(x, 0) = \sin(2\pi x)$ and with periodic boundary conditions $u(0, t) = u(1, t)$.

To plot the solution, the output of the program is directed to a text file, then the the text file was loaded into Matlab to ran the simulation.

The program contains 3 files

1. main.adb the main line.
2. lax_wendroff_pkg.ads the package interface.
3. lax_wendroff_pkg.adb the package body.

These are the steps to compile the program

```
$ gnatmake -gnat05 -gnatwa main.adb
gcc -c -gnat05 -gnatwa lax_wendroff_pkg.adb
gnatbind -x main.ali
gnatlink main.ali
```

To run the program do `./main > result.txt`

Matlab was used to plot the solution as follows

```
A=load('result.txt','-ascii');
B=reshape(A,100,100);

for i=1:size(B,2)
    plot(B(:,i))
    ylim([-1 1]);
    drawnow();

    pause(0.01);
end
```

¹thanks goes to Randy Brukardt, Pascal Obry, Shark8 and others for giving helpful advice on Ada coding at comp.lang.ada

2 Ada source code listing

2.1 main.adb

```
-- compile with gnatmake -gnataw main.adb
-- Main driver to solve  $u_t + \text{speed} * u_x = 0$ 
-- the advection PDE with periodic boundary
-- conditions and with initial conditions  $\sin(2\pi x)$ 
--
-- Using Ada 2005 OO features
-- For illustration and learning only.
--
-- By Nasser M. Abbasi
-- Match 26, 2011

with Ada.Text_IO; use Ada.Text_IO;
with Lax_Wendroff_pkg; Use Lax_Wendroff_pkg;
with Ada.Numerics; use Ada.Numerics;
with Ada.Numerics.Generic_Elementary_Functions;

procedure main is
package Math is new Ada.Numerics.Generic_Elementary_Functions(Float);
use Math;

h      : constant float := 0.01; -- grid space
courant : constant float := 0.8; -- CFL condition
speed   : constant float := 1.0; -- flow speed
k       : constant float := courant*h/speed; -- time step size
L       : constant float := 1.0; -- domain length
N       : constant positive := natural(L/h)+1; --number of grid points

-- generate initial condition
function initialize(h: float; N: natural) return solution_t is
  data : solution_t(1..N) := (others=>0.0);
  x: float:=0.0;
  begin
    for i in data'range LOOP
      data(i) := sin(2.0*pi*x);
      x := x + h;
    end LOOP;
    RETURN(data);
  end;

-- create the Lax-Wendroff object
o      : lax_Wendroff_t := make(speed=>speed , h=>h,k =>k, u0 =>initialize(h,N));

-- to print the final solution
procedure print_solution(o: lax_wendroff_t) is
  u: constant solution_t := o.get_solution;
  begin
    FOR i in u'range LOOP
      put_line(float'image(u(i)));
    END LOOP;
  end;

begin
```

```

        -- run for 100 steps for now
        FOR i in 1..100 LOOP
            o.step;
            print_solution(o);
        END LOOP;

end main;

```

2.2 lax wendroff pkg.ads

```

-- Package spec for Lax Wendroff scheme to solve 1-D
-- Advection PDE in Ada 2005
-- by Nasser M. Abbasi
--

with Ada.Numerics.Generic_Real_Arrays;
package Lax_Wendroff_pkg is
    type Lax_Wendroff_t (<>) is tagged private;
    type solution_t is array (Natural range <>) of Float;

    -- primitive operations, constructor
    function make(speed,h,k : Float; u0 : solution_t) return Lax_Wendroff_t;
    procedure step (o : in out Lax_Wendroff_t);
    function get_solution (o : Lax_Wendroff_t) return solution_t;

private
    package My_Vectors is new Ada.Numerics.Generic_Real_Arrays (Float);
    use My_Vectors;
    subtype buffer_t is Real_Vector;

    type Lax_Wendroff_t (N : Positive) is tagged record
        speed      : Float;           -- speed of flow
        h          : Float;           -- space grid size
        k          : Float;           -- delt
        u          : buffer_t (-1 .. N); -- solution
        step_number : Natural;
    end record;

end Lax_Wendroff_pkg;

```

2.3 lax wendroff pkg.add

```

-- Package body for Lax Wendroff scheme to solve 1-D
-- Advection PDE in Ada 2005
-- by Nasser M. Abbasi
--

package body Lax_Wendroff_pkg is

    -- constructor
    function make
        (speed : Float;
         h     : Float;
         k     : Float;
         u0    : solution_t)
        return Lax_Wendroff_t
    is
        o : Lax_Wendroff_t (u0'Length);
    begin

```

```

    o.speed          := speed;
    o.h              := h;
    o.k              := k;
    o.step_number    := 0;
    o.u (0 .. u0'Length - 1) := buffer_t (u0);
    o.u (o.u'First)   := 0.0;
    o.u (o.u'Last)    := 0.0;

    return (o);
end make;

-- make solution step
procedure step (o : in out Lax_Wendroff_t) is
    package my_vectors is new Ada.Numerics.Generic_Real_Arrays (Float);
    use my_vectors;
    u : buffer_t renames o.u;
    a : constant Float := o.speed * o.k / o.h;
    subtype r is Natural range o.u'First + 1 .. o.u'Last - 1;
    subtype r_plus_1 is Natural range r'First + 1 .. r'Last + 1;
    subtype r_minus_1 is Integer range r'First - 1 .. r'Last - 1;

begin
    o.step_number := o.step_number + 1;

    u (r) := u (r) -
        (a / 2.0) * (u (r_plus_1) - u (r_minus_1)) +
        (a ** 2) / 2.0 *
        (u (r_minus_1) - 2.0 * u (r) + u (r_plus_1));

    -- adjust due to periodic boundary conditions
    u (u'First) := u (u'Last - 2);
    u (u'Last)  := u (u'First + 2);

end step;

-- get the current solution
function get_solution (o : Lax_Wendroff_t) return solution_t is
begin
    return solution_t (o.u (o.u'First + 1 .. o.u'Last - 1));
end get_solution;

end Lax_Wendroff_pkg;

```