```
> restart;
  with(PDEtools):
  with(ArrayTools):
  with(plots):
```

FTCS stencil for the diffusion equation

The purpose of this worksheet is to develop a working code to numerically solve the diffusion equation. We will emply a particular finite difference stencil for the PDE that involves apprixmating the time derivatives in a one-sided manner baised toward the future (forward time or FT) and a centered approximation for the spatial derivatives (center space or CS). We'll make use of the centered_stencil and one_sided procedures developed in stencils_higher_derivatives.mws:

```
> centered_stencil := proc(r,N,{direction := spatial})
      local n, stencil, vars, beta_sol:
      n := floor(N/2):
      if (direction = spatial) then:
         stencil := D[2$r](u)(t,x) - add(beta[i]*u(t,x+i*h),i=-n..
  n);
         vars := [u(t,x),seq(D[2$i](u)(t,x),i=1..N-1)];
      else:
         stencil := D[1$r](u)(t,x) - add(beta[i]*u(t+i*h,x),i=-n..
  n);
         vars := [u(t,x),seq(D[1$i](u)(t,x),i=1..N-1)];
      fi:
      beta_sol := solve([coeffs(collect(convert(series(stencil,h,
  N),polynom),vars,'distributed'),vars)]):
      stencil := subs(beta_sol,stencil);
      if (direction = spatial) then:
         convert(stencil = convert(series(stencil,h,N+2),polynom),
  diff);
      else:
         subs(h=s,convert(stencil = convert(series(stencil,h,N+2),
  polynom),diff));
      fi:
  end proc:

  onesided_stencil := proc(r,N,{direction := spatial})
      local stencil, vars, beta_sol:
      if (direction = spatial) then:
         stencil := D[2$r](u)(t,x) - add(beta[i]*u(t,x+i*h),i=0..
  N-1);
         vars := [u(t,x),seq(D[2$i](u)(t,x),i=1..N-1)];
      else:
         stencil := D[1$r](u)(t,x) - add(beta[i]*u(t+i*h,x),i=0..
  N-1);
         vars := [u(t,x),seq(D[1$i](u)(t,x),i=1..N-1)];
      fi:
      beta_sol := solve([coeffs(collect(convert(series(stencil,h,
  N),polynom),vars,'distributed'),vars)]):
      stencil := subs(beta_sol,stencil);
      if (direction = spatial) then:
         convert(stencil = convert(series(`leadterm`(stencil),h,
  N+1),polynom),diff);
      else:
         subs(h=s,convert(stencil = convert(series(`leadterm`
  (stencil),h,N+1),polynom),diff));
      fi:
  end proc:
```

Here is the differential equation we want to solve:

```
> eq0 := diff(u(t,x),t) - d*diff(u(t,x),x,x);
```

$$eq0 := \frac{\partial}{\partial t} u(t,x) - d \left(\frac{\partial^2}{\partial x^2} u(t,x) \right) \quad \textbf{(1)}$$

We will employ a one-sided and centered stencil for the temporal and spatial derivatives, respectively. These are given by:

```
> eq1 := onesided_stencil(1,2,direction=temporal);
  eq2 := centered_stencil(2,3,direction=spatial);
```

$$eq1 := \frac{\partial}{\partial t} u(t,x) + \frac{u(t,x)}{s} - \frac{u(t+s,x)}{s} = -\frac{1}{2} \left(\frac{\partial^2}{\partial t^2} u(t,x) \right) s \quad \textbf{(2)}$$

$$eq2 := \frac{\partial^2}{\partial x^2} u(t,x) - \frac{u(t,x-h)}{h^2} + \frac{2\, u(t,x)}{h^2} - \frac{u(t,x+h)}{h^2} = -\frac{1}{12} \left(\frac{\partial^4}{\partial x^4} u(t,x) \right) h^2$$

We discard the error terms on the right and re-arrange both equations to solve for the derivatives:

```
> eq3 := isolate(lhs(eq1),diff(u(t,x),t));
  eq4 := isolate(lhs(eq2),diff(u(t,x),x,x));
```

$$eq3 := \frac{\partial}{\partial t} u(t,x) = -\frac{u(t,x)}{s} + \frac{u(t+s,x)}{s} \quad \textbf{(3)}$$

$$eq4 := \frac{\partial^2}{\partial x^2} u(t,x) = \frac{u(t,x-h)}{h^2} - \frac{2\, u(t,x)}{h^2} + \frac{u(t,x+h)}{h^2}$$

We substitute these stencils into the PDE an isolate u(t+s,x):

```
> eq5 := subs(eq3,eq4,eq0);
  eq6 := expand(isolate(eq5,u(t+s,x)));
```

$$eq5 := -\frac{u(t,x)}{s} + \frac{u(t+s,x)}{s} - d \left(\frac{u(t,x-h)}{h^2} - \frac{2\, u(t,x)}{h^2} + \frac{u(t,x+h)}{h^2} \right) \quad \textbf{(4)}$$

$$eq6 := u(t+s,x) = u(t,x) + \frac{s\, d\, u(t,x-h)}{h^2} - \frac{2\, s\, d\, u(t,x)}{h^2} + \frac{s\, d\, u(t,x+h)}{h^2}$$

eq6 is our stencil to solve the problem. It gives the value of u in the future (time t + s) in terms of the value of u in the past (time t). Given knowledge of u at time t, the future values of u are given *explicitly* by eq6, hence we call this an explicit scheme. The error in the stencil can be obtained by expanding the LHS minus RHS of eq6 in a double Taylor series in s and h:

```
> eq7 := Error = expand(series(series((lhs-rhs)(eq6),s),h));
```

$$eq7 := Error = \mathrm{D}_{1}(u)(t,x)\, s - s\, d\, \mathrm{D}_{2,2}(u)(t,x) + \frac{1}{24} \mathrm{D}_{1,1,1,1}(u)(t,x)\, s^4 + \frac{1}{120} \mathrm{D}_{1,1,1,1,1}(u)(t,x)\, s^5 + \frac{1}{2} \mathrm{D}_{1,1}(u)(t,x)\, s^2 + \frac{1}{6} \mathrm{D}_{1,1,1}(u)(t,x)\, s^3 + \mathrm{O}(s^6) - \frac{1}{12} d\, \mathrm{D}_{2,2,2,2}(u)(t,x)\, s\, h^2 + \mathrm{O}(h^4) \quad \textbf{(5)}$$

Of course, this sould be simplifed by using the original PDE:

```
> eq8 := convert(convert(dsubs(isolate(eq0,diff(u(t,x),t)),convert
  (eq7,diff)),polynom),polynom);
```

$$eq8 := Error = \frac{1}{24} d^4 \left(\frac{\partial^8}{\partial x^8} u(t,x) \right) s^4 + \frac{1}{120} d^5 \left(\frac{\partial^{10}}{\partial x^{10}} u(t,x) \right) s^5 + \frac{1}{2} d^2 \left(\frac{\partial^4}{\partial x^4} u(t, \quad \textbf{(6)}$$

$$x)\Big) s^2 + \frac{1}{6} d^3 \left(\frac{\partial^6}{\partial x^6} u(t,x)\right) s^3 - \frac{1}{12} d \left(\frac{\partial^4}{\partial x^4} u(t,x)\right) s\, h^2$$

It is not *a priori* obvious which is the dominant error term without knowing anything about the relative sizes of s and h, so we'll leave this alone for now. We now convert eq6 into a procedure that forms the basis of our numerical scheme.

```
> eq9 := subs(u(t,x)=u_middle,u(t,x-h)=u_left,u(t,x+h)=u_right,u(t+
  s,x)=u_future,eq6);
  evolve := unapply(rhs(eq9),u_left,u_middle,u_right,h,s,d);
```

$$eq9 := u\_future = u\_middle + \frac{s\, d\, u\_left}{h^2} - \frac{2\, s\, d\, u\_middle}{h^2} + \frac{s\, d\, u\_right}{h^2} \tag{7}$$

$$evolve := (u\_left, u\_middle, u\_right, h, s, d) \rightarrow u\_middle + \frac{s\, d\, u\_left}{h^2} - \frac{2\, s\, d\, u\_middle}{h^2} + \frac{s\, d\, u\_right}{h^2}$$

The mapping evolve takes information about u on a given time slice and gives the value at a future time slice in terms of h, s, and d. Before we write our own procedure exploiting this stencil, let's use pdsolve/numeric to see what the solution should look like. Now, we need to specifiy boundard and initial conditions to get a numerical solution. Since the PDE has only one time derivative, we only need to specify the initial profile of u. For boundary conditons, we fix u to be zero at x = -X/2 and x = +X/2. We also choose a time step size s, and select the spatial stepsize to be proportional to sqrt(s). Why? This will be clarified in a later worksheet on the von Neumann stabiliy analysis. We now use pdsolve/numeric to generate a movie:

```
> T := 1;
  X := 2;
  d := 1;
  s := 0.01;
  h := sqrt(s*d*2);
  f := x-> exp(-(5*x/X)^8):
  IBC := [u(0,x)=f(x),u(t,-X/2)=0,u(t,+X/2)=0]:
  pds := pdsolve(eq0,IBC,numeric,spacestep=h,timestep=s):
  pds:-animate(t=0..T,axes=boxed,frames=40);
```

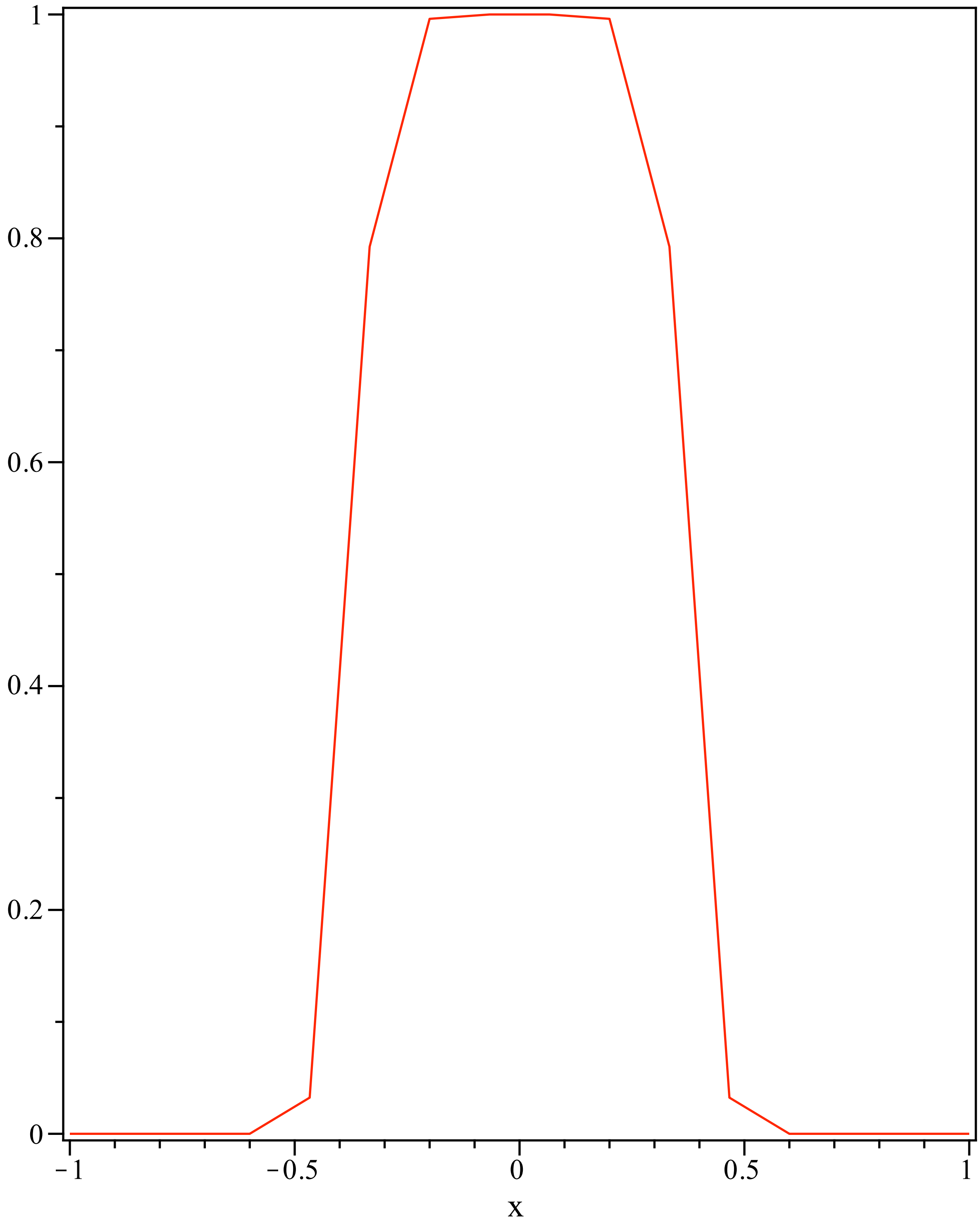
$$T := 1$$

$$X := 2$$

$$d := 1$$

$$s := 0.01$$

$$h := 0.1414213562$$



Here is our procedure to generate a similar movie using the FTCS stencil. There are comments interspered through the code:

```
> FTCS := proc(_X,_T,_s,_h,_d,_f) local N, M, x, XX, u_past,
  u_future, p, i, j:

        # We first determine the number of time and space steps

        N := round(_T/_s):
        M := round(_X/_h):
```

```
        # We define a vector containing the x-coordinates of the
spatial lattice

        x := j -> -_X/2 + (j-1)/(M-1)*_X;
        XX := Vector(1..M,[seq(x(j),j=1..M)],datatype=float):

        # The scheme will be based on two vectors u_past and
u_future
        # u_past corresponds to the field values on a given time
step
        # u_future corresponds to the field values at the next
time step
        # To begin, we fix u_past from the initial data

        u_past := Vector(1..M,[seq(_f(XX[j]),j=1..M)],datatype=
float):

        # Our goal will be a movie whose frames are plots of u at
each time step
        # Here is the first frame

        p[1] := plot(convert(Matrix([XX,u_past]),listlist),axes=
boxed);

        # Now we start the main calculation loop
        # i will run over time steps while j runs over spacesteps

        for i from 2 to N do:

            # We initialize the u_future vector and fix its
values at either end based on the BOUNDARY CONDITIONS

            u_future := Vector(1..M,datatype=float):
            u_future[1] := 0:
            u_future[M] := 0:

            # The rest of the values of u_future are obtained by
using the evolve procedure
            # Notice the arguments of evolve are the values of
u_past

            for j from 2 to M-1 do:
                 u_future[j] := evolve(u_past[j-1],u_past[j],
u_past[j+1],_h,_s,_d):
            od:

            # Now, we get ready for the next time step by making
u_future into the new u_past

            Copy(u_future,u_past):

            # Finally, another frame of our movie is obtained
from plotting the values of the new u_past

            p[i] := plot(convert(Matrix([XX,u_past]),listlist),
axes=boxed);

        od:

        # After the loop is over, we have N plots p[i] that are
the pictures of u at each time slice
        # The display command assembles these into a movie, which
```

```
is the output of the procedure

      display(convert(p,list),insequence=true):

end proc:
```

Here is the output of our procedure for the same parameters as pdsolve/numeric above. We get pretty much the same answer, but our procedure takes longer since we have made no effort to obtain an efficient code. Also, the frame rate is different.

```
> FTCS(X,T,s,h,d,f);
```