```
> restart;
  with(PDEtools):
  with(LinearAlgebra):
  with(plots):
  interface(rtablesize=20):
```

Finite difference solution of Laplace's equation

The purpose of the worksheet is to solve Laplace's equation using finite differencing. Laplace's equation is:

```
> laplace := diff(u(x,y),x,x) + diff(u(x,y),y,y) = 0;
```

$$laplace := \frac{\partial^2}{\partial x^2} u(x,y) + \frac{\partial^2}{\partial y^2} u(x,y) = 0 \qquad (1)$$

We will solve this on the unit square [0,1] x [0,1] assuming boundary conditions of the form

```
> BCs := [u(x,0)=f_bot(x),u(x,1)=f_top(x),u(0,y)=f_left(y),u(1,y)=
  f_right(y)];
```

$$BCs := [u(x,0) = f\_bot(x), u(x,1) = f\_top(x), u(0,y) = f\_left(y), u(1,y) = f\_right(y)] \qquad (2)$$

Here, f_bot, f_top, etc are given functions. We will only specialize to the unit square at the very end, up to that point we consider a general rectangular region. We will be using the fivepoint stencil of this equation. This obtained with the following code:

```
> centered_stencil := proc(r,N,{direction := x})
      local n, stencil, vars, beta_sol:
      n := floor(N/2):
      if (direction = y) then:
         stencil := D[2$r](u)(x,y) - add(beta[i]*u(x,y+i*h),i=-n..
  n);
         vars := [u(x,y),seq(D[2$i](u)(x,y),i=1..N-1)];
      else:
         stencil := D[1$r](u)(x,y) - add(beta[i]*u(x+i*h,y),i=-n..
  n);
         vars := [u(x,y),seq(D[1$i](u)(x,y),i=1..N-1)];
      fi:
      beta_sol := solve([coeffs(collect(convert(series(stencil,h,
  N),polynom),vars,'distributed'),vars)]):
      stencil := subs(beta_sol,stencil);
      convert(stencil = convert(series(stencil,h,N+2),polynom),
  diff);
  end proc:
  x_stencil := isolate(lhs(centered_stencil(2,3,direction=x)),diff
  (u(x,y),x,x));
  y_stencil := isolate(lhs(centered_stencil(2,3,direction=y)),diff
  (u(x,y),y,y));
  laplace_stencil := subs(x_stencil,y_stencil,laplace);
```

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

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

$$laplace\_stencil := \frac{u(x-h,y)}{h^2} - \frac{4\,u(x,y)}{h^2} + \frac{u(x+h,y)}{h^2} + \frac{u(x,y-h)}{h^2} + \frac{u(x,y+h)}{h^2} = 0 \qquad (3)$$

Interesting, all the h^2 terms in the denominator can be cancelled:

```
> laplace_stencil := expand(laplace_stencil*h^2);
```

$$laplace\_stencil := u(x-h,y) - 4\,u(x,y) + u(x+h,y) + u(x,y-h) + u(x,y+h) = 0 \quad (4)$$

Re-labelling the u's as follows:

```
> Subs := [seq(seq(u(x+jj*h,y+ii*h)=u[i+ii,j+jj],ii=-1..1),jj=-1.
  .1)];
```

$$Subs := \left[u(x-h,y-h) = u_{i-1,j-1}, u(x-h,y) = u_{i,j-1}, u(x-h,y+h) = u_{i+1,j-1},\right. \quad (5)$$

$$u(x,y-h) = u_{i-1,j}, u(x,y) = u_{i,j}, u(x,y+h) = u_{i+1,j}, u(x+h,y-h)$$

$$\left. = u_{i-1,j+1}, u(x+h,y) = u_{i,j+1}, u(x+h,y+h) = u_{i+1,j+1}\right]$$

Our convention is that i increases with y and j increases with x. Putting these into our stencil we get

```
> laplace_stencil := subs(Subs,laplace_stencil);
```

$$laplace\_stencil := u_{i,j-1} - 4\,u_{i,j} + u_{i,j+1} + u_{i-1,j} + u_{i+1,j} = 0 \quad (6)$$

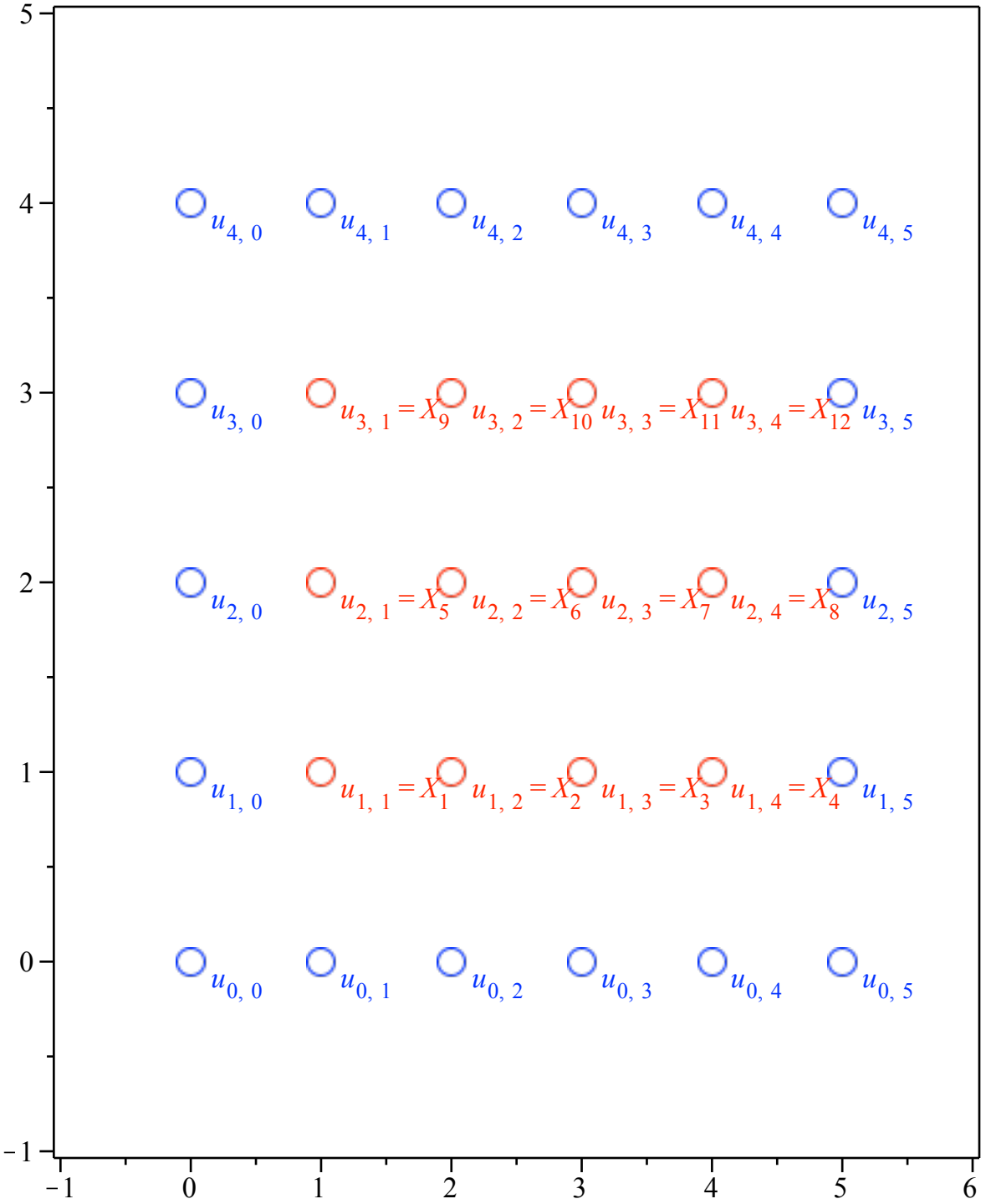
This says that the middle point is the average of the four outer points

```
> isolate(laplace_stencil,u[i,j]);
```

$$u_{i,j} = \frac{1}{4}\,u_{i,j-1} + \frac{1}{4}\,u_{i,j+1} + \frac{1}{4}\,u_{i-1,j} + \frac{1}{4}\,u_{i+1,j} \quad (7)$$

Let's consider the situation were we have M points in the x direction times N points in the y direction where we want to know the field values (this is more general than the problem of solving for u on the unit square). We assume boundary conditions are given such that u is known on all the points immediately exterior to this region. Here is a plot for a specific choice of M and N:

```
> M := 4:
  N := 3:
  p1 := plot([seq(seq([j,i],j=1..M),i=1..N)],view=[-1..M+2,-1..
  N+2],style=point,symbolsize=20,symbol=circle,axes=boxed):
  p2 := textplot([seq(seq([j+0.1,i,u[i,j]=X[(i-1)*M+j]],j=1..M),i=
  1..N)],align=[right,below],color=red):
  p3 := plot([seq([0,i],i=1..N),seq([M+1,i],i=1..N),seq([i,0],i=0..
  M+1),seq([i,N+1],i=0..M+1)],view=[-1..M+1,-1..N+1],style=point,
  symbolsize=20,symbol=circle,axes=boxed,color=blue):
  p4 := textplot([seq([0.1,i,u[i,0]],i=1..N),seq([M+1+0.1,i,u[i,
  M+1]],i=1..N),seq([i+0.1,0,u[0,i]],i=0..M+1),seq([i+0.1,N+1,u
  [N+1,i]],i=0..M+1)],align=[right,below],color=blue):
  display([p1,p2,p3,p4]);
```



We don't know u at each of the interior (red) points, but we do know u at each of the exterior (blue) points. laplace_stencil can be evaluated at each of the red points to give N*M linear equations for the N*M unknown u[i,j]'s. If we arrange he unknown u[i,j]'s into a single vector X as indicated in the plot, this is a matrix equation A.X = B. The following code generates A and B for the choices of N and M above:

```
> eq := 'eq':
  COUNT := 0:
  for i from 1 to N do:
    for j from 1 to M do:
       COUNT := COUNT + 1:
```

```
      eq[COUNT] := laplace_stencil;
      print(eq[COUNT]);
  od;
od;
vars := [seq(seq(u[i,j],j=1..M),i=1..N)];
A,B := GenerateMatrix(convert(eq,list),vars);
```

$$u_{1,0} - 4\,u_{1,1} + u_{1,2} + u_{0,1} + u_{2,1} = 0$$

$$u_{1,1} - 4\,u_{1,2} + u_{1,3} + u_{0,2} + u_{2,2} = 0$$

$$u_{1,2} - 4\,u_{1,3} + u_{1,4} + u_{0,3} + u_{2,3} = 0$$

$$u_{1,3} - 4\,u_{1,4} + u_{1,5} + u_{0,4} + u_{2,4} = 0$$

$$u_{2,0} - 4\,u_{2,1} + u_{2,2} + u_{1,1} + u_{3,1} = 0$$

$$u_{2,1} - 4\,u_{2,2} + u_{2,3} + u_{1,2} + u_{3,2} = 0$$

$$u_{2,2} - 4\,u_{2,3} + u_{2,4} + u_{1,3} + u_{3,3} = 0$$

$$u_{2,3} - 4\,u_{2,4} + u_{2,5} + u_{1,4} + u_{3,4} = 0$$

$$u_{3,0} - 4\,u_{3,1} + u_{3,2} + u_{2,1} + u_{4,1} = 0$$

$$u_{3,1} - 4\,u_{3,2} + u_{3,3} + u_{2,2} + u_{4,2} = 0$$

$$u_{3,2} - 4\,u_{3,3} + u_{3,4} + u_{2,3} + u_{4,3} = 0$$

$$u_{3,3} - 4\,u_{3,4} + u_{3,5} + u_{2,4} + u_{4,4} = 0$$

$$vars := \left[u_{1,1}, u_{1,2}, u_{1,3}, u_{1,4}, u_{2,1}, u_{2,2}, u_{2,3}, u_{2,4}, u_{3,1}, u_{3,2}, u_{3,3}, u_{3,4}\right]$$

$$A, B := \begin{bmatrix} -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 \end{bmatrix}, \begin{bmatrix} -u_{1,0} - u_{0,1} \\ -u_{0,2} \\ -u_{0,3} \\ -u_{1,5} - u_{0,4} \\ -u_{2,0} \\ 0 \\ 0 \\ -u_{2,5} \\ -u_{3,0} - u_{4,1} \\ -u_{4,2} \\ -u_{4,3} \\ -u_{4,4} - u_{3,5} \end{bmatrix} \tag{8}$$

The matrix A is composed of a block diagonal matrix plus a banded matrix. The vector B is made up of the boundary data as represented by the blue nodes in the above plot. After staring at this for a while (and playing around with the values of M and N), you'll see how to construct A for and N and M and how to construct B knowing the boundary data. These procedures do just that:

```
> AA := proc(N,M)
      local COUNT, C, E, F, G:
      COUNT := N*M:
      C := BandMatrix([1,-4,1],1,M):
      E := DiagonalMatrix([C$N]):
      F := BandMatrix([1,0$(2*M-1),1],M,COUNT):
      G := E + F:
  end proc:

  BB := proc(bottom,top,left,right)
       local N, M, q, i:
       N := Dimension(left):
       M := Dimension(bottom):
       q[1] := Vector([-bottom[1]-left[1],seq(-bottom[j],j=2..M-1),
  -bottom[M]-right[1]]);
       for i from 2 to N-1 do:
           q[i] := Vector([-left[i],0$(M-2),-right[i]]):
       od:
       q[N] := Vector([-top[1]-left[N],seq(-top[j],j=2..M-1),-top
  [M]-right[N]]);
       Vector(convert(q,list));
  end proc:
```

Notice the arguments of the BB procedure, it takes the values of the field at the bottom, top, left and right of the domain arranged into vectors. Notice that the corner nodes (u[0,0], etc) don't have to be included because the fivepoint stencil will never rference them. Hence, bottom is an M-dimensional vector, left is an N-dimensional vector, etc. We can test our procedures by generating dummy data:

```
> Bottom := Vector([seq(u[0,j],j=1..M)]):
  Top := Vector([seq(u[N+1,j],j=1..M)]):
  Left := Vector([seq(u[i,0],i=1..N)]):
  Right := Vector([seq(u[i,M+1],i=1..N)]):
  Bottom, Top, Left, Right;
```

$$\begin{bmatrix} u_{0,1} \\ u_{0,2} \\ u_{0,3} \\ u_{0,4} \end{bmatrix}, \begin{bmatrix} u_{4,1} \\ u_{4,2} \\ u_{4,3} \\ u_{4,4} \end{bmatrix}, \begin{bmatrix} u_{1,0} \\ u_{2,0} \\ u_{3,0} \end{bmatrix}, \begin{bmatrix} u_{1,5} \\ u_{2,5} \\ u_{3,5} \end{bmatrix} \tag{9}$$

Then, subtracting the A and B determined above from the direct examination of the linear system from the procedure output generates zero, as required.

```
> AA(N,M)-A,BB(Bottom,Top,Left,Right)-B;
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \textbf{(10)}$$

This procedure solve Laplace's equation on the unit square subject to boundary conditions given by f_bot, f_top, etc. We use LinearSolve with the SparseIterative option to solve the matrix equation. The output is a 3D plot:

```
> LaplaceSolve := proc(N,f_bot,f_top,f_left,f_right)
      local h, Bottom, Top, Left, Right, sol, i, Data, sys;
      h := evalf(1/(N+1)); # fix stepsize
      Bottom := Vector([seq(f_bot(i*h),i=1..N)]):  # initialize
  boundary data
      Top := Vector([seq(f_top(i*h),i=1..N)]):     # initialize
  boundary data
      Left := Vector([seq(f_left(i*h),i=1..N)]):   # initialize
  boundary data
      Right := Vector([seq(f_right(i*h),i=1..N)]): # initialize
  boundary data
      sys := AA(N,N),BB(Bottom,Top,Left,Right);    # define linear
  system to solve
      sol := LinearSolve(sys,method=SparseIterative): # solve the
  system using LinearSolve
      Data[0] := [seq([j*h,0,f_bot(j*h)],j=0..N+1)]: # following
  code sets data for surfplot
      for i from 1 to N do:
          Data[i] := [[0,i*h,Left[i]],seq([j*h,i*h,sol[N*(i-1)+j]
  ],j=1..N),[1,i*h,Right[i]]]:
      od:
      Data[N+1] := [seq([j*h,1,f_top(j*h)],j=0..N+1)]:
      Data := convert(Data,list):
      surfdata(Data,axes=boxed,labels=[`x`,`y`,`u(x,y)`],shading=
  zhue,style=patchcontour); # surfplot generates the output
  end proc:
```

Here is an example problem:

```
> bottom := x -> x*(1-x);
  top := x -> 0;
  left := x -> 0.;
  right := x -> 0.;
  solution := LaplaceSolve(15,bottom,top,left,right):
```

$$bottom := x \rightarrow x\,(1 - x)$$

$$top := x \to 0$$

$$left := x \to 0.$$

$$right := x \to 0. \qquad \textbf{(11)}$$

Here is the output:

```
> solution;
```