```
> restart;
  with(LinearAlgebra):
  with(plots):
```

Matrix methods for solving a linear ODE boundary value problem

The purpose of this worksheet is to illustrate the use of matrix methods to solve a boundary value problem for a second order linear ODE. The ODE and BCs we will work with are:

```
> ODE := x^2*diff(u(x),x,x)+x*diff(u(x),x)+(x^2-1)*u(x);
  BCs := u(1)=0,u(10)=1;
```

$$ODE := x^2 \left(\frac{d^2}{dx^2} u(x) \right) + x \left(\frac{d}{dx} u(x) \right) + (x^2 - 1) u(x) \quad (1)$$

$$BCs := u(1) = 0, u(10) = 1$$

This BVP has an analytic solution in terms of Bessel functions:

```
> analytic_sol := dsolve([ODE,u(1)=0,u(10)=1]);
```

$$analytic\_sol := u(x) = \frac{\mathrm{BesselY}(1, 1)\, \mathrm{BesselJ}(1, x)}{\mathrm{BesselY}(1, 1)\, \mathrm{BesselJ}(1, 10) - \mathrm{BesselY}(1, 10)\, \mathrm{BesselJ}(1, 1)} - \frac{\mathrm{BesselJ}(1, 1)\, \mathrm{BesselY}(1, x)}{\mathrm{BesselY}(1, 1)\, \mathrm{BesselJ}(1, 10) - \mathrm{BesselY}(1, 10)\, \mathrm{BesselJ}(1, 1)} \quad (2)$$

We will attempt to reproduce this analytical solution using matrix methods. The first step is to discretize the ODE using finite difference approximations for the derivatives. The procedure centered_stencil generates an N point centered_stencil for the rth derivative of u:

```
> centered_stencil := proc(r,N)
      local n, stencil, vars, beta_sol:
      n := floor(N/2):
         stencil := D[1$r](u)(x) - add(beta[i]*u(x+i*h),i=-n..n);
         vars := [u(x),seq(D[1$i](u)(x),i=1..N-1)];
      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:
```

We use the procedure to create stencils for the first and second derivatives of u:

```
> substencil_1 := isolate(lhs(centered_stencil(1,3)),diff(u(x),x));
  substencil_2 := isolate(lhs(centered_stencil(2,3)),diff(u(x),x));
```

$$substencil\_1 := \frac{d}{dx} u(x) = -\frac{1}{2} \frac{u(x-h)}{h} + \frac{1}{2} \frac{u(x+h)}{h} \quad (3)$$

$$substencil\_2 := \frac{d^2}{dx^2} u(x) = \frac{u(x-h)}{h^2} - \frac{2\, u(x)}{h^2} + \frac{u(x+h)}{h^2}$$

Subbing these sub-stencils into the ODE gives

```
> stencil :=subs(substencil_2,substencil_1,ODE);
```

$$stencil := x^2 \left(\frac{u(x-h)}{h^2} - \frac{2\, u(x)}{h^2} + \frac{u(x+h)}{h^2} \right) + x \left(-\frac{1}{2} \frac{u(x-h)}{h} + \frac{1}{2} \frac{u(x+h)}{h} \right) \quad (4)$$

$$+ \left(x^2 - 1\right) u(x)$$

We re-label the various quantities in the above as follows:

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

$$Subs := \left[u(x-h) = u_{j-1}, u(x) = u_j, u(x+h) = u_{j+1}, x = x_j\right] \quad \textbf{(5)}$$

$$stencil := x_j^2 \left(\frac{u_{j-1}}{h^2} - \frac{2\,u_j}{h^2} + \frac{u_{j+1}}{h^2}\right) + x_j \left(-\frac{1}{2}\,\frac{u_{j-1}}{h} + \frac{1}{2}\,\frac{u_{j+1}}{h}\right) + \left(x_j^2 - 1\right) u_j$$

It will be convenient to convert stencil into a procedure using the unapply command:

```
> Stencil := unapply(stencil,x[j],u[j-1],u[j],u[j+1],h);
```

$$Stencil := (y1, y2, y3, y4, h) \rightarrow y1^2 \left(\frac{y2}{h^2} - \frac{2\,y3}{h^2} + \frac{y4}{h^2}\right) + y1 \left(-\frac{1}{2}\,\frac{y2}{h} + \frac{1}{2}\,\frac{y4}{h}\right) + \left(y1^2 - 1\right) y3 \quad \textbf{(6)}$$

Now, if the x-lattice contains M+2 points with x[0] = 1 and x[M+1] = 10, then stencil must hold for all j = 1 to M. This gives an M-dimensional linear system for the u[j] to solve once the boundary conditions u[0] = 0 and u[M+1] = 1 are imposed. Here is an example of what the system looks like for a small value of M:

```
> u := 'u':
  M := 5:
  u[0] := 0:
  u[M+1] := 1:
  for i from 1 to M do:
     eq[i] := Stencil(x[i],u[i-1],u[i],u[i+1],h)
  od;
```

$$eq_1 := x_1^2 \left(-\frac{2\,u_1}{h^2} + \frac{u_2}{h^2}\right) + \frac{1}{2}\,\frac{x_1\,u_2}{h} + \left(x_1^2 - 1\right) u_1 \quad \textbf{(7)}$$

$$eq_2 := x_2^2 \left(\frac{u_1}{h^2} - \frac{2\,u_2}{h^2} + \frac{u_3}{h^2}\right) + x_2 \left(-\frac{1}{2}\,\frac{u_1}{h} + \frac{1}{2}\,\frac{u_3}{h}\right) + \left(x_2^2 - 1\right) u_2$$

$$eq_3 := x_3^2 \left(\frac{u_2}{h^2} - \frac{2\,u_3}{h^2} + \frac{u_4}{h^2}\right) + x_3 \left(-\frac{1}{2}\,\frac{u_2}{h} + \frac{1}{2}\,\frac{u_4}{h}\right) + \left(x_3^2 - 1\right) u_3$$

$$eq_4 := x_4^2 \left(\frac{u_3}{h^2} - \frac{2\,u_4}{h^2} + \frac{u_5}{h^2}\right) + x_4 \left(-\frac{1}{2}\,\frac{u_3}{h} + \frac{1}{2}\,\frac{u_5}{h}\right) + \left(x_4^2 - 1\right) u_4$$

$$eq_5 := x_5^2 \left(\frac{u_4}{h^2} - \frac{2\,u_5}{h^2} + \frac{1}{h^2}\right) + x_5 \left(-\frac{1}{2}\,\frac{u_4}{h} + \frac{1}{2\,h}\right) + \left(x_5^2 - 1\right) u_5$$

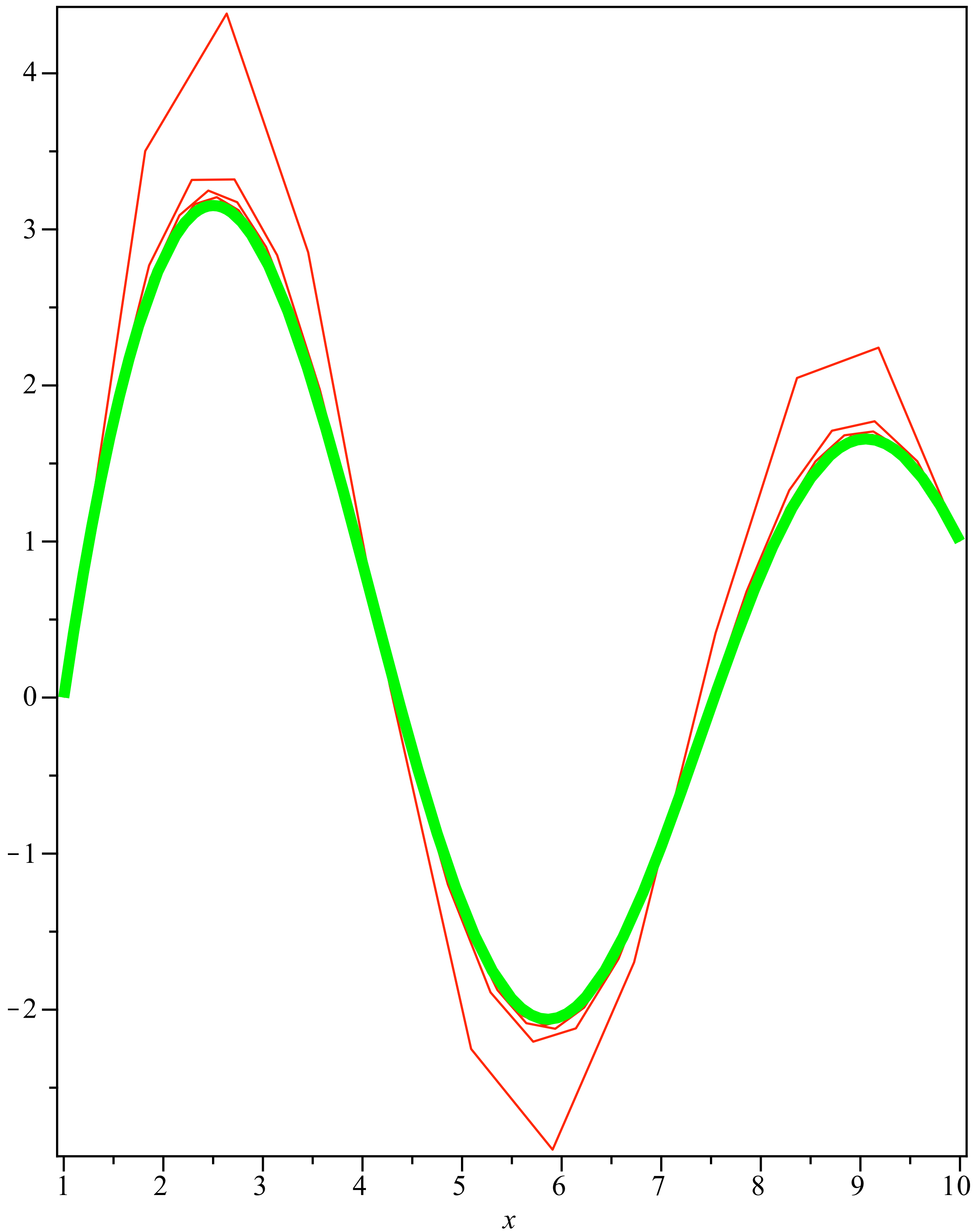
It is not difficulate to get MAPLE to solve such a system. This is what is done in BVP_solver, which takes the number of interior lattice points M as its argument. It returns the numerical solution of the BVP as a list of lists:

```
> BVP_solver := proc(M)
     local X, h, u, i, eq, u_sol:
     X := j -> evalf(1 + 9*j/(M+1));                        # X(j) is the
  x-coord of the jth lattice point
```

```
   h := X(1)-X(0);                                   # h is the
lattice spacing
   u[0] := 0:                                        # the first BC
   u[M+1] := 1:                                      # the second
BC
   for i from 1 to M do:                             # this loop
generates the linear system to be solved
      eq[i] := Stencil(X(i),u[i-1],u[i],u[i+1],h):
   od;
   u_sol := LinearSolve(GenerateMatrix(
      convert(eq,list),[seq(u[i],i=1..M)]));            # solve
using LinearSolve
   [[X(0),0],seq([X(j),u_sol[j]],j=1..M),[X(M+1),1]]:  # the
output is a list of lists
end proc:
```

Here is a plot of the output of BVP_solver for sucessively larger values of M (red) compared to the analytical solution (green).

```
> plot([seq(BVP_solver(i*10),i=1..5),rhs(analytic_sol)],x=1..10,
color=[red$5,green],thickness=[0$5,5],axes=boxed);
```



The numerical solution appears to be approaching the analytical solution as M gets larger. Here is a moving illustrating how the numerical solution improves with increasing M:

```
> p := 'p':
  for i from 1 to 30 do:
     P := plot([BVP_solver(i),rhs(analytic_sol)],x=1..10,-6..6,
  color=[red,green],axes=boxed):
     Q := textplot([1.5,5,cat(`M = `,i)],align={right}):
     p[i] := display([P,Q]):
  od:
```

```
display(convert(p,list),insequence=true);
```