```
> restart;
with(PDEtools):
with(plots):
with(LinearAlgebra):
Digits := 14:
```

Boundary value problem for ODEs

Solution using `dsolve`

```
> ode := x^2*diff(u(x),x,x)+x*diff(u(x),x)+(x^2-1)*u(x);
BCs := u(left)=a,u(right)=b;
a := 1;
b := -1;
left := 1;
right := 10;
```

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

$$BCs := u(left) = a, u(right) = b$$

$$a := 1$$

$$b := -1$$

$$left := 1$$

$$right := 10 \quad \textbf{(1.1}$$

```
> sol[exact] := subs(factor(dsolve([ode,BCs])),u(x));
```

$$sol_{exact} := \quad \textbf{(1.2}$$

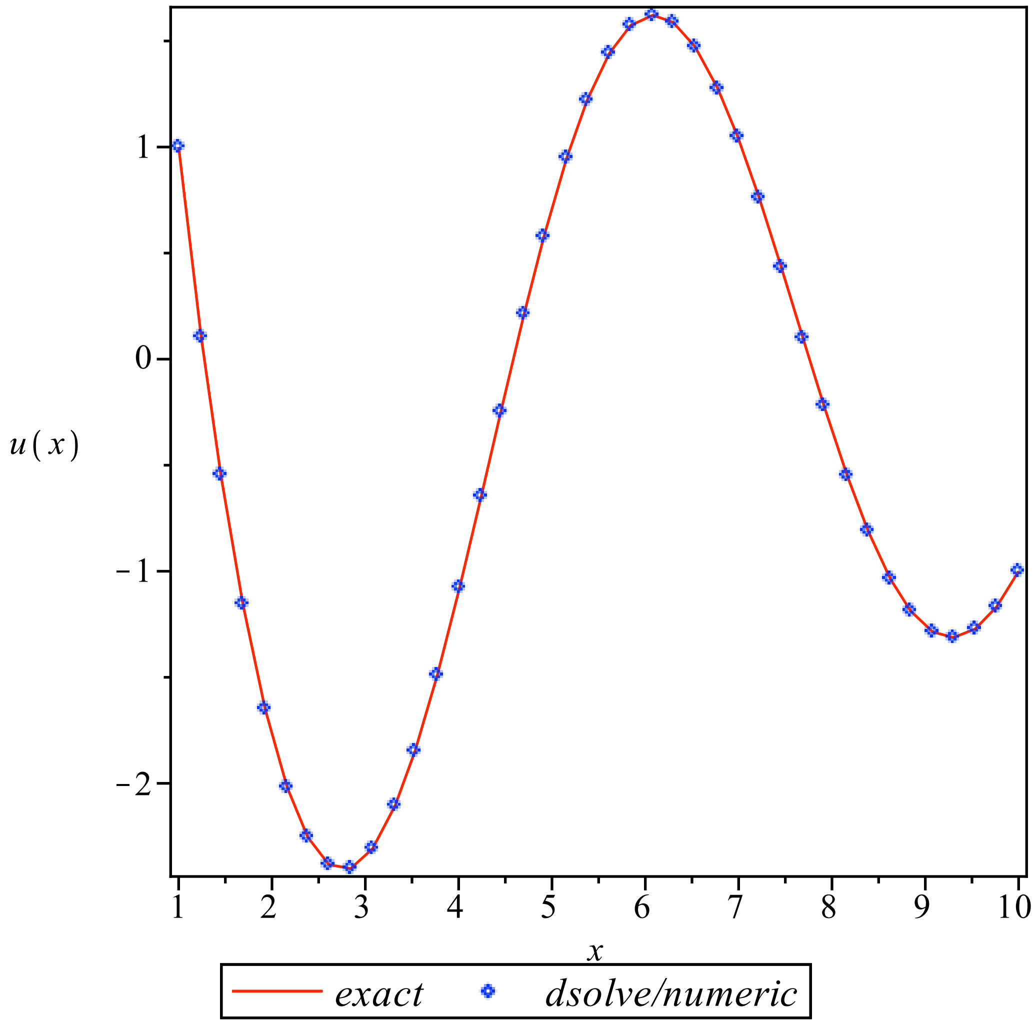
$$-\frac{1}{\mathrm{BesselY}(1, 1)\, \mathrm{BesselJ}(1, 10) - \mathrm{BesselY}(1, 10)\, \mathrm{BesselJ}(1, 1)} \left(\mathrm{BesselJ}(1, x)\, \mathrm{BesselY}(1, 1) + \mathrm{BesselJ}(1, x)\, \mathrm{BesselY}(1, 10) - \mathrm{BesselY}(1, x)\, \mathrm{BesselJ}(1, 1) - \mathrm{BesselY}(1, x)\, \mathrm{BesselJ}(1, 10)\right)$$

```
> sol[dsolve.numeric] := subs(dsolve([ode,BCs],numeric,output=listprocedure),u(x));
```

$$sol_{dsolve.numeric} := \mathbf{proc}(x) \;...\; \mathbf{end\ proc} \quad \textbf{(1.3}$$

```
> plot([sol[exact](x),sol[dsolve.numeric](x)],x=left..right,axes=boxed,numpoints=40,
adaptive=false,style=[line,point],color=[red,blue],legend=[exact,`dsolve/numeric`],labels=
```

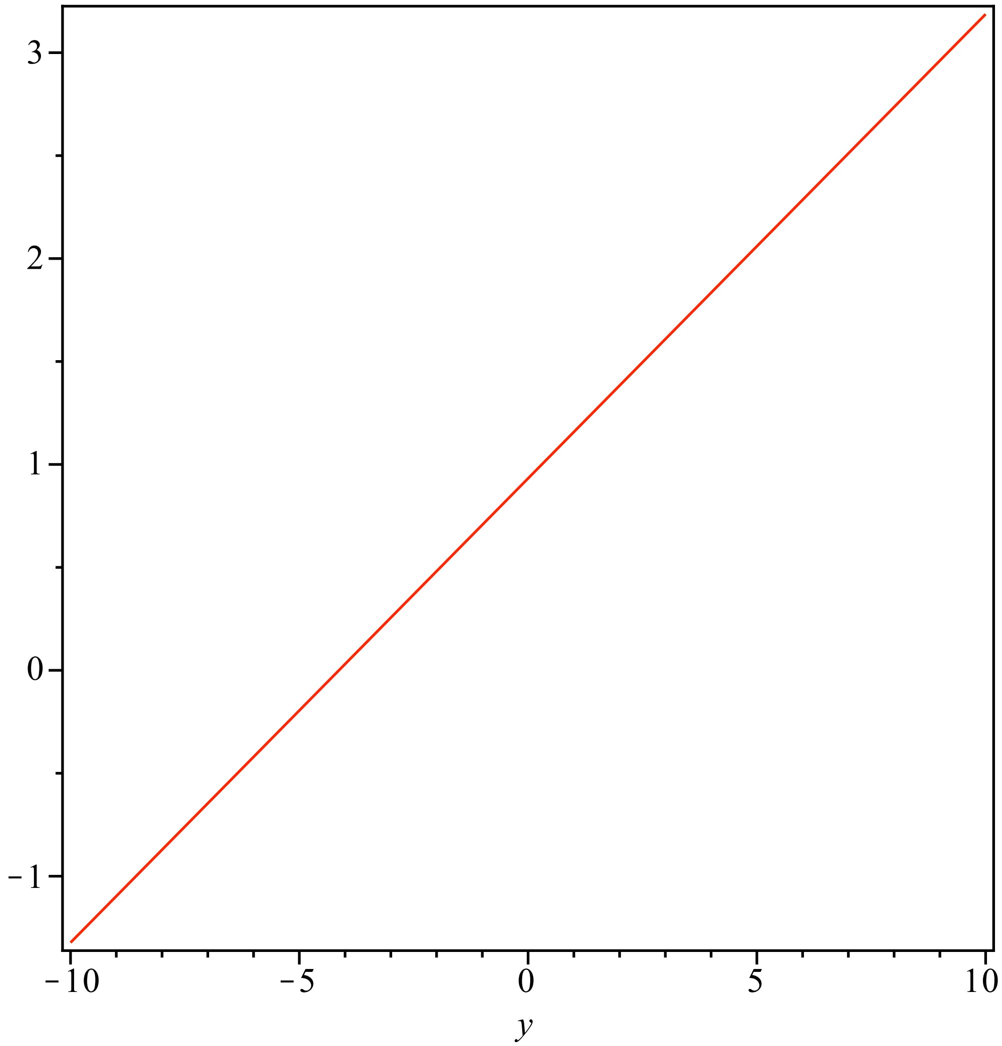
```
[x,u(x)]);
```



Solution using the shooting method

```
> F := proc(y)
     local ICs:
     if (type(y,numeric)) then:
        ICs := u(left)=a,D(u)(left)=y:
```

```
        dsolve([ode,ICs],numeric,output=Array([right]))[2][1][1,2]:
    else:
        'procname'(args):
    fi:
end proc:
> plot(F(y)-b,y=-10..10,axes=boxed);
```



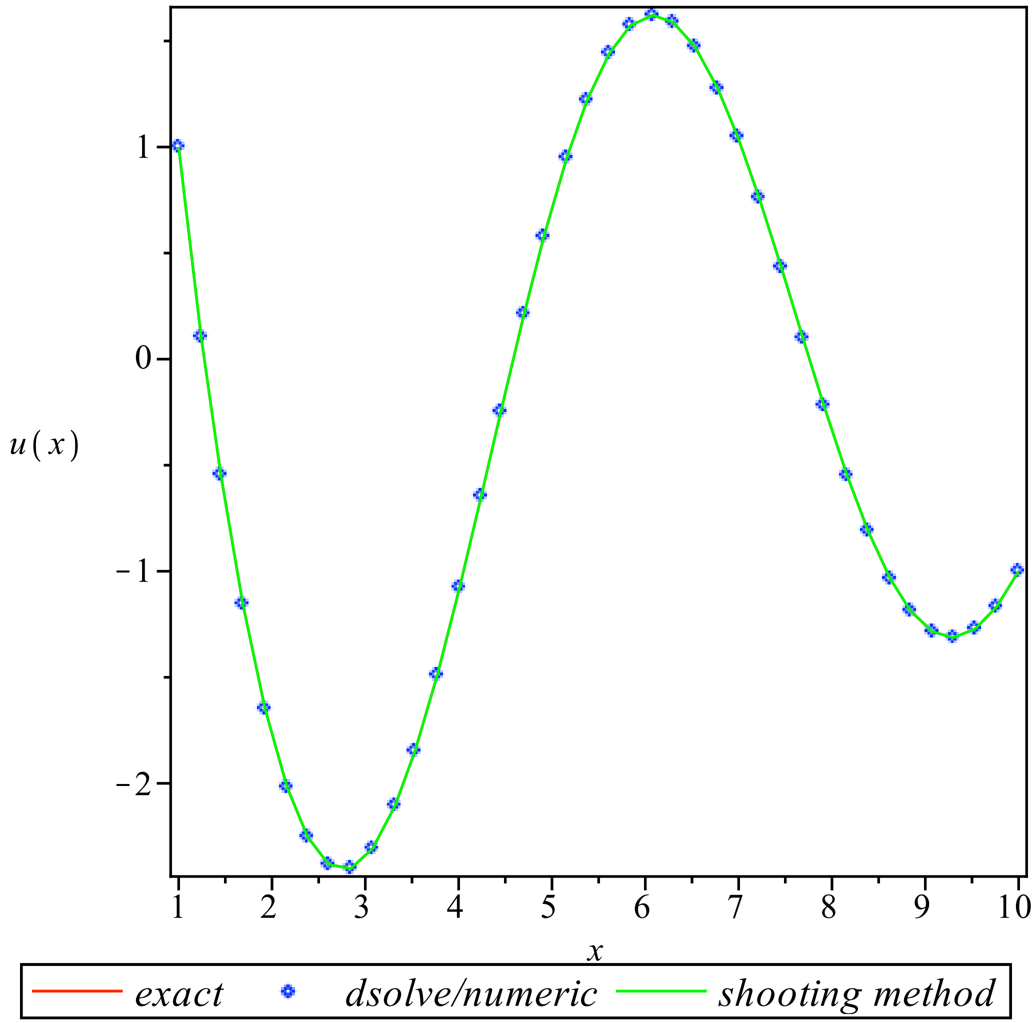

```
> y_sol := fsolve(F(y)-b);
```

$$y\_sol := -4.1343709721299 \quad \textbf{(2.1}$$

```
> ICs := u(left)=a,D(u)(left)=y_sol:
  sol[shooting] := subs(dsolve([ode,ICs],numeric,output=listprocedure),u(x));
```

$sol_{shooting}$:= **proc**(x) ... **end proc** (2.2

```
> plot([sol[exact](x),sol[dsolve.numeric](x),sol[shooting](x)],x=left..right,axes=boxed,
  numpoints=40,adaptive=false,style=[line,point,line],thickness=[1,0,0],color=[red,blue,
  green],legend=[exact,`dsolve/numeric`,`shooting method`],labels=[x,u(x)]);
```



Solution using matrix methods for linear problems

```
> GenerateStencil := proc(F,N,{orientation:=center,stepsize:=h,showorder:=true,showerror:=
  false})
      local vars, f, ii, Degree, stencil, Error, unknowns, Indets, ans, Phi, r, n, phi;

      Phi := convert(F,D);
      vars := op(Phi);
      n := PDEtools[difforder](Phi);
      f := op(1,op(0,Phi));
      if (nops([vars])<>1) then:
        r := op(1,op(0,op(0,Phi)));
      else:
        r := 1;
      fi:
      phi := f(vars);
      if (orientation=center) then:
        if (type(N,odd)) then:
          ii := [seq(i,i=-(N-1)/2..(N-1)/2)];
        else:
          ii := [seq(i,i=-(N-1)..(N-1),2)];
        fi;
      elif (orientation=left) then:
        ii := [seq(i,i=-N+1..0)]:
      elif (orientation=right) then:
        ii := [seq(i,i=0..N-1)]:
      fi;
      stencil := add(a[ii[i]]*subsop(r=op(r,phi)+ii[i]*stepsize,phi),i=1..N);
      Error := D[r$n](f)(vars) - stencil;
      Error := convert(series(Error,stepsize,N),polynom);
      unknowns := {seq(a[ii[i]],i=1..N)};
      Indets := indets(Error) minus {vars} minus unknowns minus {stepsize};
      Error := collect(Error,Indets,'distributed');
      ans := solve({coeffs(Error,Indets)},unknowns);
      if (ans=NULL) then:
         print(`Failure: try increasing the number of points in the stencil`);
         return NULL;
      fi:
      stencil := subs(ans,stencil);
      Error := convert(series(`leadterm`(D[r$n](f)(vars) - stencil),stepsize,N+20),polynom);
      Degree := degree(Error,stepsize);
```

```
      if (showorder) then:
         print(cat(`This stencil is of order `,Degree));
      fi:
      if (showerror) then:
         print(cat(`This leading order term in the error is `,Error));
      fi:
      convert(D[r$n](f)(vars) = stencil,diff);

   end proc:
> substencil[1] := GenerateStencil(diff(u(x),x),2);
  substencil[2] := GenerateStencil(diff(u(x),x,x),3);
```

This stencil is of order 2

$$substencil_1 := \frac{d}{dx}\, u(x) = -\frac{1}{2}\,\frac{u(x-h)}{h} + \frac{1}{2}\,\frac{u(x+h)}{h}$$

This stencil is of order 2

$$substencil_2 := \frac{d^2}{dx^2}\, u(x) = \frac{u(x-h)}{h^2} - \frac{2\,u(x)}{h^2} + \frac{u(x+h)}{h^2} \qquad (3.1$$

```
> stencil := subs(substencil[2],substencil[1],ode);
```

$$stencil := x^2\left(\frac{u(x-h)}{h^2} - \frac{2\,u(x)}{h^2} + \frac{u(x+h)}{h^2}\right) + x\left(-\frac{1}{2}\,\frac{u(x-h)}{h} + \frac{1}{2}\,\frac{u(x+h)}{h}\right) + \left(x^2-1\right) u(x) \qquad (3.2$$

```
> stencil := subs(u(x-h)=u[i-1],u(x)=u[i],u(x+h)=u[i+1],x=x[i],stencil);
```

$$stencil := x_i^2\left(\frac{u_{i-1}}{h^2} - \frac{2\,u_i}{h^2} + \frac{u_{i+1}}{h^2}\right) + x_i\left(-\frac{1}{2}\,\frac{u_{i-1}}{h} + \frac{1}{2}\,\frac{u_{i+1}}{h}\right) + \left(x_i^2-1\right) u_i \qquad (3.3$$

```
> stencil := unapply(stencil,u,x,i,h);
```

$$stencil := (u, x, i, h) \rightarrow x_i^2\left(\frac{u_{i-1}}{h^2} - \frac{2\,u_i}{h^2} + \frac{u_{i+1}}{h^2}\right) + x_i\left(-\frac{1}{2}\,\frac{u_{i-1}}{h} + \frac{1}{2}\,\frac{u_{i+1}}{h}\right) + \left(x_i^2-1\right) u_i \qquad (3.4$$

```
> M := 4;
  a := 'a':
  b := 'b':
  u[0] := a;
  u[M+1] := b;
  sys := Vector([seq(stencil(u,x,i,h),i=1..M)]);
  GenerateMatrix(convert(sys,list),[seq(u[i],i=1..M)]);
  u := 'u':
```

$$M := 4$$

$$u_0 := a$$

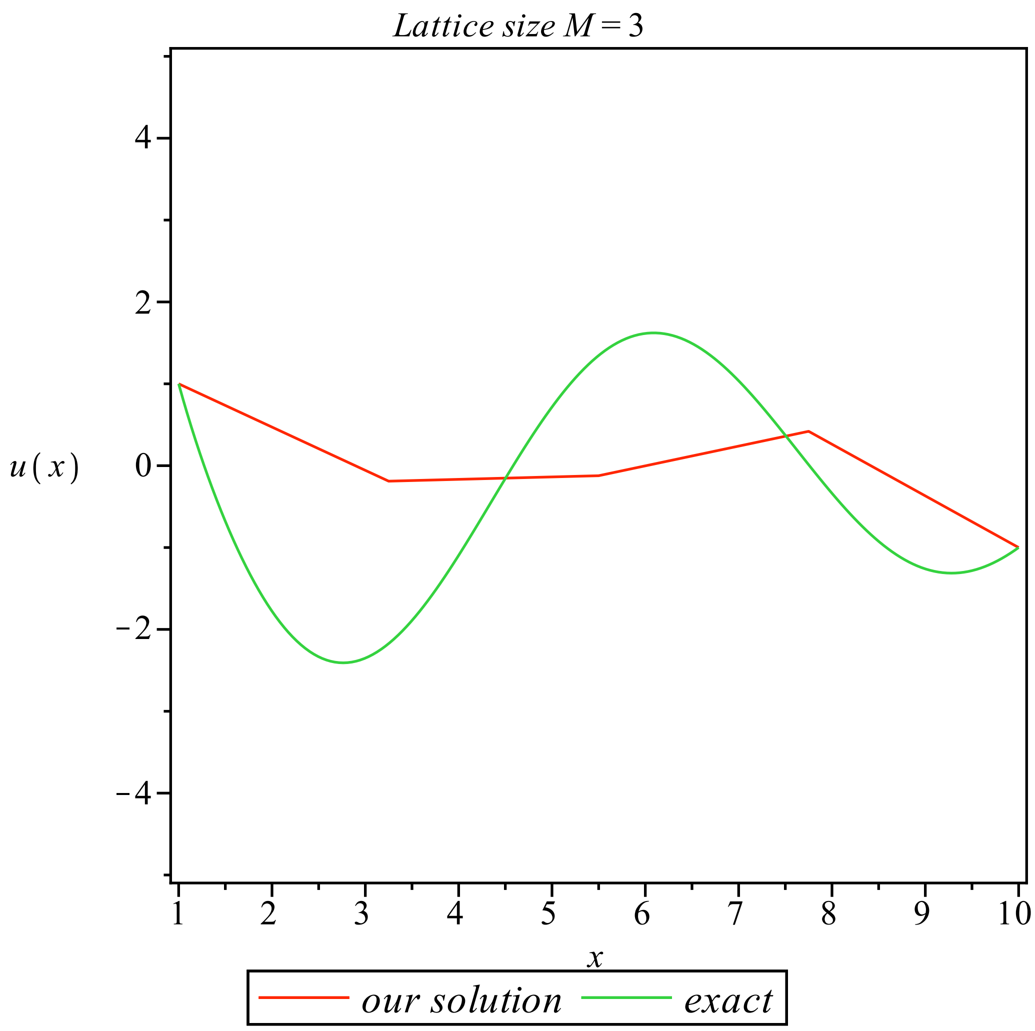
$$u_5 := b$$

$$sys := \begin{bmatrix} x_1^2 \left(\frac{a}{h^2} - \frac{2\,u_1}{h^2} + \frac{u_2}{h^2}\right) + x_1 \left(-\frac{1}{2}\,\frac{a}{h} + \frac{1}{2}\,\frac{u_2}{h}\right) + \left(x_1^2 - 1\right) u_1 \\ x_2^2 \left(\frac{u_1}{h^2} - \frac{2\,u_2}{h^2} + \frac{u_3}{h^2}\right) + x_2 \left(-\frac{1}{2}\,\frac{u_1}{h} + \frac{1}{2}\,\frac{u_3}{h}\right) + \left(x_2^2 - 1\right) u_2 \\ x_3^2 \left(\frac{u_2}{h^2} - \frac{2\,u_3}{h^2} + \frac{u_4}{h^2}\right) + x_3 \left(-\frac{1}{2}\,\frac{u_2}{h} + \frac{1}{2}\,\frac{u_4}{h}\right) + \left(x_3^2 - 1\right) u_3 \\ x_4^2 \left(\frac{u_3}{h^2} - \frac{2\,u_4}{h^2} + \frac{b}{h^2}\right) + x_4 \left(-\frac{1}{2}\,\frac{u_3}{h} + \frac{1}{2}\,\frac{b}{h}\right) + \left(x_4^2 - 1\right) u_4 \end{bmatrix}$$

$$\begin{bmatrix} x_1^2 - 1 - \frac{2\,x_1^2}{h^2} & \frac{1}{2}\,\frac{x_1}{h} + \frac{x_1^2}{h^2} & 0 & 0 \\ -\frac{1}{2}\,\frac{x_2}{h} + \frac{x_2^2}{h^2} & x_2^2 - 1 - \frac{2\,x_2^2}{h^2} & \frac{x_2^2}{h^2} + \frac{1}{2}\,\frac{x_2}{h} & 0 \\ 0 & -\frac{1}{2}\,\frac{x_3}{h} + \frac{x_3^2}{h^2} & x_3^2 - 1 - \frac{2\,x_3^2}{h^2} & \frac{x_3^2}{h^2} + \frac{1}{2}\,\frac{x_3}{h} \\ 0 & 0 & -\frac{1}{2}\,\frac{x_4}{h} + \frac{x_4^2}{h^2} & x_4^2 - 1 - \frac{2\,x_4^2}{h^2} \end{bmatrix}, \begin{bmatrix} -\frac{x_1^2\,a}{h^2} + \frac{1}{2}\,\frac{x_1\,a}{h} \\ 0 \\ 0 \\ -\frac{x_4^2\,b}{h^2} - \frac{1}{2}\,\frac{x_4\,b}{h} \end{bmatrix} \quad (3.5$$

```
> MatrixSolver := proc(M,x_range,a,b)
    local left, right, x, h, u, sys, sol:
    left := lhs(x_range):
    right := rhs(x_range):
    x := Vector([seq(left+i/(M+1)*(right-left),i=1..M)],datatype=float[8]);
    h := x[2]-x[1];
```

```
    u[0] := a;
    u[M+1] := b;
    sys := Vector([seq(stencil(u,x,i,h),i=1..M)]);
    sol := LinearSolve(GenerateMatrix(convert(sys,list),[seq(u[i],i=1..M)]));
    sol := convert(Matrix([x,sol]),listlist);
    sol := [[left,a],op(sol),[right,b]]:
end proc:
> p := 'p':
M := 'M':
for i from 3 to 35 do:
p[i] := plot([MatrixSolver(i,left..right,1,-1),sol[exact](x)],x=left..right,-5..5,axes=
boxed,title=typeset(`Lattice size `,M=i),legend=[`our solution`,`exact`],labels=[x,u(x)]);
od:
display(convert(p,list),insequence=true);
```



Solution using Newton's method for nonlinear problems

```
> i := 'i':
  left := 'left':
  right := 'right':
  ode := diff(u(x),x,x) + V(u(x))-f(x);
  BC := u(left) = a, u(right) = b;
```

$$ode := \frac{d^2}{dx^2} u(x) + V(u(x)) - f(x)$$

$$BC := u(left) = a, u(right) = b \quad (4.1$$

```
> stencil := unapply(subs(substencil[2],u(x-h)=u[i-1],u(x)=u[i],u(x+h)=u[i+1],f(x)=f[i],
  ode),u,f,i,h);
```

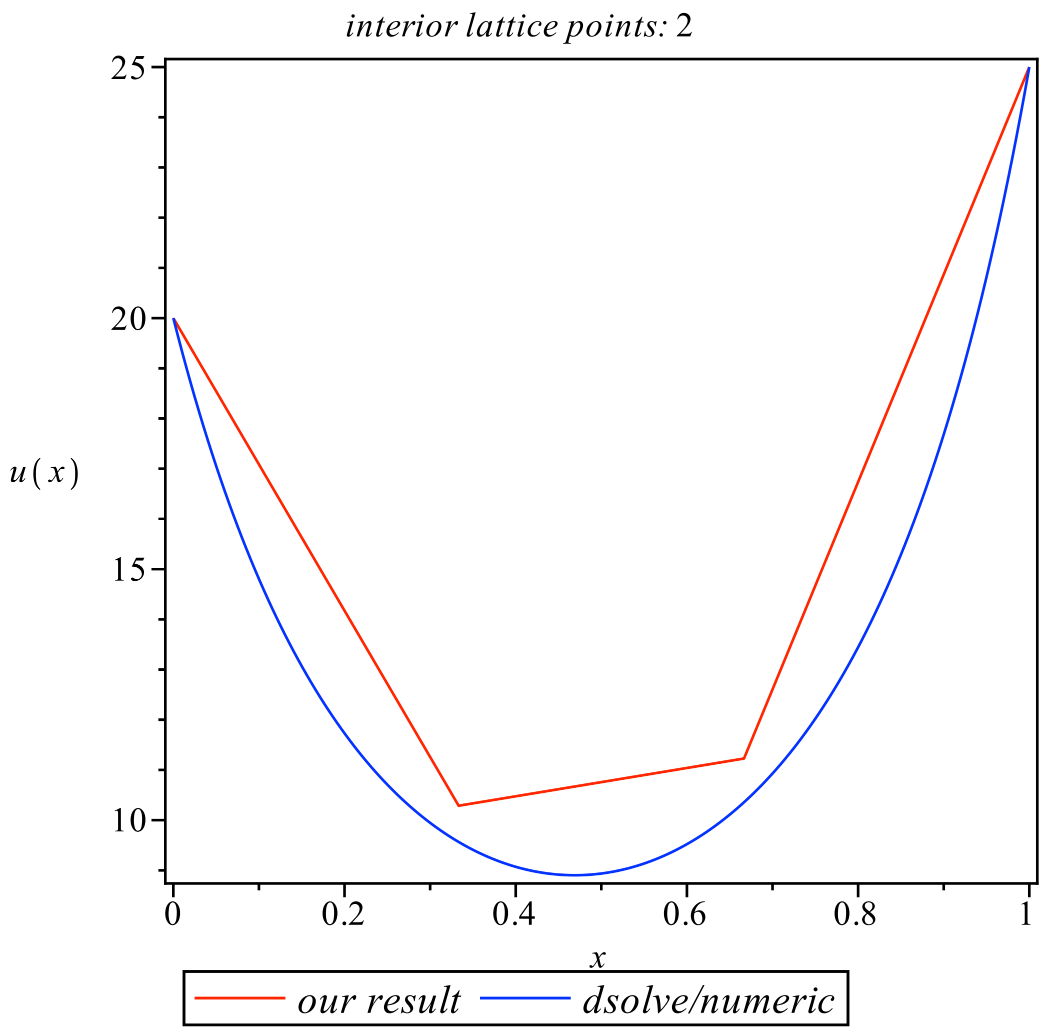
$$stencil := (u, f, i, h) \rightarrow \frac{u_{i-1}}{h^2} - \frac{2\,u_i}{h^2} + \frac{u_{i+1}}{h^2} + V\left(u_i\right) - f_i \quad (4.2$$

```
> NonlinearSolver := proc(M,x_range,a,b)
     local left, right, x, h, u, sys, sol, F, guess:
     left := lhs(x_range):
     right := rhs(x_range):
     x := Vector([seq(left+i/(M+1)*(right-left),i=1..M)],datatype=float[8]);
     h := x[2]-x[1];
     F := map(u->f(u),x);
     guess := map(u->a + (b-a)*(u-left)/(right-left),x):
     u[0] := a;
     u[M+1] := b;
     sys := [seq(stencil(u,x,i,h),i=1..M)];
     sol := Vector(Newton[vector](sys,[seq(u[i],i=1..M)],guess,1e-14)):
     [[left,a],op(convert(Matrix([x,sol]),listlist)),[right,b]]:
  end proc:

  Newton[vector] := proc(Sys,Vars,guess,eps)
       local N, vars, i, sys, linearization, linear_sys,
             A, b, X, CONTINUE, maxsteps, delta, Subs, d:
       N := nops(Sys):
       vars := convert(Vars,list):
       for i from 1 to N do:
          if (type(Sys[i],equation)) then:
            sys[i] := (lhs-rhs)(Sys[i]):
          else:
            sys[i] := Sys[i]:
          fi;
       od;
       sys := convert(sys,list);
       linearization := map(u->u=u+epsilon*d[u],vars);
       linear_sys := map(u->subs(epsilon=1,convert(series(u,epsilon,2),polynom)),subs
  (linearization,sys));
```

```
    A,b := LinearAlgebra[GenerateMatrix](linear_sys,map(u->d[u],vars));
    Subs := seq(vars[i]=q[i],i=1..N);
    A := unapply(subs(Subs,A),q);
    b := unapply(subs(Subs,b),q);
    X[0] := evalf(Vector(guess)):
    maxsteps := 50:
    CONTINUE := true:

    for i from 1 to maxsteps while (CONTINUE) do:
       X[i] := LinearAlgebra[LinearSolve](evalf(A(X[i-1])),evalf(b(X[i-1])))+X[i-1]:
       delta := X[i]-X[i-1]:
       delta := sqrt(delta^%T.delta)/N;
       if (delta<eps) then CONTINUE := false fi:
    od;

    if (CONTINUE) then:
       return `maximum number of iterations exceeded`:
    else:
       return convert(X[i-1],list):
    fi:

end proc:
> p := 'p':
f := x -> -x^3:
V := u -> u*(1-u):
left := 0:
right := 1:
a := 20:
b := 25;

sol[dsolve.numeric] := subs(dsolve([ode,BCs],numeric,output=listprocedure),u(x));
for M from 2 to 20 do:
p[M] := plot([NonlinearSolver(M,left..right,a,b),sol[dsolve.numeric](x)],x=left..right,
labels=[x,u(x)],legend=[`our result`,`dsolve/numeric`],axes=boxed,title=typeset(`interior
lattice points: `,M),color=[red,blue]);
od:
display(convert(p,list),insequence=true);
```

$$b := 25$$

$$sol_{dsolve.numeric} := \mathbf{proc}(x)\ \dots\ \mathbf{end\ proc}$$



Relaxation methods

Under construction...