Homework #3_Partial Solution, Winter 1999
MEAM 501 Differential Equation Methods in Mechanics

Consider a second order differential equation

$$-\frac{d}{dx}\left(\sin(4\pi x)\frac{du}{dx}\right)+0.1\cos(3\pi x)u=1+e^{-x(1-x)} \qquad \text{in} \qquad (0,1)$$

with the boundary condition

$$u(0)=-u(1)=0.1$$

ATTENTION !

Note that this is a typical ill-posed problem, that is, a solution does not exist in the sense that any approximation methods yield "infinity" solutions and they are all nonsense, because the coefficient $\sin(4\pi x)$ becomes negative in the given domain (0,1).

(1) Solve this problem by the element free Galerkin method which reproduce a linear polynomial.

Suppose that we shall recover up to cubic polynomials in the global sense, that is, suppose that an arbitrary function u is approximated by

$$u(x)\approx u_h(x)=\begin{Bmatrix}1 & x & x^2 & x^3\end{Bmatrix}\begin{Bmatrix}a_1\\a_2\\a_3\\a_4\end{Bmatrix}=\mathbf{p}^T\mathbf{a}$$

where the coefficient a is determined by the moving least square problem

$$\min_{\mathbf{a}}\frac{1}{2}\sum_{i=1}^{N}w_i(x)\left(\mathbf{p}(x_i)^T\mathbf{a}-u_i\right)^2$$

where $w_i(x)=\exp\left(-\alpha(x-x_i)^2\right)$, and u_i are the value of the function u at x_i. Taking the first variation of the functional of the moving least square problem, we have

$$\delta \frac{1}{2}\sum_{i=1}^{N} w_i(x)\left(\mathbf{p}(x_i)^T\mathbf{a}-u_i\right)^2 = \delta\mathbf{a}^T\sum_{i=1}^{N}\mathbf{p}(x_i)w_i(x)\left(\mathbf{p}(x_i)^T\mathbf{a}-u_i\right) = \delta\mathbf{a}^T(\mathbf{A}\mathbf{a}-\mathbf{b}) = 0$$

$$\Leftrightarrow \quad \mathbf{A}\mathbf{a}-\mathbf{b}=\mathbf{0} \quad \Leftrightarrow \quad \mathbf{a}=\mathbf{A}^{-1}\mathbf{b}$$

where

$$\mathbf{A}=\sum_{i=1}^{N}\mathbf{p}(x_i)w_i(x)\mathbf{p}(x_i)^T \quad , \quad \mathbf{b}=\sum_{i=1}^{N}\mathbf{p}(x_i)w_i(x)u_i$$

From this, the function is u is approximated by

$$u(x)\approx u_h(x)=\mathbf{p}(x)^T\mathbf{a}=\mathbf{p}(x)^T\mathbf{A}^{-1}\mathbf{b}=\sum_{i=1}^{N}\mathbf{p}(x)^T\mathbf{A}^{-1}\mathbf{p}(x_i)w_i(x)u_i=\sum_{i=1}^{N}N_i(x)u_i$$

where

$$N_i(x)=\mathbf{p}(x)^T\mathbf{A}^{-1}\mathbf{p}(x_i)w_i(x)=\mathbf{p}(x)^T\left(\sum_{j=1}^{N}\mathbf{p}(x_j)w_j(x)\mathbf{p}(x_j)^T\right)^{-1}\mathbf{p}(x_i)w_i(x)$$

Now we shall consider the Galerkin method to the boundary value problem. Assuming that v is an arbitrary function such that $v(0)=v(1)=0$, multiplying it into the both sides of the differential equation, and integrating it over the domain (0,1), we have

$$\int_0^1 v\left(-\frac{d}{dx}\left(\sin(4\pi x)\frac{du}{dx}\right)+0.1\cos(3\pi x)u\right)dx=\int_0^1 v\left(1+e^{-x(1-x)}\right)dx$$

Applying the integration by parts rule, we have

$$\int_0^1\left(\sin(4\pi x)\frac{dv}{dx}\frac{du}{dx}+0.1\cos(3\pi x)vu\right)dx=\int_0^1 v\left(1+e^{-x(1-x)}\right)dx \quad , \quad \forall v\ s.t.\ v(0)=v(1)=0$$

Substitution of

$$v=\mathbf{v}^T\mathbf{N} \quad \text{and} \quad u=\mathbf{N}^T\mathbf{u}$$

to the above formulation yields

$$\mathbf{v}^T \int_0^1 \left(\sin(4\pi x) \frac{d\mathbf{N}}{dx} \frac{d\mathbf{N}^T}{dx} + 0.1\cos(3\pi x)\mathbf{N}\mathbf{N}^T \right) dx\mathbf{u} = \mathbf{v}^T \int_0^1 \mathbf{N}\left(1 + e^{-x(1-x)}\right) dx \quad , \quad \forall \mathbf{v}$$

that is

$$\mathbf{K}\mathbf{u} = \mathbf{f}$$

with

$$\mathbf{K} = \int_0^1 \left(\sin(4\pi x) \frac{d\mathbf{N}}{dx} \frac{d\mathbf{N}^T}{dx} + 0.1\cos(3\pi x)\mathbf{N}\mathbf{N}^T \right) dx \qquad \text{and} \qquad \mathbf{f} = \int_0^1 \mathbf{N}\left(1 + e^{-x(1-x)}\right) dx$$

It is noted that the above integration is made by decomposing the domian (0,1) into a set of subdomains:

$$(0,1) = \bigcup_{e=1}^{E} (x_e, x_{e+1}) \quad , \quad x_1 = 0 \quad \text{and} \quad x_{E+1} = 1$$

and applying, for example, one-point Gaussian quadrature to each subinterval:

$$\int_0^1 \phi(x) dx = \sum_{e=1}^{E} \int_{x_e}^{x_{e+1}} \phi(x) dx \approx \sum_{e=1}^{E} (x_{e+1} - x_e) \phi\left(\frac{x_{e+1} + x_e}{2} \right).$$

The non-homogeneous boundary conditions

$$u(0) = \mathbf{N}(0)^T \mathbf{u} = 0.1 \qquad \text{and} \qquad u(1) = \mathbf{N}(1)^T \mathbf{u} = -0.1$$

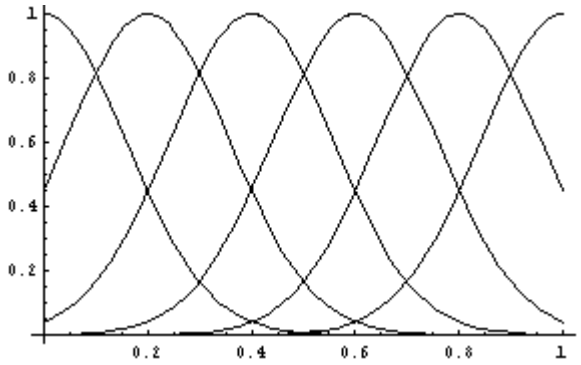
would be implemented by the penalty method. Then the following matrix must be added to **K** and **f**, respectively,

$$\mathbf{K}_p = \lambda\left(\mathbf{N}(0)\mathbf{N}(0)^T + \mathbf{N}(1)\mathbf{N}(1)^T\right) \qquad \text{and} \qquad \mathbf{f}_p = \lambda(0.1 * \mathbf{N}(0) - 0.1 * \mathbf{N}(1)).$$

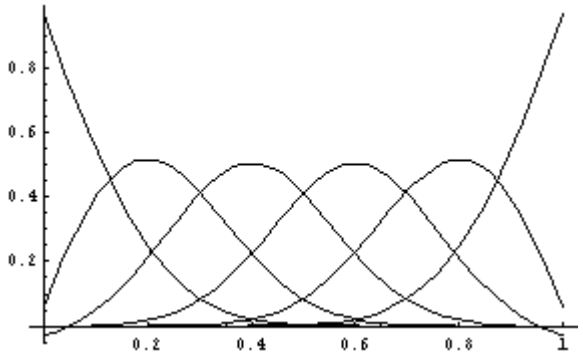
MATHEMATICA Implementation

```
m=2;
p={1,x};
Np=6;
xi=Table[(i-1)*1/(Np-1),{i,1,Np}];
alfa=20;
wj=Table[Exp[-alfa*(x-xi[[i]])^2],{i,1,Np}];
Plot[Release[wj],{x,0,1}]
pwp=Table[0,{i,1,m},{j,1,m}];
Do[pwpik=Sum[(p[[i]]/.{x->xi[[j]]})*wj[[j]]*(p[[k]]/.{x->xi[[j]]}),{j,1,Np}];
        pwp[[i,k]]=pwpik;pwp[[k,i]]=pwpik,{i,1,m},{k,i,m}]
pwpinv=Inverse[pwp];
Ni=Table[(p.pwpinv.(p/.{x->xi[[i]]}))*wj[[i]],{i,1,Np}];
Plot[Release[Ni],{x,0,1}]
```

Weighting Functions

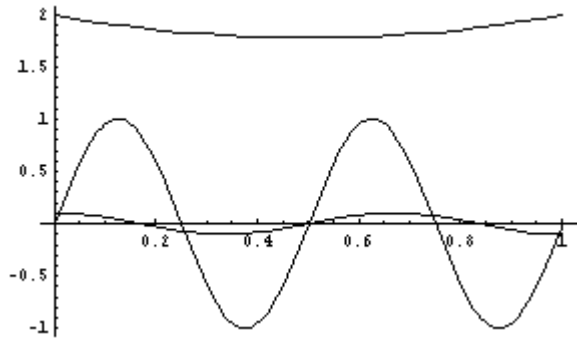


Basis Functions (which are very similar to the ones by the Bezier Splines)

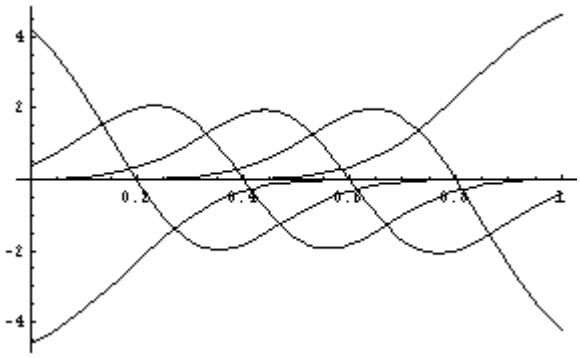



```
KM=Table[0,{i,1,Np},{j,1,Np}];
fv=Table[0,{i,1,Np}];
ax=Sin[4*Pi*x];
k0=0.1*Cos[3*Pi*x];
f=1+Exp[-x*(1-x)];
Plot[{ax,k0,f},{x,0,1}]
dNi=D[Ni,x];
Plot[Release[dNi],{x,0,1}]
Do[fv[[i]]=NIntegrate[Ni[[i]]*f,{x,0,1}];
        Print["f(",i,") = ",fv[[i]]];
        Do[KMij=NIntegrate[ax*dNi[[i]]*dNi[[j]]+k0*Ni[[i]]*Ni[[j]],{x,0,1}];
                Print["KM(",i,",",j,") = ",KMij];
                KM[[i,j]]=KMij;
                KM[[j,i]]=KMij,{j,i,Np}],{i,1,Np}]
lamuda=10^4;
Kp=lamuda*(Outer[Times,Ni/.{x->0},Ni/.{x->0}]+
                                        Outer[Times,Ni/.{x->1},Ni/.{x->1}]);
fp=lamuda*(0.1*(Ni/.{x->0})-0.1*(Ni/.{x->1}));
uv=LinearSolve[KM+Kp,fv+fp];
u=uv.Ni;
Plot[u,{x,0,1}]
```

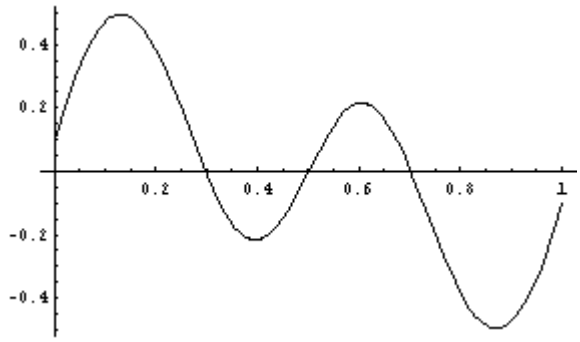
Coefficient Functions and the Right Hand Side of the Differential Equation



First Derivatives of the Basis Functions (It took a long time to calculate these.)



Solution Obtained (Required calculation is just too much.)



I have chosen only linear polynomials to be recovered globally, and introduce only 6 nodal points. However, required computing time using MATHEMATICA is excessive, and I cannot recommend this method in comparison with the standard FEM and FDM. At least I can say that MATHEMATICA implementation I have done is not recommended.

(2) Solve this problem by the wavelet Galerkin method with any appropriate wavelet function.

Since the wavelet functions are defined on the whole domain $\mathbf{R} = (-\infty, +\infty)$, we must

extend the domain $[0,1]$ in which the differential equation is defined. To this end, we shall make zero extension of the coefficient functions and the right hand side:

$$a_E(x)=\begin{cases}\sin(4\pi x) & \text{in} \quad (0,1)\\ 0 \quad \text{or} \quad \varepsilon & \text{in} \quad \text{otherwise}\end{cases}, \qquad k_E(x)=\begin{cases}0.1\cos(3\pi x) & \text{in} \quad (0,1)\\ 0 \quad \text{or} \quad \varepsilon & \text{in} \quad \text{otherwise}\end{cases}$$

$$f_E(x)=\begin{cases}1+e^{-x(1-x)} & \text{in} \quad (0,1)\\ 0 \quad \text{or} \quad \varepsilon & \text{in} \quad \text{otherwise}\end{cases}$$

Using the triangular scaling function and the wavelet function generated, we shall approximate the solution by

$$u(x)=\sum_{k=-N_1}^{k=+N_2} c_k\, 2^{j_0/2}\phi\left(2^{j_0}x-k\right)+\sum_{j=j_0}^{j=M}\sum_{k=-N_1}^{k=+N_2} d_{j,k}\, 2^{j/2}\psi\left(2^{j}x-k\right)$$

that is

$$u(x)=\sum_{k=-N_1}^{k=+N_2} c_k\, 2^{(M+1)/2}\phi\left(2^{(M+1)}x-k\right)$$

where

$$\phi(x)=\begin{cases}0 & \text{if} \quad \text{otherwise}\\ x & \text{if} \quad 0\le x\le 1\\ 2-x & \text{if} \quad 1\le x\le 2\end{cases}, \qquad \psi(x)=-\frac{1}{2}\phi(2x)+\phi(2x-1)-\frac{1}{2}\phi(2x-2)$$

and j_0 , N_1 , N_2 , M are appropriately specified positive integers. Here, we shall assume $j_0=3,\ N_1=3,\ N_2=20,\ M=j_0$. Here it is noted that the above triangular scaling function and the wavelets are not orthogonal. To make a similar expression with the first problem, we shall define

$$\mathbf{N}^T=\left\{\phi_{M+1,-N_1},\phi_{M+1,-N_1+1},...,\phi_{M+1,0},.....,\phi_{M+1,N_2}\right\}$$

and

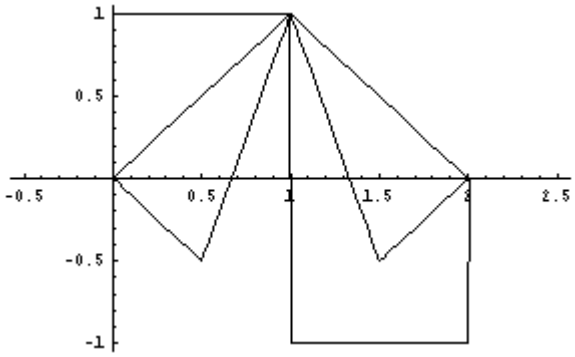
$$\mathbf{u}^T=\left\{c_{-N_1},c_{-N_1+1},.....,c_0,c_1,.....,c_{N_2}\right\}$$

We shall now develop a MATHEMATICA implementation:

(Define the scale function and its first derivative as well as the mother wavelet function)

```
sf[x_]:=If[x<0,0,If[x<1,x,If[x<2,2-x,0]]]
```

```
dsf[x_]:=If[x<0,0,If[x<1,1,If[x<2,-1,0]]]
mw[x_]:=-sf[2*x]/2-sf[2*x-2]/2+sf[2*x-1]
```



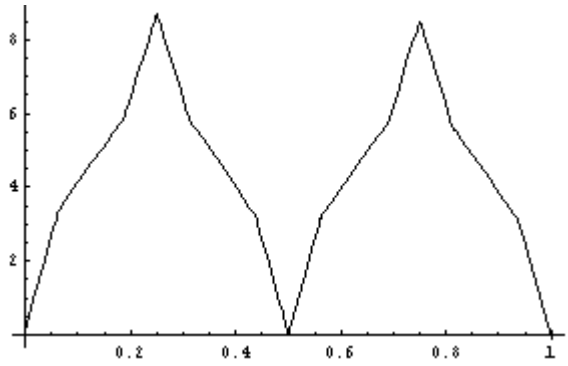
(Define a Quadrature Rule : Trapezoid Rule with 200 subintervals)

```
TrapezoidQuad[fint_,{x_,xmin_,xmax_}]:=
          Block[{nint,dx,ftrapezoid},
                    nint=201;
     dx=N[(xmax-xmin)/(nint-1)];
     ftrapezoid=N[0];
     ftrapezoid=dx*Sum[fint/.{x->xmin+dx*(i-1)},{i,1,nint}];
     ftrapezoid=ftrapezoid-0.5*dx*((fint/.{x->xmin})+(fint/.{x->xmax}));
                    ftrapezoid]
```

(Set up the Routine to Solve the Boundary Value Problem)

```
j0=3;
n1=-1;
n2=15;
M1=j0+1;
ep=0.00001;
ax=If[x<0,0,If[x<1,Sin[4*Pi*x],0]];
k0=If[x<0,0,If[x<1,0.1*Cos[3*Pi*x],0]];
fx=If[x<0,0,If[x<1,1+Exp[-x*(1-x)],0]];
Km=Table[TrapezoidQuad[
        ax*dsf[2^M1*x-i]*dsf[2^M1*x-j]*(2^M1)^2+k0*sf[2^M1*x-i]*sf[2^M1*x-j],{x,
          Min[{i/2^M1,j/2^M1}],Max[{(i+2)/2^M1,(j+2)/2^M1}]}],{i,n1,n2},{j,n1,
        n2}]
fv=Table[TrapezoidQuad[fx*sf[2^M1*x-i],{x,i/2^M1,(i+2)/2^M1}],{i,n1,n2}]
Km[[1,1]]=Km[[1,1]]+1/ep;
fv[[1]]=0.1*Km[[1,1]]
Km[[17,17]]=Km[[17,17]]+1/ep;
fv[[17]]=-0.1*Km[[17,17]];
uv=LinearSolve[Km,fv]
uh=uv.Table[sf[2^M1*x-i],{i,n1,n2}];
Plot[uh,{x,0,1}]
```

(Solution Profile: As we expect the solution is "nonsense", that is, it becomes almost infinite. This is a typical ill posed problem)



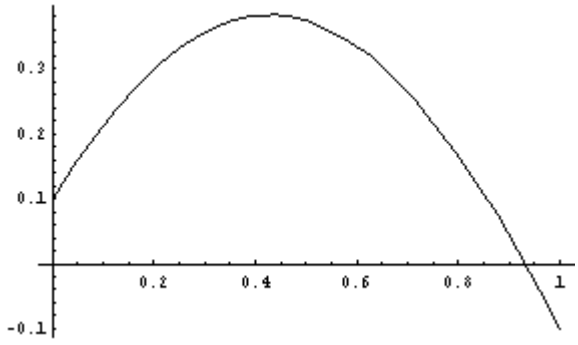
Checking Purpose

```
{0.100028, 0.174041, 0.237953, 0.290972,
 0.332398, 0.361662, 0.378371, 0.382338,
 0.37359, 0.35235, 0.318996, 0.274028, 0.218046,
 0.151758, 0.0759912, -0.00829148, -0.100001}
```

In order to check the above program can provide a reasonable solution to a well-posed problem, the following problem is solved:

$$-\frac{d}{dx}\left(1\frac{du}{dx}\right)+0.1\cos(3\pi x)u=1+e^{-x(1-x)} \qquad \text{in} \qquad (0,1)$$

with the same boundary condition.



As you can see from this solution, the wavelet base Galerkin method is identical to the finite element method, but the shape functions are defined globally by using appropriate scaling function.

(3) Solve this problem by the standard finite element or finite difference method, and make comparison with the element free and wavelet Galerkin methods.

We shall solve this by using the finite difference method. To set up the difference approximation, let us define

$$a(x)=\sin(4\pi x) \quad , \quad k(x)=0.1\cos(3\pi x) \quad , \quad f(x)=1+e^{-x(1-x)}$$

which yields the differential equation

$$-\frac{d}{dx}\left(a(x)\frac{du}{dx}\right)+k(x)u=f(x) \qquad \text{in} \qquad (0,1)$$

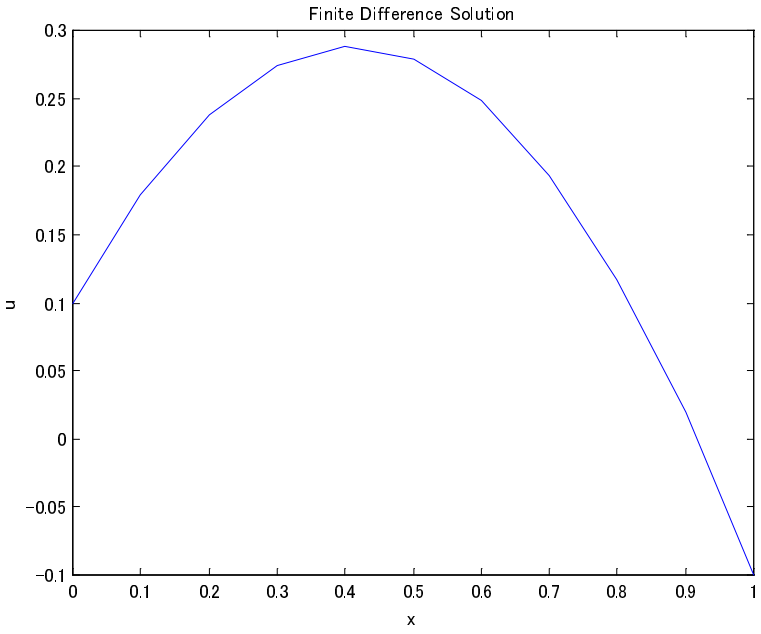
Applying the central difference scheme, we have

$$-\frac{a\left(x_{i+\frac{1}{2}}\right)\frac{u_{i+1}-u_i}{\Delta x}-a\left(x_{i-\frac{1}{2}}\right)\frac{u_i-u_{i-1}}{\Delta x}}{\Delta x}+k(x_i)u_i=f(x_i)$$

We shall develop a small MATLAB program for this problem. The following is the one to check whether a well-posed problem can be solved.

```
% Homework #3-Problem(3) : hw3p3
%
L=1;
nx=11;
dx=L/(nx-1);
x=0:dx:L;
% input functions
ax=sin(4*pi*x);
k0=0.1*cos(3*pi*x);
f=1+exp(-x.*(x-1));
plot(x,ax,x,k0,x,f)
pause
%
km=zeros(nx);
fv=zeros(nx,1);
for i=2:nx-1
   xip=(i-1)*dx+0.5*dx;
   %aip=sin(4*pi*xip);
   aip=1;
   %k0i=k0(i);
   k0i=0;
   fi=f(i);
   xim=xip-dx;
   %aim=sin(4*pi*xim);
   aim=1;
   km(i,i)=(aip+aim)/dx^2+k0i;
   fv(i)=fi;
   km(i,i-1)=-aim/dx^2;
   km(i,i+1)=-aip/dx^2;
end;
km(1,1)=1;km(nx,nx)=1;km(1,2)=0;km(2,1)=0;km(nx,nx-1)=0;km(nx-1,nx)=0;
```

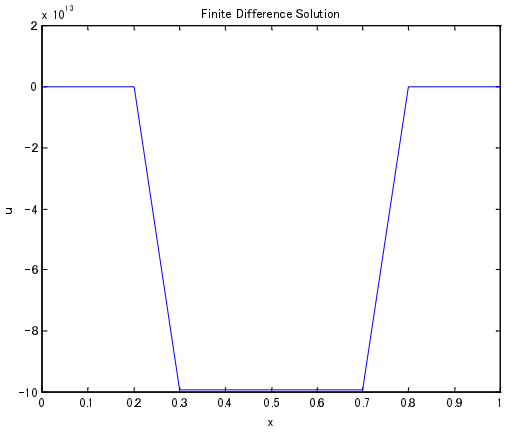
```
fv(1)=0.1;fv(nx)=-0.1;
%fv(2)=fv(2)+(sin(4*pi*dx*0.5)/dx^2)*0.1;
fv(2)=fv(2)+(1/dx^2)*0.1;
%fv(nx-1)=fv(nx-1)-(sin(4*pi*(L-0.5*dx))/dx^2)*0.1;
fv(nx-1)=fv(nx-1)-(1/dx^2)*0.1;
uv=km¥fv;
plot(x,uv)
xlabel('x')
ylabel('u')
title('Finite Difference Solution')
```



If we solve the original problem, MATLAB answers

Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 7.387819e-017.

$$-\frac{a\left(x_{i+\frac{1}{2}}\right)}{\Delta x^2}u_{i+1}+\left\{\frac{a\left(x_{i+\frac{1}{2}}\right)}{\Delta x^2}+\frac{a\left(x_{i-\frac{1}{2}}\right)}{\Delta x^2}+k(x_i)\right\}u_i-\frac{a\left(x_{i-\frac{1}{2}}\right)}{\Delta x^2}u_{i-1}=f(x_i)$$


Finite Difference Solution
x 10^13
2
0
−2
−4
−6
−8
−10
u
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x