Although I had managed to get the paper work done for the theta method, I was stuck on how to actually implement it. I also understand that I could have used a combination of the Implicit and Explicit methods to do so using Crank-Nicolson's method however I was stumped by it as well.

Discretization: $\dfrac{D_{i-\frac{1}{2}} \frac{\partial c}{\partial x}_{i-\frac{1}{2}} - D_{i+\frac{1}{2}} \frac{\partial c}{\partial x}_{i+\frac{1}{2}}}{\Delta x} - c_i + f(x)$

$$\frac{\partial c}{\partial x}_{i-\frac{1}{2}} = \frac{c_{i-1} - c_i}{\Delta x} \qquad \frac{\partial c}{\partial x}_{i+\frac{1}{2}} = \frac{c_i - c_{i+1}}{\Delta x}$$

$$\frac{1}{\Delta x^2}\left[D_{i-\frac{1}{2}} c_{i-1} - D_{i-\frac{1}{2}} c_i - D_{i+\frac{1}{2}} c_i + D_{i+\frac{1}{2}} c_{i+1}\right] - c_i + f(x) = \phi$$

$$\frac{1}{\Delta x^2}\left[D_{i-\frac{1}{2}} c_{i-1} - \left[D_{i-\frac{1}{2}} + D_{i+\frac{1}{2}} + \Delta x^2\right] c_i + D_{i+\frac{1}{2}} c_{i+1}\right] = -f(x)$$

For the explicit method:

$$\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left[D(x)\frac{\partial c}{\partial x}\right] - kc + f(x)$$

$$\frac{c_i^{t+1} - c_i^t}{\Delta t} = \frac{1}{\Delta x^2}\left[D_{i-\frac{1}{2}} c_{i-1}^t - \left(D_{i-\frac{1}{2}} + D_{i+\frac{1}{2}} + \Delta x^2\right) c_i^t + D_{i+\frac{1}{2}} c_{i+1}^t\right] + f(x)$$

$$c_i^{t+1} = \frac{\Delta t}{\Delta x^2}\left[D_{i-\frac{1}{2}} c_{i-1}^t - \left(D_{i-\frac{1}{2}} + D_{i+\frac{1}{2}} + \Delta x^2 - 1\right) c_i^t + D_{i+\frac{1}{2}} c_{i+1}^t\right] + f(x)\Delta t$$

For the implicit method

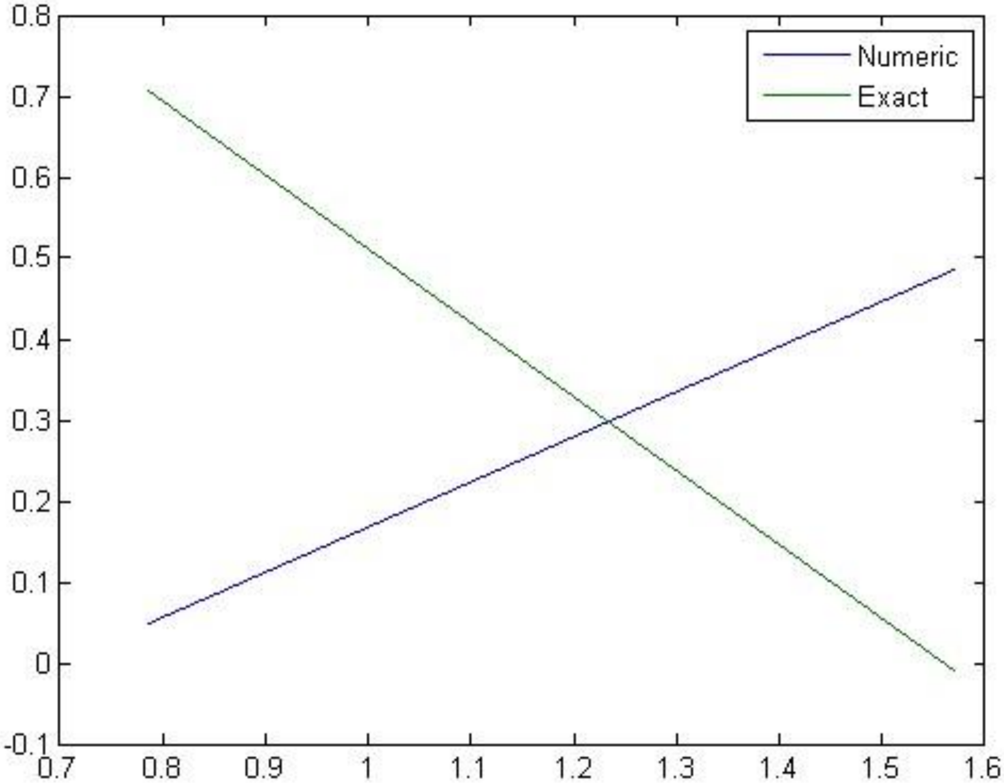
$$c_i^{t-1} = -\frac{\Delta t}{\Delta x^2}\left[D_{i-\frac{1}{2}} c_{i-1}^t - \left(D_{i-\frac{1}{2}} + D_{i+\frac{1}{2}} + \Delta x^2 + 1\right) c_i^t + D_{i+\frac{1}{2}} c_{i+1}^t\right] - f(x)$$

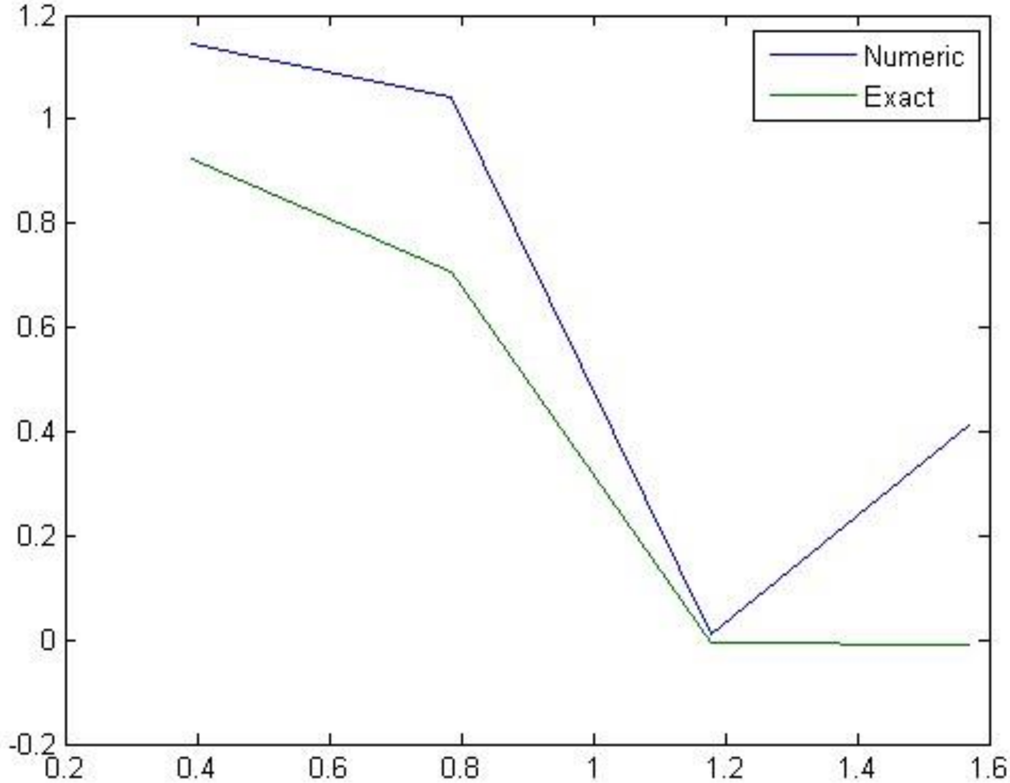
$$c_i^{t+1} - c_i^t = \Delta t \cdot \Theta\left[D_{i+\frac{1}{2}} c_{i+1}^{t+1} - \left[D_{i+\frac{1}{2}} + D_{i-\frac{1}{2}} - \Delta x^2 k\right] c_i^{t+1} + D_{i-\frac{1}{2}} c_{i-1}^{t+1} + \Delta x^2 f(x_i)\right] +$$
$$\Delta t(1-\Theta)\left[D_{i+\frac{1}{2}} c_{i+1}^t - \left[D_{i+\frac{1}{2}} + D_{i-\frac{1}{2}} - \Delta x^2 k\right] c_i^t + D_{i-\frac{1}{2}} c_{i-1}^t + \Delta x^2 f(x)\right]$$

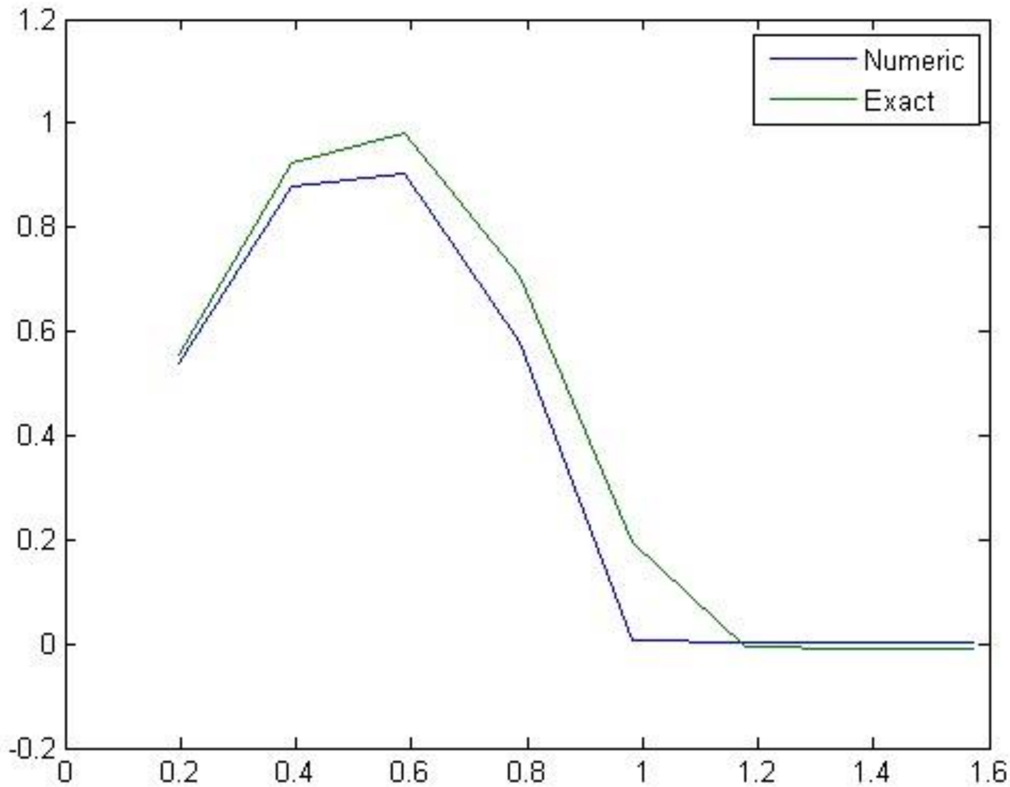
$$c_i^{t+1} = \Delta t\,\Theta\left[D_{i+\frac{1}{2}} c_{i+1}^{t+1} - \left[D_{i+\frac{1}{2}} + D_{i-\frac{1}{2}} - \Delta x^2\right] c_i^{t+1} + D_{i-\frac{1}{2}} c_{i-1}^{t+1} + f(x_i)\Delta x^2\right] +$$
$$\Delta t(1-\Theta)\left[D_{i+\frac{1}{2}} c_{i+1}^t - \left[D_{i+\frac{1}{2}} + D_{i-\frac{1}{2}} - \Delta x^2 - \Delta t(1-\Theta)\right] c_i^t + D_{i-\frac{1}{2}} c_{i-1}^t + \Delta x^2 f(x_i)\right]$$

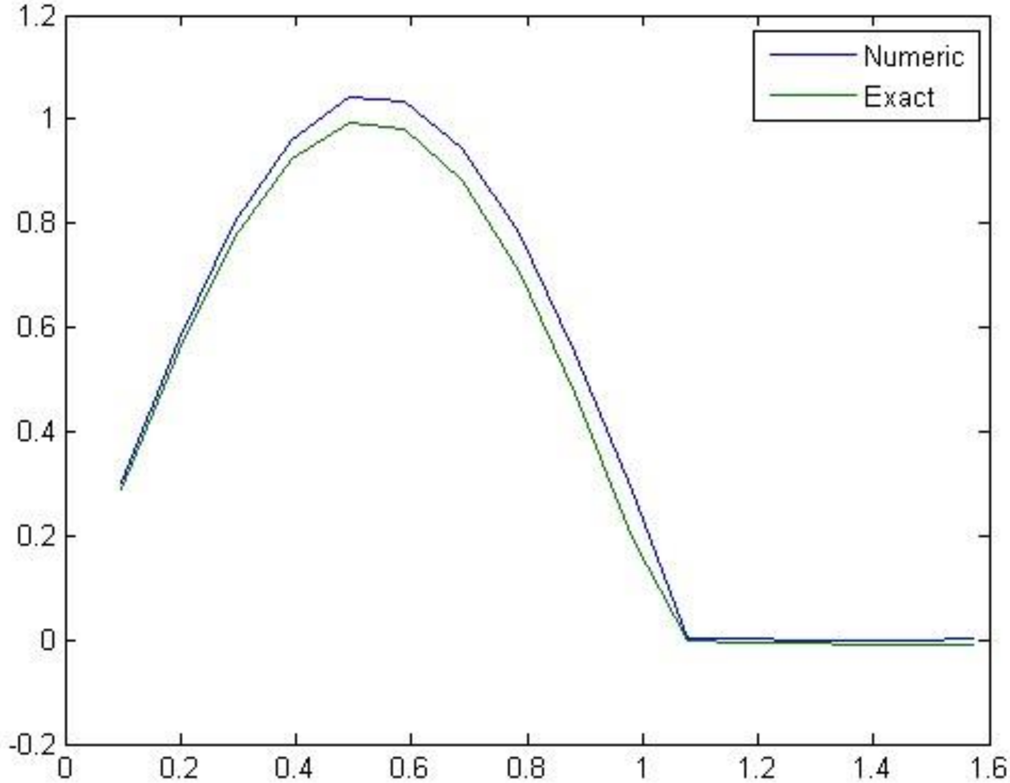
$$-\Delta t\,\Theta\left[D_{i+\frac{1}{2}} c_{i+1}^{t+1} - \left[D_{i+\frac{1}{2}} + D_{i-\frac{1}{2}} - \Delta x^2 + t\Theta\right] c_i^{t+1} + D_{i-\frac{1}{2}} c_{i-1}^{t+1} + \Delta x^2 f(x_i)\right]$$
$$= \Delta t(1-\Theta)\left[D_{i+\frac{1}{2}} c_{i+1}^t - \left[D_{i-\frac{1}{2}} + D_{i+\frac{1}{2}} - \Delta x^2 - \Delta t(1-\Theta)\right] c_i^t + D_{i-\frac{1}{2}} c_{i-1}^t + \Delta x^2 f(x_i)\right]$$

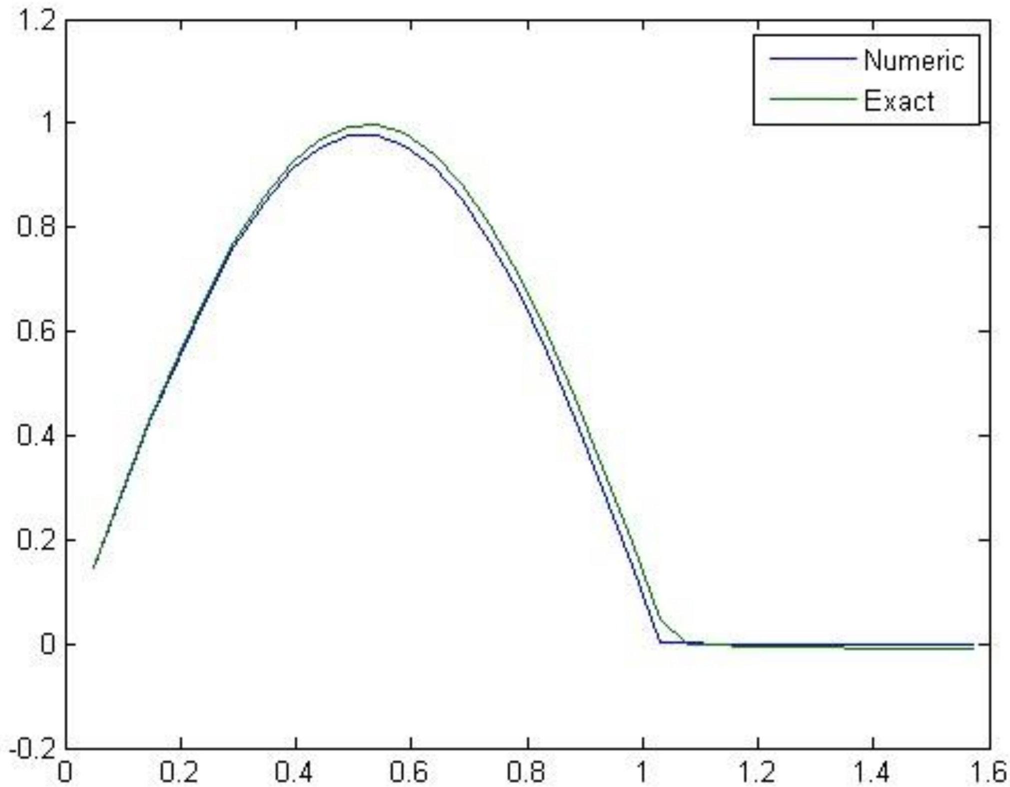
```
function [  ] = driver( )
%DRIVER Runs the code for HW6

for i = 1:8
figure(2^i)
    Solver(2^i, 0, pi/2)
end

PDE_Solver(pi/2, 10)




end
```

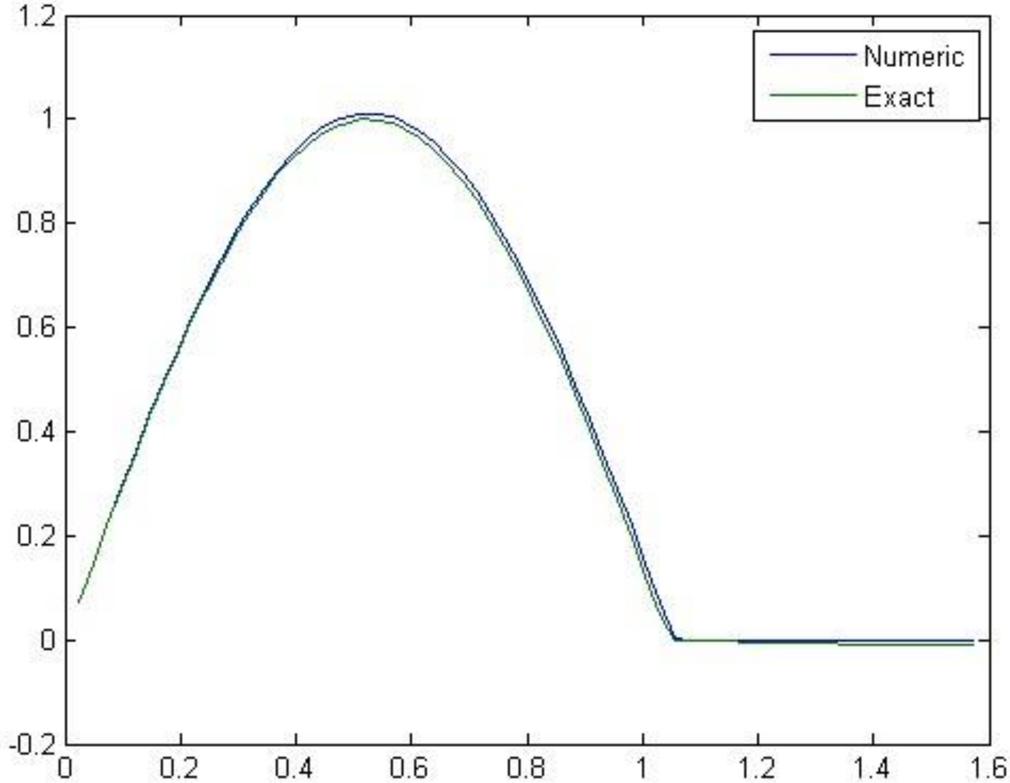


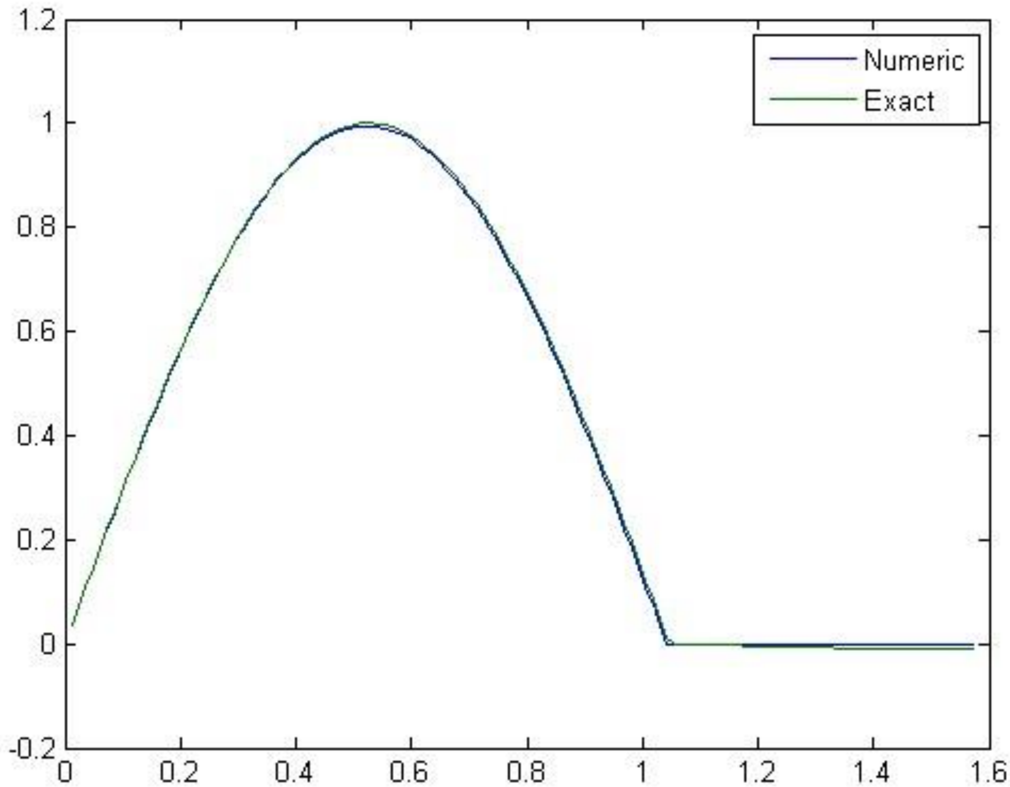

Numeric
Exact
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6

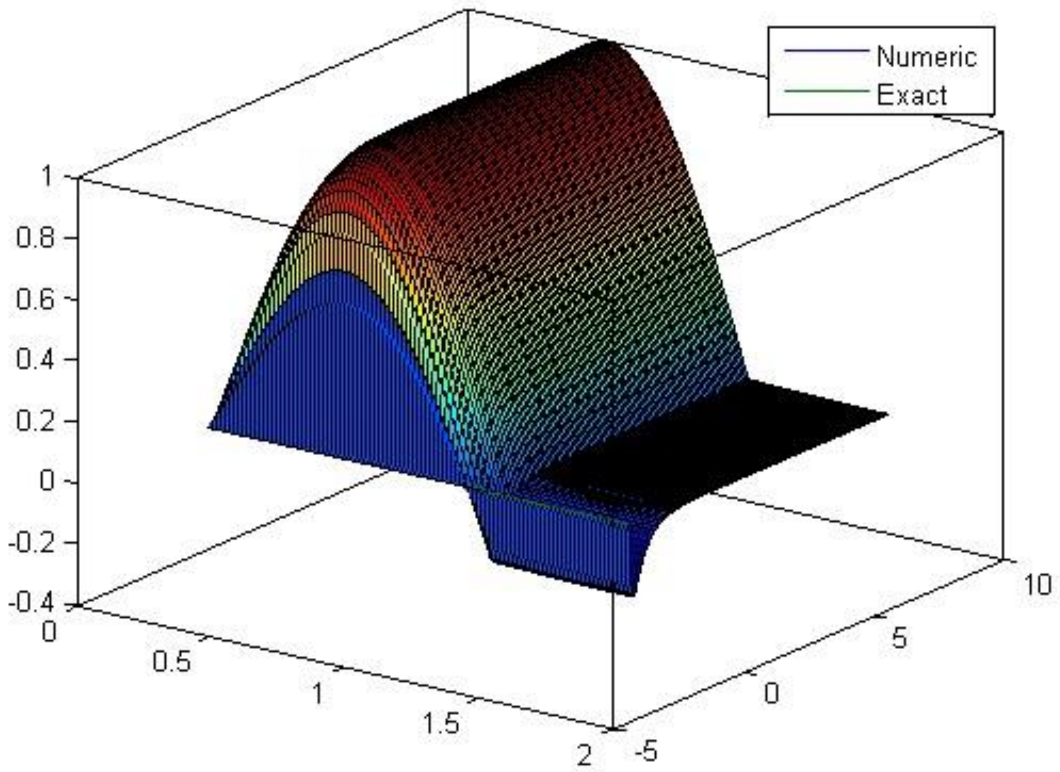

Numeric
Exact
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


Numeric
Exact
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


Numeric
Exact
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


Numeric
Exact
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6


Numeric
Exact
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6



Published with MATLAB® 7.14

```
function [  ] = PDE_Solver( L, T )
%PDE_SOLVER Uses PDEPE to solve the partial differential equation we are
% given

m = 0; x = linspace(0, L); t = linspace(0, T);

sol = pdepe(m, @pde_p2, @pde_ic, @pde_bc, x,

t); u = sol(:,:,1); surf(x,t,u)


end
```

Published with MATLAB® 7.14

```
function [ ] = Solver(steps, x0, L)
%SOLVER Solves the time-independent PDE given the number of steps,
% the initial starting position and the length of the bar.

dx = ((L)-x0)/steps;

div = (dx^2); X =

(x0+dx):dx:L;

for i = 1:steps
    M(i) = C_exact(x0+i*dx);
end

D = TriDiag(@D_x,@TD_Mid,@D_x, steps, x0, dx);

D = (1/div).*D;
F = -f_x(x0,steps,dx);
C = (D\F)';

plot(X,C,X,M) hold
all
legend('Numeric', 'Exact')
hold all


end
```

```
function
```



```
         [ Coeff ] = TriDiag( Top, Mid, Bot, steps, x0, stepsize )
%TRIDIAG Creates a tridiagonal matrix given a function for the top,
% middle and bottom diagonals, however the last spot is modified for
the % problem. Coeff = zeros(steps);

for i = 1:steps-1     dx_top =
(i-1/2)*stepsize;     dx_bot =
(i+1/2)*stepsize;
Coeff(i,i+1) = Top(x0+dx_top);
    Coeff(i, i) = Mid(x0, stepsize,
i);     Coeff(i+1,i) = Bot(x0+dx_bot);
end

dx = (steps-(3/2))*stepsize;
Coeff(steps,steps) = Bot(x0+dx)+stepsize^2;

end
```

Published with MATLAB® 7.14

```
function

         [ Mid ] = TD_Mid( x, dx, step )
%TD_MID This term is the coefficient of C_i, or the middle row of the
% tridiagonal matrix.

i1 = (step+1/2)*dx; i2 = (step-1/2)*dx;

Mid = -(D_x(x+i1) + D_x(x+i2) + (dx^2));

end
```

Published with MATLAB® 7.14

```
function

          [ Dx ] = D_x( x )
%D_X D(x) function

if x > (pi/3)
Dx = 100;
else      Dx =
1; end

end
```

Published with MATLAB® 7.14

```
function

         [ C ] = C_exact( x )
%C_EXACT Exact solution for C

if x > pi/3     C =
0.01*sin(3*x); else
C = sin(3*x); end

end
```

Published with MATLAB® 7.14

```
function


         [ F ] = f_x( x0, steps, stepsize )
%F_X is the given F(x) function

F = zeros(steps,1);

for i = 1:steps     X =
x0+(i*stepsize);     if X >
(pi/3)         F(i,1) =
9.01*sin(3*X);     else
F(i,1) = 10*sin(3*X);     end
end

end
```

Published with MATLAB® 7.14

```
function


         [ c,f,s ] = pde_p2( x, t, u, DuDx )
%PDE_P2 This is the PDE of part 2

if x > pi/3     f =
100*DuDx;     s = -
u+9.01*sin(3*x); else
f = DuDx;     s = -
u+10*sin(3*x); end

c = 1;

end
```

Published with MATLAB® 7.14

```
function


          [ pl, ql, pr, qr] = pde_bc(xl,ul,xr,ur,t)
%PDE_BC is the boundary conditions of the PDE

pl=ul;
ql=0;
pr=0;
qr=pi/2;

end
```

Published with MATLAB® 7.14

```
function [ u0 ] = pde_ic( x ) %PDE_IC
This is the PDE initial condition

u0 = 0;

end
```

Published with MATLAB® 7.14