Computational Mechanics Tools

PDE tool HW

Aren Khaloian

December 13 2019

Using the Matlab PDEtool for solving a PDE

MSc in Numerical methods for engineering

Objective:

The objective of the assignment is to find the answer for the given PDE equation using the Matlab PDEtool, for the given boundary conditions and t=10s and by refining the mesh used in the PDEtool compare the answers and see the convergence of the answers to the analytical solution.

$$u_t - \Delta u = f \qquad \text{in } \Omega = [0,1]^2,$$

where the source term is given by,

$$f(x,y,t) = -3e^{-3t}.$$

We consider an initial condition at $t = 0$:

$$u(x,y,t=0) = x^2 + xy - y^2 + 1,$$

$$\begin{aligned} u_n(x=0,y,t) &= -y, \\ u_n(x=1,y,t) &= 2+y, \\ u(x,y=0,t) &= x^2 + e^{-3t} \\ u_n(x,y=1,t) &= x-2, \end{aligned}$$

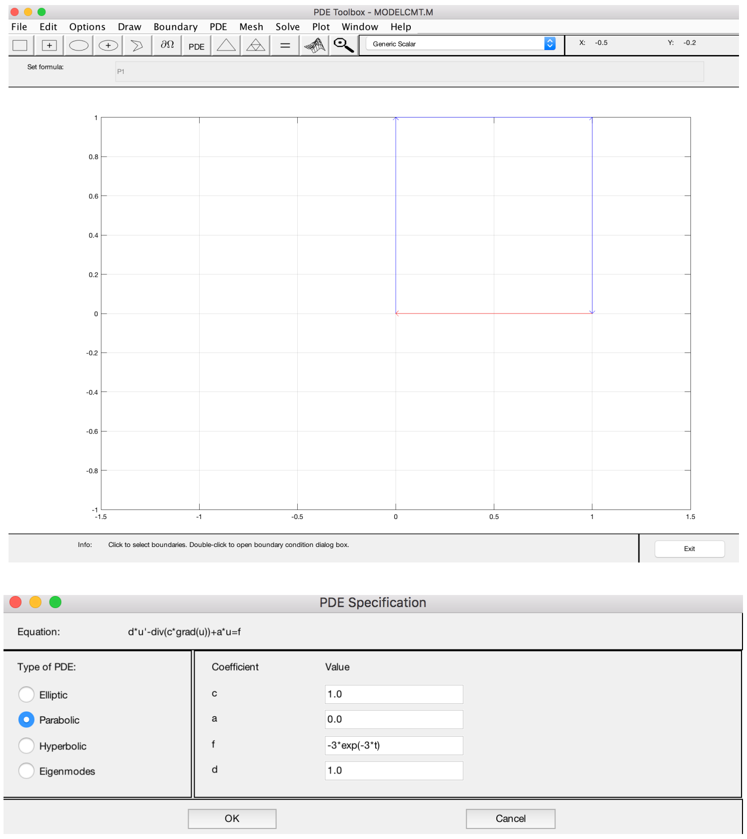
Fig1. The equation and the boundary conditions

We can see that we have a parabolic PDE with three Newman and one Dirichelet boundary conditions and the initial condition for the equation is given. The boundary is a square of 1 by 1.

Solution:

We start the problem by defining the equation and the boundary and initial conditions in PDEtool in Matlab and meshing the boundary so we can export the node coordinates,the connectivity matrix and the solution for each node. Afterwards we modify the error and convergence codes so we can evaluate the error of our numerical solution with the analytical one and plotting the error by the mesh size to evaluate the convergence of the numerical methods when we refine the mesh.

The defining of the equation in PDEtool:



Info: Click to select boundaries. Double-click to open boundary condition dialog box.

PDE Specification

Equation: d*u'-div(c*grad(u))+a*u=f

Type of PDE: Elliptic [✓ Parabolic] Hyperbolic Eigenmodes

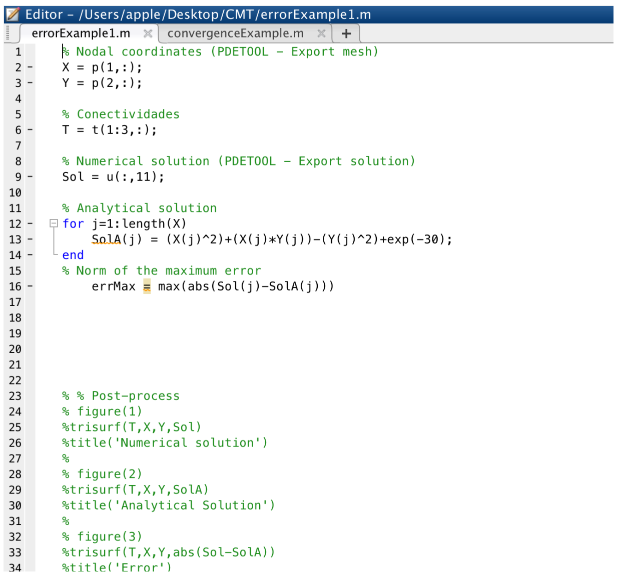
Coefficient	Value
c	1.0
a	0.0
f	-3*exp(-3*t)
d	1.0

Fig2. The geometry and the equation

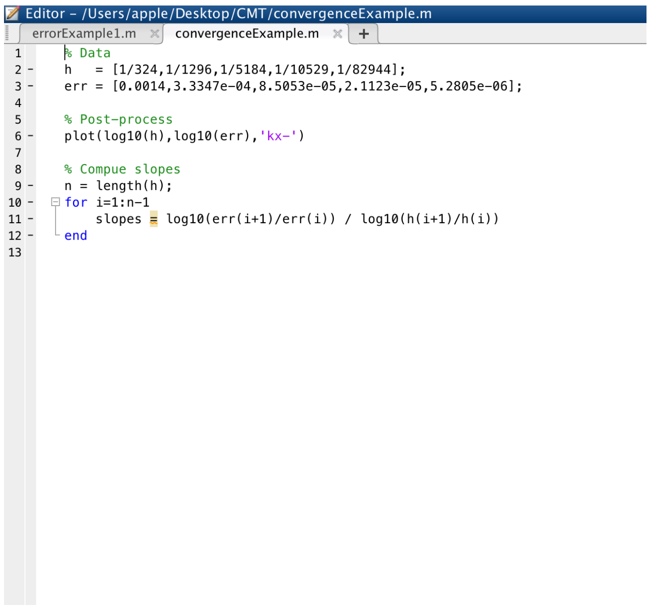
As said the geometry, equation and the boundary conditions are defined in PDEtool. We have 3 Newman and 1 Dirichelet (at y=0) boundary conditions.

As for the meshes, they were refined 4 times and the error was calculated for all 5 of the cases.

The Error and the convergence codes:



```
% Nodal coordinates (PDETOOL – Export mesh)
X = p(1,:);
Y = p(2,:);

% Conectividades
T = t(1:3,:);

% Numerical solution (PDETOOL – Export solution)
Sol = u(:,11);

% Analytical solution
for j=1:length(X)
    SolA(j) = (X(j)^2)+(X(j)*Y(j))-(Y(j)^2)+exp(-30);
end
% Norm of the maximum error
    errMax = max(abs(Sol(j)-SolA(j)))






% % Post-process
% figure(1)
%trisurf(T,X,Y,Sol)
%title('Numerical solution')
%
% figure(2)
%trisurf(T,X,Y,SolA)
%title('Analytical Solution')
%
% figure(3)
%trisurf(T,X,Y,abs(Sol-SolA))
%title('Error')
```



```
% Data
h   = [1/324,1/1296,1/5184,1/10529,1/82944];
err = [0.0014,3.3347e-04,8.5053e-05,2.1123e-05,5.2805e-06];

% Post-process
plot(log10(h),log10(err),'kx-')

% Compue slopes
n = length(h);
for i=1:n-1
    slopes = log10(err(i+1)/err(i)) / log10(h(i+1)/h(i))
end
```

Fig3. The codes for calculating the error and convergence

The code for calculating the maximum error starts by saving all the X and Y coordinates of the nodes form the p matrix that we have exported from the PDEtool. Afterwards it calculates the analytical solution for each of the nodes at t=10s. The maximum error is found from the difference of the exported PDEtool solution and the analytical one.

The maximum error for each of the meshes is saved into a matrix called err in the convergence code and the mesh size is calculated using the area of our domain (in this case 1x1) and the number of elements which is extracted from the connectivity matrix. The plot of the error and the mesh size for 4 time mesh refining is shown below:

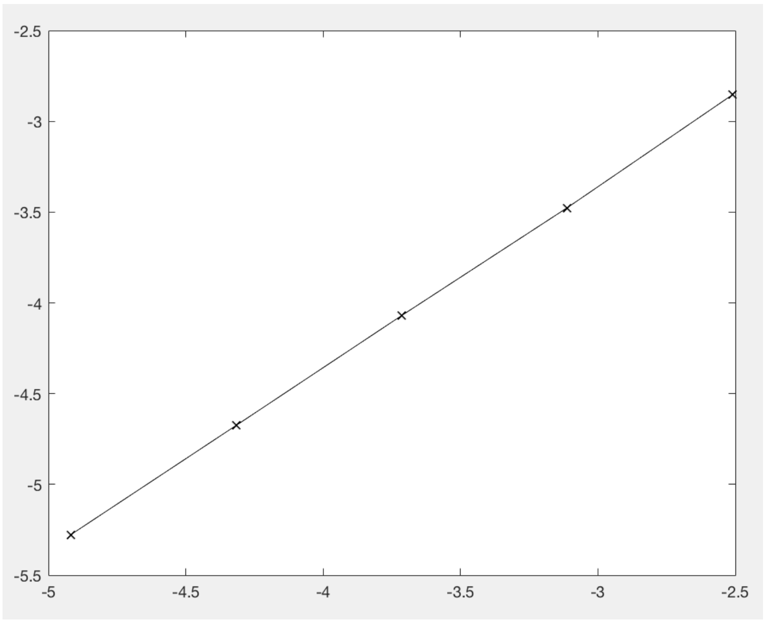


Fig4. Plot for the error and average mesh size for t=10s

For the second question if we plot the average of the solution for every second we can see that after about 4 seconds the plot turns into a line this means that the variation in our solutions doesn't change noticeably by time. this means that the problem has become steady state. So as we modify the final time if its after the 4th second we won't see much difference in our solutions.

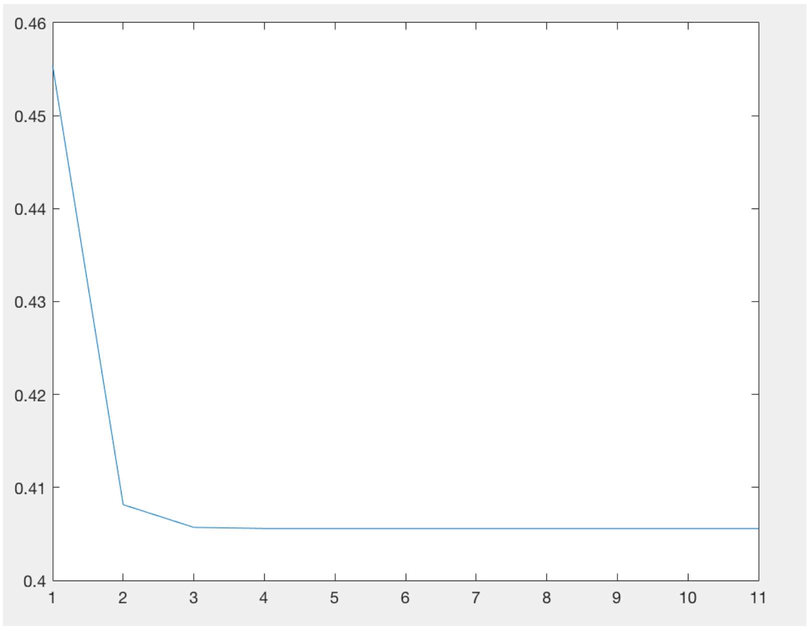


Fig5. The plot for the average of the solution by time

For the third case discussed, because e^-50 is very small we can consider it to be zero. So every term in our equation, boundary condition and initial condition that has e^-t we can neglect so basically the f term of our equation will be zero and the time dependent term in the boundary condition will disappear. From the second question we know that after about 4 seconds our solutions don't vary with time and the system becomes steady state so after 50 seconds we can consider the system as steady state and neglect the time dependable variable in the equation and change it form parabolic to elliptic so the equation will become much simpler to solve and the computational cost will be less. When we run the PDEtool for both the elliptic and parabolic cases, we see that for the elliptic case the CPU time is 271.4900 and for the parabolic case is 281.9900 so we benefit from the simplification.