RESEARCH ARTICLE OPEN ACCESS

Solving Partial Differential Equations with Matlab

M. Sundari[1], R.Vaithiyalingam[2]
[1](M.phil, Research scholar ,Department of Maths,Prist University,Puducherry ,India.)
[2](Asst. Professor in Maths,Department of Maths,Prist University,Puducherry,India.)

Abstract:

Numerical methods are commonly used for solving mathematical problems that are formulated in science and engineering where it is di?cult or even impossible to obtain exact solutions. Only a limited number of di?erential equations can be solved analytically. Numerical methods, on the other hand, can give an approximate solution to (almost) any equation. An ordinary dif-ferential equation (ODE) is an equation that contains an independent vari-able, a dependent variable, and derivatives of the dependent variable. Literal implementation of this procedure results in Euler's method, which is, how-ever, not recommended for any practical use. There are other methods more sophisticated than Euler's. Among them, there are four types of practical numerical methods for solving problems of ODEs. They are (i) Runge-Kutta Methods, (ii) Heun's Method, (iii) RK4 Method, and (iv)RKF5 Method. MATLAB is an integrated technical computing environment that com-bines numeric computation, advanced graphics and visualization and a high-level programming language. MATLAB was founded by Jack Little Cleve Moler and Steve Bangert in 1984. Color graphics makes it possible to display curves, surfaces, and solids in two and three dimensions in a way that it is both more e?ective and more engaging. The easy syntax of MATLAB makes it possible for users to get comfortable results quickly. MATLAB graphics are excellent and easy to use. It is easy to program in MATLAB, making it possible to do numerical calculations using simple steps. We can also use the symbolic capability of MATLAB to get analytical solutions.

MATLAB is becoming the most popular software package in Mathematics.

- In first chapter, we study the preliminaries needed to use MATLAB to solve problems in ODEs. We see basic definitions based on Di?erential Equation and Partial Di?erential Equation. We introduce MATLAB plotting to simulate the solution of Di?erential Equation problems.

- In second chapter, we review the software package MATLAB for high-performance numerical computation and visualization which provides an interactive environment with hundreds of built-in-functions for tech-nical computation, graphics and animation.

- In third chapter, we discuss the various Numerical Methods available for solving DEs and they are (1) Euler's method, (2) Heun's method,(3) RK4 method and (4) RKF5 method.

- In fourth chapter, we study the MATLAB programs for Solving Dif-ferential Equations. Also we study Euler methods and Runge-Kutta methods in MATLAB.

- In fifth chapter, we compare various methods and their approximation for 1st order di?erential equation and a system of ODEs. We also test the problem with the above methods.

In sixth chapter, we attempt to simulate the solution of few partial di?erential equations using MATLAB.

I. INTRODUCTION

Solving Partial Differential Equations with MATLAB

We solve the linear advection equation for u(x, t),

$$\frac{\partial u}{\partial t} = c\,\frac{\partial u}{\partial x}$$

defined on $(-\infty, \infty)$, $t \in [0, \infty)$ and with the boundary condition

u(x, 0) = f (x).

Define and initialize the variables for =1:10
end

The boundary condition (which essentially defines the "initial state")

uij(4,1) = 1;

integration to t = 0.1 and 0.2 for j = 1:2

for i = 2:9

uij(i,j+1) = -0.25*uij(i-1,j)+uij(i,j)+0.25*uij(i+1,j);
end

end

making the plots for t = 0, 0.1, and 0.2

for i = 1:10

u0(i) = uij(i,1);

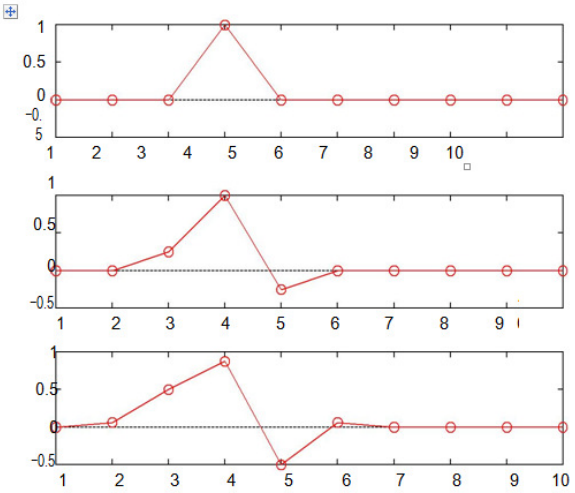
u1(i) = uij(i,2);

u2(i) = uij(i,3);

0; end

```
>>subplot(3,1,1);
>>plot(xi,z,'k--',xi,u0,'r-o');
>>axis([1 10 -0.5 1])
>>subplot(3,1,2);
>>plot(xi,z,'k--',xi,u1,'r-o');
>>axis([1 10 -0.5 1])
>>subplot(3,1,3);
>>plot(xi,z,'k--',xi,u2,'r-o');
>>axis([1 10 -0.5 1])
```

Figure 6.1: Graph for the linear advection equation



Poisson's Equation on a Unit Disc

(1) **Poisson's Equation on a Unit Disk**

The particular PDE is

$$-\triangle u = 1,$$

on the unit disk with zero-Dirichlet boundary conditions. The exact solution is

$$u(x, y) = \frac{1 - x^2 - y^2}{4}$$

Problem Definition:

The following variables will define our problem

- g: A specification function that is used by initmesh.
- b: A boundary file used by assempde.

- c,a,f: The coefficients and inhomogeneous term.

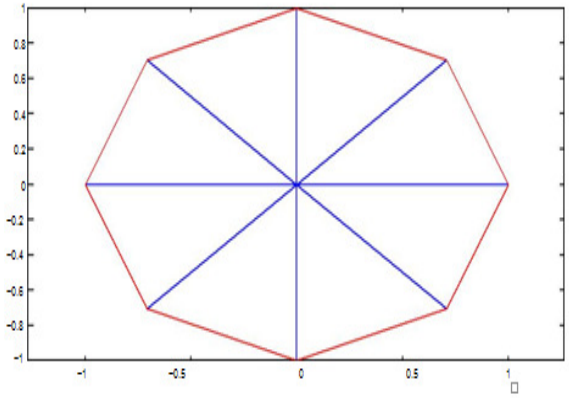
```
>>g='circleg';
>>b='circleb1';
>>c=1;
>>a=0;
>>f=1;
```

Generate Initial Mesh

The function initmesh takes a geometry specification function and returns a discretization of that domain. The 'hmax' option lets the user specify the maximum edge length. In this case, the domain is a unit disk, a maximum edge length of one creates a coarse discretization.

```
[p,e,t]=initmesh(g,'hmax',1);
figure;
pdemesh(p,e,t);
axis equal
```

Figure 6.2: Graph for Poisson equation



Refinement

We repeatedly refine the mesh until the infinity-norm of the error vector is less than a 10^{-3}.

```
er = Inf;
while er >
0.001
[p,e,t]=refineme
sh(g,p,e,t);
u=assempde(b,
p,e,t,c,a,f);
exact=(1-
p(1,:).^2-
p(2,:).^2)'/4;
er=norm(u-
exact,'inf');
fprintf('Error: %e. Number of
nodes: %d\n',er,size(p,2)); end
```

Error: 1.292265e-02. Number of nodes: 25

Error: 4.079923e-03. Number of nodes: 81

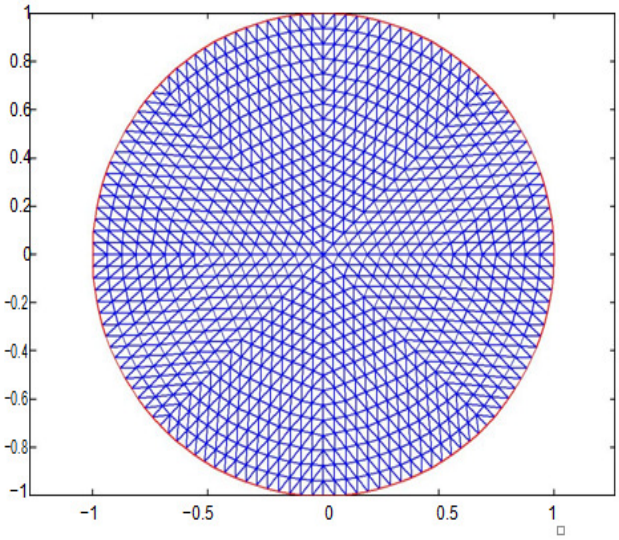
Error: 1.221020e-03. Number of nodes: 289

Error: 3.547924e-04. Number of nodes: 1089

Plot Final Mesh

```
>>figure;
>>pdemesh(p,e,t);
>>axis equal
```

Figure 6.3: Graph for Refinement

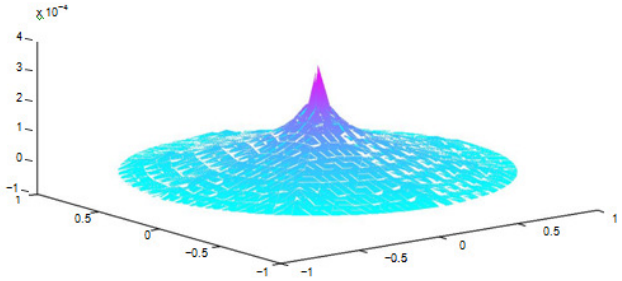


Plot Error

The scale of the vertical axis shows that the error is small and of the order 10^{-4}.

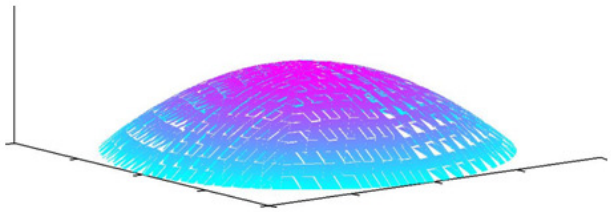
```
>>figure;
>>pdesurf(p,t,u-exact);
```

Figure 6.4: Graph for Exact



Plot FEM Solution

```
>>figure;
>>pdesurf(p,t,u);
```



Helmholtz's Equation on a Unit Disk with a Square Hole

(2) The Helmholtz equation, an elliptic equation, ie., the time-independent form of wave equation is

$$-\Delta u - k^2 u = 0$$

Problem Definition

The following variables will define our problem:

- g: A specification function that is used by initmesh.
- b: A boundary file used by assempde.
- c,a,f: The coefficients and inhomogeneous term.

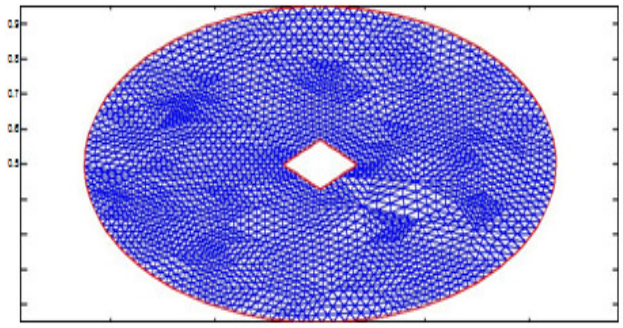
```
>>g='scatterg';
>>b='scatterb';
>>c=1;
>>k=60;
>>a=-k^2;
>>f=0;
```

Create Mesh

We need a fine mesh to resolve the waves. To achieve this, we refine the initial mesh twice.

```
>>[p,e,t]=initmesh(g);
>>[p,e,t]=refinemesh(g,p,e,t);
>>[p,e,t]=refinemesh(g,p,e,t);
>>pdemesh(p,e,t);
axis equal
```

Figure 6.6: Graph for Helmholtz equation



Solve for Complex Amplitude

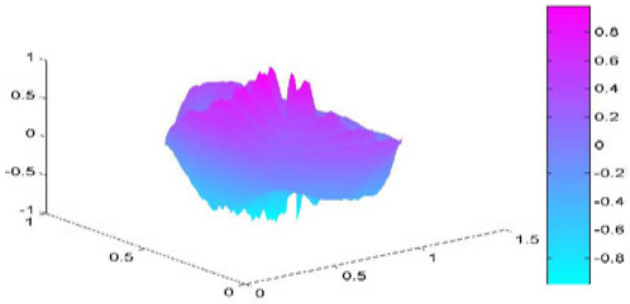
The real part of the vector u stores an approximation to a real-valued solution of the Helmholtz equation.

```
>>u=assempde(b,p,e,t,c,a,f);
```

Plot FEM Solution

```
>>figure;
>>h = newplot;
>>set(get(h,'Parent'),'Renderer','zbuffer')
>>pdeplot(p,e,t,'xydata',real(u),'zdata',real(u),'mesh','off');
>>colormap(cool)
```

Figure 6.7: Graph for Complex Amplitude

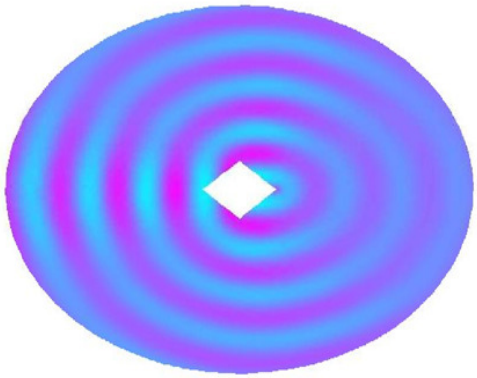


Animate Solution to Wave Equation

Using the solution to the Helmholtz equation, we construct an animation showing the corresponding solution to the time-dependent wave equation.

```
>>figure;
>>m=10;
>>h = newplot;
>>hf=get(h,'Parent');
>>set(hf,'Renderer','zbuffer')
>>axis tight,
set(gca,'DataAspectRatio',[1 1
1]); >>axis off
>>M=moviein(m,hf);
>>uu=real(exp(-j*2*pi/m*sqrt(-
1))*u); >>pdeplot(p,e,t,'xydata',uu,'colorbar',
'off','mesh','off'); >>caxis([-maxu maxu]);
>>axis tight,
set(gca,'DataAspectRatio',[1 1
1]); >>axis off;
>>M(:,j)=getframe(hf);
>>end
>>movie(hf,M,2);
```

Figure 6.8: Graph for Wave Equation



Conclusion:

In this paper we demonstrated that the different types of differential equa- tion could be solved with MATLAB.

- Ordinary differential equation and partial differential equation could be easily solved by using MATLAB coding and get the figure for any complicated equations.
- By comparing Four numerical types of methods, we arrived at the the-oretical result that Eulers method is 1st order, Heuns method is 2nd order, RK4 method is 4th order and RKF5 is 5th order method.

We attempted to simulate the solution of few partial differential equa-tions

REFERENCES

[1] Curtis F. Gerald, Patrick O. Wheatley, "Applied Numerical Analysis", Addision-Wesley, Fifth Edition

[2] Wen Shen, "Journal on Introduction to Numerical computation", xuru.org, 2012.

[3] P.Howard, "Solving Ordinary Equations in MATLAB", Fall, 2009.

[4] David Houcque, "Applications of MATLAB on Ordinary Differential Equations", Robert R. McCormick School of Engineering and Applied Science, Northwestern University.

[5] Rudra Pratap, "Getting Started with MATLAB", Department of Me-chanical Engineering, Indian Institute of Science, Bangalore, 2010.

[6] Hari Kishan, "Differential Equations", Atlantiv Publishers And Distrib-utors, 2006.