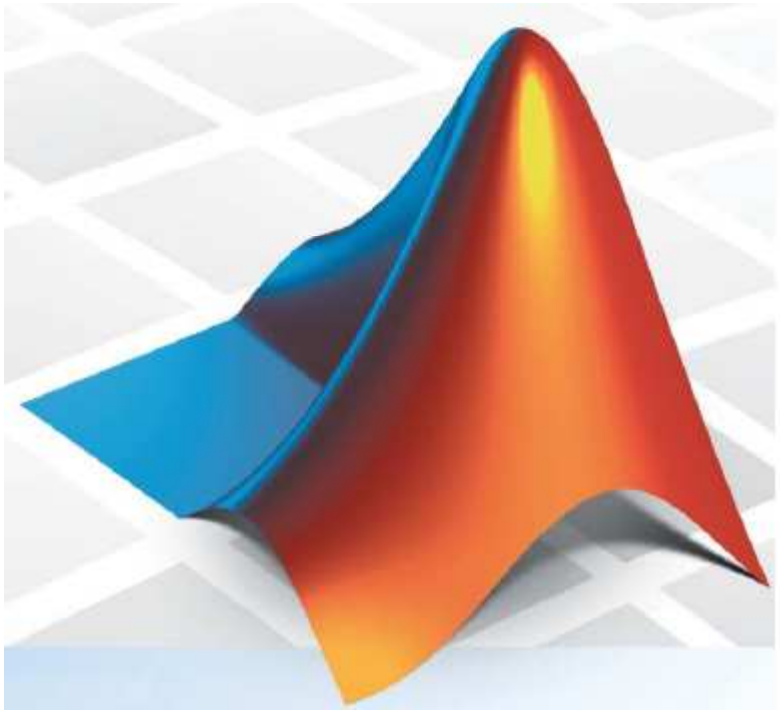
Solve ode, dde & pde using matlab

Farida Mosally

Mathematics Department

King Abdulaziz University



2014

Outline

1. Ordinary Differential Equations (ode)
   - 1.1 Analytic Solutions
   - 1.2 Numerical Solutions

2. Delay Differential Equations (dde)

3. Partial Differential Equations (pde)

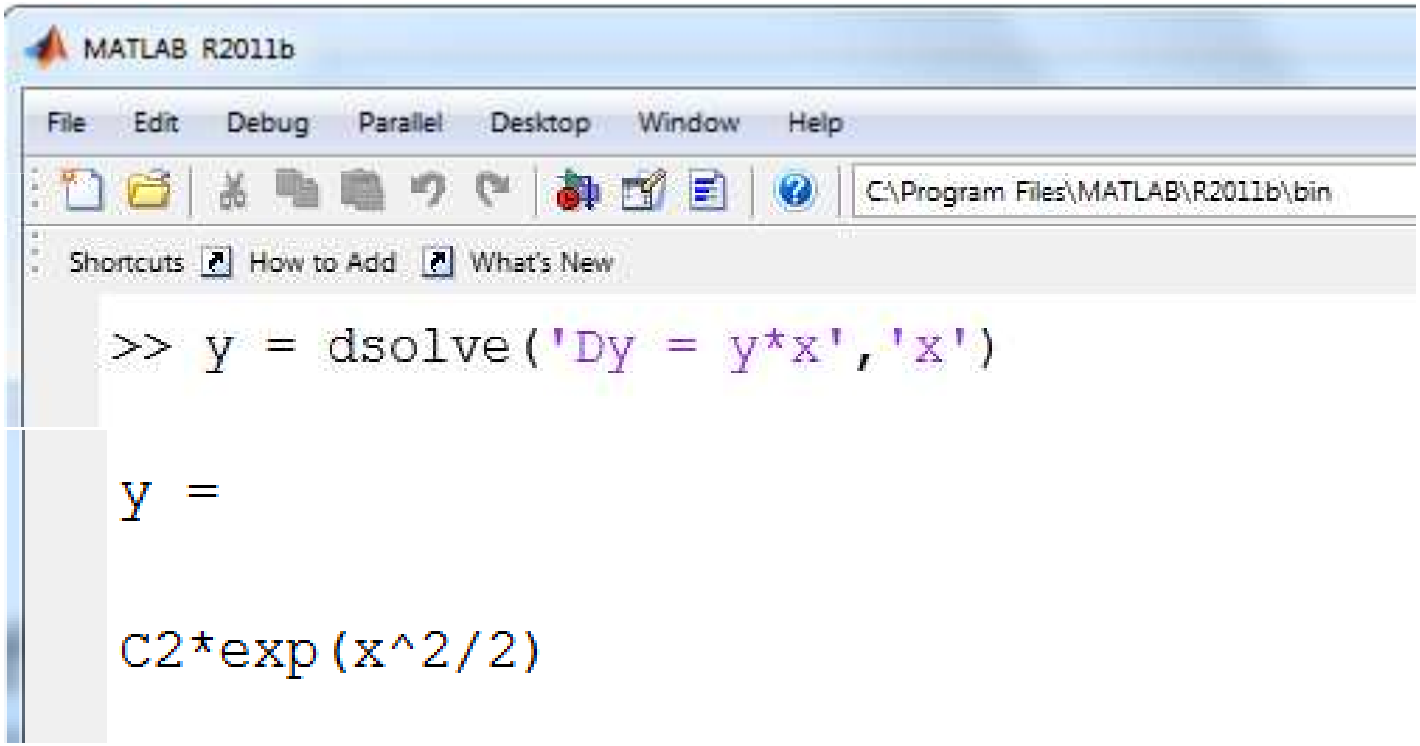
Ordinary Differential Equations

Analytic Solutions

First Order Differential Equation

To solve a first order differential equation, say for example

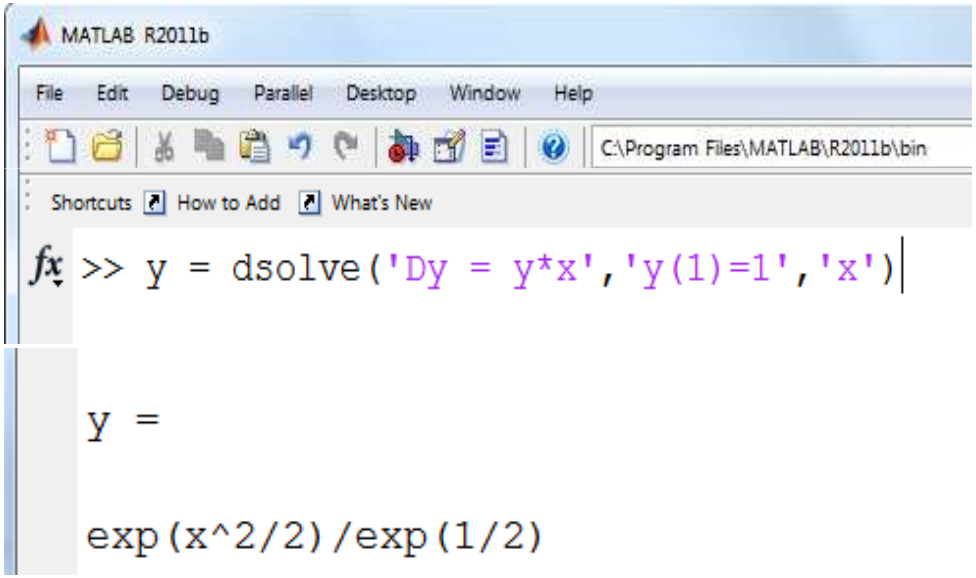
$$y' = xy$$



Initial Value Problem

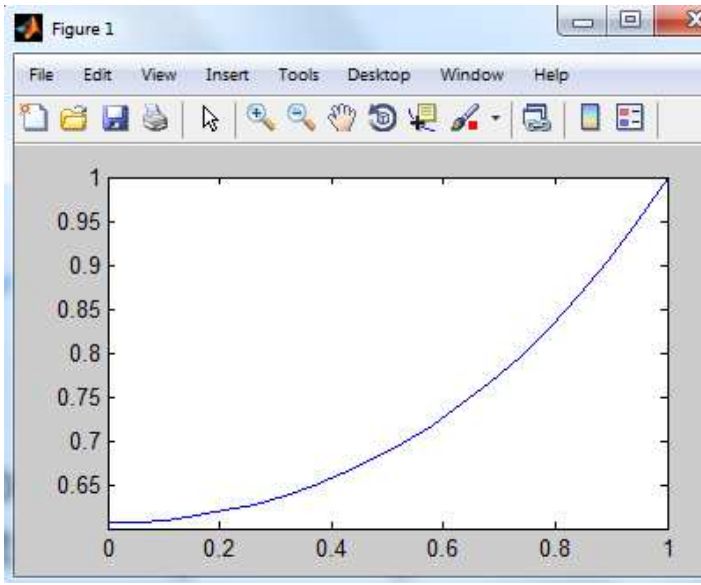
To solve an initial value problem, say,

$$y' = xy, \quad y(1) = 1$$



Suppose we want to plot the solution to get a rough idea of its behavior.

```
>> x = linspace(0,1,20);
>> z = eval(vectorize(y));
>> plot(x,z)
```



Second and Higher Order Equations

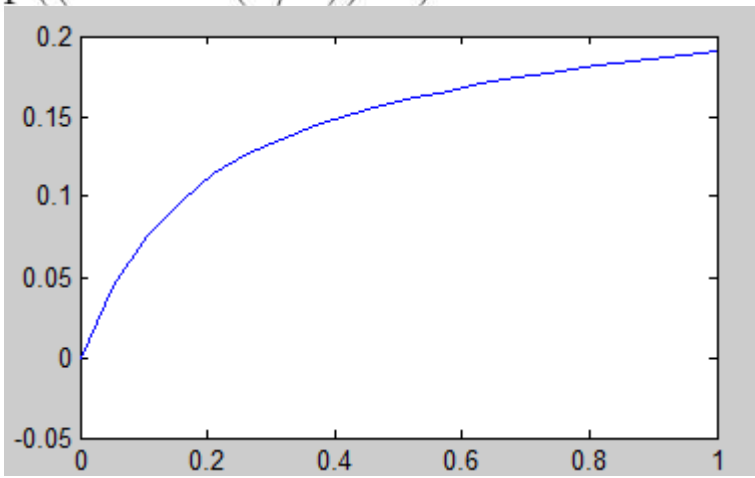
Suppose we want to solve and plot the solution to the second order equation

$$y''(x) + 8y'(x) + 2y(x) = \cos(x); \qquad y(0) = 0,\ y'(0) = 1.$$

```
>> syms x, eqn = 'D2y + 8*Dy + 2*y = cos(x)';
    ic = 'y(0)=0, Dy(0)=1';
     y = dsolve(eqn,ic,'x')
```

```
y =
1/65*cos(x)+8/65*sin(x)+(-1/130+53/1820*14^(1/2))*exp((-4+14^(1/2))*x)
-1/1820*(53+14^(1/2))*14^(1/2)*exp(-(4+14^(1/2))*x)
```

```
>> x = linspace(0,1,20);
>> z = eval(vectorize(y));
>> plot(x,z)
```



Ordinary Differential Equations

Numerical Solutions

MATLAB has a number of tools for numerically solving ordinary differential equations. In the following table we display some of them.

ode initial value problem solvers

Solver	Method used	Order of Accuracy	When to Use
ode45	Runge-Kutta (4,5) formula	Medium	Most of the time. This should be th first solver you try.
ode23	Runge-Kutta (2, 3) formula	Low	For problems with crude error tolerances or for solving moderate stiff problems.
ode113	Adams-Bashforth-Moulton solver	Low to high	For problems with stringent error tolerances or for solving computationally intensive problem
ode15s	Solver based on the numerical differentiation formulas	Low to medium	If ode45 is slow because the problem is stiff.
ode23s	Solver based on a modified Rosenbrock formula of order 2	Low	If using crude error tolerances to solve stiff systems and the mass matrix is constant.

Defining an ODE function

```
[outputs] = function_handle(inputs)
[t,state] = solver(@dstate,tspan,ICs,options)
```

An array. The solution of the ODE (the values of the state at every time).

Matlab algorithm (e.g., `ode45`, `ode23`)

Handle for function containing the derivatives

Vector that specifiecs the interval of the solution (e.g., `[t0:5:tf]`)

A vector of the initial conditions for the system (row or column)

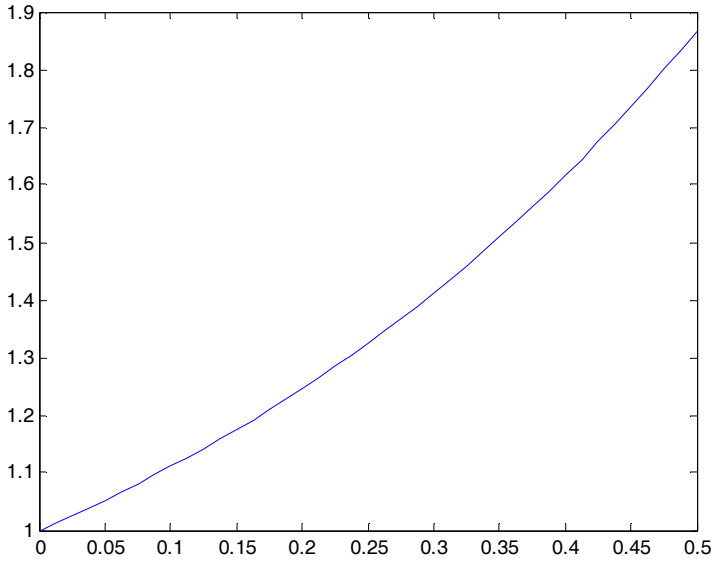
Defining an ODE function

```
sol = solver(odefun,[t0 tf],y0,options)
```

Example 1

$y' = ty + y, \qquad y(0) = 1, \qquad 0 \le t \le 0.5.$

```
>> f=inline('t*y+y')
  f =
      Inline function:
       f(t,y) = t*y+y

>> [t,y]=ode45(f,[0 0.5],1);
>> plot(t,y)
```



First Order Equations with M-files

Example 2

$$dy/dx = xy^2 + y, \qquad y(0) = 1, \qquad x\epsilon[0,0.5].$$

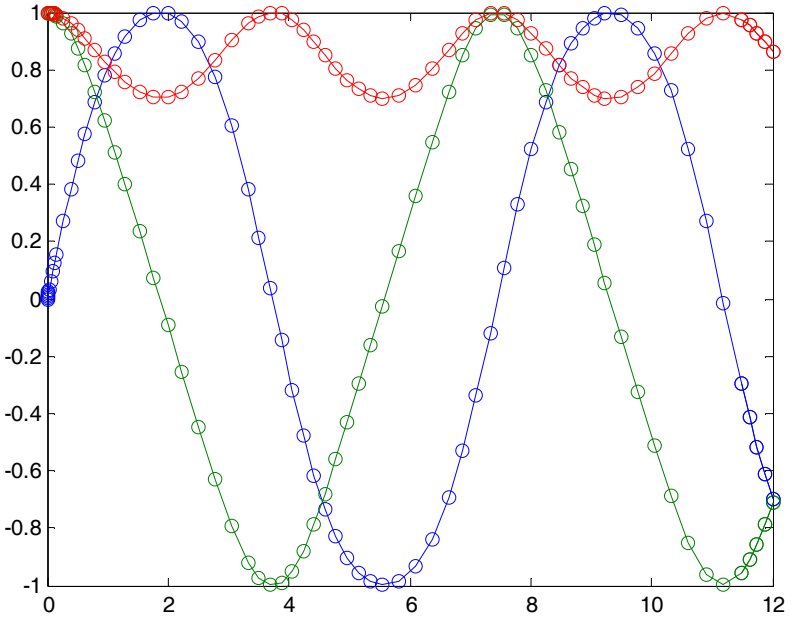
```
function out = example2()
xspan = [0,.5];
y0 = 1;
[x,y]=ode23(@firstode,xspan,y0);
plot(x,y)

function yprime = firstode(x,y);
yprime = x*y^2 + y;
```

Solving systems of first-order ODEs

Example 3

$$
\begin{aligned}
y_1' &= y_2 y_3, & y_1(0) &= 0 \\
y_2' &= -y_1 y_3, & y_2(0) &= 1 \\
y_3' &= -0.51 y_1 y_2, & y_3(0) &= 1
\end{aligned}
$$

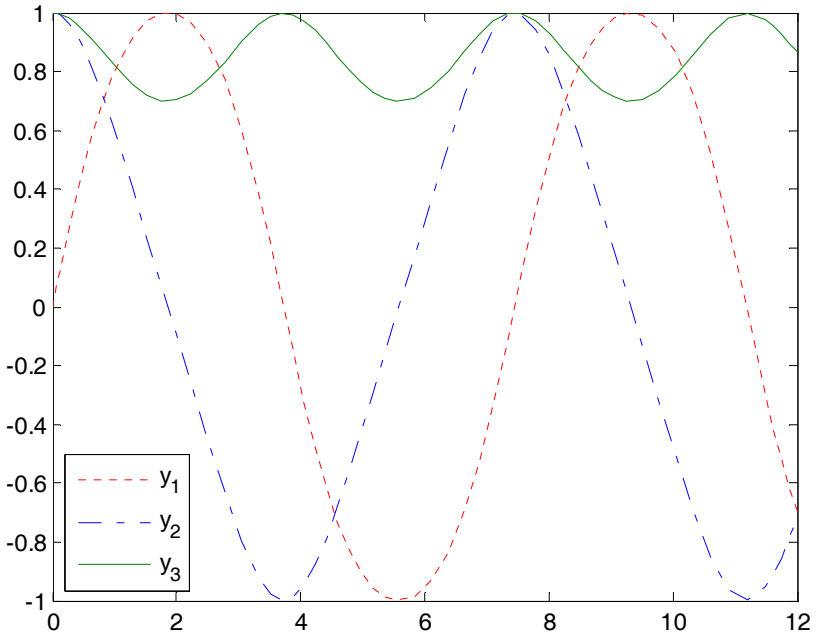
```
function example3
tspan = [0 12];
y0 = [0; 1; 1];

% Solve the problem using ode45
 ode45(@f,tspan,y0);

% -------------------------------
function dydt = f(t,y)
dydt = [ y(2)*y(3)
        -y(1)*y(3)
        -0.51*y(1)*y(2) ];
```



… Solving systems of first-order ODEs

To define each curve

```
function example3
tspan = [0 12];
y0 = [0; 1; 1];

% Solve the problem using ode45
[t,y] = ode45(@f,tspan,y0);

plot(t,y(:,1),':r',t,y(:,2),'-.',t,y(:,3))
legend('y_1','y_2','y_3',3)
% ------------------------------
function dydt = f(t,y)
dydt = [ y(2)*y(3)
        -y(1)*y(3)
        -0.51*y(1)*y(2) ];
```



Boundary Value Problems

```
sol = bvp4c(odefun,bcfun,solinit)
```

Delay Differential Equations

Functions

dde23	**Solve delay differential equations (DDEs) with constant delays**
ddesd	Solve delay differential equations (DDEs) with general delays
ddensd	Solve delay differential equations (DDEs) of neutral type

dde23

Solve delay differential equations (DDEs) with constant delays

```
sol = dde23(ddefun,lags,history,tspan)
```

Example 4

$$y_1'(t) = y_1(t-1)$$
$$y_2'(t) = y_1(t-1) + y_2(t-0.2)$$
$$y_3'(t) = y_2(t).$$

$$y_1(t) = 1$$
$$y_2(t) = 1$$
$$y_3(t) = 1$$

for $t \le 0$.

```
function ddex1
sol = dde23(@ddex1de,[1, 0.2],@ddex1hist,[0, 5]);
figure;
plot(sol.x,sol.y)
title('An example of Wille'' and Baker.');
xlabel('time t');
ylabel('solution y');


function s = ddex1hist(t)
% Constant history function for DDEX1.
s = ones(3,1);

function dydt = ddex1de(t,y,Z)
% Differential equations function for DDEX1.
ylag1 = Z(:,1);
ylag2 = Z(:,2);
dydt = [ ylag1(1)
         ylag1(1) + ylag2(2)
         y(2)                 ];
```

$$y_1(t) = 1$$

$$y_2(t) = 1$$

$$y_3(t) = 1$$

$$y_1'(t) = y_1(t-1)$$

$$y_2'(t) = y_1(t-1) + y_2(t-0.2)$$

$$y_3'(t) = y_2(t).$$

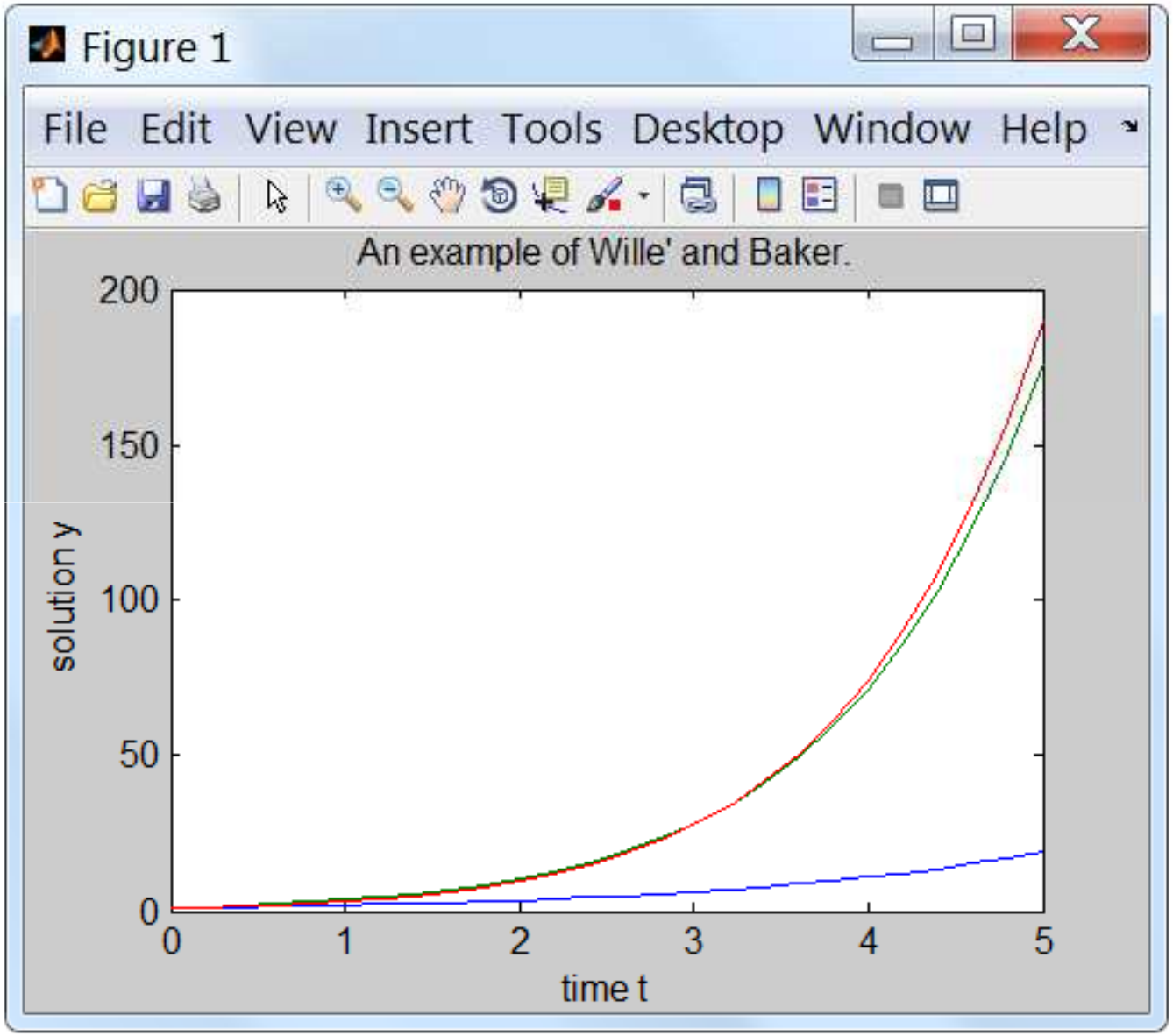


Figure 1
File Edit View Insert Tools Desktop Window Help
An example of Wille' and Baker.
200
150
100
50
0
solution y
0
1
2
3
4
5
time t

Partial Differential Equations (pde)

pdepe

Solve initial-boundary value problems for parabolic-elliptic PDEs in 1-D

Syntax

```
sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)

sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,OPTIONS)

[sol,tsol,sole,te,ie] = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan,OPTIONS)
```

Arguments

```
sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)
```

m	**A parameter corresponding to the symmetry of the problem. m can be slab = 0, cylindrical =1, or spherical = 2.**
pdefun	A handle to a function that defines the components of the PDE.
icfun	A handle to a function that defines the initial conditions.
bcfun	A handle to a function that defines the boundary conditions.
xmesh	A vector [x0, x1, ..., xn], x0 < x1 < ... < xn.
tspan	A vector [t0, t1, ..., tf] , t0 < t1 < ... < tf.
options	Some OPTIONS of the underlying ODE solver are available, See odeset for details.

Description

PDE $$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left(x^{m}f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+\mathrm{s}\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

IC $$u(x,t_0)=u_0(x)$$

BC $$p(x,t,u)+q(x,t)f\left(x,t,u,\frac{\partial u}{\partial x}\right)=0$$

PDE

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left(x^{m}f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+\mathrm{s}\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

$$\pi^{2}\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)$$

IC

$$u(x,t_0)=u_0(x)$$

BC

$$p(x,t,u)+q(x,t)f\left(x,t,u,\frac{\partial u}{\partial x}\right)=0$$

Single PDE

xample 1.

his example illustrates the straightforward formulation, computation, and plotting of the olution of a single PDE.

$$\pi^2 \frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)$$

$$m = 0$$
$$c = \pi^2$$
$$f = \frac{\partial u}{\partial x}$$
$$S = 0$$

This equation holds on an interval $0 \le x \le 1$ for times $t \ge 0$.
The PDE satisfies the initial condition

$$u(x,0) = \sin \pi x \qquad \text{IC } u_0$$

and boundary conditions

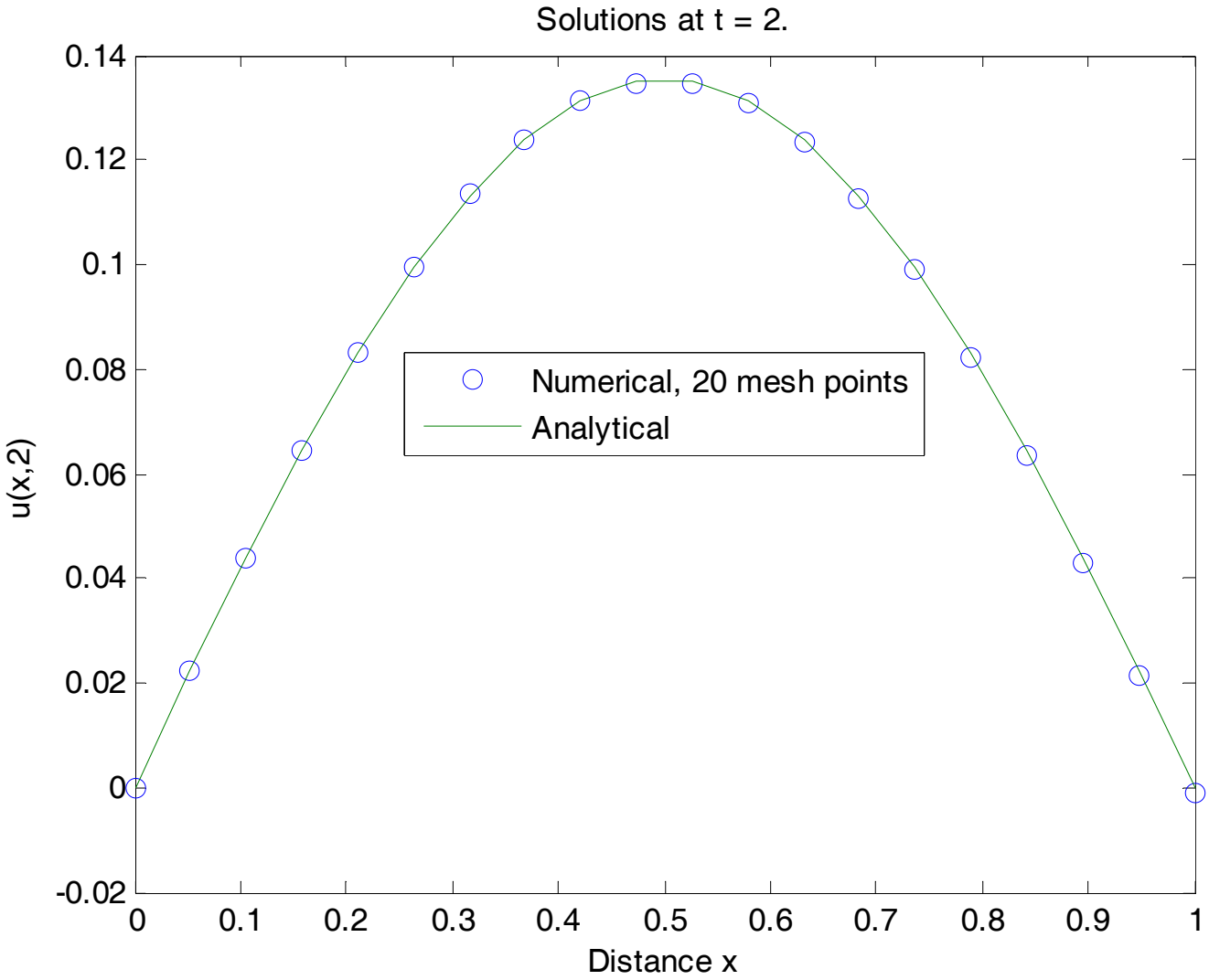
$$u(0,t) = 0$$
$$\pi e^{-t} + \frac{\partial u}{\partial x}(1,t) = 0$$

BC $\quad pl = ul, \quad ql = 0$
$\quad pr = \pi e^{-t}, \quad qr = 1$

It is convenient to use subfunctions to place all the functions required by pdepe in a single M-file.

```
nction pdex1

= 0;
= linspace(0,1,20);
= linspace(0,2,5);

l = pdepe(m,@pdecfs,@ic,@bc,x,t);

= sol(:,:,1);


 solution profile can also be illuminating.
ure;
t(x,u(end,:),'o',x,exp(-t(end))*sin(pi*x));
le('Solutions at t = 2.');
end('Numerical, 20 mesh points','Analytical',0);
bel('Distance x');
bel('u(x,2)');
```

```
function [c,f,s] = pdecfs(x,t,u,DuDx)
c = pi^2;
f = DuDx;
s = 0;

% --------------------------------

function u0 = ic(x)
u0 = sin(pi*x);

% --------------------------------
function [pl,ql,pr,qr] = bc(xl,ul,xr,ur
pl = ul;
ql = 0;
pr = pi * exp(-t);
qr = 1;
```


Solutions at t = 2.
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
u(x,2)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Distance x
Numerical, 20 mesh points
Analytical

References

1. http://www.mathworks.com/help/matlab/ref/pdepe.html

THANK YOU

اتمنى ان تكوني إستفدتي