Ordinary Differential Equation (ODE) Solvers

ODE Solvers

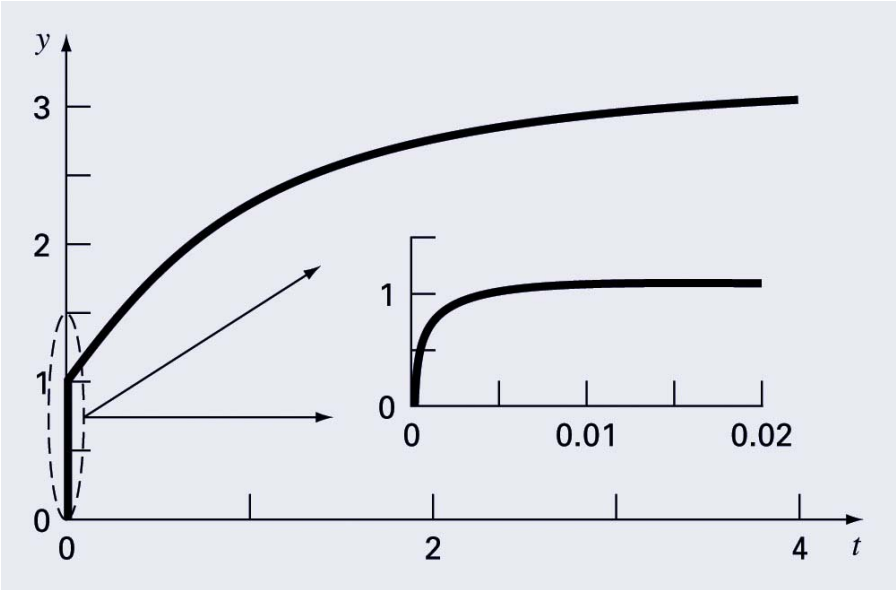
- This table lists the ***initial value problem solvers***, the kind of problem you can solve, and the method each solver uses

Solver	Solves These Kinds of Problems	Method
ode45	Nonstiff differential equations	Runge-Kutta
ode23		
ode113		Adams
ode15s	Stiff differential equations and DAEs	NDFs (BDFs)
ode23t	Moderately stiff differential equations and DAEs	Trapezoidal rule
ode23s	Stiff differential equations	Rosenbrock
ode23tb		TR-BDF2
ode15i	Fully implicit differential equations	BDFs

Stiffness

- A stiff system is one involving rapidly changing components together with slowly changing ones.
- An example of a single stiff ODE is: $\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}$

  whose solution if y(0)=0 is: $y = 3 - 0.998e^{-1000t} - 2.002e^{-t}$



First Order ODEs

- An ordinary differential equation (ODE) contains one or more derivatives of a dependent variable y with respect to a single independent variable t, usually referred to as time.
- The derivative of y with respect to t is denoted as y ′, the second derivative as y ′′, and so on.

- MATLAB solvers handle the following types of first-order ODEs:
  - Explicit ODEs of the form $y' = f(t, y)$
  - Linearly implicit ODEs of the form $M(t, y)\, y' = f(t, y)$, where $M(t, y)$ is a matrix
  - Fully implicit ODEs of the form $f(t, y, y') = 0$ **(ode15i only)**

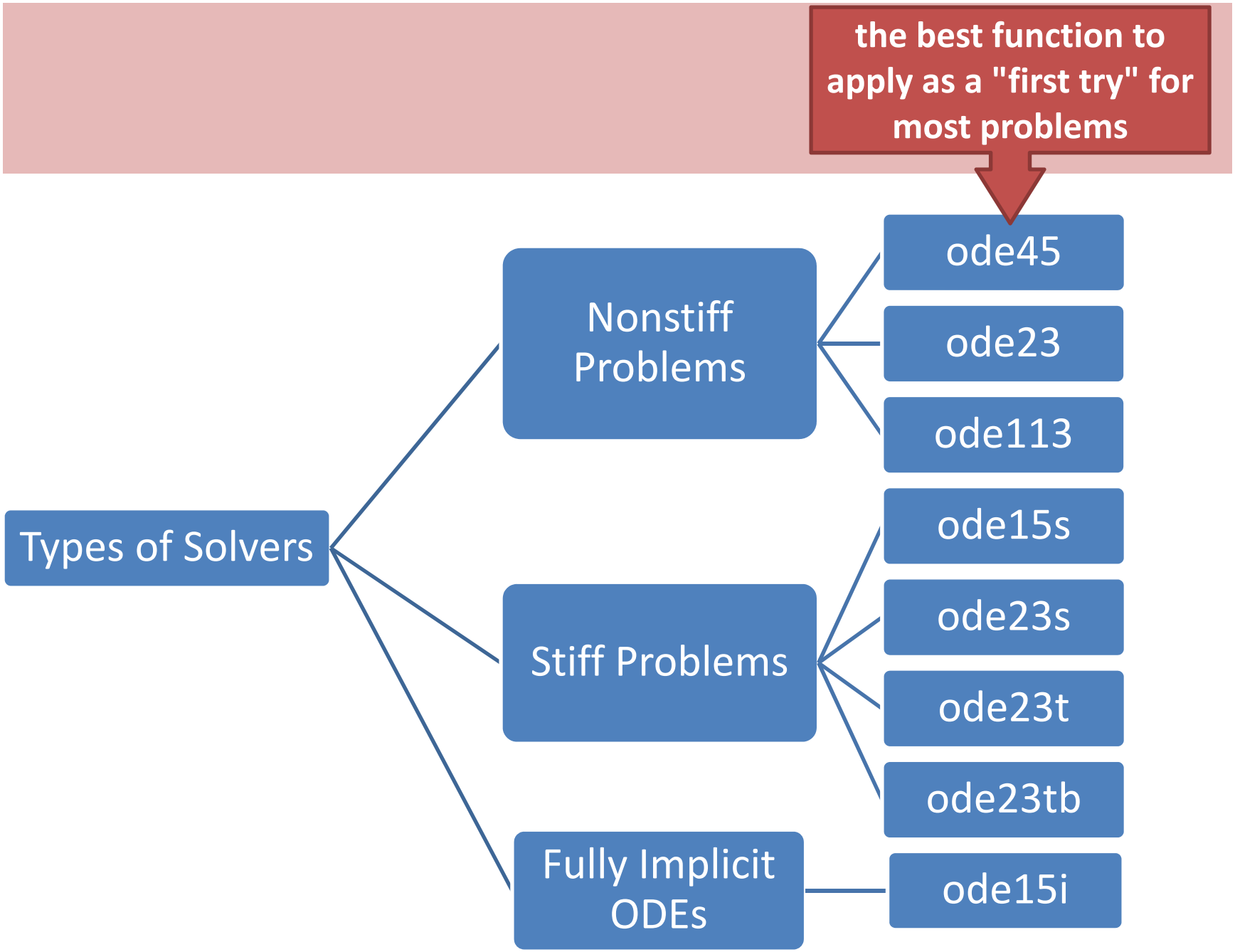
Initial Values

- Generally there are many functions y(t) that satisfy a given ODE, and additional information is necessary to specify the solution of interest.
- In an initial value problem, the solution of interest satisfies a specific initial condition, that is, y is equal to y_0 at a given initial time t_0.

- An initial value problem for an ODE is then

$$y' = f(t, y)$$

$$y(t_0) = y_0$$


the best function to apply as a "first try" for most problems
Types of Solvers
Nonstiff Problems
ode45
ode23
ode113
Stiff Problems
ode15s
ode23s
ode23t
ode23tb
Fully Implicit ODEs
ode15i

Solver Syntax

- All of the ODE solver functions, except for ode15i, share the same syntax:

```
[t,y] = solver(odefun,tspan,y0,options)
```

odefun	**Handle to a function that evaluates the system of ODEs. The function has the form dydt = odefun(t,y) where t is a scalar, and dydt and y are column vectors.**
tspan	**Vector specifying the interval of integration. The solver imposes the initial conditions at tspan(1), and integrates from tspan(1) to tspan(end).**
y0	**Vector of initial conditions for the problem.**
options	**Structure of optional parameters that change the default integration properties. Integrator Options tells you how to create the structure and describes the properties you can specify.**

Example
Van der Pol Equation (Nonstiff)

- Solve the following second order differential equation using ode45:

$$y_1'' - y_1' + y_1^2\, y_1' + y_1 = 0$$

Remember, MATLAB can one deal with first order differential equations

➔We need to rewrite the problem as a system of first-order ODEs.

$$y_1' = y_2$$
$$y_2' = y_2 - y_2 y_1^2 - y_1$$

Example
Van der Pol Equation (Nonstiff)

- Now, code the system of first-order ODEs

```
function dydt = vdp1(t,y)
dydt=zeros(2,1)
dydt(1) = y(2);
dydt(2) = y(2)-y(1)^2*y(2)-y(1);
```

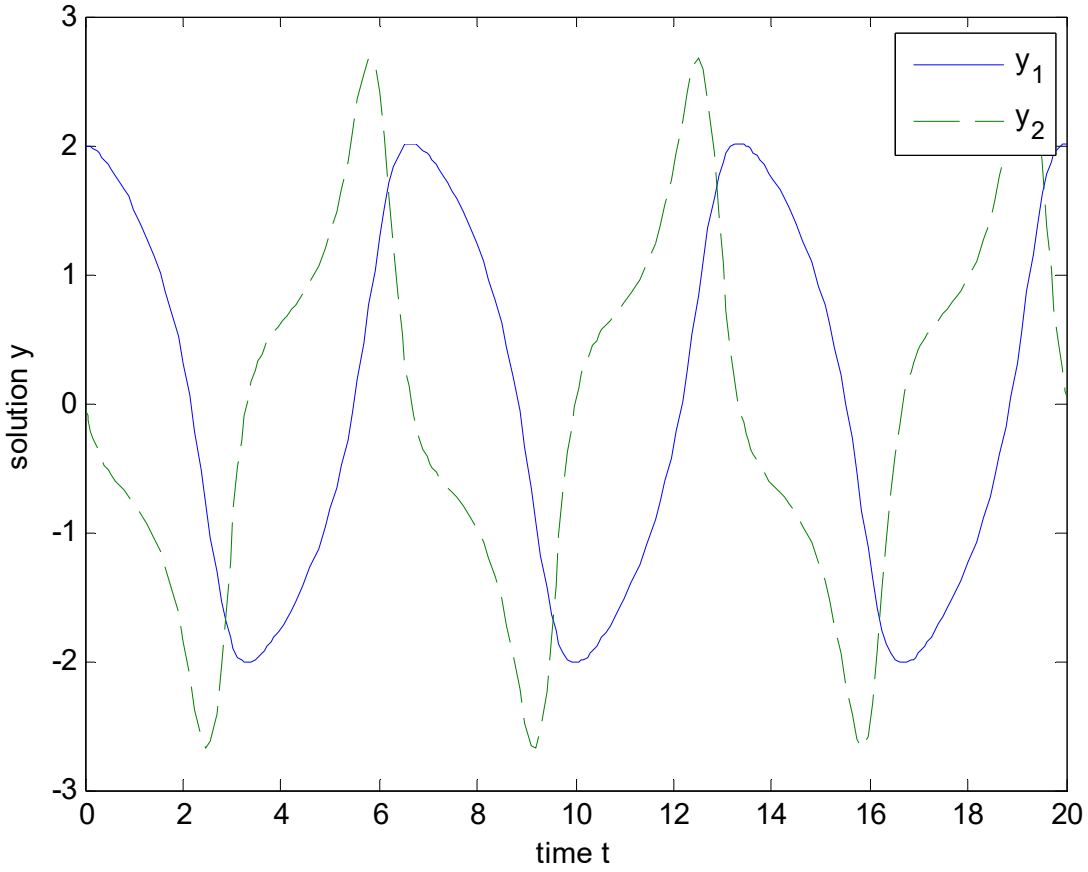
- Apply a solver to the problem

```
>>   [t,y] = ode45(@vdp1,[0,20],[2 0]);
```

- View the solver output

```
>>   plot(t,y(:,1),'-',t,y(:,2),'--'),
     xlabel('t'), ylabel('solution y'),
     legend('y_1','y_2')
```

Example
Van der Pol Equation (Nonstiff)



Example
Van der Pol Equation (Stiff)

- Solve the following second order differential equation using ode15s:

$$y_1'' - 1000y_1'(1 + y_1^2) + y_1 = 0$$

Remember, MATLAB can one deal with first order differential equations

➔We need to rewrite the problem as a system of first-order ODEs.

$$y_1' = y_2$$
$$y_2' = 1000y_2(1 - y_1^2) - y_1$$

Example
Van der Pol Equation (Stiff)

- Now, code the system of first-order ODEs

```
function dydt = vdp1000(t,y)
dydt=zeros(2,1)
dydt(1) = y(2);
dydt(2) = 1000*y(2)*(1-y(1)^2)-y(1);
```

- Apply a solver to the problem

```
>>[t,y] = ode15s(@vdp1000,[0,3000],[2 0]);
```

- View the solver output, only for y_1

```
>>   plot(t,y(:,1),'-'), xlabel('t'),
     ylabel('solution y')
```

Evaluating the Solution (van der Pol Equation)

- You can evaluate the approximate solution, S(x), at any point in the interval of integration (**`tspan`**) using the function **`deval`** and the structure **`sol`** returned by the solver.

```
>> sol = ode45(@vdp1,[0 20],[2; 0]);
>> xint = 1:5;
>> Sxint = deval(sol,xint)
Sxint =

 1.5081    0.3235   -1.8686   -1.7407   -0.8344
-0.7803   -1.8320   -1.0220    0.6260    1.3095
```

Fully Implicit ODE

- The solver ode15i solves fully implicit differential equations of the form

$$f(\mathrm{t}, y, y') = 0$$

- The basic syntax for ode15i is

```
[t,y] = ode15i(odefun,tspan,y0,yp0,options)
```

`yp0`	Vector of initial conditions for $y'(t_0)$

Fully Implicit ODE – Example

- Solve the following second order differential equation using ode15i:

$$\mathrm{t}y^2\,(y')^3 - y^3\,(y')^2 + t\left(1+t^2\right)y' - t^2\,y = 0$$

$$\text{with } y(1) = \sqrt{3/2}$$

- First, code your function

```
function res = imp(t,y,yp)
res=t*y^2*yp^3-y^3*yp^2+t*(t^2 + 1)*yp-t^2*y;
```

- But we need **`yp0`** and we don't have it

decic funtion

- Computes consistent initial conditions for ode15i.
- Use the function **decic** as a helper to compute a consistent initial value for **yp0**
- An initial guess is needed → let's make it "0"

```
>> t0 = 1;
>> y0 = sqrt(3/2);
>> yp0 = 0;
>>[y0,yp0]=decic(@imp,t0,y0,1,yp0,0);
```

- This will make **yp0=0.8165** (check it yourself)

Fully Implicit ODE – Example

- Now we can use ode15i

```
>>[t,y] = ode15i(@imp,[1 10],y0,yp0);
```

- And compare with the true solution

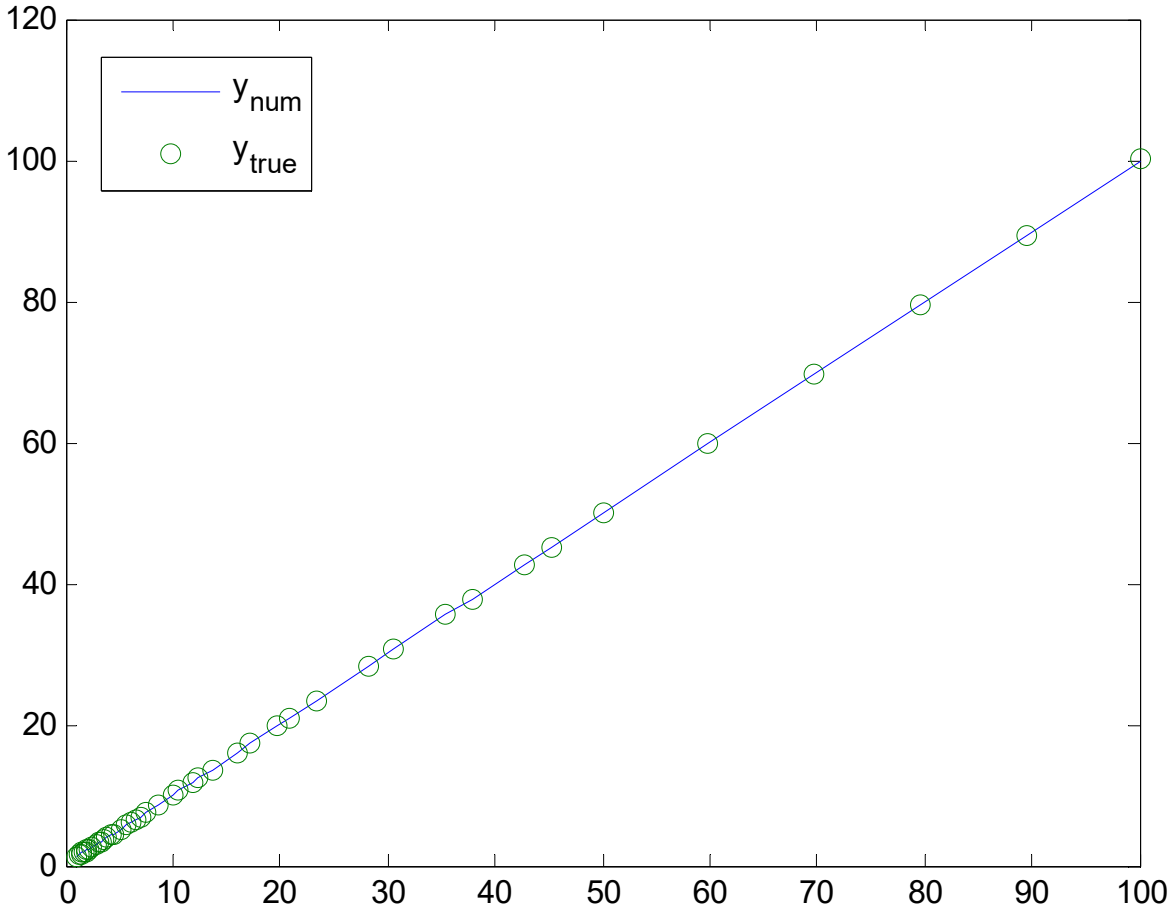
$$y = \sqrt{t^2 + 0.5}$$

```
>> ytrue = sqrt(t.^2 + 0.5);
```

- Then plot both together to compare

```
>> plot(t,y,t,ytrue,'o')
```

Fully Implicit ODE – Example



Exercise 1

- *Use* ode45 *to plot the solution of the initial value problem*

$$x' = \frac{\cos t}{2x - 2} \qquad x(0) = 3$$

- *on the interval* [0, 2π].

Solution – Exercise 1

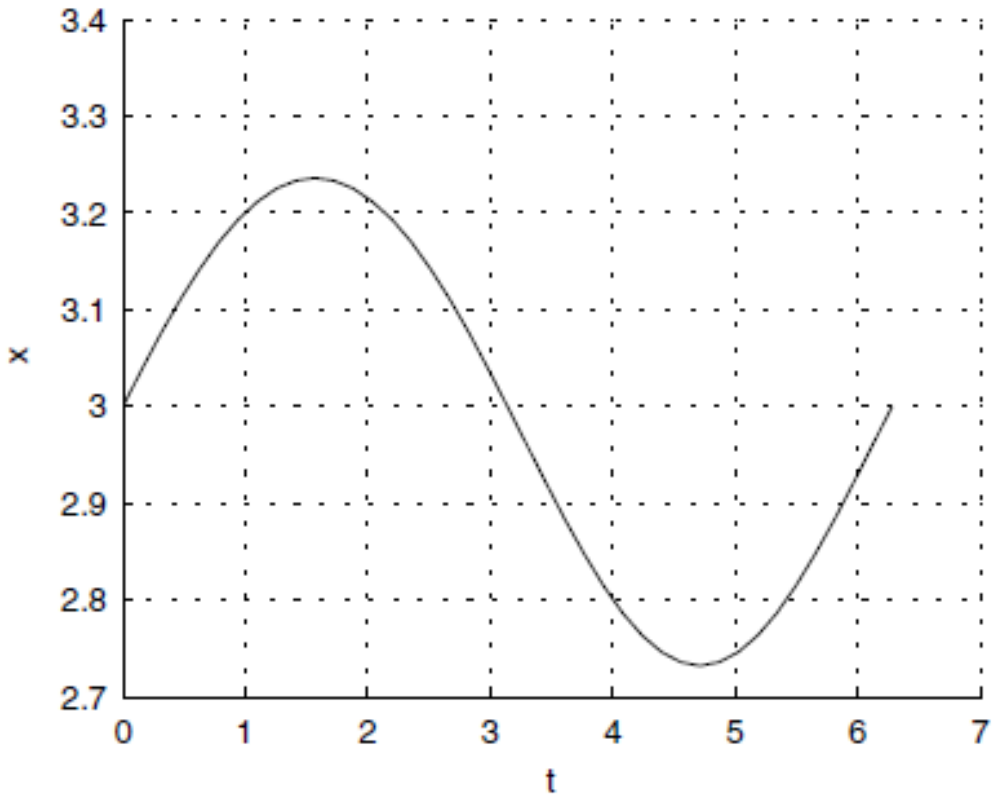
- We need to encode the odefcn.

```
function xprime = ex1(t,x)
xprime = cos(t)/(2*x - 2);
```

- Then sove it

```
>> [t,x] = ode45(@ex1,[0,2*pi],3);
>> plot(t,x)
>> title('The solution x'=cos(t)/(2x-2))
>> xlabel('t'), ylabel('x')
```

Solution – Exercise 1



Using `inline` functions

- If you do not want to save your work, the easiest way to encode the needed odefcn is to create the **`inline`** function

```
>> f = inline('cos(t)/(2*x - 2)','t','x')
f =
Inline function:
f(t,x) = cos(t)/(2*x - 2)
```

- Then, solve it as follows:

```
>> [t,x] = ode45(f,[0,2*pi],3);.
```

Exercise 2

- *Use* ode45 *to solve the initial value problem*

$$x_1' = x_2 - x_1^2,$$
$$x_2' = -x_1 - 2x_1x_2,$$

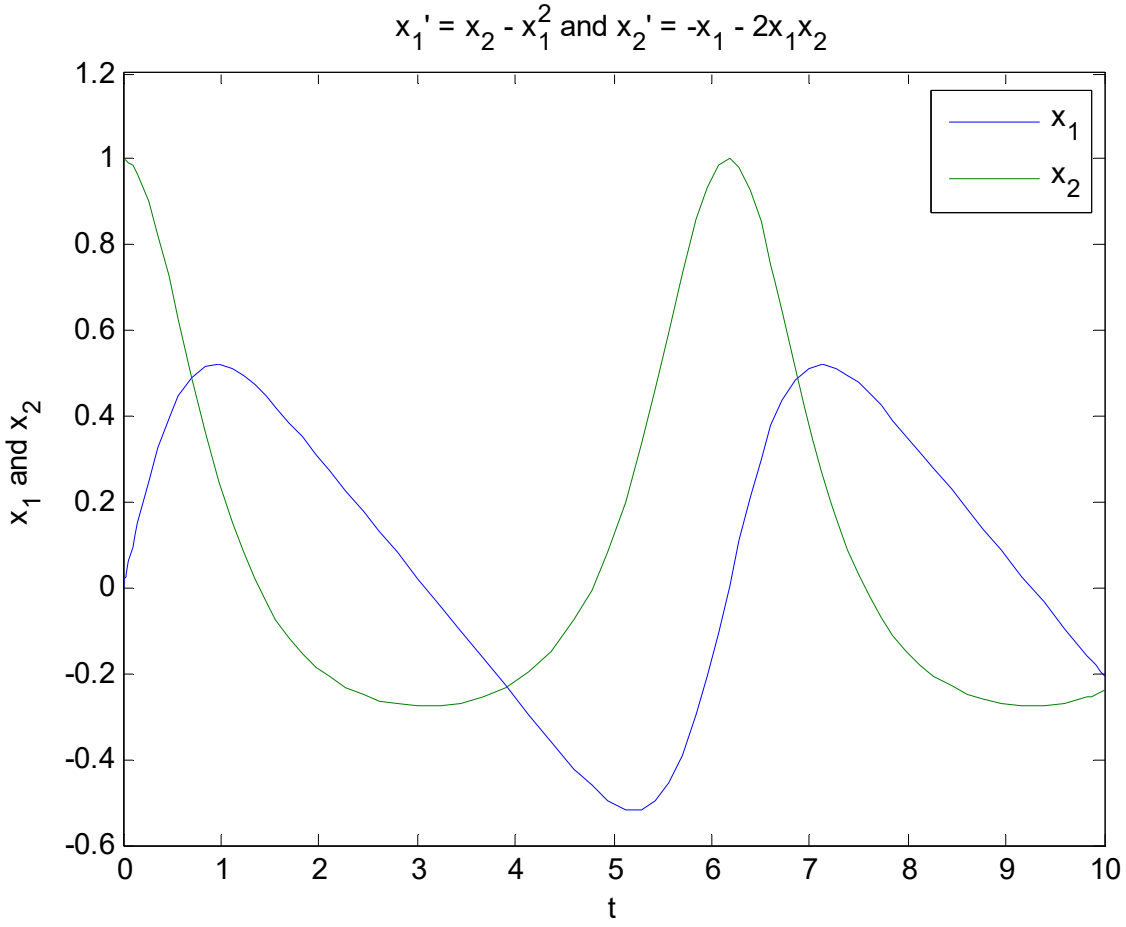
on the interval [0, 10]*, with initial conditions* $x_1(0) = 0$ *and* $x_2(0) = 1$.

Solution – Exercise 2

```
function xprime = F(t,x)
xprime = zeros(2,1);
xprime(1) = x(2) - x(1)^2;
xprime(2) = -x(1) - 2*x(1)*x(2);

>>[t,x] = ode45(@F,[0,10],[0;1]);
>> title('x_1'' = x_2 - x_1^2 and x_2'' = -
x_1 - 2x_1x_2')
>> xlabel('t'), ylabel('x_1 and x_2')
>> legend('x_1','x_2')
```

Solution – Exercise 2

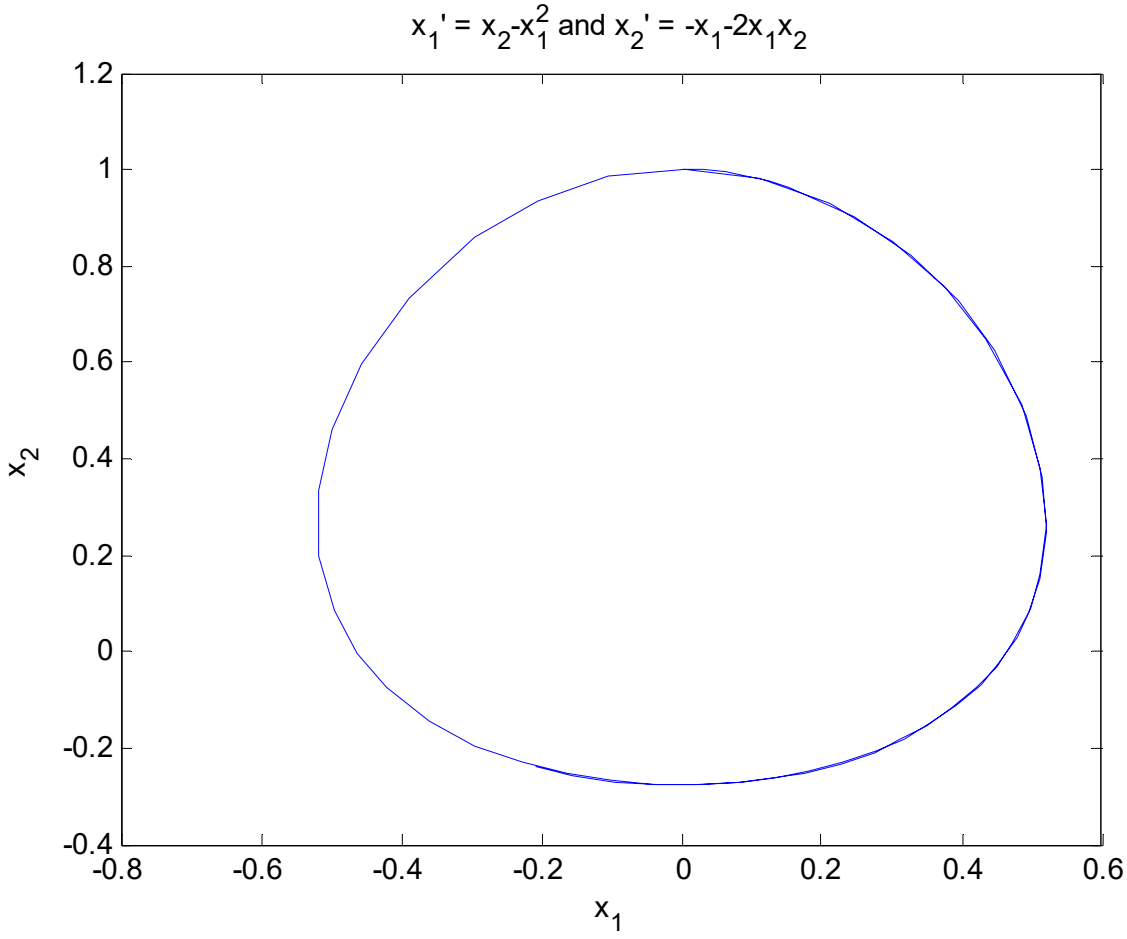


Try another plots

- x_2 vs. x_1

```
>> plot(x(:,1),x(:,2))
>> xlabel('x_1'), ylabel('x_2')
```

x_2 vs. x_1


x1' = x2-x1^2 and x2' = -x1-2x1x2
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
x2
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
x1

And 3D plot

```
>> plot3(t,x(:,1),x(:,2))
>> xlabel('t'), ylabel('x_1'),zlabel('x_2')
```