Math 422
Lia Vas

Systems of ODE in MATLAB

You can find the symbolic solutions of a system of differential equations by using the command **dsolve**. For example to solve the system

dx/dt=2x-y   dy/dt=3x-2y

you can use
>> [x,y] = dsolve('Dx = 2*x - y', 'Dy = 3*x - 2*y', 't')

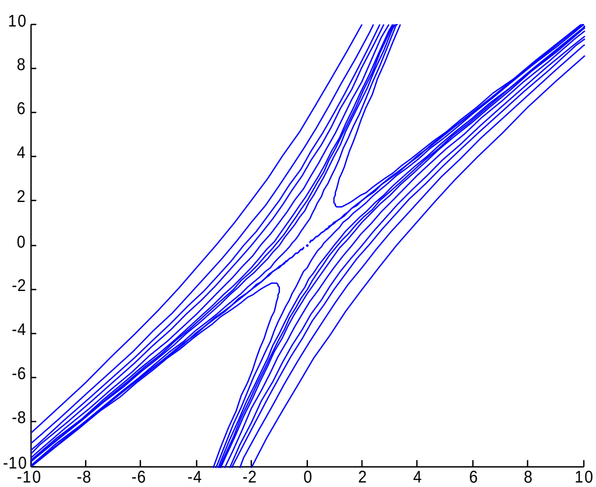
If you have the initial conditions, x(0)=1 and y(0)=2, you can use
>> [x,y] = dsolve('Dx = 2*x - y', 'Dy = 3*x - 2*y', 'x(0) = 1', 'y(0) = 2', 't')

To graph these two solutions on the interval 0≤t≤20, you can use

```
ezplot(x, [0,20])
hold on
ezplot(y, [0,20])
hold off
```

To get the trajectories in the phase plane, you can graph a few solutions with different initial conditions on the same plot. For example, to graph the solutions for initial conditions x(0) and y(0) taking integer values between -2 and 2 and the value of parameter t between -3 and 3 taking 0.1 as a step size (thus, the t values will be: -3, -2.9, -2.8,...., 2.8, 2.9, 3) you can use the following M-file.

```
close all; axes; hold on
t = -3:0.1:3;
for a = -2:2
  for b = -2:2
    echo off
    [x,y] = dsolve('Dx = 2*x - y', 'Dy = 3*x -
2*y', 'x(0) = a', 'y(0) = b', 't');
        xv = inline(vectorize(x), 't', 'a', 'b');
        yv = inline(vectorize(y), 't', 'a', 'b');
    plot(xv(t, a, b), yv(t, a, b))
  end
end
hold off
axis([-10 10 -10 10])
```



Finding symbolic solutions might be very limiting. For example, many systems of differential equations cannot be solved explicitly in terms of elementary functions. For those equations or systems of equations, numerical methods are used in order to get the approximate solution. To find numeric solutions, you can use the command **ode45**. In order to use it, the system needs to be in the form x'=f(x,y,t) and y'=g(x,y,t) and the right sides of the equations should be represented as a vector using the command **inline** first.

For example, to find the numerical solution of the system

$$dx/dt=2x-x^2-xy \qquad dy/dt=xy-y$$

with the initial conditions $x(0)=1$ and $y(0)=2$, we first inline the right side of equation to be a function of independent variable t and unknowns x and y that are represented by **y(1)** and **y(2)** respectively.

```
f = inline('[2*y(1)-y(1)^2-y(1)*y(2); y(1)*y(2)-y(2)]','t','y');
```

Note that here **y(1)** stands for unknown function x, **y(2)** stands for unknown function y. The first entry of **f** is the right side of the first equation and the second entry of **f** is the right side of the second equation. Then the command

```
[t,y]=ode45(f,[0,20],[1;2])
```

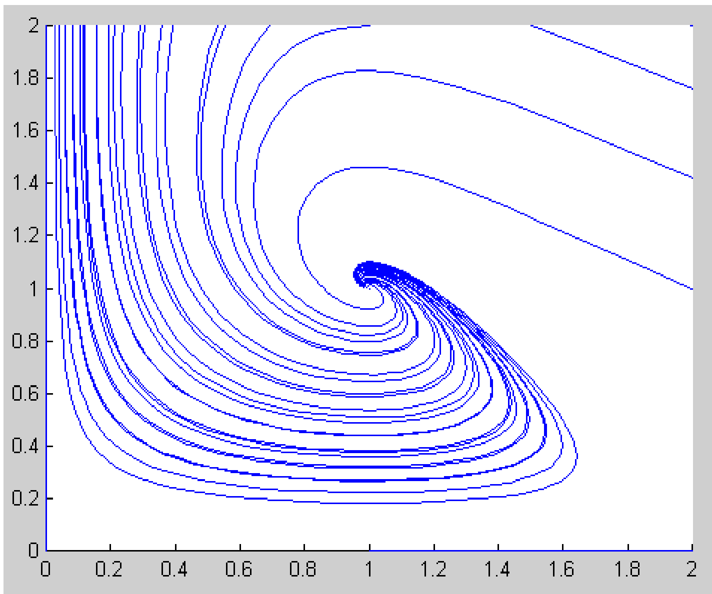
creates a table of t,x and y values for $0\le t\le 20$ starting at $t=0$, $x=1$ and $y=2$ (note the initial conditions $x(0)=1$ and $y(0)=2$). Note that here **y** is a vector whose entries will be the values of **y(1)** and **y(2).**

The command **ode45(f,[0,20],[1;2])**
will graph the two solutions on the same plot as functions of t. The function x will be graphed in blue and y in green.

To graph the trajectory of the solution with the initial conditions x(0)=1 and y(0)=2 in the phase plane, you can use:
[t,y] = ode45(f,[0,20],[1;2]); (If you end the command with ';' the values will not be displayed)
plot(y(:,1),y(:,2)) (plots the first and second entry of the vector **y**.)

The following M-file can be used to graph the trajectories in the phase plane for x(0) and y(0) taking integer initial values between 0 and 5.

```
close all; hold on
for a = 0:5
  for b = 0:5
    [t, y] = ode45(f, 0:0.2:20, [a; b]);
    plot(y(:,1), y(:,2))
  end
end
hold off
axis([0 2 0 2])
```



The outcome is the graph on the right.

Modifying the values of **a** and **b** and the **axis** command, you can get graphs of different density and position and on different window.