Solving systems of first order equations with ode45

The MATLAB numerical solver, **ode45** is designed to work with first order differential equations and *systems of first order equations*. We have already seen how to find approximate solutions to a single, first-order initial value problem, and the procedure for systems of first order equations is completely analogous.

First a *brief* summary of systems of first order equations and initial value problems.

1. Systems of first order differential equations.

A system of first order differential equations looks like this:

$$\left.\begin{array}{rcl} y_1' &=& f_1(t, y_1, y_2, \ldots, y_k) \\ y_2' &=& f_2(t, y_1, y_2, \ldots, y_k) \\ \vdots & & \vdots \\ y_k' &=& f_k(t, y_1, y_2, \ldots, y_k) \end{array}\right\} \tag{1.1}$$

where $y_1, y_2, \ldots, y_k$ are interrelated functions. A first order system like (1.1) may be accompanied by initial values

$$y_1(t_0) = \eta_1,\ y_2(t_0) = \eta_2,\ \ldots,\ y_k(t_0) = \eta_k,$$

giving a (k-dimensional) initial value problem.

A better way to view a system of first order equations like (1.1) is as a single vector-valued equation

$$\overline{\mathbf{y}}' = \mathbf{F}(t, \overline{\mathbf{y}}), \tag{1.2}$$

where

$$\overline{\mathbf{y}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}, \quad \overline{\mathbf{y}}' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_k' \end{bmatrix} \text{ and } \mathbf{F} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_k \end{bmatrix}.$$

This is the way that MATLAB views things. In this format, an initial value problem looks like

$$\overline{\mathbf{y}}' = \mathbf{F}(t, \overline{\mathbf{y}}), \quad \overline{\mathbf{y}}(t_0) = \overline{\eta},$$

where

$$\overline{\eta} = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_k \end{bmatrix}.$$

2. Using ode45 to solve vector-valued initial value problems.

Recall that to compute an approximate solution to the single (scalar-valued) initial value problem

$$y' = f(t, y), \qquad y(t_0) = y_0$$

on the interval $[t_0, t_1]$, we use the MATLAB commands

```
>> f=@(t,y) ... (definition of f(t,y));
>> [t  y] = ode45(f,[t0 t1],y0);
```

To find the approximate solutions to *vector-valued*, first order initial value problems, the syntax is almost identical, except that we have to remember that the variables and functions are now (column) vectors.

Specifically, to compute an approximate solution to the vector-valued initial value problem

$$\begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_k' \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, \ldots, y_k) \\ f_2(t, y_1, \ldots, y_k) \\ \vdots \\ f_k(t, y_1, \ldots, y_k) \end{bmatrix}; \quad \begin{bmatrix} y_1(t_0) \\ y_2(t_0) \\ \vdots \\ y_k(t_0) \end{bmatrix} = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_k \end{bmatrix} \tag{2.1}$$

on the interval $[t_0, t_1]$, we use the MATLAB commands

```
>> F=@(t,y) [def. of f_1(t,y(1),...,y(k)); ...; def. of f_k(t,y(1),...,y(k))];
>> [t  y]=ode45(F, [t0  t1], [η1  η2  ...  ηk]);
```

where now, y is a $k \times n$ matrix, whose j^{th} row y(j) contains the (approximate) values of $y_j(t)$. If we want to plot the graphs of one (or more) of the (approximate) solutions, we use the command

```
>> plot(t, y(:,j))
```

to plot y_j.

3. Examples.

Example 1. Use ode45 to plot (approximate) solutions of the system below on the interval $[0, 5]$.

$$\left.\begin{aligned} y_1' &= \frac{t y_1}{y_1^2 + y_2^2 + 1} & y_1(0) &= 1 \\ y_2' &= \frac{(y_2 - y_1)^2}{y_2^2 + y_3^2 + 1} & y_2(0) &= 0 \\ y_3' &= \frac{t^2 y_3^2}{y_1^2 + y_3^2 + 1} & y_3(0) &= -2 \end{aligned}\right\} \tag{3.1}$$

The MATLAB commands

```
>> F=@(t,y) [t.*y(1)./(y(1).^2+y(2).^2+1);
                    (y(2)-y(1)).^2./(y(2).^2+y(3).^2+1);
                                t.^2.*y(3).^2./(y(1).^2+y(3).^2+1)];
>> [t y]=ode45(F,[0 2], [1 0 -1]);
>> plot(t, y(:,1), 'r', t, y(:,2), 'g', t, y(:,3), 'b')
```

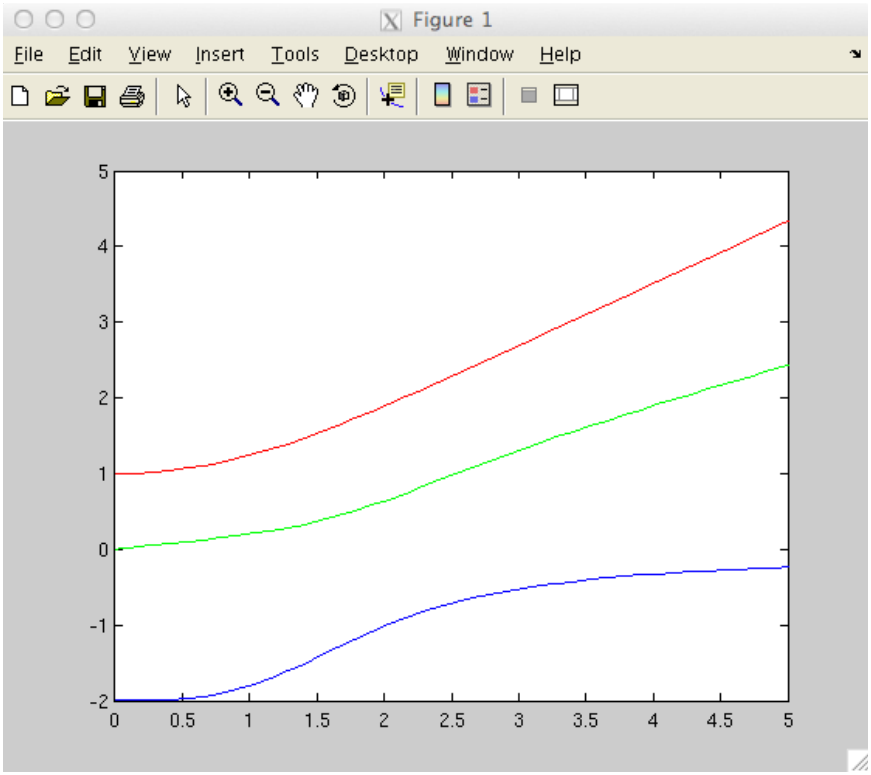

Figure 1: The output for Example 1.

plot the graph of y_1 in red, the graph of y_2 in green and the graph of y_3 in blue, producing the plot in Figure 1 above.

Example 2. Suppose a pendulum has length $\ell = 2$ feet, and at time $t = 0$, it is displaced from its equilibrium position by the angle $\vartheta_0 = \pi/6$ and released (with initial velocity 0).[†] Plot the position (angle) of the pendulum as a function of time t, assuming that there is no damping, and determine the (approximate) period of the motion (i.e., determine when the pendulum first returns to its initial position).

We want to find the solution of the second order initial value problem

$$\vartheta'' + 16.87 \sin(\vartheta) = 0, \quad \vartheta(0) = \frac{\pi}{6}, \ \vartheta'(0) = 0,$$

where $\ell = 2$ (and $g \approx 32.174$). We can't solve this problem explicitly, so instead we will compute an approximate solution using **ode45**.

To do this, we first need to transform the second order initial value problem into a pair of first order initial value problems

$$\vartheta'' + \frac{g}{\ell} \sin(\vartheta) = 0, \quad \vartheta(0) = \frac{\pi}{6}, \ \vartheta'(0) = 0 \implies \begin{cases} \vartheta_1' &= \vartheta_2 & \vartheta_1(0) = \dfrac{\pi}{6} \\ \vartheta_2' &= -16.87 \sin(\vartheta_1) & \vartheta_2(0) = 0 \end{cases}$$

where $\vartheta_1 = \vartheta$ and $\vartheta_2 = \vartheta'$.

Next we use the MATLAB commands

[†]See section 1.3 in the book for the derivation of the motion of an undamped pendulum.

```
>> Pen=@(t,Theta) [Theta(2); -16.87*sin(Theta(1))];
>> [t Theta]=ode45(Pen,[0 5],[pi/6 0]);
>> plot(t,Theta(:,1))
>> grid minor
```

to produce the plot in Figure 2, below. The last command 'grid minor', adds a fine grid to the plot which gives a better view of where the maxima are to determine the period.

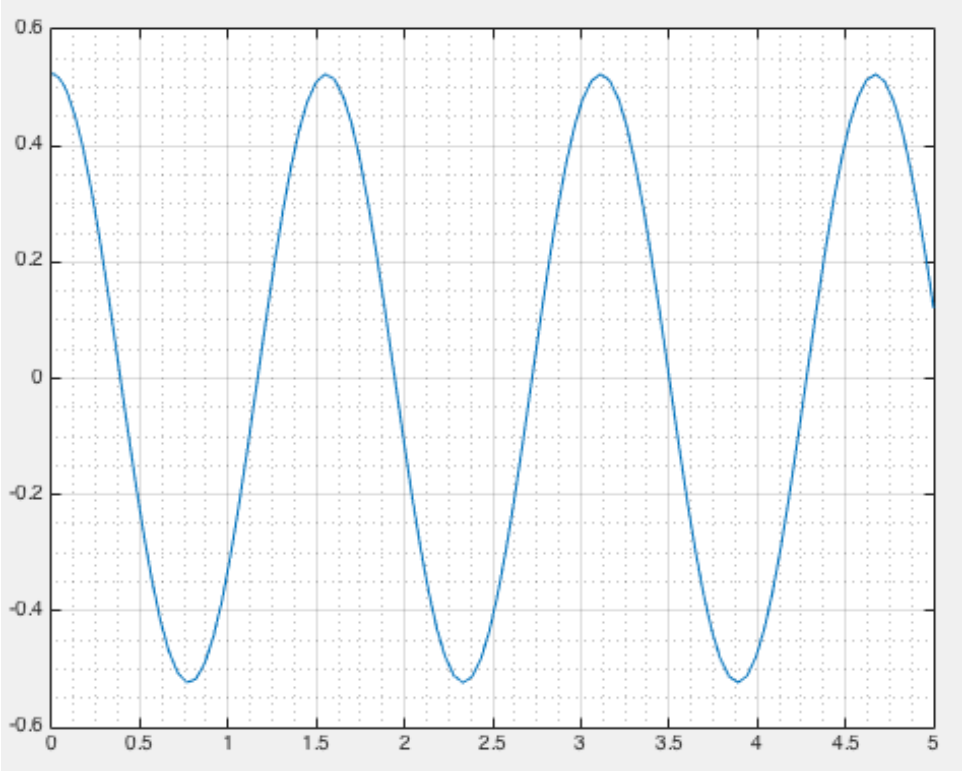

Figure 2: Plot of the position of the pendulum in Example 2.

From the plot, it appears that the pendulum first returns to its initial position about halfway between 1.5 and 1.6, i.e., at about $t_p = 1.55$.

you can study the values stored in T(:,1) to find the index where $\pi/6 \approx 0.5236$ appears for the second time. Then, with that index you can check the corresponding entry in the time vector t. If you do this, you will find that the pendulum first returns to its initial position after about 1.5505 seconds.

4. The phase plane, direction fields and phase portraits for 2x2 autonomous systems.

The solutions of the pair of first-order differential equations

$$\overline{\mathbf{x}}' = \overline{\mathbf{F}}(\overline{\mathbf{x}})$$

are distinguished by

Suppose that $\overline{\xi}(t) = \begin{bmatrix} \xi_1(t) \\ \xi_2(t) \end{bmatrix}$ is a particular solution of the system then the graph whose points are $\{(\xi_1(t), \xi_2(t)) : a < t < b\}$ is called the *trajectory* of this solution, and it is plotted in the *phase-plane*. A phase-portrait for the system of equations consists of a collection of typical trajectories for the system (corresponding to different initial conditions).

Plotting trajectories in MATLAB is easy, and plotting a collection of trajectories to produce a phase portrait is just as easy.

Example 3. Consider the nonlinear system below.

$$\left.\begin{aligned} x_1' &= 1.1x_1 - 0.3x_1x_2 \\ x_2' &= 0.2x_1x_2 - 0.5x_2 \end{aligned}\right\} \tag{4.1}$$

Using ode45, we can produce different solutions to this (nonlinear) system (corresponding to different initial conditions) and then graph all the solutions in the same plot, as in Figure 3, below.

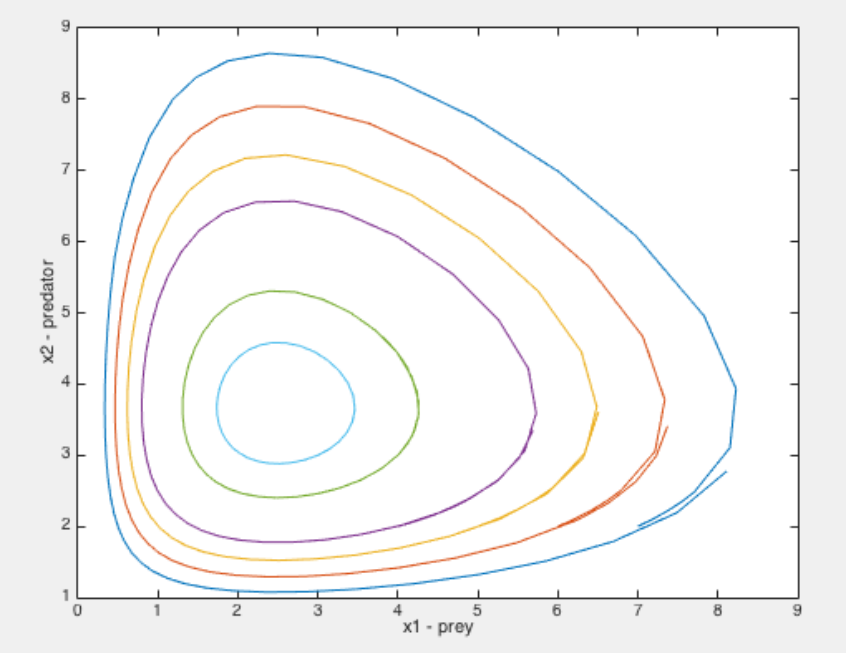

Figure 3: Phase portrait for the system (4.1).

```
>> LV=@(t,X) [1.1*X(1)-0.3*X(1).*X(2); 0.2*X(1).*X(2)-0.5*X(2)];
>> [t X]=ode45(LV,[0 10],[7 2]);
>> plot(X(:,1),X(:,2));
>> hold on
>> [t X]=ode45(vdp,[0 10],[6 2]);
>> plot(X(:,1),X(:,2));
                    etc.
```

Commands to produce Figure 3, above.

Another common way to describe the possible solutions of a pair of *autonomous* first-order differential equations is to draw a direction field in the phase plane.Each point in the phase-plane can be thought of as the initial point of a particular solution $\overline{\mathbf{x}}(t)$ of the system

$$\overline{\mathbf{x}}'(t) = \overline{\mathbf{F}}(\overline{\mathbf{x}}) \tag{4.2}$$

In other words, if (ξ_1, ξ_2) is a point of the phase-plane, then there is a particular solution $\overline{\mathbf{x}}(t)$ of the system 4.2 and a point t_0 such that

$$\begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \overline{\mathbf{x}}(t_0) = \begin{bmatrix} x_1(t_0) \\ x_2(t_0) \end{bmatrix}.$$

To produce a direction field for the system (4.2), we choose and array of points in the phase plane, and then at each point $(\xi_1, \xi_2) = \overline{\xi}^{\mathrm{T}}$ in the array, we draw an arrow parallel to $\overline{\mathbf{F}}(\overline{\xi})$ with its tail at (ξ_1, ξ_2). In MATLAB , we can use the **quiver** command to do this, as illustrated below.

Example 4. Sketch a direction field for the system (4.1) in the first quadrant. The sequence of MATLAB commands below, produces the direction field in Figure 4.

```
>> [x1 x2] = meshgrid(0:0.2:8,0:0.2:8);
>> dx1=1.1*x1-0.3*x1.*x2;
>> dx2=0.2*x1.*x2-0.5*x2;
>> L=sqrt(dx1.^2+dx2.^2);
>> quiver(x1,x2,dx1./L,dx2./L,0.6); axis tight
```

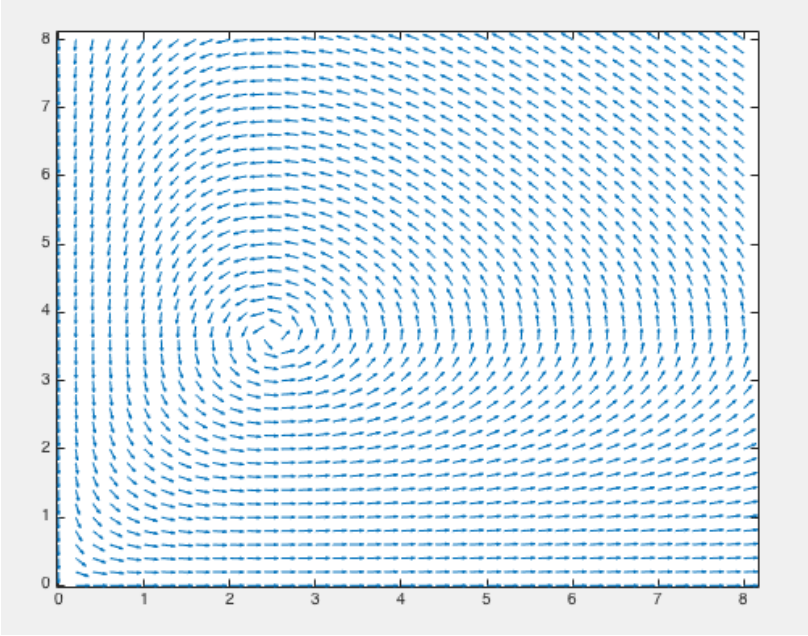

Figure 4: Direction field for the system (4.1).

Example 5. Plot a direction field and three trajectories for the system

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.5 & 2 \\ -2 & -0.5 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}.$$

Since we need some trajectories, we first find the general solution of the system:

$$\begin{bmatrix} x \\ y \end{bmatrix} = e^{-t/2}\begin{bmatrix} c_1\cos(2t) + c_2\sin(2t) \\ -c_1\cos(2t) + c_2\sin(2t) \end{bmatrix}.$$

Next, we plot three trajectories, starting at the initial points

$$\bar{\xi}_1 = \begin{bmatrix} 2 \\ 1.5 \end{bmatrix}, \quad \bar{\xi}_2 = \begin{bmatrix} 1 \\ -3 \end{bmatrix} \text{ and } \bar{\xi}_3 = \begin{bmatrix} -2 \\ -2.5 \end{bmatrix}.$$

The sequence of commands below produces the plot in Figure 5.

```
>> t=[0:0.05:8];
>> x=exp(-t/2).*(2*cos(2*t)+1.5*sin(2*t));
>> y=exp(-t/2).*(-2*sin(2*t)+1.5*cos(2*t));
>> plot(x,y,'r');
>> hold on
>> x=exp(-t/2).*(1*cos(2*t)-3*sin(2*t));
>> y=exp(-t/2).*(-1*sin(2*t)-3*cos(2*t));
>> plot(x,y,'g');
>> x=exp(-t/2).*(-2*cos(2*t)-2.5*sin(2*t));
>> y=exp(-t/2).*(2*sin(2*t)-2.5*cos(2*t));
>> plot(x,y,'b');
>> grid on
```

Since we want to plot the direction field in the same figure, we need to choose the limits of the meshgrid to be big enough to include the plot of the three trajectories.‡ Choosing $-3 \le x \le 3$ and $-3 \le y \le 3$, should work, and following the preview commands with

```
>> [x y]=meshgrid(-3:0.25:3,-3:0.25:3);
>> dx=-0.5*x+2*y;
>> dy=-2*x-0.5*y;
>> L=sqrt(x.^2+y.^2);
>> quiver(x,y,dx./L,dy./L,0.5); axis tight
```

yields the plot in Figure 6.

‡Otherwise, there will be unattractive white space in the figure.

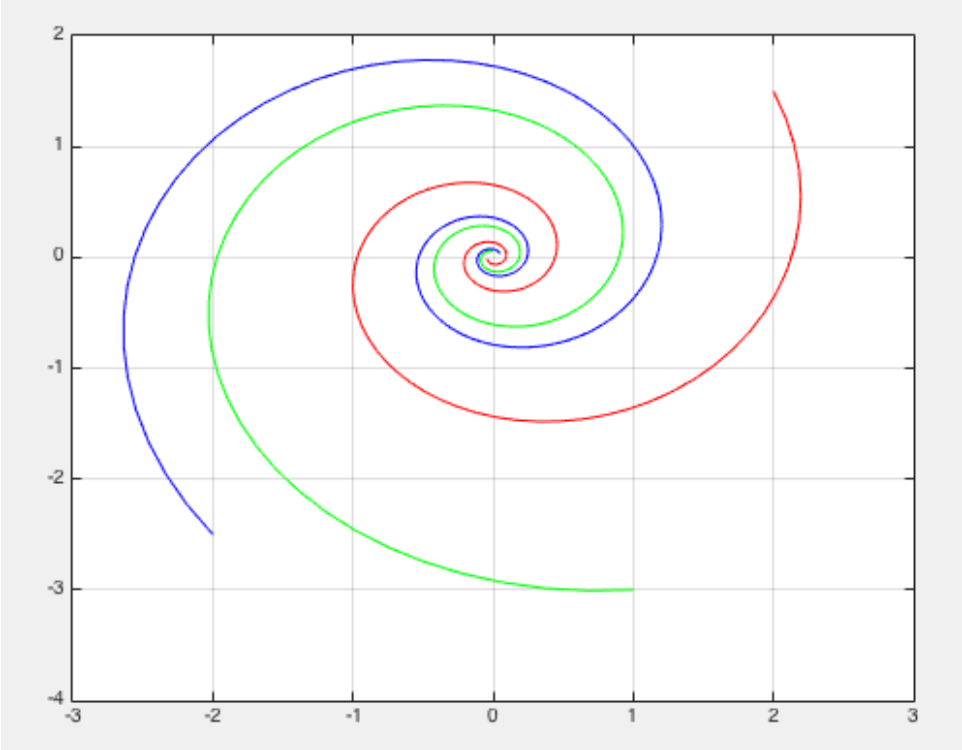

Figure 5: Trajectories for Example 5.

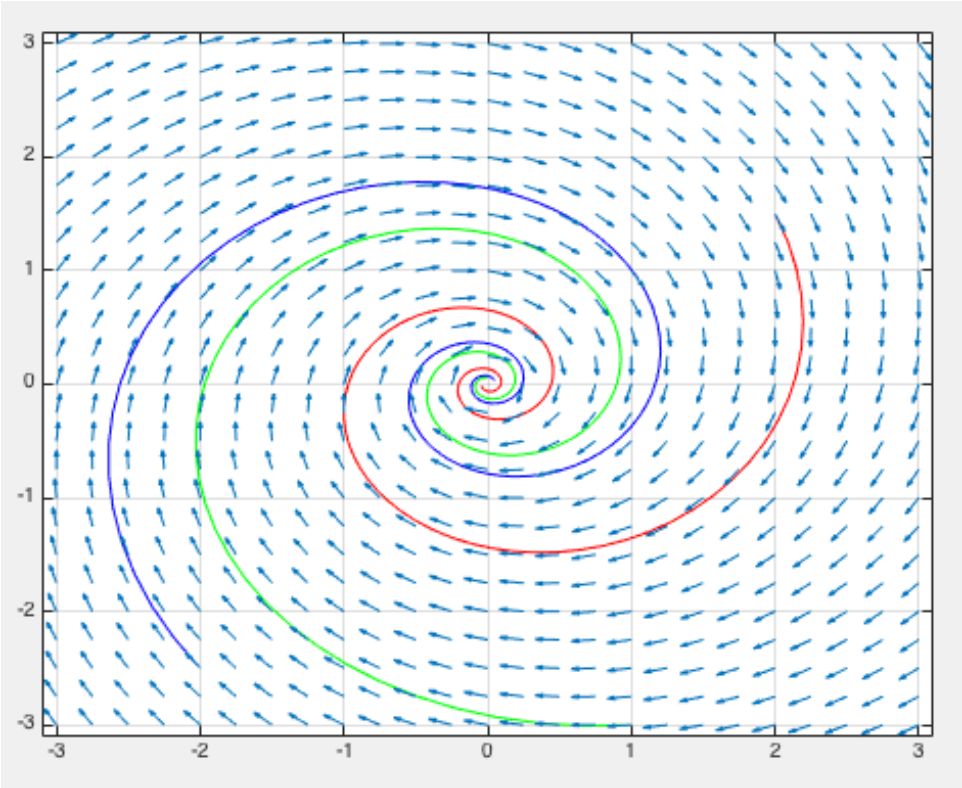

Figure 6: Trajectories and direction field for Example 5.