

Justin's Math 246 Guide and Project 3

Due Friday 16 August 2019

Contents

- What to Submit
- Multiplying Matrices
- Eigenvalues and Eigenvectors
- Solving Linear Systems of Differential Equations
- Nonlinear Systems

What to Submit

For this project you will need to create an m-file which does all the tasks listed in this project in order. You should then publish the m-file and print.

Each task should be within its own Matlab code section so that the result of each task appears directly after that task, not all at the end. If you're not sure how to properly mark up an m-file for publication like this then ask me.

Multiplying Matrices

We've seen how to take determinants of matrices. Let's just see how we can store them, multiply them and find the eigenvalues of them. First, here are a couple of matrices:

```
A = [1 2 3;-1 0 2;0 3 1]
B = [1 0 1;2 1 3;1 0 -1]
```

A =

```
     1     2     3
    -1     0     2
     0     3     1
```

B =

```
     1     0     1
     2     1     3
     1     0    -1
```

Here we multiply them:

A*B

ans =

```
      8      2      4
      1      0     -3
      7      3      8
```

Task 1: Assign the matrices and calculate the matrix products AB and BA .

$$A = \begin{bmatrix} 1 & 2 & -3 \\ 0 & 4 & 4 \\ -2 & 5 & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} -4 & 0 & 1 \\ 2 & 2 & -2 \\ 7 & -5 & 2 \end{bmatrix}$$

Eigenvalues and Eigenvectors

For finding eigenvalues and eigenvectors it can be nicer to declare the matrix as symbolic. Matlab treats it a bit differently and will often return nicer answers. All we do is wrap the matrix in `sym`. Here we find the eigenvalues and eigenvectors of a 2×2 matrix:

```
C = sym([3 -1;4 -2])
[evect,eval] = eig(C)
```

C =

```
[ 3, -1]
[ 4, -2]
```

evect =

```
[ 1/4, 1]
[  1, 1]
```

eval =

```
[ -1, 0]
[  0, 2]
```

This is a bit confusing at first. The first part, `evect`, contains the eigenvectors as columns. The second part, `eval` contains the corresponding eigenvalues as entries in a diagonal matrix. So $\lambda = -1$ corresponds to $\begin{bmatrix} 1/4 \\ 1 \end{bmatrix}$ and $\lambda = 2$ corresponds to $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Task 2: Find the eigenvectors and eigenvalues of the matrix

$$\begin{bmatrix} 2 & -5 \\ 1 & -2 \end{bmatrix}$$

Task 3: Find the eigenvectors and eigenvalues of the matrix

$$\begin{bmatrix} -1 & -1 & 0 \\ 5 & -3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Solving Linear Systems of Differential Equations

Matlab's `dsolve` command can solve systems of linear differential equations. For example if $x(t)$ and $y(t)$ satisfy the system:

$$\begin{aligned} x' &= 2x + y \\ y' &= x + 2y \end{aligned}$$

Then we can solve this in Matlab with:

```
syms x(t) y(t)
[xsoln,ysoln] = dsolve(diff(x) == 2*x+y, diff(y) == x+2*y)

xsoln =

C2*exp(3*t) - C1*exp(t)

ysoln =

C1*exp(t) + C2*exp(3*t)
```

Task 4: Solve the system of differential equations:

$$\begin{aligned}x' &= 3x - y \\ y' &= 4x - 2y\end{aligned}$$

Matlab can also solve an initial value problem:

```
syms x(t) y(t)
[xsoln,ysoln] = dsolve(diff(x) == 2*x+y, diff(y) == x+2*y,x(0)==1,y(0)==-2)

xsoln =

(3*exp(t))/2 - exp(3*t)/2

ysoln =

- exp(3*t)/2 - (3*exp(t))/2
```

Task 5: Solve the previous problem's differential equation along with $x(0) = 2$ and $y(0) = -3$.

As a note this differential equation (not the IVP) in matrix form is

$$X' = AX \text{ where } X = X(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \text{ and } A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

We can also put this in Matlab:

```
clear all
syms x(t) y(t)
X=[x;y];
A = [2 1;1 2];
[xsoln,ysoln] = dsolve(diff(X) == A*X)

xsoln =

C2*exp(3*t) - C1*exp(t)

ysoln =

C1*exp(t) + C2*exp(3*t)
```

Task 6: Solve the system of differential equations using matrix notation:

$$\begin{aligned}x' &= 2x - 5y \\ y' &= -x + 2y\end{aligned}$$

Nonlinear Systems

`dsolve` usually won't handle nonlinear systems because most such systems don't have "nice" solutions. However Matlab's `ode45` command can give numerical (approximate) solutions if an initial value is given. By an approximate solution this means a collection of points whose graph is approximately a solution.

The Matlab `ode45` command will work on linear systems too. Consider:

$$\begin{aligned}x' &= 2y & x(0) &= 1 \\y' &= -2x & y(0) &= 0\end{aligned}$$

The solution to this initial value problem is $x(t) = \cos(2t)$ and $y(t) = -\sin(2t)$ so if we plot this for $0 \leq t \leq \frac{\pi}{2}$ we ought to get a semicircle.

To plot this in Matlab we don't use `x` and `y` but rather a single `x` which we treat as a vector and reference with `x(1)` and `x(2)`. We define a function handle of the time variable `t` along with this `x`:

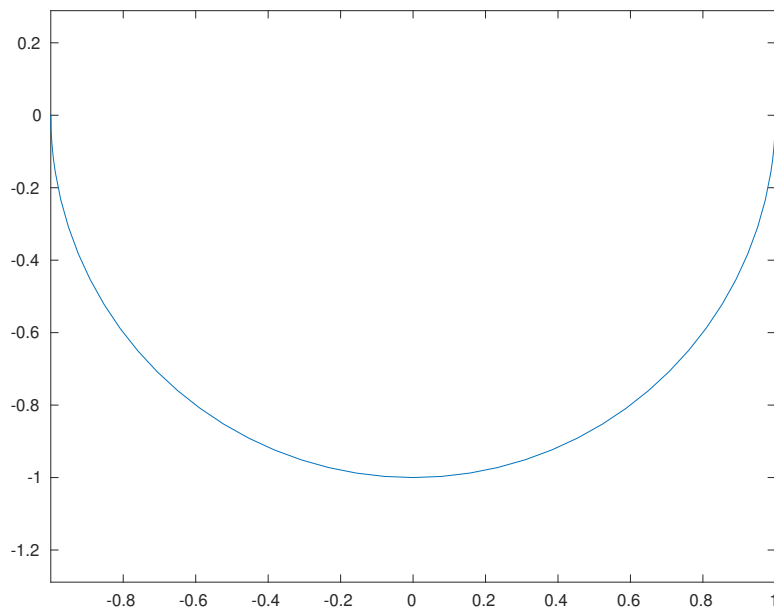
```
f = @(t,x) [2*x(2);-2*x(1)];
```

We then use `ode45`, passing it `f`, the range of t values, and the initial value:

```
[t,xsoln] = ode45(f,[0 pi/2],[1,0]);
```

Matlab returns two items, the first is a vector of t -values and the second is an array of points. We plot the array of points and make sure the axes are proportional and see the semicircle:

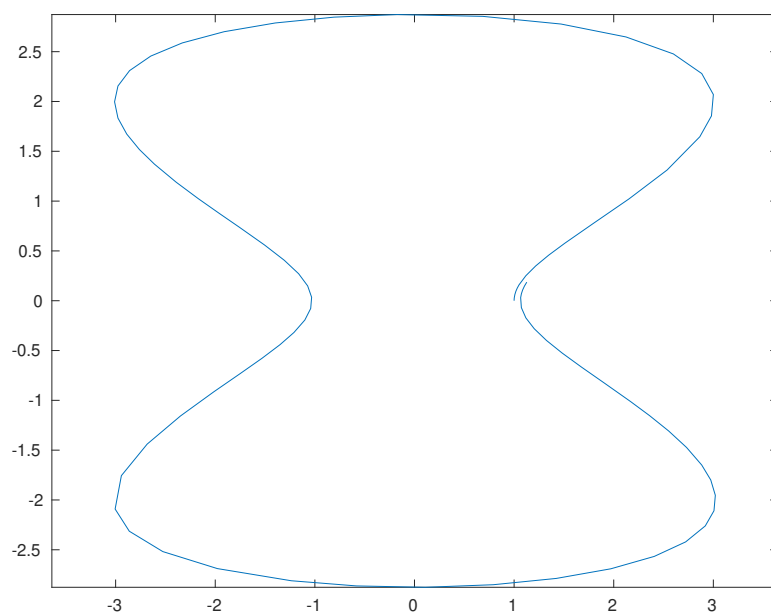
```
plot(xsoln(:,1),xsoln(:,2))  
axis equal
```



Here's another from class, plotted for $0 \leq t \leq 2\pi$.

$$\begin{aligned} x' &= 4y - y^3 & x(0) &= 1 \\ y' &= x & y(0) &= 0 \end{aligned}$$

```
f = @(t,x) [4*x(2)-x(2)^3;x(1)];
[t,xsoln] = ode45(f,[0 2*pi],[1,0]);
plot(xsoln(:,1),xsoln(:,2))
axis equal
```



Notice that the ends of the curve don't exactly meet up even though the shape forms a loop as we saw in class. This is a result of the approximation that Matlab is doing as part of the `ode45` command.

Task 6: Plot the solution to the above system with initial value $x(0) = 0$, $y(0) = 1$.

Task 7: Plot the solution the initial value problem for $0 \leq t \leq 4$.

$$\begin{aligned} x' &= 2 - y & x(0) &= 0 \\ y' &= 4 - x^2 & y(0) &= 6 \end{aligned}$$