Differential Equations, Fall 2016
Computer Project 4, Due Wednesday, December 7

The goal of this assignment is to numerically solve some force equals mass times acceleration problems. You should hand in the following things:

1. For problem 1, a printout of the file `f.m` and a printout of the commands and output.
2. For problem 2, a printout of the plot of displacement versus time.
3. For problem 3, a printout of the file `f.m` and a printout of the commands and output.
4. For problem 4, a printout of the plot of the positions of the Earth and the Moon (on the same graph).
5. For problem 5, a printout of the commands and output.

1. (3 points) The goal of this problem is to write a function `f.m` in Matlab to implement the function $\vec{G}$ appearing in a certain differential equation of the form $\vec{x}' = \vec{G}(t, \vec{x})$. The differential equation we will consider arises from a damped spring with an external force $F(t)$. We will take

$$m = 1, \quad c = 1, \quad k = 5, \quad F(t) = \cos(5t),$$

which gives the second order equation

$$x'' + x' + 5x = \cos(5t).$$

Now we want to rewrite this as a system. Introduce dependent variables $x_1 = x_1(t)$ and $x_2 = x_2(t)$, and let

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Let $x_1 = x$ and $x_2 = x'$. Then the second order equation can be rewritten as the two dimensional first order system

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}' = \begin{bmatrix} x_2 \\ -5x_1 - x_2 + \cos(5t) \end{bmatrix}.$$

The right hand side of this is the function $\vec{G}$. In other words,

$$\vec{G}\left(t, \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_2 \\ -5x_1 - x_2 + \cos(5t) \end{bmatrix}.$$

Now code this function $\vec{G}$ into the Matlab function `f.m`. The outline of the function should be

```
function val=f(t,x)

end
```

Then run the following commands:

```
>format long
>f(1,[2 3]')
>f(-2,[3 7]')
```

Turn in a printout of the file `f.m` and a printout of the commands and output.

2. (3 points) The goal of this problem is to use the Runge-Kutta Method to solve the system from the previous problem. Download `RungeKuttaMethod.m` from the class website, it can be found in the solutions for Computer Project 3. We will consider an initial value problem with $x(0) = 1$, $x'(0) = 0$, and will consider the time interval $0 \leq t \leq 30$. Then the following command will solve it, using 10,000 steps:

```
>[t x]=RungeKuttaMethod(0,30,10000,[1 0]');
```

The list of time values is stored in t and the list of $\vec{x}$ values is stored in x. Now generate a plot of the displacement of the spring versus time. Print it out and hand it in.

Food for thought (don't hand in): What happens if there is no damping (so $c = 0$) and the external force $F(t)$ is $F(t) = \cos(\sqrt{5}t)$? Can you explain this from what we learned in class? Physically, what is happening? To see what is happening it may be useful to look at a longer time interval.

3. (3 points) The goal of this problem is to write a function `f.m` to implement the function $\vec{G}$ in a different differential equation. The differential equation will be the system representing the Earth and the Moon under the law of universal gravitation. The first mass m_1 will be the Earth and the second mass m_2 will be the Moon. Take them to be

$$m_1 = 5.972 \times 10^{24} \text{ kg}, \qquad m_2 = 7.342 \times 10^{22} \text{ kg}.$$

Recall that the Newton's law of universal gravitation says that masses of m_1 kg and m_2 kg a distance r m apart each experience a gravitational force of magnitude

$$Gm_1m_2/r^2,$$

where G is the gravitational constant which we will take to be

$$G = 6.674 \times 10^{-11} \text{ N}\cdot(\text{m/kg})^2.$$

The direction of the force is from one mass to the other. In other words, suppose $\vec{p}_1$ is the position of the Earth and $\vec{p}_2$ is the position of the Moon. Then the force $\vec{F}_1$ acting on the Earth is in the direction of $\vec{p}_2 - \vec{p}_1$, and the force $\vec{F}_2$ acting on the Moon is in the direction of $\vec{p}_1 - \vec{p}_2$. Since $|\vec{p}_1 - \vec{p}_2| = |\vec{p}_1 - \vec{p}_2| = r$, the forces are

$$\vec{F}_1 = Gm_1m_2\frac{\vec{p}_2 - \vec{p}_1}{|\vec{p}_2 - \vec{p}_1|^3}, \quad \vec{F}_2 = Gm_1m_2\frac{\vec{p}_1 - \vec{p}_2}{|\vec{p}_2 - \vec{p}_1|^3}.$$

With two masses, coordinates can be chosen such that the motion always takes place in a plane, so we will assume that $\vec{p}_1$ and $\vec{p}_2$ are vectors in $\mathbb{R}^2$, and so are $\vec{F}_1$ and $\vec{F}_2$. Now introduce dependent variables x_1, x_2, x_3, x_4 and let

$$\vec{p}_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad \vec{p}_2 = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}.$$

These functions represent the positions of the Earth and the Moon. Force equals mass times acceleration then gives the equations

$$\vec{F}_1 = m_1 \vec{p}_1'', \qquad \vec{F}_2 = m_2 \vec{p}_2''.$$

They can be broken down into components, say as

$$\vec{F}_1 = \begin{bmatrix} F_{11} \\ F_{12} \end{bmatrix}, \qquad \vec{F}_2 = \begin{bmatrix} F_{21} \\ F_{22} \end{bmatrix}.$$

$F_{11}, F_{12}, F_{21}, F_{22}$ are functions which take in the 4 inputs x_1, x_2, x_3, x_4 and have 1 output. We can write this as a 4 dimensional second order system

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}'' = \begin{bmatrix} F_{11}/m_1 \\ F_{12}/m_1 \\ F_{21}/m_2 \\ F_{22}/m_2 \end{bmatrix}.$$

By introducing dependent variables x_5, x_6, x_7, x_8 and letting

$$x_5 = x_1', \quad x_6 = x_2', \quad x_7 = x_3', \quad x_8 = x_4'$$

we can write this as an 8 dimensional first order system

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}' = \begin{bmatrix} x_5 \\ x_6 \\ x_7 \\ x_8 \\ F_{11}/m_1 \\ F_{12}/m_1 \\ F_{21}/m_2 \\ F_{22}/m_2 \end{bmatrix}.$$

This is the differential equation we will consider in this problem. The left hand side is $\vec{x}'$ and the right hand side is $\vec{G}(t, \vec{x})$. Now code the function $\vec{G}$ into the Matlab function `f.m`. (You might want to save your `f.m` from the first part of this project into a seperate file first, maybe call it `fspring.m`.) The format of the function should be

```
function val=f(t,x)

end
```

Then run the following commands:

```
>format long
>f(1,[1 2 3 4 5 6 7 8]')
```

Turn in a printout of the file and a printout of the commands and output.

4. (3 points) Now we will solve the ODE from the previous problem. The distance from the Moon to the Earth is about 3.8×10^8 m. The velocity of the Moon is about 1022 m/sec. We want to choose coordinates such that the center of mass is at the origin and does not move. So we will take as initial condition

$$\vec{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 380000000 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1022 \end{bmatrix} - \begin{bmatrix} \frac{380000000m_2}{m_1+m_2} \\ 0 \\ \frac{380000000m_2}{m_1+m_2} \\ 0 \\ 0 \\ \frac{1022m_2}{m_1+m_2} \\ 0 \\ \frac{1022m_2}{m_1+m_2} \end{bmatrix}.$$

Now use the Runge-Kutta Method program to solve the system over the time interval $0 \leq t \leq 3600 * 24 * 28$ with 1000 steps. Turn in a plot of the orbit of the Earth and the Moon (on the same graph). The plot should be a two dimensional plot that shows the positions of the Earth and the Moon. It will not show time dependence.

Here is some code you can run to animate the solution. It assumes that the output of `RungeKuttaMethod` is stored into the variables `t` and `x`. You do not need to hand this in.

```
[r c]=size(x);
plot(x(1,:),x(2,:));
hold on
plot(x(3,:),x(4,:));
earth=plot(x(1,1),x(2,1),'o','MarkerFaceColor','green');
moon=plot(x(3,1),x(4,1),'o','MarkerFaceColor','blue');
for k=2:c
earth.XData=x(1,k);
earth.YData=x(2,k);
moon.XData=x(3,k);
moon.YData=x(4,k);
drawnow
end
```

5. (3 points) Matlab has several built-in functions to solve systems of ODEs. One is called `ode23`. The syntax is slightly different than the programs we created. First of

all, it works with row vectors instead of column vectors. Create a function `frow.m` which is the same as `f.m` except that the second input and the output are row vectors instead of column vectors. Then the following command will run `ode23`:

```
> [t x]=ode23(@frow,[0 3600*24*28],x0);
```

The first input tells it the name of the function to use for the differential equation. The second input tells it the time range. The program automatically uses an appropriate number of steps. The last input is the initial condition, `x0` should be the vector $\vec{x}_0$ from the previous problem except entered as a row vector. The output is similar to `RungeKuttaMethod` except the role of rows and columns is swapped. You can plot the output similarly to before, but remember that rows and columns are swapped now, so the inputs to the plot command will be slightly different. Run the following command and hand in the output:

```
> t(100),x(100,:)
```

6. (For fun only, do not turn in.) Now we put a man-made satellite into the mix. We can describe its position with some extra dependent variables, say x_9, x_{10} and its velocity with $x_{11} = x_9'$ and $x_{12} = x_{10}'$. Assume the mass is $m_3 = 1000$ kg. Since the mass is small compared to the Earth and the Moon, we take the acceleration it induces on the Earth and the Moon to be 0. The accleration x_{11}', x_{12}' of the satellite is the sum of the accelerations due to the gravitational force of the Earth and the moon. Update `f.m` to include the satellite. Then run the command

   ```
   [t x]=RungeKuttaMethod(0,3600*24*40,1000,x0);
   ```

   to solve the resulting system. The vector `x0` will need to have 4 more entries put into it to describe the initial conditions of the satellite. For example, you can use the numbers `100000000 0 0 2000`. You can then plot the trajectory of the satellite using the command `plot(x(9,:),x(10,:))`. You can animate it by modifying the animation code from the previous problem.