Differential Equations, Spring 2017
Computer Project 3, Due 11:30 am, Friday, April 14

The goal of this project is to use the Improved Euler Method to numerically solve the Kepler problem (inverse square central force field problem). Submit the following file on Canvas:

`kepler.m`

Make sure the file has exactly this name.

To simplify things we will consider a particle of mass $m = 1$ kg and assume that the motion lies within a plane, so the position of the particle at time t is $\begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$. For the Kepler problem, the force acting on the particle depends only on the position, and takes the form

$$\vec{F}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = -\frac{1}{r^3}\begin{bmatrix} x \\ y \end{bmatrix},$$

where $r = \sqrt{x^2 + y^2}$. Then the ODE for the Kepler problem is

$$\begin{bmatrix} x''(t) \\ y''(t) \end{bmatrix} = \vec{F}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right).$$

The first step is to rewrite this as a 4 dim 1st order system. You can do this by introducing variables x_1, x_2, x_3, x_4 and letting

$$x_1 = x, \quad x_2 = y, \quad x_3 = x', \quad x_4 = y'.$$

Then, with

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix},$$

the ODE takes the form

$$\vec{x}' = \vec{G}(t, \vec{x})$$

for some function $\vec{G}$. Code this function $\vec{G}$ into a Matlab function of the name `kepler`. Here is an outline of the function

```
function val=kepler(t,x)
    ...
end
```

The variable `t` will not actually be used in the function but we keep it as an argument because the `ImprovedEulerMethod` program expects a function with a `t` input and an `x` input. The Matlab variable `x` stands for the vector variable $\vec{x}$ and not the scalar variable x. Also, remember that our convention is to always use column (vertical) vectors for our vectors (instead of row vectors).

Now you can use the Improved Euler Method to solve the ODE. The program that we wrote before should still work; we just have to pass it in the vector function `kepler` which we created and enter the initial condition as a column vector with 4 entries. Download `ImprovedEulerMethod.m` from the solutions to Computer Project 2 on the class website. Here are some commands you can run to test if everything works.

```
hold on
[t x]=ImprovedEulerMethod(@kepler,0,10,100000,[.5 .2 0 2]');
plot(x(1,:),x(2,:),'b')
[t x]=ImprovedEulerMethod(@kepler,0,10,100000,[.5 .2 0 -2]');
plot(x(1,:),x(2,:),'b')
[t x]=ImprovedEulerMethod(@kepler,0,40,100000,[1 1 0 -1]');
plot(x(1,:),x(2,:),'g')
```

This should plot two solutions in the x_1x_2-plane. (The first 4 commands together plot one solution; you can think of the first ImprovedEulerMethod as going forward in time and the second as going backward in time because the initial velocities are opposite.)

That's it for what you need to hand in. If you think the solution is too boring, you can try the following variation: Take the force to depend on time as well, say

$$\vec{F}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = -\frac{.1 \cdot \sin(t) + .9}{r^3}\begin{bmatrix} x \\ y \end{bmatrix}.$$

Rewrite this as a first order system and code the right hand side of the system into the function

```
function val=keplerVariation(t,x)
    ...
end
```

Then run the test commands

```
[t x]=ImprovedEulerMethod(@keplerVariation,0,100,100000,[.5 .5 .1 -.1]');
plot(x(1,:),x(2,:))
```

You can experiment with other IC and time intervals.