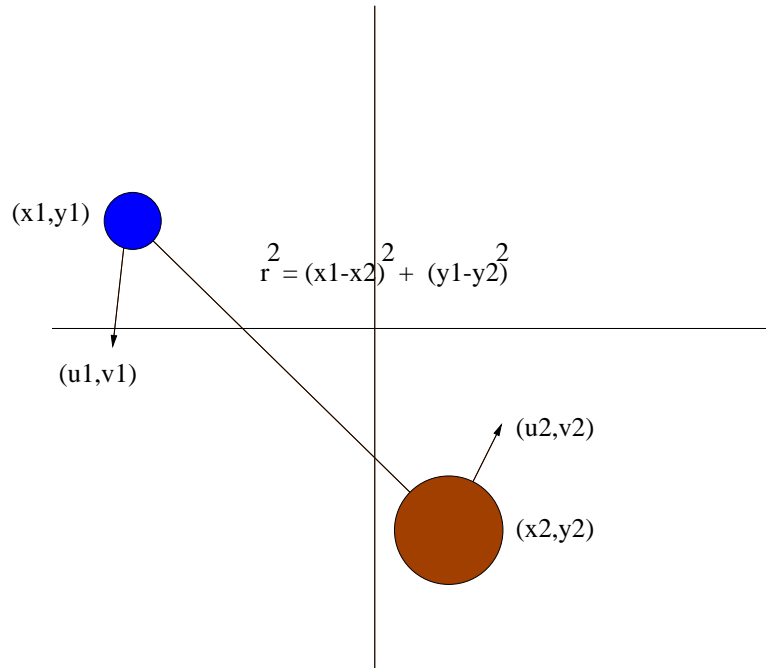


## Derivation of 2-body equations

Given bodies of mass  $m_1$  and  $m_2$  respectively, arranged in the following configuration



In the following,  $(x_1, y_1)$  is the position of mass  $m_1$ ,  $(x_2, y_2)$  is the position of mass  $m_2$ ,  $(u_1, v_1)$  is the velocity of mass  $m_1$ , and  $(u_2, v_2)$  is the velocity of mass  $m_2$ .

By Newton's law of Gravitation, the gravitational force between  $m_1$  and  $m_2$  is given by

$$\frac{Gm_1m_2}{r_{12}^2}$$

where

$$r_{12}^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

We therefore have the following set of differential equations:

$$m_1 x_1'' = \frac{Gm_1m_2}{r_{12}^3} (x_2 - x_1)$$

$$m_2 x_2'' = \frac{Gm_1m_2}{r_{12}^3} (x_1 - x_2)$$

$$m_1 y_1'' = \frac{Gm_1m_2}{r_{12}^3} (y_2 - y_1)$$

$$m_2 y_2'' = \frac{Gm_1m_2}{r_{12}^3} (y_1 - y_2)$$

In order to actually solve this numerically, in MatLab, we must write it as a first order system:

$$x_1' = u_1, u_1' = \frac{Gm_2}{r_{12}^3} (x_2 - x_1)$$

$$x_2' = u_2, u_2' = \frac{Gm_1}{r_{12}^3} (x_1 - x_2)$$

$$y_1' = v_1, v_1' = \frac{Gm_2}{r_{12}^3} (y_2 - y_1)$$

$$y_2' = v_2, v_2' = \frac{Gm_1}{r_{12}^3} (y_1 - y_2)$$

Letting  $x(1) = x_1, x(2) = u_1, x(3) = x_2, x(4) = u_2, x(5) = y_1, x(6) = v_1, x(7) = y_2, x(8) = v_2$ , we can write this as

$$\begin{aligned}x(1)' &= x(2) \\x(2)' &= \frac{Gm_2}{r_{12}^3}(x(3) - x(1)) \\x(3)' &= x(4) \\x(4)' &= \frac{Gm_1}{r_{12}^3}(x(1) - x(3)) \\x(5)' &= x(6) \\x(6)' &= \frac{Gm_2}{r_{12}^3}(x(7) - x(5)) \\x(7)' &= x(8) \\x(8)' &= \frac{Gm_1}{r_{12}^3}(x(5) - x(7))\end{aligned}$$

where

$$r_{12}^2 = (x(1) - x(3))^2 + (x(5) - x(7))^2$$

Now, we use the following Matlab script m-file (with the special case  $m_1 = m_2$ )

---

```
function xdot = twobody(t,x)

% these values work, caution when modifying ...
G = 2;
m1 = 2;
m2 = 2;

% declare and initialize 8x1 column storage
xdot = zeros(8,1);

x1 = x(1);
u1 = x(2);
x2 = x(3);
u2 = x(4);
y1 = x(5);
v1 = x(6);
y2 = x(7);
v2 = x(8);

r = sqrt( (x1-x2)^2 + (y1-y2)^2 );

xdot(1) = x(2);
xdot(2) = G*m2*(x2-x1)/r^3;
xdot(3) = x(4);
xdot(4) = G*m2*(x1-x2)/r^3;
xdot(5) = x(6);
xdot(6) = G*m1*(y2-y1)/r^3;
xdot(7) = x(8);
xdot(8) = G*m1*(y1-y2)/r^3;

% end of function
```

---

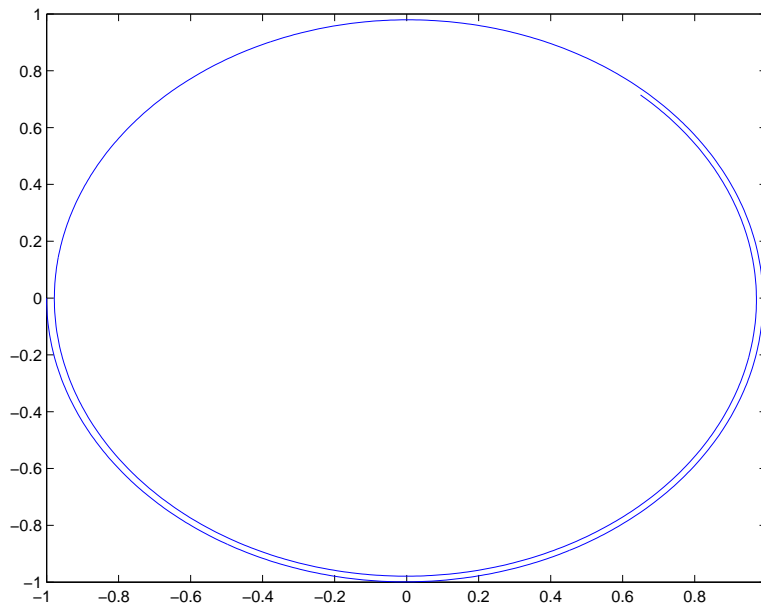
In Matlab, we use the routine 'ode45' to integrate the equations of motion. This gives fixed output (at  $dt=0.01$  intervals) on  $[0,10]$ . Initial conditions are  $x_1(0) = -1, u_1(0) = 0, x_2(0) = 1, u_2(0) = 0, y_1(0) = 0, v_1(0) = -1, y_2(0) = 0, v_2(0) = 1$ :

---

```
[t,x]=ode45('twobody',[0:0.01:10],[-1,0,1,0,0,-1,0,1]);  
plot(x(:,1),x(:,5));
```

---

This gives the following figure

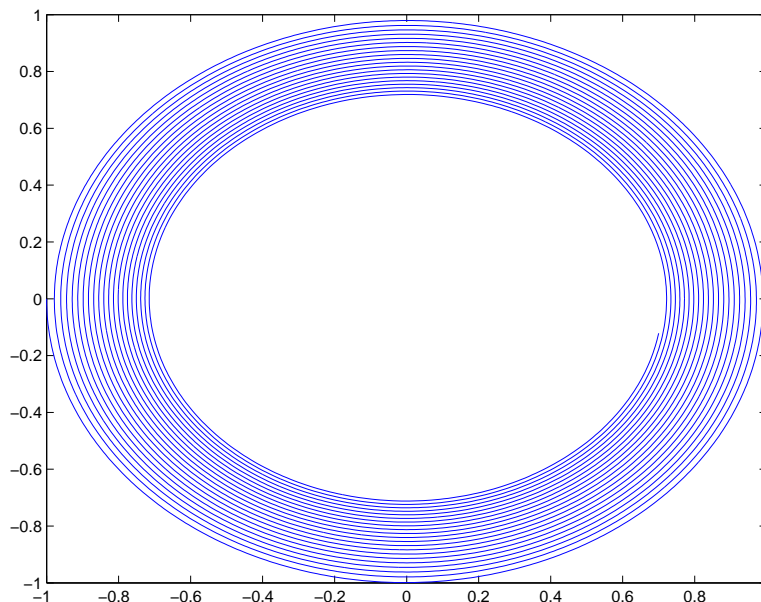


If we integrate out further in time

```
[t,x]=ode45('twobody',[0:0.1:100],[-1,0,1,0,0,-1,0,1]);  
plot(x(:,1),x(:,5));
```

---

This gives the following figure



There is clearly room for improvement here. As a further note, it is known that explicit methods tend to be “dissipative”, that is, even in a system where energy is conserved, it is lost when integrating forwards in time. Implicit methods tend to be better. Designing numerical schemes which conserve energy is a challenging research area.

1. improve the accuracy (it should be closed orbit!) Hint: try 'help odeset'
2. generalize to  $m_1 \neq m_2$
3. generalize equations to 3-body problem
4. solve the 3-body system in the case that  $m_1 = m_2 \gg m_3$ .