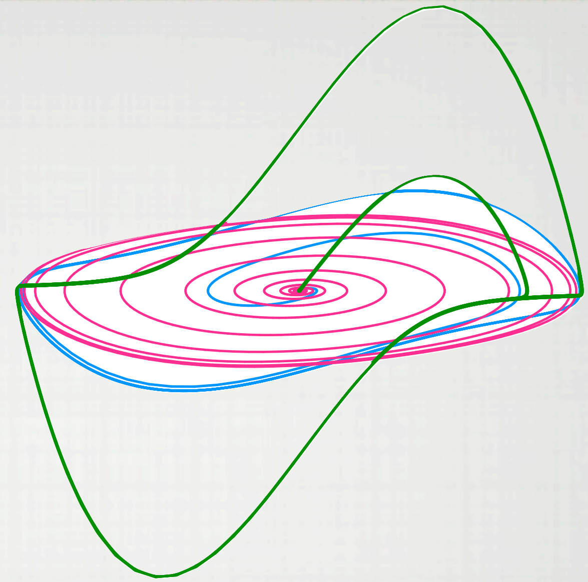


PHY-4810
COMPUTATIONAL PHYSICS

LECTURE 8: ORDINARY DIFFERENTIAL EQUATIONS

EXAMPLE OF ODES

Inhomogeneous first-order linear constant coefficient ordinary differential equation

$$\frac{du}{dx} = cu + x^2$$

Homogeneous second-order linear ordinary differential equation

$$\frac{d^2u}{dx^2} - x\frac{du}{dx} + u = 0$$

First-order nonlinear ordinary differential equation

$$\frac{du}{dx} = u^2 + 1$$

Second-order nonlinear ordinary differential equation

$$L\frac{d^2u}{dx^2} + g\sin u = 0$$

Homogeneous second-order linear constant coefficient ordinary differential equation

$$\frac{d^2u}{dx^2} + \omega^2 u = 0$$

EXAMPLES OF ODES

- ☐ Newton's Second Law
- ☐ Hamilton's equations in classical mechanics
- ☐ Radioactive decay in nuclear physics
- ☐ Newton's law of cooling in thermodynamics
- ☐ The wave equation
- ☐ Maxwell's equations in electromagnetism
- ☐ The heat equation in thermodynamics
- ☐ Laplace's equation
- ☐ Poisson's equation
- ☐ Einstein's field equation in general relativity
- ☐ The Schrödinger equation in quantum mechanics
- ☐ The geodesic equation
- ☐ The Navier–Stokes equations in fluid dynamics
- ☐ The Cauchy–Riemann equations
- ☐ The Poisson–Boltzmann equation in molecular dynamics
- ☐ The shallow water equations
- ☐ Universal differential equation
- ☐ The Lorenz equations whose solutions exhibit chaotic flow.
- ☐ Verhulst equation – biological population growth
- ☐ von Bertalanffy model – biological individual growth
- ☐ Lotka–Volterra equations – biological population dynamics
- ☐ Replicator dynamics – theoretical biology
- ☐ The Black–Scholes PDE
- ☐ Exogenous growth model
- ☐ Malthusian growth model
- ☐ The Vidale-Wolfe advertising model

GENERAL COMMENTS

$$\frac{dy}{dt} = g^3(t)y(t) \qquad \text{(linear)}$$

$$\frac{dy}{dt} = \lambda y(t) - \lambda^2 y^2(t) \qquad \text{(nonlinear)}$$

- The general solution of a first-order differential equation always contains one arbitrary constant.
- A general solution of a second-order differential equation contains two such constants, and so forth.
- For any specific problem, these constants are fixed by the initial conditions.
- Regardless of how powerful a computer you use, the mathematical fact still remains and you must know the initial conditions in order to solve the problem.

From 1 equation of order-N to N equations of order-1

REDUCTION TO A SYSTEM OF 1ST ORDER ODES

THE OBJECTIVE IS TO TRANSFORM OUR EQUATION INTO:

$$\boxed{\begin{aligned} \frac{dy^{(0)}}{dt} &= f^{(0)}(t, y^{(i)}) \\ \frac{dy^{(1)}}{dt} &= f^{(1)}(t, y^{(i)}) \\ &\ddots \\ \frac{dy^{(N-1)}}{dt} &= f^{(N-1)}(t, y^{(i)}) \end{aligned}}$$

REDUCTION: EXAMPLE OF NEWTON'S LAW

- **Newton's law:**

$$\boxed{\frac{d^2x}{dt^2} = \frac{1}{m}F\left(t, \frac{dx}{dt}, x\right)}$$

- **Define $y^{(0)}$ and $y^{(1)}$**

$$\boxed{y^{(0)}(t) \stackrel{\text{def}}{=} x(t) \qquad y^{(1)}(t) \stackrel{\text{def}}{=} \frac{dx}{dt} = \frac{dy^{(0)}}{dt}}$$

- **The second-order ODE is now two simultaneous, first-order ODEs:**

$$\boxed{\frac{dy^{(0)}}{dt} = y^{(1)}(t) \qquad \frac{dy^{(1)}}{dt} = \frac{1}{m}F(t, y^{(0)}, y^{(1)})}$$

CANONICAL FORM

$$\boxed{f^{(0)} = y^{(1)}(t) \qquad\qquad f^{(1)} = \frac{1}{m}F(t, y^{(0)}, y^{(1)})}$$

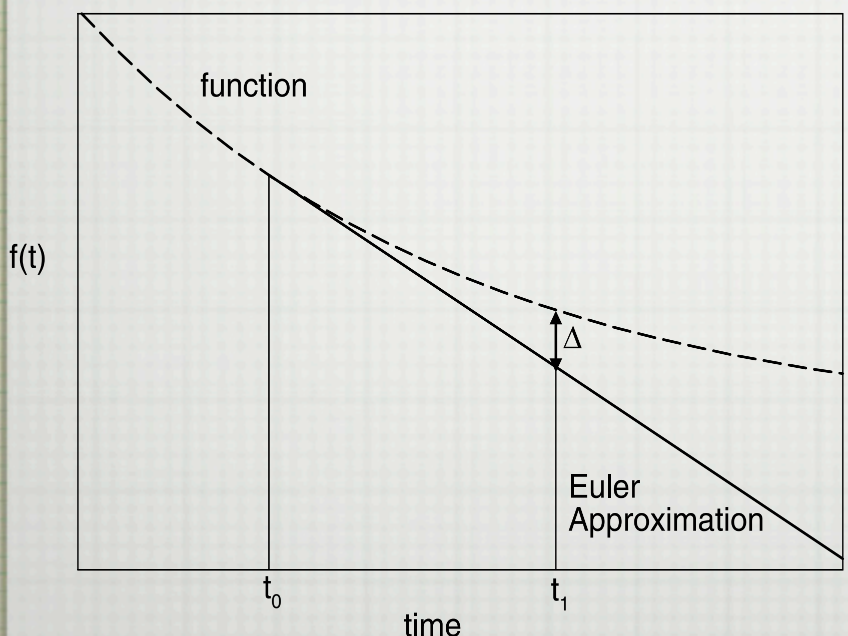
To be able to solve, we also need knowledge of initial conditions:

$y^{(1)}(0)$ and $y^{(0)}(0)$

ALGORITHM #1: EULER RULE

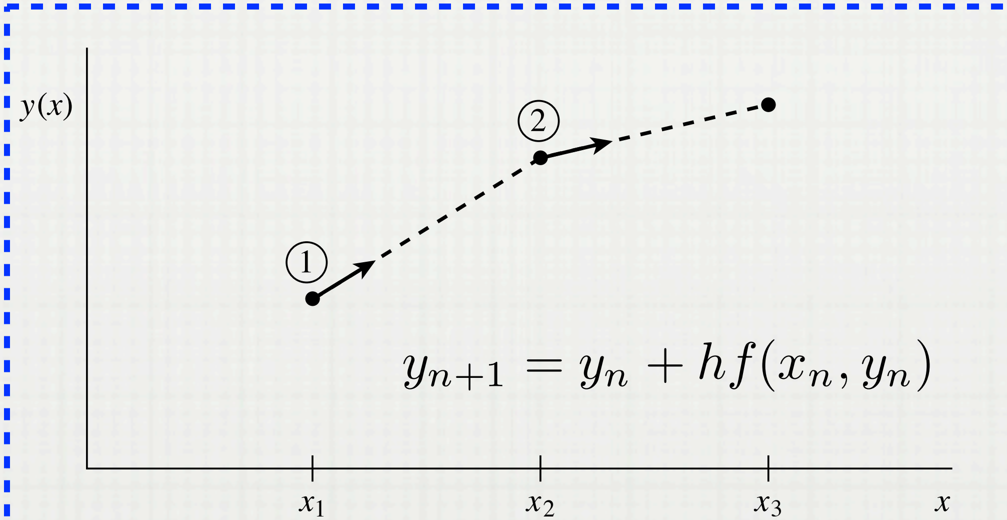
$$\frac{d\mathbf{y}(t)}{dt} \simeq \frac{\mathbf{y}(t_{n+1}) - \mathbf{y}(t_n)}{h} = \mathbf{f}(t, \mathbf{y})$$

$$\mathbf{y}_{n+1} \simeq \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n)$$



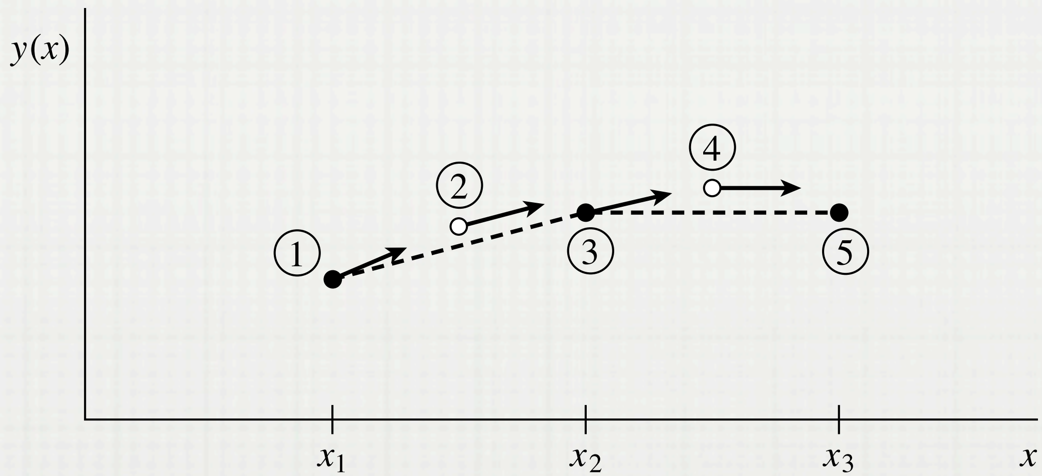
- **This simple algorithm requires very small *h* values to obtain precision**
- **However, using small values for *h* increases the number of steps and the accumulation of the round- off errors, which may lead to <u>instability</u>.**
- **<u>Problem:</u> We are using the derivative at the previous point ("following the tangent")**
- **A simple Taylor series analysis indicates the problem at first order**

PROBLEM WITH EULER



- **The derivative at the starting point of each interval is extrapolated to find the next function value.**
- **Euler advances a solution from x_n to $x_{n+1} \equiv x_n + h$.**
- **The formula is non-symmetric: It advances the solution through an interval h, but uses derivative information only at the beginning of that interval.**
- **Conclusion: the formula is neither stable nor accurate!**

ALGORITHM #2: RUNGE-KUTTA



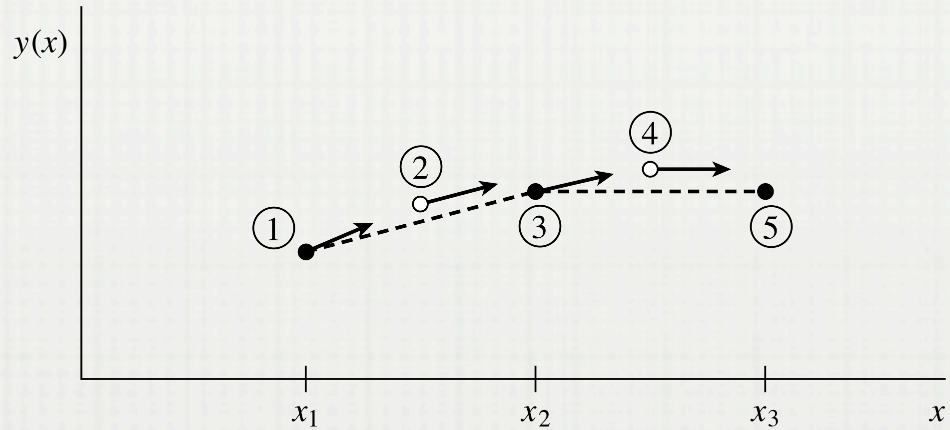
Take a "trial" step to the midpoint of the interval.

Then use the value of both *x* and xi at that midpoint to compute the "real" step across the whole interval.

RK2:

***second-order Runge-Kutta* or *midpoint* method**

MATHEMATICALLY



$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right)$$

$$y_{n+1} = y_n + k_2 + O(h^3)$$

RK2:

***SECOND-ORDER RUNGE-KUTTA* OR *MIDPOINT* METHOD**

FOURTH-ORDER RUNGE-KUTTA

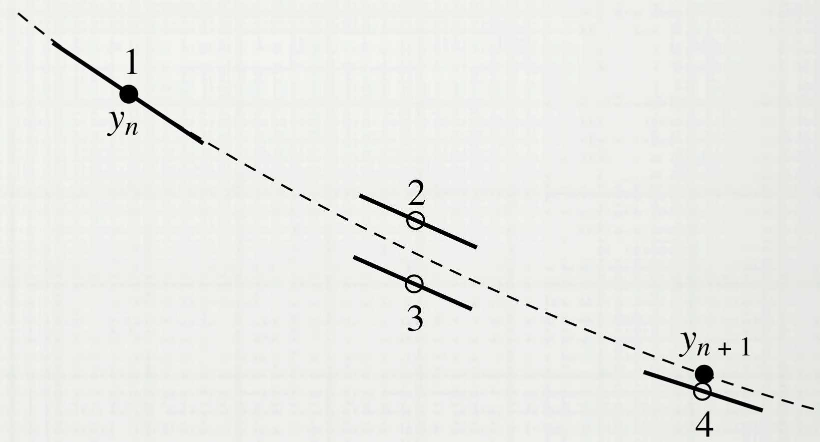
$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$

$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$

$$k_4 = hf(x_n + h, y_n + k_3)$$

$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$



- ☐ **The fourth-order Runge-Kutta method requires four evaluations of the right- hand side per step h.**
- ☐ **This will be (most of the time!) far superior to the midpoint method.**

RK4: FOURTH-ORDER RUNGE-KUTTA **METHOD**

BEYOND FOURTH ORDER

- **Adaptive Stepsize Control for Runge-Kutta (R45)**
  - **Usually the purpose of this adaptive stepsize control is to achieve some predetermined accuracy in the solution with minimum computational effort.**
  - **Many small steps should tiptoe through treacherous terrain, while a few great strides should speed through smooth uninteresting countryside.**
  - **The resulting gains in efficiency are not mere tens of percents or factors of two; they can sometimes be factors of ten, a hundred, or more.**
  - **Sometimes accuracy may be demanded not directly in the solution itself, but in some related conserved quantity that can be monitored.**
- **Higher order RK (See NumRec for eight order method)**

HOW TO USE RK? USE LIBRARIES

```
#include <iostream>
#include "nr3.h"
#include "rk4.h"

void derivs(const Doub x, VecDoub_I & y, VecDoub_O & dydx){
   Doub mu=1.0;
   dydx[0]=y[1];
   dydx[1]=(mu*(1.0-y[0]*y[0])*y[1]-y[0]);
}

int main (int argc, char * const argv[]) {
   VecDoub y(2),dydx(2);
   Doub x,xmin,xmax,kmax=100000,h=0.0001;
   VecDoub yout(2);
   int k;
   xmin=-10; xmax=10000.;
   y[0]=3.1;
   y[1]=0;
   derivs(xmin,y,dydx);
   for(k=0;k<kmax;k++){
       x=xmin+k*h;
       rk4(y, dydx,  x, h, yout, derivs);
       cout << x << " " << yout[0] << " " << yout[1] << endl;
       y[0]=yout[0];
       y[1]=yout[1];
       derivs(x,y,dydx);
   }
}
```

EXAMPLES

EXAMPLE 1: A WARM-UP EXAMPLE

VAN DER POL EQUATION

$$y'' - \mu(1 - y^2)y' + y = 0$$

$$f^{(0)} = y$$

$$f^{(1)} = y'$$

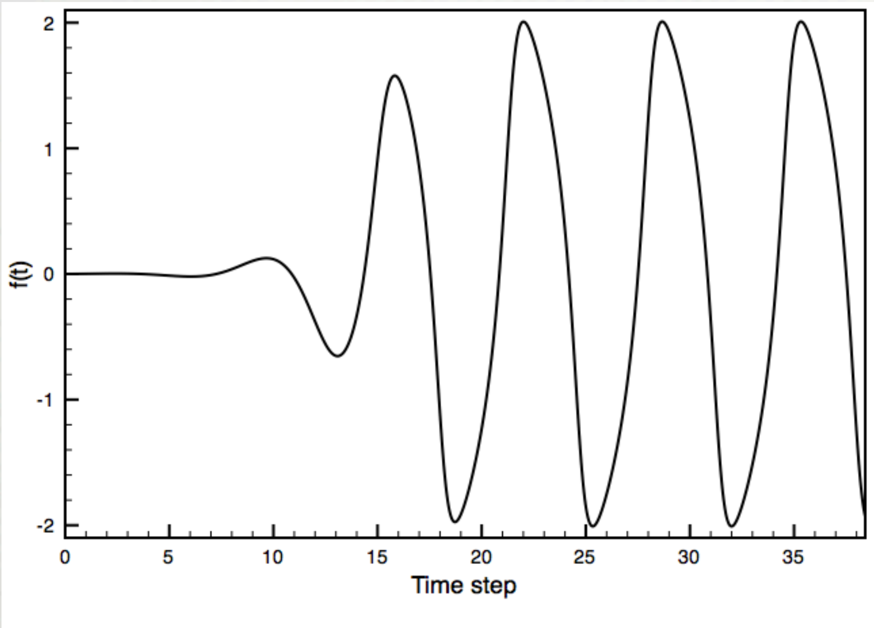
$$\frac{df^{(0)}}{dt} = f^{(1)}$$

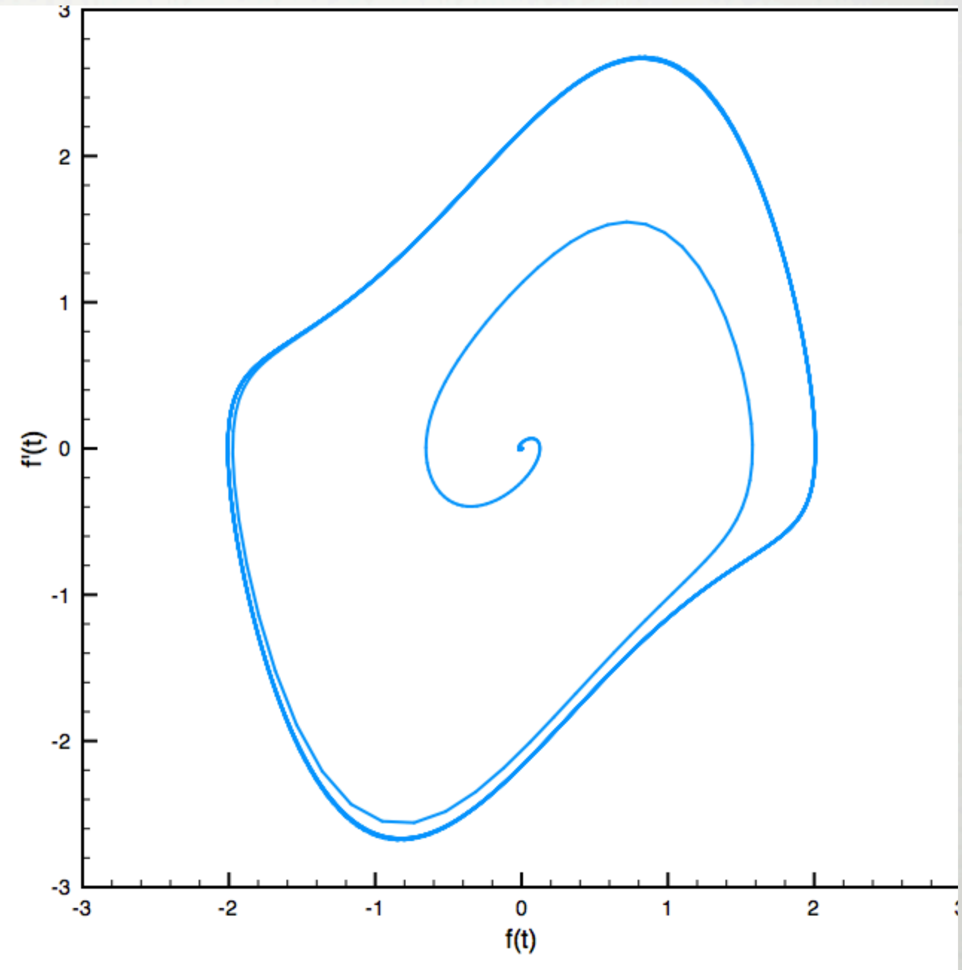
$$\frac{df^{(1)}}{dt} = \mu(1 - (f^{(0)})^2)f^{(1)} - f^{(0)}$$

```
TO USE IN C++:

DYDX[0]=Y[1];
DYDX[1]=(MU*(1.0-Y[0]*Y[0])*Y[1]-Y[0]);
```

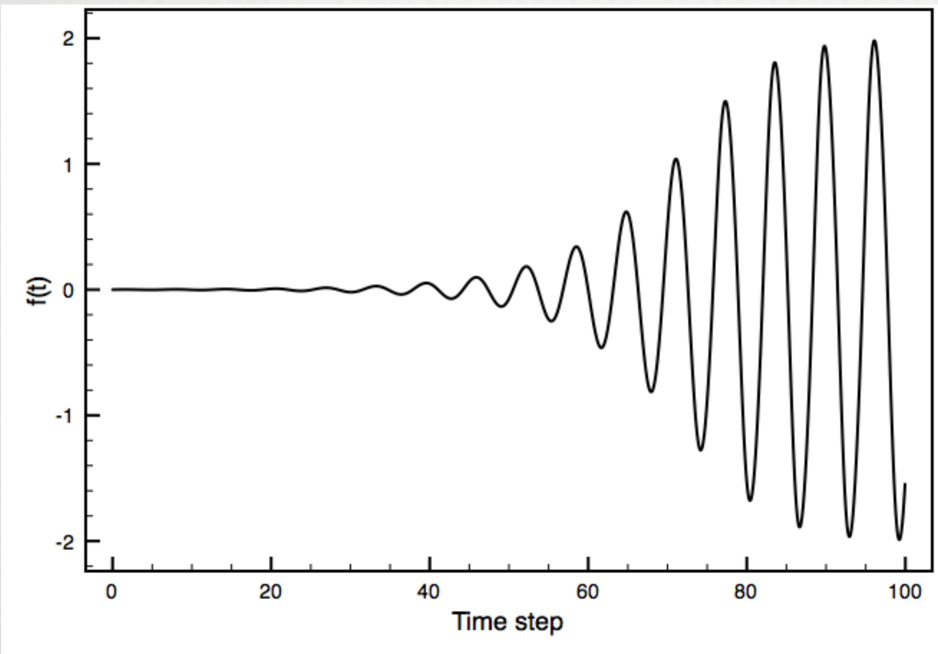
It is an equation describing self-sustaining oscillations in which energy is fed into small oscillations and removed from large oscillations. This equation arises in the study of circuits containing vacuum tubes.

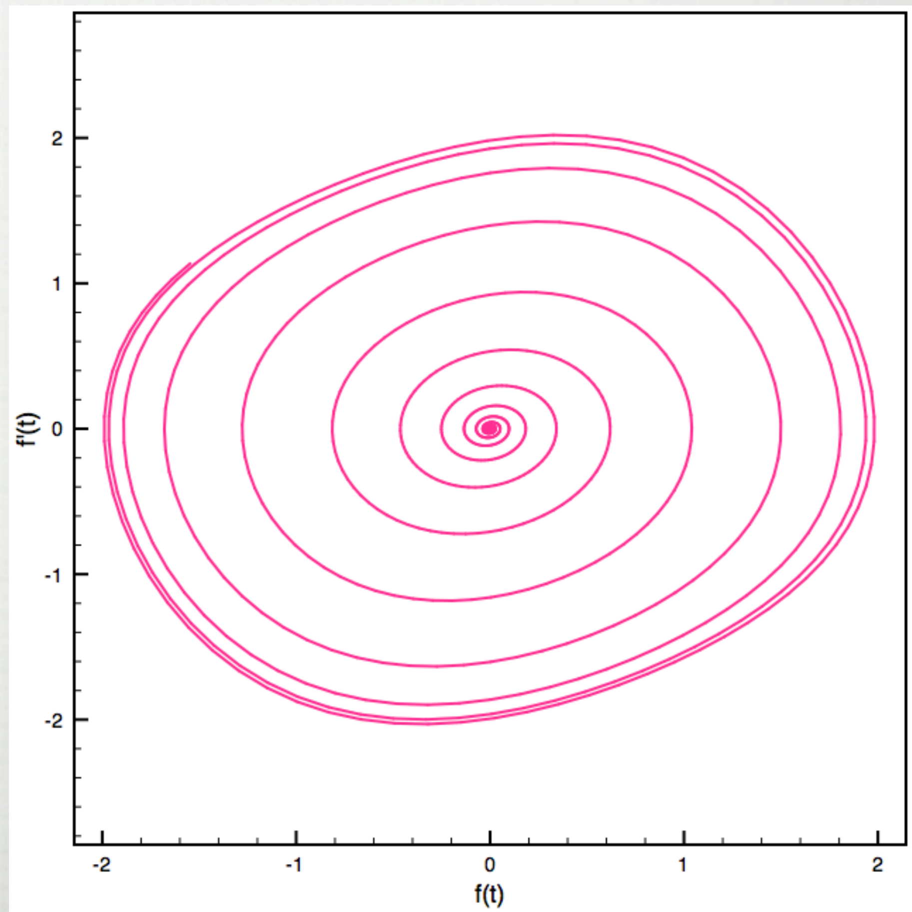
SOLUTION (1) MU=1



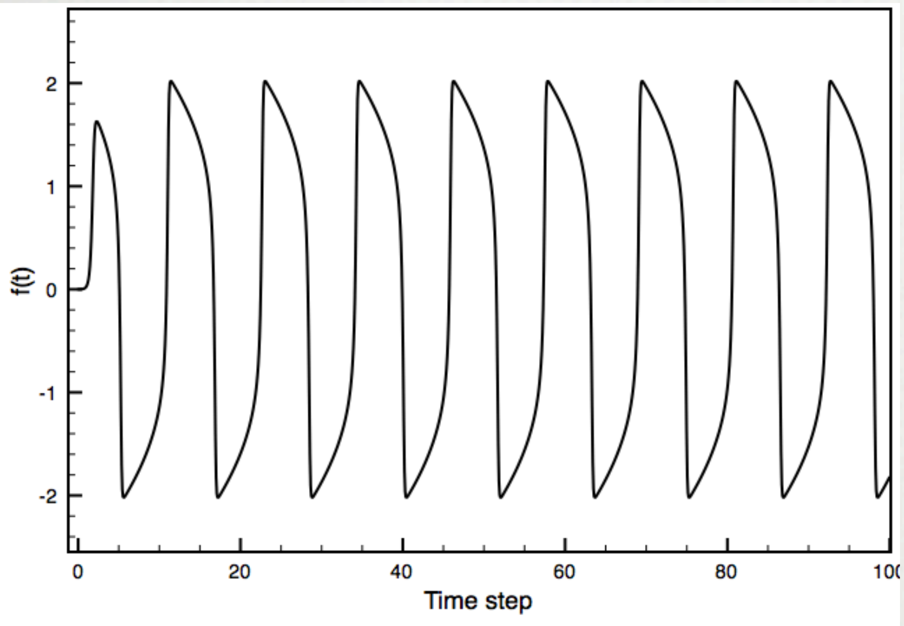


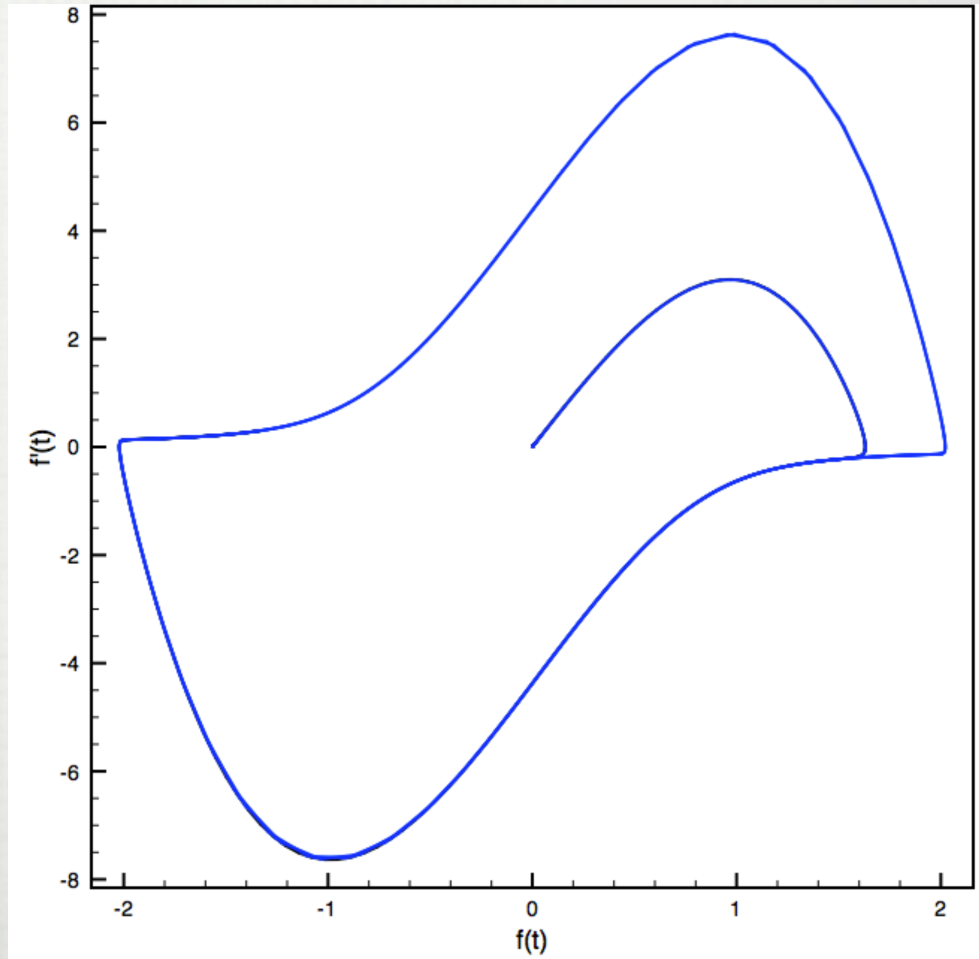
SOLUTION (2): MU=0.2



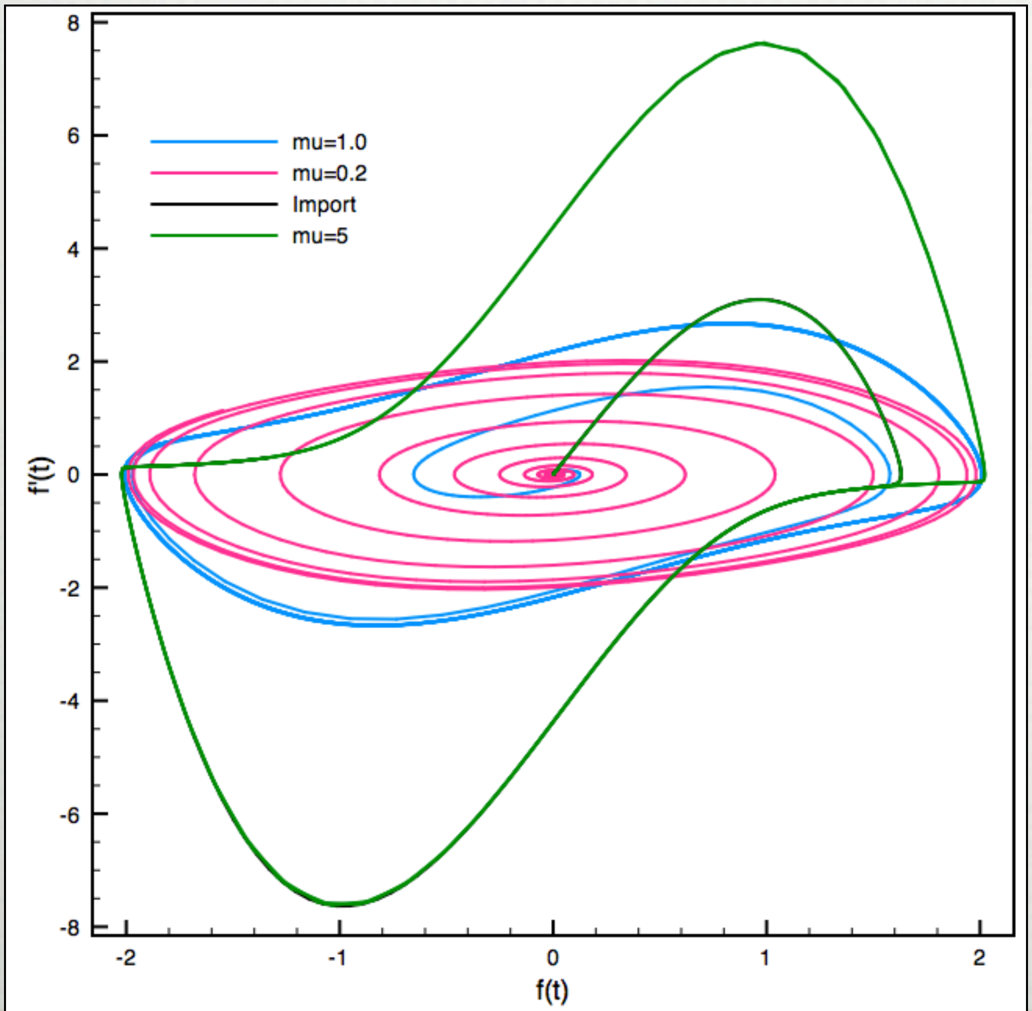


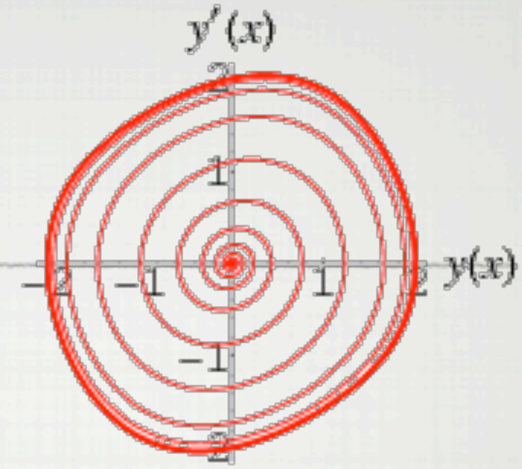
SOLUTION (3): MU=5

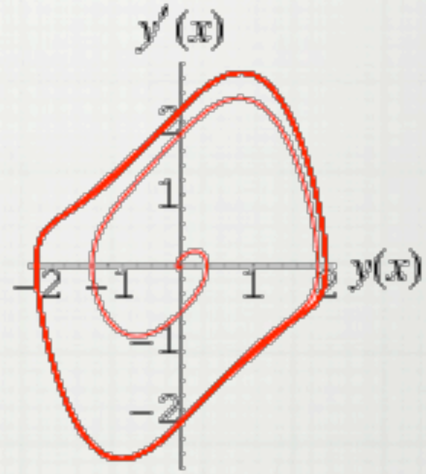


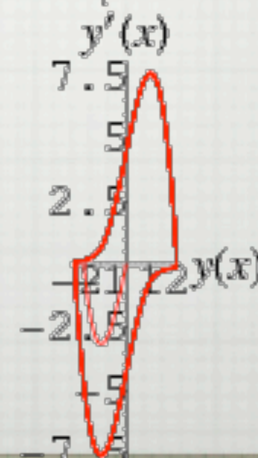


COMPARING


mu=1.0
mu=0.2
Import
mu=5
f'(t)
f(t)
8
6
4
2
0
-2
-4
-6
-8
-2
-1
0
1
2

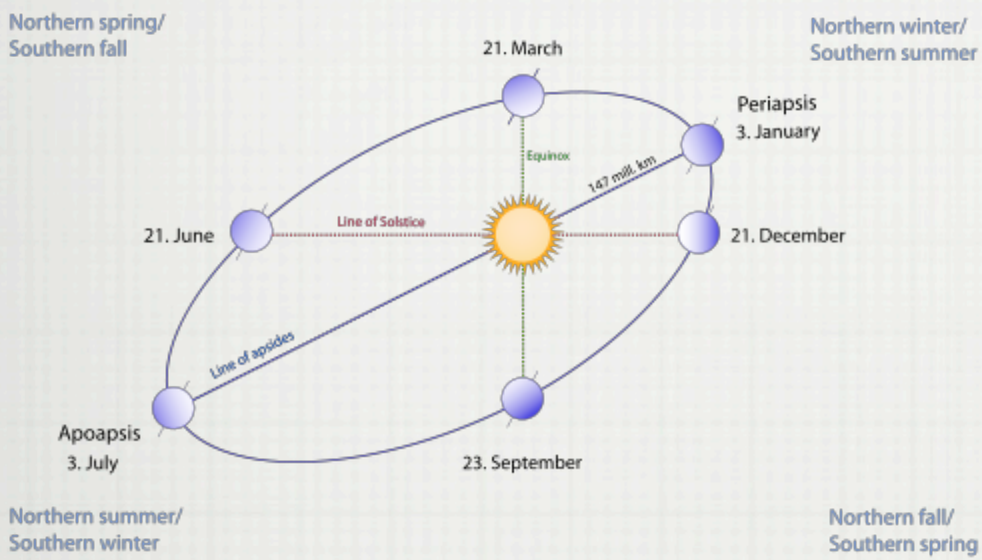

$y'(x)$
$y(x)$

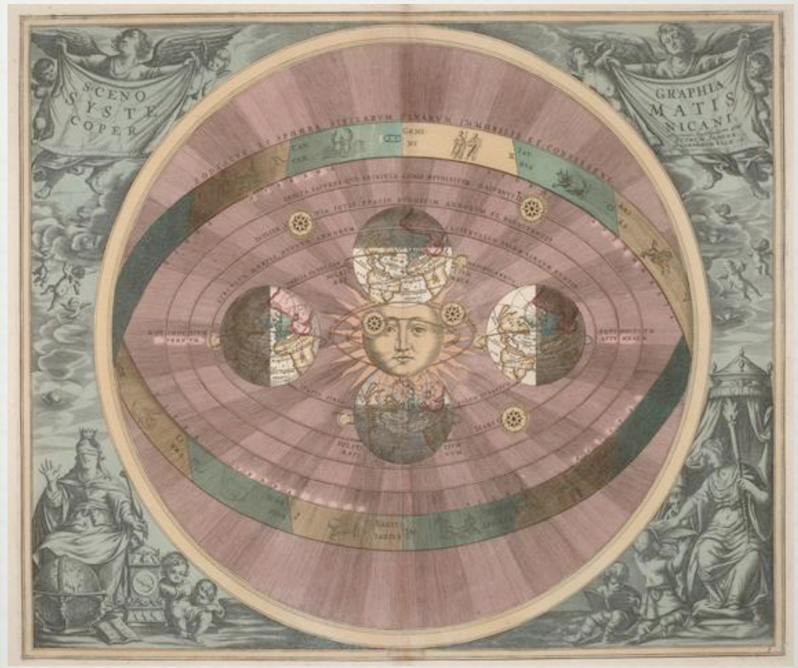

$y'(x)$
$y(x)$


$y'(x)$
$y(x)$
7.5
2.5

EXAMPLE 2: PLANETARY MOTIONS

PLANETARY MOTION (EARTH)



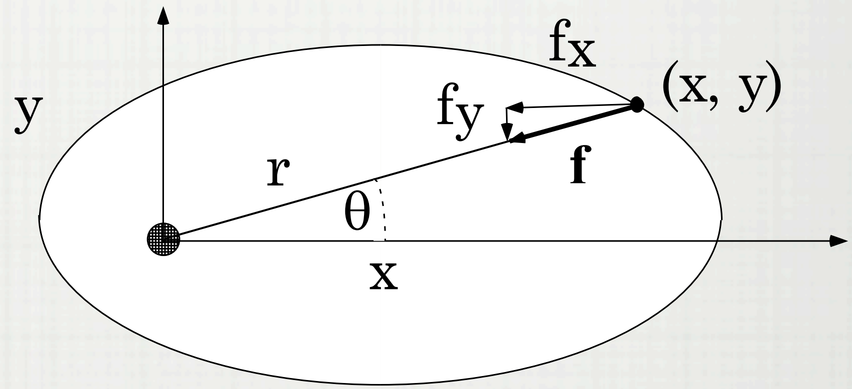


- Aphelion 152,097,701 km (1.016 AU)
- Perihelion 147,098,074 km (0.983 AU)
- Semi-major axis 149,597,887 km
- Semi-minor axis 149,576,999 km

EQUATIONS

$$f = -\frac{GmM}{r^2}$$ **(GRAV. ATTRACTION)**

$$\mathbf{f} = m\mathbf{a} = m\frac{d^2\mathbf{x}}{dt^2}$$ **(NEWTON)**



$$f_x = f\cos\theta = f\frac{x}{r} \qquad f_y = f\sin\theta = f\frac{y}{r} \qquad (r = \sqrt{x^2 + y^2})$$

$$\boxed{\frac{d^2x}{dt^2} = -GM\frac{x}{r^3} \qquad \frac{d^2y}{dt^2} = -GM\frac{y}{r^3}}$$

REDUCTION TO CANONICAL FORM...

REDUCTION TO CANONICAL FORM

$$\frac{d^2x}{dt^2} = -GM\frac{x}{r^3} \qquad \frac{d^2y}{dt^2} = -GM\frac{y}{r^3}$$

$$f^{(0)} = x$$

$$f^{(1)} = x' = v_x$$

$$f^{(2)} = y$$

$$f^{(3)} = y' = v_y$$

$$\frac{df^{(0)}}{dt} = f^{(1)}$$

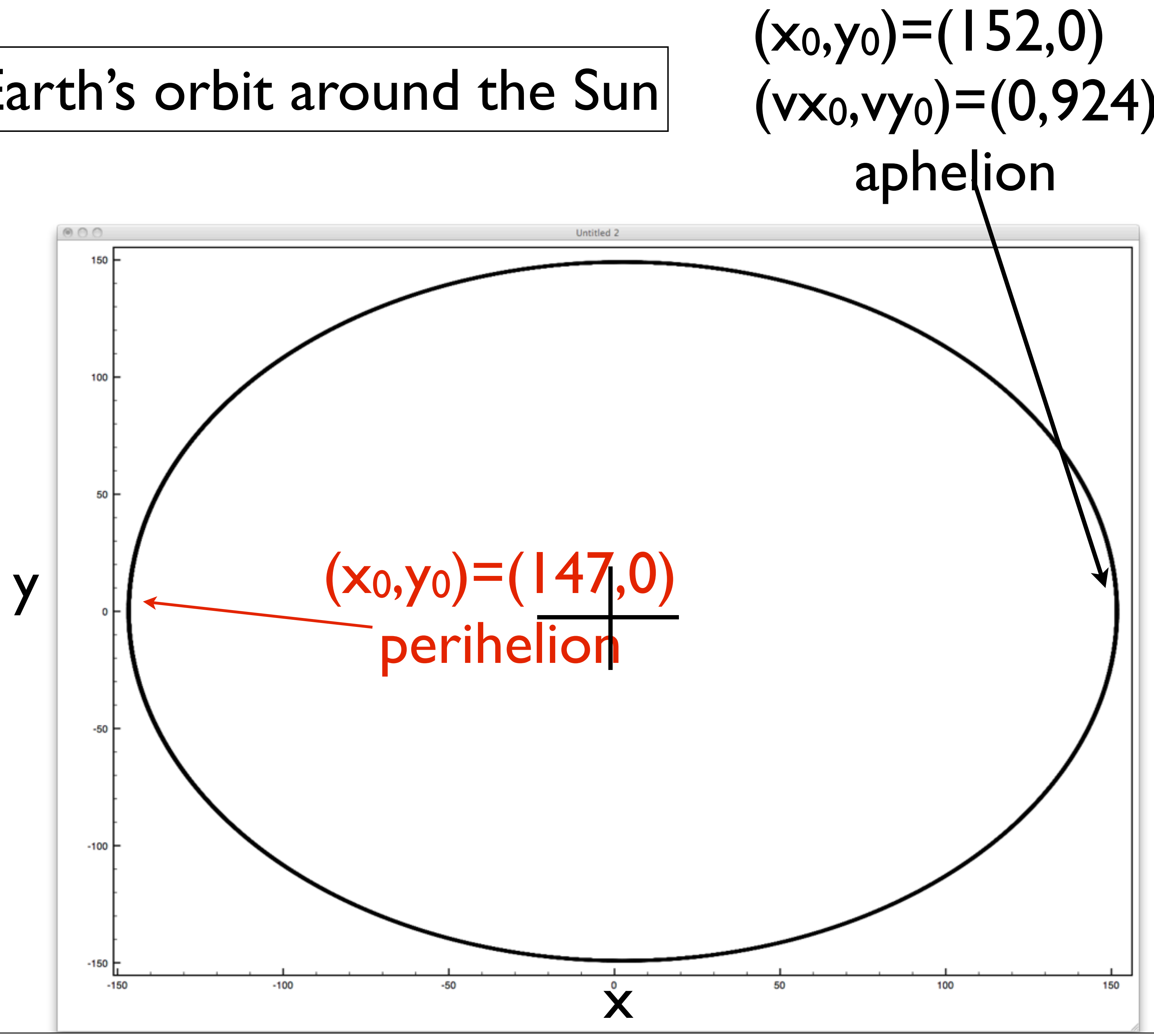
$$\frac{df^{(1)}}{dt} = -\frac{GMf^{(0)}}{r^3}$$

$$\frac{df^{(2)}}{dt} = f^{(3)}$$

$$\frac{df^{(3)}}{dt} = -\frac{GMf^{(2)}}{r^3}$$

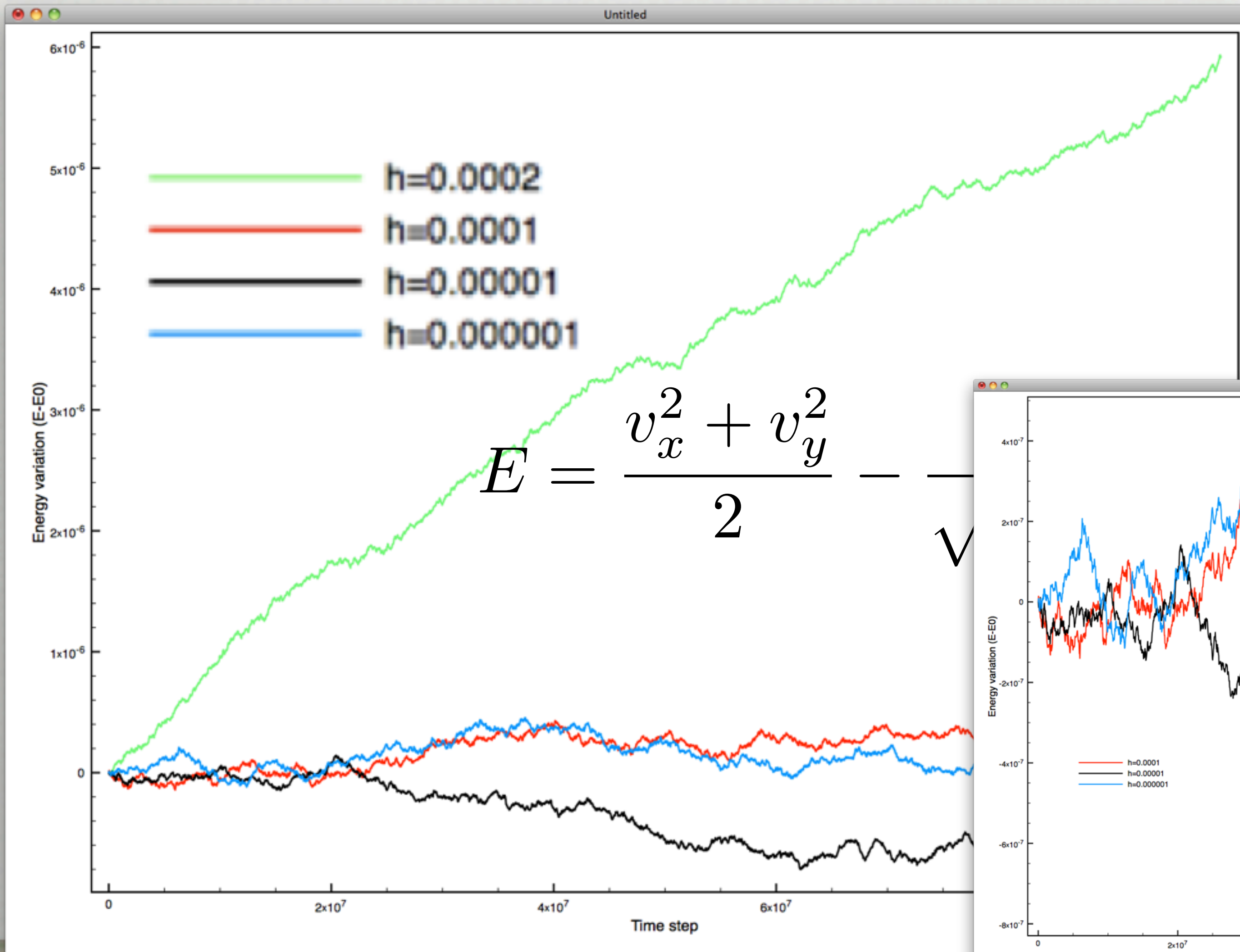
```
Doub gm=1.32167E8;

void derivsPLANET(const Doub x, VecDoub_I & y,
VecDoub_O & dydx){
   Doub val;
   val=pow(y[0]*y[0]+y[2]*y[2],-1.5);
   dydx[0]=y[1];
   dydx[1]=-gm*y[0]*val;
   dydx[2]=y[3];
   dydx[3]=-gm*y[2]*val;
}
```

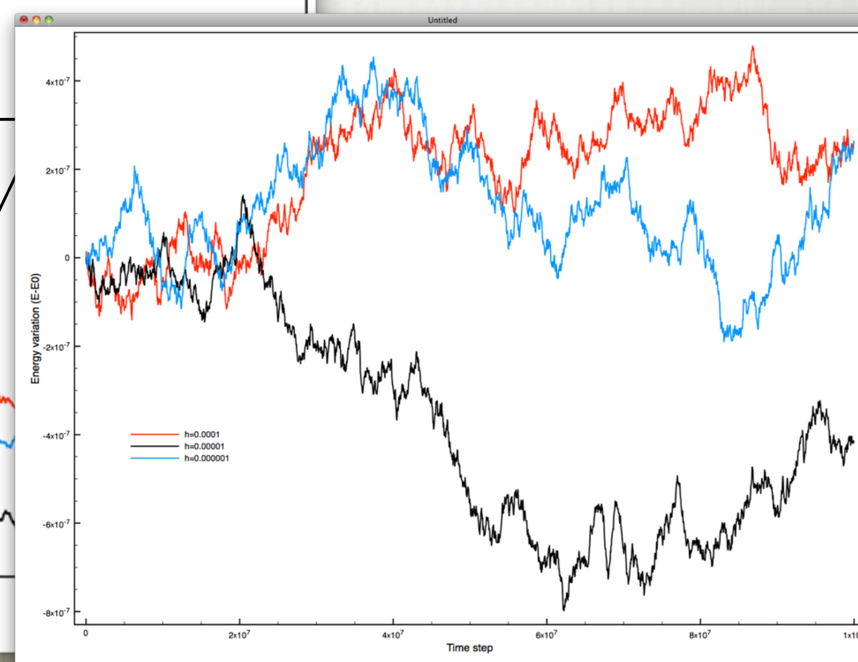
Earth's orbit around the Sun



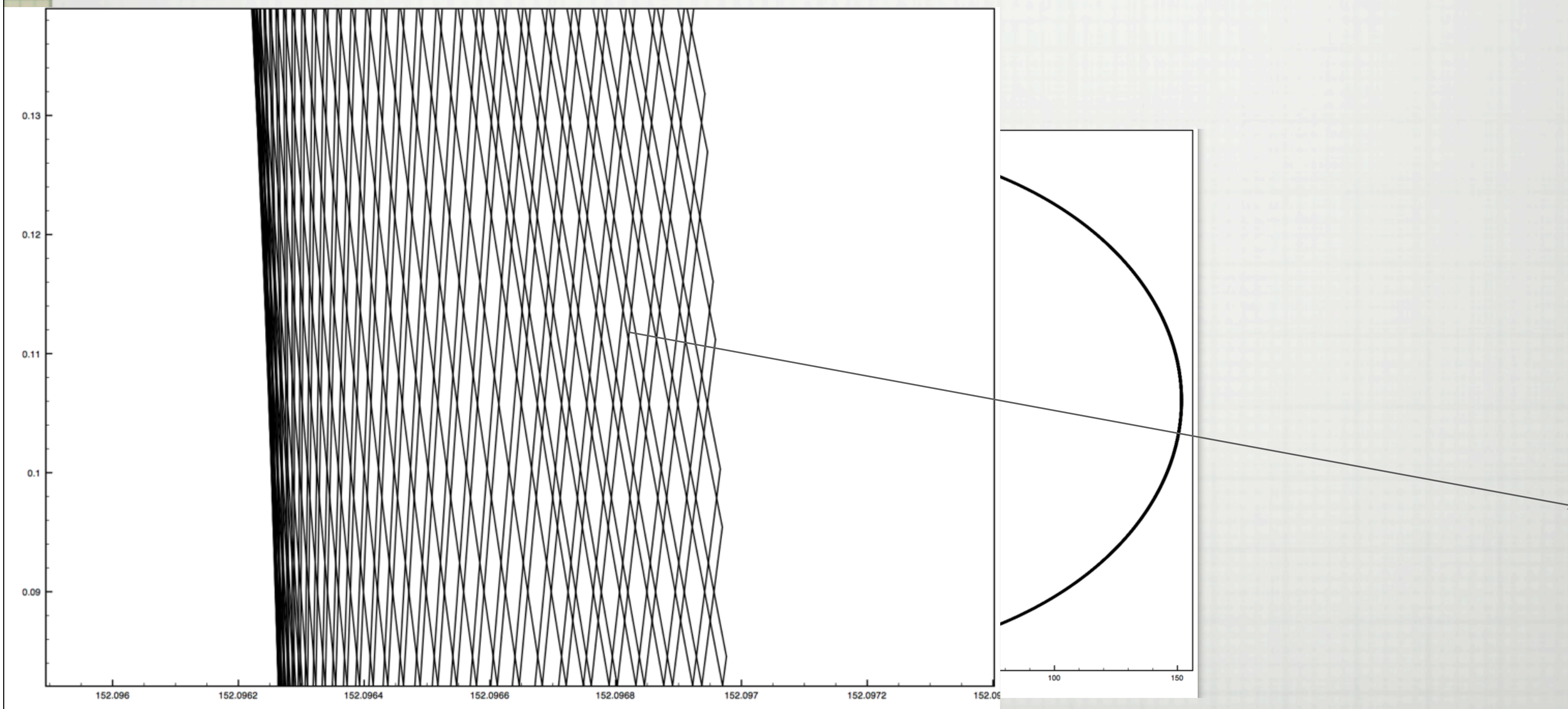
ENERGY

$$E = \frac{v_x^2 + v_y^2}{2} - \ldots$$





ENERGY FOR ORBIT (h=0.0001)?



FURTHER STUDIES

- ☐ **General relativity effects (precession of the perihelion)**
- ☐ **Many body effects**
- ☐ **Rings of Saturn**
- ☐ **...**

EXAMPLE 3: SIMPLE PENDULUM

CLASSICAL PENDULUM

- **Solution:**

$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\sin\theta$$

- **Small oscillations:**

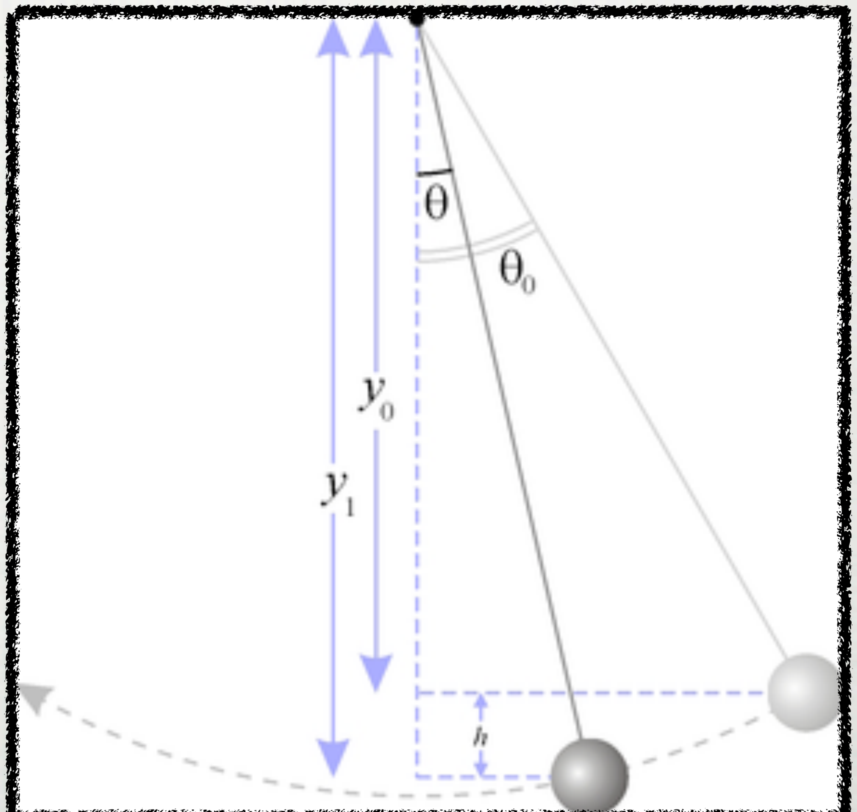
$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\theta$$

- **Damped:**

$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\sin\theta - \alpha v_\theta$$

- **With Drive:**

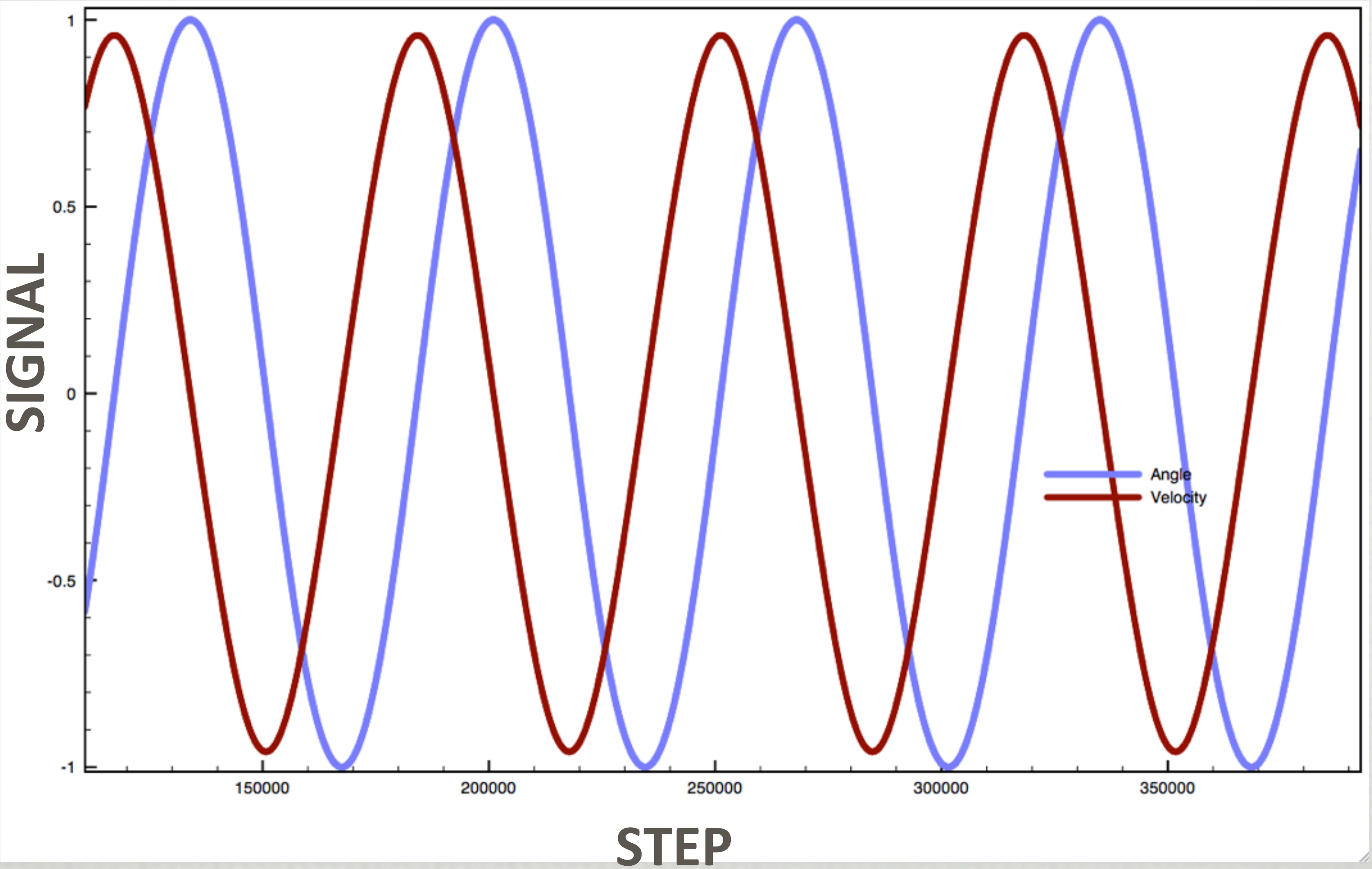
$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\sin\theta - \alpha v_\theta + b\cos\omega t$$



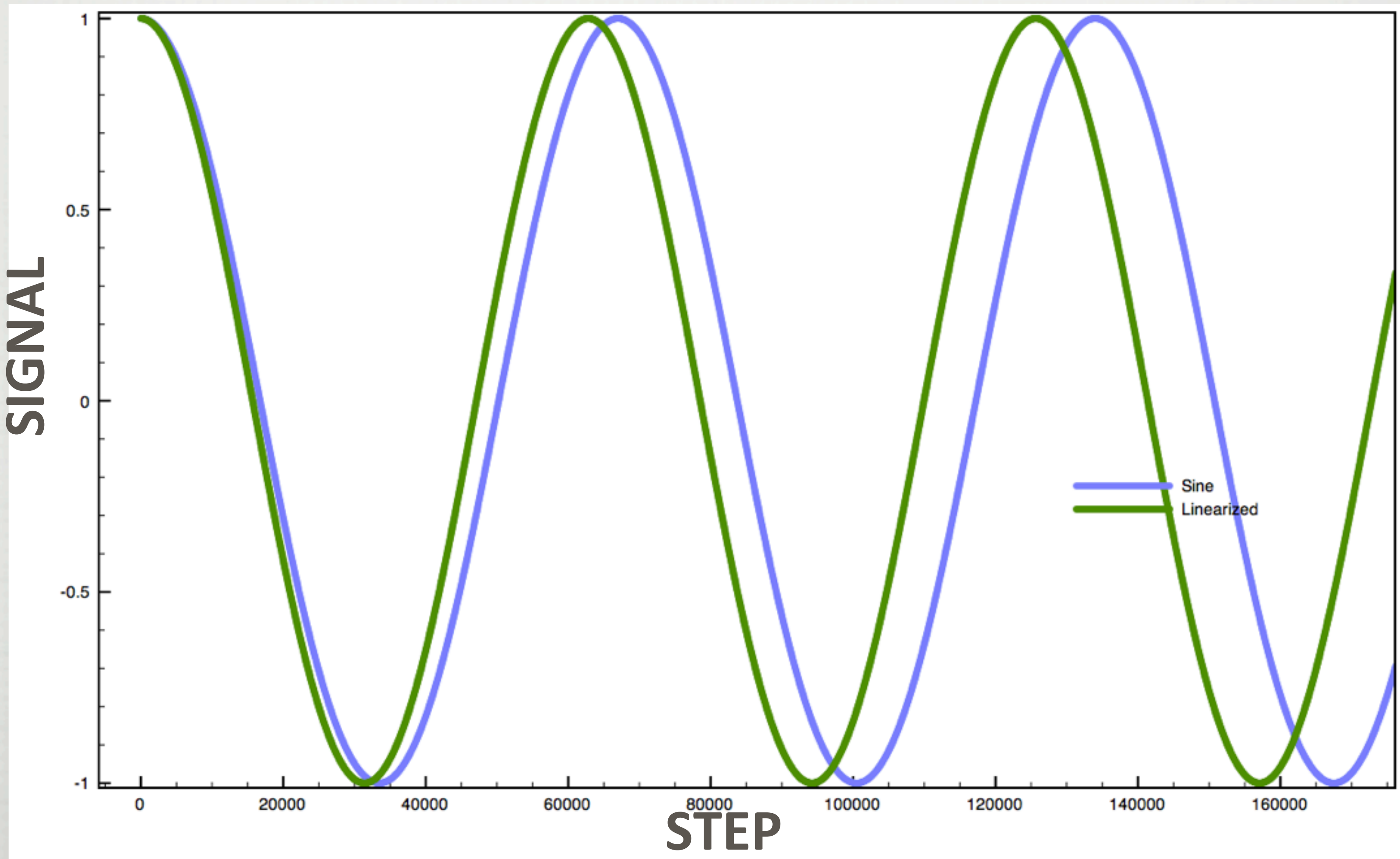
REDUCTION TO CANONICAL FORM

```
void pendulum(const Doub t, VecDoub_I & y, VecDoub_O & dydx){
  Doub omega=100;
  dydx[0]=y[1];
  dydx[1]=-sin(y[0]);
//   dydx[1]=-(y[0]); //small oscillations
//   DYDX[1]=-SIN(Y[0])-Y[1]; //DAMPING TERM

}
```

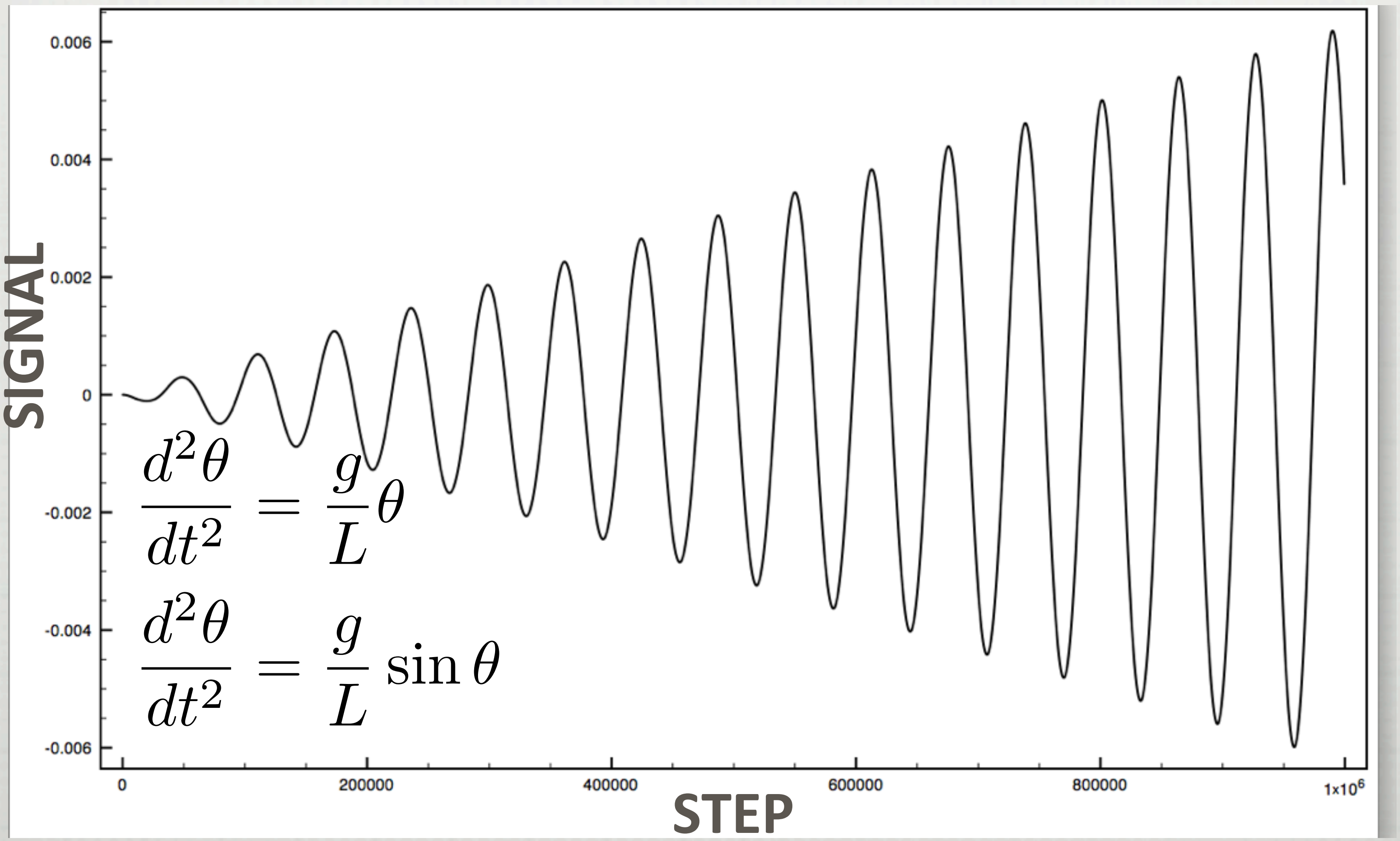
PENDULUM, STARTING AT $\theta = 1$



PENDULUM: SMALL OSCILLATIONS?



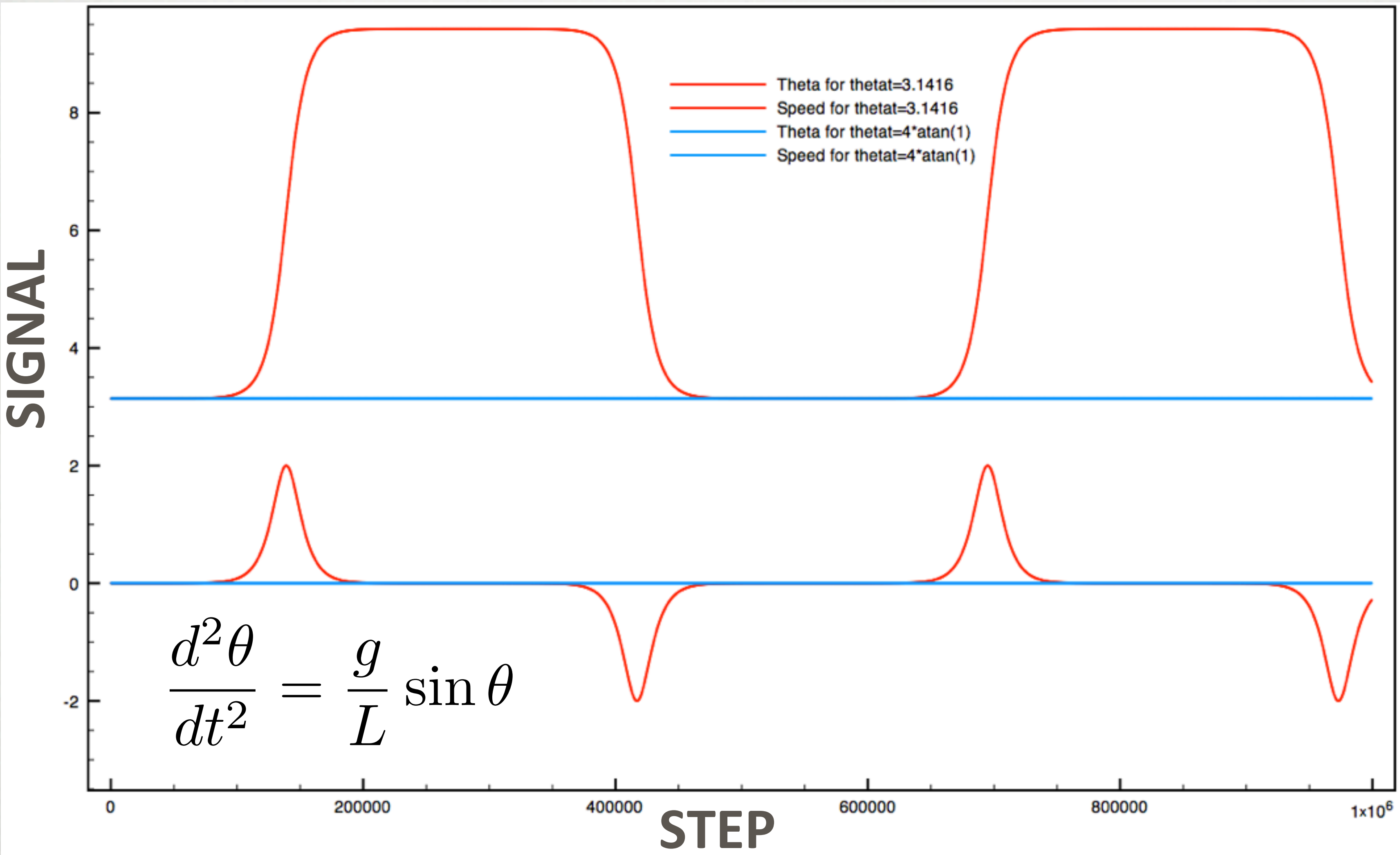
SMALL OSCILLATIONS: 0.1 LINEAR-SINE



$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\theta$$

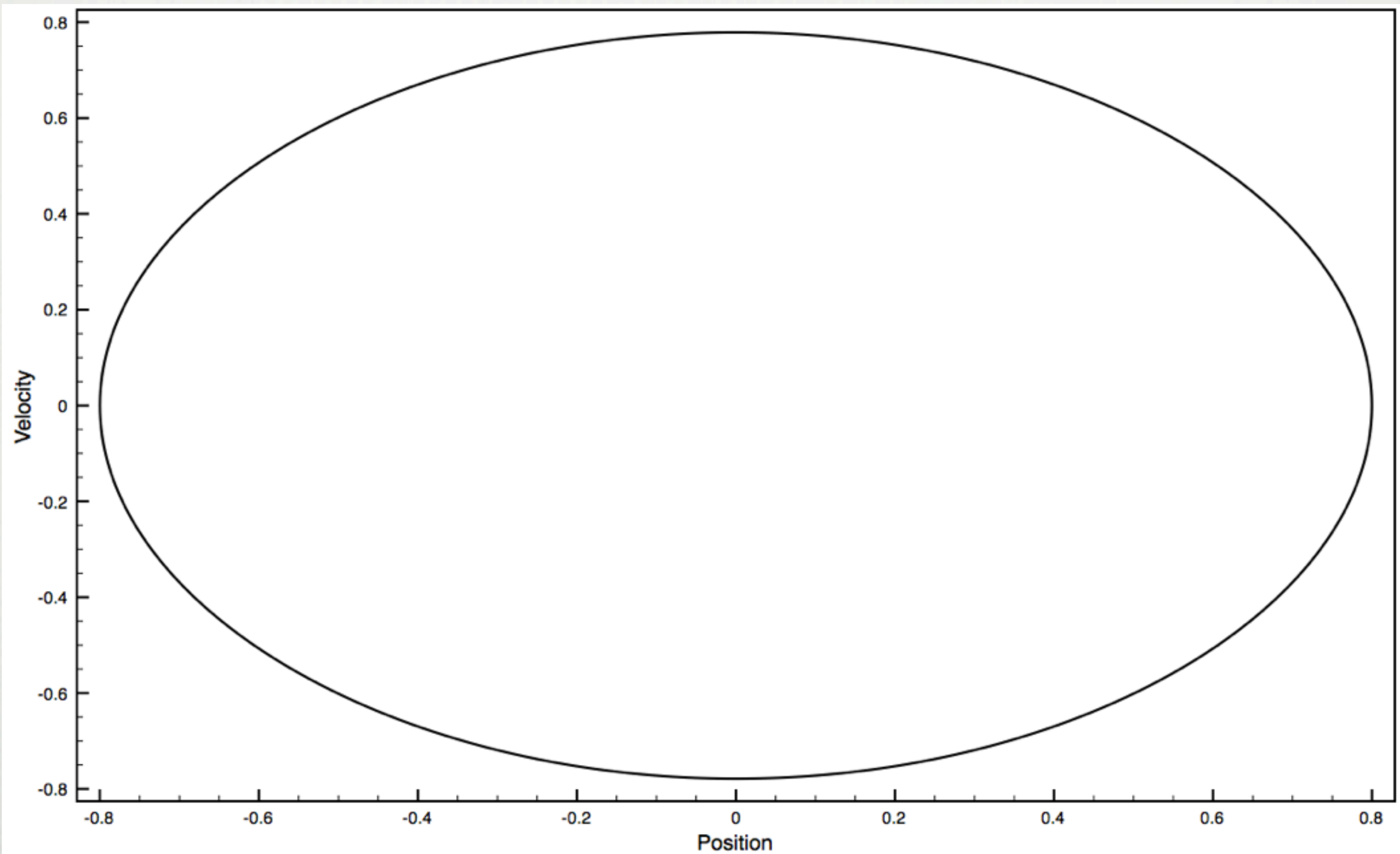
$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\sin\theta$$

PENDULUM: EXTREME CONDITIONS

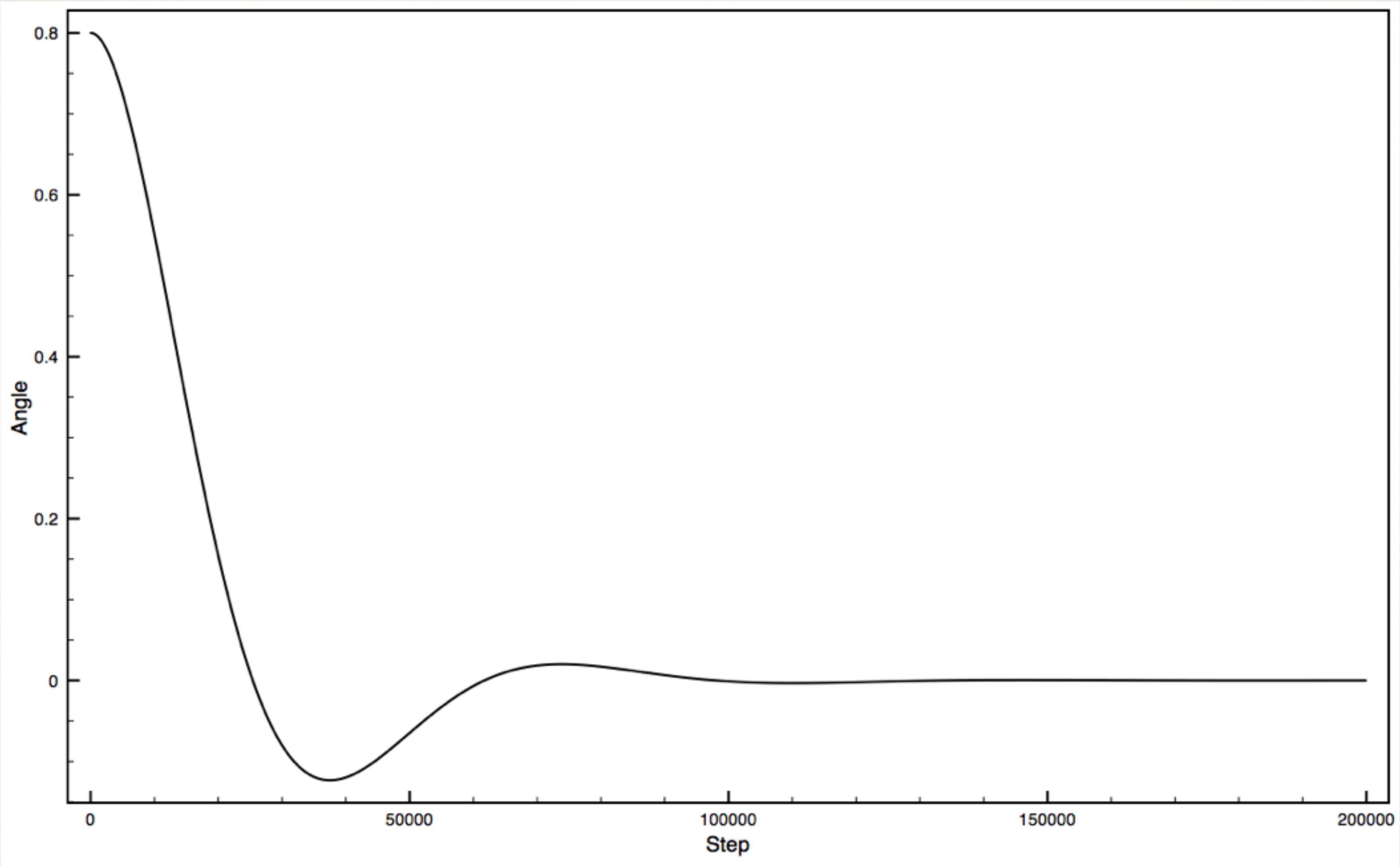


$$\frac{d^2\theta}{dt^2} = \frac{g}{L}\sin\theta$$

ENERGY CONSERVATION

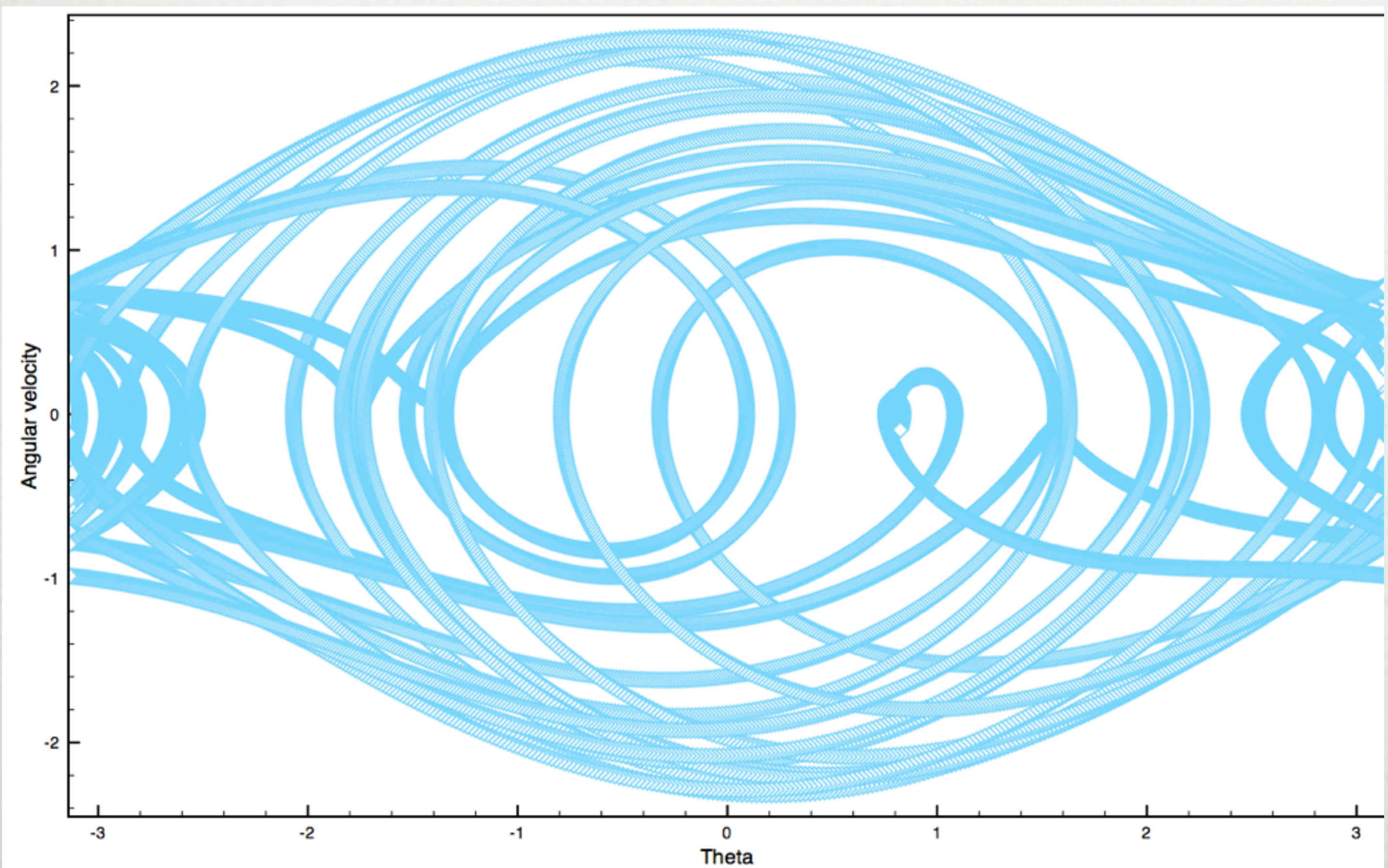


WITH DAMPING $\frac{d^2\theta}{dt^2} = \frac{g}{L}\sin\theta - \alpha v_\theta$



DAMPING+DRIVE → CHAOS?

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta - \alpha\frac{d\theta}{dt} + b\cos\omega t$$

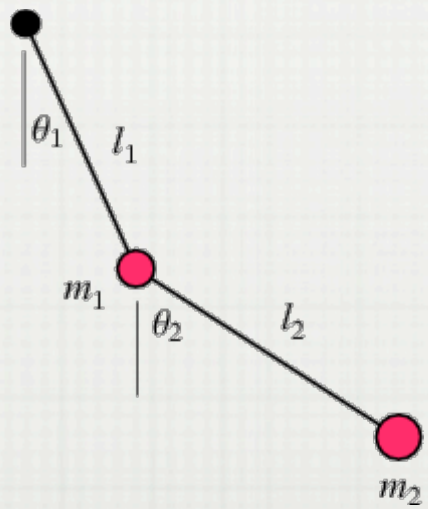


- ☐ **ω=2/3**
- ☐ ***g/L*=1**
- ☐ **α=0.5**
- ☐ ***b*=1.15**

EXAMPLE 4: DOUBLE PENDULUM

DOUBLE PENDULUM

(See: http://scienceworld.wolfram.com/physics/DoublePendulum.html)

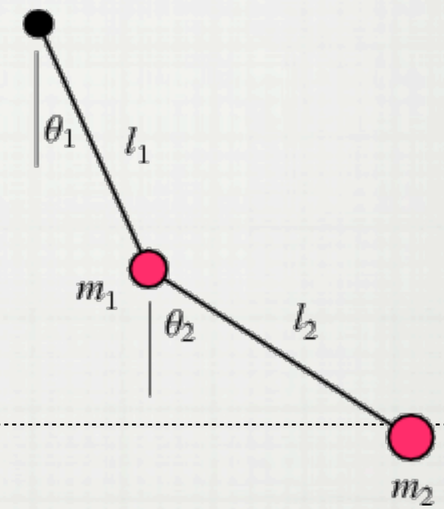


EQUATION OF MOTION IN LAGRANGIAN FORM:

$$\frac{\partial}{\partial t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_{1(2)}}\right) = \frac{\partial \mathcal{L}}{\partial \theta_{1(2)}}$$

DOUBLE PENDULUM: LAGRANGIAN

- See J.R. Taylor, Classical Mechanics



$$\frac{d\theta_1}{dt} = \omega_1$$

$$\frac{d\theta_2}{dt} = \omega_2$$

$$\frac{d\omega_1}{dt} = \frac{-sin(\theta_1 - \theta_1)(\omega_1^2 cos(\theta_1 - \theta_2) + \omega_2^2) + \frac{g}{L}(-2sin(\theta_1) + sin(\theta_2)cos(\theta_1 - \theta_2))}{2 - cos(\theta_1 - \theta_2)^2}$$

$$\frac{d\omega_2}{dt} = \frac{sin(\theta_1 - \theta_1)(2\omega_1^2 + \omega_2^2 cos(\theta_1 - \theta_2)) + 2\frac{g}{L}(-sin(\theta_1)cos(\theta_1 - \theta_2) - sin(\theta_2))}{2 - cos(\theta_1 - \theta_2)^2}$$

DOUBLE PENDULUM: CANONICAL FORM

$$\frac{d\theta_1}{dt} = \omega_1$$

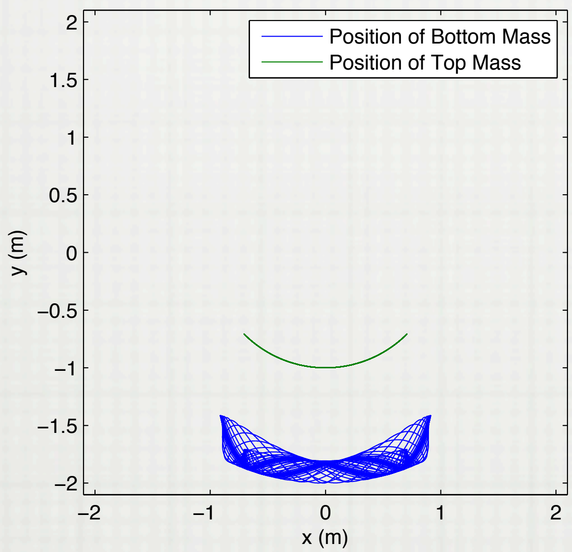
$$\frac{d\theta_2}{dt} = \omega_2$$

$$\frac{d\omega_1}{dt} = \frac{-sin(\theta_1 - \theta_1)(\omega_1^2 cos(\theta_1 - \theta_2) + \omega_2^2) + \frac{g}{L}(-2sin(\theta_1) + sin(\theta_2)cos(\theta_1 - \theta_2))}{2 - cos(\theta_1 - \theta_2)^2}$$

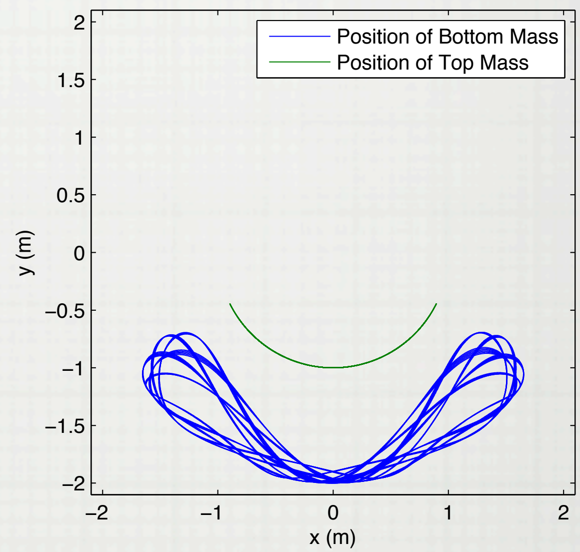
$$\frac{d\omega_2}{dt} = \frac{sin(\theta_1 - \theta_1)(2\omega_1^2 + \omega_2^2 cos(\theta_1 - \theta_2)) + 2\frac{g}{L}(-sin(\theta_1)cos(\theta_1 - \theta_2) - sin(\theta_2))}{2 - cos(\theta_1 - \theta_2)^2}$$

```
void derivs(const Doub x, VecDoub_I & y, VecDoub_O & dydx){
        Doub A,B,w,L=1,g=9.8;
        A=cos(y[0]-y[2]);
        B=sin(y[0]-y[2]);
        w=g/L;
        dydx[0]=y[1];    //y'[0]=d(theta1)/dt
        //y'[1]=y''[0]=equation of motion for theta1
        dydx[1]=(-B*(y[1]*y[1]*A+y[3]*y[3])+w*(-2*sin(y[0])+sin(y[2])*A))/(2-A*A);
        dydx[2]=y[3];    //y'[2]=d(theta2)/dt
        //y'[3]=y''[2]=equation of motion for theta2
        dydx[3]=(B*(2*y[1]*y[1]+y[3]*y[3]*A)+2*w*(-sin(y[2])+sin(y[0])*A))/(2-A*A);
}
```

DIFFERENT SOLUTIONS DEPENDING ON INITIAL CONDITIONS (I)

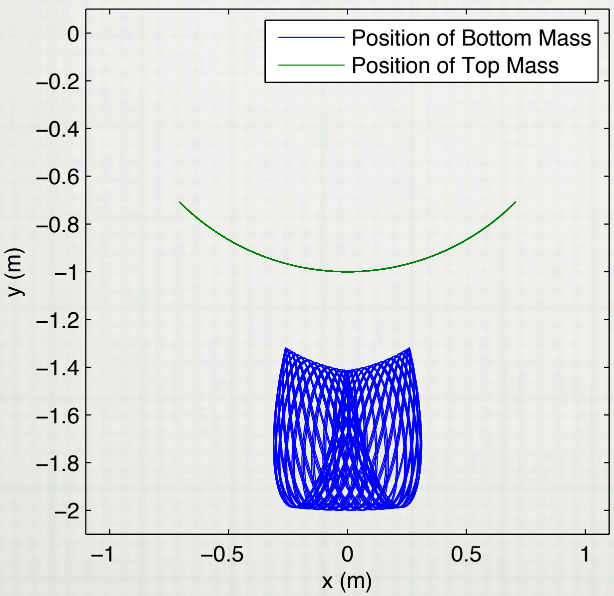


(a) Initial Conditions:
$\theta_1 = \frac{\pi}{4}, \theta_2 = 0, \frac{d\theta_1}{dt} = 0$, and $\frac{d\theta_2}{dt} = 0$.

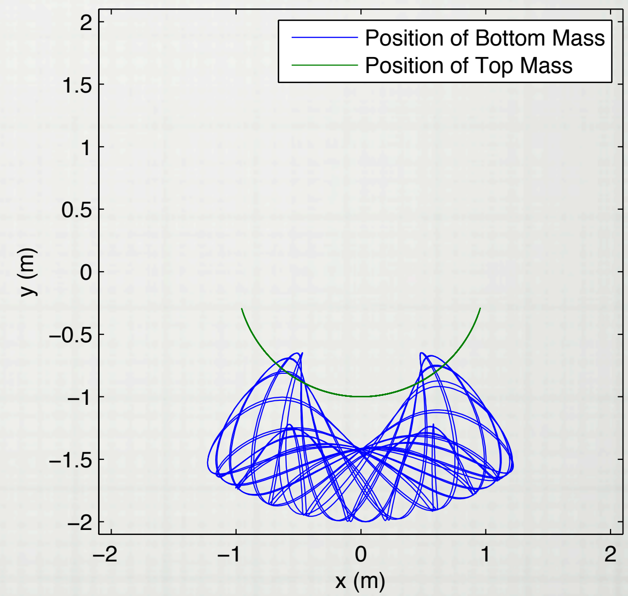


(b) Initial Conditions:
$\theta_1 = \frac{\pi}{4}, \theta_2 = 0, \frac{d\theta_1}{dt} = 2\frac{rad}{s}$, and $\frac{d\theta_2}{dt} = 2\frac{rad}{s}$.

DIFFERENT SOLUTIONS DEPENDING ON INITIAL CONDITIONS (II)

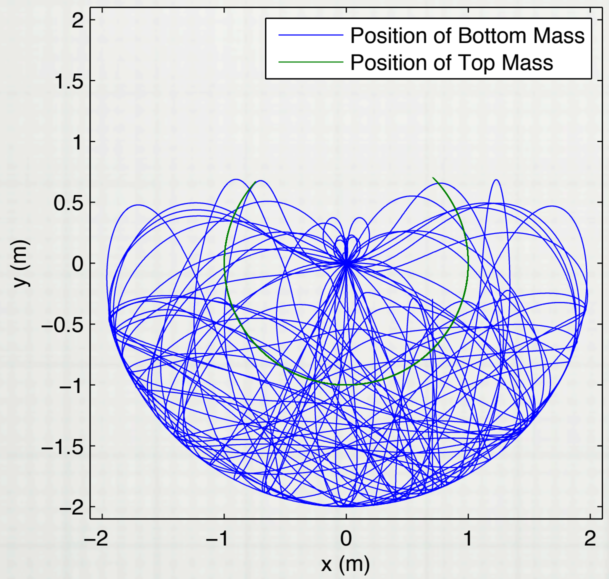


(c) Initial Conditions:
$\theta_1 = \frac{\pi}{4}, \theta_2 = -\frac{\pi}{4}, \frac{d\theta_1}{dt} = 0$, and $\frac{d\theta_2}{dt} = 0$.

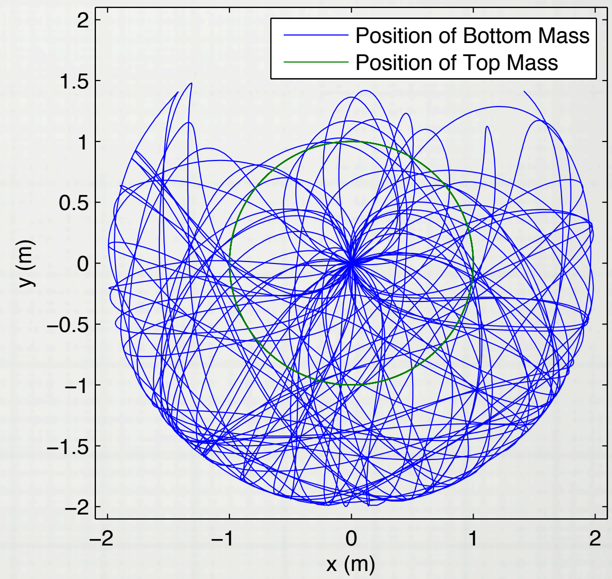


(d) Initial Conditions:
$\theta_1 = \frac{\pi}{4}, \theta_2 = -\frac{\pi}{4}, \frac{d\theta_1}{dt} = 2\frac{rad}{s}$, and $\frac{d\theta_2}{dt} = 2\frac{rad}{s}$.

DIFFERENT SOLUTIONS DEPENDING ON INITIAL CONDITIONS



(e) Initial Conditions:
$\theta_1 = \frac{3\pi}{4}, \theta_2 = 0, \frac{d\theta_1}{dt} = 0$, and $\frac{d\theta_2}{dt} = 0$.



(f) Initial Conditions:
$\theta_1 = \frac{3\pi}{4}, \theta_2 = \frac{3\pi}{4}, \frac{d\theta_1}{dt} = 0$, and $\frac{d\theta_2}{dt} = 0$.

SUMMARY

- ODEs can be solved efficiently on computers, using RK methods
- Reduction of ODEs into canonical form is the only step to take to translate physics into programming language
- Importance of initial conditions
- Importance of energy conservation! Choice of h.