Neuromechanics of Human Motion

Dynamical Systems and ODEs - A Primer

Joshua Cashaback, Ph.D.

Recap

1. download and run Python (or another language)
2. https://www.anaconda.com/distribution/
3. plot functions
4. Any questions about the course?

Lecture Objectives

1. Learn about dynamical systems
2. Understand and carry out numerical integration (Euler, RK4)
3. Program numerical integrators
4. convert nth order ODEs to n 1st order ODEs

Why do we need to know ODEs?

A lot of nature can be better understood using differential equations

Some models in this course using ODEs:

1. Nerve Models (Hodgkin-Huxley model)
2. Muscle Models (e.g., crossbridge & Hill-type)
3. Limb Dynamics
4. Control Models (LQG)
5. Adaptation Models (Multiple time-scales)

Behaviour of a Static System

Static System

a. An output that only depends on an input
b. e.g., massless spring (theoretical construct)
c. Hooke's Law ($F = -kx$)

Behaviour of a Static System

Static System

a. An output that only depends on an input

b. e.g., massless spring (theoretical construct)

c. Hooke's Law ($F = -kx$)

d. change force???

Behaviour of a Static System

Static System

a. An output that only depends on an input
b. e.g., massless spring (theoretical construct)
c. Hooke's Law ($F = -kx$)
d. change force = instantaneous length change

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

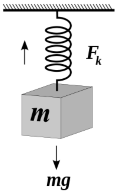
Spring length & mass position (depends on...)

→ acceleration of mass

→ sum of forces

→ input F, mg, F_k*

→ spring length



Acceleration of the mass *depends* on its position making this a dynamical system

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

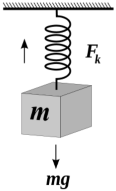
Spring length & mass position (depends on...)

$\rightarrow$ acceleration of mass

$\rightarrow$ sum of forces

$\rightarrow$ input F, mg, F_k*

$\rightarrow$ spring length



Acceleration of the mass *depends* on its position making this a dynamical system

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

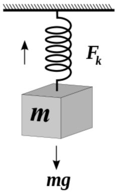
Spring length & mass position (depends on...)

$\rightarrow$ acceleration of mass

$\rightarrow$ sum of forces

$\rightarrow$ input F, mg, F_k*

$\rightarrow$ spring length



Acceleration of the mass *depends* on its position making this a dynamical system

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

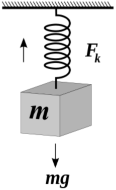
Spring length & mass position (depends on...)

$\rightarrow$ acceleration of mass

$\rightarrow$ sum of forces

$\rightarrow$ input F, mg, F_k*

$\rightarrow$ spring length



Acceleration of the mass *depends* on its position making this a dynamical system

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

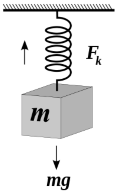
Spring length & mass position (depends on...)

$\rightarrow$ acceleration of mass

$\rightarrow$ sum of forces

$\rightarrow$ input F, mg, F_k*

$\rightarrow$ spring length



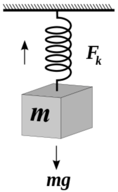
Acceleration of the mass *depends* on its position making this a dynamical system

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

Spring length & mass position (depends on...)
$\rightarrow$ acceleration of mass
$\rightarrow$ sum of forces
$\rightarrow$ input F, mg, F_k*
$\rightarrow$ spring length



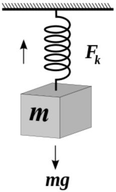
Acceleration of the mass *depends* on its position making this a dynamical system

What is a Dynamical System?

Dynamical System

A particle or ensemble of particles whose state varies over time and thus obeys differential equations involving time derivatives

Spring length & mass position (depends on...)
$\rightarrow$ acceleration of mass
$\rightarrow$ sum of forces
$\rightarrow$ input F, mg, F_k*
$\rightarrow$ spring length



Acceleration of the mass *depends* on its position making this a dynamical system

State Variable and State Derivatives

State Variables

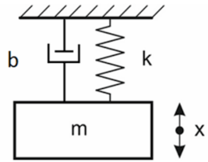
State Variables (initial conditions): the smallest possible subset of system variables that can represent the entire state of the system at any given time

State Derivatives

the derivatives of the state variables

Dynamical Systems are characterized by differential equations that relate the state derivatives to state variables

State Variable and State Derivatives

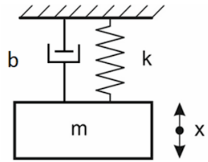


$m\ddot{x} = -kx - b\dot{x} + mg$

what are the state derivatives and state variables?

what are the state derivatives (acceleration and velocity) and state variables (velocity and position)?

State Variable and State Derivatives

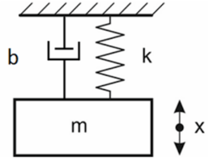


$m\ddot{x} = -kx - b\dot{x} + mg$

what are the state derivatives and state variables?

what are the state derivatives (acceleration and velocity) and state variables (velocity and position)?

State Variable and State Derivatives



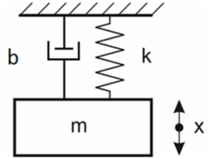
$m\ddot{x} = -kx - b\dot{x} + mg$

what are the state derivatives and state variables?

what are the state derivatives (acceleration and velocity) and state variables (velocity and position)?

System Order

The system order is defined as the highest derivative that appears in the differential equation.

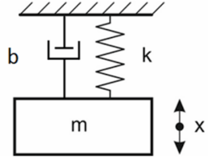


$m\ddot{x} = -b\dot{x} - kx + mg$

what is the system order(second order system)?

System Order

The system order is defined as the highest derivative that appears in the differential equation.

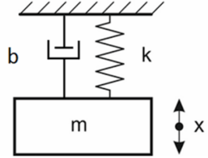


$m\ddot{x} = -b\dot{x} - kx + mg$

what is the system order? what is the system order(second order system)?

System Order

The system order is defined as the highest derivative that appears in the differential equation.



$m\ddot{x} = -b\dot{x} - kx + mg$

what is the system order(second order system)?

Coupled Differential Equations

1. Knowledge from one equation is required to solve another equation (and also sometimes vice-versa)
2. Take for example the following the Lotka-Volterra equations (predator-prey model):

$$\dot{x} = x(\alpha - \beta y) \quad (1)$$

$$\dot{y} = -y(\gamma - \delta x) \quad (2)$$

a. state variables ($x =$ prey population, $y =$ predator population)
b. state derivatives ($\dot{x} =$ prey reproduction rate, $\dot{y} =$ predator reproduction rate

Integration Schemes

Common Integration Schemes:

1. Euler
2. Runge-Kutta (RK4)

Integration Schemes — Euler

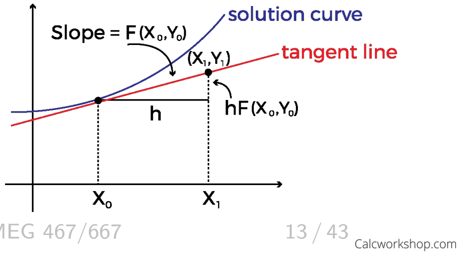
$$dy/dt = f(t, y); y(t_0) = y_0$$

$$y_{n+1} = y_n + h * f(t_n, y_n)$$

$$t_{n+1} = t_n + h$$

$$newvalue = oldvalue + stepsize \cdot slope$$

NOTE FOR GRAPH: replace t with X!



Euler — Simple Example

$$dy/dt = y' = 2t$$

$$s.t., h = 0.5; IC : t_0 = 0, y_0 = 0\, ; solve\ until\ t = 0.5s\ (1\ steps)$$

Euler — Simple Example

$$dy/dt = y' = 2t$$

$$s.t., h = 0.5; IC : t_0 = 0, y_0 = 0\,; solve\ until\ t = 0.5s\,(1\ steps)$$

white-board

n	t_n	y_n	$f_n = f(t_n, y_n)$	$h \cdot f_n$	$y_n + h \cdot f_n$
?	?	?	?	?	?

Euler — Simple Example

$$dy/dt = y' = 2t$$

$$s.t., h = 0.5; IC : t_0 = 0, y_0 = 0\,; solve\ until\ t = 0.5s\,(1\ steps)$$

white-board

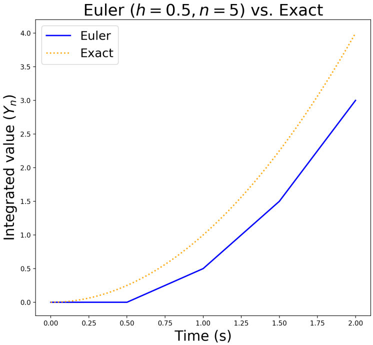
n	t_n	y_n	$f_n = f(t_n, y_n)$	$h \cdot f_n$	$y_n + h \cdot f_n$
0	0.0	0.0	0	0	0
1	0.5	0.0	1	0.5	0.5
2	1.0	0.5	2	1.0	1.5
3	1.5	1.5	3	1.5	3

Euler — Sample Python Code

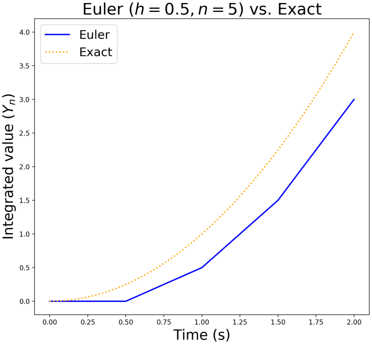
```
#Euler's Method
import numpy, math, matplotlib
from pylab import *

t0, y0, h, n = 0.0, 0, 0.5, 5 #initial conditions(t0,y0), step size, number of steps

#step, time, current integrated value (y_{n})
N, T, YN,  =  np.zeros(n), np.zeros(n), np.zeros(n)
# function (derivative), function * stepsize, n+1 integrated value (y_{n+1})
F, FH, YN1 = np.zeros(n), np.zeros(n), np.zeros(n)

for i in range(n):
    y1 = y0 + h*(2*t0)
    t1 = t0 + h
    N[i], T[i], YN[i], F[i], FH[i], YN1[i] = i, t0, y0, 2*t0, h*(2*t0), y1
    y0,t0 = y1,t1

plot(T,YN, linestyle = '-', linewidth = 2.0, color = 'blue')
show()
```

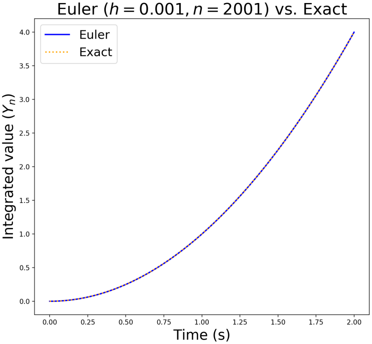
Write code to solve ODE, plot the exact value, change the steps (2001) and step size (0.001)—what do you notice?

Euler — Comparing step size



Euler — Comparing step size

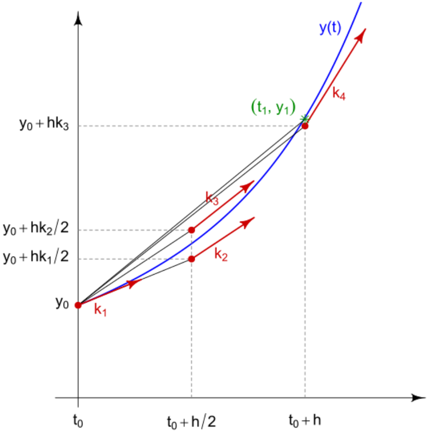




Euler Integration

1. intuitive
2. computationally fast
3. can produce compounding errors (solutions = use small step sizes or RK4)
4. error calculations outside the scope of this course

Integration Schemes — Runge-Kutta



Integration Schemes — Runge-Kutta

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$t_{n+1} = t_n + h$$

$$k_1 = h \cdot f(t_n, y_n),$$
$$k_2 = h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}),$$
$$k_3 = h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}),$$
$$k_4 = h \cdot f(t_n + h, y_n + k_3).$$

Integration Schemes — Runge-Kutta

$$dy/dt = y' = 2t$$

$$s.t., h = 0.5; IC : t_0 = 0, y_0 = 0\,; solve\ until\ t = 0.5s\,(1\ step)$$

$$k1t = t_0$$

$$k1y = y_0$$

$$k1 = h(2 \cdot k1t)$$

$$k2t = t_0 + \frac{h}{2}$$

$$k2y = y_0 + \frac{k1}{2}$$

$$k2 = h(2 \cdot k2t)$$

University of Delaware

Integration Schemes — Runge-Kutta

$$k3t = t_0 + \frac{h}{2}$$

$$k3y = y_0 + \frac{k2}{2}$$

$$k3 = h(2 \cdot k3t)$$

$$k4t = t_0 + h$$

$$k4y = y_0 + k3$$

$$k4 = h(2 \cdot k4t)$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Solution — Runge-Kutta

$$dy/dt = y' = 2t$$

$$s.t., h = 0.5; IC : t_0 = 0, y_0 = 0\,; solve\ until\ t = 0.5s\,(1\,step)$$

$$k1t = t_0; \quad [0 = 0]$$

$$k1y = y_0; \quad [0 = 0]$$

$$k1 = h(2 \cdot k1t); \quad [0 = 0.5(\,2 \cdot 0\,)]$$

$$k2t = t_0 + \frac{h}{2}; \quad \left[0.25 = 0 + \frac{0.5}{2}\right]$$

$$k2y = y_0 + \frac{k1}{2}; \quad \left[0 = 0 + \frac{0}{2}\right]$$

$$k2 = h(2 \cdot k2t); \quad [0.25 = 0.5(\,2 \cdot 0.25\,)]$$

Solution — Runge-Kutta

$$k3t = t_0 + \frac{h}{2}; \quad \left[0.25 = 0 + \frac{0.5}{2}\right]$$

$$k3y = y_0 + \frac{k2}{2}; \quad \left[0.125 = 0 + \frac{0.25}{2}\right]$$

$$k3 = h(2 \cdot k3t); \quad [0.25 = 0.5(\,2 \cdot 0.25\,)]$$

$$k4t = t_0 + h; \quad [0.5 = 0 + 0.5]$$

$$k4y = y_0 + k3; \quad [0.25 = 0 + 0.25]$$

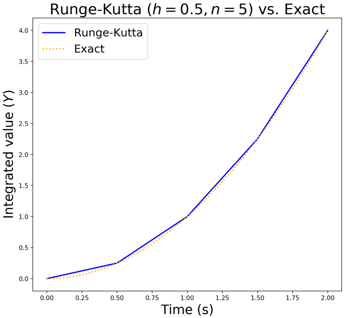
$$k4 = h(2 \cdot k4t); \quad [0.5 = 0.5(\,2 \cdot 0.5\,)]$$

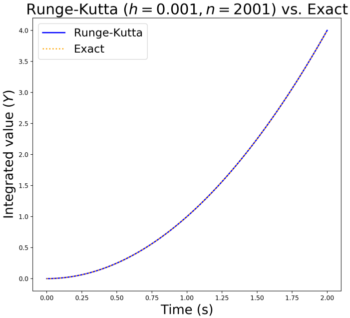
$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4); \quad \left[0.25 = 0 + \frac{1}{6}(0 + 0.5 + 0.5 + 0.5)\right]$$

RK4 — Sample Python Code

```
#4th order Runge Kutta
#Equation: dy/dt = 2t; initial conditions: y(0) = 0, t(0) = 0, h = 0.5
#exact solution y = t^2
import numpy, math, matplotlib
from pylab import *

#t0, y0, h, n = 0, 0, 0.5, 5 #initial conditions, step size, number of steps

T, YN = np.zeros(n), np.zeros(n) #place y0 in array
for i in range(n):
    k1t, k1y = t0, y0
    k1 = (2 * k1t) * h
    k2t, k2y = t0 + h/2.0, y0 + k1/2.0
    k2 = (2 * k2t) * h
    k3t, k3y = t0 + h/2.0, y0 + k2/2.0
    k3 = (2*k3t) * h
    k4t, k4y = t0 + h, y0 + k3
    k4 = (2*k4t) * h
    y1 = y0 + (1/6.0)*(k1 + 2*k2 + 2*k3 + k4)
    t1 = t0 + h
    T[i], YN[i] = t0, y0
    y0, t0 = y1, t1

plot(T,YN, linestyle = '-', linewidth = 2.0, color = 'blue')
show()
```

RK4 vs. Exact

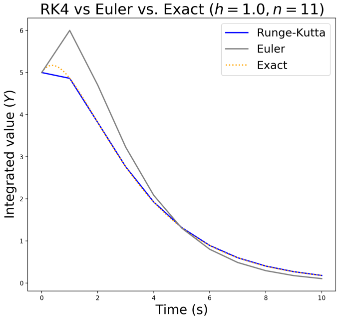

Runge-Kutta ($h = 0.5, n = 5$) vs. Exact
Runge-Kutta
Exact
Integrated value (Y)
Time (s)

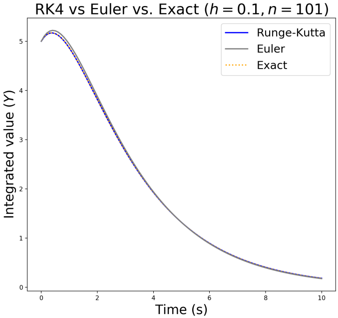

Runge-Kutta ($h = 0.001, n = 2001$) vs. Exact
Runge-Kutta
Exact
Integrated value (Y)
Time (s)

RK4 — Sample Python Code II

```
#RK4
#Equation: dy/dt = 3*exp(-t) - 0.4y; initial conditions: t(0) = 0, y(0) = 5, h = 1.5
#exact solution y = 10*exp(-0.4t) - 5*exp(-t)

#Runge-Kutta
t0, y0, h, n = 0, 0, 1.0, 11 #initial conditions, step size, number of steps
#t0, y0, h, n = 0, 5, 0.1, 101 #initial conditions, step size, number of steps

T, YN = np.zeros(n), np.zeros(n) #place y0 in array
for i in range(n):
    k1t, k1y = t0, y0
    k1 = (3*exp(-k1t) - 0.4*k1y) * h
    k2t, k2y = t0 + h/2.0, y0 + k1/2.0
    k2 = (3*exp(-k2t) - 0.4*k2y) * h
    k3t, k3y = t0 + h/2.0, y0 + k2/2.0
    k3 = (3*exp(-k3t) - 0.4*k3y) * h
    k4t, k4y = t0 + h, y0 + k3
    k4 = (3*exp(-k4t) - 0.4*k4y) * h
    y1 = y0 + (1/6.0)*(k1 + 2*k2 + 2*k3 + k4)
    t1 = t0 + h
    T[i], YN[i] = t0, y0
    y0, t0 = y1, t1

plot(T,YN, linestyle = '-', linewidth = 2.0, color = 'blue')
show()
```

RK4 vs. Euler vs. Exact





Runge-Kutta Integration

1. less intuitive
2. computationally intense
3. better approximation / smaller errors

ODE to Joy

1. Python (and other languages) have numerical integrators to make your life easier!
2. odeint
3. RK4 with adaptive step sizes.

Lorenz Attractor (coupled ODE):

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = (\rho - z)x - y$$

$$\frac{dz}{dt} = xy - \beta z$$

ODE to Joy

1. Python (and other languages) have numerical integrators to make your life easier!
2. odeint
3. RK4 with adaptive step sizes.

Lorenz Attractor (coupled ODE):

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = (\rho - z)x - y$$

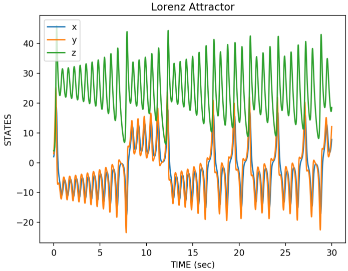
$$\frac{dz}{dt} = xy - \beta z$$

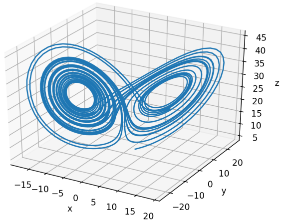
Lorenz Attractor — Sample Python Code

```
# Attractive Lorenz
#dx/dt = sigma(y-x)
#dy/dt = x(rho - z) - y
#dz/dt = x*y-beta*z
# inputs: state variables (x, y, z), time vector (t)
# outputs: state derivatives (xd, yd, zd)

from scipy.integrate import odeint
import numpy as np
from pylab import *

# function defining differential equation
def Lorenz(state, t):
    x, y, z = state[0], state[1], state[2] # unpack the state vector
    sigma, rho, beta = 10.0, 28.0, 8/3. # constants
    xd, yd, zd = sigma * (y - x), (rho-z) * x - y, x * y - beta * z # state derivatives
    return [xd, yd, zd]

# initial conditions (x, y, z)
state0 = np.array([2.0, 3.0, 4.0])

# time vector
tstart, tend, timestep = 0.0, 30.0, 0.01
t = arange(tstart, tend, timestep)

state = odeint(Lorenz, state0, t)
```

Lorenz Attractor

States vs Time & Phase Space

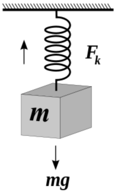




Converting Systems to 1st-Order ODEs

1. Reducing nth-order ODEs to 1st-order ODEs
2. Partial Differential Equations to ODEs (outside the scope of this course)

Convert higher order systems to 1st order ODEs



$$m\ddot{x} = -kx + mg$$

$$\ddot{x} = -\frac{k}{m}x + g$$

Here we want to convert this *2nd* order system into *two* ODEs

White board example

University of Delaware

Solution — Convert higher order systems to 1st order ODEs

$$\ddot{x} = -\frac{k}{m}x + g \quad (1)$$

We can convert an nth order differential equation into n 1st order ODEs by using a change of variable. Lets define two new functions.

$$x_1 = x \quad (2)$$

$$x_2 = \dot{x} \quad (3)$$

Now, take the derivative of eq. (2) and (3):

$$\dot{x_1} = \dot{x} \quad (4)$$

$$\dot{x_2} = \ddot{x} \quad (5)$$

Solution — Convert higher order systems to 1st order ODEs

Equations (4) and (5) will become our 2 first-order ODEs with the appropriate substitutions.

Sub (3) into (4)

$$\dot{x_1} = x_2 \quad (6)$$

Sub (1) into (5)

$$\dot{x_2} = -\frac{k}{m}x + g \quad (7)$$

Sub (2) into (7)

$$\dot{x_2} = -\frac{k}{m}x_1 + g \quad (8)$$

Solution — Convert higher order systems to 1st order ODEs

We now have our two 1st order ODEs (Eq. 6 and 8)!

$$\dot{x_1} = x_2$$

$$\dot{x_2} = -\frac{k}{m}x_1 + g$$

Which we can express in matrix form as:

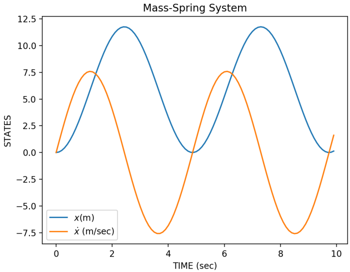
$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix}$$

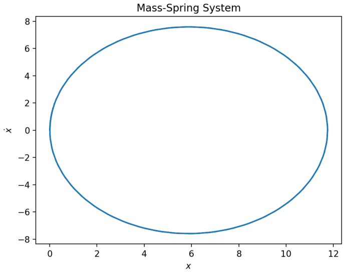
Mass-Spring — Sample Python Code

```
from scipy.integrate import odeint
import numpy as np
from pylab import *

# function defining differential equation
def MassSpring(state, t):
    x, xd = state # unpack the state vector
    k, m, g = 2.5, 1.5, 9.8 # constants
    xdd = ((-k*x)/m) + g # compute acceleration xdd
    # return the two state derivatives
    return [xd, xdd]

# initial conditions
state0 = np.array([0.0, 0.0])

# time vector
tstart, tend, timestep = 0.0, 10.0, 0.1
t = arange(tstart, tend, timestep)

state = odeint(MassSpring, state0, t)
```

Spring Mass System

States vs Time & Phase Space





Take Homes

1. Understand what is a dynamical system
2. Perform numerical integration by hand (Euler, RK4)
3. Program numerical integrators (Euler, RK4, odeint)
4. Convert nth order ODEs to n 1st order ODEs

QUESTIONS???

Next Week

1. Sensory Organs
2. Action Potentials
3. Nerve Models (Hodgkin-Huxley model)

Assignment

see handout

Office Hours

Acknowledgements

Dinant Kistemaker

Paul Gribble