3-D Programming
for Mere Mortals*

Rhett Allain
Troy Williams
Chemistry and Physics
Southeastern Louisiana University

*I stole this title*
A Problem:

Determine the motion of a mass attached to a spring
Solution Methods

- Analytically
- Numerically

Other solution methods:

- guessing
- cheating
- google
- psychic ability
Solution 1

\[
\vec{F}_{\text{spring}} = -k\Delta \vec{x}
\]

\[
\vec{F}_{\text{net}} = m\vec{a}
\]

\[
F_x = ma_x = m\frac{d^2 x}{dt^2}
\]

\[
-kx = m\frac{d^2 x}{dt^2} \quad \frac{d^2 x}{dt^2} = -\frac{kx}{m}
\]

\[
x(t) = A \cos(\omega t) \quad \omega = \sqrt{\frac{k}{m}}
\]
Numerical Solution

Key Ideas:

- Newton’s 2nd law: \( \vec{F}_{\text{net}} = \frac{d\vec{p}}{dt} \quad \vec{p} = m\vec{v} \quad \Delta \vec{p} = \vec{F}_{\text{net}} \Delta t \)

- Average velocity: \( \vec{v}_{\text{avg}} = \frac{\Delta \vec{r}}{\Delta t} \)

- For small time, \( F \) and \( v \) approx. const.
Numerical Recipe

- State initial conditions: Force, time, position, momentum
- update position \( x_f = x_i + v \cdot \Delta t \)
- calculate net force \( F = -k(x - x_0) \)
- update momentum \( p_f = p_i + F \cdot \Delta t \)
- update time \( t_f = t_i + \Delta t \)
- rinse, repeat.....
How to?

- MS Excel - or other spreadsheet
  - each increase in time is a row
- by hand
- VPython - or other language
Why VPython?

- Includes modules for 3-D animation and graphing
- Very extendable
- Built-in vector handling
- VPython is a visual module to Python (free, runs on linux, windows, mac)
```python
from visual import *
from __future__ import division
from visual.graph import *

mass = sphere(pos=vector(0,0,0), radius=0.01, color=color.yellow)
mass.m=1
mass.p=mass.m*vector(2,0,0)
ks=25

deltat =0.01
t=0

posgraph = gcurve(color=color.green)

while t<3:
    rate(30)
    fnet=-ks*(mass.pos)
    mass.p=mass.p+fnet*deltat

    mass.pos=mass.pos + (mass.p/mass.m)*deltat
    posgraph.plot(pos=(t, mass.pos.x))
    t=t+deltat
```
Creating a Simple Program

```python
from visual import *
sphere()
```

- F5 to save and run.
- Click right-mouse to rotate view
- Click both right and left to zoom in/out
Create Ground and Ball

```python
from visual import *

ball = sphere(pos=(0,0.1,0), radius=0.1, color=color.red)
floor = box(pos=(0,0,0), size=(1,0.05,1), color=color.green)
```

Now run - you should see a ball and ground
Initial Conditions

\begin{align*}
\text{ball.velocity} &= \text{vector}(0,5,0) \\
\text{ball.mass} &= 0.25 \\
\text{ball.p} &= \text{ball.velocity} \times \text{ball.mass} \\
\text{g} &= \text{vector}(0,-9.8,0) \\
\text{Fnet} &= \text{g} \times \text{ball.mass} \\
\text{dt} &= 0.01 \\
\text{t} &= 0
\end{align*}
**Baby Steps**

```python
while t<4:
    rate(300)
    ball.pos=ball.pos+(ball.p/ball.mass)*dt
    # Fnet=
    ball.p=ball.p+Fnet*dt
t=t+dt
```

- `rate(300)` says do no more than 300 calc per second
- run this and see what happens (it doesn’t work)
from visual import *

ball=sphere(pos=(0, 0.1, 0), radius=.1, color=color.red)
floor=box(pos=(0, 0, 0), size=(1, 0.05, 1), color=color.green)

# set up initial conditions
ball.velocity=vector(0, 5, 0)
baby.mass=1
ball.p=ball.velocity*ball.mass
g=vector(0, -9.8, 0)
Fnet=g*ball.mass

dt=0.01

t=0

while t<4:
    rate(300)
    ball.pos=ball.pos+(ball.p/ball.mass)*dt

    Fnet=
    ball.p=ball.p+Fnet*dt
t=t+dt
Bouncing

• Note: the program does exactly what it was told to do

• If the y position of the ball is less than the floor, then make it bounce

```python
if ball.pos.y < 0.1:
    ball.p = -ball.p
```

How would you make it lose energy after each bounce?
Air Resistance

- How to add air resistance?
  \[ \vec{F}_{air} = -\frac{1}{2} \rho C A v^2 \hat{v} \]

- \( \rho \) - density of air

- \( C \) - coefficient of drag (shape)

- \( A \) - cross-sectional area

- HINT: \( \text{mag(ball.p)} \) is magnitude of ball.p vector
  \[ \text{ball.p}/(\text{ball.mass*mag(ball.p)}) \]
Other Questions

- How high does it go?
- How long in the air?
- What happens if you shoot the ball at an angle?
Other Tools

- Graphing
- Printing data
- Trails
- Arrows
Graphing

- At the top of the program, add:

```python
from visual import *
from visual.graph import *
```

- Before the while loop

```python
posgraph = gcurve(color=color.green)
```

- In the while loop

```python
posgraph.plot(pos=(t, ball.pos.y))
```
Other Things

- Modeling motion of a rocket
- 3-body problem (or 10-body for that matter)
- Complex springs