Predictability and Interactive Visualization

Ramses van Zon

SciNet HPC Consortium

12 February 2015

Introduction

In previous session, we have talk about

- Computing for Modeling
- Analyzing Data to fit a model
- Using randomness
- Errors
- Lots of Python
- Using Public Data: Twitter API, Geocode, Exoplanets

Today, we'll focus on computation per se: when or why do we need it, and what are the limits of computability?

When must we compute?

Or more precisely, when must we calculate numbers using a computer?

- Sometimes just handy:
  E.g. what's the total after tax of a $45 item?
  By hand:

```
   113
×   45
------
   565
+ 4520
------
  5085
```

```
Using a calculator: $45 × 1.13 = $50.85
Still, the latter isn't really what we think of as 'computing'.
```

- Sometimes not a straightforward computation.
- Sometimes just a lot: 'Big Data'
- Sometimes complex.

Example that requires computation

- Okay, so a parachuter jump out of a plane 1 km above the ground.
- Let's say he's eager, draws the cord right away, so the drag coefficient is the same all the way down.
- Using a bit of math and physics, the height as a function of time is found to be given by

$$\mathbf{h(t) = h_0 - gst + gs^2(1 - e^{-t/s})}$$

(**g**=9.8, **s**=characteristic time to reach terminal velocity $\approx$ 1 sec)

- How long does it take to reach the ground?

$$\mathbf{0 = h_0 - gst + gs^2(1 - e^{-t/s})}$$

Even with parameters given, we can't solve this exactly. Must do so numerically (or try many, many times).

Counter example

- We ignore the drag, then

$$\mathbf{h(t) = h_0 - \frac{1}{2}gt^2}$$

- This, we can solve, since it's just a quadratic equation.
- Also at the end, we'd punch in numbers to compute a square root, but, again, this isn't really computing.

SciNet compute • calcul CANADA

Counter-counter example

- Let's replace the parachuter with a ball (for his sake).
- Once the ball has hit the ground (at the time we just computed), it bounces up at some fraction of the incoming velocity.
- What's the next collision, and the next, and the next, . . .
- Definitely do not want to do this by hand: automate = compute.

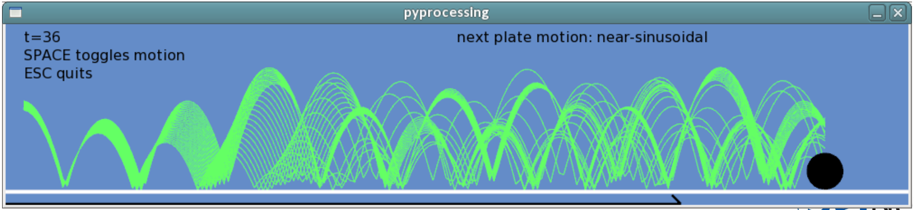
This example is a 'dynamical system'. Such models can be used for predictions. E.g. in this case: what is the maximum height in the 20th bounce?

Predictability

- Will introduce a model of a ball bouncing on a vibrating plate
- Use an interactive, visual implementation to explore how predictable this is.
- Since there are several bounces, this definitely requires computation.
- To investigate predictability, will use nearby starting conditions.
- Experimental realization: http://arxiv.org/abs/1405.3482

Predictability

- Will introduce a model of a ball bouncing on a vibrating plate
- Use an interactive, visual implementation to explore how predictable this is.
- Since there are several bounces, this definitely requires computation.
- To investigate predictability, will use nearby starting conditions.
- Experimental realization: http://arxiv.org/abs/1405.3482



How to get this Python App

Python Script for Interactive Ball on Vibrating Plate

Downloadable from: http://support.scinet.utoronto.ca/bounce.py

Requires modules pyglet and pyprocessing

```
pip install pyglet
pip install pyprocessing
```

On Window 7, I've had to do the following:

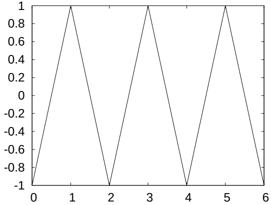
```
pip install pyglet
pip install --upgrade http://pyglet.googlecode.com/archive/tip.zip
DOWNLOAD AND RUN
https://pyprocessing.googlecode.com/files/
pyprocessing-0.1.3.22.linux-x86_64.exe
```

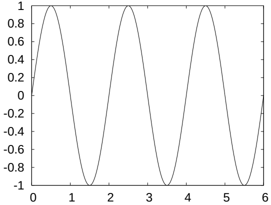
Play with it

What do we see?

Model

- The bouncing ball motion: $\mathbf{h(t) = at^2 + bt + c}$
  $\mathbf{a = -g/2}$ always, but $\mathbf{b}$ and $\mathbf{c}$ depend vary from bounce to bounce
- The floor: moves up and down either as a sawtooth or near-sinusoidal





Sawtooth

Near-sinusoidal

- Near-sinusoidal actually means piecewise quadratic, so also of the form: $\mathbf{z(t) = At^2 + Bt + C}$
- computation of the bounce time: solve a quadratic equation.

Some theory

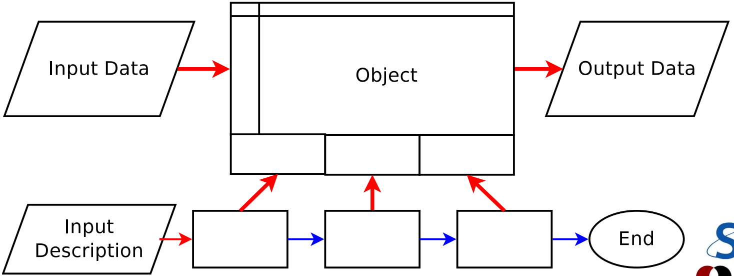
- Each bounce can be computed, but as we see from the collection of green trajectories, nearby situations diverge.
- Sawtooth and sinusoidal behave quite differently.
  - For sawtooth, clumps of trajectories stay together, but there are a couple different clumps.
  - For sinusoidal, at first the trajectories smoothly diverge, before going all over the place.
- In dynamical systems parleance, the former is called ‘non-chaotic’, the latter is call ‘chaotic’
- Other prime example of a chaotic system: The Weather.

Object Oriented Programming

- This type of simulation lends itself to so-called object programming.
- In Python (and most other languages), objects are collections of variables together with functions that act on this data
- Object can be contain other objects.
- The bounce.py is a fairly elaborate example: there are objects for the various balls and a floor object.

Objects

- Functional programming: data and the functions that can act on that data, are defined separately.
- Object oriented programming, the functions belong to the data structure.
- Better consistency, modularity, and reusability of your code.
- Implementation in python using the `class` construct.



Classes in Python

- Classes are used to group together data and code, accessing them with the `.` operator.
- One could also do this with modules. But there can be only one instance of a module, and many instances of a class.
- Inheritance: multiple base classes, derived class can override any methods of its base class or classes, and method can call a base class method with the same name.
- Objects can contain arbitrary amounts and kinds of data.
- As everything in Python, classes are dynamic: created at runtime, and can be modified further after creation.

Classes as collections of variables

```
class Apple:
    type = "Delicious"
    colour = "Green"
apple1 = Apple()
apple2 = Apple()
Apple.colour = "Golden"
print apple1.colour
```

Outputs: `Golden`

apple1 and apple2 *share* colour (class variable): tricky.

```
class Apple: pass
apple1 = Apple()
apple1.type = "Delicious"
apple1.colour = "Green"
apple2 = Apple()
apple2.type = "Delicious"
apple2.colour = "Golden"
print apple1.colour
```

Outputs: `Green`

This works, but now we have to assign each member.
Anything more workable requires writing a constructor.

Initializing objects with constructors

- Collection of variables
- Same `def` keyword to define methods.
- Constructor name is `__init__`

```
class Apple:
   def __init__(self):
      self.type="Delicious"
      self.colour="Green"
apple1 = Apple()
apple2 = Apple()
print apple1.colour
```

Outputs `Green`

Class syntax in Python

- Methods take a first argument that is an instance of the class
- This argument is explicit `self` in definition but implicit in calls.
- In methods, refer to member fields as `self.field`.
- No separation interface/implementation

```
class Apple:
   def __init__(self):
      self.type="Delicious"
      self.colour="Green"
   def describe(self):
      print self.type,
            self.colour

apple1 = Apple()
apple2 = Apple()
print apple1.colour
[Green]

apple1.describe()
[Delicious Green]
```

More special methods

- `__del__`
  A kind of destructor.
- `__str__`
  Converts object to a string for output. Used by `print`. Intended to be readable by users.
- `__repr__`
  Returns a string representation for the object. Used by python (e.g., if you just type the name of an object). Intended to be understandable by developers.
- `__enter__`
  Called when used in a 'with' construct (later)
- `__exit__`
  Called wen a 'with' construct is done (later).

Example: Particle

```
class Particle(object):
   def __init__(self,m,x0,v0):
      self.t = 0.0
      self.m = m
      self.x = x0
      self.v = v0
   def timeStep(self,dt):
      self.t += dt
      self.x += dt*self.v
   def __str__(self):
      return str(self.t)+" "+str(self.x)+" "+str(self.v)
p = Particle(2.0,0.0,-1.0)
while p.t <= 10.0:
   p.timeStep(0.1)
   print p
```

PyProcessing

- The interactive visualization used here is called pyprocessing.
- This is a python version of ‘processing’, an interactive visualization based off of Java.
- Other ports exist as well, such as in javascript or even c (partially).

From https://processing.org/:

- Free to download and open source
- Interactive programs with 2D, 3D or PDF output
- OpenGL integration for accelerated 3D
- For GNU/Linux, Mac OS X, and Windows
- Over 100 libraries extend the core software
- Well documented, with many books available

PyProcessing

- Fairly simple syntax to do drawing
- It's set up for interactive programming
- A 'draw' function is called repeatedly
- If mouse is moved, key is clicked, etc, a corresponding function can capture that event.
- Can be a nicer intro to programming than full-blown Java.
- Btw, it's on the Kahn Academy too, which some high school teachers use already, I believe.