Using Python in a Numerical Methods Course

Brian Heinold

Department of Mathematics and Computer Science
Mount St. Mary's University

August 6, 2016

About the class

- Mix of Math and CS students (counts as an elective for both)

About the class

- Mix of Math and CS students (counts as an elective for both)
- Calc I prereq, most students have also had an intro programming class

About the class

- Mix of Math and CS students (counts as an elective for both)
- Calc I prereq, most students have also had an intro programming class
- Covers floating point matters, interpolation, numerical equation solving, numerical integration and differentiation, numerical methods for differential equations, simulations

About the class

- Mix of Math and CS students (counts as an elective for both)
- Calc I prereq, most students have also had an intro programming class
- Covers floating point matters, interpolation, numerical equation solving, numerical integration and differentiation, numerical methods for differential equations, simulations
- We're a smallish liberal arts school, graduating about 10 total math and CS majors a year

What is Python

- General purpose programming language

What is Python

- General purpose programming language
- In top 5 or 10 of most lists of programming languages

What is Python

- General purpose programming language
- In top 5 or 10 of most lists of programming languages
- Popular in intro to programming courses

What is Python

- General purpose programming language
- In top 5 or 10 of most lists of programming languages
- Popular in intro to programming courses
- Used extensively in industry

What is Python

- General purpose programming language
- In top 5 or 10 of most lists of programming languages
- Popular in intro to programming courses
- Used extensively in industry
- You already have it if you have a Mac. Easy download on Windows.

Using the Python shell

Easy to show floating point gotchas:

```
>>> .2 + .1
0.30000000000000004
```

Using the Python shell

Easy to show floating point gotchas:

```
>>> .2 + .1
0.30000000000000004
```

```
>>> "{:.50f}".format(.1))
0.10000000000000000555111512312578270211815834045410
```

Using the Python shell

Easy to show floating point gotchas:

```
>>> .2 + .1
0.30000000000000004
```

```
>>> "{:.50f}".format(.1))
0.10000000000000000555111512312578270211815834045410
```

```
>>> x = (1.000000000000001 - 1) * 100000000000000
0.11102230246251565
```

Using the Python shell

Easy to show floating point gotchas:

```
>>> .2 + .1
0.30000000000000004
```

```
>>> "{:.50f}".format(.1))
0.10000000000000000555111512312578270211815834045410
```

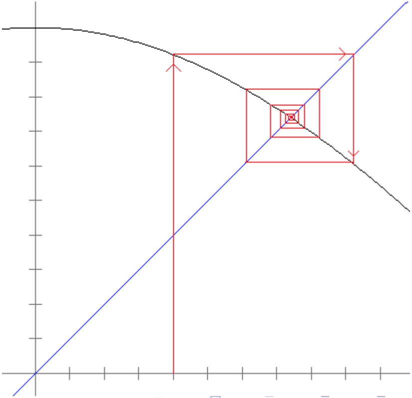
```
>>> x = (1.000000000000001 - 1) * 100000000000000
0.1110223024625156
```

```
>>> s = 0
>>> for i in range(10000000):
        s = s + .1
>>> s
999999.9998389754
```

Demonstration of Fixed Point Iteration

```
from math import cos
x = 2
for i in range(20):
    x = cos(x)
print(x)
```

-0.4161468365471424
0.9146533258523714
0.6100652997429745
0.8196106080000903
0.6825058578960018
...
0.7394108086387853
0.7388657151407354
0.7392329180769628



Python Is Easy to Work With

Python reads like pseudocode:

```
def bisection(f, a, b, n):
    for i in range(n):
        m = (a + b) / 2
        if f(a)*f(m) < 0:
            b = m
        else:
            a = m
    return m
```

Python Is Easy to Work With

Python reads like pseudocode:

```
def bisection(f, a, b, n):
    for i in range(n):
        m = (a + b) / 2
        if f(a)*f(m) < 0:
            b = m
        else:
            a = m
    return m
```

Can use anonymous functions passed as arguments:

```
bisection(lambda x:x*x-2, 0, 2, 20)
```

More Examples We Build in Class

```python
def secant(f, a, b, toler=1e-10):
    while f(b)!=0 and abs(b-a)>toler:
        a, b = b, b - f(b)*(b-a)/(f(b)-f(a))
    return b
```

```python
def trapezoid(f, a, b, n):
    dx = (b-a) / n
    return dx/2 * (f(a) + f(b) +
           2*sum(f(a+i*dx) for i in range(1,n)))
```

```python
def euler(f, y_start, t_start, t_end, h):
    t, y = t_start, y_start
    ans = [(t, y)]
    while t < t_end:
        y += h * f(t,y)
        t += h
    ans.append((t,y))
    return ans
```

Simulating Physical Systems

```
from tkinter import *
from math import *

def plot():
    v, y = 3, 1
    h = .0005
    while True:
        v, y = v + h*f(y,v), y + h*v
        a = 100*sin(y)
        b = 100*cos(y)
        canvas.coords(line, 200, 200, 200+a, 200+b)
        canvas.coords(bob, 200+a-10, 200+b-10, 200+a+10, 200+b+10)
        canvas.update()

f = lambda y, v: -9.8/1*sin(y)-v/10
root = Tk()
canvas = Canvas(width=400, height=400, bg='white')
canvas.grid()
line = canvas.create_line(0, 0, 0, 0, fill='black')
bob = canvas.create_oval(0, 0, 0, 0, fill='black')
plot()
```

Homework

- Homework usually consists of

  (a) Conceptual questions
  (b) Questions asking students to walk through an algorithm
  (c) Choice of a few programming or trickier math problems

Homework

- Homework usually consists of
  (a) Conceptual questions
  (b) Questions asking students to walk through an algorithm
  (c) Choice of a few programming or trickier math problems
- Many of our math majors don't like programming.

Homework

- Homework usually consists of
  (a) Conceptual questions
  (b) Questions asking students to walk through an algorithm
  (c) Choice of a few programming or trickier math problems
- Many of our math majors don't like programming.
- For some problems, I give the option to use a programming language or Excel.

Homework

- Homework usually consists of

  (a) Conceptual questions
  (b) Questions asking students to walk through an algorithm
  (c) Choice of a few programming or trickier math problems

- Many of our math majors don't like programming.

- For some problems, I give the option to use a programming language or Excel.

- For other problems, I give the choice to do a programming problem or a mathematical problem.

Example Exercises

- Write a Python program that implements Simpson's rule in an a manner analogous to the program we wrote in class for the trapezoid rule.

Example Exercises

- Write a Python program that implements Simpson's rule in an a manner analogous to the program we wrote in class for the trapezoid rule.
- Modify the backward Euler program we wrote in class to implement the implicit trapezoid method.

Example Exercises

- Write a Python program that implements Simpson's rule in an a manner analogous to the program we wrote in class for the trapezoid rule.

- Modify the backward Euler program we wrote in class to implement the implicit trapezoid method.

- Modify the Python code for adaptive quadrature to build up a list of all the points at which the algorithm evaluates the function while doing its thing.

Example Exercises

- Write a Python program that implements Simpson's rule in an a manner analogous to the program we wrote in class for the trapezoid rule.
- Modify the backward Euler program we wrote in class to implement the implicit trapezoid method.
- Modify the Python code for adaptive quadrature to build up a list of all the points at which the algorithm evaluates the function while doing its thing.
- Modify the Adams-Bashforth two-step program on Moodle to implement the four-step method.

Tricky Exercises

- Use the Python `Decimal` class, the Java `BigDecimal` class, or another programming language's decimal class to estimate the solution of $1-2x-x^5=0$ correct to 50 decimal places.

Tricky Exercises

- Use the Python `Decimal` class, the Java `BigDecimal` class, or another programming language's decimal class to estimate the solution of $1 - 2x - x^5 = 0$ correct to 50 decimal places.

- Use one of the numerical methods we've learned to write a method in your favorite programming language called `my_sqrt` that computes $\sqrt{n}$ as accurately as the programming language's own `sqrt` function (but without using the language's `sqrt` or power functions).

Tricky Exercises

- Use the Python `Decimal` class, the Java `BigDecimal` class, or another programming language's decimal class to estimate the solution of $1 - 2x - x^5 = 0$ correct to 50 decimal places.

- Use one of the numerical methods we've learned to write a method in your favorite programming language called `my_sqrt` that computes $\sqrt{n}$ as accurately as the programming language's own `sqrt` function (but without using the language's `sqrt` or power functions).

- Write a function that takes a sequence (a list), and returns a new sequence gotten by applying Aitken's Δ^2 method to it.

Tricky Exercises

- Use the Python `Decimal` class, the Java `BigDecimal` class, or another programming language's decimal class to estimate the solution of $1 - 2x - x^5 = 0$ correct to 50 decimal places.

- Use one of the numerical methods we've learned to write a method in your favorite programming language called `my_sqrt` that computes $\sqrt{n}$ as accurately as the programming language's own `sqrt` function (but without using the language's `sqrt` or power functions).

- Write a function that takes a sequence (a list), and returns a new sequence gotten by applying Aitken's Δ^2 method to it.

- Implement the method for estimating $\ln x$ discussed on page 33 of the notes to accurately approximate the natural log of any positive number.

More Tricky Exercises

- Write a function in a programming language that is given a list of data points, an x-value, and uses Newton's divided differences to compute the value of the interpolating polynomial at x. It's up to you how to specify how the data points are passed to your function, but make sure that it works for any number of data points.

More Tricky Exercises

- Write a function in a programming language that is given a list of data points, an x-value, and uses Newton's divided differences to compute the value of the interpolating polynomial at x. It's up to you how to specify how the data points are passed to your function, but make sure that it works for any number of data points.

- Write a program that returns the nth Chebychev polynomial, nicely formatted as a string. For instance, `cheb(5)` should return `16x^5-20x^3+5x`.

More Tricky Exercises

- Write a function in a programming language that is given a list of data points, an x-value, and uses Newton's divided differences to compute the value of the interpolating polynomial at x. It's up to you how to specify how the data points are passed to your function, but make sure that it works for any number of data points.

- Write a program that returns the nth Chebychev polynomial, nicely formatted as a string. For instance, `cheb(5)` should return `16x^5-20x^3+5x`.

- Write a Python function called `mc_integrate` that estimates $\int_a^b \int_c^d f(x,y)\,dy\,dx$. Its arguments should include the function f; the bounds a, b, c, and d; and the bounds of a box enclosing the region of integration; and an integer n specifying how many iterations to do, having a default value of 10000.

Project Ideas

- Write a program that allows the user to specify control points for Bézier curves by clicking and dragging and draws the Bézier curve determined by those points.

Project Ideas

- Write a program that allows the user to specify control points for Bézier curves by clicking and dragging and draws the Bézier curve determined by those points.
- Write a program that allows the user to add data points for interpolation and then draws the interpolating polynomial through those points.

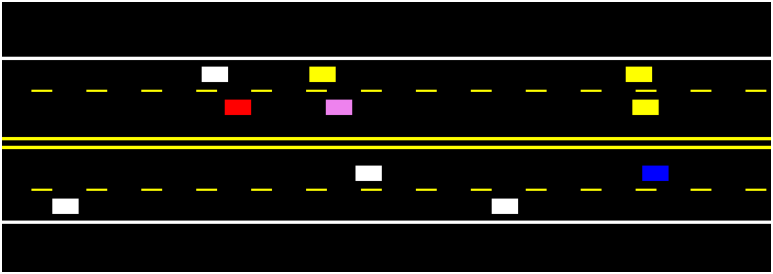
Project Ideas

- Write a program that allows the user to specify control points for Bézier curves by clicking and dragging and draws the Bézier curve determined by those points.
- Write a program that allows the user to add data points for interpolation and then draws the interpolating polynomial through those points.
- Write a program that simulates a physical system, like the pendulum programs we worked on in class. The program should graphically display the motion of the system.

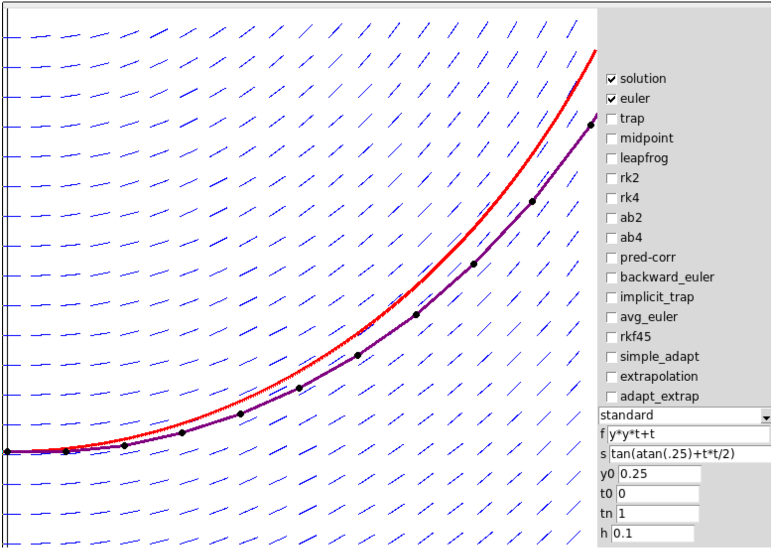
Project Ideas

- Write a program that allows the user to specify control points for Bézier curves by clicking and dragging and draws the Bézier curve determined by those points.
- Write a program that allows the user to add data points for interpolation and then draws the interpolating polynomial through those points.
- Write a program that simulates a physical system, like the pendulum programs we worked on in class. The program should graphically display the motion of the system.
- Simulations (graphical traffic flow, spread of disease, ...)
- Various non-programming ones involve writing a paper, comparing methods, ...

Screenshot from a Student Project



Diff Eq Plotter I Wrote for Class



Magic

```python
class Dual:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __add__(self, y):
        if type(y) == int or type(y) == float:
            return Dual(self.a + y, self.b)
        else:
            return Dual(y.a+self.a, y.b+self.b)

    def __mul__(self, y):
        if type(y) == int or type(y) == float:
            return Dual(self.a*y, self.b*y)
        else:
            return Dual(y.a*self.a, y.b*self.a + y.a*self.b)

    def __pow__(self, e):
        return Dual(self.a ** e, self.b*e*self.a ** (e-1))

# various other operator definitions omitted...
```

Magic, continued

```
def create_func(f, deriv):
    return lambda D: Dual(f(D.a), D.b*deriv(D.a))\
           if type(D)==Dual else f(D)

def autoderiv(s, x):
    f = eval('lambda x: ' + s.replace("^", "**"))
    return (f(Dual(x,1))-f(Dual(x,0))).b

sin = create_func(math.sin, math.cos)
exp = create_func(math.exp, math.exp)
# various other function defs omitted...

print(autoderiv("sin(x^2+exp(x+1))", 2))
```

Magic, continued

```
def create_func(f, deriv):
    return lambda D: Dual(f(D.a), D.b*deriv(D.a))\
           if type(D)==Dual else f(D)

def autoderiv(s, x):
    f = eval('lambda x: ' + s.replace("^", "**"))
    return (f(Dual(x,1))-f(Dual(x,0))).b

sin = create_func(math.sin, math.cos)
exp = create_func(math.exp, math.exp)
# various other function defs omitted...

print(autoderiv("sin(x^2+exp(x+1))", 2))
```

This is called automatic differentiation.

Results are always accurate to within machine ϵ!

Thanks!

- See `www.brianheinold.net` these slides.