Designing a Computational Mathematics Course for Math Majors

Holly Hirst and Greg Rhoads

About the Mathematical Sciences Department

- BS in Actuarial Science and BS in Mathematics, with concentrations in
  - Business **
  - Computation *
  - General Mathematics
  - Life Sciences
  - Physical Sciences *
  - Secondary Education
  - Statistics *
- Technology in the curriculum
  - Maple is used in Calculus 2 (required of all majors)
  - Matlab or Maple is used in Linear Algebra (required of all majors)
  - R is used in Statistical Data Analysis 1 (required of most majors)

Computational Mathematics Course Goals

Students will gain knowledge and insight into:

- Mechanics of numerical computation - computer representation and arithmetic of machine numbers, rounding error, algorithm analysis
- Symbolic versus numeric computation – analysis of solution techniques
- Structured programming – data types, control structures, input/output, and external procedures

Example Course Outline

1. Computer Mathematics Systems (Maple) - symbolic versus numerical calculations, functions, algebra and calculus
2. Spreadsheets (Excel) - relative versus absolute addressing; special functions: financial (fv, pmt), statistical (sum, average), logical (if, and, or, not); graphing and data fitting; pivot tables
3. Floating point representation - rounding error; IEEE standard integer and double precision representation; machine epsilon; cancellation error; conditioning
4. Structured Programming (Python) - data types (integer, floating point, list); subroutines (functions); loops (for, while); branches (if, elif); basic intrinsic function sets (math, random)
5. Applications and programming practice

Applications - Programming Practice

- Root finding (quadratic formula, bisection, Newton, secant)
- Series expansions (Taylor)
- Differentiation (difference formulas)
- Integration (rectangle, trapezoid)
- Optimization (golden section search, steepest descent)
- Initial value problems (Euler)
- Pseudorandom number generation (Mersenne prime algorithm)
- Primality testing (Fermat, Miller-Rabin)

An Example Revisited Frequently

Solve the differential equation $P'(t) = kP(t), P(0) = P0$

By hand: Separate and integrate

- In Maple: Use dsolve; dsolve-numeric; for loop
- In Excel: Discretize as $P_{t+h} = P_t + k \cdot P_t \cdot \Delta t$
- In Python: Create a for loop (and an array)

Add a rate slowing factor and a harvesting function, then solve again:

$$P'(t) = k\left(1 - \frac{P(t))}{M}\right)P(t) - H(t), P(0) = P0$$

Example

Use the Wilkinson Model (2,3,-2,2) to compute

$$0.75 \cdot (1.13 - 0.28)$$

by hand, rounding to the nearest machine number after each calculation (round up to break ties). Is the answer different if you distribute first?

Number	Base 10 Number	Gap	Relative Gap
0	0		
$.100 \times 2^{-2}$	.125	.125	
$.101 \times 2^{-2}$	.15625	.03125	.25
$.110 \times 2^{-2}$	.1875	.03125	.2
$.111 \times 2^{-2}$	.21875	.03125	.166667
$.100 \times 2^{-1}$	.25	.03125	.142857
$.101 \times 2^{-1}$	.3125	.0625	.25
$.110 \times 2^{-1}$	.375	.0625	.2
$.111 \times 2^{-1}$	.4375	.0625	.166667
$.100 \times 2^{0}$	.5	.0625	.142857
$.101 \times 2^{0}$	.625	.125	.25
$.110 \times 2^{0}$	.75	.125	.2
$.111 \times 2^{0}$	.875	.125	.166667
$.100 \times 2^{1}$	1	.125	.142857
$.101 \times 2^{1}$	1.25	.25	.25
$.110 \times 2^{1}$	1.5	.25	.2
$.111 \times 2^{1}$	1.75	.25	.166667
$.100 \times 2^{2}$	2	.25	.142857
$.101 \times 2^{2}$	2.5	.5	.25
$.110 \times 2^{2}$	3	.5	.2
$.111 \times 2^{2}$	3.5	.5	.166667

Simple Problems

Create a python function for each of the following:

1. Taking an integer n as input, calculate $\sum_{i=1}^{n}\sum_{j=1}^{i} i \cdot j$
2. Taking a floating point value x as input, calculate $T(x) = \begin{cases} e^{-0.3x} & x \leq 1 \\ 1 - .25x & x > 1 \end{cases}$ and return the result to a calling function.
3. Describe exactly what the printout will be for the following codes:

```
def newnum(x, y):
     while y > 0:
          print(x**y)
          y = y - 1

newnum(7, 2)
```

```
def testingx(x):
     if x > 0 and x < 10:
          x = 2 * x
     else:
          x = x / 2
     print (x)

testingx(2)
testingx(11)
```

Example Project

A remote European country completely enclosed by another country (to remain nameless!) has a young king, and he is the only remaining male of his royal line of Kzovcks. He is shortly to be married to a beautiful young American girl and hopes to have a male heir to inherit the crown. At any time, no male heir will result in the land being annexed to the surrounding country.
Based on past generations, we know the following: the probability that a male Kzovck will have exactly one male offspring is 6/11. The probability that he will have exactly two male offspring is 1/11, and the probability that he will have more than two offspring is zero. The king wants to know the whether within 10 generations the kingdom will be annexed. The code should have the following definition statement:

```
def kzovck(n):
```

where n is the number of runs of the simulation, and the code should print the computed probability of NOT dying out, i.e., the number of times you make it to 10 generations out of n runs. Discuss what happens for n = 200; 500; 1000 runs, and give a best estimate of the true probability.