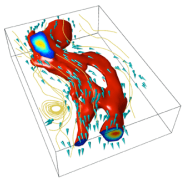
CS 403: Development of Scientific Computing Programs



Andrew Pershing
3134 Snee Hall
ajp9@cornell.edu
255-5552

Outline

- Course Description
- Details
- Policies
- Intro to CIS Tools Curriculum
- Role of Computing in Science and Engineering
- Basic Concepts
- Model problem

Course Goals

- This course will:
  - Examine the process of scientific software development
  - Discuss tools, both necessary and useful, for producing scientific software
  - Explore techniques for improving the efficiency of computer-based research

Syllabus

1. Intro, Philosophy, Model problem
2. Design of algorithms and responsible coding
3. Formal & Informal Specification
4. Editing, compiling: UNIX vs. IDE, intro to architectures
5. Language issues: C, Fortran, Java, MATLAB
6. Building with Make
7. Debugging: UNIX db vs. IDE
8. Testing for correctness
9. Improving performance--profiling, tuning
10. Software management, source code control
11. Platform issues & how to spend your advisor's money
12. Trends for the future

Course Ungoals

- This course will NOT:
  - Teach you how to program (try CS 100m)
    - You should be comfortable writing programs in some language (C, Matlab, FORTRAN, Java,…)
  - Teach you numerical methods (CS 32X, 62X)
  - Teach you UNIX
    - we will discuss some UNIX tools (Windows,too), but not general features of the UNIX OS nor how to write scripts

Course Business:

- http://www.cs.cornell.edu/Courses/cs403/2002sp
  - Contains syllabus, lecture notes, examples, homework
- Office Hours
  - Tuesday & Wednesday, 11-1 in 3134 Snee (or by appointment)
- Registration:
  - get my signature or CS Undergrad office (303 Upson)
  - # 441-198
  - S/U only, 1 credit
  - Last day to add/drop: Monday, Feb. 25!

Requirements

- No official text
- Need to find a computer where you can
  - 1. edit text and do e-mail
  - 2. compile code (mostly C)
  - 3. Check out ACCEL Facility in Carpenter Library, departmental labs

Course Policies

- 4 assignments: 1 per week, due Wednesday, 5PM by e-mail
- If you complete each assignment on time and demonstrate a basic command of the material, you will pass!
- Course policies are strict:
  - A direct consequence of the "mini-course" format
- This course operates as a contract between you and me

The Contract

- I agree to:
  - Begin and end lecture on time
  - Put lecture notes on website before lecture
  - Be available during office hours
  - Make the assignments of reasonable length (~2 hours) focusing on material from lectures

The Contract

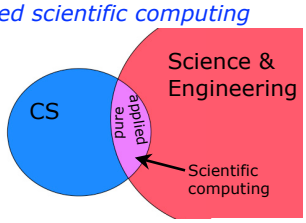
- By registering for the course, you agree to:
  - Arrive on time
  - Participate in the course by asking questions and coming to office hours
  - Turn in your assignments on time
    - Late work will not be accepted and will jeopardize you chance of passing!
    - The only exceptions are for documented, university-sanctioned reasons such as severe illness or by prior arrangement made w/ me 3 days before (includes religious holidays, sports, etc.)

CIS and FCI

- *Cornell University has recognized that computing and information science has emerged as a key enabling discipline vital to nearly all of its scholarly and scientific pursuits.*
- *The Faculty of Computing and Information is founded on the recognition that the ideas and technology of computing and information science are relevant to every academic discipline.*
- *We are united in the need to bring together a core of faculty in this field from across the traditional colleges.*

CIS Tools Curriculum

- CS 403 (should be CIS 403) is the third in a series of courses designed to teach *applied scientific computing*



CIS Tools Curriculum

- "Pure" Scientific Computing
  - Focus is on algorithms for general problems such as optimization, linear systems, differential equations
  - Concerned with accuracy, stability, and efficiency of these algorithms
- "Applied" Scientific Computing
  - How to apply general algorithms to solve scientific problems
  - Algorithms are "black boxes" that we string together to get our work done

CIS Tools Curriculum

- Fall: MATLAB
  - 401: the basics
  - 402: visualization (starts October 15)
- Spring: General tools
  - 403: Developing scientific computer programs (compilers, debuggers, managing large projects)
  - 404: Numerical libraries

Key Questions

- There are several questions we will try to address in the next 4 weeks
  - How do scientists use computers? Do scientists have unique requirements?
  - What processes are common to the development of scientific software?
  - As scientists, we're paid for scientific results, not time spent hacking. How can we make the development process more efficient?
  - What tools are available to help us? How do they work and how do they differ across platforms?

Applied Scientific Computing

- Emphasis is less on developing new algorithms, rather, it is on obtaining new scientific results.
  - We are either running a simulation, or analyzing data (perhaps from a simulation).
  - We need to be able to develop new code or modify existing code to fit our needs
  - We should make this process easier for ourselves or colleagues the next time.
  - We need to get the code to run on our system.
  - We will need to debug the code and verify that it is solving the correct problem.
  - We will need to work within (or oversee) a group of programmers

A Unique Requirement

- Scientific results must be reproducable
  - This applies to computational results, too
  - We must accurately describe
    - Inputs to our programs
    - Details of our code--algorithms, parameter values

Model Problem

- Since we're looking at the process of scientific software development, we'll focus on a single example problem
- We will work out the design and specification of a program to solve this problem
- We will debug and test it
- We will improve its performance

Model Problem: Advection-Diffusion-Reaction in 1D

- Related equations occur in many fields
  - Fluid flow in atmosphere, ocean, lakes, universe
  - Biological development
  - Chemistry
  - Ecology

RAD

$$\frac{\partial C}{\partial t} = u\frac{\partial C}{\partial x} + \frac{\partial}{\partial x}\left(k\frac{\partial C}{\partial x}\right) + r(C,x,t)$$

- This is not a math class, nor is it a course on numerical methods.
- Focus on the big picture (what we're doing, what the components are) rather than on the details

Total Change = Advection + Diffusion + Local Change Or Growth

RAD

- u and k can be functions of x and t
- Means we need to carry out d/dx in diffusion term:

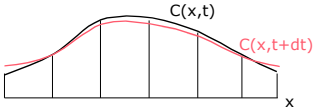
$$\frac{\partial k}{\partial x}\frac{\partial C}{\partial x} + k\frac{\partial^2 C}{\partial x^2}$$

- Can group dk/dx with u in advection term:

$$\frac{\partial C}{\partial t} = \left(u + \frac{\partial k}{\partial x}\right)\frac{\partial C}{\partial x} + k\frac{\partial^2 C}{\partial x^2} + r(C,x,t)$$

Numerical Solution

- We start with an initial distribution of C over the interval [0 1]
- Divide [0 1] into discrete points separated by dx



- C(x,t+dt) will depend on C(x), C(x-dx), & C(x+dx)

Numerical Solution

- replace partial derivatives with differences (k=constant):

$$\begin{bmatrix} -\sigma & (1+2\sigma) & -\sigma \end{bmatrix} * \begin{bmatrix} C_{x-dx}^{t+dt} \\ C_{x}^{t+dt} \\ C_{x+dx}^{t+dt} \end{bmatrix} = C_{x}^{t} + \lambda\left(C_{x+dx}^{t} - C_{x}^{t}\right)$$

- The solution of C(x,t+dt) depends on neighboring points

Numerical Solution

- We have a system of n linear equations with n unknowns (C_1, C_2,…, C_n)
- In linear algebra, we write this as a matrix problem:
  - $A * \underline{C}^{t+1} = \underline{f}^{t}$
- There are many ways to solve these problems

Numerical Solution

- Each C_x will have a row in matrix A
- All rows are the same except for first and last
  - We need to specify what happens at end points
  - Boundary conditions are a big problem
  - We'll use periodic BC's
    - C(0)=C(1), so first and last rows are:

$$\begin{bmatrix} (1+2\sigma) & -\sigma & \ldots & -\sigma \end{bmatrix}$$

$$\begin{bmatrix} -\sigma & \ldots & -\sigma & (1+2\sigma) \end{bmatrix}$$