CIS 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
Course Structure

• This course will mix lectures and computer labs:
  – Mondays: Lectures in 314 Hollister
  – Wednesdays & Fridays: Labs in ACCEL Green Room (upstairs in Engineering Library)
• You will be graded on 3 assignments (easy), a programming project, and attendance

Syllabus

1. Intro, Philosophy, Model problem
2. Software design and responsible coding
3. Editing, compiling: UNIX vs. IDE, intro to architectures
4. Formal & Informal Specification
5. Language issues: C, Fortran, Java, MATLAB
6. Building with Make
7. Testing for correctness
8. Debugging: UNIX db vs. IDE
9. Software management, source code control
10. Performance issues
11. Improving performance—profiling, tuning
12. Class projects

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
    • Try CS114, CS214
    • Also, EAS 494 Intro to the Linux Supercomputing Environment will cover a range of UNIX issues
Course Business:

  - Contains syllabus, lecture notes, examples, homework
- Office Hours
  - Tuesday & Wednesday, 10-11 in 3134 Snee (or by appointment)
- Registration:
  - get my signature or CS Undergrad office (303 Upson)
  - # 621-315
  - S/U only, 1 credit
  - Auditors welcome, but please register
  - Last day to add/drop: Monday, Feb. 27!

Requirements

- Recommended text:
  - Myler’s “Fundamentals of Engineering Programming with C & FORTRAN”
  - Not required, but an inexpensive reference for C & FORTRAN
- 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

- To pass, you must
  - Attend Monday lectures and Wed-Fri labs
  - Need permission to miss more than one lab
  - Complete 3 assignments: 1 per week, due Wednesday, 5PM by e-mail
  - These will be very easy–simple, short answer questions straight from lectures & labs
  - Designed to “certify” your knowledge of the material
  - Complete a programming project: due Friday, 2/21
    - Opportunity to apply ideas from class to a project relevant to you
    - Details in a minute
  - 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 (<1 hour) 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 your 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.)

Programming Project

Lectures and Labs are good for presenting material
- Only way to really learn something is to apply it yourself in a novel setting
- The more relevant the application is to you, the more you will learn
- This is the purpose of the projects
Programming Projects

- Projects could include:
  - Creating a new program to simulate some phenomenon or analyze/transform some data object
  - Modifying an existing program to extend its functionality or improve its performance

More important than the actual project is the process:
- I want you to think about how scientific software is developed and what tools/techniques make it easier.

To evaluate you, you must provide:
- A specification statement describing inputs and outputs
- A diagram showing structure of your program
- Source code for your program and a Makefile to build it.
- A small data set and description of how to use it in order to demonstrate correctness of your program
- A 1-2 page narrative describing and evaluating the development of your program.

Groups
- You may work in groups of no more than 3 people
- If you work in a group, I want you to divide the responsibilities and describe in the narrative how this went.
  - Ex: Person A writes subroutines X & Y, Person B writes subroutine Z and the Makefile, Person C is responsible for testing
  - No programming together--this doesn't work!
- This is only a 1 credit course, so I don't expect you to spend more than a couple hours/week on the project.
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

- CIS 403 is the third in a series of courses designed to teach applied scientific computing

  "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 reproducible
  - This applies to computational results, too
  - We must accurately describe
    - Inputs to our programs
    - Details of our code—algorithms, parameter values
    - Experimental conditions—system, compiler, compiler options

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
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.

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

- Total Change = Advection + Diffusion + Local Change or Growth

\[
\frac{\partial C}{\partial t} = \frac{u + \partial k}{\partial x} \frac{\partial C}{\partial x} + k \frac{\partial^2 C}{\partial x^2} + r(C, x, t)
\]

Numerical Solution

- We start with an initial distribution of C over the interval \([0, 1]\)
- Divide \([0, 1]\) into discrete points separated by \(dx\)
- \(C(x, t)\) will depend on \(C(x)\), \(C(x-dx)\), & \(C(x+dx)\)
Numerical Solution

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

\[
\begin{align*}
\frac{C_{i+1}^{n+1} - C_i^n}{\Delta x} &= \frac{C_{i+1}^{n+1} - C_i^n}{\Delta x} + \frac{C_{i+2}^{n+1} - C_{i+1}^{n+1}}{\Delta x} \\
C_{i+1}^{n+1} &= \phi \left( C_{i+2}^{n+1} - 2C_{i+1}^{n+1} + C_{i}^{n+1} \right) \\
\left[ \begin{array}{ccc}
-\phi & (\phi+2\sigma) & -\phi \\
-\phi & \ldots & -\phi \\
-\phi & \ldots & -\phi & (\phi+2\sigma) \\
\end{array} \right] \begin{bmatrix} C_1^n \\ C_2^n \\ \vdots \end{bmatrix} &= \begin{bmatrix} C_1^{n+1} \\ C_2^{n+1} \\ \vdots \end{bmatrix} + \phi \left( C_2^{n+1} - C_1^{n+1} \right)
\end{align*}
\]

• The solution of \( C(x,t+\Delta t) \) depends on neighboring points

Numerical Solution

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

Numerical Solution

• Each \( C_i \) 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:
      \[
      \begin{bmatrix}
      (1+2\sigma) & -\sigma & \ldots & -\sigma \\
      -\sigma & \ldots & -\sigma & (1+2\sigma) \\
      \end{bmatrix}
      \]