An Ad Hoc Approach to Undergraduate Curriculum Development in Computational Science

DOE High Performance Computing
Education Conference
Albuquerque NM
February 10, 1994 (updated)

Kris Stewart
Associate Professor
Department of Mathematical Sciences
San Diego State University
San Diego CA 92182-0314

Computational Science Curriculum Coordinator
San Diego Supercomputer Center
A National Laboratory for
Computational Science and Engineering

stewart@cs.sdsu.edu

At a state university, such as San Diego State University (SDSU), it is difficult to develop large, comprehensive academic programs. With the last few years of severe budget crisis in California, this has become even more difficult. Still, there is a need to incorporate changing technology and prepare students to satisfy the needs of local industry. One major advantage for SDSU is the fact that it was one of the original consortium members of academic institutions when the National Science Foundation founded the five centers for high performance computing. Having one of the national centers in your own town is an advantage for any state school (though easy network access is changing this).

Two courses have been accepted into the SDSU undergraduate catalog recently and I wish to describe them in this paper. Additionally, I will highlight other nationwide programs providing curriculum materials, freely, over the Internet. The first course will be described in rough terms only because it has not yet been taught. Due to the backing of the San Diego Supercomputer Center, **CS 205 Introduction to Computation Analysis** will be taught Fall Semester 1994.

CS 205 Introduction to Computational Analysis

The intent of this sophomore level course is to reinforce calculus and begin a student's introduction to programming and floating point arithmetic in a focussed manner. The focus is on the skills that are essential to students in the sciences if they plan to pursue a degree in Computational Science.

Many students become familiar with word processors, spreadsheets and communications packages in high school or at home due to the wide spread use of personal computers. The fundamental skill missing from the use of these standard software tools is an understanding of floating point arithmetic. This includes estimating errors, estimating rates of convergence, programming in a manner to avoid floating point exceptions like overflow/underflow and much more.

Familiarity with floating point arithmetic is essential in computational science whether students intend to do their own programming or they intend to use discipline-specific computational packages. The proposed course would illustrate this arithmetic using numerical approximations derived directly from Taylor Series, thereby reinforcing the calculus training that students have just received in their freshman year.

The programming environment will be MATLAB. This is a powerful software tool that facilitates learning Linear Algebra concepts, but MATLAB would be used here for its other properties.

o available and uniform over different computing platforms, including a very affordable student edition which students with home computers could purchase

o graphics capabilities are reasonably simple to use and easily extend to publication quality graphics

o programmability: although MATLAB has an extensive interactive capability to compute solutions for most standard problems of linear algebra in a numerically stable manner, MATLAB also contains a subset of the C programming language that can be used to introduce students to programming.

More details on this course will evolve as Fall Semester gets closer. Just as for the CS 575 course described below, the course materials will be made available via anonymous ftp once they exist.

National Programs Providing Curricula Materials over the Internet

Before describing the supercomputing course, which is a well developed course, I would like to highlight some alternate programs that I have either participated in, or have just been told of. My association with the San Diego Supercomputer Center has been very helpful in this regard, since SDSC tends to attract computational scientists and the SDSC staff will then tell me of other work going on.

A recent example is Dr. Ian Johnston at the School of Physics at the University of Sydney, NSW 2006, Australia. We have just begun communicating electronically and he has mailed (postal type) some articles of his concerning his work with M.U.P.P.E.T (Maryland University Project for Physics Education Technology).

I was also fortunate to attend the faculty workshop given by Dr. Lloyd Fosdick and Dr. Liz Jessup in the summer of 1992 at the University of Colorado, Boulder. Dr. Jessup is also on this panel and will provide more details. The course notes are extensive and well developed and are available via anonymous ftp from ftp.cs.colorado.edu in the directory pub/HPSC, I'd recommend retrieving the README as well as the UNIX compressed, postscript file, SciCompOverview.ps.Z.

In the last year, I have joined a DOE funded program from Ames Laboratory call UCES (Undergraduate Computational Science and Engineering). Quoting from the program's flyer: "This project strives to promote the emerging field of computational science as an interdisciplinary/multidisciplinary approach to scientific analysis. Currently our emphasis is on collecting, developing and distributing a set of educational materials in computational science." For more information on the program, contact Dr. Tom Marchioro (uces@uces.ameslab.gov). The first course to emerge from this work is a freshman level course on Engineering Computing at the University of Utah, developed and taught by Dr. Chris Johnson and Dr. Joe Zachary (who are at this conference). The materials are available via anonymous ftp from cs.utah.edu and form a tutor system which runs in conjunction with Maple to introduce students to the GUI environment; UNIX; computing, programming and visualization using Maple; as well as C and Fortran programming. This program is listed in the October 1993 issue of Computer with its cover story on "Computational Science & Engineering".

Another program which I heard of from Dr. Robert Borrelli is the Consortium for Ordinary Differenential Equations Experiments (C*ODE*E) from Harvey Mudd College. This group produces a newsletter (email codee@hmc.edu to subscribe to the hardcopy or use anonymous ftp to ftp.hmc.edu in the directory /pub/CODEE to obtain the postscript files of back issues.)

One of the recent issues described a program through St. Olaf College on modeling. There are currently eight complete projects available via anonymous ftp to ftp.stolaf.edu which cover:

Eulerian Wobble
First Order Bifurcations
Pursuit Differential Equations
Building a Model: Predator-Prey Simulations
Competing Species with Harvesting
Regular Perturbation
Regular Perturbation II: The Precession
of Perihelion of Mercury
The Salt in the Tank Problem

Each postscript file contains a problem description, notes to instructor, solution code in Maple and figures of the computed solutions. These were prepared by Dr. Steven Kennedy at St. Olaf and his group.

As a last item in my ramblings here, I present my current favorite NCSA Mosaic sites:

http://gopher.sdsc.edu/Home.html SDSC's Home Page
http://gopher.sdsc.edu/SR93/SDSC_SR.html 1993 Science Report from SDSC

http://compsci.cas.vanderbilt.edu/csep.html Computa- tional Science Education Project (CSEP) the graduate computational science program supported by DOE.

http://info.cern.ch.hypertext/DataSources/Yanoff.html Yanoff's List

http://www.gatech.edu/gvu/gvutop.html Introductory Graphics and Visualization

http://galaxy.einet.net/GJ/inidex.html Traditional structure for Galaxy's Gopher Jewels
http://galaxy.einet.net/gopher/gopher.html Searchable form for Galaxy's Jewels

file://dggis.rtpnc.epa.epa_ceam/wwwhtml/ceam_home.html EPA Center for Exposure Assessment Modeling
http://www.exploratorium.edu/ SanFranciscoExploratorium

CS 575 Supercomputing for the Sciences & Engineering

The second course at SDSU, is **CS 575 Supercom- puting for the Sciences**. This course is being taught this semester for the third time, the first two times as an "experimental CS 596" course, which can be offered at a department's discretion. The course appeared in the 1993/94 Undergraduate catalog after the campus wide scrutiny process was satisfied. This course was supported by a grant from the Divsion of Advanced Scientific Computing at NSF. It was coupled with a workshop offerred in the summer for faculty, from primarily undergraduate institutions, who were interested in incorporating **H**igh **P**erformance **C**omputing and **C**ommunications into their undergraduate curricula. These workshops, called **SUE** (Supercomputer and Undergraduate Education) were offerred in summer 1991 and 1992. These faculty workshops also received supplementary funding and support from the Cray Research Foundation.

The course CS 575 focusses on the Cray vector supercomputer and uses the text, Computer Architec- ture: A Quantitative Approach by Hennessy and Patterson. It is an interdisciplinary course to introduce students in the sciences and engineering to advanced computing techniques using the Cray C90 Supercomputer at the San Diego Supercomputer Center.

Prerequisites are an extensive programming background in Fortran or C.

Students are graded based on computing projects (75%) and one take home midterm (25%) based on material from the text by Hennessy and Patterson.

The actual syllabus of the course being taught Spring 1994 is given below:

[UNIX] The Cray is a UNIX machine. We begin the semester using the SUN SPARC 10 machine while students become reasonably functional with UNIX and electronic mail. Students are given a demonstration of using gopher to access copies of the lectures notes which are available from **ftp.sdsc.edu** (1 week)

There is a salmon-colored handout available at this meeting describing how to access all the lecture notes via anonymous ftp. This is also included as an Appendix to this paper.

[Architecture] To effectively use the Cray, you need an understanding of the hardware features of the machine that make it so fast. We use the text by Hennessy and Pattern to gain an INTUITIVE understanding of the architecture of a vector supercomputer. (6 weeks)

[Chapter 1] Fundamentals of Computer Design
[Chapter 2] Performance and Cost
[Sect. 3.3] Operand Storage in Memory - Classifying General-purpose Register Machines
[Sect. 3.7] The Role of High-Level Languages and Compilers
[Sect. 4.5] The DLX Architecture
[Chapter 6] Pipelining
Goal->[Chapter 7] Vector Processors
[Sect. 8.1] Memory-Hierarchy Design; Introduction
[Sect. 8.2] General Principles of Memory Hierarchy
[Sect. 8.4] Main Memory (Note - Cray is not a virtual memory machine)
[Sect. 8.7] More Optimizations Based on Program Behavior

[Computers and Ethics] Guest lecturer from SDSC visits the class. There is an essay assignment (which is graded on a pass/fail basis) (1 week)

[Performance Evaluation/Usage Monitoring] The class starts working on the Cray at this point. Students are frequently reminded that they have a FIXED amount of Cray time to use to complete their computer projects. Student must learn to manage their allocation of a valuable, but scarce, resource. (2 weeks)

[Impact of Architecture on Algorithms] Cray-specific. (2 weeks)

[Software Tools] Effective use of the Fortran or C compilers. Debugging and performance tools. (3 weeks)

Major Themes of the SUE Program (faculty workshop and course at SDSU)

1 Interdependence of Computer Science & Scientific Experts

The workshop faculty represented two groups: those interested in learning discipline-specific applications packages and those interested in using software tools to facilitate programming. To accommodate both groups, we presented an overview of available resources (applications packages and program optimization techniques) and encouraged faculty to seek further information independently. We identified the technical people they could contact for further information on their particular interests.

Similarly, the required programming background for the course reinforced the traditional separation between computer science and science/engineering students by encouraging the former to take the course. In both cases, the participants interacted well as they realized the benefits of working with others with different strengths.

2 CRAY architecture

The course uses the text Computer Architecture: A Quantitative Approach, by Hennessy and Patterson (published by Morgan Kaufmann) because the model developed for the Cray 1 provided a great deal of insight into the design of the Cray C90.

We try to gain an appreciation for the sources of the Cray C90's power and understand the subtleties of its design as they impact the way a programmer should approach a problem. We carefully avoid many of the sections of the text that do not directly relate to the Cray C90. The text is supplemented with documents by Cray Research, Inc.; particularly the following titles:

TR-OPT (Rev. D) cf77 & scc Features and Optimization. An excellent, Cray-specific training report covering the Fortran (cf77) and C (scc) compilers. Provides code examples, diagrams, and explanations of crucial vectorization topics (and the conditions that inhibit the compiler), memory organization, performance tools, common optimization techniques, and much more.

TR-YSAAP Cray Y-MP System Architecture for Applications Programmers. Covers material at the assembler level in more detail. Works well with the models developed in Hennessy and Patterson (unfortunately this document is no longer available)

3 Communications

Both the workshop and the course covered the resources available through the Internet, including:

a. Accessing sources of information
(news groups, anonymous FTP):
oak.oakland.edu (IBM PC software)
sumex-aim.stanford.edu (Macintosh softw)
ftp.cs.titech.ac.jp
(Dr. David Kahaner's Japan Bulletins)

b. Communicating with peers (via e-mail)

c. Using FTP to transfer programs between machines using gopher.

4. Measuring Performance

At many universities, accounts issued to students in computer courses are limited only by the amount of available file space. Therefore, the concept of monitoring CPU usage can be new. The students in the SDSU class are given programming projects on a mainframe at SDSU and are introduced to the concept of accounting using crude UNIX timing tools (e.g., *dtime*) before they move to the Cray.

When students program on the Cray, they have a fixed amount of CPU time to work with. Students are reminded that they must complete their course projects without exceeding their CPU allocations. The more sophisticated timing and resource monitoring tools are used to ensure that the student projects do not use up more time than they are allocated for the semester.

5 Computer ethics/responsibility

Students in the course choose between two computer-related scenarios and write a one-page essay on a scenario, discussing the behaviors of the individuals involved. The essays were graded on a pass/fail basis (only essays that showed a lack of thought or effort "failed"). The goal, for the instructor, was to gain a better understanding of the students' attitudes concerning computer-related issues.

Dr. Dan Sulzbach, Executive Director of SDSC, has given, in the past, a week of lectures on computer ethics, which stimulated some very interesting class discussions.

Dr. Sulzbach began his talk by stating:

"I am not a philosopher, not an ethicist, not an expert. I am a computer professional like you. I'm not here to preach because I have no license or

authority to do that. I hope only to raise some issues related to computer ethics. I will undoubtedly ask more questions than I answer."

Computer accounts on the Cray C90 are distributed only after the week-long discussions on ethics.

6 Writing

Most of the students have no problem with the computer ethics essay, but they typically have had little experience in writing up a science-oriented programming report. Therefore, the assigned programming project consists of the following components:

a. Solve a problem on a campus mainframe and document the mainframe's performance.

b. Solve the same problem on the Cray.

c. Compute an enhancement of the problem on the Cray and document the Cray's performance.

Instructor feedback after reading the document from assignment (a) greatly enhanced the organization and content of the final document in (c).

Main Project

Science-Oriented Program to be run on SDSU Mainframe and subsequently on the CRAY

The main project is crucial to this course. Students will use the Cray to run this project, and the instructor does not want them to squander Cray resources before they became familiar with their problems. Students are allowed to pick from a selection of problems provided by the instructor, or they could solve a scientific Cproblem from their particular backgrounds or work environments.

Good sources for problem statements were:

"Computing Applications to Differential Equations: Modelling in the Physical and Social Sciences," by J.M.A. Danby; Reston Publishing

"Numerical Methods and Software" by Kahaner, Moler and Nash; Prentice-Hall Publishers.

"Computational Physics" by Koonin and Meredith; Addison Wesley Publishing Company.

The MAIN PROJECT assignment:

a) Get your "science" program running on the SDSU mainframe.

b) Write a report describing your problem and your program's performance on the SDSU mainframe.

c) Get your program running on the Cray.

d) Extend your problem. For example, use a finer grid spacing or use more species in an interaction. (This will depend on your particular problem.)

e) Submit a final report on your Cray project.

<u>Topics to be covered in your report:</u>

a) Your write-up should have a self-contained statement of the problem. The reader should not have to read your code to find out what equations you are working with, or what the specific problem is that you are solving.

b) Give a complete reference to where the problem came from.

c) Define your measure of work so that comparisons can be made when you run on the Cray. You can't talk about "faster" or "better" without a specific measure of performance.

d) Discuss what conclusions are drawn from the problem itself. What is the "science" story revealed by the original problem? Why was this problem solved.

e) You should carefully organize the results. A summary of pertinent results for both the "science" of the problem and the "performance" of the program should be presented. Optionally, include an appendix for more detailed results.

Appendix: Anonymous FTP Access to Course Notes

Feb. 3, 1994

Kris Stewart, Assoc. Prof., Dept. Math. Sciences, SDSU
Computational Science Curriculum
Coordinator, SDSC

SUE = Supercomputing and Undergraduate Education
SDSU = San Diego State University
SDSC = San Diego Supercomputer Center

As described in the article "SUE workshop" in Vol. 8, No. 11-12 (Nov.-Dec. 1992) of Gather/Scatter from SDSC, the lecture notes from the SUE workshop offerred last summer at SDSC are available to the public. You can access them through the Internet by following the following directions

ftp ftp.sdsc.edu (establish ftp contact with SDSC machine)

You will be prompted for the standard ftp logon. You need to identify yourself as "anonymous" and enter your complete email address when prompted for a password.

Name: anonymous
Password: type your complete email address here

NOTE - every directory level has a README file summarizing the contents in that directory

cd pub (change directory)
cd sdsc (change directory)
cd docs_and_education (change directory)
cd education (change directory)
get README (you can later examine this file for more details on the organization of this info)

get super92.asc (this is an ascii copy of Stewart's Poster Session on SUE at Supercomputing '92)

You may want to just stop here. There is a lot more information available. I feel the files in the directory "SUE-notes" are in adequately good shape. If you would like to continue to retrieve more documents

cd SUE-notes (change directory)

get README (this file gives an overview)

Again, you may want to stop here and look at the somewhat lengthy file README in the SUE-notes directory. If you want to get it all:

prompt (turn off the automatic prompting)
mget * (get all the files in the directory SUE-notes)
bye (terminate ftp session)

This will copy a large amount of information to your own account on your home computer. You may edit the files at well, but you should give credit to their original source.

To make things simpler, I'm including the README file mentioned above in this mailing. -kris
----
README in pub/sdsc/docs_and_education/education available via anonymous ftp to ftp.sdsc.edu

NAME

supercomputers_in_education Information on using SDSC computers for courses

super92.asc ASCII document of Dr. Kris Stewart's handout in conjunction with her poster session at Supercomputing '92 in Minneapolis. This gives an overview of the SUE activities at SDSC as well as descriptions of educationally related files available in the following directories:

STEP-notes A directory of files from the Supercomputer Teacher Enhancement Program (STEP).

SUE-notes A directory of files from Dr. Kris Stewart's SUE faculty workshop offerred 1991 and 1992 at SDSC - THESE ARE MOST UP TO DATE - USE FIRST. SUE stands for Supercomputing and Undergraduate Education.

Supercomputing-Course-Notes A directory of lecture notes from Dr. Kris Stewart's course CS 575 Supercomputing for the Sciences taught at SDSU in 1991 and 1992. This is an interdisciplinary course for students with a good programming background in Fortran or C focussed primarily on using the Cray Y-MP 8/624 at SDSC. CAVEAT - the files in this directory are not in as good of shape as those in the directory SUE-notes. I recommand the files in the SUE-notes directory more than those in this directory. Examine these with caution.

DESCRIPTION pub/sdsc/docs_and/education/education is a subdirectory of SDSC's anonymous ftp archive. It contains education materials.

CONTACT SDSC Consultants (consult@sdsc.edu) (619) 534-5100

ADDITIONAL INFORMATION For additional information about SDSC's anonymous ftp, see the files in the help directory.