Sample Syllabus for GEOS 436/636

The remainder of this file based on the actual syllabus for the current trial version of the course, which is being offered as GEOS 692. The trial course was developed collaboratively by Freymueller and graduate student Ronni Grapenthin – in fact, the course was Grapenthin's brainchild. The results of the trial courses have been significant enough that we aim to integrate it into the curriculum as a formal course with a permanent instructor. More information about the course is available online:

http://www.gps.alaska.edu/programming/

The 2010 version of the trial course is also online:

http://www.gps.alaska.edu/programming_2010/

That version includes the lecture presentations in PDF format for the entire semester, as they were delivered in Fall 2010. A 2009 version is also available. Some selected material is included in PDF form. The trial course was not stacked and was offered pass/fail. One undergraduate took it in Fall 2011, which was useful in helping calibrate expectations. The syllabus here has been modified from the originally submitted version to better develop the stacked undergraduate version of the course and to incorporate feedback from the Faculty Senate. The main changes were to be more specific about the expectations for project, and to make a clearer separation between the work assigned and expectations for undergrads taking it as 436. Flexibility in the project is beneficial for the students, so keeping that is important. The students are more excited about the project and learn more when it is directly helping them do something they need to do for their research. That means there are many styles of projects, and these have been listed now in the syllabus so students can see what the expectations are. My hope is that they will do more than the minimum because that advances their research, but that is up to them. The structure of the lab and homework assignments is not conducive to having undergraduates do different or simpler assignments or only do a part of each assignment, and in reality there is no reason they should not be able to do these as well as the grad students. So the main difference is that the undergrads have a much simpler final assignment, which is just to present something they have done in the lab rather than to do independent work like the graduate students are expected to do.

Why a stacked course?

This course was originally conceived as a course to teach skills needed by graduate students to do research in today's scientific world. We have found that an increasing number of our graduate students lack these skills. There is a clear trend in our applicant pool over the years. It appears that students in geosciences do not learn computer programming unless they once intended to be in computer science, and then changed over to geosciences. In fact, most incoming students today have never interacted with a computer except through a modern graphical interface, and we find that some students have never even thought about giving commands to a computer in any other way. Reaction to the article Ronni Grapenthin and I published in the American Geophysical Union's newspaper (attached) shows that this is not specific to geosciences or to UAF.

If our incoming grad students are like this, then the same probably applies to our graduating seniors. This prompted the decision to stack the course. The intention is to offer the course to seniors with the goal of making them more competitive in terms of their skill set if they intend to move on to graduate school or do some other form of research after graduation (or before). The pre-requisite of senior standing should accomplish this, and avoid mixing graduate students with much younger, less mature and less serious undergrads. Undergraduates involved in or eager to do undergraduate research would also be good candidates, and could be added with permission of instructor if they were not yet seniors, assuming space is available.

GEOS 692: Beyond the Mouse 2011 – Programming and Automation for Geoscientists (2 Cr.)

"Programming is legitimate and necessary academic endeavor."
Donald E. Knuth

Instructors:

Ronni Grapenthin
Geophysical Institute
University of Alaska Fairbanks
903 Koyukuk Drive, P.O. Box 757320
Fairbanks, Alaska 99775-7320
office: Elvey 413H
hours: by appointment
email: ronni@gi.alaska.edu
phone: +1 (907) 474 – 7428

Jeff Freymueller
Geophysical Institute
University of Alaska Fairbanks
903 Koyukuk Drive, P.O. Box 757320
Fairbanks, Alaska 99775-7320
office: Elvey 413B
hours: by appointment
email: jeff.freymueller@gi.alaska.edu
phone: +1 (907) 474 – 7286

Overview:

In the (geo)sciences – as in many other disciplines – we collect data which need to be analyzed in ways that depend on the problem posed. The ability to modify your working environment according to your needs instead of having it dictate how you approach a problem is invaluable. This is especially true in a setting that is supposed to generate fresh knowledge. Also, and this may be even more important, *we are lazy people*. We do not want to waste time by repeating the same steps again and again, and ... again. Such boredom causes errors. And being bored by such routines is totally legitimate. A computer (the machine, and earlier the person) exists to perform such routines rapidly, reliably and repetitively: It takes in data, manipulates the data following *your* commands (YEAH!), and produces a result. The point of writing computer programs is to *automate an intellectual challenge that has been solved* and make it reusable at all times - for yourself and ideally for others. 21st century scientific research frequently involves manipulation or analysis of very large data sets, or the development of numerical models; this work can only be used effectively by scientists who can make software tools themselves. Accordingly, the geophysics graduate curriculum now expects students to be able to write simple computer programs. This course will teach you the basic techniques and skills to do this.

The class will meet for one lecture and one lab every week for 14 weeks.

What this course is:

The intent is to teach you how to make simple tools that will allow you to read in and massage data in exactly the way you want, and plot or visualize the results. We will start out manipulating your thinking, introduce you to programming in general, and then take off into specific working environments namely Unix/Linux and Matlab while teaching you how to map your data using GMT and create simple web pages by writing the HTML yourself. All of this is easier than you might think – you simply have to get up over the initial part of the learning curve. We will cover many things in a short amount of time, which means that we will give you many pointers that you can follow up on depending on your needs. There is a tremendous amount of reference material (and examples to adapt) available on the web. We encourage you to play with the tools we are teaching you to use beyond the course assignments, and do

things with them that are fun for you. The more you do, the more you will learn.

What it is not:

Complete.

Prerequisites:

GEOS 436: Senior standing or permission of instructor.

GEOS 636: Graduate standing.

Textbook:

No textbook exists for this course. Handouts and lecture slides will be provided, and we will guide you to some of the many reference sources available on the web.

Student Learning Outcomes:

By actively participating in this course you will become significantly more proficient at:

- Breaking problems down into a series of steps
- Organizing data and tools to make automated work easier
- Writing and understanding how to read computer programs in MATLAB
- Writing and understanding how to read Unix/Linux shell scripts
- Making publication-quality maps and figures using GMT (Generic Mapping Tools)
- Using HTML and CSS for web pages

Grading:

This 2 credit class is pass/fail. The class assignments are primarily lab exercises, specifically computer programs written in the computer lab. We use software that is available to students at no cost (for use within the UAF network), so all students could also install and use it on their own computer if they wish. The computer lab is also available for students to use at other times, if they need to finish an assignment outside of lab. During the first third of the semester, additional short homework assignments will be given outside of lab (these do not require any particular computer or software).

Grading is based on weekly lab exercises, homework assignments, a final project, and the presentation of that project in the form of a web page or pages. There will be a total of 12 graded lab assignments, equally weighted, and all other assignments except for the final project itself are scored points equivalent to a lab assignment or a fraction of that.

Graduate Students

Labs+Homework+Project Presentation	**70% of total**
Each Lab assignment	1 Lab
Each Homework assignment	1/2 Lab
Final Project Presentation	1 Lab

Final Project	**30% of total**
Passing (graduate)	***>= 65%***

Undergraduate Students

Labs+Homework+Final Presentation	**100% of total**
Each Lab assignment	1 Lab
Each Homework assignment	1/2 Lab
Final Presentation	1 Lab
Passing (undergraduate)	***>= 65%***

Attendance and activity in class will be taken into consideration to raise the grade by small amounts, if necessary and justified.

The homework and lab exercises consist of basic application of methods and practices presented in class. The labs help you apply things taught in class. The complexity of the labs varies. Usually they consist of a simple introduction problem to get you used to the environment, understand new commands, etc. In a second part you will apply this in a slightly more complex way to data, or simply write more complex code.

The final project will (hopefully) be specific to your research project. We want to encourage you to set up an efficient and safe environment in which you apply the methods and tools introduced in class.

Graduate students are expected to carry out a complete project within their own field of specialization (this can and should be something that helps them in their own research). The project will be presented in the form of a web page or pages, for which the student will write the HTML using the templates provided in class and used in one of the labs. Undergraduates will substitute a presentation of some of their own work from the labs in place of an independent project, also presented in the form of a web page or pages.

There are several styles of project that a student could take on, depending on their needs. Flexibility in this regard is beneficial for the students, as they learn more by doing more, and do more when they are excited about and see the relevance of the project. The project must be implemented in code using one of the tools used in the class (or a different tool with instructor permission). The students must turn in complete code, raw data files, etc, so that the instructor could run their code and replicate their results. Code must be adequately commented. All of the code and data files should be linked on the web page or pages.

Sample projects include one of these, at a minimum: (a) reading in data and doing useful manipulation and visualization of the data; (b) constructing a coherent suite of scientific figures or visualizations of data; (c) developing and running a numerical model; (d) writing a program or programs to automate a task that must be done repeatedly (for example, a data processing or analysis task), and using this program to run a substantial amount of data.

(*Graduate students only*) In the beginning of the semester you will provide us with a snapshot of your project directory (If you have one). Send rudimentary data files, and any scripts/programs should be executable. You will do the same at the end of the term through your final project, and tell us how you improved or changed the organization to make working with your data easier to automate. If your project involved doing something totally new, you

will tell us why you chose to organize things as you did.

Policies and makeup-labs:
You are subject to the UAF Student Code of Conduct (http://www.uaf.edu/catalog/current/academics/regs3.html). We will work with the Office of Disabilities Services (203 WHIT, 474-5655) to provide reasonable accommodation to student with disabilities. Makeup versions of labs will be provided if we have a convincing reason to do so. The makeup must occur prior to final project presentations.

Schedule:

The class meets: Mon (lecture+lab) + Tues (lab) 3:30-5:30 pm in REICH 316.

Sep 08	Introduction	Jeff Freymueller , Ronni Grapenthin
Sep 12,13	Lecture 1: Thinking Programs	Ronni Grapenthin
	Lab 1: Organizing your ideas	
Sep 19,20	Lecture 2: Fundamental Programming Principles I: Variables and Data Types	Ronni Grapenthin
	Lab 2: Matlab and Variables	
Sep 26,27	Lecture 3: Matlab I: (Advanced) Variables and functions	Jeff Freymueller
	Lab 3: Matlab structs and functions	
Oct 03,04	Lecture 4: Fundamental Programming Principles II: Control Structures	Ronni Grapenthin
	Lab 4: Matlab flow control	
Oct 10,11	Lecture 5: Matlab I/O I	Ronni Grapenthin
	Lab 5: Matlab I/O I (files)	
Oct 17,18	Lecture 6: Matlab I/O II	Ronni Grapenthin
	Lab 6: Matlab I/O II (plotting)	
Oct 24,25	Lecture 7: Unix Tools I	Jeff Freymueller
	Lab 7: Unix Tools	
Oct 31, Nov 01	Lecture 8: Unix Tools II	Jeff Freymueller
	Lab 8: Unix Tools	
Nov 07,08	Lecture 9: Live Shell Scripting	Ronni Grapenthin
	Lab 9: Unix Tools	
Nov 14,15	Lecture 10: Debugging	Ronni Grapenthin
	Lab 10: Debugging	
Nov 21,22	Lecture 11: GMT I	Bernie Coakley
	Lab 11: GMT – Data mapping	
Nov 28,29	Lecture 12: GMT II	Bernie Coakley
	Lab 12: GMT – Data mapping	

Dec 5-12	Independent Study: HTML Lab 13: Setting up a website for project presentation	Ronni Grapenthin

Prior to each lecture you will find handouts, examples, and problem sets here. The problem sets are supposed to get you started poking around on your system and/or change the way you approach problems. The handouts will form some sort of mini-handbook that could be placed next to your computer.

Mailing List:
To discuss issues with labs, projects and general programming issues with your fellow students, we set up the mailing list:
btm2011@gi.alaska.edu
Please sign up at http://dogbert.gi.alaska.edu/mailman/listinfo/btm2011 and use this list first to ask your questions.

Notes:
If you do not have access to a unix-linux-mac environment, we recommend that you install a similar setup on your own computer, like we'll have in the lab. We will also have a Linux-based computer that you can access remotely, and post your project webpages on. We will use virtualbox as a virtualization software which allows to run, say, a linux distribution within a running Windows (no rebooting required). Once virtualbox is installed you need to put a linux distribution of your choice (maybe ubuntu) on top of this. See Ronni (ronni <at> gi <dot> alaska <dot> edu) if you need help with that.