Mathematical & Computational Physics II

The material presented in this course is drawn from a variety of sources, including:

-*Numerical Methods for Physics*, Garcia 2nd ed. (2000)
-*Computational Physics*, Koonin & Meredith (1998)
-*Numerical Recipes*, Press et al. 3rd ed. (2007)
-*Applied Numerical Analysis*, Gerald & Wheatley 7th ed. (2003)
-*Computational Physics*, Giordano & Nakanishi 2nd ed. (2006)

A few reference texts are located in PS 227. They are *not* to be removed:

-*Mastering MATLAB 6*, Hanselman & Littlefield
-*MATLAB Programming for Engineers*, Chapman
-*MATLAB An Introduction with Applications*, Gilat
-*Computational Physics*, Koonin & Meredith
-*Computational Physics*, Giordano & Nakanishi

The text provides codes for *MATLAB*, *C++*, and *FORTRAN*. You are free to use another language (e.g., *Basic*, *Pascal*, *Python*, *Perl*, *IDL*, etc.), but you will have to work more independently and create your own codes from scratch.

The basic elements to most computer programs are:

declaring and initializing parameters
creating or reading in data
manipulating the data
outputting the results (e.g., datafile, postscript figure, computer monitor)

Most languages allow you to work with multi-dimensional *arrays* or *matrices*.

Most languages allow you to break up the computations into small, manageable components, but the exact details depend on the structure of the language chosen. For example, in *Fortran*, evaluating a Bessel function for a particular argument could be done with a *function*, whereas you would more likely carry out a series of Bessel function evaluations within a *subroutine* (a *procedure* in *Pascal* or *IDL* parlance). *Parameters* or *arguments* are passed into functions and subroutines, and their values returned to the *main program* may change upon execution.

Some programs are *object-oriented*, which "encourages you to design programs around data types and data-type hierarchies that you define yourself" while others are procedure-oriented "which encourage you to think in terms of procedures" (Winston 1994). Object-oriented (or *structure-oriented*) programming concentrates more on self-contained data structures that can be manipulated, and less on a series of tasks.

FORTRAN

One of the more popular programming languages in the astro/physics community. It comes in two flavors, *Fortran 77* and *Fortran 90*, circa 1977 and 1990, respectively. *Fortran 90* contains all of the features of *Fortran 77*, plus more. Array processing is easier (obviating the need for many 'do loops') and data structures are now available, for example. See Chapter 0 of *Fortran 90 Handbook* for a complete description of the differences in the two versions. Nonetheless, *Fortran 77* remains quite powerful and, due to inertia, is perhaps still more widely-used than *Fortran 90* (including by yours truly).

C++

C++ is related to the *C* programming language, which, as described in the introductory remarks to Winston (1994), became popular because *C* programs are fast to compile and execute, and because they are concise. *C++* is the most widely-used object-oriented language today. It is quite popular in engineering.

MATLAB

MATLAB is also quite popular in engineering, and it has useful graphics options built into the language. *MATLAB* is built around the use of matrices and vectors (arrays). A GUI-based version of *MATLAB* is available on UW's PCs; a command-line version is available on our departmental PAPC Linux network.

Assembly language

A low-level, symbolic language that is specific to each computer's architecture. Little-used today, except for certain instances where direct communication with hardware is necessary.

Useful miscellanea:

Programming languages evolve, so you shouldn't stress too much over the language you choose to use for this course. The most important thing to carry away from this class is the knowledge of how to create and modify scientific programs to solve physics problems, not the syntax of a particular language.

Regardless of the programming language you choose for this course, you may use the PAPC Linux computers to construct plots with *MATLAB*, *SM*, or *IDL*.

Fortran 77 code begins in column 7 and must end by column 72. If a line of code must extend to another line, this can be done through the use of ":" in column 6 of the next line. *Fortran 90* is more free-form.

Array indices start from 0 in *C++* and *IDL*, and from 1 in *MATLAB* and *Fortran*.

Comments can start with //, !, and %, in *C++*, *Fortran*, and *MATLAB*, respectively.

Miscellanea (continued)
MATLAB is picky about naming conventions. Don't start your filename with a number, and don't use a dash or period before ".m".

Ending a statement in MATLAB with ";" causes it to be executed. Leaving off the semi-colon executes the statement as well as printing the intermediate result. This is good for debugging, disastrous for statements within repetitive loops.

Programming Basics
You first need to open a text editor (e.g., *vi*, *emacs*, *nedit*, ...), into which you will type the code. *MATLAB* in its GUI-based format can be used with its own version of a text editor.

Comment, comment, comment! Commenting is an important aspect of writing computer code, as anyone who looks at your work (including yourself a few weeks later) will not decipher it without helpful descriptions liberally sprinkled throughout.

Scientific programs usually involve initializing parameters, reading in or creating data, processing/analyzing the data, and outputting the results.

Homework organization on the PAPC Linux network

	open a terminal (right mouse click on desktop)
mkdir hw1	make HW #1 directory
cd hw1	change to appropriate directory
nedit hw1_1.m &	type in your code in this file
nedit hw1_1.output.txt &	save your output in this file
matlab	
hw1_1	run your code
	save any figures as .jpg or .ps (not .fig)
	for multiple plots, use either *subplot* or *figure* before each *plot*
*tar cvf your_name.hw1.tar **	bundle all your HW #1 work into a *tar file*
	Turn in a sheet with each problem enumerated. Include handwritten solutions/explanations.