

# COMPUTING FOR SCIENCE AND ENGINEERING

COPYRIGHT ©2009 Robert D. Skeel. All Rights Reserved.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Nature of Computational Science and Engineering	1
1.2	Using Unix	2
1.2.1	Shell scripts	4
1.2.2	Unix tools	5
1.3	Installation of Software	5
1.3.1	Remote access and file transfer	6
1.3.2	Unix, C compiler, OpenMP, and MPI	6
1.3.3	Python, NumPy, and MPI for Python	7
<b>2</b>	<b>Files and Text Processing</b>	<b>10</b>
2.1	Scripting Languages	10
2.1.1	Python	10
2.1.2	Other scripting languages	10
2.2	Using Python	11
2.2.1	Doing computations	11
2.2.2	Developing a script	11
2.2.3	Developing a module	12
2.2.4	Documentation	13
2.2.5	The <code>os</code> module	13
2.2.6	The <code>subprocess</code> module	13
2.3	Files and I/O	14
2.4	Regular Expressions	16
2.5	Data	18
2.5.1	XML	18
2.5.2	Relational databases	18
<b>3</b>	<b>Mathematical Computing</b>	<b>19</b>
3.1	Computing with Python	19
3.1.1	Control structures	19
3.1.2	Variables and data	20
3.1.3	Functions	22

3.1.4	Quality assurance and verification	24
3.1.5	Debugging	25
3.2	Floating-point Computation	26
3.2.1	Floating-point numbers	26
3.2.2	Rounding	27
3.2.3	Floating-point operations	28
3.2.4	Other types of arithmetic	28
3.3	Pseudo-Random Numbers	28
3.4	Graphics	29
3.5	Differentiating Computer Code	30
3.6	Computer Algebra	30
3.6.1	Python	30
3.6.2	<i>Mathematica</i>	30
3.6.3	Maple and Matlab	31
3.7	NumPy	31
3.7.1	Array constructors	31
3.7.2	Indexing	32
3.7.3	Array operations	32
3.8	Other Tools	33
3.8.1	SciPy	33
3.8.2	Matlab and Octave	33
3.8.3	R	33
3.8.4	Fortran, C, and C++	34
<b>4</b>	<b>C Programming</b>	<b>35</b>
4.1	Pointers, Arrays, and Memory Management	36
4.2	Parameter Passing	37
4.3	Scope	38
4.4	Structures	39
4.4.1	Abstract data types	40
4.4.2	Data structures	42
4.4.3	C shortcuts	43
4.5	Compiling and Debugging	44
4.6	Interfacing Python to Compiled Code	44
4.6.1	Creating dynamically linked libraries	45
4.6.2	The <code>ctypes</code> Python module	46
4.6.3	The Numpy <code>ctypes</code> attribute	47
<b>5</b>	<b>Parallel Computing</b>	<b>48</b>
5.1	Computer Organization	48
5.2	Distributed Memory Programming Paradigm	49
5.2.1	Point-to-point operations	49
5.2.2	Collective operations	50

5.2.3	* MPI with Python	51
5.2.4	Grid computing and Condor	51
5.3	Computing on Radon and PBS Job Submission	52
5.4	Parallel Algorithms	53
5.4.1	Divide and conquer	54
5.5	Shared Memory Programming Paradigm	54
5.5.1	POSIX Threads	54
5.5.2	OpenMP	55
5.5.3	General purpose computing on graphics processing units	56
5.5.4	Unified Parallel C (UPC)	57
<b>6</b>	<b>Developing Software</b>	<b>58</b>
6.1	Object-oriented Programming	58
6.2	GUI Programming	60
6.2.1	Web interfaces	61
6.3	Software Engineering	61
6.3.1	Version control	61
6.3.2	Automated builds	62
6.3.3	Coding standards	63
6.3.4	Documentation	63
6.3.5	Development process	63
6.3.6	Integrated development environments	63
<b>7</b>	<b>Error</b>	<b>65</b>
7.1	Approximation	65
7.2	Error Propagation	67
7.2.1	Addition/subtraction	67
7.3	Computational Error	68
7.3.1	Rounding error	68
7.3.2	Statistical error	69
7.4	Algorithms and Numerical Stability	69
<b>8</b>	<b>Functions</b>	<b>72</b>
8.1	Polynomial approximation	72
8.1.1	Polynomial representation	72
8.1.2	Existence and uniqueness	73
8.1.3	Lagrange form	73
8.1.4	Interpolation	74
8.1.5	Bivariate linear polynomials	74
8.2	Piecewise Polynomial Approximation	75
8.2.1	Piecewise polynomials	76
8.2.2	Multivariate piecewise linear polynomials	76
8.3	Trigonometric Approximation	77

<b>9</b>	<b>Matrices</b>	<b>80</b>
9.1	Matrix Computations	80
9.1.1	Partitioned matrices	80
9.1.2	Structured matrices	81
9.1.3	NumPy matrix operations	81
9.1.4	Efficient matrix operations	82
9.2	Triangular Factorizations	82
9.2.1	Division by a triangular matrix	83
9.2.2	LU factorization	84
9.2.3	Symmetric matrices	86
9.2.4	NumPy and LAPACK functions	86
9.3	Orthogonal Factorizations	88
9.3.1	Full and reduced QR factorizations	88
9.3.2	Singular value decomposition	89
9.3.3	NumPy and LAPACK functions	90
9.4	Spectral Factorizations	91
9.4.1	General matrices	92
9.4.2	Symmetric matrices	93
9.4.3	NumPy and LAPACK functions	94
9.5	Parallel Algorithms	94
9.5.1	Divide and conquer	94
9.5.2	Relaxation methods and reordering	95

# Preface

The purpose of the course is to expose students to computational concepts, tools, and skills useful for research in computational science and engineering, beyond what is learned in a first programming course (and basic mathematics courses).

Ideally computational science and engineering (CS&E) is practiced by using convenient input devices to define (a model of) physical reality and the questions that are being asked. Results are displayed in a way that most effectively conveys the information—usually graphically.

Even in the best cases, application-oriented software and environments require computational skills from the user. The user will have to write scripts for the application and for job control. He(/she) will have to choose algorithm parameters and make judgments about the accuracy of the results.

In less than best cases, the computational scientist will have to do some software development, even if “only” to modify the code. In the worst case, he(/she) may have to develop algorithms.

Prerequisites for this course are

- mathematical knowledge and maturity of an MS student in the physical sciences. (It is not assumed that the student is in the physical sciences, only that he(/she) has this level of mathematics.) In particular, some familiarity with matrix algebra and elementary probability is expected.
- programming experience in C, C++, Java, or Fortran or extensive scripting experience; also, commensurate computer skills.

There is no suitably comprehensive textbook; hence these notes with their links to web resources. A good reference is [Software Carpentry](#). Some useful books:

- Python in a Nutshell, by Alex Martelli, O’Reilly, March 2003, ISBN: 0-596-00188-6,
- Learning Python, 2nd Edition, by Mark Lutz, and David Ascher, O’Reilly, December 2003, ISBN: 0-596-00281-5,
- Python Scripting for Computational Science, 3rd Edition, 2nd Printing  
by Hans Petter Langtangen, Springer, 2008,
- The C Programming Language, 2nd Edition, by Brian W. Kernighan and Dennis M. Ritchie, Prentice-Hall, 1988, ISBN 0-13-110362-8,

- An Introduction to Parallel Computing: Design and Analysis of Algorithms, 2nd Edition, by Ananth Grama, George Karypis, Vipin Kumar, and Anshul Gupta, Addison-Wesley, 2003, ISBN: 0-201-64865-2.

The course outline is

*Theme I: scripting to get things done*

1. introduction to CS&E, Unix, software installation
2. files and text processing in Python, including regular expressions
3. mathematical computing in Python, including floating-point arithmetic, graphics, Numpy, computer algebra

*Theme II: programming for performance*

4. C programming, including data structures, interfacing to Python
5. parallel programming, including computer organization, PBS, MPI, OpenMP

*Theme III: developing software*

6. developing software: object-oriented programming, GUIs, version control, automated builds

*Theme IV: numerical computing*

7. error: data error, error propagation, computational error, algorithms, numerical stability
8. functions: polynomials, piecewise polynomials, trigonometric polynomials
9. matrices: operations, factorizations, parallelism

Theme IV is to be covered concurrently with Themes I–III as a source of applications.