

Numerical methods for physicists

Logistics

- Course Web page:
www.pha.jhu.edu/~neufeld/numerical
– login: Student; password: Raphson
- Lecture schedule
Th – F 9 – 10:20 a.m. in Bloomberg 278

Contact information

Instructor: David Neufeld

509 Bloomberg

410-516-8582

neufeld@pha.jhu.edu

Office hours: Fridays 1:30 – 2:30 PM, or by appointment

TA: Divya Singh

346 Bloomberg

410-516-6694 / 301-975-5660

divyas@pha.jhu.edu

Office Hours: Tuesdays 2:30 – 3:30 PM, or by appointment

Assignments

There will be problem sets roughly every week.

Solutions will be posted on the course website soon after they are turned in.

Many (most) of the problems will be made much easier by incorporating routines from *Numerical Recipes*; a complete set [in Fortran, ANSI C, KR C (original C dialect), and C++] can be found at <eta:/usr/site/numerical-recipes>.

Grades will be based entirely on the problem sets; there will be no exams.

Ground rules for Problem Sets

- Unlimited discussion with fellow students is allowed
- But, your actual writing of the solutions, and your writing of the supporting computer code, must be done entirely independently

Late homework policy

- Homeworks will typically be due, in class, on a Friday.
- Late homeworks will be accepted – for reduced credit – up to one week after the due date.
- Homeworks more than 7 days late will not be accepted.
- Exceptions to all the above: extensions granted – IN ADVANCE – for medical reasons or because of research-related or emergency travel

Ethics

- The strength of the university depends on academic and personal integrity. In this course, you must be honest and truthful. Ethical violations include cheating on exams, plagiarism, reuse of assignments, improper use of the Internet and electronic devices, unauthorized collaboration, alteration of graded assignments, forgery and falsification, lying, facilitating academic dishonesty, and unfair competition.
- Report any violations you witness to the instructor. You may consult the associate dean of students and/or the chairman of the Ethics Board beforehand. See the guide on "Academic Ethics for Undergraduates".

Course overview

- **Goal:** to learn how to apply numerical methods to a variety of physical problems
- **Background:**
 - Many canned routines are available (e.g. Mathematica), and you can try to use them without understanding how they work

BUT.....

Course overview

- You really need to know the strengths and limitations of the various methods in order to make an informed choice
- Many numerical problems have pathologies: you need to learn how to recognize them
- You may need to improve the canned routines (better precision, or faster execution)
- Some methods are not widely available as canned routines (e.g. Monte Carlo simulations)
- The title “Numerical Recipes” is perhaps poorly-chosen

Syllabus

Preliminaries	Chap 1
Linear Equations	Chap 2
Integration	Chap 4
Random Numbers	Chap 7
Root Finding	Chap 9
Minimization	Chap10
Eigenvalues/Eigenvectors	Chap 11
Fourier Transforms	Chap 12
Ordinary Diff. Eqs	Chap 16
Partial Diff. Eqs	Chap 19
Statistical Methods	Chaps 14+15 + notes
Neural Networks	notes

Lecture 1: Preliminaries

(see “Recipes” Chapter 1)

Analytic techniques have been discussed in the previous semester of this course

- **Advantages of analytic methods:**
 - Behavior of a physical system described by finite set of mathematical expressions
 - Limiting behaviors easily studied: important for making a “sanity check” and in understanding the underlying physics
 - Expressions often make behavior easy to visualize

Numerical vs. analytic methods

- **Disadvantages of analytic methods**

- Many problems (most real-life problems) are simply not soluble by analytical methods, especially

- for complex systems involving several interacting physical processes

"God does not care about our mathematical difficulties. He integrates empirically." – A. Einstein

- when comparison is made with real observational data (with instrumental effects, selection effects)

Numerical vs. analytic methods

- **Disadvantages of analytic methods**

- Stochastic processes are not well described by analytic methods

- Average properties can often be derived, but not observables that involve correlations between degrees of freedom

- Has led to the development of a great many techniques (“Monte Carlo” simulations)

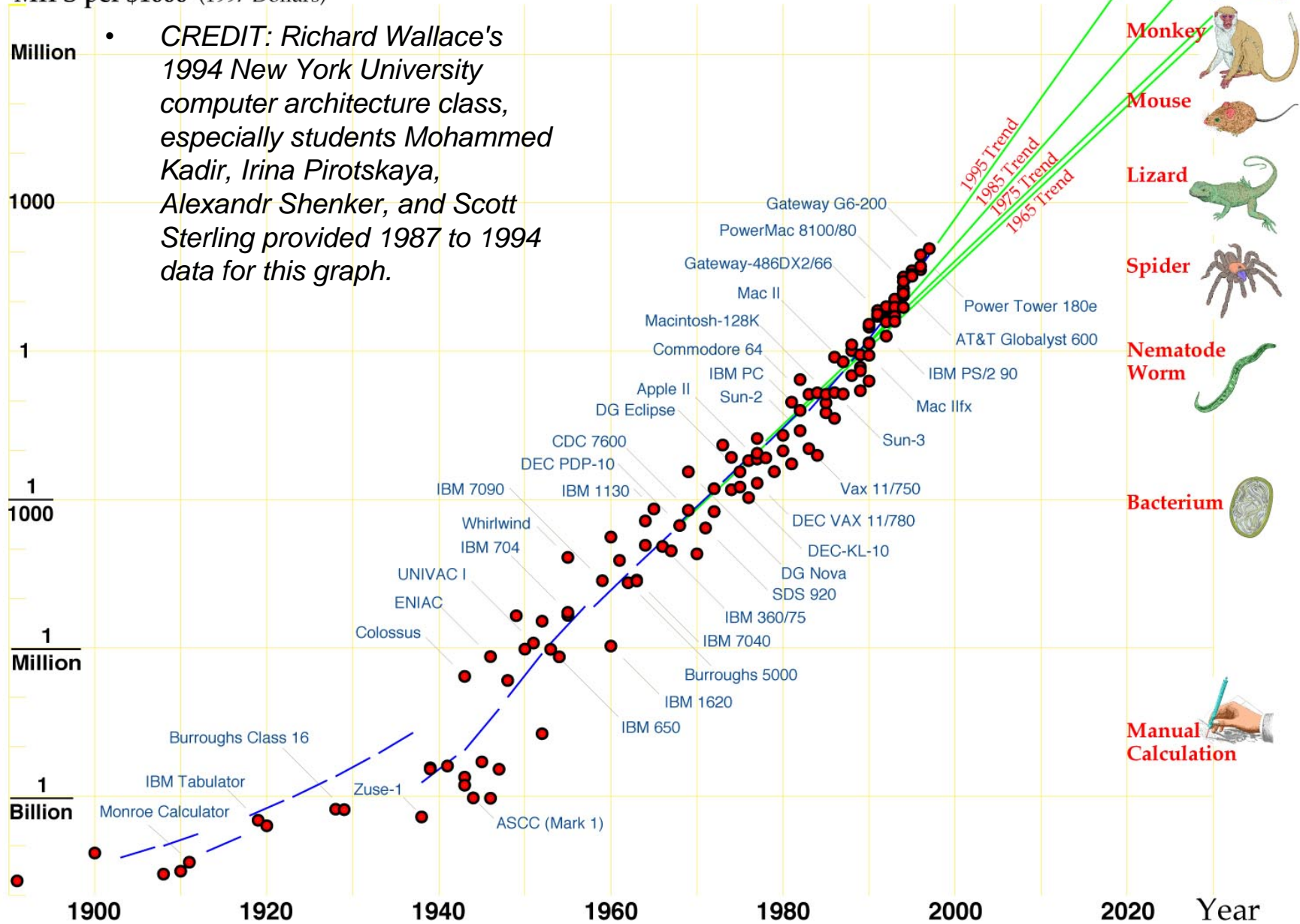
The microprocessor revolution

- Numerical methods have been revolutionized by advances in microprocessors
- The following plot appeared in
 - ROBOT: Mere Machine to Transcendent: by Hans Moravec (Oxford University Press, 1998)

Evolution of Computer Power/Cost

MIPS per \$1000 (1997 Dollars)

• *CREDIT: Richard Wallace's 1994 New York University computer architecture class, especially students Mohammed Kadir, Irina Pirotskaya, Alexandr Shenker, and Scott Sterling provided 1987 to 1994 data for this graph.*



Numerical vs. analytic methods

- **Advantages of numerical methods**
 - Can solve real problems in a “straightforward manner”
 - Very natural for stochastic processes

Numerical vs. analytic methods

- **Disadvantages of numerical methods**

- Solutions can be difficult to interpret: must often vary parameters to get a full understanding
- Limiting cases not always easy .. beware of $0 \div 0$
- Visualization must be built-in: no simple expressions to view
- Checking of results is non-trivial – bugs
- Pathological behavior: difficulties with precision, singularities, stability

Programming languages

- Numerical solutions always involve writing computer programs
- Languages:
 - Fortran: simple, old-fashioned, good at numerical computation, good handling of vectors and matrices
 - C: more powerful for strings/non-numerical data, weaker than Fortran for numerical calculations, more powerful control structures (relative to Fortran-77 but not Fortran-90)
 - C++: object oriented approach, poorer numerical performance relative to Fortran/C, industrial standard

Programming style

- Keep it simple, stupid (KISS)
- Comment extensively
 - eases debugging
 - eases reuse of the code (by others, and by you when you come back to it years later!)
- Use modules (subroutines and functions)
- Use “structured programming” approach
- Name variables appropriately and uniformly
- Avoid explicit branches (i.e. use if..then..else instead of if..goto)

Considerations beyond the scope of this course (and the book)

- In real-life computing, for many problems, a key factor is writing computer code that runs rapidly, by
 - facilitating parallel computing (“vectorizing”)
 - controlling storage (to keep the most-accessed variables in the fastest memory)
- This is really a computer science problem (that is highly hardware-dependent)

Errors and precision

- **Storage of numbers in computers**

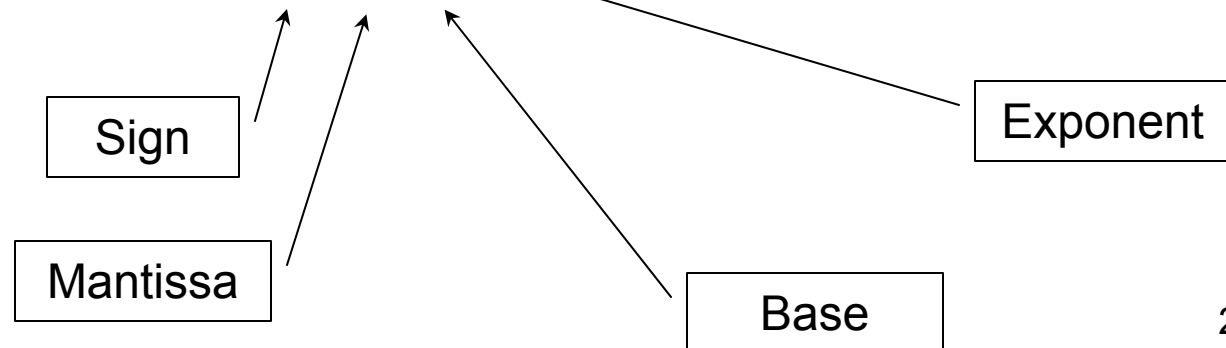
- Integers: exact, discrete

- Used for loop indices, array indices

- Floating-point ("real"): approximate, continuous

- Used for actual numerical calculations

- Represented as $S M B^e$



Floating point numbers

- **Single precision**

32 bits assigned to each number: 8 bits to e,
24 bits to M and S

Range: $2^{-127} < |R| < 2^{128}$ ($10^{-38} < |R| < 10^{38}$)

Precision: 1 part in 2^{23} (few parts in 10^8)

- **Double precision**

64 bits assigned in various ways, leading to
(for example)

Range: $10^{-76} < |R| < 10^{76}$

Precision \sim 1 part in 10^{16}

Floating point numbers

- **Quad precision**
 - 128 bits assigned
 - (available on some compilers)
- **Complex**
 - Real and imaginary parts

Roundoff error

- The finite precision of floating point numbers leads to roundoff error, a serious issue that must be minimized.
- Most problematic when numbers of very different size are added or subtracted
- In single-precision,
 $1.0 + 0.001 = 1.0010000 \quad \dots\text{BUT}\dots$
 $1.0 + 0.0000000001 = 1.00000000$

Roundoff error

- Effects can often be minimized by paying careful attention to the order of operations:
e.g. suppose we are interested in the sum of N numbers of widely-varying size, a_n . We want to add them together in increasing order of size.

$$\underbrace{1.0 + 1.0 + 1.0 + 1.0 + \dots + 1.0}_{10^9 \text{ entries}} + 1.0 \times 10^9 = 2.0 \times 10^9$$

BUT

$$1.0 \times 10^9 + \underbrace{1.0 + 1.0 + 1.0 + 1.0 + \dots + 1.0}_{10^9 \text{ entries}} = ???$$

Roundoff error

- Even so, you often encounter problems if you add $\sim 10^7$ single precision numbers
- Solution: go to double precision
- Downside: double precision additions (multiplications) may take 2 (4) times as long (depending upon the CPU)

Underflow/overflow errors

- Overflow errors: $10^{30} \times 10^{30} = \text{NaN}$
 - program will usually crash
- Underflow errors: $10^{-30} \times 10^{-30} = 0$
 - program will continue (but a good compiler should warn you what happened)

“Not a Number”

- Solutions

- rescale, or use a logarithmic scale
- go to double precision
- pay attention to the order of operations

e.g. $(6.7 \times 10^{-27} \times 6.7 \times 10^{-27}) / (9.1 \times 10^{-28}) = 0$

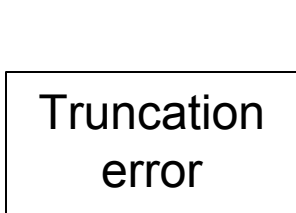
whereas $(6.6 \times 10^{-27} / 9.1 \times 10^{-28}) \times 6.6 \times 10^{-27} = 4.7... \times 10^{-27}$

Truncation errors versus roundoff errors

- Truncation errors are the errors inherent in the numerical method adopted, even on a perfect machine (with infinite precision)

Example: numerical integration using the trapezoid rule

$$\int_{x_1}^{x_2} f(x) dx = \frac{1}{2} h \{f(x_1) + f(x_2)\} + O(h^3 f'')$$



Truncation
error

Stability

- Under certain circumstances, the accumulation of roundoff errors can lead to a rapidly (e.g. exponentially-growing) overall error
- This is called instability
- Example: consider the “Golden Mean”,
 $\phi \equiv \frac{1}{2} (\sqrt{5} - 1) = 0.618\dots$,
which obeys the recurrence relation

$$\phi^{n+1} = \phi^{n-1} - \phi^n$$

Stability

- So couldn't you use the recurrence relation to obtain powers of ϕ by means of subtraction (very fast) rather than multiplication?

e.g. $\phi^2 = 1 - \phi$

$$\phi^3 = \phi - \phi^2$$

$$\phi^4 = \phi^2 - \phi^3$$

.....

Stability

- Problem: second solution to recurrence relation is $-\frac{1}{2}(\sqrt{5} + 1)$
 - This solution has absolute value > 1 , so powers of it grow exponentially
 - Any small component of this solution, introduced by roundoff error, grows exponentially (whereas the solution we want shrinks exponentially)
 - Pretty soon, the results of applying the recurrence relation are completely wrong

Summary

- Numerical methods are powerful
- Numerical methods are delicate

It's very easy to get the wrong answer!

➔ much of art of applying numerical methods to real world problems is figuring out how to check whether your program is giving the right answer