Basics of Computational Mathematics

Juan Meza
Lawrence Berkeley National Laboratory

Outline

- General Topic Areas
- What can you do / can't do
- Computer arithmetic / floating point
- Stability versus ill-conditioning
- Power of Algorithms versus Hardware

General Topic Areas

- Dense Linear Algebra
- Sparse Linear Algebra
- Spectral Methods (FFTs)
- Structured Grids
- Unstructured Grids
- N-Body Methods
- Monte Carlo
- http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

What can you do/can't do

- In general
  - Only solve to certain level of accuracy, either model or computer
  - Only solve linear systems
  - Only find local minima

Always want to consider what can go wrong

- How much can we trust the final computed answer?
- How good is the final answer?
- What's the relative accuracy of a solution?
- How sensitive is the answer?

The purpose of computing is insight, not numbers.

R.W. Hamming

Floating Point Representation

- Standard form to represent numbers is called normalized scientific notation
- Usually of the form

$$x = r * 10^m, \quad .1 \leq r < 1$$

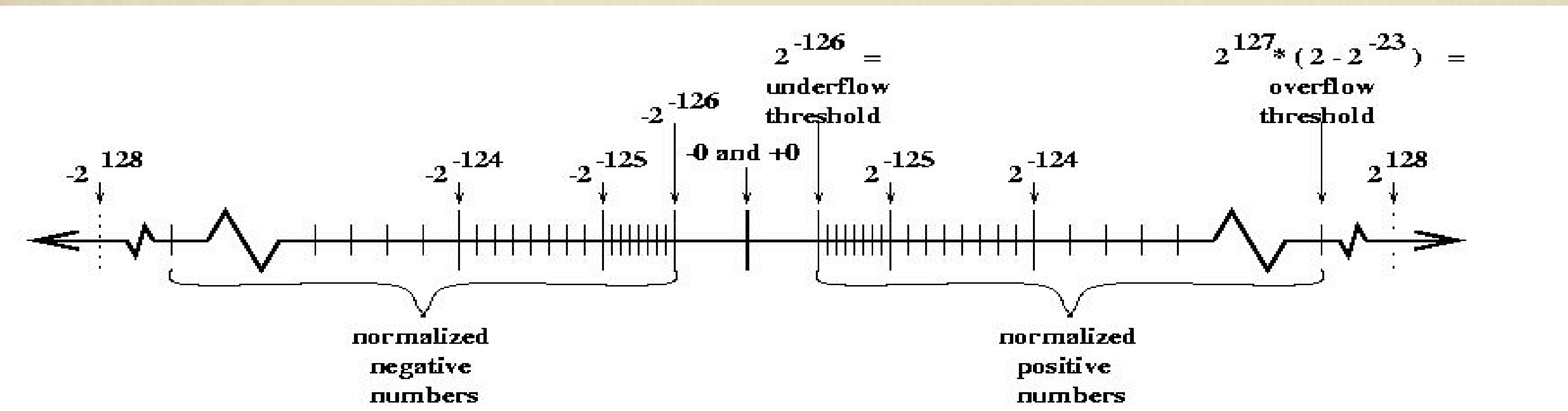
- Computers use binary numbers so

$$x = q * 2^m, \quad \frac{1}{2} \leq q < 1$$

- The set of real numbers that can be represented on a computer are called the machine numbers

IEEE Floating Point

0	1-11	12-63
SIGN	EXPONENT	MANTISSA



IEEE 64-bit Floating-Point

- Approximate decimal exponent range: 10^{-308} to 10^{308}
- Approximate decimal accuracy: 16 digits
- Largest whole number that can be represented exactly: $2^{53} = 9.0072 \times 10^{15}$

Exception Handling

- Exception handling deals with operations that either generate an invalid number, or one that is too small or too large to represent
- Some common exceptions:
  - Overflow - exact result > OV, too large to represent
  - Underflow - exact result nonzero and < UN, too small to represent
  - Divide-by-zero - nonzero/0
  - Invalid - 0/0, sqrt(-1), ...

Some Useful Facts

- Integers are exact – unless they overflow
- Every arithmetic operation is rounded off
  - too big yields overflow
  - tiny but nonzero yields underflow
- Some operations are exact, e.g. x = -y
- Machine epsilon is defined to be the Smallest floating point number such that:
  - $1 + \epsilon > 1$

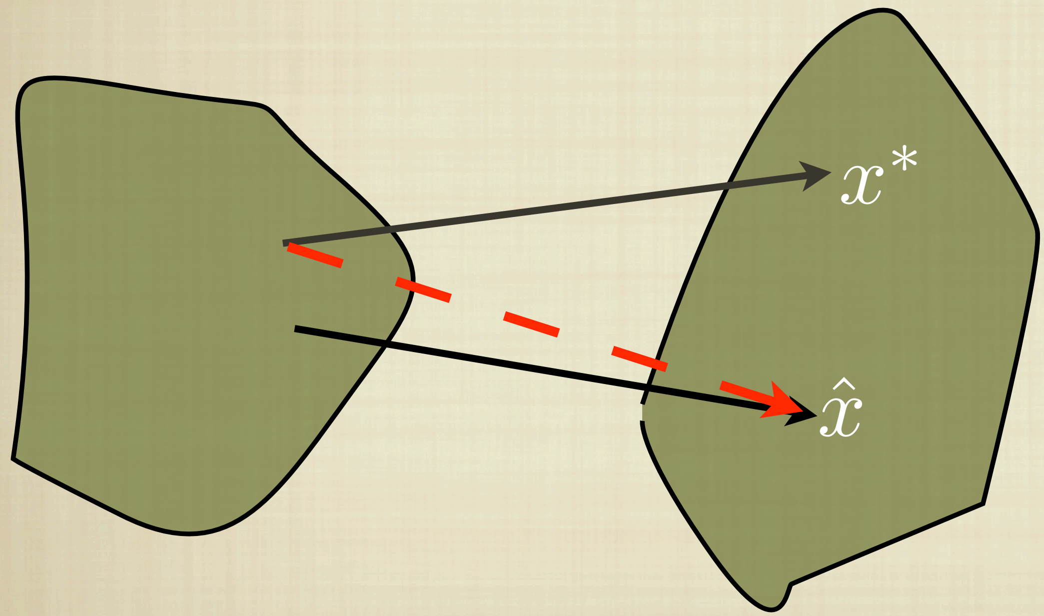
High Precision Arithmetic

- What if 64 or 80 bits is not enough?
  - Very large problems on very large machines may need more.
- High precision can be simulated efficiently using standard floating points ops.
- Each extended precision number s is represented by an array $(s_1, s_2, \ldots, s_n)$ where:
  - each s_k is a floating point number
  - $s = s_1 + s_2 + \ldots + s_n$ in exact arithmetic
  - $s_1 >> s_2 >> \ldots >> s_n$
- Much more in Wednesday's talk by David Bailey

Further References on Floating Point Arithmetic

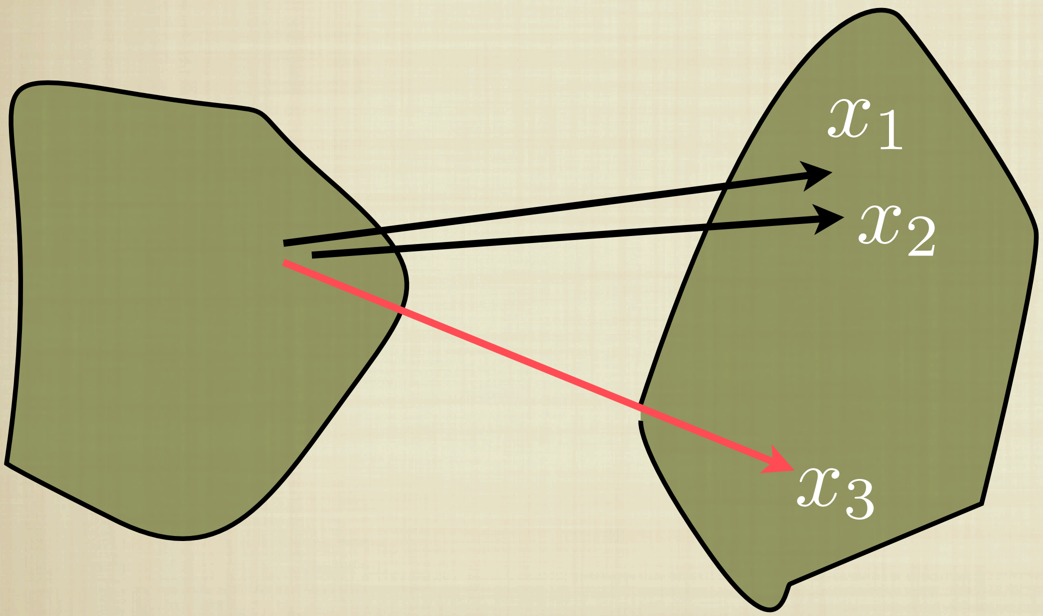
- W. Kahan
  - CS267 lecture from 1996
  - www.cs.berkeley.edu/~wkahan/ieee754status/cs267fp.ps
  - "Lecture Notes on IEEE 754"
  - www.cs.berkeley.edu/~wkahan/ieeestatus/ieee754.ps
- David Bailey's web pages
  - crd.lbl.gov/~dhbailey
- MSRI-UP wiki (look in references
  - msri-up.msri.org

What's the best that we can expect?



- The computed solution should be the exact solution to a a slightly perturbed problem
- Def: An algorithm is stable if it computes the exact solution to a nearby problem
- Recall, with computer arithmetic we're always solving perturbed

What about the relative accuracy?



- Def: A problem is ill-conditioned if small changes in the data can cause large changes in the solution.
- Need to emphasize the can and not will.

Example: Ill-conditioning

$$\begin{aligned} x + 2y &= 3 \\ 0.499x + 1.001y &= 1.5 \end{aligned}$$

- By inspection the solution is x = y = 1.0
- If we change 0.499 to .500 (a relative error of 0.001 then the solution becomes: x = 3, y = 0
- A relative change in the output of 100%
- Irrespective of the method you choose to solve this problem with

Summary: Stability vs. Ill-conditioning

- Algorithms are stable / unstable
  - An algorithm is stable if it solves a nearby problem
- Problems are well/ill conditioned
  - A problem is ill-conditioned if small changes in the input cause large changes in the output
- Many people confuse the two leading to all sorts of problems and unwise decisions

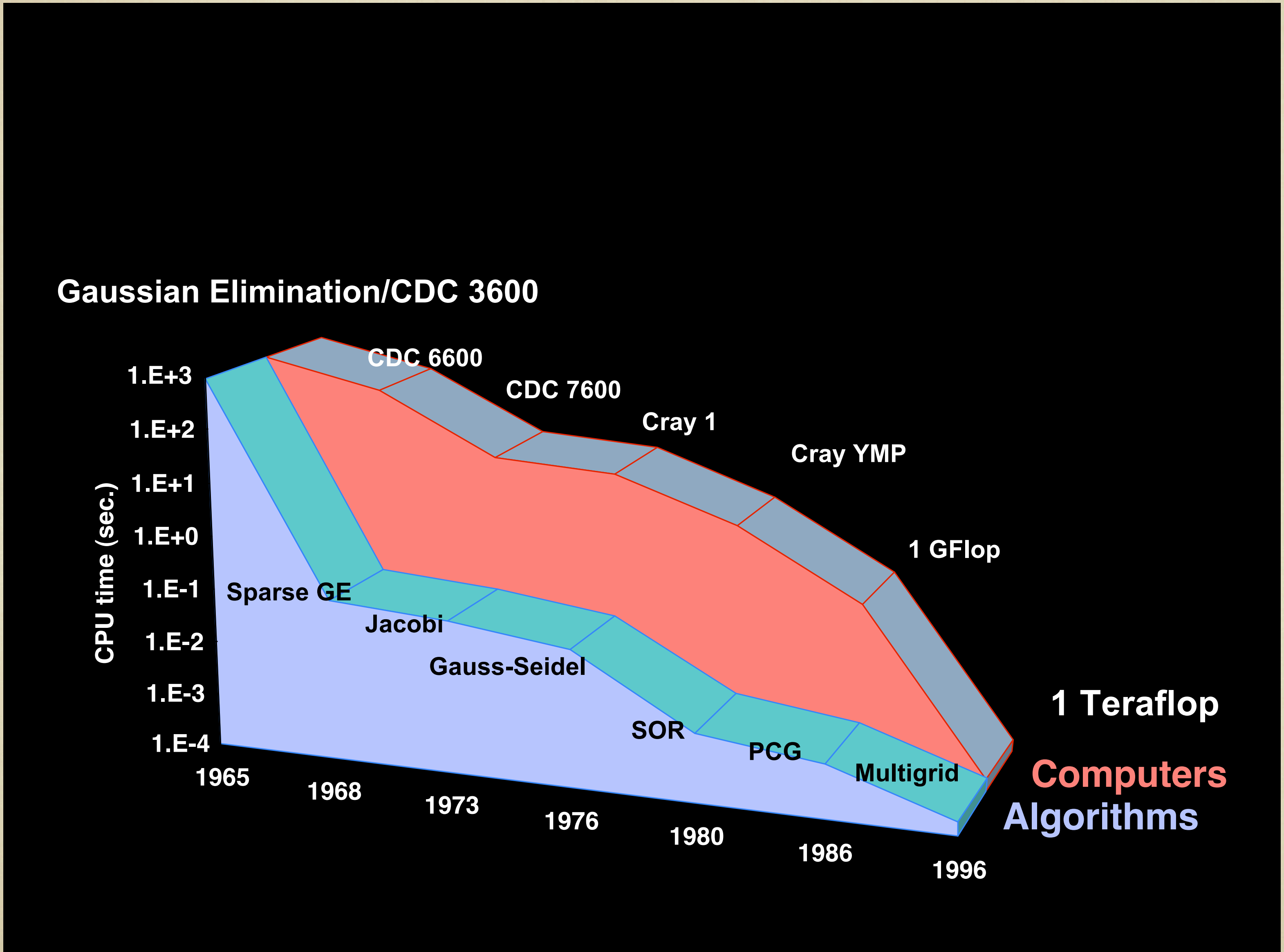
Def:

A Numerical Analyst is a person who solves a nearby problem, i.e. one that you are almost interested in

Improvements in Algorithms have outpaced those in hardware

- Bixby ran a set of tests in 2002
- 680 test problems with up to 7 million equality constraints
- Compared to algorithms and machines dating to 1990
- Results show improvements in simplex algorithms have yielded a speedup of 960 compared to 800 for hardware

Similar improvements in linear algebra



Summary

- Insight not numbers
- Always have to deal with inaccuracies
- Don't want just the solution, but a sense of how accurate/sensitive it is to perturbations
- Algorithms have yielded greater improvement than computer hardware in many important problems

Problem Set 1

- Search the web to find a problem of interest to you that uses computational mathematics.
- Determine the mathematics areas that are used for the given problem.
- Explain why this problem is interesting, what mathematics is used, and why those mathematics areas were used.
- You should also think about how you would improve either the methods they use or how you would extend the work.
- You will be asked to give a short presentation (5 minutes)