Scientific Computing

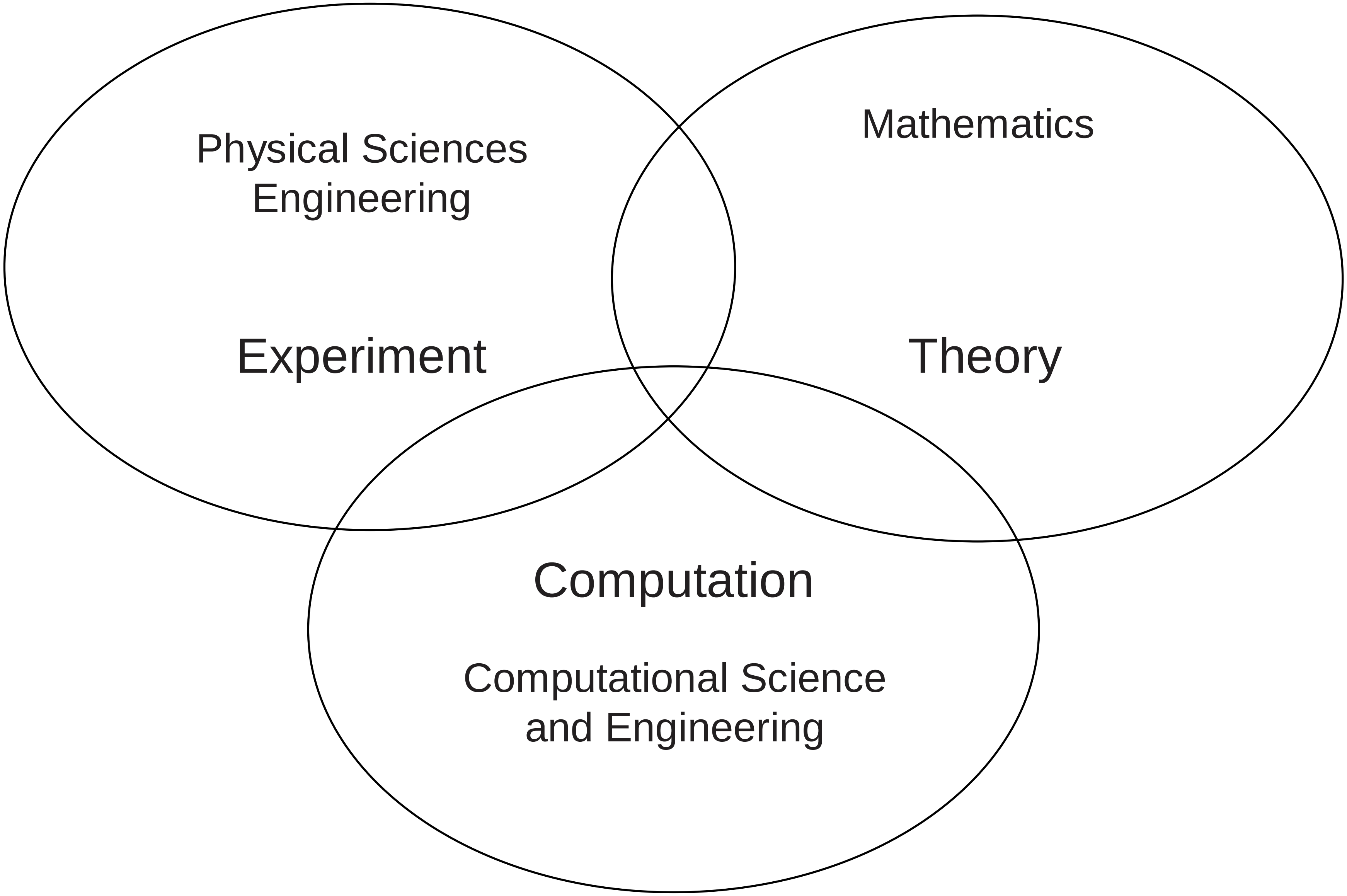
Ian Mitchell
Department of Computer Science
The University of British Columbia

research supported by
National Science and Engineering Research Council of Canada
ONR Computational Methods for Collaborative Control MURI (N00014-02-1-0729)

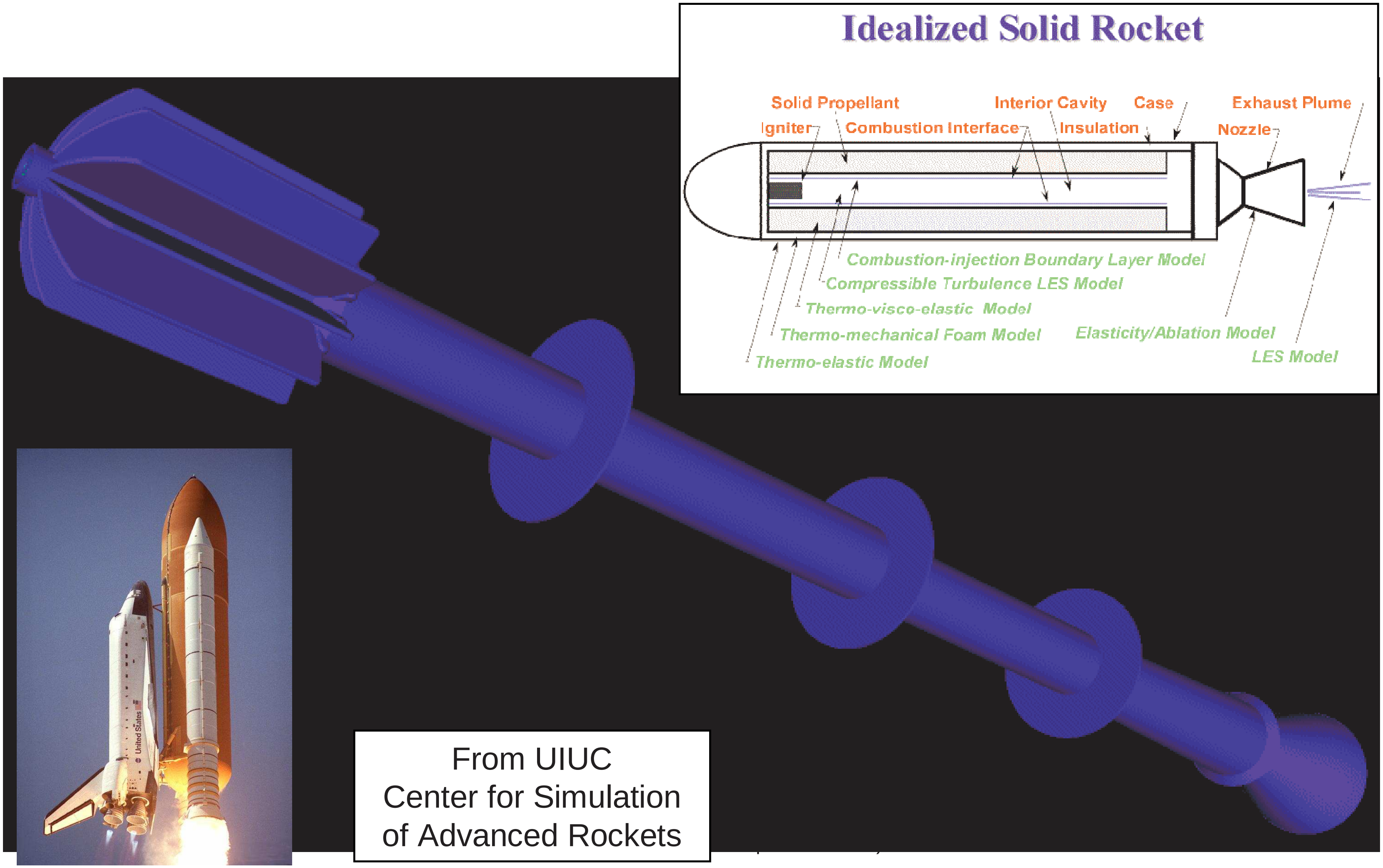
What's in a Name?

- Various incarnations, each with its own subtle implications
  - Numerical analysis
  - Scientific computing
  - Applied mathematics
  - Computational science
  - Mathematical engineering?
- Trefethen definition: "The study of algorithms for the problems of continuous mathematics"

Exploring the World



Space Shuttle Solid Rocket Booster



From UIUC
Center for Simulation
of Advanced Rockets

No Such Thing as a Digital Circuit

- Digital circuits: some elements do not behave discretely
- Metastable points provably separate digital domains
- For example
  - Arbitration and synchronization circuits
  - Mixed analog / digital design

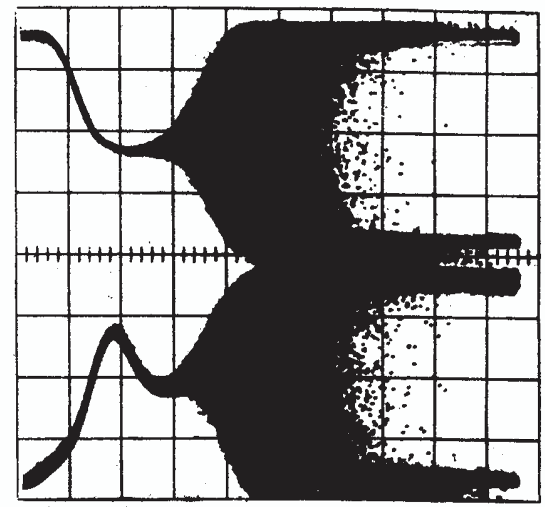


Fig. 1. Q and $\overline{Q}$ of ECL clocked R-S flip-flop with clock and data inputs changing simultaneously (5 ns/div, 0.25 V/div).

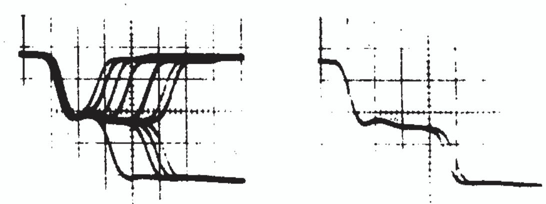


Fig. 2. Selected responses of ECL clocked R-S flip-flop to clock and data inputs changing simultaneously (10 ns/div, 0.2V/div).

From: Thomas J. Chaney and Charles E. Molnar, "Anomalous Behavior of Synchronizer and Arbiter Circuits", *IEEE Transactions on Computers*, pp. 421-422 (April 1973).

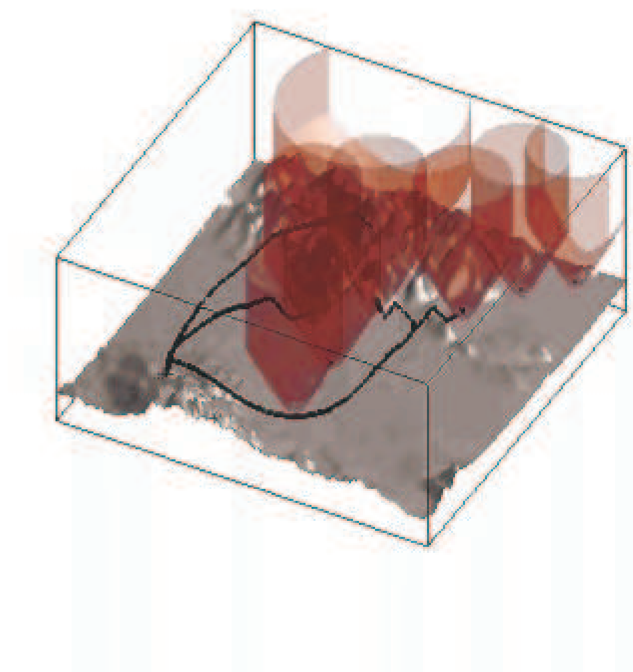
Process of Computational Simulation

1. Develop a mathematical model—usually equations—of the physical phenomenon or system
2. Develop algorithms to solve the equations numerically
3. Implement the algorithms in computer software
4. Run the software to simulate the process
5. Visualize the results in a comprehensible form
6. Interpret and validate the results, repeating steps as necessary
   - From “Scientific Computing: An Introductory Survey” by Heath

- Success is measured by (in no particular order)
  - Efficiency
  - Accuracy
  - Reliability

How can Computer Scientists Contribute?

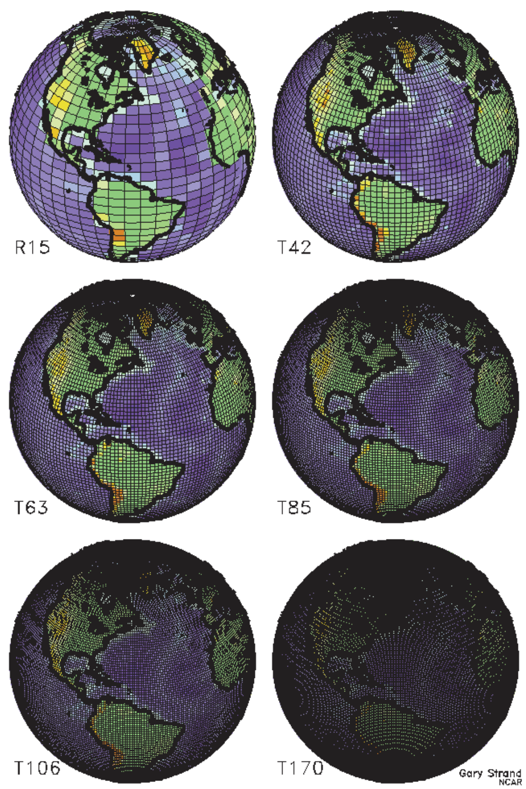
- Many similar equations allow common algorithms



How can Computer Scientists Contribute?

- Lots of data, lots of operations, lots of bandwidth, lots of code: the same problems faced by many other computer scientists

Parallel Climate Model



TOP500 SUPERCOMPUTER SITES

the globus toolkit™
www.globustoolkit.org

PETSc

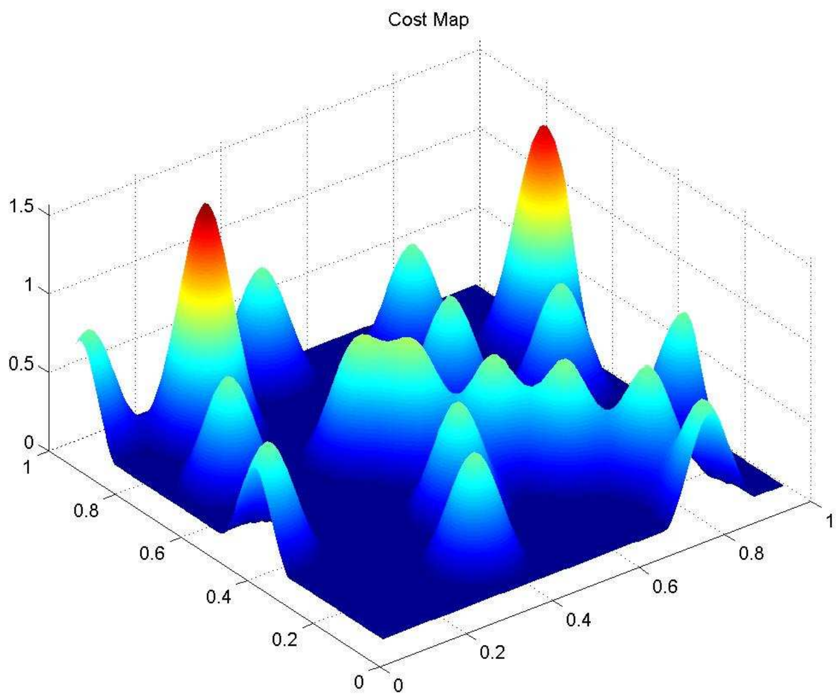
Portable Extensible Toolkit for Scientific Computing

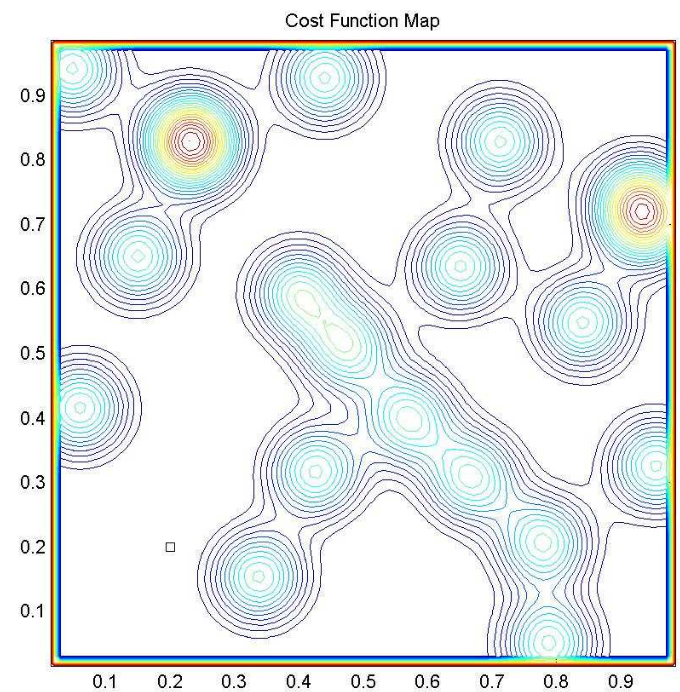
Active Research Areas

- Linear Algebra
- Differential Equations
- Optimization
- Randomized (Monte Carlo) Algorithms
- Multiresolution Approximation (eg. wavelets, multipole, multigrid)
- Multiphysics Simulation (eg. fluid/solid interaction)
- Inverse Problems (eg. tomography, medical imaging)
- Parallel computing
- Applications
  - Engineering and sciences
  - In CS: animation, vision & image processing, robotics, search engines & data mining, hardware verification, machine learning & AI, protein folding, etc.

Example: Robotic Path Planning

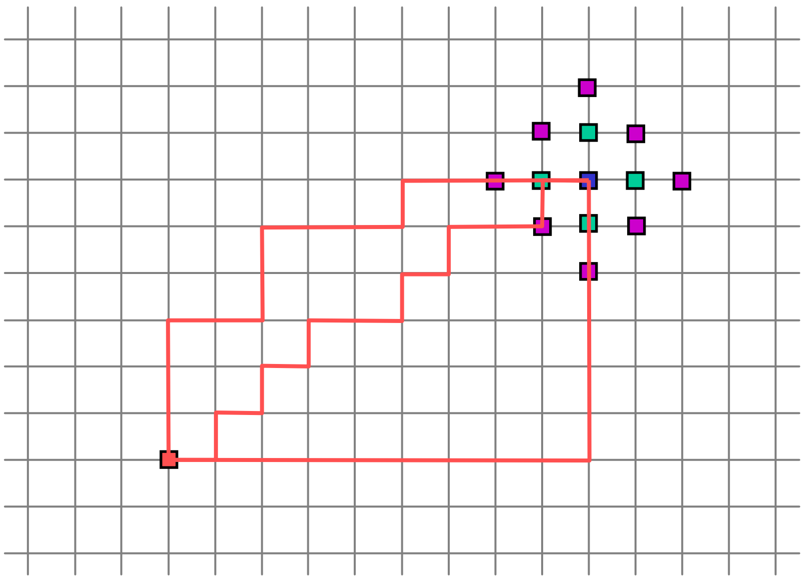
- Find the optimal path $p(s)$ to a target (or from a source)
- Inputs
  - Cost to pass through each state in the state space
  - Set of targets or sources (provides boundary conditions)





Dijkstra's Method

- Solution of dynamic programming on a discrete graph
  1. Set all interior nodes to a dummy value infinity ∞
  2. For all boundary nodes ***x*** and all ***y*** ∈ ***N(x)*** approximate ***V***(***y***) by DPP
  3. Sort all interior nodes with finite values in a list
  4. Pop node ***x*** with minimum value from the list and update ***V***(***y***) by DPP for all ***y*** ∈ ***N(x)***
  5. Repeat from (3) until all nodes have been popped



Constant cost map ***c***(***y***→***x***) = 1

- Boundary node ***V***(***x***) = 0
- First Neighbors ***V***(***x***) = 1
- Second Neighbors ***V***(***x***) = 2
- Distant node ***V***(***y***) = 15

Optimal path?

Basic Tools

- Fundamental problems
  - Discrete representation of a continuous domain
  - Finite approximation of infinite or uncomputable processes
- General strategies
  - Take advantage of continuity (in its many forms)
  - Finite dimensional approximations (parametric or nonparametric)
  - Algorithms that converge rapidly to neighborhood of true solution
  - Differentials become differences
  - Nonlinear becomes linear
- Essential questions
  - Sensitivity and conditioning of original problem
  - Stability of numerical algorithm
  - Data, truncation and rounding errors
  - Cost (time, memory, communication, programmer, etc)

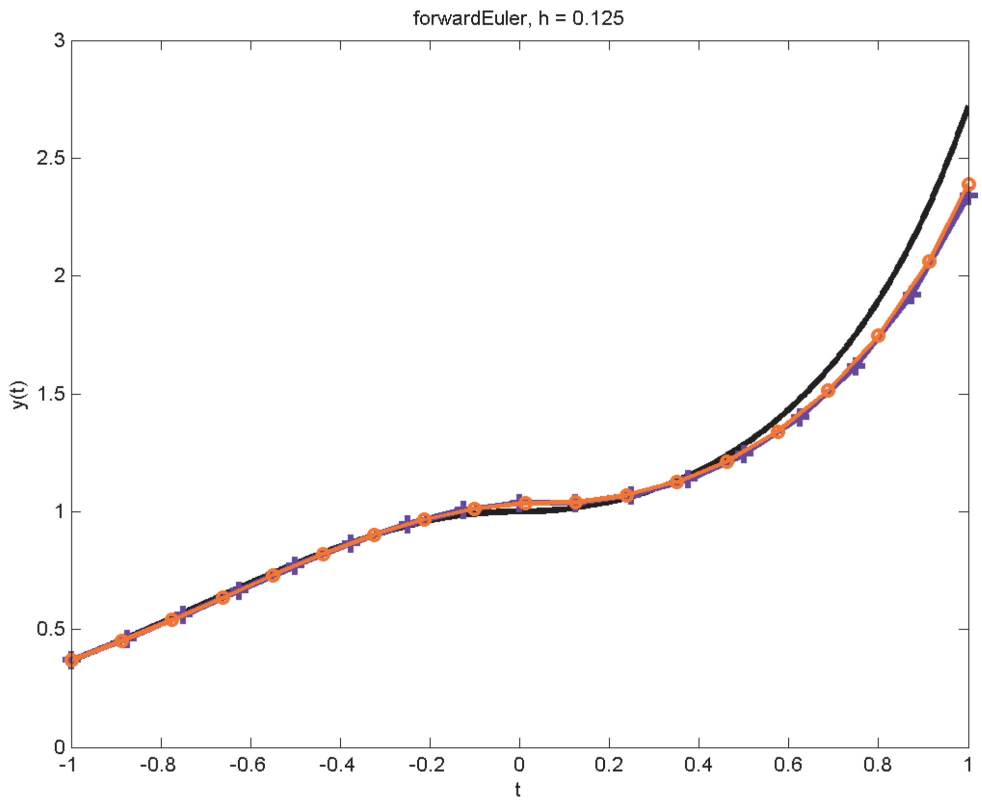
Example: Algorithm Convergence

- Simple differential equation

$$\frac{dy(t)}{dt} = 2|t|y(t) \qquad y(-1) = \exp(-1)$$

- Discrete solution by Forward Euler scheme

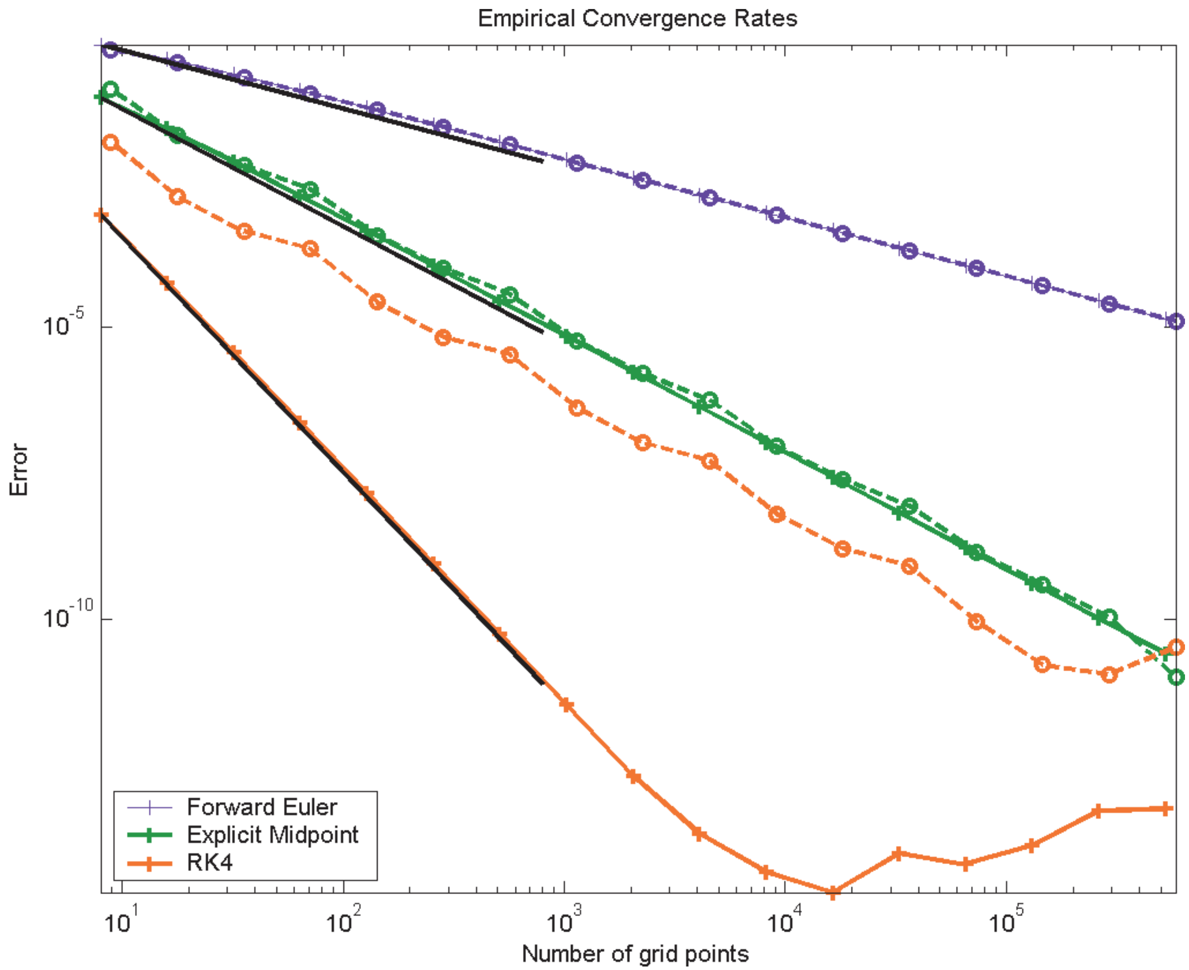
$$y(t_{n+1}) = y(t_n) + 2|t_n|y(t_n) \quad t_0 = -1$$



Example: Algorithm Convergence

- Can evaluate error from analytic solution

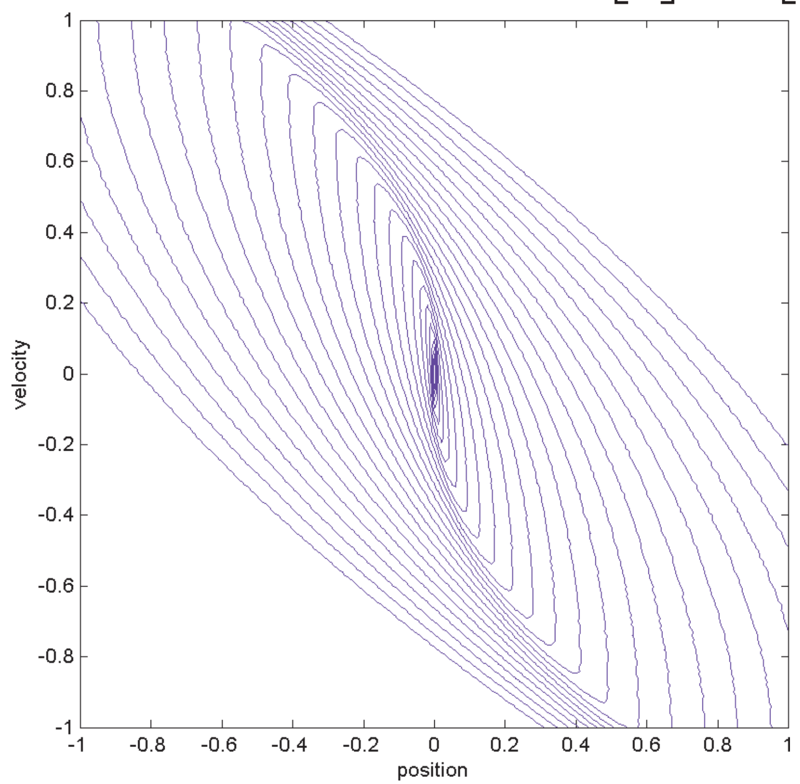
$$y(t) = \exp(-t^2 \mathrm{sign}(t))$$



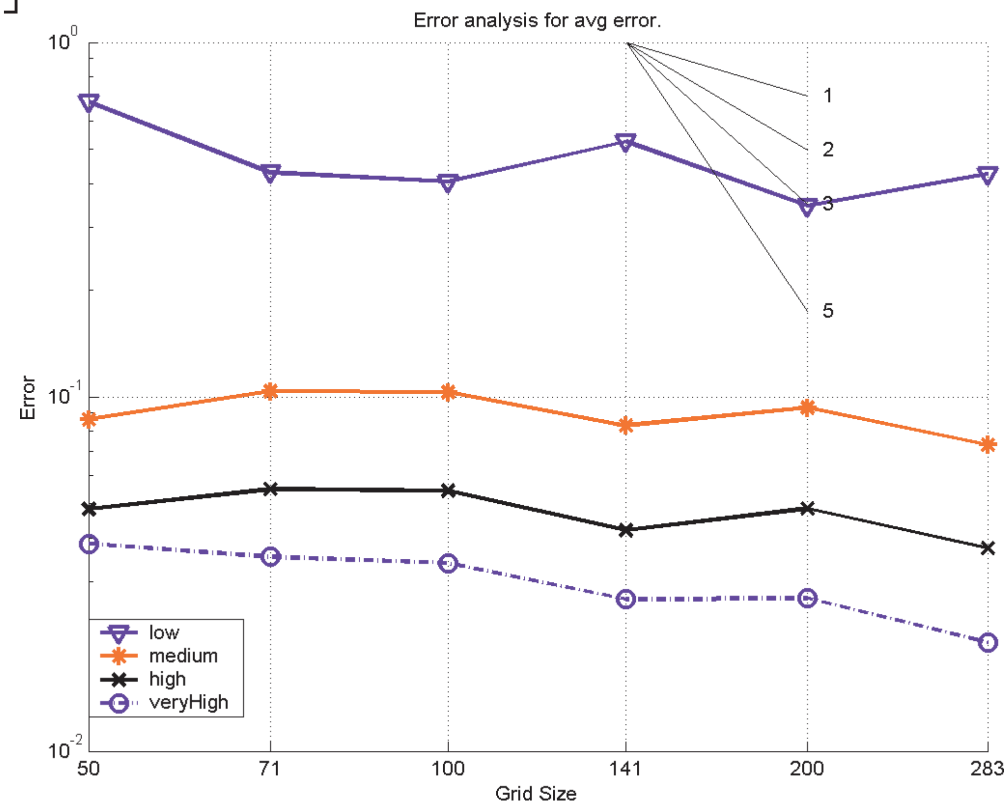
Research is Messy

- Attempting to approximate minimum time to reach the origin for a double integrator

$$\frac{d}{dt}\begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} v \\ a \end{bmatrix} \qquad \text{choose } |a| \leq 1$$



True Solution

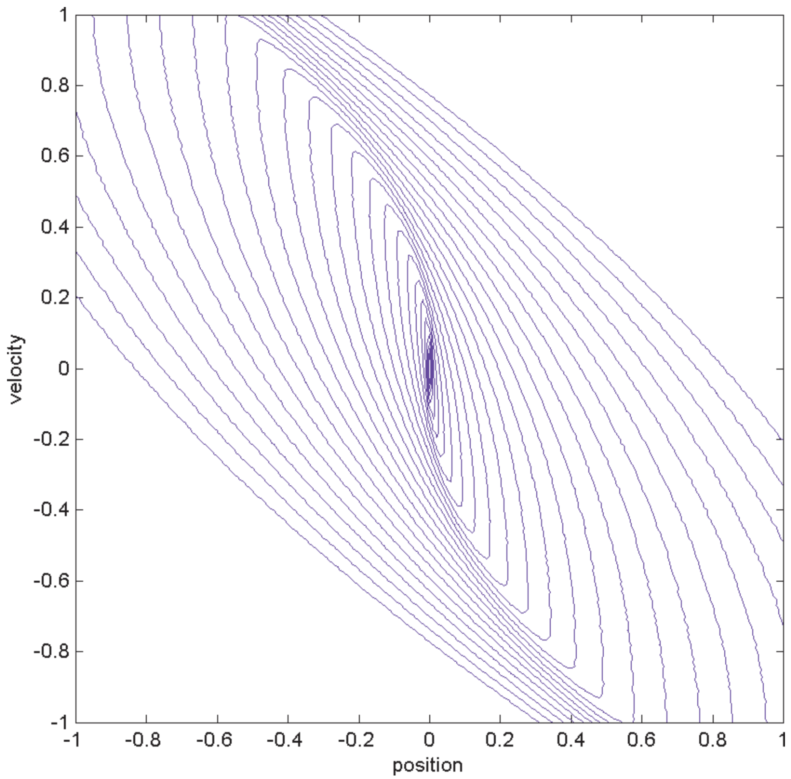


Early Attempt

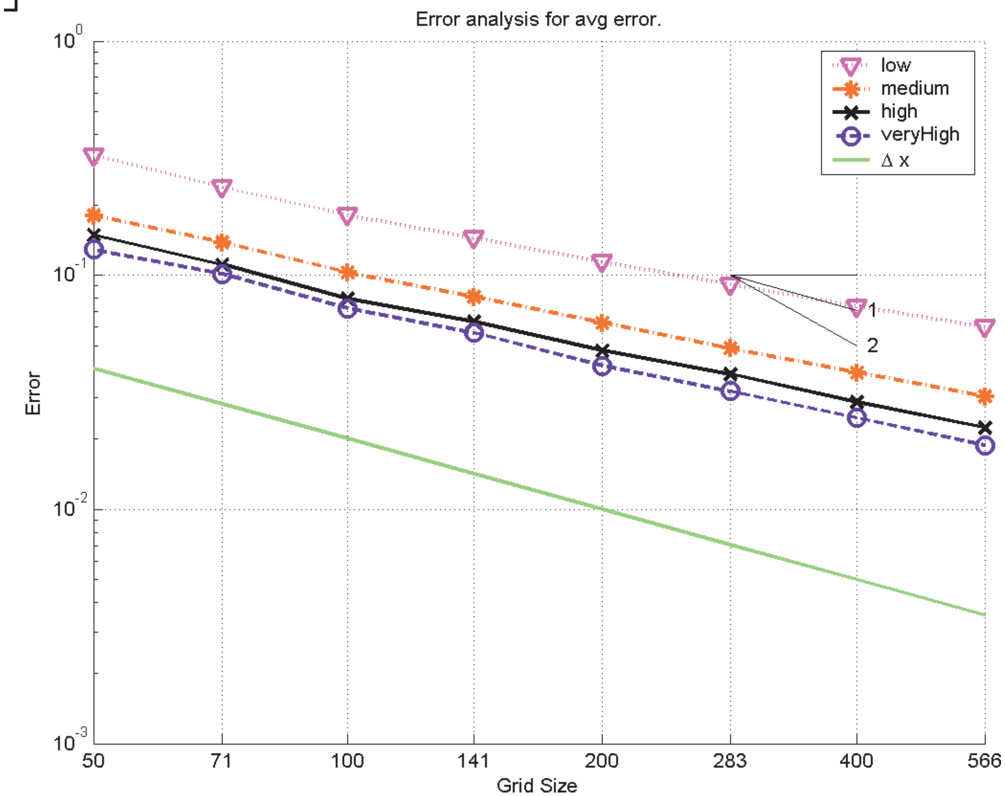
And It May Not Succeed

- Attempting to approximate minimum time to reach the origin for a double integrator

$$\frac{d}{dt}\begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} v \\ a \end{bmatrix} \qquad \text{choose } |a| \leq 1$$



True Solution



Latest Result

But I Have “Numerical Recipes in *”

- There are lots of existing software packages and environments
  - Environments: Matlab, Maple, Mathematica, Octave, Scilab
  - General collections: Netlib, GAMS, Numerical Recipes
  - Problem specific packages: LAPACK, PETSc, SUNDIALS, AMPL, NEOS, and many, many more
- Why study numerical algorithms?
  - To choose the appropriate software for the problem
  - To formulate the problem in a manner appropriate to the problem
  - To detect, understand and correct errors
  - To modify or create software appropriate to a particular problem

Scientific Computing

For more information contact

Ian Mitchell

Department of Computer Science

The University of British Columbia

mitchell@cs.ubc.ca

http://www.cs.ubc.ca/~mitchell