Overview of Optimization Methods for Scientific Computing

Juan Meza

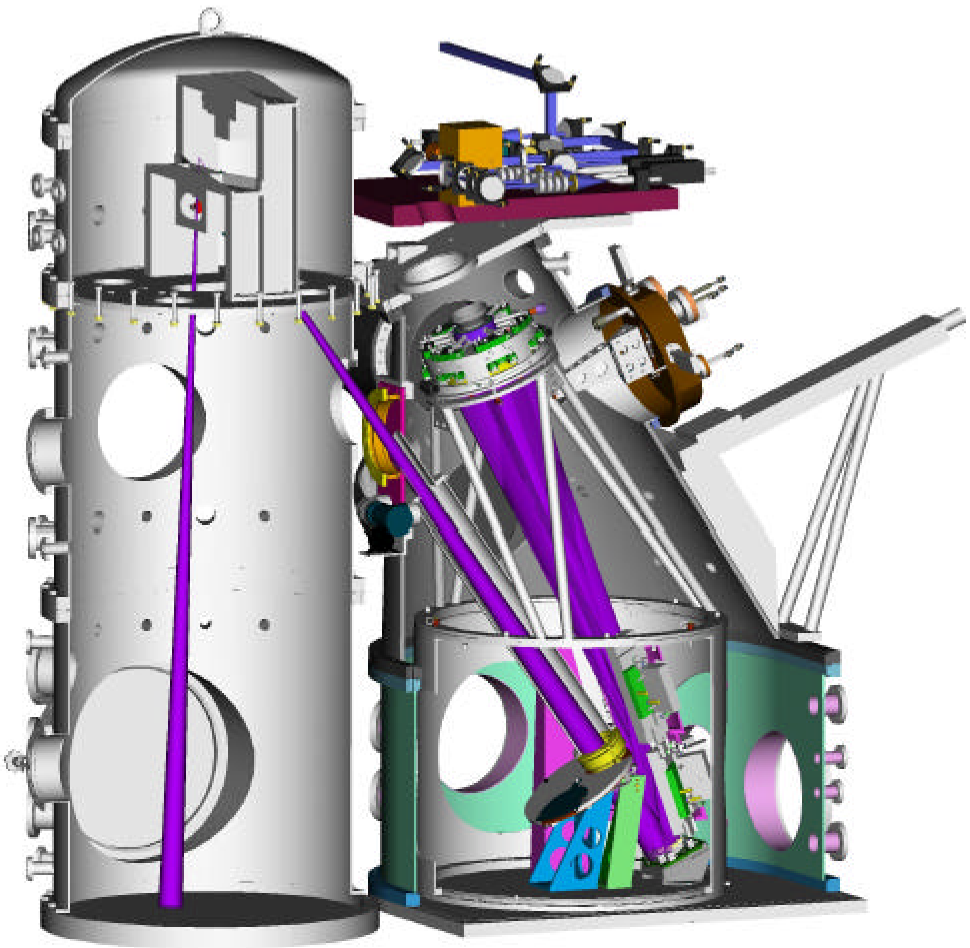
Lawrence Berkeley National Laboratory

http://www.nersc.gov/~meza

Outline

- Motivation for optimization in scientific applications
- Brief review of basic optimization techniques
- Some practical issues
- Parallel optimization methods
- Constrained optimization
- Summary
- Future directions

Parameter identification example

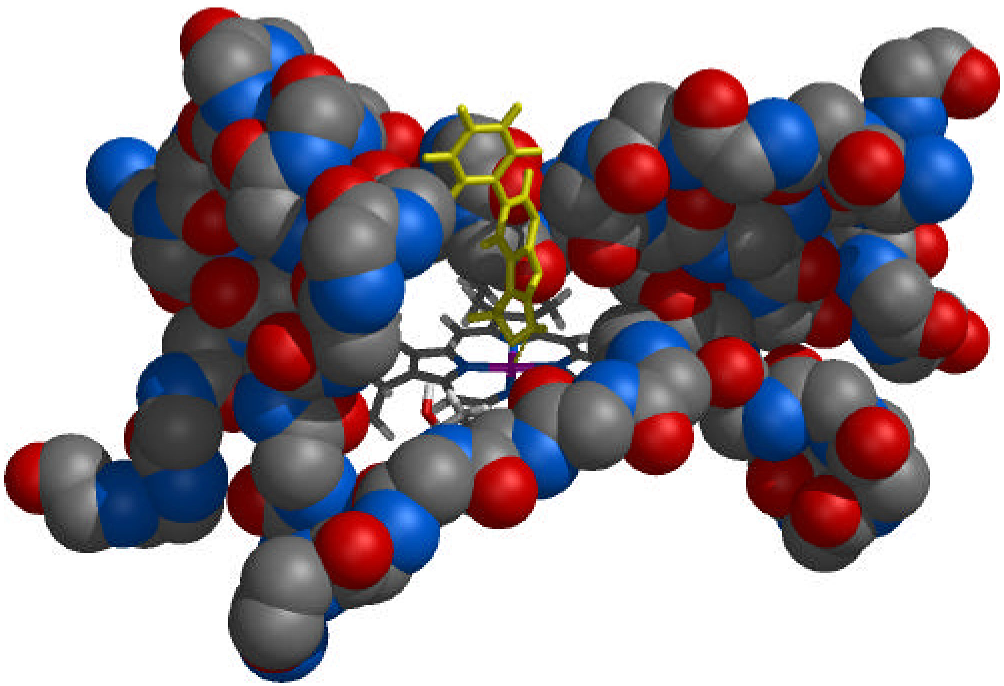


- Find model parameters, satisfying some bounds, for which the simulation matches the observed temperature profiles
- Computing objective function requires running thermal analysis code

$$\min_{x} \quad \sum_{i=1}^{N} (T_i(x) - T_i^*)^2$$

$$\text{s.t.} \quad 0 \le x \le u$$

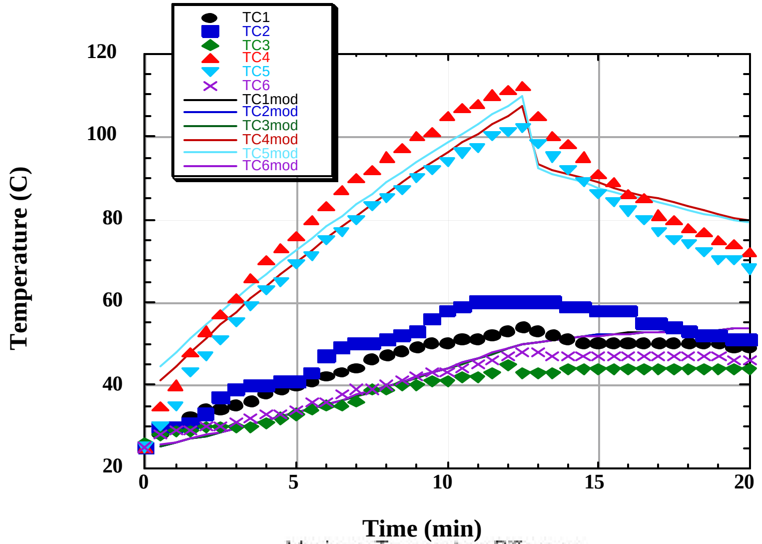
Energy minimization problem

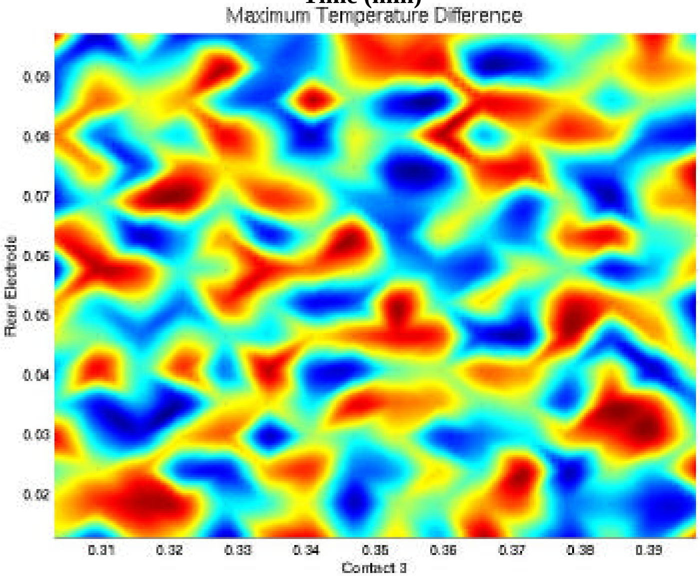


Docking model for environmental carcinogen bound in *Pseudomonas Putida* cytochrome P450

- A single new drug may cost over $500 million to develop and the design process typically takes more than 10 years
- There are thousands of parameters and constraints
- There are thousands of local minima

Data Fitting Example





- Objective function consists of computing the max temperature difference over 5 curves
- Each simulation requires approximately 7 hours on 1 processor
- Uncertainty in both the measurements and the model parameters

General observations

- Many optimization problems have expensive objective functions
  - Objective function requires solution to a large-scale PDE or similar type of simulation
  - One function evaluation can take several CPU hours even on a parallel processor
- Adding more processors to the function evaluation is not always efficient or productive
  - Many applications do not scale well
- May not even be able to parallelize the objective function
  - Black-box functions

The Basic Ideas of Optimization

General Optimization Problem

$$\min_{x \in \Re^n} f(x),$$ Objective function

$$s.t. \;\; h_i(x) = 0,$$ Equality constraints

$$g_j(x) \geq 0$$ Inequality constraints

Lagrangian:

$$L = f(x) + y^T h(x) - w^T g(x)$$

Some special cases

- Linear programming
  - » Objective function and constraints are linear
- Quadratic programming
  - » Objective function is quadratic, constraints are linear
- Discrete/Integer programming
  - » Optimization parameters have discrete values or must take on integer values only
- Stochastic optimization
  - » Stochastic components inherent in the problem

Some standard assumptions

- Objective function has infinite (machine) precision
- Objective function is smooth
  - » First and second derivatives available
  - » Both derivatives are also “nice”
- Constraints are linearly independent and smooth

General Philosophy

- Build an approximate model of the nonlinear objective function
  - » Usually quadratic
  - » Some interesting new research on other models
- Solve the model for its minimum
- See how well you did and either accept the answer or throw it away
- Repeat until you run out of time and or money

Derivation of Newton equations

- Build quadratic model

$$q(x_k + s) = f(x_k) + g(x_k)^T s + \tfrac{1}{2} s^T H(x_k) s$$

- Find the minimizer of the quadratic

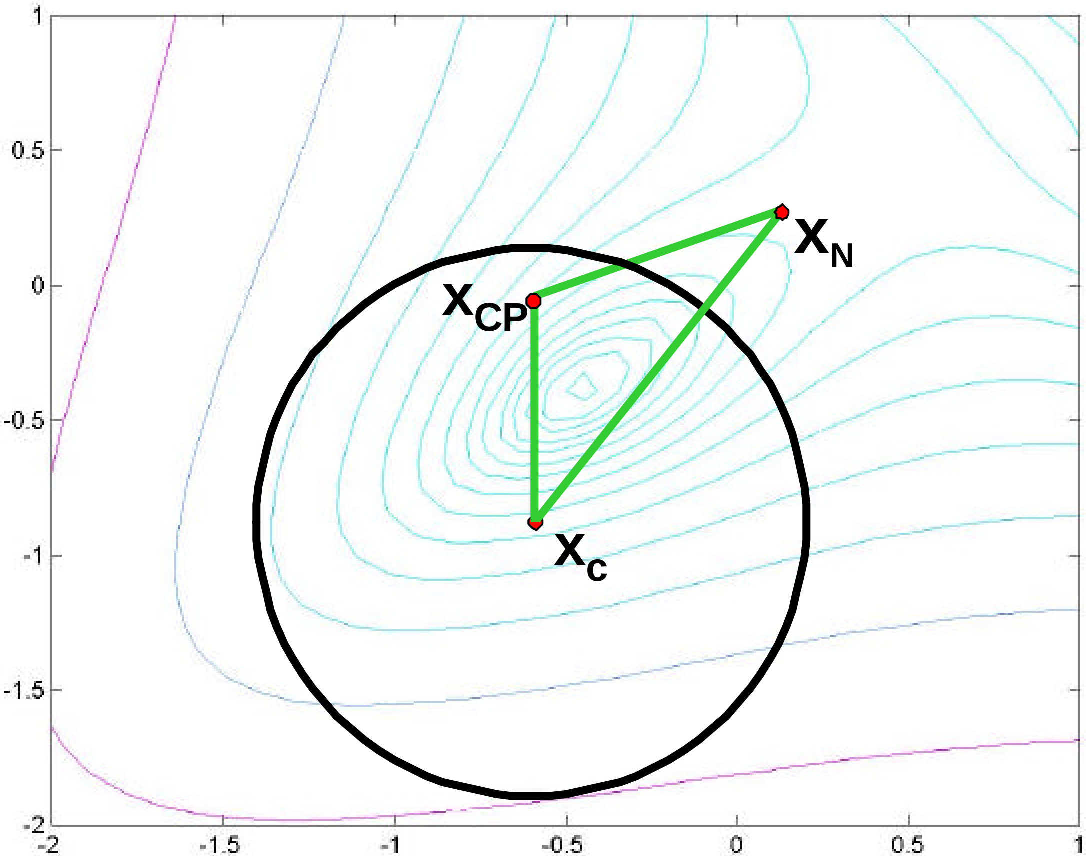
$$\min q(x) \quad \Leftrightarrow \quad \nabla q(x) = g + Hs = 0$$

$$\begin{array}{ll} Solve & Hs_k = -g \\ Set & x_{k+1} = x_k + a \cdot s_k \end{array}$$

- Check how well you did, i.e. is

$$f(x_{k+1}) < f(x_k)$$

Newton Methods



- Fast convergence properties
- Good global convergence properties
- Quasi-Newton approximations work well in practice
- Inherently serial
- Difficulties with noisy functions

Practicalities

or

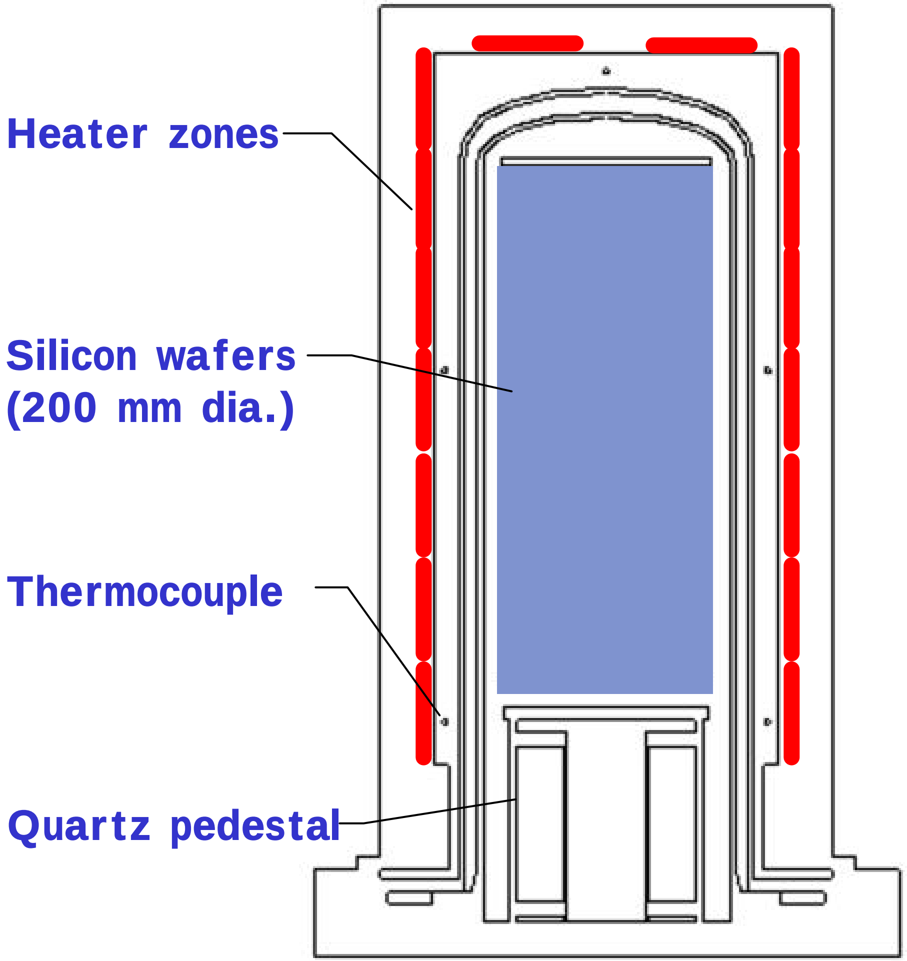
What you don’t know will probably hurt you!

Facts of life for practical optimization

- Objective function has infinite (machine) precision
  - » Sometimes true, but many simulation-based optimization problems can create noisy behavior
- Objective function is smooth
  - » Probably differentiable, but how do you prove it
  - » What do you do if you're not given derivative information
- Constraints are linearly independent and smooth
  - » Users can sometimes over specify or incorrectly guess constraints
  - » You have to beware of "hidden" constraints

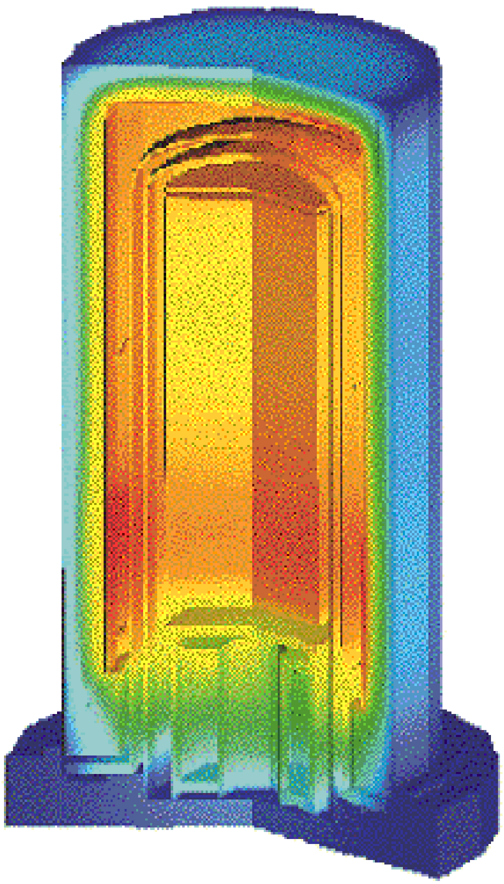
What does this mean in a Practical Problem?

Optimizing the performance of a LPCVD furnace



- Temperature uniformity is critical
  - » between wafers
  - » across a wafer
- Independently controlled heater zones regulate temperature
- Wafers are radiatively heated

Computing the objective function requires the solution of a PDE



Temperature fields in a vertical, stacked-wafer, low-pressure, chemical-vapor-deposition furnace

- Finding temperatures involves solving a heat transfer problem with radiation
- Two-point boundary value problem solved by finite differences
- Adjusting tolerances in the PDE solution trades off noise with CPU time
  - » Larger tolerances lead to
    - – Less accurate PDE solutions
    - – Less time per function evaluation

The goal is to find heater powers that yield optimal uniform temperature

$$\min \; F(\mathbf{p}) = \sum_{i=1}^{N} (T_i(\mathbf{p}) - T^*)^2,$$

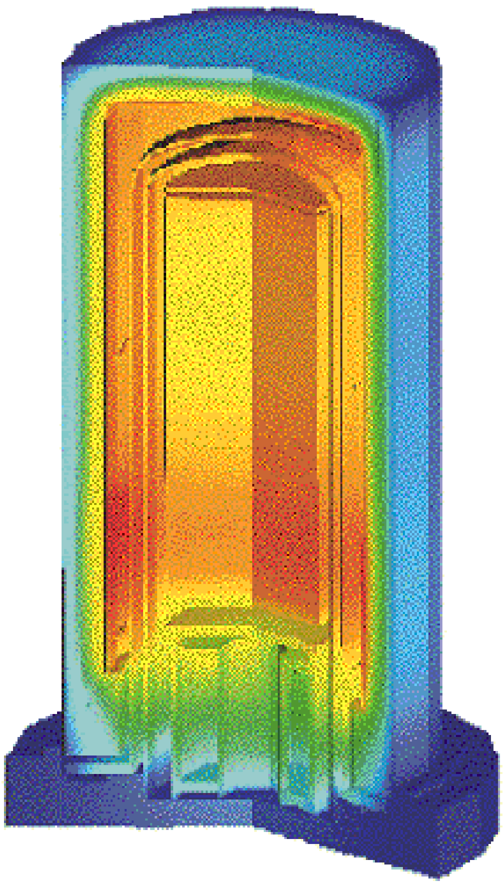
where $\mathbf{p}$ is the vector containing the heater powers,

$T_i(\mathbf{p})$ is the temperature at discretization point i given powers $\mathbf{p}$,

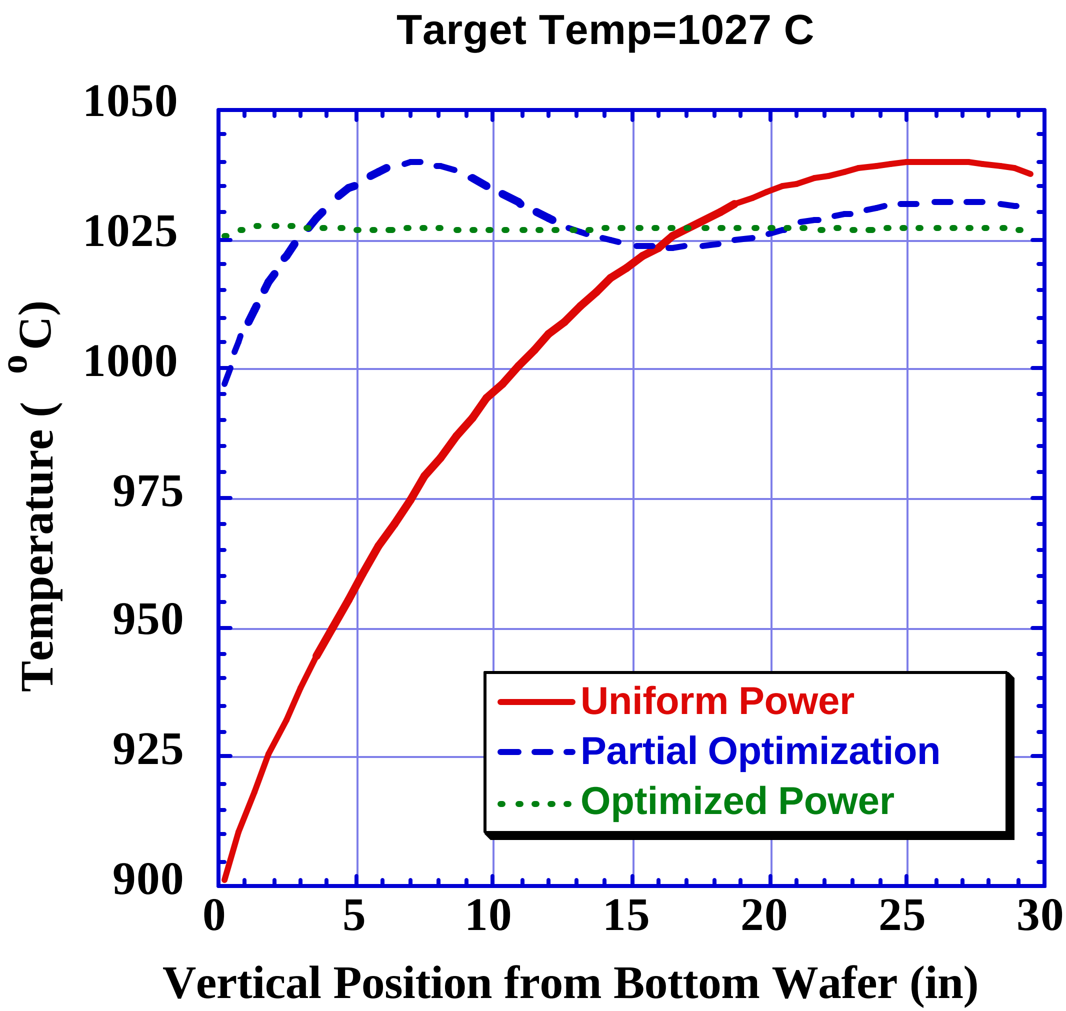
T^* is the target temperature, and

N is the total number of discretization points

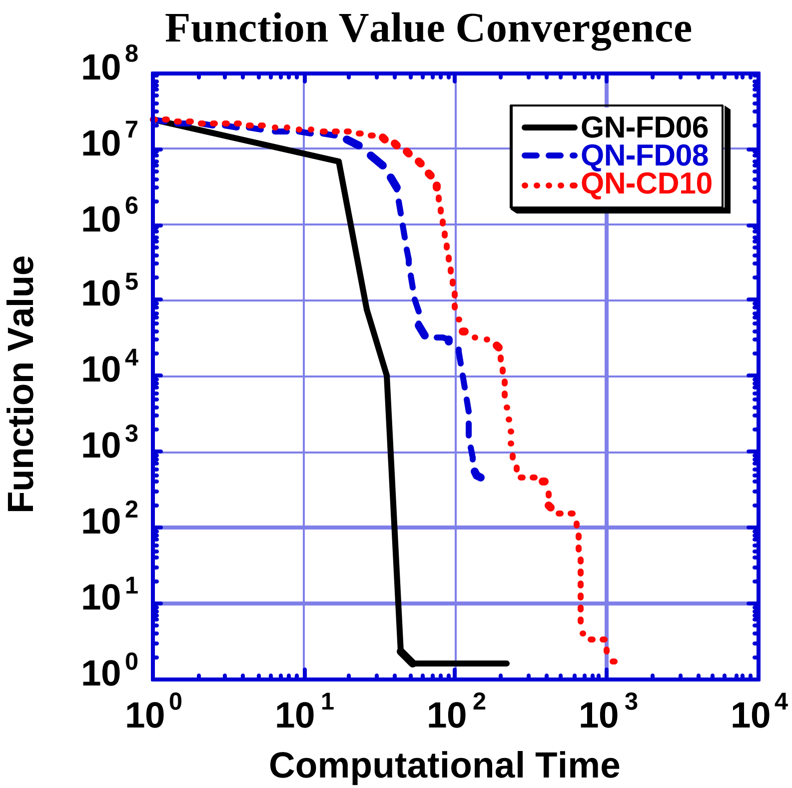
Optimized Power Distribution

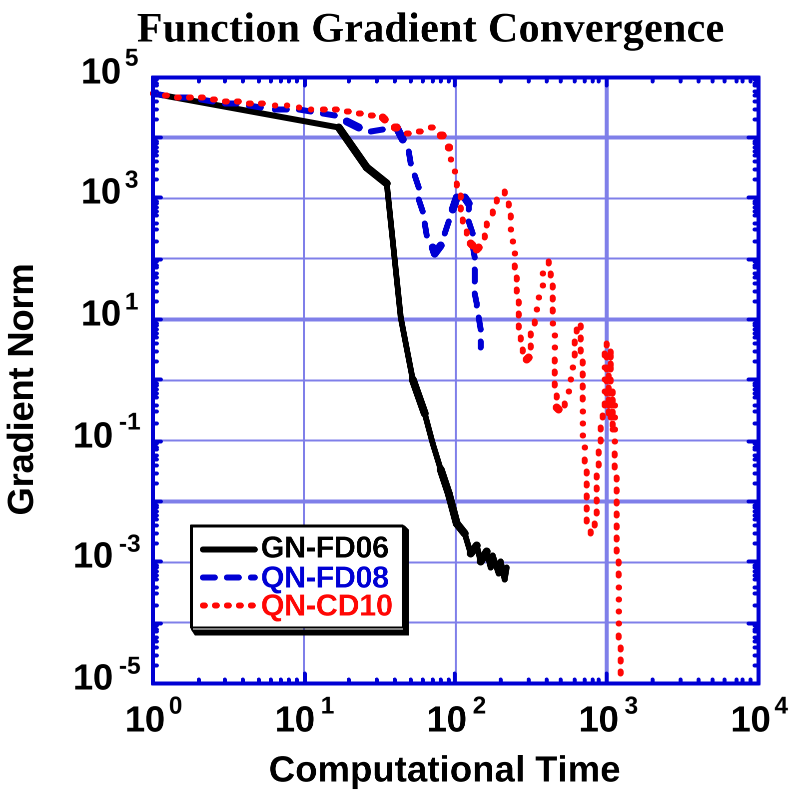


Temperature fields in a vertical, stacked-wafer, low-pressure, chemical-vapor-deposition furnace



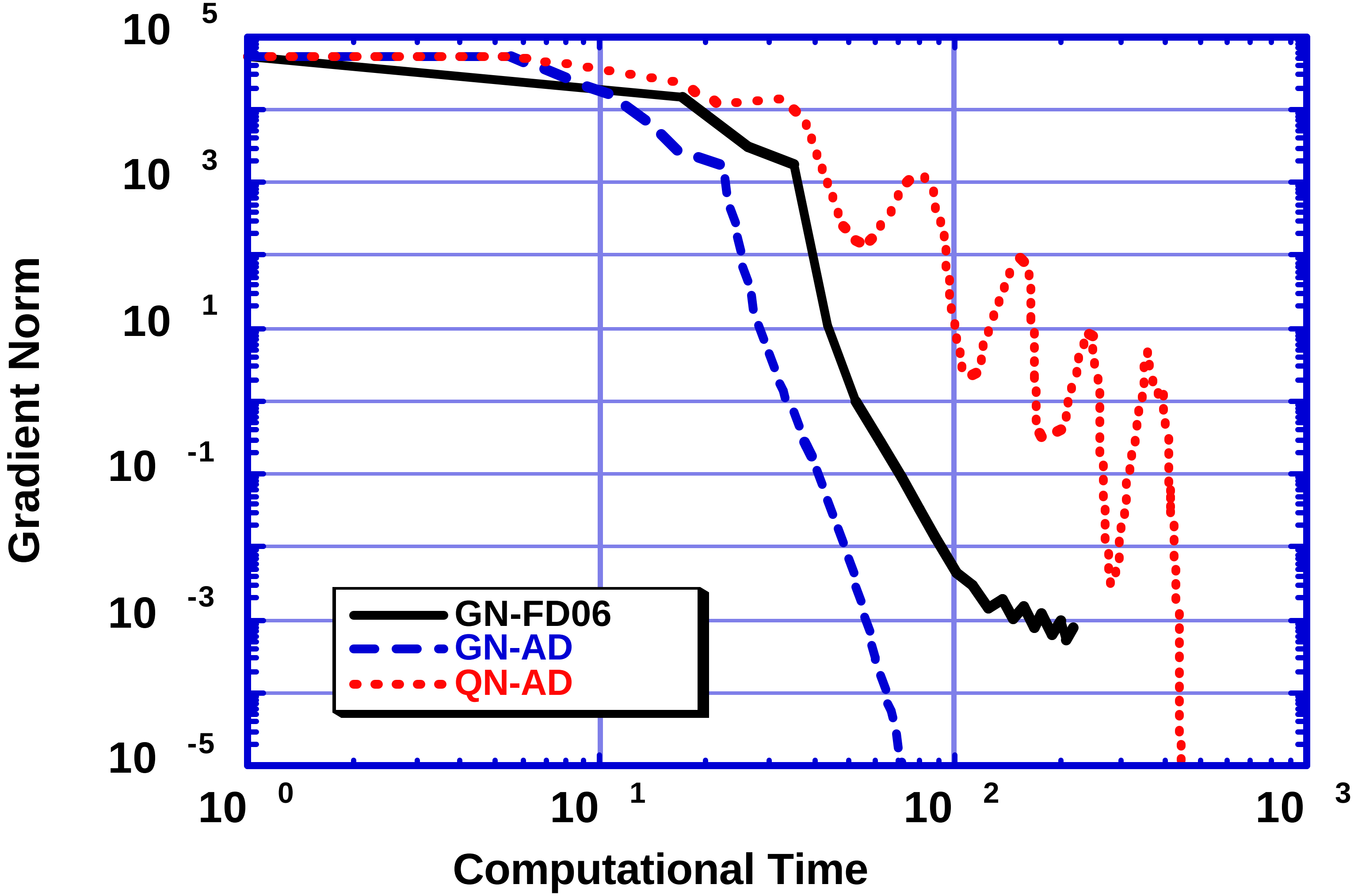
Finite-Difference Gradients





- **7 Zone furnace configuration**
- **Quasi-Newton method exhibits "stair-stepping"**

Analytic Gradients vs. Finite-Differences



Parallel Optimization

Schnabel (1995) identified three levels for introducing parallelism into optimization

1. Parallelize evaluation of functions, gradients, and or constraints
2. Parallelize linear algebra
3. Parallelize optimization algorithm at a high level

Parallelism is easily introduced when finite-difference gradients are used

- Option 1 in Schnabel's taxonomy
- Components of the gradient can be computed independently on separate processors
- Components of the gradient can be computed speculatively (Byrd, Schnabel, Shultz, 1988)
  - » trial point is accepted 60-80% of the time
  - » compute components of the gradient simultaneously with the function value
  - » difficult to do better than this strategy

Parallelize the linear algebra

- Much research in this area
- Outstanding progress in recent years

- BUT, this is really only useful for large-scale optimization problems
  - » If the function evaluation dominates the computational time, then this option will not prove effective

Direct Search Methods

- Methods that “in their heart” do not use gradient information
- Main operation is function comparisons
- Useful whenever the derivative of the objective function is not available or is too expensive to compute

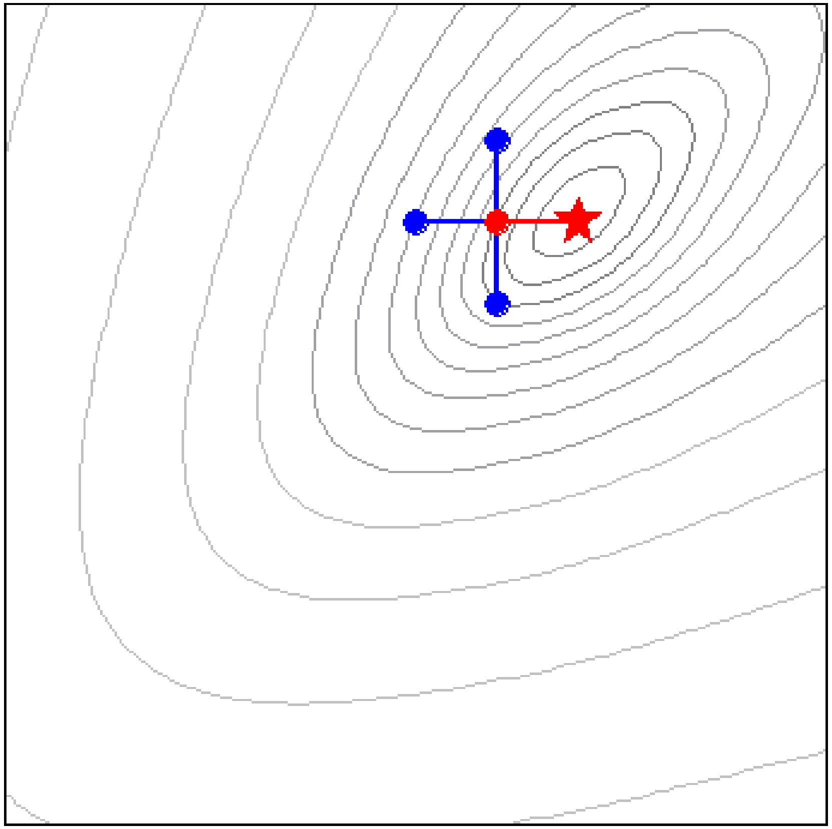
Direct Search Methods (cont.)

- Line search
- Conjugate Direction
- Simulated Annealing
  - Based on annealing – cooling of a liquid to a solid
  - Allows uphill directions
  - Claim to find global minimum
- Evolutionary Algorithms / Genetic algorithms
  - Based on “evolutionary” concepts
  - Can be used for discrete variable problems
  - Claim to find global minimum
- ...

Direct Search Methods (cont.)

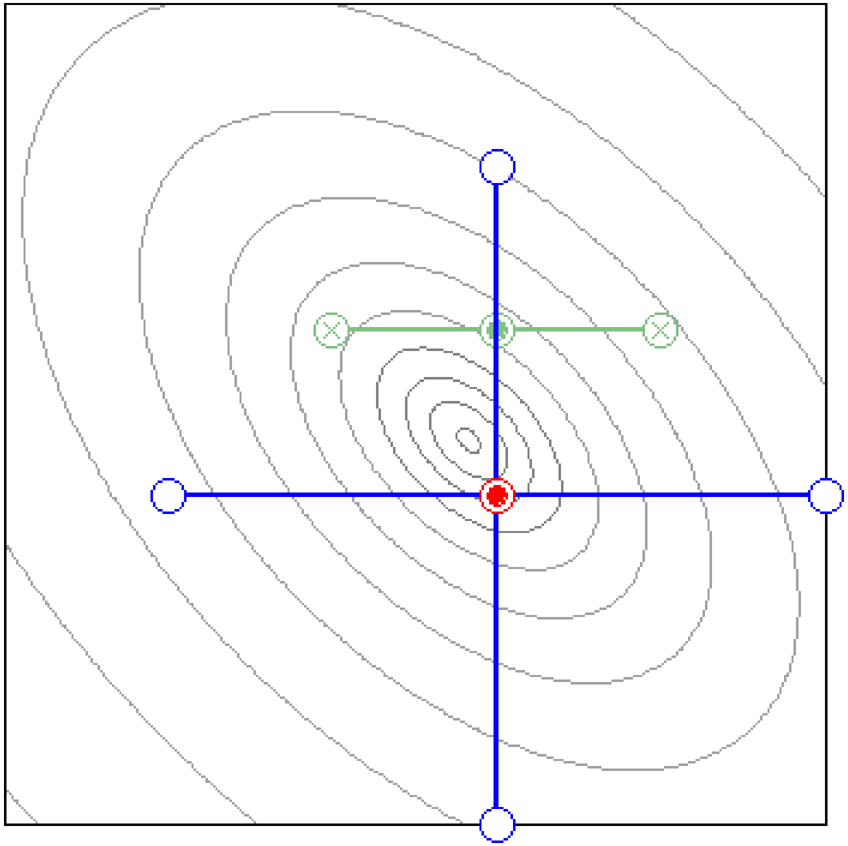
- Nelder-Meade Simplex
  - » One of the most popular methods
  - » Can construct examples where method fails (convex, differentiable, 2 variables)
- Multidirectional Search, Dennis &Torczon, (1989, 1991)
- Asynchronous Parallel Pattern Search, Hough, Kolda, Torczon, (2000)

Pattern Search



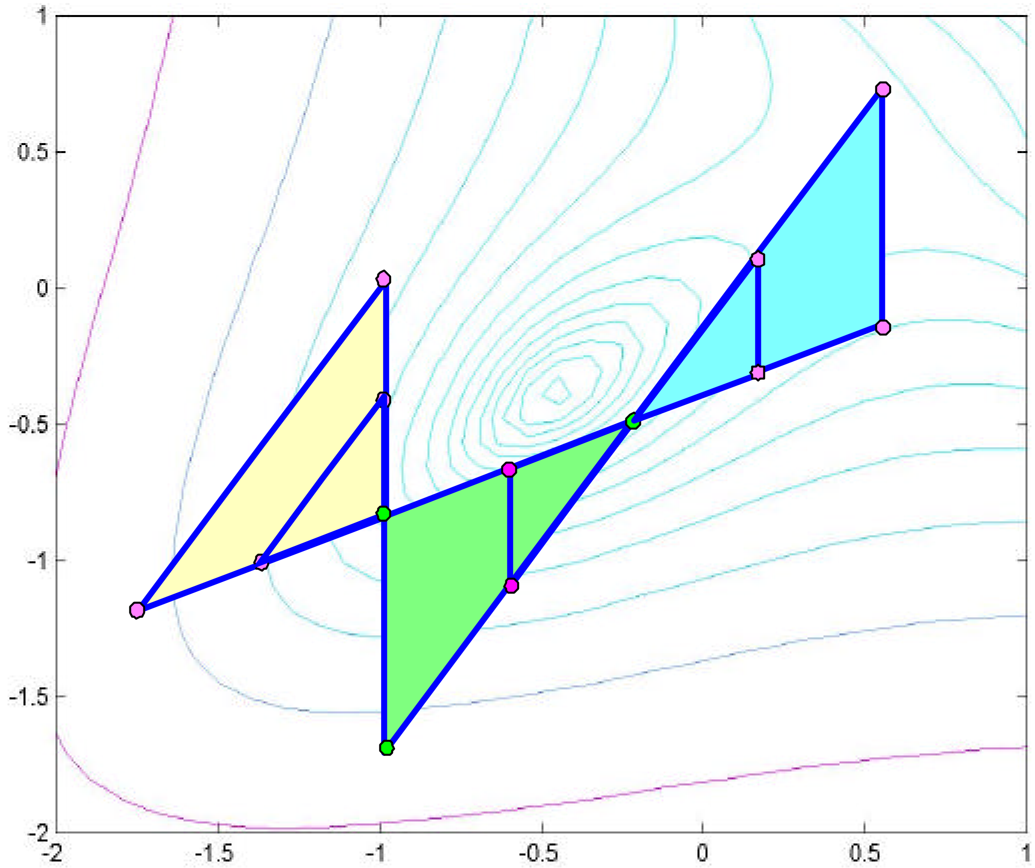
Special thanks to Tammy Kolda for this slide

Basic Parallel Pattern Search



Special thanks to Tammy Kolda for this slide

Parallel Direct Search (PDS) method

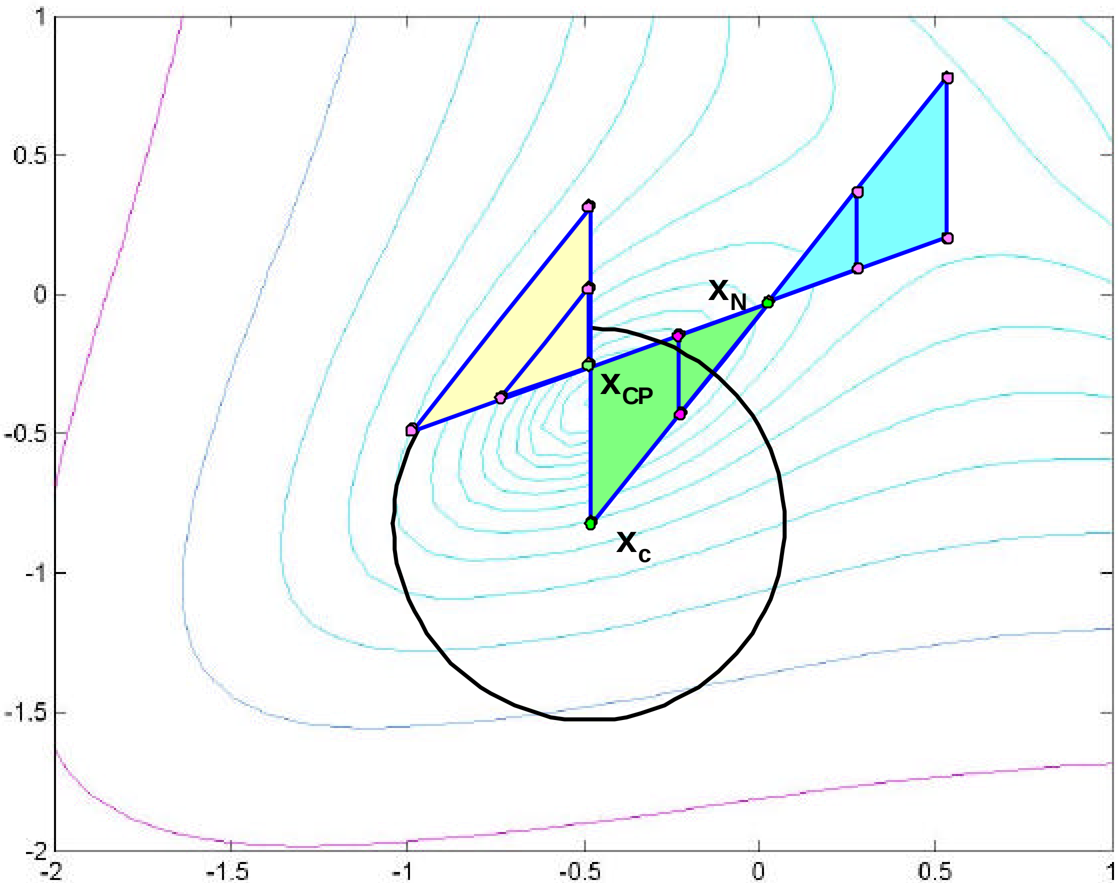


Advantages and Disadvantages

- Do not require derivative information
- Can handle noisy functions
  - Since methods only rely on function comparisons
- Inherently parallel
  - Many methods easily parallelized

- Convergence can be painfully slow
- Scant convergence theory

Can we combine direct search with Newton ideas?



- Fast convergence properties of Newton method
- Good global convergence properties of trust region approach
- Inherent parallelism of PDS
- Ability to handle noisy functions

A Class of Trust Region Methods for Parallel Optimization, P.D. Hough and J.C. Meza, to be published in SIAM Journal on Optimization

General statement of TRPDS algorithm

Given x_0, g_0, H_0, δ_0, and η

for k=0,1, ... until convergence do

1. Solve $H_k s_N = -g_k$

for i=0, 1, ... until step accepted do

2. Form initial simplex using s_N

3. Compute s that approximately minimizes $f(x_k + s)$, subject to trust region constraint

if *ared/pred* > η **then**

5. Set $x_{k+1} = x_k + s$; Evaluate g_{k+1}, H_{k+1}

endif

6. Update δ

end for

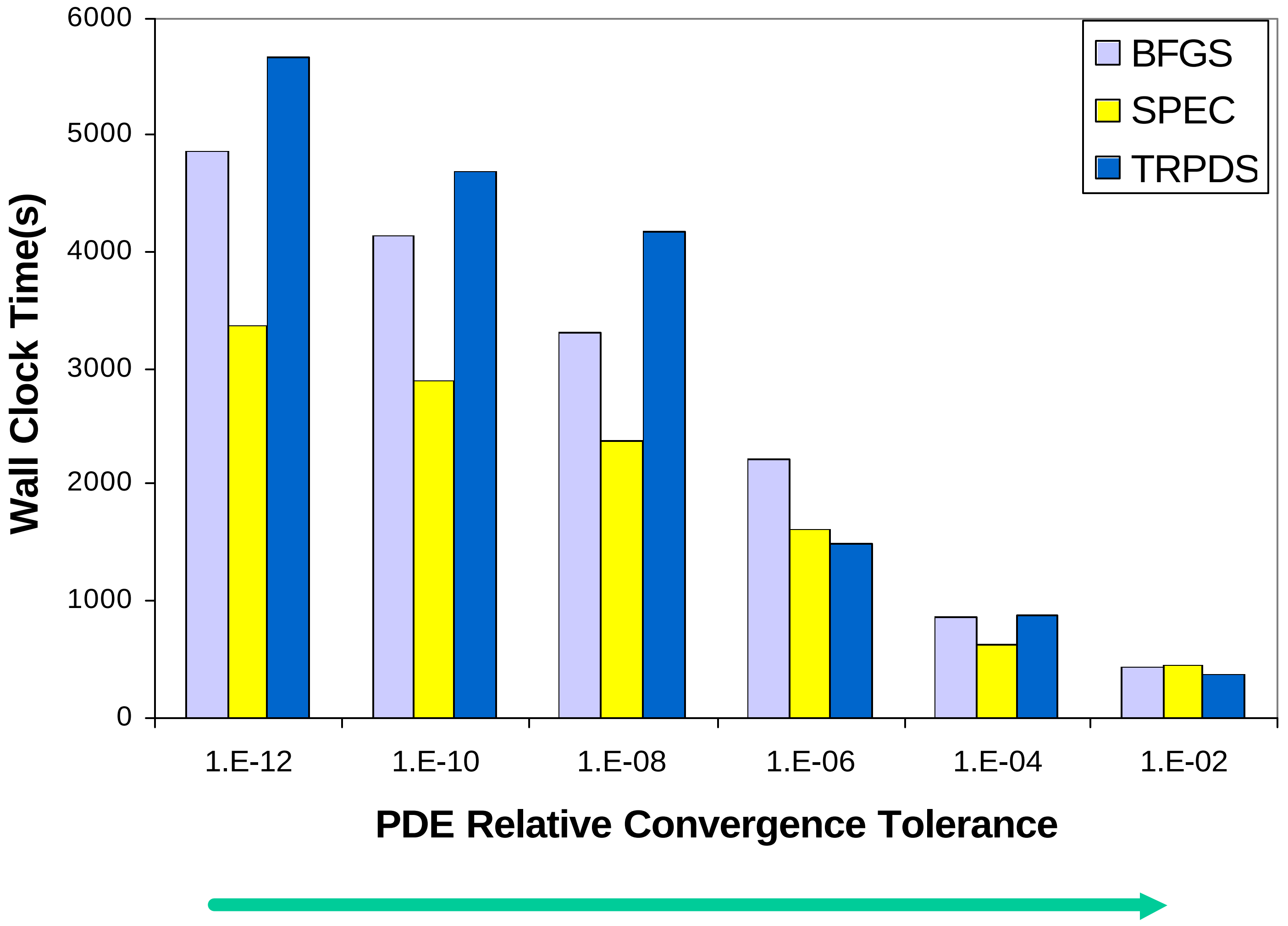
end for

A Class of Trust Region Methods for Parallel Optimization, P.D. Hough and J.C. Meza, to be published in SIAM Journal on Optimization

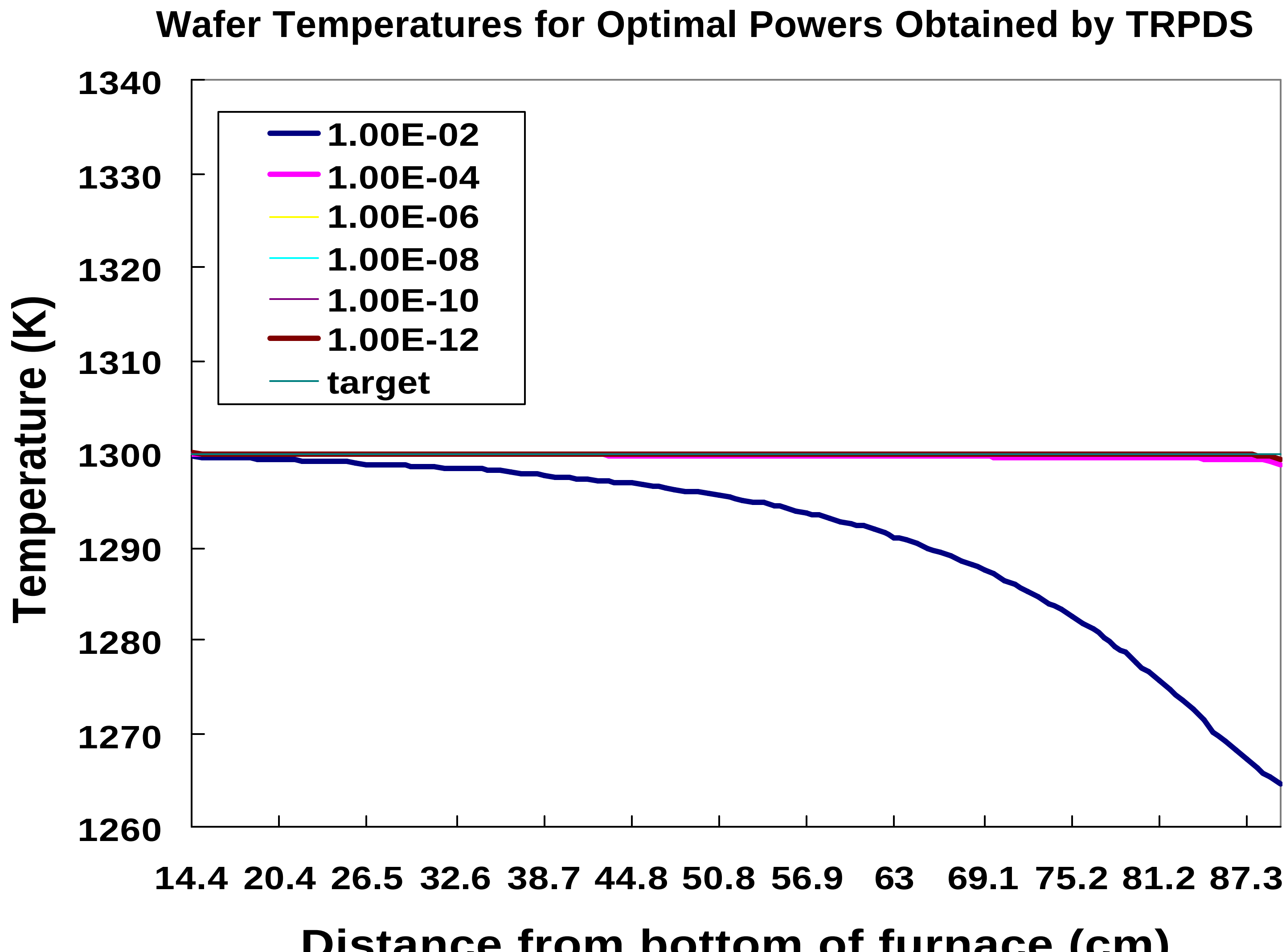
Convergence of TRPDS follows from theory of Alexandrov, Dennis, Lewis, and Torczon (1997)

- Assume
  - » Function uniformly continuously differentiable and bounded below; Hessian approximations uniformly bounded
  - » Approximation model satisfies the following conditions:
    1. $a(x_k) = f(x_k)$
    2. $\nabla a(x_k) = \nabla f(x_k)$
  - » Steps satisfy fraction of Cauchy decrease condition
- Then
  - » $\liminf_{k \to \infty} || \nabla f(x_k) || = 0$

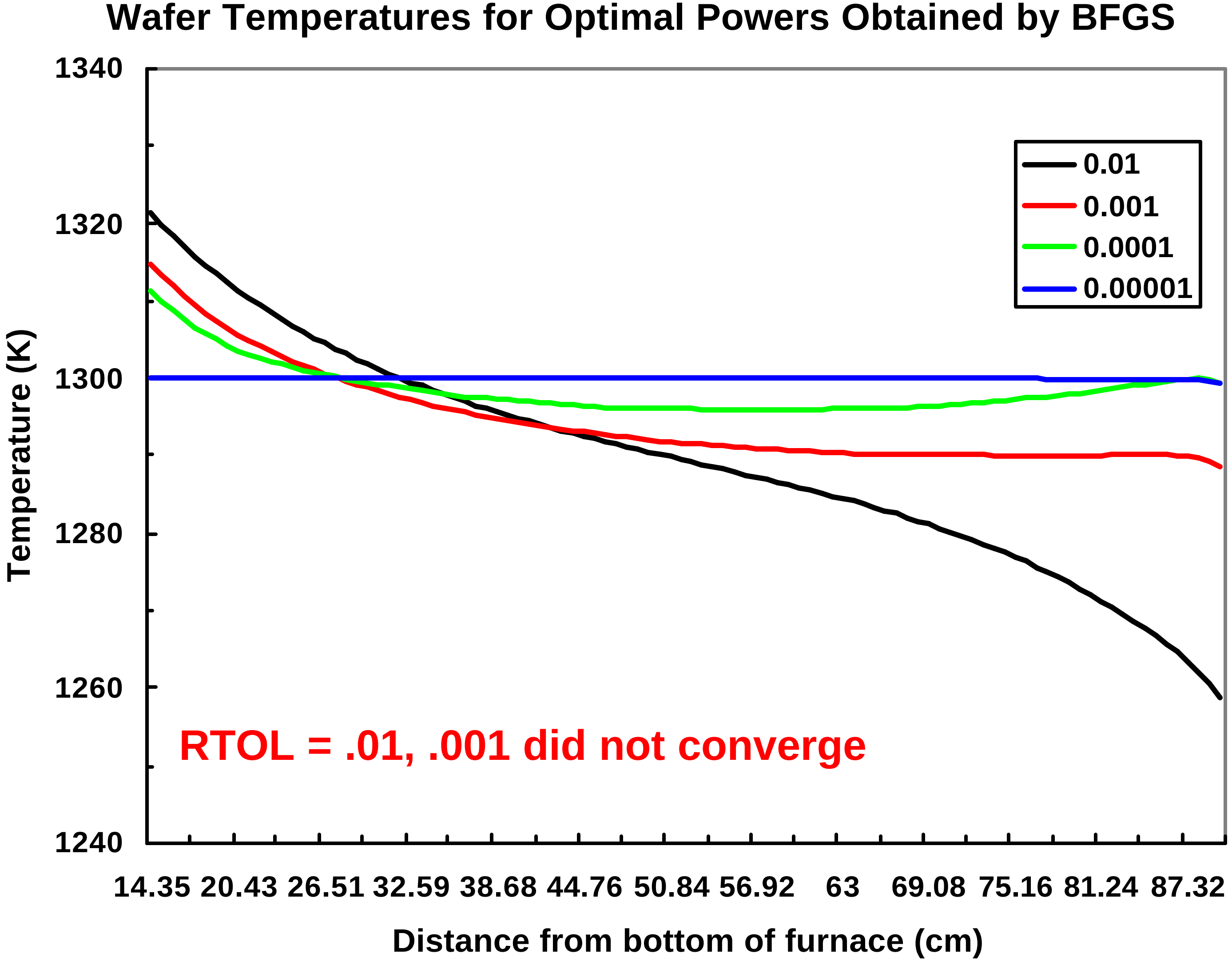
TRPDS becomes more competitive with standard methods as accuracy decreases



TRPDS is more robust than standard methods when we have fewer digits of accuracy



BFGS may not converge when simulations have fewer digits of accuracy



Constrained Optimization

Constrained Optimization Ideas

- Generally speaking a more difficult problem
  - » Hard versus soft constraints
- Bound ("box") constraints fairly easy to handle
- Linear equality constraints
  - » Various elimination techniques available
  - » Need to be careful about ill-conditioning
- Linear inequality constraints more complicated
  - » Need to guess which constraints are active at the solution

Methods for nonlinearly constrained problems have a rich history

- Barrier/Penalty methods
  - » Add terms to the objective function to "induce" the algorithms to stay feasible
  - » Have to worry about ill-conditioning
- Successive Quadratic Programming methods
  - » Reformulate problem as a sequence of quadratic programming problems
  - » Normally infeasible methods
- Interior Point methods
  - » Try to maintain feasibility by following a "central path"
  - » Good for problems with hard constraints

NIPS: Nonlinear Interior Point Solver

- Based on Newton's method for a particular system of equations (perturbed KKT equations, slack variable form)
- Can handle general nonlinear constraints
- Can handle strict feasibility

$$F(m) = \begin{bmatrix} \nabla f(x) + \nabla h(x) y - \nabla g(x) w \\ w - z \\ h(x) \\ g(x) - s \\ ZSe - me \end{bmatrix} = 0$$

Summary

- Optimization arises in many applications
  - » Parameter identification
  - » Optimal Design/Control
  - » Minimization
- Practical problems often exhibit characteristics that make standard methods difficult/impossible to use
- Few good methods for parallel optimization
  - » Many possibilities for new research topics

Future Directions

- Better interior point methods
- Mixed integer nonlinear programming problems
- Optimization under uncertainty
- Surrogate methods for expensive functions
- Non-smooth optimization
- ...

Optimization References

- Dennis and Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, 1983
- Gill, Murray, Wright, *Practical Optimization*, Academic Press, 1981
- Stephen J. Wright, *Primal-Dual Interior-Point Methods*, SIAM 1997
- El-Bakry, Tapia, Tsuchiya, Zhang, *On the Formulation and Theory of the Newton Interior-Point Method for Nonlinear Programming*, JOTA, Vol. 89, No.3, pp.507-541, 1996
- M.J.D. Powell, *Direct search algorithms for optimization calculations*, Acta Numerica 1998
- M.H. Wright, *Direct search methods: once scorned, now respectable*, Numerical Analysis, 1995
- P. D. Hough, T. G. Kolda, and V. J. Torczon. *Asynchronous Parallel Pattern Search for Nonlinear Optimization.* SIAM J. Scientific Computing, 23(1):134-156, June 2001

Software References

- DOE ACTS Collection
  - » http://acts.nersc.gov/
- APPSPACK
  - » http://csmr.ca.sandia.gov/projects/apps.html
- NEOS – Network Enabled Optimization Software
  - » http://www-neos.mcs.anl.gov/neos
- General Software
  - » http://sal.kachinatech.com/B/3/index.shtml
- Moré and Wright, Optimization Software Guide, SIAM, 1993

The End