Derivative-free optimization for parameter estimation in computational nuclear physics

Stefan Wild

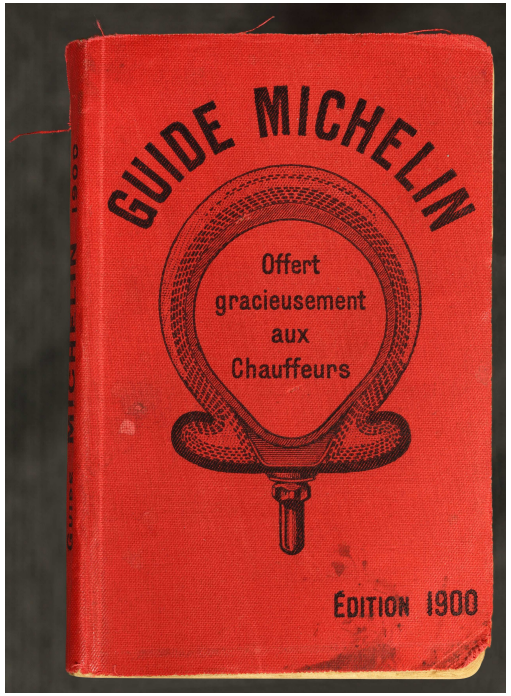
Argonne National Laboratory
Mathematics and Computer Science Division

Joint work with Jorge Moré, Todd Munson, and Jason Sarich
and grateful to many physicist collaborators:
A. Ekström, C. Forssén, G. Hagen, M. Hjorth-Jensen, G.R. Jansen, M. Kortelainen, T. Lesinski, R. Machleidt, J. McDonnell, W. Nazarewicz, E. Olsen, T. Papenbrock, P.-G. Reinhardt, N. Schunck, M. Stoitsov, J. Vary, K. Wendt, and others

November 24, 2014

Roadmap

- → Simulation-based optimization problems
- → Calibration of EDFs
- → The POUNDERS algorithm and software
- → Revisit Optimization and Uncertainties from UNEDF
- → Optimization with the new Argonne Masses



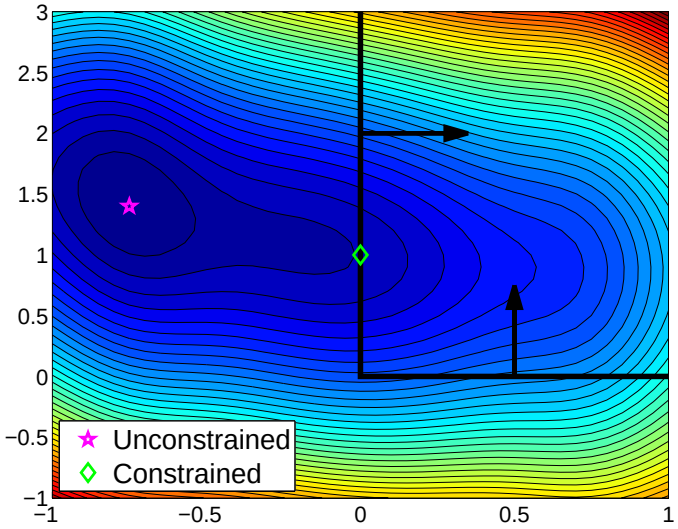
Mathematical/Numerical Nonlinear Optimization

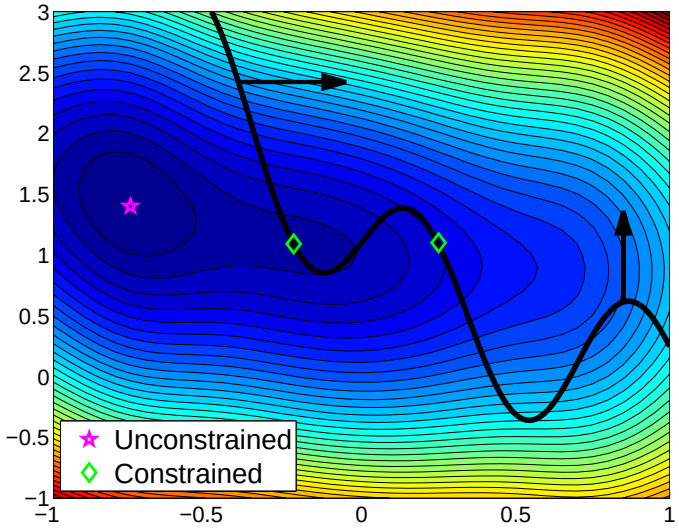
Optimization is the *"science of better"*

Find parameters (controls) $x = (x_1, \ldots, x_n)$ in domain Ω to improve objective f

$$\min \{ f(x) : x \in \Omega \subseteq \mathbb{R}^n \}$$

- (Unless Ω is very special) Need to evaluate f at many x to find a good $\hat{x}_*$
- Focus on local solutions: $f(\hat{x}_*) \leq f(x) \; \forall x \in \mathcal{N}(\hat{x}_*) \cap \Omega$






Argonne National Laboratory


IBM
MIRA

$\min_{x \in \mathbb{R}^n} \{ f(x) = F[x, S(x)] : c_S(x) = 0, x \in \Omega \}$

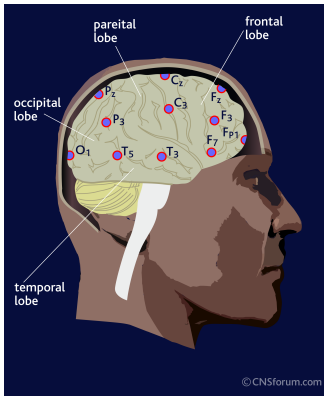
Optimize expensive, nonlinear functions arising in science & engineering

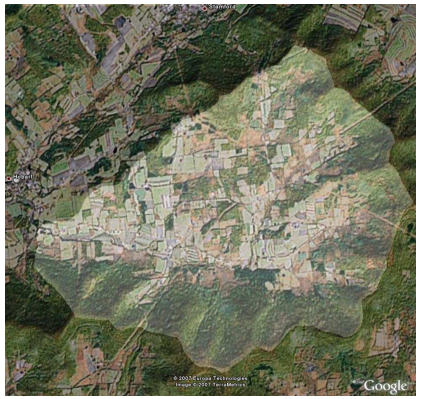
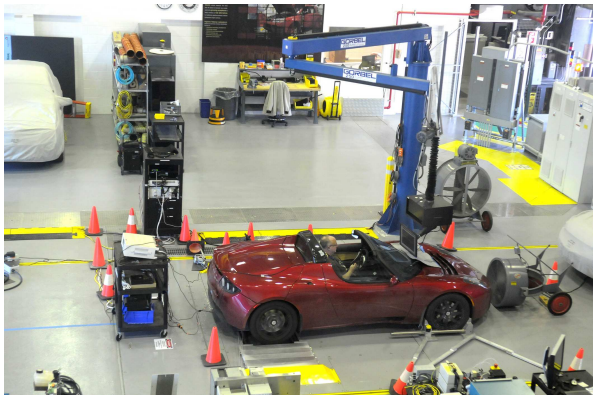
"parameter estimation", "model calibration", "design optimization", ...

- $f : \mathbb{R}^n \to \mathbb{R}$ objective, c constraints, $S : \mathbb{R}^n \to \mathbb{R}^p$ numerical simulation,
- Evaluating S means running a simulation modeling some (smooth) process

  Ex- S = solving PDEs via finite elements
  - Here: assume f is from a deterministic computer simulation
- S can contribute to objective and/or constraints, possibly noisy
- S (could/must be parallelized) takes secs/mins/hrs for 1 x

Evaluation is a bottleneck for optimization

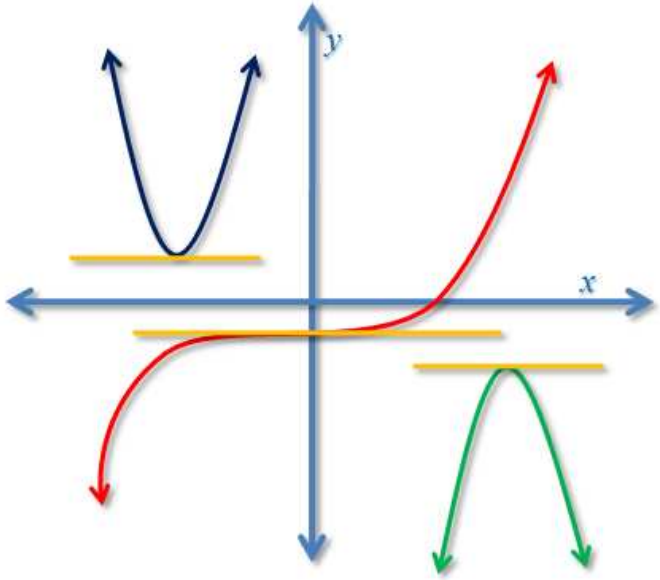


Optimization Tightly Coupled With Derivatives

Typical optimality (no noise, smooth functions)

$$\nabla_x f(x_*) + \lambda^T \nabla_x c_E(x_*) = 0, c_E(x_*) = 0$$

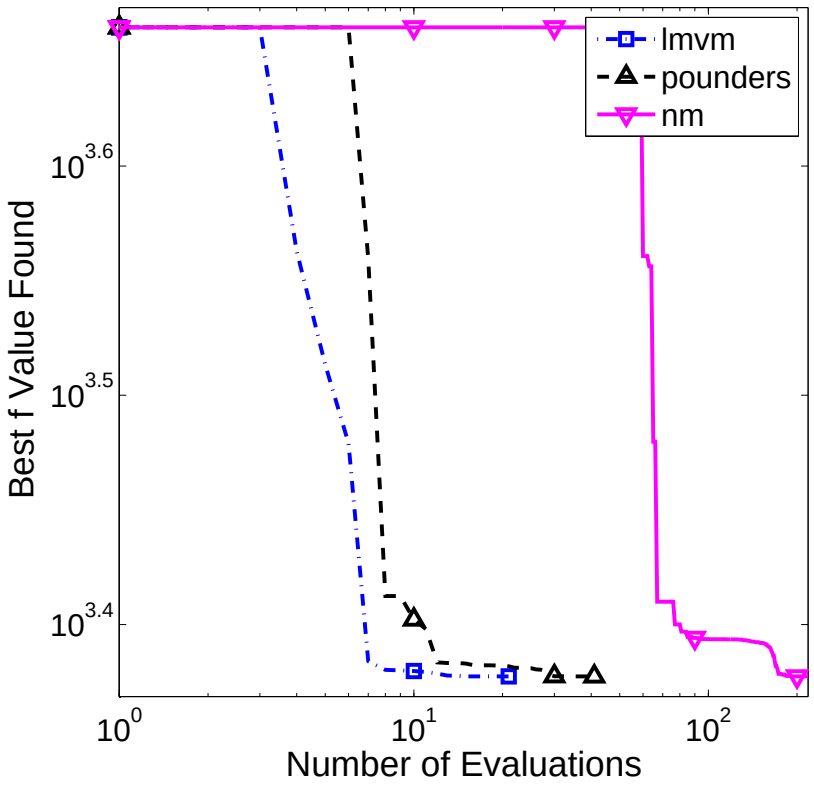


(sub)gradients $\nabla_x f$, $\nabla_x c$ enable:

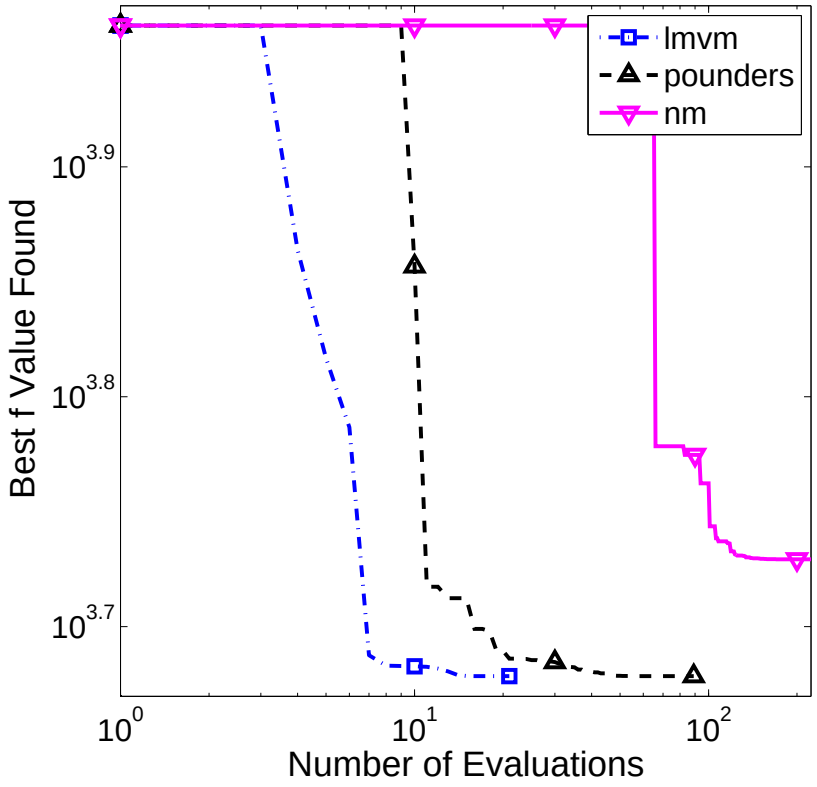
- Faster feasibility
- Faster convergence
  - Guaranteed descent
  - Approximation of nonlinearities
- Better termination
  - Measure of criticality $\|\nabla_x f\|$ or $\|\mathcal{P}_\Omega(\nabla_x f)\|$

But derivatives $\nabla_x S(x)$ are not always available/do not always exist

The Price of Algorithm Choice (I)



$n = 3$



$n = 6$

lmvm Uses available $\nabla_x f$

nm Assumes $\nabla_x f$ unavailable

pounders Assumes $\nabla_x f$ unavailable, exploits problem structure

Example: Calibration of EDFs ("χ^2 minimization")

$$\min_{x \in \Omega} f(x) = \sum_{i=1}^{p} \left(\frac{s_i(x;\theta_i) - d_i}{w_i} \right)^2$$

$s_i(x;\theta)$ Simulated nucleus observable

e.g., energies, radii, odd-even mass diffs, energy gaps, single particle energies

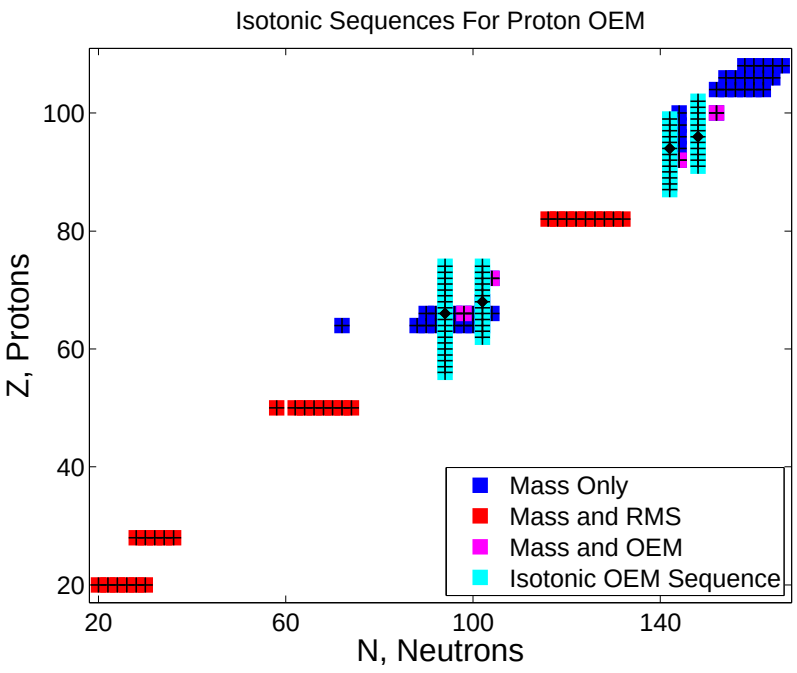
$\rightarrow$ HFBTHO code output

d_i Observable i (exp.) data

w_i Weight/error for data type i

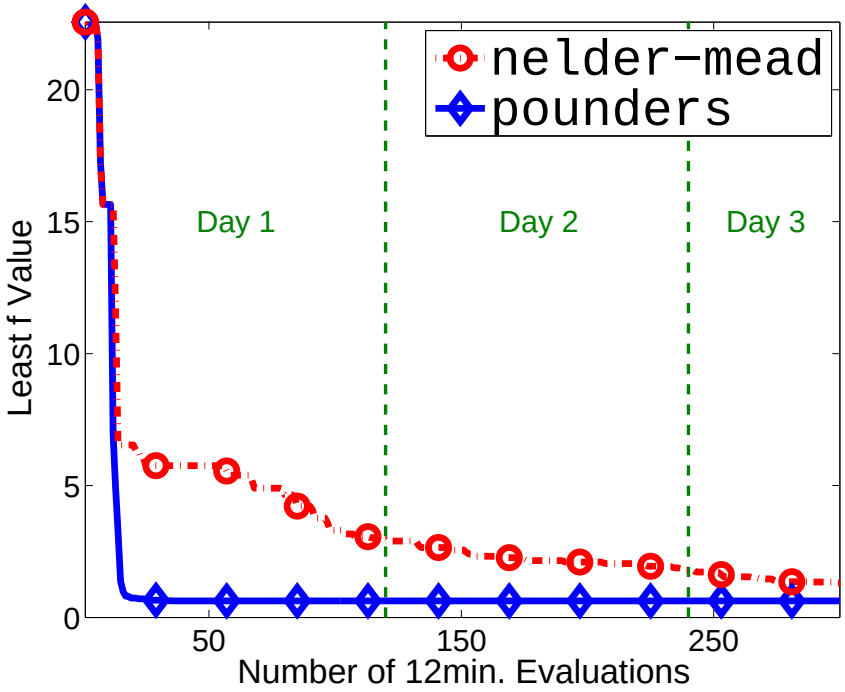
x Free parameters (coupling constants)

Ω Bound constraints

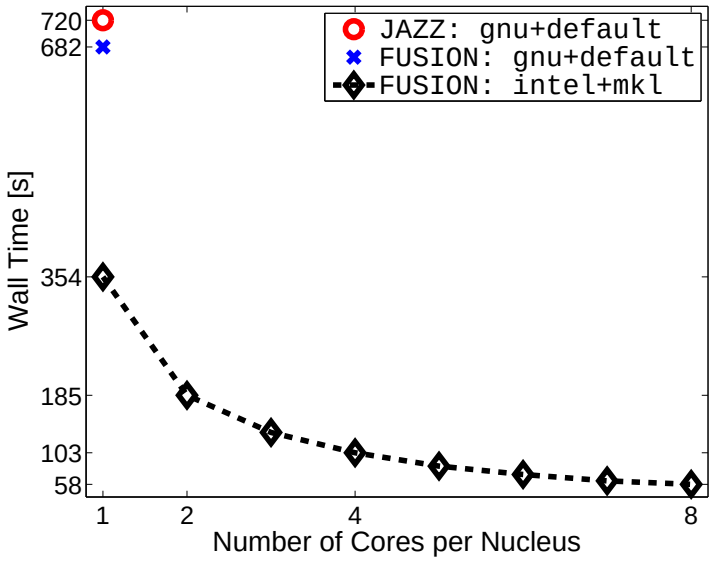


Observables in UNEDF0

The Price of Algorithm Choice (II)



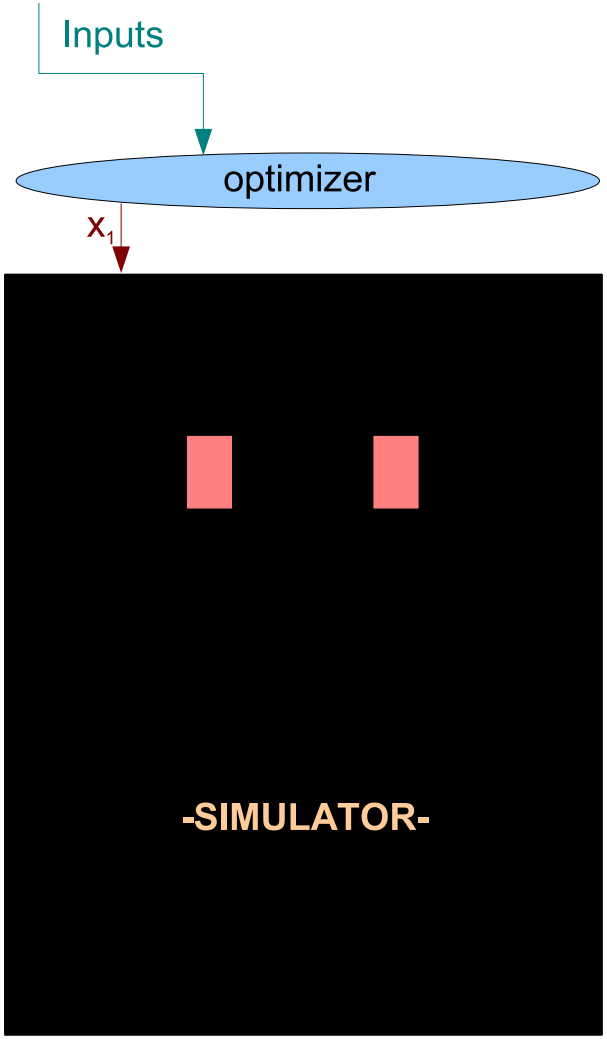
Significantly reduced the number of DFT simulations required



- UNEDF0: [Kortelainen et al., PRC 2010]
- UNEDF1: [Kortelainen et al., PRC 2012]
- UNEDF2: [Kortelainen et al., PRC 2014]

Algorithms

"Simplest" Formulation: Blackbox f



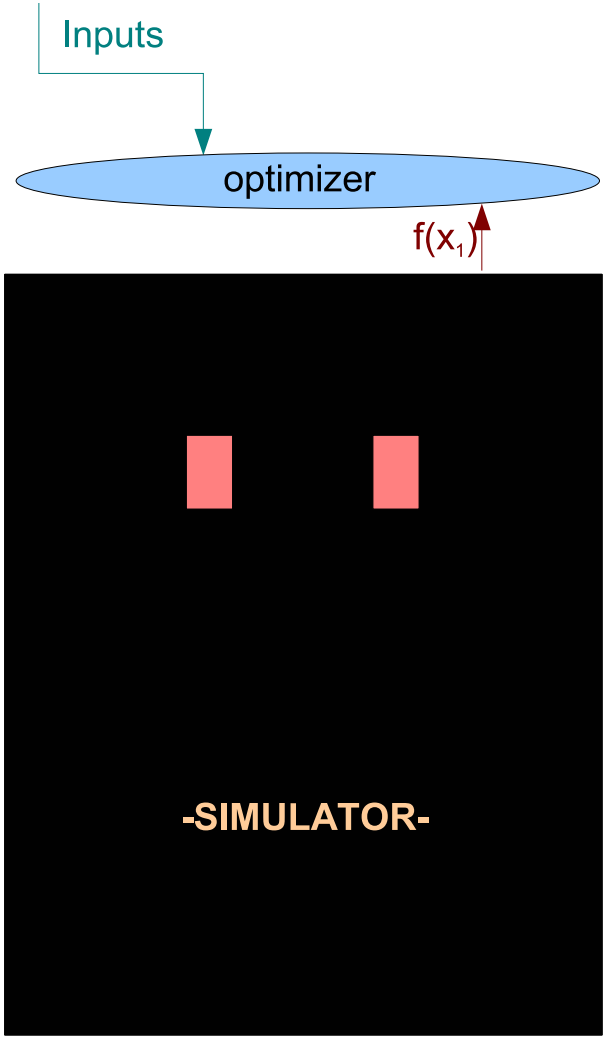
Optimizer gives x, physicist provides $f(x)$

- f can be a blackbox (executable only or proprietary/legacy codes)
- Only give a single output
  - no derivatives with respect to x: $\nabla_x S(x), \nabla^2_{x,x} S(x)$
  - no problem structure

Good solutions guaranteed in the limit, but:

- Usually have computational budget (due to scheduling, finances, deadlines)
- Limited number of evaluations

"Simplest" Formulation: Blackbox f



Optimizer gives x, physicist provides $f(x)$

- f can be a blackbox (executable only or proprietary/legacy codes)
- Only give a single output
  - no derivatives with respect to x: $\nabla_x S(x), \nabla^2_{x,x} S(x)$
  - no problem structure

Good solutions guaranteed in the limit, but:

- Usually have computational budget (due to scheduling, finances, deadlines)
- Limited number of evaluations

Derivative-Free, Local Algorithms

Solve general problems $\min\{f(x) : x \in \mathbb{R}^n\}$:

- Only require function values (no $\nabla f(x)$)
- Don't rely on finite-difference approximations to $\nabla f(x)$
  - Can be misleading due to noise
  - Can be inefficient (each set of $n+1$ evaluations useful for a single step only)
- Seek greedy and rapid decrease of function value

Derivative-Free, Local Algorithms

Solve general problems $\min\{f(x) : x \in \mathbb{R}^n\}$:

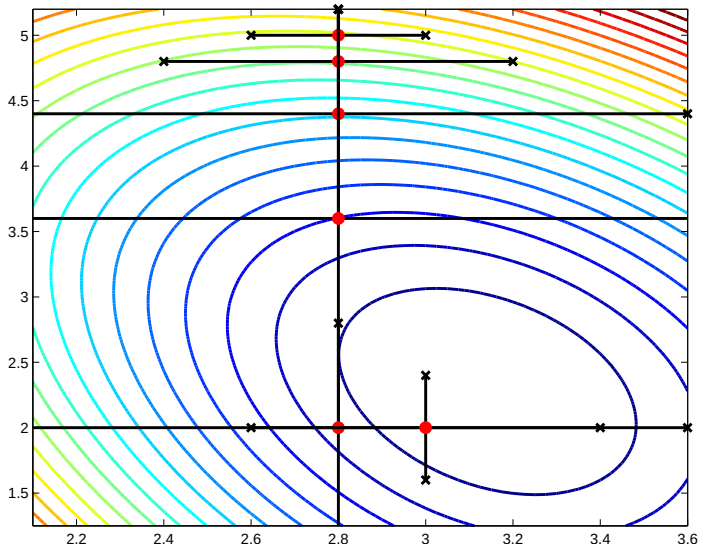
- Only require function values (no $\nabla f(x)$)
- Don't rely on finite-difference approximations to $\nabla f(x)$
  - Can be misleading due to noise
  - Can be inefficient (each set of $n+1$ evaluations useful for a single step only)
- Seek greedy and rapid decrease of function value

Two main styles of local DFO algorithms

- Direct search methods (pattern search, Nelder-Mead, ...)
- Model-based methods (quadratics, radial basis functions, ...)

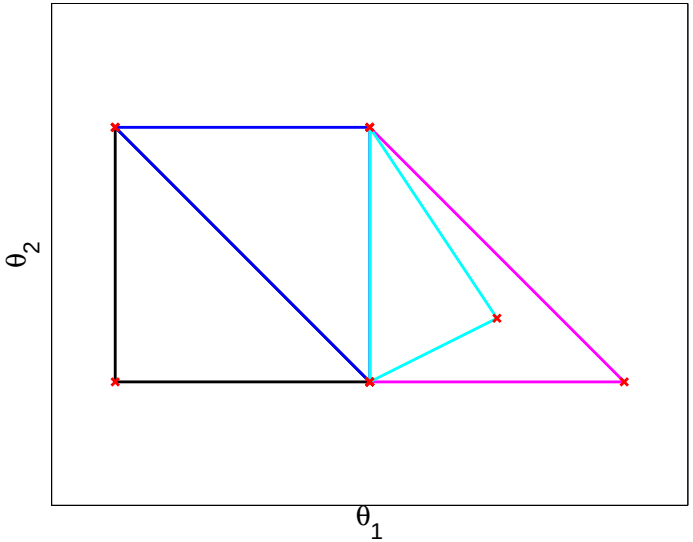
Algorithms: Direct Search Methods

Pattern Search



Easy to parallelize f evaluations

Nelder-Mead



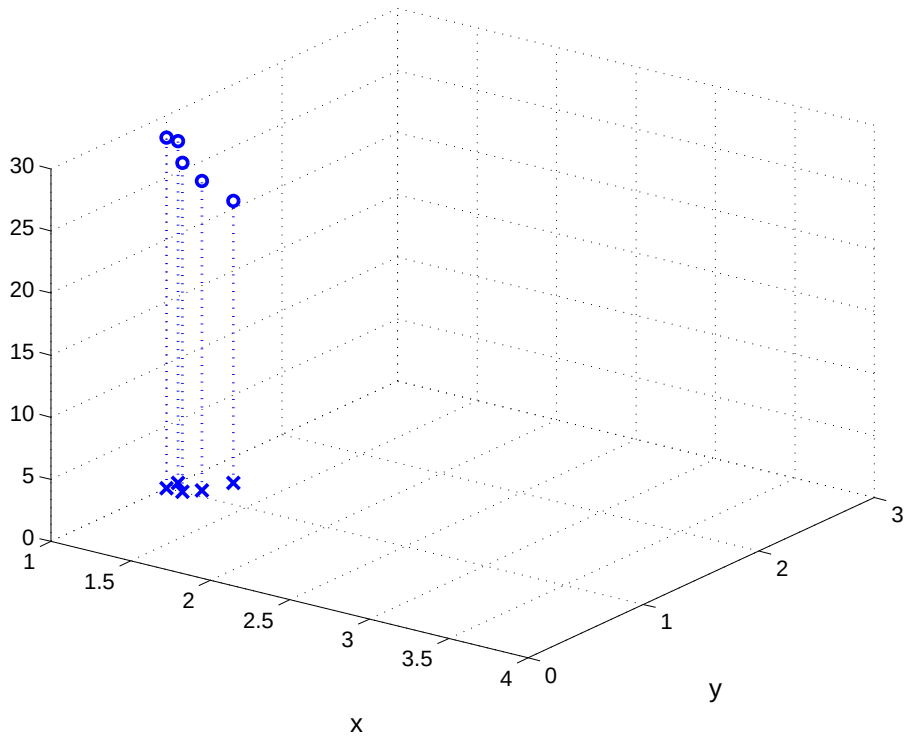
Popularized by *Numerical Recipes*

- Rely on indicator functions: $[f(x_k + s) <^? f(x_k)]$
- Work with **black-box** $f(x)$, do not exploit structure $F[x, S(x)]$

$\longrightarrow$[Kolda, Lewis, Torczon, SIREV 2003]

Making the Most of Little Information on f

- f is expensive $\Rightarrow$ can afford to make better use of points
- Overhead of the optimization routine is negligible relative to the cost of evaluating the simulation.



Bank of data, $\{x_i, f(x_i)\}_{i=1}^{k}$:

= Points (& function values) evaluated so far

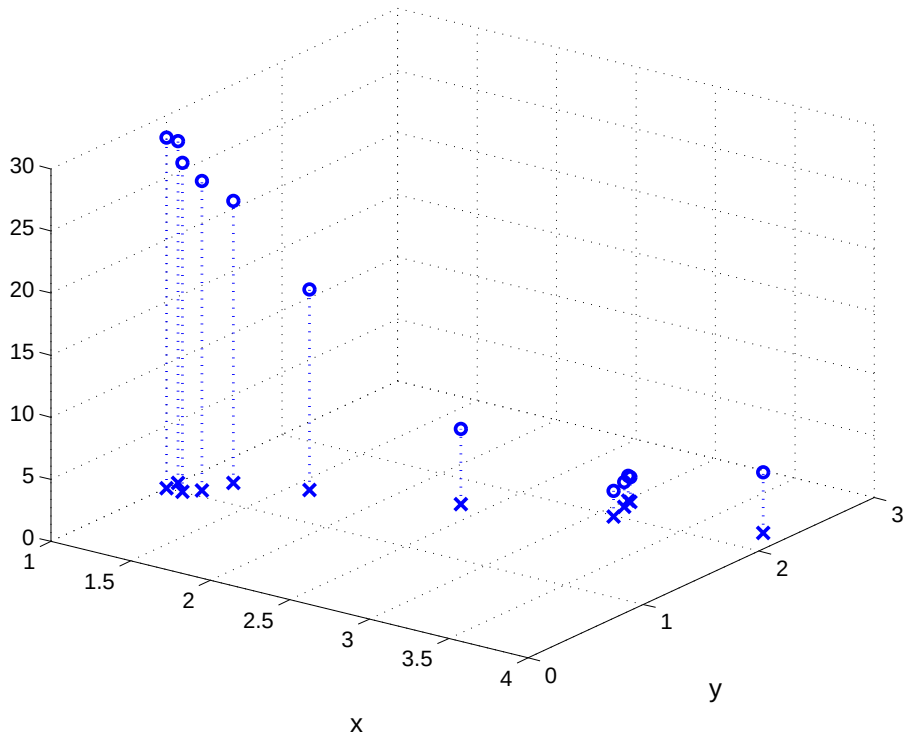
= Everything known about f

Goal:

- Make use of growing Bank as optimization progresses

Making the Most of Little Information on f

- f is expensive $\Rightarrow$ can afford to make better use of points
- Overhead of the optimization routine is negligible relative to the cost of evaluating the simulation.



Bank of data, $\{x_i, f(x_i)\}_{i=1}^{k}$:

= Points (& function values) evaluated so far

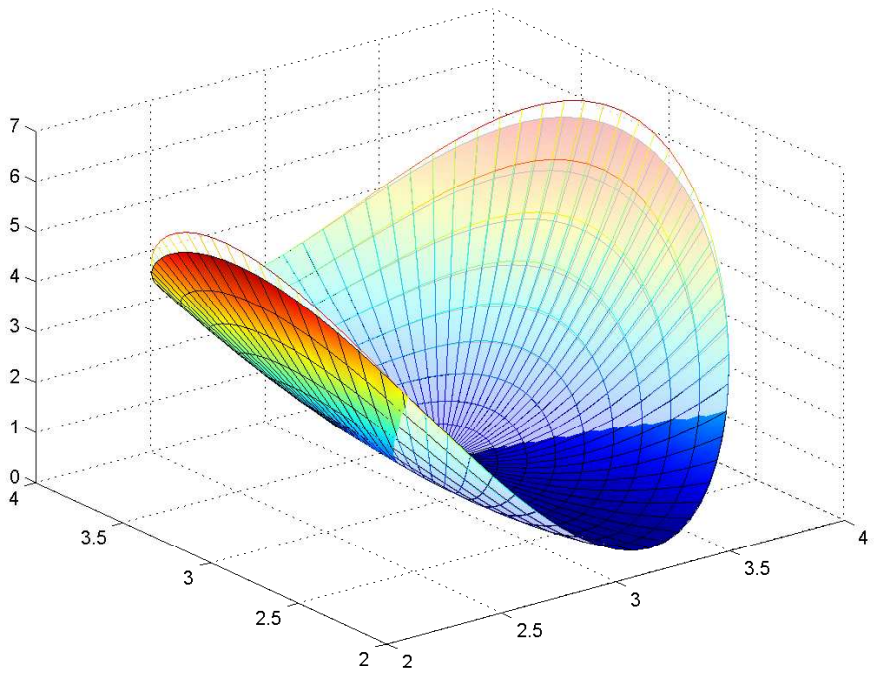
= Everything known about f

Goal:

- Make use of growing Bank as optimization progresses

Trust-Region Methods Use Models Instead of f

To reduce the number of expensive f evaluations
$\rightarrow$ Replace difficult optimization problem $\min f(x)$ with a much simpler one
$\min \{\mathbf{m}(\mathbf{x}) : \mathbf{x} \in \mathcal{B}\}$

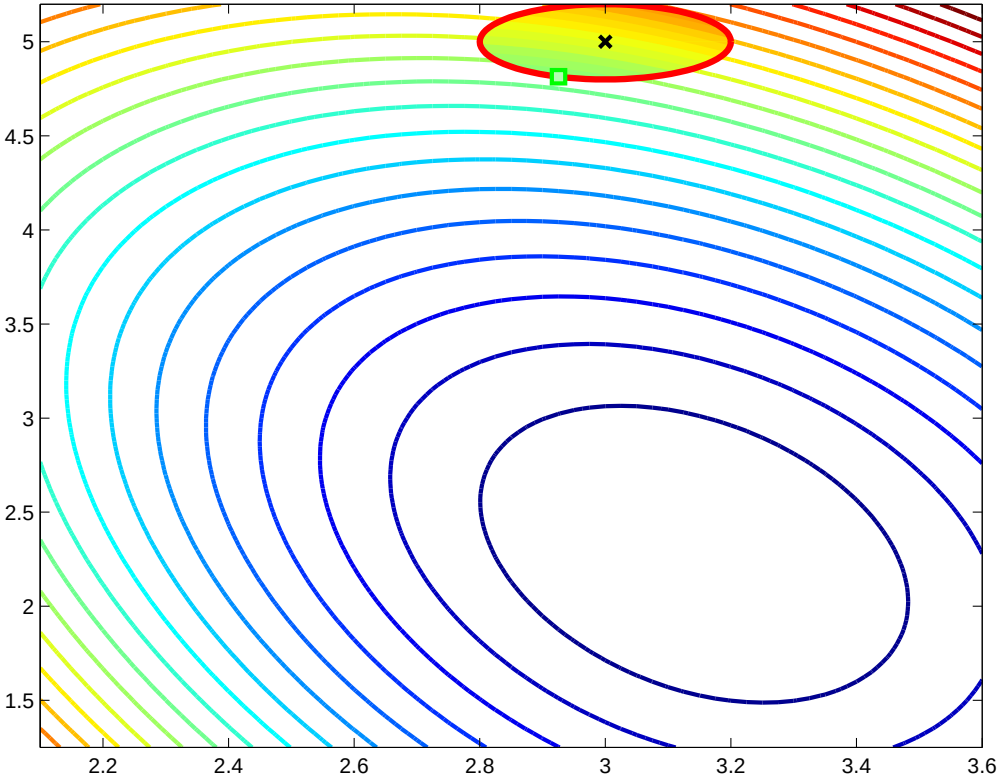


Classic NLP Technique:

- f Original function: computationally expensive, no derivatives
- m Surrogate model: computationally attractive, analytic derivatives

Basic Trust-Region Idea

Use a surrogate $\mathbf{m}(\mathbf{x})$ in place of the unwieldy $f(x)$

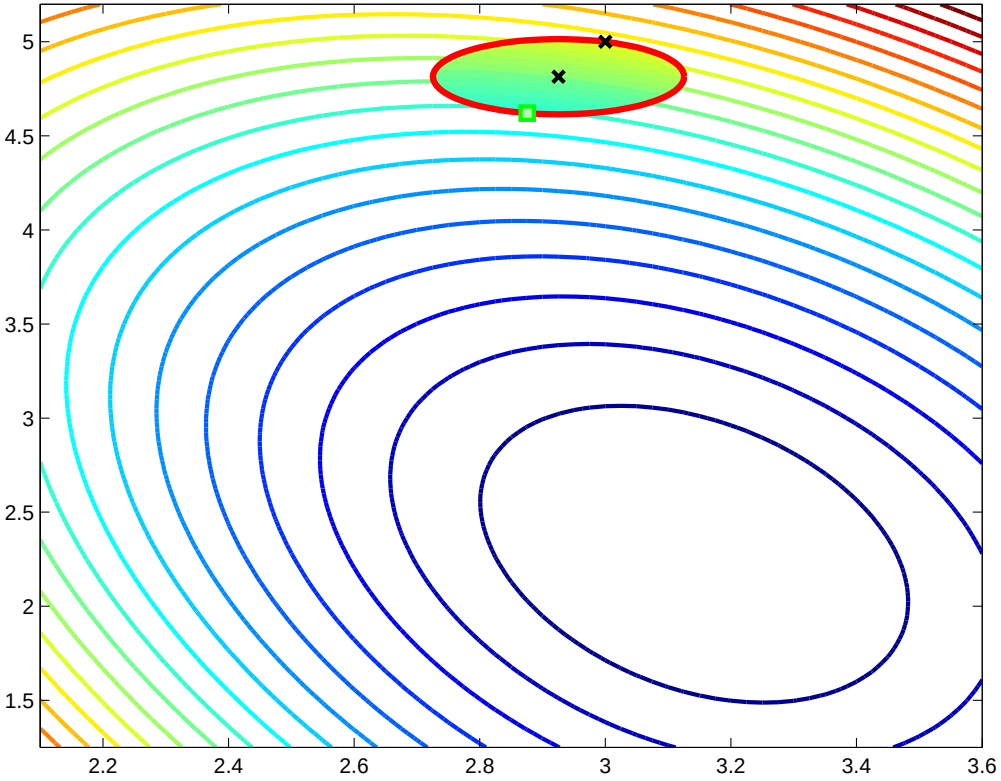


Optimize over $\mathbf{m}$ to avoid expense of f:

- Trust m to approximate f within $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$,
- Obtain next point from $\min\{m(x) : x \in \mathcal{B}\}$,
- Evaluate function and update (x_k, Δ_k) based on how good the model's prediction was.

Basic Trust-Region Idea

Use a surrogate $\mathbf{m}(\mathbf{x})$ in place of the unwieldy $f(x)$

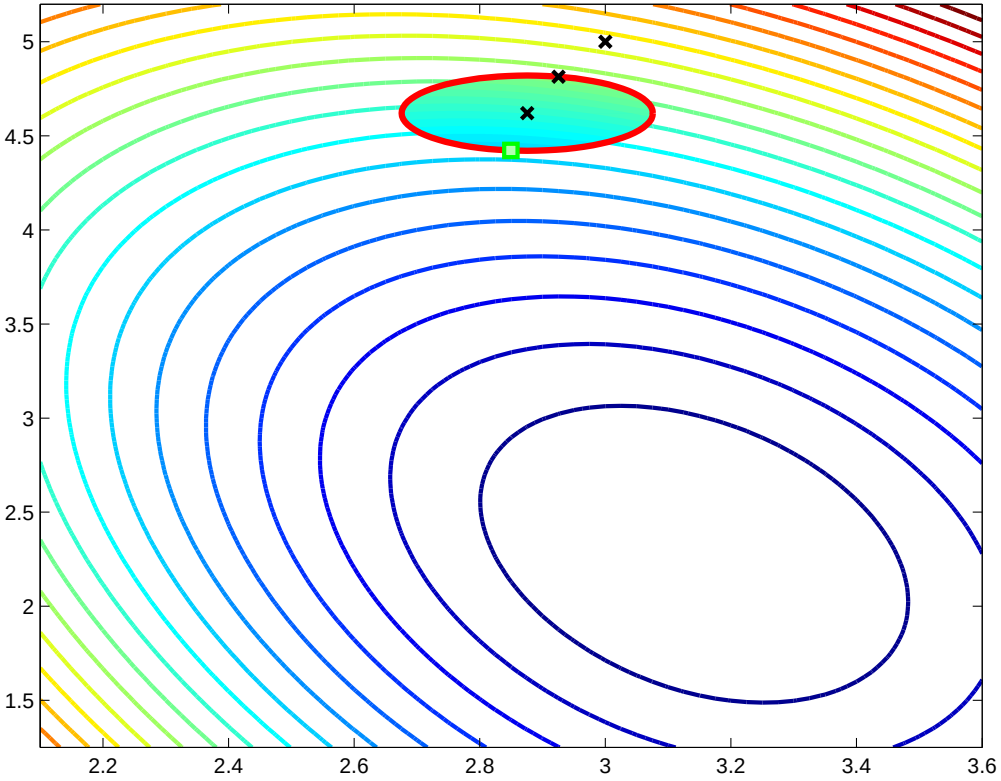


Optimize over $\mathbf{m}$ to avoid expense of f:

- Trust m to approximate f within $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$,
- Obtain next point from $\min\{m(x) : x \in \mathcal{B}\}$,
- Evaluate function and update (x_k, Δ_k) based on how good the model's prediction was.

Basic Trust-Region Idea

Use a surrogate $\mathbf{m}(\mathbf{x})$ in place of the unwieldy $f(x)$

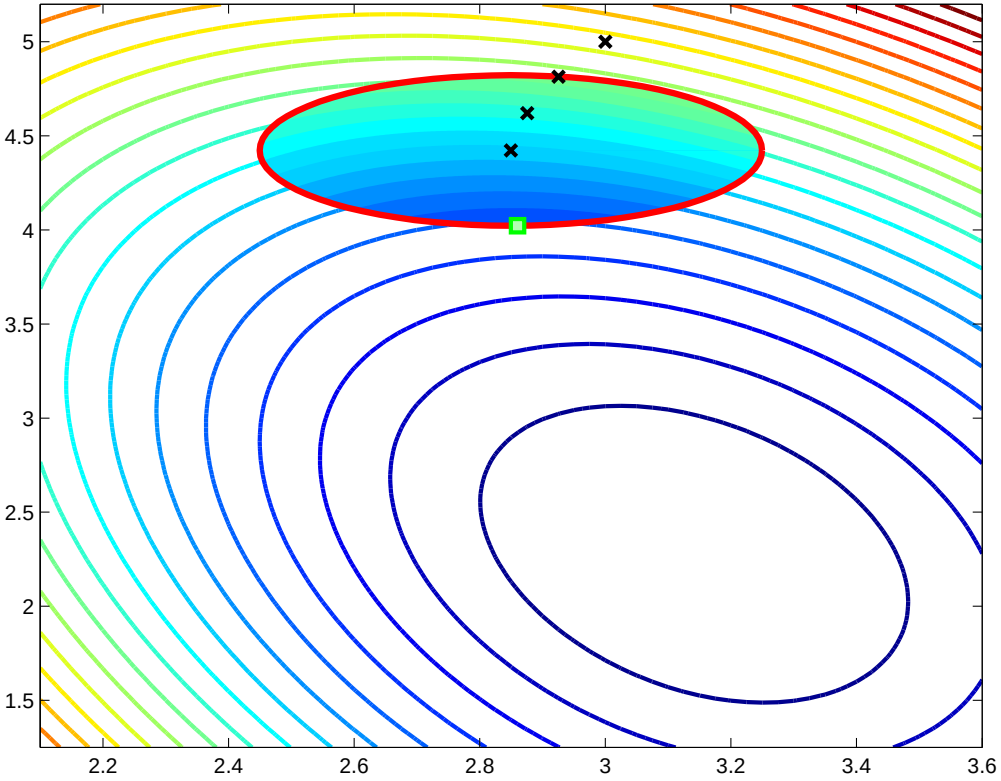


Optimize over $\mathbf{m}$ to avoid expense of f:

- Trust m to approximate f within $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$,
- Obtain next point from $\min\{m(x) : x \in \mathcal{B}\}$,
- Evaluate function and update (x_k, Δ_k) based on how good the model's prediction was.

Basic Trust-Region Idea

Use a surrogate $\mathbf{m}(\mathbf{x})$ in place of the unwieldy $f(x)$

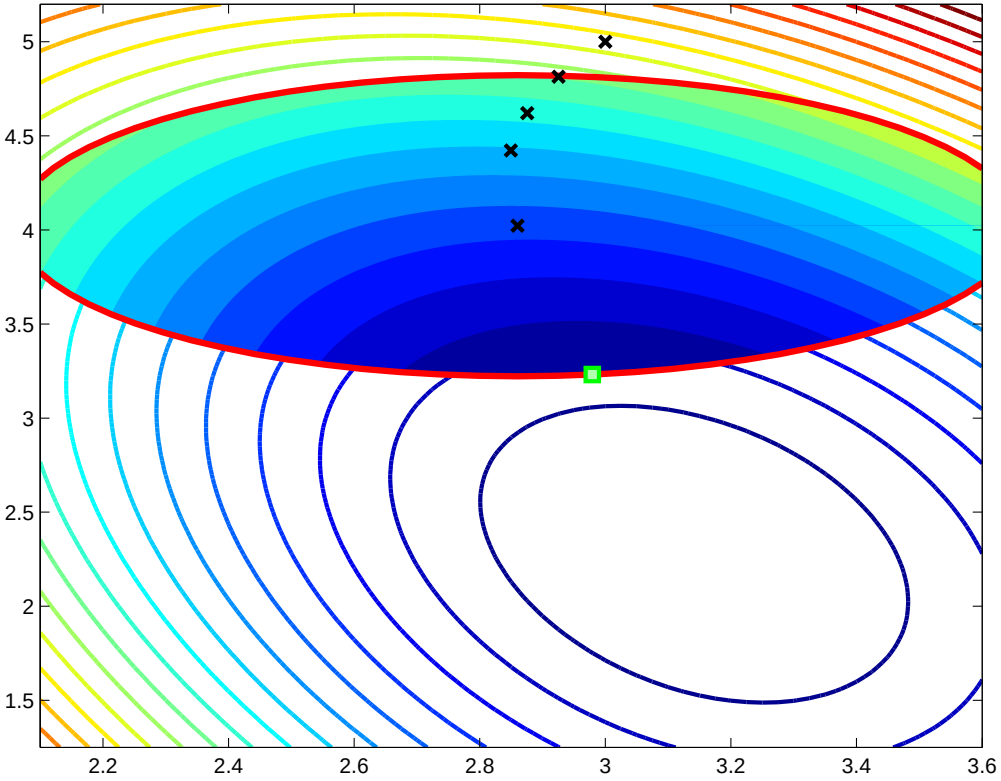


Optimize over $\mathbf{m}$ to avoid expense of f:

- Trust m to approximate f within $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$,
- Obtain next point from $\min\{m(x) : x \in \mathcal{B}\}$,
- Evaluate function and update (x_k, Δ_k) based on how good the model's prediction was.

Basic Trust-Region Idea

Use a surrogate $\mathbf{m}(\mathbf{x})$ in place of the unwieldy $f(x)$

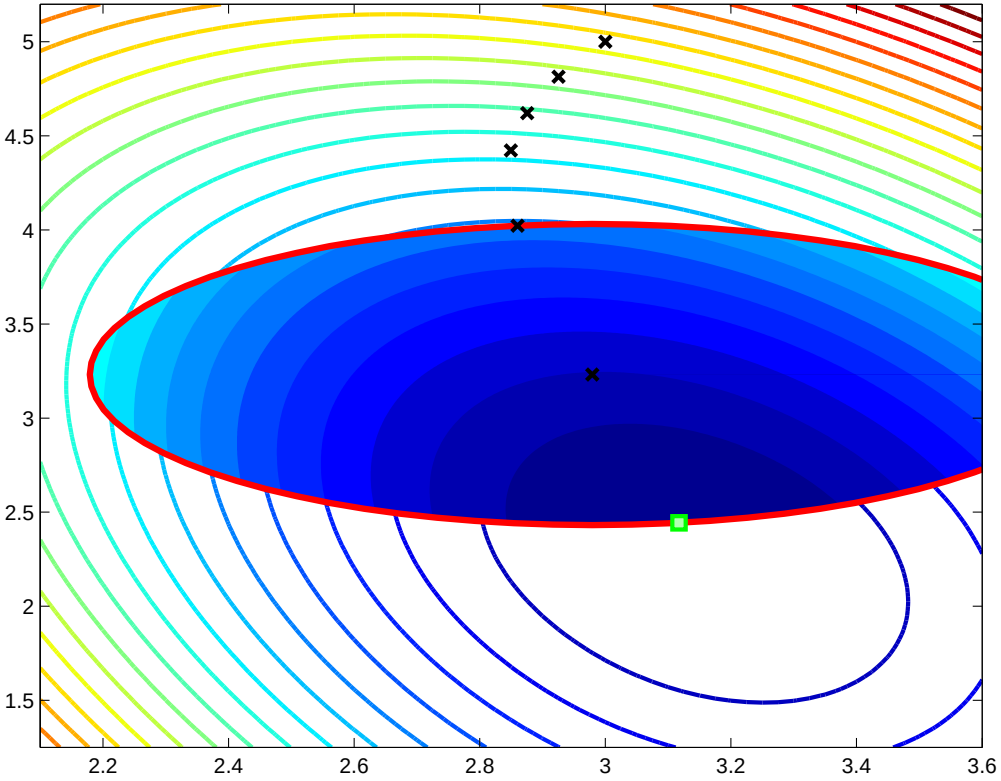


Optimize over $\mathbf{m}$ to avoid expense of f:

- Trust m to approximate f within $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$,
- Obtain next point from $\min\{m(x) : x \in \mathcal{B}\}$,
- Evaluate function and update (x_k, Δ_k) based on how good the model's prediction was.

Basic Trust-Region Idea

Use a surrogate $\mathbf{m}(\mathbf{x})$ in place of the unwieldy $f(x)$

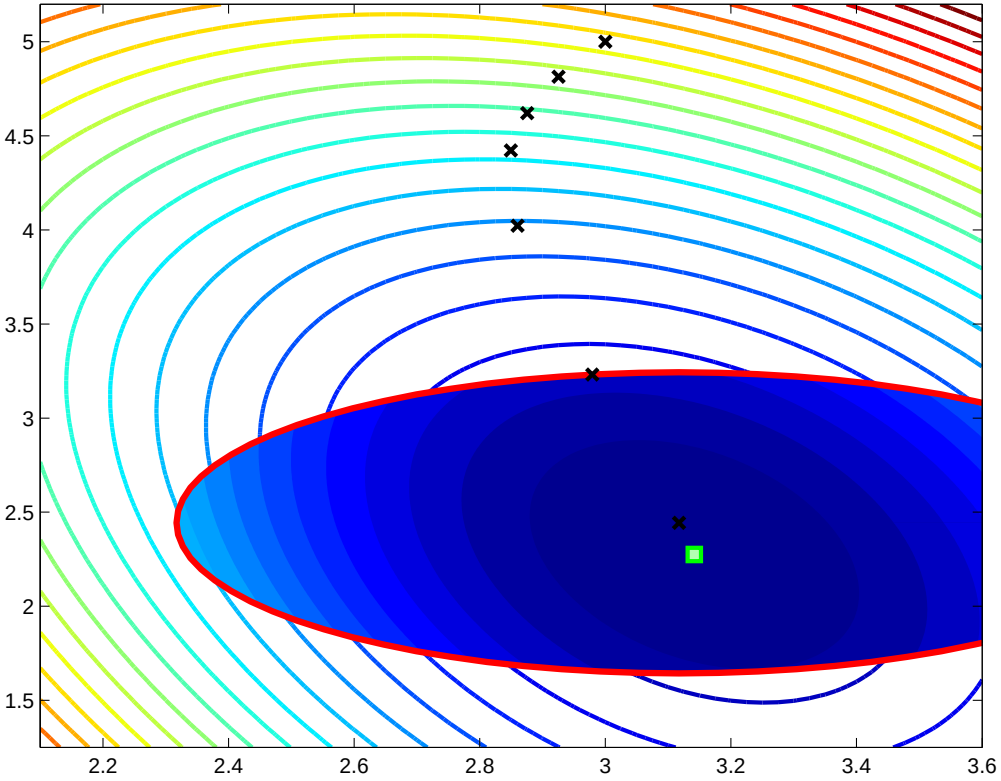


Optimize over $\mathbf{m}$ to avoid expense of f:

- Trust m to approximate f within $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$,
- Obtain next point from $\min\{m(x) : x \in \mathcal{B}\}$,
- Evaluate function and update (x_k, Δ_k) based on how good the model's prediction was.

Basic Trust-Region Idea

Use a surrogate $\mathbf{m}(\mathbf{x})$ in place of the unwieldy $f(x)$

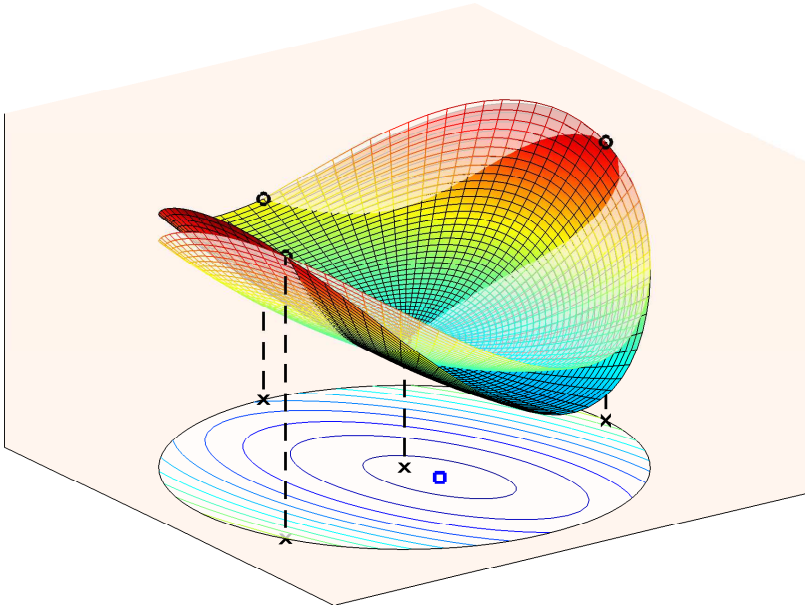


Optimize over $\mathbf{m}$ to avoid expense of f:

- Trust m to approximate f within $\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}$,
- Obtain next point from $\min\{m(x) : x \in \mathcal{B}\}$,
- Evaluate function and update (x_k, Δ_k) based on how good the model's prediction was.

Building Models Without Derivatives

$$m_k(x) = f(x_k) + g_k^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top H_k (x - x_k)$$



$n = 2$, $|\mathcal{Y}_k| = 4$

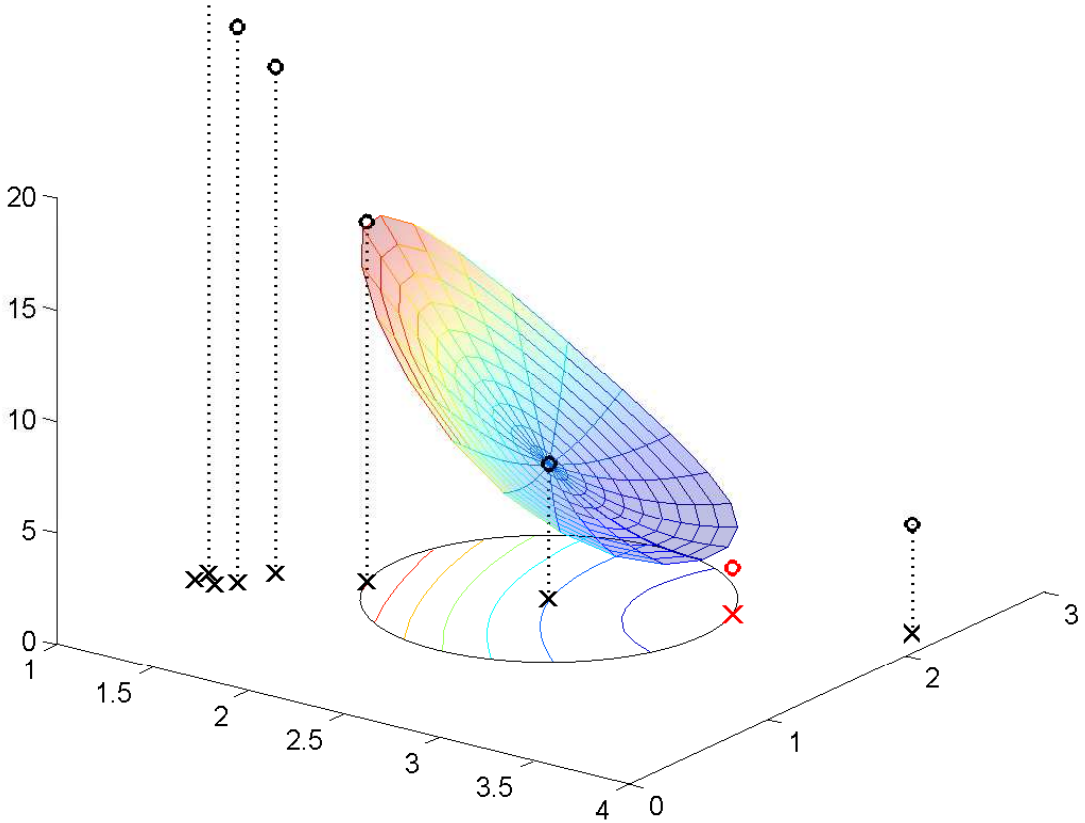
Determine (g_k, H_k) from function values

- Only need local approximation
- Numerical differentiation too expensive ($\frac{(n+1)(n+2)}{2}$ evaluations per iteration for full quadratics)
- $m_k(y_i) = f(y_i), \quad \forall y_i \in \mathcal{Y}_k$

Practical concerns

- Interpolation of scattered data (ideally, from the history of the algorithm)
- Models typically underdetermined, $|\mathcal{Y}_k| < \frac{(n+1)(n+2)}{2}$

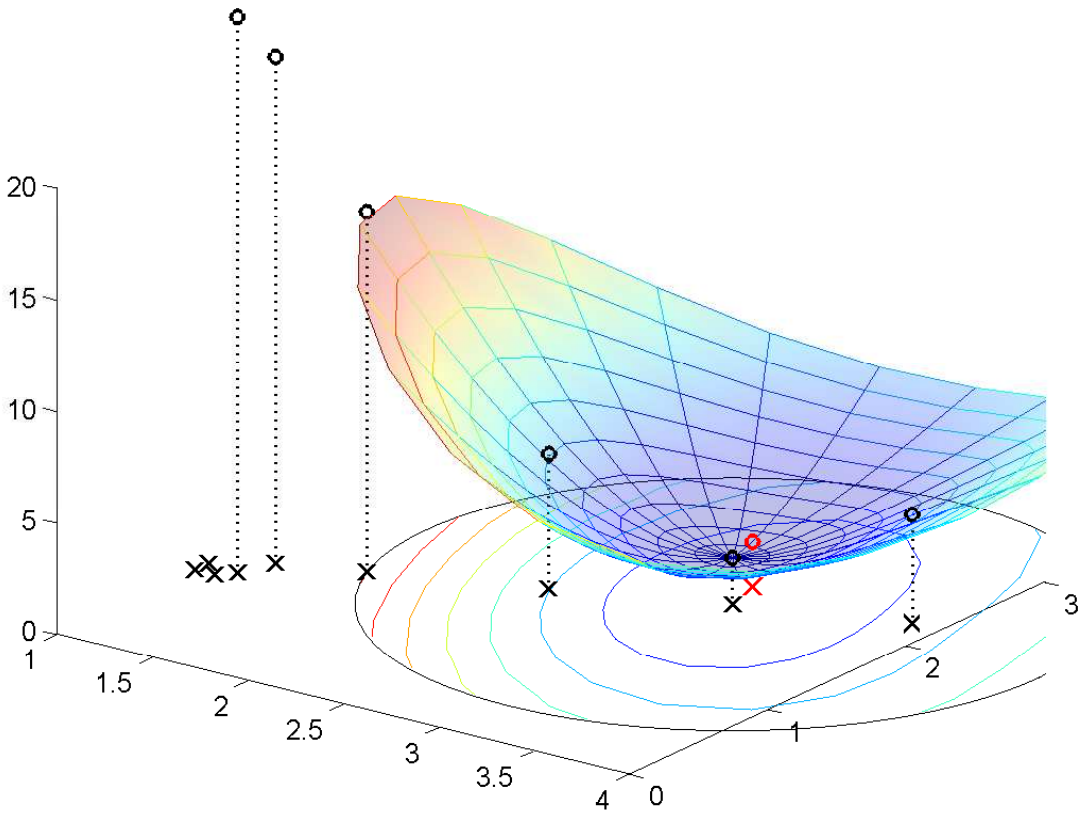
Interpolation-based Trust-region Methods



Iteration k:

- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was

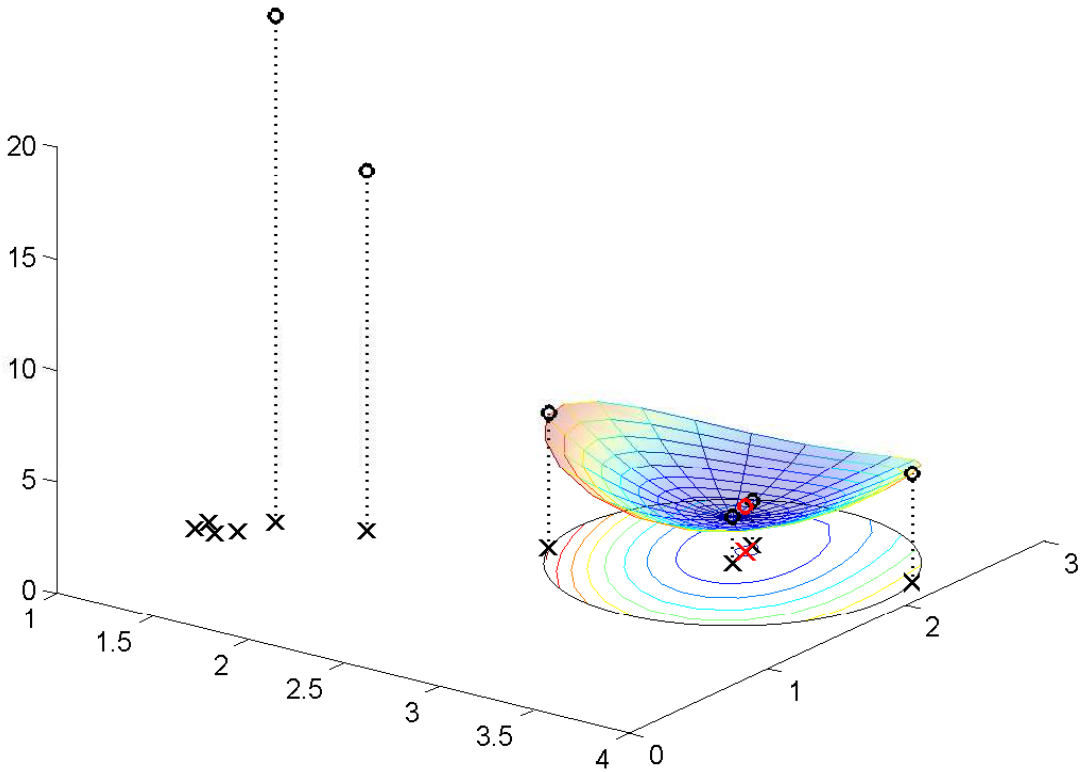
Interpolation-based Trust-region Methods



Iteration k:

- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was

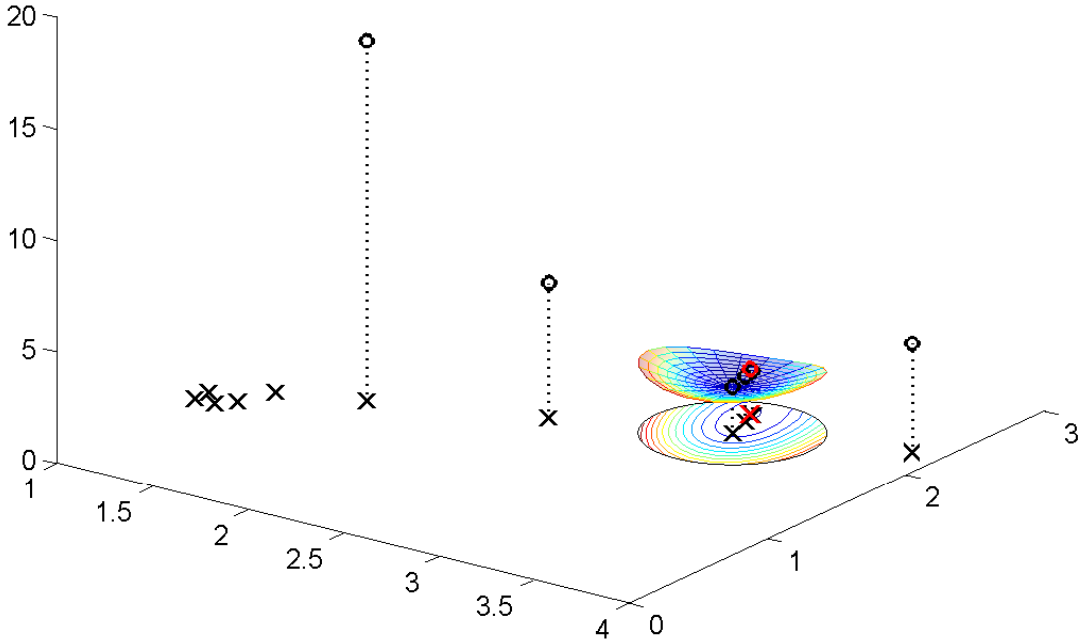
Interpolation-based Trust-region Methods



Iteration k:

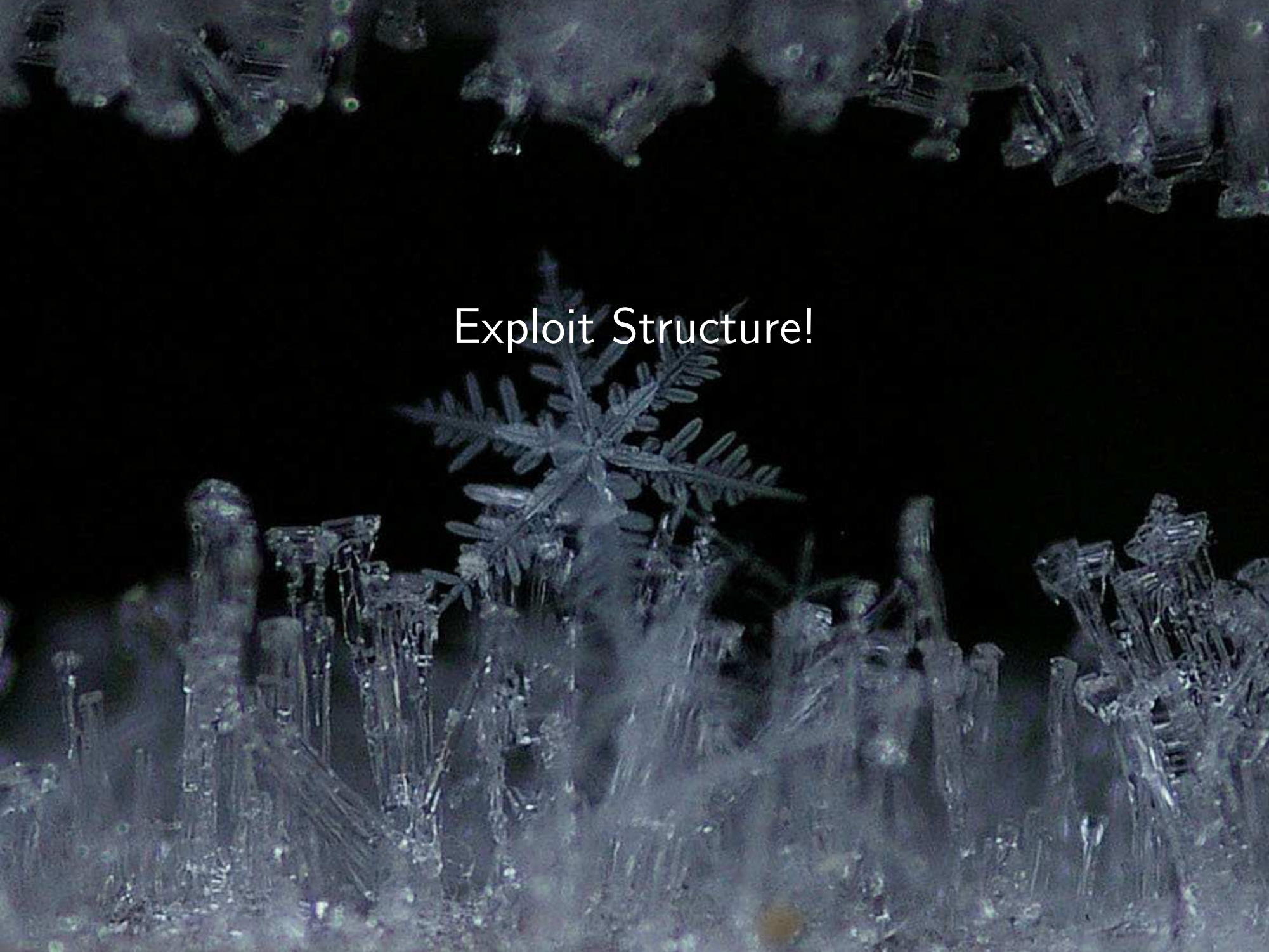
- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was

Interpolation-based Trust-region Methods

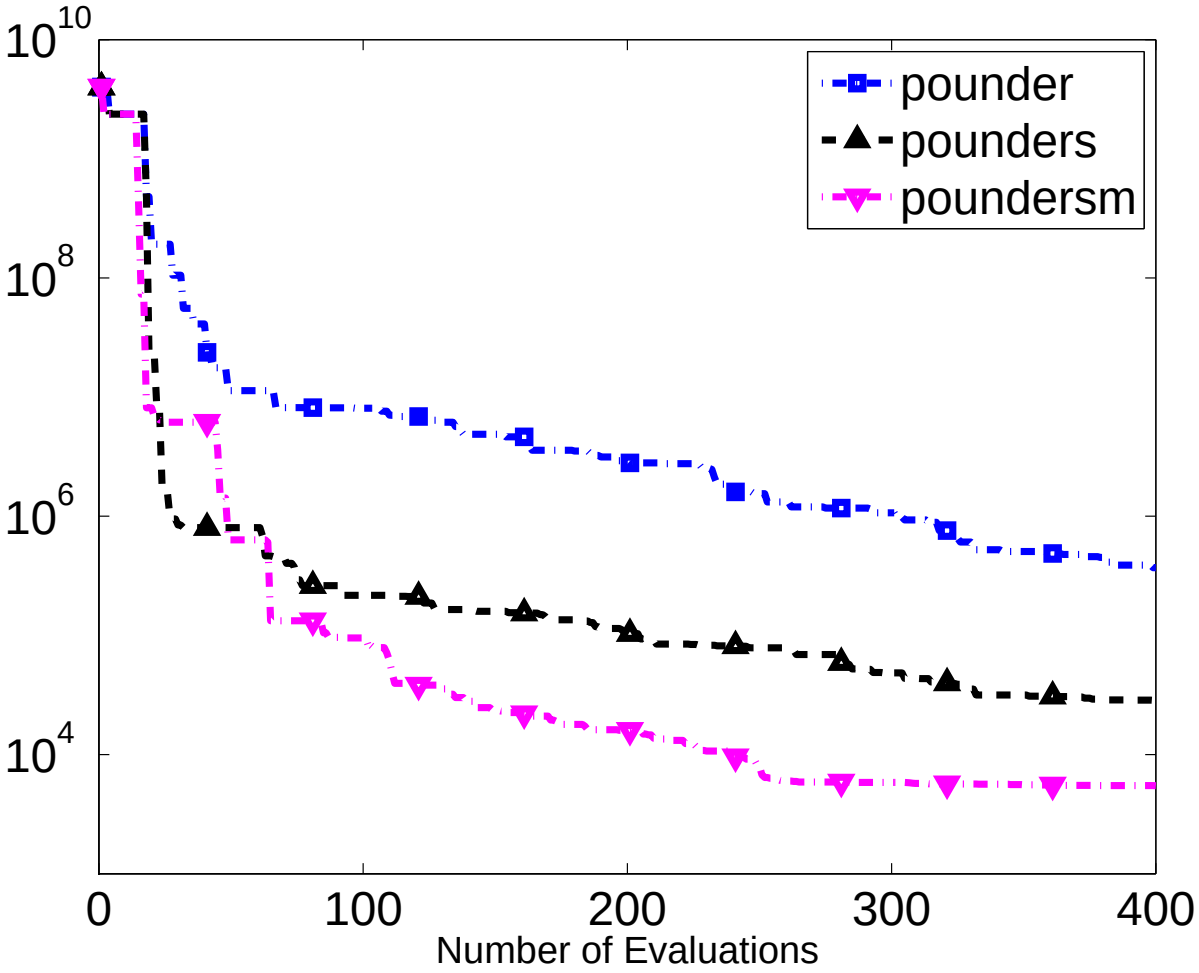


Iteration k:

- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was


Exploit Structure!

Performance of Model-Based Methods



Optimizing EDF in [Bertolli et al., PRC 2012]

Calibration is not a Blackbox Problem

Generic:

$$\min_{\mathbf{x}} \{ f(\mathbf{x}) : \mathbf{x} \in \Omega \subseteq \mathbb{R}^{n_x} \}$$

- $\mathbf{x}$ n_x decision variables
- f $: \mathbb{R}^{n_x} \to \mathbb{R}$ objective function
- Ω feasible region, $\{\mathbf{x} : c_E(\mathbf{x}) = 0, c_I(\mathbf{x}) \leq 0\}$
  - c_E (vector of) equality constraints
  - c_I (vector of) inequality constraints

Calibration is not a Blackbox Problem

Generic:

$$\min_{\mathbf{x}} \{ f(\mathbf{x}) : \mathbf{x} \in \Omega \subseteq \mathbb{R}^{n_x} \}$$

- $\mathbf{x}$ n_x decision variables
- f $: \mathbb{R}^{n_x} \to \mathbb{R}$ objective function
- Ω feasible region, $\{\mathbf{x} : c_E(\mathbf{x}) = 0, c_I(\mathbf{x}) \leq 0\}$
  - c_E (vector of) equality constraints
  - c_I (vector of) inequality constraints

Typical calibration problem:

$$f(\mathbf{x}) = \|\mathbf{F}(\mathbf{x})\|_2^2 = \sum_{i=1}^{n_d} F_i(\mathbf{x})^2$$

- $\mathbf{x}$ n_x coupling constants
- F_i $: \mathbb{R}^{n_x} \to \mathbb{R}$ residual function
  - Ex.- $\frac{1}{w_i}\left(S(\mathbf{x}; \boldsymbol{\nu}_i) - d_i\right)$
    - $S(\mathbf{x}; \boldsymbol{\nu}_i)$: numerical simulation
  - Ex.- Obtain $\chi^2(\mathbf{x})$ by $\frac{1}{n_d - n_x} f(\mathbf{x})$
- Ω $= \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$
  - Finite bounds (for some x_i)
  - Often dictated by dom($\mathbf{S}$)

[Ekström et al, PRL 2013] [Kortelainen et al, PRC 2014]

Calibration is not a Blackbox Problem

Generic:

$$\min_{\mathbf{x}} \{ f(\mathbf{x}) : \mathbf{x} \in \Omega \subseteq \mathbb{R}^{n_x} \}$$

- $\mathbf{x}$ n_x decision variables
- f $: \mathbb{R}^{n_x} \to \mathbb{R}$ objective function
- Ω feasible region, $\{\mathbf{x} : c_E(\mathbf{x}) = 0, c_I(\mathbf{x}) \leq 0\}$
  - c_E (vector of) equality constraints
  - c_I (vector of) inequality constraints

Typical calibration problem:

$$f(\mathbf{x}) = \|\mathbf{F}(\mathbf{x})\|_2^2 = \sum_{i=1}^{n_d} F_i(\mathbf{x})^2$$

- $\mathbf{x}$ n_x coupling constants
- F_i $: \mathbb{R}^{n_x} \to \mathbb{R}$ residual function
  - Ex.- $\frac{1}{w_i}\left(S(\mathbf{x}; \boldsymbol{\nu}_i) - d_i\right)$
    - $S(\mathbf{x}; \boldsymbol{\nu}_i)$: numerical simulation
  - Ex.- Obtain $\chi^2(\mathbf{x})$ by $\frac{1}{n_d - n_x} f(\mathbf{x})$
- Ω $= \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$
  - Finite bounds (for some x_i)
  - Often dictated by dom($\mathbf{S}$)

[Ekström et al, PRL 2013] [Kortelainen et al, PRC 2014]

- Taking advantage of structure should further reduce # of expensive evaluations

Exploiting Nonlinear Least Squares Structure

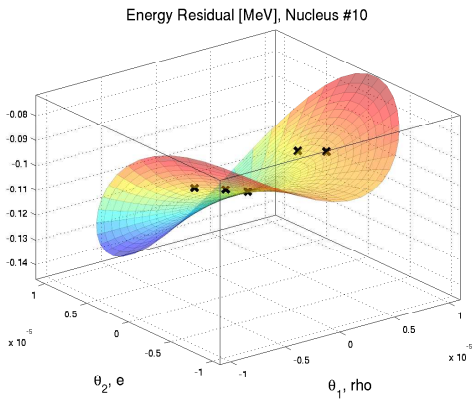
Obtain a vector of output $F_1(x), \ldots, F_p(x)$

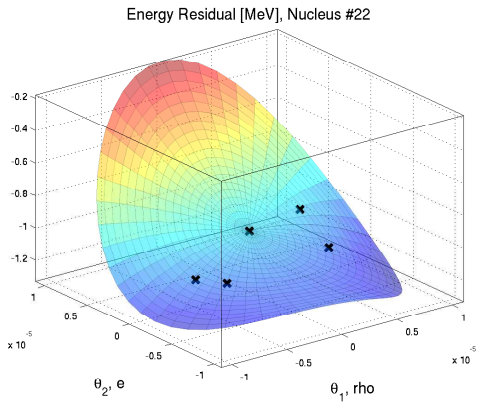
- Model each F_i

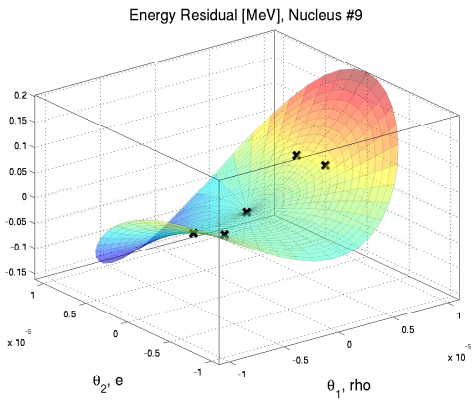
$$F_i(x) \approx q_k^{(i)}(x) = F_i(x_k) + (x - x_k)^\top g_k^{(i)} + \frac{1}{2}(x - x_k)^\top H_k^{(i)}(x - x_k)$$

- Employ models in the approximation

$$\begin{aligned} \nabla f(x) &= \textstyle\sum_i \nabla \mathbf{F_i}(\mathbf{x}) F_i(x) && \rightarrow \textstyle\sum_i \mathbf{g_k^{(i)}}(\mathbf{x}) F_i(x) \\ \nabla^2 f(x) &= \textstyle\sum_i \nabla \mathbf{F_i}(\mathbf{x}) \nabla \mathbf{F_i}(\mathbf{x})^T + F_i(x) \nabla^2 \mathbf{F_i}(\mathbf{x}) && \rightarrow \textstyle\sum_i \mathbf{g_k^{(i)}}(\mathbf{x}) \mathbf{g_k^{(i)}}(\mathbf{x})^T + F_i(x) \mathbf{H_k^{(i)}}(\mathbf{x}) \end{aligned}$$







The POUNDERS Method & Open-Source Software

Practical Optimization Using No DERivatives for sums of Squares

- a local, model-based, full Newton-like, trust-region algorithm
- for unconstrained and bound-constrained
- nonlinear-least squares problems
- in the absence of some derivatives (derivative-free)

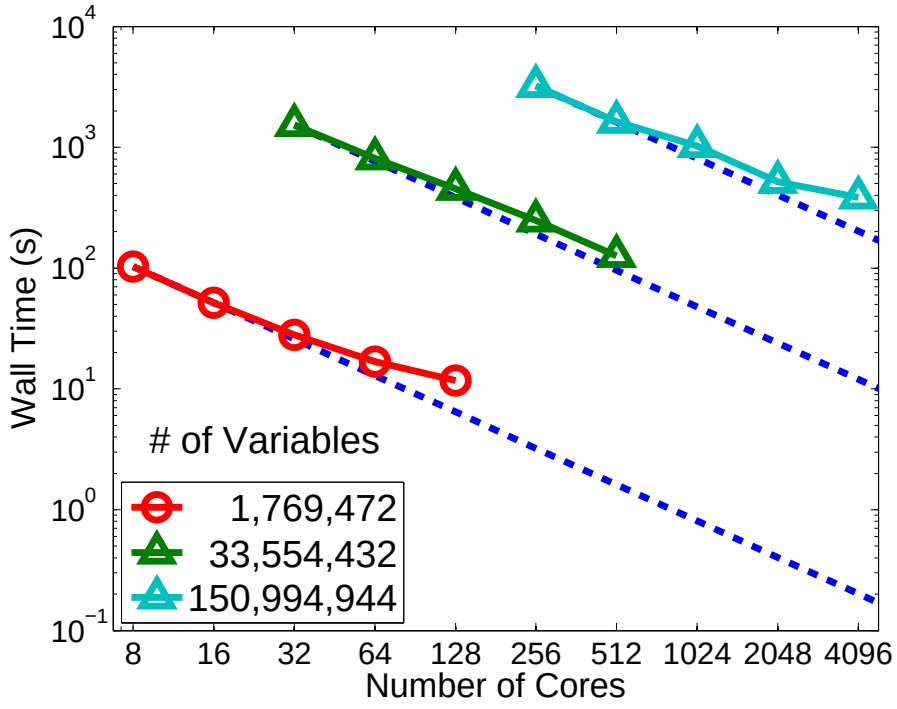
that

- is a misnomer (uses some derivatives)
- is robust to noise/poor local minima
- has a simple interface (provide routine for $\mathbf{S}$)
- allows for parallel evaluation of $\mathbf{S}$
- has asymptotic convergence guarantees
- performs well in practice
- is available in TAO [http://mcs.anl.gov/tao]

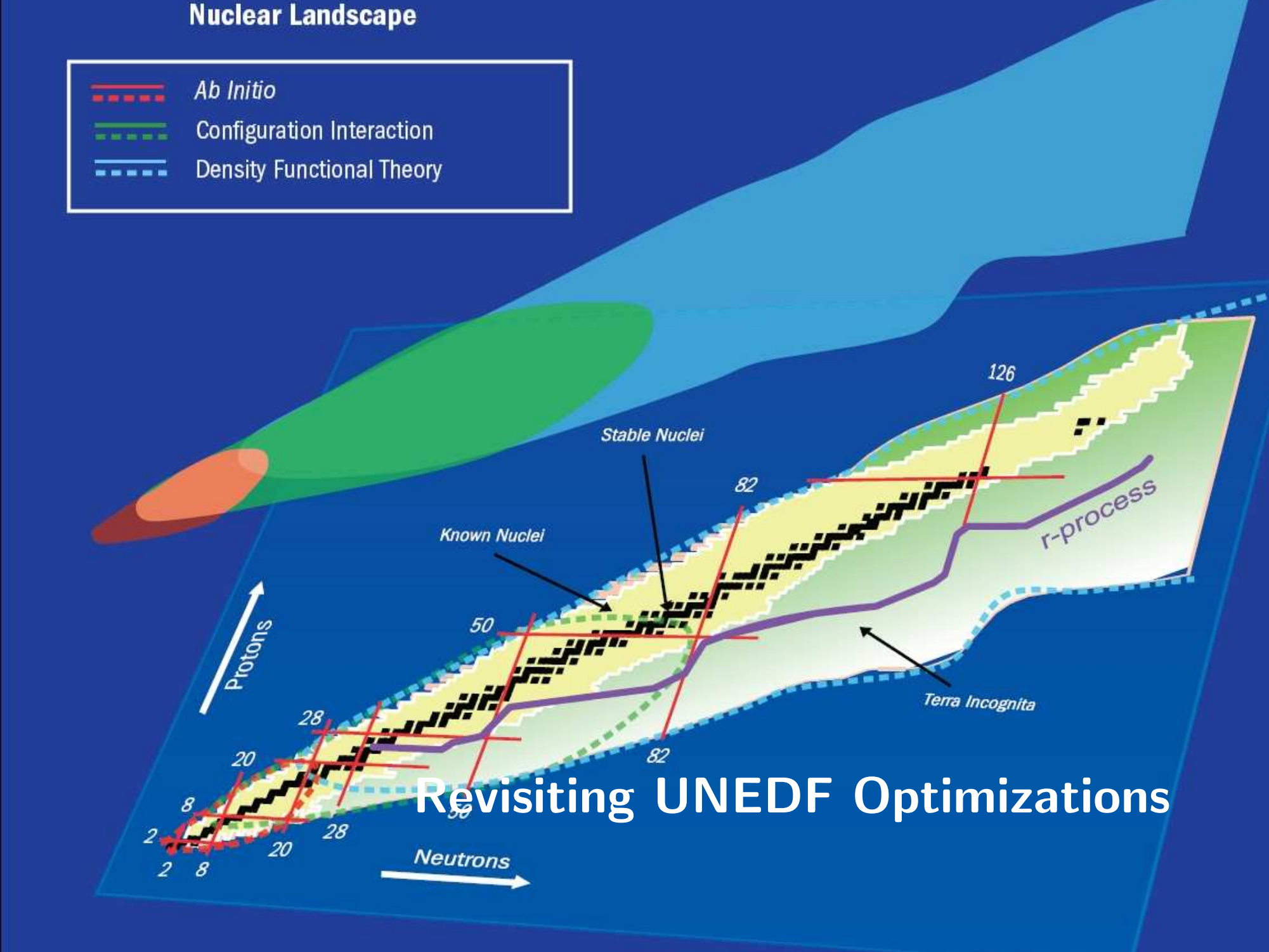
What is TAO?

Toolkit for Advanced Optimization

- Open-source library for large-scale optimization problems
- Emphasize portability, performance, scalable parallelism, and architecture-independent interface
- Bi-directional connection to lower level linear algebra
- Works with PETSc (3.3)
- Current version 2.1
- `www.mcs.anl.gov/tao`



Strong scaling of TAO's PDE-constrained optimization solver on an elliptic problem



Revisiting UNEDF Optimizations

Validation of the UNEDF DFT Optimizations

Changes to evaluation engine $\mathbf{S}(\mathbf{x})$

- ◇ HFBTHO Ver 201 [Stoitsov, Schunck, et al., CPC 2013]

2010 MPI-only
single core per nucleus, 72 nuclei evaluated concurrently

2014 Hybrid MPI+OpenMP
98 nuclei evaluated concurrently using 784 cores

Validation of the UNEDF DFT Optimizations

Changes to evaluation engine $\mathbf{S}(\mathbf{x})$

- HFBTHO Ver 201 [Stoitsov, Schunck, et al., CPC 2013]

2010 MPI-only
single core per nucleus, 72 nuclei evaluated concurrently

2014 Hybrid MPI+OpenMP
98 nuclei evaluated concurrently using 784 cores

Look back over previous results

- Local optimization
- Computational noise – changes in the code
- Sensitivity analysis – error bars, σ_i

UNEDF0, revisited

	Starting from SLy4		Starting from UNEDF0		Scaled Diff.
	$\hat{\mathbf{x}}^{1,\text{initial}}$	$\hat{\mathbf{x}}^{1,\text{final}}$	$\hat{\mathbf{x}}^{2,\text{initial}}$	$\hat{\mathbf{x}}^{2,\text{final}}$	
ρ_c	0.159539	**0.160**486	0.160526	**0.160**482	0.005
E^{NM}/A	-15.9721	-**16.06**85	-16.0559	-**16.05**81	-0.189
K^{NM}	229.901	**<u>230</u>**	<u>230</u>	**<u>230</u>**	-
a_{sym}^{NM}	32.0043	**31**.3393	30.5429	**30**.6643	0.221
L_{sym}^{NM}	45.9618	54.2493	45.0804	**4**6.2421	0.200
$1/M_s^*$	1.43955	**<u>0.9</u>**	<u>0.9</u>	**<u>0.9</u>**	-
$C_0^{\rho\Delta\rho}$	-76.9962	-**55**.2344	-55.2607	-**55.3**410	0.063
$C_1^{\rho\Delta\rho}$	15.6571	-64.1619	-55.6226	-**55**.4700	-0.153
V_0^n	-258.2	-**170**.796	-170.374	-**170**.575	-0.105
V_0^p	-258.2	-**19**7.782	-199.202	-**198**.759	0.292
$C_0^{\rho\nabla J}$	-92.25	-**7**7.9436	-79.5308	-**79**.4576	0.442
$C_1^{\rho\nabla J}$	-30.75	27.4519	45.6302	**45**.3042	-0.606
$f(\hat{\mathbf{x}})$	1188.75	**67**.9034	67.9854 67.310	**67**.8692	

- Scaled differences: $\frac{\hat{x}_i^{1,\text{final}} - \hat{x}_i^{2,\text{final}}}{\sigma_{\mathbf{i}}}$
- XXX: Reported in UNEDF0 [Kortelainen et al., PRC 2010]
- **XXX**: Agreement with UNEDF0
- <u>XXX</u>: Active bound constraints

Many-Start Results for UNEDF Optimizations

UNEDF0, UNEDF1 [Kortelainen et al., PRC 2012], UNEDF2 [Kortelainen et al., PRC 2014]:

POUNDERS obtains consistent solutions (relative to the original reported uncertainties)

Explanations (other than "very lucky!") include

- POUNDERS is robust: avoids poor local minimizers
- starting points conducive to staying in same basin of attraction
- the data $\mathbf{d}$ and simulation S result in an objective function that is not very multimodal in this part of the parameter space

→ Likely some combination of the above

Contributes further confidence in process

- optimization
- Jacobian-based error bars/SA

Jacobian

Jacobian matrix $J(\mathbf{x}) \in \mathbb{R}^{n_d \times n_x}$

- $[J(\mathbf{x})]_{i,j} = \frac{\partial F_i(\mathbf{x})}{\partial x_j} = \frac{1}{w_i} \frac{\partial S(\mathbf{x};\boldsymbol{\nu}_i)}{\partial x_j}$
- Obtain through algorithmic differentiation (AD) [autodiff.org]
- Approximate by numerical differentiation

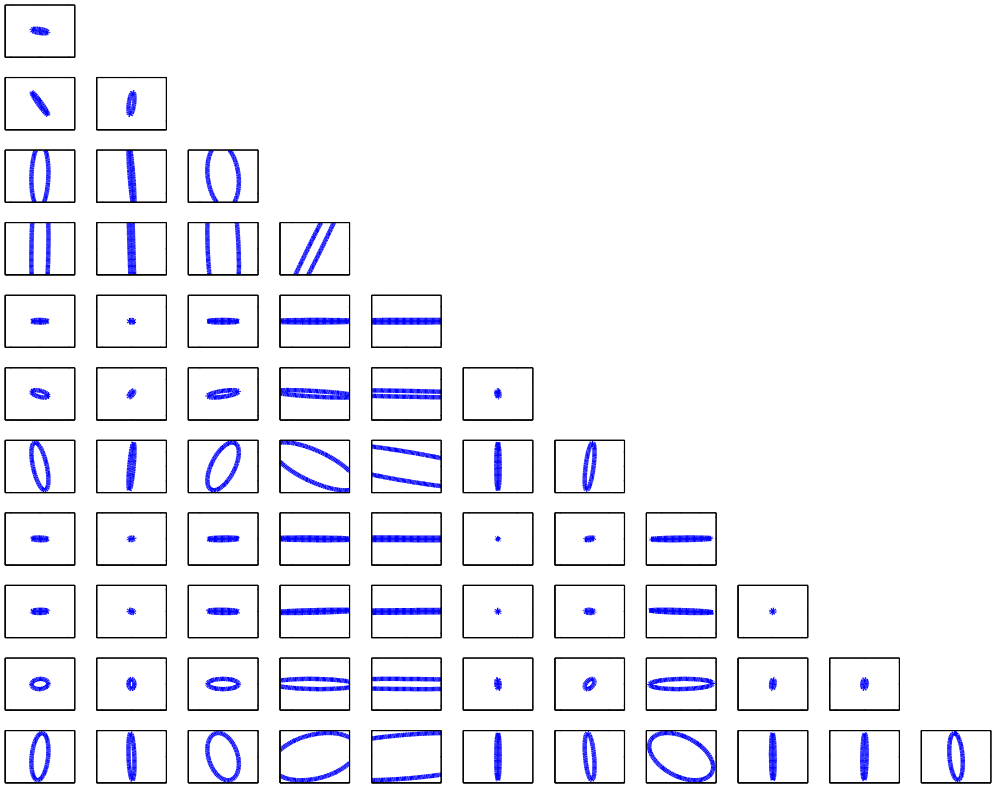
$$\frac{1}{2 w_i h^*_{i,j}} \left(S(\mathbf{x} + h^*_{i,j}\mathbf{e}_j; \boldsymbol{\nu}_i) - S(\mathbf{x} - h^*_{i,j}\mathbf{e}_j; \boldsymbol{\nu}_i) \right)$$

  $h^*_{i,j}$: optimal step size [Moré & W., TOMS 2011] based on the noise in $S(\mathbf{x};\boldsymbol{\nu}_i)$ along the direction $\mathbf{e}_j$
- Requires $2 n_d n_x$ (=2*130*14 for UNEDF2) concurrent evaluations

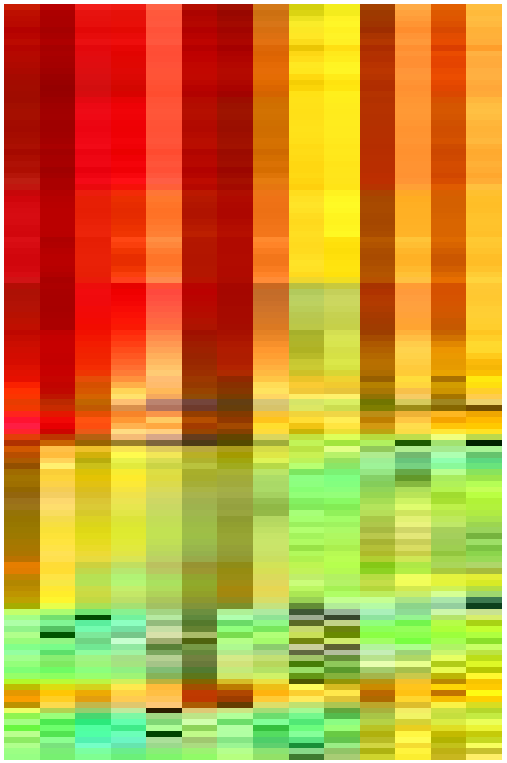
Jacobian-based Sensitivity Analysis

$$\nabla^2_{x,x} f \approx J^T J$$

$$\mathsf{Cov}(x_*) \propto \left(\nabla^2_{x,x} f(x_*)\right)^{-1}$$



Pairwise 95% confidence regions for UNEDF0



Jacobian structure for UNEDF2

Applications Using the Jacobian $\hat{J} = J(\hat{\mathbf{x}})$

Residual $\mathbf{F}(\mathbf{x}) \in \mathbb{R}^{n_d}$ undergoes a change by $\boldsymbol{\epsilon} \in \mathbb{R}^{n_d}$

- Ex.- normalized datum $\frac{d_i}{w_i}$ is changed to $\frac{d_i}{w_i} + \epsilon_i$

$$\hat{\mathbf{x}} \in \arg\min_{\hat{\mathbf{x}} \in \mathbb{R}^{n_x}} f^0(\mathbf{x}) = \|\mathbf{F}(\mathbf{x})\|_2^2 \qquad \hat{\mathbf{x}}_\epsilon \in \arg\min_{\hat{\mathbf{x}} \in \mathbb{R}^{n_x}} f(\mathbf{x}) = \|\mathbf{F}(\mathbf{x}) + \boldsymbol{\epsilon}\|_2^2$$

A second-order expansion of $f = \|\mathbf{F}(\mathbf{x}) + \boldsymbol{\epsilon}\|_2^2$ about $\hat{\mathbf{x}}$:

$$f(\hat{\mathbf{x}}) + 2\boldsymbol{\epsilon}^T \hat{J}(\mathbf{x} - \hat{\mathbf{x}}) + \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^T \left(\nabla^2 f^0(\hat{\mathbf{x}}) + 2\sum_{i=1}^{n_d} \epsilon_i \nabla^2 F_i(\hat{\mathbf{x}}) \right) (\mathbf{x} - \hat{\mathbf{x}}),$$

When $\boldsymbol{\epsilon}$ is small, this quadratic will be convex and hence minimized at

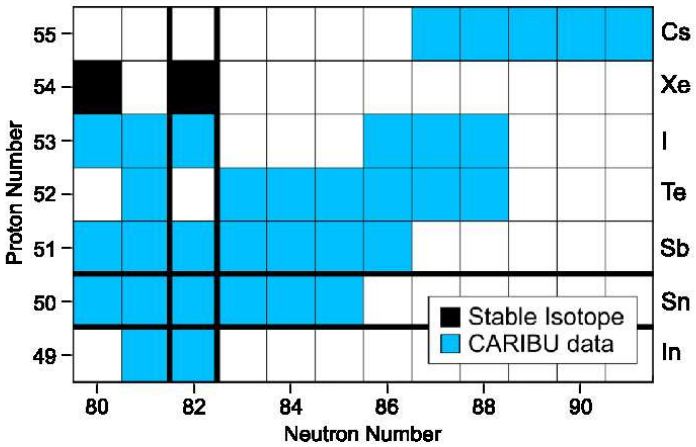
$$\mathbf{x}_\epsilon - \hat{\mathbf{x}} = 2\left(\nabla^2 f^0(\hat{\mathbf{x}})\right)^{-1} \hat{J}^T \boldsymbol{\epsilon} + \mathcal{O}(\|\boldsymbol{\epsilon}\|^2).$$

When $\mathbf{F}(\hat{\mathbf{x}})$ is small, $\nabla^2 f^0(\hat{\mathbf{x}}) \approx 2\hat{J}^T \hat{J}$ and

$$\tilde{\mathbf{x}}_\epsilon \approx \hat{\mathbf{x}} + \left(\hat{J}^T \hat{J}\right)^{-1} \hat{J}^T \boldsymbol{\epsilon}$$

Changes to the Data: Argonne Masses

New nuclear mass data from CARIBU [van Schelt et al., PRL 2014] for 17 neutron-rich, even-even nuclei



UNEDF1 [Kortelainen et al., PRC 2012]:
$n_d = 115$ data

- 75 Masses/binding energies
- 28 RMS radii
  - 8 Pairing gaps/Odd-even mass differences
  - 4 Excitation energies of fission isomers

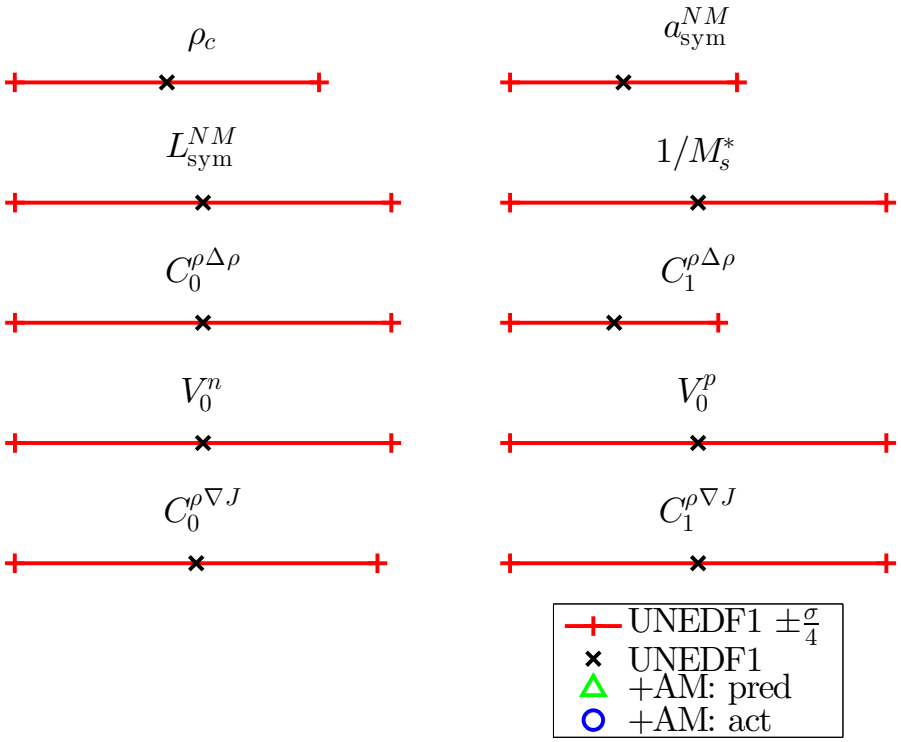
from 75 even-even nuclei

- ◇ 39 deformed (mass only)
- ◇ 28 spherical (mass and rms)
- ◇ 8 deformed (mass and pairing),

11 with $A < 66$, 64 with $A > 106$

Changes to the Data: Argonne Masses (II)

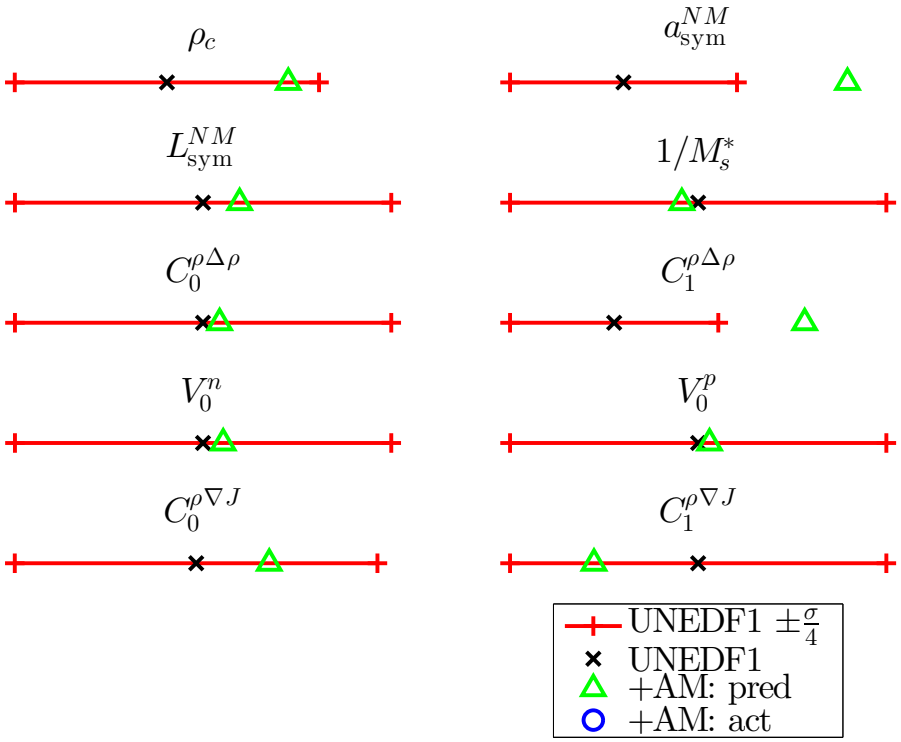
- $\hat{\mathbf{x}}$ UNEDF1 parameterization
- $\mathbf{d}$ UNEDF1 + AM predictions
- $\boldsymbol{\epsilon}$ Zeros except for new mass data
- σ Uncertainties based on $\hat{J}$



Changes to the Data: Argonne Masses (II)

- $\hat{\mathbf{x}}$ UNEDF1 parameterization
- $\mathbf{d}$ UNEDF1 + AM predictions
- ϵ Zeros except for new mass data
- σ Uncertainties based on $\hat{J}$

$\hat{\mathbf{x}}_\epsilon$ predicts that inactive parameters remain within UNEDF1 standard deviations

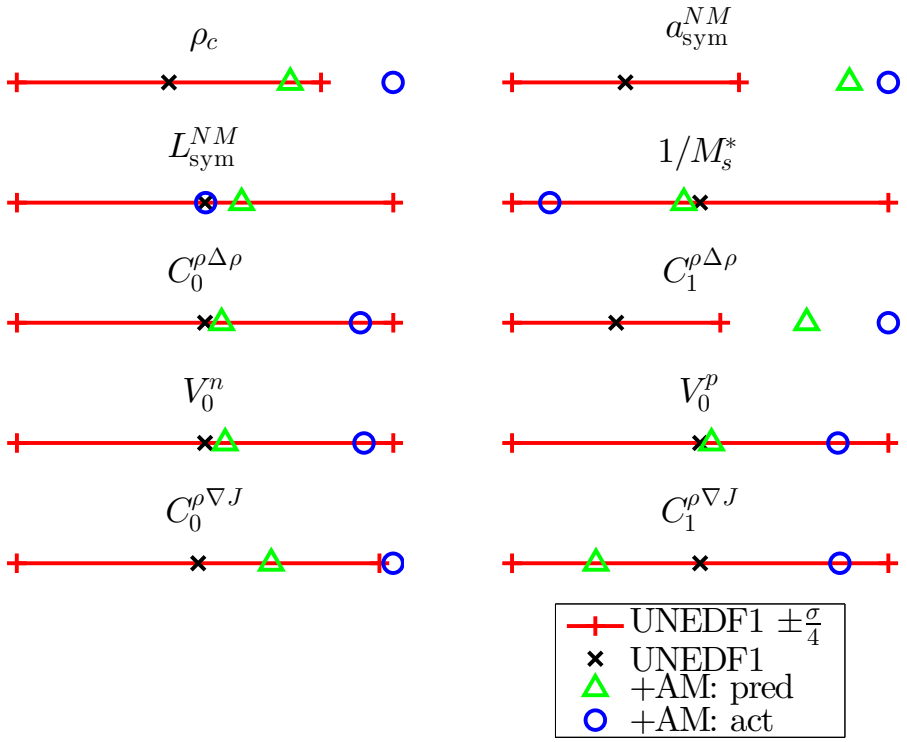


Changes to the Data: Argonne Masses (II)

- $\hat{\mathbf{x}}$ UNEDF1 parameterization
- $\mathbf{d}$ UNEDF1 + AM predictions
- ϵ Zeros except for new mass data
- σ Uncertainties based on $\hat{J}$

$\hat{\mathbf{x}}_\epsilon$ predicts that inactive parameters remain within UNEDF1 standard deviations

- $\mathbf{x}_*$ Actual POUNDERS solution
  - $\chi^2 = \frac{f(\hat{\mathbf{x}})}{n_d - n_x} = \frac{51.058}{103} = 0.496$
  - $\chi^2 = \frac{f(\mathbf{x}_*)}{n_d - n_x} = \frac{54.01}{120} = 0.450$



Summary

- Move beyond "blackbox" optimization
- Exploiting structure yields better solutions, in fewer simulations
- Promote optimization/modeling considerations during code development
- Bridge theoretical calculations to experiments and ab initio approaches to DFT codes through calibration for new observables
- `www.mcs.anl.gov/tao` (Optimization toolkit)
- `www.mcs.anl.gov/~wild` (Get in touch!)

Grateful to Coauthors

M. Bertolli, A. Ekström, C. Forssén, G. Hagen, M. Hjorth-Jensen, D. Higdon, G.R. Jansen, M. Kortelainen, T. Lesinski, R. Machleidt, J. McDonnell, J. Moré, T. Munson, H. Nam, W. Nazarewicz, E. Olsen, T. Papenbrock, A. Pastore, P.-G. Reinhardt, J. Sarich, N. Schunck, M. Stoitsov, J. Vary, K. Wendt, *and others*

Summary

- Move beyond "blackbox" optimization
- Exploiting structure yields better solutions, in fewer simulations
- Promote optimization/modeling considerations during code development
- Bridge theoretical calculations to experiments and ab initio approaches to DFT codes through calibration for new observables
- `www.mcs.anl.gov/tao` (Optimization toolkit)
- `www.mcs.anl.gov/~wild` (Get in touch!)

Grateful to Coauthors

M. Bertolli, A. Ekström, C. Forssén, G. Hagen, M. Hjorth-Jensen, D. Higdon, G.R. Jansen, M. Kortelainen, T. Lesinski, R. Machleidt, J. McDonnell, J. Moré, T. Munson, H. Nam, W. Nazarewicz, E. Olsen, T. Papenbrock, A. Pastore, P.-G. Reinhardt, J. Sarich, N. Schunck, M. Stoitsov, J. Vary, K. Wendt, *and others*

Merci!

.

Optimization Formulation Plays a Crucial Role

- Solution speed and quality
- Comparing results and extrapolation
  - *What is your χ^2 value?*

$\longrightarrow$ What is your χ^2 function?

Optimization Formulation Plays a Crucial Role

- Solution speed and quality
- Comparing results and extrapolation
  - *What is your χ^2 value?*

$\longrightarrow$ What is your χ^2 function?

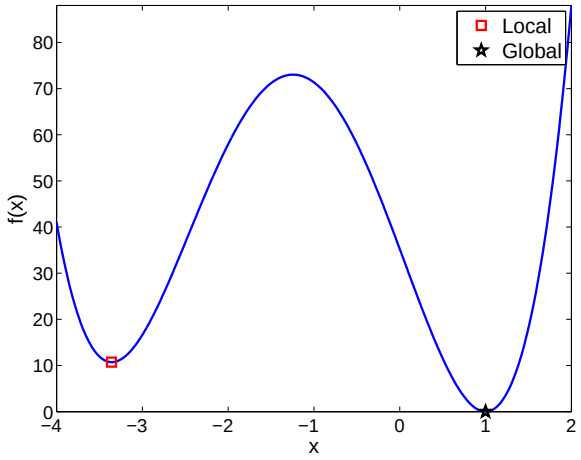
Ex.-: Different norms in fitting

- Now: $f(\mathbf{x}) = \|\mathbf{S}(\mathbf{x}) - \mathbf{d}\|_A^2$, $A =$diag(w_i^{-1})
- Correlations: $f(\mathbf{x}) = \|\mathbf{S}(\mathbf{x}) - \mathbf{d}\|_A^2 = \sum_i \sum_j A_{ij} \left(S_i(\mathbf{x}) - d_i\right)\left(S_j(\mathbf{x}) - d_j\right)$, $A \succ 0$
- $f(\mathbf{x}) = \sum_i \left(\frac{S_i(\mathbf{x}) - d_i}{w_i(\mathbf{x})}\right)^2 + \lambda \|\mathbf{x} - \mathbf{x}^{\mathrm{nom}}\|_0$
- Other regularization terms (total variation, Tikhonov, ...)

Global Optimization, $\min_{x \in \Omega} f(x)$

Careful:

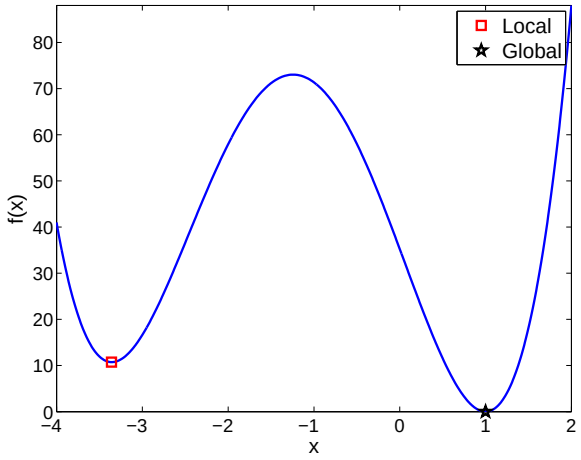
- Global convergence: Convergence (to a local solution/stationary point) from anywhere in Ω
- Convergence to a global minimizer: Obtain x_* with $f(x_*) \leq f(x) \, \forall x \in \Omega$



Global Optimization, $\min_{x \in \Omega} f(x)$

Careful:

- Global convergence: Convergence (to a local solution/stationary point) from anywhere in Ω
- Convergence to a global minimizer: Obtain x_* with $f(x_*) \leq f(x) \, \forall x \in \Omega$



Anyone selling you global solutions when derivatives are unavailable:

either assumes more about your problem (e.g., convex f)

or expects you to wait forever

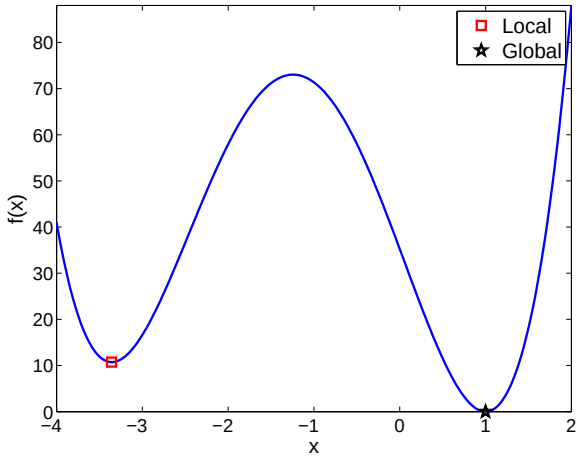
Törn and Žilinskas: An algorithm converges to the global minimum for any continuous f if and only if the sequence of points visited by the algorithm is dense in Ω.

or cannot be trusted

Global Optimization, $\min_{x \in \Omega} f(x)$

Careful:

- Global convergence: Convergence (to a local solution/stationary point) from anywhere in Ω
- Convergence to a global minimizer: Obtain x_* with $f(x_*) \leq f(x) \, \forall x \in \Omega$



Anyone selling you global solutions when derivatives are unavailable:

either assumes more about your problem (e.g., convex f)

or expects you to wait forever

Törn and Žilinskas: An algorithm converges to the global minimum for any continuous f if and only if the sequence of points visited by the algorithm is dense in Ω.

or cannot be trusted

Instead:

- Rapidly find good local solutions and/or be robust to poor solutions
- Consider multistart approaches and/or structure of multimodality

Heuristics: Experimental X-ray Beam Design