Optimization Problems in Nuclear Theory

Stefan Wild

Mathematics and Computer Science Division
Argonne National Laboratory

Joint work with Jared O'Neal
and many physicist collaborators:
A. Ekström, C. Forssén, G. Hagen, M. Hjorth-Jensen, G.R. Jansen, M. Kortelainen, T. Lesinski, A. Lovell, R. Machleidt, J. McDonnell, H. Nam, N. Michel, W. Nazarewicz, F.M. Nunes, E. Olsen, T. Papenbrock, P.-G. Reinhardt, N. Schunck, M. Stoitsov, J. Vary, K. Wendt, **and others**

October 30, 2018

Acknowledgments and Plan

NUCLEI
Nuclear Computational Low-Energy Initiative

SciDAC
Scientific Discovery through Advanced Computing

ISNET-*

1. Optimization background
   - Local and global
   - Derivatives and no derivatives
2. Typical optimization-based formulations
   - Nonlinear least squares
   - POUNDERS
3. Optimization and supercomputing
4. Optimization under uncertainty

Mathematical/Numerical Nonlinear Optimization

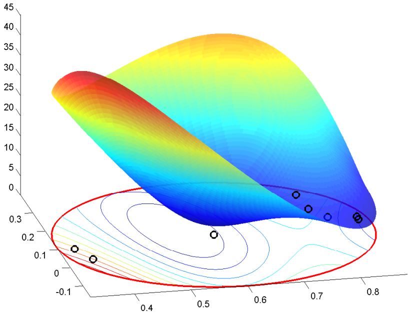
Find parameters $\mathbf{x} = (x_1, \ldots, x_n)$ in domain $\boldsymbol{\Omega}$ to improve objective f

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

- (Unless $\boldsymbol{\Omega}$ is very special) Need to evaluate f at many $\mathbf{x}$ to find a good $\hat{\mathbf{x}}_*$

Here:

- Assume f is deterministic (and smooth except where noted)
- Assume that uncertainty modeled through constraints and objective(s)
- Assumes sensitivity analysis, uncertainty quantification, and validations



(Computationally Expensive) Simulation-Based Optimization

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

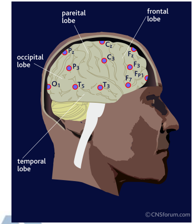
"parameter estimation", "model calibration", "design optimization", . . .

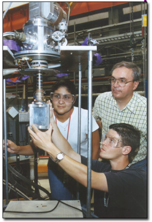
- Evaluating $\mathbf{S}$ means running a simulation modeling some (smooth) process
- Derivatives $\nabla_x S$ often unavailable or prohibitively expensive to obtain
- $\mathbf{S}$ (even when parallelized) takes secs/mins/days

Evaluation is a bottleneck for optimization

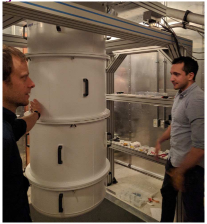
- $\boldsymbol{\Omega}$ compact, known region (e.g., finite bound constraints)

Functions of complex (numerical/physical) simulations arise everywhere

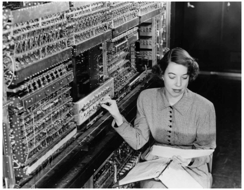








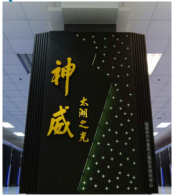
Computing Advances Drive Research in Simulation-Based Optimization



Argonne's AVIDAC
(1953 vacuum tubes)

Argonne's BlueGene/Q
(2012 0.79M cores)

Argonne's Theta
(2017 0.23M cores)



Sunway TaihuLight
(2016 11M cores)

The simulations underlying today's SBO problems were nearly unthinkable a generation ago

Argonne's "A21"
(2021 ??? cores)

Parameter Estimation is NOT a Generic/Blackbox Optimization Problem

Generic:

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

- $\mathbf{x}$ n decision variables
- f $: \mathbb{R}^n \to \mathbb{R}$ objective function
- $\boldsymbol{\Omega}$ feasible region, $\{\mathbf{x} : \mathbf{c}_E(\mathbf{x}) = 0, \mathbf{c}_I(\mathbf{x}) \leq 0\}$
  - $\mathbf{c}_E$ (vector of) equality constraints
  - $\mathbf{c}_I$ (vector of) inequality constraints

Parameter Estimation is NOT a Generic/Blackbox Optimization Problem

Generic:

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

- $\mathbf{x}$ n decision variables
- f $: \mathbb{R}^n \to \mathbb{R}$ objective function
- $\mathbf{\Omega}$ feasible region, $\{\mathbf{x} : \mathbf{c}_E(\mathbf{x}) = 0, \mathbf{c}_I(\mathbf{x}) \leq 0\}$
  - $\mathbf{c}_E$ (vector of) equality constraints
  - $\mathbf{c}_I$ (vector of) inequality constraints

Typical calibration problem:

$$f(\mathbf{x}) = \|\mathbf{R}(\mathbf{x})\|_2^2 = \sum_{i=1}^p R_i(\mathbf{x})^2$$

- $\mathbf{x}$ n coupling constants
- R_i $: \mathbb{R}^n \to \mathbb{R}$ residual function
  - Ex.- $\frac{1}{w_i}(S(\mathbf{x};\boldsymbol{\theta}_i) - d_i)$
    - $S(\mathbf{x};\boldsymbol{\theta}_i)$: numerical simulation
  - Ex.- Obtain $\chi^2(\mathbf{x})$ by $\frac{1}{p-n} f(\mathbf{x})$
- $\mathbf{\Omega}$ $= \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$
  - Finite bounds (for some x_i)
  - Often dictated by dom$(\mathbf{S})$

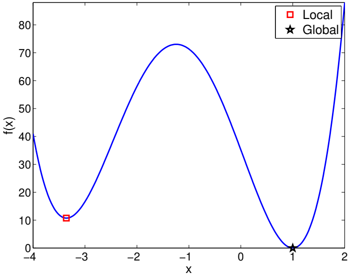
[Ekström et al, PRL 2013] [Kortelainen et al, PRC 2014]

Taking advantage of structure should reduce expense/improve accuracy

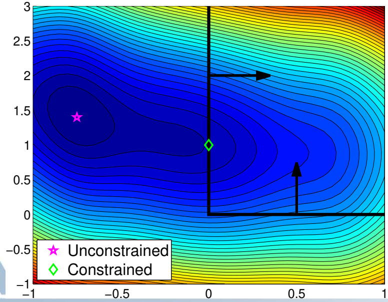
Careful: Local and Global Solutions

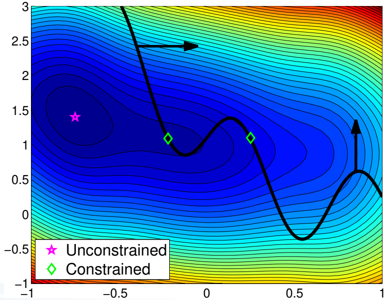
- **Local minimizer** $\hat{\mathbf{x}}_*$:

$$f(\hat{\mathbf{x}}_*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{N}(\hat{\mathbf{x}}_*) \cap \mathbf{\Omega}$$



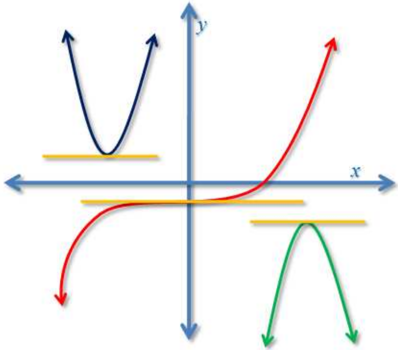
- **Global convergence**: Convergence (to a local solution/stationary point) from anywhere in $\mathbf{\Omega}$
- **Convergence to a global minimizer**: Obtain $\mathbf{x}_*$ with $f(\mathbf{x}_*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbf{\Omega}$





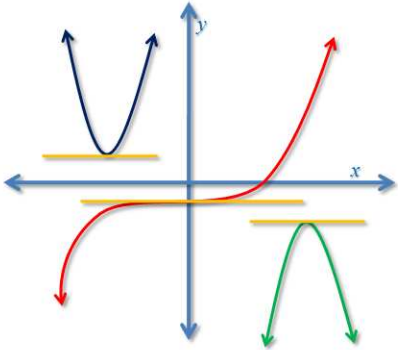
Optimization Tightly Coupled With Derivatives (WRT Parameters)

Typically necessary for optimality: $\nabla_{\mathbf{x}} f(\mathbf{x}_*) + \lambda^T \nabla_{\mathbf{x}} \mathbf{c}_E(\mathbf{x}_*) = 0, \mathbf{c}_E(\mathbf{x}_*) = 0$



Optimization Tightly Coupled With Derivatives (WRT Parameters)

Typically necessary for optimality: $\nabla_{\mathbf{x}} f(\mathbf{x}_*) + \lambda^T \nabla_{\mathbf{x}} \mathbf{c}_E(\mathbf{x}_*) = 0, \mathbf{c}_E(\mathbf{x}_*) = 0$



Algorithmic/Automatic Differentiation (AD)

"Exact* derivatives!"

? No black boxes allowed

? Not always automatic/"cheap"

Finite Differences (FD)

"Nonintrusive", "Numerical Differentiation"

? Expense grows with n

? Sensitive to stepsize choice/noise
→[Moré & W.; SISC 2011], [Moré & W.; TOMS 2012]

But some derivatives are not always available/do not always exist

Typical Optimization-Based Formulations

Standard "χ^2"-based objective

$$f(\mathbf{x}) = \frac{1}{p-n}\sum_{i=1}^{p} R_i(\mathbf{x})^2 = \frac{1}{p-n}\sum_{i=1}^{p}\left(\frac{S(\mathbf{x};\boldsymbol{\theta}_i) - d_i}{\sigma_i}\right)^2$$

- $\{(\boldsymbol{\theta}_1, d_1), \cdots, (\boldsymbol{\theta}_p, d_p)\}$: the data
- $S(\mathbf{x};\boldsymbol{\theta}_i)$: the ith simulation (modeled/theory) output given parameters $\mathbf{x}$
- $\sigma_1, \ldots, \sigma_p$: the (inverse) weights

Typical Optimization-Based Formulations

Standard "χ^2"-based objective

$$f(\mathbf{x}) = \frac{1}{p-n} \sum_{i=1}^{p} R_i(\mathbf{x})^2 = \frac{1}{p-n} \sum_{i=1}^{p} \left(\frac{S(\mathbf{x};\boldsymbol{\theta}_i) - d_i}{\sigma_i} \right)^2$$

- $\{(\boldsymbol{\theta}_1, d_1), \cdots, (\boldsymbol{\theta}_p, d_p)\}$: the data
- $S(\mathbf{x};\boldsymbol{\theta}_i)$: the ith simulation (modeled/theory) output given parameters $\mathbf{x}$
- $\sigma_1, \ldots, \sigma_p$: the (inverse) weights

NB-

- Multiplying f by positive constant does not affect the solution of $\min_{\mathbf{x}} f(\mathbf{x})$
- $\Rightarrow$ all σ_i could be multiplied by a common constant
- $\Rightarrow$ interpretation of $f(\mathbf{x})$ values comes from something other than the optimization

Relationship to Covariance Matrices

- Errors independent and normally distributed: $\mathbf{d} \sim N(\mu, \boldsymbol{\Sigma})$,

$$d_i = \mu(\boldsymbol{\theta}_i; \mathbf{x}_*) + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \sigma_i^2) \qquad i = 1, \ldots, p$$

  $\boldsymbol{\Sigma}$ is a $p \times p$ diagonal matrix, with ith diagonal entry σ_i^2
- Model, $S(\boldsymbol{\theta}; \mathbf{x})$ with Gaussian errors:

$$[S(\boldsymbol{\theta}_1; \mathbf{x}), \cdots, S(\boldsymbol{\theta}_p; \mathbf{x})]^T \sim N\left(\mu(\cdot; \mathbf{x}), \mathbf{C}\right)$$

- $\mathbf{C}$ a ($p \times p$ symmetric positive definite) covariance matrix accounting for correlation between model outputs (i.e., $\mathrm{Cov}(S(\boldsymbol{\theta}_i; \mathbf{x}), S(\boldsymbol{\theta}_j; \mathbf{x})) = C_{i,j}$)

Relationship to Covariance Matrices

- Errors independent and normally distributed: $\mathbf{d} \sim N(\mu, \mathbf{\Sigma})$,

$$d_i = \mu(\boldsymbol{\theta}_i; \mathbf{x}_*) + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \sigma_i^2) \qquad i = 1, \ldots, p$$

  $\mathbf{\Sigma}$ is a $p \times p$ diagonal matrix, with ith diagonal entry σ_i^2
- Model, $S(\boldsymbol{\theta}; \mathbf{x})$ with Gaussian errors:

$$[S(\boldsymbol{\theta}_1; \mathbf{x}), \cdots, S(\boldsymbol{\theta}_p; \mathbf{x})]^T \sim N\left(\mu(\cdot; \mathbf{x}), \mathbf{C}\right)$$

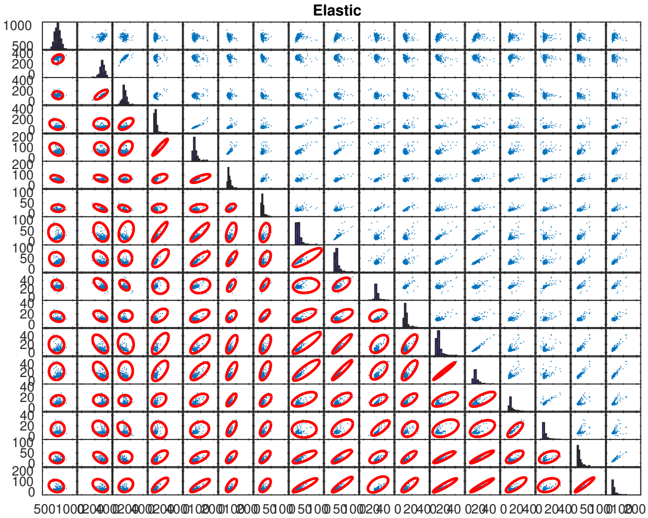
- $\mathbf{C}$ a ($p \times p$ symmetric positive definite) covariance matrix accounting for correlation between model outputs (i.e., Cov$(S(\boldsymbol{\theta}_i; \mathbf{x}), S(\boldsymbol{\theta}_j; \mathbf{x})) = C_{i,j}$)
- Assuming model errors are independent of data errors,

$$[m(\hat{\mathbf{x}}; \boldsymbol{\theta}_1) - d_1, \cdots, m(\hat{\mathbf{x}}; \boldsymbol{\theta}_p) - d_p]^T \sim N(0, \mathbf{C} + \mathbf{\Sigma})$$

- Joint likelihood $l(\mathbf{x}; \boldsymbol{\theta}; \mathbf{d}) \propto \exp\left[-\frac{1}{2}\mathbf{R}(\mathbf{x}; \boldsymbol{\theta})^T \left(\mathbf{C} + \mathbf{\Sigma}\right)^{-1} \mathbf{R}(\mathbf{x}; \boldsymbol{\theta})\right]$

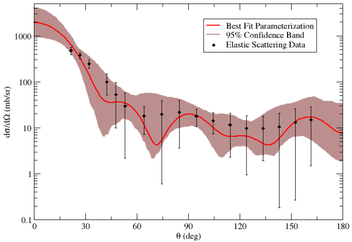
Warning: $\mathbf{C}$,$\mathbf{\Sigma}$ can no longer hide behind constants of proportionality

Incorporating Covariances $\mathrm{Cov}(S(\mathbf{x};\theta_i), S(\mathbf{x};\theta_j))$ in W

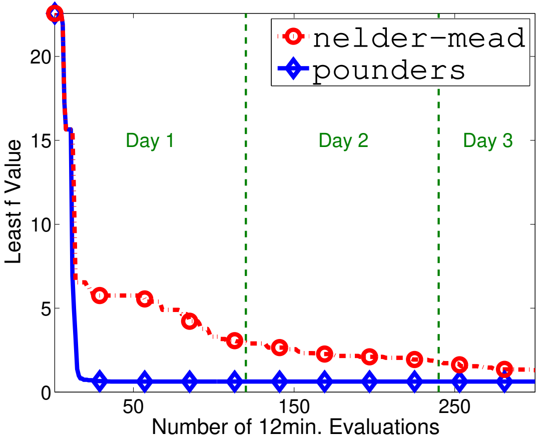


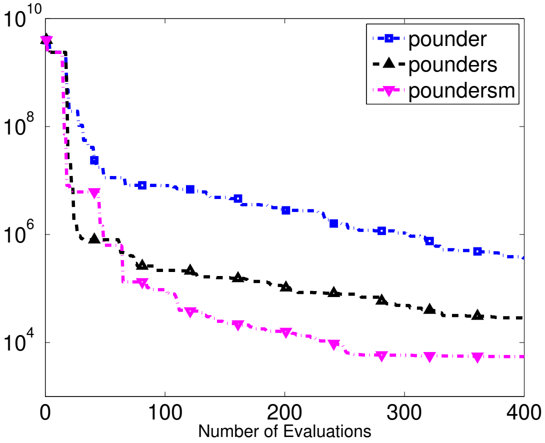
Ex.- optical potentials

[Lovell et al, PRC 2017]



Exploiting Structure Allows One to Solve Difficult Problems





[Kortelainen et al, PRC 2010], [Bertolli et al, PRC 2012], [Kortelainen et al, PRC 2012], [Ekström et al, PRL 2013], [Kortelainen et al, PRC 2014], . . .

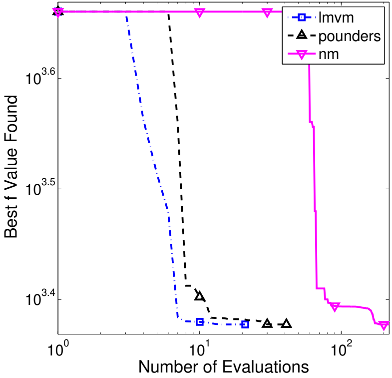
The POUNDERS Method & Open-Source Software

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 PETSc/TAO [http://mcs.anl.gov/tao]



TAO solvers

- **nm** $\nabla_x f$ unavailable, black box
- **pounders** $\nabla_x f$ unavailable, exploits problem structure
- **lmvm** Uses available $\nabla_x f$

Exploiting Nonlinear Least Squares Structure

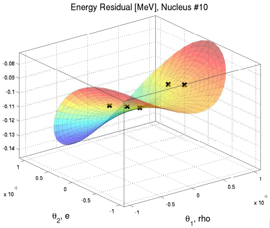
Obtain a *vector* of output $R_1(\mathbf{x}), \ldots, R_p(\mathbf{x})$

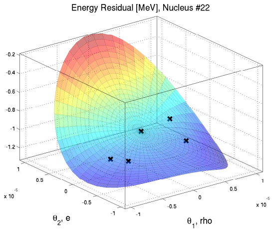
- (Locally) Model each R_i by a surrogate $q_k^{(i)}$

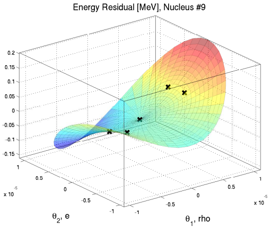
$$R_i(\mathbf{x}) \approx q_k^{(i)}(\mathbf{x}) = R_i(\mathbf{x}_k) + (\mathbf{x}-\mathbf{x}_k)^\top \mathbf{g}_k^{(i)} + \frac{1}{2}(\mathbf{x}-\mathbf{x}_k)^\top \mathbf{H}_k^{(i)}(\mathbf{x}-\mathbf{x}_k)$$

- Employ models in the approximation

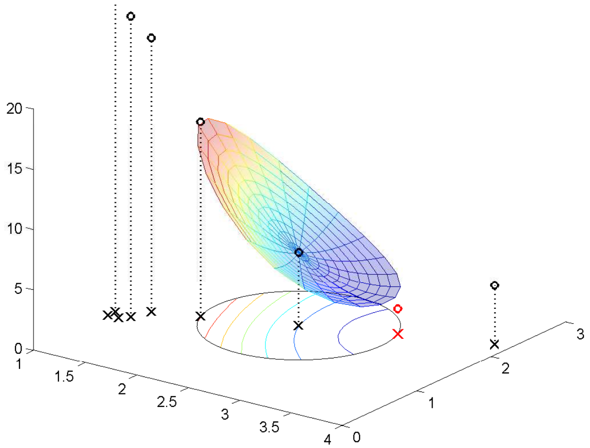
$$\begin{aligned}
\nabla f(\mathbf{x}) &= \textstyle\sum_i \nabla \mathbf{R_i}(\mathbf{x}) R_i(\mathbf{x}) & &\rightarrow \textstyle\sum_i g_k^{(i)}(\mathbf{x}) R_i(\mathbf{x}) \\
\nabla^2 f(\mathbf{x}) &= \textstyle\sum_i \nabla \mathbf{R_i}(\mathbf{x}) \nabla \mathbf{R_i}(\mathbf{x})^T + R_i(\mathbf{x})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







All Together: Model-Based Trust-Region Algorithms

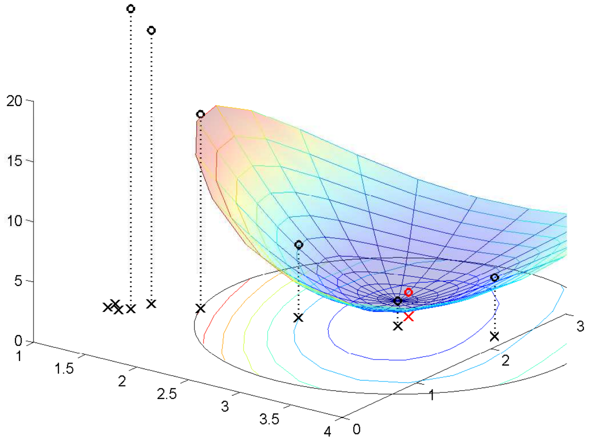


Basic trust region iteration:

- Build surrogate model m (POUNDERS: for each residual R_i)
- Trust approximation of m within region $\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_k\}$
- Use m to obtain next point within $\mathcal{B}$ for evaluation

Incorporate prior knowledge through scaling, norm selection, initial Δ_0, etc.

All Together: Model-Based Trust-Region Algorithms

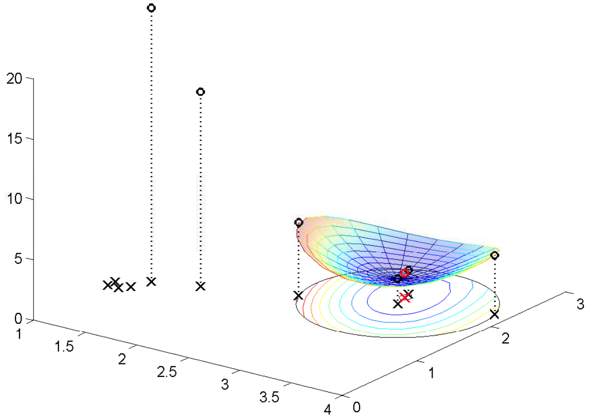


Basic trust region iteration:

- Build surrogate model m (POUNDERS: for each residual R_i)
- Trust approximation of m within region $\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_k\}$
- Use m to obtain next point within $\mathcal{B}$ for evaluation

Incorporate prior knowledge through scaling, norm selection, initial Δ_0, etc.

All Together: Model-Based Trust-Region Algorithms

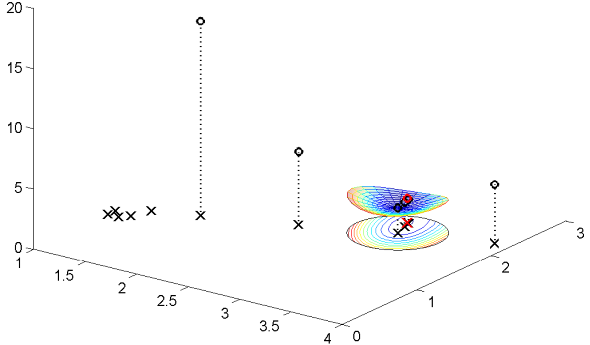


Basic trust region iteration:

- Build surrogate model m (POUNDERS: for each residual R_i)
- Trust approximation of m within region $\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_k\}$
- Use m to obtain next point within $\mathcal{B}$ for evaluation

Incorporate prior knowledge through scaling, norm selection, initial Δ_0, etc.

All Together: Model-Based Trust-Region Algorithms



Basic trust region iteration:

- Build surrogate model m (POUNDERS: for each residual R_i)
- Trust approximation of m within region $\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_k\}$
- Use m to obtain next point within $\mathcal{B}$ for evaluation

Incorporate prior knowledge through scaling, norm selection, initial Δ_0, etc.

Other Deterministic Objective/Loss/Training Function Forms

Standard "χ^2": Assumes independence

$$f(\mathbf{x}) = \frac{1}{p-n}\sum_{i=1}^{p} R_i(\mathbf{x})^2 = \frac{1}{p-n}\sum_{i=1}^{p}\left(\frac{S(\mathbf{x};\theta_i)-d_i}{\sigma_i}\right)^2$$

Correlated: For $\mathbf{W}$ symmetric positive definite:

$$f(\mathbf{x}) = \sum_i \sum_j W_{i,j} R_i(\mathbf{x}) R_j(\mathbf{x}) = \|\mathbf{R}(\mathbf{x})\|_{\mathbf{W}}^2$$

Gaussian priors: $f(\mathbf{x}) = \|\mathbf{R}(\mathbf{x})\|_{\mathbf{W}}^2 + \|\mathbf{x}-\hat{\mathbf{x}}\|_{\mathbf{C}}^2$

(Censored) L1 loss: (LAD)

$$f(\mathbf{x}) = \sum_i w_i \left|d_i - S_i(\mathbf{x})\right| \quad \text{or} \quad f(\mathbf{x}) = \sum_i w_i \left|d_i - \max\left\{S_i(\mathbf{x}), c_i\right\}\right|$$

Solvers exist for many forms of objective; objective form matters!

Nonsmooth Compositions Require Additional Care

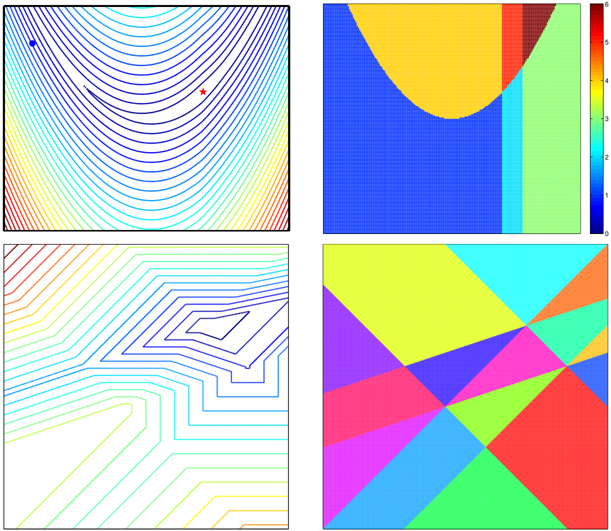
L1 Loss:

$$\sum_{i=1}^{p} |d_i - S_i(\mathbf{x})|$$

Censored L1 loss:

$$\sum_{i=1}^{p} |d_i - \max\{S_i(\mathbf{x}), c_i\}|$$

NB- Can truncate some multimodality

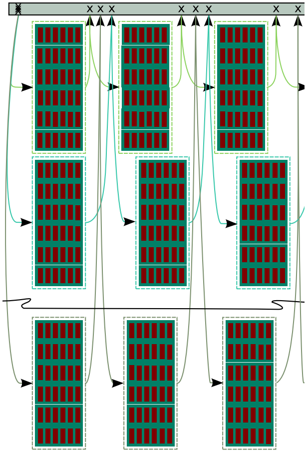


→ Manifold sampling: [Larson, Menickelly, W.; SIOPT 2016], [Khan, Larson, W.; SIOPT 2018]

Exploiting Concurrency is Vital in the Supercomputing Era

Considerations:

- Load balancing
- Variability in run times for a particular nuclei or observable
- Variability in run times across observables
- Degree to which you can predict the run time of an observables

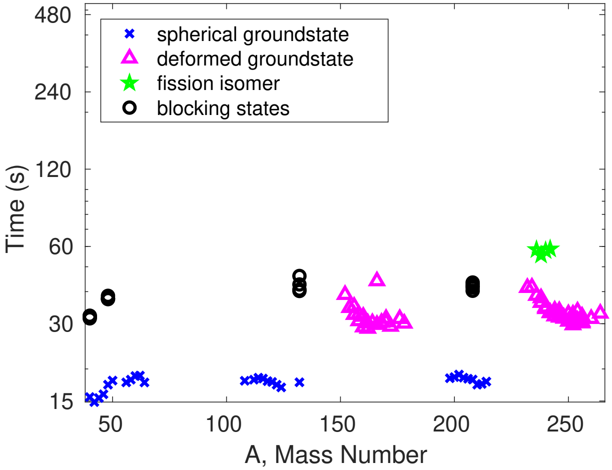


Exploiting Concurrency is Vital in the Supercomputing Era

Considerations:

- Load balancing
- Variability in run times for a particular nuclei or observable
- Variability in run times across observables
- Degree to which you can predict the run time of an observables

Median: UNEDF2 nuclei, Broadwell 9 threads/nuclei

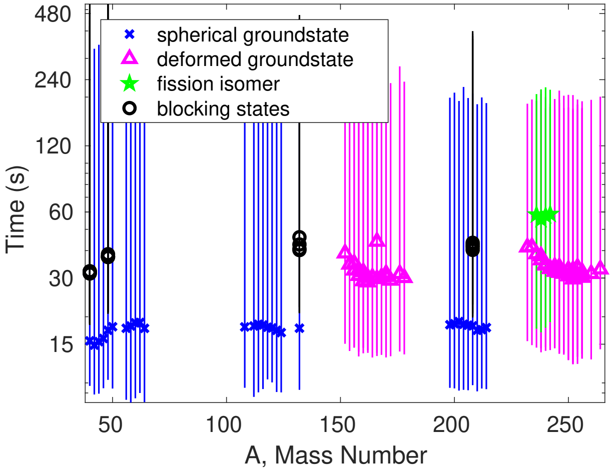


Exploiting Concurrency is Vital in the Supercomputing Era

Considerations:

- Load balancing
- Variability in run times for a particular nuclei or observable
- Variability in run times across observables
- Degree to which you can predict the run time of an observables

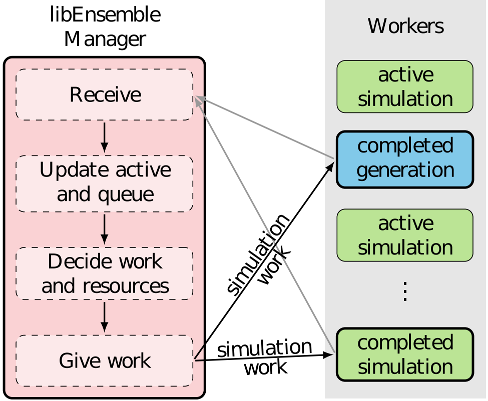
Extrema: UNEDF2 nuclei, Broadwell 9 threads/nuclei



LibEnsemble: Managing Tightly Coupled Ensembles of Calculations

Moving beyond local optimization requires (many) more forward model evaluations

- python based, available via Spack
- Tackles higher-level problems (optimization, UQ, Sensitivity analysis, machine learning, stochastic sampling, . . .
- Graceful exit of libEnsemble when time has expired or when persistent/nonpersistent worker(s) are unresponsive/busy
- Simulations can be PETSc-based or use their own communicator objective



Related: Training in Supervised Learning

Obtain model prediction $S(\cdot, \mathbf{x})$ by solving

$$\min_{\mathbf{x}} \sum_{i=1}^{N} l\left(S(\boldsymbol{\theta}^i, \mathbf{x}), y^i\right)$$

- $\mathbb{T} = \{(\boldsymbol{\theta}^i, y^i)\}_{i=1}^{N} \subset \mathbb{R}^d \times \mathbb{R}$ — Training data
- $y^i \in \mathbb{R}$ — label associated with input $\boldsymbol{\theta}^i$
- $\mathbf{x} \in \mathbb{R}^n$ — weights
- $S : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}$ — trained model
- $l : \mathbb{R}^2 \to \mathbb{R}$ — loss function

e.g., $l(a, b) = (a - b)^2$

Related: Optimization Under Uncertainty

$\rightarrow$ $\mathbf{u}$ denotes vector of uncertain variables

Examples

- Stochastic optimization: $\mathbf{u} \sim P$

  $$\min_{\mathbf{x}} \mathbb{E}_{\mathbf{u}}\left[F(\mathbf{x}, \mathbf{u})\right]$$

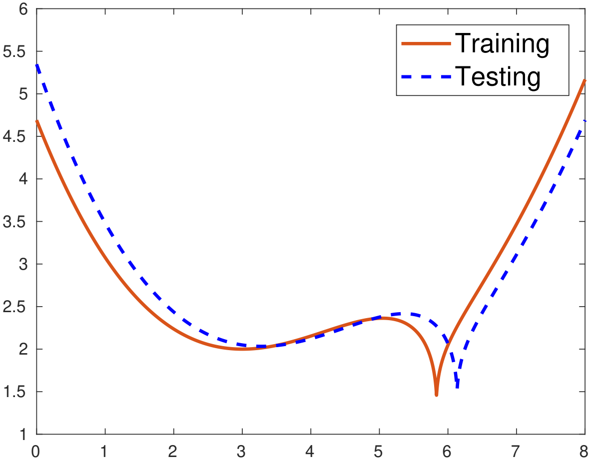
- Robust optimization: Guard against worst-case uncertainty in the problem data

  $$\min_{\mathbf{x}} \max_{\mathbf{u} \in \boldsymbol{\mathcal{U}}} f(\mathbf{x}, \mathbf{u}) \qquad \text{or} \qquad \min_{\mathbf{x}} \left\{ f(\mathbf{x}) : \ |R_i(\mathbf{x}; \mathbf{u})| \le \kappa \, \forall \mathbf{u} \in \boldsymbol{\mathcal{U}}, \forall i \right\}$$

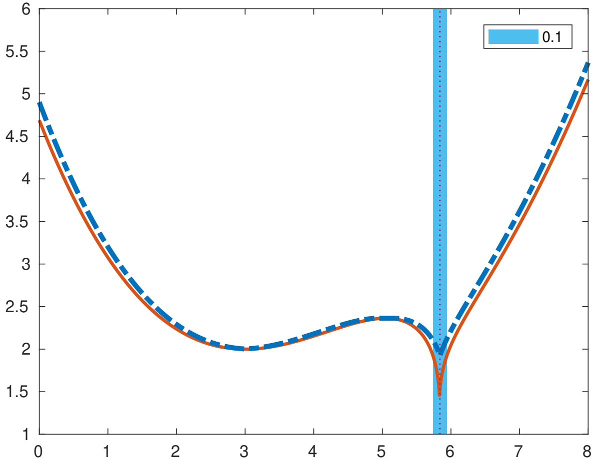
- Trimmed/quantile loss: determine outliers on the fly (as $\mathbf{x}$ changes)

  $$f(\mathbf{x}) = \sum_{i=1}^{q} \left| R_{(i)}(\mathbf{x}) \right| \qquad \text{where } \left| R_{(i)}(\mathbf{x}) \right| \le \left| R_{(i+1)}(\mathbf{x}) \right|, \ i = 1, \ldots, p-1 \left(\ge q \right)$$

Robust Optimization: Deterministic Incorporation of Robustness Desires



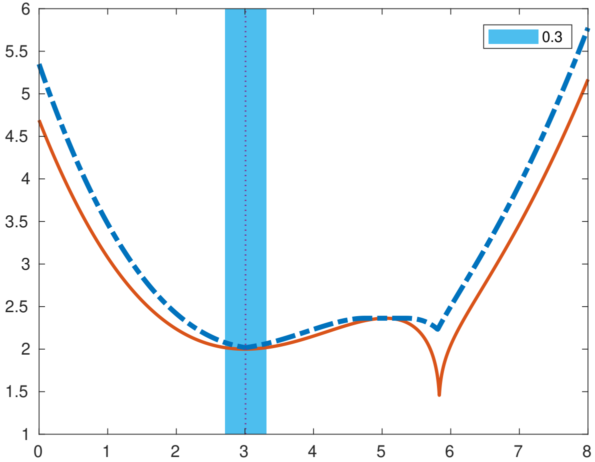
Robust Optimization: Deterministic Incorporation of Robustness Desires



$$\Psi(\mathbf{x}) = \max_{\mathbf{u}} \left\{ f(\mathbf{x} + \mathbf{u}) : \|\mathbf{u}\| \leq \alpha \right\}$$

Game: You choose $\mathbf{x}$ to minimize $\Psi(\mathbf{x})$, opponent chooses $\mathbf{u}$ to maximize $f(\mathbf{x} + \mathbf{u})$

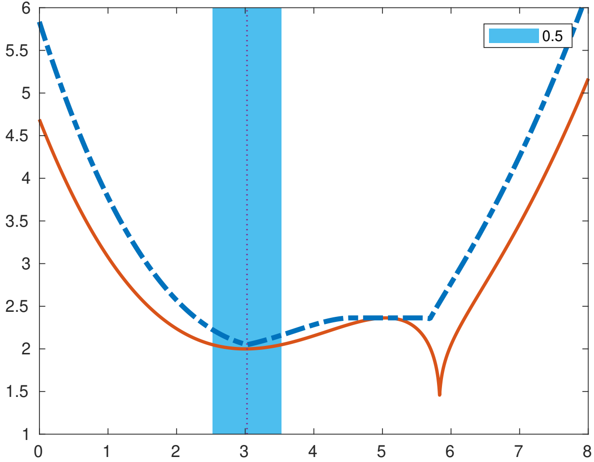
Robust Optimization: Deterministic Incorporation of Robustness Desires



$$\Psi(\mathbf{x}) = \max_{\mathbf{u}} \{ f(\mathbf{x}+\mathbf{u}) : \|\mathbf{u}\| \leq \alpha \}$$

Game: You choose $\mathbf{x}$ to minimize $\Psi(\mathbf{x})$, opponent chooses $\mathbf{u}$ to maximize $f(\mathbf{x}+\mathbf{u})$

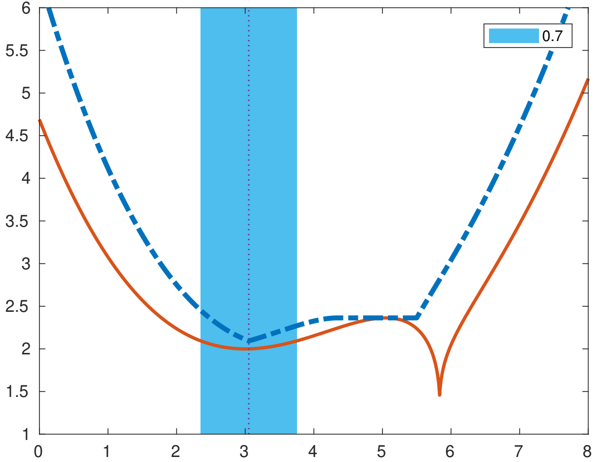
Robust Optimization: Deterministic Incorporation of Robustness Desires



$$\Psi(\mathbf{x}) = \max_{\mathbf{u}} \left\{ f(\mathbf{x}+\mathbf{u}) : \|\mathbf{u}\| \leq \alpha \right\}$$

Game: You choose $\mathbf{x}$ to minimize $\Psi(\mathbf{x})$, opponent chooses $\mathbf{u}$ to maximize $f(\mathbf{x}+\mathbf{u})$

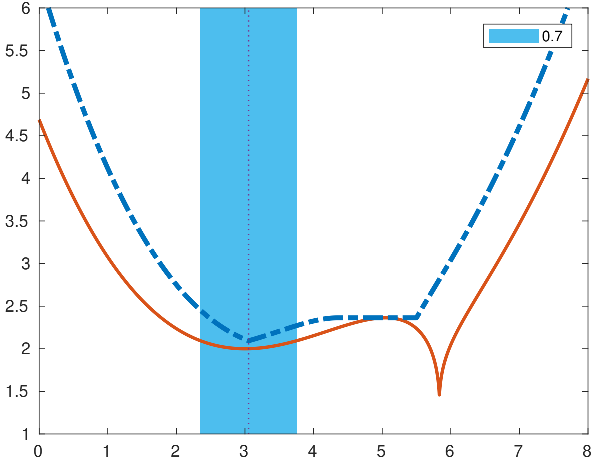
Robust Optimization: Deterministic Incorporation of Robustness Desires



$$\Psi(\mathbf{x}) = \max_{\mathbf{u}} \{f(\mathbf{x}+\mathbf{u}) : \|\mathbf{u}\| \leq \alpha\}$$

Game: You choose $\mathbf{x}$ to minimize $\Psi(\mathbf{x})$, opponent chooses $\mathbf{u}$ to maximize $f(\mathbf{x}+\mathbf{u})$

Robust Optimization: Deterministic Incorporation of Robustness Desires



$$\Psi(\mathbf{x}) = \max_{\mathbf{u}} \{ f(\mathbf{x}+\mathbf{u}) : \|\mathbf{u}\| \le \alpha \}$$

Game: You choose $\mathbf{x}$ to minimize $\Psi(\mathbf{x})$, opponent chooses $\mathbf{u}$ to maximize $f(\mathbf{x}+\mathbf{u})$

Possible challenges

? Ability to compute $\Psi(\mathbf{x})$

$\ldots \partial\Psi(\mathbf{x})$

? Determination of $\alpha > 0$

. . . uncertainty set

Ex.- $\mathcal{U} = \{\mathbf{u} : \|\mathbf{u}\| \le \alpha\}$

Optimization, UQ, Supercomputing, and Nuclear Theory

- Exploiting structure yields better solutions, in fewer simulations
- Optimization problem formulation matters
- Supercomputing is opening algorithmic frontiers for calibration under uncertainty
- Expanded opportunity for scalable parallelism through optimization, sensitivity analysis, UQ

Optimization, UQ, Supercomputing, and Nuclear Theory

- Exploiting structure yields better solutions, in fewer simulations
- Optimization problem formulation matters
- Supercomputing is opening algorithmic frontiers for calibration under uncertainty
- Expanded opportunity for scalable parallelism through optimization, sensitivity analysis, UQ

http://www.mcs.anl.gov/~wild

(Get in touch!)

Thank you!