SAND2016-0894PE

Dakota Software Training

Optimization

http://dakota.sandia.gov

Exceptional service in the national interest

Module Learning Goals

- Understand why you might want to perform optimization
- Have terminology and a practical process for design optimization at your disposal
- Be able to formulate your problem and present it to Dakota
- Know how to select a Dakota optimization method
- Be able to formulate, run, and interpret initial Dakota optimization studies

Module Outline

- Examples to illustrate why you might use optimization
- Optimization process, goals, and terminology
- Posing basic problem information to Dakota

- Exercise: basic optimization with various methods

- Selecting an optimization method
- Basic solution approaches (methods)
- Presenting constrained optimization problems to Dakota

- Exercise: constrained optimization
- Exercise: global optimization competition

Why Use Optimization?

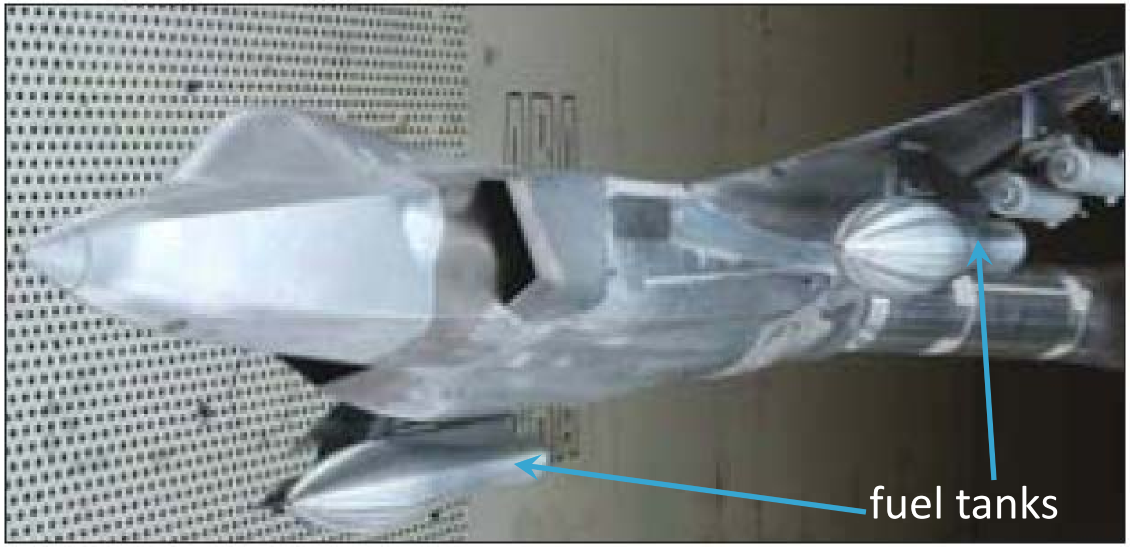
What: Determine parameter values that yield extreme values (max/min) of objectives, while satisfying constraints.

Why?

- Identify system designs with maximal performance
  - E.g., case geometry that minimizes drag and weight, yet is sufficiently strong and safe
- Determine operational settings that maximize system performance
  - E.g., fuel re-loading pattern yielding a sufficiently smooth nuclear reactor power distribution while maximizing power output
- Identify minimum-cost system designs/operational settings
  - E.g., delivery network that minimizes cost while meeting environmental limits
- Identify best/worst case scenarios
  - E.g., impact conditions that incur the most damage

- Calibration (specialized subset of optimization): adjust parameters to maximize agreement between model and data, another model, or a desired target. *Addressed in separate module*

Example: SNL/Lockheed Martin Optimization of F-35 External Fuel Tank



This wind tunnel model of F-35 features an optimized external fuel tank.

F-35: stealth and supersonic cruise

~ $20 billion cost

~ 2600 aircraft (USN, USAF, USMC, UK & other foreign buyers)

Simulate with LM CFD code

- *Expensive (at the time)*: 8 hrs/job on 16 processors
- Fluid flow around tank *highly sensitive* to shape changes

Optimization Problem

- Goal: Minimize DRAG and YAW over possible values of shape parameters
- Shape parameters must be bounded to fit within prescribed area
- Design must be sufficiently safe and strong

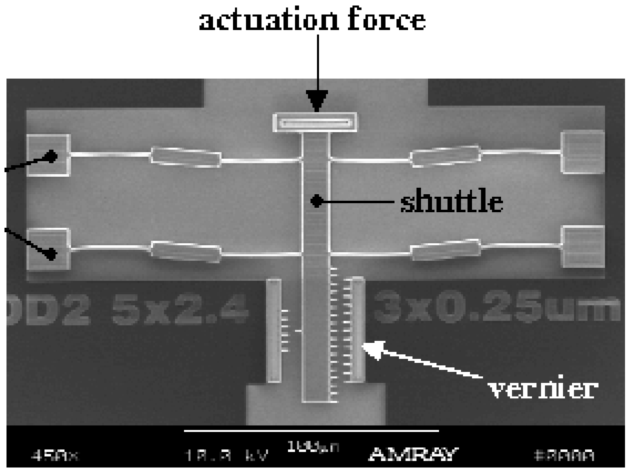
Objective Function: quantity for which we are trying to find the extreme value over parameter ranges

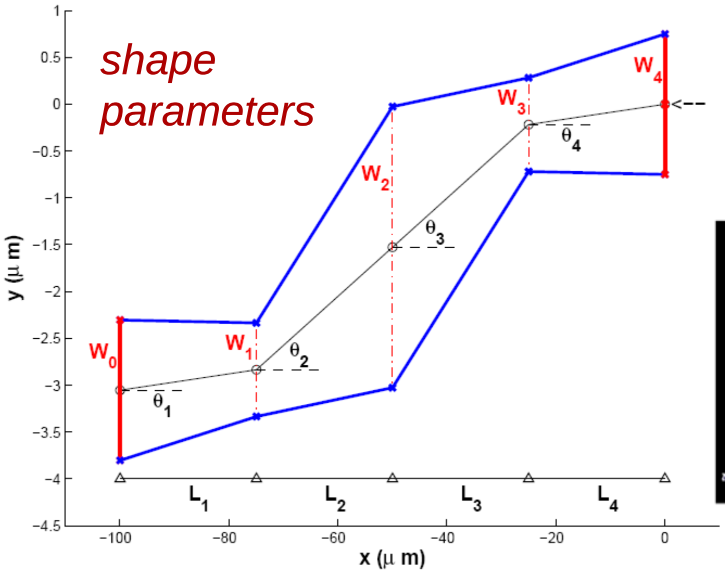
Parameters: quantities to be varied

Constraints: conditions that cannot be violated

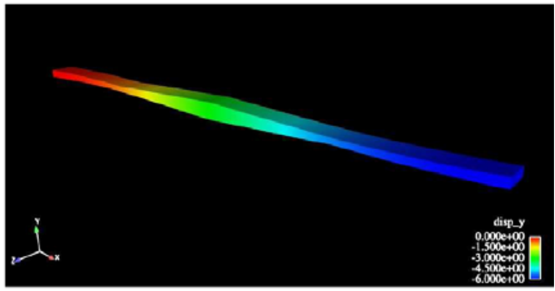
Example: MEMS Shape Optimization

- Micro-electromechanical systems: typically made from silicon, polymers, or metals; used as micro-scale sensors, actuators, switches, and machines
- Manufactured MEMS are highly variable in performance/reliability due to materials and micromachining, photo lithography, etching processes
- Goal: shape optimize a bistable MEMS switch to…
  - Achieve prescribed reliability in actuation force, while meeting stress, max force limits
  - Minimize sensitivity to uncertainties (robustness); maintain bi-stability





for each candidate design, finite element analysis predicts force/displacement relationship



Practical Process for Optimization

1. What are your optimization goals and restrictions? Consider:
   - Design problem or other (function/simulator output)?
   - What / how much improvement defines success?
2. Identify design variables; specific objectives and constraints
3. What are the model characteristics/behaviors? Recall:
   - Simulation cost, model robustness, input/output properties such as kinks, discontinuities, multi-modal, noise, disparate regimes

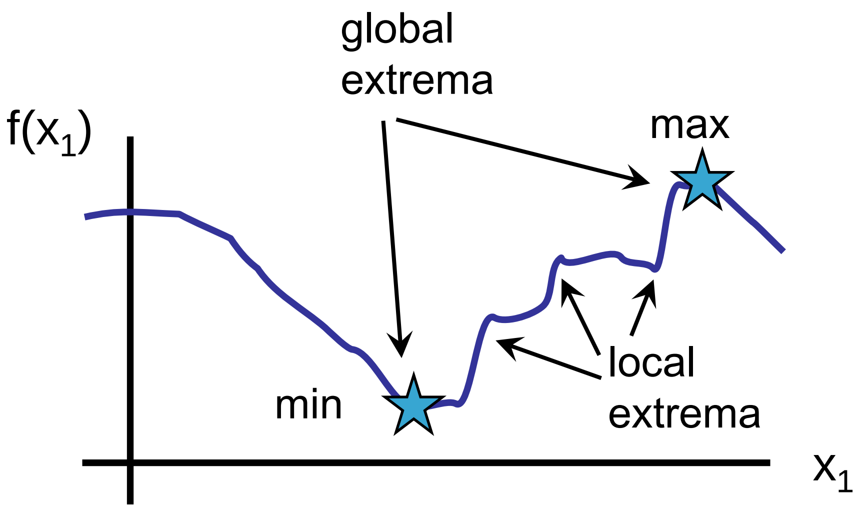
   covered in other modules
4. Use sensitivity analysis to screen parameters and down-select to those influencing trends in objective and constraints

   covered in other modules
5. Select a method appropriate to variables, goal, and problem
6. Set up Dakota input file and interface to simulation

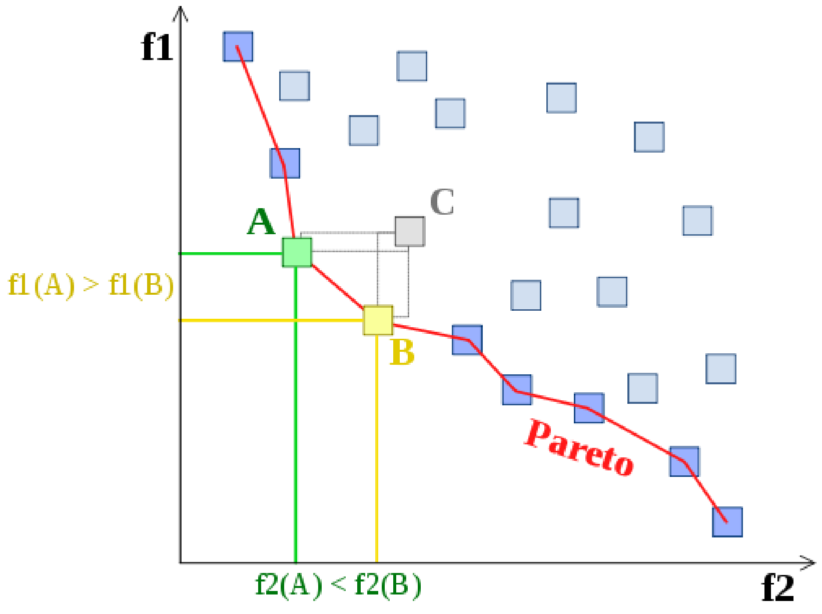
   later in this module
7. Run study and interpret results; refine as needed

Up next: Discuss 1 and 2, relate to Dakota, and see a simple example of 6, 7

Optimization Goals Come in Multiple Forms



- Some applications: local improvement suffices, e.g., 5% performance improvement
- Others: must find global minimum at any cost



May want tradeoffs between multiple objectives, e.g., cost vs. risk *(advanced topic)*

Anatomy of an Optimization Problem: Mapping to Dakota vs. Simulation

Computed by simulation and reported to Dakota

minimize:	$f(x_1,\ldots,x_N)$	***objective function(s)***
subject to:	$g_{LB} \le g(x) \le g_{UB}$	***nonlinear inequality constraints***
	$h(x) = h_E$	***nonlinear equality constraints***
	$A_I x \le b_I$	***linear inequality constraints***
	$A_E x = b_E$	***linear equality constraints***
	$x_{LB} \le x \le x_{UB}$	***bound constraints***

Specified in Dakota input file

Constraints will be addressed later. For now, we'll focus on continuous variables x and one objective function f(x).

Anatomy of an Optimization Problem: Mapping to Dakota Input File Blocks

Computed by simulation and reported to Dakota

minimize:	$f(x_1,\ldots,x_N)$	***objective function(s)***	*interface, responses*
subject to:	$g_{LB} \le g(x) \le g_{UB}$	***nonlinear inequality constraints***	
	$h(x) = h_E$	***nonlinear equality constraints***	
	$A_I x \le b_I$	***linear inequality constraints***	*method*
	$A_E x = b_E$	***linear equality constraints***	
	$x_{LB} \le x \le x_{UB}$	***bound constraints***	*(design) variables*

Dakota Example: Optimization for the Fuel Tank Problem

A possible simple statement of the fuel tank optimization problem, with Dakota input:

Find the fuel tank shape, parameterized by design variables s1, s2, that minimizes drag. Hold the tank angle fixed at 85.0 degrees.

Mathematically:

minimize (over s1, s2)

drag(s1, s2)

such that

$4.5 < s1 < 6.7$

$0.1 < s2 < 2.3$

```
method
  optpp_q_newton

variables
  continuous_design = 2
    descriptors    's1'   's2'
    upper_bounds   6.7    2.3
    lower_bounds   4.5    0.1

  continuous_state = 1
    descriptors 'angle'
    initial_state = 85.0

responses
  descriptors   'drag'
    objective_functions = 1
  numerical_gradients
  no_hessians
```

Brief Group Discussion: Optimization Practice

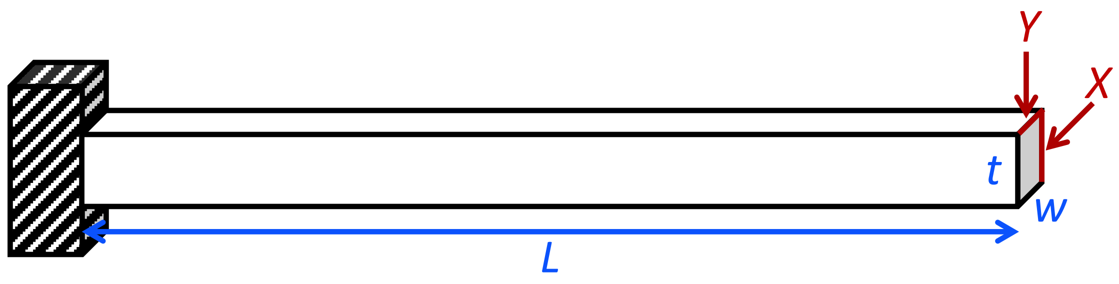
- Give an example of a simple optimization problem you might care about in your personal life.

- What types of system design, performance, and cost questions do you ask in your work domain?
- What metrics do you use to assess design quality, performance level, and costs?

- How do you answer your questions currently?
- What are the key challenges you face?

- What challenges do you foresee with automated design optimization with Dakota?

Discussion: Optimize Cantilever Beam



Parameters:

L: length (in)

w: width (in)

t: thickness (in.)

ρ: density (lb/ft^3)

E: Young's modulus (lb/in^2)

X: horizontal load (lb)

Y: vertical load (lb)

Responses:

M: mass (lb)

S: stress (lb/in^2)

D: displacement (in)

- What might be some optimal design objectives (goals) of interest?
- What might you have control over (variables) to meet those goals?
- What model characteristics do you recall from previous module?
- What might you expect the results of optimizing a design to be?

Exercise 1: First Optimization Studies

Explore:

- The directory exercises/optimization/1 contains three Dakota input files that perform simple bound-constrained optimization on the cantilever beam problem
- Run each study and examine the final results near the end of the console output: best parameters, objective values, number of function evaluations

Discuss with your neighbor:

- Which are the design variables? Fixed parameters? What is being minimized?
- What is different between the various input files?
- What differs in the runtime behavior and final results summary?

Module Outline

- Examples to illustrate why you might use optimization
- Optimization process, goals, and terminology
- Posing basic problem information to Dakota

- Exercise: basic optimization with various methods

Coming Next:

- Selecting an optimization method
- Basic solution approaches (methods)
- Presenting constrained optimization problems to Dakota

- Exercise: constrained optimization
- Exercise: global optimization competition

Practical Process for Optimization

1. What are your optimization goals and restrictions? Consider:
   - Design problem or other (function/simulator output)?
   - What / how much improvement defines success?
2. Identify design variables; specific objectives and constraints
3. What are the model characteristics/behaviors? Recall:
   - Simulation cost, model robustness, input/output properties such as kinks, discontinuities, multi-modal, noise, disparate regimes

   covered in other modules
4. Use sensitivity analysis to screen parameters and down-select to those influencing trends in objective and constraints

   covered in other modules
5. Select a method appropriate to variables, goal, and problem
6. Set up Dakota input file and interface to simulation
7. Run study and interpret results; refine as needed

Based on variables, objectives, constraints, and properties, select an appropriate method…

Basic Classes of Optimization Approaches (the “method” block)

Gradient Descent

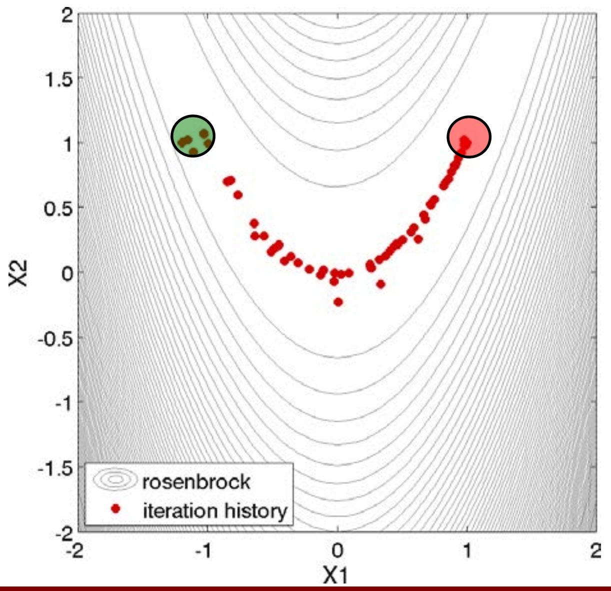
- Looks for improvement based on derivative
- Requires analytic or numerical derivatives *(more soon)*
- Efficient/scalable for smooth problems
- Converges to local extreme

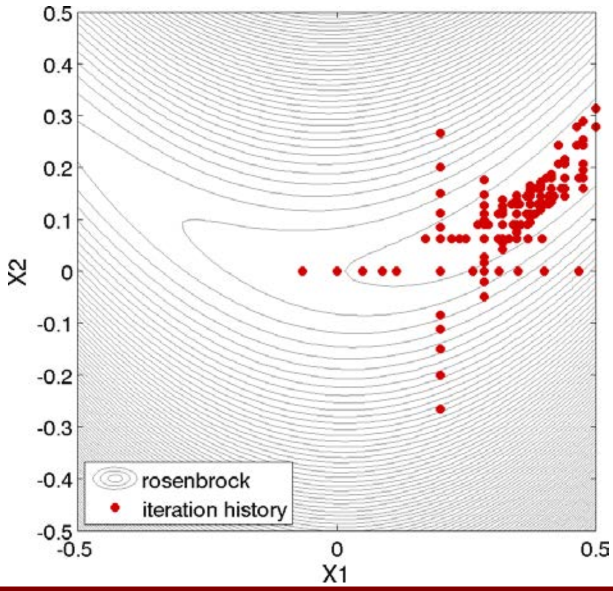
Derivative-Free Local

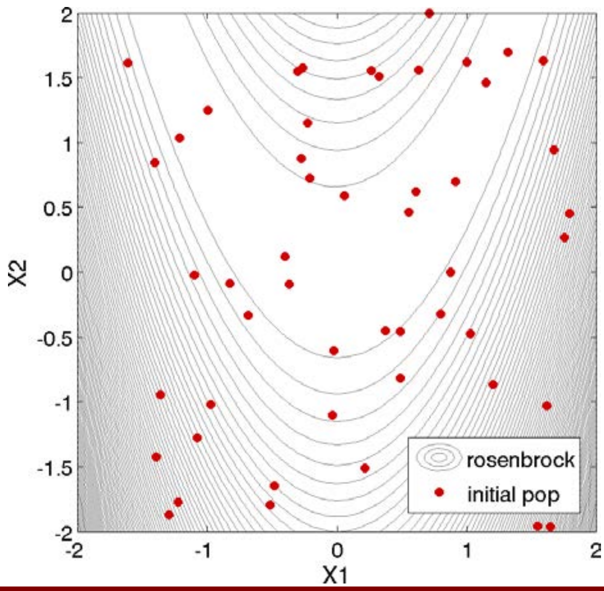
- Sampling with bias/rules toward improvement
- Requires only function values
- Good for noisy, unreliable or expensive derivatives
- Converges to local extreme

Derivative-Free Global

- Broad exploration with selective exploitation
- Requires only function values
- Typically computationally intensive
- Converges to global extreme

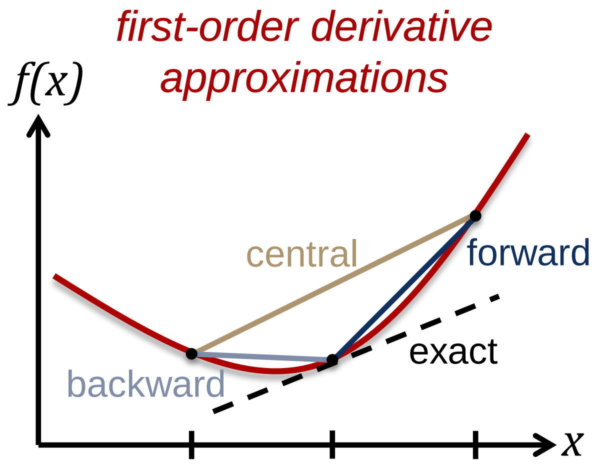






Gradients for Derivative-based Methods

- Akin to Newton's method for root-finding, minimize the objective by going "downhill" based on the gradient of the objective function: $\nabla f_x(x) = \left[\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_N} \right]$
- Most simulations don't calculate derivatives
- Dakota approximates gradients (and Hessians if needed) by running the simulation at $x \pm \Delta x$ as needed



First-order Forward Difference	Second-order Central Difference
$\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$	$\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x) - 2f(x) + f(x+\Delta x)}{2\Delta x}$

```
responses
  numerical_gradients
    forward
    fd_step_size 1.0e-3
```
	```
	responses
	  numerical_gradients
	    central
	    fd_step_size 1.0e-3
	```

Selecting a Method

A reasoning process: To select a Dakota method, ask yourself

- What is the improvement goal and (single- or multi-) objective ?
- Are there bound, linear, and/or nonlinear constraints? *(discussed later)*
- Are there discrete variables, or only continuous?
- What are the problem characteristics, including cost, robustness?
- *Get help from the Dakota team if you can't figure it out!*

Scenario 1

- Local optimization
- Unconstrained
- Can afford 10s to 100s of runs
- Unreliable gradients

Pattern Search

Scenario 2

- Global optimization
- Nonlinearly constrained
- Regions with different behavior
- Discrete variables

Genetic Algorithm

Guide to Optimization Methods

See Usage Guidelines in User's Manual

Category	Dakota method names	Continuous Variables	Categorical/ Discrete Variables	Bound Constraints	General Constraints
Gradient-Based Local (smooth)	`optpp_cg`	x			
	`dot_bfgs, dot_frcg, conmin_frcg`	x		x	
	`npsol_sqp, nlpql_sqp, dot_mmfd, dot_slp, dot_sqp, conmin_mfd, optpp_newton, optpp_q_newton, optpp_fd_newton`	x		x	x
Gradient-Based Global (smooth)	`hybrid, multi_start`	x		x	x
Derivative-Free Local (nonsmooth)	`optpp_pds`	x		x	
	`coliny_cobyla, coliny_pattern_search, coliny_solis_wets, surrogate_based_local`	x		x	x
	`asynch_pattern_search, mesh_adaptive_search`	x	x	x	x
Derivative-Free Global (nonsmooth)	`ncsu_direct, genie_direct, genie_opt_darts`	x		x	
	`coliny_direct, efficient_global, surrogate_based_global`	x		x	x
	`coliny_ea, soga, moga (multiobjective)`	x	x	x	x

For multi-objective problems: use weighted sum with any method, pareto_set, or moga.

CONSTRAINED OPTIMIZATION

Group Discussion: Constrained Optimization

- Give an example of a simple constrained optimization problem you might care about in your personal life.

- What kinds of constraints do you face in designing things for your work?

- For each, describe your example using Dakota optimization terminology.

Anatomy of an Optimization Problem: Mapping to Dakota Input File Blocks

Computed by simulation and reported to Dakota

minimize:	$f(x_1,\ldots,x_N)$	***objective function(s)***	*interface, responses*
subject to:	$g_{LB} \le g(x) \le g_{UB}$	***nonlinear inequality constraints***	
	$h(x) = h_E$	***nonlinear equality constraints***	
	$A_I x \le b_I$	***linear inequality constraints***	*method*
	$A_E x = b_E$	***linear equality constraints***	
	$x_{LB} \le x \le x_{UB}$	***bound constraints***	*(design) variables*

Example: Adding Constraints to Dakota Input for the Fuel Tank Problem

Minimize drag, given bounds on the shape parameters:

Mathematically:

minimize

drag(s)

such that

$4.5 < s1 < 6.7$

$0.1 < s2 < 2.3$

```
method
  optpp_q_newton






variables
  continuous_design  2
    descriptors    's1'  's2'
    upper_bounds   6.7   2.3
    lower_bounds   4.5   0.1

responses  # returned by simulation
  descriptors  'drag'
  objective_functions  1


  numerical_gradients
  no_hessians
```

Example: Adding Constraints to Dakota Input for the Fuel Tank Problem

Minimize drag, given bounds on the shape parameters, bounding the total cost of horizontal (s1) vs. vertical (s2) struts:

Mathematically:

minimize

drag(s)

such that

$4.5 < s1 < 6.7$

$0.1 < s2 < 2.3$

$22.0*s1 + 99.0*s2 < 199.0$

```
method
  optpp_q_newton
  linear_inequality_constraint_matrix
    22.0  99.0
  linear_inequality_upper_bounds
    199.0

variables
  continuous_design  2
    descriptors    's1'  's2'
    upper_bounds   6.7   2.3
    lower_bounds   4.5   0.1

responses  # returned by simulation
  descriptors  'drag'
  objective_functions  1


  numerical_gradients
  no_hessians
```

Example: Adding Constraints to Dakota Input for the Fuel Tank Problem

Minimize drag, …, and requiring sufficient strength

Mathematically:

minimize

drag(s)

such that

$4.5 < s1 < 6.7$

$0.1 < s2 < 2.3$

$22.0*s1 + 99.0*s2 < 199.0$

strength(s) > 5600.0

```
method
  optpp_q_newton
  linear_inequality_constraint_matrix
    22.0  99.0
  linear_inequality_upper_bounds
    199.0

variables
  continuous_design  2
    descriptors    's1'   's2'
    upper_bounds   6.7    2.3
    lower_bounds   4.5    0.1

responses  # returned by simulation
  descriptors  'drag'  'strength'
  objective_functions  1
  nonlinear_inequality_constraints  1
    nonlinear_inequality_lower_bounds  5600.0
  numerical_gradients
  no_hessians
```

Practical Process for Optimization

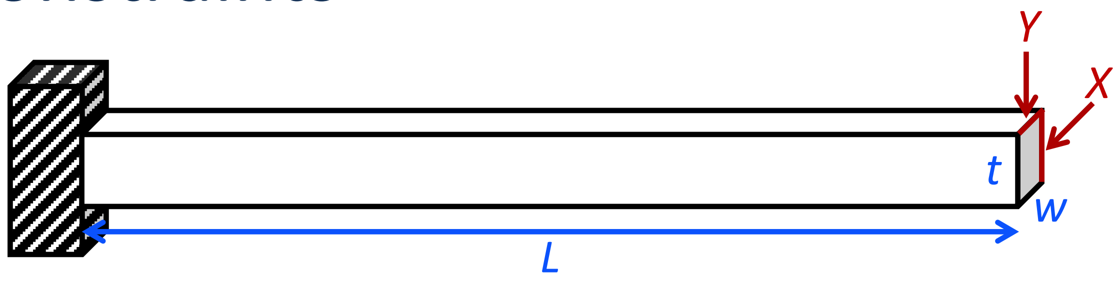
1. What are your optimization goals and restrictions? Consider:
   - Design problem or other (function/simulator output)?
   - What / how much improvement defines success?
2. Identify design variables; specific objectives and constraints
3. What are the model characteristics/behaviors? Recall:
   - Simulation cost, model robustness, input/output properties such as kinks, discontinuities, multi-modal, noise, disparate regimes

   covered in other modules
4. Use sensitivity analysis to screen parameters and down-select to those influencing trends in objective and constraints

   covered in other modules
5. Select a method appropriate to variables, goal, and problem
6. Set up Dakota input file and interface to simulation
7. Run study and interpret results; refine as needed

Let's put it all together with an exercise…

Discussion: Optimize Cantilever Beam with Constraints



Parameters:

L: length (in)

w: width (in)

t: thickness (in.)

ρ: density (lb/ft^3)

E: Young's modulus (lb/in^2)

X: horizontal load (lb)

Y: vertical load (lb)

Responses:

M: mass (lb)

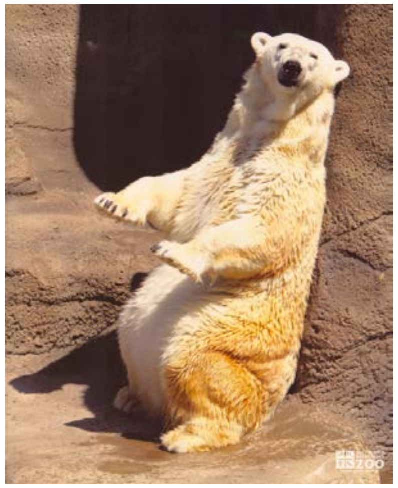
S: stress (lb/in^2)

D: displacement (in)

- What might be some optimal design objectives of interest?
- What are some design constraints that could come into play?
- What might you expect the results of optimizing a design to be?
- What methods are appropriate for your proposed constrained problem?

Exercise 2: Constrained Optimization of Cantilever Beams

Scenario: Your boss is concerned about the cost of the coat hooks you are proposing. She further insists that she can hang her chainmail coat on them and that the local wildlife can lean against them.


http://resourcelibrary.clemetzoo.com/photos/82

https://commons.wikimedia.org/wiki/File:
Eastern_riveted_armor.JPG

Use Dakota with the cantilever beam simulator to minimize the mass of the coat hooks, while ensuring that the displacement is not too great under loading and the *stress is manageable*.

Exercise 2: Constrained Optimization of Cantilever Beams

Exercise: Minimize the cantilever beam mass, while ensuring that the displacement is not too great and the stress is manageable. Specifically:

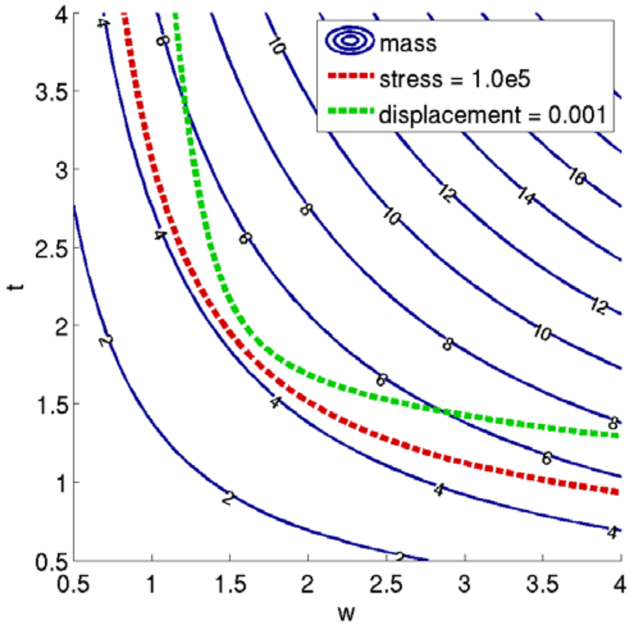
Modify `exercises/optimization/2/dakota_opt_cantilever.in` to:

- Design the cross section (w, t) of a 5 in long coat hook to minimize mass.
- Width and thickness must each be between 0.5 and 4.0 inches.
- Use state variables to enforce the operating constraints and materials:
  - Vertically support a 500 lb. chainmail coat
  - Support your office's resident 350 lb. female grizzly bear horizontally leaning on it
  - Made from steel with density ρ=500.0, Young's modulus E=2.9e+7
- The beam must displace no more than 0.001 in, with stress < 1.0e+5 lb/in^2.
- Choose a Dakota method appropriate to this problem (see method selection guide).
- Then compare to other methods or try different method controls.
- Compare your results to your neighbor.

Brief Group Discussion: Optimization Exercises

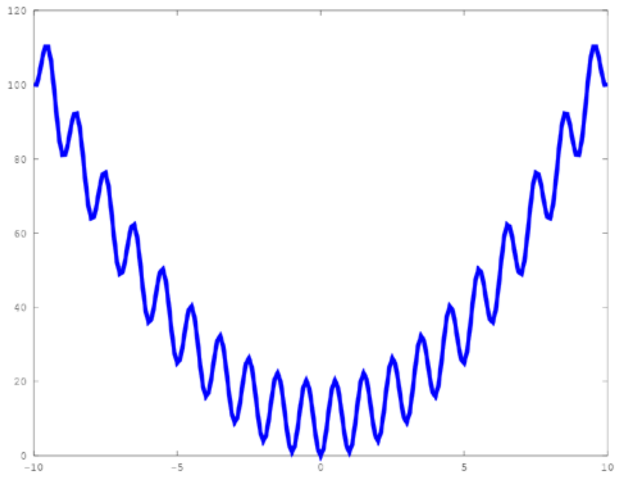
- Are the results what you expected? As good as your neighbor's? Why or why not?
- What do you see as the limitations of the methods used?
- What alternative methods might you try?

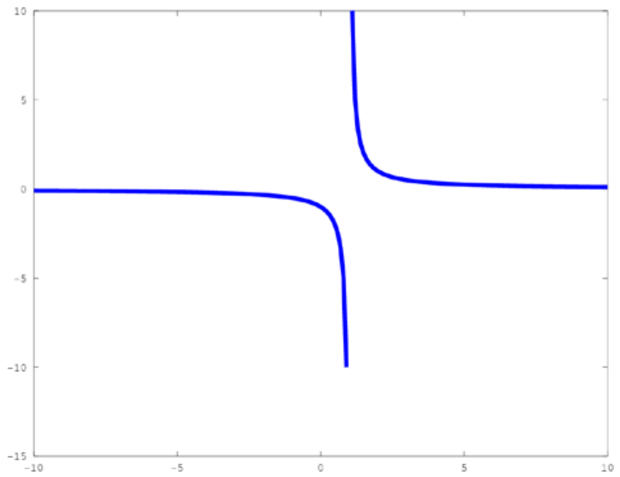
- What questions arose?

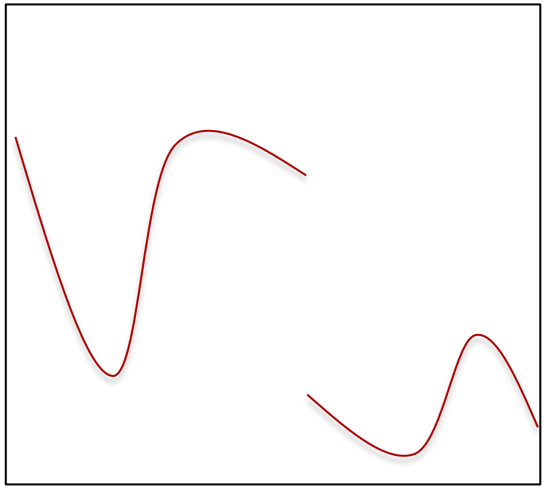


Discussion: Model Characteristics and Method Choice

- What challenges might you face in local optimization of each of these functions? Global?
- What methods might be applicable?







Exercise 3:
Global Optimization Competition

- See `exercises/optimization/3/dakota_opt_quasisine.in`
- Try to find the global minimum of the smooth, but multi-modal, quasi-sine function over the bounded domain [-1,1] x [-1,1]
- Choose one method to get the optimization working, then try as many as you like.
- For each method you try, keep track of the best point found and number of function evaluations required to get there.

- Compare your answers to a neighbor's.
- *The best answer in the class wins a prize!*

Advanced Optimization Topics

Not covered in this module, but can be discussed in office hours or advanced topics:

- Advanced methods, e.g., local and global surrogate-based optimization, meta-methods, details on specific methods
- Multi-objective optimization: analyzing tradeoff spaces
- Treating discrete or mixed continuous/discrete variables
- Optimization under uncertainty, robust design

Summary of Dakota Methods

Q: I’ve been using method X; is it available in Dakota?

A: Dakota has:

- Newton variants: MFP, NIP, SLP/SQP, FRCG/CG, Quasi for smooth, local
- Pattern/mesh search, Solis-Wets, COBYLA for noisy local without derivatives
- Genetic algorithms, OptDarts, DiRECT for global optimization
- Surrogate-based: efficient global (EGO), trust region local, global for costly simulations
- Branch and bound for mixed-integer
- Advanced Meta-methods: hybrid, multi-start, Pareto

Optimization References

- J. Nocedal and S. J. Wright, “Numerical Optimization”, Second Edition, Springer Science and Business Media, LLC, New York, New York, 2006.

- S. S. Rao, “Engineering Optimization: Theory and Practice”, Fourth Edition, John Wiley and Sons, Inc., Hoboken, New Jersey, 2009.

- Dakota User’s Manual
  - Optimization Capabilities
  - Surrogate-Based Minimization
  - Advanced Strategies
  - Advanced Model Recursions: Optimization Under Uncertainty
- Dakota Reference Manual

Practical Process for Optimization

1. What are your optimization goals and restrictions? Consider:
   - Design problem or other (function/simulator output)?
   - What / how much improvement defines success?
2. Identify design variables; specific objectives and constraints
3. What are the model characteristics/behaviors? Recall:
   - Simulation cost, model robustness, input/output properties such as kinks, discontinuities, multi-modal, noise, disparate regimes

   covered in other modules
4. Use sensitivity analysis to screen parameters and down-select to those influencing trends in objective and constraints

   covered in other modules
5. Select a method appropriate to variables, goal, and problem
6. Set up Dakota input file and interface to simulation
7. Run study and interpret results; refine as needed

Ask Dakota team or dakota-users list for help!

Module Learning Goals Revisited
Did We Meet Them?

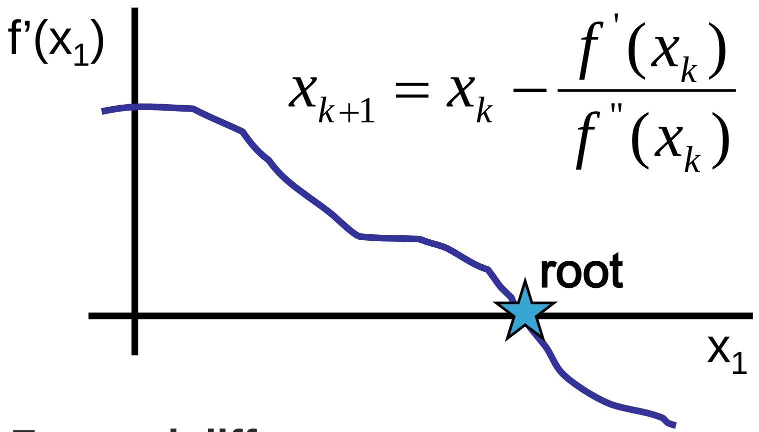
- Understand why you might want to perform optimization
- Have terminology and a practical process for design optimization at your disposal
- Be able to formulate your problem and present it to Dakota
- Know how to select a Dakota optimization method
- Be able to formulate, run, and interpret initial Dakota optimization studies

Method-related

BACKUP SLIDES

All following slides are strict subset of SAND2012-7265P

Variations on Gradient-Based Optimizers



$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

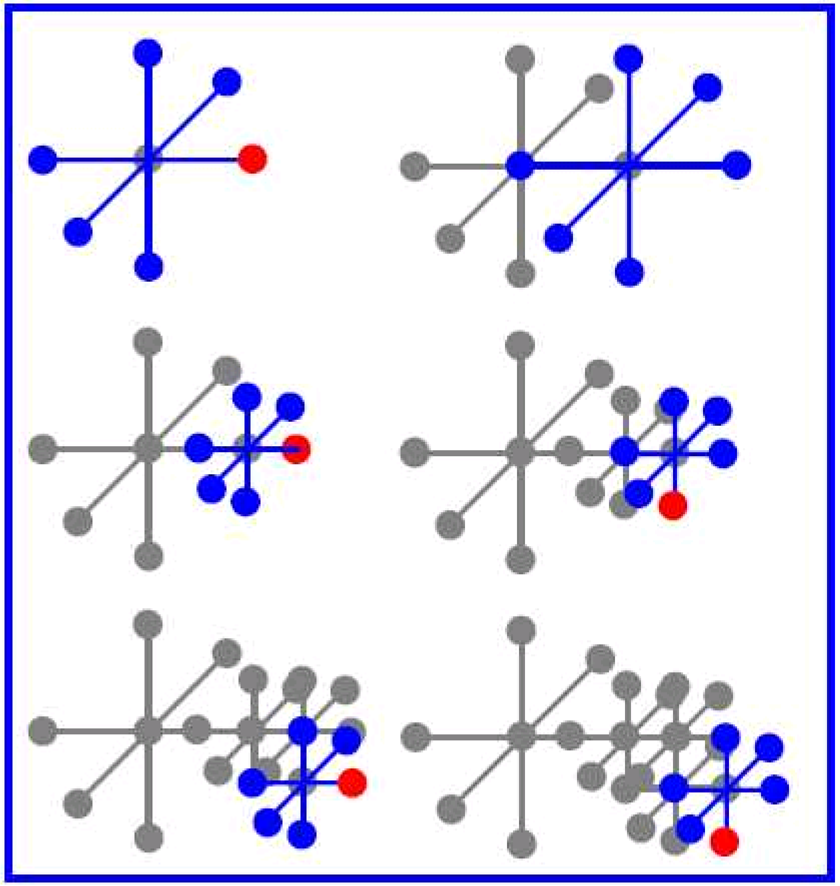
Forward difference

$$\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

- Go downhill
  - e.g., steepest descent, conjugate gradient, Newton and variants
  - second derivatives differentiate minima from maxima, inflection points; Hessian approximations often used in practice (quasi-Newton)
- Require reliable derivatives of objectives and nonlinear constraints w.r.t. decision variables:
  - analytic evaluation: code them into the simulation
  - finite differences: no code modification and provided by most optimizers
  - automatic differentiation: source transformation, operator overloading

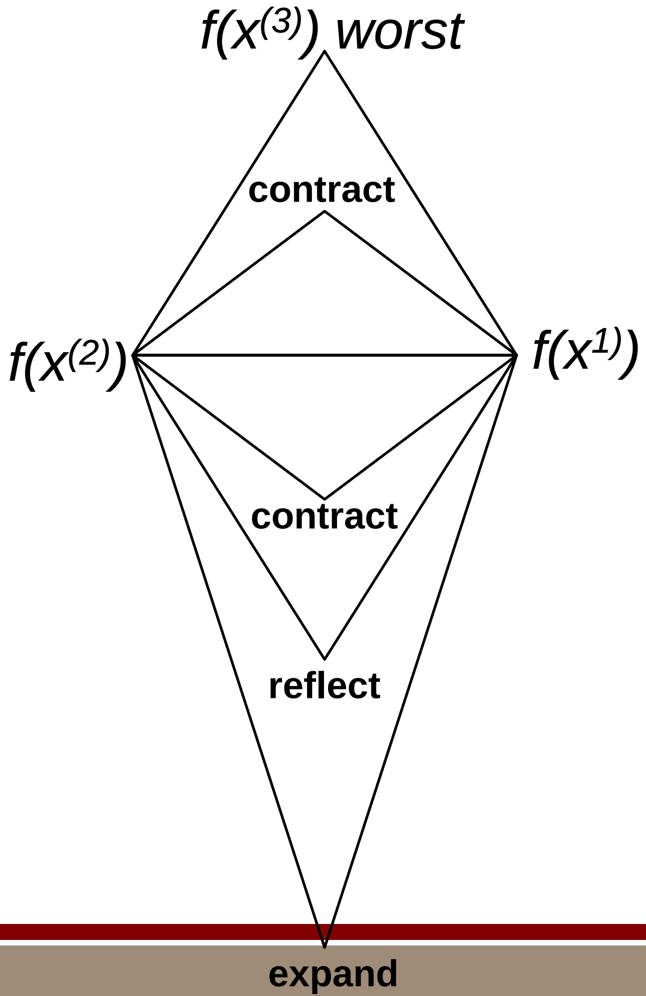
- **Strategies for managing convergence:**
  - **line search: find a step in the Newton direction to ensure sufficient decrease**
  - **trust region: use quadratic model in an expanding/contracting trust region**
- **Handling nonlinear constraints**
  - **reduced gradient**
  - **sequential linear or quadratic programming (SLP/SQP)**
  - **augmented Lagrangian or exact penalty methods**
  - **interior point / barrier, filter methods**

Variations on Derivative-Free Optimizers

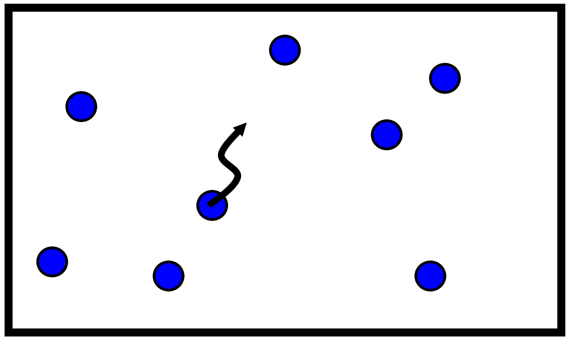


Pattern Search methods search using a stencil, often that defines some basis, that is iteratively re-centered and resized.

Nelder Mead searches using a simplex that is iteratively reflected through a centroid and resized.

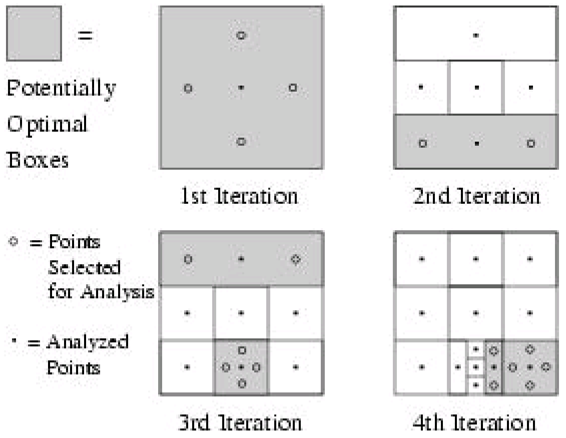


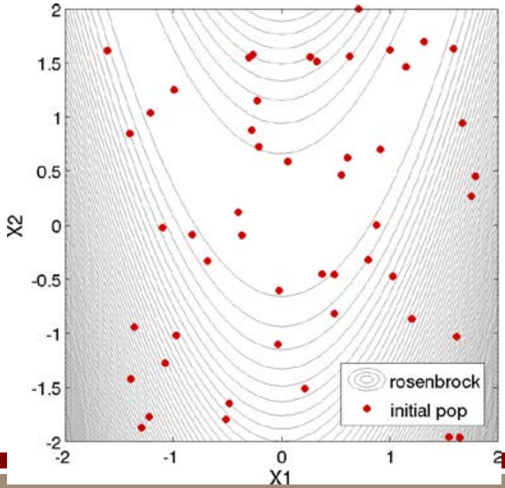
Variations on Global Optimizers

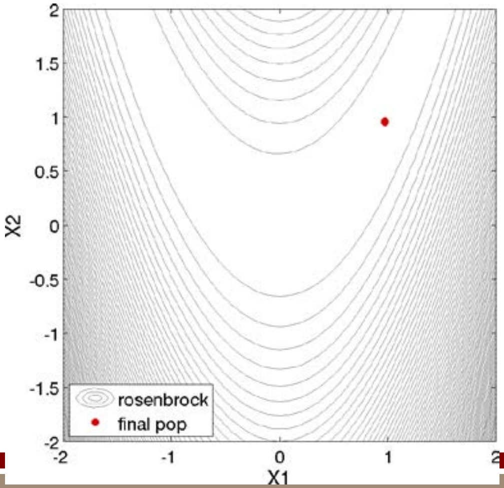


Multi-Start Local Optimization involves initiating a local optimization method at multiple points, with the goal of identifying multiple local minimizers from which the lowest can be chosen.

Division of RECTangles (DiRECT) iteratively subdivides the search domain based on size and rank of each existing subdivision.







Evolutionary/Genetic Algorithms evolve an initial random sample over generations, according a “fitness” function, until the minimum is found.

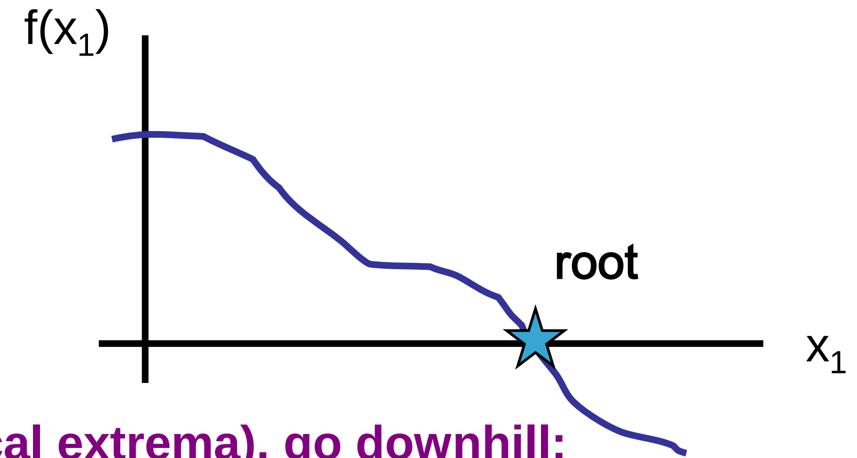
Optional Examples: Advanced Optimization Problems and Methods

- Constrained
  - Exercise: Minimize an objective given constraints
- Multi-start local
  - Exercise: Provide multiple starting points to a local optimizer to find multiple local minima
- Global
  - Exercise: Find the global extreme value
- Multi-objective
  - Exercise: Optimize across multiple competing objectives
- Surrogate-based/multifidelity
  - Exercise: Reduce the computational cost (i.e., number of function evaluations) of optimization
- Hybrid
  - Exercise: Use multiple optimization methods to solve a single problem

Gradient-based Optimization: Go Downhill

Modify Newton's root-finding method for solving f(x) = 0.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

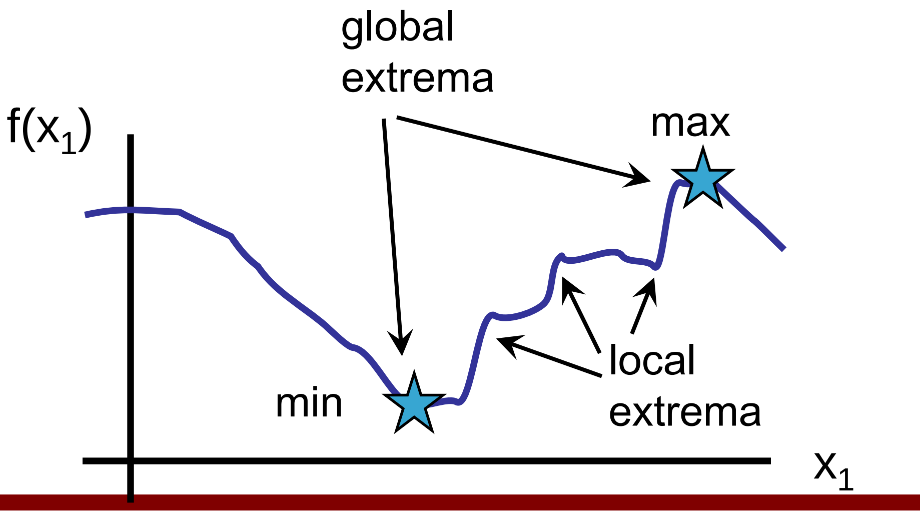


For optimization: find zeros of f'(x) = 0 (local extrema), go downhill; loosely

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

These derivatives extend to gradients and Hessians in the multivariate case:

$$\nabla_x f(x), \quad \nabla_x^2 f(x)$$



Constraint Progression

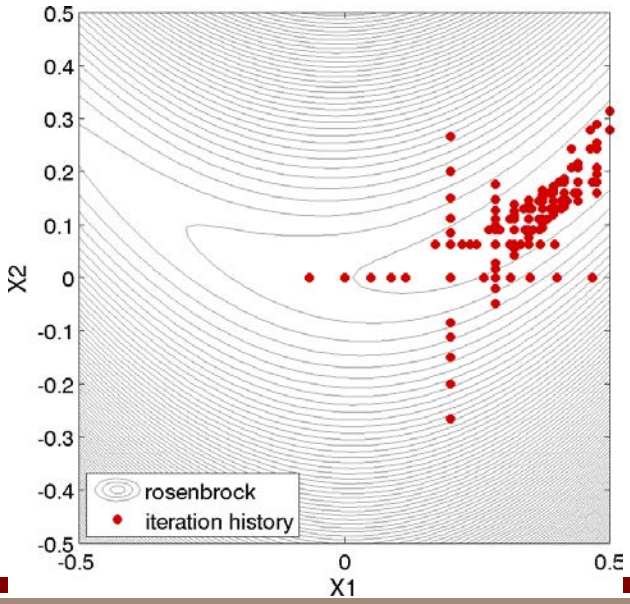
Listed in order of (typically) increasing algorithm complexity and computational cost needed to solve.

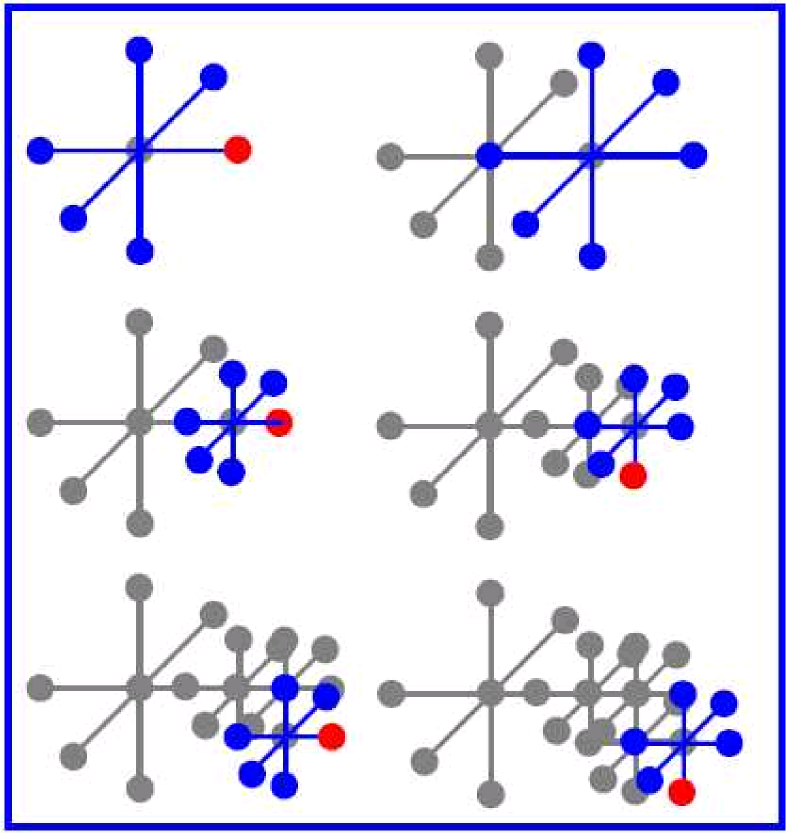
- Unconstrained problem: neither bound constraints nor linear/ nonlinear constraints
- Bound-constrained problem: bound (variable space x) constraints only (no linear/nonlinear constraints)
- Linearly-constrained problem: constraints are linear with respect to the x-variables (may also have bound constraints)
- Nonlinearly-constrained problem: the g(x) and h(x) constraints, nonlinear w.r.t. the x variables, are present (may also have bound constraints) *perhaps most typical in engineering and science applications*

Typically, it is important to specify constraints as specifically as possible, e.g., don't specify a linear constraint as nonlinear if the solver supports linear constraints.

Coordinate-basis Pattern Search

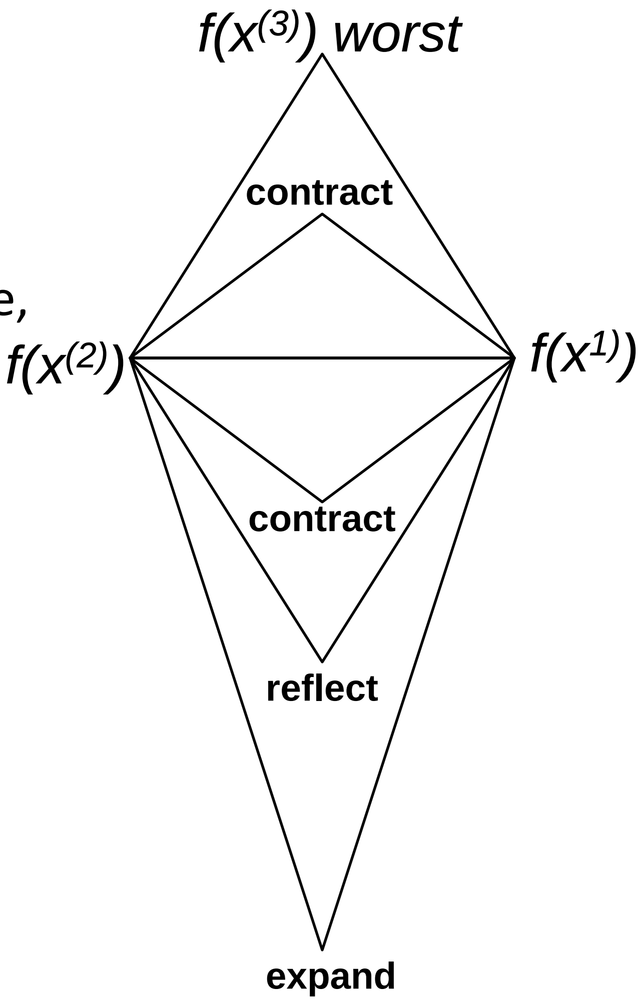
- Evaluate model at a stencil of points; recenter at point with best function value
- If no improvement, contract stencil (to achieve local convergence)
- Stencils typically are simplexes or align with coordinate directions





Nelder-Mead

- Simplex stencil of d+1 points,
  with centroid of d best
- Adapt the simplex iteratively to go downhill:
  - Reflect through centroid and if better value, replace worst
  - If improved, attempt to expand further in that direction
  - If no improvement, attempt contraction of the simplex to find a better point
  - If fails, shrink simplex

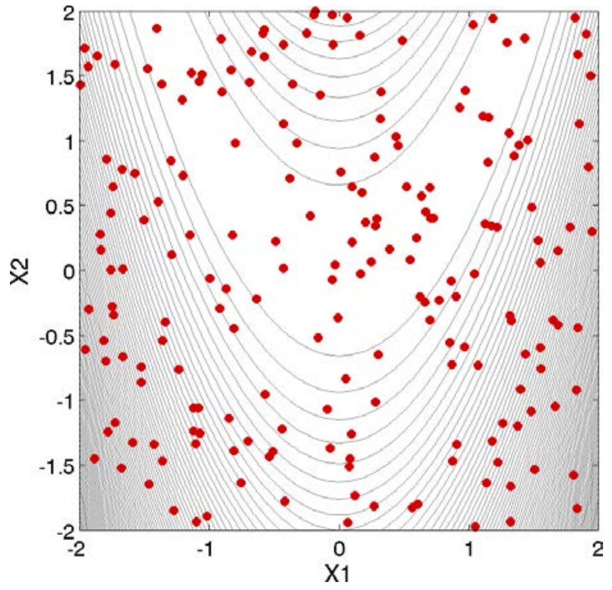


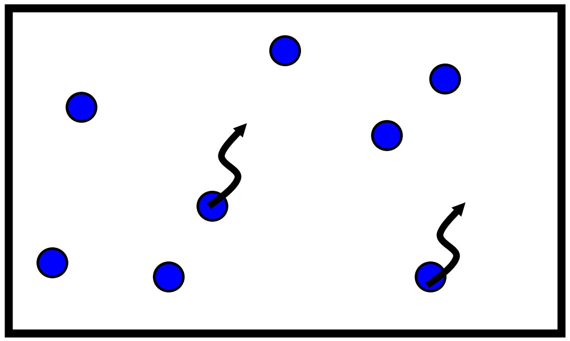
Global (derivative-free) Algorithms

- Global solvers attempt broad exploration of the design space with a strategy for selective exploitation of promising regions
- Can be extremely costly, but will deliver good results when you have a large computational budget
- A few approaches:
  - Random (Monte Carlo) sampling
  - Multi-start local search
  - Box decomposition, e.g., DIRECT and other Lipschitzian approaches
  - Population search, e.g., genetic/evolutionary algorithms
  - Global surrogate-based algorithms
  - *For more see Cindy's talk on Friday, including meta-heuristics, e.g., simulated annealing, tabu search, ant colony optimization*

Random and Multi-start

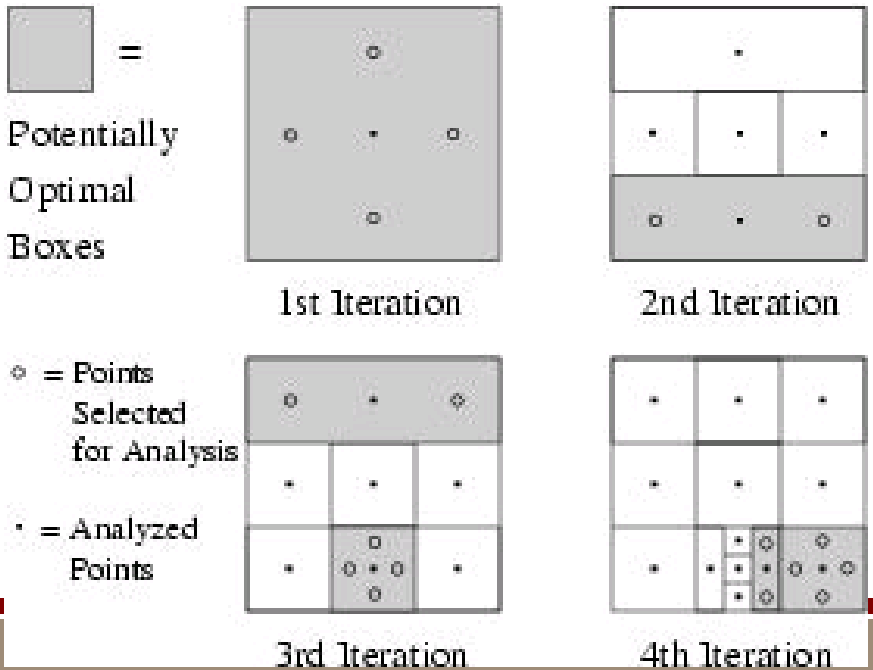
- Random sampling (Monte Carlo or more optimal space-filling designs) offers broad exploration;
  then just take the best point
- Can combine with local optimization by refining each of the set of promising optima using a local optimizer:





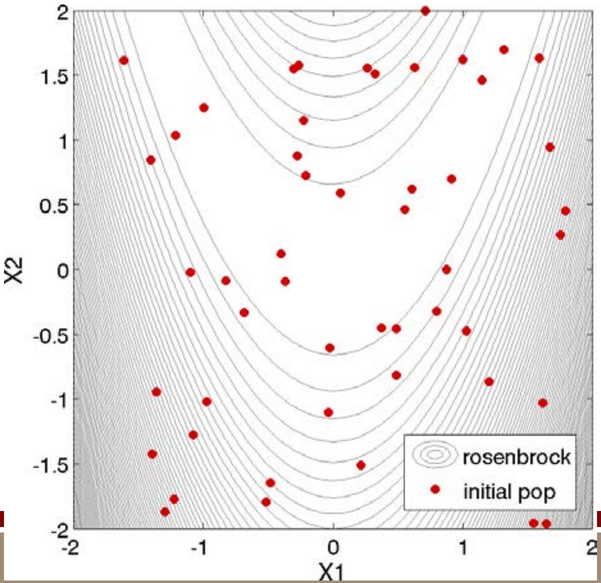
Division of RECTangles (DiRECT)

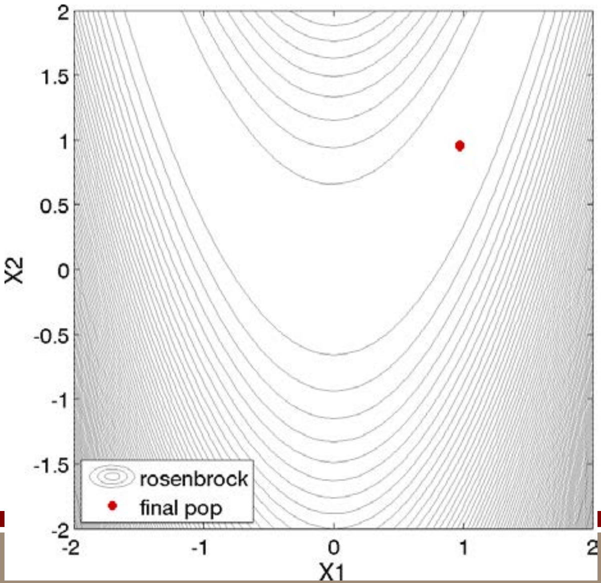
- Subdivide the search domain into non-overlapping ‘boxes’
- Boxes are ranked with estimates of their best value
- Boxes are selected for subdivision based on their rank and box size
- Successive refinement ensures that a near-optimal point will be found in finite time



Evolutionary/Genetic Algorithms

- Based on Darwin's theory of survival of the fittest
- Random initial population of design points
- Design parameters values are a unique "genetic string," analogous to DNA
- Sequence of generations, where most "fit" survive and reproduce
- Simulates natural selection, breeding, and mutation
- Ultimately identifies a design point (or family of points) satisfying optimization problem





Advanced Considerations

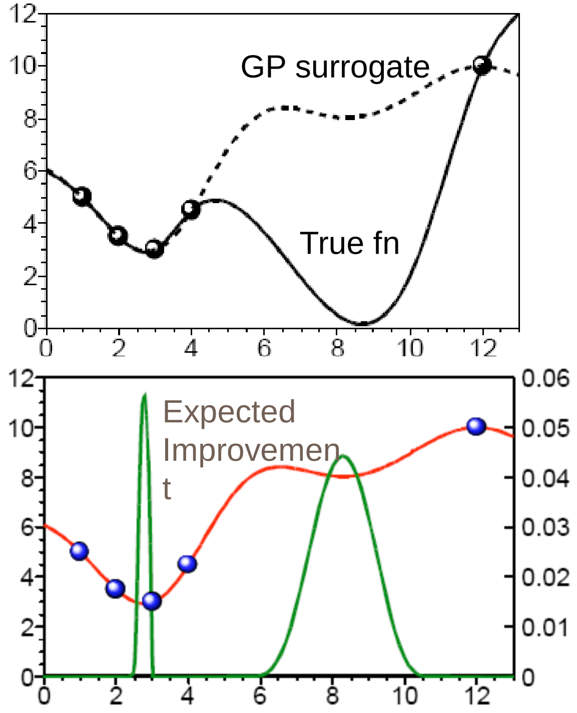
- Mixed-integer nonlinear programming (MINLP): not discussed here, but see Cindy's talk Friday
  - Use branch and bound over integer variables
  - For each discrete scenario, apply any of the nonlinear optimization techniques discussed here
- Multi-objective (trade-off) optimization
  - Few solvers treat directly (however some GAs and other heuristics can map out the necessary Pareto frontier)
  - Explicit weighting of objectives: limits intuition, but allows use of any solver:

$$f(x) = \sum_k \omega_k f_k(x)$$

- Surrogate-based optimization (local and global)
- Uncertainty: optimization under uncertainty and robustness (uncertainty of optima)

Efficient Global Optimization

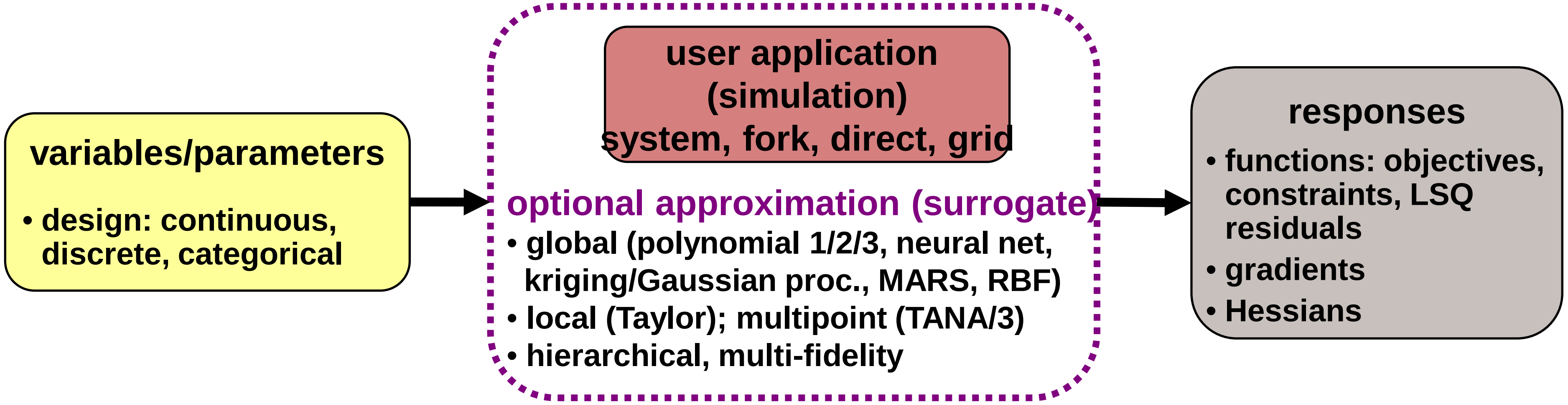
- Technique due to Jones, Schonlau, Welch
- Build global Gaussian process approximation to initial sample
- Balance global exploration (add points with high predicted variance) with local optimality (promising minima) via an "expected improvement function"



From Jones, Schonlau, Welch, 1998

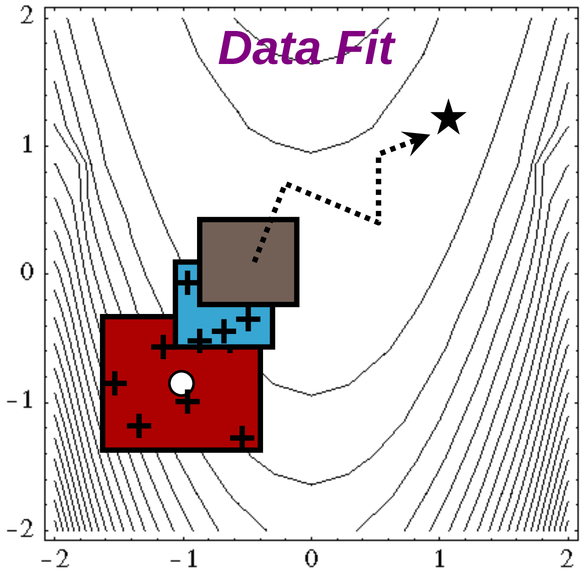
Surrogate-based Minimization (Calibration and Optimization)

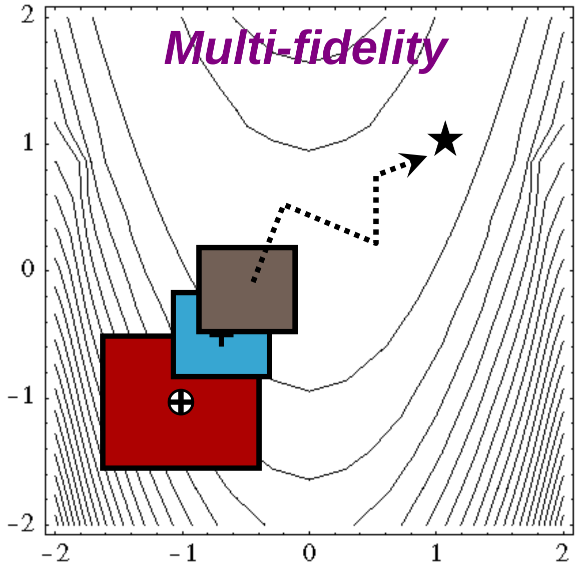
- Surrogate-based techniques replace or augment costly model evaluations with a less expensive stand-in; a key approach to make hard optimization problems tractable
- Response surface or meta-models are most common: use design of experiments to sample variable space and then build an approximation

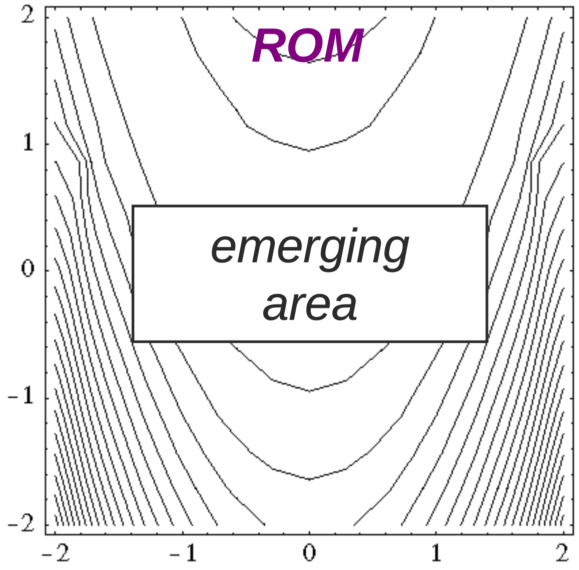

variables/parameters
• design: continuous, discrete, categorical
user application (simulation)
system, fork, direct, grid
optional approximation (surrogate)
• global (polynomial 1/2/3, neural net, kriging/Gaussian proc., MARS, RBF)
• local (Taylor); multipoint (TANA/3)
• hierarchical, multi-fidelity
responses
• functions: objectives, constraints, LSQ residuals
• gradients
• Hessians

- Multi-fidelity approaches useful when you have a physics-based surrogate, empirical approximation, or low-fidelity model option

Trust Region Surrogate-based Minimization







Data fit surrogates

- Global: polynomials, splines, neural network, Kriging, RBFs
- Local: 1st/2nd-order Taylor

Data fits in SBO

- Smoothing: extract global trend
- DACE: limited # design vars
- Must balance local consistency with global accuracy

Multifidelity surrogates:

- Coarser discretizations, looser conv. tols., reduced element order
- Omitted physics: e.g., Euler CFD, panel methods

Multifidelity SBO

- HF scale better w/ des. vars.
- Requires smooth LF model
- May require design mapping
- Correction quality is crucial

ROM surrogates:

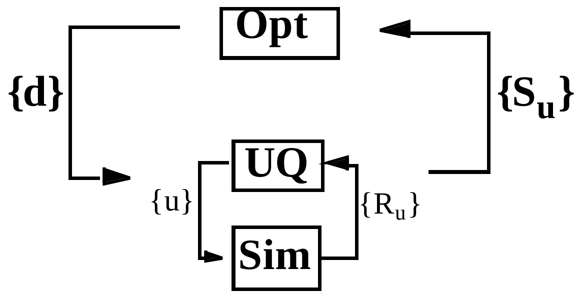
- Spectral decomposition
- POD/PCA w/ SVD
- KL/PCE (random fields, stochastic processes)

ROMs in SBO

- Key issue: parametrize (extended or spanning ROM)
- Otherwise like data fit case

Optimization Under Uncertainty

Rather than design and then post-process to evaluate uncertainty…
actively design optimize while accounting for uncertainty/reliability metrics
$s_u(d)$, e.g., mean, variance, reliability, probability:



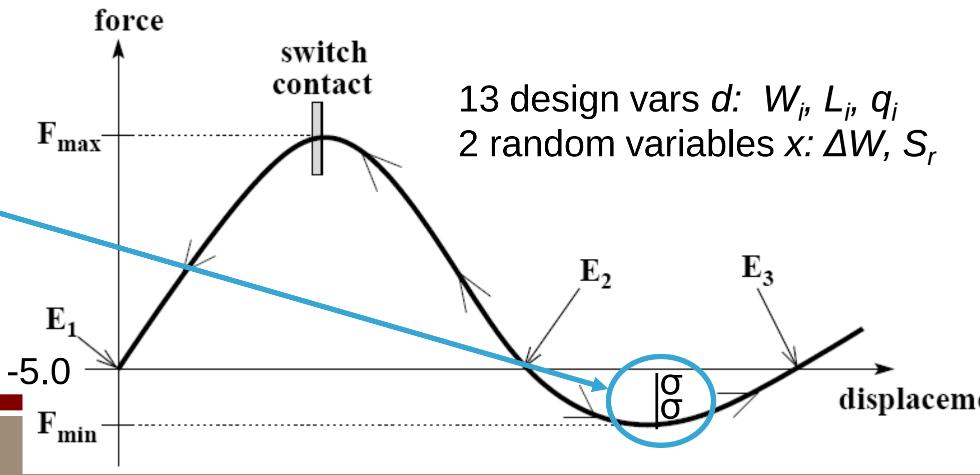
(nested paradigm)

$$
\begin{array}{ll}
\text{min} & f(d) + W s_u(d) \\
\text{s.t.} & g_l \le g(d) \le g_u \\
 & h(d) = h_t \\
 & d_l \le d \le d_u \\
 & a_l \le A_i\, s_u(d) \le a_u \\
 & A_e\, s_u(d) = a_t
\end{array}
$$

Bistable switch problem formulation (Reliability-Based Design Optimization):

simultaneously reliable and robust designs

$$
\begin{array}{lrcccl}
\text{max} & & & \text{E}\left[F_{min}(\mathbf{d},\mathbf{x})\right] & & \\
\text{s.t.} & 2 & \le & \beta_{ccdf}(\mathbf{d}) & & \\
 & 50 & \le & \text{E}\left[F_{max}(\mathbf{d},\mathbf{x})\right] & \le & 150 \\
 & & & \text{E}\left[E_2(\mathbf{d},\mathbf{x})\right] & \le & 8 \\
 & & & \text{E}\left[S_{max}(\mathbf{d},\mathbf{x})\right] & \le & 3000
\end{array}
$$

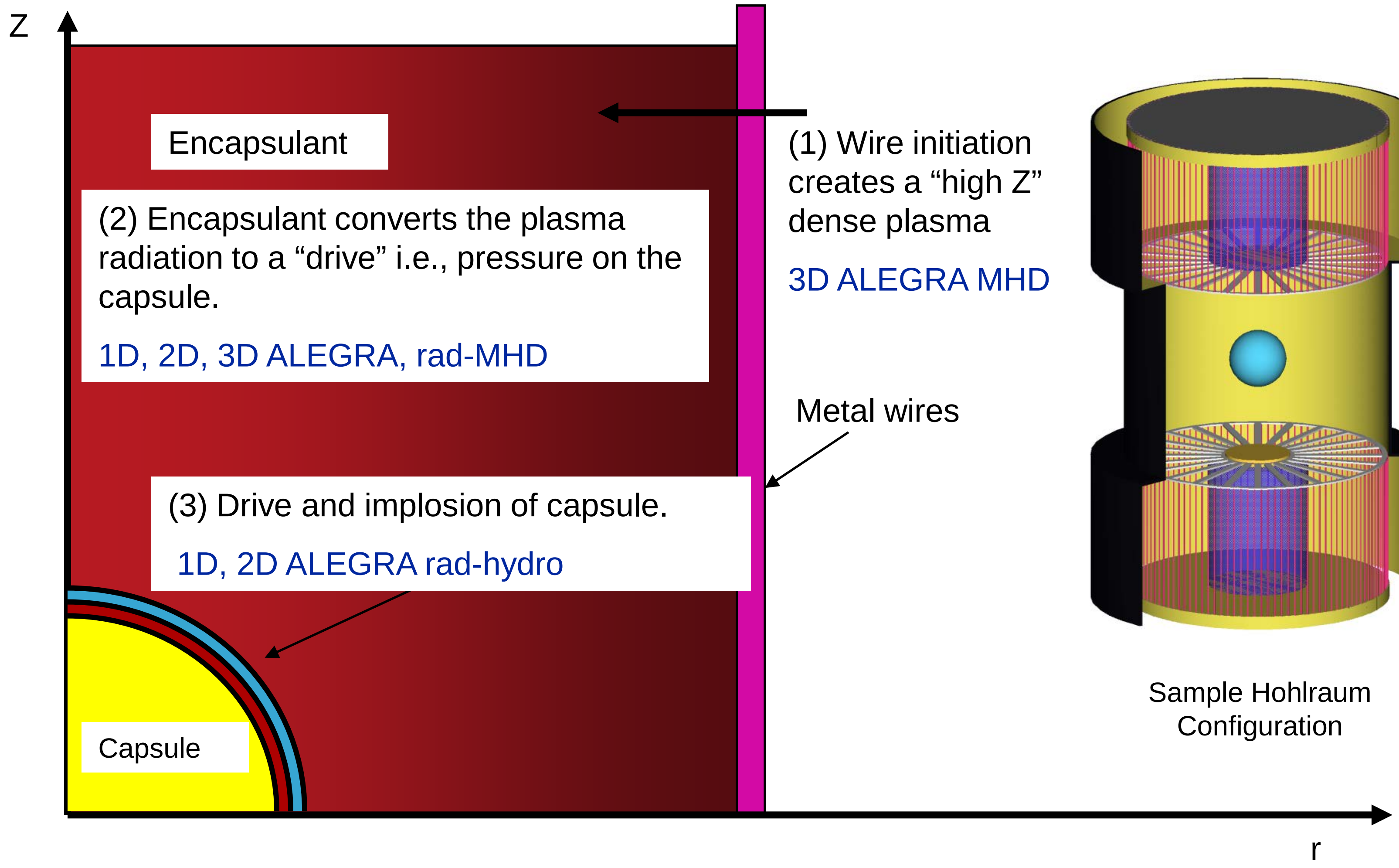


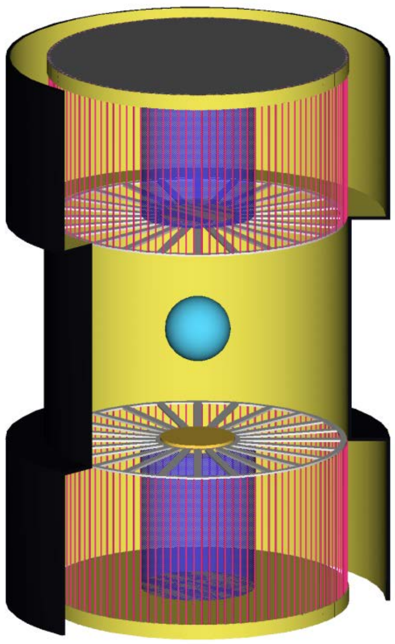
13 design vars d: W_i, L_i, q_i
2 random variables x: ΔW, S_r

Application Examples

BACKUP SLIDES

Robust Hohlraum Design for Inertial Confinement Fusion


Z
Encapsulant
(2) Encapsulant converts the plasma radiation to a "drive" i.e., pressure on the capsule.
1D, 2D, 3D ALEGRA, rad-MHD
(1) Wire initiation creates a "high Z" dense plasma
3D ALEGRA MHD
Metal wires
(3) Drive and implosion of capsule.
1D, 2D ALEGRA rad-hydro
Capsule
r

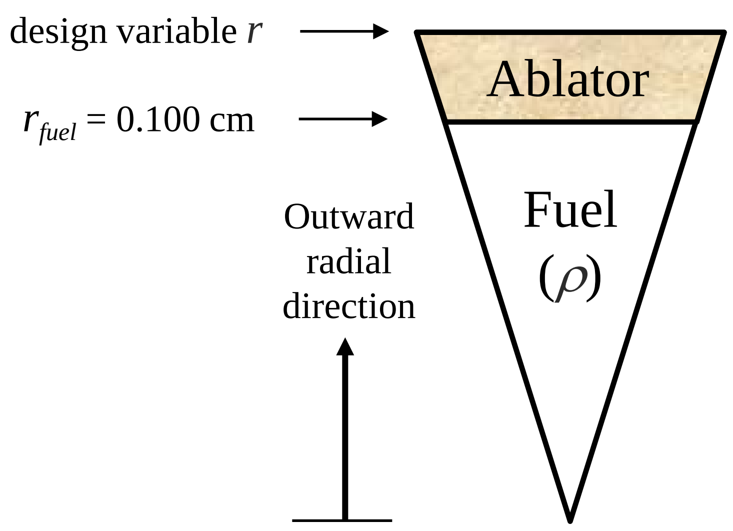


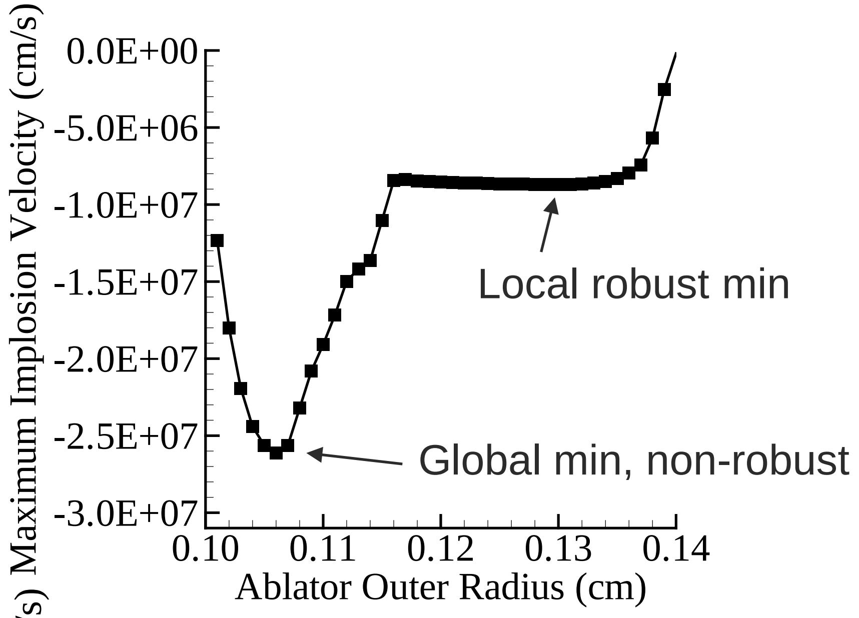
Sample Hohlraum Configuration

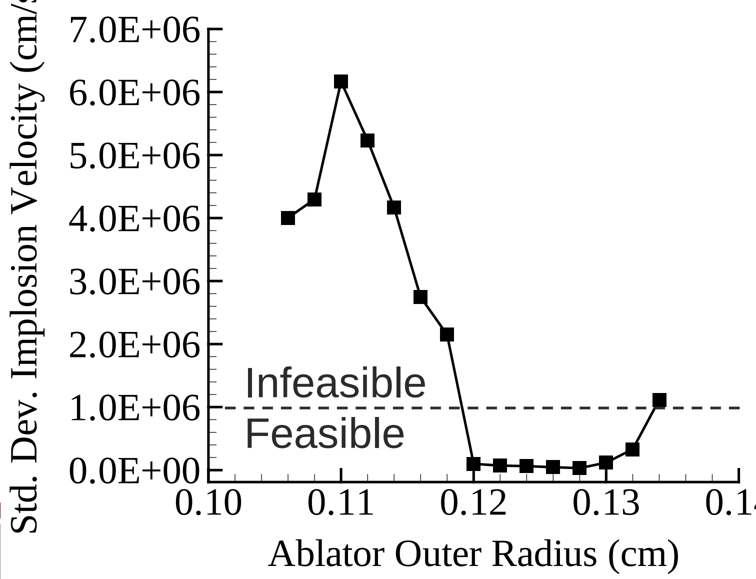
Uncertainties in plasma, drive, and capsule characteristics

ICF Capsule Robust Design

Design goal: maximize the implosion velocity w.r.t. ablator radius *r* and fuel density ρ, but remain robust w.r.t. manufacturing variability





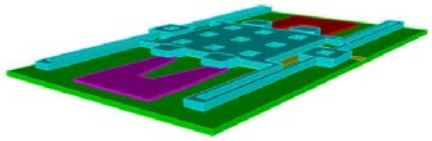


Minimize $V(r, \rho)$

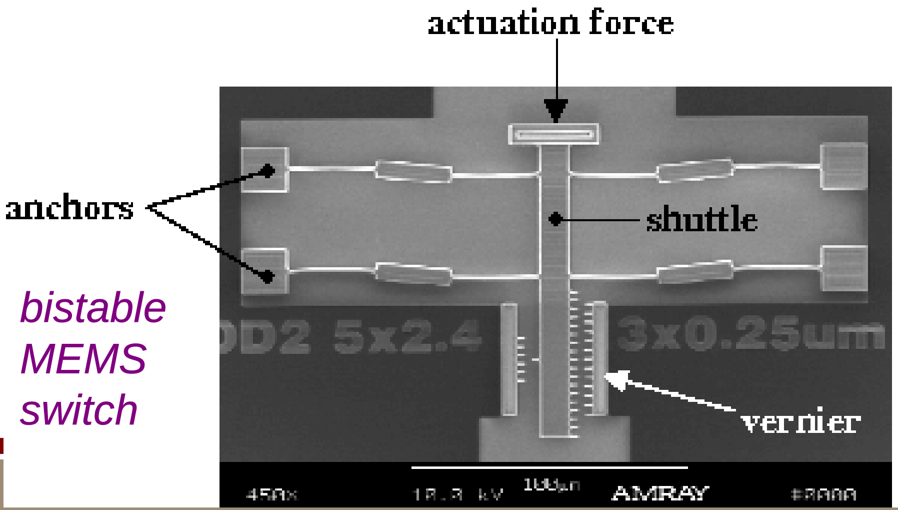
Subject to $\sigma_V(r, \rho) \leq$ target value

uniform: +/- 2.5% range in r, ρ

Shape Optimization of Compliant MEMS

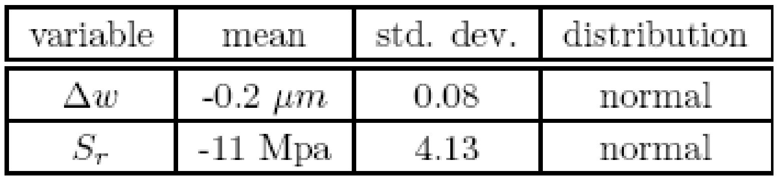


- Micro-electromechanical system (MEMS): typically made from silicon, polymers, or metals; used as micro-scale sensors, actuators, switches, and machines
- MEMS designs are subject to substantial variability and lack historical knowledge base. Materials and micromachining, photo lithography, etching processes all yield uncertainty.
- Resulting part yields can be low or have poor cycle durability
- Goal: shape optimize finite element model of bistable switch to…
  - Achieve prescribed reliability in actuation force
  - Minimize sensitivity to uncertainties (robustness)



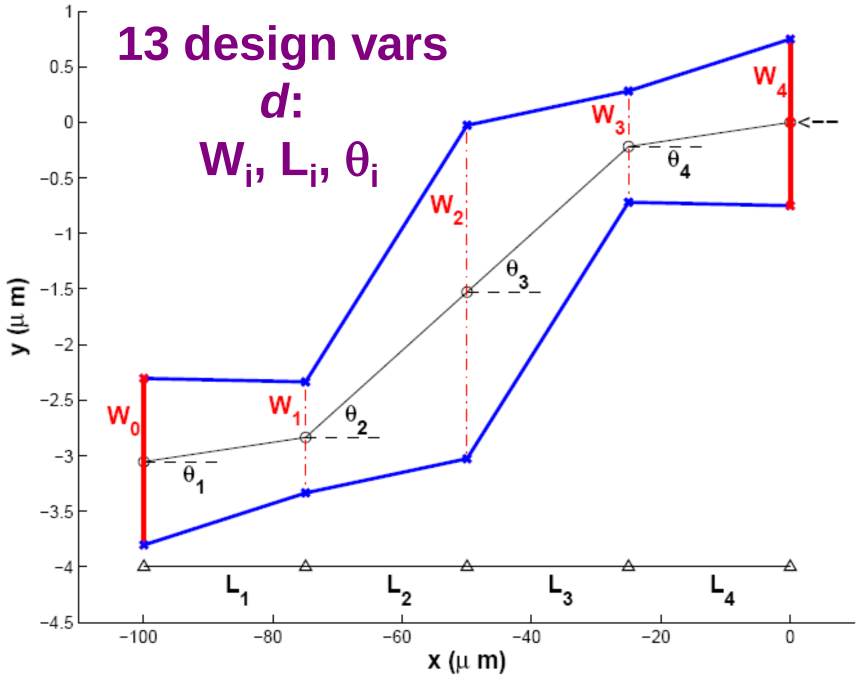
bistable MEMS switch

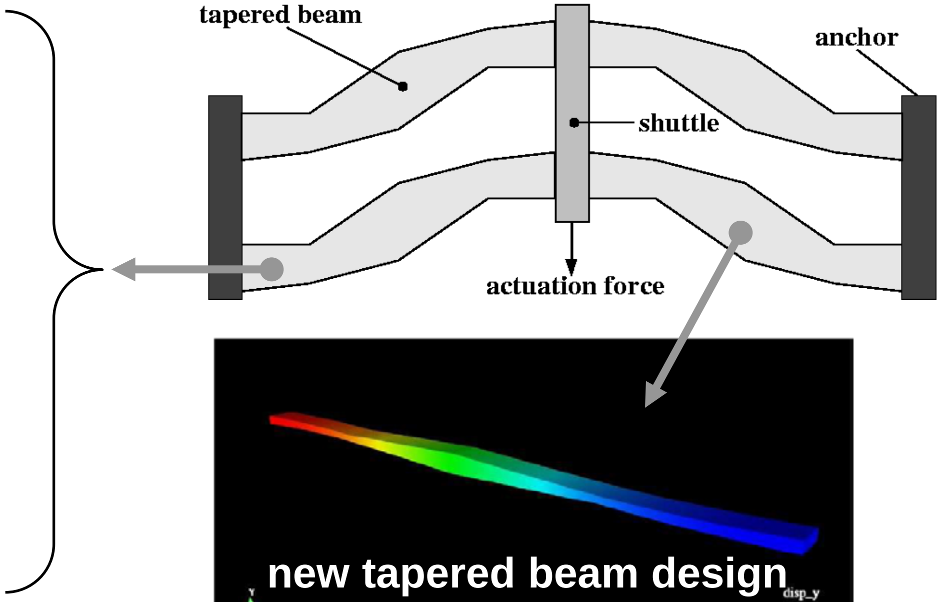
uncertainties to be considered (edge bias and residual stress)

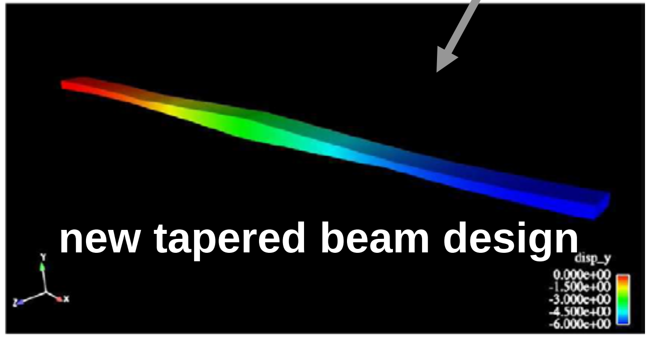


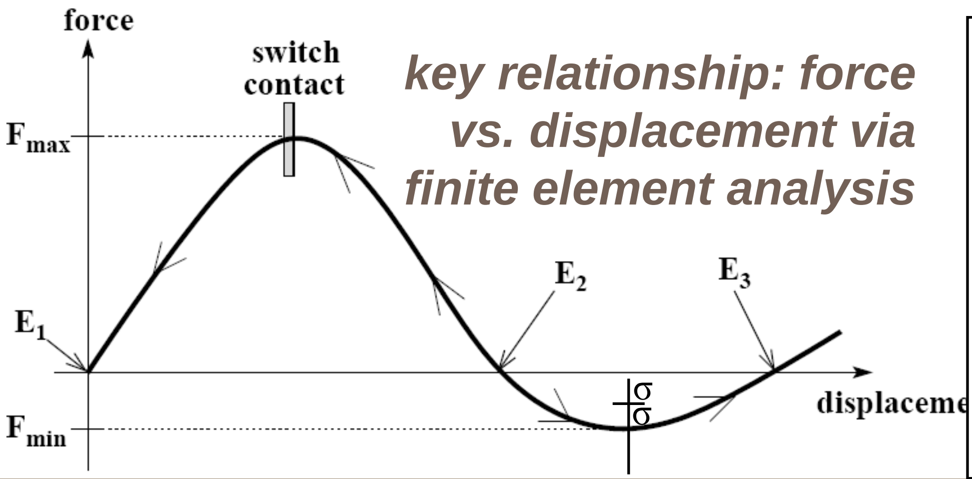
variable	mean	std. dev.	distribution
Δw	-0.2 μm	0.08	normal
S_r	-11 Mpa	4.13	normal

MEMS Switch Design: Geometry Optimization









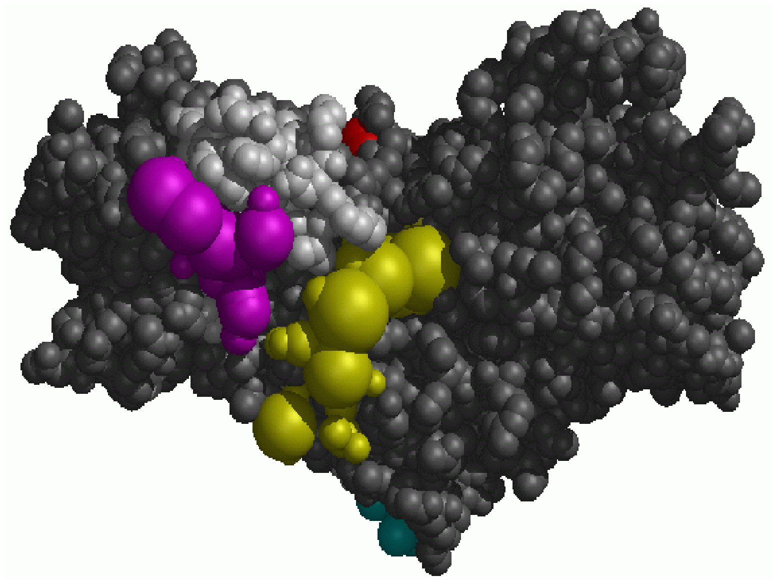
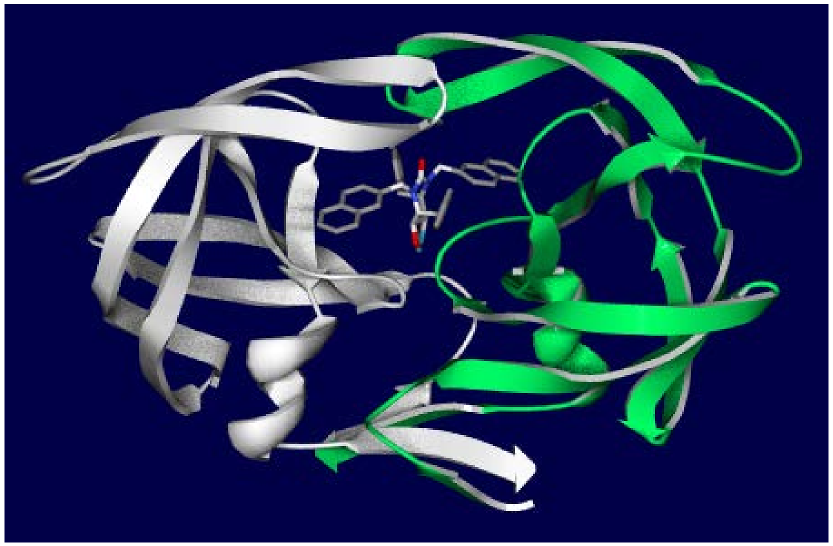
key relationship: force vs. displacement via finite element analysis

Typical design specifications:

- **actuation force F_{min} reliably 5 μN**
- **bistable ($F_{max} > 0$, $F_{min} < 0$)**
- **maximum force: $50 < F_{max} < 150$**
- **equilibrium E2 < 8 μm**
- **maximum stress < 1200 MPa**

Drug Docking (Courtesy Bill Hart)

- Problem: find the optimal binding for a small ligand in a binding site

- Application: lead compound development
- Impact: limit lab experimentation required to develop new drugs
- Approach: heuristic global optimization of flexible docking empirical potential functions