Introduction to Design Optimization

16.90

7 May, 2014

Today's Topics

1. The basics of a design optimization problem
2. Unconstrained optimization algorithms
3. Computing gradients

Design Variables

Design vector **x** contains *n* variables that form the design space

During design space exploration or optimization we change the entries of **x** in some rational fashion to achieve a desired effect

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \text{aspect ratio [-]} \\ \text{transmit power [W]} \\ \text{\# of apertures [-]} \\ \text{orbital altitude [km]} \\ \vdots \\ \text{control gain [V/V]} \end{bmatrix}$$

x_i can be …..

Real: $x_i \in \mathcal{R}$

Integer: $x_i \in \mathcal{I}$

Binary: $x_i \in \{0, 1\}$

Boolean: $x_i \in \{\text{true}, \text{false}\}$

⇨ Design variables are “controlled” by the designers

Objectives

The objective can be a vector **J** of *z* system responses or characteristics we are trying to maximize or minimize

$$\mathbf{J} = \begin{bmatrix} J_1 \\ J_2 \\ J_3 \\ J_i \\ \vdots \\ J_z \end{bmatrix} = \begin{bmatrix} \text{cost} \quad [\$] \\ \text{range} \quad [\text{km}] \\ \text{weight} \ [\text{kg}] \\ \text{data rate} \ [\text{bps}] \\ \vdots \\ \text{ROI} \quad [\%] \end{bmatrix}$$

Often the objective is a scalar function, but for real systems often we attempt multi-objective optimization:

Some objectives can be conflicting.

Parameters

Parameters **p** are quantities that affect the objective **J**, but are not degrees of freedom in the optimization search.

Quantifying the sensitivity of optimization results to parameters gives insight about assumptions and problem specifications.

Sometimes parameters **p** can be turned into design variables x_i to enlarge the design space.

Sometimes parameters **p** are former design variables that were fixed at some value because they were found not to affect any of the objectives J_i or because their optimal level was predetermined.

Constraints

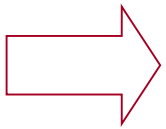
Constraints act as boundaries of the design space **x** and typically occur due to finiteness of resources or technological limitations of some design variables.

Often, but not always, optimal designs lie at the intersection of several active constraints

Inequality constraints: $g_j(\mathbf{x}) \le 0 \quad j = 1,2,\ldots,m_1$

Equality constraints: $h_k(\mathbf{x}) = 0 \quad k = 1,2,\ldots,m_2$

Bounds: $x_{i,LB} \le x_i \le x_{i,UB} \quad i = 1,2,\ldots,n$



Objectives are what we are trying to achieve

Constraints are what we cannot violate

Design variables are what we can change

Design Optimization Problem Statement

The design problem may be formulated as a problem of Nonlinear Programming (NLP)

$$\min \mathbf{J}(\mathbf{x},\mathbf{p})$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{x},\mathbf{p}) \le 0$$

$$\mathbf{h}(\mathbf{x},\mathbf{p}) = 0$$

$$x_{i,LB} \le x_i \le x_{i,UB} \quad (i = 1,\ldots,n)$$

where $\mathbf{J} = \left[J_1(\mathbf{x}) \quad \cdots \quad J_z(\mathbf{x}) \right]^T$

$\mathbf{x} = \left[x_1 \quad \cdots \quad x_i \quad \cdots \quad x_n \right]^T$

$\mathbf{g} = \left[g_1(\mathbf{x}) \cdots g_{m_1}(\mathbf{x}) \right]^T$

$\mathbf{h} = \left[h_1(\mathbf{x}) \cdots h_{m_2}(\mathbf{x}) \right]^T$

Iterative Optimization Procedures

Many optimization algorithms are iterative:

$$\mathbf{x}^q = \mathbf{x}^{q-1} + \alpha^q \mathbf{S}^q$$

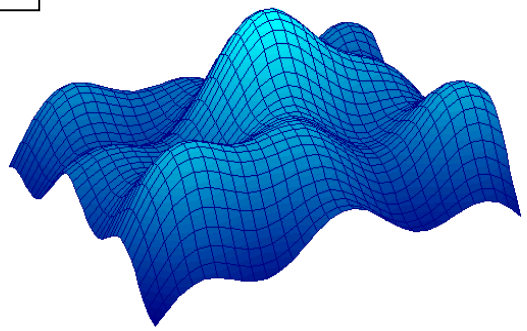
where

q=iteration number

S=vector search direction

α=scalar distance

and the initial solution $\mathbf{x}^0$ is given

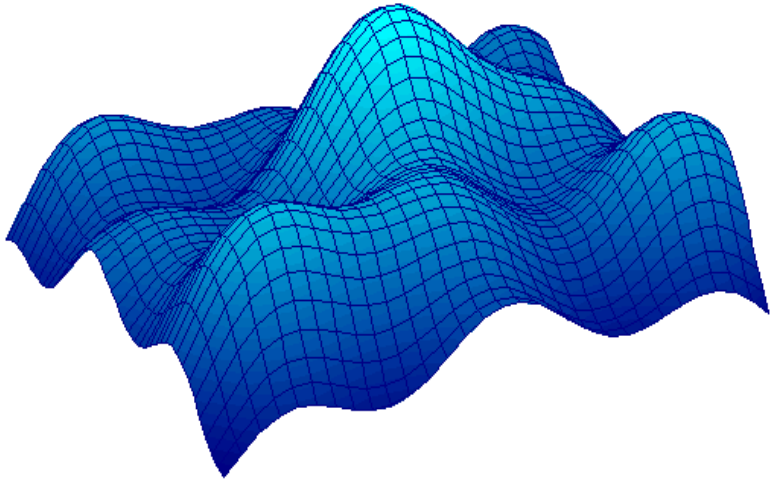


The algorithm determines the search direction **S** according to some criteria.

Gradient-based algorithms use gradient information to decide where to move. Gradient-free algorithms use sampling and/or heuristics.

Iterative Optimization Procedures

Matlab demo



Gradient Vector

Consider a function $J(\mathbf{x})$, $\mathbf{x}=[x_1,x_2,\ldots,x_n]^T$

The gradient of $J(\mathbf{x})$ at a point $\mathbf{x}^0$ is a vector of length n:

$$\nabla J(\mathbf{x}^0) = \begin{bmatrix} \frac{\partial J}{\partial x_1}(\mathbf{x}^0) \\ \frac{\partial J}{\partial x_2}(\mathbf{x}^0) \\ \vdots \\ \frac{\partial J}{\partial x_n}(\mathbf{x}^0) \end{bmatrix}$$

Each element in the vector is evaluated at the point $\mathbf{x}^0$.

Hessian Matrix

Consider a function $J(\mathbf{x})$, $\mathbf{x}=[x_1,x_2,\ldots,x_n]^T$

The second derivative of $J(\mathbf{x})$ at a point $\mathbf{x}^0$ is a matrix of size $n \times n$:

$$\mathbf{H}(\mathbf{x}^0) \equiv \nabla^2 J(\mathbf{x}^0) = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \cdots & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_n} \\ \frac{\partial^2 J}{\partial x_1 \partial x_2} & \ddots & & & \vdots \\ \vdots & & \ddots & & \\ \vdots & & & \ddots & \vdots \\ \frac{\partial^2 J}{\partial x_1 \partial x_n} & & & & \frac{\partial^2 J}{\partial x_n^2} \end{bmatrix}$$

Each element in the matrix is evaluated at the point $\mathbf{x}^0$.

Taylor Series

Consider scalar case:

$$f(z) = f(z^0) + \left.\frac{df}{dz}\right|_{z^0} (z - z^0) + \frac{1}{2}\left.\frac{d^2 f}{dz^2}\right|_{z^0} (z - z^0)^2 + \cdots$$

When function depends on a vector:

$$J(\mathbf{x}) = J(\mathbf{x}^0) + \underbrace{\left[\nabla J(\mathbf{x}^0)\right]^T}_{1\times n} \underbrace{(\mathbf{x} - \mathbf{x}^0)}_{n\times 1}$$

$$+ \frac{1}{2} \underbrace{(\mathbf{x} - \mathbf{x}^0)^T}_{1\times n} \underbrace{\mathbf{H}(\mathbf{x}^0)}_{n\times n} \underbrace{(\mathbf{x} - \mathbf{x}^0)}_{n\times 1} + \cdots$$

The gradient vector and Hessian matrix can be approximated using finite differences if they are not available analytically or using adjoints.

Types of Optimization Algorithms

- Useful resource: Prof. Steven Johnson's open-source library for nonlinear optimization http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms

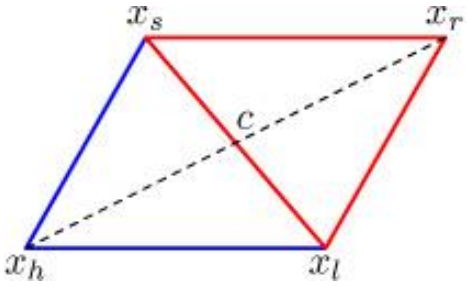
- Global optimization
- Local derivative-free optimization
- Local gradient-based optimization

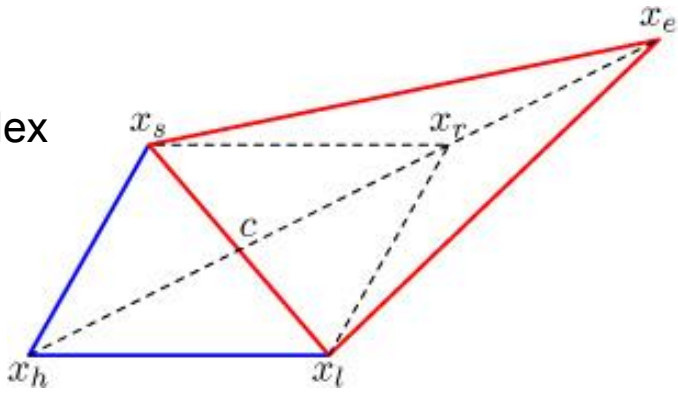
Most methods have some convergence analysis and/or proofs.

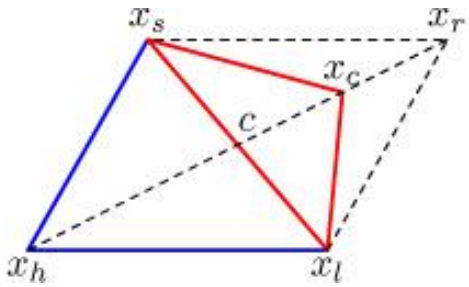
- Heuristic methods

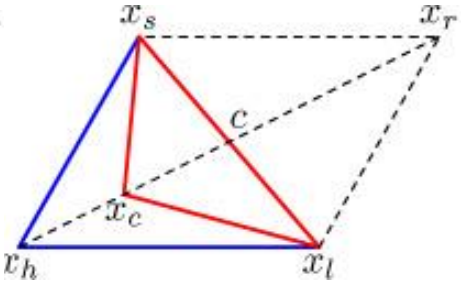
Local Derivative-Free Optimization: Nelder-Mead Simplex

- A simplex is a special polytope of $N + 1$ vertices in N dimensions
  - e.g., line segment on a line, triangle in 2D, tetrahedron in 3D
- Form an initial simplex around the initial guess $\mathbf{x}^0$
- Repeat the following general steps:
  - Compute the function value at each vertex of the simplex
  - Order the vertices according to function value, and discard the worst one
  - Generate a new point by "reflection"
  - If the new point is acceptable, generate a new simplex. Expand or contract simplex size according to quality of new point.
- Converges to a local optimum when the objective function varies smoothly and is unimodal, but can converge to a non-stationary point in some cases
- "fminsearch" in Matlab





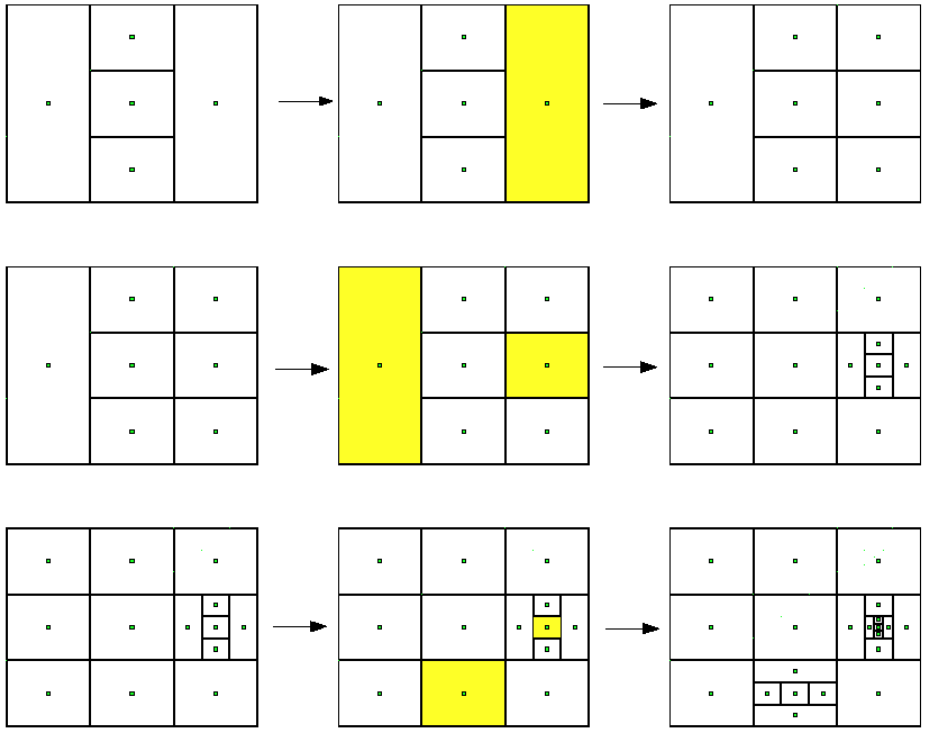




Figures from http://www.scholarpedia.org/article/Nelder-Mead_algorithm

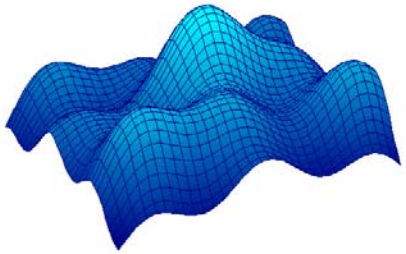
Global Derivative-Free Optimization: DIRECT

- DIRECT: DIviding RECTangles algorithm for global optimization (Jones et al., 1993)
- Initialize by dividing domain into hyper-rectangles
- Repeat
  - Identify potentially optimal hyper-rectangles
  - Divide potentially optimal hyper-rectangles
  - Sample at centers of new hyper-rectangles
- Balances local and global search
  - Global convergence to the optimum
  - May take a large, exhaustive search



Figures from "DIRECT Optimization Algorithm User Guide," D.E. Finkel, 2003.

Gradient-Based Optimization Process



Unconstrained Problems: Gradient-Based Optimization Methods

- First-Order Methods
  - use gradient information to calculate **S**
  - steepest descent method
  - conjugate gradient method
  - quasi-Newton methods
- Second-Order Methods
  - use gradients and Hessian to calculate **S**
  - Newton method

- Often, a constrained problem can be cast as an unconstrained problems and these techniques used.

Steepest Descent

$$\mathbf{S}^q = -\nabla J(\mathbf{x}^{q-1})$$

$-\nabla J(\mathbf{x})$ is the direction of max decrease of J at $\mathbf{x}$

Algorithm:

choose $\mathbf{x}^0$, set $\mathbf{x}=\mathbf{x}^0$

repeat until converged:

$\mathbf{S} = -\nabla J(\mathbf{x})$

choose α to minimize $J(\mathbf{x}+\alpha\mathbf{S})$

$\mathbf{x} = \mathbf{x} + \alpha\mathbf{S}$

- doesn't use any information from previous iterations
- converges slowly
- α is chosen with a 1-D search (interpolation or Golden section)

Conjugate Gradient

$$\mathbf{S}^1 = -\nabla J(\mathbf{x}^0)$$
$$\mathbf{S}^q = -\nabla J(\mathbf{x}^{q-1}) + \beta^q \mathbf{S}^{q-1}$$

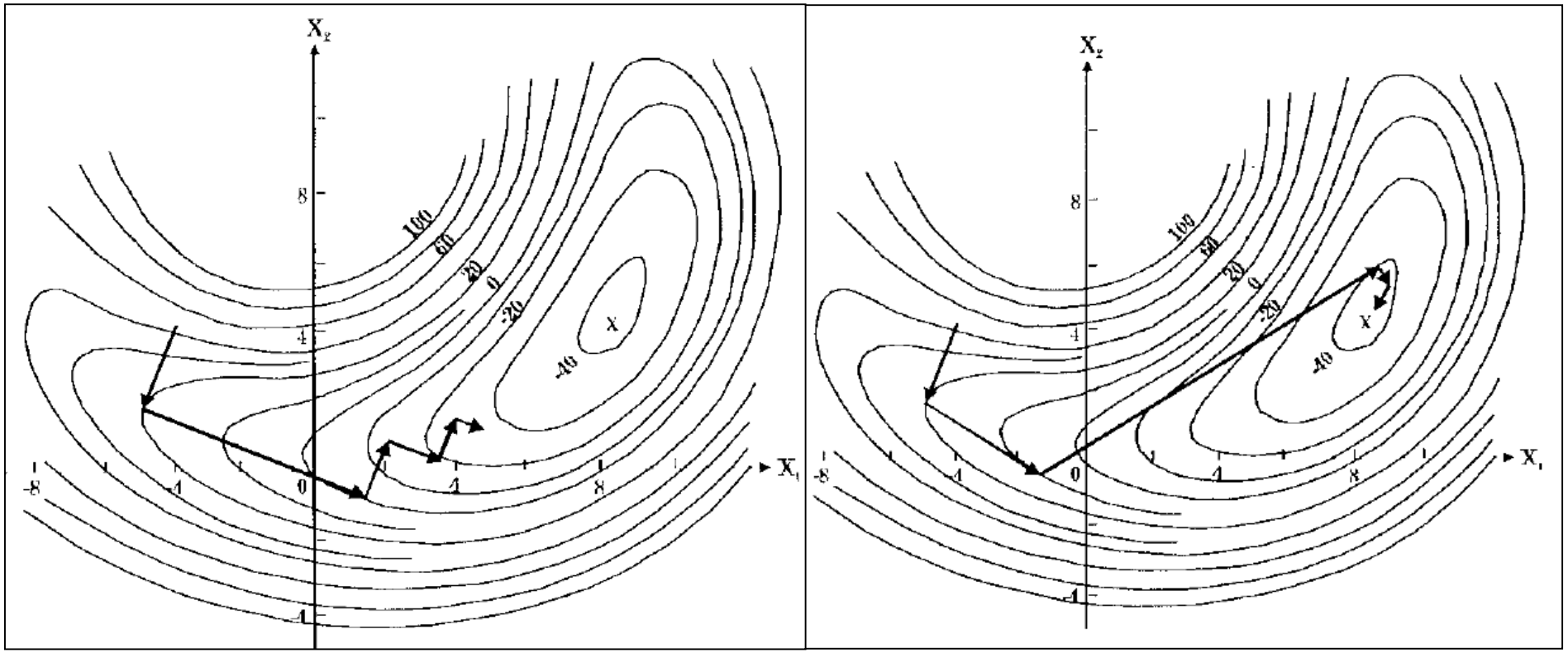
$$\beta^q = \frac{\left|\nabla J(\mathbf{x}^{q-1})\right|^2}{\left|\nabla J(\mathbf{x}^{q-2})\right|^2}$$

- search directions are now conjugate
- directions $\mathbf{S}^j$ and $\mathbf{S}^k$ are conjugate if $\mathbf{S}^{jT}\,\mathbf{H}\,\mathbf{S}^k = 0$ (also called H-orthogonal)
- makes use of information from previous iterations without having to store a matrix

Geometric Interpretation

Steepest descent

Conjugate gradient



Figures from “Optimal Design in Multidisciplinary Systems,” AIAA Professional Development Short Course Notes, September 2002.

16.90 Computational Methods in Aerospace Engineering

Spring 2014