MIT esd

16.888
ESD.77

Multidisciplinary System Design Optimization (MSDO)

Numerical Optimization I

Lecture 7

Karen Willcox

Today's Topics

- Existence & Uniqueness of an Optimum Solution
- Karush-Kuhn-Tucker Conditions
- Convex Spaces
- Unconstrained Problems

Disclaimer!

- This is not a classic optimization class ...
- The aim is not to teach you the details of optimization algorithms, but rather to expose you to different methods.
- We will utilize optimization techniques – the goal is to understand enough to be able to utilize them wisely.
- If you plan to use optimization extensively in your research, you should take an optimization class, e.g. 15.093

Learning Objectives

After the next two lectures, you should:

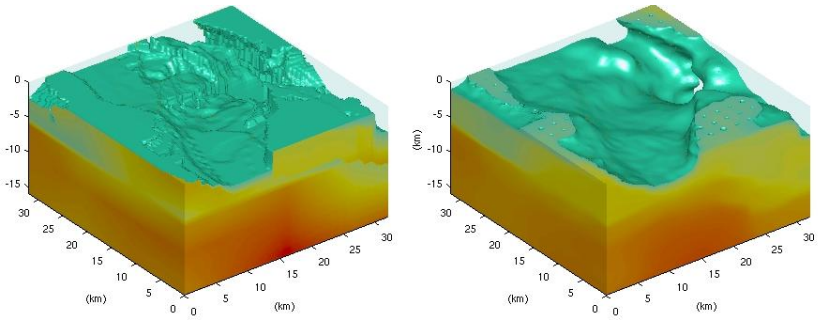
- be familiar with what gradient-based (and some gradient-free) optimization techniques are available
- understand the basics of how each technique works
- be able to choose which optimization technique is appropriate for your problem
- understand what to look for when the algorithm terminates
- understand why the algorithm might fail

How to Choose an Algorithm?

- Number of design variables
- Type of design variables (real/integer, continuous/discrete)
- Linear/nonlinear
- Continuous/discontinuous objective behavior
- Equality/inequality constraints
- Discontinuous feasible spaces
- Initial solution feasible/infeasible
- Availability of gradient information
- Simulation code (forward problem) runtime

Gradient-Based Optimization Applications

- Gradient-based methods can be used to solve large-scale, highly complex engineering problems
- Many recent advances in nonlinear optimization, e.g. in PDE-constrained optimization, exploiting structure of the problem, adjoints, preconditioning, specialized solvers, parallel computing, etc.
- We will discuss some basic methods – not state-of-the-art



Earthquake inverse modeling (CMU Quake Project, 2003): Inversion of surface observations for 17 million elastic parameters (right: target; left: inversion result). Optimization problem solved in 24 hours on 2048 processors of an HP AlphaServer system.

Accelerator shape optimization (SLAC): Next generation accelerators have complex cavities that require shape optimization for improved performance and reduced cost.

Examples from O. Ghattas (UT Austin) and collaborators.

Standard Problem Definition

$$\min \; J(\mathbf{x})$$

$$\text{s.t.} \; g_j(\mathbf{x}) \le 0 \quad j = 1,..,m_1$$

$$h_k(\mathbf{x}) = 0 \quad k = 1,..,m_2$$

$$x_i^{\ell} \le x_i \le x_i^{u} \quad i = 1,..,n$$

- For now, we consider a single objective function, $J(\mathbf{x})$.
- There are n design variables, and a total of m constraints ($m=m_1+m_2$).
- The bounds are known as **side constraints**.

Linear vs. Nonlinear

The objective function is a linear function of the design variables if each design variable appears only to the first power with constant coefficients multiplying it.

$$J(\mathbf{x}) = x_1 + 2x_2 - 3.4x_3$$ is linear in $\mathbf{x}=[x_1\ x_2\ x_3]^T$

$$J(\mathbf{x}) = x_1x_2 + 2x_2 - 3.4x_3$$ is nonlinear in $\mathbf{x}$

$$J(\mathbf{x}) = \cos(x_1) + 2x_2 - 3.4x_3$$ is nonlinear in $\mathbf{x}$

Linear vs. Nonlinear

A constraint is a linear function of the design variables if each design variable appears only to the first power with constant coefficients multiplying it.

$$6x_1 - x_2 - 2x_3 < 10$$ is linear in **x**

$$6x_1 - x_2 - 2x_3^2 < 10$$ is nonlinear in **x**

$$6x_1 - \sin(x_2) - 2x_3 < 10$$ is nonlinear in **x**

Iterative Optimization Procedures

Many optimization algorithms are iterative:

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

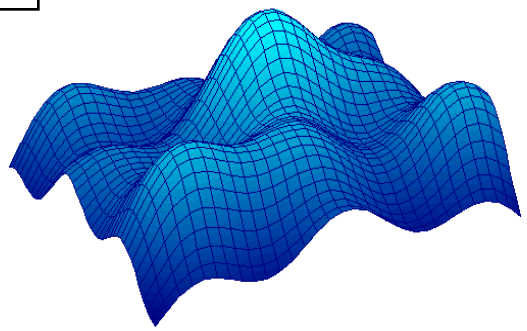
where

q=iteration number

S=vector search direction

α=scalar distance

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

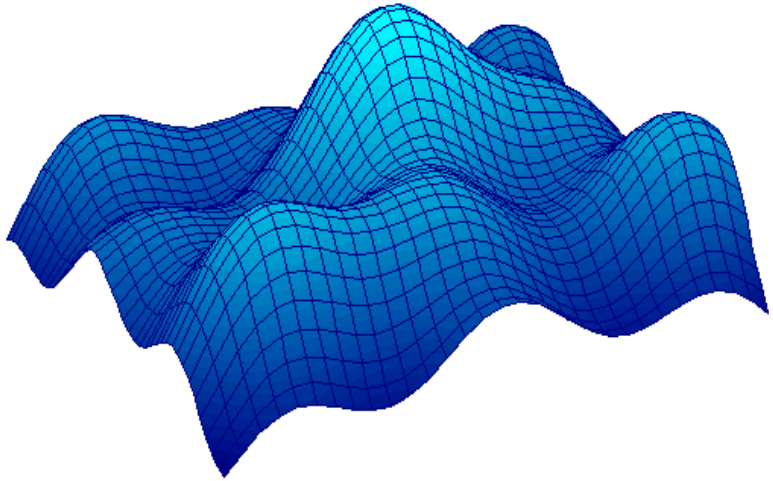


The algorithm determines the search direction **S** according to some criteria.

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 $J(\mathbf{x})$, $\mathbf{x}=[x_1,x_2,\ldots,x_n]$

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 $J(\mathbf{x})$, $\mathbf{x}=[x_1,x_2,\ldots,x_n]$

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 $\mathbf{x}^0$.

Gradients & Hessian Example

$$J(\mathbf{x}) = 3x_1 + x_1x_2 + x_3^2 + 6x_2^3x_3$$

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 (L8).

Existence & Uniqueness of an Optimum Solution

- Usually cannot guarantee that global optimum is found
  - multiple solutions may exist
  - numerical ill-conditioning

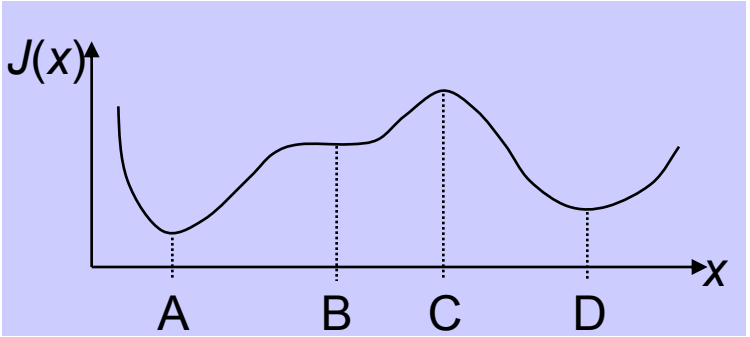
    →start from several initial solutions
- Can determine mathematically if have relative minimum
- Under certain conditions can guarantee global optimum (special class of optimization problems or with global optimization methods)
- It is very important to interrogate the “optimum” solution

Existence & Uniqueness: Unconstrained Problems

- Unconstrained problems: at minimum, gradient must vanish

$$||\nabla J(\mathbf{x}^*)|| = 0$$

  - $\mathbf{x}^*$ is a stationary point of J
  - necessary but not sufficient
  - here A,B,C,D all have ∇J=0



- Calculus: at minimum, second derivative > 0

- Vector case: at minimum, $\mathbf{H}(\mathbf{x}^*)$ >0 (positive definite)

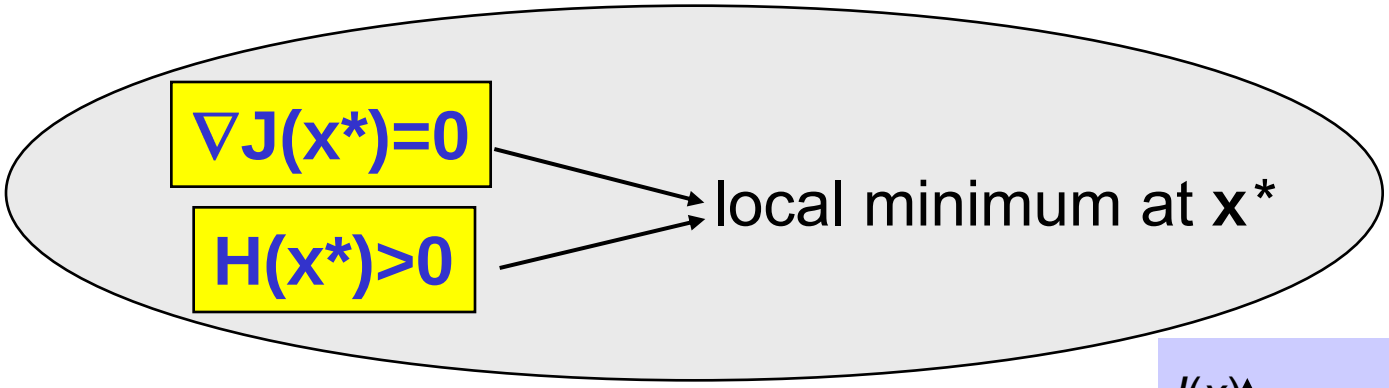
SPD Matrices

- Positive definiteness: $\mathbf{y}^T\mathbf{H}\mathbf{y} > 0$ for all $\mathbf{y} \neq 0$
- Consider the i^{th} eigenmode of **H**: $\mathbf{H}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
- If **H** is a symmetric matrix, then the eigenvectors of **H** form an orthogonal set: $\mathbf{v}_i^T\mathbf{v}_j = \delta_{ij}$
- Any vector **y** can be thought of as a linear combination of eigenvectors: $\mathbf{y} = \sum a_i \mathbf{v}_i$
- Then: $$\mathbf{y}^T\mathbf{H}\mathbf{y} = \sum_i a_i \mathbf{v}_i^T \mathbf{H} \sum_j a_j \mathbf{v}_j = \sum_{i,j} a_i \mathbf{v}_i^T \lambda_j a_j \mathbf{v}_j = \sum_i a_i^2 \lambda_i$$
- Therefore, if the eigenvalues of **H** are all positive, **H** is SPD.

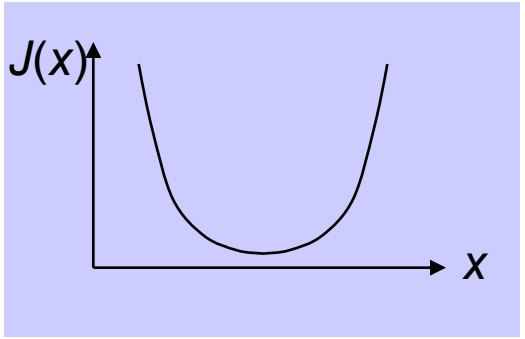
Existence & Uniqueness: Unconstrained Problems

Necessary and sufficient conditions for a minimum (unconstrained problem):

1. Gradient must vanish
2. Hessian must be positive definite



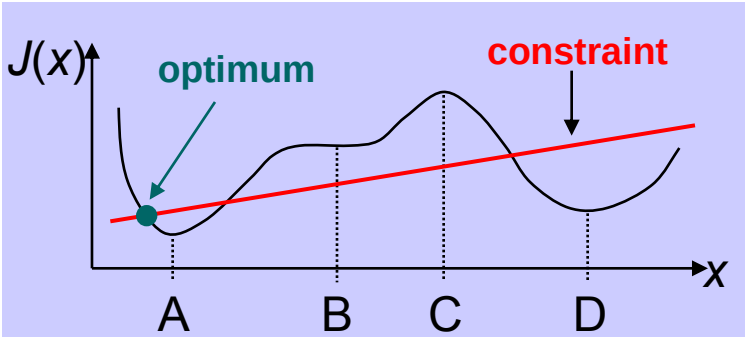
The minimum is only guaranteed to be a **global** optimum if **H**(**x**)>0 for all values of **x** (*e.g.* simple parabola).



Existence & Uniqueness: Constrained Problems

At optimum:

- at least one constraint on design is active
- ∇J does not have to be zero



In order to improve design:

- move in direction that decreases objective
- move in direction that does not violate constraints

Usable direction = any direction that reduces objective

$$\mathbf{S}^T \nabla J(\mathbf{x}) \leq 0$$

Feasible direction = a direction in which a small move will not violate constraints

$$\mathbf{S}^T \nabla g_i(\mathbf{x}) \leq 0 \qquad \text{(for all active constraints } i\text{)}$$

Note that these conditions may be relaxed for certain algorithms.

Lagrangian Functions

- A constrained minimization can be written as an unconstrained minimization by defining the Lagrangian function:

$$L(\mathbf{x},\lambda) = J(\mathbf{x}) + \sum_{j=1}^{m_1} \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^{m_2} \lambda_{m_1+k} h_k(\mathbf{x})$$

- $L(\mathbf{x},\lambda)$ is called the **Lagrangian function**.
- λ_i is the j^{th} **Lagrange multiplier**
- It can be shown that a stationary point $\mathbf{x}^*$ of $L(\mathbf{x},\lambda)$ is a stationary point of the original constrained minimization problem.

Lagrangian Example

$$\min J(\mathbf{x}) = x_1^2 + 3x_2^2$$

$$\text{s.t.} \quad x_1 + x_2 = 1$$

Karush-Kuhn-Tucker (KKT) Conditions

If **x*** is optimum, these conditions are satisfied:

1. **x*** is feasible
2. $\lambda_j \, g_j(\mathbf{x}^*) = 0, \;\; j = 1,..,m_1$ and $\lambda_j \geq 0$
3. $$\nabla J(x^*) + \sum_{j=1}^{m_1} \lambda_j \nabla g_j(x^*) + \sum_{k=1}^{m_2} \lambda_{m_1+k} \nabla h_k(x^*) = 0$$

$$\lambda_j \geq 0$$

λ_{m_1+k} unrestricted in sign

The Kuhn-Tucker conditions are necessary and sufficient if the design space is convex.

KKT Conditions: Interpretation

Condition 1: the optimal design satisfies the constraints

Condition 2: if a constraint is not precisely satisfied, then the corresponding Lagrange multiplier is zero

- *the j^{th} Lagrange multiplier represents the sensitivity of the objective function to the j^{th} constraint*
- *can be thought of as representing the "tightness" of the constraint*
- *if λ_j is large, then constraint j is important for this solution*

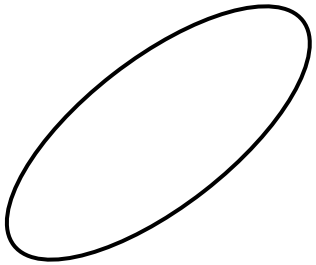
Condition 3: the gradient of the Lagrangian vanishes at the optimum

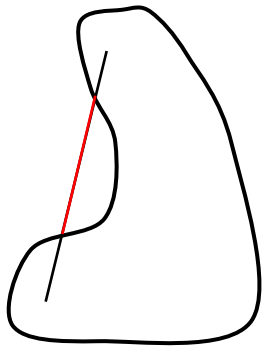
Convex Sets

Consider a set, and imagine drawing a line connecting any two points in the set.

If every point along that line is inside the set, then the set is **convex**.

If **any** point along that line is outside the set, then the set is **non-convex**.



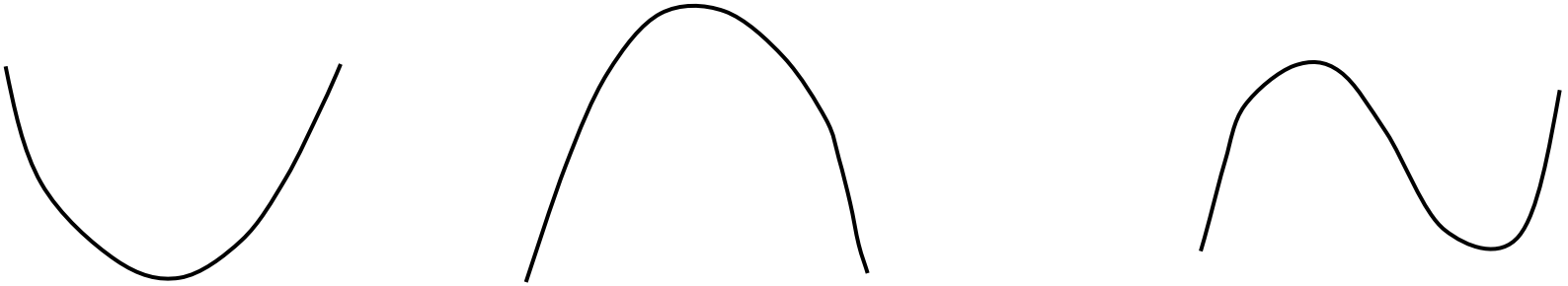


The line connecting points x^1 and x^2 is given by

$$w = \theta x^1 + (1-\theta)x^2, \qquad 0 \leq \theta \leq 1$$

Convex Functions

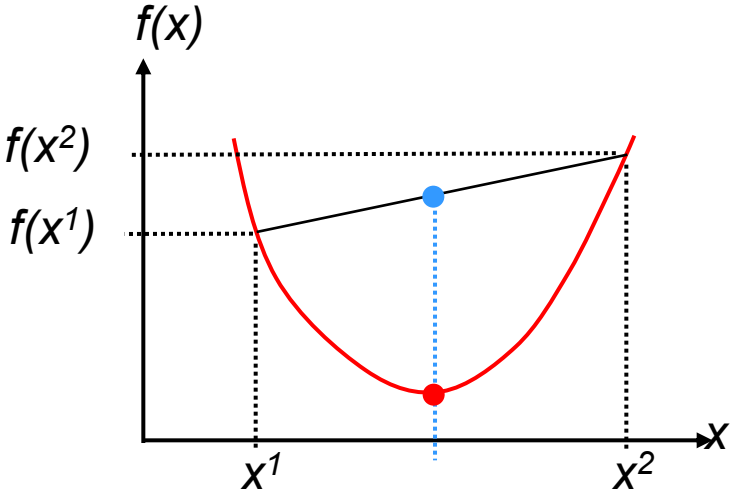
Informal definition: a convex function is one that will hold water, while a concave function will not hold water...



convex concave neither

A function $f(x)$ bounding a convex set is convex if:

$$f[\theta \mathbf{x}^1 + (1-\theta)\mathbf{x}^2] \leq \theta f(\mathbf{x}^1) + (1-\theta) f(\mathbf{x}^2)$$



Convex Spaces

Pick any two points in the feasible region. If all points on the line connecting these points lie in the feasible region, then the constraint surfaces are convex.

If the objective function is convex, then it has only one optimum (the global one) and the Hessian matrix is positive definite for **all possible** designs.

If the objective function and all constraint surfaces are convex, then the design space is convex, and the KKT conditions are sufficient to guarantee that $\mathbf{x}^*$ is a global optimum.

In general, for engineering problems, the design space is not convex ...

Convergence Criteria

Algorithm has converged when ...

no change in the objective function is obtained

OR the maximum number of iterations is reached

Once the “optimal” solution has been obtained, the KKT conditions should be checked.

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

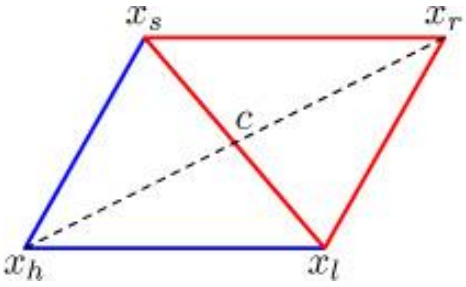
- Global optimization
- Local derivative-free optimization
- Local gradient-based optimization

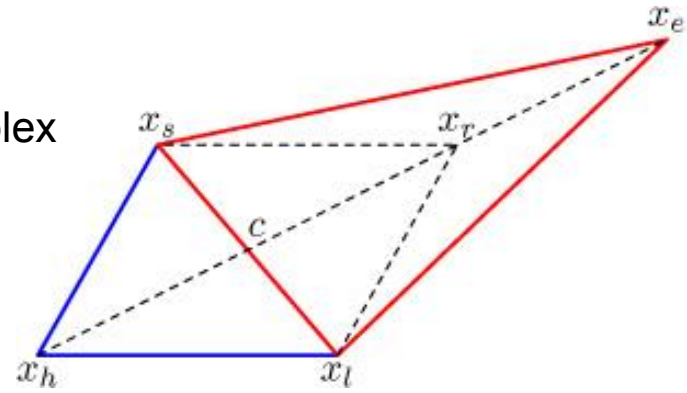
Most methods have some convergence analysis and/or proofs.

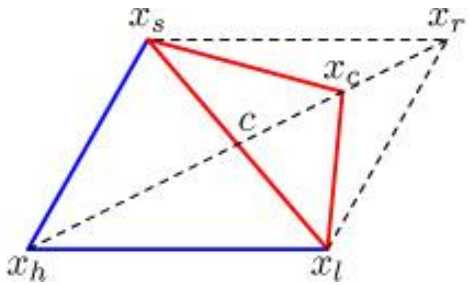
- Heuristic methods

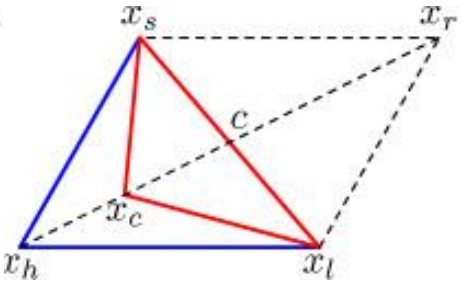
Local Derivative-Free Optimization: Nelder-Mead Simplex

- 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

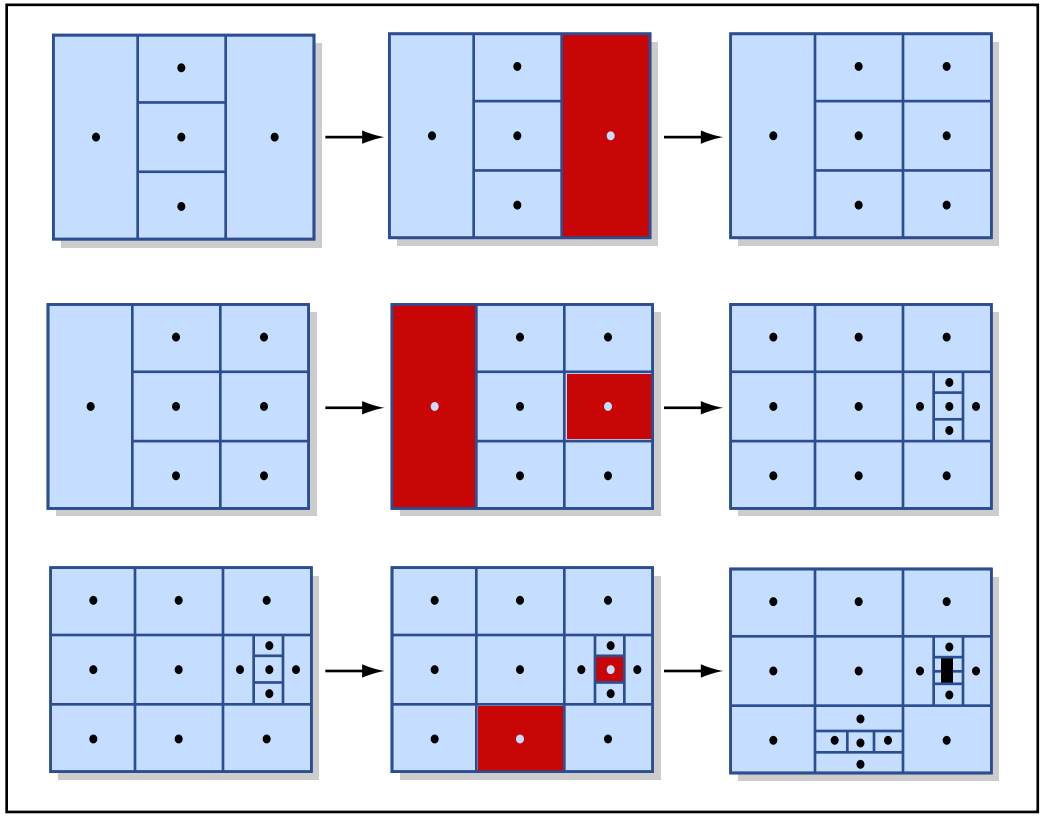
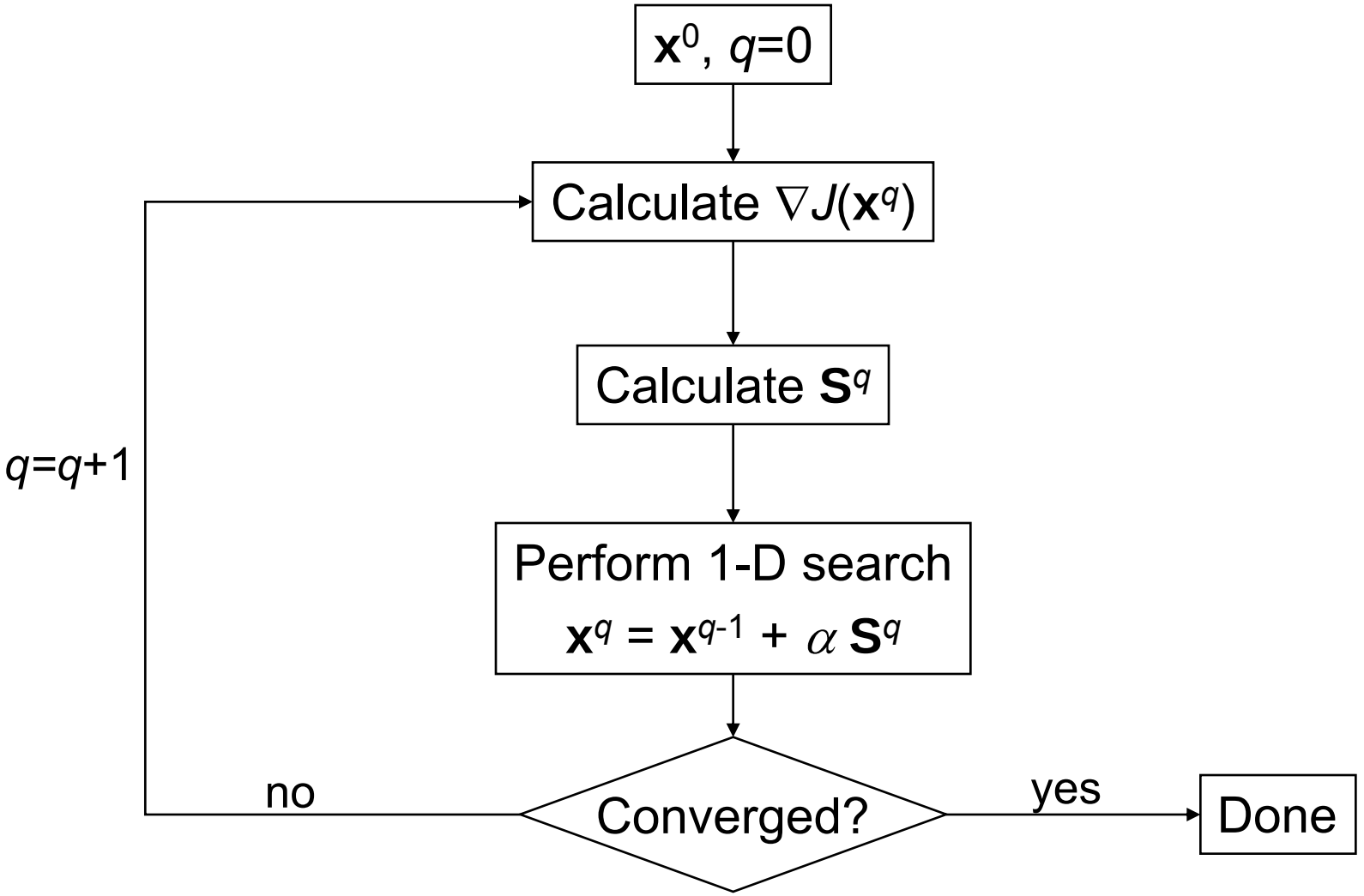


Image by MIT OpenCourseWare.

Gradient-Based Optimization Process



Unconstrained Problems: Gradient-Based Solution 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
- Methods to calculate gradients in Lecture 8.
- Often, a constrained problem can be cast as an unconstrained problems and these techniques used.

Steepest Descent

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

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

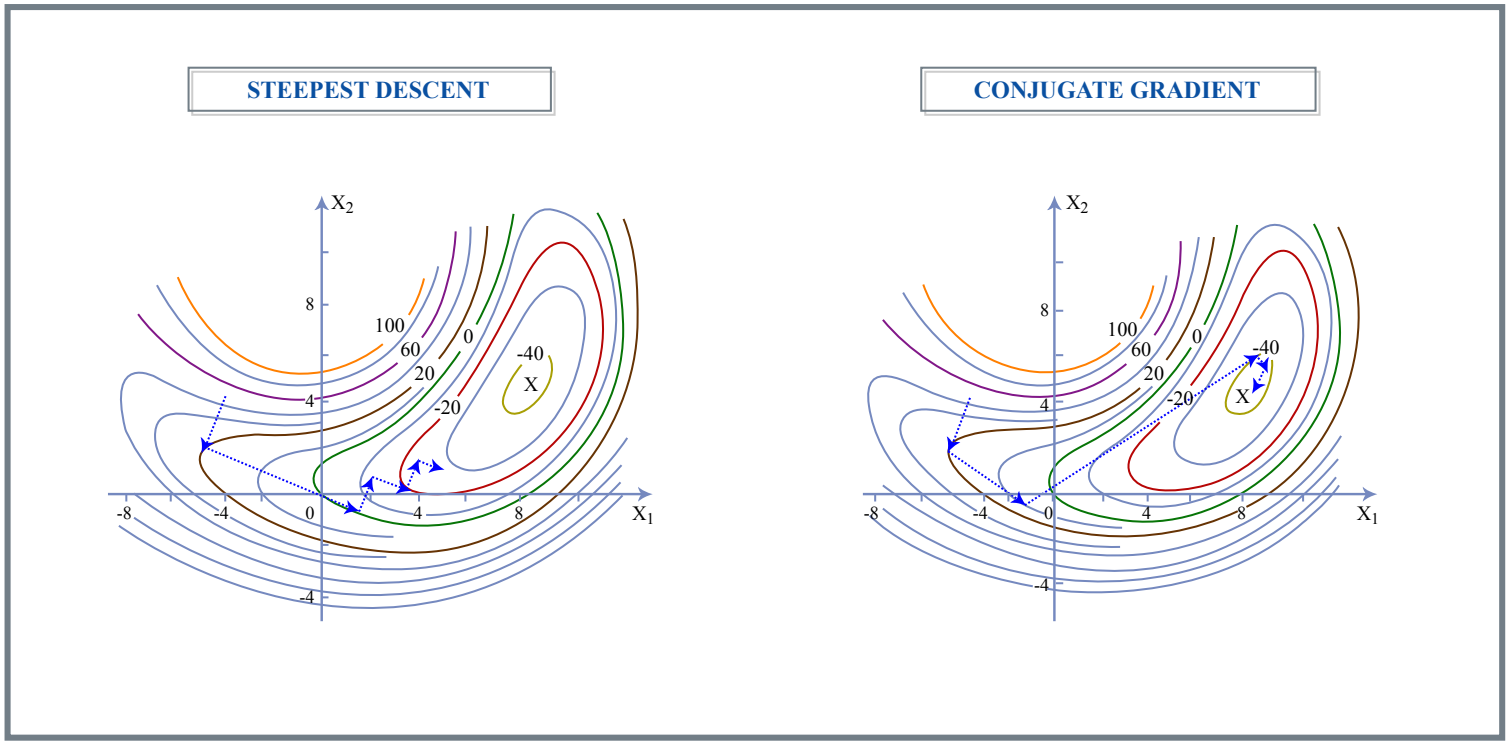
Conjugate Gradient

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

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

- 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



Adapted from: "Optimal Design in Multidisciplinary System." *AIAA Professional Development Short Course Notes*. September 2002.

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

Newton's Method

Taylor series:

$$J(\mathbf{x}) \approx J(\mathbf{x}^0) + \nabla J(\mathbf{x}^0)^T \delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^T \mathbf{H}(\mathbf{x}^0)\delta\mathbf{x}$$

where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^0$

differentiate: $\nabla J(\mathbf{x}) \approx \nabla J(\mathbf{x}^0) + \mathbf{H}(\mathbf{x}^0)\delta\mathbf{x}$

at optimum $\nabla J(\mathbf{x}^*)=0$

$$\Rightarrow \nabla J(\mathbf{x}^0) + \mathbf{H}(\mathbf{x}^0)\delta\mathbf{x} = 0$$

$$\boxed{\delta\mathbf{x} = -\left[\mathbf{H}(\mathbf{x}^0)\right]^{-1} \nabla J(\mathbf{x}^0)}$$

Newton's Method

$$\mathbf{S} = -\left[\mathbf{H}(\mathbf{x}^0)\right]^{-1} \nabla J(\mathbf{x}^0)$$

- if $J(\mathbf{x})$ is quadratic, method gives exact solution in one iteration
- if $J(\mathbf{x})$ not quadratic, perform Taylor series about new point and repeat until converged
- a very efficient technique if started near the solution
- **H** is not usually available analytically, and finite difference is too expensive ($n \times n$ matrix)
- **H** can be singular if J is linear in a design variable

Quasi Newton

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

- Also known as variable metric methods
- Objective and gradient information is used to create an approximation to the inverse of the Hessian
- **A** approaches $\mathbf{H}^{-1}$ during optimization of quadratic functions
- Convergence is similar to second-order methods (strictly 1st order)

- Initially: $\boldsymbol{A}=\mathbf{I}$, so $\mathbf{S}^1$ is steepest descent direction

  then: $\mathbf{A}^{q+1} = \mathbf{A}^q + \boldsymbol{D}^q$

  where **D** is a symmetric update matrix

  $$\mathbf{D}^q = \text{fn}(\mathbf{x}^q - \mathbf{x}^{q-1}, \nabla J(\mathbf{x}^q) - \nabla J(\mathbf{x}^{q-1}), \mathbf{A}^q)$$

- Various methods to determine **D**

  e.g. Davidon-Fletcher-Powell (DFP)

  Broydon-Fletcher-Goldfarb-Shanno (BFGS)

One-Dimensional Search (Choosing α)

- Polynomial interpolation
  - pick several values for α
  - fit polynomials to $J(\alpha)$
  - efficient, but need to be careful with implementation
- Golden section search
  - easy to implement, but inefficient
- The one-dimensional search is one of the more challenging aspects of implementing a gradient-based optimization algorithm

Polynomial Interpolation Example

$$\nabla J = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix} \quad \mathbf{s} = \begin{Bmatrix} -1 \\ -2 \end{Bmatrix} \qquad J(\alpha) = c_1 + c_2\alpha + c_3\alpha^2 + c_4\alpha^3$$

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial x_1}\frac{\partial x_1}{\partial \alpha} + \frac{\partial J}{\partial x_2}\frac{\partial x_2}{\partial \alpha} = \nabla J^T \mathbf{s}$$

α	J	dJ/dα
0	10	-5
1	6	-5
2	8	-5

Lecture Summary

- Gradient vector and Hessian matrix
- Existence and uniqueness
- Optimality conditions
- Convex spaces
- Unconstrained Methods

The next lecture will focus on gradient-based techniques for nonlinear constrained optimization. We will consider SQP and penalty methods. These are the methods most commonly used for engineering applications.

ESD.77 / 16.888 Multidisciplinary System Design Optimization

Spring 2010