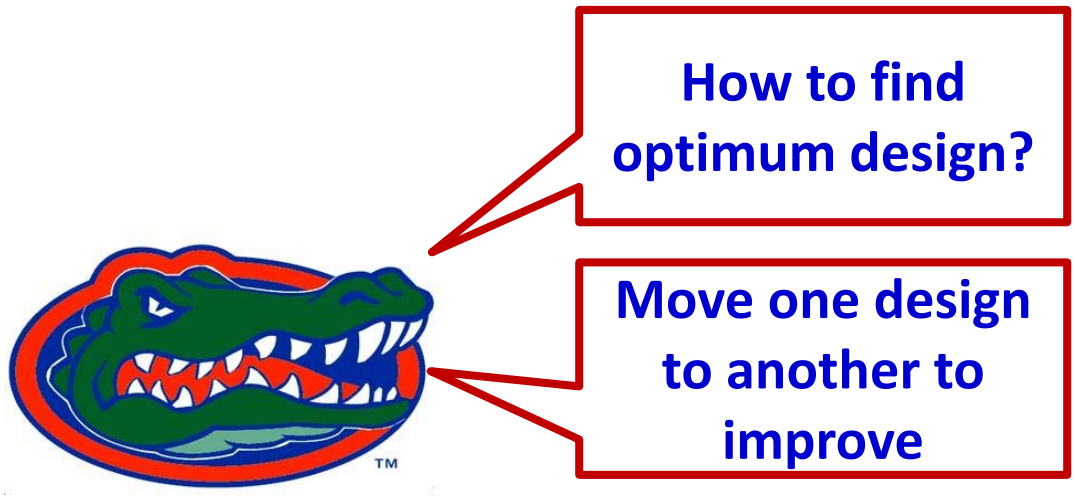
UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

Numerical Optimization

Gradient-based unconstrained optimization



NUMERICAL OPTIMIZATION ALGORITHMS

- Optimality criteria are only applicable for limited cases
  - When the expressions of obj and constraints are known
  - Compliance objective in topology optimization
- Most other cases, numerical methods are used to find optimum design
- Some numerical algorithms start from a design and find a new design that can improve the objective function
- Other algorithms populate designs until they cannot find a design that can improve the objective function

NUMERICAL METHODS

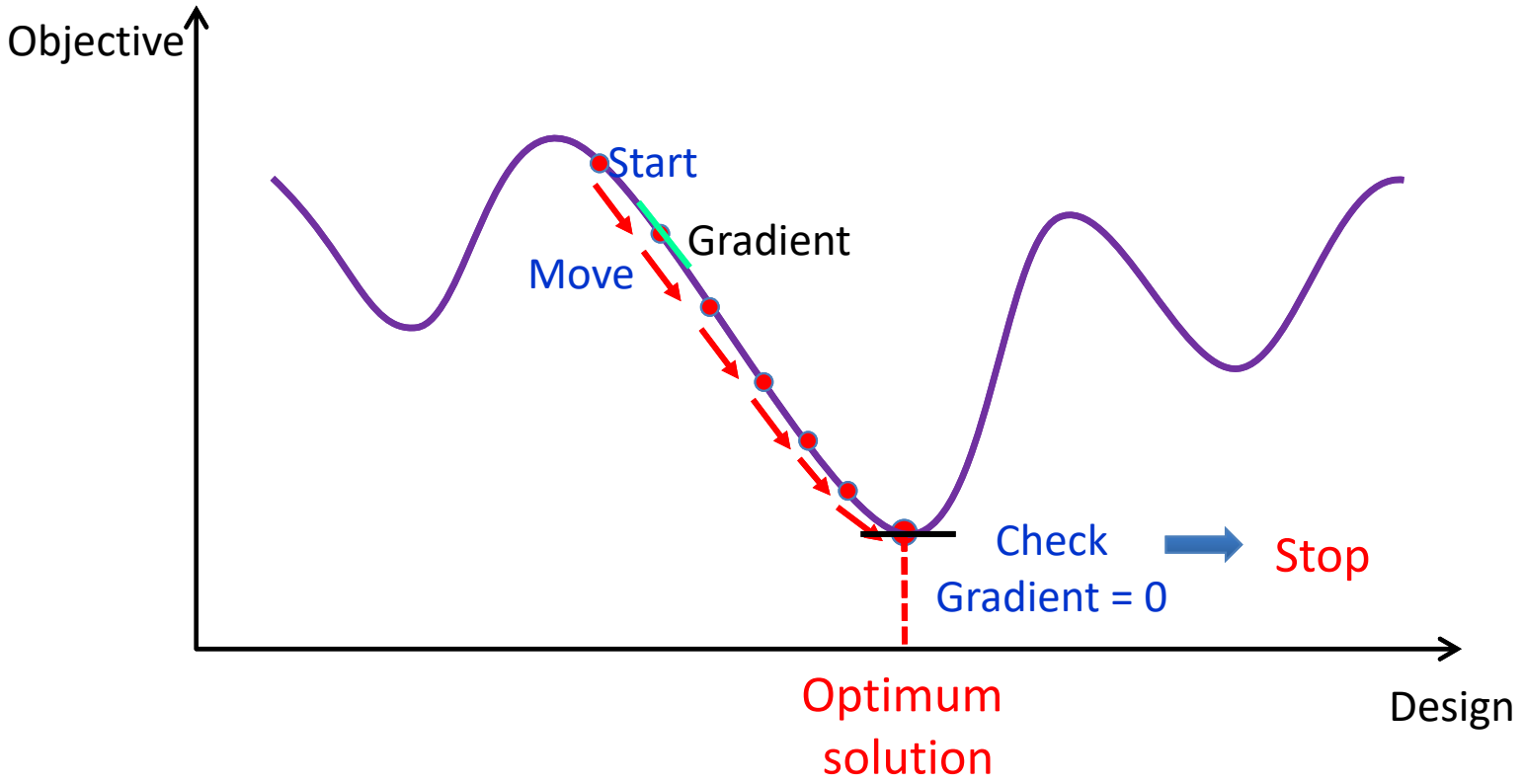
- In FEA, we don't know the explicit relationship, $f(\mathbf{b})$, between design parameters and objective (or constraint) function.
- But, we can calculate $f(\mathbf{b})$ for a given $\mathbf{b}$ by solving the finite element equation.
- Then, how can we find the optimum design?

Use Numerical Methods

- Numerical Method
  - From the current design, find the next design that can reduce the objective function and satisfy constraints
  - Repeat the search until there is no way to improve the objective function further

GRADIENT-BASED METHODS

- We do not know the function before optimization
- We can only evaluate the function and gradient at a given design



- Basic Algorithm

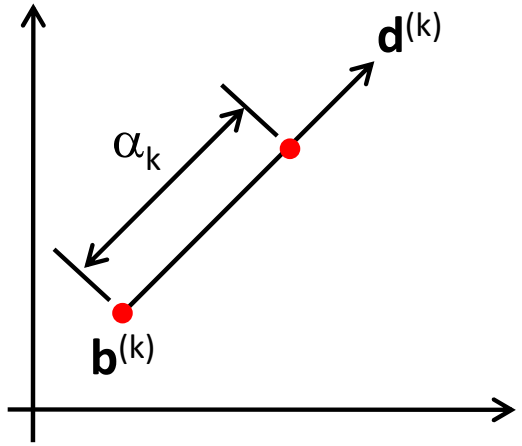
1. Start with $\mathbf{b}^{(k)}$ and $k = 0$ — Initial design must be given
2. Evaluate function values and their gradients — Using FEA
3. Using information from Step 2, determine $\Delta\mathbf{b}^{(k)}$ — Design change
4. Check for termination — Stop if converged
5. Update design

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \Delta\mathbf{b}^{(k)}$$

6. Increase $k = k + 1$ and go to Step 2 — Design iteration (cycle)

- Change in Design

$$\Delta \mathbf{b}^{(k)} = \alpha_K \, \mathbf{d}^{(k)}$$



  - α_k: Step size
  - $\mathbf{d}^{(k)}$: Search direction vector
  - Design change means the determination of search direction and step size.

UNCONSTRAINED PROBLEM

- Minimize $f(\mathbf{b})$
- Requirement: $f(\mathbf{b}^{(k+1)}) < f(\mathbf{b}^{(k)})$

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \Delta\mathbf{b}^{(k)}$$

$$f_k = f(\mathbf{b}^{(k)})$$

$$\mathbf{c}^{(k)} = \nabla f(\mathbf{b}^{(k)})$$

Objective gradient

- Taylor series expansion:

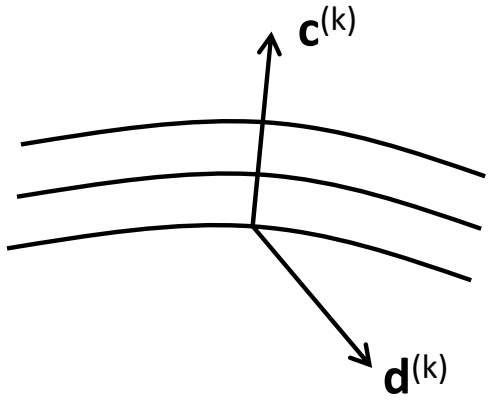
$$f(\mathbf{b}^{(k+1)}) = f(\mathbf{b}^{(k)}) + \nabla f(\mathbf{b}^{(k)}) \cdot \Delta\mathbf{b}^{(k)} + \text{H.O.T.}$$

$$\cong f_k + \mathbf{c}^{(k)} \cdot \Delta\mathbf{b}^{(k)} < f_k$$

- Descent direction:

$$\mathbf{c}^{(k)} \cdot \Delta\mathbf{b}^{(k)} < 0$$

$$\alpha_k \mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)} < 0$$



$$\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)} < 0$$

Descent direction

Search direction must satisfy this condition

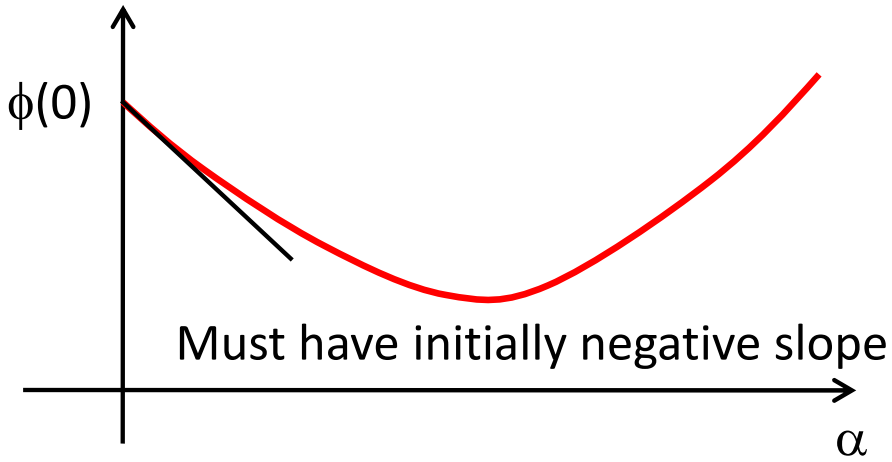
DETERMINATION OF STEP SIZE

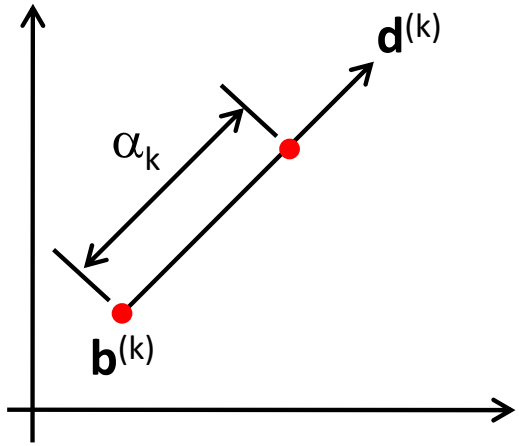
- For the moment, let's assume that the descent direction $\mathbf{d}^{(k)}$ is given
- Step size calculation : find α_k that minimizes the objective function in the direction of $\mathbf{d}^{(k)}$

$$\text{Minimize} \quad f(\mathbf{b}^{(k)} + \alpha_K \mathbf{d}^{(k)})$$

- Single variable (α_k) problem: 1D line search

$$\phi(\alpha_k) = f(\mathbf{b}^{(k)} + \alpha_k \mathbf{d}^{(k)})$$





DETERMINATION OF STEP SIZE *cont.*

- Step size termination criterion

$$\phi(0) = f_K$$

- At minimum $\phi' = 0, \quad \phi'' > 0$

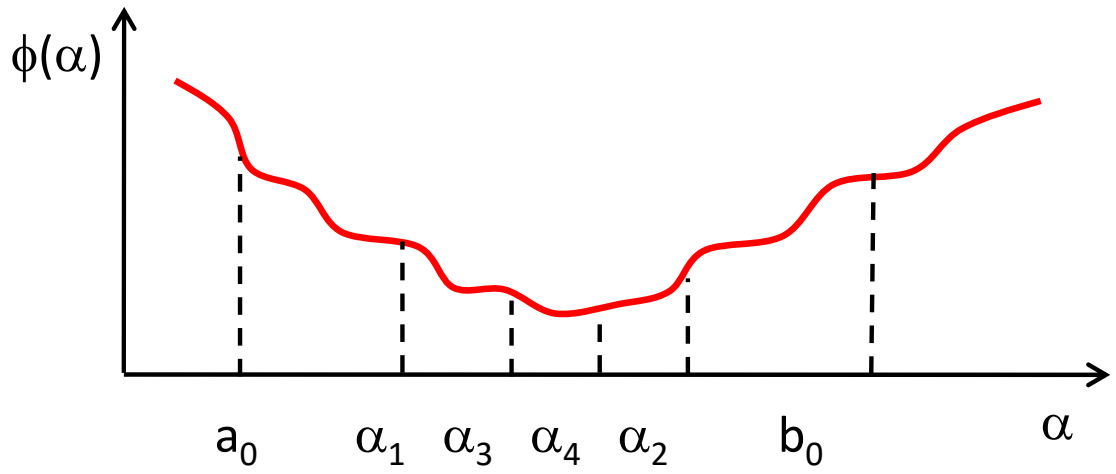
$$\phi' = \frac{\partial f}{\partial \mathbf{b}^{(k+1)}} \cdot \frac{\mathrm{d}\mathbf{b}^{(k+1)}}{\mathrm{d}\alpha} = \mathbf{c}^{(k+1)} \cdot \mathbf{d}^{(k)} = 0$$

$\mathbf{c}^{(k+1)} \cdot \mathbf{d}^{(k)} = 0$ Step size termination criterion

Determine α_k to satisfy this condition

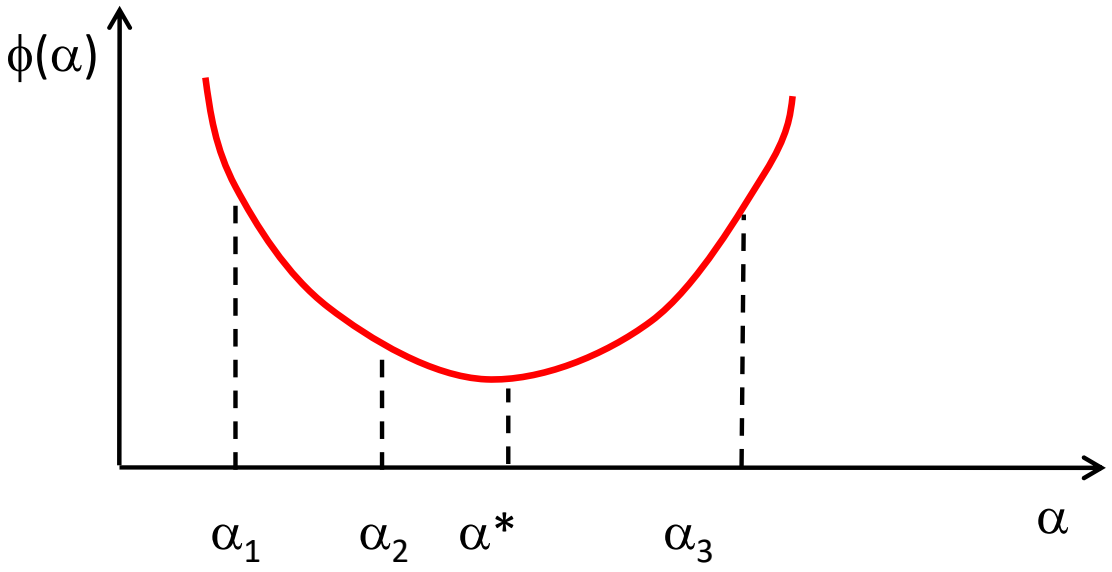
NUMERICAL LINE SEARCH

- 0th-order methods: Bracketing, Quad. Interpolation, Fibonacci & golden section search
- 1st-order methods: Bisection, cubic interpolation
- 2nd-order methods: Newton's method
- Fibonacci & golden section method
  - Reducing interval based on function value
  - Maintain the reduction ratio constant



NUMERICAL LINE SEARCH *cont.*

- Quadratic interpolation



$$\phi(\alpha) = a\alpha^2 + b\alpha + c$$

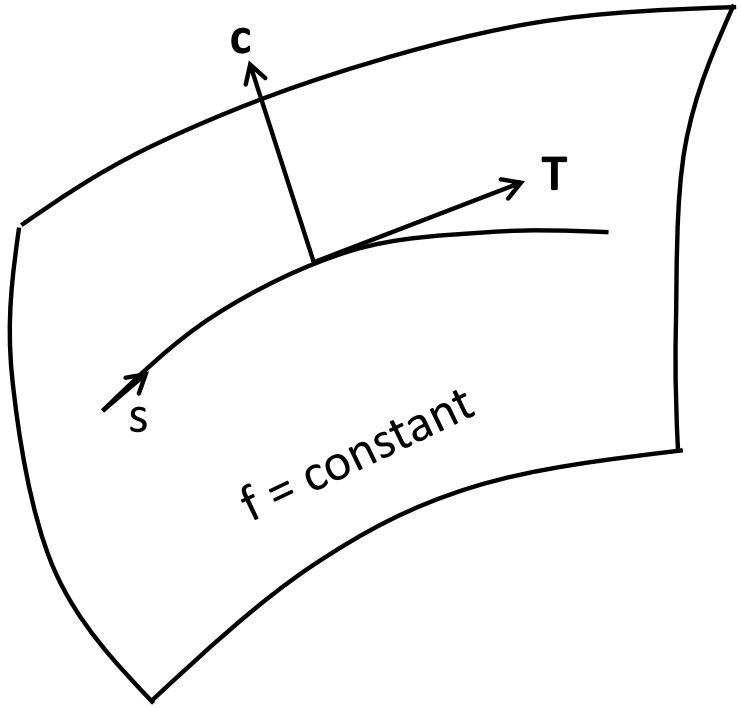
$$\phi'(\alpha) = 2a\alpha + b = 0$$

$$\phi''(\alpha) = 2a > 0$$

$$\alpha^* = -\frac{b}{2a} \quad a > 0,\ b < 0$$

SEARCH (DESCENT) DIRECTION

- Properties of $\mathbf{c}^{(k)} = \nabla f^{(k)}$
  - Normal to the surface f = constant
  - $f(\mathbf{b})$ increases in the direction of $\mathbf{c}^{(k)}$
  - Increasing ratio is max in the direction of $\mathbf{c}^{(k)}$



$$\mathbf{T} = \left\{ \frac{db_1}{ds}, \frac{db_2}{ds}, \ldots, \frac{db_N}{ds} \right\}^T$$

$$\frac{df}{ds} = \frac{df}{d\mathbf{b}} \cdot \frac{d\mathbf{b}}{ds} = \mathbf{c} \cdot \mathbf{T} = 0$$

$$\therefore \mathbf{c} \perp \mathbf{T}$$

T: Tangent vector

STEEPEST DESCENT METHOD (SDM)

- Choose

$$\mathbf{d}^{(k)} = -\mathbf{c}^{(k)}$$

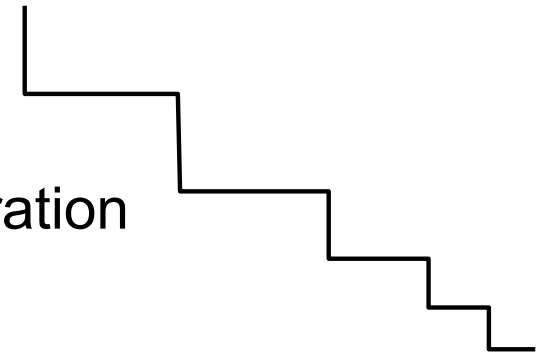
- Descent condition

$$\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)} = -\left\|\mathbf{c}^{(k)}\right\|^2 < 0$$

- Converges slowly near the optimum point
- From step-size termination criterion:

$$\mathbf{c}^{(k+1)} \cdot \mathbf{d}^{(k)} = -\mathbf{c}^{(k+1)} \cdot \mathbf{c}^{(k)} = 0 \qquad \therefore \mathbf{c}^{(k+1)} \perp \mathbf{c}^{(k)}$$

- Search directions are perpendicular
  - Not using the information from the previous iteration



NEWTON'S METHOD

Minimize $f(\mathbf{b})$ $\qquad \mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \Delta\mathbf{b}^{(k)}$

- Quadratic approximation

$$\phi(\Delta\mathbf{b}^{(k)}) = f(\mathbf{b}^{(k)} + \Delta\mathbf{b}^{(k)}) \approx f_K + \mathbf{c}^{(k)} \cdot \Delta\mathbf{b}^{(k)} + \frac{1}{2}\Delta\mathbf{b}^{(k)} \cdot \mathbf{H} \cdot \Delta\mathbf{b}^{(k)}$$

Hessian matrix

- Find $\Delta\mathbf{b}^{(k)}$ that minimizes $\phi(\Delta\mathbf{b}^{(k)})$

Minimize $\phi(\Delta\mathbf{b}^{(k)})$

$$\frac{\partial\phi}{\partial\Delta\mathbf{b}^{(k)}} = \mathbf{c}^{(k)} + \mathbf{H} \cdot \Delta\mathbf{b}^{(k)} = 0$$

$$\therefore \Delta\mathbf{b}^{(k)} = -\mathbf{H}^{-1} \cdot \mathbf{c}^{(k)}$$

Fast convergence (2nd-order)

- Unstable when $\mathbf{H}^{(k)}$ is singular
- Expensive to calculate $\mathbf{H}^{(k)}$
- Accuracy of $\mathbf{H}^{(k)}$ affects convergence

NEWTON'S METHOD *cont*.

- Check descent condition

$$\mathbf{c}^{(k)} \cdot \Delta\mathbf{b}^{(k)} < 0$$

$$-\mathbf{c}^{(k)} \cdot \mathbf{H} \cdot \mathbf{c}^{(k)} < 0$$

- If $\mathbf{H}^{(k)}$ is positive definite, then $\Delta\mathbf{b}^{(k)}$ is a descent direction
- Modified Newton's method (add line search)

$$\Delta\mathbf{b}^{(k)} = \alpha_K \mathbf{d}^{(k)}$$

$$\mathbf{d}^{(k)} = -\mathbf{H}^{-1} \cdot \mathbf{c}^{(k)}$$

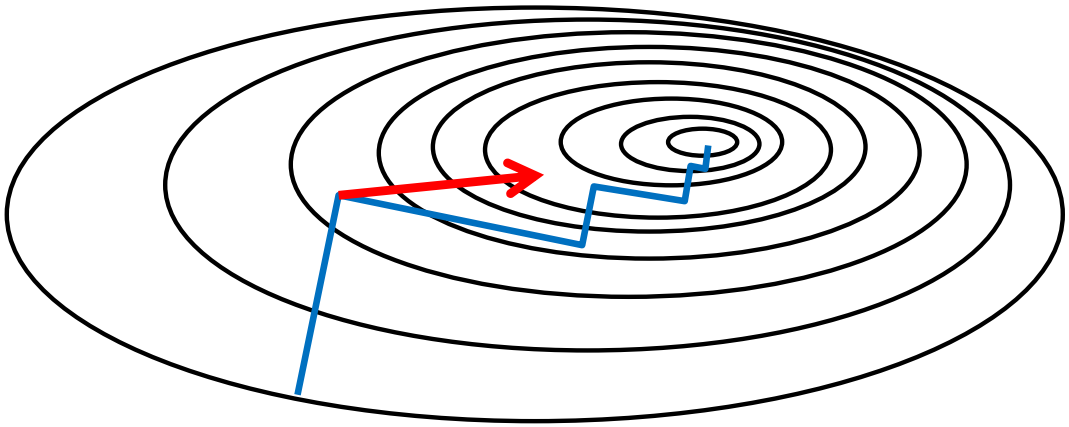
CONJUGATE GRADIENT METHOD

- Use gradient information from the previous iteration
- Let $\mathbf{d}^{(0)} = -\mathbf{c}^{(0)}$ (Steepest descent method)

$$\mathbf{d}^{(k)} = -\mathbf{c}^{(k)} + \beta_k \mathbf{d}^{(k-1)}$$

$$\beta_k = \left(\frac{\left\| \mathbf{c}^{(k)} \right\|}{\left\| \mathbf{c}^{(k-1)} \right\|} \right)^2 \geq 0$$

At optimum, $\beta_k = 0$



QUASI-NEWTON METHOD

- Newton's Method
  - Fast converges, but expensive to calculate the Hessian matrix
- Quasi-Newton Method
  - Approximate the Hessian matrix or its inverse

$$\left.\begin{aligned}\mathbf{d}^{(k)} &= -\mathbf{H}^{-1(k)} \cdot \mathbf{c}^{(k)} \\ \mathbf{d}^{(k)} &= -\mathbf{A}^{(k)} \cdot \mathbf{c}^{(k)}\end{aligned}\right\} + \text{line search}$$

H or **A** need to be Positive Definite

- Defining terms

$$\mathbf{s}_k = \mathbf{b}^{(k+1)} - \mathbf{b}^{(k)} = \alpha_K \mathbf{d}^{(k)} \quad : \text{Design change}$$

$$\mathbf{y}_k = \mathbf{c}^{(k+1)} - \mathbf{c}^{(k)} \quad : \text{Gradient change}$$

} Given

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \Delta\mathbf{H}_k, \qquad \mathbf{H}_0 = \mathbf{I} \quad : \text{Hessian change}$$

Want to update

- Quasi-Newton condition

$$\mathbf{c}^{(k+1)} = \mathbf{c}(\mathbf{b}^{(k)} + \mathbf{s}_k) = \mathbf{c}^{(k)} + \mathbf{H}_k \cdot \mathbf{s}_k + \text{H.O.T.}$$

$$\Rightarrow \mathbf{H}_k \cdot \mathbf{s}_k \approx \mathbf{c}^{(k+1)} - \mathbf{c}^{(k)} = \mathbf{y}_k$$

$$\therefore \mathbf{y}_k \approx \mathbf{H}_k \cdot \mathbf{s}_k$$

$$\Rightarrow \mathbf{s}_k \cdot \mathbf{y}_k \approx \mathbf{s}_k \cdot \mathbf{H}_k \cdot \mathbf{s}_k \quad \text{(curvature)}$$

- We want

Maintaining curvature

$$\mathbf{s}_k \cdot \mathbf{H}_{k+1} \cdot \mathbf{s}_k = \mathbf{s}_k \cdot \mathbf{H}_k \cdot \mathbf{s}_k = \mathbf{s}_k \cdot \mathbf{y}_k$$

$$\Rightarrow \mathbf{s}_k \cdot (\mathbf{H}_{k+1} \cdot \mathbf{s}_k - \mathbf{y}_k) = 0$$

$$\mathbf{H}_{k+1} \cdot \mathbf{s}_k = \mathbf{y}_k$$

$$\mathbf{s}_k = \mathbf{A}_{k+1} \cdot \mathbf{y}_k$$

Quasi-Newton condition

QUASI-NEWTON METHOD *cont.*

- Rank-1 updating (approximation of **H** or **A**)

$$\mathbf{H}_{k+1} = \mathbf{H}_k + a_k \mathbf{u}\mathbf{u}^{\mathrm{T}}$$

Choose **u** to satisfy quasi-Newton cond.

- Rank-2 Hessian updating

$$\mathbf{H}_{k+1} = \mathbf{H}_k + a_k \mathbf{u}\mathbf{u}^{\mathrm{T}} + b_k \mathbf{v}\mathbf{v}^{\mathrm{T}}$$

$$\mathbf{H}_{k+1} \cdot \mathbf{s}_k = \mathbf{H}_k \cdot \mathbf{s}_k + a_k \mathbf{u}\mathbf{u}^{\mathrm{T}} \cdot \mathbf{s}_k + b_k \mathbf{v}\mathbf{v}^{\mathrm{T}} \cdot \mathbf{s}_k = \mathbf{y}_k$$

– Choose $\mathbf{u} = \mathbf{y}_k$, then

$$\mathbf{H}_k \cdot \mathbf{s}_k + a_k \mathbf{y}_k (\mathbf{y}_k \cdot \mathbf{s}_k) + b_k \mathbf{v} (\mathbf{v} \cdot \mathbf{s}_k) = \mathbf{y}_k$$

$$a_k = \frac{1}{\mathbf{y}_k \cdot \mathbf{s}_k}, \quad b_k = -\frac{1}{\mathbf{v} \cdot \mathbf{s}_k}$$

$$\Rightarrow \mathbf{H}_{k+1} \cdot \mathbf{s}_k = \mathbf{H}_k \cdot \mathbf{s}_k + \mathbf{y}_k - \mathbf{v}$$

$$\therefore \mathbf{v} = \mathbf{H}_k \cdot \mathbf{s}_k$$

$$\mathbf{u} = \mathbf{y}_k$$
$$\mathbf{v} = \mathbf{H}_k \cdot \mathbf{s}_k$$

Rank-2 updating

QUASI-NEWTON METHOD *cont.*

- Summary (Rank-2 Hessian updating)

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k \cdot \mathbf{s}_k} - \frac{(\mathbf{H}_k \cdot \mathbf{s}_k)(\mathbf{H}_k \cdot \mathbf{s}_k)^T}{\mathbf{s}_k \cdot \mathbf{H}_k \cdot \mathbf{s}_k}$$

BFGS update

- Properties:
  - $\mathbf{H}_{k+1}$ is positive definite
  - If $f(\mathbf{b})$ is quadratic, $\mathbf{H}_N$ will be the exact Hessian
- Rank-2 Hessian inverse update

$$\mathbf{A}_{k+1} = \mathbf{A}_k + a_k \mathbf{u}\mathbf{u}^T + b_k \mathbf{v}\mathbf{v}^T$$

  - Choose $\mathbf{u} = \mathbf{s}_k$

$$\mathbf{A}_{k+1} \cdot \mathbf{y}_k = \mathbf{A}_k \cdot \mathbf{y}_k + a_k \mathbf{s}_k (\mathbf{s}_k \cdot \mathbf{y}_k) + b_k \mathbf{v}(\mathbf{v} \cdot \mathbf{y}_k) = \mathbf{s}_k$$

$$a_k = \frac{1}{\mathbf{s}_k \cdot \mathbf{y}_k}, \quad b_k = -\frac{1}{\mathbf{v} \cdot \mathbf{y}_k}$$

- Quasi-Newton Condition

$$\Rightarrow \mathbf{A}_{k+1} \cdot \mathbf{y}_k = \mathbf{A}_k \cdot \mathbf{y}_k + \mathbf{s}_k - \mathbf{v} = \mathbf{s}_k$$

$$\therefore \mathbf{v} = \mathbf{A}_k \cdot \mathbf{y}_k$$

- Summary

$$\mathbf{A}_{k+1} = \mathbf{A}_k + \frac{\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}}}{\mathbf{s}_k \cdot \mathbf{y}_k} - \frac{(\mathbf{A}_k \cdot \mathbf{y}_k)(\mathbf{A}_k \cdot \mathbf{y}_k)^{\mathrm{T}}}{\mathbf{y}_k \cdot \mathbf{A}_k \cdot \mathbf{y}_k}$$

DFP update

- Have a similar properties with updated Hessian

Rate of convergence

- Convergence ratio, β, indicates how fast the algorithm converges to the optimum solution
- Sequence of solution $\{x^{(k)}\} \rightarrow x^*$
- Order of convergence
  - A sequence $\{x^{(k)}\}$ is said to converge to x^* with order p where p is the largest number such that

$$0 \leq \lim_{k \to \infty} \frac{\left\| x^{(k+1)} - x^* \right\|}{\left\| x^{(k)} - x^* \right\|^p} < \infty$$

- Convergence ratio

$$\beta = \lim_{k \to \infty} \frac{\left\| x^{(k+1)} - x^* \right\|}{\left\| x^{(k)} - x^* \right\|^p}$$

Rate of convergence *cont*.

- If $p = 1$, the sequence displays linear convergence, and β must be < 1 for the sequence to converge
- $p = 2$: quadratic convergence
- If $\beta = 0$ when $p = 1$, then the convergence is super-linear
- Ex) $x^{(k)} = a^k, a < 1, x^* = 0$
  - $p = 1$: $\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^1} = \frac{a^{k+1}}{a^k} = a < \infty$

    Order of convergence = 1
  - $p = 2$: $\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^2} = \frac{a^{k+1}}{a^{2k}} = \frac{1}{a^{k-1}} \to \infty$
  - With $p = 1$, $\beta = \frac{a^{k+1}}{a^k} = a$
  - Therefore, linear convergence with convergence ratio $\beta = a$

Rate of convergence *cont*.

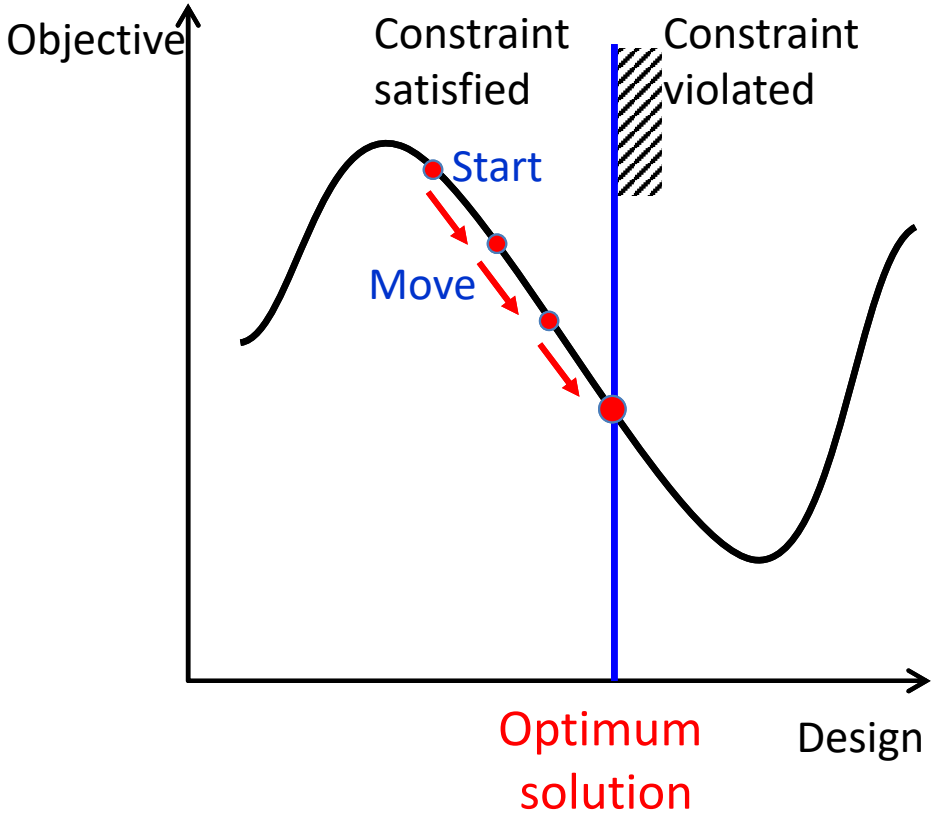
- Ex) $x^{(k)} = \frac{1}{k}, x^* = 0$
  - $p = 1$: $\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^1} = \frac{k}{k+1} = \frac{1}{1+1/k} \to 1 < \infty$
  - $p = 2$: $\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^2} = \frac{k^2}{k+1} \to \infty$
  - Therefore, the order of convergence $p = 1$
  - $\beta = \frac{k}{k+1} = \frac{1}{1+1/k} \to 1$
  - Therefore, linear convergence with convergence ratio=1

EXERCISE: UNCONSTRAINED ALGORITHMS

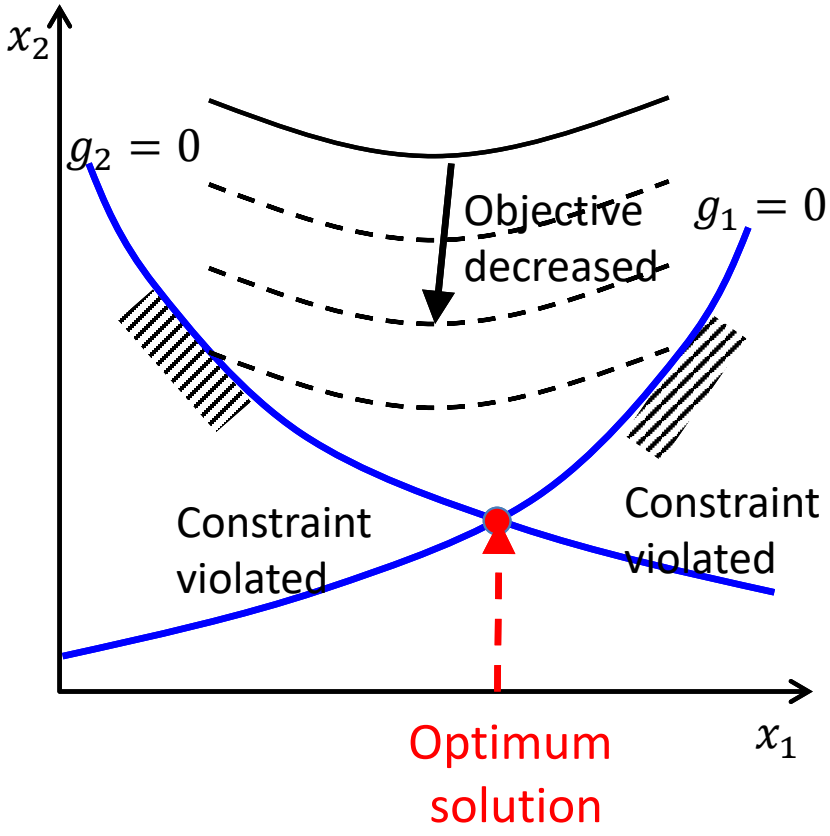
- Explain the differences and commonalities of steepest descent, conjugate gradients, Newton's method, and quasi-Newton methods for unconstrained minimization.
- Use fminunc to minimize the Rosenbrock Banana function and compare the trajectories of fminsearch and fminunc starting from (-1.2,1), with and without the routine for calculating the gradient. Plot the three trajectories.

HOW CONSTRAINTS PLAY IN OPTIMIZATION?

- Most cases, constraints determine optimal design



Single constraint example



Two constraints example

- Constrained Optimization Problem

$$\begin{aligned} &\text{Minimize} \quad f(\mathbf{x}) \\ &\text{subject to} \quad \mathbf{h}(\mathbf{x}) = 0 \\ &\qquad\qquad\quad \mathbf{g}(\mathbf{x}) \le 0 \end{aligned}$$

- Transform to equivalent unconstrained problem
- Lagrange Multiplier Method

$$\underset{\mathbf{b},\lambda,\mu}{\text{Minimize}} \quad L(\mathbf{b},\lambda,\mu) = f(\mathbf{b}) + \sum_{i=1}^{M} \lambda_i h_i(\mathbf{b}) + \sum_{j=1}^{K} \mu_j g_j^{+}(\mathbf{b})$$

  - Use unconstrained optimization algorithms to solve

Active inequality constraints

- SUMT (Sequential Unconstrained Minimization Tech.)
  - Quadratic loss function

$$\Phi(\mathbf{b},\omega) = f(\mathbf{b}) + P(\mathbf{h},\mathbf{g},\omega)$$

ω: Penalty parameter

$$P(\mathbf{h},\mathbf{g},\omega) = \omega\left\{\sum_{i=1}^{M} h_i^2 + \sum_{j=1}^{K} (g_j^+)^2\right\}$$

$$g_j^+ = \max(0, g_j)$$

Violated inequality constraints

  - Gradually increase ω during optimization iteration
  - As ω approaches infinity, solution converges to the original constrained problem
  - As ω increases, the Hessian matrix becomes ill-conditioned

EXAMPLE

- Equality constraint

  Minimize $f(\mathbf{b}) = b_1^2 + 10b_2^2$

  subject to $h(\mathbf{b}) = b_1 + b_2 = 4$

- Loss function

  $$\phi(\mathbf{b},\omega) = b_1^2 + 10b_2^2 + \omega(4 - b_1 - b_2)^2$$

- Optimum design as a function of w

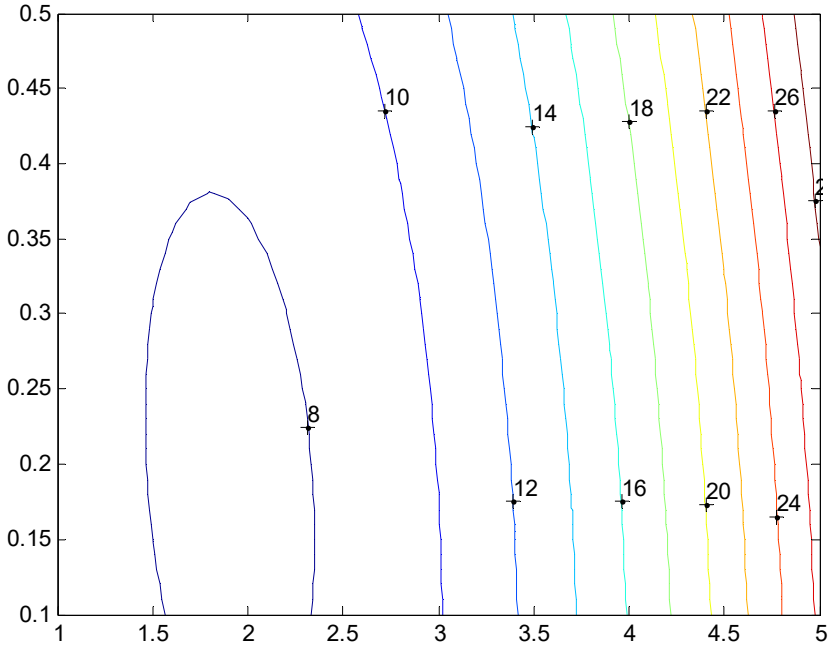
$$b_1 = \frac{40\omega}{10 + 11\omega}$$

$$b_2 = \frac{4\omega}{10 + 11\omega}$$

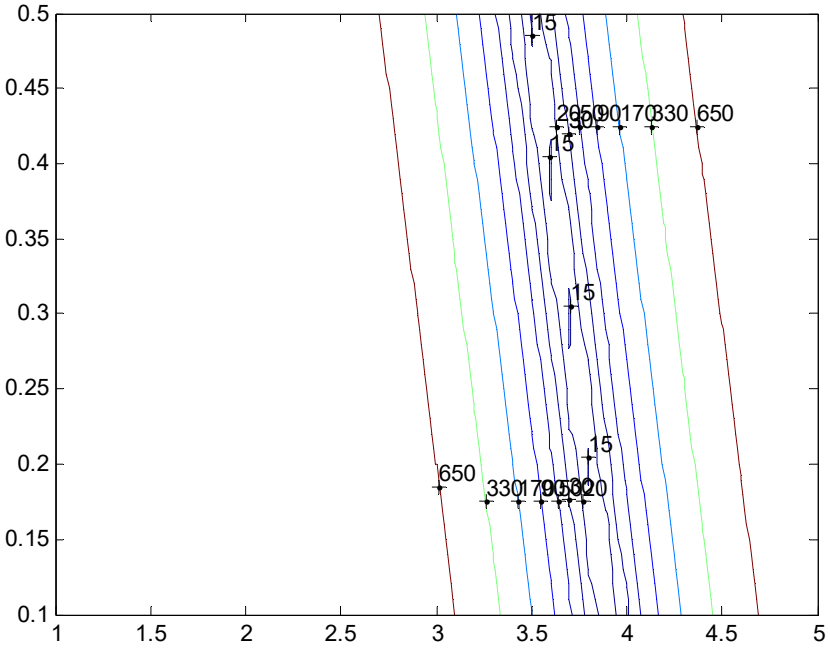
ω	b_1	b_2	f	ϕ
1	1.905	0.1905	3.992	7.619
10	3.333	0.3333	12.220	13.333
100	3.604	0.3604	14.288	14.144
1000	3.633	0.3633	14.518	14.532

CONTOUR OF LOSS FUNCTION

$\omega = 1$



$\omega = 1000$



For non-derivative methods can avoid this by having penalty proportional to absolute value of violation instead of its square!

EXERCISE: PENALTY

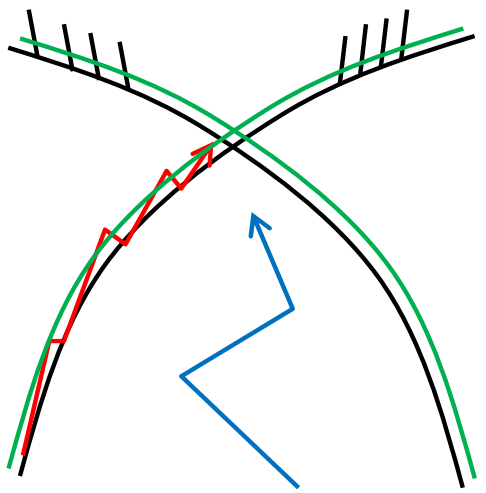
- With an extremely robust algorithm, we can find a very accurate solution with a penalty function approach by using a very high ω. However, at some high value the algorithm will begin to falter, either taking very large number of iterations or not reaching the solution. Test fminunc and fminsearch on the example starting from $\mathbf{b}_0$=[2,2]. Start with ω = 1000 and increase.

DIRECT METHODS FOR CONSTRAINED OPTIMIZATION

- Constrained optimization (Direct solution method)

$$\text{Minimize} \quad f(\mathbf{b})$$
$$\text{subject to} \quad \mathbf{h}(\mathbf{b}) = 0$$
$$\mathbf{g}(\mathbf{b}) \le 0$$

- Optimum point is on the boundary of the feasible set
  - Follow the boundary
  - Move into the feasible domain



- Descent function (merit function)
  - Must be reduced from iteration to iteration
  - Should be the same with $f(\mathbf{b})$ at opt
- ε-active constraint set

$$I_\varepsilon = \{ i \mid g_i + \varepsilon \ge 0, \varepsilon > 0 \}$$ Active or violated constraints

SEQUENTIAL LINEAR PROGRAMMING (SLP)

- Linearization of the problem at the current point

$$f(\mathbf{b}^{(k+1)}) \approx f(\mathbf{b}^{(k)}) + \mathbf{c}^{\mathrm{T}}\Delta\mathbf{b}$$

$$h_i(\mathbf{b}^{(k+1)}) \approx h_i(\mathbf{b}^{(k)}) + \nabla h_i^{\mathrm{T}}\Delta\mathbf{b}$$

$$g_j(\mathbf{b}^{(k+1)}) \approx g_j(\mathbf{b}^{(k)}) + \nabla g_j^{\mathrm{T}}\Delta\mathbf{b}, \quad j \in I_\varepsilon$$

$$\mathbf{n}_i = \nabla h_i \qquad \mathbf{N} = [\mathbf{n}_i]$$

$$\mathbf{a}_i = \nabla g_j \qquad \mathbf{A} = [\mathbf{a}_j]$$

Minimize $\mathbf{c}^{\mathrm{T}}\Delta\mathbf{b}$

subject to $\mathbf{n}_i^{\mathrm{T}}\Delta\mathbf{b} + h_i = 0$

$\mathbf{a}_j^{\mathrm{T}}\Delta\mathbf{b} + g_j \le 0$

$\Delta\mathbf{b}^L \le \Delta\mathbf{b} \le \Delta\mathbf{b}^U$

- Solve $\Delta\mathbf{b}$ using a simplex method $\Rightarrow \mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \Delta\mathbf{b}$

QUADRATIC PROGRAMMING (QP) SUBPROBLEM

- QP1: Minimize $\mathbf{c}^T\mathbf{d}$

  subject to $\mathbf{N}^T\mathbf{d} = -\mathbf{h}$

  $\mathbf{A}^T\mathbf{d} \leq -\mathbf{g}$

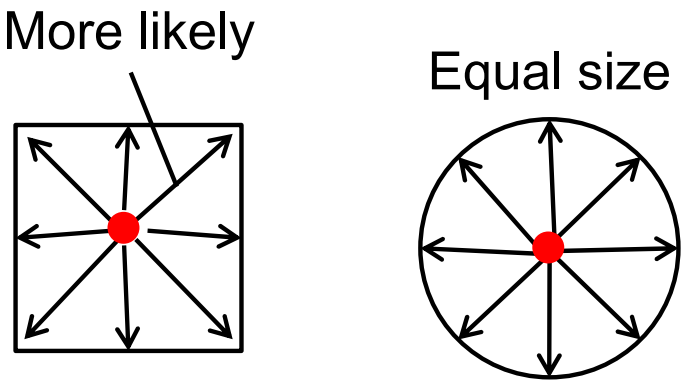
  – Step size constraint

  $$\frac{1}{2}\mathbf{d}^T\mathbf{d} \leq \xi^2$$

- QP2: Minimize $\mathbf{c}^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{d}$

  subject to $\mathbf{N}^T\mathbf{d} = -\mathbf{h}$

  $\mathbf{A}^T\mathbf{d} \leq -\mathbf{g}$



- QP1 with step-size constraint is equivalent to QP2

  – Convex objective function + convex feasible set → Global optimum

CONSTRAINED SDM

- Descent function

$$\Phi = f + \omega_k V \qquad V = \max\left\{0, |h_i|, g_j \;\; j \in I_\varepsilon\right\} \qquad \omega_k = \sum_{i=1}^{M+K} \left|\lambda_i^{(k)}\right|$$

Max. violation

- Set $\mathbf{b}^{(0)}$, k = 0, $\omega_0 = 1$
- Compute $f(\mathbf{b}^{(k)})$, $h_i(\mathbf{b}^{(k)})$, $g_j(\mathbf{b}^{(k)})$, **c**, **N**, **A**, V
- Using QP subproblem, solve for $\mathbf{d}^{(k)}$
- Check for convergence $\|\mathbf{d}^{(k)}\| < \varepsilon_2$ & $V < \varepsilon_1$
- Update ω_k
- Line search: $\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
- $k = k+1$, go to step 2

FEASIBLE DIRECTION METHOD

- Good for inequality constraints Minimize $f(\mathbf{b})$

  subject to $\mathbf{g}(\mathbf{b}) \leq 0$

- Linearization: Minimize $\bar{f} = f_0 + \mathbf{c}^T\mathbf{d}$

  subject to $\bar{g}_j = g_{j0} + \mathbf{a}_j^T\mathbf{d} \leq 0 \qquad g_{j0} \leq 0$

- Feasible direction: $\mathbf{a}_j^T\mathbf{d} \leq 0$
- Usable-feasible direction: $\mathbf{c}^T\mathbf{d} \leq 0$
- Define $\beta = \max\left\{\mathbf{c}^T\mathbf{d}, \mathbf{a}_i^T\mathbf{d}\right\}$

Minimize $\beta(\mathbf{d})$

subject to $\mathbf{c}^T\mathbf{d} \leq \beta$

$\mathbf{a}_i^T\mathbf{d} \leq \beta$

$-1 \leq \mathbf{d} \leq 1$

SEQUENTIAL QUADRATIC PROGRAMMING (SQP)

- Equality constraint case

Minimize $f(\mathbf{b})$

subject to $h_i(\mathbf{b}) = 0$

$$\Rightarrow \quad L(\mathbf{b},\lambda) = f(\mathbf{b}) + \sum \lambda_i h_i(\mathbf{b})$$

- Optimality condition (KKT)

$$\nabla_{\mathbf{b}} L = \nabla f + \mathbf{N}\boldsymbol{\lambda} = 0$$

$$\nabla_{\lambda} L = \mathbf{h} = 0$$

- Nonlinear system of equations

$$\mathbf{F}(\mathbf{y}) = \mathbf{0} \qquad \mathbf{F} = \begin{bmatrix} \nabla_{\mathbf{b}} L \\ \mathbf{h} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} \mathbf{b} \\ \boldsymbol{\lambda} \end{bmatrix} \qquad (1)$$

- Solve using Newton's method

$$\mathbf{F}(\mathbf{y}^{(k)} + \Delta\mathbf{y}^{(k)}) = \mathbf{F}(\mathbf{y}^{(k)}) + (\nabla_{\mathbf{y}}\mathbf{F})^{\mathrm{T}} \Delta\mathbf{y}^{(k)} = \mathbf{0}$$

$$\begin{bmatrix} \nabla_{\mathbf{bb}} L & \mathbf{N} \\ \mathbf{N}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \Delta \mathbf{b} \\ \Delta \boldsymbol{\lambda} \end{Bmatrix} = - \begin{Bmatrix} \nabla_{\mathbf{b}} L \\ \mathbf{h} \end{Bmatrix} \qquad \Delta \mathbf{b} = \mathbf{d}, \;\; \Delta \boldsymbol{\lambda} = \boldsymbol{\lambda}^{(K+1)} - \boldsymbol{\lambda}^{(K)}$$

- Use

$$\begin{bmatrix} \nabla_{\mathbf{bb}} L & \mathbf{N} \\ \mathbf{N}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{d}^{(k)} \\ \boldsymbol{\lambda}^{(k+1)} \end{Bmatrix} = - \begin{Bmatrix} \nabla_{\mathbf{b}} f \\ \mathbf{h} \end{Bmatrix}$$

k-th iteration to find KKT condition (2)

- Equivalent QP problem:

$$\text{Minimize} \quad \mathbf{c}^{\mathrm{T}}\mathbf{d} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}[\nabla_{\mathbf{bb}} L]\mathbf{d}$$

$$\text{subject to} \quad \mathbf{N}^{\mathrm{T}}\mathbf{d} = -\mathbf{h} \qquad (3)$$

$$\tilde{L} = \mathbf{c}^{\mathrm{T}}\mathbf{d} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}(\nabla_{\mathbf{bb}} L)\mathbf{d} + \boldsymbol{\lambda}^{\mathrm{T}}(\mathbf{N}^{\mathrm{T}}\mathbf{d} + \mathbf{h})$$

$$\nabla \tilde{L} = \begin{Bmatrix} \mathbf{c} + (\nabla_{\mathbf{bb}} L)\mathbf{d} + \mathbf{N}\boldsymbol{\lambda} \\ \mathbf{N}^{\mathrm{T}}\mathbf{d} + \mathbf{h} \end{Bmatrix} = \mathbf{0}$$

SQP *cont*.

- Solve QP (3) to find k-th iteration of (1)
- Use BFGS to update $[\nabla_{\mathbf{bb}} L]$

- For inequality constraints

$$\text{Minimize} \quad \mathbf{c}^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T[\nabla_{\mathbf{bb}} L]\mathbf{d}$$

$$\text{subject to} \quad \mathbf{N}^T\mathbf{d} = -\mathbf{h}$$

$$\mathbf{A}^T\mathbf{d} \leq -\mathbf{g}$$

MATLAB OPTIMIZATION TOOLBOX

- Matlab has many optimization functions

Type	Function
Scalar min	fminbnd
Unconstrained min	fminunc, fminsearch
Linear programming	linprog
Quadratic programming	quadprog
Constrained min	fmincon
Goal attainment	fgoalattain
Minmax	fminmax
Semi-inf min	fseminf
Binary integer prog	bintprog

- fmincon is most commonly used – nonlinear obj & const. (read Matlab fmincon manual)

MATLAB fmincon FUNCTION

- Example Minimize $f(\mathbf{b}) = e^{b_1}(4b_1^2 + 2b_2^2 + 4b_1b_2 + 2b_2 + 1)$

  subject to $g_1(\mathbf{b}) = b_1b_2 - b_1 - b_2 + 1.5 \le 0$

  $g_2(\mathbf{b}) = -b_1b_2 - 10 \le 0$

- Create m-file objfun.m

```
function f = objfun(b)
f = exp(b(1))*(4*b(2)^2 + 2*b(2)^2 + 4*b(1)*b(2) + 2*b(2) + 1);
```

- Create m-file confun.m

```
function [c, ceq] = confun(b)
% Nonlinear inequality constraint
c = [1.5 + b(1)*b(2) – b(1) – b(2);
     -b(1)*b(2) – 10];
% Nonlinear equality constraints
ceq = [];
```

MATLAB fmincon FUNCTION *cont*.

- Invoke constrained optimization routine

```
b0 = [-1,1]; % Make a starting guess at the solution
options = optimset('LargeScale','off');
[b, fval, exitflag, output, lambda, grad, hessian] = ...
fmincon(@objfun,b0,[],[],[],[],[],[],@confun, options)
```

  - b contains optimum design variable
  - fval contains optimum objective
  - You can evaluate constraints at optimum
  - [c, ceq] = confun(b)

MATLAB fmincon FUNCTION *cont*.

- Possible to provide lower- & upper-bounds of design variable
- Possible to provide gradients of objective and constraints
- optimset controls algorithm, convergence criteria, results display, etc
- Refer to fmincon manual page

MATLAB fmincon FUNCTION *cont.*

- exitflag

exitflag	Meaning
1	First-order optimality measure was less than options.TolFun and maximum constraint violation was less than options.TolCon.
2	Change in x was less than options.TolX.
3	Change in the objective function value was less than options.TolFun.
4	Magnitude of the search direction was less than 2*options.TolX and constraint violation was less than options.TolCon.
5	Magnitude of directional derivative in search direction was less than 2*options.TolFun and maximum constraint violation was less than options.TolCon.
0	Number of iterations exceeded options.MaxIter or number of function evaluations exceeded options.FunEvals.
-1	Algorithm was terminated by the output function.
-2	No feasible point was found.

MATLAB fmincon FUNCTION *cont*.

- optimset

Option	Data	Meaning
Display	‘off’ ‘iter’ ‘final’ ‘notify’	Level of display
GradObj	‘on’ ‘off’	Obj gradient
Jacobian	‘on’ ‘off’	Cons gradient
LargeScale	‘on’ ‘off’	Algorithm
MaxFunEvals	Integer	
MaxIter	Integer	
TolFun	Real	
TolX	Real	

EXAMPLE: QUADRATIC FUNCTION AND CONSTRAINT

```
function f=quad2(b)
  global a
  f=b(1)^2+a*b(2)^2;
end
%
function [c,ceq]=ring(b)
  global ri ro
  c(1)=ri^2-b(1)^2-b(2)^2;
  c(2)=b(1)^2+b(2)^2-ro^2;
  ceq=[];
end
%
b0=[1,10]; a=10;ri=10.; ro=20;
[b,fval]=fmincon(@quad2,b0,[],[],[],[],[],[],@ring)
b =10.0000   -0.0000      fval =100.0000
```

Minimize $f = b_1^2 + ab_2^2 \quad a > 1$

subject to $r_i^2 \leq b_1^2 + b_2^2 \leq r_o^2$

fmincon OUTPUT

[b,fval,flag,output,lambda]=fmincon(@quad2,b0,[],[],[],[],[],[],@ring)

Warning: The default trust-region-reflective algorithm does not solve …. FMINCON will use the active-set algorithm instead.

Local minimum found ….

Optimization completed because the objective function is non-decreasing in feasible directions, to within the default value of the function tolerance, and constraints are satisfied to within the default value of the constraint tolerance.

fmincon OUTPUT

iterations: 6

funcCount: 22

lssteplength: 1 stepsize: 9.0738e-06

algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'

firstorderopt: 9.7360e-08

constrviolation: -8.2680e-11

lambda.ineqnonlin’=1.0000 0

MAKING IT HARDER FOR fmincon

```
a=1.1;
[b,fval,flag,output,lambda]=fmincon(@quad2,b0,[],[],[],[],[],[],@ring)
Maximum number of function evaluations exceeded;
 increase OPTIONS.MaxFunEvals.
b =4.6355    8.9430   fval =109.4628  flag=0
iterations: 14
      funcCount: 202
      lssteplength: 9.7656e-04
      stepsize: 2.2830
      algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
      firstorderopt: 5.7174
      constrviolation: -6.6754
```

RESTART SOMETIMES HELPS

b0=b

b0 = 4.6355 8.9430

[b,fval,flag,output,lambda]=fmincon(@quad2,b0,[],[],[],[],[],[],@ring)

b =10.0000 0.0000 fval =100.0000

flag = 1

iterations: 15

funcCount: 108

lssteplength: 1

stepsize: 4.6293e-04

algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'

firstorderopt: 2.2765e-07

constrviolation: -2.1443e-07

EXERCISE: fmincon

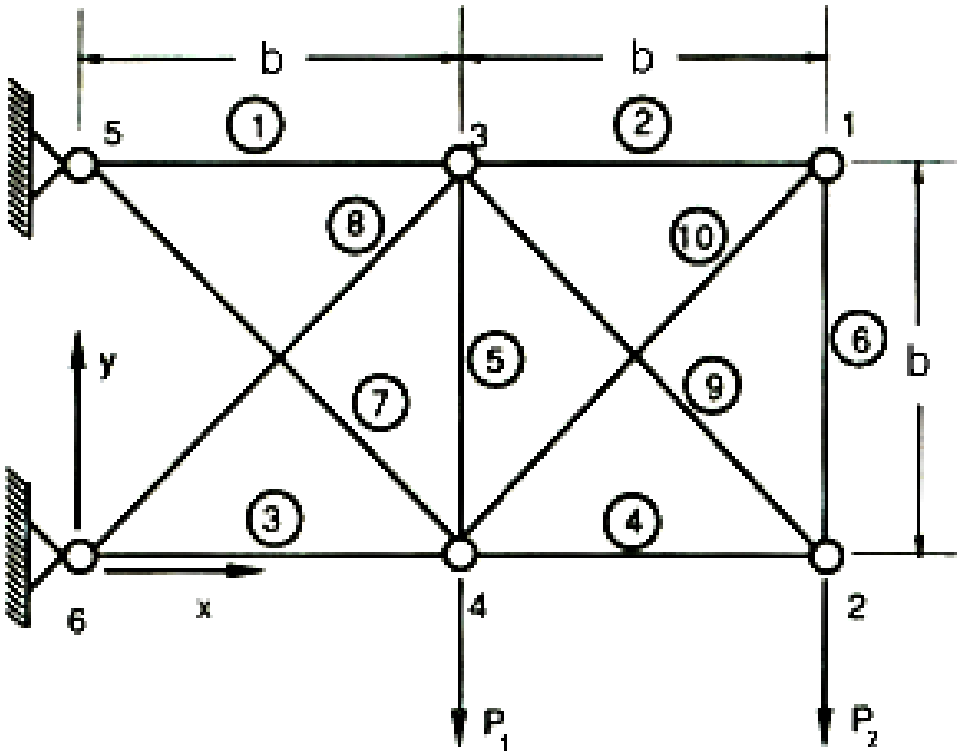
- For the ring problem with a=10, r_o=20, can you find a starting point within a circle of radius 30 around the origin that will prevent fmincon of finding the optimum?

EXERCISE: INEQUALITY

- Solve the problem of minimizing the surface area of the cylinder subject to a minimum volume constraint as an inequality constraint. Do also with Matlab by defining non-dimensional radius and height using the cubic root of the volume.

EXERCISE: TEN-BAR TRUSS

- Minimize the weight of ten-bar truss. Design variables are cross-sectional areas ($A_1 \sim A_{10}$)
- Minimum area = $0.1in^2$, b = 360 in., $P_1 = P_2 = 66.67$ kips.
- All members has stress allowables (25ksi), except for member 9 (75 ksi)



$$a_{11} = \left(\frac{1}{A_1} + \frac{1}{A_3} + \frac{1}{A_5} + \frac{2\sqrt{2}}{A_7} + \frac{2\sqrt{2}}{A_8} \right)$$

$$a_{12} = a_{21} = \frac{1}{A_5}$$

$$a_{22} = \left(\frac{1}{A_2} + \frac{1}{A_4} + \frac{1}{A_5} + \frac{1}{A_6} + \frac{2\sqrt{2}}{A_9} + \frac{2\sqrt{2}}{A_{10}} \right)$$

$$b_1 = \sqrt{2} \left[\frac{P_2}{A_1} - \frac{P_1 + 2P_2}{A_3} - \frac{P_2}{A_5} - \frac{2\sqrt{2}(P_1 + P_2)}{A_7} \right]$$

$$b_2 = \left[\frac{-\sqrt{2}P_2}{A_4} - \frac{\sqrt{2}P_2}{A_5} - \frac{4P_2}{A_9} \right]$$

EXERCISE: TEN-BAR TRUSS *cont*.

- Analytical method of calculating member forces

$$N_8 = \frac{b_1 a_{22} - a_{12} b_2}{a_{11} a_{22} - a_{12} a_{21}}$$

$$N_{10} = \frac{a_{11} b_2 - a_{21} b_1}{a_{11} a_{22} - a_{12} a_{21}}$$

$$N_1 = P_2 - \frac{1}{\sqrt{2}} N_8$$

$$N_2 = -\frac{1}{\sqrt{2}} N_{10}$$

$$N_3 = -P_1 - 2P_2 - \frac{1}{\sqrt{2}} N_8$$

$$N_4 = -P_2 - \frac{1}{\sqrt{2}} N_{10}$$

$$N_5 = -P_2 - \frac{1}{\sqrt{2}} N_8 - \frac{1}{\sqrt{2}} N_{10}$$

$$N_6 = -\frac{1}{\sqrt{2}} N_{10}$$

$$N_7 = \sqrt{2}\left(P_1 + P_2\right) + N_8$$

$$N_9 = \sqrt{2} P_2 + N_{10}$$

- Stresses by dividing by cross-sectional area
- We will practice FEA version later

EXERCISE

- Make Matlab codes for objfun and confun for ten-bar truss
  - Objective: minimize weight (volume) of truss
  - Constraints: member stress(i) ≤ allowable stress(i)
- Make Matlab code to optimize the ten-bar truss using fmincon
  - Use initial cross-sectional area $A_i = 0.5\text{in}^2$
  - Plot objective history (objective value versus iteration)
  - Discuss about EXITFLAG
  - Discuss why some members have minimum area
  - Bonus points: Study the difference between statically determinant and indeterminant systems and discuss how this can affect optimization