Multidisciplinary System Design Optimization (MSDO)

Numerical Optimization II

Lecture 8

Karen Willcox

Today’s Topics

- Sequential Linear Programming
- Penalty and Barrier Methods
- Sequential Quadratic Programming

Technique Overview

UNCONSTRAINED

Steepest Descent

Conjugate Gradient

Quasi-Newton

Newton

CONSTRAINED

Simplex – linear

SLP – often not effective

SQP – nonlinear, expensive, common in engineering applications

Exterior Penalty – nonlinear, discontinuous design spaces

Interior Penalty – nonlinear

Generalized Reduced Gradient – nonlinear

Method of Feasible Directions – nonlinear

Mixed Integer Programming

Standard Problem Definition

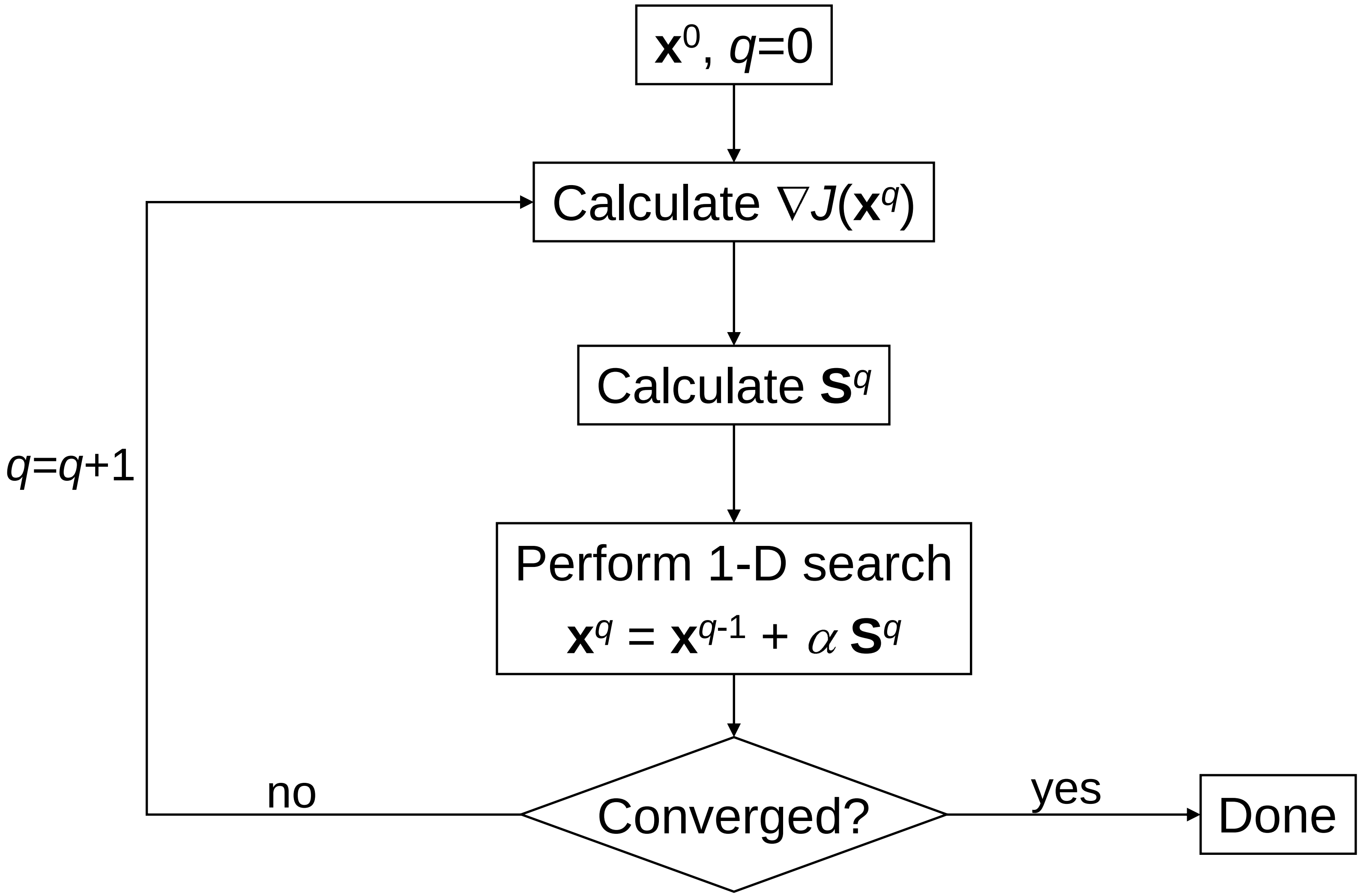
$$\begin{aligned} &\min \; J(\mathbf{x}) \\ &\text{s.t.} \; g_j(\mathbf{x}) \le 0 \quad j = 1,..,m_1 \\ &\quad\;\; h_k(\mathbf{x}) = 0 \quad k = 1,..,m_2 \\ &\quad\;\; x_i^{\ell} \le x_i \le x_i^{u} \quad i = 1,..,n \end{aligned}$$

For now, we consider a single objective function, $J(\mathbf{x})$.

There are n design variables, and a total of m constraints ($m=m_1+m_2$).

For now we assume all x_i are real and continuous.

Optimization Process


$\mathbf{x}^0$, $q=0$
Calculate $\nabla J(\mathbf{x}^q)$
Calculate $\mathbf{S}^q$
Perform 1-D search
$\mathbf{x}^q = \mathbf{x}^{q-1} + \alpha\, \mathbf{S}^q$
Converged?
no
yes
Done
$q=q+1$

Constrained Optimization

Definitions:

- Feasible design: a design that satisfies all constraints
- Infeasible design: a design that violates one or more constraints
- Optimum design: the choice of design variables that minimizes the objective function while satisfying all constraints

In general, constrained optimization algorithms try to cast the problem as an unconstrained optimization and then use one of the techniques we looked at in Lecture 7.

Linear Programming

Most engineering problems of interest are nonlinear

- Can often simplify nonlinear problem by linearization
- LP is often the basis for developing more complicated NLP algorithms

Standard LP problem:

$$\boxed{\begin{aligned}&\min\ J(\mathbf{x}) = \sum_{i=1}^{n} c_i x_i\\ &\sum_{i=1}^{n} a_{ij} x_i = b_j \quad j = 1,..,m\\ &x_i \geq 0 \quad i = 1,\ldots,n\end{aligned}}$$

$$\boxed{\begin{aligned}&\min\ J(\mathbf{x}) = \mathbf{c}^T\mathbf{x}\\ &\mathbf{A}\mathbf{x} = \mathbf{b}\\ &x_i \geq 0 \quad i = 1,\ldots,n\end{aligned}}$$

All LP problems can be converted to this form.

Linear Programming

To convert inequality constraints to equality constraints, use additional design variables:

$$\sum_{i=1}^{n} a_{ji} x_i \leq b_j \quad \text{becomes} \quad \sum_{i=1}^{n} a_{ji} x_i + x_{n+1} = b_j$$

where $x_{n+1} \geq 0$

x_{n+1} is called a **slack variable**

e.g. the constraint

$$x_1 + x_2 \leq 1$$

can be written

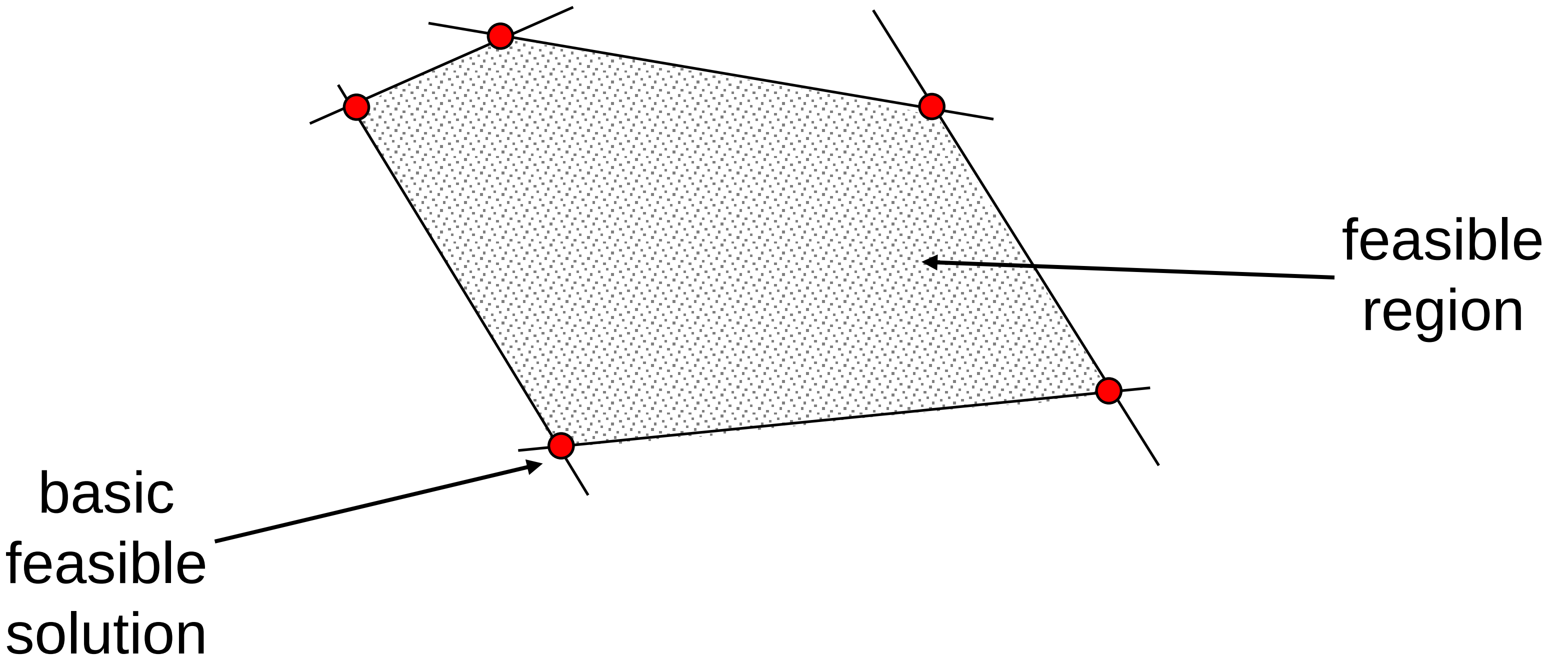
$$x_1 + x_2 + x_3 = 1$$
$$x_i \geq 0, \quad i = 1,2,3$$

if $x_3=0$, this constraint is active

Simplex Method

Solutions at the "vertices" of the design space are called **basic feasible solutions**.

The Simplex algorithm moves from BFS to BFS so that the objective always improves.



Sequential Linear Programming

Consider a general nonlinear problem linearized via first order Taylor series:

$$\begin{aligned}
\text{min} \quad & J(\mathbf{x}) \approx J(\mathbf{x}^0) + \nabla J(\mathbf{x}^0)^T \delta\mathbf{x} \\
\text{s.t.} \quad & g_j(\mathbf{x}) \approx g_j(\mathbf{x}^0) + \nabla g_j(\mathbf{x}^0)^T \delta\mathbf{x} \le 0 \\
& h_k(\mathbf{x}) \approx h_k(\mathbf{x}^0) + \nabla h_k(\mathbf{x}^0)^T \delta\mathbf{x} = 0 \\
& x_i^\ell \le x_i + \delta x_i \le x_i^u
\end{aligned}$$

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

This is an LP problem with the design variables contained in $\delta\mathbf{x}$. The functions and gradients evaluated at $\mathbf{x}^0$ are constant coefficients.

Sequential Linear Programming

1. Initial guess $\mathbf{x}^0$

2. Linearize about $\mathbf{x}^0$ using first-order Taylor series

3. Solve resulting LP to find $\delta\mathbf{x}$

4. Update: $\mathbf{x}^1 = \mathbf{x}^0 + \delta\mathbf{x}$

5. Linearize about $\mathbf{x}^1$ and repeat:

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

where $\delta\mathbf{x}$ is the solution of an LP (model linearized about $\mathbf{x}^{q-1}$).

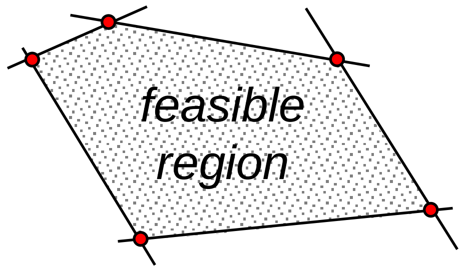
Sequential Linear Programming

- Linearization approximation is only valid close to $\mathbf{x}^0$
- Need to restrict size of update $\delta\mathbf{x}$
- Not considered to be a good method

Nonlinear vs. Linear Constraints

Linear constraints:

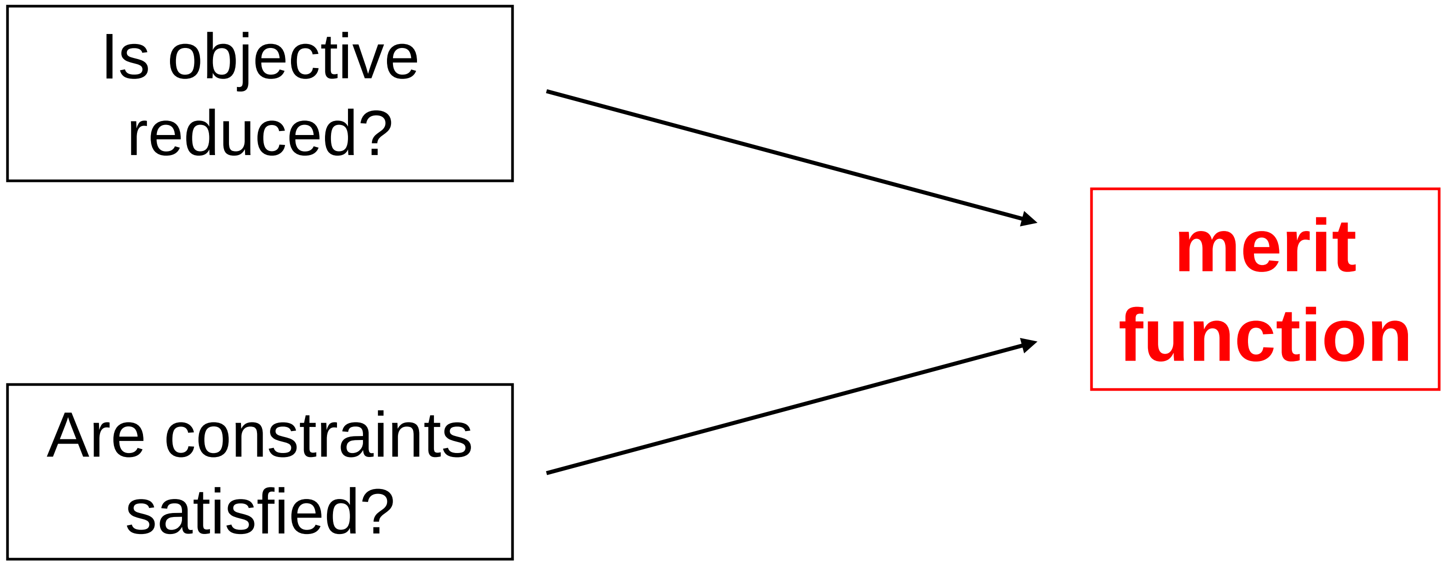
- start from initial feasible point, then all subsequent iterates are feasible
- can construct search direction & step length so that constraints are satisfied
- improvement from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ based on $J(\mathbf{x})$



Nonlinear constraints:

- not straightforward to generate a sequence of feasible iterates
- if feasibility is not maintained, then need to decide whether $\mathbf{x}_{k+1}$ is "better" than $\mathbf{x}_k$

Merit Function



$$\text{merit function} = f(J(\mathbf{x}), \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x}))$$

examples: penalty function methods, barrier function methods

Subproblems

- Many optimization algorithms get to the optimum by generating and solving a sequence of subproblems that are somehow related to the original problem.
- Typically there are two tasks at each iteration:
  1. Calculate search direction
  2. Calculate the step length
- Sometimes, the initial formulation of a subproblem may be defective

  i.e. the subproblem has no solution or the solution is unbounded
- A valid subproblem is one for which a solution exists and is well defined
- The option to abandon should be available if the subproblem appears defective
- It is possible that a defective subproblem is an accurate reflection of the original problem having no valid solution

Penalty and Barrier Methods

General Approach:

- minimize objective as unconstrained function
- provide penalty to limit constraint violations
- magnitude of penalty varies throughout optimization
- called sequential unconstrained minimization techniques (SUMT)
- create pseudo-objective:

$$\Phi(\mathbf{x}, r_p) = J(\mathbf{x}) + r_p P(\mathbf{x})$$

$J(\mathbf{x})$ = original objective function

$P(\mathbf{x})$ = imposed penalty function

r_p = scalar multiplier to determine penalty magnitude

p = unconstrained minimization number

Penalty method approaches useful for incorporating constraints into derivative-free and heuristic search algorithms.

Penalty Methods

Four basic methods:

(i) Exterior Penalty Function Method

(ii) Interior Penalty Function Method

(Barrier Function Method)

(iii) Log Penalty Function Method

(iv) Extended Interior Penalty Function Method

Effect of penalty function is to create a local minimum of unconstrained problem “near” $\mathbf{x}^*$.

Penalty function adds a penalty for infeasibility, barrier function adds a term that prevents iterates from becoming infeasible.

Exterior Penalty Function Method

$$\boxed{\Phi(\mathbf{x},\rho_p) = J(\mathbf{x}) + \rho_p P(\mathbf{x})}$$

$$\boxed{P(\mathbf{x}) = \sum_{j=1}^{m_1} \max\left[0, g_j(\mathbf{x})\right]^2 + \sum_{k=1}^{m_2} h_k(\mathbf{x})^2}$$

- if all constraints are satisfied, then $P(\mathbf{x})=0$
- ρ_p = penalty parameter; starts as a small number and increases
- if ρ_p is small, $\Phi(x,\rho_p)$ is easy to minimize but yields large constraint violations
- if ρ_p is large, constraints are all nearly satisfied but optimization problem is numerically ill-conditioned
- if optimization stops before convergence is reached, the design will be infeasible

Quadratic Penalty Function

$$\boxed{\Phi_Q(\mathbf{x},\rho_p) = J(\mathbf{x}) + \rho_p \left(\sum_{j=1}^{\hat{m}_1} \left[\hat{g}_j(\mathbf{x}) \right]^2 + \sum_{k=1}^{m_2} h_k(\mathbf{x})^{\;2} \right)}$$

- $\hat{g}_j(\mathbf{x})$ contains those inequality constraints that are violated at **x**.
- It can be shown that $\lim_{\rho_p \to \infty} \mathbf{x}^*(\rho_p) = \mathbf{x}^*$
- $\Phi_Q(\mathbf{x},\rho_p)$ is well-defined everywhere

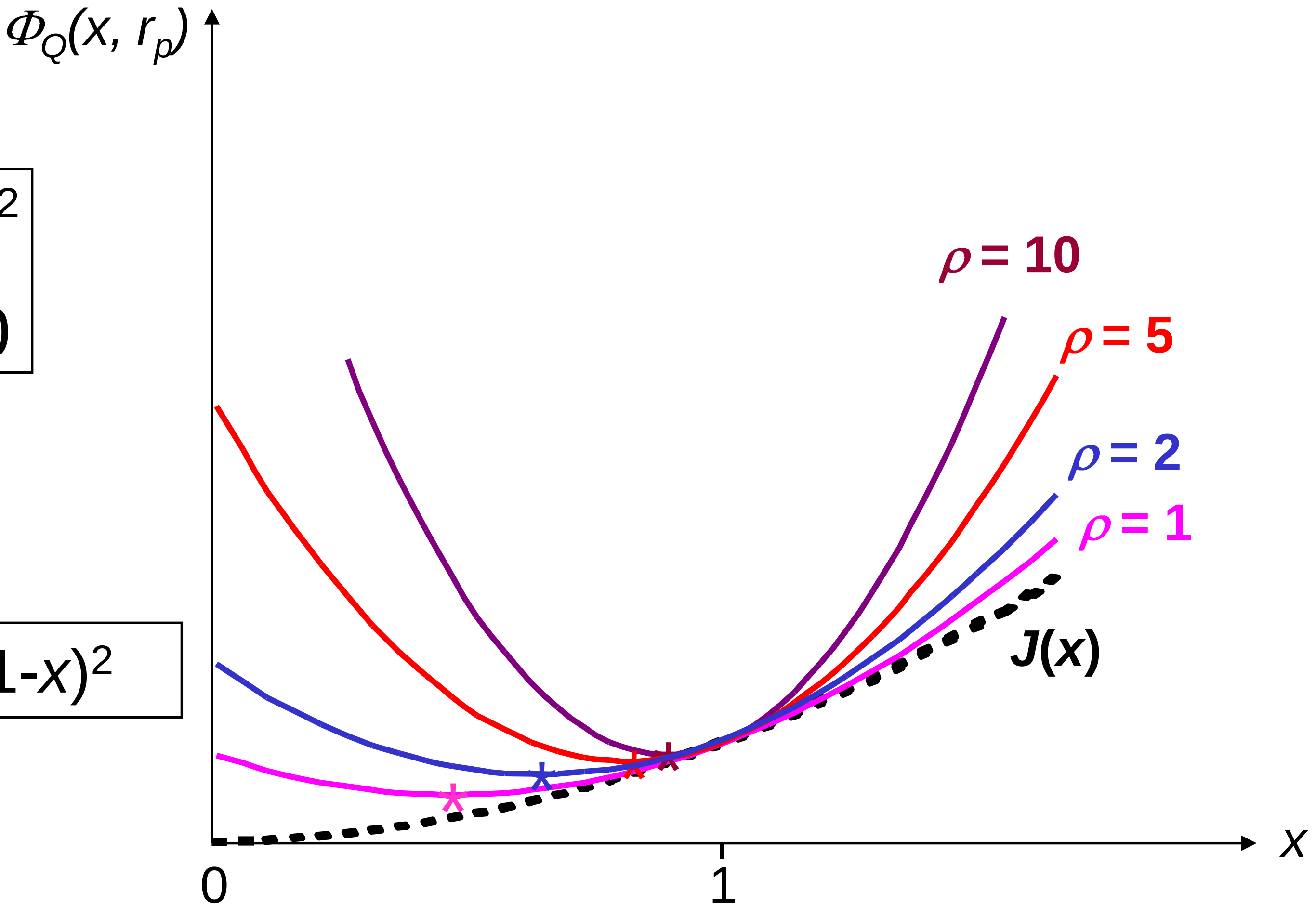
```
Algorithm:  -choose $\rho_0$, set $k$=0
            -find min $\Phi_Q(\mathbf{x},\rho_k) \Rightarrow \mathbf{x}_k^*$
            -if not converged, set $\rho_{k+1} > \rho_k$, $k \leftarrow k+1$ and repeat
```

Note: $\mathbf{x}_k^*$ could be infeasible wrt the original problem

Quadratic Penalty Function Example

$$\min J(x) = x^2$$
$$\text{s.t. } x - 1 = 0$$

$$\Phi_Q(x,\rho) = x^2 + \rho(1-x)^2$$



Absolute Value Penalty Function

$$\boxed{\Phi_A(\mathbf{x},\rho_p) = J(\mathbf{x}) + \rho_p\left(\sum_{j=1}^{\hat{m}_1}\left|\hat{g}_j(\mathbf{x})\right| + \sum_{k=1}^{m_2}\left|h_k(\mathbf{x})\right|\right)}$$

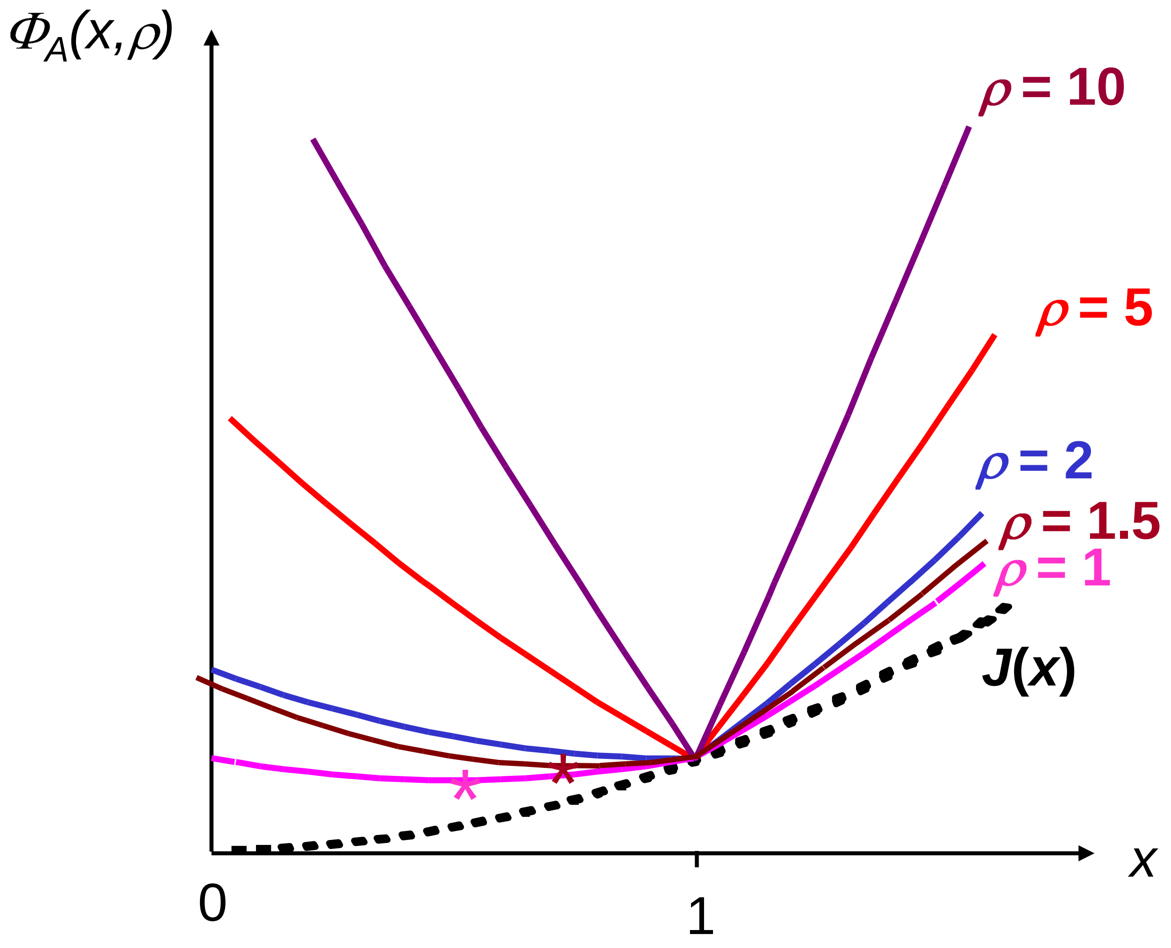
where $\hat{g}_j(\mathbf{x})$ contains those inequality constraints which are violated at **x**.

- Φ_A has discontinuous derivatives at points where $\hat{g}_j(\mathbf{x})$ or $h_k(\mathbf{x})$ are zero.
- The crucial difference between Φ_Q and Φ_A is that we do not require $\rho_p \to \infty$ for **x*** to be a minimum of Φ_A, so we can avoid ill-conditioning
- Instead, for some threshold $\bar{\rho}_p$, **x*** is a minimum of Φ_A for any $\rho_p > \bar{\rho}_p$
- Sometimes called exact penalty functions

Absolute Value Penalty Function Example

$$\boxed{\begin{aligned} &\min J(x) = x^2 \\ &\text{s.t. } \; x - 1 = 0 \end{aligned}}$$

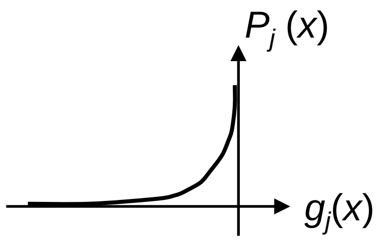
$$\boxed{\Phi_A(x,\rho) = x^2 + \rho\,|1-x|}$$



Interior Penalty Function Method

(Barrier Function Method)

$$\boxed{P(\mathbf{x}) = \sum_{j=1}^{\hat{m}_1} \frac{-1}{\hat{g}_j(\mathbf{x})}}$$



$$\boxed{\Phi(\mathbf{x}, r_p, \rho_p) = J(\mathbf{x}) + r_p \sum_{j=1}^{\hat{m}_1} \frac{-1}{\hat{g}_j(\mathbf{x})} + \rho_p \sum_{k=1}^{m_2} h_k(\mathbf{x})^2}$$

- r_p = barrier parameter; starts as a large positive number and decreases
- barrier function for inequality constraints only
- sequence of improving feasible designs
- $\Phi(x, r_p, \rho_p)$ discontinuous at constraint boundaries

Barrier Function Method

$$\boxed{P(\mathbf{x}) = \sum_{j=1}^{m_1} -\ln\left[-g_j(\mathbf{x})\right]}$$

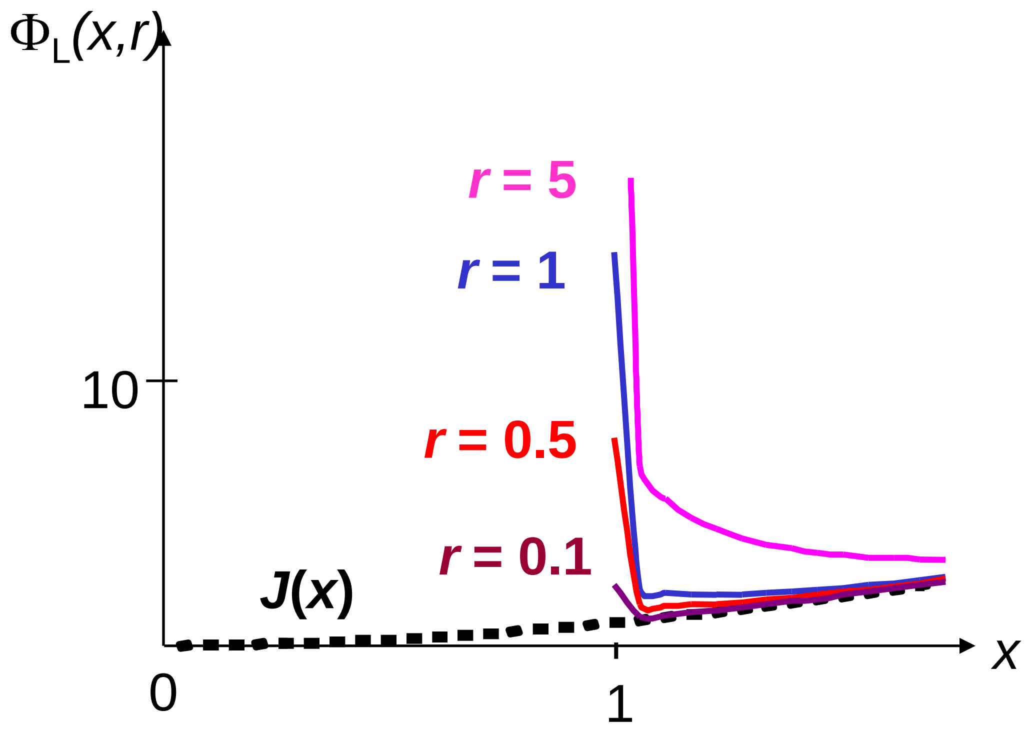
$$\boxed{\Phi_L(\mathbf{x}, r_p, \rho_p) = J(\mathbf{x}) - r_p \sum_{j=1}^{\hat{m}_1} \ln \; -\hat{g}_j(\mathbf{x})}$$

- often better numerically conditioned than $\dfrac{-1}{g_j(\mathbf{x})}$
- penalty function has a positive singularity at the boundary of the feasible region
- penalty function is undefined for $g_i(\mathbf{x})>0$
- $\lim_{r_p \to 0} \mathbf{x}^*(r_p) = \mathbf{x}^*$

Barrier Function Example

$$\min J(x) = x^2$$
$$\text{s.t.} \quad x - 1 \geq 0$$

$$\Phi_L(x,r) = x^2 - r\ln(x-1)$$



Extended Interior Penalty Function Method

Combine features of interior/exterior methods

$$P(\mathbf{x}) = \sum_{j=1}^{m_1} \tilde{g}_j(\mathbf{x})$$

-linear extended penalty function

where $$\tilde{g}_j(\mathbf{x}) = \frac{-1}{g_j(\mathbf{x})} \quad \text{if } g_j(\mathbf{x}) \le \varepsilon$$

$$= -\frac{2\varepsilon - g_j(\mathbf{x})}{\varepsilon^2} \quad \text{if } g_j(\mathbf{x}) > \varepsilon$$

-ε is a small negative number, and marks transition from interior penalty to extended penalty

-ε must be chosen so that Φ has positive slope at constraint boundary

-can also use quadratic extended and variable penalty functions

Sequential Quadratic Programming

- Create a quadratic approximation to the Lagrangian
- Create linear approximations to the constraints
- Solve the quadratic problem to find the search direction, **S**
- Perform the 1-D search
- Update the approximation to the Lagrangian

Sequential Quadratic Programming

Create a subproblem

with **quadratic objective function:**

$$\boxed{\min Q(\mathbf{S}^k) = J(\mathbf{x}^k) + \nabla J(\mathbf{x}^k)^{\mathrm{T}}\mathbf{S}^k + \frac{1}{2}\mathbf{S}^T\mathbf{B}\mathbf{S}}$$

and **linear constraints**

$$\boxed{\begin{aligned} \text{s.t. } & \nabla g_j(\mathbf{x}^k)^{\mathrm{T}}\mathbf{S}^k + g_j(\mathbf{x}^k) \le 0 \quad j = 1,\ldots,m_1 \\ & \nabla h_k(\mathbf{x}^k)^{\mathrm{T}}\mathbf{S}^k + h_k(\mathbf{x}^k) = 0 \quad k = 1,\ldots,m_2 \end{aligned}}$$

- Design variables are the components of **S**
- **B**=**I** initially, then is updated to approximate **H** (as in quasi-Newton)

Incompatible Constraints

- The constraints of the subproblem can be incompatible, even if the original problem has a well-posed solution
- For example, two linearized constraints could be linearly dependent
- This is a common occurrence in practice
- Likelihood of incompatible constraints reduced by allowing flexibility in RHS (e.g. allow scaling factors in front of $g_j(\mathbf{x}^k)$ term)

$$\boxed{\nabla g_j(\mathbf{x}^k)^{\mathrm{T}}\mathbf{S}^k + \gamma_j g_j(\mathbf{x}^k) \leq 0 \quad j = 1,\ldots,m_1}$$

- Typically γ=0.9 if constraint is violated and γ=1 otherwise
- Doesn't affect convergence, since specific form of constraint is only crucial when $\mathbf{x}^k$ is close to $\mathbf{x}^*$

Sequential Quadratic Programming

- Widely used in engineering applications e.g. NLOpt
- Considered to be one of the best gradient-based algorithms
- Fast convergence for many problems
- Strong theoretical basis
- MATLAB®: fmincon (medium scale)

Lecture Summary

We have seen the following nonlinear techniques:

- Penalty and Barrier Methods
- Sequential Quadratic Programming

It is important to understand

- when it is appropriate to use these methods
- the basics of how the method works
- why the method might fail
- what your results mean (numerically vs. physically)

References

Practical Optimization, P.E. Gill, W. Murray and M.H. Wright, Academic Press, 1986.

Numerical Optimization Techniques for Engineering Design, G.N. Vanderplaats, Vanderplaats R&D, 1999.

Optimal Design in Multidisciplinary Systems, AIAA Professional Development Short Course Notes, September 2002.

NLOPT: Open-source library for nonlinear optimization
http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms