Carnegie Mellon

18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

Overview

- **Unconstrained Optimization**
  - Gradient method
  - Newton method

Unconstrained Optimization

- **Linear regression with regularization**

$$A\alpha = B \quad \Rightarrow \quad \min_{\alpha} \ \|A\alpha - B\|_2^2 + \lambda \cdot \|\alpha\|_1$$

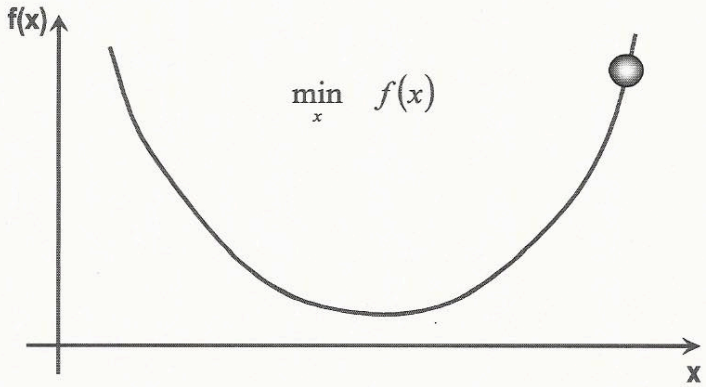
- **Unconstrained optimization: minimizing a cost function without any constraint**
  - Golden section search
  - Downhill simplex method

  → **Non-derivative method**
  - Gradient method
  - Newton method

  → **Rely on derivatives (this lecture)**

Gradient Method

- **If a cost function is smooth, its derivative information can be used to search optimal solution**



Gradient Method

- For illustration purpose, we start from a one-dimensional case

$$\min_x \quad f(x)$$

$\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = -\lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$

$\lambda^{(k)}$: step size; $\left.\frac{df}{dx}\right|_{x^{(k)}}$: derivative

$(\lambda^{(k)} > 0)$

$x^{(k+1)} = x^{(k)} - \lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$

$(\lambda^{(k)} > 0)$

Gradient Method

- One-dimensional case (continued)

$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = -\lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \quad \& \quad f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot \Delta x^{(k)}$$

$f[x^{(k+1)}] \approx f[x^{(k)}] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot [-\lambda^{(k)} \left.\frac{df}{dx}\right|_{x^{(k)}}]$

$\approx f[x^{(k)}] - \lambda^{(k)} [\left.\frac{df}{dx}\right|_{x^{(k)}}]^2$

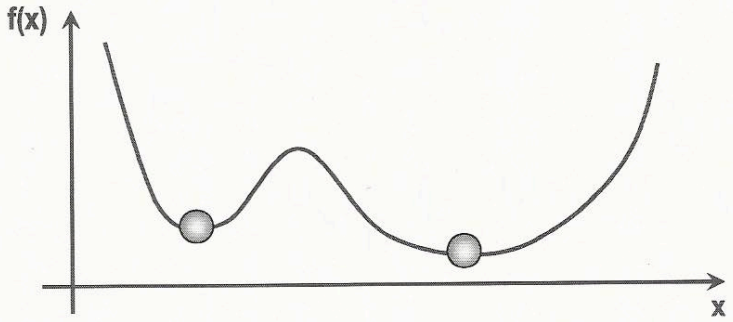
$\lambda^{(k)} > 0 \qquad \geq 0$

Gradient Method

- **One-dimensional case (continued)**

$$\Delta x = -\lambda \cdot df/dx = 0$$

Derivative is zero at local optimum (gradient method converges)



Gradient Method

- **Two-dimensional case**

$$\min_{x_1, x_2} \quad f(x_1, x_2)$$

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \end{bmatrix}$$

$$\begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} = \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix} = -\lambda^{(k)} \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

$$\lambda^{(k)} > 0$$

Gradient Method

■ **Two-dimensional case (continued)**

$$\begin{bmatrix}\Delta x_1^{(k)}\\ \Delta x_2^{(k)}\end{bmatrix} = -\lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] + \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \begin{bmatrix}\Delta x_1^{(k)}\\ \Delta x_2^{(k)}\end{bmatrix}$$

⇩

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] - \lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

⇩

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] - \lambda^{(k)} \cdot \left\|\nabla f\left[x_1^{(k)}, x_2^{(k)}\right]\right\|_2^2 \qquad \lambda^{(k)} > 0$$

The cost function $f(x_1, x_2)$ keeps decreasing if the gradient is non-zero

Gradient Method

■ **N-dimensional case**

$$\min_X \quad f(X)$$

$$\nabla f(x) = \begin{bmatrix}\frac{df}{dx_1}\\ \frac{df}{dx_2}\\ \vdots\end{bmatrix}$$

$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$$

$$= -\lambda^{(k)} \cdot \nabla f\left[x^{(k)}\right]$$

$$\lambda^{(k)} > 0$$

Newton Method

- **Gradient method relies on first-order derivative information**
  - Each iteration is simple, but it converges to optimum slowly
  - I.e., require a large number of iteration steps

- **The step size $\lambda^{(k)}$ can be optimized by one-dimensional search for fast convergence**

$$\min_{\lambda^{(k)}} \quad f\left\{X^{(k)} - \lambda^{(k)} \cdot \nabla f\left[X^{(k)}\right]\right\}$$

- **Alternative algorithm: Newton method**
  - Rely on both first-order and second-order derivatives
  - Each iteration is more expensive
  - But it converges to optimum more quickly, i.e., requires a smaller number of iterations to reach convergence

Newton Method

- **One-dimensional case**

$$\min_{x} \quad f(x)$$

$\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ $\qquad \left.\frac{df}{dx}\right|_{x^{(k+1)}} = 0$

$$\left.\frac{df}{dx}\right|_{x^{(k+1)}} = \left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \cdot \left[x^{(k+1)} - x^{(k)}\right]$$

$$0 = \left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \left[x^{(k+1)} - x^{(k)}\right]$$

Newton Method

- **One-dimensional case (continued)**

$$\left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \cdot \underbrace{\left[x^{(k+1)} - x^{(k)}\right]}_{\Delta x^{(k)}} = 0$$

$$\Delta x^{(k)} = -\left[\left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}\right]^{-1} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$$

$$x^{(k+1)} = x^{(k)} - \left[\left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}\right]^{-1} \left.\frac{df}{dx}\right|_{x^{(k)}}$$

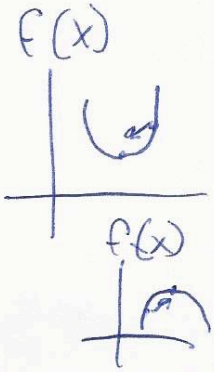
Newton Method

- **One-dimensional case (continued)**

$$\Delta x^{(k)} = -\left.\frac{d^2 f}{dx^2}\right|^{-1}_{x^{(k)}} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \quad \& \quad f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot \Delta x^{(k)}$$

$$f\left[x^{(k+1)}\right] \geq f\left[x^{(k)}\right] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot \left[-\left.\frac{d^2 f}{dx^2}\right|^{-1}_{x^{(k)}} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}\right]$$

$$\geq f\left[x^{(k)}\right] - \underbrace{\left.\frac{d^2 f}{dx^2}\right|^{-1}_{x^{(k)}} \cdot \left[\left.\frac{df}{dx}\right|_{x^{(k)}}\right]^2}_{\text{Convex function}}$$



Newton Method

- **One-dimensional case (continued)**

$$x^{(k+1)} = x^{(k)} - \lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$$

Gradient method

$$x^{(k+1)} = x^{(k)} - \left.\frac{d^2 f}{dx^2}\right|^{-1}_{x^{(k)}} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$$

Newton method

- Newton method gives an estimation of the optimal step size $\lambda^{(k)}$ using second-order derivative

$$\lambda^{(k)} = \left.\frac{d^2 f}{dx^2}\right|^{-1}_{x^{(k)}} > 0 \quad \text{(convex function)}$$

Newton Method

- **One-dimensional case (continued)**
  - The step size estimation is based on the following linear approximation for first-order derivative

$$\left.\frac{df}{dx}\right|_{x^{(k+1)}} \approx \left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \cdot \left[x^{(k+1)} - x^{(k)}\right]$$

  - The approximation is exact if and only if f(x) is quadratic and, therefore, df/dx is linear

$$f(x) = ax^2 + bx + c \quad \Rightarrow \quad \frac{df}{dx} = 2ax + b$$

Newton Method

- **One-dimensional case (continued)**
  - If the actual f(x) is not quadratic, the following step size estimation may be non-optimal

$$\lambda^{(k)} = \left.\frac{d^2 f}{dx^2}\right|^{-1}_{x^{(k)}}$$

  - Using this step size may result in bad convergence
  - In these cases, a damping factor β is typically introduced

$$x^{(k+1)} = x^{(k)} - \beta \cdot \left.\frac{d^2 f}{dx^2}\right|^{-1}_{x^{(k)}} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \quad (0 < \beta < 1)$$

Newton Method

- **Two-dimensional case**

$$\min_{x_1, x_2} \quad f(x_1, x_2)$$

$$\nabla f(x_1, x_2) = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{bmatrix} \quad \& \quad \nabla^2 f(x_1, x_2) = \begin{bmatrix} \partial^2 f / \partial x_1^2 & \partial^2 f / \partial x_1 \partial x_2 \\ \partial^2 f / \partial x_1 \partial x_2 & \partial^2 f / \partial x_2^2 \end{bmatrix}$$

$\nabla f[x_1^{(k+1)}, x^{(k+1)}] \approx \nabla f[x_1^{(k)}, x_2^{(k)}] +$

$\nabla^2 f[x_1^{(k)}, x_2^{(k)}] \times \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix}$

Newton Method

■ **Two-dimensional case (continued)**

$$\nabla f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx \nabla f\left[x_1^{(k)}, x_2^{(k)}\right] + \nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right] \cdot \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix} = 0$$

$$\Downarrow$$

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix} = -\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

Newton Method

■ **Two-dimensional case (continued)**

$$\begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} = -\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] + \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix}$$

$$\Downarrow$$

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] - \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \underline{\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1}} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

Positive definite (convex function)

Newton Method

- **Two-dimensional case (continued)**

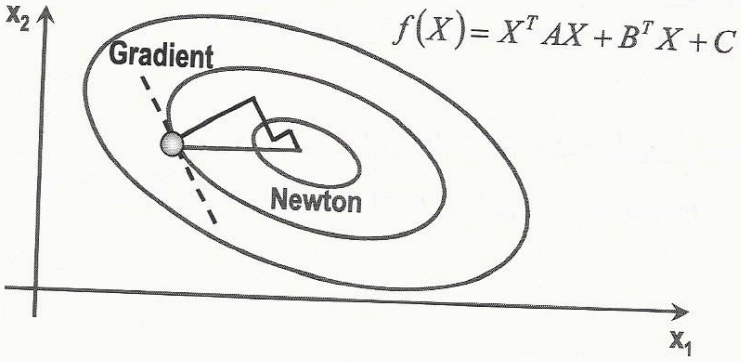
$$\Delta X^{(k)} = -\lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

Gradient method

$$\Delta X^{(k)} = -\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

Newton method

- Gradient method and Newton method do not move along the same direction



$$f(X) = X^T AX + B^T X + C$$

Newton Method

- **Newton method can be extended to high-dimensional cases**
- **The following Hessian matrix is N×N if we have N variables**

$$\nabla^2 f(X) = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 & \cdots & \partial^2 f/\partial x_1 \partial x_N \\ \partial^2 f/\partial x_1 \partial x_2 & \partial^2 f/\partial x_2^2 & \cdots & \partial^2 f/\partial x_2 \partial x_N \\ \vdots & \vdots & \vdots & \vdots \\ \partial^2 f/\partial x_1 \partial x_N & \partial^2 f/\partial x_2 \partial x_N & \cdots & \partial^2 f/\partial x_N^2 \end{bmatrix}$$

- **Numerically computing the Hessian matrix and its inverse (by Cholesky decomposition) can be quite expensive for large N**

$$X^{(k+1)} = X^{(k)} - \nabla^2 f\left[X^{(k)}\right]^{-1} \cdot \nabla f\left[X^{(k)}\right]$$

Newton Method

- **A number of modified algorithms were developed to address this complexity issue**
  - E.g., quasi-Newton method
- **The key idea is to approximate the Hessian matrix and its inverse so that:**
  - The computational cost is significantly reduced
  - Fast convergence can still be achieved
- **More details can be found at**

  Numerical Recipes: The Art of Scientific Computing, 2007

Summary

- **Unconstrained Optimization**
  - Gradient method
  - Newton method

18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

Overview

- **Constrained Optimization**
  - Linear equality constraint
  - Lagrange multiplier

Constrained Nonlinear Optimization

$$\min_X \quad f(X)$$
$$\text{S.T.} \quad \begin{cases} g_1(X) \le 0 \\ g_2(X) \le 0 \\ \vdots \end{cases}$$

- **Equality constraint can be written as two inequality constraints**

$$\min_X \quad f(X)$$
$$\text{S.T.} \quad g(X) = 0$$

$$\min_x \quad f(x)$$
$$\text{S.T.} \quad g(x) \le 0$$
$$-g(x) \le 0$$

Linear Equality Constraint

$$\min_X \quad f(X)$$
$$\text{S.T.} \quad AX = B$$

- **Linear equality constraint can be efficiently handled by a number of optimization algorithms**
  - We do not write AX = B as two inequality constraints
  - It can be directly solved with high efficiency

Subspace Reduction

- **Eliminate linear equality constraint**

$$AX = B$$

P×N P×1

N > P

$X = F \cdot Z + D$ $(\forall Z \in R^{N-P})$

F: N×(N-P), Z: (N-P)×1, D: N×1

- X = FZ+D is the (non-unique) solution of under-determined linear equation AX = B
- For any Z value, AX is equal to B

A Simple Example

$$x_1 + x_2 = 1$$

$x_1 = Z$

$x_2 = 1 - x_1 = 1 - Z$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} Z + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

X F D

Subspace Reduction

$$\min_{X} \quad f(X) \qquad \rightarrow \min_{Z} f(FZ+D)$$
$$\text{S.T.} \quad AX = B$$

$$X = FZ + D$$

- **Solve the optimal value Z by unconstrained optimization – minimizing f(FZ+D)**
- **Calculate the optimum X = FZ+D**

Lagrange Multiplier

- **Equality constraint can also be handled by Lagrange multiplier**
- **If X* is a local minimum of**

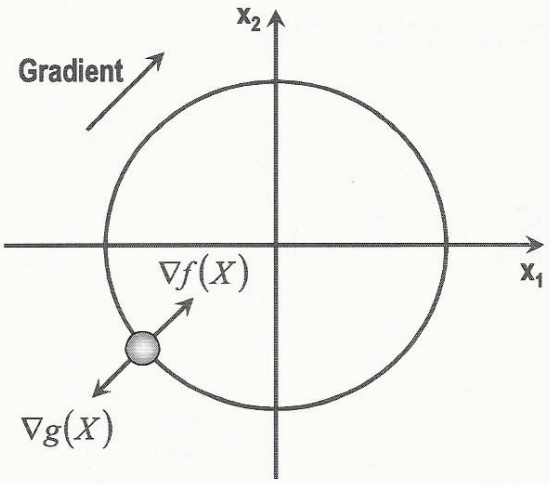
$$\min_{X} \quad f(X)$$
$$\text{S.T.} \quad g_i(X) = 0 \quad (i = 1,2,\cdots,P)$$

  - there exist $\lambda_1, \lambda_2, \ldots, \lambda_P$, called Lagrange multipliers, such that

$$\nabla f(X^*) + \sum_{i=1}^{P} \lambda_i \cdot \nabla g_i(X^*) = 0$$

A Simple Example

$$\min_{x_1,x_2} \quad f(X) = x_1 + x_2$$
$$\text{S.T.} \quad g(X) = x_1^2 + x_2^2 - 2 = 0$$



$x^* = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$

$\nabla f(x^*) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$\nabla g(x^*) = \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$

$\nabla f(x^*) + \frac{1}{2} \nabla g(x^*) = 0$

λ

Lagrange Multiplier

$$\min_X \quad f(X)$$
$$\text{S.T.} \quad g_i(X) = 0 \quad (i = 1,2,\cdots,P) \quad \Rightarrow \quad \nabla f(X^*) + \sum_{i=1}^{P} \lambda_i \cdot \nabla g_i(X^*) = 0$$

- **Optimality condition for linear constraints**

$$\min_X \quad f(X)$$
$$\text{S.T.} \quad AX = B$$

ith row $\begin{bmatrix} \cdots A \cdots \end{bmatrix} \begin{bmatrix} \\ X \\ \\ \end{bmatrix} - \begin{bmatrix} \\ B \\ \\ \end{bmatrix} = 0$

$A(i,:) \cdot x - B(i) = 0$

$= g_i(x)$

$\nabla g_i(x) \quad A^T(:,i)$

Lagrange Multiplier

$$\min_X \quad f(X) \qquad \text{S.T.} \quad g_i(X)=0 \quad (i=1,2,\cdots,P) \quad \Rightarrow \quad \nabla f(X^*)+\sum_{i=1}^{P}\lambda_i \cdot \nabla g_i(X^*)=0$$

- **Optimality condition for linear constraints**

$$\nabla g_i(X)=A(i,:)^T$$

$$\sum_i \lambda_i \cdot \nabla g_i(x)=\sum_i \lambda_i \cdot A(i,:)^T$$

$$\begin{bmatrix} A(1,:)^T & | & A(2,:)^T & | & \cdots \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \end{bmatrix}$$

A^T $\quad V$

Linear Equality Constrained Quadratic Programming

$$\min_X \quad f(X) \qquad \text{S.T.} \quad AX=B \quad \Rightarrow \quad \nabla f(X^*)+A^TV=0$$

- **We first consider quadratic cost function**
  - Any smooth nonlinear cost function can be locally approximated as a quadratic function (2nd-order Taylor expansion)

$$\min_X \quad f(X)=\frac{1}{2}X^TQX+R^TX+C \qquad \text{S.T.} \quad AX=B$$

$$\nabla f(x)=Qx+R$$

$$Qx^*+R+A^TV=0$$

Linear Equality Constrained Quadratic Programming

$$\begin{aligned} \min_X \quad & \frac{1}{2} X^T Q X + R^T X + C \\ \text{S.T.} \quad & AX = B \end{aligned}$$

- **Optimality condition for quadratic programming**

$$QX^* + R + A^T V = 0$$
$$AX^* = B$$

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} X^* \\ V \end{bmatrix} = \begin{bmatrix} -R \\ B \end{bmatrix}$$

Linear Equality Constrained Nonlinear Programming

$$\begin{aligned} \min_X \quad & \frac{1}{2} X^T Q X + R^T X + C \\ \text{S.T.} \quad & AX = B \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} X^* \\ V \end{bmatrix} = \begin{bmatrix} -R \\ B \end{bmatrix}$$

- **Minimize nonlinear function f(X) given linear constraint AX = B**

$$\begin{aligned} \min_X \quad & f(X) \\ \text{S.T.} \quad & AX = B \end{aligned}$$

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

$$f[x^{(k+1)}] \simeq f[x^{(k)}] + \nabla f[x^{(k)}] \cdot \Delta x^{(k)} + \frac{1}{2} \Delta x^{(k)} \cdot \nabla^2 f[x^{(k)}] \cdot \Delta x^{(k)}$$

$$A \Delta x^{(k)} = A \cdot [x^{(k+1)} - x^{(k)}] = A x^{(k+1)} - A x^{(k)} = B - A \cdot x^{(k)}$$

Linear Equality Constrained Nonlinear Programming

$$\min_X \ \frac{1}{2}X^T Q X + R^T X + C \quad \text{S.T.} \quad AX = B \quad \Rightarrow \quad \begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} X^* \\ V \end{bmatrix} = \begin{bmatrix} -R \\ B \end{bmatrix}$$

$$\min_{\Delta X} \ f\left[X^{(k+1)}\right] \approx \frac{1}{2} \cdot \Delta X^T \cdot \nabla^2 f\left[X^{(k)}\right] \cdot \Delta X + \nabla f\left[X^{(k)}\right]^T \cdot \Delta X + f\left[X^{(k)}\right]$$

$$A \cdot \Delta X = B - AX^{(k)}$$

$$\begin{bmatrix} \nabla^2 f[x^{(k)}] & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{(k)} \\ V \end{bmatrix} = \begin{bmatrix} -\nabla f[x^{(k)}] \\ B - Ax^{(k)} \end{bmatrix}$$

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

Linear Equality Constrained Nonlinear Programming

$$\min_X \ f(X) \quad \text{S.T.} \quad AX = B \quad \Rightarrow \quad \begin{bmatrix} \nabla^2 f\left[X^{(k)}\right] & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta X \\ V \end{bmatrix} = \begin{bmatrix} -\nabla f\left[X^{(k)}\right] \\ B - AX^{(k)} \end{bmatrix}$$

- **If $X^{(k)}$ is a feasible solution** $Ax^{(k)} = B$
  - We can start from an initial solution $X^{(0)}$ that is feasible
  - Even if $X^{(0)}$ is not feasible, $X^{(1)}$ is feasible after one iteration

$$B - Ax^{(k)} = 0$$

$$\begin{bmatrix} \nabla^2 f[x^{(k)}] & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ V \end{bmatrix} = \begin{bmatrix} -\nabla f[x^{(k)}] \\ 0 \end{bmatrix}$$

A Simple Example

$$\min_{x_1,x_2} \quad x_1^4 + x_2^4$$
$$\text{S.T.} \quad x_1 + x_2 = 1$$

$$\min_{X} \quad f(X)$$
$$\text{S.T.} \quad AX = B$$

$$X^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(Feasible solution)

$$f(X) = x_1^4 + x_2^4$$
$$A = \begin{bmatrix} 1 & 1 \end{bmatrix}$$
$$B = 1$$

$$\nabla^2 f(x) = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 12x_2^2 \end{bmatrix}$$

$$\nabla f(x) = \begin{bmatrix} 4x_1^3 \\ 4x_2^3 \end{bmatrix}$$

A Simple Example

$$\min_{x_1,x_2} \quad x_1^4 + x_2^4$$
$$\text{S.T.} \quad x_1 + x_2 = 1$$

$$\begin{bmatrix} \nabla^2 f\left[X^{(k)}\right] & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta X \\ V \end{bmatrix} = \begin{bmatrix} -\nabla f\left[X^{(k)}\right] \\ 0 \end{bmatrix}$$

$$X^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\nabla^2 f[x^{(0)}] = \begin{bmatrix} 12 & 0 \\ 0 & 12 \end{bmatrix} \quad \nabla f[x^{(k)}] = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 12 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta X \\ V \end{bmatrix} = \begin{bmatrix} -4 \\ 0 \\ 0 \end{bmatrix}$$

$$\nabla^2 f(X) = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 12x_2^2 \end{bmatrix}$$

$$\nabla f(X) = \begin{bmatrix} 4x_1^3 \\ 4x_2^3 \end{bmatrix}$$

$$\Delta X = \begin{bmatrix} -0.33 \\ 0.30 \end{bmatrix}$$

$$X^{(1)} = X^{(0)} + \Delta X = \begin{bmatrix} 0.67 \\ 0.33 \end{bmatrix}$$

A Simple Example

$$\min_{x_1,x_2} \quad x_1^4 + x_2^4$$
$$\text{S.T.} \quad x_1 + x_2 = 1$$

$$X^{(1)} = \begin{bmatrix} 0.67 \\ 0.33 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\nabla^2 f(X) = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 12x_2^2 \end{bmatrix}$$

$$\nabla f(X) = \begin{bmatrix} 4x_1^3 \\ 4x_2^3 \end{bmatrix}$$

$$\begin{bmatrix} \nabla^2 f\left[X^{(k)}\right] & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta X \\ V \end{bmatrix} = \begin{bmatrix} -\nabla f\left[X^{(k)}\right] \\ 0 \end{bmatrix}$$

$$\nabla^2 f\left[X^{(1)}\right] = \begin{bmatrix} 5.39 & 0 \\ 0 & 1.31 \end{bmatrix} \quad \nabla f\left[X^{(1)}\right] = \begin{bmatrix} 1.20 \\ 0.14 \end{bmatrix}$$

$$\begin{bmatrix} 5.39 & 0 & 1 \\ 0 & 1.31 & 1 \\ 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta X \\ V \end{bmatrix} = \begin{bmatrix} -1.20 \\ -0.14 \\ 0 \end{bmatrix}$$

$$\Delta X = \begin{bmatrix} -0.16 \\ 0.16 \end{bmatrix} \quad X^{(2)} = \begin{bmatrix} 0.51 \\ 0.49 \end{bmatrix}$$

Linear Equality Constrained Nonlinear Programming

- **Linear equality constraints can be efficiently handled by subspace reduction or Lagrange multiplier**
- **Nonlinear equality constraints and inequality constraints must be handled by a different algorithm**
  - Interior point method (also referred to as barrier method)
  - More details in future lectures

Summary

- **Constrained optimization**
  - Linear equality constraint
  - Lagrange multiplier