

# Interior-Point Methods

$$\begin{aligned} \min & f(x) \\ \text{s.t.} & g(x) \geq 0 \end{aligned}$$

"infeasible" interior-point methods  
 $x^{(k)}$  can be infeasible for the original problem.

$$\begin{aligned} \min & f(x) & f: \mathbb{R}^n &\rightarrow \mathbb{R} \\ \text{s.t.} & g(x) - w = 0 & g: \mathbb{R}^n &\rightarrow \mathbb{R}^m \\ & w \geq 0 & & \end{aligned}$$

Barrier subproblem:

$$\begin{aligned} \min & f(x) - \mu \sum_{i=1}^m \log(w_i) \\ \text{s.t.} & g(x) - w = 0 \end{aligned}$$

Instead of  $g(x^k) > 0$  at each iteration  $k$ , we only need  $w^k > 0$  at each iteration  $k$ .

$$\mathcal{L}(x, w, \lambda) = f(x) - \mu \sum_{i=1}^m \log(w_i) - \lambda^T (g(x) - w)$$

$$\frac{\partial \mathcal{L}}{\partial x} = \nabla f(x) - A(x)^T \lambda = 0 \quad (A(x) \text{ is the Jacobian})$$

$$\frac{\partial \mathcal{L}}{\partial w_i} = -\frac{\mu}{w_i} + \lambda_i = 0 \quad \left( \begin{array}{l} -\mu W^{-1} e + \lambda = 0, \\ \text{where } W = \begin{bmatrix} w_1 & w_2 & \dots & w_m \end{bmatrix}, e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \end{array} \right)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = g(x) - w = 0$$

$$-\mu W^{-1} e + \lambda = 0$$

$$\Rightarrow \lambda = \mu W^{-1} e$$

$$\Rightarrow W \lambda = \mu e$$

$$\Rightarrow W \Delta e = \mu e \quad \left( \Delta = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{bmatrix} \right)$$

$$\Rightarrow \Delta W e = \mu e \quad \Delta W = \mu I$$

$$\nabla f(x) - A(x)^T \lambda = 0$$

$$W \Delta e - \mu e = 0 \quad (\mu\text{-perturbed complementarity conditions})$$

$$g(x) - w = 0$$

$$\Delta x = - \frac{f(x)}{f'(x)} \quad \text{Newton's Method}$$

$$f'(x) \Delta x \Rightarrow f(x)$$

$$\begin{bmatrix} H(x, \lambda) & \emptyset & -A(x)^T \\ \emptyset & \Delta & W \\ A(x) & -I & \emptyset \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A(x)^T \lambda \\ W \Delta e - \mu e \\ g(x) - w \end{bmatrix}$$

$$H(x, \lambda) = \nabla^2 f(x) - \sum_{i=1}^m \lambda_i \nabla^2 g_i(x)$$

Hessian of the Lagrangian

$$\begin{bmatrix} -H(x, \lambda) & \emptyset & A(x)^T \\ \emptyset & -\Delta W^{-1} & -I \\ A(x) & -I & \emptyset \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta w \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} -\nabla f(x) + A(x)^T \lambda \\ \mu W^{-1} e - \lambda \\ g(x) - w \end{bmatrix}$$

sparse and symmetric

$$\Delta w = -W \Delta^{-1} (-\mu W^{-1} e + \lambda + \Delta \lambda)$$

$$(*) \begin{bmatrix} -H(x, y) & A(x)^T \\ A(x, y) & W \Delta^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + A(x)^T \lambda \\ g(x) - w - \mu \Delta^{-1} e + w \end{bmatrix}$$

has the same # of nonzeros

- use Newton's Method with (\*) until optimal solution to the barrier subproblem is solved, then update  $\mu$ . (or close to optimality within  $\epsilon_\mu$ )

• in practice,  $\mu$  is updated at each iteration (whenever possible)

KNITRO, IPOPT:  $\mu$  updated infrequently at start, then every iteration when  $m$  is below a threshold

LOQO, PenNLP:  $\mu$  updated at each iteration,  
not guaranteed to decrease unless  
close to optimal solution

$$\mu = r \gamma \left( \frac{w^T \lambda}{m} \right) \quad \gamma(\cdot) \text{ a fn of complementarity}$$

at the end of every iteration:

Knitro, IPOPT: check if  $r \gamma \left( \frac{w^T \lambda}{m} \right) < \mu$   
if so update, otherwise keep  
old  $\mu$ .

LOQO, PenNLP: update  $\mu$  with  $r \gamma \left( \frac{w^T \lambda}{m} \right)$

Step 0: Start at  $x^{(0)}, w^{(0)}, \lambda^{(0)}$ . Let  $\mu = r \gamma \left( \frac{w^{(0)T} \lambda^{(0)}}{m} \right)$   
 $k=0$

Step 1: Check if  $x^{(k)}, w^{(k)}, \lambda^{(k)}$  satisfy  
first-order conditions for  $\mathcal{L}$  with  
 $\mu < \epsilon$ .

If yes, stop.

→ Step 2: Solve (\*) to obtain  $\Delta x, \Delta w, \Delta \lambda$ .

Step 3: Compute a steplength  $\alpha$ .

Step 4: Let  $x^{k+1} = x^k + \alpha \Delta x^k, w^{k+1} = w^k + \alpha \Delta w^k, \lambda^{k+1} = \lambda^k + \alpha \Delta \lambda^k$

Let  $\mu = r \gamma \left( \frac{w^k \lambda^k}{m} \right)$  as allowed.

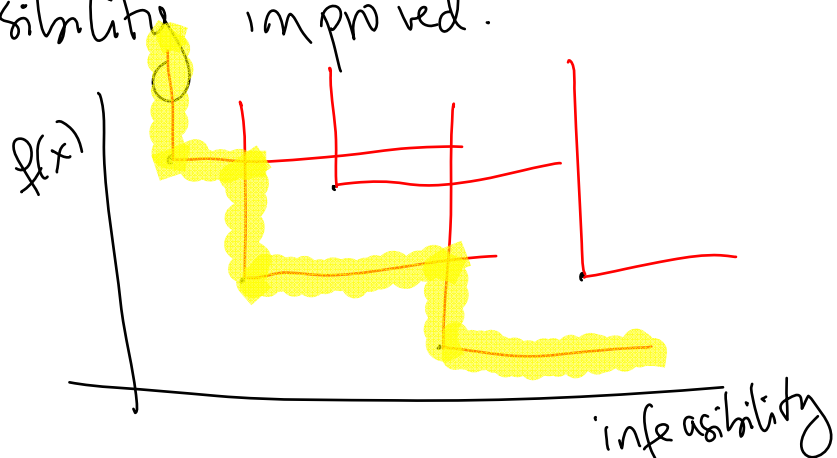
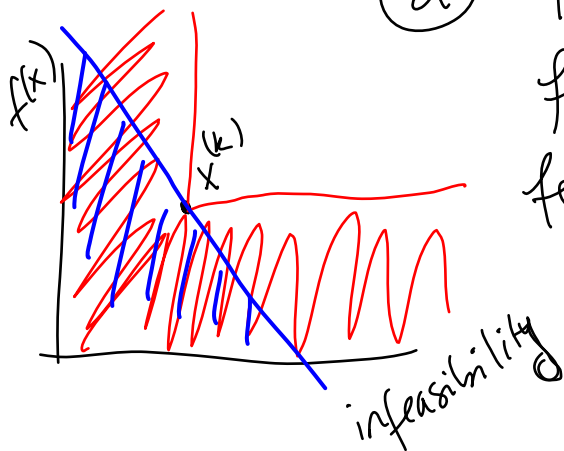
Let  $k = k+1$ .

Go to Step 1.

How do we compute  $\alpha$ ?

① Merit function: Decrease a linear combination of optimality and infeasibility if  $x^k + \alpha \Delta x$  gives a lower value of  $f(x) - \beta \sum_{i=1}^m \min(g_i(x), 0)$  than  $x^k$ , use  $\alpha$ . (perform bisection to find suitable  $\alpha$ ).

② Filter method: Break up merit function: either optimality or feasibility improved.



To find  $\Delta x, \Delta w, \Delta \lambda$ , we need

- $H(x, \lambda) \succ 0$  (positive definite)
- and •  $W \Lambda^{-1} \succ 0$

Starting with  $w > 0$  and keep  $w > 0$  for the barrier terms.

$\Rightarrow$  Start  $\lambda > 0$  and keep  $\lambda > 0$  for  $W \Lambda^{-1} \succ 0$  at each iteration

(take  $\alpha$  so that these holds)

*gives control over the dual problem*

## Primal-Dual Method

If  $H(x, \lambda) \not\succeq 0$ , use  $H(x, \lambda) + sI$  instead (find  $s$  such that  $H(x, \lambda) + sI \succ 0$ )

instead of Newton Step, look for something closer to the steepest descent direction.

Factor  $\begin{bmatrix} -H & A^T \\ A & W\Lambda^{-1} \end{bmatrix} = LDL^T$

$$D = \begin{bmatrix} d_1 & & & & & & \\ & d_2 & & & & & \\ & & \ddots & & & & \\ & & & d_n & & & \\ & & & & d_{n+1} & & \\ & & & & & \ddots & \\ & & & & & & d_{n+m} \end{bmatrix} \left. \begin{array}{l} \} < 0 \\ \} > 0 \end{array} \right\}$$

$$\text{Min } f(x) - \mu \sum_{i=1}^m \log(w_i)$$

$$\text{s.t. } g(x) - w = 0$$

$$\boxed{w \geq 0}$$

$$\text{min } f(x) - \mu \sum_{i=1}^m \log(w_i + \xi_i) - \mu \sum_{i=1}^m \log(\xi_i)$$

$$\text{s.t. } g(x) - w = 0$$

$$\left. \begin{array}{l} w + \xi \geq 0 \\ \xi \geq 0 \end{array} \right\}$$

$$\begin{array}{l}
 x_1, x_2 = 0 \\
 x_1 \geq 0 \quad w_2 \\
 x_2 \geq 0 \quad w_3
 \end{array}
 \quad
 \begin{bmatrix}
 x_2 & x_1 & 0 & 0 \\
 1 & 0 & w_2/\lambda_2 & 0 \\
 0 & 1 & 0 & w_3/\lambda_3
 \end{bmatrix}$$

---


$$\lambda_2 \leq p_2, \quad \lambda_3 \leq p_3$$


---

Sequential Quadratic Programming (SQP)

$$\begin{bmatrix}
 -H(x, \lambda) & A(x)^T \\
 A(x) & W\Delta^{-1}
 \end{bmatrix}
 \begin{bmatrix}
 \Delta x \\
 \Delta \lambda
 \end{bmatrix}
 = -
 \begin{bmatrix}
 -\nabla f(x) + A(x)^T \lambda \\
 g(x) - \mu \Delta^{-1} e
 \end{bmatrix}$$

2<sup>nd</sup>-order Taylor polynomial (quadratic)

Linearization of the constraints

$$\min f(x)$$

$$\text{s.t. } g(x) \geq 0$$

Choose a set of active constraints,  $B$ .

$$\Rightarrow g_b(x) = 0 \quad \forall b \in B.$$

Solve

$$\min \frac{1}{2} \Delta x^T D^2 L(x^k) \Delta x + \nabla L(x^k) \Delta x$$

$$\text{s.t. } \nabla g_b(x^k) \Delta x = 0, b \in B$$

Get  $\Delta x$ , find  $\alpha$ . Let  $x^{k+1} = x^k + \alpha \Delta x$

Calculate  $\lambda$  using first-order condition.

If  $\lambda_b < 0$ , then remove  $b$  from the active set.

If  $\lambda_n > 0$  ( $n \notin B$ ), then  $n$  to the active set.

SNOPT, filter SQP

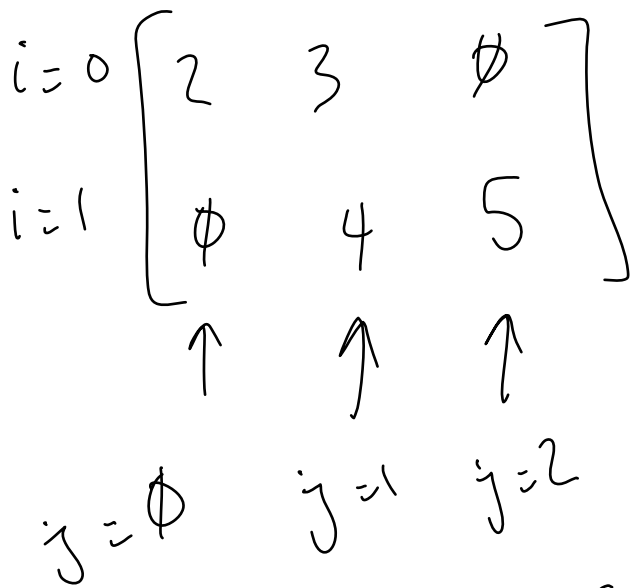
$$A = \begin{bmatrix} x & & & & \\ & x & & & \\ & & x & & \\ & & & x & \\ & & & & x \\ & & x & & \\ & & & x & \\ & & & & x \end{bmatrix}$$

$k A$  = start point of column  $k$  in  $i A$  and  $A$

$i A$  = row #s of the nonzeros

$A$  = actual values of the nonzeros

$2 \times (\# \text{ of nonzeros}) + (\# \text{ of columns})$



$$kA(0) = 0$$

$$iA(0) = \emptyset$$

$$A(0) = 2$$

$$kA(1) = 1$$

$$iA(1) = \emptyset$$

$$A(1) = 3$$

$$iA(2) = 1$$

$$A(2) = 4$$

$$kA(2) = 3$$

$$iA(3) = 1$$

$$A(3) = 5$$

$$kA(3) = 4$$

for (j=0; j ≤ 3; j++) {

for (k=kA(j); k < kA(j+1); k++) {

print (row=iA(k), col=j,  
value=A(k));

}

}

# of nonzeros

$$2 \times n^2 + (n+1) \leq mn$$

$$n^2 \leq \frac{mn - (n+1)}{2}$$

Trust-region method: include a constraint in each subproblem of the form  $\| \Delta x \| \leq d$  ← trust-region radius

$$\begin{aligned} \max \quad & C \cdot X \\ \text{s.t.} \quad & A \cdot X \leq b \end{aligned}$$

replace with  $vv^T$

$$\text{rank}(X) = 1$$

$$X \succeq 0$$

$$X \in \mathbb{R}^{n \times n}$$

QCAP

$$X = VV^T$$

$$X = \underset{\substack{\uparrow \\ n \times 1}}{v} \underset{\substack{\uparrow \\ 1 \times n}}{v^T}$$

reduce to  
finding  $v$