


# Computational Optimization

Sequential Quadratic  
Programming  
NW Chapter 18



# Sequential Quadratic Programming (SQP)

Basic Idea:

QP with constraints are easy. For any guess of active constraints, just have to solve system of equations.

So why not solve general problem as a series of constrained QPs.

**Which QP should be used?**





# Use KKT


## ● Problem

$$\begin{aligned} \min \quad & f(x) \\ (NLP) \quad & s.t. \quad g_i(x) = 0 \quad i \in E \end{aligned}$$

## ● Use Lagrangian

$$L(x, \lambda) = f(x) - \lambda' g(x)$$

$$\nabla_x L(x, \lambda) = \nabla f(x) - \lambda' \nabla g(x) = 0$$

$$\nabla_\lambda L(x, \lambda) = g(x) = 0$$


# Solve for primal and dual using Newton's Method

Newton step

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} + \begin{bmatrix} p_k \\ v_k \end{bmatrix}$$

where

$$\nabla^2 L(x_k, \lambda_k) \begin{bmatrix} p_k \\ v_k \end{bmatrix} = -\nabla L(x_k, \lambda_k)$$

$\Downarrow$

$$\begin{bmatrix} \nabla_{xx}^2 L(x_k, \lambda_k) & -\nabla g(x_k) \\ -\nabla g(x_k)' & 0 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x_k, \lambda_k) \\ g(x_k) \end{bmatrix}$$

# SQP

- Equations are first order KKT of

$$\begin{aligned} \min_p \quad & \frac{1}{2} p' \left( \nabla_{xx}^2 L(x_k, \lambda_k) \right) p + p' \left( \nabla_x L(x_x, \lambda_k) \right) \\ \text{s.t.} \quad & \nabla_x g(x_x)' p + g(x_k) = 0 \end{aligned}$$

*NOTE* :  $\nu_k$  is the Lagrangian multiplier for constraints  
Lagrangian of QP approximation

$$\hat{L}(p, \nu) = \frac{1}{2} p' \left( \nabla_{xx}^2 L(x_k, \lambda_k) \right) p + p' \left( \nabla_x L(x_x, \lambda_k) \right) - \nu' \left( \nabla_x g(x_x)' p + g(x_k) \right)$$

*KKT*

$$\nabla_p \hat{L}(p, \nu) = \left( \nabla_{xx}^2 L(x_k, \lambda_k) \right) p + \nabla_x L(x_x, \lambda_k) - \nabla_x g(x_x) \nu = 0$$

$$\nabla_x g(x_x)' p + g(x_k) = 0$$



# Local SQP Algorithm

- First shot at an algorithm while not done
  - Solve QP subproblem for  $(p, v)$
  - Add to iterate.

Like any Newton's method – only works if close enough to solution



# Local SQP Algorithm

Try on (you verify solution)

$$\min f(x_1, x_2) = e^{3x_1+4x_2} \quad s.t. \quad g(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0$$

$$\text{Solution } x^* = \left[-\frac{3}{5}, -\frac{4}{5}\right]' \quad \lambda^* = -\frac{5}{2} e^{-5}$$

$$\text{Start } x_0 = [-.7, -.7]' \quad \lambda_0 = -.01$$

$$\nabla f(x_1, x_2) = e^{3x_1+4x_2} \begin{bmatrix} 3 \\ 4 \end{bmatrix} \quad \nabla^2 f(x_1, x_2) = e^{3x_1+4x_2} \begin{bmatrix} 9 & 12 \\ 12 & 16 \end{bmatrix}$$

$$g(x_1, x_2) = x_1^2 + x_2^2 - 1 = -.02$$

$$\nabla g(x_1, x_2) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 1.4 \end{bmatrix} \quad \nabla^2 g(x_1, x_2) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\nabla_x L = \nabla f - \lambda \nabla g = \begin{bmatrix} .008340 \\ .015786 \end{bmatrix} \quad \nabla_{xx}^2 L = \nabla^2 f - \lambda \nabla^2 g = \begin{bmatrix} .08702 & .08936 \\ .08936 & .13915 \end{bmatrix}$$

# First iteration

## Solve

$$\begin{bmatrix} \nabla_{xx}^2 L(x_k, \lambda_k) & -\nabla g(x_k) \\ -\nabla g(x_k)' & 0 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x_k, \lambda_k) \\ g(x_k) \end{bmatrix}$$



$$\begin{bmatrix} .08720 & .08936 & 1.4 \\ .08936 & .13915 & 1.4 \\ 1.4 & 1.4 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ v \end{bmatrix} = \begin{bmatrix} -.008340 \\ -.015786 \\ -.020000 \end{bmatrix} \quad \begin{bmatrix} p_1 \\ p_2 \\ v \end{bmatrix} = \begin{bmatrix} .14196 \\ -.15624 \\ -.004808 \end{bmatrix}$$






# New point

Yields

$$x_1 = x_0 + p_0 = \begin{bmatrix} -.55804 \\ -.85624 \end{bmatrix}$$

$$\lambda_1 = \lambda_0 + \nu_0 = -.014808$$


k	x <sub>k</sub>		λ <sub>k</sub>	del L	g
0	-.7000	-.7000	-.0100	2e-2	2e-2
1	-.5580	-.8562	-.0148	2e-3	5e-2
2	-.6077	-.7978	-.0168	2e-4	6e-3
3	-.5999	-.8001	-.0168	2e-6	7e-5
4	-.6000	-.8000	-.0168	2e-9	3e-8
5	-.6000	-.8000	-.0168	2e-16	4e-15

# Inequalities

- Local form easily extended to inequalities.
- Just solve this QP at each iteration

$$\min_p \quad \frac{1}{2} p' \left( \nabla_{xx}^2 L(x_k, \lambda_k) \right) p + p' \left( \nabla_x L(x_k, \lambda_k) \right)$$

$$s.t. \quad \begin{aligned} \nabla_x g_i(x_k)' p + g_i(x_k) &= 0 \quad i \in E \\ \nabla_x g_i(x_k)' p + g_i(x_k) &\geq 0 \quad i \in I \end{aligned}$$

*NOTE* :  $v_k$  is the Lagrangian multiplier for constraints



# But....

- Local form not used because of classic Newton flaws
    - May not converge
    - Expensive
    - Linearization may be infeasible
  - Similar solutions
    - Add linesearch
    - Modified factorization to force hessian pd
    - Quasi-Newton Approximations
- 




# Merit Functions

- Line search needs to worry about objective value and feasibility
- Use linesearch but with merit function to force toward feasibility

Inexact

$$M(x) = f(x) + \rho g(x)' g(x) = f(x) + \rho \sum_i g_i(x)^2$$

Exact

$$M(x) = f(x) + \rho \|g(x)\|_1 = f(x) + \rho \sum_i |g_i(x)|$$




# Plus Usual Tricks

- Use Modify Cholesky to insure descent directions
  - Use Quasi Newton approximation of Hessian of Lagrangian
  - Can add linearization of  $g(x)$  for inequality constraints too.
  - Change things a bit (see damped Newton)
- 

# General Problem Case

• Same things works for inequalities

• Use QP

$$\min_p \quad \frac{1}{2} p' \left( \nabla_{xx}^2 L(x_k, \lambda_k) \right) p + p' \left( \nabla_x L(x_k, \lambda_k) \right)$$

$$s.t. \quad \nabla_x g_1(x_k)' p + g_1(x_k) = 0$$

$$\nabla_x g_2(x_k)' p + g_2(x_k) \geq 0$$

• Merit function

$$M(x) = f(x) + \rho \left( \sum_i |g_{1i}(x)| + \sum_i \max(0, -g_{2i}(x)) \right)$$



# Devil in the details

- Check out Practical SQP algorithm  
Algorithm 18.3 Practical Line Search  
Algorithm



# Trust Region Works Great

- We only trust approximation locally so limit step to this region by adding constraint to QP

$$\min_p \quad \frac{1}{2} p' \left( \nabla_{xx}^2 L(x_k, \lambda_k) \right) p + p' \left( \nabla_x L(x_k, \lambda_k) \right)$$

$$s.t. \quad \nabla_x g(x_k)' p + g(x_k) = 0$$

$$\|p\| \leq \Delta_k$$

Trust region

No stepsize needed!



# How to pick trust region?


- Give it a try by solving the QP

Things go better than expected (great decrease), increase trust region,

Things go as expected (okay decrease), keep trust region the same,

Things go worse than expected (insufficient decrease), shrink trust region and try again.

Use  $I_2$  merit function to decide if things are okay.





# Devil in the details

- Check out Practical SQP algorithm


Algorithm 18.4 Byrd et al Trust Region  
Line Search Algorithm


Also might want to add a Filter





# SQP Methods

- Good when number of active constraints is close to number of variables
  - Require few function evaluations (compared to augmented Lagrangian)
  - Robust on badly scaled problems
  - Very successful and widely used in practice (SNOPT and FILTER)
  - Can suffer from Maratos effect (merit eliminates good steps)
- 



# NLP Family of Algorithms

Basic Method	Sequential Quadratic Programming	Sequential Linear Prog	Augmented Lagrangian	Projection or Reduced Gradient	
Directions	Steepest Descent	Newton	Quasi Newton	Conjugate Gradient	
Space	Direct	Null	Range		
Constraints	Active Set	Barrier	Penalty		
Step Size	Line Search	Trust Region			

