

# Computational Optimization

Conjugate Gradient +  
Termination Criteria 2/15

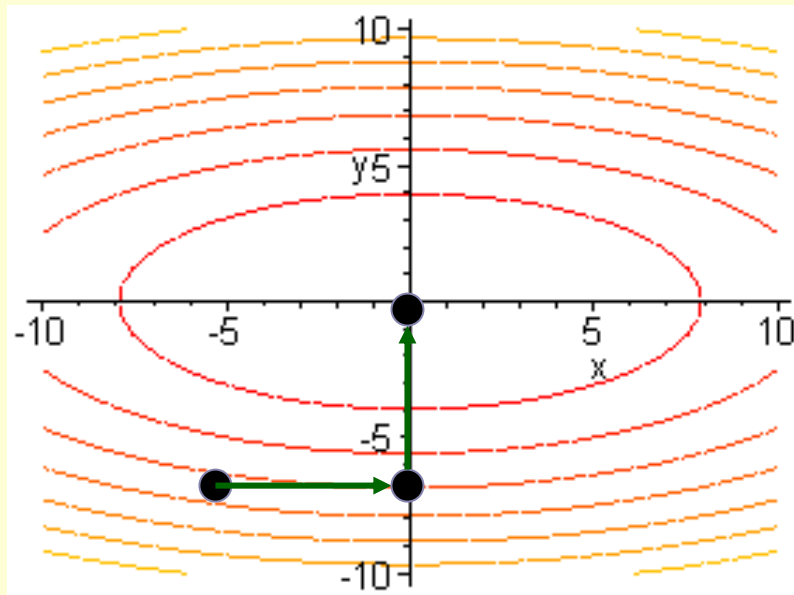


# Conjugate Gradient (CG)

- Method for minimizing quadratic function
  - Low storage method  
CG only stores vector information
  - CG superlinear convergence for nice problems or when properly scaled
- 

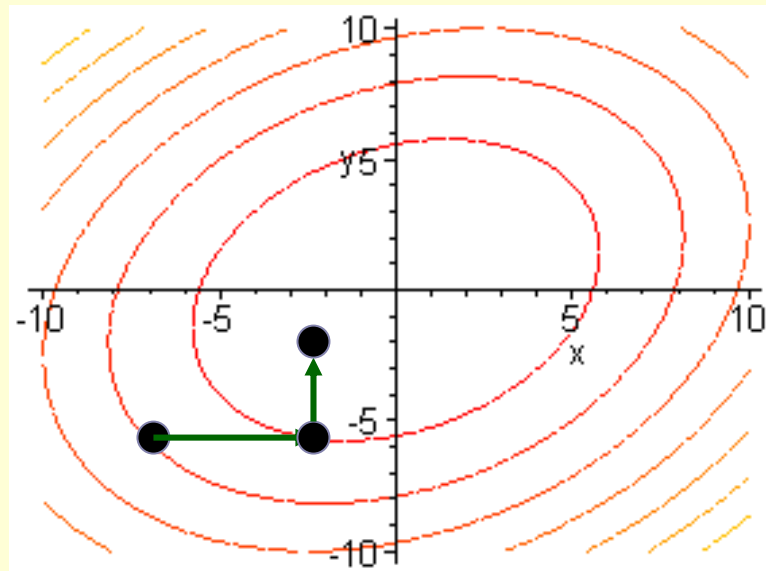
# Decompose Problem

- Consider  $\min x^2 + 4y^2$
- Same as solving  $\min x^2 + \min 4y^2$



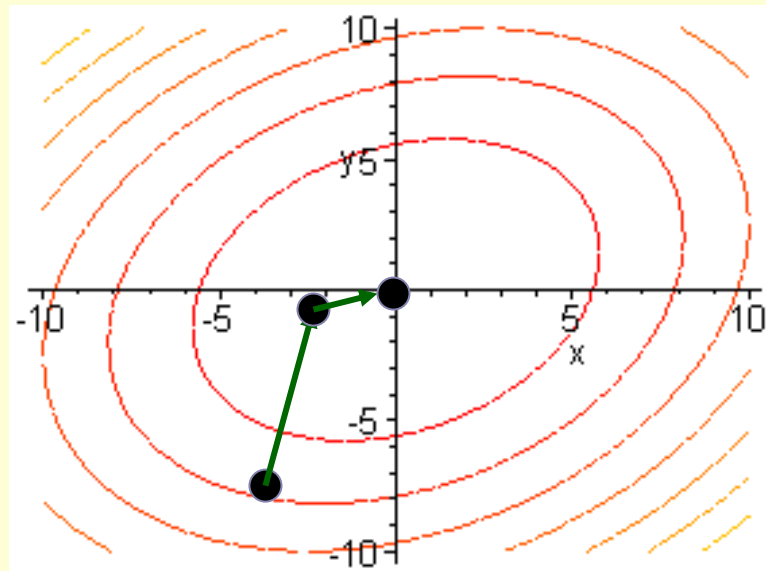
# Decompose Problem

- Consider  $\min x^2 + y^2 - .5 * x * y$
- Can't decompose by  $x$  and  $y$

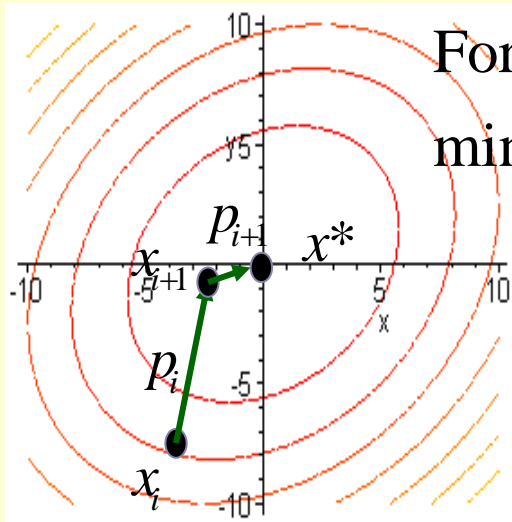


# Decompose Space

- Could solve in two steps if we take in account Hessian  $Q$



# How to pick directions?



For  $\min 1/2x'Qx-b'x$

$$\min_{\alpha} f(x_i + \alpha p_i) \Rightarrow \nabla f(x_i + \alpha p_i)' p_i = 0$$

$$\Rightarrow p_i'(Q(x_i + \alpha p_i) - b) = 0$$

$$\Rightarrow p_i'(Qx_{i+1} - b) = 0 \text{ since } x_{i+1} = x_i + \alpha p_i$$

$$\Rightarrow p_i'(Qx^* - b) = 0 \text{ since } x^* \text{ is solution}$$

Subtract

$$p_i'Q(x_{i+1} - x^*) = 0$$

$$\Rightarrow p_i'Qp_{i+1} = 0$$

**Say  $p_i$  and  $p_{i+1}$  are Q-Conjugate**

# Q-Conjugate is good

- Say we know a set of  $n$  conjugate vectors,  $\mathbf{p}_1, \dots, \mathbf{p}_n$ . They are lin. indep.

$$\text{Let } y = \sum_{i=1}^n \alpha_i p_i$$

$$f(y) = \frac{1}{2} y' Q y - b' y$$

$$= \frac{1}{2} \left( \sum_i \alpha_i p_i \right)' Q \left( \sum_j \alpha_j p_j \right) - b' \left( \sum_j \alpha_j p_j \right)$$

$$= \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j p_i' Q p_j - \sum_j \alpha_j b' p_j$$

$$= \frac{1}{2} \sum_j \alpha_j^2 p_j' Q p_j - \sum_j \alpha_j b' p_j$$

$$\Rightarrow \min_y f(y) = \min_{\alpha} f\left(\sum_{i=1}^n \alpha_i p_i\right)$$

$$= \sum_j \left[ \min_{\alpha_j} \frac{1}{2} \alpha_j^2 p_j' Q p_j - \alpha_j b' p_j \right]$$

# In class exercise

Find the optimal solution of

$$\min_{\alpha_j} \frac{1}{2} \alpha_j^2 p_j' Q p_j - \alpha_j b' p_j$$

The optimal solution satisfies

$$g'(\alpha_j) = \alpha_j p_j' Q p_j - b' p_j = 0$$

so we know

$$\hat{\alpha}_j = \frac{b' p_j}{p_j' Q p_j}$$

# Conjugate Gradient

## • Equivalent problem

$$\begin{aligned} \min_y f(y) &= \min_{\alpha} f\left(\sum_{i=1}^n \alpha_i p_i\right) \\ &= \sum_j \left[ \min_{\alpha_j} \frac{1}{2} \alpha_j^2 p_j' Q p_j - \alpha_j b' p_j \right] \end{aligned}$$

• Means that can solve problem in at most  $n$  subproblems for  $n$  dim problems!

# Conjugate Gradient

## preliminary - page 108 NW

• Set  $r_0 := Qx_0 - b$  (negative grad)

$p_0 := -r_0$ ,  $k := 0$

• While ( $\|r_k\| > \varepsilon$ )

$$\alpha_k := \frac{r_k' p_k}{p_k' Q p_k}$$

$$x_{k+1} := x_k + \alpha_k p_k$$

$$r_{k+1} := Qx_{k+1} - b$$

$$\beta_{k+1} := \frac{r_{k+1}' Q p_k}{p_k' Q p_k}$$

$$p_{k+1} := -r_{k+1} + \beta_{k+1} p_k$$

$$k := k + 1$$



# Claims

- Properties

$$r_i \cdot r_j = 0 \text{ for } i \neq j$$

$$r_i \cdot p_j = 0 \text{ for } i > j$$

$$p_i \cdot Qp_j = 0 \text{ for } i > j$$

- Therefore converges in at most  $n$  steps





# Intuition

● At each iteration, do exact linesearch

$$\alpha_{k+1} = \arg \min_{\alpha} \frac{1}{2} (x_k + \alpha p_k)' Q (x_k + \alpha p_k) - b' (x_k + \alpha p_k)$$

so

$$0 = p_k' [Q (x_k + \alpha_{k+1} p_k) - b] = p_k' r_{k+1} = 0$$



# Intuition

- Choice of  $\beta$  enforces conjugacy

$$\begin{aligned} p_2' Q p_1 &= [-r_2 + \beta_2 p_1]' Q p_1 \\ &= -r_2 Q p_1 + \beta_2 p_1' Q p_1 \\ &= -r_2 Q p_1 + \frac{r_2' Q p_1}{p_1' Q p_1} p_1' Q p_1 = 0 \end{aligned}$$

# Make CG more efficient 1

NEW

$$\alpha_k := \frac{r_k' r_k}{p_k' Q p_k}$$

OLD

$$\alpha_k := \frac{r_k' p_k}{p_k' Q p_k}$$

$$r_k' p_k = r_k' (r_k + \beta_{k-1} p_{k-1}) = r_k' r_k$$

+ Don't need to store  $r$  just  $\|r\|$  (already computed)

# Make CG more efficient 2

NEW

$$\beta_{k+1} := \frac{r_{k+1}' r_{k+1}}{r_k' r_k}$$

Since  $\alpha_k Q p_{k-1} = (r_k - r_{k+1})$

$$p_k = r_k + \beta_k p_{k-1}$$

$$r_k p_i = 0$$

OLD

$$\beta_{k+1} := \frac{r_{k+1}' Q p_k}{p_k' Q p_k}$$

See NW for details  
+ just need one divide

# Conjugate Gradient Algorithm

## page 112 NW

• Set  $r_0 := Qx_0 - b$  (negative grad)

$p_0 := -r_0$ ,  $k := 0$

• While ( $\|r_k\| > \varepsilon$ )

$$\alpha_k := \frac{r_k' r_k}{p_k' Q p_k}$$

$$x_{k+1} := x_k + \alpha_k p_k$$

$$r_{k+1} := r_k + \alpha_k Q p_k$$

$$\beta_{k+1} := \frac{r_{k+1}' r_{k+1}}{r_k' r_k}$$

$$p_{k+1} := -r_{k+1} + \beta_{k+1} p_k$$

$$k := k + 1$$

# Convergence Rate

• Can show

$$\frac{\|x_{k+1} - x^*\|_Q}{\|x_k - x^*\|_Q} \leq \frac{\sqrt{\text{cond}(Q) - 1}}{\sqrt{\text{cond}(Q) + 1}} \quad \text{where} \quad \|y\|_Q^2 = \frac{1}{2} y' Q y$$

- Effected by conditioning.
- Preconditioning used to make faster
- Still at most  $n$  iterations for  $n$  dimensions but use for very large  $n$ .

# Nonlinear Conjugate Gradient

● Set  $x_0=0$ ,  $b_0=0$      $r_0 = \nabla f(x_0)$  (gradient)

$p_0=0$ ,  $\alpha_0=0$

● For  $i=0, \dots, 1$

● If optimal  $\|\nabla f(x_i)\| < \varepsilon$  , then stop

● For  $i>0$

$$\beta_i := \frac{\nabla f(x_i)' \nabla f(x_i)}{\nabla f(x_{i-1})' \nabla f(x_{i-1})} \quad \text{stepsize}$$

$$p_i := -\nabla f(x_i) + \beta_i p_{i-1}$$

use linesearch for  $\alpha_i$

$$x_{i+1} := x_i + \alpha_i p_i$$

# Other Possible Updates

- Equivalent for quadratic case
- But Polak Ribiere is best in practice

$$\beta_i := \frac{\nabla f(x_i)' \nabla f(x_i)}{\nabla f(x_{i-1})' \nabla f(x_{i-1})} \quad \text{Fletcher Reeves}$$

$$\beta_i := \frac{y_{i-1}' \nabla f(x_i)}{\nabla f(x_{i-1})' \nabla f(x_{i-1})} \quad \text{Polak Ribiere (Best)}$$

$$\beta_i := \frac{\nabla f(x_i)' \nabla f(x_i)}{y_{i-1}' p_{i-1}} \quad \text{Heslenes Stiefel}$$

where  $y_{i-1} = \nabla f(x_i) - \nabla f(x_{i-1})$



# Comments

- Only need to store 3 to 5 vectors.
  - Only dot products needed. Can be used for huge problems.
  - Easy to implement
  - Preconditioning improves convergence
  - Needs exact or close to exact stepsize
  - Best Methods are truncated Newton or Limited memory BFGS – these would make good projects
- 



# Termination Rules

- Computer uses finite precision floating point arithmetic

.5167 x 10<sup>-6</sup>

mantissa    exponent

So many bits are allocated to each bit  
(we have 32 or 64 bit machines)

Not possible to represent all numbers






# Simple Example

- Say we can only represent  
+/-1 digit in mantissa and exponent can  
be  $10^{-1}$ ,  $10^0$ ,  $10^1$

What are all the numbers that can be  
represented?






# Machine epsilon

- Bit more tricky in binary arithmetic
  - Have machine specific  $\varepsilon$   
(approx  $10^{-16}$ )
  - Numbers smaller in absolute value than  $\varepsilon$  become 0 (called underflow)
  - Numbers greater than largest number become largest number. (called overflow)
- 



# Error Tolerances

- Function  $f(x)-f(x^*)$  off by  $\varepsilon$
  - Gradient of by square root of  $\varepsilon$
  - So tolerances of  $10e-8$  are all you can usually hope for.
- 



# Termination Criteria

- We would like  $\|\nabla f(x)\| = 0$
- But this may never happen
- So could take

$$\|\nabla f(x)\| \leq \sqrt{\varepsilon} = 10^{-8}$$



# Scale independent

- Don't want to depend on scaling

$f(x)$  = kilometers

$f(x)$  = meters \* 1000

- So use

$$\frac{\|\nabla f(x)\|}{1+|f(x)|} \leq \tau \text{ or } \|\nabla f(x)\| \leq \tau(1+|f(x)|)$$

# Gill, Murray and Wright

- Recommend in addition

$$\frac{\|f(x_k) - f(x_{k-1})\|}{|1 + f(x_k)|} \leq \tau_2$$

$$\frac{\|x_k - x_{k-1}\|}{1 + \|x_k\|} \leq \tau_3$$

# Choice of parameter

- Machine precision is about  $10e-16$

$$\tau_1 = \tau_3 = \sqrt{\tau_2} \quad \text{or} \quad \sqrt[3]{\tau_2}$$

$$\tau_2 \approx \text{machine accuracy } 10^{-16}$$

$$\tau_1 = \tau_3 \leq 10^8$$


- Difference is because gradient is less accurate than function evaluation
- Recommend infinity norm for very large problems.



# Exercise

- Use your newton and steepest descent codes to optimize the following function:

$$\min f(x_1, x_2) = 5x_1^4 + 6x_2^4 - 6x_1^2 + 2x_1 x_2 + 5x_1 - 7x_2 + 13$$

- How small can you make the tolerance before you reach limits of precision of your computer?
- 



# Scaling

- Wise to scale variables if very different scales.
  - This can improve conditioning of problem. See previous discussion.
- 