§6.5.6 Conjugate Gradient Method. (non-linear)

- alt. to N.M.
- no need 2nd deriv.
- doesn't store approx to Hessian
- uses gradients, but removes components along old search directions.

S.D.

$$x_{k+1} = x_k + \alpha s_k$$
$$s_k = -\nabla f(x_k)$$
$$\phi(\alpha) = f(x_k + \alpha s_k)$$

C.G.

$$s_0 = -\nabla f(x_0)$$

$$s_k = -\nabla f(x_k) + \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})} s_{k-1}$$

- n iterations for quadratic functions
- restart at n iter $s_k = -\nabla f(x_k)$.

$$g_k = \nabla f(x_k)$$
$$s_{k+1} = -g_{k+1} + \beta_{k+1} s_k$$

Algorithm 11.1	vs.	Algorithm 6.6
$x_0 =$ initial guess		$x_0 =$ initial guess
$r_0 = b - Ax_0$		$g_0 = \nabla f(x_0)$
$s_0 = r_0$		$s_0 = -g_0$
for $k = 0, 1, 2, \ldots$		for $k = 0, 1, 2, \ldots$
$\alpha_k = r_k^T r_k / s_k^T A s_k$		Choose α_k to min. $f(x_k + \alpha_k s_k)$
$x_{k+1} = x_k + \alpha_k s_k$		$x_{k+1} = x_k + \alpha_k s_k$
$r_{k+1} = r_k - \alpha_k A s_k$		$g_{k+1} = \nabla f(x_{k+1})$
$\beta_{k+1} = \dfrac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$		$\beta_{k+1} = \dfrac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$
$s_{k+1} = r_{k+1} + \beta_{k+1} s_k$		$s_{k+1} = -g_{k+1} + \beta_{k+1} s_k$
end.		end.
Linear CG		Non-linear CG.

CG 11.5.5

LECTURE 12

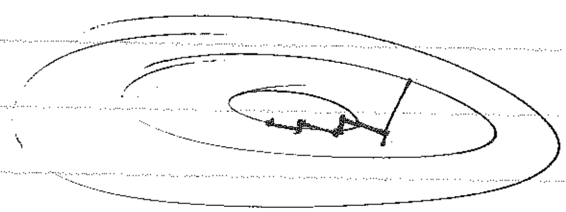
Quadratic Form, A s.p.d.

$$f(x) = \frac{1}{2} x^T A x - b^T x + c \quad \Rightarrow \nabla f = Ax - b$$

solution $Ax = b$.

Recall Steepest Descent Method.

$$S_k = r_k = -\nabla f(x_k) = b - Ax_k$$



each dir is orthogonal to previous dir

convergence is slow.

Conjugate Gradients.

$$e_0 = \sum_{i=0}^{n-1} a_i s_i \qquad x_0 - x^* = e_0$$
$$\Rightarrow x^* = x_0 - \sum_{i=0}^{n-1} a_i s_i$$

$$e_k = x_k - x^* = x_0 + \sum_{i=0}^{k-1} \alpha_i s_i - x^* = e_0 + \sum_{i=0}^{k-1} \alpha_i s_i \Rightarrow \alpha_i = -a_i$$

① s_i mutually orthogonal

$$s_k^T e_0 = \sum_{i=0}^{n-1} a_i s_k^T s_i = a_k s_k^T s_k$$

$$\Rightarrow a_k = \frac{s_k^T e_0}{s_k^T s_k} = \frac{s_k^T (e_0 + \sum_{i=0}^{k-1} \alpha_i s_i)}{s_k^T s_k} = \frac{s_k^T e_k}{s_k^T s_k}.$$

This tells us what α_k should be, but we don't know e_k!

② assume s_i are A-orthogonal

$$s_i^T A s_k = 0 \quad \forall\, i \neq k$$

$$e_0 = \sum_{i=0}^{n-1} a_i s_i$$

$$s_k^T A e_0 = \sum_{i=0}^{n-1} a_i s_k^T A s_i = a_k s_k^T A s_k$$

$$\Rightarrow a_k = \frac{s_k^T A e_0}{s_k^T A s_k} = \frac{s_k^T A e_k}{s_k^T A s_k} = \frac{s_k^T r_k}{s_k^T A s_k} \quad \checkmark$$

good, r_k we do know.

How to find s_k? (s.t. $s_k \perp A s_i$, $i < k$)
Gram-Schmidt A-orthogonalization

Gram-Schmidt A-Orthogonalization

Set of n linearly indep vectors $u_0, \dots, u_{n-1}$

s_i choose u_i + transform to be A-conj to $s_0, \dots, s_{i-1}$

$$\cancel{s_i = \left(u_i + \sum_{j=0}^{i-1} \beta_{ij} s_j\right.}$$

$$s_i = u_i + \sum_{j=0}^{i-1} \beta_{ij} s_j$$

$s_k^T A s_i =$

goal: $s_i^T A s_k = 0$ $\forall\, k < i$

$$0 \le k < i \qquad s_k^T A s_i = s_k^T A u_i + \sum_{j=0}^{i-1} s_k^T A \beta_{ij} s_j$$

(assume prev s_i were A-orthogonal)

$$0 = s_k^T A s_i = s_k^T A u_i + \beta_{ik} s_k^T A s_k$$

$$\Rightarrow \boxed{\beta_{ik} = \frac{-s_k^T A u_i}{s_k^T A s_k}}$$

Difficulty: need to keep all the old s_i in memory.

Idea: use residuals as $\{u_i\}$