This gives idea on how to find solution to $Ax = b$.

$$x^{(k+1)} = x^{(k)} + \underbrace{t_k}_{\text{step size}} \underbrace{v^{(k)}}_{\text{"search direction"}}$$

If $v^{(k)}$ is given then our calculation reveals that the best possible stepsize we can take (i.e. the one that reduces the most value of objective function $q(x)$) is:

$$t_k = \frac{\langle v^{(k)}, b - Ax^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle} = \frac{\langle v^{(k)}, r^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$$

where $r^{(k)} = b - Ax^{(k)}$ = residual at step k.

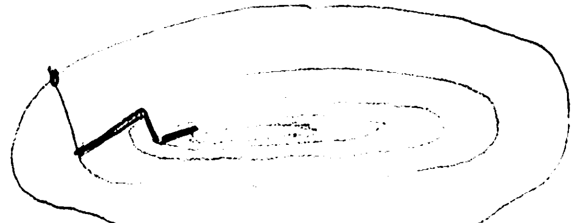
One example of such method is:

<u>Steepest descent</u> where $v^{(k)} = r^{(k)} = -\nabla q(x^{(k)})$

Algorithm.

```
for $k = 1, 2, \ldots$
    $v = b - Ax$
    $t = \frac{(v, v)}{(v, Av)}$
    $x = x + tv$
```

This is not a very efficient method. It has slow, typically "zig-zagging" convergence to solution:



level sets of $q(x)$ are ellipses when $x \in \mathbb{R}^2$.

<u>Note</u>: $r^{(k)}$ is $\perp$ to level sets of $q(x)$.

Conjugate Gradient method (Hestenes & Stiefel, 1952)

$x_{k+1} = x_k + \alpha_k p_k$ where α_k, p_k is determined s.t. x_{k+1} solves

$$\boxed{(1) \quad \min_{x \in x_0 + \text{span}\{p_0, \dots, p_{k-1}, r_k\}} \frac{1}{2} x^T A x - b^T x}$$

$x = \hat{x} + x_0$

$$\Leftrightarrow \boxed{(2) \quad \min_{\hat{x} \in \text{span}\{p_0, \dots, p_{k-1}, r_k\}} \frac{1}{2} \hat{x}^T A \hat{x} - \hat{x}^T r_0}$$

__Recall__: $\text{span}\{u_1, u_2, \dots u_n\} = \left\{ \sum_{i=1}^{n} \alpha_i u_i \;\middle|\; \alpha_i \in \mathbb{R} \text{ for } i = 1, \dots, n \right\}$

= set of all possible linear combinations of vector $u_1, \dots u_n$

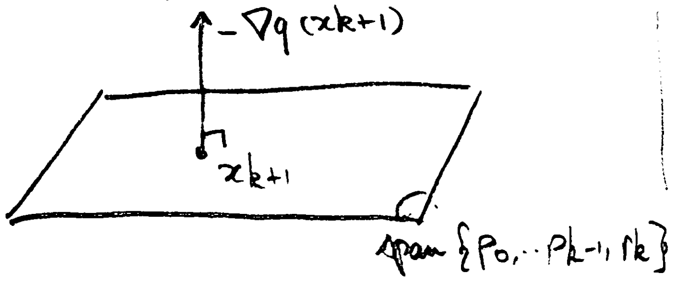
= __linear span__ of family $\{u_1, \dots u_n\}$.

It can be shown that $\hat{x}_{k+1}$ solves (2) iff:

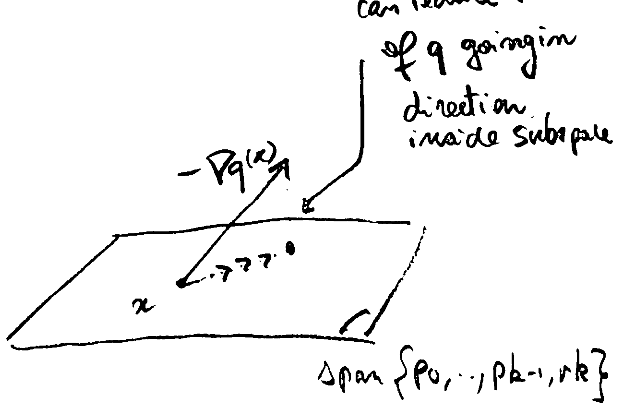
$$(A\hat{x}_{k+1} - r_0)^T v = 0 \quad \forall v \in \text{span}\{p_0, \dots, p_{k-1}, r_k\}$$

$$\Leftrightarrow \underbrace{(A x_{k+1} - b)}_{= -\nabla q(x_{k+1})}{}^T v = 0 \quad \text{" " "}$$

__Intuitively__:



OPTIMAL: no direction of descent in span$\{\dots\}$


can reduce value of q going in direction inside subspace
$-\nabla q(x)$
x
span$\{p_0, \dots, p_{k-1}, r_k\}$

SUB-OPTIMAL

<u>note</u>: v is called a <u>descent direction</u> for q if

$$v^T \nabla q(x) < 0.$$

Looking at Taylor expansion around x:

$$q(x+tv) = q(x) + \underbrace{t\,\nabla q(x)^T v}_{\substack{<0 \text{ when } t>0 \\ \text{and } \nabla q(x)^T v < 0.}} + o(t)$$

$\Rightarrow$ means there is a step length $t>0$ for which $q(x+tv) < q(x)$.

Now look at previous step $\hat{x}_k$, which must solve:

$$\min_{\hat{x} \in \text{span}\{p_0, \dots, p_{k-1}\}} \frac{1}{2}\hat{x}^T A \hat{x} - \hat{x}^T r_0 \iff v^T(b - Ax_k) = 0, \quad \forall v \in \text{span}\{p_0, \dots p_{k-1}\}$$

$$\iff \boxed{\begin{array}{l} p_j^T(b - Ax_k) = 0 \\ \text{for } j = 0, \dots, k-1. \end{array}}$$

We also have:

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

$$0 = (Ax_{k+1} - b)^T p_j = \overset{=0}{\cancel{(Ax_k - b)^T p_j}} + \alpha_k p_k^T A p_j$$

for $j = 0, \dots, k-1$

$\Rightarrow$ $\boxed{p_k \text{ is } A\text{-orthogonal to previous } k \text{ directions}}$

<u>A-orthogonal</u> means orthogonality w.r.t. inner product:

$$\langle u, v \rangle_A = \langle u, Av \rangle = u^T A v.$$

It is not hard to check $\langle u, v \rangle_A$ is indeed an inner product when A is s.p.d.

So we can obtain p_k be A-orthogonalizing family of vectors $\{r_0, r_1, \dots, r_{k-1}, r_k\}$.

Use Gram-Schmidt orthogonalization.

$$p_0 = r_0$$
$$p_k = r_k - \sum_{j=0}^{k-1} \underbrace{\frac{r_k^T A p_j}{p_j^T A p_j} p_j}_{\substack{= \perp \text{ proj w.r.t. } \langle \cdot, \cdot \rangle_A \text{ inner prod} \\ \text{of } r_k \text{ along } p_j.}}$$

Now take the optimal step size along this direction:

$$\boxed{\alpha_k = \frac{p_k^T (b - Ax_k)}{p_k^T A p_k} = \frac{p_k^T r_k}{p_k^T A p_k}}$$

Gram Schmidt + this choice of step size is essentially a preliminary version of CG.

However in this version:
- cost of iteration grows linearly with iteration number.
- storage needed " " " " " .

Fortunately the Gram-Schmidt orthogonalization takes a simpler form if we look closely at subspaces we use.

The subspace that CG uses to look for new direction is:

$$\text{span}\{p_0, p_1, \dots, p_{k-1}, r_k\} = \text{span}\{p_0, \dots, p_{k-1}, p_k\}$$
$$= \text{span}\{r_0, \dots, r_{k-1}, r_k\}$$
$$= \mathcal{K}_{k+1}(A, r_0)$$
$$= \underline{\text{Krylov subspace}}$$

<u>why?</u>

$$r_0 = b - Ax_0 \in \mathcal{K}_1(A, r_0)$$

If $r_{k-1} = b - Ax_{k-1} \in \mathcal{K}_k(A, r_0)$ then

$$r_k = b - Ax_k = b - A(x_{k-1} + \alpha p_{k-1})$$
$$= \underbrace{b - Ax_{k-1}}_{= r_{k-1} \in \mathcal{K}_k} - \underbrace{\alpha A \underbrace{p_{k-1}}_{\in \mathcal{K}_k}}_{\in \mathcal{K}_{k+1}}$$

$$\Rightarrow \boxed{r_k \in \mathcal{K}_{k+1}}$$

Conjugate Gradient forms part of a large family of iterative solvers called <u>Krylov Subspace methods</u>.

<u>Why did we introduce these subspaces?</u>

Recall optimality conditions, now written with Krylov subspaces.

$$r_{k+1}^T v = 0, \quad \forall v \in \mathcal{K}_{k+1}(A, r_0)$$
$$r_k^T v = 0, \quad \forall v \in \mathcal{K}_k(A, r_0)$$

Take $p_i \in \mathcal{K}_{k-1}(A, r_0) \Rightarrow Ap_i \in \mathcal{K}_k(A, r_0)$

$$\Rightarrow r_k^T A p_i = 0, \quad \text{for } i = 0, \dots, k-2$$

This orthogonality greatly simplifies Gram Schmidt and reduces sum to only one term:

$$p_k = r_k - \underbrace{\boxed{\frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}}}_{\beta_k} p_{k-1}$$

- Storage and cost of iteration remains the same regardless of k.
- One needs roughly only 5 vectors of length n.
- One requires only knowledge of action of A on a vector.
  $\Rightarrow$ no need to know entries of A as in direct methods.
  A could be even specified via a "black-box" code.

In the classical formulation of CG α_k and β_k take different forms which can be obtained by applying optimality conditions:

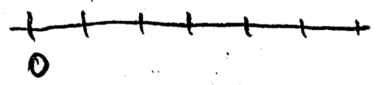
$$\alpha_k = \frac{\|r_k\|^2}{p_k^T A p_k}$$

$$\beta_k = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$$

Convergence of CG:

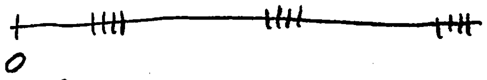
- If A is s.p.d. CG converges in at most n steps, where n = dimension of A.
- In general convergence of CG depends on how many "clusters" of eigenvalues does A have.

Spectrum of A (all eigenvalues)



slow convergence

(S1)



fast convergence in roughly as many iterations as there are clusters (here ~ iter)

(S2)

Preconditioning

To speed up convergence of CG (i.e. going from (S1) to (S2)) one solves system:

$$M^{-1}Ax = M^{-1}b, \quad \text{where } M \text{ is invertible}$$

$$\Leftrightarrow \qquad Ax = b$$

M = (left) preconditioner

= approximation of A that is easy to invert.

One can also think of right preconditioners:

$$A\underbrace{M^{-1}y}_{=x} = b \qquad \Leftrightarrow \quad Ax = b$$

For symmetric systems, we don't want to loose symmetry, so letting

$$M = CC^T, \ M \text{ invertible}:$$

$$Ax = b \quad \Leftrightarrow \quad \underbrace{C^{-1}AC^{-T}}_{\text{s.p.d. if } A \text{ s.p.d.}} C^T x = C^{-1}b$$

<u>Note</u>: there is no single way of choosing a preconditioner. A preconditioner that works for a particular problem may not work as well for another.

Here are some preconditioning techniques that work (see Trefethen and Bau for a broader overview)

- incomple LU or Cholesky factorization: drop elements that are below a threshold to get sparse LU (L) factors that give cheap systems to solve
- Jacobi : use diagonal or block-diagonal of A
- Discretization based : use solution to a coarse grid problem, a constant coeff problem, a periodic problem or solve problem in alternating directions (we will see this more closely when we get to ADI methods)

The final algorithm is as follows:

CONJUGATE GRADIENT (unpreconditioned)

```
$x_0$ = given
$r_0 = b - Ax_0$
$p_0 = r_0$
for $n = 1, 2, 3, \dots$
    $\alpha_n = \dfrac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}}$
    $x_n = x_{n-1} + \alpha_n p_{n-1}$
    $r_n = r_{n-1} - \alpha_n A p_{n-1}$   ← check convergence here by looking at $\|r_n\|$
    $\beta_n = \dfrac{r_n^T r_n}{r_{n-1}^T r_{n-1}}$
    $p_n = r_n + \beta_n p_{n-1}$
```

and the preconditioned version is:

PRECONDITIONED CONJUGATE GRADIENT

```
$x_0$ = given
$M$ = preconditioner = given
$r_0 = b - Ax_0$
Solve $My_0 = r_0$ for $y_0$
$p_0 = y_0$
for $n = 1, 2, 3, \dots$
    $\alpha_n = \dfrac{r_{n-1}^T y_{n-1}}{p_{n-1}^T A p_{n-1}}$
    $x_n = x_{n-1} + \alpha_n p_{n-1}$
    $r_n = r_{n-1} - \alpha_n A p_{n-1}$    ← check convergence here.
    Solve $My_n = r_n$ for $r_n$.
    $\beta_n = \dfrac{y_n^T r_n}{y_{n-1}^T r_{n-1}}$
    $p_n = y_n + \beta_n p_{n-1}$
```

note: This is equivalent to applying CG to system

$$C^{-1} A C^{-T} C^T x = C^{-1} b$$

where $M = CC^T$ = preconditioner..
we will not see equivalence but notice:

$$y_n = M^{-1} r_n \Rightarrow \quad r_n^T y_n = r_n^T M^{-1} r_n$$
$$= (C^{-1} r_n)^T (C^{-1} r_n)$$

prec. residual