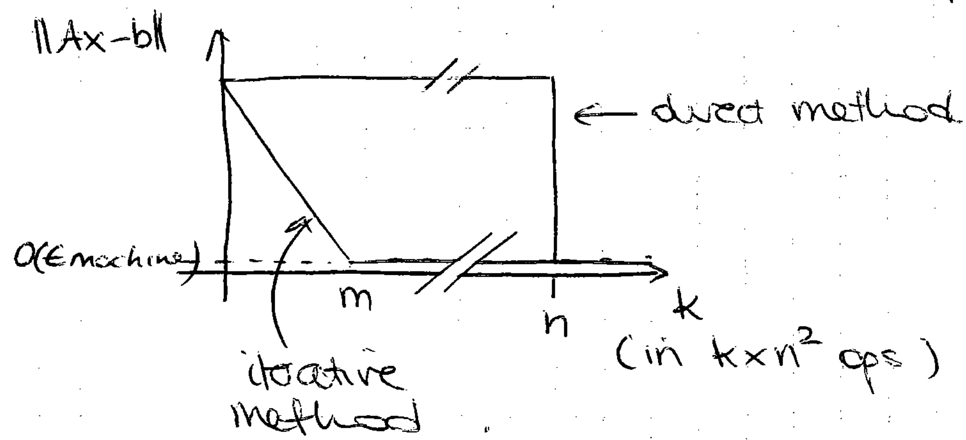
VI Iterative methods for linear systems

① Introduction

- We now consider systems $Ax=b$ where A is extremely large (& dense).
- Direct methods are all $O(n^3)$ for A an $n\times n$ matrix. (Best known direct algorithms $O(n^{2.376})$ see Trefethen & Bau for review, also NR.)
- For $n > 10^4$ typically direct methods are unpractical both in terms of time & memory
- Idea ①. For certain types of matrices, a good approximation to the solution x_* of $Ax=b$ can be obtained iteratively in $m\times n^2$ steps instead of n^3 steps, where $m \ll n$.

  . Furthermore, we can design algorithms that yields the exact solution should it be carried out $n \times n^2$ steps



- The convergence rate in these algorithms depends a lot on the <u>condition number</u> of the matrix A. → ill-conditioned matrices have v. slow convergence rate

(see later)

→ well-conditioned matrices have v. fast convergence rate.

<u>Idea (2)</u>: Instead of solving the system

$$Ax = b$$

solve

$$KAx = Kb$$

where the matrix KA is well-conditioned

→ This is called pre-conditioning.

Good preconditioners can reduce the convergence time by <u>order(s) of magnitude</u>.

(2) Solution of a real, symmetric linear system as a minimization problem

We saw that the least-square minimization problem could be turned into a square, symmetric linear system as

$$A^T A x = A^T b .$$

Conversely, any real, symmetric linear system can be equivalently viewed as the minimization of a function f:

$$A x_* = b \iff x_* \text{ minimizes}$$
$$f(x) = \frac{1}{2} x^T A x - x^T b$$

Indeed
$$\nabla f)_i = \frac{\partial}{\partial x_i}\left(\frac{1}{2} \sum_{j,k} x_j a_{jk} x_k - \sum_k x_k b_k \right)$$
$$= (1/2)\left(\sum_k x_i a_{ik} + \sum_j x_j a_{ji} - b_i \right)$$
$$= (Ax-b)_i$$

So $\nabla f = Ax - b$

and the solution to $Ax - b = 0$ is also a stationnary pt of f.

Note: x_* is a minimum if A is positive definite (see proof in Atkinson p. 563).

In what follows, we only consider positive definite matrices.

③ Iterative methods for minimizing functions

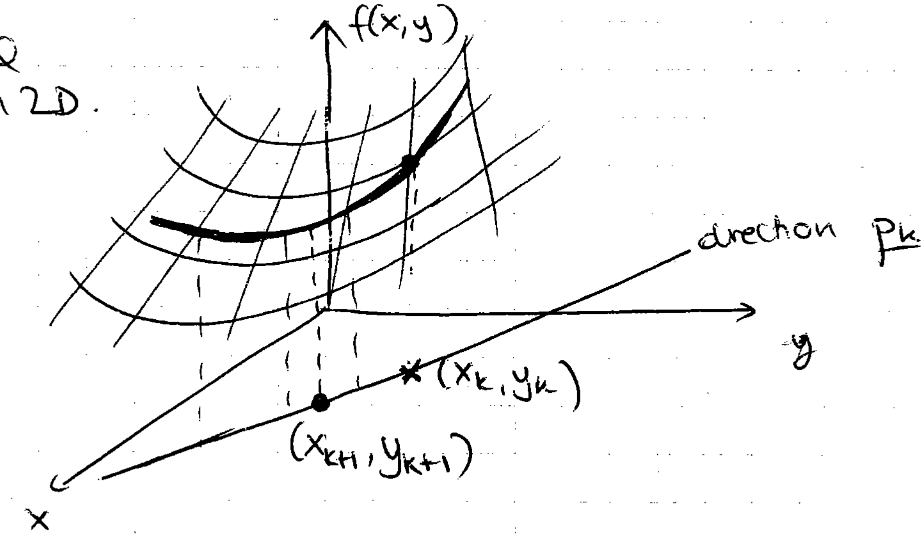
Idea. Start with an initial guess for the minimum. Select a direction, & minimize the function in this direction. This yields a new guess, select a new direction & repeat ...

$$\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{p}_k$$

new guess — old guess — direction searched

where $f(\underline{x}_k + \alpha_k p_k) = \min_{\alpha \in \mathbb{R}} f(\underline{x}_k + \alpha \underline{p}_k)$

Graphical example in 2D.



Note that in our case, since f is a quadratic function, the minimization can be done analytically:

$$f(\underline{x}_k + \alpha \underline{p}_k) = \frac{1}{2}(x_k^T + \alpha p_k^T) A (x_k + \alpha p_k) - x_k^T b - \alpha p_k^T b$$

so

$$\frac{\partial f}{\partial \alpha} = \frac{\partial}{\partial \alpha}\Big[\frac{1}{2} x_k^T A x_k - x_k^T b + \frac{\alpha}{2}(p_k^T A x_k + x_k^T A p_k) - \alpha p_k^T b + \frac{\alpha^2}{2} p_k^T A p_k\Big]$$

$$\Rightarrow \frac{\partial f}{\partial \alpha} = 0 \iff p_k^T A x_k + \alpha_k p_k^T A p_k = \alpha_k p_k^T b$$

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

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

Steepest descent

- One of the difficultes of the method is the choice of the directions p_k to follow
- A possibility that comes straightforwardly to mind is the method of steepest descent

→ choose the directions p_k to follow the direction of steepest descent, i.e,

$$p_k = -\nabla f \big|_{x = x_k}$$

$$= b - Ax_k = r_k$$

Algorithm for minimization with steepest descent

(equi. solving $Ax = b$).

$x_0 = \mathbf{0}$

$p_0 = b$

For $k = 1 \ldots$

$$x_k = x_{k-1} + \underbrace{\frac{p_{k-1}^T (b - Ax_{k-1})}{p_{k-1}^T A p_{k-1}}}_{\alpha_{k-1}} p_{k-1}$$

Problem: • the method of steepest descent often yields v. slow convergence, in particular as $x \to x_*$.
• Nothing guarantees that a direction that was once used isn't re-used again later.

(4) Conjugate directions

We now define a new concept, <u>A-conjugate directions</u> through a new inner product

$$\langle u, v \rangle_A = u^T A v$$

$\Rightarrow$ We define two vectors u and v to be A-conjugate if $u^T A v = 0$. (for $u \neq v$).

A set of vectors $\{p_i\}$ form a conjugate set (w.r.t A) if

$$p_i^T A p_j = 0 \quad \forall \ i \neq j$$

<u>Note</u>: for example if A is a real, symmetric matrix then the eigenvectors form a conjugate set.

<u>Theorem</u>: If A is real symmetric, there exist an A-conjugate set of n vectors $(p_0 \dots p_{n-1})$ forming a basis for $\mathbb{R}^n$

$n \times n$ matrix

Then, let's write

$$x_* = \alpha_0 p_0 + \dots \alpha_{n-1} p_{n-1}$$

this implies that

$$\alpha_k = \frac{p_k^T A x_*}{p_k^T A p_k} = \frac{p_k^T b}{p_k^T A p_k}.$$

$\Rightarrow$ If we knew a set of conjugate vectors for A we would be able to write the solution for $Ax = b$ explicitly.

In practice, this is not feasible for large matrices A. Instead, let's construct the following sequences:

$$\begin{cases} x_k = \alpha_1 p_1 + \cdots \quad \alpha_k p_k & 1 \le k \le n \\ r_k = b - A x_k \end{cases}$$

Since for $k = n$, $x_n = x_*$ and $r_n = 0$, then as $k \to n$ $x_k \to x_*$ and $\|r_k\| \to 0$.

<u>Hope</u>: If we are fortunate, $\|r_k\| \to 0$ for $k \ll n$ already (i.e, we don't have to go all the way to $k = n$)

So we are left with the problem of how to <u>construct</u> the directions p_k iteratively.

⑤ Conjugate gradient method for symmetric, positive-definite matrices

We would like to write an algorithm of the kind of the steepest descent (which was

$$x_0 = \cdots$$
$$p_0 = \cdots$$

for $k = 1, n$

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

where

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

$$p_k = r_k = b - A x_k$$

but this would not guarantee that $p_k^T A p_j = 0 \quad j < k$

Indeed, suppose we had managed to select all $\{p_i\}_{i<k}$ conjugate

$$\Rightarrow \quad p_k^T A p_j = r_k^T A p_j$$
$$= (b^T - x_k^T A^T) A p_j$$
$$= b^T A p_j - x_k^T A^T A p_j$$

there is no reason why this should be 0

this could be 0 by assumption with, say $p_0 = b$

$\Rightarrow$ even if all previous $\{p_i\}_{i<k}$ are conjugate, steepest descent implies that the next chosen vector isn't conjugate to them.

<u>Idea</u>: Write instead that

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

add a small residual of previous p_{k-1} here.

and choose β_{k-1} such that $p_k^T A p_{k-1} = 0$

$\Rightarrow$ This expression guarantees that

see Golub & Van Loan p 524 for proof

- the new p_k is conjugate to all previous ones
- the new p_k is the vector closest in direction to the steepest descent direction

So for $p_k^T A p_{k-1} = 0$

$$\Leftrightarrow \quad (r_k^T + \beta_{k-1} p_{k-1}^T) A p_{k-1} = 0$$

$$\Leftrightarrow \quad \beta_{k-1} = - \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}$$

$\Rightarrow$ Conjugate gradient algorithm (in theory)

① Choose an initial guess for solution + initial direction:

$$\begin{cases} x_0 = 0 \\ p_0 = b \\ r_0 = b \qquad (\text{since } \quad r_0 = b - Ax_0 = b) \end{cases}$$

② Iterate for $k = 1 \ldots$

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

$\leftarrow$ Given a chosen direction p_k, this α_k minimizes $f(\alpha)$ (see earlier)

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

$\leftarrow$ so that's the new guess

$$r_{k+1} = b - Ax_{k+1}$$

$\leftarrow$ that's new error

$$\left.\begin{aligned} \beta_k &= -\frac{p_k^T A r_{k+1}}{p_k^T A p_k} \\ p_{k+1} &= r_{k+1} + \beta_k p_k \end{aligned}\right\}$$

this selects the new conjugate direction p_{k+1}.

Conjugate gradient algorithm (in practise)

① Choose $\begin{cases} x_0 = 0 \\ p_0 = b \\ r_0 = b \end{cases}$

② Iterate for $k = 1 \ldots$

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k} \qquad (*)$$

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

$$r_{k+1} = r_k - \alpha_k A p_k$$

if $\|r_{k+1}\|^2 \leq \epsilon$ we're done, otherwise

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \qquad (**)$$

$$p_{k+1} = r_{k+1} + \beta_k p_k.$$

The difference in (*) comes from the fact that

$$r_k^T r_k = p_k^T r_k$$

and since $r_k^T r_k$ was calculated at previous iteration it saves time to use it.

To prove $r_k^T r_k = p_k^T r_k$, use induction:

- at $k=0$: $r_0 = p_0 = b \Rightarrow r_0^T r_0 = p_0^T p_0$ trivially.
- assume it's true at $k = k_0$:

$$r_{k_0}^T r_{k_0} = p_{k_0}^T r_{k_0}$$

- then $r_{k_0+1} = r_{k_0} - \alpha_{k_0} A p_{k_0}$

$$p_{k_0+1} = r_{k_0+1} + \beta_{k_0+1} p_{k_0}$$

$$\rightarrow \quad p_{k_0+1}^T r_{k_0+1} = r_{k_0+1}^T r_{k_0+1} + \beta_{k_0+1} p_{k_0}^T r_{k_0+1}$$

$\Rightarrow$ we want to prove that $p_{k_0}^T r_{k_0+1} = 0$

$$\Rightarrow \quad p_{k_0}^T r_{k_0+1} = p_{k_0}^T r_{k_0} - \alpha_{k_0} p_{k_0}^T A p_{k_0}$$

$$= r_{k_0}^T r_{k_0} - \frac{r_{k_0}^T r_{k_0}}{p_{k_0}^T A p_{k_0}} p_{k_0}^T A p_{k_0} = 0$$

✓

The difference in (**) is for the same reason, and one can show by induction (Homework!) that

$$\frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} = - \frac{r_{k+1}^T A p_k}{p_k^T A p_k}.$$