

An approach for symmetric positive definite matrices

We now assume that the system matrix is symmetric and positive definite (SPD), and discuss a different iterative approach.

Recall the problem we want to solve: given $\underline{b} \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times n}$, we look for $\underline{x}^* \in \mathbb{R}^n$ solution of

$$A\underline{x}^* = \underline{b} \quad (1)$$

Since A is SPD, we can define a scalar product associated with A : $(A\underline{x}, \underline{y}) = \underline{y}^T A \underline{x}$. If A is also positive definite, then

$$(A\underline{x}, \underline{x}) > 0 \quad \forall \underline{x} \neq \underline{0}.$$

Then we can introduce the functional $F : \mathbb{R}^n \rightarrow \mathbb{R}$ defined as:

$$F(\underline{v}) := \frac{1}{2}(A\underline{v}, \underline{v}) - (\underline{b}, \underline{v}) \quad \forall \underline{v} \in \mathbb{R}^n \quad (2)$$

Theorem 1

If $A \in \mathbb{R}^{n \times n}$ is SPD, problem (1) has a unique solution, and is equivalent to the following minimum problem for the functional defined in (2):

$$\begin{cases} \text{find } \underline{u} \in \mathbb{R}^n \text{ such that} \\ F(\underline{u}) \leq F(\underline{v}) \quad \forall \underline{v} \in \mathbb{R}^n \end{cases} \quad (3)$$

(that is, (3) has a unique solution $\underline{u} \in \mathbb{R}^n$, and $\underline{u} \equiv \underline{x}^$).*

Proof.

Since A is positive definite, problem (1) has a unique solution ($\det(A) \neq 0$). Now, F is a quadratic functional (hence, differentiable), and

$$\underline{\nabla} F(\underline{v}) = \begin{bmatrix} \frac{\partial F}{\partial v_1} \\ \frac{\partial F}{\partial v_2} \\ \vdots \\ \frac{\partial F}{\partial v_n} \end{bmatrix} = A\underline{v} - \underline{b} \quad H(F) = A \quad (H(F) = \text{Hessian matrix})$$

Since A is positive definite, the matrix $H(F)$ has positive eigenvalues (and real because A is symmetric). Hence, F is strictly convex, that is, it has a unique minimum. Let $\underline{u} \in \mathbb{R}^n$ be the point of minimum. As such, it verifies

$$\underline{\nabla} F(\underline{u}) = \underline{0} \quad \longrightarrow \quad A\underline{u} - \underline{b} = \underline{0}.$$

Since the solution of (1) is unique, $\underline{u} \equiv \underline{x}^*$.



Descent Methods

Given the equivalence between the linear system (1) and the minimum problem (3), we look for $\underline{x}^*$ as minimum point for $F(\underline{x})$.

Starting from an initial guess $\underline{x}^{(0)}$ (any), we want to construct a sequence $\underline{x}^{(k)}$ converging to $\underline{x}^*$ in the following way:

$\underline{x}^{(0)}$ given. Then, for $k = 0, 1, 2, \dots$ set $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \alpha_k \underline{p}^{(k)}$

- $\underline{p}^{(k)}$ are directions of descent,
- α_k are numbers that tell us how much to descend along $\underline{p}^{(k)}$.

They have to be chosen to guarantee descent, that is, to guarantee that

$$F(\underline{x}^{(k+1)}) < F(\underline{x}^{(k)}) \quad \forall k.$$

Descent methods

The optimal value of α_k can be computed by imposing

$$\frac{\partial}{\partial \alpha} F(x^{(k)} + \alpha p^{(k)}) = 0$$

which guarantees maximum descent of F along the descent direction $p^{(k)}$. Indeed,

$$\begin{aligned} F(x^{(k)} + \alpha p^{(k)}) &= \frac{1}{2} \left(A(x^{(k)} + \alpha p^{(k)}), x^{(k)} + \alpha p^{(k)} \right) - \left(b, x^{(k)} + \alpha p^{(k)} \right) \\ &= \frac{\alpha^2}{2} \left(A p^{(k)}, p^{(k)} \right) + \alpha \left(A x^{(k)} - b, p^{(k)} \right) + \left(\frac{1}{2} A x^{(k)} - b, x^{(k)} \right) \end{aligned}$$

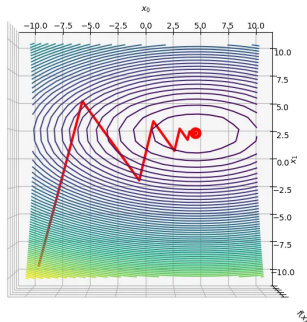
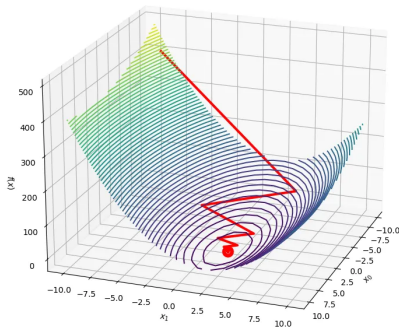
With respect to the variable α , this function is a quadratic polynomial, its graph is a parabola and its minimum occurs at the stationary point:

$$\frac{\partial}{\partial \alpha} F(x^{(k)} + \alpha p^{(k)}) = \alpha \left(A p^{(k)}, p^{(k)} \right) + \left(A x^{(k)} - b, p^{(k)} \right) = 0$$

$$\alpha_k = \text{optimal } \alpha = \frac{(\underline{b} - A \underline{x}^{(k)}, \underline{p}^{(k)})}{(A \underline{p}^{(k)}, \underline{p}^{(k)})} = \frac{(\underline{r}^{(k)}, \underline{p}^{(k)})}{(A \underline{p}^{(k)}, \underline{p}^{(k)})}$$

Steepest Descent Method (or Gradient Method)

$$\underline{p}^{(k)} = \text{direction of steepest descent} = -\underline{\nabla} F(\underline{x}^{(k)}) = \underline{b} - A\underline{x}^{(k)} = \underline{r}^{(k)}$$



Steepest Descent Method (or Gradient Method)

$$\underline{p}^{(k)} = \text{direction of steepest descent} = -\underline{\nabla} F(\underline{x}^{(k)}) = \underline{b} - A\underline{x}^{(k)} = \underline{r}^{(k)}$$

This is also seen from the Taylor expansion:

$$F(\underline{x}^{(k)} + \alpha \underline{p}^{(k)}) = F(\underline{x}^{(k)}) + (\alpha \underline{p}^{(k)})^T \underline{\nabla} F(\underline{x}^{(k)}) + O(\|\alpha \underline{p}^{(k)}\|^2)$$

Thus $F(\underline{x}^{(k+1)}) \leq F(\underline{x}^{(k)})$ if $\underline{p}^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)})$ and $\alpha > 0$ is small enough:

$$F(\underline{x}^{(k+1)}) = F(\underline{x}^{(k)}) - \alpha \|\underline{\nabla} F(\underline{x}^{(k)})\|^2 + O(\alpha^2)$$

Pseudocode for Steepest Descent Method

Steepest descent

Input: A SPD, b , $x^{(0)}$, $a_r, r_r \geq 0$, $\text{maxiter} \geq 1$.

Set $r^{(0)} = b - Ax^{(0)}$ and $\eta = a_r + r_r \|b\|$.

If $\|r^{(0)}\| \leq \eta$, return $x^{(0)}$ with success.

for $k = 1, \dots, \text{maxiter}$

$$y = Ar^{(k-1)}$$

$$\alpha_{k-1} = (r^{(k-1)}, r^{(k-1)}) / (y, r^{(k-1)})$$

$$x^{(k)} = x^{(k-1)} + \alpha_{k-1} r^{(k-1)}$$

$$r^{(k)} = b - Ax^{(k)}$$

If $\|r^{(k)}\| \leq \eta$, return $x^{(k)}$ with success.

end

Return $x^{(k)}$ and report that the tolerance was not met.

In exact arithmetic, the denominator is positive whenever the residual is nonzero. In floating point, a nonpositive denominator signals breakdown. For dense A , the leading cost is $\sim 2n^2$ FLOPs per iteration. Residuals may be updated recursively, with explicit recomputation before accepting convergence.

Convergence of Steepest Descent Method

Theorem 2

The Steepest Descent method converges for every initial guess $\underline{x}^{(0)}$. Moreover it holds:

$$\left\| \underline{x} - \underline{x}^{(k)} \right\|_A \leq \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \right)^k \left\| \underline{x} - \underline{x}^{(0)} \right\|_A$$

where $\|v\|_A = \sqrt{v^T A v}$ is the A -norm.

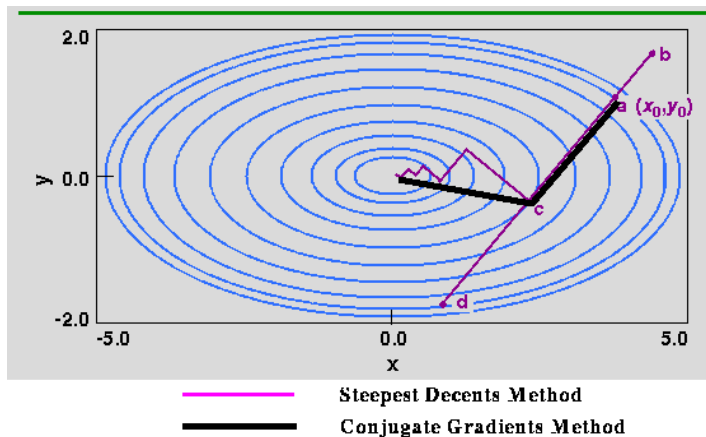
Convergence is guaranteed, but can be very slow if A is ill-conditioned.

Summary and extensions of gradient methods...

We have our functional $F(\underline{v}) := \frac{1}{2}(A\underline{v}, \underline{v}) - (\underline{b}, \underline{v})$ to minimize and use $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \alpha_k \underline{p}^{(k)}$, where $\underline{p}^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)}) = \underline{b} - A\underline{x}^{(k)}$. Possible alternatives are:

- to simplify the calculation of $\underline{p}^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)})$, e.g. in the **stochastic gradient descent** method, used in machine learning: we save time per each iteration at the expenses of an increased number of iterations to reach a given accuracy;
- to find better descent directions $\underline{p}^{(k)}$, such that the convergence at a given tolerance requires fewer iterations, as in the **conjugate gradient** method

Steepest descents vs. Conjugate Gradient



Conjugate Gradient method

with $\underline{p}^{(0)} = -\underline{\nabla}F(\underline{x}^{(0)})$, at each iteration k take $\underline{p}^{(k)}$ in the plane $\text{span}\{\underline{r}^{(k)}, \underline{p}^{(k-1)}\}$, that is:

$$\underline{p}^{(k)} = \underline{r}^{(k)} - \beta_k \underline{p}^{(k-1)}$$

where β_k is chosen so that $\underline{p}^{(k)}$ is A -orthogonal to $\underline{p}^{(k-1)}$, i.e.
 $(\underline{p}^{(k)})^T A \underline{p}^{(k-1)} = 0$ (orthogonal in the scalar product associated with A).
It can be proven that

$$(\underline{p}^{(k)})^T A \underline{p}^{(j)} = 0, \quad j = 0, \dots, k-1.$$

This approach is faster than the steepest descent. Actually, the method terminates in at most n iterations in exact arithmetic (n =dimension of the system), although it is used as an iterative solver in floating-point arithmetic.

Matlab function: $\underline{x} = \text{pcg}(A, b, \dots)$

Pseudocode for Conjugate Gradient Method

Conjugate gradient

Input: A SPD, b , $x^{(0)}$, $a_r, r_r \geq 0$, $\text{maxiter} \geq 1$.

$r^{(0)} = b - Ax^{(0)}$, $p^{(0)} = r^{(0)}$, $\eta = a_r + r_r \|b\|$.

If $\|r^{(0)}\| \leq \eta$, return $x^{(0)}$ with success.

for $k = 1, \dots, \text{maxiter}$

$y = Ap^{(k-1)}$; if $(y, p^{(k-1)}) \leq 0$, report breakdown.

$\alpha_{k-1} = (p^{(k-1)}, r^{(k-1)}) / (y, p^{(k-1)})$

$x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)}$

$r^{(k)} = r^{(k-1)} - \alpha_{k-1} y$

If $\|r^{(k)}\| \leq \eta$, verify $\|b - Ax^{(k)}\| \leq \eta$ and return on success.

$\beta_k = (y, r^{(k)}) / (y, p^{(k-1)})$

$p^{(k)} = r^{(k)} - \beta_k p^{(k-1)}$

end

Return $x^{(k)}$ and report that the tolerance was not met.

If the explicit residual check fails, restart from $x^{(k)}$ using its true residual and the remaining iteration budget. The minus sign agrees with our definition of β_k .