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 $b \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times n}$, we look for $x^* \in \mathbb{R}^n$ solution of

$$Ax^* = b \tag{1}$$

Write $(x, y) = \sum_{i=1}^{n} x_i y_i$ for the Euclidean scalar product. Since A is SPD, it also defines the scalar product

$$(x, y)_A := (Ax, y), \qquad (x, x)_A > 0 \quad \forall x \neq 0.$$

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

$$F(x) := \frac{1}{2}(x, x)_A - (b, x) \qquad \forall x \in \mathbb{R}^n \tag{2}$$

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):

$$\begin{cases} \text{find } u \in \mathbb{R}^n \text{ such that} \\ F(u) \leq F(x) \qquad \forall x \in \mathbb{R}^n \end{cases} \tag{3}$$

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

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

$$\nabla F(x) = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{bmatrix} = Ax - b \quad H(F) = A \qquad (H(F) = \text{Hessian matrix})$$

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 $u \in \mathbb{R}^n$ be the point of minimum. As such, it verifies

$$\nabla F(u) = 0 \quad \longrightarrow \; Au - b = 0.$$

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

Example: gradient and Hessian in two variables

Let $x = (x_1, x_2)^T$, $b = (b_1, b_2)^T$ and $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ be SPD, so $a_{21} = a_{12}$.

Expanding the products gives

$$\begin{aligned} F(x) &= \frac{1}{2}(Ax, x) - (b, x) \\ &= \frac{1}{2}a_{11}x_1^2 + a_{12}x_1x_2 + \frac{1}{2}a_{22}x_2^2 - b_1x_1 - b_2x_2. \end{aligned}$$

Differentiate with respect to one variable, keeping the other constant:

$$\begin{aligned} \frac{\partial F}{\partial x_1} &= \frac{1}{2}a_{11} \cdot 2x_1 + a_{12}x_2 - b_1 = a_{11}x_1 + a_{12}x_2 - b_1, \\ \frac{\partial F}{\partial x_2} &= a_{12}x_1 + \frac{1}{2}a_{22} \cdot 2x_2 - b_2 = a_{21}x_1 + a_{22}x_2 - b_2. \end{aligned}$$

Collecting these derivatives in a column gives

$$\nabla F(x) = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 - b_1 \\ a_{21}x_1 + a_{22}x_2 - b_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = Ax - b.$$

Differentiating each component once more gives the Hessian:

$$H(F) = \begin{pmatrix} \partial^2 F/\partial x_1^2 & \partial^2 F/\partial x_2 \partial x_1 \\ \partial^2 F/\partial x_1 \partial x_2 & \partial^2 F/\partial x_2^2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = A.$$

Descent Methods

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

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

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

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

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

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

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,

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

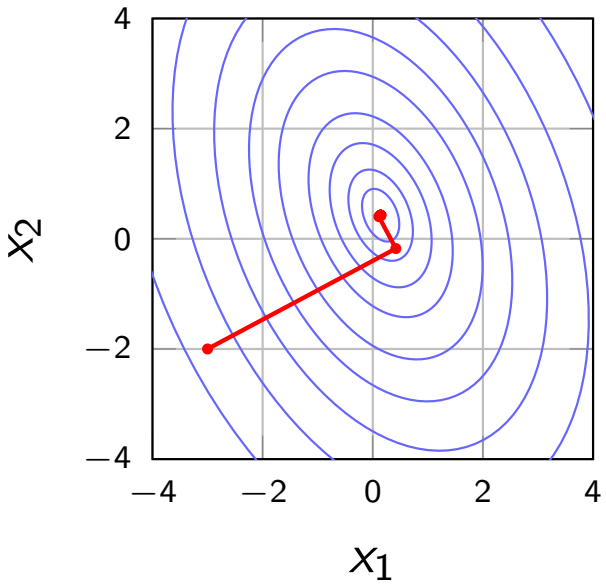
With respect to the variable α, this function is a quadratic polynomial, its graph is a parabola and its minimum occurs at the stationary point:

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

$$\alpha_k = \text{ optimal } \alpha = \frac{(b - Ax^{(k)}, p^{(k)})}{(p^{(k)}, p^{(k)})_A} = \frac{(r^{(k)}, p^{(k)})}{(p^{(k)}, p^{(k)})_A}$$

Steepest Descent Method (or Gradient Method)

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



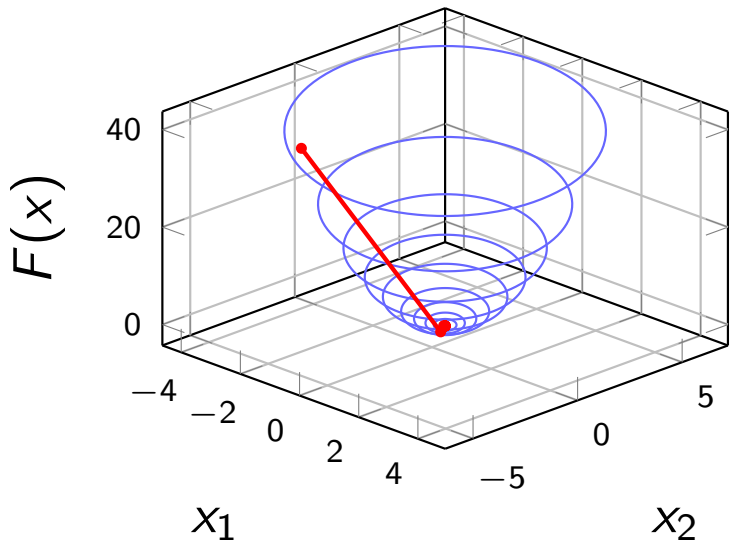


Illustration: $A = \left(\begin{smallmatrix} 4 & 1 \\ 1 & 2 \end{smallmatrix}\right)$, $b = (1,1)^T$, starting from $x^{(0)} = (-3,-2)^T$.

Steepest Descent Method (or Gradient Method)

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

This is also seen from the Taylor expansion:

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

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

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

Pseudocode for Steepest Descent Method

Steepest descent

Input: A SPD, b, $x^{(0)}$, $0 < \varepsilon < 1$, maxiter ≥ 1.
Set $r^{(0)} = b - Ax^{(0)}$, $p^{(0)} = r^{(0)}$, $\eta = \varepsilon \|b\|$.
If $\|r^{(0)}\| \leq \eta$, return $x^{(0)}$ with success.
for $k = 1, \ldots,$ maxiter
  $y = Ap^{(k-1)}$
  $\alpha_{k-1} = (p^{(k-1)}, r^{(k-1)})/(y, p^{(k-1)})$
  $x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)}$
  $r^{(k)} = b - Ax^{(k)}$ (or $r^{(k)} = r^{(k-1)} - \alpha_{k-1} y$)
  If $\|r^{(k)}\| \leq \eta$, return $x^{(k)}$ with success.
  $p^{(k)} = r^{(k)}$
end
Return $x^{(k)}$ and report that the tolerance was not met.

Since $y = Ap^{(k-1)}$, substituting the update of $x^{(k)}$ gives

$$b - Ax^{(k)} = b - Ax^{(k-1)} - \alpha_{k-1} Ap^{(k-1)} = r^{(k-1)} - \alpha_{k-1} y.$$

The residual shortcut avoids another product with A; $p^{(k)}$ reuses $r^{(k)}$. With the fast residual update, the cost is $\sim 2n^2$ FLOPs per iteration.

Convergence of Steepest Descent Method

Theorem 2

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

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

where $\|x\|_A = \sqrt{(x, x)_A}$ *is the* $A-$*norm.*

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

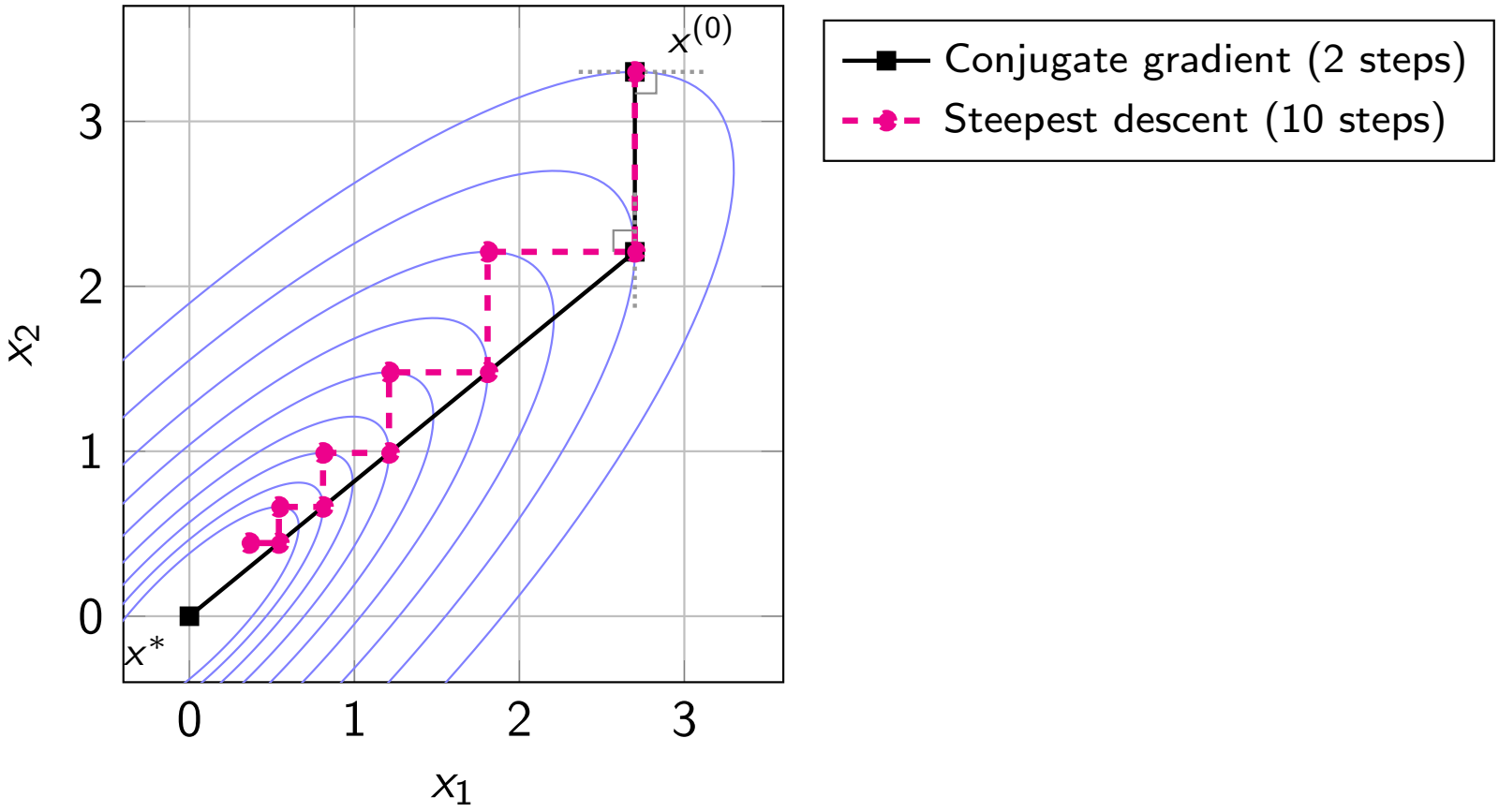
Summary and extensions of gradient methods...

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

- to simplify the calculation of $p^{(k)} = -\nabla F(x^{(k)})$, 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 $p^{(k)}$, such that the convergence at a given tolerance requires fewer iterations, as in the **conjugate gradient** method

Steepest descent vs. Conjugate Gradient

$A = \frac{1}{2}\begin{pmatrix} 11 & -9 \\ -9 & 11 \end{pmatrix}$, $b = 0$, $x^{(0)} = (2.7, 3.3)^T$, $x^* = 0$.



Each steepest descent direction (gradient) is orthogonal to the level curve at its starting point.

Conjugate Gradient method

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

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

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

$$\left(p^{(k)}, p^{(j)}\right)_A = 0, \qquad j = 0, \ldots, k-1.$$

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: $x =$**pcg**$(A, b, ...)$

Pseudocode for Conjugate Gradient Method

Conjugate gradient

Input: A SPD, b, $x^{(0)}$, $0 < \varepsilon < 1$, maxiter ≥ 1.
$r^{(0)} = b - Ax^{(0)}$, $p^{(0)} = r^{(0)}$, $\eta = \varepsilon \|b\|$.
If $\|r^{(0)}\| \leq \eta$, return $x^{(0)}$ with success.
for $k = 1, \ldots,$ maxiter
  $y = Ap^{(k-1)}$; if $(y, p^{(k-1)}) \leq 0$, report breakdown.
  $\alpha_{k-1} = (p^{(k-1)}, r^{(k-1)})/(y, p^{(k-1)})$
  $x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)}$
  $r^{(k)} = b - Ax^{(k)}$ (or $r^{(k)} = r^{(k-1)} - \alpha_{k-1} y$)
  If $\|r^{(k)}\| \leq \eta$, return $x^{(k)}$ with 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.

The fast residual formula follows from $y = Ap^{(k-1)}$ and the update of $x^{(k)}$, as in steepest descent. For $p^{(k)}$, reuse the residual just computed: $b - Ax^{(k)} = r^{(k)}$. Both shortcuts avoid another product with A.