Iterative methods

Consider the linear system

$$A\underline{x} = \underline{b}$$

Iterative methods start from an initial guess $\underline{x}^{(0)}$ and construct a sequence of approximate solutions $\{\underline{x}^{(k)}\}$ such that

$$\underline{x} = \lim_{k \to \infty} \underline{x}^{(k)}.$$

Splitting methods

The matrix A is split as

$$A = M - N$$

Splitting methods go like

$$\underline{x}^{(0)} \text{ given} \quad \text{solve} \quad M\underline{x}^{(k)} = \underline{b} + N\underline{x}^{(k-1)} \quad k = 1, 2, \cdots \qquad (1)$$

With iterative methods we give up the idea of computing the exact solution, but we want low operational costs. In particular:

- the system (1) must be much easier to deal with than the original system $Ax = b$, that is, the matrix M must be as simple as possible, and of course non-singular;
- the sequence $\{\underline{x}^{(k)}\}$ must converge to $\underline{x}$ for any initial guess $\underline{x}^{(0)}$;
- the convergence must be fast.

Different choices for M give rise to different iterative methods.

Jacobi method

take $M = diag(A)$ (and hence $N = M - A$), applicable if $a_{ii} \neq 0 \ \forall i$. At each iteration k we have to solve a diagonal system

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} - \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ \vdots \\ x_n^{(k-1)} \end{pmatrix}$$

Thus we obtain

$$x_i^{(k)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k-1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) / a_{ii} \quad i = 1, \cdots, n$$

The number of operations for each component is $\sim 2n$, so that the cost for one Jacobi iteration is $\sim 2n^2$.

Gauss-Seidel method

take $M = tril(A)$, applicable if $a_{ii} \neq 0 \ \forall i$. At each iteration k we have to solve a lower triangular system

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} - \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ \vdots \\ x_n^{(k-1)} \end{pmatrix}$$

Thus we obtain

$$x_i^{(k)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) / a_{ii} \quad i = 1, \cdots, n$$

The difference with respect to Jacobi method is in the first sum of the formula, where the updated $x_j^{(k)}$ are used instead of the old $x_j^{(k-1)}$. The number of operations is exactly the same: for each component is $\sim 2n$, so that the cost for one Gauss-Seidel iteration is $\sim 2n^2$.

Convergence analysis for splitting methods

In all cases we want convergence for any initial guess $\underline{x}^{(0)}$. With paper and pencil we study the error at each iteration.
Let $e^{(k)} = \underline{x} - \underline{x}^{(k)}$ be the error at the k^{th} iteration.
Since $\underline{x}$ and $\underline{x}^{(k)}$ are solutions of

$$M\underline{x} = \underline{b} + N\underline{x}, \qquad M\underline{x}^{(k)} = \underline{b} + N\underline{x}^{(k-1)},$$

by subtracting we get

$$M(\underline{x} - \underline{x}^{(k)}) = N(\underline{x} - \underline{x}^{(k-1)}) \implies e^{(k)} = \underbrace{M^{-1}N}_{B} e^{(k-1)}$$

where $B = M^{-1}N$ is the iteration matrix.

$$e^{(k)} = Be^{(k-1)} \quad k = 1, 2, \cdots, \implies e^{(k)} = B^k e^{(0)}.$$

If we want $\lim_{k\to\infty} e^{(k)} = 0$ we need $\lim_{k\to\infty} B^k = 0$.

Convergent matrices

A matrix $B \in \mathbb{R}^{n \times n}$ is convergent if

$$\lim_{k \to \infty} B^k = 0,$$

where 0 is the matrix identically zero. Then:

Lemma 1

Let $B \in \mathbb{R}^{n \times n}$. We have

$$\lim_{k \to \infty} B^k = 0 \iff \max_i |\lambda_i(B)| < 1.$$

The proof is not trivial for a generic B.

A useful property of natural norm of matrices

Lemma 2

Let $|||A|||$ be any natural norm of matrix. Then

$$\max_i |\lambda_i(A)| \leq |||A||| \quad \forall A \in \mathbb{R}^{n \times n}.$$

Proof.

Let λ be an eigenvalue of A, and let $\underline{v} \neq 0$ an eigenvector associated to λ, that is $A\underline{v} = \lambda\underline{v}$. From the properties of the norms we immediately have

$$|\lambda|\|\underline{v}\| = \|\lambda\underline{v}\| = \|A\underline{v}\| \leq |||A|||\|\underline{v}\|,$$

then $|\lambda|\|\underline{v}\| \leq |||A|||\|\underline{v}\|$, and then $|\lambda| \leq |||A|||$. □

The quantity $\max_i |\lambda_i(A)|$ is called the spectral radius of A, and denoted as $\rho(A)$.

the matrix $|||\cdot|||_\infty$ norm

Given $B \in \mathbb{R}^{n\times n}$, $\quad |||B|||_\infty := \sup_{v\in\mathbb{R}^n} \frac{\|Bv\|_\infty}{\|v\|_\infty} = \max_i \sum_{j=1}^n |B_{ij}|.$

proof of the last equivalence:

$$\|Bv\|_\infty = \max_i |(Bv)_i| = \max_i \left|\sum_{j=1}^n B_{ij} v_j\right|$$

$$\leq \max_i \sum_j |B_{ij}||v_j| \leq \|v\|_\infty \max_i \sum_j |B_{ij}|$$

therefore $\quad \frac{\|Bv\|_\infty}{\|v\|_\infty} \leq \max_i \sum_j |B_{ij}|$

and $\quad \sup_{v\in\mathbb{R}^n} \frac{\|Bv\|_\infty}{\|v\|_\infty} \leq \max_i \sum_j |B_{ij}|$

the matrix $|||\cdot|||_\infty$ norm

on the other hand, let i^* the index such that

$$\sum_j |B_{i^*j}| = \max_i \sum_j |B_{ij}|$$

then, selecting $w_j = \operatorname{sign} B_{i^*j}$, since $\|w\|_\infty = 1$, we have

$$\max_i \sum_j |B_{ij}| - \sum_j |B_{i^*j}| = \sum_j B_{i^*j} w_j = \left| \sum_j B_{i^*j} w_j \right|$$

$$\leq \max_i \left| \sum_j B_{ij} w_j \right| = \|Bw\|_\infty$$

$$= \frac{\|Bw\|_\infty}{\|w\|_\infty} \leq \sup_{v \in \mathbb{R}^n} \frac{\|Bv\|_\infty}{\|v\|_\infty} = |||B|||_\infty$$

Classes of matrices for which we have convergence results

Lemma 3

If A is diagonally dominant, i.e.,

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad \forall i = 1, 2, \cdots, n$$

both Jacobi and Gauss-Seidel converge.

Proof.

We shall prove the Lemma for Jacobi method. The iteration matrix B_J is given by

$$B_J = \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \cdots & -\frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \cdots & 0 \end{bmatrix}$$

Since A is diagonally dominant, $|||B_J|||_\infty = \max_i \sum_{j \neq i} |\frac{a_{ij}}{a_{ii}}| < 1$, and we deduce (from Lemma 1) that $\max_i |\lambda_i(B_J)| < 1$. □

Lemma 4

If A is symmetric and positive definite Gauss-Seidel converges. Jacobi might or might not converge.

Stopping criteria

As usual with iterative procedures, we need sound stopping criteria to decide when to stop. Given a tolerance τ for example $\sim 10^{-3}$, or 10^{-4})

- **test on the iterates:** at each iteration check if

$$\frac{\|\underline{x}^{(k)} - \underline{x}^{(k-1)}\|}{\|\underline{x}^{(k-1)}\|} \leq \tau$$

for some norm of vectors;

- **test on the residual:** when the test on the iterates is satisfied, check if

$$\frac{\|\underline{r}^{(k)}\|}{\|\underline{b}\|} \leq \tau \qquad (\underline{r}^{(k)} := \underline{b} - A\underline{x}^{(k)} \text{ is the residual})$$

When both are satisfied, stop and take $\underline{x}^{(k)}$ as solution.

Pseudocode for splitting methods

$$M\underline{x}^{(k)} = \underline{b} + N\underline{x}^{(k-1)} = \underline{b} + (M - A)\underline{x}^{(k-1)}$$
$$= \underline{b} - A\underline{x}^{(k-1)} + Mx^{(k-1)}$$

$$\Longrightarrow \underline{x}^{(k)} = \underline{x}^{(k-1)} + M^{-1}\underline{r}^{(k-1)}$$

M is usually referred as a *preconditioner*.

Splitting ietartive method

Input: $A \in \mathbb{R}^{n\times n}$ and $\underline{b} \in \mathbb{R}^n$
Choose $M \in \mathbb{R}^{n\times n}$, $\underline{x}^{(0)} \in \mathbb{R}^n$ and set $\underline{r}^{(0)} = b - A\underline{x}^{(0)}$
for $k = 1, 2, \ldots,$ until convergence:
    Solve $M\underline{p}^{(k-1)} = \underline{r}^{(k-1)}$
    $\underline{x}^{(k)} = \underline{x}^{(k-1)} + \underline{p}^{(k-1)}$
    $\underline{r}^{(k)} = \underline{b} - A\underline{x}^{(k)}$
end

Error analysis

Unfortunately, the fact that the residual is small does not guarantee that the error $\underline{x} - \underline{x}^{(k)}$ is small.

$$\underline{r}^{(k)} := \underline{b} - A\underline{x}^{(k)} = A\underline{x} - A\underline{x}^{(k)} \longrightarrow \underline{x} - \underline{x}^{(k)} = A^{-1}\underline{r}^{(k)}.$$

Taking the norm in both sides we have

$$\begin{aligned}\|\underline{x} - \underline{x}^{(k)}\| = \|A^{-1}\underline{r}^{(k)}\| &\leq |||A^{-1}|||\,\|\underline{r}^{(k)}\| \\ &\leq |||A^{-1}|||\,\frac{\|\underline{r}^{(k)}\|}{\|\underline{b}\|}\,\|A\underline{x}\| \leq |||A^{-1}|||\,|||A|||\,\|\underline{x}\|\frac{\|\underline{r}^{(k)}\|}{\|\underline{b}\|}.\end{aligned}$$

Then we obtain

$$\frac{\|\underline{x} - \underline{x}^{(k)}\|}{\|\underline{x}\|} \leq |||A^{-1}|||\,|||A|||\frac{\|\underline{r}^{(k)}\|}{\|\underline{b}\|}.$$

If the number $\kappa(A) := |||A^{-1}|||\,|||A|||$ is big there is no control on the error, no matter how small the residual is. $\kappa(A)$ is called "condition number of A", and if $\kappa(A) >> 1$ the matrix is said to be ill-conditioned.

Concept of conditioning

When dealing with ill-conditioned matrices, any numerical method (direct or iterative) might produce unsatisfactory results.

Roughly speaking, a problem is *well-conditioned* if "small" perturbations on the data determine "small" perturbations on the results.

To clarify the concept of *conditioning* of a problem, let us consider a generic problem: find u solution of

$$(P) \qquad\qquad F(u, d) = 0,$$

where d are the data, and F is the law relating u to d.

Concept of conditioning

More precisely, let u be the solution of the problem

$$(P) \qquad F(u, d) = 0$$

corresponding to data d, and let δd be a perturbation on the data. Denote by δu the corresponding perturbation on the solution u. Then, instead of solving (P) we are solving

$$(\widetilde{P}) \qquad F(u + \delta u, d + \delta d) = 0.$$

Assuming (P) is well-posed (that is, there exists a unique the solution for any given datum), we define its (relative) condition number as the smallest constant $\kappa > 0$ that satisfies

$$\frac{\|\delta u\|}{\|u\|} \leq \kappa \frac{\|\delta d\|}{\|d\|}$$

Example: conditioning of the linear system $A\underline{x} = \underline{b}$

Consider a simple case: assume that the possible errors are only on the right-hand side (and not on the matrix). Let $\delta\underline{b}$ be the error on $\underline{b}$, and let $\widetilde{\underline{x}} = \underline{x} + \delta\underline{x}$ be the solution for the right-hand side $\widetilde{\underline{b}} = \underline{b} + \delta\underline{b}$, that is:

$$A\underline{x} = \underline{b} \quad \text{and} \quad A(\underline{x} + \delta\underline{x}) = \underline{b} + \delta\underline{b},$$

subtracting we get $A\delta\underline{x} = \delta\underline{b}$ and therefore $\delta\underline{x} = A^{-1}\delta\underline{b}$. Proceeding as we did before we have

$$\|\delta\underline{x}\| = \|A^{-1}\delta\underline{b}\| \leq |||A^{-1}|||\|\delta\underline{b}\| = |||A^{-1}|||\frac{\|\delta\underline{b}\|}{\|\underline{b}\|}\|\underline{b}\|$$

$$= |||A^{-1}|||\frac{\|\delta\underline{b}\|}{\|\underline{b}\|}\|A\underline{x}\| \leq |||A^{-1}|||\frac{\|\delta\underline{b}\|}{\|\underline{b}\|}|||A|||\|\underline{x}\|$$

We found

$$\frac{\|\delta \underline{x}\|}{\|\underline{x}\|} \leq |||A^{-1}||| \, |||A||| \frac{\|\delta \underline{b}\|}{\|\underline{b}\|} = \kappa(A) \frac{\|\delta \underline{b}\|}{\|\underline{b}\|}$$

A simple example to understand how a big condition number might affect the results.

$$\begin{pmatrix} 10^6 & 10^{-12} \\ 0 & 10^{-6} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10^6 \\ 10^{-6} \end{pmatrix}$$

Exact solution $x_2 = 1, x_1 \simeq 1; \kappa_\infty(A) \simeq 10^{12}$. Now perturb only the first component of the right-hand side by 10^{-6}, and then only the second component by 10^{-6}. In both cases $\|\delta \underline{b}\|_\infty / \|\underline{b}\|_\infty \leq 10^{-12}$. What happens to the solution?

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

$$A\underline{x}^* = \underline{b} \tag{2}$$

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

$$(A\underline{x}, \underline{x}) > 0 \quad \forall \underline{x} \not\equiv \underline{0}.$$

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

$$F(\underline{v}) := \frac{1}{2}(A\underline{v}, \underline{v}) - (\underline{b}, \underline{v}) \qquad \forall \underline{v} \in \mathbb{R}^n \tag{3}$$

Theorem 5

If $A \in \mathbb{R}^n \times \mathbb{R}^n$ is SPD, problem (2) has a unique solution, and is equivalent to the following minimum problem for the functional defined in (3):

$$\begin{cases} \text{find } \underline{u} \in \mathbb{R}^n \text{ such that} \\ F(\underline{u}) \leq F(\underline{v}) \qquad \forall \underline{v} \in \mathbb{R}^n \end{cases} \tag{4}$$

(that is, (4) has a unique solution $\underline{u} \in \mathbb{R}^n$, and $\underline{u} \equiv \underline{x}^*$).

Proof.

Since A is positive definite, problem (2) has a unique solution ($det(A) \neq 0$). Now, F is a quadratic functional (hence, differentiable), and

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

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

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

Descent Methods

Given the equivalence between the linear system (2) and the minimum problem (4), 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:

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

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

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

With respect to the variable α, this function is an U-shaped parabola (it has a unique minimum).

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

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

Gradient method: the "steepest descent"

- the gradient $\nabla F(\underline{x}^{(k)})$ gives the direction and rate of fastest increase at a point $\underline{x}^{(k)}$. Since we want to minimize, it make sense to go in the direction of fastest decrease, that is, steepest descent

$$\underline{p}^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)}) = b - A\underline{x}^{(k)} = \underline{r}^{(k)}$$

- The steepest descent method converges for all initial guess $\underline{x}^{(0)}$. 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 $\|v\|_A = \sqrt{v^T A v}$ is the $A-$norm. The formula above says that convergence is guaranteed, but can be very slow if A is ill-conditioned (indeed $\kappa_2(A)$ very large means that $\left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)$ is very close to 1).

Pseudocode for Steepest Descent Method

Steepest Descent Method

Input: $A \in \mathbb{R}^{n \times n}$ SPD, $\underline{b} \in \mathbb{R}^n$, $\underline{x}^{(0)} \in \mathbb{R}^n$, $tol \in \mathbb{R}^+$, maxiter $\in \mathbb{N}$

$\underline{r}^{(0)} = \underline{b} - A\underline{x}^{(0)}$

for $k = 1, 2, \ldots,$ maxiter:

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

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

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

$\underline{r}^{(k)} = \underline{b} - A\underline{x}^{(k)} = \underline{r}^{(k-1)} - \alpha_{k-1}\underline{y}$

If Stopping criteria are satisfied exit the loop

end

Output: $\underline{x}^{(k)}$

Like for all iterative methods, the dominant computational cost at each iteration is given by the matrix-vector product with A, that costs about $2n^2$ FLOPs (n^2 multiplications and $\sim n^2$ sums).

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 $p^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)}) = b - A\underline{x}^{(k)}$. Possible alternatives are:

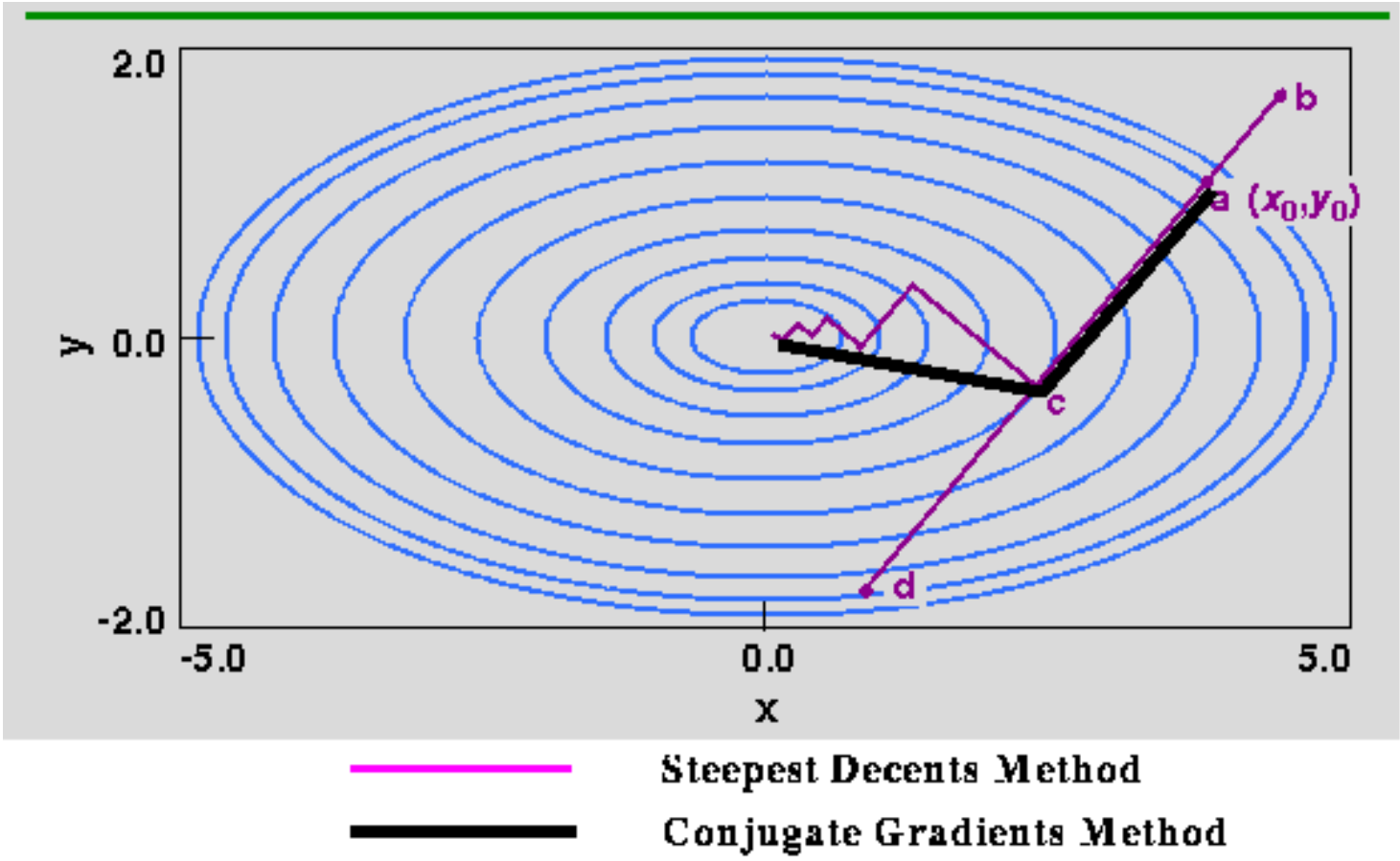
- to simplify the calculation of $p^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)})$, e.g. in the **stochastic gradient descent** method, used in machine learning: we save time per each iteration at the expenses of an increased number of iterations to reach a given accuracy;
- to find better descent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

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 $p^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)}) = b - A\underline{x}^{(k)}$. Possible alternatives are:

- to simplify the calculation of $p^{(k)} = -\underline{\nabla} F(\underline{x}^{(k)})$, 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 $p^{(k)}$, such that the convergence at a given tolerance requires less 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 $span\{\underline{r}^{(k)}, \underline{p}^{(k-1)}\}$

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

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

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

This approach is faster than the steepest descent. Actually, the method converges in less than n iterations (n=dimension of the system), so it can be considered a direct method.

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

Pseudocode for Conjugate Gradient Method

Conjugate Gradient Method

Input: $A \in \mathbb{R}^{n \times n}$ SPD, $\underline{b} \in \mathbb{R}^n$, $\underline{x}^{(0)} \in \mathbb{R}^n$, $tol \in \mathbb{R}^+$, maxiter $\in \mathbb{N}$

$\underline{r}^{(0)} = \underline{b} - A\underline{x}^{(0)}$

$\underline{p}^{(0)} = \underline{r}^{(0)}$

for $k = 1, 2, \ldots,$ maxiter:

$\underline{y} = A\underline{p}^{(k-1)}$

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

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

$\underline{r}^{(k)} = \underline{b} - A\underline{x}^{(k)} = \underline{r}^{(k-1)} - \alpha_{k-1}\underline{y}$

$\beta_{k-1} = \left(\underline{y}, \underline{r}^{(k)}\right) / \left(\underline{y}, \underline{p}^{(k-1)}\right)$

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

If Stopping criteria are satisfied exit the loop

end

Output: $\underline{x}^{(k)}$

Convergence of the Conjugate Gradient Method

We have the following bound on the relative error for CG:

$$\left\|x - x^{(k)}\right\|_A \leq 2\left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}\right)^k \left\|x - x^{(0)}\right\|_A$$

recalling that, when A is SPD, it holds

$$\kappa_2(A) := \|A\| \left\|A^{-1}\right\| = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

Then the larger $\kappa_2(A)$ is, the slower the method converges, however, it should be noted that if we compare:

- the reducing factor of k Steepest Descend iterations: $\left(\frac{\kappa_2(A)-1}{\kappa_2(A)+1}\right)^k$ and
- the reducing factor of k Conjugate Gradient iter.: $2\left(\frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1}\right)^k$

we see that Conjugate Gradient is more favourable.

Preconditioners

To speedup the convergence, we can use a preconditioner. In this case, the original linear system $Ax = b$ with the equivalent one

$$P^{-1}Ax = P^{-1}b$$

where P is a nonsingular matrix called preconditioner.

A good preconditioner has two features:

- An iterative method applied to the new system should converge in less iterations than for the original system. This typically means that the eigenvalues of $P^{-1}A$ should be clustered (in the case of CG, this means $\lambda_{\max}(P^{-1}A) \approx \lambda_{\min}(P^{-1}A)$).

- At each iteration of an iterative method we need to compute a matrix-vector product with the system matrix. In the preconditioned case, this is done in two steps

$$v \longrightarrow Av \longrightarrow P^{-1}(Av)$$

Thus, computing matrix-vector products with P^{-1} (or equivalently solving a linear system with P) should be fast. Note that we never need to compute the matrix $P^{-1}A$ explicitly.

Preconditioners

- Hence a good preconditioner P should be as similar as possible to A, while being easy to invert:
- Let us consider two extreme cases: $P = I_n$ and $P = A$
  - If $P = A$, $P^{-1}A = I_n$ and any iterative method would converge in just 1 iteration (note that $\lambda_{\max}(P^{-1}A) = \lambda_{\min}(P^{-1}A) = 1$). On the other hand, applying P^{-1} is as difficult as solving the original system.
  - If $P = I_n$, then applying P^{-1} has no cost. On the other hand $P^{-1}A = A$, so there is no reduction in the number od iterations.
- A good preconditioner should find a balance between these two extremes.

Preconditioning for CG

- In the case of CG, similarly as A is required to be SPD, also the preconditioner is required to be SPD.
- In general, the problem of finding a good preconditioner is very problem-specific.
- Some black-box preconditioners:
  - **Jacobi:** $P = \text{diag}(A)$.
  - **Symmetric Gauss-Seidel:** $P = L_* \text{diag}(A)^{-1} L_*^T$ where $L_* = \text{tril}(A)$.
  - **Incomplete Cholesky:** An approximated Cholesky factorisation, where no fill-in is introduced:

    $$P = LL^T \approx A, \quad \text{such that if } A_{ij} = 0 \Longrightarrow L_{ij} = 0$$

    In other words, we impose that L has the same sparsity pattern as A.

Sparse matrices and iterative solvers

Recall that:

- The **sparsity** of an $n \times n$ matrix A is

$$\frac{\text{nnz}(A)}{n^2}$$

  where $\text{nnz}(A) = \#$ of nonzero entries in A. A matrix is **sparse** if its sparsity is $\ll 1$.
- Sparse matrices are extremely common in engineering and computer science, e.g., Network theory, data analysis and machine learning, discretization of differential equations.
- direct solvers suffer from the fill-in phenomenon.

Sparse matrices and iterative solvers

- Recall that at each iteration of an iterative we have to compute a matrix-vector product

$$v \longrightarrow Av$$

If A is sparse, only the nonzero entries of A are involved in the computation:

$$(Av)_i = \sum_{j=1}^{n} a_{ij} v_j = \sum_{j \; s.t. \; a_{ij} \neq 0}^{n} a_{ij} v_j$$

The cost of a matrix-vector product is then $2\text{nnz}(A)$, versus $2n^2$ for dense matrices.

- Iterative solvers do not suffer from fill-in. In particular, the main memory consumption is just the storing of A. Hence iterative methods typically require much less memory than direct methods.

Summary on Linear Systems

available solvers for $A\underline{x} = \underline{b}$, with $A \in \mathbb{R}^{n \times n}$ non singular...

Direct Methods

Methods	Requirements on A	Cost
GEM / LU	[1] $\det(A_i) \neq 0,\ i = 1, \ldots, n$	$\sim 2/3\ n^3$ FLOPs
GEM / LU + Pivoting	none	$\sim 2/3\ n^3$ FLOPs

[1] A_i is the matrix obtained considering only the first i rows and the first i columns of A. This condition is automatically satisfied if A is diagonally dominant or if A is SPD.

Iterative Methods ($\sim 2n^2$ FLOPs for each iteration)

Methods	Requirements on A	Sufficient conditions for convergence
Jacobi	$A_{ii} \neq 0,\ i = 1, \ldots, n$	A diagonally dominant
Gauss-Seidel	$A_{ii} \neq 0,\ i = 1, \ldots, n$	A diagonally dominant or A SPD
Steepest Descent Method	A SPD	always ensured
Conjugate Gradient Method	A SPD	always ensured in less than n iterations