ITERATIVE METHODS FOR LINEAR SYSTEMS OF EQUATIONS

Let us consider the following linear system of equations

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

with

$A \in \mathbb{R}^{n \times n}$

$\boldsymbol{b} \in \mathbb{R}^{n}$

$\boldsymbol{x} \in \mathbb{R}^{n}$

$$\boxed{det(A) \neq 0}$$

In general, direct methods (i.e., methods based on a “manipulation” of A”) are not suitable whenever:

- n is large. Except for selected cases, the average cost of direct methods scale as n^3.

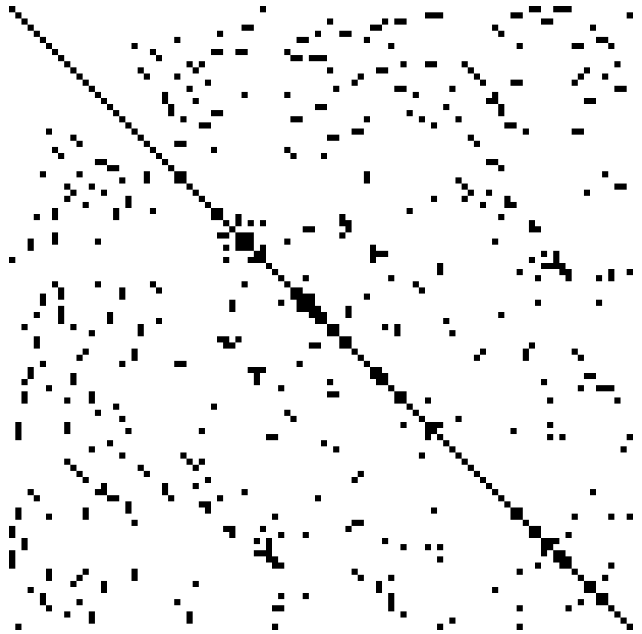
  For example: if the peak performance is 1 PetaFLOPS (10^{15} floating point operations per second), then

$$n = 10^7 \longrightarrow \approx 10^6 s = 11 \text{ days}$$

- A is sparse. Direct methods suffer from fill-in phenomenon (see later). Sparse matrices arise in many application problems, for example the approximation with Finite Elements or Finite Differences of partial differential equations

Definition of a sparse matrix A

Let $A \in \mathbb{R}^{n \times n}$, we say that A is sparse the number of non-zero elements (nnz(A)) is roughly equal to the number of rows/columns n, i.e. nnz(A) $\sim n$



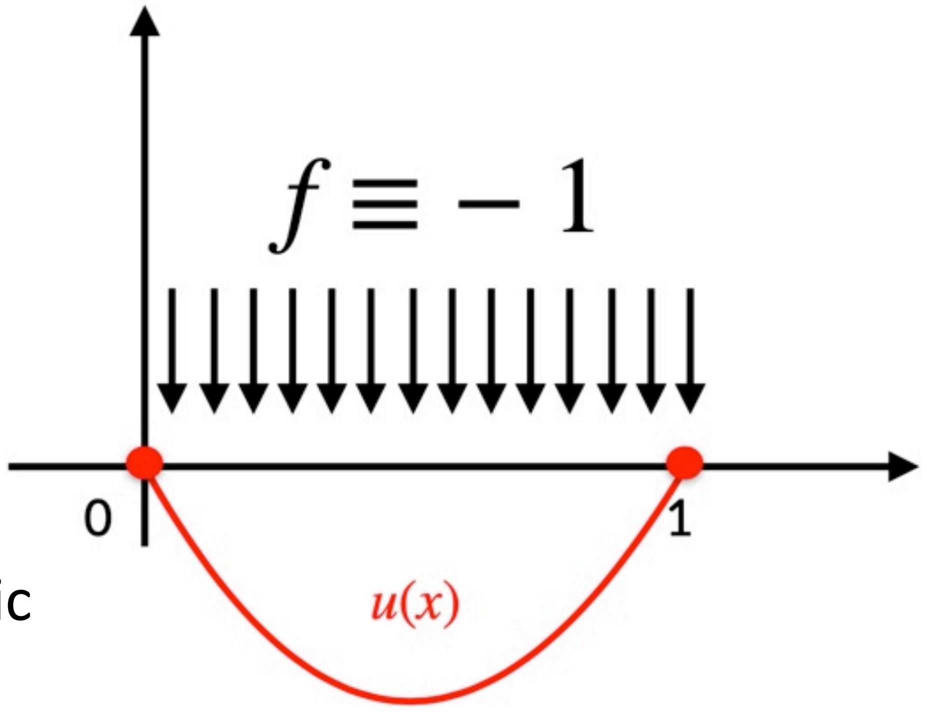
A sparse matrix obtained when solving a finite element problem in two dimensions. The non-zero elements are shown in black.

Examples

Example

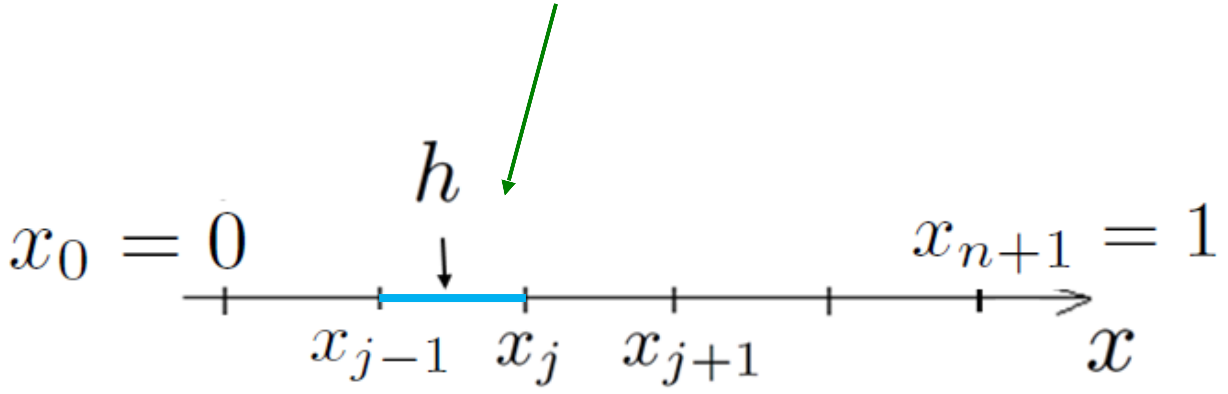
Let us consider the following differential problem:

$$-u''(x) = f(x) \qquad x \in (0,1)$$
$$u(0) = u(1) = 0$$



which describes (for example) the displacement of an elastic body constrained to the extremes under the action of a force *f(x)*

Computational domain discretization



Example

In each node x_j we can write the following approximate problem, obtained by approximating the second derivative and introducing the numerical solution u_j

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f(x_j)$$

It is easy to verify that the previous set of equations is a linear system

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \vdots \\ 0 & \cdots & 0 & 0 & -1 & 2 \end{pmatrix} \qquad \mathrm{b} = h^2 \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_j) \\ \vdots \\ f(x_n) \end{bmatrix}$$

sparse: $\sim 3n$ non-zero elements

A is SPD (exercise)

Definition of iterative methods

Definition

We introduce a sequence $\mathbf{x}^{(k)}$ of vectors determined by a recursive relation that identifies the method.

In order to "initialise" the iterative process, it is necessary to provide a starting point (initial vector) $\mathbf{x}^{(0)}$based for example on physical / engineering applications

$$\mathbf{x}^{(0)} \longrightarrow \mathbf{x}^{(1)} \longrightarrow \ldots \mathbf{x}^{(k)} \longrightarrow \mathbf{x}^{(k+1)} \longrightarrow \ldots$$

For the method to make sense, it must satisfy the convergence property:

$$\boxed{\lim_{k \to +\infty} \mathbf{x}^{(k)} = \mathbf{x}}$$

Convergence must not depend on the choice of $\mathbf{x}^{(0)}$

What do we expect from the accuracy of an iterative method?

Since convergence is only guaranteed after an infinite number of iterations, from a practical point of view we will have to stop the iterative process after a finite number of iterations, when we believe we have arrived "sufficiently close" to the solution.

For the choice of specific stopping criteria, see below

We can therefore conclude that even in exact arithmetic, an iterative method will inevitably be affected by a numerical error

The Jacobi and Gauss-Seidel methods

The Jacobi method

Let's start from the i-th line of the linear system:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \rightarrow a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i$$

Formally the solution x_i for each i is given by:

$$x_i = \frac{b_i - \sum_{j \neq i} a_{ij} x_j}{a_{ii}}$$

Obviously the previous identity cannot be used in practice because we do not know x_j, for $j \neq i$

However, we could think of introducing an iterative method that updates $x_i^{(k+1)}$ step $k+1$ using the others $x_j^{(k)}$ obtained in the previous step k

The Jacobi method

$$\boxed{x_i^{(k+1)} = \frac{b_i - \sum_{j\neq i} a_{ij} x_j^{(k)}}{a_{ii}}} \qquad \forall i = 1, \ldots, n$$

In general, each iteration costs $\sim n^2$operations, so the Jacobi method is competitive if the number of iterations is less than n

If A is sparse matrice, then the cost is only $\sim n$ flops per iteration!

It should be noted that the solutions $x_i^{(k+1)}$ can be computed fully in parallel (very competitive for large scale systems).

The Gauss Seidel method

Let's start from Jacobi's method: $$x_i^{(k+1)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}}{a_{ii}}$$

At iteration $(k+1)$, let's consider the computation of $x_i^{(k+1)}$. We observe that for $j < i \, (i \geq 2)$, $x_j^{(k+1)}$ is known (we have already calculated it). We can therefore think of using the quantities at step $(k+1)$ if $j < i$ and (as in the Jacobi method) those at the previous step k if $j > i$:

$$\Downarrow$$

$$\boxed{x_i^{(k+1)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}}{a_{ii}}}$$

The computational costs are comparable to those of the Jacobi method

Unlike Jacobi method, GS method is not fully parallelizable

Linear iterative methods

Linear iterative methods

$\boxed{A\mathbf{x} = \mathbf{b}}$

In general, we consider linear iterative methods of the following form:

$$\mathbf{x}^{(k+1)} = B\mathbf{x}^{(k)} + \mathbf{f} \qquad k \geq 0$$

where $B \in \mathbb{R}^{n\times n}, \mathbf{f} \in \mathbb{R}^n$.

B is called iteration matrix. Its choice (together with that of $\mathbf{f}$) uniquely identify the method.

How to choose $B \in \mathbb{R}^{n\times n}, \mathbf{f} \in \mathbb{R}^n$?

1. **Consistency**: if $\mathbf{x}^{(k)}$ is the exact solution $\mathbf{x}$, then $\mathbf{x}^{(k+1)}$ is again equal to $\mathbf{x}$ (no update if the exact solution is found):

$$\boldsymbol{x} = B\boldsymbol{x} + \boldsymbol{f} \quad \rightarrow \quad \boxed{\boldsymbol{f} = (I - B)\boldsymbol{x} = (I - B)A^{-1}\boldsymbol{b}}$$

The former identity gives a relationship between B and $\mathbf{f}$ as a function of the data. It provides a necessary condition for convergence (but it into that is not sufficient, see later)

Linear iterative methods

2. **Convergence**:

Let us introduce the error at step $(k+1)$: $\mathbf{e}^{(k+1)} = \mathbf{x} - \mathbf{x}^{(k+1)}$ and a suitable vector norm $\|\cdot\|$ (for example the Euclidean norm). Then we have

$$\begin{aligned} \|\boldsymbol{e}^{(k+1)}\| &= \|\boldsymbol{x} - \boldsymbol{x}^{(k+1)}\| = \|\boldsymbol{x} - B\boldsymbol{x}^{(k)} - \boldsymbol{f}\| = \xleftarrow{\quad} \text{consistency} \\ &= \|\boldsymbol{x} - B\boldsymbol{x}^{(k)} - (I-B)\boldsymbol{x}\| = \|B\boldsymbol{e}^{(k)}\| \leq \\ &\leq \|B\| \, \|\boldsymbol{e}^{(k)}\| \end{aligned}$$

Note that $\|B\|$ is the matrix norm induced by the vector norm $\|\cdot\|$

By recursion we obtain

$$\begin{aligned} \|\boldsymbol{e}^{(k+1)}\| &\leq \|B\| \, \|B\| \, \|\boldsymbol{e}^{(k-1)}\| \leq \\ &\leq \|B\| \, \|B\| \, \|B\| \, \|\boldsymbol{e}^{(k-2)}\| \leq \ldots \leq \|B\|^{k+1} \, \|\boldsymbol{e}^{(0)}\| \end{aligned}$$

$$\lim_{k\to\infty} \|\boldsymbol{e}^{(k+1)}\| \leq \left(\lim_{k\to\infty} \|B\|^{k+1}\right) \|\boldsymbol{e}^{(0)}\|$$

If $\boxed{\|B\| < 1} \Longrightarrow \lim_{k\to\infty} \|\boldsymbol{e}^{(k+1)}\| = 0$

Sufficient condition for convergence

$$\rho(B) = \max_j |\lambda_j(B)|$$

We define $\rho(B) = \max_j |\lambda_j(B)|$, where $\lambda_j(B)$ are the eigenvalue of B

$\rho(B)$ is the spectral radius of B

Remark. If B is SPD, then $\rho(B) = \|B\|_2$

Property. Let $C \in \mathbb{R}^{n\times n}$ then $\rho(C) = \inf\{\,\|C\| \quad \forall$ induced matrix norm $\|\cdot\|\}$

From Property 1, it follows

$$\rho(B) \leq \|B\| \quad \forall \text{ induced matrix norm} \|\cdot\| \qquad (1)$$

Theorem (necessary and sufficient condition for convergence):

A consistent iterative method with iteration matrix B converges if and only if $\rho(B) < 1$

Remark. Thanks to (1) we can observe that if

$$\exists \|\cdot\| \text{ such that } \|B\| < 1 \longrightarrow \rho(B) < 1 \text{ (and thus convergence)}$$

Convergence of Jacobi (J) e Gauss-Seidel (GS) methods

Let

- D: the diagonal part of A
- $-E$: lower triangular part of A
- $-F$: upper triangular part of A

$$A = \begin{bmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{bmatrix}$$

The Jacobi method can be rewritten as

$$D\boldsymbol{x}^{(k+1)} = (E+F)\boldsymbol{x}^{(k)} + \boldsymbol{b}$$

Its iteration matrix is given by

$$B_J = D^{-1}(E+F) = D^{-1}(D-A) = I - D^{-1}A$$

Convergence of Jacobi e Gauss-Seidel methods

For the Gauss-Seidel method we have:

$$(D-E)\boldsymbol{x}^{(k+1)} = F\boldsymbol{x}^{(k)} + \boldsymbol{b}$$

$$A = \begin{bmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{bmatrix}$$

The iteration matrix is given by

$$B_{GS} = (D-E)^{-1}F$$

Both methods are consistent (exercise)

Theorem (sufficient condition for convergence)

If A is strictly diagonally dominant by rows, i.e.

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}|, \ i = 1, \ldots, n,$$

then J e GS converge

For the Jacobi method

$$-D^{-1}A = \begin{bmatrix} -\frac{a_{11}}{a_{11}} & & & & \\ & \ddots & & -\frac{a_{ij}}{a_{ii}} & \\ & & \ddots & & \\ & -\frac{a_{ij}}{a_{ii}} & & \ddots & \\ & & & & -\frac{a_{nn}}{} \end{bmatrix} \Rightarrow B_J = I - D^{-1}A = \begin{bmatrix} 0 & & & \\ & \ddots & & -\frac{a_{ij}}{a_{ii}} \\ & & \ddots & \\ & -\frac{a_{ij}}{a_{ii}} & & \ddots \\ & & & & 0 \end{bmatrix}$$

I take the following norm $\|C\| = \|C\|_1 = \max_i \left(\sum_j |C_{ij}| \right)$

I obtain $\|B_J\|_1 = \max_i \left(\sum_j |(B_J)_{ij}| \right) = \max_i \left(\sum_j \left| \frac{a_{ij}}{a_{ii}} \right| \right)$

from which it follows $\|B_J\| < 1$ since A is strictly diagonally

Theorem (sufficient conditions for convergence)

- If A is strictly diagonally dominant by columns, then J e GS methods are convergent.
- If A is SPD then the the GS method is convergent
- If A is tridiagonal it can be shown that

$$\rho^2(B_J) = \rho(B_{GS})$$

therefore both methods converge or fail to converge at the same time. In the former case GS method is more rapidly convergent than the J method.

Example:

$$A = \begin{bmatrix} 3 & -2 & 0 & & \\ -1 & 3 & -2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & \\ & & & -1 & 3 \end{bmatrix}$$

A is not is strictly diagonally dominant by row/columns (3=|-2-1|)

A is not symmetric

Can we claim convergence? A priori no! The theorem states sufficient conditions for convergence!

Compute $\rho(B_J)$ and observe that $\rho(B_J) < 1$. Therefore the Jacobi method converges. Moreover, since A is tridiagonal, we can also conclude that the Gauss-Seidel method converges (and it converges faster that the Jacobi method)

STOPPING CRITERIA

A practical test is needed to determine when to stop the iteration

Stopping criteria

user-defined tolerance

Idea: we terminate the iterations when $\boxed{\dfrac{\|\mathbf{x}-\mathbf{x}^k\|}{\|\mathbf{x}^k\|} \le \varepsilon}$

Unfortunately the error is not know!

1. Residual-based stopping criterion.

$$\frac{\|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}\|} \le K(A)\frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{b}\|} \Longrightarrow \boxed{\frac{\|\boldsymbol{r}^{(k)}\|}{\|\boldsymbol{b}\|} \le \varepsilon}$$

Good stopping criterion whenever $\kappa(A)$ is “small”

Usually employed for preconditioned scheme

$$\frac{\|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}\|} \le K(P^{-1}A)\frac{\|\boldsymbol{z}^{(k)}\|}{\|\boldsymbol{b}\|} \Longrightarrow \boxed{\frac{\|\boldsymbol{z}^{(k)}\|}{\|\boldsymbol{b}\|} \le \varepsilon} \qquad \boldsymbol{z}^{(k)} = P^{-1}\boldsymbol{r}^{(k)}$$

Stopping criteria

2. Distance between consecutive iterates. Define $\boldsymbol{\delta}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$

$$\boxed{\|\boldsymbol{\delta}^{(k)}\| \leq \varepsilon}$$

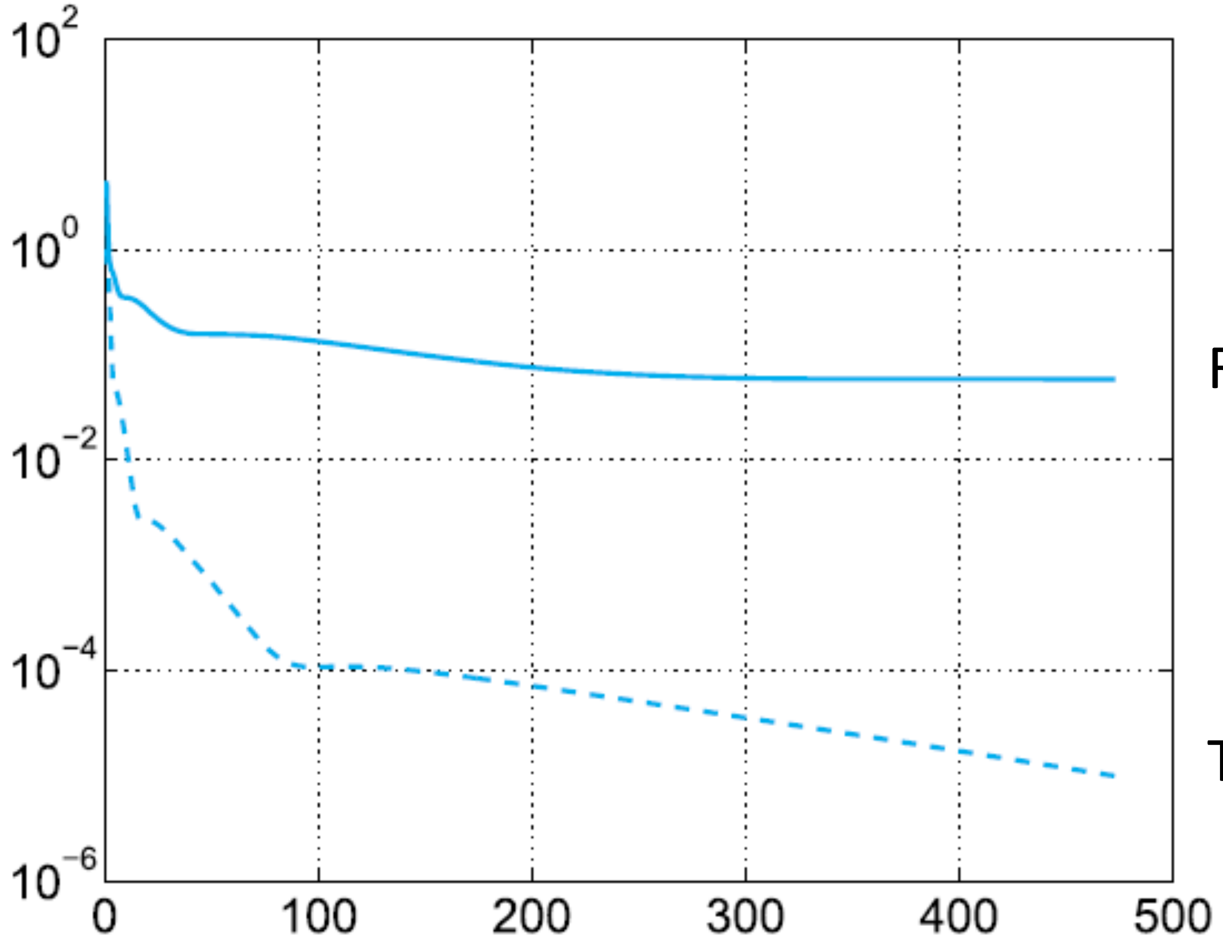
It can be shown: $\boxed{\|\mathbf{e}^{(k)}\| \leq \dfrac{1}{1-\rho(\mathrm{B})}\|\boldsymbol{\delta}^{(k)}\|}$ Relation between the true error and $\delta^{(k)}$

Indeed: $$\|\boldsymbol{e}^{(k)}\| = \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| = \|\boldsymbol{x} - \boldsymbol{x}^{(k+1)} + \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\| =$$

$$= \|\boldsymbol{e}^{(k+1)} + \boldsymbol{\delta}^{(k)}\| \leq \rho(B)\|\boldsymbol{e}^{(k)}\| + \|\boldsymbol{\delta}^{(k)}\|$$

Therefore this is a “good” stopping criterion only if $\rho(B) < < 1$.

Example: Gauss-Seidel method, Hilbert matrix.



Relative residual

True error

The stationary Richardson method

The stationary Richardson method

Given $\mathbf{x}^{(0)} \in \mathbb{R}^n, \alpha \in \mathbb{R}$, It is based on the following recursive update:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha \left(\boxed{\boldsymbol{b} - A\boldsymbol{x}^{(k)}} \right) \longrightarrow \text{residual} \quad \boldsymbol{r}^{(k)}$$

The idea is to update the numerical solution by adding a quantity proportional to the residual. Indeed, it is expected that if the residual is "large" ("small"), the solution at step k should be corrected "a lot" ("slightly")

From (2) it follows $\mathbf{x}^{(k+1)} = (I - \alpha A)\mathbf{x}^{(k)} + \alpha\mathbf{b}$. Therefore, the stationary Richardson method is characterised by the iteration matrix $B_\alpha = I - \alpha A$ and by $\mathbf{f} = \alpha\mathbf{b}$

The stationary Richardson method is consistent, indeed

$$B_\alpha = I - \alpha A \qquad \mathbf{x} = \mathbf{x} + \alpha(\mathbf{b} - A\mathbf{x})$$

Stability and convergence of the stationary Richardson method:

Let A be SPD

We introduce the eigenvalues of A (which are real and positive):

$$\lambda_{max}(A) = \lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A) = \lambda_{min}(A) > 0$$

Theorem: Let A be a symmetric and positive definite matrix. The stationary Richardson method is convergent if and only if

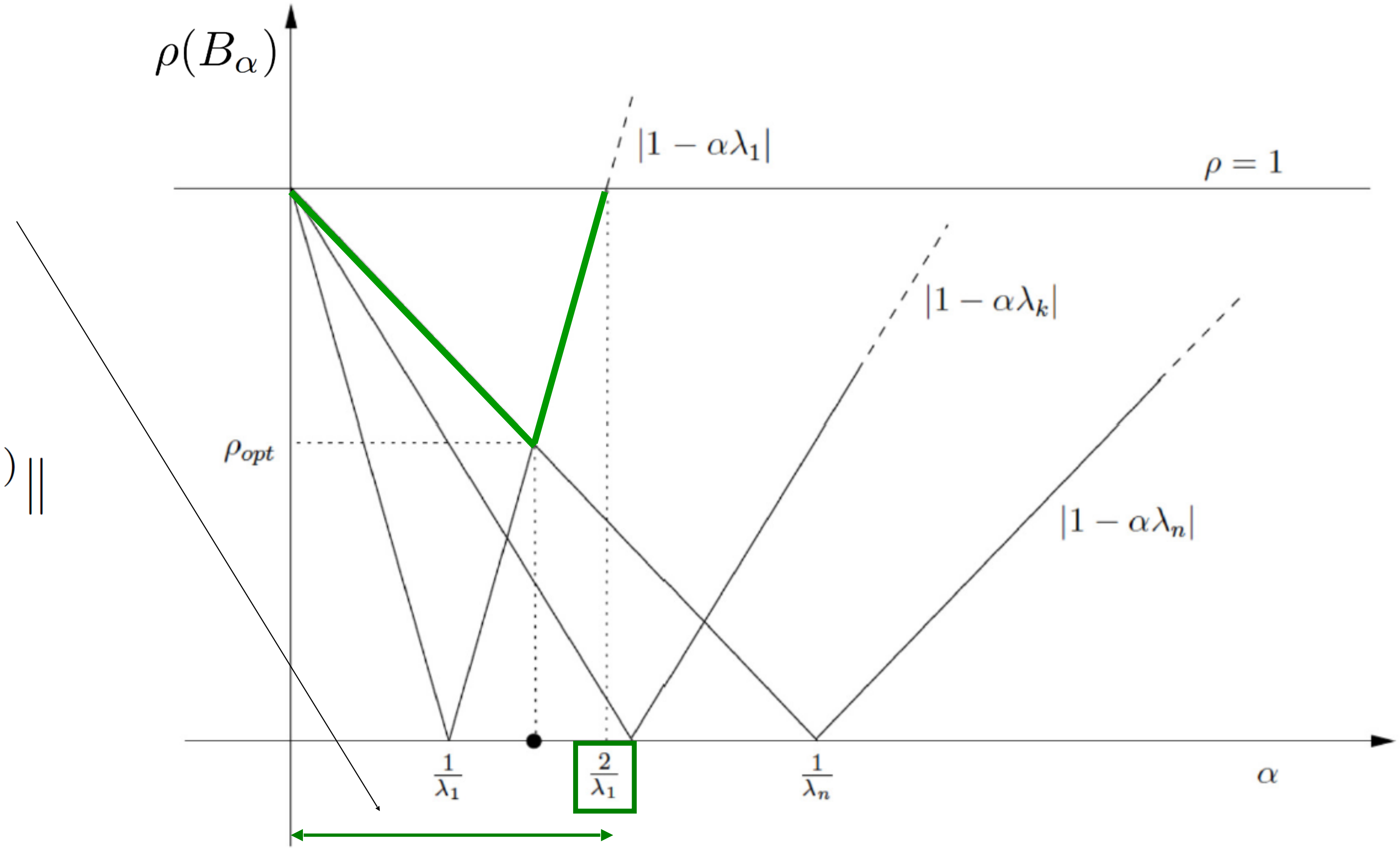
$$\boxed{0 < \alpha < \frac{2}{\lambda_{max}(A)}}$$

Sketch of the proof.

$$\boxed{0 < \alpha < \frac{2}{\lambda_{max}(A)}}$$

$$\lambda_{max}(A) = \lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A) = \lambda_{min}(A) > 0$$

$$\|e^{(k+1)}\| \quad \leq \rho(B_\alpha)\|e^{(k)}\|$$



Proof: Let $\mu_i, i = 1,\ldots,n,$ be the eigenvalues of $B_\alpha = I - \alpha A$. We have

$$\mu_i = 1 - \alpha\lambda_i, i = 1,\ldots,n$$

To have convergence it must hold $|1 - \alpha\lambda_i(A)| < 1, \ \forall i = 1,\ldots,n$

$$-1 < 1 - \alpha\lambda_i(A) < 1, \ \forall i = 1,\ldots,n$$

$$\boxed{0 < \alpha\lambda_i(A) < 2, \ \forall i = 1,\ldots,n}$$

The first bound holds if $0 < \alpha$

The second bound holds if

$$\alpha < \frac{2}{\lambda_i(A)} \quad \forall i = 1,\ldots,n$$

The thesis follows $\square$

Choice of the optimal parameter α:

We now ask ourselves what is the value of the parameter α, among those that guarantee convergence, which maximises the speed of convergence

We introduce the following A-induced norm (remember A is SPD)

$$\|\boldsymbol{z}\|_A = \sqrt{\sum_{i,j=1}^{n} a_{ij} z_i z_j} \longleftrightarrow \|\boldsymbol{z}\|_A = \sqrt{(A\boldsymbol{z}, \boldsymbol{z})} = \sqrt{\boldsymbol{z}^T A \boldsymbol{z}}$$

Recall that $\|\boldsymbol{e}^{(k+1)}\|_A \leq \rho(B_\alpha)\|\boldsymbol{e}^{(k)}\|_A$

We look for $0 < \alpha_{\text{opt}} < 2/\lambda_{\text{max}(A)}$ such that $\rho(B_\alpha)$ is minimum. That is

$$\alpha_{opt} = \underset{0<\alpha<2/\lambda_{max}(A)}{\operatorname{argmin}} \left\{ \max_i |1 - \alpha\lambda_i(A)| \right\}$$

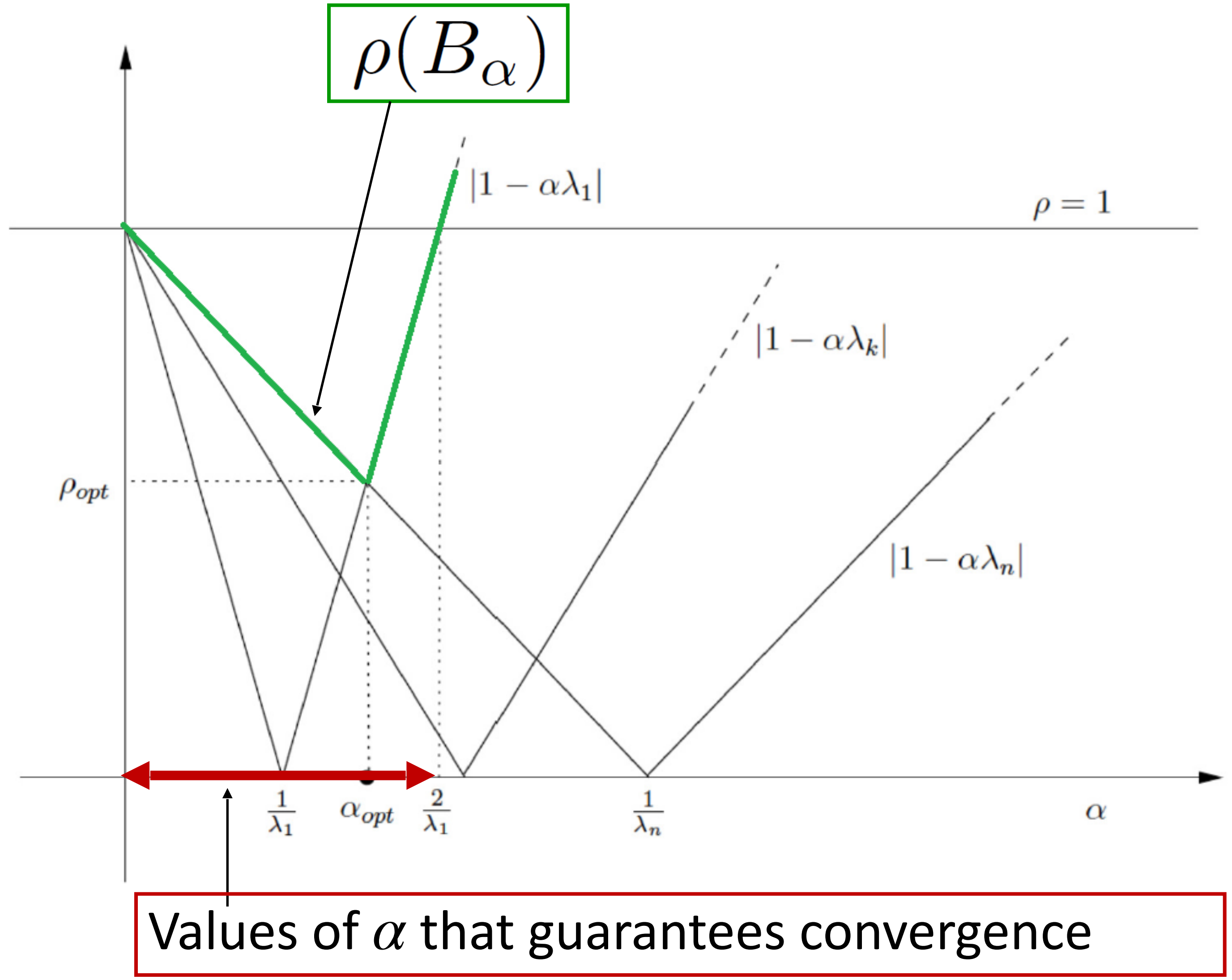
Choice of the optimal parameter α:

We plot on the x-axis the values of α and on the y-axis $|1-\alpha\lambda_i(A)|, i = 1\ldots, n$

The optimal value is given by the intersection between the curves $|1-\alpha\lambda_1(A)|$ and $|1-\alpha\lambda_n(A)|$

$$-1+\alpha\lambda_1(A) = 1-\alpha\lambda_n(A)$$

$$\boxed{\alpha_{opt} = \frac{2}{\lambda_{min}(A)+\lambda_{max}(A)}}$$



Maximum convergence speed:

We now calculate the value of the "optimal" spectral radius ρ_{opt}, at the value of the optimal parameter α_{opt}:

$$\rho_{opt} = \rho(B_{\alpha_{opt}}) = -1 + \alpha_{opt}\lambda_{max}(A) = 1 - \alpha_{opt}\lambda_{min}(A) =$$

$$= \frac{\lambda_{max}(A) - \lambda_{min}(A)}{\lambda_{max}(A) + \lambda_{min}(A)}$$

Since A is SPD, we have $\|A\|_2 = \lambda_{\text{max}}(A)$. Moreover, $\lambda_i(A^{-1}) = 1/\lambda_i(A), i = 1,\ldots,n$

$$\boxed{\rho_{opt} = \frac{K(A) - 1}{K(A) + 1}}$$

Preconditioning techniques

Condition number and speed of convergence:

The optimal value $\rho_{\text{opt}} = \dfrac{K(A) - 1}{K(A) + 1}$ expresses the maximum convergence speed that can be attained with a Stationary Richardson method.

Badly conditioned matrices ($K(A) > > 1$) are characterised by a very low convergence rate. How can we improve the speed of convergence?

IDEA. We introduce a SDP matrix P^{-1} (the so called preconditioner). Then, solving $A\mathbf{x} = \mathbf{b}$ is equivalent to the following preconditioned system:

$$P^{-1/2}AP^{-1/2}\mathbf{z} = P^{-1/2}\mathbf{b} \quad (3)$$

where $\mathbf{x} = P^{-1/2}\mathbf{z}$.

RULE OF THUMB. Choose P^{-1} such that $K(P^{-1/2}AP^{-1/2}) < < K(A)$

Preconditioned Richardson Method:

Suppose that $P^{-1}A$ has real and positive eigenvalues. We apply the stationary Richardson method to $P^{-1}A$, i.e.,

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha P^{-1}\left(\boldsymbol{b} - A\boldsymbol{x}^{k}\right) = \boldsymbol{x}^{(k)} + \alpha P^{-1}\boldsymbol{r}^{(k)}$$

We obtain the same convergence results as in the non-preconditioned case, provided we replace A with $P^{-1}A$:

Convergence:

$$0 < \alpha < \frac{2}{\lambda_{max}(P^{-1}A)}$$

Optimal values:

$$\alpha_{opt} = \frac{2}{\lambda_{min}(P^{-1}A) + \lambda_{max}(P^{-1}A)}$$

$$\rho_{opt} = \frac{K(P^{-1}A) - 1}{K(P^{-1}A) + 1}$$

Preconditioned Richardson Method:

Therefore, if $K(P^{-1}A) << K(A)$we obtain a higher convergence rate wrt the unpreconditioned case.

$$\rho_{opt} = \frac{K(P^{-1}A) - 1}{K(P^{-1}A) + 1}$$

Can we therefore say that such a preconditioned method is faster than the non-preconditioned case? No!

For the moment we can only say that the number of iterations #Iterneeded to satisfy a certain stopping criterion is lower in the preconditioned case than in the un-reconditioned one.

Indeed, the CPU time is given by

$$\text{CPU time} = \#Iter \times C$$

C= the cost of one iteration

Preconditioned Richardson Method:

Remember that $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha P^{-1}\boldsymbol{r}^{(k)}$

We defined the preconditioned residual : $\boldsymbol{z}^{(k)} = P^{-1}\boldsymbol{r}^{(k)}$

The pseudo-algorithm. For any $k = 0,1,2,3,\dots$.

```
Compute   $\alpha_{opt} = \frac{2}{\lambda_{min}(P^{-1}A) + \lambda_{max}(P^{-1}A)}$
Update    $\boldsymbol{r}^{(k)} = \boldsymbol{b} - A\boldsymbol{x}^{(k)}$
Solve     $P\boldsymbol{z}^{(k)} = \boldsymbol{r}^{(k)}$
Update    $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_{opt}\boldsymbol{z}^{(k)}$
```

In the third step we have to solve a linear system in P.

How should the preconditioner be chosen?

$\boxed{\text{CPU time } = \#Iter \times C}$ We need

$$\boxed{K(P^{-1}A) << K(A)}$$

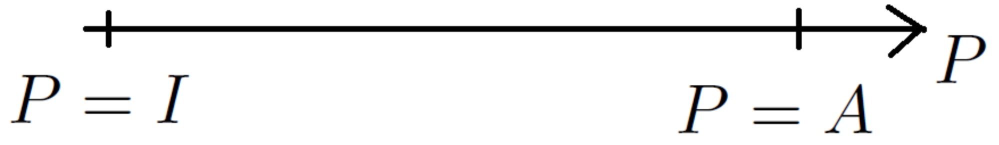
As seen, this allows us to reduce #Iter. To this end, the preconditioner P should be chosen such that $P^{-1}A$ have clustered eigenvalues

$\boxed{\text{The linear system in P must be “easily solved”}}$

This allows to have C not too large. To this aim, P should have a special structure, for example, P (block) diagonal, (block) triangular….

P “easy” to be inverted $K(P^{-1}A) \simeq K(A)$

P “difficult” to be inverted $K(P^{-1}A) << K(A)$



Example. Inexact LU factorization:

It is of interest for linear systems with sparse matrices

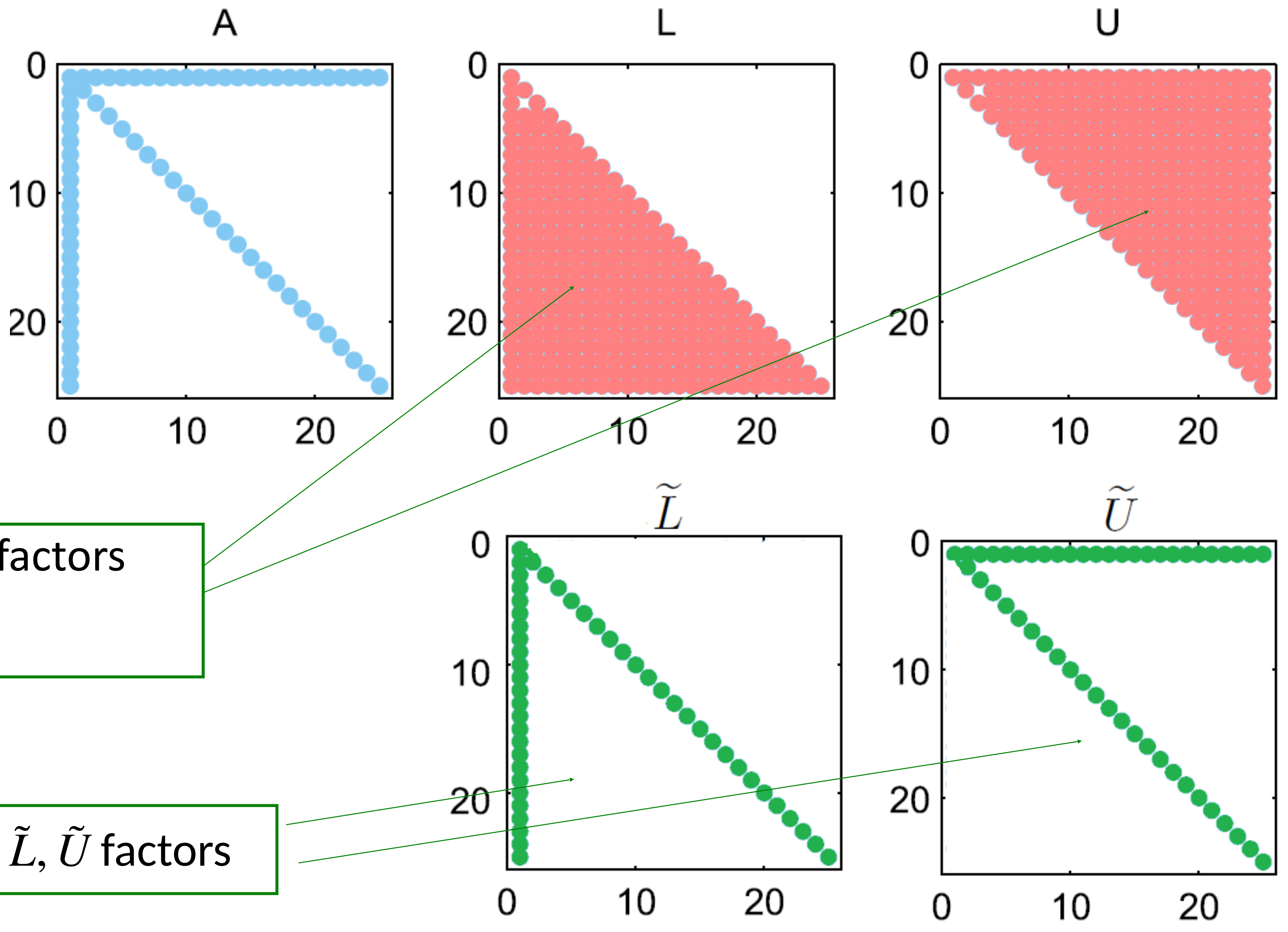
We take $P_{ILU} = \widetilde{L}\widetilde{U}$ where $\widetilde{L}, \widetilde{U}$ are the incomplete L, U factors, i.e . $A \approx \widetilde{L}\widetilde{U}$ with $\widetilde{\ell}_{ij} = 0, \widetilde{u}_{ij} = 0$ if $\widetilde{a}_{ij} = 0$

In this way, the two factors $\widetilde{L}, \widetilde{U}$ have the same sparsity pattern as A and therefore memory occupation is the same (no fill-in)

Clearly $A \neq \widetilde{L}\widetilde{U}$, therefore the idea is therefore to use incomplete LU factorisation as a preconditioner because 1) it contains information on A and 2) it is "easy" to compete the action of the preconditioner (incomplete LU factorisation + 2 triangular systems $\sim n^2$ FLOPS).

Example. Inexact LU factorization:

Sparse matrix A



L, U factors

incomplete $\tilde{L}, \tilde{U}$ factors

The gradient method

Formulation as a Problem of Minimization

In the following, let $A \in \mathbb{R}^n$, $n \geq 1$ be a symmetric and positive definite matrix, i.e.

$A = A^T$ and $\mathbf{y}^T A\mathbf{y} > 0$ for any $\mathbf{0} \neq \mathbf{y} \in \mathbb{R}^n$.

In this case, solving the linear system $A\mathbf{x} = \mathbf{b}$ is equivalent to minimizing the quadratic function $\Phi : \mathbb{R}^n \longrightarrow \mathbb{R}$

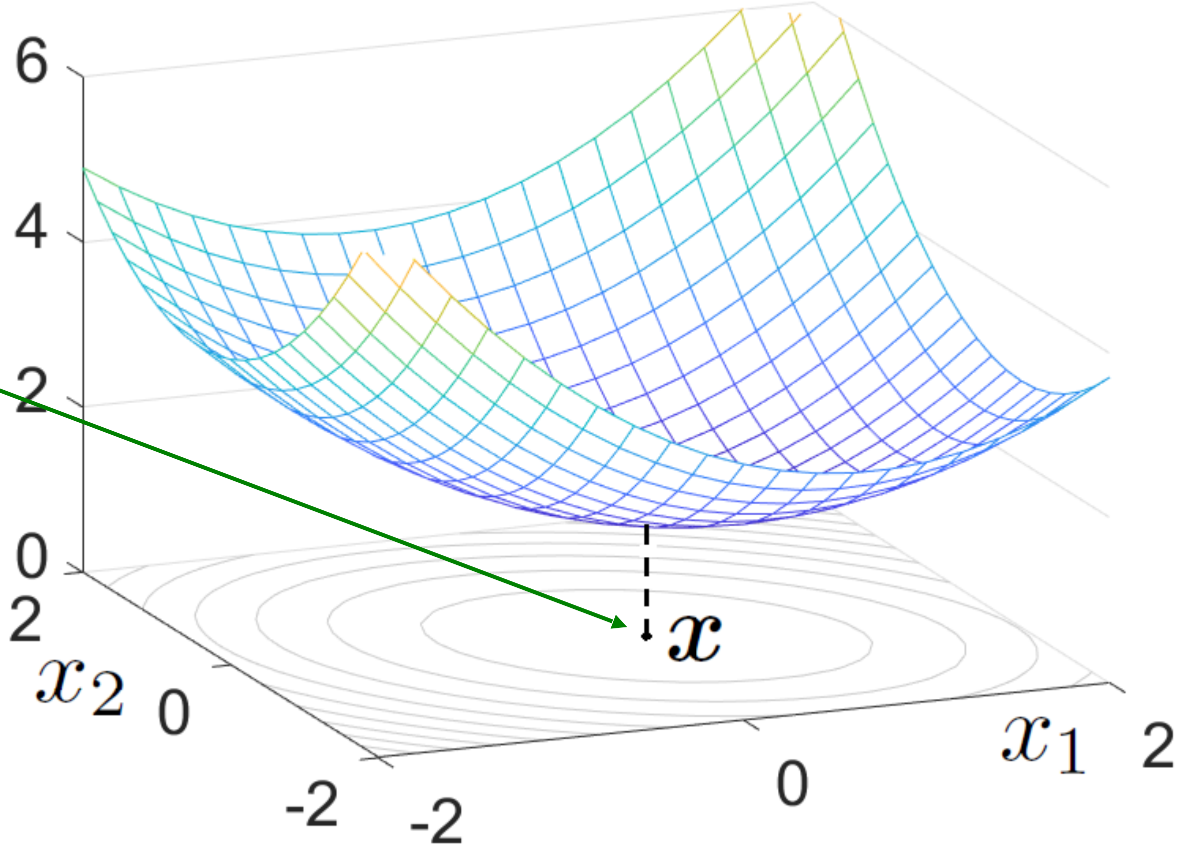
$$\Phi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T A\mathbf{y} - \mathbf{y}^T\mathbf{b}$$

As A is positive definite, the hyperplane given by $\mathbf{z} = \Phi(\mathbf{y})$ defines a paraboloid in $\mathbb{R}^{n+1}$with n-dimensional ellipsoids as isosurfaces $\Phi(\mathbf{y}) = const$. and $\Phi(\,\cdot\,)$ has a global minimum in $\mathbf{x}$ (the equivalence of the problems is obvious, see later).

Remember that A is SPD. We introduce the following energy function $\Phi : \mathbb{R}^n \longrightarrow \mathbb{R}$

$$\Phi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T A\mathbf{y} - \mathbf{y}^T\mathbf{b}$$

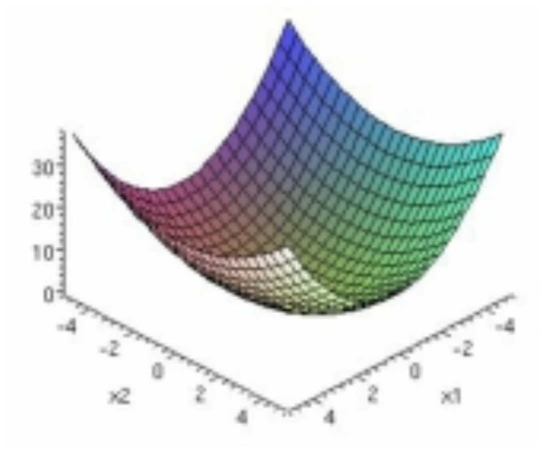
If A is SDP, the energy is a convex function that admits a unique minimum



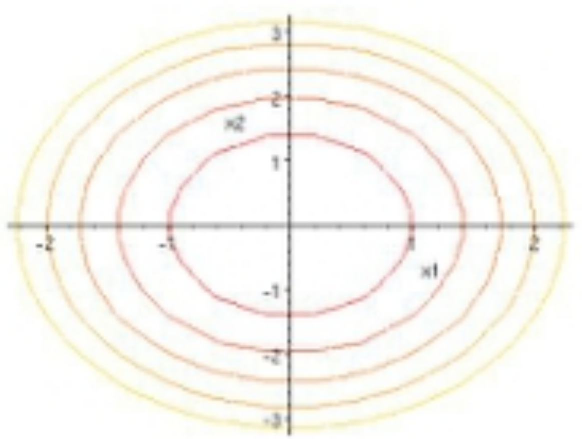
Since $\nabla\Phi(\mathbf{y}) = A\mathbf{y} - \mathbf{b}$ we have that the minimum ($\nabla\Phi(\mathbf{x}) = \mathbf{0}$) coincides with the solution of $A\mathbf{x} - \mathbf{b}$

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This is equivalent to minimizing the quadratic function $\Phi(x, y) = x^2 + \frac{1}{2}y^2$.



graph of $\Phi(x, y)$



isolines of $\Phi(x, y) = const$

An initial remark

Every perturbation $\mathbf{e} \neq \mathbf{0}$ of the solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$ increases the value of $\Phi(\mathbf{x})$

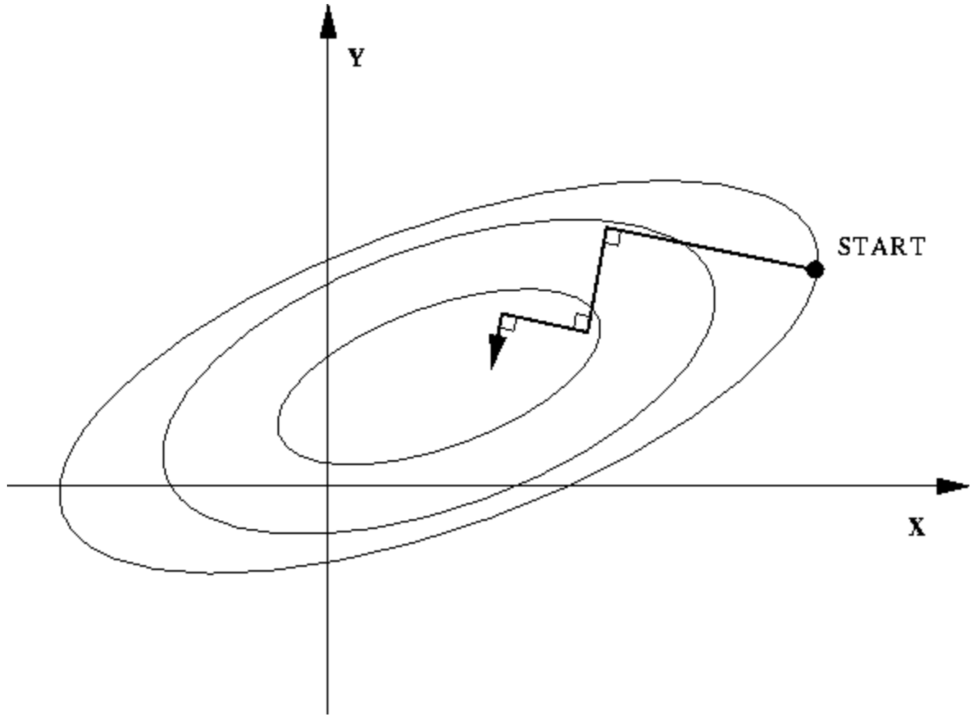
$$\Phi(\mathbf{x} + \mathbf{e}) = \ldots = \Phi(\mathbf{x}) + \frac{1}{2}\mathbf{e}^T A\mathbf{e} > \Phi(\mathbf{x})$$

A possibility to find the minimum $\mathbf{x}$ is provided by the method of steepest descent.

The method of steepest descent

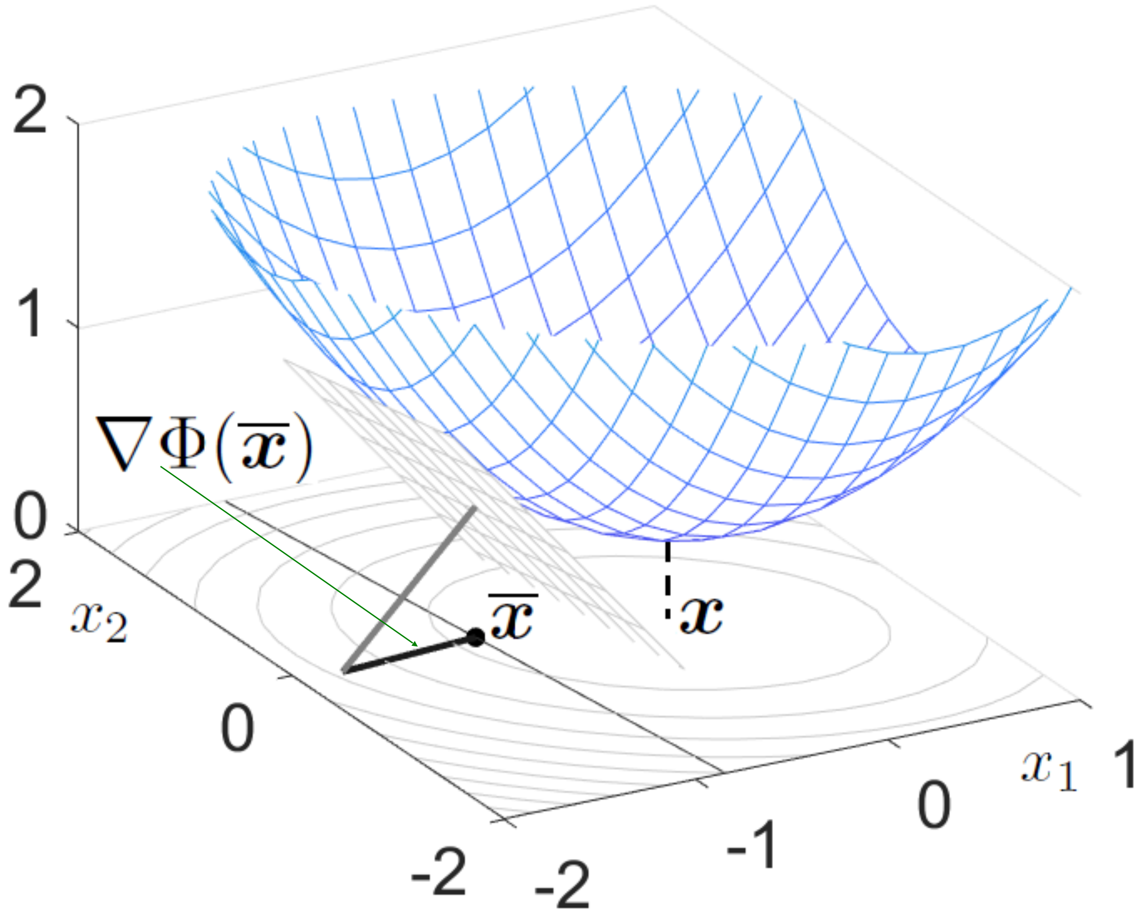
The method of steepest descent tries to find an update $\mathbf{x}^{(k+1)}$ in the direction of the steepest descent of our quadratic function, i.e., in the direction of the negative gradient

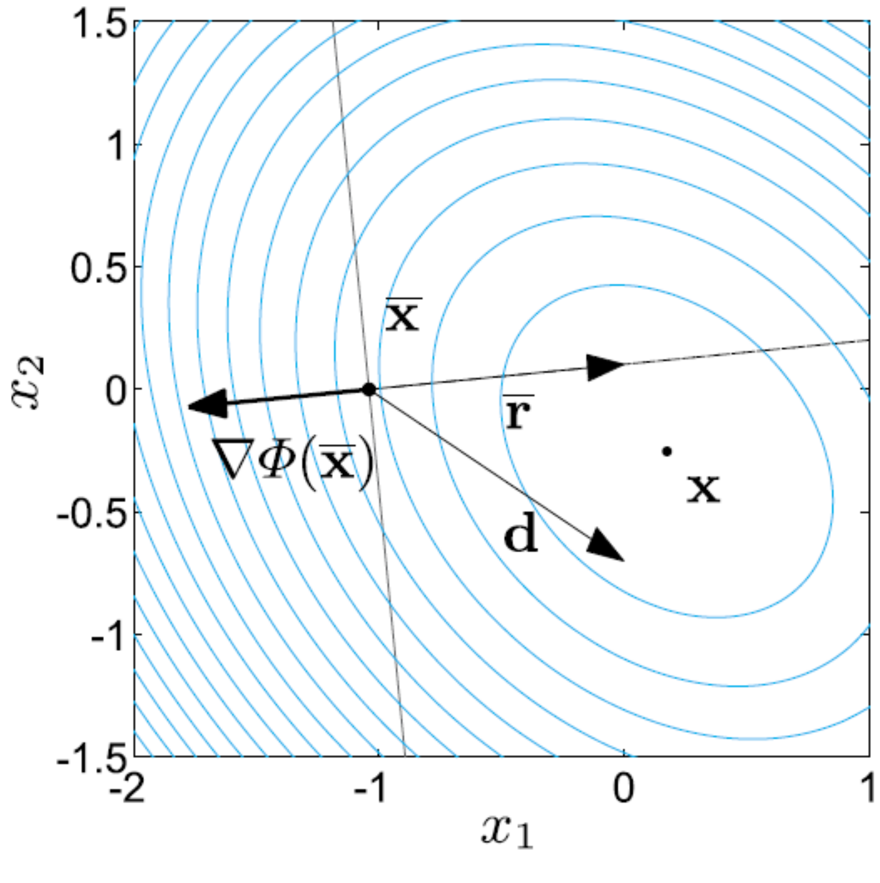
$$-\nabla\Phi(\mathbf{x}^{(k)}) = \mathbf{b} - A\mathbf{x}^{(k)} = \mathbf{r}^{(k)}$$

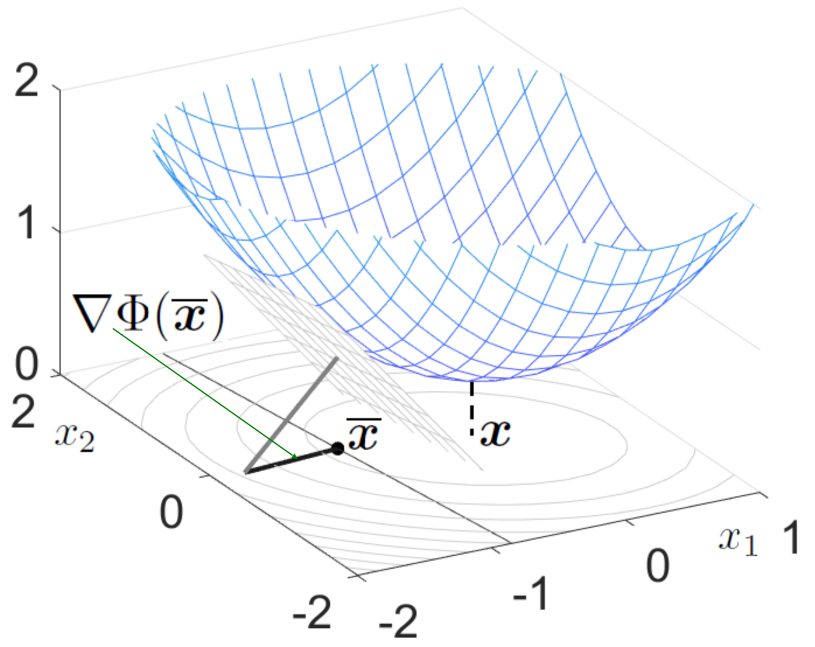


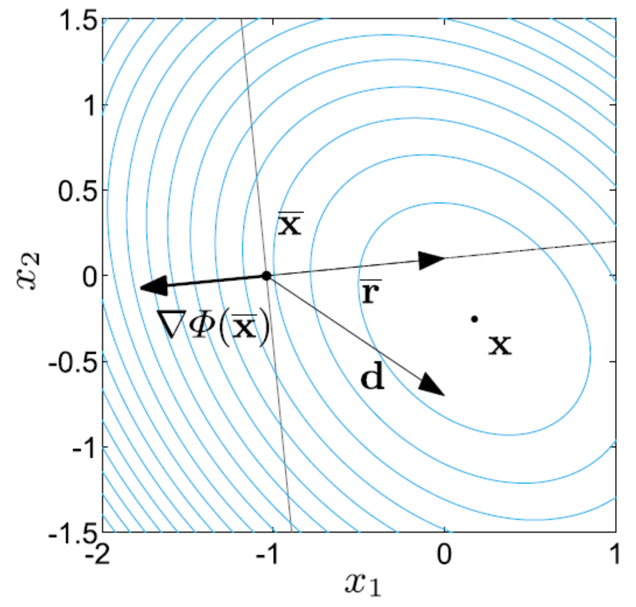
Remark: it would be better (of course!) to search in the direction of the error $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$, but -unfortunately- the error is unknown.

Indeed, for a given $\bar{\mathbf{x}}$, the vector $-\nabla\Phi(\bar{\mathbf{x}}) = \bar{\mathbf{r}}$ is the direction of steepest descent (orthogonal to the isoline with energy equal to $\Phi(\bar{\mathbf{x}}) = \mathbf{0}$)









Thus, if we want to minimize $\Phi(\,\cdot\,)$, we might think of taking a guess at $\mathbf{x}^{(k)}$, evaluating the gradient $\nabla\Phi(\mathbf{x}^{(k)})$, and taking a step in the opposite direction, i.e.,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla\Phi(\mathbf{x}^{(k)})$$

where α_k is a parameter (that I'm still free co choose).

Also other linear iterative methods (Jacobi, Gauss-Seidel, SOR, Richardson) use the residual as a direction for improving the solution:

- Richardson: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)}$
- Jacobi: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + D^{-1}\mathbf{r}^{(k)}$
- Gauss-Seidel: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + (D-E)^{-1}\mathbf{r}^{(k)}$
- JOR: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega D^{-1}\mathbf{r}^{(k)}$

Steepest descendent: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}$

? How we choose α_k, at each k?

The Gradient method

Remember that $\nabla\Phi(\mathbf{y}) = A\mathbf{y} - \mathbf{b} = -\mathbf{r}$, we obtain

$$-\nabla\Phi(\mathbf{x}^{(k)}) = \mathbf{b} - A\mathbf{x}^{(k)} = \mathbf{r}^{(k)}$$

Therefore the gradient method can be interpreted as a Richardson method with dynamic parameter α_k.

What's the advantage?

That the parameter α_k can be chosen in an optimal way at each iteration k, that the above algorithm does no longer requires knowing the eigenvalues of A (which often need to be approximated numerically)

The Gradient method

The optimal parameter α_k is chosen in order to minimize the energy by moving in the direction of the gradient. This is equivalent to asking that

$$\boxed{\frac{\mathrm{d}\Phi(\mathbf{x}^{(k+1)})}{\mathrm{d}\alpha_k} = 0}$$

We have

$$\boxed{\begin{aligned}\Phi(\boldsymbol{x}^{(k+1)}) &= \frac{1}{2}\left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{r}^{(k)}\right)^T A \left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{r}^{(k)}\right) - \left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{r}^{(k)}\right)^T \boldsymbol{b} \\ &= \frac{1}{2}\left(\boldsymbol{x}^{(k)}\right)^T A\boldsymbol{x}^{(k)} + \frac{1}{2}\alpha_k \left(\boldsymbol{x}^{(k)}\right)^T A\boldsymbol{r}^{(k)} + \frac{1}{2}\alpha_k \left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{x}^{(k)} + \frac{1}{2}\alpha_k^2 \left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{r}^{(k)} \\ &\quad - \left(\boldsymbol{x}^{(k)}\right)^T \boldsymbol{b} - \alpha_k \left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{b}\end{aligned}}$$

The Gradient method. Choosing α_k

Since $A = A^T$ we can write

$$\left(\boldsymbol{x}^{(k)}\right)^T A\boldsymbol{r}^{(k)} = \left(\boldsymbol{x}^{(k)}\right)^T A^T\boldsymbol{r}^{(k)} = \left(A\boldsymbol{x}^{(k)}\right)^T \boldsymbol{r}^{(k)} = \left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{x}^{(k)}$$

Therefore

$$0 = \frac{d\Phi(\boldsymbol{x}^{(k+1)})}{d\alpha_k} = \left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{x}^{(k)} + \alpha_k \left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{r}^{(k)} - \left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{b}$$

$$= -\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)} + \alpha_k \left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{r}^{(k)}$$

and we obtain

$$\boxed{\alpha_k = \frac{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)}}{\left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{r}^{(k)}}}$$

The Gradient method. Choosing α_k

$$\boxed{\alpha_k = \frac{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)}}{\left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{r}^{(k)}}}$$

We observe that:

$$\mathbf{r}^{(k+1)} = \mathbf{b} - A\mathbf{x}^{(k+1)} = \mathbf{b} - A(\mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}) = \mathbf{r}^{(k)} - \alpha_k A\mathbf{r}^{(k)}$$

The vector-matrix product $A\mathbf{r}^{(k)}$ in the calculation of α_k is then also employed in the update of the residual $\mathbf{r}^{(k+1)}$. Computing $\mathbf{r}^{(k+1)}$ costs the sum of two vectors.

The Gradient method. Pseudo-algorithm

```
Given $\mathbf{x}^{(0)}$, Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$

While (STOPPING CRITERIA)

    $\alpha_k = \frac{\left(\boldsymbol{r}^{(k)}\right)^T \boldsymbol{r}^{(k)}}{\left(\boldsymbol{r}^{(k)}\right)^T A\boldsymbol{r}^{(k)}}$  $\longrightarrow$  $\sim n$ FLOPS if $A$ is sparse

    $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{r}^{(k)}$  $\longrightarrow$  $\sim n$ FLOPS

    $\boldsymbol{r}^{(k+1)} = (I - \alpha_k A)\boldsymbol{r}^{(k)}$  $\longrightarrow$  $\sim n$ FLOPS

end
```

The Gradient method

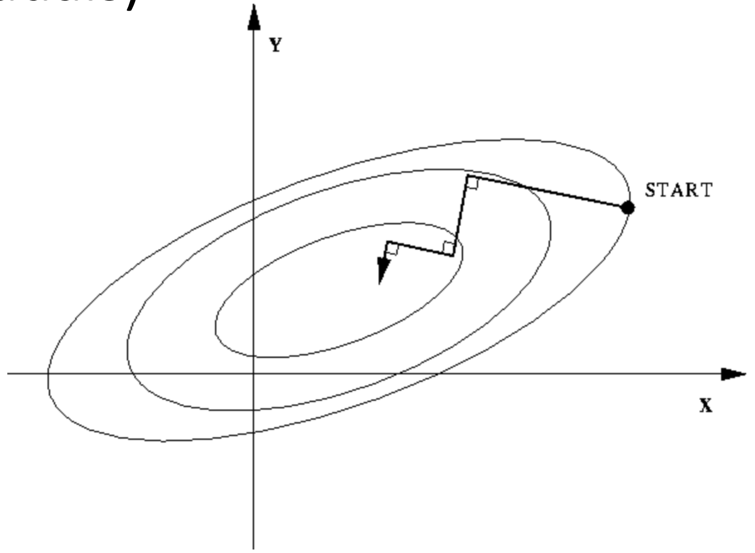
The convergence rate of the gradient method is the same as that of stationary Richardson's method with optimal parameter:

$$\|\mathbf{e}^{(k+1)}\|_A \leq \frac{K(A)-1}{K(A)+1}\|\mathbf{e}^{(k)}\|_A \Longrightarrow \boxed{\|\mathbf{e}^{(k)}\|_A \leq \left(\frac{K(A)-1}{K(A)+1}\right)^k \|\mathbf{e}^{(0)}\|_A}$$

The Conjugate Gradient method

In the gradient method, two consecutive directions (the residuals) are orthogonal by constructions. Indeed,

$$0 = (\nabla\Phi(\mathbf{x}^{(k)}), \mathbf{r}^{(k)}) = -(\mathbf{r}^{(k+1)}, \mathbf{r}^{(k)})$$



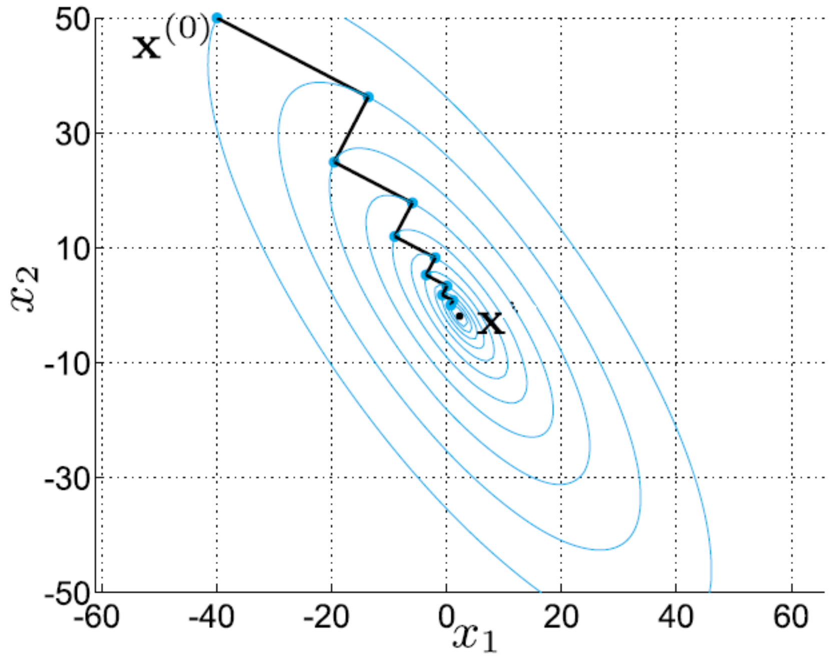
Two consecutive steepest descent directions are therefore optimal, but this is no longer true in general, i.e., the convergence behaviour of the method of steepest descent is in general poor.

To overcome this limit, we introduce a new updating direction, $\mathbf{d}^{(k+1)}$, in such a way that it is A-conjugate to all the previous directions $\mathbf{d}^{(j)}, j \leq k$ (i.e., orthogonal with respect to the scalar product induced by A)

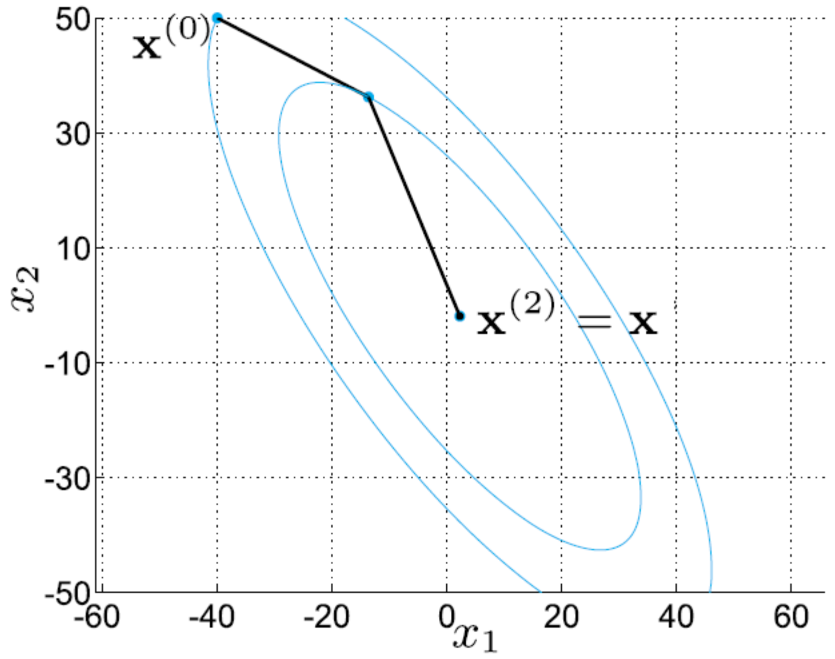
$$(\mathbf{d}^{(k+1)}, \mathbf{d}^{(j)})_A = (\mathbf{d}^{(k+1)}, A\mathbf{d}^{(j)}) = 0 \qquad \forall j \leq k$$ conjugate directions

The Conjugate Gradient method (GC):

Conjugacy of the search directions improves convergence



Gradient method



Conjugate gradient method

The Conjugate Gradient method (GC):

Theorem: In exact arithmetic the GC method converges to the exact solution in at most n iterations. At each iteration k, the error $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$ can be bounded by

$$\boxed{\|\boldsymbol{e}^{(k)}\|_A \leq \frac{2c^k}{1+c^{2k}}\|\boldsymbol{e}^{(0)}\|_A} \quad \text{with} \quad \boxed{c = \frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1}}$$

Round-of-errors can affect the performance.

The Gradient and Conjugate Gradient methods:

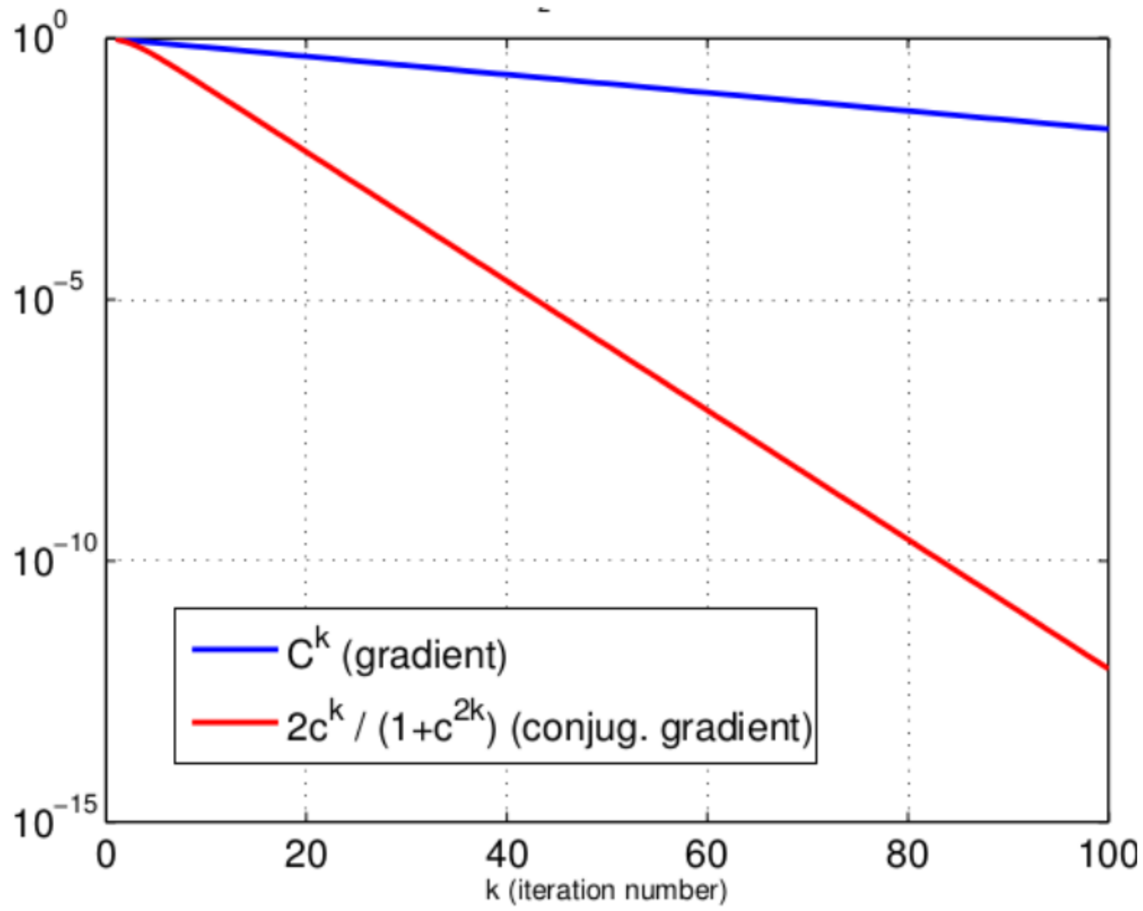
Gradient method:

$$\|\boldsymbol{e}^{(k)}\|_A \leq \left(\frac{K(A)-1}{K(A)+1}\right)^k \|\boldsymbol{e}^{(0)}\|_A$$

CG method:

$$\|\boldsymbol{e}^{(k)}\|_A \leq \frac{2c^k}{1+c^{2k}} \|\boldsymbol{e}^{(0)}\|_A$$

$$\boxed{c = \frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1}}$$



$K(A) = 50$

The CG method. Pseudo-algorithm

Given $\mathbf{x}^{(0)}$, Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$, set $\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$

Each iteration has a cost comparable with that of Richardson and Gadient methods.

```
While (STOPPING CRITERIA)
    $\alpha_k = \frac{(\mathbf{d}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{d}^{(k)})^T \mathrm{A}\mathbf{d}^{(k)}},$
    $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$   $\longrightarrow$ Update $\mathbf{x}^{(k+1)}$ along $\mathbf{d}^{(k)}$
    $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{d}^{(k)},$   $\longrightarrow$ Update $\mathbf{r}^{(k+1)}$
    $\beta_k = \frac{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{r}^{(k+1)}}{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{d}^{(k)}}$
    $\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$   $\longrightarrow$ Update $\mathbf{d}^{(k+1)}$
end
```

The preconditioned gradient and CG methods

Both methods converge slowly for linear system of equations with poorly conditioned matrices. A preconditioner with the same requirements discussed above is introduced, in order to accelerate convergence.

Let A e P be SDP. We consider the following preconditioned system

$$\widehat{\mathrm{A}}\widehat{\mathbf{x}} = \widehat{\mathbf{b}}$$

$$\underbrace{\mathrm{P}^{-1}\mathrm{A}\mathrm{P}^{-T}}_{\widehat{\mathrm{A}}}\underbrace{\mathrm{P}^{T}\mathbf{x}}_{\widehat{\mathbf{x}}} = \underbrace{\mathrm{P}^{-1}\mathbf{b}}_{\widehat{\mathbf{b}}}$$

The PCG method. Pseudo-algorithm

```
Given $\mathbf{x}^{(0)}$, Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$, set $\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$

  While (STOPPING CRITERIA)
        $\alpha_k = \frac{{\mathbf{z}^{(k)}}^T \mathbf{r}^{(k)}}{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{d}^{(k)}},$
        $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$
        $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{d}^{(k)},$
        $\mathrm{P}\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)},$    $\longrightarrow$ Compute the action of the preconditioner $P$ on $\mathbf{r}^{(k+1)}$
        $\beta_k = \frac{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{z}^{(k+1)}}{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{d}^{(k)}},$
        $\mathbf{d}^{(k+1)} = \mathbf{z}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$
end
```

The PCG method, error bounds

$$\|\boldsymbol{e}^{(k)}\|_A \leq \frac{2c^k}{1+c^{2k}}\|\boldsymbol{e}^{(0)}\|_A$$ with $$c = \frac{\sqrt{K(P^{-1}A)}-1}{\sqrt{K(P^{-1}A)}+1}$$

If the preconditioner is a “good” preconditioner then

$$\frac{\sqrt{K(P^{-1}A)}-1}{\sqrt{K(P^{-1}A)}+1} < \frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1}$$

The PCG method, error bounds

$$\|\boldsymbol{e}^{(k)}\|_A \leq \frac{2c^k}{1+c^{2k}}\|\boldsymbol{e}^{(0)}\|_A$$

with

$$c = \frac{\sqrt{K(P^{-1}A)}-1}{\sqrt{K(P^{-1}A)}+1}$$

If the preconditioner is a “good” preconditioner then

$$\frac{\sqrt{K(P^{-1}A)}-1}{\sqrt{K(P^{-1}A)}+1} < \frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1}$$

Krylov-space methods

From linear iterative methods to Krylov space methods

For linear iterative methods (with $P = I, \alpha_k = 1\ \forall k$) as those we seen before, we have

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{r}^{(k)} \qquad k \geq 1 \qquad (1)$$

The following recursive relation for the residuals holds

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - A\mathbf{r}^{(k)} \qquad k \geq 1$$

From the above identity, it follows by induction, that

$$\mathbf{r}^{(k)} = p_{k-1}(A)\mathbf{r}^{(0)} \in \text{span}\left\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, \ldots, A^{k-1}\mathbf{r}^{(0)}\right\}$$

where $p_r(z) = (1-z)^r$ is a polynomial of exact degree r.

From (1), we have

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathbf{r}^{(0)} + \ldots + \mathbf{r}^{(k-1)} = \mathbf{x}^{(0)} + p_{k-1}(A)\mathbf{r}^{(0)}$$

From linear iterative methods to Krylov space methods

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathbf{r}^{(0)} + \ldots + \mathbf{r}^{(k-1)} = \mathbf{x}^{(0)} + p_{k-1}(A)\mathbf{r}^{(0)}$$

So $\mathbf{x}^{(k)}$ lies in the affine space

$$\mathbf{x}^{(0)} + \text{span}\left\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, \ldots, A^{k-1}\mathbf{r}^{(0)}\right\}$$

obtained by shifting the subspace of $\mathbf{r}^{k-1}$.

Definition (Krylov (sub)space). Given a nonsingular $A \in \mathbb{R}^{n\times n}$ and $\mathbf{y} \in \mathbb{R}^n, \mathbf{y} \neq \mathbf{0}$, the kth Krylov (sub)space $\mathscr{K}_k(A, \mathbf{y})$ generated by A from $\mathbf{y}$ is

$$\mathscr{K}_k(A, \mathbf{y}) := \text{span}(\mathbf{y}, A\mathbf{y}, \ldots, A^{k-1}\mathbf{y})$$

Clearly, it holds

$\mathscr{K}_1(A, \mathbf{y}) \subseteq \mathscr{K}_2(A, \mathbf{y}) \subseteq \ldots$

Krylov spaces

It seems it seems clever to choose the kth approximate solution $\mathbf{x}^{(k)}$

$\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathscr{K}_k(A, \mathbf{r}^{(0)})$

But can we expect to find the exact solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$ in one of those affine space?

Lemma. Let $\mathbf{x}$ be the solution of $A\mathbf{x} = \mathbf{b}$ and let $\mathbf{x}^{(0)}$ be any initial approximation of it and $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$ the corresponding residual. Moreover, let $\nu = \nu(\mathbf{r}^{(0)}, A)$ be the so called grade of $\mathbf{r}^{(0)}$ with respect to A. Then,

$$\mathbf{x} \in \mathbf{x}^{(0)} + \mathscr{K}_\nu(A, \mathbf{r}^{(0)})$$

Grade of $\mathbf{y}$ with respect to A

Lemma. There is a positive integer $\nu = \nu(\mathbf{y}, A)$, called grade of $\mathbf{y}$ with respect to A, such that

$$\dim(\mathscr{K}_s(A, y)) = \begin{cases} s \text{ if } s \leq \nu \\ \nu \text{ if } s \geq \nu \end{cases}$$

$\mathscr{K}_\nu(A, \mathbf{y})$ is is the smallest A–invariant subspace that contains $\mathbf{y}$.

Lemma. The nonnegative integer $\nu = \nu(\mathbf{y}, A)$ of $\mathbf{y}$ with respect to A satisfies.

$$\nu(\mathbf{y}, A) = \min \left\{ s \,|\, A^{-1}\mathbf{y} \in \mathscr{K}_s(A, y) \right\}$$

Krylov space methods

The idea behind Krylov space solvers is to generate a sequence of approximate solutions $\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathscr{K}_k(A, \mathbf{r}^{(0)})$ of $A\mathbf{x} = \mathbf{b}$ so that the corresponding residuals $\mathbf{r}^{(k)} \in \mathscr{K}_{k+1}(A, \mathbf{r}^{(0)})$ "converge" to the zero vector $\mathbf{0}$.

Here, "converge" may also mean that after a finite number of steps, $\mathbf{r}^{(k)} = \mathbf{0}$, so that $\mathbf{x}^k = \mathbf{x}$ and the process stops. This is in particular true (in exact arithmetic) if a method ensures that the residuals are linearly independent: then $\mathbf{r}^{(\nu)} = \mathbf{0}$. In this case we say that the method has the finite termination property

Krylov space methods

Definition. A (standard) Krylov space method for solving a linear system $A\mathbf{x} = \mathbf{b}$ or, briefly, a Krylov space solver is an iterative method starting from some initial approximation $\mathbf{x}^{(0)}$ and the corresponding residual $\mathbf{r}^{(0)}$ and generating for all, or at least most k, until it possibly finds the exact solution, iterates $\mathbf{x}^{(\mathbf{k})}$ such that

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + p_{k-1}(A)\mathbf{r}^{(0)}$$

with a polynomial $p_{k-1}(A)$ of exact degree $k-1$. For some k, $\mathbf{x}^{(k)}$ may not exist or $p_{k-1}(A)$ may have lower degree.

Krylov space methods

When applied to large real-world problems Krylov space solvers often converge very slowly — if at all. In practice, Krylov space solvers are therefore nearly always applied with preconditioning:

$$A\mathbf{x} = \mathbf{b} \qquad \Longleftrightarrow \qquad P^{-1}APP^{-1}\mathbf{x} = P^{-1}\mathbf{b} \qquad \Longleftrightarrow \qquad \hat{A}\hat{\mathbf{z}} = \hat{\mathbf{b}}, \quad \hat{P}\hat{\mathbf{z}} = \mathbf{x}$$

Applying a preconditioned Krylov space solver just means to apply the method to $\hat{A}\hat{\mathbf{z}} = \hat{\mathbf{b}}$.

The CG method is a Krylov space solver

CG is the archetype of a Krylov space solver that is an orthogonal projection method. By definition, such a method chooses the step length α_k so that $\mathbf{x}^{(k+1)}$ is locally optimal on the search line

$$\alpha_k = \frac{(\mathbf{d}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{d}^{(k)})^T \mathrm{A}\mathbf{d}^{(k)}},$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{d}^{(k)},$$

$$\beta_k = \frac{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{r}^{(k+1)}}{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{d}^{(k)}}$$

$$\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$$

But does it also yield the best

$$\mathbf{x}^{(k+1)} \in \mathbf{x}^{(0)} + \text{span}\left\{\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k\right\}?$$

The next result shows that

$$\text{span}\left\{\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k\right\} = \mathscr{K}_{k+1}(A, \mathbf{r}^{(0)})$$

The CG method is a Krylov space solver

Theorem. The CG method yields approximate solutions $\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathscr{K}_k(A, \mathbf{r}^{(0)})$, that are optimal in the sense that they minimize the energy norm (A-norm) of the error (i.e., the A^{-1}-norm of the residual) for $\mathbf{x}^{(k)}$ from this affine space.

$$\alpha_k = \frac{(\mathbf{d}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{d}^{(k)})^T \mathrm{A}\mathbf{d}^{(k)}},$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathrm{A}\mathbf{d}^{(k)},$$

$$\beta_k = \frac{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{r}^{(k+1)}}{(\mathrm{A}\mathbf{d}^{(k)})^T \mathbf{d}^{(k)}}$$

$$\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$$

Solving nonsymmetric linear systems iteratively with Krylov space solvers.

Krylov space solvers for nonsymmetric systems

Solving nonsymmetric linear systems iteratively with Krylov space solvers is considerably more difficult and costly than symmetric systems. There are two different ways to generalize CG:

- Maintain the orthogonality of the projection and the related minimality of the error by constructing either orthogonal residuals $\mathbf{x}^{(k)}$ (generalized CG - GCG). Then, the recursions involve all previously constructed residuals or search directions and all previously constructed iterates.
- Maintain short recurrence formulas for residuals, direction vectors and iterates (biconjugate gradient (BiCG) method, Lanczos-type product methods (LTPM)). The resulting methods are at best oblique projection methods. There is no minimality property of error or residuals vectors.

The biconjugate gradient (BiCG) method

While CG (for spd A) has mutually orthogonal residuals $\mathbf{r}^{(k)}$ with

$$\mathbf{r}^{(k)} = p_k(A)\mathbf{r}^{(0)} \in \text{span}\left\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, \dots, A^k\mathbf{r}^{(0)}\right\} = \mathscr{K}_{k+1}(A, \mathbf{r}^{(0)})$$

BiCG constructs in the same spaces residuals that are orthogonal to a dual Krylov space spanned by "*shadow residuals*"

$$\tilde{\mathbf{r}}^{(k)} = p_k(A^T)\tilde{\mathbf{r}}^{(0)} \in \text{span}\left\{\tilde{\mathbf{r}}^{(0)}, A^T\tilde{\mathbf{r}}^{(0)}, \dots, (A^T)^k\tilde{\mathbf{r}}^{(0)}\right\} = \mathscr{K}_{k+1}(A^T, \tilde{\mathbf{r}}^{(0)}) \equiv \tilde{\mathscr{K}}_{k+1}$$

The initial shadow residual $\tilde{\mathbf{r}}^{(0)}$ can be chosen freely. So, BiCG requires two matrix-vector multiplications to extend $\mathscr{K}_k$ and $\tilde{\mathscr{K}}_k$: one multiplication by A and one by A^T.

The BiCG method - algorithm

$$A\mathbf{x} = \mathbf{b}$$
$$\hat{\mathbf{x}}^T A^T = \mathbf{b}^T$$

```
Chose $\mathbf{x}^{(0)}$, $\widehat{\mathbf{x}}^{(0)}$ , Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$, Compute $\hat{\mathbf{r}}^{(0)} = \mathbf{b} - \hat{\mathbf{x}}^{(0)}A^T$
$\mathbf{d}_0 = \mathbf{r}^{(0)}, \hat{\mathbf{d}}_0 = \hat{\mathbf{r}}^{(0)}$
WHILE(STOPPING CRITERIA)
    $\alpha_k = \dfrac{\hat{\mathbf{r}}^{(k)}\mathbf{r}^{(k)}}{\hat{\mathbf{d}}_k A\mathbf{d}k}$
    $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}_k$
    $\hat{\mathbf{x}}^{(k+1)} = \hat{\mathbf{x}}^{(k)} + \alpha_k \hat{\mathbf{d}}_k$
    $\mathbf{r}^{(k+1)} = \mathbf{b} - A\mathbf{x}^{(k+1)}$
    $\hat{\mathbf{r}}^{(k+1)} = \hat{\mathbf{b}} - \hat{\mathbf{x}}^{(k+1)}A^T$
    $\beta_k = \dfrac{\hat{\mathbf{r}}^{(k+)}\mathbf{r}^{(k+1)}}{\hat{\mathbf{r}}^{(k)}\mathbf{r}^{(k)}}$
    $\mathbf{d}_{k+1} = \mathbf{r}^{(k+1)} + \beta_k \mathbf{d}_k$
    $\hat{\mathbf{d}}_{k+1} = \hat{\mathbf{r}}^{(k+1)} + \beta_k \hat{\mathbf{d}}_k$
END
```

The BiCGSTAB method

The biconjugate gradient stabilized method (BiCGSTAB) is a variant of the biconjugate gradient method (BiCG) and has faster and smoother convergence than the original BiCG.

It is unnecessary to explicitly keep track of the residuals and search directions of BiCG. In other words, the BiCG iterations can be performed implicitly.

Unlike the original BiCG method, it does not require multiplication by A^T

The GMRES method

The Generalized Minimum Residual Method (GMRES) is a projection method.

The method approximates the solution by the vector in a Krylov subspace with minimal residual
The Arnoldi iteration is used to find this vector.

Recall that the k-th Krilov space $\mathscr{K}_k = \mathscr{K}_k(A, \mathbf{r}^{(0)})$ is given by

$$\mathscr{K}_k(A, \mathbf{r}^{(0)}) = \text{span}\left\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, \ldots, A^{k-1}\mathbf{r}^{(0)}\right\}.$$

Idea: GMRES approximates the exact solution of $A\mathbf{x} = \mathbf{b}$ by the vector

$$\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathscr{K}_k(A, \mathbf{r}^{(0)})$$

that minimizes the Euclidean norm of the residual $\mathbf{r}^{(k)} = b - A\mathbf{x}^{(k)}$.

The GMRES method

Remarks:

The vectors $\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, \ldots, A^{k-1}\mathbf{r}^{(0)}$ might be close to being linearly dependent, so instead of this basis, a suitable method is used to find orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$ which form a basis for $\mathscr{K}_n(A, \mathbf{r}^{(0)})$.

The GMRES method (pseudo-algorithm)

```
Chose $\mathbf{x}^{(0)}$ , Compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$,

Set $\mathbf{q}_1 = \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|_2$

WHILE(STOPPING CRITERIA)
    Compute $\mathbf{q}_k$ (with a suitable method)
    Form $\mathbf{Q}_k$ as the $n \times k$ matrix formed by $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$
    Find $\mathbf{y}^{(k)}$ which minimise $\|\mathbf{r}^{(k)}\|_2$
    Compute $\mathbf{x}^{(k+1)} = \mathbf{x}^{(0)} + Q_k\mathbf{y}^{(k)}$
end
```

GMRES some remarks

Remarks

- At every iteration k, a matrix-vector product must be computed, which costs about n^2 FLOPS for dense matrices. If A is sparse, this cost is $O(n)$ FLOPS.
- In addition to the matrix-vector product, $O(kn)$ FLOPS operations must be computed at the k-th iteration.
- The k-th iterate minimises the residual in the Krylov subspace $\mathscr{K}_k(A, \mathbf{r}^{(0)})$. In exact arithmetic, since every subspace is contained in the next subspace, the residual does not increase. Therefore, after $n = \text{size}(A)$ iterations, the Krylov space $\mathscr{K}_n(A, \mathbf{r}^{(0)})$ is the whole of $\mathbb{R}^n$ (hence the GMRES method has finite termination property). This, unfortunately, does not happen in practice.

GMRES convergence, special cases

- If $A_S = (A + A^T)/2$ is SPD, then

$$\|\mathbf{r}^{(k)}\|_2 \leq \left[1 - \frac{\lambda^2_{\min}(A_S)}{\lambda_{\max}(A^TA)}\right]^{k/2} \|\mathbf{r}^{(0)}\|_2$$

- If A is SPD, then

$$\|\mathbf{r}^{(k)}\|_2 \leq \left[\frac{[K_2(A)]^2 - 1}{[K_2(A)]^2}\right]^{k/2} \|\mathbf{r}^{(0)}\|_2$$