High Performance Linear Algebra

Lecture 2.1: Introduction to iterative methods

Ph.D. program in High Performance Scientific Computing

Salvatore Filippone **Pasqua D'Ambra** **Fabio Durastante**

March 10 2026 — 13.00:15.00

Table of Contents

Direct Solvers?

We have seen various *direct* methods for the solution of linear systems, linear least squares, eigenvalues.

Direct methods?

The existence of a direct method is a nontrivial proposition:

- A method that *guarantees* an answer in finite time;
- A method that *would* give the correct answer, if only we had exact arithmetic in our computers.

Examples: Linear Systems, Least Squares, Discrete Fourier Transforms.

Direct Solvers?

We have seen various *direct* methods for the solution of linear systems, linear least squares, eigenvalues.

Direct methods?

The existence of a direct method is a nontrivial proposition:

- A method that *guarantees* an answer in finite time;
- A method that *would* give the correct answer, if only we had exact arithmetic in our computers.

Examples: Linear Systems, Least Squares, Discrete Fourier Transforms.

Many problems, probably the vast majority, do NOT admit a direct method even when a solution is guaranteed to exist
(cfr. fundamental theorem of algebra vs Abel-Galois theorem).
Note: a direct method also qualifies as an *algorithm* in the theoretical computer science sense.

Algorithm

Derived from the name “Abū Jaʿfar Muḥammad ibn Mūsā al-Khwārizmī” a persian mathematician active between 780 and 850 CE, of whom we know practically nothing. The title of his book “al-Kitāb al-mukhtaṣar fī ḥisāb al-jabr wa al-muqābala” is the origin of the term *algebra*; we can find there the first systematic exposition of the system that to this day goes under the name of “arabic numerals” (but in reality originated in India and later came *through* the arabic world), a system made popular in Europe by Leonardo Pisano aka Fibonacci with his “Liber abbaci”. Khwārizmī is a region in central Asia, currently split between Turkmenistan, Kazakhstan and Uzbekistan, seat of a muslim empire comprising also Iran and Afghanistan between 1000 and 1200 CE, and including the legendary Samarkand, which later fell to the Mongols of Gengis Khan in 1231.

Definition of Algorithm

What is an algorithm?

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

1. Input;

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

1. Input;
2. Output;

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

1. Input;
2. Output;
3. Definitess (no ambiguity);

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

1. Input;
2. Output;
3. Definitess (no ambiguity);
4. Effectiveness (every step is executable in a finite time, even on a machine);

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

1. Input;
2. Output;
3. Definitess (no ambiguity);
4. Effectiveness (every step is executable in a finite time, even on a machine);
5. Finiteness (always executes a finite number of steps).

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

1. Input;
2. Output;
3. Definitess (no ambiguity);
4. Effectiveness (every step is executable in a finite time, even on a machine);
5. Finiteness (always executes a finite number of steps).

Definition of Algorithm

What is an algorithm?

A procedure to solve a problem, i.e. a method that gives an answer starting from the problem data, satisfying the following constraints:

1. Input;
2. Output;
3. Definitess (no ambiguity);
4. Effectiveness (every step is executable in a finite time, even on a machine);
5. Finiteness (always executes a finite number of steps).

By definition an algorithm *always* answers in a *finite* time with *the* solution.

Definition of Algorithm

The fact that an algorithm answers in a finite time should be no comfort to you.
In 1950 Claude Shannon wrote an article designing a program for the game of chess; such a program could simply consider all possible move sequences and board configurations (thanks to some specific rules there are a finite number of them).

Definition of Algorithm

The fact that an algorithm answers in a finite time should be no comfort to you.
In 1950 Claude Shannon wrote an article designing a program for the game of chess; such a program could simply consider all possible move sequences and board configurations (thanks to some specific rules there are a finite number of them).

1. Average number of legal half-moves per board configuration: 30;
2. Average number of combinations for a full (White-Black) move: $\approx 10^3$;
3. Average game length: 40 moves;

Definition of Algorithm

The fact that an algorithm answers in a finite time should be no comfort to you.
In 1950 Claude Shannon wrote an article designing a program for the game of chess; such a program could simply consider all possible move sequences and board configurations (thanks to some specific rules there are a finite number of them).

1. Average number of legal half-moves per board configuration: 30;
2. Average number of combinations for a full (White-Black) move: $\approx 10^3$;
3. Average game length: 40 moves;

Hence we have to enumerate

$$10^{120}$$

possible board configurations.

Definition of Algorithm

The fact that an algorithm answers in a finite time should be no comfort to you.
In 1950 Claude Shannon wrote an article designing a program for the game of chess; such a program could simply consider all possible move sequences and board configurations (thanks to some specific rules there are a finite number of them).

1. Average number of legal half-moves per board configuration: 30;
2. Average number of combinations for a full (White-Black) move: $\approx 10^3$;
3. Average game length: 40 moves;

Hence we have to enumerate

$$10^{120}$$

possible board configurations.
Put into perspective:

- Estimated number of atoms in the universe: 10^{80}
- Diameter of the universe measured in electron diameters: 10^{39}

Direct Solvers (maybe)

Many problems do not admit a direct solver:

- General eigenvalue computations;
- Solution of ODEs (a closed form solution may or –more likely– may not be available);
- Solution of PDEs (a closed form solution is exceedingly rare);

and even if they do, evaluating a closed form solution may not be the most accurate procedure.

Direct Solvers (maybe)

Many problems do not admit a direct solver:

- General eigenvalue computations;
- Solution of ODEs (a closed form solution may or –more likely– may not be available);
- Solution of PDEs (a closed form solution is exceedingly rare);

and even if they do, evaluating a closed form solution may not be the most accurate procedure. In all of these cases we have no choice but to resort to a

computational scheme

A method that looks like an algorithm, but fails to be one because:

It is not guaranteed to terminate in a finite time

since it basically generates a (hopefully convergent) series.

Of course, a computational scheme has to be complemented with a *stopping criterion* to ensure we are not running out of time: hence *there cannot be any guarantee* that we are getting *the* solution.

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.
Ingredients:

- Initial guess x_0;

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.
Ingredients:

- Initial guess x_0;
- Stopping criterion;

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.
Ingredients:

- Initial guess x_0;
- Stopping criterion;
- Advancement rule $x_{i+1} = f(x_0, x_1, \ldots, x_i)$.

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.
Ingredients:

- Initial guess x_0;
- Stopping criterion;
- Advancement rule $x_{i+1} = f(x_0, x_1, \ldots, x_i)$.

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.
Ingredients:

- Initial guess x_0;
- Stopping criterion;
- Advancement rule $x_{i+1} = f(x_0, x_1, \ldots, x_i)$.

Why?

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.
Ingredients:

- Initial guess x_0;
- Stopping criterion;
- Advancement rule $x_{i+1} = f(x_0, x_1, \ldots, x_i)$.

Why?
Final goal: "solve" (i.e. find a good enough approximation for) an $N \times N$ system in less than $O(N^3)$ operations (even better, in less than $O(N^2)$ operations).

Iterative Solvers for Linear Systems

Nonetheless, let's look at what an iterative linear system solver would look like.
We want to solve

$$Ax = b$$

by constructing a series $x_1, x_2, \ldots, x_k$ that converges to the solution $\hat{x}$.
Ingredients:

- Initial guess x_0;
- Stopping criterion;
- Advancement rule $x_{i+1} = f(x_0, x_1, \ldots, x_i)$.

Why?
Final goal: "solve" (i.e. find a good enough approximation for) an $N \times N$ system in less than $O(N^3)$ operations (even better, in less than $O(N^2)$ operations). As a first example, if sequence stops at $j \ll N$ and each iteration costs $O(N^2)$ then we have a total cost

$$O(jN^2) \ll O(N^3)$$

Initial guess

Different choices of x_0 determine different outcomes of the iteration.

- If you *do not* have any specific knowledge, then the most common choice is to start from either the null vector or a random vector;
- If you *do* have knowledge of the problem the system comes from, then the initial guess may be suggested by the problem structure.

As an example, if you are solving a linear system to generate an advancement step of a time-dependent PDE, then the solution at the previous time step is an obvious candidate as an initial guess.
The choice of an initial guess is normally considered *outside* the scope of the iterative solution method itself.

Stopping criteria for iterative solvers

Criteria usually based on the residual

$$r = b - Ax,$$

Most common criteria:

1. $$\|r\| \leq \epsilon;$$
2. $$\|r\| \leq \epsilon\|b\|;$$
3. $$\|r\| \leq \epsilon(\|A\|\|x\| + \|b\|)$$

The second is probably the most commonly used, but it can be misleading for very ill-conditioned systems.
Again, if you have detailed knowledge of the problem the linear system comes from, you may devise a specific stopping criterion.

It remains to address the *advancement rule*

$$x_{i+1} = f(x_0, x_1, \ldots, x_i),$$

and we will take a close look at multiple possibilities in this and in the next lectures.

Table of Contents

Stationary iterations

First idea: split the matrix

$$A = M + (A - M)$$

Hence

$$M\hat{x} = (M - A)\hat{x} + b \Rightarrow \hat{x} = M^{-1}(M - A)\hat{x} + M^{-1}b$$

which we can use to define the iteration:

$$\begin{aligned} Mx_{k+1} &= (M - A)x_k + b \Rightarrow \\ x_{k+1} &= M^{-1}(M - A)x_k + M^{-1}b \end{aligned}$$

This originates from the idea of “relaxation”, altering one unknown at a time to satisfy its equation.

Defining:

$$G = M^{-1}(M - A), \qquad f = M^{-1}b$$

we have

$$x_{k+1} = Gx_k + f, \qquad \hat{x} = G\hat{x} + f.$$

If we now let

$$e_k = x_k - \hat{x},$$

and subtract term by term, we have

$$e_{k+1} = Ge_k = G^{k+1}e_0$$

Stationary iterations

We hope to achieve

$$\lim_{k\to\infty} \|e_k\| = 0$$

But we have

$$\|e_k\| = \|G^k e_0\| \leq \|G^k\| \|e_0\| \leq \|G\|^k \|e_0\|$$

Therefore: convergence condition:

$$\rho(G) < 1$$

- Sufficient condition (in theory);
- There may exist x_0 such that it's not necessary (theoretically);
- But in practice it's always necessary;
- And it may not be sufficient!

In fact, *non-normal* matrices may exhibit *large humps* in the convergence history.

Stationary iterations

Let's assume G normal, so we have decomposition

$$G = \sum_i \lambda_i v_i v_i^T$$

Therefore

$$G^k x_o = \sum_i \lambda_i^k (v_i^T x_o) v_i$$

If we have $|\lambda_1| > 1$, $|\lambda_i| < 1$ for $i \neq 1$, and $(v_1^T x_0) = 0$, in theory we have convergence.

Stationary iterations

Let's assume G normal, so we have decomposition

$$G = \sum_i \lambda_i v_i v_i^T$$

Therefore

$$G^k x_o = \sum_i \lambda_i^k (v_i^T x_o) v_i$$

If we have $|\lambda_1| > 1$, $|\lambda_i| < 1$ for $i \neq 1$, and $(v_1^T x_0) = 0$, in theory we have convergence. However in finite-precision arithmetic

$$\hat{y} = \widehat{Gx_0} = G(x_0 + \delta x_0)$$

where δx_0 is essentially a random vector which will, in practically all cases, reintroduce a component in the direction of v_1; hence divergence will occur.

A general convergence result

We have already seen that if

$$\rho(G) < 1$$

then the iteration

$$x_{k+1} = Gx_k + f = Gx_k + M^{-1}b$$

converges. Let's now prove the reverse implication: if we examine

$$(x_{k+1} - x_k) = G(x_k - x_{k-1}) = \cdots = G^k(f - (I - G)x_0)$$

we can easily see that if the iteration converges for *any* x_0 and b, then $G^k v$ converges to zero for any v, hence we must have $\rho(G) < 1$.

An example

Consider

Richardson's iteration

$$x_{k+1} = x_k + \alpha(b - Ax_k),$$

where $\alpha \geq 0$ is a scalar.

Assume all of A's eigenvalues are real and $\lambda_{\min} \leq \lambda_i \leq \lambda_{\max}$; then the eigenvalues of $G = (I - \alpha A)$ satisfy

$$1 - \alpha\lambda_{\max} \leq \mu_i \leq 1 - \alpha\lambda_{\min},$$

and moreover $\lambda_{\min} > 0$. We then derive

$$1 - \alpha\lambda_{\min} < 1, \qquad 1 - \alpha\lambda_{\max} > -1.$$

An example

Hence

$$0 < \alpha < \frac{2}{\lambda_{\max}},$$

and the optimal value requires that

$$-1 + \alpha\lambda_{\max} = 1 - \alpha\lambda_{\min},$$

which gives

$$\alpha_{opt} = \frac{2}{\lambda_{\min} + \lambda_{\max}},$$

implying that

$$\rho_{opt} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}.$$

Thus:

- Convergence can be extremely slow;
- We need eigenvalue estimates to define α.

Regular splittings

Let A, M and N satisfy

$$A = M - N.$$

This is a *regular splitting* if:

1. M is nonsingular;
2. $M^{-1}N$ is nonnegative.

Using such a splitting to define a stationary iteration leads us to

$$x_{k+1} = M^{-1}Nx_k + M^{-1}b,$$

or $G = M^{-1}N$.

Theorem (Regular splitting)

Let M, N be a regular splitting of A; then $\rho(M^{-1}N) < 1$ if and only if A is nonsingular and A^{-1} is nonnegative.

See [1] for a proof.

Theorem

$\rho(A) = \lim_{k\to\infty} \|A^k\|^{1/k}$

Thus $\rho(A)$ determines the asymptotics; however, consider:

$$J = \lambda \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} \Rightarrow J^k = \lambda^k \begin{pmatrix} 1 & k\alpha \\ 0 & 1 \end{pmatrix}$$

from which we get

$$\frac{\|J^{k+1}\|_\infty}{\|J^k\|_\infty} = \lambda \frac{1 + (k+1)\alpha}{1 + k\alpha}$$

therefore for a non normal matrix

$$\|J^{k+1}\|_\infty < \|J^k\|_\infty \quad \text{only when} \quad k > \frac{\lambda(1+\alpha) - 1}{(1-\lambda)\alpha}.$$

Thus there may be a transient peak, even a very significant one capable of practically destroying convergence.

Let us define:

$$\Lambda_\epsilon(A) = \left\{ z \in \mathbb{C} : \|(zI - A)^{-1}\| \geq \epsilon^{-1} \right\}$$

This is equivalent to

$$\Lambda_\epsilon(A) = \{ z \in \mathbb{C} : \exists \Delta A : z \in \Lambda(A + \Delta A) \qquad \|\Delta A\| \leq \epsilon \}$$

Considering

$$\frac{du}{dt} = Au + e^{zt} f \Rightarrow u(t) = e^{zt}(zI - A)^{-1} f,$$

If $z \in \Lambda_\epsilon(A)$, there exists f such that $\|u(t)\| / \|e^{zt} f\|$ is arbitrarily close to ϵ^{-1}
See [2]

An analysis of solve time

Total iteration time:

$$T_{\text{tot}} = T_{\text{setup}} + N_{\text{it}} \times T_{\text{it}}$$

T_{setup} is the time it takes to initialize the iteration (before you make any steps towards the solution);

N_{it} is the number of iterations to convergence (related to the interaction between the method and the properties of A);

T_{it} is the (average) time to execute a single iteration (related to the method and the efficiency of the code implementing it)

Let's look more closely at the various terms in the previous slides for stationary iterations:

$$G = M^{-1}(M - A)$$

and we want $\rho(G)$ small.
Therefore we need:

- M easy (fast) to determine; T_{setup}
- M^{-1} easily applied; T_{it}
- $M \approx A \Rightarrow G \approx 0$; N_{it}

These requisites cannot be satisfied simultaneously!
So, we need to find good tradeoffs.

Jacobi method

$$M = diag(A)$$

```
m  = diag(diag(a));
mi = m;
mi(mi ~= 0) = 1.0 ./ mi(mi ~= 0) ;
n= (m-a);
bn2=norm(b);
for i=1:itmax
   x=mi*(n*x+b);
   rn2 = norm(b-a*x);
   if (rn2 < eps*rb)
      break;
   end
end
```

Gauss-Seidel method

$$A = L + D + U \Rightarrow M = (L + D)$$

```
m = tril(a);
n = -triu(a,1);
bn2=norm(b);
for i=1:itmax
   x=m\(n*x+b);
   rn2 = norm(b-a*x);
   if (rn2 < eps*rb)
      break;
   end
end
```

Convergence

A matrix A is

- Diagonally dominant if $|a_{jj}| \geq \sum_{i \neq j} |a_{ij}|, j = 1 \ldots n$;
- Strictly diagonally dominant if $|a_{jj}| > \sum_{i \neq j} |a_{ij}|, j = 1 \ldots n$;
- Irreducibly diagonally dominant if $|a_{jj}| \geq \sum_{i \neq j} |a_{ij}|, j = 1 \ldots n$, with strict inequality for at least one value of j, and is irreducible.

Theorem (Convergence of Jacobi and Gauss-Seidel)

If A is either strictly diagonally dominant or irreducibly diagonally dominant, then the Jacobi and Gauss-Seidel iterations converge.

In particular this is true for M-matrices; see [1] for a proof.

SOR (Successive Over-Relaxation) method

$$M_\omega x_{k+1} = N_\omega x_k + \omega b$$

with $M_\omega = (\omega L + D)$ e $N_\omega = (1-\omega)D - \omega U$

SOR (Successive Over-Relaxation) method

$$M_\omega x_{k+1} = N_\omega x_k + \omega b$$

with $M_\omega = (\omega L + D)$ e $N_\omega = (1-\omega)D - \omega U$

```
m = omega*tril(a,-1) + diag(diag(a));
n = (1-omega)*diag(diag(a))-omega*triu(a,1);
bn2=norm(b);
for i=1:itmax
   x=m\(n*x+b);
   rn2 = norm(b-a*x);
   if (rn2 < eps*rb)
      break;
   end
end
```

Orthogonal polynomials

Consider a space of polynomials with the scalar product :

$$(p, q) = \int_D pq d\mu$$

and define an orthogonal basis:

$$\{p_i(x)\}$$

with monic polynomials

$$p_i(x) = x^i + \sum_{0 \le k \le i-1} a_k x^k.$$

We now prove that the basis can be built by induction starting from

$$p_0(x) = 1 \qquad p_1(x) = x - \alpha_1;$$

with the induction step defined as

$$p_{n+1}(x) = x p_n(x) - \alpha_{n+1} p_n(x) - \beta_{n+1} p_{n-1}(x)$$

Let us apply the orthogonality condition:

$$0 = \int p_{n+1} p_n = \int x p_n p_n - \alpha_{n+1} \int p_n p_n - \beta_{n+1} \int p_n p_{n-1}.$$

By induction the third term is zero. Thus

$$\int x p_n^2 - \alpha_{n+1} \int p_n^2 = 0 \Rightarrow \alpha_{n+1} = \frac{\int x p_n^2}{\int p_n^2}.$$

Applying orthogonality again

$$0 = \int p_{n+1} p_{n-1} = \int x p_n p_{n-1} - \alpha_{n+1} \int p_n p_{n-1} - \beta_{n+1} \int p_{n-1}^2 \Rightarrow$$

$$\int x p_n p_{n-1} - \beta_{n+1} \int p_{n-1}^2 = 0 \Rightarrow \beta_{n+1} = \frac{\int x p_n p_{n-1}}{\int p_{n-1}^2},$$

and all other terms are zero:

$$\int x p_n p_{n-2} = \int p_n g_{n-1}$$

Orthogonal polynomials

We have just proved that *all* orthogonal polynomial bases can be built by induction

$$
\begin{aligned}
p_0(x) &= 1 \\
p_1(x) &= x - \alpha_1 \\
p_{n+1}(x) &= x p_n(x) - \alpha_{n+1} p_n(x) - \beta_{n+1} p_{n-1}(x) \\
\alpha_{n+1} &= \frac{\int x p_n^2}{\int p_n^2} \\
\beta_{n+1} &= \frac{\int x p_n p_{n-1}}{\int p_{n-1}^2}
\end{aligned}
$$

with the coefficients α_i and β_i depending on the function space and measure, i.e. on the specific type of scalar product. This applies to polynomials of: Jacobi (Gegenbauer, Legendre, Chebychev), Laguerre, Hermite.

Consider the sequence $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$,

$$x^{(k+1)} = Gx^{(k)} + c;$$

let us build an approximation y

$$y^{(k)} = \sum_{0 \le j \le k} \nu_j(k) x^{(j)},$$

with the constraint that the exact solution should be reproducible

$$\sum_{0 \le j \le k} \nu_j(k) = 1.$$

Now we want to minimize the error

$$y^{(k)} - x = \sum_{0 \le j \le k} \nu_j(k)(x^{(j)} - x) = \sum_{0 \le j \le k} \nu_j(k) G^j e_0,$$

and taking norms we have

$$\|y^{(k)} - x\| \le \|p_k(G)\| \|e_0\|$$

$$p_k(z) = \sum_{0 \le j \le k} \nu_j(k) z^j$$

When G is symmetric we have

$$-1 < \alpha \le \lambda_1 \le \lambda_2 \ldots \lambda_n \le \beta < 1,$$

hence

$$\|p_k(G)\|_2 = \max_{\alpha \le \lambda \le \beta} |p_k(\lambda)|;$$

therefore we are looking for a polynomial p_k which takes a small absolute value on $[\alpha, \beta]$; moreover we need $p_k(1) = 1$.
Chebychev polynomials:

$$\begin{aligned} c_0(x) &= 1 \qquad c_1(x) = x \\ c_n(x) &= 2xc_{n-1}(x) - 2c_{n-2}(x), \end{aligned}$$

fit the bill.

Hence:

$$p_k(z) = \frac{c_k\left(-1 + 2\frac{z-\alpha}{\beta-\alpha}\right)}{c_k(\mu)},$$

with

$$\mu = -1 + 2\frac{1-\alpha}{\beta-\alpha},$$

satisfies our needs; therefore we can build

$$\|y^{(k)} - x\|_2 \leq \frac{\|x - x^{(0)}\|_2}{|c_k(\mu)|},$$

and for a large μ we can have a significant acceleration of convergence.

Note that $y^{(k)}$ is built via the recurrence relations

$$\begin{aligned} \omega_{k+1} &= 2\frac{2-\beta-\alpha}{\beta-\alpha}\frac{c_k(\mu)}{c_{k+1}(\mu)}, \\ y^{(k+1)} &= \omega_{k+1}(y^{(k)} - y^{(k-1)} + \gamma z^{(k)}) + y^{(k-1)}, \\ Mz^{(k)} &= b - Ay^{(k)}, \\ \gamma &= 2/(2-\alpha-\beta). \end{aligned}$$

Where is the catch? We need to have estimates on the eigenvalues!

Final considerations

Congratulations!

You are now acquainted with the state of the art in iterative solution techniques.

Final considerations

Congratulations!

You are now acquainted with the state of the art in iterative solution techniques.
And this is 1952.

Final considerations

Congratulations!

You are now acquainted with the state of the art in iterative solution techniques.
And this is 1952.
Queen Elizabeth II has just acceeded to the throne of the UK, the summer Olympics are going to be held in Helsinki, Sputnik and the space race are still a few years in the future.
Italy's government is headed by Alcide De Gasperi, the president is Luigi Einaudi.

In the next few lectures we will see what happens in this year, then we will jump on our DeLorean DMC-12 and check the future status in 1970, 1990, 2000, 2025.

References

3 Bibliography

[1] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718003. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718003.

[2] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.