# MÉTHODES ET PROGRAMMATION NUMÉRIQUES AVANCÉES

## Iterative solvers

Cédric Chevalier

January 15, 2025

# Introduction to Iterative Methods

# Iterative solver for linear systems

Given a matrix $A$ and a vector $b$, solving a linear equation $Ax = b$ is finding the vector $x$.
$A$ is the linear operator, $b$ the right hand side and $x$ the unknown.
We can compute $x$:

- directly, using Gaussian elimination techniques or determinant.
- iteratively, creating an iterative sequence $(x^{(k)})$ converging to the solution $x$

# Generic Iterative Solver

## Definition (Recuring relation)

$$x^{(k+1)} = F(x^{(k)}).$$

## The goal is to define a function $F$ such as

- it is easy to compute
- it converges rapidly, meaning few iterations are needed to have a good approximation of the solution

# Iterative solver algorithm

```
Vector iterative_solve(Matrix A, Vector b, Vector guess) {
  int iter = 0;
  Vector current = guess;
  while (!checkConvergence(current, iter)) {
    current = F(guess);
    iter++;
  }
}
```

- we need to define $F$
- we need to define when to stop the algorithm
  - intuitively, stop when the solution is *"good enough"*.

# Example: Richardson iteration

The Richardson iteration is $x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)})$, with a given $\omega \neq 0$.

## Convergence

For some matrices, it is possible to choose $\omega$ that makes the sequence $(x^{(k)})$ to converge towards a value $x^*$
For *k large enough*, $x^{(k+1)} \approx x^{(k)} \approx x^*$, so

$$\omega(b - Ax^*) = 0$$

and therefore $x^*$ is solution of $Ax = b$.

# Error

## Definition (Error $e^{(k)}$)

The error $e^{(k)}$ at iteration $k$ is $e^{(k)} = x - x^{(k)}$.

- Usually, we focus on a norm of the error $||e^{(k)}||$
- It is not always possible to compute on the fly: what is the value of $x$?

# Residual

**Definition (Residual $r^{(k)}$)**

The residual $r^{(k)}$ at iteration $k$ is $r^{(k)} = Ax^{(k)} - b$

- Usually, we focus on a norm of the residual $||r^{(k)}||$
- Easy to compute

## Relation between error and residual

Given $\tilde{x}$, the associated residual $\tilde{r} = A\tilde{x} - b$, and the associated error $\tilde{e}$ we have:

- $$\frac{||\tilde{e}||}{||x||} = \frac{||x - \tilde{x}||}{||x||} = \frac{||A^{-1}\tilde{r}||}{||x||} \leq \frac{||A^{-1}|| \cdot ||\tilde{r}||}{||x||} \leq \frac{||A|| \cdot ||A^{-1}|| \cdot ||\tilde{r}||}{||b||} = \kappa(A)\frac{||\tilde{r}||}{||b||},$$

  where $\kappa(A) = ||A||||A^{-1}||$ is the condition number of $A$
- On the other side:

$$\frac{1}{\kappa(A)}\frac{||\tilde{r}||}{||b||} \leq \frac{||b - A\tilde{x}||}{||A|| \cdot ||A^{-1}|| \cdot ||b||} \leq \frac{||A(A^{-1}b - \tilde{x})||}{||A|| \cdot ||x||} \leq \frac{||x - \tilde{x}||}{||x||} = \frac{||\tilde{e}||}{||x||}$$

### Theorem (Error bounds)

$$\frac{1}{\kappa(A)}\frac{||\tilde{r}||}{||b||} \leq \frac{\tilde{e}}{||x||} \leq \kappa(A)\frac{||\tilde{r}||}{||b||}$$

# Stopping criteria

Given a threshold $t$

## Absolute Residual

$||r^{(k)}|| \leq t$

## Relative Residual

$\frac{||r^{(k)}||}{||b||} \leq t$

## Progress

$\frac{||r^{(k)}||}{||r_0||} \leq t$

## Number of iterations

$k > \text{Max}_{iters}$

# Stationary Iterative Methods

# General Scheme

- Write $A = M - N$, where $M$ is *easily invertible*
- We define an iterative method with this relation:

$$Mx^{(k+1)} = Nx^{(k)} + b.$$

# General Scheme

- Write $A = M - N$, where $M$ is *easily invertible*
- We define an iterative method with this relation:
$$Mx^{(k+1)} = Nx^{(k)} + b.$$
- The iteration matrix $C$ such that $e^{(k+1)} = Ce^{(k)}$ is thus:
$$C = I - M^{-1}A = M^{-1}N$$

# General Scheme

- Write $A = M - N$, where $M$ is *easily invertible*
- We define an iterative method with this relation:
$$Mx^{(k+1)} = Nx^{(k)} + b.$$
- The iteration matrix $C$ such that $e^{(k+1)} = Ce^{(k)}$ is thus:
$$C = I - M^{-1}A = M^{-1}N$$
- The iterative method converges iff the spectral radius $\rho(C) < 1$ (so $\lim_{k\to\infty} C^k = 0$).

# Jacobi

$A = D + L + U$, we will take $M = D$ and $N = -(L + U)$.
We have the relation

$$Dx^{(k+1)} = -(L + U)x^{(k)} + b.$$

# Jacobi

$A = D + L + U$, we will take $M = D$ and $N = -(L + U)$.
We have the relation

$$Dx^{(k+1)} = -(L + U)x^{(k)} + b.$$

For each row $i$:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right).$$

# Gauss-Seidel

$A = D + L + U$, we will take $M = D + L$ and $N = -U$.
We have the relation

$$(D + L)x^{(k+1)} = -Ux^{(k)} + b.$$

# Gauss-Seidel

$A = D + L + U$, we will take $M = D + L$ and $N = -U$.
We have the relation

$$(D + L)x^{(k+1)} = -Ux^{(k)} + b.$$

For each row $i$:

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

# Richardson iteration

For $\omega \neq 0$, $M = \frac{1}{\omega} I$, where $I$ is the identity matrix of size $n$

# Richardson iteration

For $\omega \neq 0$, $M = \frac{1}{\omega} I$, where $I$ is the identity matrix of size $n$
We have the relation

$$x^{(k+1)} = x^{(k)} + \omega \left( b - A x^{(k)} \right).$$

# Damped Jacobi method

It is a variant of the Jacobi method:
For $\omega \neq 0$, we choose $M = \frac{1}{\omega} D$

# Damped Jacobi method

It is a variant of the Jacobi method:
For $\omega \neq 0$, we choose $M = \frac{1}{\omega} D$
We have the relation:

$$x^{(k+1)} = \omega D^{-1} \left( b - (L + U) x^{(k)} \right) + (1 - \omega) x^{(k)}$$

# Successive over-relaxation (SOR)

SOR is a variant of the Gauss-Seidel method:
For $\omega \neq 0$, we take $M = D + \omega L$

# Successive over-relaxation (SOR)

SOR is a variant of the Gauss-Seidel method:
For $\omega \neq 0$, we take $M = D + \omega L$
We have the relation:

$$x^{(k+1)} = (D + \omega L)^{-1} \left( \omega b - (\omega U + (\omega - 1) D) x^{(k)} \right)$$

# Successive over-relaxation (SOR)

SOR is a variant of the Gauss-Seidel method:
For $\omega \neq 0$, we take $M = D + \omega L$
We have the relation:

$$x^{(k+1)} = (D + \omega L)^{-1} \left( \omega b - (\omega U + (\omega - 1) D) x^{(k)} \right)$$

or

$$x^{(k+1)} = (1 - \omega) x^{(k)} + \omega D^{-1} \left( b - L x^{(k+1)} - U x^{(k)} \right)$$

# Conjuguate Gradient

# Exploration

## Experiments

https://github.com/cedricchevalier19/mpna/tree/main/cg

## Based on:

https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf
https://ilyakuzovkin.com/ml-ai-rl-cs/
the-concept-of-conjugate-gradient-descent-in-python/

# Algorithm

$r_0 \leftarrow b - Ax_0$;
$p_0 \leftarrow r_0$;
**for** $k \leftarrow 0, \ldots$ **do**
    $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$;
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
    $r_{k+1} \leftarrow r_k - \alpha_k A p_k$;
    **if** $r_{k+1}$ *is sufficiently small* **then**
        exit loop;
    **end**
    $\beta_k \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$;
    $p_{k+1} \leftarrow r_{k+1} + \beta_k p_k$;
**end**
return $x_{k+1}$;

- Works for symmetric positive definite matrix $A$
- Converges in $n$ iterations
- Residual is computed implicitly

# Algorithm

$r_0 \leftarrow b - Ax_0$;
$p_0 \leftarrow r_0$;
**for** $k \leftarrow 0, \ldots$ **do**
    $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$;
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
    $r_{k+1} \leftarrow r_k - \alpha_k A p_k$;
    **if** $r_{k+1}$ *is sufficiently small* **then**
        exit loop;
    **end**
    $\beta_k \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$;
    $p_{k+1} \leftarrow r_{k+1} + \beta_k p_k$;
**end**
return $x_{k+1}$;

- Works for symmetric positive definite matrix $A$
- Converges in $n$ iterations
- Residual is computed implicitly
  - In finite precision, it might be useful to explicitly compute residual from time to time
- Fastest convergence, so usually the number of iterations is low

# Krylov Methods

# Krylov Methods

## Definition (Definition)

In linear algebra, the order-$r$ Krylov subspace generated by an $n \times n$ matrix $A$ and a vector $b$ of dimension $n$ is the linear subspace spanned by the images of $b$ under the first $r$ powers of $A$ (starting from $A^0 = I$), that is,

$$\mathcal{K}_r(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{r-1}b\}.$$

## Properties

Conjugate Gradient is a Krylov method: it builds a Krylov subspace and express the new iterate in this subspace

# Some Krylov methods for solving linear equations

- Conjugate gradient for symmetric positive define matrices
- BiCG, BiCGSTAB, BiCGSTAB(l): might fail, *lucky* or *unlucky* breakdown
- MINRES, for symmetric matrices
- GMRES: Converges, but not always in practice

# BiCG

We solve together two equations: $Ax = b$ and $x^*A^* = b*$ (conjugate transpose)

1 Choose initial guess $x_0$, two other vectors $x_0^*$ and $b^*$;
2 $r_0 \leftarrow b - A\,x_0$;
3 $r_0^* \leftarrow b^* - x_0^*\,A^*$;
4 $p_0 \leftarrow r_0$;
5 $p_0^* \leftarrow r_0^*$;
6 **for** $k = 0, 1, \ldots$ **do**
7 $\quad \alpha_k \leftarrow \frac{r_k^* r_k}{p_k^* A p_k}$;
8 $\quad x_{k+1} \leftarrow x_k + \alpha_k \cdot p_k$;
9 $\quad x_{k+1}^* \leftarrow x_k^* + \overline{\alpha_k} \cdot p_k^*$;
10 $\quad r_{k+1} \leftarrow r_k - \alpha_k \cdot A p_k$;
11 $\quad r_{k+1}^* \leftarrow r_k^* - \overline{\alpha_k} \cdot p_k^*\,A^*$;
12 $\quad \beta_k \leftarrow \frac{r_{k+1}^* r_{k+1}}{r_k^* r_k}$;
13 $\quad p_{k+1} \leftarrow r_{k+1} + \beta_k \cdot p_k$;
14 $\quad p_{k+1}^* \leftarrow r_{k+1}^* + \overline{\beta_k} \cdot p_k^*$;
15 **end**

# GMRES

## Principle

GMRES (Generalized Minimal RESidual) approximates the exact solution of $Ax = b$ by the vector $ $ x_n \in x_0 + K_n $ $ that minimizes the Euclidean norm of the residual $r_n = b - Ax_n$.

## In practice

The GMRES algorithm is implemented with the Arnoldi iteration for numerical stability. The Arnoldi iteration produces $H_n$, an $(n+1) \times n$ upper Hessenberg matrix, and $Q_n$, the matrix containing the basis vectors of $K_n(A, b)$, such that $AQ_n = Q_{n+1}H_n$.
We are looking for $x_n = Q_n y_n + x_0$ for some $y_n R^n$ which minimizes the norm of $bAx_n$.
Since the columns of $Q_n$ are orthonormal, we can compute the residual equivalently as

$$||b - Ax_n||_2 = ||Q_{n+1}(\beta e_1 - H_n y_n)||_2 = ||H_n y_n - \beta e_1||_2$$

# GMRES

```
1  Q ← empty(size(b, k + 1));
2  H ← zeros(k + 1, k);
3  r_0 ← b − A ∗ x_0;
4  Q[:, 0] ← r_0 / ||r_0||;
5  for n ← 1 . . . k do
6  |   Set entries of Q and H as an Arnoldi iteration;
7  |   Compute the residual res and the least squares
   |    solution y_n for the part of H so far created;
8  |   if res < tol then
9  |   |   break;
10 |   end
11 end
12 return Q[:, n + 1], res;
```

# GMRES

```
1  Q ← empty(size(b, k + 1));
2  H ← zeros(k + 1, k);
3  r_0 ← b − A ∗ x_0;
4  Q[:, 0] ← r_0 / ||r_0||;
5  for n ← 1 . . . k do
6  |   Set entries of Q and H as an Arnoldi iteration;
7  |   Compute the residual res and the least squares
   |    solution y_n for the part of H so far created;
8  |   if res < tol then
9  |   |   break;
10 |   end
11 end
12 return Q[:, n + 1], res;
```

- No more 3-terms recurrence: memory consumption is high!
- Orthogonalization is very sensitive to rounding errors

# GMRES

$Q \leftarrow \text{empty}(\text{size}(b, k+1))$;
$H \leftarrow \text{zeros}(k+1, k)$;
$r_0 \leftarrow b - A * x_0$;
$Q[:, 0] \leftarrow \frac{r_0}{||r_0||}$;
**for** $n \leftarrow 1 \dots k$ **do**
    Set entries of $Q$ and $H$ as an Arnoldi iteration;
    Compute the residual *res* and the least squares solution $y_n$ for the part of $H$ so far created;
    **if** $res < tol$ **then**
        break;
    **end**
**end**
return $Q[:, n+1], res$;

- No more 3-terms recurrence: memory consumption is high!
- Orthogonalization is very sensitive to rounding errors

In practice, we choose a maximum size for the basis and we *restart* the algorithm from the current solution

# Arnoldi iteration

## Skeleton

With an arbitrary vector $q_1$ with norm 1 ;
**for** $k \leftarrow 2 \ldots$ **do**
    $q_k \leftarrow A q_{k-1}$ ;
    **for** $k \leftarrow 1 \ldots k-1$ **do**
        $h_{j,k-1} \leftarrow q_j^* q_k$ ;
        $q_k \leftarrow q_k - h_{j,k-1} q_j$ ;
    **end**
    $h_{k,k-1} \leftarrow ||q_k||$ ;
    $q_k \leftarrow \frac{q_k}{h_{k,k-1}}$ ;
**end**

# Preconditionners

# Preconditionner principle

## Right

We solve $AP^{-1}(Px) = b$.

## Left

We solve $P^{-1}(Ax - b) = 0$

## Two-Sided

$QAP^{-1}(Px) = Qb$.