
# Mini-course on Stationary Iterative Methods

Lecture 2 – Fixed point methods

Felix Kwok
Département de mathématiques et de statistique, Université Laval

May 31, 2021

2021 CRM Summer School: Solving large systems efficiently in multi-physics numerical simulations

# Outline

- Definitions and basic notions
- Iterative methods for linear problems

# Definitions and basic notions

# Iterative methods

Suppose we want to solve the nonlinear system

$$F(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2 \ldots, x_n) \\ f_2(x_1, x_2 \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2 \ldots, x_n) \end{bmatrix} = 0 \qquad \text{for} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

We assume that $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is (Fréchet-) differentiable.

- If $F(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$, then the system is linear and can be solved by e.g. Gaussian elimination. Iterative methods can also be used, see the next section.
- If $F$ is nonlinear, there is no *direct* method that solves $F(\mathbf{x}) = 0$ in a finite number of operations for *arbitrary* $F$.
- An **iterative method** generates a sequence of approximations $\mathbf{x}^1, \mathbf{x}^2, \ldots,$ such that $\mathbf{x}^k$ (hopefully) converges to the solution $\mathbf{x}^*$ as $k \to \infty$.

## Example : Newton's method

Newton's method starts with an initial guess $\mathbf{x}^0$ and generates the next iterate $\mathbf{x}^1$ as follows :

## Example : Newton's method

Newton's method starts with an initial guess $\mathbf{x}^0$ and generates the next iterate $\mathbf{x}^1$ as follows :

1. Approximate $F(\mathbf{x})$ by a first order Taylor expansion about the point $\mathbf{x}^0$ :

$$F(\mathbf{x}) \approx F(\mathbf{x}^0) + F'(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0) + O(\|\mathbf{x} - \mathbf{x}^0\|^2).$$

Here, $F'(\mathbf{x}^0)$ is the **Jacobian** matrix :

$$F'(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}^0) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}^0) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}^0) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}^0) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}^0) & \cdots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}^0) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}^0) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}^0) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}^0) \end{bmatrix}.$$

## Example : Newton's method

Newton's method starts with an initial guess $\mathbf{x}^0$ and generates the next iterate $\mathbf{x}^1$ as follows :

1. Approximate $F(\mathbf{x})$ by a first order Taylor expansion about the point $\mathbf{x}^0$ :

$$F(\mathbf{x}) \approx F(\mathbf{x}^0) + F'(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0) + O(\|\mathbf{x} - \mathbf{x}^0\|^2).$$

2. Find the root of the *linearized* equation and call it $\mathbf{x}^1$ :

$$0 = F(\mathbf{x}^0) + F'(\mathbf{x}^0)(\mathbf{x}^1 - \mathbf{x}^0) \implies \mathbf{x}^1 = \mathbf{x}^0 - (F'(\mathbf{x}^0))^{-1} F(\mathbf{x}^0).$$

## Example : Newton's method

Newton's method starts with an initial guess $\mathbf{x}^0$ and generates the next iterate $\mathbf{x}^1$ as follows :

1. Approximate $F(\mathbf{x})$ by a first order Taylor expansion about the point $\mathbf{x}^0$ :

$$F(\mathbf{x}) \approx F(\mathbf{x}^0) + F'(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0) + O(\|\mathbf{x} - \mathbf{x}^0\|^2).$$

2. Find the root of the *linearized* equation and call it $\mathbf{x}^1$ :

$$0 = F(\mathbf{x}^0) + F'(\mathbf{x}^0)(\mathbf{x}^1 - \mathbf{x}^0) \implies \mathbf{x}^1 = \mathbf{x}^0 - (F'(\mathbf{x}^0))^{-1} F(\mathbf{x}^0).$$

3. Repeat the process using the newly calculated point :

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (F'(\mathbf{x}^k))^{-1} F(\mathbf{x}^k), \qquad k = 1, 2, \ldots$$

## Example : Newton's method

Newton's method starts with an initial guess $\mathbf{x}^0$ and generates the next iterate $\mathbf{x}^1$ as follows :

1. Approximate $F(\mathbf{x})$ by a first order Taylor expansion about the point $\mathbf{x}^0$ :

$$F(\mathbf{x}) \approx F(\mathbf{x}^0) + F'(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0) + O(\|\mathbf{x} - \mathbf{x}^0\|^2).$$

2. Find the root of the *linearized* equation and call it $\mathbf{x}^1$ :

$$0 = F(\mathbf{x}^0) + F'(\mathbf{x}^0)(\mathbf{x}^1 - \mathbf{x}^0) \implies \mathbf{x}^1 = \mathbf{x}^0 - (F'(\mathbf{x}^0))^{-1} F(\mathbf{x}^0).$$

3. Repeat the process using the newly calculated point :

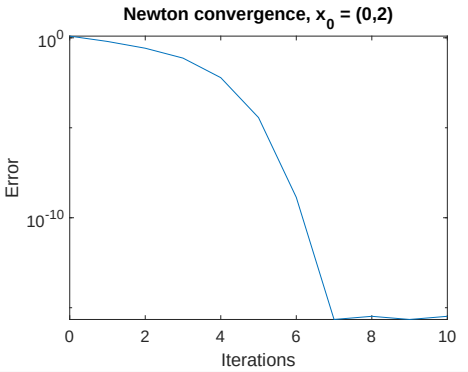
$$\mathbf{x}^{k+1} = \mathbf{x}^k - (F'(\mathbf{x}^k))^{-1} F(\mathbf{x}^k), \qquad k = 1, 2, \ldots$$

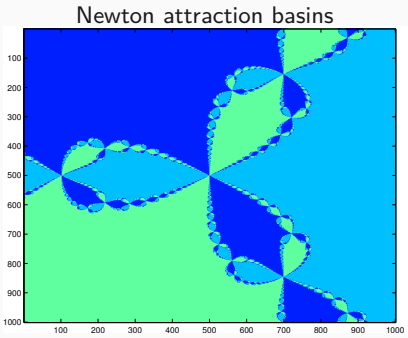
4. Stop when $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$ are very close (or when $\|F(\mathbf{x}^k)\|$ is very small, or both).

## Example : Newton's method

Apply Newton's method to the real and imaginary parts of the equation $z^3 - 1 = 0$ for $z = x + iy$ :

$$F(x,y) = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix} = \begin{bmatrix} x^3 - 3xy^2 - 1 \\ 3x^2y - y^3 \end{bmatrix} = 0$$



## Newton's method – remarks

- Instead of computing the matrix inverse in $\mathbf{x}^{k+1} = \mathbf{x}^k - (F'(\mathbf{x}^k))^{-1}F(\mathbf{x}^k)$ the Newton step is usually implemented as follows :
  1. Solve $F'(\mathbf{x}^k)\mathbf{z} = F(\mathbf{x}^k)$ for $\mathbf{z}$ *using an appropriate linear solver* ;
  2. Update $\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{z}$.

  A good linear solver for $F'$ is therefore crucial !
- Newton's method can diverge or stagnate when $\mathbf{x}_0$ is too far from the solution !
- Jacobian evaluation is expensive ; one could freeze the Jacobian at the initial guess (or update every few iterations) to obtain

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (F'(\mathbf{x}^0))^{-1}F(\mathbf{x}^k), \qquad k = 0, 1, 2, \ldots$$

  This is known as the **simplified Newton** method.

## Fixed point methods

- Newton's method can be written in the general form $\mathbf{x}^{k+1} = G(\mathbf{x}^k)$, where $G : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is the **fixed point map**. For Newton, we have

$$G(\mathbf{x}) = \mathbf{x} - (F'(\mathbf{x}))^{-1} F(\mathbf{x}).$$

- If a fixed point method converges and $G$ is continuous, then the limit point $\mathbf{x}^*$ satisfies

$$\mathbf{x}^* = \lim_{k\to\infty} \mathbf{x}^k = \lim_{k\to\infty} G(\mathbf{x}^{k-1}) = G\left(\lim_{k\to\infty} \mathbf{x}^{k-1}\right) = G(x^*).$$

  The method therefore converges to a **fixed point** of $G$.

- **Example :** the fixed point for Newton's method is

$$\mathbf{x}^* = G(\mathbf{x}^*) = \mathbf{x}^* - (F'(\mathbf{x}^*))^{-1} \mathcal{F}(\mathbf{x}^*) \implies (F'(\mathbf{x}^*))^{-1} F(\mathbf{x}^*) = 0.$$

  If $F'(\mathbf{x}^*)$ is non-singular, then $F(\mathbf{x}^*) = 0 \implies \mathbf{x}^*$ is a solution to the nonlinear system.

UNIVERSITÉ LAVAL

**Definition :** Let $\mathbf{x}^*$ be a fixed point of $G$ and let $\mathbf{x}^0, \mathbf{x}^1, \ldots$ be iterates that satisfy $\mathbf{x}^{k+1} = G(\mathbf{x}^k)$, $k = 0, 1, 2, \ldots$. Then the **error** of the $k$th iterate is defined as

$$\mathbf{e}^k := \mathbf{x}^k - \mathbf{x}^*.$$

To understand how the error behaves as a function of $k$, we calculate

$$\begin{aligned}
\mathbf{e}^{k+1} &= \mathbf{x}^{k+1} - \mathbf{x}^* \\
&= G(\mathbf{x}^k) - G(\mathbf{x}^*) \\
&= G(\mathbf{x}^*) + G'(\mathbf{x}^*)(\mathbf{x}^k - \mathbf{x}^*) + O(\|\mathbf{x}^k - \mathbf{x}^*\|^2) - G(\mathbf{x}^*) \\
&= G'(\mathbf{x}^*)\mathbf{e}^k + O(\|\mathbf{e}^k\|^2).
\end{aligned}$$

# Example 1

To find one of the roots of $x^3 - 3x + 1 = 0$, we start with $x_0 = 0$ and consider the method

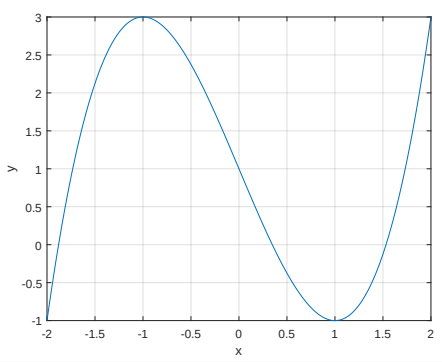
$$x^{k+1} = \frac{1}{3}((x^k)^3 + 1).$$

Then $G(x) = \frac{1}{3}(x^3+1)$, $G'(x) = x^2$. For the fixed point $x^* \approx 0.3473$, we have $G'(x^*) \approx 0.1206$, so for $x^k$ close to $x^*$, we have

$$e^{k+1} \approx 0.1206\, e^k + O((e^k)^2),$$

so the error is reduced by about a factor of $8$ with each additional iteration.

$y = x^3 - 3x + 1$



| $k$ | $x^k$ | $e^k$ | $e^{k+1}/e^k$ |
|---|---|---|---|
| 0 | 0 | -0.347296355333861 | 0.040204919476061 |
| 1 | 0.333333333333333 | -0.013963022000527 | 0.115830440439082 |
| 2 | 0.345679012345679 | -0.001617342988182 | 0.120053933035803 |
| 3 | 0.347102186947062 | -0.000194168386799 | 0.120547337021927 |
| 4 | 0.347272948851898 | -0.000023406481962 | 0.120606629621737 |
| 5 | 0.347293532356960 | -0.000002822976901 | 0.120613778000051 |
| 6 | 0.347296014843951 | -0.000000340489909 | 0.120614640035547 |
| 7 | 0.347296314265793 | -0.000000041068068 | 0.120614743217285 |
| 8 | 0.347296350380446 | -0.000000004953414 | 0.120614753443415 |
| 9 | 0.347296354736406 | -0.000000000597455 | 0.120614636167759 |
| 10 | 0.347296355261799 | -0.000000000072062 | – |

## What about Newton's method ?

In one dimension :

$$G(x) = x - \frac{f(x)}{f'(x)}$$
$$G'(x) = 1 - \frac{f'(x) \cdot f'(x) - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2}$$

- If $x^*$ is a root of $f(x)$ such that $f'(x^*) \neq 0$ (i.e., a simple root), then

$$G'(x^*) = \frac{0 \cdot f''(x^*)}{(f'(x^*))^2} = 0.$$

So for $|x^k - x^*|$ small, we have

$$e^{k+1} = G'(x^*)e^k + O(|e^k|^2) = O(|e^k|^2)$$

Thus, we have **quadratic convergence** !

## Example 2 : Newton's method

To solve $x^3 - 3x + 1 = 0$ using Newton's method, we write

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} = x^k - \frac{(x^k)^3 - 3x^k + 1}{3(x^k)^2 - 3}$$

The number of correct digits **doubles** after every iteration $\Longrightarrow$ quadratic convergence

| $k$ | $x^k$ | $e^k$ | $e^{k+1}/(e^k)^2$ |
|---|---|---|---|
| 0 | 0 | -0.347296355333861 | -0.115765451777953 |
| 1 | 0.333333333333333 | -0.013963022000527 | -0.380236133250081 |
| 2 | 0.347222222222222 | -0.000074133111638 | -0.394851507080994 |
| 3 | 0.3472963533163868 | -0.000000002169993 | 0 |
| 4 | 0.347296355333861 | 0 | – |

## Some problems with Newton

- Recall :

$$e^{k+1} = G'(x^*)e^k + O(|e^k|^2) = O(|e^k|^2)$$

  If $|e^k|$ is large, then

$$|e^{k+1}| \approx C|e^k|^2 > |e^k| \qquad \text{if } |Ce^k| > 1.$$

  So Newton's method may not converge in this case ; it may cycle or diverge.

- If $x^*$ is a **double root** of $f$, i.e., if $f(x^*) = f'(x^*) = 0$ (but $f''(x^*) \neq 0$), then

$$\lim_{x \to x^*} G'(x) = \lim_{x \to x^*} f''(x) \frac{f(x)}{(f'(x))^2} = f''(x^*) \lim_{x \to x^*} \frac{f'(x)}{2f'(x)f''(x)} = \frac{1}{2}.$$

  So Newton converges **linearly** with $1/2$ as the asymptotic contraction factor.

- A more involved analysis leads to analogous results for higher dimensions

## Problems with Newton : some fixes

UNIVERSITÉ LAVAL

- Line search :
  - Instead of using the full Newton correction $\mathbf{x}^{k+1} = \mathbf{x}_k - (F'(\mathbf{x}^k))^{-1}F(\mathbf{x}^k)$, make a shorter step
$$\mathbf{x}^{k+1} = \mathbf{x}_k - \alpha(F'(\mathbf{x}^k))^{-1}F(\mathbf{x}^k), \qquad 0 < \alpha < 1$$
  - $\alpha$ is chosen to ensure $\|F(\mathbf{x}^{k+1})\| < \|F(\mathbf{x}^k)\|$
  - Under some hypotheses, one can show convergence to a solution $F(\mathbf{x}^*) = 0$.
- Preconditioning : transform $F(x) = 0$ to some equivalent problem $H(x) = 0$ with the same solution $\mathbf{x}^*$, but which is easier to solve for Newton (e.g., closer to a linear function)
- Continuation : If the problem depends continuously on a parameter $\alpha$, i.e.,
$$F(x(\alpha); \alpha) = 0,$$
start with an "easy" $\alpha$ and vary $\alpha$ step by step, using the solution from the previous step as an initial guess

# Iterative methods for linear problems

Suppose we want to solve the **linear system**

$$\begin{bmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{bmatrix} \mathbf{x} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix}.$$

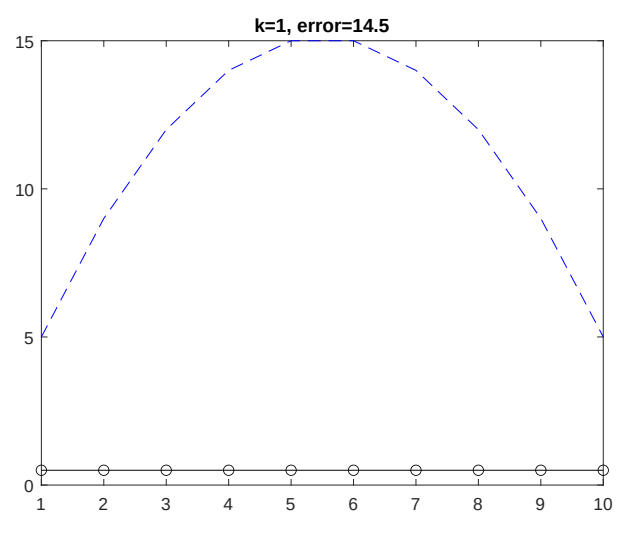
Instead of eliminating unknowns, Jacobi (1845) has the following idea :

1. Make some initial guess $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)^T$
2. Use the $i$th equation to solve for the new $x_i$ using the previous data :

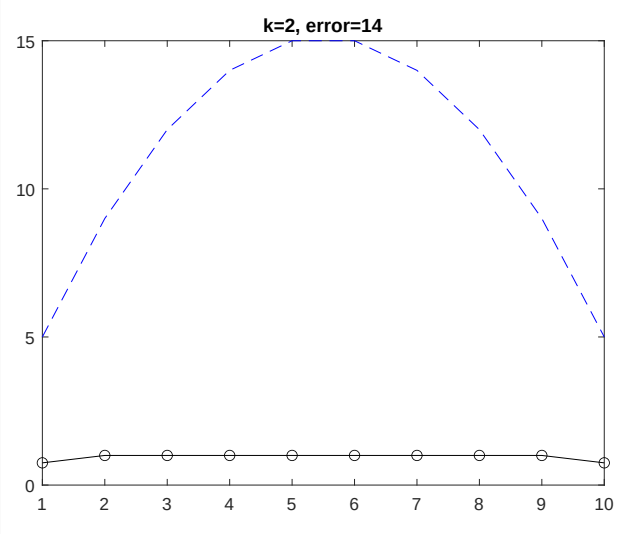
$$-x_{i-1} + 2x_i - x_{i+1} = f_i \implies x_i^{k+1} = \frac{1}{2}(x_{i-1}^k + x_{i+1}^k + f_i), \qquad k = 1, \ldots, n$$

3. Iterate until convergence (?!).

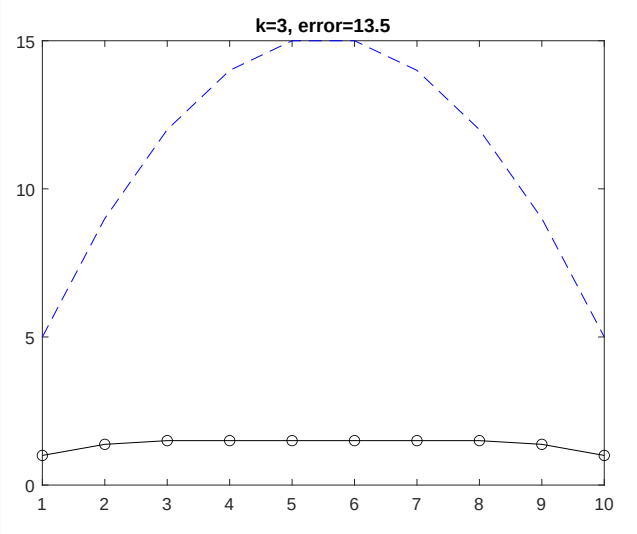
## Jacobi method, $n = 10$



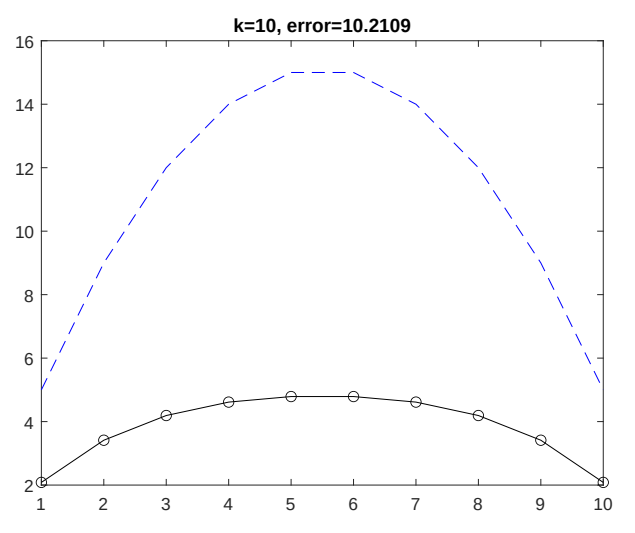
## Jacobi method, $n = 10$



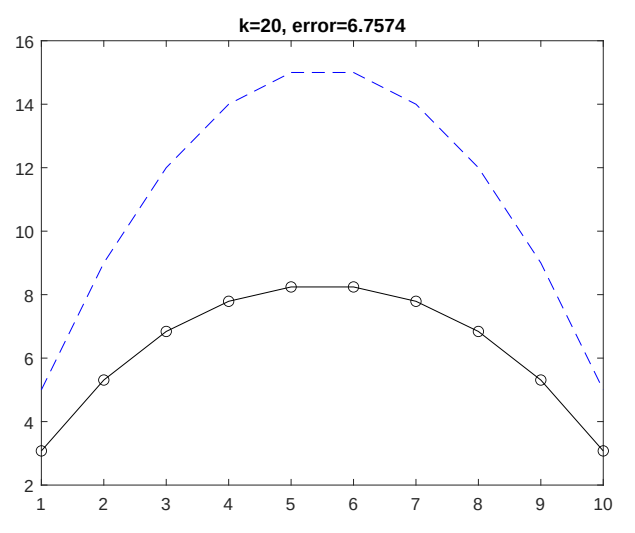
## Jacobi method, $n = 10$



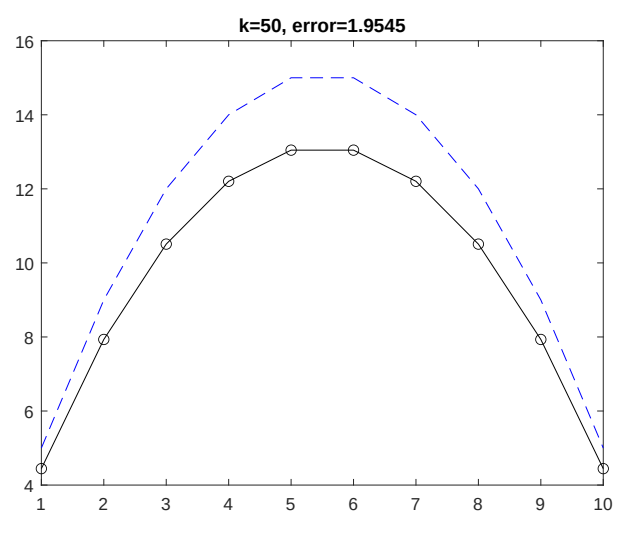
## Jacobi method, $n = 10$



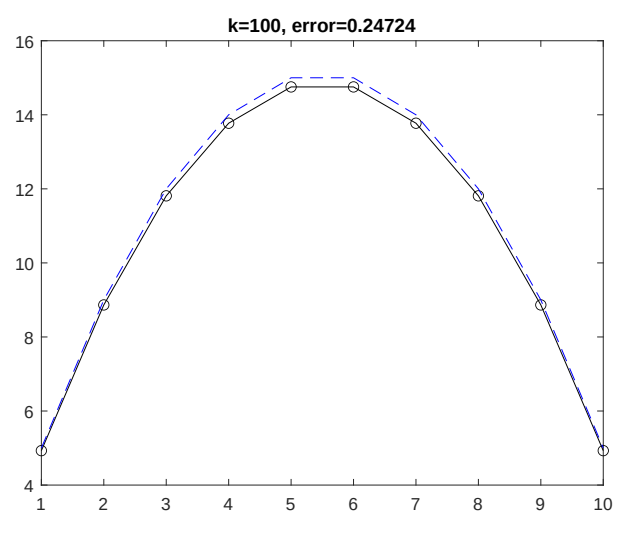
## Jacobi method, $n = 10$



## Jacobi method, $n = 10$



## Jacobi method, $n = 10$



# Gauss-Seidel method

Make a programming mistake :

- Jacobi :

$$x_i^{k+1} = \frac{1}{2}(x_{i-1}^k + x_{i+1}^k + f_i), \qquad i = 1, \ldots, n$$

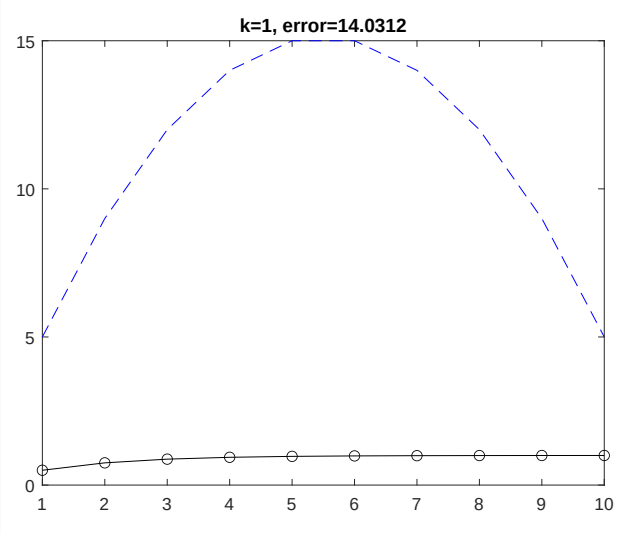
- Gauss-Seidel :

$$x_i^{k+1} = \frac{1}{2}(x_{i-1}^{k+1} + x_{i+1}^k + f_i), \qquad i = 1, \ldots, n$$

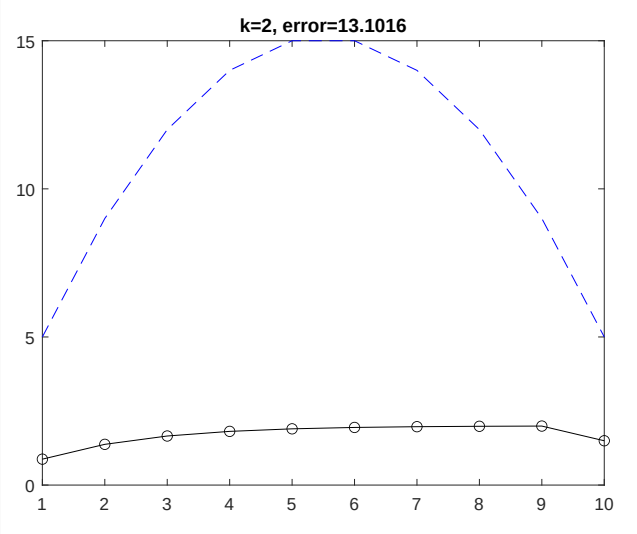
Gauss explained in a letter in 1823 how to use this method to solve a least squares problem :

> *"I recommend this method to you for imitation. You will hardly ever again eliminate directly, at least not when you have more than 2 unknowns. The indirect procedure can be done while half asleep, or while thinking about other things."*

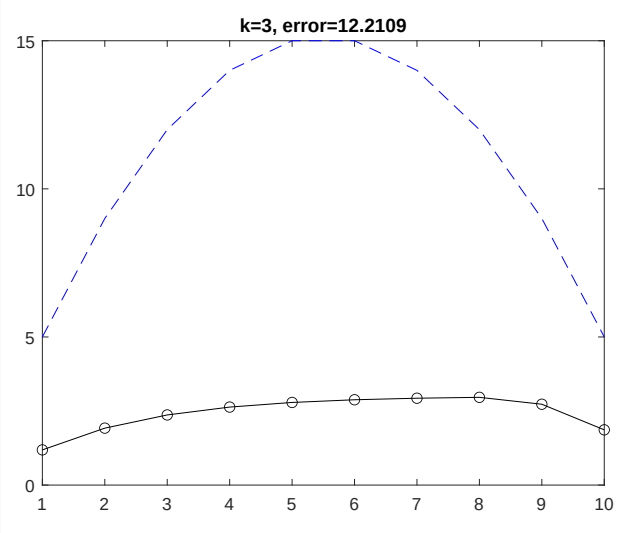
## Gauss-Seidel method, $n = 10$



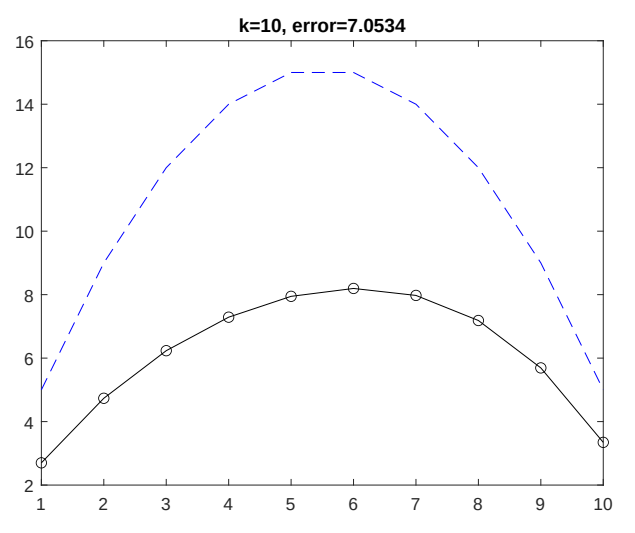
## Gauss-Seidel method, $n = 10$



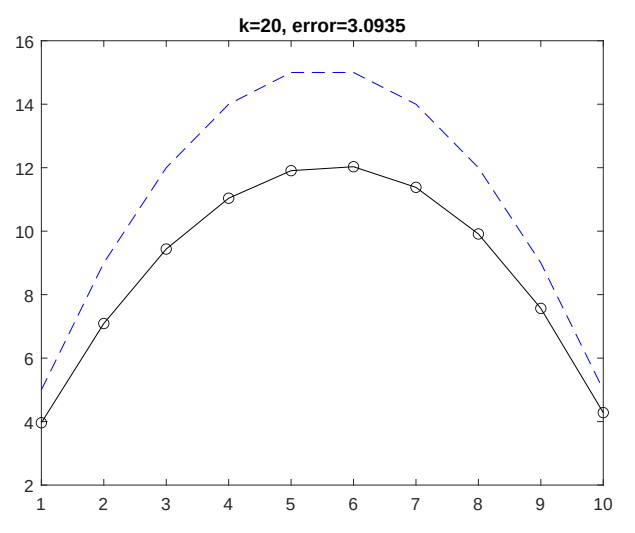
## Gauss-Seidel method, $n = 10$



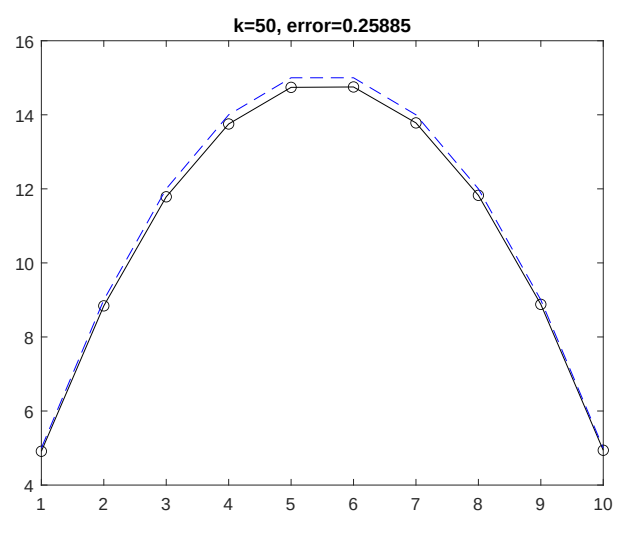
## Gauss-Seidel method, $n = 10$



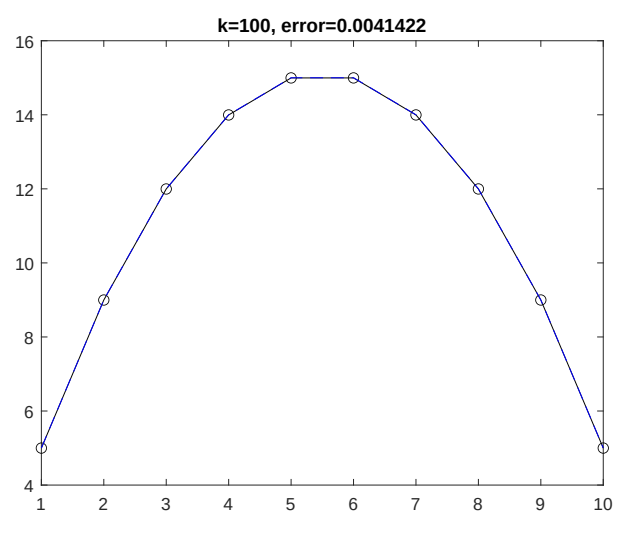
## Gauss-Seidel method, $n = 10$



## Gauss-Seidel method, $n = 10$



## Gauss-Seidel method, $n = 10$



## Stationary iterative method : General form

The Jacobi and Gauss-Seidel methods for solving $A\mathbf{x} = \mathbf{f}$ can be rewritten in the general form

$$M\mathbf{x}^{k+1} = N\mathbf{x}^k + \mathbf{f},$$

where $M$ is a non-singular matrix, and $N = M - A$.

- Jacobi : $M = D = \mathrm{diag}(a_{11}, a_{22}, \ldots, a_{nn})$
- Gauss-Seidel : $M = L + D = \begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & \cdots & \cdots & a_{nn} \end{bmatrix}$, i.e., the lower triangular part of $A$
- Other choices are possible, as long as systems involving $M$ are **cheap to solve**.

Note that we need $N = M - A$ so that the fixed point $\mathbf{x}^* = \lim \mathbf{x}^k$ satisfies

$$M\mathbf{x}^* = N\mathbf{x}^* + \mathbf{f} \iff (M - N)\mathbf{x}^* = \mathbf{f} \iff A\mathbf{x}^* = \mathbf{f}.$$

$$\boxed{M\mathbf{x}^{k+1} = N\mathbf{x}^k + \mathbf{f}}$$

- This is a **fixed point method**, because we can write it in terms of a fixed point map $\mathcal{G}(\mathbf{x})$ :

$$\mathbf{x}^{k+1} = M^{-1}N\mathbf{x}^k + M^{-1}\mathbf{f} =: \mathcal{G}(\mathbf{x}^k),$$

- Another way to write this is in terms of the **residual** $\mathbf{r}^k = \mathbf{f} - A\mathbf{x}^k$ :

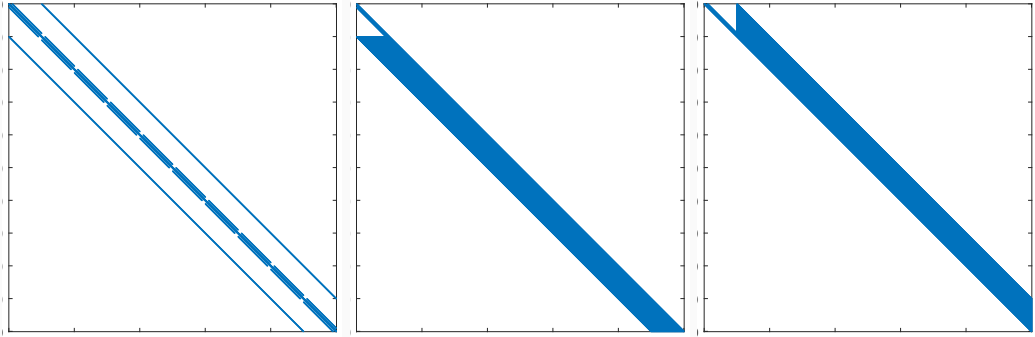
$$M\mathbf{x}^{k+1} = (M - A)\mathbf{x}^k + \mathbf{f} = M\mathbf{x}^k + \mathbf{r}^k \implies \mathbf{x}^{k+1} = \mathbf{x}^k + M^{-1}\mathbf{r}^k,$$

  so the method will keep updating $\mathbf{x}^{k+1}$ as long as $\mathbf{r}^k \neq 0$.
- A stopping criterion is usually needed to decide when the solution $\mathbf{x}^k$ is "close enough". Some options are $\|\mathbf{r}^k\|$, $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$, or some other norm of the error/residual.

# Direct vs iterative method

- Recall : cost of direct methods depends heavily on the density of $LU$ factors
- Iterative methods can be matrix free, with much lower cost per iteration
- But : speed of convergence is paramount !

$\mathrm{nnz}(A) = 6400$ $\mathrm{nnz}(L) = 91909$ $\mathrm{nnz}(U) = 91909$



## Convergence of a stationary iterative method

UNIVERSITÉ LAVAL

$$\boxed{M\mathbf{x}^{k+1} = N\mathbf{x}^k + \mathbf{f}} \qquad (*)$$

- Define the error $\mathbf{e}^k = \mathbf{x}^k - \mathbf{x}^*$.
- Subtracting the above from $(*)$ gives

$$M(\mathbf{x}^{k+1} - \mathbf{x}^*) = N(\mathbf{x}^k - \mathbf{x}^*) \implies M\mathbf{e}^{k+1} = N\mathbf{e}^k.$$

- Therefore, we have

$$\mathbf{e}^{k+1} = G\mathbf{e}^k, \qquad \text{where } G = M^{-1}N = I - M^{-1}A.$$

- Applying the above relation several times yields

$$\mathbf{e}^k = G\mathbf{e}^{k-1} = G(G\mathbf{e}^{k-2}) = G^2(G\mathbf{e}^{k-3}) = \cdots = G^k\mathbf{e}^0,$$

where $\mathbf{e}^0 = \mathbf{x}^0 - \mathbf{x}^*$ is the initial error (generally unknown).

Let $\mathbf{e}^0 \neq 0$ be an **eigenvector** of $G$, i.e., $G\mathbf{e}^0 = \lambda \mathbf{e}^0$ for some $\lambda$ (which may be complex). Then

$$\mathbf{e}^k = G^k \mathbf{e}^0 = \lambda G^{k-1} \mathbf{e}^0 = \cdots = \lambda^k \mathbf{e}^0 \implies \|\mathbf{e}^k\| = |\lambda|^k \|\mathbf{e}^0\|$$

Several cases :

- If $|\lambda| < 1$, then $|\lambda|^k \to 0$ as $k \to \infty \implies \|\mathbf{e}^k\| \to 0$, i.e., convergence !
- If $|\lambda| > 1$, then $|\lambda|^k \to \infty \implies \|\mathbf{e}^k\| \to \infty$, i.e., divergence.
- If $|\lambda| = 1$, then $\|\mathbf{e}^k\| = \|\mathbf{e}^0\|$ for all $k \implies$ no convergence to $\mathbf{x}^*$.

In general, $\mathbf{e}^0$ is a (usually unknown) linear combination of eigenvectors of $G$.

**Theorem :** A stationary iterative method $M\mathbf{x}^{k+1} = N\mathbf{x}^k + \mathbf{f}$ converges to the unique solution $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{f}$ for all initial guess $\mathbf{x}^0$ if and only if **all** the eigenvalues of $G = M^{-1}N$ have modulus strictly less than 1.

## Convergence rate

**Definition :** The **spectral radius** $\rho(G)$ of a matrix $G$ is defined as the maximum modulus of all the eigenvalues, i.e.,

$$\rho(G) = \max_{1 \leq i \leq n} |\lambda_i|,$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $G$.

Let $\rho(G) < 1$, so that the stationary method converges. If $G$ is diagonalizable with eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, then we have

$$\mathbf{e}^0 = \sum_{i=1}^{n} c_i \mathbf{v}_i \implies \mathbf{e}^k = G^k \mathbf{e}^0 = \sum_{i=1}^{n} c_i \lambda_i^k \mathbf{v}_i$$

$$\|\mathbf{e}^k\| \leq \sum_{i=1}^{n} |c_i| \, |\lambda_i|^k \|\mathbf{v}_i\| \leq C(\rho(G))^k$$

Therefore, the error norm is multiplied by roughly $\rho(G)$ at each iteration.

$$\boxed{\|\mathbf{e}^k\| \leq C(\rho(G))^k}$$

- $\rho(G)$ gives the contraction factor of the method :
  - $\rho(G) \ll 1 \implies$ fast convergence
  - $\rho(G) \approx 1 \implies$ slow convergence
- The "best" method is when $\rho(G) = 0$, which happens when $G = 0$ :

$$G = I - M^{-1}A = 0 \iff M^{-1}A = I \iff M = A.$$

  But solving with $M$ would be just as expensive as solving with $A$ !
- Designing a good method involves a tradeoff :
  - $M$ is a "good" approximation of $A$ (in some sense)
  - Systems involving $M$ are cheap(er) to solve than those involving $A$

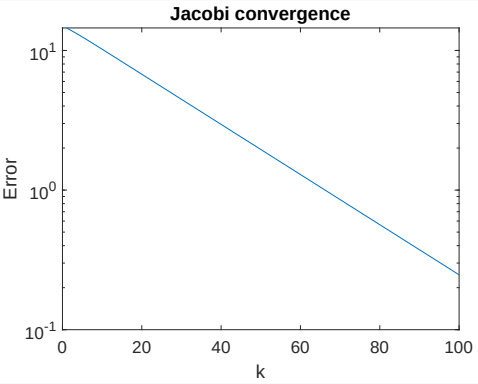
## Example : Jacobi and Gauss-Seidel

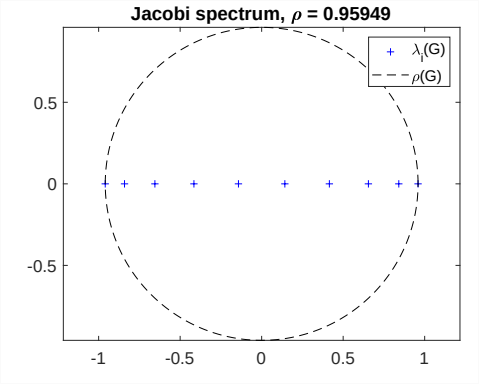
Consider the $10 \times 10$ system

$$\begin{bmatrix} a & -1 & & & \\ -1 & a & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & a & -1 \\ & & & -1 & a \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \end{bmatrix}$$

for $a = 2$ and $a = 4$.

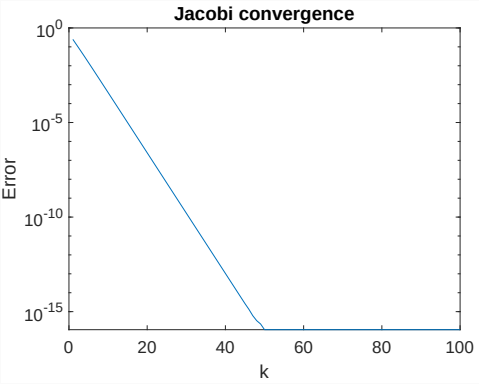
## Example : Jacobi, $a = 2$

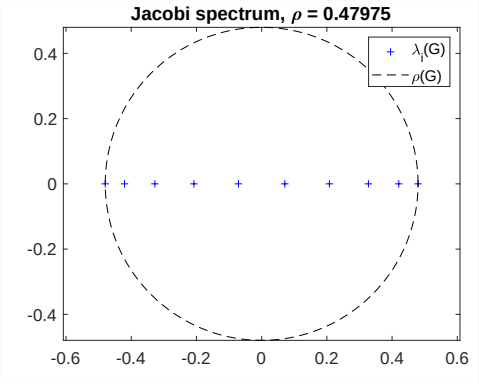



A tenfold reduction in error requires $\rho^k = 0.1 \implies k = 55.68$ iterations !

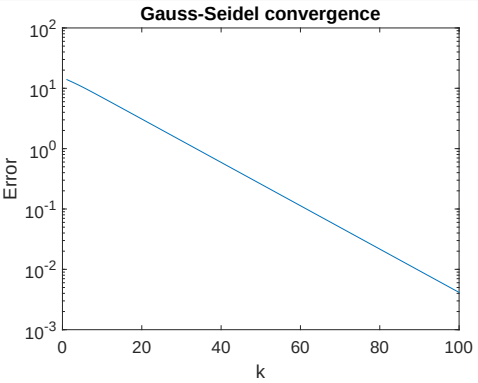
## Example : Jacobi, $a = 4$

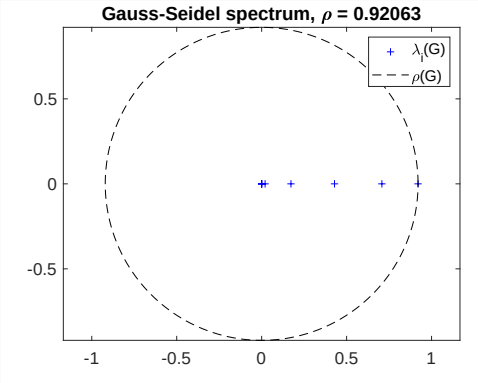



A tenfold reduction in error requires $\rho^k = 0.1 \implies k = 3.13$ iterations !

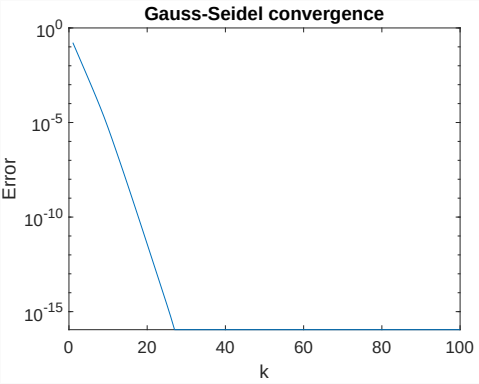
## Example : Gauss-Seidel, $a = 2$

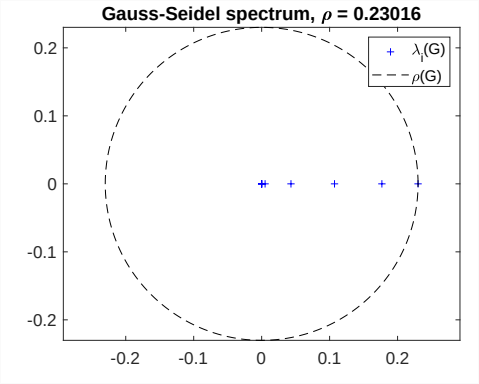



A tenfold reduction in error requires $\rho^k = 0.1 \implies k = 27.84$ iterations !

## Example : Gauss-Seidel, $a = 4$




A tenfold reduction in error requires $\rho^k = 0.1 \implies k = 1.57$ iterations !

## Classical convergence theorems

**Theorem :** Let $A$ be strictly diagonally dominant, i.e., the diagonal element in each row is greater than the sum of the off-diagonal entries (in absolute value) in that row. Then the Jacobi and Gauss-Seidel methods, applied to $A\mathbf{x} = \mathbf{f}$, both converge for all initial guess.

**Theorem :** Let $A$ be symmetric positive definite. Then the Gauss-Seidel method, applied to $A\mathbf{x} = \mathbf{f}$, converges for all initial guess.

**Note :** for the matrices in our example, it can be shown that the eigenvalues of the Gauss-Seidel matrix are the **squares** of the eigenvalues of the Jacobi matrix. Thus, Gauss-Seidel converges twice as fast as Jacobi in those examples.

## Block methods

Suppose we have a coupled multiphysics problem :

$$\begin{cases} F(\mathbf{u}, \mathbf{v}) = 0, \\ G(\mathbf{u}, \mathbf{v}) = 0. \end{cases}$$

If we solve this problem by Newton's method, we need to solve linear systems of the form

$$\underbrace{\begin{bmatrix} F_{\mathbf{u}}(\mathbf{u}^j, \mathbf{v}^j) & F_{\mathbf{v}}(\mathbf{u}^j, \mathbf{v}^j) \\ G_{\mathbf{u}}(\mathbf{u}^j, \mathbf{v}^j) & G_{\mathbf{v}}(\mathbf{u}^j, \mathbf{v}^j) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \mathbf{u}^{j+1} - \mathbf{u}^j \\ \mathbf{v}^{j+1} - \mathbf{v}^j \end{bmatrix}}_{\mathbf{x}} = \underbrace{- \begin{bmatrix} F(\mathbf{u}^j, \mathbf{v}^j) \\ G(\mathbf{u}^j, \mathbf{v}^j) \end{bmatrix}}_{\mathbf{f}}.$$

Block Jacobi is defined as $M\mathbf{x}^{k+1} = N\mathbf{x}^k + \mathbf{f}$, where M contains only a single "physics" :

$$M := \begin{bmatrix} F_{\mathbf{u}}(\mathbf{u}^j, \mathbf{v}^j) & 0 \\ 0 & G_{\mathbf{v}}(\mathbf{u}^j, \mathbf{v}^j) \end{bmatrix}.$$

In other words, **within each Newton step** $j$, we solve for $k = 0, 1, 2, \ldots$

$$F_{\mathbf{u}}(\mathbf{u}^j, \mathbf{v}^j)(\mathbf{u}^{j+1,k+1} - \mathbf{u}^j) = -F(\mathbf{u}^j, \mathbf{v}^j) - F_{\mathbf{v}}(\mathbf{u}^j, \mathbf{v}^j)(\mathbf{v}^{j+1,k} - \mathbf{v}^j),$$
$$G_{\mathbf{v}}(\mathbf{u}^j, \mathbf{v}^j)(\mathbf{v}^{j+1,k+1} - \mathbf{v}^j) = -G(\mathbf{u}^j, \mathbf{v}^j) - G_{\mathbf{u}}(\mathbf{u}^j, \mathbf{v}^j)(\mathbf{u}^{j+1,k} - \mathbf{u}^j).$$

One can also use block Gauss-Seidel to maintain some coupling by keeping one off-diagonal block :

$$M := \begin{bmatrix} F_{\mathbf{u}}(\mathbf{u}, \mathbf{v}) & 0 \\ G_{\mathbf{u}}(\mathbf{u}, \mathbf{v}) & G_{\mathbf{v}}(\mathbf{u}, \mathbf{v}) \end{bmatrix}.$$

Of course, one could also consider a **nonlinear** block Jacobi or Gauss-Seidel fixed-point iteration by solving directly the nonlinear problems

$$F(\mathbf{u}^{k+1}, \mathbf{v}^k) = 0,$$
$$G(\mathbf{u}^k, \mathbf{v}^{k+1}) = 0, \qquad k = 0, 1, 2, \ldots$$

## Example : CPR method (Wallace et al.,1985)

UNIVERSITÉ LAVAL

To solve the Jacobian systems arising from two-phase flow equations :

$$\text{Water :} \qquad \frac{\partial}{\partial t}(\phi \rho_w S) = \nabla \cdot (K(\mathbf{x}) \rho_w \lambda_w(S) \nabla p) + q_w,$$

$$\text{Oil :} \qquad \frac{\partial}{\partial t}(\phi \rho_o (1 - S)) = \nabla \cdot (K(\mathbf{x}) \rho_o \lambda_o(S) \nabla p) + q_o,$$

1. Add the two equations to obtain the discrete pressure equation

$$-\nabla \cdot (K_T(\mathbf{x}, S) \nabla p) = \tilde{q},$$

2. Use e.g. algebraic multigrid to solve the pressure equation.
3. Compute the new velocity field $\mathbf{v}_T$.
4. Solve local, $2 \times 2$ systems to update the saturation $S$, in decreasing order of pressure, or using an incomplete LU factorization.
5. Iterate to convergence.

## Example : cooling by fluids

| | | |
|---|---|---|
| Momentum : | $\nabla \cdot (2\eta \nabla^s \mathbf{v}) - \rho_f (\mathbf{v} \cdot \nabla)\mathbf{v} - \nabla p = 0$ | in $\Omega_f$, |
| Mass : | $\nabla \cdot (\rho_f \mathbf{v}) = 0$ | in $\Omega_f$, |
| Heat : | $-\nabla \cdot (K_f \nabla T) + \nabla \cdot (\mathbf{v}\, \rho_f C_f T) = 0$ | in $\Omega_f$, |
| Heat : | $-\nabla \cdot (K_s \nabla T) = q$ | in $\Omega_s$, |
| Coupling : | $K_s \nabla T \cdot \mathbf{n}_s + K_f \nabla T \cdot \mathbf{n}_f = 0$ | on $\Gamma$. |

Apply block Jacobi/Gauss-Seidel to sub-problems :

- Solve Navier-Stokes for $(\mathbf{v}, p)$ at fixed temperature, and heat equations at fixed $\mathbf{v}$
- Further split Navier-Stokes into a momentum equation and a pressure equation
- Split the heat equation into two subproblems based on the material

Performance depends on "coupling strength", which depends on $\rho$, $\mathbf{v}$, ...

## Implementation

- In the exercise session on Tuesday, you will simulate the cooling problem in MEF++, the finite element software developed by GIREF at Université Laval.
- MEF++ uses PETSc[1] to handle solution to linear problems arising from Newton's method
- PETSc offers block preconditioners under the `FieldSplit` option. Preconditioners available :
  - Additive Schwarz (Block Jacobi + overlap)
  - Multiplicative Schwarz (Block Gauss-Seidel + overlap)
  - Symmetric Multiplicative Schwarz (Forward + backward Gauss-Seidel)
  - Schur complement preconditioning (Eliminate (1,1)block exactly)
- Try different configurations to see what works best for your problem !

[1] `https://www.mcs.anl.gov/petsc/`