Sistemas Lineares
Métodos Iterativos Não Estacionários

Lucia Catabriga

Laboratório de Computação de Alto Desempenho (LCAD)
Departamento de Informática
Universidade Federal do Espírito Santo - UFES, Vitória, ES, Brasil

- Introdução
- Método dos Gradientes
- Método dos Gradientes Conjugados
- Método do Resíduo Mínimo Generalizado (GMRES)
- Método das Direções Conjugadas à Esquerda (LCD)
- Precondicionamento

- **Características:**
  - São método diretos na sua essência, mas na prática são usados como métodos iterativos. Portanto, a solução aproximada depende de uma tolerância pré-fixada.
  - A matriz dos coeficientes A e o vetor dos termos independentes b não são alterados durante o processo iterativo.
  - Dependem de critérios de convergência relacionados a matriz dos coeficientes A.

- **Características:**
  - São método diretos na sua essência, mas na prática são usados como métodos iterativos. Portanto, a solução aproximada depende de uma tolerância pré-fixada.
  - A matriz dos coeficientes A e o vetor dos termos independentes b não são alterados durante o processo iterativo.
  - Dependem de critérios de convergência relacionados a matriz dos coeficientes A.
- **Objetivo:**
  - Transformar o sistema $Ax = b$ em um problema de minimização do resíduo em cada iteração k
    $\|r_k\|_* = min\|b - Ax_k\|_*$

- **Características:**
  - São método diretos na sua essência, mas na prática são usados como métodos iterativos. Portanto, a solução aproximada depende de uma tolerância pré-fixada.
  - A matriz dos coeficientes A e o vetor dos termos independentes b não são alterados durante o processo iterativo.
  - Dependem de critérios de convergência relacionados a matriz dos coeficientes A.
- **Objetivo:**
  - Transformar o sistema $Ax = b$ em um problema de minimização do resíduo em cada iteração k
    $\|r_k\|_* = min\|b - Ax_k\|_*$
- **Complexidade**: $O(n^2)$ por iteração

- **Características:**
  - São método diretos na sua essência, mas na prática são usados como métodos iterativos. Portanto, a solução aproximada depende de uma tolerância pré-fixada.
  - A matriz dos coeficientes A e o vetor dos termos independentes b não são alterados durante o processo iterativo.
  - Dependem de critérios de convergência relacionados a matriz dos coeficientes A.
- **Objetivo:**
  - Transformar o sistema $Ax = b$ em um problema de minimização do resíduo em cada iteração k
    $\|r_k\|_* = min\|b - Ax_k\|_*$
- **Complexidade**: $O(n^2)$ por iteração
- Conhecidos como **Métodos baseados nos espaços vetoriais de Krylov**
- Os Métodos baseados nos espaços de Krylov representam a classe de métodos mais usadas em Dinâmica dos Fluidos Computacional.

Ideias Gerais dos Métodos Iterativos Não Estacionários

- Uma aproximação na iteração k é caracterizada por:
  - $x_k = x_0 + z$ é uma solução aproximada de $Ax = b$, onde:
  - $z \in K_k = span[r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0]$
  - $z = \sum_{i=1}^{k} y_i u_i$
  - $r_0 = b - Ax_0$

Ideias Gerais dos Métodos Iterativos Não Estacionários

- Uma aproximação na iteração k é caracterizada por:
  - $x_k = x_0 + z$ é uma solução aproximada de $Ax = b$, onde:
  - $z \in K_k = span[r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0]$
  - $z = \sum_{i=1}^{k} y_i u_i$
  - $r_0 = b - Ax_0$
- Principais operações:
  - Produto Matriz Vetor: $p = Av$ (matvec)
  - Produto Esclar: $r = p^T v$ (dot)
  - Adição de uma constante (a) vezes um vetor com outro vetor: $z = z + ax$ (saxpy)

Método dos Gradientes

Dado o sistema $Ax = b$

- A é simétrica ($a_{ij} = a_{ji}$) e definida positiva ($\forall v \neq 0 \in \Re^n, v^t Av > 0$, se $v = 0, v^t Av = 0$)
- Seja uma aproximação inicial x_0, o objetivo principal é reduzir o resíduo $r_k = b - Ax_k$ a partir do resíduo r_0.
- Para que o resíduo diminua, tomamos uma direção v e corrigimos x_0 nessa direção, ou seja, $x_1 = x_0 + \lambda v$ de forma que $r_1 = b - Ax_1 < r_0$
- Construimos uma sequência x_k convergente para a solução de $Ax = b$, tal que, $r_{k+1} < r_k$.

Método dos Gradientes

Dado o sistema $Ax = b$

- A é simétrica ($a_{ij} = a_{ji}$) e definida positiva ($\forall v \neq 0 \in \Re^n, v^t Av > 0$, se $v = 0, v^t Av = 0$)
- Seja uma aproximação inicial x_0, o objetivo principal é reduzir o resíduo $r_k = b - Ax_k$ a partir do resíduo r_0.
- Para que o resíduo diminua, tomamos uma direção v e corrigimos x_0 nessa direção, ou seja, $x_1 = x_0 + \lambda v$ de forma que $r_1 = b - Ax_1 < r_0$
- Construimos uma sequência x_k convergente para a solução de $Ax = b$, tal que, $r_{k+1} < r_k$.

Seja a Função Erro $f(x) = \frac{1}{2} r^t A^{-1} r$:

- $\forall r, f \geq 0$ e se x é solução exata $f(x) = 0$
- $f(x) = \frac{1}{2}(b - Ax)^t A^{-1}(b - Ax)$
- $f(x) = \frac{1}{2} x^t Ax - b^t x + c$, onde $c = \frac{1}{2} bA^{-1} b$

$$\begin{aligned} x &= min_{x \in \Re^n} f(x) \\ Ax &= b \\ f(x) &= \frac{1}{2} x^t A x - b^t x + c \end{aligned}$$

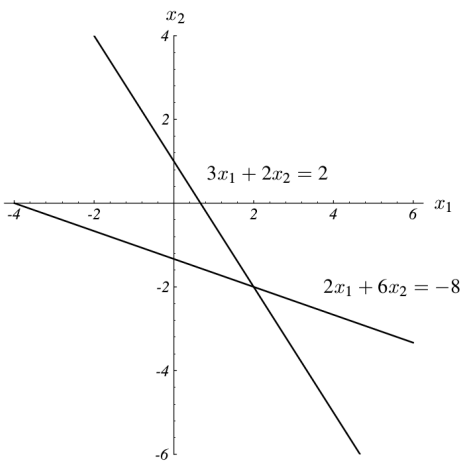
Encontrar a solução de

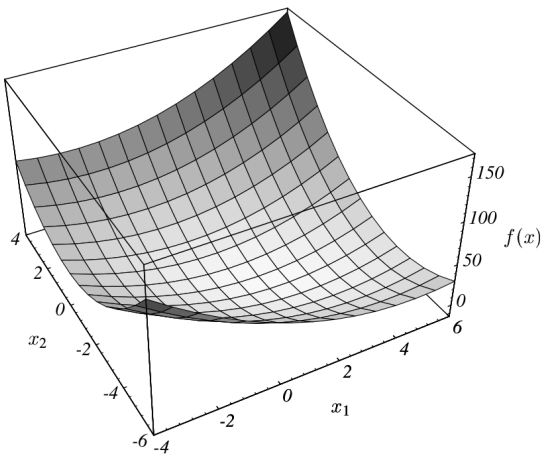
$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ -8 \end{bmatrix}, \text{ sendo a sol. exata igual a } \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

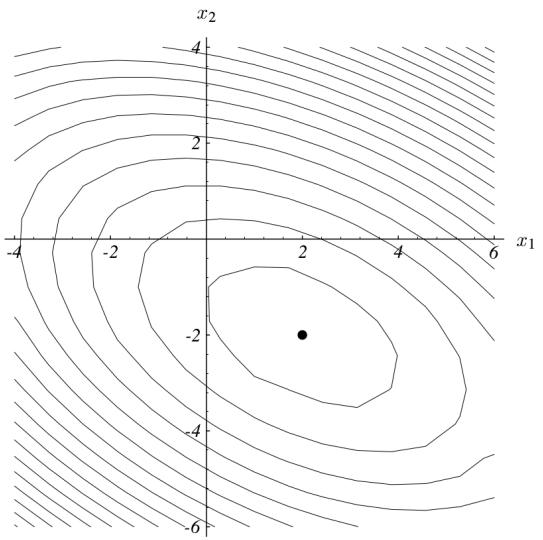
$$\begin{aligned} x &= min_{x \in \Re^n} f(x) \\ Ax &= b \\ f(x) &= \frac{1}{2} x^t A x - b^t x + c \end{aligned}$$

Encontrar a solução de

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ -8 \end{bmatrix}, \text{ sendo a sol. exata igual a } \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$







Mínimo de $f(x)$

Afirmação: O ponto de mínimo de $f(x)$ é a solução de $Ax = b$

- O ponto mínimo de $f(x)$ é um ponto crítico, ou seja, $f'(x) = \nabla f(x) = 0$

$$f(x) = \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}x_i x_j - \sum_{i=1}^{n} b_i x_i + c$$

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - b1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - b2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n - bn \end{bmatrix} = Ax - b$$

Mínimo de $f(x)$

Afirmação: O ponto de mínimo de $f(x)$ é a solução de $Ax = b$

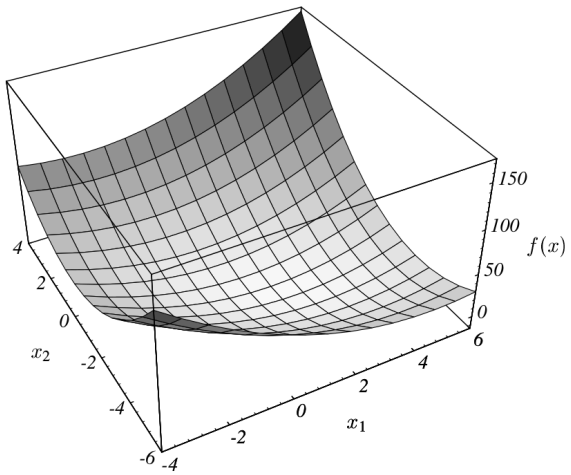
- O ponto mínimo de $f(x)$ é um ponto crítico, ou seja, $f'(x) = \nabla f(x) = 0$

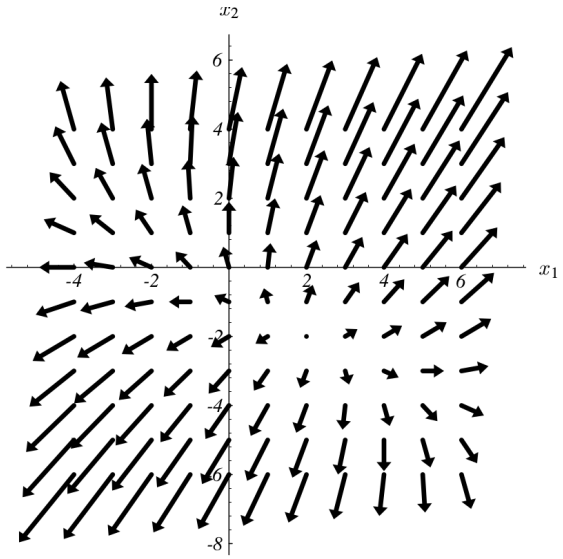
$$f(x) = \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}x_i x_j - \sum_{i=1}^{n} b_i x_i + c$$

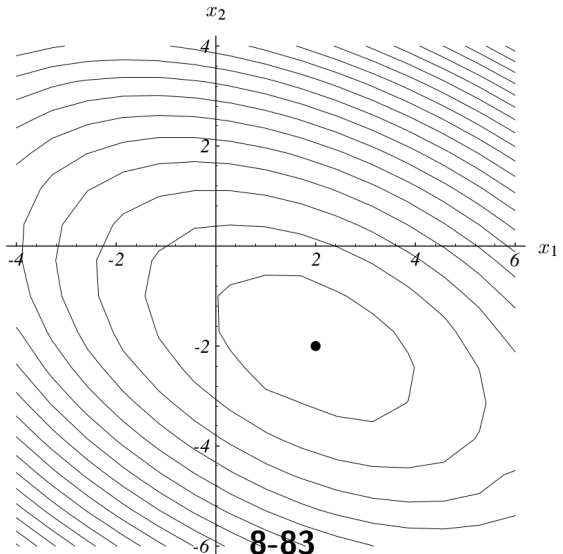
$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - b1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - b2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n - bn \end{bmatrix} = Ax - b$$

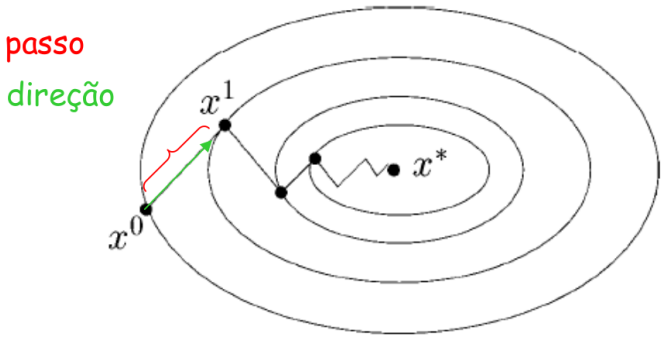
- $\nabla f(x) = 0 \Rightarrow Ax - b = 0$
- x é ponto de mínimo, pois $f(x)$ é sempre postiva, uma vez que A é definida positiva.

Direção de crescimento e decrescimento de $f(x)$:
$\nabla f(x)$ representa a direção de maior crescimento de $f(x)$:







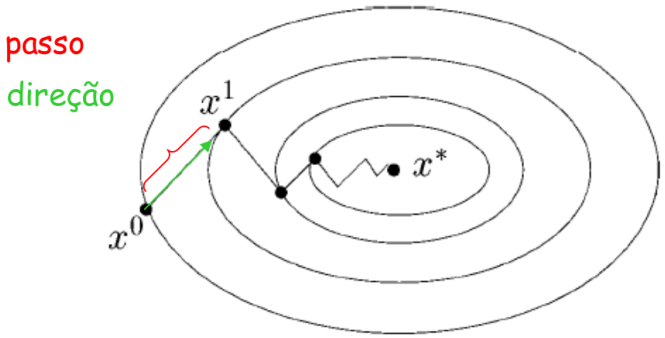


Processo iterativo:

$$x_1 = x_0 + \lambda_1 r_0 \text{ mesma direção de } r_0$$

$$\lambda_1 : \text{ minimiza } f(x) \text{ quando } \frac{\partial f(x_1)}{\partial \lambda_1} = 0$$

Mas



Processo iterativo:

$$x_1 = x_0 + \lambda_1 r_0 \text{ mesma direção de } r_0$$

$$\lambda_1 : \text{ minimiza } f(x) \text{ quando } \frac{\partial f(x_1)}{\partial \lambda_1} = 0$$

Mas

$$\frac{\partial f(x_1)}{\partial \lambda_1} = \left(f'(x_1)\right)^T \frac{\partial x_1}{\partial \lambda_1} = -r_1^T r_0 = 0$$

Portanto, o passo λ_1 ideal é aquele tal que r_1 é ortogonal a r_0.
Assim $r_{i+1}^T r_i = 0 \quad \forall i$

Processo iterativo:

- Cáculo de λ_i:

$$\begin{aligned} r_{i+1}^T r_i &= 0 \\ (b - Ax_{i+1})^t r_i &= 0 \\ (b - A(x_i + \lambda_i r_i))^T r_i &= 0 \\ r_i^T r_i - \lambda_i r_i^T A r_i &= 0 \\ \lambda_i &= \frac{r_i^T r_i}{r_i^T A r_i} \end{aligned}$$

Processo iterativo:

- Cáculo de λ_i:

$$\begin{aligned} r_{i+1}^T r_i &= 0 \\ (b - Ax_{i+1})^t r_i &= 0 \\ (b - A(x_i + \lambda_i r_i))^T r_i &= 0 \\ r_i^T r_i - \lambda_i r_i^T A r_i &= 0 \\ \lambda_i &= \frac{r_i^T r_i}{r_i^T A r_i} \end{aligned}$$

- Atualização do resíduo:

$$\begin{aligned} x_{i+1} &= x_i + \lambda_i r_i \\ b - Ax_{i+1} &= b - A(x_i + \lambda_i r_i) \\ r_{i+1} &= r_i - \lambda_i A r_i \end{aligned}$$

Algoritmo do Método dos Gradientes:

Dados $x_0 = 0$, A, b, N_{max}, tol

$r_0 = b$, $\delta = \delta_0 = r_0^t r_0 = \|r_0\|_2^2$

$i = 0$

while $\delta > tol^2 \delta_0$ **and** $i \leq N_{max}$ **do**

$v = Ar_i$

$\lambda_i = \frac{\delta}{r_i^t v}$

$x_{i+1} = x_i + \lambda_i r_i$

$r_{i+1} = r_i - \lambda_i v$

$\delta = r_{i+1}^t r_{i+1} = \|r_{i+1}\|_2^2$

$i = i + 1$

end while

A matriz triadiagonal onde $a_{ii} = 10, a_{i,i-1} = a_{i,i+1} = 1$, b tal que $x = [1, 1, \cdots, 1]$, $tol = 5 \times 10^{-6}$

$$x_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad x_1 = \begin{bmatrix} 0.93024 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 0.93024 \end{bmatrix} \quad x_2 = \begin{bmatrix} 0.99776 \\ 1.00560 \\ 0.99724 \\ 0.99724 \\ 0.99724 \\ 0.99724 \\ 0.99724 \\ 0.99724 \\ 1.00560 \\ 0.99776 \end{bmatrix} \quad \cdots x_5 = \begin{bmatrix} 0.99999 \\ 1.00002 \\ 0.99999 \\ 1.00002 \\ 1.00001 \\ 1.00001 \\ 1.00002 \\ 0.99999 \\ 1.00002 \\ 0.99999 \end{bmatrix} \quad x_6 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- $\|x_6\|_\infty = 1.000002$
- $\|r_6\|_\infty = 7.5765 \times 10^{-05}$

Propriedades:

- Dois vetores x e y são conjugados se $x^T Ay = y^T Ax = 0 \Rightarrow x$ e y são ditos A-ortogonais e linearmente independentes.
- No processo iterativo do método dos gradientes é possível que uma direção considerada na iteração i já tenha sido usada em iterações anteriores.
- O método dos Gradientes Conjugados sugere que, dada uma aproximação inicial x_0 para o sistema $Ax = b$, seja considerado um conjunto de n vetores (direções) conjugados $\{d_0, d_1, \cdots, d_{n-1}\}$ tal que $x_n = x_0 + \sum_{i=0}^{n-1} \lambda_i d_i$ é a solução exata de $Ax = b$ a menos de erros de arredondamento.

- Propriedades:
  - O resíduo no passo i é ortogonal ao resíduo no passo anterior $\Rightarrow r_i^T r_{i-1} = 0$
  - O resíduo no passo i é ortogonal à direção conjugada no passo $i \Rightarrow r_i^T d_i = 0$
  - O resíduo no passo i é ortogonal à direção conjugada do passo anterior $\Rightarrow r_i^T d_{i-1} = 0$
  - A direção conjugada no passo i é ortogonal com relação a A à direção conjugada no passo anterior $\Rightarrow d_i^T A d_{i-1} = 0$

- Propriedades:
  - O resíduo no passo i é ortogonal ao resíduo no passo anterior $\Rightarrow r_i^T r_{i-1} = 0$
  - O resíduo no passo i é ortogonal à direção conjugada no passo $i \Rightarrow r_i^T d_i = 0$
  - O resíduo no passo i é ortogonal à direção conjugada do passo anterior $\Rightarrow r_i^T d_{i-1} = 0$
  - A direção conjugada no passo i é ortogonal com relação a A à direção conjugada no passo anterior $\Rightarrow d_i^T A d_{i-1} = 0$

- Dado x_0, seja $d_0 = r_0 = b - Ax_0$
- $x_{i+1} = x_i + \lambda_i d_i$
- $r_{i+1} = r_i - \lambda_i A d_i$
- Cálculo de λ_i?
- Cálculo de d_{i+1}?

- As direções conjugadas satisfazem a relação:

$$d_{i+1} = r_{i+1} + \beta_{i+1} d_i$$

- Cálculo de λ_i

$$\begin{aligned} r_{i+1}^T r_i &= 0 \\ (r_i - \lambda_i A d_i)^t r_i &= 0 \Rightarrow r_i^T r_i - \lambda d_i^T A r_i = 0 \\ \lambda &= \frac{r_i^T r_i}{d_i^T A r_i} \end{aligned}$$

- As direções conjugadas satisfazem a relação:

$$d_{i+1} = r_{i+1} + \beta_{i+1} d_i$$

- Cálculo de λ_i

$$\begin{aligned}
r_{i+1}^T r_i &= 0 \\
(r_i - \lambda_i A d_i)^t r_i &= 0 \Rightarrow r_i^T r_i - \lambda d_i^T A r_i = 0 \\
\lambda &= \frac{r_i^T r_i}{d_i^T A r_i} \\
\text{Mas ...} & \\
d_i &= r_i + \beta_i d_{i-1} \Rightarrow r_i = d_i - \beta_i d_{i-1} \\
d_i^T A r_i &= d_i^T A d_i - \beta_i d_i^T A d_{i-1} \\
d_i^T A r_i &= d_i^T A d_i \quad (\text{Atenção! } r_i \neq d_i) \\
\lambda_i &= \frac{r_i^T r_i}{d_i^T A d_i} \quad \text{ou} \quad \lambda_i = \frac{\|r_i\|_2^2}{d_i^T A d_i}
\end{aligned}$$

- Cálculo de β_i:

$$\begin{aligned} d_{i+1}^T A d_i &= 0 \\ (r_{i+1} + \beta_{i+1} d_i) A d_i &= 0 \\ \beta_{i+1} &= -\frac{r_{i+1}^T A d_i}{d_i^T A d_i} \end{aligned}$$

Mas,

- Cálculo de β_i:

$$\begin{aligned} d_{i+1}^T A d_i &= 0 \\ (r_{i+1} + \beta_{i+1} d_i) A d_i &= 0 \\ \beta_{i+1} &= -\frac{r_{i+1}^T A d_i}{d_i^T A d_i} \end{aligned}$$

Mas,

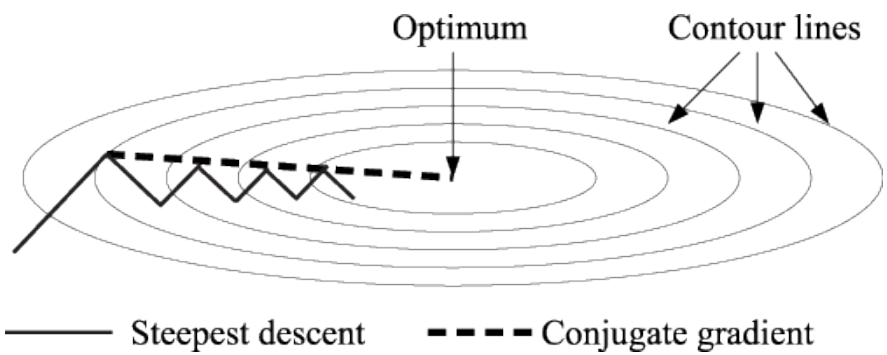
$$Ad_i = -\frac{1}{\lambda_i}(r_{i+1} - r_i) \quad \Rightarrow \quad \beta_{i+1} = -\frac{r_{i+1}^T (r_{i+1} - r_i)}{d_i^T A d_i} \left(-\frac{1}{\lambda_i}\right)$$

$$\beta_{i+1} = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} \quad \text{ou} \quad \beta_{i+1} = \frac{\|r_{i+1}\|_2^2}{\|r_i\|_2^2}$$

Algoritmo do Método dos Gradientes Conjugados

Dados $x_0 = 0$, A, b

$d_0 = r_0 = b$
for $i = 0, 1, \cdots, n-1$ **do**
  $v = Ad_i$
  $\lambda_i = \frac{\|r_i\|_2^2}{d_i^t v}$
  $x_{i+1} = x_i + \lambda_i d_i$
  $r_{i+1} = r_i - \lambda_i v$
  $\beta_{i+1} = \frac{\|r_{i+1}\|_2^2}{\|r_i\|_2^2}$
  $d_{i+1} = r_{i+1} + \beta_{i+1} d_i$
end for



Método Direto!!!

A matriz triadiagonal onde $a_{ii} = 10, a_{i,i-1} = a_{i,i+1} = 1$, b tal que $x = [1, 1, \cdots, 1]$

$$x_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad x_1 = \begin{bmatrix} 0.93024 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 1.01480 \\ 0.93024 \end{bmatrix} \quad x_2 = \begin{bmatrix} 0.99872 \\ 1.00656 \\ 0.99819 \\ 0.99819 \\ 0.99819 \\ 0.99819 \\ 0.99819 \\ 0.99819 \\ 1.00656 \\ 0.99872 \end{bmatrix} \quad x_4 = \begin{bmatrix} 1.00000 \\ 1.00000 \\ 0.99999 \\ 1.00005 \\ 0.99996 \\ 0.99996 \\ 1.00005 \\ 0.99999 \\ 1.00000 \\ 1.00000 \end{bmatrix} \quad x_5, \cdots, x_{10} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\|r_1\| = 1.0673 \quad \|r_2\| = 0.10069 \quad \|r_3\| = 0.0095837 \quad \|r_4\| = 8.2349 \times 10^{-04}$$

$$\|r_5\| = 9.1184 \times 10^{-19} \quad \ldots \quad \|r_{10}\| = 4.2772 \times 10^{-24}$$

$$A = \begin{bmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & & \ddots & & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ \vdots \\ 6 \\ 5 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

$$n = 100 \Rightarrow \|x\|_\infty = 1.0, \|r\|_\infty = 1.7764 \times 10^{-15}$$
$$n = 200 \Rightarrow \|x\|_\infty = 1.0, \|r\|_\infty = 1.7764 \times 10^{-15}$$
$$n = 250 \Rightarrow \|x\|_\infty = 1.0, \|r\|_\infty = 2.2204 \times 10^{-15}$$
$$n = 260 \Rightarrow \|x\|_\infty = 1.0, \|r\|_\infty = 2.6645 \times 10^{-15}$$
$$n = 261 \Rightarrow \|x\|_\infty = 1.0, \|r\|_\infty = 1.7764 \times 10^{-15} \text{ (divisão por zero)}$$
$$n = 300 \Rightarrow \|x\|_\infty = 0.0, \|r\|_\infty = 0.0 \text{ (divisão por zero)}$$

Algoritmo do Método Iterativo dos Gradientes Conjugado

Dados $x_0 = 0$, A, b, N_{max}, tol_ϵ

$d_0 = r_0 = b$
$\delta_{new} = \|r_0\|_2^2$
$\delta_0 = \delta_{new}$
$i = 0$
while $\delta_{new} > {tol_\epsilon}^2 \delta_0$ **and** $i \leq N_{max}$ **do**
  $v_i = Ad_i$
  $\lambda_i = \frac{\delta_{new}}{d_i^t v_i}$
  $x_{i+1} = x_i + \lambda_i d_i$
  $r_{i+1} = r_i - \lambda_i v$
  $\delta_{old} = \delta_{new}$
  $\delta_{new} = \|r_{i+1}\|_2^2$
  $\beta_{i+1} = \frac{\delta_{new}}{\delta_{old}}$
  $d_{i+1} = r_{i+1} + \beta_{i+1} d_i$
  $i = i + 1$
end while

Complexidade e Armazenamento do Método dos Gradientes Conjugados:

- i-ésima iteração:
  - Número de operações:
    - 1 matvec
    - 3 saxpy
    - 2 dot
  - Complexidade:
    - $\theta(n^2)$
  - Armazenamento:
    - 3 vetores
    - 1 matriz A

Exemplos - CG (iterativo)

A é diagonal dominante

$$A = \begin{bmatrix} 4 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & & \ddots & & & & \\ & & & 1 & 4 & 1 & \\ & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ \vdots \\ 6 \\ 5 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

$$n = 300 \quad \Rightarrow \quad tol = 1 \times 10^{-6}, \quad 10 \text{ iterações } \|r\|_\infty = 9.7417 \times 10^{-6}$$
$$n = 300 \quad \Rightarrow \quad tol = 1 \times 10^{-10}, \quad 17 \text{ iterações } \|r\|_\infty = 9.6544 \times 10^{-10}$$
$$n = 600 \quad \Rightarrow \quad tol = 1 \times 10^{-6}, \quad 10 \text{ iterações } \|r\|_\infty = 9.8035 \times 10^{-6}$$
$$n = 600 \quad \Rightarrow \quad tol = 1 \times 10^{-10}, \quad 17 \text{ iterações } \|r\|_\infty = 9.7204 \times 10^{-10}$$

Exemplos - CG (iterativo)

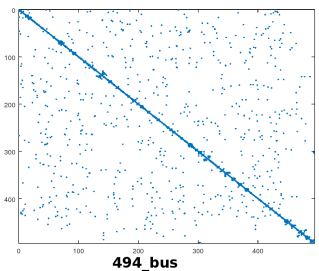
A não é diagonal dominante

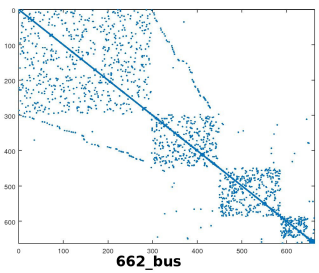
$$A = \begin{bmatrix} 2 & 1 & & & & & \\ 1 & 2 & 1 & & & & \\ & & \ddots & & & & \\ & & & 1 & 2 & 1 & \\ & & & & 1 & 2 & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ \vdots \\ 4 \\ 3 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

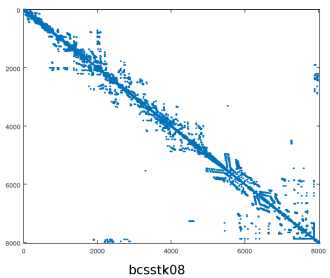
$$n = 300 \quad \Rightarrow \quad tol = 1 \times 10^{-6}, \quad 150 \text{ iterações } \|r\|_\infty = 5.7176 \times 10^{-14}$$
$$n = 300 \quad \Rightarrow \quad tol = 1 \times 10^{-10}, \quad 150 \text{ iterações } \|r\|_\infty = 5.7176 \times 10^{-14}$$
$$n = 600 \quad \Rightarrow \quad tol = 1 \times 10^{-6}, \quad 300 \text{ iterações } \|r\|_\infty = 1.2124 \times 10^{-13}$$
$$n = 600 \quad \Rightarrow \quad tol = 1 \times 10^{-10}, \quad 300 \text{ iterações } \|r\|_\infty = 1.2124 \times 10^{-13}$$

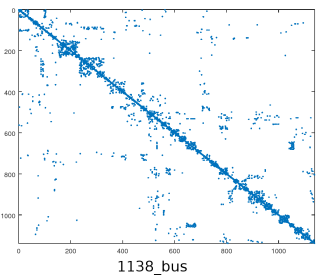
Exemplos de Matrizes Esparsas

A matriz Esparsa do Repositório CISE e b tal que $x = [1, 1, \cdots, 1]^T$, $tol = 10^{-7}$









A	n	tempo (seg)	N_{iter}	$\|x\|_\infty$	cond(A)
494_bus	494	2.714×10^{-2}	857	1.000925	$2.4154 \times 10^{+6}$
662_bus	662	1.690×10^{-2}	441	1.000094	$7.9413 \times 10^{+5}$
bcsstk08	1074	8.121×10^{-2}	1131	1.097432	$2.5988 \times 10^{+7}$
1138_bus	1138	9.406×10^{-2}	1946	1.000012	$8.5726 \times 10^{+6}$

Gradiente Conjugado no Octave

`[x,flag,relres,iter,resvec] = pcg(A,b,tol,maxit)`

- `A`: Matriz dos coeficientes simétrica definida positiva[1];
- `b`: Vetor dos termos independentes;
- `tol`: Tolerância relativa;
- `maxit`: número máximo de iterações;
- `x`: vetor solução aproximada;
- `flag`: 0 - convergência atingida; 1 - número máximo de iterações atingido; 3 - estagnação do resíduo
- `relres`: valor final do resíduo relativo
- `iter`: número de iterações executadas
- `resvec`: vetor contendo o resíduo relativo em cada iteração

[1] *default*: armazenamento na estrutura CCR (Compressed Column Sparse)

A é a matriz de ondem n
$b = A * \texttt{ones(n)}$, $tol = 0.00001$, $maxit = 10000$

Matriz	n	flag	Número de Iterações	Tempo (seg)
bcsstk01	48	0	31	0.0035
bcsstk15	3948	0	4555	2.0486
Dubcova1	16129	0	83	0.0898
bcircuit	68902	0	9004	21.0656

Definições:

- $x_{i+1} = x_0 + z$ (Solução aproximada de $Ax = b$)
- $\mathcal{K}^m = span\{u_0, u_1, \cdots, u_{m-1}\}$ (Espaço de Busca)
- $\mathcal{L}^m = span\{w_0, w_1, \cdots, w_{m-1}\}$ (Espaço de Restrição)
- $U = [u_0, u_1, \cdots, u_{m-1}]$ (base de $\mathcal{K}^m$)

$$z \in \mathcal{K}^m \Rightarrow z = \sum_{j=0}^{m-1} y_j u_j = [u_0, u_1, \cdots, u_{m-1}] \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_{m-1} \end{bmatrix} \quad \text{para } y \in \mathbb{R}^m$$

- $x_{i+1} = x_0 + Uy$ para $y \in \mathcal{L}^m$

A escolha do vetor y define as restrições impostas para avançar no processo iterativo

Os métodos baseados nos espaços de Krylov se diferenciam pela escolha do espaço das restrições

Método do Resíduo Mínimo Generalizado (GMRES) - Ideias Gerais

- Proposto em 1986 por Yousef Saad[2]
- Descrição:

$$Ax = b \iff min_{x \in x_0 + \mathcal{K}^k} \|b - Ax\|_2$$

  - A solução aproximada $x_k = x_0 + z$
  - $z \in \mathcal{K}^k = span\{r_0, Ar_0, A^2 r_0, \cdots, A^{k-1} r_0\} = [u_1, u_2, \cdots, u_k] = U$
  - $z = \sum_{i=1}^{k} y_i u_i$ ou $z = Uy$ onde $y_i \in \mathbb{R}$ ou $y \in \mathbb{R}^k$
- Próximos Passos:
  - Montagem da base $[U]$
  - Cálculo dos coeficientes y_i

[2]Yousef Saad and Martin H Schultz. 1986. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. 7, 3 (July 1986), 856-869.

Montagem da Base Ortonormal de $\mathcal{K}^k$:

Processo de Gram-Schmidt

$[r_0, Ar_0, A^2 r_0, \cdots, A^{k-1} r_0] = [u_1, u_2, \cdots, u_k] = U_k$

$u_1 = \frac{r_0}{\|r_0\|_2}$
for $i = 1, \cdots, k$ **do**
  $\tilde{u}_{i+1} = A u_i$
  for $j = 1, \cdots, i$ **do**
    $\beta_{i+1,j} = \tilde{u}_{i+1}^T u_j$
    $\tilde{u}_{i+1} = \tilde{u}_{i+1} - \beta_{i+1,j} u_j$
  end for
  $u_{i+1} = \frac{\tilde{u}_{i+1}}{\|\tilde{u}_{i+1}\|_2}$
end for

O processo de Gram-Schmidt define a Matriz de Hessemberg superior de ordem $(k+1) \times k$

$$H_k = \begin{bmatrix} \beta_{21} & \beta_{31} & \cdots & \beta_{k1} & \beta_{k+1,1} \\ \|\tilde{u}_2\| & \beta_{32} & \cdots & \beta_{k2} & \beta_{k+1,2} \\ & \|\tilde{u}_3\| & \cdots & \beta_{k3} & \beta_{k+1,3} \\ & & \ddots & \vdots & \vdots \\ & & & \beta_{k,k-1} & \beta_{k+1,k-1} \\ & & & \|\tilde{u}_k\| & \beta_{k+1,k} \\ & & & & \|\tilde{u}_{k+1}\| \end{bmatrix}$$

Pode-se mostrar que:

$$AU_k = U_{k+1} H_k$$

O problema de Minimização em $\mathcal{K}^k$:

$$
\begin{aligned}
z &= \sum_{i=1}^{k} y_i u_i \text{ para } y \in \mathbb{R}^k \text{ e } z \in \mathcal{K}^k \\
e &= \{\|r_0\|, 0, \cdots, 0\}^T \quad (k+1) \text{ termos} \\
r_0 &= U_{k+1} e \quad \text{pois} \quad u_1 = \frac{r_0}{\|r_0\|} \text{ e } U_{k+1} \text{ é uma base de } \mathcal{K}^{k+1}
\end{aligned}
$$

$$
\begin{aligned}
\|b - Ax\| &= \|b - A(x_0 + z)\| = \|b - Ax_0 - A\sum_{i=1}^{k} y_i u_i\| = \|r_0 - AU_k y\| \\
&= \|U_{k+1} e - U_{k+1} H_k y\| = \|e - H_k y\|, \quad \text{pois} \quad \|U_{k+1}\| = 1
\end{aligned}
$$

$$min_{z \in \mathcal{K}^k} \|b - A(x_0 + z)\| = min_{y \in \mathbb{R}^k} \|e - H_k y\|$$

O problema de minimização em $\mathcal{K}^k$ foi transformado em um problema de minimização em $\mathbb{R}^k$

GMRES - algoritmo Inicial

Dados $x_0 = 0$, A, b, l_{max}, tol

$r = b$, $\rho = \|r\|_2$, $u_1 = r/\rho$
$\epsilon = tol\|b\|_2$, $k = 1$
while $\rho > \epsilon$ **and** $k \leq l_{max}$ **do**
  $\tilde{u}_{k+1} = Au_k$
  for $j = 1, \cdots, k$ **do**
    $h_{j,k} = \tilde{u}_{k+1}^T u_j$
    $\tilde{u}_{k+1} = \tilde{u}_{k+1} - h_{j,k} u_j$
  end for
  $h_{k+1,k} = \|\tilde{u}_{k+1}\|_2$
  $u_{k+1} = \frac{\tilde{u}_{k+1}}{\|\tilde{u}_{k+1}\|_2}$
  $e = (\|r\|, 0, \cdots, 0)^T$
  $min_{y \in \mathbb{R}^k} \|e - H_k y\|$
  $\rho = \|e - H_k y\|$
  $k = k + 1$
end while
$x_k = x_0 + U_k y$

GMRES - Sistema Linear auxiliar

$$min_{y \in \mathbb{R}^k} \|e - H_k y\| \Rightarrow H_k y = e$$

$$\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1,k-1} & h_{1,k} \\ \|\tilde{u}_2\| & h_{22} & \cdots & h_{2,k-1} & h_{2,k} \\ & \|\tilde{u}_3\| & \cdots & h_{3,k-1} & h_{3,k} \\ & & \ddots & \vdots & \vdots \\ & & & h_{k-1,k-1} & h_{k-1,k} \\ & & & \|\tilde{u}_k\| & h_{k,k} \\ & & & & \|\tilde{u}_{k+1}\| \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{k-1} \\ y_k \end{bmatrix} = \begin{bmatrix} \|r_0\| \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \bar{H}_k y = \bar{e} \Rightarrow \begin{bmatrix} \bar{h}_{11} & \bar{h}_{12} & \cdots & \bar{h}_{1k} \\ & \bar{h}_{22} & \cdots & \bar{h}_{2k} \\ & & \ddots & \vdots \\ & & & \bar{h}_{kk} \\ & & & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} \bar{e}_1 \\ \bar{e}_2 \\ \vdots \\ \bar{e}_k \\ \bar{e}_{k+1} \end{bmatrix}, \text{ sendo } \bar{e}_{k+1} = \|e - H_k y\|$$

$\bar{H}_k y = \bar{e}$ tem solução única quando $\bar{e}_{k+1} \approx 0$

Cálculo de $\bar{H}_k$ - Algoritmo QR - Rotação de Givens

$$\begin{aligned} \bar{H}_k &= G_k G_{k-1} \cdots G_2 G_1 H_k \\ \bar{e} &= G_k G_{k-1} \cdots G_2 G_1 e \end{aligned}$$

$$G_j = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & c_j & s_j & \cdots & 0 \\ 0 & \cdots & -s_j & c_j & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$H_k = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1,k-1} & h_{1,k} \\ h_{21} & h_{22} & \cdots & h_{2,k-1} & h_{2,k} \\ & h_{31} & \cdots & h_{3,k-1} & h_{3,k} \\ & & \ddots & \vdots & \vdots \\ & & & h_{k-1,k-1} & h_{k-1,k} \\ & & & h_{k,k-1} & h_{k,k} \\ & & & & h_{k+1,k} \end{bmatrix}$$

$$\begin{aligned} G_j &= G_j(c_j, s_j) \\ r_j &= \sqrt{h_{jj}^2 + h_{j+1,j}^2} \\ c_j &= \frac{h_{jj}}{r_j} \\ s_j &= \frac{h_{j+1,j}}{r_j} \end{aligned}$$

$$G_1 H_k = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} H_k$$

$$= \begin{bmatrix} \bar{h}_{11} & \bar{h}_{12} & \cdots & \bar{h}_{1k} \\ & \bar{h}_{22} & \cdots & \bar{h}_{2k} \\ & h_{32} & \cdots & h_{3k} \\ & & \ddots & \vdots \\ & & & h_{k+1,k} \end{bmatrix}$$

Dados $x = 0$, A, b, l_{max}, tol

$\epsilon = tol\|b\|_2$, $i = 1$

$u_i = b$, $e_i = \|u_i\|_2$, $u_i = r/e_i$, $\rho = e_i$

while $\rho > \epsilon$ **and** $i \leq l_{max}$ **do**

Ortogonalização de Gram-Schmidt

$\tilde{u}_{i+1} = Au_i$

for $j = 1, \cdots, i$ **do**

$h_{j,i} = \tilde{u}_{i+1}^T u_j$, $\tilde{u}_{i+1} = \tilde{u}_{i+1} - h_{j,i} u_i$

end for

$h_{i+1,i} = \|\tilde{u}_{i+1}\|_2$, $u_{i+1} = \frac{\tilde{u}_{i+1}}{h_{i+1,i}}$

Algoritmo QR

$\vdots$ (descrição ao lado)

$i = i + 1$

end while

$i = i - 1$

Solução do Sistema Auxíliar

for $j = i, \cdots, 1$ **do**

$y_i = \frac{e_j - \sum_{t=j+1}^{i} h_{jt} y_t}{h_{jj}}$

end for

$x = \sum_{j=1}^{i} y_j u_j$

Algoritmo QR

for $j = 1, \cdots, i-1$ **do**

$ji = h_{ji}$

$h_{ji} = c_j h_{ji} + s_j h_{j+1,i}$

$h_{j+1,i} = -s_j ji + c_j h_{j+1,i}$

end for

$r_i = \sqrt{h_{ii}^2 + h_{i+1,i}^2}$

$c_i = \frac{h_{ii}}{r_i}$

$s_i = \frac{h_{i+1,i}}{r_i}$

$h_{ii} = r_i$ $(h_{i+1,i} = 0)$

$e_{i+1} = -s_i e_i$

$e_i = c_i e_i$

$\rho = |e_{i+1}|$

Complexidade e Armazenamento:

- i-ésima iteração:
  - Número de operações:
    - 1 matvec
    - i saxpy
    - i dot
    - $\approx i^3$ fatoração QR
  - Complexidade:
    - $\theta(i \times n) + \theta(i^3) + \theta(\text{matvec})$, sendo que $\theta(\text{matvec}) \approx n^2$
    - $\Rightarrow$ pode ser maior que $\theta(n^2)$, se $i \approx n$.
  - Armazenamento:
    - i vetores
    - 1 matriz A
    - 1 matriz H_k de ordem $(i+1) \times i$

- Cada iteração do método GMRES é muito custosa $\Rightarrow$ método computacionalmente inviável.
- O processo de reinicialização da base torna o método computacionalmente competitivo.
- Supor tamanho máximo da base igual a 20 vetores:
  - Dados A, b, x_0

$$\begin{aligned} U_{20} &= [u_1, u_2, \cdots, u_{20}] \\ y &= [y_1, y_2, \cdots, y_{20}]^T \\ x_{20} &= x_0 + \sum_{j=1}^{20} y_i u_i \end{aligned}$$

  - Dados A, b, x_{20}

$$\begin{aligned} \bar{U}_{20} &= [\bar{u}_1, \bar{u}_2, \cdots, \bar{u}_{20}] \\ y &= [\bar{y}_1, \bar{y}_2, \cdots, \bar{y}_{20}]^T \\ x_{40} &= x_{20} + \sum_{j=1}^{20} \bar{y}_i \bar{u}_i \neq x_0 + \sum_{j=1}^{40} y_i u_i \end{aligned}$$

Dados $x = 0$, A, b, l_{max}, tol, k

$\epsilon = tol\|b\|_2$, $l = 1$

repeat

$i = 1$, $u_i = b - Ax$, $e_i = \|u_i\|_2 = \rho$,

$u_i = r/e_i$

while $\rho > \epsilon$ **and** $i < k$ **do**

Ortogonalização de Gram-Schmidt

$\vdots$ (descrição ao lado)

Algoritmo QR

$\vdots$ (descrição ao lado)

$i = i + 1$

end while

$i = i - 1$

for $j = i, \cdots, 1$ **do**

$y_i = \frac{e_j - \sum_{t=j+1}^{i} h_{jt} y_t}{h_{jj}}$

end for

$x = \sum_{j=1}^{i} y_j u_j$

$l = l + 1$

until $\rho < \epsilon$ **or** $l \geq l_{max}$

Ortogonalização de Gram-Schmidt

$\tilde{u}_{i+1} = Au_i$

for $j = 1, \cdots, i$ **do**

$h_{j,i} = \tilde{u}_{i+1}^T u_j$, $\tilde{u}_{i+1} = \tilde{u}_{i+1} - h_{j,i} u_i$

end for

$h_{i+1,i} = \|\tilde{u}_{i+1}\|_2$, $u_{i+1} = \frac{\tilde{u}_{i+1}}{h_{i+1,i}}$

Algoritmo QR

for $j = 1, \cdots, i-1$ **do**

$ji = h_{ji}$

$h_{ji} = c_j h_{ji} + s_j h_{j+1,i}$

$h_{j+1,i} = -s_j ji + c_j h_{j+1,i}$

end for

$r_i = \sqrt{h_{ii}^2 + h_{i+1,i}^2}$

$c_i = \frac{h_{ii}}{r_i}$

$s_i = \frac{h_{i+1,i}}{r_i}$

$h_{ii} = r_i$ $(h_{i+1,i} = 0)$

$e_{i+1} = -s_i e_i$

$e_i = c_i e_i$

$\rho = |e_{i+1}|$

```
[x,flag,relres,iter,resvec] = gmres(A,b,k,rtol,maxit)
```

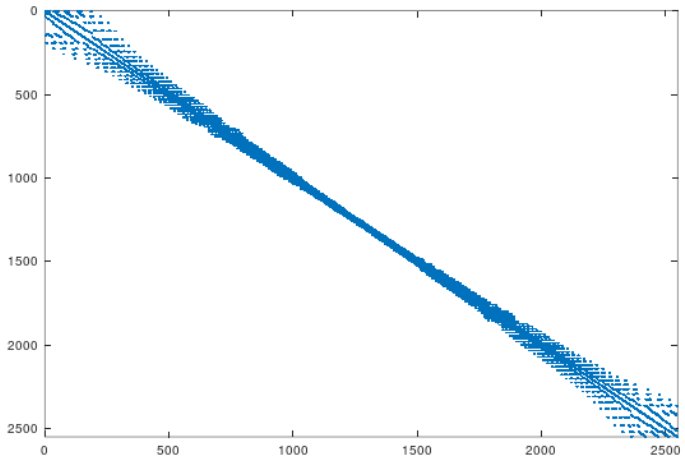
- `A`: Matriz dos coeficientes[3];
- `b`: Vetor dos tewrmos independentes;
- `k`: Número de vetores para o *restart*;
- `rtol`: Tolerância relativa;
- `maxit`: número máximo de ciclos;
- `x`: vetor solução aproximada;
- `flag`: 0 - convergência atingida; 1 - número máximo de iterações atingido; 3 - estagnação do resíduo
- `relres`: valor final do resíduo relativo
- `iter`: vetor contendo o número de ciclos (`iter(1,1)`) e o número de iterações do último ciclo (`iter(1,2)`)[4]
- `resvec`: vetor contendo o resíduo relativo em cada iteração

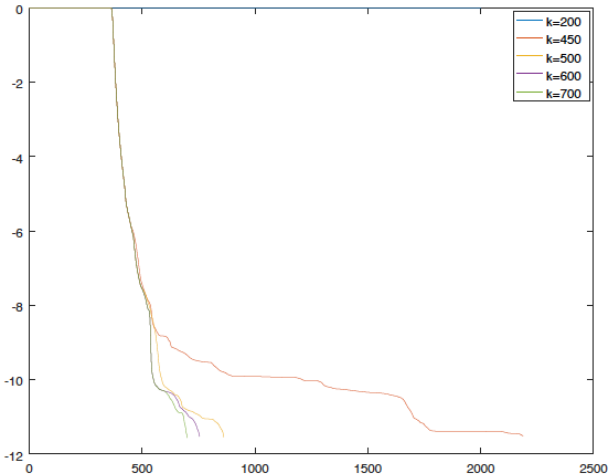
[3] *default*: armazenamento na estrutura CCR (Compressed Column Sparse)

[4] número de iterações gmres é igual a `iter(1,1)*k+iter(1,2)`

A é a matriz `cz2548.mat` de ondem $n = 2548$

$b = A * \texttt{ones}(2548)$, $tol = 0.00001$, $maxit = 10$





Número de vetores na base	flag	Número de Iterações	Tempo (seg)
200	1	2000 (10 ciclos)	7.5686
300	1	3000 (10 ciclos)	20.1890
400	1	4000 (10 ciclos)	43.4056
450	0	2187 (4 ciclos + 387)	28.2763
500	0	861 (1 ciclo + 361)	11.2033
600	0	754 (1 ciclo + 154)	13.8827
700	0	699 (0 ciclos + 699)	20.7482

- Inicialmente apresentado por Yuan, Golub, Plemmons and Cecílio[5] (2002→2004).
- Catabriga, Coutinho and Franca (2004)[6] introduziram restart apresentando comparações com GMRES, Bi-CGSTAB e TFQMR para formulações de elementos finitos.
- Dai and Yuan (2004)[7] introduziram um novo algoritmo (limitação de memória e 1 produto matriz-vetor) usando formulações de diferenças finitas.
- Catabriga, Valli, Melotti, Pessoa and Coutinho (2006)[8] avaliaram o novo algoritmo para equações lineares e não lineares usando o método de Newton inexato.

[5] Study on semi-conjugate direction methods.... IJNME, 60:1383-1399

[6] Evaluating the LCD algorithm .... IJNME, 60:1513-1534

[7] Study on semi-conjugate direction methods .... IJNME, vol. 60, pp. 1383-1399

[8] Performance of LCD .... CNME, vol. 22, pp. 643-656

- **Definição**: Os vetores $p_1, p_2, \cdots, p_n \in \mathbb{R}^n$ são chamados vetores de direções conjugadas à esquerda (LCD) de uma matriz real não singular A se:

$$\begin{aligned} p_i^T A p_j &= 0 \text{ para } i < j \\ p_i^T A p_j &\neq 0 \text{ para } i = j \\ &\Downarrow \\ P^T A P &= L, \quad \text{Matriz triangular inferior} \end{aligned}$$

  onde $P = [p_1, p_2, \cdots, p_n]$

- P é linearmente Independente

$$x \in \mathbb{R}^n \Rightarrow x = x_0 + \sum_{i=1}^{n} \alpha_i p_i$$

Sejam

- x^* solução exata de $Ax = b$
- x_0 arpoximação inicial
- $P = [p_1, p_2, \cdots, p_n]$ vetores de direções conjugadas à esquerda.

$$x \in \mathbb{R}^n \Rightarrow x = x_0 + \sum_{i=1}^{n} \alpha_i p_i$$

$$\begin{aligned} Ax^* &= b \\ x^* &= x_0 + \sum_{i=1}^{n} \alpha_i p_i \\ & A(x_0 + \sum_{i=1}^{n} \alpha_i p_i) = b \\ Ax^* &= Ax_0 + \sum_{i=1}^{n} \alpha_i Ap_i \end{aligned}$$

$$\begin{aligned} r &= b - Ax^* \\ &= b - Ax_0 - \sum_{i=1}^{n} \alpha_i Ap_i \\ &= r_0 - \sum_{i=1}^{n} \alpha_i Ap_i \end{aligned}$$

Seja x^* solução exata $\Rightarrow r = 0$, portanto r é ortogonal a todo p_i:

$$\begin{aligned}
\forall i = 1, \cdots, n,\ p_i^T r &= 0 \\
p_i^T (r_0 - \sum_{i=1}^{n} \alpha_i A p_i) &= 0 \\
p_i^T (r_0 - \alpha_1 A p_1 - \alpha_2 A p_2 - \cdots - \alpha_i A p_i - \cdots - \alpha_n A p_n) &= 0 \\
p_i^T (r_0 - \sum_{j=1}^{i-1} \alpha_j A p_j - \alpha_i A p_i - \cdots - \alpha_n A p_n) &= 0 \\
p_i^T r_{i-1} - \alpha_i p_i^T A p_i - \alpha_{i+1} p_i^T A p_{i+1} - \cdots - \alpha_n p_i^T A p_n) &= 0
\end{aligned}$$

$$\alpha_i = \frac{p_i^T r_{i-1}}{p_i^T A p_i} \Rightarrow x_i = x_0 + \sum_{k=1}^{i} \alpha_k p_k \Rightarrow x^* = x_0 + \sum_{k=1}^{n} \alpha_k p_k$$

Algoritmo Inicial:

Dados $x_0 = 0$, A, b, N_{max}, tol

"Escolha" $[p_1, p_2, \cdots, p_n]$

$r_0 = b$, $\epsilon = tol * \|b\|_2$

$\rho = \|r_0\|_2$, $i = 1$

while $\rho > \epsilon$ **and** $i \leq N_{max}$ **do**

$v_i = Ap_i$

$\alpha_i = \frac{p_i^T r_{i-1}}{p_i^T v_i}$

$x_i = x_{i-1} + \alpha_i p_i$

$r_i = r_{i-1} - \alpha_i v_i$

$\rho = \|r_i\|_2$

$i = i + 1$

end while

Como obter o conjunto de direções conjugadas à esquerda $[p_1, p_2, \cdots, p_n]$?

LCD - Cálculo do conjunto $[p_1, p_2, \cdots, p_n]$

É possível demonstrar a existência de uma relação de recorrência entre $[p_1, p_2, \cdots, p_i]$ e r_i para calcular p_{i+1}:

$$p_{i+1} = r_i + \sum_{j=1}^{k} \beta_j p_j$$

Sabendo que:

$$p_j^T A p_{i+1} = 0 \ \forall j = 1, \cdots, i$$

$$\begin{aligned}
p_{i+1} &= r_i + \beta_1 p_1 + \cdots + \beta_i p_i \\
p_1^T A p_{i+1} &= p_1^T A r_i + p_1^T A \beta_1 p_1 + p_1^T A \beta_2 p_2 \cdots + p_1^T A \beta_i p_i = 0 \\
p_2^T A p_{i+1} &= p_2^T A r_i + p_2^T A \beta_1 p_1 + p_2^T A \beta_2 p_2 \cdots + p_2^T A \beta_i p_i = 0 \\
&\vdots \\
p_i^T A p_{i+1} &= p_i^T A r_i + p_i^T A \beta_1 p_1 + p_i^T A \beta_2 p_2 \cdots + p_i^T A \beta_i p_i = 0
\end{aligned}$$

Cálculo do conjunto $[p_1, p_2, \cdots, p_n]$:

$$\begin{aligned} p_1^T p_1 \beta_1 &= -p_1^T A r_k \\ +p_2^T A p_1 \beta_1 + p_2^T A p_2 \beta_2 &= -p_2^T A r_i \\ &\vdots \\ p_i^T A p_1 \beta_1 + p_i^T A p_2 \beta_2 + \cdots + p_i^T A p_i \beta_i &= -p_i^T A r_i \end{aligned}$$

$$\begin{bmatrix} p_1^T A p_1 & & & \\ p_2^T A p_1 & p_2^T A p_2 & & \\ \vdots & \vdots & \ddots & \\ p_i^T A p_1 & p_i^T A p_2 & \cdots & p_i^T A p_i \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_i \end{bmatrix} = \begin{bmatrix} -p_1^T A r_i \\ -p_2^T A r_i \\ \vdots \\ -p_i^T A r_i \end{bmatrix}$$

Sendo p_1 escolhido tal que $p_1^T A p_1 \neq 0$
Escolha conveniente: $p_1 = r_0$

Cálculo do conjunto $[p_1, p_2, \cdots, p_n]$:
Seja $p_{i+1} = r_i$

$$
\begin{aligned}
\beta_1 &= -\frac{p_1^T A p_{i+1}}{p_1^T A p_1} \\
p_{i+1} &= p_{i+1} + \beta_1 p_1 \\
\beta_2 &= -\frac{p_2^T A r_i + p_2^T A \beta_1 p_1}{p_2^T A p_2} = -\frac{p_2^T A p_{i+1}}{p_2^T A p_2} \\
p_{i+1} &= p_{i+1} + \beta_2 p_2 \\
\beta_3 &= -\frac{p_3^T A (r_i + \beta_1 p_1 + \beta_2 p_2)}{p_3^T A p_3} \\
&\vdots \\
p_{i+1} &= p_{i+1} + \beta_{i-1} p_{i-1} \\
\beta_i &= -\frac{p_i^T A p_{i+1}}{p_i^T A p_i} \\
p_{i+1} &= p_{i+1} + \beta_i p_i
\end{aligned}
$$

Cálculo de p_{i+1}, supondo conhecidos $p_1, p_2, \cdots, p_i$:

$p_{i+1} = r_i$
for $j = 1, \cdots, i$ **do**
$\beta_j = -\frac{p_j^T A p_{i+1}}{p_j^T A p_j}$
$p_{i+1} = p_{i+1} + \beta_j p_j$
end for

Algoritmo (versão p):

Dados $x_0 = 0$, A, b, N_{max}, tol

$r_0 = b$, $\rho = \|r_0\|_2$

"Escolha" p_1 tal que $p_1^T A p_1 \neq 0$ ($p_1 = r_0$)

$\epsilon = tol * \|b\|_2$, $i = 1$

while $\rho > \epsilon$ **and** $i \leq N_{max}$ **do**

$v_i = Ap_i$

$\alpha_i = \frac{p_i^T r_{i-1}}{p_i^T v_i}$

$x_i = x_{i-1} + \alpha_i p_i$

$r_i = r_{i-1} - \alpha_i v_i$

$p_{i+1} = r_i$

for $j = 1, \cdots, i$ **do**

$\beta_i = -\frac{p_j^T A p_{i+1}}{p_j^T vj}$

$p_{i+1} = p_{i+1} + \beta_j p_j$

end for

$\rho = \|r_i\|_2$

$i = i + 1$

end while

Em cada iteração i são necessários $i + 1$ produtos matriz-vetor!!!
Algoritmo Ineficiente!!!

Algoritmo (versão pq):

Supor $q_i = Ap_i$, então $\alpha_i = \frac{p_i^T r_{i-1}}{p_i^T q_i}$
Inicializando o loop de cálculo de p_{i+1} com $q_{i+1} = Ap_{i+1}$ tem-se:

$$\begin{aligned} \beta_j &= -\frac{p_j^T q_{i+1}}{p_j^T q_j} \\ p_{i+1} &= p_{i+1} + \beta_j p_j \\ Ap_{i+1} &= Ap_{i+1} + \beta_j Ap_j \\ q_{i+1} &= q_{i+1} + \beta_j q_j \end{aligned}$$

Algoritmo (versão pq):

Dados $x_0 = 0$, A, b, N_{max}, tol

$r_0 = b$, $\rho = \|r_0\|_2$, $\epsilon = tol * \|b\|_2$

"Escolha" p_1 tal que $p_1^T A p_1 \neq 0$ $(p_1 = r_0)$

$q_1 = A p_1$, $i = 1$

while $\rho > \epsilon$ **and** $i \leq N_{max}$ **do**

$\alpha_i = \frac{p_i^T r_{i-1}}{p_i^T q_i}$

$x_i = x_{i-1} + \alpha_i p_i$

$r_i = r_{i-1} - \alpha_i v_i$

$p_{i+1} = r_i$, $q_{i+1} = A p_{i+1}$

for $j = 1, \cdots, i$ **do**

$\beta_j = -\frac{p_j^T q_{i+1}}{p_j^T q_j}$

$p_{i+1} = p_{i+1} + \beta_j p_j$

$q_{i+1} = q_{i+1} + \beta_j q_j$

end for

$\rho = \|r_i\|_2$

$i = i + 1$

end while

Complexidade e Armazenamento do Método LCD:

- i-ésima iteração:
  - Número de operações:
    - 1 matvec
    - $2i + 2$ saxpy
    - $2i + 2$ dot
  - Complexidade:
    - $\theta((i \times n) + \theta(i^2) + \theta(\text{matvec})$, sendo que $\theta(\text{matvec}) \approx n^2$
    - $\Rightarrow$ será tanto maior quanto maior for i.
  - Armazenamento:
    - $2i$ vetores
    - 1 matriz A

Algoritmo (versão pq) com restart:

Dados $x_0 = 0$, A, b, tol, l_{max}, k

$r = b$, $\rho = \|r\|_2$, $\epsilon = tol * \|b\|_2$; $l = 1$

"Escolha" p_1 tal que $p_1^T A p_1 \neq 0$ ($p_1 = r$)

repeat

$q_1 = A p_1$, $i = 1$

while $\rho > \epsilon$ **and** $i \leq k$ **do**

$\alpha_i = \frac{p_i^T r}{p_i^T q_i}$

$x = x + \alpha_i p_i$, $r = r - \alpha_i v_i$

$p_{i+1} = r_i$, $q_{i+1} = A p_{i+1}$

for $j = 1, \cdots, i$ **do**

$\beta_j = -\frac{p_j^T q_{i+1}}{p_j^T q_j}$

$p_{i+1} = p_{i+1} + \beta_j p_j$, $q_{i+1} = q_{i+1} + \beta_j q_j$

end for

$\rho = \|r\|_2$, $i = i + 1$

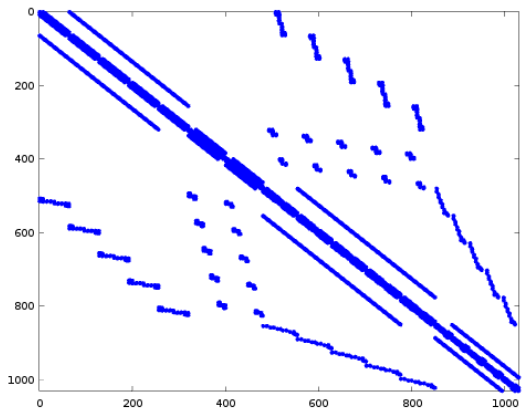
end while

$p_1 = r$ (ou $p_1 = p_{i+1}$), $l = l + 1$

until $\rho \leq \epsilon$ **or** $l > l_{max}$

LCD $\times$ GMRES - Exemplo - `orsirr1.mtx`

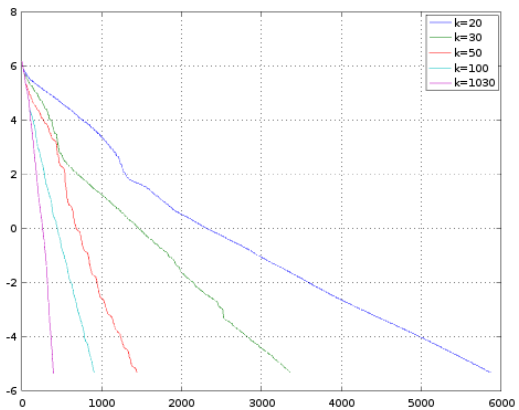
A é a matriz `orsirr1.mtx` de ordem $n = 1030$

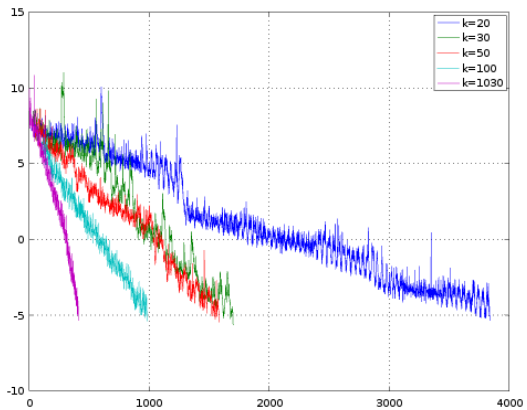


$$b = A * \texttt{ones}(1030),\ tol = 0.00001$$

k	$GMRES(k)_{iter}$	$tempo_{GMRES(k)}(seg)$	$LCD(k)_{iter}$	$tempo_{LCD(k)}(seg)$
20	5872	7.57	3838	3.51
30	3375	4.92	1699	1.95
50	1441	3.17	1581	2.64
100	911	3.63	983	2.90
1030	396	5.69	410	4.61

OBS: Código em octave sem otimizações

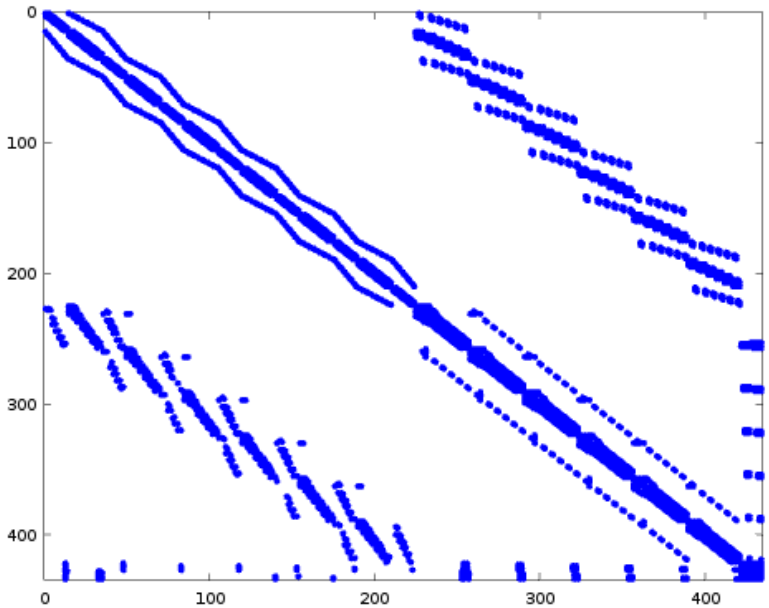




$GMRES - iter \times log(\|r\|)$ $LCD - iter \times log(\|r\|)$

OBS: Lembrar que os critérios de parada dos métodos LCD e GMRES são diferentes!

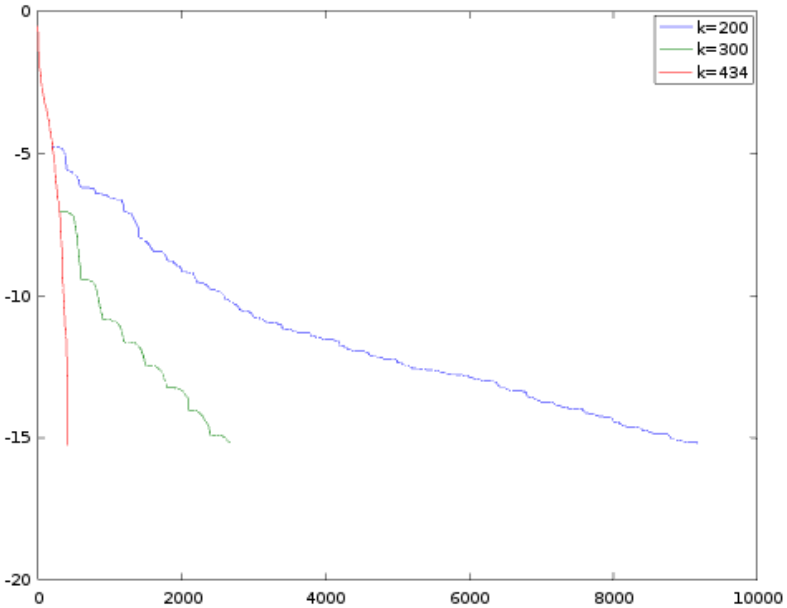
A é a matriz `hor131.mtx` de ordem $n = 434$

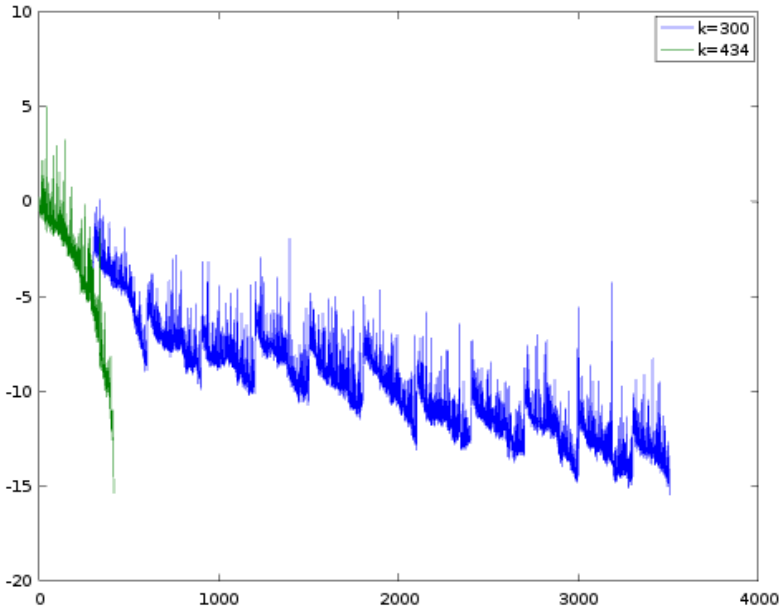


$b = A * \text{ones}(434),\ tol = 0.0000001$

k	$GMRES(k)_{iter}$	$tempo_{GMRES(k)}(seg)$	$LCD(k)_{iter}$	$tempo_{LCD(k)}(seg)$
100	†		†	
200	9173	67.13	†	
300	2670	27.88	3506	25.57
434	416	6.14	417	4.29

OBS: Código em octave sem otimizações





$GMRES - iter \times log(\|r\|)$ $LCD - iter \times log(\|r\|)$

OBS: Lembrar que os critérios de parada dos métodos LCD e GMRES são diferentes!

Ideia Geral:

- Objetivo:
  Acelerar a convergência dos métodos iterativos.
- Dado o sistema $Ax = b$, encontrar uma matrix M (ou matrizes M e N) tal que:

$$\begin{aligned} M^{-1}Ax &= M^{-1}b \text{ ou} \\ AN^{-1}y &= b, \text{ onde } x = N^{-1}y \text{ ou} \\ M^{-1}AN^{-1}y &= M^{-1}b, \text{ onde } x = N^{-1}y \end{aligned}$$

- M é denominada matriz precondicionadora.
- M deve ser uma boa aproximação de A.
- A matriz resultante $\tilde{A} = M^{-1}A$ ou $\tilde{A} = AN^{-1}$ ou $\tilde{A} = M^{-1}AN^{-1}$ não é calculada explicitamente, mas **sua ação é executada no produto matriz vetor** durante o processo iterativo.

Fatoração Jacobi, Seidel e SOR:

Seja $A = L + D + U$, onde:

- L é a parte estritamente inferior de A,
- U é a parte estritamente superior de A.
- D é a diagonal de A.

Fatoração Jacobi:

$$x_{k+1} = G_J x_k + f$$

onde

$$G_J = -D^{-1}(L+U) = I - D^{-1}A$$
$$f = D^{-1}b$$

Fatoração Seidel:

$$x_{k+1} = G_S x_k + f$$

onde

$$G_S = -(D+L)^{-1}U = I - (D+L)^{-1}A$$
$$f = (D+L)^{-1}b$$

Seja $A = M - N$:

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

- $G = M^{-1}(M - A) = I - M^{-1}A$
- Jacobi: $M = D$ e $N = D - A$
- Seidel: $M = D + L$ e $N = M - A$

$$\begin{aligned} x_{k+1} &= Gx_k + f \\ &\Downarrow \\ (I - G)x &= f \\ G &= I - M^{-1}A \\ M^{-1}Ax &= M^{-1}b \end{aligned}$$

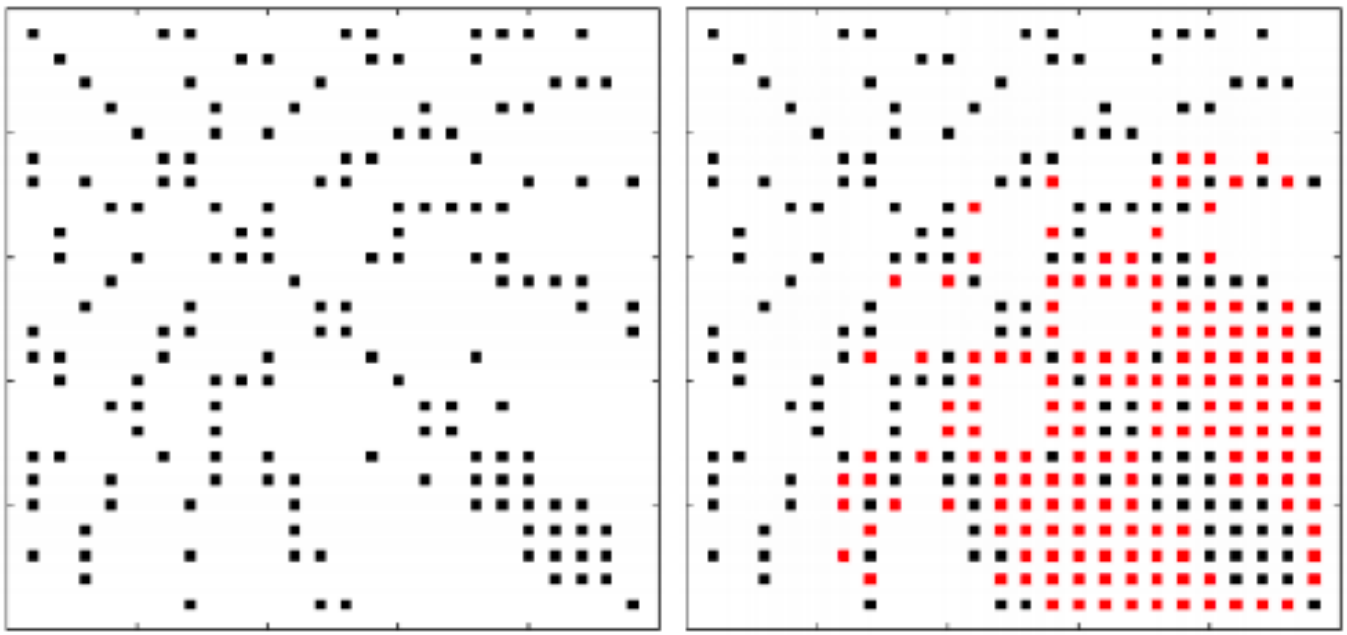
$$
\begin{aligned}
M_J &= D \quad \text{Precondicionador Jacobi} \\
M_{SSOR} &= (D + wL)D^{-1}(D + wU) \quad \text{Precondicionador SOR} \\
M_S &= (D + L)D^{-1}(D + U), \quad \text{Precondicionador Seidel } (w = 1)
\end{aligned}
$$

Sendo $\bar{L} = (D + L)D^{-1}$ e $\bar{U} = (D + U)$

$$
\begin{aligned}
M_S &= \bar{L}\bar{U} \\
\bar{L} &= (D + L)D^{-1} = I + LD^{-1} \\
\bar{U} &= D + U
\end{aligned}
$$

OBS: Não requer armazenamento extra!

Fatoração LU Incompleta: A fatoração LU prevê preenchimento de posições originalmente nulas da matriz original A:



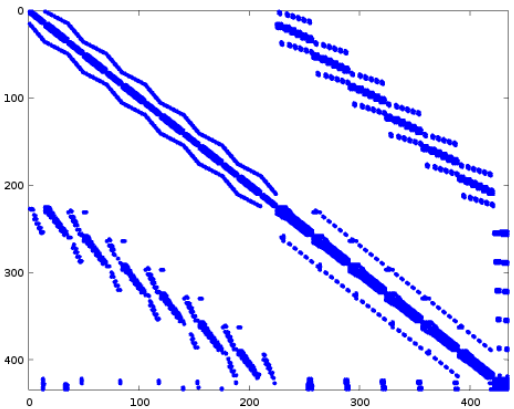
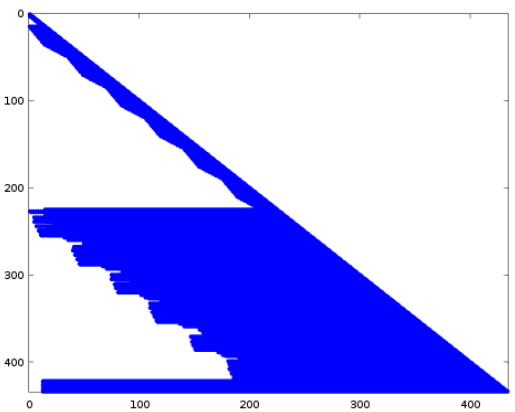
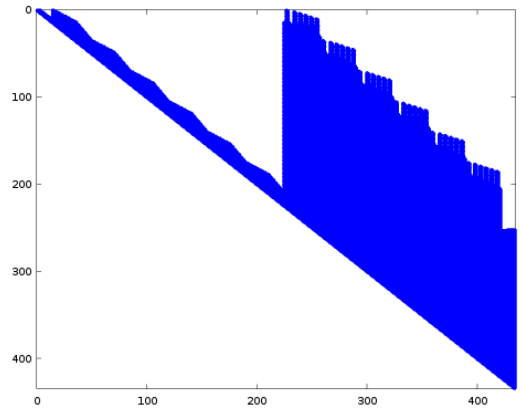
Fill-in = 156

O número de posições preenchidas é denominada *fill-in*.

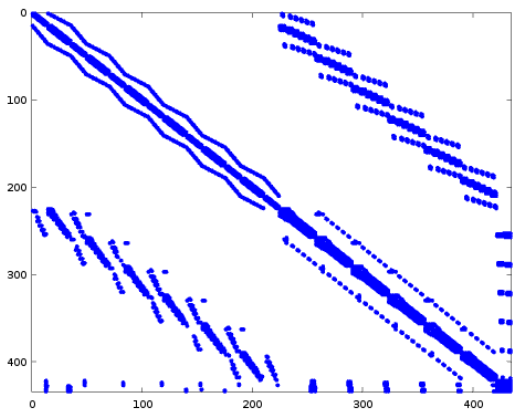
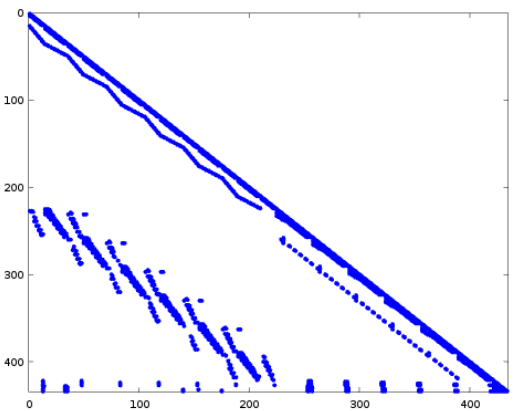
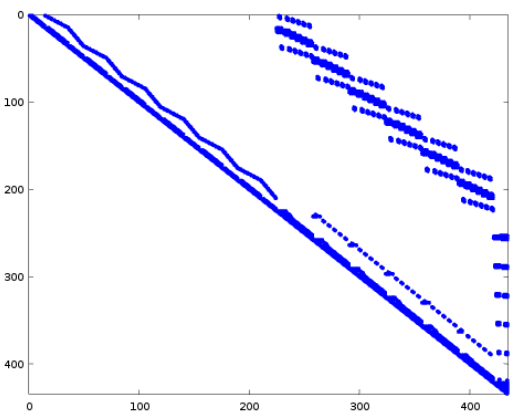
Operação básica: $a_{ij} = a_{ij} + m_{ik} a_{kj}$

- A fatoração incompleta $ILU(k)$ descarta preenchimento de posições originalmente nulas da matriz original A a partir da $(k+1)$-ésima etapa da fatoração.
- A fatoração incompleta $ILU(0)$ não prevê preenchimento de posições originalmente nulas da matriz original A.
- Os fatores $\tilde{L}$ e $\tilde{U}$ resultante da fatoração $ILU(k)$ são tais que: $\tilde{L}\tilde{U} \neq A$.
- O precondicionador $ILU(k)$ é definido por $M = \tilde{L}\tilde{U}$.
- O precondicionador $ILU(k)$ requer armazenamento extra da matriz M.

Fatoração LU $\times$ Fatoração ILU - `hor131.mtx`:

Fatoração LU – $n = 434$ – $nnz(A) = 4182$ – $nnz(L + U) = 94985$ – $\|LU - A\| = 1.5 \times 10^{-16}$

Fatoração $ILU(0)$ – $n = 434$ – $nnz(A) = 4182$ – $nnz(L + U) = 4182$ – $\|\tilde{L}\tilde{U} - A\| = 6.0 \times 10^{-2}$

$$Ax = b \Rightarrow M^{-1}Ax = M^{-1}b \Rightarrow \tilde{A}x = \tilde{b}$$

- M^{-1} não é calculada.
- O produto $M^{-1}A$ não é calculado explicitamente.
- A ação do precondicionador se dá durante o processo iterativo quando for necessário executar operações com $\tilde{A}$ e $\tilde{b}$:

$$\begin{aligned} \text{No início do algoritmo: } \tilde{b} &= M^{-1}b \Rightarrow M\tilde{b} = b \\ \text{Em toda iteração: } p &= \tilde{A}u \Rightarrow p = M^{-1}Au \\ (1) \quad & z = Au \\ (2) \quad & p = M^{-1}z \Rightarrow Mp = z \end{aligned}$$

- O precondicionador será tão mais eficiente quanto mais eficiente e trivial for a solução dos sistemas $M\tilde{b} = b$ e $Mp = z$.

Ação do Precondicionador M_J:

$$
\begin{aligned}
M_J &= D \\
\text{No início do algoritmo:} \quad \tilde{b} &= D^{-1}b \Rightarrow \tilde{b}_i = \frac{b_i}{a_{ii}} \\
\text{Em toda iteração:} \quad p &= \tilde{A}u \Rightarrow p = D^{-1}Au \\
(1) \quad & z = Au \\
(2) \quad & p = D^{-1}z \Rightarrow p_i = \frac{z_i}{a_{ii}}
\end{aligned}
$$

Ação do Precondicionador M_S (M_{SSOR} para $w = 1$):

$$
\begin{aligned}
M_S &= (D+L)D^{-1}(D+U) \text{ ou} \\
&= (I+LD^{-1})(D+U) = \bar{L}\bar{U} \\
\text{No início do algoritmo:} \quad \tilde{b} &= (\bar{L}\bar{U})^{-1}b \Rightarrow \bar{L}\bar{U}\tilde{b} = b \\
&(1) \quad \bar{L}z = b \\
&(2) \quad \bar{U}\tilde{b} = z \\
\text{Em toda iteração:} \quad p &= \tilde{A}u \Rightarrow p = (\bar{L}\bar{U})^{-1}Au \\
&(1) \quad z = Au \\
&(2) \quad p = (\bar{L}\bar{U})^{-1}z \Rightarrow \bar{L}\bar{U}p = z \\
&\quad (2.1) \quad \bar{L}v = z \\
&\quad (2.2) \quad \bar{U}p = v
\end{aligned}
$$

$\Rightarrow$ Resolver sistemas triangulares triviais em cada iteração

$\Rightarrow$ Não há armazenamento extra

Ação do Precondicionador M_S (M_{SSOR} para $w = 1$)

$$\bar{L}v = z \Rightarrow \begin{bmatrix} 1 & & & & \\ \frac{a_{21}}{a_{22}} & 1 & & & \\ \frac{a_{31}}{a_{33}} & \frac{a_{32}}{a_{33}} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \frac{a_{n1}}{a_{nn}} & \cdots & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{bmatrix} \Rightarrow v_i = z_i - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} v_j$$

$$\bar{U}p = v \Rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ & a_{22} & a_{23} & \cdots & a_{2n} \\ & & a_{33} & \cdots & a_{3n} \\ & & & \ddots & \vdots \\ & & & & a_{nn} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix}$$

$$\Rightarrow p_i = \left(v_i - \sum_{j=i+1}^{n} a_{ij} p_j \right) / a_{ii}$$

Ação do Precondicionador $ILU(k)$:

$$
\begin{array}{rrll}
 & M & = & \tilde{L}\tilde{U} \\
\text{No início do algoritmo:} & \tilde{b} & = & (\tilde{L}\tilde{U})^{-1}b \Rightarrow \tilde{L}\tilde{U}\tilde{b} = b \\
 & & (1) & \tilde{L}z = b \\
 & & (2) & \tilde{U}\tilde{b} = z \\
\text{Em toda iteração:} & p & = & \tilde{A}u \Rightarrow p = (\tilde{L}\tilde{U})^{-1}Au \\
 & & (1) & z = Au \\
 & & (2) & p = (\tilde{L}\tilde{U})^{-1}z \Rightarrow \tilde{L}\tilde{U}p = z \\
 & & & (2.1) \quad \tilde{L}v = z \\
 & & & (2.2) \quad \tilde{U}p = v
\end{array}
$$

$\Rightarrow$ Resolver sistemas triangulares triviais em cada iteração

$\Rightarrow$ Armazenamento extra dos coeficientes de $\tilde{L}$ e $\tilde{U}$ em M.

Ação do Precondicionador $ILU(k)$:

$$\tilde{L}v = z \Rightarrow \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ m_{21} & m_{22} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{bmatrix} \Rightarrow v_i = z_i - \sum_{j=1}^{i-1} m_{ij} v_j$$

$$\tilde{U}p = v \Rightarrow \begin{bmatrix} m_{11} & m_{12} & m_{13} & \cdots & m_{1n} \\ & m_{22} & m_{23} & \cdots & m_{2n} \\ & & m_{33} & \cdots & m_{3n} \\ & & & \ddots & \vdots \\ & & & & m_{nn} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix}$$

$$\Rightarrow p_i = \left(v_i - \sum_{j=i+1}^{n} m_{ij} p_j \right) / m_{ii}$$

Dados $x = 0$, A, b, M, l_{max}, tol, k

$\epsilon = tol\|M^{-1}b\|_2$, $l = 1$

repeat

$i = 1$, $u_i = M^{-1}(b - Ax)$,

$e_i = \|u_i\|_2 = \rho$, $u_i = r/e_i$

while $\rho > \epsilon$ **and** $i < k$ **do**

Ortogonalização de Gram-Schmidt

$\vdots$ (descrição ao lado)

Algoritmo QR

$\vdots$ (descrição ao lado)

$i = i + 1$

end while

$i = i - 1$

for $j = i, \cdots, 1$ **do**

$y_i = \frac{e_j - \sum_{t=j+1}^{i} h_{jt} y_t}{h_{jj}}$

end for

$x = \sum_{j=1}^{i} y_j u_j$

$l = l + 1$

until $\rho < \epsilon$ **or** $l \geq l_{max}$

Ortogonalização de Gram-Schmidt

$\tilde{u}_{i+1} = M^{-1}Au_i$

for $j = 1, \cdots, i$ **do**

$h_{j,i} = \tilde{u}_{i+1}^T u_j$, $\tilde{u}_{i+1} = \tilde{u}_{i+1} - h_{j,i}u_i$

end for

$h_{i+1,i} = \|\tilde{u}_{i+1}\|_2$, $u_{i+1} = \frac{\tilde{u}_{i+1}}{h_{i+1,i}}$

Algoritmo QR

for $j = 1, \cdots, i-1$ **do**

$ji = h_{ji}$

$h_{ji} = c_j h_{ji} + s_j h_{j+1,i}$

$h_{j+1,i} = -s_j ji + c_j h_{j+1,i}$

end for

$r_i = \sqrt{h_{ii}^2 + h_{i+1,i}^2}$

$c_i = \frac{h_{ii}}{r_i}$

$s_i = \frac{h_{i+1,i}}{r_i}$

$h_{ii} = r_i$ $(h_{i+1,i} = 0)$

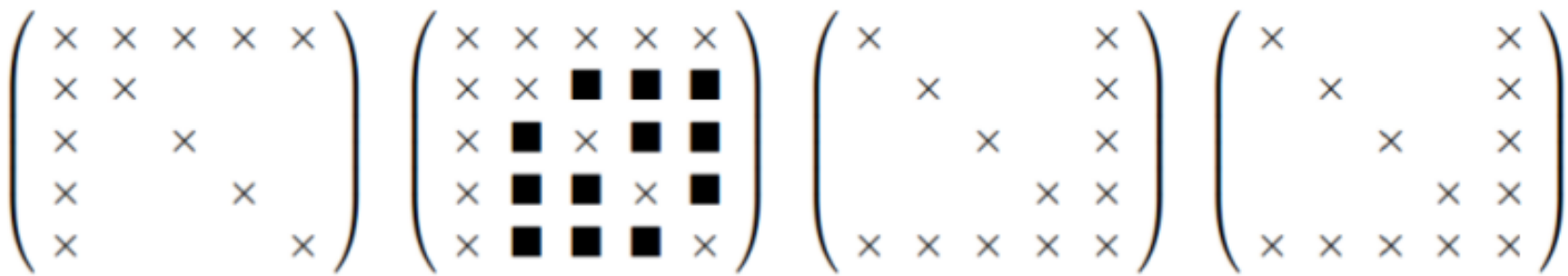
$e_{i+1} = -s_i e_i$

$e_i = c_i e_i$

$\rho = |e_{i+1}|$

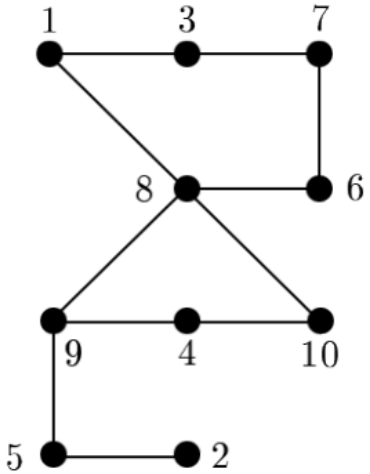
Reordenamento de linhas e colunas de sistemas lineares:

- O preenchimento representa um fator limitante a eficiência dos precondicionadores $ILU(k)$ quando k cresce.
- O Reordenamento é usado para reduzir o preenchimento causado pelo precondicionador $ILU(k)$, para $k > 0$.

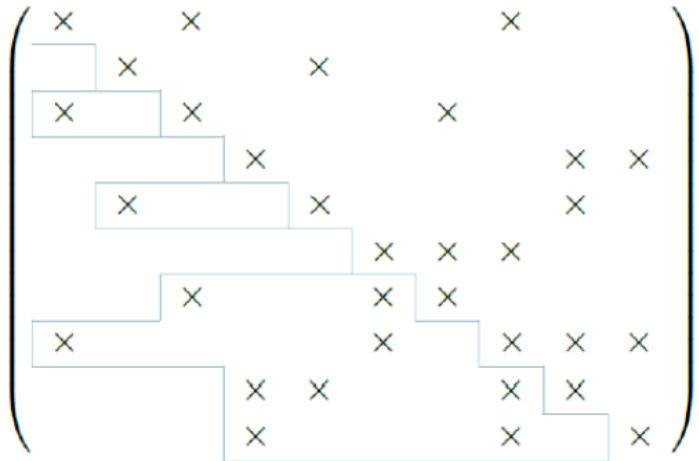


- Consequentemente o tempo de CPU é reduzido

Rerrotulação de Grafos × Matriz de Adjacência:

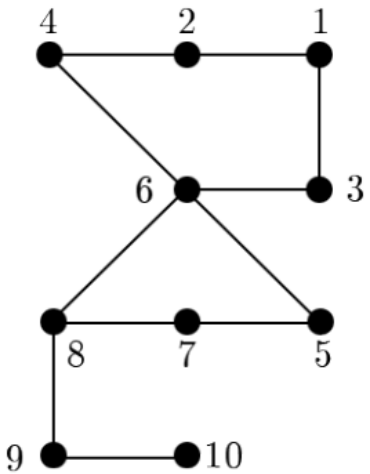


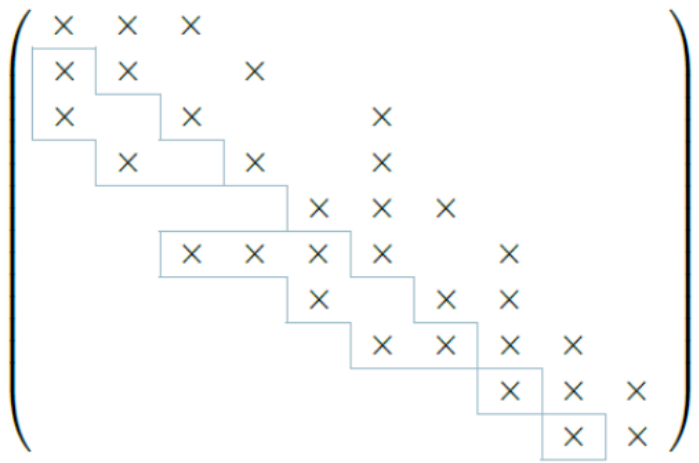
(a) Grafo $G1$



(b) Matriz de Adjacência $M1$ de $G1$

Métricas para $M1$:
$banda = 7$
$Envelope = 27$
$Fill - in = 34$





Métricas para $M2$:
$banda = 3$
$Envelope = 14$
$Fill - in = 6$

Algoritmos de Reordenamento:

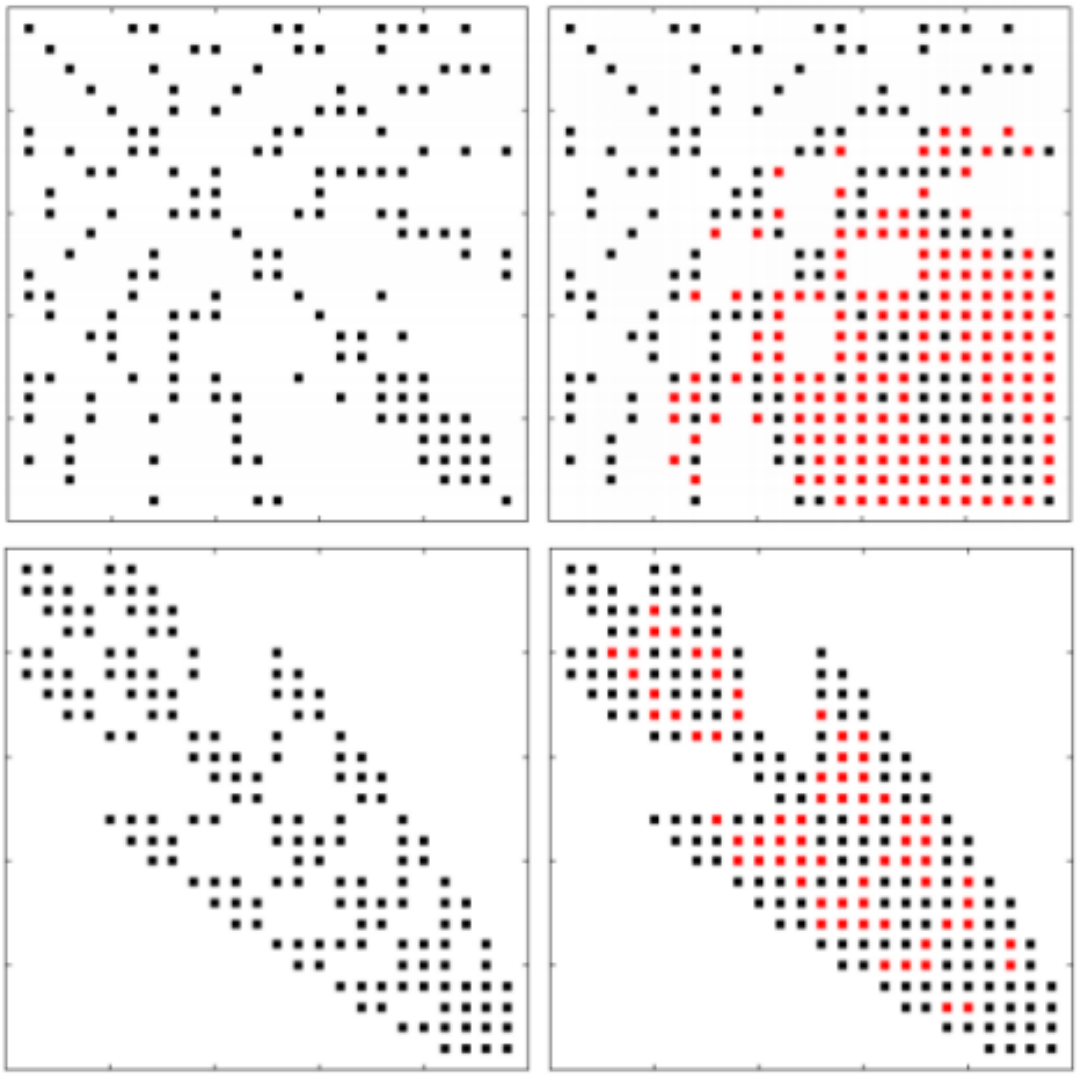
- Ideia Principal:

$$Ax = b \Rightarrow PAP^T Px = Pb \quad (1)$$

  onde P é uma matriz de permutação.
- Exemplos de Algoritmos:
  - Reverse Cuthill MacKee (RCM)[9]: rotula os vértices de maneira que os nós adjacentes recebam rótulos da forma mais ordenada possível.
  - Sloan: funciona atribuindo estados a cada vértice: inativo, ativo, pós-ativo.
  - Espectral[10]: o novo reordenamento é calculado a partir da permutação do vetor de autovetores associado ao segundo menor autovalor (vetor de Fiedler - conectividade algébrica) da matriz laplaciana associada ao grafo.
  - Nested Dissection[11]: baseado em vértices separadores.

[9] Cuthill and McKee (1969), George (1971),George and Liu (1979)
[10] Barnard, Pothen, Simon(1993)
[11] George (1973)

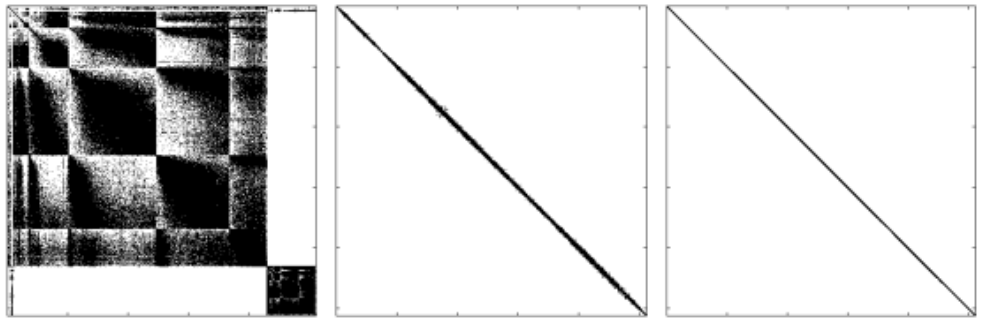
Reordenamento: Efeito do Reordenamento RCM



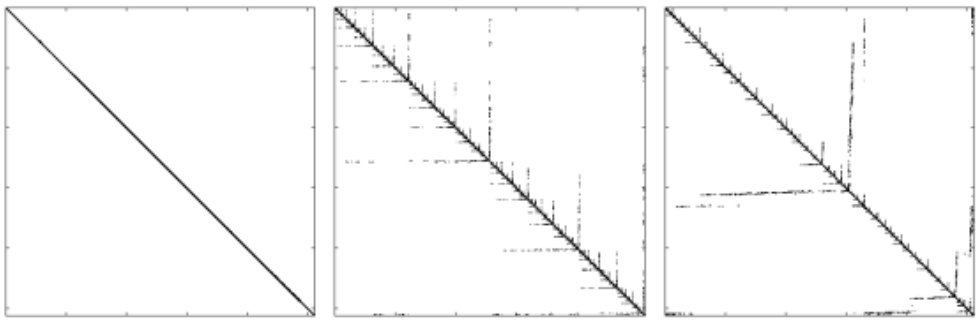
Fill-in = 156

Fill-in = 70

Exemplo Precondicionador ILU(k) - Matriz thermomech_TK: $n = 102158$, $nnz = 711558$



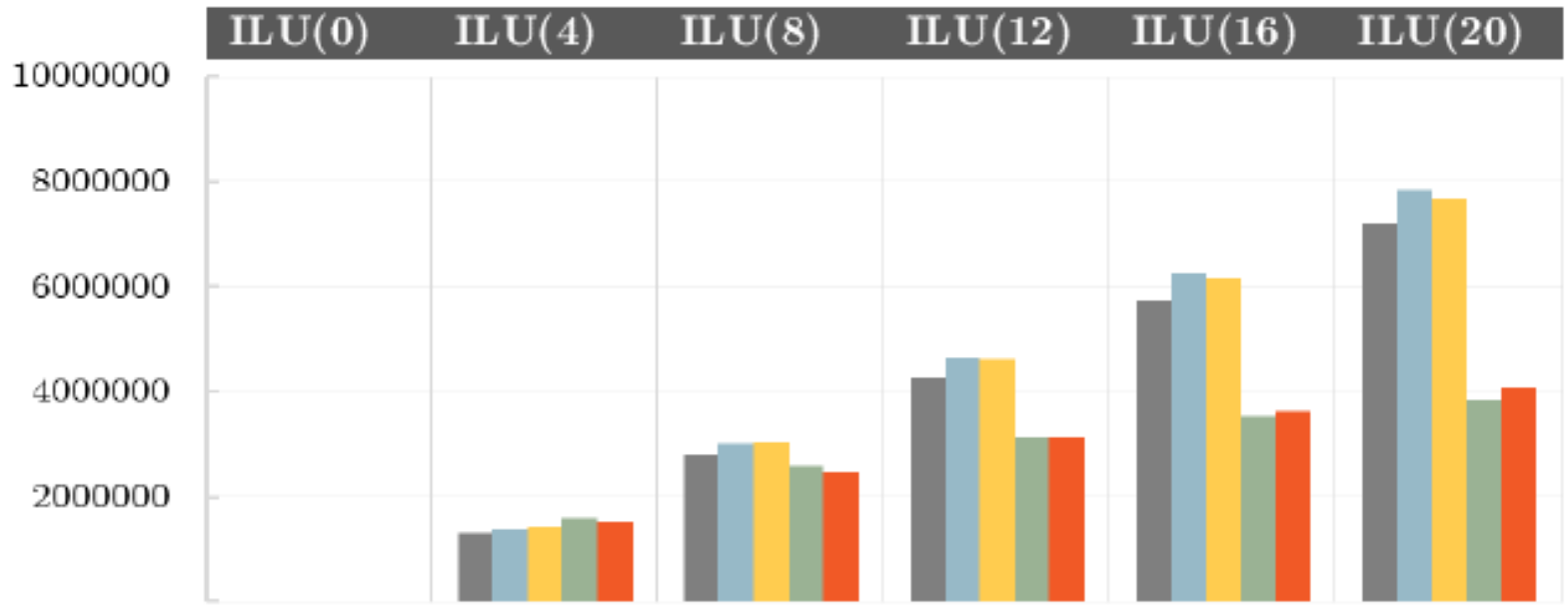
(a) Original (b) SLOAN (c) RCM



(d) ESPECTRAL (e) ND (f) AMD

	env(.)	lb(.)	Tempo de CPU
Original	2.667.823.445	102.138	-
Sloan	16.072.717	676	2,112
RCM	17.882.411	253	0,581
Espectral	14.635.485	389	2,738
ND	-	-	0,863
AMD	-	-	0,192

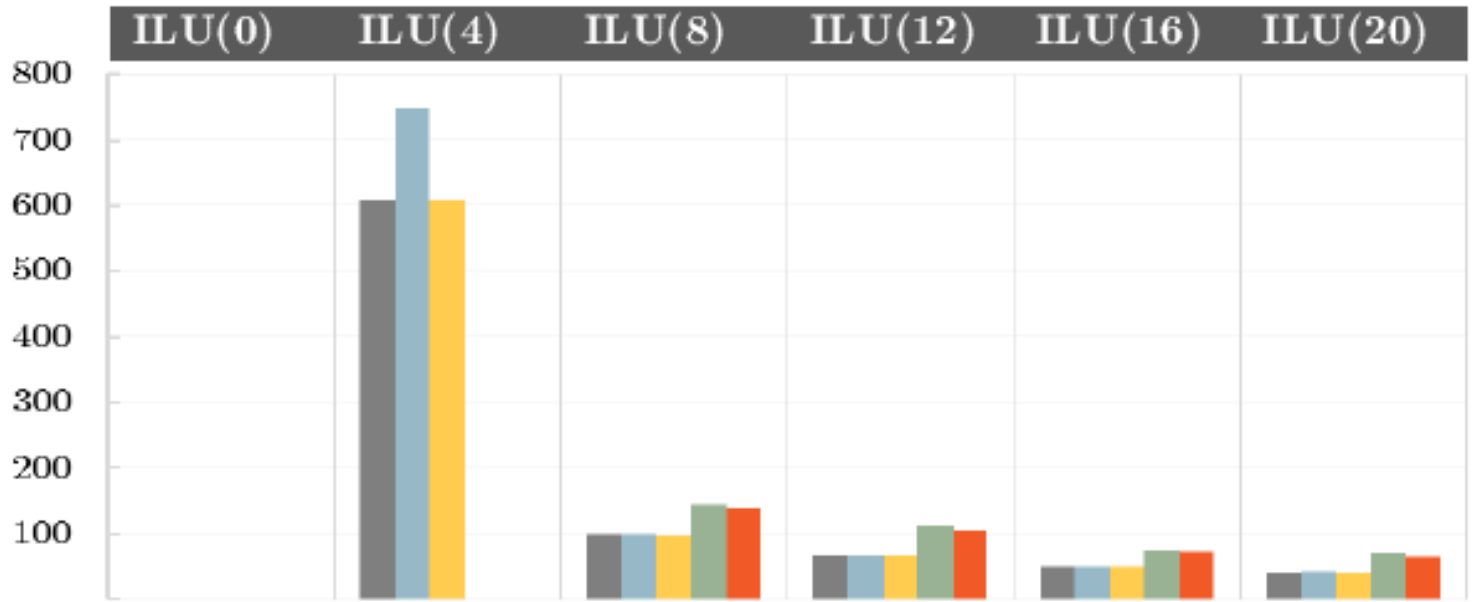
Precondicionador ILU(k) - Matriz thermomech_TK - GMRES(150)



		ILU(0)	ILU(4)	ILU(8)	ILU(12)	ILU(16)	ILU(20)
S/ reord.	-		2.418.982	6.234.694	11.386.296	17.735.910	25.168.278
Sloan	-		1.299.766	2.794.478	4.281.708	5.745.876	7.175.972
RCM	-		1.377.086	3.012.390	4.648.154	6.261.578	7.840.310
Espectral	-		1.416.644	3.031.658	4.619.810	6.168.354	7.666.906
ND	-		1.594.042	2.563.166	3.137.532	3.530.792	3.833.678
AMD	-		1.505.844	2.445.502	3.118.058	3.636.438	4.055.210

Preenchimento

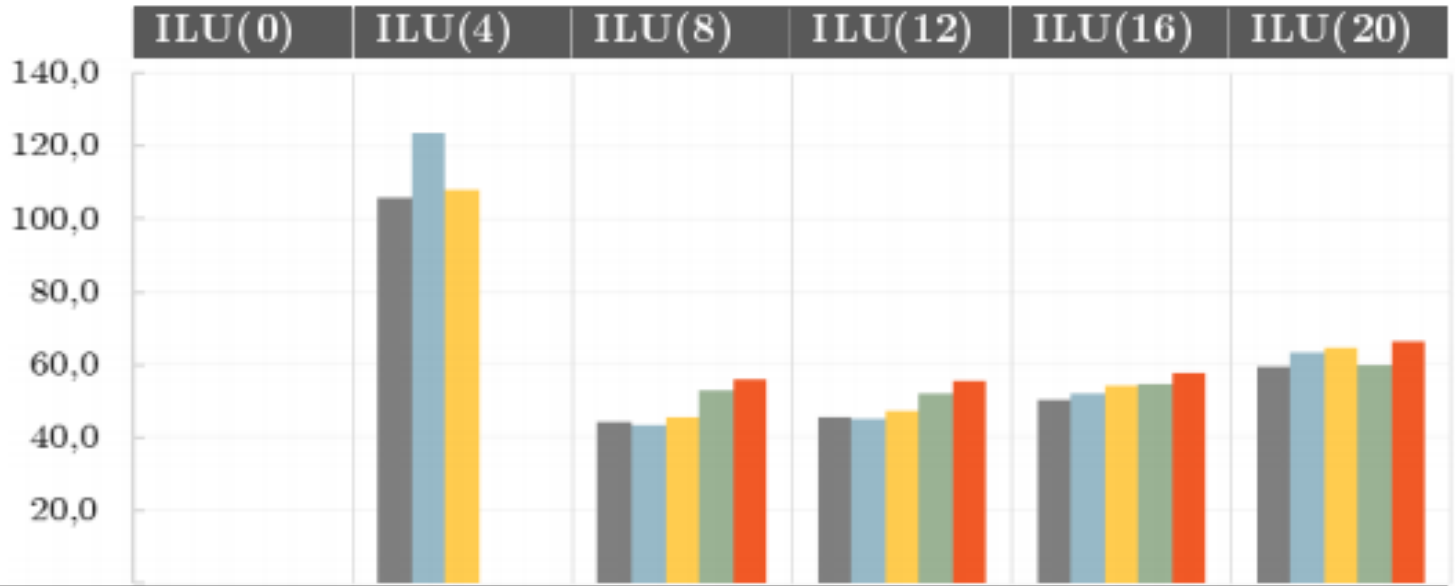
Precondicionador ILU(k) - Matriz thermomech_TK - GMRES(150)



	ILU(0)	ILU(4)	ILU(8)	ILU(12)	ILU(16)	ILU(20)
S/ reord.	†	†	141	97	75	59
Sloan	†	607	99	68	51	41
RCM	†	748	100	68	51	42
Espectral	†	606	98	66	51	41
ND	†	†	143	111	75	70
AMD	†	†	138	104	72	65

Número de Iterações

Precondicionador ILU(k) - Matriz thermomech_TK - GMRES(150)



	ILU(0)	ILU(4)	ILU(8)	ILU(12)	ILU(16)	ILU(20)
S/ reord.	†	†	95,87	244,41	797,94	2155,92
Sloan	†	105,90 •	44,07	45,71	50,32 •	59,32 •
RCM	†	123,45	43,45 •	45,11 •	52,04	63,24
Espectral	†	107,80	45,45	47,09	54,03	64,46
ND	†	†	52,72	51,90	54,73	59,82
AMD	†	†	55,94	55,47	57,61	66,34

Tempo de Processamento

- $M^{-1}A$ deve ser simétrica definida positiva.
- Condição: Deve existir uma matriz E, tal que $M = EE^T - E$ pode ser obtida através da fatoração de Cholesky, para tal M deve ser simétrica definida positiva:

$$\begin{aligned} Ax &= b \\ &\Downarrow \\ E^{-1}AE^{-T}E^Tx &= E^{-1}b \\ &\Downarrow \\ \tilde{A}\tilde{x} &= \tilde{b} \end{aligned}$$

onde

$$\begin{aligned} \tilde{A} &= E^{-1}AE^{-T} \\ \tilde{x} &= E^Tx \\ \tilde{b} &= E^{-1}b \end{aligned}$$

- $E^{-1}AE^{-T}$ é simétrica definida positiva

$$\begin{aligned}\tilde{r}_i &= \tilde{b}_i - \tilde{A}\tilde{x}_i \\ &= E^{-1}b - E^{-1}AE^{-T}E^T x \\ &= E^{-1}r_i\end{aligned}$$

Similarmente:

$$\begin{aligned}\tilde{d}_i &= E^T d_i \\ \tilde{x}_i &= E^T x_i\end{aligned}$$

Observando que $E^{-T}E^{-1} = M^{-1}$, temos:

$$d_0 = M^{-1}r_0 \quad \lambda_i = \frac{r_i^T M^{-1} r_i}{d_i^T A d_i} \quad \beta_{i+1} = \frac{r_{i+1}^T M^{-1} r_{i+1}}{r_i^T M^{-1} r_i} \quad d_{i+1} = M^{-1}r_{i+1} + \beta_{i+1} d_i$$

Dados $x_0 = 0$, A, b, M, N_{max}, tol_ϵ

$r_0 = b$, $d_0 = M^{-1} r_0$

$\delta_{new} = r_0^T d_0$

$\delta_0 = \delta_{new}$, $i = 0$

while $\delta_{new} > tol_\epsilon{}^2 \delta_0$ **and** $i \leq N_{max}$ **do**

$v_i = A d_i$

$\lambda_i = \frac{\delta_{new}}{d_i^t v_i}$

$x_{i+1} = x_i + \lambda_i d_i$

$r_{i+1} = r_i - \lambda_i v$

$\delta_{old} = \delta_{new}$

$s = M^{-1} r_{i+1}$

$\delta_{new} = r_{i+1}^T s$

$\beta_{i+1} = \frac{\delta_{new}}{\delta_{old}}$

$d_{i+1} = s + \beta_{i+1} d_i$

$i = i + 1$

end while

Precondicionamento no Octave

```
[x,flag,relres,iter,resvec] = gmres(A,b,k,rtol,maxit,M1,M2)
```

- `A`: Matriz dos coeficientes;
- `b`: Vetor dos tewrmos independentes;
- `k`: Número de vetores para o *restart*;
- `rtol`: Tolerância relativa;
- `maxit`: número máximo de ciclos;
- `M1,M2`: matrizes que definem o precondicionamento $M1^{-1}AM2^{-1}M2x = M1^{-1}b$ ou
- `P`: matriz que define o precondicionamento $(M1 * M2)^{-1}Ax = (M1 * M2)^{-1}b$
- `x`: vetor solução aproximada;
- `flag`: 0 - convergência atingida; 1 - n. máx. iter.; 3 - estagnação do resíduo
- `relres`: valor final do resíduo relativo
- `iter`: vetor contendo o número de ciclos (`iter(1,1)`) e o número de iterações do último ciclo (`iter(1,2)`)
- `resvec`: vetor contendo o resíduo relativo em cada iteração

Gradiente Conjugado no Octave

```
[x,flag,relres,iter,resvec] =
pcg(A,b,tol,maxit,M1,M2)
```

- `A`: Matriz dos coeficientes simétrica definida positiva;
- `b`: Vetor dos termos independentes;
- `tol`: Tolerância relativa;
- `maxit`: número máximo de iterações;
- `M1,M2`: matrizes que definem o precondicionamento $M1^{-1}AM2^{-1}M2x = M1^{-1}b$ ou
- `P`: matriz que define o precondicionamento $(M1 * M2)^{-1}Ax = (M1 * M2)^{-1}b$
- `x`: vetor solução aproximada;
- `flag`: 0 - convergência atingida; 1 - número máximo de iterações atingido; 3 - estagnação do resíduo
- `relres`: valor final do resíduo relativo
- `iter`: número de iterações executadas
- `resvec`: vetor contendo o resíduo relativo em cada iteração