



Métodos numéricos em Ciências Mecânicas

Programa de Pós-Graduação em Ciências Mecânicas

Encontro 7

- Decomposição de Crout e Doolittle;
- LU e inversão matricial;
- Matrizes especiais (TDMA e decomposição de Cholesky);
- Métodos iterativos: Gauss-Seidel e convergência;
- Um exemplo aplicado em Engenharia nuclear;

Um pouco mais de terminologia

Matriz 1 [U]

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{bmatrix}$$

Matriz 2 [L]

$$\begin{bmatrix} 1 & 0 & 0 \\ f_{21} & 1 & 0 \\ f_{31} & f_{32} & 1 \end{bmatrix}$$

Decomposição de Doolittle ou fatoração

Matriz 1 [U]

$$\begin{bmatrix} 1 & a_{12} & a_{13} \\ 0 & 1 & a'_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Matriz 2 [L]

$$\begin{bmatrix} f_{11} & 0 & 0 \\ f_{21} & f_{22} & 0 \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Decomposição de Crout

Fórmulas fechadas para a decomposição de Crout



$\ell_{i1} = a_{i1} \rightarrow i = 1, 2, \dots, n$ → monta a primeira coluna de [L]

$u_{11} = 1$
 $u_{1j} = a_{1j}/\ell_{11} \rightarrow j = 2, 3, \dots, n$ → monta a primeira linha de [U]

$i = j, j+1, \dots, n \rightarrow \ell_{ij} = a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj}$ → monta o resto de [L] até a penúltima coluna

$k = j+1, j+2, \dots, n \rightarrow u_{jk} = \frac{a_{jk} - \sum_{i=1}^{j-1} \ell_{ji} u_{ik}}{\ell_{jj}}$ → monta o resto de [U]

$\ell_{nn} = a_{nn} - \sum_{k=1}^{n-1} \ell_{nk} u_{kn}$ → monta o último termo de [L]

Decomposição L.U e inversão matricial

- Sabemos que a matriz inversa $[A]^{-1}$ de $[A]$ é aquela para a qual $[A]^{-1}[A]=[I]$;
- Podemos usar a própria decomposição L.U para calcular a inversa de $[A]$;
- Vamos apresentar aqui o algoritmo para isso...
- Note que: $[A] = [L][U] \rightarrow [U]\{x\} = \{d\}$ e $[L]\{d\} = \{b\}$



Para casa!



E será que isso funciona?



Aqui $\{x_1\}$ é a coluna 1, $\{x_2\}$ a coluna 2 e por aí vai até a coluna final da matriz $\rightarrow \{x_n\}$



1 - Calcula-se $[U]$ e $[L]$:

2 - Calculamos o vetor $\{d_1\}$ por substituição progressiva que satisfaça: $[L]\{d_1\} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

3 - Calcula-se agora o vetor $\{x_1\}$ por substituição progressiva que satisfaça:

$$[U]\{x_1\} = \{d_1\}$$

4 - Façamos agora $[L]\{d_2\} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$ e obtemos $\{d_2\}$

5 - Calcula-se agora o vetor $\{x_2\}$ por substituição progressiva que satisfaça:

$$[U]\{x_2\} = \{d_2\}$$

6 - Fazemos isso até calcularmos todos os vetores $\{x_n\}$ e usamos cada um desses vetores como uma coluna da matriz $[A]^{-1}$

Métodos para matrizes especiais

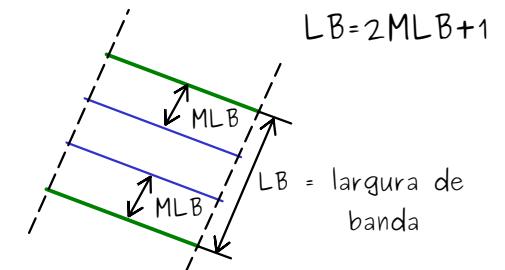
Matrizes simétricas

$$a_{ij} = a_{ji} \rightarrow \text{para } i = 1, n \text{ e } j = 1, n \rightarrow [A] = [A]^T$$

- Matrizes simétricas aparecem naturalmente na descrição de sistemas físicos, como por exemplo no tensor de tensões de um fluido Newtoniano;

Matrizes de banda

$$[A] = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{bmatrix}$$



- Já matrizes de banda surgem espontaneamente no processo de transformação de equações diferenciais em sistemas aproximativos de equações algébricas utilizando técnicas como diferenças finitas, elementos finitos ou volumes finitos;
- Em ambos os casos, quando a matriz dos coeficientes de um sistema linear calha de ser uma matriz simétrica ou uma matriz de banda, podemos conceber algoritmos mais eficientes para resolver o sistema linear que demandam um número menor de operações quando comparados com as técnicas de eliminação vistas até aqui;
- Um tipo muito comum de sistema linear envolvendo matrizes de banda são os sistemas tridiagonais $\rightarrow LB=3$

$$\begin{bmatrix} f_1 & g_1 & & & & \\ e_2 & f_2 & g_2 & & & \\ e_3 & f_3 & g_3 & & & \\ \vdots & \vdots & \ddots & & & \\ & & & & & \\ e_{n-1} & f_{n-1} & g_{n-1} & & & \\ e_n & f_n & & & & \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{Bmatrix} = \begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ \vdots \\ r_{n-1} \\ r_n \end{Bmatrix}$$

Nesse caso trocamos a notação dos coeficientes visando economizar espaço de armazenamento de grandes quantidades de zeros inúteis, principalmente na matriz $[A]$

$$e = [2, \dots, n] \rightarrow n - 1 \text{ termos}$$

$$f = [1, \dots, n] \rightarrow n \text{ termos}$$

$$g = [1, \dots, n - 1] \rightarrow n - 1 \text{ termos}$$

Armazenamos $3n - 2$ termos ao invés de n^2 na matriz $[A]$;

Algoritmo de Thomas para sistemas tridiagonais

Vamos introduzir esse algoritmo com um exemplo: $\begin{bmatrix} 2,04 & -1 & & \\ -1 & 2,04 & -1 & \\ & -1 & 2,04 & -1 \\ & & -1 & 2,04 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \begin{Bmatrix} 40,8 \\ 0,8 \\ 0,8 \\ 200,8 \end{Bmatrix}$ a ideia aqui é aplicar a decomposição L.U a esse sistema;

1 - Começamos então aplicando a técnica de eliminação Gaussiana para decompor $[A]$ em $[L]$ e $[U]$:

a) Calcula-se o fator $f_{21} = \frac{a_{21}}{a_{11}} \rightarrow f_{21} = -0.49$;

b) Multiplicamos a linha 1 por f_{21} e subtraímos o resultado da linha 2: $[U] = \begin{bmatrix} 2.04 & -1 & 0 & 0 \\ 0 & 1.55 & -1 & 0 \\ 0 & -1 & 2.04 & -1 \\ 0 & 0 & -1 & 2.04 \end{bmatrix}$

c) Armazenamos o fator f_{21} sobre uma matriz $[I]$ pré-existente para criar a matriz $[L]$

$$[L] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.49 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) Calcula-se agora o fator $f_{32} = \frac{a_{32}}{a_{22}} \rightarrow -0.645$;

$$[U] = \begin{bmatrix} 2.04 & -1 & 0 & 0 \\ 0 & 1.55 & -1 & 0 \\ 0 & 0 & 1.395 & -1 \\ 0 & 0 & -1 & 2.04 \end{bmatrix}$$

e) Multiplicamos a linha 2 por f_{32} e subtraímos o resultado da linha 3

f) Armazenamos o fator f_{32} sobre a matriz $[L]$ para obter: $[L] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.49 & 1 & 0 & 0 \\ 0 & -0.645 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

g) Calcula-se o fator $f_{43} = \frac{a_{43}}{a_{33}} \rightarrow -0.717$;

h) Multiplicamos a linha 3 por f_{43} e subtraímos o resultado da linha 4 $\rightarrow [U] = \begin{bmatrix} 2.04 & -1 & 0 & 0 \\ 0 & 1.55 & -1 & 0 \\ 0 & 0 & 1.395 & -1 \\ 0 & 0 & 0 & 1.323 \end{bmatrix}$

i) Finalmente, armazenamos o fator f_{43} sobre a matriz $[L]$ para obter: $\rightarrow [L] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.49 & 1 & 0 & 0 \\ 0 & -0.645 & 1 & 0 \\ 0 & 0 & -0.717 & 1 \end{bmatrix}$

2 - Agora, precisamos estimar o vetor $\{d\} \rightarrow [L]\{d\} = \{b\}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.49 & 1 & 0 & 0 \\ 0 & -0.645 & 1 & 0 \\ 0 & 0 & -0.717 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} 40.8 \\ 0.8 \\ 0.8 \\ 200.8 \end{bmatrix} \rightarrow \begin{array}{l} d_1 = 40.8 \\ d_2 = 20.8 \\ d_3 = 14.22 \\ d_4 = 210.996 \end{array}$$

$$\begin{bmatrix} 2.04 & -1 & 0 & 0 \\ 0 & 1.55 & -1 & 0 \\ 0 & 0 & 1.395 & -1 \\ 0 & 0 & 0 & 1.323 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 40.8 \\ 20.8 \\ 14.22 \\ 210.996 \end{bmatrix}$$

3 - Com o valor de $\{d\}$ calculamos $\{x\} \rightarrow [U]\{x\} = \{d\}$

$$\begin{array}{l} x_4 = 159.48 \\ x_3 = 124.538 \\ x_2 = 93.778 \\ x_1 = 65.97 \end{array}$$

Qual a diferença entre o método da decomposição L.U e essa solução?

- A resposta é: NENHUMA;

- O que muda é apenas o algoritmo;

- Sabendo que o sistema é tridiagonal, o algoritmo de solução via decomposição L.U fica um pouco mais simples e recebe o nome de Algoritmo de Thomas;

```

! Decomposição L.U
do k=2,n
e(k)= e(k)/f(k-1)
f(k) = f(k) - e(k)*g(k-1)
end do

!Substituição progressiva = cálculo de {d}
do k=2,n
r(k)=r(k) - e(k)*r(k-1)
end do

!Substituição regressiva = cálculo de {x}
x(n) = r(n)/f(n)
do k= n-1,1
x(k)=(r(k) - g(k)*x(k+1))/f(k)
end do

```

Pseudocódigo em FORTRAN

Decomposição de Cholesky

- Técnica de fatoração (ou decomposição) em que uma matriz simétrica é decomposta em um produto de uma matriz triangular pela sua transposta;
- Útil na solução de sistemas lineares para os quais a matriz dos coeficientes é simétrica;
- Introduzida por André-Louis Cholesky, um cartógrafo francês que serviu nas forças armadas como engenheiro e morreu próximo ao fim da primeira guerra mundial;



- Seja $[L] = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$ e $[L]^T = \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ 0 & l_{22} & l_{23} \\ 0 & 0 & l_{33} \end{bmatrix}$ considere calcular $[L][L]^T$

$$[L][L]^T = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31} \\ l_{11}l_{21} & (l_{21}^2 + l_{22}^2) & (l_{21}l_{31} + l_{22}l_{32}) \\ l_{31}l_{11} & (l_{21}l_{31} + l_{22}l_{32}) & (l_{31}^2 + l_{32}^2 + l_{33}^2) \end{bmatrix}$$

Note que essa matriz é simétrica!

- Podemos então, generalizar e afirmar que se a matriz $[A]$ é simétrica, então a mesma pode ser decomposta em termos do produto de uma matriz triangular inferior por sua transposta: $[A] = [L][L]^T \rightarrow$ para $[A] = [A]^T$
- Além disso, para esse caso 3×3 , temos as seguintes relações entre os termos da matriz $[A]$ e da matriz $[L]$:

$$\begin{aligned} a_{11} &= l_{11}^2, & a_{12} = a_{21} &= l_{11}l_{21}, & a_{13} = a_{31} &= l_{11}l_{31} \\ a_{22} &= l_{21}^2 + l_{22}^2, & a_{23} = a_{32} &= l_{21}l_{31} + l_{22}l_{32} \\ a_{33} &= l_{31}^2 + l_{32}^2 + l_{33}^2 \end{aligned}$$

Invertendo e generalizando

$$\begin{aligned} l_{ki} &= \frac{a_{ki} - \sum_{j=1}^{i-1} l_{ij}l_{kj}}{\ell_{ii}} \rightarrow \text{para } i = 1, 2, \dots, k-1 \\ l_{kk} &= \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2} \end{aligned}$$

Decomposição de Cholesky

- Note que os termos diagonais da matriz $[L]$ dependem de raízes quadradas de coeficientes associados à matriz $[A]$, de tal sorte que podemos experinciar erros de execução caso tenhamos números negativos na matriz $[A]$;
- Entretanto, esse problema não acontece quando a matriz $[A]$, além de simétrica é uma matriz positivo-definida;
- Matrizes positivo-definidas são análogas à números reais positivos, porém para o contexto da álgebra linear;
- Uma definição geral de uma matriz positivo definida é aquela para a qual $\{z\}^T [A] \{z\} > 0$ sendo $\{z\}$ um vetor real arbitrário;
- Como muitas das matrizes simétricas que surgem na engenharia são de fato positivo-definidas, o algoritmo de Cholesky tem vasta aplicação;
- Além disso, a simetria de $[A]$ faz com que precisemos gastar apenas metade do espaço de armazenamento e sistemas lineares associados à matrizes simétricas acabam demandando também menos tempo computacional pelo mesmo motivo;
- Note que se $[A] = [L][L]^T \rightarrow [L][L]^T \{x\} = \{b\}$, chamando $[L]^T \{x\} = \{y\}$ temos $[L]\{y\} = \{b\}$ (1) e $[L]^T \{x\} = \{y\}$ (2)
- Como $[L]$ é uma matriz triangular, podemos usar (1) e (2) para obter a solução por substituição progressiva e regressiva respectivamente;

Pseudocódigo em FORTRAN
para aplicação da decomposição
de Cholesky

```

do k=1,n
do i=1,k-1
soma = 0.0
do j= 1, i-1
soma = soma + a(i,j)*a(k,j)
end do
a(k,i)=(a(k,i)-soma)/a(i,i)
end do
soma = 0.0
do j=1,k-1
soma = soma + a(k,j)**2.0
end do
a(k,k)=sqrt(a(k,k)-soma)
end do

```

Método de Gauss-Seidel

- Esquema iterativo, onde chutamos uma solução inicial e vamos melhorando essa solução em função dos valores da matriz dos coeficientes e do vetor associado aos termos fontes;
- Para entendermos o método, começemos com um sistema 3×3 : \rightarrow
- Isolando x_1, x_2, x_3 desse nosso sisteminha de referência, temos:

$$\begin{aligned}x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3) / a_{11} \\x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3) / a_{22} \\x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2) / a_{33}\end{aligned}$$

Transformando em
um esquema iterativo
(opção 1)

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3\end{aligned}$$

Transformando em
um esquema iterativo
(opção 2)

$$\begin{aligned}x_1^{j+1} &= (b_1 - a_{12}x_2^j - a_{13}x_3^j) / a_{11} & j = 0, N_{iter} \\x_2^{j+1} &= (b_2 - a_{21}x_1^{j+1} - a_{23}x_3^j) / a_{22} & \text{número de iterações} \\x_3^{j+1} &= (b_3 - a_{31}x_1^{j+1} - a_{32}x_2^{j+1}) / a_{33}\end{aligned}$$

$x_1^0, x_2^0, x_3^0 \rightarrow$ chute inicial

Método de Gauss-Seidel

$$\begin{aligned}x_1^{j+1} &= (b_1 - a_{12}x_2^j - a_{13}x_3^j) / a_{11} & j = 0, N_{iter} \\x_2^{j+1} &= (b_2 - a_{21}x_1^j - a_{23}x_3^j) / a_{22} & \downarrow \text{número de iterações} \\x_3^{j+1} &= (b_3 - a_{31}x_1^j - a_{32}x_2^j) / a_{33}\end{aligned}$$

$x_1^0, x_2^0, x_3^0 \rightarrow$ chute inicial

Iteração de Jacobi

Primeira Iteração

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3) / a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3) / a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2) / a_{33}$$

Segunda Iteração

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3) / a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3) / a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2) / a_{33}$$

(a)

(b)

- Embora, em alguns casos o método de Jacobi possa ser útil, o método de Gauss-Seidel costuma ser o preferido pelo fato de já ir substituindo valores mais atualizados gradativamente conforme avançamos na solução iterativa do sistema;
- Para acompanhar a convergência do método, avaliamos se:

$$\max(|\varepsilon_{a,i}|) = \max \left(\left| \frac{x_i^{j+1} - x_i^j}{x_i^j} \right| \right) \times 100\% < tol$$

Convergência do Método de Gauss-Seidel

- Para falarmos sobre a convergência do método de Gauss-Seidel, vamos voltar a um problema mais simples: raízes de equações;
- Um análogo simples ao método de Gauss-Seidel no campo de determinação de zeros de funções é o método da iteração de ponto fixo;
- Considere uma função qualquer $f(x)$ cuja raiz queremos encontrar. Nesse caso, podemos manipular $f(x)$ para isolar x e obter esse valor x como uma função $g(x)$, ou seja: $x = g(x)$;
- Exemplo:
$$f(x) = x + \frac{x^2 - 3}{\sqrt{x}} = 0 \rightarrow x = -\frac{(x^2 - 3)}{\sqrt{x}} = g(x)$$
- Nesse campo, pequenas "roubadas" também podem ser aplicadas, como: $f(x) = \sin(x) = 0$ se somarmos x dos dois lados, obtemos: $x = x + \sin(x) = g(x)$
- A partir do momento em que conseguimos isolar x e escrevê-lo em termos de $g(x)$, podemos obter a raiz que buscamos de forma iterativa: $x_{i+1} = g(x_i) \rightarrow$ para $i = 0, 1, \dots, N_{iter} - 1$

- Suponha agora que a solução verdadeira seja $x_r = g(x_r) \rightarrow x_r - x_{i+1} = g(x_r) - g(x_i)$
- Agora, se $g(x)$ e $g'(x)$ são contínuas no intervalo $a \leq x \leq b$, então existe um valor de $x = \zeta$ no intervalo $[a, b]$ para o qual:

$$g'(\zeta) = \frac{g(b) - g(a)}{b - a}$$

Teorema do valor médio para derivadas
- Agora, se fazemos $a = x_i$ e $b = x_r$, podemos então escrever: $g(x_r) - g(x_i) = (x_r - x_i)g'(\zeta)$
- Mas se $x = g(x)$, então:

$x_r - x_{i+1} = (x_r - x_i)g'(\zeta)$

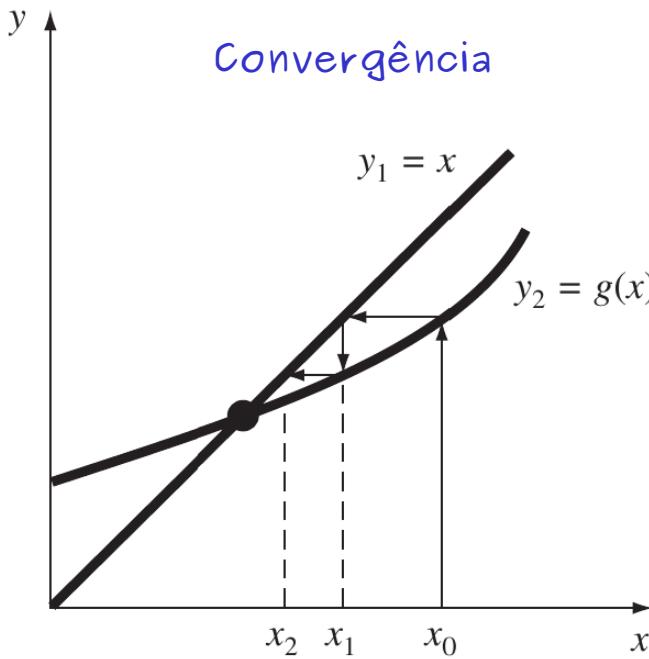
$x_i \leq \zeta \leq x_r$

aqui estamos usando a ideia central do método da iteração de ponto fixo: $x_{i+1} = g(x_i)$
- Se o erro verdadeiro para a i -ésima iteração for definido como: $\varepsilon_{r,i} = x_r - x_i \rightarrow \varepsilon_{r,i+1} = \varepsilon_{r,i}g'(\zeta)$

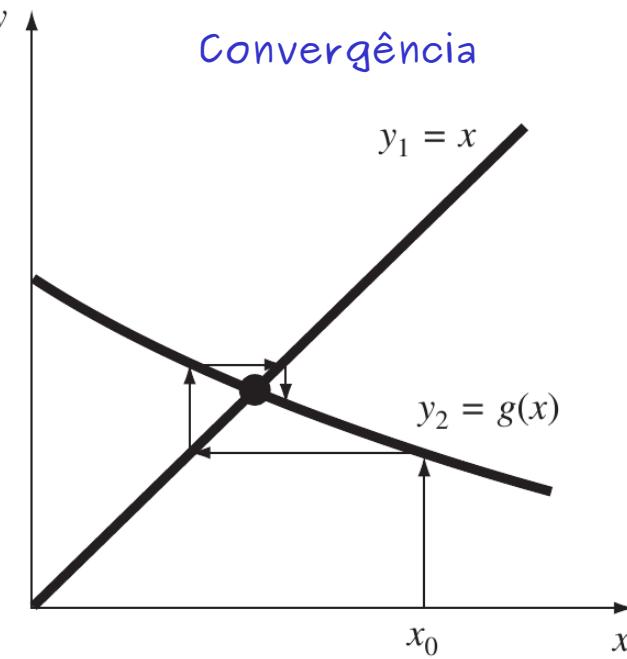
↓

$|g'(\zeta)| < 1 \rightarrow$ erros diminuem a cada iteração
 $|g'(\zeta)| > 1 \rightarrow$ erros aumentam a cada iteração
- Note que no caso convergente, os erros vão diminuindo proporcionalmente a cada iteração, por isso dizemos que o método da iteração de ponto fixo é linearmente convergente;
- Podemos enxergar também essa convergência visualmente. Se $x_{i+1} = g(x_i)$, a solução converge para $y_1 = y_2$

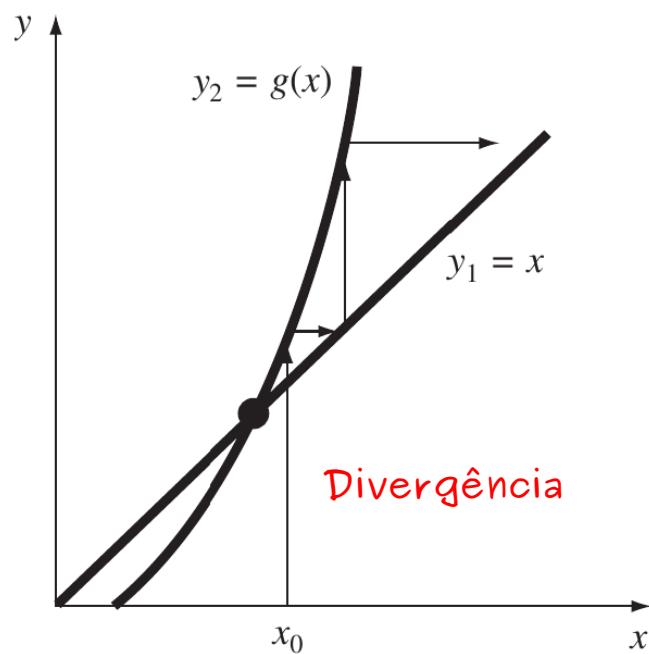
$y_1 = x$ $y_2 = g(x)$



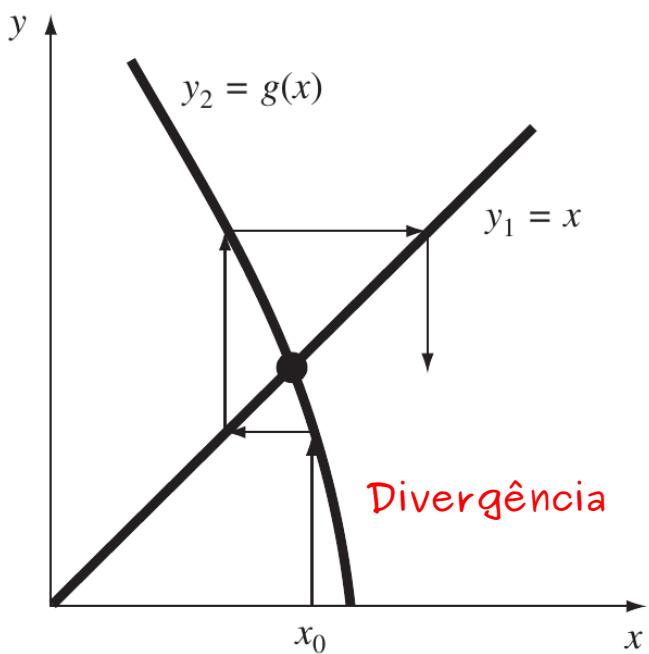
(a)



(b)



(c)

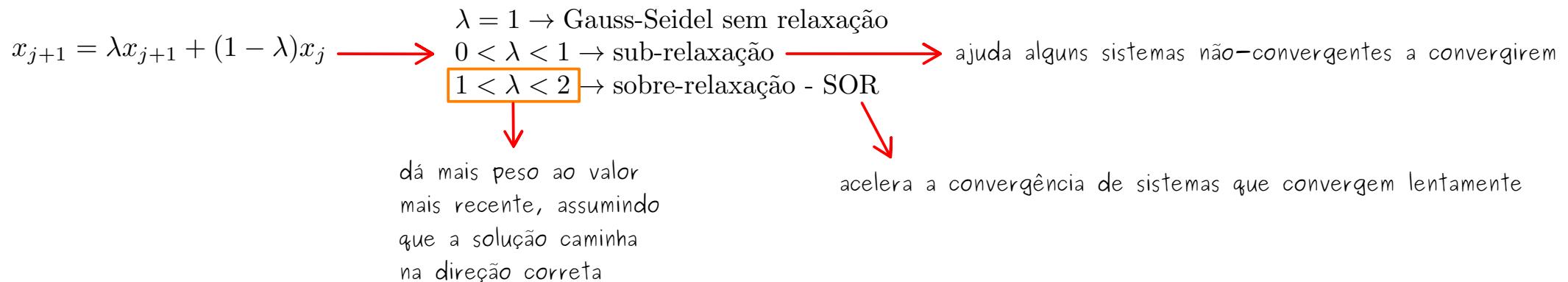


(d)

E o que tudo isso tem a ver com a convergência do método de Gauss-Seidel?

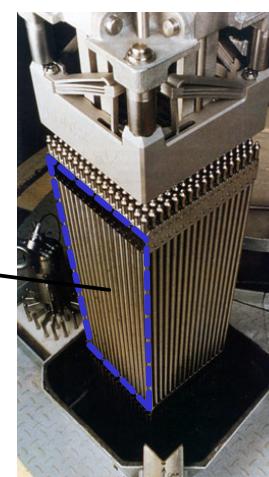
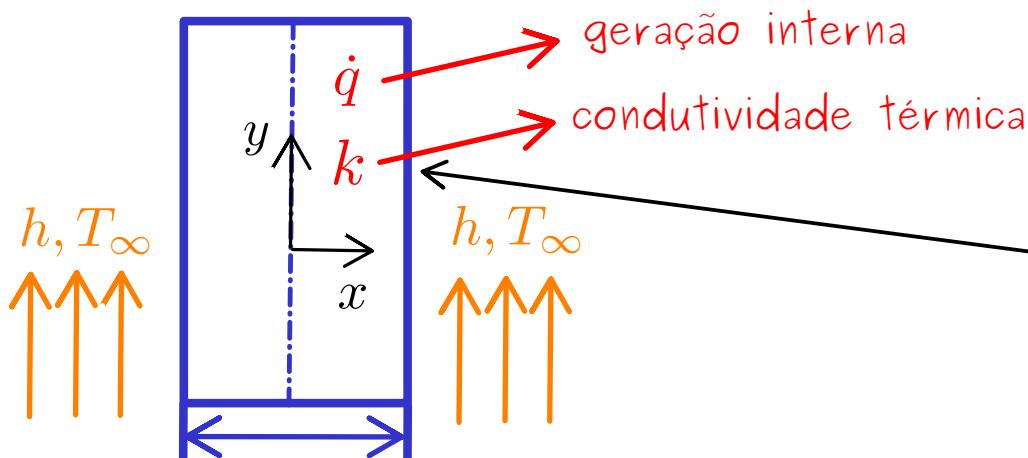
- Considere um sistema linear simples 2×2 : $u(x, y) = 0$ $v(x, y) = 0$ → aqui u e v são funções de 2 variáveis que compõe o próprio sistema e são análogas ao caso 1D para $g(x)$;
- No caso 1D, a iteração de ponto fixo apresenta convergência em função do valor de $g'(\zeta)$;
- No caso do sistema 2×2 podemos estender esse racional para avaliar a convergência do método de Gauss-Seidel;
- Nesse contexto, o método será convergente se: $\left| \frac{\partial u}{\partial x} \right| + \left| \frac{\partial u}{\partial y} \right| < 1$ e $\left| \frac{\partial v}{\partial x} \right| + \left| \frac{\partial v}{\partial y} \right| < 1$
- Para esse problema 2×2 temos:
$$x_1^{i+1} = \frac{b_1}{a_{11}} - \frac{a_{12}}{a_{11}}x_2^i = u$$
$$x_2^{i+1} = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}}x_1^i = v$$
→
$$\begin{aligned} \frac{\partial u}{\partial x_1} &= 0, & \frac{\partial u}{\partial x_2} &= -\frac{a_{12}}{a_{11}} \\ \frac{\partial v}{\partial x_1} &= -\frac{a_{21}}{a_{22}}, & \frac{\partial v}{\partial x_2} &= 0 \end{aligned}$$
- Para convergência do método, os elementos da diagonal devem ser maiores que os elementos não-diagonais em cada linha: diagonal dominante!
$$\left| \frac{a_{12}}{a_{11}} \right| < 1 \text{ e } \left| \frac{a_{21}}{a_{22}} \right| < 1$$
- Para um sistema com n equações, podemos generalizar o critério de convergência: $|a_{ii}| > \sum_{j=1}^n |a_{ij}|$
- Finalmente, em alguns contextos de lenta convergência podemos tentar acelerar o processo usando uma variante do método de Gauss-Seidel que aplica o conceito de relaxação;

- No método de Gauss-Seidel com relaxação, após o cálculo do valor atual de x_{j+1} aplicamos uma correção do tipo:



Exemplo prático e sugestão de programa para casa

Determinando a distribuição de temperatura no combustível sólido de um reator nuclear
 (Eng. Mecânica/Nuclear)



Hipóteses de modelagem

- simetria adiabática;
- caso 1D;
- Regime transiente

Eq. da condução transiente 1D com geração interna:

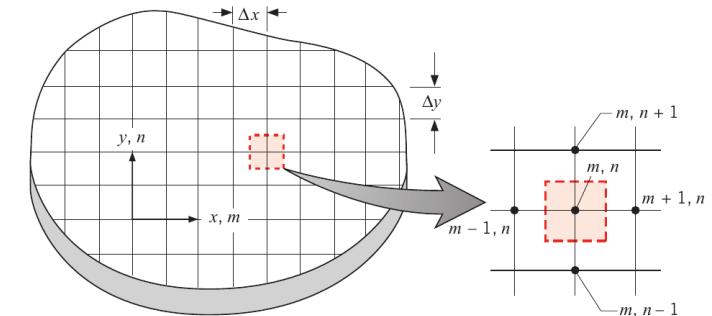
$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} + \frac{\dot{q}}{\rho C_p} \quad (1)$$

difusividade térmica

- Vamos resolver esse problema aplicando o método das diferenças finitas, nos quais as derivadas serão aproximadas por termos de primeira ordem, baseadas na ideia de série de Taylor;

$$T(x, t) \rightarrow \frac{\partial T}{\partial t} \approx \frac{T_m^{p+1} - T_m^p}{\Delta t}$$

instante de tempo
posição no espaço
passo de tempo

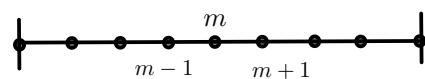


$$\left(\frac{\partial T}{\partial x} \right)_{m+\frac{1}{2}} \approx \frac{T_{m+1} - T_m}{\Delta x} \quad \text{e} \quad \left(\frac{\partial T}{\partial x} \right)_{m-\frac{1}{2}} \approx \frac{T_m - T_{m-1}}{\Delta x}$$

$$\rightarrow \left(\frac{\partial^2 T}{\partial x^2} \right)_m \approx \frac{\partial T / \partial x|_{m+1/2} - \partial T / \partial x|_{m-1/2}}{\Delta x} = \frac{T_{m+1} + T_{m-1} - 2T_m}{\Delta x^2}$$

- Usando essas aproximações podemos reescrever (1) em sua versão discretizada como:

$$\frac{T_m^{p+1} - T_m^p}{\Delta t} = \alpha \left(\frac{T_{m+1}^Q + T_{m-1}^Q - 2T_m^Q}{\Delta x^2} \right) + \frac{\dot{q}}{\rho C_p}$$



- No esquema explícito a solução parte de um ponto de partida (condição inicial) e evolui por substituição progressiva;
- Nesse esquema não há necessidade de resolvemos um sistema linear;
- Essa solução pode no entanto ser instável para maiores passos de tempo;
- O esquema implícito acopla a informação num dado nó com seus vizinhos instantaneamente, o que demanda que o problema seja resolvido como um sistema linear;
- É um método mais estável numericamente e aceita maiores passos de tempo quando comparado com o esquema explícito;

Aqui temos 3 opções para o instante Q no qual iremos avaliar os termos do lado direito da equação:

- Opção 1 → $Q = p \rightarrow$ esquema explícito;
 Opção 2 → $Q = p + 1 \rightarrow$ esquema implícito;
 Opção 3 → $p < Q < p + 1 \rightarrow$ esquema semi-implícito;

- O esquema semi-implícito demanda a solução de um sistema linear, assim como o método implícito, mas permite a inclusão de um parâmetro numérico para controle da convergência do sistema;
- Para ficar mais claro como trabalhamos o método semi-implícito, vamos reescrever nossa equação discretizada como:

$$T_m^{p+1} - T_m^p = Fo \left(T_{m+1}^Q + T_{m-1}^Q - 2T_m^Q \right) + A \rightarrow \text{com } Fo = \frac{\alpha \Delta t}{\Delta x^2} \text{ e } A = \frac{\dot{q}}{\rho C_p} \rightarrow \text{termo fonte}$$

↓
número de Fourier

- No esquema semi-implícito, a equação acima é reescrita como:

$$T_m^{p+1} - T_m^p = Fo \left[\beta \left(T_{m+1}^p + T_{m-1}^p - 2T_m^p \right) + (1 - \beta) \left(T_{m+1}^{p+1} + T_{m-1}^{p+1} - 2T_m^{p+1} \right) \right] + A$$

↓
Parâmetro numérico de controle de convergência

$\beta = 0 \rightarrow$ esquema implícito
 $\beta = 1 \rightarrow$ esquema explícito
 $0 < \beta < 1 \rightarrow$ esquema semi-implícito
 $\beta = \frac{1}{2} \rightarrow$ método de Crank-Nicholson

- Vamos agora resolver o nosso problema específico, relacionado à determinação da distribuição de temperatura no interior do combustível sólido de um reator nuclear;

- Para isso, vamos considerar uma malha simples de 7 nós:



- O ponto de partida aqui consiste na organização das equações discretizadas para os nós internos. Vamos para isso considerar a utilização do esquema implícito;
- Os nós de contorno serão tratados posteriormente, no momento da aplicação das condições de contorno;

Nó Eq. discretizada (esquema implícito)

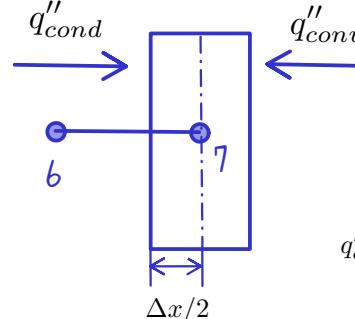
$$\begin{aligned} 2 &\rightarrow (1 + 2Fo)T_2^{p+1} - Fo(T_1^{p+1} + T_3^{p+1}) = T_2^p + A \\ 3 &\rightarrow (1 + 2Fo)T_3^{p+1} - Fo(T_2^{p+1} + T_4^{p+1}) = T_3^p + A \\ 4 &\rightarrow (1 + 2Fo)T_4^{p+1} - Fo(T_3^{p+1} + T_5^{p+1}) = T_4^p + A \\ 5 &\rightarrow (1 + 2Fo)T_5^{p+1} - Fo(T_4^{p+1} + T_6^{p+1}) = T_5^p + A \\ 6 &\rightarrow (1 + 2Fo)T_6^{p+1} - Fo(T_5^{p+1} + T_7^{p+1}) = T_6^p + A \end{aligned}$$

Equacionando todos os termos do balanço de energia para o nós 7 e reorganizando os termos, obtemos:

$$(1 + 2Fo + 2FoBi)T_7^{p+1} - 2FoT_6^{p+1} = 2FoBiT_\infty + \frac{Fo\dot{q}\Delta x^2}{k} + T_7^p$$

$$Bi = \frac{h\Delta x}{k} \rightarrow \text{número de Biot}$$

Para os nós de contorno, vamos aplicar um balanço de energia para cada nó, começando pelo nó 7:



$$\begin{aligned} V &= \frac{A\Delta x}{2} \\ q''_{cond} \times A + q''_{conv} \times A + \dot{q} \times V &= \dot{E}_a \\ q''_{cond} = k \left(\frac{T_6^{p+1} - T_7^{p+1}}{\Delta x} \right) & \\ \dot{E}_a = \frac{\rho C_p}{\Delta t} (T_7^{p+1} - T_7^p) A\Delta x & \\ q''_{conv} = h (T_\infty - T_7^{p+1}) & \end{aligned}$$

- Para o nó 1, a análise é mais simples, pois assumimos simetria adiabática, o que equivale à pensar nesse nó como se fosse um nó interno que possui o mesmo vizinho pela esquerda e pela direita. Aplicando esse raciocínio para a equação geral de um nó interno qualquer, temos: $(1 + 2Fo)T_1^{p+1} - 2FoT_2^{p+1} = T_1^p + A$
- Finalmente, podemos escrever todas essas equações na forma de um sistema linear do tipo:

$$\left(\begin{array}{cccccc} (1 + 2Fo) & -2Fo & 0 & 0 & 0 & 0 \\ -Fo & (1 + 2Fo) & -Fo & 0 & 0 & 0 \\ 0 & -Fo & (1 + 2Fo) & -Fo & 0 & 0 \\ 0 & 0 & -Fo & (1 + 2Fo) & -Fo & 0 \\ 0 & 0 & 0 & -Fo & (1 + 2Fo) & -Fo \\ 0 & 0 & 0 & 0 & -Fo & (1 + 2Fo + 2FoBi) \\ 0 & 0 & 0 & 0 & 0 & -2Fo \end{array} \right) \cdot \begin{pmatrix} T_1^{p+1} \\ T_2^{p+1} \\ T_3^{p+1} \\ T_4^{p+1} \\ T_5^{p+1} \\ T_6^{p+1} \\ T_7^{p+1} \end{pmatrix} = \begin{pmatrix} T_1^p + A \\ T_2^p + A \\ T_3^p + A \\ T_4^p + A \\ T_5^p + A \\ T_6^p + A \\ T_7^p + Fo(2BiT_\infty + \frac{\dot{q}\Delta x^2}{k}) \end{pmatrix}$$

- A solução desse mesmo problema, na ausência de termos de geração, ou seja, quando $\dot{q} = 0$ é possível de ser obtida por meio da técnica analítica de separação de variáveis;

- A solução exata é dada por:

$$\theta(x^*, Fo) = \sum_{n=1}^{\infty} \left[\frac{4 \sin(\lambda_n)}{2\lambda_n + \sin(2\lambda_n)} \right] \cos(\lambda_n x^*) e^{-\lambda_n^2 Fo} \quad \text{com} \quad \lambda_n = Bi \cot(\lambda_n)$$

$$\frac{T - T_{\infty}}{T_i - T_{\infty}} = \frac{x}{L}$$

- A dedução dessa solução encontra-se disponível no canal do Youtube: Ciência e Brisa



- Escreva primeiramente um programa que resolva numericamente o problema, usando o método das diferenças finitas para o caso sem geração interna e valide sua solução numérica com base em uma comparação com a solução exata. Utilize o algoritmo de Thomas para sistemas tridiagonais na hora de resolver o seu problema e um esquema implícito de evolução no tempo;
- Em seguida, acrescente o termo de geração interna à solução solução numérica, usando como base a malha e passo de tempo ajustados no passo anterior para validação da solução no limite assintótico em que $\dot{q} \rightarrow 0$
- Teste diferentes formas de representação visual da sua solução (perfils, campos de cores, animações, etc.);
- Pesquise valores realistas para o problema, tendo em vista o contexto de aplicação aqui desenvolvido;

Como esses métodos aparecem em contextos modernos de Engenharia?

- Exemplo: OpenFOAM

→ Open ∇ FOAM®



OpenFOAM	Método Clássico	Uso Típico
smoothSolver	Gauss-Seidel suavizado, Symmetric Gauss-Seidel e Diagonal Incomplete Cholesky (DIC)	Iterativo simples; atua como suavizador em métodos multigrid.
PCG	Gradiente Conjugado (pré-cond.)	Bom para sistemas simétricos e definidos positivos.
PBiCG	Gradiente Bi-Conjugado com pré-condicionadores como DILU (Diagonal Incomplete LU) ou DIC	Para sistemas não simétricos; pode apresentar oscilações.
GAMG	Multigrid Algebraico	Muito eficiente em grandes malhas; frequentemente usado para pressão.
diagonal	Resolução direta (diagonal)	Apenas para sistemas triviais e desacoplados.

Variantes de
gradientes
conjugados

↓
Veremos isso
no próximo
módulo: otimização