Métodos para solucionar sistemas lineares

11 de abril de 2017

Raquel Lopes de Oliveira
Vinícius Campos Tinoco Ribeiro
Vitor Rodrigues Greati

Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte
Natal-RN

Agenda

Objetivo
- Contexto
- Descrição

Desenvolvimento
- LU
- Cholesky
- Jacobi
- Gauss-Seidel

Resultados
- Métodos direto
- Métodos Indireto

Resolver um sistema linear possível e determinado $Ax = b$, ou seja, encontrar o vetor x que, transformado por A, produz b.

Implementações dos métodos diretos e indiretos para solucionar sistemas lineares possíveis e determinados:

- LU com pivotamento parcial
- Cholesky
- Jacobi
- Gauss-Seidel

LU com matriz de pivotação I

Objetivo

Fatorar uma matriz A em $PA = LU$ armazenando os efeitos da pivotação parcial em uma matriz P, para resolver novos sistemas lineares de mesma A usando a **mesma fatoração com pivotação**.

Como sabemos, a pivotação parcial consiste em uma troca de linhas a cada iteração (considerando os casos em que não trocar uma linha é trocar uma linha consigo própria). Essa operação possui uma matriz associada, chamada de **matriz de permutação**.

Lembrando o processo da fatoração LU, temos uma matriz A, a partir da qual calculamos a primeira transformada de Gauss, L_1. Se não quiséssemos realizar a pivotação, faríamos simplesmente

$$U_1 = L_1 A$$

LU com matriz de pivotação II

Entretanto, quando a pivotação é desejada, utilizamos uma matriz P_1 de permutação, e U_1 é, então, dada por

$$U_1 = L_1 P_1 A$$

onde L_1 é computada sobre a matriz $P_1 A$.

Assim, considerando as demais iterações, podemos escrever U da forma

$$U = L_{m-1} P_{m-1} \dots L_2 P_2 L_1 P_1 A$$

Contudo, o produto

$$L_{m-1} P_{m-1} \dots L_2 P_2 L_1 P_1$$

não é necessariamente triangular superior, e, com isso, não temos ainda uma fatoração LU. Porém, podemos, para cada L_k,

LU com matriz de pivotação III

encontrar uma L'_k, que é uma permutação da subdiagonal da primeira, escrita como

$$L'_k = P_{m-1} \dots P_{k+1} L_k P_{k+1}^{-1} \dots P_{m-1}^{-1}$$

O cálculo das L'_k nos permite escrever

$$U = (L'_{m-1} \dots L'_1)(P_{m-1} \dots P_1)A$$

Fazendo

$$L = (L'_{m-1} \dots L'_1)^{-1} \text{ e}$$

$$P = (P_{m-1} \dots P_2 P_1),$$

chegamos à fatoração

$$PA = LU$$

LU com matriz de pivotação IV

Pensando no sistema linear , note que

$$Ax = b \iff PAx = Pb \iff LUx = Pb$$

Fazendo $y = Ux$, resolvemos facilmente $Ly = Pb$ e, depois, $Ux = y$, assim como na fatoração LU sem pivotação.

Dessa maneira, com L, U e P, podemos resolver *qualquer* sistema linear que tenha a matriz A como coeficientes, com erros de arrendondamento controlados graças à pivotação.

LU melhorada

Input: Matriz $A_{n\times n}$
Output: Matrizes U, L, P
begin
  $U \leftarrow A$
  $L \leftarrow I_{n\times n}$
  $P \leftarrow I_{n\times n}$
  for $k = 1$ **to** $n\text{-}1$ **do**
    $i \leftarrow argmax_{k\leq m\leq n}|u_{mk}|$
    $u_{k,k:n} \leftrightarrow u_{i,k:n}$
    $l_{k,1:k-1} \leftrightarrow l_{i,1:k-1}$
    $p_{k,:} \leftrightarrow p_{i,:}$
    for $j = k + 1$ **to** n **do**
      $l_{jk} \leftarrow u_{jk}/u_{kk}$
      $u_{j,k:n} \leftarrow u_{j,k:n} - l_{jk}u_{k,k:n}$
    end
  end
end

Figura: Fatoração LU, seguindo o algoritmo no livro "Numerical Linear Algebra", de Trefethen.

Cholesky - Intuição I

Vamos usar uma $A_{3\times3}$ para obter uma intuição sobre o funcionamento do algoritmo de Cholesky. O objetivo é encontrar R tal que

$$A = RR^T$$

Ou seja, queremos o seguinte:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} r_{11} & 0 & 0 \\ r_{21} & r_{22} & 0 \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} r_{11} & r_{21} & r_{31} \\ 0 & r_{22} & r_{32} \\ 0 & 0 & r_{33} \end{pmatrix}$$

Multiplicando o lado direito, temos:

$$\begin{pmatrix} r_{11}^2 & r_{11} \cdot r_{21} & r_{11} \cdot r_{31} \\ r_{11} \cdot r_{21} & r_{21}^2 + r_{22}^2 & r_{21} \cdot r_{31} + r_{22} \cdot r_{32} \\ r_{11} \cdot r_{31} & r_{21} \cdot r_{31} + r_{22} \cdot r_{32} & r_{31}^2 + r_{32}^2 + r_{33}^2 \end{pmatrix}$$

Cholesky - Intuição II

Para encontrar R, temos de encontrar todos os r_{ij}. Agora, sabemos como: a partir de elementos de A, já conhecidos, e de elementos da própria R!

Começamos por r_{11}. Comparando com a_{11}, temos a relação:

$$a_{11} = r_{11}^2 \iff r_{11} = \sqrt{a_{11}}$$

Para os outros elementos da primeira coluna ($j = 1, 2 \leq i \leq 3$), note que:

$$a_{i1} = r_{11} \cdot r_{i1} \iff r_{i1} = a_{i1}/r_{11}$$

Note que temos, agora, os ingredientes para calcular r_{22}:

$$a_{22} = r_{21}^2 + r_{22}^2 \iff r_{22} = \sqrt{a_{22} - r_{21}^2}$$

Com r_{22}, podemos calcular r_{32}:

$$a_{32} = r_{21} \cdot r_{31} + r_{22} \cdot r_{32} \iff r_{32} = (a_{32} - r_{21} \cdot r_{31})/r_{22}$$

Cholesky - Intuição III

Finalmente, só nos resta calcular r_{33}:

$$a_{33} = r_{31}^2 + r_{32}^2 + r_{33}^2 \iff r_{33} = \sqrt{a_{33} - r_{31}^2 + r_{32}^2}$$

E para matrizes $A_{n \times n}$? Como generalizar? Observando os cálculos anteriores, o algoritmo geral é:

1. Faça $r_{11} = \sqrt{a_{11}}$
2. Calcule a primeira coluna de R, fazendo $r_{i1} = a_{i1}/r_{11}$
3. Para cada coluna $2 \leq j \leq n-1$
   3.1 Calcule o elemento da diagonal: $r_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} r_{jk}^2}$
   3.2 Calcule os demais elementos da coluna j:
   $r_{ij} = (a_{ij} - \sum_{k=1}^{j-1} r_{jk} \cdot r_{ik})/r_{jj}$
4. Para a última coluna de R, só precisamos calcular r_{nn}:

$$r_{nn} = \sqrt{a_{nn} - \sum_{k=1}^{n-1} r_{nk}^2}$$

Cholesky - Algoritmo

Input: Matriz $A_{n \times n}$, simétrica positiva definida
Output: Matriz L, tal que $A = LL^T$
begin
  $l_{11} \leftarrow \sqrt{a_{11}}$
  for $j = 2$ **to** n-1 **do**
    $l_{j1} \leftarrow a_{j1}/l_{11}$
  end
  for $i = 2$ **to** n-1 **do**
    $l_{ii} \leftarrow \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$
    for $j = i + 1$ **to** n **do**
      $l_{ji} \leftarrow \frac{a_{ji} - \sum_{k=1}^{i-1} l_{jk} l_{ik}}{l_{ii}}$
    end
  end
  $l_{nn} \leftarrow \sqrt{a_{nn} - \sum_{k=1}^{n-1} l_{nk}^2}$
end

Figura: Método de Cholesky, implementado a partir do algoritmo no livro "Numerical Analysis", de Burden.

Métodos indiretos

Partimos de

$$Ax = b.$$

Fazendo

$$A = M + N$$

e supondo M invertível, temos:

$$(M+N)x = b$$
$$x = M^{-1}b - M^{-1}Nx$$

Daqui podemos tirar um método iterativo tal que:

Iteração

Escolher um vetor inicial: x^0
Iterar: $x^{k+1} = M^{-1}b - M^{-1}Nx^k$

Seja: $M = D$ e $N = L + U$

Iteração - Método Iterativo

Escolher um vetor inicial: x^0

Iterar: $x^{k+1} = M^{-1}b - M^{-1}Nx^k$

Iteração - Jacobi

Escolher um vetor inicial: x^0

Iterar: $x^{k+1} = D^{-1}b - D^{-1}(L + U)x^k$

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

Input: Matriz $A_{n \times n}$, vetor b, vetor x^e, ε
Output: Aproximação para x
checkConvergence();
begin
    repeat
        $x^{aux} \leftarrow x^e$
        $x^s \leftarrow Tx^e$
        $x^e \leftarrow x^s$
    until $||x^e - x^{aux}|| < \varepsilon$;
    return x^s
end

Figura: Método de Jacobi

Seja: $M = L + D$ e $N = U$

Iteração - Método Iterativo

Escolher um vetor inicial: x^0
Iterar: $x^{k+1} = M^{-1}b - M^{-1}Nx^k$

Iteração - Gauss

Escolher um vetor inicial: x^0
Iterar: $x^{k+1} = D^{-1}b - D^{-1}Lx^{k+1} - D^{-1}Ux^k$

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

Gauss-Seidel
Método Indireto

Input: Matriz $A_{n\times n}$, vetor b, vetor x^e, ε
Output: Aproximação para x
begin
  checkConvergence(A);
  repeat
    $x^{aux} \leftarrow x^e$
    for $i = 1$ **to** n **do**
      $$x_i^s \leftarrow -\frac{\sum_{j=1, j\neq i}^{i} a_{ij}x_j^s + \sum_{j=i+1}^{n} a_{ij}x_j^e - b_i}{a_{ii}}$$
    end
    $x^e \leftarrow x^s$
  until $||x^e - x^{aux}|| < \varepsilon$;
  return x^s
end

Figura: Pseudo código: Gauss-Seidel

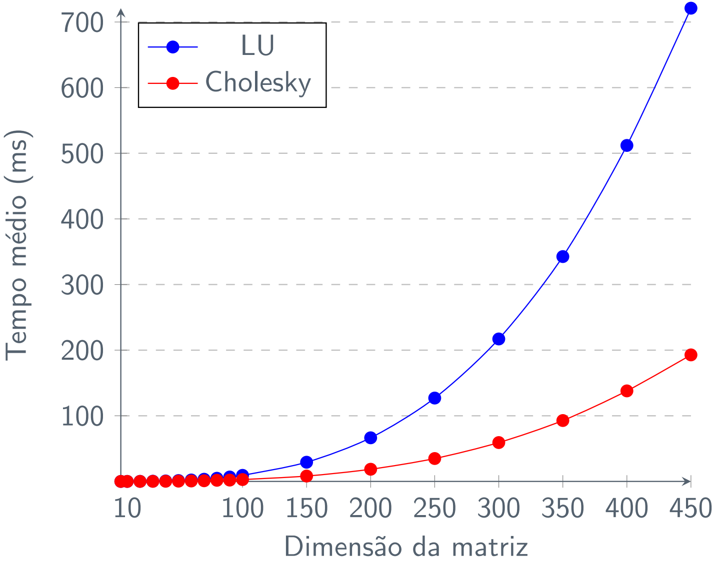
Dimensão	LU	Cholesky	Didático LU	Didático Cholesky
5x5	0.015286	0.012809	0.050517	0.040998
10x10	0.038966	0.026361	0.464253	0.869828
20x20	0.134158	0.05492	16.0118	6.89225
30x30	0.360649	0.128106	32.9281	-
40x40	0.747341	0.251166	102.131	103.648
50x50	1.33657	0.432224	249.097	251.084
60x60	2.20472	0.690858	514.509	517.351
70x70	3.3237	1.023	946.277	-
80x80	4.82391	1.72569	1616.41	1696.81
90x90	6.73681	2.03473	2620.9	2599

Tabela: Tempo de execução em milissegundos

Resultados
Métodos diretos

Dimensão	LU	Cholesky	Didático LU	Didático Cholesky
100x100	9.25865	2.85575	3970.46	3950.56
150x150	29.2534	8.17796	20035.2	-
200x200	66.4677	18.548	69281.5	67986.9
250x250	127.024	34.8935	164388	-
300x300	217.056	59.2172	343029	369484
350x350	342.593	92.778	636412	630656
400x400	511.915	137.945	+15min	+15min
450x450	721.025	192.795	+15min	+15min

Tabela: Tempo de execução em milissegundos



LU
Cholesky
Tempo médio (ms)
700
600
500
400
300
200
100
10
100
150
200
250
300
350
400
450
Dimensão da matriz

Matriz	Vetor Inicial	Jacobi	Iterações	Gauss-Seidel	Iterações
1 (3x3)	0	0.070105	4	0.011437	3
2 (3x3)	0	0.037312	4	0.009241	3
3 (4x4)	0	0.08077	6	0.019188	5
4 (2x2)	0	0.029007	5	0.01726	3
5 (2x2)	0	0.041	4	0.012726	3

Tabela: Tempo de execução em milissegundos e iterações

Matriz	Vetor Inicial	Jacobi	Iterações	Gauss-Seidel	Iterações
1 (3x3)	5	0.030223	5	0.005229	4
2 (3x3)	5	0.018607	6	0.003728	3
3 (4x4)	5	0.044471	7	0.007003	5
4 (2x2)	5	0.013156	6	0.003506	4
5 (2x2)	5	0.011672	6	0.003193	4

Tabela: Tempo de execução em milissegundos e iterações

Resultados
Métodos indiretos

Matriz	Vetor Inicial	Jacobi	Iterações	Gauss-Seidel	Iterações
1 (3x3)	10	0.091356	6	0.017651	4
2 (3x3)	10	0.063236	7	0.014118	4
3 (4x4)	10	0.123886	9	0.027253	6
4 (2x2)	10	0.044007	7	0.012765	5
5 (2x2)	10	0.039837	7	0.011702	5

Tabela: Tempo de execução em milissegundos e iterações

Matriz	Vetor Inicial	Jacobi	Iterações	Gauss-Seidel	Iterações
1 (3x3)	50	0.101314	7	0.019505	5
2 (3x3)	50	0.072548	9	0.016688	5
3 (4x4)	50	0.138285	11	0.034124	8
4 (2x2)	50	0.052632	9	0.012408	5
5 (2x2)	50	0.04776	9	0.013516	6

Tabela: Tempo de execução em milissegundos e iterações

Matriz	Vetor Inicial	Jacobi	Iterações	Gauss-Seidel	Iterações
1 (3x3)	100	0.101298	8	0.022492	6
2 (3x3)	100	0.115226	9	0.017085	5
3 (4x4)	100	0.175004	13	0.034043	8
4 (2x2)	100	0.087302	10	0.014584	6
5 (2x2)	100	0.075164	10	0.013837	6
1 (3x3)	1000	0.116178	10	0.022372	6
2 (3x3)	1000	0.090088	12	0.020001	6
3 (4x4)	1000	0.177455	16	0.041847	10
4 (2x2)	1000	0.064203	12	0.016232	7
5 (2x2)	1000	0.059864	12	0.015376	7
1 (3x3)	100000	0.068154	14	0.013407	9
2 (3x3)	100000	0.052454	17	0.010834	8
3 (4x4)	100000	0.109593	25	0.026044	15
4 (2x2)	100000	0.0362	17	0.009444	10
5 (2x2)	100000	0.034481	17	0.009338	10
1 (3x3)	1000000	0.049761	16	0.010497	10
2 (3x3)	1000000	0.060622	19	0.00893	9
3 (4x4)	1000000	0.102968	30	0.021374	17
4 (2x2)	1000000	0.040368	20	0.007645	11
5 (2x2)	1000000	0.047839	20	0.006955	11

Tabela: Tempo de execução em milissegundos e iterações