CAPÍTULO 3

DECOMPOSIÇÃO LU COM PIVOTEAMENTO PARCIAL: ASPECTOS TEÓRICOS E COMPUTACIONAIS

https://doi.org/10.22533/at.ed.1232517103

Negrini, Eloisa de M.
Universidade Estadual do Oeste do Parana´/Licenciatura em Matema´tica, Centro de Engenharias e Cieˆncias Exatas/Foz do Iguaçu, PR

Rezak, Fahir L. C.
Universidade Estadual do Oeste do Parana´/Licenciatura em Matema´tica, Centro de Engenharias e Cieˆncias Exatas/Foz do Iguaçu, PR

Portilho Jr, Emídio S.
Universidade Estadual do Oeste do Parana´/Licenciatura em Matema´tica, Centro de Engenharias e Cieˆncias Exatas/Foz do Iguaçu, PR

RESUMO: Sistemas lineares surgem em numerosas aplicações, incluindo, ciência da computação, discretização de equações diferenciais, problemas de otimização, além de muitos outros. Visando resolver sistemas lineares de forma eficaz, neste artigo, mediante uma revisão bibliográfica, abordamos aspectos teóricos e computacionais da decomposição LU com pivoteamento parcial. A vantagem de calcular a decomposição LU com pivoteamento parcial para uma matriz A de um sistema linear é que é mais fácil resolver sistemas linea- res quando as matrizes são triangulares, como as matrizes L e U desta decomposição. Na sequência, utilizando o ambiente de desenvolvimento integrando Code::Blocks desenvol- vemos uma implementação em linguagem C para decomposição LU com pivoteamento parcial e aplicamos a resolução de alguns exemplos. O código se mostrou eficiente na resolução de sistemas lineares não singulares.

PALAVRAS-CHAVE: Sistemas Lineares; Decomposição LU; Linguagem C.

INTRODUÇÃO

Este trabalho foi desenvolvido sob a orientação do professor Emídio Santos Portilho Ju´nior visando aplicar as ferramentas computacionais estudadas no Projeto de Ensino Introdução aos *Métodos Computacionais da Matemática Aplicada.*

Um sistema linear pode ser reformulado como uma equação matricial na qual cada elemento de matriz ou de vetor pertence a um corpo, normalmente o dos nu´meros re- ais R (Cormen; Rivest; Stein, 2012). O objetivo deste trabalho é resolver sistemas de equações lineares através de uma implementação em Linguagem C do método denomi- nado decomposição LU com pivoteamento parcial.

Como mencionado anteriormente, inu´meras aplicações em computação, engenharias e ciências aplicadas nos remetem a necessidade de resolver conjuntos de equações lineares de forma simulta^nea, o que, justifica a escolha do tema de deste trabalho.

Outro fator motivador na escolha do método da decomposição LU com privoteamento parcial como tema se deve a sua eficiência na resolução de sistemas lineares. Eficiência já consolidada na literatura matemática (Ruggiero; Lopes, 1997).

METODOLOGIA

Como se trata de um trabalho de revisão bibliográfica e implementação computacio- nal. Nesta Seção de Metodologia apresentaremos de forma sucinta e ordeira os aspectos teóricos que nos levaram a implementação do método.

UMA VISÃO GERAL DA DECOMPOSIÇÃO LU COM PIVOTEAMENTO PARCIAL

Começamos com um conjunto de n equações lineares com n incógnitas:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &= b_2 \\ \qquad\qquad\qquad\qquad\quad \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nn}x_n &= b_n \end{cases} \qquad (1)$$

Uma solução para as equações (1) é um conjunto de valores para x1, x2, . . . , xn que sa- tisfaz todas as equações simultâneamente. Um modo conveniente de expressar as equacões (1) é sob a forma de uma equação matricial de vetores.an1

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

o que é equivalente a escrever

$$Ax = b. \qquad (2)$$

Se A é não singular, possui uma inversa A^{-1}, e

$$x = A^{-1}b \qquad (3)$$

é o vetor solução.

Ou seja, para resolver o sistema (2), poderíamos calcular A^{-1} e, depois, usando a equação (3), multiplicando b por A^{-1} e obter $x = A^{-1}b$. Na prática, essa abordagem demanda um grande custo computacional. Além de sofrer de instabilidade numérica. Felizmente, a decomposição LU com pivoteamento parcial é numéricamente mais estável e mais rápida na prática (Cormen; Rivest; Stein, 2012).

A ideia por trás da decomposição LU com pivoteamento parcial é encontrar três ma- trizes n × n, L, U e P , tais que

$$PA = LU \qquad (4)$$

onde L é uma matriz triangular inferior unitária, U é uma matriz triangular superior e P é uma matriz de permutação.

A vantagem de calcular a decomposição LU com pivoteamento parcial para a matriz A é que é mais fácil resolver sistemas lineares quando as matrizes

são triangulares, como é o caso das matrizes L e U . Uma vez determinada uma decomposição PA = LU para A, podemos resolver a equação (2) resolvendo somente sistemas lineares triangulares da maneira mostrada a seguir. Multiplicando ambos os lados de Ax = b por P obtemos a equação equivalente PAx = Pb que, equivale a permutar as equações em (1).

Usando a decomposição (4), obtemos

$$LUx = Pb$$

Agora definimos y = Ux, onde x é o vetor solução desejado. Primeiro, resolvemos o sistema triangular inferior

$$Ly = Pb \qquad (5)$$

para o vetor incógnita y por um método denominado "substituição direta". Depois de resolvido y, resolvemos o sistema triangular superior

$$Ux = y \qquad (6)$$

para a incógnita x por um método denominado "substituição reversa".

SUBSTITUIÇÃO DIRETA E INVERSA

A *substituição direta* pode resolver o sistema triangular inferior (5). Por conveniência, representamos a permutação P compactamente por um arranjo [1, 2, . . . , n]. Para i = 1, 2, . . . , n, a entrada π [i] inddica que Pi,π[i] = 1 e Pij = 0 para j ≠π [i]. Visto que L é triangular inferior unitária, a equação (5) pode ser reescrita como

$$\begin{cases} y_1 & = & b_{\pi[1]} \\ l_{21}y_1 + y_2 & = & b_{\pi[2]} \\ & \vdots & \\ l_{n1}y_1 + l_{n2}y_2 + l_{n3}y_3 + \ldots + y_n & = & b_{\pi[n]} \end{cases}$$

Em geral, substituimos y1, y2, . . . , yi–1 “diretamente” na i-ésima equação para resol- ver para yi:

$$y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij}y_j.$$

Agora que resolvemos para y, resolvemos para x na equação (6) usando substituição reversa, que é semelhante a substituição direta. Aqui, resolvemos primeiro a n-ésima equação e trabalhamos em sentido contrário até a primeira equação. Visto que U é triangular superior, podemos reescrever o sistema (6) como

$$\begin{cases} u_{11}x_1 + u_{12}x_2 + \ldots + u_{1n}x_n & = & y_1 \\ u_{22}x_2 + \ldots + u_{2n}x_n & = & y_2 \\ & \vdots & \\ u_{nn}x_n & = & y_n \end{cases}$$

Assim, podemos resolver sucessivamente para xn, xn–1, . . . , x1, da seguinte maneira:

CALCULANDO UMA DECOMPOSIÇÃO LU

Como visto, se podemos criar uma decomposição LU com pivoteamento parcial para uma matriz não singular A, substituição direta e substituição inversa podem resolver o sistema Ax = b de equações lineares.

Para calcularmos a decomposição LU usamos o **método de eliminação de Gauss**. Começamos subtraindo multiplos da primeira equação das outras equações para eliminar a primeira variável dessas equações. Então, subtráimos mu´ltiplos da segunda equação da terceira equação e das subsequentes, de modo que agora a

primeira e a segunda variáveis são eliminadas dessas equações. Continuamos esse processo até que o sistema remanes- cente tenha forma triangular superior, na verdade, ele é a matriz U . A matriz L é formada pelos multiplicadores de linha que provocam a eliminação de variáveis (Cormen; Rivest; Stein, 2012).

ESTRATÉGIA DE PIVOTEAMENTO PARCIAL

Em geral, quando resolvemos um sistema de equações lineares Ax = b, temos de pivotear em elementos de A que estão fora da diagonal para evitar divisão por zero e por valores pequenos, mesmo que A seja não singular, visto que isso pode produzir insta- bilidades numéricas. Então, tentamos pivotear em um valor grande (Ruggiero; Lopes, 1997).

A estratégia de pivoteamento parcial consiste em:

1. no início da etapa k da fase de eliminação, escolher para pivoˆ o elemento de maior módulo entre os coeficientes: a_{ik}^{k-1}, i = k, k + 1, . . . , n;
2. trocar as linhas k e i se for necessário.

RESULTADOS E DISCUSSÃO

As considerações teóricas feitas até aqui nos permitiram elaborar o algoritmo (ou pseudocódigo) exposto na Subseção 3.1. Este algoritmo viabilizou a implementação em Linguagem C da fatoração LU com pivoteamento parcial assim como seu uso na resolução dos sistemas lineares apresentados na Subseção 3.2.

ALGORITMO

No que segue, apresentamos o pseudocódigo que resolve o sistema linear (1) via Decomposição LU com Pivoteamento Parcial.

Algorithm 1 Decomposição LU com Pivoteamento Parcial

Algorithm 1 Decomposição LU com Pivoteamento Parcial

procedure DECOMPOSIÇÃOLUP(n, A, x)
  for k de 1 até $n-1$ **do**
    $MA \leftarrow |a_{kk}|$;
    $r \leftarrow k$;
    for i de k até n **do**
      if $|a_{ik}| > MA$ **then**
        $MA \leftarrow |a_{ik}|$;
        $r \leftarrow i$;
    for j de k até $n+1$ **do**
      $aux \leftarrow a_{kj}$;
      $a_{kj} \leftarrow a_{rj}$;
      $a_{rj} \leftarrow aux$;
    for i de $k+1$ até n **do**
      $m_{ik} \leftarrow -\frac{a_{ik}}{a_{kk}}$;
      for j de $k+1$ até $n+1$ **do**
        $a_{ij} \leftarrow a_{ij} + m_{ik} \cdot a_{kj}$
  $x_n \leftarrow \frac{a_{n,n+1}}{a_{nn}}$;
  for i de $n-1$ até 1 **do**
    $Soma \leftarrow 0$;
    for j de $i+1$ até n **do**
      $Soma \leftarrow Soma + a_{ij} \cdot x_j$;
    $x_i \leftarrow \frac{1}{a_{ii}}(a_{i,n+1} - Soma)$;
  return x;

O pseudocódigo acima foi transferido para Linguagem C no ambiente de desenvolvi- mento integrado Code::Blocks.

Os conceitos computacionais necessários a implementação, como Tipo de variáveis, Vetores, Controle de Fluxo, Funções, Cabeçalhos e etc., foram estudados no projeto de ensino *Introdução aos Métodos Computacionais da Matemática Aplicada* que teve sua parte computacional baseada no livro C: *Como Programar* (Deitel; Deitel, 2011).

EXEMPLOS NUMÉRICOS

Nesta Subseção apresentaremos alguns exemplos resolvidos pelo código computa- cional que desenvolvemos. Estes exemplos foram tomados de (Ruggiero; Lopes, 1997). Todos os testes foram realizados com o sistema operacional 64-bit Windows 10, processa- dor Intel Core I7-8550U, 1,99 Ghz, 16 GB de memória Ram. As soluções serão exibidas por meio de fotos da tela de console do Code::Blocks.

Exemplo 1: Resolva o sistema linear

$$\begin{cases} 5x_1 + x_2 + x_3 &= 5 \\ 3x_1 + 4x_2 + x_3 &= 6 \\ 3x_1 + 3x_2 + 6x_3 &= 0 \end{cases}$$

A solução obtida pelo código computacional está exibida na Figura 1.

```
 Exibindo Indices do vetor p:

    0
    1
    2

 Exibindo o conteudo do vetor Solucao:

X[0] =   1.00000
X[1] =   1.00000
X[2] =  -1.00000

Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.
```

Figura 1 - Solução do Exemplo 1.

Fonte: Os Próprios Autores (2024).

O vetor p mencionado na Figura 1 indica quais linhas foram permutadas. Como se pode notar, neste exemplo não foram necessárias permutações. Os dados subsequentes representam a solução obtida para o problema. Esta solução pode ser facilmente verificada dado que se trata de um sistema de pequeno porte.

Exemplo 2: Resolva o sistema linear

$$\begin{cases} x_1 + 0.5x_2 - 0.1x_3 + 0.1x_4 = 0.2 \\ 0.2x_1 + x_2 - 0.2x_3 - 0.1x_4 = -2.6 \\ -0.1x_1 - 0.2x_2 + x_3 + 0.2x_4 = 1.0 \\ 0.1x_1 + 0.3x_2 + 0.2x_3 + x_4 = -2.5 \end{cases}$$

A solução obtida pelo código computacional está exibida na Figura 2. Neste exemplo também não houve a necessidade de se executar permutações nas linhas, como pode ser notado no vetor de permutação p. A solução também pode ser facilmente verifica por substiuição direta no sistema.

```
 Exibindo Indices do vetor p:

    0
    1
    2
    3

 Exibindo o conteudo do vetor Solucao:

X[0] =   2.00000
X[1] =  -3.00000
X[2] =   1.00000
X[3] =  -2.00000

Process returned 0 (0x0)   execution time : 1.152 s
Press any key to continue.
```

Figura 2 - Solução do Exemplo 2.

Fonte: Os Próprios Autores (2024).

Exemplo 3: Resolva o sistema linear

$$\begin{cases} 3x_1 & -4x_2 & +x_3 & = & 9 \\ x_1 & +2x_2 & +2x_3 & = & 3 \\ 4x_1 & & -3x_3 & = & -2 \end{cases}$$

Este exemplo, embora d e pequeno porte, foi tomado de forma proposital, visto sa- bermos da necessidade da execução de permutações pelo método para fins de resolução. Como podemos notar pelo vetor *p* da Figura 3, durante o processo de resolução a primeira linha da matriz passou a ser a segunda. A segunda linha da disposição original passou a ser a terceira, ao passo que a terceira linha da disposição original passou a ser a primeira. Com este exemplo podemos verificar a efetividade do método em fazer permutações das linhas quando essas são necessárias.

```
Exibindo Indices do vetor p:

   2
   0
   1

Exibindo o conteudo do vetor Solucao:

X[0] =   1.00000
X[1] =  -1.00000
X[2] =   2.00000

Process returned 0 (0x0)   execution time : 0.813 s
Press any key to continue.
```

Figura 3 - Solução do Exemplo 3.

Fonte: Os Próprios Autores (2024).

Exemplo 3: Neste exemplo iremos resolver um sistema de ordem 10 dado por

$$\begin{bmatrix} 2 & 1 & 7 & 4 & -3 & -1 & 4 & 4 & 7 & 0 \\ 4 & 2 & 2 & 3 & -2 & 0 & 3 & 3 & 4 & 1 \\ 3 & 4 & 4 & 2 & 1 & -2 & 2 & 1 & 9 & -3 \\ 9 & 3 & 5 & 1 & 0 & 5 & 6 & -5 & -3 & 4 \\ 2 & 0 & 7 & 0 & -5 & 7 & 1 & 0 & 1 & 6 \\ 1 & 9 & 8 & 0 & 3 & 9 & 9 & 0 & 0 & 5 \\ 4 & 1 & 9 & 0 & 4 & 3 & 7 & -4 & 1 & 3 \\ 6 & 3 & 1 & 1 & 6 & 8 & 3 & 3 & 0 & 2 \\ 6 & 5 & 0 & -7 & 7 & -7 & 6 & 2 & -6 & 1 \\ 1 & 6 & 3 & 4 & 8 & 3 & -5 & 0 & -6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_10 \end{bmatrix} = \begin{bmatrix} 86.0 \\ 45.0 \\ 52.5 \\ 108.0 \\ 66.5 \\ 90.5 \\ 139.0 \\ 61.0 \\ -43.5 \\ 31.0 \end{bmatrix}$$

Assim como nos outros exemplos, o método implementado, se mostrou eficiente na resolução do sistema linear. A solução está exposta na Figura 4, onde também podemos observar o vetor p que evidencia que neste caso houve uma necessidade maior de permutação das linhas do sistema linear.

```
Exibindo Indices do vetor p:

   3
   5
   6
   9
   8
   0
   1
   7
   2
   4

 Exibindo o conteudo do vetor Solucao:

X[0] =   3.00000
X[1] =  -4.50000
X[2] =   7.00000
X[3] =   8.00000
X[4] =   3.50000
X[5] =   2.00000
X[6] =   4.00000
X[7] =  -3.50000
X[8] =   2.00000
X[9] =   1.50000

Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

Figura 4 - Solução do Exemplo 4.

Fonte: Os Próprios Autores (2024).

CONSIDERAÇÕES FINAIS

Concluímos que a revisão bibliográfica foi efetiva em fornecer as informações matemáticas necessárias para o desenvolvido do pseudocódigo expresso na Subseção 3.1. Este pseudocódigo permitiu a implementação sistemática da DecomposiçãoLU com Pi- voteamento Parcial. Vale ressaltar que embora o pseudocódigo não seja tão extenso, o mesmo não acontece ao se fazer a implementação em Linguagem C na IDE CodeBlocks. A implementação se mostrou extensa e laboriosa. Apesar do duro trabalho no desen- volvimento do código em Linguagem C, os resultados obtidos na resolução dos sistemas lineares apresentados na Subseção 3.2 foram satisfatórios, evidenciando a eficiência do método e da implementação desenvolvida.

REFERÊNCIAS

CORMEN, T. H.; RIVEST, R. L.; STEIN, C.; LEISERSON, C. **Algoritmos: Teoria e Prática.** 3. ed. São Paulo: GEN LTC, 2012.

DEITEL, P.; DEITEL, H. **C: Como Programar**. 6. ed. São Paulo: Pearson, 2011.

RUGGIERO, M. A. G.; LOPES, V. L. R. **Cálculo Numérico: Aspectos Teóricos e Computacionais.** 2. ed. São Paulo: Makron Books, 1997.