Encontro Regional de Matemática Aplicada e Computacional
30 de novembro a 03 de dezembro de 2021

Análise da eficiência computacional da Regra de Cramer pelo método de Laplace para solução de sistemas lineares mal condicionados

Analysis of the computational efficiency of Cramer's rule by Laplace's method for solving ill-conditioned linear systems

Pedro Fontes Montano
Programa de Pós-Graduação em Ciências Computacionais - IME - UERJ
pedro.montano@pos.ime.uerj.br
Cristiane Oliveira de Faria
Departamento de Análise Matemática - IME - UERJ
cofaria@ime.uerj.br

Resumo: Métodos numéricos computacionais para resolução de sistemas lineares realizam muitas operações elementares em pontos flutuantes para encontrar uma solução. Devido à limitação da conversão de números decimais para binários pelo computador, podem ocorrer erros de precisão. Isso ocorre geralmente quando há um grande volume de variáveis e equações envolvidas nessas operações. Os resultados são arredondados ou truncados durante a execução do algoritmo. Consequentemente, estes erros em sequência podem resultar em uma solução incorreta do problema. Com um cunho didático, este trabalho demonstra através de um experimento computacional de resolução de sistemas lineares com as seguintes matrizes mal condicionadas: Hilbert, Cauchy, Vandermonde e Toeplitz, a influência do formato numérico de ponto flutuante na solução aproximada encontrada. Essas matrizes representaram os coeficientes de matrizes ampliadas de um sistema $Ax = b$. Esse experimento foi realizado com o objetivo de analisar a propagação desses erros numéricos. Para isso foi utilizado o método de resolução de sistemas de equações lineares, conhecido como Regra de Cramer, no qual cada determinante foi calculado pelo método de Laplace. A metodologia utilizada foi o cálculo do erro quadrático médio e do tempo de execução para cada teste. Ao final, foram analisados esses fatores para concluir qual dos sistemas mal condicionados seriam os mais estáveis numericamente para serem resolvidos por esses métodos computacionais.

Palavras-chave: Métodos Numéricos; Álgebra Linear; Complexidade Computacional.

Área Temática: Matemática Aplicada e Ensino.

1 Introdução

Na definição da Análise Numérica, um dos ramos da matemática, o condicionamento de um problema é uma medida que indica se o problema tem boas condições para ser tratado numericamente. Um problema com um número de condição pequeno é chamado de bem condicionado. Enquanto os problemas que possuem um número de condição elevado são denominados mal condicionados [3].

Por exemplo, o condicionamento associado ao sistema linear $Ax = b$ é a medida que estabelece uma estimativa da precisão que se pode obter para uma solução aproximada de x. Isso antes dos efeitos dos erros de arredondamento serem levados em consideração. O condicionamento, portanto, é uma propriedade da matriz ampliada do sistema, não do algoritmo ou da precisão em ponto flutuante do computador usado para resolver o sistema correspondente. Nas próximas subseções serão abordadas as matrizes mal condicionadas de Cauchy, Hilbert, Vandermonde e Toeplitz.

2 Metodologia

Nesta seção apresentamos as definições das matrizes mal condicionadas consideradas no experimento, bem como os algoritmos para obtê-las. Também revisamos os métodos numéricos escolhidos para resolução dos sistemas lineares e os algoritmos implementados.

2.1 Matrizes Mal Condicionadas

2.1.1 Matriz de Cauchy

Uma matriz de Cauchy, nomeada em homenagem a Augustin-Louis Cauchy, é uma matriz $m \times n$ com elementos a_{ij} na forma:

$$a_{ij} = \frac{1}{x_i - y_j}; \quad x_i - y_j \neq 0, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n \tag{1}$$

onde, x_i e y_j são elementos de um conjunto $\mathcal{F}$ de sequências injetivas (contêm elementos distintos).

Algoritmo 1: Algoritmo de Geração da Matriz de Cauchy

```
início
    x_1 ← 0
    para i ← 1 até n + 1 faça
        x_i ← x_i + 1
        y_i ← x_i − 0.5
    fim para
    para i ← 1 até n + 1 faça
        para j ← 1 até n + 1 faça
            C_{i,j} ← 1/(x_i − y_j)
        fim para
    fim para
    retorna C_{n,n}
fim
```

2.1.2 Matriz de Hilbert

A matriz de Hilbert $H_{n \times n}$ é aquela onde cada elemento a_{ij} respeita a seguinte regra:

$$a_{ij} = \frac{1}{(i + j - 1)} \tag{2}$$

Algoritmo 2: Algoritmo de Geração da Matriz de Hilbert

```
início
    para i ← 1 até n + 1 faça
        para j ← 1 até n + 1 faça
            H_{i,j} ← 1/(i + j − 1)
        fim para
    fim para
    retorna H_{n,n}
fim
```

2.1.3 Matriz de Vandermonde

Uma matriz de Vandermonde é uma matriz $V_{n\times n}$ em que os termos de cada linha estão em progressão geométrica, na forma $V_{i,j} = \alpha_i^{j-1}$. A transposta dessa matriz, onde a progressão geométrica acontece nas colunas, também é considerada uma matriz de Vandermonde.

Algoritmo 3: Algoritmo de Geração da Matriz de Vandermonde

```
1  início
2  |  para i ← 1 até n + 1 faça
3  |  |  para j ← 1 até n + 1 faça
4  |  |  |  V_{i,j} ← (i + 1)^j
5  |  |  fim para
6  |  fim para
7  |  retorna V_{n,n}
8  fim
```

2.1.4 Matriz de Toeplitz

Uma matriz de Toeplitz, ou também conhecida como matriz de diagonais constantes, é uma matriz em que cada diagonal descendente da esquerda para a direita tem valor constante. De modo geral, se o elemento $a_{i,j}$ de uma matriz $T_{n\times n}$ for denotado por $a_{i,j} = a_{i-1,j-1}$.

Algoritmo 4: Algoritmo de Geração da Matriz de Toeplitz

```
1  início
2  |  para i ← 1 até n + 1 faça
3  |  |  para j ← 1 até n + 1 faça
4  |  |  |  se i = j então
5  |  |  |  | T_{i,j} ← n ;
6  |  |  |  senão
7  |  |  |  |  T_{i,j} ← (i − j) ;
8  |  |  |  se j > i então
9  |  |  |  |  T_{i,j} ← −T_{i,j}
10 |  |  |  fim se
11 |  |  fim para
12 |  fim para
13 |  retorna T_{n,n}
14 fim
```

2.2 Métodos Numéricos

2.2.1 Teorema de Laplace

A função Determinante é definida como uma função matricial que associa a cada matriz quadrada um escalar, isto é, uma função que transforma uma matriz quadrada em um número real.

O Teorema de Laplace permite calcular o determinante de matrizes de qualquer ordem através de uma expansão em cofatores, demonstrado por diversos autores [5]. Na prática, o cálculo do determinante é realizado obtendo-se os determinantes de matrizes de ordem inferior. Por exemplo, o determinante de uma matriz de ordem 5 é calculado a partir dos determinantes de cinco matrizes de ordem 4. Isto é, para calcular o determinante de uma matriz de ordem n, é necessário calcular n determinantes de matrizes de ordem $n - 1$. Pode-se portanto implementar o Teorema através de um algoritmo recursivo, pois para resolver o problema precisa-se recorrer à própria definição do problema. O caso básico da recursão ocorre

quando temos que calcular o determinante de matrizes de ordem igual 1, pois o determinante é o único elemento da matriz, conforme pode ser visto no algoritmo a seguir.

Algoritmo 5: Método de Laplace

```
1  início
2      para i ← 1 até n faça
3          l_i ← 0
4      fim para
5      para i ← 1 até n faça
6          Insere(i, l_n)
7      fim para
8      se tamanho(M) = 1 então
9          det ← M_{1,1}
10         retorna det
11     senão
12         k ← 0
13         enquanto l_k ≤ tamanho(l_n) faça
14             para j ← 1 até n faça
15                 M ← M[1 :]
16                 para i ← 1 até tamanho(M) faça
17                     M_i ← M[i, 0 : j] + M[i, j + 1 :]
18                 fim para
19                 det ← det + [(−1)^{j mod 2} × A_{0,j} × Laplace(M)]
20             fim para
21             k ← k + 1
22         fim enqto
23     fim se
24     retorna det
25 fim
```

O Algoritmo 5 pode ser analisado em função da sua complexidade computacional. Exceto quando a maioria dos elementos da matriz é igual a zero, calcular determinantes usando Teorema de Laplace é um método lento mesmo para matrizes não muito grandes. Uma vez que, o algoritmo de resolução é recursivo e ao final calcula recursivamente uma sequência fatorial de determinantes de ordem 1.

Dessa forma, para uma matriz $n \times n$, o número de multiplicações necessárias para calcular o determinante é da ordem de n fatorial. Então, por exemplo, uma matriz de ordem 15, ou seja, uma matriz com 15 colunas e 15 linhas vai precisar resolver 15! operações, isso equivale ao cálculo de 1.307.674.368.000 operações. Essa é uma tarefa que é difícil de ser resolvida até mesmo por computadores modernos [4].

2.2.2 Regra de Cramer

A Regra de Cramer é fácil de ser calculada quando as matrizes de coeficientes são de baixa ordem. Infelizmente para sistemas grandes, os cálculos de todos os determinantes necessário tornam esse método, em geral, computacionalmente impraticável [4]. Por conta disso, esse método não é muito usado para cálculos numéricos complexos, uma vez que o número de operações que esse método envolve é muito grande, quando são utilizadas muitas equações [1], conforme será demonstrado em seguida.

No cálculo do determinante de uma matriz de ordem n, é necessário calcular $n!$ produtos de n fatores, e depois somá-los. Para resolver um sistema n x n pela Regra de Cramer é preciso calcular $(n + 1)$ determinantes de ordem n, o número de operações se elevaria a $[(n + 1)(n!(n - 1)]$, o que gera uma complexidade assintótica de pelo menos $O(n^2n!)$. Como um exemplo, para resolver um sistema de 10 equações e 10 incógnitas, pela Regra de Cramer são necessárias pelo menos $10^210! = 362.880.000$ operações,

enquanto que pelos métodos diretos de eliminação por linhas como Gauss e Gauss-Jordan, levaria em torno de n^3 operações [1].

Algoritmo 6: Regra de Cramer por Laplace

```
1  início
2      para i ← 1 até n faça
3          d_i, x_i ← 0
4      fim para
5      para i ← 1 até n faça
6          M_{n×n} ← A_{n×n}
7          para j ← 1 até n faça
8              M_{i,j} ← b_j
9          fim para
10         d_i ← Laplace(M_{n×n})
11         x_i ← d_i / Laplace(A_{n×n})
12     fim para
13     retorna x_n
14 fim
```

3 Resultados e análises

3.1 Implementação

Os algoritmos descritos foram implementados em linguagem de programação para obtenção dos resultados dos métodos de forma computacional para cada uma das matrizes. A linguagem escolhida para o experimento foi o Python, utilizando o compilador para o sistema operacional Windows. Por ela ser uma linguagem de propósito geral considerada de alto nível, multi-paradigma: podendo ser utilizada tanto no paradigma orientado a objetos quanto no paradigma imperativo, funcional e procedural, com tipagem dinâmica de variáveis. É uma linguagem muito utilizada no meio científico por sua fácil manipulação.

Dessa forma, uma de suas principais características é permitir a fácil interpretação do programa. Uma vez que são necessárias poucas linhas de código, se comparado a um mesmo programa feito em outras linguagens como Java, por exemplo. Levando isso tudo em consideração na hora da escolha, os códigos foram implementados, compilados e executados nessa linguagem no seguinte ambiente: - Processador (CPU): Intel Quad Core i7-2600K de 64 bits com frequência de 3.4 GHz (Giga Hertz); - Memória (RAM): 16 GBs (Gigabytes) Padrão DDR3 com frequência de 1.600 Mega Hertz (MHz); - Sistema Operacional: Windows 10 Profissional x 64 Versão 20H2; - Ambiente Integrado de Desenvolvimento (IDE): Microsoft Visual Studio Code Versão 1.55.1; - Compilador: Python versão 3.8.5 com o Pacote Anaconda.

3.2 Análise dos Resultados

Utilizando esse ambiente computacional, foi proposto um sistema modelo $Ax = b$ onde a matriz A é uma das matrizes mal condicionadas descritas na seção anterior e o vetor b é obtido através do produto de A com o vetor unitário $x_{n\times 1} = [1, 1, ..., 1]$. O experimento retornou um vetor x de solução pela Regra Cramer para cada matriz ampliada. Cada um desses vetores solução foram então comparados com a solução exata original unitária utilizando as técnicas descritas nas próximas seções.

3.2.1 Erros Computacionais

Os métodos descritos ao serem aplicados a sistemas grandes necessitaram de muitas operações aritméticas em ponto flutuante, o que resulta em um acúmulo de erros de arredondamento, mesmo em um ambiente computacional de alta precisão [4]. Uma forma estatística de medir esse acúmulo de erros é conhecida

como Erro Quadrático Médio (EQM), que calcula a média quadrática da diferença entre dois conjuntos de valores: o vetor de elementos esperado V_i com os estimados V_i'. O primeiro seria o conjunto dos valores considerados ideais numa amostra e outro conjunto seriam os valores observados nessa mesma amostra. A diferença entre o valor esperado e o valor estimado é chamado de erro, ou também, como desvio do valor esperado [2].

Esta diferença é utilizada em forma absoluta, ou seja, em módulo, para evitar que essas diferenças sejam negativas. Porque quando há diferenças negativas, isso tendenciaria a uma análise contrária da realidade dos dados observados. Em seguida, as diferenças absolutas são somadas e dividas pelo número de elementos da amostra para que venham a se equilibrar trazendo uma estimativa média dos erros da amostra. Por fim, quanto maior for o valor do EQM de uma amostra para a outra, maior a diferença entre o valor esperado e o estimado dos elementos observados dessa amostra em relação a outra [2].

A equação abaixo representa a fórmula de cálculo do EQM, seguindo esse conceito apresentado:

$$EQM = \frac{\sum_{i=1}^{n} (V_i - V_i')^2}{n} \tag{3}$$

Essa equação acima pode ser obtida através da implementação do seguinte algoritmo:

Algoritmo 7: Algoritmo para o Cálculo do Erro Quadrático Médio (EQM)

```
1 início
2 |  EQM ← 0
3 |  para i ← 1 até n + 1 faça
4 |  |  EQM ← EQM + (V_i − V_i')^2
5 |  fim para
6 |  retorna EQM
7 fim
```

Para verificar a influência das matrizes mal-condicionadas na solução calculada foi utilizado o método EQM. A Tab. 1 representa o valor aproximado encontrado para cada erro quadrático médio de cada um dos vetores solução do sistema linear, de acordo com o tipo de matriz utilizados variando a ordem de dimensão.

Tabela 1: Erro Quadrático Médio (EQM) da Regra de Cramer por Laplace

Ordem	Cauchy	Hilbert	Vandermonde	Toeplitz	Maior Ordem de Grandeza
4	1,23 x 10^{-32}	1.52 x 10^{-23}	0	0	Hilbert
5	2,96 x 10^{-32}	3.75 x 10^{-19}	0	0	Hilbert
7	3,17 x 10^{-31}	3,16 x 10^{-5}	0	0	Hilbert
8	2,29 x 10^{-31}	1,41 x 10^{1}	0	0	Hilbert

Nota-se que a matriz de Hilbert é a mais instável numericamente, enquanto a de Vandermonde e Toeplitz, conforme suas fórmulas de geração aqui descritas, são as mais estáveis, não gerando nenhum erro significativo. Grande parte dessa estabilidade encontrada no experimento deriva do fato do método não produzir muitos números decimais, não gerando erros de arredondamento no sistema de ponto flutuante.

3.2.2 Tempo de Execução

Um estudo experimental da complexidade do método de Cramer por Laplace, pode ser apresentado através do tempo de execução para diversas dimensões das matrizes Mal Condicionadas: Cauchy, Hilbert, Vandermonde e Toeplitz. Temos que $t = t_0 * n$, onde t é o tempo total de execução do algoritmo, t_0 é o tempo de um único passo computacional, n é o número de passos executados no método. Logo, o tempo de execução é diretamente proporcional a complexidade do método. Os resultados obtidos são apresentados na Tab.2.

Tabela 2: Tempo de Execução da Regra de Cramer por Laplace em segundos

Ordem	Cauchy	Hilbert	Vandermonde	Toeplitz
4	2,41 x 10^{-3}	2,1 x 10^{-3}	2,58 x 10^{-3}	2,91 x 10^{-3}
5	1,26 x 10^{-2}	1,21 x 10^{-2}	1,91 x 10^{-2}	1,31 x 10^{-2}
7	7,05 x 10^{-1}	6,59 x 10^{-1}	6,75 x 10^{-1}	6,55 x 10^{-1}
8	5,95 x 10^{0}	5,9 x 10^{0}	6,1 x 10^{0}	5,85 x 10^{0}

O resultado final do tempo mostra valores semelhantes de tempo para as quatro matrizes. No entanto, percebe-se que a medida que a dimensão do sistema aumenta o tempo de execução necessário sofre um aumento de ordem exponencial. A complexidade de tempo não mudaria mesmo otimizando os algoritmos com as regras que permitem o cálculo direto de determinantes de ordem dois pela Regra de Sarrus. Sabe-se que quanto mais zeros nas linha ou coluna escolhidas para o cálculo possui, mais fácil será calcular o determinante, mas para isso, deve-se procurar tal linha ou coluna, caso ela exista. O que representaria mais custo computacional [1].

4 Conclusões

Conclui-se que a Regra de Cramer pelo método de Laplace é um método numericamente estável para solucionar sistemas lineares. Uma vez que pela análise do erro quadrático médio conclui-se as quatros matrizes geraram erros computacionais relativamente pequenos. Isso aconteceu devido ao fato do método utilizar poucas frações em relação a outros métodos como o de Gauss, por exemplo. No entanto, o crescimento de tempo é fatorial em função da entrada, como pode ser observado no resultado do tempo do experimento. Dessa forma, comprova-se que não compensa utilizá-los num contexto prático onde são necessários computar enorme quantidade de equações e incógnitas, devido ao imensurável tempo de execução que isso levaria. Os métodos iterativos de Jacobi e Gauss-Seidel, apesar de terem problemas de estabilidade numérica, por outro lado, têm complexidade polinomial de tempo de execução. No entanto, o fato da Regra de Cramer e do Teorema de Laplace serem ineficientes em tempo não significa que sejam métodos inúteis, pois ambos são alguns dos Teoremas mais importantes da Álgebra devido aos subsequentes corolários. Ressaltamos que este estudo tem um propósito didático de possibilitar um primeiro contato em resolução de sistemas lineares considerando casos especias de matrizes mal-condicionadas, já que seguindo o algoritmo o experimento pode ser reproduzido por leitores. E também apresentar questões trazidas com a influência do erro de truncamento a partir das operações básicas, a relação do tempo de execução com a dimensão do sistema também puderam ser analisadas experimentalmente.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

[1] Boldrini, J. L. *Algebra linear.* Harper & Row do Brasil, 1980.

[2] da Fonseca, J. S., Andrade Martins, G., and Toledo, G. L. *Estatísca aplicada.* Atlas, 1982.

[3] Franco, N. B. *Cálculo numérico.* Prentice Hall Brasil, 2006.

[4] Holt, J. *Álgebra Linear com Aplicações.* W. H. Freeman, Macmillan Learning, 2017.

[5] Strang, G. *Linear Algebra and Its Applications.* Wellesley-Cambridge Press, Wellesley, MA, 2009.