CComp PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS COMPUTACIONAIS

Revisão sobre os métodos iterativos

Vitor Guilherme Coutinho de Barros Junior
Jefson Rosa Florêncio

Resumo

O uso de métodos iterativos para resolver grandes sistemas esparsos de equações lineares é amplamente estudado pela sua velocidade. Existem muitos métodos para resolver tais problemas e o truque é encontrar o método mais eficaz para o problema em questão. Infelizmente, um método que funciona bem para um tipo de problema pode não funcionar tão bem para outro ou pode não funcionar. Dado isso, faz-se uma revisão dos métodos mais utilizados e aplica-se alguns exercícios para o método do gradiente conjugado.

1 Introdução

É bastante comum encontrar sistemas lineares que envolvem uma grande porcentagem de coeficientes nulos. Esses sistemas são chamados de sistemas esparsos. Para esses tipos de sistemas, o método de Eliminação de Gauss não é o mais apropriado, pois ele não preserva essa esparsidade, que pode ser útil por facilitar a resolução do sistema.

Os métodos iterativos apresentarão um resultado aproximado, que será tão próximo do resultado real conforme o número de iterações realizadas. Para determinar a solução de um sistema linear por métodos iterativos, precisamos transformar o sistema dado em um outro sistema onde possa ser definido um processo iterativo.

A solução obtida para o sistema transformado deve ser também solução do sistema original.

Os Métodos iterativos são mais econômicos no que tange a memória dos computadores. Podem ser usados para reduzir os erros de arredondamento na solução obtida por métodos exatos e em alguns casos podem ser aplicados para resolver conjuntos de equações não lineares.

2 Revisão da literatura

Foi feita uma análise detalhada do referencial teórico indicado para o estudo numérico e resolução de sistemas lineares.

Como base referencial de Cálculo Numérico foram utilizados os livros "Cálculo Numerico - Aspectos Teóricos e Computacionais"[Ruggiero Lopes, 2000] [4] e "Métodos Numéricos"[Cristina Cunha, 2013] [2] .

A descrição dos métodos iterativos de resolução de sistemas lineares descrito no capítulo 8 do livro "Numerical Mathematics and Computing"[Cheney and Kincaid, 2007] [6] foi a base teórica para a análise representada neste trabalho.

Sobre os critérios de convergência do método Gauss-Seidel foi utilizado o artigo "Generalized line criterion for gauss-seidel method"[Garcia Stern] [5]

Especificamente sobre o método iterativo gradiente conjugado, também foi tomada como base referencial o livro Matrix Computations [Golub and Van Loan, 1996] [3].

3 Condicionamento

A taxa de convergência dos métodos iterativos depende de propriedades espectrais da matriz de coeficientes portanto pode-se tentar transformar o sistema linear em um que seja equivalente no sentido de ter a mesma solução, mas com propriedades espectrais mais favoráveis. Um pré-condicionador é uma matriz que efetua tal transformação. Por exemplo, se uma matriz M se aproxima da matriz de coeficientes A de alguma forma, o sistema transformado $M^{-1}Ax = M^{-1}b$ possui a mesma solução do sistema original $Ax = b$, mas as suas propriedades espectrais são mais favoráveis.
Ao criar um pré-condicionador, nos deparamos com um dilema entre encontrar uma matriz M que se aproxime de A, e para a qual resolver um sistema seja mais fácil do que resolver outro com A, ou encontrar uma matriz M que se aproxime de A^{-1}, de modo que apenas a multiplicação por M seja necessário. A maioria dos pré-condicionadores se enquadra na primeira categoria.

3.1 Condicionamento pela parte simétrica

Métodos de gradiente para sistemas não autoadjuntos requerem o armazenamento de vetores previamente calculados [1] portanto é um tanto notável que o pré-condicionamento pela parte simétrica $(A + A^T)/2$ da matriz de coeficientes A leva a um método que não precisa desse armazenamento estendido. No entanto, resolver um sistema com a parte simétrica de uma matriz pode não ser mais fácil do que resolver um sistema com a matriz completa. Este problema pode ser resolvido impondo um método iterativo aninhado, onde um pré-condicionador baseado na parte simétrica é usado.

4 Métodos Iterativos Estacionários

Como alternativa aos métodos diretos de resolução de sistemas lineares, os métodos iterativos operam através de aproximação, produzindo uma sequência de vetores $(x^{(0)}, x^{(1)}, x^{(2)}, ..., x^{(n)})$ com soluções aproximadas para o sistema Ax = b. O método iterativo é projetado para que a sequência de vetores convirja para a solução real. As interações são interrompidas no momento que for alcançada a precisão definida inicialmente.

Um algoritmo iterativo geral para métodos estacionários proposto por Kincaid e Cheney [6] para resolução de um sistema é o seguinte: Selecione um matriz não singular Q, e tendo escolhido um vetor inicial arbitrário $x^{(0)}$, gere vetores $x^{(1)}, x^{(2)}, x^{(n)}$, recursivamente da equação

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b \tag{1}$$

bem que (k = 1, 2, . . ., n).

Supondo que a sequência $x^{(k)}$ convirja, para um vetor x*, então, considerando o limite como $k \to \infty$ no sistema, obtemos a seguinte representação

$$Qx^* = (Q - A)x^* + b \tag{2}$$

Isso leva a

$$Ax^* = b \tag{3}$$

Assim, se a resolução das sequências for convergente, seu limite é uma solução para o sistema original. Um método é chamado de estacionário quando ele pode ser expresso na forma simples $x^{(k)} = Bx^{(k-1)} + c$.

4.1 Método de Jacobi

O método de Jacobi, criado pelo matemático alemão Carl Gustav Jakob Jacobi, pode ser obtido a partir do sistema linear:

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = y_1 \tag{4}$$
$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = y_2 \tag{5}$$
$$a_{31}x_1 + a_{32}x_2 + ... + a_{3n}x_n = y_3 \tag{6}$$
$$a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = y_n \tag{7}$$

Em que isolando o elemento da primeira equação temos a seguinte representação

$$x_1^{(k+1)} = \frac{y_1 - (a_{12}x_2^{(k)} + ... + a_{1n}x_n^{(k)})}{a_{11}} \tag{8}$$

O resultado do sistema linear é obtido na iteração que obtiver maior convergência dentro o número limite de iterações definidas.

4.2 Método de Gauss-Seidel

Da mesma forma que no método de Jacobi, no método de Gauss-Seidel isolamos o elemento da equação.

$$x_1^{(k+1)} = \frac{y_1 - (a_{12}x_2^{(k)} + ... + a_{1n}x_n^{(k)})}{a_{11}} \tag{9}$$

Observamos que a equação para $x_2^{(k+1)}$ repete o valor de $x_1^{(k)}$ na iteração k. Com isso podemos pensar em usar o valor que foi calculado na equação anterior e temos a seguinte equação:

$$x_2^{(k+1)} = \frac{y_2 - (a_{21}x_1^{(k+1)} + a_{23}x3^{(k)} + ... + a_{2n}x_n^{(k)})}{a_{11}} \tag{10}$$

No artigo [5] diz que dimensionar as equações e variáveis de um sistema linear Ax=b é a operação de multiplicar a matriz de coeficientes pelas diagonais não singulares esquerda e direita matrizes obtendo o sistema equivalente EAWy=Eb. O escalonamento é amplamente utilizado em álgebra linear computacional para pré-condicionamento, controle de erro de ponto flutuante acumulação. Sendo E e W matrizes diagonais não singulares. Então se o ponto de convergência de Gauss-Seidel converge em Ax = b, então converge em EAWy=Eb.

4.3 Método de Sobre-Relaxação Sucessiva (Successive Over-Relaxation – SOR)

O Método de Sobre-Relaxação Sucessiva (Successive Over-Relaxation - SOR) é visto como uma versão aprimorada do método de Gauss-Seidel para a resolução de sistemas de equações lineares. A equação que representa a sequência é:

$$x_i^{(k)} = \omega\{[-\sum_{\substack{j=1 \\ j<i}}^{n}(a_{ij}/a_{ii})x_j^{(k)}$$

$$- \sum_{\substack{j=1 \\ j>i}}^{n}(a_{ij}/a_{ii})x_j^{(k-1)} + (b_i/a_{ii})]\} + (1 - \omega)x_i^{(k-1)} \quad (11)$$

Onde o termo ω pode acelerar a convergência para a solução do sistema.

5 Métodos Iterativos Não Estacionários

Os métodos não estacionários diferem dos métodos estacionários porque os cálculos envolvem informações que mudam a cada iteração. Normalmente, as constantes são calculadas tomando produtos internos de resíduos ou de outros vetores decorrentes do método iterativo.

5.1 Método do Gradiente e Gradiente Conjugado

Os métodos do tipo gradiente são uma coleção de métodos iterativos bastante populares na resolução de sistemas de equações lineares esparsos. Em especial, o método do gradiente conjugado é muito útil porque tem uma razoável complexidade de tempo e de armazenamento.
Define-se simétrica a matriz real $A_{[n\times n]}$ tal que $A^T = A$.
O produto interno de dois vetores $u = [u_1, u_2, ..., u_n]^T$ e $v = [v_1, v_2, ..., v_n]^T$ pode ser escrito como

$$\langle u, v\rangle = u^T v = \sum_{i=1}^{n} u_i v_i \quad (12)$$

resultando no somatório das coordenadas.
Outra definição necessária para a resolução do método gradiente conjugado é que uma matriz $A_{[n\times n]}$ é positiva se $\langle x, x\rangle_A > 0$ para todos os vetores diferentes de zero $x \in R^n$. Expressões como $< u, v >$ e $< u, v >_A$ podem reduzir o sistema para uma matriz 1×1 que são tratados como valores escalares. Assim, a função quadrática a ser minimizada com respeito a um vetor de escalar é representada como:

$$f(x) = \frac{1}{2}\langle x, x\rangle_A - \langle b, x\rangle + c \quad (13)$$

Sendo A uma matriz simétrica e positiva, x e b vetores e c uma constante escalar, aplicando gradiente a função temos $grad(f(x))$ que possui a seguinte representação:

$$f'(x) = \frac{1}{2}A^T x + \frac{1}{2}Ax - b \quad (14)$$

Como a solução de $Ax = b$ é um ponto crítico de f(x) para A simétrica e positiva, percebe-se que uma forma alternativa de resolver o sistema linear $Ax = b$ é encontrar um vetor x que minimize f(x).

O método do gradiente conjugado se baseia na escolha de um pequeno conjunto de vetores de direção conjugadas $p^{(k)}$ que resultam numa sequência eficiente para obter uma boa aproximação do vetor solução. Supondo que $\{p^{(1)}, p^{(2)}, ..., p^{(k)}, ..., p^{(n)}\}$ é um conjunto contendo uma sequência de n vetores de direção conjugados, então eles formam uma base para o espaço R^n. Assim, podemos expandir o vetor solução x* como uma combinação linear desses vetores de base:

$$\alpha_k = \langle p^{(k)}, b\rangle / \langle p^{(k)}, p^{(k)}\rangle_A \quad (15)$$

Com uma estimativa inicial $x^{(0)}$ até a solução verdadeira x*, podemos assumir sem perda de generalidade que $x^{(0)}$ é o vetor nulo, mas qualquer outro chute inicial deve ser válido. A verdadeira solução x* é também o único minimizador de

$$f(x) = \frac{1}{2}\langle x, x\rangle_A - \langle x, x\rangle = \frac{1}{2}x^T Ax - x^T x \quad (16)$$

para $x \in R^n$. Isso sugere que o primeiro vetor de base $p^{(1)}$ seja o gradiente de f em x = x(0) , que é igual -b. Os outros vetores na base agora são conjugados ao gradiente, daí o nome método do gradiente conjugado. O k-ésimo vetor residual é

$$r^{(k)} = b - Ax^{(k)} \quad (17)$$

O método do gradiente se move na direção e em sentido oposto a $r^{(k)}$. Pegando a direção mais próxima do vetor gradiente $r^{(k)}$, insistindo que os vetores de direção $p^{(k)}$ sejam conjugados entre si. Juntando tudo isso, obtemos a expressão

$$p^{(k+1)} = r^{(k)} - (\langle p^{(k)}, r^{(k)} \rangle_A / \langle p^{(k)}, p^{(k)} \rangle_A) p^{(k)} \tag{18}$$

5.2 MINRES e SYMMLQ

Os métodos MINRES e SYMMLQ são variantes que podem ser aplicadas a sistemas indefinidos, isto é, existem vetores x e y tais que $\langle x, x \rangle_A > 0 > \langle y, y \rangle_A$. Isso aumento a quantidade de problemas que podem ser resolvidos.
No método Gradiente Conjugado pode ocorrer uma quebra do algoritmo correspondente a um pivô zero se a matriz for indefinida. Além disso, para matrizes indefinidas, a propriedade de minimização do método Conjugate Gradient não é mais bem definida. Os métodos MINRES e SYMMLQ são variantes do método Gradiente Conjugado que não sofrem quebras por matrizes indefinidas. MINRES minimiza o resíduo na norma euclidiana e o SYMMLQ resolve o sistema projetado, mas não minimiza nada (mantém o resíduo ortogonal a todos os anteriores).

Quando A não é positiva, mas é simétrica, ainda podemos construir uma base ortogonal para o subespaço por meio de relações de recorrência de três termos. Eliminar as direções de busca nas equações (15) e (18) dá uma recorrência

$$Ar^{(i)} = r^{(i+1)} t_{i+1,i} + r^{(i)} t_{i,i} + r^{(i-1)} t_{i-1,i} \tag{19}$$

que pode ser escrito em forma matricial como

$$AR_i = R_{i+1} \overline{T}_i \tag{20}$$

onde $\overline{T}_i$ é uma matriz tridiagonal com $i+1$ linhas e i colunas.
Embora nós encontremos um problema já que $< .,. >_A$ não representa mais um produto interno, podemos minimizar o residual com respeito à norma euclidiana obtendo

$$x^{(i)} \in (r^{(0)}, Ar^{(0)}, ..., A^{i-1} r^{(0)}),\ x^{(i)} = R_i \overline{y} \tag{21}$$

que minimiza

$$||Ax^{(i)} - b|| = ||AR_i \overline{y} - b|| = ||R_{i+1} \overline{T}_i y - b|| \tag{22}$$

Agora é possível explorar o fato que se $D_{i+1} = diag(||r^{(0)}||, ||r^{(1)}||, ..., ||r^{(i)}||)$, então $R_{i+1} D_{i+1}^{-1}$ é uma transformação ortogonal com respeito ao subespaço no qual estamos

$$||Ax^{(i)} - b|| = ||D_{i+1} \overline{T}_i y - ||r^{(0)}|| e^{(1)}|| \tag{23}$$

e essa última equação pode ser entendida como um problema de minimização de mínimos quadrados. O elemento na posição (i+1,i) pode ser aniquilado por uma rotação de Givens e o sistema resultante será bidiagonal superior e pode ser resolvido; esse é o método MINRES.
Outra possibilidade é resolver o sistema $T_i y = ||r^{(0)}|| e^{(1)}$ (T_i é a matriz superior $[ixi]$ parte de $\overline{T}_i$). Uma alternativa é decompor T_i por uma decomposição LQ; esse é o método SYMMLQ.

6 Resultados

6.1 Implementação do Método Gradiente Conjugado

Para resolução dos exercícios propostos, foi implementado o método gradiente e do método gradiente conjugado ,computacionalmente, utilizando a linguagem de programação Python.

Listing 1: Método Gradiente

```
import sys
import numpy as np

def gradient_method_wrapper(A: np.ndarray, b: np.ndarray,
x: np.ndarray,
    tol: float = 0.1**5) -> np.ndarray:
    '''
        Gradient Method - Wrapper to the real function
        A: Matrix
        b: output array
        x: solution array
        tol: tolerance of the solution
    '''
    r: np.ndarray = b - A.dot(x)
    delta: float = r.dot(r)
    s: float = delta/(r.dot(A.dot(r)))
    x = x + s*r
    if np.sqrt(delta) < tol:
        return x
    else:
        return gradient_method_wrapper(A, b, x, tol)

def gradient_method(A: np.ndarray, b: np.ndarray,
 max: int = 200,
    tol: float = 0.1**5) -> np.ndarray:
    '''
        Gradient Method
        A: Matrix
        b: output array
        max: maximum number of iterations
        tol: tolerance of the solution
    '''
    sys.setrecursionlimit(max)
    x = b/np.diag(A) #initial guess
    try:
        return gradient_method_wrapper(A, b, x, tol)
    except RuntimeError:
        print(f'Solution did not converge with {max}
        iterations.')
        return x
```

Listing 2: Método Gradiente Conjugado

```
import sys
```

```
import numpy as np

def conjugated_gradient_method_wrapper(
    A: np.ndarray,
    b: np.ndarray,
    x: np.ndarray, r: np.ndarray, aux: float,
    v: np.ndarray,
    tol: float = 0.1**5) -> np.ndarray:
    '''
        Conjugated Gradient Method
        - Wrapper to the real
        function
        A: Matrix
        b: output array
        x: solution array
        r: residue array associated with x
        aux: residue value associated with x
        v: array that represents
        the direction to go
        tol: tolerance of the solution
    '''
    z: np.ndarray = A.dot(v)
    s: float = aux/z.dot(v)
    x = x + s*v
    r = r - s*z
    delta: float = r.dot(r)
    if np.sqrt(delta) < tol:
        print(np.sqrt(delta))
        return x
    else:
        m = delta/aux
        aux = delta
        v = r + m*v
        return
        conjugated_gradient_method_wrapper(
        A, b, x, r, aux, v, tol)

def conjugated_gradient_method(A: np.ndarray,
b: np.ndarray,
    max: int = 200,
    tol: float = 0.1**5) -> np.ndarray:
    '''
        Conjugated Gradient Method
        A: Matrix
        b: output array
        max: maximum number of iterations
        tol: tolerance of the solution
    '''
    sys.setrecursionlimit(max)
    x = b/np.diag(A) #initial guess
    r = b - A.dot(x)
    aux = r.dot(r)
    v = b - A.dot(x)
    try:
        return
        conjugated_gradient_method_wrapper(
        A, b, x, r, aux, v, tol)
    except RuntimeError:
        print(f'Solution did not converge
        with {max} iterations.')
        return x
```

6.2 Exercício 9

Sendo a matriz

$$A = \begin{bmatrix} 2 & -0.3 & -0.2 \\ -0.3 & 2 & -0.1 \\ -0.2 & -0.1 & 2 \end{bmatrix} \tag{24}$$

e o vetor

$$b = \begin{bmatrix} 7 \\ 5 \\ 3 \end{bmatrix} \tag{25}$$

, ao aplicarmos o algoritmo do gradiente conjugado ao sistema Ax=b, temos como solução o vetor $x^* = [4.19304619, 3.23300467, 2.08095485]^T$.

6.3 Exercício 10-a

Sendo a matriz A dada no exercício 10 a de [Cheney and Kincaid, 2007] [6], na página 340 e b um vetor obtido pela aplicação da matriz A à discretização da parábola $(x-1)(x+1)$. Supondo-se que a viga esteja apoiada nos pontos $x = -1$ e $x = 1$, não podemos aplicar diretamente o método do gradiente conjugado ao problema já que a matriz não é simétrica, portanto resolve-se o sistema auxiliar $A^TAx = A^Tb$. Assim, se x^* for solução de $Ax = b$ então teremos que $A^TAx^* = A^Tb$. É interessante notar que o inverso não é sempre verdade. Ao aplicarmos o algoritmo do gradiente conjugado ao sistema $A^TAx = A^Tb$, temos soluções bem próximas daquelas encontradas pelo método disponível na biblioteca do Python para valores de n até 30. Ao aumentarmos a discretização do problema, surgem cada vez mais autovalores próximos de zero que atrapalham a convergência e acabam gerando cancelamentos catastróficos dos dígitos presentes.

6.4 Exercício 10-b

Sendo a matriz A dada no exercício 10 b de [Cheney and Kincaid, 2007] [6], na página 341 e b um vetor obtido pela aplicação da matriz A à discretização da parábola x^2. Supondo-se que a viga esteja engastada no ponto $x = 0$ e livre no ponto $x = 1$, não podemos aplicar diretamente o método do gradiente conjugado ao problema já que a matriz não é simétrica, portanto resolve-se o sistema auxiliar $A^TAx = A^Tb$. Assim, se x^* for solução de $Ax = b$ então teremos que $A^TAx^* = A^Tb$. É interessante notar que o inverso não é sempre verdade. Ao aplicarmos o algoritmo do gradiente conjugado ao sistema $A^TAx = A^Tb$, temos soluções bem próximas daquelas encontradas pelo método disponível na biblioteca do Python para valores de n até 20. Ao aumentarmos a discretização do problema, surgem cada vez mais autovalores próximos de zero (em módulo) que atrapalham a convergência e acabam gerando cancelamentos catastróficos dos dígitos presentes.

6.5 Exercício 11

Sendo $A_{[nxn]}$ matriz construída pelas seguintes regras

$$a_{i,i} = 3 \tag{26}$$
$$a_{i,j} = 0.5, \text{ se } i + j = n - 1 \wedge i \neq j \tag{27}$$
$$a_{i,j} = -1, \text{ se } j = i + 1 \vee i = j + 1 \tag{28}$$
$$a_{i,j} = 0, \quad \text{caso contrário} \tag{29}$$

e o vetor $b_{[n]}$ é um vetor tal que

$$b_1 = 3 \tag{30}$$
$$b_{\lceil n/2 \rceil} = 1 \tag{31}$$
$$b_i = 1.5, \quad \text{caso contrário} \tag{32}$$

, ao aplicarmos o algoritmo do gradiente conjugado ao sistema Ax=b, temos como solução o vetor $x^* = [1, 1, ..., 1]^T$. Alguns erros de arredondamento são encontrados, mas ainda com a tolerância desejada de 6 casas decimais. Não se encontrou nenhum resultado inesperado ao se variar n de 5 a 95 de 10 em 10.

7 Conclusão

O método do gradiente conjugado é um método popular para resolver sistemas grandes e esparsos de equações lineares. É particularmente adequado para resolver sistemas simétricos e positivos definidos e pode convergir mais rapidamente do que outros métodos para esses tipos de sistemas. Ele não funciona bem para sistemas não simétricos ou indefinidos, e outros métodos, como o método MINRES e SYMMLQ, devem ser usados nesses casos.

Referências

[1] R. Barret. *Templates for the solution of linear systems: Building blocks for iterative methods*. 1993.

[2] M. C. C. Cunha. *Métodos Numéricos*. 2013.

[3] G. H. Golub. *Matrix Computations*. 1996.

[4] V. L. d. R. L. Marcia A. Gomes Ruggiero. *Cálculo Numérico - Aspectos teóricos e Computacionais*. 2000.

[5] C. H. J. M.V.P. GARCIA and J. STERN. Generalized line criterion for gauss-seidel method. *Comp. Appl. Math.*, 22(1), 2003.

[6] D. K. Ward Cheney. *Numeral Mathematics and Computing*. 2007.