Métodos Iterativos para Resolução de Sistemas Lineares

Gustavo Henrique Silva Sarturi
Análise Numérica I
gustavo.sarturi@ufpr.br

1. Introdução

O objetivo deste trabalho é resolver sistemas lineares do tipo $A\mathbf{x} = \mathbf{b}$ onde A é uma matriz quadrada de ordem n e $\mathbf{x}$ e $\mathbf{b}$ são vetores de n componentes. O método de Gauss-Jordan é o mais clássico para se resolver esse tipo de problema, porém, muitas vezes, esse método acaba tornando-se caro computacionalmente falando, por essa razão, será apresentado neste trabalho métodos iterativos para a resolução deste mesmo problema. Tais métodos iterativos começam com uma proposta de solução inicial aproximada qualquer $\mathbf{x}^0$ para $\mathbf{x}$ e gera uma sequência de vetores $\{\mathbf{x}\}_{k=0}^{\infty}$ que converge para $\mathbf{x}$. Serão apresentados, no entanto, o método de Jacobi, método de Gauss-Siedel e o método SOR para a resolução destes problemas. Os métodos apresentados à seguir seguem como referência mãe o livro do Burden, citado em [1] no índice de referências.

2. Método de Jacobi

2.1. Motivação

Considere um sistema linear $A\mathbf{x} = \mathbf{b}$ onde A é uma matriz não singular tais que $a_{ii} \neq 0$ para $i = 1, \cdots, n$. Podemos escrever cada equação deste sistema linear da seguinte forma

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \quad\vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots & \quad\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{bn}x_n &= b_n \end{cases} \tag{1}$$

Ou ainda

$$\begin{cases} x_1 = \dfrac{b_1}{a_{11}} - \left(\dfrac{a_{12}x_2}{a_{11}} + \dfrac{a_{13}x_3}{a_{11}} + \cdots + \dfrac{a_{1n}x_n}{a_{11}} \right) \\ x_2 = \dfrac{b_2}{a_{22}} - \left(\dfrac{a_{21}x_1}{a_{22}} + \dfrac{a_{23}x_3}{a_{22}} + \cdots + \dfrac{a_{2n}x_n}{a_{22}} \right) \\ \quad\vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \\ x_n = \dfrac{b_n}{a_{nn}} - \left(\dfrac{a_{n1}x_1}{a_{nn}} + \dfrac{a_{n2}x_2}{a_{nn}} + \cdots + \dfrac{a_{n,n-1}x_{n-1}}{a_{nn}} \right) \end{cases} \tag{2}$$

Dada uma proposta de solução inicial $\mathbf{x}^0$ para o problema, o método de Jacobi define uma sequência de vetores $\{\mathbf{x}^k\}_{k>0}$ através da seguinte relação de recorrência

$$\begin{cases} x_1^{k+1} = \dfrac{b_1}{a_{11}} - \left(\dfrac{a_{12}x_2^k}{a_{11}} + \dfrac{a_{13}x_3^k}{a_{11}} + \cdots + \dfrac{a_{1n}x_n^k}{a_{11}} \right) \\ x_2^{k+1} = \dfrac{b_2}{a_{22}} - \left(\dfrac{a_{21}x_1^k}{a_{22}} + \dfrac{a_{23}x_3^k}{a_{22}} + \cdots + \dfrac{a_{2n}x_n^k}{a_{22}} \right) \\ \quad\vdots \quad\quad\quad\quad\quad\quad \vdots \quad\quad\quad\quad\quad\quad \vdots \quad\quad\quad\quad\quad\quad \vdots \\ x_n^{k+1} = \dfrac{b_n}{a_{nn}} - \left(\dfrac{a_{n1}x_2^k}{a_{nn}} + \dfrac{a_{n2}x_3^k}{a_{nn}} + \cdots + \dfrac{a_{n,n-1}x_{n-1}^k}{a_{nn}} \right) \end{cases} \tag{3}$$

Podemos simplificar o Sistema Linear acima dizendo que cada equação é da forma

$$x_i^k = \frac{1}{a_{ii}} \sum_{\substack{i=1 \\ i \neq j}}^{n} \left(-a_{ij} x_j^{k-1} \right) + b_i$$

para $i = 1, 2, \cdots, n$.

Em geral, métodos iterativos para a solução de sistemas lineares envolve um processo que converte o sistema original $A\mathbf{x} = \mathbf{b}$ em um sistema equivalente da forma $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ para uma matriz T fixada e um vetor $\mathbf{c}$. Basicamente, após a primeira iteração com o vetor inicial $\mathbf{x}^0$ a sequência de soluções aproximadas é gerada calculando $\mathbf{x}^k = T\mathbf{x}^{k-1} + \mathbf{c}$. Se você observar bem, essa ideia lembra muito o Método do Ponto Fixo para Zeros de Funções, na realidade, funciona de maneira análoga como veremos em breve. O Método do Ponto Fixo será apenas citado aqui porém, aos curiosos, no capítulo 2 da Referência 1 [Burden] o leitor poderá conferir com mais informações como funciona o método.

Sendo assim, o método de Jacobi também pode ser escrito na forma $\mathbf{x}^k = T\mathbf{x}^{k-1} + \mathbf{c}$ separando A em uma matriz diagonal e duas partes não-diagonais. Para visualizar melhor essa ideia, tome D uma matriz diagonal cujas as entradas das diagonais são as mesmas das diagonais de A. Seja $-L$ e $-D$ matrizes estritatmente triangular inferior e superior de A respectivamente, temos então que $A = D - L - U$. Desta forma, o sistema $A\mathbf{x} = \mathbf{b}$ é transformado em $(D - L - U)\mathbf{x} = \mathbf{b}$. Operando sob esta equação obtemos ainda

$$(D - L - U)\mathbf{x} = \mathbf{b}$$

$$D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} = D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b}$$

Como D também é uma matriz não-singular, a sua inversa existe. Desta forma, o Método de Jacobi é expressada matricialmente da seguinte maneira:

$$\mathbf{x}^k = D^{-1}(L + U)\mathbf{x}^{k-1} + D^{-1}\mathbf{b} \tag{4}$$

para $k = 1, 2, \cdots$

Introduzimos então a notação $T_j = D^{-1}(L + U)$ e $\mathbf{c}_j = D^{-1}\mathbf{b}$ que nos dá uma nova forma de representar o Método de Jacobi matricialmente da seguinte forma

$$\mathbf{x}^k = T_j\mathbf{x}^{k-1} + \mathbf{c}_j \tag{5}$$

Para esse método, iteramos uma nova solução enquanto for satisfeito o critério de parada

$$\frac{\|\mathbf{x}^k - \mathbf{x}^{k-1}\|_\infty}{\|\mathbf{x}^k\|_\infty} < \tau$$

onde $\tau > 0$.

2.1.1. Exemplo

Vamos utilizar o Método de Jacobi pra encontrar as soluções para o seguinte sistema de equações:

$$\begin{cases} 2x_1 + x_2 = 1 \\ 3x_1 + 4x_2 = -1 \end{cases}$$

como proposta de solução inicial, tomemos $\mathbf{x}^0 = (0, 0)^t$. Logo, a primeira iteração é calculada da seguinte forma

$$\begin{cases} x_1^1 = \dfrac{1}{2} \ - \ x_2^0 \\ x_2^1 = \dfrac{1}{3} - \dfrac{2x_1^0}{3} \end{cases} \Rightarrow \mathbf{x}^1 = \left(\frac{1}{2}, -\frac{1}{4}\right)^t$$

Repetimos o processo, agora com $\mathbf{x}^1$ para encontrar $\mathbf{x}^2$ e assim sucessivamente. Utilizando a linguagem Julia, foi observado que com o critério de parada $\tau = 10^{-8}$ o Método convergiu na 38^o iteração, onde $\mathbf{x}^{38} = (1, -1)^t$.

k	0	1	5	10	15	20	25
x_1	0.000000	0.500000	0.929688	0.992584	0.999479	0.999945	0.999996
x_2	-0.000000	-0.250000	-0.894531	-0.992584	-0.999218	-0.999945	-0.999994

O algoritmo implementado em Julia Lang pode ser visualizado em um repositório do Github disponível em (`https://github.com/gustavosarturi/gustavosarturi/edit/master/Julia/An%C3%A1lise%20Num%C3%A9rica/Jacobi.jl`).

No Reposítório, o leitor pode conferir as imagens que representam a convergência do Método.

2.2. Algoritmo: Método Iterativo de Jacobi

Algorithm 1 Método Iterativo de Jacobi

function JACOBI($A = (a_{ij})_{n\times n}$, $XO = x^0 \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, MaxTol, MaxIt)
  $k \leftarrow 1$
  while $k \leq MaxIt$ **do**
    for i **do**1n
      $\mathrm{x}_i = \dfrac{1}{a_{ii}} \left[b_i - \sum_{j=1, i\neq j}^{n} (a_{ij} XO_j) \right]$
    if $\|\mathbf{x} - \mathbf{XO}\| < MaxTol$ **then**
      Retorna x
    $k \leftarrow k + 1$
    for i **do**1n
      $XO_i = x_i$
    Return: O número de iterações foi excedido!
    =0

3. Método de Gauss-Siedel

O método de Gauss-Seidel é um aprimoramento do Método de Jacobi apresentado anteriormente. Lembrando que no método de Jacobi, cada equação do sistema linear estava na seguinte forma

$$x_i^k = \frac{1}{a_{ii}} \left[\sum_{\substack{i=1 \\ i\neq j}}^{n} \left(-a_{ij} x_j^{k-1} \right) + b_i \right]$$

o método de Gauss-Seidel basicamente utiliza da componente anterior para calcular parte da nova iteração. Ou seja

$$x_i^k = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} \left(a_{ij} x_j^k \right) - \sum_{j=i+1}^{n} \left(a_{ij} x_j^{k-1} \right) \right] \tag{6}$$

para cada $i = 1, 2, \cdots, n$. Em forma de sistema linear temos

$$\begin{cases} x_1^{k+1} = \dfrac{b_1}{a_{11}} - \left(\dfrac{a_{12}x_2^k}{a_{11}} + \dfrac{a_{13}x_3^k}{a_{11}} + \cdots + \dfrac{a_{1n}x_n^k}{a_{11}} \right) \\ x_2^{k+1} = \dfrac{b_2}{a_{22}} - \left(\dfrac{a_{21}x_1^{k+1}}{a_{22}} + \dfrac{a_{23}x_3^k}{a_{22}} + \cdots + \dfrac{a_{2n}x_n^k}{a_{22}} \right) \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_n^{k+1} = \dfrac{b_n}{a_{nn}} - \left(\dfrac{a_{n1}x_2^{k+1}}{a_{nn}} + \dfrac{a_{n2}x_3^{k+1}}{a_{nn}} + \cdots + \dfrac{a_{n,n-1}x_{n-1}^{k+1}}{a_{nn}} \right) \end{cases} \tag{7}$$

Note, novamente, que utilizamos parte da solução da iteração anterior para calcular a próxima iteração. Para esse método, o critério de parada segue o mesmo do Método de Jacobi, ou seja

$$\frac{\|\mathbf{x}^k - \mathbf{x}^{k-1}\|_\infty}{\|\mathbf{x}^k\|_\infty} < \tau$$

onde $\tau > 0$.

Para chegarmos na forma matricial do método de Gauss-Siedel, multiplicamos a equação 8 por a_{ii} e depois, separamos os membros que contém x^k dos que contém x^{k-1}. Lembrando que estamos assumindo que $a_{ii} \neq 0$. Obtendo, no entanto a seguinte relação

$$\sum_{j=1}^{i} \left(a_{ij}x_j^k \right) = b_i - \sum_{j=i+1}^{n} \left(a_{ij}x_j^{k-1} \right) \tag{8}$$

Note que o índice de um do somatório à esquerda foi incrementado por causa da outra parcela. No entanto, temos, portanto que

$$\begin{array}{rcc} a_{11}x_1^k & = & b_1 - a_{12}x_2^{k-1} - a_{13}x_3^{k-1} - \cdots - a_{1n}x_n^{k-1} \\ a_{11}x_1^k + a_{12}x_2^k & = & b_2 - a_{23}x_3^{k-1} - \cdots - a_{1n}x_n^{k-1} \\ \vdots & \vdots & \vdots \\ a_{n1}x_1^k + a_{n2}x_2^k + \cdots + a_{nn}x_n^k & = & b_n \end{array} \tag{9}$$

Assim, com as definições das matrizes D, L e U citadas anteriormente, temos que o método de Gauss-Siedel pode ser representado matricialmente por

$$(D - L)\mathbf{x}^k = U\mathbf{x}^{k-1} + \mathbf{b} \tag{10}$$

Assim, temos que a $k-$ésima iteração, matricialmente é dada por

$$\mathbf{x}^k = (D - L)^{-1}U\mathbf{x}^{k-1} + (D - L)^{-1}\mathbf{b} \tag{11}$$

Denotando $T_g = (D - L)^{-1}U$ e $\mathbf{c}_g = (D - L)^{-1}\mathbf{b}$, o Método de Gauss Siedel fica na forma

$$\mathbf{x}^k = T_g\mathbf{x}^{k-1} + \mathbf{c}_g \tag{12}$$

Assim, para que a matriz $D - L$ seja singular, é necessário e suficiente que $a_{ii} \neq 0$, o que, por hipótese, já ocorre.

No algoritmo abaixo, MaxTol é a tolerância máxima que é utilizado como um critério de parada, e, MaxIt é o número máximo de iterações.

3.1. Exemplo

Voltamos ao exemplo anterior onde o objetivo era encontrar uma solução para

$$\begin{cases} 2x_1 + x_2 = 1 \\ 3x_1 + 4x_2 = -1 \end{cases}$$

Utilizando Julia Lang, obtemos que o Método Converge já na 15^o iteração. A tabela abaixo mostra os valores obtidos pelo método.

k	0	1	2	5	10	14	15
x_1	0.000000	0.500000	0.812500	0.990112	0.999927	0.999999	0.999999
x_2	-0.000000	-0.625	-0.859375	-0.992584	-0.999945	-0.999999	-1.0

No Repositório citado nas referências, o leitor pode conferir as imagens que representam a convergência do Método.

3.2. Algoritmo: Método Iterativo de Gauss-Seidel

Algorithm 2 Método Iterativo de Gauss-Seidel

function GAUSSSEIDEL($A = (a_{ij})_{n\times n}$, $XO = x^0 \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, MaxTol, MaxIt)
  $k \leftarrow 1$
  while $k \leq MaxIt$ **do**
    for i **do**1n
      $x_i = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1}(a_{ij}x_j) - \sum_{j=i+1}^{n}(a_{ij}XO_j)\right]$
    if $\|\mathbf{x} - \mathbf{XO}\| < MaxTol$ **then**
      Retorna x
    $k \leftarrow k + 1$
    for i **do**1n
      $XO_i = x_i$
    Saída: O número de iterações foi excedido!

4. Critérios de Convergências

Nesta seção, serão apresentados Teoremas, Corolários e Lemas do capítulo 7.3 da Referência Principal [Burden]. As demonstrações serão omitidas no momento, porém, as demonstrações podem ser encontradas na mesma referência. Basicamente, para o estudo de convergência dos Métodos Iterativos, precisamos analizar a equação que encontramos em nossos métodos dada por

$$\mathbf{x}^k = T\mathbf{x}^{k-1} + \mathbf{c}$$

onde $\mathbf{x}^0$ é arbitrário.

Definição 4.1 *O raio espectral de uma matriz A, denotado por $\rho(A)$ é definido como sendo $\rho(A) = \max(|\lambda|)$ onde λ são os autovalores da matriz A.*

Lema 4.1 *Se o raio espectral $\rho(T) < 1$, então, $(1-T)^{-}1$ existe, além disto,*

$$(1-T)^{-1} = \sum_{j=0}^{\infty} T^j$$

Teorema 4.1 *Para qualquer $\boldsymbol{x}^0 \in \mathbb{R}^n$, a sequência $\{\boldsymbol{x}^k\}_\infty^\infty$ definida por $\boldsymbol{x}^k = T\boldsymbol{x}^{k-1} + \boldsymbol{c}$ para cada $k \geq 1$ converge a uma única solução de $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$ se, e somente se, o raio espectalda matriz for estritamente menor que 1, ou seja, $\rho(T) < 1$.*

O Teorema acima é um forte resultado pois, dita uma condição necessária e suficiente para a convergência de um Método Iterativo para a resolução de Sistemas Lineares.

Teorema 4.2 *Se A é estritamente diagonal, então, para qualquer valor inicial $\boldsymbol{x}^0$, tanto o Método de Jacobi quanto o Método de Gauss-Seidel dará uma sequência $\{\boldsymbol{x}^k\}_{k=0}^\infty$ que converge para uma solução única de $A\boldsymbol{x} = \boldsymbol{b}$.*

5. Método SOR

O Método de SOR (Sobre-Relaxamento Sucessivo) é um melhoramento do Método de Gauss-Seidel para a solução de sistemas lineares. O Método SOR gera uma sequência tal que cada elemento é da forma:

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

Onde ω é uma constante que estudaremos a seguir que poderá acelerar a convergência para a solução do sistema linear.

Para chegarmos no Método, precisamos antes definir a ideia de vetor resíduo. Se suponharmos que $x' \in \mathbb{R}^n$ é uma aproximação para a solução do sistema linear definido por $A\mathbf{x} = \mathbf{b}$, então, o vetor resíduo para $\mathbf{x}'$ com respeito ao seu sistema é definido como sendo $\mathbf{r} = \mathbf{b} - A\mathbf{x}'$. Nos Métodos apresentados até agora, o vetor resíduo está associado à cada iteração calculado. O verdadeiro objetivo deste vetor é gerar uma sequência de aproximações que irá fazer com que esse vetor resíduo convirja rapidamente para zero.

Suponha que $\mathbf{r}_i^k = (r_{1i}^k, \cdots, r_{2i}^k)^t$. O vetor resíduo do Método de Gauss Seidel correspondente à uma aproximação de solução é uma iteração $\mathbf{x}_i^k$ definida por

$$\mathbf{x}_i^k = (x_1^k, x_2^k, \cdots, x_{i-1}^k, x_i^k, \cdots, x_n^{k-1})^t$$

assim, a m-ésima equação correspondente a $\mathbf{r}_i^k$ é

$$\mathbf{r}_{mi}^k = b_m - \sum_{j=1}^{i-1} a_{mj}x_j^k - \sum_{j=i}^{n} a_{mj}x_j^{k-1}$$

ou equivalentemente, abrindo um dos termos do somatório à direita

$$\mathbf{r}_{mi}^k = b_m - \sum_{j=1}^{i-1} a_{mj}x_j^k - \sum_{j=i+1}^{n} a_{mj}x_j^{k-1} - a_{mi}x_i^{k-1} \tag{14}$$

para cada $m = 1, \cdots, n$.

Em particular, se somarmos $a_{mi}x_i^{k-1}$ em ambos os lados, o i-ésimo componente do vetor $\mathbf{r}_i^k$ é

$$a_{mi}x_i^{k-1} + \mathbf{r}_{ii}^k = b_m - \sum_{j=1}^{i-1} a_{mj}x_j^k - \sum_{j=i+1}^{n} a_{mj}x_j^{k-1} \tag{15}$$

Agora, lembremos que no Método de Gauss-Seidel, x_i^k é escolhide de forma tal que

$$x_i^k = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} \left(a_{ij} x_j^k \right) - \sum_{j=i+1}^{n} \left(a_{ij} x_j^{k-1} \right) \right]$$

se multiplicarmos que a_{ii} ambos os lados, e substituindo seu valor na equação em que estávamos construindo, temos que ela pode ser reescrita da seguinte maneira

$$a_{ii} x_i^{k=1} + r_{ii}^k = a_{ii} x_i^k$$

e consequentemente, temos que o Método de Gauss-Seidel pode ser caracterizado escolhendo x_i^k que satisfaz a seguinte equação:

$$x_i^k = x_i^{k-1} + \frac{r_{ii}^k}{a_{ii}} \tag{16}$$

Uma outra conexão entre o vetor resíduo e o Método de Gauss-Seidel pode ser encontrado considerando $\mathbf{r}_{i+1}^k$ associado com o vetor $\mathbf{x}_{i+1}^k = (x_1^k, \cdots, x_i^k, x_{j+1}^{k-1}, \cdots, x_n^{k-1})^t$. Pela equação 14 temos que o i-ésimo componente de $\mathbf{r}_{i+1}^k$ é

$$\mathbf{r}_{i,j+1}^k = b_i - \sum_{j=1}^{i} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1}$$

pelo já calculado acima, então, temos algo análogo

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

Da maneira que x_i^k é escolhida pelo Método de Gauss-Seidel, percebemos que $r_{i,i+1}^k = 0$. Sendo assim, cada iteração do Método de Gauss Seidel é caracterizada por escolher cada x_{i+1}^k de odo que o i-ésimo componente do vetor residual é zero.

Escolhendo x_{i+1}^k, uma das coordenadas do vetor residual é zero, de qualquer forma, não é necessariamente a maneira mais eficiente de se reduzir a norma do vetor residual $\mathbf{r}_{i+1}^k$. Se modificarmos o Método de Gauss-Seidel dada pela equação 16 por

$$x_i^k = x_i^{k-1} + \omega \frac{r_{ii}^k}{a_{ii}} \tag{17}$$

para algum ω positivo que reduz a norma do vetor resíduo e acelera significativamente a convergência do Método. Estamos interessados em escolher um ω tal que $\omega > 1$, tal procedimento é chamado de Método de Sobre-Relaxamento, e é utilizado para acelerar a convergência de sistemas lineareas que são convergentes pelo Método de Gauss Seidel. Tal Método é abreviado por SOR (Do inglês, Successive Over-Relaxation).

Para ilustrar a forma matricial do Método SOR, note que usando a equação 15, podemos reformular 17 para

$$x_i^k = (1 - \omega) x_i^{k-1} + \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^{k-1} \right]$$

reescrevendo a equação acima, obtemos, ainda

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

que tem a seguinte forma matricial

$$(D - \omega L)\mathbf{x}^k = [(1-\omega)D + \omega U]\mathbf{x}^{k-1} + \omega\mathbf{b}$$

ou seja

$$\mathbf{x}^k = (D - \omega L)^{-1}[(1-\omega)D + \omega U]\mathbf{x}^{k-1} + (D - \omega L)^{-1}\omega\mathbf{b}$$

Tomando $T_\omega = (D - \omega L)^{-1}[(1-\omega)D + \omega U]$ e $\mathbf{c}_\omega = \omega U]\mathbf{x}^{k-1} + (D - \omega L)^{-1}\omega\mathbf{b}$ reduzimos então a expressão para o Método SOR para a forma

$$\mathbf{x}^k = T_\omega \mathbf{x}^{k-1} + \mathbf{c}_\omega \tag{18}$$

No entanto, apesar de ω ser arbitrário, a melhor escolha para ω irá se basear nos seguintes teoremas

Teorema 5.1 *Se $a_{ii} \neq 0$ para cada $i = 1, 2, \cdots, n$ então $\rho(T_\omega) \leq |\omega - 1|$. Isso implica que o Método SOR só converge se $0 < \omega < 2$.*

Teorema 5.2 *Se A é uma matriz definida positiva e $0 < \omega < 2$, então o método SOR converge para qualquer aproximação inicial $\boldsymbol{x}^0$.*

Teorema 5.3 *Se A é uma Matriz definida positiva e tridiagonal, então $\rho(T_g) = [\rho(T_j)]^2 < 1$ e a melhor escolha para ω para o Método SOR é*

$$\omega = \frac{2}{1 + \sqrt{1 - [\rho(T_j)]^2}}$$

para essa escolha de ω temos que $\rho(T_\omega) = \omega - 1$

5.1. Exemplo

Para exemplificar a teoria, vamos resolver o seguinte sistema lineaar pelo Método SOR

$$\begin{cases} 9x_1 + 4x_2 = 20 \\ 4x_1 + 9x_2 - x_3 = 12 \\ -x_2 + 9x_3 = 51 \end{cases} \tag{19}$$

que, em sua forma matricial é representada por

$$\begin{pmatrix} 9 & 4 & 0 \\ 4 & 9 & -1 \\ 0 & -1 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 20 \\ 12 \\ 51 \end{pmatrix} \tag{20}$$

para se calcular $\rho(T_j)$, precisamos, no entanto, saber os valores de T_j. No caso, $T_j = D^-1(L + U)$

$$T_j = \begin{pmatrix} 1/9 & 0 & 0 \\ 0 & 1/9 & 0 \\ 0 & 0 & 1/9 \end{pmatrix} \left(\begin{pmatrix} 0 & 0 & 0 \\ -4 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & -4 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \right) = \begin{pmatrix} 0 & 4/9 & 0 \\ 4/9 & 0 & 1/9 \\ 0 & 1/9 & 0 \end{pmatrix} \tag{21}$$

Calculando $\det(T_j - I) = 0$, obtemos $-\lambda(\lambda^2 - 17/81) = 0$, cujas as raízes, são, respectivamente $\lambda_1 = 0$, $\lambda_2 = -\sqrt{17}/9$, $\lambda_3 = \sqrt{17}/9$. Logo, $\rho(T_j) = \max(|\lambda_1|, |\lambda_2|, |\lambda_3|) = \sqrt{17}/9$. Utilizando o último Teorema enunciado, temos no entanto que o melhor ω a se escolher é

$$\omega = \frac{2}{1 + \sqrt{1 - \left(\sqrt{17}/9\right)^2}} \approx 1.0588235294117647$$

.

Assim, temos a seguinte relação de recorrência

$$\begin{aligned} x_1^k &= (1-\omega)x_1^{k-1} + \frac{\omega}{9}(20 - 4x_2^{k-1}) \\ x_2^k &= (1-\omega)x_2^{k-1} + \frac{\omega}{9}(12 - 4x_1^k + x_3^{k-1} \\ x_3^k &= (1-\omega)x_3^{k-1} + \frac{\omega}{9}(51 + x_2^k) \end{aligned} \tag{22}$$

5.1.1. Algoritmo: Método SOR

Algorithm 3 Método Iterativo de Gauss-Seidel

function GAUSSSEIDEL($A = (a_{ij})_{n\times n}$, $XO = x^0 \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, $\omega > 0, MaxTol, MaxIt$)
  $k \leftarrow 1$
  while $k \leq MaxIt$ **do**
    for i **do**ln
      $\mathrm{x}_i = (1-\omega)XO_i + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1}(a_{ij}x_j) - \sum_{j=i+1}^{n}(a_{ij}XO_j)\right]$
    if $\|\mathbf{x} - \mathbf{XO}\| < MaxTol$ **then**
      Retorna x
    $k \leftarrow k + 1$
    for i **do**ln
      $XO_i = x_i$
    Saída: O número de iterações foi excedido!

5.2. Conclusões finais

Dos Métodos apresentados, o Método SOR por ser um aperfeiçoamento do método de Gauss-Seidel, foi o que trouxe melhores resultados em relação a sua convergência. Foram realizados testes com matrizes tridiagonais e esparsas aleatórias em Julia Lang. Em particular, os três métodos foram submetidos a resolver o seguinte Sistema Linear:

$$\begin{cases} 9x_1 + 4x_2 = 20 \\ 4x_1 + 9x_2 - x_3 = 12 \\ -x_2 + 9x_3 = 51 \end{cases} \tag{23}$$

Da qual, o Método SOR, como já previsto, se saiu melhor. Com uma tolerância de $10^{(-10)}$ o Método SOR convergiu em apenas 12 iterações, enquanto que o Método de Gauss-Seidel convirgiu em 18 e o de Jacobi em 31. Porém, para encontrar o melhor valor de ω para o Método SOR como previsto na teoria, exige-se o cálculo de um determinante para encontrar os seus autovalores, o que, dependendo do método pode custar caro computacionalmente falando, por esse motivo, o Método Gauss-Seidel torna-se mais eficaz apesar da velocidade de convergência ser inferior a do Método SOR. Nestes casos, Métodos Iterativos para o cálculo de Autovalores podem auxiliar na melhor acurácia para encontrar os autovalores da matriz do sistema, como por exemplo, o Método da Potência.

Referências

[1] R. L. Burden e J. D. Faires, Numerical Analysis, 9a ed. Cengage Learning, 2010.

[2] The Julia Language. url: julialang.org.

[3] Repositório Github - Gustavo H S Sarturi: `https://github.com/gustavosarturi/gustavosarturi/edit/master/Julia/An%C3%A1lise%20Num%C3%A9rica/Jacobi.jl`