MAC 300 – Métodos Numéricos de Álgebra Linear
Nome: Vítor Kei Taira Tamada
NUSP: 8516250
Exercício-programa 2 – Métodos iterativos para sistemas lineares: Gradientes Conjugados

Resumo sobre Gradientes Conjugados

Motivação: Deseja-se resolver uma equação de sistemas lineares da forma $Ax = b$, sendo A uma matriz $n \times n$, não-singular, esparsa e muito grande, e x e b são vetores de tamanho n. Em outras palavras, equações que métodos diretos não conseguem resolver ou são muito ineficientes e que métodos iterativos se apresentam como uma alternativa melhor.

Método iterativo: Seja x o vetor solução da equação $Ax = b$. O método iterativo consiste em encontrar um vetor $x^{(k)}$ suficientemente próximo de x tal que ele pode ser aceito como a solução da equação. Ou seja, existe um resíduo $r^{(k)}$ denotado por

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

tal que quanto mais próximo $r^{(k)}$ estiver de 0, mais próximo $x^{(k)}$ está de x.

Gradiente: Considere um ponto $p = (x_0, x_1, \ldots, x_n)$ no espaço. Gradiente é o vetor que indica a direção e o sentido para o qual deve partir desse ponto p para que haja o maior aumento em seu valor.

Método gradiente: Seja A uma matriz de tamanho $n \times n$, definida positiva e simétrica. O método gradiente consiste em resolver a equação de sistema lineares $Ax = b$ enxergando-a como um problema de minimzação. Seja $J: \mathbb{R}^n \rightarrow \mathbb{R}$ tal que

$$J(y) = \frac{1}{2} * (y^T A y) - y^T b$$

O vetor y que minimizar a função J será a solução de $Ax = b$; ou seja, esse y será igual ou suficientemente próximo de x.

O método gradiente recebe esse nome uma vez que, ao calcular o gradiente de J, obtém-se a seguinte equação:

$$grad(J) = Ay - b$$

Pela definição de gradiente, sabemos que $grad(J)$ é o vetor que indica o aumento da diferença $Ay - b$. Além disso, como o resíduo de uma aproximação é dado por $r = b - Ax$, temos que $grad(J)$ é o negativo do resíduo de y. Em outras palavras, enquanto $grad(J)$ é o vetor que aponta a direção e sentido de maior aumento da diferença entre Ay e b, fazendo com que y seja cada vez mais diferente do vetor solução x, o vetor r aponta para a direção e sentido em que essa diferença é a menor possível – o sentido oposto.
Logo, o ponto em que $grad(J)$ for igual a zero, e, portanto, o resíduo for zero, será a solução de $Ax = b$.

***Descent method*:** Para se encontrar o y tal que $grad(J)$ seja (ou pelo menos tenda) a zero,

utiliza-se métodos iterativos. A partir de um vetor $x^{(0)}$, gera-se uma sequência de iterações $x^{(0)}$, $x^{(1)}$, $x^{(2)}$, … tal que $J(x^{(k+1)}) <= J(x^{(k)})$, tendo preferência para $J(x^{(k+1)}) < J(x^{(k)})$. Dessa forma, eventualmente chega-se a um $x^{(k)}$ tal que $Ax^{(k)} = b$ uma vez que $J(x^{(k)})$ será mínimo.

Para ir de $x^{(k)}$ para $x^{(k+1)}$, é necessário ter duas informações:
1) a direção de busca escolhida;
2) uma linha de busca na direção escolhida.

Escolher uma direção de busca consiste em escolher um vetor $p^{(k)}$ que indica a direção que seguirá para sair de $x^{(k)}$ e ir para $x^{(k+1)}$. Isso é mais fácil de se imaginar no plano cartesiano, apesar de não se restringir ao mesmo.

Uma vez que temos o vetor $p^{(k)}$, é necessário escolher um vetor $x^{(k+1)}$ presente na linha $\{x^{(k)} + a*p^{(k)}\}$ sendo a um número real. Logo, temos a seguinte equação geral para encontrar $x^{(k+1)}$:

$$x^{(k+1)} = x^{(k)} + a_k p^{(k)}$$

para algum a_k real. O processo de escolha desse a_k específico dentre todos os a reais é a linha de busca mencionada acima.

Uma vez que se deseja encontrar um $x^{(k+1)}$ tal que $J(x^{(k+1)}) <= J(x^{(k)} + a_k p^{(k)})$, é possível garantir essa inequação por meio da equação $J(x^{(k+1)}) = min_a\ J(x^{(k)} + ap^{(k)})$, sendo a um número real. Nesse caso, a busca é considerada *exata*. A fórmula para encontrar esse a é dada pela equação:

$$a_k = (p^{(k)T} r^{(k)})/(p^{(k)T} A p^{(k)})$$

sendo $r^{(k)} = b - Ax^{(k)}$.

É bom destacar o fato de que $a_k = 0$ é um valor que pode ser desconsiderado já que não buscamos um $x^{(k+1)}$ que seja igual a $x^{(k)}$. Isso só ocorreria se $p^{(k)T} r^{(k)} = 0$, o que indicaria que os vetores em questão são ortogonais entre si. Ou seja, basta escolher um $p^{(k)}$ que não seja ortogonal a $r^{(k)}$ e $r^{(k)}$ não ser igual a zero. De fato, se $r^{(k)} = 0$, então $x^{(k)}$ é a solução; logo, não é necessário continuar a busca nesse caso.

***Steepest Descent method*:** Este método, que é um caso específico do *descent method*, escolhe $p^{(k)} = r^{(k)}$ como direção de busca para realizar a linha de busca exata. Como foi mencionado anteriormente, $r^{(k)} = -grad(J(x^{(k)}))$, ou seja, é o vetor que mais reduz a diferença $b - Ay$, que faz a diferença ter a maior queda. Logo, o nome *steepest descent*.

Como foi visto anteriormente, a solução aproximada para cada passo em encontrar um $x^{(k+1)}$ suficientemente próximo de x, utilizamos $x^{(k+1)} = x^{(k)} + a_k p^{(k)}$, sendo a_k como visto logo acima. Além disso, também precisamos ter $r^{(k)}$, que é dado por $r^{(k)} = b - Ax^{(k)}$. Como realizar a multiplicação matriz-vetor é, normalmente, uma operação muito cara e seria necessário fazer duas para ter $r^{(k)}$ e $x^{(k+1)}$, é possível utilizar a recursão

$$r^{(k+1)} = r^{(k)} - a_k A p^{(k)}$$

obtidas da substituição de $r^{(k)} = b - Ax^{(k)}$ em $x^{(k+1)} = x^{(k)} + a_k p^{(k)}$.

Para facilitar a leitura e escrita, utilizaremos $q^{(k)} = Ap^{(k)}$

Como é possível ver no *descent method* e no *steepest descent method*, o $x^{(k)}$ e o $r^{(k)}$ são utilizados apenas para encontrar $x^{(k+1)}$ e $r^{(k+1)}$; ou seja, $x^{(k)}$ e $r^{(k)}$ são utilizados uma única vez, de forma que, no algoritmo, $x^{(k+1)}$ e $r^{(k+1)}$ são escritos em cima de $x^{(k)}$ e $r^{(k)}$ fazendo com que não haja memória dos vetores anteriores.

<u>Método do gradiente conjugado</u>: Este método é uma variação do *steepest descent method*, mas que possui memória dos vetores anteriores – o que faz deste um método melhor.

Porém, o que seria essa "memória de vetores anteriores" e por que isso faz o método do gradiente conjugado ser superior?

Essa memória de vetores anteriores – atributo que diferencia este método do *steepest descent* – é uma variável β que armazena a divisão de v_{k+1}/v_k, onde $v_k = r^{(k)T}r^{(k)}$ a cada iteração. Essa variável é utilizada para atualizar o valor do vetor $p^{(k)}$ de uma maneira diferente também. Normalmente, no *steepest descent method*, $p^{(k)}$ recebe $r^{(k)}$ apenas ($p^{(k)} = r^{(k)}$). No método do gradiente conjugado, $p^{(k)}$ recebe a soma $r^{(k)} + \beta p^{(k)}$ ($p^{(k)} = r^{(k)} + \beta p^{(k)}$).

Por conta disso, o método recebe o nome de gradiente **conjugado** – porque os vetores gradientes (negativo dos vetores resíduo) estão emparelhados, juntos.

Agora, só falta explicar o porquê de o método dos gradientes conjugados ser melhor que o *steepest descent method*.

Os dois métodos mencionados têm um espaço S_j gerado pelas direções de busca $p^{(0)}$, …, $p^{(j-1)}$. Entretanto, as direções de busca que cada um utiliza são diferentes apesar de o espaço gerado ser o mesmo. Como o *steepest descent method* faz busca de linha exata, $x^{(j)}$ minimiza a função J ao longo das j vezes que faz $x^{(k)} + ap^{(k)}$, $k = 0, \ldots, j - 1$, enquanto o método de gradientes conjugados consegue escolher o $x^{(j)}$ que minimiza J de todo o espaço S_j. Logo, o método dos gradientes conjugados consegue escolher o melhor $x^{(j)}$.