

**MAP-2121 - Segundo exercício programa - 2011**  
**Resolvendo sistemas esparsos por Gradientes Conjugados**

**Instruções gerais -**

Os exercícios computacionais pedidos na disciplina Cálculo Numérico têm por objetivo fundamental familiarizar o aluno com problemas práticos que requeiram técnicas numéricas em sua solução. Neste segundo programa sua tarefa será aprender e implementar um novo algoritmo para resolução de sistemas lineares, além de aplicá-lo na solução de dois problemas.

Seu programa deve ser entregue no sistema moodle **map2121.ime.usp.br** até o dia 17 de novembro. Não deixe para fazê-lo no final do prazo. O programa deve ser escrito em Linguagem C e ser compilado e executado com o compilador disponível através da página da disciplina (**www.ime.usp.br/~map2121**). Caso você desenvolva seu programa em outro compilador, confira se ele também compila e executa no compilador indicado. Programas que não compilarem terão notas muito baixas. Ao desenvolver seu projeto você possivelmente trocará idéias com seus colegas. Esta interação é saudável e desejável, vocês estarão aprendendo mais. O seu programa deve, no entanto, ser desenvolvido por você individualmente, para que você realmente saiba fazê-lo. Haverá controle de cópias e caso estas sejam detectadas, os envolvidos terão nota zero no programa e o caso será levado à coordenação do biênio. Esperamos sinceramente não encontrar nenhum caso deste tipo. Bom trabalho e divirta-se com sua tarefa!

**Método dos Gradientes Conjugados**

O Método dos Gradientes Conjugados consiste em um algoritmo para solução de sistemas lineares cujas matrizes são simétricas positivas definidas. Uma matriz real  $A$  de tamanho  $n \times n$  é simétrica quando é igual à sua transposta.  $A$  é positiva definida se o produto escalar  $\langle Ax, x \rangle$  for positivo para todo vetor não nulo  $x \in R^n$  (o produto escalar entre dois vetores é o usual, definido como:  $\langle w, z \rangle = \sum_{i=1}^n w_i z_i$ ). Quando  $A$  é simétrica positiva definida podemos definir um novo produto interno induzido pela matriz  $A$ :  $\langle x, y \rangle_A = \langle Ax, y \rangle$ , onde  $\langle Ax, y \rangle$  denota o produto escalar usual entre os vetores  $Ax$  e  $y$ . A comutatividade do produto interno induzido por  $A$  decorre da simetria da matriz, enquanto que o fato da matriz ser positiva definida garante que  $\langle x, x \rangle_A > 0$ , se  $x \neq 0$ .

O método para a solução de uma equação  $Ax = b$  parte de uma aproximação inicial qualquer  $x_0$ , a partir da qual define-se uma direção inicial:

$$d_0 = r_0 = b - Ax_0$$

Se denotarmos a solução exata do sistema linear por  $\bar{x}$ , obtemos a seguinte relação entre o resíduo  $r_0$  e o erro  $e_0 = \bar{x} - x_0$ :

$$r_0 = Ae_0$$

Vamos procurar eliminar do erro qualquer componente na direção inicial  $d_0$ . Para isso poderíamos em princípio subtrair do erro sua projeção ortogonal na direção  $d_0$  (para o que teríamos que avaliar  $\langle e_0, d_0 \rangle$ ). Isto não é viável uma vez que não conhecemos  $e_0$ . O problema é no entanto contornável se ao invés de utilizarmos o produto interno usual do  $R^n$ , optarmos pelo produto interno induzido pela matriz  $A$ , uma vez que  $\langle e_0, d_0 \rangle_A = \langle Ae_0, d_0 \rangle = \langle r_0, d_0 \rangle$ , que podemos calcular mesmo desconhecendo  $e_0$ . Assim obteremos um novo erro  $e_1$ , que será ortogonal à direção inicial  $d_0$ , dado por

$$e_1 = e_0 - \frac{\langle r_0, d_0 \rangle}{\langle d_0, d_0 \rangle_A} d_0$$

Note que a equação anterior equivale a definirmos uma nova aproximação

$$x_1 = x_0 + \frac{\langle r_0, d_0 \rangle}{\langle d_0, d_0 \rangle_A} d_0$$

que podemos calcular apenas através da matriz  $A$  e da aproximação inicial. A nova aproximação  $x_1$  define um novo resíduo  $r_1 = b - Ax_1$ , valendo também que  $r_1 = Ae_1$ . Este novo resíduo forma com a direção inicial  $d_0$  uma base de um plano (espaço de dimensão 2). Vamos agora querer tornar o novo erro  $e_1$  ortogonal a este plano. Para tal vamos primeiramente tornar a base do plano ortogonal (em relação ao produto interno induzido pela matriz  $A$ ), definindo a direção  $d_1$  ortogonal a  $d_0$ , como:

$$d_1 = r_1 - \frac{\langle r_1, d_0 \rangle_A}{\langle d_0, d_0 \rangle_A} d_0$$

Para tornarmos o erro  $e_1$  ortogonal ao plano temos apenas que retirar sua projeção ortogonal na direção  $d_1$  (uma vez que ele já é ortogonal a  $d_0$ ). Assim obteremos:

$$e_2 = e_1 - \frac{\langle r_1, d_1 \rangle}{\langle d_1, d_1 \rangle_A} d_1$$

o que equivale a definirmos a nova aproximação

$$x_2 = x_1 + \frac{\langle r_1, d_1 \rangle}{\langle d_1, d_1 \rangle_A} d_1$$

Esta nova aproximação definirá um novo resíduo que formará junto com as direções anteriores um espaço de dimensão 3. Vamos tornar a base deste espaço ortogonal, retirando do novo resíduo sua projeção ortogonal nas direções anteriores. Aí só teremos que projetar o erro nesta nova direção, somando o resultado à aproximação anterior, obtendo assim um erro ortogonal a este espaço de dimensão 3. O processo continua com a adição de novas direções definidas através dos resíduos (ortogonalizados em relação às direções anteriores), sempre tornando o erro ortogonal a esses espaços de dimensão crescente. O processo terminará no máximo após  $n$  etapas, quando teremos o erro ortogonal a um sub-espaço de dimensão  $n$  do próprio  $R^n$  (ou seja, o sub-espaço será o próprio

$R^n$ ). Isto só é possível com erro igual a zero, ou seja, se chegarmos à solução exata da equação. Note que todo o processo só se torna possível ao utilizarmos o produto interno induzido pela matriz  $A$ , que nos permite calcular a projeção do erro em diversas direções, mesmo sem conhecer este erro. O algoritmo completo fica então definido como:

- Escolha  $x_0$  e calcule  $d_0 = r_0 = b - Ax_0$ . Defina  $k = 0$ .
- Enquanto  $k < n$  e  $r_k \neq 0$  faça
  - Defina  $\alpha_k = \langle r_k, d_k \rangle / \langle d_k, d_k \rangle_A$  ( $= \langle e_k, d_k \rangle_A / \langle d_k, d_k \rangle_A$ )
  - Calcule  $x_{k+1} = x_k + \alpha_k d_k$
  - Calcule  $r_{k+1} = b - Ax_{k+1} = r_k - \alpha_k A d_k$  ( $A d_k$  já foi calculado)
  - Defina  $\beta_k = \langle r_{k+1}, d_k \rangle_A / \langle d_k, d_k \rangle_A$
  - Defina  $d_{k+1} = r_{k+1} - \beta_k d_k$
  - Incremente  $k = k + 1$
- A solução é dada por  $x_k$ !

Se você é observador e entendeu a idéia do processo estará se perguntando porque na ortogonalização da base retiramos de  $r_{k+1}$  apenas sua projeção na direção  $d_k$ . Não está faltando retirarmos as projeções de  $r_{k+1}$  nas direções anteriores ( $d_0$  a  $d_{k-1}$ )? Não seria este o processo de Gram-Schmidt? Sim, isto é verdade, mas acontece que o novo resíduo já resulta automaticamente ortogonal a estas direções anteriores, não sendo portanto necessário retirar suas projeções nestas direções, aumentando consideravelmente a eficiência do esquema. Tente provar este fato (dá uma indução complicada ...), ou ao menos verifique-o computacionalmente ao implementar o algoritmo.

## Matrizes esparsas

Um ingrediente importante do Método dos Gradientes Conjugados é o produto da matriz  $A$  por vetores, como podemos observar no algoritmo acima. Se  $A$  for tal que a porcentagem de seus elementos nulos é alta e soubermos tirar proveito desta estrutura, então produtos dela por vetores podem ser calculados eficientemente. Matrizes deste tipo são as mundialmente famosas matrizes *esparsas* e são em grande parte responsáveis pela popularidade do Método dos Gradientes Conjugados e seus descendentes.

Sistemas lineares (incluindo sistemas sobredeterminados) cujas matrizes são esparsas (e grandes) aparecem em várias aplicações tais como resolução numérica de equações diferenciais parciais, programação matemática, engenharia civil, engenharia química, redes e circuitos elétricos, métodos numéricos em estatística, etc. Exemplos podem ser encomendados no estabelecimento MatrixMarket <sup>1</sup>, atendendo a gostos variados.

<sup>1</sup><http://math.nist.gov/MatrixMarket>

Para nos beneficiarmos da grande quantidade de elementos nulos, devemos usar alguma estrutura de dados especial. O objetivo principal é armazenar apenas os elementos não nulos de tal forma que possamos realizar operações comuns como produtos com vetores. Uma estrutura de dados bem simples usa três vetores: um vetor real  $VA$  contendo os elementos não nulos da matriz  $A$ , um vetor inteiro  $IL$  e um outro vetor inteiro  $IC$  contendo os respectivos índices de linha e coluna dos elementos de  $VA$ . Os três vetores tem tamanho  $NZ$ , o número de elementos não nulos da matriz.

Por exemplo, se a matriz  $A$   $n \times m$  for dada por ( $n = 6, m = 5$ )

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 0. & 3. & 4. & 0. & 5. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \\ 0. & 13. & 14. & 0. & 0. \end{pmatrix}$$

teremos

$VA$	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
$IL$	1	1	2	2	2	3	3	3	3	4	4	5	6	6
$IC$	1	4	2	3	5	1	3	4	5	3	4	5	2	3

A estrutura acima sugere uma forma mais compacta para o armazenamento. Usamos os mesmos vetores  $VA$  e  $IC$  mas trocamos o vetor de índices das linhas por um vetor de ponteiros inteiros  $IA$  indicando em que posição cada linha começa. Para este exemplo obtemos:

$VA$	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.
$IC$	1	4	2	3	5	1	3	4	5	3	4	5	2	3
$IA$	1	3	6	10	12	13								

Se acrescentarmos um elemento a  $IA$  na posição  $n + 1$  igual a  $IA(1) + NZ$ , obtemos o número de elementos na linha  $i$ ,  $1 \leq i \leq m$ , igual a  $IA(i+1) - IA(i)$ , o que é conveniente para contas. Neste exemplo acrescentaríamos  $IA(7) = 15$ .

Especificamente, para uma matriz  $A$   $n \times m$  com  $NZ$  elementos diferentes de zero temos:

- Um vetor real  $VA$  de tamanho  $NZ$  contendo os elementos não nulos de  $A$  armazenados linha por linha.
- Um vetor inteiro  $IC$  de tamanho  $NZ$  contendo os índices das colunas dos elementos de  $A$  armazenados em  $VA$ .
- Um vetor inteiro  $IA$  de tamanho  $n + 1$  contendo os ponteiros para o início de cada linha nos vetores  $VA$  e  $IC$ . Logo,  $IA(i)$  é a *posição* nos vetores  $VA$  e  $IC$  onde a linha  $i$  começa,  $1 \leq i \leq n$ . O elemento  $IA(n + 1)$  armazena o valor  $IA(1) + NZ$ , o endereço onde começaria a linha fictícia  $n + 1$ .

Este formato de armazenamento recebe o nome *Compressed Sparse Row* em inglês e é designado pela sigla CSR. Ele será usado por você na implementação do Método dos Gradientes Conjugados.

### Tarefa 1

Escreva um programa em C tal que dada uma matriz esparsa  $A$   $n \times n$  simétrica positiva definida e  $b \in R^n$ , resolva o sistema linear  $Ax = b$  pelo Método dos Gradientes Conjugados. A matriz deve ser armazenada no formato CSR e o produto dela por vetores deve ser implementado usando este formato. A cada passo do método você deve imprimir o resíduo  $r_k$ , a nova direção  $d_k$  e a aproximação  $x_k$ . Inicie a solução a partir de  $x_0 = 0$ . Teste o programa com os dados

$$A = \begin{pmatrix} 94 & 0 & 0 & 0 & 0 & 28 & 0 & 0 & 32 & 0 \\ 0 & 59 & 13 & 5 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 13 & 72 & 34 & 2 & 0 & 0 & 0 & 0 & 65 \\ 0 & 5 & 34 & 114 & 0 & 0 & 0 & 0 & 0 & 55 \\ 0 & 0 & 2 & 0 & 70 & 0 & 28 & 32 & 12 & 0 \\ 28 & 0 & 0 & 0 & 0 & 87 & 20 & 0 & 33 & 0 \\ 0 & 0 & 0 & 0 & 28 & 20 & 71 & 39 & 0 & 0 \\ 0 & 10 & 0 & 0 & 32 & 0 & 39 & 46 & 8 & 0 \\ 32 & 0 & 0 & 0 & 12 & 33 & 0 & 8 & 82 & 11 \\ 0 & 0 & 65 & 55 & 0 & 0 & 0 & 0 & 11 & 100 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

### Tarefa 2

Consideremos um sistema linear do tipo  $Cx = d$ , onde  $C$  é uma matriz  $n \times m$ , com  $n > m$  e  $d \in R^n$  (ou seja, temos um sistema linear com mais equações que incógnitas). Um tal sistema normalmente não tem solução. O produto  $Cx$  define um vetor em  $R^n$ , que é combinação linear das  $m$  colunas da matriz  $C$ . Como  $n > m$ , estas  $m$  colunas ( $m$  vetores) não podem gerar todo vetor  $d \in R^n$ . A solução aproximada que podemos procurar é o vetor  $x \in R^m$  tal que  $y = Cx$  seja o vetor de  $R^n$  (no espaço gerado pelas colunas de  $C$ ) mais próximo de  $d$ , segundo a distância usual entre dois vetores em  $R^n$  (dada por  $\|y - d\| = (\langle y - d, y - d \rangle)^{1/2}$ ). Conforme visto no curso este  $x$  é dado pela solução do sistema linear:

$$C^t C x = C^t d$$

obtido multiplicando-se o sistema original por  $C^t$  (matriz transposta de  $C$ ). A matriz  $A = C^t C$  é uma matriz simétrica positiva definida, caso as colunas de  $C$  sejam linearmente independentes. Podemos então resolver o sistema linear  $Ax = b$  (com  $A = C^t C$  e  $b = C^t d$ ), pelo método de gradientes conjugados.

Escreva um programa em C tal que dada uma matriz  $C$   $n \times m$ , com  $n > m$  e  $d \in R^n$ , resolva o sistema linear acima pelo método dos gradientes conjugados. Para usar o algoritmo, a matriz  $C$  tem de estar no formato CSR. Não

contrua a matriz  $A$  explicitamente, pois isso é desnecessário e muda o padrão de esparsidade. O produto  $w = Av$  pode ser calculado como  $z = Cv$ ,  $w = C^t z$ .

O método de gradientes conjugados pode ser utilizado como um esquema iterativo, gerando uma nova iteração a cada passo. Necessitaríamos  $n - 1$  passos para chegar à solução exata, mas podemos obter boas aproximações para a solução em bem menos iterações.

Teste seu programa com a matriz  $C$   $200 \times 100$  dada no arquivo de entrada *tarefa2.txt* e com vetor  $d$   $200 \times 1$  com todas as componentes iguais a 1. Atenção: a matriz não está no formato CSR no arquivo de entrada. A estrutura deste arquivo é a seguinte: na primeira linha temos três valores inteiros para  $n$ ,  $m$  e  $NZ$ ; em cada linha seguinte, três valores para o elemento da matriz e seus respectivos índices de linha e coluna. Sim, é o formato simples descrito inicialmente. Você terá de convertê-lo para o formato CSR.

A cada iteração do método de gradientes conjugados você deve imprimir a norma do resíduo  $r_k$ , e parar o processo assim que esta norma seja menor que  $\epsilon = 10^{-8}$ . Comece da aproximação inicial nula. Trabalhe com variáveis double! Imprima o valor obtido da solução  $x_1, x_2, \dots, x_{100}$ . Obs: Note que você não precisa armazenar todas as direções  $d_k$  calculadas durante o método de gradientes conjugados, uma vez que no cálculo de  $d_{k+1}$  apenas  $d_k$  é utilizada. Ter rotinas para o cálculo de produtos escalares de vetores e de produto de matrizes (esparsas) por vetores é interessante em ambas tarefas.