MAC 0110 – Introdução à Computação para Matemática Aplicada e Computacional

IME – Primeiro Semestre de 2004

http://panda.ime.usp.br

Terceiro Exercício Programa — Entrega: **7 de julho de 2004**

A fatoração QR e o cálculo da matriz inversa

1 Introdução

Em uma aula desse curso eu descrevi o método de eliminação gaussiana para resolver sistemas de equações lineares $Ax = b$, onde A é uma matriz $n \times n$ e x e b são vetores n-dimensionais. Eu disse que esse método é equivalente a fatorar a matriz A em um produto LU, onde L é uma matriz triangular inferior e U é uma matriz triangular superior. Falei ainda que a partir dai poderíamos calcular o determinante de A e a sua inversa $A^{-1} = U^{-1}L^{-1}$.

Na mesma aula eu disse que, em geral, é uma má idéia calcular a inversa de uma matriz e que quase sempre há métodos mais eficientes para resolver os problemas nos quais os leigos apelariam para a inversão de matrizes. Porém, há casos em que é interessante calcular a inversa da matriz A e nesse EP você aprenderá um método razoável para fazer isso. Esse método usa a fatoração

$$A = QR.$$

Esta abordagem parece com o $A = LU$ da aula e gera matrizes Q e R para as quais é muito fácil calcular as inversa Q^{-1} e resolver o sistema $Rx = b$.

Na próxima seção eu falarei sobre a matemática envolvida na fatoração QR, direi como são as matrizes Q e R e como elas podem ser usadas para calcular a inversa da matriz A. Baseado nessa matemática, na seção "Resumo do programa" eu esboçarei um programa para calcular matrizes inversas e, finalmente, na seção "O EP" eu listarei as funções que você deverá escrever para implementar e testar o método de inversão de matrizes aqui proposto.

O objetivo desse EP é que você aprenda a lidar com matrizes e vetores em `C`. Matrizes e vetores são o beabá da matemática aplicada e quanto antes você aprender a manipular esses conceitos computacionalmente melhor.

2 A Matemática

Nesta seção eu explico a versão "de Givens" da fatoração QR. Há outra versão, chamada "de Householder", que você pode encontrar na internet. No final da seção eu mostro como usar a fatoração $A = QR$ para calcular a inversa da matriz A.

O papel desta seção é explicar a matemática, o seu papel é entender o que está escrito aqui e transformar isso em funções em `C`.

A versão de Givens da fatoração QR é parecida com a fatoração LU: a cada passo combinamos duas linhas de modo a eliminar o elemento não zero mais à esquerda da linha de baixo. Por isso,

vale a pena fazer uma revisão da fatoração LU. Conforme vimos em classe, a operação básica da fatoração LU é zerar o elemento a_{ki} da linha l_k através da combinação de linhas

$$l_k \leftarrow l_k - \frac{a_{ki}}{a_{ii}} l_i$$

Por exemplo, se

$$A = \begin{pmatrix} 1 & 1 & 3 \\ 2 & 3 & 5 \\ 3 & 7 & 4 \end{pmatrix}$$

então durante a eliminação gaussiana fazemos as seguintes operações e geramos as seguintes matrizes (os índices começam de 0, pois esse é um curso de C). Começamos com

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad U = A = \begin{pmatrix} 1 & 1 & 3 \\ 2 & 3 & 5 \\ 3 & 7 & 4 \end{pmatrix}$$

e atualizamos L e U como nos passos abaixo

$$\text{movimento em } A: \ l_1 \leftarrow l_1 - 2l_0, \quad L \text{ resultante } = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad U \text{ resultante } = \begin{pmatrix} 1 & 1 & 3 \\ 0 & 1 & -1 \\ 3 & 7 & 4 \end{pmatrix},$$

$$\text{movimento em } A: l_2 \leftarrow l_2 - 3l_0, \quad L \text{ resultante } = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}, \quad U \text{ resultante } = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ 0 & 4 & -5 \end{pmatrix},$$

$$\text{movimento em } A: l_2 \leftarrow l_2 - 4l_1: \quad L \text{ resultante } = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{pmatrix}, \quad U \text{ resultante } = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ 0 & 0 & -1 \end{pmatrix}.$$

(Note que as operações em L são DIFERENTES das operações em A, veja abaixo)

O processo funciona assim:

1. A cada passo, determinamos qual é o coeficiente apropriado para combinar linhas de U de modo a zerar um elemento escolhido. Por exemplo, no primeiro passo esse coeficiente é 2 e a combinação de linhas $l_1 \leftarrow l_1 - 2l_0$ anula u_{10}. Combinamos então as linhas de U.

2. Para mantermos a igualdade $A = LU$ fazemos uma operação análoga nas *colunas* de L. Ao fazermos a operação

   $$l_k \leftarrow l_k - \frac{a_{ki}}{a_{ii}} l_i$$

   na linha l_k de U devemos, para compensar, fazer a operação

   $$c_i \leftarrow c_i + \frac{a_{ki}}{a_{ii}} c_k$$

   na coluna c_i de L. Note que o sinal de $+$ na equação acima é o oposto do $-$ na atualização de A.

3. Repetimos esses passos até que a matriz U seja triangular superior.

Como no caminho mantemos $A = LU$ e não estragamos a parte superior de L (verifique!!), ao fim do processo teremos a fatoração LU desejada.

A fatoração QR de Givens é parecida. Ela gera uma matriz *ortogonal* Q e uma matriz triangular superior R, muito parecida com a matriz U da fatoração LU. Dizer que a matriz Q é ortogonal significa que a sua inversa pode ser calculada muito facilmente: basta trocar os elementos q_{ij} e q_{ji}. Em outras palavras, a inversa de Q é a sua transposta.

A fatoração QR é feita assim:

1. Começamos com Q igual à identidade e R igual à matriz A.

2. A cada passo, para zerar o elemento r_{ki} de R, calculamos coeficientes c e s usando as expressões

$$\begin{aligned} c &= \frac{r_{ii}}{\sqrt{r_{ii}^2 + r_{ki}^2}} \\ s &= \frac{r_{ki}}{\sqrt{r_{ii}^2 + r_{ki}^2}} \end{aligned}$$

3. Combinamos então as linhas l_i e l_k de R fazendo

$$\begin{aligned} t &\leftarrow l_i && (1) \\ l_i &\leftarrow cl_i + sl_k && (2) \\ l_k &\leftarrow cl_k - st. && (3) \end{aligned}$$

   Note que "salvamos" uma cópia de l_i na linha temporária t, pois desejamos usar o valor antigo de l_i na atribuição (3).

4. Para preservar a igualdade $A = QR$ fazemos uma operação análoga nas *colunas* de Q. Se aplicarmos (1)–(3) às linhas l_i e l_k de R devemos então, para compensar, fazer as operações

$$\begin{aligned} t &\leftarrow c_i \\ c_i &\leftarrow cc_i + sc_k \\ c_k &\leftarrow cc_k - st, \end{aligned}$$

   nas colunas c_i e c_k de Q.

5. Repetimos esses passos até que a matriz R seja triangular superior.

Como no caminho mantemos $A = QR$, zeramos a parte inferior de R e preservamos a ortogonalidade de Q (acredite!!), ao fim do processo teremos a fatoração QR desejada.

Para verificar se você entendeu essa explicação, tome uma calculadora e execute os passos acima para a matriz A do exemplo para a fatoração LU. Você pode arredondar as contas para umas poucas casas, mas faça cada um dos passos e veja o que acontece.

Bom, agora que já sabemos fazer a fatoração $A = QR$ poderíamos calcular a inversa de A usando as identidades $A^{-1} = R^{-1}Q^{-1} = R^{-1}Q^t$, onde Q^t indica a transposta de Q. A transposta de Q é fácil de calcular, basta trocar q_{ij} por q_{ji}. Teríamos portanto que calcular apenas a inversa de R, o que também não é difícil, pois R é triangular superior. Porém, podemos calcular o

produto $R^{-1}Q^t$ sem calcular a inversa de R explicitamente (o que ilustra minha afirmação de que, em geral, não é preciso calcular a inversa de uma matriz, mesmo quando você deseja usar essa inversa.)

O "truque" para calcular $R^{-1}Q^t$ sem calcular R^{-1} se baseia em dois fatos:

- O produto de matrizes pode ser feito coluna a coluna. Se M e B são matrizes compatíveis e B tem colunas $c_0, c_1, \ldots, c_{n-1}$ então o produto MB tem colunas $Mc_0, Mc_1, \ldots, Mc_{n-1}$.
- Se b é um vetor então calcular $x = M^{-1}b$ é equivalente a resolver o sistema $Mx = b$.

No nosso caso as colunas de Q^t são as linhas de Q. Portanto a coluna c_i de $R^{-1}Q^t$ pode ser obtida resolvendo o sistema $Rc_i = l_i$, onde l_i é a linha correspondente de Q.

Essa é a matemática que você precisa para calcular a matriz inversa $A^{-1} = R^{-1}Q^t$. Agora é só sentar, digerir o que está escrito ai acima, escrever o código em C, testar, testar, testar, entregar o EP e sair de férias.

3 Resumo do programa

Sua tarefa principal nesse EP é implementar o seguinte programa para calcular a matriz inversa de `A`, que será armazenada na matriz `AInv`:

```
                Algoritmo para inverter a matriz A

fatore A como Q R
para cada linha de Q
  monte o vetor b correspondente à esta linha
  tente resolver o sistema R x = b
  se o x resultante for muito grande ou infinito retorne ERRO
  caso contrário escreva x na coluna correspondente de AInv
retorne SUCESSO
```

4 O EP

Você deverá entregar um arquivo qr.c contendo as seguintes funções

```
int  ehIdentidade(double A[N][N], double tol)
void escrevaColuna(double A[N][N], double coluna[N], int jColuna)
int  escrevaMatrizNxN(double matriz[N][N], FILE* f)
void fatoracaoQR(double Q[N][N], double R[N][N], double A[N][N])
void leiaLinha(double linha[N], double A[N][N], int iLinha)
int  leiaMatrizNxN(double matriz[N][N], FILE* f)
int  main(int argc, char* argv[])
void matMult(double P[N][N], double A[N][N], double B[N][N])
int  matrizInversa(double AInv[N][N], double A[N][N], double tol)
void matrizTransposta(double AT[N][N], double A[N][N])
int  sistemaTriangularSuperior(double x[N], double R[N][N], double b[N], double tol)
```

A constante `N` deve valer 30 e ser especificada através de um `#define N` no começo do arquivo, juntamente com as constantes `ERRO`, `SUCESSO`, `SIM` e `NAO`. Em todas as funções você pode assumir que não há conflitos entre os argumentos. Por exemplo, não se preocupe com a possibilidade das matrizes `Q` e `A` na função `fatoracaoQR`serem iguais.

Aqui está o resumo do que cada função deve fazer e para o que ela serve:

- `ehIdentidade`: Verifica se a matriz `A` está próxima da matriz identidade. Esta função retorna `SIM` se a diferença entre `A[i][j]` e a entrada correspondente da matriz identidade, em módulo, for menor que `tol` para todo `i` e `j`. Caso contrário ela retorna `NAO`. Você deve usar esta função para testar se a função `matrizInversa` funciona corretamente (na teoria $AA^{-1} = A^{-1}A = I$) e para avaliar a qualidade da sua fatoração QR (na teoria $QQ^t = I$).

- `escrevaColuna`: Escreve o vetor `coluna`na coluna `jColuna` da matriz `A`. Usada no algoritmo principal.

- `escrevaMatrizNxN`: Escreve a matriz `matriz` no arquivo texto `f`. O arquivo terá `N` linhas e cada linha conterá `N` números `double` no formato `%14.8lf`.

- `fatoracaoQR`: Faz a fatoração `QR` de `A`, ou seja, encontra uma matriz ortogonal `Q` e uma matriz triangular superior `R` tais que `A = Q`R. Usada no algoritmo principal.

- `leiaLinha`: Copia a linha `iLinha` da matriz `A` no vetor `linha`. Será usada no algoritmo principal

- `leiaMatrizNxN`: Lê a matriz `matriz` do arquivo texto `f`. O arquivo terá `N` linhas e cada linha conterá `N` números `double` no formato `%14.8lf`.

- `main` : Esta é a função que eu usarei para testar o seu EP. Ela será chamada com três parâmetros, como no exemplo

  ```
  qr aInv.txt a.txt tol
  ```

  O primeiro parâmetro indica o arquivo onde será escrita a matriz inversa, o segundo parâmetro indica o arquivo onde está a matriz a inverter e o terceiro parâmetro é o maior valor permitido para as entradas de aInv. A função main deverá retornar 0 se a matriz for invertida com sucesso e 3 caso contrário. As matrizes serão lidas e escritas de acordo com o formato definido na função `leiaMatrizNxN`.

- `matMult`: multiplica as matrizes `A` e `B` e atribui o produto AB à matriz `P`. Será usada para testar as demais funções (veja a descrição da função `ehIdentidade`).

- `matrizInversa`: Tenta calcular a matriz inversa de `A`. Se os elementos da matriz inversa `AInv` não forem maiores que a tolerância `tol` então o cálculo será efetuado, o resultado será atribuído à `AInv` e a função retornará `SUCESSO`. Caso contrário a função retornará `ERRO`. Esta função implementa o algoritmo principal.

- `matrizTransposta`: Calcula a transposta da matriz `A`, que é definida por `AT[i][j] = A[j][i]`. Esta função será usada juntamente com `ehIdentidade` e `matMult` para testar a fatoração QR, através da verificação da igualdade $QQ^t = I$.

- `sistemaTriangularSuperior`: Tenta resolver o sistema `Rx = b`, assumindo que a matriz `R` é triangular superior. Se os elementos do vetor `x` a ser calculado forem menores que a tolerância `tol`, em módulo, então a função calcula `x` e retorna `SUCESSO`. Caso contrário ela retorna `ERRO`. Esta função é usada no algoritmo principal.

Bom trabalho!