



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Métodos Numéricos

Trabajo práctico 3

Hay que poner un poquito más de esfuerzo...

Resumen

Integrante	LU	Correo electrónico
Danós, Alejandro	381/10	adp007@gmail.com
Gandini, Luciano	207/10	gl.gandini@gmail.com
Russo, Christian Sebastián	679/10	christian.russo@gmail.com

Palabras claves:

Índice

1. Introduccion Teorica	3
1.1. Factorizacion QR	3
1.1.1. Given	3
1.1.2. Householder	3
2. Desarrollo	4
2.1. Archivo de entrada	4
2.1.1. Explicacion	4
2.2. Archivo de salida	4
2.2.1. Explicacion	4
2.3. Factorizacion QR	4
2.3.1. Explicacion	4
2.3.2. Pseudocodigo	4
2.4. Método Uno: Usando Cuadrados Mínimos	5
2.4.1. Cuadrados Mínimos: General con Funciones Normales	5
2.4.2. Cuadrados Mínimos: Nuestro trabajo QR????	6
3. Experimentacion	7
4. Resultados	7
5. Apendice	8
5.1. Metodo de compilacion	8
5.2. Equipo de pruebas	8
5.3. Referencias bibliográficas	8

1. Introduccion Teorica

1.1. Factorizacion QR

Definicion: Se dice que una matriz tiene **factorizacion QR** si puede ser expresada de la forma

$$A = Q^*R$$

El algoritmo para llevar a una matriz a su forma QR tiene costo $O(n^3)$. Tiene la misma ventaja que la factorizacion LU de permitir resolver un sistema de ecuaciones en orden $O(n^2)$, pero con la ventaja que **toda matriz tiene factorizacion QR**

$$Ax = b$$

$$QRx = b$$

$$Q^t Q R x = Q^t b$$

$$R x = Q^t b$$

con Rx un **sistema triangular superior**

Para poder calcular la matriz R se pueden aplicar los metodos de **Givens o Householder**

1.1.1. Given

Para eliminar el elemento en la posicion (i,j) aplicamos la siguiente matriz:

$$G(i, j, \theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

Figura 1: Matriz de Givens

con $c = \cos(\theta)$ y $s = \sin(\theta)$. Luego aplicando $G(i,j,\theta) * A$ queda en 0 el elemento (i,j). Entonces aplicamos sucesivamente este procedimiento para todos los elementos que queremos poner en 0 obteniendo asi nuestra matriz R. Luego $Q^t = \prod_{i=n}^1 G_i$

1.1.2. Householder

Con este metodo vamos eliminando los 0 de abajo de la diagonal columna a columna. Sea $x = col_i(A)$, $y = (\|x\|_2, 0, \dots, 0)$ y sea $u = x - y$. Definimos $H_i = I - \frac{2uu^t}{u^t u}$. Luego aplicando $H_i * A$ queda triangulada la columna i de A. Aplicamos este procedimiento iterativamente sobre $A^{(i)}$ hasta dejar triangulada la matriz. Quedando $Q^t = \prod_{i=n}^1 H_i$

2. Desarrollo

En esta sección describiremos los métodos usados para resolver el problema, cada uno con sus ventajas y desventajas.

2.1. Archivo de entrada

2.1.1. Explicacion

2.2. Archivo de salida

2.2.1. Explicacion

2.3. Factorizacion QR

2.3.1. Explicacion

2.3.2. Pseudocodigo

Algorithm 1 FactorizacionQR(Matrix A, Matrix Qt, Matrix R)

```

R = A
Matrix square
Qt = square(A.n, A.n)
Q = identidad();
for  $i = 0$  hasta  $A.m$  do
    Matrix tmp(Qt.n,Qt.m) = identidad()
    Matrix subR(R.n - i, R.m - i)
    Matrix subQt(Qt.n - i, Qt.m - i) = identidad()
    if subR.n > 1 then
        generarSubMatrix(subR, R, i)
        triangularColumna(subR, subQt)
        agregarSubMatrix(subR, R, i)
        agregarSubMatrix(subQt, tmp, i)
    end if
    Qt = tmp*Qt
end for
```

Algorithm 2 generarSubMatrix(Matrix sub, Matrix A, int i)

```

w = 0
for  $j = i$  hasta  $A.n$  do
    p = 0
    for  $k = i$  hasta  $A.m$  do
         $sub_{w,p} = A_{j,k}$ 
        p = p + 1
    end for
    w = w + 1
end for
```

Algorithm 3 triangularColumna(Matrix sub, Matrix subQt)

```

Matrix x(sub.n,1)
Matrix y(sub.n,1)
Matrix u(sub.n,1)
for  $i = 0$  hasta  $x.n$  do
     $x_{i,0} = sub_{i,0}$ 
end for
 $y_{0,0} = x.normVector()$ 
 $u = x - y$ 
 $normaU = u.normVector()^2$ 
 $uTranspuesto = u.transpuesta()$ 
Matrix aux(u.n,u.m) =  $uTranspuesto * sub$ 
Matrix aux2(sub.n,sub.n) =  $u * aux$ 
 $coeficiente = 2/normaU$ 
 $sub = sub - (aux2 * coeficiente)$ 
 $aux = uTranspuesto * subQt$ 
 $aux2 = u * aux$ 
 $subQt = subQt - (aux2 * coeficiente)$ 

```

Algorithm 4 agregarSubMatrix(Matrix sub, Matrix A, int i)

```

 $w = 0$ 
for  $j = i$  hasta  $A.n$  do
     $p = 0$ 
    for  $k = i$  hasta  $A.m$  do
         $A_{w,p} = sub_{j,k}$ 
         $p = p + 1$ 
    end for
     $w = w + 1$ 
end for

```

2.4. Método Uno: Usando Cuadrados Mínimos

Nuestro primer enfoque fue mirar al problema como si fuera uno de analizar los datos obtenidos en un experimento y tratásemos de describir la distribución de estos mediante una función.

En esta perspectiva, nuestra entrada sería el tiempo y la salida la posición en la cancha de la pelota. Además, como las variaciones en las coordenadas x e y de la pelota son independientes podemos dividir al problema en una entrada y dos salidas. De esta forma, deberíamos resolver dos problemas de cuadrados mínimos.

2.4.1. Cuadrados Mínimos: General con Funciones Normales

La siguiente información sobre Cuadrados Mínimos fue sacada de [1, p 501]:

El problema general de aproximar un conjunto de datos, $\{(x_i, y_i) | i = 1, 2, \dots, m\}$ con un polinomio

$$P_n = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0,$$

de grado $n < m - 1$, se reduce a elegir las constantes a_0, a_1, \dots, a_n que minimicen el cuadrado mínimo $E = E_2(a_0, a_1, \dots, a_n)$ donde

$$\begin{aligned}
E &= \sum_{i=1}^m (y_i - Pn(x_i))^2 \\
&= \dots^1 = \\
&= \sum_{i=1}^m y_i^2 - 2 \sum_{j=0}^n a_j \left(\sum_{i=1}^m y_i x_i^j \right) + \sum_{j=0}^n \sum_{k=0}^n a_j a_k \left(x_i^{j+k} \right)
\end{aligned}$$

Para encontrar los mínimos, dado que es una función convexa (falta demostrar), debemos buscar los puntos críticos y éstos serán los mínimos globales. Para ello, diferenciaremos la función por sus constantes a_i y las igualaremos a 0:

$$0 = \frac{\partial E}{\partial a_j}$$

Lo que daría, después de algunos pasos¹, $n + 1$ ecuaciones del siguiente estilo, con $0 \leq k \leq n$:

$$a_0 \sum i = 1mx_i^k + a_1 \sum i = 1mx_i^{k+1} + a_2 \sum i = 1mx_i^{k+2} + \dots + a_n \sum i = 1mx_i^{k+n} = \sum i = 1my_i x_i^k$$

Estas ecuaciones terminarían formando las filas de nuestra matriz extendida y el sistema lineal² resultante tiene solución única si los x_i distintos.

2.4.2. Cuadrados Mínimos: Nuestro trabajo QR????

Adaptado a nuestro trabajo, tendríamos dos sistemas lineales de ecuaciones de $n + 1$ filas y $n + 1$ columnas representando las funciones normales con entrada t tiempo y salida coordenada x ; y entrada t tiempo y salida coordenada y , respectivamente.

Es decir, para poder describir el comportamiento de ambas variables con un polinomio de grado como máximo n , necesitaríamos $n+1$ tiempos.

¹Para el desarrollo completo referirse al libro citado

²Es lineal dado que las incógnitas están relacionadas entre ellas de forma lineal

3. Experimentacion

4. Resultados

5. Apendice

5.1. Metodo de compilacion

5.2. Equipo de pruebas

5.3. Referencias bibliográficas

Referencias

- [1] Richard L. Burden and J. Douglas Faires *Numerical Analysis*. 2005.