Métodos Numéricos

Trabajo práctico 3

Hay que poner un poquito más de *e*sfuerzo...

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

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**

$$Ax = b$$

$$QR\ x = b$$

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

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

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(θ) y s = sen(θ). Luego aplicando G(i,j,θ) * 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 e limando los 0 de abajo de la diagonal columna a columna. Sea x = $col_i(A)$, y = ($||x||_2$, 0 ,. . . 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

El estudio de Cuadrados Mínimos nació al querer describir el comportamiento de datos con funciones polinómicas. Normalmente, las mediciones traen inherentemente una cuota de ruido y si se sospecha que éstas siguen un crecimiento de un polinomio de grado como máximo n, es díficil encontrar los coeficientes de este polinomio dado que el ruido afecta a los puntos. Cuadrados Mínimos trata de solucionar este problema.

Más formalmente, si se tiene m entradas y para cada una de ellas una salida asociada, x_i e y_i respectivamente, y se los quiere describir con un polinomio $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ de grado máximo fijo n, entonces la técnica de Cuadrados Mínimos busca a los $n+1$ coeficientes

a_i $\forall i = 0 \cdots n$ resolviendo el problema buscar el vector a tal que minimice a la norma de $A \times a - b$ al cuadrado, con $A \in \mathbb{R}^{m\times(n+1)}$, $a \in \mathbb{R}^m$ y $b \in \mathbb{R}^n$ los siguientes:

$$A = \begin{pmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_m^n & x_m^{n-1} & \cdots & x_m^1 & 1 \end{pmatrix},\ a = \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} \text{ y } b = \begin{pmatrix} y_m \\ y_{m-1} \\ \vdots \\ y_1 \\ y_0 \end{pmatrix}$$

La diferencia entre resolver directamente el sistema $A \times a = b$ y minimizar a $\|A \times a - b\|$ consta en que el primero busca a los coeficientes tal que el polinimio pasa exactamente por los puntos y_i, es decir, $P(x_i) = y_i$ $\forall i = 0..m$, mientras que el segundo trata de buscar los coeficientes que minimicen a $\sum i = 0m(P(x_i) - y_i)^2$, o la suma de los errores.

2.4.2. Cuadrados Mínimos: Específico a nuestro trabajo

En nuestro caso, deberíamos resolver dos problemas de Cuadrados Mínimos dado que para cada tiempo t_i tenemos dos coordenadas independientes: x_i e y_i. Si seguimos la notación anterior, la matriz A no cambiaría entre una coordenada y otra aunque sí el vector b sí tendría dos casos a parte, que llamaremos b_x y b_y.

2.5. Demostraciones

En esta sección daremos demostraciones de los supuestos considerados en los algoritmos usados en el trabajo.

Sean $A \in \mathbb{R}^{m\times(n+1)}$ con:

$$A = \begin{pmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_m^n & x_m^{n-1} & \cdots & x_m^1 & 1 \end{pmatrix},\ a = \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix} \text{ y } b = \begin{pmatrix} y_m \\ y_{m-1} \\ \vdots \\ y_1 \\ y_0 \end{pmatrix}$$

LemaSi m>n+1, entonces A tiene rango de columnas máximo.

Prueba: $A = (C_1, C_2, \ldots, C_n, C_{n+1})$ si la miramos como columnas. Asumamos que no tiene rango máximo. Eso es equivalente a que:

$\alpha_1 C_1 + \alpha_2 C_2 + \ldots + \alpha_n C_n + \alpha_{n+1} C_{n+1} = 0$ con $\alpha_i \in R$ y $\alpha_i \neq 0$ para algún i, que es equivalente a que $A * \alpha = 0$. O sea, que el polinomio $P(x)$ de grado n tendría m>n+1 raíces dado que cada fila sería una evaluación en un punto distinta del polinomio dado que los x_i son distintos. Absurdo.

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.