Métodos de Computación Científica

Trabajo práctico de programación #2

05-09-2012

Victoria Martínez de la Cruz - LU. 87620

1) Para calcular las distintas normas matriciales (p-normas, norma euclídea y norma de Frobenius) utilizamos el comando `norm(A,p)` de Octave.

```
octave:30> A = [2 1 6; -1 6 -4; 3 -4 2];
octave:31> A
A =

   2   1   6
  -1   6  -4
   3  -4   2

// p = 1
octave:32> l1 = norm(A,1);
octave:33> l1
l1 =  12

// p = 2
octave:34> l2 = norm(A,2);
octave:35> l2
l2 =  9.4937

// p = inf
octave:36> li = norm(A,Inf);
octave:37> li
li =  11

// p = Frobenius
octave:38> lf = norm(A,"fro");
octave:39> lf
lf =  11.091
```

Métodos de Computación Científica

Trabajo práctico de programación #2

05-09-2012

2) Podemos resolver este sistema de ecuaciones de dos maneras diferentes

Utilizando el operador \

```
octave:25> A = [-4 1 1 0; 1 -4 0 1; 1 0 -4 1; 0 1 1 -4];
octave:26> A
A =

  -4   1   1   0
   1  -4   0   1
   1   0  -4   1
   0   1   1  -4

octave:27> b = [-200 -400 0 -200]';
octave:28> b
b =

  -200
  -400
     0
  -200

octave:29> x = A\b
x =

   100.000
   150.000
    50.000
   100.000
```

O aplicando la fórmula $x = A^{-1} b$. En este caso usamos el comando built-in `inv(x)`.

```
octave:44> x = inv(A)*b;
octave:45> x
x =

  100.000
  150.000
   50.000
  100.000
```

Métodos de Computación Científica

Trabajo práctico de programación #2

05-09-2012

3) Dado que A es la matriz de Hilbert, podemos generarla utilizando el comando `hilb(n)` provisto por MatLab. Este comando genera la matriz de Hilbert de nxn, donde n es un valor definido por el usuario.
De la misma forma, MatLab provee un segundo comando `invhilb(n)` que retorna la matriz de Hilbert inversa de nxn. Almacenamos la matriz inversa en B.

```
octave:1> A = hilb(50)
A =

 Columns 1 through 15:

   1.000000   0.500000   0.333333   0.250000   0.200000     ...
   0.500000   0.333333   0.250000   0.200000   0.166667     ...
   0.333333   0.250000   0.200000   0.166667   0.142857     ...
   0.250000   0.200000   0.166667   0.142857   0.125000     ...
   0.200000   0.166667   0.142857   0.125000   0.111111     ...
   0.166667   0.142857   0.125000   0.111111   0.100000     ...
   0.142857   0.125000   0.111111   0.100000   0.090909     ...
   0.125000   0.111111   0.100000   0.090909   0.083333     ...
   0.111111   0.100000   0.090909   0.083333   0.076923     ...
   0.100000   0.090909   0.083333   0.076923   0.071429     ...
   0.090909   0.083333   0.076923   0.071429   0.066667     ...
   0.083333   0.076923   0.071429   0.066667   0.062500     ...

(...)

octave:2> B = invhilb(50)
B =

 Columns 1 through 12:

   2.5000e+03  -3.1238e+06   1.2995e+09  -2.6975e+11   3.3503e+13  ...
  -3.1238e+06   5.2042e+09  -2.4356e+12   5.3928e+14  -6.9770e+16  ...
   1.2995e+09  -2.4356e+12   1.2158e+15  -2.8043e+17   3.7317e+19  ...
  -2.6975e+11   5.3928e+14  -2.8043e+17   6.6528e+19  -9.0374e+21  ...
   3.3503e+13  -6.9770e+16   3.7317e+19  -9.0374e+21   1.2472e+24  ...
  -2.7640e+15   5.9205e+18  -3.2326e+21   7.9530e+23  -1.1112e+26  ...
   1.6215e+17  -3.5457e+20   1.9667e+23  -4.8990e+25   6.9143e+27  ...
  -7.0972e+18   1.5765e+22  -8.8537e+24   2.2277e+27  -3.1703e+29  ...

(...)
```

Métodos de Computación Científica

Trabajo práctico de programación #2

05-09-2012

```
octave:3> C = A*B
C =

 Columns 1 through 12:

  -4.9377e+22   1.2115e+26  -7.3392e+28   1.9797e+31  -2.9949e+33  ...
  -4.8345e+22   1.1843e+26  -7.1712e+28   1.9349e+31  -2.9335e+33  ...
  -4.6845e+22   1.1466e+26  -6.9467e+28   1.8720e+31  -2.8415e+33  ...
  -4.5848e+22   1.1245e+26  -6.8137e+28   1.8358e+31  -2.7811e+33  ...
  -4.4527e+22   1.0916e+26  -6.6143e+28   1.7861e+31  -2.6989e+33  ...
  -4.3651e+22   1.0701e+26  -6.4890e+28   1.7464e+31  -2.6476e+33  ...
  -4.2377e+22   1.0409e+26  -6.3204e+28   1.7002e+31  -2.5762e+33  ...
  -4.1643e+22   1.0206e+26  -6.1871e+28   1.6667e+31  -2.5239e+33  ...
  -4.0518e+22   9.9469e+25  -6.0261e+28   1.6284e+31  -2.4630e+33  ...

(...)
```

```
octave:4> E = sum(sum(abs(C)))
E =  3.6588e+60
octave:5> ea = E - 50
ea =  3.6588e+60
octave:6> er = (E - 50)/50
er =  7.3176e+58
```

Se optó por representar las matrices con todos sus decimales ya que no es posible representar la matriz de Hilbert inversa con n=50 con valores fraccionarios sin perder precisión. En tal caso se vería un * en vez del elemento deseado.

Podemos notar que con valores tan grandes la computadora no es capaz de calcular con la precisión requerida y se comete un error proporcional al resultado obtenido.