Actividad A_2.

Algebra Lineal, metodos directos

Ignacio Sica

25/03/21

El objetivo es trabajar en la resolución de sistemas de ecuaciones utilizando los métodos de descomposición PLU y Cholesky.

Contents

Puntos pre-actividad 2

Estudie la función SustitucionAdelante que aparece en el archivo con el mismo nombre para construir las funciones que se solicitan.

Construya una función que implemente la Sustitución hacia atrás que tenga por entradas una matriz y un vector y como salidas un vector con la solución y un mensaje de éxito o fracaso del algoritmo. Puede asumir que la matriz de entrada es triangular superior.

Construya una función que descomponga a una matriz cuadrada M en matrices PLU.

Construya una función que descomponga a una matriz simétrica positiva definida M usando el método de Cholesky.

Utilizando las funciones anteriores pertinentes construya una función que resuelva el sistema de ecuaciones Ax=b usando la descomposición PLU.

Utilizando las funciones anteriores pertinentes construya una función que resuelva el sistema de ecuaciones Ax=b usando la descomposición de Cholesky.

Utilizando las funciones anteriores pertinentes construya una función que calcule la matriz inversa de una matriz dada utilizando la descomposición PLU.

Utilizando las funciones anteriores pertinentes construya una función que calcule la matriz inversa de una matriz simétrica positiva definida dada utilizando la descomposición de Cholesky.

```
load datos
```

Ejercicio n#1

Utilizando los algoritmos desarrollados resuelva los sistemas de ecuaciones $a_i * y_i = b_i$ usando la descomposición PLU y para cada caso reporte el tiempo requerido por la computadora para resolver el problema, así como también calcule el error relativo con respecto a los vectores b $E_i = (a_i * y - b_i)/b_i$, el número de dígitos significativos que se obtienen y el número de condición de la matriz de coeficientes. Comente sus resultados.

```
disp("a1:")
ej1(a1, b1);

disp("a2:")
ej1(a2, b2);

disp("a3:")
ej1(a3, b3);

disp("a4:")
ej1(a4, b4);

disp("a5:")
ej1(a5, b5);
```

```
a1:
Elapsed time is 0.000793934 seconds.
Error: 4.60834e-15
Cifras significativas: 14
Numero de condición: 135.367

a2:
Elapsed time is 0.0316641 seconds.
Error: 8.27359e-15
Cifras significativas: 13
Numero de condición: 1069.78

a3:
Elapsed time is 0.181818 seconds.
Error: 1.3744e-14
Cifras significativas: 13
Numero de condición: 2734.99

a4:
Elapsed time is 0.431606 seconds.
Error: 3.51864e-14
Cifras significativas: 13
Numero de condición: 2565.64

a5:
Elapsed time is 6.59511 seconds.
Error: 9.99503e-14
Cifras significativas: 12
Numero de condición: 16714.7
```

Se puede observar que la cantidad de digitos significativos es de los resultados ronda entre los 14 en el caso de mayor precision en las matrices mas pequeñas y 12 en el caso de las matrices mas grandes. Tambien se puede notar como a medida que el tamaño de la matriz aumenta la cantidad de tiempo que cuesta computar las operaciones aumenta de manera considerable. Este tiempo varia desde la milesima de segundos hasta 6 segundo en el caso de la matriz a5

Ejercicio n#2

Utilizando los algoritmos desarrollados resuelva los sistemas de ecuaciones $mi * yi = bi$ usando la descomposición PLU y la descomposición de Cholesky y para cada caso reporte el tiempo requerido por la computadora para resolver el problema, así como también calcule el error relativo con respecto a los vectores b $E_i = (m_i * y - b_i)/b_i$, el número de dígitos significativos que se obtienen y el número de condición de la matriz de coeficientes. Comente sus resultados, compare el desempeño de los algoritmos ente sí.

```
disp("m1:")
ej2(m1, b1);

disp("m2:")
ej2(m2, b2);

disp("m3:")
ej2(m3, b3);

disp("m4:")
ej2(m4, b4);

disp("m5:")
ej2(m5, b5);
```

```
m1:
Numero de condición: 18324.2
Cholesky
```

```
Elapsed time is 0.000603199 seconds.
Error: 5.05433e-14
Cifras significativas: 12
PLU
Elapsed time is 0.000393867 seconds.
Error: 3.08209e-14
Cifras significativas: 13

m2:
Numero de condición: 1.14444e+06
Cholesky
Elapsed time is 0.037379 seconds.
Error: 8.58865e-12
Cifras significativas: 10
PLU
Elapsed time is 0.030694 seconds.
Error: 5.1589e-12
Cifras significativas: 10

m3:
Numero de condición: 7.48017e+06
Cholesky
Elapsed time is 0.224896 seconds.
Error: 8.61561e-12
Cifras significativas: 10
PLU
Elapsed time is 0.179577 seconds.
Error: 4.20243e-12
Cifras significativas: 11

m4:
Numero de condición: 6.58249e+06
Cholesky
Elapsed time is 0.484178 seconds.
Error: 2.16023e-11
Cifras significativas: 10
PLU
Elapsed time is 0.403557 seconds.
Error: 1.13747e-11
Cifras significativas: 10

m5:
Numero de condición: 2.79381e+08
Cholesky
Elapsed time is 5.70053 seconds.
Error: 1.70574e-10
Cifras significativas: 9
PLU
Elapsed time is 6.35617 seconds.
Error: 5.20213e-11
Cifras significativas: 9
```

Se puede observar que la cantidad de digitos significativos es de los resultados ronda entre los 13 en el caso de mayor precision en las matrices mas pequeñas y 9 en el caso de las matrices mas grandes. Tambien se puede notar como a medida que el tamaño de la matriz aumenta la cantidad de tiempo que cuesta computar las operaciones aumenta de manera considerable. Este tiempo varia desde la milesima de segundos hasta 6 segundo en el caso de la matriz m5. Otra aspecto a mencionar es es que el numero de condicion de las matrices aumenta exponencialmente conforme al aumento del tamaño de las matrices. En el caso de la martiz mas pequeña este numero es 18324 y en el caso de la matriz mas grande es 2.79381e+08

Ejercicio n#3

Calcule la inversa de las matrices m_i utilizando la función que emplea la descomposición PLU y la función que emplea la descomposición de Cholesky. Reporte el tiempo de ejecución de las funciones, calcule el error relativo con respecto a la matriz identidad del producto $m_i m_i^{-1}$. Comente sus resultados.

```
disp("m1:")
ej3(m1);

disp("m2:")
ej3(m2);

disp("m3:")
ej3(m3);

disp("m4:")
ej3(m4);
```

```
m1:
Cholesky reciprocal matrix
Elapsed time is 0.000926018 seconds.
Cholesky error: 2.17659e-13
PLU reciprocal matrix
Elapsed time is 0.00101399 seconds.
PLU error: 2.48294e-13

m2:
Cholesky reciprocal matrix
Elapsed time is 0.191553 seconds.
Cholesky error: 1.08972e-10
PLU reciprocal matrix
Elapsed time is 0.175024 seconds.
PLU error: 1.03518e-11

m3:
Cholesky reciprocal matrix
Elapsed time is 1.20653 seconds.
Cholesky error: 4.69667e-10
PLU reciprocal matrix
Elapsed time is 1.17676 seconds.
PLU error: 4.70036e-11

m4:
Cholesky reciprocal matrix
Elapsed time is 2.65137 seconds.
Cholesky error: 2.23214e-10
PLU reciprocal matrix
Elapsed time is 2.53465 seconds.
PLU error: 4.84284e-11
```

Se puede observar que la cantidad de digitos significativos es de los resultados ronda entre los 13 en el caso de mayor precision en las matrices mas pequeñas y 10 en el caso de las matrices mas grandes. Tambien se puede notar como a medida que el tamaño de la matriz aumenta la cantidad de tiempo que cuesta computar las operaciones aumenta de manera considerable. Este tiempo varia desde la milesima de segundos hasta casi 3 segundo en el caso de la matriz m5. Tambien se puede destacar que en la mayoria de los casos el algoritmo de PLU demora menos tiempo que el Cholesky siendo esto lo opuesto a lo que uno esperaria ua que el de cholesky es una optiomizacion del primero. Esto se debe a que los mismo se conforman de distintas operaciones fundamentales; en el caso de Cholesky se utiliza la raiz cuadrada que es una operacion significativamente costosa

Bibliografia

Eric Walter, Springer, Numerical Methods and Optimization

Richard Khoury & Douglas Wilhelm Harder, Springer Numerical Methods and Modelling for Engineering

Javier Segura, Universidad de Cantabria, Introduccion al analisis numerico (https://personales.unican.es/segurajj/intro.pdf)

Errors for Linear Systems (http://terpconnect.umd.edu/~petersd/460/linsysterrn.pdf)

Inv Function (https://octave.sourceforge.io/octave/function/inv.html)

Descomposición Plu

```
function [P,L,U] = plu (M)
  n = size(M);
  P = eye(n);
  L = zeros(n);
  U = M;

  ColumnIndex = 1;
  while (ColumnIndex <= n - 1)
    ColumnVector = U(ColumnIndex:end, ColumnIndex);
    [max_value, IndexOfMaximum] = max(abs(ColumnVector));
    IndexOfMaximum = IndexOfMaximum + ColumnIndex - 1;

    P([ColumnIndex IndexOfMaximum],:) = P([IndexOfMaximum ColumnIndex],:);
    L([ColumnIndex IndexOfMaximum],:) = L([IndexOfMaximum ColumnIndex],:);
    U([ColumnIndex IndexOfMaximum],:) = U([IndexOfMaximum ColumnIndex],:);

    RowIndex = ColumnIndex + 1;
    while (RowIndex <= n)
```

```
      s = (-1 * U(RowIndex, ColumnIndex)) / U(ColumnIndex, ColumnIndex);
      U(RowIndex, :) = U(RowIndex, :) + U(ColumnIndex, :) * s;
      L(RowIndex, ColumnIndex) = s * -1;
      RowIndex = RowIndex + 1;
    endwhile
    ColumnIndex = ColumnIndex + 1;
  endwhile
  L = L + eye(n);
endfunction
```

Descomposición cholesky

```
function [L] = cholesky (M)
  L = zeros(size(M));
  Index = 1;
  while(Index <= size(M))
    L(Index, Index) = sqrt(M(Index, Index) - sum(L(1 : Index, Index).^2));
    BelowIndex = Index + 1;
    while(BelowIndex <= size(M))
      L(Index, BelowIndex) = (M(BelowIndex, Index) - sum(L(1 : Index, Index).*L(1 : Index, BelowIndex)))/L(Index, Index);
      BelowIndex = BelowIndex + 1;
    endwhile
    Index = Index + 1;
  endwhile
endfunction
```

Sistemas de ecuaciones x PLU

```
function [x] = sem_plu(A, b)
  [P, L, U] = plu(A);
  %[L, U, P] = lu(A);

  [y, msg] = SustitucionAdelante (L, P*b);
  [x, msg] = SustitucionAtras (U, y);
endfunction
```

Sistemas de ecuaciones x Cholesky

```
function [x] = sem_cholesky(A, b)
  [L] = cholesky(A);
  [y, msg] = SustitucionAdelante (L', b);
  [x, msg] = SustitucionAtras(L, y);
endfunction
```

Matriz inversa x PLU

```
function [Minv] = rm_plu(M)
  [n, m] = size(M);
  I = eye(n);
  [P,L,U] = plu(M);
  %[L, U, P] = lu(M);
  Minv = zeros(n);

  for(i = 1 : n)
    m_i = SustitucionAdelante(L, P*I(:, i));
    x_i = SustitucionAtras(U, m_i);
    Minv(:, i) = x_i;
  endfor
endfunction
```

Matriz inversa x Cholesky

```
function [Minv] = rm_cholesky(M)
  [n, m] = size(M);
  I = eye(n);
  [L] = cholesky(M);

  Minv = zeros(n);

  for(i = 1 : n)
    m_i = SustitucionAdelante(L', I(:, i));
    x_i = SustitucionAtras(L, m_i);
    Minv(i, :) = x_i;
  endfor
endfunction
```

SustitucionAdelante

```
function [y,msg] = SustitucionAdelante (L, b)
  [mL,nL] = size(L);
  [mb,nb] = size(b);

  ##
  if mL ~= nL
    msg = 0;
    y = inf;
    disp('L no es cuadrada')
    return; # fin de la funcion
  elseif (mL ~= mb) || nb ~= 1
    msg = 0;
    y = inf;
    disp('L y b son de dimensiones incompatibles')
    return; # fin de la funcion
  endif
  for ii=1:mL
    if abs(L(ii,ii)) < eps
      msg = 0;
      y = inf;
      disp('L tiene pivote menor que eps')
      return; # fin de la funcion
    endif
  endfor

  ## Inicio del algoritmo
  y = 0*b;

  if (L(1,1)==1)
    y(1) = b(1);
  else
    y(1) = b(1)/L(1,1);
  endif
  for ii=2:mL
    y(ii) = b(ii)-L(ii,1:ii-1)*y(1:ii-1);
    pivote = L(ii,ii);
    if (pivote ~=1)
      y(ii) = y(ii)/pivote;
    endif
  endfor

  msg = 1;
 % disp('Algoritmo finalizo normalmente');

  endfunction
```

SustitucionAtras

```
function [y,msg] = SustitucionAtras (L, b)
  [mL,nL] = size(L);
  [mb,nb] = size(b);

  if mL ~= nL
    msg = 0;
    y = inf;
    disp('L no es cuadrada')
    return; # fin de la funcion
  elseif (mL ~= mb) || nb ~= 1
    msg = 0;
    y = inf;
    disp('L y b son de dimensiones incompatibles')
    return; # fin de la funcion
  endif
  for ii=1:mL
    if abs(L(ii,ii)) < eps
      msg = 0;
      y = inf;
      disp('L tiene pivote menor que eps')
      return; # fin de la funcion
    endif
  endfor

  ## Inicio del algoritmo
  y = 0*b;

  if (L(end,end)==1)
    y(end) = b(end);
  else
    y(end) = b(end)/L(end,end);
  endif
  for ii=1:nL-1
```

```
    y(nL-ii) = b(nL-ii)-L(nL-ii,nL-ii:end)*y(nL-ii:end);
    pivote = L(nL-ii,nL-ii);
    if (pivote ~=1)
      y(nL-ii) = y(nL-ii)/pivote;
    endif
  endfor

  msg = 1;
 % disp('Algoritmo finalizo normalmente');

endfunction
```

Codigo ej1

```
function retval = ej1 (a, b)
  tic();
  y = sem_plu(a, b);
  toc ();
  error = norm(a * y - b) / norm(b);
  printf("Error: %d", error),
  disp ("")
  Sd=floor(-log10(error/0.5));
  printf("Cifras significativas: %d", Sd),
  disp (""),
  c=cond(a);
  printf("Numero de condición: %d", c),
  disp (""),disp ("")
endfunction
```

Codigo ej2

```
function retval = ej2 (m, b)
  c=cond(m);
  printf("Numero de condición: %d", c),
  disp ("")
  disp ("Cholesky")
  tic();
  y = sem_cholesky(m, b);
  toc ();
  error = norm(m * y - b) / norm(b);
  printf("Error: %d", error),
  disp ("")
  Sd=floor(-log10(error/0.5));
  printf("Cifras significativas: %d", Sd),
  disp ("")
  disp ("PLU")
  tic();
  y = sem_plu(m, b);
  toc ();
  error = norm(m * y - b) / norm(b);
  printf("Error: %d", error),
  disp ("")
  Sd=floor(-log10(error/0.5));
  printf("Cifras significativas: %d", Sd),
  disp (""),disp ("")
endfunction
```

Codigo ej3

```
function retval = ej3 (m)
  disp ("Cholesky reciprocal matrix")
  tic ();
  m_inv_chol = rm_cholesky(m);
  toc ()
  error = norm(m * m_inv_chol - eye(size(m))) / norm(eye(size(m)));
  printf("Cholesky error: %d", error),
  disp ("")

  disp ("PLU reciprocal matrix")
  tic ();
  m_inv_plu = rm_plu(m);
  toc ()
  error = norm(m * m_inv_plu  - eye(size(m))) / norm(eye(size(m)));
  printf("PLU error: %d", error),
  disp (""),disp ("")
endfunction
```
