Facultad de Ciencias

Tarea 3

Análisis Númerico

Oscar Andrés Rosas Hernandez

Alarcón Alvarez Aylin

Laurrabaquio Rodríguez Miguel Salvador

Pahua Castro Jesús Miguel Ángel

Noviembre 2018

Índice

1. Problemas de Computadora

Una nota importante es que al inicio de CADA script se incluyen los algoritmos, porfavor cambia la primera linea de cada script para que el path sea el correcto, porfavor.

Esta linea:

```
getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework2/Code/Algorithms')
```

Para ejecutar cada uno basta con hacer algo como:

```
exec("/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework2/Code/22a.sce", -1)
```

1.1. 23

Ejecuta los scripts que esta dentro de Code llamado: 23.sce

En este código muestra justo lo que se nos pide, por el método de ecuaciones normales veremos la solución aproximada para cada una de las 2 b que nos dan y vemos a aunque son muy parecidas los resultados son completamente diferentes, y si, aunque puede parecer que o el algoritmo esta mal o algo raro acaba de pasar basta con dar un vistazo a la matriz A y verla como lo que la estamos interpretando, un conjunto de vectores.

Recuerda que lo que estamos haciendo es encontrar una combinación lineal de estos vectores para aproximar a nuestros vectores b.

Pues si nuestros vectores no son ortogonales lo que pasa es que un pequeño cambio en el resultado nos da una combinación lineal muy diferente. Es justo lo que esta pasando estamos trabajando con vectores muy parecidos.

La mejor forma de explicar este fenomeno y mostrar porque la ortogonalidad es tan importante es ver este video que justo habla de porque $1, x, x^2, \ldots$ es una base horrible para los polinomios. El link es $https://youtu.be/pYoGYQOXqTk$

Esto es un pdf y no me puedo quejar del espacio que utilizo, así que dejo el código:

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework3/Code/Algorithms')
clc;

A23 = [
    0.16 0.10;
    0.17 0.11;
    2.02 1.29;
]

b23 = [
    0.26;
    0.28;
    3.31;
]

b2_23 = [
    0.27;
    0.25;
    3.33;
]

disp("Ax = b")

disp("A:")
disp(A23)

disp("b:")
disp(b23)

x23 = LeastSquares(A23, b23)
disp("x:")
disp(x23)

disp("Ax:")
disp(A23 * x23)

disp("b_2:")
disp(b2_23)

x2_23 = LeastSquares(A23, b2_23)
disp("x_2:")
disp(x2_23)

disp("Ax_2:")
disp(A23 * x2_23)
```

1.2. 24

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework3/Code/Algorithms')
clc;

x24 = [
    1.02;
    0.95;
    0.87;
    0.77;
    0.67;
    0.56;
    0.44;
    0.30;
    0.16;
    0.01;
]

y24 = [
    0.39;
    0.32;
    0.27;
    0.22;
    0.18;
    0.15;
    0.13;
    0.12;
    0.13;
    0.15;
]

A24 = eye(10, 5)
b24 = x24

for i = (1 : 10)
    A24(i, 1) = y24(i) * y24(i)
    A24(i, 2) = x24(i) * y24(i)
    A24(i, 3) = x24(i)
    A24(i, 4) = y24(i)
    A24(i, 5) = 1

    b24(i) = x24(i) * x24(i)
end

disp("Ax = b")

disp("xs:")
disp(x24)

disp("ys:")
disp(y24)

disp("A:")
disp(A24)

disp("b:")
disp(b24)

estimedx24 = LeastSquares(A24, b24)
disp("x:")
disp(estimedx24)

disp("Ax:")
disp(A24 * estimedx24)

clf()



function [x] = solveEquation(a, b, c)
    x = (-1 * b + sqrt(b*b - 4*a*c)) / (2 * a)
endfunction

function [y] = solve(coefficients, x)
    a = coefficients(1)
    b = coefficients(2)
    c = coefficients(3)
    d = coefficients(4)
    e = coefficients(5)

    reala = (a)
    realb = (b * x + d)
```

```
    realc = (c * x  + e - x * x)

    y = solveEquation(reala, realb, realc)
endfunction

someX = linspace(0, 1.10, 50)
someY = someX

for i = (1 : 50)
    someY(i) = solve(estimedx24, someX(i))
end

plot(someX, someY, 'r-')

scatter(x24, y24)
```

1.3. 25

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework2/Code/Algorithms')
clc;


for k = (1 : 10)
    epsilon = 10 ** (-2 * k)

    A25 = [
        epsilon 1;
        1 1;
    ];

    b25 = [
        1 + epsilon;
        2;
    ];

    realX25 = [
        1;
        1;
    ]

    disp("A")
    disp(A25)

    disp("b")
    disp(b25)

    [x] = GaussianElimination(A25, b25);

    disp("Estimated Solution: A \tilde x")
    disp(A25 * x)

    disp("Real Solution: A x")
    disp(A25  * realX25)

    disp("Difference of Error: \tilde x - x")
    disp(x -  realX25)

    disp("Estimated Condition")
    disp( Condition(A25, 10) )

    disp("Real Condition")
    disp( cond(A25) )

end
```

1.4. 26

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework2/Code/Algorithms')
clc;


digitsAltered = eye(12, 1)
digitsClassic = eye(12, 1)
digitsTwo = eye(12, 1)
data = eye(12, 1)

for i = (2 : 12)
    H = eye(i, i)
    data(i) = i

    for i = (1 : i)
        for j = (1 : i)
            H(i, j) = 1 / (i + j - 1)
        end
    end


    [Q26Classic, R26] = GramSchmidt(H, 0)
    [Q26Altered, R26] = GramSchmidt(H, 1)
    [Q26Two, R26] = GramSchmidt(Q26Classic, 0)

    digitsClassic(i) = -log10(norm(eye(i, i) - Q26Classic' * Q26Classic))
    digitsAltered(i) = -log10(norm(eye(i, i) - Q26Altered' * Q26Altered))
    digitsTwo(i)     = -log10(norm(eye(i, i) - Q26Two' * Q26Two))

    disp("Q of GramSchmidt Classical:")
    disp(Q26Classic)

    disp("Digits:")
    disp(digitsClassic)

    disp("Q of GramSchmidt Altered:")
    disp(Q26Altered)

    disp("Digits:")
    disp(digitsAltered)
end

plot(data,digitsClassic, '.b-')
plot(data,digitsAltered, '.r-')
plot(data,digitsTwo, '.g-')

hl=legend(['Classical GramSchmidt'; 'Altered GramSchmidt'; 'Two times GramSchmidt']);
```

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework2/Code/Algorithms')
clc;


digitsAltered = eye(12, 1)
digitsClassic = eye(12, 1)
digitsTwo = eye(12, 1)
digitsHouse = eye(12, 1)
data = eye(12, 1)

for i = (2 : 12)
    H = eye(i, i)
    data(i) = i

    for i = (1 : i)
        for j = (1 : i)
            H(i, j) = 1 / (i + j - 1)
        end
    end


    [Q26Classic, R26] = GramSchmidt(H, 0)
    [Q26Altered, R26] = GramSchmidt(H, 1)
    [Q26Two, R26] = GramSchmidt(Q26Classic, 0)
    [Q26H, R26] = HouseHolder(H)

```

```
    digitsClassic(i) = -log10(norm(eye(i, i) - Q26Classic' * Q26Classic))
    digitsAltered(i) = -log10(norm(eye(i, i) - Q26Altered' * Q26Altered))
    digitsTwo(i)     = -log10(norm(eye(i, i) - Q26Two' * Q26Two))
    digitsHouse(i)   = -log10(norm(eye(i, i) - Q26H' * Q26H))

    disp("Q of GramSchmidt Classical:")
    disp(Q26Classic)

    disp("Digits:")
    disp(digitsClassic)

    disp("Q of GramSchmidt Altered:")
    disp(Q26Altered)

    disp("Digits:")
    disp(digitsAltered)
end

plot(data,digitsClassic, '.b-')
plot(data,digitsAltered, '.r-')
plot(data,digitsTwo, '.g-')
plot(data,digitsHouse, '.black-')

title("Digits of precisition")

hl=legend(['Classical GramSchmidt'; 'Altered GramSchmidt'; 'Two times GramSchmidt'; 'HouseHolder']);
```

```
// @Author: Rosas H26ernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework2/Code/Algorithms')
clc;


digitsAltered = eye(12, 1)
digitsClassic = eye(12, 1)
digitsTwo = eye(12, 1)
digitsNormal = eye(12, 1)
data = eye(12, 1)

for i = (2 : 12)
    H26 = eye(i, i)
    data(i) = i

    for i = (1 : i)
        for j = (1 : i)
            H26(i, j) = 1 / (i + j - 1)
        end
    end


    [Q26Classic, R26] = GramSchmidt(H26, 0)
    [Q26Altered, R26] = GramSchmidt(H26, 1)
    [Q26Two, R26] = GramSchmidt(Q26Classic, 0)
    [L26] = CholeskyGaussian(H26' * H26)
    Q26N = H26 * L26'

    digitsClassic(i) = -log10(norm(eye(i, i) - Q26Classic' * Q26Classic))
    digitsAltered(i) = -log10(norm(eye(i, i) - Q26Altered' * Q26Altered))
    digitsTwo(i)     = -log10(norm(eye(i, i) - Q26Two' * Q26Two))
    digitsNormal(i)  = -log10(norm(eye(i, i) - Q26N' * Q26N))

    disp("Q of GramSchmidt Classical:")
    disp(Q26Classic)

    disp("Q of GramSchmidt Altered:")
    disp(Q26Altered)

end

plot(data,digitsClassic, '.b-')
plot(data,digitsAltered, '.r-')
plot(data,digitsTwo, '.g-')
plot(data,digitsNormal, '.black-')

title("Digits of precisition")

hl=legend(['Classical GramSchmidt'; 'Altered GramSchmidt'; 'Two times GramSchmidt'; 'Normal
    equation']);
```

1.5. 27

Este proceso lo haremos hasta $\epsilon = 2^{-10}$ porque ϵ más pequeñas simplemente ya hacer que se trate a la matriz como singular pues se ve como una matriz con 3 columnas iguales, por lo tanto causa errores al tomarse como una matriz singular.

Lo que podemos ver que mientras epsilon más y más pequeña nos acercamos al claro resultado donde $x_1 = x_2 = x_3 = 0.333$

Algo curioso y diferente de los métodos anteriores, del parcial pasado es que practicamente todos los métodos dan el mismo resultado, curiosamente GramSchmidt es el único diferente en ciertos ϵ.

```
// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

getd('/Users/mac/Documents/Projects/Learning/UNAM/NumericalAnalysis/Homework3/Code/Algorithms')
clc;


values = [
    2;
    1.1;
    1;
    0.9;
    2e-01;
    2e-02;
    2e-05;
    2e-08;
    2e-10;
    2e-11;
]

[numberOfValues, _] = size(values)

for i = (1 : numberOfValues)

    e = values(i)
    disp("============")
    disp("epsilon:")
    disp(e)

    A27 = [
        1 1 1;
        e 0 0;
        0 e 0;
        0 0 e;
    ]

    b27 = [
        1;
        0;
        0;
        0;
    ]

    disp("Ax = b")

    disp("A:")
    disp(A27)

    try
        x27LeastSquares = LeastSquares(A27, b27)

        [Q27HH, R27HH] = HouseHolder(A27)
        x27HouseHolder = QRDecomposition(Q27HH, R27HH, b27)

        [Q27GS, R27GS] = GramSchmidt(A27, 0)
        x27GramSchmidt = QRDecomposition(Q27GS, R27GS, b27)

        [Q27G, R27G] = Givens(A27)
        x27Givens = QRDecomposition(Q27G, R27G, b27)


        disp("x of Least Squares:")
        disp(x27LeastSquares)
```

```
        disp("Ax of Least Squares:")
        disp(A27 * x27LeastSquares)

        disp("x of HouseHolder:")
        disp(x27HouseHolder)

        disp("Ax of HouseHolder:")
        disp(A27 * x27HouseHolder)

        disp("x of GramSchmidt:")
        disp(x27GramSchmidt)

        disp("Ax of GramSchmidt:")
        disp(A27 * x27GramSchmidt)

        disp("x of Givens:")
        disp(x27Givens)

        disp("Ax of Givens:")
        disp(A27 * x27Givens)
    catch
        disp("Error: Singular matrix")
    end




end
```

2. Anexo

2.1. Givens

```
// Get the estimated solution to Ax = b using the HouseHolder transformation
// @param: A a matriz in M_{m x n} where m > n
// @return: Q a matriz in M_{m x m} that is ortogonal
// @return: R a matriz in M_{m x n} that is triangular superior

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [Q, R] = Givens(A)
    [m, n] = size(A);
    Q = eye(m, m);
    R = A;

    for j = (1 : n)
        for i = (m : -1 : j + 1)
            GivenMatrix = eye(m, m);
            [c, s] = GivensRotations(R(i-1, j), R(i, j));
            GivenMatrix([i-1, i], [i-1, i]) = [c -s; s c];

            R = GivenMatrix' * R;
            Q = Q * GivenMatrix;
        end
    end

end

function [c,s] = GivensRotations(a,b)
    if (b == 0)
        c = 1;
        s = 0;
    else
        if abs(b) > abs(a)
            r = a / b;
            s = 1 / sqrt(1 + r**2);
            c = s * r;
        else
            r = b / a;
            c = 1 / sqrt(1 + r**2);
            s = c * r;
        end
    end

end
```

2.2. GramSchmidt

```
// Get the estimated solution to Ax = b using the Gramm-Schmmidt transformation
// @param: A a matriz in M_{m x n} where m > n
// @param: option if 1 then is the alterate else is the classic algorithm
// @return: Q a matriz in M_{m x m} that is ortogonal
// @return: R a matriz in M_{m x n} that is triangular superior


// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [Q, R] = GramSchmidt(A, option)
    [m, n] = size(A);

    Q = zeros(m, n);
    R = A

    for j = (1:n)

        Q(:,j) = A(:, j)

        for i = (1 : j - 1)
            if (option == 1)
                R(i, j) = Q(:, i)' * Q(:, j);
            else
                R(i, j) = Q(:, i)' * A(:, j);
            end

            Q(:, j) = Q(:, j) - R(i, j) * Q(:, i);
        end

        R(j, j) = norm(Q(:, j));
        Q(:, j) = Q(:, j) / R(j, j);
    end

endfunction
```

2.3. HouseHolder

```
// Get the estimated solution to Ax = b using the HouseHolder transformation
// @param: A a matriz in M_{m x n} where m > n
// @return: Q a matriz in M_{m x m} that is ortogonal
// @return: R a matriz in M_{m x n} that is triangular superior

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [Q, R] = HouseHolder(A, b)
    [m, n] = size(A);
    Q = eye(m, m);


    for i = (1 : n)
        aei = zeros(m - i + 1, 1);

        alpha = -sign(A(i,i)) * norm(A(i : m, i));
        aei(1) = alpha

        v = A(i : m, i) - aei;
        HouseHolder = eye(m - (i-1), m-(i-1)) -2 * ((v * v') / (v' * v));

        RealHouseHolder = eye(m, m);
        RealHouseHolder(i : m, i : m) = HouseHolder;

        A = RealHouseHolder * A;
        Q = Q * RealHouseHolder;
    end
    R = A;

endfunction
```

2.4. LeastSquares

```
// Get the estimated solution to Ax = b using least squares
// @param: A a matriz in M_{m x n} where m > n
// @param: b a vector of m rows
// @return: x a estimated solution

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [x] = LeastSquares(A, b)
    [L] = CholeskyBanachiewicz(A' * A, 1);

    y = FowardSubstitution(L, A' * b);
    x = BackwardSubstitution(L', y);
endfunction
```

2.5. BackwardSubstitution

```
// Solve a system Ux = y where U is an upper triangular
// using the famous algorithm backward substitution
// @param: U triangular superior matrix
// @param: b the b in Ux = b
// @return: x the solution vector

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [x] = BackwardSubstitution(U, b)
    [m, n] = size(U);
    x = zeros(n, 1);

    for i = (n : -1 : 1)
        if (U(i, i) == 0)
            error('Error: Singular matrix');
            return;
        end

        x(i) = b(i) / U(i,i);

        for j = (1 : i - 1)
            b(j) = b(j) - U(j, i) * x(i);
        end
    end
endfunction
```

2.6. CholeskyBanachiewicz

```
// Factor A as A = L * L^T
// using the famous algorithm called Cholesky using this really awesome propierty
// First A = L U then we make U a unit upper triangular matrix so we have L D L' and then
// we do L D2 D2 L' were D2(i, j) = sqrt(D(i, j)) finally we associate and we have A = L2 * L2'
// where L2 = L * D2
// @param: A a positive defined matrix (so A is symetric)
// @param: option if 1 then A = L * L else A = L * D * L
// @return: L lower triangule matrix

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [L, D] = CholeskyBanachiewicz(A, option)
     [m, n] = size(A);
    D = eye(n, n);
    L = eye(m, n);
    U = A;

    for step = (1 : n - 1)
        if (A(step, step) == 0)
            error('Error: Singular matrix');
            return;
        end

        for row = (step + 1 : n)
            L(row, step) = U(row, step) / U(step, step);
            for column = (1 : n)
                U(row, column) = U(row, column) - L(row, step) * U(step, column);
            end
        end
    end

    if option == 1
        for step = (1 : n)
            for row = (step : n)
                L(row, step) = L(row, step) * sqrt(U(step, step));
            end
        end
    else
        for step = (1 : n)
            D(step, step) = U(step, step);
        end
    end



endfunction
```

2.7. CholeskyGaussian

```
// Factor A as A = L * L^T
// using the famous algorithm called Cholesky using a modification of Gaussian Elimination
// @param: A a positive defined matrix (so A is symetric)
// @return: L lower triangule matrix

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [L] = CholeskyGaussian(A)

    [m, n] = size(A);
    L = zeros(m, n);

    for step = (1 : n)
        A(step, step) = sqrt( A(step, step) );

        for column = (step + 1 : n)
            A (column, step) = A(column, step) / A(step, step);
        end

        for i = (step + 1 : n)
            for j = (step + 1 : n)
                A(i, j) = A(i, j) - A(i, step) * A(j, step);
            end
        end
    end

    for row = (1 : n)
        for column = (1 : n)
            if (row >= column)
                L(row, column) = A(row, column);
            end
        end
    end

endfunction
```

2.8. CompleteLUDecomposition

```
// Factor A as PAQ = LU
// @param: A a not singular matrix
// @return: L (not sure) lower triangule matrix
// @return: U upper triangule matrix
// @return: P permutation matrix
// @return: Q permutation matrix

// JUST BECAUSE I WRITE IT, IT DOES NOT MEAN IT IS FAST, IT IS NOT FAST!

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [L, U, P, Q] = CompleteLUDecomposition(A)
    [m, n] = size(A);
    if (m ~= n) == 0 then
        error('Error: Not square matrix');
    end
    P = eye(n, n);
    Q = eye(n, n);
    L = eye(n, n);
    U = A;

    for step = (1 : n - 1)
        Qi = eye(n, n);
        Pi = eye(n, n);

        [maxIndex, index] = max(abs(A(step : n, step : n)));
        index(1) = index(1) + step - 1;
        index(2) = index(2) + step - 1;

        if (maxIndex == 0)
            error('Error: Singular matrix');
        end

        temporal = Pi(step, :);
        Pi(step, :) = Pi(index(1), :);
        Pi(index(1), :) = temporal;

        temporal = Qi(:, step);
        Qi(:, step) = Qi(:, index(2));
        Qi(:, index(2)) = temporal;

        temporal = U(step, :);
        U(step, :) = U(index(1), :);
        U(index(1), :) = temporal;

        temporal = U(:, step);
        U(:, step) = U(:, index(2));
        U(:, index(2)) = temporal;

        for row = (step + 1 : n)
            L(row, step) = U(row, step) / U(step, step);
            for column = (1 : n)
                U(row, column) = U(row, column) - L(row, step) * U(step, column);
            end
        end

        Q = Q * Qi;
        P = Pi * P;
    end

endfunction
```

2.9. FowardSubstitution

```
// Solve a system Ly = b where L is triangular inferior
// using the famous algorithm foward substitution
// @param: L triangular inferior matrix
// @param: b the b in Ly = b
// @return: x the solution vector

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [y] = FowardSubstitution(L, b)
    [m, n] = size(L);
    y = zeros(n, 1);

    for i = (1 : n)
        if (L(i, i) == 0)
            error('Error: Singular matrix');
            return;
        end

        y(i) = b(i) / L(i, i);

        for j = (i + 1 : n)
            b(j) = b(j) - L(j, i) * y(i);
        end
    end
endfunction
```

2.10. LUDecomposition

```
// Factor A as A = L * U
// @param: A a not singular matrix
// @return: L lower triangule matrix
// @return: U upper triangule matrix

// @Author: Rosas Hernandez Oscar Andres
// @Author: Alarcón Alvarez Aylin Yadira Guadalupe
// @Author: Laurrabaquio Rodríguez Miguel Salvador
// @Author: Pahua Castro Jesús Miguel Ángel

function [L, U] = LUDecomposition(A)
    [m, n] = size(A);
    L = eye(m, n);
    U = A;

    for step = (1 : n - 1)

        if (A(step, step) == 0)
            error('Error: Singular matrix');
            return;
        end

        for row = (step + 1 : n)
            L(row, step) = U(row, step) / U(step, step);

            for column = (1 : n)
                U(row, column) = U(row, column) - L(row, step) * U(step, column);
            end
        end
    end

endfunction
```