UNIVERSIDAD EAFIT
IT and Systems Deparment
Project Choice
First Delivery

Course: Numerical Analysis

Teacher: Edwar Samir Posada Murillo

Semester: 2022-1

Delivery date of this report: 06-04-2020

Name of the project: Numerical Methods

Project web address (repository): https://github.com/alejo128/Metodos-Numerico

Members:

José Alejandro Díaz Urrego

Project description: the development of a web page will be carried out where the different methods that will be seen throughout the semester are gathered

For this first installment, a document will be produced in which the codes implemented for the following methods will be evidenced:

- Solution of equations in one variable
    - Bisection
    - Incremental searches
    - False rule
    - Fixed point
    - Newton
    - Secant
    - Multiple roots
- Solution of systems of linear equations
    - Gaussian elimination
    - Partial pivot
    - Full pivot

Incremental Search Method

- Pseudocode:

```
Process Incremental search method
        Read Xo, Delta, Iter
        Yo = f(Xo)

        If Yo = 0 Then
            Show: 'Xo is Root'
        Else If
            X1 = Xo + Delta
            Count = 1
            Y1 = f(X1)
            While Yo * Y1 > 0 & Count < Iter Do
                Xo = X1
                Yo = Y1
                X1 = Xo + Delta
                Y1 = f(X1)
                Count = Count + 1
            End While
            If Y1 = 0 Then
                    Show: 'Xo is Root'
            Else If Yo * Y1 < 0 Then
                        Show: 'There's a Root Between Xo and X1'
                Else If
                        Show: 'Fail in 'Iter' Iterations'
                End If
            End If
        End If
End Process
```

Bisection Method

- Pseudocode

```
Process Bisection method
        Read Xi, Xs, Tol, Iter
        Yi = f(Xi)
        Ys = f(Xs)

        If Yi = 0 Then
            Show: 'Xi is Root'
        Else If
            If Ys = 0 Then
                Show}: 'Xs is Root'
            Else If
                    If Yi * Ys < 0 Then
                        Xm = (Xi + Xs) / 2
                        Count = 1
                        Ym = f(Xm)
                        Error = Tol + 1
                            While Error > Tol & Ym ≠ 0 & Count < Iter Do
                                If Yi * Ym < 0 Then
                                    Xs = Xm
                                    Ys = Ym
                                Else If
                                    Xi = Xm
                                    Yi = Ym
                                End If
                                    Xaux = Xm
                                    Xm = (Xi + Xs) / 2
                                    Ym = f(Xm)
                                    Error = Abs(Xm - Xaux)
                                    Count = Count + 1
                            End While
                            If Ym = 0 Then
                                    Show: 'Xm is Root'
                            Else If Error < Tol Then
                                    Show: 'Xm is approximation to a root with a tolerance 'Tol''
                                Else If
                                    Show: 'Fail in 'Iter' iterations'
                                End If
                            End If
                    Else If
                        Show: 'The interval is inadequate'
                    End If
                End If
        End If
End Process
```

False Rule Method

- Pseudocode

```
Process False rule method
        Read Xi, Xs, Tol, Iter
        Yi = f(Xi)
        Ys = f(Xs)

        If Yi = 0 Then
            Show: 'Xi is Root'
        Else If
            If Ys = 0 Then
                Show: 'Xs is Root'
            Else If
                If Yi * Ys < 0 Then
                    Xm = Xi - ((Yi * (Xs - Xi)) / (Ys - Yi))
                    Count = 1
                    Ym = f(Xm)
                    Error = Tol + 1

                    While Error > Tol & Ym ≠ 0 & Count < Iter Do
                        If Yi * Ym < 0 Then
                            Xs = Xm
                            Ys = Ym
                        Else If
                            Xi = Xm
                            Yi = Ym
                        End If
                        Xaux = Xm
                        Xm = Xi - ((Yi * (Xs - Xi)) / (Ys - Yi))
                        Ym = f(Xm)
                        Error = Abs(Xm - Xaux)
                        Count = Count + 1
                    End While
                    If Ym = 0 Then
                        Show: 'Xm is Root'
                    Else If Error < Tol Then
                            Show: 'Xm is approximation to a root with a tolerance 'Tol''
                        Else If
                            Show: 'Fail in 'Iter' iteration'
                        End If
                    End If
                Else If
                    Show: 'The interval is inadequate'
                End If
            End If
        End If
End Process
```

Fixed Point Method

- Pseudocode

```
Process Fixed point method
        Read Xo, Tol, Iter
        Yo = f(Xo)
        Count = 0
        Error = Tol + 1

        While Yo ≠ 0 & Error > Tol & Count < Iter Do
            Xn = g(Xo)
            Yo = f(Xn)
            Error = abs((Xn - Xo) / Xn)
            Xo = Xn
            Count = Count + 1
        End While
            If Yo = 0 Then
                Show: 'Xo is Root'
            Else If Error < Tol Then
                Show: ''Xo' is approximation to a root with a tolerance 'Tol''
                Else If
                    Show: 'Failure in 'Iter' iterations'
                End If
            End If
End Process
```

Newton’s Method

- Pseudocode

```
Process Newton's method
        Read Xo, Tol, Iter
        Yo = f(Xo)
        Bo = f`(Xo)
        Count = 0
        Error = Tol + 1

        While Yo ≠ 0 & Bo ≠ 0 & Error > Tol & Count < Iter Do
            X1 = Xo - (Yo / Bo)
            Yo = f(X1)
            Bo = f`(X1)
            Error = abs((X1 - Xo) / X1)
            Xo = X1
            Count = Count + 1
        End While
            If Yo = 0 Then
                Show: 'Xo is Root'
            Else If Error < Tol Then
                Show: ''Xo' is approximation to a root with a tolerance 'Tol''
                Else If Do = 0 Then
                        Show: ''Xo' is possibly a multiple root'
                    Else If
                        Show: 'Failure in 'Iter' iterations'
                    End If
                End If
            End If
End Process
```

Secant Method

- Pseudocode

```
Process Secant method
    Read X1, Xo, Tol, Iter
    Yo = f(Xo)
    If Yo = 0 Then
        Show: 'Xo is root'
        Else If
            Yi = f(X1)
            Count = 0
            Error = Tol + 1
            Aux = Y1 - Yo

            While Error > Tol & Y1 ≠ 0 & Aux ≠ 0 & Count < Iter Do
                X2 = X1 - (( Y1 * (X1 - Xo)) / Aux)
                Error = Abs((X2- X1) / X2)
                Xo = X1
                Yo = Y1
                X1 = X2
                Y1 = f(X1)
                Aux = Y1 - Yo
                Count = Count + 1
            End While

            If Y1 = 0 Then
                Show: 'X1 is Root'
            Else If Error < Tol Then
                    Show: ''X1' is approximation to a root with a tolerance 'Tol''
                Else If Aux = 0 Then
                    Show: 'There's possibly a multiple root'
                Else If
                    Show: 'Failure in 'Iter' iterations'
                End If
            End If
        End If
    End If
End Process
```

Multiple Roots Method

- Pseudocode

```
Process Multiple roots method
    Read Xo, Tol, Iter
    Yo = f(Xo)
    D1o = f`(Xo)
    D2o = f``(Xo)
    Deno = D1o^2 - (Yo * D2o)
    Count = 0
    Error = Tol + 1

    While Yo ≠ 0 & Deno ≠ 0 & Error > Tol & Count < Iter Do
        X1 = Xo - ((Yo * D1o) / Deno)
        Yo = f(X1)
        D1o = f`(X1)
        D2o = f``(X1)
        Error = abs((X1 - Xo) / X1)
        Deno = D1o^2 - (Yo * D2o)
        Xo = X1
        Count = Count + 1
    End While
        If Yo = 0 Then
            Show: 'Xo is Root'
        Else If Error < Tol Then
                Show: ''Xo' is an approximate root with a tolerance 'Tol''
            Else If Deno = 0 Then
                    Show: 'The denominator becomes zero'
                Else If
                    Show: 'Failure in 'Iter' iterations'
                End If
            End If
        End If
End Process
```

Simple Gaussian Elimination Method

- Pseudocode

```
Process Simple gaussian elimination method
    Read A, b
    (n,m) = size(A)
    a = AugmentedMatrixForm (A,b)

    If n = m Then
        for k = 1 to n-1
            for i = k + 1 to n
                M(ik) = a(ik) / a(kk)
                for j = k to n + 1
                    a(ij) = a(ij) - m(ik)a(kj)
                End for
            End for
        End for

        for i = n to 1
            sum = 1
            for p = i + 1 to n
                sum = sum + a(ipXp)
            End for
            Xi = (bi - sum) / a(ii)
        End for
    Else If
        show: 'The matrix is not square'
    End If
    print a
    print x
End Process
```

Gaussian Elimination Method with Partial Pivoting

- Pseudocode

```
Process partial pivoting method
    Read A, b
    (n,m) = size(A)
    a = AugmentedMatrixForm (A,b)

    If n = m Then
        for k = 1 to n-1
            higher = 0
            filem = k
            for p = k  to n
                If higher < |a(pk)| Then
                    higher = |a(pk)|
                    filam = p
                End If
            End for
            If higher = 0 Then
                Show: 'Suspended the process, infinite solutions'
            Else If
                If filam ≠ k Then
                    for j = 1 to n+1
                        Aux = a(k,j)
                        a(k,j) = a(filam,j)
                        a(filam, j) = Aux
                    End for
                End If
            End If
            for i=k+1 to n
                m(ik) = a(ik) / a(kk)
                for j=k to n+1
                    a(ij) = a(ij) - m(ik)a(kj)
                End for
            End for
        End for
        for i = n to 1
            sum = 0
            for p = i + 1 to n
                sum = sum + a(ip X p)
            End for
            Xi = (bi - sum) / a(ii)
        End for
    Else If
        show: 'The matrix is not square'
    End If
    print a
    print x
End Process
```

Gaussian Elimination Method with Total Pivoting

- Pseudocode

```
Process total pivoting method
    Read A, b
    (n,m) = size(A)
    a = AugmentedMatrixForm (A,b)

    If n = m Then
        for i = 1 to n
            mark(i) = i
        End for
        for k = 1  to n
            higher = 0
            filem = k
            columnm = k
            for p = k to n
                for r = k to n
                    If higher < |a(pr)| Then
                        higher = |a(pr)|
                        filem = p
                        columnm = r
                    End If
                End for
            End for
            If higher = 0 Then
                Show: 'Suspended the process, infinite solutions'
            Else If
                If filem ≠ k Then
                    for j = 1 to n+1
                        Aux = a(k,j)
                        a(k,j) = a(filem,j)
                        a(filem,j) = Aux
                    End for
                    Aux = mark(k)
                    mark(k) = mark(columnm)
                    mark(columnm) = Aux
                End If
            End If
            for i = k + 1 to n
                m(ik) = a(ik) / a(kk)
                for j = k to n+1
                    a(ij) = a(ij) - m(ik)a(kj)
                End for
            End for
            for i = n to 1
                sum = 0
                for p = i + 1 to n
                    sum = sum + a(ip X p)
                End for
                x(i) = (b(i) - sum) / a(ii)
            End for
            for i = n to 1
```

```
                for j = 1 to n
                    If mark(j) = i Then
                        k = j
                    End If
                End for
                Aux = x(k)
                x(k) = x(i)
                x(i) = Aux
                Aux = mark(k)
                mark(k) = mark (i)
                mark(i) = Aux
            End for
        End for
    Else If
        Show:'The matrix is not square'
    End If
     print a
     print x
End Process
```