EAFIT UNIVERSITY

DEPARTMENT OF INFORMATICS AND SYSTEMS

PROJECT CHOICE

Second Report

September 27, 2022

Course: Numerical Analysis

Teacher: Edwar Samir Posada Murillo

Semester: 2022-2

Jacobo Rave Londoño

Juan David Echeverri Villada

Kevin Alejandro Sossa Chavarria

Juan Sebastián Guerra Hernández

Next, you will find recreated methods in pseudocode:

1: Bisection

```
function Bisection(f,left,right,tol,niter)
    fRight ← f(right)
    Left ← f(right)
    if fRight = 0 then
        WRITE "right is a root"
    else if fLeft = 0 then
        WRITE "left is a root"
    else if fLeft * fRight < 0 then
        mid ← (left * right)/2
        fmid ← f(mid)
        counter ← 1
        error ← tol + 1
        while error > tol AND fmid ≠ 0 AND counter < niter do
            if fLeft * fmid < 0 then
                right ← mid
                fRight ← fmid
            else
                left ← mid
                fLeft ← fmid
            end if
            aux ← mid
            mid ← (right + left)/2
            fmid ← f(mid)
            error ← |mid − aux|
            Counter ← counter + 1
        end while
        if fmid = 0 then
            WRITE "mid is a root"
        else if error < tol then
            WRITE "mid ia an approximation with tolerance"
        else
            WRITE "The method fails in niter iterations"
        end if
    end if
end function
```

2: Newton

```
function Newton(f, fder, tol, x0, niter)
    fx ← f(x0)
    dfx ← fder(x0)
    counter ← 1
    error ← tol + 1
    while error > tolerance & fx ≠ 0 & counter < niter do
        x1 ← x0 − (fx/dfx)
        fx ← f(x1)
        dfx ← fder(x1)
        error ← |x1 − x0|
        x1 ← x0
        counter ← counter + 1
    end while
    if fx = 0 then
        WRITE "x0 is a root of f"
    else if error < tolerance then
        WRITE "x1 is a root approximation with tolerance tol"
    else
```

```
        WRITE "The method failed at niter iteration"
    end if
end function
```

3: Incremental Search

```
function IncrementalSearch(f,x0,delta,niter)
    fx0 ← f(x0)
    if fx0 = 0 then
        answer ←x0 is a root
    else
        x1 ← x0 + delta
        counter ← 1
        fx1 ← f(x1)
        while fx0 * fx1 > 0 AND counter > niter do
            x0 ← x1
            fx0 ← fx1
            x1 ← x0 + delta
            fx1 ← f(x1)
            counter ← counter + 1
        end while
        if fx1 = 0 then
            answer ←x1 is a root
        else if fx0 * fx1 < 0 then
            answer ←There is at least one root between x0 and x1
        else
            answer ←The method fails in niter iterations
        end if
    end if
end function
```

4: Fixed Point

```
function Fixed point(f, g, tol, x0, niter)
    fx ← f(x0)
    gx ← convergentformoff(x)
    counter ← 1
    error ← tol + 1
    while error > tolerance & f(x0) ≠ 0 & counter < niter do
        x1 ← g(x0)
        error ← |x1 − x0|
        x0 ← x1
        counter ← counter + 1
    end while
    if fx = 0 then
        WRITE "x1 is a root approximation with tolerance tol"
    else if error < toleranc then
        WRITE "x1 is a root approximation with tolerance tol"
    else
        WRITE "The method failed at niter iteration"
```

5: Multiple Root

```
function MultipleRoot(f,fl,f2,x0,tolerance,nMax)
    xi ← x0
    fxi ← f(xi)
    if fxi = 0 then
        WRITE "A root was found: xi"
    else
        counter ← 0
```

```
        f1xi ← f1(xi)
        f2xi ← f2(xi)
        error ← tolerance + 1
        det ← (f1xi^2) − (fxi ∗ f2xi)
        while fxi ≠ 0 & error > tolerance & counter < nMax & det ≠ 0 do
            xiAux ← xi
            x1 ← x1 − ((fxi ∗ f1xi)/((f1xi^2) − (fxi ∗ f2xi)))
            fxi ← f(xi)
            f1xi ← f1(xi)
            f2xi ← f2(xi)
            error ← |xi − xiAux|
            det ← (f1xi^2) − (fxi ∗ f2xi)
            counter ← counter + 1
        end while
        if fx1 = 0 then
            WRITE "A root was found: xi"
        else if error ≤ tolerance then
            WRITE "One approach is: xi"
        else if det = 0 then
            WRITE "Method failure"
        else
            WRITE "The method fails with the maximum number of iterations given"
        end if
    end if
    x ← xi
end function=0
```

6: Secant

```
function Secant(x0, x1, tol, iter, f)
    y0 ← f(x0)
    if y0 = 0 then
        WRITE "x0 is a root of f"
    else
        y1 ← f(x1)
        d ← y1 − y0
        error ← tol + 1
        cont ← 0
        while y1 ≠ 0 & error > tol & cont < iter & d ≠ 0 do
            x2 ← x1 − ((y1 ∗ (x1 − x0))/(d))
            error ← |x2 − x1|
            x0 ← x1
            y0 ← y1
            y1 ← f(x2)
            x1 ← x2
            d ← y1 − y0
            counter ← counter + 1
        end while
        if y1 = 0 then
            WRITE "x1 is a root of f"
        else
            if error < tol then
                WRITE "x1 is an approximation to a root with a tolerance tol"
            else
                if d = 0 then
                    WRITE "denominator is zero, FAILURE"
                else
                    WRITE "failure in iter iterations"
                end if
            end if
```

```
        end if
    end if
end function
```

7: False Rule

```
function falseRule(Fun, Xi, Xs, Tol, Iter)
    Yi ← f(Xi)
    Ys ← f(Xs)
    if Yi = 0 then
        WRITE Xi is the root
    else
        if Ys = 0 then
            WRITE Xs is the root
        else
            if Yi * Ys < 0 then
                Xm ← (Xi) − ((f(Xi) * (Xi − Xs))/(f(Xi) − f(Xs)))
                Ym ← f(Xm)
                Error ← Tol + 1
                Cont ← 1
                while Ym ≠ 0 & Error > Tol & Cont < Iter do
                    if Yi*Ym<0 then
                        Xs ← Xm
                        Ys ← Ym
                    else
                        Xi ← Xm
                        Yi ← Ym
                    end if
                    Xaux ← Xm
                    Xm ← (Xi) − ((f(Xi) * (Xi − Xs))/(f(Xi) − f(Xs)))
                    Ym ← f(Xm)
                    Error ← |Xm − Xaux|/Xm
                    Cont ← Cont + 1
                end while
                if Ym = 0 then
                    WRITE "Xm is a root of f"
                else
                    if Error < Tol then
                        WRITE "Xm is an approximation to a root with a tolerance Tol"
                    else
                        WRITE "failure in Iter iterations"
                    end if
                end if
                WRITE Iterations | Xi | Xs | Xm | Ym | Error
            else
                WRITE "The interval is inadequate"
            end if
        end if
    end if
end function
```

8: GaussSimple

```
function GaussSimple(A,b,n,delta,niter)
    A ← Concat(A, b)
    for i ← 1, n − 1 do
        if Ai, i = 0 then
            WRITE Mathematical Error! Stop
        end if
        for j ← i + 1, n do
            ratio ← Aj, i/Ai, i
```

```
            for k ← 1, n + 1 do
                Aj, k ← Aj, k − ratio ∗ Ai, k
            end for
        end for
    end for
    Xn ← An, n + 1/An, n
    for i ← n − 1, 1, step = −1 do
        Xi ← Ai, n + 1
        for j ← i + 1, n do
            Xi ← Xi − Ai, j ∗ Xj
        end for
        Xi ← Xi/Ai, i
    end for
    WRITE "Answer vector"
    for i ← 1, n do
        WRITE Xi
    end for
end function
```

9: GaussPartial

```
function GaussPartial(A,b,n,delta,niter)
    A ← Concat(A, b)
    for i ← 1, n − 1 do
        WRITE changeRows(A,i)
        if Ai, i = 0 then
            WRITE Mathematical Error! Stop
        end if
        for j ← i + 1, n do
            ratio ← Aj, i/Ai, i
            for k ← 1, n + 1 do
                Aj, k ← Aj, k − ratio ∗ Ai, k
            end for
        end for
    end for
    Xn ← An, n + 1/An, n
    for i ← n − 1, 1, step = −1 do
        Xi ← Ai, n + 1
        for j ← i + 1, n do
            Xi ← Xi − Ai, j ∗ Xj
        end for
        Xi ← Xi/Ai, i
    end for
    WRITE "Answer vector"
    for i ← 1, n do
        WRITE Xi
    end for
end function
```

10: GaussTotal

```
function GaussTotal(A,b,n,delta,niter)
    A ← Concat(A, b)
    for i ← 1, n − 1 do
        WRITE changeRowsAndColumns(A,i)
        if Ai, i = 0 then
            WRITE Mathematical Error! Stop
        end if
        for j ← i + 1, n do
            ratio ← Aj, i/Ai, i
            for k ← 1, n + 1 do
```

```
            Aj,k ← Aj,k − ratio ∗ Ai,k
        end for
    end for
end for
Xn ← An,n+1/An,n
for i ← n−1,1,step = −1 do
    Xi ← Ai,n+1
    for j ← i+1,n do
        Xi ← Xi − Ai, j ∗ Xj
    end for
    Xi ← Xi/Ai,i
end for
WRITE "Answer vector"
for i ← 1,n do
    WRITE Xi
end for
end function
```

Repository

The repository with the evidence related to the project will be: NumericalAnalysisProject

Description

This project aims to develop a web app to calculate data using different numerical methods as well as an API that let users make request to solve their problems. Also, the web app will have the option of visualising the data in a 2D graph.

Added Values

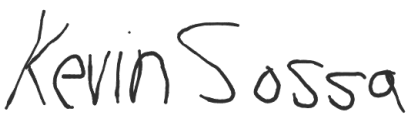
- The project will be done in English
- The project will have its documentation in LaTeX
- The numerical algorithms can be found in multiple programming languages.
- The project will have extra numerical methods

Members signatures

1: Jacobo Rave

2: David Echeverri

3: Kevin Sossa



4: Sebastián Guerra