EAFIT UNIVERSITY
DEPARTMENT OF INFORMATICS AND SYSTEMS
PROJECT CHOICE

Third Report
October 31, 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

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

Pseudocodes

1: Incremental Search

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

2: Bisection

```
function Bisection(f,left,right,tol,niter)
    fRight ← f(right)
    fLeft ← f(left)
    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 is an approximation with tolerance = tol
        else
            WRITE The method failed at niter iterations
        end if
    else
        WRITE Inappropriate interval
    end if
end function
```

3: False Rule

```
function falseRule(f, Xi, Xs, Tol, Iter)
    Yi ← f(Xi)
    Ys ← f(Xs)
    if Yi = 0 then
        WRITE Xi is a root
    else
        if Ys = 0 then
            WRITE Xs is a 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 AND Error > Tol AND 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
            else
                WRITE The interval is inadequate
            end if
        end if
    end if
end function
```

4: Fixed Point

function Fixed point(f, g, tol, x0, niter) ▷ g is the convergent form of f
    $fx \leftarrow f(x0)$
    $counter \leftarrow 0$
    $error \leftarrow tol + 1$
    while $fx \neq 0$ AND $error > tol$ AND $counter < niter$ **do**
        $x1 \leftarrow g(x0)$
        $fx \leftarrow f(x1)$
        $error \leftarrow |x1 - x0|$
        $x0 \leftarrow x1$
        $counter \leftarrow counter + 1$
    end while
    if $fx = 0$ **then**
        WRITE x1 is a root
    else if $error < tol$ **then**
        WRITE x1 is a root approximation with tolerance = tol
    else
        WRITE The method failed at niter iteration
    end if
end function

5: Newton

function Newton(f, fder, tol, x0, niter) ▷ fder is the first derivative of f
    $fx \leftarrow f(x0)$
    $dfx \leftarrow fder(x0)$
    $counter \leftarrow 0$
    $error \leftarrow tol + 1$
    while $error > tol$ AND $fx \neq 0$ AND $dfx \neq 0$ AND $counter < niter$ **do**
        $x1 \leftarrow x0 - (fx/dfx)$
        $fx \leftarrow f(x1)$
        $dfx \leftarrow fder(x1)$
        $error \leftarrow |x1 - x0|$
        $x0 \leftarrow x1$
        $counter \leftarrow counter + 1$
    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 if $dfx = 0$ **then**
        WRITE x1 is a possible multiple root
    else
        WRITE The method failed at niter iteration
    end if
end function

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 AND error > tol AND cont < iter AND 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 There is a multiple root
                else
                    WRITE failure in iter iterations
                end if
            end if
        end if
    end if
end function
```

7: Multiple Root

```
function MultipleRoot(f,f1,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 AND error > tolerance AND counter < nMax AND 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
```

8: Simple Gauss Elimination

```
function GaussSimple(A,b,n)
    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
            if Aj,i! = 0 then
                Aj ← Aj − (Aji/Aii) ∗ Ai
            end if
        end for
        display(A)
    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: Gauss Elimination with Partial Pivoting

```
function GaussPartial(A,b,n)
    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
            if Aj,i! = 0 then
                Aj ← Aj − (Aji/Aii) ∗ Ai
            end if
        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: Gauss Elimination with Total Pivoting

```
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
```

11: Steffensen

```
function Steffensen(f,x0,tol,n)
    for i = 1...n do
        x1 ← f(x0)
        x2 ← f(x1)
        p ← p0 − (p1 − p0)^2/(p2 − 2 ∗ p1 + p0)
        error ← absoluteValue(p − p0)
        p0 ← p
        count ← count + 1
        p0 ← p
    end for
    if f(p0) = 0 then
        WRITE p0 is a root
    else if error < tol then
        WRITE p0 is an approximation with tolerance = tol
    else
        WRITE failed to converge
    end if
end function
```

12: Aitken

```
function Aitken(f,x0,tol,n)
    x1 ← f(x0)
    x2 ← f(x1)
    fxi ← f(x2)
    det ← (x2 − x1) − (x1 − x0)
    counter ← 4
    while fxi ≠ 0 AND error > tol AND counter < n AND det ≠ 0 do
        xi ← ((x2 − x1)^2)/det
        fxi ← f(xi)
        det ← (x2 − x2) − (x1 − x0)
        error ← absoluteValue(xi − x2)
        x0 ← xi
        x1 ← f(x0)
        x2 ← f(x1)
        counter ← counter + 1
    end while
    if fxi = 0 then
        WRITE "xi is a root"
    else if error <= tol then
        WRITE "xi is an approximation"
    else if det = 0 then
        WRITE "Error during method execution"
    else
        WRITE "The method fails with the maximum number of iterations"
    end if
end function
```

13: Muller

```
function Muller(f, x0, tol, nMax)
    fx0 ← f(x0)
    fx1 ← f(x1)
    x1 ← (x0 + x1)/2
    fx2 ← f(x2)
    h0 ← x1 − x0
    h1 ← x2 − x1
    delta0 ← (fx1 − fx0)/h0
    delta1 ← (fx2 − fx1)/h1
    a ← (delta1 − delta0)/(h1 − h0)
    b ← a ∗ h1 + delta1
    c ← fx2
    xi ← x2 + (−2 ∗ c)/(b + (b/|b|) ∗ sqrt(b^2 − 4 ∗ a ∗ c)
    fxi ← f(xi)
    error ← tol + 1
    counter ← 0
    while fx1 ≠ 0 & error > tol & counter < nMax do
        x2Aux ← x2
        x1Aux ← x1
        x2 ← x1
        x1 ← x2Aux
        x0 ← x1Aux
        fx0 ← f(x0)
        fx1 ← f(x1)
        fx2 ← f(x2)
        h0 ← x1 − x0
        h1 ← x2 − x1
        delta0 ← (fx1 − fx0)/h0
        delta1 ← (fx2 − fx1)/h1
        a ← (delta1 − delta0)/(h1 − h0)
        b ← a ∗ h1 + delta1
        c ← fx2
        xi ← x2 + (−2 ∗ c)/(b + b/|b|) ∗ sqrt(b^2 − 4 ∗ a ∗ c)
        fxi ← f(xi)
        error ← tol + 1
        counter ← counter + 1
    end while
    if fxi = 0 then
        WRITE A root has been found it is xi
    else if error <= tol then
        WRITE One approach has been found and it is xi
    else
        WRITE The method fails with the macimum number of iterations given
    end if
end function
```

14: Tridiagonal

```
function Tridiagonal(A,B,C,D)
    N ← lenght of D
    C[0] ← C[0]/B[0]
    D[0] ← D[0]/B[0]
    for i = 1...N do
        aux ← B[i] − (A[i] ∗ C[i − 1])
        C[i] ← C[i]/aux
        D[i] ← (D[i] − A[i] ∗ D[i − 1])/aux
    end for
    x ←vector of lenght N
    for i = 0...N do
        x[i] ← 0
    end for
    x[n − 1] ← D[N − 1]
    for i = 0...N − 1 do
        x[i] ← D[i] − C[i] ∗ x[i + 1]
    end for
end function
```

15: Trisection

```
function Trisection(f, left, right, tol, nIter)
    fRight ← f(right)
    fLeft ← f(left)
    counter ← 0
    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
        xmid1 ← left + (right − left)/3
        xmid2 ← right − (right − left)/3
        fXmid1 ← f(xmid1)
        fXmid2 ← f(xmid2)
        counter ← 1
        error1 ← tol + 1
        error2 ← tol + 1
        while error1 > tol & error2 > tol & fXmid1 ≠ 0 & fXmid2 ≠ 0 & counter < nIter do
            if fLeft * fXmid1 < 0 then
                right ← xmid1
                fRight ← fXmid1
            else if fXmid1 * fXmid2 < 0 then
                left ← xmid1
                fLeft ← fXmid1
                right ← xmid2
                fRight ← fXmid2
            else
                left ← xmid2
                fLeft ← fXmid2
            end if
            xAux1 ← xmid1
            xAux2 ← xmid2
            xmid1 ← left + (right − left)/3
            fXmid1 ← f(xmid1)
            xmid2 ← right − (right − left)/3
            fXmid2 ← f(xmid2)
            error1 ← |xmid1 − xAux1)|
            error2 ← |xmid2 − xAux2)|
            counter ← counter + 1
        end while
        if fXmid1 = 0 then
            WRITE xmid1 is a root
        else if fXmid2 = 0 then
            WRITE xmid2 is a root
        else if error1 < tol then
            WRITE xmid1 is an approximation with tolerance tol
        else if error2 < tol then
            WRITE xmid2 is an approximation with tolerance tol
        else
            WRITE The method fails in nIter iterations
        end if
    end if
```

16: Jacobi

```
function Jacobi(A,b,x,iter,tol)
    if foundDeterminant(A) then
        WRITE The determinant is zero, the problem has no unique solution return
    end if
    d ← findDiagOfMatrix(A)
    p ← findUpperTriangular(A)
    o ← findLowerTriangular(A)
    l ← d − o
    u ← d − p
    T ← findInverseOfMatrix(d) ∗ (l + u)
    re ← foundSpectralRadius(T)
    if re > 1 then
        WRITE Spectral radius greater than 1: the method does not converge. return
        C ← finInverseOfMatrix(d) ∗ b
        i ← 0
        err ← tol + 1
        while err > tol AND i < iter do
            xi ← T ∗ x + C
            err ← findNormOfVector(xi − x)
            x ← xi
            i ← i + 1
        end while
        if i >= iter then
            WRITE The method fails with maximum number of iterations given return
        end if
        WRITE x is an approximation with tolerance tol
```

17: Vandermonde

```
function Vandermonde(x,y)
    n ← findVectorLenght(x)
    A ← generateMatrixOfOnes(n)
    for i = 1...n do
        for j = 1...n − 1 do
            A_ij ← x_i^(n−j)
        end for
    end for
end function=0
```

18: Crout

```
function Crout(A,B)
    lower ← matrixnxninitializedinceros
    uppper ← identitymatrixnxn
    for i = 0...n do
        for i=j...n do
            sum ← 0
            for k = 0...j do
                sum ← sum + lower[i][k] * upper[k][j]
            end for
            lower[i][j] ← A[i][j] − sum
        end for
        for i = j + 1...n do
            sum ← 0
            for k = 0...j do
                sum ← sum + lower[j][k] * upper[k][i]
            end for
            upper[j][i] ← (A[j][j] − sum)/lower[j][j]
        end for
    end for
    Z ← regressiveSubstitution(lower|B)
    X ← progressiveSubstitution(upper|Z)
end function
```

19: Doolittle

```
function Doolittle(A)
    n ←lenght of A
    lower ←matrix nxn initialized in ceros
    upper ←identity matrix nxn
    for i = 0...n do
        for k = i...n do
            sum ← 0
            for j = i...n do
                sum ← lower[i][k] * upper[j][k]
            end for
            upper[i][k] ← A[i][k] − sum
        end for
        for k = i...n do
            if i = k then
                lower[i][i] ← 1
            else
                sum ← 0
                for j = 0...i do
                    sum ← sum + lower[k][j] * upper[j][i]
                end for
                lower[k][i] ← (A[k][i] − sum)/upper[i][i]
            end if
        end for
    end for
    Z ← regressiveSubstitution(lower|B)
    X ← progressiveSubstitution(upper|Z)
end function
```

20: Seidel

```
function Seidel(A,b,x,iter,tol)
    if foundDeterminant(A) then
        WRITE The determinant is zero, the problem has no unique solution. return
    end if
    d ← findDiagOfMatrix(A)
    p ← findUpperTriangular(A)
    o ← findLowerTriangular(A)
    l ← d − o
    u ← d − p
    T ← findInverseOfMatrix(d − l) ∗ u
    re ← foundSpectralRadius(T)
    if re > 1 then
        WRITE Spectral radius greater than 1: the method does not converge. return
    end if
    C ← findInverseOfatrix(d − l) ∗ b
    i ← 0
    err ← tol + 1
    while err > tol AND i < iter do
        xi ← T ∗ x + C
        err ← findNormOfVector(xi − x)
        x ← xi
        i ← i + 1
    end while
    if i >= iter then
        WRITE The method fails with the maximum number of iterations given return
    end if
    WRITE x is an approximation with tolerance tol
end function
```

21: Heun

```
function Heun(f,x,y,h,n)
    counter ← 0
    while counter <= n do
        k1 ← f(x, y)
        k2 ← f(x + h, y + (k1 ∗ h))
        y ← y + ((k1 ∗ k2) ∗ (h/2))
        x ← x + h
        counter ← counter + 1
    end while
end function
```

22: Cholesky

```
function Cholesky(A,B)
    n ←lenght of A
    lower ←matrix nxn initialized in zeros
    for j = 0...n do
        for i = 0...j − 1 do
            lower[i][j] ← (A[i][j] − Σ_{k=0}^{i−1} lower[k][i] ∗ lower[k][j]/lower[i][i])
        end for
        lower[i][j] ← √(A[i][j] − Σ_{k=0}^{i−1} lower[k][j]^2)
    end for
    upper ← lower^T
    Z ← regressiveSubstitution(lower|B)
    X ← progressiveSubstitution(upper|Z)
end function
```

23: Lagrange

```
function Lagrange(x,y)
    yp ← 0
    p ← 0
    for i = 1...length(x) do
        p ← 1
        for j = 1...length(x) do
            if x! = j then
                p ← p * (xp − x[j])/(x[i] − x[j])
            end if
        end for
        yp ← yp + p * y[i]
    end for
end function
```

24: Simpson 1/3

```
function Simpson(a,b,f,n)
    deltaX ← (b − a)/n
    A ← 0
    for i = 0...n do
        xi ← a + i * deltaX
        fxi ← f(xi)
        if i > 0 AND i < n then
            if i module 2 == 0 then
                fxi ← 2 * fxi
            else
                fxi ← 4 * fxi
            end if
        end if
        A ← A + fxi
    end for
    A ← A * (deltaX/3)
    WRITE A is the result of the integral
end function
```

25: Simpson 3/8

```
function Simpson(a,b,f,n)
    deltaX ← (b − a)/n
    A ← 0
    for i = 0...n do
        xi ← a + i * deltaX
        fxi ← f(xi)
        if i > 0 AND i < n then
            if i module 2 == 0 then
                fxi ← 2 * fxi
            else
                fxi ← 3 * fxi
            end if
        end if
        A ← A + fxi
    end for
    A ← A * (3 * deltaX/8)
    WRITE A is the result of the integral
end function
```

26: SOR

```
function SOR(A,b,x,iter,tol)
    if foundDeterminant(A) then
        WRITE The determinant is zero, the problem has no unique solution. return
    end if
    d ← findDiagOfMatrix(A)
    p ← findUpperTriangular(A)
    o ← findLowerTriangular(A)
    l ← d − o
    u ← d − p
    T ← findInverseOfMatrix(d − w ∗ l) ∗ [(1 − w)d + w ∗ u]
    re ← foundSpectralRadius(T)
    if re > 1 then
        WRITE Spectral radius greater than 1: the method does not converge. return
    end if
    C ← w ∗ findInverseOfatrix(d − w ∗ l) ∗ b
    i ← 0
    err ← tol + 1
    while err > tol AND i < iter do
        xi ← T ∗ x + C
        err ← findNormOfVector(xi − x)
        x ← xi
        i ← i + 1
    end while
    if i >= iter then
        WRITE The method fails with the maximum number of iterations given return
    end if
    WRITE x is an approximation with tolerance tol
end function
```

27: Lineal Spline

```
function LinealSpline(x,y)
    zerosA ←create a square matrix of 0s x.lenght
    zerosB ←create matrix of 0s x.lenght by 1
    m ← 2 ∗ (n − 1)
    z ← 0
    for i = 1...length(x) do
        zerosA[i][z] ← x[i]
        zerosA[i][z + 1] ← 1
        z ← z + 2
        zerosB[i][z] ← y[i]
    end for
    zerosA[0][0] ← x[0]
    zerosA[0][1] ← 1
    zerosB[0][0] ← y[0]
    for i = 1...length(x) do
        zerosA[x.length][z] ← x[i]
        zerosA[x.length][z + 2] ← −x[i]
        zerosA[x.length][z + 1] ← 1
        zerosA[x.length][x + 3] ← −1
        zerosB[x.length][0] ← 0
        z ← z + 2
    end for
end function
```

28: Cuadratic Spline

```
function CuadraticSpline(x,y)
    zerosA ←create a square matrix of 0s x.lenght
    zerosB ←create matrix of 0s x.lenght by 1
    m ← 3 * (n − 1)
    z ← 0
    for i = 1...length(x) do
        zerosA[i][z] ← x[i]^2
        zerosA[i][z + 1] ← x[i]
        zerosA[i][z + 2] ← 1
        z ← z + 3
        zerosB[i][0] ← y[i]
    end for
    zerosA[0][0] ← x[0]^2
    zerosA[0][1] ← x[0]
    zerosA[0][2] ← 1
    zerosB[0][0] ← y[0]
    for i = 1...length(x) do
        zerosA[x.length][z] ← x[i]^2
        zerosA[x.length][z + 1] ← x[i]
        zerosA[x.length][z + 2] ← 1
        zerosA[x.length][z + 3] ← −x[i]^2
        zerosA[x.length][x + 4] ← −x[i]
        zerosA[x.length][x + 5] ← −1
        zerosB[x.length][0] ← 0
        z ← z + 3
    end for
    z ← 0
    for i = 2...length(x) − 1 do
        zerosA[2 * x.length − 3][z] ← x[i − 1]^2
        zerosA[2 * x.length − 3][z + 1] ← 1
        zerosA[2 * x.length − 3][z + 2] ← 0
        zerosA[2 * x.length − 3][z + 3] ← −(x[i − 1] * 2)
        zerosA[2 * x.length − 3][x + 4] ← −1
        zerosA[2 * x.length − 3][x + 5] ← 0
        z ← z + 3
    end for
    zerosA[m − 1][0] ← 2
    zerosB[m − 1][0] ← 0
end function
```

29: Cubic Spline

```
function CubicSpline(x,y)
    zerosA ←create a square matrix of 0s x.lenght
    zerosB ←create matrix of 0s x.lenght by 1
    m ← 3 ∗ (n − 1)
    z ← 0
    for i = 1...length(x) do
        zerosA[i][z] ← x[i]^3
        zerosA[i][z + 1] ← x[i]^2
        zerosA[i][z + 2] ← x[i]
        zerosA[i][z + 3] ← 1
        z ← z + 4
        zerosB[i][0] ← y[i]
    end for
    zerosA[0][0] ← x[0]^3
    zerosA[0][1] ← x[0]^2
    zerosA[0][2] ← x[0]
    zerosA[0][3] ← 1
    zerosB[0][0] ← y[0]
    z ← 0
    for i = 2...length(x) − 1 do
        zerosA[2 ∗ x.length − 2 + i][z] ← x[i − 1]^3
        zerosA[2 ∗ x.length − 2 + i][z + 1] ← x[i − 1]^2
        zerosA[2 ∗ x.length − 2 + i][z + 2] ← x[i − 1]
        zerosA[2 ∗ x.length − 2 + i][z + 3] ← 1
        zerosA[2 ∗ x.length − 2 + i][x + 4] ← −(x[i − 1]^3)
        zerosA[2 ∗ x.length − 2 + i][x + 5] ← −(x[i − 1]^2)
        zerosA[2 ∗ x.length − 2 + i][x + 6] ← −x[i − 1]
        zerosA[2 ∗ x.length − 2 + i][x + 7] ← −1
        z ← z + 4
        zerosB[x.length − 1 + i][0] ← 0
    end for
    z ← 0
    for i = 2...length(x) − 1 do
        zerosA[2 ∗ x.length − 4 + i][z] ← x[i − 1]^2 ∗ 3
        zerosA[2 ∗ x.length − 4 + i][z + 1] ← x[i − 1]^1 ∗ 2
        zerosA[2 ∗ x.length − 4 + i][z + 2] ← 1
        zerosA[2 ∗ x.length − 4 + i][z + 3] ← 0
        zerosA[2 ∗ x.length − 4 + i][x + 4] ← −(x[i − 1]^3 ∗ 3)
        zerosA[2 ∗ x.length − 4 + i][x + 5] ← −(x[i − 1] ∗ 2)
        zerosA[2 ∗ x.length − 4 + i][x + 6] ← −1
        zerosA[2 ∗ x.length − 4 + i][x + 7] ← 0
        z ← z + 4
        zerosB[2 ∗ x.length − 3 + i][0] ← 0
    end for
    z ← 0
    for i = 2...length(x) − 1 do
        zerosA[3 ∗ x.length − 6 + i][z] ← x[i − 1] ∗ 6
        zerosA[3 ∗ x.length − 6 + i][z + 1] ← 2
        zerosA[3 ∗ x.length − 6 + i][z + 2] ← 0
        zerosA[3 ∗ x.length − 6 + i][z + 3] ← 0
        zerosA[3 ∗ x.length − 6 + i][x + 4] ← −(x[i − 1] ∗ 6)
        zerosA[3 ∗ x.length − 6 + i][x + 5] ← −2
        zerosA[3 ∗ x.length − 6 + i][x + 6] ← 0
        zerosA[3 ∗ x.length − 6 + i][x + 7] ← 0
        z ← z + 4
        zerosB[2 ∗ x.length − 3 + i][0] ← 0
    end for
    zerosA[m − 2][0] ← x[0] ∗ 6
```

```
    zerosA[m − 2][0] ← 2
    zerosA[m − 1][0] ← x[x.length] ∗ 6
    zerosA[m − 1][0] ← 2
    zerosB[m − 1][0] ← 0
    zerosA[m − 2][0] ← 0
end function
```

30: Compound Trapeze

```
function CompoundTrapeze(a,b,f,n)
    deltaX ← (b − a)/n
    A ← 0
    for i = 0...n do
        xi ← a + i ∗ deltaX
        fxi ← f(xi)
        if i > 0 AND i < n then
            fxi ← 2 ∗ fxi
        end if
        A ← A + fxi
    end for
    A ← A ∗ (deltaX/2)
    WRITE A is the result of the integral
end function
```

31: Euler

```
function Euler(f,xi,yi,xf,h)
    n ← (xf − xi)/h
    for i = 0...n do
        y1 ← f(xi, yi)
        hyi ← g ∗ y1
        newarray.push([xi, y1])
        yi ← yi + hy1
        xi ← xi + h
    end for
end function
```

32: Simple LU

```
function SimpleLu(A,b)
    [U, L] ← foundMatrixUandL(A)
    z ← progressiveSubstitution(L, b)
    x ← backwardSubstitution(U, z)
    WRITE x is the vector solution
end function
```

33: Pivot LU

```
function PivotLu(A,b)
    [U, L, P] ← foundMatrixUandLandPWithPartialPivoting(A)
    Bn ← P ∗ b
    z ← progressiveSubstitution(L, Bn)
    x ← backwardSubstitution(U, z)
    WRITE x is the vector solution
end function
```

34: Divided Diferences

```
function DividedDiferences(x,y)
    zerosArr ←create a square matrix of 0s x.length
    w ← 0
    for i = 1...x.length do
        z ← i
        while y.length > z do
            aux1 ← (zerosArr[z][w] − zerosArr[z − 1][w])/(x[z] − x[z − i])
            zerosArr[z][i] ← aux1
            z ← z + 1
        end while
        w ← z + nxa1
    end for
end function
```

Results

1: Incremental Search

```
answer =

    "There is at least one root between -2.5 and -2"


matrix =

  3×6 string array

    "iteration"    "x0"      "x1"      "fx0"         "fx1"         "fx0*fx1"
    "1"            "-3"      "-2.5"    "-0.48028"    "-0.19386"    "0.093108"
    "2"            "-2.5"    "-2"      "-0.19386"    "0.10258"     "-0.019886"
```

2: Bisection

```
answer =

    "0.9364 is an approach with tolerance 1e-06"


A =

  21×6 string array

    "counter"    "left"         "right"        "xmid"         "fmid"            "error"
    "1"          "0"            "1"            "0.5"          "-0.29311"        "1"
    "2"          "0.5"          "1"            "0.75"         "-0.1184"         "0.25"
    "3"          "0.75"         "1"            "0.875"        "-0.036818"       "0.125"
    "4"          "0.875"        "1"            "0.9375"       "0.00063392"      "0.0625"
    "5"          "0.875"        "0.9375"       "0.90625"      "-0.017772"       "0.03125"
    "6"          "0.90625"      "0.9375"       "0.92188"      "-0.0084866"      "0.015625"
    "7"          "0.92188"      "0.9375"       "0.92969"      "-0.0039054"      "0.0078125"
    "8"          "0.92969"      "0.9375"       "0.93359"      "-0.0016304"      "0.0039062"
    "9"          "0.93359"      "0.9375"       "0.93555"      "-0.00049694"     "0.0019531"
    "10"         "0.93555"      "0.9375"       "0.93652"      "6.8822e-05"      "0.00097656"
    "11"         "0.935547"     "0.936523"     "0.936035"     "-0.000213974"    "0.000488281"
    "12"         "0.936035"     "0.936523"     "0.936279"     "-7.25548e-05"    "0.000244141"
    "13"         "0.936279"     "0.936523"     "0.936401"     "-1.86098e-06"    "0.00012207"
    "14"         "0.936401"     "0.936523"     "0.936462"     "3.3482e-05"      "6.10352e-05"
    "15"         "0.936401"     "0.936462"     "0.936432"     "1.58108e-05"     "3.05176e-05"
    "16"         "0.936401"     "0.936432"     "0.936417"     "6.97501e-06"     "1.52588e-05"
    "17"         "0.936401"     "0.936417"     "0.936409"     "2.55703e-06"     "7.62939e-06"
    "18"         "0.936401"     "0.936409"     "0.936405"     "3.48029e-07"     "3.8147e-06"
    "19"         "0.936401"     "0.936405"     "0.936403"     "-7.56477e-07"    "1.90735e-06"
    "20"         "0.936403"     "0.936405"     "0.936404"     "-2.04223e-07"    "9.53674e-07"
```

3: False Rule

```
0.936405 is an approximation to a root with a tolerance 1e-06


Iterations | Xi |  Xs | Xm | Ym | Error
   1.000000000000000                  0   1.000000000000000   0.933940380718216  -0.001429076703686   1.000001000000000
   2.000000000000000   0.933940380718216   1.000000000000000   0.936506051665625   0.000058756008358   0.002739620254291
   3.000000000000000   0.933940380718216   0.936506051665625   0.936404730742641   0.000000086782541   0.000108202062268
   4.000000000000000   0.933940380718216   0.936404730742641   0.936404581100869   0.000000000128154   0.000000159804614
```

4: Fixed Point

```
answer =

    "-0.37444 is an approximation to a root with a tolerance 1e-06"


matrix =

  27×4 string array

    "iteration"    "xn"           "f(xn)"          "error"
    "0"            "-0.5"         "0.20689"        "1"
    "1"            "-0.29311"     "-0.12671"       "0.20689"
    "2"            "-0.41982"     "0.073517"       "0.12671"
    "3"            "-0.3463"      "-0.044654"      "0.073517"
    "4"            "-0.39096"     "0.026553"       "0.044654"
    "5"            "-0.36441"     "-0.016021"      "0.026553"
    "6"            "-0.38043"     "0.0095895"      "0.016021"
    "7"            "-0.37084"     "-0.0057689"     "0.0095895"
    "8"            "-0.37661"     "0.0034602"      "0.0057689"
    "9"            "-0.37315"     "-0.0020792"     "0.0034602"
    "10"           "-0.37522"     "0.0012481"      "0.0020792"
    "11"           "-0.373977"    "-0.00074963"    "0.00124806"
    "12"           "-0.374726"    "0.000450082"    "0.00074963"
    "13"           "-0.374276"    "-0.000270295"   "0.000450082"
    "14"           "-0.374546"    "0.000162302"    "0.000270295"
    "15"           "-0.374384"    "-9.74644e-05"   "0.000162302"
    "16"           "-0.374482"    "5.85256e-05"    "9.74644e-05"
    "17"           "-0.374423"    "-3.51447e-05"   "5.85256e-05"
    "18"           "-0.374458"    "2.1104e-05"     "3.51447e-05"
    "19"           "-0.374437"    "-1.26729e-05"   "2.1104e-05"
    "20"           "-0.37445"     "7.60996e-06"    "1.26729e-05"
    "21"           "-0.374442"    "-4.56974e-06"   "7.60996e-06"
    "22"           "-0.374447"    "2.7441e-06"     "4.56974e-06"
    "23"           "-0.374444"    "-1.64781e-06"   "2.7441e-06"
    "24"           "-0.374446"    "9.89502e-07"    "1.64781e-06"
    "25"           "-0.374445"    "-5.9419e-07"    "9.89502e-07"
```

5: Newton

```
answer =

    "0.9364 is a root approximation with a tolerance of 1e-06"


matrix =

  6×5 string array

    "iteration"    "xn"         "f(xn)"         "f'(xn)"     "erro"
    "1"            "0.5"        "-0.29311"      "0.68421"    "1"
    "2"            "0.92839"    "-0.0046622"    "0.58461"    "0.42839"
    "3"            "0.93637"    "-2.1913e-05"   "0.57911"    "0.0079748"
    "4"            "0.9364"     "-4.9834e-10"   "0.57908"    "3.7839e-05"
    "5"            "0.9364"     "-1.1102e-16"   "0.57908"    "8.6057e-10"
```

6: Secant

```
0.936405 is an approximation to a root with a tolerance 1e-06

Iterations |  Xn |   y1  |  Error
                   0   1.000000000000000   0.035366079380240   1.000001000000000

   1.000000000000000   0.946166222306525   0.005619392737864   0.053833777693475
   2.000000000000000   0.935996580791173  -0.000236322174701   0.010169641515352
   3.000000000000000   0.936407002376704   0.000001402235891   0.000410421585531
   4.000000000000000   0.936404581473120   0.000000000343716   0.000002420903584
   5.000000000000000   0.936404580879561  -0.000000000000000   0.000000000593558
```

7: Multiple Roots

```
>> raicesMul(h,hder,hder2,1,tol,N)
    "counter"    "xi"              "fxi"            "error"
    "0"          "1"               "0.71828"        "1"
    "1"          "-0.23421"        "0.025406"       "1.2342"
    "2"          "-0.0084583"      "3.5671e-05"     "0.22575"
    "3"          "-1.189e-05"      "7.0688e-11"     "0.0084464"
    "4"          "-4.2186e-11"     "0"              "1.189e-05"

A root was found: -4.21859069894e-11

ans =

     -4.21859069893579e-11
```

8: Simple Gauss Elimination

```
Stage 1
                        2             -1              0                  3                 1
                        0              1              3                6.5               0.5
                        0             13             -2                 11                 1
                        0             12             -2                -18                -6

Stage 2
                        2             -1              0                  3                 1
                        0              1              3                6.5               0.5
                        0              0            -41              -73.5              -5.5
                        0              0            -38                -96               -12

Stage 3
                        2             -1              0                  3                 1
                        0              1              3                6.5               0.5
                        0              0            -41              -73.5              -5.5
                        0              0              0  -27.8780487804878  -6.90243902439024


ans =

        0.0384951881014872   -0.180227471566054   -0.309711286089239   0.247594050743657
```

9: Gauss Elimination with Partial Pivoting

```
Stage 0
                    2                  -1                   0                   3                   1
                    1                 0.5                   3                   8                   1
                    0                  13                  -2                  11                   1
                   14                   5                  -2                   3                   1

Stage 1
                   14                   5                  -2                   3                   1
                    0   0.142857142857143    3.14285714285714    7.78571428571429   0.928571428571429
                    0                  13                  -2                  11                   1
                    0   -1.71428571428571   0.285714285714286    2.57142857142857   0.857142857142857

Stage 2
                   14                   5                  -2                   3                   1
                    0                  13                  -2                  11                   1
                    0                   0    3.16483516483516    7.66483516483516   0.917582417582418
                    0 2.22044604925031e-16  0.021978021978022    4.02197802197802   0.989010989010989

Stage 3
                   14                   5                  -2                   3                   1
                    0                  13                  -2                  11                   1
                    0                   0    3.16483516483516    7.66483516483516   0.917582417582418
                    0 2.22044604925031e-16                  0             3.96875   0.982638888888889


ans =

   0.0384951881014873  -0.180227471566054  -0.309711286089239   0.247594050743657
```

10: Gauss Elimination with Total Pivoting

```
Stage 0
                    2                  -1                   0                   3                   1
                    1                 0.5                   3                   8                   1
                    0                  13                  -2                  11                   1
                   14                   5                  -2                   3                   1

Stage 1
                   14                   5                  -2                   3                   1
                    0   0.142857142857143    3.14285714285714    7.78571428571429   0.928571428571429
                    0                  13                  -2                  11                   1
                    0   -1.71428571428571   0.285714285714286    2.57142857142857   0.857142857142857

Stage 2
                   14                   5                  -2                   3                   1
                    0                  13                  -2                  11                   1
                    0                   0    3.16483516483516    7.66483516483516   0.917582417582418
                    0 2.22044604925031e-16  0.021978021978022    4.02197802197802   0.989010989010989

Stage 3
                   14                   5                   3                  -2                   1
                    0                  13                  11                  -2                   1
                    0                   0    7.66483516483516    3.16483516483516   0.917582417582418
                    0 2.22044604925031e-16                  0   -1.63870967741936    0.50752688172043


ans =

   0.0384951881014873  -0.180227471566054  -0.309711286089239   0.247594050743657
```

11: Trisection

```
answer =

    "-0.9364 is an approximation with tolerance 1e-06"


matrix =

  15×9 string array

    "iteration"    "left"         "right"        "xmid1"        "xmid2"        "f(xmid1)"       "f(xmid2)"       "error1"         "error2"
    "1"            "-1"           "0"            "-0.66667"     "-0.33333"     "-0.17619"       "-0.3983"        "1"              "1"
    "2"            "-1"           "-0.66667"     "-0.88889"     "-0.77778"     "-0.028277"      "-0.099628"      "0.22222"        "0.44444"
    "3"            "-1"           "-0.88889"     "-0.96296"     "-0.92593"     "0.01513"        "-0.0061059"     "0.074074"       "0.14815"
    "4"            "-0.96296"     "-0.92593"     "-0.95062"     "-0.93827"     "0.0081593"      "0.0010799"      "0.012346"       "0.012346"
    "5"            "-0.93827"     "-0.92593"     "-0.93416"     "-0.93004"     "-0.0013036"     "-0.003699"      "0.016461"       "0.0082305"
    "6"            "-0.93827"     "-0.93416"     "-0.9369"      "-0.93553"     "0.00028672"     "-0.00050781"    "0.0027435"      "0.005487"
    "7"            "-0.9369"      "-0.93553"     "-0.93644"     "-0.93599"     "2.2025e-05"     "-0.00024282"    "0.00045725"     "0.00045725"
    "8"            "-0.93644"     "-0.93599"     "-0.93629"     "-0.93614"     "-6.624e-05"     "-0.00015452"    "0.00015242"     "0.00015242"
    "9"            "-0.93644"     "-0.93629"     "-0.93639"     "-0.93634"     "-7.3953e-06"    "-3.6817e-05"    "0.00010161"     "0.00020322"
    "10"           "-0.93644"     "-0.93639"     "-0.93643"     "-0.93641"     "1.2218e-05"     "2.4115e-06"     "3.387e-05"      "6.774e-05"
    "11"           "-0.936409"    "-0.936392"    "-0.936403"    "-0.936397"    "-8.57402e-07"   "-4.12634e-06"   "2.25801e-05"    "1.12901e-05"
    "12"           "-0.936409"    "-0.936403"    "-0.936407"    "-0.936405"    "1.32188e-06"    "2.32238e-07"    "3.76335e-06"    "7.52671e-06"
    "13"           "-0.936405"    "-0.936403"    "-0.936404"    "-0.936404"    "-1.30975e-07"   "-4.94189e-07"   "2.5089e-06"     "1.25445e-06"
    "14"           "-0.936405"    "-0.936404"    "-0.936405"    "-0.936405"    "1.11167e-07"    "-9.90421e-09"   "4.1815e-07"     "8.36301e-07"
```

12: Steffensen

```
ans =

    "-0.37445 is an approximation with tolerance 1e-06"


matrix =

  7×3 string array

    "iteration"    "p"           "error"
    "0"            "1"           ""
    "1"            "-1.1614"     "2.1614"
    "2"            "-0.29639"    "0.86501"
    "3"            "-0.37339"    "0.076997"
    "4"            "-0.37444"    "0.0010573"
    "5"            "-0.37445"    "1.9539e-07"
```

13: Muller

```
    "Counter"    "xi"                       "Fxi"                         "Error"
    "0"          "-1.5043"                  "0.19094"                     "1"
    "1"          "-1.1709"                  "0.11434"                     "0.33342"
    "2+0i"       "-0.72832-0.64074i"        "-0.019849+0.53616i"          "0.77873+0i"
    "3+0i"       "-0.80786-0.054804i"       "-0.078939+0.036014i"         "0.59131+0i"
    "4+0i"       "-0.90184-0.046692i"       "-0.019714+0.028149i"         "0.09433+0i"
    "5+0i"       "-0.93066-0.0058347i"      "-0.0033277+0.003402i"        "0.049997+0i"
    "6+0i"       "-0.93694-0.00081382i"     "0.00031304+0.00047096i"      "0.0080461+0i"
    "7+0i"       "-0.93642-1.093e-06i"      "7.8828e-06+6.3292e-07i"      "0.00096849+0i"
    "8+0i"       "-0.9364+1.9213e-08i"      "-5.1544e-09-1.1126e-08i"     "1.3667e-05+0i"
    "9+0i"       "-0.9364-3.3272e-13i"      "6.6058e-14+1.9267e-13i"      "2.1175e-08+0i"

An approximation has been found and is: -0.93640458088-3.32717949702e-13i
```

14: Aitken

```
>> aitken(f, 1, tol, N)
    "counter"    "xi"            "fxi"           "error"
    "1"          "1"             "0.035366"      "2"
    "2"          "0.035366"      "-0.49875"      "0.96463"
    "3"          "-0.49875"      "-0.29396"      "0.53412"
    "4"          "-1.1614"       "0.11061"       "0.66265"
    "5"          "-1.3199"       "0.16184"       "0.83202"
    "6"          "-0.25797"      "-0.43694"      "0.21639"
    "7"          "-0.33039"      "-0.39994"      "0.0048809"
    "8"          "-0.36486"      "-0.38016"      "0.0060154"
    "9"          "-0.37176"      "-0.37605"      "0.00075755"
    "10"         "-0.37375"      "-0.37486"      "0.00027184"
    "11"         "-0.374259"     "-0.374557"     "6.50585e-05"
    "12"         "-0.374396"     "-0.374475"     "1.79039e-05"
    "13"         "-0.374432"     "-0.374453"     "4.68176e-06"
    "14"         "-0.374442"     "-0.374447"     "1.24684e-06"
    "15"         "-0.374444"     "-0.374446"     "3.29869e-07"

An approximation has been found and is: -0.374444108719
```

15: Crout

```
Stage  4 :

Matrix L:
+----------+----------+----------+----------+
|    x0    |    x1    |    x2    |    x3    |
+----------+----------+----------+----------+
| 4.00000  | 0.00000  | 0.00000  | 0.00000  |
| 1.00000  | 15.75000 | 0.00000  | 0.00000  |
| 0.00000  | -1.30000 | -3.75238 | 0.00000  |
| 14.00000 | 8.50000  | -3.61905 | 13.94924 |
+----------+----------+----------+----------+

Matrix U:
+---------+----------+---------+----------+
|    x0   |    x1    |    x2   |    x3    |
+---------+----------+---------+----------+
| 1.00000 | -0.25000 | 0.00000 | 0.75000  |
| 0.00000 | 1.00000  | 0.19048 | 0.46032  |
| 0.00000 | 0.00000  | 1.00000 | -0.45262 |
| 0.00000 | 0.00000  | 0.00000 | 1.00000  |
+---------+----------+---------+----------+

Answer:
+---------+---------+----------+----------+
|    x0   |    x1   |    x2    |    x3    |
+---------+---------+----------+----------+
| 0.52511 | 0.25546 | -0.41048 | -0.28166 |
+---------+---------+----------+----------+
```

16: Doolittle

```
Stage  4 :

Matrix L:
+---------+----------+---------+---------+
|    x0   |    x1    |    x2   |    x3   |
+---------+----------+---------+---------+
| 1.00000 | 0.00000  | 0.00000 | 0.00000 |
| 0.25000 | 1.00000  | 0.00000 | 0.00000 |
| 0.00000 | -0.08254 | 1.00000 | 0.00000 |
| 3.50000 | 0.53968  | 0.96447 | 1.00000 |
+---------+----------+---------+---------+

Matrix U:
+---------+----------+----------+----------+
|    x0   |    x1    |    x2    |    x3    |
+---------+----------+----------+----------+
| 4.00000 | -1.00000 | 0.00000  | 3.00000  |
| 0.00000 | 15.75000 | 3.00000  | 7.25000  |
| 0.00000 | 0.00000  | -3.75238 | 1.69841  |
| 0.00000 | 0.00000  | 0.00000  | 13.94924 |
+---------+----------+----------+----------+

Answer:
+---------+---------+----------+----------+
|    x0   |    x1   |    x2    |    x3    |
+---------+---------+----------+----------+
| 0.52511 | 0.25546 | -0.41048 | -0.28166 |
+---------+---------+----------+----------+
```

17: Cholesky

```
Stage  4 :

Matrix L:
+------------------+-------------------+------------------+------------------+
|        x0        |         x1        |        x2        |        x3        |
+------------------+-------------------+------------------+------------------+
| (2.0000 + 0.00i) |  (0.0000 + 0.00i) | (0.0000 + 0.00i) | (0.0000 + 0.00i) |
| (0.5000 + 0.00i) |  (3.9686 + 0.00i) | (0.0000 + 0.00i) | (0.0000 + 0.00i) |
| (0.0000 + 0.00i) | (-0.3276 + 0.00i) | (0.0000 + 1.94i) | (0.0000 + 0.00i) |
| (7.0000 + 0.00i) |  (2.1418 + 0.00i) | (0.0000 + 1.87i) | (3.7349 + 0.00i) |
+------------------+-------------------+------------------+------------------+

Matrix U:
+------------------+-------------------+------------------+-------------------+
|        x0        |         x1        |        x2        |         x3        |
+------------------+-------------------+------------------+-------------------+
| (2.0000 + 0.00i) | (-0.5000 + 0.00i) | (0.0000 + 0.00i) |  (1.5000 + 0.00i) |
| (0.0000 + 0.00i) |  (3.9686 + 0.00i) | (0.7559 + 0.00i) |  (1.8268 + 0.00i) |
| (0.0000 + 0.00i) |  (0.0000 + 0.00i) | (0.0000 + 1.94i) | (0.0000 + -0.88i) |
| (0.0000 + 0.00i) |  (0.0000 + 0.00i) | (0.0000 + 0.00i) |  (3.7349 + 0.00i) |
+------------------+-------------------+------------------+-------------------+

Answer:
+------------------+------------------+---------------------+--------------------+
|        x0        |        x1        |          x2         |         x3         |
+------------------+------------------+---------------------+--------------------+
| (0.5251 + 0.00i) | (0.2555 + 0.00i) | (-0.4105 + -0.00i) | (-0.2817 + 0.00i) |
+------------------+------------------+---------------------+--------------------+
```

18: LU Simple

```
Stage  3
L:
+------+---------------------+--------------------+-----+----+
|  L0  |          L1         |         L2         |  L3 |    |
+------+---------------------+--------------------+-----+----+
| 1.0  |         0.0         |        0.0         | 0.0 | L0 |
| 0.25 |         1.0         |        0.0         | 0.0 | L1 |
| 0.0  | -0.08253968253968254 |        1.0         | 0.0 | L2 |
| 3.5  |  0.5396825396825397  | 0.9644670050761421 | 0.0 | L3 |
+------+---------------------+--------------------+-----+----+
U:
+-----+-------+--------------------+--------------------+----+
|  U0 |   U1  |         U2         |         U3         |    |
+-----+-------+--------------------+--------------------+----+
| 4.0 |  -1.0 |        0.0         |        3.0         | U0 |
| 0.0 | 15.75 |        3.0         |        7.25        | U1 |
| 0.0 |  0.0  | -3.7523809523809524 | 1.6984126984126986 | U2 |
| 0.0 |  0.0  |        0.0         | 13.949238578680202 | U3 |
+-----+-------+--------------------+--------------------+----+

X:
+--------------------+--------------------+---------------------+----------------------+
|         x0         |         x1         |          x2         |          x3          |
+--------------------+--------------------+---------------------+----------------------+
| 0.5251091703056769 | 0.2554585152838428 | -0.4104803493449783 | -0.28165938864628826 |
+--------------------+--------------------+---------------------+----------------------+
```

19: LU Parcial

```
stage 4
L:
+--------------------+---------------------+---------------------+-----+----+
|         L0         |          L1         |          L2         |  L3 |    |
+--------------------+---------------------+---------------------+-----+----+
|        1.0         |          0          |          0          |  0  | L0 |
| 0.07142857142857142 |         1.0         |          0          |  0  | L1 |
|        0.0         | -0.0858490566037736  |         1.0         |  0  | L2 |
|  0.2857142857142857 | -0.16037735849056603 | -0.28831562974203334 | 1.0 | L3 |
+--------------------+---------------------+---------------------+-----+----+
U:
+------+-------------------+-------------------+-------------------+----+
|  U0  |         U1        |         U2        |         U3        |    |
+------+-------------------+-------------------+-------------------+----+
| 14.0 |        5.0        |        -2.0       |        30.0       | U0 |
|  0   | 15.142857142857142 | 3.142857142857143  | 5.857142857142858  | U1 |
|  0   |         0         | -3.730188679245283 | 1.6028301886792455 | U2 |
|  0   |         0         |         0         | -4.169954476479514 | U3 |
+------+-------------------+-------------------+-------------------+----+

X:
+--------------------+--------------------+---------------------+---------------------+
|         x0         |         x1         |          x2         |          x3         |
+--------------------+--------------------+---------------------+---------------------+
| 0.5251091703056769 | 0.25545851528384284 | -0.41048034934497823 | -0.28165938864628826 |
+--------------------+--------------------+---------------------+---------------------+
```

20: Vandermonde

```
Vandermonde matrix
+-----+------+------+------+
|  x0 |  x1  |  x2  |  x3  |
+-----+------+------+------+
| 1.0 | -1.0 | 1.0  | -1.0 |
| 1.0 | 0.0  | 0.0  | 0.0  |
| 1.0 | 3.0  | 9.0  | 27.0 |
| 1.0 | 4.0  | 16.0 | 64.0 |
+-----+------+------+------+

Polynomial coefficients:
+--------------------+--------------------+-------------------+--------------------+
|         x0         |         x1         |         x2        |         x3         |
+--------------------+--------------------+-------------------+--------------------+
| 2.9999999999999995 | -5.5333333333333336 | 5.825000000000001 | -1.1416666666666666 |
+--------------------+--------------------+-------------------+--------------------+

Vandermonde polynom
 (3.00000) - (5.53333x^1) + (5.82500x^2) - (1.14167x^3)
```

21: Divided Differences

```
Polynomial coefficients:
+------+-------+--------------------+---------------------+
|  x0  |   x1  |         x2         |          x3         |
+------+-------+--------------------+---------------------+
| 15.5 | -12.5 | 3.5416666666666665 | -1.1416666666666666 |
+------+-------+--------------------+---------------------+

Newton's Divided Difference Table
+---+------+------+--------------------+---------------------+---------------------+
| n |  xi  |  0º  |         1º         |          2º         |          3º         |
+---+------+------+--------------------+---------------------+---------------------+
| 0 | -1.0 | 15.5 |        0.0         |         0.0         |         0.0         |
| 1 | 0.0  | 3.0  |       -12.5        |         0.0         |         0.0         |
| 2 | 3.0  | 8.0  | 1.6666666666666667 |  3.5416666666666665 |         0.0         |
| 3 | 4.0  | 1.0  |        -7.0        | -2.1666666666666665 | -1.1416666666666666 |
+---+------+------+--------------------+---------------------+---------------------+

Newton's polynom
P(X) = 15.5 -12.5(x +1.0) + 3.5416666666666665(x +1.0)x -1.1416666666666666(x +1.0)x(x - 3.0)
```

22: SOR

```
|     45     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 1.4171445010240167e-06 |
|     46     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 1.0892751433509372e-06 |
|     47     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 8.372616104627896e-07  |
|     48     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 6.435538045166008e-07  |
|     49     | 0.52511 | 0.25546 | -0.41048 | -0.28166 |  4.94662081692539e-07  |
|     50     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 3.8021779405880007e-07 |
|     51     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 2.922511974041505e-07  |
|     52     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 2.2463643338285424e-07 |
|     53     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 1.7266492211138134e-07 |
|     54     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 1.3271745935937605e-07 |
|     55     | 0.52511 | 0.25546 | -0.41048 | -0.28166 | 1.0201217740946792e-07 |
+------------+---------+---------+----------+----------+------------------------+

X:
+--------------------+---------------------+---------------------+---------------------+
|         x0         |          x1         |          x2         |          x3         |
+--------------------+---------------------+---------------------+---------------------+
| 0.5251090135820877 | 0.25545842034112953 | -0.41048026945294747 | -0.2816592803943774 |
+--------------------+---------------------+---------------------+---------------------+
```

23: Seidel

```
| 20 |  0.5250913141268982 | 0.25544775542089165 | -0.41047195039685425 | -0.28164870252249136 | 1.662666e-05 |
| 21 |  0.5250984657470914 |  0.2554520648789292 | -0.41047531427933714 | -0.2816529824470867  | 9.967475e-06 |
| 22 |  0.5251027530550474 | 0.25545464834578435 | -0.4104773308853288  | -0.2816555482090081  | 5.975378e-06 |
| 23 |  0.5251053232432021 |  0.2554561971022483 | -0.4104785398157079  | -0.28165708635158293 | 3.582165e-06 |
| 24 |  0.5251068640392493 | 0.25545712556261535 | -0.4104792645545353  | -0.28165800844905464 | 2.147464e-06 |
| 25 |  0.5251077877274448 | 0.25545768216313536 | -0.41047969902650905 | -0.2816585612350974  | 1.287378e-06 |
| 26 |  0.5251083414671069 | 0.25545801583827094 | -0.4104799594870898  | -0.28165889262350097 | 7.717673e-07 |
| 27 |  0.5251086734271935 |  0.2554582158723925 | -0.4104801156299903  | -0.2816590912867551  | 4.626649e-07 |
| 28 |  0.5251088724331644 | 0.25545833579037724 | -0.41048020923573025 | -0.2816592103829217  | 2.773619e-07 |
| 29 |  0.5251089917347855 | 0.25545840767972766 | -0.41048026535121496 | -0.28165928177960226 | 1.662750e-07 |
| 30 |  0.5251090632546336 | 0.25545845077650514 | -0.4104802989917548  | -0.28165932458103016 | 9.967983e-08 |
+----+---------------------+---------------------+----------------------+----------------------+--------------+

X:
+--------------------+---------------------+---------------------+----------------------+
|         x0         |          x1         |          x2         |          x3          |
+--------------------+---------------------+---------------------+----------------------+
| 0.5251090632546336 | 0.25545845077650514 | -0.4104802989917548 | -0.28165932458103016 |
+--------------------+---------------------+---------------------+----------------------+
```

24: Jacobi

```
| 42 | 0.5251061096738538 | 0.2554566590121258 | -0.4104786523061102 | -0.28165693242790885 | 1.524697e-06 |
| 43 | 0.5251068640739631 | 0.25545711655920955 | -0.4104790705966158 | -0.2816575378368934 | 1.148893e-06 |
| 44 | 0.5251074325174725 | 0.2554574613168408 | -0.4104793857868888 | -0.2816579940341588 | 8.657059e-07 |
| 45 | 0.5251078608548293 | 0.255457721104288 | -0.4104796232873669 | -0.2816583377800866 | 6.523267e-07 |
| 46 | 0.525108183611137 | 0.25545791685470737 | -0.4104798022484174 | -0.2816585968021261 | 4.915377e-07 |
| 47 | 0.5251084268152715 | 0.25545806435813706 | -0.4104799370983646 | -0.281658791977543 | 3.703829e-07 |
| 48 | 0.5251086100726915 | 0.2554581755032365 | -0.4104800387102189 | -0.2816589390467072 | 2.790893e-07 |
| 49 | 0.5251087481608395 | 0.25545825925365306 | -0.4104801152763964 | -0.2816590498651434 | 2.102988e-07 |
| 50 | 0.5251088522122708 | 0.25545832236061267 | -0.41048017297035166 | -0.2816591333690937 | 1.584635e-07 |
| 51 | 0.5251089306169734 | 0.2554583699130022 | -0.4104802164436999 | -0.28165919629051855 | 1.194050e-07 |
| 52 | 0.5251089896961394 | 0.2554584057444048 | -0.4104802492016183 | -0.28165924370300127 | 8.997367e-08 |
+----+--------------------+--------------------+---------------------+----------------------+--------------+
T Matrix:
+----------------------+----------------------+---------------------+---------------------+----+
|          T0          |          T1          |         T2          |         T3          |    |
+----------------------+----------------------+---------------------+---------------------+----+
|         -0.0         |         0.25         |        -0.0         |        -0.75        | T0 |
| -0.06451612903225806 |         -0.0         | -0.1935483870967742 | -0.5161290322580645 | T1 |
|         -0.0         |        -0.325        |        -0.0         |        0.275        | T2 |
| -0.4666666666666667  | -0.16666666666666666 | 0.06666666666666667 |        -0.0         | T3 |
+----------------------+----------------------+---------------------+---------------------+----+
The spectral radius is:
0.7535169428701507

X:
+--------------------+--------------------+---------------------+----------------------+
|         x0         |         x1         |         x2          |          x3          |
+--------------------+--------------------+---------------------+----------------------+
| 0.5251089896961394 | 0.2554584057444048 | -0.4104802492016183 | -0.28165924370300127 |
+--------------------+--------------------+---------------------+----------------------+
```

25: Lagrange

```
 L0(x) = (x-0) * (x-3) * (x-4)/(-1-0) * (-1-3) * (-1-4) = -x**3/20 + 7*x**2/20 - 3*x/5

 L1(x) = (x--1) * (x-3) * (x-4)/(0--1) * (0-3) * (0-4) = x**3/12 - x**2/2 + 5*x/12 + 1

 L2(x) = (x--1) * (x-0) * (x-4)/(3--1) * (3-0) * (3-4) = -x**3/12 + x**2/4 + x/3

 L3(x) = (x--1) * (x-0) * (x-3)/(4--1) * (4-0) * (4-3) = x**3/20 - x**2/10 - 3*x/20

Lagrange´s polynom
15.5*L0 + 3*L1 + 8*L2 + 1*L3
```

26: Heun

```
▶ python3 heun.py
Approximate solution at x =  0.1  is  20.28942
```

27: Euler

```
▶ python3 euler.py
Approximate solution at x =  0.1  is  1.11167
```

28: Compound Trapeze

```
▶ python3 CompoundTrapeze.py
-0.998687096792685 is the result of the integral
```

29: Simpson 1/3

```
▶ python3 simpson1-3.py
1.82785 is the result of the integral
```

30: Simpson 3/8

```
▶ python3 simpson3-8.py
13.51055 is the result of the integral
```

31: Linear Spline

```
polynoms by segments:
 x = [-1,0]
3.0 - 12.5*x
 x = [0,3]
1.66666666666667*x + 3.0
 x = [3,4]
29.0 - 7.0*x
```

32: Cuadratic Spline

```
 x = [-1,0]
 1.7763568394002505e-14x^2 + -12.499999999999982x + 3.0
 x = [0,3]
 4.7222222222222216x^2 + -12.499999999999982x + 3.0
 x = [3,4]
 -22.833333333333332x^2 + 152.83333333333326x + -244.99999999999986
```

33: Cubic Spline

```
polynoms by segment:
 x = [-1,0]
2.533333333333333x^3 + 7.6x^2 - 7.433333333333333x + 3.0
 x = [0,3]
-1.5222222222222222x^3 + 7.6x^2 - 7.433333333333333x + 3.0
 x = [3,4]
2.033333333333333x^3 - 24.4x^2 + 88.5666666666667x - 93.0
```

Members signatures

1: Jacobo Rave

Jacobo Rave Londoño

2: David Echeverri

3: Kevin Sossa

Kevin Sossa

4: Sebastián Guerra