# EAFIT UNIVERSITY

## DEPARTMENT OF INFORMATICS AND SYSTEMS

PROJECT CHOICE

Second Report

September 28, 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**

```
1:  function IncrementalSearch(f,x0,delta,niter)
2:      fx0 ← f(x0)
3:      if fx0 = 0 then
4:          WRITE x0 is a root
5:      else
6:          x1 ← x0 + delta
7:          counter ← 1
8:          fx1 ← f(x1)
9:          while fx0 * fx1 > 0 AND counter < niter do
10:             x0 ← x1
11:             fx0 ← fx1
12:             x1 ← x0 + delta
13:             fx1 ← f(x1)
14:             counter ← counter + 1
15:         end while
16:         if fx1 = 0 then
17:             WRITE x1 is a root
18:         else if fx0 * fx1 < 0 then
19:             WRITE There is at least one root between x0 and x1
20:         else
21:             WRITE The method fails in niter iterations
22:         end if
23:     end if
24: end function
```

**2: Bisection**

```
1: function Bisection(f,left,right,tol,niter)
2:     fRight ← f(right)
3:     fLeft ← f(left)
4:     if fRight = 0 then
5:         WRITE right is a root
6:     else if fLeft = 0 then
7:         WRITE left is a root
8:     else if fLeft * fRight < 0 then
9:         mid ← (left + right)/2
10:        fmid ← f(mid)
11:        counter ← 1
12:        error ← tol + 1
13:        while error > tol AND fmid ≠ 0 AND counter < niter do
14:            if fLeft * fmid < 0 then
15:                right ← mid
16:                fRight ← fmid
17:            else
18:                left ← mid
19:                fLeft ← fmid
20:            end if
21:            aux ← mid
22:            mid ← (right + left)/2
23:            fmid ← f(mid)
24:            error ← |mid − aux|
25:            counter ← counter + 1
26:        end while
27:        if fmid = 0 then
28:            WRITE mid is a root
29:        else if error < tol then
30:            WRITE mid is an approximation with tolerance = tol
31:        else
32:            WRITE The method failed at niter iterations
33:        end if
34:    else
35:        WRITE Inappropriate interval
36:    end if
37: end function
```

**3: False Rule**

```
1: function falseRule(f, Xi, Xs, Tol, Iter)
2:     Yi ← f(Xi)
3:     Ys ← f(Xs)
4:     if Yi = 0 then
5:         WRITE Xi is a root
6:     else
7:         if Ys = 0 then
8:             WRITE Xs is a root
9:         else
10:            if Yi * Ys < 0 then
11:                Xm ← (Xi) − ((f(Xi) * (Xi − Xs))/(f(Xi) − f(Xs)))
12:                Ym ← f(Xm)
13:                Error ← Tol + 1
14:                Cont ← 1
15:                while Ym ≠ 0 AND Error > Tol AND Cont < Iter do
16:                    if Yi * Ym < 0 then
17:                        Xs ← Xm
18:                        Ys ← Ym
19:                    else
20:                        Xi ← Xm
21:                        Yi ← Ym
22:                    end if
23:                    Xaux ← Xm
24:                    Xm ← (Xi) − ((f(Xi) * (Xi − Xs))/(f(Xi) − f(Xs)))
25:                    Ym ← f(Xm)
26:                    Error ← |Xm − Xaux|/Xm
27:                    Cont ← Cont + 1
28:                end while
29:                if Ym = 0 then
30:                    WRITE Xm is a root of f
31:                else
32:                    if Error < Tol then
33:                        WRITE Xm is an approximation to a root with a tolerance = Tol
34:                    else
35:                        WRITE failure in Iter iterations
36:                    end if
37:                end if
38:            else
39:                WRITE The interval is inadequate
40:            end if
41:        end if
42:    end if
43: end function
```

**4: Fixed Point**

```
1: function Fixed point(f, g, tol, x0, niter)                ▷ g is the convergent form of f
2:     fx ← f(x0)
3:     counter ← 0
4:     error ← tol + 1
5:     while fx ≠ 0 AND error > tol AND counter < niter do
6:         x1 ← g(x0)
7:         fx ← f(x1)
8:         error ← |x1 − x0|
9:         x0 ← x1
10:        counter ← counter + 1
11:    end while
12:    if fx = 0 then
13:        WRITE x1 is a root
14:    else if error < tol then
15:        WRITE x1 is a root approximation with tolerance = tol
16:    else
17:        WRITE The method failed at niter iteration
18:    end if
19: end function
```

**5: Newton**

```
1: function Newton(f, fder, tol, x0, niter)                  ▷ fder is the first derivative of f
2:     fx ← f(x0)
3:     dfx ← fder(x0)
4:     counter ← 0
5:     error ← tol + 1
6:     while error > tol AND fx ≠ 0 AND dfx ≠ 0 AND counter < niter do
7:         x1 ← x0 − (fx/dfx)
8:         fx ← f(x1)
9:         dfx ← fder(x1)
10:        error ← |x1 − x0|
11:        x0 ← x1
12:        counter ← counter + 1
13:    end while
14:    if fx = 0 then
15:        WRITE x0 is a root of f
16:    else if error < tolerance then
17:        WRITE x1 is a root approximation with tolerance tol
18:    else if dfx = 0 then
19:        WRITE x1 is a possible multiple root
20:    else
21:        WRITE The method failed at niter iteration
22:    end if
23: end function
```

**6: Secant**

```
1: function Secant(x0, xl, tol, iter, f)
2:     y0 ← f(x0)
3:     if y0 = 0 then
4:         WRITE x0 is a root of f
5:     else
6:         y1 ← f(x1)
7:         d ← y1 − y0
8:         error ← tol + 1
9:         cont ← 0
10:        while y1 ≠ 0 AND error > tol AND cont < iter AND d ≠ 0 do
11:            x2 ← x1 − ((y1 ∗ (x1 − x0))/(d))
12:            error ← |x2 − x1|
13:            x0 ← x1
14:            y0 ← y1
15:            y1 ← f(x2)
16:            x1 ← x2
17:            d ← y1 − y0
18:            counter ← counter + 1
19:        end while
20:        if y1 = 0 then
21:            WRITE x1 is a root of f
22:        else
23:            if error < tol then
24:                WRITE x1 is an approximation to a root with a tolerance = tol
25:            else
26:                if d = 0 then
27:                    WRITE There is a multiple root
28:                else
29:                    WRITE failure in iter iterations
30:                end if
31:            end if
32:        end if
33:    end if
34: end function
```

7: Multiple Root

```
1: function MultipleRoot(f,fl,f2,x0,tolerance,nMax)
2:     xi ← x0
3:     fxi ← f(xi)
4:     if fxi = 0 then
5:         WRITE A root was found: xi
6:     else
7:         counter ← 0
8:         f1xi ← f1(xi)
9:         f2xi ← f2(xi)
10:        error ← tolerance + 1
11:        det ← (f1xi^2) − (fxi ∗ f2xi)
12:        while fxi ≠ 0 AND error > tolerance AND counter < nMax AND det ≠ 0 do
13:            xiAux ← xi
14:            x1 ← x1 − ((fxi ∗ f1xi)/((f1xi^2) − (fxi ∗ f2xi)))
15:            fxi ← f(xi)
16:            f1xi ← f1(xi)
17:            f2xi ← f2(xi)
18:            error ← |xi − xiAux|
19:            det ← (f1xi^2) − (fxi ∗ f2xi)
20:            counter ← counter + 1
21:        end while
22:        if fx1 = 0 then
23:            WRITE A root was found: xi
24:        else if error ≤ tolerance then
25:            WRITE One approach is: xi
26:        else if det = 0 then
27:            WRITE Method failure
28:        else
29:            WRITE The method fails with the maximum number of iterations given
30:        end if
31:    end if
32:    x ← xi
33: 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,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 k ← 1, n + 1 do
            Aj, k ← Aj, k − ratio ∗ Ai, k
        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**

```
1: function Aitken(f,x0,tol,n)
2:     x1 ← f(x0)
3:     x2 ← f(x1)
4:     fxi ← f(x2)
5:     det ← (x2 − x1) − (x1 − x0)
6:     counter ← 4
7:     while fxi ≠ 0 AND error > tol AND counter < n AND det ≠ 0 do
8:         xi ← ((x2 − x1)^2)/det
9:         fxi ← f(xi)
10:        det ← (x2 − x2) − (x1 − x0)
11:        error ← absoluteValue(xi − x2)
12:        x0 ← xi
13:        x1 ← f(x0)
14:        x2 ← f(x1)
15:        counter ← counter + 1
16:    end while
17:    if fxi = 0 then
18:        WRITE "xi is a root"
19:    else if error <= tol then
20:        WRITE "xi is an approximation"
21:    else if det = 0 then
22:        WRITE "Error during method execution"
23:    else
24:        WRITE "The method fails with the maximum number of iterations"
25:    end if
26: end function
```

**13: Muller**

```
1:  function Muller(f, x0, tol, nMax)
2:      fx0 ← f(x0)
3:      fx1 ← f(x1)
4:      x1 ← (x0 + x1)/2
5:      fx2 ← f(x2)
6:      h0 ← x1 − x0
7:      h1 ← x2 − x1
8:      delta0 ← (fx1 − fx0)/h0
9:      delta1 ← (fx2 − fx1)/h1
10:     a ← (delta1 − delta0)/(h1 − h0)
11:     b ← a ∗ h1 + delta1
12:     c ← fx2
13:     xi ← x2 + (−2 ∗ c)/(b + (b/|b|) ∗ sqrt(b^2 − 4 ∗ a ∗ c)
14:     fxi ← f(xi)
15:     error ← tol + 1
16:     counter ← 0
17:     while fx1 ≠ 0 & error > tol & counter < nMax do
18:         x2Aux ← x2
19:         x1Aux ← x1
20:         x2 ← x1
21:         x1 ← x2Aux
22:         x0 ← x1Aux
23:         fx0 ← f(x0)
24:         fx1 ← f(x1)
25:         fx2 ← f(x2)
26:         h0 ← x1 − x0
27:         h1 ← x2 − x1
28:         delta0 ← (fx1 − fx0)/h0
29:         delta1 ← (fx2 − fx1)/h1
30:         a ← (delta1 − delta0)/(h1 − h0)
31:         b ← a ∗ h1 + delta1
32:         c ← fx2
33:         xi ← x2 + (−2 ∗ c)/(b + b/|b|) ∗ sqrt(b^2 − 4 ∗ a ∗ c)
34:         fxi ← f(xi)
35:         error ← tol + 1
36:         counter ← counter + 1
37:     end while
38:     if fxi = 0 then
39:         WRITE A root has been found it is xi
40:     else if error <= tol then
41:         WRITE One approach has been found and it is xi
42:     else
43:         WRITE The method fails with the macimum number of iterations given
44:     end if
45: end function
```

**14: Tridiagonal**

```
1:  function Tridiagonal(A,B,C,D)
2:      N ← lenght of D
3:      C[0] ← C[0]/B[0]
4:      D[0] ← D[0]/B[0]
5:      for i = 1...N do
6:          aux ← B[i] − (A[i] ∗ C[i − 1])
7:          C[i] ← C[i]/aux
8:          D[i] ← (D[i] − A[i] ∗ D[i − 1])/aux
9:      end for
10:     x ←vector of lenght N
11:     for i = 0...N do
12:         x[i] ← 0
13:     end for
14:     x[n − 1] ← D[N − 1]
15:     for i = 0...N − 1 do
16:         x[i] ← D[i] − C[i] ∗ x[i + 1]
17:     end for
18: end function
```

**15: Trisection**

```
1:  function Trisection(f, left, right, tol, nIter)
2:      fRight ← f(right)
3:      fLeft ← f(left)
4:      counter ← 0
5:      if fRight = 0 then
6:          WRITE right is a root
7:      else if fLeft = 0 then
8:          WRITE left is a root
9:      else if fLeft * fRight < 0 then
10:         xmid1 ← left + (right − left)/3
11:         xmid2 ← right − (right − left)/3
12:         fXmid1 ← f(xmid1)
13:         fXmid2 ← f(xmid2)
14:         counter ← 1
15:         error1 ← tol + 1
16:         error2 ← tol + 1
17:         while error1 > tol & error2 > tol & fXmid1 ≠ 0 & fXmid2 ≠ 0 & counter < nIter do
18:             if fLeft * fXmid1 < 0 then
19:                 right ← xmid1
20:                 fRight ← fXmid1
21:             else if fXmid1 * fXmid2 < 0 then
22:                 left ← xmid1
23:                 fLeft ← fXmid1
24:                 right ← xmid2
25:                 fRight ← fXmid2
26:             else
27:                 left ← xmid2
28:                 fLeft ← fXmid2
29:             end if
30:             xAux1 ← xmid1
31:             xAux2 ← xmid2
32:             xmid1 ← left + (right − left)/3
33:             fXmid1 ← f(xmid1)
34:             xmid2 ← right − (right − left)/3
35:             fXmid2 ← f(xmid2)
36:             error1 ← |xmid1 − xAux1)|
37:             error2 ← |xmid2 − xAux2)|
38:             counter ← counter + 1
39:         end while
40:         if fXmid1 = 0 then
41:             WRITE xmid1 is a root
42:         else if fXmid2 = 0 then
43:             WRITE xmid2 is a root
44:         else if error1 < tol then
45:             WRITE xmid1 is an approximation with tolerance tol
46:         else if error2 < tol then
47:             WRITE xmid2 is an approximation with tolerance tol
48:         else
49:             WRITE The method fails in nIter iterations
50:         end if
51:     end if
```

# 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
```

## Members signatures

1: Jacobo Rave

2: David Echeverri

3: Kevin Sossa

4: Sebastián Guerra