


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

```
1: function IncrementalSearch(f,x0,delta,niter)
2:    $fx0 \leftarrow f(x0)$ 
3:   if  $fx0 = 0$  then
4:     WRITE x0 is a root
5:   else
6:      $x1 \leftarrow x0 + \text{delta}$ 
7:      $counter \leftarrow 1$ 
8:      $fx1 \leftarrow f(x1)$ 
9:     while  $fx0 * fx1 > 0$  AND  $counter < niter$  do
10:       $x0 \leftarrow x1$ 
11:       $fx0 \leftarrow fx1$ 
12:       $x1 \leftarrow x0 + \text{delta}$ 
13:       $fx1 \leftarrow f(x1)$ 
14:       $counter \leftarrow counter + 1$ 
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:    $fRight \leftarrow f(right)$ 
3:    $fLeft \leftarrow f(left)$ 
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:      $mid \leftarrow (left + right)/2$ 
10:     $fmid \leftarrow f(mid)$ 
11:     $counter \leftarrow 1$ 
12:     $error \leftarrow tol + 1$ 
13:    while  $error > tol$  AND  $fmid \neq 0$  AND  $counter < niter$  do
14:      if  $fLeft * fmid < 0$  then
15:         $right \leftarrow mid$ 
16:         $fRight \leftarrow fmid$ 
17:      else
18:         $left \leftarrow mid$ 
19:         $fLeft \leftarrow fmid$ 
20:      end if
21:       $aux \leftarrow mid$ 
22:       $mid \leftarrow (right + left)/2$ 
23:       $fmid \leftarrow f(mid)$ 
24:       $error \leftarrow |mid - aux|$ 
25:       $counter \leftarrow counter + 1$ 
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:    $Y_i \leftarrow f(X_i)$ 
3:    $Y_s \leftarrow f(X_s)$ 
4:   if  $Y_i = 0$  then
5:     WRITE Xi is a root
6:   else
7:     if  $Y_s = 0$  then
8:       WRITE Xs is a root
9:     else
10:      if  $Y_i * Y_s < 0$  then
11:         $X_m \leftarrow (X_i) - ((f(X_i) * (X_i - X_s)) / (f(X_i) - f(X_s)))$ 
12:         $Y_m \leftarrow f(X_m)$ 
13:         $Error \leftarrow Tol + 1$ 
14:         $Cont \leftarrow 1$ 
15:        while  $Y_m \neq 0$  AND  $Error > Tol$  AND  $Cont < Iter$  do
16:          if  $Y_i * Y_m < 0$  then
17:             $X_s \leftarrow X_m$ 
18:             $Y_s \leftarrow Y_m$ 
19:          else
20:             $X_i \leftarrow X_m$ 
21:             $Y_i \leftarrow Y_m$ 
22:          end if
23:           $X_{aux} \leftarrow X_m$ 
24:           $X_m \leftarrow (X_i) - ((f(X_i) * (X_i - X_s)) / (f(X_i) - f(X_s)))$ 
25:           $Y_m \leftarrow f(X_m)$ 
26:           $Error \leftarrow |X_m - X_{aux}| / X_m$ 
27:           $Cont \leftarrow Cont + 1$ 
28:        end while
29:        if  $Y_m = 0$  then
30:          WRITE  $X_m$  is a root of f
31:        else
32:          if  $Error < Tol$  then
33:            WRITE  $X_m$  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:    $fx \leftarrow f(x_0)$ 
3:    $counter \leftarrow 0$ 
4:    $error \leftarrow tol + 1$ 
5:   while  $fx \neq 0$  AND  $error > tol$  AND  $counter < niter$  do
6:      $x1 \leftarrow g(x_0)$ 
7:      $fx \leftarrow f(x1)$ 
8:      $error \leftarrow |x1 - x0|$ 
9:      $x0 \leftarrow x1$ 
10:     $counter \leftarrow counter + 1$ 
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:    $fx \leftarrow f(x_0)$ 
3:    $dfx \leftarrow fder(x_0)$ 
4:    $counter \leftarrow 0$ 
5:    $error \leftarrow tol + 1$ 
6:   while  $error > tol$  AND  $fx \neq 0$  AND  $dfx \neq 0$  AND  $counter < niter$  do
7:      $x1 \leftarrow x0 - (fx/dfx)$ 
8:      $fx \leftarrow f(x1)$ 
9:      $dfx \leftarrow fder(x1)$ 
10:     $error \leftarrow |x1 - x0|$ 
11:     $x0 \leftarrow x1$ 
12:     $counter \leftarrow counter + 1$ 
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, x1, tol, iter, f)
2:    $y0 \leftarrow f(x0)$ 
3:   if  $y0 = 0$  then
4:     WRITE x0 is a root of f
5:   else
6:      $y1 \leftarrow f(x1)$ 
7:      $d \leftarrow y1 - y0$ 
8:      $error \leftarrow tol + 1$ 
9:      $cont \leftarrow 0$ 
10:    while  $y1 \neq 0$  AND  $error > tol$  AND  $cont < iter$  AND  $d \neq 0$  do
11:       $x2 \leftarrow x1 - ((y1 * (x1 - x0)) / (d))$ 
12:       $error \leftarrow |x2 - x1|$ 
13:       $x0 \leftarrow x1$ 
14:       $y0 \leftarrow y1$ 
15:       $y1 \leftarrow f(x2)$ 
16:       $x1 \leftarrow x2$ 
17:       $d \leftarrow y1 - y0$ 
18:       $counter \leftarrow counter + 1$ 
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,f1,f2,x0,tolerance,nMax)
2:    $xi \leftarrow x0$ 
3:    $f_{xi} \leftarrow f(xi)$ 
4:   if  $f_{xi} = 0$  then
5:     WRITE A root was found:  $xi$ 
6:   else
7:     counter  $\leftarrow 0$ 
8:      $f1_{xi} \leftarrow f1(xi)$ 
9:      $f2_{xi} \leftarrow f2(xi)$ 
10:    error  $\leftarrow tolerance + 1$ 
11:     $det \leftarrow (f1_{xi}^2) - (f_{xi} * f2_{xi})$ 
12:    while  $f_{xi} \neq 0$  AND error  $> tolerance$  AND counter  $< nMax$  AND  $det \neq 0$  do
13:       $xi_{Aux} \leftarrow xi$ 
14:       $x1 \leftarrow x1 - ((f_{xi} * f1_{xi}) / ((f1_{xi}^2) - (f_{xi} * f2_{xi})))$ 
15:       $f_{xi} \leftarrow f(xi)$ 
16:       $f1_{xi} \leftarrow f1(xi)$ 
17:       $f2_{xi} \leftarrow f2(xi)$ 
18:      error  $\leftarrow |xi - xi_{Aux}|$ 
19:       $det \leftarrow (f1_{xi}^2) - (f_{xi} * f2_{xi})$ 
20:      counter  $\leftarrow counter + 1$ 
21:    end while
22:    if  $f_{x1} = 0$  then
23:      WRITE A root was found:  $xi$ 
24:    else if error  $\leq tolerance$  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:   $x \leftarrow xi$ 
33: end function

```

8: Simple Gauss Elimination

```

1: function GaussSimple(A,b,n)
2:    $A \leftarrow \text{Concat}(A, b)$ 
3:   for  $i \leftarrow 1, n-1$  do
4:     if  $A_{i,i} = 0$  then
5:       WRITE Mathematical Error! Stop
6:     end if
7:     for  $j \leftarrow i+1, n$  do
8:       if  $A_{j,i} \neq 0$  then
9:          $A_j \leftarrow A_j - (A_{ji}/A_{ii}) * A_i$ 
10:      end if
11:    end for
12:    display(A)
13:  end for
14:   $X_n \leftarrow A_{n,n+1}/A_{n,n}$ 
15:  for  $i \leftarrow n-1, 1, \text{step} = -1$  do
16:     $X_i \leftarrow A_{i,n+1}$ 
17:    for  $j \leftarrow i+1, n$  do
18:       $X_i \leftarrow X_i - A_{i,j} * X_j$ 
19:    end for
20:     $X_i \leftarrow X_i/A_{i,i}$ 
21:  end for
22:  WRITE "Answer vector"
23:  for  $i \leftarrow 1, n$  do
24:    WRITE  $X_i$ 
25:  end for
26: end function

```

9: Gauss Elimination with Partial Pivoting

```

1: function GaussPartial(A,b,n)
2:    $A \leftarrow \text{Concat}(A, b)$ 
3:   for  $i \leftarrow 1, n-1$  do
4:     WRITE changeRows(A,i)
5:     if  $A_{i,i} = 0$  then
6:       WRITE Mathematical Error! Stop
7:     end if
8:     for  $j \leftarrow i+1, n$  do
9:       if  $A_{j,i} \neq 0$  then
10:         $A_j \leftarrow A_j - (A_{ji}/A_{ii}) * A_i$ 
11:      end if
12:    end for
13:  end for
14:   $X_n \leftarrow A_{n,n+1}/A_{n,n}$ 
15:  for  $i \leftarrow n-1, 1, \text{step} = -1$  do
16:     $X_i \leftarrow A_{i,n+1}$ 
17:    for  $j \leftarrow i+1, n$  do
18:       $X_i \leftarrow X_i - A_{i,j} * X_j$ 
19:    end for
20:     $X_i \leftarrow X_i/A_{i,i}$ 
21:  end for
22:  WRITE "Answer vector"
23:  for  $i \leftarrow 1, n$  do
24:    WRITE  $X_i$ 
25:  end for
26: end function

```


10: Gauss Elimination with Total Pivoting

```

1: function GaussTotal(A,b,n,delta,niter)
2:    $A \leftarrow \text{Concat}(A,b)$ 
3:   for  $i \leftarrow 1, n-1$  do
4:     WRITE changeRowsAndColumns(A,i)
5:     if  $A_{i,i} = 0$  then
6:       WRITE Mathematical Error! Stop
7:     end if
8:     for  $j \leftarrow i+1, n$  do
9:        $ratio \leftarrow A_{j,i}/A_{i,i}$ 
10:      for  $k \leftarrow 1, n+1$  do
11:         $A_{j,k} \leftarrow A_{j,k} - ratio * A_{i,k}$ 
12:      end for
13:    end for
14:  end for
15:   $X_n \leftarrow A_{n,n+1}/A_{n,n}$ 
16:  for  $i \leftarrow n-1, 1, step = -1$  do
17:     $X_i \leftarrow A_{i,n+1}$ 
18:    for  $j \leftarrow i+1, n$  do
19:       $X_i \leftarrow X_i - A_{i,j} * X_j$ 
20:    end for
21:     $X_i \leftarrow X_i/A_{i,i}$ 
22:  end for
23:  WRITE "Answer vector"
24:  for  $i \leftarrow 1, n$  do
25:    WRITE  $X_i$ 
26:  end for
27: end function

```

11: Steffensen

```

1: function Steffensen(f,x0,tol,n)
2:   for  $i = 1 \dots n$  do
3:      $x1 \leftarrow f(x0)$ 
4:      $x2 \leftarrow f(x1)$ 
5:      $p \leftarrow p0 - (p1 - p0)^2 / (p2 - 2 * p1 + p0)$ 
6:      $error \leftarrow \text{absoluteValue}(p - p0)$ 
7:      $p0 \leftarrow p$ 
8:      $count \leftarrow count + 1$ 
9:      $p0 \leftarrow p$ 
10:  end for
11:  if  $f(p0) = 0$  then
12:    WRITE p0 is a root
13:  else if  $error < tol$  then
14:    WRITE p0 is an approximation with tolerance = tol
15:  else
16:    WRITE failed to converge
17:  end if
18: end function

```

12: Aitken

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

```

14: Tridiagonal

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

15: Trisection

```

1: function Trisection(f, left, right, tol, nIter)
2:    $fRight \leftarrow f(right)$ 
3:    $fLeft \leftarrow f(left)$ 
4:    $counter \leftarrow 0$ 
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:     $xmid1 \leftarrow left + (right - left)/3$ 
11:     $xmid2 \leftarrow right - (right - left)/3$ 
12:     $fXmid1 \leftarrow f(xmid1)$ 
13:     $fXmid2 \leftarrow f(xmid2)$ 
14:     $counter \leftarrow 1$ 
15:     $error1 \leftarrow tol + 1$ 
16:     $error2 \leftarrow tol + 1$ 
17:    while  $error1 > tol \ \& \ error2 > tol \ \& \ fXmid1 \neq 0 \ \& \ fXmid2 \neq 0 \ \& \ counter < nIter$  do
18:      if  $fLeft * fXmid1 < 0$  then
19:         $right \leftarrow xmid1$ 
20:         $fRight \leftarrow fXmid1$ 
21:      else if  $fXmid1 * fXmid2 < 0$  then
22:         $left \leftarrow xmid1$ 
23:         $fLeft \leftarrow fXmid1$ 
24:         $right \leftarrow xmid2$ 
25:         $fRight \leftarrow fXmid2$ 
26:      else
27:         $left \leftarrow xmid2$ 
28:         $fLeft \leftarrow fXmid2$ 
29:      end if
30:       $xAux1 \leftarrow xmid1$ 
31:       $xAux2 \leftarrow xmid2$ 
32:       $xmid1 \leftarrow left + (right - left)/3$ 
33:       $fXmid1 \leftarrow f(xmid1)$ 
34:       $xmid2 \leftarrow right - (right - left)/3$ 
35:       $fXmid2 \leftarrow f(xmid2)$ 
36:       $error1 \leftarrow |xmid1 - xAux1|$ 
37:       $error2 \leftarrow |xmid2 - xAux2|$ 
38:       $counter \leftarrow counter + 1$ 
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

```

16: Jacobi

```

1: function Jacobi(A,b,x,iter,tol)
2:   if foundDeterminant(A) then
3:     WRITE The determinant is zero, the problem has no unique solution return
4:   end if
5:    $d \leftarrow \text{findDiagOfMatrix}(A)$ 
6:    $p \leftarrow \text{findUpperTriangular}(A)$ 
7:    $o \leftarrow \text{findLowerTriangular}(A)$ 
8:    $l \leftarrow d - o$ 
9:    $u \leftarrow d - p$ 
10:   $T \leftarrow \text{findInverseOfMatrix}(d) * (l + u)$ 
11:   $re \leftarrow \text{foundSpectralRadius}(T)$ 
12:  if  $re > 1$  then
13:    WRITE Spectral radius greater than 1: the method does not converge. return
14:     $C \leftarrow \text{finInverseOfMatrix}(d) * b$ 
15:     $i \leftarrow 0$ 
16:     $err \leftarrow tol + 1$ 
17:    while  $err > tol$  AND  $i < iter$  do
18:       $xi \leftarrow T * x + C$ 
19:       $err \leftarrow \text{findNormOfVector}(xi - x)$ 
20:       $x \leftarrow xi$ 
21:       $i \leftarrow i + 1$ 
22:    end while
23:    if  $i \geq iter$  then
24:      WRITE The method fails with maximum number of iterations given return
25:    end if
26:    WRITE x is an approximation with tolerance tol
27:

```

17: Vandermonde

```

1: function Vandermonde(x,y)
2:    $n \leftarrow \text{findVectorLenght}(x)$ 
3:    $A \leftarrow \text{generateMatrixOfOnes}(n)$ 
4:   for  $i = 1 \dots n$  do
5:     for  $j = 1 \dots n - 1$  do
6:        $A_{ij} \leftarrow x_i^{n-j}$ 
7:     end for
8:   end for
9: end function=0

```

18: Crout

```

1: function Crout(A,B)
2:   lower ← matrixnxninitializedinceros
3:   uppper ← identitymatrixnxn
4:   for i = 0...n do
5:     for i=j...n do
6:       sum ← 0
7:       for k = 0...j do
8:         sum ← sum + lower[i][k] * upper[k][j]
9:       end for
10:      lower[i][j] ← A[i][j] - sum
11:    end for
12:    for i = j + 1...n do
13:      sum ← 0
14:      for k = 0...j do
15:        sum ← sum + lower[j][k] * upper[k][i]
16:      end for
17:      upper[j][i] ← (A[j][i] - sum) / lower[j][j]
18:    end for
19:  end for
20:  Z ← regressiveSubstitution(lower|B)
21:  X ← progressiveSubstitution(upper|Z)
22: end function

```

19: Doolittle

```

1: function Doolittle(A)
2:   n ← lenght of A
3:   lower ← matrix nxn initialized in ceros
4:   upper ← identity matrix nxn
5:   for i = 0...n do
6:     for k = i...n do
7:       sum ← 0
8:       for j = i...n do
9:         sum ← lower[i][k] * upper[j][k]
10:      end for
11:      upper[i][k] ← A[i][k] - sum
12:    end for
13:    for k = i...n do
14:      if i = k then
15:        lower[i][i] ← 1
16:      else
17:        sum ← 0
18:        for j = 0...i do
19:          sum ← sum + lower[k][j] * upper[j][i]
20:        end for
21:        lower[k][i] ← (A[k][i] - sum) / upper[i][i]
22:      end if
23:    end for
24:  end for
25:  Z ← regressiveSubstitution(lower|B)
26:  X ← progressiveSubstitution(upper|Z)
27: end function

```

20: Seidel

```

1: function Seidel(A,b,x,iter,tol)
2:   if foundDeterminant(A) then
3:     WRITE The determinant is zero, the problem has no unique solution. return
4:   end if
5:    $d \leftarrow \text{findDiagOfMatrix}(A)$ 
6:    $p \leftarrow \text{findUpperTriangular}(A)$ 
7:    $o \leftarrow \text{findLowerTriangular}(A)$ 
8:    $l \leftarrow d - o$ 
9:    $u \leftarrow d - p$ 
10:   $T \leftarrow \text{findInverseOfMatrix}(d - l) * u$ 
11:   $re \leftarrow \text{foundSpectralRadius}(T)$ 
12:  if  $re > 1$  then
13:    WRITE Spectral radius greater than 1: the method does not converge. return
14:  end if
15:   $C \leftarrow \text{findInverseOfMatrix}(d - l) * b$ 
16:   $i \leftarrow 0$ 
17:   $err \leftarrow tol + 1$ 
18:  while  $err > tol$  AND  $i < iter$  do
19:     $xi \leftarrow T * x + C$ 
20:     $err \leftarrow \text{findNormOfVector}(xi - x)$ 
21:     $x \leftarrow xi$ 
22:     $i \leftarrow i + 1$ 
23:  end while
24:  if  $i \geq iter$  then
25:    WRITE The method fails with the maximum number of iterations given return
26:  end if
27:  WRITE x is an approximation with tolerance tol
28: end function

```

21: Heun

```

1: function Heun(f,x,y,h,n)
2:   counter  $\leftarrow$  0
3:   while counter  $\leq$  n do
4:      $k1 \leftarrow f(x,y)$ 
5:      $k2 \leftarrow f(x + h, y + (k1 * h))$ 
6:      $y \leftarrow y + ((k1 * k2) * (h/2))$ 
7:      $x \leftarrow x + h$ 
8:     counter  $\leftarrow$  counter + 1
9:   end while
10: end function

```

22: Cholesky

```

1: function Cholesky(A,B)
2:    $n \leftarrow \text{lenght of } A$ 
3:   lower  $\leftarrow$  matrix nxn initialized in zeros
4:   for  $j = 0 \dots n$  do
5:     for  $i = 0 \dots j - 1$  do
6:        $\text{lower}[i][j] \leftarrow (A[i][j] - \sum_{k=0}^{i-1} \text{lower}[k][i] * \text{lower}[k][j] / \text{lower}[i][i])$ 
7:     end for
8:      $\text{lower}[i][j] \leftarrow \sqrt{A[i][j] - \sum_{k=0}^{i-1} \text{lower}[k][j]^2}$ 
9:   end for
10:  upper  $\leftarrow \text{lower}^T$ 
11:  Z  $\leftarrow \text{regressiveSubstitution}(\text{lower}|B)$ 
12:  X  $\leftarrow \text{progressiveSubstitution}(\text{upper}|Z)$ 
13: end function

```


23: Lagrange

```

1: function Lagrange(x,y)
2:   yp ← 0
3:   p ← 0
4:   for i = 1...length(x) do
5:     p ← 1
6:     for j = 1...length(x) do
7:       if x[i] = x[j] then
8:         p ← p * (xp - x[j]) / (x[i] - x[j])
9:       end if
10:    end for
11:    yp ← yp + p * y[i]
12:  end for
13: end function

```

24: Simpson 1/3

```

1: function Simpson(a,b,f,n)
2:   deltaX ← (b - a) / n
3:   A ← 0
4:   for i = 0...n do
5:     xi ← a + i * deltaX
6:     fxi ← f(xi)
7:     if i > 0 AND i < n then
8:       if i module 2 == 0 then
9:         fxi ← 2 * fxi
10:      else
11:        fxi ← 4 * fxi
12:      end if
13:    end if
14:    A ← A + fxi
15:  end for
16:  A ← A * (deltaX/3)
17:  WRITE A is the result of the integral
18: end function

```

25: Simpson 3/8

```

1: function Simpson(a,b,f,n)
2:   deltaX ← (b - a) / n
3:   A ← 0
4:   for i = 0...n do
5:     xi ← a + i * deltaX
6:     fxi ← f(xi)
7:     if i > 0 AND i < n then
8:       if i module 2 == 0 then
9:         fxi ← 2 * fxi
10:      else
11:        fxi ← 3 * fxi
12:      end if
13:    end if
14:    A ← A + fxi
15:  end for
16:  A ← A * (3 * deltaX/8)
17:  WRITE A is the result of the integral
18: end function

```

26: SOR

```

1: function SOR(A,b,x,iter,tol)
2:   if foundDeterminant(A) then
3:     WRITE The determinant is zero, the problem has no unique solution. return
4:   end if
5:    $d \leftarrow \text{findDiagOfMatrix}(A)$ 
6:    $p \leftarrow \text{findUpperTriangular}(A)$ 
7:    $o \leftarrow \text{findLowerTriangular}(A)$ 
8:    $l \leftarrow d - o$ 
9:    $u \leftarrow d - p$ 
10:   $T \leftarrow \text{findInverseOfMatrix}(d - w * l) * [(1 - w)d + w * u]$ 
11:   $re \leftarrow \text{foundSpectralRadius}(T)$ 
12:  if  $re > 1$  then
13:    WRITE Spectral radius greater than 1: the method does not converge. return
14:  end if
15:   $C \leftarrow w * \text{findInverseOfMatrix}(d - w * l) * b$ 
16:   $i \leftarrow 0$ 
17:   $err \leftarrow tol + 1$ 
18:  while  $err > tol$  AND  $i < iter$  do
19:     $xi \leftarrow T * x + C$ 
20:     $err \leftarrow \text{findNormOfVector}(xi - x)$ 
21:     $x \leftarrow xi$ 
22:     $i \leftarrow i + 1$ 
23:  end while
24:  if  $i \geq iter$  then
25:    WRITE The method fails with the maximum number of iterations given return
26:  end if
27:  WRITE x is an approximation with tolerance tol
28: end function

```

27: Lineal Spline

```

1: function LinealSpline(x,y)
2:   zerosA  $\leftarrow$  create a square matrix of 0s x.length
3:   zerosB  $\leftarrow$  create matrix of 0s x.length by 1
4:    $m \leftarrow 2 * (n - 1)$ 
5:    $z \leftarrow 0$ 
6:   for  $i = 1 \dots \text{length}(x)$  do
7:      $\text{zerosA}[i][z] \leftarrow x[i]$ 
8:      $\text{zerosA}[i][z + 1] \leftarrow 1$ 
9:      $z \leftarrow z + 2$ 
10:     $\text{zerosB}[i][z] \leftarrow y[i]$ 
11:  end for
12:   $\text{zerosA}[0][0] \leftarrow x[0]$ 
13:   $\text{zerosA}[0][1] \leftarrow 1$ 
14:   $\text{zerosB}[0][0] \leftarrow y[0]$ 
15:  for  $i = 1 \dots \text{length}(x)$  do
16:     $\text{zerosA}[\text{x.length}][z] \leftarrow x[i]$ 
17:     $\text{zerosA}[\text{x.length}][z + 2] \leftarrow -x[i]$ 
18:     $\text{zerosA}[\text{x.length}][z + 1] \leftarrow 1$ 
19:     $\text{zerosA}[\text{x.length}][x + 3] \leftarrow -1$ 
20:     $\text{zerosB}[\text{x.length}][0] \leftarrow 0$ 
21:     $z \leftarrow z + 2$ 
22:  end for
23: end function

```

28: Quadratic Spline

```

1: function QuadraticSpline(x,y)
2:   zerosA ← create a square matrix of 0s x.length
3:   zerosB ← create matrix of 0s x.length by 1
4:    $m \leftarrow 3 * (n - 1)$ 
5:    $z \leftarrow 0$ 
6:   for  $i = 1 \dots \text{length}(x)$  do
7:     zerosA[i][z] ←  $x[i]^2$ 
8:     zerosA[i][z + 1] ←  $x[i]$ 
9:     zerosA[i][z + 2] ← 1
10:     $z \leftarrow z + 3$ 
11:    zerosB[i][0] ←  $y[i]$ 
12:  end for
13:  zerosA[0][0] ←  $x[0]^2$ 
14:  zerosA[0][1] ←  $x[0]$ 
15:  zerosA[0][2] ← 1
16:  zerosB[0][0] ←  $y[0]$ 
17:  for  $i = 1 \dots \text{length}(x)$  do
18:    zerosA[x.length][z] ←  $x[i]^2$ 
19:    zerosA[x.length][z + 1] ←  $x[i]$ 
20:    zerosA[x.length][z + 2] ← 1
21:    zerosA[x.length][z + 3] ←  $-x[i]^2$ 
22:    zerosA[x.length][z + 4] ←  $-x[i]$ 
23:    zerosA[x.length][z + 5] ← -1
24:    zerosB[x.length][0] ← 0
25:     $z \leftarrow z + 3$ 
26:  end for
27:   $z \leftarrow 0$ 
28:  for  $i = 2 \dots \text{length}(x) - 1$  do
29:    zerosA[2 * x.length - 3][z] ←  $x[i - 1]^2$ 
30:    zerosA[2 * x.length - 3][z + 1] ← 1
31:    zerosA[2 * x.length - 3][z + 2] ← 0
32:    zerosA[2 * x.length - 3][z + 3] ←  $-(x[i - 1] * 2)$ 
33:    zerosA[2 * x.length - 3][z + 4] ← -1
34:    zerosA[2 * x.length - 3][z + 5] ← 0
35:     $z \leftarrow z + 3$ 
36:  end for
37:  zerosA[m - 1][0] ← 2
38:  zerosB[m - 1][0] ← 0
39: end function

```

29: Cubic Spline

```

1: function CubicSpline(x,y)
2:   zerosA  $\leftarrow$  create a square matrix of 0s x.length
3:   zerosB  $\leftarrow$  create matrix of 0s x.length by 1
4:    $m \leftarrow 3 * (n - 1)$ 
5:    $z \leftarrow 0$ 
6:   for  $i = 1 \dots \text{length}(x)$  do
7:     zerosA[i][z]  $\leftarrow x[i]^3$ 
8:     zerosA[i][z + 1]  $\leftarrow x[i]^2$ 
9:     zerosA[i][z + 2]  $\leftarrow x[i]$ 
10:    zerosA[i][z + 3]  $\leftarrow 1$ 
11:     $z \leftarrow z + 4$ 
12:    zerosB[i][0]  $\leftarrow y[i]$ 
13:   end for
14:   zerosA[0][0]  $\leftarrow x[0]^3$ 
15:   zerosA[0][1]  $\leftarrow x[0]^2$ 
16:   zerosA[0][2]  $\leftarrow x[0]$ 
17:   zerosA[0][3]  $\leftarrow 1$ 
18:   zerosB[0][0]  $\leftarrow y[0]$ 
19:    $z \leftarrow 0$ 
20:   for  $i = 2 \dots \text{length}(x) - 1$  do
21:     zerosA[2 * x.length - 2 + i][z]  $\leftarrow x[i - 1]^3$ 
22:     zerosA[2 * x.length - 2 + i][z + 1]  $\leftarrow x[i - 1]^2$ 
23:     zerosA[2 * x.length - 2 + i][z + 2]  $\leftarrow x[i - 1]$ 
24:     zerosA[2 * x.length - 2 + i][z + 3]  $\leftarrow 1$ 
25:     zerosA[2 * x.length - 2 + i][x + 4]  $\leftarrow -(x[i - 1]^3)$ 
26:     zerosA[2 * x.length - 2 + i][x + 5]  $\leftarrow -(x[i - 1]^2)$ 
27:     zerosA[2 * x.length - 2 + i][x + 6]  $\leftarrow -x[i - 1]$ 
28:     zerosA[2 * x.length - 2 + i][x + 7]  $\leftarrow -1$ 
29:      $z \leftarrow z + 4$ 
30:     zerosB[x.length - 1 + i][0]  $\leftarrow 0$ 
31:   end for
32:    $z \leftarrow 0$ 
33:   for  $i = 2 \dots \text{length}(x) - 1$  do
34:     zerosA[2 * x.length - 4 + i][z]  $\leftarrow x[i - 1]^2 * 3$ 
35:     zerosA[2 * x.length - 4 + i][z + 1]  $\leftarrow x[i - 1]^1 * 2$ 
36:     zerosA[2 * x.length - 4 + i][z + 2]  $\leftarrow 1$ 
37:     zerosA[2 * x.length - 4 + i][z + 3]  $\leftarrow 0$ 
38:     zerosA[2 * x.length - 4 + i][x + 4]  $\leftarrow -(x[i - 1]^3 * 3)$ 
39:     zerosA[2 * x.length - 4 + i][x + 5]  $\leftarrow -(x[i - 1]^2 * 2)$ 
40:     zerosA[2 * x.length - 4 + i][x + 6]  $\leftarrow -1$ 
41:     zerosA[2 * x.length - 4 + i][x + 7]  $\leftarrow 0$ 
42:      $z \leftarrow z + 4$ 
43:     zerosB[2 * x.length - 3 + i][0]  $\leftarrow 0$ 
44:   end for
45:    $z \leftarrow 0$ 
46:   for  $i = 2 \dots \text{length}(x) - 1$  do
47:     zerosA[3 * x.length - 6 + i][z]  $\leftarrow x[i - 1] * 6$ 
48:     zerosA[3 * x.length - 6 + i][z + 1]  $\leftarrow 2$ 
49:     zerosA[3 * x.length - 6 + i][z + 2]  $\leftarrow 0$ 
50:     zerosA[3 * x.length - 6 + i][z + 3]  $\leftarrow 0$ 
51:     zerosA[3 * x.length - 6 + i][x + 4]  $\leftarrow -(x[i - 1] * 6)$ 
52:     zerosA[3 * x.length - 6 + i][x + 5]  $\leftarrow -2$ 
53:     zerosA[3 * x.length - 6 + i][x + 6]  $\leftarrow 0$ 
54:     zerosA[3 * x.length - 6 + i][x + 7]  $\leftarrow 0$ 
55:      $z \leftarrow z + 4$ 
56:     zerosB[2 * x.length - 3 + i][0]  $\leftarrow 0$ 
57:   end for
58:   zerosA[m - 2][0]  $\leftarrow x[0] * 6$ 

```

```

59:   zerosA[m - 2][0] ← 2
60:   zerosA[m - 1][0] ← x[x.length] * 6
61:   zerosA[m - 1][0] ← 2
62:   zerosB[m - 1][0] ← 0
63:   zerosA[m - 2][0] ← 0
64: end function

```

30: Compound Trapeze

```

1: function CompoundTrapeze(a,b,f,n)
2:   deltaX ← (b - a) / n
3:   A ← 0
4:   for i = 0...n do
5:     xi ← a + i * deltaX
6:     fxi ← f(xi)
7:     if i > 0 AND i < n then
8:       fxi ← 2 * fxi
9:     end if
10:    A ← A + fxi
11:  end for
12:  A ← A * (deltaX/2)
13:  WRITE A is the result of the integral
14: end function

```

31: Euler

```

1: function Euler(f,xi,yi,xf,h)
2:   n ← (xf - xi) / h
3:   for i = 0...n do
4:     y1 ← f(xi,yi)
5:     hyi ← g * y1
6:     newarray.push([xi,y1])
7:     yi ← yi + hy1
8:     xi ← xi + h
9:   end for
10: 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	1.0000000000000000	0.035366079380240	1.0000010000000000
1.0000000000000000	0.946166222306525	0.005619392737864	0.053833777693475
2.0000000000000000	0.935996580791173	-0.000236322174701	0.010169641515352
3.0000000000000000	0.936407002376704	0.000001402235891	0.000410421585531
4.0000000000000000	0.936404581473120	0.000000000343716	0.000002420903584
5.0000000000000000	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 =

15x9 [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

Etapa 0 :

Matriz L:

x0	x1	x2	x3
4.00000	0.00000	0.00000	0.00000
1.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
14.00000	0.00000	0.00000	0.00000

Matriz U:

x0	x1	x2	x3
1.00000	-0.25000	0.00000	0.75000
0.00000	1.00000	0.00000	0.00000
0.00000	0.00000	1.00000	0.00000
0.00000	0.00000	0.00000	1.00000

Etapa 2 :

Matriz 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	0.00000

Matriz 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

Etapa 1 :

Matriz 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	0.00000	0.00000
14.00000	8.50000	0.00000	0.00000

Matriz 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	0.00000
0.00000	0.00000	0.00000	1.00000

Etapa 3 :

Matriz 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

Matriz 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

16: Doolittle

Etapa 0 :

Matriz L:

x0	x1	x2	x3
1.00000	0.00000	0.00000	0.00000
0.25000	1.00000	0.00000	0.00000
0.00000	0.00000	1.00000	0.00000
3.50000	0.00000	0.00000	1.00000

Matriz U:

x0	x1	x2	x3
4.00000	-1.00000	0.00000	3.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000

Etapa 1 :

Matriz 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	0.00000	1.00000

Matriz 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	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000

Etapa 2 :

Matriz 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

Matriz 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	0.00000

Etapa 3 :

Matriz 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

Matriz 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

Prueba: (analiza con la matriz ingresada)

```
[[ 4. -1.  0.  3.]
 [ 1. 15.5  3.  8.]
 [ 0. -1.3 -4.  1.1]
 [14.  5. -2. 30. ]]
```

Members signatures

1: Jacobo Rave

Jacobo Rave Londoño

2: David Echeverri



3: Kevin Sossa

Kevin Sossa

4: Sebastián Guerra

