

EAFIT UNIVERSITY

IT AND SYSTEMS DEPARTMENT

CHOICE OF THE PROJECT

Second report

Objective: Implementation of the numerical methods seen until this point in the semester, including the additional ones.

Course: Numerical analysis.

Teacher: Edwar Samir Posada Murillo.

Semester: 2020-2.

Due date for the second report: Tuesday October 13.

Project name: Numerical views.

Project Repository: [Link Repo](#)

Members:

Mariana Ramírez Duque (marami21@eafit.edu.co)

Nicolás Roldán Ramírez (nroldanr@eafit.edu.co)

Mateo Sánchez Toro (msanchezt@eafit.edu.co)

Maria Cristina Castrillon (Mcastri6@eafit.edu.co)

Report description: We will write the pseudocode of each method followed by an example execution. The code can be found on the repository link cited above

1 Methods:

Gaussian Elimination with partial pivot

Gaussian Elimination with total pivot

Fixed Point

Algorithm 1: Incremental search

```
input : f: function to find root of,  
        X0: first root approximation,  
        Delta: delta(x),  
        N: maximum number of iterations,  
  
if ( $y=0$ ) then  
    | x0 is root;  
else  
    |  $x1 \leftarrow [x0 \text{ } \textit{delta}]$   
    |  $y1 \leftarrow [f(x1)]$   
    |  $\textit{counter} \leftarrow [1]$   
while  $y1 * y0 > 0$  and  $\textit{counter} < n$  do do  
    |  $x0 \leftarrow [x1]$   
    |  $y0 \leftarrow [y1]$   
    |  $x1 \leftarrow [x0 + \textit{delta}]$   
    |  $y1 \leftarrow [f(x1)]$   
    |  $\textit{counter} \leftarrow \textit{counter} + 1$   
if ( $y1==0$ ) then  
    | x1 is root;  
else  
    | if ( $y1 * y0 < 0$ ) then  
    | |  $[x0,x1]$  define an interval;  
    | else  
    | | print("didn't find interval in N iterations")
```

Algorithm 2: Bisection

```
input  : xi: lower limit of the interval,
         xs: the upper range,
         f: function to find,
         N: maximum number of iterations,
         tol: tolerance,

 $yi \leftarrow f(xi)$ 
 $ys \leftarrow f(xs)$ 
if ( $yi == 0$ ) then
  | print("xi is root")
else
  | if ( $ys == 0$ ) then
  | | print("xs is root")
else
  | if ( $yi * ys > 0$ ) then
  | | print("Inappropriate range")
else
  |  $xm \leftarrow (xi + xs)/2$ 
  |  $ym \leftarrow f(xm)$ 
  |  $error \leftarrow tol + 1$ 
  |  $counter \leftarrow [1]$ 
while  $ym \neq 0$  and ( $error > tol$ ) and ( $counter < n$ ) do do
  ( $yi * ym > 0$ )  $xi \leftarrow xm, yi \leftarrow ym$ 
else
  |  $xs \leftarrow xm, ys \leftarrow ym$ 
  |  $xaux \leftarrow xm$ 
  |  $xm \leftarrow xi + xs/2$ 
  |  $ym \leftarrow f(xm)$ 
  |  $error \leftarrow xm - xaux$ 
  |  $counter \leftarrow +1$ 
if ( $ym == 0$ ) then
  | print("xm is root")
else
  | if ( $error < tol$ ) then
  | | print("Xm is root with an error equal to error")
else
  | print("The method failed")
```

Algorithm 3: False Rule

```
input  : a: lower limit of the interval,
         b: the upper range,
         f: function to find,
         N: maximum number of iterations,
         tol: tolerance,

 $ya \leftarrow f(a)$ 
 $yb \leftarrow f(b)$ 
if ( $ya == 0$ ) then
|   print("a is root")
else
|   if ( $yb == 0$ ) then
|   |   print("b is root")
else
|   if ( $ya * yb > 0$ ) then
|   |   print("Inappropriate range")
else
|    $p \leftarrow a - f(a) * (b - a) / f(b) - f(a)$ 
|    $yp \leftarrow f(p)$ 
|    $error \leftarrow tol + 1$ 
|    $counter \leftarrow [1]$ 
while  $ypf = 0$  and ( $error > tol$ ) and ( $contador < n$ ) do do
( $ya * yp > 0$ )   $a \leftarrow p, ya \leftarrow yp$ 
else
|    $b \leftarrow p, yb \leftarrow yp$ 
|    $p_{aux} \leftarrow p$ 
|    $p \leftarrow a - f(a) * (b - a) / f(b) - f(a)$ 
|    $yp \leftarrow f(p)$ 
|    $error \leftarrow p - p_{aux}$ 
|    $counter \leftarrow +1$ 
if ( $yp == 0$ ) then
|   print("p is root")
else
|   if ( $error < tol$ ) then
|   |   print("p is root with an error equal to error")
else
|   print("The method failed")
```

Algorithm 4: Multiple roots

input : f: function to find root of,
fprime: derivative of f,
f2prime: second derivative of f,
tol: error tolerance,
N: maximum number of iterations,
X0: first root approximation

output: root approximation x

```
x ← x0
fun ← (x)
funprime ← fprime(x)
fun2prime ← f2prime(x)
error ← infinity
for i ← 0 to N do
    if error ≤ tol(i,i) = 0 then
        | break;
    error ← x
    x ← x0
    fun ← f(x)
    funprime ← fprime(x)
    fun2prime ← f2prime(x)
    error ← infinity
    error ← abs(error - x)
```

Algorithm 5: Gaussian elimination

input : nxn matrix A, column vector b

output: solution vector x

```
if (A is not square) or (size of A and size of b are not computable) then
    | break ;
if det(A) = 0 then
    | break ;
A ← [A b]
for i ← 1 to n - 1 do
    if  $\bar{A}(i,i) = 0$  then
        | find l such that  $\bar{A}(l,i) \neq 0$  ;
        | switch  $\bar{A}(i)$  and  $\bar{A}(l)$ 
        for j ← i + 1 to n do
            | multiplier  $M_{j,i} \leftarrow \frac{\bar{A}(j,i)}{\bar{A}(i,i)}$ 
            |  $\bar{A}_j \leftarrow \bar{A}_j - M_{j,i} * \bar{A}_i$ 
x ← susreg( $\bar{A}$ )
```

Algorithm 6: Gaussian elimination with partial pivot

input : square $n \times n$ matrix A , n vector b
output: solution vector x

```
if  $\det(A) == 0$  then
| break ;
for  $i \leftarrow 1$  to  $n - 1$  do
| champion  $\leftarrow i$ 
| for  $j \leftarrow \text{champion} + 1$  to  $n - 1$  do
| | if  $\text{abs}(A(j,i)) == \text{abs}(A(\text{champion},i))$  then
| | | champion  $\leftarrow i$ 
| Swap row in  $A(\text{champion})$  with  $A(i)$ 
| Swap value in  $b(\text{champion})$  with  $b(i)$ 
| for  $j \leftarrow i + 1$  to  $n - 1$  do
| | multiplicand  $\leftarrow \frac{A(j,i)}{A(i,i)}$ 
| |  $A_j \leftarrow A_j - M_{j,i} * A_i$ 
 $x \leftarrow \text{susreg}(A)$ 
```

Algorithm 7: Gaussian elimination with total pivot

input : square $n \times n$ matrix A , n vector b
output: solution vector x

```
if  $\det(A) == 0$  then
| break ;
posStamp  $\leftarrow [from 0 \text{ to } n - 1]$ 
for  $i \leftarrow 1$  to  $n - 1$  do
| row  $\leftarrow i$ 
| col  $\leftarrow i$ 
| for  $j \leftarrow row$  to  $n - 1$  do
| | for  $k \leftarrow col$  to  $n - 1$  do
| | | if  $\text{abs}(A(i,j)) == \text{abs}(A(row,col))$  then
| | | | row  $\leftarrow i$  col  $\leftarrow j$ 
| if col  $\neq i$  then
| | Swap column  $A(:,col)$  with  $A(:,i)$  Swap value in posStamp(col) with
| | posStamp(i)
| if row  $\neq i$  then
| | Swap row  $A(row)$  with  $A(i)$  Swap value in b(row) with b(i)
| for  $j \leftarrow i + 1$  to  $n - 1$  do
| | multiplicand  $\leftarrow \frac{A(j,i)}{A(i,i)}$ 
| |  $A_j \leftarrow A_j - M_{j,i} * A_i$ 
 $x \leftarrow \text{susreg}(A)$ 
Sort  $x$  using posStamp
```

Algorithm 8: Fixed Point

input : $f(x)$, $g(x)$, x_0 , tol , N
output: The approximate root

Define variables
int $i = 0$;
double $x = 0$;
Start While ($\text{AbsoluteValue}(f(x)) \leq \text{tol}$ $i \leq N$)
 $x = g(x)$;
 $i++$;
End While
Start if ($\text{AbsoluteValue}(f(x)) \leq 1E-8$)
print "Root:" + x ;
else
print "Not possible to obtain root"
End if

Algorithm 9: Secant

input : iter , x_i , $f(x_i)$, E
output: The approximate root

$I = f(X_0)$
If $I = 0$ Then
Show: " X_0 is Root"
 $Y_1 = f(X_1)$
Counter = 0
Error = Tolerance + 1
While
 $X_2 = X_1 - ((Y_1 * (X_1 - X_0)) / \text{Den})$
Error = Abs $((X_2 - X_1) / X_2)$
 $X_0 = X_1$
 $I = Y_1$
 $X_1 = X_2$
 $Y_1 = f(X_1)$
Counter = Counter + 1
End while
If $Y_1 = 0$ Then
Show: " X_1 is Root"
Otherwise If Error \leq Tolerance Then
Show: " X_1 'is an approximate root with a tolerance' Tolerance "
Otherwise If Den = 0 Then
Show: 'There is possibly a multiple root'

Secant Newton

Algorithm 10: Newton

```
input : f(x), g(x), x0, tol, N
output: The approximate root

if fx == 0:
    return the root
end if
if dfun == 0:
    return Error, derivative cannot be 0.
end if
counter = 0
error = tolerance + 1
while error > tol and fx! = 0 and dfx! = 0 and counter < iterations:
    xn = xi - (fx / dfx)
    fx = fun.evaluate2 (xn)
    dfx = dfun.evaluate2 (xn)
end while
if typeerror == 0 :
    error = — xn-xi —
else:
    error = — (xn-xi) / xn —
end if
xi = xn
counter +1
if fx == 0:
    return xi is root
elif error < tol:
    return xn is an approximation to a root with a tolerance ”
elif dfx == 0:
    return xn is a possible multiple root
else:
    return The method failed in n iterations
end if
```

Extra method Aikten

Extra method Steffensen

Algorithm 11: Extra method Aikten

input : function f , float tolerance, integer maxIterations
output: solution vector x

$x \leftarrow f(1)$
 $x1 \leftarrow f(2)$
 $x2 \leftarrow f(3)$
 $\text{aikten1} \leftarrow 1$ $\text{aikten2} \leftarrow 0$
for $i \leftarrow 1$ **to** maxIterations **do**
 if $\text{abs}(\text{aikten1} - \text{aikten0}) \neq \text{tolerance}$ **then**
 | break ;
 $\text{aikten1} \leftarrow \text{aikten2}$ $\text{aikten2} \leftarrow \text{aiktkenEcuation}(x, x1, x2)$
 $x \leftarrow x1$ $x1 \leftarrow x2$ $x2 \leftarrow f(i + 3)$

Algorithm 12: Extra method Steffensen

input : function f , float tolerance, integer maxIterations , float approximation
output: root of f

$x0 \leftarrow \text{approximation}$
 $x1 \leftarrow f(x0)$
 $x2 \leftarrow f(x1)$
 $x3 \leftarrow \text{aiktkenEcuation}(x0, x1, x2)$
for $i \leftarrow 1$ **to** maxIterations **do**
 if $\text{abs}(x0 - x3) \neq \text{tolerance}$ **then**
 | break ;
 $x0 \leftarrow x1$
 $x1 \leftarrow x2$
 $x2 \leftarrow f(i + 3)$
 $x3 \leftarrow \text{aiktkenEcuation}(x0, x1, x2)$
 $x \leftarrow x3$

Algorithm 13: Extra method Muller

input : function f , root approximation 1 x_0 , root approximation 2 x_1 , root approximation 3 x_2 , tolerance tol , Maximum number of iterations N

output: root approximation

```
h1 = x1 - x0
h2 = x2 - x1
t1 = (f(x1) - f(x0))/h1
t2 = (f(x2) - f(x1))/h2
d = (t2 - t1)/(h2 + h1)
for  $i \leftarrow 3$  to  $N$  do
     $b = t2 + h2 * d$ 
     $D = (b^2 - 4 * f(x2) * d)^{1/2}$ 
    if  $abs(b - D) < abs(b + D)$  then
        |  $E = b + D$ ;
    else
        |  $E = b - D$ 
     $h = (-2 * f(x2))/E$ 
     $e = x$ 
     $x = x2 + h$ 
     $e = abs(e - x)$ 
    if  $abs(e) \leq tol$  then
        | break;
     $x_0 = x_1$ 
     $x_1 = x_2$ 
     $x_2 = x$ 
     $h1 = x1 - x0$ 
     $h2 = x2 - x1$ 
     $t1 = (f(x1) - f(x0))/h1$ 
     $t2 = (f(x2) - f(x1))/h2$ 
     $d = (t2 - t1)/(h2 + h1)$ 
```

2 Pruebas

Incremental search

INCREMENTAL SEARCHES

Results:

```
There is a root of f in [ -2.5 , -2.0 ]
There is a root of f in [ -1.0 , -0.5 ]
There is a root of f in [ 0.5 , 1.0 ]
There is a root of f in [ 2.0 , 2.5 ]
There is a root of f in [ 4.0 , 4.5 ]
There is a root of f in [ 5.0 , 5.5 ]
There is a root of f in [ 7.0 , 7.5 ]
There is a root of f in [ 8.0 , 8.5 ]
There is a root of f in [ 10.0 , 10.5 ]
There is a root of f in [ 11.5 , 12.0 ]
There is a root of f in [ 13.5 , 14.0 ]
There is a root of f in [ 14.5 , 15.0 ]
There is a root of f in [ 16.5 , 17.0 ]
There is a root of f in [ 17.5 , 18.0 ]
There is a root of f in [ 19.5 , 20.0 ]
There is a root of f in [ 21.0 , 21.5 ]
There is a root of f in [ 22.5 , 23.0 ]
There is a root of f in [ 24.0 , 24.5 ]
There is a root of f in [ 26.0 , 26.5 ]
There is a root of f in [ 27.0 , 27.5 ]
There is a root of f in [ 29.0 , 29.5 ]
There is a root of f in [ 30.0 , 30.5 ]
There is a root of f in [ 32.0 , 32.5 ]
There is a root of f in [ 33.5 , 34.0 ]
There is a root of f in [ 35.0 , 35.5 ]
There is a root of f in [ 36.5 , 37.0 ]
There is a root of f in [ 38.5 , 39.0 ]
There is a root of f in [ 39.5 , 40.0 ]
There is a root of f in [ 41.5 , 42.0 ]
There is a root of f in [ 43.0 , 43.5 ]
There is a root of f in [ 44.5 , 45.0 ]
There is a root of f in [ 46.0 , 46.5 ]
```

```
>>>
```

Bisection

BISECTION

Results:

iter	a	xm	b	f(Xm)	E
1.0	0.5000000000	0.7500000000	1.0000000000	-0.11839639385	0.2500000000
2.0	0.7500000000	0.8750000000	1.0000000000	-0.03681769076	0.1250000000
3.0	0.8750000000	0.9375000000	1.0000000000	0.00063391616	0.0625000000
4.0	0.8750000000	0.9062500000	0.9375000000	-0.01777228923	0.0312500000
5.0	0.9062500000	0.9218750000	0.9375000000	-0.00848658221	0.0156250000
6.0	0.9218750000	0.9296875000	0.9375000000	-0.00390535863	0.0078125000
7.0	0.9296875000	0.9335937500	0.9375000000	-0.00163043812	0.0039062500
8.0	0.9335937500	0.9355468750	0.9375000000	-0.00049693532	0.0019531250
9.0	0.9355468750	0.9365234375	0.9375000000	0.00006882244	0.0009765625
10.0	0.9355468750	0.93603515625	0.9365234375	-0.00021397351	0.00048828125
11.0	0.93603515625	0.93627929688	0.9365234375	-0.00007255479	0.00024414062
12.0	0.93627929688	0.93640136719	0.9365234375	-0.00000186098	0.00012207031
13.0	0.93640136719	0.93646240234	0.9365234375	0.00003348203	0.00006103516
14.0	0.93640136719	0.93643188477	0.93646240234	0.00001581085	0.00003051758
15.0	0.93640136719	0.93641662598	0.93643188477	0.00000697501	0.00001525879
16.0	0.93640136719	0.93640899658	0.93641662598	0.00000255703	0.00000762939
17.0	0.93640136719	0.93640518188	0.93640899658	0.00000034803	0.00000381470
18.0	0.93640136719	0.93640327454	0.93640518188	-0.00000075648	0.00000190735
19.0	0.93640327454	0.93640422821	0.93640518188	-0.00000020422	0.00000095367
20.0	0.93640422821	0.93640470505	0.93640518188	0.00000007190	0.00000047684
21.0	0.93640422821	0.93640446663	0.93640470505	-0.00000006616	0.00000023842
22.0	0.93640446663	0.93640458584	0.93640470505	0.00000000287	0.00000011921
23.0	0.93640446663	0.93640452623	0.93640458584	-0.00000003164	0.00000005960

0.936404526236731 is root with tol: 5.960464477539063e-08

Figure 2: Proof of Bisection

False Rule

FALSE RULE

Results:

iter	a	xm	b	f(Xm)	E
1.0	0.93394038072	0.93650605167	1.00000000000	0.00005875601	0.00256567095
2.0	0.93394038072	0.93640473074	0.93650605167	0.00000008678	0.00010132092
3.0	0.93394038072	0.93640458110	0.93640473074	0.00000000013	0.00000014964
4.0	0.93394038072	0.93640458088	0.93640458110	0.00000000000	0.00000000022

0.936404580879889 is root with tol: 2.2098023411132317e-10

>>>

Figure 3: Proof of False Rule

Multiple roots

Raices multiples

iter	xi		f(xi)		E
0	1.000000e+00		7.182818e-01		
1	-2.342106e-01		2.540578e-02		1.234211e+00
2	-8.458280e-03		3.567061e-05		2.426689e-01
3	-1.189018e-05		7.068790e-11		8.470170e-03
4	-4.218591e-11		0.000000e+00		1.189023e-05
5	-4.218591e-11		0.000000e+00		8.437181e-11

A root approximation was found at -4.218590698935789e-11

Figure 4: Proof of Gaussian Elimination

```

{2.0, -1.0, 0.0, 3.0, 1.0}
{1.0, 0.5, 3.0, 8.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{14.0, 5.0, -2.0, 3.0, 1.0}

Phase 1
{14.0, 5.0, -2.0, 3.0, 1.0}
{1.0, 0.5, 3.0, 8.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{2.0, -1.0, 0.0, 3.0, 1.0}

{14.0, 5.0, -2.0, 3.0, 1.0}
{0.0, 0.1428571428571429, 3.142857142857143, 7.785714285714286, 0.9285714285714286}
{0.0, 13.0, -2.0, 11.0, 1.0}
{0.0, -1.7142857142857142, 0.2857142857142857, 2.5714285714285716, 0.8571428571428572}

Phase 2
{14.0, 5.0, -2.0, 3.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{0.0, 0.1428571428571429, 3.142857142857143, 7.785714285714286, 0.9285714285714286}
{0.0, -1.7142857142857142, 0.2857142857142857, 2.5714285714285716, 0.8571428571428572}

{14.0, 5.0, -2.0, 3.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{0.0, 0.0, 3.1648351648351647, 7.664835164835164, 0.9175824175824177}
{0.0, 2.220446049250313E-16, 0.021978021978021955, 4.021978021978022, 0.989010989010989}

Phase 3
{14.0, 5.0, -2.0, 3.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{0.0, 0.0, 3.1648351648351647, 7.664835164835164, 0.9175824175824177}
{0.0, 2.220446049250313E-16, 0.021978021978021955, 4.021978021978022, 0.989010989010989}

{14.0, 5.0, -2.0, 3.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{0.0, 0.0, 3.1648351648351647, 7.664835164835164, 0.9175824175824177}
{0.0, 2.220446049250313E-16, 0.0, 3.96875, 0.982638888888889}

{14.0, 5.0, -2.0, 3.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{0.0, 0.0, 3.1648351648351647, 7.664835164835164, 0.9175824175824177}
{0.0, 2.220446049250313E-16, 0.0, 3.96875, 0.982638888888889}

Answers{0.03849518810148731, -0.18022747156605426, -0.30971128608923887, 0.24759405074365706}

```

Figure 5: Proof Partial Pivot

```

{2.0, -1.0, 0.0, 3.0, 1.0}
{1.0, 0.5, 3.0, 8.0, 1.0}
{0.0, 13.0, -2.0, 11.0, 1.0}
{14.0, 5.0, -2.0, 3.0, 1.0}

Phase 1
{11.0, 13.0, -2.0, 0.0, 1.0}
{8.0, 0.5, 3.0, 1.0, 1.0}
{3.0, -1.0, 0.0, 2.0, 1.0}
{3.0, 5.0, -2.0, 14.0, 1.0}

{11.0, 13.0, -2.0, 0.0, 1.0}
{0.0, -8.954545454545455, 4.454545454545455, 1.0, 0.2727272727272727}
{0.0, -4.545454545454545, 0.5454545454545454, 2.0, 0.7272727272727273}
{0.0, 1.454545454545455, -1.4545454545454546, 14.0, 0.7272727272727273}

Phase 2
{11.0, 0.0, -2.0, 13.0, 1.0}
{0.0, 14.0, -1.4545454545454546, 1.454545454545455, 0.7272727272727273}
{0.0, 2.0, 0.5454545454545454, -4.545454545454545, 0.7272727272727273}
{0.0, 1.0, 4.454545454545455, -8.954545454545455, 0.2727272727272727}

{11.0, 0.0, -2.0, 13.0, 1.0}
{0.0, 14.0, -1.4545454545454546, 1.454545454545455, 0.7272727272727273}
{0.0, 0.0, 0.7532467532467532, -4.753246753246753, 0.6233766233766234}
{0.0, 0.0, 4.558441558441559, -9.058441558441558, 0.22077922077922077}

Phase 3
{11.0, 0.0, 13.0, -2.0, 1.0}
{0.0, 14.0, 1.454545454545455, -1.4545454545454546, 0.7272727272727273}
{0.0, 0.0, -9.058441558441558, 4.558441558441559, 0.22077922077922077}
{0.0, 0.0, -4.753246753246753, 0.7532467532467532, 0.6233766233766234}

{11.0, 0.0, 13.0, -2.0, 1.0}
{0.0, 14.0, 1.454545454545455, -1.4545454545454546, 0.7272727272727273}
{0.0, 0.0, -9.058441558441558, 4.558441558441559, 0.22077922077922077}
{0.0, 0.0, 0.0, -1.6387096774193548, 0.5075268817204301}

{11.0, 0.0, 13.0, -2.0, 1.0}
{0.0, 14.0, 1.454545454545455, -1.4545454545454546, 0.7272727272727273}
{0.0, 0.0, -9.058441558441558, 4.558441558441559, 0.22077922077922077}
{0.0, 0.0, 0.0, -1.6387096774193548, 0.5075268817204301}

Answers{0.03849518810148732, -0.18022747156605426, -0.30971128608923887, 0.24759405074365706}

```

Figure 6: Proof Total Pivot

Gaussian Elimination

Eliminaci3n Gaussiana Simple

Etapas 0

```
[[ 2.  -1.   0.   3.   1. ]  
[ 1.   0.5  3.   8.   1. ]  
[ 0.  13.  -2.  11.   1. ]  
[14.   5.  -2.   3.   1. ]]
```

Etapas 1

```
[[ 2.  -1.   0.   3.   1. ]  
[ 0.   1.   3.   6.5  0.5]  
[ 0.  13.  -2.  11.   1. ]  
[ 0.  12.  -2. -18.  -6. ]]
```

Etapas 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. ]]
```

Etapas 3

```
[[ 2.      -1.      0.      3.      1.      ]  
[ 0.      1.      3.      6.5     0.5     ]  
[ 0.      0.     -41.     -73.5    -5.5     ]  
[ 0.      0.      0.     -27.878049 -6.902439]]
```

Despu3s de sustituci3n regresiva:

x:

```
[[ 0.03849519]  
[-0.18022747]  
[-0.30971128]  
[ 0.24759405]]
```

Figure 7: Proof of Gaussian Elimination

Muller Method


```

Muller method
iter|      x0      |      x1      |      x2      |      Root      |      E
0 | -1.000000e+00 | 0.000000e+00 | 1.000000e+00 | -2.220446e-16 |
1 | -1.000000e+00 | 0.000000e+00 | 1.000000e+00 | -2.220446e-16 |
2 | 0.000000e+00 | 1.000000e+00 | -2.220446e-16 | -2.220446e-16 | 0.000000e+00
A root aproximation was found at -2.220446049250313e-16

```

Figure 8: Proof of Muller Method

Fixed Point

```

iter xi g(xi) f(xi) E
0 -0.5 -0.6842068330717285 -0.2931087267313766 0.10000010000000001
1 -0.6842068330717285 -0.6999584211588118 -0.16388637436010217 0.18420683307172847
2 -0.6999584211588118 -0.6964329939682882 -0.15288781535493412 0.015751588087083324
3 -0.6964329939682882 -0.6972792521639637 -0.15534454046026097 0.0035254271905236223
4 -0.6972792521639637 -0.697079089789521 -0.1547545467399959 0.00084625819567552
5 -0.697079089789521 -0.6971266036840224 -0.15489408050278758 0.00020016237444264728
6 -0.6971266036840224 -0.6971153345307346 -0.1548609575705832 4.751389450130539e-05
7 -0.6971153345307346 -0.6971180078402816 -0.1548688134852106 1.1269153287751799e-05
8 -0.6971180078402816 -0.6971173736982946 -0.15486694987383254 2.6733095469522183e-06
9 -0.6971173736982946 -0.6971175241262941 -0.15486739194527666 6.34141986921577e-07
10 -0.6971175241262941 -0.69711748844261 -0.15486728707928488 1.5042799950126806e-07
-0.69711748844261 is an aproximation with tolerance of 1e-07 and after 11 iterations

```

Figure 9: Proof of Fixed Point

Secant

```

Iter xi f(xi) E
0 0.5 -0.2931087267313766
1 1 0.03536607938024017
2 0.946166222306525 0.005619392737863826 0.05383377769347497
3 0.9359965807911726 -0.00023632217470054284 0.010169641515352379
4 0.9364070023767038 1.4022358909571153e-06 0.00041042158553117325
5 0.9364045814731196 3.4371649970665885e-10 2.420903584265943e-06
6 0.9364045808795615 -4.996003610813204e-16 5.935580915661376e-10
0.9364045808795624 is the root

```

Figure 10: Proof of Secant

Newton

```
Iter xi f(xi) E
0 0.5 -0.2931087267313766 1.0000001
1 0.9283919899125719 -0.004662157097372055 0.4283919899125719
2 0.9363667412673313 -2.1912619882713535e-05 0.007974751354759446
3 0.9364045800189902 -4.98339092214195e-10 3.783875165885853e-05
0.9364045808795621 its an aproximation to a root with a tolerance of 1e-07
```

Figure 11: Proof of Newton