

UNIVERSIDAD EAFIT
IT and Systems Department
Project Choice
First Delivery

Course: Numerical Analysis

Teacher: Edwar Samir Posada Murillo

Semester: 2022-1

Delivery date of this report: 06-04-2020

Name of the project: Numerical Methods

Project web address (repository): <https://github.com/alejo128/Metodos-Numerico>

Members:

José Alejandro Díaz Urrego

Project description: the development of a web page will be carried out where the different methods that will be seen throughout the semester are gathered

For this first installment, a document will be produced in which the codes implemented for the following methods will be evidenced:

- Solution of equations in one variable
 - o Bisection
 - o Incremental searches
 - o False rule
 - o Fixed point
 - o Newton
 - o Secant
 - o Multiple roots
- Solution of systems of linear equations
 - o Gaussian elimination
 - o Partial pivot
 - o Full pivot

Incremental Search Method

- Pseudocode:

```
1  Process Incremental search method
2      Read Xo, Delta, Iter
3      Yo = f(Xo)
4
5      If Yo = 0 Then
6          Show: 'Xo is Root'
7      Else If
8          X1 = Xo + Delta
9          Count = 1
10         Y1 = f(X1)
11         While Yo * Y1 > 0 & Count < Iter Do
12             Xo = X1
13             Yo = Y1
14             X1 = Xo + Delta
15             Y1 = f(X1)
16             Count = Count + 1
17         End While
18         If Y1 = 0 Then
19             Show: 'Xo is Root'
20         Else If Yo * Y1 < 0 Then
21             Show: 'There's a Root Between Xo and X1'
22         Else If
23             Show: 'Fail in 'Iter' Iterations'
24         End If
25     End If
26 End If
27 End Process
```

Bisection Method

- Pseudocode

```
1 Process Bisection method
2   Read Xi, Xs, Tol, Iter
3   Yi = f(Xi)
4   Ys = f(Xs)
5
6   If Yi = 0 Then
7     Show: 'Xi is Root'
8   Else If
9     If Ys = 0 Then
10      Show: 'Xs is Root'
11    Else If
12      If Yi * Ys < 0 Then
13        Xm = (Xi + Xs) / 2
14        Count = 1
15        Ym = f(Xm)
16        Error = Tol + 1
17        While Error > Tol & Ym ≠ 0 & Count < Iter Do
18          If Yi * Ym < 0 Then
19            Xs = Xm
20            Ys = Ym
21          Else If
22            Xi = Xm
23            Yi = Ym
24          End If
25          Xaux = Xm
26          Xm = (Xi + Xs) / 2
27          Ym = f(Xm)
28          Error = Abs(Xm - Xaux)
29          Count = Count + 1
30        End While
31        If Ym = 0 Then
32          Show: 'Xm is Root'
33        Else If Error < Tol Then
34          Show: 'Xm is approximation to a root with a tolerance 'Tol''
35        Else If
36          Show: 'Fail in 'Iter' iterations'
37        End If
38      End If
39    Else If
40      Show: 'The interval is inadequate'
41    End If
42  End If
43 End If
44 End Process
```

False Rule Method

- Pseudocode

```
1  Process False rule method
2      Read Xi, Xs, Tol, Iter
3      Yi = f(Xi)
4      Ys = f(Xs)
5
6      If Yi = 0 Then
7          Show: 'Xi is Root'
8      Else If
9          If Ys = 0 Then
10             Show: 'Xs is Root'
11         Else If
12             If Yi * Ys < 0 Then
13                  $X_m = X_i - ((Y_i * (X_s - X_i)) / (Y_s - Y_i))$ 
14                 Count = 1
15                 Ym = f(Xm)
16                 Error = Tol + 1
17
18                 While Error > Tol & Ym ≠ 0 & Count < Iter Do
19                     If Yi * Ym < 0 Then
20                         Xs = Xm
21                         Ys = Ym
22                     Else If
23                         Xi = Xm
24                         Yi = Ym
25                     End If
26                     Xaux = Xm
27                      $X_m = X_i - ((Y_i * (X_s - X_i)) / (Y_s - Y_i))$ 
28                     Ym = f(Xm)
29                     Error = Abs(Xm - Xaux)
30                     Count = Count + 1
31                 End While
32                 If Ym = 0 Then
33                     Show: 'Xm is Root'
34                 Else If Error < Tol Then
35                     Show: 'Xm is approximation to a root with a tolerance 'Tol''
36                 Else If
37                     Show: 'Fail in 'Iter' iteration'
38                 End If
39             End If
40         Else If
41             Show: 'The interval is inadequate'
42         End If
43     End If
44 End If
45 End Process
```

Fixed Point Method

- Pseudocode

```
1  Process Fixed point method
2      Read Xo, Tol, Iter
3      Yo = f(Xo)
4      Count = 0
5      Error = Tol + 1
6
7      While Yo ≠ 0 & Error > Tol & Count < Iter Do
8          Xn = g(Xo)
9          Yo = f(Xn)
10         Error = abs((Xn - Xo) / Xn)
11         Xo = Xn
12         Count = Count + 1
13     End While
14     If Yo = 0 Then
15         Show: 'Xo is Root'
16     Else If Error < Tol Then
17         Show: ''Xo' is approximation to a root with a tolerance 'Tol''
18     Else If
19         Show: 'Failure in 'Iter' iterations'
20     End If
21 End If
22 End Process
```

Newton's Method

- Pseudocode

```
1  Process Newton's method
2      Read Xo, Tol, Iter
3      Yo = f(Xo)
4      Bo = f'(Xo)
5      Count = 0
6      Error = Tol + 1
7
8      While Yo ≠ 0 & Bo ≠ 0 & Error > Tol & Count < Iter Do
9          X1 = Xo - (Yo / Bo)
10         Yo = f(X1)
11         Bo = f'(X1)
12         Error = abs((X1 - Xo) / X1)
13         Xo = X1
14         Count = Count + 1
15     End While
16     If Yo = 0 Then
17         Show: 'Xo is Root'
18     Else If Error < Tol Then
19         Show: ''Xo' is approximation to a root with a tolerance 'Tol''
20     Else If Do = 0 Then
21         Show: ''Xo' is possibly a multiple root'
22     Else If
23         Show: 'Failure in 'Iter' iterations'
24     End If
25     End If
26 End If
27 End Process
```

Secant Method

- Pseudocode

```
1  ∨ Process Secant method
2      Read X1, Xo, Tol, Iter
3      Yo = f(Xo)
4  ∨  If Yo = 0 Then
5      Show: 'Xo is root'
6  ∨  Else If
7      Yi = f(X1)
8      Count = 0
9      Error = Tol + 1
10     Aux = Y1 - Yo
11
12  ∨  While Error > Tol & Y1 ≠ 0 & Aux ≠ 0 & Count < Iter Do
13      X2 = X1 - (( Y1 * (X1 - Xo)) / Aux)
14      Error = Abs((X2- X1) / X2)
15      Xo = X1
16      Yo = Y1
17      X1 = X2
18      Y1 = f(X1)
19      Aux = Y1 - Yo
20      Count = Count + 1
21  End While
22
23  ∨  If Y1 = 0 Then
24      Show: 'X1 is Root'
25  ∨  Else If Error < Tol Then
26      Show: ''X1' is approximation to a root with a tolerance 'Tol''
27  ∨  Else If Aux = 0 Then
28      Show: 'There's possibly a multiple root'
29  ∨  Else If
30      Show: 'Failure in 'Iter' iterations'
31  End If
32  End If
33  End If
34  End If
35  End Process
```

Multiple Roots Method

- Pseudocode

```
1  Process Multiple roots method
2      Read Xo, Tol, Iter
3      Yo = f(Xo)
4      D1o = f'(Xo)
5      D2o = f''(Xo)
6      Deno = D1o^2 - (Yo * D2o)
7      Count = 0
8      Error = Tol + 1
9
10     While Yo ≠ 0 & Deno ≠ 0 & Error > Tol & Count < Iter Do
11         X1 = Xo - ((Yo * D1o) / Deno)
12         Yo = f(X1)
13         D1o = f'(X1)
14         D2o = f''(X1)
15         Error = abs((X1 - Xo) / X1)
16         Deno = D1o^2 - (Yo * D2o)
17         Xo = X1
18         Count = Count + 1
19     End While
20     If Yo = 0 Then
21         Show: 'Xo is Root'
22     Else If Error < Tol Then
23         Show: ''Xo' is an approximate root with a tolerance 'Tol''
24     Else If Deno = 0 Then
25         Show: 'The denominator becomes zero'
26     Else If
27         Show: 'Failure in 'Iter' iterations'
28     End If
29     End If
30 End If
31 End Process
```


Simple Gaussian Elimination Method

- Pseudocode

```
1  Process Simple gaussian elimination method
2      Read A, b
3      (n,m) = size(A)
4      a = AugmentedMatrixForm (A,b)
5
6      If n = m Then
7          for k = 1 to n-1
8              for i = k + 1 to n
9                  M(ik) = a(ik) / a(kk)
10                 for j = k to n + 1
11                     a(ij) = a(ij) - m(ik)a(kj)
12                 End for
13             End for
14         End for
15
16         for i = n to 1
17             sum = 1
18             for p = i + 1 to n
19                 sum = sum + a(ipXp)
20             End for
21             Xi = (bi - sum) / a(ii)
22         End for
23     Else If
24         show: 'The matrix is not square'
25     End If
26     print a
27     print x
28 End Process
```

Gaussian Elimination Method with Partial Pivoting

- Pseudocode

```
1 Process partial pivoting method
2   Read A, b
3   (n,m) = size(A)
4   a = AugmentedMatrixForm (A,b)
5
6   If n = m Then
7     for k = 1 to n-1
8       higher = 0
9       filam = k
10      for p = k to n
11        If higher < |a(pk)| Then
12          higher = |a(pk)|
13          filam = p
14        End If
15      End for
16      If higher = 0 Then
17        Show: 'Suspended the process, infinite solutions'
18      Else If
19        If filam ≠ k Then
20          for j = 1 to n+1
21            Aux = a(k,j)
22            a(k,j) = a(filam,j)
23            a(filam, j) = Aux
24          End for
25        End If
26      End If
27      for i=k+1 to n
28        m(ik) = a(ik) / a(kk)
29        for j=k to n+1
30          a(ij) = a(ij) - m(ik)a(kj)
31        End for
32      End for
33    End for
34    for i = n to 1
35      sum = 0
36      for p = i + 1 to n
37        sum = sum + a(ip X p)
38      End for
39      Xi = (bi - sum) / a(ii)
40    End for
41  Else If
42    show: 'The matrix is not square'
43  End If
44  print a
45  print x
46 End Process
```

Gaussian Elimination Method with Total Pivoting

- Pseudocode

```
1  Process total pivoting method
2  Read A, b
3  (n,m) = size(A)
4  a = AugmentedMatrixForm (A,b)
5
6  If n = m Then
7      for i = 1 to n
8          mark(i) = i
9      End for
10     for k = 1 to n
11         higher = 0
12         filem = k
13         columnm = k
14         for p = k to n
15             for r = k to n
16                 If higher < |a(pr)| Then
17                     higher = |a(pr)|
18                     filem = p
19                     columnm = r
20                 End If
21             End for
22         End for
23     If higher = 0 Then
24         Show: 'Suspended the process, infinite solutions'
25     Else If
26         If filem ≠ k Then
27             for j = 1 to n+1
28                 Aux = a(k,j)
29                 a(k,j) = a(filem,j)
30                 a(filem,j) = Aux
31             End for
32             Aux = mark(k)
33             mark(k) = mark(columnm)
34             mark(columnm) = Aux
35         End If
36     End If
37     for i = k + 1 to n
38         m(ik) = a(ik) / a(kk)
39         for j = k to n+1
40             a(ij) = a(ij) - m(ik)a(kj)
41         End for
42     End for
43     for i = n to 1
44         sum = 0
45         for p = i + 1 to n
46             sum = sum + a(ip X p)
47         End for
48         x(i) = (b(i) - sum) / a(ii)
49     End for
50     for i = n to 1
```

```
51         for j = 1 to n
52             If mark(j) = i Then
53                 k = j
54             End If
55         End for
56         Aux = x(k)
57         x(k) = x(i)
58         x(i) = Aux
59         Aux = mark(k)
60         mark(k) = mark (i)
61         mark(i) = Aux
62     End for
63 End for
64 Else If
65     Show: 'The matrix is not square'
66 End If
67 print a
68 print x
69 End Process
```