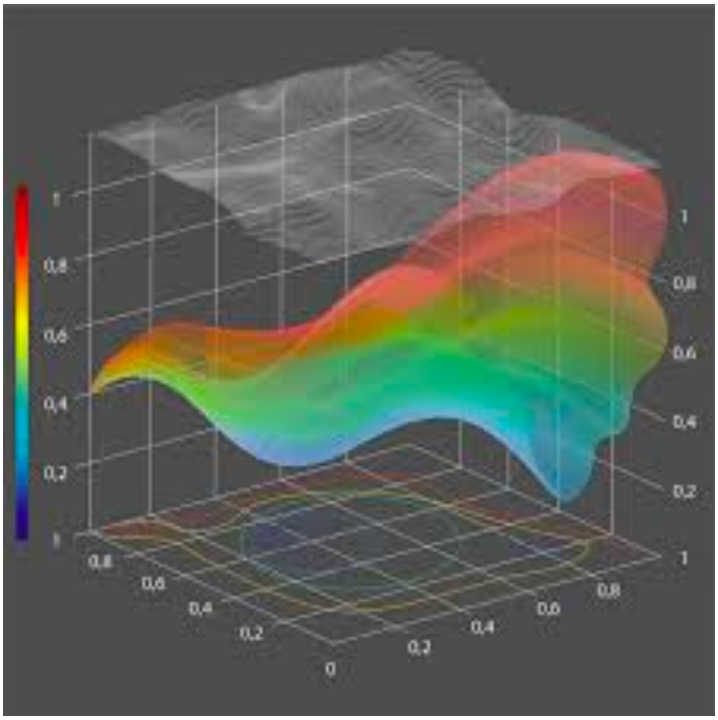
Final project - Matlab

Numerical analysis

- December 30, 2020

By :

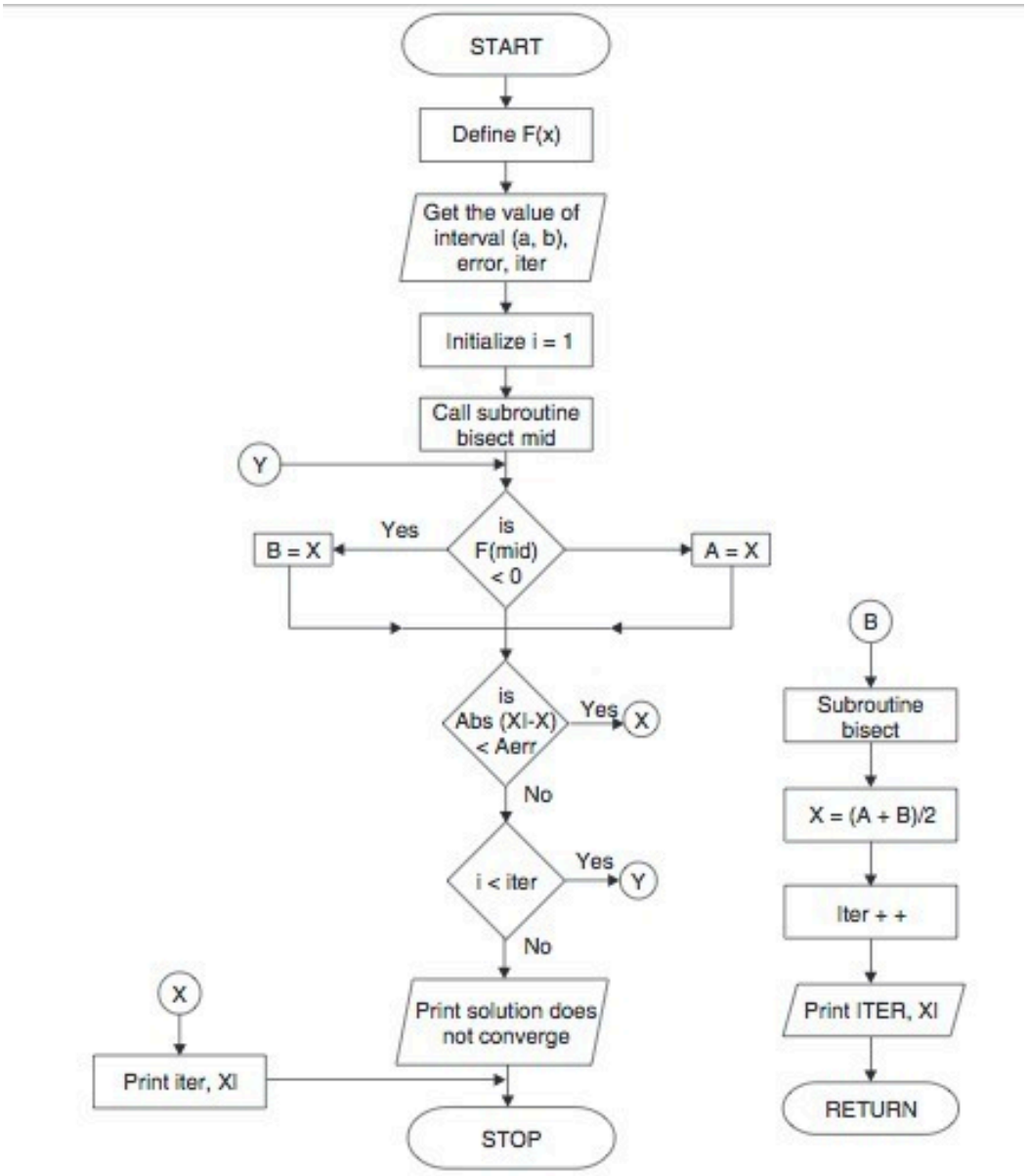
Yara Gamal Salem. ID:5895
Nadine Tarek Elghamry. ID:5345
Salma Saad Salem. ID: 5553
Maryam Yasser Zaki. ID: 5787
Malak Mohamed Kassem. ID: 5979



Bisection:

The algorithm we used:

```
1. Start

2. Define function f(x)

3. Input
    a. Lower and Upper guesses x0 and x1
    b. tolerable error e

4. If f(x0)*f(x1) > 0
    print "Incorrect initial guesses"
    goto 3
   End If

5. Do
    x2 = (x0+x1)/2

    If f(x0)*f(x2) < 0
        x1 = x2
    Else
        x0 = x2
    End If

   while abs(f(x2) > e

6. Print root as x2

7. Stop
```

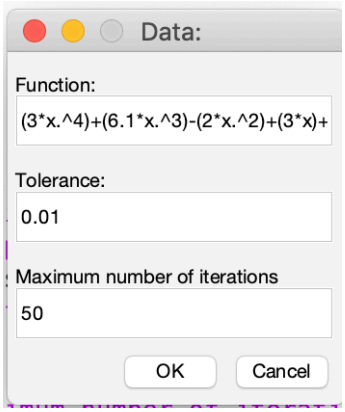

START
Define F(x)
Get the value of interval (a, b), error, iter
Initialize i = 1
Call subroutine bisect mid
Y
is F(mid) < 0
Yes
B = X
A = X
is Abs (XI-X) < Aerr
Yes
X
No
i < iter
Yes
Y
No
X
Print iter, XI
Print solution does not converge
STOP
B
Subroutine bisect
X = (A + B)/2
Iter + +
Print ITER, XI
RETURN

Examples:

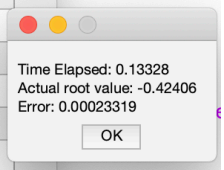
1)f(x) = 3x^4 + 6.1x^3 - 2x^2 + 3x +2

Tolerance : 10^-2

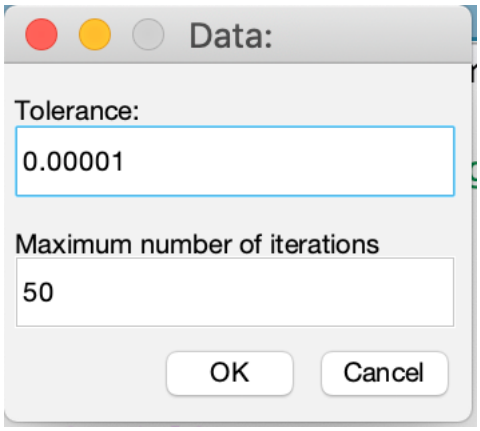
Interval [-1,0]

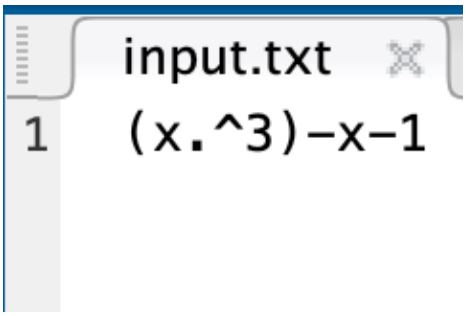


Step	X-Lower	X-Upper	X-Root	Error %
1	-1	0	-0.5	-
2	-0.5	0	-0.25	100
3	-0.5	-0.25	-0.375	33.33333333
4	-0.5	-0.375	-0.4375	14.28571429
5	-0.4375	-0.375	-0.40625	7.692307692
6	-0.4375	-0.40625	-0.421875	3.703703704
7	-0.4375	-0.421875	-0.4296875	1.818181818
8	-0.4296875	-0.421875	-0.4257812500	0.9174311927

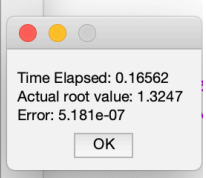


2)$f(x)=x^3-x-1$
Tolerance: 10^-5
Interval [1,2]

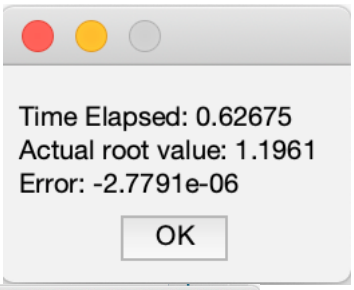




Step	X-Lower	X-Upper	X-Root	Error %
1	1	2	1.5	-
2	1	1.5	1.25	20
3	1.25	1.5	1.375	9.090909091
4	1.25	1.375	1.3125	4.761904762
5	1.3125	1.375	1.34375	2.325581395
6	1.3125	1.34375	1.328125000	1.176470588
7	1.3125	1.328125000	1.320312500	0.5917159763
8	1.320312500	1.328125000	1.324218750	0.2949852507



3)$f(x) = x^2 - sin(x) - 0.5$
Tolerance : 10^-3
Interval [0, 2]



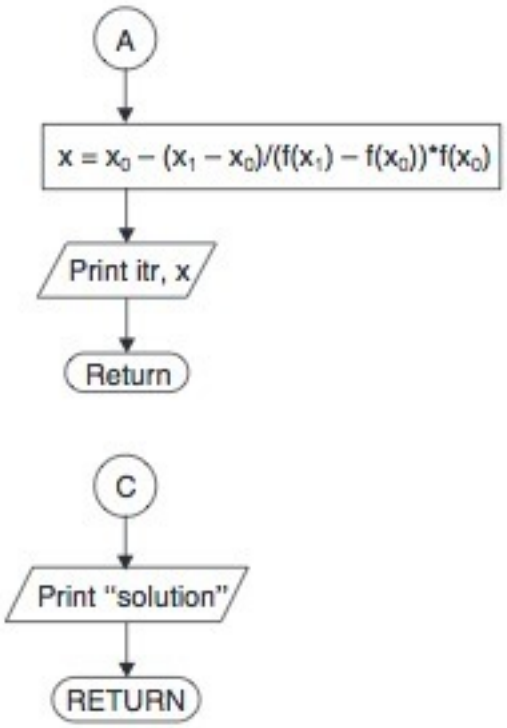
Step	X-Lower	X-Upper	X-Root	Error %
1	0	2	1	-
2	1	2	1.5	33.33333333
3	1	1.5	1.25	20
4	1	1.25	1.125	11.11111111
5	1.125	1.25	1.1875	5.263157895
6	1.1875	1.25	1.21875	2.564102564
7	1.1875	1.21875	1.203125000	1.298701299
8	1.1875	1.203125000	1.195312500	0.6535947712

Regula Falsi:

The algorithm we used:

```
1. Start

2. Define function f(x)

3. Input
    a. Lower and Upper guesses x0 and x1
    b. tolerable error e

4. If f(x0)*f(x1) > 0
    print "Incorrect initial guesses"
    goto 3
   End If

5. Do

    x2 = x0 - ((x0-x1) * f(x0))/(f(x0) - f(x1))

    If f(x0)*f(x2) < 0
        x1 = x2
    Else
        x0 = x2
    End If
```

```
   While abs(f(x2) > e
6. Print root as x2
7. Stop
```

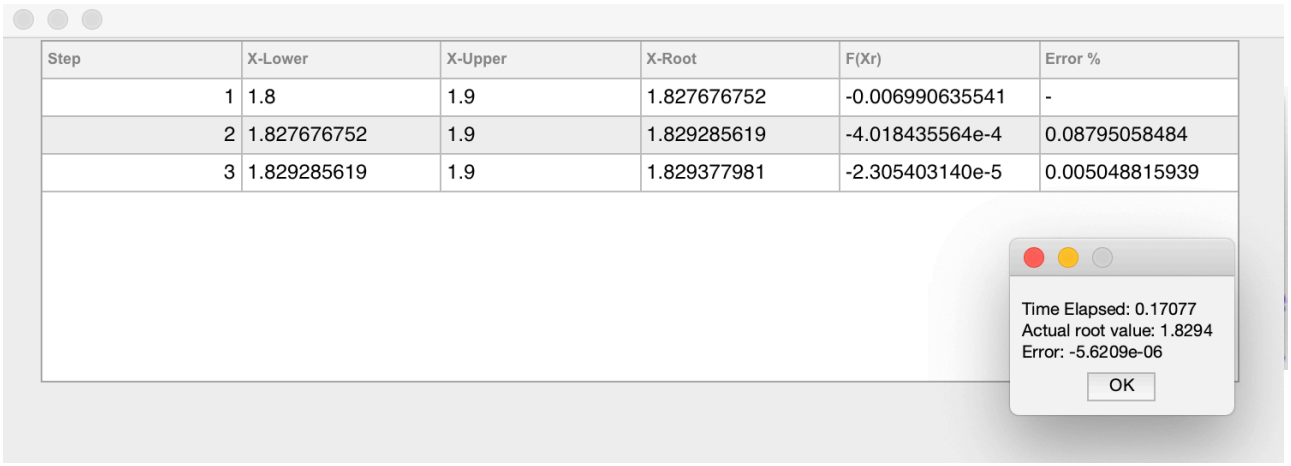


Examples:

1) $f(x)=2x^3-2x-5$

Tolerance : 10^-3

Interval [1,2]

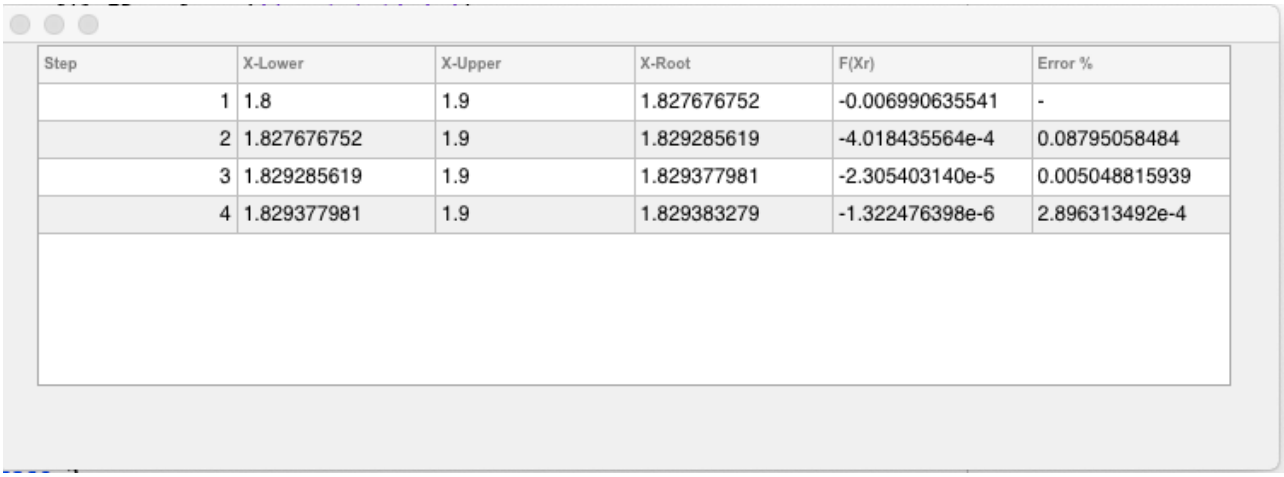


Step	X-Lower	X-Upper	X-Root	F(Xr)	Error %
1	1.8	1.9	1.827676752	-0.006990635541	-
2	1.827676752	1.9	1.829285619	-4.018435564e-4	0.08795058484
3	1.829285619	1.9	1.829377981	-2.305403140e-5	0.005048815939

2) f(x) = e^x + 2^-x + 2 cos (x) -6

Tolerance : 10^−3

two initial guesses [1.8,1.9]



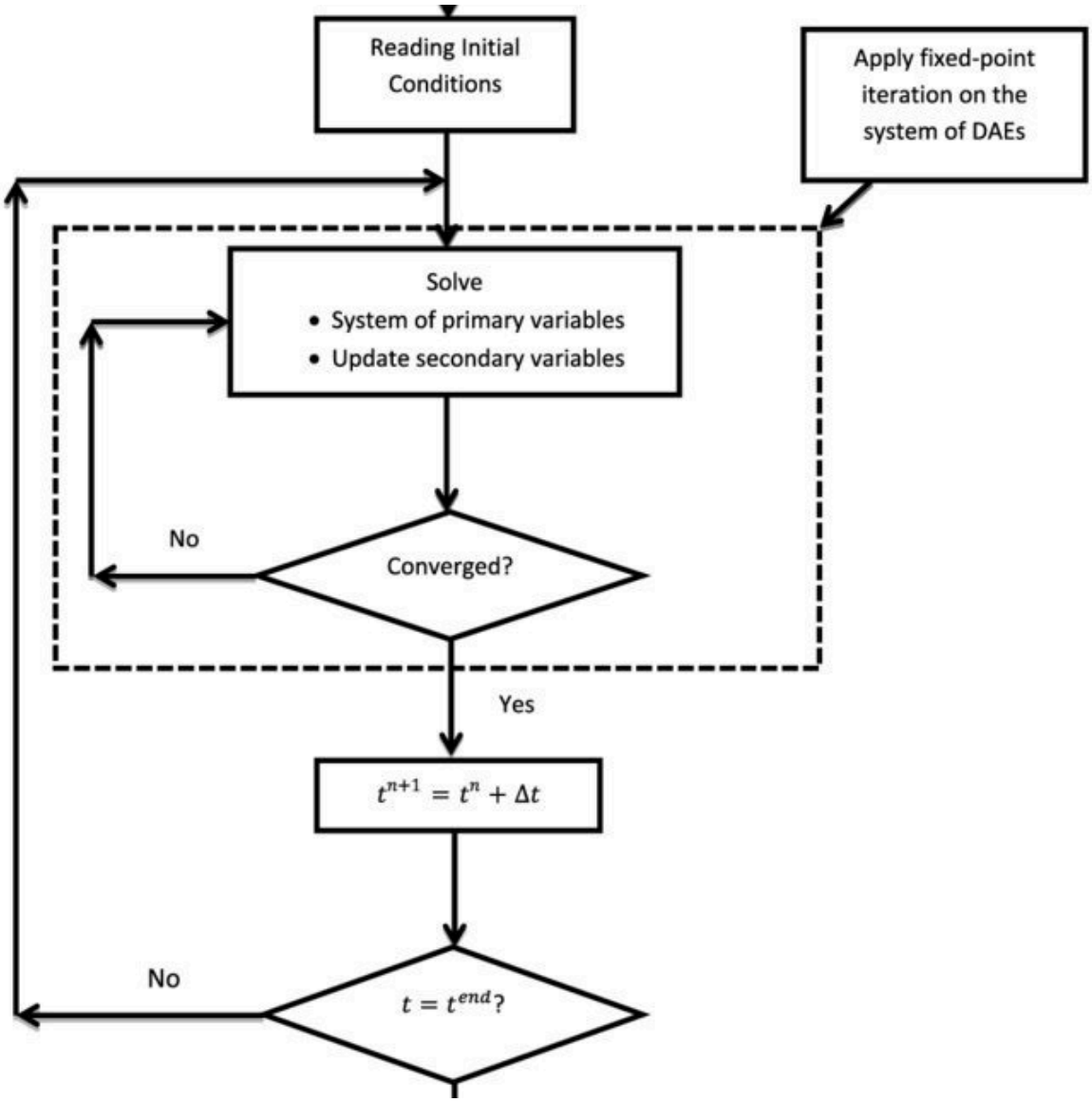
Step	X-Lower	X-Upper	X-Root	F(Xr)	Error %
1	1.8	1.9	1.827676752	-0.006990635541	-
2	1.827676752	1.9	1.829285619	-4.018435564e-4	0.08795058484
3	1.829285619	1.9	1.829377981	-2.305403140e-5	0.005048815939
4	1.829377981	1.9	1.829383279	-1.322476398e-6	2.896313492e-4

Fixed point Approximation

The algorithm we used:

```
1. Start

2. Define function as f(x)

3. Define convergent form g(x)

4. Input:
    a. Initial guess x0
    b. Tolerable Error e
    c. Maximum Iteration N

5. Initialize iteration counter: step = 1

6. Do

    x1 = g(x0)

    step = step + 1

    If step > N
        Print "Not Convergent"
        Stop
    End If

    x0 = x1

   While abs f(x1) > e

7. Print root as x1

8. Stop
```

Note: g(x) is obtained by rewriting f(x) in the form of x = g(x)

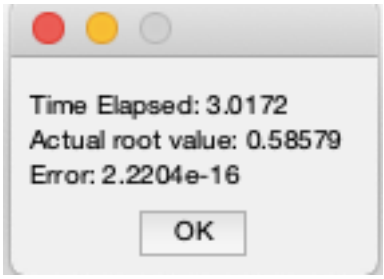


Examples:

1) f(x) = x^3− 7x^2+ 14x − 6

Xo = 0.5

Interval = [0,1]

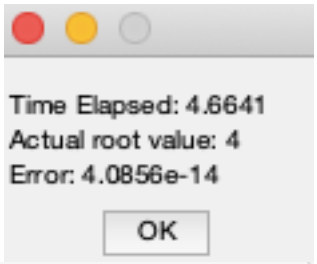


Step	X(i)	F(Xi)	F'(Xi)	X(i+1)	Error%
1	0.5	-0.625	7.75	0.5806451613	13.88888889
2	0.5806451613	-0.03524554396	6.882414152	0.5857662632	0.8742568817
3	0.5857662632	-1.377620357e-4	6.828638661	0.5857864373	0.003443944206
4	0.5857864373	-2.133754506e-9	6.828427128	0.5857864376	5.334386937e-8

2)f(x) = x^2− 2x − 8

Interval=[-2,4]

Xo = 3.9



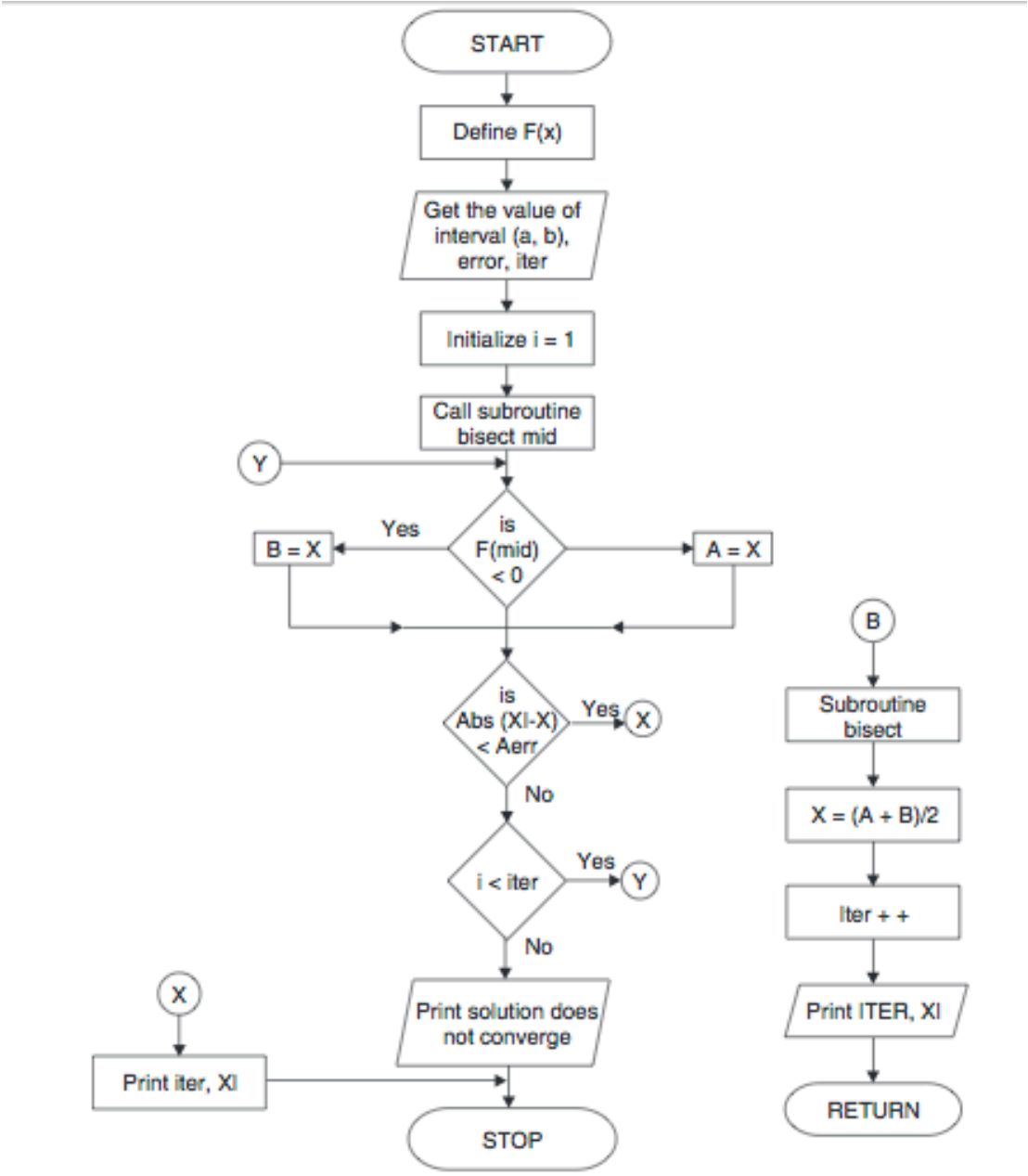
Step	X(i)	F(Xi)	F'(Xi)	X(i+1)	Error%
1	3.9	-0.59	5.8	4.001724138	2.542007755
2	4.001724138	0.01034780024	6.003448276	4.000000495	0.04309106401
3	4.000000495	2.970944413e-6	6.000000990	4	1.237893301e-5

Newton Rasphon:

The algorithm we used:

```
1. Start

2. Define function as f(x)

3. Define derivative of function as g(x)

4. Input:
    a. Initial guess x0
    b. Tolerable Error e
    c. Maximum Iteration N

5. Initialize iteration counter step = 1

6. Do
    If g(x0) = 0
        Print "Mathematical Error"
        Stop
    End If

    x1 = x0 - f(x0) / g(x0)
    x0 = x1

    step = step + 1

    If step > N
        Print "Not Convergent"
        Stop
    End If

   While abs f(x1) > e

7. Print root as x1
```

8. Stop



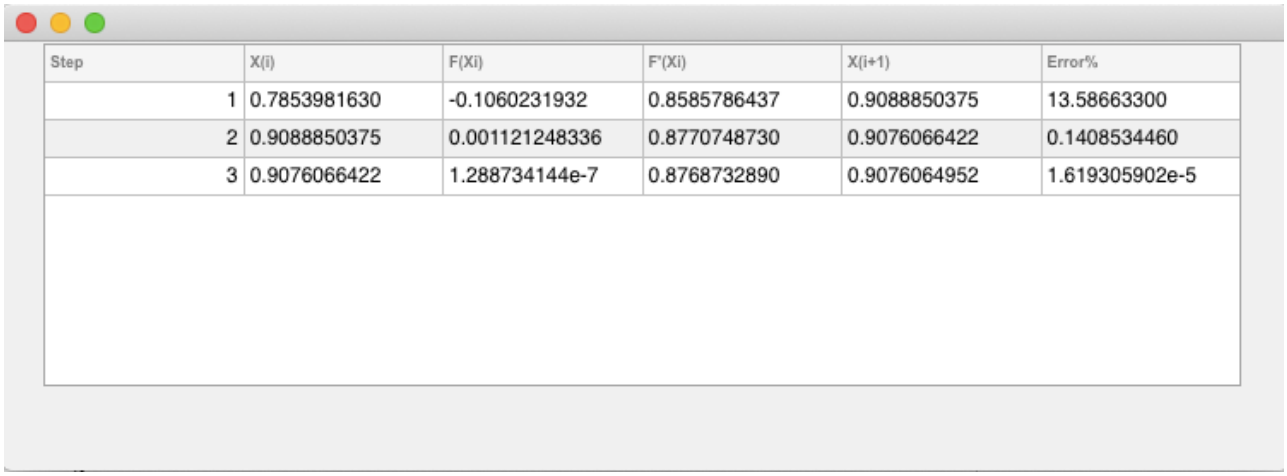
Examples:

1)f(x)=x − 0. 75 − 0. 2sin (x)

Tolerance : 10^−1

Interval [0, π/2]

Xo= π/4

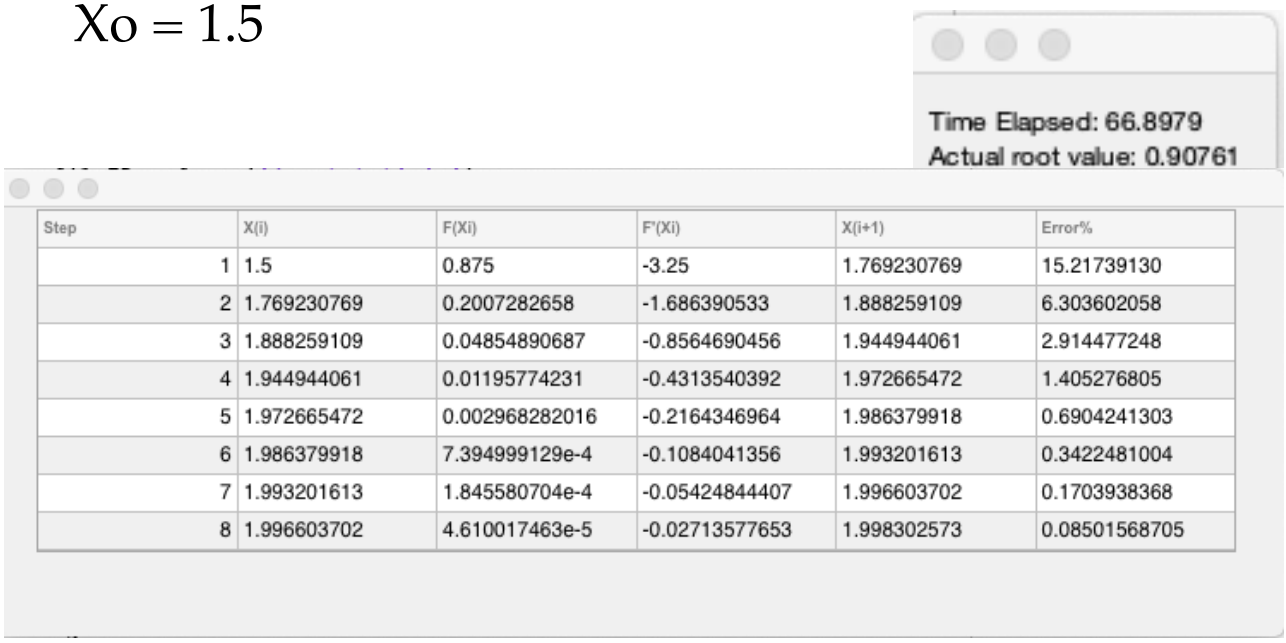


Step	X(i)	F(Xi)	F'(Xi)	X(i+1)	Error%
1	0.7853981630	-0.1060231932	0.8585786437	0.9088850375	13.58663300
2	0.9088850375	0.001121248336	0.8770748730	0.9076066422	0.1408534460
3	0.9076066422	1.288734144e-7	0.8768732890	0.9076064952	1.619305902e-5

2)f(x)=x3− 2x2− 4x + 8

Tolerance: 10^-1

Xo = 1.5



Time Elapsed: 66.8979
Actual root value: 0.90761

Step	X(i)	F(Xi)	F'(Xi)	X(i+1)	Error%
1	1.5	0.875	-3.25	1.769230769	15.21739130
2	1.769230769	0.2007282658	-1.686390533	1.888259109	6.303602058
3	1.888259109	0.04854890687	-0.8564690456	1.944944061	2.914477248
4	1.944944061	0.01195774231	-0.4313540392	1.972665472	1.405276805
5	1.972665472	0.002968282016	-0.2164346964	1.986379918	0.6904241303
6	1.986379918	7.394999129e-4	-0.1084041356	1.993201613	0.3422481004
7	1.993201613	1.845580704e-4	-0.05424844407	1.996603702	0.1703938368
8	1.996603702	4.610017463e-5	-0.02713577653	1.998302573	0.08501568705

Secant:

The algorithm we used:

```
1. Start

2. Define function as f(x)

3. Input:
    a. Initial guess x0, x1
    b. Tolerable Error e
    c. Maximum Iteration N


4. Initialize iteration counter step = 1


5. Do
    If f(x0) = f(x1)
        Print "Mathematical Error"
        Stop
    End If

    x2 = x1 - (x1 - x0) * f(x1) / ( f(x1) -
f(x0) )
    x0 = x1
    x1 = x2
    step = step + 1
    If step > N
        Print "Not Convergent"
        Stop
    End If

   While abs f(x2) > e

6. Print root as x2
```

7. Stop

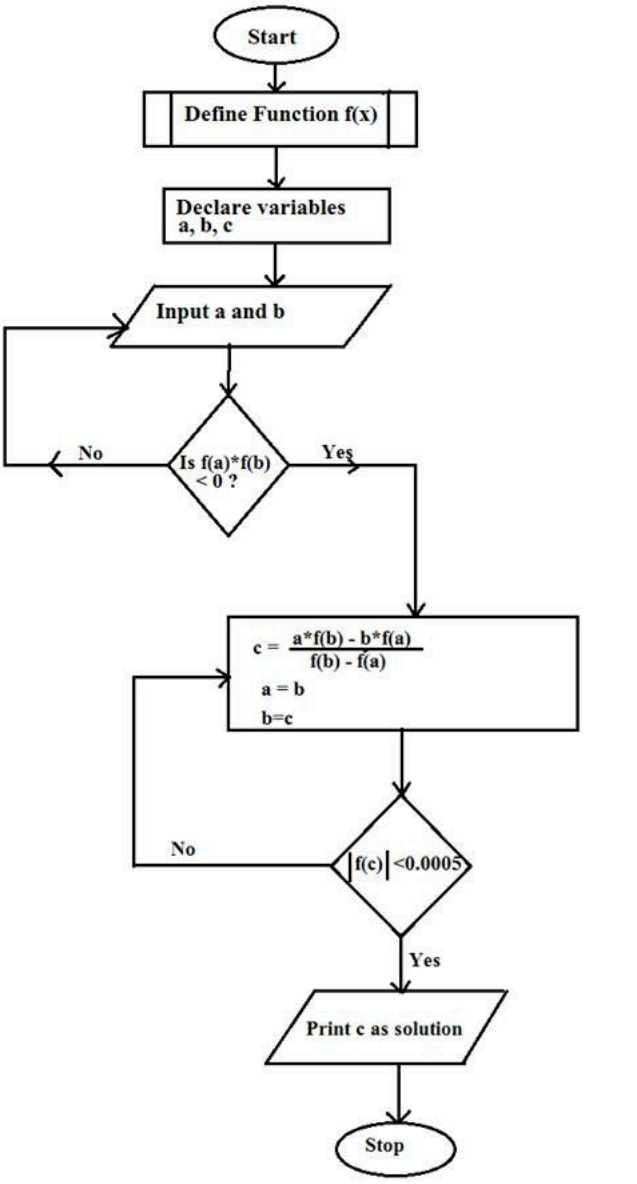


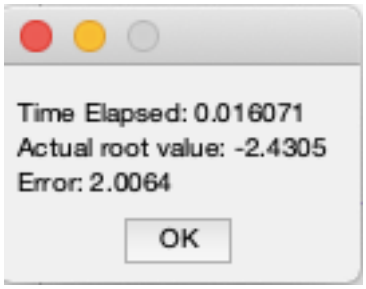
Fig. Flow Chart for Secant Method

Examples:

2)f(x) = 3x^4+ 6. 1x^3− 2x^2+3x + 2

Interval [-1, -3]

Tolerance : 10 ^-2

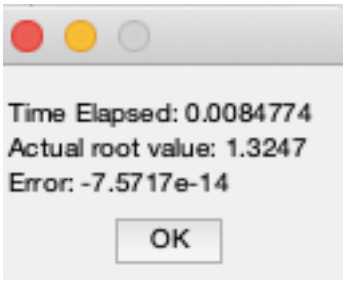


Step	X(i-1)	X(i)	F(X(i-1))	F(X(i))	X(i+1)	Error%
1	-1	-3	-6.1	53.3	-1.205387205	-148.8826816
2	-3	-1.205387205	53.3	-8.872216564	-1.461485415	-17.52314509
3	-1.205387205	-1.461485415	-8.872216564	-12.01165338	-0.4816399503	-203.4394083
4	-1.461485415	-0.4816399503	-12.01165338	-0.4289832965	-0.4453497568	-8.148695009
5	-0.4816399503	-0.4453497568	-0.4289832965	-0.1535174205	-0.4251251941	-4.757319253
6	-0.4453497568	-0.4251251941	-0.1535174205	-0.007531149433	-0.4240818480	-0.2460246971
7	-0.4251251941	-0.4240818480	-0.007531149433	-1.452235312e-4	-0.4240613336	-0.004837620113
8	-0.4240818480	-0.4240613336	-1.452235312e-4	-1.423944722e-7	-0.4240613134	-4.748035557e-6

2)f(x)=x^3-x-1

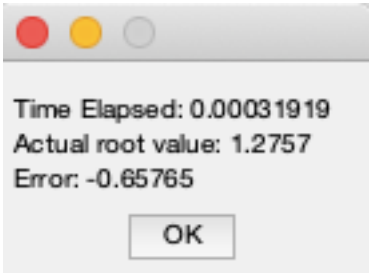
Interval [1,2]

Tolerance: 10^-4



Step	X(i-1)	X(i)	F(X(i-1))	F(X(i))	X(i+1)	Error%
1	1	2	-1	5	1.166666667	71.42857143
2	2	1.166666667	5	-0.5787037037	1.253112033	6.898454746
3	1.166666667	1.253112033	-0.5787037037	-0.2853630296	1.337206446	6.288812988
4	1.253112033	1.337206446	-0.2853630296	0.05388058668	1.323850096	1.008901951
5	1.337206446	1.323850096	0.05388058668	-0.003698115440	1.324707937	0.06475692647
6	1.323850096	1.324707937	-0.003698115440	-4.273426281e-5	1.324717965	7.570533496e-4
7	1.324707937	1.324717965	-4.273426281e-5	3.458221465e-8	1.324717957	6.121414230e-7

3) f(x) =x^5 - 5 x + 3
Interval [0.5,1.5]
Tolerance : 10^-3



Step	X(i-1)	X(i)	F(X(i-1))	F(X(i))	X(i+1)	Error%
1	0.5	1.5	0.53125	3.09375	0.2926829268	412.5
2	1.5	0.2926829268	3.09375	1.538733132	-0.9019914327	-132.4485263
3	0.2926829268	-0.9019914327	1.538733132	6.912905294	0.6347421246	242.1036036
4	-0.9019914327	0.6347421246	6.912905294	-0.07067518890	0.6191900830	2.511674858
5	0.6347421246	0.6191900830	-0.07067518890	-0.004933950793	0.6180228855	0.1888599141
6	0.6191900830	0.6180228855	-0.004933950793	4.741677498e-5	0.6180339959	0.001797692490
7	0.6180228855	0.6180339959	4.741677498e-5	-3.037805918e-8	0.6180339887	1.150973372e-6

Data Structures:

1- B=ArrayFun(func,A)

applies the function `func` to the elements of `A`, one element at a time. `arrayfun` then concatenates the outputs from `func` into the output array `B`, so that for the `i`th element of `A`, `B(i) = func(A(i))`. The input argument `func` is a function handle to a function that takes one input argument and returns a scalar. The output from `func` can have any data type, so long as objects of that type can be concatenated. The arrays `A` and `B` have the same size.

2-HPF decimal class

Very often I see people asking for a tool that offers more than 16 digits or so of accuracy
Good practices of numerical analysis are worth far more than any high precision tool. Even so, there are times when you will have a use for a bit of extra precision. And some of you will just want to play in the huge number sandbox. While some of you may use tools like that written by Ben Barrowes, HPF is written purely in MATLAB, so no compiles are needed. For all of you, whatever your reasons, I offer HPF, a High Precision Floating point tool.

3-Tic

`tic` works with the `toc` function to measure elapsed time. The `tic` function records the current time, and the `toc` function uses the recorded value to calculate the elapsed time.
`timerVal = tic` stores the current time in `timerVal` so that you can pass it explicitly to the `toc` function. Passing this value is useful when there are multiple calls to `tic` to time different parts of the same code. `timerVal` is an integer that has meaning only for the `toc` function.

4-Toc

`toc` reads the elapsed time since the stopwatch timer started by the call to the `tic` function. MATLAB® reads the internal time at the execution of the `toc` function and displays the elapsed time since the most recent call to the `tic` function without an output. The elapsed time is expressed in seconds.
`toc(timerVal)` displays the elapsed time since the call to the `tic` function corresponding to `timerVal`.

Functions used:

Fevel : evaluates a function using its name or its handle, and using the input arguments `x1,…,xM`.
fzero: Root of nonlinear function

Gui functions:

uifigure: Create figure for designing apps
uitable: Create table user interface component
msgbox: Create message dialog box
inputdlg: Create dialog box to gather user input
uiwait: Block program execution and wait to resume
num2str: Convert numbers to character array
str2double: Convert strings to double precision values