First Partial Project

Juan Carlos Aguilera Pérez
Jesús Amauri Guillén García
Marcopolo Iván Gil Melchor

Scilab solution of
Bisection
Secant
Newton Raphson
Bairstows
Numerical Methods

Ing. Raquel Landa

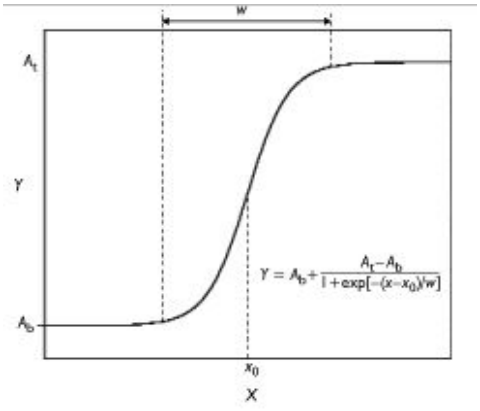
Abstract

Unlike the school, in the real world in most of the cases we end up having non linear equations that can't be solved analytically, for example in studies areas like Chemistry or Physic engineering, the phenomena that we shape with equations are nonlinear, and have to be solved by certain mathematical methods. As Zimmerman says in his book "Process Modelling and Simulation with Finite Element Method", there are only two algorithms that are really useful - the bisection method and Newton's Raphson method. In this project we worked with the example of this two methods mentioned by Zimmerman but we also include Secant and Bairstow method examples to have four options at the time of solving nonlinear equations.

This methods are in the engineering interest because of the possibility of describing invers function, because with most of this functions, the "forward direction", y = f(u), is well described, but the inverse function of u = f'(y), may be analytically indescribable, multivalued(non-unique), or even non-existent. But if it exist, then numerical description of an inverse function is identical to a root finding problem - find u such that F(u) = 0 is equivalent to F(u) - y = 0. Since the goal of most analysis is to find a solution to a set of constraints on a system, this is equivalent to inverting the set of constraints.

Showing some real world examples of use in the area of Chemistry and Physics we can demonstrate how useful this methods are, and how the methods allow the researchers to find an answer to the equations that otherwise will be impossible, if there were only analytical options.

In this project we are explaining the flowcharts of the methods that shows the process to follow in case of solving nonlinear equations, besides the flow charts we use the computational help to solve this methods, and like Zimmerman says ***"you better do it this way since you will have to plot the graph of the solution y(x) to make sense of it",*** going through code examples and showing how we implement the algorithms in ***Scilab*** for each of the 4 methods mentioned above , to show how does the method start solving the problem, and goes into a loop of actions until a predefined minimum error is achieved or N amounts of cycles have been done, we add the code and an output of it with graphical example showing the graph of the function and the root the method find.



Chemistry situations

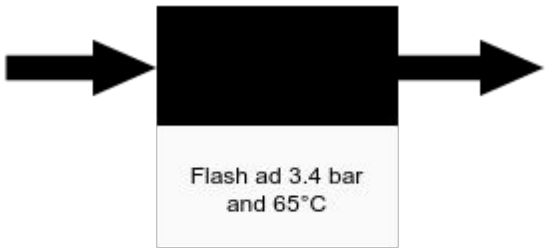
Root finding: Application to flash distillation

Chemical thermodynamics harbors many common applications of root finding, since the constraints of chemical equilibrium and mass conservation are frequently sufficient, along with constitutive models like equation of state to provide the same number of constraints as unknowns. We will take flash distillation as an example of simple root finding for one degree freedom of the system which is conveniently taken as the phase fraction.

A liquid hydrocarbon mixture undergoes a flash to 3.4 bar and 65°C. The composition of the liquid feed stream and the 'K' value of each component for the flash condition are given in the table. We want to determine composition of the vapour and liquid feed stream and fraction of feed leaving the flash as liquid. Table 1.1 gives the initial composition of the batch.

Table 1.1 charge to the flash unit

Component	**X1**	**Ki at 65°C and 3.4 bar**
Ethane	0.0079	16.2
Propane	0.1281	5.2
I-Butane	0.0849	2.6
n-Butane	0.2690	1.98
i-Pentane	0.0589	.91
n-Pentane	0.1361	.72
Hexane	0.3151	.28



A material balance for component i gives the relation

$$y_i = \frac{X_i}{1-\phi\left(1-\frac{1}{k_i}\right)}$$

Where X_i is the mole fraction in the feed(liquid) x_i is the mole fraction in the liquid product stream yi is the mole fraction in the vapour product, and f is the ratio of liquid product to feed molar flow rate. The definition of the equilibrium coefficient is $K_i = y_{i/X_i}$ ki=. Using this to eliminate xi from the balance relation results in a single equation between y_i adn X_i:

$$f(\phi) = 1 - \sum_{i=1}^{n} \frac{X_i}{1-\phi(1-\frac{1}{k_i})} = 0$$

Since the yi must sum to 1 we have a nonlinear equation for (ϕ)

$$f' = \sum_{i=1}^{n} \frac{Xi(1-\frac{1}{K_i})}{\left[1-\phi\left(1-\frac{1}{k_i}\right)\right]}$$

Where n is the number of components. This function $f(\phi)$ can be solved for the root(s) (ϕ), Which allows back substitution to find all the mole fractions in the product stream. The Newton-Raphson method requires the derivate $f''(\phi k_n)$ at the current stimate to determine the improved stimate.

Physical situations

Root finding: Orbital motion

Orbital Motion In two-dimensions, with a spherically symmetric potential (meaning here that $V(x,y,z) = V(r)$, a function of a single variable, $r = \sqrt{x^2 + y^2 + z^2}$, the distance to the origin) we can use circular coordinates to write the total energy of a test particle moving under the influence of this potential as:

$$E = 1/2\, m(\dot{r}^2 + r^2\dot{\Phi}^2) + V(r)$$

Conservation of momentum tells us that the z-component of angular momentum is conserved, with L $L_z = (r \times p)_z = mr^2\dot{\Phi}$, so we can rewrite the 1 of 13 3.1 energy as:

$$E = 1/2\, m\dot{r}^2 + 1/2\, L^2/mr^2 + V(r)$$

where $U(r)$ defines an "effective potential" – we have turned a two-dimensional problem into a one-dimensional problem for the coordinate r, and an effective potential that governs the motion in this setting.

Since we know the energy of the system in terms of r, we can invert the question before. to get:

$$\dot{r}^2 = 2\frac{E - U(r)}{m} = F(r)$$

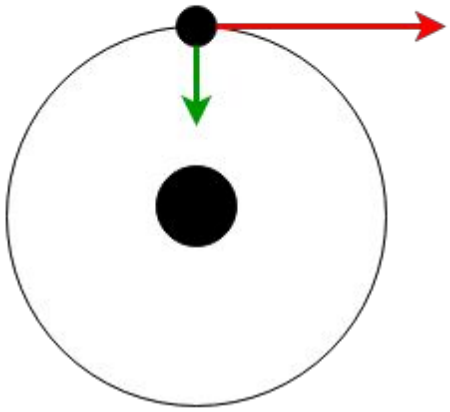
Now we can ask for the "turning points" of orbital motion (if/when they exist), those points at which $\dot{r} = 0$ – the answer is provided by radial locations r_i for which:

$$F(r_i)) = 0$$

precisely the sort of root-finding problem of interest.

In cases like Newtonian gravity, where $V(r) \approx 1/r$, the resulting $F(r)$ is just a polynomial (in fact, quadratic), so we don't need any fancy numerical solutions. But for more complicated potentials, root-finding can be used efficiently to isolate, at least numerically, the zeroes of the function $F(r)$.

Gurley. (2010). Curve Fitting Techniques. En Computer Methods(89-112). -: -.



Explanation Bisection:
On the Bisection method the program will receive as a first parameter the function to evaluate, this is submitted. It receive as parameters the xl, and xu that are the first two reference values that will help to start the iterations of the method, also you will receive as parameter the minimum error expected.

First the program calculate the initial Xm with the initial parameters then it will enter in a while function , the code will iterate until the error is achieved, in each iteration the program will evaluate the previous values and will solve the following equations.
The value of the variables will be updated with each iteration.

Estimation of the new Root.

$$Xm = \frac{(Xl + Xu)}{2}$$

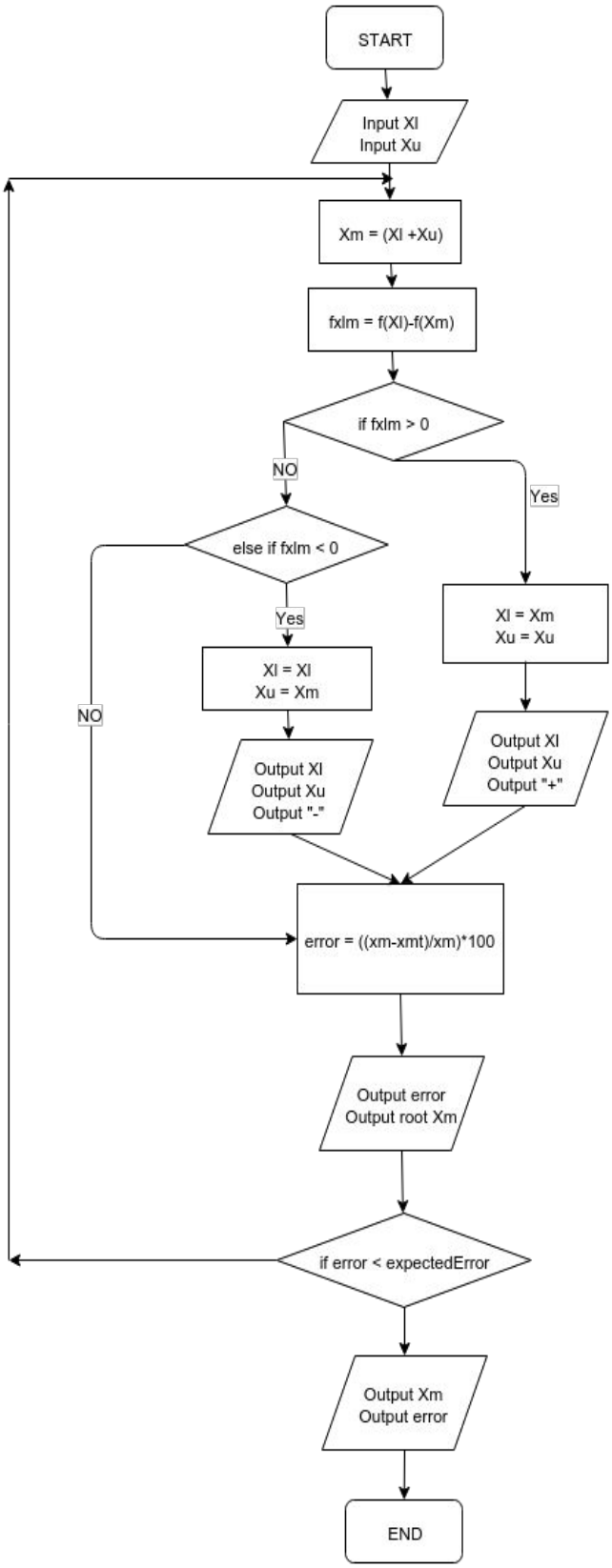
Inside the while there are two ifs were the result of the equations will be compared with 0 and will define the sign of the result, and display it on the screen. Finally the program will display the error on the screen by getting it with the following equation.

Absolute relativa approximate error.

$$|\in_a| = \left|\frac{X_m^{new} - X_m^{old}}{X_m^{new}}\right| \times 100$$

$$X_m^{new} = \textit{previous estimate of root}$$

$$X_m^{old} = \textit{current estimate of root}$$



Explanation Secant:
On the Secant method the program will receive as a first parameter the function to evaluate. It receive as parameters the x0, and x1 that are the first two reference values that will help to start the iterations of the method, also you will receive as parameter the minimum error expected.

With a while function , the code will iterate until the error is achieved, in each iteration the program will evaluate the previous values and it will enter in a comparison, if the initial values are equal then it will display, values x1 and x2 are equal and abort the equation, if that's not the case. The following function will be evaluated with the parameters first parameters or the updated ones.
The value of the variables will be updated with each iteration.

Estimation of the new root from two initial guesses.

$$X_{i+1} = \frac{f(X_i)(X_i - X_{i-1})}{f(X_i) - f(X_{i-1})}$$

Finally the program will display the error on the screen by getting it with the following equation.

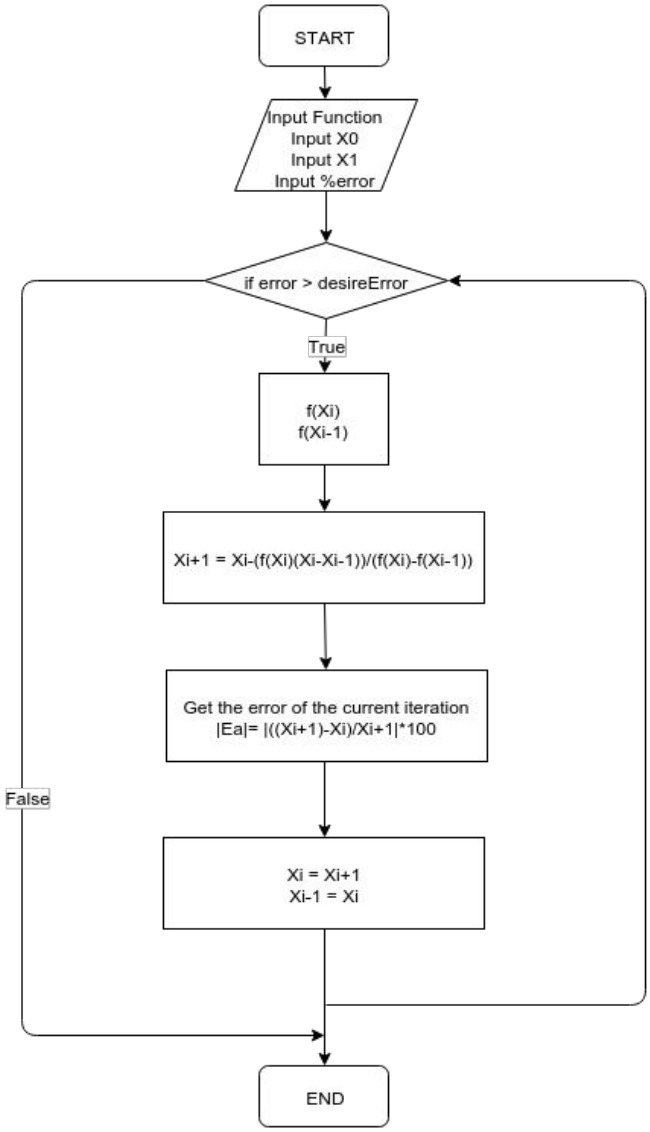
Absolute relativa approximate error.

$$X_{i+1} = \textit{previous estimate of root}$$

$$|\in_a| = \left|\frac{X_{i+1} - X_i}{X_{i+1}}\right| \times 100$$

$$X_i = \textit{current estimate of root}$$

$$X_{i+1} = \textit{new calculated estimate of root}$$



Explanation Newton Raphson:
On the Newton Raphson method the program will receive as a first parameter the function to evaluate. It receive as parameter the x1 is the first two reference values that will help to start the iterations of the method, also you will receive as parameter the minimum error expected.

With a while function , the program will derive the function that has received and use it as part of the following function, that will be evaluated with the parameter that it was initially entered for the first iteration. The subsequent iterations will use the new values that has been updated with the previous iterations.

Estimation of the new root from two initial guesses.

$$X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)}$$

Finally the program will display the error on the screen by getting it with the following equation.

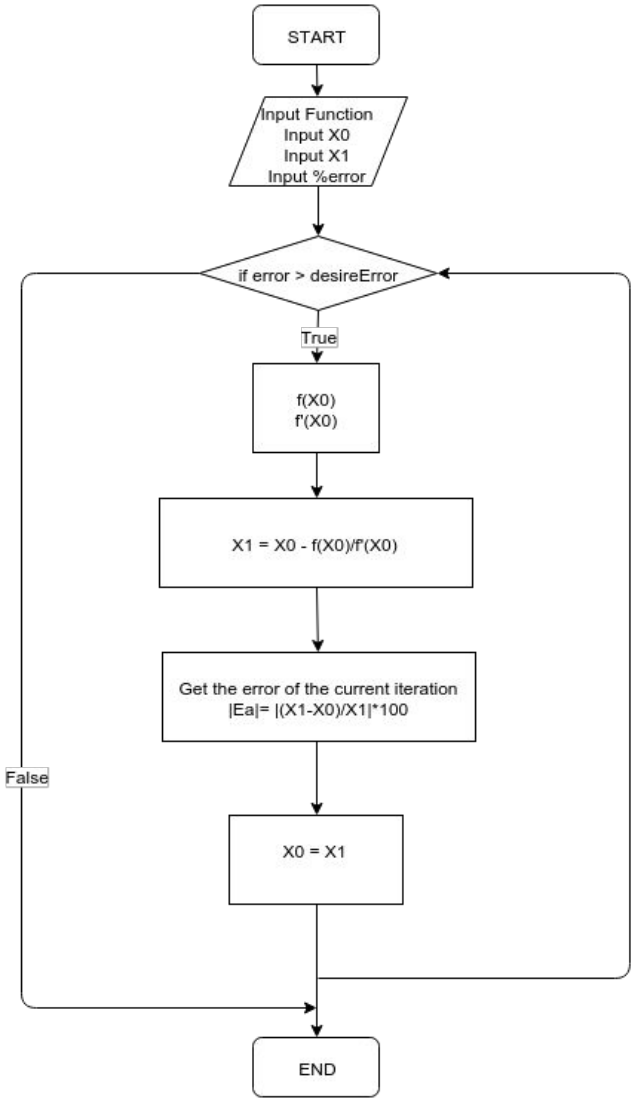
Absolute relativa approximate error.

$X_{i+1} =$ *previous estimate of root*

$$|\in_a| = \left|\frac{X_{i+1} - X_i}{X_{i+1}}\right| \times 100$$

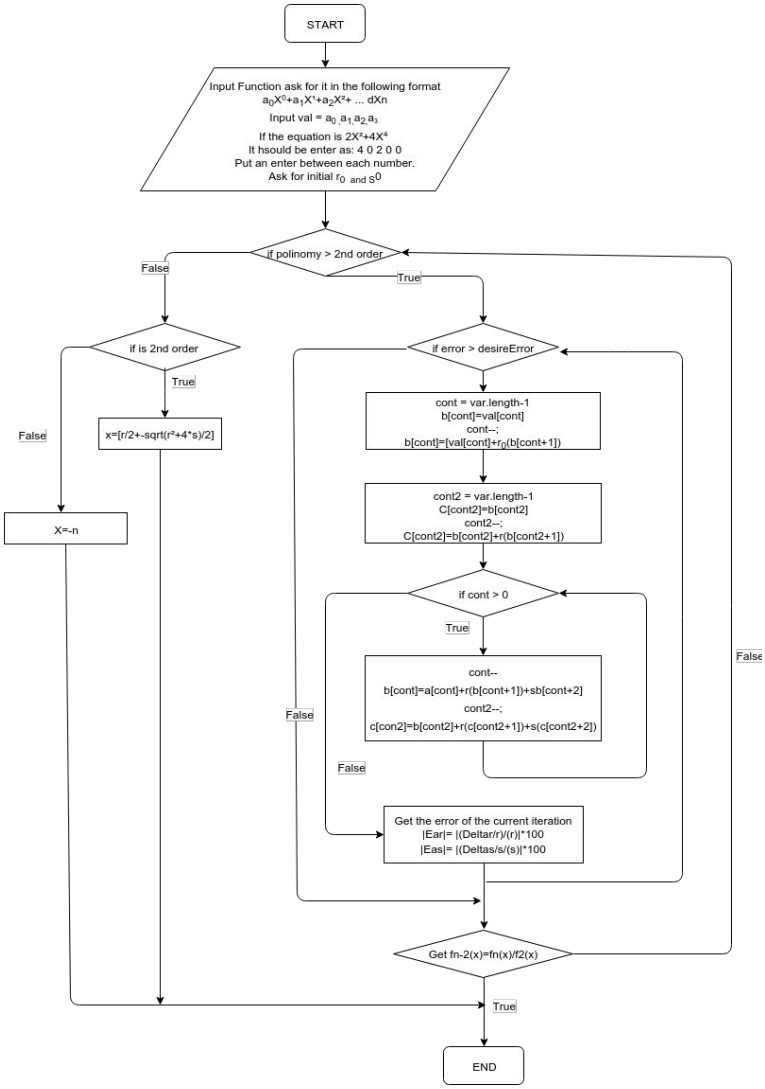
$X_i =$ *current estimate of root*

$X_{i+1} =$ *new calculated estimate of root*



Explanation Bairstow's:

On Bairstow method first it will be asked for the polinomy, to be inserted in the following format:
4 0 2 0 0 where each of the values will be as the following polinomy $4X^4 + 0X^3 + 2X^2 + 0X^1 + 0X^0$ then the polinomy will be clasified, if it's greater than second order, it will enter on the loop to calculate the values of the r,s and will get the values of the b and a to form the next reduced polinomy. In that loop will be ask if the counter is greater than 0 the code will continue calculating the values of the polinomy, until it reach the zero in which it will have found the end of the inserted polinomy. After that it will check the error and if the error it's smaller than the expected error, then it will finish that iteration and it will asked if the function fn-2(x)=fn(x)/f2n(x), if its true the solution, will be display, if not the program should go to the start of the loop and ask once again if the polinomy if greater than second order, and do the process all over again.
If the polinomy is of 2nd order or smaller it will go as false on the first if, and it will enter on a second if that will ask if it's from second or first order.
If it's of first order x = -n if it's of second order it will be $X = r/2 +- sqrt(r^2+4*S)$.



Example Outputs

Bisection:

Inputs:
(function): y=x^4-x-10
(xl value): 1.5
(xu value): 2
(max error value in %): 1

iteration	xl	xu	xm	sign	error
1	1.5	2	1.75	+	---
2	1.75	2	1.875	-	6.66667%
3	1.75	1.875	1.8125	+	3.44837%
4	1.8125	1.875	1.84375	+	1.6949153%
5	1.84375	1.875	1.859375	-	0.8403361%

Secant:

Input

(Function): y=1*x^3-0.165*x^2+0.0003993

(x0 value:) 0.02

(x1 value:) 0.05

(max error value in %): 1

Iteration	xi-1	xi	f(xi-1)	f(xi)	xi+1	error
1	0.02	0.05	0.0003413	0.0001118	0.0644144	---
2	0.05	0.0644144	0.0001118	-0.0000198	0.0624144	3.5247136%
3	0.0646144	0.0624144	-0.0000198	-0.0000003	0.0623774	0.0594676%

Newthon-R:

(function): y=1*x^3-0.165*x^2+0.0003993
(x0 value:) 0.02
(max error value in %) : 1

!iteration xi f(xi) derivate f(xi) error !

Iteration	xi	f(xi)	derivative f(xi)	error
1	0.02	0.0003413	-0.0054	---
2	0.0832037	-0.0001670	-0.0066887	75.962609%
3	0.0582414	0.0000372	-0.0090435	42.859999%
4	0.0623514	0.0000002	-0.0089129	6.5915692%
5	0.0623776	1.518D-11	-0.0089117	0.0420318%

Bairstow:

input: [1,1,3,4,6]
(r0 value): -2.1
(s0 value): -1.9
(max error value in %): 1

	Initial r	Initial s			
Iteration 1	-2.1	-1.9			
	A	B	C		
--1--	1	1	1		
--2--	1	-1.1	-3.2		
--3--	3	3.41	8.23		
--4--	4	-1.071	-12.274		
--5--	6	1.7701	11.9085		
delta r	delta s	r	s	error r	error s
0.1106972	-0.0499882	-1.9893028	-1.9499882	-5.5646218%	2.5635126%

	Initial r	Initial s			
Iteration 2	-1.9893028	-1.9499882			
	A	B	C		
--1--	1	1	1		
--2--	1	-0.9893028	-2.9786056		
--3--	3	3.0180347	6.9933951		
--4--	4	-0.0746561	-8.1783909		
--5--	6	0.2633816	2.8956398		
delta r	delta s	r	s	error r	error s
-0.0106899	-0.0501628	-1.9999928	-2.000151	0.5344994%	2.50795%

	Initial r	Initial s			
Iteration 3	-1.9999928	-2.000151			
	A	B	C		
--1--	1	1	1		
--2--	1	-0.9999928	-2.9999855		
--3--	3	2.9998273	6.9996257		
--4--	4	0.0005035	-7.9982733		
--5--	6	-0.0011147	1.9950659		
delta r	delta s	r	s	error r	error s
-0.0000072	0.0001510	-2	-2	0.0003613%	-0.0075494%

Root values

- 1. + i

- 1. - i

Feedback Homework:

The project became a challenge in different points of view, we don't thought that will take us that long and that we will challenge us in that way. The comprehension of the methods became really clear because in order to create the functions, we should understand really good how does the methods work, and how we will make the algorithm ,which was not that easy specially with the Bairstow's method. As a project we think that was interesting and complex, in some way hard and challenging, but we got enough time to make the project as it should be, so in conclusion we thought that was a really great project and show us with the research how we can use it on a real life problem.

Self-Evaluation

Juan Carlos Aguilera Pérez: We had different roles on the work and all of the team member do their work, in my opinion my work was done as it should be done, and I were attending the group work and what we were missing of the project. I think that my grade should be 100 because we all made our work

Marcopolo Gil Melchor: I really spent too much time coding the algorithms. We decided to code the Bairstow method, so I coded that one too. In my opinion I deserve a 100.

Jesús Amauri Guillén García:
I did the research part of the work, so i expend alot of time trying to fin example where people use the methods seen in class instead of using some computer base program to solve their work, and i also was in charge o making the plots in the work and some coding investigation os what kind of function could simplify the programs we were making. out team gather several times to finish this project and work really well togather.

Peer-Evaluation of the team members

	Juan Carlos	Marcopolo	Jesús Amauri
Juan Carlos evaluation for	X	100	100
Explanation	X	He work really hard with the codes and organizing all of them. He was in charge of the final result of the program	He help with the plotting examples and the plot cases, also he set a start point with examples of the base program, which later were modified.
Marcopolo evaluation for	100	X	100
Explanation	He worked with the flow charts, and the whole organization, the document was his responsibility.	X	He worked with the research and plotting the function.
Jesús Amauri evaluation for	100	100	X
Explanation	he did all the work we gave to him, and did it on time	he had expend a lot of time finding a way to do the bairstow program.	X

References

Gurley. (2010). Curve Fitting Techniques. En Computer Methods(89-112). -: -.

Physics. (14 de Febrero 2011). Physics: Finding Roots. 12 de Septiembre 2015, de - Sitio web:http://academic.reed.edu/physics/courses/P200.L.S11/Physics200Lab/files/Bisection.pdf