Table of Contents

[INTRODUCTION 1](#_Toc444004334)

[Question 1 2](#_Toc444004335)

[Question 2: 3](#_Toc444004336)

[Proving the Reduced Cubic Equation will have at least one Real Root 3](#_Toc444004337)

[Reducing the Equation to the Canonical Form 4](#_Toc444004338)

[Special Case when α = 0 5](#_Toc444004339)

[Test Cases 6](#_Toc444004340)

[Case I 6](#_Toc444004341)

[Case II 6](#_Toc444004342)

[Case III 7](#_Toc444004343)

[Command Line used to Compile Testing Program 7](#_Toc444004344)

[Question 3: 8](#_Toc444004345)

[Part i. 8](#_Toc444004346)

[Part ii. 8](#_Toc444004347)

[Part iii. 9](#_Toc444004348)

[Part iv. 10](#_Toc444004349)

[Part v. 11](#_Toc444004350)

[Question 4 14](#_Toc444004351)

[Derivation to Reduced Cubic Form 14](#_Toc444004352)

[Neon 14](#_Toc444004353)

[Triple Point 15](#_Toc444004354)

[Question 5 17](#_Toc444004355)

[Mastery Section 19](#_Toc444004356)

[Conclusion 19](#_Toc444004357)

[Appendix 20](#_Toc444004358)

[Tables 20](#_Toc444004359)

[Representative C code and files 23](#_Toc444004360)

All work done in this project is my own unless stated otherwise

INTRODUCTION

This project deals with creating a numerical cubic equation solver in the programming language C and later applying it to a modified version of the ideal gas equation. Part 1 - 4 involve cubic equations that have real coefficients.

In Question 1 we go back to Project 1a and split our program into respective files such that it is easier to access/modify the modules for use in multiple programs.

In Question 2 we implement a very basic cubic equation solver by reducing the equations to canonical form and then applying the Newton-Raphson method on it to find one root to the equation. We then use our quad\_roots function developed in Project 1a to find the other two roots.

In Question 3 we recognize and deal with many specific cases presented and compare the outputs prior to and after optimization.

In Question 4 we look at solving the Van der Waals Equation of State for gases, specifically Neon, and also find the triple point for Neon.

In Question 5 we add simple code to print out the time and date and also find out how to print out the Euro symbol.

The mastery section allows for cubic equations with complex coefficients and produces a plot on the complex plane for the results found for a given specific set case.

# Question 1

The program from project 1a is split into three files

So now there are 3 files involved:

1. prog\_1.c - contains the main function
2. lin\_root.c - contains the linear root finding function
3. quad\_roots.c - contains the quadratic root finding function plus other small functions used in its implementation.

which I compile using the command: “gcc –o prog\_1 prog\_1.c order\_nums.c lin\_root.c quad\_roots.c”

Find the appropriate files in the appendix.

# Question 2:

In this part of the project we create a cubic equation solver. Rather than dealing with an entirely generic cubic equation we deal with a reduced cubic equation which has 1 as the coefficient for the term. That is

In order to solve this cubic equation numerically we reduce the equation further to its canonical form of

where and implement the Newton-Raphson method to find the first root iteratively using the formula

where

which relies on a good initial estimate of for convergence to occur. Once one root, lets call it , has been found we can then return to our initial reduced cubic equation and factor out the root to form a quadratic equation of the form

We can easily solve this using the quad\_roots function developed in Project 1a.

In order to avoid overflow issues we make sure to deal with trivial cases where there is no need to call the Newton-Raphson method and cases where we cannot reduce our equation to the canonical form. Further optimizations are done in Question 3.

## Proving the Reduced Cubic Equation will have at least one Real Root

We can easily show that the reduced cubic equation will have at least one real root using the Intermediate Value Theorem (IVT). The theorem states that for any real continuous function in the domain if and then must also take on all values between and for.

To implement this, we first take the limits to positive and negative infinity on our reduced cubic equation

That is as we note that and as we note that . We also note that is a continuous function – it is a polynomial of degree 3 - and so we can use the IVT to say takes on all values between in the domain of . Thus its range is . We conclude that for some value of , . That is has at least one real root.

## Reducing the Equation to the Canonical Form

We use the substitution

and are trying to achieve the form

substituting into we get

we multiple both sides by . Note is necessary for this to work.

we now compare this with our initial equation that is

so we now know that

this implies

and using the other two equations we get that

diving by we can get rid of p to get an equation for .

substituting this back into the value of p we get that

obviously this method holds the restriction that . So we must make deal with the special case when as we cannot then reduce our cubic equation to the form

### Special Case when α = 0

This occurs when .

Substituting we get that and so when the reduction to canonical form is not possible.

To solve this first we consider expanding with

then substituting we get

we know that as that is the specific case we are dealing with so we can note that the order zero term vanishes and we’re left with:

using our earlier definition of – note this definition involves an term but we can arbitrarily now choose rather than the definition of from before – this then reduces to

So now we know is a solution to this equation and the rest can be solved by setting the quadratic in the brackets to zero. So is a solution for our reduced cubic equation. Setting an arbitrary value for , lets choose for simplicity, we can now solve for the other roots using our quad\_roots function by rewriting in terms of and .

Once we have our all our roots for the equations we can compare them to see if there are any repeated roots and order them by size and return the appropriate integer.

## Test Cases

Here I test my program on the given test cases

### Case I

The reduced cubic equation is

My input and output looks like:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 11:43:58

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 2 -5 -6

There are three distinct real roots.

r1 = 2, r2 = -1, r3 = -3

f(r1) = 5.329070518e-15

f(r2) = -8.881784197e-16

f(r3) = 0

### Case II

The reduced cubic equation is

My input and output looks like:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 11:43:58

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 5 3 -9

There are three distinct real roots.

r1 = 1, r2 = -3, r3 = -3

f(r1) = 3.552713679e-15

f(r2) = -1.065814104e-14

f(r3) = -1.065814104e-14

It is worth noting that my cubic solver thinks that there are 3 distinct roots here but in reality 2 of the roots are repeated. This case is dealt with later in Question 3.

### Case III

The reduced cubic equation is

My input and output looks like:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 11:48:18

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 1 -1 -15

There is one real root (r1) and two complex roots (r2,r3).

r1 = 2.291909782, r2 = -1.645954891 + 1.958466933i, r3 = -1.645954891 - 1.958466933i

f(r1) = 3.552713679e-15

## Command Line used to Compile Testing Program

I use the following command line in my terminal to compile my testing program:

gcc -o prog\_2 prog\_2.c lin\_root.c quad\_roots.c rcubic\_roots.c

The quad\_roots.c and lin\_root.c files are the same throughout the project. Find the rcubic\_roots.c and prog\_2.c in the appendix.

# Question 3:

In this question we optimize our rcubic\_roots function from Question 2 by dealing with some special cases.

## Part i.

If then we are dealing with a simple case where which implies that the roots are

similarly, if then we get that

which we can then substitute back for getting the roots of .

An example case of input and output that utilizes this case would be:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:04:59

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 0 0 2

There is one real root (r1) and two complex roots (r2,r3).

r1 = -1.25992105, r2 = -0.6299605249 + 1.091123636i, r3 = -0.6299605249 - 1.091123636i

f(r1) = 0

which is actually the same as the output for prog\_2.c

## Part ii.

If then we get that

which has one root at and the other roots we can simply find by calling our quadratic solver. I have also dealt with this case in Part 2 of the project to avoid getting which would break the reduced quadratic we form later.

An example case of input and output that utilizes this case would be:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 14:00:44

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 2 2 0

There is one real root (r1) and two complex roots (r2,r3).

r1 = 0, r2 = -1 + 1i, r3 = -1 - 1i

f(r1) = 0

whereas using the non-optimized code from prog\_2 outputs:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:11:00

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 2 2 0

There are three distinct real roots.

r1 = 1.11022e-16, r2 = 0, r3 = -2

f(r1) = 2.220446049e-16

f(r2) = 0

f(r3) = -4

We can easily see that the newly optimized solution is gives the correct answer whereas the non-optimized code does not give the correct answer.

## Part iii.

If then we can write

which can be factorized into

giving us the roots

An example case of input and output that utilizes this case would be:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:17:54

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: -3 -16 18

There are three distinct real roots.

r1 = 5.3589, r2 = 1, r3 = -3.3589

f(r1) = 0

f(r2) = 0

f(r3) = -1.421085472e-14

whereas using the non-optimized code from prog\_2 outputs:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:11:00

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: -3 -6 18

There are three distinct real roots.

r1 = 3, r2 = 2.44949, r3 = -2.44949

f(r1) = 2.131628207e-14

f(r2) = 2.131628207e-14

f(r3) = -2.131628207e-14

and again we can clearly see how the optimized code is more accurate.

## Part iv.

If and then we can write out cubic equation as

which can be factorized as

this means that there are three identical roots at

An example case of input and output that utilizes this case would be:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:21:56

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 6 12 8

There are triple repeated real roots.

r1 = r2 = r3 = -2

f(r1) = 0

whereas using the non-optimized code from prog\_2 outputs:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:11:00

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 6 12 8

There is a pair of repeated real roots.

r1 = -2, r2 = -2, r3 = -2

f(r1) = 0

f(r2) = 0

f(r3) = 0

We can see that although the non-optimized program calculates the correct roots to the same accuracy is fails to deal with the fact that the 3 values are the same and hence are a triple repeated root whereas the new optimized code does deal with this properly.

## Part v.

If there is a repeated root,, then we know that the derivative at the point of the repeated root will be zero.

We already know that our canonical cubic, is of the form

so

and then

Solving these two equations we get that

substituting that into we get that

and so we get that

For close to we get that

since there is a repeated root we can try to factorize out so that we are left with where is the other root of the equation. So we can write

comparing the final RHS with the LHS gives us that

We now see if the solution works by substituting the values into (\*) and (\*\*).

(\*) is now

we know so so (\*) is satisfied.

Now we check (\*\*) which is

which is the same as above and so is also true. Thus is the other root of the canonical cubic equation. We get our desired roots by substituting back for .

The definition of very close to must be considered. As we do some algebra to compute , and , we might loose accuracy in their computed values and so I say that rather than simply where is the double floating point accuracy stored in the <float.h> file.

An example case that utilizes this optimization would be the Test Case II from Question 2. My input and output now looks like:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:21:56

Date: Feb 23 2016

Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0

in the order a2,a1,a0, separated by spaces: 5 3 -9

There is a pair of repeated real roots.

r1 = 1, r2 = -3, r3 = -3

f(r1) = -1.776356839e-14

f(r2) = 0

f(r3) = 0

as we can see now the program realizes that two of roots are repeated. Furthermore, the accuracy of our solutions has increased.

# Question 4

In this part of the project we solve the ideal gas equation and the improved Van der Waals equation of state.

## Derivation to Reduced Cubic Form

The ideal gas equation is

The Van der Waals improved equation is

we multiply by to get

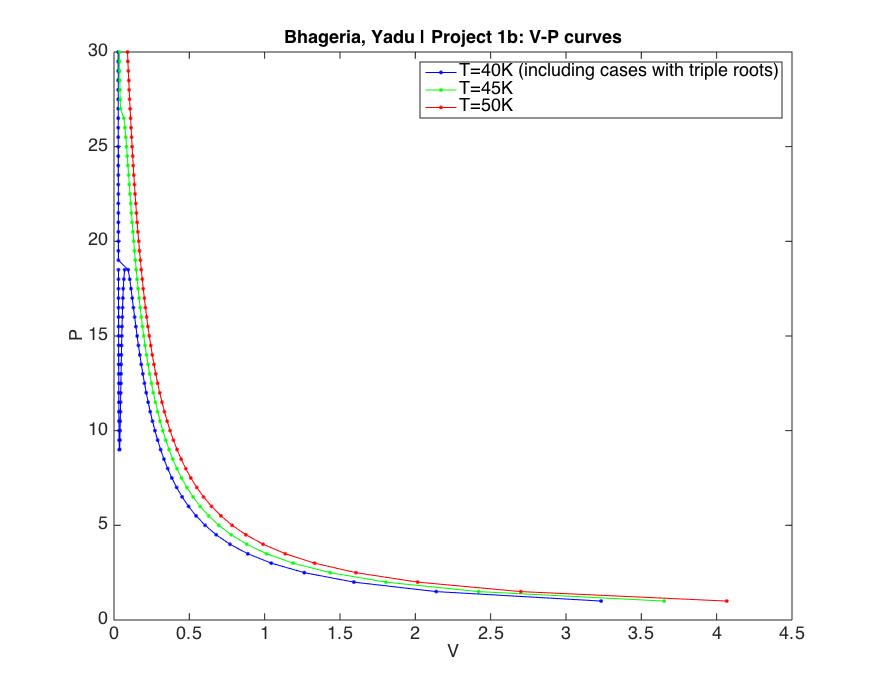
divide by to get a reduced cubic equation

For given , , P and T we can now solve this equation in our cubic solver created in the previous part.

## Neon

In the case of Neon, we know and (given and https://en.wikipedia.org/wiki/Van\_der\_Waals\_constants\_(data\_page)) so we iterate over the asked values of T and P to get print out the solutions for the ideal gas equation and the Van Der Waals equation.

The outputted **table** is quite large and hence **has been placed in the appendix**.

The 3 V-P curves for T = 40, 45 & 50K are in the figure below:

## Triple Point

To find the triple point of Neon we can consider what conditions need to be fulfilled for a triple root. From Part 3 (iv) we know that when

and

we have a triple real root. This means that

and

which implies that

so

and then substituting back we find that

Using these formulae for and we get that takes three identical values for

and

Now using these values to solve the Van der Waals cubic equation I get the output:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 13:59:44

Date: Feb 23 2016

T: 45.108893

P: 27.073848

There are triple repeated real roots.

r1 = r2 = r3 = 0.05127

f(r1) = -1.355252716e-19

It is worth noting that this value of T and P are close to one of our inputted values of T=45 and P = 27 but not close enough and thus our code still gives three distinct roots.

# Question 5

Running the program on various compilers I got different results. I have included two extra lines of Code for the Euro symbol so my program now looks like:

#include <stdio.h>

int main(void) {

/\* Bhageria, Yadu, M3SC \*/

printf("\n What is this? : \234");

printf("\n Time : %s ",\_\_TIME\_\_);

printf("\n Date : %s ",\_\_DATE\_\_);

printf("\n Euro : \xE2\x82\xAC");

printf("\n Euro : \u20AC");

printf("\n \n");

return(0);

}

On ICL I got:

? : £

Time : 11:32:07

Date : Feb 16 2016

Euro : Ôé¼

Euro : Ç

By piping the data from ICL to a text file with “prog\_5 > data\_out.txt" I got:

?    : œ

Time : 11:32:07

Date : Feb 16 2016

Euro : â‚¬

Euro : €

On Cygwin I got:

?    : ▒

Time : 11:11:15

Date : Feb 16 2016

Euro : €

Euro : €

Using GCC on my Mac I got:

What is this? : ?

Time : 15:00:36

Date : Feb 11 2016

Euro : €

Euro : €

The second and third lines of output correspond to the Time and Date respectively (as is shown by my changing the “What is this?”). This is useful to have in a program as it means that one can know when the program was run and keep track of it.

The first line gives different output on different compilers. GCC on my Mac and Cygwin did not recognize the character. ICL printed “£” but when I copied the character to a text file it converted to “œ”.

For the euro symbol, “\u20AC” - Unicode Character - gives an output of the Euro sign in GCC and Cygwin but not in ICL although the text file does recognize it. “\xE2\x82\xAC” - hexadecimal character - does not get converted even in the text file.

Both of these can be explained by the fact that ICL uses MS-DOS formatting but the text file that I created does not. MS-DOS formatting does not support the euro symbol as it was discontinued in 2000 before the euro symbol’s release.

# Mastery Section

In this part of the project we deal with reduced cubic equation as well but with complex coefficients.

We again use the Newton-Raphson method but instead on the explicit equation. That is we choose a starting value for

and then iterate using the formula

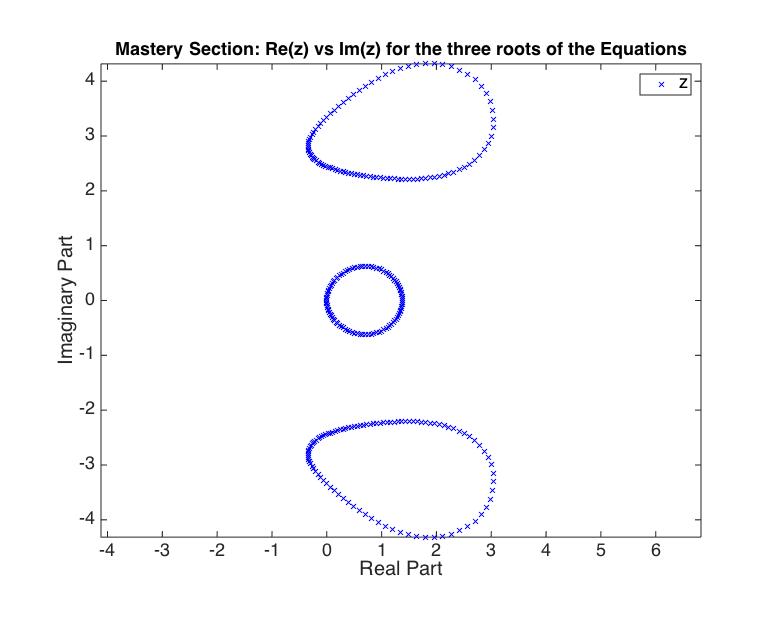
I again implement the basic optimizations taken care of as in Question 3.

For the given values of

where

for

The table of outputted roots has been placed in the appendix as it is quite large. The Plot produced of the roots on the complex plane is as follows



Appendix

# Tables

## Table from Question 4

Below is the table that I output using my main function from prog\_4.c to get the roots to the Van der Waals Equation.

I output my values in csv format for easy formatting to an excel spread sheet – I recommend using the command “prog\_4.c > data\_out.csv” or copying the data to Excel and splitting the data using the comma values.

Table 1: Output for Part 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| T | P | V\_ideal | V\_vdw1 | V\_vdw2 | V\_vdw3 |
| 40 | 1.0 | 3.28231 | 3.23373 |  |  |
| 40 | 1.5 | 2.18821 | 2.13930 |  |  |
| 40 | 2.0 | 1.64116 | 1.59191 |  |  |
| 40 | 2.5 | 1.31292 | 1.26333 |  |  |
| 40 | 3.0 | 1.09410 | 1.04415 |  |  |
| 40 | 3.5 | 0.93780 | 0.88748 |  |  |
| 40 | 4.0 | 0.82058 | 0.76988 |  |  |
| 40 | 4.5 | 0.72940 | 0.67831 |  |  |
| 40 | 5.0 | 0.65646 | 0.60496 |  |  |
| 40 | 5.5 | 0.59678 | 0.54486 |  |  |
| 40 | 6.0 | 0.54705 | 0.49470 |  |  |
| 40 | 6.5 | 0.50497 | 0.45216 |  |  |
| 40 | 7.0 | 0.46890 | 0.41563 |  |  |
| 40 | 7.5 | 0.43764 | 0.38388 |  |  |
| 40 | 8.0 | 0.41029 | 0.35602 |  |  |
| 40 | 8.5 | 0.38615 | 0.33135 |  |  |
| 40 | 9.0 | 0.36470 | 0.30934 | 0.03752 | 0.03493 |
| 40 | 9.5 | 0.34551 | 0.28957 | 0.03916 | 0.03387 |
| 40 | 10.0 | 0.32823 | 0.27168 | 0.04039 | 0.03325 |
| 40 | 10.5 | 0.31260 | 0.25541 | 0.04150 | 0.03278 |
| 40 | 11.0 | 0.29839 | 0.24052 | 0.04256 | 0.03241 |
| 40 | 11.5 | 0.28542 | 0.22683 | 0.04359 | 0.03209 |
| 40 | 12.0 | 0.27353 | 0.21417 | 0.04464 | 0.03180 |
| 40 | 12.5 | 0.26258 | 0.20242 | 0.04570 | 0.03155 |
| 40 | 13.0 | 0.25249 | 0.19145 | 0.04680 | 0.03133 |
| 40 | 13.5 | 0.24313 | 0.18116 | 0.04794 | 0.03112 |
| 40 | 14.0 | 0.23445 | 0.17147 | 0.04915 | 0.03093 |
| 40 | 14.5 | 0.22637 | 0.16228 | 0.05043 | 0.03075 |
| 40 | 15.0 | 0.21882 | 0.15352 | 0.05181 | 0.03058 |
| 40 | 15.5 | 0.21176 | 0.14511 | 0.05332 | 0.03043 |
| 40 | 16.0 | 0.20514 | 0.13697 | 0.05499 | 0.03028 |
| 40 | 16.5 | 0.19893 | 0.12900 | 0.05688 | 0.03014 |
| 40 | 17.0 | 0.19308 | 0.12109 | 0.05907 | 0.03001 |
| 40 | 17.5 | 0.18756 | 0.11304 | 0.06173 | 0.02988 |
| 40 | 18.0 | 0.18235 | 0.10450 | 0.06518 | 0.02976 |
| 40 | 18.5 | 0.17742 | 0.09436 | 0.07051 | 0.02964 |
| 40 | 19.0 | 0.17275 | 0.02953 |  |  |
| 40 | 19.5 | 0.16832 | 0.02943 |  |  |
| 40 | 20.0 | 0.16412 | 0.02933 |  |  |
| 40 | 20.5 | 0.16011 | 0.02923 |  |  |
| 40 | 21.0 | 0.15630 | 0.02913 |  |  |
| 40 | 21.5 | 0.15267 | 0.02904 |  |  |
| 40 | 22.0 | 0.14920 | 0.02895 |  |  |
| 40 | 22.5 | 0.14588 | 0.02887 |  |  |
| 40 | 23.0 | 0.14271 | 0.02878 |  |  |
| 40 | 23.5 | 0.13967 | 0.02870 |  |  |
| 40 | 24.0 | 0.13676 | 0.02862 |  |  |
| 40 | 24.5 | 0.13397 | 0.02855 |  |  |
| 40 | 25.0 | 0.13129 | 0.02847 |  |  |
| 40 | 25.5 | 0.12872 | 0.02840 |  |  |
| 40 | 26.0 | 0.12624 | 0.02833 |  |  |
| 40 | 26.5 | 0.12386 | 0.02826 |  |  |
| 40 | 27.0 | 0.12157 | 0.02820 |  |  |
| 40 | 27.5 | 0.11936 | 0.02813 |  |  |
| 40 | 28.0 | 0.11723 | 0.02807 |  |  |
| 40 | 28.5 | 0.11517 | 0.02800 |  |  |
| 40 | 29.0 | 0.11318 | 0.02794 |  |  |
| 40 | 29.5 | 0.11126 | 0.02788 |  |  |
| 40 | 30.0 | 0.10941 | 0.02783 |  |  |
| 45 | 1.0 | 3.69260 | 3.65150 |  |  |
| 45 | 1.5 | 2.46173 | 2.42043 |  |  |
| 45 | 2.0 | 1.84630 | 1.80480 |  |  |
| 45 | 2.5 | 1.47704 | 1.43534 |  |  |
| 45 | 3.0 | 1.23087 | 1.18896 |  |  |
| 45 | 3.5 | 1.05503 | 1.01291 |  |  |
| 45 | 4.0 | 0.92315 | 0.88082 |  |  |
| 45 | 4.5 | 0.82058 | 0.77803 |  |  |
| 45 | 5.0 | 0.73852 | 0.69574 |  |  |
| 45 | 5.5 | 0.67138 | 0.62838 |  |  |
| 45 | 6.0 | 0.61543 | 0.57219 |  |  |
| 45 | 6.5 | 0.56809 | 0.52461 |  |  |
| 45 | 7.0 | 0.52751 | 0.48379 |  |  |
| 45 | 7.5 | 0.49235 | 0.44837 |  |  |
| 45 | 8.0 | 0.46158 | 0.41734 |  |  |
| 45 | 8.5 | 0.43442 | 0.38992 |  |  |
| 45 | 9.0 | 0.41029 | 0.36551 |  |  |
| 45 | 9.5 | 0.38869 | 0.34364 |  |  |
| 45 | 10.0 | 0.36926 | 0.32392 |  |  |
| 45 | 10.5 | 0.35168 | 0.30604 |  |  |
| 45 | 11.0 | 0.33569 | 0.28975 |  |  |
| 45 | 11.5 | 0.32110 | 0.27484 |  |  |
| 45 | 12.0 | 0.30772 | 0.26113 |  |  |
| 45 | 12.5 | 0.29541 | 0.24849 |  |  |
| 45 | 13.0 | 0.28405 | 0.23678 |  |  |
| 45 | 13.5 | 0.27353 | 0.22591 |  |  |
| 45 | 14.0 | 0.26376 | 0.21577 |  |  |
| 45 | 14.5 | 0.25466 | 0.20629 |  |  |
| 45 | 15.0 | 0.24617 | 0.19740 |  |  |
| 45 | 15.5 | 0.23823 | 0.18905 |  |  |
| 45 | 16.0 | 0.23079 | 0.18117 |  |  |
| 45 | 16.5 | 0.22379 | 0.17373 |  |  |
| 45 | 17.0 | 0.21721 | 0.16668 |  |  |
| 45 | 17.5 | 0.21101 | 0.15998 |  |  |
| 45 | 18.0 | 0.20514 | 0.15361 |  |  |
| 45 | 18.5 | 0.19960 | 0.14752 |  |  |
| 45 | 19.0 | 0.19435 | 0.14170 |  |  |
| 45 | 19.5 | 0.18936 | 0.13612 |  |  |
| 45 | 20.0 | 0.18463 | 0.13074 |  |  |
| 45 | 20.5 | 0.18013 | 0.12556 |  |  |
| 45 | 21.0 | 0.17584 | 0.12055 |  |  |
| 45 | 21.5 | 0.17175 | 0.11568 |  |  |
| 45 | 22.0 | 0.16785 | 0.11093 |  |  |
| 45 | 22.5 | 0.16412 | 0.10628 |  |  |
| 45 | 23.0 | 0.16055 | 0.10170 |  |  |
| 45 | 23.5 | 0.15713 | 0.09717 |  |  |
| 45 | 24.0 | 0.15386 | 0.09263 |  |  |
| 45 | 24.5 | 0.15072 | 0.08804 |  |  |
| 45 | 25.0 | 0.14770 | 0.08332 |  |  |
| 45 | 25.5 | 0.14481 | 0.07833 |  |  |
| 45 | 26.0 | 0.14202 | 0.07278 |  |  |
| 45 | 26.5 | 0.13934 | 0.06574 |  |  |
| 45 | 27.0 | 0.13676 | 0.04340 |  |  |
| 45 | 27.5 | 0.13428 | 0.04074 |  |  |
| 45 | 28.0 | 0.13188 | 0.03936 |  |  |
| 45 | 28.5 | 0.12956 | 0.03839 |  |  |
| 45 | 29.0 | 0.12733 | 0.03763 |  |  |
| 45 | 29.5 | 0.12517 | 0.03701 |  |  |
| 45 | 30.0 | 0.12309 | 0.03649 |  |  |
| 50 | 1.0 | 4.10289 | 4.06771 |  |  |
| 50 | 1.5 | 2.73526 | 2.69997 |  |  |
| 50 | 2.0 | 2.05145 | 2.01603 |  |  |
| 50 | 2.5 | 1.64116 | 1.60562 |  |  |
| 50 | 3.0 | 1.36763 | 1.33198 |  |  |
| 50 | 3.5 | 1.17225 | 1.13648 |  |  |
| 50 | 4.0 | 1.02572 | 0.98982 |  |  |
| 50 | 4.5 | 0.91175 | 0.87572 |  |  |
| 50 | 5.0 | 0.82058 | 0.78442 |  |  |
| 50 | 5.5 | 0.74598 | 0.70969 |  |  |
| 50 | 6.0 | 0.68382 | 0.64739 |  |  |
| 50 | 6.5 | 0.63121 | 0.59466 |  |  |
| 50 | 7.0 | 0.58613 | 0.54943 |  |  |
| 50 | 7.5 | 0.54705 | 0.51022 |  |  |
| 50 | 8.0 | 0.51286 | 0.47589 |  |  |
| 50 | 8.5 | 0.48269 | 0.44557 |  |  |
| 50 | 9.0 | 0.45588 | 0.41861 |  |  |
| 50 | 9.5 | 0.43188 | 0.39447 |  |  |
| 50 | 10.0 | 0.41029 | 0.37272 |  |  |
| 50 | 10.5 | 0.39075 | 0.35303 |  |  |
| 50 | 11.0 | 0.37299 | 0.33512 |  |  |
| 50 | 11.5 | 0.35677 | 0.31874 |  |  |
| 50 | 12.0 | 0.34191 | 0.30371 |  |  |
| 50 | 12.5 | 0.32823 | 0.28987 |  |  |
| 50 | 13.0 | 0.31561 | 0.27708 |  |  |
| 50 | 13.5 | 0.30392 | 0.26522 |  |  |
| 50 | 14.0 | 0.29306 | 0.25419 |  |  |
| 50 | 14.5 | 0.28296 | 0.24391 |  |  |
| 50 | 15.0 | 0.27353 | 0.23430 |  |  |
| 50 | 15.5 | 0.26470 | 0.22529 |  |  |
| 50 | 16.0 | 0.25643 | 0.21683 |  |  |
| 50 | 16.5 | 0.24866 | 0.20887 |  |  |
| 50 | 17.0 | 0.24135 | 0.20136 |  |  |
| 50 | 17.5 | 0.23445 | 0.19426 |  |  |
| 50 | 18.0 | 0.22794 | 0.18755 |  |  |
| 50 | 18.5 | 0.22178 | 0.18118 |  |  |
| 50 | 19.0 | 0.21594 | 0.17513 |  |  |
| 50 | 19.5 | 0.21040 | 0.16938 |  |  |
| 50 | 20.0 | 0.20514 | 0.16389 |  |  |
| 50 | 20.5 | 0.20014 | 0.15866 |  |  |
| 50 | 21.0 | 0.19538 | 0.15366 |  |  |
| 50 | 21.5 | 0.19083 | 0.14888 |  |  |
| 50 | 22.0 | 0.18650 | 0.14430 |  |  |
| 50 | 22.5 | 0.18235 | 0.13990 |  |  |
| 50 | 23.0 | 0.17839 | 0.13568 |  |  |
| 50 | 23.5 | 0.17459 | 0.13162 |  |  |
| 50 | 24.0 | 0.17095 | 0.12771 |  |  |
| 50 | 24.5 | 0.16746 | 0.12394 |  |  |
| 50 | 25.0 | 0.16412 | 0.12030 |  |  |
| 50 | 25.5 | 0.16090 | 0.11679 |  |  |
| 50 | 26.0 | 0.15780 | 0.11339 |  |  |
| 50 | 26.5 | 0.15483 | 0.11010 |  |  |
| 50 | 27.0 | 0.15196 | 0.10691 |  |  |
| 50 | 27.5 | 0.14920 | 0.10381 |  |  |
| 50 | 28.0 | 0.14653 | 0.10080 |  |  |
| 50 | 28.5 | 0.14396 | 0.09788 |  |  |
| 50 | 29.0 | 0.14148 | 0.09503 |  |  |
| 50 | 29.5 | 0.13908 | 0.09225 |  |  |
| 50 | 30.0 | 0.13676 | 0.08955 |  |  |

## Table from Mastery Section

This is the table of the 303 roots outputted by my complex cubic roots function for the given cases.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| j | Re(z1) | Im(z1) | Re(z2) | Im(z2) | Re(z1) | Im(z2) |
| 0 | 3 | 3 | 3 | -3 | 0 | 0 |
| 1 | 3.036029 | -3.147397 | 2.94331 | 2.865782 | 0.000987 | -0.031411 |
| 2 | 3.047349 | -3.303789 | 2.870769 | 2.747239 | 0.00395 | -0.06279 |
| 3 | 3.031375 | -3.463726 | 2.787186 | 2.645175 | 0.008896 | -0.094107 |
| 4 | 2.987364 | -3.621202 | 2.696774 | 2.559041 | 0.015833 | -0.125327 |
| 5 | 2.916411 | -3.77042 | 2.602853 | 2.487427 | 0.024776 | -0.156417 |
| 6 | 2.821124 | -3.9064 | 2.50782 | 2.42852 | 0.035741 | -0.187337 |
| 7 | 2.705139 | -4.02534 | 2.413275 | 2.380442 | 0.048751 | -0.218046 |
| 8 | 2.572612 | -4.12473 | 2.320207 | 2.341437 | 0.063831 | -0.248497 |
| 9 | 2.427794 | -4.203292 | 2.22917 | 2.309967 | 0.081009 | -0.278638 |
| 10 | 2.274707 | -4.260799 | 2.140429 | 2.284726 | 0.100318 | -0.308408 |
| 11 | 2.116944 | -4.29786 | 2.054064 | 2.264634 | 0.121793 | -0.337737 |
| 12 | 1.970042 | 2.248805 | 1.957559 | -4.315693 | 0.145472 | -0.366543 |
| 13 | 1.888267 | 2.236519 | 1.799035 | -4.315921 | 0.171392 | -0.394731 |
| 14 | 1.808611 | 2.227192 | 1.643301 | -4.300408 | 0.199591 | -0.422189 |
| 15 | 1.730933 | 2.22035 | 1.491788 | -4.271121 | 0.230105 | -0.448784 |
| 16 | 1.655093 | 2.21561 | 1.345494 | -4.230043 | 0.262962 | -0.474361 |
| 17 | 1.580956 | 2.212659 | 1.20506 | -4.179101 | 0.298179 | -0.498737 |
| 18 | 1.508399 | 2.211243 | 1.070846 | -4.120133 | 0.335754 | -0.521701 |
| 19 | 1.437312 | 2.211153 | 0.943002 | -4.05486 | 0.375662 | -0.543013 |
| 20 | 1.367595 | 2.212222 | 0.821532 | -3.984878 | 0.417838 | -0.5624 |
| 21 | 1.299164 | 2.214309 | 0.706354 | -3.911657 | 0.462171 | -0.579565 |
| 22 | 1.231948 | 2.217301 | 0.59735 | -3.836532 | 0.508489 | -0.594196 |
| 23 | 1.165886 | 2.221106 | 0.556547 | -0.605978 | 0.494406 | -3.760705 |
| 24 | 1.100928 | 2.225647 | 0.606027 | -0.614612 | 0.397443 | -3.685232 |
| 25 | 1.037035 | 2.230862 | 0.656532 | -0.619847 | 0.306433 | -3.611014 |
| 26 | 0.974177 | 2.2367 | 0.707602 | -0.621498 | 0.221403 | -3.538785 |
| 27 | 0.912331 | 2.243122 | 0.758737 | -0.61947 | 0.142424 | -3.469103 |
| 28 | 0.851482 | 2.250095 | 0.809424 | -0.613768 | 0.069584 | -3.40235 |
| 29 | 0.85917 | -0.604496 | 0.791623 | 2.257596 | 0.002967 | -3.338745 |
| 30 | 0.907531 | -0.591844 | 0.732751 | 2.265607 | -0.057377 | -3.278366 |
| 31 | 0.954132 | -0.576066 | 0.674871 | 2.274116 | -0.111444 | -3.221182 |
| 32 | 0.998672 | -0.557456 | 0.617993 | 2.283118 | -0.159289 | -3.16708 |
| 33 | 1.04093 | -0.536322 | 0.562131 | 2.292611 | -0.201024 | -3.115903 |
| 34 | 1.080752 | -0.51297 | 0.507305 | 2.302599 | -0.23681 | -3.067466 |
| 35 | 1.118044 | -0.487688 | 0.453543 | 2.313091 | -0.266845 | -3.02158 |
| 36 | 1.152756 | -0.460742 | 0.400873 | 2.3241 | -0.291359 | -2.978061 |
| 37 | 1.184873 | -0.432368 | 0.349334 | 2.335641 | -0.310597 | -2.936737 |
| 38 | 1.214404 | -0.402775 | 0.298967 | 2.347735 | -0.324811 | -2.897452 |
| 39 | 1.241375 | -0.372144 | 0.24982 | 2.360405 | -0.33426 | -2.860067 |
| 40 | 1.26582 | -0.340632 | 0.201948 | 2.373678 | -0.339195 | -2.824458 |
| 41 | 1.28778 | -0.308375 | 0.155411 | 2.387584 | -0.339864 | -2.790516 |
| 42 | 1.307296 | -0.275492 | 0.110278 | 2.402155 | -0.336505 | -2.758146 |
| 43 | 1.324409 | -0.242085 | 0.066621 | 2.417424 | -0.329346 | -2.727261 |
| 44 | 1.339158 | -0.208242 | 0.024524 | 2.433428 | -0.318605 | -2.697787 |
| 45 | 1.351577 | -0.174044 | -0.015925 | 2.450204 | -0.30449 | -2.669654 |
| 46 | 1.361695 | -0.139561 | -0.05463 | 2.467793 | -0.287196 | -2.6428 |
| 47 | 1.369539 | -0.104857 | -0.091486 | 2.486234 | -0.266908 | -2.617169 |
| 48 | 1.375128 | -0.069993 | -0.126381 | 2.505568 | -0.243804 | -2.592706 |
| 49 | 1.378475 | -0.035022 | -0.159193 | 2.525838 | -0.218047 | -2.569362 |
| 50 | 1.37959 | 0 | -0.189795 | 2.547088 | -0.189795 | -2.547088 |
| 51 | 1.378475 | 0.035022 | -0.159193 | -2.525838 | -0.218047 | 2.569362 |
| 52 | 1.375128 | 0.069993 | -0.126381 | -2.505568 | -0.243804 | 2.592706 |
| 53 | 1.369539 | 0.104857 | -0.091486 | -2.486234 | -0.266908 | 2.617169 |
| 54 | 1.361695 | 0.139561 | -0.05463 | -2.467793 | -0.287196 | 2.6428 |
| 55 | 1.351577 | 0.174044 | -0.015925 | -2.450204 | -0.30449 | 2.669654 |
| 56 | 1.339158 | 0.208242 | 0.024524 | -2.433428 | -0.318605 | 2.697787 |
| 57 | 1.324409 | 0.242085 | 0.066621 | -2.417424 | -0.329346 | 2.727261 |
| 58 | 1.307296 | 0.275492 | 0.110278 | -2.402155 | -0.336505 | 2.758146 |
| 59 | 1.28778 | 0.308375 | 0.155411 | -2.387584 | -0.339864 | 2.790516 |
| 60 | 1.26582 | 0.340632 | 0.201948 | -2.373678 | -0.339195 | 2.824458 |
| 61 | 1.241375 | 0.372144 | 0.24982 | -2.360405 | -0.33426 | 2.860067 |
| 62 | 1.214404 | 0.402775 | 0.298967 | -2.347735 | -0.324811 | 2.897452 |
| 63 | 1.184873 | 0.432368 | 0.349334 | -2.335641 | -0.310597 | 2.936737 |
| 64 | 1.152756 | 0.460742 | 0.400873 | -2.3241 | -0.291359 | 2.978061 |
| 65 | 1.118044 | 0.487688 | 0.453543 | -2.313091 | -0.266845 | 3.02158 |
| 66 | 1.080752 | 0.51297 | 0.507305 | -2.302599 | -0.23681 | 3.067466 |
| 67 | 1.04093 | 0.536322 | 0.562131 | -2.292611 | -0.201024 | 3.115903 |
| 68 | 0.998672 | 0.557456 | 0.617993 | -2.283118 | -0.159289 | 3.16708 |
| 69 | 0.954132 | 0.576066 | 0.674871 | -2.274116 | -0.111444 | 3.221182 |
| 70 | 0.907531 | 0.591844 | 0.732751 | -2.265607 | -0.057377 | 3.278366 |
| 71 | 0.85917 | 0.604496 | 0.791623 | -2.257596 | 0.002967 | 3.338745 |
| 72 | 0.851482 | -2.250095 | 0.809424 | 0.613768 | 0.069584 | 3.40235 |
| 73 | 0.912331 | -2.243122 | 0.758737 | 0.61947 | 0.142424 | 3.469103 |
| 74 | 0.974177 | -2.2367 | 0.707602 | 0.621498 | 0.221403 | 3.538785 |
| 75 | 1.037035 | -2.230862 | 0.656532 | 0.619847 | 0.306433 | 3.611014 |
| 76 | 1.100928 | -2.225647 | 0.606027 | 0.614612 | 0.397443 | 3.685232 |
| 77 | 1.165886 | -2.221106 | 0.556547 | 0.605978 | 0.494406 | 3.760705 |
| 78 | 1.231948 | -2.217301 | 0.59735 | 3.836532 | 0.508489 | 0.594196 |
| 79 | 1.299164 | -2.214309 | 0.706354 | 3.911657 | 0.462171 | 0.579565 |
| 80 | 1.367595 | -2.212222 | 0.821532 | 3.984878 | 0.417838 | 0.5624 |
| 81 | 1.437312 | -2.211153 | 0.943002 | 4.05486 | 0.375662 | 0.543013 |
| 82 | 1.508399 | -2.211243 | 1.070846 | 4.120133 | 0.335754 | 0.521701 |
| 83 | 1.580956 | -2.212659 | 1.20506 | 4.179101 | 0.298179 | 0.498737 |
| 84 | 1.655093 | -2.21561 | 1.345494 | 4.230043 | 0.262962 | 0.474361 |
| 85 | 1.730933 | -2.22035 | 1.491788 | 4.271121 | 0.230105 | 0.448784 |
| 86 | 1.808611 | -2.227192 | 1.643301 | 4.300408 | 0.199591 | 0.422189 |
| 87 | 1.888267 | -2.236519 | 1.799035 | 4.315921 | 0.171392 | 0.394731 |
| 88 | 1.970042 | -2.248805 | 1.957559 | 4.315693 | 0.145472 | 0.366543 |
| 89 | 2.116944 | 4.29786 | 2.054064 | -2.264634 | 0.121793 | 0.337737 |
| 90 | 2.274707 | 4.260799 | 2.140429 | -2.284726 | 0.100318 | 0.308408 |
| 91 | 2.427794 | 4.203292 | 2.22917 | -2.309967 | 0.081009 | 0.278638 |
| 92 | 2.572612 | 4.12473 | 2.320207 | -2.341437 | 0.063831 | 0.248497 |
| 93 | 2.705139 | 4.02534 | 2.413275 | -2.380442 | 0.048751 | 0.218046 |
| 94 | 2.821124 | 3.9064 | 2.50782 | -2.42852 | 0.035741 | 0.187337 |
| 95 | 2.916411 | 3.77042 | 2.602853 | -2.487427 | 0.024776 | 0.156417 |
| 96 | 2.987364 | 3.621202 | 2.696774 | -2.559041 | 0.015833 | 0.125327 |
| 97 | 3.031375 | 3.463726 | 2.787186 | -2.645175 | 0.008896 | 0.094107 |
| 98 | 3.047349 | 3.303789 | 2.870769 | -2.747239 | 0.00395 | 0.06279 |
| 99 | 3.036029 | 3.147397 | 2.94331 | -2.865782 | 0.000987 | 0.031411 |
| 100 | 3 | 3 | 3 | -3 | 0 | 0 |

# Representative C code and files

Note that function dependencies from each file are noted at the top of each respective .c file.

## Question 1

The files are below in order lin\_root.c, quad\_roots.c, and prog\_1.c

### lin\_root.c – THIS IS THE VERSION I WANT TESTED

*#include <stdio.h>*

*#include <math.h>*

*/\* Computes the root for a linear equation \*/*

**int** lin\_root**(double** a1**,** **double** a0**,** **double\*** r**)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

**if** **(**a1**==0){**

**if** **(**a0**==0){**

**return(0);** *// any number is a root*

**}** **else{**

**return(-1);** *// contradictory*

**}**

**}**

**\***r **=** a0 **==** **0** **?** **0** **:** **-**a0**/**a1**;** *// real root*

**return(1);**

**}**

### quad\_roots.c – THIS IS THE VERSION I WANT TESTED

*#include <stdio.h>*

*#include <math.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** lin\_root**(double,** **double,** **double** **\*);**

*/\* -Functions-implemented-in-current-file------------------------------------- \*/*

**int** sgn**(double);**

**void** order\_2**(double** **\*,** **double** **\*);**

**int** quad\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*);**

**void** print\_statements**();**

*/\* --------------------------------------------------------------------------- \*/*

**int** quad\_roots**(double** a2**,** **double** a1**,** **double** a0**,** **double** **\*** r1**,** **double** **\*** r2**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** two**=2,**zero**=0,**one**=1,**a1\_half**,**e**,**coef**;**

**int** d\_sgn**,**add\_sgn**;**

**if** **(**a2**==0)** **{**

**return** **(-2** **+** lin\_root**(**a1**,**a0**,**r1**));**

**}** **else** **if** **(**a0**==0)** **{**

lin\_root**(**a2**,**a1**,**r1**);**

**\***r2 **=** zero**;**

order\_2**(**r1**,**r2**);**

**return** **\***r1 **==** **\***r2 **?** **1** **:** **2;**

**}**

a1\_half **=** a1**/**two**;**

**if** **(**fabs**(**a1\_half**)>=**fabs**(**a0**)){**

e**=**one**-((**a0**/**a1\_half**)\*(**a2**/**a1\_half**));**

coef **=** fabs**(**a1\_half**);**

d\_sgn **=** sgn**(**e**);**

**}** **else** **{**

e**=(**a1\_half**\*(**a1\_half**/**a0**))-**a2**;**

coef **=** sqrt**(**fabs**(**a0**));**

d\_sgn **=** sgn**(**a0**)\***sgn**(**e**);**

**}**

**if** **(**d\_sgn**==-1){**

**\***r1**=(-**a1\_half**)/(**a2**);** *//Real Part*

**\***r2**=**coef**\***sqrt**(**fabs**(**e**))/**a2**;** *//Imaginary Part*

**return(0);**

**}** **else** **if** **(**d\_sgn**==0){**

**\***r1**=(-**a1\_half**)/(**a2**);**

**\***r2**=(-**a1\_half**)/(**a2**);**

**return(1);**

**}** **else** **{**

add\_sgn **=** a1 **==** zero **?** **1** **:** sgn**(**a1**);**

**\***r1**=(-**a1\_half**-**add\_sgn**\***coef**\***sqrt**(**fabs**(**e**)))/**a2**;**

**\***r2**=(**a0 **/** **\***r1**)/**a2**;**

order\_2**(**r1**,**r2**);**

**return(2);**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** order\_2**(double** **\*** r1**,** **double** **\*** r2**)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

**if** **(\***r2**>\***r1**)** **{**

**double** dummy **=** **\***r1**;**

**\***r1 **=** **\***r2**;**

**\***r2 **=** dummy**;**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**int** sgn**(double** x**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** zero **=** **0;**

**if** **(**x**<**zero**){return(-1);}**

**else** **if** **(**x**>**zero**){return(1);}**

**return(0);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** print\_statements**(){**

*/\* Bhageria, Yadu, M3SC \*/*

printf**(** " Name: Bhageria, Yadu"**);**

printf**(**"\n CID: 00733164"**);**

printf**(**"\n Course Code: M3SC"**);**

printf**(**"\nEmail Address: yrb13@ic.ac.uk"**);**

printf**(**"\n Time: %s "**,**\_\_TIME\_\_**);**

printf**(**"\n Date: %s "**,**\_\_DATE\_\_**);**

printf**(**"\n \n"**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

### prog\_1.c

*#include <stdio.h>*

*#include <math.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** quad\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*);**

**void** print\_statements**();** *//contained in the quad\_roots.c file*

*/\* --------------------------------------------------------------------------- \*/*

**int** main**(void)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

print\_statements**();**

**double** a2**,**a1**,**a0**,**r1**,**r2**;**

**int** quad\_case**;**

printf**(**"Enter coefficients of Linear Equation a2\*x^2+a1\*x+a0=0\n"**);**

printf**(**"in the order a2,a1,a0, seperated by spaces: "**);**

scanf**(**"%lf %lf %lf"**,&**a2**,&**a1**,&**a0**);**

quad\_case **=** quad\_roots**(**a2**,**a1**,**a0**,&**r1**,&**r2**);**

**switch** **(**quad\_case**)** **{**

**case** **-3:** printf**(**"The values of a2, a1, a0 resulted in a contradictory equation\n"**);** **break;**

**case** **-2:** printf**(**"a2=a1=a0=0 so any real number is a root as the equation is tautological\n"**);** **break;**

**case** **-1:** printf**(**"a2=0 so we are dealing with a linear equation with one root\nr1 = %.10g\n"**,** r1**);** **break;**

**case** **0:** printf**(**"The roots are complex.\nr1 = %.10g + %.10gi, r2 = %.10g - %.10gi \n"**,** r1**,**r2**,**r1**,**r2**);** **break;**

**case** **1:** printf**(**"There are repeated real roots.\n r1 = r2 = %.10g \n"**,** r1**);** **break;**

**case** **2:** printf**(**"There are two real roots.\nr1 = %.10g, r2 = %.10g \n"**,** r1**,**r2**);** **break;**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

## Question 2

We use the same lin\_root.c and quad\_roots.c file as in Question 1.

The files in this part are rcubic\_roots.c and prog\_2.c – presented below in that order.

### rcubic\_roots.c

*#include <stdio.h>*

*#include <math.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**void** order\_2**(double** **\*,** **double** **\*);**

**int** quad\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*);**

*/\* -Functions-implemented-in-current-file------------------------------------- \*/*

**void** order\_3**(double** **\*,** **double** **\*,** **double** **\*);**

**void** set\_starting\_y0**(double,** **double** **\*);**

**double** iterate\_y**(double,** **double);**

**double** newton\_rapheson**(double);**

**int** roots\_return**(double,** **double,** **double,** **double** **\*,** **double** **\*,** **double** **\*);**

**int** rcubic\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*,** **double** **\*);**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Cubic Solver using the Newton-Rapheson approximation for the first real root \*/*

**int** rcubic\_roots**(double** a2**,** **double** a1**,** **double** a0**,** **double** **\***r1**,** **double** **\***r2**,** **double** **\***r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** zero**=0,**one**=1,**two**=2,**three**=3;**

**double** alpha**,**beta**,**p**,**y**;**

*/\* a2=a1=a0=0. So x^3 = 0 => r1=r2=r3=0 \*/*

**if** **(**a2 **==** zero **&** a1 **==** zero **&** a0 **==** zero**)** **{**

**\***r1**=**zero**;**

**return(1);**

**}**

beta**=-**a2**/**three**;**

alpha**=**cbrt**(**two**\***beta**\***beta**\***beta **-** a1**\***beta **-** a0**);**

*/\*trivial case where we only need to solve a quadratic equation of the form x(x^2 + a2\*x + a1) = 0 \*/*

**if** **(**alpha **==** zero**){**

**\***r1 **=** beta**;**

**return(**roots\_return**(**one**,-**two**\***beta**,-**two**\***beta**\***beta**+**a1**,**r1**,**r2**,**r3**));**

**}** **else** **{**

p**=(**three**\***beta**\***beta**-**a1**)/(**alpha**\***alpha**);**

y**=**newton\_rapheson**(**p**);**

**\***r1**=** alpha**\***y **+** beta**;**

**}**

**return(**roots\_return**(**one**,**a2**+(\***r1**),-**a0**/(\***r1**),**r1**,**r2**,**r3**));**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* finds yn+1 for given yn and p \*/*

**double** iterate\_y**(double** p**,** **double** y**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** one**=1,**three**=3;**

**return(**y**-(**y**\***y**\***y**-**p**\***y**-**one**)/(**three**\***y**\***y**-**p**));**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Sets starting value of yn based on p \*/*

**void** set\_starting\_y0**(double** p**,** **double** **\***y0**){**

*/\* Bhageria, Yadu, M3SC \*/*

**if** **(**p**>11.0/3.0){\***y0**=**sqrt**(**p**);}**

**else** **if** **(**p**<-1.92)** **{\***y0**=-1.0/**p**;}**

**else** **{\***y0**=1.0** **+** p**/3.0** **-** **(**p**\***p**\***p**)/81.0;}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Computes first real root using the Newton-Rapheson approximation method \*/*

**double** newton\_rapheson**(double** p**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** zero**=0,**one**=1,**yn**,**yn1**,**diff**,**diff1**;**

**int** count**=1;**

**if** **(**p**==**zero**){**yn1**=**one**;}**

**else** **{**

set\_starting\_y0**(**p**,&**yn**);**

yn1 **=** iterate\_y**(**p**,**yn**);**

**while** **(**yn**!=**yn1 **&&** **(** count**<4** **||** diff1 **<** diff **))** **{**

yn**=**yn1**;**

yn1 **=** iterate\_y**(**p**,**yn**);**

diff1 **=** fabs**(**yn**-**yn1**);**

diff **=** diff1**;**

count **=** count **+** **1;**

*/\* printf("yn = %.10g, yn+1 = %.10g\n", yn,yn1); //to test \*/*

**}**

**}**

**return(**yn1**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Computes r2 and r3 once r1 has been found. Returns an integer corresponding to the number of real roots found \*/*

**int** roots\_return**(double** a2**,** **double** a1**,** **double** a0**,** **double** **\***r1**,** **double** **\***r2**,** **double** **\***r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

**int** quad\_case **=** quad\_roots**(**a2**,**a1**,**a0**,**r2**,**r3**);**

**switch** **(**quad\_case**)** **{**

**case** **0:** **return(0);**

**case** **1:** **if** **(**r1**==**r2**){return(1);}** **else** **{return(2);}**

**case** **2:** order\_3**(**r1**,**r2**,**r3**);** **if** **(**r1**==**r2**){return(2);}** **else** **if** **(**r2**==**r3**){return(2);}** **else** **{return(3);}**

*/\* In case 2 r1 cannot equal r3 as r1,r2,r3 are ordered and hence we would have dealt with that in case 1 \*/*

**}**

**return(-1);** */\* for errors \*/*

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** order\_3**(double** **\***r1**,** **double** **\***r2**,** **double** **\***r3**)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

order\_2**(**r1**,**r3**);**

order\_2**(**r1**,**r2**);**

order\_2**(**r2**,**r3**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

### prog\_2.c

*#include <stdio.h>*

*#include <math.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** rcubic\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*,** **double** **\*);**

**void** print\_statements**();** *//contained in the quad\_roots.c file*

*/\* -Functions-implemented-in-current-file------------------------------------- \*/*

**double** f\_x**(double,** **double,** **double,** **double);**

*/\* --------------------------------------------------------------------------- \*/*

**int** main**(void)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

print\_statements**();**

**double** a2**,**a1**,**a0**,**r1**,**r2**,**r3**;**

**int** cubic\_case**;**

printf**(**"Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0\n"**);**

printf**(**"in the order a2,a1,a0, separated by spaces: "**);**

scanf**(**"%lf %lf %lf"**,&**a2**,&**a1**,&**a0**);**

printf**(**"\n"**);**

cubic\_case **=** rcubic\_roots**(**a2**,**a1**,**a0**,&**r1**,&**r2**,&**r3**);**

**switch** **(**cubic\_case**)** **{**

**case** **0:** printf**(**"There is one real root (r1) and two complex roots (r2,r3).\n r1 = %.10g, r2 = %.10g + %.10gi, r3 = %.10g - %.10gi \n"**,** r1**,**r2**,**r3**,**r2**,**r3**);** printf**(**" f(r1) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**));** **break;**

**case** **1:** printf**(**"There are triple repeated real roots.\n r1 = r2 = r3 = %.10g \n"**,** r1**);** printf**(**"f(r1) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**));** **break;**

**case** **2:** printf**(**"There is a pair of repeated real roots.\n r1 = %.10g, r2 = %.10g, r3 = %.10g \n"**,** r1**,**r2**,**r3**);** printf**(**" f(r1) = %.10g\n f(r2) = %.10g\n f(r3) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**),**f\_x**(**a2**,**a1**,**a0**,**r2**),**f\_x**(**a2**,**a1**,**a0**,**r3**));** **break;**

**case** **3:** printf**(**"There are three distinct real roots.\n r1 = %g, r2 = %g, r3 = %g \n"**,** r1**,**r2**,**r3**);** printf**(**" f(r1) = %.10g\n f(r2) = %.10g\n f(r3) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**),**f\_x**(**a2**,**a1**,**a0**,**r2**),**f\_x**(**a2**,**a1**,**a0**,**r3**));** **break;**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**double** f\_x**(double** a2**,** **double** a1**,** **double** a0**,** **double** x**){**

**return(**x**\***x**\***x **+** a2**\***x**\***x **+** a1**\***x **+** a0**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

## Question 3

This question again uses the lin\_root.c and quad\_roots.c file from Question 1.

It has two new files called rcubic\_roots\_optimized.c and prog\_3.c as below

### rcubic\_roots\_optimized.c – THIS IS THE VERSION I WANT TESTED

*#include <float.h> //for DBL\_EPSILON*

*#include <stdio.h>*

*#include <math.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**void** order\_2**(double** **\*,** **double** **\*);**

**int** quad\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*);**

*/\* -Functions-implemented-in-current-file------------------------------------- \*/*

**void** order\_3**(double** **\*,** **double** **\*,** **double** **\*);**

**void** set\_starting\_y0**(double,** **double** **\*);**

**double** iterate\_y**(double,** **double** **);**

**double** newton\_rapheson**(double** **);**

**int** roots\_of\_unity**(double,** **double** **\*,** **double** **\*,** **double** **\*);**

**int** roots\_return**(double,** **double,** **double,** **double** **\*,** **double** **\*,** **double** **\*);**

**int** rcubic\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*,** **double** **\*);**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Cubic Solver using the Newton-Rapheson approximation for the first real root\*/*

**int** rcubic\_roots**(double** a2**,** **double** a1**,** **double** a0**,** **double** **\***r1**,** **double** **\***r2**,** **double** **\***r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** zero**=0,**one**=1,**two**=2,**three**=3;**

**double** alpha**,**beta**,**p**,**y**;**

**double** p2**=3.0\***cbrt**(0.25),**y2 **=** **-**cbrt**(0.5);;**

**if** **(**a2 **==** zero **&&** a1 **==** zero **&&** a0 **==** zero**)** **{** */\* a2=a1=a0=0. So x^3 = 0 => r1=r2=r3=0 \*/*

**\***r1**=**zero**;**

**return(1);**

**}** **else** **if** **(**a2 **==** zero **&&** a1 **==** zero**)** **{** */\* Part 3 i \*/*

**return(**roots\_of\_unity**(**cbrt**(-**a0**),**r1**,**r2**,**r3**));**

**}**

beta**=-**a2**/**three**;**

alpha**=**cbrt**(**two**\***beta**\***beta**\***beta **-** a1**\***beta **-** a0**);**

p**=(**three**\***beta**\***beta**-**a1**)/(**alpha**\***alpha**);**

**if** **(**a0**==**zero**){** */\* Part 3 ii \*/*

**\***r1 **=** zero**;**

**return(**roots\_return**(**one**,**a2**,**a1**,**r1**,**r2**,**r3**));**

**}** **else** **if** **(**a0**==**a1**\***a2**){** */\* Part 3 iii \*/*

**\***r1 **=** **-**a2**;**

**if** **(**a1**>**zero**){**

**\***r2 **=** zero**;**

**\***r3 **=** sqrt**(**a1**);**

**return(0);**

**}** **else** **{**

**\***r2 **=** sqrt**(-**a1**);**

**\***r3 **=** **-**sqrt**(-**a1**);**

order\_3**(**r1**,**r2**,**r3**);**

**return** **\***r1**==\***r2 **||** **\***r2**==\***r3 **?** **2** **:** **3;**

**}**

**}** **else** **if** **(** fabs**(3.0\***a1**-**a2**\***a2**)<**DBL\_EPSILON **&&** fabs**(27.0\***a0**-**a2**\***a2**\***a2**)<**DBL\_EPSILON **){** */\* Part 3 iv \*/*

**\***r1 **=** **-**a2**/3.0;**

**return(1);**

**}** **else** **if** **(**alpha **==** zero**){** */\* When our degeneration doesn't makes sense \*/*

**\***r1 **=** beta**;**

**return(**roots\_return**(**one**,-**two**\***beta**,-**two**\***beta**\***beta**+**a1**,**r1**,**r2**,**r3**));**

**}** **else** **if** **(** fabs**(** p **-** p2**)** **<** **10.0\***DBL\_EPSILON **){** */\* Part 3 v: we choose 10.0\*DBL\_EPSILON as we loose accuracy in the calculation of alpha, beta, and p. \*/*

**\***r1 **=** alpha**\***y2 **+** beta**;**

**\***r2 **=** alpha**\***y2 **+** beta**;**

**\***r3 **=** alpha**/(**y2**\***y2**)** **+** beta**;**

order\_3**(**r1**,**r2**,**r3**);**

**return** **\***r1**==\***r2 **||** **\***r2**==\***r3 **?** **2** **:** **3;**

**}** **else** **if** **(**p**==**zero**)** **{**

roots\_of\_unity**(**one**,**r1**,**r2**,**r3**);**

**\***r1 **=** alpha **\*** **(\***r1**)** **+** beta**;**

**\***r2 **=** alpha **\*** **(\***r2**)** **+** beta**;**

**return(0);**

**}** **else** **{**

y**=**newton\_rapheson**(**p**);**

**\***r1**=** alpha**\***y **+** beta**;**

**}**

**return(**roots\_return**(**one**,**a2**+(\***r1**),-**a0**/(\***r1**),**r1**,**r2**,**r3**));**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* finds yn+1 for given yn and p \*/*

**double** iterate\_y**(double** p**,** **double** y**){**

*/\* Bhageria, Yadu, M3SC \*/*

**return(**y**-(**y**\***y**\***y**-**p**\***y**-1.0)/(3.0\***y**\***y**-**p**));**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Sets starting value of yn based on p \*/*

**void** set\_starting\_y0**(double** p**,** **double** **\***y0**){**

*/\* Bhageria, Yadu, M3SC \*/*

**if** **(**p**>11.0/3.0){\***y0**=**sqrt**(**p**);}**

**else** **if** **(**p**<-1.92)** **{\***y0**=-1.0/**p**;}**

**else** **{\***y0**=1.0** **+** p**/3.0** **-** **(**p**\***p**\***p**)/81.0;}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Computes first real root using the Newton-Rapheson approximation method \*/*

**double** newton\_rapheson**(double** p**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** zero**=0,**one**=1,**yn**,**yn1**,**diff**,**diff1**;**

**int** count**=1;**

**if** **(**p**==**zero**){**yn1**=**one**;}**

**else** **{**

set\_starting\_y0**(**p**,&**yn**);**

yn1 **=** iterate\_y**(**p**,**yn**);**

**while** **(**yn**!=**yn1 **&&** **(** count**<4** **||** diff1 **<** diff **))** **{**

yn**=**yn1**;**

yn1 **=** iterate\_y**(**p**,**yn**);**

diff1 **=** fabs**(**yn**-**yn1**);**

diff **=** diff1**;**

count **=** count **+** **1;**

*/\* printf("yn = %.10g, yn+1 = %.10g\n", yn,yn1); //to test \*/*

**}**

**}**

**return(**yn1**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

*/\* Computes r2 and r3 once r1 has been found. Returns an integer corresponding to the number of real roots found \*/*

**int** roots\_return**(double** a2**,** **double** a1**,** **double** a0**,** **double** **\***r1**,** **double** **\***r2**,** **double** **\***r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

**int** quad\_case **=** quad\_roots**(**a2**,**a1**,**a0**,**r2**,**r3**);**

**switch** **(**quad\_case**)** **{**

**case** **0:** **return(0);**

**case** **1:** **return** r1**==**r2 **?** **1** **:** **2;**

**case** **2:** order\_3**(**r1**,**r2**,**r3**);** **if** **(**r1**==**r2**){return(2);}** **else** **{return** r2**==**r3 **?** **2** **:** **3;}**

*/\* In case 2 r1 cannot equal r3 as r1,r2,r3 are ordered and hence we would have dealt with that in case 1 \*/*

**}**

**return(-1);** */\* for errors \*/*

**}**

*/\* --------------------------------------------------------------------------- \*/*

**int** roots\_of\_unity**(double** r**,** **double** **\***r1**,** **double** **\***r2**,** **double** **\***r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

**\***r1 **=** r**;** *//Real Root*

**\***r2 **=** r**/2.0;** *//Real Part of Complex Roots*

**\***r3 **=** fabs**(**r**)\***sqrt**(3.0)/2.0;** *//Positive Imaginary Part of Complex Roots*

**return(0);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** order\_3**(double** **\***r1**,** **double** **\***r2**,** **double** **\***r3**)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

order\_2**(**r1**,**r3**);**

order\_2**(**r1**,**r2**);**

order\_2**(**r2**,**r3**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

### prog\_3.c

*#include <stdio.h>*

*#include <math.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** rcubic\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*,** **double** **\*);**

**void** print\_statements**();** *//contained in the quad\_roots.c file*

*/\* -Functions-implemented-in-current-file------------------------------------- \*/*

**double** f\_x**(double,** **double,** **double,** **double);**

*/\* --------------------------------------------------------------------------- \*/*

**int** main**(void)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

print\_statements**();**

**double** a2**,**a1**,**a0**,**r1**,**r2**,**r3**;**

**int** cubic\_case**;**

printf**(**"Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0\n"**);**

printf**(**"in the order a2,a1,a0, separated by spaces: "**);**

scanf**(**"%lf %lf %lf"**,&**a2**,&**a1**,&**a0**);**

printf**(**"\n"**);**

*//double alp=0.2135,bet=0.01709,R=0.0820578,T,P;*

*//P = alp/(27.0\*bet\*bet);*

*//T = (P/R)\*(sqrt(3.0\*alp/P)-bet);*

*//printf(" T: %f\n", T);*

*//printf(" P: %f\n", P);*

*//printf("\n");*

*//a2=-bet-R\*T/P;*

*//a1=alp/P;*

*//a0=-(alp\*bet)/P;*

cubic\_case **=** rcubic\_roots**(**a2**,**a1**,**a0**,&**r1**,&**r2**,&**r3**);**

**switch** **(**cubic\_case**)** **{**

**case** **0:** printf**(**"There is one real root (r1) and two complex roots (r2,r3).\n r1 = %.10g, r2 = %.10g + %.10gi, r3 = %.10g - %.10gi \n"**,** r1**,**r2**,**r3**,**r2**,**r3**);** printf**(**" f(r1) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**));** **break;**

**case** **1:** printf**(**"There are triple repeated real roots.\n r1 = r2 = r3 = %.10g \n"**,** r1**);** printf**(**"f(r1) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**));** **break;**

**case** **2:** printf**(**"There is a pair of repeated real roots.\n r1 = %.10g, r2 = %.10g, r3 = %.10g \n"**,** r1**,**r2**,**r3**);** printf**(**" f(r1) = %.10g\n f(r2) = %.10g\n f(r3) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**),**f\_x**(**a2**,**a1**,**a0**,**r2**),**f\_x**(**a2**,**a1**,**a0**,**r3**));** **break;**

**case** **3:** printf**(**"There are three distinct real roots.\n r1 = %g, r2 = %g, r3 = %g \n"**,** r1**,**r2**,**r3**);** printf**(**" f(r1) = %.10g\n f(r2) = %.10g\n f(r3) = %.10g\n"**,** f\_x**(**a2**,**a1**,**a0**,**r1**),**f\_x**(**a2**,**a1**,**a0**,**r2**),**f\_x**(**a2**,**a1**,**a0**,**r3**));** **break;**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**double** f\_x**(double** a2**,** **double** a1**,** **double** a0**,** **double** x**){**

**return(**x**\***x**\***x **+** a2**\***x**\***x **+** a1**\***x **+** a0**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

## Question 4

This section uses the rcubic\_roots\_optimized.c file along with the quad\_roots.c and lin\_roots.c file from Questions 3 and 1 respectively.

The main program file for this section is as below:

### prog\_4.c

*#include <stdio.h>*

*#include <math.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** lin\_root**(double,** **double,** **double** **\*);**

**int** rcubic\_roots**(double,** **double,** **double,** **double** **\*,** **double** **\*,** **double** **\*);**

**void** print\_statements**();** *//contained in the quad\_roots.c file*

*/\* --------------------------------------------------------------------------- \*/*

**int** main**(void)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

print\_statements**();**

**double** a2**,**a1**,**a0**,**r1**,**r2**,**r3**;**

**double** alp**=0.2135,**bet**=0.01709,**P**,**R**=0.0820578,**V\_ideal**;**

**int** cubic\_case**,**T**;**

printf**(**" T, P, V\_ideal, V\_vdw1, V\_vdw2, V\_vdw3\n"**);**

**for** **(**T**=40;**T**<=50;**T**=**T**+5){**

**for** **(**P**=1;**P**<=30;**P**=**P**+0.5){**

a2**=-**bet**-**R**\***T**/**P**;**

a1**=**alp**/**P**;**

a0**=-(**alp**\***bet**)/**P**;**

lin\_root**(**P**,-**R**\***T**,&**V\_ideal**);**

cubic\_case **=** rcubic\_roots**(**a2**,**a1**,**a0**,&**r1**,&**r2**,&**r3**);**

**switch** **(**cubic\_case**)** **{**

**case** **0:** printf**(**"%d, %5.1f, %9.5f, %9.5f,\n"**,**T**,**P**,**V\_ideal**,**r1**);** **break;**

**case** **1:** printf**(**"%d, %5.1f, %9.5f, %9.5f,%9.5f,%9.5f, triple repeated roots\n"**,**T**,**P**,**V\_ideal**,**r1**,**r2**,**r3**);** **break;**

**case** **2:** printf**(**"%d, %5.1f, %9.5f, %9.5f,%9.5f,%9.5f, two roots repeated\n"**,**T**,**P**,**V\_ideal**,**r1**,**r2**,**r3**);** **break;**

**case** **3:** printf**(**"%d, %5.1f, %9.5f, %9.5f, %9.5f, %9.5f\n"**,**T**,**P**,**V\_ideal**,**r1**,**r2**,**r3**);** **break;**

**}**

**}**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

## Question 5

This part of the project utilizes only a prog\_5.c file given below

### prog\_5.c

*#include <stdio.h>*

**int** main**(void)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

printf**(**"\n What is this? : \234"**);**

printf**(**"\n Time : %s "**,**\_\_TIME\_\_**);**

printf**(**"\n Date : %s "**,**\_\_DATE\_\_**);**

printf**(**"\n Euro : \xE2\x82\xAC"**);** *//Check if this works in Windows*

printf**(**"\n Euro : \u20AC"**);**

printf**(**"\n \n"**);**

**return(0);**

**}**

## Question 6

Here we do not uuse the lin\_root.c and quad\_roots.c files from before as they cannot handle complex numbers. Instead the files used for this part are:

quad\_roots\_complex.c – solves a linear or quadratic equation for complex coefficients

rcubic\_roots\_complex.c – solves the cubic equation for complex coefficients

prog\_mastery.c – allows for an input of and outputs the roots of the equation

prog\_m.c – outputs a table of roots for the given cases in .csv format.

### quad\_roots\_complex.c

*#include <stdio.h>*

*#include <math.h>*

*#include <complex.h>*

*/\* -Functions-implemented-in-current-file------------------------------------- \*/*

**int** lin\_root**(double** complex**,** **double** complex**,** **double** complex **\*);**

**int** quad\_roots**(double** complex**,** **double** complex**,** **double** complex**,** **double** complex **\*,** **double** complex **\*);**

**void** print\_statements**();**

*/\* --------------------------------------------------------------------------- \*/*

**int** lin\_root**(double** complex a1**,** **double** complex a0**,** **double** complex **\*** r**)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

**double** complex zero**=0;**

**if** **(**a1**==**zero**){**

**if** **(**a0**==**zero**){**

**return(0);** *// any number is a root*

**}** **else{**

**return(-1);** *// contradictory*

**}**

**}** **\***r **=** **-**a0**/**a1**;** *// real root*

**return(1);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**int** quad\_roots**(double** complex a2**,** **double** complex a1**,** **double** complex a0**,** **double** complex **\*** r1**,** **double** complex **\*** r2**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** complex four**=4,**two**=2,**zero**=0,**d**,**dr**;**

**if** **(**a2**==**zero**)** **{**

**return** **(-2** **+** lin\_root**(**a1**,**a0**,**r1**));**

**}** **else** **if** **(**a0**==**zero**)** **{**

lin\_root**(**a2**,**a1**,**r1**);**

**\***r2**=**zero**;**

**return(2);**

**}**

d**=**a1**\***a1**-**four**\***a2**\***a0**;**

**if** **(**d**==**zero**)** **{**

**\***r1**=(-**a1**)/(**two**\***a2**);**

**\***r2**=(-**a1**)/(**two**\***a2**);**

**return(1);**

**}** **else** **{**

dr**=**csqrt**(**d**);**

**\***r1 **=** **(-**a1**-**dr**)/(**two**\***a2**);**

**\***r2 **=** **(-**a1**+**dr**)/(**two**\***a2**);**

**return(2);**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** print\_statements**(){**

*/\* Bhageria, Yadu, M3SC \*/*

printf**(** " Name: Bhageria, Yadu"**);**

printf**(**"\n CID: 00733164"**);**

printf**(**"\n Course Code: M3SC"**);**

printf**(**"\nEmail Address: yrb13@ic.ac.uk"**);**

printf**(**"\n Time: %s "**,**\_\_TIME\_\_**);**

printf**(**"\n Date: %s "**,**\_\_DATE\_\_**);**

printf**(**"\n \n"**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

### rcubic\_roots\_complex.c

*#include <stdio.h>*

*#include <math.h>*

*#include <complex.h>*

*#include <float.h> //for DBL\_EPSILON*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** quad\_roots**(double** complex**,double** complex**,double** complex**,double** complex **\*,double** complex **\*);**

*/\* -Functions-implemented-in-current-file------------------------------------- \*/*

**double** complex iterate\_z**(double** complex**,** **double** complex**,** **double** complex**,** **double** complex**);**

**double** complex newton\_rapheson**(double** complex**,** **double** complex**,** **double** complex**);**

**void** order\_2**(double** complex **\*,** **double** complex **\*);**

**void** order\_3**(double** complex **\*,** **double** complex **\*,** **double** complex **\*);**

**int** roots\_return**(double** complex **\*,** **double** complex **\*,** **double** complex **\*);**

**int** rcubic\_roots**(double** complex**,** **double** complex**,** **double** complex**,** **double** complex **\*,** **double** complex **\*,** **double** complex **\*);**

**void** direct\_method**(double** complex**,** **double** complex**,** **double** complex**,** **double** complex **\*,** **double** complex **\*,** **double** complex **\*);** *//to comapre with the NR method*

*/\* --------------------------------------------------------------------------- \*/*

**int** rcubic\_roots**(double** complex a2**,** **double** complex a1**,** **double** complex a0**,** **double** complex **\*** r1**,** **double** complex **\*** r2**,** **double** complex **\*** r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** complex zero**=0,**one**=1.0;**

**if** **(**a2 **==** zero **&** a1 **==** zero **&** a0 **==** zero**)** **{**

**\***r1**=**zero**;**

**return(1);**

**}** **else** **if** **(**a2 **==** zero **&&** a1 **==** zero**)** **{** */\* Part 3 i \*/*

**\***r1 **=** cpow**(-**a0**,1.0/3.0);**

**\***r2 **=** **\***r1 **\*** **(0.5** **+** I **\*** sqrt**(3.0)/2.0);**

**\***r3 **=** **\***r2 **\*** **(0.5** **+** I **\*** sqrt**(3.0)/2.0);**

**}** **else** **if** **(**a0 **==** zero**){**

**\***r1 **=** zero**;**

quad\_roots**(**one**,**a2**,**a1**,**r2**,**r3**);**

**}** **else** **if** **(**a0**==**a1**\***a2**){** */\* Part 3 iii \*/*

**\***r1 **=** **-**a2**;**

**\***r2 **=** csqrt**(-**a1**);**

**\***r3 **=** **-**csqrt**(-**a1**);**

**}** **else** **if** **(** **3.0\***a1**==**a2**\***a2 **&&** **27.0\***a0**==**a2**\***a2**\***a2 **){** */\* Part 3 iv \*/*

**\***r1 **=** **-**a2**/3.0;**

**return(1);**

**}** **else** **{**

**\***r1 **=** newton\_rapheson**(**a2**,**a1**,**a0**);**

quad\_roots**(**one**,** a2**+(\***r1**),** **-**a0**/(\***r1**),**r2**,**r3**);**

*//direct\_method(a2,a1,a0,r1,r2,r3);*

**}**

**return(**roots\_return**(**r1**,**r2**,**r3**));**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**double** complex iterate\_z**(double** complex a2**,** **double** complex a1**,** **double** complex a0**,** **double** complex z**){**

*/\* Bhageria, Yadu, M3SC \*/*

**return(**z **-** **(**z**\***z**\***z **+** a2**\***z**\***z **+** a1**\***z **+** a0**)/(3.0\***z**\***z **+** **2.0\***a2**\***z **+** a1**));**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**double** complex newton\_rapheson**(double** complex a2**,** **double** complex a1**,** **double** complex a0**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** complex zn**,**zn1**;**

**int** count**=1;**

zn **=** **1.0** **+** I**;** *//Set starting value*

zn1 **=** iterate\_z**(**a2**,**a1**,**a0**,**zn**);**

**while** **(**cabs**(**zn**-**zn1**)>**DBL\_EPSILON **&&** count**<20** **){**

zn**=**zn1**;**

zn1 **=** iterate\_z**(**a2**,**a1**,**a0**,**zn**);**

count **=** count **+** **1;**

**}** **return(**zn1**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** order\_2**(double** complex **\***r1**,** **double** complex **\***r2**){**

*/\* Bhageria, Yadu, M3SC \*/*

**double** complex dummy**;**

**if** **(** creal**(\***r1**)** **<** creal**(\***r2**)** **)** **{**

dummy **=** **\***r2**;**

**\***r2 **=** **\***r1**;**

**\***r1 **=** dummy**;**

**}** **else** **if** **(** creal**(\***r1**)** **==** creal**(\***r2**)** **){**

**if** **(** cimag**(\***r1**)** **<** cimag**(\***r2**)** **){**

dummy **=** **\***r2**;**

**\***r2 **=** **\***r1**;**

**\***r1 **=** dummy**;**

**}**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** order\_3**(double** complex **\***r1**,** **double** complex **\***r2**,** **double** complex **\***r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

order\_2**(**r1**,**r3**);**

order\_2**(**r1**,**r2**);**

order\_2**(**r2**,**r3**);**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**int** roots\_return**(double** complex **\***r1**,** **double** complex **\***r2**,** **double** complex **\***r3**){**

*/\* Bhageria, Yadu, M3SC \*/*

order\_3**(**r1**,**r2**,**r3**);**

**if** **(\***r1**==\***r2 **&&** **\***r2**==\***r3**){**

**return(1);**

**}** **else** **{**

**return** **(\***r1**==\***r2**)** **||** **(\***r2**==\***r3**)** **?** **2** **:** **3;**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

**void** direct\_method**(double** complex a2**,** **double** complex a1**,** **double** complex a0**,** **double** complex **\*** r1**,** **double** complex **\*** r2**,** **double** complex **\*** r3**){**

**double** complex A**,**B**;**

*//C = -2.0\*a2\*a2\*a2 + 9.0\*a2\*a1 - 27.0\*a0 + 3.0\*sqrt(3.0)\*csqrt(-a2\*a2\*a1\*a1 + 4.0\*a1\*a1\*a1 + 4.0\*a2\*a2\*a2\*a0 - 18.0\*a2\*a1\*a0 + 27.0\*a0\*a0);*

A **=** cpow**(-2.0\***a2**\***a2**\***a2 **+** **9.0\***a2**\***a1 **-** **27.0\***a0 **+** **3.0\***sqrt**(3.0)\***csqrt**(-**a2**\***a2**\***a1**\***a1 **+** **4.0\***a1**\***a1**\***a1 **+** **4.0\***a2**\***a2**\***a2**\***a0 **-** **18.0\***a2**\***a1**\***a0 **+** **27.0\***a0**\***a0**),1.0/3.0)/(3.0\***cbrt**(2));**

B **=** **(-**a2**\***a2 **+** **3.0\***a1**)/(9.0\***A**);**

**\***r1 **=** **-**a2**/3.0** **+** A **-** B**;**

**\***r2 **=** **-**a2**/3.0** **+** **((-1.0** **-** I **\*** sqrt**(3.0))/2.0)\***A **-** **((-1.0** **+** I **\*** sqrt**(3.0))/2.0)\***B**;**

**\***r3 **=** **-**a2**/3.0** **+** **((-1.0** **+** I **\*** sqrt**(3.0))/2.0)\***A **-** **((-1.0** **-** I **\*** sqrt**(3.0))/2.0)\***B**;**

**}**

*/\* --------------------------------------------------------------------------- \*/*

### prog\_mastery.c

*#include <stdio.h>*

*#include <math.h>*

*#include <complex.h>*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** rcubic\_roots**(double** complex**,** **double** complex**,** **double** complex**,** **double** complex **\*,** **double** complex **\*,** **double** complex **\*);**

**void** print\_statements**();**

*/\* --------------------------------------------------------------------------- \*/*

**int** main**(void)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

print\_statements**();**

**double** complex a2**,**a1**,**a0**,**r1**,**r2**,**r3**;**

**double** r2real**,**r2imag**,**r1real**,**r1imag**,**r0real**,**r0imag**;**

**int** cubic\_case**;**

printf**(**"Enter coefficients of Equation x^3+a2\*x^2+a1\*x+a0=0\n"**);**

printf**(**"in the order a2,a1,a0, seperated by spaces: "**);**

scanf**(**"%lf %lf %lf %lf %lf %lf"**,&**r2real**,&**r2imag**,&**r1real**,&**r1imag**,&**r0real**,&**r0imag**);**

a2 **=** r2real **+** I **\*** r2imag**;**

a1 **=** r1real **+** I **\*** r1imag**;**

a0 **=** r0real **+** I **\*** r0imag**;**

*//printf("a2: %f + %fi\n",creal(a2),cimag(a2));*

*//printf("a1: %f + %fi\n",creal(a1),cimag(a1));*

*//printf("a0: %f + %fi\n",creal(a0),cimag(a0));*

cubic\_case **=** rcubic\_roots**(**a2**,**a1**,**a0**,&**r1**,&**r2**,&**r3**);**

**switch** **(**cubic\_case**)** **{**

**case** **0:** printf**(**"There is one real root (r1) and two complex roots (r2,r3).\n r1 = %.10g + %.10gi, r2 = %.10g + %.10gi, r3 = %.10g - %.10gi \n"**,** creal**(**r1**),**cimag**(**r1**),**creal**(**r2**),**cimag**(**r2**),**creal**(**r3**),**cimag**(**r3**));** **break;**

**case** **1:** printf**(**"There are triple repeated real roots.\n r1 = r2 = r3 = %.10g + %.10gi \n"**,** creal**(**r1**),** cimag**(**r2**));** **break;**

**case** **2:** printf**(**"There is a pair of repeated real roots.\n r1 = %.10g + %.10gi, r2 = %.10g + %.10gi, r3 = %.10g + %.10gi \n"**,** creal**(**r1**),**cimag**(**r1**),**creal**(**r2**),**cimag**(**r2**),**creal**(**r3**),**cimag**(**r3**));** **break;**

**case** **3:** printf**(**"There are three distinct real roots.\n r1 = %.10g + %.10gi, r2 = %.10g + %.10gi, r3 = %.10g + %.10gi \n"**,** creal**(**r1**),**cimag**(**r1**),**creal**(**r2**),**cimag**(**r2**),**creal**(**r3**),**cimag**(**r3**));** **break;**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*

### prog\_m.c

*#include <stdio.h>*

*#include <math.h>*

*#include <complex.h>*

*#ifndef M\_PI /\*incase Pi is not defined on a compiler \*/*

*# define M\_PI 3.14159265358979323846*

*#endif*

*/\* -Functions-needed-from-other-files----------------------------------------- \*/*

**int** rcubic\_roots**(double** complex**,** **double** complex**,** **double** complex**,** **double** complex **\*,** **double** complex **\*,** **double** complex **\*);**

**void** print\_statements**();**

*/\* --------------------------------------------------------------------------- \*/*

**int** main**(void)** **{**

*/\* Bhageria, Yadu, M3SC \*/*

print\_statements**();**

**double** complex a2**,**a1**,**a0**,**r1**,**r2**,**r3**,**sj**;**

**int** j**;**

printf**(**" j, Re(z1), Im(z1), Re(z2), Im(z2), Re(z3), Im(z3)\n"**);**

**for** **(**j**=0;**j**<101;**j**++)** **{**

sj **=** cos**(**j**\***M\_PI**/50.0)** **+** I **\*** sin**(**j**\***M\_PI**/50.0);**

a0 **=** **(18.0\***sj**\***sj **-** **18\***sj**)** **/** **(3.0\***sj **-1.0);**

a1 **=** **(30.0\***sj **+** **6.0)** **/** **(3.0\***sj **-** **1.0);**

a2 **=** **-(8.0\***sj **+** **4.0)** **/** **(3.0\***sj **-** **1.0);**

*/\**

*printf("a2 = %.1f%+.1fi\n", creal(a2), cimag(a2));*

*printf("a1 = %.1f%+.1fi\n", creal(a1), cimag(a1));*

*printf("a0 = %.1f%+.1fi\n", creal(a0), cimag(a0));*

*\*/*

rcubic\_roots**(**a2**,**a1**,**a0**,&**r1**,&**r2**,&**r3**);**

printf**(**"%3d,%10.6f,%10.6f,%10.6f,%10.6f,%10.6f,%10.6f\n"**,** j**,**creal**(**r1**),**cimag**(**r1**),**creal**(**r2**),**cimag**(**r2**),**creal**(**r3**),**cimag**(**r3**));**

**}**

**}**

*/\* --------------------------------------------------------------------------- \*/*