Table of Contents

[INTRODUCTION 3](#_Toc445247015)

[Question 1 3](#_Toc445247016)

[Test Cases 3](#_Toc445247017)

[Linear Equations 3](#_Toc445247018)

[Quadratic Equations 4](#_Toc445247019)

[Cubic Equations 5](#_Toc445247020)

[Command Line used to Compile Testing Program 6](#_Toc445247021)

[Question 2: 7](#_Toc445247022)

[Choosing Sign in the Quadratic Polynomials 7](#_Toc445247023)

[Optimization Cases 8](#_Toc445247024)

[(i) a0 = 0 8](#_Toc445247025)

[(ii) a3 = a1 = 0 8](#_Toc445247026)

[(iii) a3 = a2 = a1 = 0 9](#_Toc445247027)

[Other Test Cases 9](#_Toc445247028)

[Case I – given 9](#_Toc445247029)

[Case II 10](#_Toc445247030)

[Case III 10](#_Toc445247031)

[Command Line used to Compile Testing Program 10](#_Toc445247032)

[Question 3: 11](#_Toc445247033)

[Derivation 11](#_Toc445247034)

[X = 3/8 and Y = 1/2 12](#_Toc445247035)

[Maximum Value of b 12](#_Toc445247036)

[Satellite orbiting above the Earth 15](#_Toc445247037)

[Cassini Probe 16](#_Toc445247038)

[Personal Fictitious Satellite 16](#_Toc445247039)

[Command Line Used to Compile Testing Program 17](#_Toc445247040)

[Mastery Section 18](#_Toc445247041)

[Appendix 20](#_Toc445247042)

[Tables 20](#_Toc445247043)

[Table from Mastery Section 20](#_Toc445247044)

[Representative C code and files 21](#_Toc445247045)

[Question 1 21](#_Toc445247046)

[lin\_root.c – THIS IS THE VERSION I WANT TESTED 21](#_Toc445247047)

[quad\_roots.c – THIS IS THE VERSION I WANT TESTED 22](#_Toc445247048)

[rcubic\_roots.c – THIS IS THE VERSION I WANT TESTED 23](#_Toc445247049)

[prog\_1.c 27](#_Toc445247050)

[Question 2 28](#_Toc445247051)

[rquartic\_roots.c – THIS IS THE VERSION I WANT TESTED 28](#_Toc445247052)

[prog\_2.c 31](#_Toc445247053)

[Question 3 32](#_Toc445247054)

[prog\_3.c 32](#_Toc445247055)

[prog\_3b.c 33](#_Toc445247056)

[prog\_3c.c 33](#_Toc445247057)

[Mastery Section 35](#_Toc445247058)

[quad\_roots\_complex.c 35](#_Toc445247059)

[rcubic\_complex.c 36](#_Toc445247060)

[rquartic\_complex.c 38](#_Toc445247061)

[prog\_m.c 40](#_Toc445247062)

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

INTRODUCTION

This project deals with creating a numerical quartic equation solver in the programming language C. Part 1 - 3 involve quartic equations that have real coefficients.

In Question 1 I go back to Project 1b and rewrite my lin\_root(), quad\_roots(), and rcubic\_roots() functions so that they now take arrays of coefficients and roots for input and output.

In Question 2 I implement a quartic equation solver by reducing the equations to a cubic equation to find a real value (the largest real root outputted by rcubic\_roots() ) and then using that to factor my quartic equation into two quadratic equations that I can solve easily using the quad\_roots() function developed in project 1a. Parts of the question also deal with optimising the code for specific cases where calling the rcubic\_roots function can be bypassed.

In Question 3 I apply the rquartic\_roots() function to find roots for a polynomials that describe normal lines to a ellipse. I then use this in various specific cases to find the height of satellites orbiting above the surface of a planet.

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

# Question 1

The program from project 1b is modified by changing the input from multiple doubles and pointers to 2 arrays. Now my functions are declared as follows:

**int** lin\_root**(double** **\***a**,** **double** **\***root**)**

**int** quad\_roots**(double** **\***a**,** **double** **\*** root**)**

**int** rcubic\_roots**(double** **\***a**,** **double** **\***root**)**

## Test Cases

Here I have created a program where I can choose the degree of polynomial I want to solve and then input the appropriate coefficients and get an output.

### Linear Equations

Case 1: a contradictory linear equation

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 22:23:47

Date: Mar 8 2016

Choose the degree of polynomial to be tested (1,2,3): 1

Enter coefficients of Equation a[1]\*x+a[0]=0

in the order a[1], a[0], separated by spaces: 0 1

Lin\_case: -1

Root: 6.95322e-310

Case 2: a tautological linear equation:

Choose the degree of polynomial to be tested (1,2,3): 1

Enter coefficients of Equation a[1]\*x+a[0]=0

in the order a[1], a[0], separated by spaces: 0 0

Lin\_case: 0

Root: 6.95323e-310

Case 3: a standard linear equation:

Choose the degree of polynomial to be tested (1,2,3): 1

Enter coefficients of Equation a[1]\*x+a[0]=0

in the order a[1], a[0], separated by spaces: 1e10 -1

Lin\_case: 1

Root: 1e-10

### Quadratic Equations

Case 4: both roots are complex

Choose the degree of polynomial to be tested (1,2,3): 2

Enter coefficients of Equation a[2]\*x^2+a[1]\*x+a[0]=0

in the order a[2], a[1], a[0], separated by spaces: 1 1 1

Quad\_case: 0

Roots: -0.5 0.866025

Case 5: repeated roots

Choose the degree of polynomial to be tested (1,2,3): 2

Enter coefficients of Equation a[2]\*x^2+a[1]\*x+a[0]=0

in the order a[2], a[1], a[0], separated by spaces: 1 -2 1

Quad\_case: 1

Roots: 1 1

Case 6: 2 distinct real roots

Choose the degree of polynomial to be tested (1,2,3): 2

Enter coefficients of Equation a[2]\*x^2+a[1]\*x+a[0]=0

in the order a[2], a[1], a[0], separated by spaces: 1 0 -1e10

Quad\_case: 2

Roots: 100000 -100000

Case 7: again 2 distinct real roots

Choose the degree of polynomial to be tested (1,2,3): 2

Enter coefficients of Equation a[2]\*x^2+a[1]\*x+a[0]=0

in the order a[2], a[1], a[0], separated by spaces: 1 5 -14

Quad\_case: 2

Roots: 2 -7

### Cubic Equations

Case 8: 3 distinct real roots

Choose the degree of polynomial to be tested (1,2,3): 3

Enter coefficients of Equation x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 2 -5 -6

Cubic\_case: 3

Roots: 2 -1 -3

Case 9: pair of repeated roots

Choose the degree of polynomial to be tested (1,2,3): 3

Enter coefficients of Equation x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 5 3 -9

Cubic\_case: 2

Roots: 1 -3 -3

Case 10: triple repeated real roots

Choose the degree of polynomial to be tested (1,2,3): 3

Enter coefficients of Equation x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: -9 27 -27

Cubic\_case: 1

Roots: 3 3 3

Case 11: a complex conjugate pair of roots and a real rool

Choose the degree of polynomial to be tested (1,2,3): 3

Enter coefficients of Equation x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 1 -1 -15

Cubic\_case: 0

Roots: 2.29191 -1.64595 1.95847

## Command Line used to Compile Testing Program

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

gcc -o prog\_1 prog\_1.c lin\_root.c quad\_roots.c rcubic\_roots.c

You can find the appropriate code in the appendix along with prog\_1 that deals with inputting and outputting solutions to the cubic polynomials.

# Question 2:

In this part of the project I create a quartic equation solver. Rather than dealing with an entirely generic quartic equation I use the method suggested in the “A Guide to Finding the roots of a Quartic Polynomial”[[1]](#footnote-1) on Dr Moore’s website. I reduce my quartic polynomial to a cubic polynomial of the form

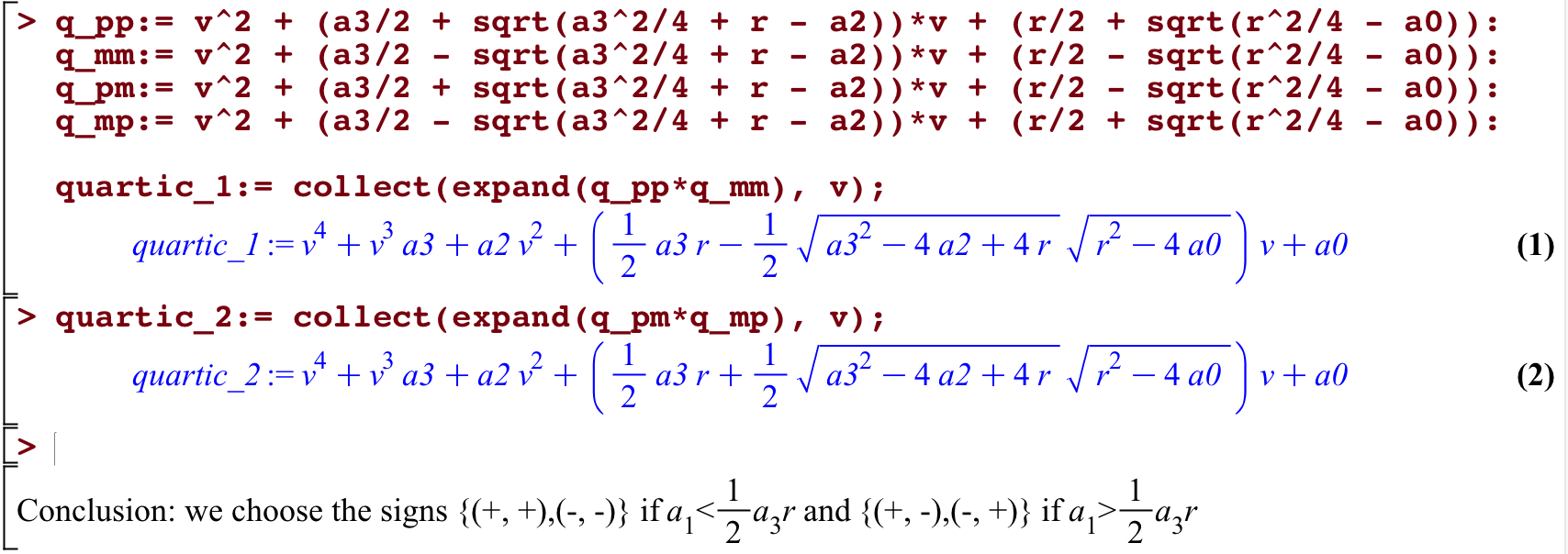
where

I can solve this using my rcubic\_roots() function from project\_1b and will get at least 1 real root which I call r – in the case of multiple real roots I use the largest one. I use this root to factor the quartic polynomial into two quadratic polynomials of the form:

which I can then easily solve to get the 4 roots using my quad\_roots() function. The next section shows how I decide how to choose a permutation of the plus and minus in the above equations for the two quadratic polynomials.

## Choosing Sign in the Quadratic Polynomials

In order to choose the whether I want the signs to be {(+,+),(-,-)} or {(+,-),(-,+)} I use some maple to see what criterion must be met:



It is immediately visible that changing the signs of the terms in the two different ways doesn't change the values of the coefficients when comparing with the original quartic. It does however depend on the value. I get that for I use {(+,+),(-,-)} and for I use {(+,-),(-,+)}.

## Optimization Cases

Here I demonstrate how I deal with the three optimization cases mentioned which do not require me calling a reduced cubic equation to resolve my quartic equation to two quadratic equations.

### (i) a0 = 0

Here it can immediately be sees that one root is 0 and so I can deal with a cubic equation as

So I find the three roots using the rcubic\_roots() function developed in project\_1b and then compare and order the roots with the previously known root of 0 to output the appropriate integer for the quartic function.

#### Test Case:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 22:23:47

Date: Mar 8 2016

Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: -9 27 -27 0

case: 1

roots: 3, 3, 3, 0

### (ii) a3 = a1 = 0

This case is where the quartic equation can actually be solved as a quadratic equation. This can easily been seen using the substitution of

The quartic equation currently is

which can be rewritten as

so I can solve this using the quad\_roots function from project\_1a and then square root the 2 roots found for the quadratic to find the 4 roots for the quartic. i.e.

Care must be taken when the quad\_roots() function computes complex roots. Also when it gives negative roots then the quartic ends up having complex conjugate roots.

Test Case:

Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 0 -2 0 1

case: 3

roots: 1, 1, -1, -1

### (iii) a3 = a2 = a1 = 0

This is the case of finding roots of unity for a 4th degree polynomial. The quartic is now

so if then so there are 2 real roots and 2 complex roots. If then all the roots are complex.

Test Case 1:

Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 0 0 0 -4

case: 2

roots: 1.414213562, -1.414213562, 0, 1.414213562

Test Case 2:

Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 0 0 0 4

case: 0

roots: 1, 1, -1, 1

## Other Test Cases

Here I test my program on the given test cases

### Case I – given

The reduced quartic equation is

My input and output looks like:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 21:54:13

Date: Mar 7 2016

Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 3 -39 -47 210

case: 4

roots: 5, 2, -3, -7

### Case II

My input and output looks like:

Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: 4 6 4 1

case: 1

roots: -1, -1, -1, -1

### Case III

My input and output looks like:

Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0

in the order a, separated by spaces: -2 2 -2 1

case: 2

roots: 1, 1, -0, 1

## 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 rquartic\_roots.c

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

# Question 3:

In this question I solve quartic equations that describe a normal line from an ellipse that goes through an arbitrary point.

## Derivation

An ellipse can be parametrically be described as and , with which can be rewritten as .

Now the tangent to the ellipse at a point on this ellipse is given by

For lines with perpendicular gradients we know that

So the gradient of the normal to the ellipse at a point is given by

The equation of a line can be written as

substituting and we get that

multiplying both sides by gives

and collecting the powers of t together finally gives

## X = 3/8 and Y = 1/2

Using the specified values of X and Y, I get the following table as output with b from 0.05 to 0.95:

Table 1: Values of t that satisfy the given quartic polynomial for varying b

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| b | case | t[1] | t[2] | t[3] | t[4] | phi[1] | phi[2] | phi[3] | phi[4] |
| 0.05 | 4 | 0.681 | -0.020 | -0.665 | -109.796 | 68.526 | -2.303 | -67.267 | -178.956 |
| 0.10 | 4 | 0.687 | -0.041 | -0.654 | -54.592 | 68.940 | -4.673 | -66.366 | -177.901 |
| 0.15 | 4 | 0.690 | -0.063 | -0.640 | -36.054 | 69.235 | -7.186 | -65.226 | -176.823 |
| 0.20 | 4 | 0.693 | -0.087 | -0.622 | -26.684 | 69.420 | -9.941 | -63.772 | -175.708 |
| 0.25 | 4 | 0.694 | -0.115 | -0.600 | -20.980 | 69.503 | -13.076 | -61.885 | -174.542 |
| 0.30 | 4 | 0.694 | -0.148 | -0.570 | -17.109 | 69.487 | -16.825 | -59.352 | -173.310 |
| 0.35 | 4 | 0.692 | -0.191 | -0.529 | -14.287 | 69.373 | -21.667 | -55.714 | -171.992 |
| 0.40 | 4 | 0.689 | -0.260 | -0.460 | -12.119 | 69.158 | -29.147 | -49.445 | -170.566 |
| 0.45 | 2 | 0.685 | -10.388 |  |  | 68.838 | -169.003 |  |  |
| 0.50 | 2 | 0.680 | -8.964 |  |  | 68.404 | -167.269 |  |  |
| 0.55 | 2 | 0.673 | -7.763 |  |  | 67.846 | -165.320 |  |  |
| 0.60 | 2 | 0.664 | -6.731 |  |  | 67.146 | -163.099 |  |  |
| 0.65 | 2 | 0.653 | -5.830 |  |  | 66.287 | -160.534 |  |  |
| 0.70 | 2 | 0.640 | -5.035 |  |  | 65.242 | -157.531 |  |  |
| 0.75 | 2 | 0.625 | -4.328 |  |  | 63.979 | -153.979 |  |  |
| 0.80 | 2 | 0.606 | -3.700 |  |  | 62.462 | -149.754 |  |  |
| 0.85 | 2 | 0.585 | -3.149 |  |  | 60.650 | -144.763 |  |  |
| 0.90 | 2 | 0.560 | -2.678 |  |  | 58.503 | -139.045 |  |  |
| 0.95 | 2 | 0.532 | -2.294 |  |  | 55.994 | -132.898 |  |  |

This table is generated using prog\_3.c which can be found in the appendix.

In total I get 54 values of t that satisfy the quartic equations. The matching values of for t[1], t[2], t[3], t[4] are also given respectively in degrees.

## Maximum Value of b

The maximum value of b can be found by using the method of the discriminant[[2]](#footnote-2).

The discriminant can be written as

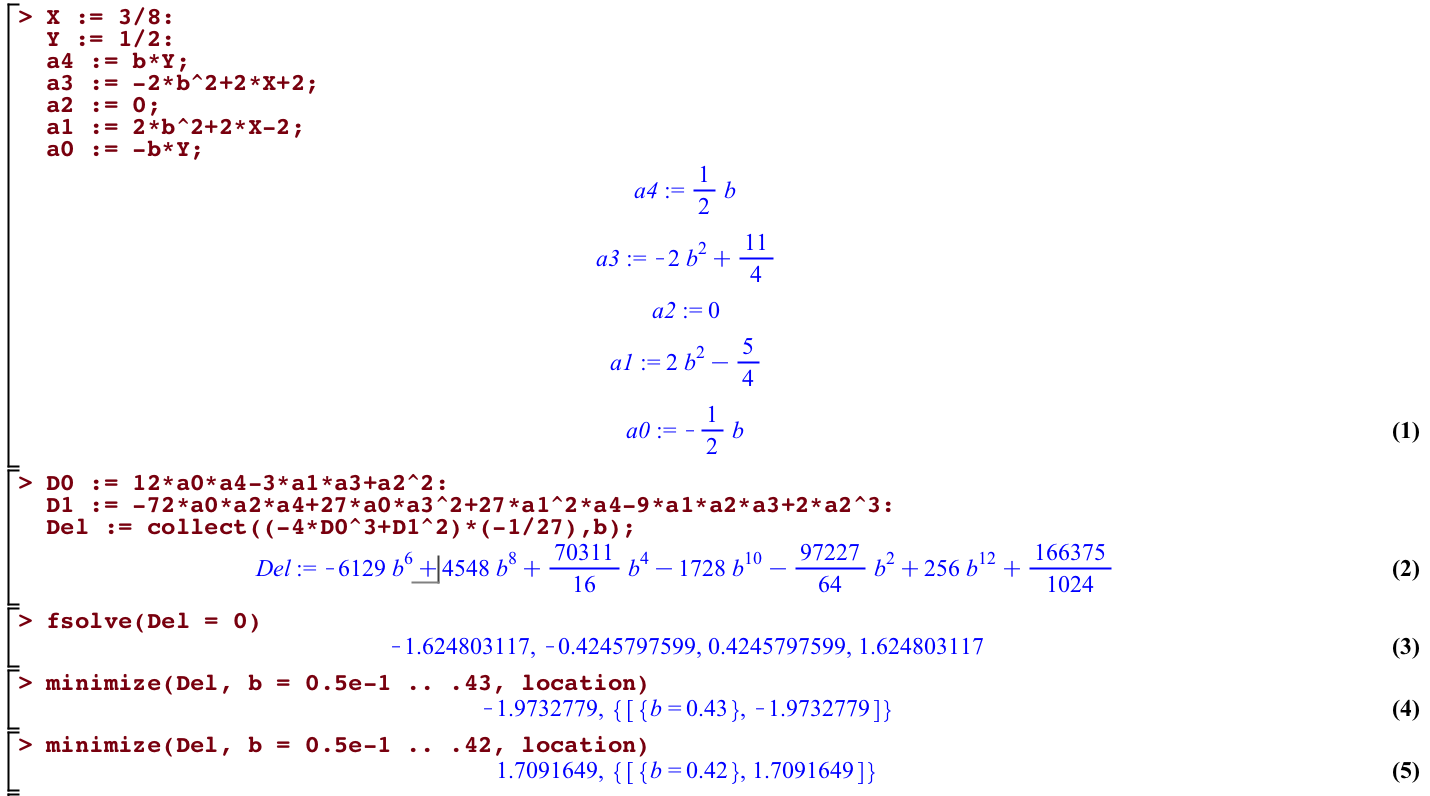
Where

and

Using the initial quartic equation, I can now final a numerical value of the discriminant which if negative tells us that there are 2 real and 2 complex roots and if positive tells us that there are either 4 real roots or 4 complex roots.

This is extremely useful as in our case the discriminant will always have at least 1 real root. This is due to the geometric interpretation of our problem of a finding a point of the ellipse such that the normal line from that point passes through (X,Y) which occurs at least once for every (X,Y) s.t. .

In the case of and the coefficients simplify down to

I now use this to find the discriminant in terms of b. The algebra here is quite tedious and so I use maple to simply this expression and find a numerical solution for b between 0 and 1.

As can be seen that for a value of b between 0.42 and 0.43 the discriminant changes from positive to negative. That is it goes from 4 real roots to 2 real and 2 complex roots.

Solving this numerically using fsolve I get that the maximum value of b in our range that achieves this is

For this value of I get values of t that satisfy the quartic equation using prog\_3b.c as

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

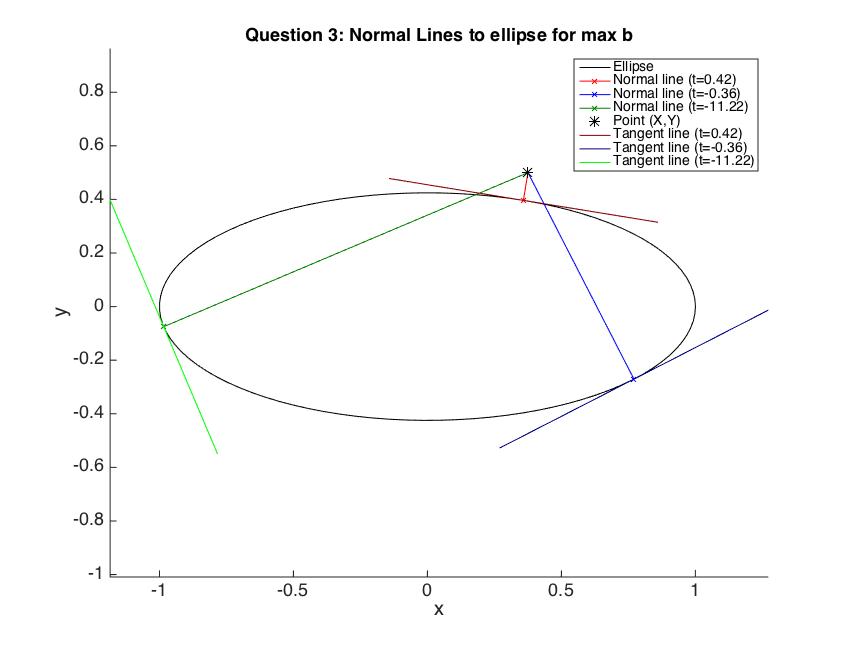
Time: 23:52:06

Date: Mar 7 2016

b, t[1], t[2], t[3], t[4], phi[1], phi[2], phi[3], phi[4],

0.42, 0.6875, -0.3600, -0.3600, -11.2231, 69.0145, -39.5986, -39.5994, -169.8166

Plotting this using MATLAB I get that



## Satellite orbiting above the Earth

I solve this problem by normalizing the radius of the earth to 1 at the equator and then I get a value of b for y coordinates as

Then again dividing the coordinates of the satellite by I get that

Solving this using the quadratic equation in prog\_3c.c for normal lines to the ellipse I get

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 11:19:10

Date: Mar 8 2016

Quartic Case: 2

b, t[1], phi[1], t[2], phi[2],

0.996647, 0.413230, 44.903786, -2.440993, -135.445178,

The satellite is located in the positive quadrant and so I am interest in between 0 and 90 degrees. That is the value for . Using this I find that the point directly below the satellite is

Coordinate directly below the Satellite (rescaled to actual size)

X = 4517590.99 m, Y = 4487348.076 m

Angle to Equator = 45

Distance = 654321 m

This distance between these two points can be simply found the formula

and the geodetic latitude angle can be found using the formula (for radians, converted to angles by multiplying by )

This is because we are considering the gradient of the normal to the curve at the point just below the satellite as this make the required angle with the equator.

## Cassini Probe

Using exactly the same method as for the satellite above the earth in prog\_3c.c and I use the values

R = 60330000

b = 53542792.751/R

X = 47887108.74334894/R

Y = 49584769.77356416/R

And I find for the Cassini probe the following information:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 11:56:57

Date: Mar 8 2016

Quartic Case: 2

b, t[1], phi[1], t[2], phi[2],

0.887499, 0.447308, 0.841226, -2.976458, -2.493350,

Coordinate directly below the Satellite (rescaled to actual size)

X = 40212911.67 m, Y = 39914067.27 m

Angle to Equator = 51.56620156

Distance = 12345678.9 m

Using the same formulas as for the satellite above the surface of the earth I get distance and geodetic latitude are

## Personal Fictitious Satellite

My satellite is located at the coordinates

I again normalize this using the radius of the earth at the equator and so my values are

R = 6378137

b = 6356752/R

X = 4317364/R

Y = 5461337/R

I then compute the point directly below it on the earth using prog\_3c.c again as

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:16:00

Date: Mar 8 2016

Quartic Case: 2

b, t[1], phi[1], t[2], phi[2],

0.996647, 0.485035, 0.903209, -2.077515, -2.244369,

Coordinate directly below the Satellite (rescaled to actual size)

X = 3948662.694 m, Y = 4992067.588 m

Angle to Equator = 51.8435566

Distance = 596786.7578 m

I get distance and geodetic latitude angle values of

## Command Line Used to Compile Testing Program

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

gcc -o prog\_3 prog\_3.c lin\_root.c quad\_roots.c rcubic\_roots.c rquartic\_roots.c

and

gcc -o prog\_3b prog\_3b.c lin\_root.c quad\_roots.c rcubic\_roots.c rquartic\_roots.c

gcc -o prog\_3c prog\_3c.c lin\_root.c quad\_roots.c rcubic\_roots.c rquartic\_roots.c

The rquartic\_roots.c is the same as for Question 2. Find prog\_3.c, prog\_3b.c, prog\_3c.c in the appendix.

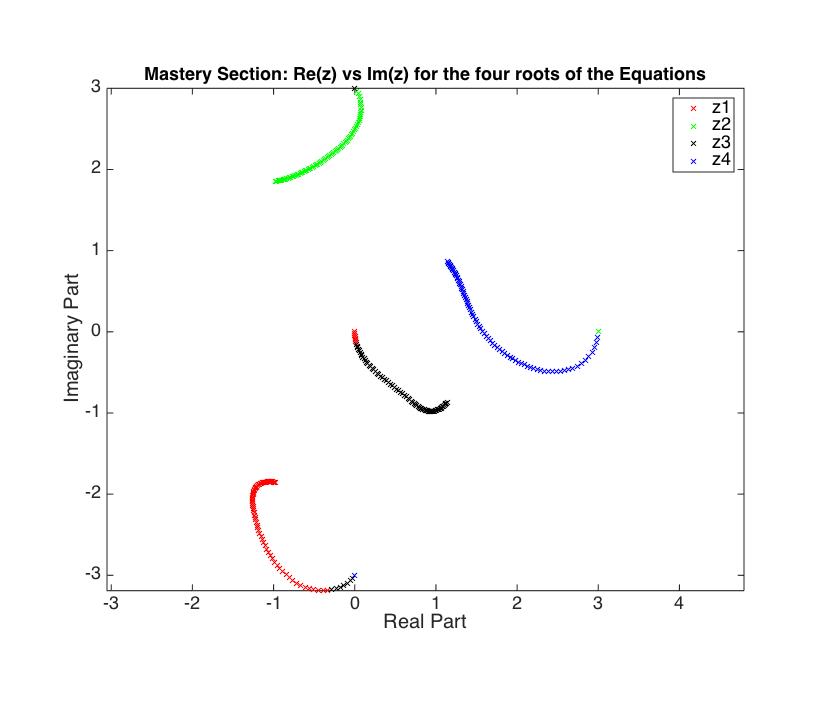
# Mastery Section

In this part of the project I deal with a reduced quartic equation as well but with complex coefficients.

I again solve this in exactly the same way as for the real case but instead with complex coefficients.

The coefficients are

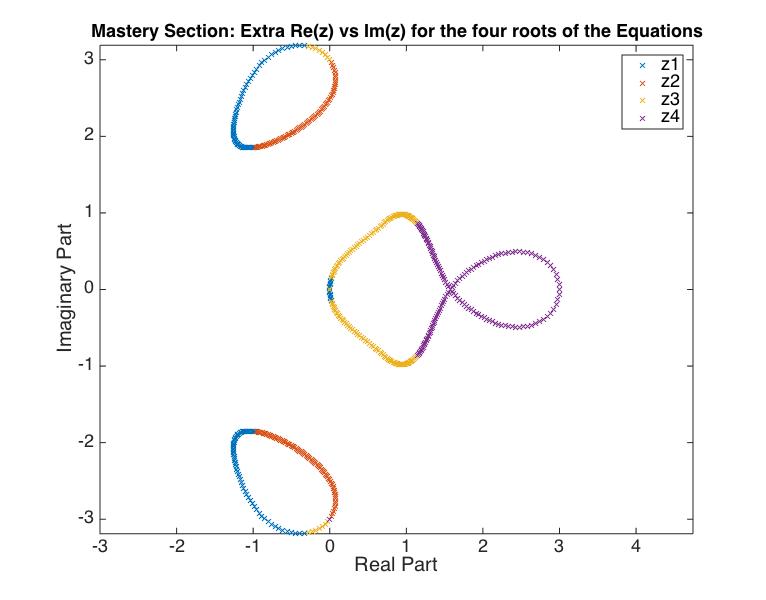
where

I get the following plot by inputting my outputted values with Matlab:

The table for the outputted roots is placed in the appendix as its quite large.

Looking at the plot it can be seen the it looks incomplete. This is because we are only considering which means that our value of does not go through a complete cycle of its potential values for integer .

Plotting instead for I get the full plot that is:



Appendix

# Tables

## Table from Mastery Section

This is the table of the 324 roots outputted by my complex quartic roots function for the given cases.

Table 2: Values of z that are roots for the given equations with varying j

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| j | Re(z1) | Im(z1) | Re(z2) | Im(z2) | Re(z3) | Im(z3) | Re(z4) | Im(z4) |
| 0 | 0.000 | 0.000 | 3.000 | 0.000 | 0.000 | 3.000 | 0.000 | -3.000 |
| 1 | 0.000 | -0.020 | 0.022 | 2.965 | -0.027 | -3.034 | 2.995 | -0.068 |
| 2 | 0.002 | -0.039 | 0.040 | 2.930 | -0.058 | -3.066 | 2.980 | -0.135 |
| 3 | 0.003 | -0.059 | 0.054 | 2.895 | -0.095 | -3.096 | 2.956 | -0.198 |
| 4 | 0.006 | -0.079 | 0.065 | 2.861 | -0.135 | -3.123 | 2.923 | -0.256 |
| 5 | 0.010 | -0.098 | 0.072 | 2.827 | -0.180 | -3.146 | 2.884 | -0.308 |
| 6 | 0.014 | -0.118 | 0.077 | 2.795 | -0.229 | -3.165 | 2.840 | -0.353 |
| 7 | 0.019 | -0.137 | 0.079 | 2.763 | -0.281 | -3.179 | 2.791 | -0.391 |
| 8 | -0.335 | -3.187 | 0.078 | 2.733 | 0.025 | -0.157 | 2.741 | -0.422 |
| 9 | -0.391 | -3.189 | 0.076 | 2.703 | 0.032 | -0.177 | 2.688 | -0.447 |
| 10 | -0.447 | -3.186 | 0.072 | 2.675 | 0.039 | -0.196 | 2.635 | -0.466 |
| 11 | -0.503 | -3.178 | 0.067 | 2.648 | 0.047 | -0.216 | 2.583 | -0.479 |
| 12 | -0.558 | -3.164 | 0.060 | 2.622 | 0.056 | -0.236 | 2.531 | -0.487 |
| 13 | -0.612 | -3.145 | 0.052 | 2.597 | 0.066 | -0.255 | 2.480 | -0.491 |
| 14 | -0.664 | -3.122 | 0.044 | 2.573 | 0.077 | -0.275 | 2.431 | -0.492 |
| 15 | -0.714 | -3.095 | 0.034 | 2.550 | 0.089 | -0.294 | 2.383 | -0.489 |
| 16 | -0.761 | -3.064 | 0.024 | 2.528 | 0.101 | -0.314 | 2.337 | -0.483 |
| 17 | -0.805 | -3.031 | 0.013 | 2.507 | 0.115 | -0.333 | 2.292 | -0.475 |
| 18 | -0.847 | -2.995 | 0.001 | 2.486 | 0.129 | -0.353 | 2.248 | -0.465 |
| 19 | -0.886 | -2.958 | -0.011 | 2.467 | 0.144 | -0.372 | 2.207 | -0.453 |
| 20 | -0.922 | -2.919 | -0.023 | 2.447 | 0.160 | -0.392 | 2.166 | -0.439 |
| 21 | -0.955 | -2.879 | -0.036 | 2.429 | 0.177 | -0.411 | 2.127 | -0.424 |
| 22 | -0.985 | -2.839 | -0.050 | 2.411 | 0.195 | -0.431 | 2.089 | -0.408 |
| 23 | -1.014 | -2.798 | -0.063 | 2.394 | 0.214 | -0.450 | 2.052 | -0.390 |
| 24 | -1.040 | -2.757 | -0.077 | 2.377 | 0.234 | -0.470 | 2.015 | -0.372 |
| 25 | -1.063 | -2.716 | -0.091 | 2.360 | 0.254 | -0.489 | 1.980 | -0.353 |
| 26 | -1.085 | -2.676 | -0.106 | 2.344 | 0.275 | -0.508 | 1.946 | -0.333 |
| 27 | -1.105 | -2.636 | -0.120 | 2.329 | 0.298 | -0.528 | 1.913 | -0.312 |
| 28 | -1.123 | -2.596 | -0.135 | 2.314 | 0.320 | -0.547 | 1.880 | -0.290 |
| 29 | -1.140 | -2.558 | -0.150 | 2.299 | 0.344 | -0.567 | 1.848 | -0.267 |
| 30 | -1.155 | -2.520 | -0.165 | 2.284 | 0.368 | -0.586 | 1.817 | -0.243 |
| 31 | -1.169 | -2.484 | -0.180 | 2.270 | 0.393 | -0.606 | 1.786 | -0.219 |
| 32 | -1.181 | -2.448 | -0.196 | 2.256 | 0.418 | -0.625 | 1.756 | -0.194 |
| 33 | -1.192 | -2.413 | -0.211 | 2.243 | 0.443 | -0.645 | 1.727 | -0.168 |
| 34 | -1.203 | -2.380 | -0.227 | 2.230 | 0.468 | -0.665 | 1.699 | -0.141 |
| 35 | -1.212 | -2.348 | -0.243 | 2.217 | 0.494 | -0.685 | 1.672 | -0.113 |
| 36 | -1.220 | -2.317 | -0.259 | 2.204 | 0.519 | -0.705 | 1.646 | -0.085 |
| 37 | -1.227 | -2.287 | -0.275 | 2.191 | 0.543 | -0.725 | 1.621 | -0.055 |
| 38 | -1.234 | -2.259 | -0.291 | 2.179 | 0.567 | -0.745 | 1.597 | -0.026 |
| 39 | -1.239 | -2.232 | -0.307 | 2.167 | 0.591 | -0.765 | 1.575 | 0.004 |
| 40 | -1.244 | -2.205 | -0.323 | 2.155 | 0.614 | -0.784 | 1.554 | 0.035 |
| 41 | -1.248 | -2.181 | -0.340 | 2.143 | 0.635 | -0.803 | 1.534 | 0.065 |
| 42 | -1.251 | -2.157 | -0.356 | 2.132 | 0.656 | -0.821 | 1.516 | 0.096 |
| 43 | -1.254 | -2.134 | -0.373 | 2.120 | 0.676 | -0.838 | 1.498 | 0.126 |
| 44 | -1.256 | -2.113 | -0.389 | 2.109 | 0.696 | -0.854 | 1.482 | 0.156 |
| 45 | -1.257 | -2.093 | -0.406 | 2.098 | 0.714 | -0.870 | 1.467 | 0.185 |
| 46 | -1.257 | -2.073 | -0.422 | 2.087 | 0.732 | -0.884 | 1.453 | 0.215 |
| 47 | -1.257 | -2.055 | -0.439 | 2.077 | 0.749 | -0.897 | 1.440 | 0.243 |
| 48 | -1.256 | -2.038 | -0.456 | 2.067 | 0.765 | -0.910 | 1.427 | 0.271 |
| 49 | -1.254 | -2.022 | -0.472 | 2.056 | 0.781 | -0.921 | 1.415 | 0.298 |
| 50 | -1.252 | -2.006 | -0.489 | 2.046 | 0.797 | -0.931 | 1.404 | 0.325 |
| 51 | -1.249 | -1.992 | -0.506 | 2.037 | 0.812 | -0.940 | 1.393 | 0.351 |
| 52 | -1.246 | -1.978 | -0.523 | 2.027 | 0.826 | -0.948 | 1.383 | 0.376 |
| 53 | -1.242 | -1.965 | -0.540 | 2.018 | 0.840 | -0.955 | 1.373 | 0.401 |
| 54 | -1.238 | -1.954 | -0.556 | 2.008 | 0.854 | -0.961 | 1.363 | 0.425 |
| 55 | -1.233 | -1.942 | -0.573 | 1.999 | 0.868 | -0.966 | 1.354 | 0.448 |
| 56 | -1.227 | -1.932 | -0.590 | 1.991 | 0.881 | -0.970 | 1.345 | 0.471 |
| 57 | -1.221 | -1.922 | -0.607 | 1.982 | 0.894 | -0.974 | 1.336 | 0.494 |
| 58 | -1.215 | -1.913 | -0.624 | 1.974 | 0.906 | -0.976 | 1.327 | 0.516 |
| 59 | -1.208 | -1.905 | -0.640 | 1.965 | 0.919 | -0.978 | 1.318 | 0.537 |
| 60 | -1.200 | -1.897 | -0.657 | 1.957 | 0.931 | -0.979 | 1.310 | 0.557 |
| 61 | -1.193 | -1.890 | -0.674 | 1.950 | 0.943 | -0.979 | 1.301 | 0.578 |
| 62 | -1.184 | -1.884 | -0.691 | 1.942 | 0.955 | -0.979 | 1.293 | 0.597 |
| 63 | -1.175 | -1.878 | -0.707 | 1.935 | 0.966 | -0.977 | 1.285 | 0.616 |
| 64 | -1.166 | -1.873 | -0.724 | 1.928 | 0.978 | -0.976 | 1.276 | 0.635 |
| 65 | -1.157 | -1.868 | -0.740 | 1.921 | 0.989 | -0.973 | 1.268 | 0.653 |
| 66 | -1.147 | -1.864 | -0.757 | 1.915 | 1.000 | -0.970 | 1.260 | 0.671 |
| 67 | -1.136 | -1.860 | -0.773 | 1.909 | 1.011 | -0.966 | 1.252 | 0.688 |
| 68 | -1.126 | -1.857 | -0.789 | 1.903 | 1.022 | -0.962 | 1.243 | 0.705 |
| 69 | -1.115 | -1.855 | -0.805 | 1.897 | 1.032 | -0.957 | 1.235 | 0.721 |
| 70 | -1.103 | -1.852 | -0.821 | 1.892 | 1.043 | -0.952 | 1.227 | 0.736 |
| 71 | -1.092 | -1.851 | -0.837 | 1.886 | 1.053 | -0.946 | 1.218 | 0.752 |
| 72 | -1.080 | -1.849 | -0.853 | 1.882 | 1.063 | -0.939 | 1.210 | 0.766 |
| 73 | -1.067 | -1.849 | -0.869 | 1.877 | 1.073 | -0.932 | 1.201 | 0.781 |
| 74 | -1.055 | -1.848 | -0.884 | 1.873 | 1.083 | -0.924 | 1.193 | 0.795 |
| 75 | -1.042 | -1.848 | -0.899 | 1.869 | 1.093 | -0.916 | 1.184 | 0.808 |
| 76 | -1.028 | -1.849 | -0.914 | 1.865 | 1.103 | -0.907 | 1.176 | 0.821 |
| 77 | -1.015 | -1.849 | -0.929 | 1.862 | 1.112 | -0.898 | 1.167 | 0.833 |
| 78 | -1.001 | -1.851 | -0.944 | 1.859 | 1.121 | -0.889 | 1.158 | 0.845 |
| 79 | -0.987 | -1.852 | -0.959 | 1.856 | 1.131 | -0.879 | 1.149 | 0.857 |
| 80 | -0.973 | -1.854 | -0.973 | 1.854 | 1.140 | -0.868 | 1.140 | 0.868 |

# 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** **\***a**,** **double** **\***root**)** **{**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**if** **(**a**[1]==0){**

**if** **(**a**[0]==0){**

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

**}** **else{**

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

**}**

**}**

root**[1]** **=** a**[0]** **==** **0** **?** **0** **:** **-**a**[0]/**a**[1];** *// real root*

**return(1);**

**}**

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

*#include <stdio.h>*

*#include <math.h>*

*#include <float.h>*

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

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

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

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

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

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

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

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

**int** quad\_roots**(double** **\***a**,** **double** **\*** root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

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

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

**if** **(**a**[2]==0)** **{**

**return** **(-2** **+** lin\_root**(**a**,**root**));**

**}** **else** **if** **(**a**[0]==0)** **{**

lin\_root**(**a**,**root**);**

root**[2]** **=** zero**;**

order**(**root**,2);**

**return** root**[1]** **==** root**[2]** **?** **1** **:** **2;**

**}**

a1\_half **=** a**[1]/**two**;**

**if** **(**fabs**(**a1\_half**)>=**fabs**(**a**[0])){**

e**=**one**-((**a**[0]/**a1\_half**)\*(**a**[2]/**a1\_half**));**

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

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

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

e**=(**a1\_half**\*(**a1\_half**/**a**[0]))-**a**[2];**

coef **=** sqrt**(**fabs**(**a**[0]));**

d\_sgn **=** sgn**(**a**[0])\***sgn**(**e**);**

**}**

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

root**[1]=(-**a1\_half**)/(**a**[2]);** *//Real Part*

root**[2]=**coef**\***sqrt**(**fabs**(**e**))/**a**[2];** *//Imaginary Part*

**return(0);**

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

root**[1]** **=** root**[2]** **=** **(-**a1\_half**)/(**a**[2]);**

**return(1);**

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

add\_sgn **=** a**[1]** **==** zero **?** **1** **:** sgn**(**a**[1]);**

sub\_sgn **=** **-**add\_sgn**;**

root**[1]** **=** **(** **-**a1\_half**-**add\_sgn**\***coef**\***sqrt**(**fabs**(**e**))** **)** **/** a**[2];**

root**[2]** **=** **(** fabs**(**a**[0])<10\***DBL\_EPSILON **&&** fabs**(**a**[2])<10\***DBL\_EPSILON **)** **?** **(**a**[0]/**a**[2])/**root**[1]** **:** **(**a**[0]** **/** root**[1])/**a**[2];**

order**(**root**,2);**

**return(2);**

**}**

**}**

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

**void** order**(double** **\*** nums**,** **int** n**)** **{**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**int** i**,**j**;**

**for** **(**i**=1;**i**<**n**+1;**i**++){**

**for** **(**j**=**n**;**j**>**i**;**j**--){**

**if** **(**nums**[**i**]<**nums**[**j**])** **{**

**double** dummy **=** nums**[**j**];**

nums**[**j**]** **=** nums**[**i**];**

nums**[**i**]** **=** dummy**;**

**}**

**}**

**}**

**}**

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

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

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

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

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

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

**return(0);**

**}**

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

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

*/\* Bhageria, Yadu, 00733164, 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.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**(double** **\*,** **int);**

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

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

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

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

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

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

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

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

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

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

**int** rcubic\_roots**(double** **\***a**,** **double** **\***root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

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

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

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

*/\* Cases where reducing the polynomial is uncessary \*/*

**if** **(**a**[2]** **==** zero **&&** a**[1]** **==** zero **&&** a**[0]** **==** zero**)** **{** */\* a[2]=a[1]=a[0]=0. So x^3 = 0 => root[1]=root[2]=root[3]=0 \*/*

root**[3]=**root**[2]=**root**[1]=**zero**;**

**return(1);**

**}** **else** **if** **(**a**[2]** **==** zero **&&** a**[1]** **==** zero**)** **{** */\* Part 3 i \*/*

**return(**roots\_of\_unity**(**cbrt**(-**a**[0]),**root**));**

**}** **else** **if** **(** a**[0]==**zero **&&** a**[1]==**zero **){** */\* Linear equation with 2 roots equal to zero \*/*

root**[1]** **=** root**[2]** **=** zero**;**

root**[3]** **=** **-**a**[2];**

order**(**root**,3);**

**return** **2;**

**}** **else** **if** **(**a**[0]** **==** zero**)** **{** */\* Part 3 ii \*/*

root**[1]** **=** zero**;**

**return(**roots\_return**(**one**,**a**[2],**a**[1],**root**));**

**}** **else** **if** **(**fabs**(**a**[0]-**a**[1]\***a**[2])<**DBL\_EPSILON**){** */\* Part 3 iii \*/*

root**[1]** **=** **-**a**[2];**

**if** **(**a**[1]>**zero**){**

root**[2]** **=** zero**;**

root**[3]** **=** sqrt**(**a**[1]);**

**return(0);**

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

root**[2]** **=** sqrt**(-**a**[1]);**

root**[3]** **=** **-**sqrt**(-**a**[1]);**

order**(**root**,3);**

**return** root**[1]==**root**[2]** **||** root**[2]==**root**[3]** **?** **2** **:** **3;**

**}**

**}** **else** **if** **(** fabs**(3.0\***a**[1]-**a**[2]\***a**[2])** **<** DBL\_EPSILON **&&** fabs**(27.0\***a**[0]-**a**[2]\***a**[2]\***a**[2])** **<** DBL\_EPSILON **){** */\* Part 3 iv \*/*

root**[3]** **=** root**[2]** **=** root**[1]** **=** **-**a**[2]/3.0;**

**return(1);**

**}**

*/\* End of Cases \*/*

beta**=-**a**[2]/**three**;**

**if** **(**fabs**(**beta**)>**one**)** **{**

alpha\_coef **=** cbrt**(**two **-** **(**a**[1]/**beta**)/**beta **-** **((**a**[0]/**beta**)/**beta**)/**beta **);**

alpha **=** beta **\*** alpha\_coef**;**

p **=** **((**three **/** alpha\_coef**)** **/** alpha\_coef**)** **-** **((**a**[1]** **/** alpha**)** **/** alpha**);**

*//p = (three-(a[1]/beta)/beta)/(alpha\_coef\*alpha\_coef);*

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

alpha **=** cbrt**(**two**\***beta**\***beta**\***beta **-** a**[1]\***beta **-** a**[0]);**

p **=** **((**three**\***beta**\***beta **-** a**[1])** **/** alpha**)** **/** alpha**;**

**}**

*#ifdef DEBUG*

printf**(**"DEBUG REPORT| In RCubic\_Roots"**);**

printf**(**"DEBUG REPORT| beta = %.20g\n"**,** beta**);**

printf**(**"DEBUG REPORT| alpha = %.20g\n"**,** alpha**);**

printf**(**"DEBUG REPORT| p = %.20g\n"**,** p**);**

printf**(**"DEBUG REPORT| p2 = %.20g\n"**,** p2**);**

printf**(**"DEBUG REPORT| diff = %.20g\n"**,** fabs**(**p**-**p2**));**

printf**(**"DEBUG REPORT| allowed = %.20g\n"**,** **10\***DBL\_EPSILON**);**

printf**(**"DEBUG REPORT| Exitted RCubic\_Roots"**);**

*#endif*

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

root**[1]** **=** beta**;**

**return(**roots\_return**(**one**,-**two**\***beta**,-**two**\***beta**\***beta**+**a**[1],**root**));**

**}** **else** **if** **(** fabs**(**p **-** p2**)** **<** **10.0\***DBL\_EPSILON **&&** fabs**(1.0/**alpha**)** **>** **10e-10){** */\* Part 3 v: we choose 10.0\*DBL\_EPSILON as we loose accuracy in the calculation of alpha, beta, and p. \*/*

root**[1]** **=** alpha**\***y2 **+** beta**;**

root**[2]** **=** alpha**\***y2 **+** beta**;**

root**[3]** **=** alpha**/(**y2**\***y2**)** **+** beta**;**

order**(**root**,3);**

**return** root**[1]==**root**[2]** **||** root**[2]==**root**[3]** **?** **2** **:** **3;**

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

roots\_of\_unity**(**one**,**root**);**

root**[1]** **=** alpha **\*** **(**root**[1])** **+** beta**;**

root**[2]** **=** alpha **\*** **(**root**[2])** **+** beta**;**

root**[3]** **=** fabs**(**alpha **\*** root**[3]);**

**return(0);**

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

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

root**[1]=** alpha**\***y **+** beta**;**

**}**

root**[1]** **=** fabs**(**root**[1])<10\***DBL\_EPSILON **?** **0.0** **:** root**[1];**

**int** return\_val **=** roots\_return**(**one**,** a**[2]+(**root**[1]),** **(**root**[1]==**zero**)?** a**[1]** **:** **-**a**[0]/(**root**[1])** **,** root**);**

*/\* Makes sure that if 2 real roots are close and p was close to p2 when the*

*program thinks there are 3 reals roots then to use case of p=p2 \*/*

**if** **(**return\_val**==3** **&&** fabs**(**p**-**p2**)<1e-10** **&&** **(**fabs**(**root**[1]-**root**[2])<1e-5** **||** fabs**(**root**[2]-**root**[3])<1e-5** **)){**

root**[1]** **=** alpha**\***y2 **+** beta**;**

root**[2]** **=** alpha**\***y2 **+** beta**;**

root**[3]** **=** alpha**/(**y2**\***y2**)** **+** beta**;**

order**(**root**,3);**

**return** root**[1]==**root**[2]** **||** root**[2]==**root**[3]** **?** **2** **:** **3;**

**}**

**return(**return\_val**);**

**}**

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

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

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

*/\* Bhageria, Yadu, 00733164, 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, 00733164, 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, 00733164, 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** **(** count **<** **4** **||** **(**fabs**(**yn**-**yn1**)>**DBL\_EPSILON **&&** 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 root[2] and root[3] once root[1] has been found. Returns an integer corresponding to the number of real roots found \*/*

**int** roots\_return**(double** a2**,** **double** a1**,** **double** a0**,** **double** **\***root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**double** b**[3];**

b**[0]** **=** a0**;**

b**[1]** **=** a1**;**

b**[2]** **=** a2**;**

**int** quad\_case **=** quad\_roots**(**b**,**root**+1);**

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

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

**case** **1:** **return** root**[1]==**root**[2]** **?** **1** **:** **2;**

**case** **2:** order**(**root**,3);** **if** **(**root**[1]==**root**[2]){return(2);}** **else** **{return** root**[2]==**root**[3]** **?** **2** **:** **3;}**

*/\* In case 2 root[1] cannot equal root[3] as root[1],root[2],root[3] are ordered and hence we would have dealt with that in case 1 \*/*

**}**

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

**}**

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

**int** roots\_of\_unity**(double** r**,** **double** **\***root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

root**[1]** **=** r**;** *//Real Root*

root**[2]** **=** **-**r**/2.0;** *//Real Part of Complex Roots*

root**[3]** **=** fabs**(**r**)\***sqrt**(3.0)/2.0;** *//Positive Imaginary Part of Complex Roots*

**return(0);**

**}**

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

### prog\_1.c

*#include <stdio.h>*

*#include <math.h>*

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

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

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

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

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

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

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

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

print\_statements**();**

**double** a**[3],**root**[4];**

**int** lin\_case**,** quad\_case**,** cubic\_case**,** poly**;**

printf**(**"Choose the degree of polynomial to be tested (1,2,3): "**);**

scanf**(**"%d"**,** **&**poly**);**

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

**switch** **(**poly**){**

**case** **1:** **{**

printf**(**"Enter coefficients of Equation a[1]\*x+a[0]=0\n"**);**

printf**(**"in the order a[1], a[0], separated by spaces: "**);**

scanf**(**"%lf %lf"**,&**a**[1],&**a**[0]);**

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

lin\_case **=** lin\_root**(**a**,**root**);**

printf**(**"Lin\_case: %d\n"**,** lin\_case**);**

printf**(**" Root: %g\n"**,** root**[1]);**

**break;}**

**case** **2:** **{**

printf**(**"Enter coefficients of Equation a[2]\*x^2+a[1]\*x+a[0]=0\n"**);**

printf**(**"in the order a[2], a[1], a[0], separated by spaces: "**);**

scanf**(**"%lf %lf %lf"**,&**a**[2],&**a**[1],&**a**[0]);**

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

quad\_case **=** quad\_roots**(**a**,**root**);**

printf**(**"Quad\_case: %d\n"**,** quad\_case**);**

printf**(**" Roots: %g %g\n"**,** root**[1],** root**[2]);**

**break;}**

**case** **3:** **{**

printf**(**"Enter coefficients of Equation x^3+a[2]\*x^2+a[1]\*x+a[0]=0\n"**);**

printf**(**"in the order a, separated by spaces: "**);**

scanf**(**"%lf %lf %lf"**,&**a**[2],&**a**[1],&**a**[0]);**

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

cubic\_case **=** rcubic\_roots**(**a**,**root**);**

printf**(**"Cubic\_case: %d\n"**,** cubic\_case**);**

printf**(**" Roots: %g %g %g\n"**,** root**[1],** root**[2],** root**[3]);**

**break;}**

**}**

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

## Question 2

### rquartic\_roots.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**(double** **\*,** **int);**

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

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

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

**void** swap**(double** **\*,** **int);**

**int** squared\_quadratic**(double** **\*,** **double** **\*);**

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

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

**int** quartic\_roots\_of\_unity**(double,** **double** **\*);**

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

**int** rquartic\_roots**(double** **\***a**,** **double** **\***root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

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

**int** qc1**,**qc2**;**

*/\* Optimization Cases that Skip the R-Quartic Method \*/*

*/\* Part i \*/*

**if** **(**a**[0]==**zero**){**

root**[1]** **=** **0;**

qc1 **=** rcubic\_roots**(**a**+1,**root**+1);**

**return** return\_roots**(** root**[1]** **==** root**[2]** **?** **1** **:** **2** **,** qc1 **>** **1** **?** qc1 **-** **1** **:** qc1 **,** root**);**

**}** */\* Part iii \*/*

**else** **if** **(**a**[3]==**zero **&&** a**[2]==**zero **&&** a**[1]==**zero**)** **{**

**return** quartic\_roots\_of\_unity**(-**a**[0],** root**);**

**}** */\* Part ii \*/*

**else** **if** **(**a**[3]==**zero **&&** a**[1]==**zero**){**

**return** squared\_quadratic**(**a**,** root**);**

**}**

**double** coefs\_q1**[3],**coefs\_q2**[3];**

coefs\_q1**[2]** **=** coefs\_q2**[2]** **=** **1.0;**

**double** b**[]** **=** **{4.0\***a**[0]\***a**[2]** **-** a**[1]\***a**[1]** **-** a**[0]\***a**[3]\***a**[3]** **,** a**[1]\***a**[3]** **-** **4.0\***a**[0]** **,** **-**a**[2]};**

**int** cubic\_case **=** rcubic\_roots**(**b**,**root**);**

**double** r **=** **(** fabs**(**root**[1])<**DBL\_EPSILON **)** **?** **0** **:** root**[1];**

**if** **(**cubic\_case**>1** **&&** r **==** zero**){**

r **=** root**[2]==**zero **?** root**[3]** **:** root**[2];**

**}**

**double** v0\_coef **=** sqrt**(** **(**r**/2.0)\*(**r**/2.0)** **-** a**[0]** **);**

**double** v1\_coef **=** sqrt**(** a**[3]\***a**[3]/4.0** **+** r **-** a**[2]** **);**

*#ifdef DEBUG*

printf**(**"DEBUG REPORT| Not an Optimization Case\n"**);**

printf**(**"DEBUG REPORT| b = %g, %g, %g\n"**,** b**[2],**b**[1],**b**[0]);**

printf**(**"DEBUG REPORT| cubic\_case = %d\n"**,** cubic\_case**);**

printf**(**"DEBUG REPORT| roots = %g %g %g\n"**,** root**[1],** root**[2],** root**[3]);**

printf**(**"DEBUG REPORT| r = %.20g\n"**,** r**);**

printf**(**"DEBUG REPORT| v1\_coef = %g, v0\_coef = %g\n"**,** v1\_coef**,** v0\_coef**);**

printf**(**"DEBUG REPORT| a[1] = %g, comparing val = %g, diff = %g\n"**,** a**[1],** r**\***a**[3]/2.0** **,** fabs**(**a**[1]-**r**\***a**[3]/2.0));**

*#endif*

**if** **(**a**[1]** **<** r**\***a**[3]/2.0){**

coefs\_q1**[0]** **=** r**/2.0** **+** v0\_coef **,** coefs\_q1**[1]** **=** a**[3]/2.0** **+** v1\_coef**;**

coefs\_q2**[0]** **=** r**/2.0** **-** v0\_coef **,** coefs\_q2**[1]** **=** a**[3]/2.0** **-** v1\_coef**;**

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

coefs\_q1**[0]** **=** r**/2.0** **+** v0\_coef **,** coefs\_q1**[1]** **=** a**[3]/2.0** **-** v1\_coef**;**

coefs\_q2**[0]** **=** r**/2.0** **-** v0\_coef **,** coefs\_q2**[1]** **=** a**[3]/2.0** **+** v1\_coef**;**

**}**

qc1 **=** quad\_roots**(**coefs\_q1**,**root**);**

qc2 **=** quad\_roots**(**coefs\_q2**,**root**+2);**

*#ifdef DEBUG*

*//printf("DEBUG REPORT| 2 coefs1 = %g %g %g\n", coefs\_q1[0], coefs\_q1[1], coefs\_q1[2]);*

*//printf("DEBUG REPORT| 2 coefs2 = %g %g %g\n", coefs\_q2[0], coefs\_q2[1], coefs\_q2[2]);*

printf**(**"DEBUG REPORT| qc1 = %d, qc2 = %d\n"**,** qc1**,** qc2**);**

printf**(**"DEBUG REPORT| roots = %g %g %g %g\n"**,** root**[1],** root**[2],** root**[3],** root**[4]);**

*#endif*

**return** return\_roots**(**qc1**,** qc2**,** root**);**

**}**

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

**int** return\_roots**(int** q1**,** **int** q2**,** **double** **\*** root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

*#ifdef DEBUG*

printf**(**"DEBUG REPORT| return\_roots() called\n"**);**

*#endif*

**if** **(**q1 **==** **0** **&&** q2 **==** **0){**

**if** **(**root**[1]** **<** root**[3]){**

swap**(**root**,2);**

**}** **return** **0;**

**}** **else** **if** **(**q1 **==** **0){**

swap**(**root**,2);**

order**(**root **,** **2);**

**return** **2;**

**}** **else** **if** **(**q2 **==** **0){**

order**(**root**,2);**

**return** **2;**

**}** **else** **if** **(**q1 **==** **1){**

**if** **(**root**[1]** **==** root**[3]** **||** root**[1]** **==** root**[4]){**

order**(**root**,4);**

**return** **1;**

**}**

**}** **else** **if** **(**q2 **==** **1){**

**if** **(**root**[3]** **==** root**[1]** **||** root**[3]** **==** root**[2]){**

order**(**root**,4);**

**return** **1;**

**}**

**}**

*#ifdef DEBUG*

printf**(**"DEBUG REPORT| q1 = %d, q2 = %d\n"**,** q1**,**q2**);**

*#endif*

order **(**root**,4);**

**return** q1**+**q2**;**

**}**

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

**void** swap**(double** **\***nums**,** **int** n**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**double** temp**;**

**int** i**;**

*#ifdef DEBUG\_FULL*

printf**(**"DEBUG REPORT| Roots before calling swap: %g %g %g %g\n"**,**nums**[1],**nums**[2],**nums**[3],**nums**[4]);**

*#endif*

**for** **(**i**=1;**i**<**n**+1;**i**++){**

temp **=** nums**[**i**];**

nums**[**i**]** **=** nums**[**n**+**i**];**

nums**[**n**+**i**]** **=** temp**;**

**}**

*#ifdef DEBUG\_FULL*

printf**(**"DEBUG REPORT| Roots after calling swap: %g %g %g %g\n"**,**nums**[1],**nums**[2],**nums**[3],**nums**[4]);**

*#endif*

**}**

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

**int** quartic\_roots\_of\_unity**(double** r**,** **double** **\***root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**if** **(**r**>0){**

root**[1]** **=** root**[4]** **=** sqrt**(**sqrt**(**r**));**

root**[2]** **=** **-**root**[1];**

root**[3]** **=** **0;**

**return** **2;**

**}**

root**[1]** **=** root**[2]** **=** root**[4]** **=** sqrt**(**sqrt**(-**r**)/2);**

root**[3]** **=** **-**root**[1];**

**return** **0;**

**}**

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

**int** squared\_quadratic**(double** **\***a**,** **double** **\***root**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**double** b**[3]** **=** **{** a**[0],** a**[2]** **,** **1};**

**double** qroot**[3];**

**int** q1**,** q2**;**

**int** q **=** quad\_roots**(**b**,**qroot**);**

**if** **(**q **==** **0){**

root**[1]** **=** sqrt**(** **(**qroot**[1]** **+** sqrt**(**qroot**[1]\***qroot**[1]** **+** qroot**[2]\***qroot**[2])** **)** **/** **2.0** **);**

root**[3]** **=** **-**root**[1];**

root**[2]** **=** root**[4]** **=** sqrt**(** **(**qroot**[1]** **-** sqrt**(**qroot**[1]\***qroot**[1]** **+** qroot**[2]\***qroot**[2])** **)** **/** **2.0** **);**

**return** **0;**

**}** **else** **if** **(**q **==** **1){**

b**[0]** **=** **-**qroot**[1];** b**[1]** **=** **0;** b**[2]** **=** **1;**

q1 **=** quad\_roots**(**b**,**root**);**

root**[3]** **=** root**[1];**

root**[4]** **=** root**[2];**

**if** **(**q1**==2){**order**(**root**,4);return** **3;}**

**return** **0;**

**}**

b**[0]** **=** **-**qroot**[1];** b**[1]** **=** **0;** b**[2]** **=** **1;**

**double** c**[]** **=** **{-**qroot**[2],0,1};**

q1 **=** quad\_roots**(**b**,**root**);**

q2 **=** quad\_roots**(**c**,**root**+2);**

**if** **(**q1**+**q2 **==** **4){**

order**(**root**,4);**

**}** **else** **if** **(**q1**==0** **&&** q2**!=0){**swap**(**root**,2);}**

**return(**q1**+**q2**);**

**}**

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

### prog\_2.c

*#include <stdio.h>*

*#include <math.h>*

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

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

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

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

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

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

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

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

print\_statements**();**

**double** a**[4],**root**[5];**

printf**(**"Enter coefficients of Equation x^4+a[3]\*x^3+a[2]\*x^2+a[1]\*x+a[0]=0\n"**);**

printf**(**"in the order a, separated by spaces: "**);**

scanf**(**"%lf %lf %lf %lf"**,&**a**[3],&**a**[2],&**a**[1],&**a**[0]);**

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

**int** quartic\_case **=** rquartic\_roots**(**a**,**root**);**

printf**(**"case: %d\n"**,** quartic\_case**);**

printf**(**"roots: %g, %g, %g, %g\n"**,** root**[1],**root**[2],**root**[3],**root**[4]);**

**}**

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

**double** f\_x**(double** **\***a**,** **double** x**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**return(**a**[3]\***x**\***x**\***x **+** a**[2]\***x**\***x **+** a**[1]\***x **+** a**[0]);**

**}**

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

## Question 3

### prog\_3.c

*#include <stdio.h>*

*#include <math.h>*

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

*# define M\_PI 3.14159265358979323846*

*#endif*

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

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

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

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

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

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

print\_statements**();**

**double** a**[4],**t**[5],**phi**[5],**X**=3.0/8.0,**Y**=1.0/2.0,**b**;**

**int** quartic\_case**;**

a**[2]** **=** **0;**

a**[0]** **=** **-1.0;**

printf**(**" b , case , t[1] , t[2] , t[3] , t[4] , phi[1] , phi[2] , phi[3] , phi[4]\n"**);**

**for** **(**b**=0.05;** b**<1;** b**=**b**+0.05){**

a**[3]** **=** **2.0\*(**X **+** **1.0** **-** b**\***b**)/(**b**\***Y**);**

a**[1]** **=** **2.0\*(**X **-** **1.0** **+** b**\***b**)/(**b**\***Y**);**

quartic\_case **=** rquartic\_roots**(**a**,**t**);**

*//in degrees*

phi**[1]** **=** **2.0\***atan**(**t**[1])** **\*** **(180.0/**M\_PI**);;**

phi**[2]** **=** **2.0\***atan**(**t**[2])** **\*** **(180.0/**M\_PI**);;**

phi**[3]** **=** **2.0\***atan**(**t**[3])** **\*** **(180.0/**M\_PI**);;**

phi**[4]** **=** **2.0\***atan**(**t**[4])** **\*** **(180.0/**M\_PI**);;**

**switch** **(**quartic\_case**)** **{**

**case** **0:** printf**(**"%5.2f\n"**,** b**);** **break;**

**case** **1:** printf**(**"%5.2f , %5d , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f\n"**,** b**,** quartic\_case**,** t**[1],**t**[2],**t**[3],**t**[4],**phi**[1],**phi**[2],**phi**[3],**phi**[4]);** **break;**

**case** **2:** printf**(**"%5.2f , %5d , %9.4f , %9.4f , , , %9.4f , %9.4f ,\n"**,** b**,** quartic\_case**,** t**[1],**t**[2],**phi**[1],**phi**[2]);** **break;**

**case** **3:** printf**(**"%5.2f , %5d , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f\n"**,** b**,** quartic\_case**,** t**[1],**t**[2],**t**[3],**t**[4],**phi**[1],**phi**[2],**phi**[3],**phi**[4]);** **break;**

**case** **4:** printf**(**"%5.2f , %5d , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f , %9.4f\n"**,** b**,** quartic\_case**,** t**[1],**t**[2],**t**[3],**t**[4],**phi**[1],**phi**[2],**phi**[3],**phi**[4]);** **break;**

**}**

**}**

**}**

### prog\_3b.c

*#include <stdio.h>*

*#include <math.h>*

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

*# define M\_PI 3.14159265358979323846*

*#endif*

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

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

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

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

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

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

print\_statements**();**

**double** a**[4],**t**[5],**phi**[5],**X**=3.0/8.0,**Y**=1.0/2.0;**

**int** quartic\_case**;**

printf**(**" b, t[1], t[2], t[3], t[4], phi[1], phi[2], phi[3], phi[4],\n"**);**

**double** b **=** **0.4245797599;** *//numerical solution*

a**[3]** **=** **2.0\*(**X **+** **1.0** **-** b**\***b**)/(**b**\***Y**);**

a**[2]** **=** **0;**

a**[1]** **=** **2.0\*(**X **-** **1.0** **+** b**\***b**)/(**b**\***Y**);**

a**[0]** **=** **-1.0;**

quartic\_case **=** rquartic\_roots**(**a**,**t**);**

*/\*in degrees\*/*

phi**[1]** **=** **2.0\***atan**(**t**[1])** **\*(180.0/**M\_PI**);**

phi**[2]** **=** **2.0\***atan**(**t**[2])** **\*(180.0/**M\_PI**);**

phi**[3]** **=** **2.0\***atan**(**t**[3])** **\*(180.0/**M\_PI**);**

phi**[4]** **=** **2.0\***atan**(**t**[4])** **\*(180.0/**M\_PI**);**

**switch** **(**quartic\_case**)** **{**

**case** **0:** printf**(**"%5.2f\n"**,** b**);** **break;**

**case** **1:** printf**(**"%5.2f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f\n"**,** b**,** t**[1],**t**[2],**t**[3],**t**[4],**phi**[1],**phi**[2],**phi**[3],**phi**[4]);** **break;**

**case** **2:** printf**(**"%5.2f, %9.4f, %9.4f, , , %9.4f, %9.4f,\n"**,** b**,** t**[1],**t**[2],**phi**[1],**phi**[2]);** **break;**

**case** **3:** printf**(**"%5.2f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f\n"**,** b**,** t**[1],**t**[2],**t**[3],**t**[4],**phi**[1],**phi**[2],**phi**[3],**phi**[4]);** **break;**

**case** **4:** printf**(**"%5.2f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f, %9.4f\n"**,** b**,** t**[1],**t**[2],**t**[3],**t**[4],**phi**[1],**phi**[2],**phi**[3],**phi**[4]);** **break;**

**}**

**}**

### prog\_3c.c

*#include <stdio.h>*

*#include <math.h>*

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

*# define M\_PI 3.14159265358979323846*

*#endif*

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

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

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

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

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

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

print\_statements**();**

**double** a**[4],**t**[5],**phi**[5];**

**int** quartic\_case**;**

**double** R **=** **6378137;**

**double** b **=** **6356752/**R**;**

**double** X **=** **4980265.80632/**R**;**

**double** Y **=** **4950022.89191/**R**;**

a**[3]** **=** **2.0\*(**X **+** **1.0** **-** b**\***b**)/(**b**\***Y**);**

a**[2]** **=** **0;**

a**[1]** **=** **2.0\*(**X **-** **1.0** **+** b**\***b**)/(**b**\***Y**);**

a**[0]** **=** **-1.0;**

quartic\_case **=** rquartic\_roots**(**a**,**t**);**

printf**(**"Quartic Case: %d\n"**,** quartic\_case**);**

*/\*in degrees\*/*

phi**[1]** **=** **2.0\***atan**(**t**[1]);**

phi**[2]** **=** **2.0\***atan**(**t**[2]);**

phi**[3]** **=** **2.0\***atan**(**t**[3]);**

phi**[4]** **=** **2.0\***atan**(**t**[4]);**

**switch** **(**quartic\_case**)** **{**

**case** **2:**

printf**(**" b, t[1], phi[1], t[2], phi[2],\n"**);**

printf**(**"%8.6f, %15.6f, %15.6f, %15.6f, %15.6f,\n"**,** b**,** t**[1],**phi**[1],**t**[2],**phi**[2]);**

**break;**

**case** **4:**

printf**(**" b, t[1], phi[1], t[2], phi[2], t[3], phi[3], t[4], phi[4],\n"**);**

printf**(**"%8.6f, %15.6f, %15.6f, %15.6f, %15.6f, %15.6f, %15.6f, %15.6f, %15.6f\n"**,** b**,** t**[1],**phi**[1],**t**[2],**phi**[2],**t**[3],**phi**[3],**t**[4],**phi**[4]);**

**break;**

**}**

printf**(**"\nCoordinate directly below the Satellite (rescaled to actual size)\n" **);**

printf**(**"X = %20.10g m, Y = %20.10g m\n\n"**,** R**\***cos**(**phi**[1]),** R**\***b**\***sin**(**phi**[1]));**

printf**(**"Angle to Equator = %20.10g\n"**,** atan**(**tan**(**phi**[1])/**b**)** **\*(180.0/**M\_PI**));**

printf**(**"Distance = %20.10g m\n"**,** R**\***sqrt**(** **(** X**-**cos**(**phi**[1]))\*(** X**-**cos**(**phi**[1]))** **+** **(** Y**-**b**\***sin**(**phi**[1]))\*(** Y**-**b**\***sin**(**phi**[1]))** **));**

**}**

## Mastery Section

### 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, 00733164, 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, 00733164, 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, 00733164, M3SC \*/*

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

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

printf**(**"\n Course Code: 00733164, M3SC"**);**

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

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

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

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

**}**

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

### rcubic\_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, 00733164, 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, 00733164, 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, 00733164, 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, 00733164, 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, 00733164, 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, 00733164, 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**;**

**}**

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

### rquartic\_complex.c

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

*#include <stdio.h>*

*#include <math.h>*

*#include <complex.h>*

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

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

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

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

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

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

**int** rquartic\_roots**(double** complex a3**,** **double** complex a2**,** **double** complex a1**,** **double** complex a0**,** **double** complex **\***r1**,** **double** complex **\***r2**,** **double** complex **\***r3**,** **double** complex **\***r4**){**

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

**int** qc1**,** qc2**;**

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

**double** epsilon **=** **1e-5;**

*/\* Optimization Cases that Skip the R-Quartic Method \*/*

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

**\***r1 **=** **0;**

qc1 **=** rcubic\_roots**(**a3**,**a2**,**a1**,**r2**,**r3**,**r4**);**

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

**if** **(**qc1 **==** **1){**

**return** **0;**

**}**

**return** **2;**

**}**

**return** **4;**

**}**

**else** **if** **(**a3**==**zero **&&** a2**==**zero **&&** a1**==**zero**)** **{**

**\***r1 **=** csqrt**(**csqrt**(-**a1**));**

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

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

**\***r4 **=** **-**csqrt**(-**csqrt**(-**a1**));**

**return** **4;**

**}** */\* Part ii \*/*

**else** **if** **(**a3**==**zero **&&** a1**==**zero**){**

qc1 **=** quad\_roots**(1.0,**a2**,**a0**,**r1**,**r2**);**

**\***r3 **=** csqrt**(\***r1**);**

**\***r4 **=** **-**csqrt**(\***r2**);**

**if** **(**qc1**==1){**

**\***r1 **=** **\***r3**;**

**\***r2 **=** **\***r4**;**

**return** **2;**

**}**

**\***r1 **=** **-** **\***r3**;**

**\***r2 **=** **-** **\***r4**;**

**return** **4;**

**}**

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

**double** complex coefs\_q1**[3],**coefs\_q2**[3];**

coefs\_q1**[2]** **=** coefs\_q2**[2]** **=** **1.0;**

**double** complex b**[]** **=** **{4.0\***a0**\***a2 **-** a1**\***a1 **-** a0**\***a3**\***a3 **,** a1**\***a3 **-** **4.0\***a0 **,** **-**a2**};**

rcubic\_roots**(**b**[2],**b**[1],**b**[0],**r1**,**r2**,**r3**);**

**double** complex r **=** **\***r1**;**

**if** **(**cabs**(\***r2**)** **>** cabs**(\***r1**)** **&&** cabs**(\***r2**)** **>** cabs**(\***r3**)){**

r **=** **\***r2**;**

**}** **else** **if** **(**cabs**(\***r3**)** **>** cabs**(\***r1**)** **&&** cabs**(\***r3**)** **>** cabs**(\***r2**)){**

r **=** **\***r3**;**

**}**

**double** complex v0\_coef **=** csqrt**(** **(**r**/2.0)\*(**r**/2.0)** **-** a0 **);**

**double** complex v1\_coef **=** csqrt**(**a3**\***a3**/4.0** **+** r **-** a2**);**

**double** complex a1\_comparison **=** **(**r**/2.0** **+** v0\_coef**)\*(**a3**/2.0** **-** v1\_coef**)** **+** **(**a3**/2.0** **+** v1\_coef**)\*(**r**/2.0** **-** v0\_coef**);**

**if** **(**cabs**(**a1 **-** a1\_comparison**)** **<** epsilon**){**

coefs\_q1**[0]** **=** r**/2.0** **+** v0\_coef **,** coefs\_q1**[1]** **=** a3**/2.0** **+** v1\_coef**;**

coefs\_q2**[0]** **=** r**/2.0** **-** v0\_coef **,** coefs\_q2**[1]** **=** a3**/2.0** **-** v1\_coef**;**

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

coefs\_q1**[0]** **=** r**/2.0** **+** v0\_coef **,** coefs\_q1**[1]** **=** a3**/2.0** **-** v1\_coef**;**

coefs\_q2**[0]** **=** r**/2.0** **-** v0\_coef **,** coefs\_q2**[1]** **=** a3**/2.0** **+** v1\_coef**;**

**}**

qc1 **=** quad\_roots**(**coefs\_q1**[2],**coefs\_q1**[1],**coefs\_q1**[0],**r1**,**r2**);**

qc2 **=** quad\_roots**(**coefs\_q2**[2],**coefs\_q2**[1],**coefs\_q2**[0],**r3**,**r4**);**

**return** **(**qc1 **==** **1** **&&** qc2**)** **==** **1** **?** **0** **:** **(**qc1 **==** **1** **||** qc2 **==** **1)** **?** **2** **:** **4;**

**}**

### 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** rquartic\_roots**(double** complex**,** **double** complex**,** **double** complex**,** **double** complex**,** **double** complex **\*,** **double** complex **\*,** **double** complex **\*,** **double** complex **\*);**

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

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

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

*/\* Bhageria, Yadu, 00733164, M3SC \*/*

print\_statements**();**

**double** complex d4**,**d3**,**d2**,**d1**,**d0**,**r1**,**r2**,**r3**,**r4**,**sj**;**

**int** j**;**

printf**(**" j, Re(z1), Im(z1), Re(z2), Im(z2), Re(z3), Im(z3), Re(z4), Im(z4)\n"**);**

**for** **(**j**=0;**j**<81;**j**++)** **{**

sj **=** cos**(**j**\***M\_PI**/80.0)** **+** I **\*** sin**(**j**\***M\_PI**/80.0);**

d4 **=** **4\***sj**-2;**

d0 **=** **-27\***sj**\*(**sj**-1)/**d4**;**

d1 **=** **-9\*(5\***sj**+1)/**d4**;**

d2 **=** **3\*(5\***sj**+1)/**d4**;**

d3 **=** **-(2+4\***sj**)/**d4**;**

rquartic\_roots**(**d3**,**d2**,**d1**,**d0**,&**r1**,&**r2**,&**r3**,&**r4**);**

printf**(**"%3d,%10.6f,%10.6f,%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**),**creal**(**r4**),**cimag**(**r4**));**

**}**

**}**

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

1. “A Guide to Finding the roots of a Quartic Polynomial” by Dr Dan Moore, <http://wwwf.imperial.ac.uk/~drmii/M3SC_2016/QuarticGuide_2016.pdf>, accessed in March 2016 [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Quartic_function#Solving_a_quartic_equation> [↑](#footnote-ref-2)