Table of Contents

[INTRODUCTION 3](#_Toc445167142)

[Question 1 3](#_Toc445167143)

[Test Cases 3](#_Toc445167144)

[Command Line used to Compile Testing Program 4](#_Toc445167145)

[Question 2: 5](#_Toc445167146)

[Choosing Sign in the Quadratic Polynomials 5](#_Toc445167147)

[Optimization Cases 6](#_Toc445167148)

[Here I demonstrate how I deal with the three optimization cases mentioned. 6](#_Toc445167149)

[(i) a0 = 0 6](#_Toc445167150)

[(ii) a3 = a1 = 0 6](#_Toc445167151)

[(iii) a3 = a2 = a1 = 0 6](#_Toc445167152)

[Test Cases 6](#_Toc445167153)

[Case I – given 6](#_Toc445167154)

[Case II – optimization (i) 7](#_Toc445167155)

[Case II – optimization (ii) 7](#_Toc445167156)

[Case II – optimization (iii) 7](#_Toc445167157)

[Command Line used to Compile Testing Program 7](#_Toc445167158)

[Question 3: 8](#_Toc445167159)

[Derivation 8](#_Toc445167160)

[X = 3/8 and Y = 1/2 8](#_Toc445167161)

[Maximum Value of b 9](#_Toc445167162)

[Command Line Used to Compile Testing Program 11](#_Toc445167163)

[Mastery Section 12](#_Toc445167164)

[Appendix 13](#_Toc445167165)

[Tables 13](#_Toc445167166)

[Table from Mastery Section 13](#_Toc445167167)

[Representative C code and files 15](#_Toc445167168)

[Question 1 15](#_Toc445167169)

[lin\_root.c – THIS IS THE VERSION I WANT TESTED 15](#_Toc445167170)

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 very basic 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 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

Case 1:

Name: Bhageria, Yadu

CID: 00733164

Course Code: M3SC

Email Address: yrb13@ic.ac.uk

Time: 12:09:13

Date: Mar 7 2016

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 1

There is one real root (root[1]) and two complex roots (root[2],root[3]).

root[1] = -1, root[2] = 0 + 1i, root[3] = 0 - 1i

f(root[1]) = 0

Case 2:

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

in the order a, separated by spaces: 25 249 560

There is one real root (root[1]) and two complex roots (root[2],root[3]).

root[1] = -3.088285891, root[2] = -10.95585705 + 7.829403468i, root[3] = -10.95585705 - 7.829403468i

f(root[1]) = 1.136868377e-13

Case 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

There is a pair of repeated real roots.

root[1] = 1, root[2] = -3, root[3] = -3

f(root[1]) = -1.776356839e-14

f(root[2]) = 0

f(root[3]) = 0

Case 4:

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

There are three distinct real roots.

root[1] = 2, root[2] = -1, root[3] = -3

f(root[1]) = 5.329070518e-15

f(root[2]) = -8.881784197e-16

f(root[3]) = 0

Case5:

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

There is one real root (root[1]) and two complex roots (root[2],root[3]).

root[1] = 2.291909782, root[2] = -1.645954891 + 1.958466933i, root[3] = -1.645954891 - 1.958466933i

f(root[1]) = 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\_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

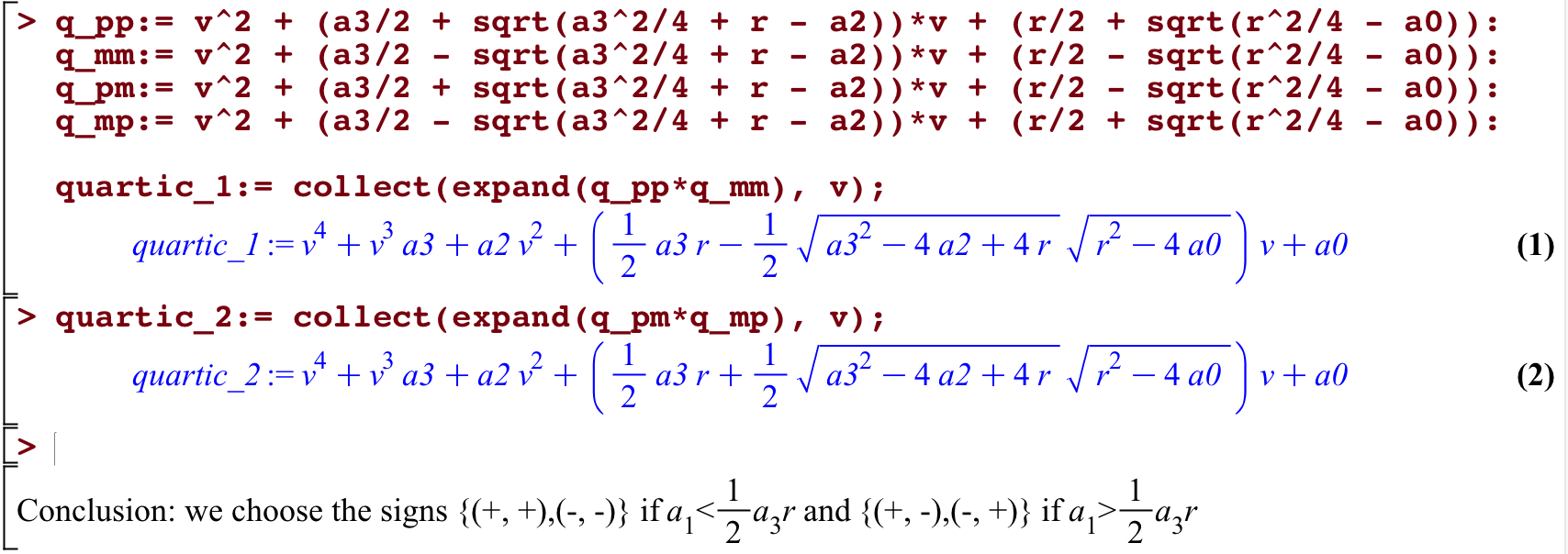
where

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.

### (i) a0 = 0

Here it can immediately be sees that one root is 0 and so we can deal with the cubic as

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

### (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.

### (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 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 – optimization (i)

### Case II – optimization (ii)

### Case II – optimization (iii)

## 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 ouput with b from 0.05 to 0.95:

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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| b | t[1] | t[2] | t[3] | t[4] | phi[1] | phi[2] | phi[3] | phi[4] |
| 0.05 | 0.681 | -0.020 | -0.665 | -109.796 | 68.526 | -2.303 | -67.267 | -178.956 |
| 0.10 | 0.687 | -0.041 | -0.654 | -54.592 | 68.940 | -4.673 | -66.366 | -177.901 |
| 0.15 | 0.690 | -0.063 | -0.640 | -36.054 | 69.235 | -7.186 | -65.226 | -176.823 |
| 0.20 | 0.693 | -0.087 | -0.622 | -26.684 | 69.420 | -9.941 | -63.772 | -175.708 |
| 0.25 | 0.694 | -0.115 | -0.600 | -20.980 | 69.503 | -13.076 | -61.885 | -174.542 |
| 0.30 | 0.694 | -0.148 | -0.570 | -17.109 | 69.487 | -16.825 | -59.352 | -173.310 |
| 0.35 | 0.692 | -0.191 | -0.529 | -14.287 | 69.373 | -21.667 | -55.714 | -171.992 |
| 0.40 | 0.689 | -0.260 | -0.460 | -12.119 | 69.158 | -29.147 | -49.445 | -170.566 |
| 0.45 | 0.685 | -10.388 |  |  | 68.838 | -169.003 |  |  |
| 0.50 | 0.680 | -8.964 |  |  | 68.404 | -167.269 |  |  |
| 0.55 | 0.673 | -7.763 |  |  | 67.846 | -165.320 |  |  |
| 0.60 | 0.664 | -6.731 |  |  | 67.146 | -163.099 |  |  |
| 0.65 | 0.653 | -5.830 |  |  | 66.287 | -160.534 |  |  |
| 0.70 | 0.640 | -5.035 |  |  | 65.242 | -157.531 |  |  |
| 0.75 | 0.625 | -4.328 |  |  | 63.979 | -153.979 |  |  |
| 0.80 | 0.606 | -3.700 |  |  | 62.462 | -149.754 |  |  |
| 0.85 | 0.585 | -3.149 |  |  | 60.650 | -144.763 |  |  |
| 0.90 | 0.560 | -2.678 |  |  | 58.503 | -139.045 |  |  |
| 0.95 | 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 40 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

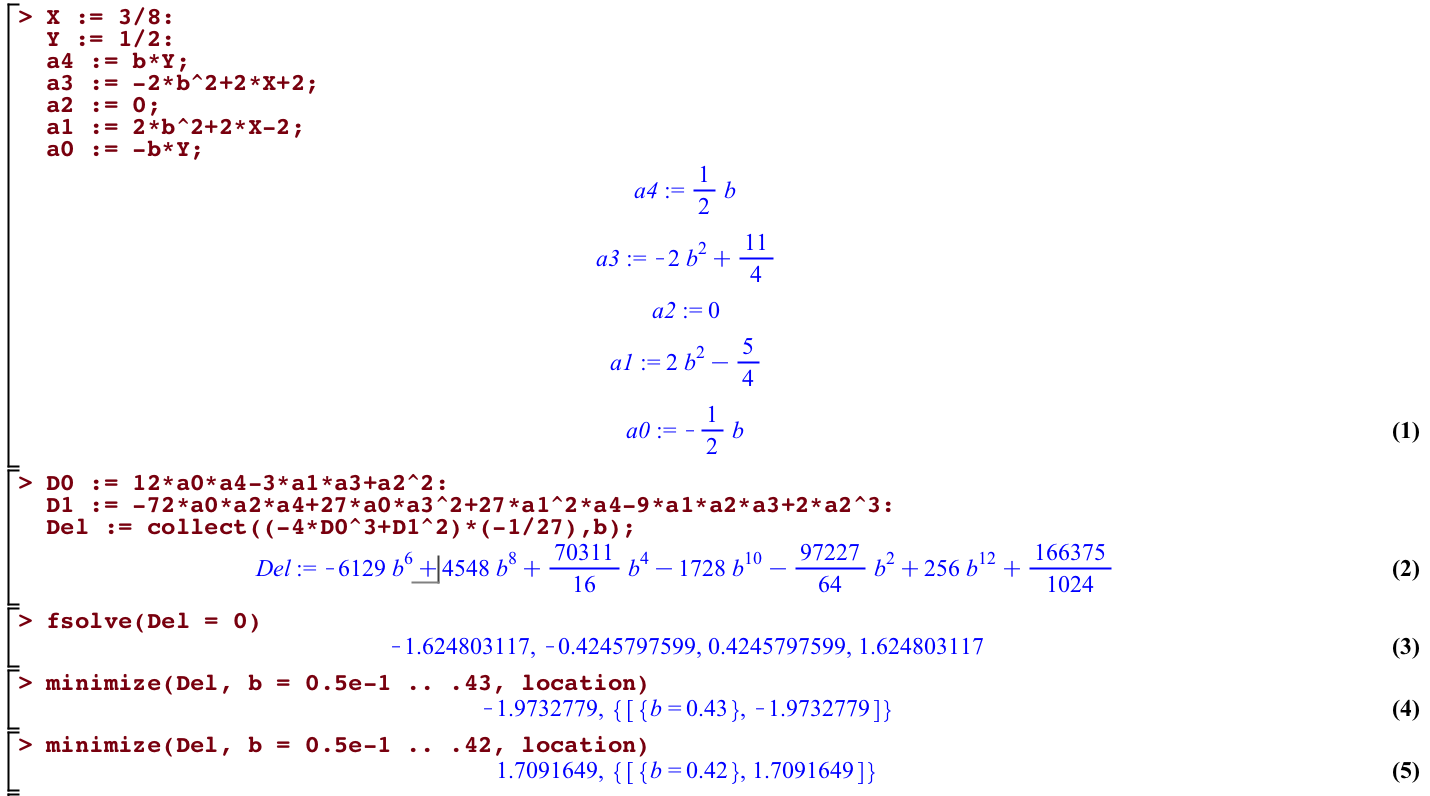
Where

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 the case being dealt with as it 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).

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 maple I get that

## 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

The rquartic\_roots.c is the same as for Question 2. Find prog\_3.c and prog\_3b.c in the appendix along with some Matlab code for producing the plots.

# 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. It has the roots ordered by greater real part first and then greater imaginary part for roots with the same real part.

The plot produced of all the roots to all the equations on the complex plane is as follows

Appendix

# Tables

## Table from Mastery Section

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

Table 3: Values of the three for j between 0 and 100 inclusive. (Mastery Section)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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

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)