F5 Newton Tivenan

Stephen Tivenan

May 2019

1 Newton Final Problem

Main Code

deg= the degree of the polynomial Variable: f_z = the original function that is put into Newton's method
Input Variable: z_n= the first derivative of the original function
Input Variable: TOL= the difference between the last term generated and the previous term generated in Newtons method
Input Variable: x= the real number solution for the function (red solution
Input Variable: z= one of the imaginary roots for the function
Input Variable: n= another imaginary roots for the function
Input Variable: N=determines number of points on the imaginary axis which is (N by N)
Input Variable: P=determines of the numbers of iterations for newtons method
Input Variable: $Size$=this is the size of the marker
Input Variable: S= Is the absolute value of where the graph starts and ends

$x0$=generates the number of values between the maximum and minimum on the x axis as an 1D array
$y0$= generates the number of values between the maximum and minimum on the y axis as an 1D array
for jj in range (1, len(y0)):
———for ii in range (len(x0): (for every element in y0, there is the all the elements of x0)
———mm=complex(y0[jj],x[ii])
———aa=mm.real is the real part of a point in the graph
———bb=mm.image is the imaginary part of point in the graph
———plt.plot(aa,bb,'ko',markersize=size)
This For loop plots all the points black in imaginary plane in the range $-S$ to S for $N * N$ points.
———-for ll in range(P): This For loop uses Newtons method with the variable m, which is a complex number, to arrive to answer b with a being the last difference of the last two iterations

————if TOL≥abs(a):
————————aaa=b.real takes the real part of the answer
————————bbb=b.imag takes the imaginary part of the answer
————————if (aaa-TOL)≤x and x≤(aaa+TOL): (This if statement checks the real part and imaginary part to see if number collected is close enough approximation to one of the answer that is given.)
————————plt.plot(aa,bb,'ro',markersize=size) (If it satisfy the relation, than a red, green, or blue dot is place over the black dot that was initially put at the spot of aa and bb
———- break (Once it reaches its iterations or plots the color dot for the given point the for loop breaks.)

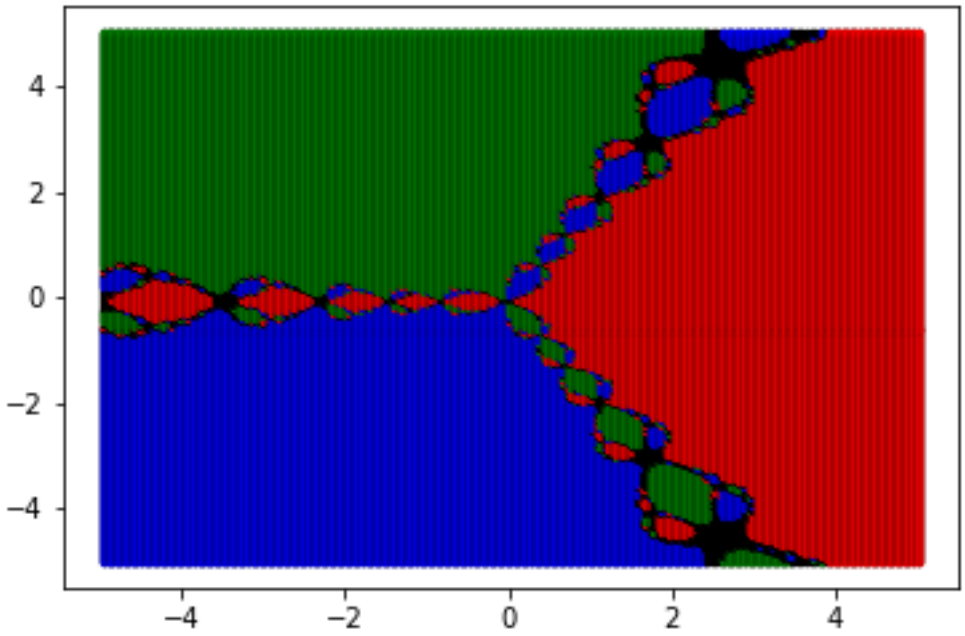
2 Newton Final Problem Part 2

The following section is for the Laguerre method.
Main Code
Input Variable: deg= the degree of the polynomial
Input Variable: f_z = the original function that is put into Newton's method
Input Variable: z_n= the first derivative of the original function
Input Variable: TOL= the difference between the last term generated and the previous term generated in Newtons method
Input Variable: x= the real number solution for the function (red solution
Input Variable: z= one of the imaginary roots for the function
Input Variable: n= another imaginary roots for the function
Input Variable: N=determines number of points on the imaginary axis which is (N by N)
Input Variable: P=determines of the numbers of iterations for newtons method
Input Variable: $Size$=this is the size of the marker
Input Variable: S= Is the absolute value of where the graph starts and ends

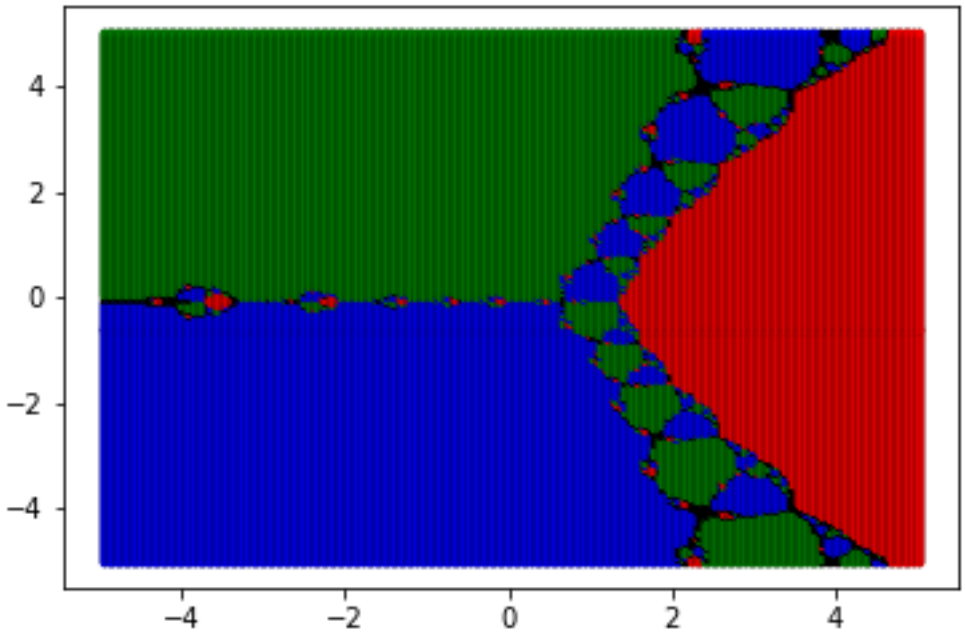
$x0$=generates the number of values between the maximum and minimum on the x axis as an 1D array
$y0$= generates the number of values between the maximum and minimum on the y axis as an 1D array
for jj in range (1, len(y0)):
———for ii in range (len(x0): (for every element in y0, there is the all the elements of x0)
———mm=complex(y0[jj],x[ii])
———aa=mm.real is the real part of a point in the graph
———bb=mm.image is the imaginary part of point in the graph
———plt.plot(aa,bb,'ko',markersize=size)
This For loop plots all the points black in imaginary plane in the range $-S$ to S for $N * N$ points.
———-for ll in range(P): This For loop uses Laguerre method with the variable m, which is a complex number, to arrive to answer a and with a being the last

difference of the last two iterations
————g=a+f
————h=a-f ((Languerre method produces two denominators possibilities)
————if abs(g)<abs(h):
————————-p=h
————if abs(h)<abs(g):
————————p=g
The next two lines determine which denominator is bigger. The denominator that is chosen is the bigger of the two possibilities.
————lll=(deg)/p
————a=mm-lll
————m=a
The lines above determine the next value of *mm* that needs to be plug in and determines what *mm* has to be subtracted by
————aaa=a.real takes the real part of the answer
————bbb=a.imag takes the imaginary part of the answer
————if (aaa-TOL)≤x and x≤(aaa+TOL): (This if statement checks the real part and imaginary part to see if number collected is close enough approximation to one of the answer that is given.)
————————plt.plot(aa,bb,'ro',markersize=size) (If it satisfy the relation, than a red, green, or blue dot is place over the black dot that was initially put at the spot of *aa* and *bb*
————- break (Once it reaches its iterations or plots the color dot for the given point the for loop breaks.)

3 Picture of Newton's Method

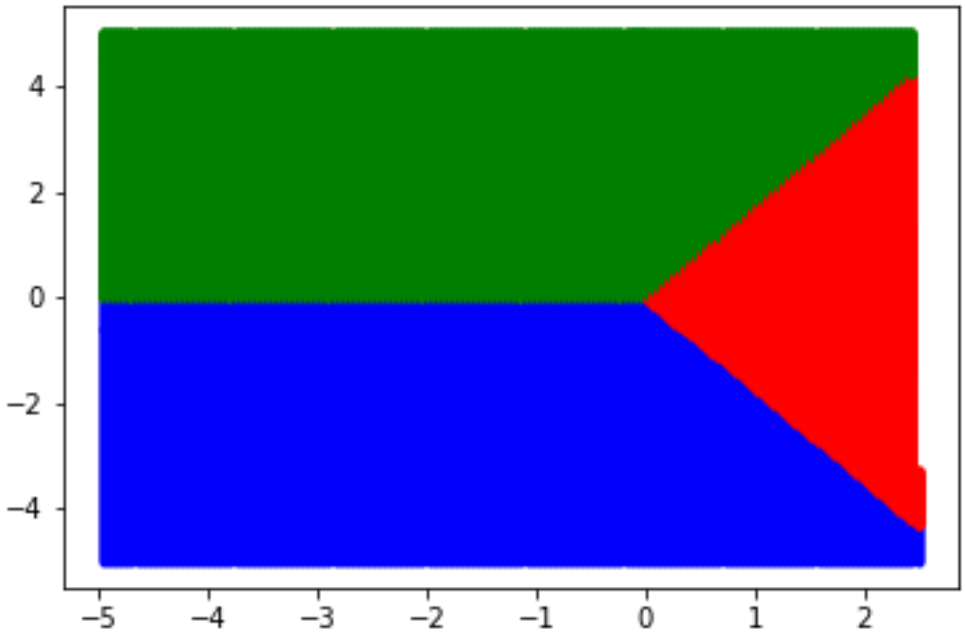


The graphic above is from the equation $z^3 - 1$

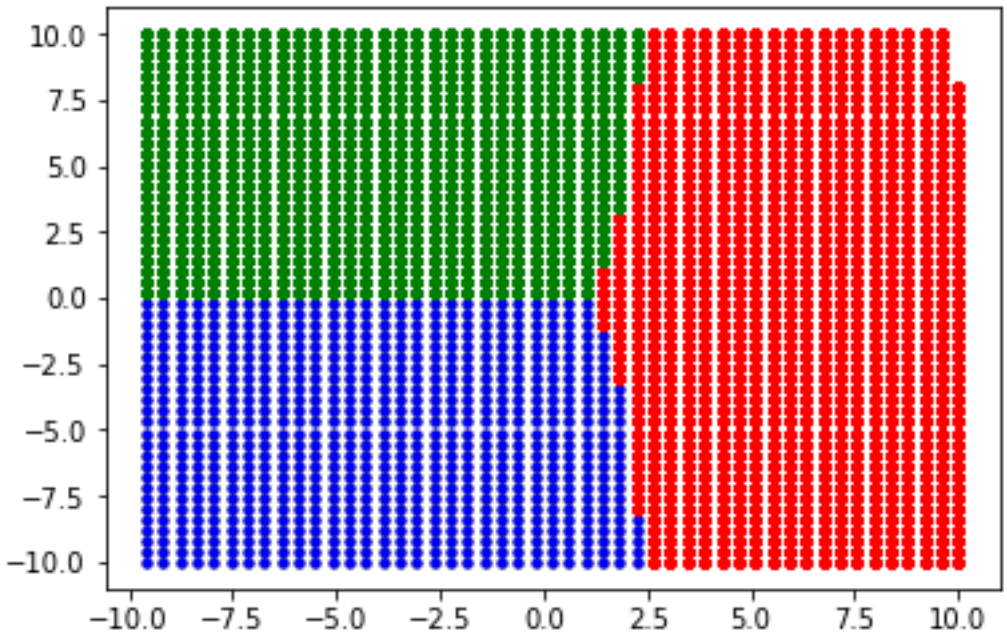


The graphic above is from the equation $z^3 - 2z^2 - 1$

4 Picture of Languerre's Method



The graphic above is from the equation $z^3 - 1$



The graphic above is from the equation $z^3 - 2z^2 - 1$ (I was unable to finish iterating a full good picture with 200 by 200)

5 Discussion

The polynomial for $z^3 - 1$ seems to have more real answers than the $z^3 - 2z^2 - 1$, polynomial. In the Languerre method it seem to be the opposite there were more real answers in the $z^3 - 2z^2 - 1$ polynomial than $z^3 - 1$. In Newton's method there more intertangled solutions compared to Languerre method. There seem to be defined lines which separated the solution in the Languerre method. But in the Newton's method, there was plenty overlapping of converging answers. The Newton method was much quicker than the Languerre Method. It took so long that I could even print out a full 200 by 200 graph for the last polynomial and had to print 100 by 100 graph. I did not make any changes to my code to make it run faster, but if I were I would used the Newton method and add more iterations in order for the For loop to cover all possible values with in the range.