ROOTS OFIMAGINARY TRANSIENT EQUATIONS USING MATLAB® PROGRAM

VenuAkhil Kumar Parakala*

Lucky Purushwani*

Debduhita Bose#

*SMBS, VIT University, Chennai Campus, Vandalur-kelambakam road, Chennai-600127,

#School of Electrical and ElectronicsEngineering, SRM University, Kattankulathur, Chennai

Email id: venu.akhil.94@gmail.com

ABSTRACT

This paper deals with the implementation of idea in which we have modified the Newton – Raphson method which is a approximation method to obtain the roots of transient equation to the Complex form, then from that we get new iterative formula which contain two parts the real part and the imaginary part and then we solve for both the parts and obtain the real and imaginary part of the root then using this algorithm. We have also generated a matlab® code which dividesthe whole equation in two part and solve them simultaneously. This code is applicable for all kinds of non- linear transient equations.

Keywords – Complex roots, Newton-Raphson method, Transient Equation, Matlab® Coding.

INTRODUCTION-

Complex analysis can roughly be thought of as a subject that applies the theory of calculus to imaginary numbers. But what is exactly are imaginary numbers? We cannot actually take the square root of a negative number, but let's pretend we can, only by using the symbol (-1)^1/2, symbolically called iota denoted by 'i' .Also, numerical analysis is concerned with the mathematical derivation description and analysis of methods of obtaining numerical solutions of mathematical problems; such as finding complex roots of linear and non – linear equations by using iteration methods.

ALGORITHM FOR THE CODE

COMPLEX NEWTON – RAPHSON FORMULA (COMPLEX ITERATION) :

For computing complex and real root of non – linear transient equations of form f(x)=0. Depending on given initial point, we write the real iterative formula and the imaginary iterative formula of Newton – Raphson in the following procedure such that it is computing all roots of equations step by step andthe number of iterations depended on the value of tolerance .

In Newton formula Replace the variable x, with the variable in both sides then we get the following form z

$Z_{k+1} = Z_k - f(Z_k)/f'(Z_k)$ for k=0,1,2.....

Now we replace Z by x+iy in the above formula to get the complex form of equation.

$(x+iy)_{k+1} = (x+iy)_k - f(x+iy)_k / f'(x+iy)_k$ for k=0,1,2....

$x_{k+1}+iy_{k+1} = (xk+iyk-f(xk+iyk)/f'(xk+iyk)$ for k=0,1,2....

then on simplifying the equation we get-

$x_{k+1}+iy_{k+1} = [x_k - f(x_f)/f'(x_f)] + i[y_k - f(y_f)/f'(y_f)]$ for k=0,1,2....

from the above equations we can obtain the real and imaginary part and then these equations were used to evaluate real and imaginary roots of non – linear equations respectively.

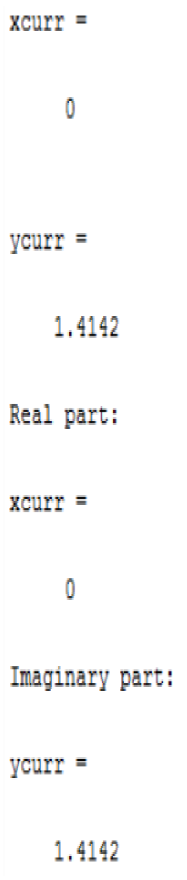
We use this algorithm to generate matlab code.

MATLAB CODE

```
clc

clear all

syms x y z

choice=input('Enter 1 if expression is real or 2
if imaginary.')

if(choice==1)

   f=input('Enter real expression in terms of
x.')

fx=diff(f, x)

   f0=input('Enter closest value of function
where it changes sign.');

   f1=input('Enter closest value of function
where it changes sign
again.');

xprev=(f0+f1)/2;

xcurr=1;

iter=0;

check=0;

while((check==0)&&(iter<=200))

xcurr=xprev-((subs(f, x, xprev))/(subs(fx, x,
xprev)))

if((vpa(xprev, 3))==(vpa(xcurr, 3)))

check=1;

else

xprev=xcurr

xstore(iter+1)=xcurr;

iter=iter+1

end

end

xcurr

else

   f=input('Enter imaginary expression in
terms of z.')

fz=diff(f, z)

   f0=input('Enter closest value of function
where it changes sign.');

   f1=input('Enter closest value of function
where it changes sign
again.');

xprev=(f0+f1)/2;

yprev=xprev;

xcurr=1;

ycurr=1;

iter=0;

check=0;

while((check==0)&&(iter<=200))

xcurr=(xprev-real((subs(subs(subs(f, z,
(x+1i*y)), x, xprev), y,
yprev))/(subs(subs(subs(fz, z, (x+1i*y)), x,
xprev), y, yprev))))

ycurr=(yprev-imag((subs(subs(subs(f, z,
(x+1i*y)), x, xprev), y,
yprev))/(subs(subs(subs(fz, z, (x+1i*y)), x,
xprev), y, yprev))))
```

```
    if(((vpa(xprev, 3))==(vpa(xcurr,
3)))&&((vpa(yprev, 3))==(vpa(ycurr, 3))))

check=1;
else
xprev=xcurr
xstore(iter+1)=xcurr;
yprev=ycurr
ystore(iter+1)=ycurr;
iter=iter+1
end
end
disp('Real part:')
xcurr
disp('Imaginary part:')
ycurr
end
```

SAMPLE OUTPUT



```
xcurr =

     0


ycurr =

   1.4142

Real part:

xcurr =

     0

Imaginary part:

ycurr =

   1.4142
```

First this code will ask the user weather the root is imaginary or real, when the user enter his choice then code will ask for the equation of which root has to be found.
After we give the equation code finds the differentiation of that equation then it asks for boundary values between which the root exists. After that the code will run for number of iterations until the next value is same as previous value. And then it will give the xcurr and ycurr value as shown and hence we can obtain the roots of the given equation.

CONCLUSION

In this paper we have modified Newton – Raphson algorithms to compute the complex roots of non – linear transient equations, then we had written down a program code usingMatlab®software. This code is applicable for both complex and real equations solution as, by taking complex initial point to get the complex roots (if they exist), if not we take real initial points to get real roots of the equationWe have also increased the accuracy of the code by increasing the number of iterations performed and checking is done till the number of roots match the order of the input equation.

REFERENCES

Websites:

[1] http://www.ijcee.org/papers/714-SE0066.pdf

[2] http://benisrael.net/NEWTON-GI-2.pdf

[3] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4104639&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4104639

[4] http://www.iosrjen.org/Papers/vol4_issue4%20(part-1)/A04410107.pdf

[5] http://www.numbertheory.org/book/cha5.pdf

Books:

[6] Steven C Chapra, (2008): Applied Numerical Methods with MatlabFor Engineering and science, 2nd edition tata McGraw–Hill Publishing Company.