**Inverse Quadratic Interpolation Method**

In numerical analysis, inverse quadratic interpolation is a root-finding algorithm, meaning that it is an algorithm for solving equations of the form . The idea is to use Lagrange quadratic interpolation to approximate the .

The Methods requires three initial guesses , and . Starting with initial values , and , we construct a Lagrange interpolating polynomial of 2nd order of inverse of using the initial points given.

We put in the Lagrange interpolating polynomial of 2nd order of inverse of to get

We then use this new value of  as  and repeat the process, using ,  and instead of , and . We continue this process, solving for , , etc., until we reach a sufficiently high level of precision .

**Algorithm**

mu=0.006; % drift

sigma=0.000012; % diffusion

n=100; % number of stocks

k=1.1; % the final value of stock is k times the initial value

f\_x = @(x)exp(mu\*x-(sigma\*sigma\*x/2)+(sigma\*(4\*sin(100\*x)+3\*cos(150\*x)+5\*sin(170\*x)+8\*cos(190\*x)+0.9)))/50-k;

x\_0=1000; % initial guess 1

x\_1=2000; % initial guess 2

x\_2=3000; % initial guess 3

accuracy=0.00001; % to converge the method

inverse\_quadratic\_method(x\_0,x\_1,x\_2,f\_x,accuracy);

function[Root]=inverse\_quadratic\_method(x\_0,x\_1,x\_2,f\_x,accuracy)

while true

count=count+1;

if count>1000 % number of iterations becomes large

fprintf('The method doesnt converge')

break

elseif f\_x(x\_1)==0 % x\_1 is already the root

Root=x\_1;

fprintf('Number of iterations are %f\n',count)

fprintf('Root is %f\n',Root)

break

elseif f\_x(x\_2)==0 % x\_2 is already the root

Root=x\_2;

fprintf('Number of iterations are %f\n',count)

fprintf('Root is %f\n',Root)

break

elseif f\_x(x\_0)==0 % x\_0 is already the root

Root=x\_0;

fprintf('Number of iterations are %f\n',count)

fprintf('Root is %f\n',Root)

break

elseif f\_x(x\_1)==f\_x(x\_2) % denominator in lagrange interpolation becomes zero

fprintf('The denominator becomes zero. Try other initial guesses')

break

elseif f\_x(x\_0)==f\_x(x\_2) % denominator in lagrange interpolation becomes zero

fprintf('The denominator becomes zero. Try other initial guesses')

break

elseif f\_x(x\_0)==f\_x(x\_1) % denominator in lagrange interpolation becomes zero

fprintf('The denominator becomes zero. Try other initial guesses')

break

else

if abs(f\_x(x\_2))<accuracy % condition for convergence of root

Root=x\_2;

fprintf('Number of iterations are %f\n',count)

fprintf('Root is %f\n',Root)

break

elseif abs(f\_x(x\_1))<accuracy % condition for convergence of root

Root=x\_1;

fprintf('Number of iterations are %f\n',count)

fprintf('Root is %f\n',Root)

break

elseif abs(f\_x(x\_0))<accuracy % condition for convergence of root

Root=x\_0;

fprintf('Number of iterations are %f\n',count)

fprintf('Root is %f\n',Root)

break

else

% hardcoding the inverse lagrange interpolating polynomial

t=x\_2;

x\_2=(f\_x(x\_2)\*f\_x(x\_1)\*x\_0)/((f\_x(x\_0)-f\_x(x\_1))\*(f\_x(x\_0)-f\_x(x\_2)))+(f\_x(x\_2)\*f\_x(x\_0)\*x\_1)/((f\_x(x\_1)-f\_x(x\_2))\*(f\_x(x\_1)-f\_x(x\_0)))+(f\_x(x\_0)\*f\_x(x\_1)\*x\_2)/((f\_x(x\_2)-f\_x(x\_1))\*(f\_x(x\_2)-f\_x(x\_0)));

x\_0=x\_1;

x\_1=t;

end

end

end

end