```
%Choose our parameters
M = 5000; %5000 seems to be the cutoff before PC takes off like a airplane
roots = [];
%Calculate the first 3 roots
a1 = 2.4;
b1 = 2.5;
p1 = getP(a1, b1);
roots(1) = bisection(p1, a1, b1, 1e-15);
a2 = 5.5;
b2 = 5.6;
p2 = getP(a2, b2);
roots(2) = bisection(p2, a2, b2, 1e-15);
a3 = 8;
b3 = 9;
p3 = getP(a3, b3);
roots(3) = bisection(p3, a3, b3, 1e-15);
%Iterate over the roots M times, using the average distance between the first 3
%roots to increment a
diff = roots(3)-roots(2);
a = a3;
b = b3;
for i=4:M
    a = a+diff;
    b = b+diff;
    while sameSign(f(a), f(b))
        b = b+1;
    end
    roots(i) = bisection(getP(a, b), a, b, 1e-12);
    diff = abs(roots(i) - roots(i-1));
    if((f(roots(i)))> 1e-5)
        disp("Broke at " + i + " with x = " + roots(i)+" and f(x) = "+f(roots(i)));
    end
end
close all
%Plot the bessel
z = 0:0.1:20;
plot(z, besselj(0, z));
line(xlim, [0,0], 'Color', 'k');
figure;
m = linspace(1, M, M);
%Scatterplot of data
plot(m, roots, "r*");
hold on
%Compute our linear approximation
p1 = polyfit(m, roots, 1);
f1 = polyval(p1, m);
plot(m, f1, "b");
hold off
%In finding the slope, we can now compute the asymptotic behaviour of xM, by solving for↙
x and taking a limit
format longg
```

```
disp("alpha = ")
disp(p1(1))
disp("beta = ")
disp(p1(2)/p1(1))

%-----------------------------------FUNCTIONS-----------------------------------------↙
%
%Function that runs the bisection method
function p = bisection(p, a, b, tol)
    simp = true;
    while simp
        [p, a, b] = bisectionIteration(p, a, b);
        if f(p) == 0 || abs(b-a)/2 < tol
            simp = false;
        end
    end
end
%Function that runs the one iteration of the bisection method
function [p, a, b] = bisectionIteration(p, a, b)
    if sameSign(f(p), f(a))
        a = p;
        p = getP(a, b);
    elseif sameSign(f(p), f(b))
        b = p;
        p = getP(a, b);
    else
        error("f(a) & f(b) must have opposite signs");
    end
end
%Function that defines our function
function y = f(x)
    y = besselj(0, x);
end
%Function that computes p for the bisection method
function p = getP(a, b)
    p = a + (b-a)/2;
end
%Functon that checks if two variables have the same sign
function isSame = sameSign(a, b)
    if(a>=0 && b>=0 || a<0 && b<0)
        isSame = true;
    else
        isSame = false;
    end
end
```