# Computing Assignment 4:Finding the Roots of Bessel Functions

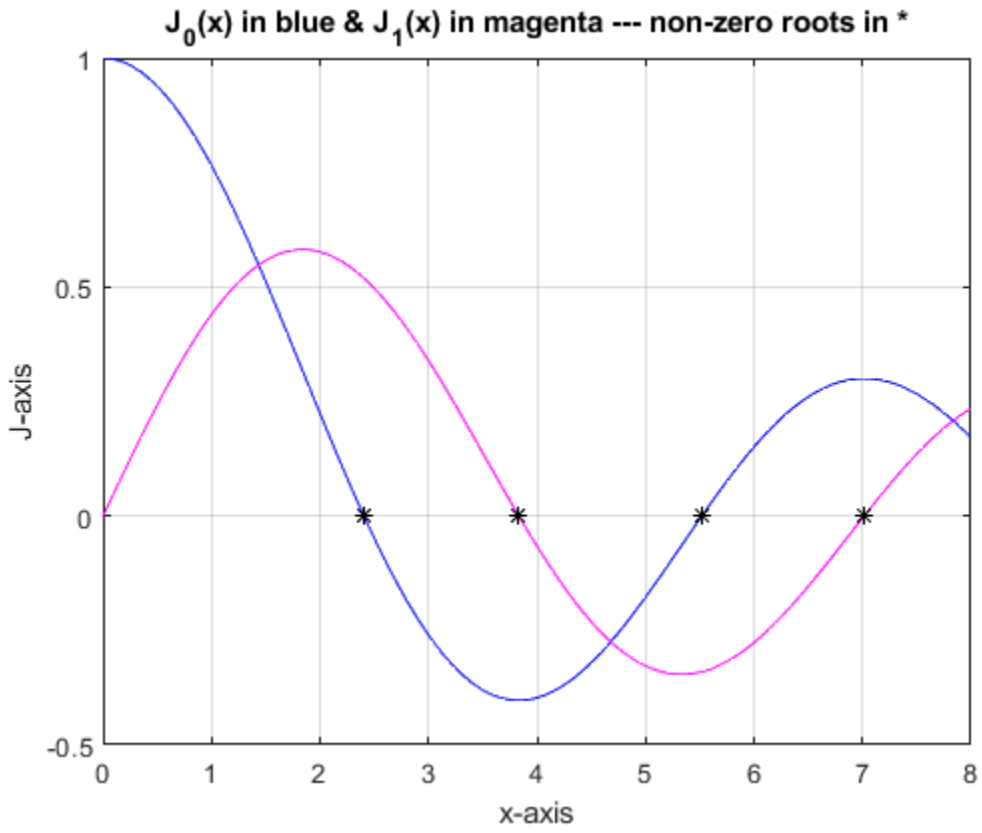
```
        CA4_bessel.m -- djm -- 03 feb 2019
        Editted by Kai Sackville-Hii (Feb 11, 2019)

clear;

% parameters:
mat_size = 16;

%  matrix size (kP = # of zeros, mP-1 = max bessel index)
kP = 2;
mP = kP;

%  matrix for zeros
Amk_1 = zeros(mP,kP);
Amk_2 = zeros(mP,kP);

% Count for function evals
Nevals=0;
Nevals1_arr = [];
Nevals2_arr = [];

% Convergence tolerance
% tol=1e-6;
tol=1e-6;

%  define bessel function
bfunc = @(x,mm) besselj(mm,x);

disp("Finding the roots of the Bessel Funtion:");
disp("")
%
 ========================================================================
 %
%                         PART 1: MATLABS FZERO
     %
%
 ========================================================================
 %

disp(" ===== Part 1: Matlabs fzero ===== ")
%  fzero options - 2 versions
opt1 = optimset('TolX',tol,'Display','final');
opt2 = optimset('TolX',tol,'Display','iter');
opt3 = optimset('TolX',tol,'Display','off');

%
% The following section shows how to find some roots using fzero
% You can use this to create your all fzero code
```

```
%


ri = [2.4,2.6];    % Initial Bracket
mm = 0;

for row=1:mat_size

    mm = row-1;

    for col=1:mat_size
        Nevals = 0;

        % get zero
        [Amk_1(row,col), err, exitflag,output] = fzero(@(x)
 bfunc(x,mm),ri, opt3);

        % next root estimate
        root_est = Amk_1(row, col)+pi;
        a = root_est - (pi/2);
        b = root_est + (pi/2);

        % set brackets and set Nevals
        ri = [a, b];
        Nevals = Nevals + output.funcCount;
        Nevals1_arr(end+1) = Nevals;
    end

    % set bracket for next Jmm
    brac_a = Amk_1(row, 1);
    brac_b = Amk_1(row, 2);
    ri =  [brac_a, brac_b];

end

%
 ========================================================================
 %
%                          PART 2: NEWTONS METHOD.
     %
%
 ========================================================================
 %
%
% The following section shows how to find some roots using Newton's
 method
% You can use this section to create your all Newton's method code
%

disp(" ===== Part 2: Newtons Method ===== ")

nm = @(x,m) x - besselj(m,x)/(0.5*(besselj(m-1,x)-besselj(m+1,x)));

%Initial guess for z_{1,1}
```

```
zi = 1;
mm = 0;
Nevals=0;

for row=1:mat_size
    mm = row-1;

    for col=1:mat_size

        Nevals = 0;
        zn=zi;
        check=1;
        %Root finding loop

        while abs(check) > tol
            Amk_2(row,col) = nm(zn,mm);
            check = Amk_2(row,col)-zn;

            % Here we track function evals for Newton
            Nevals = Nevals + 3;
            zn = Amk_2(row,col);
        end

        Nevals2_arr(end+1) = Nevals;
        zi = zn+pi;

    end

    brac_a = Amk_2(row, 1);
    brac_b = Amk_2(row, 2);
    zi =  brac_a + ((brac_b-brac_a)/2);

end


%
% Output 2x2 matrix and Nevals for checking
%
A22 = Amk_1(1:2,1:2);
Zmk = Amk_2;

disp("avg Nevals of fzero")
disp(mean(Nevals1_arr))
disp("avg Nevals of Netowns")
disp(mean(Nevals2_arr))

%  plot bessel function (useful for testing?)
figure(1000);  clf
xx = 0:0.05:8;
plot(xx,besselj(0,xx),'b');  hold on
plot(xx,besselj(1,xx),'m');
plot(A22,0*A22,'k*')
grid on
axis normal
```

```
title('J_0(x) in blue & J_1(x) in magenta --- non-zero roots in *')
xlabel('x-axis')
ylabel('J-axis')
```

```
Finding the roots of the Bessel Funtion:
 ===== Part 1: Matlabs fzero =====
 ===== Part 2: Newtons Method =====
avg Nevals of fzero
    6.2305

avg Nevals of Netowns
    8.9531
```




*Published with MATLAB® R2018b*