
```

%
% CA4_nm.m -- dhpham -- 12 feb 2019
%
% MAIN USAGE: Create a 16x16 matrix, each row k containing 16
% positive roots
% of the m-th Bessel function, where m = 0,1,...,15, k = 1,2,...,16.
% Find these roots using MATLAB's fzero.
%
% INPUT: Use an existing 'tolfz' as the convergence tolerance if it
% exists,
% otherwise use tol = 1e-6.
%
% OUTPUT:
% 'Amk': the 16x16 matrix containing roots to the m Bessel
% funcs.;
% 'mean_Nevals': the mean # of func. evals per iteration
% 'bsl_reserr': the residual error of J_m(x) evaluated at each
% root z_{m,k} in Amk.
%

disp('==== USING FZERO ====')

if (exist('tolfz','var') ~= 1)
    % Default convergence tolerance
    tol=1e-6
else
    % check if existant tol is scalar, set to default if not
    if (sum(size(tolfz)==[1 1])~=2)
        disp('tol is not correct size! setting default tol = 1e-6')
        tol = 1e-6;
    else
        tol = tolfz;
    end
end

% matrix size (kP = # of zeros, mP-1 = max bessel index)
kP = 16;
mP = kP;

% matrix for zeros
Amk = zeros(mP,kP);

% Count for function evals
Nevals = 0;

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

% fzero options, no output unless non-convergence
opt1 = optimset('TolX',tol,'Display','notify');
```

```

% For Bessel functions J_m(x), 0 <= m <= 'mP-1'
for mm = 0:mP-1
    if mm < 1
        % Initial bracket for first root z_{0,1}:
        % using the hint given in CA4.pdf where z_{0,1} = 2.5
        ri = [2 3];
    else
        % For Bessel functions m > 0, first root z_{m,1} bracketed
        by:
        % z_{m-1,1} < z_{m,1} < z_{m-1,2}
        ri = [Amk(mm,1) Amk(mm,2)];
    end

    % Find 'kP' zeros, 1 <= 'kk' <= 'kP'
    for kk = 1:kP

        % The k-th root is approx. at z_{m,k-1} + pi, where k > 1
        if kk > 1
            % left endpoint of interval
            rL = floor(Amk(mm+1, kk-1) + pi);

            % check for sign change at J_m(rL) and J_m(rL+ii)
            ii = [1 2];
            sign_changes = diff(sign([bfunc(rL, mm) bfunc(rL+ii, mm)]));
            iix = find(sign_changes, 1);

            ri = [rL rL+ii(iix)];
        end

        % Find z_{m,k} using fzero
        [Amk(mm+1, kk), err, exitflag, output] = fzero( @(x)
bfunc(x, mm), ri, opt1);

        Nevals = Nevals + output.funcCount;
    end
end

%
% Output mean Nevals and take residual error as rms
%
mean_Nevals = Nevals / (kP^2)

% residual error as rms when J_m(x) evaluated at roots in Amk
Jmk = zeros(kP);
for mm = 0:mP-1
    Jmk(mm+1, :) = besselj(mm, Amk(mm+1, :));
end
bsl_reserr = rms(Jmk(:))

```

Published with MATLAB® R2018a