```
%
%  CA4_tols.m -- dhpham -- 12 feb 2019
%
%  Run CA4_nm.m and CA4_fzero.m for every value in 'tols', a vector of
%  convergence tolerances.
%
%  For each tolerance value, record the:
%       1)  mean # of function evals per iter. for each procedure,
%       2)  rms of the residual error given by J_m(x), at every root
 found
%       3)  maximum abs. value of the res error when solving the DE in
%           CA4_BesselMovie for 0
%
%   Create a log-plot of the residual errors, and a plot of the mean
%   Nevals.
%

clear

%  vector of convergence tolerances to test
tols = [3:15];

%    mean_Nevals:  mean # of function evals per iter. of procedure
%     bsl_reserr:  take rms of J_m(x) evaluated at all roots found
%  diffeq_reserr:  max abs val of res err; given by CA4_BesselMovie

fzero_data = struct( 'mean_Nevals', zeros(size(tols)), ...
                     'bsl_reserr',  zeros(size(tols)), ...
                     'diffeq_reserr',  zeros(size(tols)) );

nm_data = struct( 'mean_Nevals',    zeros(size(tols)), ...
                  'bsl_reserr',      zeros(size(tols)), ...
                  'diffeq_reserr',     zeros(size(tols)) );

%  Run CA4_nm.m for each value in 'tols'
for toli = tols(1):tols(end)
    %  i-th iteration
    ix = toli - tols(1) + 1;

    %  Set tol for CA4_nm.m
    tolnm = 1 * 10^(-toli);

    %  Run NM
    CA4_nm_submsn;
    %  Store data produced by NM
    nm_data.mean_Nevals(ix) = mean_Nevals;
    nm_data.bsl_reserr(ix) = bsl_reserr;

    Zmk = Amk;          %  Prepare Zmk for CA4_BesselMovie
    nofigs = true;        %  skip creating the gif
    CA4_BesselMovie_submsn;
    %  Store data produced by CA4_BesselMovie.m
    nm_data.diffeq_reserr(ix) = max(abs(real(test(:,end))/mP));
end

%  Run CA4_fzero.m for each value in 'tols'
for toli = tols(1):tols(end)
    %  i-th iteration
    ix = toli - tols(1) + 1;

    %  Set tol for CA4_fzero.m
    tolfz = 1 * 10^(-toli);

    %  Run fzero
    CA4_fzero_submsn;
    %  Store data produced by NM
    fzero_data.mean_Nevals(ix) = mean_Nevals;
    fzero_data.bsl_reserr(ix) = bsl_reserr;

    Zmk = Amk;          %  Prepare Zmk for CA4_BesselMovie
    nofigs = true;        %  skip creating the gif
    CA4_BesselMovie_submsn;

    %  Store data produced by CA4_BesselMovie.m
    fzero_data.diffeq_reserr(ix) = max(abs(real(test(:,end))/mP));
end

%  Create log-plot figure of error results for NM and fzero
figure(4000); clf

%  Set figure parameters
ax1 = gca;
title('Log-plot of Residual Errors vs. Convg. Tolerances of NM,
 fzero')
hold on;    grid on;

xlabel('log_{10}(convg. tol)')
ax1.XAxisLocation = 'top';  ax1.XDir = 'reverse';
ax1.XLim = [min(-tols-1) max(-tols+1)];
ax1.XTick = [ax1.XLim(1):ax1.XLim(end)];

ylabel('log_{10}(res err)')
ax1.YLim = [-18 0];
ax1.YTick = [ax1.YLim(1):ax1.YLim(end)];

%  Create legend using dummy varaibles
invis = [NaN NaN];
dum(1,1) = plot(invis,'.k');     lbl(1,1) = "rms(res err) of J_{m}
(x)";
dum(2,1) = plot(invis,'xk');     lbl(2,1) = "max(abs(res err)) of DE";
dum(3,1) = plot(invis,'--r');    lbl(3,1) = "NM";
dum(4,1) = plot(invis,'--b');    lbl(4,1) = "fzero";
legend(ax1,dum(:,1),lbl(:,1),'AutoUpdate','off','Location','best')

%  plot NM res errs
```

```
plot(-tols,log10(nm_data.bsl_reserr),'.r')
plot(-tols,log10(nm_data.diffeq_reserr),'xr')

%  plot fzero res errs
plot(-tols,log10(fzero_data.bsl_reserr),'.b')
plot(-tols,log10(fzero_data.diffeq_reserr),'xb')

%  Create figure of mean function evals. at each 'tol' for NM and
 fzero
figure(4001); clf
title('Mean # of Function Evals/iter. of NM, fzero')

%  Set figure parameters
ax2 = gca;
legend(ax2,'Location','northwest')
hold on;    grid on;

xlabel('log_{10}(convg. tol)')
ax2.XAxisLocation = 'bottom';   ax2.XDir = 'reverse';
ax2.XLim = [min(-tols-1) max(-tols+1)];
ax2.XTick = [ax2.XLim(1):ax2.XLim(end)];

ylabel('# of func. evals. per iter')
ax2.YLim = [0 nm_data.mean_Nevals(end)+1];
ax2.YTick = [ax2.YLim(1):ax2.YLim(end)];

plot(-tols,nm_data.mean_Nevals,'*r', ...
        'DisplayName','NM')
plot(-tols,fzero_data.mean_Nevals,'*b', ...
        'DisplayName', 'fzero')
```

Published with MATLAB® R2018a