Computing Assignment: Root Finding 2D contour

BMethod function file, Kai Sackville-Hii (feb 4, 2019)

```
function xB = BMethod(ff, a, b, tol)
%BMethod Implementation of the bisection method.
%Pre:
%   f = function to be approximated.
%   a, b = Initial guess/bounds.
%   tol = The tolorence condition.
%Post:
%   p = The approximated value.

%%%BEGIN:   this part does sqrt-specific bracketting

%  bracketting list
xlist = [a:0.1:b];
flist = ff(xlist);
sign_check = sign(flist);

%  check for any exact zeros!!
indX = find(sign_check==0);
if (isempty(indX)~=1)
 x_sqrt = xlist(indX);
    disp(['square root = ' num2str(x_sqrt) ' & err = '
 num2str(flist(indX))])
 return
end

%  find sign-change interval
indX = find(diff(sign_check),1);
if (isempty(indX)==1)
 disp('no sign change')
 return
end

%%%END:   this part does sqrt-specific bracketting

%  0)  set sign-change interval
xL = xlist(indX  );
% xL = a;
fL = ff(xL);
xR = xlist(indX+1);
% xR = b;
fR = ff(xR);
Nevals = 2;

%  1)  compute first midpoint
% figure(101);  clf;  hold on;  grid on
check = (xR-xL)/2;
```

```
% plot(Nevals,log10(abs(check)),'rx')

% fprintf('\t %d \t %16.15f \t %+6.5e \t %+16.15f \t %+6.5e \t %+6.5e
 \n', ...
%   [Nevals, xL, fL, xR, fR, check])

xB  = xL + check;
Nevals_arr = [];

%  root-finding loop
while (abs(check)>tol)
 %  2)  function evaluation
 fB = ff(xB);
    Nevals_arr(Nevals) = Nevals + 1;
 Nevals = Nevals + 1;

 %  3)  decision
 if (fB==0)
  xL = xB;  fL = fB;
  xR = xB;  fR = fB;
 else
  if (fL*fB>0)
   xL = xB;  fL = fB;
  else
   xR = xB;  fR = fB;
  end
 end

%  fprintf('\t %d \t %16.15f \t %+6.5e \t %+16.15f \t %+6.5e \t %+6.5e
 \n', ...
%   [Nevals, xL, fL, xR, fR, check])

 %  4)  prepare next iteration
 check = (xR-xL)/2;
%  plot(Nevals,log10(abs(check)),'kx')

 xB  = xL + check;
%     disp(xB)
end

% disp(max(Nevals_arr));
end
```

Not enough input arguments.

Error in BMethod (line 16)
xlist = [a:0.1:b];

Published with MATLAB® R2018b