```
function c=pseout(Q,w,epsilons,xrange,yrange,hx,hy)

% PSEOUT(Q,w,epsilon,xrange,yrange,hx,hy) estimates the spectrum and,
% for several epsilons, the boundaries of the epsilon-pseudospectra of
% a square matrix polynomial of the form
%
%                m         m-1
%   P(z)  =  A z  +  A    z    + ... + A z  + A
%             m       m-1               1      0
%
% with a nonsingular leading coefficient  A .
%                                          m
%
% The algorithm is based on the construction of exclusion discs
% centered only at exterior points.
%
% (Note that the command tic-toc prints the execution time.)
%
%  INPUT:  Q = {A0,A1,...,Am} array of the matrix-coefficients,
%          w = [w0,w1,...,wm] array of the corresponding weights,
%          epsilons = [eps1,eps2,...,epsN] array of epsilons,
%          xrange = [left bound on x-axis, right bound on x-axis],
%          yrange = [lower bound on y-axis, upper bound on y-axis],
%          hx = length of the grid for the x-axis,
%          hy = length of the grid for the y-axis.
%
%  OUTPUT: c = contour plot points
%          (boundaries of the pseudospectra and eigenvalues of P(z)).
%
% Authors: Stavros Fatouros and Panayiotis Psarrakos,
%          Department of Mathematics,
%          National Technical University of Athens.
%
% Research supported by a grant of the EPEAEK project PYTHAGORAS II.
% The project is co-funded by the European Social Fund (75%) and
% Greek National Resources (25%).

tic

% detaching the epsilons (e1>=e2>=...>=eN)
epsilons=detach(epsilons);
N=length(epsilons);

% Setting sizes and the degree of the matrix polynomial P(z)
[t,n]=size(Q);       % here n=m+1=degree+1
degree=n-1;          % the degree of the matrix polynomial
[t,d]=size(Q{1});    % here d is the order of matrix-coefficients

% Reversing the array of weights
rw=[];

for j=1:n
    rw(1,j)=w(1,n-j+1);
end  % for

% Constructing the grid
n1=round((xrange(2)-xrange(1))/hx);
n2=round((yrange(2)-yrange(1))/hy);

% Setting initial values on the grid
Z=zeros(n1+1,n2+1);
Iterations=0;
expoints=0;

% Computing factorials
```

```
par=1;

for j=1:degree
    par=par*j;
    p(j)=par;
end  % for

% Main part of the program
for x=xrange(1):hx:xrange(2)
    I=round(((x-xrange(1))/hx)+1);

    for y=yrange(1):hy:yrange(2)
        J=round(((y-yrange(1))/hy)+1);
        mu=x+(y*i);
        absmu=abs(mu);

        if Z(I,J)==0
            r1=0;
            B0=Q{1};
            wm0=w(1);

            for j=1:degree
                B0=B0+(mu^j)*Q{j+1};
                wm0=wm0+(absmu^j)*w(j+1);
            end  % for

            s=min(svd(B0));
            Z(I,J)=s/wm0;
            Iterations=Iterations+1;

            if Z(I,J)>epsilons(1)
                expoints=expoints+1;

                for k=1:n
                    Lamda0(k)=absmu^(k-1);
                end  % for

                a=deriv(Q,absmu,p,degree);
                r1=wderiv1(rw,epsilons(1),s,wm0,degree,Lamda0);
                r1=r1-absmu;
                L=0;
                R=1;

                for k=1:11
                    g=(L+R)/2;
                    rg=r1*g;
                    lambda=absmu+rg;
                    a(n)=w(1);

                    for j=1:degree
                        a(n)=a(n)+w(j+1)*(lambda^j);
                    end  % for

                    a(n)=-s+epsilons(1)*a(n);
                    star=a(n);

                    for j=1:degree
                        star=star+a(j)*(rg^(n-j));
                    end  % for

                    if star>0
                        R=g;
                    else
                        L=g;
                    end  % if
```

```
                end  % for

                r=(rg-r1*0.01)/1.4;
                q1=round(r/hx);
                q2=round(r/hy);

                for k1=0:q1

                    for k2=0:q2
                        Z(I+k1,J+k2)=Z(I,J);
                        Y(J+k2)=yrange(1)+(J+k2-1)*hy;
                    end  % for

                    X(I+k1)=xrange(1)+(I+k1-1)*hx;
                end  % for

            end  % if Z(I,J)>epsilons(1)

        end  % if Z(I,J)=0

        Y(J)=yrange(1)+(J-1)*hy;
    end  % for

    X(I)=xrange(1)+(I-1)*hx;
end  % for

% Giving the number of iterations and plotting the figure
number_of_total_grid_points=(n1+1)*(n2+1)
number_of_iterations=Iterations
number_of_iterations_at_exterior_points=expoints

X = X(2:end-5);
Y = Y(2:end-5);
Z = Z(2:end-5, 2:end-5);
mesh(X, Y, Z')

hold on

for j=1:N
    ZZ=Z-epsilons(j)*ones(size(Z));
    contour(X,Y,ZZ',[0 0],'b')
end  % for

toc

axis([xrange(1),xrange(2),yrange(1),yrange(2)])
xlabel('Real  Axis')
ylabel('Imaginary  Axis')

% Computing the companion matrix and the eigenvalues
C0=[];

for j=1:degree
    C0=[C0 -Q{j}];
end  % for

[l1,l2]=size(C0);
C=[zeros(l2-l1,l1) eye(l2-l1); inv(Q{n})*C0];
eigen=eig(C)
plot(real(eigen),imag(eigen),'r*')
hold off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a=deriv(Q,mu,p,degree);
```