

```

function [x,Lce>ErrorQ,rvalues,U]=thqrbp1(F1,F2,Ic,Ntot,LceFlag,Nskip,OrthFlag,p);
% COMPUTATION OF THE FIRST p LCE'S using HQRBP1 using the method.
%
% This method is efficient when n is close to p. If p is small, hqrbp2 is
% more efficient than hqrbp1 (for this case use hqrbp2 instead of hqrbp1).
% Here n denotes the number of states of the system and p the number of LCEs sought.
%
% thqrbp1 Computes the trajectory and the Lyapunov Characteristic
% Exponents for discrete systems via The Householder QR Based Method.
%
% Information on the method can be found in: "A note on the computation of the
% largest p LCEs of discrete dynamical systems" by Firdaus E. Udwardia,
% Hubertus F. von Bremen and Wlodek Proskuroswki, to appear in Applied Mathematics
% and
% Computation.
%
% Additional information on computing all LCEs, see "An efficient QR based method for
% the
% computation of Lyapunov Exponents", by Hubertus F. von Bremen,
% Firdaus E. Udwardia and Wlodek Proskuroswki, Physica D, vol 101 pp. 1-16, 1997.
%
% Program written by H. von Bremen 4/17/2000
%
%*****
%
% [x,Lce>ErrorQ]=thqrbp1('xfile','tanmfile',Ic,Ntot,LceFlag,Nskip,OrthFlag,p)
% computes the trajectory of a discrete dynamical system which is
% specified in the M-file xfile.m . It also computes the first s Lyapunov
% characteristic exponents (Lce's), provided the tangent maps are
% specified in the M-file tanmfile.m. The trajectory is computed for
% up to Ntot iterations. The LCE's are computed starting at the Nskip+1
% iteration of the trajectory, and are computed for Ntot-Nskip iterations.

%
% INPUT:
% xfile      string containing the user-supplied iteration scheme for
%             the trajectory.
%             CALL: xmap=fun(x,iter) where fun = 'xfile'
%             x      - state vector
%             iter   - iteration number
%             xmap   - returned next iteration
%                   {x(iter+1)=xmap=fun(x(iter),iter)}
% tanmfile   string containing the user-supplied file that defines the
%             tangent map to the trajectory.
%             CALL: tanmap=fun(x,iter) where fun = 'tanmfile'
%             x      - state vector

```

```

%          iter    - iteration number
%          tanmap- returned next iteration
%          {Tangent_Map(iter+1)=tanmap=fun(x(iter),iter)}
% Ic:       Initial conditions for the Trajectory, given as a column
%          vector
% Ntot:     Number of iterations (time steps) to be used to calculate
%          the trajectory of the system.
% LceFlag:  flag used to determine if Lce's are computed.
%          -If LceFlag = 1, then Lce's are computed.
%          -If LceFlag = 0, then LCE's are not computed and only
%          the trajectory is computed.
% Nskip:    Number of iterations to be skipped before the LCE's are
%          to be computed.
% OrthFlag: flag used to determine if error in orthogonality is to
%          be computed, and if computed what type.
%          -If OrthFlag = 0, then the error in orthogonality is not
%          computed. (Recommended value)
%          -If OrthFlag = 1, then the determinant error in orthogonality is
%          is computed. Error in Orthogonality = abs(1-abs(det(Q)))
%          When s is not equal to n, OrthFlag is then set to 2.
%          -If OrthFlag = 2, then the Two-Norm error in orthogonality is
%          is computed. Error in Orthogonality = Norm(Q*Q-I)
% NOTE: computing the error in orthogonality will increase the computation time.
%
% p        Number of first Lces' to be computed.
%
% OUTPUT:
%   x      Trajectory. Given in matrix form, the k-th
%          column are the trajectory coordinates for the
%          k-th iteration (the first column is Ic ). size(x)=[n,Ntot+1].
%   Lce    Computed Lyapunov Exponents. Given in matrix
%          form, the k-th column are the LCE's for the k-th
%          iteration of LCE's. size(Lce)=[n,Ntot-Nskip].
%   ErrorQ Error in othogonality
%          -If OrthFlag = 1, ErrorQ = abs(1-abs(det(Q)))
%          -If OrthFlag = 2, ErrorQ = Norm(Q*Q-I)
%          Given in vector form, the k-th entry is the error
%          in orthogonality of the k-th Lce iteration.
%          size(NormQ)=[Ntot-Nskip,1].
%
% **EXAMPLE:
% -----
% Sample Call:
% Ic=[0.2 0.3]'; Ntot= 10000; LceFlag=1; Nskip=1000; OrthFlag=1;
% [x,Lce,ErrorQ]=thqrbp1('cplgst','tglgst',Ic,Ntot,LceFlag,Nskip,OrthFlag,s);
%

```

```

% -----
%
% -----
% Sample for xfile='cplgst'; coupled logistic maps (stored in m-file cplgst.m)
% function xmap=cplgst(x,iter)
% d=.2; r=3.6;
% xm(1)=d*r*x(1)*(1-x(1)) + (1-d)*r*x(2)*(1-x(2));
% xm(2)=(1-d)*r*x(1)*(1-x(1)) + d*r*x(2)*(1-x(2));
% xmap=[xm(1) xm(2)];
% -----
% Sample for tanmfile='tglgst'; Jacobian for coupled logistic maps (stored in m-file
tglgst.m)
% function [T]=tglgst(x)
% d=.2; r=3.6;
% T= [ d*r*(1-2*x(1))    (1-d)*r*(1-2*x(2))
%      (1-d)*r*(1-2*x(1))    d*r*(1-2*x(2)) ];
% -----
% -----
% Sample to plot output:
% To plot the k-th LCE for each iteration: plot(Lce(k,:))
% To plot the k-tk coordinate of the trajectory: plot(x(k,:))
% To plot the error in orthogonality: plot(ErrorQ)
% -----
%
% NOTE: As an additional example, the n-dimensional generalization of coupled logistic
maps
% is provided as follows.
% -----
% Sample for xfile='cplgstn'; the generalized coupled logistic maps (stored in m-file
cplgstn.m)
% function xmap=cplgstn(x,iter)
% This is an nd-map generalization of the 2d-logistic map . . .
% d=2.01;r=1.8;
% n=length(x);
% y=(x-x.^2);
%          %      y=[x(1)*(1-x(1)) x(2)*(1-x(2)) x(3)*(1-x(3))...
%          %      x(4)*(1-x(4)) x(5)*(1-x(5)) x(6)*(1-x(6))...
%          %      x(7)*(1-x(7)) x(8)*(1-x(8)) x(9)*(1-x(9))...
%          %      x(10)*(1-x(10))];
% A=d*r*diag(ones(n,1))+(-1+d/2)*r*diag(ones(n-1,1),1)+...
% (-1+d/2)*r*diag(ones(n-1,1),-1);
% xmap=A*y;
% -----
% Sample for tanmfile='tglgstn'; Jacobian for coupled logistic maps (stored in m-file
tglgstn.m)
% function [T]=tglgstn(x)

```

```

% % this ia an nd-map generalization of the 2d-logistic map . . .
% d=2.01;r=1.8;
% n=length(x);
% y=(1-2*x)';
% A=d*r*diag(ones(n,1))+(-1+d/2)*r*diag(ones(n-1,1),1)+...
% (-1+d/2)*r*diag(ones(n-1,1),-1);
% for i=1:n
% B(i,:)=A(i,:).*y;
% end;
% T=B;
% -----

```

```

% INITIALIZATION PART

```

```

m=Ntot-Nskip;s=p;
n=max(size(Ic));
r=ones(1,s);
q=eye(n);sl=zeros(s,1);Lce=zeros(s,m);rvalues=zeros(s,m);
gama=zeros(s,1);
if OrthFlag==1|OrthFlag==2
    ErrorQ=zeros(m,1);
end % (if)
x=zeros(n,Ntot+1);

```

```

temp=2*n^2*p+(5/6)*p^3+5.5*n*p-4.5*n*p^2-2*n^2;
if (temp>0)
    disp('For the number of LCEs you are computing it may take')
    disp('fewer flops if you use HQRBP2 instead of HQRBP1')
end

```

```

if (OrthFlag==1) & (s~=n)
    disp('Not a square orthogonal matrix (s not eq. n) ')
    disp('OrthFlag will be set to 2')
    OrthFlag=2;
end
% END INITIALIZATION

```

```

x(:,1)=Ic;
for j=2:Nskip+1
    Ic=feval(F1,Ic);
    x(:,j)=Ic;
end % (j)

```

```

if LceFlag==0
    for j=1:m

```

```

    Ic=feval(F1,Ic);
    x(:,j+Nskip+1)=Ic;
    end % (j)
else

    A=feval(F2,Ic);
    if OrthFlag==0
        a=A;
    end;
    N2=Nskip+1;
    for i=1:m
        if OrthFlag~=0
            a=A*q;
            Ic=feval(F1,Ic);
            x(:,i+N2)=Ic;
            A=feval(F2,Ic);
            B=eye(n);
        elseif OrthFlag == 0
            Ic=feval(F1,Ic);
            x(:,i+N2)=Ic;
            B=feval(F2,Ic);
        end % (if OrthFlag)
    % Computation of the Factorization
        if s==n
            w=n-1;
        else
            w=s;
        end
        for k=1:w
    % computation of the reflector
    % *****
            if a(k,k)<0
                b=-1;
            else
                b=1;
            end
            sig=sqrt(a(k:n,k)*a(k:n,k));
            gama(k)=sig*(sig+abs(a(k,k)));
            r(k)=-b*sig;
            a(k,k)=a(k,k)-r(k);
    % end computation of the reflector
            p=k+1;
            a(p:n,p:s)=a(p:n,p:s)-repmat((a(k:n,k)*a(k:n,p:s))/gama(k),[n-
p+1,1]).*repmat(a(p:n,k),[1,s-p+1]);
    % Computation of B*Q

```

```

if k~=w
    tem=a(k:n,k)/sqrt(gama(k));
    B(:,k:n)=B(:,k:n)-repmat(tem',[n,1]).*repmat((B(:,k:n)*tem),[1,n-k+1]);

else % k==w
    for j=1:n
        b=B(j,k:n)*a(k:n,k)/gama(k);
        B(j,k:s)=B(j,k:s)-a(k:s,k)'*b;
    end
end % if k
end % end for k
if s==n
    r(n)=a(n,n);
end
% *****
% End Computation of the factorization
if OrthFlag~=0
    q=B(:,1:s);
    if OrthFlag==1
        ErrorQ(i)=abs(1-abs(det(q)));
    elseif OrthFlag==2
        ErrorQ(i)=norm(q'*q-eye(s));
    end % (if)
elseif OrthFlag == 0 % (if OrthFlag)
    a=B(:,1:s);
end % (if OrthFlag)
sl=sl+log(abs(r'));
rvalues(:,i)=r';
Lce(:,i)=sl/i;
end % (i)
end % (else)

```