function [x,U,D] = thornton(xin,Phi,Uin,Din,Gin,Q)  
%  
% function [x,U,D] = thornton(xin,Phi,Uin,Din,Gin,Q)  
%  
% M. S. Grewal, L. R. Weill and A. P. Andrews  
% Global Positioning Systems, Inertial Navigation and Integration  
% John Wiley & Sons, 2000.  
%  
% Catherine Thornton's modified weighted Gram-Schmidt  
% orthogonalization method for the predictor update of  
% the U-D factors of the covariance matrix  
% of estimation uncertainty in Kalman filtering  
%  
% INPUTS(with dimensions)  
% xin(n,1) corrected estimate of state vector  
% Phi(n,n) state transition matrix  
% Uin(n,n) unit upper triangular factor (U) of the modified Cholesky  
% factors (U-D factors) of the covariance matrix of  
% corrected state estimation uncertainty (P+)   
% Din(n,n) diagonal factor (D) of the U-D factors of the covariance  
% matrix of corrected estimation uncertainty (P+)  
% Gin(n,r) process noise distribution matrix (modified, if necessary to  
% make the associated process noise covariance diagonal)  
% Q(r,r) diagonal covariance matrix of process noise  
% in the stochastic system model  
% OUTPUTS:  
% x(n,1) predicted estimate of state vector  
% U(n,n) unit upper triangular factor (U) of the modified Cholesky  
% factors (U-D factors) of the covariance matrix of  
% predicted state estimation uncertainty (P-)   
% D(n,n) diagonal factor (D) of the U-D factors of the covariance  
% matrix of predicted estimation uncertainty (P-)  
%  
x = Phi\*xin; % state update  
[n,r] = size(Gin); % get dimensions of state(n) and process noise (r)  
G = Gin; % move to internal array for destructive updates  
U = eye(n); % initialize lower triangular part of U  
PhiU = Phi\*Uin; % rows of [PhiU,G] are to be orthononalized  
for i=n:-1:1,  
 sigma = 0;  
 for j=1:n,  
 sigma = sigma + PhiU(i,j)^2 \*Din(j,j);  
 if (j <= r)  
 sigma = sigma + G(i,j)^2 \*Q(j,j);  
 end;  
 end;  
 D(i,i) = sigma;  
 for j=1:i-1,  
 sigma = 0;  
 for k=1:n  
 sigma = sigma + PhiU(i,k)\*Din(k,k)\*PhiU(j,k);  
 end;  
 %  
 for k=1:r,  
 sigma = sigma + G(i,k)\*Q(k,k)\*G(j,k);  
 end;  
 U(j,i) = sigma/D(i,i);  
 for k=1:n,  
 PhiU(j,k) = PhiU(j,k) - U(j,i)\*PhiU(i,k);  
 end;  
 %  
 for k=1:r,  
 G(j,k) = G(j,k) - U(j,i)\*G(i,k);  
 end;  
 end;  
end;