```
function [x, P] = nonLinKFprediction(x, P, f, Q, type)
%NONLINKFPREDICTION calculates mean and covariance of predicted state
%   density using a non-linear Gaussian model.
%
%Input:
%   x           [n x 1] Prior mean
%   P           [n x n] Prior covariance
%   f           Motion model function handle
%               [fx,Fx]=f(x)
%               Takes as input x (state),
%               Returns fx and Fx, motion model and Jacobian evaluated
 at x
%               All other model parameters, such as sample time T,
%               must be included in the function
%   Q           [n x n] Process noise covariance
%   type        String that specifies the type of non-linear filter
%
%Output:
%   x           [n x 1] predicted state mean
%   P           [n x n] predicted state covariance
%

    [fx,Fx]=f(x);
    n=length(x);
    switch type
        case 'EKF'
            x=fx;
            P=Fx*P*Fx'+Q;


        case 'UKF'
            [SP,W] = sigmaPoints(x, P, 'UKF');

            for i=0:2*n
                [fx,Fx]=f(SP(:,i+1));
                x(:,i+1)=fx*W(:,i+1);
            end
            x=sum(x,2);

            for i=0:2*n
                [fx,Fx]=f(SP(:,i+1));
                P(:,:,i+1)=(fx-x)*(fx-x)'*W(:,i+1);
            end
            P=sum(P,3)+Q;

            % Make sure the covariance matrix is semi-definite
            if min(eig(P))<=0
                [v,e] = eig(P, 'vector');
                e(e<0) = 1e-4;
                P = v*diag(e)/v;
            end
        case 'CKF'
            [SP,W] = sigmaPoints(x, P, 'CKF');

            for i=1:2*n
                [fx,Fx]=f(SP(:,i));
                x(:,i)=fx*W(:,i);
            end
            x=sum(x,2);

            for i=1:2*n
                [fx,Fx]=f(SP(:,i));
                P(:,:,i)=(fx-x)*(fx-x)'*W(:,i);
            end
            P=sum(P,3)+Q;
        otherwise
            error('Incorrect type of non-linear Kalman filter')
    end

end
```

Published with MATLAB® R2020a