```
function [xf, Pf, xp, Pp] = nonLinearKalmanFilter(Y, x_0, P_0, f, Q,
 h, R, type)

%NONLINEARKALMANFILTER Filters measurement sequence Y using a
% non-linear Kalman filter.
%
%Input:
%   Y           [m x N] Measurement sequence for times 1,...,N
%   x_0         [n x 1] Prior mean for time 0
%   P_0         [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
%   Q           [n x n] Process noise covariance
%   h                   Measurement model function handle
%                       [hx,Hx]=h(x,T)
%                       Takes as input x (state),
%                       Returns hx and Hx, measurement model and
 Jacobian evaluated at x
%   R           [m x m] Measurement noise covariance
%
%Output:
%   xf          [n x N]     Filtered estimates for times 1,...,N
%   Pf          [n x n x N] Filter error convariance
%   xp          [n x N]     Predicted estimates for times 1,...,N
%   Pp          [n x n x N] Filter error convariance
%

% Your code here. If you have good code for the Kalman filter, you
 should re-use it here as
% much as possible.
```

Parameters

```
N = size(Y,2); n = length(x_0); m = size(Y,1);
```

Data allocation

```
xf = zeros(n,N+1); Pf = zeros(n,n,N+1);
xp = zeros(n,N); Pp = zeros(n,n,N);

%initial
xf(:,1) = x_0; Pf(:,:,1) = P_0;

%kalman
for i=2:N+1
    [xp(:,i-1), Pp(:,:,i-1)] = nonLinKFprediction(xf(:,i-1),
 Pf(:,:,i-1), f, Q, type);
    [xf(:,i), Pf(:,:,i)] = nonLinKFupdate(xp(:,i-1), Pp(:,:,i-1),
 Y(:,i-1), h, R, type);
```

```
end

%output
xf = xf(:,2:end); Pf = Pf(:,:,2:end);

end
```

Published with MATLAB® R2020a