**Midterm 2**

Ryan Dreifuerst

**P4-1)**

**Code**

function [fxg] = p4\_1(xg)

x1 = xg(1);

x2 = xg(2);

top = x1 + x2 + x1\*x2+5;

bot = x1^2+2\*x2-x2^2-2;

fxg = [top; bot];

normal = norm(fxg, 2);

end

function [xg\_new] = gradient\_descent(xg)

x1 = xg(1);

x2 = xg(2);

V = [1 + x2, 1+x1; 2\*x1, 2-2\*x2];

xg\_new = xg - inv(V)\*p4\_1(xg);

end

% Addition to PS4 #1

% generate all possible initial values

clear all;

close all;

x1 = -10:10;

x2 = -10:10;

% total possible convergences

final = zeros(21, 21, 2);

uniquevalues = [];

% run each possible input and determine if it converges

for i=-10:10

for j=-10:10

xg = [i; j];

% ensure convergence

for k=1:20

xg = P4\_1\_grad\_desc(xg);

end

cost = norm([xg(1) + xg(2) + xg(1)\*xg(2)+5; xg(1)^2+2\*xg(2)-xg(2)^2-2], 2);

if cost < 1e-5

final(i+11, j+11, :) = round(xg, 4); % for determining uniqueness use rounding

if ~ ismember(final(i+11, j+11, :), uniquevalues)

uniquevalues = [uniquevalues, final(i+11, j+11, :)];

end

end

% note that [0; 0] is not a convergent spot so any of these are not

% convergent

end

end

% determine uniqueness

% plot which values are convergent

uniquevalues = squeeze(uniquevalues);

figure

plot(uniquevalues(1, 1), uniquevalues(1, 2) , 'rx', 'MarkerSize',12);

hold on

title("Initial Guesses and Convergent Points")

xlim([-10, 10])

ylim([-10, 10])

xlabel("X1")

ylabel("X2")

plot(uniquevalues(2, 1), uniquevalues(2, 2), 'ro', 'MarkerSize',12);

hold on

nonconv = 0;

for i=-10:10

for j=-10:10

if squeeze(final(i+11, j+11, :)) == uniquevalues(1, :)'

plot(i, j, 'bx')

hold on

elseif squeeze(final(i+11, j+11, :)) == uniquevalues(2, :)'

plot(i, j, 'bo')

hold on

else

nonconv = nonconv + 1;

end

end

end

nonconv

nonconv =

13



**P4-3)**

**Code**

function [H] = H\_4\_3(xg, t)

x1 = xg(1);

x2 = xg(2);

x3 = xg(3);

H = zeros(3, 11);

for j=1:11

H(:, j) = [cos(x2\*t(j)+x3), -x1\*t(j)\*sin(x2\*t(j)+x3), -x1\*sin(x2\*t(j)+x3)];

end

function [cost] = P43cost(zhist, thist, xg, R)

cost = sum(abs((zhist-xg(1)\*cos(xg(2)\*thist+xg(3)))'\*inv(R)\*(zhist-xg(1)\*cos(xg(2)\*thist+xg(3))))^(1/2));

end

function [hxg] = littleh(xg,thist)

hxg = zeros(11, 1);

for j=1:11

hxg(j) = xg(1)\*cos(xg(2)\*thist(j) + xg(3));

end

end

%% Initial setup

% Model

% z\_j = x1\*cost(x2\*t\_j+x3)+w\_j

clear all

nz = 11;

nx = 3;

R = zeros(nz, nz);

for i=1:nz

for j=1:nz

if abs(i-j) == 1

R(i, j) = 0.5;

elseif i==j

R(i, j) = 1;

end

end

end

thist =[0; 0.1000; 0.2000; 0.3000; 0.4000; 0.5000;

0.6000; 0.7000; 0.8000; 0.9000; 1.0000];

% old zhist

% zhist = [7.7969; 1.4177; -3.0970; -7.6810; -9.8749; -6.1828;

% -0.8212; 4.5074; 8.2259; 9.5369; 6.2827];

% new zhist

zhist = [0.380522980099649

-0.937893887500619

-2.568861909017112

-5.780043794471402

-5.484192626331365

-2.987309363580615

-1.065589854303113

1.286281960892803

3.687753634356679

6.673091483609936

7.148528020590929];

%% Apply the iterations

% start somewhere

xg = [9, 6, 6];

xgbest = xg;

bestcost = P43cost(zhist, thist, xg, R);

nx = 3;

nk = 11;

alpha = 1;

initialcost = P43cost(zhist, thist, xg, R)

for j=0:20

for l = 0:20

xg = [j, 6, l]';

jold = P43cost(zhist, thist, xg, R);

for i=1:100

H = inv(R)\*(H\_4\_3(xg, thist))';

% Use QR factorization

[Q, R1] = qr(H);

R1 = R1(1:nx, :);

Q1 = Q(:, 1:nx);

lh = littleh(xg, thist);

delta = (inv(R1)\*Q1'\*(zhist-lh));

jnew = P43cost(zhist, thist, xg+alpha\*delta, R);

while jnew >= jold +1e-8

alpha = alpha/2.0;

jnew = P43cost(zhist, thist, xg+alpha\*delta, R);

end

xg = xg+alpha\*delta;

end

cost = P43cost(zhist, thist, xg, R);

if cost < bestcost

xgbest = xg;

bestcost = cost;

end

end

end

plot(thist, zhist, thist, littleh(xg, thist))

title("Zhist vs Estimator Estimates")

legend("Zhist", "h(xg)")

xlabel("Thist")

H = (H\_4\_3(xgbest, thist))';

Pxx = inv(H'\*inv(R)\*H)

xgbest

bestcost

----------------------------------------------------------------

Pxx =

0.313100894131948 -0.008081619406640 0.001671685991253

-0.008081619406640 0.088978473971459 -0.035296282686897

0.001671685991253 -0.035296282686897 0.023432882909826

xgbest =

5.755819380843929

5.097187868494098

1.380984869874769

bestcost =

3.038427726169493



**P4-4)**

**Code**

function [initial\_cond] = P4\_4\_init(rhoa,rhob, dl, thist)

% assumes starting point for radar a is rad\_a\_x

rad\_a\_x = 4.1e5;

rad\_b\_x = rad\_a\_x + dl;

% General Method

theta = acos((-rhob.^2+rhoa.^2+dl^2)./(2.\*rhoa\*dl));

y = real(rhoa.\*sin(theta));

x = real(rad\_a\_x + rhoa.\*cos(theta));

%% select x(2), x(11), y(2), y(11) for parabola creation

x1 = x(2);

x2 = x(end/2+1);

h1 = y(2);

h2 = y(end/2+1);

a = -(h1-h2)/(x1 - x2)^2;

% the plot should be similar/fit on those points

% with parabola -a\*(x-x2)^2+h2

figure

plot(x, y, x, -a.\*(x-x2).^2 + h2);

title("X vs Y initial estimate")

legend("X,Y from two points", "X, Y parabola function");

%% initial condition when t = 0

% x is just initial position + initial velocity \* time

delta\_x = x2 - x1;

delta\_t = thist(11) - thist(2);

Vx = delta\_x/delta\_t;

X\_0 = x1 - Vx\*thist(2);

Y\_0 = -a\*(X\_0 - x2)^2+h2;

Vy = (h2 + 1/2\*9.81\*thist(11).^2-Y\_0)/thist(11);

if Y\_0 < 0

Y\_0 = 0;

end

initial\_cond = [X\_0, Y\_0, Vx, Vy]';

end

function [cost, x, y, ra\_hat, rb\_hat, ta\_hat, tb\_hat] = Cost4\_4(xg, thist, ra, rb, ta, tb, nk)

Rs = diag([10^2\*ones(1, nk), 30^2\*ones(1, nk)]);

tb\_hat = 0;

ta\_hat = 0;

x = xg(1) + xg(3).\*thist;

y = xg(2) + xg(4).\*thist - 4.9\*thist.^2;

ra\_hat = sqrt((4.1e5 - x).^2 + y.^2);

rb\_hat = sqrt((4.4e5 - x).^2 + y.^2);

if isscalar(ta)

r\_hat = [ra\_hat; rb\_hat];

r = [ra; rb];

else

ta\_hat = atan2(y, -x + 4.1e5);

tb\_hat = atan2(y, -x + 4.4e5);

Rs = diag([10^2\*ones(1, nk), 30^2\*ones(1, nk), 0.01^2\*ones(1, nk), 0.03^2\*ones(1, nk)]);

r\_hat = [ra\_hat; rb\_hat; ta\_hat; tb\_hat];

r = [ra; rb; ta; tb];

end

cost = sqrt((r-r\_hat)'\*inv(Rs)\*(r-r\_hat));

end

function [subh, h, H, subH] = H\_4\_4(xg, t, nk, nx)

X\_0 = xg(1);

Y\_0 = xg(2);

Vx = xg(3);

Vy = xg(4);

h = zeros(nk, nx);

H = zeros(nk, 4, nx);

subH = zeros(nk\*nx, 4);

subh = zeros(nk\*nx, 1);

for j=1:nk

delta\_y1 = (4.1e5 - X\_0 - t(j)\*Vx);

delta\_y2 = (Y\_0 + t(j)\*Vy - 4.9\*t(j)^2);

delta\_y3 = (4.4e5 - X\_0 - t(j)\*Vx);

h(j, 1:2) = [sqrt(delta\_y1^2 + delta\_y2^2); sqrt(delta\_y3^2 + delta\_y2^2)];

% order is time (j), dx\_i, h\_k i=1,2,3,4; k=1,2

H(j, 1, 1:2) = [-delta\_y1/h(j, 1); -delta\_y3/h(j, 2)];

H(j, 2, 1:2) = [delta\_y2/h(j, 1); delta\_y2/h(j, 2)];

H(j, 3, 1:2) = [-t(j)\*delta\_y1/h(j, 1); -t(j)\*delta\_y3/h(j, 2)];

H(j, 4, 1:2) = [t(j)\*delta\_y2/h(j, 1); t(j)\*delta\_y2/h(j, 2)];

% extension part

% see notes for the equations

if nx > 2

% there are now more h equations so k=1,2,3,4

h(j, 3:4) = [atan2(delta\_y2, delta\_y1); atan2(delta\_y2, delta\_y3)];

H(j, 1, 3:4) = [-delta\_y2/(h(j, 1))^2\*(-1); -delta\_y2/(h(j, 2))^2\*(-1)];

H(j, 2, 3:4) = [-delta\_y1/(h(j, 1))^2; -delta\_y3/(h(j, 2))^2];

H(j, 3, 3:4) = [delta\_y2/(h(j, 1))^2\*t(j); delta\_y2/(h(j, 2))^2\*t(j)];

H(j, 4, 3:4) = [delta\_y1/(h(j, 1))^2\*t(j); delta\_y3/(h(j, 2))^2\*t(j)];

end

end

subH = [squeeze(H(:, :, 1)); squeeze(H(:, :, 2))];

subh = [h(:, 1); h(:, 2)];

if nx > 2

subH = [squeeze(H(:, :, 1)); squeeze(H(:, :, 2)); squeeze(H(:, :, 3)); squeeze(H(:, :, 4))];

subh = [h(:, 1); h(:, 2); h(:, 3); h(:, 4)];

end

end

% Complete Problem 4-4

%% Initial setup

% model is z = h(x) + w

% Use bearing in the cost for comparison, but do not add information yet

clear all;

close all;

nk = 28;

nz = 2;

alpha = 0.2;

%

% thist = [5;15;25;35;45;55;65;75;85;95;105;115];

% rhoahist = [352862.778204296;340399.972183495;328375.512355275;316754.551972577;305482.784782397;294500.455434987;283702.165996909;273085.433533899;262551.991714750;252050.375922058;241496.208747970;230858.792975043];

% rhobhist = [382864.360971427;370335.998477794;358259.255367332;346495.952598628;335082.681393646;323823.527400930;312866.241273828;301958.579383796;291075.842630371;280329.997342317;269465.468386008;258410.947432348];

load('radarmeasdata\_missile\_new.mat');

z = [rhoahist; rhobhist];

xg = P4\_4\_init(rhoahist, rhobhist, 3e4, thist);

[initial\_cost, x, y, ra\_hat, rb\_hat] = Cost4\_4(xg, thist, rhoahist, rhobhist, 1, 1, nk);

%% Iterate to find best xg

for i=1:50

[subh, ~ , ~, subH] = H\_4\_4(xg, thist, nk, nz); % H is 12x4x2

% initial cost

if i == 1

Cost4\_4(xg, thist, rhoahist, rhobhist, 1, 1, nk);

end

%% Update guess

xg = xg + alpha \* inv(subH'\*subH)\*subH'\*[z-subh];

end

%% Results

[cost, x, y, ra\_hat, rb\_hat] = Cost4\_4(xg, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

figure

plot(thist, rhoahist, thist, ra\_hat);

title("Ra Comparison");

legend("Measured", "Predicted");

figure

plot(thist, rhobhist, thist, rb\_hat);

title("Rb Comparison");

legend("Measured", "Predicted");

R = diag([10^2\*ones(1,nk), 30^2\*ones(1, nk)]);

Pxx = inv(subH'\*inv(R)\*subH)

cost

xg

%% Add bearing measurement

disp("Adding bearing information and recalculating")

nz = 4; % four usable measurements

xg = P4\_4\_init(rhoahist, rhobhist, 3e4, thist);

z = [rhoahist; rhobhist; thetaahist; thetabhist];

[initial\_cost, ~, ~, ~, ~, ~, ~] = Cost4\_4(xg, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

%% Iterate to find best xg

for i=1:50

[subh, ~ , ~, subH] = H\_4\_4(xg, thist, nk, nz); % H is 12x4x2

% initial cost

if i == 1

Cost4\_4(xg, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

end

%% Update guess

xg = xg + alpha \* inv(subH'\*subH)\*subH'\*(z-subh);

end

[cost, x, y, ra\_hat, rb\_hat, ta\_hat, tb\_hat] = Cost4\_4(xg, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

figure

plot(thist, thetaahist, thist, ta\_hat);

title("Theta A Comparison");

legend("Measured", "Predicted");

figure

plot(thist, thetabhist, thist, tb\_hat);

title("Theta B Comparison");

legend("Measured", "Predicted");

R = diag([10^2\*ones(1, nk), 30^2\*ones(1, nk), 0.01^2\*ones(1, nk), 0.03^2\*ones(1, nk)]);

Pxx = inv(subH'\*inv(R)\*subH)

cost

xg











Pxx =

36.0180 136.4236 -0.2949 -0.9709

136.4236 862.0366 -0.7137 -4.4707

-0.2949 -0.7137 0.0084 0.0207

-0.9709 -4.4707 0.0207 0.0631

cost =

10.554503155959429

\*\* Note that this cost is compared vs predicting the theta information as well, even though it is not receiving theta information to make its predictions. otherwise it would be artificially lower, especially for such noisy thetahist data. The unadjusted cost is available by giving a scalar as ta to the cost function and comes back around 6.86 \*\*

xg =

1.0e+03 \*

1.512296126834710

1.835329779093162

0.899837188762494

1.498376843783257

Adding bearing information and recalculating

Pxx =

35.6556 134.5409 -0.2892 -0.9515

134.5409 852.2383 -0.6842 -4.3699

-0.2892 -0.6842 0.0083 0.0204

-0.9515 -4.3699 0.0204 0.0621

cost =

10.554503145931690

xg =

1.0e+03 \*

1.512296127348853

1.835329781940411

0.899837188754850

1.498376843755186

Thus, adding the theta information did not improve the estimates very much. It is possible if the theta measurements were less noisy they could be more useful, but most of the information is captured in the position and its change, so having measurements of the bearing at each step is not as indicative of the initial state which is being estimated. The jagged jumping in thetabhist seems unusual given its relatively small variance, suggesting that those are very accurate measurements, even for the small scale of theta.

d)

%% Bearing only measurement

% xg = xg;

% xg = P4\_4\_init(rhoahist, rhobhist, 3e4, thist);

z = [thetaahist; thetabhist];

for i=1:50

[subh, ~ , ~, subH] = H\_4\_4ext(xg, thist, nk, nz); % H is 12x4x2

% initial cost

if i == 1

Cost4\_4(xg, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

end

%% Update guess

delta = inv(subH'\*subH)\*subH'\*(z-subh);

jold = Cost4\_4(xg, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

jnew = Cost4\_4(xg+alpha\*delta, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

while jnew >= jold + 1e-3

alpha = alpha/2.0;

jnew = Cost4\_4(xg+alpha\*delta, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

end

xg = xg + alpha \* inv(subH'\*subH)\*subH'\*(z-subh);

end

[cost, x, y, ra\_hat, rb\_hat, ta\_hat, tb\_hat] = Cost4\_4(xg, thist, rhoahist, rhobhist, thetaahist, thetabhist, nk);

R = diag([0.01^2\*ones(1, nk), 0.03^2\*ones(1, nk)]);

format short

Pxx = inv(subH'\*inv(R)\*subH)

format long

cost

xg

rank(subH)

Pxx =

1.0e+07 \*

9.8583 1.9080 -0.0571 0.0130

1.9080 0.5256 -0.0114 0.0034

-0.0571 -0.0114 0.0005 -0.0002

0.0130 0.0034 -0.0002 0.0001

cost =

10.588004743478447

xg =

1.0e+03 \*

1.511825778980535

1.838483888327534

0.899840973473972

1.498385314748654

ans =

4

This results in no solution if the value for xg is not already close enough. Using a two point quadratic fit like before results in a cost of over 1000. Using the best xg from before, the cost actually increases a little bit, so possibly with better alpha selection it could reduce the cost slightly. H is full rank, however, so at least with a good enough initial guess, the system is observable using only the bearing measurements. With even a slight shift, using a scale of 1.005 factor, the system is convergent but the cost increases to 63.7, and using a 1% increase the cost rises to 123. So, it is very dependent on the initial guess of x, likely due to the first order approximations used in the calculation of H.

e)

The error covariance values are very large, likely due to the lack of information that the bearing contains related to the initial measurements. The diagonal elements are all quite large and this makes them not useful. Having an error covariance in the initial x position by more than 1000 times the actual value is largely pointless.

**P5-3)**

%% Solve A SLTI Kalman Filter problem

% Load data

clear all;

close all;

kf\_example02a;

Qk = 10;

Rk - 0.025;

xhat = zeros(51, 2);

xhat(1, :) = xhat0;

P = zeros(51, 2, 2);

P(1, :, :) = P0;

ev = zeros(50);

%% Begin Filter

for k=1:50

%% State and Cov. Prop

xbar = Fk\*xhat(k, :)';

Pbar = Fk\*squeeze(P(k, :, :))\*Fk' + Gammak\*Qk\*Gammak';

%% Measurement Update

nu = zhist(k) - Hk\*xbar;

s = Hk\*Pbar\*Hk' + Rk;

W = Pbar\*Hk'\*s^-1;

xhat(k+1, :) = xbar + W\*nu;

P(k+1, :, :) = Pbar - W\*s\*W';

ev(k) = nu\*inv(s)\*nu';

end

figure

plot(thist, xhat(2:end, 1), 'x', thist, sqrt(P(2:end, 1, 1)), 'o')

legend("Xhat(1)", "sqrt(P11(k))");

title("Prediction of X1 and prediction std")

xlabel("thist")

figure

plot(thist, xhat(2:end, 2), 'x', thist, sqrt(P(2:end, 2, 2)), 'o')

legend("Xhat(2)", "sqrt(P22(k))")

title("Prediction of X2 and prediction std")

xlabel("thist")

figure

plot(ev(:, 1), 'o')

legend("ev")

xlabel("k")

title("ev vs k")

P50 = squeeze(P(51, :, :))

Xhat50 = squeeze(xhat(51, :))

****

****

****

P50 =

0.000548311870423 0.000344465510828

0.000344465510829 0.044877377129582

Xhat50 =

0.043711606210112 -0.480298586307570

**P6-2)**

% Problem 2 from Problem set 6

% First KF then SRIF

clear all

close all

plotting = 1;

for example=1:2

if example==1

kf\_example03a;

else

kf\_example03b;

end

%% Solve with regular KF

xhat = zeros(51, 3);

xhat(1, :) = xhat0;

P = zeros(51, 3, 3);

P(1, :, :) = P0;

%% Begin regular KF

for k=1:50

%% State and Cov. Prop

xbar = Fk\*xhat(k, :)';

Pbar = Fk\*squeeze(P(k, :, :))\*Fk' + Gammak\*Qk\*Gammak';

%% Measurement Update

nu = zhist(k) - Hk\*xbar;

s = Hk\*Pbar\*Hk' + Rk;

W = Pbar\*Hk'\*s^-1;

xhat(k+1, :) = xbar + W\*nu;

P(k+1, :, :) = Pbar - W\*s\*W';

end

xhat50 = xhat(end, :)

P50 = squeeze(P(end, :, :))

%% Begin SRIF

returnx = 1;

nv = size(Gammak, 2);

nx = size(xhat, 2);

xhat2 = zeros(51, nx);

Rxx = zeros(51, nx, nx);

Rvv = zeros(nv, nv);

zx = zeros(51, nx);

Pxx2 = zeros(51, 3, 3);

Rxx(1, :, :) = [inv(chol(P0))]';

Rvv = [inv(chol(Qk))]';

zv = 0;

zx(1, :) = squeeze(Rxx(1, :, :))\*xhat0;

% static Rk

Ra = chol(Rk);

invRat = inv(Ra)';

for k=1:50

% propagation step

[Qa, RA] = qr([Rvv, zeros(nv, nx); -1\*squeeze(Rxx(k, :, :))\*inv(Fk)\*Gammak, squeeze(Rxx(k, :, :))\*inv(Fk)]);

dumbmatrixtoseperate = Qa'\*[zeros(nv, 1); zx(k, :)'];

zxbar = dumbmatrixtoseperate(nv+1:end);

Rxxbar = squeeze(RA(nv+1:end, nv+1:end, :, :));

% measurement update

za = invRat\*zhist(k);

Ha = invRat\*Hk;

[Qb, Rb] = qr([Rxxbar; Ha]);

dumbmatrix2 = Qb'\*[zxbar; za'];

zx(k+1, :) = dumbmatrix2(1:nx);

Rxx(k+1, :, :) = Rb(1:nx, :);

if returnx == 1

xhat2(k+1, :) = inv(squeeze(Rxx(k+1, :, :)))\*zx(k+1, :)';

Pxx2(k+1, :, :) = inv(squeeze(Rxx(k+1, :, :)))\*inv(squeeze(Rxx(k+1, :, :)))';

end

end

xhat50\_srif = xhat2(end, :)

P50\_srif = squeeze(Pxx2(end, :, :))

%% plot the results

if plotting==1

figure

plot(thist, xhat(2:end, 1), 'x', thist, xhat2(2:end, 1), 'o')

legend("KF", "SRIF");

title("Prediction of X1 and prediction std")

xlabel("thist")

figure

plot(thist, xhat(2:end, 2), 'x', thist, xhat2(2:end, 2), 'o')

legend("KF", "SRIF")

title("Prediction of X2 and prediction std")

xlabel("thist")

figure

plot(thist, xhat(2:end, 3), 'x', thist, xhat2(2:end, 3), 'o')

legend("KF", "SRIF");

title("Prediction of X3 and prediction std")

xlabel("thist")

end

end

Results from the example 03a







xhat50 =

1.0e+05 \*

-2.213564595810351 -0.054409163808035 -0.000767594055628

P50 =

8.636634889253187 -0.739095368822987 -6.416535033588190

-0.739095368751634 1.229226449189468 -0.595574771560763

-6.416535033587692 -0.595574771560763 8.608927649130877

xhat50\_srif =

1.0e+05 \*

-2.213564595810351 -0.054409163808035 -0.000767594055628

P50\_srif =

8.636634889207805 -0.739095368788383 -6.416535033611877

-0.739095368788383 1.229226449206625 -0.595574771558362

-6.416535033611877 -0.595574771558362 8.608927649149841

Results from example 03b







xhat50 =

1.0e+04 \*

-3.955861426224565 0.030921585896061 0.003253001286183

P50 =

6.579444345465284 -0.666453045460415 -5.246538254544291

-0.666453045597990 0.725808866506867 -0.785164687415588

-5.246538254548895 -0.785164687415687 6.816867629380383

xhat50\_srif =

1.0e+04 \*

-3.955861544458148 0.030921620993587 0.003253049324714

P50\_srif =

6.579445366357171 -0.666453123750737 -5.246539118855695

-0.666453123750737 0.725808943772337 -0.785164763793937

-5.246539118855695 -0.785164763793937 6.816868646443567

So it actually did not appear to have a particular problem with this example, with some slight variations occurring between the KF and SRIF. This was actually due to the ill conditioning occurring in the R value, which is never squared and even though it is small, we are adding a value which is not small and taking the inverse, so it is fairly inconsequential to the conditioning of the filter. Even reducing R to 1e-24 still produces very similar results. So, no there is not a significant difference between the two. Specifically, the difference is:

xdiff =

-0.001182335829071 0.000350975256310 0.000480385307881

Pdiff =

1.0e-05 \*

0.102089188747101 -0.007829032233619 -0.086431140466914

-0.007815274705170 0.007726547013309 -0.007637834931185

-0.086430680035221 -0.007637824983586 0.101706318478279

Which are exceptionally small differences regardless of the conditioning of R. This may have been more significant if 32 bit values were used, and definitely would have been problematic for 16 bit variables.