```
//###############################################################################
//
// The discrete Kalman Filter
//
// According to:
// Robert Grover Brown, Patrick Y. C. Hwang
// Introduction to Random Signals and Applied Kalman Filtering
// John Wiley & Sons, Inc, 1997
// See Problem 5.2

// Compute Kalman gain that minimizes the mean-square estimation error
// Enter prior estimate and its error covariance
//
function K = Compute_Gain (H, R_var, P_Minus)
    K = P_Minus * H' * inv(H * P_Minus * H' + R_var);
endfunction;

// Update estimate with measurement z
//
function x_Hat = Update_Estimate (H, K, x_Hat_Minus, z)
    // z <= Measurement
    x_Hat = x_Hat_Minus + K * (z - H * x_Hat_Minus);
endfunction;

// Compute error covariance for updated estimate
//
function P = Compute_Covariance (H, K, P_Minus)
    P = P_Minus - K * H * P_Minus;
endfunction;

// Project ahead
//
function [x_Hat_Minus_Plus_One, P_Minus_Plus_One] = Project_Ahead (Phi, Q, P, x_Hat)
    x_Hat_Minus_Plus_One = Phi * x_Hat;
    P_Minus_Plus_One = Phi * P * Phi' + Q;
endfunction;

//###############################################################################
//
// A simple application of the Kalman Filter

// --- --- ---
funcprot(0);

// Delta_T, sampling interval
D = 1;

// Number of meaurements
Nb_Of_Exps = 100;
Distance = 1.3;

// State transition matrix
Phi = [1, D, D*D/2;..
       0, 1, D;..
       0, 0, 1];

// Noiseless connection between measurement and state vector
H = [1, 0, 0];

// Power spectral density of white noise
```

```
W = 10;

// Covariance matrix
Q = W * [D^5/20, D^4/8, D^3/6;..
         D^4/8,  D^3/3, D^2/2;..
         D^3/6,  D^2/2, D];

R_var = 0; // Will be set

// Measurement, will be set
z = 0.0;

// Variances of the measurement errors
R_vars = zeros(Nb_Of_Exps);

// Simulating measurement erros
for Iv = 1 : Nb_Of_Exps
    R_vars(Iv) = (Distance / 2.0)^2;
end;

// Measurements
Measurements = zeros(Nb_Of_Exps);

// Simulating measurements
for Iy = 1 : Nb_Of_Exps
    Measurements(Iy) = grand(1, 1, "nor", (Iy - 1) * Distance, sqrt(R_vars(Iy)));
end;

// Some 'noise' ...

N = 30;
Measurements(N) = Measurements(N) + Measurements(N) / 3.0;
R_vars(N) = 100.0;

N = 40;
Measurements(N) = Measurements(N) - Measurements(N) / 3.0;
R_vars(N) = 200.0;

N = 50;
Measurements(N) = Measurements(N) + Measurements(N) / 2.0;
R_vars(N) = 400.0;

N = 60;
Measurements(N) = Measurements(N) - Measurements(N) / 3.0;
R_vars(N) = 1000.0;

N = 61;
Measurements(N) = Measurements(N) - Measurements(N) / 2.0;
R_vars(N) = 4000.0;

N = 62;
Measurements(N) = Measurements(N) - Measurements(N) / 2.5;
R_vars(N) = 3000.0;

// The Kaman filtered results
Result = zeros(Nb_Of_Exps);

// Initial values for the Kalman Filter ...

x_Hat_Minus = [0;..
               0;..
```

```
               0];

P_Minus = eye(3,3); // Better then 'zeros'

// P_Minus = zeros(3,3);
// P_Minus(3,3) = 1;

// Kalman Filter Loop ...

for Ix = 1 : Nb_Of_Exps

   R_var = R_vars(Ix);

   K = Compute_Gain (H, R_var, P_Minus);

   z = Measurements(Ix);

   x_Hat = Update_Estimate (H, K, x_Hat_Minus, z);

   Result(Ix) = x_Hat(1);

   P = Compute_Covariance (H, K, P_Minus);

  [x_Hat_Minus_Plus_One, P_Minus_Plus_One] = Project_Ahead (Phi, Q, P, x_Hat);

   // Prepare next loop
   //
   x_Hat_Minus = x_Hat_Minus_Plus_One;

   P_Minus = P_Minus_Plus_One;

end;

// Plotting data ...

xdata = linspace(0, Nb_Of_Exps * D, Nb_Of_Exps);

// Simulated measurements
//
plot(xdata, Measurements, '--');

// Kalman filtered smoothed data
//
plot(xdata, Result, '+-', 'colo', 'red');

// xs2png (0, "kalman_free_walk.png");
```