Application of Kalman Filters (Stochastic Observers)

Dr. Thomas Chmielewski

12 May 2018

Application of Kalman Filters

- **Definitions, Equations and Block Diagrams**
- Target Tracking Example 2nd order
- Extended Kalman Filter
- Smoothing
- Code Examples/demos

Based on a presentation made Jan 26, 2011
and snippets from a text in progress

Kalman filter

The Kalman filter is a recursive filter that estimates the state of a linear system (or linearized system) in the presence of both measurement noise and state noise.

- Used extensively during Apollo program
- Linear-Quadratic-Gaussian Controller
  - Similar to combining the Luenbeger Observer with state variable feedback, the KF can be combined with the Linear Quadratic Regulator (optimal control problem) in order to find state (time varying) feedback gains

Kalman filter

- Used for linear time invariant model with state and measurement noise
- There are continuous time and discrete time formulations. Use discrete time formulation – easier to understand for a brief presentation
- The Kalman Filter is a stochastic observer and "***Plays to the Model***" as do all observers

Kalman filter

The Kalman filter is a **minimum mean square error** (**MMSE**) estimator. I.e. it minimizes the mean squared error of the trace of P the covariance matrix.

Minimizes the square of difference between actual and estimated states

Kalman filter which is a minimum variance unbiased estimator of the states for the case of Gaussian noise. Gaussian noise is the worst case noise.

In this case the error covariance is Gaussian and estimate is the conditional mean. It is the best MSE estimator.

Kalman Filter $\Rightarrow$ min ||P|| : $P = cov(\tilde{y} = y - \hat{y}) = E[\tilde{y}] = 0$

If the Gaussian noise assumption for the noises is removed, the Kalman filter gives the best linear estimate of the state (minimum variance). Better ones are not linear estimates of state.

Kalman filter

In Summary:

The Kalman filter is the best (minimum variance estimator) linear

filter for any distribution and is the **best possible estimator if the plant, measurement noise and initial states are Gaussian**.

references: [Goodwin and Sin], [Sage and Melsa]

Normal Estimation Problem
Stochastic Plant & Measurement Noise

plant: $y(k) = \varphi\, y(k-1) + \psi\, w(k-1)$

measurement: $z(k) = h\, y(k) + v(k)$

y: n-vector — φ: n by n state transition matrix

z: m-vector — ψ: n by r input/state transition matrix

w: r-vector — h: m by n measurement matrix

v: m-vector

statistics:

$w: N(\bar{w}, Q_w)$ is white, stationary and

Qw ≥ 0 (positive semidefinite); $\bar{w}$ = E[w(k)],

Qw δ(k-j) = E[w(k)w'(j)] - $\bar{w}\bar{w}^T$

$v: N(\bar{v}, R)$ is white, stationary and

R > 0 (positive definite $\bar{v}$= E[v(k)],

R δ(k-j) = E[v(k)v'(j)] - $\bar{v}\bar{v}^T$

{w, v} are statistically independent.

Discrete Time Kalman Filter Equations
Predictor-Corrector Form

Kalman Filter ⇒ min ||P|| : $P = cov(\tilde{y} = y - \hat{y}) = E[\tilde{y}] = 0$

Kalman Filter Equations for Linear Time Invariant System with Stationary Noise

prediction: $\hat{y}$(k+1|k) = φ $\hat{y}$ k|k) + ψ $\bar{w}$

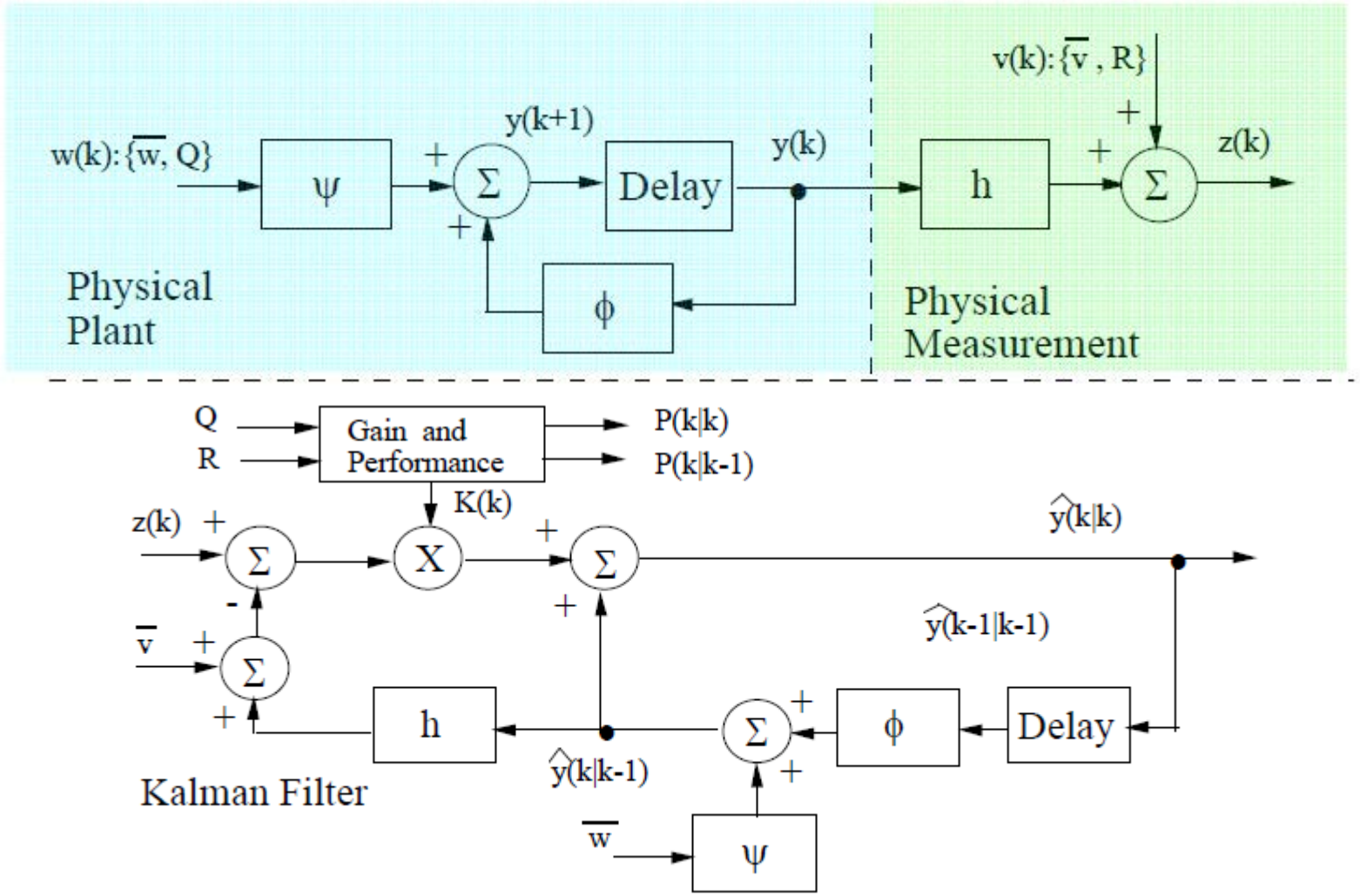
correction: $\hat{y}$ (k+1|k+1) = $\hat{y}$(k+1|k) + K(k+1) [z(k+1) – h $\hat{y}$(k+1|k) - $\bar{v}$]

predicted covariance: P(k+1|k) = φ P(k|k) φ' + Q ; Q = ψ Q_w ψ‘

filter gain: K(k+1) = P(k+1|k) h' [h P(k+1|k) h' + R]$^{-1}$

updated covariance: P(k+1|k+1) = [I - K(k+1) h] P(k+1|k)

Block Diagram: Physical Plant, measurement and Kalman Filter



Conditions for Existence of Kalman Filter

If Time-invariant system: {φ, h, ψ} are constants,
Stationary noises: {Q, R} are constants, and
Eigenvalues of the system are stabilizable and detectable;

then P(k|k) and P(k|k-1) converge to steady state values (as time ->infinity)

If the pair [φ,h] is completely observable, and
the pair [φ, D] is completely controllable, where Q = DD'

Then solution of the Riccati equation converges to $P_p > 0$

Interpretation [**Bar-Shalom**]

- Observability guarantees a "steady flow" of information about each component, thus preventing the uncertainty from becoming unbounded.
- Controllability guarantees that process noise enters each state component and prevents the covariance of the state estimate from converging to zero.
- The observability condition yields the existence of the steady state solution and the controllability causes it to be positive definite.

System Model – estimate states x[k] from measurements z[k+1]

Consider a linear time invariant discrete-time system subjected to a random vector noise, w.

dynamics: $x[k+1] = A\,x[k] + B\,w[k]$

x: n-state vector

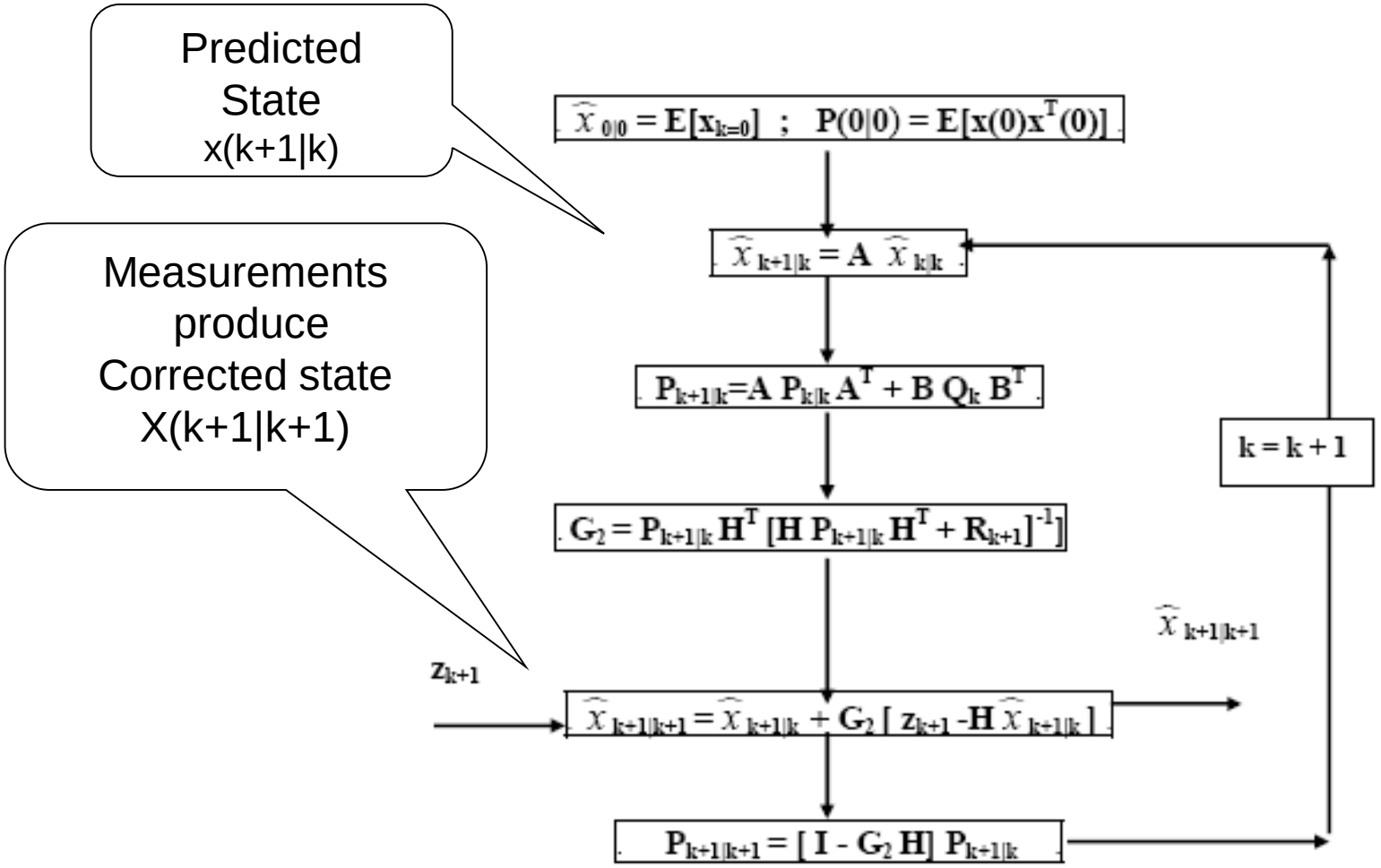
w: p-random vector

measurement: $z[k+1] = H\,x[k+1] + n[k+1]$

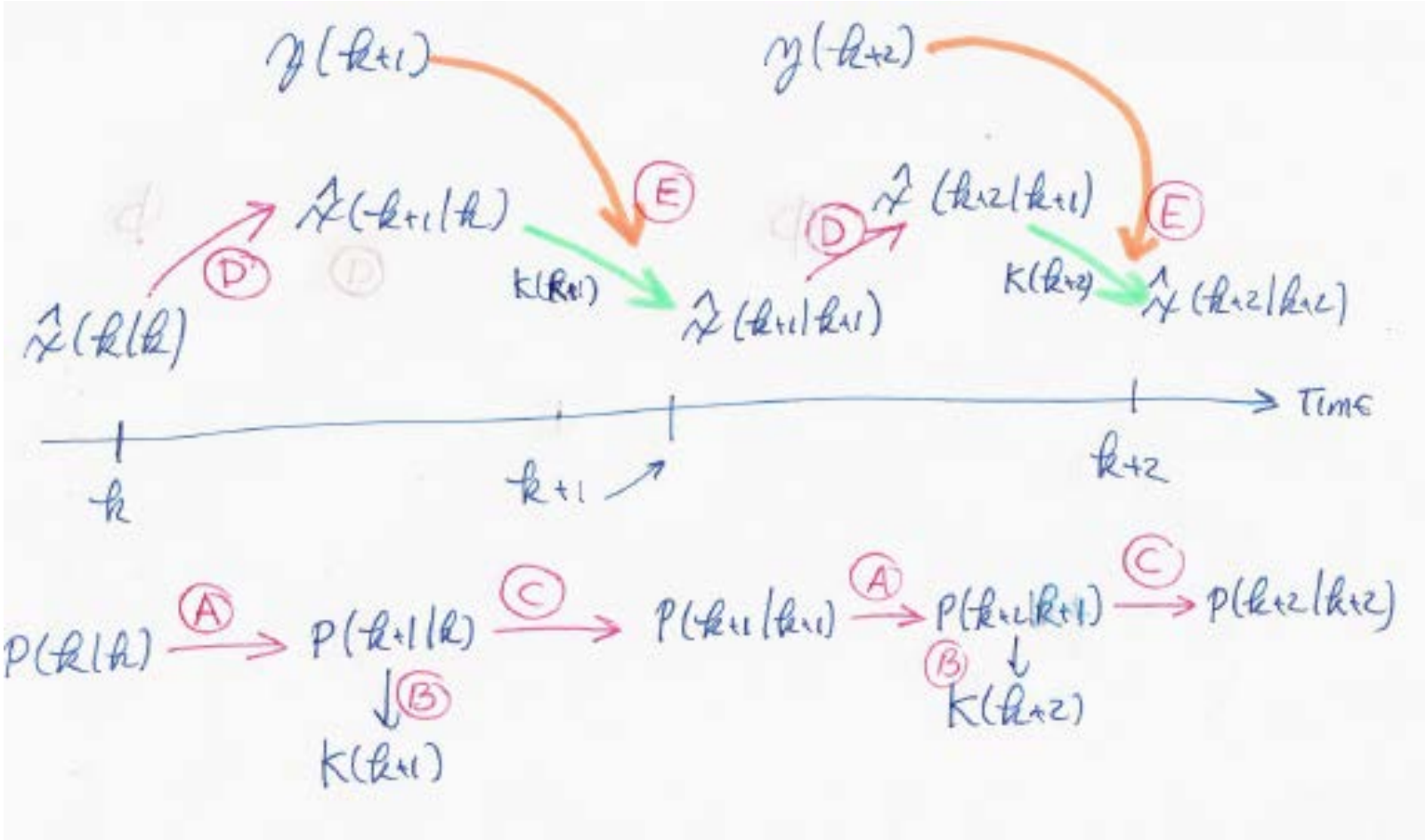
z, n: q-vectors

If the random vectors are zero mean and uncorrelated, the input random vector has covariance $E[ww^T] = Q[k]$ and the measurement random vector has covariance $E[nn^T] = R[k]$

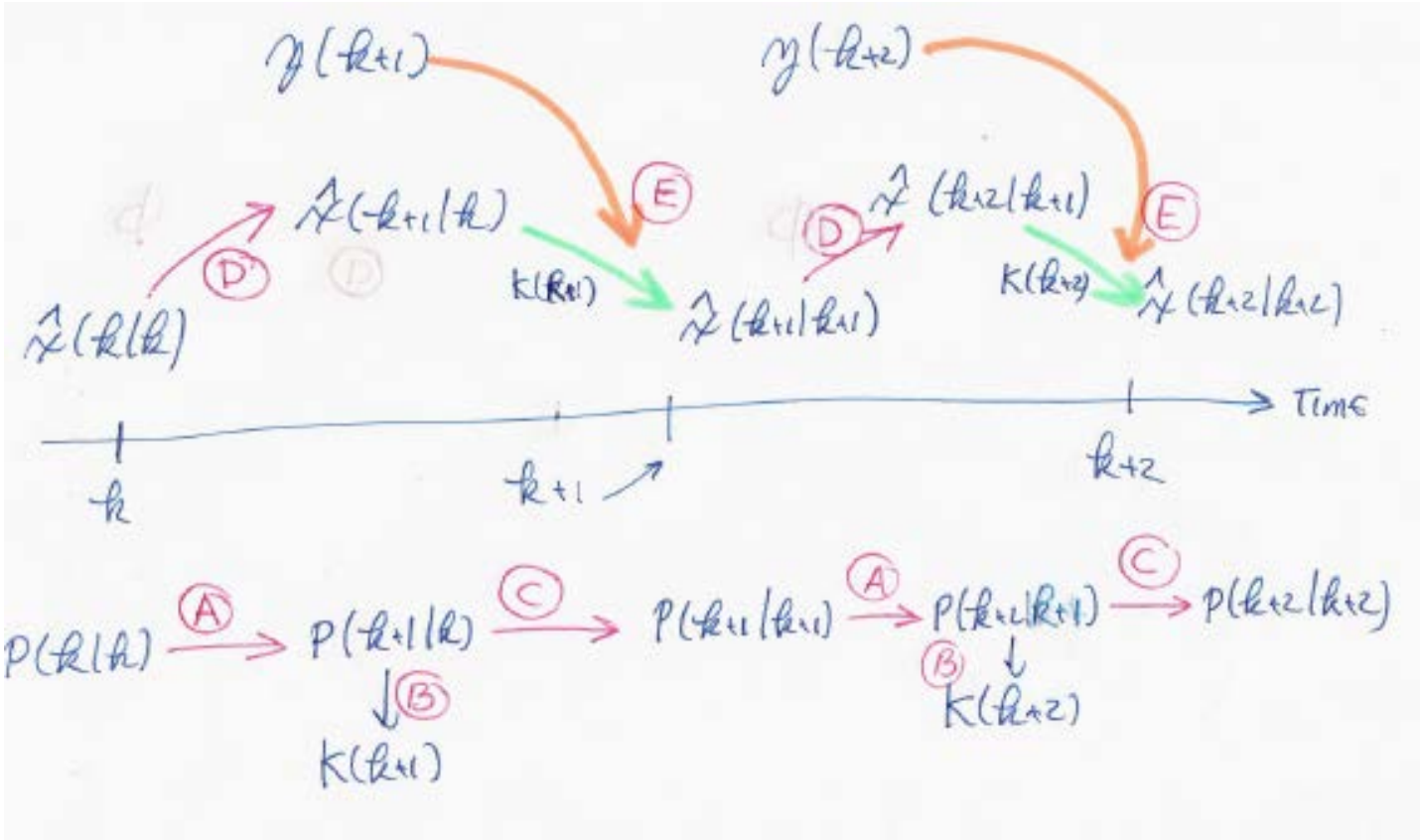
Predictor-Corrector Kalman Filter Recursive State and Performance



Graphic of Updates
State and Covariance



Discussion
Missing one or more measurements



If we miss a measurement we can still predict, however the covariance value will increase. This is called coasting, similar to deterministic observer if measurement is lost and residual set to zero to let observer dynamics perform estimate.

Some Comments on Kalman filter

- *If $\{w_k\}$ and $\{n_k\}$ are zero mean, uncorrelated, and white, then the KF is the best linear solution to minimizing the variance of the state error*
- Computes *predicted state estimate* and corresponding covariance (performance measure)
- After measurements, computes *corrected state estimate* and corresponding covariance
- For Gaussian noise - the square root of covariance = standard deviation of each state.
- Performance measure is not reflective of actual data presented to algorithm. Performance is based on propagation of mean and variance.
- Under certain conditions the gains and performance reach steady state values. You do not need to compute each iteration.

Application of Kalman Filters

- **Definitions, Equations and Block Diagrams**
- **Target Tracking Example 2nd order**
- Extended Kalman Filter
- Smoothing
- Code Examples/demos

Kalman Filter as Target Tracker

$$\begin{bmatrix} x(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} w[k]$$

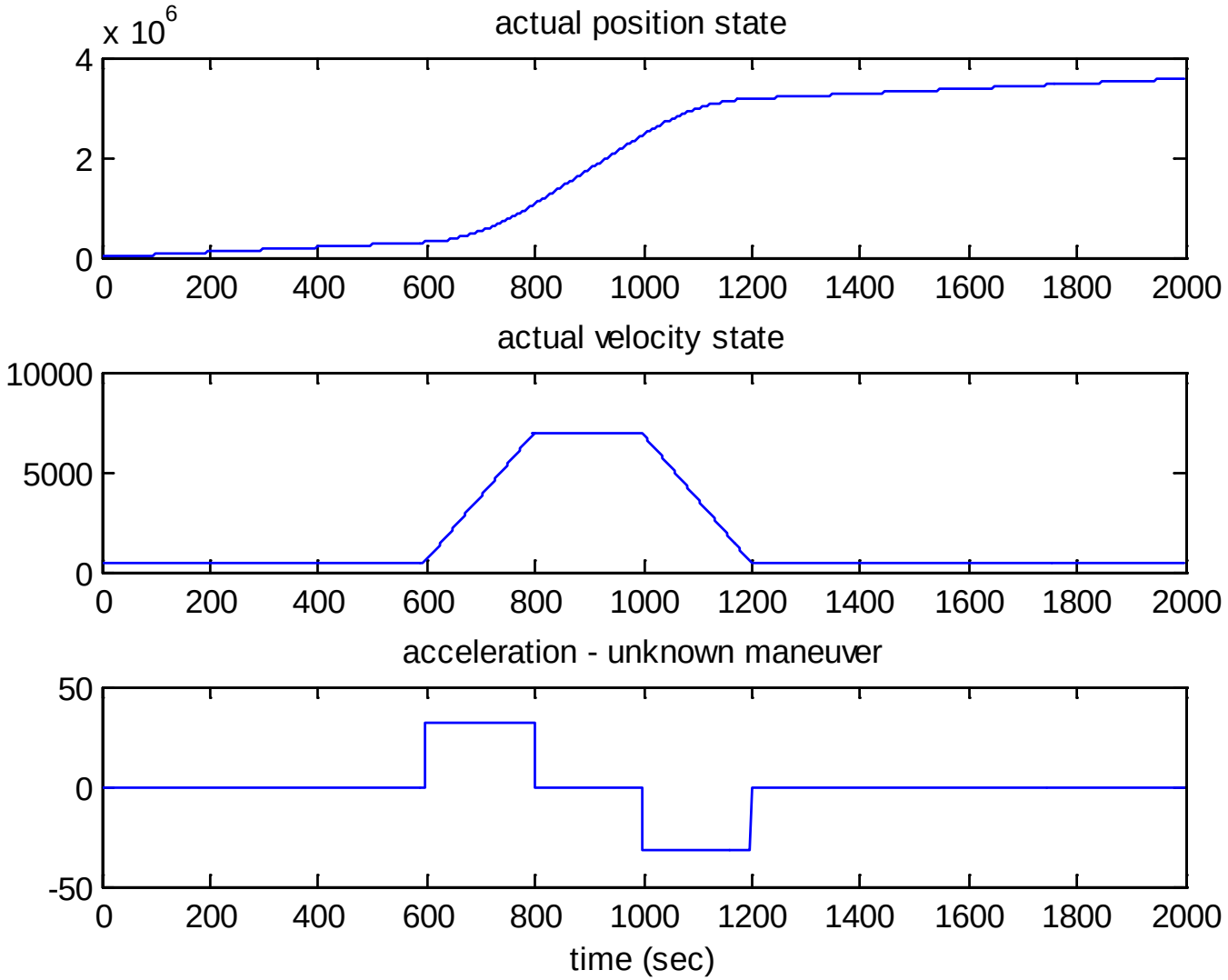
$$z = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} + n[k]$$

- One dimensional kinematic model
- Unknown input waveform w[k] has variance of 32.2 ft/sec^2 based on 1 g maneuver for acceleration
- Measurement noise variance 100^2
- w[k] and n[k] are uncorrelated

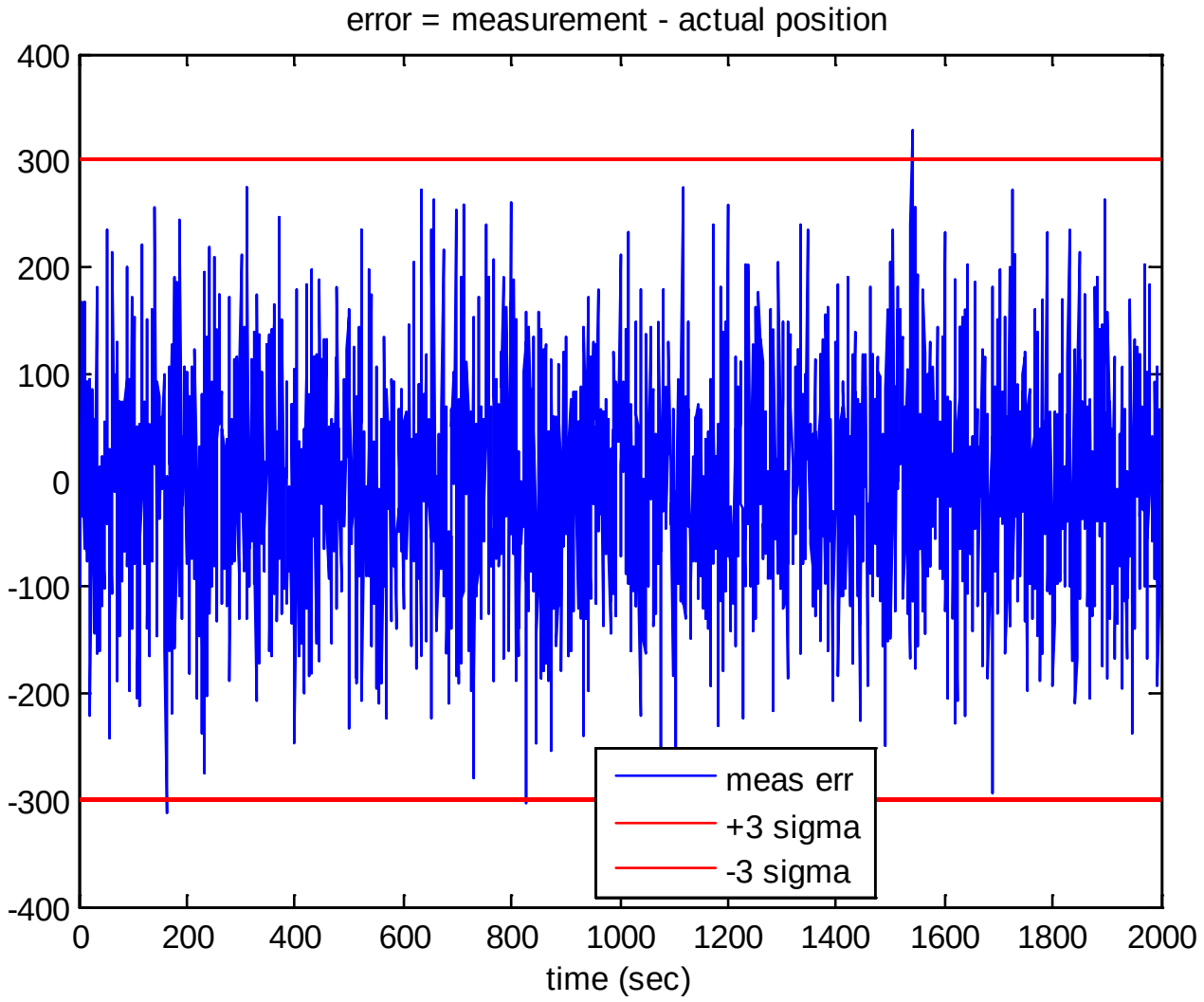
For Demonstration of target tracking

- Generate ground truth signals using an input that has known variance (this is deterministic but unknown – such as an aircraft maneuver)
- Add measurement noise to obtain {z[k]}
- Use {z[k]} to estimate position and velocity state
- Compare results to ground truth

Ground Truth Target tracks



Error = Position Measurement -Actual Position



Code Segment
2nd order Target Tracker

Initialize

```
%%  initialize P(k|k) and P(K+1|k)
PCINIT = 1e6*eye(order); % this is p(0|0)
PC = PCINIT;
PP = FI*PC*FI' + QA; %  P(1|0);  % use KF equation


xpinit = [0;0;0]; %  initial guess
xpinit = x_ic;
xp = xpinit;
xphst = xp;
```

Predictor Corrector Form

```
for i = 1:n
   K = PP*MA'*inv(MA*PP*MA' + R);
    khist = [khist,K]; % store
    xc = xp + K*(xm(i) - MA*xp);
    xfhst = [xfhst,xc]; % store
    PC = (eye(order)-K*MA)*PP;
    PCkhst = [PCkhst,PC]; % store
    xp = FI*xc;
    xphst = [xphst,xp];  % store
    PP = FI*PC*FI' + QA;
    PPkhst = [PPkhst, PP]; %store
end
```

Simulation Results: 2nd Order Target Tracker Steady State Values

PC =

1.0e+003 *

5.4941	2.1614
2.1614	2.1171

PP =

1.0e+004 *

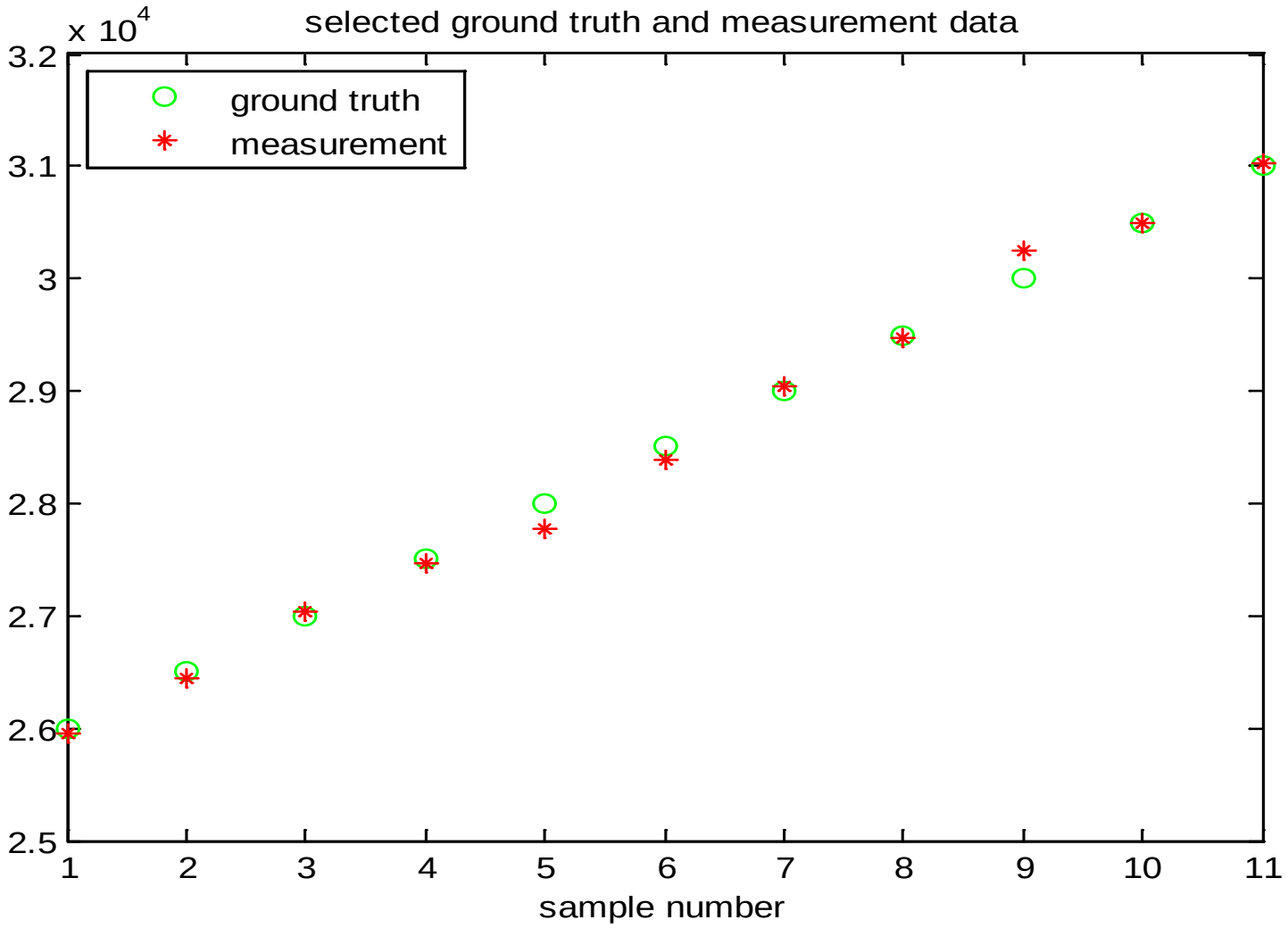
1.2193	0.4797
0.4797	0.3154

K =

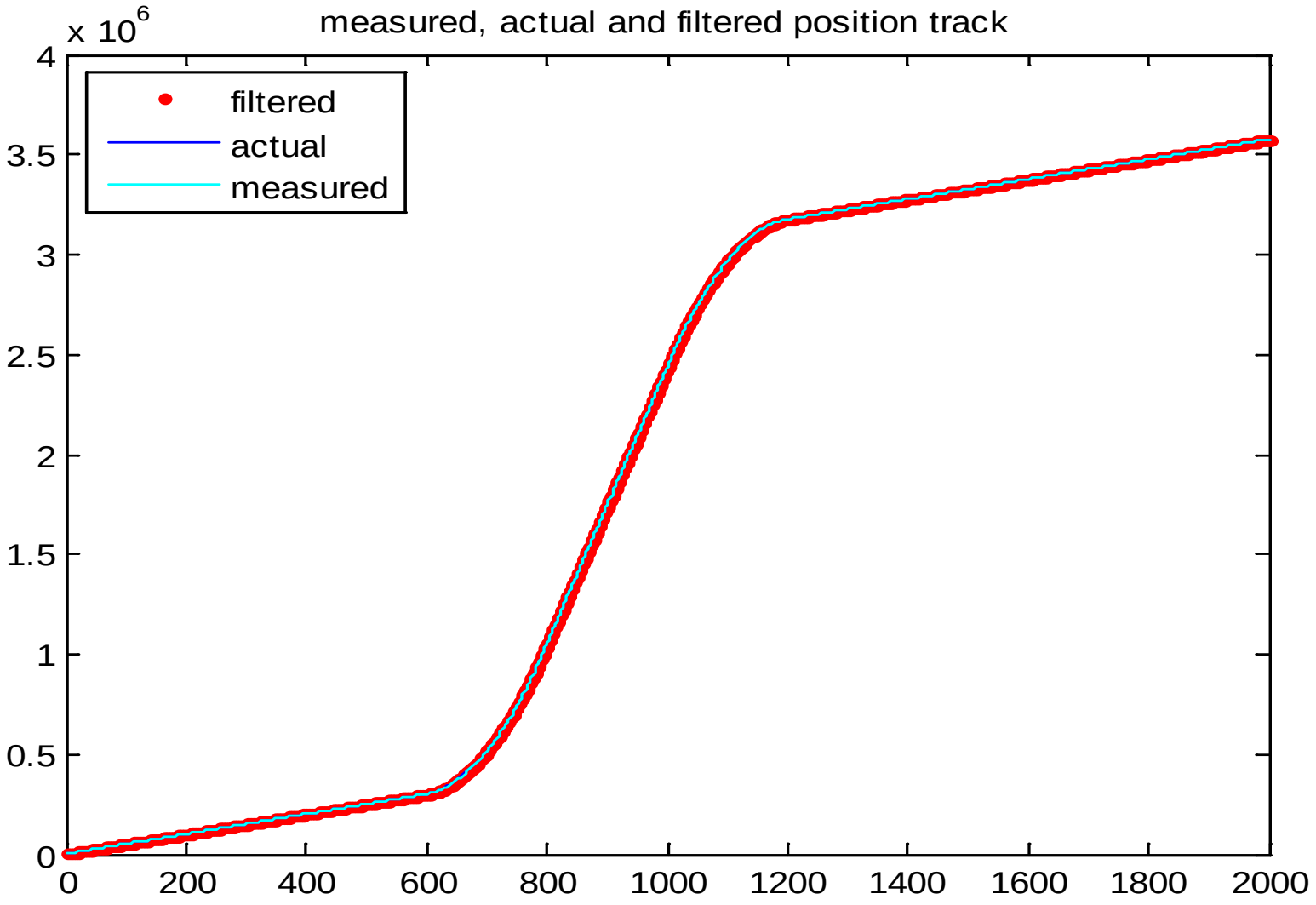
0.5494
0.2161

- Steady state Corrected Covariance Matrix
- Steady State Predicted Covariance Matrix PP>PC
- Steady State Kalman gains

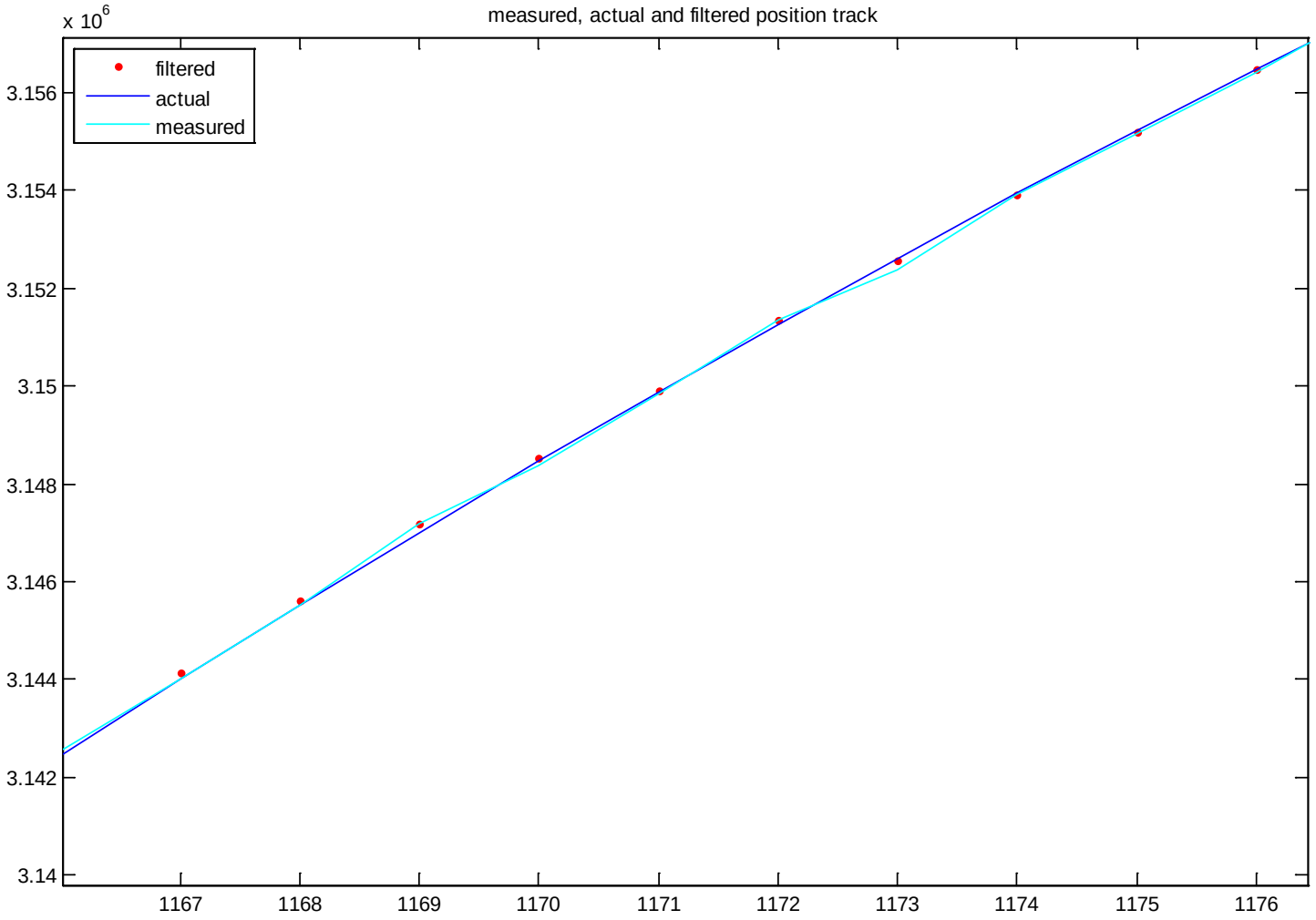
Measurement and Ground Truth Discrete Time Points



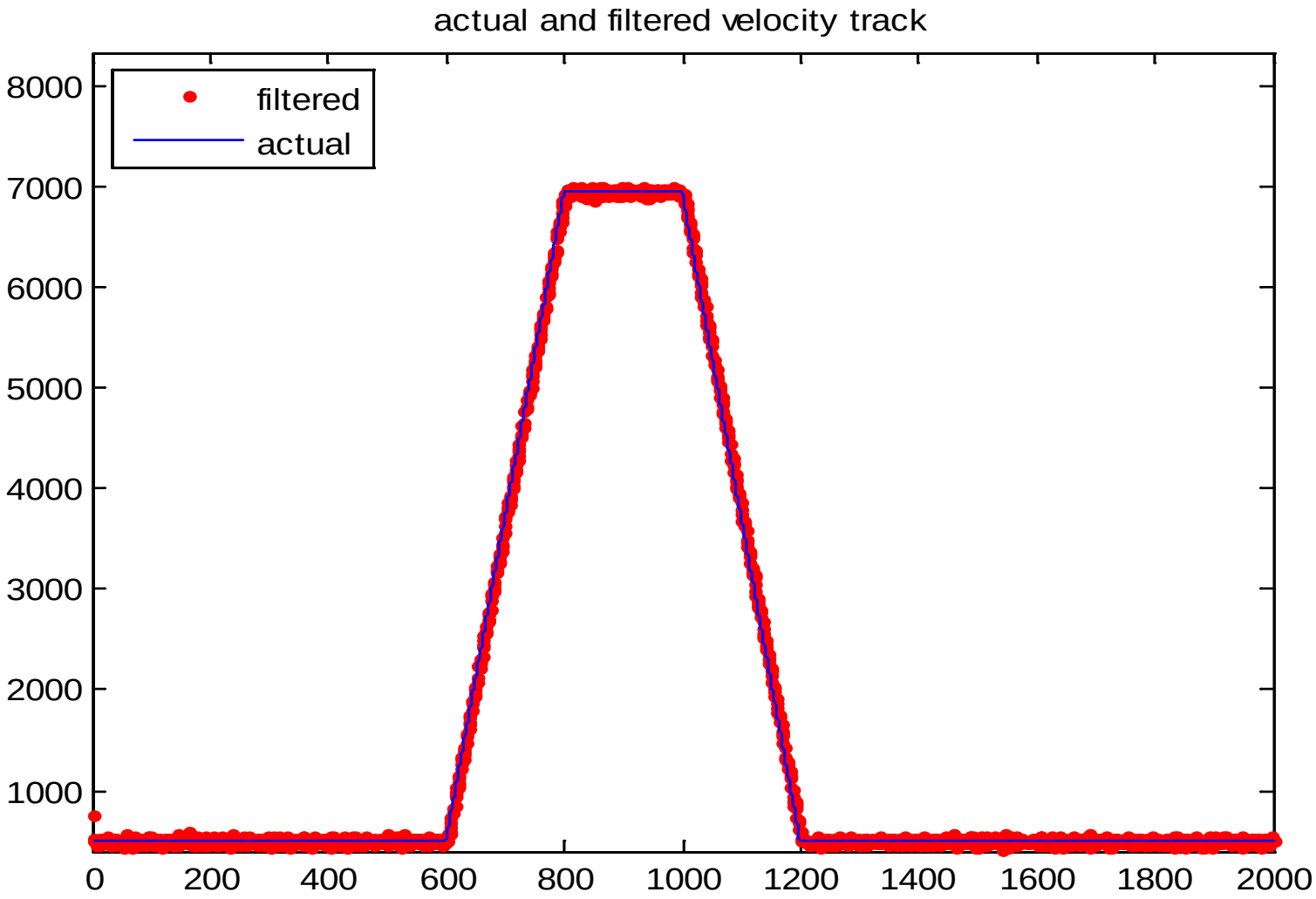
Result of filtering: Position



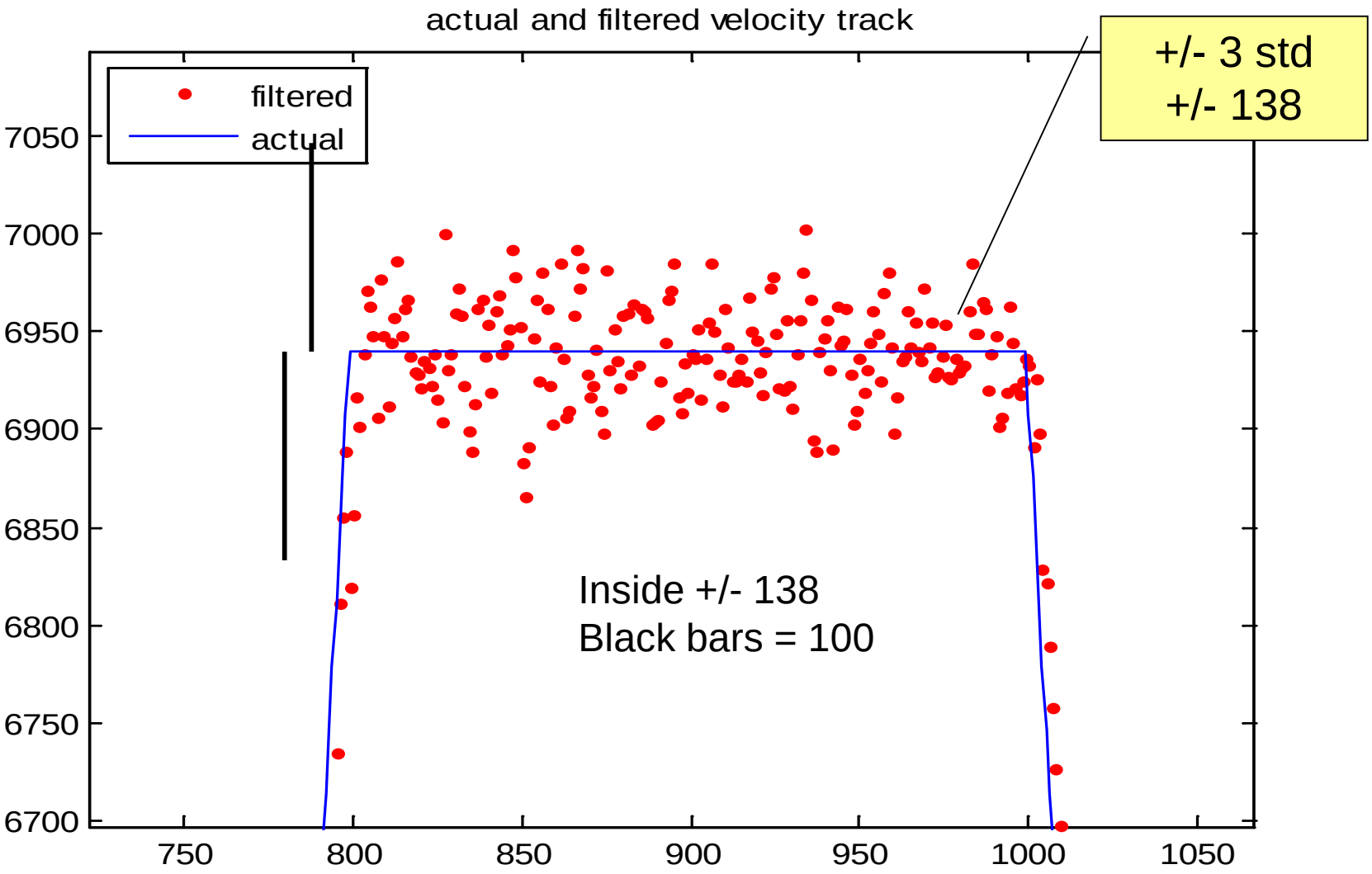
Result of filtering: Position



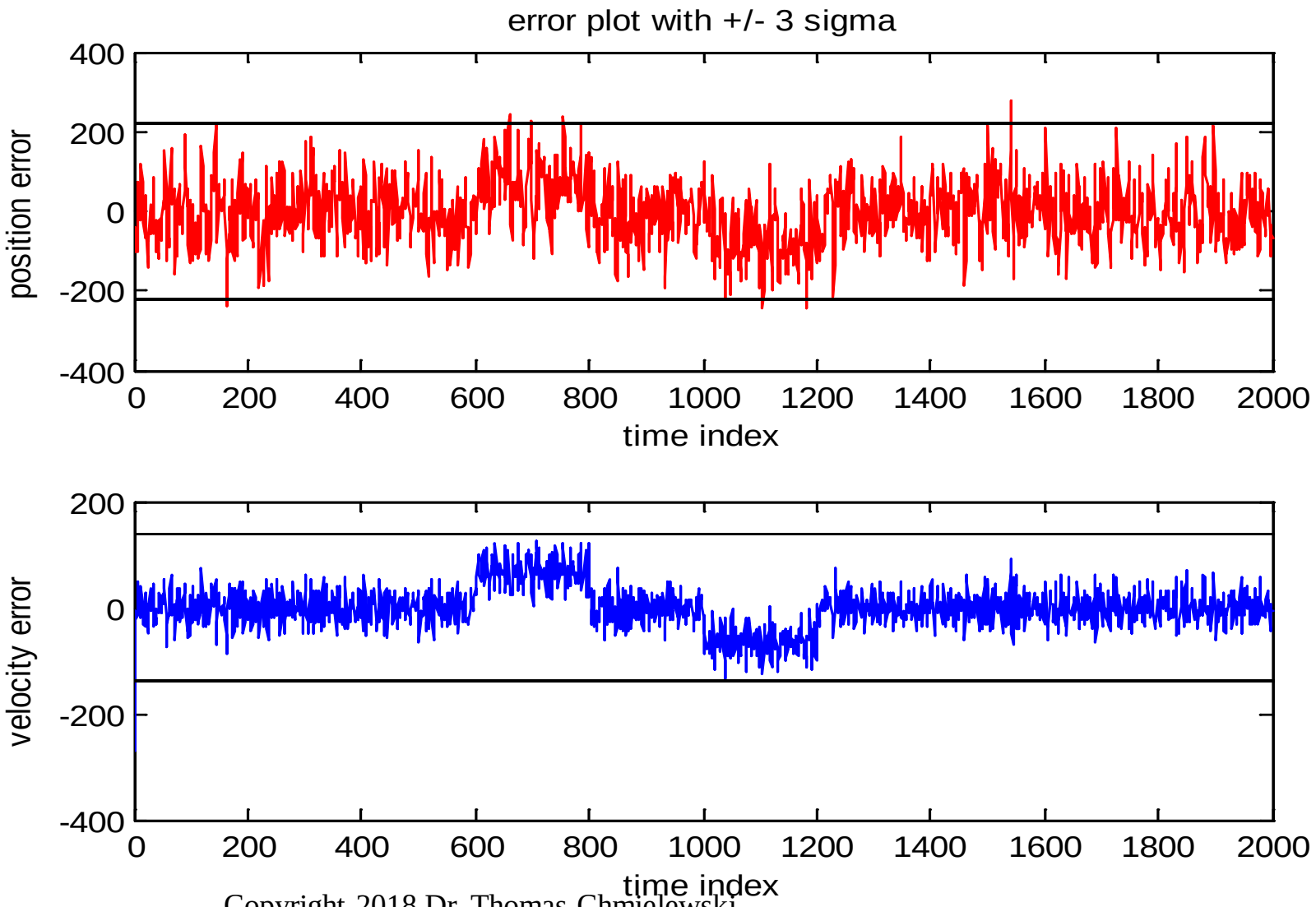
Result of Filtering: Velocity



Constant velocity portion estimate is better than WC KF



Error = actual - estimated



The statistics

- The unknown input waveform w[k] was characterized by a variance of 32.2ft/sec^2 – maneuver noise
- Measurement noise had variance of 100^2 or +/- 300 (3 standard deviations)
- Error of corrected position state had variance of 5491 or +/-222 (3 standard deviations)
- Error of corrected velocity state had variance of 2117 or +/- 138 (3 standard deviations)
- Take away – state estimates improve better with KF

Application of Kalman Filters

- **Definitions, Equations and Block Diagrams**
- **Target Tracking Example 2nd order**
- **Extended Kalman Filter**
- Smoothing
- Code Examples/demos

Extended Kalman Filter

- Allows the KF to be applied to a **non-linear** system
  - $x_k = f_{k-1}[x_{k-1}, u_{k-1}, w_{k-1}]$
  - $y_k = h_k[x_k, v_k]$
- Taylor series expansion about state $\hat{x}_{k-1}^{+}$ and $w_{k-1} = 0$
- Allows use of linear state space equation and linear measurement equation
- EKF is most widely applied *state estimation algorithm for non linear systems*
- **EKK can give unreliable results if non linearity is severe**.

Parameter Estimation via EKF

- Consider nonlinear discrete time system where p is an unknown parameter
- We want to estimate the constant p (and the state)
- Augmented state, sometimes don't need all states, x_k
- Parameter dynamics are constant (plus noise)

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}_k(\boldsymbol{p}) + \boldsymbol{G}_k(\boldsymbol{p})\boldsymbol{u}_k + \boldsymbol{L}_k(\boldsymbol{p})$$

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k$$

$$\boldsymbol{x}'_k = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{p} \end{bmatrix}$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{w}_{pk}$$

Parameter Estimation via EKF

Using augmented states form new system model

$$\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{p}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_k(\boldsymbol{p})\boldsymbol{x}_k + \boldsymbol{G}_k(\boldsymbol{p})\boldsymbol{u}_k + \boldsymbol{L}_k(\boldsymbol{p})\boldsymbol{w}_k \\ \boldsymbol{p}_k + \boldsymbol{w}_{pk} \end{bmatrix}$$

$$\boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{H}_k & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{p}_k \end{bmatrix} + \boldsymbol{v}_k$$

EKF used to estimate augmented state.

Note in this case we do not necessarily reach steady state.

Parameter Estimation via EKF

- 1st order system with "p" unknown
- Nonlinear augmented formulation (2nd order)
- Linearized matrices for use with KF (state and noise matrix)

$$x_{k+1} = px_k + w_k$$

$$y_k = x_k + v_k$$

$$\begin{bmatrix} x_{k+1} \\ p_{k+1} \end{bmatrix} = \begin{bmatrix} px_k + w_k \\ p_k + w_{pk} \end{bmatrix}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} + \begin{bmatrix} x_k \\ p_k \end{bmatrix} + v_k$$

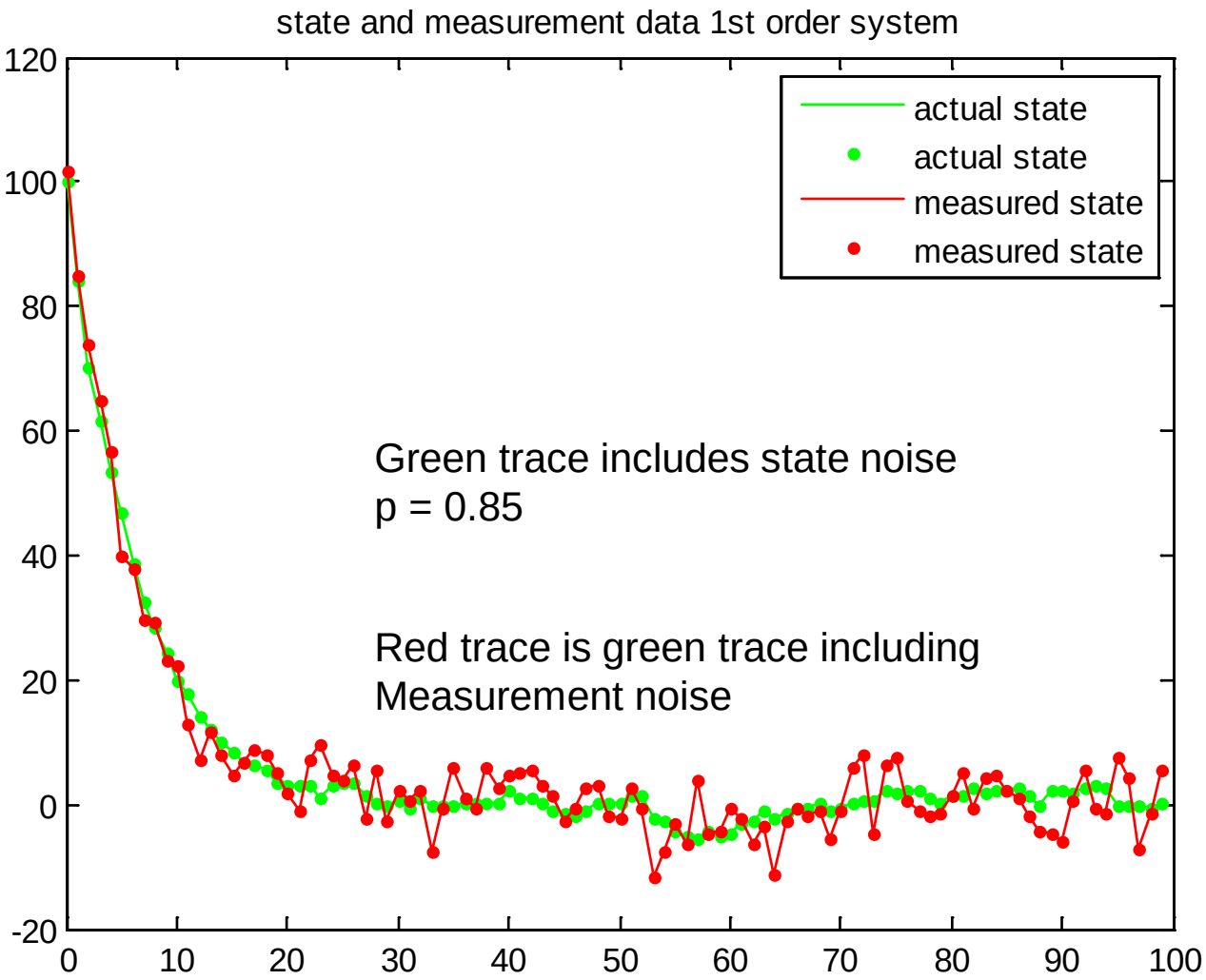
$$F_{k-1} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial p} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial p} \end{bmatrix}_{\hat{x}_{k-1}} = \begin{bmatrix} pk & xk \\ 0 & 1 \end{bmatrix}$$

$$L_{k-1} = \begin{bmatrix} \frac{\partial f_1}{\partial w} & \frac{\partial f_1}{\partial w_p} \\ \frac{\partial f_2}{\partial w} & \frac{\partial f_2}{\partial w_p} \end{bmatrix}_{\hat{x}_{k-1}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

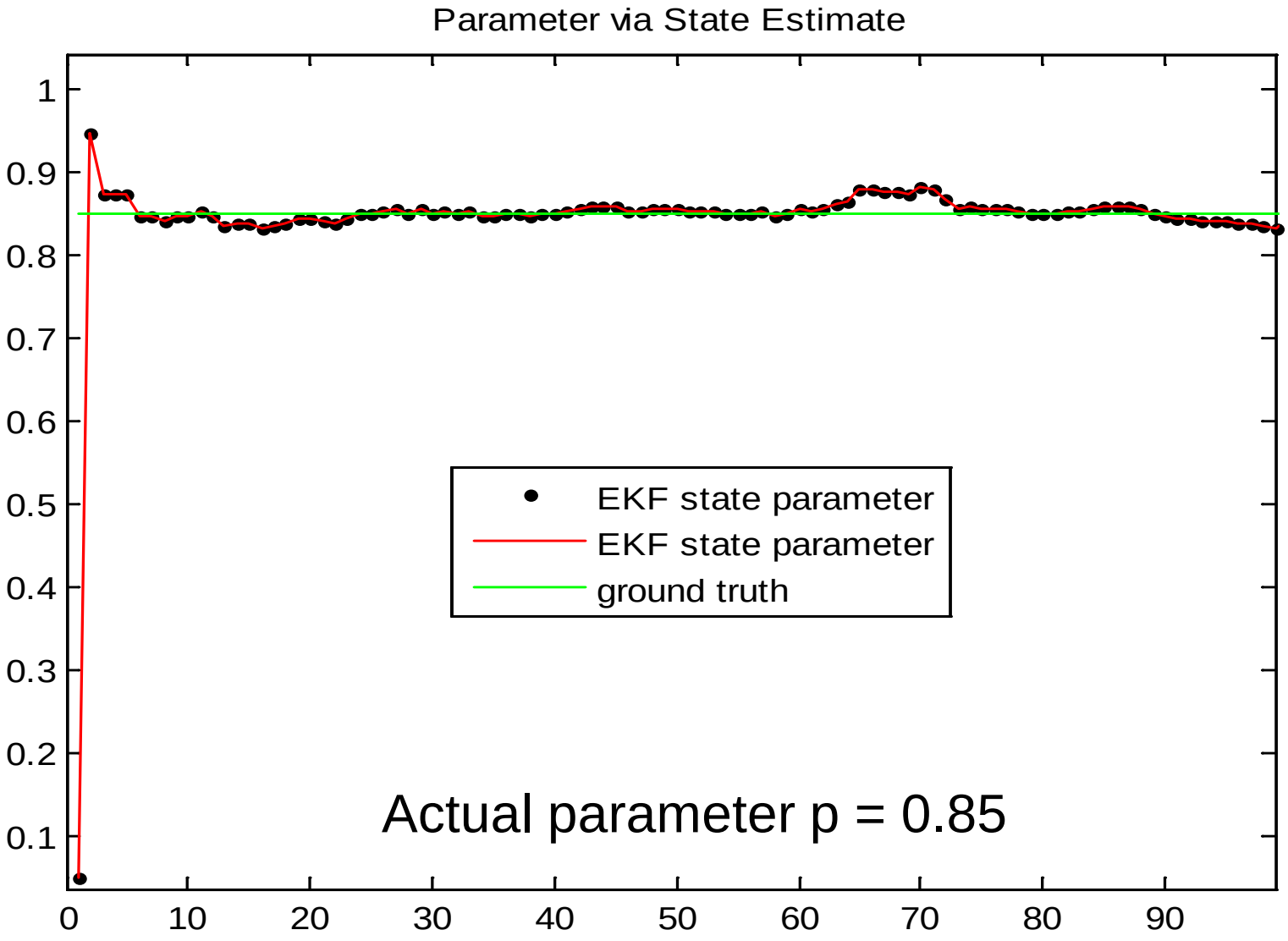
Example
Parameter Estimation via EKF

- System with “p” unknown
  - Ground truth p = 0.85
- Measurement noise statistics
  - Gaussian: mean = 0; variance = 4^2
- State noise statistics
  - Gaussian: mean = 0; variance = 1^2

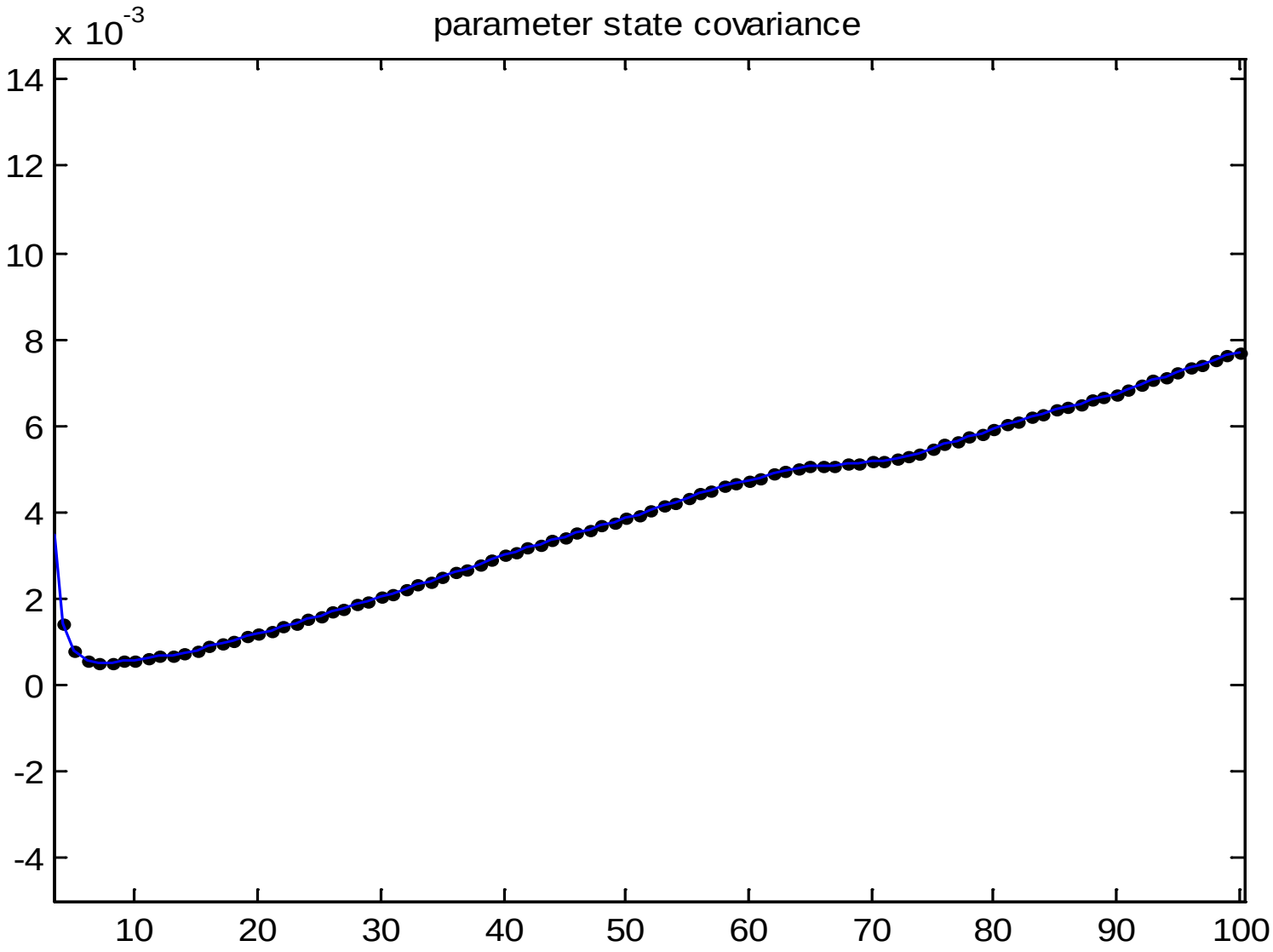
Simulation: Data Creation and Ground Truth



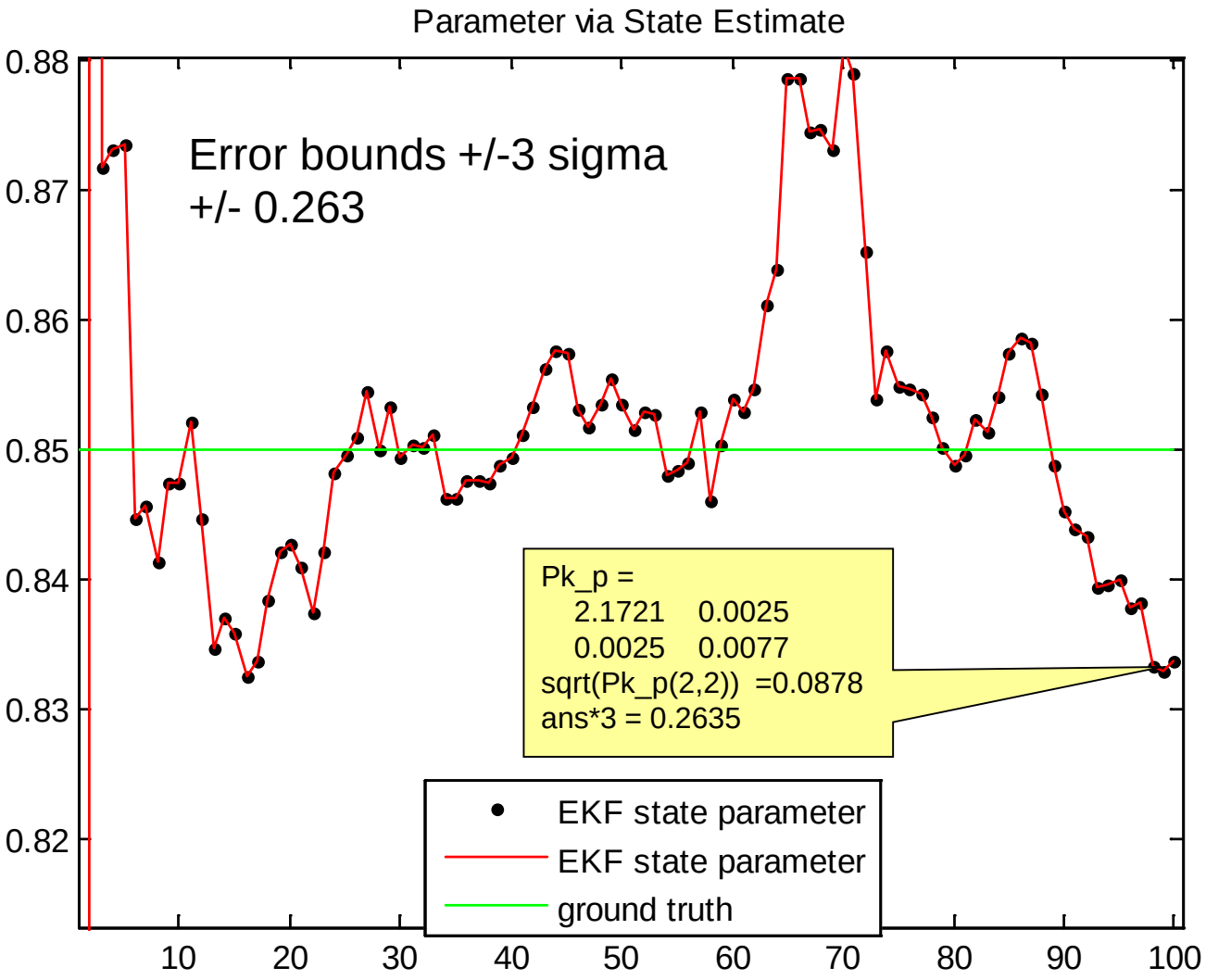
Estimated parameter “p”



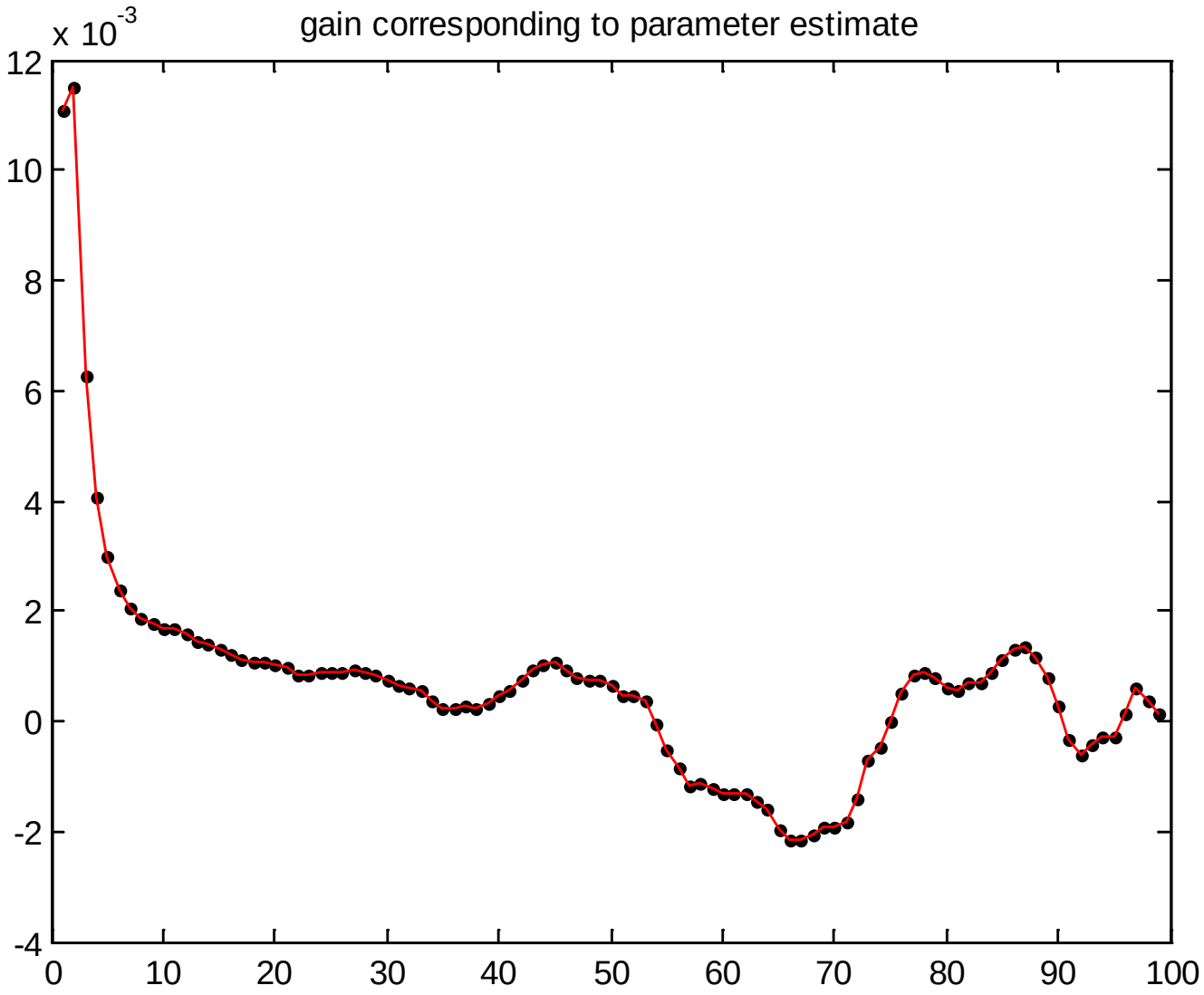
Covariance for parameter “p”



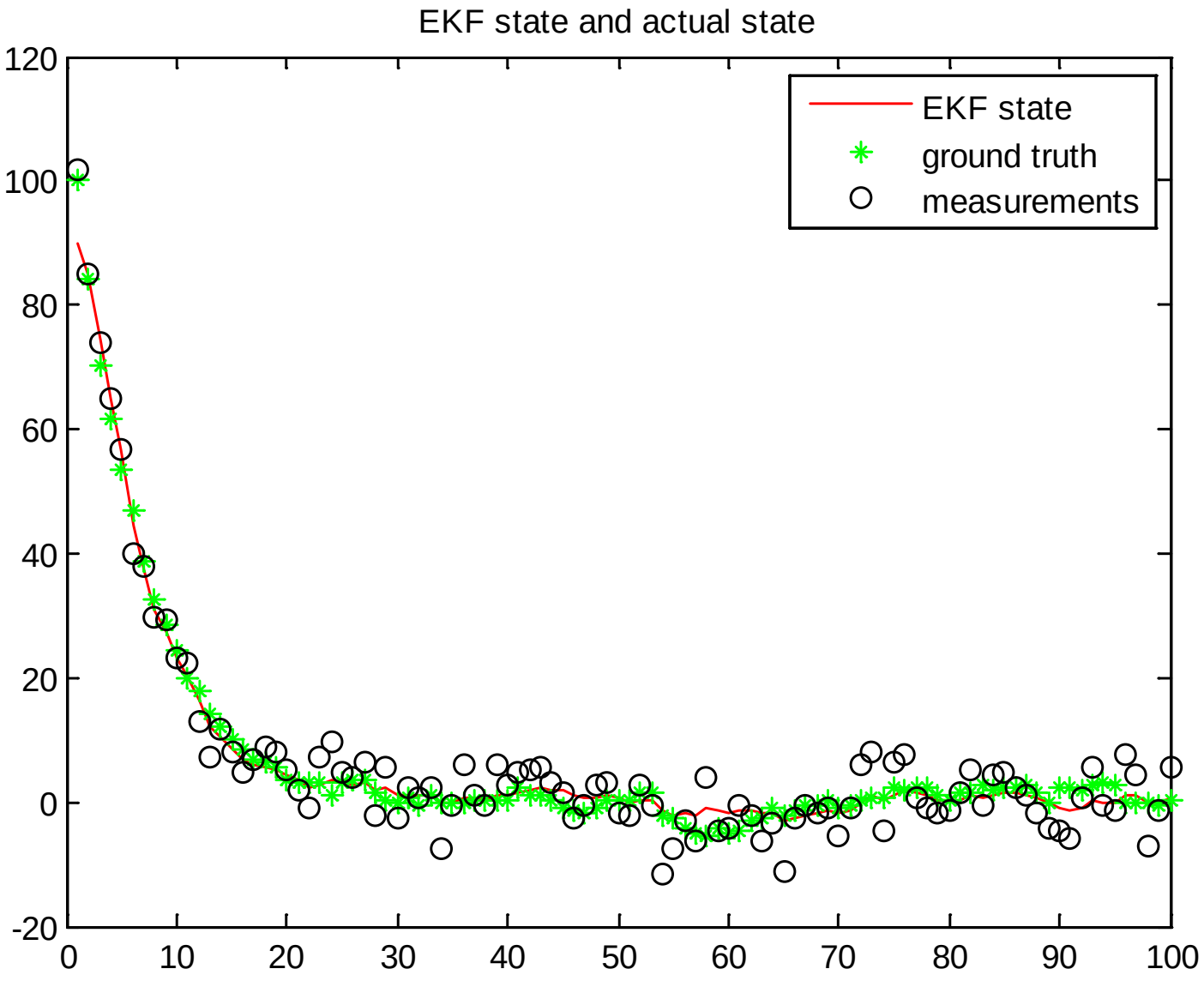
Trajectory for state “p”

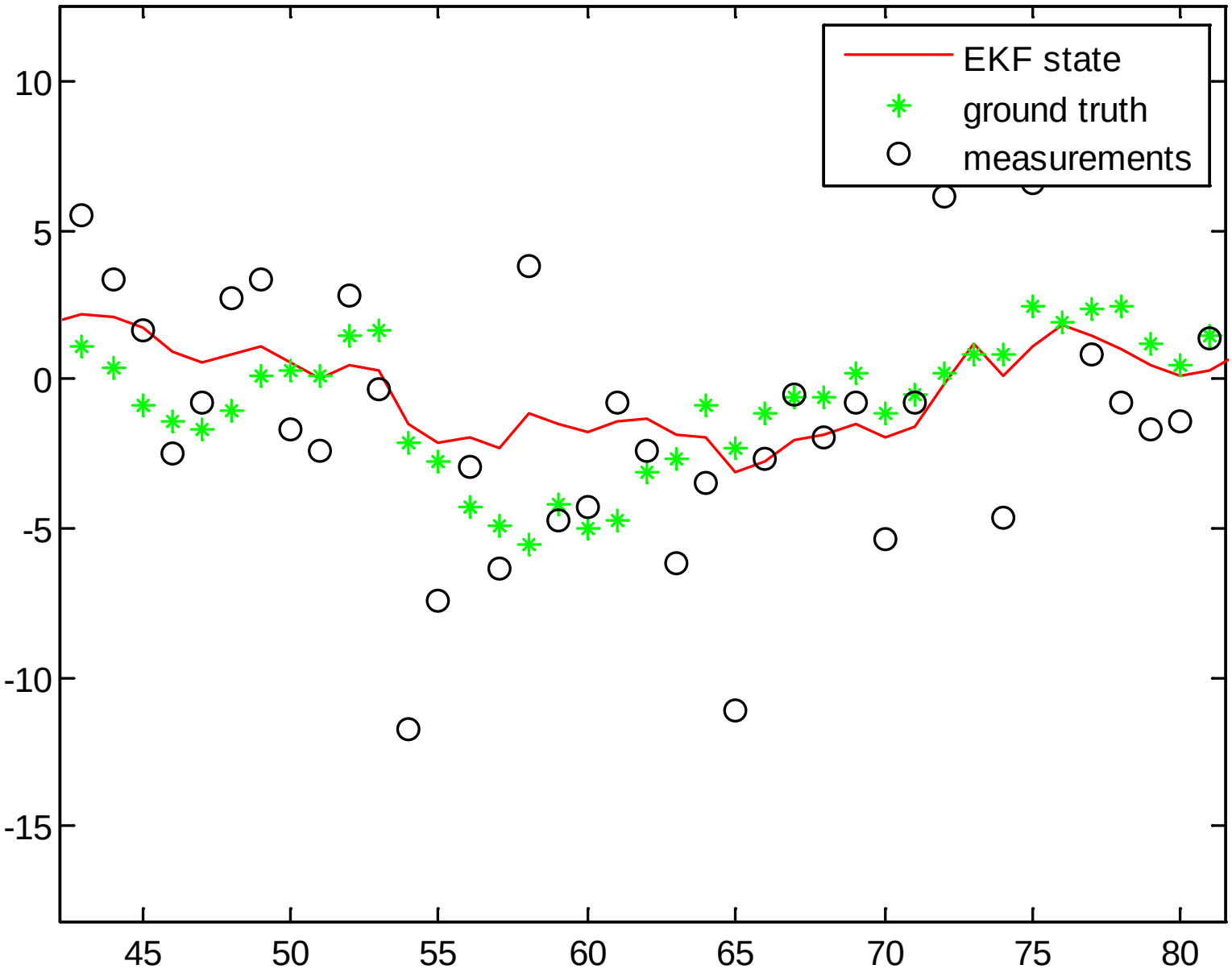


Gain for Parameter estimate



Estimated state x (based on est. of ‘p’)





EKF Summary

- The Extended Kalman Filter can be used to simultaneously estimate unknown parameter(s) and estimate the state of the system

- Contrasting to the Alpha-Beta or Alpha-Beta-Gamma Target tracker (which can use steady state values), the EKF exhibits time varying gains for both the state and parameter estimate. The covariance gives the error variance.

Application of Kalman Filters

- **Definitions, Equations and Block Diagrams**
- **Target Tracking Example 2nd order**
- **Extended Kalman Filter**
- **Smoothing**
- Code Examples/demos

Smoothing

In smoothing we wish to improve the estimate of the state x[t_1] given the measurements/observations:$Z[t_2] = \{ z[t], t_0 \leq t \leq t_2 \}$ and $t_2 > t_1$.

Estimate the state at a time prior to the current time wrt observation sequence. More info to help refine state estimate.

- Fixed-interval smoothing:
  - Observation interval is fixed t_2 is constant and t_1 is varied from t_0 to t_2
- Fixed-point smoothing:
  - Time of estimate t_1 is fixed and $t_2 \geq t_1$. Refine a single point
- Fixed-lag smoothing:
  - Difference between t_2 and t_1 =T is fixed, $t_2 = t_1 + T$

Track Smoothing

- Track smoothing requires state estimates based on all the measurements rather than only the data up to the given time as in the tracking process. Similar to a fixed point smoother.

- Track smoothing is achieved by the optimal combination of two independent tracking filters: a forward in time tracker (real time) and a backwards in time prediction tracker (after the fact).

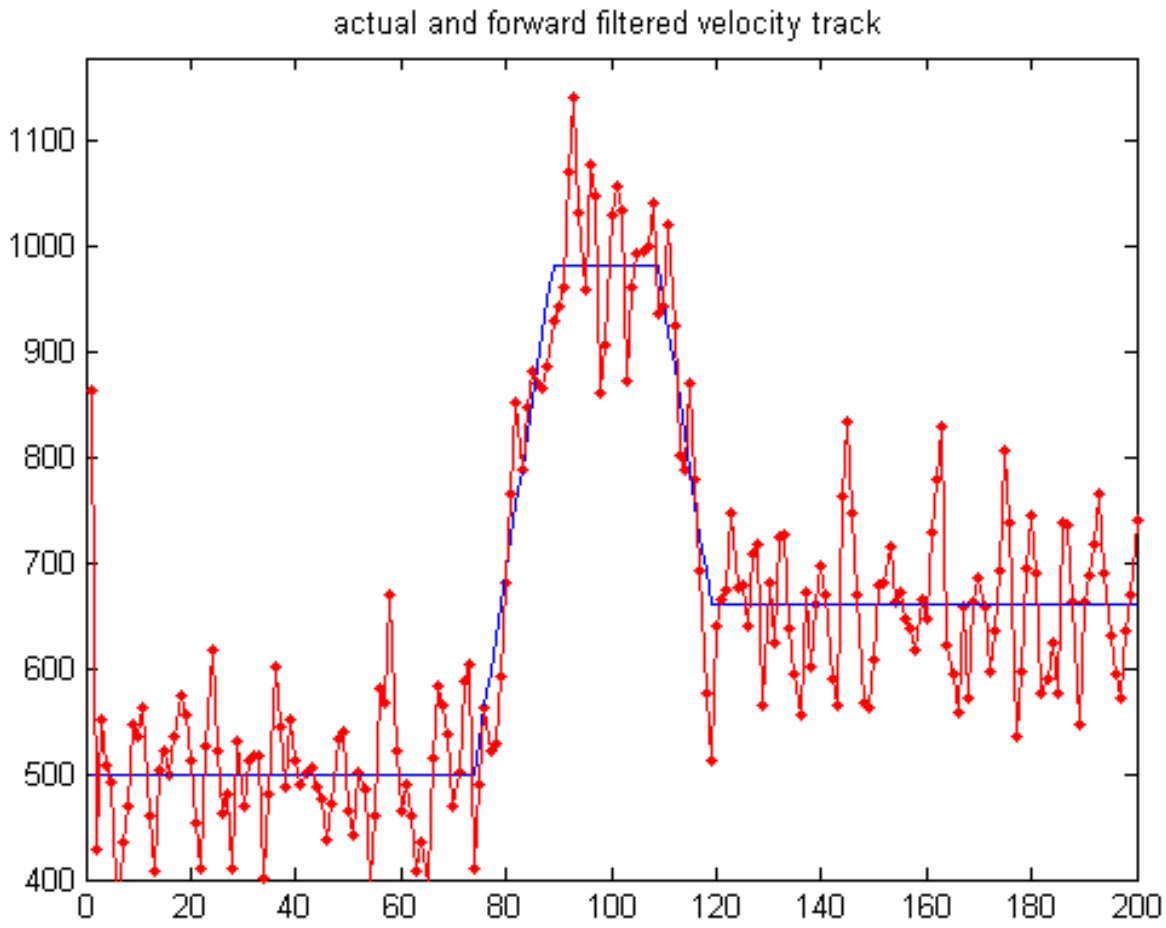
- The combination coefficients and the smoother performance are presented in closed form and are dependent on the Tracking Index

Table defining maneuver waveform

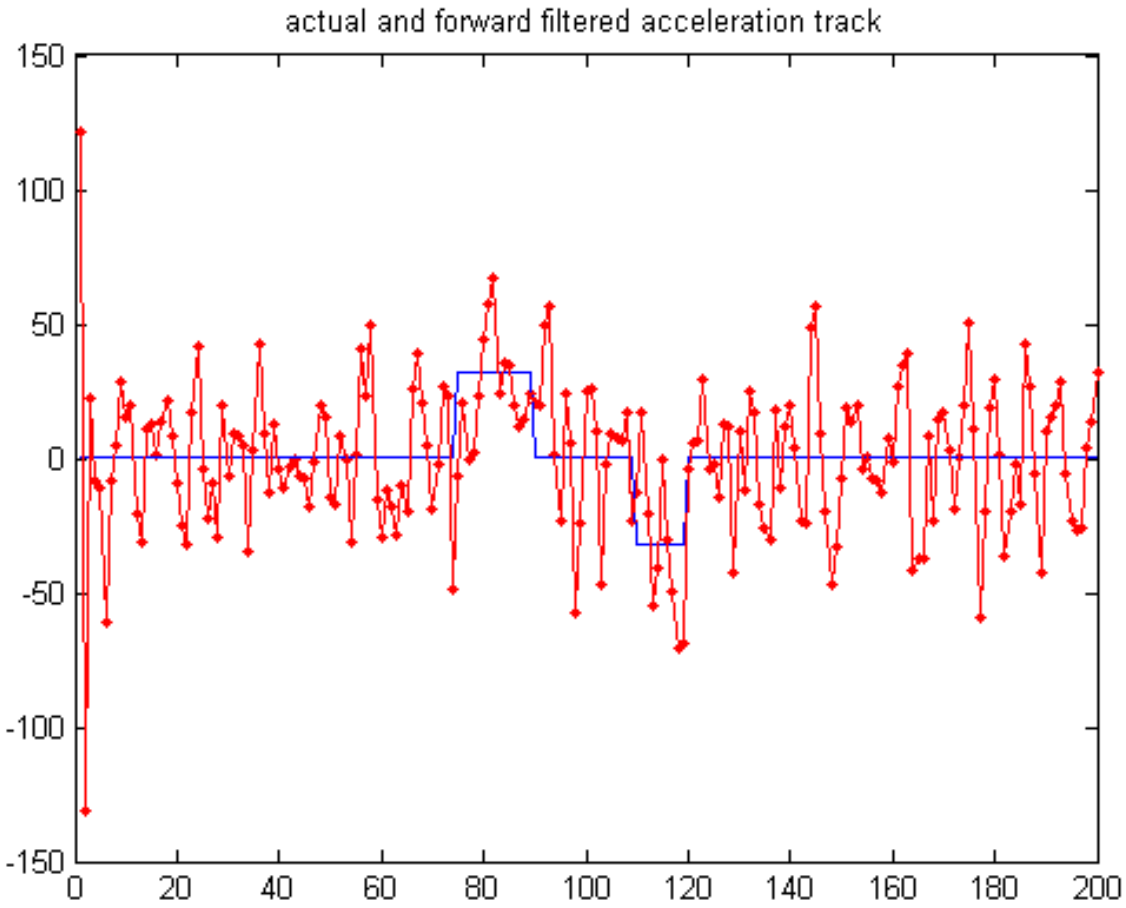
Table VI-a: Simulation Characteristics

time(sec)	maneuver
0	v_o = 500 ft/sec
75	apply +1.0 g acceleration
90	remove the 1.0 g acceleration
110	apply -1.0 g acceleration
120	remove the -1.0 g acceleration
track period, T:	1 sec.
measurement noise, σ_n:	100 feet
maneuver noise, σ_w:	1 g (32.2 ft/sec^2)

Forward Velocity Track



Forward Acceleration Track



Forward Statistics
Prediction and Correction

T = 1

R =10000 = 100^2 sigma_n^2

wn =1.0368e+003 = 32.2^2 sigma_omega^2

K =

0.7460

0.4921

0.1623

PC =

1.0e+003 *

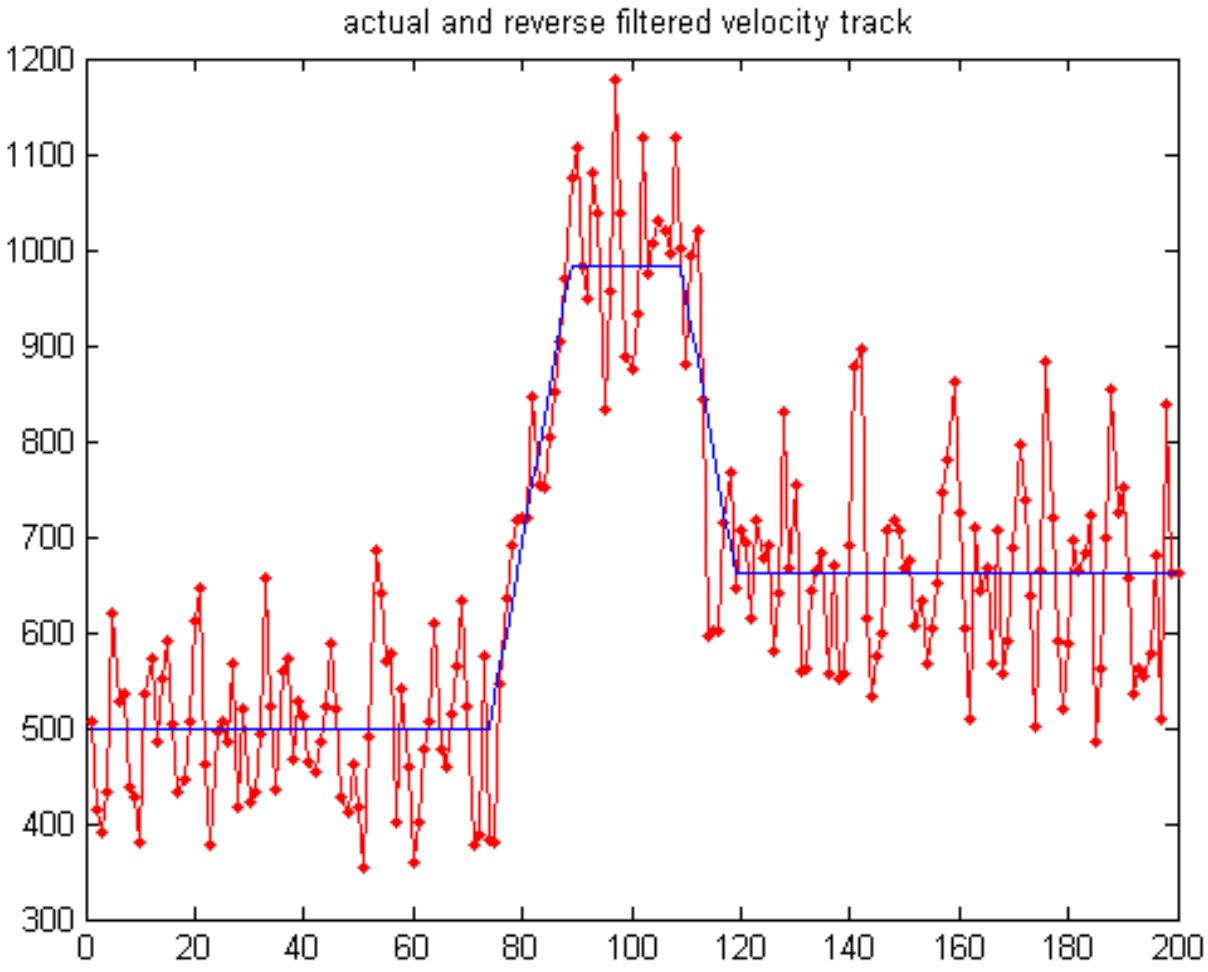
7.4600	4.9206	1.6228
4.9206	6.4657	3.1943
1.6228	3.1943	2.1070

PP =

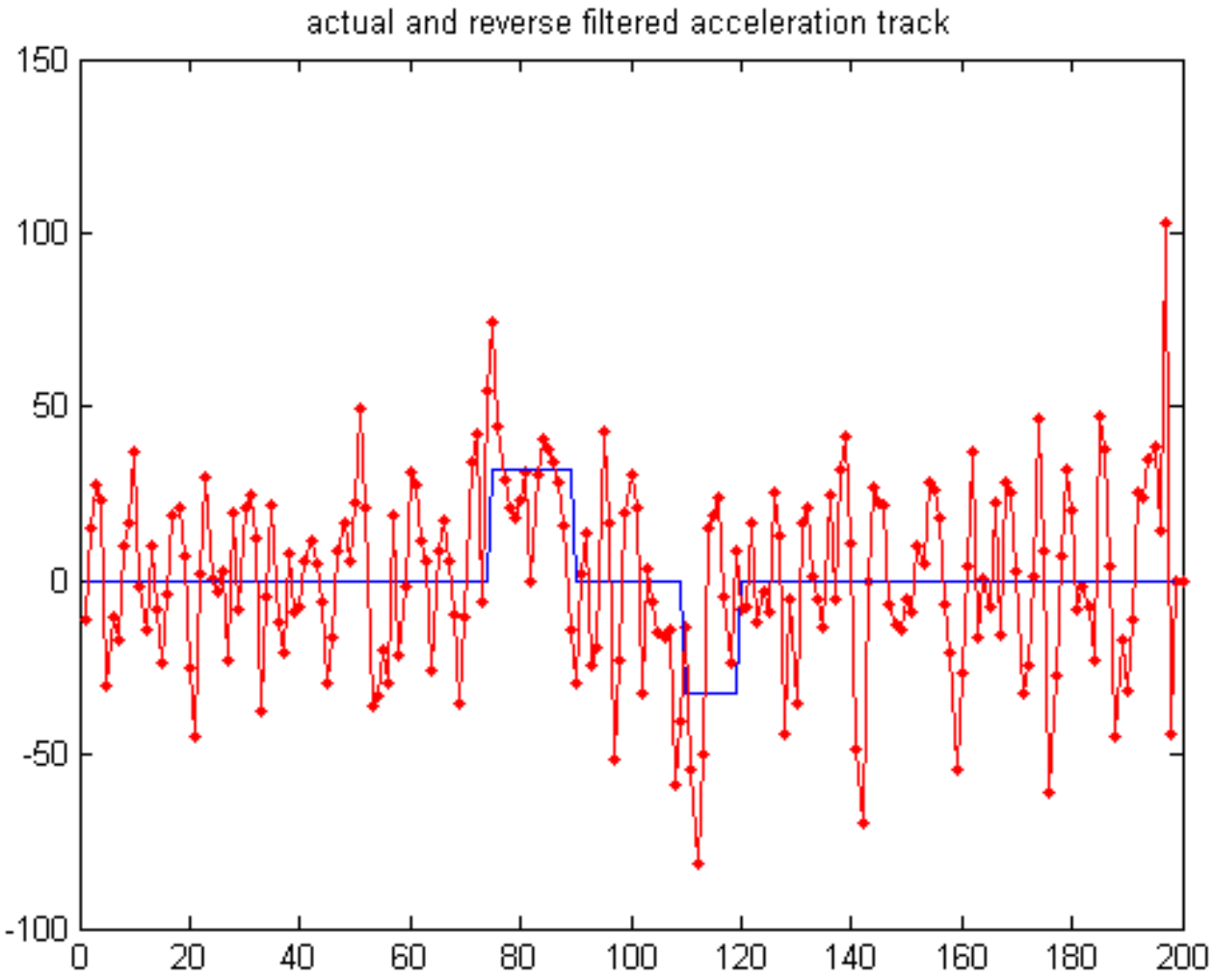
1.0e+004 *

2.9370	1.9373	0.6389
1.9373	1.5998	0.6338
0.6389	0.6338	0.3144

Reverse Velocity Track



Reverse Acceleration Track



Reverse Statistics
Prediction only

T = 1

R =10000 = 100^2 sigma_n^2

wn =1.0368e+003 = 32.2^2 sigma_omega^2

PPr =

1.0e+004 *

2.9370	-1.9373	0.6389
-1.9373	1.5998	-0.6338
0.6389	-0.6338	0.4181

Kpr =

1.3192

-0.6543

0.1623

Combiner Statistics

PS3 = smoother performance formula

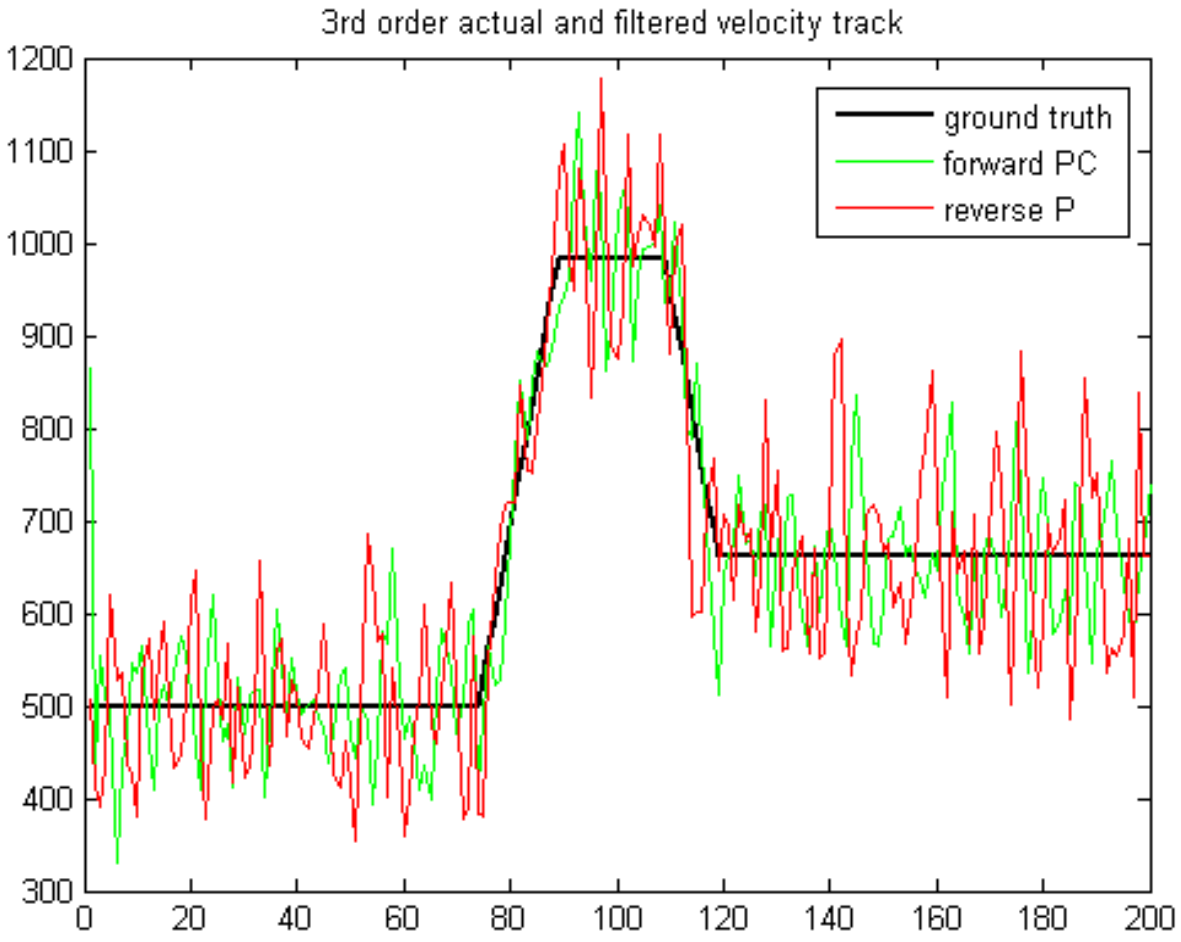
1.0e+003 *

2.2814	0	-0.4963
0	0.6174	0.2423
-0.4963	0.2423	0.4845

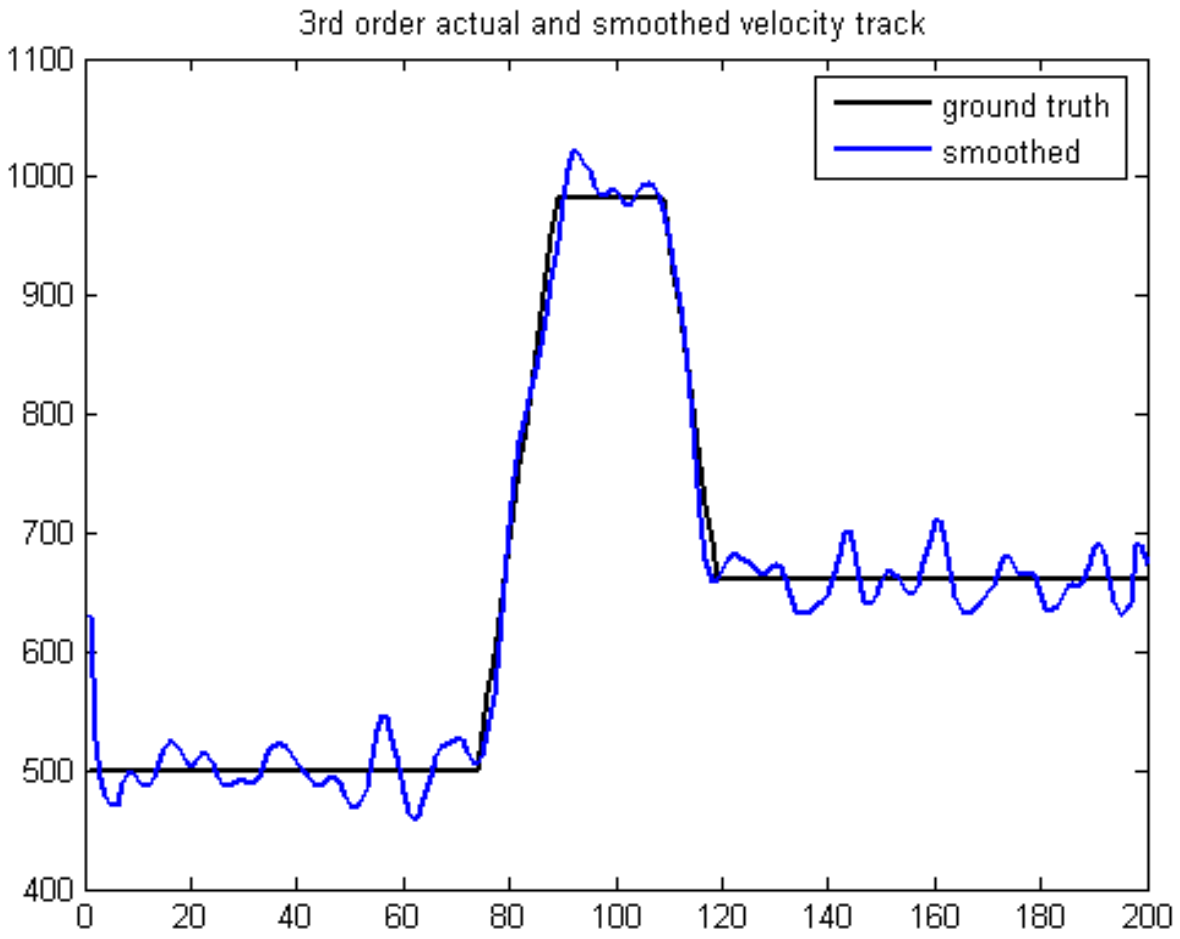
combiner matrix

0.6141	-0.4673	0
-0.2281	0.5000	-0.4673
-0.0496	-0.2281	0.6141

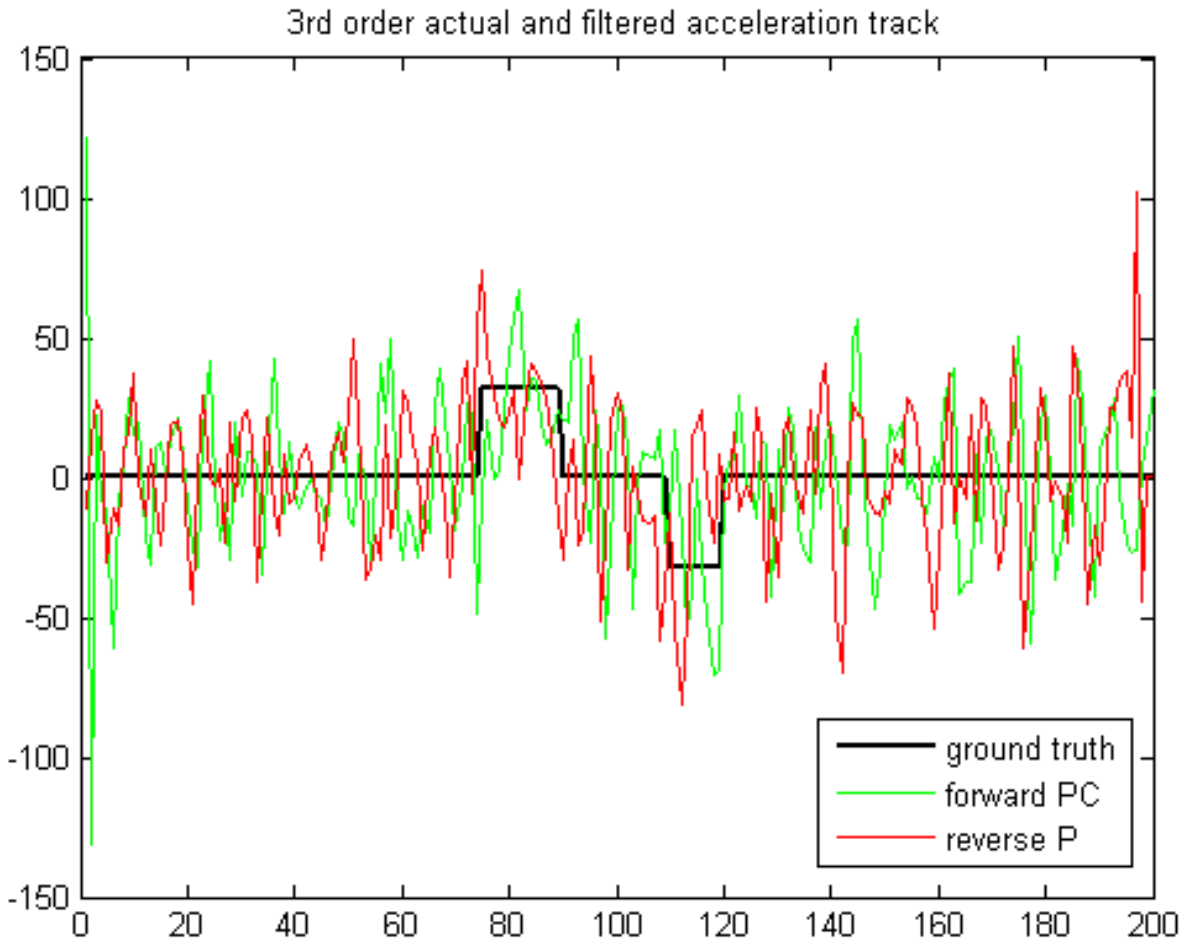
Overlay Forward and Reverse Velocity Tracks



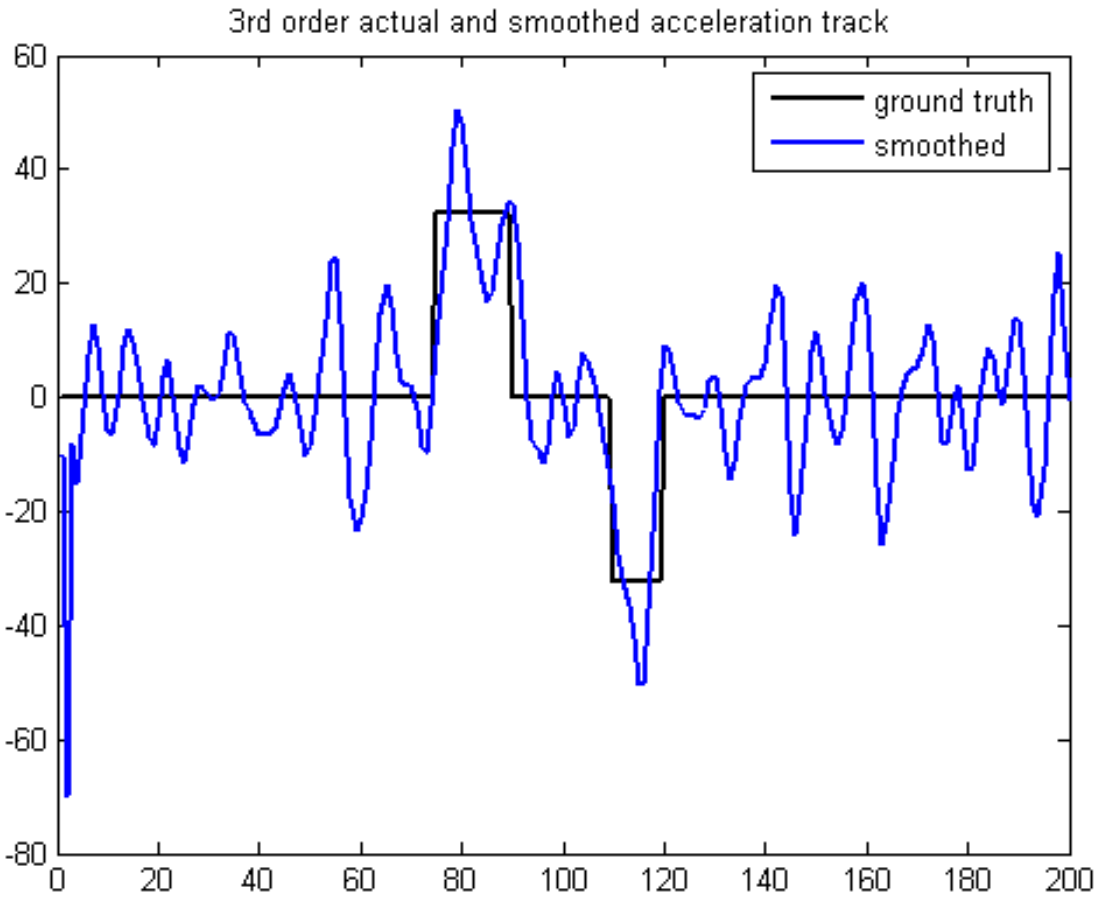
Smoothed Velocity Track



Overlay Forward and Reverse Acceleration Tracks



Smoothed Acceleration Track



Performance Summary

Table VII-a 3rd order Tracking/Smoothing Parameters

Tracking Index: $\Lambda = 0.3220$

Tracking/Prediction parameters:

$\alpha = 0.7460$	$\alpha_p = 1.3193$
$\beta = 0.4921$	$\beta_p = 0.6544$
$\gamma = 0.3246$	$\gamma_p = 0.3246$

Smoothing combination gain matrix:

$$C = \begin{bmatrix} 0.6141 & -0.4673 & 0 \\ -0.2281 & 0.5000 & -0.4673 \\ -0.0496 & -0.2282 & 0.6141 \end{bmatrix}$$

Table VII-b $\alpha-\beta-\gamma$ Performance Summary

	Filter	**Predictor-r**	**Smoother**
$\sigma^2_{\tilde{x}}$	7,460	29,370	2,281
$\sigma^2_{\tilde{x}\tilde{v}}$	4,921	-19,374	0
$\sigma^2_{\tilde{v}}$	6,467	16,001	618
$\sigma^2_{\tilde{x}\tilde{a}}$	1,623	6,390	-496
$\sigma^2_{\tilde{v}\tilde{a}}$	3,195	-6,339	242
$\sigma^2_{\tilde{a}}$	2,107	4,182	485

Application of Kalman Filters

- **Definitions, Equations and Block Diagrams**
- **Target Tracking Example 2nd order**
- **Extended Kalman Filter**
- **Smoothing**
- **Code Examples/demos**

END

End of deck