Kalman Filter and its Applications

Semester Project
Statistical Methods 4

Indian Statistical Institute, Kolkata

Spring, 2022

Srijan Chattopadhyay (BS2126)
Swapnaneel Bhattacharya(BS2105)
Sevantee Basu(BS2129)
Rudrashis Bardhan(BS2118)

Table of Contents

Introduction

- One of the biggest challenges of tracking and control systems is providing an accurate and precise estimation of the hidden states in the presence of uncertainty.
- In GPS receivers, the measurement uncertainty depends on many external factors, such as thermal noise, atmospheric effects, and slight changes in satellite positions.
- The Kalman Filter produces estimates of hidden variables based on inaccurate and uncertain measurements. Also, the Kalman Filter predicts the future system state based on past estimations.
- Today the Kalman filter is used in target tracking (Radar), location and navigation systems, control systems, computer graphics, and much more.

Let us Start!

An Example, $\alpha - \beta - \gamma$ Filter

The situation

Let's begin with an example. Say an aeroplane is moving in one dimension in a straight line with some velocity and constant acceleration, and we are getting measurements about its location, velocity, acceleration, etc, but there are certain **measurement noises**. Also, the (deterministic) equation of motion(Newton) may not be strictly applicable due to **process noise** like air turbulence, air, etc. We will be counting all those errors in this problem.

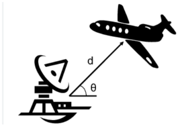


Figure: Aeroplane

Some Notations

Let us first introduce some notation for this problem first:

- $\hat{x}_{n+1,n}$ is the prediction of location of (n+1)th stage by nth measurement.
- $\hat{x}_{n,n}$ is the estimate of the location of nth stage after getting nth update.
- z_n is the nth measurement, $\dot{x}$ and $\ddot{x}$ of corresponding indices represent velocity and acceleration respectively.

State Extrapolation equation

By equation from the Law of Motion,

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\triangle t + \hat{\ddot{x}}_{n,n}\triangle \frac{t^2}{2}$$
$$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\triangle t$$
$$\hat{\ddot{x}}_{n+1,n} = \hat{\ddot{x}}_{n,n}[\text{as acceleration is constant}]$$

These sets of equations are called prediction **State Extrapolation equation**s, as it is used to predict.

State update Equations

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha(z_n - \hat{x}_{n,n-1})$$

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta\frac{z_n - \hat{x}_{n,n-1}}{\triangle t}$$

$$\hat{\ddot{x}}_{n,n} = \hat{\ddot{x}}_{n,n-1} + \gamma\frac{z_n - \hat{x}_{n,n-1}}{0.5\triangle t^2}$$

These are called **State update Equation**s, because from (n-1)th update, we assigned some values for nth step and we are updating it again by nth new measurement, we assume linearity between the predicted and new estimated one with the new measurement.

Increasing the Dimension!!

Multidimensional Kalman Filter

State Extrapolation Equation

Most real-life examples are of higher dimensions, hence let's move into the multidimensional version of Kalman filter. Let us consider the previous example once again in space this time, i.e. the location is a 3-dimensional vector (x, y, z). By the equation of laws of motion:

$$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_{n,n}$$

where, $\hat{x}_{n+1,n} = [\hat{x}_{n+1,n}, \hat{y}_{n+1,n}, \hat{z}_{n+1,n}, \hat{\dot{x}}_{n+1,n}, \hat{\dot{y}}_{n+1,n}, \hat{\dot{z}}_{n+1,n}]^T$

$$\hat{x}_{n,n} = [\hat{x}_{n,n}, \hat{y}_{n,n}, \hat{z}_{n,n}, \hat{\dot{x}}_{n,n}, \hat{\dot{y}}_{n,n}, \hat{\dot{z}}_{n,n}]^T$$

$$u_{n,n} = [\ddot{x}_n, \ddot{y}_n, \ddot{z}_n]^T$$

State Extrapolation Equation

$$F = \begin{bmatrix} 1 & 0 & 0 & \triangle t & 0 & 0 \\ 0 & 1 & 0 & 0 & \triangle t & 0 \\ 0 & 0 & 1 & 0 & 0 & \triangle t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 0.5\triangle t^2 & 0 & 0 \\ 0 & 0.5\triangle t^2 & 0 \\ 0 & 0 & 0.5\triangle t^2 \\ \triangle t & 0 & 0 \\ 0 & \triangle t & 0 \\ 0 & 0 & \triangle t \end{bmatrix}$$

State Extrapolation Equation

Hence we now have some (minimal) idea about how the prediction equation (State Extrapolation equation) will look like in multivariate case:

$$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + G\hat{u}_{n,n} + w_n$$

$\hat{x}_{n+1,n}$ is a predicted system state vector at time n+1
$\hat{x}_{n,n}$ is the estimated system vector at time n
$\hat{u}_{n,n}$ is a control variable or input variable
w_n is a process noise or disturbance
F is a state transition matrix G is a control matrix or input transition matrix(mapping control to state variables)

Covariance Extrapolation Equation

From the state extrapolation equation, we get the following expression for covariance in the prediction of (n+1)th stage($P_{n+1,n}$).

$$\begin{aligned} & P_{n+1,n} \\ = & Cov(F\hat{x}_{n,n}) + Cov(G\hat{u}_{n,n}) + Cov(w_n) \\ = & FP_{n,n}F^T + Q_n \end{aligned}$$

Covariance Extrapolation Equation

Also, there can be some measurement error, if the measurement equation is $z_n = Hx_n + v_n$, where z_n is the measurement vector, x_n is the true system vector and v_n is the random noise and H is the observation matrix, R_n is the covariance matrix of the measurement noise and equal to $E(v_n v_n^T)$

State Update Equation

Extending the similar equations of one dimension, the state update equation in the matrix form is given by the following:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - H\hat{x}_{n,n-1})$$

$\hat{x}_{n,n}$ is an estimated system state vector at time step n

$\hat{x}_{n,n-1}$ is a predicted system state vector at time step n - 1

K_n is called kalman gain

z_n is a measurement

H is an observation matrix

Covariance Update Equation

$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - H\hat{x}_{n,n-1}) = \hat{x}_{n,n-1} + K_n(Hx_n + v_n - H\hat{(}x_{n,n-1})$

Now, e_n

$= x_n - \hat{x}_{n,n}$

$= x_n - \hat{x}_{n,n-1} - K_n(z_n - H\hat{x}_{n,n-1})$

$= x_n - \hat{x}_{n,n-1} - K_n(Hx_n + v_n - H\hat{x}_{n,n-1})$

$= (I - K_n H)(x_n - \hat{x}_{n,n-1}) - K_n v_n$

$P_{n,n} = E(e_n e_n^T)$

$= E(((I - K_n H)(x_n - \hat{x}_{n,n-1}) - K_n v_n)((x_n - \hat{x}_{n,n-1})^T (I - K_n H)^T - (K_n v_n)^T))$

Covariance Update Equation

$= (I - K_n H) E((x_n - \hat{x}_{n,n-1})(x_n - \hat{x}_{n,n-1})^T)(I - K_n H)^T +$

$K_n E(v_n v_n^T) K_n^T$[Assuming v_n to be uncorrelated with others]

$= (I - K_n H) P_{n,n-1} (I - K_n H)^T + K_n R_n K_n^T$

Hence the Covariance update equation is:

$$\boxed{P_{n,n} = (I - K_n H) P_{n,n-1} (I - K_n H)^T + K_n R_n K_n^T}$$

Kalman Gain

We desire our filter to be optimal in the sense of variation, i.e. confidence of estimation(or, prediction). Thus, we want a Kalman Gain that minimizes the variance of the estimate. To minimize the variance, we need to minimize the main diagonal of the covariance matrix $P_{n,n}$, i.e. $tr(P_{n,n})$. So we differentiate it w.r.t. K_n and set it 0.
We got the result in the last slide:
$P_{n,n} =$
$P_{n,n-1} - P_{n,n-1}H^T K_n^T - K_n H P_{n,n-1} + K_n(HP_{n,n-1}H^T + R_n)K_n^T$
Hence, $tr(P_{n,n}) =$
$tr(P_{n,n-1}) - 2tr(K_n H P_{n,n-1}) + tr(K_n(HP_{n,n-1}H^T + R_n)k_n^T)$

Kalman Gain Derivation

Now, from multivariate calculus, we know $\frac{d}{dA}(tr(AB)) = B^T$ and

$$\frac{d}{dA}(tr(ABA^T)) = 2AB$$

So, differentiating,

$\frac{d}{dK_n}(tr(P_{n,n})) = 0 - 2(HP_{n,n-1})^T + 2K_n(HP_{n,n-1}H^T + R_n) = 0$

$$(HP_{n,n-1})^T = K_n(HP_{n,n-1}H^T + R_n)$$

$$K_n = (HP_{n,n-1})^T(HP_{n,n-1}H^T + R_n)^{-1}$$

Hence, the Kalman Gain is:

$$\boxed{K_n = P_{n,n-1}H^T(HP_{n,n-1}H^T + R_n)^{-1}}$$

A Simpler Formula

And, putting the value, simplified covariance update equation will be:

$$\boxed{P_{n,n} = (I - K_n H) P_{n,n-1}}$$

Summary Till Now

- **State Extrapolation Equation(Prediction Equation):**$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n + w_n$
- **Covariance Extrapolation:**$P_{n+1,n} = FP_{n,n}F^T + Q$
- **State Update Equation(Update/Correction Equation):**$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - H\hat{x}_{n,n-1})$
- **Covariance Update Equation:** $P_{n,n} = (I - K_nH)P_{n,n-1}(I - K_nH)^T + K_nR_nK_n^T$
- **Kalman Gain:**$K_n = P_{n,n-1}H^T(HP_{n,n-1}H^T + R_n)^{-1}$
- **Measurement Equation:**$z_n = Hx_n + v_n$
- **Measurement Error:** $R_n = E(v_nv_n^T)$
- **Process Noise Uncertainty:** $Q_n = E(w_nw_n^T)$

But....How to Apply All These!!

Applications

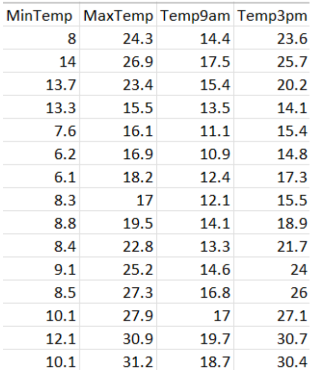
- Weather Prediction
- Stock Price Prediction
- Covid Cases Prediction
- GPS/Position Tracking

Application I

Weather (Most of the time feels like temperature) Prediction

The Scenario

We have 4 types of measures of temperature in our data:



MinTemp	MaxTemp	Temp9am	Temp3pm
8	24.3	14.4	23.6
14	26.9	17.5	25.7
13.7	23.4	15.4	20.2
13.3	15.5	13.5	14.1
7.6	16.1	11.1	15.4
6.2	16.9	10.9	14.8
6.1	18.2	12.4	17.3
8.3	17	12.1	15.5
8.8	19.5	14.1	18.9
8.4	22.8	13.3	21.7
9.1	25.2	14.6	24
8.5	27.3	16.8	26
10.1	27.9	17	27.1
12.1	30.9	19.7	30.7
10.1	31.2	18.7	30.4

Figure: Weather Data

Generating the measurement of mostly felt temperature

Assumption:

- We have made an assumption, that the measurement of the temperature that we feel most of the time a day is a convex combination of the above four measurements. That is, $a_i = \alpha_i b_i + \beta_i c_i + \gamma_i d_i + \delta_i e_i$,and $\alpha_i + \beta_i + \gamma_i + \delta_i = 1$ We have randomized the selection of the coefficients. We have generated $\alpha_i, \beta_i, \gamma_i$ iid from $U(0, \frac{1}{3})$ and $\delta_i = 1 - \alpha_i - \beta_i - \gamma_i$. Then, we got the measurement of the most felt temperature.
- Now we will apply the Kalman filter on it, to predict the most frequent actual temperature of that day and of the next day. Assuming the temperature curve to be continuous and differentiable, we assume $F = 1$ here and apply the One-dimensional Kalman filter here.

Graphical Representation

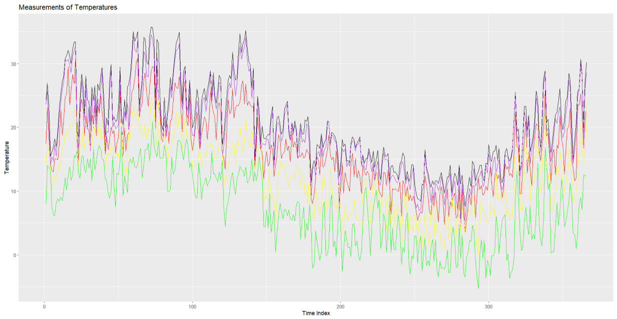


Figure: Maximum, Minimum, at 3 P.M., at 9 A.M., Most felt Temperature (daywise)

Results of Kalman filter (graphically)

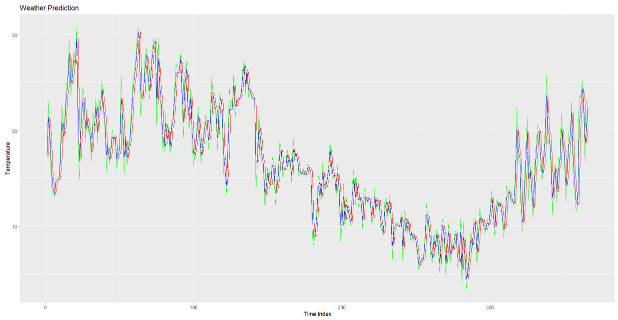


Figure: Measurement of, Forecasted(future), Predicted(Today's Actual
Most of the time feels like temperature

Animation(Screenshots)

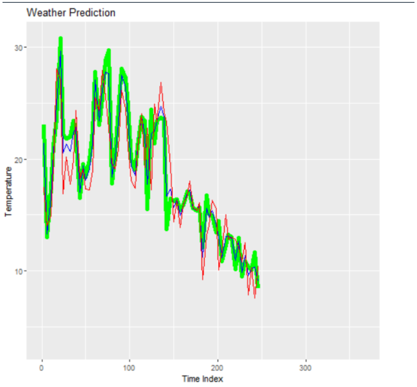


Figure: Animation-Take 1

Animation(Screenshots)

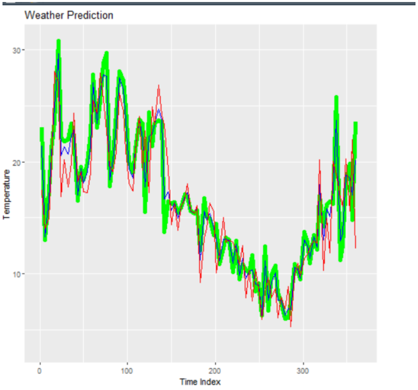


Figure: Animation-Take 2

Application II

Prediction of Stock Market Prices

A Few Definitions

- In a model of stock prices, the **drift** refers to the **average rate of return on the stock over time**(μ) and it determines the **slope** of the expected price increase over time. The initial drift is taken to be 0.05.
- **Volatility** refers to the amount of **random fluctuation** or **uncertainty** in the stock's price movements over time(σ) and it determines the **standard deviation** of the random fluctuations in the stock's returns.The initial volatility is taken to be 0.2..

Drift and Volatility

- The **time horizon** is the number of trading days in a year which is 252.
- The **time steps** is the discrete intervals of time at which the stock prices are recorded or modeled. For example, if we are modeling daily stock prices, each time step would correspond to one trading day. If we are modeling hourly stock prices, each time step would correspond to one hour. The **time steps** is taken to be 200.

Simulating The model

- We generated 200 samples from $N(\frac{0.05}{252}, \frac{0.2}{\sqrt{252}})$.
- In the model, $measurement_{model}$ refers to the scaling factor that relates true stock prices to observed prices. However, in practice, there may be some noise that causes the observed and true prices to differ and that is accounted by $measurement_{error}$.
- $System_{dynamics}$ is defined as a list with two elements: F and Q. Here $F = 1$. This is appropriate for modeling systems where the underlying state does not change significantly over short periods of time, such as stock prices over a single trading day.
- $Q = 0.05$. In this implementation, the process noise is assumed to be Gaussian with zero mean and a constant variance over time.

Graphical Representation

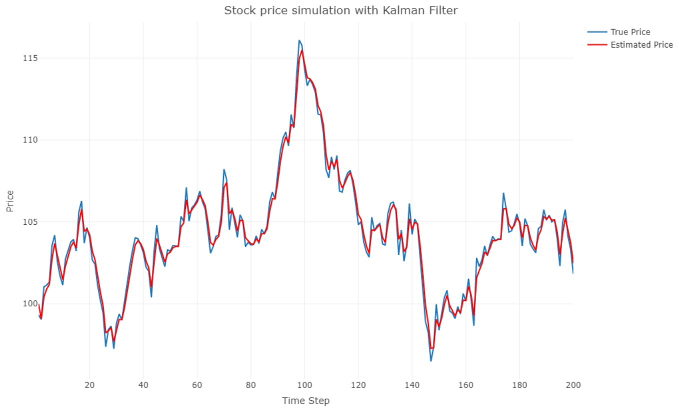
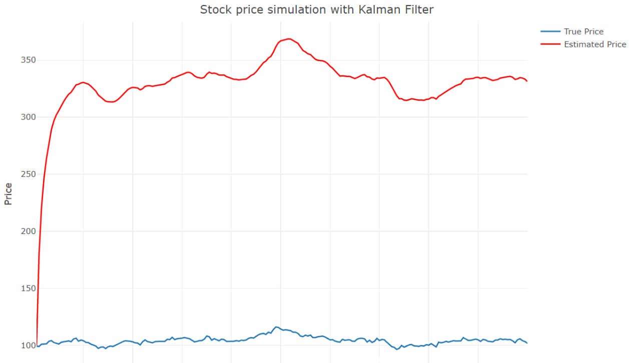


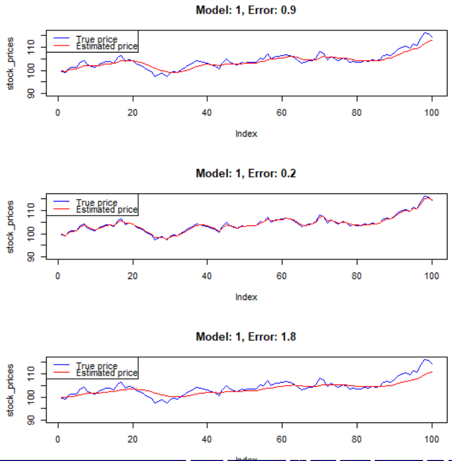
Figure: True Price and Estimated Price

Influence of measurement errors and measurement model

An essential aspect is in our model, the measurement error is taken to be 0.2 and the measurement model is taken to be 1. We make some changes in the following and observe the results.



Measurement model and error variations



Animation(Take-I)

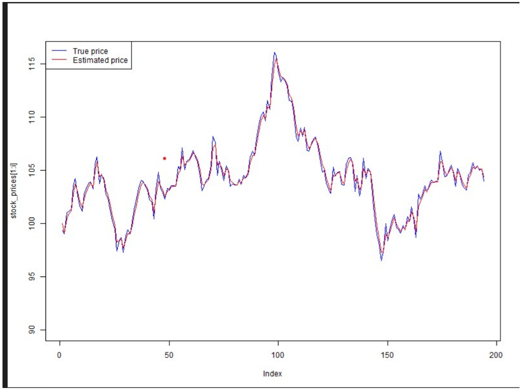


Figure: True Price and Estimated Price

Prediction of stock prices

Using our Kalman filter model, we can predict the price of stock given our initial price. However, it will depend strongly on the $measurement_{error}$ parameter, which indicates the market turmoil and other factors like company news, economic conditions, etc.
Let us give an example:
We wish to predict the price of a stock of **Apple—Inc** based on data from 2018. The following shows the graph corresponding.

Prediction of Apple stock market price

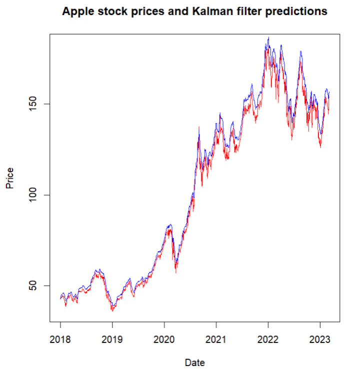


Figure: True Price and Estimated Price

Application III

Prediction of Covid Cases

Simulation of Actual Data

We simulated 10000 cases of covid from $N(50000, 100)$, because it makes sense to predict the number of cases when the average number of cases per day is high.

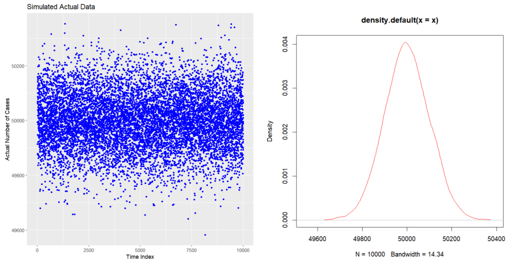
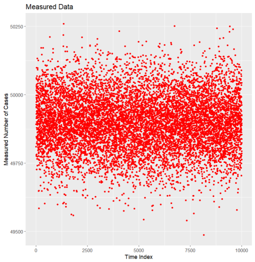


Figure: Scatter-Plot and Density of Simulated Actual Data

Adding Noise

Then we added iid noises from $N(-100, 1)$ to each observation. We deliberately took the mean to be -100, because the number of cases reported is always lower than the actual, and due to the invariance of the collection method, we took the variance less.



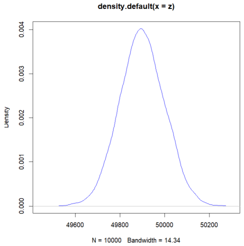


Figure: Measurement of the data(with random noise)

Assumptions

We took F = 1 here because we assumed that the increment(or, decrement) is linear and continuous, i.e. the number of cases won't drastically increase(or decrease). We took Q=1 here, as the error has a variance of 1, and then applied the one-dimensional Kalman Filter here.

Graphical Representation

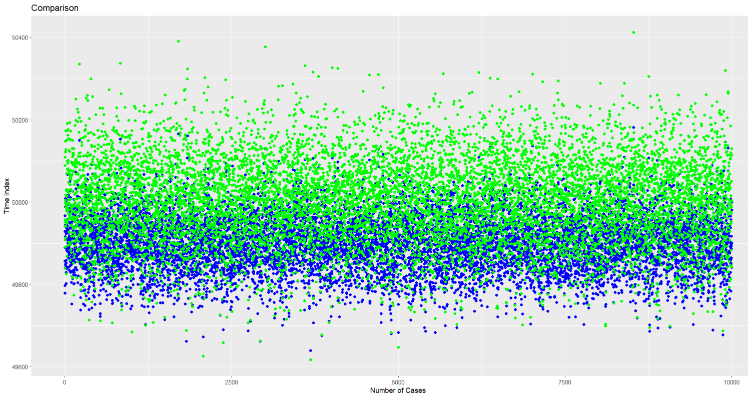


Figure: Actual and Predicted from the Measurement

Graphical Representation

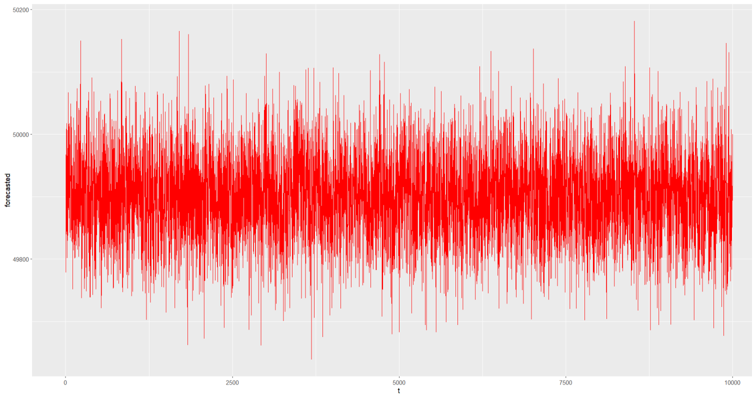


Figure: Forecasted number of cases of next day

Animation Screenshots

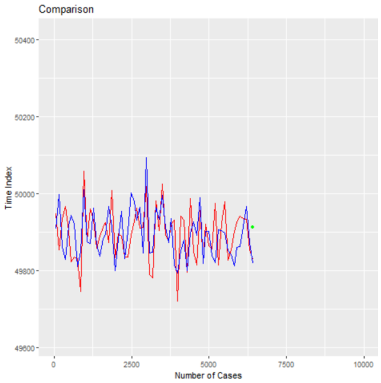


Figure: Animation Take 1

Animation Screenshots

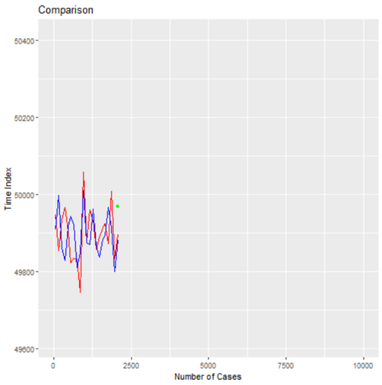


Figure: Animation Take 2

Application IV

GPS/Position Tracking

The Scenario

Let us consider our last(Simulated) example. A particle moves by the formula $x(t)^2 + y(t)^2 = 1$. So, its time equation can be parameterized as $x = \sin t, y = \cos t$, Accordingly $\dot{x} = \cos t, \dot{y} = -\sin t$.

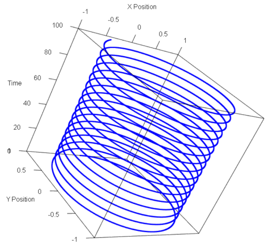


Figure: True Position of the particle

Measurement got from the instrument

Then we assumed the measurement error to be normally distributed with mean 0 and sd=0.1.

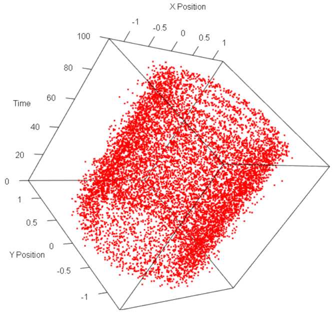


Figure: Position determined by instrument(i.e., with measurement error)

Determining state equation

Then we must determine F, i.e. the deterministic equation of $x_{n+1,n}$ on $x_{n,n}$.
$x(t+\alpha) - x(t) = \sin(t+\alpha) - \sin t = \alpha \cos(t+\epsilon_\alpha) \approx \alpha \cos t = \alpha \dot{x}(t)$[by LMVT and if α is small].
Similarly,
$\dot{x}(t+\alpha) - \dot{x}(t) = -\alpha x(t)$
$y(t+\alpha) - y(t) = \alpha \dot{y}(t)$
$\dot{y}(t+\alpha) - \dot{y}(t) = -\alpha y(t)$

Deterministic matrices

Hence our deterministic matrix F is:

$$F = \begin{bmatrix} 1.00 & dt & 0.00 & 0.00 \\ -dt & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & dt \\ 0.00 & 0.00 & -dt & 1.00 \end{bmatrix}$$

Here H is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

For simplicity, we only took up to velocity, So $G = 0$

X position comparison

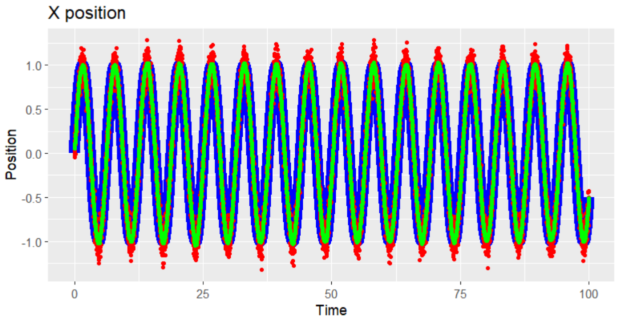


Figure: X-Position(Actual Position, Position with measurement noise, Predicted Position)

Y position comparison

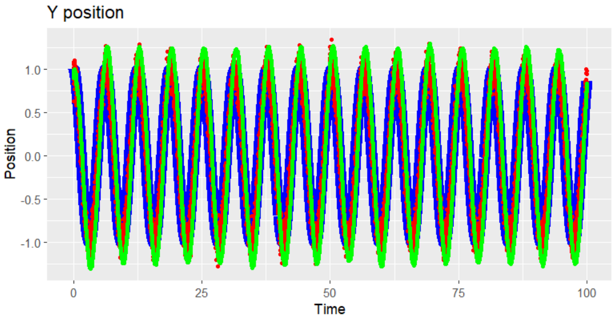


Figure: Y-Position(Actual Position, Position with measurement noise, Predicted Position)

3D Plot, View 1

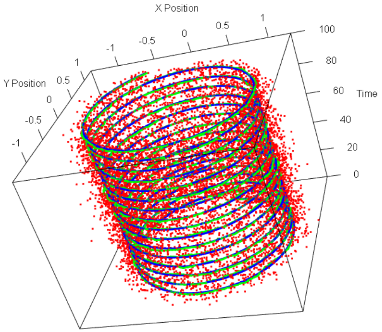


Figure: 3D plot of Actual Position, Position with measurement noise, Predicted Position in X,Y,Time axes(View-1)

3D Plot, View 2

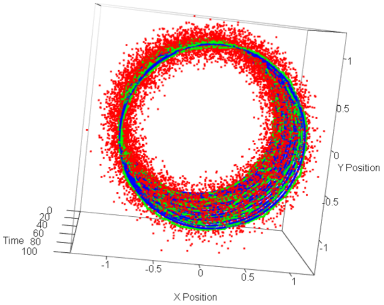


Figure: 3D plot of Actual Position, Position with measurement noise, Predicted Position in X,Y,Time axes(View-2)

3D Plot, View 3

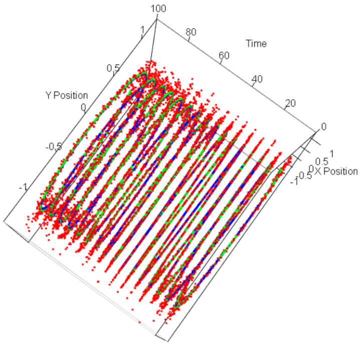


Figure: 3D plot of Actual Position, Position with measurement noise, Predicted Position in X,Y,Time axes(View-3)