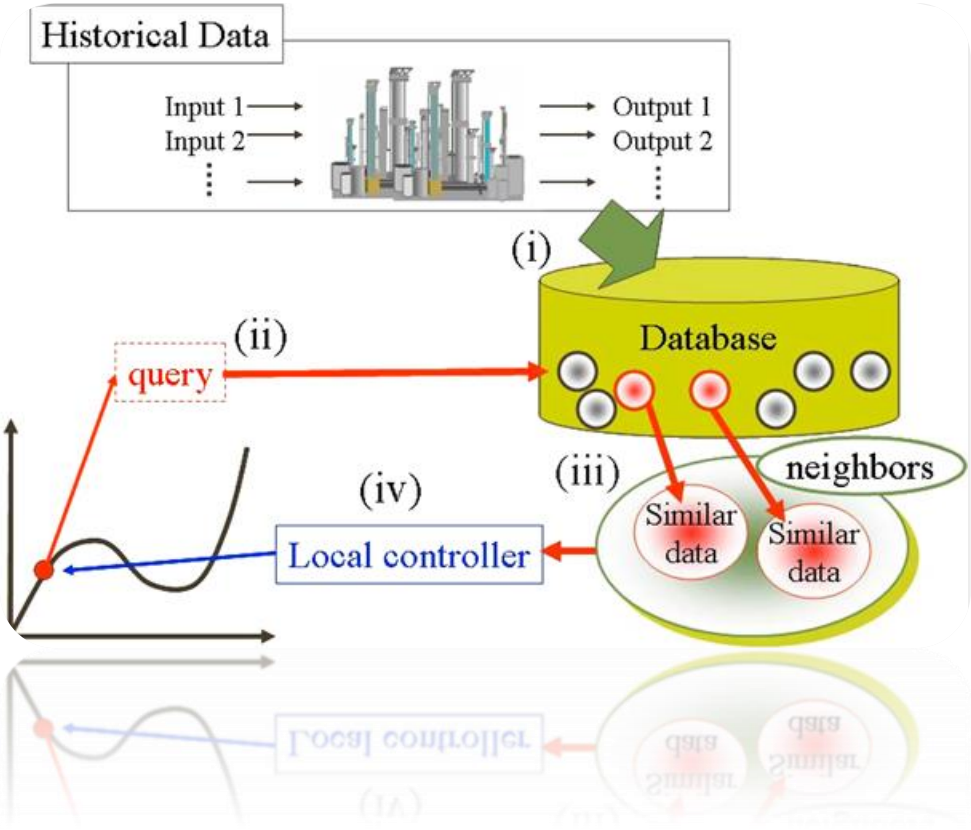
ENPM 667: CONTROL OF ROBOTIC SYSTEMS

PROJECT 1

TOPIC: STUDY ON KALMAN FILTER BASED PID CONTROLLER



Aditya vaishampayan | UID: 116077354

adityav@terpmail.umd.edu | +1-(240)-743-0530

CONTENTS

Introduction

This study proposes a self-tuning proportional-integral-derivative（PID) controller design method based on a Kalman filter. Recently, data driven controllers have gained much attention because of their ability, to directly tune the control parameters using closed loop data, and without obtaining the model of the system.

Many methods that are available, that largely depend on the model of the system to accurately describe the system properties. Thus, to construct a highly accurate model, it becomes mandatory to identify system parameters using the input output data. However, it isn't possible to add test signals every time in an industrial setting. Thus, it becomes very essential to directly tune PID parameters using the input and output data. In this paper, a PID control scheme is proposed that doesn't take into consideration any descriptive model. The scheme suggests that by using input, output closed loop data and some user defined parameters, the PID control parameters can be obtained.

Some systems tend to have fixed PID gains, but in industrial settings where the output of the system changes every hour, an online parameter tuning method is required. It has been known that in systems that have continuously varying output, fixed PID gains tend to destabilize the system hence, online parameter tuning helps us obtain an optimized output of the system/plant.

The author suggests that if the output of the system is made to track the desired output obtained from another system as a reference, then we can calculate the control parameters directly by studying the input output closed data of the system, and thus we can completely avoid system model identification and hence can significantly reduce the cost. Our method searches for the global optimum of the desired criteria, and thus we can achieve a good controller approximation from the restricted complexity global optimal controller.

In order to estimate the PID controller gains, the paper uses a Kalman Filter. As the Kalman filter is universally well-known algorithm for prediction of a state given its earlier values and the current measurement, it serves a s a very hand tool to our purpose. Also, it is very robust in nature as it gives very reliable estimates in noisy environments too. Hence, we use a confluence of the PID controller and Kalman filter to perform system parameter estimation.

This report starts with a basic general introduction of the concepts involved such as Optimization and (DD) data driven controller. Then the report delves deeper by explanation of the velocity PID algorithm and Kalman Filter. Later the Kalman filter equations and the equations given by the author have been compared and how the data from the graphs has been obtained is explained. In the end other data driven parameter estimation techniques have been portrayed and the possible applications of the paper are given. The report finally concludes by giving a summary of the entire data driven controller parameter estimation technique.

WHAT IS A DATA DRIVEN CONTROLLER?

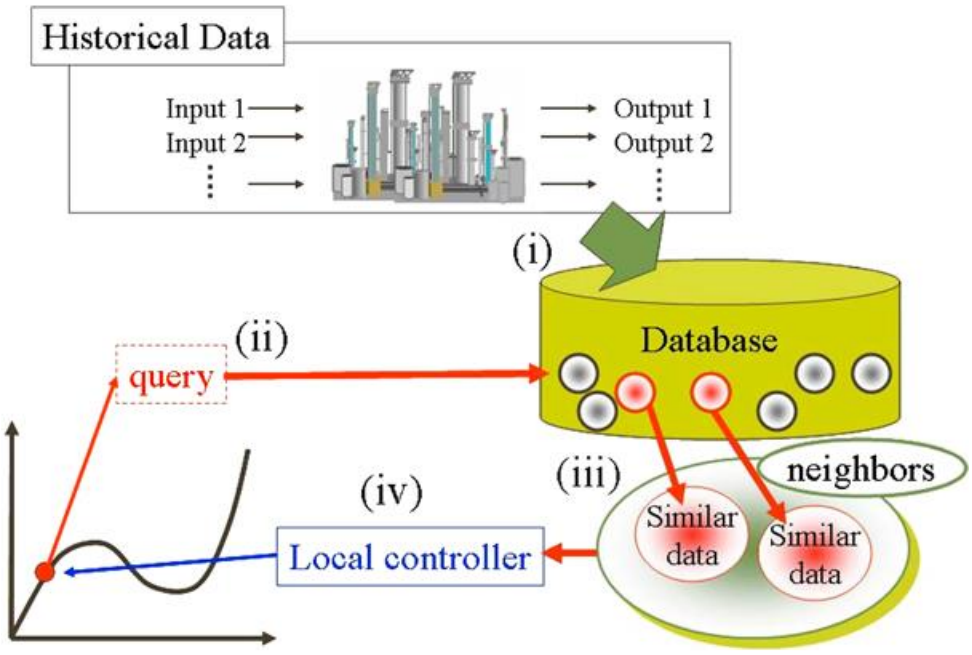


Fig 1: Pictorial diagram of a data driven controller.

Many systems use model-based controllers which perform through system identification based on operating data. However, it is not easy to obtain the control parameters every time in a real time system just by obtaining the system model because if the system models are not described accurately, the desired control performance cannot be obtained. It is very essential to determine the model with a very high degree of accuracy and to estimate the controller gains and the obtained gains are adjusted multiple times to find the optimal solution.

On the other hand, if we use data driven schemes, we can reduce the temporal and monetary costs because no system identification is done here. Generally, the system characteristics often change due to variations in environmental and operating conditions. By having fixed PID control parameters, sometimes we cannot obtain desired characteristics and the systems output becomes unstable if the initial control parameters are applied to all the situations.

As seen from the pictorial representation above, data driven techniques rely on historical data to obtain a database of parameters which it later uses to tune the local controller. The output of the system is taken and then it is compared with the historical data present in the database. The necessary corrections are made and then again, the new set of values is given to the local controller. This process goes on and on till the desired output is obtained.

For creating the Data Driven (DD) technique, we undertake the following mathematical consideration.

We assume a discrete time nonlinear system of the form $y(t) = f\big(\phi(t-1)\big)$ where y(t) denotes the system output and f (•) denotes the nonlinear function. Thus $\phi(t-1)$ is called the information vectors and is defined by the equation

$$\phi(t-1) \coloneqq \big[y(t-1), \dots ., \quad t\big(t-n_y\big), u(t-1), \dots \dots u(t-n_u)\big]$$

Here u(t) denotes the control input, n_y = order of system output and n_u = order of the system input. We also assume that the sign of partial derivative of $y(t)$ with respect to $u(t-1)$ is known.

The velocity type PID control law is taken which is derived later in the paper.

$$\triangle U(t) = K_i e(t) - K_p \triangle y(t) - K_D \triangle^2 y(t)$$

Here e(t) is the error between the reference output i.e. and the obtained system output. Thus, we can write the system as

$$u(t) = g\left(\acute{\phi}(t)\right)$$

Where, $\acute{\phi}(t) \coloneqq [K(t), r(t), y(t), y(t-1), y(t-2), u(t-1)]$ and,

$K(t) \coloneqq [K_P(t), K_I(t), K_D(t)]$

After substituting we get

$$\tilde{\phi}(t) \coloneqq [y(t), \dots, y\big(t-n_y+1\big), K(t), r(t), u(t-1), \dots ., u(t-n_u+1)]$$

Further we can write the PID parameters in the form:

$$\bar{\phi}(t) \coloneqq [y(t+1), y(t) \dots, y\big(t-n_y+1\big), r(t), u(t-1), \dots ., u(t-n_u+1)]$$

However, since we can't obtain the future output at $y(t+1)$ we have to replace it by $r(t+1)$ as it is known to us. Thus $\bar{\phi}(t)$ becomes:

$$\bar{\phi}(t) \coloneqq [r(t+1), r(t) \dots, y(t), u(t-1), \dots ., u(t-n_u+1)]$$

Now we can design a PID control scheme based on the data driven technique. The algorithm of a data driven technique is summarized as follows:

- Data is first stored in the form of the information vector ϕ
- Query of $\phi(t-1)$ is required.
- Similar neighbors to the query are selected from the database.
- Local controller corresponding to the query is designed.

STEP 1: As this is a data driven controller, we first need to store the historical data that has been created by the system to be controlled and then PID gains are calculated by advanced methods such as Zeigler Nicholas or Cohen Coon method.

Hence we can write $\phi(j) \coloneqq [\bar{\phi}(j), K(j)]$, where j = 1, 2,…,N

STEP 2: Next we calculate distance and select neighbors. It is necessary to determine the distance between the query and the information vectors. We calculate as follows:

$$d\left(\overline{\emptyset}(t), \overline{\emptyset}(j)\right) = \sum_{l=1}^{n_y + n_u + 1} \left[\frac{\overline{\emptyset}_l(t) - \overline{\emptyset}_l(j)}{max_m \overline{\emptyset}_l(m) - min_m \overline{\emptyset}_l(m)} \right] \; (j = 1,2, \ldots., N(t))$$

Here $N(t)$ denotes the number of information vectors stored in the database when the query $\overline{\emptyset}(t)$ is given.

$\overline{\emptyset}_l(j) =$ denotes the l[th] element of the j[th] information vector

$\overline{\emptyset}_l(t) =$ denotes the l[th] element of the query at t.

Here k pieces with the smallest distance between them are chosen from all information vectors

STEP 3: COMPUTE PID parameters. Now we use the k-neighbors selected in the previous step and compute the suitable set of PID parameters.

$$K^{old}(t) = \sum_{i=1}^{k} w_i K(i) \,, \sum_{i=1}^{k} w_i = 1$$

Here ω_i denotes the weight corresponding to the i[th] information vector $\overline{\emptyset}(i)$ in the selected neighbors.

STEP 4: In the case where suitable control performance can't be obtained using PID parameters, these control parameters must be updated and stored in the database. We need to adjust the PID parameters in way so that the error is decreased. A steepest descent method is utilized to modify the PID parameters:

$$K^{new}(t) = K^{old}(t) - \eta \frac{\partial J(t+1)}{\partial K(t)}$$

$$\eta \coloneqq diag\{\eta_P, \eta_I, \eta_D\}$$

Here η denotes the learning rate, and the following $J(t+1)$ denotes the error criterion:

$$J(t+1) \coloneqq \frac{1}{2}\epsilon(t+1)^2$$

And $\epsilon(t) \coloneqq y_r(t) - y(t)$

The output of the reference model is denoted by $y_r(t)$, given by

$$y_r(t) = \frac{z^{-1}T(1)}{T(z^{-1})}r(t)$$

$$T(z^{-1}) \coloneqq 1 + t_1 z^{-1} + t_2 z^{-2}$$

Where $T(z^{-1})$ is based in the rise time and the damping coefficient.

Also $T(z^{-1})$ is the denominator of the discrete version of the a desirable second-order continuous time transfer function G(s):

$$G(s) = \frac{1}{1 + \sigma s + \mu(\sigma s)^2}$$

The step response of G(s) shows the Binomial model response and the Butterworth model response, respectively.

Also, each partial differentiation is developed as:

$$\frac{\partial J(t+1)}{\partial K_p(t)} = \frac{\partial J(t+1)}{\partial \epsilon(t+1)}\frac{\partial \epsilon(t+1)}{\partial y(t+1)}\frac{\partial y(t+1)}{\partial u(t)}\frac{\partial u(t)}{\partial K_p(t)} = \epsilon(t+1)(y(t) - y(t+1))\frac{\partial y(t+1)}{\partial u(t)}$$

$$\frac{\partial J(t+1)}{\partial K_I(t)} = \frac{\partial J(t+1)}{\partial \epsilon(t+1)}\frac{\partial \epsilon(t+1)}{\partial y(t+1)}\frac{\partial y(t+1)}{\partial u(t)}\frac{\partial u(t)}{\partial K_I(t)} = -\epsilon(t)e(t)\frac{\partial y(t+1)}{\partial u(t)}$$

$$\frac{\partial J(t+1)}{\partial K_D(t)} = \frac{\partial J(t+1)}{\partial \epsilon(t+1)}\frac{\partial \epsilon(t+1)}{\partial y(t+1)}\frac{\partial y(t+1)}{\partial u(t)}\frac{\partial u(t)}{\partial K_D(t)}$$
$$= \epsilon(t+1)(y(t) - 2y(t-1) + y(t-2))\frac{\partial y(t+1)}{\partial u(t)}$$

STEP 5: A time constraint is present while applying the Data driven technique in real time i.e. the calculations shown in step 2 to step 4 need to be completed within the sampling time. Hence storing the redundant data on the data database wastes a certain amount of the cycle time that could be used on the process. Hence, we need an algorithm to avoid excessive storage of data.

- In information vectors where K neighbors are expected before hand, the information vectors that satisfy the following condition are extracted:

  $$d\left(\overline{\emptyset}(t), \overline{\emptyset}(i)\right) \leq \alpha_1, i = 1,2,3,,,,,,,,N(t) - k$$

  The distance between the vectors is computed only using the input/ output data. Thus, by applying the above condition we ensure that only those vectors who have a shortest distance from the query are extracted.

- From the vectors extracted using first condition only those vectors are further extracted which follow the following condition:

  $$\sum_{l=1}^{3}\{\frac{K_l(i) - K_l^{new}(t)}{K_l^{new}(t)}\}^2 \leq \alpha_2$$

Here K_1, K_2, K_3 are K_P, K_I, K_D

Hence using the above procedure, we can extract and delete the redundant data from the database. In this process those information vectors are extracted and deleted which have high similarity in their newly generated PID gains. If several information vectors exist which satisfy the second condition, then only those vectors are removes which have the smallest value according to the second condition. In practice α_1 and α_2 are generally set between 0.1 and 1 but some trial and error may be necessary.

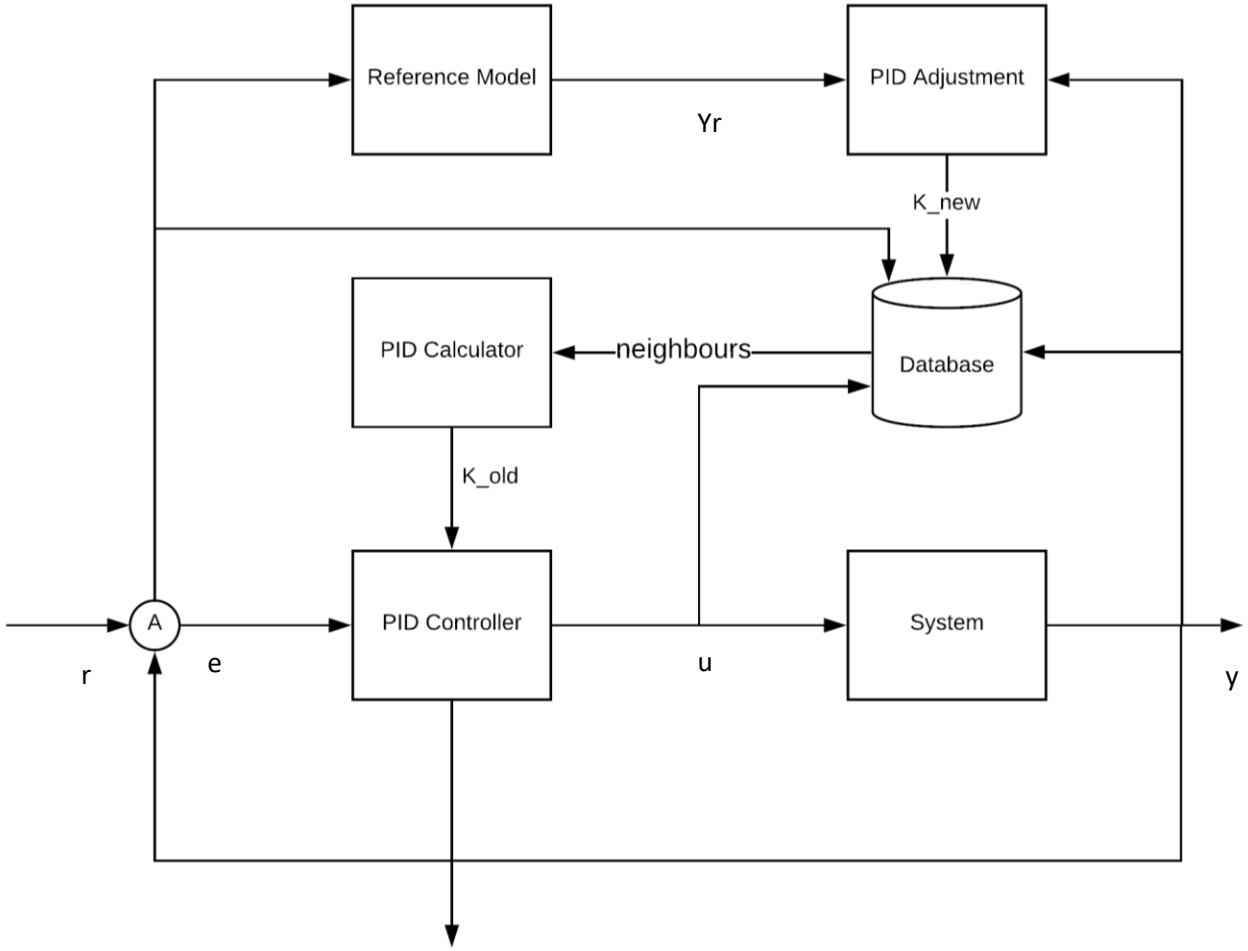


Fig 2: Block diagram of a data driven controller method

From the block diagram above we can summarize that the system output is compared against the reference output and the PID gains adjustment is made accordingly. Further the gains are stored in the data base and the gains which are very close to each other are removed because of redundancy. The new gains are then given to the local controller. The output of the system is taken, and this cycle is repeated.

As we have seen in the above discussion of data driven control, we have realized that to obtain an optimized output we need to perform control parameter optimization. Hence now the report gives a brief introduction about Optimization and the methods that can be used to perform optimization.

What is controller optimization?

Controller optimization is a way of getting the best performance from a controller and for finding its limits with keeping good performance in mind. To get the best desired output of a control system, its gains should be chosen wisely and hence controller parameter optimization turns out to be a key tool. Sometimes it is important to solve a problem optimally and at other times we can get a solution closest to an optimal solution when there are multiple parameters to judge the solution of the problem.

The principle of Optimality states that optimization over time can often be regarded as optimization in stages where the tradeoff is between obtaining the lowest possible cost at present stage by compromising the implication of function cost at a future stage. The best action minimizes the sum of the cost incurred at the current stage and the least total cost that can be incurred from all subsequent stages, consequent on this decision. This is known as the Principle of Optimality.

What is the difference b/w optimization and minimization?

Optimization is the process of finding the best suitable value over a range of possible solutions whereas minimization or maximization finds the minimum or maximum within the possible solutions. The following example shows optimization in lay man's terms.

If we take an example of a company which manufactures ball point pens, then to produce the pens the company needs to acquire materials such as plastic covers and refills. The company also has several other costs such as labor, machinery etc. Hence the total cost is the cumulative sum of costs. We know that the aim of a company must minimize the total costs incurred while maximizing its profits. Thus, to optimize the production means to have minimum total cost and maximum total profit. And hence we can say that minimization is not the same as optimization.

Optimization basically means to find the *most suitable* value of a function. This optimum value might be the minimum value or a maximum value of the function, however that is not necessary in every case. In the example above, we can say that the minimum cost of the company is zero, but it is not optimum because profits are zero when the cost is zero, thus zero is not an optimum solution. The optimal solution depends on the *constraints* which a function is under. The function may be a profit function (i.e. to maximize the profit) or a cost function (to minimize the cost) and the *optimum solution* for these functions will be governed by these constraints, like production cost, material cost, profit per pen, etc.

Now after getting a brief introduction of optimization we delve a little deeper by explaining the optimization method used in the paper.

Least Square (LS) Regression:

We assume that we have a set of measurements y_n that are gathered from some independent parameter values x_n. Also, that y_n is proportional to x_n by some random constant p but is corrupted by some random measurement errors ε_n. Then we can say that

$$y_n = px_n + \varepsilon_n$$

The method of Least Square Regression aims to determine the value of p that will minimize the value of the following equation.

$$\min_p \sum_{n=1}^{N} (y_n - px_n)^2$$

If we plot the measurements as a function of x_n, we obtain the slope of the line through the origin that best fits the measurements. Upon writing the error expression in vector form by collecting the $\acute{y}_n s$ and $\acute{x}_n s$ as column vectors

$$\min_p \|\vec{y} - p\vec{x}\|^2$$

The above equation can be also expressed as inner product of:

$$\min_p (\vec{y} - p\vec{x})^T (\vec{y} - p\vec{x})$$

Recursive Least Square (RLS):

Suppose we want to find an average of N numbers, then it is given by the following formula:

$$A(N) = \frac{x_1 + x_2 + \ldots\ldots x_N}{N}$$

If we add a new number to the current range of values, then the new mean becomes

$$A(N+1) = \frac{x_1 + x_2 + \ldots\ldots x_{N+} x_{N+1}}{N+1}$$

Thus, we can say that

$$(N+1)A(N+1) = x_1 + x_2 + \ldots\ldots x_{N+} x_{N+1}$$

$$(x_1 + x_2 + \ldots\ldots x_N) + x_{N+1} = NA(N) + x_{N+1}$$

Upon rearranging the above equation, we can say that $\mathrm{A(N+1)} = \mathrm{A(N)} + \frac{1}{N+1}(x_{N+1} - A(N))$

Hence, we can write the recursive definition as:

$$A_{new} = A_{old} + K(A_{old} - A_{data})$$

Here K is referred to as the gain and $(A_{old} - A_{data})$ is referred to as the innovation. And K depends upon the number of samples processed.

Writing the above equation in matric forms

$$\begin{pmatrix} a_{new} \\ b_{new} \end{pmatrix} = \begin{pmatrix} a_{old} \\ b_{old} \end{pmatrix} + \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} (y_{data} - (a_{old} x_{data} + b_{old}))$$

Algorithm of the above procedure:

- We write the formulas for N and N+1 data point
- In the formula for N+1 data point, replace all expressions involving first N data points by the formula for N data points.
- Next, we simplify the equation.
- We end up with an equation which is the form $H^{-1} - (H + vv^T)^{-1}$ with v as a vector
- Next, use matrix inversion to get $H^{-1} - (H + vv^T)^{-1} = \frac{H^{-1}vv^TH^{-1}}{1+ v^TH^{-1}v}$
- Next, we write matrix gain K in terms of H

Recursive least Squares Method is the special case of best linear unbiased estimate (BLUE) method which itself is a subset of Kalman Filter method.

What is the difference between LS and RLS?

The recursive least squares algorithm (RLS) is the recursive application of the well-known least squares (LS) regression algorithm, so that each new data point is taken in account to modify or correct a previous estimate of the parameters from some linear correlation of the model to the observed system. The method allows for the dynamical application of LS to time series acquired in real-time. As with LS, there may be several correlation equations and a set of dependent (observed) variables. The author tends to minimize the cost function which is made of the error between reference model output and the actual process output and thus obtain a suitable set of PID gain values. Hence the Recursive least Squares algorithm is introduced as an optimization problem.

EXTENDED OUTPUT DERIVATION AND TARGET TRACKING PROBLEM

To obtain the values of inputs and outputs for a data driven controller, the velocity PID algorithm is used. Hence, we first explain how a velocity PID algorithm is obtained.

Velocity PID algorithm

We obtain a discrete transfer function of a derivative as:

$$y(k) = \frac{f(k) - f(k-1)}{T_s}$$

$$Y[z] = \frac{F[z] - z^{-1}F[z]}{T_s} \text{ thus } \frac{Y[z]}{F[z]} = \frac{z-1}{zT_s}$$

Similarly, we obtain the discrete transfer function of an integral as follows:

$$y(k) = y(k-1) + \frac{f(k) - f(k-1)}{2} T_s$$

$$y[z] = z^{-1}[y(z)] + \frac{T_s}{2} F[z](1 + z^{-1})$$

$$\text{Thus } Y[z](1 - z^{-1}) = \frac{T_s}{2} F[z](1 + z^{-1})$$

$$\frac{Y[z]}{F[z]} = \frac{T_s}{2} \frac{z+1}{z-1}$$

Upon writing the PID in the discrete form we get:

$$\frac{U[z]}{E[z]} = K_p + K_i \int \tau \, d\tau + K_d \frac{d}{dt} \tau$$

$$= K_p + K_i \frac{T_s}{2} \left[\frac{z+1}{z-1}\right] + K_d \left[\frac{z-1}{z+1}\right]$$

$$= \frac{K_p(z^2 - 1) + K_i \frac{T_s}{2} z(z+1) + \frac{K_d}{T_s}[z-1]^2}{(z-1)z}$$

$$\frac{U[z]}{E[z]} = \frac{\left(K_p + K_i \frac{T_s}{2} + \frac{K_d}{T_s}\right) z^2 + \left(-K_p + K_i \frac{T_s}{2} - \frac{2K_d}{T_s}\right) z + \frac{K_d}{T_s}}{z^2 - z}$$

$$\frac{U[z]}{E[z]} = \frac{\left(K_p + K_i \frac{T_s}{2} + \frac{K_d}{T_s}\right) + \left(-K_p + K_i \frac{T_s}{2} - \frac{2K_d}{T_s}\right) z^{-1} + \frac{K_d}{T_s} z^{-2}}{1 - z^{-1}}$$

$$U[z](1 - z^{-1}) = \left[\left(K_p + K_i \frac{T_s}{2} + \frac{K_d}{T_s}\right) + \left(-K_p + K_i \frac{T_s}{2} - \frac{2K_d}{T_s}\right) z^{-1} + \frac{K_d}{T_s} z^{-2}\right] E[z]$$

$U[z] = z^{-1}U[z] + aE[z] + bz^{-1}E[z] + cz^{-2}E[z]$

Where z^{-1} is the backward operator and it has the following operation $z^{-1}y(k) = y(k-1)$

Thus we get,

$$U[k] = U[k-1] + ae(k) + be(k-1) + ce(k-2)$$

Where:

- a = $\left(K_p + K_i\frac{T_s}{2} + \frac{K_d}{T_s}\right)$
- b = $\left(-K_p + K_i\frac{T_s}{2} - \frac{2K_d}{T_s}\right)$
- c = $\frac{K_d}{T_s}$

We can write the output of a PID controller as:

$$m(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right]$$

The error is given by $e(t) = r(t) - c(t)$

$\left(\frac{df}{dt}\right)_k = \frac{f_k - f_{k-1}}{\triangle t}$ and $\int e(t)dt = \sum_{k=1}^{n} e_k \triangle t$

Writing the PID output in a discretized manner we get,

$$m(n) = K_p\left[T_d\left(\frac{e(n) - e(n-1)}{\triangle t}\right) + e(n) + \frac{1}{T_i}\sum_{k=1}^{n} e_k \triangle t\right]$$

Where $\triangle t = T_s$ (Sampling time)

The equation given below is called position algorithm. It shows the absolute value of actuator position

$m(n) = K_p e(n) + K_i e(n) + K_d[e(n) - e(n-1)]$

Here $K_i = \frac{K_p T_s}{T_i}$ and $K_p = \frac{K_p T_d}{T_s}$

Velocity algorithm is an alternative form of PID control algorithm and it is widely used to provide bump less transfer. The algorithm gives change in value of manipulated variable at each sample time.

$$\frac{dm(t)}{dt} = K_p\left(\frac{de(t)}{dt} + \frac{1}{T_i}e(t) + T_d\frac{d^2e(t)}{dt^2}\right)$$

$\triangle m = m(n) - m(n-1)$

$$= K_p[e(n) - e(n-1)] + \frac{\triangle t}{T_i}e(n) + \frac{T_d}{\triangle t}[e(n) - 2e(n-1) + e(n-2)]$$

Now upon grouping in terms of e(n), e(n-1) and, e(n-2)

$$\triangle m(n) = k_1 e(n) + k_2 e(n-1) + k_3 e(n-2)$$

$$\triangle m(n) = K_p\left[1 + \frac{T_s}{T_i} + \frac{T_d}{T_s}\right]e(n) - K_p\left[1 + \frac{2T_d}{T_s}\right]e(n-1) + \frac{K_p T_d}{T_s}e(n-2)$$

$$= \left[K_p + K_i + K_d\right]e(n) - \left[K_p + 2K_d\right]e(n-1) + K_d e(n-2)$$

$$= K_p[e(n) - e(n-1)] + K_i[e(n)] + K_d[e(n) - 2e(n-1) + e(n-2)]$$

$$\triangle U(k) = K_i e(k) - K_p \triangle y(k) - K_D \triangle^2 y(k)$$

Also, $\Delta = y_{r+1} - y_r$ and

$$\Delta(\Delta y_r) = \Delta^2(y_r) = y_{r+2} - y_{r+1} - y_{r+1} + y_r$$

$$= y_{r+2} - 2y_{r+1} + y_r$$

$$\Delta U(k) + \left(K_p + K_i + K_D\right)y(k) + \left(K_p + 2K_d\right)y(k-1) + K_d y(k-2) - K_i r(k) = 0$$

$$\Delta U(k) + K_p[y(k) - y(k-1)] - K_i[-y(k) + r(k)] + K_d[y(k) - 2y(k-1) + y(k-2)] = 0$$

$$\Delta U(k) = K_i[r(k) - y(k)] - \underbrace{K_p[y(k) - y(k-1)]}_{\Delta} - \underbrace{K_d[y(k) - 2y(k-1) + y(k-2)]}_{\Delta^2}$$

Hence the extended output is derived as

$$r(k) = \emptyset(k) = a_1(k)\Delta U(k) + a_2(k)\{y(k) - y(k-2)\} + a_3(k)\{y(k-1) - y(k-2)\} + y(k-2)$$

Here the only assumption is that in the extended output is $\emptyset(k)$, and $K_I(k) \neq 0$

In paper we are given that:

$$K_p = \frac{2a_2 + a_3 - 2}{a_1}$$

$$K_i = \frac{1}{a_1}$$

$$K_d = \frac{1 - a_2 - a_3}{a_1}$$

For solving the equations, we can see that:

$$a_1 - \frac{2}{K_p}a_2 - \frac{a_3}{K_p} = -\frac{2}{K_p}$$

$$a_1 + 0a_2 + 0a_3 = \frac{1}{K_i}$$

$$a_1 + \frac{a_2}{K_d} + \frac{a_3}{K_d} = \frac{1}{K_d}$$

Thus, we form the matrices as follows:

$$\begin{bmatrix} 1 & -\frac{2}{K_p} & -\frac{1}{K_p} \\ 1 & 0 & 0 \\ 1 & \frac{0}{K_d} & \frac{1}{K_d} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} -\frac{2}{K_p} \\ \frac{1}{K_i} \\ \frac{1}{K_d} \end{bmatrix}$$

Thus, we get

$$a_1 = \frac{1}{K_i}$$

$$a_2 = \frac{(K_p + K_d)}{K_i} + 1$$

$$a_3 = \frac{-(2K_d + K_p)}{K_i}$$

Since we are using a data driven controller, we don't have a system model thus no transfer function of the system. Since there is no transfer function, we can't directly get an error that has to be minimized because of which there is no direct reference. However, we need our output to follow a desired reference. Hence the reference is given by the reference model of our system $G_m(z^{-1})$. Hence the reference output that needs to be tracked is given by:

$$G_m(z^{-1})r(k)$$

Where r(k) is the input reference signal for our model. The values of $r(k)$ are given as follows:

$$\begin{aligned} r(k) &= 100\ (0\ s \le t \le 0.5\ s\ and\ 1 \le t < 1.5\ s) \\ &= 150\ (0.5\ s \le t \le 1\ s\ and\ 1.5 \le t < 2\ s) \end{aligned}$$

We compare the actual output of the controller with our reference model output. This way we obtain an error that needs to be minimized. Hence, we need to minimize the error given by

$$\begin{aligned} \varepsilon(k) &= y(k) - G_m(z^{-1})r(k) \\ &= y(k) - G_m(z^{-1})\emptyset(k) \\ &= y(k) - G_m(z^{-1})y(k-2) - G_m(z^{-1})\overline{\emptyset}(k) \end{aligned}$$

We know from earlier part of the derivation that

$$\overline{\emptyset}(k) = \emptyset(k) - y(k-2)$$

Thus, substituting the value into the error equation, we get:

$$\varepsilon(k) = y(k) - G_m(z^{-1})\emptyset(k)$$

$$= y(k) - G_m(z^{-1})\left[y(k-2) + \overline{\emptyset}(k)\right]$$

$$= y(k) - G_m(z^{-1})y(k-2) - G_m(z^{-1})\overline{\emptyset}(k)$$

$$= \underbrace{[y(k) - G_m(z^{-1})y(k-2)]}_{\tilde{y}}$$

$$- \underbrace{[G_m(z^{-1})[a_1\Delta U(k) + a_2(k)\{y(k) - y(k-2)\} + a_3(k)\{y(k-1) - y(k-2)\}]}_{\psi^T\theta^-}$$

As stated earlier, the objective of a controller is to make the output follow reference model output. Thus, the objective is minimization of:

$$\varepsilon(k) = y(k) - G_m(z^{-1})r(k)$$

where $G_m(z^{-1})$ is the discrete time transfer function of the reference model where $G_m(z^{-1})$ is given by

$$G_m(z^{-1}) = \frac{z^{-1}P(1)}{P(z^{-1})}$$

Where $P(z^{-1}) = 1 + p1z^{-1} + p2z^{-2}$

$$p1 = -2e^{\left(\frac{-\rho}{2\mu}\right)}\cos\left(\frac{\sqrt{4\mu-1}}{2\mu}\rho\right)$$

$$p2 = e^{\left(\frac{-\rho}{\mu}\right)}$$

$$\rho := \frac{T_s}{\rho}$$

$$\mu := 0.25(1-\delta) + 0.51\delta$$

In the above equation T_s is a sampling interval. σ and δ indicate rise time and damping property of a closed loop system. We assume that the value of as $(0 \leq \delta \leq 2.0)$.

The values of $p1, p2, \rho \; and \; \mu$ are given in the paper "A Practical reference model for Control System Design" written by Takashi Shigemasa et al. Unfortunately, the paper has been written in Japanese so currently it was not possible to further derive the origins of these parameters. Till now, we have covered how we can get the data for our data driven controller. Now in the subsequent section, we will show how the acquired data can be used for parameter estimation using Kalman Filter.

KALMAN FILTERING

Kalman filter is an iterative mathematical process that uses equations and consecutive data inputs to quickly estimate true value, position, velocity etc. of object being measures when measured values contain unpredicted or random errors.

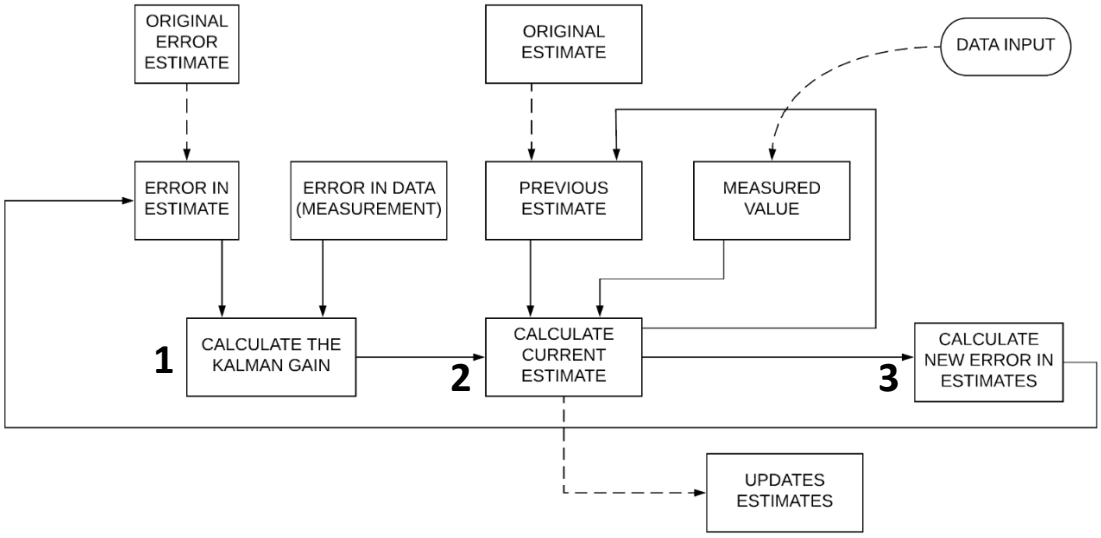


Fig: block diagram of a kalman filter

The following steps are the essence of a Kalman Filter.

- The dotted arrows from the ORIGINAL ERROR ESTIMATE block to ERROR ESTIMATE block and from ORIGINAL ESTIMATE to PREVIOUS ESTIMATE block shows that they are considered only once i.e. at the beginning of the Kalman filter when it initially needs some value to start.

- The ERROR IN DATA (MEASUREMENT) block and the ERROR IN ESTIMATE block are used together to obtain the KALMAN GAIN as depicted by the block 1.

- The PREVIOUS ESTIMATE block and the MEASURED VALUE block together are used to calculate the CURRENT ESTIMATE.

- The MEASURED value block continuously receieves updated values of data from the DATA INPUT block.

- The PREVIOUS ESTIMATE block gets updated every cycle from the value obtained from the CURRENT ESTIMATE BLOCK

- The data from the CURRENT EASTIMATE BLOCK is next sent to a block that calculates the NEW ERROR IN ESTIMATES.

Hence we can summerise the entire KALMAN FILTER in three essential steps:

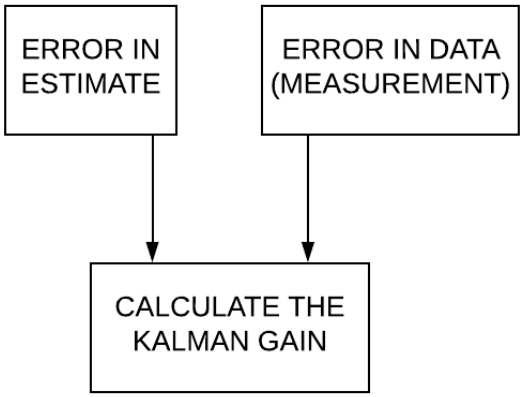
1) Calculating kalman gain
2) Calculating current estimate
3) Calculating new error in estimate

The process flow of the Kalman Filter is given below:

Step 1)

We assume the variables as follows:

KALMAN GAIN : KG
ERROR IN ESTIMATE: E_{EST}
ERROR IN MEASUREMENT ESTIMATE: E_{MEA}



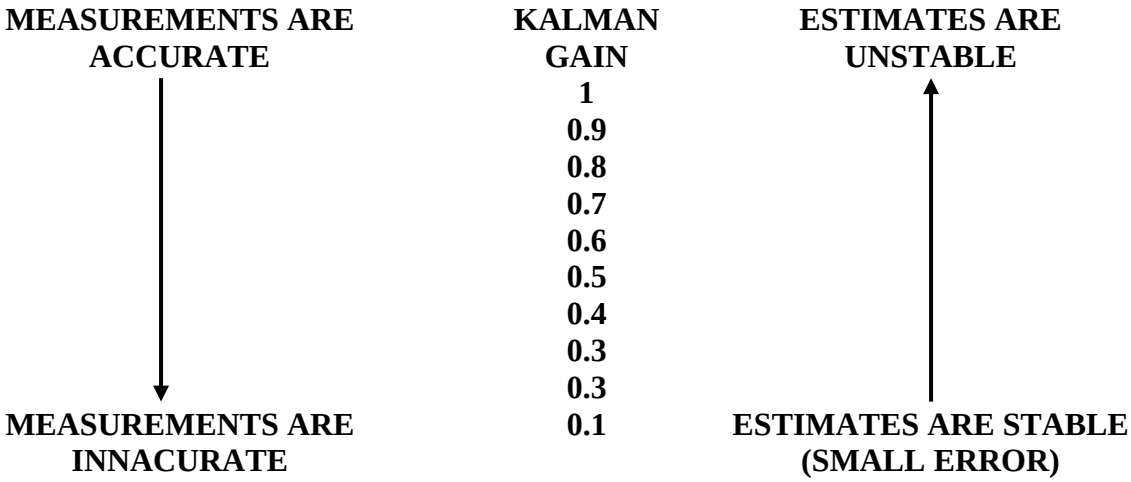
$$KG = \frac{E_{EST}}{E_{EST} + E_{MEA}}$$

Next we calculate the Current estimate at time T using the previous estimates, kalman gain and also the measured value.

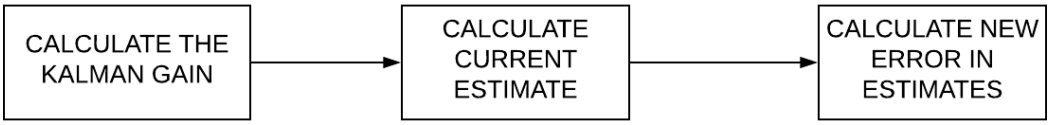
CURRENT ESTIMATE : EST_t
CURRENT ESTIMATE : EST_{t-1}

$EST_t = EST_{t-1} + KG[MEA - EST_{t-1}]$

$$EST_t = EST_{t-1} + KG[MEA - EST_{t-1}]$$



From the table above, we can see the value of Kalman gain varies from 0 to 1. Also, if the accuracy of measurements decreases then the Kalman gain also decreases. This is because we don't want to rely much on the measured observations and thus, we reduce the Kalman gain. So, this shows we don't trust the measured value.



$$KG = \frac{\mathrm{E_{EST}}}{\mathrm{E_{EST}} + \mathrm{E_{MEA}}}$$

$$EST_t = EST_{t-1} + KG[MEA - EST_{t-1}]$$

$$E_{EST} = \frac{(E_{MEA}) * (E_{EST_{t-1}})}{E_{MEA} + E_{EST_{t-1}}}$$

thus, we get

$$E_{EST_t} = [1 - KG]\, E_{EST_{t-1}}$$

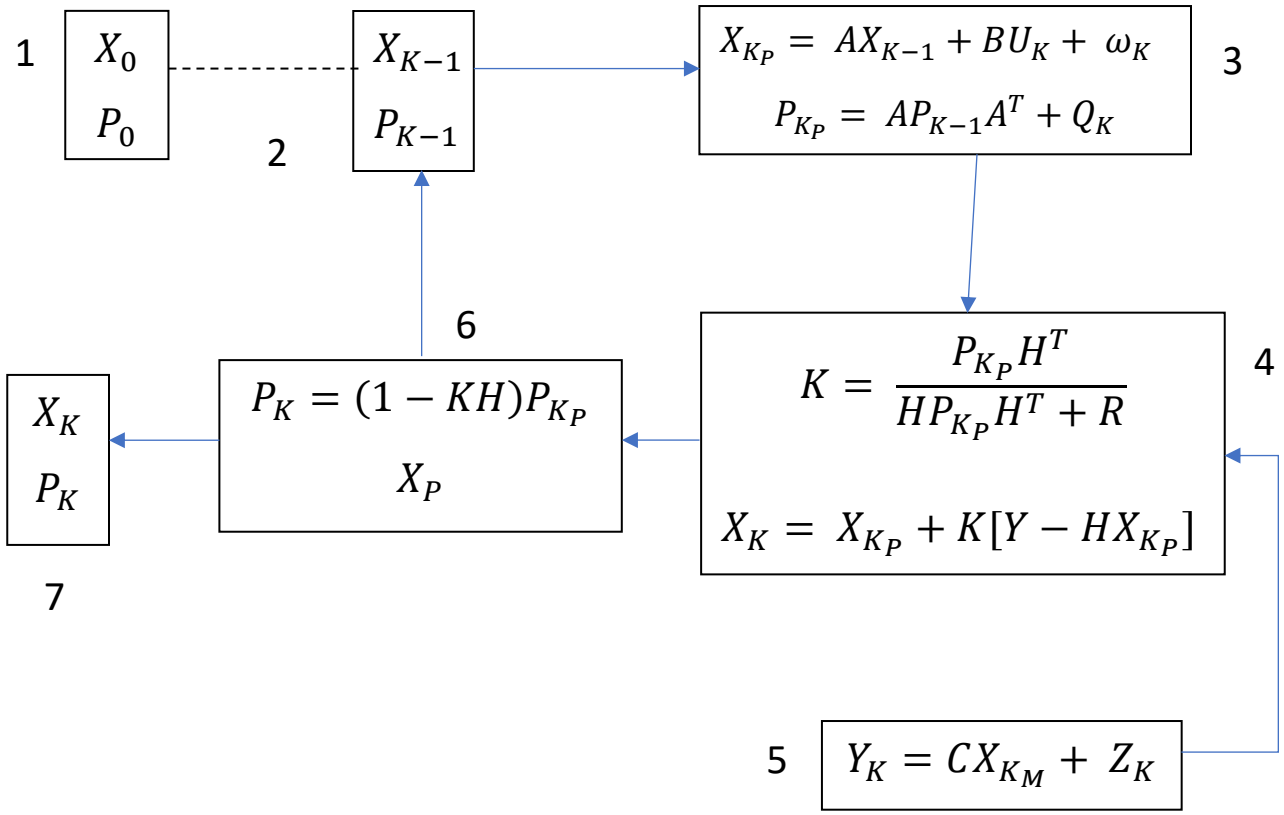


Fig: Kalman Filter step wise procedure

Here,

U = control variable matrix

ω = Predicted State noise matrix

Q = Process noise covariance

Y = measurement of the state

Z_K = Measurement Noise

K = Kalman gain

R = sensor noise covariance matrix (measurement error)

I = identity matrix

X = state matrix (X_i , $\dot{X}_i$)

P = process covariance matrix (represents error in the estimate/process)

Now we describe the Kalman filter process step by step:

$$P_{K_P} = AP_{K-1}A^T + Q_K$$

$$K = \frac{P_{K_P} + H}{HP_{K_P}H^T + R}$$

P = state covariance matrix (error in estimate)

Q = Process noise covariance matrix (Keeps state covariance matrix from becoming too small or going to 0)

R = measurement covariance matrix (error in measurement)

K = Kalman gain (Weight factor based on comparing error in estimate to the error in measurement)

Here A, A^T, H, H^T allow matrix dimension matching. They are identity matrices hence they are used. Also, they make sure dimensionality matching i.e. format of one matrix to fall into another.

Thus, we can say that if:

Observation	Conclusion
R → 0 then K → 1	Adjust primarily with measurement noise
R → "very large value" then K → 0	Adjust primarily with predicted state
P → 0	Measurements are mostly ignored

If there is more error in measured value, we put lesser weight to it and thus Kalman gain is small.

WHAT IS VARIANCE-COVARIANCE MATRIX?

X_i = individual measurements

$\bar{X}$ = average of measurements

$\bar{X} - X_i$ = deviation from average

$(\bar{X} - X_i)^2$ = square of standard deviation

$\sigma_x{}^2 = \frac{\sum_{i=1}^{N}(\bar{X}-X_i)^2}{N}$ = variance

$\sigma_x\sigma_y = \frac{\sum_{i=1}^{N}(\bar{X}-X_i)(\bar{Y}-Y_i)}{N}$ = covariance

$\sigma_x = \sqrt{\sigma_x{}^2} = \sqrt{\frac{\sum_{i=1}^{N}(\bar{X}-X_i)^2}{N}}$ = standard deviation

If you are ±1 S.D., then about 2/3 of all values fall within the range. If you take variance then almost 100% of values fall within range.

The matrices given below are called covariance matrix but technically it it's a variance-covariance matrix.

$$1D = \left[\frac{\sum_{i=1}^{N}(\bar{X}-X_i)^2}{N}\right]$$

$$2D = \begin{bmatrix} \frac{\sum_{i=1}^{N}(\bar{X}-X_i)^2}{N} & \frac{\sum_{i=1}^{N}(\bar{X}-X_i)(\bar{Y}-Y_i)}{N} \\ \frac{\sum_{i=1}^{N}(\bar{X}-X_i)(\bar{Y}-Y_i)}{N} & \frac{\sum_{i=1}^{N}(\bar{X}-X_i)^2}{N} \end{bmatrix}$$

$$3D = \begin{bmatrix} {\sigma_x}^2 & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_x\sigma_y & {\sigma_y}^2 & \sigma_y\sigma_z \\ \sigma_x\sigma_z & \sigma_y\sigma_z & {\sigma_z}^2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Here $a_{12}, a_{13}, a_{23}, a_{21}, a_{31}, a_{32}$are covariances and,

a_{11}, a_{22}, a_{33} = variances

We can get to the know the maximum variation by looking at a_{11}, a_{22}, a_{33}.

Also, if the off-diagonal elements or covariances are negative it means that means the two compared values are inversely related. Thus, if one value increases the other decreases. Similarly, if the off-diagonal elements are same then the values increase proportionally with respect to each other.

If estimate error for one variable is completely independent of other variable, then the covariance elements = 0.

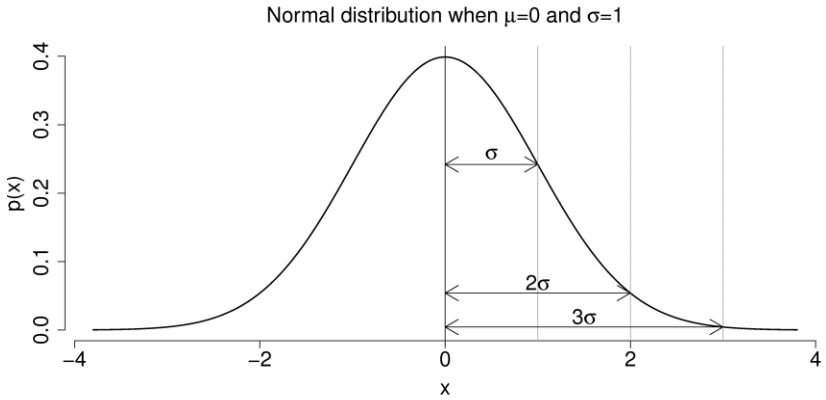


Fig: Gaussian distribution

In gaussian distribution:

$\pm 1\sigma$ = 68.3% of all values

$\pm 2\sigma$ = 95% of all values

$\pm 3\sigma$ = 99.99% of all values

By using covariance, we can get how far from the mean (or average values) we can expect the values to be.

Sequence of steps in Kalman filtering:

1.) First, we find the initial state of the system and then find the next state given by
$$X_{K_P} = AX_{K-1} + BU_K + \omega_K$$
2.) Next, we use the initial process covariance matrix and then calculate the next covariance matrix using the formula $P_{K_P} = AP_{K-1}A^T + Q_K$
3.) The Kalman gain K is calculated using the formula $K = \frac{P_{K_P}H^T}{HP_{K_P}H^T+R}$
4.) Next, we find the value of the measurement using the formula $Y_K = CX_{K_M} + Z_K$
5.) To find the next predicted state matrix we plug in the state value obtained earlier into $X_K = X_{K_P} + K\left[Y - HX_{K_P}\right]$
6.) And, the predicted process covariance is given by $P_K = (1 - KH)P_{K_P}$
7.) Finally, we update the values of state and process covariance matrices as follows: $X_K \rightarrow X_{K-1}$ and $Y_K \rightarrow Y_{K-1}$
8.) The cycle continues like this

We know that the state space equations are given using the formula:

$$\dot{X} = AX + BU$$

And
$$Y = CX + D$$

The above equations can be approximated as follows:

$$x(k+1) = A(k)x(k) + b(k)\{\tilde{u}(k) + \xi_v(t)\}$$

$$\tilde{y}(k) = c(k)x(k) + \xi_w(t)$$

Here $\tilde{u}(k)$ and $\tilde{y}(k)$ are inputs and outputs of state space model and the noises $\xi_v(t)$ and $\xi_w(t)$ are assumed to be white gaussian noise. Also, their mean and variances are $N(0, \sigma_v^2)$ and $N(0, \sigma_w^2)$.

Here, $\xi_v(t)$ is the system noise and $\xi_w(t)$ is the observation noise.

We have considered the noise to be additive independent white gaussian noise.

- It is additive because it is transmitted to the signal and not multiplied. So, the signal becomes $y(t) = x(t) + n(t)$, where $x(t)$ is the original clean estimated and $n(t)$ is the noise or the disturbance.
- It is white noise because it contains same amount of all the frequency or has the same power of all frequencies. Thus, the noise is equally present with the same power at all frequencies. Thus, we can say that, noise level is flat throughout every frequency.

Kalman filter gives us the algorithm to obtain that helps us minimize the minimum mean square error by using the output data.

$X^*(k)$ and $X^*(k) = arg_{\substack{min \\ \hat{x}(k)}} J\big(\hat{x}(k)\big)$ where $X(k)$ is a state variable which in our case are the control parameters $a_1, a_2, and\ a_3$. Also

$$J\big(\hat{x}(k)\big) = E\left[\big(x(k) - \hat{x}(k)\big)^T \big(x(k) - \hat{x}(k)\big)\right]$$

Thus, we can say that we construct a state estimate $\hat{x}(k)$ that minimizes the steady state covariance P. where P is given by:

$$P = \lim_{t \to \infty} E(\{x - \hat{x}\}\{x - \hat{x}\})$$

COMPARISION OF EQUATIONS

Now we correlate the equations given in the paper with actual Kalman filter equations

$$\hat{x}^-(k) = A(k-1)\hat{x}^-(k-1) + b(k)u(k-1) \qquad X_{K_P} = AX_{K-1} + BU_K + \omega_K$$

$$P^-(k) = A(k-1)P(k-1)A^T(k-1) + \sigma_v^2(k-1)b(k-1)b^T(k-1) \qquad P_{K_P} = AP_{K-1}A^T + Q_K$$

$$g(k) = \frac{P^-(k)c(k)}{C^T(k)P^-(k)c(k) + \sigma_\omega^2(k)} \qquad K = \frac{P_{K_P}H^T}{HP_{K_P}H^T + R}$$

$$\hat{x}(k) = \hat{x}^-(k) + g(k)\big(\tilde{y}(k) - c(k)^T\hat{x}^-(k)\big) \qquad X_K = X_{K_P} + K\big[Y - HX_{K_P}\big]$$

$$P(k) = \big(I - g(k)C^T(k)\big)P^-(k) \qquad P_K = (1 - KH)P_{K_P}$$

$$X(k+1) = A(k)X(k) + b(k)\big\{\tilde{U}(k) + \xi_v(t)\big\} \qquad \theta(k+1) = \theta(k) + b\xi_v(t)$$

$$\tilde{y}(k) = c(k)X(k) + \xi_\omega(t) \qquad \tilde{y}(k) = G_m(z^{-1})\psi^T(k)\theta(k) + \xi_\omega(t)$$

It is given that mean and variance of noise ξ_v and ξ_ω are 0 and σ_v^2, σ_w^2 . Thus, we can see that: the state variable $X = \theta$, and

$A(k)$ is given by the identity matrix. $A(k) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Also, $\tilde{U}(k) = 0$, $C(k) = G_m(z^{-1})\psi^T(k)$ and $b(k) = \psi(k)\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

For the reference model we have taken the sampling time $T_s = 1ms$, $\sigma(\text{rise time})$ = 30ms,

Initial values of the states, which serve as initial values for the Kalman Filter.

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 91.34 \\ 190.4 \\ -365.6 \end{pmatrix}$$

The consecutive PID gains $K_p = 0.144, K_I = 0.0109, K_D = 1.93$

The system and observation noises are $\sigma_v^2 = \sigma_w^2 = 0.00001$

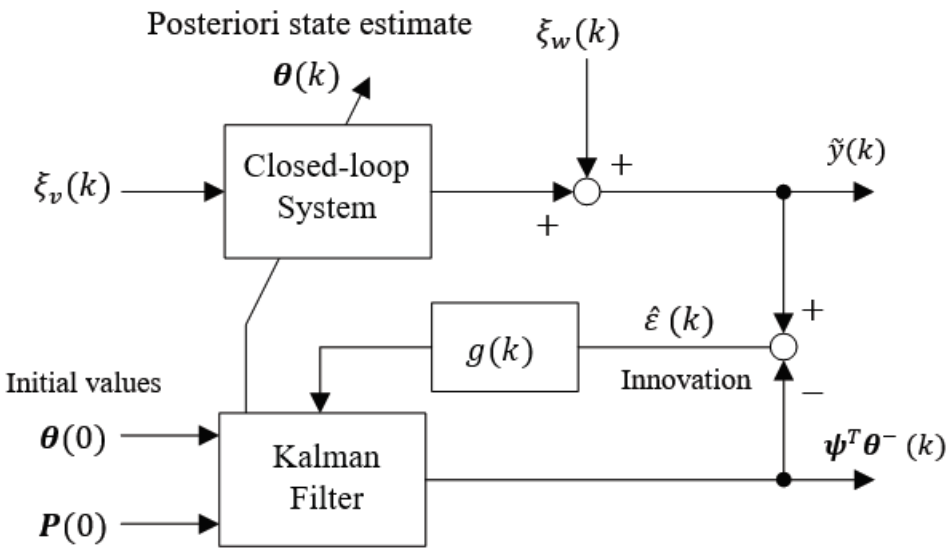


Fig: PID parameter estimation using Kaman filter

From the block diagram above we can conclude that, initially we need a set of initial state and covariance values for the Kalman filter to begin with. The states of the Kalman filter are the controller parameters given in the form a_1, a_2 *and* a_3 . These values are passed to the closed loop system whose output is the process value to which the system noise and the measurement noise get added. Thus the closed loop system output is taken and is compared with the predicted state output of the Kalman filter. The error function thus obtained is called the innovation, and this difference is again fed back to the Kalman Filter.

Since this is a data driven controller, we need to have the values of the input and output data over a period. However, the author fails to mention the exact given values of output and input at every time instant, thus I have taken the input and output values over certain discrete intervals of time. The graphs given below are of the input, output values and the reference signal that is fed to the reference model.

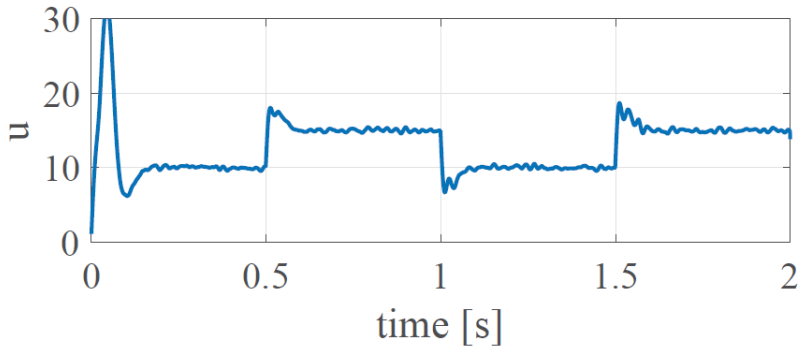


Fig: actual input signal to the data driven controller as given in the paper

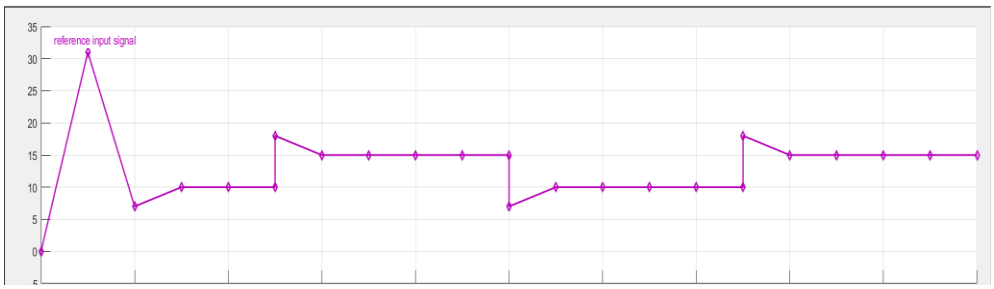


Fig: Input signal that I have generated using distinct points from the graph

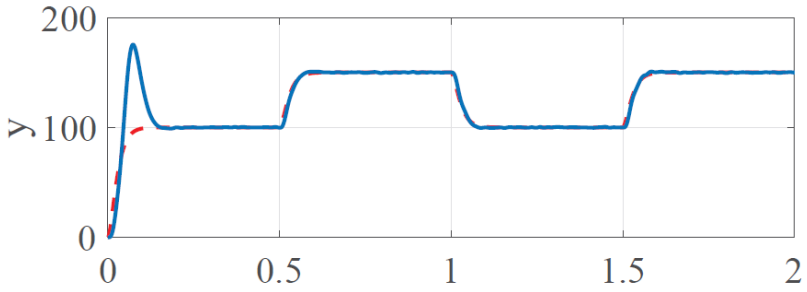


Fig: The output signal for the data driven controller as given in the paper

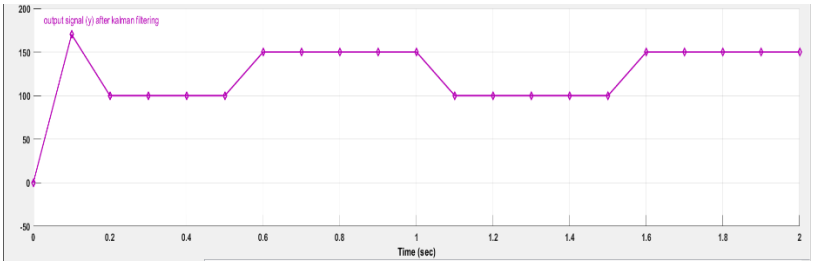


Fig: Output signal that I have generated using distinct points from the graph

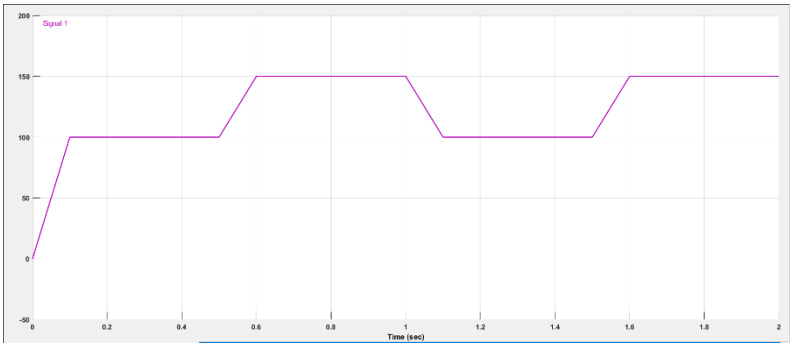


Fig: reference input signal r(k)

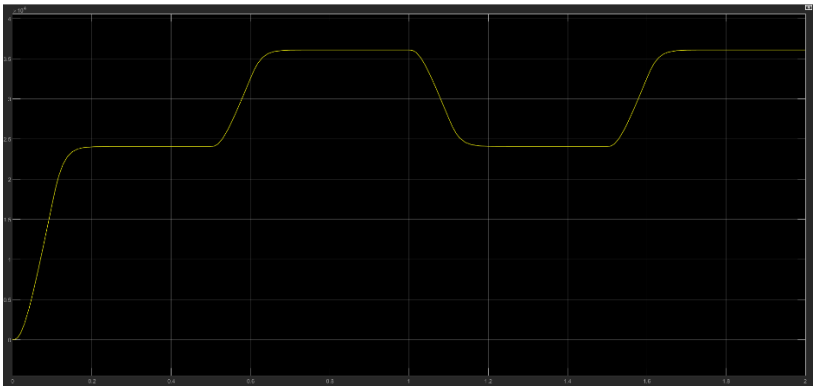


Fig: Output signal obtained after multiplying the reference model G_m with the reference signal

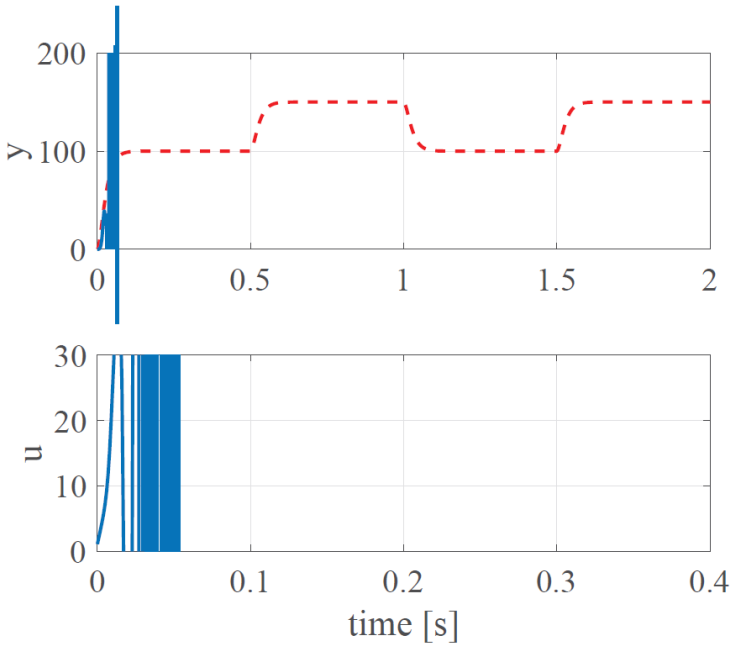


Fig: Control result obtain by the Recursive Least Square method

Here I can't simulate the result obtained by Recursive least Square algorithm because one can't comprehend the data from the graphs as seen above. However, the author does provide the initial values of initial control parameters obtained from Fig:3 in the paper

With the above discussion we get a brief idea of the process of implementing the Kalman Filter for PID parameters estimation. But as discussed earlier, there are a few more methods that can be used for system parameter estimation. I have given a brief information about these methods below. The methods that have been considered are VRFT (Virtual reference feedback tuning), FRIT (Fictitious reference iterative tuning).

Fictious Reference Iterative Tuning (FRIT)

In the FRIT method we try to minimize the cost function by iteration of experiments along the update rule of nonlinear optimization because the required quantitates such as gradients are described by data.

We perform some experimentation on a closed loop system using some initial parameters ρ_{ini}

And thus, obtain the initial data $u_{ini} := u(\rho_{ini})$ and $y_{ini} := y(\rho_{ini})$. Here the assumption is that $C(\rho_{ini})$ has temporarily stabilized the system such that y_{ini} and u_{ini} are bounded.

Thus, we compute the fictious reference signal as:

$$\tilde{r}(\rho) = C(\rho)^{-1}u_{ini} + y_{ini}$$

Also, the cost function to be minimized is given by:

$$J_F(\rho) = \|y_{ini} - T_d\tilde{r}(\rho)\|^2$$

where $J_F(\rho)$ is the cost function that has to be minimized.

Next, we assume that $y_{ini} = Gu_{ini}$

Thus, we get $T(\rho)\tilde{r}(\rho) = \frac{GC(\rho)}{1+GC(\rho)}\tilde{r}(\rho)$

$$= \frac{GC(\rho)}{1 + GC(\rho)} * \frac{1}{C(\rho)}u_{ini} + y_{ini}$$

Thus $T(\rho)\tilde{r}(\rho) = y_{ini}$

The output of the close loop $T(\rho)$ with reference to the fictious reference output is completely equal to the actual output. Also, the minimization of $J_F(\rho)$ is equivalent to minimization of

$$\left\| 1 - \frac{T_d}{T(\rho)} y_{ini} \right\|^2$$

The algorithm for FRIT is shown as follows:

- set the initial parameters ρ_{ini}
- obtain u_{ini} and y_{ini} using ρ_{ini}
- minimize the cost function $J_F(\rho)$ with fictitious reference $\tilde{r}(\rho)$
- implement ρ^* and repeat the entire loop again where $\rho^* = \min J_F(\rho)$

Virtual Reference Feedback tuning (VRFT)

Suppose that a controller results in a closed-loop system whose transfer function is M(z). Then, if the closed-loop system is fed by any reference signal r(t), its output equals M(z)r(t). Hence, a necessary condition for the closed-loop system to have the same transfer function as the reference model is that the output of the two systems is the same for a given r(t).

Algorithm of VRFT:

- A virtual reference $\tilde{r}$ is chosen such that $y(t) = M(z)\bar{r}(t)$
- The corresponding tracking error is given by $e(t) = \bar{r}(t) - y(t)$ with an assumption that $M(z) \neq 1$ or else $e(t) = 0$
- Next, the signals $e(t)$ and $u(t)$ are filtered with $L(z)$ such that $e_L(t) = L(z)e(t)$ and $u_L(t) = L(z)u(t)$
- The control parameter vector $\hat{\theta}_N$ is selected such that it minimizes the following criterion:

$$J_{VR}^{N}(\theta) = \frac{1}{N}\sum_{t=1}^{N}(u_L(t) - C(z,\theta)e_L(t))^2$$

When C(z, θ) = $\beta^T(z)\theta$ we can write the above equation as:

$$J_{VR}^{N}(\theta) = \frac{1}{N}\sum_{t=1}^{N}(u_L(t) - \varphi_L^T(t)\theta(t))^2$$

Where $\varphi_L(t) = \beta(z)\, e_L(t)$

And the parameter vector $\hat{\theta}_N$ is given by

$$\hat{\theta}_N = \left[\sum_{t=1}^{N}\varphi_L(t)\,\varphi_L^T(t)\right]^{-1}\sum_{t=1}^{N}\varphi_L(t)u_L(t)$$

The main features of VRFT can be summarized as follows:

- VRFT requires just a single set of I/O data;
- It does not require the identification of a mathematical model of the plant (i.e. it is a direct method);
- The control parameters are determined in one-shot, with no need for iterations;
- Controller complexity can be fixed in the beginning when the controller class is selected. So, no controller complexity reduction is required;
- the controller provided by VRFT is only approximately optimal.

Hence, we can conclude that FRIT and VRFT don't require a lot of data i.e. they work on a single shot basis. Also, the FRIT method works on the output of the system and VRFT method works on the input. However, both work on one shot experimental data.

CONCLUSION

In this paper a data oriented self-tuning controller scheme has been proposed that uses Kalman Filter for estimating the gains of a PID controller. The proposed scheme can give good control performance for time-variant systems and, there is no system identification required thus it saves time and considerable monetary costs. Also because of the use of a Kalman Filter, the robustness of the system gets ensured as the Kalman filter takes into consideration possible system and measurement noises. The paper uses a user defined reference model to track the output of the system and the error function thus obtained is fed back to the Kalman Filter. Thus by the use of data driven techniques, the tuning of controllers can be done optimally and hence, we can say that the method has expanded the application area of the PID controller.

REFERENCES:

1. Ashida, Y., Hayashi, K., Wakitani, S., and Yamamoto, T. (2016). A novel approach in designing pid controllers using closed-loop data. Proc. of the 2016 American Control Conference, 5308 – 5313.
2. Ashida, Y., Wakitani, S., and Yamamoto, T. (2017). Preprints of the 20th world congress the international federation of automatic control (ifac 2017). Design of an Implicit Self-Tuning PID Controller Based on the Generalized Output, 14511 – 14516.
3. Kaneko, O. (2013). Data-driven controller tuning: Frit approach. Proc. of 11th IFAC International Workshop on Adaptation and Learning in Control and Signal Processing, 326–336.
4. M.C.Campi (2002). Virtual reference feedback tuning (vrft) : A direct method for the design of feedback controllers. Automatica, 38(8), 1337–1346.
5. Vilanova, R. and Visioli, A. (2012). PID control in the Third Millennium: lessons learned and new approaches. Springer. Visioli, A. (2006). Practical PID control. Springer Science & Business Media.
6. Wakitani, S., Yamamoto, T., Sato, T., and Araki, N. (2011). Design and experimental evaluation of a performance-driven adaptive controller. Proc. of IFAC World Congress, 7322–7327.
7. https://math.stackexchange.com/questions/631498/simple-example-of-recursive-least-squares-rls
8. https://www.mathworks.com/help/control/ref/pidtune.html
9. https://www.mathworks.com/help/control/examples/creating-discrete-time-models.html
10. https://www.mathworks.com/help/ident/ug/algorithms-for-online-estimation.html

SIMULATION RESULTS

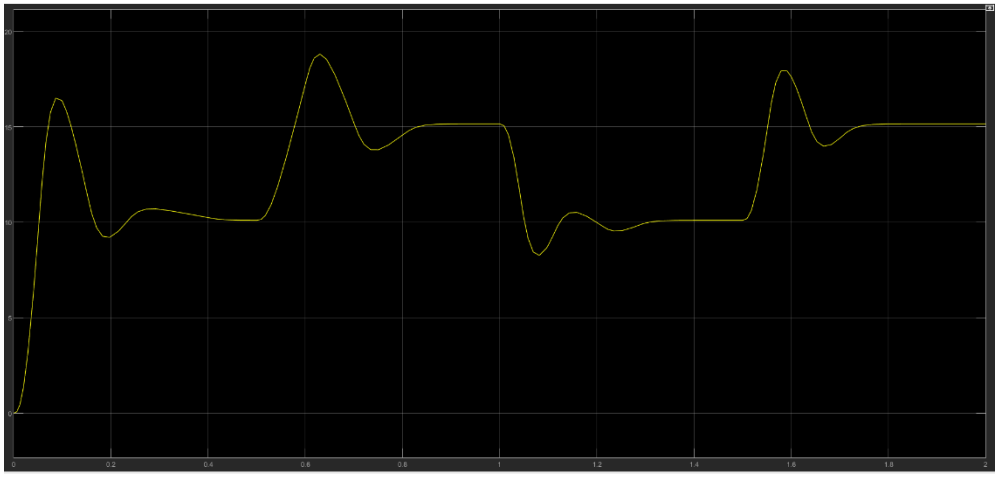


Fig: input signal as shown in figure 2 in the paper

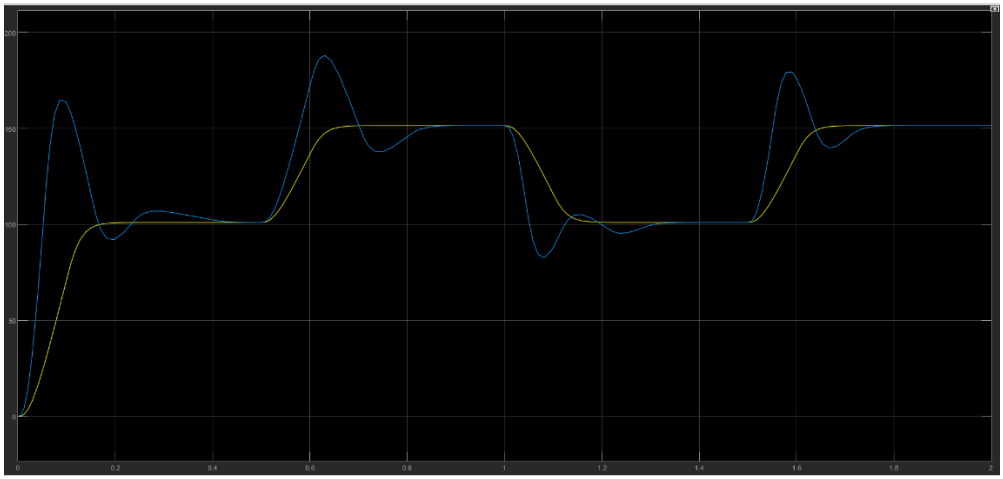


Fig: output as shown in figure 2 compared with the reference signal

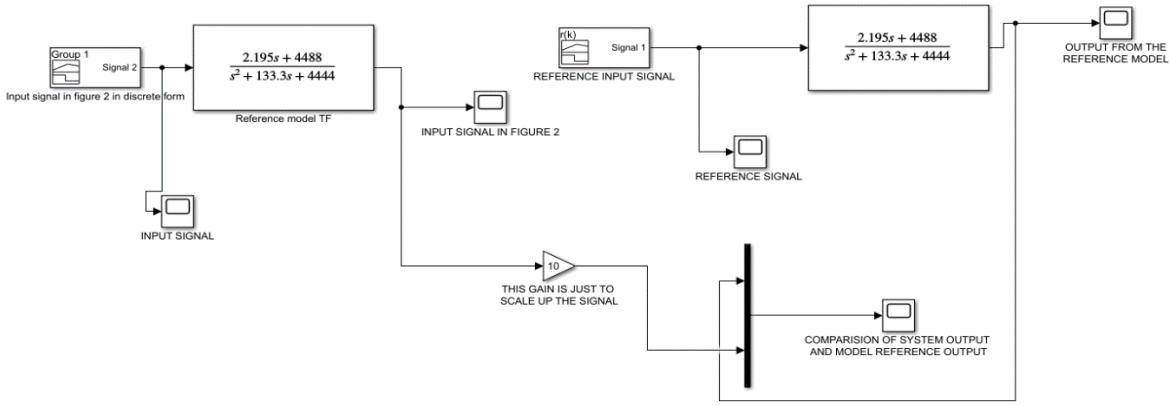


Fig: SIMULINK block diagram

PROJECT #1

PID GAIN ESTIMATION USING KALMAN FILTERING

ENPM : 667 CONTROL OF ROBOTIC SYSTEMS

NAME: ADITYA VAISHAMPAYAN

UID: 116077354

EMAIL: adityav@terpmail.umd.edu

CONTACT NO: +1-(240)-743-0530

Table of Contents


```
clc;
close all;
```

INPUT OUTPUT & REFERENCE SIGNALS

```
% values of time for input
t_input = [0 0.1 0.2 0.3 0.4 0.5 0.51 0.6 0.7 0.8 0.9 1.0 1.01 1.1 1.2 1.3 1.4 1.5 1.51 1.6 1.7 1.8 1.9
% values of input u
u_input = [0 31 7 10 10 10 18 15 15 15 15 15 7 10 10 10 10 10 18 15 15 15 15 15];


% values of time for output
t_output = [0 0.1 0.2 0.3 0.4 0.5 0.51 0.6 0.7 0.8 0.9 1.0 1.01 1.1 1.2 1.3 1.4 1.5 1.51 1.6 1.7 1.8 1.9
% values of output y;
y_output = [0 180 100 100 100 100 100 100 150 150 150 150 150 100 100 100 100 100 100 150 150 150 150 15

%in the above vectors I have take time instants such as 0.51 1.01 1.51
%becuase of the step nature of the input signal as seen in figure 4 of the
%paper. Since we know that a real time input signal can't have two values
%for the same instant of time, i have taken a very small interval of 0.01
%second after the time instant so as to approximate a step change in the value of the signal.
```

```
%In order to generate the output from the reference model we have to give inputs values to it. Our refer
% in transfer function form. There is a command in matlab LSIM that allows us to get the set of output v
% given a set of input values however, the only condition for the LSIM coomand to work is that the time
% LSIM function need to be evenly spaces.

%value of time for reference output
t_reference2 = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0];
%values of reference output
r_output2 = [0 100 100 100 100 100 150 150 150 150 150 100 100 100 100 100 150 150 150 150 150];

%reason for taking reference output 2 is becuause lsim takes in only evenly
%spaced values. Since r_ref wasn't evenly spaced we couldn't use it to
%obtain the value which we can repeat for a particular time instant, we
%used refoututput 2

% value of time for reference output
t_reference = [0 0.1 0.2 0.3 0.4 0.5 0.51 0.6 0.7 0.8 0.9 1.0 1.01 1.1 1.2 1.3 1.4 1.5 1.51 1.6 1.7 1.8
% reference signal r(k) with time approximation
r_output = [0 100 100 100 100 100 100 150 150 150 150 150 150 100 100 100 100 100 100 150 150 150 150 15

%after feeding the reference input values to the transfer function, a reference output signal is generat
%approximated to incorporate time itervals 0.51 1.01 and 1.51

% this is Y_multiplication_output i.e GMzinverse_rt = Y_multiplication_output
% values of time of the reference output siganl
t_GMzinverse_rt = [0 0.1 0.2 0.3 0.4 0.5 0.51 0.6 0.7 0.8 0.9 1.0 1.01 1.1 1.2 1.3 1.4 1.5 1.51 1.6 1.7
% values of the output reference signal
GMzinverse_rt = [0.00 16985.77 24003.68 24042.14 24042.22 24042.22 24042.22 32535.11 36044.07 36063.29 3
```

PLOTTING INPUT OUTPUT AND REFERENCE SIGNALS

```
%-----------------------PLOTTING INPUT SIGNAL-------------------
plot(t_input, u_input, 'Color','r');
title('Input to the system')
xlabel('time [s]');
ylabel('input u')
```

```
%----------PLOTTING OUTPUT SIGNAL----------------
plot(t_output, y_output,'Color','g');
title('Output of the system');
xlabel('time [s]');
ylabel('output y');
```

```
%----------------PLOTTING REFERENCE SIGNAL r(k)---------------
% r(k) varies as follows:
%---------------r(k) = 100 (0<= t <= 0.5) & (1.0<= t<= 1.5)----------
%---------------r(k) = 150 (0.5<= t <= 1) & (1.5<= t<= 2)------------
plot(t_reference, r_output,'Color','b');
title('reference signal for the system output to follow')
xlabel('time [s]');
ylabel('reference signal r(k)')
```

INITIALIZING VARIABLES FOR KALMAN FILTER

```
% we initialise the component of state vectors for kalman filter
syms a1 a2 a3;
% initialising PID gains
syms Kp Ki Kd;
% initialising terms for the kalman filter
syms theta_minus theta_initial P_minus K E ksi;

%state vector for kalman filter
theta = [a1; a2; a3];

%initital values of theta
theta_initial = [91.34;
                 190.4;
                -365.6];

%initial covariance matrix
P_initial = [100 0 0;
             0 100 0;
             0 0 100];

theta_predicted = [0; 0; 0];

b = [1;
     1;
     1];

BB_transpose = [1 1 1; 1 1 1; 1 1 1];

V = 0.00001;%variance of system nosie
W = 0.00001;%variance of observation noise


ksi = [1;
       1;
       1];

syms theta_predicted Y_tilda
syms Error Measured_Value P_predicted
syms output_index
```

GENERATING MODEL FOR THE REFERENCE OUTPUT

```
%reference output
syms p1 p2 mu rho sigma delta Ts sigma;

sigma = 0.030;% rise time in seconds [30 ms]
delta = 0; % damping coefficient [0]
Ts = 0.001; %Sampling interval in seconds [1 ms]
```

```
rho = Ts/sigma ;
mu = 0.25*(1 - delta) + 0.51*delta;
p1 = -2*exp((-1*rho)/(2*mu)) * cos((sqrt(4*mu - 1)/(2*mu))*rho);
p2 = exp(-rho/mu);
num = [0 0.0042];
denum = [1 p1 p2];

%transfer function of the reference model in discrete form
G = filt(num,denum,Ts)
```

```
G =

          0.0042 z^-1
  ----------------------------
  1 - 1.871 z^-1 + 0.8752 z^-2

Sample time: 0.001 seconds
Discrete-time transfer function.
```

```
%conversion of discrete form to continuous form
G1 = d2c(G)
```

```
G1 =

     2.195 s + 4488
  --------------------
  s^2 + 133.3 s + 4444

Continuous-time transfer function.
```

```
%reference output obtained from the reference model. We will compare our
%process output against the reference output
Y_multiplication_output = lsim(G1,r_output2,t_reference2);

%plotting output signal of the reference tf
plot(t_reference2,Y_multiplication_output)
title('output from reference model with r(k) as its input');
xlabel('time[s]');
ylabel('reference model output');
```

INITIALIZING ARRAYS FOR Kp Ki and Kd to store their values

```
%storing Kp Ki Kd in an array;
ValueofKp = zeros(1,24);
ValueofKi = zeros(1,24);
ValueofKd = zeros(1,24);
```

KALMAN FILTERING PROCESS

```
%initialising counter
```

```
output_index = 1;

%here for loop iterates 24 times. length(t_input) is 24 and t_input signify
%the time intervals
for i = 1:1:length(t_input)

    %----------------------INITIAL STATE AND COVARIANCE MATRICES----------------------

    %taking initial values of state vector where theta_minus is that state
    %vector made of [a1;a2;a3]
    theta_minus = theta_initial;

    %P_minus is the covariance matrix
    %formula for calculating the covariance matrix
    P_minus = P_initial + V*diag(BB_transpose);

    %--------------------KALMAN FILTERING PROCEDURE--------------------

    %calculating kalman Gain
    K = (P_minus * ksi)/((ksi' * P_minus*ksi) + W);

    %in the paper the author is taking the difference of the actual output
    %signal with the output of the reference model thus generating an error signal. This error function
    %called as the innovation which gets later fed back to the kalman filter.

    %calculaing the error "innovation"
    %y_output(i) is the output of the system at ith instant of time
    %GMzinverse(i) is the output of the reference model at ith instant of time
    Error = y_output(i) - GMzinverse_rt(i);

    %incrementing the output index for the next iteration
    output_index = output_index + 1;

    %---------------------------PREDICTING THE NEW STATE VECTOR---------------------------
    %predicted the new state vector
    theta_predicted = theta_minus + (K*Error);

    %-------------------PREDICTING THE NEW COVARIANCE MATRIX-------------

    %updating the new covariance matrix
    P_predicted = (eye(3) - K*ksi')*P_minus;

    %-----------OBTAINING THE VALUES OF Kp Ki Kd FROM CONTROL----------------
    %--------------PARAMETERS a1 a2 and a3----------------------

    %obtaining PID gains from state vectors
    Kp = (2*theta_minus(2) + theta_minus(3)-2)/theta_minus(1);
    Ki = 1/theta_minus(1);
    Kd = (1 - theta_minus(2) - theta_minus(3))/theta_minus(1);

    %storing value of Kp Kd Ki in an array
    ValueofKp(i) = Kp;
    ValueofKd(i) = Kd;
    ValueofKi(i) = Ki;

    %--------UPDATING STATE AND COVARIANCE MATRICES FOR NEXT CYCLE-----

    %updating the values for the next cycle
```

```
    theta_initial = theta_predicted;
    P_initial = P_predicted;

end
```

PLOTTING Kp Ki AND Kd GAINS

```
%because of lack of data, i had to make up approximated data and hence the
%pid gains arent the same as outhors. But we can see that the theoritical PID gains
%are getting stabilised over time and hence i was able to stabilise the
%system using just the input output data without obtaining the transfer
%function or model of the system

%----------plotting Kp----------
plot(t_output,ValueofKp)
title('Proportional gain');
xlabel('time[s]');
ylabel('Kp');
```

```
%---------plotting Ki------------
plot(t_output,ValueofKd);
title('Integral gain');
xlabel('time[s]');
ylabel('Ki');
```

```
%------plotting kd---------
plot(t_output,ValueofKi)
title('Derivative gain');
xlabel('time[s]');
ylabel('Kd');
```