Tutorial on a very simple yet useful filter : the first order IIR filter[*]

J. Arzi
E-mail : `contact AT tsdconseil.fr`

April 1, 2016

Foreword

This very simple numerical filter (only one tap!), also known as **exponential filter** because of its impulse response, is, as its equivalent in the analogic domain (the RC electrical filter), very easy to tune and implement.

Because of its extrem simplicity, it is often disregarded in favor of more sophisticated filters, and yet it finds a lot of applications, and even, it is the optimal filter for some problems. We will present in this small tutorial:

1. How to interpret intuitively this filter (from the points of view of smoothing and control),
2. How to design easily this filter (computing the coefficient) according to physical and intuitive parameters (cut-off frequency, time constant),
3. Some constraints and traps to be avoided during implementation,
4. How this filter can be understood as a Kalman filter (hence, in specific conditions, it is an optimal estimator).

[*]The most recent version of this document is available on the following web page: `http://www.tsdconseil.fr/tutos/index-en.html`

Contents

1 Definition et interpretations

1.1 First formulation : as a smoother

A first way to express this filter with an equation is the following:

$$\boxed{y_n = (1-\gamma)y_{n-1} + \gamma x_n} \tag{1}$$

where x_n is the input signal, y_n the output signal, and γ the (fixed) coefficient of the filter. This coefficient γ is also sometimes known as the **"forget factor"**: the closer γ is to 1, the faster the filter "forgets" the old inputs.

In this interpretation, we consider the filter as a smoother: each output sample is a **weighted mean** betwen the last input sample and the previously computed value.

The relation with an electrical RC filter appears clearly if we apply this filter to an inversed step[1] ($x_{n\leq 0} = 1, x_{n>0} = 0$) :

$$y_n = (1-\gamma)y_{n-1} = (1-\gamma)^n y_0 = (1-\gamma)^n \text{ supposing that } y_0 = 1$$

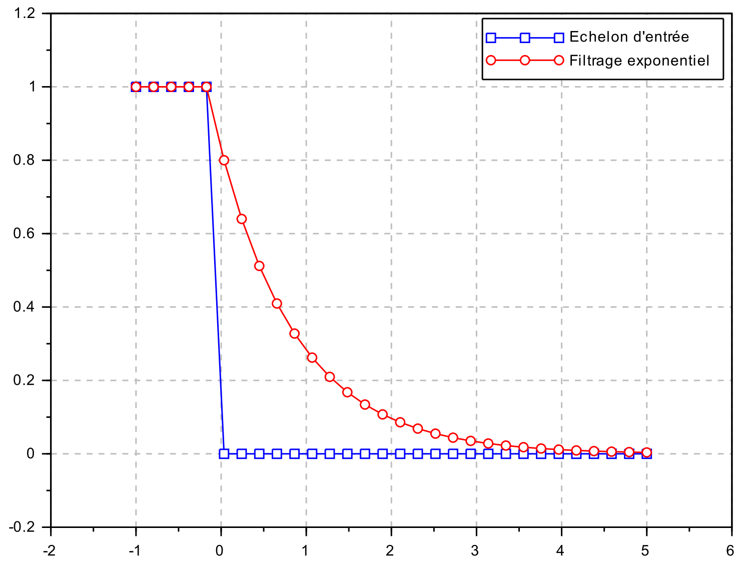

Figure 1: *Step response*

In other words the response decrease according to an exponential law. If we want to shape i as an analog RC filter, we just have to write:

[1]We have course the same behavior with a positive step $x_{n\leq 0} = 0, x_{n>0} = 1$, only the computings are just simpler with an inversed step.

$$(1-\gamma)^n = e^{n\log(1-\gamma)} = e^{-t\cdot\frac{-\log(1-\gamma)}{T_s}}$$

where T_s is the sampling period. Hence, the factor $\frac{-T_s}{\log 1-\gamma}$ is the time constant τ of the corresponding RC filter, and more generally (see appendix A page 7), we can compute the forget factor γ as a function of the desired time constant:

$$\boxed{\gamma = 1 - e^{-T_e/\tau}} \tag{2}$$

Alternatively, we can also express γ as a function of the desired cut-off frequency f_c:

$$\boxed{\gamma = 1 - e^{-2\pi f_c/f_s}} \tag{3}$$

où f_s is the sampling frequency.

So, this filter is very easy to tune from intuitive and with physical meaning parameters, and can also be retuned dynamically.

1.2 Second formulation: tracking

A second way (completely equivalent) to write the equation (1) is the following:

$$\boxed{y_n = y_{n-1} + \gamma(x_n - y_{n-1})} \tag{4}$$

This interpretation is closer to the **system control** view: x_n is a (noisy) signal to be tracked, y_n is the current estimation, and $x_n - y_{n-1}$ is the tracking error. In other words, γ can be seen as the correction gain (or feedback) to be applied on the error: the higher is γ, the faster the system react to changes in the input.

We will see at the end of this tutorial, that in this form, the first order IIR filter is actually a Kalman filter (for a very simple model and in steady state)!

2 Implementation (in C)

If one uses a processor with floating point computing available, then the implementation is trivial and correspond directly to the equation (1) :

```
y = (1 - gamma) * y + gamma * x;
```

where `y` is used both as an accumulator and as an output variable.

In contrary, for a **fixed point** implementation (that is, when one can use only integer computing, such as is usually the case on small micro-controlers), the forget factor γ will be represented by an integer between 0 and $N = 2^k$, and we will consider conventionnaly that `gamma=`2^k correspond to $\gamma = 1$.

The equation (1) becomes then (first naive implementation):

```
y = ((N - gamma) * y + gamma * x) >> k;
```

The shift of k bits to the right (division by 2^k) enables to simulate a fractional multiplication by γ using its integer representation `gamma` (in fact, $\gamma = \frac{\texttt{gamma}}{2^k}$).

The problem with this implementation is that the accumulator (and the output variable) y is represented with the same precision as the input x, so that the small variations of y during tracking of the input x are truncated at each iteration, and since, according to the forget factor, the output y can evolve slower than the input x, the filter can never converge to x!

To solve this problem, one has to represent the accumulator in fixed point, just as the `gamma` coefficient. For instance, one can add k bits to the representation of y, and the update equation becomes:

```
y = (((N - gamma) * y) >> k) + gamma * x;
```

3 Interpretation as a Kalman filter

The Kalman filter is a very generic (and extensible) algorithm to compute optimally the (hidden) state of a system with linear evolution and for which we have linear observations (with gaussian and memory-free noise). The general model of this problem is what is called the *state / space representation:*

$$\begin{aligned} \mathbf{x}_{n+1} &= A\mathbf{x}_n + B\mathbf{u}_n + \mathbf{v}_n \\ \mathbf{y}_n &= C\mathbf{x}_n + D\mathbf{u}_n + \mathbf{w}_n \end{aligned}$$

where :

$\mathbf{x}_n$ is the state vector (hidden variables of the system, which we are seeking to compute, for instance the current position of a robot),

$\mathbf{u}_n$ is the input / command vector (input that we control / known, for instance the current that we inject in the motors),

$\mathbf{y}_n$ is the output / observation vector (for instance, the values measured by available sensors: accelerometers, gyroscopes, ...),

$\mathbf{v}_n, \mathbf{w}_n$ are (respectively) the process and observation noises (hypotheses: **gaussian white noise**, of known covariance matrix Q and R)

A, B, C, D are the matrix defining the system evolution and observation (e.g. **linear** system)

Without going into the details of the Kalman theory, in steady-state regime (after enough iterations), for a stable and observable system, the equation to solve this system becomes:

$$\hat{x}_{n+1} = A\hat{x}_n + K \cdot (y_n - C\hat{x}_n) \tag{5}$$

where K is a fixed gain (called asymptotic gain), which can be computed from the system matrix and noise covariance matrix (using the Riccati equation). Under this shape, we can recognize almost our first order IIR filter, as expressed in equation (4) :

- $y_n - C\hat{x}_n$ is the tracking error (error between expected and real observations)

- K is our forget factor γ

Of course, the Kalman formulation is more general since it can handle multi-dimensionnal systems (A, K et C can be matrix). That said, in the special case of a scalar system:

$$x_n = x_{n-1} + v_n \tag{6}$$

$$y_n = y_{n-1} + w_n \tag{7}$$

That is, for a system representing a **random walk** (random movement of v_n at each iteration), observed in an imperfect manner (observation noise w_n), then the Kalman asymptotic solution coincides exactly with the first order IIR filter:

$$\hat{x}_{n+1} = \hat{x}_n + K \cdot (y_n - \hat{x}_n) \tag{8}$$

In other words, *the simple first order IIR filter is the optimal filter to track a scalar value subject to random variations*, and moreover the Kalman theory (Riccati equation) gives us the theorical tools to compute the optimal forget factor value.

We can show that (see annex B page 8) :

$$K_{opt} = \frac{1}{1 + 1/X} \tag{9}$$

with :

$$X = \frac{\sigma_v^2}{2\sigma_w^2} \left(1 + \sqrt{1 + 4\frac{\sigma_w^2}{\sigma_v^2}} \right) \tag{10}$$

where σ_v and σ_w are the standard deviations of, respectively, the process noise (random walk) and the observation. Note that the ratio $\frac{\sigma_v^2}{\sigma_w^2}$ can be interpreted as the signal to noise ratio (SNR), σ_v determining the evolution of the signal, et σ_w the measurement errors.

In the figure below (figure 3 on page 7), we have plotted the optimal value of K (that is, γ) as a funciton of the SNR $\frac{\sigma_v^2}{\sigma_w^2}$. We can note on this figure some important special cases:

1. If we observe a value that varies slowly, but with a high observation noise ($\sigma_v \ll \sigma_w$), then the optimal gain K_{opt} is low, that can be understood in the sense that asymptoticaly (after a sufficient number of iterations), we know well the value and we don't need to take into account too much the new observations.

2. In the inverse case, if we observe a quickly varying value, and with a low observation noise ($\sigma_v \gg \sigma_w$), then K_{opt} becomes close to 1, meaning that at each iteration, we take in account mostly the last observation and less the preceding ones, which is also logical.

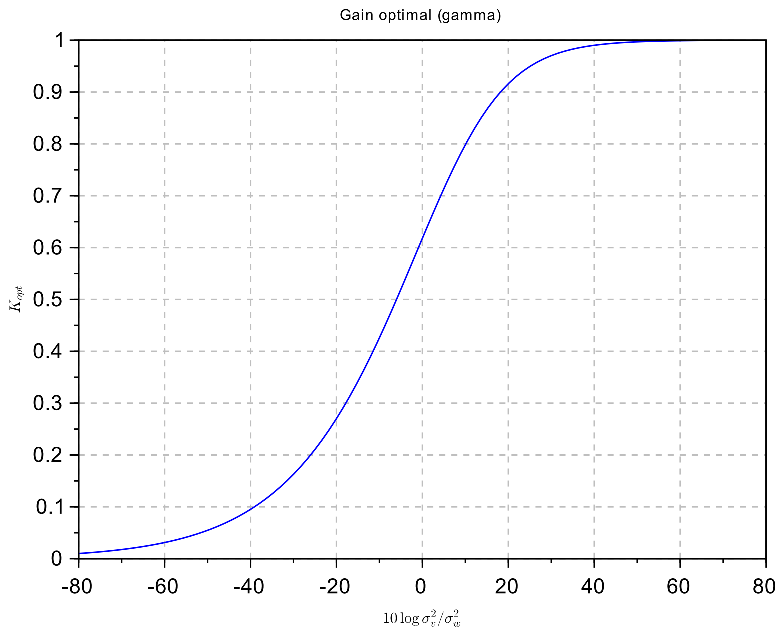

Figure 2: *Optimal gain as a function of the SNR*

Appendices

A Determination of the γ coefficient (for filtering purpose)

The γ coefficient (forget factor) can be expressed as a function of the time constant, by considering the step response of the filter ($x_n = 1, y_0 = 0$):

$$y_{n+1} = y_n + \gamma(1 - y_n)$$

So:

$$y_{n+1} - 1 = y_n(1-\gamma) + \gamma - 1 = (1-\gamma)(y_n - 1)$$

And so:

$$y_n = 1 - (1-\gamma)^n \tag{11}$$

For an analogical RC filter, the response is of the form: $y(t) = 1 - e^{-t/\tau}$, where is the τ **filter time constant**, and by identifying $t = nT_s$ (with $T_s = 1/f_s$: sampling period), we get:

$$(1-\gamma)^n = e^{-nT_s/\tau}$$

Thus :

$$n\log 1 - \gamma = -nT_s/\tau$$

And so:

$$\boxed{\gamma = 1 - e^{-T_s/\tau}} \tag{12}$$

Alternatively, we can also compute the coefficient as a function of the **desired cut-off frequency** $f_c = \frac{1}{2\pi\tau}$:

$$\boxed{\gamma = 1 - e^{-2\pi T_s f_c} = 1 - e^{-2\pi f_c/f_s}} \tag{13}$$

Inversely, we can also compute the cut-off frequency as a function of the forget factor:

$$f_c = \frac{-f_s \log(1-\gamma)}{2\pi} \tag{14}$$

B Computing the optimal gain for a random walk

In the general case, the Riccati equation (discret time case) is written (see [1]):

$$X = Q + AXA^T - A\left(I + \frac{1}{XC^TR^{-1}C}\right)^{-1} XA^T \tag{15}$$

and then :

$$K_{opt} = \frac{XC^T}{CXC^T + R} \tag{16}$$

In our very simple case (scalar), $A = C = 1$, $Q = \sigma_v^2$ and $R = \sigma_w^2$, and so the equation (15) becomes:

$$\sigma_v^2 - X\frac{1}{1 + \frac{\sigma_w^2}{X}} = 0 \Leftrightarrow \sigma_v^2 = \frac{X^2}{\sigma_w^2 + X} \Leftrightarrow X^2 - \sigma_v^2 X - \sigma_v^2\sigma_w^2 = 0$$

So:

$$X = \frac{\sigma_v^2}{2}\left(1 + \sqrt{1 + 4\frac{\sigma_w^2}{\sigma_v^2}}\right) \tag{17}$$

and:

$$K_{opt} = \frac{1}{1 + \sigma_w^2/X} \tag{18}$$

C References

[1] I. RIBEIRO, *Kalman and Extended Kalman Filters: Concept, Derivation and Properties*, 2004