Developing an appropriate PID controller for RTVC using a Kalman filter for state estimation

Oliver Arend

2017–12

1 Model development

Typically, a system is designed to operate in a certain state described by one or more state variables x. The desired state is described by the setpoint x_0, and the difference between these two values is the error e. A controller is then designed to give an input u to the system such that the error is minimized. Such a system is shown in figure 1. The behavior of the system is not completely deterministic since some unknown process noise w is acting on the system, hence the need for a controller.

In our case, the system is basically comprised of two subsystems, considering one axis of yaw/pitch:

1. The servo being controlled by the controller through the command u and accordingly exerting a torque M on
2. the rocket, which has certain known characteristics such as its moment of inertia J, but is also subject to unknown disturbances (process noise) from wind or lack of detail in the model.

Such a system is shown in figure 2.

2 PID controller design

A PID controller is described by three values K_P, K_I and K_D determining the output u as a function of the error e (in continuous form):

$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \tag{1}$$

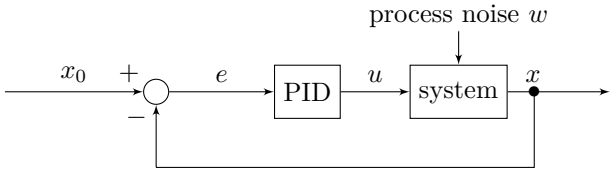


Abbildung 1: Simple system including a PID controller

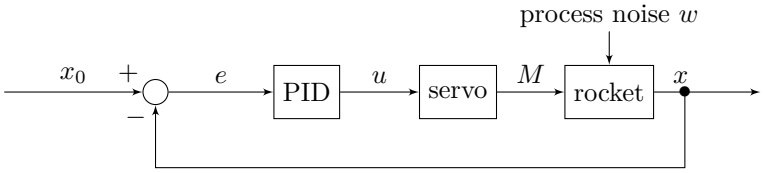


Abbildung 2: Separating complete system into servo and rocket subsystems

i. e. it takes into account

- the current magnitude of the error $e(t)$ at time t and producing a proportional output with a proportionality factor K_P – the larger the error, the larger the corrective action. This in turn means that the error would never disappear entirely, hence the need for an
- integral output. If the error accumulates over time $\int e(t)dt$ the corrective action will be greater, with a factor of K_I.
- The last part is an output proportional to the current rate of change of the error $\frac{de(t)}{dt}$ with a factor of K_D; if it changes more quickly, the corrective action will need to be greater.

The values K_P, K_I and K_D were determined empirically using the SCILAB model shown in figure 3. It includes an environment where airspeed V increases linearly from 0 to 10 m/s in 0.4 s with an angle of attack $\alpha = \min\left(1.2, \frac{2\ \mathrm{m/s}}{V}\right)$ rad.

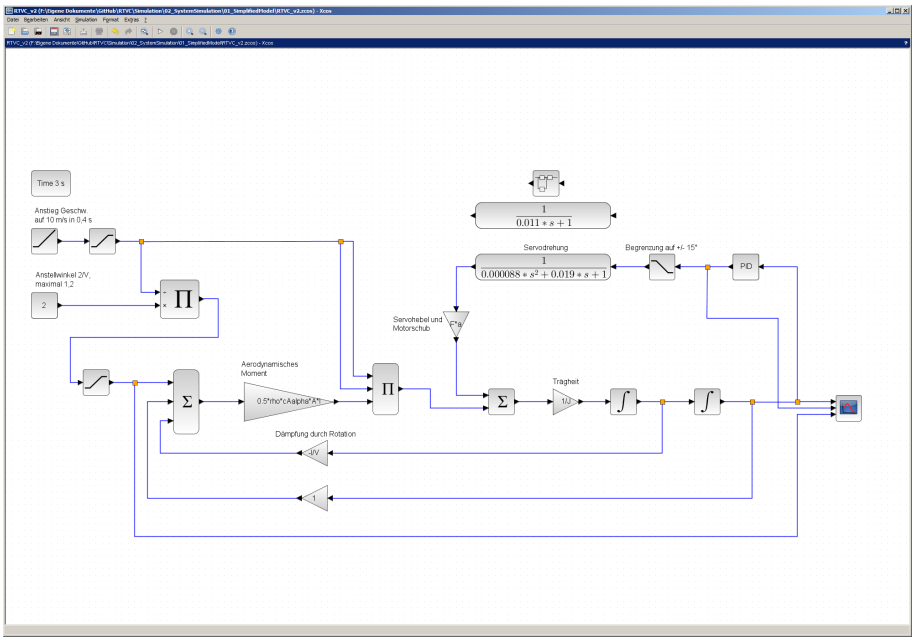


Abbildung 3: SCILAB model used for PID controller design

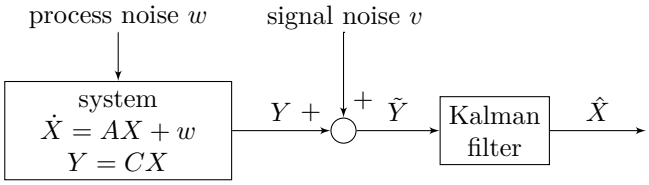


Abbildung 4: Typical Kalman filter approach

Through this angle of attack the rocket, in particular the nosecone, creates an aerodynamic torque

$$M_{aero} = \frac{1}{2}\rho V^2 c_{A_\alpha} \alpha A \ell \tag{2}$$

around its yaw/pitch axis. Air density is assumed standard conditions $\rho = 1.225\ \mathrm{kg/m^3}$, the coefficient of lift gradient is $c_{A_\alpha} = 2$ for nosecones, the reference section is $A = 0.00353\ \mathrm{m^2}$ for a rocket diameter of 67 mm and the moment arm $\ell = 0.1$ m as the distance between the nosecone's center of pressure and the rocket's center of gravity. The torque translates into an angular acceleration $\ddot{\gamma} = \frac{M_{aero}}{J}$ of the rocket's path γ with the rocket's inertia $J = 8.1 \cdot 10^{-4}\ \mathrm{kg\,m^2}$, integrated twice to yield path angular velocity $\dot{\gamma}$ and path angle, respectively. The angular velocity results in an effective reduction of the angle of attack through moment arm and velocity $\Delta\alpha = -\frac{\ell}{V}\dot{\gamma}$.

The PID controller factors yielding a satisfactory behavior of the rocket as described above are

$$K_P = -5 \tag{3}$$
$$K_I = -5 \tag{4}$$
$$K_D = -1 \tag{5}$$

with the output as the setpoint for a servo motor controlling the angle φ (in the same unit as the path angle of the rocket) of the rocket motor relative to the longitudinal axis of the rocket.

The servo of course does not move instantaneously, but the reaction to a change in the setpoint is very similar to a PT2 element with the transfer function

$$G(s) = \frac{1}{0.000088s^2 + 0.019s + 1} \tag{6}$$

corresponding to the two time constants $T_1 = 0.008$ s and $T_2 = 0.019$ s. This behavior can also be described in the state space as

$$\dot{X} = \begin{pmatrix} 0 & 1 \\ -11364 & -216 \end{pmatrix} X + \begin{pmatrix} 0 \\ 11364 \end{pmatrix} u \tag{7}$$

with $X = \begin{pmatrix} \varphi \\ \dot{\varphi} \end{pmatrix}$ and u the servo angle setpoint.

3 Kalman filter design

A Kalman filter can be used to estimate the current state of a system by using knowledge about the behavior of the system, the quality of the measurement

and of course the measurements itself. Based on this, for every timestep the filter will first predict the current state of the system and then correct this estimate based on a measurement and the prediction's certainty. The behavior of a system can be described as

$$\dot{X} = AX + BU + w \tag{8}$$
$$Z = CX + v \tag{9}$$

with the state of the system X, commands U, process noise w (which includes uncertainties about the behavior of the system and unknown outside influences), measurements Z, measurement noise v (which is more or less known) and matrices A, B, C and D describing the relationships between these. From a time-discrete perspective, this can also be written as

$$X_i = AX_{i-1} + BU_i + w_{i-1} \tag{10}$$
$$Z_i = HX_i + v_i \tag{11}$$

The Kalman filter uses this information to estimate the system's state using two steps:

Prediction In this step the current state will be predicted from the last estimate according to the system's behavior

$$\hat{X}_i^- = A\hat{X}_{i-1} + BU_i \tag{12}$$
$$P_i^- = AP_{i-1}A + Q \tag{13}$$

with the system error covariance matrix Q and the error covariance P. Q is unknown and can only be estimated. P can be initialized to the identity matrix.

Correction The predicted state will now be corrected according to the measured values describing (part of) the state:

$$K_i = P_i^- H^T (HP_i^- H^T + R)^{-1} \tag{14}$$
$$\hat{X}_i = \hat{X}_i^- + K_i(Z_i - H\hat{X}_i^-) \tag{15}$$
$$P_i = (I - K_i H)P_i^- \tag{16}$$

A rough sketch of how a Kalman filter estimates the system's state from its measurement output(s) is shown in figure 4.

3.1 Model rocket apogee detection

As a simple first example, the Kalman filter is applied to a model rocket flying up and whose apogee should be detected. The state of the rocket can be described as

$$X = \begin{pmatrix} h \\ v \\ a \end{pmatrix} \tag{17}$$

with altitude h, (vertical) velocity v and (vertical) acceleration a. The evolution from one state to the next is done through simple integration with a time

constant Δt. No commands U are imparted to the rocket, and the measurement is simply the altitude h, with a measurement noise covariance of $R = 0.25$. The system variables are in basic SI units (meters and seconds). The relevant matrices are thus

$$A = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}, \qquad B = 0 \tag{18}$$

$$C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \qquad D = 0 \tag{19}$$

Data was created with a time increment of 0.01 s from a rocket flight simulation, with additional noise added to the altitude measurement. The results from `kalman_rocketflight_altitude.py` are shown in figure 5a. The measured altitude is shown in light blue, and its estimate in orange. The filter reduces the noisiness quite a bit, but does not completely eliminate it ($Q = 0.01I$ was chosen). Velocity (true red, estimate green) and acceleration (true dark purple, estimate light purple) were also estimated, but do not match the true values closely. Only apogee detection, with $v = 0$, would work reliably.

Results can be improved considerably using an accelerometer, hence changing the measurement matrix to

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{20}$$

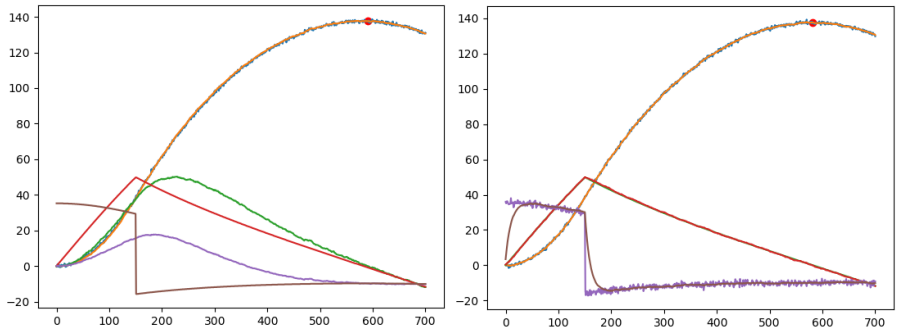
and the measurement covariance matrix to

$$R = \begin{pmatrix} 0.25 & 0 \\ 0 & 1 \end{pmatrix}, \tag{21}$$

assuming independent error. The results from `kalman_rocketflight_altitude_accel.py` are shown in figure 5b. Velocity matches very well, while the acceleration estimate takes some time to converge to the true value.

3.2 Simple model rocket attitude control

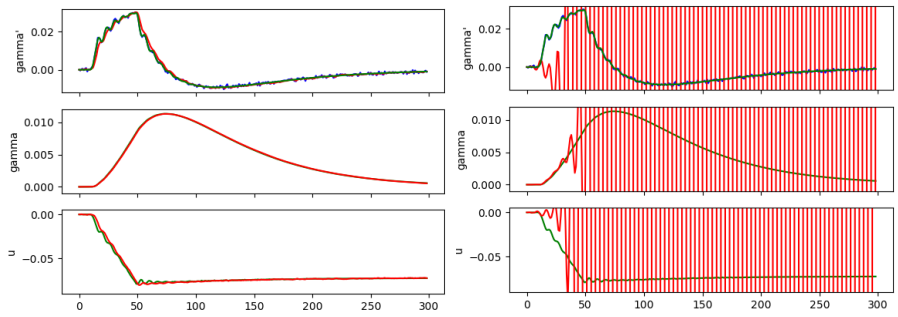
The actual application of the Kalman filter towards an active control of the rocket's attitude involves measuring the rocket's angular velocity and estimating



(a) From altitude measurement only

(b) From both altitude and acceleration measurements

Abbildung 5: Estimating altitude and apogee



(a) Ignoring u ($B = 0$)

(b) Translating u directly to angular acceleration

Abbildung 6: Estimating the rocket's attitude using only path angle and derivatives as state

its path angle, which should be kept at or very near 0. It is simplified to a single pitch/yaw axis, because both axes can be regarded as independent of each other.

A simple model would only involve a state representing the rocket's attitude and its evolution

$$X = \begin{pmatrix} \gamma \\ \dot{\gamma} \\ \ddot{\gamma} \end{pmatrix} \tag{22}$$

with the path angle γ and its time derivatives. Reusing the SCILAB model to design the PID controller (see section 2), noisy data representing the angular velocity $\dot{\gamma}$ is created. For comparison, clean data for path angle γ, angular velocity and the corrective command to the servo u is also exported.

Two approaches were used:

- completely ignoring the servo command in the prediction, such that

$$A = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{23}$$

- and directly translating the setpoint of the servo u to an angular acceleration such that

$$A = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ \frac{Fa}{J} \end{pmatrix} \tag{24}$$

  with the motor's thrust F, the moment arm of the motor a and the rocket's angular inertia J.

The results of both approaches are shown in figure 6. Obviously the approach considering the torque applied by the rotating motor leads to a divergent estimate, see figure 6b, since the servo does not actually move that fast – the timestep in the simulation is 0.01 s, whereas the time constant of the servo is about 0.019 s. Ignoring u, as shown in figure 6a, leads to a decent estimate of the path angle γ, but some error and phase offset in angular velocity $\dot{\gamma}$ and the servo command u.

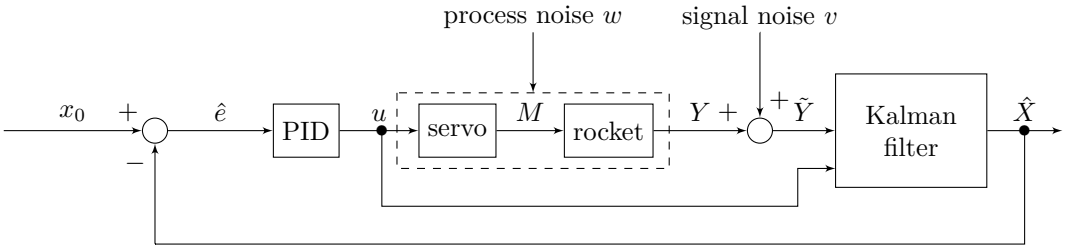


Abbildung 7: Full model including PID controller, Kalman filter and system separation

3.3 Complex attitude estimate and control

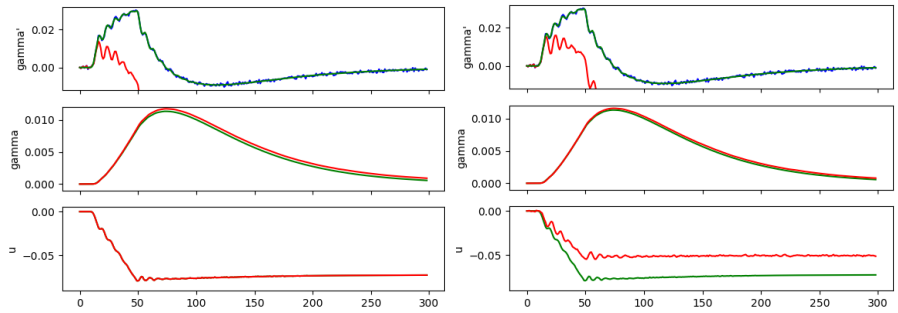
A possible approach to improve the state estimate and thus the control signal is to include the servo's rotation in the state

$$X = \begin{pmatrix} \gamma \\ \dot{\gamma} \\ \ddot{\gamma} \\ \varphi \\ \dot{\varphi} \end{pmatrix} \tag{25}$$

and to have the servo behave as described in equation 7. The system and command matrices thus change to

$$A = \begin{pmatrix} 1 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & \frac{Fa}{J} & 0 \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & -11364\Delta t & 1 - 216\Delta t \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 11364\Delta t \end{pmatrix} \tag{26}$$

The measurement matrix simply reflects the enlargement of the state $C = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix}$.



(a) Using saved values for u

(b) Calculating u from state variables during runtime

Abbildung 8: Estimating the rocket's attitude using a more complex state including the servo angle

For comparison, the command u was either taken from the simulation or calculated during runtime of `kalman_attitude_servo.py` from the state variable estimates according to the PID controller constants described in section 2. Keeping the system error covariance as before at $Q = 0.01I$, the path angle estimate diverges slightly, whereas the error on the angular velocity $\dot{\gamma}$ is rather large and, in the case of live calculation of u, results in a significant error. The results are shown in figure 8.

4 Conclusion

Attitude estimation yields better results using a simple model only taking the path angle and its derivatives into account and regarding servo input and other "disturbances" as process noise. The reasons for this are unclear. Of course the SCILAB simulation is not exactly reproduced in the Kalman filter implementation, and the simple $X_{i+1} = \begin{pmatrix} 1 & \Delta t & 0 \end{pmatrix} X_i$ integration will diverge from the original data, which was obtained through integration by more refined methods.

The underlying question, which Kalman filter architecture and which PID controller will be able to successfully control RTVC, could not be answered with sufficient certainty.