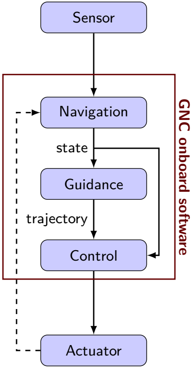
GNC onboard software

GNC onboard software contains three components:

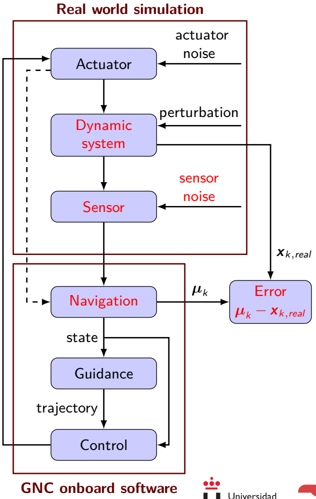
- The **navigation** system estimates the **state** (e.g., position, velocity, orientation, angular velocity).
- The **guidance** system calculates the **reference trajectory** (e.g., velocity and yaw angle for PX4 multicopter).
- The **control** system
  - compares reference trajectory and state;
  - calculates **force and torque setpoints**;
  - generates **control signals to actuators** (e.g., motor speed for multicopter, throttle and servo angle deflection for fixed-wing aircraft).



Universidad Rey Juan Carlos GISAT

GNC design environment

- A typical GNC design environment contains two high-level modules, **GNC onboard software** and **Real-world simulation**. The real-world simulation contains **three simulation models**.
  - The **actuator model** takes the control signals as inputs, adds **actuator noise**, and generates outputs that are **simulated force and torque**.
  - The **dynamic system model** takes the simulated force and torque as inputs, adds **perturbations**, and generates output that is the **simulated real-world state** of the vehicle.
  - The **sensor model** takes the simulated real-world state as inputs, adds **sensor noise**, and generates outputs that are **simulated sensor measurements**.
- The **process noise covariance matrix R** should take into account **actuator noise, perturbation, and sensor noise**. The **measurement noise covariance matrix Q** should take into account **sensor noise**. These two covariance matrices should be **tuned** to improve the KF state estimation performance.



Problem statement

A robot on the floor is moving in a **straight line** with a **constant acceleration**. Its **acceleration and position** are measured by **accelerometer and range finder**, respectively. It is assumed that guidance system, control system, and actuators are perfect, such that the robot is indeed moving with the commanded constant acceleration. There is also no perturbation. The only noises should be taken into account in the noise covariance matrices are the sensor noises. Complete the following tasks

1. Design Kalman Filter for GNC onboard software module.
2. Design dynamic system model and sensor model for real world simulation module.
3. Implement both modules in Matlab with the following reference data:
   - constant time step interval $\Delta t = 0.1\,\mathrm{s}$.
   - Real-world simulation parameters: initial position $p_0 = 1\,\mathrm{m}$, initial velocity $v_0 = 1\,\mathrm{m\,s^{-1}}$, commanded constant acceleration $u_{cmd} = 1\,\mathrm{m\,s^{-2}}$, accelerometer standard deviation $\sigma_{acc} = 1\,\mathrm{m/s^2}$, range finder standard deviation $\sigma_{range} = 1\,\mathrm{m}$.
   - Onboard software parameters: accelerometer standard deviation $\sigma_{acc} = 1\,\mathrm{m/s^2}$, range finder standard deviation $\sigma_{range} = 1\,\mathrm{m}$, initial state estimate $\boldsymbol{\mu}_0 = [10\,\mathrm{m}; \quad 10\,\mathrm{m/s}]$, initial error covariance $\boldsymbol{\Sigma}_0 = [1\,\mathrm{m^2} \quad 0\,\mathrm{m^2/s}; \quad 0\,\mathrm{m^2/s} \quad 1\,\mathrm{m^2/s^2}]$.
4. Modify one of the given data, keep the rest of reference data untouched, observe the change in KF state estimation performance, and explain these changes.
   - Range finder standard deviation in the onboard software module: $0.001\,\mathrm{m}$, $0.1\,\mathrm{m}$, $10\,\mathrm{m}$, and $100\,\mathrm{m}$.
   - Accelerometer standard deviation in the onboard software module: $0.001\,\mathrm{m\,s^{-2}}$, $1\,\mathrm{m\,s^{-2}}$, $10\,\mathrm{m\,s^{-2}}$, and $100\,\mathrm{m\,s^{-2}}$.
   - Increase the error in the initial state estimate: $\boldsymbol{\mu}_0 = [100\,\mathrm{m}; \quad 100\,\mathrm{m\,s^{-1}}]$.

Onboard software module

- Define state $\boldsymbol{x}_k \in \mathbb{R}^{2\times1}$, control $\boldsymbol{u}_k \in \mathbb{R}^{1\times1}$, and measurement $\boldsymbol{z}_k \in \mathbb{R}^{1\times1}$.

$$\boldsymbol{x}_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} \quad , \quad \boldsymbol{u}_k = \begin{bmatrix} a_{k,acc} \end{bmatrix} \quad , \quad \boldsymbol{z}_k = \begin{bmatrix} p_{k,range} \end{bmatrix}$$

- Construct state model ($\boldsymbol{A}_k \in \mathbb{R}^{2\times2}$, $\boldsymbol{B}_k \in \mathbb{R}^{2\times1}$, $\boldsymbol{R}_k \in \mathbb{R}^{2\times2}$).

$$\boldsymbol{x}_k = \boldsymbol{A}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{\epsilon}_k$$

$$\boldsymbol{A}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad , \quad \boldsymbol{B}_k = \begin{bmatrix} \Delta t^2/2 \\ \Delta t \end{bmatrix} \quad , \quad \boldsymbol{R}_k = \boldsymbol{B}_k \sigma_{acc}^2 \boldsymbol{B}_k^T = \sigma_{acc}^2 \begin{bmatrix} \Delta t^4/4 & \Delta t^3/2 \\ \Delta t^3/2 & \Delta t^2 \end{bmatrix} \ .$$

- Construct observation model ($\boldsymbol{C}_k \in \mathbb{R}^{1\times2}$, $\boldsymbol{Q}_k \in \mathbb{R}^{1\times1}$).

$$\boldsymbol{z}_k = \boldsymbol{c}_k \boldsymbol{x}_k + \boldsymbol{\delta}_k$$

$$\boldsymbol{C}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \ , \quad \boldsymbol{Q}_k = \sigma_{range}^2 \ .$$

Real-world simulation module

- Simulate the state.

$$\boldsymbol{x}_{k,real} = \boldsymbol{A}_k \boldsymbol{x}_{k-1,real} + \boldsymbol{B}_k \boldsymbol{u}_{k,cmd}$$

- Simulate range finder and provide measurement history.

$$\boldsymbol{z}_k = \boldsymbol{C}_k \boldsymbol{x}_{k-1,real} + \boldsymbol{\delta}_k$$

- Simulate accelerometer and provide control history.

$$\boldsymbol{u}_k = \boldsymbol{u}_{k,cmd} + \boldsymbol{\epsilon}_k$$

- Hint: use *randn* function to generate random numbers with normal distribution.

Matlab simulation

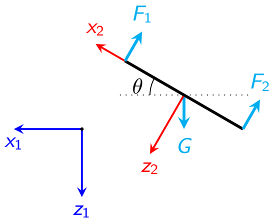
Complete the matlab codes:

- "main.m"
  - Provide the numerical values of the parameters required to carry out the simulation and KF algorithm;
  - Plot the results. You can modify the plot functions as you prefer.
- "KF_straight.m":
  - Simulate the real world;
  - KF algorithm to update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

Result analysis

- Compare KF estimated position, real-world simulated position, and range finder measurement.
- Compare KF estimated velocity and real-world simulated velocity.
- Observe how innovation varies.
- Observe how the error covariance matrix varies.

Problem statement



We will design an EKF for a 2D quadcopter moving in a **straight line at a constant altitude**. Its **angular velocity and position** are measured by **rate gyro and GPS**, respectively. The state, control, and measurement vectors are defined as

$$\boldsymbol{x}_k = \begin{bmatrix} p_k = x_{k,1} \\ v_k = \dot{x}_{k,1} \\ \theta_k \end{bmatrix}, \quad \boldsymbol{u}_k = \begin{bmatrix} \dot{\theta}_{k,gyro} \end{bmatrix}, \quad \boldsymbol{z}_k = \begin{bmatrix} p_{k,gps} \end{bmatrix}.$$

It is assumed that guidance system, control system, and actuators are perfect, such that the robot is indeed moving with the commanded control, that is a **constant angular velocity**. There is also no perturbation. The only noises should be taken into account in the noise covariance matrices are the sensor noises. Complete the following tasks.

- Design Extended Kalman Filter for GNC onboard software module.
- Design dynamic system model and sensor model for real world simulation module.
- Implement both modules in Matlab with reasonable reference data that you will select for the test environment.
- Analyse the result.

Dynamic system model

- Second order differential equations **with respect to the observation frame (fr.)**

$$\boldsymbol{F} = \left.\frac{\mathrm{d}(m\boldsymbol{v})}{\mathrm{d}t}\right|_{\text{fr. observation}} = m\boldsymbol{a}$$ expressed in the **observation-fr vector basis**

$$\boldsymbol{M}_G = \left.\frac{\mathrm{d}(\boldsymbol{I}_G\boldsymbol{\omega})}{\mathrm{d}t}\right|_{\text{fr. observation}} = \left.\frac{\mathrm{d}(\boldsymbol{I}_G\boldsymbol{\omega})}{\mathrm{d}t}\right|_{\text{fr. body}} + \boldsymbol{\omega} \times \boldsymbol{I}_G\boldsymbol{\omega}$$ expressed in the **body-fr vector basis**

- First order differential equations

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \quad \Rightarrow \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} f_1(x_1, \dots, x_n, u_1, \dots, u_m) \\ f_2(x_1, \dots, x_n, u_1, \dots, u_m) \\ \vdots \\ f_n(x_1, \dots, x_n, u_1, \dots, u_m) \end{bmatrix}$$

- Discretization and approximation of integration

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) \approx \boldsymbol{x}_{k-1} + \dot{\boldsymbol{x}}_{k-1}\Delta t$$

Submission

- 2 students a group.
- Zip folder with name "KF_Group_XX".
  - Matlab codes for both exercises.
  - One presentation that you will use to explain your **model** and **results** in the oral exam.
- Submission deadline: 23:55, Monday, 25 of December.