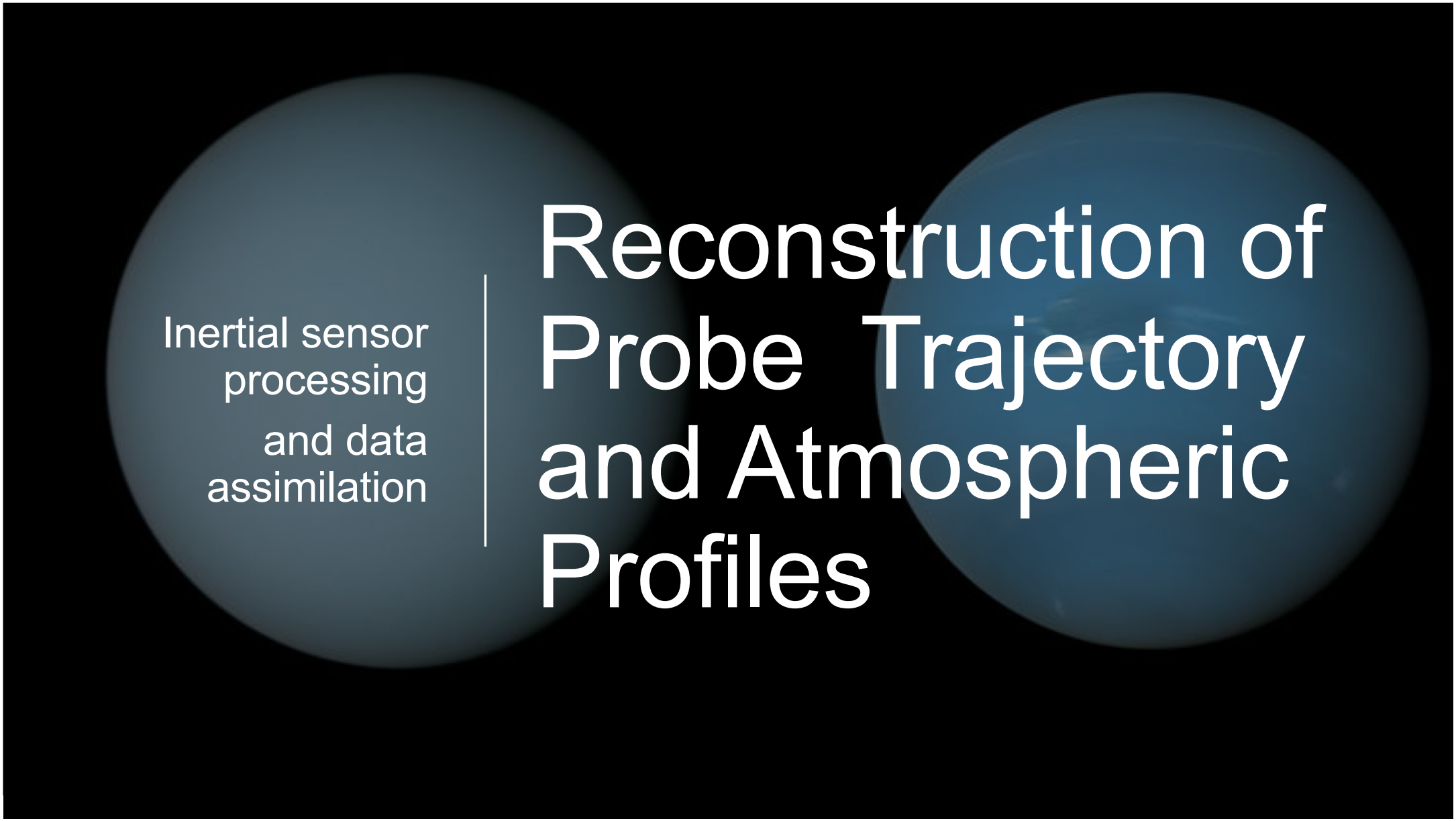
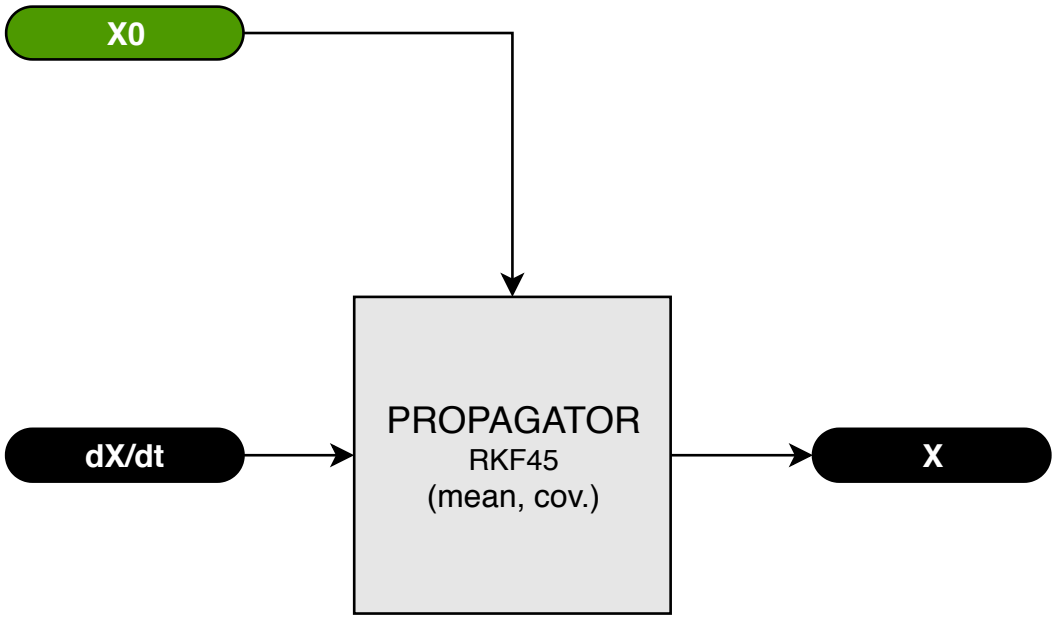

Inertial sensor processing and data assimilation
Reconstruction of Probe Trajectory and Atmospheric Profiles

State Propagation

Basic building block of more complex algorithms

- Several integration algorithms can be used: RK4, RKF45, Euler (pay attention to attitude)
- Specify:
  - State vector X
    (i.e. the parameters of interest)
  - State equations f()
    (i.e. how the state vector evolves with time)



$$\dot{x} = f(x, u)$$

State Propagation With Uncertainty

State vector modeled as Gaussian r.v.

- State vector is composed by
  - Mean
  - Covariance matrix
    (upper triangular part stacked by row)
- Mean evolves according to state equations
- Covariance propagates according to Ricatti equation
- Specify:
  - A: jacobian of state eq. wrt to the state vector
  - B: jacobian of state eq. wrt to the noise sources (e.g. inputs)

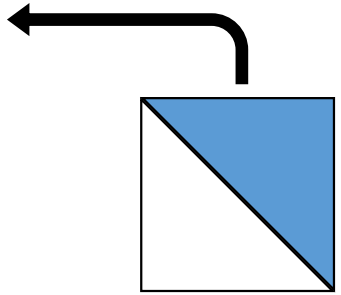
$$X \begin{cases} \text{Mean} \\ \text{Covariance} \end{cases}$$

$$\dot{m} = f(m, u)$$

$$\dot{P}_x = A\,P + P\,A^T + B\,Q\,B^T$$

$$A = \nabla_x f(m, u)$$

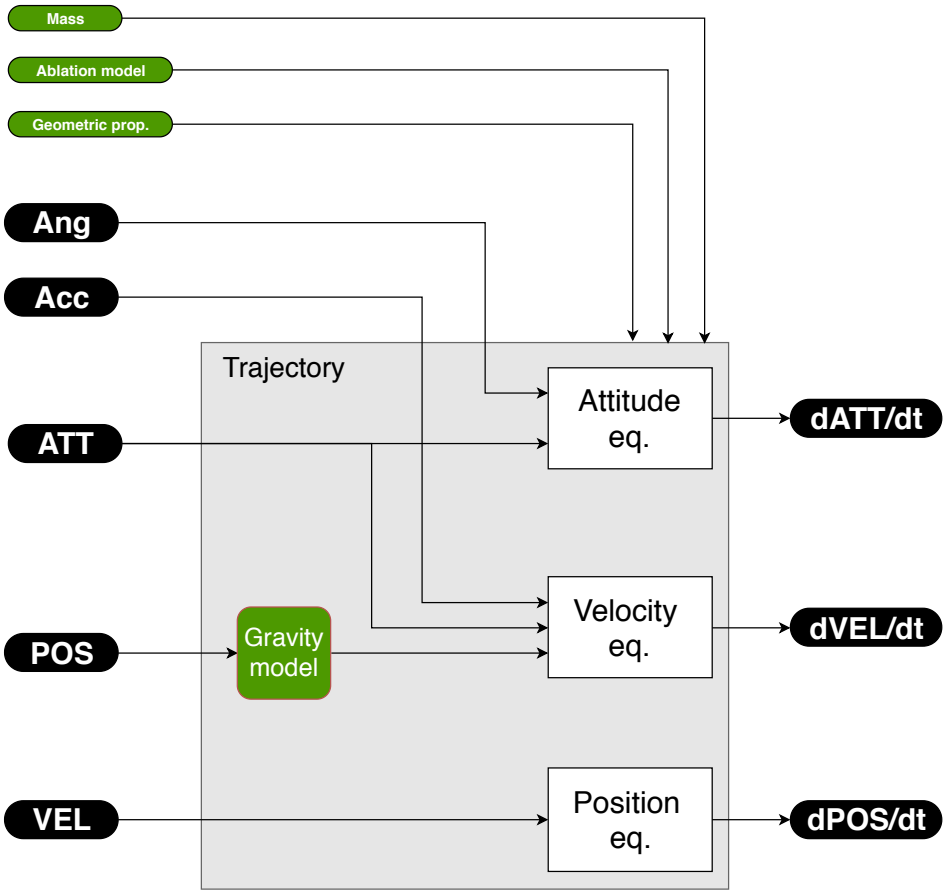
$$B = \nabla_u f(m, u)$$



Trajectory Reconstruction

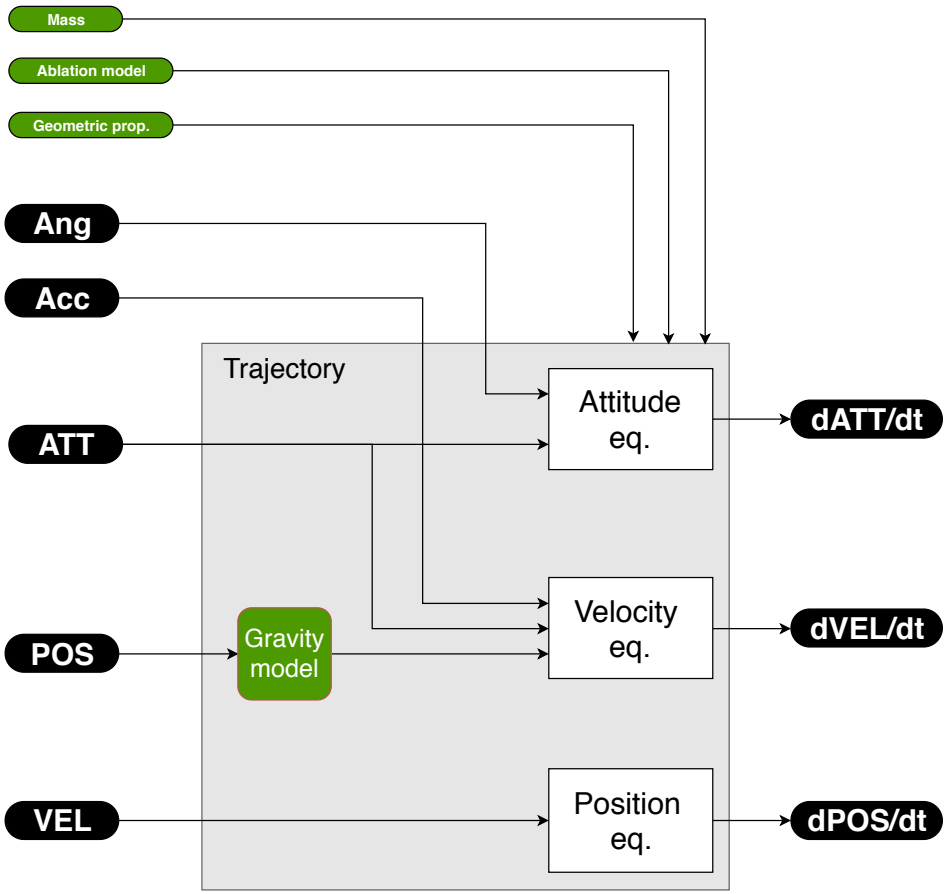
State model for the direct integration of inertial data

- Specify:
  - Reference frames
  - Attitude parametrization (quaternions are the default choice)
  - Gravity model
  - Probe mass/ablation models/other geometric properties



Trajectory Reconstruction

State model for the direct integration of inertial data



$$\dot{q} = K(q)\, w$$

$$\dot{v} = a_g + C^T(q)\, a_a$$

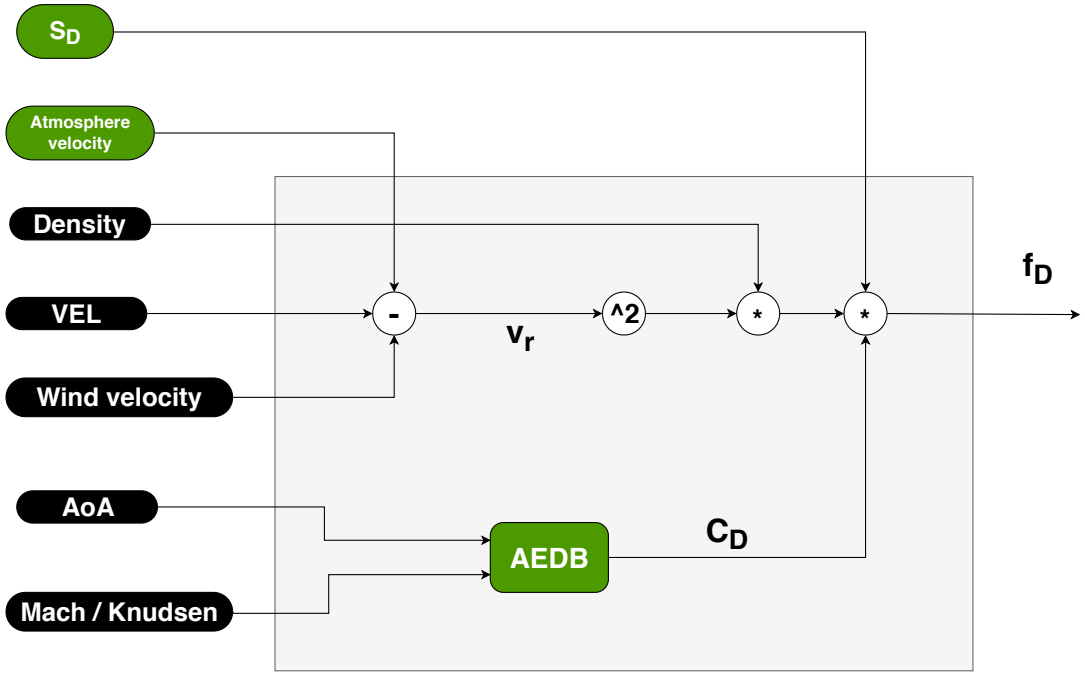
$$\dot{p} = v$$

Drag Equation

Links measured acceleration with atmospheric density

- Atmospheric reconstruction is in any case based on the probe AEDB
- Relative velocity v_r is function of winds
- Drag coeff. CD is a function of
  - Angle of attack
  - Mach/Knudsen numbers

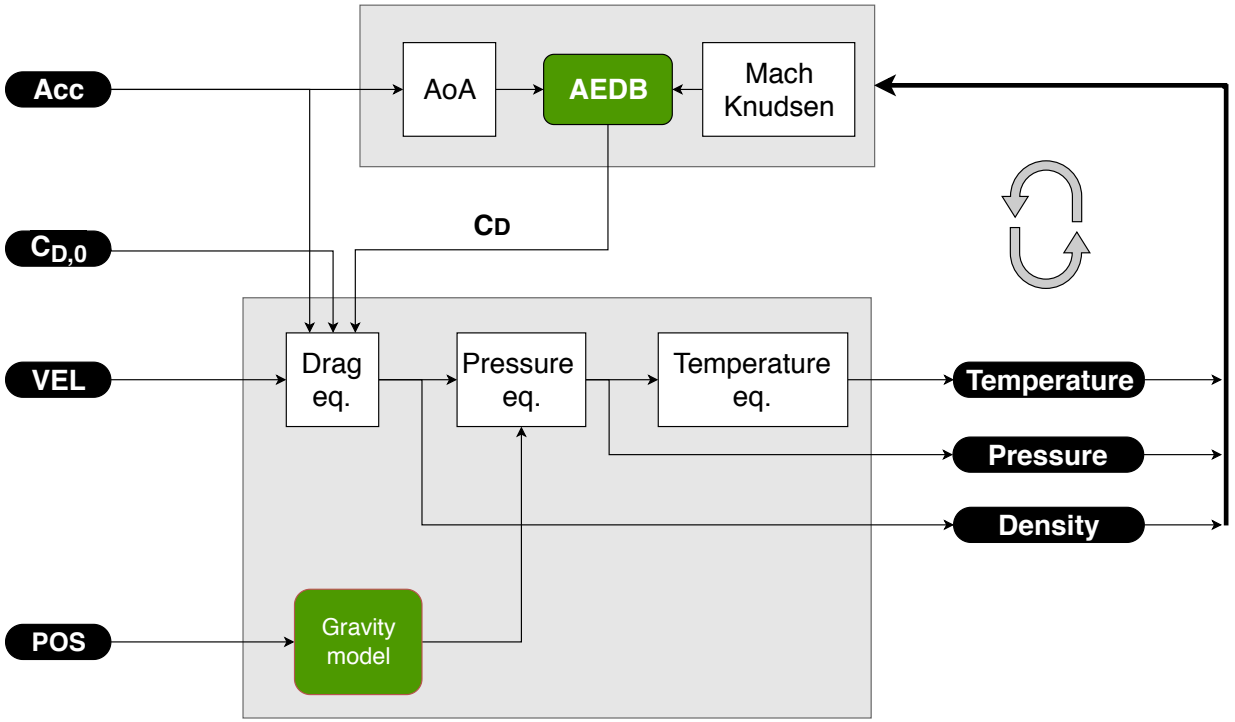
$$f_D = \frac{1}{2}\rho\, v_r^2\, C_D\, S_D$$



Atmospheric Profiles Reconstruction

Direct reconstruction

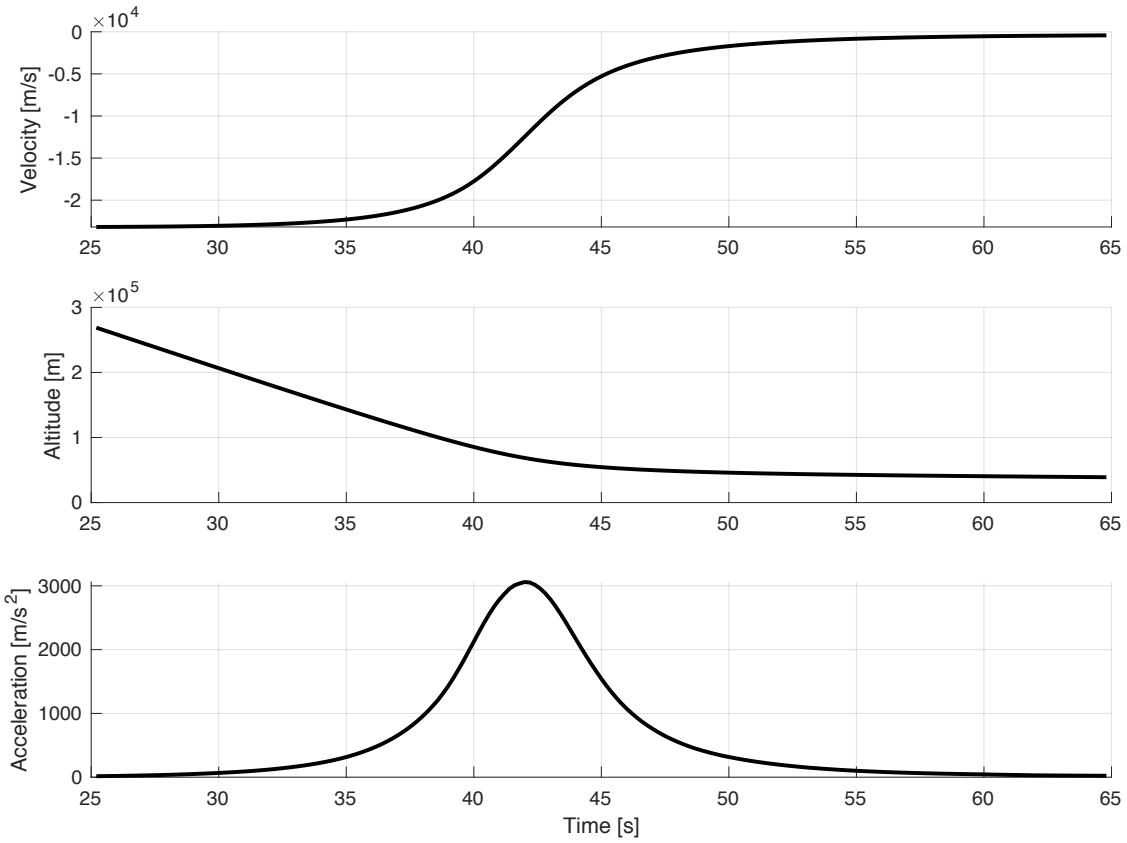
- Assumptions:
  - A-priori atmospheric profiles used to initialize the algorithm (compute $C_{A,0}$)
  - Zero winds
  - Hydrostatic equilibrium
  - Perfect gas law
- Outer loop:
  - Update atmosphere wrt acceleration
  - Update acceleration (C_A) wrt to atmosphere
- Inner loop computes the pressure integrating estimated density

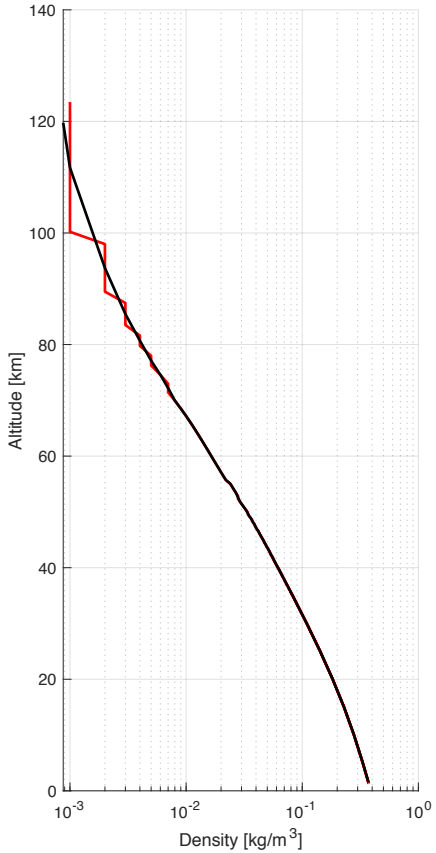


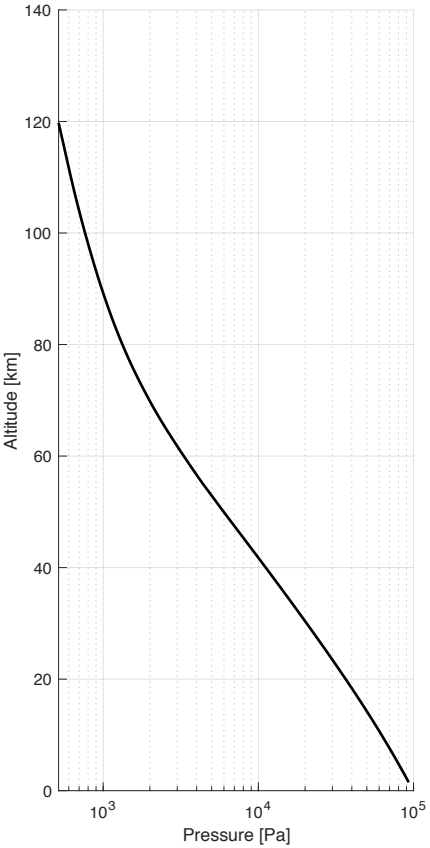
$$\rho = \frac{2\,m\,a}{v_r^2\,C_D\,S_D} \qquad \frac{dP}{dh} = -\rho\,g \qquad T = \frac{P\,M}{\rho\,k_B\,N_A}$$

Neptune Entry Simulation

1D entry simulation with velocity/altitude/density/pressure similar to a Neptune mission

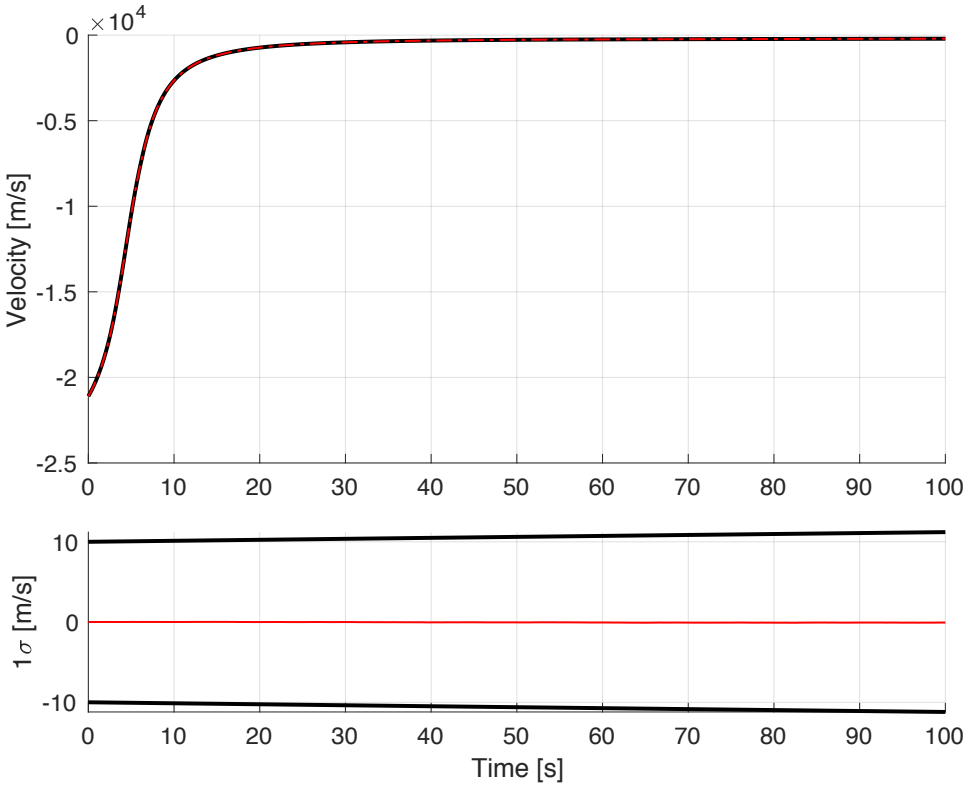


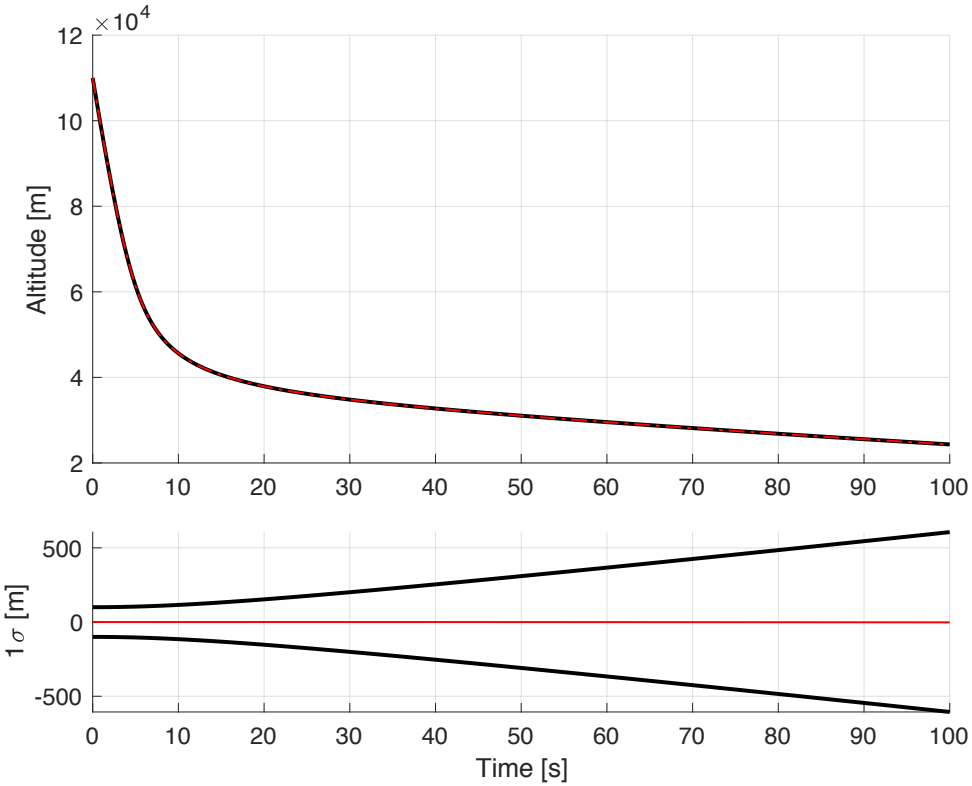




Trajectory Reconstrution Example

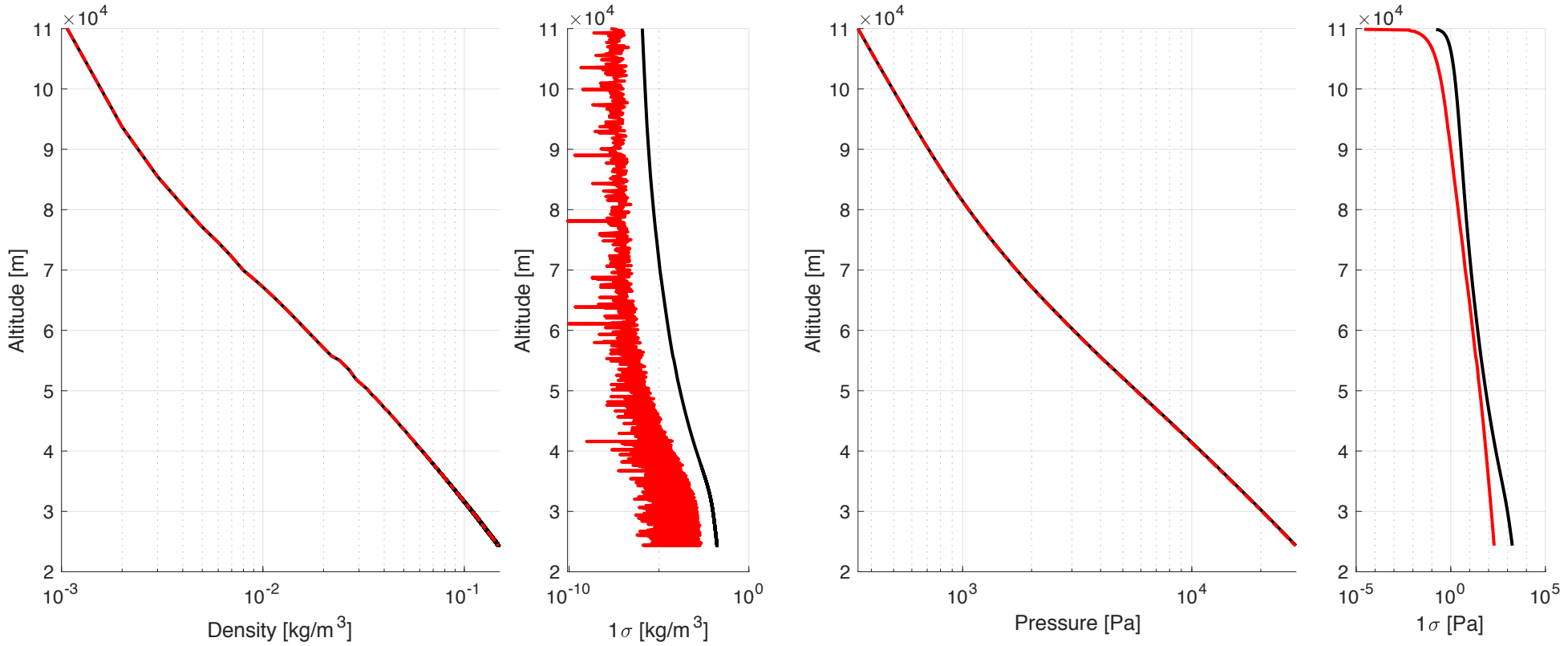
Direct reconstruction of entry trajectory





Atmosphere Reconstruction Example

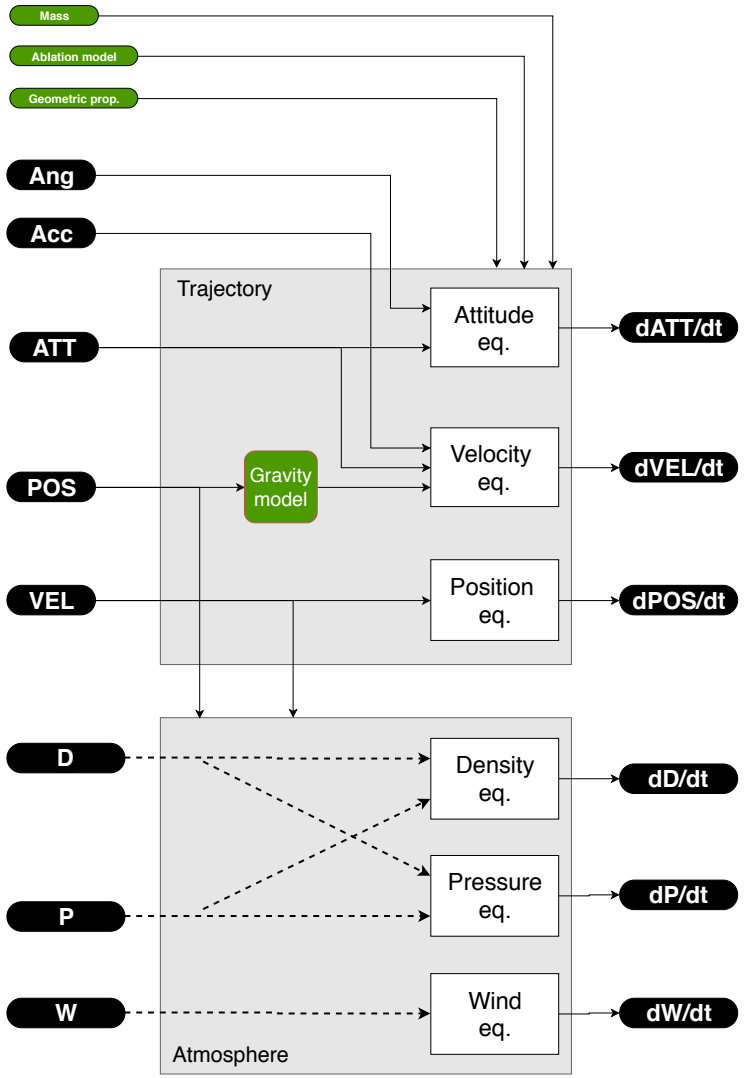
Direct reconstruction of atmospheric profiles



Augmented State Model

Adding atmospheric parameters to state

- Density, pressure and (optionally) winds are added to the state vector
- Specify changes wrt time:
  - Density state eq.
  - Pressure state eq.
  - Winds state eq. (optional)
- Several design choices are possible!



Density, Pressure and Winds modeling

A

$$\dot{\rho} = 0$$

$$\dot{p} = -g\,\rho\,v$$

$$\dot{w} = 0$$

- No constraints on density
- Hydrostatic eq. used for pressure
- No constraints on winds
  (winds estimation very poor, possible numerical issues)

B

$$\dot{\rho} = \alpha\,\rho$$

$$\dot{p} = -g\,\rho\,v$$

$$\dot{w} = 0$$

- Smoothness constraint on density (depending on alpha)
- Hydrostatic eq. used for pressure
- No constraints on winds
  (winds estimation very poor, possible numerical issues)

C

$$\dot{\rho} = \frac{\mu\,\rho^2\,v}{r^2\,p}$$

$$\dot{p} = \frac{\mu\,\rho\,v}{r^2}$$

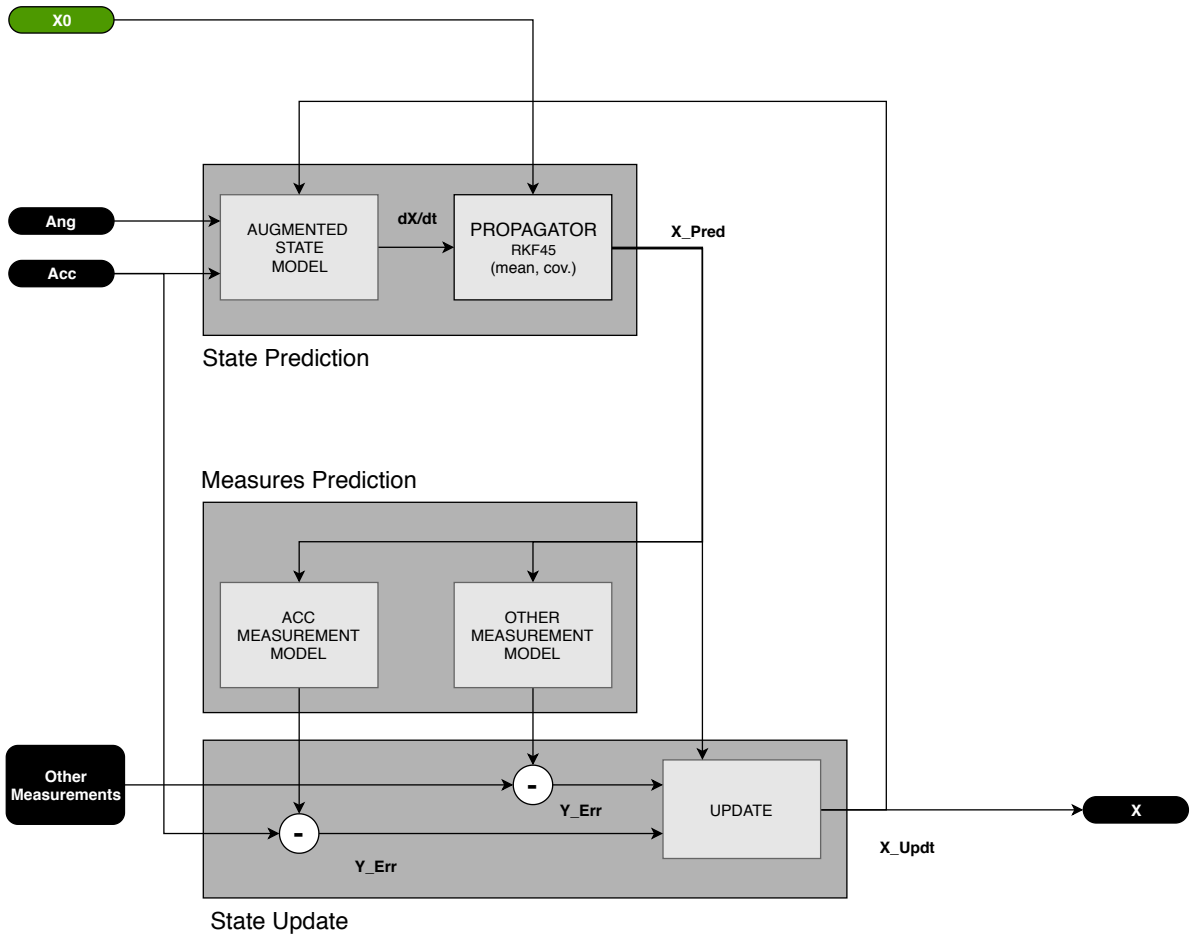
$$\dot{w} = 0$$

- Density and pressure both constrained by
  - hydrostatic eq.
  - ideal gas law
- No constraints on winds

Bayesian Data Assimilation

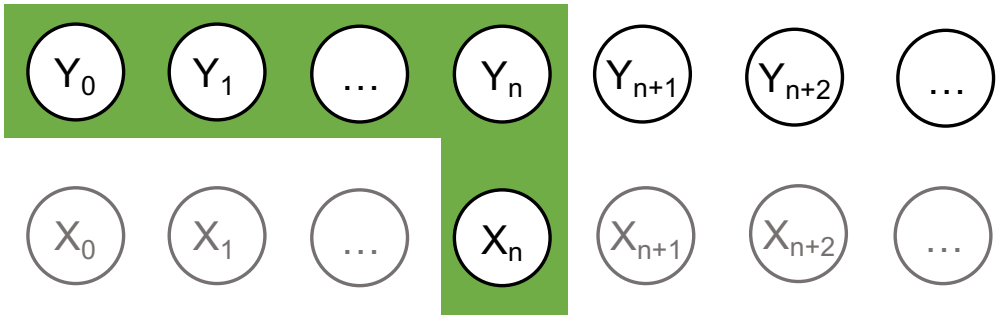
Update the state vector using the available measurements

- Several different sensors can be used to update the state
- For each sensor specify:
  - Sensor model function h()
  - The Jacobian matrix H()
- The innovation, i.e. the error between predicted measurements and the actual values used to check the results
  (it should be white noise)



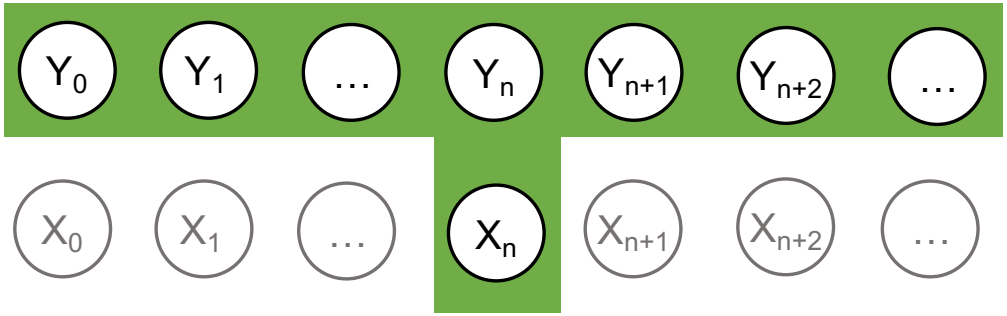
State Update, Filtering, Smoothing

Filtering



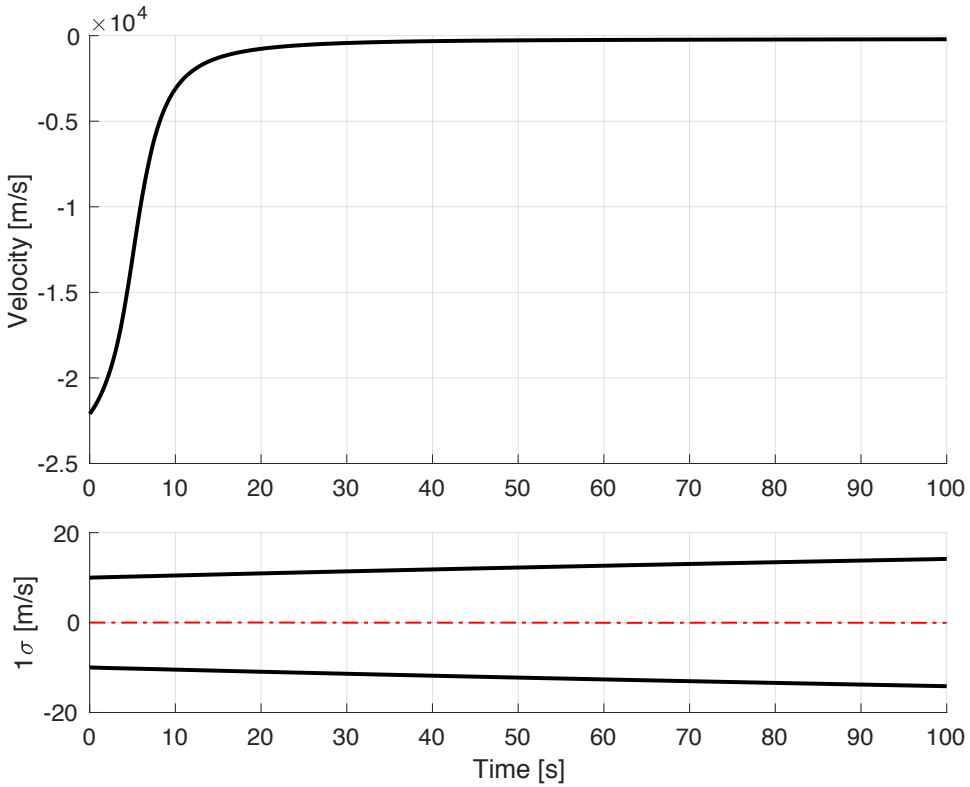
$$v_n = y_n - h(m_n^-)$$
$$S_n = H(m_n^-)\, P_n^-\, H^T(m_n^-) + R_n$$
$$K_n = P_n^-\, H^T(m_n^-)\, S_n^{-1}$$
$$m_n = m_n^- + K_n\, v_n$$
$$P_n = P_n^- - K_n\, S_n\, K_n^T$$

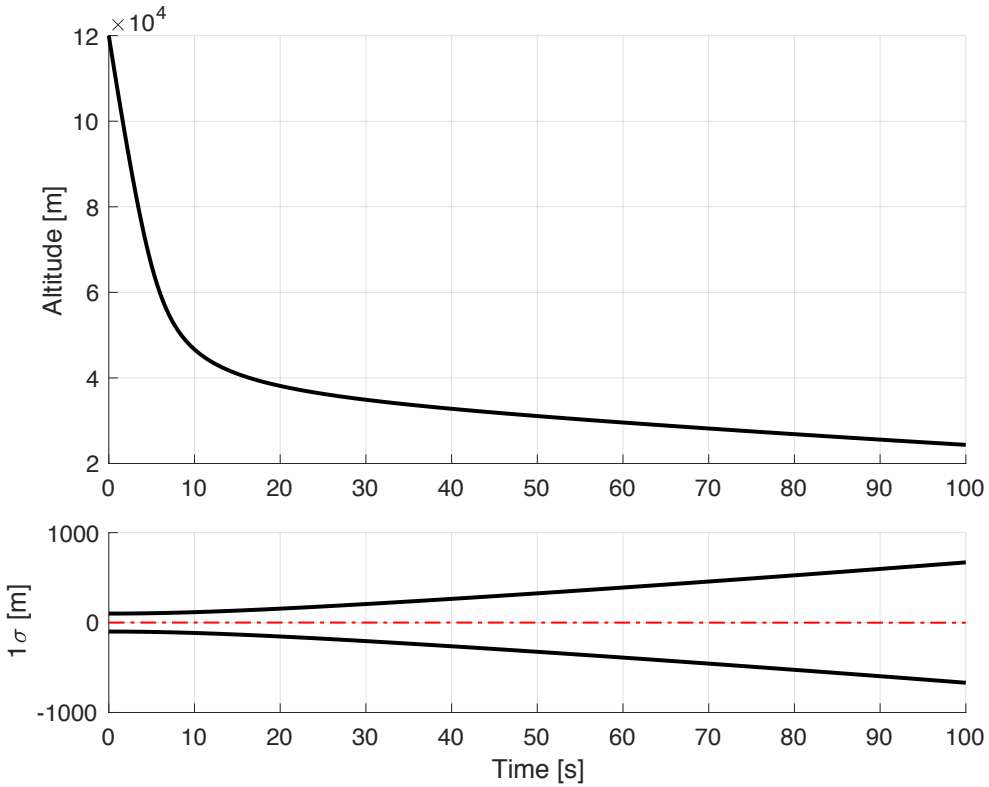
Smoothing



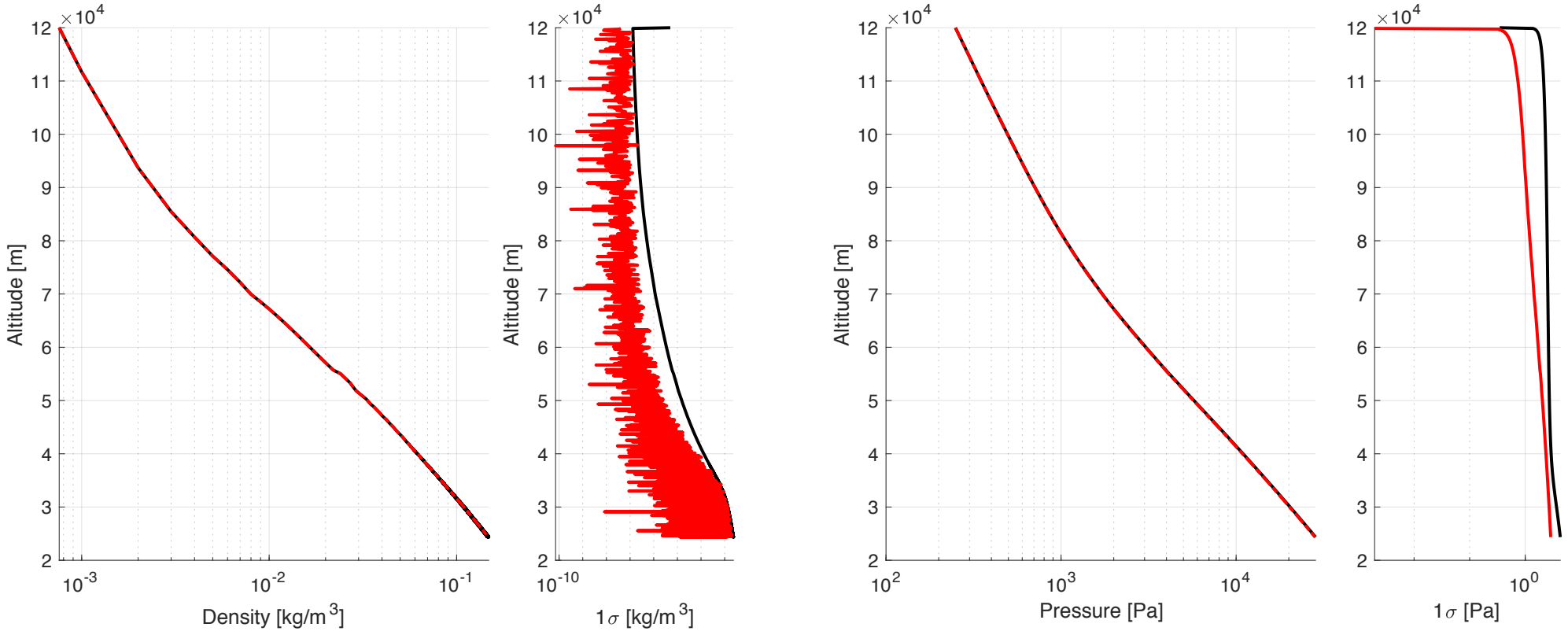
$$G_n = P_n\, A^T(m_n)\, \left[P_{n+1}^-\right]^{-1}$$
$$m_n^s = m_n + G_n\, \left[m_{n+1}^s - m_{n+1}^-\right]$$
$$P_n^s = P_n + G_n\, \left[P_{n+1}^s - P_{n+1}^-\right]\, G_n^T$$

Model C Trajectory



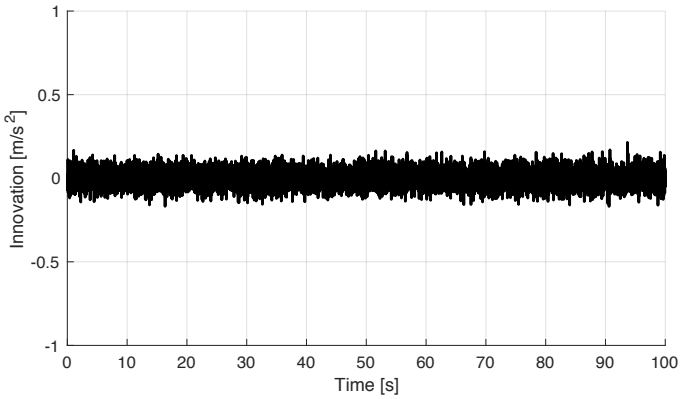


Model C Atmospheric Profiles



Innovation Checking

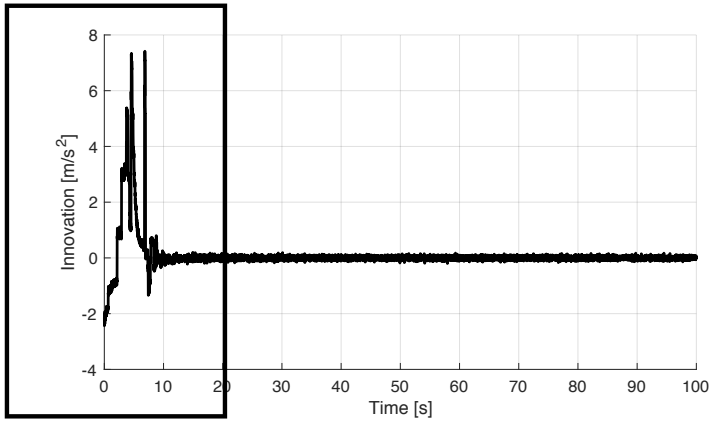
Innovation should be white noise, meaning that all the information in the measurement has been processed



In the 'real world' innovation is not exactly white

It is usefull tool because it provides

- evidence of unmodelled effects
- the need of more accurate 'tuning' of the algorithm (noise PSD)



Conclusions

Keep it simple

- Start from direct integration methods
- Then move to more complex algorithms
  - Requiring implementation effort
  - And parameter tuning
  - EKF could provide good solutions
    (it is the base for more advanced estimators)
- Include sensors step by step
  - Check how the results changes/improves
  - Check innovation

Focus on modelling and be aware of assumptions!