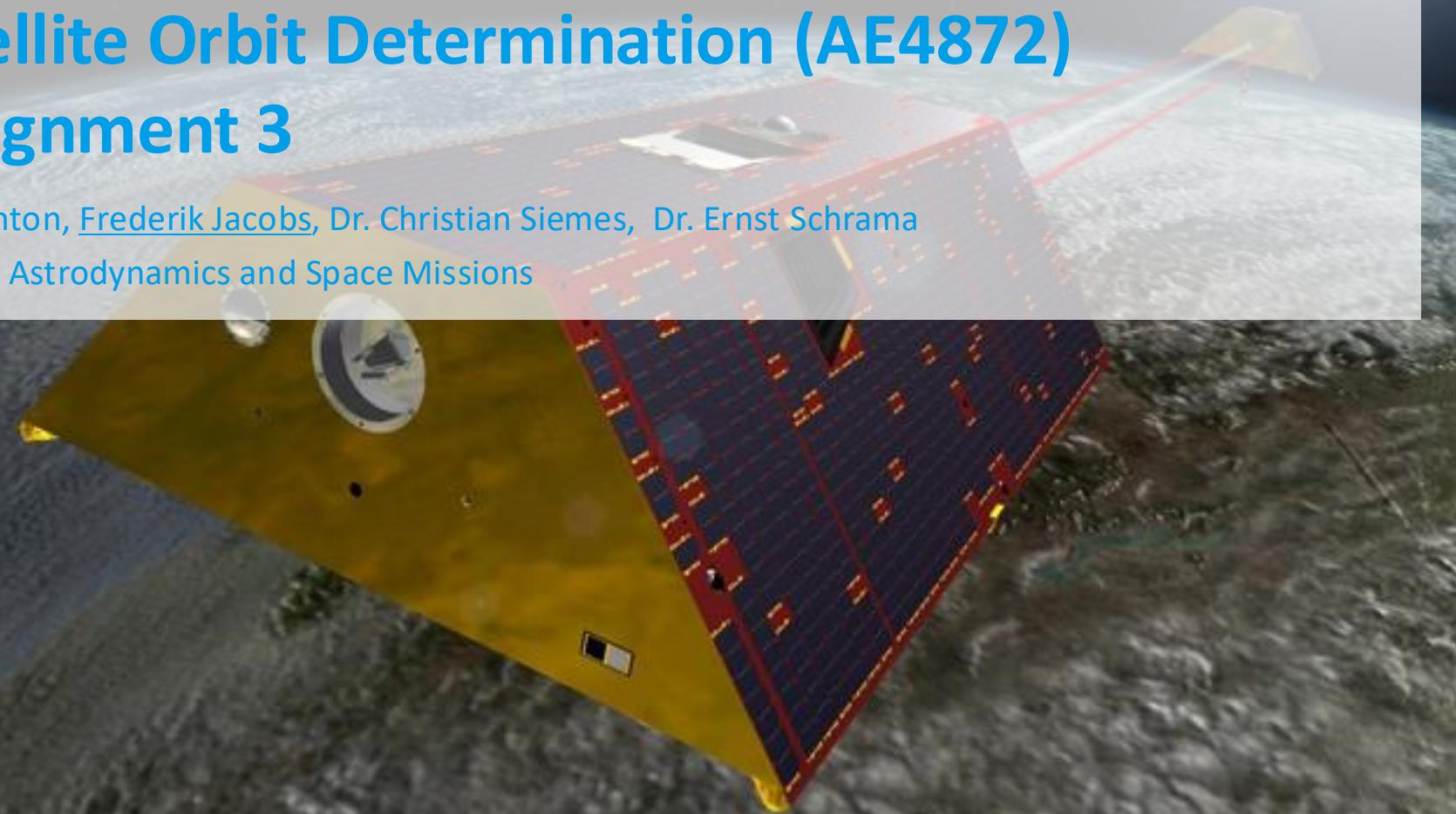


Satellite Orbit Determination (AE4872)

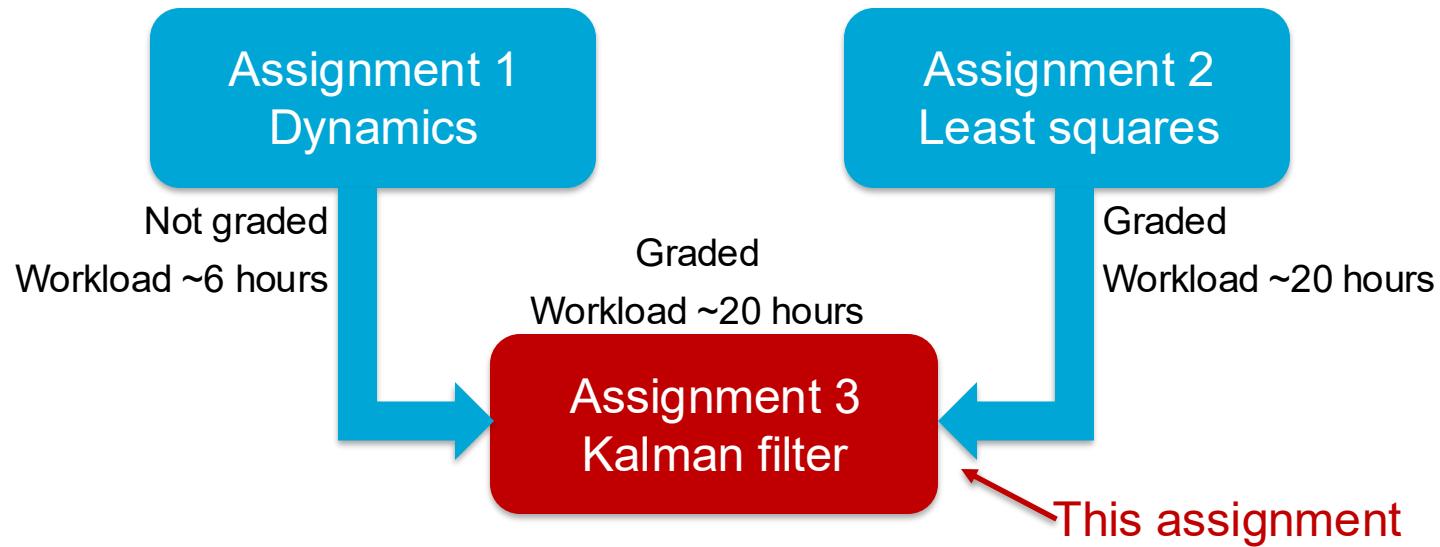
Assignment 3

Sabin Anton, Frederik Jacobs, Dr. Christian Siemes, Dr. Ernst Schrama

Section: Astrodynamics and Space Missions



Assignments recap



- The final grade for the assignments is the average of the grades of the second and third assignments
 - The assignment grade you obtain this year will be kept for the next year, but in the next year, we do not have replacement assignments.
- All assignments are individual

Before we start

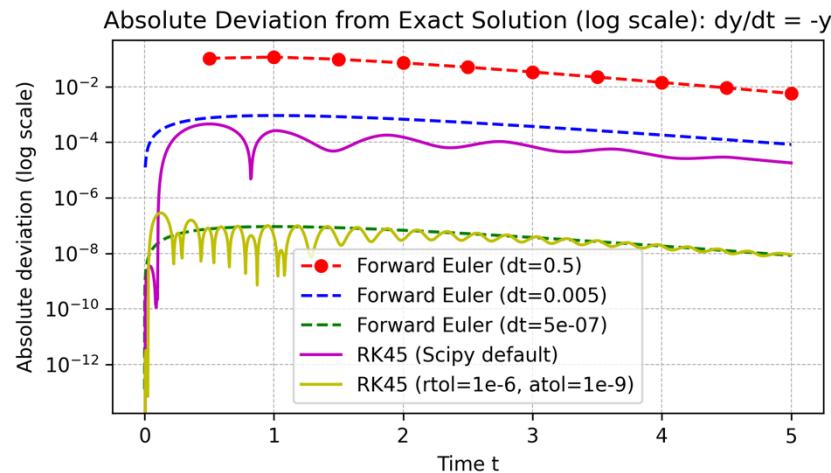
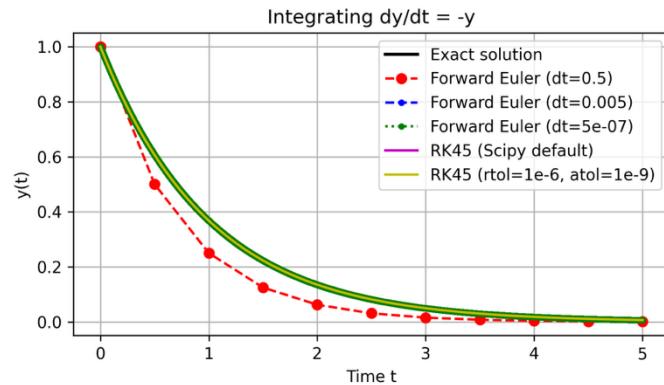
- Feedback on assignment 1 grades?
- Feedback on assignment 2?
- Feedback on ANS platform?
- Reminder: upload your code but we do not read it, we just check it for plagiarism

Feedback assignment 1: integrator

Python: `solve_ivp`

Matlab: `ode45`

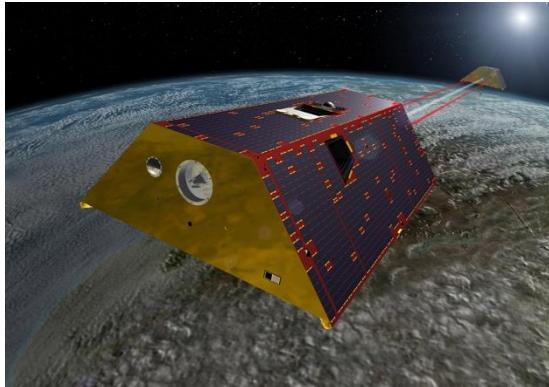
Case	#Function evaluations
FE 0.5	10
FE 0.005	1000
FE 0.0000005	10 000 000
RK45 default	44
RK45 (1e-6,1e-9)	140



Learning objectives

At the end of the instruction session, you should be able to

- Estimate satellite orbits from dynamics and observations



Method is comparable to the
onboard navigation solution

Content

- Combining dynamics and observations
- Satellite dynamics
- Kalman filter
- Assignment 3

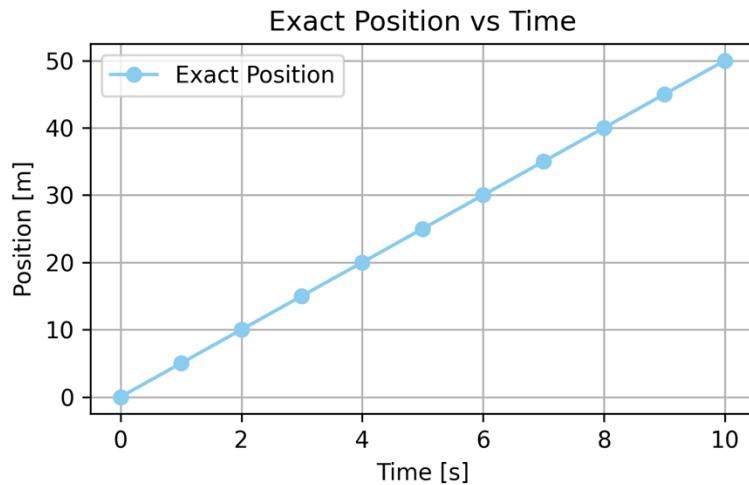
Combining dynamics and observations

Simple example

- A car moves with constant speed on a straight line

$$x = \int_0^t v dt = vt$$

- Every second we measure (with noise) the distance d to the starting point

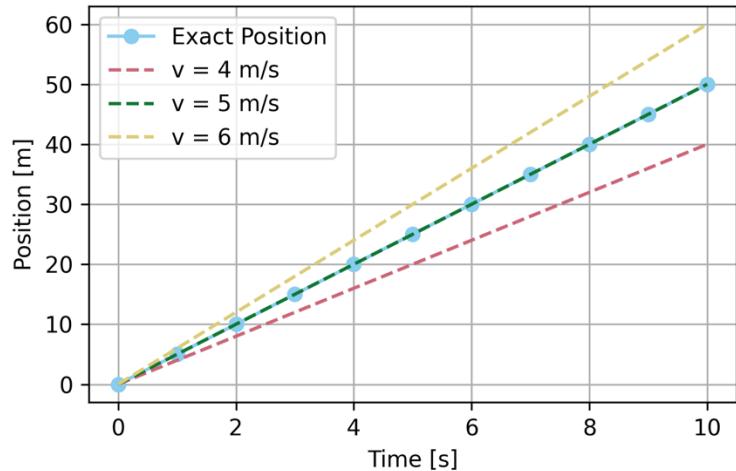


Dynamical model (assignment 1)

- Assume we are aware that it has a constant velocity
- This dynamical model allows us to predict the position

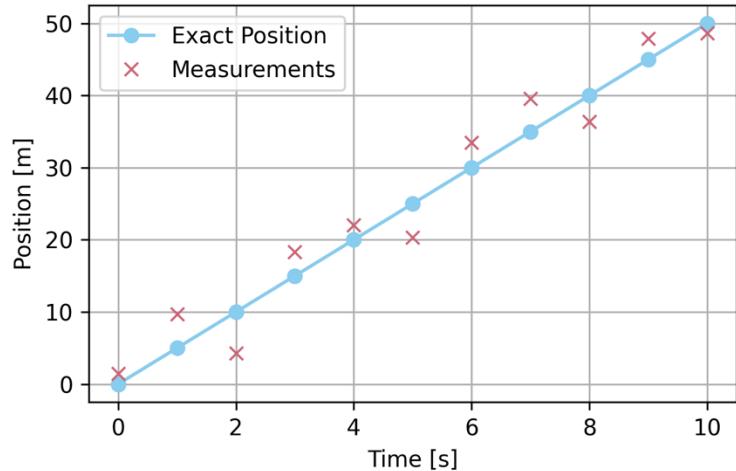
$$x = \int_0^t v dt = vt$$

- But what if our dynamical model has errors?



Least squares (assignment 2)

- We could predict the position using the measurements with LSQ
$$\bar{z} = (d(t_i)) = (x(t_i)) = (1)(x(t_i)) = H\bar{x}$$
- Solving this LSQ problem results in
$$x(t_i) = d(t_i)$$
- But what if we have measurement noise?



Kalman filter (Assignment 3)

- Extend the state vector

$$\bar{y} = \begin{pmatrix} x \\ v \end{pmatrix} \rightarrow \bar{z} = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix}$$

- Step 1: predicting next state with state transition matrix

$$\bar{y}_1 = \begin{pmatrix} x_0 + v\Delta t \\ v \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \bar{y}_0 = \phi_{0,1} \bar{y}_0$$

- Step 2: combining prediction with observation

$$\hat{y}_1 = \bar{y}_1 + K_k (\bar{z}_1 - H \bar{y}_1)$$

Warning: this is only valid for linear problems! For non-linear problems, the state and state transition matrix need to be integrated.

Kalman filter vs. recursive parameter estimation

The Kalman filter can be derived from *recursive parameter estimation*. The derivations are too lengthy to show here. You can find them, for example, in this book:

K. R. Koch: Parameter Estimation and Hypothesis Testing in Linear Models. Second edition, 1999, Springer.

Chapters 1.3.4 on *matrix identities* and 3.2.8 on *recursive parameter estimation* are relevant. The ebook is available in TU Delft's library.

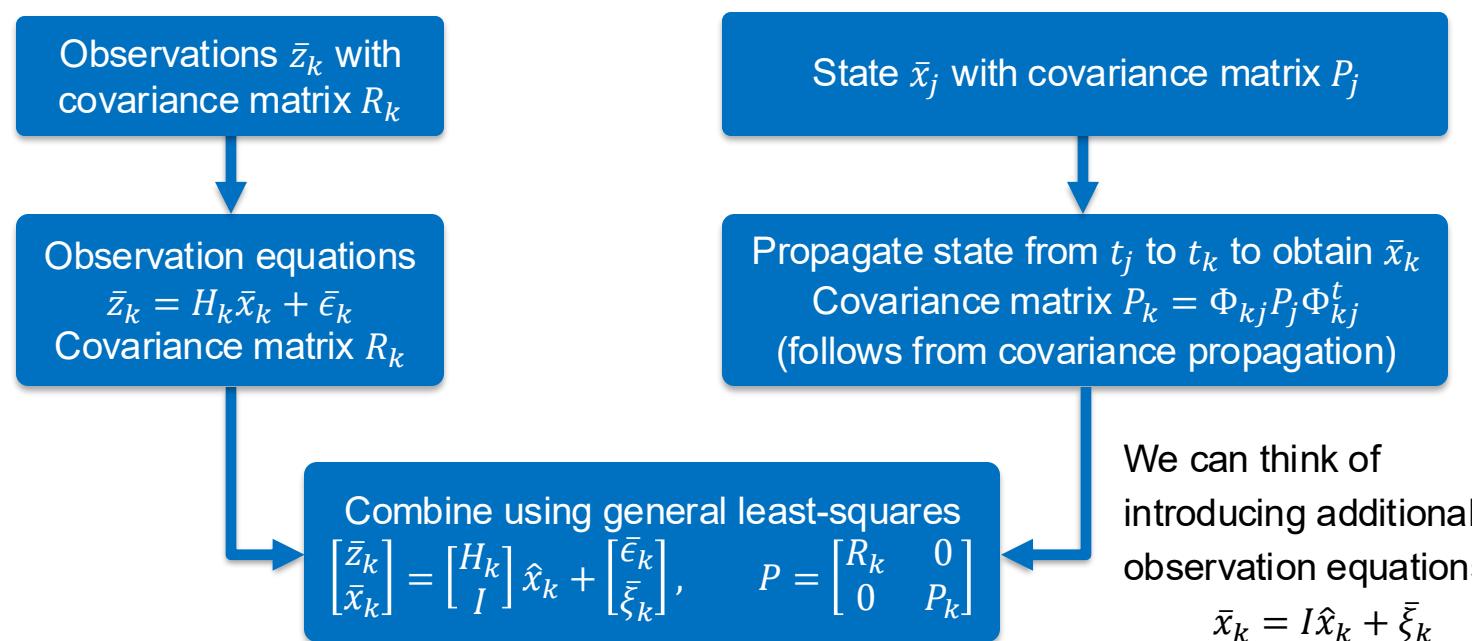
You do not need to study this background information to complete the assignment. We provide it because we were asked a few times how to derive the Kalman gain matrix.

Sequential parameter estimation

We will only estimate position and velocity, so let us simplify the notation:

$$\bar{y} = \bar{x}(t_k) = \bar{x}_k$$

$$\Phi_{kj} = \Phi(t_k, t_j)$$



Kalman filter

Kalman reformulated the problem into a two-step procedure

1. Propagation

$\bar{y}_j \rightarrow \bar{y}_k$... numerical integration

$P_k = \Phi_{kj} P_j \Phi_{kj}^t$... covariance propagation

2. Update

$\bar{z}_k = H_k \bar{y}_k + \bar{\epsilon}_k$... observation equations

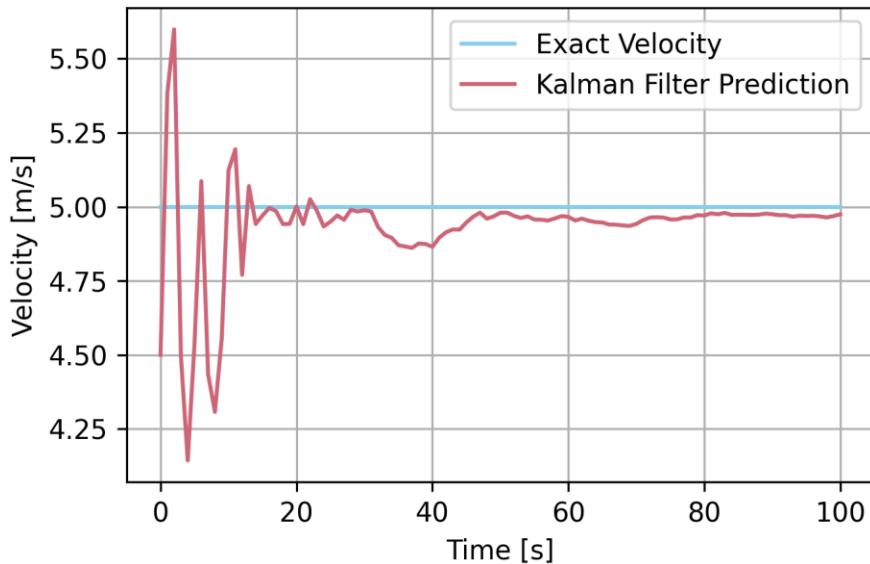
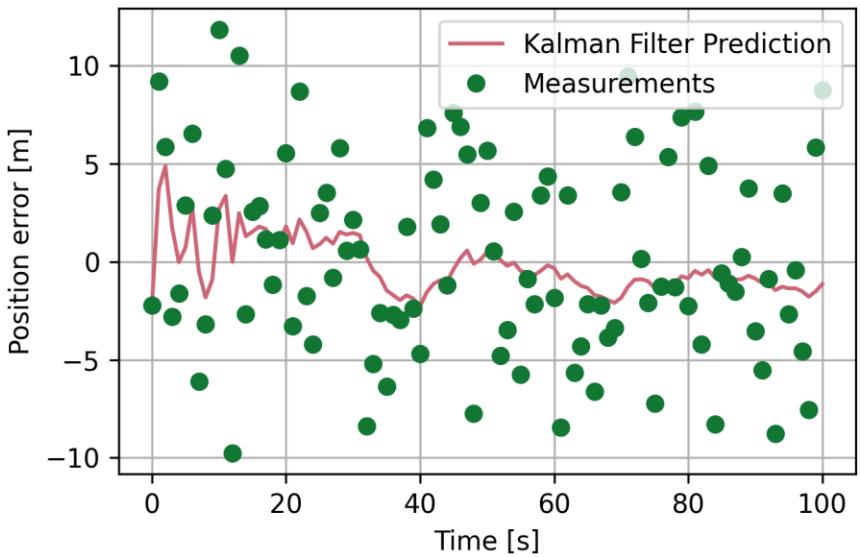
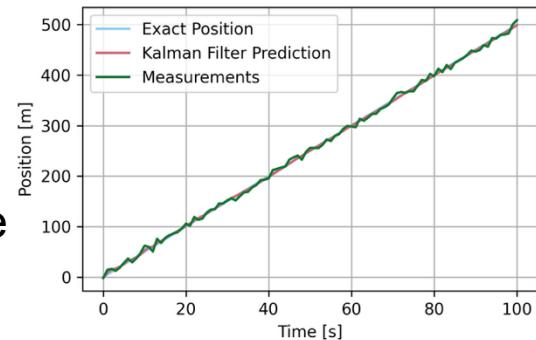
$K_k = P_k H_k^t (H_k P_k H_k^t + R_k)^{-1}$... Kalman gain matrix

$\hat{y}_k = \bar{y}_k + K_k (\bar{z}_k - H_k \bar{x}_k)$... updated state

$\hat{P}_k = (I - K_k H_k) P_k$... covariance matrix of updated state

Result

As the Kalman filter gets more “confident” in the propagated state, it becomes inert to new observations



Kalman filter – Process noise

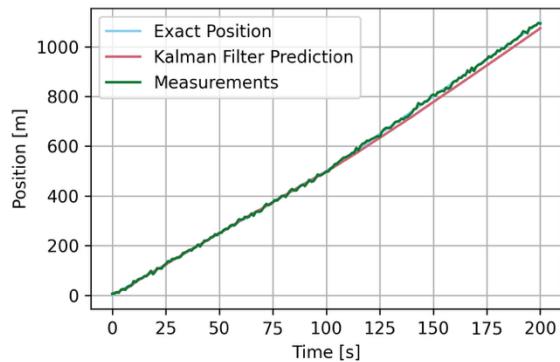
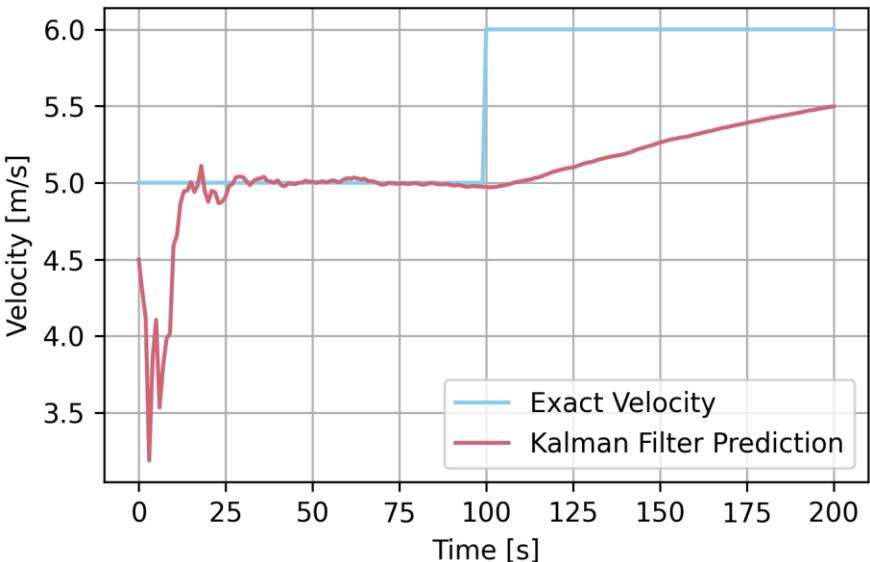
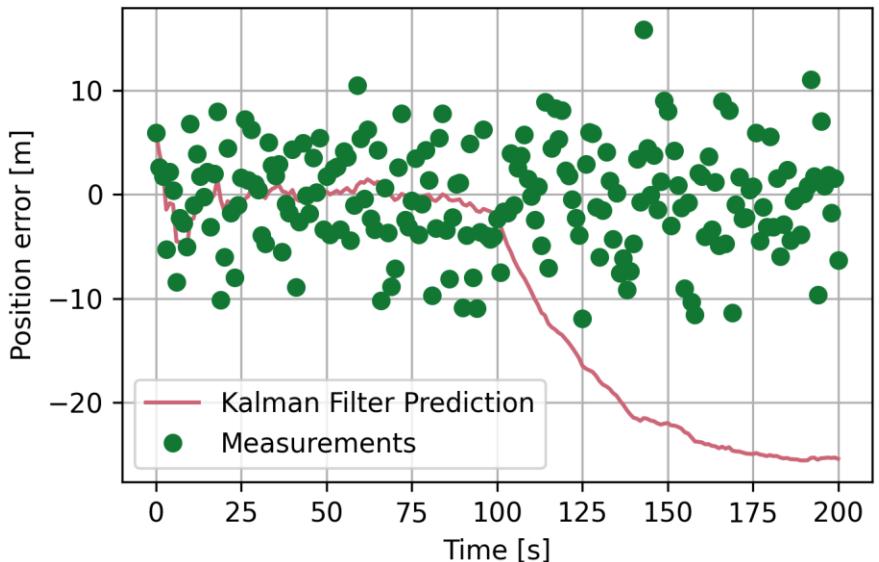
As we include more observations in the estimation at each update step, the covariance matrix \hat{P}_j gets smaller, and so does the covariance matrix $P_k = \Phi_{kj} \hat{P}_j \Phi_{kj}^t$

However, the observation covariance matrix does not get smaller as we move forward (typically)

As the Kalman filter gets more “confident” in the propagated state, it becomes inert to new observations

Changing dynamics

After 100s the driver suddenly increases the speed to 6m/s...



Kalman filter – Process noise

The solution is to tell the Kalman filter that it should not get too confident by adding the process noise covariance matrix:

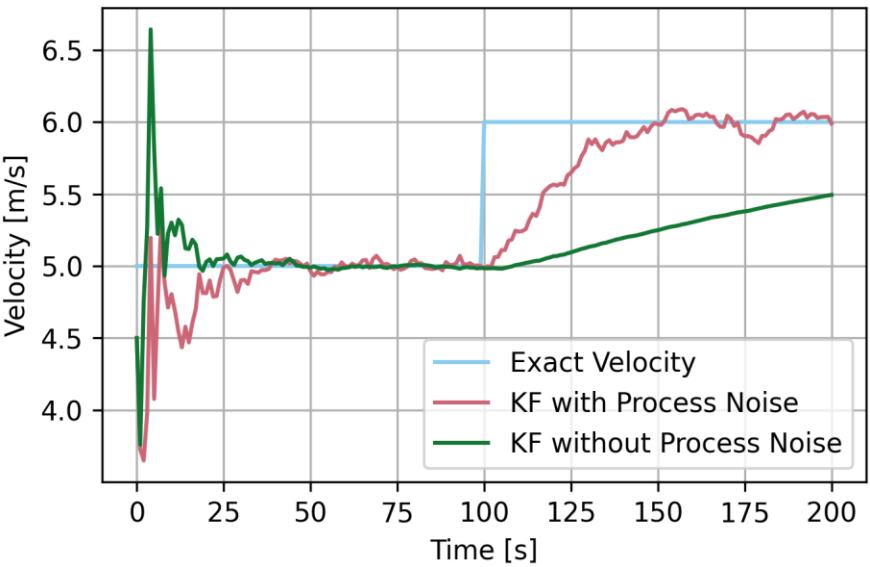
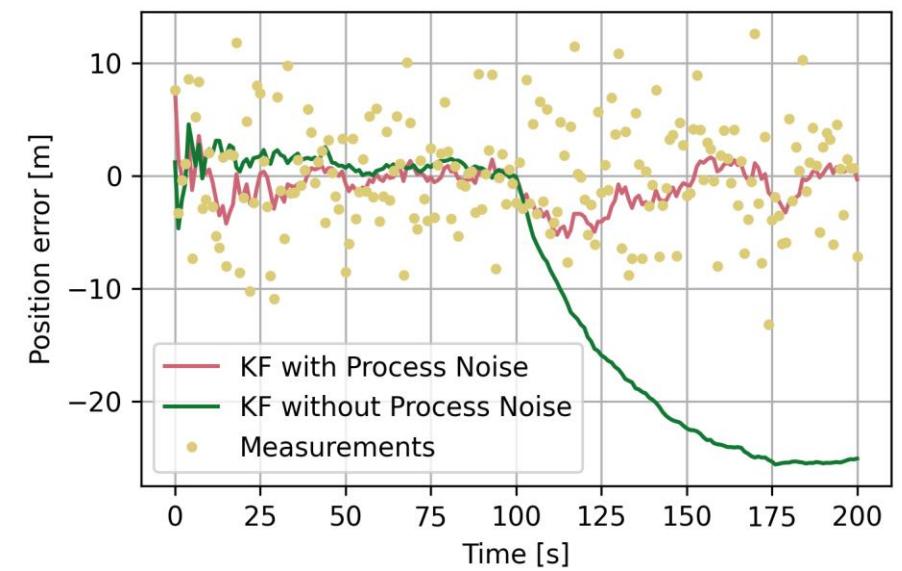
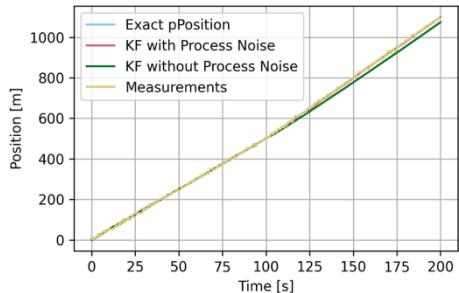
$$P_k = \Phi_{kj} P_j \Phi_{kj}^t + Q_k$$

This prevents that the covariance matrix P_k gets too small and so the Kalman filter remains sensitive to new observations

Pitfall: Do not add random noise to the state!

(Note: The lecture notes write $\Gamma_{kj} Q_k \Gamma_{kj}^t$, but we simplify this to Q_k)

Process noise solution



Satellite dynamics

From point positioning to initial state estimation

We have to rename the observation vector to \bar{z} to avoid confusion with state vector $\bar{y}(t)$

Observation equations (nonlinear)

$$\bar{z} = \bar{h}(\bar{y}) + \bar{\epsilon}$$

$$\bar{y} = \begin{bmatrix} \bar{x}(t) \\ \bar{p} \end{bmatrix}$$

State vector
(position and velocity)

Force parameters
(drag, radiation pressure,...)

From point positioning to initial state estimation

Linearisation of observation equations

$$\bar{z} = \bar{h}(\bar{y}) + \bar{\epsilon}$$

requires calculating the partials

$$\frac{\partial \bar{h}}{\partial \bar{y}} = \begin{bmatrix} \frac{\partial \bar{h}}{\partial \bar{x}(t_k)} & \frac{\partial \bar{h}}{\partial \bar{p}} \end{bmatrix}$$

Observations at epoch t depend on state $\bar{x}(t)$, which in turn depends on the previous state $\bar{x}(t_k)$ via the dynamical model

- Need the relation between $\bar{x}(t)$ and $\bar{x}(t_k)$
- Likewise, need the relation between $\bar{x}(t)$ and \bar{p}

State transition matrix and sensitivity matrix

Use the chain rule

Partial design
matrix H

$$\frac{\partial \bar{h}}{\partial \bar{y}} = \begin{bmatrix} \frac{\partial \bar{h}}{\partial \bar{x}(t_k)} & \frac{\partial \bar{h}}{\partial \bar{p}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \bar{h}}{\partial \bar{x}(t)} \cdot \frac{\partial \bar{x}(t)}{\partial \bar{x}(t_k)} & \frac{\partial \bar{h}}{\partial \bar{x}(t)} \cdot \frac{\partial \bar{x}(t)}{\partial \bar{p}} \end{bmatrix}$$

$$= H \cdot [\Phi(t, t_k) \quad S(t)]$$

State transition
matrix $\Phi(t, t_k)$

6 x 6

Sensitivity
matrix $S(t)$

6 x n_p

How do we calculate $\Phi(t, t_k)$ and $S(t)$?

State transition matrix and sensitivity matrix

Differential equation describing the motion of the satellite

$$\bar{x}(t) = \begin{bmatrix} \bar{r}(t) \\ \bar{v}(t) \end{bmatrix} \rightarrow \frac{d}{dt} \bar{x}(t) = \begin{bmatrix} \bar{v}(t) \\ \bar{a}(t, \bar{x}(t), \bar{p}) \end{bmatrix} = \bar{f}(t, \bar{x}(t), \bar{p})$$

Examples:

Drag (depends
on velocity)

Gravity field coefficients
(depend on position)

State transition matrix and sensitivity matrix

Differential equation describing the motion of the satellite

$$\bar{x}(t) = \begin{bmatrix} \bar{r}(t) \\ \bar{v}(t) \end{bmatrix} \rightarrow \frac{d}{dt} \bar{x}(t) = \begin{bmatrix} \bar{v}(t) \\ \bar{a}(t, \bar{x}(t), \bar{p}) \end{bmatrix} = \bar{f}(t, \bar{x}(t), \bar{p})$$

Differentiate with respect to state $\bar{x}(t_k)$:

$$\boxed{\frac{\partial}{\partial \bar{x}(t_k)} \frac{d}{dt} \bar{x}(t)} = \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t_k)} = \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t)} \cdot \frac{\partial \bar{x}(t)}{\partial \bar{x}(t_k)} = \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t)} \cdot \Phi(t, t_k)$$

Swap the sequence of differential and partial derivative

$$\frac{\partial}{\partial \bar{x}(t_k)} \left(\frac{d}{dt} \bar{x}(t) \right) = \frac{d}{dt} \left(\frac{\partial \bar{x}(t)}{\partial \bar{x}(t_k)} \right) = \frac{d}{dt} \Phi(t, t_k)$$

State transition matrix and sensitivity matrix

Thus, the differential equation for the state transition matrix reads

$$\frac{d}{dt} \Phi(t, t_k) = \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t)} \cdot \Phi(t, t_k)$$

Analogously, differentiate with respect to force parameters

$$\frac{d}{dt} \frac{\partial \bar{x}(t)}{\partial \bar{p}} = \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t)} \cdot \frac{\partial \bar{x}(t)}{\partial \bar{p}} + \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{p}}$$

Direct:
Forces acting
at epoch t

Thus, the differential equation for the sensitivity matrix reads

$$\frac{d}{dt} S(t) = \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t)} \cdot S(t) + \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{p}}$$

Indirect:
State depends
on “history” of
force parameters

State transition matrix and sensitivity matrix

Since $\bar{f}(t, \bar{x}(t), \bar{p}) = \begin{bmatrix} \bar{v}(t) \\ \bar{a}(t, \bar{x}(t), \bar{p}) \end{bmatrix}$ we get for the partials:

$$\frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t)} = \frac{\partial}{\partial \bar{x}(t)} \begin{bmatrix} \bar{v}(t) \\ \bar{a}(t, \bar{x}(t), \bar{p}) \end{bmatrix} = \begin{bmatrix} \frac{\partial \bar{v}(t)}{\partial \bar{r}(t)} & \frac{\partial \bar{v}(t)}{\partial \bar{v}(t)} \\ \frac{\partial \bar{a}(t, \bar{x}(t), \bar{p})}{\partial \bar{r}(t)} & \frac{\partial \bar{a}(t, \bar{x}(t), \bar{p})}{\partial \bar{v}(t)} \end{bmatrix}$$

Identity matrix

Zero matrix $= \begin{bmatrix} 0_{3 \times 3} & I_{3 \times 3} \\ \frac{\partial \bar{a}(t, \bar{x}(t), \bar{p})}{\partial \bar{r}(t)} & \frac{\partial \bar{a}(t, \bar{x}(t), \bar{p})}{\partial \bar{v}(t)} \end{bmatrix}$

6 x 6 matrix

$$\frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{p}} = \frac{\partial}{\partial \bar{p}} \begin{bmatrix} \bar{v}(t) \\ \bar{a}(t, \bar{x}(t), \bar{p}) \end{bmatrix} = \begin{bmatrix} 0_{3 \times n_p} \\ \frac{\partial \bar{a}(t)}{\partial \bar{p}(t)} \end{bmatrix}$$

6 x n_p matrix

State transition matrix and sensitivity matrix

Sorting inside matrices

$$\frac{\partial \bar{a}}{\partial \bar{r}} = \begin{bmatrix} \frac{\partial a_x}{\partial r_x} & \frac{\partial a_x}{\partial r_y} & \frac{\partial a_x}{\partial r_z} \\ \frac{\partial a_y}{\partial r_x} & \frac{\partial a_y}{\partial r_y} & \frac{\partial a_y}{\partial r_z} \\ \frac{\partial a_z}{\partial r_x} & \frac{\partial a_z}{\partial r_y} & \frac{\partial a_z}{\partial r_z} \end{bmatrix} \quad \begin{array}{c} \downarrow \bar{a} \\ \xrightarrow{\bar{r}} \end{array}$$

State transition matrix and sensitivity matrix

Drag equation in vector form:

$$\bar{a} = -\frac{1}{2} \frac{AC_D}{m} \rho |\bar{v}| \bar{v}$$

Task: Calculate $\frac{\partial \bar{a}}{\partial \bar{r}}$ and $\frac{\partial \bar{a}}{\partial \bar{v}}$

Constants:
A ... area
m ... mass
 C_D ... drag coefficient
 ρ ... atmospheric density

Let see how much time you need
(max 10 minutes)

Acceleration in rotating reference frame

Differential equation in a rotating frame (see assignment1)

$$y(t) = \begin{bmatrix} \bar{r}_{rot}^{rot}(t) \\ \bar{v}_{rot}^{rot}(t) \end{bmatrix}, \quad \frac{dy}{dt} = f(y, t) = \begin{bmatrix} \bar{v}_{rot}^{rot}(t) \\ \bar{a}_{rot}^{rot}(t) \end{bmatrix}$$

Position and velocity as observed
in the rotating frame

where

$$\bar{a}_{rot}^{rot}(t) = -\Omega^2 \bar{r}^{rot}(t) - 2 \Omega \bar{v}_{rot}^{rot}(t) + \bar{a}_{in}^{rot}(t)$$

Acceleration as observed in
the rotating frame

Acceleration in the
rotating frame
(e.g., Earth's gravity)

Kalman filtering

Dynamics (assignment 1)

In assignment 1, we used a dynamical model to propagate the initial state $\bar{x}(t_k)$

$$\bar{x}(t_k) = \begin{bmatrix} \bar{r}(t_k) \\ \bar{v}(t_k) \end{bmatrix}$$



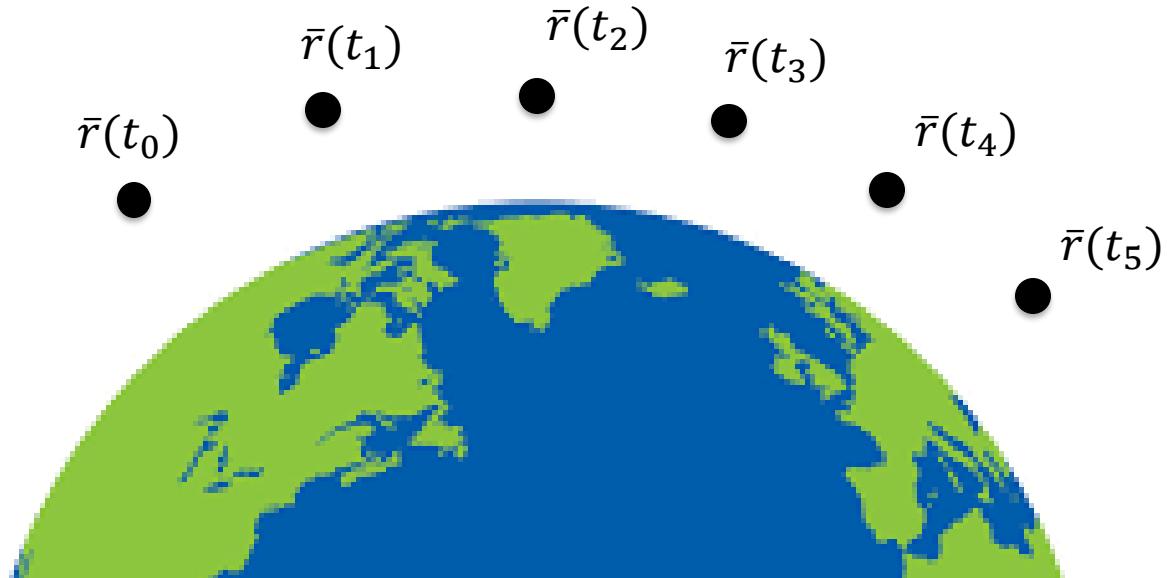
$$\bar{x}(t) = \begin{bmatrix} \bar{r}(t) \\ \bar{v}(t) \end{bmatrix}$$



Point positioning using observations (assignment 2)

In assignment 2 we estimated the positions $\bar{r}(t_n)$ for all epochs t_n

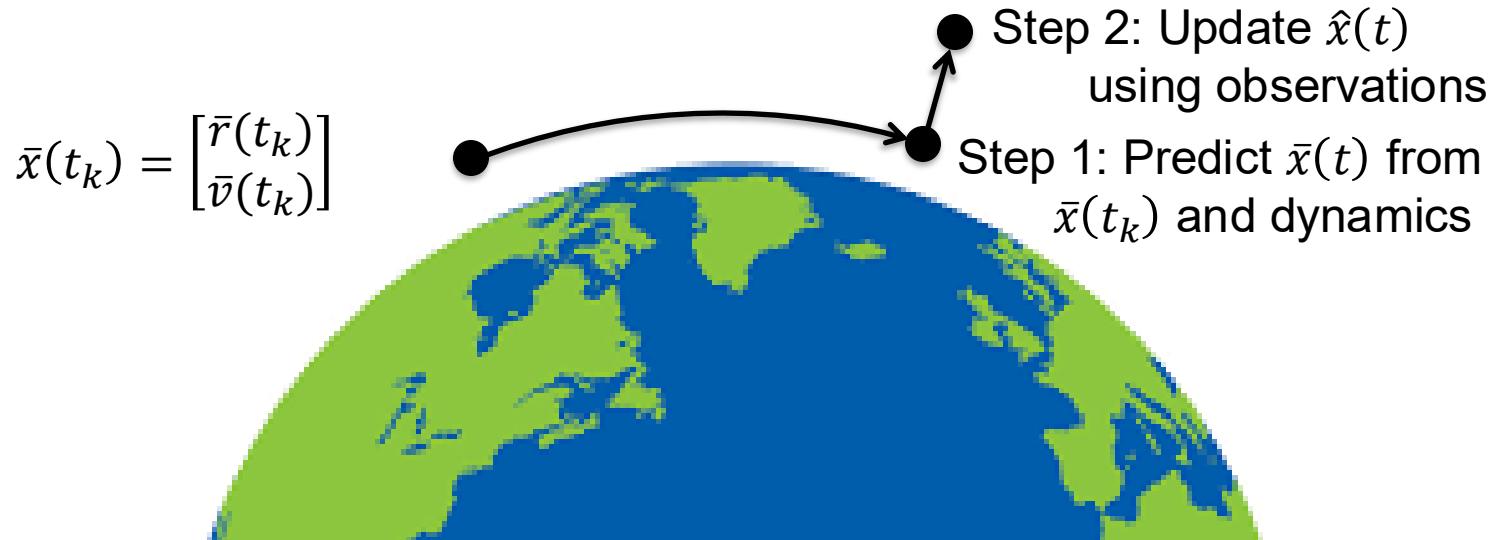
- The estimated positions depend only on the observations



Dynamics and observations (assignment 3)

Estimate the state $\bar{x}(t)$ from the

- observations at epoch t
- state $\bar{x}(t_k)$
- forces acting on the satellite along the orbit (“dynamics”)

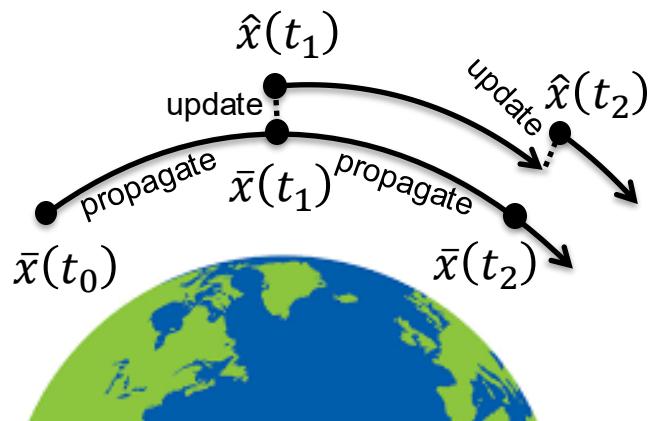


Linearized vs. extended Kalman filter

The observation equations are typically non-linear in satellite orbit determination, so we need to linearize them. Here, we have two options:

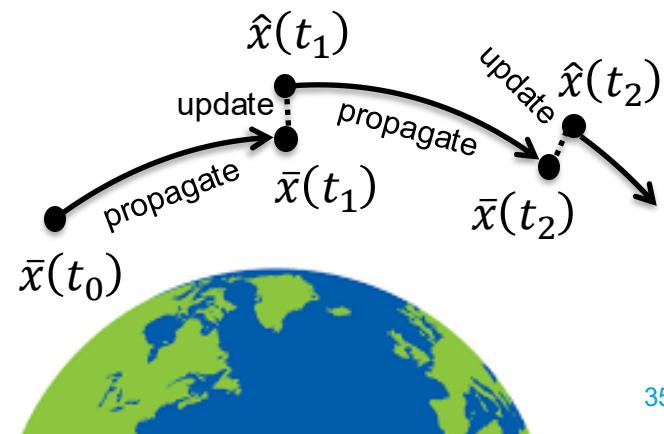
Linearized Kalman filter

Integrate reference orbit $\bar{x}(t)$
→ use $\bar{x}(t)$ as Taylor point for linearization



Extended Kalman filter

Use updated state $\hat{x}(t)$ as Taylor point for linearization



A note on the state transition matrix

Remember how the state transition matrix was defined:

$$\Phi_{kj} = \Phi(t_k, t_j) = \frac{\partial \bar{x}(t_k)}{\partial \bar{x}(t_j)}$$

We may interpret that the state transition matrix relates a *change* in $\bar{x}(t_k)$ to a *change* in $\bar{x}(t_j)$:

$$\Phi_{kj} d\bar{x}(t_j) = d\bar{x}(t_k)$$

Therefore, we can use the state transition matrix for error propagation:

$$P_k = \Phi_{kj} P_j \Phi_{kj}^t$$

However, $\Phi_{kj} \bar{x}(t_j) = \bar{x}(t_k)$ is incorrect for a non-linear problem. We can obtain $\bar{x}(t_k)$ only by numerically integrating from $\bar{x}(t_j)$. (see before)

Linearized vs. extended Kalman filter

State transition matrix

When using the reference orbit for linearization (linear KF):

- Start integration at t_0 with $\Phi(t_0, t_0) = I$ and integrate $\Phi(t, t_0)$ together with state vector $\bar{x}(t)$ to obtain $\Phi(t_k, t_0)$
- Then, $\Phi_{kj} = \Phi(t_k, t_j) = \Phi(t_k, t_0) \Phi(t_j, t_0)^{-1}$

When using the updated state for linearization (extended KF):

- Restart integration at every epoch t_j , i.e., $\Phi(t_j, t_j) = I$, and integrate to epoch t_k to obtain $\Phi_{kj} = \Phi(t_k, t_j)$

How to integrate the matrices and the state vector?

Start at epoch t_k with

$$\Phi(t_k, t_k) = I_{6 \times 6}$$

$$S(t_k) = 0_{6 \times n_p}$$

The initial state is the initial state itself
→ Identity matrix

The initial state does not depend on the forces acting along the orbit
→ Zero matrix

For epochs $t > t_k$ integrate

$$\frac{d}{dt} \Phi(t, t_k) = \frac{\partial \bar{f}(t, \bar{x}(t), \bar{p})}{\partial \bar{x}(t)} \cdot \Phi(t, t_k)$$

together with the state vector $\bar{x}(t)$ because $\Phi(t, t_k)$ depends on $\bar{x}(t)$

(similar for the sensitivity matrix)

How to integrate the matrices and the state vector?

Arrange all differential equations in one vector (any sorting will work):

$$\bar{g} = \begin{bmatrix} \bar{x}(t) \\ \Phi_{11}(t, t_k) \\ \Phi_{21}(t, t_k) \\ \vdots \\ \Phi_{66}(t, t_k) \end{bmatrix}$$

State vector
First column of matrix $\Phi(t, t_k)$
 \vdots
Last column of matrix $\Phi(t, t_k)$

- Plug this into the function for numerical integration
- In the function for calculating the differential, you can sort back and forth
- The function will return a vector, which you can sort back

Matlab:

(Python: numpy.reshape)

`Phi = reshape(Phi,36,1)` ... from 6 x 6 matrix to 36 x 1 vector

`Phi = reshape(Phi,6,6)` ... from 36 x 1 vector to 6 x 6 matrix

Assignment 3: extended Kalman filter

Procedure step-by-step walkthrough

Given: \bar{x}_0 , P_0 , R_0

1. Update with observations \bar{z}_0

- [Apply corrections for light time and relativity effects]
- Calculate design matrix H_0 and $\Delta\bar{z}_0 = \bar{z}_0 - \bar{h}_0(\bar{x}_0)$
- Calculate Kalman gain matrix $K_0 = P_0 H_0^t (H_0 P_0 H_0^t + R_0)^{-1}$
- Update state $\hat{x}_0 = \bar{x}_0 + K_0 \Delta\bar{z}_0$
- Update covariance matrix $\hat{P}_0 = (I - K_0 H_0) P_0$

Note where the transpose is
and that P_0 is not inverted

Assignment 3: extended Kalman filter

Procedure step-by-step walk through

2. Propagate to next epoch t_1

- Initialize $\Phi(t_0, t_0) = I$
- Integrate together from t_0 to t_1
 - \hat{x}_0 to obtain \bar{x}_1
 - $\Phi(t_0, t_0)$ to obtain $\Phi_{1,0} = \Phi(t_1, t_0)$
- Propagate covariance matrix $P_1 = \Phi_{1,0}\hat{P}_0\Phi_{1,0}^t$

Assignment 3: extended Kalman filter

Procedure step-by-step walkthrough

3. Update with observations \bar{z}_1

- [Apply corrections for light time and relativity effects]
- Calculate design matrix H_1 and $\Delta\bar{z}_1 = \bar{z}_1 - \bar{h}_1(\bar{x}_1)$
- Calculate Kalman gain matrix $K_1 = P_1 H_1^t (H_1 P_1 H_1^t + R_1)^{-1}$
- Update state $\hat{x}_1 = \bar{x}_1 + K_1 \Delta\bar{z}_1$
- Update covariance matrix $\hat{P}_1 = (I - K_1 H_1) P_1$

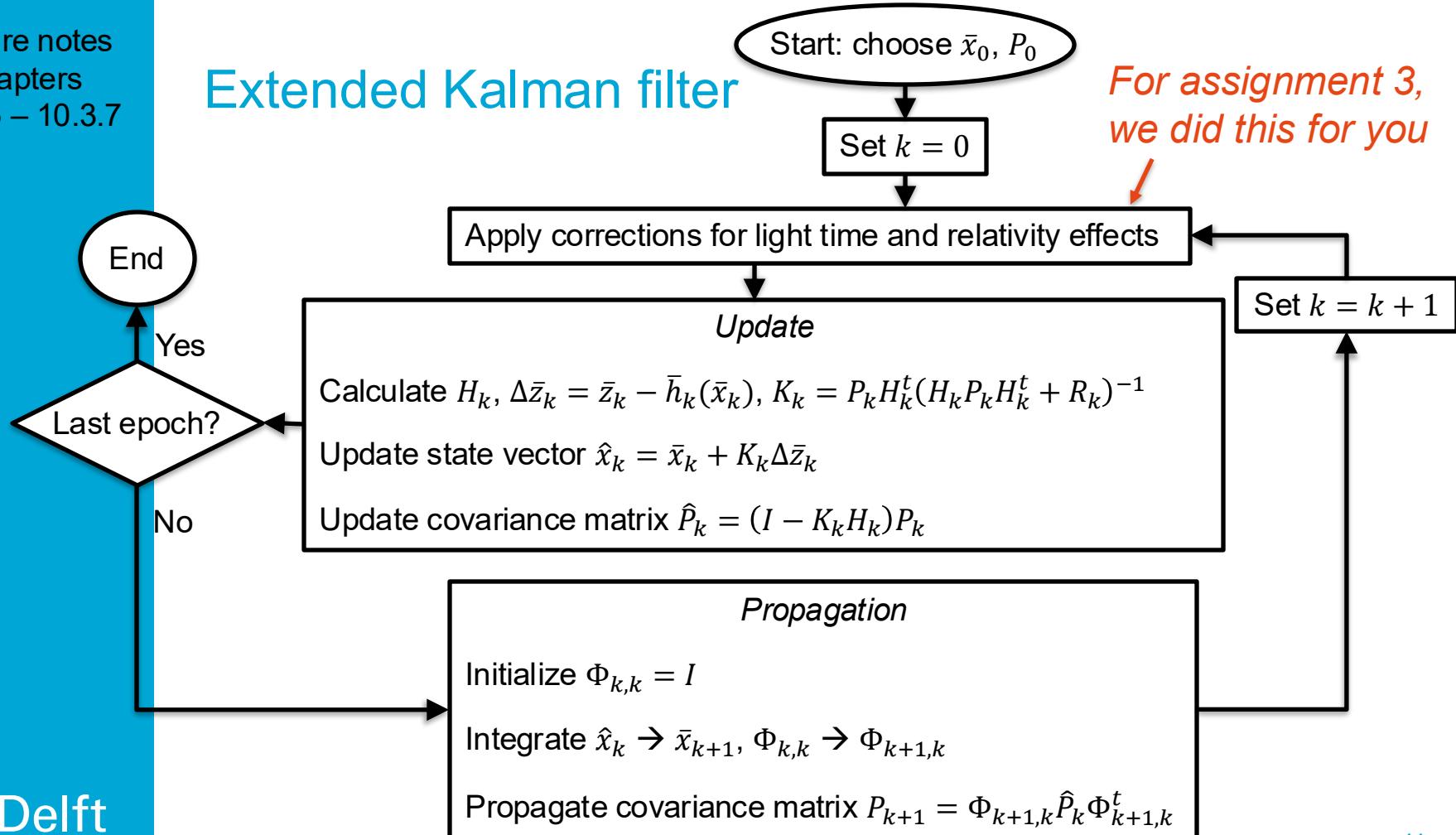
Assignment 3: extended Kalman filter

Procedure step-by-step walkthrough

4. Propagate to next epoch t_2

- Initialize $\Phi(t_1, t_1) = I$
- Integrate together from t_1 to t_2
 - \hat{x}_1 to obtain \bar{x}_2
 - $\Phi(t_1, t_1)$ to obtain $\Phi_{2,1} = \Phi(t_2, t_1)$
- Propagate covariance matrix $P_2 = \Phi_{2,1}\hat{P}_1\Phi_{2,1}^t$

Extended Kalman filter



Kalman filter

In non-linear least-squares, you would perform several iterations until the procedure converges

The linearization is handled differently in the Kalman filter:

- At each epoch, linearization is performed once
- The procedure converges as it moves forward to the next epochs

→ When using a “bad” initial guess to initialize the Kalman filter, you will see the linearization error at the first epoch and how it decreases over the following epochs

Assignment 3

Assignment 3

- a) Construct covariance matrices
- b) Integrate state transition matrix
- c) Implement and run Kalman filter
- d) Analyse position and velocity errors
- e) Linear vs. extended Kalman filter
- f) Effect of process noise
- g) Interpret size of observation residuals
- h) More accurate dynamical model

For task f, you may use $Q_k = \begin{bmatrix} \sigma_a^2 I_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & \sigma_b^2 I_{3 \times 3} \end{bmatrix}$ with suitably chosen σ_a^2 and σ_b^2 , representing the process noise variance in position and velocity, respectively

Light time and relativistic corrections, GPS clock offsets, and GRACE-C receiver clock offsets have already been applied

What?	Variable	Unit	System	Files
Speed of light	c	km/s		299792.458 km/s
Earth rotation rate	ω_E	rad/s		7.292115e-5 rad/s
Epochs	t	s	GPS time	t.txt
Pseudorange observations	$\rho(t)$	km		CA_range.txt
PRN of tracked GPS satellites				PRN_ID.txt
Positions of the GPS satellites (transmitters)	$r_t(t)$	km	ECR frame	rx_gps.txt, ry_gps.txt, rz_gps.txt
Velocities of the GPS satellites (transmitters)	$v_t(t)$	km/s	ECR frame	vx_gps.txt, vy_gps.txt, vz_gps.txt

The following variables are provided only for comparison and debugging

Precise positions of the GRACE-C satellite (receiver)	$r_r(t)$	km	ECR frame	rx.txt, ry.txt, rz.txt
Precise velocities of the GRACE-C satellite (receiver)	$v_r(t)$	km/s	ECR frame	vx.txt, vy.txt, vz.txt