

Assignment 02 - Navigation

© 2022 F. Topputo, P. Di Lizia, A. Morselli, and M. Maestrini, Politecnico di Milano. All rights reserved.

This teaching material was prepared by F. Topputo, P. Di Lizia, A. Morselli, and M. Maestrini.

No part of this material may be reproduced, used, or transmitted in any form by any means without permission.

Spacecraft Guidance and Navigation

AY 2022-2023



POLITECNICO
MILANO 1863

➤ Remarks:

- For those who will not submit the report by the deadlines, you'll be asked to do it before the official exam sessions
- Once you receive the evaluation of the assignments, please wait for the in-person meeting during the exam sessions to discuss about the evaluation


Laboratory sessions

- Laboratory sessions will not be recorded
 - we are here to give you answers while you are working
- When writing in the forum:
 - Reply to the proper thread
 - Check if your question was already posted
 - Be patient (do not re-ask a question that still has to receive an answer)
 - Do **not** attach/send code asking for debugging

Timetable		Room
25 Nov	09.15-12.15	BL.28.21 (+ Maestrini virtual room)
29 Nov	16.15-18.15	BL.27.12 (+ Maestrini virtual room)
30 Nov	14.15-16.15	BL.28.22 (+ Maestrini virtual room)
02 Dec	09.15-12.15	BL.28.21 (+ Maestrini virtual room)
06 Dec	16.15-18.15	BL.27.12 (+ Maestrini virtual room)
TBD	TBD	TDB extra session (Maestrini virtual room)
15 Dec	23.30 CET	Assignment 2 deadline

Delivery of assignments

Assignment will be delivered through **WeBeep**:

DEADLINE  Dec 15, 2022, 23:30:00 CET

- 1) Click on the link to **load Assignment 2** in your Overleaf

https://bit.ly/SGN_22_Assignment2

- 2) **Fill the report** and be sure it is compiled properly
- 3) **Download the PDF** and merge it in a **zipped file** with MATLAB code.

Rename it `lastname123456_Assign2.zip`

- 4) **Submit the compressed file** by uploading it on Webeep

- **Max Size:** 10 Mb

Do **NOT** put spaces
in file names

Assignment 02 – Topics

3 Exercises

- Uncertainty propagation
- Batch filters
- Sequential filters

Suggested MATLAB Version: R2021b (or newer)

1 Impulsive guidance

Exercise 1

Let $\mathbf{x}(t) = \varphi(\mathbf{x}_0, t_0; t)$ be the flow of the geocentric two-body model. 1) Using one of Matlab's built-in integrators, implement and validate [\[1\]](#) a propagator that returns $\mathbf{x}(t)$ for given \mathbf{x}_0 , t_0 , t , and μ . 2) Given the pairs $\{\mathbf{r}_1, \mathbf{r}_2\}$ and $\{t_1, t_2\}$, develop a solver that finds \mathbf{v}_1 such that $\mathbf{r}(t_2) = \mathbf{r}_2$, where $(\mathbf{r}(t), \mathbf{v}(t))^T = \varphi((\mathbf{r}_1, \mathbf{v}_1)^T, t_1; t_2)$ (Lambert's problem). To compute the derivatives of the shooting function, use either a) finite differences or b) the state transition matrix $\Phi = d\varphi/d\mathbf{x}_0$. Validate the algorithms against the classic Lambert solver. 3) Using the propagator of point 1) in the heliocentric case, and reading the motion of the Earth and Mars from SPICE, solve the shooting problem

$$\min_{\mathbf{x}_1, t_1, t_2} \Delta v \quad \text{s.t.} \quad \begin{cases} \mathbf{r}_1 = \mathbf{r}_E(t_1) \\ \mathbf{r}(t_2) = \mathbf{r}_M(t_2) \\ t_1^L \leq t_1 \leq t_1^U \\ t_2^L \leq t_2 \leq t_2^U \\ t_2 \geq t_1 \end{cases} \quad (1)$$

where $\Delta v = \Delta v_1 + \Delta v_2$, $\Delta \mathbf{v}_1 = \mathbf{v}_1 - \mathbf{v}_E(t_1)$, $\Delta \mathbf{v}_2 = \mathbf{v}(t_2) - \mathbf{v}_M(t_2)$. $\mathbf{x}_1 = (\mathbf{r}_1, \mathbf{v}_1)^T$, and $(\mathbf{r}(t), \mathbf{v}(t))^T = \varphi(\mathbf{x}_1, t_1; t_2)$. Define lower and upper bounds, and make sure to solve the problem stated in Eq. [\[1\]](#) for different initial guesses.

Write your answer here

- Develop the exercises in one Matlab script; name the file `lastname123456_Assign1.m`.
- Organize the script in sections, one for each exercise; use local functions if needed.
- Download the [PDF](#) from the Main menu.
- Create a single [.zip file](#) containing both the report in PDF and the [MATLAB file](#). The name shall be `lastname123456_Assign1.zip`.
- Red text indicates where answers are needed; be sure there is no red stuff in your report.
- In your answers, be concise: to the point.
- Deadline for the submission: Nov 11 2021, 23:30.
- Load the compressed file to the Assignments folder on Webeep.

Exercise

NB: The code is not your report!

Answer the question **in the report** and add any plot you think is relevant for your answers there.

Any description or result missing in the report but present in the code **will not contribute** to the evaluation!

Answer here

Script

```
LastName123456_Assign2.m
1 % Spacecraft Guidance and Navigation (2022/2023)
2 % Assignment #2
3 % Exercise #1
4 % Author: Name SURNAME
5
6
7 clearvars; close all; clc; cspice_kclear
8
9 %% Ex 1.1
10 % Write your code here
11
12
13 %% Ex 1.2
14 % Write your code here
15
16
17 %% Ex 1.3
18 % Write your code here
19
20
21 %% Functions
22 % Define your functions at the end of the script
23
24 function [y1, y2] = MyFunction(x1, x2)
25
26 % Function code ...
27
28 y1 = 2*x1;
29 y2 = 3*x2 + x1;
30
31 end
```

Name of
the script
(one per exercise)

One section
per exercise point

Functions at the
end of the script

Exercise 1 – Uncertainty propagation

Exercise 1: Uncertainty propagation

After its launch on November 11, 2022, the upper stage of the Ariane 5 launcher (ID: 87654) is cruising along its highly-elliptical transfer orbit before releasing its embarked payload, the in-orbit servicer unit Orbital Repair Satellite (ORS).

You have been provided with an estimate of the Ariane 5 upper stage state at the pericenter epoch $t_0 = 2022-11-11T19:08:49.824$ (UTC) in terms of its mean and covariance, as reported in Table 1. Assume Keplerian motion can be used to model the spacecraft dynamics.

1. Propagate the initial mean and covariance to all apocenter and pericenter epochs included in the subsequent four revolutions of the upper stage around the Earth using both a linearized approach (LinCov) and the unscented transform (UT). Compare the results in terms of both propagated mean and covariance. Elaborate on the results and the differences between the two approaches.
2. Perform the same uncertainty propagation process to the same epochs using a Monte Carlo (MC) simulation (using at least 100 samples drawn from the initial covariance). Compute the sample mean and sample covariance, and compare them with the results obtained at the previous point. Plot the propagated samples of the MC simulation, and the mean and covariance obtained with all methods, on the equatorial plane in the ECI reference frame. Compare the results and discuss on the validity of the linear and Gaussian assumption for uncertainty propagation.

Table 1: Estimate of the ORS state at t_0 provided in ECI J2000.

Parameter	Value
Ref. epoch t_0 [UTC]	2022-11-11T19:08:49.824
Mean state \hat{x}_0 [km, km/s]	$\hat{r}_0 = [6054.30795817484, -3072.03883303992, -133.115352431876]$ $\hat{v}_0 = [4.64750094824087, 9.18608475681236, -0.62056520749034]$
Covariance P_0 [km ² , km ² /s, km ² /s ²]	$\begin{bmatrix} +5.6e-3 & +3.5e-3 & -7.1e-4 & 0 & 0 & 0 \\ +3.5e-3 & +9.7e-3 & +7.6e-4 & 0 & 0 & 0 \\ -7.1e-4 & +7.6e-4 & +8.1e-4 & 0 & 0 & 0 \\ 0 & 0 & 0 & +2.8e-7 & 0 & 0 \\ 0 & 0 & 0 & 0 & +2.7e-7 & 0 \\ 0 & 0 & 0 & 0 & 0 & +9.6e-8 \end{bmatrix}$

Exercise 2 – Batch filters

Exercise 2: Batch filters

You have been asked to track the Ariane 5 upper stage to improve the accuracy of its state estimate. To this aim, you are allowed to task the observations of the two ground stations reported in Table 2.

- 1. *Compute visibility windows.* By using the mean state reported in Table 1 and by assuming Keplerian motion, predict the upper stage trajectory over a uniform time grid (one point per minute) and compute all the visibility time windows from the available stations in the time interval from $t_0 = 2022-11-12T04:30:00.000$ (UTC) to $t_f = 2022-11-14T16:30:00.000$ (UTC). Plot the resulting predicted Azimuth and Elevation profiles in the visibility windows.
- 2. *Simulate measurements.* The latest available Two-Line Elements (TLE) set of the upper stage are reported in Table 3 (and in WeBeep as 87654.tle). Use SGP4 and the provided TLE to simulate the measurements acquired by the sensor network in Table 2 by:
 - (a) Computing the spacecraft position over the visibility windows identified in Point 1 and deriving the associated expected measurements.
 - (b) Simulating the measurements by adding a random error to the expected measurements (assume a Gaussian model to generate the random error, with noise provided in Table 2).

Table 3: Latest available TLE of the Ariane 5 upper stage.

1_87654U_22110B_22316.00967942_0.00000002_00000-0_32024-3_0_9990
2_87654_3.6309_137.1541_8138191_196.1632_96.6141_1.26411866_834

! Do not copy-paste from PDF

Table 2: Sensor network to track the Ariane 5 upper stage: list of stations, including their features.

Station name	KOUROU	PERTH
Coordinates	LAT = 5.25144° LON = -52.80466° ALT = -14.67 m	LAT = -31.80252° LON = 115.88516° ALT = 22.16 m
Type	Radar (monostatic)	Radar (monostatic)
Provided measurements	Az, El [deg] Range (one-way) [km]	Az, El [deg] Range (one-way) [km]
Measurements noise (diagonal noise matrix R)	$\sigma_{Az,El} = 100$ mdeg $\sigma_{range} = 0.01$ km	$\sigma_{Az,El} = 100$ mdeg $\sigma_{range} = 0.01$ km
Minimum elevation	10 deg	5 deg

Exercise 2 – Batch filters

3. *Solve the navigation problem.* Using the measurements simulated at the previous point:
- (a) Find the least squares (minimum variance) solution to the navigation problem without a priori information using
 - the epoch t_0 as reference epoch;
 - the reference state as the state derived from the TLE set in Table 3 at the reference epoch;
 - the simulated measurements obtained for the PERTH ground station only;
 - pure Keplerian motion to model the spacecraft dynamics.
 - (b) Repeat step 3a by using all simulated measurements from both ground stations.
 - (c) Repeat step 3b by using J2-perturbed motion to model the spacecraft dynamics.

Exercise 3 – Sequential filters

Exercise 3: Sequential filters

After the release from the Ariane 5 upper stage, the OSR unit unfurls its solar panels and uses its low-thrust plasma engines to reach its target spacecraft SGN-I. SGN-I is moving on a circular geostationary (GEO) orbit. Once in the GEO regime, the OSR unit approaches the target SGN-I to start its in-orbit servicing mission. The target can be modeled as a parallelepiped with properties reported in Table 4.

SGN-I has lost its capability of stabilizing its attitude and therefore it is tumbling according to Euler's equation of rigid body motion:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \boldsymbol{\omega} \wedge \mathbf{J}\boldsymbol{\omega} \quad (1)$$

Where $\boldsymbol{\omega}$ represents the target's angular velocity with respect to the inertial frame, expressed in the target's body frame. In this equation \mathbf{J} represents the matrix of inertia of the target. To complete the description of the dynamics, the target's attitude with respect to the inertial frame can be expressed via quaternions, which evolve according to the following differential equation:

$$\dot{\mathbf{q}} = \begin{bmatrix} 0 & -\omega(1) & -\omega(2) & -\omega(3) \\ \omega(1) & 0 & \omega(3) & -\omega(2) \\ \omega(2) & -\omega(3) & 0 & \omega(1) \\ \omega(3) & \omega(2) & \omega(1) & 0 \end{bmatrix} \mathbf{q} \quad (2)$$

The quaternion \mathbf{q} can be used to extract the director cosine matrix necessary to express in the target body frame a point whose coordinates are given in the inertial frame (i.e., through matlab's *quat2dcm*):

$$\mathbf{C}_{T,I} = \begin{bmatrix} \mathbf{q}(1)^2 + \mathbf{q}(2)^2 - \mathbf{q}(3)^2 - \mathbf{q}(4)^2 & 2(\mathbf{q}(2)\mathbf{q}(3) + \mathbf{q}(1)\mathbf{q}(4)) & 2(\mathbf{q}(2)\mathbf{q}(4) - \mathbf{q}(1)\mathbf{q}(3)) \\ 2(\mathbf{q}(2)\mathbf{q}(3) - \mathbf{q}(1)\mathbf{q}(4)) & \mathbf{q}(1)^2 - \mathbf{q}(2)^2 + \mathbf{q}(3)^2 - \mathbf{q}(4)^2 & 2(\mathbf{q}(3)\mathbf{q}(4) + \mathbf{q}(1)\mathbf{q}(2)) \\ 2(\mathbf{q}(2)\mathbf{q}(4) + \mathbf{q}(1)\mathbf{q}(3)) & 2(\mathbf{q}(3)\mathbf{q}(4) - \mathbf{q}(1)\mathbf{q}(2)) & \mathbf{q}(1)^2 - \mathbf{q}(2)^2 - \mathbf{q}(3)^2 + \mathbf{q}(4)^2 \end{bmatrix} \quad (3)$$

It is assumed that SGN-I can correctly estimate its attitude and provide it to the chaser via a direct link with the OSR. The initial value of the states for the target's attitude are reported in Table 5.

Therefore, the only state that needs to be estimated to retrieve the relative pose is the relative position and velocity expressed in the target's LVLH frame.

Table 4: Parameters of SGN-I.

Parameter	Value
Size [m]	$[l, h, d] = [10, 5, 3]$
Density [kg/m ³]	$\rho = 1420$
Mass [kg]	$m = lhd\rho = 213000$
Matrix of Inertia [kg/m ²]	$\mathbf{J} = \frac{m}{12} \text{diag}([d^2 + h^2, l^2 + h^2, l^2 + d^2])$
Vertices [m]	$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \end{bmatrix} = \begin{bmatrix} l/2 & -d/2 & -h/2 \\ l/2 & d/2 & -h/2 \\ l/2 & d/2 & h/2 \\ l/2 & -d/2 & h/2 \\ -l/2 & -d/2 & -h/2 \\ -l/2 & d/2 & -h/2 \\ -l/2 & d/2 & h/2 \\ -l/2 & -d/2 & h/2 \end{bmatrix}$

Table 5: Initial quaternion and angular velocity of the target body frame with respect to the inertial one.

Parameter	Value
Ref. epoch t_0 [UTC]	2023-04-01T14:55:12.023
Initial quaternion [-]	$\mathbf{q}(t_0) = \begin{bmatrix} 0.674156352338764 \\ 0.223585877389611 \\ 0.465489474399161 \\ 0.528055032413102 \end{bmatrix}$
Initial Angular velocity [rad/s]	$\boldsymbol{\omega}(t_0) = \begin{bmatrix} -0.001262427155865 \\ 0.001204540074343 \\ -0.000039180139156 \end{bmatrix}$

Exercise 3 – Sequential filters

The motion is assumed to be correctly described by the linear, Clohessy-Wiltshire (CW) equations⁴

$$\begin{aligned}\ddot{x} &= 3n^2x + 2n\dot{y} \\ \ddot{y} &= -2n\dot{x} \\ \ddot{z} &= -n^2z\end{aligned}\quad (4)$$

where x , y , and z are the relative position components expressed in the LVLH frame, whereas n is the mean motion of the target along its GEO trajectory. Remember that the LVLH frame

You are asked to develop a sequential filter to narrow down the uncertainty on the knowledge of the OSR relative state vector before executing the rendezvous procedure. To this aim, you can exploit the measurements obtained by a stereo camera onboard the OSR, whose features are reported in Table 7. It is assumed that the camera is pointed towards the V-axis of the LVLH frame during the entire navigation window, hence the needed Director Cosine Matrix that allows to rotate a vector from LVLH frame to camera frame can be expressed as:

$$\mathbf{C}_{\text{cam},L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}\quad (6)$$

The overall geometry of the observation is described in Figure 1.

The camera provides measurements of the vertices of the RSO in terms of horizontal pixel, vertical pixel, and disparity according to the model:

$$\mathbf{y} = \left[u_0 - Dfoc \frac{Y}{Z}, v_0 + Dfoc \frac{X}{Z}, \frac{bDfoc}{Z} \right]\quad (7)$$

where $[X, Y, Z]$ represent the coordinates of any vertex of the RSO expressed in a reference frame fixed with the camera and centered in the chaser. In the framework of this exercise, you are provided with a tool (i.e., *meas_sim*) to simulate the measurements acquired by the stereo camera during the entire navigation window. This tool will provide a set of stereo measurement for each *visible* vertex of the RSO and the ID of the corresponding vertex as indicated in Table 4. Visibility is automatically computed by taking into account both illumination conditions and relative position between target and chaser. For some configurations it may be impossible to view any vertex⁵.

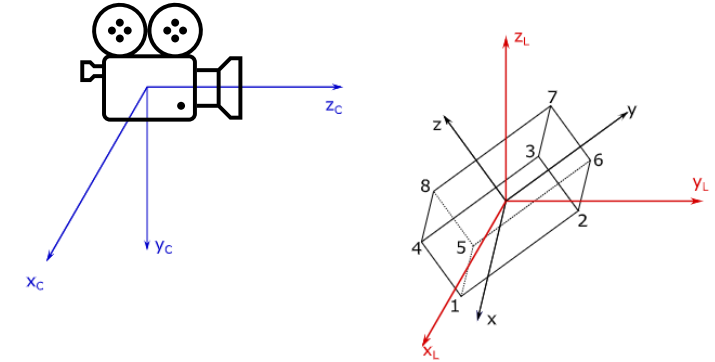


Figure 1: Pay attention to the three different reference frames, in blue the camera frame (C pedices), in red the LVLH frame (L pedices), and in black the target body frame.

Table 6: Estimate of the initial relative state state at t_0 in the target relative LVLH frame.

Parameter	Value
Ref. epoch t_0 [UTC]	2023-04-01T14:55:12.023
Mean state $\hat{\mathbf{x}}_0$ [m, m/s]	$\hat{\mathbf{r}}_0 = [15.792658268071492 \ -59.044939772661586 \ 3.227106250277039]$ $\hat{\mathbf{v}}_0 = [-0.053960274403210 \ -0.053969644762889 \ -0.089140748762173]$
Covariance P_0 [m ² , m ² /s, m ² /s ²]	diag([10, 10, 10, 0.1, 0.1, 0.1])

Exercise 3 – Sequential filters

1. Use the function `meas_sim` to simulate the measurements acquired by the stereo camera during a navigation window of 1 day when following the **nominal trajectory** (reference for error estimation) obtained by setting $\mathbf{r}_0 = [12.0, -60.0, 0.0]$ and $\mathbf{v}_0 = [1e - 4, -2n\mathbf{r}_0(1), -1.2e - 3]$. Compute the number of visible corners at each measurement instant and visualize the relative nominal trajectory. Assume that the camera is providing measurements with a frequency of 1Hz.

The provided function `meas_sim` requires at each time instant t : the mean motion n , the relative state \mathbf{r} , the quaternion of the target \mathbf{q} , the time t passed since the initial epoch, and a structure `Camera` containing the camera parameters indicated with the following nomenclature: `Camera.f` for the focal length, `Camera.d` for pixel density, `Camera.p0` for the central pixel coordinates, `Camera.Cframe` for the director cosine matrix responsible for the rotation from LVLH to camera frame, and finally the parameter `Camera.R` to indicate the variance of the measurements.

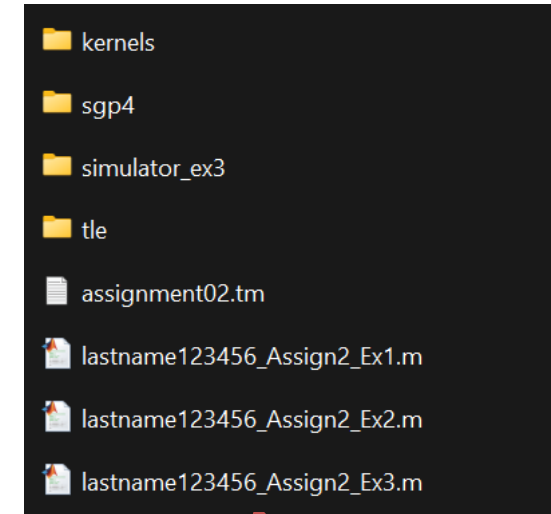
2. Verify that the visible features of the target SGN-I lie inside the field-of-view of the camera during the entire navigation window.
3. Using both an extended Kalman filter (EKF) and an unscented Kalman filter (UKF) update sequentially the spacecraft state (in terms of mean and covariance) by processing the acquired measurements in chronological order.
4. Compute the error along the navigation window between the estimated mean states and the true trajectory. Check the consistency between the covariance matrices estimated by the filter during the navigation window and the computed error. Elaborate on the comparison of the results obtained with the two filters.

Table 7: Parameters of the stereo camera.

Parameter	Value
Focal length [mm]	$foc = 3$
Pixel Density [pix/mm]	$D = 54$
Baseline [m]	$b = 1$
Center Pixel [pix]	$[u_0, v_0] = [960, 600]$
Sensor Size [pix]	$[1920, 1200]$
Measurement Noise [pix ²]	$\sigma_u^2 = \sigma_v^2 = \sigma_d^2 = 10$

Supporting material

- Available on [WeBeep](#), inside the folder Laboratories
- Compressed folder **Assignment02.zip** contains:
 - Subfolder **kernels** with all required SPICE kernels
 - Subfolder **sgp4** with all sgp4 functions (Lab03)
 - Subfolder **tle** with two-line elements
 - Meta-kernel **assignment02.tm**
 - Subfolder **simulator_ex3** with *meas_sim* function



How to prepare your script

- Unzip the folder and work on your Matlab script inside it
- Remember to **load the meta-kernel** and to **add the path to sgp4 subfolder** in your script
 - Suggestion: use relative paths
- **NB:** do not to upload the supporting material when preparing your delivery on WeBeep, we have it.

lastname123456_Assign1_Ex1.m
lastname123456_Assign1_Ex2.m
lastname123456_Assign1_Ex3.m

A quick recap (1) – Orbit propagator with J2

NB: The J2 perturbation is **positional** and depends on the satellite position in **ECEF** frame.

1. Compute the matrix to convert position from ECI to ECEF

```
rotm = cspice_pxform('J2000', 'ITRF93', et);
```

2. Convert the ECI position in ECEF

```
pos_ECEF = rotm * pos_ECI;
```

3. Compute the J2 acceleration ($J_2 = 0.0010826269$) in ECEF:

$$\mathbf{a}_{J_2} = \frac{3}{2} \mu J_2 \frac{\mathbf{r}}{r^3} \left(\frac{R_E}{r} \right)^2 \left(5 \left(\frac{z}{r} \right)^2 - \begin{Bmatrix} 1 \\ 1 \\ 3 \end{Bmatrix} \right)$$

4. Convert it to ECI
5. Add it to the Keplerian acceleration

A quick recap (2) – SGP4/SDP4

Usage of SGP4

- a. Initialize the satellite record with
 - `twoline2rv` (requires tle strings)
 - `read_TLE` (requires TLE file name or sat id)
- b. Call SGP4
 - Provide time since reference epoch (minutes)
- c. Convert from/to TEME (if needed)
 - Compute `ttt`
 - Centuries from TDT 2000 January 1 00:00:00
 - TEME acceleration can be set to [0,0,0]
 - Get offsets `dPsi` and `dEpsilon`
 - Convert them in rad if provided in arcsec
 - Call functions `teme2eci` or `eci2teme`

```
typerun    = 'u'; % user-provided inputs to SGP4 Matlab function
opsmode    = 'a'; % afspc approach ('air force space command')
whichconst = 72; % WGS72 constants (radius, gravitational parameter)

% initialize the satrec structure, using the function twoline2rv
satrec = twoline2rv(longstr1, longstr2, typerun, 'e', opsmode, whichconst)

satrec = read_TLE(123456, whichconst);

% Evaluate the TLE
et0 = cspice_str2et('TLE REF EPOCH'); % Reference time of the TLE
et = 12345.6789; % Ephemeris time for the SGP4 propagation
tsince = (et-et0)/60; % Time since TLE reference epoch [min]
[~, rteme, vteme] = sgp4(satrec, tsince);

% Centuries from TDT 2000 January 1 00:00:00.000
ttt = cspice_unitim(et, 'ET', 'TDT')/cspice_jyear()/100;

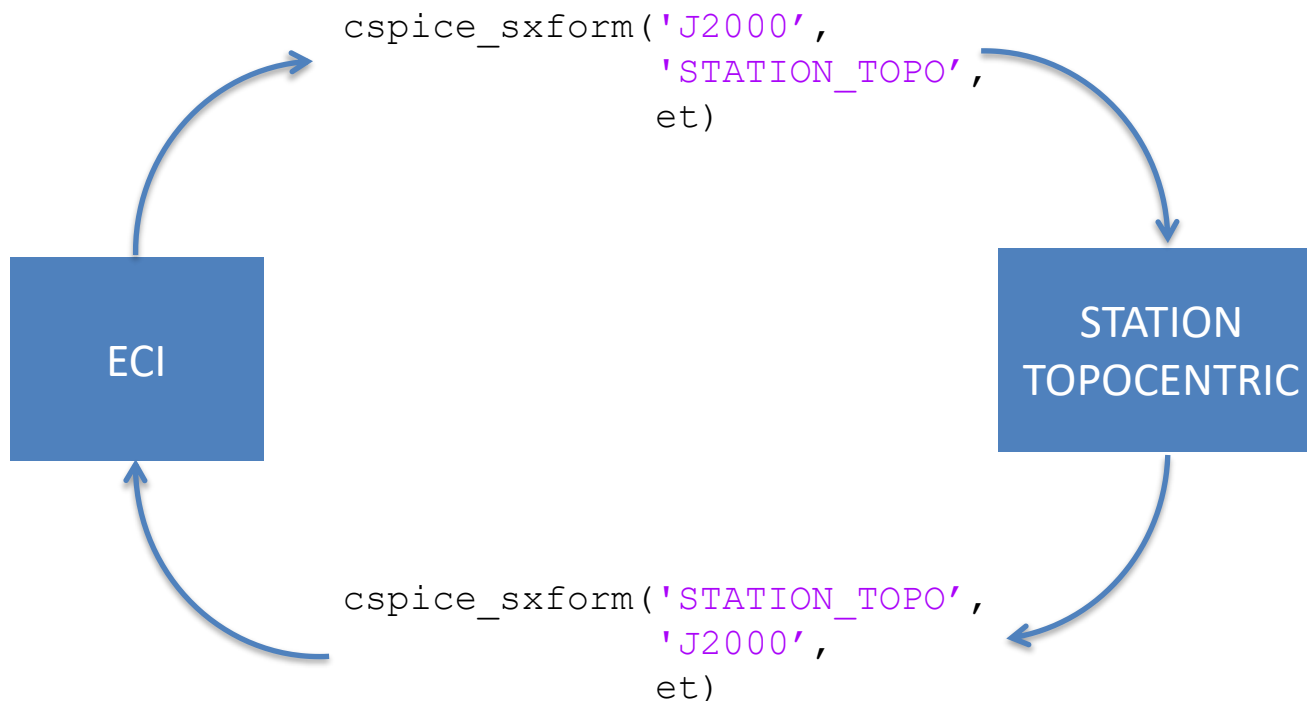
% ----- teme2eci - transform teme to eci vectors
ateme = [0;0;0];
[reci, veci, aeci] = teme2eci(rteme, vteme, ateme, ttt, ddpsi, ddeps);

% ----- eci2teme - transform eci to teme vectors
[rteme, vteme] = eci2teme(reci, veci, aeci, ttt, ddpsi, ddeps);
```


A quick recap (3) – Reference frames

Reference frame conversions

With station kernel generated with pinpoint



Without station kernel

```
% Define station coordinates
lat = 45.50122; % deg
lon = 9.15461; % deg
alt = 20; % m

% Get Earth radii (equatorial and polar)
% (requires pck00010.tpc kernel)
radii = cspice_bodvrd('EARTH', 'RADII', 3);
re = radii(1); rp = radii(3);

% Compute flattening
flat = (re - rp) / re;

% Convert to radians and km
lat_rad = deg2rad(lat); lon_rad = deg2rad(lon);
alt_km = alt / 1000;

% Compute station pos wrt Earth center (ECEF)
pos_ECEF = cspice_pgrrec(...
    'EARTH', lon_rad, lat_rad, alt_km, re, flat);
% Compute ECEF2TOPO rotation matrix
ECEF2TOPO = cspice_eul2m(...
    lat_rad - pi, pi - lon_rad, pi / 2, 2, 1, 2);
```

18

A quick recap (4a) – Generation of multivariate normal random numbers

Use `mvnrnd` to generate multivariate normal random numbers

- **Generate sample points for Monte Carlo simulations**

Example: generate 100 vectors with given mean `[1 3]` and covariance `diag([0.01 0.02])`:

```
n = 100; mu = [1 3]; Sigma = diag([0.01 0.02]);
```

```
R = mvnrnd(mu, Sigma, n); % R size: 100x2
```

- **Add random noise to measurements**

Example: add noise with covariance `diag([0.01 0.02])` to a set of angular measurements :

```
mu = [1 3; 1.5 2; 2 1]; Sigma = diag([0.01 0.02]);
```

```
R = mvnrnd(mu, Sigma); % R size: 3x2
```

Notes:

- When Sigma is diagonal, you can directly pass the diagonal (e.g. `Sigma = [0.01 0.02];`)
- It is possible to define a different Sigma for each measurement (using the 3rd dimension)

A quick recap (4b) – Correlation coefficients

A covariance matrix can be also expressed in terms of correlation coefficients ρ

$$P = \begin{bmatrix} \sigma_x^2 & \rho_{xy}\sigma_x\sigma_y & \rho_{xz}\sigma_x\sigma_z \\ \rho_{xy}\sigma_x\sigma_y & \sigma_y^2 & \rho_{yz}\sigma_y\sigma_z \\ \rho_{xz}\sigma_x\sigma_z & \rho_{yz}\sigma_y\sigma_z & \sigma_z^2 \end{bmatrix} \quad \text{with} \quad \begin{cases} \rho_{xy} = \frac{\sigma_{xy}}{\sigma_x\sigma_y} \\ \rho_{xz} = \frac{\sigma_{xz}}{\sigma_x\sigma_z} \\ \rho_{yz} = \frac{\sigma_{yz}}{\sigma_y\sigma_z} \end{cases}$$

Measurements noise
(noise matrix R)

$$\sigma_{Az} = 1.5 \text{ mdeg}$$

$$\sigma_{El} = 1.3 \text{ mdeg}$$

$$\sigma_{range} = 75 \text{ m}$$

$$\rho_{Az-El} = 0.1$$

$$\rho_{Az-rng} = \rho_{El-rng} = 0$$

$$\sigma_{Az} = 1.5 \text{ mdeg}$$

$$\sigma_{El} = 1.3 \text{ mdeg}$$

$$\sigma_{range} = 75 \text{ m}$$

$$\rho_{Az-El} = 0.1$$

$$\rho_{Az-rng} = \rho_{El-rng} = 0$$

A quick recap (5a) – Non-linear least squares in MATLAB

Solve nonlinear least-squares problems in MATLAB

```
[x, resnorm, residual, exitflag, ~, ~, jac] = lsqnonlin(fun, x0, lb, ub, opt);
```

Inputs:

- `fun`: handle of the cost function to be minimized
 - `fun` must take as input only the `x` and must return the residual
 - the residual can be a vector, `lsqnonlin` implicitly computes the norm
- `x0`: initial point
- `lb`, `ub`: lower and upper bounds
- `opt`: to be defined using the `optimoptions` function

Outputs:

- `x`: solution
- `resnorm`: squared norm of the residual
- `residual`: value of `fun(x)`
- `exitflag`: reason the solver stopped (convergence if `exitflag > 0`)
- `jac`: jacobian matrix of `fun` with respect to `x`

A quick recap (5b) – Non-linear least squares in MATLAB example

Solve nonlinear least-squares problems in MATLAB - example

```
% Variables of the problem
x_0 = ...; var_1 = ...; var_2 = ...;

% Encapsuate the variables in the function handle
fun = @(x) costfunction(x, var_1, var_2, ...);

% Optimization options
opt = optimoptions('lsqnonlin', 'Algorithm', 'levenberg-marquardt', 'Display', 'iter');

% Execute least squares
[x, resnorm, residual, exitflag, ~, ~, jac] = lsqnonlin(fun, x_0, [], [], opt);

% Compute resulting covariance
Jac = full(jac);
P_ls = resnorm / (length(residual)-length(x_0)) .* inv(Jac'*Jac);

function residual = costfunction(x, var_1, var_2, ... )
    residual = []; % Initialize output variable
    % If least squares with "a priori" append it to the residual
    diff_apriori_weighted = W_ap * (x - x0);
    residual = [residual; diff_apriori_weighted(:)];
    % Propagate x to the epochs of the measurements
    x_prop = ...;
    % Compute predicted measurements
    meas_pred = ...;
    % Compute the residual of the measurements and append it to the output
    diff_meas_weighted = W_m * (meas_pred - meas_real);
    residual = [residual; diff_meas_weighted(:)];
end
```

Weight for «a priori»

$$W_{ap} = \text{sqrtm}(\text{inv}(P_0));$$

Weight for measurements:

$$W_m = \text{diag}(1 ./ \text{sigma_meas});$$

A quick recap (6) – UT

$\mathbf{x}, \mathbf{P}_\mathbf{x}$

Create sigma points

$$\begin{aligned}\chi_0 &= \mathbf{x} & c &= \alpha^2(N + \kappa) \\ \chi_i &= \mathbf{x} + \Delta\mathbf{x}_i \text{ for } i = 1, \dots, 2N \\ \Delta\mathbf{x}_i &= (\sqrt{c\mathbf{P}_\mathbf{x}})_i \text{ for } i = 1, \dots, N \\ \Delta\mathbf{x}_{i+N} &= -(\sqrt{c\mathbf{P}_\mathbf{x}})_i \text{ for } i = 1, \dots, N\end{aligned}$$

χ

Apply transformation

$$\gamma = f(\chi)$$

NB:

- N is the size of the state
- $(\sqrt{c\mathbf{P}_\mathbf{x}})_i$ indicates the i -th column of the matrix square root (*sqrtm* matlab)

Parameters to be selected (with common values)

$$\alpha = 1e - 3$$

$$\beta = 2$$

$$\kappa = 0$$

Compute mean and covariance

$$W_0^{(m)} = 1 - \frac{N}{\alpha^2(N + \kappa)} \quad W_0^{(c)} = (2 - \alpha^2 + \beta) - \frac{N}{\alpha^2(N + \kappa)}$$

$$W_i^{(m)} = \frac{1}{2\alpha^2(N + \kappa)} \text{ for } i = 1, \dots, 2N \quad W_i^{(c)} = \frac{1}{2\alpha^2(N + \kappa)} \text{ for } i = 1, \dots, 2N$$

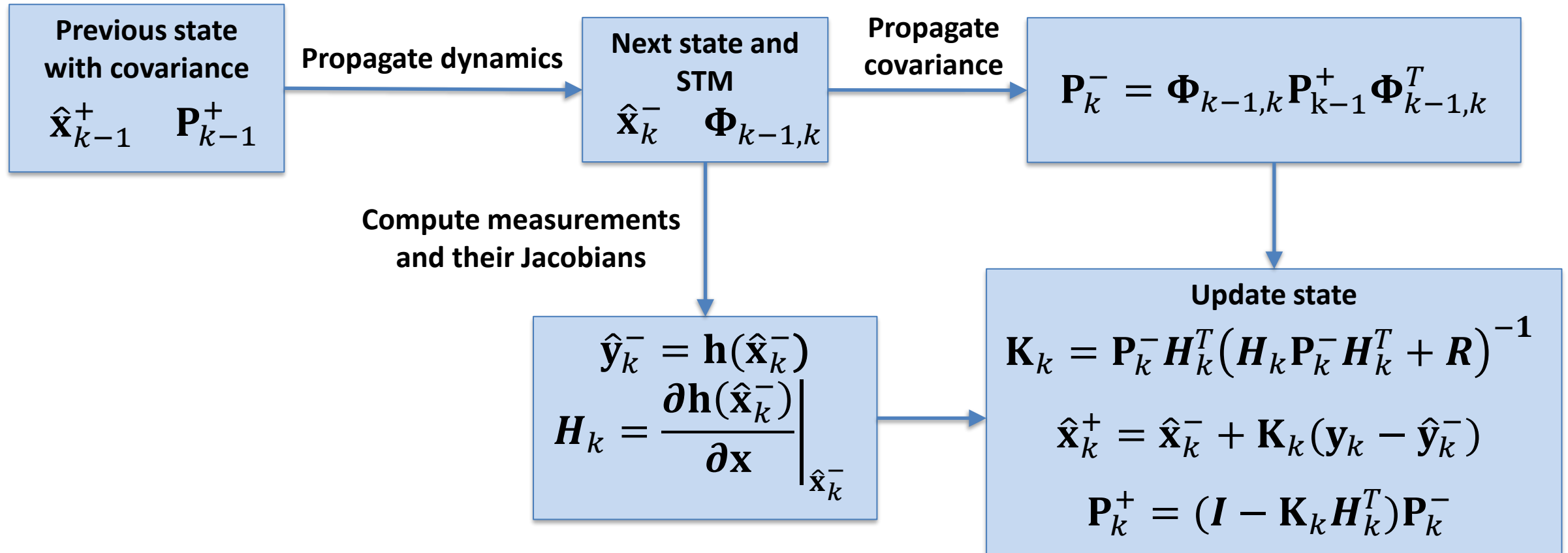
$$\mathbf{y} = \sum_{i=0}^{2N} W_i^{(m)} \gamma_i$$

$$\mathbf{P}_\mathbf{y} = \sum_{i=0}^{2N} W_i^{(c)} [\gamma_i - \mathbf{y}][\gamma_i - \mathbf{y}]^T$$

REF: Van der Merwe, Rudolph, and Eric A. Wan. "The Square-Root Unscented Kalman Filter for State and Parameter-Estimation." 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221), 6:3461–64. Salt Lake City, UT, USA: IEEE, 2001. <https://doi.org/10.1109/ICASSP.2001.940586>.

A quick recap (7) – EKF

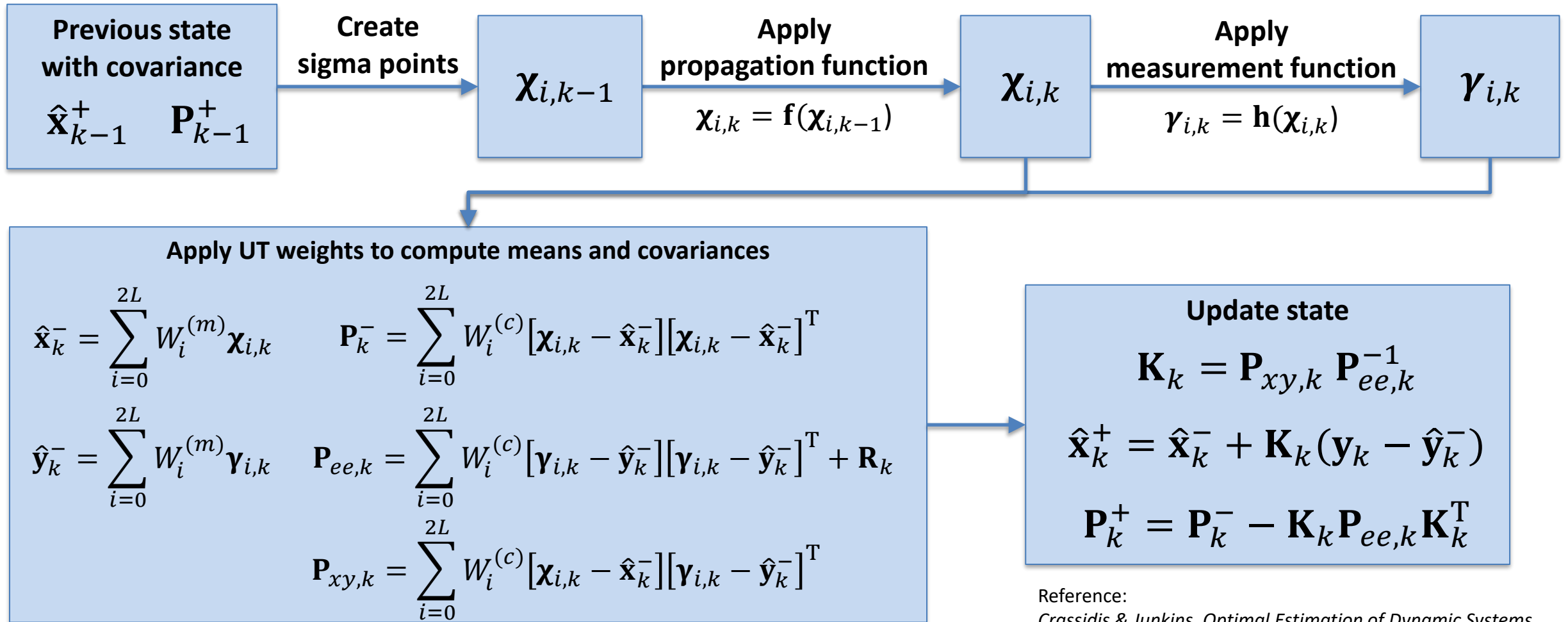
Extended Kalman Filter (EKF)



Reference:
Crassidis & Junkins, Optimal Estimation of Dynamic Systems

A quick recap (8) – UKF

Unscented Kalman Filter (UKF)

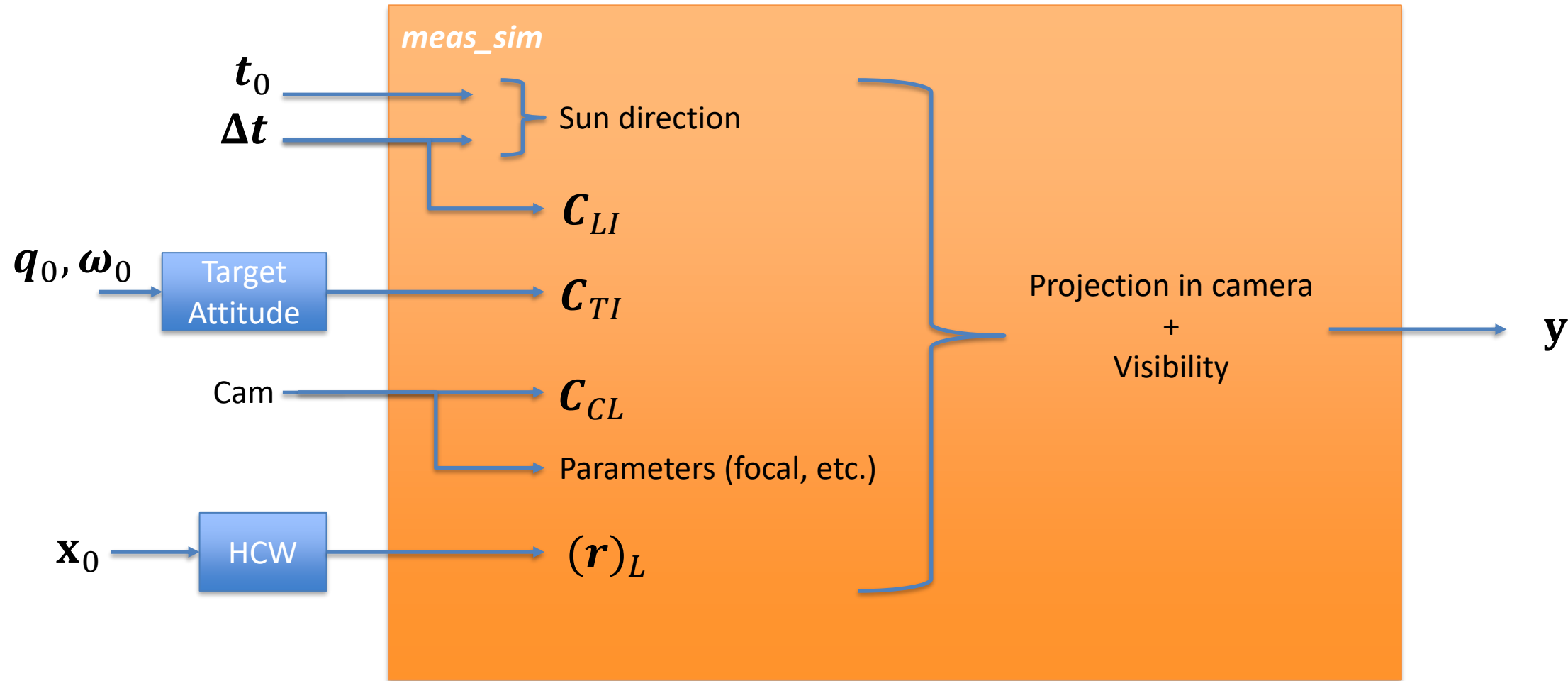


Reference:
Crassidis & Junkins, *Optimal Estimation of Dynamic Systems*

26

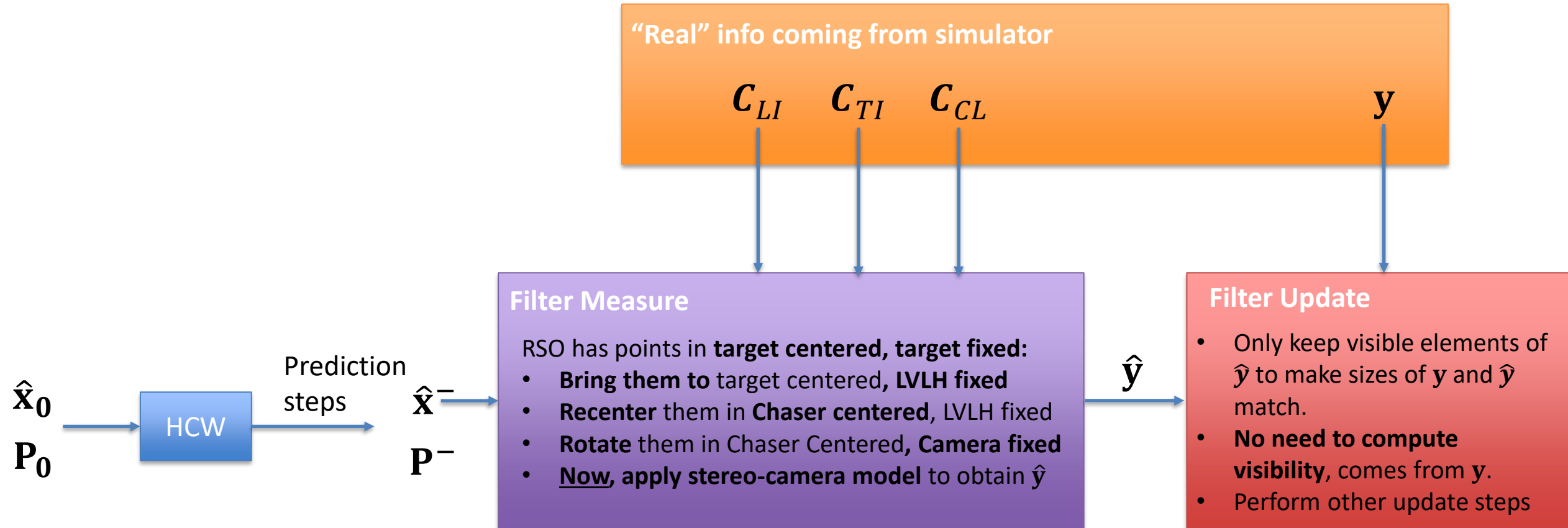
Some help on Ex 3 (8)

You will have to create a “simulator” of reality which is **separate** from the filter state. It uses **real initial conditions (see tutorial)**



Some help on Ex 3 (8)

Simulator is **separate** from the filter; however, some info need to be leaked to the filter!



General hints

Report content

- **State vectors** must be provided with corresponding **reference frame and origin**
- Always put the (correct!) units
- Do not mix position and velocities when reporting errors

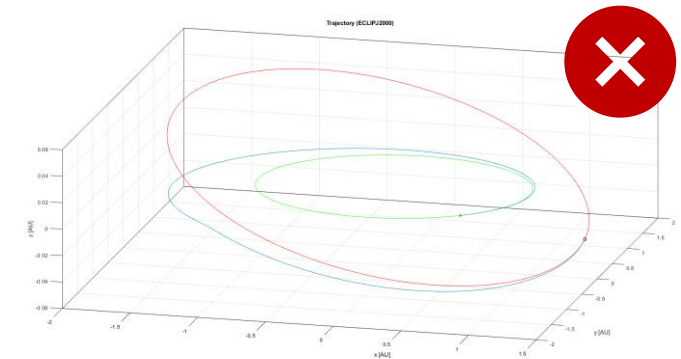
r_x [km]	r_y [km]	r_z [km]
1234.567	123.456	234.567

Satellite position (@Sun ECLIPJ2000)



Plots

- Make sure that axis labels and titles are clearly readable with page zoom at 100%
- Remember to put the (correct!) units on each axis
- When plotting trajectories specify the **reference frame and origin** in the plot title or caption



General hints

LaTeX

- Do not put multiplication symbols, especially as asterisks
- Clearly distinguish between vectors and scalar: write vector using underline, arrow or bold
 - Latin alphabet: `\mathbf{x}` or `\mathbf{r}`
 - Greek symbols: `\boldsymbol{\lambda}`
 - **Obs:** No need to use norm with this notation $\|\lambda_x\| \leftrightarrow \lambda_x$
- Write scalar product using `\cdot`
- Prefer the use of `\dfrac{ }{ }` over `\frac{ }{ }` when writing equations
- When inserting text in an equation remember to use `\mathrm{ }`