Assignment 02 - Navigation

Spacecraft Guidance and Navigation

AY 2024-2025

POLITECNICO
MILANO 1863

Exam

➢ **Remarks**:

- The deadline for the submission of assignment 2 is on **20/12/2024 23:59:00 CET**.
  - ➢ An extension of a few days is possible only for well motivated issues.
- Those who will not be able to submit within the deadline will perform the exam on the following exam date (currently on January 30).
  - ➢ The deadline for taking part to this second session is on **20/01/2024** for both Assignment 1 and 2.

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 (CTRL+F)**
  - 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
27 Nov	14.15-16.15	BL.27.03 (+ Bonaccorsi virtual room)
29 Nov	12.15-14.15	B8.2.2 (+ Bonaccorsi virtual room)
03 Dec	16.15-18.15	LM.1 (+ Bonaccorsi virtual room)
04 Dec	14.15-16.15	BL.27.03 (+ Bonaccorsi virtual room)
06 Dec	11.15-14.15	B8.2.2 (+ Bonaccorsi virtual room)
10 Dec	16.15-18.15	LM.1 (+ Bonaccorsi virtual room)
13 Dec	11.15-14.15	B8.2.2 (+ Bonaccorsi virtual room)
20 Dec	**23.59 CET**	**Assignment 2 deadline**

Delivery of assignments

Assigment will be delivered through **WeBeep**:

DEADLINE $\Rightarrow$ Dec 20, 2024, 23:59:00 CET

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

https://bit.ly/SGN_24_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

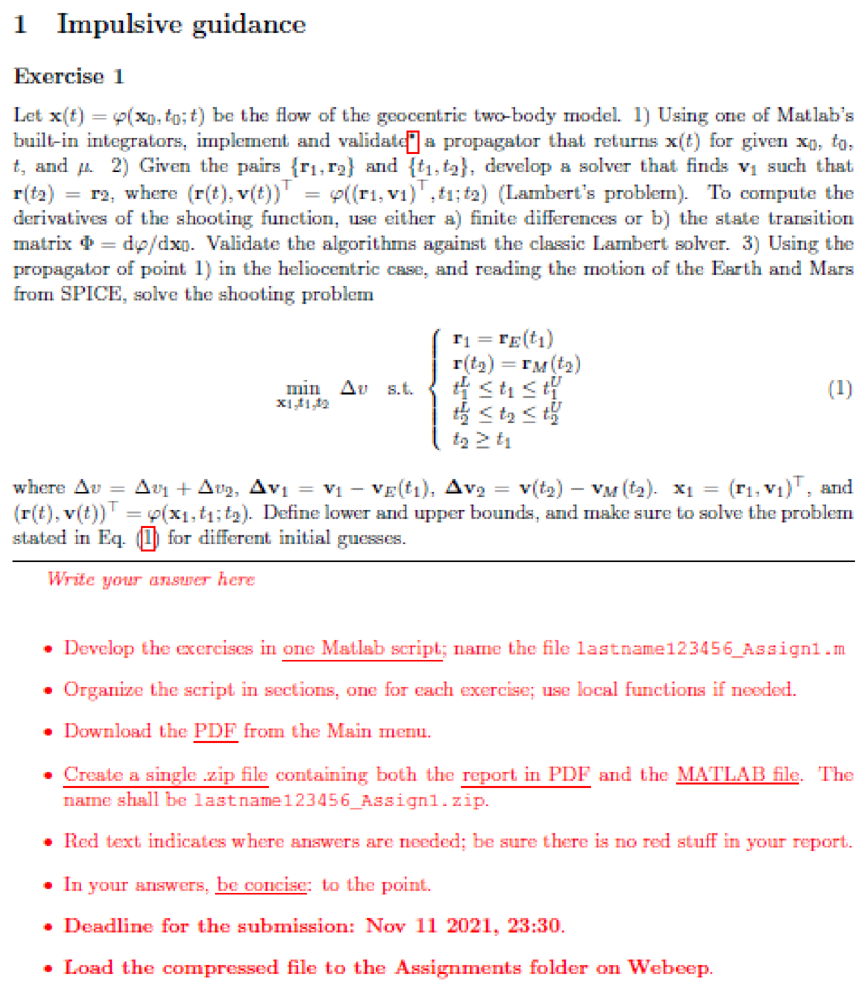
Assignment 02 – Topics

3 Exercises

- Uncertainty propagation
- Batch filters
- Sequential filters

Suggested MATLAB Version: R2021b (or newer)

Report



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 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 $(\mathbf{r}(t), \mathbf{v}(t))^\top = \varphi((\mathbf{r}_1, \mathbf{v}_1)^\top, t_1; t_2)$ (Lambert's problem). To compute the derivatives of the shooting function, use either a) finite differences or b) the state transition matrix $\Phi = \mathrm{d}\varphi/\mathrm{d}\mathbf{x}_0$. 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

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

where $\Delta v = \Delta v_1 + \Delta v_2$, $\mathbf{\Delta v}_1 = \mathbf{v}_1 - \mathbf{v}_E(t_1)$, $\mathbf{\Delta v}_2 = \mathbf{v}(t_2) - \mathbf{v}_M(t_2)$. $\mathbf{x}_1 = (\mathbf{r}_1, \mathbf{v}_1)^\top$, and $(\mathbf{r}(t), \mathbf{v}(t))^\top = \varphi(\mathbf{x}_1, t_1; t_2)$. 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 <u>one Matlab script</u>; name the file `lastname123456_Assign1.m`
- Organize the script in sections, one for each exercise; use local functions if needed.
- Download the <u>PDF</u> from the Main menu.
- <u>Create a single .zip file</u> containing both the <u>report in PDF</u> and the <u>MATLAB file</u>. 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, <u>be concise</u>: to the point.
- **Deadline for the submission: Nov 11 2021, 23:30.**
- **Load the compressed file to the Assignments folder on Webeep.**

Exercise

Answer here

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!

POLITECNICO MILANO 1863

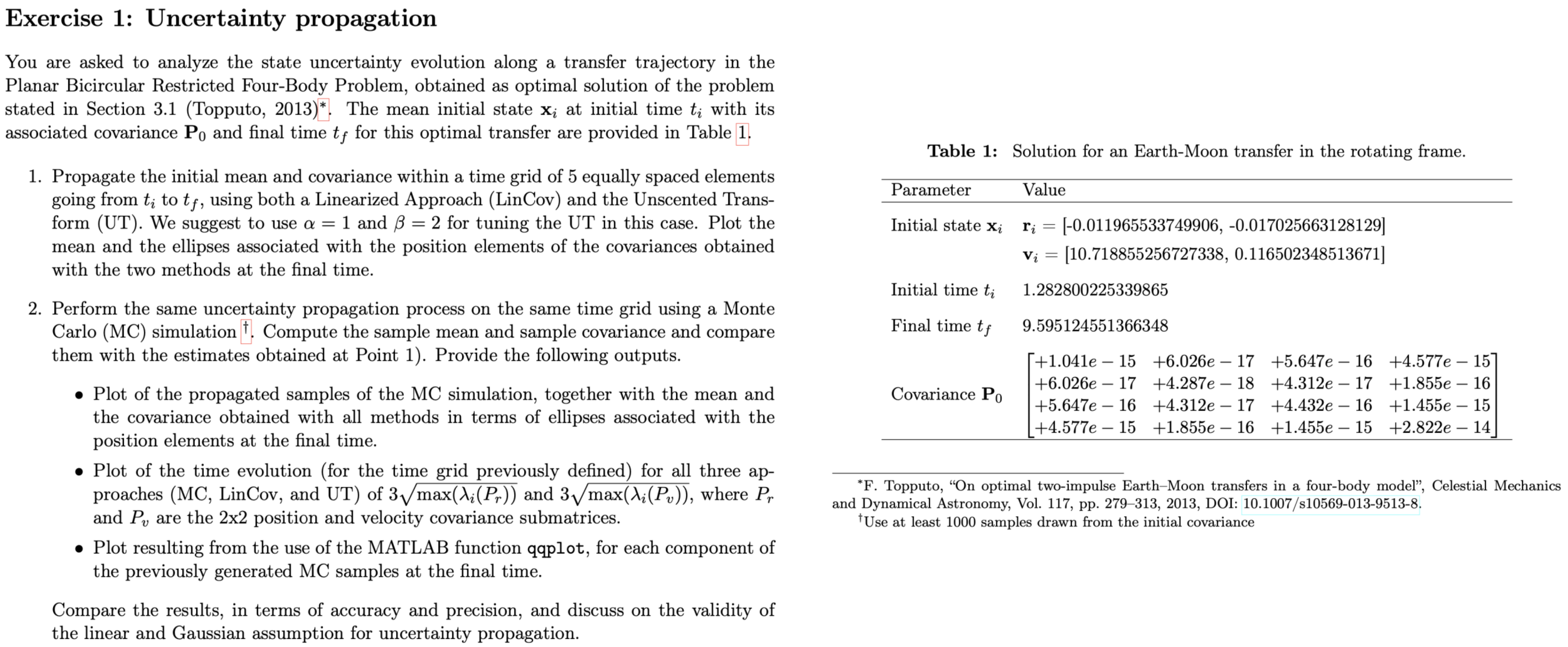
Script

- Name of the script (one per exercise): `lastname123456_Assign2_Ex1.m`
- Header:

```
% Spacecraft Guidance and Navigation (2023/2024)
% Assignment # 2, Exercise 1
% Author: Name Lastname
```

- Functions in a section at the end of the script: `%% Functions`

Exercise 1 – Uncertainty propagation



Exercise 1: Uncertainty propagation

You are asked to analyze the state uncertainty evolution along a transfer trajectory in the Planar Bicircular Restricted Four-Body Problem, obtained as optimal solution of the problem stated in Section 3.1 (Topputo, 2013)[*]. The mean initial state $\mathbf{x}_i$ at initial time t_i with its associated covariance $\mathbf{P}_0$ and final time t_f for this optimal transfer are provided in Table 1.

1. Propagate the initial mean and covariance within a time grid of 5 equally spaced elements going from t_i to t_f, using both a Linearized Approach (LinCov) and the Unscented Transform (UT). We suggest to use $\alpha = 1$ and $\beta = 2$ for tuning the UT in this case. Plot the mean and the ellipses associated with the position elements of the covariances obtained with the two methods at the final time.

2. Perform the same uncertainty propagation process on the same time grid using a Monte Carlo (MC) simulation [†]. Compute the sample mean and sample covariance and compare them with the estimates obtained at Point 1). Provide the following outputs.

   - Plot of the propagated samples of the MC simulation, together with the mean and the covariance obtained with all methods in terms of ellipses associated with the position elements at the final time.
   - Plot of the time evolution (for the time grid previously defined) for all three approaches (MC, LinCov, and UT) of $3\sqrt{\max(\lambda_i(P_r))}$ and $3\sqrt{\max(\lambda_i(P_v))}$, where P_r and P_v are the 2x2 position and velocity covariance submatrices.
   - Plot resulting from the use of the MATLAB function `qqplot`, for each component of the previously generated MC samples at the final time.

   Compare the results, in terms of accuracy and precision, and discuss on the validity of the linear and Gaussian assumption for uncertainty propagation.

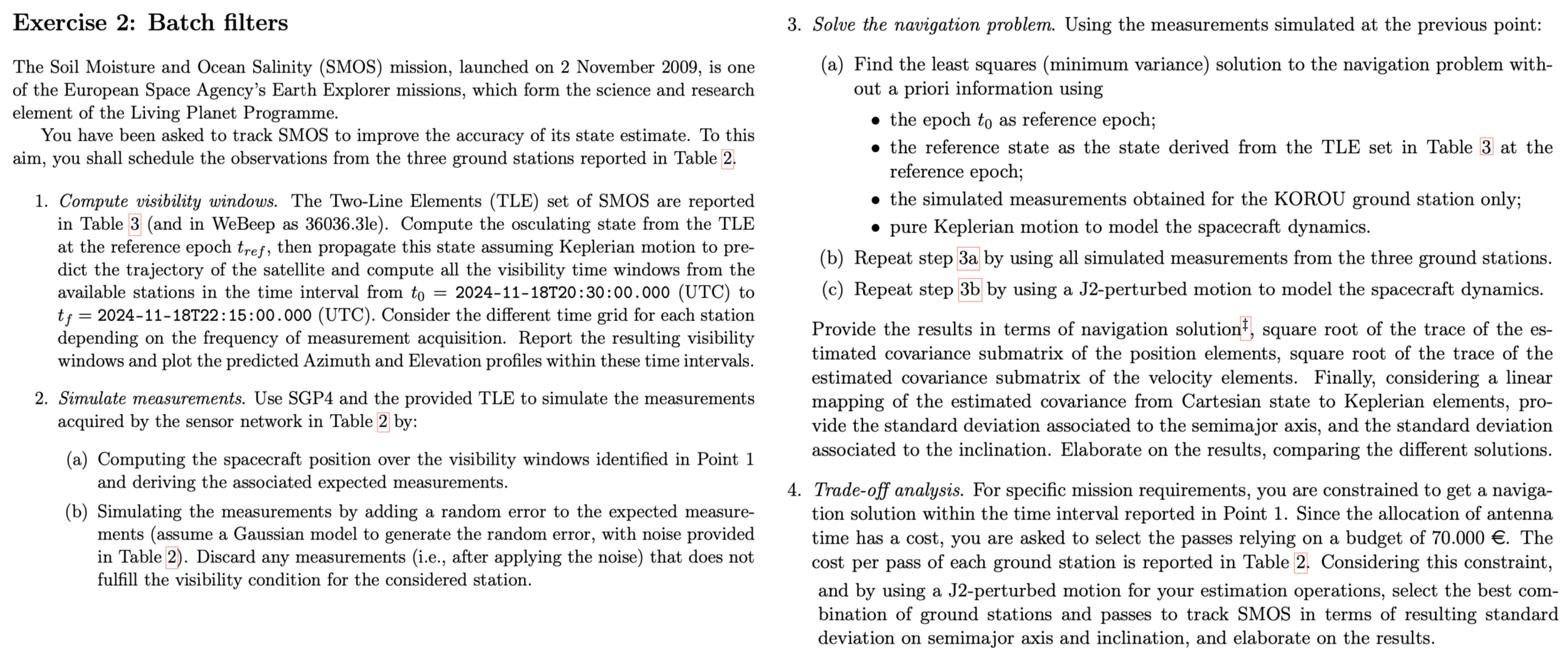
Table 1: Solution for an Earth-Moon transfer in the rotating frame.

Parameter	Value
Initial state $\mathbf{x}_i$	$\mathbf{r}_i$ = [-0.011965533749906, -0.017025663128129]
	$\mathbf{v}_i$ = [10.718855256727338, 0.116502348513671]
Initial time t_i	1.282800225339865
Final time t_f	9.595124551366348
Covariance $\mathbf{P}_0$	$\begin{bmatrix} +1.041e-15 & +6.026e-17 & +5.647e-16 & +4.577e-15 \\ +6.026e-17 & +4.287e-18 & +4.312e-17 & +1.855e-16 \\ +5.647e-16 & +4.312e-17 & +4.432e-16 & +1.455e-15 \\ +4.577e-15 & +1.855e-16 & +1.455e-15 & +2.822e-14 \end{bmatrix}$

[*]F. Topputo, "On optimal two-impulse Earth–Moon transfers in a four-body model", Celestial Mechanics and Dynamical Astronomy, Vol. 117, pp. 279–313, 2013, DOI: 10.1007/s10569-013-9513-8.

[†]Use at least 1000 samples drawn from the initial covariance

Exercise 2 – Batch filters



Exercise 2: Batch filters

The Soil Moisture and Ocean Salinity (SMOS) mission, launched on 2 November 2009, is one of the European Space Agency's Earth Explorer missions, which form the science and research element of the Living Planet Programme.

You have been asked to track SMOS to improve the accuracy of its state estimate. To this aim, you shall schedule the observations from the three ground stations reported in Table 2.

1. *Compute visibility windows.* The Two-Line Elements (TLE) set of SMOS are reported in Table 3 (and in WeBeep as 36036.3le). Compute the osculating state from the TLE at the reference epoch t_{ref}, then propagate this state assuming Keplerian motion to predict the trajectory of the satellite and compute all the visibility time windows from the available stations in the time interval from t_0 = `2024-11-18T20:30:00.000` (UTC) to t_f = `2024-11-18T22:15:00.000` (UTC). Consider the different time grid for each station depending on the frequency of measurement acquisition. Report the resulting visibility windows and plot the predicted Azimuth and Elevation profiles within these time intervals.

2. *Simulate measurements.* 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). Discard any measurements (i.e., after applying the noise) that does not fulfill the visibility condition for the considered station.

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 KOROU ground station only;
   - pure Keplerian motion to model the spacecraft dynamics.

   (b) Repeat step 3a by using all simulated measurements from the three ground stations.

   (c) Repeat step 3b by using a J2-perturbed motion to model the spacecraft dynamics.

   Provide the results in terms of navigation solution[‡], square root of the trace of the estimated covariance submatrix of the position elements, square root of the trace of the estimated covariance submatrix of the velocity elements. Finally, considering a linear mapping of the estimated covariance from Cartesian state to Keplerian elements, provide the standard deviation associated to the semimajor axis, and the standard deviation associated to the inclination. Elaborate on the results, comparing the different solutions.

4. *Trade-off analysis.* For specific mission requirements, you are constrained to get a navigation solution within the time interval reported in Point 1. Since the allocation of antenna time has a cost, you are asked to select the passes relying on a budget of 70.000 €. The cost per pass of each ground station is reported in Table 2. Considering this constraint, and by using a J2-perturbed motion for your estimation operations, select the best combination of ground stations and passes to track SMOS in terms of resulting standard deviation on semimajor axis and inclination, and elaborate on the results.

POLITECNICO MILANO 1863

Exercise 2 – Batch filters

5. *Long-term analysis.* Consider a nominal operations scenario (i.e., you are not constrained to provide a navigation solution within a limited amount of time). In this context, or for long-term planning in general, you could still acquire measurements from multiple locations but you are tasked to select a set of prime and backup ground stations. For planning purposes, it is important to have regular passes as this simplifies passes scheduling activities. Considering the need to have *reliable* orbit determination and *repeatable* measurements, discuss your choices and compare them with the results of Point 4.

Table 3: TLE of SMOS.

```
1␣36036U␣09059A␣␣␣24323.76060260␣␣.00000600␣␣00000-0␣␣20543-3␣0␣␣9995
2␣36036␣␣98.4396␣148.4689␣0001262␣␣95.1025␣265.0307␣14.39727995790658
```

Table 2: Sensor network to track SMOS: list of stations, including their features.

Station name	KOUROU	TROLL	SVALBARD
Coordinates	LAT = 5.25144° LON = -52.80466° ALT = -14.67 m	LAT = -72.011977° LON = 2.536103° ALT = 1298 m	LAT = 78.229772° LON = 15.407786° ALT = 458 m
Type	Radar (monostatic)	Radar (monostatic)	Radar (monostatic)
Measurements type	Az, El [deg] Range (one-way) [km]	Az, El [deg] Range (one-way) [km]	Az, El [deg] Range (one-way) [km]
Measurements noise (diagonal noise matrix R)	$\sigma_{Az,El}$ = 125 mdeg σ_{range} = 0.01 km	$\sigma_{Az,El}$ = 125 mdeg σ_{range} = 0.01 km	$\sigma_{Az,El}$ = 125 mdeg σ_{range} = 0.01 km
Minimum elevation	6 deg	0 deg	8 deg
Measurement frequency	60 s	30 s	60 s
Cost per pass	30.000 €	35.000 €	35.000 €

! Do not copy-paste from PDF

Exercise 3 – Sequential filters

Exercise 3: Sequential filters

An increasing number of lunar exploration missions will take place in the next years, many of them aiming at reaching the Moon's surface with landers. In order to ensure efficient navigation performance for these future missions, space agencies have plans to deploy lunar constellations capable of providing positioning measurements for satellites orbiting around the Moon.

Considering a lander on the surface of the Moon, you have been asked to improve the accuracy of the estimate of its latitude and longitude (considering a fixed zero altitude). To perform such operation you can rely on the use of a lunar orbiter, which uses its Inter-Satellite Link (ISL) to acquire range measurements with the lander while orbiting around the Moon. At the same time, assuming the availability of a Lunar Navigation Service, you are also receiving measurements of the lunar orbiter inertial position vector components, such that you can also estimate the spacecraft state within the same state estimation process.
To perform the requested tasks you can refer to the following points.

1. *Check the visibility window.* Considering the initial state $\mathbf{x}_0$ and the time interval with a time-step of 30 seconds from t_0 to t_f reported in Table 4, predict the trajectory of the satellite in an inertial Moon-centered reference frame assuming Keplerian motion. Use the estimated coordinates given in Table 5 to predict the state of the lunar lander. Finally, check that the lander and the orbiter are in relative visibility for the entire time interval.

2. *Simulate measurements.* Always assuming Keplerian motion to model the lunar orbiter dynamics around the Moon, compute the time evolution of its position vector in an inertial Moon-centered reference frame and the time evolution of the relative range between the satellite and the lunar lander. Finally, simulate the measurements by adding a random error to the spacecraft position vector and to the relative range. Assume a Gaussian model to generate the random error, with noise provided in Table 4 for both the relative range and the components of the position vector. Verify (graphically) that the applied noise level is within the desired boundary.

3. *Estimate the lunar orbiter absolute state.* As a first step, you are asked to develop a sequential filter to narrow down the uncertainty on the knowledge of the lunar orbiter absolute state vector. To this aim, you can exploit the measurements of the components of its position vector computed at the previous point. Using an Unscented Kalman Filter (UKF), provide an estimate of the spacecraft state (in terms of mean and covariance) by sequentially processing the acquired measurements in chronological order. To initialize the filter in terms of initial covariance, you can refer to the first six elements of the initial covariance $\mathbf{P}_0$ reported in Table 4. For the initial state, you can perturb the actual initial state $\mathbf{x}_0$ by exploiting the MATLAB function `mvnrnd` and the previously mentioned initial covariance. We suggest to use $\alpha = 0.01$ and $\beta = 2$ for tuning the UT in this case. Plot the time evolution of the error estimate together with the 3σ of the estimated covariance for both position and velocity.

Exercise 3 – Sequential filters

4. *Estimate the lunar lander coordinates.* To fulfill the goal of your mission, you are asked to develop a sequential filter to narrow down the uncertainty on the knowledge of the lunar lander coordinates (considering a fixed zero altitude). To this aim, you can exploit the measurements of the components of the lunar orbiter position vector together with the measurements of the relative range between the orbiter and the lander computed at the previous point. Using an UKF, provide an estimate of the spacecraft state and the lunar lander coordinates (in terms of mean and covariance) by sequentially processing the acquired measurements in chronological order. To initialize the filter in terms of initial covariance, you can refer to the initial covariance $\mathbf{P}_0$ reported in Table 4. For the initial state, you can perturb the actual initial state, composed by $\mathbf{x}_0$ and the latitude and longitude given in Table 5, by exploiting the MATLAB function `mvnrnd` and the previously mentioned initial covariance. We suggest to use $\alpha = 0.01$ and $\beta = 2$ for tuning the UT in this case. Plot the time evolution of the error estimate together with the 3σ of the estimated covariance for both position and velocity.

Table 4: Initial conditions for the lunar orbiter.

Parameter	Value
Initial state $\mathbf{x}_0$ [km, km/s]	$\mathbf{r}_0$ = [4307.844185282820, -1317.980749248651, 2109.210101634011] $\mathbf{v}_0$ = [-0.110997301537882, -0.509392750828585, 0.815198807994189]
Initial time t_0 [UTC]	2024-11-18T16:30:00.000
Final time t_f [UTC]	2024-11-18T20:30:00.000
Measurements noise	$\sigma_p = 100$ m
Covariance $\mathbf{P}_0$ [km^2, km^2/s, km^2/s^2]	`diag([10,1,1,0.001,0.001,0.001,0.00001,0.00001])`

Table 5: Lunar lander - initial guess coordinates and horizon mask

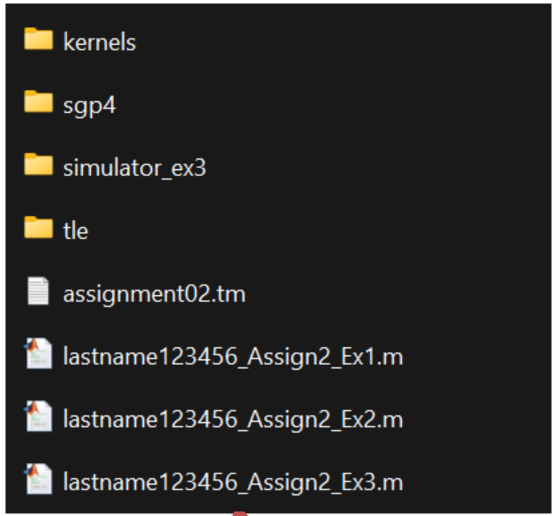
Lander name	MOONLANDER

Coordinates	LAT = 78°
	LON = 15°
	ALT = 0 m

Minimum elevation	0 deg

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 (Lab04)
  - Subfolder `tle` with two-line elements
  - Meta-kernel `assignment02.tm`



kernels
sgp4
simulator_ex3
tle
assignment02.tm
lastname123456_Assign2_Ex1.m
lastname123456_Assign2_Ex2.m
lastname123456_Assign2_Ex3.m

lastname123456_Assign2_Ex1.m

lastname123456_Assign2_Ex2.m

lastname123456_Assign2_Ex3.m

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.

POLITECNICO MILANO 1863

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:

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

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

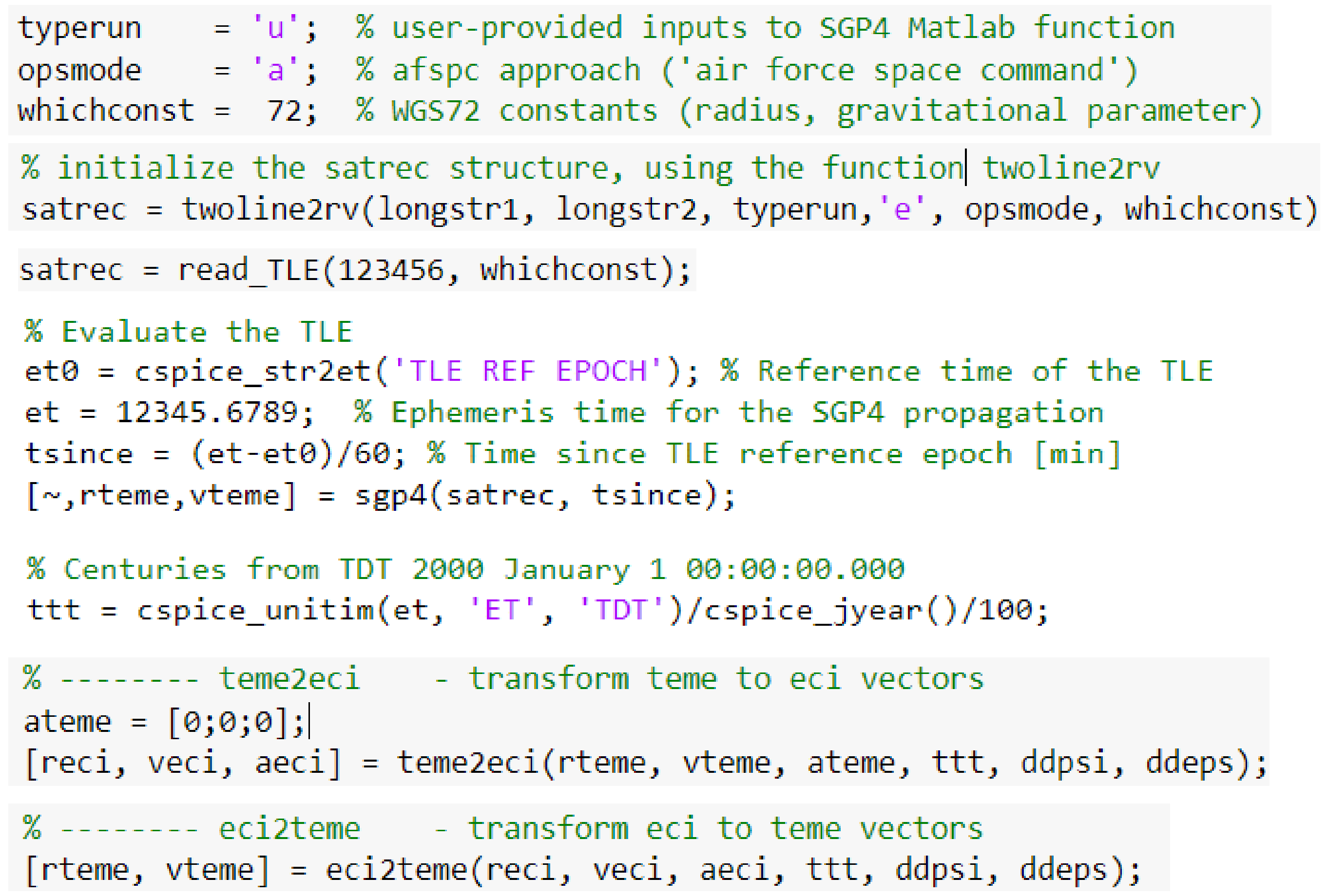
A quick recap (2) – SGP4/SDP4

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`



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

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

```matlab
satrec = read_TLE(123456, whichconst);
```

```matlab
% 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);
```

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

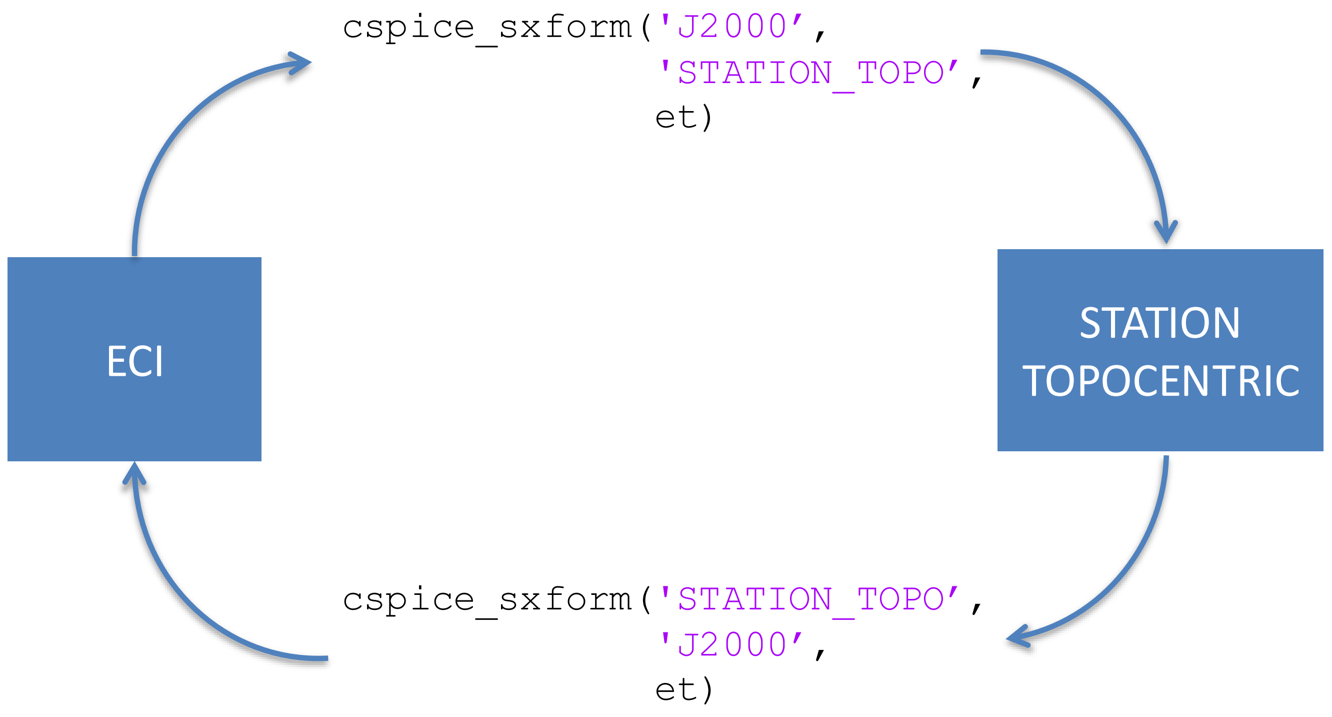
```matlab
% -------- teme2eci    - transform teme to eci vectors
ateme = [0;0;0];
[reci, veci, aeci] = teme2eci(rteme, vteme, ateme, ttt, ddpsi, ddeps);
```

```matlab
% -------- 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`



```
cspice_sxform('J2000',
              'STATION_TOPO',
              et)
```

```
cspice_sxform('STATION_TOPO',
              'J2000',
              et)
```

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);
```

A quick recap (4) – 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 (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 = ...;

% Encapsulate 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
    % 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 measurements:

```
W_m = diag(1 ./ sigma_meas);
```

A quick recap (6) – UT

$$\begin{tikzcd}
\mathbf{x}, \mathbf{P}_\mathbf{x} \arrow[r] & \text{Create sigma points} \arrow[r] & \boldsymbol{\chi} \arrow[r] & \text{Apply transformation} \arrow[d] \\
 & & & \text{Compute mean and covariance}
\end{tikzcd}$$

Create sigma points

$$\boldsymbol{\chi}_0 = \mathbf{x} \qquad c = \alpha^2(N+\kappa)$$

$$\boldsymbol{\chi}_i = \mathbf{x} + \Delta\mathbf{x}_i \text{ for } i = 1, \dots, 2N$$

$$\Delta\mathbf{x}_i = \left(\sqrt{c\mathbf{P}_\mathbf{x}}\right)_i \text{ for } i = 1, \dots, N$$

$$\Delta\mathbf{x}_{i+N} = -\left(\sqrt{c\mathbf{P}_\mathbf{x}}\right)_i \text{ for } i = 1, \dots, N$$

Apply transformation

$$\boldsymbol{\gamma} = \mathrm{f}(\boldsymbol{\chi})$$

NB:

- N is the size of the state
- $\left(\sqrt{c\mathbf{P}_\mathbf{x}}\right)_i$ 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

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

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

$$\mathbf{y} = \sum_{i=0}^{2N} W_i^{(m)} \boldsymbol{\gamma}_i \qquad \mathbf{P}_\mathbf{y} = \sum_{i=0}^{2N} W_i^{(c)} [\boldsymbol{\gamma}_i - \mathbf{y}][\boldsymbol{\gamma}_i - \mathbf{y}]^\mathrm{T}$$

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 (8) – UKF

Unscented Kalman Filter (UKF)

Previous state with covariance

$$\hat{\mathbf{x}}_{k-1}^{+} \qquad \mathbf{P}_{k-1}^{+}$$

$$\begin{tikzcd}
\boxed{\hat{\mathbf{x}}_{k-1}^{+} \quad \mathbf{P}_{k-1}^{+}} \arrow[r, "\text{Create sigma points}"] & \boxed{\boldsymbol{\chi}_{i,k-1}} \arrow[r, "\text{Apply propagation function}", "\boldsymbol{\chi}_{i,k} = \mathbf{f}(\boldsymbol{\chi}_{i,k-1})"'] & \boxed{\boldsymbol{\chi}_{i,k}} \arrow[r, "\text{Apply measurement function}", "\boldsymbol{\gamma}_{i,k} = \mathbf{h}(\boldsymbol{\chi}_{i,k})"'] & \boxed{\boldsymbol{\gamma}_{i,k}}
\end{tikzcd}$$

Apply UT weights to compute mean and covariances

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \boldsymbol{\chi}_{i,k} \qquad \mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\boldsymbol{\chi}_{i,k} - \hat{\mathbf{x}}_k^-][\boldsymbol{\chi}_{i,k} - \hat{\mathbf{x}}_k^-]^{\mathrm{T}}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \boldsymbol{\gamma}_{i,k} \qquad \mathbf{P}_{ee,k} = \sum_{i=0}^{2L} W_i^{(c)} [\boldsymbol{\gamma}_{i,k} - \hat{\mathbf{y}}_k^-][\boldsymbol{\gamma}_{i,k} - \hat{\mathbf{y}}_k^-]^{\mathrm{T}} + \mathbf{R}_k$$

$$\mathbf{P}_{xy,k} = \sum_{i=0}^{2L} W_i^{(c)} [\boldsymbol{\chi}_{i,k} - \hat{\mathbf{x}}_k^-][\boldsymbol{\gamma}_{i,k} - \hat{\mathbf{y}}_k^-]^{\mathrm{T}}$$

Update state

$$\mathbf{K}_k = \mathbf{P}_{xy,k}\, \mathbf{P}_{ee,k}^{-1}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-)$$

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{ee,k} \mathbf{K}_k^{\mathrm{T}}$$

Reference:
Crassidis & Junkins, Optimal Estimation of Dynamic Systems

General hints

Report content

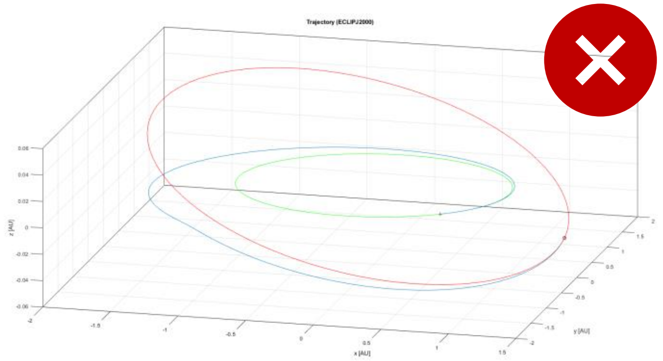
- **State vectors** must be provided with corresponding **reference frame and origin**
- Always put the (correct!) units
- Do not mix position and velocities when reporting errors

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 $\|\boldsymbol{\lambda}_x\| \leftrightarrow \lambda_x$
- 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{}`