Navigation Function justification file

Author : Gérard Grigore Bousquet

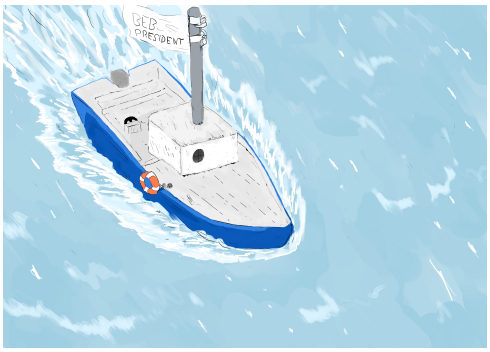

Figure 1: copyright - Jordi Oprisa Bousquet

Object : GN&C Navigation Function design justification file
Version : Simulator version 9
GitHub Release :
https://github.com/GerardGrigore/Casteldos_GNC/releases/tag/0.0.4

Contents

Acronym list

EFK Extended Kalman Filter

GNC Guidance Navigation and Control

GNSS Global Navigation Satellite System

GPS Global Positioning System

INS Inertial Navigation System

KF Kalman Filter

LS Least Squares

MAG Magnetometer

ReLS Regularized Least Squares

WLS Weighted Leasat Squares

Chapter 1

The Navigation function in the GN&C loop

This file gathers the mathematical equations and algorithms for the Navigation function embedded in the Software and used by the computer to estimate its states. In a general way, the Navigation function serves to estimate the current parameters of the ship and to answer the broader question, "Where is the ship actually located ?". This question involves the notion of position on a map; however, the Navigation function is also responsible for estimating the velocities, attitude angles, and angular rates, for example. This is the discipline dealing with the sensors and their signal processing. Both aspects of hardware and software are treated by this entity.

In this explanatory document, several mathematical derivations of well-known fundamental results from information and signal processing theory are revisited. This work stems from a personal exploratory initiative with educational objectives, aiming to gain a deeper understanding of the underlying mathematical principles behind these classical methods. The document is also intended to demonstrate a solid mastery of foundational concepts in algebra, geometry, probability theory, and mathematical optimization. Consequently, even though some derivations may not be strictly necessary for the practical implementation of the associated algorithms, they are included here for explanatory and self-training purposes.

1.1 Guidance, Navigation & Control

The design of a robust estimation function is crucial since the Guidance and Control functions will produce orders and actions based on what has actually been measured and estimated by the Navigation function. Therefore, special care was taken when designing this part of the algorithms. As a reminder, the following figure can be useful for understanding the flow of the data through a mission and on the embedded computer inside the ship:

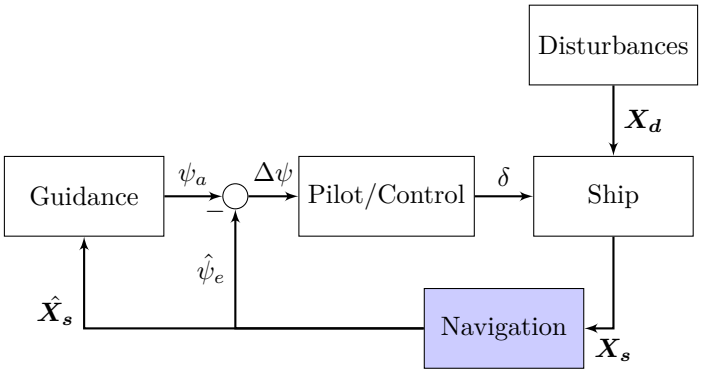

Figure 1.1: GN&C loop for heading control and path-following.

Without loss of generality, one can assume that several environmental perturbations can act on the ship,

such as wind, current, waves and wave-induced for example. Therefore the vector $\boldsymbol{X_d}$ was used on the previous figure, assuming that several components can act on the vessel. In the last version of Simulator Case 9 for instance, only the waves are considered to perturb the motion of the ship by a contribution ψ_w on the total heading. In this particular case, the vector $\boldsymbol{X_d}$ becomes a scalar ψ_w. Additionally, the motion of the ship is captured through the state vector $\boldsymbol{X_s}$. It is in fact composed by the total heading of the ship ψ_t, its longitudinal position x_s and its lateral position y_s in the local horizontal plane.

In function of the estimated states of the ship $\hat{\boldsymbol{X}}_{\boldsymbol{s}}$; the heading $\hat{\psi}_e$, the local longitudinal estimated position $\hat{x}_s$ and the local lateral estimated position $\hat{y}_s$, the Guidance function computes the required aimed heading to reach the following waypoint. Indeed, the usual Guidance schemes involve the use of a predefined trajectory and an online real-time comparison between the current estimated position of the ship and the initial aimed one. Then the heading error $\Delta\psi$ is elaborated, and the controller computes the aimed rudder angle δ to be applied by the ship in order to correct its trajectory and aim for the newt point of the trajectory (waypoint). Eventually, what will be measured by the sensors will be the total heading ψ_t which takes into account the contributions from the low-frequency ship control ψ_s and the ones from the usually high-frequency disturbances ψ_w. On the other hand, the positions of the ship will be returned by a GPS. Therefore, as previously mentioned, it is critical to correctly estimate the three state parameters of the ship in order for the Ship to be efficiently controlled.

1.2 The Navigation function and its components

In order for the ship to intelligently determine its current state, several algorithms must be used. It is possible to further decompose the previous Figure into the following one, by developing the Navigation block function in further sub-blocks:

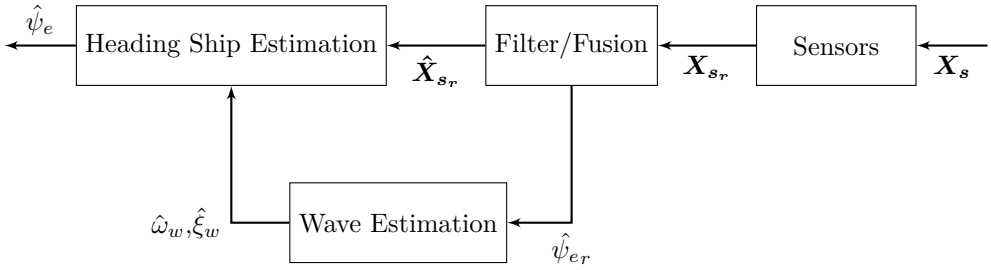

Figure 1.2: Navigation sub-function units.

All the sub-units will be described in the following sections. However, as a first general explanation, the Navigation function can be divided into four sub-functions. The simulated state vector $\boldsymbol{X_s}$ enters in the Navigation function and the sensor models return the raw state data $\hat{\boldsymbol{X}}_{\boldsymbol{s_r}}$. The raw data from the sensors are then filtered and fused through a special kind of algorithm. Eventually, thanks to a recursive online algorithm, an attempt to estimate the perturbations and some of their features is performed. Those parameters, as extensively described later in this document, are ones such as wave pulsation and damping, denoted respectively $\hat{\omega}_w$ and $\hat{\xi}_w$ on the previous Figure. Based on these estimates, one can retrieve an approximation of the actual heading $\hat{\psi}_e$ using an observer.

The important element here to clarify is that the sensors capture the parameter oscillations due to various perturbations. If one makes the assumption that the waves are causing most of the high-frequency perturbations, then one can reasonably expect to obtain and observe these contributions in the sensor measurements. The goal of the block Wave Estimation is to quantify and characterize the perturbations, in order for the observer to be able, in real-time, to filter efficiently the states for the Guidance and Control external functions.

Note that the goal of the project is to be able to perform path-following since the thrust power (velocity) of the ship is an input controlled by the user in real-time, manually. Therefore, no requirements were expected nor designed concerning potential velocities to be reached at certain waypoints on the trajectory. The algorithms must then be designed accordingly and be robust to velocity changes. In the next chapters and sections, the four blocks will be described in details.

Chapter 2

Navigation sub-function units description

In this chapter, a precise analysis of the embedded Navigation algorithms will be done, alongside with the major assumptions and simplifications. Each sub-functions of the previous Figure 1.2 will be detailed. As a reminder, these sub-functions are part of the higher level Navigation function observable on Figure 1.1. As previously explained, this function is critical since it is the eyes of the ship, allowing it to understand its position with a certain degree of accuracy. Therefore, the on-board computer calling frequency of this block must be much higher that the other ones, since there is a need of a lot of information on the states of the ship in order to take action.

2.1 Sensor models description

Various sets of sensor models were used throughout the simulator cases from 1 to 9 referenced in the GitHub of the project. In most of the cases, the use of Matlab/Simulink libraries was done, especially the *Navigation Toolbox*, allowing the use of the GPS and INS models as well as their initial tuning.

Some preliminary choice of hardware sensors has been done. The objective being to control the heading of the ship and ensure path-following, there is a need to measure the heading as well as the local GPS coordinates of the vessel. For that, the use of a magnetic compass coupled with a GPS generating GNSS measurements can be done. An additional use of an INS can be useful if one wants to have another source of attitude-yaw heading measurement.

Therefore, using the GPS and the magnetometer, the embedded software of the ship can estimate its position with respect to a reference as well as a heading angle measure. In terms of raw measurements $\boldsymbol{X_s}$ as depicted in Figure 1.2, the GPS will therefore provide $x_{s_{GPS_r}}$, $y_{s_{GPS_r}}$, $V_{s_{GPS_r}}$ and $\psi_{s_{GPS_r}}$, which are respectively the local longitudinal position of the ship (East), the local lateral position (North), the norm of its velocity vector and the heading of the ship, where r stands for raw and s for Ship. On the other hand, the magnetometer provides with a measure of the heading $\psi_{s_{MAG_r}}$.

In the next sub-sections, one will find some explanations on the adopted model for the GPS as well as for the magnetometer.

2.1.1 GPS sensor description and model

As previously explained, the GPS is used as a sensor allowing the ship to estimate its position, velocity and heading. The GPS normally returns positions estimates in the LLA frame. In this geographical coordinate frame, a point M is determined through its Latitude L, Longitude l and altitude h. For algorithm calculations, the use of a local horizontal frame will be done: the ENU frame. This corresponds to the local East, North, Up frame associated with a point on the surface of the Earth, and given with respect to a reference point on the Earth's surface.

Since the algorithms have been designed in order to deal with a local representation, there is a need to convert, in real-time, the LLA coordinates computed by the GPS to the ENU frame. To this end, it can

be said that the conversion needs to be first done from the LLA to the ECEF frame. Then, from the ECEF to the ENU frame. In the following Figure, one can find the three previously introduced reference systems:

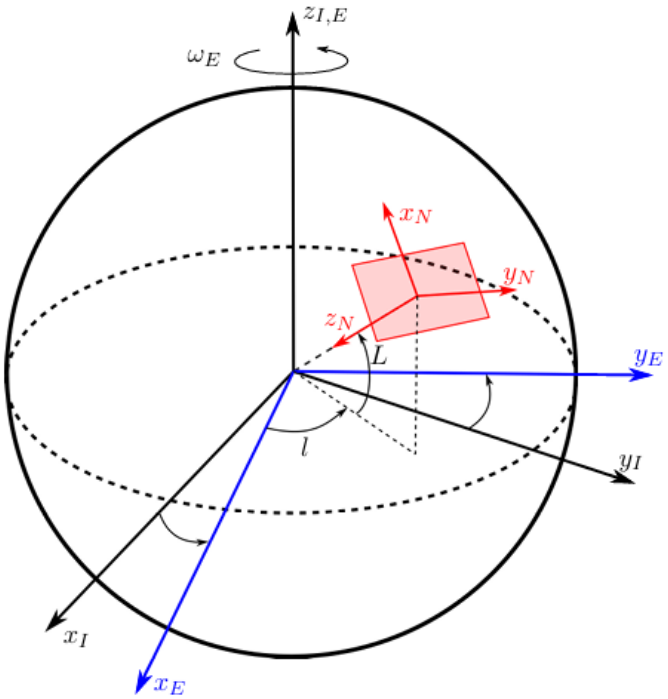

Figure 2.1: GPS frames [1].

In the previous Figure, the ECEF frame $(O; \boldsymbol{x_E}, \boldsymbol{y_E}, \boldsymbol{z_E})$ fixed with the Earth can be seen in blue. In red, a specific point on the Earth's surface can be located using the latitude and longitude angle, leading to the LLA frame. The local horizontal frame $(O; \boldsymbol{x_N}, \boldsymbol{y_N}, \boldsymbol{z_N})$ however, is the NED frame. NED stands for North, East and Down. Therefore, as in our project we will use the ENU frame, one simply needs to understand that $\boldsymbol{x_{ENU}} = \boldsymbol{y_{NED}} = \boldsymbol{y_N}$, $\boldsymbol{y_{ENU}} = \boldsymbol{x_{NED}} = \boldsymbol{x_N}$ and $\boldsymbol{z_{ENU}} = -\boldsymbol{z_{NED}} = -\boldsymbol{z_N}$. The last frame on the Figure in black $(O; \boldsymbol{x_I}, \boldsymbol{y_I}, \boldsymbol{z_I})$ is considered to be the inertial fixed one with respect to distant stars. The ECEF frame rotates at an angular velocity of ω_E around $\boldsymbol{z_{I,E}}$. This rotation models the Earth's revolution around itself.

As mentioned earlier, the first step of the conversion is to switch from the LLA to the ECEF frame. Assuming that the Earth is of perfect spherical shape, expressing the vector $\boldsymbol{OM} = R.\boldsymbol{z_{ENU}} = -R.\boldsymbol{z_N}$ (from the center of the Earth to the local horizontal point in the red $(O; \boldsymbol{x_N}, \boldsymbol{y_N}, \boldsymbol{z_N})$ frame) in the ECEF frame would lead to:

$$x_E = \boldsymbol{OM} \cdot \boldsymbol{x_E} = R.cos(L).cos(l) \tag{2.1}$$

$$y_E = \boldsymbol{OM} \cdot \boldsymbol{y_E} = R.cos(L).sin(l) \tag{2.2}$$

$$z_E = \boldsymbol{OM} \cdot \boldsymbol{z_E} = R.sin(L) \tag{2.3}$$

However, the GPS embedded model consider the Earth in a more realistic way, as being ellipsoidal. Therefore, an additional term can be found in the previous equations [2][3]:

$$x_E = (N(L) + R).cos(L).cos(l) \tag{2.4}$$

$$y_E = (N(L) + R).cos(L).sin(l) \tag{2.5}$$

$$z_E = ((1 - e^2).N(L) + R).sin(L) \tag{2.6}$$

The term $N(L)$ is called the Prime Vertical Radius of Curvature and depends on the latitude L. It can

be expressed in function of the Earth's semi-major axis a and eccentricity e as if it was considered to be an ellipsoid [4]:

$$N(L) = \frac{a}{\sqrt{1 - e^2.sin^2(L)}} \tag{2.7}$$

As it will be shown, using equations 2.4 to 2.6 will make it possible to convert the coordinates of a vector from the ECEF freame to LLA frame.

However, the subject of the reference point with which all the measurements will be compared with must be treated. As one specific point on the Earth can be located through three measures of latitude, longitude and altitude, the conversion to a local horizontal plane cannot be done directly. Since there is a change of representation from a spherical coordinate frame to a local cartesian horizontal one, there is a need to define a reference point in the local horizontal plane with respect to which all the measurement will be given. This can be done by firstly defining a chosen reference in the ECEF frame such as:

$$\boldsymbol{x_{ref}} = \begin{pmatrix} x_{ECEF_{ref}} \\ y_{ECEF_{ref}} \\ z_{ECEF_{ref}} \end{pmatrix}$$

Then, by supposing that the current position of the ship is known in the ECEF frame using the equations 2.4 to 2.6, the difference with respect to the reference point can be computed through:

$$\boldsymbol{x_{\Delta_{ECEF}}} = \boldsymbol{x_{s_{ECEF}}} - \boldsymbol{x_{ref}}$$

With $\boldsymbol{x_{s_{ECEF}}} = \begin{pmatrix} x_E & y_E & z_E \end{pmatrix}$ where s stands for Ship. Eventually, the switch from ECEF to ENU frame can be done using the general Figure 2.1 and its various projections:

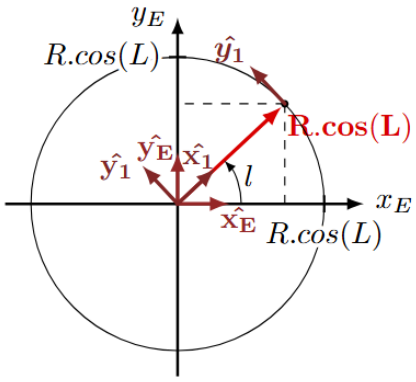

Figure 2.2: Frame ECEF to intermediate frame F1.

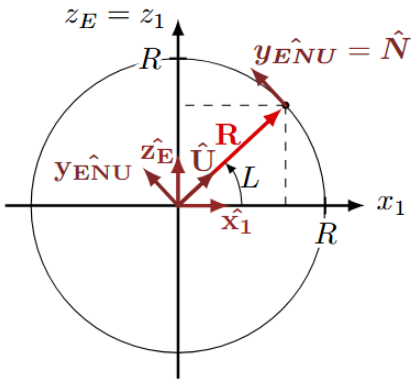

Figure 2.3: Intermediate frame F1 to ENU.

Therefore, using the unit vectors denoted with a special ˆ hat character, one can retrieve the following relations for the first frame change:

$$\hat{\boldsymbol{x}}_{\mathbf{1}} = cos(l).\hat{\boldsymbol{x}}_{\boldsymbol{E}} + sin(l).\hat{\boldsymbol{y}}_{\boldsymbol{E}} \tag{2.8}$$

$$\hat{\boldsymbol{y}}_{\mathbf{1}} = -sin(l).\hat{\boldsymbol{x}}_{\boldsymbol{E}} + cos(l).\hat{\boldsymbol{y}}_{\boldsymbol{E}} \tag{2.9}$$

$$\hat{\boldsymbol{z}}_{\mathbf{1}} = \hat{\boldsymbol{z}}_{\boldsymbol{E}} \tag{2.10}$$

According to the Figure 2.3, the second rotation can be expressed through the following equations:

$$\hat{\boldsymbol{U}} = \hat{\boldsymbol{z}}_{\boldsymbol{ENU}} = cos(L).\hat{\boldsymbol{x}}_{\mathbf{1}} + sin(L).\hat{\boldsymbol{z}}_{\mathbf{1}} \tag{2.11}$$

$$\hat{\boldsymbol{N}} = \hat{\boldsymbol{y}}_{\boldsymbol{ENU}} = -sin(L).\hat{\boldsymbol{x}}_{\mathbf{1}} + cos(L).\hat{\boldsymbol{z}}_{\mathbf{1}} \tag{2.12}$$

$$\hat{\boldsymbol{E}} = \hat{\boldsymbol{x}}_{\boldsymbol{ENU}} = \hat{\boldsymbol{y}}_{\mathbf{1}} \tag{2.13}$$

Hence, the base vectors of the ENU frame can be expressed with respect to the ones of the ECEF frame through:

$$\hat{\boldsymbol{U}} = \hat{\boldsymbol{z}}_{\boldsymbol{ENU}} = cos(L).cos(l).\hat{\boldsymbol{x}}_{\boldsymbol{E}} + cos(L).sin(l).\hat{\boldsymbol{y}}_{\boldsymbol{E}} + sin(L).\hat{\boldsymbol{z}}_{\boldsymbol{E}} \tag{2.14}$$

$$\hat{\boldsymbol{N}} = \hat{\boldsymbol{y}}_{\boldsymbol{ENU}} = -sin(L).cos(l).\hat{\boldsymbol{x}}_{\boldsymbol{E}} - sin(L).sin(l).\hat{\boldsymbol{y}}_{\boldsymbol{E}} + cos(L).\hat{\boldsymbol{z}}_{\boldsymbol{E}} \tag{2.15}$$

$$\hat{\boldsymbol{E}} = \hat{\boldsymbol{x}}_{\boldsymbol{ENU}} = \hat{\boldsymbol{y}}_{\mathbf{1}} = -sin(l).\hat{\boldsymbol{x}}_{\boldsymbol{E}} + cos(l).\hat{\boldsymbol{y}}_{\boldsymbol{E}} \tag{2.16}$$

Or more compactly:

$$\begin{pmatrix} \hat{\boldsymbol{E}} \\ \hat{\boldsymbol{N}} \\ \hat{\boldsymbol{U}} \end{pmatrix} = \begin{pmatrix} -sin(l) & cos(l) & 0 \\ -sin(L).cos(l) & -sin(L).sin(l) & cos(L) \\ cos(L).cos(l) & cos(L).sin(l) & sin(L) \end{pmatrix} . \begin{pmatrix} \hat{\boldsymbol{x}}_{\boldsymbol{E}} \\ \hat{\boldsymbol{y}}_{\boldsymbol{E}} \\ \hat{\boldsymbol{z}}_{\boldsymbol{E}} \end{pmatrix} \tag{2.17}$$

This algorithm has been created in the project and can be found via this following link: conversion from LLA to ENU.

2.1.2 Implementation in the Simulator

As explained in the previous sections, the GPS provides with the coordinates in the LLA frame. Hence, as the usual coordinates for local navigation on the Earth's surface deals with ENU frame representation, this transformation has to be done. This in fact will be the case for the real actual final embedded software. In our actual simulators, no conversion is done in real time. Indeed, the path to be followed is generated based on a specified trajectory by the user. See this script for the generation of a specific trajectory: waypoints trajectory generation. Here, the trajectory is generated on a small salted lake near the Mediterranean Sea in the very south of France and close to the Spanish border, such as on the following Figure:

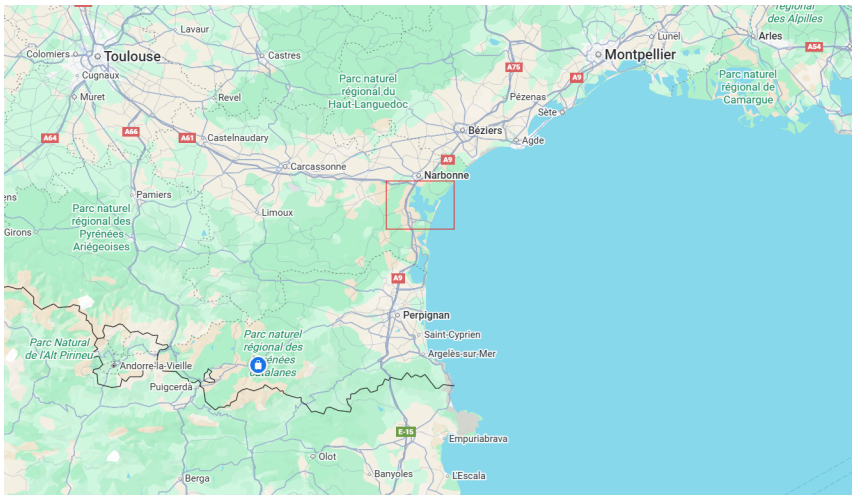

Figure 2.4: Zone used for Trajectory plot (red square).

The use of the Matlab library *geoplot* has been done for the plotting of the trajectory, resulting in a global upper view without details:

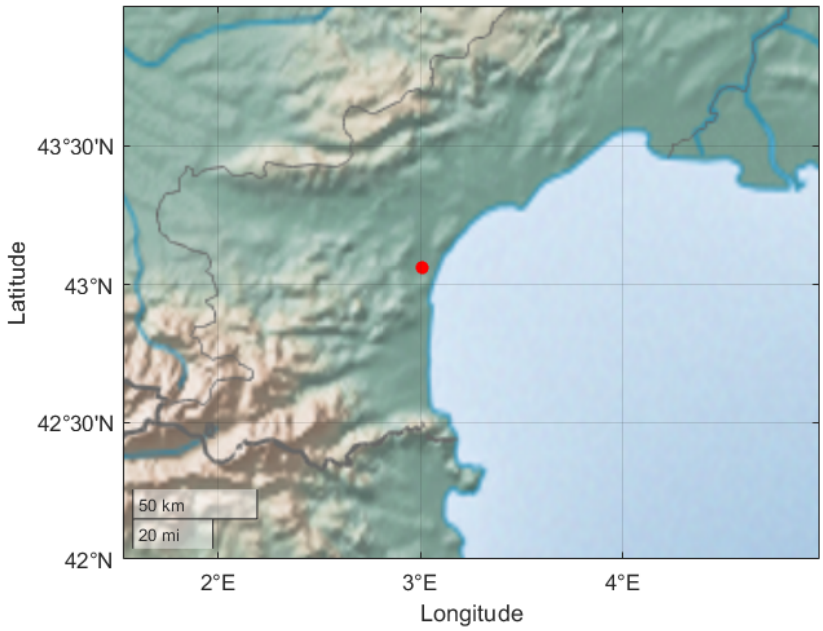

Figure 2.5: Generated trajectory (without local blue background).

Due to the lack of precision in the *geoplot* toolbox for the position of various environments; such as the small salted lakes, the background is not blue at the generated trajectory location. By zooming and supposing that there is blue sea background, the following trajectory is computed:

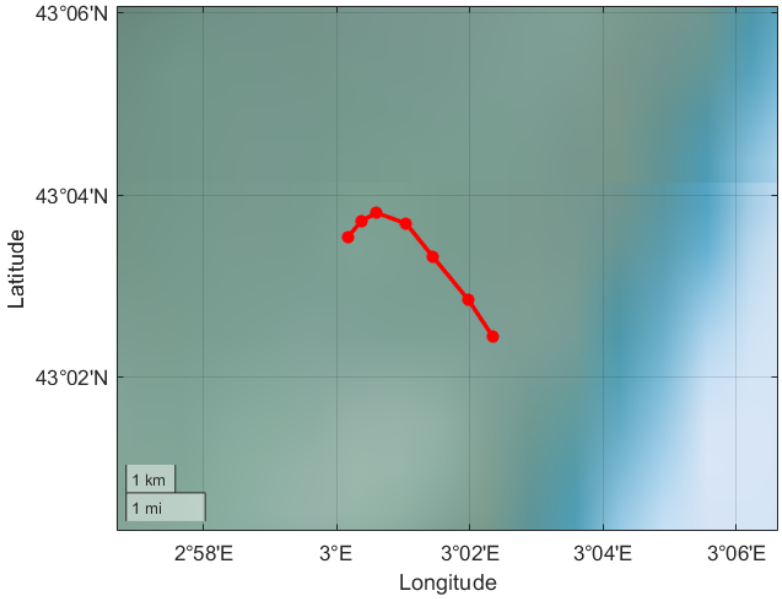

Figure 2.6: Generated trajectory zoomed on (without local blue background).

The conversion from LLA to ENU is done by taking a reference initial point. This point is expressed in LLA such as being:

$$\boldsymbol{x_{ref}} = \begin{pmatrix} x_{LLA_{ref}} \\ y_{LLA_{ref}} \\ z_{LLA_{ref}} \end{pmatrix} = \begin{pmatrix} 43.059023 \\ 3.002915 \\ 0 \end{pmatrix}$$

This point corresponds to a nautical base named Port-Mahon, which can be found on the following Figure:

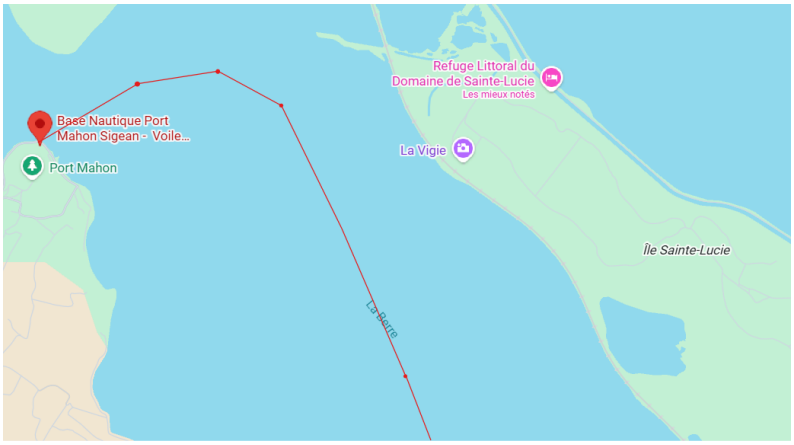

Figure 2.7: Generated trajectory zoomed on (with local blue background).

To close this illustrative aside, the LLA reference point will then be corresponding to the $(0, 0, 0)$ reference point in the ENU local frame. Since we have adopted a very simple model for the GPS sensor, it is sufficient to just perturb the propagated dynamics of the ship in order to simulate the GPS measurements. Indeed, the dynamics of the ship, based on the ENU expression of the velocity and as it is going to be demonstrated later, can be represented by the two following equations:

$$\dot{x}_s(t) = V_s(t) \cdot \sin(\psi_s(t)) \tag{2.18}$$

$$\dot{y}_s(t) = V(t) \cdot \cos(\psi_s(t)) \tag{2.19}$$

The equations and their origin have been described in the Simulator justification file and more precisely in its last section: simulator justification file. According to our introductory description, these are the longitudinal and lateral velocity states of the ship captured in the $\boldsymbol{X_s}$ vector in output of the ship model block of Figure 1.1. It is straightforward that integrating these equations will result in having access to the position of the ship. That is precisely what is being done in the kinematics propagation block of Simulator Case 9 accessible through this link: Simulator Case 9. From the high level point of view, one must access to the following blocks: **Dynamics-Kinematics-Ship-Propagation**, then **Kinematics-Ship-Propagation**. This block is as follows:

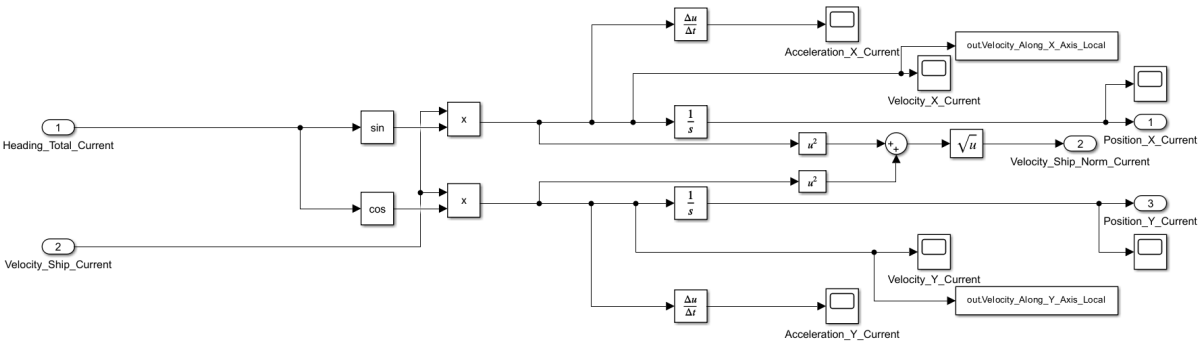

Figure 2.8: Details of the kinematics propagation for the ship.

Therefore, depending on the release number of the simulator and the algorithms, the following sensor blocks that emulate the states of the ship can be relatively simple or complex. For example, the GPS sensor block inside the block Sensor of Figure 1.2 (which in inside the Navigation block of Figure 1.1) is as follows for the actual most up-to-date release number 0.0.4 Simulator Case 9:

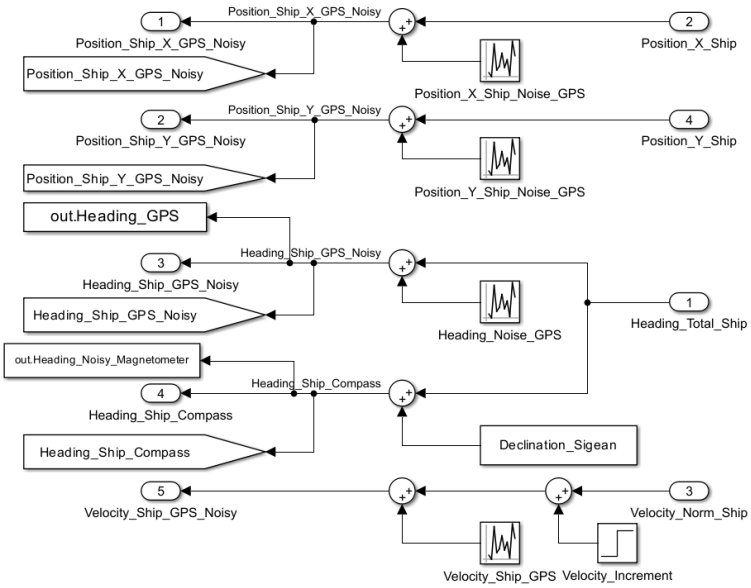

Figure 2.9: Simplified sensor models (GPS and Magnetometer).

Therefore, only a random walk noise perturbation of a certain specified amplitude is added to the simulated positions of the ship. Hence, no conversion from LLA to ENU needs to be done in that case. In previous versions however, such as Simulator Case 8 that can be seen in the release number 0.0.3, the Sensors block was much more complex. The use of Matlab/Simulink toolboxes to simulate the sensors was done and even the differences in frequencies sampling and quantization were present in the simulator:

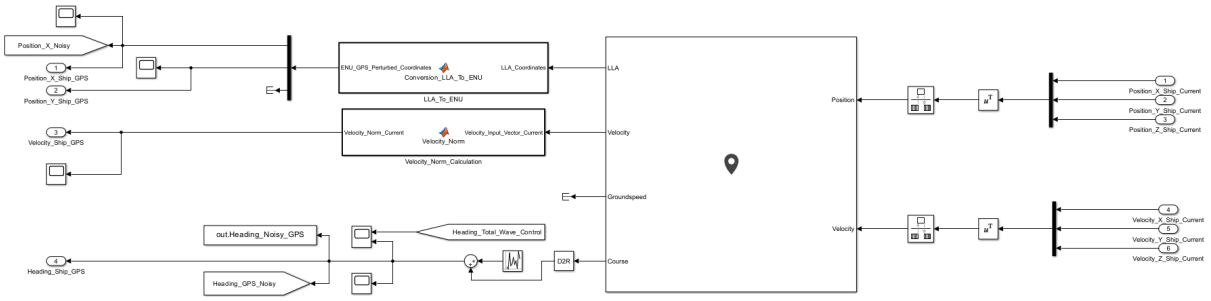

Figure 2.10: Developed model of the GPS.

On the previous Figure, one can see the embedded Matlab Function LLA to ENU as described in the previous subsection. The reason for this simplification from the simulator number 8 to 9 is due to the goal of finding the source of a divergence problem spotted in the Simulator Case 8. It was wrongly identified that this source could potentially be the sensor models. In fact, the origin of this divergence, as already discussed in a lesser way in the introduction, was the too rudimentary Guidance algorithm. Hence, in the next releases, the more developed and complex model of the sensors will be added.

This long explanation had the goal of illustrating why the use of some algorithms is done in some release or not. Finally, there is no more work done on the model of the GPS sensor apart from the random-walk noise or the Matlab/Simulink GPS block from the *Navigation Toolbox*. Note that the selected GPS module for the project is of increased quality: the BN-880 Beitian, allowing to return a norm of the velocity and a heading measurement, which seems to not be the case of all the GPS.

2.1.3 Magnetometer sensor description

Concerning the magnetometer, this is a device allows to measure the Earth's magnetic North direction from the current point of measure. In terms of semantics, the true north is the Earth's north direction aligned with its rotation axis. The magnetic north returned by magnetometer devices is the one aligned with the north direction of the magnetic field generated by the Earth.

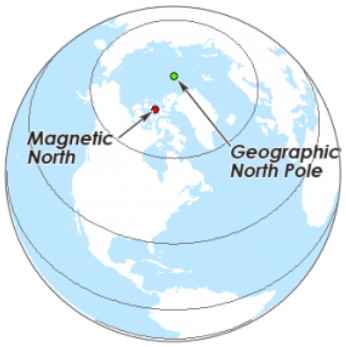

Figure 2.11: Magnetic and true North.

From a specific location on Earth, the difference between these two values is the declination angle. The value of the declination at the location where the trajectory has been computed is of around:

$$D(\boldsymbol{r}) = 2.116^{\circ}$$

This can either be determined analytically, or using an online declination estimator such as the magnetic field calculator of the NOAA official website. In a practical way, it means that the true north for Navigation is:

$$N_T = D(\boldsymbol{r}) + N_M$$

Where N_T stands for the true north, $D(\boldsymbol{r})$ the declination at the current location $\boldsymbol{r}$ and N_M the magnetic north. As pointed out in the documentation of the sensor [5], the magnetic field acting on a point on the Earth's surface can be seen as a vector as it follows:

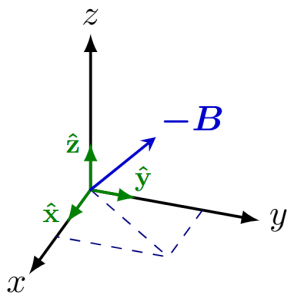

Figure 2.12: Magnetic field as detected by the magnetometer.

Therefore, only the projection of the magnetic field vector on the (x, y)-plane is needed to compute the direction to the Magnetic North. By denoting the projection of the magnetic vector of the Earth B on the (x, y)-plane by B_p, and ψ_m the magnetic heading angle between B_p and the x-axis of the magnetometer, one gets that:

$$\psi_m = atan2(y, x) \tag{2.20}$$

Hence, in order to get the heading associated with Navigation, it is possible to compute it by using:

$$\psi_t = \psi_m + D(\boldsymbol{r}) = atan2(y, x) + D(\boldsymbol{r}) \tag{2.21}$$

With x and y the magnetic measurements provided by the magnetometer in the local magnetometer frame. One can find the implemented algorithm in C++ (Arduino) for future Software implementation through this algorithm.

2.1.4 Implementation in the simulator

In terms of implementation in the project, as no Toolboxes seem to possess such sensor already coded, a very simplistic model as the one of the GPS in the release number 0.0.4 has been chosen and designed. By observing Figure 2.9 and zooming on it, the adopted model is based on the total heading angle from the dynamics of the ship inside the vector $\boldsymbol{X_s}$:

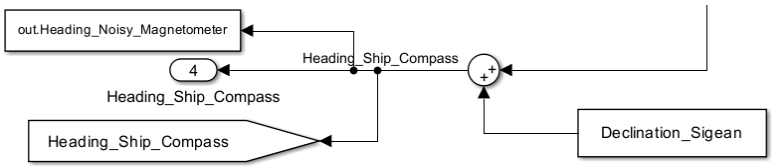

Figure 2.13: Simplified Magnetometer model.

Thus, it can be deduced that the very important step of calibration of the sensor has been avoided here. Also, the uncertainty model of the sensor is very simplified because it only consists in a constant declination value. In further releases, a focus on the sensor models will be done to improve them based on the evaluation of the hardware. In these initial releases, no further development on the sensors have been done since no evaluations have been done on any actual hardware components yet. However, it has been considered that by taking sufficient margins in the synthesis of the controller, in the error of the sensor models and by designing a realistic Guidance algorithm, the uncertainty on the sensor model accuracy could be over-passed and corrected later if needed.

2.2 Filtering and Fusion of the measurements

The measurements provided by the sensors are noisy. Several sources of noise can be identified. The first one is inherent to the sensor design and usually indicated and specified in its specification sheets. It can take the form of a specified variance error for each of the computed states provided by the sensor. In the case of ship control, other contributions are the ones due to wind, current and waves. In the frame of our project, the perturbation due to the waves on the vessel will be the only one for the moment.

NOTA - The GN&C system designed here is for a small fishing boat used to make some lake trip for pleasure and fishing. The lake on which the ship navigates is a salted lake/pond close to the Mediterranean Sea. Therefore it is not very deep and in function of the wind, the waves can be inexistent, so does the currents. However, the area is subject to a very powerful wind called Tramontane [6]. When this wind is active, very important surface waves, compared to the ship size, can be generated. Hence, there is a need of designing a robust control system allowing to pilot the ship in the case of a trip under windy conditions.

In this project, the first step was to filter the noise from sensors and fuse the data given by the Sensors block. As described in the previous section, the GPS and the magnetometer return the same heading angle. One of the solution in order to keep just one measurement to be sent to the Guidance and Control functions is to fuse the measurements coming from these two sensors, while suppressing the sensor noise from it. This will result in a single fused and cleansed heading value.

A frequently used solution in the industry for this kind of problems is the Kalman Filter. Kalman Filters are for linear systems, whereas Extended Kalman Filter are for non-linear systems. In this project, several Extended Kalman Filters have been designed for several explored cases in terms of sensor used, adopted models and explored mathematical programming. They are all accessible through the following repository: Extended Kalman Filters and Estimators. This is the goal of the Filter/Fusion block observable on the Figure 1.2.

In the next sections, the theory behind Kalman Filters and Extended Kalman Filters will be explained, since both of them will be used as Navigation sub-functions of this project. Then, since a model of the wave will be used to predict and estimate the wave contribution, a presentation of wave theory restrained to GN&C applications will be done. Moreover, the chosen prediction model for the EKF and KF will be described, as well as the associated covariance matrices. Eventually, the theory behind Regularized Extended Least Squares (RELS) algorithm will be given, since an online ARMA-RELS based algorithm is used in order to estimate the parameters of the waves.

2.2.1 Kalman Filters for linear systems

In this section, the details on the determination of the equations of the Kalman Filter will be given. All the reasoning is based on [7]. However, the demonstrations in this document have been voluntarily brief, or even avoided in some cases. For the sake of curiosity and understanding, the details of the demonstration will be proposed in this section.

Such filters iterates between two steps. The first one is the **prediction step**. During this part of the algorithm, a prediction of the current states is done over a certain time window based on a dynamical embedded model **and** the previous estimate of the states. The second step is the **measurement update**. During this part, an estimation of the current state is done using the prediction model and the newly available set of measurements from the sensors.

Prediction step: the objective is to estimate the state vector at the current step time $n \in \mathbb{N}$ denoted $\boldsymbol{x_n} = \boldsymbol{x(t_n)}$. This estimation has to be done considering the previous set of measurement data: $\boldsymbol{y_{1:n-1}} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, ...\boldsymbol{y_{n-1}}\}$ as well as the estimate of the state at the previous step time $n-1$ denoted $\boldsymbol{x_{n-1}}$. During this step, the estimation of the state at t_n given the data up to t_{n-1}, denoted $\boldsymbol{\hat{x}_{n|n-1}}$, which can be seen as the mean of the prediction, will be used. Note that the covariance of the predicted state will be denoted $P_{n|n-1}$.

Measurement step: the goal is to combine the newly obtained measurement values $\boldsymbol{y_n}$ from the sensors with the use of the previous Prediction step to estimate the current state vector $\boldsymbol{x_n}$. This time, the mean of the updated state estimate at this current step time will be denoted $\boldsymbol{\hat{x}_{n|n}}$. Note that the covariance

of the measured state will be $P_{n|n}$.

As already presented, the goal here is to filter a continuous linear system of the shape:

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{A}.\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{B}.\boldsymbol{w}(\boldsymbol{t}) \tag{2.22}$$

$$\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{G}.\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{r}(\boldsymbol{t}) \tag{2.23}$$

Here, $\boldsymbol{x}(\boldsymbol{t}) \in \mathbb{R}^{n,1}$ denotes the state vector of the process of size n-by-1. $\boldsymbol{w}(\boldsymbol{t}) \in \mathbb{R}^{m,1}$ is the driving stochastic process of size m-by-1. $\boldsymbol{y}(\boldsymbol{t}) \in \mathbb{R}^{p,1}$ is the measurement vector of size p-by-1 and $\boldsymbol{r}(\boldsymbol{t}) \in \mathbb{R}^{p,1}$ is the sensor noise vector of size p-by-1.

In function of the discretization method, fully described in the next sections, it can be shown [7] that the discrete-time system equivalent to the previous continuous one is the following:

$$\boldsymbol{x_n} = F_n.\boldsymbol{x_{n-1}} + \boldsymbol{q_n} \tag{2.24}$$

$$\boldsymbol{y_n} = G_n.\boldsymbol{x_n} + \boldsymbol{r_n} \tag{2.25}$$

NOTA - Usually, the state-space equation of the process model in the literature for the KF is usually of the form $\dot{\boldsymbol{x}}(\boldsymbol{t}) = A.\boldsymbol{x}(\boldsymbol{t}) + B.\boldsymbol{u}(\boldsymbol{t}) + E.\boldsymbol{w}(\boldsymbol{t})$ hence, with a term corresponding to the input vector $\boldsymbol{u}(\boldsymbol{t})$ of the model. This is the equation that will be used in the project. When on discrete time, this would be: $\boldsymbol{x_n} = F_n.\boldsymbol{x_{n-1}} + G_n.\boldsymbol{u_{n-1}} + \boldsymbol{w_{n-1}}$. However, for the sake of simplification of the demonstration of the KF equations, the simplified model of equation 2.22 has been kept.

We will make the assumption that the noise in the state-space model equation 2.24 is of zero mean $\mathbb{E}[\boldsymbol{q_n}] = 0$, as well as the one for the measurement model of equation 2.25, $\mathbb{E}[\boldsymbol{r_n}] = 0$. Their covariance matrices are supposed to be known, and their determination will be detailed in the dedicated section. They will be denoted as $cov(\boldsymbol{q_n}) = \boldsymbol{Q_n}$ for the state-space process model and $cov(\boldsymbol{r_n}) = \boldsymbol{R_n}$ for the measurement model. Additionally, we will note that the mean of the first initial state is supposedly known to be $\mathbb{E}[\boldsymbol{x_0}] = m_0$ along with an initial associated covariance value $cov(\boldsymbol{x_0}) = P_0$.

2.2.1.1 Measurement update step equations determination

In order to determine the equations of the measurement update part of the KF, we have to make the assumption that there is a prediction of the mean of the state $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$ and its covariance $P_{n|n-1}$ currently known. This is the prior, called apriori knowledge of the state at the current step time t_n. Then, the measurement provides the actual noisy information about the true state.
The goal is to minimize the measurement error and noise while taking into consideration the apriori state estimation from the prediction step into account. It seems that a well suited cost candidate function for this optimization problem is the one of Regularized Least Squares. Before giving its mathematical expression, let us recall the basics from the least squares and the minimization of a cost function:

Cost function minimization and least squares methods

The KF algorithm uses and fuses all the available measurements by employing an embedded model to estimate the states in an optimal way with respect to a cost function. In a broad and general way, the goal is to minimize the sensor measurement noise. Without loss of generality, we can state that the general scalar measurement model is as follows:

$$y_n = g_n(\boldsymbol{x}) + r_n \tag{2.26}$$

With the measurement function $g_n(x)$, linear in equation 2.25 and equal to $\boldsymbol{g_n(x)} = G_n.\boldsymbol{x_n}$ when using the vectorial form, presented in the further development. Therefore, in order to minimize the sensor noise r_n, one shall minimize $y_n - g_n(x)$. This quantity will be the error at the current time step t_n:

$$e_n = y_n - g_n(\boldsymbol{x}) \tag{2.27}$$

Usually, there can be two typical functions. The first one is an absolute error function, which impacts all error in an equal way such that $|e_n| = |y_n - g_n(\boldsymbol{x})|$. The second one is a quadratic cost function, which will impact more the larger errors and that is of shape: $e_n^2 = (y_n - g_n(\boldsymbol{x}))^2$.

The goal of the estimation algorithm is in fact to minimize the error over the entire set of measurements and not only one. That is why the cost function must be a sum of all the errors such that:

$$J_{LS}(\boldsymbol{x}) = \sum_{n=1}^{N} e_n^2 = \sum_{n=1}^{N} (y_n - g_n(\boldsymbol{x}))^2 \tag{2.28}$$

With N the number of measurements up until the current step time and where LS stands for Least Squares. If the measurement is vectorial, then $\boldsymbol{e_n} = \boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x})$ and $\boldsymbol{e_n}^2 = (\boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x}))^2 = (\boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x}))^T.(\boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x}))$. In that case, the cost function becomes:

$$J_{LS}(\boldsymbol{x}) = \sum_{n=1}^{N} e_n^2 = \sum_{n=1}^{N} (\boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x}))^T.(\boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x})) \tag{2.29}$$

If the vectors are assumed to be of the form:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y_1} & \boldsymbol{y_2} & \dots & \boldsymbol{y_N} \end{pmatrix}^T$$

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{g_1}(\boldsymbol{x}) & \boldsymbol{g_2}(\boldsymbol{x}) & \dots & \boldsymbol{g_N}(\boldsymbol{x}) \end{pmatrix}^T$$

$$\boldsymbol{r} = \begin{pmatrix} \boldsymbol{r_1} & \boldsymbol{r_2} & \dots & \boldsymbol{r_N} \end{pmatrix}^T$$

Then the cost function equation 2.29 can be developed and simplified as follows:

$$J_{LS}(\boldsymbol{x}) = (\boldsymbol{y_1} - \boldsymbol{g_1}(\boldsymbol{x})).^T(\boldsymbol{y_1} - \boldsymbol{g_1}(\boldsymbol{x})) + (\boldsymbol{y_2} - \boldsymbol{g_2}(\boldsymbol{x})).^T(\boldsymbol{y_2} - \boldsymbol{g_2}(\boldsymbol{x})) + \dots + (\boldsymbol{y_N} - \boldsymbol{g_N}(\boldsymbol{x})).^T(\boldsymbol{y_N} - \boldsymbol{g_N}(\boldsymbol{x}))$$

$$\begin{aligned} J_{LS}(\boldsymbol{x}) = (\boldsymbol{y_1}^T.\boldsymbol{y_1} + \boldsymbol{y_2}^T.\boldsymbol{y_2} + \dots + \boldsymbol{y_N}^T.\boldsymbol{y_N}) - (\boldsymbol{y_1}^T.\boldsymbol{g_1}(\boldsymbol{x}) + \boldsymbol{y_2}^T.\boldsymbol{g_2}(\boldsymbol{x}) + \dots + \boldsymbol{y_N}^T.\boldsymbol{g_N}(\boldsymbol{x})) - \\ (\boldsymbol{g_1}(\boldsymbol{x})^T.\boldsymbol{y_1} + \boldsymbol{g_2}(\boldsymbol{x})^T.\boldsymbol{y_2} + \dots + \boldsymbol{g_N}(\boldsymbol{x})^T.\boldsymbol{y_N}) + (\boldsymbol{g_1}(\boldsymbol{x})^T.\boldsymbol{g_1}(\boldsymbol{x}) + \\ \boldsymbol{g_2}(\boldsymbol{x})^T.\boldsymbol{g_2}(\boldsymbol{x}) + \dots + \boldsymbol{g_N}(\boldsymbol{x})^T.\boldsymbol{g_N}(\boldsymbol{x})) \end{aligned}$$

$$\begin{aligned} J_{LS}(\boldsymbol{x}) &= \boldsymbol{y}^T.\boldsymbol{y} - \boldsymbol{y}^T.\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})^T.\boldsymbol{y} + \boldsymbol{g}(\boldsymbol{x})^T.\boldsymbol{g}(\boldsymbol{x}) \\ J_{LS}(\boldsymbol{x}) &= (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x}))^T.\boldsymbol{y} - (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x}))^T.\boldsymbol{g}(\boldsymbol{x}) \\ J_{LS}(\boldsymbol{x}) &= (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x}))^T.(\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x})) \end{aligned} \tag{2.30}$$

Moreover, it might be the case where not all the measurements transport the same amount of information. Indeed, some measurements might be more important than other and then shall contribute more the the solution than the other ones. Therefore: one can define the Weighted Least Square cost function such as:

$$\forall n \in [\![1, n]\!], \exists\ w_n > 0 : J_{WLS}(\boldsymbol{x}) = \sum_{n=1}^{N} w_n.(y_n - g_n(\boldsymbol{x}))^2$$

With the scalar weight w_n for the measurement n. As in the previous development, one can find the compact form of this criterion if subject to various vectorial measurements such as in 2.29:

$$J_{WLS}(\boldsymbol{x}) = \sum_{n=1}^{N} (\boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x}))^T.W_n.(\boldsymbol{y_n} - \boldsymbol{g_n}(\boldsymbol{x})) \tag{2.31}$$

With W_n a positive definite matrix of the weights such that for $\boldsymbol{x} \neq \boldsymbol{0}$, then $\boldsymbol{x}^T.W_n.\boldsymbol{x} > \boldsymbol{0}$.
As for the classical Least Square cost function equation 2.30, one can retrieve the compact form. The result is directly given to be:

$$J_{WLS}(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x}))^T.\boldsymbol{W}.(\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x})) \tag{2.32}$$

Note that the scalar weighting matrix and the vectorial one are:

$$W = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ . & . & \dots & . \\ 0 & 0 & \dots & w_N \end{pmatrix}$$

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W_1} & 0 & ... & 0 \\ 0 & \boldsymbol{W_2} & ... & 0 \\ . & . & ... & . \\ 0 & 0 & ... & \boldsymbol{W_N} \end{pmatrix}$$

The choice of the weights will not be discussed here since our goal is to derive the Kalman Filter equations. However, one can note that a good practice for the choice of the weights is to take them as the inverse of the covariance of the current measurement noise. Indeed, in that case, if the uncertainty of this current measurement is high, so is its covariance. Hence, the weight will be small and this highly uncertain measurement will therefore not been took into consideration.

In some cases, it can be useful to regularize the estimate by forcing it, for example, to be small or to be in some predefined part of the space. It seems that its utility is whenever the estimate is not unique. Indeed, in that case, a regularization term can be used to select the estimate that corresponds to the area where we believe that the estimate should be. This is the prior belief. The compact form of the cost function is given by:

$$J_{ReLS}(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{g(x)})^T.\boldsymbol{R^{-1}}.(\boldsymbol{y} - \boldsymbol{g(x)}) + (\boldsymbol{x} - \boldsymbol{m})^T.\boldsymbol{P^{-1}}.(\boldsymbol{x} - \boldsymbol{m}) \tag{2.33}$$

With the quadratic regularization term defined through the parameters $\boldsymbol{m}$ and $\boldsymbol{P}$. Here, one can see that the selected weighting term $\boldsymbol{W}$ is $\boldsymbol{R^{-1}}$. Indeed, this this is the case because here, the regularization term can be interpreted as a Gaussian distribution with mean $\boldsymbol{m}$ and covariance $\boldsymbol{P}$ [7].

As previously explained before the aside on the least square methods, the Regularized Least Square criterion is the one chosen as the optimization objective cost function here. As a consequence, the estimate of the state during the measurement update step is obtained by solving the following Regularized Least Square problem:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \underset{\boldsymbol{x_n} \in \boldsymbol{S}}{\arg\min}\, J_{ReLS}(\boldsymbol{x}) = \{\boldsymbol{x_n} \in S \mid J_{ReLS}(\boldsymbol{s}) \geq J_{ReLS}(\boldsymbol{x_n}),\ \forall\ s \in S\} \tag{2.34}$$

The cost function is analytical and linear, so it is possible to derive a closed-form solution for this optimization problem based on the relation 2.33. In order to find this form, one shall solve the following equation for $\boldsymbol{x}$:

$$\frac{\partial J_{ReLS}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{0}$$

This problem can be broken down:

$$J_{ReLS}(\boldsymbol{x}) = J_1(\boldsymbol{x}) + J_2(\boldsymbol{x})$$
$$J_1(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{g(x)})^T.\boldsymbol{R^{-1}}.(\boldsymbol{y} - \boldsymbol{g(x)})$$
$$J_2(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{m})^T.\boldsymbol{P^{-1}}.(\boldsymbol{x} - \boldsymbol{m})$$

In the frame of the Kalman Filter, we previously stated that it concerns a linear measurement system such as the one of equation 2.25. In this special case: $\boldsymbol{g(x)} = \boldsymbol{G}.\boldsymbol{x}$. Hence, $J_1(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{G}.\boldsymbol{x})^T.\boldsymbol{R^{-1}}.(\boldsymbol{y} - \boldsymbol{G}.\boldsymbol{x})$
Let us develop $J_1(\boldsymbol{x})$:

$$J_1(\boldsymbol{x}) = (\boldsymbol{y}^T - \boldsymbol{x}^T.\boldsymbol{G}^T).(\boldsymbol{R^{-1}}.\boldsymbol{y} - \boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x})$$
$$J_1(\boldsymbol{x}) = \boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} - \boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} - \boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + \boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x}$$

It can be shown that the middle terms involving the transpose are equals:

$$\boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} = (\boldsymbol{R^{-1}}.\boldsymbol{y})^T.(\boldsymbol{x}^T.\boldsymbol{G}^T)^T = (\boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y})^T$$

$$\boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} = (\boldsymbol{G}.\boldsymbol{x})^T.(\boldsymbol{y}^T.\boldsymbol{R^{-1}})^T = (\boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x})^T$$

By denoting that the $\boldsymbol{R}$ matrix is symmetric, hence the transpose of $\boldsymbol{R^{-1}}$ is $\boldsymbol{R^{-1}}$. On the other and, it is possible to show that $\boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x}$ and $\boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y}$ are in fact scalar quantities. Indeed, $\boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} \in \mathbb{R}^{1\times m}\times\mathbb{R}^{m\times m}\times\mathbb{R}^{m\times n}\times\mathbb{R}^{n\times 1}$ hence $\boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} \in \mathbb{R}^1$. If s is a scalar, then it is known that $s = s^T$. Therefore:

$$\boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} = \boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y}$$

Hence:

$$J_1(\boldsymbol{x}) = \boldsymbol{y}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} - 2.\boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + \boldsymbol{x}^T.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} \tag{2.35}$$

Consequently, the derivative of this first part of the cost function with respect to the state vector is [8]:

$$\frac{\partial J_1(\boldsymbol{x})}{\partial \boldsymbol{x}} = 0 - 2.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + (\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G} + (\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G})^T).\boldsymbol{x}$$

Due to the symmetry of the term $\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G}$, then $\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G} = (\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G})^T$, leading to:

$$\frac{\partial J_1(\boldsymbol{x})}{\partial \boldsymbol{x}} = -2.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + 2.\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} \tag{2.36}$$

The exact same mathematical development for $J_2(\boldsymbol{x})$ function leads to:

$$\frac{\partial J_2(\boldsymbol{x})}{\partial \boldsymbol{x}} = 2.\boldsymbol{P^{-1}}.\boldsymbol{x} - 2.\boldsymbol{P^{-1}}.\boldsymbol{m} \tag{2.37}$$

Therefore:

$$\frac{\partial J_{ReLS}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial J_1(\boldsymbol{x})}{\partial \boldsymbol{x}} + \frac{\partial J_2(\boldsymbol{x})}{\partial \boldsymbol{x}} = -2.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + 2.\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} - 2.\boldsymbol{P^{-1}}.\boldsymbol{m} + 2.\boldsymbol{P^{-1}}.\boldsymbol{x} \tag{2.38}$$

Thus, the equation to be solved exposed before this aside leads to the following:

$$-2.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + 2.\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G}.\boldsymbol{x} - 2.\boldsymbol{P^{-1}}.\boldsymbol{m} + 2.\boldsymbol{P^{-1}}.\boldsymbol{x} = \boldsymbol{0}$$

Leading to:

$$(2.\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G} + 2.\boldsymbol{P^{-1}}).\boldsymbol{x} = 2.\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + 2.\boldsymbol{P^{-1}}.\boldsymbol{m}$$

$$\boldsymbol{x} = \hat{\boldsymbol{x}}_{\boldsymbol{ReLS}} = (\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G} + \boldsymbol{P^{-1}})^{-1}.(\boldsymbol{G}^T.\boldsymbol{R^{-1}}.\boldsymbol{y} + \boldsymbol{P^{-1}}.\boldsymbol{m}) \tag{2.39}$$

In consequence, this estimator can be understood such as being a weighted average of the data. The reader can refer to [7] for the demonstration of the covariance of the estimator. It can be shown that for the Regularized Least Square cost function:

$$cov(\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}}) = (\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G} + \boldsymbol{P^{-1}})^{-1} \tag{2.40}$$

Nonetheless, it is possible to show that this estimator can be expressed in an alternative but equivalent form, especially useful for dynamical sequential least squares and Kalman Filtering. For that, one must define the Woodbury matrix identity such that for given matrices $\boldsymbol{A}$, $\boldsymbol{U}$, $\boldsymbol{C}$ and $\boldsymbol{V}$:

$$(\boldsymbol{A} + \boldsymbol{U}.\boldsymbol{C}.\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}.\boldsymbol{U}.(\boldsymbol{C}^{-1} + \boldsymbol{V}.\boldsymbol{A}^{-1}.\boldsymbol{U})^{-1}.\boldsymbol{V}.\boldsymbol{A}^{-1} \tag{2.41}$$

Then, one can notice that:

$$(\boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G} + \boldsymbol{P^{-1}})^{-1} = (\boldsymbol{P^{-1}} + \boldsymbol{G^T}.\boldsymbol{R^{-1}}.\boldsymbol{G})^{-1} = \boldsymbol{P} - \boldsymbol{P}.\boldsymbol{G}^T.(\boldsymbol{R} + \boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T)^{-1}.\boldsymbol{G}.\boldsymbol{P}$$

Let the gain $\boldsymbol{K}$ to be cumputed using:

$$\boldsymbol{K} = \boldsymbol{P}.\boldsymbol{G}^T.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T + \boldsymbol{R})^{-1} \tag{2.42}$$

In fact, this gain is not considered or fixed in an intuitive way but has been derived using an optimality condition here [9]. Indeed, with a little anticipation on the established closed-form

solution equation 2.44, one can note that the goal is to find the best estimator possible. The best, in the sense of a minimization of the mean quadratic error of the state estimation. Therefore, the problem consists in finding the optimal gain $\boldsymbol{K_n}$ that minimizes $\boldsymbol{P_{n|n}}$ from equation 2.47, again, from anticipation. However, equation 2.47 of $\boldsymbol{P_{n|n}}$ has several form. For example, it is equivalent to the Joseph form, the one that will be used to derive the expression of the gain $\boldsymbol{K_n}$. Indeed:

$$\boldsymbol{P_{n|n}} = (\boldsymbol{I} - \boldsymbol{K_n}.\boldsymbol{G_n}).\boldsymbol{P_{n|n-1}}$$
$$\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{K_n}.\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}$$
$$\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T.\boldsymbol{K_n}^T + \boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T.\boldsymbol{K_n}^T - \boldsymbol{K_n}.\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}$$

However, one may recall that:

$$\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T.\boldsymbol{K_n}^T = \boldsymbol{K_n}.\boldsymbol{K_n}^T.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T.\boldsymbol{K_n}^T$$

And that:

$$\boldsymbol{K_n}^T.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T = (\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T + \boldsymbol{R_n})$$

Hence:

$$\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T.\boldsymbol{K_n}^T - \boldsymbol{K_n}.\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}} + \boldsymbol{K_n}.(\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T + \boldsymbol{R_n}).\boldsymbol{K_n}^T$$

$$\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T.\boldsymbol{K_n}^T - \boldsymbol{K_n}.\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}} + \boldsymbol{K_n}.\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T.\boldsymbol{K_n}^T + \boldsymbol{K_n}.\boldsymbol{R_n}.\boldsymbol{K_n}^T$$

Leading to the Joseph form of the covariance:

$$\boldsymbol{P_{n|n}} = (\boldsymbol{I} - \boldsymbol{K_n}.\boldsymbol{G_n}).\boldsymbol{P_{n|n-1}}.(\boldsymbol{I} - \boldsymbol{K_n}.\boldsymbol{G_n})^T + \boldsymbol{K_n}.\boldsymbol{R_n}.\boldsymbol{K_n}^T$$

Before continuing, one can note that for any matrix $\boldsymbol{A}$ and symmetric matrix $\boldsymbol{B}$ [9]:

$$\frac{\partial\ tr(\boldsymbol{A}.\boldsymbol{B}.\boldsymbol{A}^T)}{\partial \boldsymbol{A}} = 2.\boldsymbol{A}.\boldsymbol{B}$$

And that minimizing $\boldsymbol{P_{n|n}}$ is equivalent to minimize its trace $tr(\boldsymbol{P_{n|n}})$. Indeed, by definition, the covariance matrix is the expected value of a product of a vector by its transpose:

$$\boldsymbol{P} = \mathbb{E}(\boldsymbol{x}^T.\boldsymbol{x})$$

if $\boldsymbol{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}^T \in \mathbb{R}^{n,1}$, then one can see by developing the calculation that: $tr(\boldsymbol{x}.\boldsymbol{x}^T) = \boldsymbol{x}^T.\boldsymbol{x}$. Hence:

$$\mathbb{E}(tr(\boldsymbol{x}.\boldsymbol{x}^T)) = tr(\mathbb{E}(\boldsymbol{x}.\boldsymbol{x}^T)) = \mathbb{E}(\boldsymbol{x}^T.\boldsymbol{x}) = \boldsymbol{P}$$

Hence differentiating the trace of the Joseph form of the covariance matrix and settling it to zero for optimality leads to:

$$-2.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T + 2.\boldsymbol{K_n}.\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n}^T + 2.\boldsymbol{K_n}.\boldsymbol{R_n} = \boldsymbol{0}$$

Which allows to formulate the expression of the Kalman gain 2.42:

$$\boldsymbol{K} = \boldsymbol{P}.\boldsymbol{G}^T.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T + \boldsymbol{R})^{-1}$$

Thus:

$$\boldsymbol{P}.\boldsymbol{G}^T = \boldsymbol{K}.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T + \boldsymbol{R})$$

Then, equation 2.39 becomes:

$$\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}} = (\boldsymbol{P} - \boldsymbol{P}.\boldsymbol{G}^T.(\boldsymbol{R} + \boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T)^{-1}.\boldsymbol{G}.\boldsymbol{P}).(\boldsymbol{G}^T.\boldsymbol{R}^{-1}.\boldsymbol{y} + \boldsymbol{P}^{-1}.\boldsymbol{m})$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}} = \boldsymbol{m}+\boldsymbol{P}.\boldsymbol{G}^T.\boldsymbol{R}^{-1}.\boldsymbol{y}-\boldsymbol{P}.\boldsymbol{G}^T.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T+\boldsymbol{R})^{-1}.\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T.\boldsymbol{R}^{-1}.\boldsymbol{y}-\boldsymbol{P}.\boldsymbol{G}^T.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T+\boldsymbol{R})^{-1}.\boldsymbol{G}.\boldsymbol{m}$$

And by injecting $\boldsymbol{K}$ in it:

$$\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}} = \boldsymbol{m} + \boldsymbol{K}.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T + \boldsymbol{R}).\boldsymbol{R}^{-1}.\boldsymbol{y} - \boldsymbol{K}.\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T.\boldsymbol{R}^{-1}.\boldsymbol{y} - \boldsymbol{K}.\boldsymbol{G}.\boldsymbol{m}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}} = \boldsymbol{m} + (\boldsymbol{K}.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T + \boldsymbol{R}).\boldsymbol{R}^{-1} - \boldsymbol{K}.\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T.\boldsymbol{R}^{-1}).\boldsymbol{y} - \boldsymbol{K}.\boldsymbol{G}.\boldsymbol{m}$$

Which leads to, after simplification:

$$\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}} = \boldsymbol{m} + \boldsymbol{K}.(\boldsymbol{y} - \boldsymbol{G}.\boldsymbol{m}) \tag{2.43}$$

This close the numerical demonstration aside leading to the measurement update Kalman Filter equations. Indeed, the equation 2.43 can be seen as the closed-form solution since $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$ is the mean of the prediction of the state hence $\boldsymbol{m}$:

Closed-form solution of the estimated state

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K}_{\boldsymbol{n}}.(\boldsymbol{y}_{\boldsymbol{n}} - \boldsymbol{G}_{\boldsymbol{n}}.\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) \tag{2.44}$$

With $\boldsymbol{K}_{\boldsymbol{n}}$ the Kalman gain at the current step time n that was previous defined through equation 2.42. In a more rigorous manner, the Kalman gain is then:

Kalman Filter gain

$$\boldsymbol{K}_{\boldsymbol{n}} = \boldsymbol{P}_{\boldsymbol{n|n-1}}.{\boldsymbol{G}_{\boldsymbol{n}}}^T.(\boldsymbol{G}_{\boldsymbol{n}}.\boldsymbol{P}_{\boldsymbol{n|n-1}}.{\boldsymbol{G}_{\boldsymbol{n}}}^T + \boldsymbol{R}_{\boldsymbol{n}})^{-1} \tag{2.45}$$

Concerning the covariance of the current estimated state, it is based on the equation 2.40:

$$cov(\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}}) = (\boldsymbol{G}^{\boldsymbol{T}}.\boldsymbol{R}^{-\boldsymbol{1}}.\boldsymbol{G} + \boldsymbol{P}^{-\boldsymbol{1}})^{-1} = \boldsymbol{P} - \boldsymbol{P}.\boldsymbol{G}^T.(\boldsymbol{R} + \boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T)^{-1}.\boldsymbol{G}.\boldsymbol{P}$$

And recall that:

$$\boldsymbol{P}.\boldsymbol{G}^T = \boldsymbol{K}.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T + \boldsymbol{R})$$

As a consequence:

$$cov(\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}}) = (\boldsymbol{G}^{\boldsymbol{T}}.\boldsymbol{R}^{-\boldsymbol{1}}.\boldsymbol{G} + \boldsymbol{P}^{-\boldsymbol{1}})^{-1} = \boldsymbol{P} - (\boldsymbol{K}.(\boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T + \boldsymbol{R})).(\boldsymbol{R} + \boldsymbol{G}.\boldsymbol{P}.\boldsymbol{G}^T)^{-1}.\boldsymbol{G}.\boldsymbol{P}$$

Leading to:

$$cov(\hat{\boldsymbol{x}}_{\boldsymbol{ReLS}}) = (\boldsymbol{I} - \boldsymbol{K}.\boldsymbol{G}).\boldsymbol{P} \tag{2.46}$$

Or using a more rigorous notation:

Covariance of the state

$$\boldsymbol{P}_{\boldsymbol{n|n}} = (\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{n}}.\boldsymbol{G}_{\boldsymbol{n}}).\boldsymbol{P}_{\boldsymbol{n|n-1}} \tag{2.47}$$

NOTA - In the literature, the usual form used for the filter might be the following equivalent one: $\boldsymbol{P}_{\boldsymbol{n|n}} = \boldsymbol{P}_{\boldsymbol{n|n-1}} - \boldsymbol{K}_{\boldsymbol{n}}.(\boldsymbol{G}_{\boldsymbol{n}}.\boldsymbol{P}_{\boldsymbol{n|n-1}}.{\boldsymbol{G}_{\boldsymbol{n}}}^T + \boldsymbol{R}_{\boldsymbol{n}}).{\boldsymbol{K}_{\boldsymbol{n}}}^T$.

2.2.1.2 Prediction step equations determination

In this part, we want to determine the predicted mean $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$ and the predicted state covariance $\boldsymbol{P}_{\boldsymbol{n|n-1}}$. Two important properties to be defined before the demonstration are the following. First, the conditional expectation of $\boldsymbol{x}_{\boldsymbol{n}}$ given the data $\boldsymbol{y}_{\boldsymbol{n}}$ is denoted:

$$\mathbb{E}\{\boldsymbol{x}_{\boldsymbol{n}}|\boldsymbol{y}_{\boldsymbol{1:n}}\} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n}} \tag{2.48}$$

Then the covariance of $\boldsymbol{x}_{\boldsymbol{n}}$ given the data $\boldsymbol{y}_{\boldsymbol{n}}$ is:

$$cov\{\boldsymbol{x}_{\boldsymbol{n}}|\boldsymbol{y}_{\boldsymbol{1:n}}\} = \boldsymbol{P}_{\boldsymbol{n|n}} \tag{2.49}$$

We can start by noticing that $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$ is the mean of the current state knowing prior measurements. Hence:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \mathbb{E}\left\{\boldsymbol{x_n}|\boldsymbol{y_{1:n-1}}\right\}$$

Then, based on the linear state-space process model assumption equation 2.24:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \mathbb{E}\left\{F_n.\boldsymbol{x_{n-1}} + \boldsymbol{q_n}|\boldsymbol{y_{1:n-1}}\right\}$$

And by using the linear properties of the expected value:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \mathbb{E}\left\{F_n.\boldsymbol{x_{n-1}}|\boldsymbol{y_{1:n-1}}\right\} + \mathbb{E}\left\{\boldsymbol{q_n}|\boldsymbol{y_{1:n-1}}\right\}$$

Since the error noise of the state-model process is assumed to be of zero-mean: $\mathbb{E}\left\{\boldsymbol{q_n}|\boldsymbol{y_{1:n-1}}\right\} = 0$, this leads to:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = F_n.\mathbb{E}\left\{\boldsymbol{x_{n-1}}|\boldsymbol{y_{1:n-1}}\right\} = F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}} \tag{2.50}$$

In the literature, the state-space process model, as previously defined through equation 2.24 for example, is often of the form $\boldsymbol{x_{n+1}} = F_n.\boldsymbol{x_n} + G_n.\boldsymbol{u_n} + \boldsymbol{w_n}$. It is straightforward that using this model is then equivalent to $\boldsymbol{x_n} = F_n.\boldsymbol{x_{n-1}} + G_n.\boldsymbol{u_{n-1}} + \boldsymbol{w_{n-1}}$ (with an abuse of notation on the indices of the matrices) and will lead to:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \mathbb{E}\left\{\boldsymbol{x_n}|\boldsymbol{y_{1:n-1}}\right\}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \mathbb{E}\left\{F_n.\boldsymbol{x_{n-1}} + G_n.\boldsymbol{u_{n-1}} + \boldsymbol{w_{n-1}}|\boldsymbol{y_{1:n-1}}\right\}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = F_n.\mathbb{E}\left\{\boldsymbol{x_{n-1}}|\boldsymbol{y_{1:n-1}}\right\} + G_n.\mathbb{E}\left\{\boldsymbol{u_{n-1}}|\boldsymbol{y_{1:n-1}}\right\} + \mathbb{E}\left\{\boldsymbol{w_{n-1}}|\boldsymbol{y_{1:n-1}}\right\}$$

But the input $\boldsymbol{u_n}$ is considered to be known precisely. And, as usual, the process noise is supposed to be with a zero-mean value:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}} + G_n.\boldsymbol{u_{n-1}}$$

Which is effectively the form that will be used in the next sections to describe our model.

Concerning the covariance, without loss of generality, one can write that based on the auto-covariance matrix expression of:

$$cov(X, X) = \mathbb{E}((X - \mathbb{E}(X)).(X - \mathbb{E}(X)^T)$$

Then here:

$$\boldsymbol{P_{n|n-1}} = \mathbb{E}((\boldsymbol{x_n} - \mathbb{E}\left\{\boldsymbol{x_n}|\boldsymbol{y_{1:n-1}}\right\}).(\boldsymbol{x_n} - \mathbb{E}\left\{\boldsymbol{x_n}|\boldsymbol{y_{1:n-1}}\right\})^T|\boldsymbol{y_{1:n-1}})$$

According to equation 2.50:

$$\mathbb{E}\left\{\boldsymbol{x_n}|\boldsymbol{y_{1:n-1}}\right\} = F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}$$

Then, by using the state-space process model and the previous relation, one gets:

$$\boldsymbol{P_{n|n-1}} = \mathbb{E}((F_n.\boldsymbol{x_{n-1}} + \boldsymbol{q_n} - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).(F_n.\boldsymbol{x_{n-1}} + \boldsymbol{q_n} - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}})$$

Developing the expression will lead to the following one:

$$\begin{aligned}\boldsymbol{P_{n|n-1}} = \mathbb{E}(F_n.\boldsymbol{x_{n-1}}.(F_n.\boldsymbol{x_{n-1}})^T - F_n.\boldsymbol{x_{n-1}}.(F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}.(F_n.\boldsymbol{x_{n-1}})^T + \\ F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}.(F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}}) + \mathbb{E}(\boldsymbol{q_n}.(F_n.\boldsymbol{x_{n-1}} - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}}) + \\ \mathbb{E}((F_n.\boldsymbol{x_{n-1}} - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).\boldsymbol{q_n}^T|\boldsymbol{y_{1:n-1}}) + \mathbb{E}(\boldsymbol{q_n}.\boldsymbol{q_n}^T|\boldsymbol{y_{1:n-1}})\end{aligned}$$

Using the same assumptions on the noise of the process model, it is straightforward that:

$$\mathbb{E}(\boldsymbol{q_n}.(F_n.\boldsymbol{x_{n-1}} - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}}) = 0$$

$$\mathbb{E}((F_n.\boldsymbol{x_{n-1}} - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).{\boldsymbol{q_n}}^T|\boldsymbol{y_{1:n-1}}) = 0$$

Thus:

$$\boldsymbol{P_{n|n-1}} = \mathbb{E}(F_n.\boldsymbol{x_{n-1}}.(F_n.\boldsymbol{x_{n-1}})^T - F_n.\boldsymbol{x_{n-1}}.(F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}.(F_n.\boldsymbol{x_{n-1}})^T + F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}.(F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}}) + \mathbb{E}(\boldsymbol{q_n}.{\boldsymbol{q_n}}^T|\boldsymbol{y_{1:n-1}})$$

Numerically, we can notice that the interior of the first term of the right-hand side of the previous equation is:

$$F_n.\boldsymbol{x_{n-1}}.(F_n.\boldsymbol{x_{n-1}})^T - F_n.\boldsymbol{x_{n-1}}.(F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T - F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}.(F_n.\boldsymbol{x_{n-1}})^T + F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}.(F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}} = (\boldsymbol{F_n}.\boldsymbol{x_{n-1}} - \boldsymbol{F_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).(\boldsymbol{F_n}.\boldsymbol{x_{n-1}} - \boldsymbol{F_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T$$

Therefore, leading to:

$$\boldsymbol{P_{n|n-1}} = \mathbb{E}((\boldsymbol{F_n}.\boldsymbol{x_{n-1}} - \boldsymbol{F_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).(\boldsymbol{F_n}.\boldsymbol{x_{n-1}} - \boldsymbol{F_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}}) + \mathbb{E}(\boldsymbol{q_n}.{\boldsymbol{q_n}}^T|\boldsymbol{y_{1:n-1}})$$

Concerning the first term of the right-hand side of the previous equation, we can note that:

$$\mathbb{E}((\boldsymbol{F_n}.\boldsymbol{x_{n-1}} - \boldsymbol{F_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).(\boldsymbol{F_n}.\boldsymbol{x_{n-1}} - \boldsymbol{F_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})^T|\boldsymbol{y_{1:n-1}}) = \boldsymbol{F_n}.\mathbb{E}((\boldsymbol{x_{n-1}} - \mathbb{E}(\boldsymbol{x_{n-1}}|\boldsymbol{y_{1:n-1}})).(\boldsymbol{x_{n-1}} - \mathbb{E}(\boldsymbol{x_{n-1}}|\boldsymbol{y_{1:n-1}}))^T|\boldsymbol{y_{1:n-1}}).{\boldsymbol{F_n}}^T = \boldsymbol{F_n}.\boldsymbol{P_{n-1|n-1}}.{\boldsymbol{F_n}}^T$$

For the second term of the right-hand side of the equation, one has:

$$\mathbb{E}(\boldsymbol{q_n}.{\boldsymbol{q_n}}^T|\boldsymbol{y_{1:n-1}}) = \mathbb{E}((\boldsymbol{q_n} - \mathbb{E}(\boldsymbol{q_n})).(\boldsymbol{q_n} - \mathbb{E}(\boldsymbol{q_n}))^T|\boldsymbol{y_{1:n-1}}) = cov(\boldsymbol{q_n}) = \boldsymbol{Q_n}$$

The overall equation leads then to:

$$\boldsymbol{P_{n|n-1}} = \boldsymbol{F_n}.\boldsymbol{P_{n-1|n-1}}.{\boldsymbol{F_n}}^T + \boldsymbol{Q_n} \tag{2.51}$$

The predicted mean is therefore base on equation 2.50:

Predicted mean

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}} \tag{2.52}$$

Whereas the predicted covariance is the equation 2.51:

Predicted covariance

$$\boldsymbol{P_{n|n-1}} = \boldsymbol{F_n}.\boldsymbol{P_{n-1|n-1}}.{\boldsymbol{F_n}}^T + \boldsymbol{Q_n} \tag{2.53}$$

As a result, based on the previously established equations for the prediction and measurement update steps, one can build the following classical and generic Kalman Filter algorithm:

Algorithm 1 Kalman Filter

Initialization:
$\hat{x}_{0|0} = m_0, \quad P_{0|0} = P_0$
for $n \in [\![1, N]\!]$ **do**
  Prediction step calculation:
    $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = F_n.\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}$
    $\boldsymbol{P_{n|n-1}} = \boldsymbol{F_n}.\boldsymbol{P_{n-1|n-1}}.\boldsymbol{F_n^T} + \boldsymbol{Q_n}$
  Measurement update step calculation:
    $\boldsymbol{K_n} = \boldsymbol{P_{n|n-1}}.{\boldsymbol{G_n}}^T.\left(\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}.{\boldsymbol{G_n}}^T + \boldsymbol{R_n}\right)^{-1}$
    $\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.\left(\boldsymbol{y_n} - \boldsymbol{G_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}\right)$
    $\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{K_n}.\left(\boldsymbol{G_n}.\boldsymbol{P_{n|n-1}}.\boldsymbol{G_n^T} + \boldsymbol{R_n}\right).\boldsymbol{K_n^T}$
end for

From an analytical perspective, if the predicted error covariance increases; either due to large un-

certainty in the process noise covariance $\boldsymbol{Q_n}$ or to a large value of the state transition factor $\boldsymbol{F_n}$, then the predicted covariance $\boldsymbol{P_{n|n-1}}$ becomes large. As a direct consequence, the Kalman gain $\boldsymbol{K_n}$ also increases. During the measurement update, where the state estimate is computed as $\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.\left(\boldsymbol{y_n} - \boldsymbol{G_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}\right)$, the filter therefore places greater weight on the innovation term $\boldsymbol{K_n}.(\boldsymbol{y_n} - \boldsymbol{G_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})$, relying more heavily on the measurement and less on the predicted state $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$, which is deemed uncertain. Conversely, if a small uncertainty is imposed on the process model; indicating strong confidence in the implemented model, then $\boldsymbol{P_{n|n-1}}$ is reduced. This leads to a smaller Kalman gain $\boldsymbol{K_n}$, and during the estimation step, the filter relies predominantly on the predicted state $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$ while assigning less weight to the measurement correction term.

In the state estimation equation, the term $\boldsymbol{G_n}.\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$ can be interpreted as the predicted measurement and is therefore denoted $\hat{\boldsymbol{y}}_{\boldsymbol{n}}$. Consequently, equation 2.44 can be rewritten as $\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \boldsymbol{K_n}.(\boldsymbol{y_n} - \hat{\boldsymbol{y}}_{\boldsymbol{n}}) = \boldsymbol{K_n}.\Delta\boldsymbol{y}$. The quantity $\Delta\boldsymbol{y}$ thus represents the innovation, that is, the discrepancy between the actual measurement and its prediction. A large value of $\Delta\boldsymbol{y}$ indicates that the predicted state is significantly different from the true state and therefore requires a substantial correction. Conversely, a small value of $\Delta\boldsymbol{y}$ implies that the predicted state is already close to the true state and only a minor correction is needed. This mechanism constitutes the innovation step of the Kalman filter.

2.2.2 Extended Kalman Filters for non-linear systems

As mentioned earlier, the Kalman Filter is for linear systems of the form of equations 2.24 and 2.25. However, when considering the general form:

$$\boldsymbol{x_n} = \boldsymbol{f}(\boldsymbol{x_{n-1}}) + \boldsymbol{q_n} \tag{2.54}$$

$$\boldsymbol{y_n} = \boldsymbol{g}(\boldsymbol{x_n}) + \boldsymbol{r_n} \tag{2.55}$$

Then closed-form solutions are usually not definable. In the previous equations, note that $cov(\boldsymbol{q_n}) = \boldsymbol{Q_n}$ and $cov(\boldsymbol{r_n}) = \boldsymbol{R_n}$ and that $\mathbb{E}(\boldsymbol{q_n}) = \mathbb{E}(\boldsymbol{r_n}) = 0$, which are features of white-noise processes.

The goal is therefore to approximate the previous set of equations into a linear one using the first order Taylor Series expansions of the non-linear dynamic and measurement models. Let I be an interval of $\mathbb{R}$, $n \in \mathbb{N}^*$ and f a function at least one time differentiable on I such that $f'(x_0)$ exists, with x_0 an equilibrium point. Then, at first order, the following approximation holds:

$$f(x) = f(x_0) + (x - x_0).f'(x_0) \tag{2.56}$$

2.2.2.1 Prediction step equations determination

We will also assume here that the state estimate and its covariance up to the previous step $n-1$ are known and available, respectively $\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}$ and $\boldsymbol{P_{n-1|n-1}}$. One can then linearize, in the frame of Taylor series approximation, equation 2.54 around the equilibrium point of the state estimate at the previous step time $\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}$ such that:

$$\boldsymbol{x_n} \approx \boldsymbol{f}(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}) + \boldsymbol{f'}(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).(\boldsymbol{x_{n-1}} - \boldsymbol{x_{n-1|n-1}}) + \boldsymbol{q_n} \tag{2.57}$$

Denoting $\boldsymbol{f'}(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})$ to be the Jacobian matrix of the vector-value function $\boldsymbol{f}$. Most of the time, this Jacobian matrix is denoted $\boldsymbol{F}_x$, and its elements can be determined through:

$$\boldsymbol{F}_x(i,j) = \boldsymbol{f'}(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})(i,j) = \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{x}_j}(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}) \tag{2.58}$$

Where here i and j are indexes in $\mathbb{N}^*$ depending on the size of $\boldsymbol{f}$ and $\boldsymbol{x_n}$. Using this reformulation and linearization allows us to retrieve and predict the mean and covariance of the state $\boldsymbol{x_n}$ at t_n given the previous set of measurements, exactly as it was done for the linear system and the Kalman Filter through equations 2.50 and 2.51. The following result has been obtained for the mean of the state:

Predicted mean

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}) \tag{2.59}$$

And for the covariance of the predicted state:

Predicted covariance

$$\boldsymbol{P_{n|n-1}} = \boldsymbol{F}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).\boldsymbol{P_{n-1|n-1}}.\boldsymbol{F}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}) + \boldsymbol{Q_n} \tag{2.60}$$

2.2.2.2 Measurement update step equations determination

The determination of the set of equations for this part of the EKF is similar to the one explained in the previous subsection on the KF. Therefore, the main results from the literature will be directly extracted and presented hereafter. As it was the case for the state-space process model, the measurement model shall be linearized around an equilibrium point, this time chosen to be the predicted mean $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$, such that:

$$\boldsymbol{y_n} \approx \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).(\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{r_n} \tag{2.61}$$

Where this time, $\boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})$ denotes the Jacobian matrix of $\boldsymbol{g}$. As for the Jacobian matrix of the state model, this one will usually be noted such that:

$$\boldsymbol{G}_x(i,j) = \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})(i,j) = \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{x}_j}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) \tag{2.62}$$

Where here i and j are indexes in $\mathbb{N}^*$ depending on the size of $\boldsymbol{g}$ and $\boldsymbol{y_n}$.

Let us demonstrate that suing a proper change of variable, the cost function to be minimized in the case of the EKF is the same as for the KF in one from the propper previous section. Based on equation 2.33, the regularized least square criterion can be written such as:

$$J_{ReLS}(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x}))^T.\boldsymbol{R^{-1}}.(\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x})) + (\boldsymbol{x} - \boldsymbol{m})^T.\boldsymbol{P^{-1}}.(\boldsymbol{x} - \boldsymbol{m}) \tag{2.63}$$

However, based on equation 2.55, $\boldsymbol{g}(\boldsymbol{x_n}) = \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).(\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})$. Hence the criterion can be written:

$$\begin{aligned} J_{ReLS}(\boldsymbol{x}) = (\boldsymbol{y_n} - (\boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).(\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})))^T.\boldsymbol{R^{-1}}. \\ (\boldsymbol{y_n} - (\boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).(\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}))) + \\ (\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})^T.\boldsymbol{P^{-1}}.(\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) \end{aligned} \tag{2.64}$$

By letting $\boldsymbol{z_n} = \boldsymbol{y_n} - \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}$, the regularized least squares criterion can, in fact, be put under the form:

$$\begin{aligned} J_{ReLS}(\boldsymbol{x}) = (\boldsymbol{z_n} - \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{x_n})^T.\boldsymbol{R^{-1}}.(\boldsymbol{z_n} - \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{x_n}) + \\ (\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})^T.\boldsymbol{P^{-1}}.(\boldsymbol{x_n} - \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})) \end{aligned} \tag{2.65}$$

Which is linear in $\boldsymbol{x_n}$ such as in the previous subsection. Therefore, the same optimization problem shall be solved. The following closed-form solution is then established as it was done in the prior section:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.(\boldsymbol{z_n} - \boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) \tag{2.66}$$

As well as the Kalman gain and the covariance:

$$\boldsymbol{K_n} = \boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).(\boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{R_n})^{-1} \tag{2.67}$$

$$\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{K_n}.(\boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{R_n}).\boldsymbol{K_n}^T \tag{2.68}$$

Eventually, for the measurement update equation 2.66, by taking back the variable $\boldsymbol{y_n}$, one has:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.(\boldsymbol{z_n} - \boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.(\boldsymbol{y_n} - \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{g'}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} - \boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.(\boldsymbol{y_n} - \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})) \tag{2.69}$$

Therefore, the measurement equations for the Extended Kalman Filter are the following ones:

Closed-form solution of the estimated state

$$\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.(\boldsymbol{y_n} - \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}})) \tag{2.70}$$

Kalman Filter gain

$$\boldsymbol{K_n} = \boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).(\boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{R_n})^{-1} \tag{2.71}$$

Covariance of the state

$$\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{K_n}.(\boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{R_n}).\boldsymbol{K_n}^T \tag{2.72}$$

As previously done in the KF section, one will find hereafter the implemented algorithm for the embedded EKF estimation and filtering algorithm:

Algorithm 2 Extended Kalman Filter

Initialization:
$\hat{x}_{0|0} = m_0, \quad P_{0|0} = P_0$
for $n \in [\![1, N]\!]$ **do**
  Prediction step calculation:
    $\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}})$
    $\boldsymbol{P_{n|n-1}} = \boldsymbol{F}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}).\boldsymbol{P_{n-1|n-1}}.\boldsymbol{F}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n-1|n-1}}) + \boldsymbol{Q_n}$
  Measurement update step calculation:
    $\boldsymbol{K_n} = \boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).(\boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{R_n})^{-1}$
    $\hat{\boldsymbol{x}}_{\boldsymbol{n|n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}} + \boldsymbol{K_n}.(\boldsymbol{y_n} - \boldsymbol{g}(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}))$
    $\boldsymbol{P_{n|n}} = \boldsymbol{P_{n|n-1}} - \boldsymbol{K_n}.(\boldsymbol{G}_x(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}).\boldsymbol{P_{n|n-1}}.\boldsymbol{G}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{n|n-1}}) + \boldsymbol{R_n}).\boldsymbol{K_n}^T$
end for

Note that several versions of this algorithms were designed, verified and validated through simulations for the project. They can be found in the Navigation repository of the project.

2.2.3 Implementation in the Simulator

In the following sections, the chosen model will be detailed, as well as the preliminary tuning of the previously presented algorithms. The real sensibility analysis, in-depth details of tuning and performance evaluation will be done and presented through another study, not yet realized.

2.2.3.1 Model description for prediction

It is clear for now that the KF and EKF base themselves on an embedded model. Indeed, due to the uncertainty of the sensors measurements, the ship might be biased in term of knowledge of its state. The goal of the model is to provide to the ship an idea of what we think that the situation really is. This is done through the use of a mathematical model that best represents the circumstances and the actual environment.

In the introductory part of this justification file, it was said that the goal is to design a heading controller for path-following. Hence, the goal is to enslave the heading of the ship in order for it to follow a predefined trajectory. This problem can be seen as a 3 degrees of freedom in terms of state variables. Indeed, the situation can be depicted through a local horizontal view from above of the ENU frame with the ship such that:

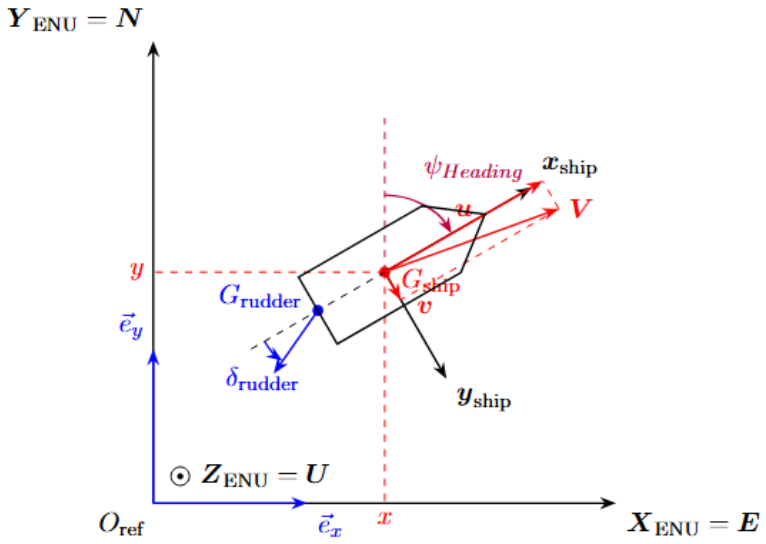

Figure 2.14: Ship kinematics, reference frames, velocity decomposition and rudder deflection.

What needs to be understood from the previous Figure is that whenever the rudder bar is used to create an angle δ_{rudder}, the ship is turning. Due to this, the velocity vector will be moved, that is the projections of $\boldsymbol{V}$ on the axis $\boldsymbol{x_{ship}}$ and $\boldsymbol{y_{ship}}$, respectively $\boldsymbol{u}$ and $\boldsymbol{v}$, can be observed. Note that whenever the ship is not turning, hence when $\delta_{rudder} = 0$, then the velocity vector $\boldsymbol{V}$ is aligned with the longitudinal axis of the ship $\boldsymbol{x_{ship}}$. In practice, due to perturbations, the velocity vector will never be perfectly aligned to the ship's main longitudinal axis. The previous figure will be used to build our mathematical model for the description of the kinematics and dynamics of the ship embedded inside of the EKF.

One can therefore note that, by letting $\boldsymbol{x_s} = \boldsymbol{x_{ship}}$ and $\boldsymbol{y_s} = \boldsymbol{y_{ship}}$:

$$\boldsymbol{V} = u.\boldsymbol{x_s} + v.\boldsymbol{y_s} \tag{2.73}$$

On the other hand:

$$\boldsymbol{V} = \frac{d\,\boldsymbol{O_{ref}G_s}}{dt} = \dot{x}.\boldsymbol{e_x} + \dot{y}.\boldsymbol{e_y} \tag{2.74}$$

By using the relations between the reference base vectors of each frame; the inertial fixed one in ENU $(O_{ref}; \boldsymbol{e_x}, \boldsymbol{e_y}, \boldsymbol{e_z})$ and the one fixed to the ship $(G_s; \boldsymbol{x_s}, \boldsymbol{y_s}, \boldsymbol{z_s})$:

$$\boldsymbol{x_s} = \sin(\psi).\boldsymbol{e_x} + \cos(\psi).\boldsymbol{e_y}$$
$$\boldsymbol{y_s} = \cos(\psi).\boldsymbol{e_x} - \sin(\psi).\boldsymbol{e_y}$$

This leads to:

$$\boldsymbol{V} = u.(\sin(\psi).\boldsymbol{e_x} + \cos(\psi).\boldsymbol{e_y}) + v.(\cos(\psi).\boldsymbol{e_x} - \sin(\psi).\boldsymbol{e_y}) \tag{2.75}$$

Identifying this expression with equation 2.74 due to the uniqueness of the velocity vector, one retrieves:

$$\dot{x} = u.\sin(\psi) + v.\cos(\psi) \tag{2.76}$$

$$\dot{y} = u.\cos(\psi) - v.\sin(\psi) \tag{2.77}$$

In practice, we will make the assumption that in most of the cases, even when turning especially in a slowly way: $u >> v$. In that case, one can suppose that $V \approx u.\boldsymbol{x_s}$ in comparison with equation 2.73. The previous set of equations becomes:

$$\dot{x}(t) = V(t).\sin(\psi(t)) \tag{2.78}$$

$$\dot{y}(t) = V(t).\cos(\psi(t)) \tag{2.79}$$

These equations are continuous. However, the embedded KF/EKF are run on a computer, handling discrete equations. Therefore, there is a need to retrieve the associated discrete time equations. For that,

we will use the first order Euler method widely described and used in the literature and industry. In our case, this will implies that:

$$\dot{x}(t) = \frac{dx}{dt} \approx \frac{x_{k+1} - x_k}{\delta t_k} \tag{2.80}$$

$$\dot{y}(t) = \frac{dy}{dt} \approx \frac{y_{k+1} - y_k}{\delta t_k} \tag{2.81}$$

$$\psi(t) \approx \psi_k \tag{2.82}$$

$$V(t) \approx V_k \tag{2.83}$$

Where $(.)_k$ is the parameter $(.)$ evaluated at the time step t_k. The time constant δt_k is the discretization step. Injecting equations 2.80 to 2.83 in equations 2.78 and 2.79 will lead to:

$$x_{k+1} = x_k + \delta t_k.(V_k.\sin(\psi_k)) \tag{2.84}$$

$$y_{k+1} = y_k + \delta t_k.(V_k.\cos(\psi_k)) \tag{2.85}$$

These are the first set of equations of our model since we need to know the location of the ship in terms of positions. Since these equations need the heading angle ψ_k and the velocity V_k at the current step time, one needs also to establish which relations to use for these variables. Concerning the velocity, we will make the assumption that it will remain constant with a certain amount of uncertainty:

$$V_k = V_{k-1} + \delta t_{k-1}.\delta V_{k-1} \tag{2.86}$$

In fact, it is known that the velocity depends on the acceleration through the following relations:

$$a(t) = \frac{dV}{dt}(t) \approx \frac{V_{k+1} - V_k}{\delta t_k}$$

Which can be put under the following discretized form:

$$V_k = V_{k-1} + \delta t_{k-1}.a_{k-1} \tag{2.87}$$

However, since the final onboard sensors have not yet been fully selected and characterized, there is no guarantee that acceleration measurements will be available. An Extended Kalman Filter has nevertheless been designed under the assumption that such measurements can be provided, and its implementation is available at the following link: detailed EKF with accelerations. At present, the most up-to-date version of the simulator relies on the simplified velocity model described by equation 2.86. As this model lacks accuracy, the associated uncertainty can be deliberately increased through the filter covariance matrix, while simultaneously reducing the uncertainty assigned to the velocity measurements provided by the sensor.

NOTA - In the following sections and as it was done here for the velocity, the terms $\delta(.)_{k-1}$, with $(.)$ a certain parameter will represent the noise of the parameter $(.)$ at the step time $k-1$.

For the heading angle, two options arise in terms of model to be used. Either the selection of a very simple model, where we say that as for the velocity, the heading angle is constant over a step time with some perturbations such that $\psi_k = \psi_{k-1} + \delta\psi_k$. However, the heading angle is not constant along the mission and vary in function of the ship dynamics and the rudder angle in input. Or, as it was shown in the first Simulator justification file and in the relevant literature on the topic, one can use the Nomoto's first order equation such that [10][11]:

$$\frac{r}{\delta}(s) = \frac{K}{1 + T \cdot s} \tag{2.88}$$

Or equivalently:

$$\frac{\psi}{\delta}(s) = \frac{K}{s \cdot (1 + T \cdot s)} \tag{2.89}$$

Where K is the static gain constant of the ship, T its constant time period, s the Laplace function, ψ the heading angle, r the yaw rate angular rate and δ the rudder angle. As previously done, the discretization of equation 2.88 can be done as it follows, coming back to the time differential equation:

$$\frac{r}{\delta}(s) = \frac{K}{1 + T \cdot s}$$

$$r + T.s.r = K.\delta$$

Which gives in the time domain:

$$r(t) + T.\frac{dr}{dt}(t) = r(t) + T.\dot{r}(t) = K.\delta(t)$$

$$\dot{r}(t) = -\frac{r(t)}{T} + \frac{K}{T}.\delta(t) \tag{2.90}$$

However, this equation has been obtained through simplification assumptions on the dynamics and perturbations acting on the ship [12]. Therefore, a bias term can be included that will represent all the neglected actions as well as a noise contribution supposed to be random $\delta r(t)$ such that the previous equation can be rewritten:

$$\dot{r}(t) = -\frac{r(t)}{T} + \frac{K}{T}.\delta(t) + b(t) + \delta r(t) \tag{2.91}$$

The time derivative of the bias itself may be considered to be constant by a perturbation such as for the velocity, which is a random-walk model:

$$\dot{b}(t) = \delta b_{k-1}$$

$$b_k = b_{k-1} + \delta t_{k-1}.\delta b_{k-1} \tag{2.92}$$

The discretization of equation 2.91 leads to the following:

$$r_k = r_{k-1}+\delta t_{k-1}.(-\frac{1}{T}.r_{k-1}+\frac{K}{T}.\delta_{k-1}+b_{k-1}+\delta r_{k-1}) = (1-\frac{\delta t_{k-1}}{T}).r_{k-1}+\delta t_{k-1}.\frac{K}{T}.\delta_{k-1}+\delta t_{k-1}.b_{k-1}+\delta t_{k-1}.\delta r_{k-1} \tag{2.93}$$

Eventually, based on the angular yaw rate dynamic of the ship r, the heading ψ can be retrieved:

$$\frac{d\psi}{dt} = \dot{\psi} = r$$

Leading to the following discretized equation:

$$\psi_k = \psi_{k-1} + \delta t_{k-1}.r_{k-1} \tag{2.94}$$

By using the previously defined equations, the mathematical model of the dynamics and kinematics of the ship is defined in the filter. Since the model is non-linear, the EKF algorithm will be used.

Mathematical model of the ship

$$V_k = V_{k-1} + \delta t_{k-1}.\delta V_{k-1} \tag{2.95}$$

$$\psi_k = \psi_{k-1} + \delta t_{k-1}.r_{k-1} \tag{2.96}$$

$$r_k = (1 - \frac{\delta t_{k-1}}{T}).r_{k-1} + \delta t_{k-1}.\frac{K}{T}.\delta_{k-1} + \delta t_{k-1}.b_{k-1} + \delta t_{k-1}.\delta r_{k-1} \tag{2.97}$$

$$b_k = b_{k-1} + \delta t_{k-1}.\delta b_{k-1} \tag{2.98}$$

$$x_k = x_{k-1} + \delta t_{k-1}.(V_{k-1}.\sin(\psi_{k-1})) \tag{2.99}$$

$$y_k = y_{k-1} + \delta t_{k-1}.(V_{k-1}.\cos(\psi_{k-1})) \tag{2.100}$$

Hence, we can define the following state-space vector as in the algorithmic description part of the KF/EKF such that being $\boldsymbol{x_k} \in \mathbb{R}^{6,1}$:

$$\boldsymbol{x_k} = \begin{pmatrix} V_k & \psi_k & r_k & b_k & x_k & y_k \end{pmatrix}^T \tag{2.101}$$

In this process dynamics description, the input is the rudder angle δ that will be applied to the ship:

$$\boldsymbol{u_{k-1}} = \delta_{k-1} \ \in \mathbb{R} \tag{2.102}$$

Eventually, the noise vector of the process model is:

$$\boldsymbol{q_{k-1}} = \begin{pmatrix} \delta V_{k-1} & \delta r_{k-1} & \delta b_{k-1} \end{pmatrix} \in \mathbb{R}^{3,1} \tag{2.103}$$

In fact, the control input is bounded to be inside the interval of $[\delta_m; \delta_M]$ with δ_m the minimal rudder angle allowed and δ_M the maximal one. In practice: $\delta_m = -\delta_M$. Therefore, $\boldsymbol{u_{k-1}} \in \mathcal{U} = [\delta_m; \delta_M] \subset \mathbb{R}$.

In comparison with equation 2.54 from the non-linear state-space process model of the EKF recalled hereafter:

$$\boldsymbol{x_k} = \boldsymbol{f}(\boldsymbol{x_{k-1}}) + \boldsymbol{q_k} \tag{2.104}$$

It is possible to identify the terms, with a small modification. In our case, the noise is matrix-additive and there is an input. It will be shown that this means that the following form can be used:

$$\boldsymbol{x_k} = \boldsymbol{f}(\boldsymbol{x_{k-1}}, \boldsymbol{u_{k-1}}, \boldsymbol{q_{k-1}} = \boldsymbol{0}) + \boldsymbol{\Gamma}.\boldsymbol{q_{k-1}} \tag{2.105}$$

The matrix Γ will be described in the next section. In this case, the non-linear $\boldsymbol{f}$ vector function is:

$$\begin{aligned} \boldsymbol{f} : \mathcal{X} \times \mathcal{U} \times \mathcal{Q} &\longrightarrow \mathcal{X} \\ (\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{q_k}) &\longmapsto \boldsymbol{f}(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{q_k}) \end{aligned}$$

With $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{Q}$ the definition sets for respectively the state vector, the input vector and the noise vector, and where:

$$\boldsymbol{f}(\boldsymbol{x_{k-1}}, \boldsymbol{u_{k-1}}, \boldsymbol{q_{k-1}}) = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} = \begin{pmatrix} V_{k-1} + \delta t_{k-1}.\delta V_{k-1} \\ \psi_{k-1} + \delta t_{k-1}.r_{k-1} \\ (1 - \frac{\delta t_{k-1}}{T}).r_{k-1} + \delta t_{k-1}.\frac{K}{T}.\delta_{k-1} + \delta t_{k-1}.b_{k-1} + \delta t_{k-1}.\delta r_{k-1} \\ b_{k-1} + \delta t_{k-1}.\delta b_{k-1} \\ x_{k-1} + \delta t_{k-1}.(V_{k-1}.\sin(\psi_{k-1})) \\ y_{k-1} + \delta t_{k-1}.(V_{k-1}.\cos(\psi_{k-1})) \end{pmatrix} \tag{2.106}$$

2.2.3.2 Filter equations and matrices determination

Based on the establishment of the state-space process model, it is possible to build the equations and matrices useful for the estimation, filtering and fusion process using an EKF.

Prediction step data: based on the EKF section, we shall determine the non linear function to calculate the predicted state vector of the current step time based on equation 2.59 recalled here for convenience:

$$\boldsymbol{\hat{x}_{k|k-1}} = \boldsymbol{f}(\boldsymbol{\hat{x}_{k-1|k-1}})$$

However, as previously explained, we do have an input in our process model. Hence, the equilibrium point is still $\boldsymbol{\hat{x}_{n-1|n-1}}$ and considered to be at null disturbances. We shall then determine the function:

$$\boldsymbol{\hat{x}_{k|k-1}} = \boldsymbol{f}(\boldsymbol{\hat{x}_{k-1|k-1}}, \boldsymbol{u_{k-1}}, \boldsymbol{q_{k-1}} = \boldsymbol{0}) \tag{2.107}$$

This function can be summarized through the following result, or equivalently by using the definition of $\boldsymbol{f}$ in equation 2.107:

Equations of the function used for state prediction in EKF

$$V_k = V_{k-1} \tag{2.108}$$

$$\psi_k = \psi_{k-1} + \delta t_{k-1}.r_{k-1} \tag{2.109}$$

$$r_k = (1 - \frac{\delta t_{k-1}}{T}).r_{k-1} + \delta t_{k-1}.\frac{K}{T}.\delta_{k-1} + \delta t_{k-1}.b_{k-1} \tag{2.110}$$

$$b_k = b_{k-1} \tag{2.111}$$

$$x_k = x_{k-1} + \delta t_{k-1}.(V_{k-1}.\sin(\psi_{k-1})) \tag{2.112}$$

$$y_k = y_{k-1} + \delta t_{k-1}.(V_{k-1}.\cos(\psi_{k-1})) \tag{2.113}$$

This function has been created in Matlab inside an embedded Matlab function accessible through: function for state prediction in EKF.

The next step is to be able to calculate the prediction of the covariance of the state through equation 2.60 also recalled here:

$$\boldsymbol{P_{k|k-1}} = \boldsymbol{F}_x(\boldsymbol{\hat{x}_{k-1|k-1}}).\boldsymbol{P_{k-1|k-1}}.\boldsymbol{F}_x^T(\boldsymbol{\hat{x}_{k-1|k-1}}) + \boldsymbol{Q_k} \tag{2.114}$$

However, note that since we have used a non-additive noise model for the state-space process model, the real implemented used relation is in fact slightly more different:

$$\boldsymbol{P_{k|k-1}} = \boldsymbol{F}_x(\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}}).\boldsymbol{P_{k-1|k-1}}.\boldsymbol{F}_x^T(\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}}) + \boldsymbol{\Gamma}.\boldsymbol{Q_k}.\boldsymbol{\Gamma}^T \tag{2.115}$$

We will therefore describe how to calculate $\boldsymbol{F}_x(\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}})$ the Jacobian of the state model associated with the state vector and $\boldsymbol{\Gamma}$, the Jacobian associated with the noise vector, whose components are defined such that:

$$\boldsymbol{\Gamma} = \boldsymbol{\Gamma}(i,j) = \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{q}_j}(\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}}) \tag{2.116}$$

Again, with i and j indexes in $\mathbb{N}^*$ depending on the size of $\boldsymbol{f}$ and $\boldsymbol{q_k}$.

Using the equation 2.58 for the calculation of the components of the Jacobian matrix of the state vector process model, we obtain the following matrix:

$$\boldsymbol{F}_x(\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \delta t_{k-1} & 0 & 0 & 0 \\ 0 & 0 & (1 - \frac{\delta t_{k-1}}{T}) & \delta t_{k-1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \delta t_{k-1}.\sin\left(\hat{\psi}_{k-1|k-1}\right) & \hat{V}_{k-1|k-1}.\delta t_{k-1}.\cos\left(\hat{\psi}_{k-1|k-1}\right) & 0 & 0 & 1 & 0 \\ \delta t_{k-1}.\cos\left(\hat{\psi}_{k-1|k-1}\right) & -\hat{V}_{k-1|k-1}.\delta t_{k-1}.\sin\left(\hat{\psi}_{k-1|k-1}\right) & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.117}$$

This function has been created in an embedded Matlab function, available through this link: jacobian state function for the EKF.
Using the equation 2.216, we can retrieve the Jacobian state matrix associated with the noise vector, or simply by noting that $\boldsymbol{\Gamma}$ can be obtained:

$$\boldsymbol{x_k} = \boldsymbol{f}(\boldsymbol{x_{k-1}}, \boldsymbol{u_{k-1}}, \boldsymbol{q_{k-1}} = \boldsymbol{0}) + \begin{pmatrix} \delta t_{k-1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \delta t_{k-1} & 0 \\ 0 & 0 & \delta t_{k-1} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} . \begin{pmatrix} \delta V_{k-1} \\ \delta r_{k-1} \\ \delta b_{k-1} \end{pmatrix} \tag{2.118}$$

Implying therefore that in fact, equation 2.104 can be put under the form of equation 2.105:

$$\boldsymbol{x_k} = \boldsymbol{f}(\boldsymbol{x_{k-1}}, \boldsymbol{u_{k-1}}, \boldsymbol{q_{k-1}} = \boldsymbol{0}) + \boldsymbol{\Gamma}.\boldsymbol{q_k}$$

In order to finish with the prediction step data determination, we shall build the state-space process model covariance matrix $\boldsymbol{Q_k}$. We recall that in the KF section, we stated that the covariance matrix of the process model is $cov(\boldsymbol{q_k}) = \boldsymbol{Q_k}$. By definition of the covariance matrix, it is possible to write that:

$$\boldsymbol{Q_k} = cov(\boldsymbol{q_k}) = \begin{pmatrix} \sigma^2_{\delta V_{k-1}} & 0 & 0 \\ 0 & \sigma^2_{\delta r_{k-1}} & 0 \\ 0 & 0 & \sigma^2_{\delta b_{k-1}} \end{pmatrix} \in \mathbb{R}^{3,3} \tag{2.119}$$

Here, the diagonal terms represent the variance of the process-model variables V_k, r_k and b_k. For the sake of simplification, the error of the model is considered to be constant at each step time. In future application, some methods could be explored to improve this model and avoid this assumption. As it will be shown in the future performance report of the simulations, the tuning of these values is complicated and requires most of the time trial/error adjustments.

Measurement model step data: For the measurement model, it is necessary to identify the measurement vector available at each time step, based on the expected characteristics of the onboard sensors. As described in the first chapter, the following set of measurements can be defined and aggregated into a measurement vector:

$$\boldsymbol{y_k} = \begin{pmatrix} \psi_{k_{MAG}} & x_{k_{GPS}} & y_{k_{GPS}} & V_{k_{GPS}} & \psi_{k_{GPS}} \end{pmatrix}^T \in \mathbb{R}^{5,1} \tag{2.120}$$

Hence, it is assumed that heading measurements are available using a magnetometer through $\psi_{k_{MAG}}$ and the GPS; through $\psi_{k_{GPS}}$, as well as the positions and velocity from the GPS $x_{k_{GPS}}$, $y_{k_{GPS}}$ and $V_{k_{GPS}}$. Recall also that the measurement vector is related with the state vector by the apriori non-linear relation 2.55:

$$\boldsymbol{y_k} = \boldsymbol{g}(\boldsymbol{x_k}) + \boldsymbol{r_k} \tag{2.121}$$

In our case, one sees directly that:

$$\boldsymbol{y_k} = \begin{pmatrix} \psi_{k_{MAG}} \\ x_{k_{GPS}} \\ y_{k_{GPS}} \\ V_{k_{GPS}} \\ \psi_{k_{GPS}} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} . \begin{pmatrix} V_k \\ \psi_k \\ r_k \\ b_k \\ x_k \\ y_k \end{pmatrix} + \begin{pmatrix} \delta\psi_{k_{MAG}} \\ \delta x_{k_{GPS}} \\ \delta y_{k_{GPS}} \\ \delta V_{k_{GPS}} \\ \delta\psi_{k_{GPS}} \end{pmatrix} \tag{2.122}$$

Consequently identifying the linear connection between the state and the measurement vector is straightforward:

$$\boldsymbol{g}(\boldsymbol{x_k}) = \boldsymbol{G_k}.\boldsymbol{x_k} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} . \begin{pmatrix} V_k \\ \psi_k \\ r_k \\ b_k \\ x_k \\ y_k \end{pmatrix} \tag{2.123}$$

And the sensor uncertainties are gathered in a vector for each measurements:

$$\boldsymbol{r_k} = \begin{pmatrix} \delta\psi_{k_{MAG}} \\ \delta x_{k_{GPS}} \\ \delta y_{k_{GPS}} \\ \delta V_{k_{GPS}} \\ \delta\psi_{k_{GPS}} \end{pmatrix} \tag{2.124}$$

As such, there is no need to use the non-linear measurement step as described in the EKF section, and we can constraint our analysis to the linear one explained in the KF part. Eventually, as determined in the previous prediction part, it remains to build the covariance matrix $cov(\boldsymbol{r_k}) = \boldsymbol{R_k}$. This time, we can note that:

$$\boldsymbol{R_k} = cov(\boldsymbol{r_k}) = \begin{pmatrix} \sigma^2_{\delta\psi_{k-1_{MAG}}} & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{\delta x_{k-1_{GPS}}} & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\delta y_{k-1_{GPS}}} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\delta V_{k-1_{GPS}}} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\delta\psi_{k-1_{GPS}}} \end{pmatrix} \in \mathbb{R}^{5,5} \tag{2.125}$$

A way to tune this matrix is to retrieve the datasheet of the sensors and find out which value can be used in terms of uncertainty. Therefore, if this method of tuning is used, the numerical values chosen here in the algorithm must be the same as the ones used to simulate the sensors and their noise and bias as described in the Figure 2.9. For example, for the Simulator Case 9, the embedded uncertainty for the sensors has been chosen such that:

- Variance of the heading measurement done by the magnetometer of around 2 degrees: $\sigma^2_{\delta\psi_{k-1_{MAG}}} = (2.(\pi/180))^2$.
- Variance of the longitudinal position (East) measurement done by the GPS of around 2 meters: $\sigma^2_{\delta x_{k-1_{GPS}}} = (2)^2$.
- Variance of the lateral position (North) measurement done by the GPS of around 2 meters: $\sigma^2_{\delta y_{k-1_{GPS}}} = (2)^2$.
- Variance of the norm of the velocity measurement done by the GPS of around 0.1 m/s: $\sigma^2_{\delta V_{k-1_{GPS}}} = (0.1)^2$.
- Variance of the heading measurement done by the GPS of around 5 degrees: $\sigma^2_{\delta\psi_{k-1_{GPS}}} = (5.(\pi/180))^2$.

As a mean of example and preliminary performance analysis, using Simulator Case 9 for a simulation duration of 500 seconds over a certain set of initial conditions and parametrization, we obtain the following headings:

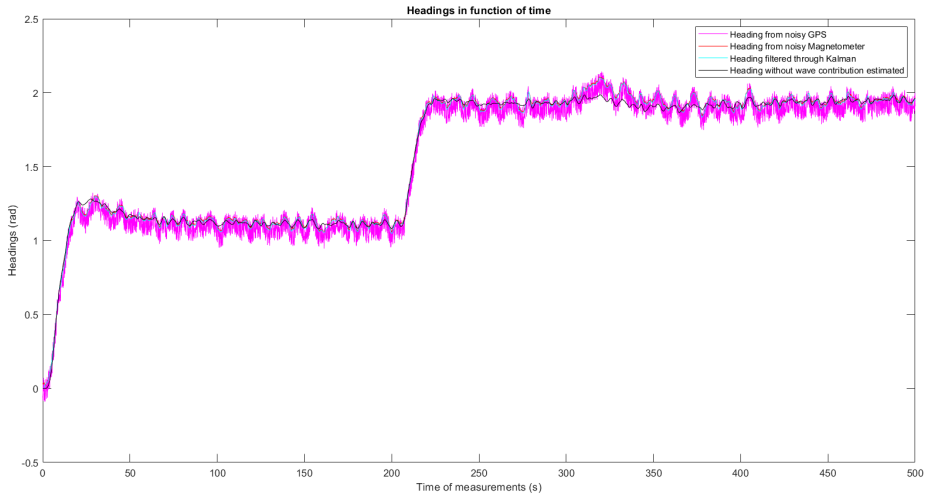

Figure 2.15: Headings simulated using Simulator Case 9.

By focusing on a certain time window:

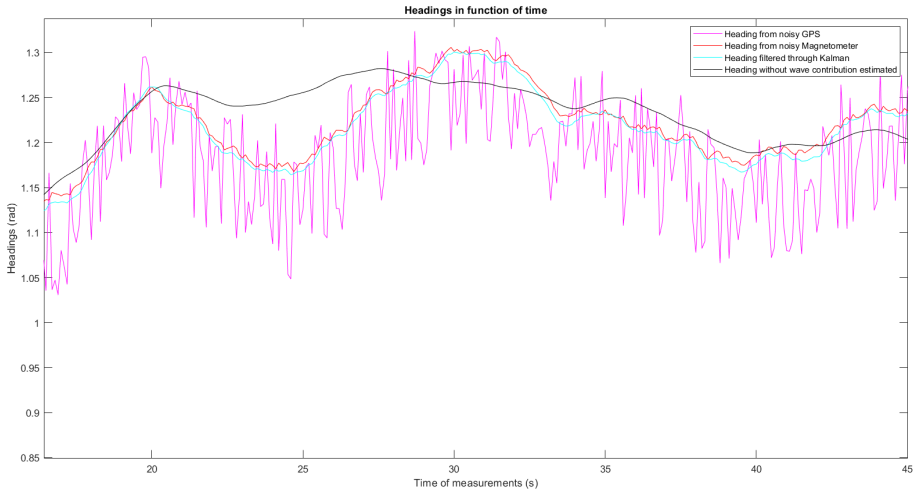

Figure 2.16: Headings simulated using Simulator Case 9 in between 10 seconds to 45 seconds.

As depicted on the legend of the figure, on purple, these are the heading measurements provided by the GPS. Whereas in red, the one provided by the magnetometer. In cyan, the filtered data given in output of the Kalman filter. For the black plot, it will be explained in the next sections. The efficiency of the filtering sub-function is then illustrated, since the final output of the EKF, the cyan plot, is less noisy than the measurements.

The last precision concerning this section is that the coefficients K and T from the Nomoto's model depend in fact on various parameters such as, for example, the hydrodynamics coefficients, the ship physical features, the density of the water and the velocity. This last parameter is of important interest. Indeed, the equation 2.88 can in fact be seen such that:

$$\frac{r}{\delta}(s) = \frac{K(V)}{1 + T(V) \cdot s}$$

This is important because the user will have the authority to act on the velocity of the ship. Therefore, if for the Simulator Cases of number 1 to 7 were using a simple Linear Time Invariant (LTI) system of

equation 2.88 for the Nomoto's model, this new simulator possesses the implemented LPV System of the ship, able to simulate a more realistic behavior of the ship in function of its velocity. Since this behavior is simulated, it is possible in fact to include it in the algorithms. Therefore, as the EKF returns an estimate of the velocity cleaned and fused, we can use it to compute more realistic parameters for the static gain of the ship and the time constant, using the algorithm ship parameters calculations. Note that since the Navigation, Guidance and Control algorithms are using these values of K and T, a bad estimate could lead to a bad control and a potential failure of the system.

This closes the discussion about the design of the EKF that was used in order to filter and fuse the data from the sensors. Up to this point, a clean state vector $\boldsymbol{x_k}$ is available at each step time for the system an on-board computer software. According to the initial Figure 1.2, we therefore have access to $\boldsymbol{x_k} = \hat{\boldsymbol{X}}_{\boldsymbol{s_r}} = \begin{pmatrix} \hat{V}_k & \hat{\psi}_k & \hat{r}_k & \hat{b}_k & \hat{x}_k & \hat{y}_k \end{pmatrix}^T$. However, concerning the heading, one can see on the previous figure that an oscillatory motion seems to be visible, especially for the colored plots of the heading. This is due to the fact that the ship is subject to a high frequency perturbation motion due to waves during this simulation case. Therefore, in order to provide the Guidance and Control algorithms with the cleanest heading possible, without perturbation, there is a need to estimate the wave contribution in the heading and remove it from the measurement. For this however, we will therefore present how the perturbations are simulated in our GN&C simulators.

2.3 Environment acting on the ship

For our algorithms to be efficiently tuned, there is the need to take into consideration, during their design, the perturbations that could act on the ship. As previously explained in the beginning of the section on the Filtering and Fusion of the data, it was considered that the main source of perturbation acting on the vessel here is the waves.

2.3.1 Wave perturbation motion

As explained in the relevant literature on ship Control, this wave perturbation contribution can be implemented through the simulation of wave induced-heading motion on the heading generated by the Nomoto's model. Indeed, it can be noted that the model of Nomoto states:

$$\frac{\psi_s}{\delta}(s) = \frac{K}{s \cdot (1 + T \cdot s)}$$

This heading angle of the ship ψ_s is seen as a low-frequency one of the total heading angle ψ_t. However, we use in the simulator a wave model that comes to perturb the simulated heading, such that the total measured heading by the sensors is:

$$\psi_t = \psi_s + \psi_w \tag{2.126}$$

ψ_w is the heading due to the waves and constitutes the high-frequency part of the total measured heading angle ψ_t. In fact, a second-order, noise-driven filter is usually appropriate for modeling wave-induced motion acting on the ship such as [12][11][13]:

$$H_w(s) = \frac{\psi_w}{\nu} = \frac{K_w.s}{s^2 + 2.\omega_w.\xi_w.s + \omega_w^2} \tag{2.127}$$

With ψ_w the heading waves, ν the noise driver of the wave generation; a zero-mean white noise Gaussian process, K_w the gain of the wave model, ω_w the wave pulsation and ξ_w the wave model damping coefficient. Usually, $\xi_w \approx 0.1$ since we want the wave motion to behave as an undamped system. Concerning the pulsation (equivalently the frequency) of the wave system, it usually varies in between 0.3 rad/s to 1.3 rad/s.

Depending on the pulsation value used in the simulation, the ship's heading will exhibit more or less pronounced oscillations. One can clearly see the influence of the pulsation values of the wave on the total observed heading. If the pulsation is close to its maximal usual value previously specified, the shape of the total heading is:

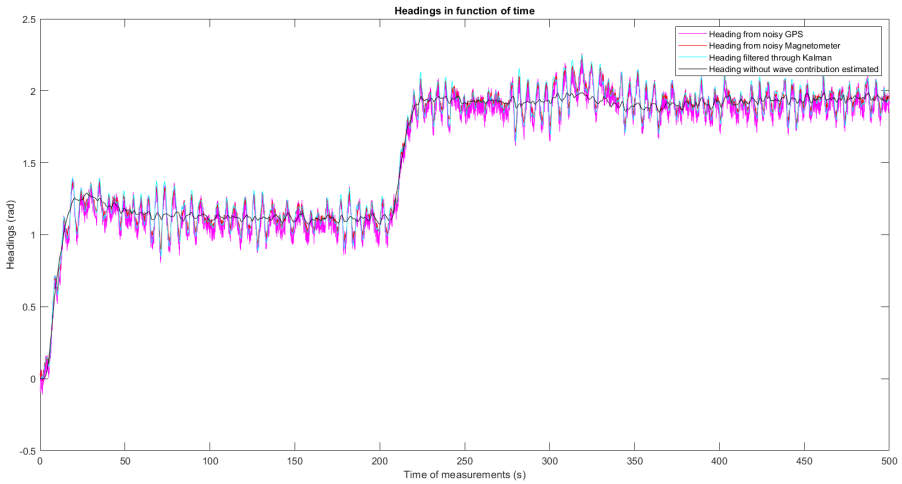

Figure 2.17: Total heading with a high frequency wave perturbation (1.2 rad/s).

Now the same figure, but with a much smaller wave pulsation:

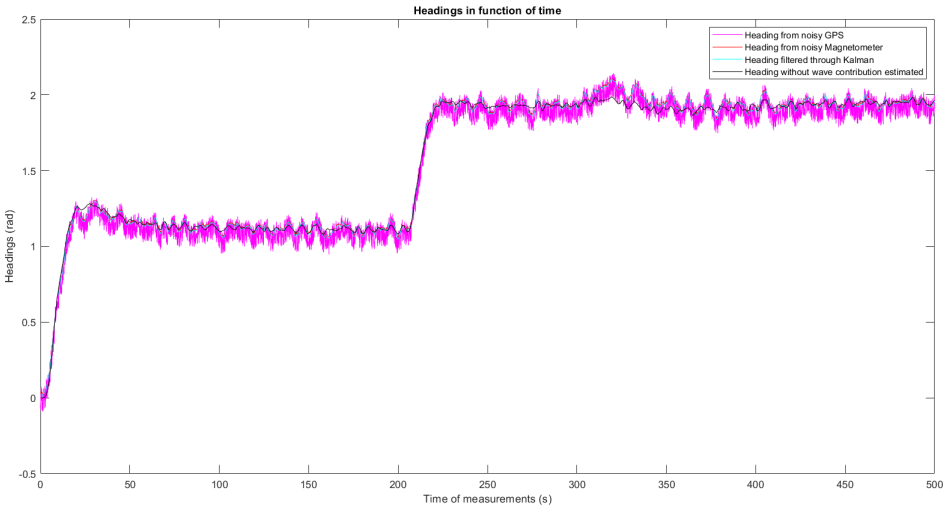

Figure 2.18: Total heading with a mdium to low frequency wave perturbation (0.6 rad/s).

Clearly, the pulsation acts on the total estimated heading. Therefore, the goal is to be able to estimate, in real-time and online, the contributions due to the wave and filters them in order to return the heading angle ψ_s to the Guidance and Control functions, and not the total ψ_t heading that contains the perturbation term ψ_w. One of the main reasons for this is that if the ship is subject to relatively important waves, the total heading without proper wave filtering will then present a highly oscillatory motion. This will imply a frequent use and activation of the controller and actuators to compensate these perturbations but could cause a premature damage of the electrical components or worst; their failure.

The following script is launched in every pre-processes of the simulators and must be run before the launch of any simulation. It contains the data concerning the wave model used in the simulator case: wave data generation.

2.4 Online wave features estimation

The goal is to be able to estimate the features of the wave in real-time online using an identification algorithm. In terms of data flow inside the Navigation function, it corresponds to the block Wave Estimation of the Figure 1.2. For that, an embedded model will be used by the algorithms, and it will be based on the noise-driven filter wave generation presented in the previous section. Note that this is a strong

assumption, since the wave motion may not necessarily acts according to this model in real-life. That is why, for our application, if no waves are spotted on the salted lake or observed during the exploration day, this feature will be deactivated. The entire reasoning will be based on the following reference: [13]. Note however that the algorithms writing and tuning was independently realized and deduced from the in-depth analysis of the article.

2.4.1 Auto-regressive model of the wave-induced heading

As presented, the wave identification is based upon an auto-regressive model (AR). This is a type of regression for temporal series in which the time series is in fact explained by its previous values in time. More precisely, the auto-regressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term which is an imperfectly predictable term. Therefore, the model is in the form of a stochastic difference equation or recurrence relation. This is going to be shown by starting from the wave model:

$$H_w(s) = \frac{\psi_w}{\nu} = \frac{K_w.s}{s^2 + 2.\omega_w.\xi_w.s + \omega_w^2}$$

$$(s^2 + 2.\omega_w.\xi_w.s + \omega_w^2).\psi_w = (K_w.s).\nu \tag{2.128}$$

This equation is in the continuous Laplace domain.

We will discretize it by using the Tustin bilinear transform and the change of variable:

$$s = \frac{2}{T_s}.\frac{1-z}{1+z} = \frac{2}{T_s}.\frac{1-z^{-1}}{1+z^{-1}} \tag{2.129}$$

Where s is the Laplace variable, T_s is the sampling time for the discretization and z the frequency-domain Z-transform variable. Injecting his in equation 2.128 leads to:

$$((\frac{2}{T_s}.\frac{1-z^{-1}}{1+z^{-1}})^2 + 2.\omega_w.\xi_w.(\frac{2}{T_s}.\frac{1-z^{-1}}{1+z^{-1}}) + \omega_w^2).\psi_w = (K_w.(\frac{2}{T_s}.\frac{1-z^{-1}}{1+z^{-1}})).\nu$$

By multiplying the previous equation by $(1+z^{-1})^2$, one has then:

$$(\frac{4}{T_s^2}.(1-z^{-1})^2 + \frac{4.\omega_w.\xi_w}{T_s}.(1-z^{-1}).(1+z^{-1}) + \omega_w^2.(1+z^{-1})^2).\psi_w = (\frac{2.K_w}{T_s}.(1-z^{-1}).(1+z^{-1})).\nu$$

Developing and rearranging the terms gets:

$$((\frac{4}{T_s^2} + \frac{4.\omega_w.\xi_w}{T_s} + \omega_w^2) + (2.\omega_w^2 - \frac{8}{T_s^2}).z^{-1} + (\frac{4}{T_s^2} - \frac{4.\omega_w.\xi_w}{T_s} + \omega_w^2).z^{-2}).\psi_w(k) = (\frac{2.K_w}{T_s} - \frac{2.K_w}{T_s}.z^{-2}).\nu(k) \tag{2.130}$$

By letting:

$$\alpha_0 = \frac{4}{T_s^2} + \frac{4.\omega_w.\xi_w}{T_s} + \omega_w^2$$

$$\alpha_1 = 2.\omega_w^2 - \frac{8}{T_s^2}$$

$$\alpha_2 = \frac{4}{T_s^2} - \frac{4.\omega_w.\xi_w}{T_s} + \omega_w^2$$

$$\beta_0 = \frac{2.K_w}{T_s}$$

Then equation 2.130 can be written as:

$$(\alpha_0 + \alpha_1.z^{-1} + \alpha_2.z^{-2}).\psi_w(k) = \beta_0(1 - z^{-2}).\nu(k) \tag{2.131}$$

Which in turns, can be seen by introducing two function of z^{-1} $A(z^{-1})$ and $B(z^{-1})$, such as the ARMA model representing the continuous equation 2.128:

$$A(z^{-1}).\psi_w(k) = B(z^{-1}).\nu(k) \tag{2.132}$$

NOTA - In the article, however, it was pointed out that, due to normalization, the expression of the righ-hand side of the previous equation $B(z^{-1})$ is in fact $B(z^{-1}) = 1 + \beta_1.z^{-1}.$, whereas to one of the left-hand side is $A(z^{-1}) = 1 + \alpha_1.z^{-1} + \alpha_2.z^{-2}$

Therefore, in order to be aligned with the considerations of the authors, equation 2.131 will be replaced by the slightly different equation:

$$(1 + \alpha_1.z^{-1} + \alpha_2.z^{-2}).\psi_w(k) = (1 + \beta_1.z^{-1}).\nu(k) \tag{2.133}$$

This leads to the following recursive relation between the wave-induced heading and the noisy Gaussian input:

$$\psi_{w_k} = -\alpha_1.\psi_{w_{k-1}} - \alpha_2.\psi_{w_{k-2}} + \beta_1.\nu_{k-1} + \nu_k \tag{2.134}$$

By considering an estimate of the wave-induced heading ψ_{w_k}, namely $\hat{\psi}_{w_k}$, the previous equation rewrites as:

$$\hat{\psi}_{w_k} = -\alpha_1.\hat{\psi}_{w_{k-1}} - \alpha_2.\hat{\psi}_{w_{k-2}} + \beta_1.\nu_{k-1} + \nu_k \tag{2.135}$$

Which in turn, can be written in a matrix form:

$$y_k = \begin{pmatrix} -y_{k-1} & -y_{k-2} \end{pmatrix} . \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} + \nu_{k-1}.\beta_1 + \nu_k \tag{2.136}$$

By noting $y_k \triangleq \hat{\psi}_k \in \mathbb{R}$, $\boldsymbol{\varphi}_{k_w}^T \triangleq \begin{pmatrix} -y_{k-1} & -y_{k-2} \end{pmatrix} \in \mathbb{R}^{1,2}$, $\boldsymbol{\theta}_w^T \triangleq \begin{pmatrix} \alpha_1 & \alpha_2 \end{pmatrix} \in \mathbb{R}^{1,2}$, $\varphi_{n_k} \triangleq \nu_{k-1} \in \mathbb{R}$, $\theta_n \triangleq \beta_1 \in \mathbb{R}$ one can translate equation 2.136 into:

$$y_k = \boldsymbol{\varphi}_{k_w}^T.\boldsymbol{\theta}_w + \varphi_{n_k}.\theta_n + \nu_k \tag{2.137}$$

Which can even be further simplified by noting $\boldsymbol{\varphi}_k^T \triangleq \begin{pmatrix} \boldsymbol{\varphi}_{k_w}^T & \varphi_{n_k} \end{pmatrix} \in \mathbb{R}^{1,3}$ and $\boldsymbol{\theta}^T \triangleq \begin{pmatrix} \boldsymbol{\theta}_w & \theta_n \end{pmatrix} \in \mathbb{R}^{1,3}$ one has in fact:

$$y_k = \boldsymbol{\varphi}_{k_w}^T.\boldsymbol{\theta}_w + \varphi_{n_k}.\theta_n + \nu_k = \begin{pmatrix} \boldsymbol{\varphi}_{k_w}^T & \varphi_{n_k} \end{pmatrix} . \begin{pmatrix} \boldsymbol{\theta}_w \\ \theta_n \end{pmatrix} + \nu_k = \boldsymbol{\varphi}_k^T.\boldsymbol{\theta} + \nu_k \tag{2.138}$$

Here, the vectors $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_w$ gather the parameters to be estimated whereas $\boldsymbol{\varphi}_{k_w}$ and $\boldsymbol{\varphi}_k$ are the information vectors.

2.4.2 Identification & estimation algorithm

In the next section, two algorithms for identification will be presented. The first one is the classical recursive least square algorithm. Whereas the second one is build upon the same algorithm but with an extra stage for an improvement of the calculation speed for online embedded real-time applications.

2.4.2.1 One stage recursive extended least square algorithm

The identified algorithm used in order to estimate the previously introduced parameters and vector is the one based on the Recursive Extended Least Square method, presented extensively in the previous sections. However, the criterion used will not be based here on the regularized least square criterion but on the classical least square criterion recalled here from equation 2.30:

$$J_{LS}(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x}))^T.(\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x})) = \|\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x})\|^2 \tag{2.139}$$

Or even equation of the ReLS 2.33:

$$J_{ReLS}(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x}))^T.\boldsymbol{R}^{-1}.(\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{x})) + (\boldsymbol{x} - \boldsymbol{m})^T.\boldsymbol{P}^{-1}.(\boldsymbol{x} - \boldsymbol{m}) \tag{2.140}$$

But with the matrices $\boldsymbol{R} = \boldsymbol{I}$ and $\boldsymbol{P^{-1}} = \boldsymbol{0}$. However, when applied to our newly established system represented by equation 2.138, the criterion locally is:

$$J_{LS}(\boldsymbol{\theta}) = (y_k - \boldsymbol{\varphi}_k^T.\boldsymbol{\theta})^T.(y_k - \boldsymbol{\varphi}_k^T.\boldsymbol{\theta}) = J(\boldsymbol{\theta}) = \|y_k - \boldsymbol{\varphi}_k^T.\boldsymbol{\theta}\|^2 \tag{2.141}$$

Let us then determine, as in the previous section, the shape of the estimator and associated vectors: least square gain and covariance matrices.

As previously shown, the minimization of the RELS criterion with an adaptation of the notations leads to the following estimates based on the criterion 2.140:

$$\hat{\boldsymbol{x}}_{\boldsymbol{k|k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} + \boldsymbol{K_k}.(\boldsymbol{y_k} - \boldsymbol{G_k}.\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}) \tag{2.142}$$

$$\boldsymbol{K_k} = \boldsymbol{P_{k|k-1}}.\boldsymbol{G_k}^T.(\boldsymbol{G_k}.\boldsymbol{P_{k|k-1}}.\boldsymbol{G_k}^T + \boldsymbol{R_k})^{-1} \tag{2.143}$$

$$\boldsymbol{P_{k|k}} = (\boldsymbol{I} - \boldsymbol{K_k}.\boldsymbol{G_k}).\boldsymbol{P_{k|k-1}} \tag{2.144}$$

By adapting the notations of our current ARMA model based criterion modification, let: $\hat{\boldsymbol{x}}_{\boldsymbol{k|k}} \triangleq \hat{\boldsymbol{\theta}}_k$, $\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} \triangleq \hat{\boldsymbol{\theta}}_{k-1}$, $\boldsymbol{K_k} \triangleq \boldsymbol{L_k}$, $\boldsymbol{G_k} \triangleq \boldsymbol{\varphi}_k^T$ and $\boldsymbol{P_{k|k-1}} \triangleq \boldsymbol{P}_{k-1}$. This leads to the following set of equations:

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{L_k}.(y_k - \boldsymbol{\varphi}_k^T.\hat{\boldsymbol{\theta}}_{k-1}) \tag{2.145}$$

$$\boldsymbol{L_k} = \boldsymbol{P}_{k-1}.\boldsymbol{\varphi}_k.(\boldsymbol{\varphi}_k^T.\boldsymbol{P}_{k-1}.\boldsymbol{\varphi}_k + \boldsymbol{I})^{-1} = \frac{\boldsymbol{P}_{k-1}.\boldsymbol{\varphi}_k}{\boldsymbol{\varphi}_k^T.\boldsymbol{P}_{k-1}.\boldsymbol{\varphi}_k + \boldsymbol{I}} \tag{2.146}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{L_k}.\boldsymbol{\varphi}_k^T).\boldsymbol{P}_{k-1} \tag{2.147}$$

Note however that in equation 2.138, the information vector $\boldsymbol{\varphi}_{k_w}^T$ containing $\varphi_{n_k} = \nu_{k-1}$ inside of $\boldsymbol{\varphi}_k$ is not directly measurable. Therefore, it is impossible to retrieve a direct measure from $\hat{\boldsymbol{\theta}}_k$. To avoid this problem, let us take a look to the equation:

$$y_k = \boldsymbol{\varphi}_k^T.\boldsymbol{\theta} + \nu_k$$

And note that therefore:

$$\nu_k = y_k - \boldsymbol{\varphi}_k^T.\boldsymbol{\theta} \tag{2.148}$$

Based on that, an estimate of the current noise input term is:

$$\hat{\nu}_k = y_k - \hat{\boldsymbol{\varphi}}_k^T.\hat{\boldsymbol{\theta}} \tag{2.149}$$

We then obtain the final ReLS algorithm equations for the estimation of the wave-induced mode:

Equations of the ReLS algorithms for the wave-induced model identification

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{L_k}.(y_k - \hat{\boldsymbol{\varphi}}_k^T.\hat{\boldsymbol{\theta}}_{k-1}) \tag{2.150}$$

$$\boldsymbol{L_k} = \frac{\boldsymbol{P}_{k-1}.\hat{\boldsymbol{\varphi}}_{\boldsymbol{k}}}{\hat{\boldsymbol{\varphi}}_k^T.\boldsymbol{P}_{k-1}.\hat{\boldsymbol{\varphi}}_k + \boldsymbol{I}} \tag{2.151}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{L_k}.\hat{\boldsymbol{\varphi}}_k^T).\boldsymbol{P}_{k-1} \tag{2.152}$$

$$\hat{\boldsymbol{\varphi}}_k = \begin{pmatrix} \boldsymbol{\varphi}_{k_w}^T \\ \hat{\varphi}_{n_k} \end{pmatrix} \tag{2.153}$$

$$\hat{\varphi}_{n_k} = \hat{\nu}_{k-1} \tag{2.154}$$

$$\boldsymbol{\varphi}_{k_w} = \begin{pmatrix} -y_{k-1} \\ -y_{k-2} \end{pmatrix} \tag{2.155}$$

$$\hat{\nu}_k = y_k - \hat{\boldsymbol{\varphi}}_k^T.\hat{\boldsymbol{\theta}} \tag{2.156}$$

The formal algorithm won't be described since the implemented one correspond to a two-staged recursive least square algorithm, presented in the following section.

2.4.2.2 Two-stage recursive extended least square algorithm

In order to increase the computational efficiency, the authors in [13] demonstrated that the use of a two staged RELS is better than the first usual one. Initially, the goal for this project was to study and implement classical solutions in the domains in order for discovering the mathematical aspects and notions behind ship Guidance & Control. However, the goal of this special article seemed very aligned to onboard efficiency implementations, that is why this algorithm was implemented. For building the two-stage algorithm, one shall define two intermediate variables such that:

$$y_{1_k} = y_k - \varphi_{n_k}.\theta_n \tag{2.157}$$

$$y_{2_k} = y_k - \varphi_{k_w}^T.\boldsymbol{\theta}_w \tag{2.158}$$

Based on the expression of y_k from equation 2.138, these equations become:

$$y_{1_k} = \varphi_{k_w}^T.\boldsymbol{\theta}_w + \nu_k \tag{2.159}$$

$$y_{2_k} = \varphi_{n_k}.\theta_n + \nu_k \tag{2.160}$$

Again, the goal being to minimize the error, or more exactly here, the noise generated input of the wave model ν_k, we can define two cost functions:

$$J_2(\boldsymbol{\theta}) = \sum_{i=1}^{N} \|y_{1_i} - \boldsymbol{\varphi}_{i_w}^T.\boldsymbol{\theta}_w\|^2 \tag{2.161}$$

$$J_3(\boldsymbol{\theta}) = \sum_{i=1}^{N} \|y_{2_i} - \varphi_{n_i}.\theta_n\|^2 \tag{2.162}$$

Which are used to derive the estimation equations as previously done in the prior section. For the second cost function:

$$\hat{\boldsymbol{\theta}}_{w_k} = \hat{\boldsymbol{\theta}}_{w_{k-1}} + \boldsymbol{L}_{\boldsymbol{w_k}}.(y_{1_k} - \boldsymbol{\varphi}_{k_w}^T.\hat{\boldsymbol{\theta}}_{w_{k-1}}) \tag{2.163}$$

$$\boldsymbol{L}_{\boldsymbol{w_k}} = \boldsymbol{P}_{w_{k-1}}.\boldsymbol{\varphi}_{k_w}.(\boldsymbol{\varphi}_{k_w}^T.\boldsymbol{P}_{w_{k-1}}.\boldsymbol{\varphi}_{k_w} + \boldsymbol{I})^{-1} = \frac{\boldsymbol{P}_{w_{k-1}}.\boldsymbol{\varphi}_{k_w}}{\boldsymbol{\varphi}_{k_w}^T.\boldsymbol{P}_{w_{k-1}}.\boldsymbol{\varphi}_{k_w} + \boldsymbol{I}} \tag{2.164}$$

$$\boldsymbol{P}_{w_k} = (\boldsymbol{I} - \boldsymbol{L}_{\boldsymbol{w_k}}.\boldsymbol{\varphi}_{k_w}^T).\boldsymbol{P}_{w_{k-1}} \tag{2.165}$$

And for the third one:

$$\hat{\theta}_{n_k} = \hat{\theta}_{n_{k-1}} + L_{n_k}.(y_{2_k} - \varphi_{n_k}.\hat{\theta}_{n_{k-1}}) \tag{2.166}$$

$$L_{n_k} = \frac{P_{n_{k-1}}.\varphi_{n_k}}{P_{n_{k-1}}.\varphi_{n_k}^2 + 1} \tag{2.167}$$

$$P_{n_k} = (1 - L_{n_k}.\varphi_{n_k}).P_{n_{k-1}} \tag{2.168}$$

Eventually, one sees that the injection of equations 2.159 and 2.160 into respectively equations 2.163 and 2.166 leads to:

$$\hat{\boldsymbol{\theta}}_{w_k} = \hat{\boldsymbol{\theta}}_{w_{k-1}} + \boldsymbol{L}_{\boldsymbol{w_k}}.(y_k - \varphi_{n_k}.\theta_n - \boldsymbol{\varphi}_{k_w}^T.\hat{\boldsymbol{\theta}}_{w_{k-1}}) \tag{2.169}$$

$$\hat{\theta}_{n_k} = \hat{\theta}_{n_{k-1}} + L_{n_k}.(y_k - \boldsymbol{\varphi}_{k_w}^T.\boldsymbol{\theta}_w - \varphi_{n_k}.\hat{\theta}_{n_{k-1}}) \tag{2.170}$$

Similarly as in the previous section, the unmeasured term ν_{k-1} inside the term φ_{n_k} as well as the unknown vectors to be identified $\boldsymbol{\theta}_w$ and θ_n cannot allow the two previous equations to directly give estimates $\hat{\boldsymbol{\theta}}_{w_k}$ and $\hat{\theta}_{n_k}$. Again, the use of the equation:

$$\hat{\nu}_k = y_k - \hat{\boldsymbol{\varphi}}_k^T.\hat{\boldsymbol{\theta}} \tag{2.171}$$

allows to estimate the noise input. Finally, the equations for the two-stage ReLS algorithm are the following:

Equations of the two stages ReLS algorithms

$$\hat{\boldsymbol{\theta}}_{w_k} = \hat{\boldsymbol{\theta}}_{w_{k-1}} + \boldsymbol{L_{w_k}}.(y_k - \hat{\varphi}_{n_k}.\hat{\theta}_n - \boldsymbol{\varphi}_{k_w}^T.\hat{\boldsymbol{\theta}}_{w_{k-1}}) \tag{2.172}$$

$$\hat{\theta}_{n_k} = \hat{\theta}_{n_{k-1}} + L_{n_k}.(y_k - \hat{\boldsymbol{\varphi}}_{k_w}^T.\hat{\boldsymbol{\theta}}_w - \varphi_{n_k}.\hat{\theta}_{n_{k-1}}) \tag{2.173}$$

$$\boldsymbol{L_{w_k}} = \frac{\boldsymbol{P}_{w_{k-1}}.\boldsymbol{\varphi}_{k_w}}{\boldsymbol{\varphi}_{k_w}^T.\boldsymbol{P}_{w_{k-1}}.\boldsymbol{\varphi}_{k_w} + \boldsymbol{I}} \tag{2.174}$$

$$\boldsymbol{P}_{w_k} = (\boldsymbol{I} - \boldsymbol{L_{w_k}}.\boldsymbol{\varphi}_{k_w}^T).\boldsymbol{P}_{w_{k-1}} \tag{2.175}$$

$$L_{n_k} = \frac{P_{n_{k-1}}.\hat{\varphi}_{n_k}}{P_{n_{k-1}}.\hat{\varphi}_{n_k}^2 + 1} \tag{2.176}$$

$$P_{n_k} = (1 - L_{n_k}.\hat{\varphi}_{n_k}).P_{n_{k-1}} \tag{2.177}$$

$$\hat{\nu}_k = y_k - \hat{\boldsymbol{\varphi}}_k^T.\hat{\boldsymbol{\theta}} \tag{2.178}$$

$$\boldsymbol{\varphi}_{k_w} = \begin{pmatrix} -y_{k-1} \\ -y_{k-2} \end{pmatrix} \tag{2.179}$$

$$\hat{\varphi}_{n_k} = \hat{\nu}_{k-1} \tag{2.180}$$

2.4.2.3 Wave features online calculation

The last step is then to retrieve, based on the estimates calculated through the previous algorithm, the characteristics of the wave. That is, the pulsation ω_w and the damping ξ_w. If we suppose that the estimate from equation 2.172 is available, namely $\hat{\boldsymbol{\theta}}_{w_k}$, then one has:

$$\hat{\boldsymbol{\theta}}_{w_k} = \begin{pmatrix} \hat{\alpha}_{1_k} \\ \hat{\alpha}_{2_k} \end{pmatrix}$$

Thus, we can use the estimated coefficients at each step time that are involved in the ARMA model equation 2.132. As stated in the literature in digital control, it is known that the mapping from the z-plane frequency domain to the continuous one represented by the s variable from Laplace space is given by:

$$z = e^{\delta t.s} \tag{2.181}$$

With the sampling period δt. Note that if a function $F(s)$ has a pole at $s = a$, then the function $F(z)$ has one at $z = e^{\delta t.a}$. Therefore, the goal here is to use the estimates to calculate the poles of the function $A(z)$ described in the ARMA model. Using the previous property will lead us to the poles of $A(s)$, which is in fact equals to $s^2 + 2.\omega_w.\xi_w.s + \omega_w^2$. This will directly lead us to the estimation of the pulsation and the damping coefficients of the wave, for each time step. Again, as stated, using the coefficients of the estimate, one can build $A(z)$ since:

$$A(z) = 1 + \alpha_1.z^{-1} + \alpha_2.z^{-2}$$

Hence here:

$$\hat{A}(z) = 1 + \hat{\alpha}_{1_k}.z^{-1} + \hat{\alpha}_{2_k}.z^{-2}$$

Therefore, by defining $\Delta = \hat{\alpha}_{1_k}^2 - 4.\hat{\alpha}_{2_k}$, one can easily solve this second degree equation. This leads, in function of the sign of Δ, to three possible solutions. For example, the first ones when $\Delta > 0$:

$$z_{1,2_k} = \frac{-\hat{\alpha}_{1_k} \pm \sqrt{\hat{\alpha}_{1_k}^2 - 4.\hat{\alpha}_{2_k}}}{2} \tag{2.182}$$

Therefore, based on equation 2.181, we have:

$$s_{1,2_k} = \frac{\log(z_{1,2_k})}{\delta t_k} \tag{2.183}$$

Using the root of the continuous function $A(s)$ leads effectively to:

$$\hat{\omega}_{w_k} = |s_{1,2_k}| \tag{2.184}$$

And using the estimate of $\hat{\omega}_{w_k}$ allows finding an estimate of the damping coefficient $\hat{\xi}_{w_k}$.

Indeed, we know that solving the equation:

$$A(s) = 0 \Leftrightarrow s^2 + 2.\omega_w.\xi_w.s + \omega_w^2 = 0$$

Again, by defining the delta of the equation as $\Delta_s = 4.\omega_w.(\xi_w^2 - 1)$, the solutions of this previous equation vary in function of its sign. In this project, we stated that we will impose a very important oscillatory motion for the wave, which means that we want a very low damping lower than 1. This discriminant will therefore be negative, leading to the following usual solutions:

$$s_{1,2} = -\xi_w.\omega_w \pm j.\omega_w.\sqrt{1 - \xi_w} \tag{2.185}$$

One can then see that the use of the calculation of the module of the previous complex number will lead to equation 2.184. Eventually, if one gets to establish a routine allowing to identify the real and imaginary parts of the complex number $s_{1,2}$, denoting R its real one, we can note that:

$$R = -\omega_w.\xi_w$$

Therefore leading to:

$$\hat{\xi}_{w_k} = \frac{-R}{\hat{\omega}_{w_k}} \tag{2.186}$$

2.4.2.4 Algorithm implementation

Therefore, using the previously defined equations in the prior sections, it is possible to construct the online algorithm retrievable written in Matlab through the following repository: online identification algorithm.

The implemented algorithm is detailed as follows:

Algorithm 3 Two-Stage Recursive Extended Least Squares and online wave feature estimation

Initialization:
$\hat{\boldsymbol{\theta}}_{w_0} = \begin{pmatrix}\hat{\alpha}_{1_0} & \hat{\alpha}_{2_0}\end{pmatrix}^T, \quad \hat{\theta}_{n_0} = \gamma_w$
$\boldsymbol{P}_{w_0} = \gamma_w \boldsymbol{I}_2, \quad P_{n_0} = \gamma_n$
$y_{-1} = 0, \quad y_{-2} = 0, \quad \hat{\nu}_{-1} = 0$
for $k \in [\![1, N]\!]$ **do**
  Measurement acquisition:
    $y_k \leftarrow \hat{\psi}_{w_k}$
  Information vectors construction:
    $\boldsymbol{\varphi}_{k_w} = \begin{pmatrix}-y_{k-1} \\ -y_{k-2}\end{pmatrix}$
    $\hat{\varphi}_{n_k} = \hat{\nu}_{k-1}$
  Wave AR parameters:
    $\boldsymbol{L}_{w_k} = \frac{\boldsymbol{P}_{w_{k-1}}\boldsymbol{\varphi}_{k_w}}{\boldsymbol{\varphi}_{k_w}^T \boldsymbol{P}_{w_{k-1}}\boldsymbol{\varphi}_{k_w}+1}$
    $\hat{\boldsymbol{\theta}}_{w_k} = \hat{\boldsymbol{\theta}}_{w_{k-1}} + \boldsymbol{L}_{w_k}\left(y_k - \hat{\varphi}_{n_k}\hat{\theta}_{n_{k-1}} - \boldsymbol{\varphi}_{k_w}^T \hat{\boldsymbol{\theta}}_{w_{k-1}}\right)$
    $\boldsymbol{P}_{w_k} = (\boldsymbol{I} - \boldsymbol{L}_{w_k}\boldsymbol{\varphi}_{k_w}^T)\boldsymbol{P}_{w_{k-1}}$
  Noise model parameters:
    $L_{n_k} = \frac{P_{n_{k-1}}\hat{\varphi}_{n_k}}{P_{n_{k-1}}\hat{\varphi}_{n_k}^2+1}$
    $\hat{\theta}_{n_k} = \hat{\theta}_{n_{k-1}} + L_{n_k}\left(y_k - \boldsymbol{\varphi}_{k_w}^T \hat{\boldsymbol{\theta}}_{w_{k-1}} - \hat{\varphi}_{n_k}\hat{\theta}_{n_{k-1}}\right)$
    $P_{n_k} = (1 - L_{n_k}\hat{\varphi}_{n_k})P_{n_{k-1}}$
  White-noise estimation:
    $\hat{\boldsymbol{\varphi}}_k = \begin{pmatrix}\boldsymbol{\varphi}_{k_w} \\ \hat{\varphi}_{n_k}\end{pmatrix}$
    $\hat{\boldsymbol{\theta}}_k = \begin{pmatrix}\hat{\boldsymbol{\theta}}_{w_k} \\ \hat{\theta}_{n_k}\end{pmatrix}$
    $\hat{\nu}_k = y_k - \hat{\boldsymbol{\varphi}}_k^T \hat{\boldsymbol{\theta}}_k$
  Wave feature estimation:
    $\hat{\alpha}_{1_k} \leftarrow \hat{\boldsymbol{\theta}}_{w_k}(1), \quad \hat{\alpha}_{2_k} \leftarrow \hat{\boldsymbol{\theta}}_{w_k}(2)$
    $\Delta_k = \hat{\alpha}_{1_k}^2 - 4\hat{\alpha}_{2_k}$
    $z_{1,2_k} = \frac{-\hat{\alpha}_{1_k} \pm \sqrt{\Delta_k}}{2}$
    $s_{1,2_k} = \frac{1}{\delta t_k}.\log(z_{1,2_k})$
    $\hat{\omega}_{w_k} = |s_{1,2_k}|$
    $\hat{\xi}_{w_k} = \frac{-\mathrm{Re}(s_{1,2_k})}{\hat{\omega}_{w_k}}$
  Measurement history update:
    $y_{k-2} \leftarrow y_{k-1}, \quad y_{k-1} \leftarrow y_k$
end for

This algorithm allows to estimate, in real time, the wave parameters based on the dynamical model. After a certain time of convergence, one sees that the value of the estimated pulsation ω_w converges to the initial input value settled in the wave model environment. A more detailed analysis of the performance of this algorithm will be done in future studies, however, the following results were obtained for a case where the pulsation of the wave was of higher frequency that the one of the behavior of the ship:

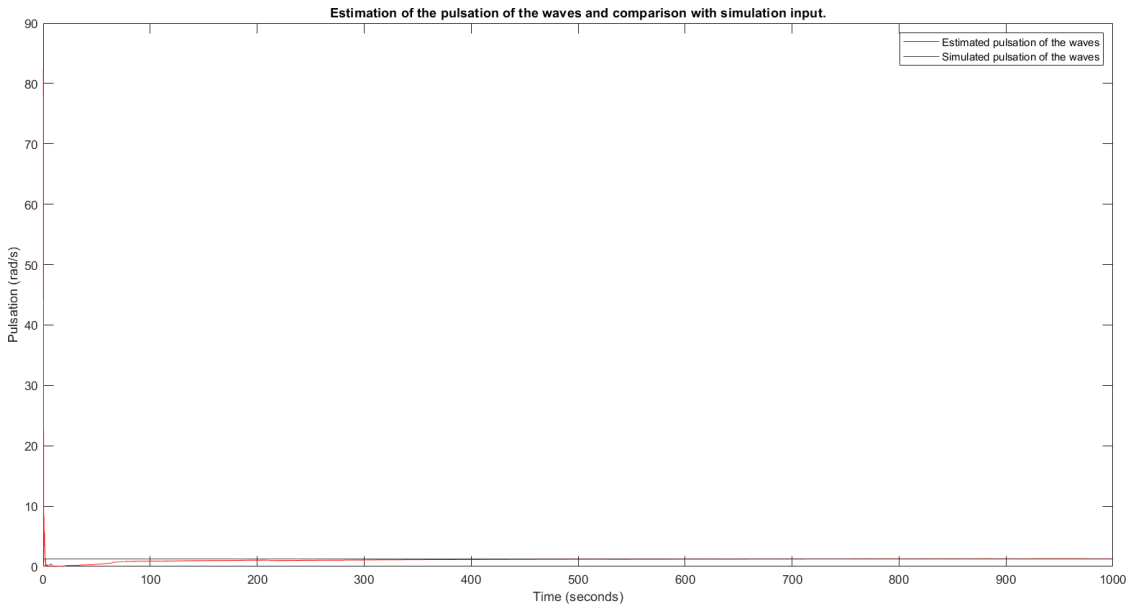

Figure 2.19: Estimation of the pulsation of the wave during a simulation (wave pulsation input of 1.2 rad/s).

In a very general way, one sees that the estimate is very poor for the first time steps and very high. After some steps, the filter begins to converge.

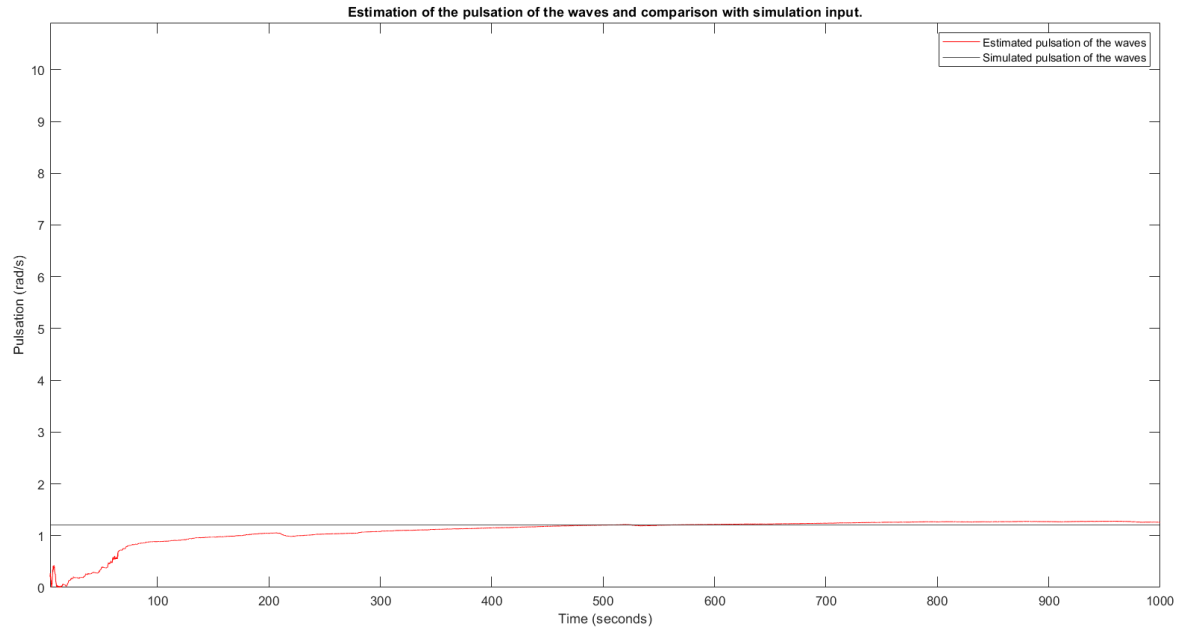

Figure 2.20: Estimation of the pulsation of the wave during a simulation (wave pulsation input of 1.2 rad/s).

Indeed, one sees that the estimate (in red) converges to its simulation value (in black) near the very beginning of the simulation. The time of crossing of the value of 1.2 rad/s seems to occur at around 500 seconds:

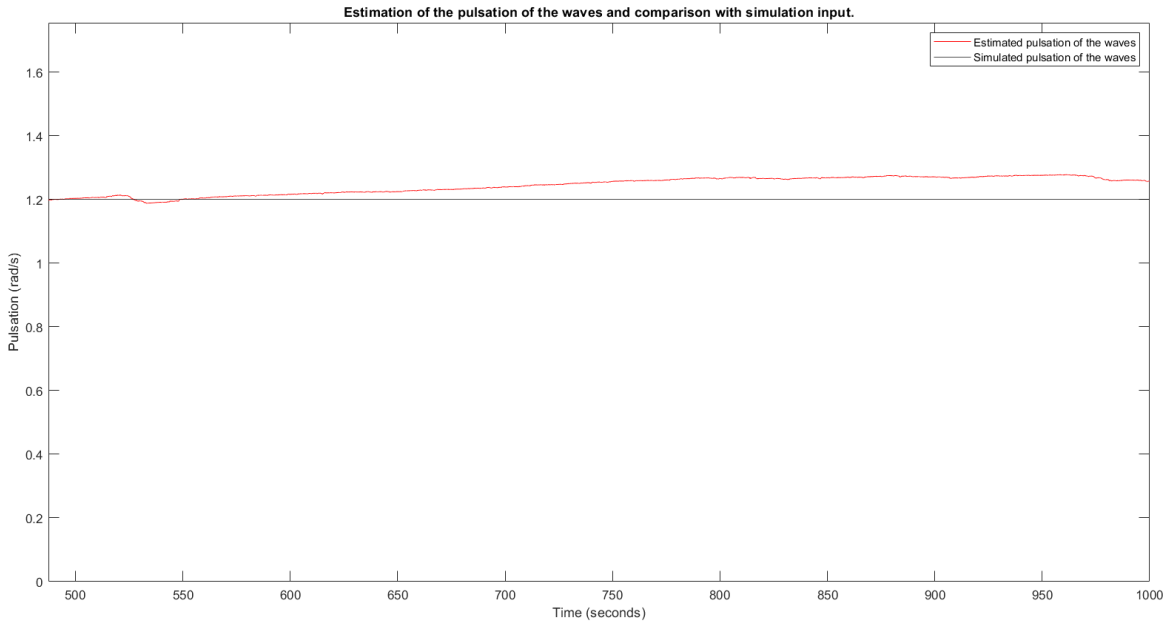

Figure 2.21: Estimation of the pulsation of the wave during a simulation (wave pulsation input of 1.2 rad/s).

By further zooming into the figure, it can be observed that the estimated value converges smoothly and stably toward the expected value. This allows us to conclude that, in the presence of high-frequency perturbations, the identification algorithm exhibits satisfactory performance with respect to pulsation estimation. The analysis of the damping coefficient estimation is intentionally omitted in this initial discussion. The reason is that the choice of a very low damping value appears to be a governing factor. Indeed, preliminary performance analyses indicate that even when the simulator input specifies a damping coefficient of, for instance, 0.4, while the value used in the wave observer is tuned to 0.1, the estimation of the wave profile and the induced motion remains acceptable. A more detailed investigation of this behavior will be addressed in future studies.

2.5 Heading of the ship estimation without perturbations

This last section will close this version of the Navigation function justification file. Having now a filtered and fused estimate of the heading of the ship, as well as an estimation of certain features of the waves (pulsation and damping), we can determine the actual real contribution to the heading without the wave action. This will concern the last block of the Navigation chain presented in Figure 1.2, namely the Heading Ship Estimation block.

As stated before in equation 2.126, the heading can be decomposed such that:

$$\psi_t = \psi_s + \psi_w \tag{2.187}$$

The goal here is to determine an estimate of ψ_w that will allow us to isolate ψ_s and send it to the next Guidance and Control functions. For that we will have to build and use an estimator. An estimator is an algorithm that allows to construct some parameters that are not directly observable or measurable, by deducing them using their mathematical relationship with variables that are actually measurable. In the literature in Automatic & Control, several kind of observers are mainly used, such as the Luenberger observer. Here, we decided to implement a Kalman Filter observer based on the work done and presented in [12]. Again, as for the previous used article on wave identification, the algorithms and mathematical justification, absent from the article, will be derived here.

2.5.1 Observer design and models

As stated, the Kalman Filter can be used to create an observer. As it was shown in the dedicated sections, the Kalman Filter is composed by a prediction step and a measurement update one. The prediction relies

upon a mathematical mode, which we are going to detail here for the purpose of wave heading estimation. We are starting from the assumption that the wave-induced heading behaves as a second-order system driven by input noise, as in the previous section, leading to the model:

$$H_w(s) = \frac{\psi_w}{\nu} = \frac{K_w.s}{s^2 + 2.\omega_w.\xi_w.s + \omega_w^2} \tag{2.188}$$

$$(s^2 + 2.\omega_w.\xi_w.s + \omega_w^2).\psi_w = (K_w.s).\nu \tag{2.189}$$

Which is equivalent to the time domain differential equation:

$$\ddot{\psi_w}(t) + 2.\omega_w.\xi_w.\dot{\psi_w}(t) + \omega_w^2.\psi_w(t) = K_w.\dot{\nu}(t) \tag{2.190}$$

Then we define a physically interpretable variable $\dot{\xi} = \psi_w$. Additionally, let $x_1 = \xi$ and $x_2 = \psi_w$. In this case: $\dot{x}_1 = \dot{\xi}$ and $\dot{x}_2 = \dot{\psi}_w$. By denoting $\dot{\nu}$ as the usual input variable u, one has the following system of equations:

$$\dot{x}_1 = x_2 \tag{2.191}$$

$$\dot{x}_2 = -\omega_w^2.x_1 - 2.\omega_w.\xi_w.x_2 + K_w.u \tag{2.192}$$

By developing the previous equation, one has:

$$\dot{x}_2 = -\omega_w^2.x_1 - 2.\omega_w.\xi_w.x_2 + K_w.u$$

$$\dot{\psi}_w = -\omega_w^2.\xi - 2.\omega_w.\xi_w.\psi_w + K_w.u$$

By differentiating it with respect to time, it leads to the equation 2.190:

$$\ddot{\psi}_w = -\omega_w^2.\psi_w - 2.\omega_w.\xi_w.\dot{\psi}_w + K_w.\dot{u}$$

Therefore, the final system in continuous time space is the following:

$$\dot{\xi} = \psi_w \tag{2.193}$$

$$\dot{\psi_w} = -\omega_w^2.\xi - 2.\omega_w.\xi_w.\psi_w + K_w.u \tag{2.194}$$

Recall that as expressed in previous sections, this model has to be in a tractable form for on-board software computer. There is a need to discretize it. The usual Euler discretization at order 1 has also been used here leading to the following equations:

$$\xi_k = \xi_{k-1} + \delta t_{k-1}.\psi_{w_{k-1}} \tag{2.195}$$

$$\psi_{w_k} = \psi_{w_{k-1}} + \delta t_{k-1}.(-\omega_w^2.\xi_{k-1} - 2.\omega_w.\xi_w.\psi_{w_{k-1}}) + \delta t_{k-1}.K_w.u_{k-1} \tag{2.196}$$

NOTA - Here, the input u_{k-1} which in fact corresponds to the time derivative of ν_{k-1}, will be treated in the KF as a noise of the process model, namely $\delta\psi_{w_k}$ such that $\psi_{w_k} = \psi_{w_{k-1}} + \delta t_{k-1}.(-\omega_w^2.\xi_{k-1} - 2.\omega_w.\xi_w.\psi_{w_{k-1}}) + \delta t_{k-1}.K_w.\delta\psi_{w_{k-1}}$. On the other hand, it seems that it is classical to consider the gain of the wave transfer function K_w to be equal to ω_w^2. ***Also note that here the parameters ω_w and the damping coefficient ξ_w are the ones estimated by the identification algorithm described in the previous section.***

However, as a purpose of exploration, a second discretization technique has also been explored and applied, which is the one designated as being the exact discretization using the exponential of matrices.

Indeed, an exact discretization form can be found by starting from the general continuous state-space scalar model equation with perturbation such that:

$$\dot{x}(t) = A.x(t) + B.u(t) + E.w(t) \tag{2.197}$$

We want to solve this equation in an interval $\delta t_{k-1} = t_k - t_{k-1}$. To do so, we define the integrating factor $e^{-A.t}$ such that:

$$\frac{d}{dt}(e^{-A.t}.x(t)) = -A.e^{-A.t}.x(t) + e^{-A.t}.\dot{x}(t) \tag{2.198}$$

Then, by multiplying equation 2.197 by the integrating factor:

$$e^{-A.t}.\dot{x}(t) = e^{-A.t}.A.x(t) + e^{-A.t}.B.u(t) + e^{-A.t}.E.w(t)$$

Rearranging the terms leads to:

$$\frac{d}{dt}(e^{-A.t}.x(t)) = e^{-A.t}.B.u(t) + e^{-A.t}.E.w(t) \tag{2.199}$$

Multiplying the previous equation by the time differential dt and integrating on the specified interval leads to:

$$\int_{t_{k-1}}^{t_k} d(e^{-A.t}.x(t)) = \int_{t_{k-1}}^{t_k} e^{-A.t}.B.u(t).dt + \int_{t_{k-1}}^{t_k} e^{-A.t}.E.w(t).dt \tag{2.200}$$

Equivalently:

$$[e^{-A.t}.x(t)]_{t_{k-1}}^{t_k} = e^{-A.t_k}.x(t_k) - e^{-A.t_{k-1}}.x(t_{k-1}) = \int_{t_{k-1}}^{t_k} e^{-A.t}.B.u(t).dt + \int_{t_{k-1}}^{t_k} e^{-A.t}.E.w(t).dt \tag{2.201}$$

Which can be solved for $x(t_k)$, leading to:

$$x(t_k) = e^{A.(t_k - t_{k-1})}.x(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{A.(t_k-t)}.B.u(t).dt + \int_{t_{k-1}}^{t_k} e^{A.(t_k-t)}.E.w(t).dt \tag{2.202}$$

The similar result can be obtained for the more general matrix case where the initial continuous system is of the form:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}.\boldsymbol{x}(t) + \boldsymbol{B}.\boldsymbol{u}(t) + \boldsymbol{E}.\boldsymbol{w}(t)$$

And by denoting that the matrix of exponential for a square matrix $\boldsymbol{A}$ is:

$$\boldsymbol{e}^{\boldsymbol{A}} = \sum_{k=0}^{+\infty} \frac{1}{k!}.\boldsymbol{A}^k \tag{2.203}$$

By applying the same development than for the scalar case, the final solution will lead to:

$$\boldsymbol{x}(t_k) = \boldsymbol{e}^{A.\delta t_{k-1}}.\boldsymbol{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{B}.\boldsymbol{u}(t).dt + \int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{E}.\boldsymbol{w}(t).dt \tag{2.204}$$

By denoting:

$$\boldsymbol{x}(t_k) = \boldsymbol{x}_k, \quad \boldsymbol{\Phi}_k = \boldsymbol{e}^{A.\delta t_{k-1}}, \quad \boldsymbol{\Delta}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{B}.\boldsymbol{u}(t).dt, \quad \boldsymbol{\Gamma}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{E}.\boldsymbol{w}(t).dt$$

Then equation 2.204 can be put under the form:

$$\boldsymbol{x}_k = \boldsymbol{\Phi}_k.\boldsymbol{x}_{k-1} + \boldsymbol{\Delta}_k + \boldsymbol{\Gamma}_k \tag{2.205}$$

Under the assumption of the zero order hold (ZOH), the vectors $\boldsymbol{u}(t)$ and $\boldsymbol{w}(t)$ are considered constant over the time interval considered. Therefore, they can be put outside of the integrals:

$$\boldsymbol{\Delta}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{B}.\boldsymbol{u}(t).dt = (\int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{B}.dt).\boldsymbol{u}_k = \boldsymbol{\delta}_k.\boldsymbol{u}_k$$

$$\boldsymbol{\Gamma}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{E}.\boldsymbol{w}(t).dt = (\int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{E}.dt).\boldsymbol{w}_k = \boldsymbol{\gamma}_k.\boldsymbol{w}_k$$

We can observe that, by letting $\sigma = t_k - t$ that:

$$\boldsymbol{\delta}_k = \int_{t_{k-1}}^{t_k} \boldsymbol{e}^{A.(t_k-t)}.\boldsymbol{B}.dt = \int_0^{\delta t_{k-1}} \boldsymbol{e}^{A.\sigma}.\boldsymbol{B}.d\sigma \tag{2.206}$$

Therefore, if $\boldsymbol{A}$ is invertible, then:

$$\boldsymbol{\delta}_k = (\int_0^{\delta t_{k-1}} \boldsymbol{A^{-1}}.\frac{d}{d\sigma}\boldsymbol{e}^{A.\sigma}.d\sigma.)\boldsymbol{B} = \boldsymbol{A^{-1}}.(\int_0^{\delta t_{k-1}} \frac{d}{d\sigma}\boldsymbol{e}^{A.\sigma}.d\sigma.)\boldsymbol{B} = \boldsymbol{A^{-1}}.(\boldsymbol{e}^{A.\delta t} - \boldsymbol{I}).\boldsymbol{B} \quad (2.207)$$

By definition, as previously stated:

$$\boldsymbol{e}^{\boldsymbol{A}} = \sum_{k=0}^{+\infty} \frac{1}{k!}.\boldsymbol{A}^k$$

Hence:

$$\boldsymbol{e}^{A.\delta t_{k-1}} = \sum_{k=0}^{+\infty} \frac{1}{k!}.(\boldsymbol{A}.\delta t_{k-1})^k = \Phi_k \quad (2.208)$$

Hence, equation 2.208 leads to:

$$\boldsymbol{\Delta}_k = \boldsymbol{A^{-1}}.(\boldsymbol{e}^{A.\delta t} - \boldsymbol{I}).\boldsymbol{B}.\boldsymbol{u}_k = \boldsymbol{\delta}_k.\boldsymbol{u}_k \quad (2.209)$$

The same reasoning will also lead to:

$$\boldsymbol{\Gamma}_k = \boldsymbol{A^{-1}}.(\boldsymbol{e}^{A.\delta t} - \boldsymbol{I}).\boldsymbol{E}.\boldsymbol{w}_k = \boldsymbol{\gamma}_k.\boldsymbol{w}_k \quad (2.210)$$

Therefore, equation 2.205 can be written as:

$$\boldsymbol{x}_k = \boldsymbol{\Phi}_k.\boldsymbol{x}_{k-1} + \boldsymbol{\delta}_k.\boldsymbol{u_k} + \boldsymbol{\gamma}_k.\boldsymbol{w_k} \quad (2.211)$$

Which is said to be the exact discrete form of the continuous general vector-matrix state-space $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}.\boldsymbol{x}(t) + \boldsymbol{B}.\boldsymbol{u}(t) + \boldsymbol{E}.\boldsymbol{w}(t)$.

2.5.1.1 Mathematical model of the observer and predictor

Concerning the **prediction steps of the filter**, equations 2.195 and 2.196 are the first equations that will be used by the embedded KF observer to estimate the heading without perturbations. To complete the mathematical model, we will use the equations describing the low-frequency motion of the ship, namely the equations coming from Nomoto's first order model recalled here for convenience:

$$\psi_{s_k} = \psi_{s_{k-1}} + \delta t_{k-1}.r_{k-1} \quad (2.212)$$

$$r_k = (1 - \frac{\delta t_{k-1}}{T}).r_{k-1} + \delta t_{k-1}.\frac{K}{T}.\delta_{k-1} + \delta t_{k-1}.b_{k-1} + \delta t_{k-1}.\delta r_{k-1} \quad (2.213)$$

$$b_k = b_{k-1} + \delta t_{k-1}.\delta b_{k-1} \quad (2.214)$$

Therefore, as in the section describing the implementation of the KF for the estimation and sensor fusion, we can define the state vector used in this observer here:

$$\boldsymbol{x_k} = \begin{pmatrix} \xi_k & \psi_{w_k} & \psi_{s_k} & r_k & b_k \end{pmatrix}^T \quad (2.215)$$

With $\boldsymbol{x_k} \in \mathbb{R}^{5,1}$. For relevant notations and parameters definition, refer to the KF/EKF sections. The state-space process model for the prediction will here take the form of:

$$\boldsymbol{x_k} = \boldsymbol{A_k}.\boldsymbol{x_{k-1}} + \boldsymbol{B_k}.\boldsymbol{u_{k-1}} + \boldsymbol{E_k}.\boldsymbol{q_{k-1}} \quad (2.216)$$

With here $\boldsymbol{A_k} \in \mathbb{R}^{5,5}$ the state space matrix, $\boldsymbol{B_k} \in \mathbb{R}^{5,1}$ the input state matrix and $\boldsymbol{E_k} \in \mathbb{R}^{5,3}$ the noise model matrix. $\boldsymbol{x_k}$ is the state vector at the current step time k, $\boldsymbol{u_{k-1}} = \delta_{k-1} \in \mathbb{R}$ is the input rudder angle of the ship and $\boldsymbol{q_{k-1}} \in \mathbb{R}^{3,1}$ the noise vector. The noise vector is here:

$$\boldsymbol{q_{k-1}} = \begin{pmatrix} \delta\psi_{w_{k-1}} \\ \delta r_{k-1} \\ \delta b_{k-1} \end{pmatrix} \quad (2.217)$$

This time, the state-space function is linear; hence the use of the KF. The previously introduced matrices are the following ones based on the presented equations, where K and T are the coefficients of the

Nomoto's transfer function of the ship:

$$\boldsymbol{A_k} = \begin{pmatrix} 1 & \delta t_{k-1} & 0 & 0 & 0 \\ -\delta t_{k-1}.\omega_w^2 & 1 - 2.\xi_w.\omega_w.\delta t_{k-1} & 0 & 0 & 0 \\ 0 & 0 & 1 & \delta t_{k-1} & 0 \\ 0 & 0 & 0 & 1 - \frac{\delta t_{k-1}}{T} & \delta t_{k-1} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.218}$$

$$\boldsymbol{B_k} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \delta t_{k-1}.\frac{K}{T} \\ 0 \end{pmatrix} \tag{2.219}$$

$$\boldsymbol{E_k} = \begin{pmatrix} 0 & 0 & 0 \\ \delta t_{k-1}.\omega_w^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \delta t_{k-1} & 0 \\ 0 & 0 & \delta t_{k-1} \end{pmatrix} \tag{2.220}$$

Again, by definition of the covariance matrix, it is possible to write that:

$$\boldsymbol{Q_k} = cov(\boldsymbol{q_k}) = \begin{pmatrix} \sigma^2_{\delta\psi_{w_{k-1}}} & 0 & 0 \\ 0 & \sigma^2_{\delta r_{k-1}} & 0 \\ 0 & 0 & \sigma^2_{\delta b_{k-1}} \end{pmatrix} \tag{2.221}$$

Note that the tuning of these diagonal terms is as hard to do as the previous EKF. The noise is also considered constant, hence here in the equation 2.221, the parameters inside the matrix are considered to be constant along the simulation. A dedicated document will be produced in the future and will be a performance note in which the description of the tunings and their influence will be discussed.

For the **measurement update step** of the KF observer, the model is much more simpler that in the previous EKF section. We will suppose here that the only total measurement available is the heading provided by the magnetic compass such that based on the general measurement model given by the equation 2.121, we have:

$$\boldsymbol{y_k} = \psi_{t_k} = \psi_{s_k} + \psi_{w_k} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \end{pmatrix} . \begin{pmatrix} \xi_k \\ \psi_{w_k} \\ \psi_{s_k} \\ r_k \\ b_k \end{pmatrix} + \delta\psi_{t_k} = \boldsymbol{G_k}.\boldsymbol{x_k} + \boldsymbol{r_k} \tag{2.222}$$

Where here, the measurement matrix $\boldsymbol{G_k} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \end{pmatrix} \in \mathbb{R}^{1,5}$ and the measurement sensor noise vector of the magnetometer is $\boldsymbol{x_k} = \delta\psi_{t_k} \in \mathbb{R}$. Therefore, in this case, the measurement covariance matrix is in fact a scalar $\boldsymbol{R_k} = cov(\boldsymbol{r_k}) = \sigma^2_{\delta\psi_{k-1_{MAG}}}$.

2.5.1.2 Implementation in the simulator and analysis

The observer with the classical Euler first order discretization is available through this link in the repository: observer using classical discretization. The one using the matrices exponential exact discretization is accessible via: observer using exact exponential discretization. After implementation and simulations, one gets the following result using the Simulator Case 9 and a specific mission:

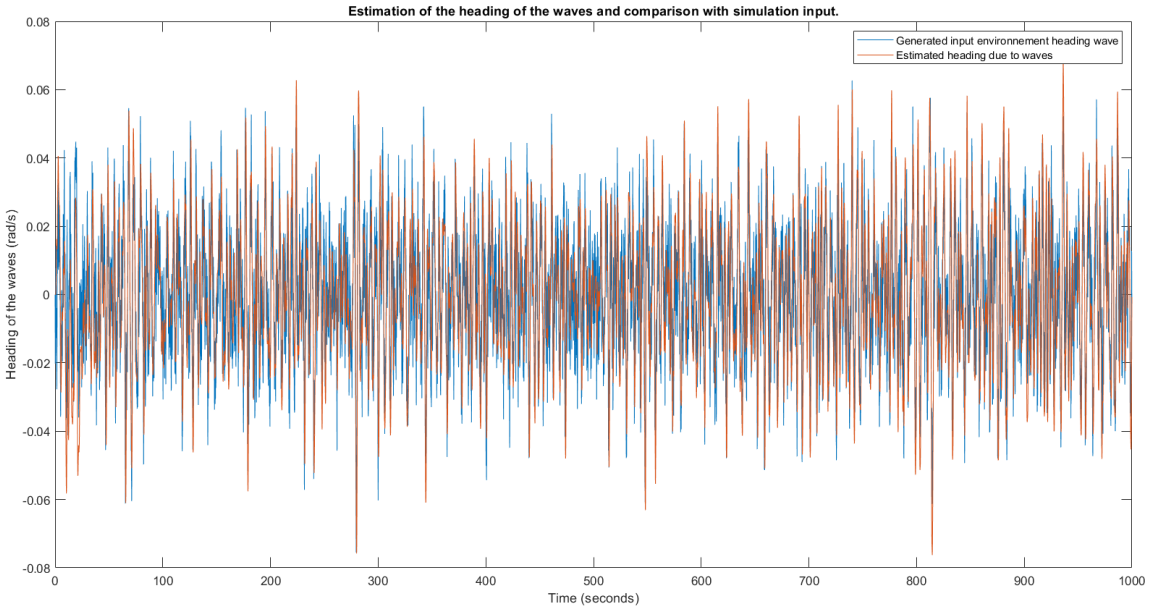

Figure 2.22: Estimation of the heading of the wave during a simulation (wave pulsation input of 1.2 rad/s).

The previous Figure is a general view of the output of the simulation. The output is the estimated heading due to waves ψ_w plotted (in orange) in comparison with the one specified from the environment block in input (in blue). Note that inside the wave observator algorithm, before 500 seconds, predefined values of pulsation and damping of the waves are used. After this time, the use of the estimated parameters is done.

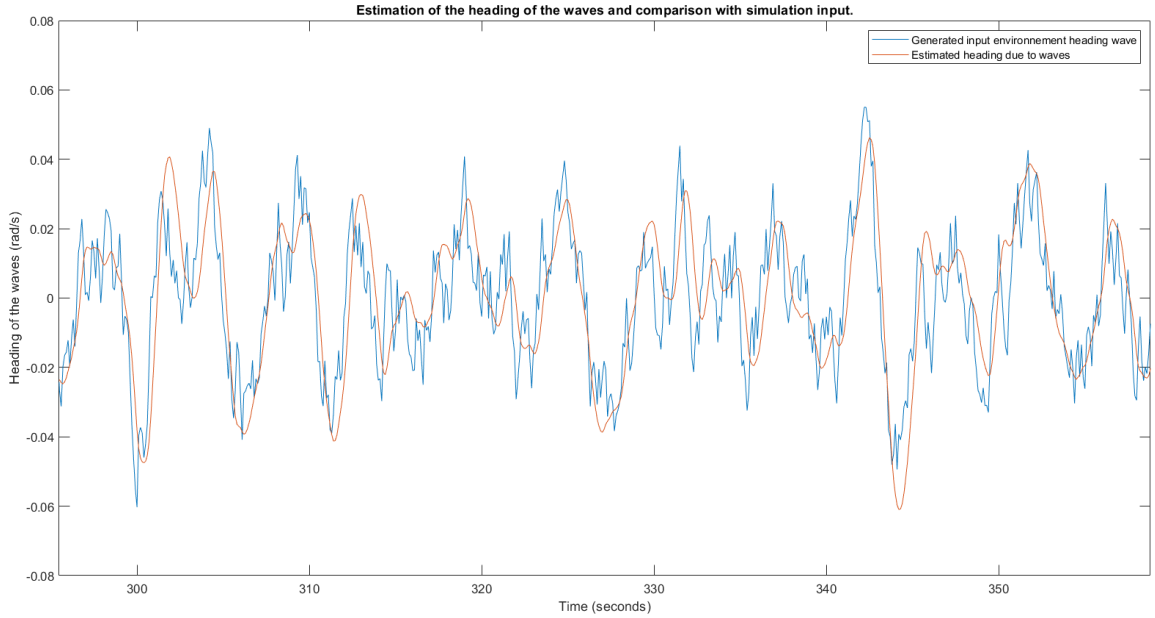

Figure 2.23: Estimation of the heading of the wave during a simulation before 500 seconds.

One sees that the estimate is quite good, but it is of this good quality because the parameters are known in advance. After 500 seconds however, one can see that the estimator is also quite good too:

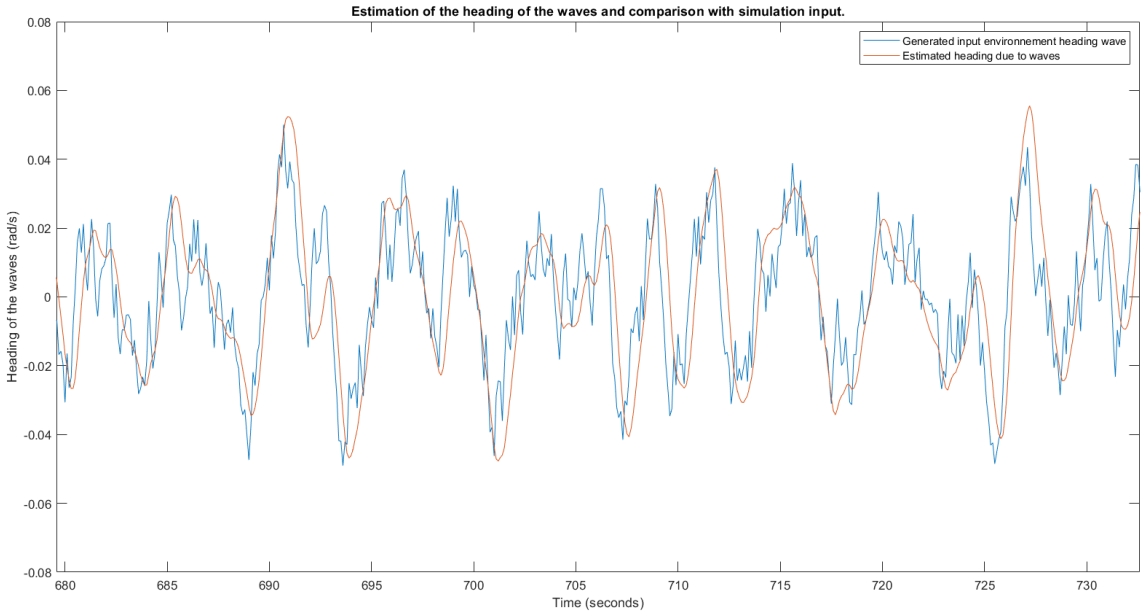

Figure 2.24: Estimation of the heading of the wave during a simulation after 500 seconds.

By taking a look at the various available headings simulated through this simulation, one has:

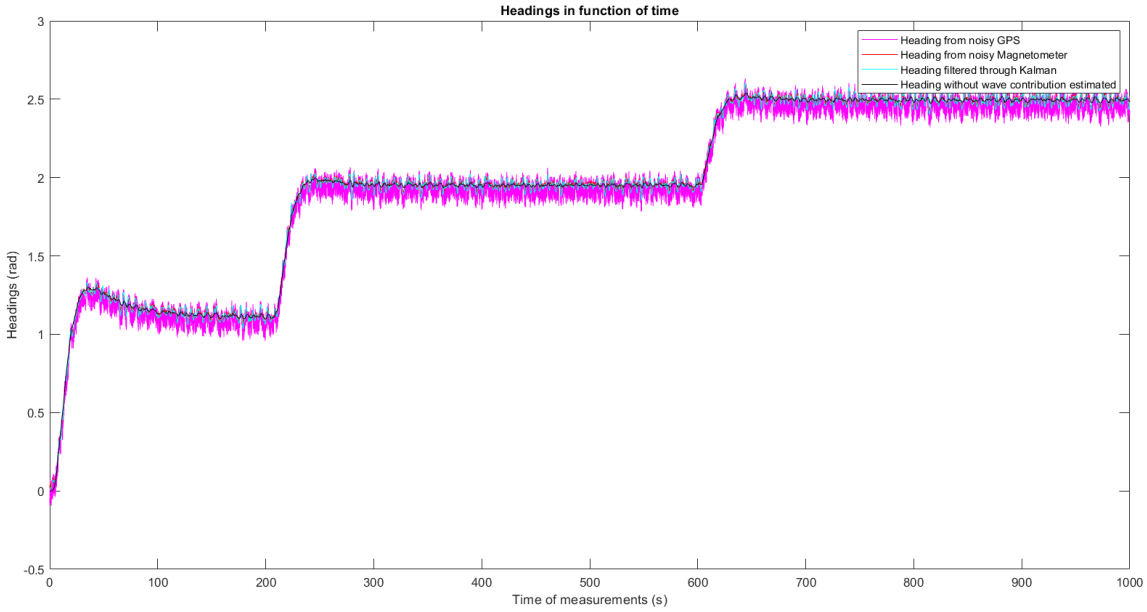

Figure 2.25: Estimation of the headings during the entire simulation.

The black plot represents the heading that the ship is supposed to have without the wave contribution, that is, ψ_s. Obviously, due to the effective steps of filtering and fusing, the black plot is of better shape than the perturbed one due to GPS and magnetometer noise perturbations. However, the tuning of the algorithm is not yet very optimal. Indeed, for some portions of the simulation, one can see the following:

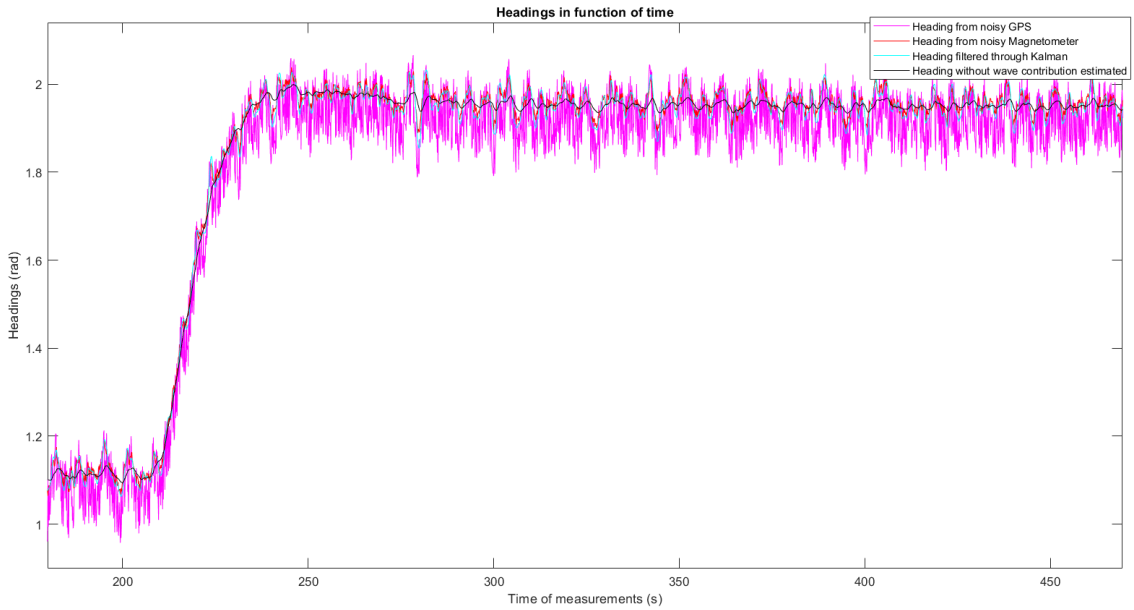

Figure 2.26: Estimation of the headings during a certain portion of the simulation.

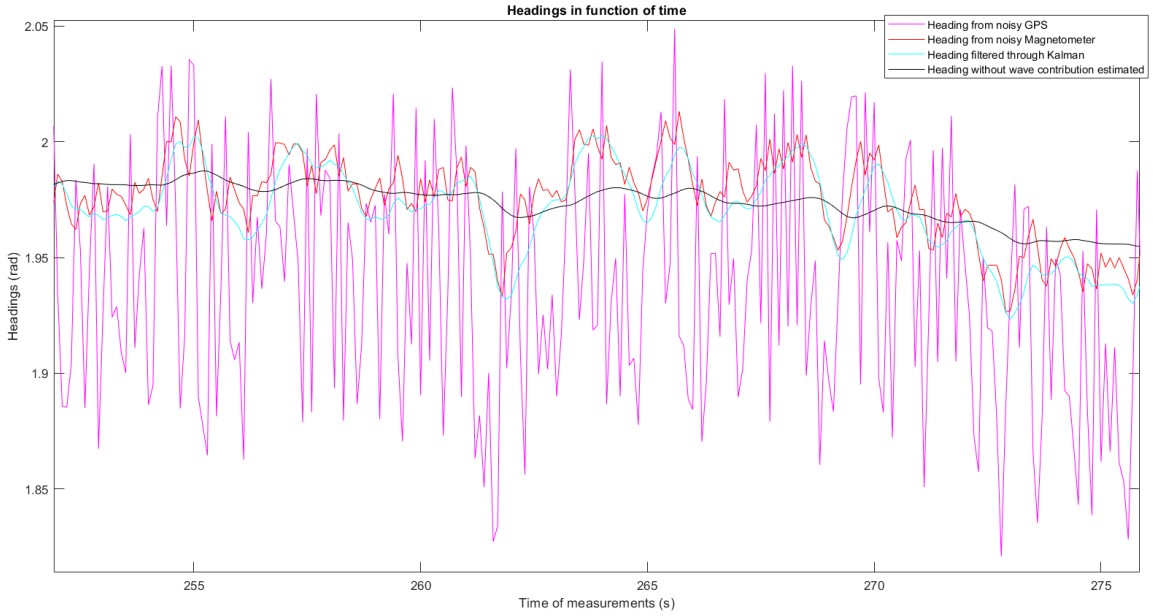

Figure 2.27: Estimation of the headings during a small portion of the simulation.

One sees on the previous picture that it seems that the waves oscillatory motion has been suppressed, in comparison with the cyan plot which consists on the headings measurements filtered and fused by the EKF but where the wave induced heading was not deleted. The red plot from the magnetometer also behaves with a strong oscillatory motion. So this portion seems to have been properly filtered in terms of wave motion.

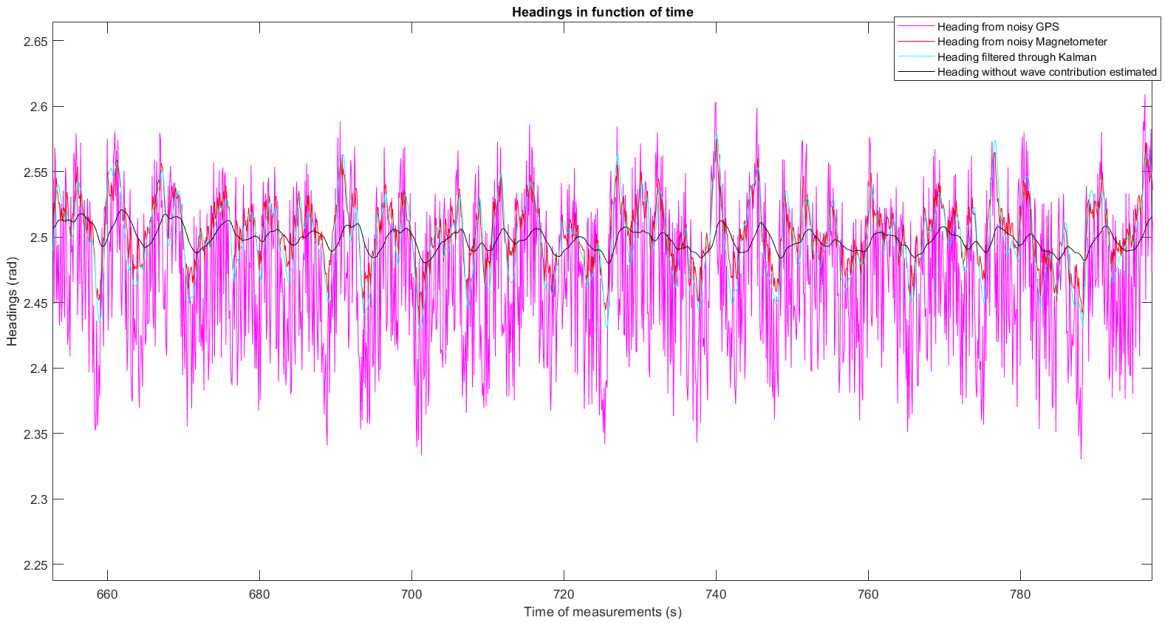

Figure 2.28: Estimation of the headings during a small portion of the simulation after 500 seconds.

One sees that on this portion, the black plot presents too an important oscillatory motion. Indeed, less important in terms of amplitude that the measurements, but yet, with some oscillating features. The tuning will be extensively analyzed in the future and a special document will be produced to estimate the performance quality of the algorithms and their tunings, as well as their qualified domain.

Bibliography

[1] B. Hérissé, "Contrôle optimal et planification de trajectoire pour le guidage des systèmes aérospatiaux," Université Paris Saclay, Thesis, 2023. [Online]. Available: https://theses.hal.science/tel-04095485/.

[2] S. P. Drake, "Converting gps coordinates to navigation coordinates," DSTO Electronics and Surveillance Research Laboratory, Technical Note, 2002. [Online]. Available: https://www.researchgate.net/profile/Samuel-Drake.

[3] A. Noureldin, "Fundamentals of inertial navigation, satellite-based positioning and their integration," in Springer, 2013, ch. 2.

[4] J. R. Clynch, "Radius of the earth - radii used in geodesy," Naval Postgraduate School, Tech. Rep., 2002.

[5] D. Magnetomter, "Compass heading using magnetometers," Honeywell, Tech. Rep.

[6] https://en.wikipedia.org/wiki/Tramontane.

[7] R. Hostettler and S. Sarkka, "Basics of sensor fusion," Aalto University, Finland, Tech. Rep., 2020.

[8] M. Weber and R. Zhang, "Formulaire de dérivation matricielle," ENS Ulm, Tech. Rep., 2009.

[9] I. Reid, "Lecture notes on estimation: Estimation 2.," Oxford University, Lecture, 2001. [Online]. Available: https://www.robots.ox.ac.uk/~ian/Teaching/Estimation/LectureNotes2.pdf.

[10] G. G. Bousquet, "Simulator justification file gnc v0.0.1," Casteldos GNC, France, Technical report 1, May 2025, Available on the referenced GitHub repository.

[11] T. I. Fossen, *Guidance Control of Ocean Vehicles*. John Wiley Sons, 1994.

[12] T. I. Fossen and T. Perez, "Kalman filtering for positioning and heading control of ships and offshore rigs," IEEE, Article, 2009. [Online]. Available: https://www.fossen.biz/publications/2009%20Fossen%20and%20Perez%20IEEE%20CST.pdf.

[13] J. YUAN and S. AN, "A wave peak frequency tracking method based on two-stage recursive extended least squares identification algorithm," IEEE, Article, 2021. [Online]. Available: https://www.researchgate.net/publication/349134766_A_Wave_Peak_Frequency_Tracking_Method_Based_on_Two-Stage_Recursive_Extended_Least_Squares_Identification_Algorithm.