



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Design and Implementation of a Mars Mission Analysis Software

Document

Report and User Manual

Author

Horia Ghionoiu Martínez

Supervisor

Dr. Miquel Sureda

Degree

MSc Aeronautical Engineering

Semester:

2020-2021 Spring

MASTER FINAL THESIS



Aknowledgements

To my mum, and my big brother, two of my most appreciated pillars.



Abstract

This project covers the entire design, construction, and release of a software artifact written in Python, which features a graphical user interface, and a MATLAB astrodynamics core. The software is to be used on mission design tasks, mainly focused on sustained human interplanetary mission design.



Contents

ACKNOWLEDGEMENTS	I
ABSTRACT	II
CONTENTS	III
FIGURES	V
TABLES	VI
ABBREVIATIONS	VII
INTRODUCTION	1
AIM.....	1
SCOPE.....	1
JUSTIFICATION	1
CONTEXT	2
STATE OF THE ART	3
PROJECT MANAGEMENT.....	7
<i>Task Analysis</i>	7
<i>WBS Diagram</i>	7
<i>WBS Dictionary</i>	7
<i>Gantt Diagram</i>	8
SOFTWARE ENGINEERING	10
ANALYSIS.....	10
<i>Requirements</i>	10
<i>Use Cases</i>	12
DESIGN	18
<i>Class Diagram</i>	19
<i>Graphical User Interface</i>	20
IMPLEMENTATION	25
<i>Architecture</i>	25
<i>Code</i>	26
<i>Used Technologies</i>	26
<i>Hardware and Software Resources</i>	26
TESTING	26
BUDGET	29
CONCLUSIONS.....	30
REFERENCES.....	31
USER GUIDE	32
INTRODUCTION.....	32
TARGET AUDIENCE.....	32
SOFTWARE DESCRIPTION	32
INSTALLING THE SOFTWARE.....	32
<i>Getting The Code</i>	33
<i>Configuring the IDE (PyCharm)</i>	36
<i>Other Configurations</i>	39
<i>Test The Installation</i>	40
USING THE SOFTWARE.....	42
<i>Graphical User Interface</i>	42
Main Window	42
PCP Table	43
S/C List	44
Buttons	44



Mission Tab	44
S/C Tab	45
Actions Tab	46
PCP Manager Tab	47
S/C Info	48
Status bar	48
Keyboard Shortcuts.....	48
Canvas Window	49
S/C Info	50
Trajectories Filter	51
Active Trips.....	51
<i>Edit filters Window</i>	51
Select S/C and Trip	54
Filter by Dates	54
Filter by Time of Flight	56
Filter by Energy	56
Auto Trajectory Selection	57
Active Filters View	57
Keyboard Shortcuts	58
<i>PCP Manager Window</i>	58
Generate PCP Tab	59
Convert PCP Tab.....	60
Working PCP Tab.....	61
<i>View PCP Window</i>	61
<i>Generate PCP Window</i>	62
TUTORIALS.....	62
ANNEXES	63
DOWNLOAD CODE FROM GITHUB (ZIP FILE).....	63

Figures

FIGURE 1. LEFT, INPUT WINDOW OF PCP_PLANET2PLANET.M. RIGHT, INTERACTIVE PORKCHOP PLOT GENERATED BY ASTROLIB'S PCP_VIEWER.M.	3
FIGURE 2. WBS DIAGRAM.....	7
FIGURE 3. GANTT DIAGRAM.....	9
FIGURE 4. USE CASE DIAGRAM - APP.	12
FIGURE 5. USE CASE DIAGRAM – START APP.....	13
FIGURE 6. USE CASE DIAGRAM – MANAGE WORKING PCP.....	14
FIGURE 7. USE CASE DIAGRAM – MODIFY S/C.....	16
FIGURE 8. USE CASE DIAGRAM – INSPECT S/C.....	17
FIGURE 9. CLASS DIAGRAM – APP.....	19
FIGURE 10. UI PROTOTYPE - MAIN WINDOW.....	21
FIGURE 11. UI PROTOTYPE - PCP MANAGER WINDOW.....	22
FIGURE 12. UI PROTOTYPE - PCP FILTER WINDOW.....	23
FIGURE 13. UI PROTOTYPE - S/C VIEWER (CANVAS) WINDOW.....	24
FIGURE 14. GET THE CODE FROM VERSION CONTROL (GITHUB).....	34
FIGURE 15. REPOSITORY URL CONFIGURATION.....	34
FIGURE 16. CODE HAS BEEN CLONED INTO YOUR COMPUTER, NOW YOU HAVE A LOCAL WORKCOPY.....	35
FIGURE 17. PYCHARM OVERVIEW WITH THE CODE DOWNLOADED.....	35
FIGURE 18. PYCHARM PYTHON INTERPRETER SETTINGS. LOCATED AT THE BOTTOM RIGHT AREA.....	36
FIGURE 19. ADDING A PYTHON 3.7 INTERPRETER, INTO A NEW VIRTUALENV.....	37
FIGURE 20. INSTALL NEW PYTHON PACKAGES, INTO THE CREATED VIRTUALENV.....	37
FIGURE 21. INSTALLING PYSIDE2 PACKAGE. DO THE SAME FOR PANDAS, AND SCIPY PACKAGES.....	38
FIGURE 22. INSTALL MATLABENGINEFORPYTHON PACKAGE. BE SURE THAT (YOUR VIRTUALENV PYTHON NAME) APPEARS AT THE LEFT SIDE OF EACH TERMINAL LINES, WHICH INDICATES THAT YOUR ARE OPERATING WITH OUR NEW INSTALLED VIRTUALENV. IF IT DOESN'T APPEARS, CLOSE AND OPEN THE TERMINAL AGAIN.....	38
FIGURE 23. MATLABENGINEFORPYTHON PACKAGE INSTALLED.....	39
FIGURE 24. CONFIGURING THE PATHS.....	39
FIGURE 25. RUNNING THE APP.....	40
FIGURE 26. APP WELCOME SPLASH IMAGE. THE MATLAB ENGINE TAKES A WHILE TO LOAD.....	41
FIGURE 27. SONET MARS MISSION PLANNER MAIN WINDOW.....	41
FIGURE 28. MAIN WINDOW GENERAL VIEW.....	42
FIGURE 29. USING THE PCP TABLE.....	43
FIGURE 30. PCP TABLE NAVIGATION, INSPECTING EARTH-MARS & MARS-EARTH TRANSITS FOR A ONE-WAY S/C. THE BOTTOM IMAGE IS THE FIRST ONE, BUT WITH A TOO RESTRICTIVE FILTER APPLIED, RESULTING IN ZERO AVAILABLE TRAJECTORIES.....	44
FIGURE 31. MAIN WINDOW MISSION TAB.....	45
FIGURE 32. MAIN WINDOW S/C TAB.....	46
FIGURE 33. MAIN WINDOW ACTIONS TAB.....	47
FIGURE 34. MAIN WINDOW PCP MANAGER TAB.....	48
FIGURE 35. CANVAS WINDOW GENERAL VIEW.....	50
FIGURE 36. EDIT FILTERS WINDOW OVERVIEW.....	53
FIGURE 37. EDIT FILTERS WINDOW, SELECT S/C AND TRIP COMBOS.....	54
FIGURE 38. EDIT FILTERS WINDOW, DATE - SIMPLEDATE FILTER.....	55
FIGURE 39. EDIT FILTERS WINDOW, DATE - COMPLEXDATE FILTER.....	55
FIGURE 40. EDIT FILTERS WINDOW, TIME OF FLIGHT FILTER.....	56
FIGURE 41. EDIT FILTERS WINDOW, ENERGY FILTER.....	57
FIGURE 42. EDIT FILTERS WINDOW, AUTO TRAJECTORY SELECTION FILTER.....	57
FIGURE 43. EDIT FILTERS WINDOW, FILTERS TABLE.....	58
FIGURE 44. PCP MANAGER WINDOW, OVERVIEW.....	59
FIGURE 45. PCP MANAGER WINDOW, GENERATE PCP TAB.....	60
FIGURE 46. PCP MANAGER WINDOW, CONVERT PCP TAB.....	61
FIGURE 47. VIEW PCP WINDOW.....	62
FIGURE 49. GET THE CODE DIRECTLY INTO A ZIP FILE, FROM GITHUB.....	63
FIGURE 50. CREATE A PYCHARM PROJECT FROM THE DOWNLOADED CODE.....	63



Tables

TABLE 1. WBS DICTIONARY.	7
TABLE 2. FUNCTIONAL REQUIREMENTS.	11
TABLE 3. NON-FUNCTIONAL REQUIREMENTS.....	11
TABLE 4. BUDGET.....	29
TABLE 5. MAIN WINDOW KEYBOARD SHORTCUTS.....	48
TABLE 6. EDIT FILTERS WINDOW KEYBOARD SHORTCUTS.	58



Abbreviations

API – Application Programming Interface

App – Application

DRA – Design Reference Architecture (NASA)

E-M – Earth-Mars

GMAT – General Mission Analysis Tool

GUI – Graphical User Interface

IDE – Integrated Development Environment

LOpp – Launch Opportunity

M-E – Mars-Earth

MANE – Mission Analysis Environment

N/A – Not Applicable

OOP – Object Oriented Programming

OS – Operative System

PCP – Pork Chop Plot (a plot with interplanetary trajectories)

PM – Project Management

S/C – Spacecraft

SONet – Sustainable Offworld Network

TOF – Time Of Flight

UI – User Interface

UML – Unified Modeling Language

WBS – Work Breakdown Structure

Widget – GUI element/object



Introduction

Aim

This project is aimed at design and implementation of a desktop cross-platform application, which at this point is called **SONetMarsAPP**. SONetMarsAPP helps trajectory designers and mission planners build an overall launch and flight sequence for human Mars reference missions.

Scope

The project will be done once the main work on **SONetMarsAPP** is finished. First of all, a series of meetings are held with the primary stakeholders (the project tutor), to agree the requirements, scope, and dates for the project.

Following, the **Project Scope Statement** is presented:

The main assumptions are:

- The project will be done by 0.5 software engineers. This means that I will work part time on the project.
- The project will be done by a novice software engineer.

Some constrains are identified:

- Part time work implies extended delivery times.
- Simultaneous full time work and studies during project development could lead to delays.

It remains out of scope:

- Provide an executable for the app.
- Draw the mission sequence.¹

The project scope, and related project management activities, are further described in the Project Management section, for example:

- Task Analysis, or activities identification.
- A graphical description of the deliverables ([WBS Diagram](#)).
- A detailed description of the deliverables (WBS Dictionary).
- Project timeline (Gantt Diagram).
- Cost estimation (found in Budget section).

Justification

This final master thesis project has the aim of helping in the SONet project², which consists of a set of academics and public/private entities dedicated to the development of human settlements on other worlds, especially the Moon and Mars. The SONet initiative is part of a global movement, whose objective is to provide the humanity with the capacity of exploring new worlds. Pursuing that objective would allow to develop a number of technologies which could improve the wellbeing on Earth.

¹ In the sense seen on NASA DRA 5.0 [6].

² <https://sonet-hub.com/>



Examples of this could be the development of new crop techniques in extreme conditions, recycling and reuse of resources, effects of microgravity on the human body and other health treatments, development in material science, among others.

My contribution to this effort has to do with the development of a desktop application which can be used in the preliminary design phases of a manned Mars mission towards a future human settlement offworld. Thus helping the designer or mission planner in the task of planning the sequence of launches, landings and transits between Earth and Mars.

The developed application could serve as a foundation stone for further, more-complex applications with the above-mentioned aim. It has the advantage of being programmed in a powerful OOP language as Python and furthermore being open-source licensed.

The intended user of the application has knowledge of astrodynamics and mission planning.

Context

The context, or starting point of this project is the idea to match or adapt an existing project to a new set of requirements. The existing project is a doctoral thesis [1], whose one of the deliverables is a MATLAB code, which performs astrodynamical computations. The new set of requirements respond to the interest of the SONet group, whose main stakeholder on my side was prof. Miquel Sureda, on developing a computer code, which helps in the preliminary mission planning and design, especially for human interplanetary missions to Mars.

The mentioned MATLAB code, here named AstroLib toolbox [1], is one of the main inputs to this project. Only a small part of AstroLib toolbox was ceded, which consist on two main functions, and 41 auxiliary functions, which are their dependencies. These two main functions are:

- **PCP_Planet2Planet.m**, which displays a window to choose the simulation parameters (Figure 1, left), solves the Lambert's problem, and finally generates the Porkchop data in MATLAB matrix files (.mat), for later plotting the Porkchop plots (PCP).
- **PCP_Viewer.m** which displays the Porkchop plot in an interactive window, read from the previously generated PCP data .mat files. Figure 1 (right) shows a general view of the PCP_Viewer window.

As commented, a relevant part of the initial efforts of this project, were to understand, extend, and modify the functionality offered by AstroLib's toolbox.

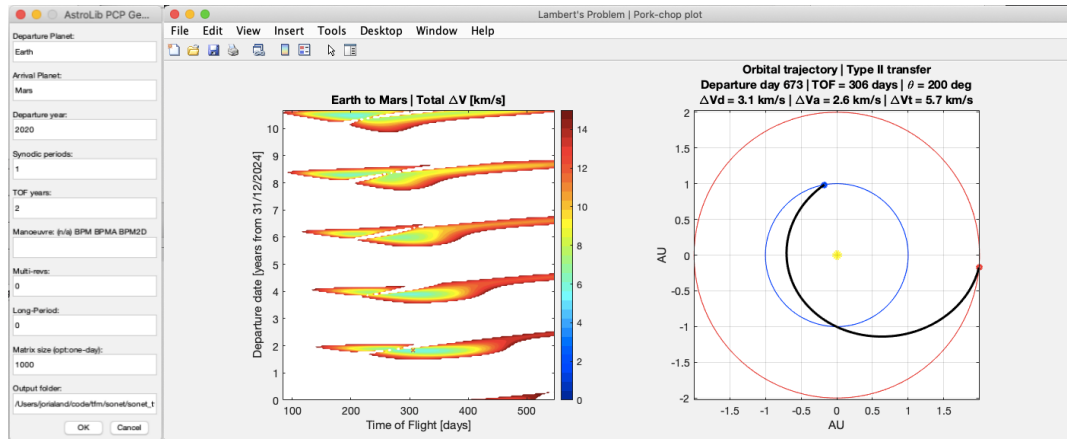


Figure 1. Left, input window of `PCP_Planet2Planet.m`. Right, interactive `Porkchop` plot generated by `AstroLib's PCP_Viewer.m`.

Also, there have been reviewed concepts of astrodynamics learnt during the previous career years, covering topics such as the two-body problem, orbital maneuvers, interplanetary trajectories, among others. The reference book for this review, has been [2].

Finally, the chosen programming language for developing the application is Python, as it is a highly popular choice for multi-purpose software, with lots of libraries available, and abundant literature where to find solutions to the problems arisen. This decision was one of the main challenges of the project, as involves starting from zero using a new language, with all that that entails.

State of the Art

Since ancient times, humans have been dreaming to explore other worlds [3], [4]. More recently in the 20th century, there have been different proposals to travel to Mars from the Earth.

Wernher von Braun was the first person to make a detailed technical study of a sustainable human Mars mission in the early fifties [5]. More recently, NASA developed the Mars Design Reference Mission series, the first one published in 1990, and the last one (5.0) in 2009 [6]. In these series, different mission architectures are investigated, on an early and conceptual manner, to draw the overall strategy of human sustained presence on Mars.



(Top) Mahoma riding its horse 'El-Borak', conducted by Gabriel archangel, went through the celestial sphere [3]. (Bottom) NASA's Artemis Program Moon surface mission recreation [7].

Traditionally, the major actors in the space exploration were the space agencies of the richest countries in the world, namely; NASA, ESA, or Roscosmos, although nowadays there are also commercial partners entering in the game, see SpaceX or BlueOrigins. All of them have plans to develop sustainable human presence in other worlds, although some of them with greater budget than others. It is clear that to conquer other worlds, one has first to develop the expertise, and mature the technologies in the near-Earth space. One of these initiatives is the NASA's Artemis Program [7]. In its phases I-II, with a requested budget of 27971.1 M\$, they plan to deliver a sustained human presence in the near-Moon region, the known as the Gateway, which will serve for the phases III and subsequent, and will allow later sustained human presence in the Moon. Once this technologies are proven, the next step would be to jump to the deep space, and plan a sustained human mission to. [7], [6].

In the early phases of space mission design, the designers have to perform a variety of tasks, examples are; identifying mission opportunities, defining performance requirements, determining launch and space vehicle sizes, support subsystem analysis, and perform a wide-ranging sensitivity analyses and trade off studies. It is from the results of such studies that candidate missions are selected for funding. [8]

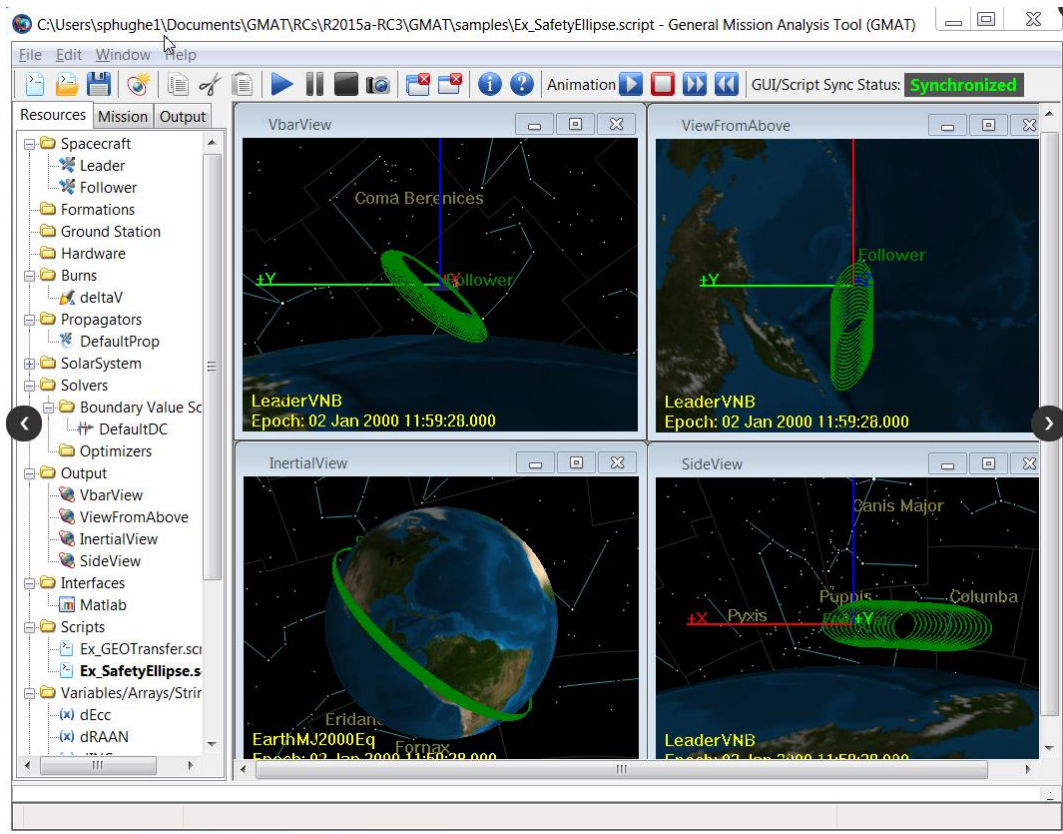


Figure 2. NASA's GMAT tool example mission and user interface. [9]

To conduct such studies, some software aided solutions are required. Examples of such softwares are NASA's General Mission Analysis Tool (GMAT) [9], NASA's Mission Analysis Low-Thrust Optimizer (MALTO) [10], or SpaceFlightSolution's Mission Analysis Environment (MANE). A general list of NASA's used software design tools can be found on [11]. Also, a list of NASA's open source projects can be found on [12].

GMAT software is an example of mission-approved, and space tested open-source tool for space mission design and navigation, it has been used by NASA to plan, design and operate space missions for decades, last 2020a version features interoperability with another mission design tools, like Copernicus [13] or JPL's MONTE [14].

MANE software is an example of mission-approved and space tested commercial tool for mission design [15]. It is considered among the most advanced commercial software products available for preliminary heliocentric mission analysis purposes. It can simulate and optimize missions with both low and high-thrust electric, chemical and nuclear propulsion models. It also interfaces with DE430 planetary ephemeris file and the comprehensive Dastcom5 ephemeris files, to provide a wide database of planets, comets and asteroids to which interact with. Also features an application that automates the appending of previously developed trajectory segments to form an extended mission. All these functionalities are integrated within a smooth and intuitive graphical user interface.

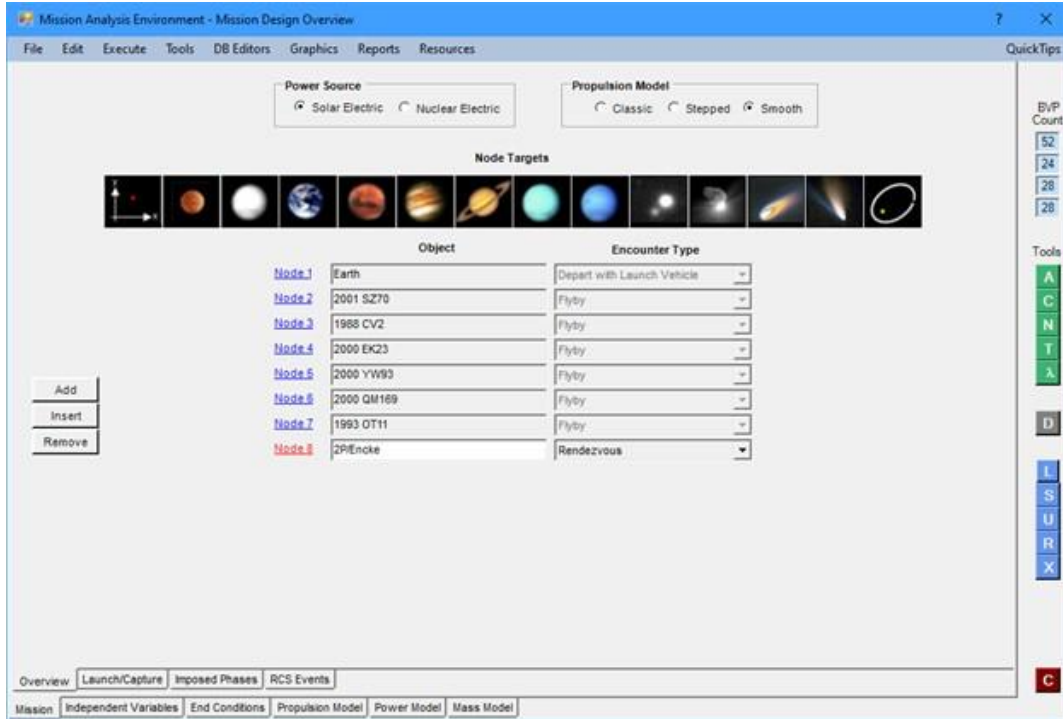


Figure 3. Spaceflightsolutions MANE software user interface example. [15]

As an example of mission design for sustained human presence in Mars, a series of NASA papers are published to help mission designers and planners to conduct their studies. MANE software has been used to conduct such studies in references [16] and [17], for example for drawing the Porkchop Plots and to perform sensibility studies of the relevant parameters under analysis.

Although the missions described within this project are based on the NASA's Design Reference Architecture [6] proposed Mars Surface Reference Missions, it is not known which software has been used to design them. In Figure 4, a sequence of Mars Long-Stay mission is plotted with an unknown software.

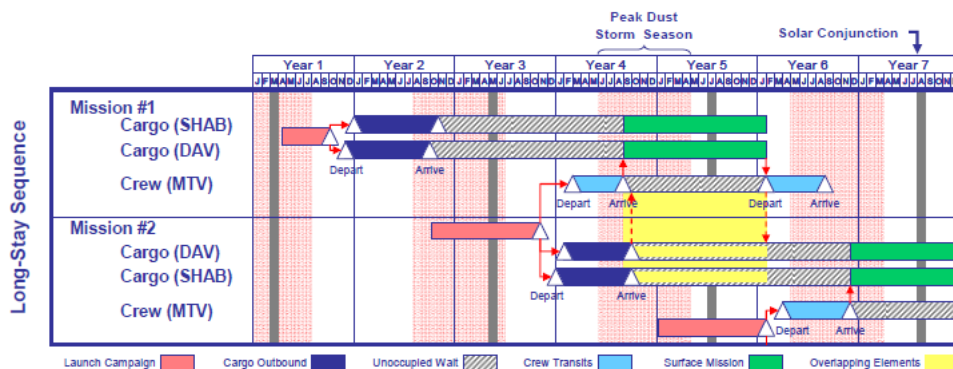


Figure 4. NASA's DRA 5.0 Long-Stay Mission Sequence example.

More closely related to this project development, the (here called) AstroLib package is an astrodynamics MATLAB code which can compute interplanetary trajectories and plot them (Figure 1). Part of AstroLib's code has been ceded by its author [1] to develop this project.

Project Management

Task Analysis

One of the first tasks regarding the project consists of identifying the activities that have to be performed to reach the marked goals, as well as the planification of how and when they have to be performed, in order to fulfill the planned calendar and to be within the assigned budget.

To do so, the project deliverables and work packages are hierarchically presented by using the WBS technique, through its graphical (WBS Diagram) and table representations (WBS Dictionary). Also, the project timeline is shown by assigning a time window to each activity present in the WBS (Gantt Diagram).

WBS Diagram

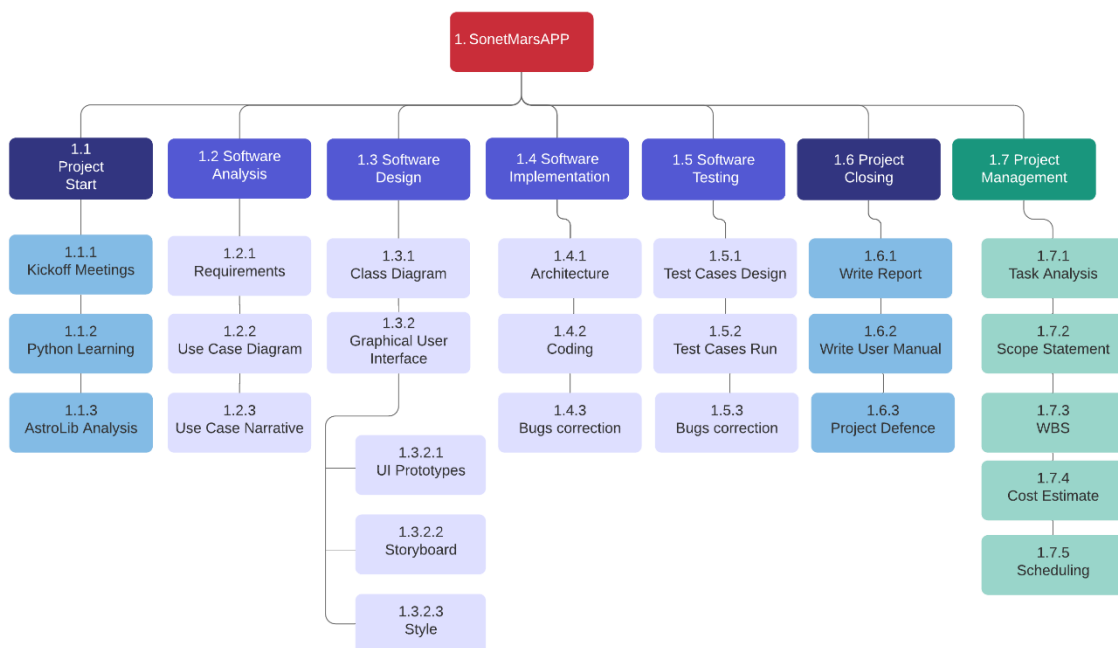


Figure 5. WBS Diagram.

WBS Dictionary

The WBS diagram above presented (Figure 5), is detailed in the WBS dictionary shown in Table 1.

Table 1. WBS Dictionary.

Level	WBS code	WBS name	WBS description
1	1,1	Project Start	Activities related to the start of the project.
2	1,1,1	Kickoff Meetings	Initial meetings with the stakeholders.
2	1,1,2	Python Learning	Transversal activity, held throughout the project.
2	1,1,3	AstroLib Analysis	Transversal activity, needed to understand and extend the initial functionality of the MATLAB package AstroLib.
1	1,2	Software Analysis	SW analysis phase.

2	1,2,1	Requirements	Functional and non-functional requirements gathering.
2	1,2,2	Use Case Diagram	Application functionality representation through UML Use Case diagrams.
2	1,2,3	Use Case Narrative	Use Case diagrams description through Use Case Narratives.
1	1,3	Software Design	SW design phase.
2	1,3,1	Class Diagram	Class structure design and class diagram generation.
2	1,3,2	Graphical User Interface	GUI design.
3	1,3,2,1	UI Prototypes	Prototyping the UI.
3	1,3,2,2	Storyboard	Prototyping the different UI screens and the user interaction.
3	1,3,2,3	Style	Defining the app's style.
1	1,4	Software Implementation	SW implementation phase.
2	1,4,1	Architecture	Define the app's architecture.
2	1,4,2	Coding	Codification.
2	1,4,3	Bugs correction	Bugs identification and correction.
1	1,5	Software Testing	SW validation phase.
2	1,5,1	Test Cases Design	Design the Test Cases.
2	1,5,2	Test Cases Run	Execute the Test Cases and gather the results.
2	1,5,3	Bugs correction	Bugs identification and correction.
1	1,6	Project Closing	Activities related to the closing of the project.
2	1,6,1	Write Report	Report confection.
2	1,6,2	Write User Manual	User Manual confection.
2	1,6,3	Project Defence	Presentation preparation and defence.
1	1,7	Project Management	PM related activities.
2	1,7,1	Task Analysis	Tasks identification.
2	1,7,2	Scope Statement	Project scope definition.
2	1,7,3	WBS	WBS diagram and dictionary generation.
2	1,7,4	Cost Estimate	Estimation of the costs.
2	1,7,5	Scheduling	Project Timeline, including Gantt diagram.

Gantt Diagram

Once the activities are established, we proceed to the temporal estimation of each one of them. The result of such planification is a Gantt Diagram. Each bar represents a planned activity, and its length, the duration, in days. See Figure 6.

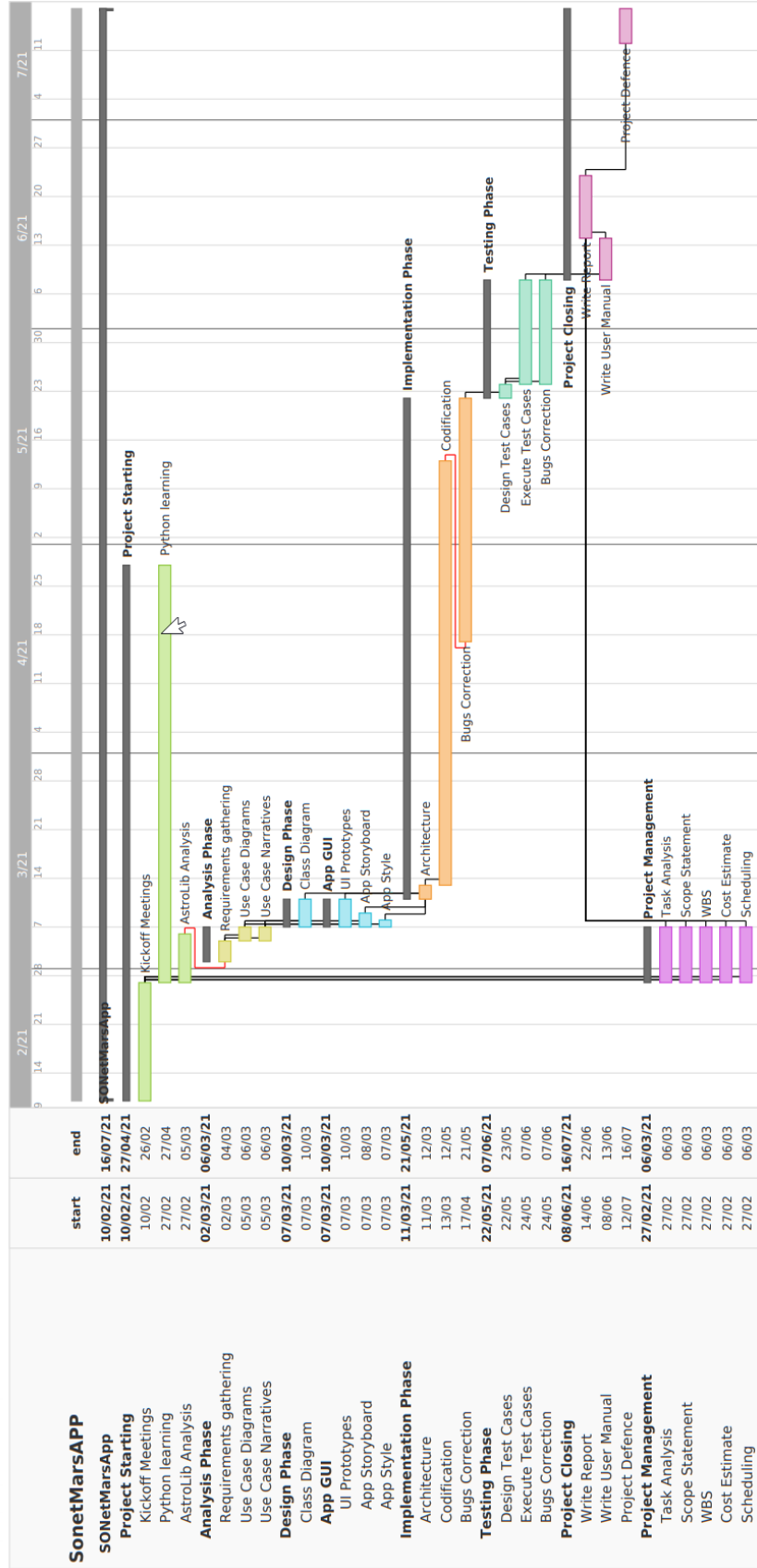


Figure 6. Gantt Diagram



Software Engineering

Once the aim and scope of this project are presented, we proceed to describe the process that has been followed in the design and implementation of the application, by using software engineering schemes.

Software engineering means writing quality software with the budget and time constraints. This is explained qualitatively in the so-called devil's square of Sneed. Designing software comprehends several stages:

Analysis

Consists on the realisation of an assessment on the specific tasks that the program has to perform as well as how they perform it or what response should they have for certain user inputs. This assessment is done by means of:

- Requirements specification.
- Use Cases.

Design

It consists of determining the structure of the building blocks which conform the application. This phase is important because it sets up the path which will be followed in the implementation, that is, writing of the code. It consists of:

- Class Diagram.
- UI Design.

Implementation

Entails the transformation of the structural design and qualitative steps into real machine-executable code that is going to turn into something functional. This process is described by means of:

- Architecture.
- Used Technologies.

Testing

During the implementation phase, and after it, several tests are performed with the aim of assuring that the application fulfills the requirements set at the beginning of the project. The main technique is the black box testing, which consists of testing whether the output under certain conditions is the desired. These tests are periodically run during the implementation and when one of them fails, a debugging phase has to be started.

Analysis

Requirements

In this phase the foundation stones of the program are defined. They have to be clear and concise. Not only that, it should be also specified, what the software must not do. There are several types of requirements, here we classify them into functional and non-functional ones.

User Roles

There are no special roles intended (admin user, normal user, etc.). The only role is a generic user.

Functional Requirements

Those regarding things that the user can see and interact with. They are shown in Table 2 and Table 3.

Table 2. Functional requirements.

Requirement ID	Description
REQ_FUN_1	The user shall be able to predefine a mission, and load it automatically when starting the app, without the need of starting from zero each time.
REQ_FUN_2	The user should be able to generate new PCP trajectories.
REQ_FUN_3	The user should be able to limit the maximum delta-v (total) for new generated PCP trajectories.
REQ_FUN_4	The user should be able to change the app's working PCP trajectories database.
REQ_FUN_5	The user should be able to change a s/c state by modifying the filter applied to the PCP trajectories (e.g. max delta-v, departure dates, among others).
REQ_FUN_6	The user should be able to insert a filter which automatically choses a transit , based on a rule, regarding the maximum or minimum value of a given parameter (e.g. max delta-v total).
REQ_FUN_7	The user should be able to change a s/c state by modifying the transit selected, either selecting a trajectory or unselecting it. (from the PCP table).
REQ_FUN_8	The user should be able to change a s/c state by selecting a transit directly through the PCP plot.
REQ_FUN_9	The user should be able to inspect the trajectories available for a s/c and transit by inspecting a table.
REQ_FUN_10	The user should be able to inspect the trajectories available for a s/c and transit by inspecting a PCP plot.
REQ_FUN_11	The user should be able to inspect the full state of a s/c into a separate window.
REQ_FUN_12	The user should be able to save the working session, to load it for later uses.
REQ_FUN_13	The user should be able add/remove s/c.
REQ_FUN_14	Start the app (with no predefined mission).

Non-Functional Requirements

Non-functional requirements describe the criteria that can be used to judge the operation of the software.

Table 3. Non-functional requirements.

Requirement ID	Description
REQ_NFUN_1	The device running the app should be a desktop computer.
REQ_NFUN_2	The device running the app should be running Window, MacOS, or Linux OS.
REQ_NFUN_3	The system running the app should have installed Python.
REQ_NFUN_4	The system running the app should have installed Matlab.
REQ_NFUN_5	The Python version running the app should be compatible with the Matlab version installed.
REQ_NFUN_6	The following Python packages should be installed in the Python instance: Pandas, Scipy, PySide2.

Use Cases

Following, the system's behaviour is described, from the user point of view. The used technique are the Use Case diagrams. Each Use Case diagram has an associated Narrative, which describes the context where it is developed.

The Use Case diagram for the entire system (the app) is presented (Figure 7), six main Use Cases are identified, which cover the Requirements defined above.

- Start App (Figure 8) [REQ_FUN_1, REQ_FUN_13, REQ_FUN_14]
- Manage Working PCP (Figure 9) [REQ_FUN_2, REQ_FUN_3, REQ_FUN_4]
- Add S/C [REQ_FUN_13]
- Remove S/C [REQ_FUN_13]
- Modify S/C (Figure 10) [REQ_FUN_5, REQ_FUN_6, REQ_FUN_7, REQ_FUN_8]
- Inspect S/C (Figure 11) [REQ_FUN_9, REQ_FUN_10, REQ_FUN_11]

Observing the application Use Case diagram shown in Figure 7, one can get the general idea of the its functionality. This is, the app starts, then the user can manage the PCP trajectories, create new ones, and set the one desired as the application trajectories database. Now, the user can proceed creating (or removing) s/c, modify them, and inspecting their current state.

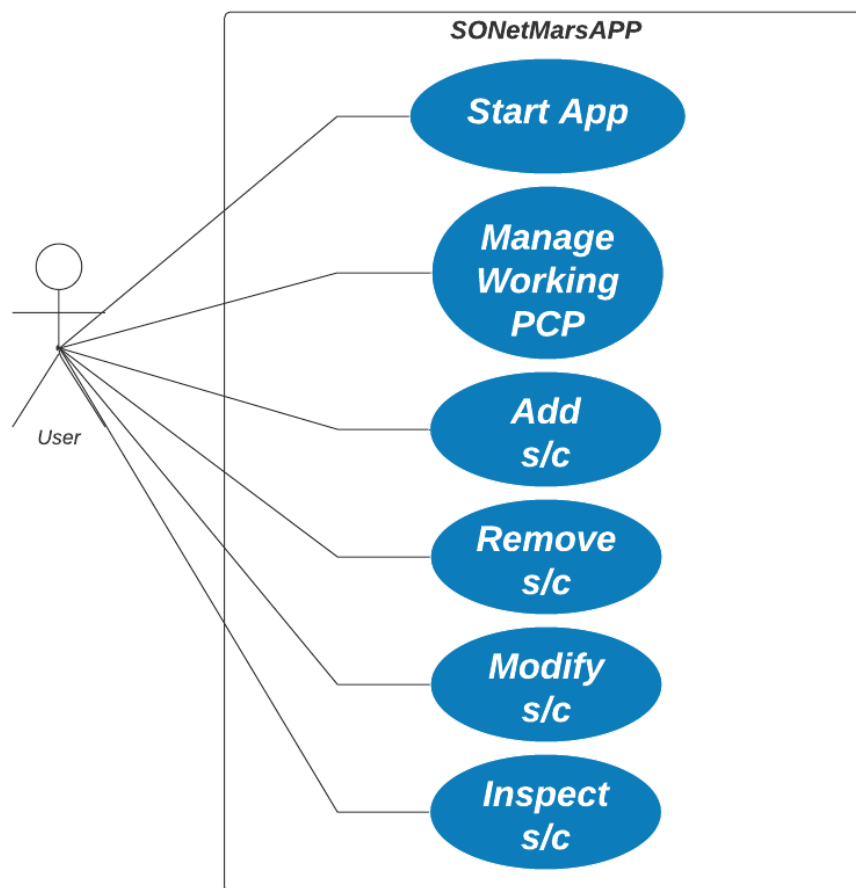


Figure 7. Use Case diagram - App.

Use Case - Start App

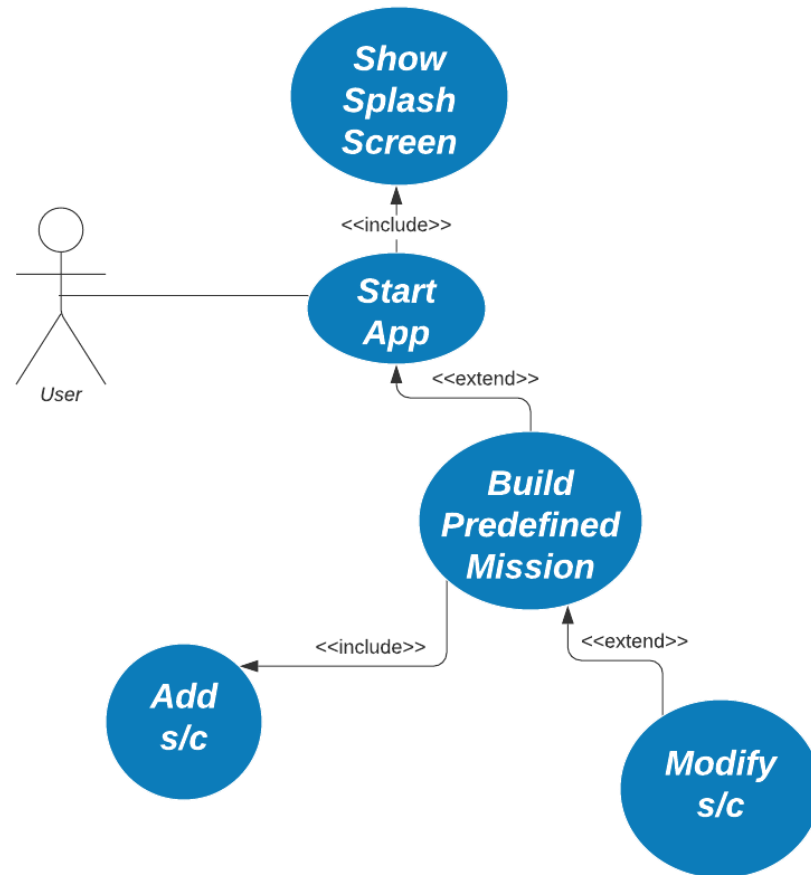


Figure 8. Use Case diagram – Start App.

Narrative – Start App

Actor: The user.

Preconditions: N/A.

Objectives: Start the app.

Main path:

- The app starts.
- The splash screen is displayed, while the Matlab core is loading.
- The app main window is shown.

Alternative path:

- If activated by the user, a predefined mission is built.
- S/C are automatically created.
- S/C are automatically modified.
- The app main window is shown, together with the Canvas window.
- The Canvas window shows the s/c state when user selects it.

Use Case - Manage Working PCP

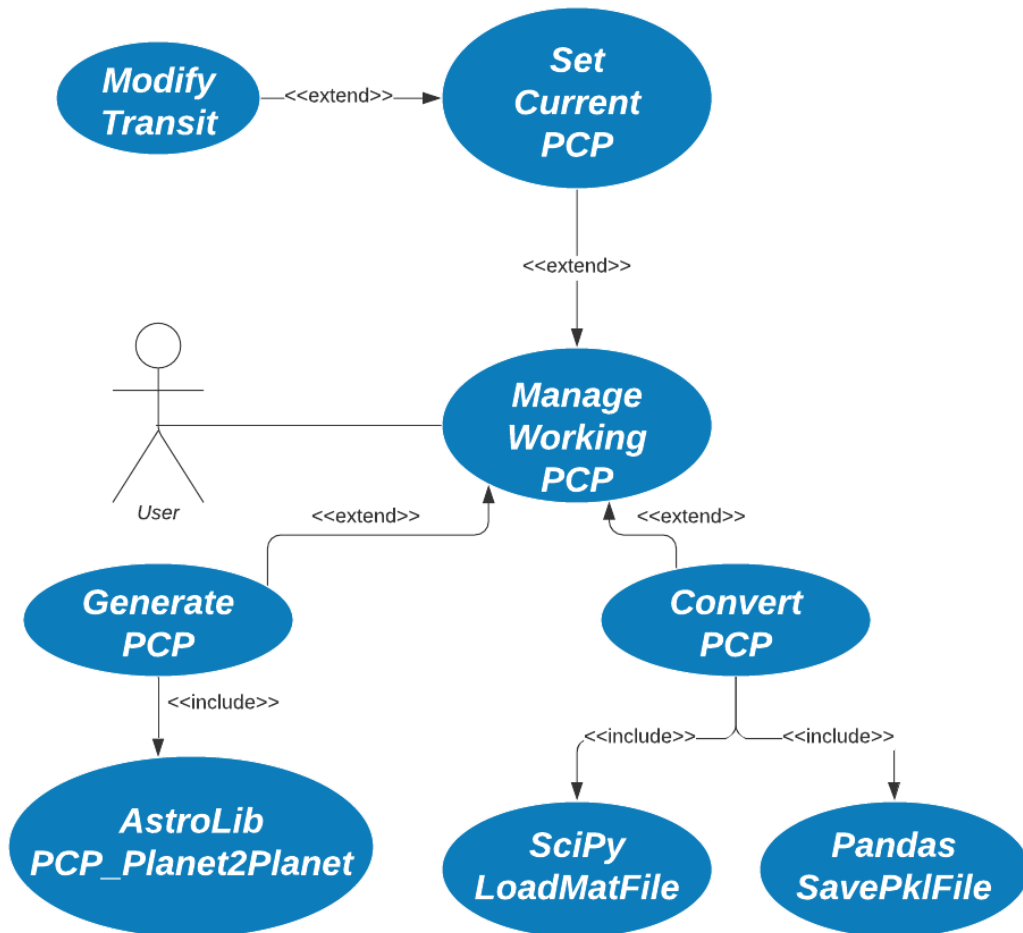


Figure 9. Use Case diagram – Manage Working PCP.

Narrative - Manage Working PCP

Actor: The user.

Preconditions: The app has started.

Objectives: Manage the PCP database, creating new PCPs and set the working one for the app.

Main path:

- The user access this function through a button which invokes a separate window.
- The predefined PCP will be shown.

Alternative path 1:

- The user wants to generate a new PCP matrix file.
- The user access this function through a button which invokes a separate window.
- The user configures the parameters through the window, and once she's done, click OK.
- The AstroLib Matlab application is called with the set parameters, and it calculates the PCP.
- The PCP matrix file (.mat) is output for later use.

Alternative path 2:

- The user wants to convert PCP matrix file to a PCP tabular file.
- The user access this function through a button which invokes a separate window.



- The user can optionally set a limit delta-v (total) through the window, and once she's done, click OK.
- The Python SciPy package is called to read the incoming mat file, the app process it, and then calls to the Python Pandas package to write it in tabular form.
- The PCP tabular file (.pkl) is output for later use.

Alternative path 3:

- The user wants to change the PCP working file for the app.
- The user access this function through a button which invokes a separate window.
- The user chooses the PCP tabular file to be used.
- The loaded file is analysed and set as working PCP file.

Use Case - Add S/C

There is no detailed Use Case diagram for the Add S/C Use Case.

Narrative - Add S/C

Actor: The user.

Preconditions: The app has started.

Objectives: Add a new s/c.

Main path:

- The user writes a name.
- The user selects the payload type.
- The user selects the type of transit (one-way or two-way).
- The user clicks a button and the s/c is added to the database.

Alternative path:

- If the s/c name is already on the database.
- A message is shown in the status bar and no action is performed.

Use Case - Remove S/C

There is no detailed Use Case diagram for the Remove S/C Use Case.

Narrative - Remove S/C

Actor: The user.

Preconditions: The app has started.

Objectives: Remove a s/c.

Main path:

- The user selects a s/c and clicks a button to remove it.

Alternative path:

- If the user doesn't select a s/c and just clicks the remove button.
- The last added s/c is removed.
- If no s/c left, a message is shown in the status bar, and no action is performed.

Use Case - Modify S/C

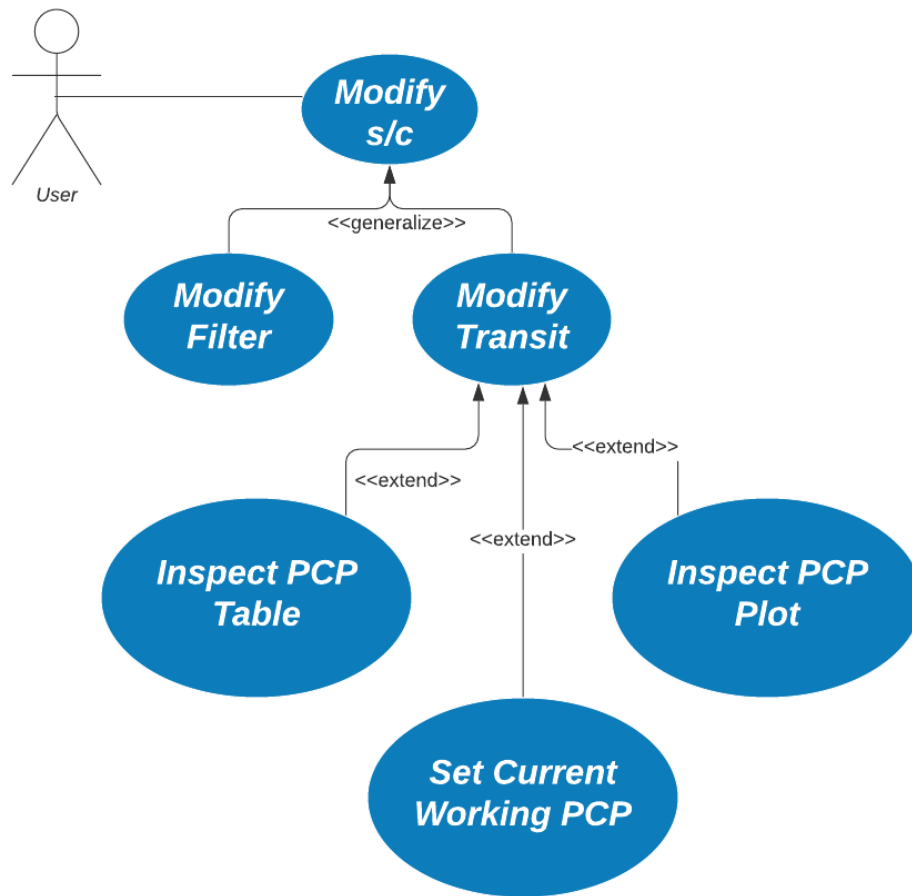


Figure 10. Use Case diagram – Modify S/C.

Narrative - Modify S/C

Actor: The user.

Preconditions: There is at least a s/c.

Objectives: Modify its state.

Main path 1 (Modify Filter):

- The user wants to edit the s/c filter.
- Selects a s/c and clicks a button to open a separate window, where to edit the filter.
- Accepts the window, which is closed.
- If the user had selected a s/c before the window has opened, this s/c is updated with the new filter and its new state is shown in the main window, or canvas window, if opened.
- If the user had chosen the option, a transit is automatically selected for the current s/c and the information is updated in the main window, or canvas window, if opened.

Alternative path 1:

- If the user cancels the window, it is closed and no action is performed.

Main path 2 (Inspect PCP Table):

- The user wants to modify a s/c transit.
- She selects a trajectory manually from the PCP table and clicks a button.

- The trajectory is added as chosen transit for the current selected s/c.

Alternative path 2:

- The user could also remove an already selected transit, by clicking another button.

Main path 3 (Inspect PCP Plot):

- The user wants to modify a s/c transit, by directly viewing the PCP plot and selecting a point from it.
- She selects a trajectory manually from the PCP plot and closes the window.
- The trajectory is added as chosen transit for the current selected s/c.

Alternative path 3:

- The user closes the PCP plot with no trajectory selected, no action would be performed.

Main path 4 (Set Current Working PCP):

- The user changes the working PCP.
- All the current selected transits should be reset (removed).
- All the s/c with transits selected, loses them.

Use Case - Inspect S/C

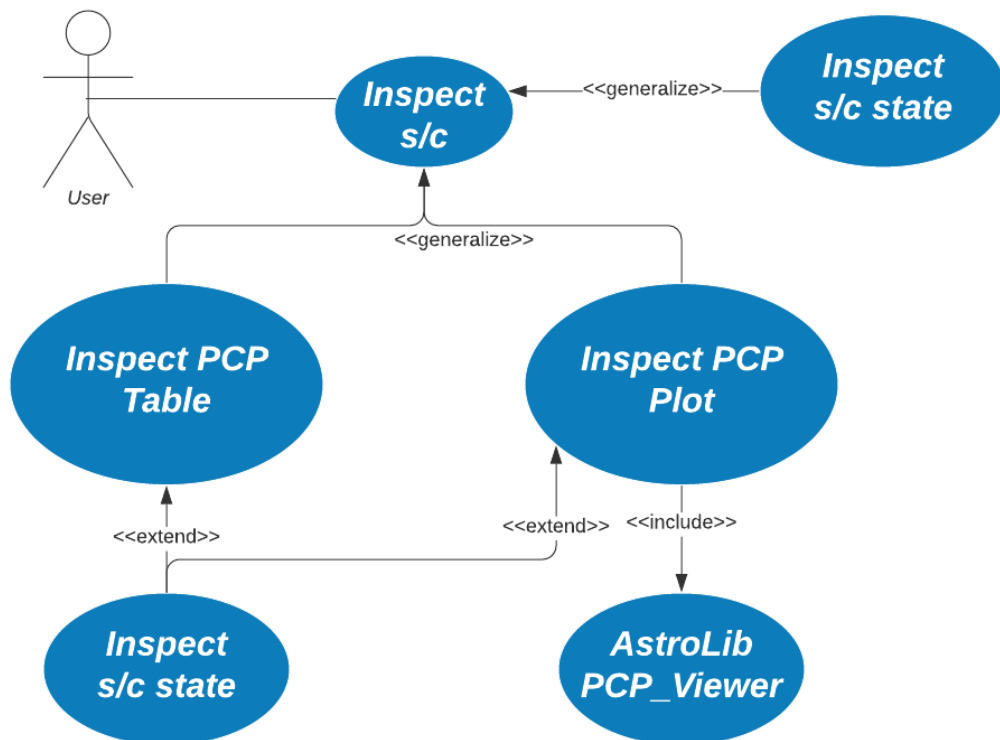


Figure 11. Use Case diagram – Inspect S/C.



Narrative - Inspect S/C

Actor: The user.

Preconditions: There is at least a s/c.

Objectives: Inspect its state.

Main path:

- The user can inspect the current state of a s/c by inspecting the PCP table in the main window.

Alternative path:

- The user can also inspect graphical the current available trajectories by inspecting the s/c PCP plot.
- Click the button which invokes the PCP plot.
- This, calls to AstroLib, which is in charge of plotting the PCP in a separate window.
- The user can close the window when done.

Alternative path:

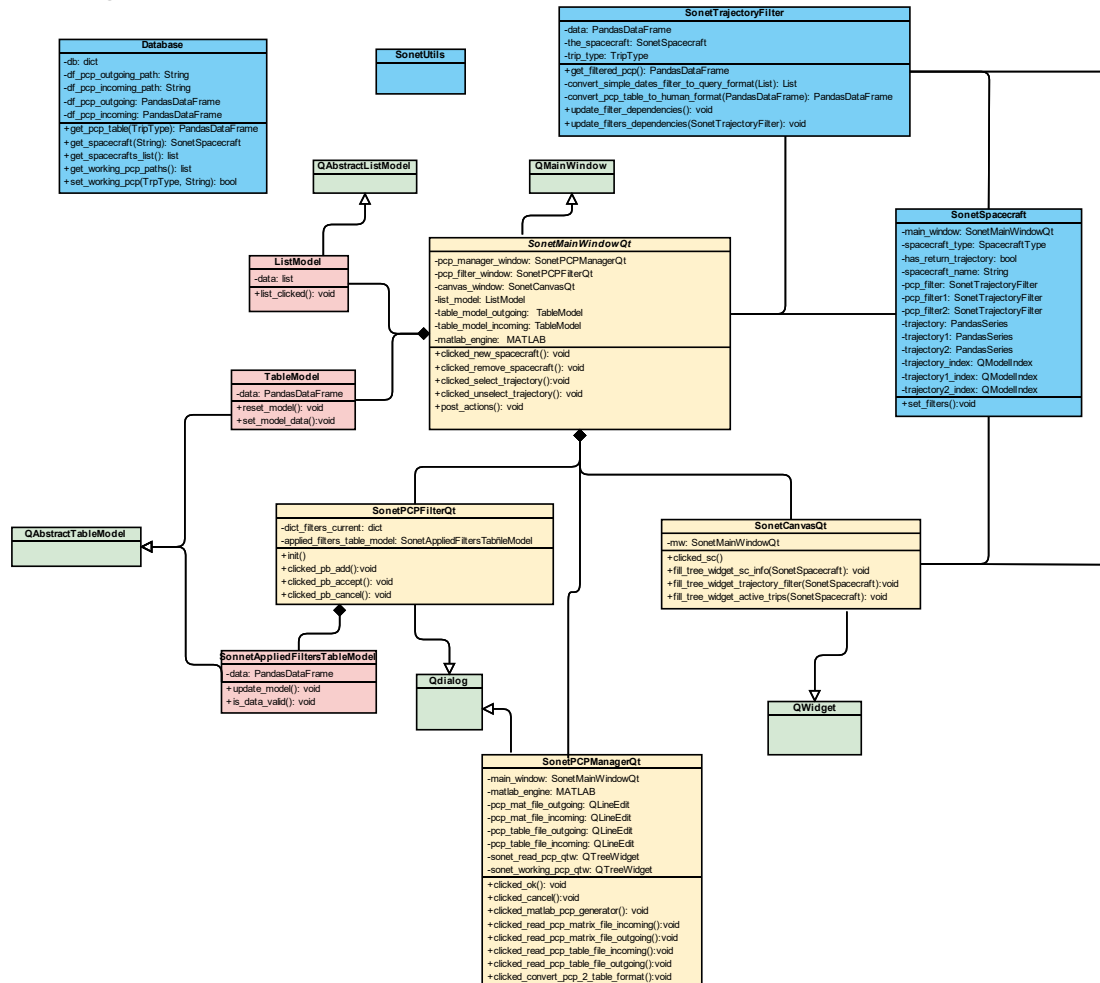
- The user can see a full s/c state resume into a separate window, the canvas window.
- Click a button to open the canvas window.
- When opened, if the user selects a s/c in the main window, the canvas window shows its current state, which is the s/c info, its filter, and its current selected transits.

Design

Once the app's requirements, and also its behavior have been defined, we can proceed to the design phase. The goal within this section is to create the basic system structure, which will be implemented in the Implementation/Coding phase. The main tasks in this section are two:

- Class Diagram design: a UML diagram where all the classes to be implemented are represented. Some level of detail is added by showing also the relations between them, and their most relevant attributes and methods.
- Graphical User Interface design: covering the design of the graphical elements which the user will interact with, including screens, style, among othes.

Class Diagram



Visual Paradigm Online Free Edition

Figure 12. Class diagram – App.

Following, the classes are described. The classes which represent a window are the following:

- **SonetMainWindowQt**: the main window, from where the rest of windows are invoked. It is responsible of the main app's logic, including starting the app, creating/removing s/c, and calling and managing the other windows.
- **SonetPCPManagerQt**: window in charge of managing the different .mat, .pkl trajectories files, their generation, conversion, and setting of the current working ones.
- **SonetPCPFilterQt**: window in charge of managing the filter object of each s/c and transit.
- **SonetCanvasQt**: window in charge of displaying full s/c state.

Furthermore, there are two additional blocks, which are coming from the MATLAB code from AstroLib's package [1]. These will also generate two additional windows. One of this windows is the **PCP generation window**, called by SonetPCPManagerQt, which generates the matrix .mat files. The second one is the **PCP Viewer window**, called from the main window, which is in charge of plotting the PCP trajectories, and enables the user to select one from the plot.

Also, there are two classes, which are created to implement the app business logic, these are:

- **SonetSpacecraft**: the central object used in the app. It's a simplified model of a spacecraft (s/c), and contains one or two transits (E-M or E-M+M-E³), and one or two filters.
- **SonetTrajectoryFilter**: The filters, which live within the SonetSpacecraft objects. It contains a set of conditions to limit the trajectories candidates for a given s/c.

Last, there are three classes, which represent two tables and a list, used in SonetMainWindowQt and SonetPCPFilterQt windows. These classes are derived from Qt's QAbstractTableModel and QAbstractListModel classes, respectively.

- **TableModel**: used in the main window to store the trajectories in table format.
- **ListModel**: used in the main window to show the list of s/c.
- **SonetAppliedFiltersTableModel**: used in SonetPCPFilterQt to show the filter in table format.

The used libraries are:

- Qt for Python (PySide2), for the GUI.
- Pandas, for tabular data handling.
- SciPy, for mat files handling in Python.
- Matlab Engine for Python, for calling Matlab functions from Python.

Graphical User Interface

UI Prototypes

The UI prototypes are design phase artifacts, which are going to be used to implement the user interface. A UI prototype is a draft which shows which type of graphical elements (widgets) the user is going to interact with on the different app's screens, along with their organization and location within the UI.

The UI prototypes have been designed according to the needs stated in the Analysis section. Furthermore, in the Class Diagram section, it has been stated that there will be four different Python classes, which will implement four separate windows or GUIs. One window will be the main one, and the other will raise from this mother window. Following are each window's prototypes.

UI Prototype – Main Window

The main window is the app's welcome window and the main working area. It consists of:

- Two table widgets, where the PCP is shown in table format. Both tables are stacked in a unique widget, and the user will change between them by clicking on the corresponding tab. Each table represents the trajectory possibilities for an Earth-Mars transit, and vice versa.
- A list widget, where the current existing s/c are listed. The user clicks on the desired s/c, and the rest of widgets, from this and other windows, update showing information related to it.
- A buttons area, where all the buttons and associated widgets are placed. The widgets of this area will be grouped according to their function.
- A status area, where some relevant information for the current selected s/c is to be displayed.
- A status bar area, where eventual messages appear, to inform the user when needed.

The UI prototype for the main window is shown in Figure 13.

³ See abbreviations section.

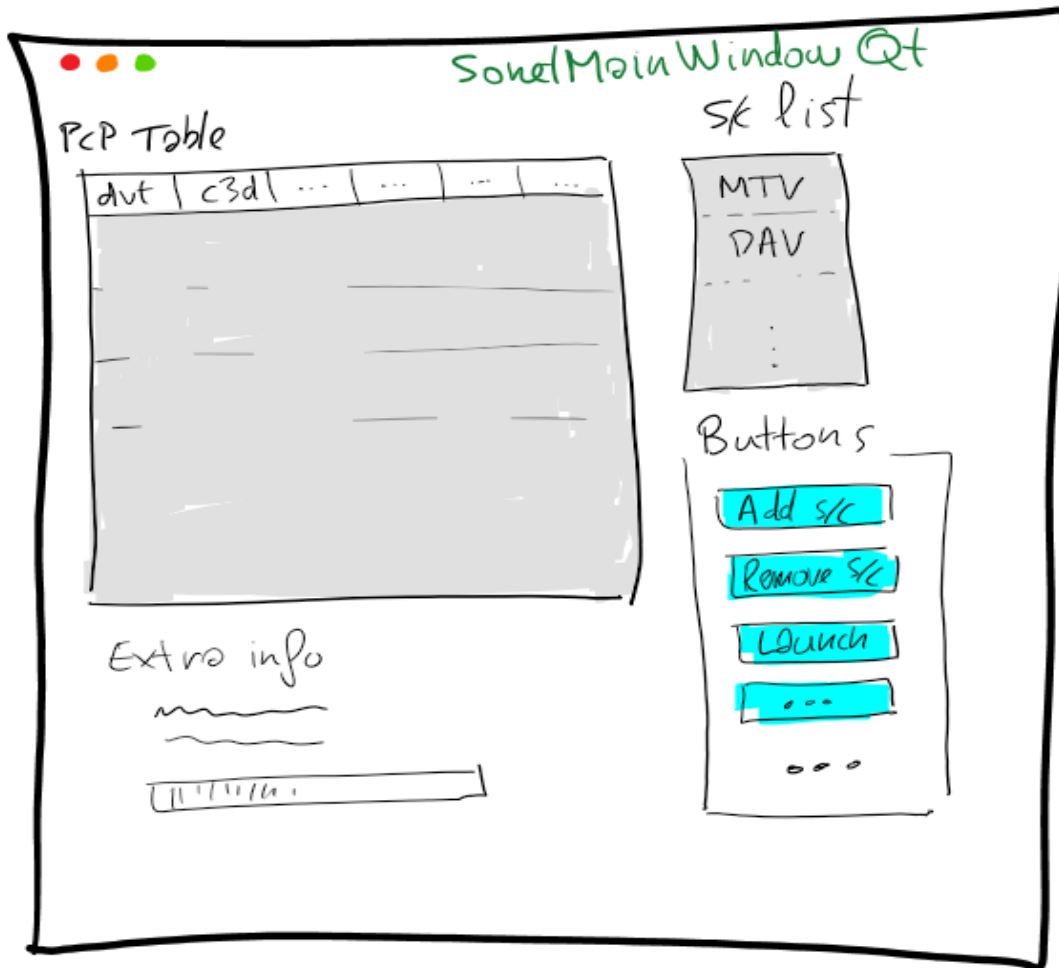


Figure 13. UI Prototype - Main Window.

UI Prototype – PCP Manager Window

The PCP Manager window will raise from the main window. Its main functionality is to manage the trajectories PCP database used while working with the app. It is structured in three areas, according to its intended functionality.

The top group box (a group box is a widget containing other widgets) will host a button to launch the AstroLib's matlab function which invokes a separate window and generates the PCP matrix files.

The center group box will host widgets to load PCP matrix files, and to display information for them. Buttons and line edit widgets (a text field) are used load and see the files paths, respectively. A tree widget is proposed to display the loaded files information. Once loaded, the user will hit another button to proceed to their conversion to PCP tabular form.

The bottom group box will have a similar structure as the center one. As its main functionality is to load the PCP tabular forms which are used during the app execution. Buttons, line edits, and a tree view are used also here.

The UI prototype for the PCP Manager window is shown in Figure 14.

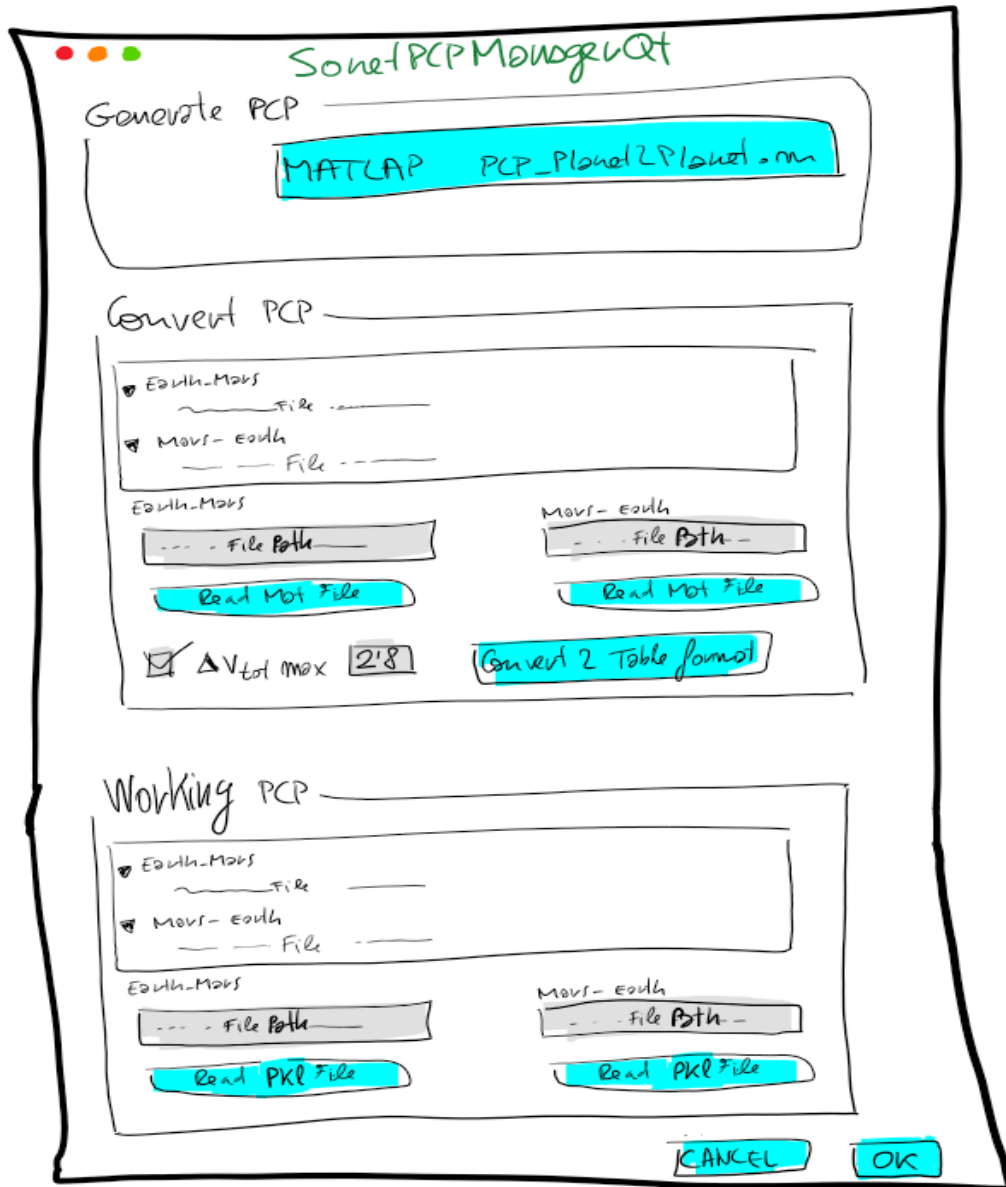


Figure 14. UI Prototype - PCP Manager Window.

UI Prototype – PCP Filter Window

The PCP Filter window will raise from the main window. Its main functionality is to edit the trajectories filter object which lives inside each s/c (SonetTrajectoryFilter class seen in Class Diagram). It is structured in three areas, according to its intended functionality.

On the top, two buttons horizontally aligned, which will serve to select a s/c and one transit (Earth-Mars/Mars-Earth), among the available ones.

On the middle, a group of collapsible (stacked) tabs, each of one will have one of the available filters to add to the s/c + transit selection. Each one of these tabs, will consist in combo boxes, for multi-selection options, and spin boxes, for numerical input. A spin box, is a text field, specialized for numerical input.

At the bottom, the filters table. A table with the current filters applied for the selected s/c + transit selection.

Also, there will be an undetermined number of buttons, conveniently placed, to add functionality, like adding/removing filters, resetting the filter fields, etc.

The UI prototype for the PCP Filter window is shown in Figure 15.

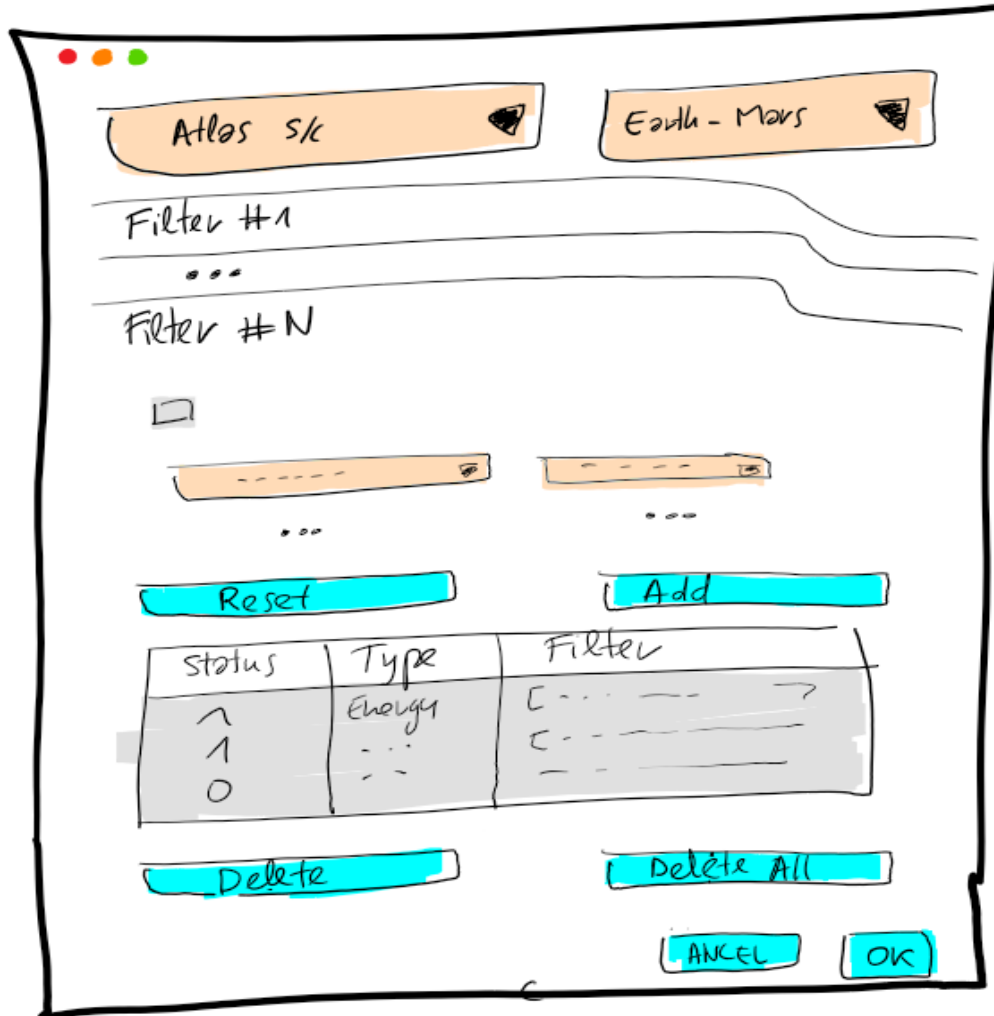


Figure 15. UI Prototype - PCP Filter Window.

UI Prototype – S/C Info Window (also called Canvas Window)

The S/C Info or Canvas window will raise from the main window. Its main functionality is to display a full state overview of the currently selected s/c. It is structured in three tree view areas.

On the top, the S/C Info tree view area where the selected s/c name and type is shown (e.g. s/c ATLAS, type Human). Also, if you see a start (*) next to the type name (e.g. Human*), it means that this s/c has both Earth-Mars, and Mars-Earth transits. If not, it means that it has only Earth-Mars transit. Also, the dependencies (s/c on which this s/c depends on) and dependents (s/c which depend on this s/c) are displayed here, in a tree view manner too. You can see the following symbols next to a dependency/dependent s/c:

- (>>) Relation of the selected s/c E-M⁴ transit with the dependency/dependent s/c E-M transit.
- (><) Relation of the selected s/c E-M transit with the dependency/dependent s/c M-E transit.
- (<>) Relation of the selected s/c M-E transit with the dependency/dependent s/c E-M transit.
- (<<) Relation of the selected s/c M-E transit with the dependency/dependent s/c M-E transit.

In the middle, the Trajectories Filter tree view area, a very similar view as the filters table seen at the bottom of the PCP Filter window, but this one will only display the activated filters (the ones with 1 in the first column), and both transits E-M and M-E are shown in the same tree view.

On the bottom, the Active Trips tree view area, the same structure, but here, the current selected transits are show, if any.

The UI prototype for the PCP S/C Info window is shown in Figure 16.

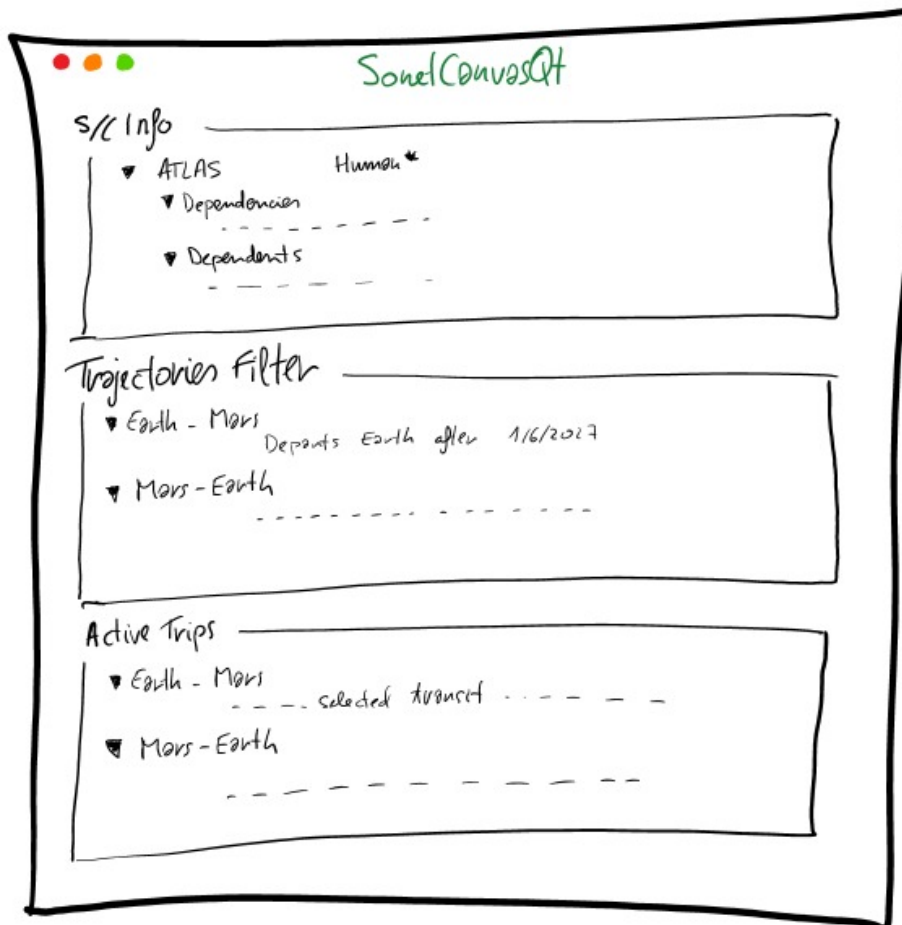


Figure 16. UI Prototype - S/C Viewer (Canvas) Window.

⁴ E-M means Earth-Mars. M-E, Mars-Earth.



Storyboard

A storyboard is a sequence of screens shown with the objective of telling a story, which narrates the sequence of events the user will encounter while using the app. By posing this narrative, the implementation process is facilitated, as the app's interaction with the user remains clear, from a graphical user interface viewpoint.

Although the storyboard is a software design activity prior to the software implementation, for this project, the storyboard technique has been somehow adapted, and placed into the app's **User Manual**, attached to this deliverable.

Style

As the language used to develop the app (Python) and the API used to generate the UI (Qt – PySide2) are both cross platform, it has been decided to leave the app's style the default one for each operative system, resulting in an elegant, non-temporal, easy to maintain UI. See the User Manual, for the actual style of the app, executed on MacOS systems.

Optionally, it will be prototyped (but not finally delivered) the possibility to customize the app's style, by using the Python package **qrainbowstyle**.

Implementation

Architecture

As shown in Class Diagram section, the app follows a sort of monolithic architecture, where the components are highly coupled, in this case, to the main window, from which all other windows are born.

Monolithic architectures cover a variety of use cases, and are, generally considered, a good starting point for small applications, with one or few developers. This becomes specially true when the programmer is novice, and has no prior experience with the programming language, nor implementing architectural patterns.

Although monolithic architecture suffers from scalability and maintainability, one must accept that things won't be perfect and design it anyway. It is left for future app's developers to evolve to a more modular architecture, as could be a layered or microservice patterns.

Also, as the GUI is built on Qt API, which uses a Model-View-Controller (MVC) architectural pattern, it can be also considered that the app features a MVC pattern, in some of its components.

However, by following an OOP philosophy, some good practices have been achieved during coding phase. Examples of this practices are; the single responsibility principle, encapsulation, favor composition over inheritance, avoid code duplication, open/closed principle, on both method and class levels. The reference in this regard has been [18].

Finally, the designed solution consists in; four classes each one representing a different GUI window, two classes representing the s/c and trajectories filter concepts, and three additional classes reimplemented from Qt API, to enable custom table/list views behaviour. The trajectories database, is stored in tabular form in pickle binary files (.pkl) and stored in memory during app execution. This is one of the reasons the app becomes slow with larger database files (+1M trajectories).



Code

The implementation phase is by large the funniest and largest one. The code has 3700 lines, written in Python. The version control system used during the code development is Git, and all the code history is available at https://github.com/horiaghionoiu/sonet_tfm_horia.

Used Technologies

- Python: cross-platform, highly readable and OOP programming language.
- MATLAB: cross-platform, highly readable, matricial computations oriented (also with OOP capabilities) programming language.
- Qt API for Python (PySide2): cross-platform GUI API.
- Qt Designer for UI Design (ui XML files): tool from Qt used to build the UI files (.ui) XML files.
- pyside2-uic: tool from Qt used for converting .ui files to .py files, which are loaded and used during app's execution.

Hardware and Software Resources

At this stage of the project/app development, both developer and user roles would need the same software and hardware environment:

Software:

- Windows, MacOS, Linux OS.
- PyCharm IDE for Python development.

Hardware:

- Any modern workstation. For this project, a MacBook Air 15" (Early 2015) has been used.

Testing

The objective of this section is to perform a battery of tests which cover the most common fail cases identified during development. This effectively increases the the quality of the softwar. This tests are based on the functional requirements defined in the Requirements section.

Although a prototype of automated GUI testing has been built (based on Squish suite), all the following tests are performed manually, and passed or failed by visual inspection of the engineer.

Furthermore, during the implementation phase, special care has been taken to avoid the user to input wrong data to the application. One example would be in the Edit Filters window, the developer will find a number of methods which activate and deactivate the window widgets, based on user interaction, which directs the user to only input correct data to the application.

Test 1	App initialization [REQ_FUN_14]
Preconditions	The development environment described in the User Manual.
Action	Start app.
Verification Point	The main window should be displayed, and no warnings nor errors shown in the console.

Result **PASS**

Test 2	App initialization with a predefined mission [REQ_FUN_1]
Preconditions	The development environment described in the User Manual.
Action	Start app, defining a mission in SonetUtils.build_example_mission method



Verification Point	The main window should be displayed, and no warnings nor errors shown in the console. The predefined mission should be correctly loaded.
Result	PASS
Test 3	Generate new PCP matrix files, limiting the delta-v (total) [REQ_FUN_2, REQ_FUN_3]
Preconditions	App is started.
Action	Launch PCP Manager window. Go to Generate PCP tab. Generate a PCP matrix file.
Verification Point	Inspect the generated file with MATLAB.
Result	PASS
Test 4	Change the app's working PCP files [REQ_FUN_4].
Preconditions	App is started. Have already a PCP matrix file generated.
Action	Launch PCP Manager window. Go to Working PCP tab. Set the working PCP tabular files. Accept the window.
Verification Point	Inspect the resulting trajectories available.
Result	PASS
Test 5	Apply filters to a s/c [REQ_FUN_5]
Preconditions	App is started. At least a s/c has been created.
Action	Launch Edit Filters window. Work with it applying different filters. Accept the window.
Verification Point	Inspect the resulting trajectories left for the s/c.
Result	PASS
Test 6	Apply automatic trajectory filter [REQ_FUN_6]
Preconditions	App is started. At least a s/c has been created.
Action	Launch Edit Filters window. Apply an auto trajectory selection filter. Accept the window.
Verification Point	Inspect the resulting trajectories left for the s/c. A proper trajectory is to be selected, depending on the applied filter.
Result	PASS
Test 7	Select and unselect trajectories for a s/c, from the PCP table in the main window [REQ_FUN_7, REQ_FUN_9]
Preconditions	App is started. At least a s/c has been created.
Action	Click the select and unselect buttons from the main window for a given s/c.
Verification Point	Inspect the results, in the main window, the label and progress bar are updated. In the canvas window, the bottom tree view are updated.
Result	PASS
Test 8	Select a trajectory for a s/c, from the PCP plot viewer. [REQ_FUN_8, REQ_FUN_10]
Preconditions	App is started. At least a s/c has been created.
Action	Launch PCP Viewer window. Click a point on the PCP. Close the window.
Verification Point	Inspect the results, in the main window, the label and progress bar are updated. In the canvas window, the bottom tree view are updated.
Result	PASS



Test 9	Inspect the full state of a s/c in the Canvas window [REQ_FUN_11]
Preconditions	App is started. At least two s/c have been created. The s/c has filters applied, with dependency and dependents conditions. The s/c has transits selected.
Action	Launch Canvas window. Click on the s/c in the main window.
Verification Point	Inspect the Canvas window.
Result	PASS
Test 10	Save the current working session [REQ_FUN_12]
Preconditions	Have a full working session, with several s/c with filters applied, and transits selected.
Action	Save the session through the button in the main window.
Verification Point	Failed, the button isn't implemented.
Result	FAIL
Test N	Add and remove s/c [REQ_FUN_13]
Preconditions	App is started.
Action	Add and remove s/c using the convenient buttons in the main window.
Verification Point	Inspect the s/c list area in the main window.
Result	PASS

Budget

The activities defined within Task Analysis section, which form part of the project WBS are taken into account to compute the project estimated costs. Table 4 summarizes the project overall cost, which is estimated to be around 505000€. Three roles have been defined to compute the rate/hour price. A student salary, for tasks related to programming, but which involve learning basic things which a professional developer would know. A programmer salary, for the tasks related to programming. A software engineer salary, for tasks related to software engineering roles, which could be related to project management, software architecture design, and also programming itself.

Table 4. Budget

WBS code	WBS name	Labour		Materials		Traveling	Rent	Misc	Budget	
		Rate €/hour	Effort hou	Cost €	Amount	Cost per t	Cost			
1,1	Project Start	-		7846,32	1	4000	4000	100	3500	50533,36
1,1,1	Kickoff Meetings		25,11	136						
1,1,2	Python Learning		7	480						
1,1,3	AstroLib Analysis		16,74	64						
1,2	Software Analysis	-		1406,16						
1,2,1	Requirements		25,11	24						
1,2,2	Use Case Diagram		25,11	16						
1,2,3	Use Case Narrative		25,11	16						
1,3	Software Design	-		1740,96						
1,3,1	Class Diagram		25,11	32						
1,3,2	Graphical User Interface	-								
1,3,2,1	UI Prototypes		16,74	32						
1,3,2,2	Storyboard		16,74	16						
1,3,2,3	Style		16,74	8						
1,4	Software Implementation	-		12990,24						
1,4,1	Architecture		25,11	16						
1,4,2	Coding		16,74	472						
1,4,3	Bugs correction		16,74	280						
1,5	Software Testing	-		4285,44						
1,5,1	Test Cases Design		16,74	16						
1,5,2	Test Cases Run		16,74	120						
1,5,3	Bugs correction		16,74	120						
1,6	Project Closing	-		3615,84						
1,6,1	Write Report		25,11	72						
1,6,2	Write User Manual		16,74	48						
1,6,3	Project Defence		25,11	40						
1,7	Project Management	-		11048,4						
1,7,1	Task Analysis		25,11	88						
1,7,2	Scope Statement		25,11	88						
1,7,3	WBS		25,11	88						
1,7,4	Cost Estimate		25,11	88						
1,7,5	Scheduling		25,11	88						
				Sum Cost	42933,36					
	Student Salary		7							
	Programmer Salary		16,74							
	Software Engineer Salary		25,11							



Conclusions

This project covered the entire design, construction, and release of a software artifact written in Python, which features a graphical user interface, and a MATLAB astrodynamics core. The software is now ready to be used on preliminary interplanetary Mars mission design tasks, and the codebase of over 3700 lines is available to be further extended in the future. My focus was the development of an intuitive, comprehensive, proven and documented tool that others with appropriate training and experience could use productively with confidence, we will see in the future if that goal was achieved.

I feel very happy of submitting this project. In it, I have spent (invested) nine years of my life. Now, I officially finish my formal learning process in the university, and I begin a larger one in the life, this one with no written exams, but full of challenges, wins and losses, friends, enemies, lessons learned, and a large etcetera.

References

- [1] D. d. I. T. Sangrà, Optimization of Interplanetary Trajectories with Gravity Assist, Barcelona, 2020.
- [2] C. D. Brown, Spacecraft Mission Design 2nd Edition..
- [3] A. Ribera, La astronàutica, Barcelona: Plaza & Janes editores, 1973.
- [4] NASA, «Why do we Explore?,» [En línia]. Available: <https://www.nasa.gov/feature/the-human-desire-for-exploration-leads-to-discovery>.
- [5] W. V. Braun, The Mars Project, Illinois: University of Illinois Press, 1953.
- [6] NASA, Human Exploration of Mars Design Reference Architecture, 2009.
- [7] NASA, «NASA's Lunar Exploration Program Overview,» 2020.
- [8] E. A. Stephen Kemble, «Interplanetary Mission Analysis».
- [9] NASA, «GMAT - General Mission Analysis Tool,» [En línia]. Available: <https://sourceforge.net/projects/gmat/>.
- [10] NASA, «MALTO - Mission Analysis Low-Thrust Optimizer,» [En línia]. Available: <https://software.nasa.gov/software/NPO-43625-1>.
- [11] NASA, «NASA Space Mission Design Tools,» [En línia]. Available: <https://www.nasa.gov/smallsat-institute/space-mission-design-tools>.
- [12] NASA, «NASA Open-Source Projects,» [En línia]. Available: <https://code.nasa.gov/>.
- [13] NASA, «Copernicus Trajectory Design and Optimization System,» [En línia]. Available: <https://www.nasa.gov/centers/johnson/copernicus/index.html>.
- [14] NASA-JPL, «MONTE - Mission Analysis, Operations, and Navigation Toolkit Environment,» [En línia]. Available: <https://montepy.jpl.nasa.gov/>.
- [15] spaceflightsolutions, «MANE Software,» [En línia]. Available: spaceflightsolutions.com.
- [16] L. K. L.E. George, Interplanetary Mission Design Handbook: Earth-to-Mars Mission Opportunities and Mars-to-Earth Return Opportunities 2009–2024, NASA, 2009.
- [17] R. D. F. a. M. L. M. Laura M. Burke, Interplanetary Mission Design Handbook: Earth-to-Mars Mission Opportunities 2026 to 2045, Cleveland, Ohio: Glenn Research Center, 2010.
- [18] A. Shvets, Dive Into Design Patterns.

User Guide

Introduction

This user manual aims at describing the functionality of the program SONet Mars Mission Flight Sequence Planner, from now on **SONetMarsAPP**. It covers the installation of the program, the description of its GUI and briefly introduces tutorials to the user..

Target audience

It is assumed that the user has a certain knowledge of astrodynamics and space mission planning. In the same way, some basic computer skills are also needed, which include the installation of the Python IDE and its libraries and executing scripts.

Software Description

The main purpose of the application is to help on planning the sequence of launches, arrival/landings and transit of spacecrafts in Mars-Earth interplanetary missions, specially those aimed at establishing long-term human settlements in Mars. In this regard, it allows the user a faster accomplishment of the preliminary design of the missions in comparison with other traditionally manual methods.

The application is capable of computing the interplanetary trajectories between the Earth and Mars, which are represented in the so-called Porkchop Plots (PCP). It allows also to save these diagrams and work with them.

The user can work on the PCPs, creating missions which are composed of several spacecrafts and applying specific requirements to each one of them, thus setting up a preliminary scheme/map/diagram of the interplanetary transits. This could be later used to propose more complex setups, or also the application functionality could be extended, as its source code is published.

The generation and visualization of the PCPs is done by third-party library called AstroLib, which was developed in MATLAB and kindly handed by the Dr. D. de la Torre Sangrà [1]. This application is built around the mentioned library, thus adding a new functionality to it.

Installing The Software

The application is designed to be executed on desktop systems which run on Windows, MacOS, or Linux OS⁵. For generating this User Manual , the application ran on Apple MacBook Air 15" (early 2015 version).

In order to run this application, a development environment based on Python, MATLAB and PyCharm is needed. The author generated a script for automatizing this process, so that the user only has to run an executable. This process served as a proof-of-concept and unfortunately is not detailed in the report nor provided the file itself.

Following, the development environment installation process is detailed.

⁵ It hasn't been tested on Linux OS.

MATLAB

It is necessary to have the MATLAB environment installed⁶, as the application has functions of it embodied, which are called. A MATLAB license is needed for the correct function. It would be possible to just compile the MATLAB code and call the binary executables from Python and not having the need of the license. This user manual does not cover the MATLAB installation and configuration, as this topic is generic and widely covered in the Mathworks documentation⁷.

Python

To run the application, it is necessary to have installed the Python programming language in its 3.7⁸ version.

PyCharm

It is highly recommended having installed an Integrated Development Environment (IDE) from which the application can be run. PyCharm is recommended⁹. The installation of it is not covered as it is also generic¹⁰. The configuration is treated in hereafter

· At this stage, it's assumed you've successfully installed MATLAB, Python, and PyCharm in your workstation.

Getting The Code

The code is open source and it's available online¹¹, you can get it from GitHub¹². Following, is explained how to get it by cloning the online repository into your computer. An alternative procedure found in the annex, in order to to get the code ([Download Code From GitHub \(zip file\)](#)).

- **Step 1** Open PyCharm. You see the Welcome window. Click Get from Version Control. See [Figure 17](#).
- **Step 2** Select Repository URL> Git. Fill the fields as in [Figure 18](#), and click Clone.
- **Step 3** After a while, you should see the code available in PyCharm. **However** click File > Close Project, to go to the Welcome window again. See [Figure 20](#). Now, click on the project.
- **Step 4** Now, you should finally see ([Figure 20](#)) PyCharm, with the code to the left, and at the bottom right corner, a Python interpreter automatically configured (you should see Python 3.7, although in the figure is 3.9 seen).
- **Step 5** Congratulations! You have cloned the Git repository from GitHub (online) to your local computer. Now, your code is synched with the one in the master branch of the official code repository¹³

⁶ It has been tested that MATLAB R2020a version has an acceptable behaviour.

⁷ <https://es.mathworks.com/help/install/install-products.html>

⁸ Available on <https://www.python.org/downloads/release/python-370/>. It's also possible to run the app with other Python versions (3.x), but they hadn't been tested.

⁹ Gratis en su versión Community. Versión Professional gratis para estudiantes y empleados de la UPC. Durante el desarrollo de este proyecto, se ha usado la versión Community y Professional, 2020.x.

¹⁰ <https://www.jetbrains.com/es-es/pycharm/download/>.

¹¹ It is protected by no licence, but if you use it, it would be nice to name the author, me 😊.

¹² https://github.com/horiaghionoiu/sonet_tfm_horia.

¹³ Check <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control> to learn more about Git Control Versioning System.

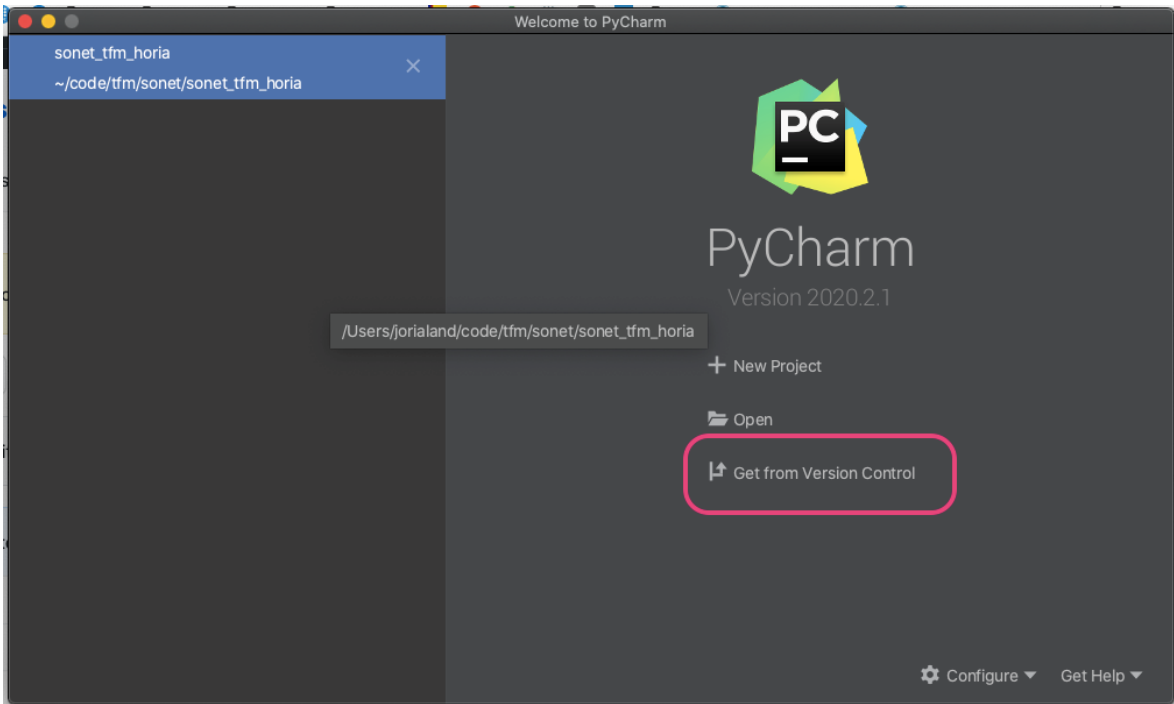


Figure 17. Get the code from Version Control (GitHub).

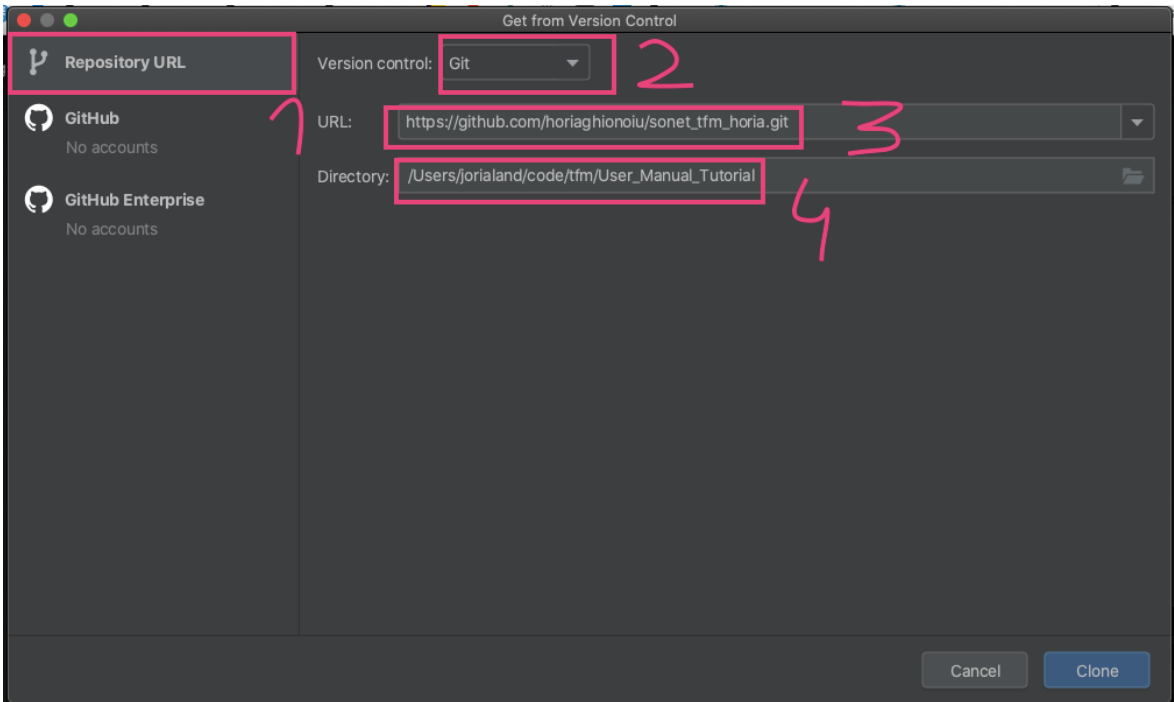


Figure 18. Repository URL configuration.

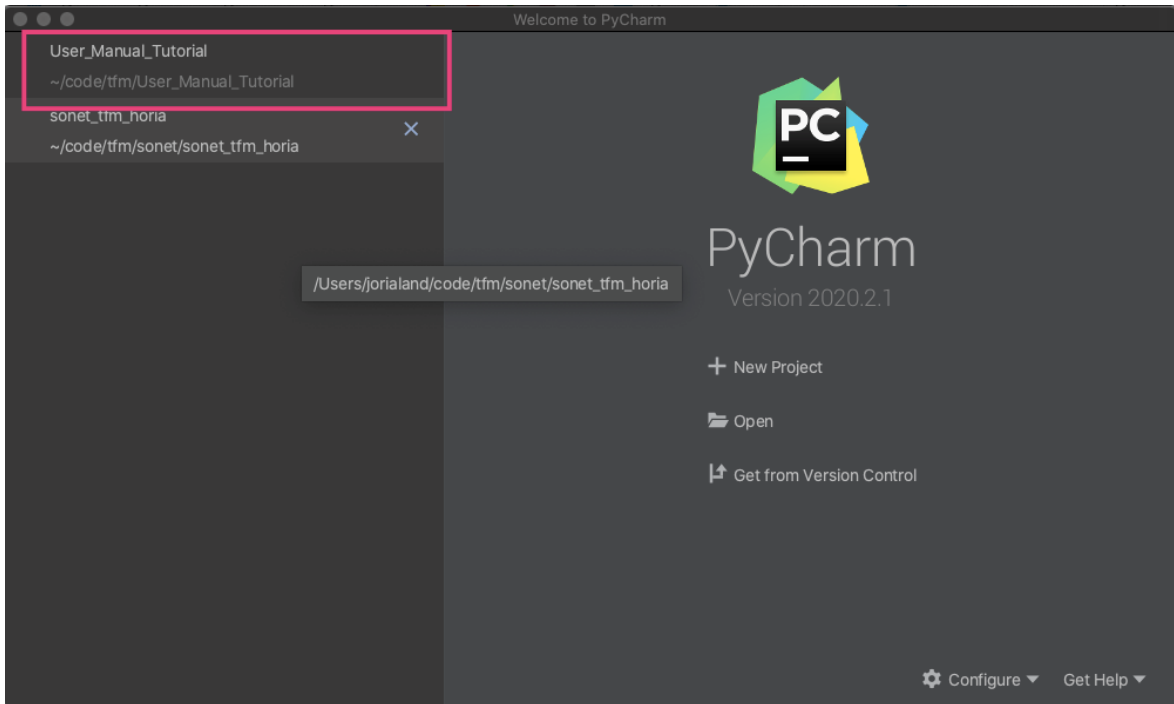


Figure 19. Code has been cloned into your computer, now you have a local workcopy.

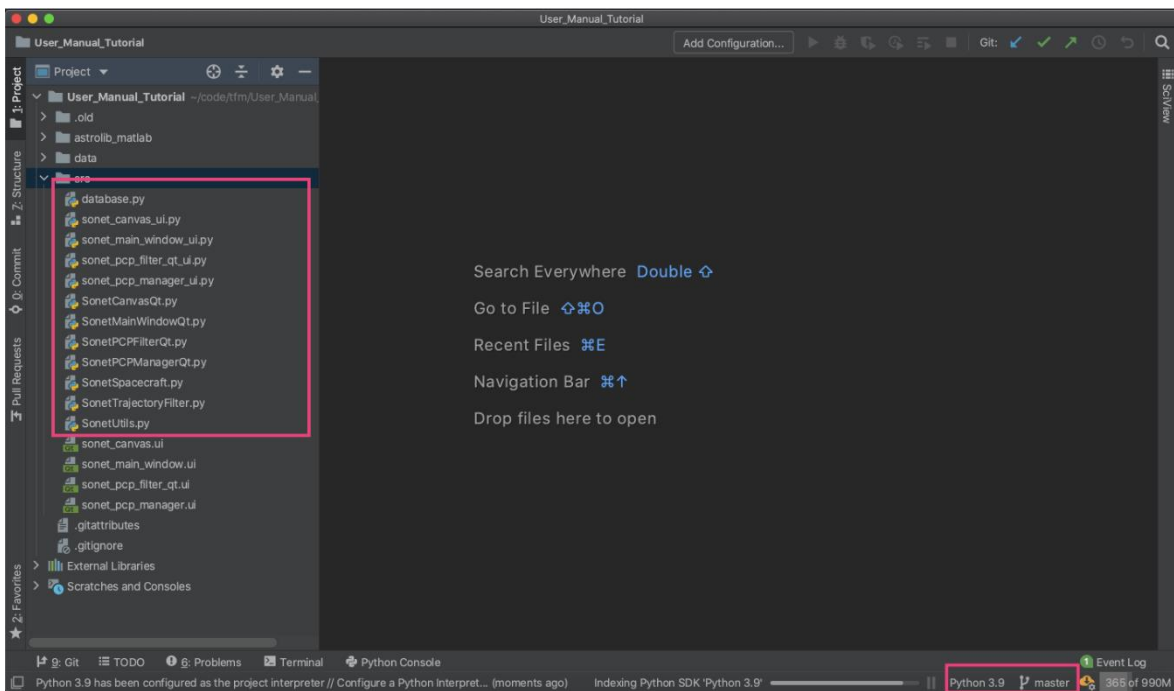


Figure 20. PyCharm overview with the code downloaded.

Configuring the IDE (PyCharm)

Now, we will add the required Python packages. To do so, we will first create a Virtual environment (Virtualenv), which is a sort of isolated container, where all installed Python packages live, and they are only available for this PyCharm project¹⁴.

- **Step 1** Creating the Virtualenv.
Add interpreter (yellow arrow in [Figure 21](#)). Configure it as seen in [Figure 22](#). OK.
- **Step 2** Installing the Packages.
Interpreter Settings (green arrow in [Figure 21](#)). Install new packages by clicking on the + cross on the bottom window area ([Figure 23](#)). Look for PySide2, pandas, and scipy packages, and install them ([Figure 24](#)). **Be advised!** PyCharm will identify packages dependencies and will end up installing more packages than the ones asked for, don't care about this, all is fine.
- **Step 3** Installing MATLAB for Python (separate package, not from PyCharm).
To install the MATLAB package for Python (named matlabengineforpython) we will proceed differently, as it is recommended in the official Mathworks documentation¹⁵. This process is graphically explained in [Figure 25](#). Open a terminal within PyCharm (marked as step 0 in [Figure 25](#)), check that (<Virtualenv name>) is between parentheses at the left hand side (step 1 in yellow), and execute the two commands marked in pink squares in steps 2 and 3. Once the process is done, you should see matlabengineforpython successfully installed within the Interpreter settings, as displayed in [Figure 26](#).
- **Step 4** Congratulations! You have just installed the environment needed to develop (or just run) the application.

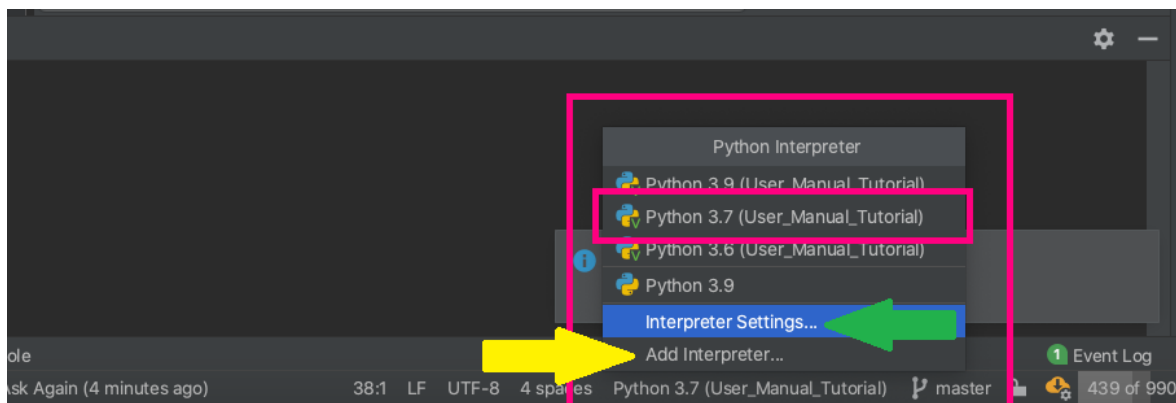


Figure 21. PyCharm Python interpreter settings. Located at the bottom right area.

¹⁴ This is specially useful when playing around with various projects in the same computer, as it avoids mixing configurations from one project to another, keeping the workcopy clean and safe of incidents.

¹⁵ https://es.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html.

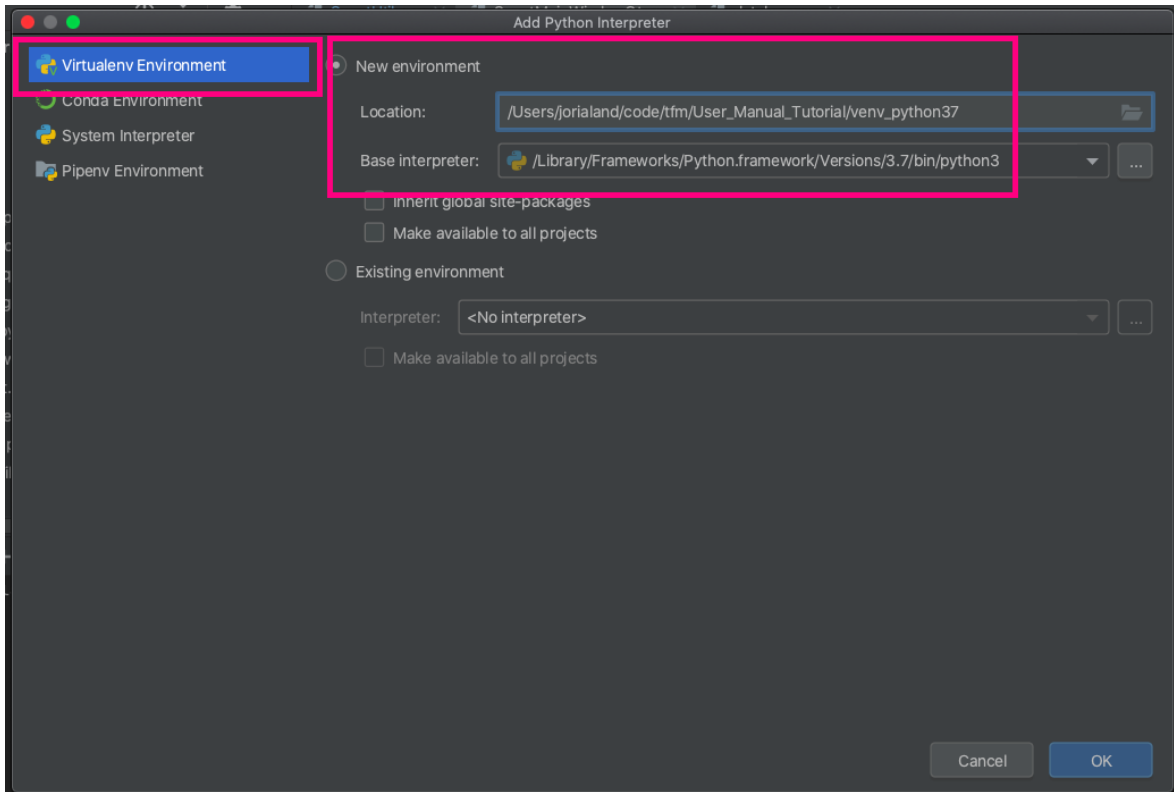


Figure 22. Adding a Python 3.7 interpreter, into a new Virtualenv.

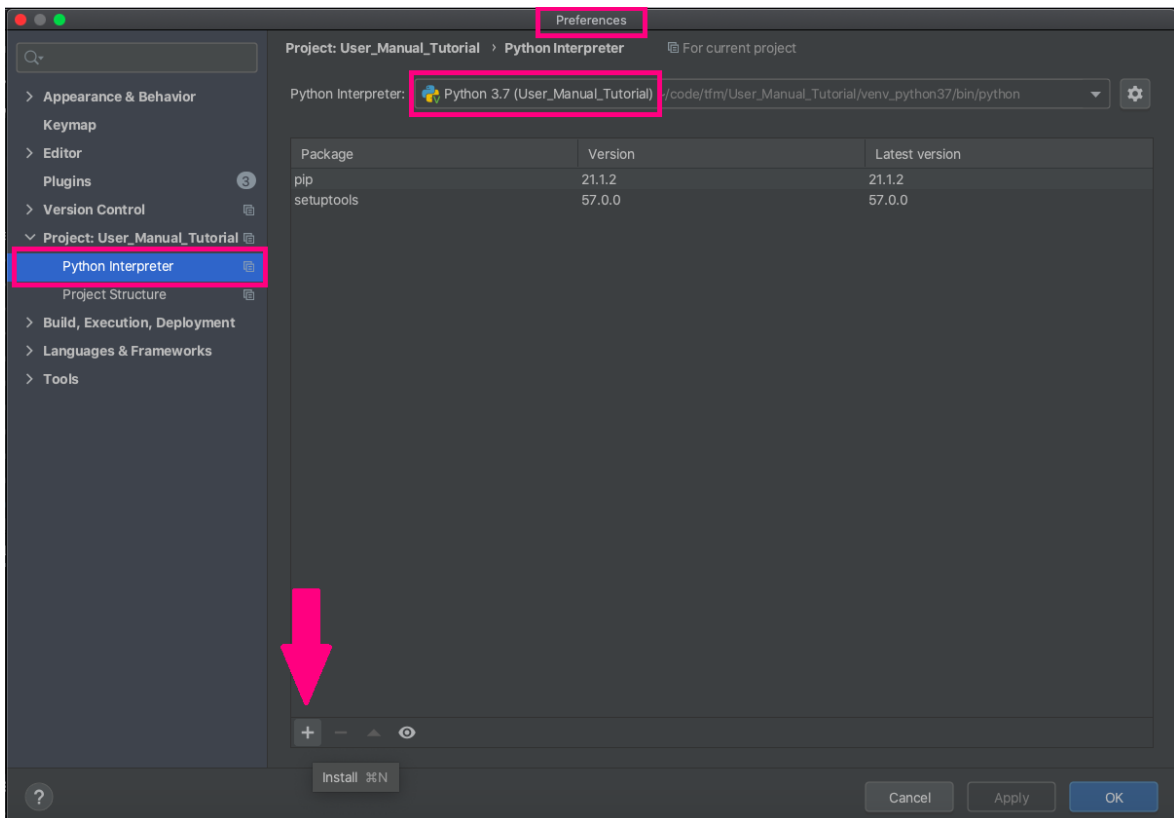


Figure 23. Install new Python packages, into the created Virtualenv.

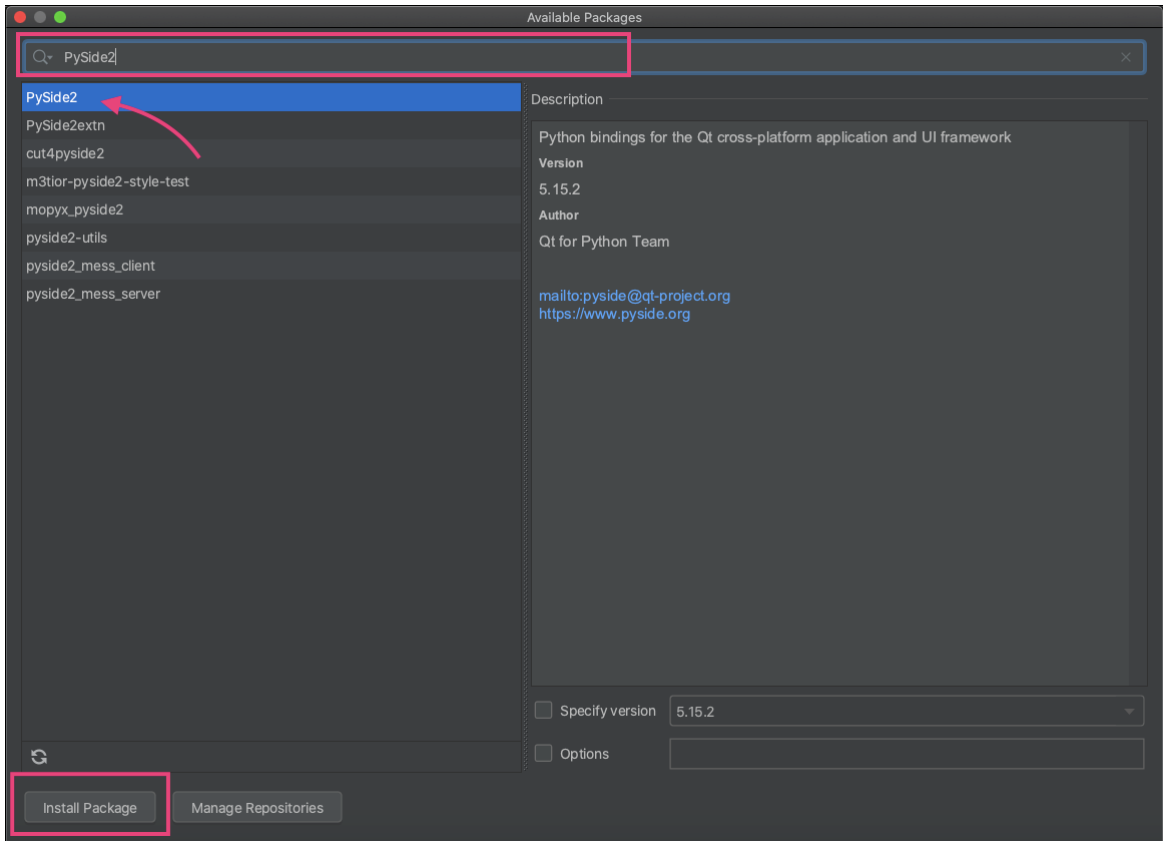


Figure 24. Installing PySide2 package. Do the same for pandas, and scipy packages.

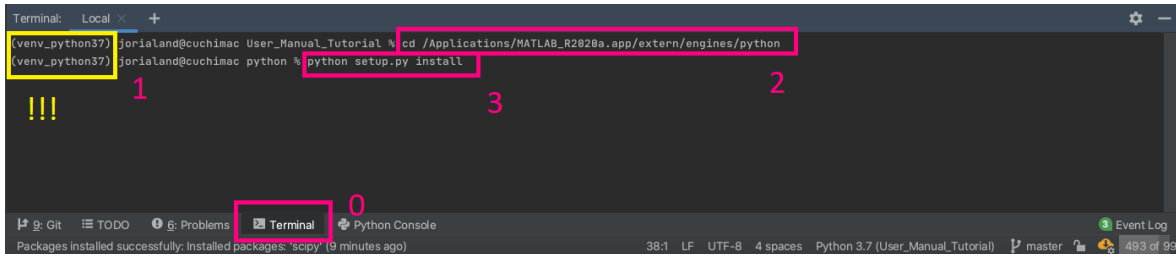


Figure 25. Install matlabengineforpython package. Be sure that (your Virtualenv Python name) appears at the left side of each terminal lines, which indicates that your are operating with our new installed Virtualenv. If it doesn't appears, close and open the Terminal again.

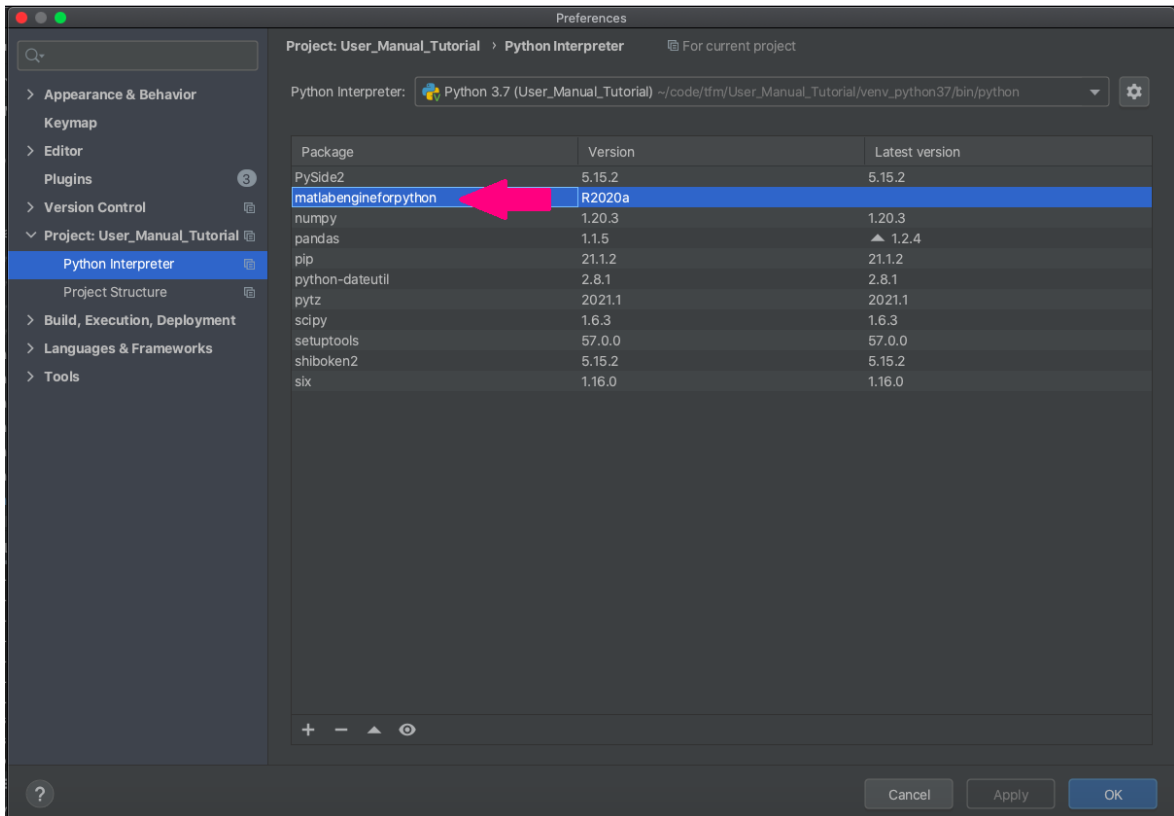


Figure 26. matlabengineforpython package installed.

Other Configurations

Configuring the paths

The application needs a path to your workcopy to be set. See [Figure 27](#), the SONET_DIR variable should be set according to your local path to your workcopy, where all the Python code lives.

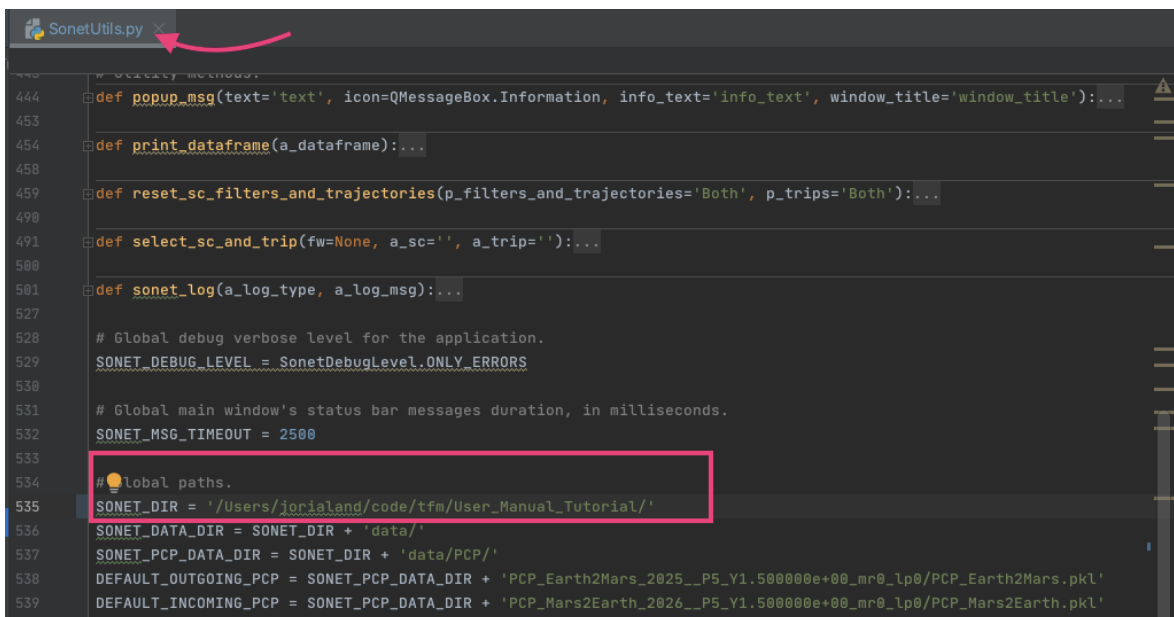


Figure 27. Configuring the paths.

Test The Installation

Congratulations! You have successfully installed the application, and now you can proceed to run it and design missions to Mars! If you experienced problems during the installation, just post a message on the GitHub repository associated to this application development¹⁶, I will revise it and help you as soon as I can.

Finally, to run the app, just proceed with the step shown in [Figure 28](#). The app will hopefully initialize ([Figure 29](#)) and finally, you should see its main window, as shown in [Figure 30](#).

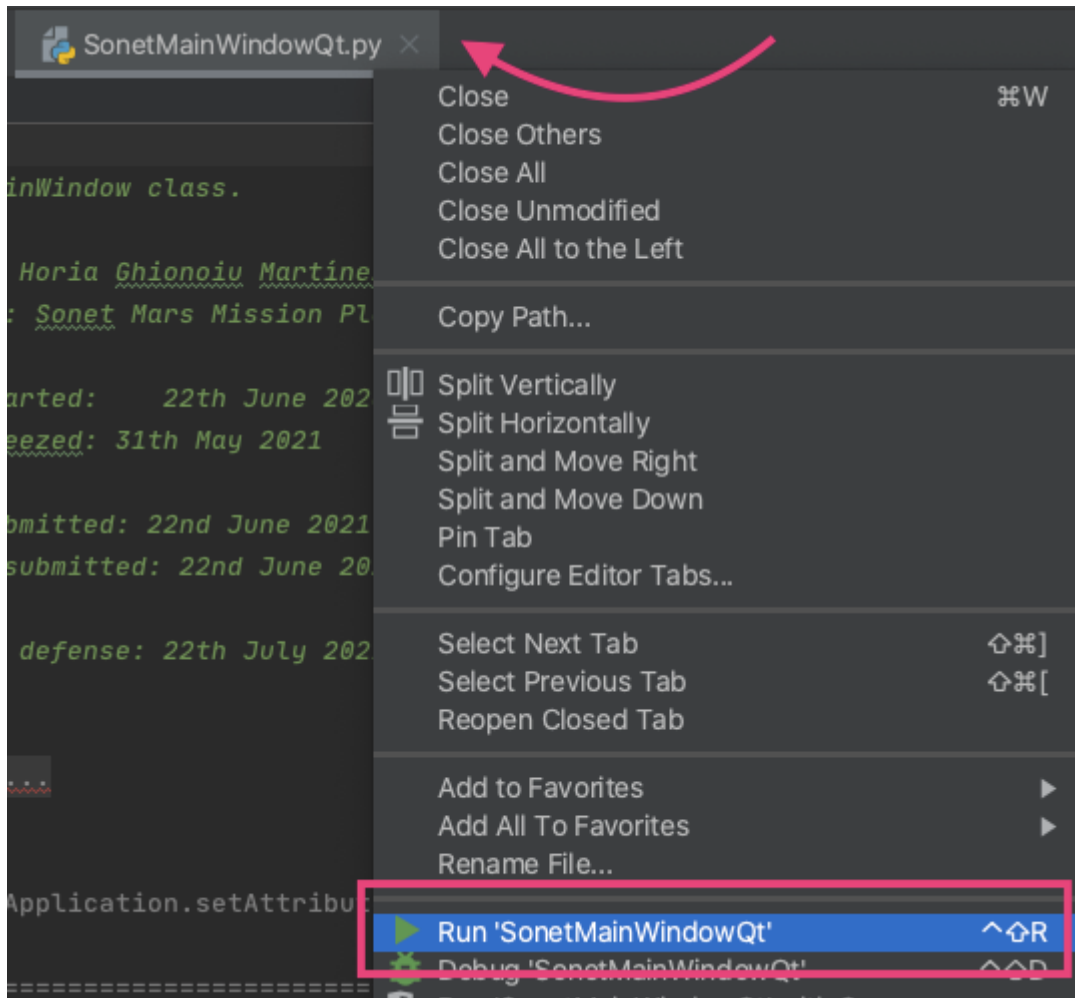


Figure 28. Running the app.

¹⁶ https://github.com/horiaghionoiu/sonet_tfm_horia.

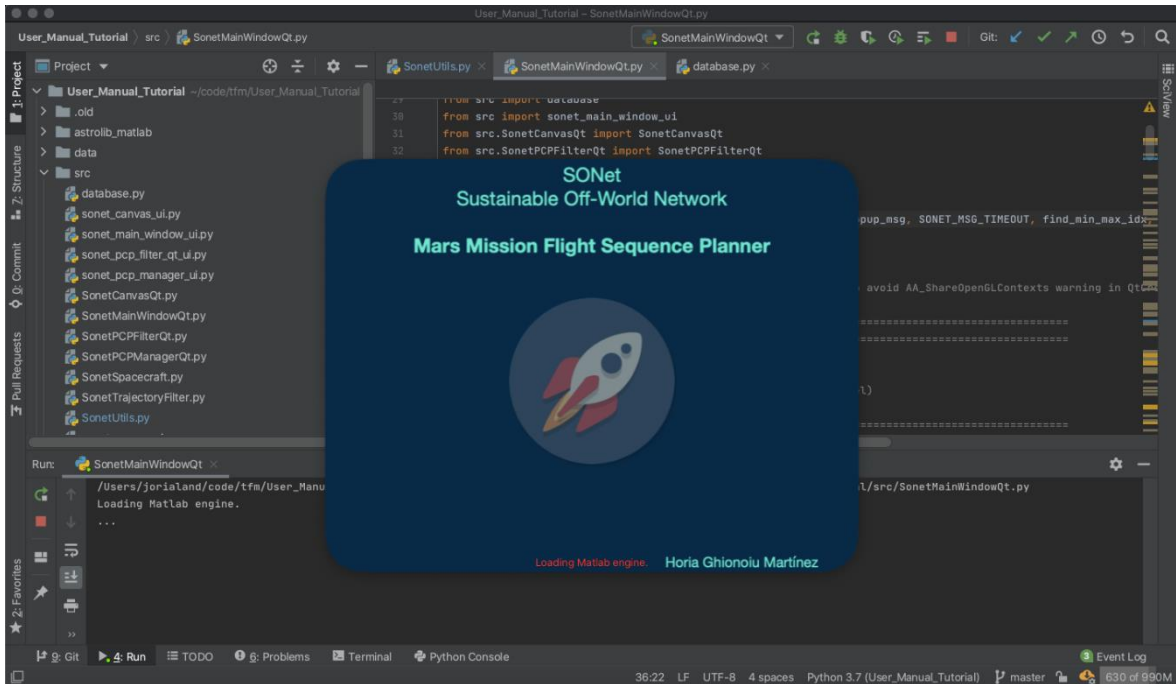


Figure 29. App welcome splash image. The MATLAB engine takes a while to load.

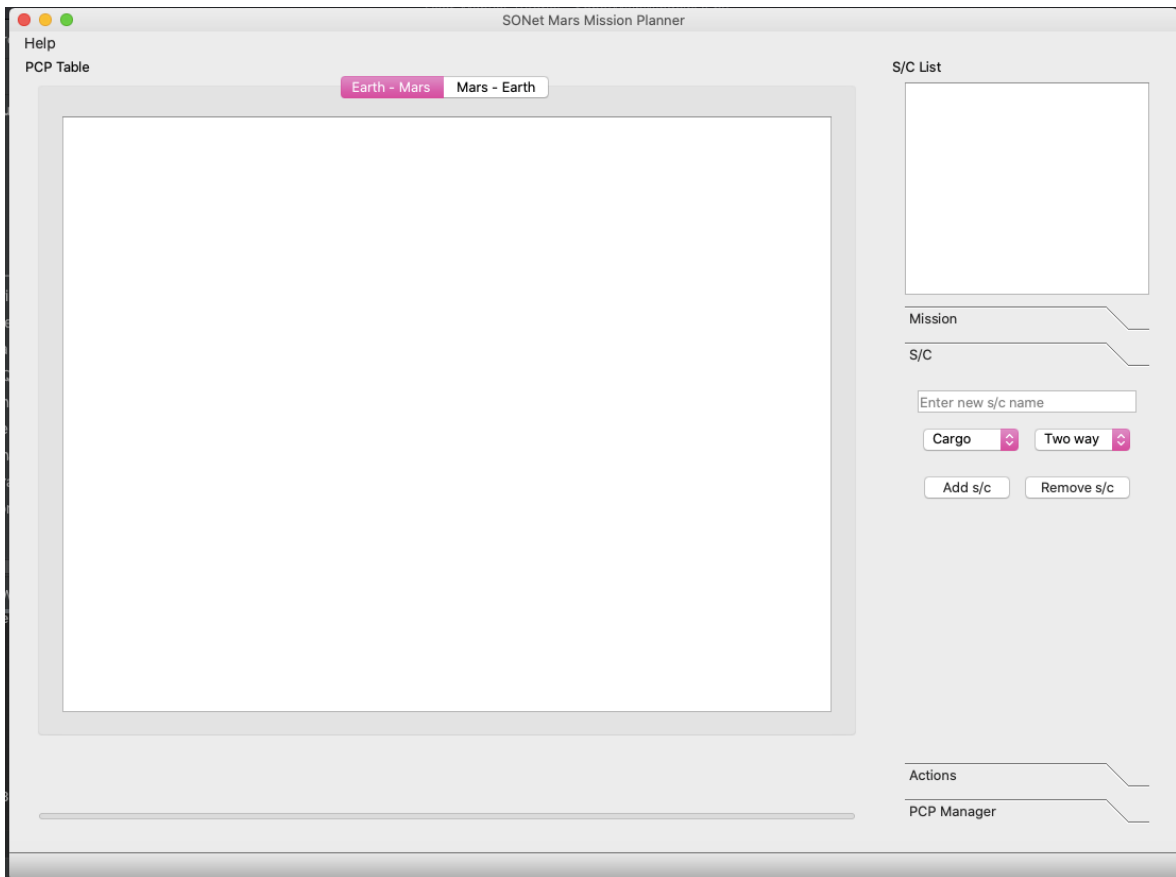


Figure 30. SONet Mars Mission Planner main window.

Using The Software

Graphical User Interface

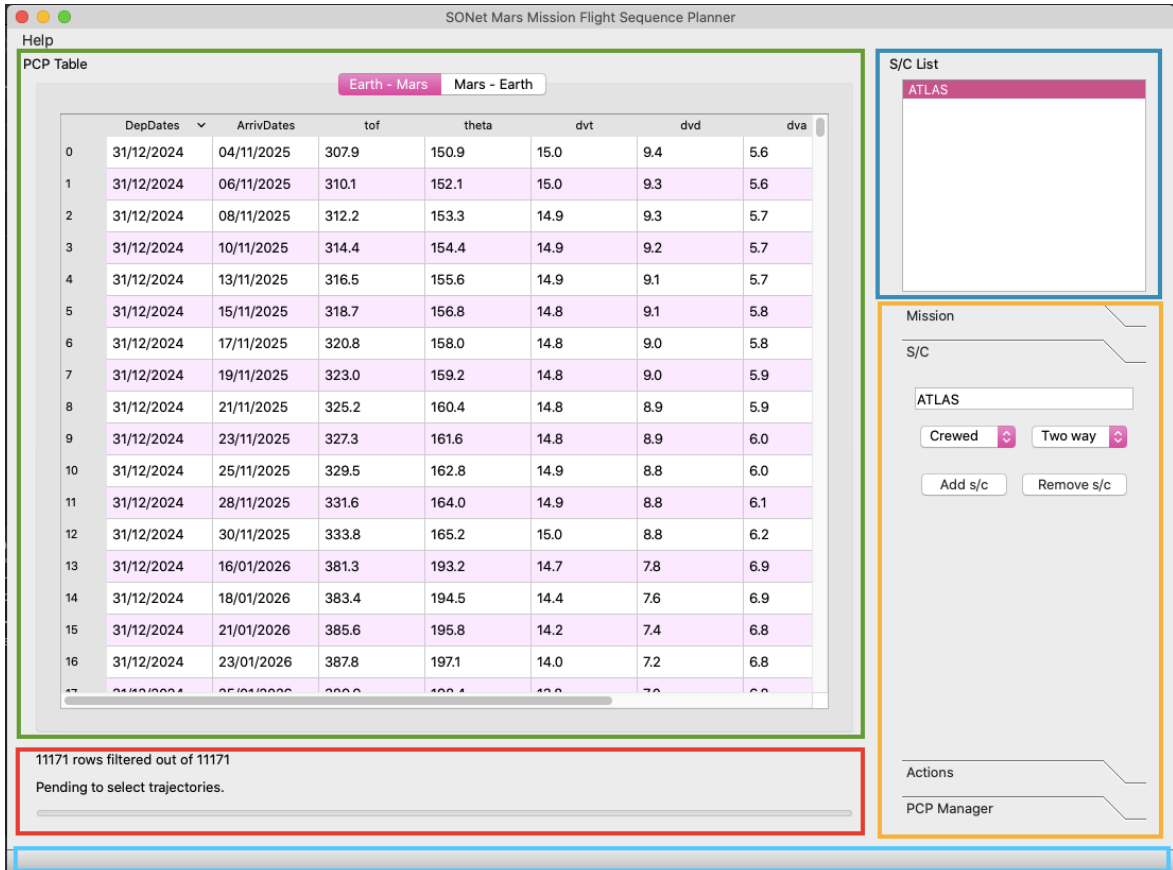
In this section, the app's Graphical User Interface, GUI from now on, is described. This description is from a user viewpoint, for technical details about its design and implementation, check out the Software Engineering section.

The app can be understood by reviewing the four windows it's composed of, named; (1) [Main Window](#), (2) [Canvas Window](#), (3) [Edit filters Window](#), and (4) [PCP Manager Window](#).

Main Window

It is the first window that is shown when running the app. It can be considered the Mother window, from which other windows are generated. It's the only window from where the s/c can be created or removed. [Figure 31](#) shows a general view of the main window. It can be divided into five differentiated sections:

- [PCP Table](#) (green) Current trajectories available for the selected s/c.
- [S/C List](#) (blue) Current s/c list.
- [Buttons](#) (yellow) Buttons which do different actions.
- [S/C Info](#) (red) Information for the selected s/c.
- [Status bar](#) (cyan) Eventual information when performing some actions.



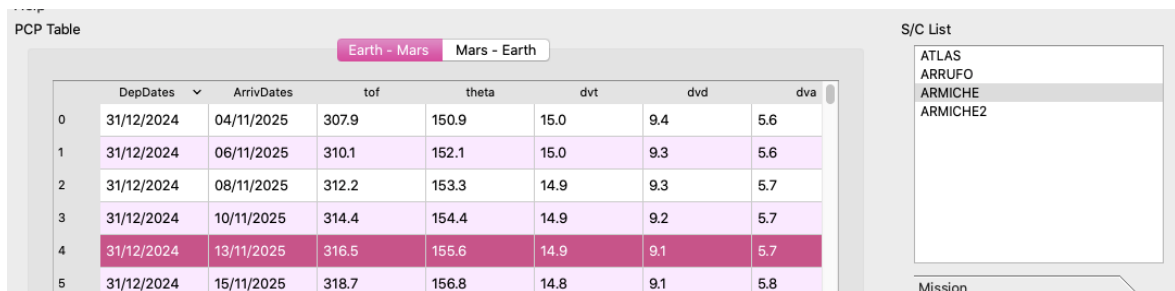
	DepDates	ArrivDates	tof	theta	dvt	dvd	dva
0	31/12/2024	04/11/2025	307.9	150.9	15.0	9.4	5.6
1	31/12/2024	06/11/2025	310.1	152.1	15.0	9.3	5.6
2	31/12/2024	08/11/2025	312.2	153.3	14.9	9.3	5.7
3	31/12/2024	10/11/2025	314.4	154.4	14.9	9.2	5.7
4	31/12/2024	13/11/2025	316.5	155.6	14.9	9.1	5.7
5	31/12/2024	15/11/2025	318.7	156.8	14.8	9.1	5.8
6	31/12/2024	17/11/2025	320.8	158.0	14.8	9.0	5.8
7	31/12/2024	19/11/2025	323.0	159.2	14.8	9.0	5.9
8	31/12/2024	21/11/2025	325.2	160.4	14.8	8.9	5.9
9	31/12/2024	23/11/2025	327.3	161.6	14.8	8.9	6.0
10	31/12/2024	25/11/2025	329.5	162.8	14.9	8.8	6.0
11	31/12/2024	28/11/2025	331.6	164.0	14.9	8.8	6.1
12	31/12/2024	30/11/2025	333.8	165.2	15.0	8.8	6.2
13	31/12/2024	16/01/2026	381.3	193.2	14.7	7.8	6.9
14	31/12/2024	18/01/2026	383.4	194.5	14.4	7.6	6.9
15	31/12/2024	21/01/2026	385.6	195.8	14.2	7.4	6.8
16	31/12/2024	23/01/2026	387.8	197.1	14.0	7.2	6.8

Figure 31. Main window general view.

PCP Table

It's surrounded by the green rectangle in [Figure 31](#). It consists of two table views, one for Earth-Mars transits, and a second one for Mars-Earth ones. You can change between the Earth-Mars / Mars-Earth table by clicking on the top tabs named accordingly.

The way the PCP Table is meant to be used is (see [Figure 32](#)) first selecting a s/c from the [S/C List](#) and then inspect its current available trajectories displayed in the PCP Table area. Consider that each time you click on a s/c, its trajectories are calculated, for large databases (1M+ trajectories), this may take a while¹⁷. You can see that row 4 has been selected, now you can mark this trajectory as the one selected for the current selected s/c (seen in [Actions Tab](#)).



The screenshot shows the PCP Table interface with two tabs: 'Earth - Mars' (selected) and 'Mars - Earth'. The table displays the following data:

	DepDates	ArrivDates	tof	theta	dvt	dvd	dva
0	31/12/2024	04/11/2025	307.9	150.9	15.0	9.4	5.6
1	31/12/2024	06/11/2025	310.1	152.1	15.0	9.3	5.6
2	31/12/2024	08/11/2025	312.2	153.3	14.9	9.3	5.7
3	31/12/2024	10/11/2025	314.4	154.4	14.9	9.2	5.7
4	31/12/2024	13/11/2025	316.5	155.6	14.9	9.1	5.7
5	31/12/2024	15/11/2025	318.7	156.8	14.8	9.1	5.8

On the right, the S/C List shows the following entries: ATLAS, ARRUF0, ARMICHE, and ARMICHE2. The 'Mission' field is empty.

Figure 32. Using the PCP Table.

Depending on the state of the selected s/c you can see different things in the PCP Table. [Figure 33](#) resumes the three possible states. From top to bottom:

- You see some available trajectories.
- You see no table. The selected s/c has no Mars-Earth transit (e.g. just goes and stays in Mars)¹⁸.
- You see zero available trajectories (e.g. you applied a too restrictive filter to the selected s/c transit).

¹⁷ The app example used while writing this User Manual was using a database of approx 12k trajectories, and the user experience had no delay. For larger databases you may experience some delay when clicking on a s/c.

¹⁸ You'll never see a Earth-Mars transit with no table, as it has not considered in the app logic a s/c which launches from other planet than Earth 😊.

PCP Table

Earth - Mars Mars - Earth

	DepDates	ArrivDates	tof	theta	dvt	dvd	dva
0	31/12/2024	04/11/2025	307.9	150.9	15.0	9.4	5.6
1	31/12/2024	06/11/2025	310.1	152.1	15.0	9.3	5.6
2	31/12/2024	08/11/2025	312.2	153.3	14.9	9.3	5.7
3	31/12/2024	10/11/2025	314.4	154.4	14.9	9.2	5.7
4	31/12/2024	13/11/2025	316.5	155.6	14.9	9.1	5.7

PCP Table

Earth - Mars Mars - Earth

PCP Table

Earth - Mars Mars - Earth

	DepDates	ArrivDates	tof	theta	dvt	dvd	dva
--	----------	------------	-----	-------	-----	-----	-----

Figure 33. PCP Table navigation, inspecting Earth-Mars & Mars-Earth transits for a one-way s/c. The bottom image is the first one, but with a too restrictive filter applied, resulting in zero available trajectories.

S/C List

It's surrounded by the blue rectangle in [Figure 31](#). It consists of a list view. Its main function is to display a list of all the s/c. The s/c you create will appear in the list, and the ones you remove (seen in [S/C Tab](#)) will disappear. Also, when you select a s/c in the list, the rest of the app will update showing information for this s/c¹⁹.

Buttons

It's surrounded by the yellow rectangle in [Figure 31](#). It's the area where all the main window's buttons live, grouped under four tabs, which are explained following.

Mission Tab

As seen in [Figure 34](#), there are three buttons. 'View mission' button executes the [Canvas Window](#). The other two buttons are supposed to save/load a mission, but their functionality isn't implemented yet. See Testing section, where a requirement is failed due to this.

¹⁹ If you are working with a large trajectories database (close and more than 1M trajectories), after clicking on a s/c the app may stay irresponsive for a seconds, while calculating the trajectories to be displayed.

	DepDates	ArrivDates	tof	theta	dvt	dvd	dva
0	31/12/2024	04/11/2025	307.9	150.9	15.0	9.4	5.6
1	31/12/2024	06/11/2025	310.1	152.1	15.0	9.3	5.6
2	31/12/2024	08/11/2025	312.2	153.3	14.9	9.3	5.7
3	31/12/2024	10/11/2025	314.4	154.4	14.9	9.2	5.7
4	31/12/2024	13/11/2025	316.5	155.6	14.9	9.1	5.7
5	31/12/2024	15/11/2025	318.7	156.8	14.8	9.1	5.8
6	31/12/2024	17/11/2025	320.8	158.0	14.8	9.0	5.8
7	31/12/2024	19/11/2025	323.0	159.2	14.8	9.0	5.9
8	31/12/2024	21/11/2025	325.2	160.4	14.8	8.9	5.9
9	31/12/2024	23/11/2025	327.3	161.6	14.8	8.9	6.0
10	31/12/2024	25/11/2025	329.5	162.8	14.9	8.8	6.0
11	31/12/2024	28/11/2025	331.6	164.0	14.9	8.8	6.1
12	31/12/2024	30/11/2025	333.8	165.2	15.0	8.8	6.2
13	31/12/2024	16/01/2026	381.3	193.2	14.7	7.8	6.9
14	31/12/2024	18/01/2026	383.4	194.5	14.4	7.6	6.9
15	31/12/2024	21/01/2026	385.6	195.8	14.2	7.4	6.8
16	31/12/2024	23/01/2026	387.8	197.1	14.0	7.2	6.8
17	31/12/2024	25/01/2026	390.0	198.4	13.8	7.0	6.8

Figure 34. Main window Mission Tab.

S/C Tab

This tab serves for creating/removing s/c. As seen in [Figure 35](#), there are four widgets, arranged in three lines.

- A text box, where you write the name of the s/c.
- Two combos, where you specify if you want to create a Cargo/Crewed s/c and a One way/Two way s/c. Any combination is possible.
- Two buttons, to add a new s/c, and another, to remove the current selected s/c in the S/C List. If no s/c is selected, the last added is removed. If no s/c are left, then a message is shown in the [Status bar](#).

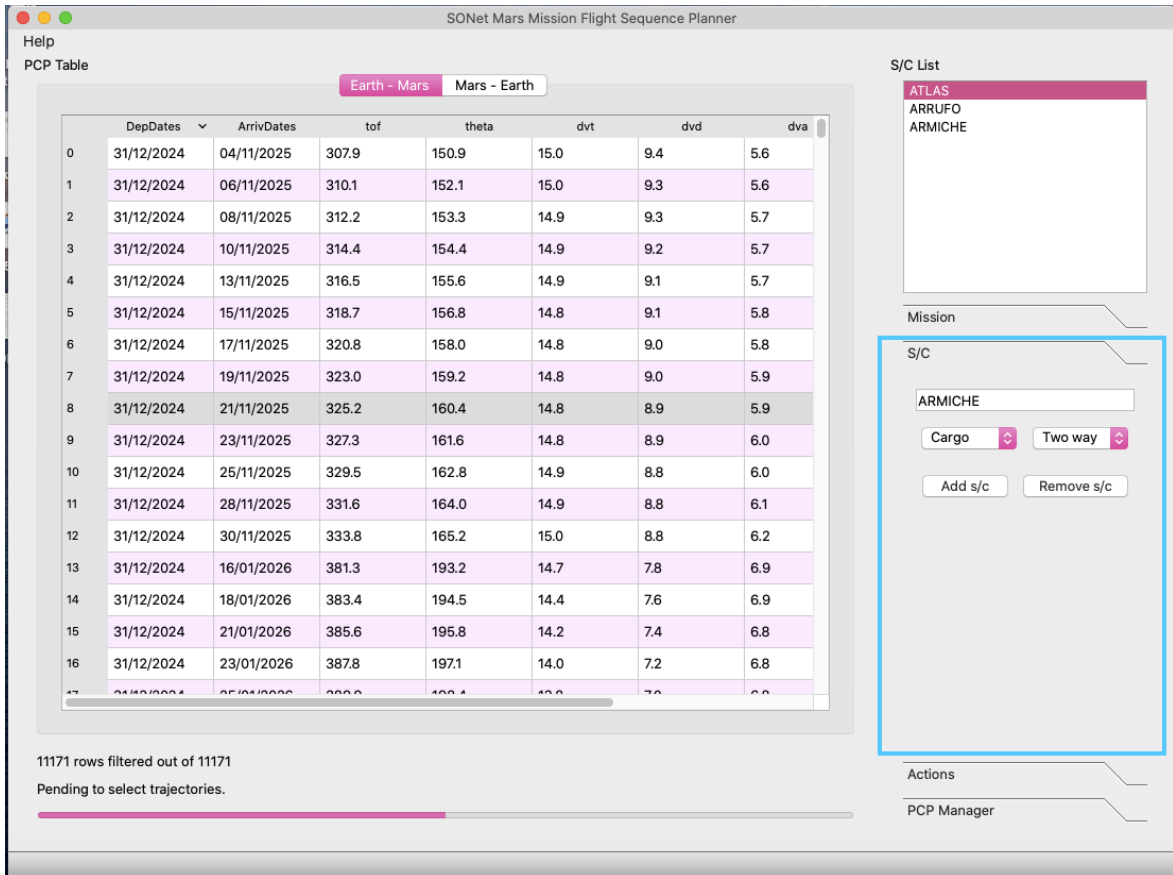


Figure 35. Main window S/C Tab.

You should understand the following about the s/c objects, which are the central objects of the app.

- S/C are objects which may be of type cargo/crewed, attending to the payload they carry on.
- S/C may also be one-way/two-way, depending on whether they have one (Earth-Mars) or two transits (Earth-Mars & Mars-Earth).
- Each s/c has a filter for each transit it has (one, or maximum two).
- This filter contains the conditions to apply to the trajectories database to get the filtered trajectories.
- If the user chooses a trajectory among the available ones, this trajectory is stored for the current selected s/c and transit.
- A s/c is fully defined when it has all its trajectories chosen.
- A mission is considered fully defined, when all the s/c are fully defined. The app job is done when the mission is fully defined.
- Unfortunately, you can't save the job done, this functionality is left for future works, see [Testing_section](#).

Actions Tab

This tab accommodates buttons for change the state of a given s/c.

- **Edit s/c filter** Executes the [Edit filters Window](#).

- **View s/c trajectories** Executes the [View PCP Window](#).
- **Select/Unselect trajectory** Chooses/Discards the current selected trajectory for the current selected transit, for the current selected s/c. If you try to choose a trajectory, and no trajectory is selected in the PCP Table, nothing happens. If you try to discard a trajectory, and the s/c has no trajectory chosen, a message is displayed in the status bar. If you try to choose/discard a trajectory, and no s/c is selected in the S/C List, a message is shown in the status bar.

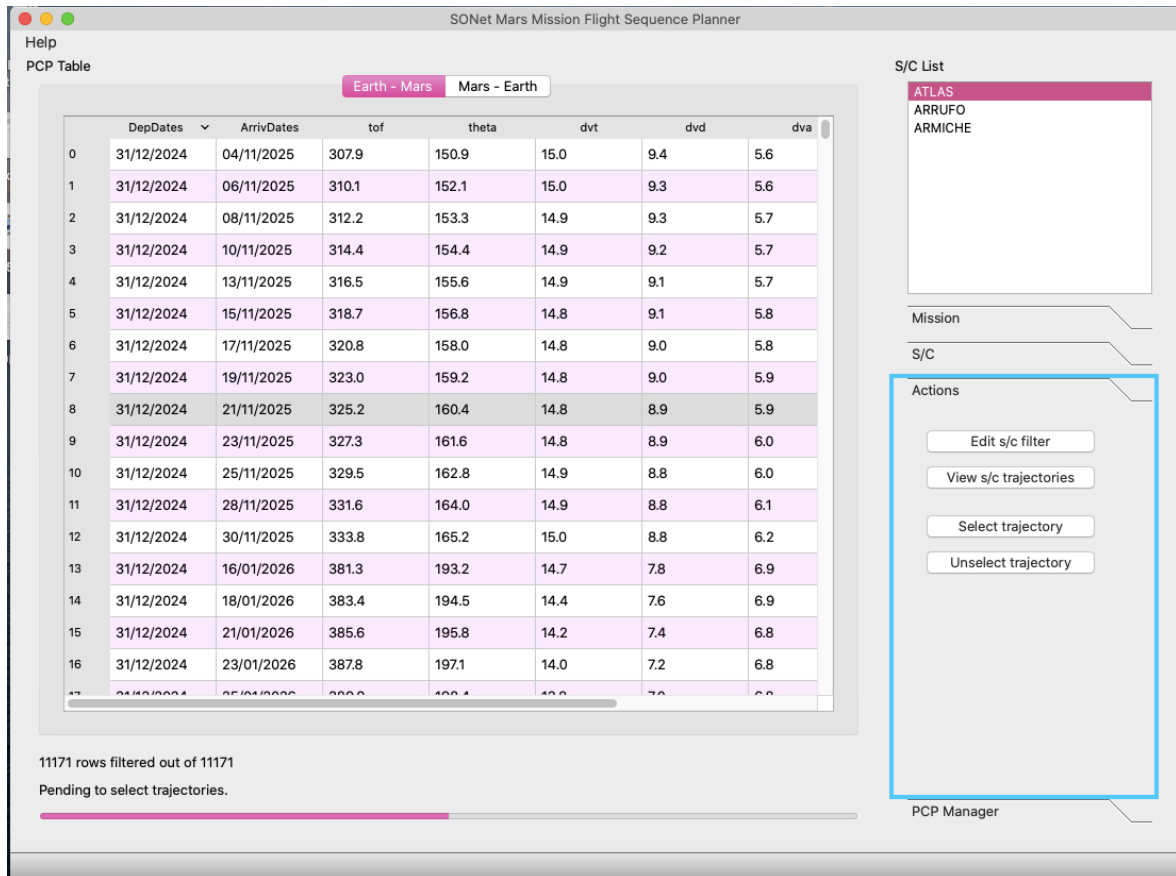


Figure 36. Main window Actions Tab.

PCP Manager Tab

As seen in [Figure 37](#), there is only one button, named 'PCP Manager'. This button executes the [PCP Manager Window](#).

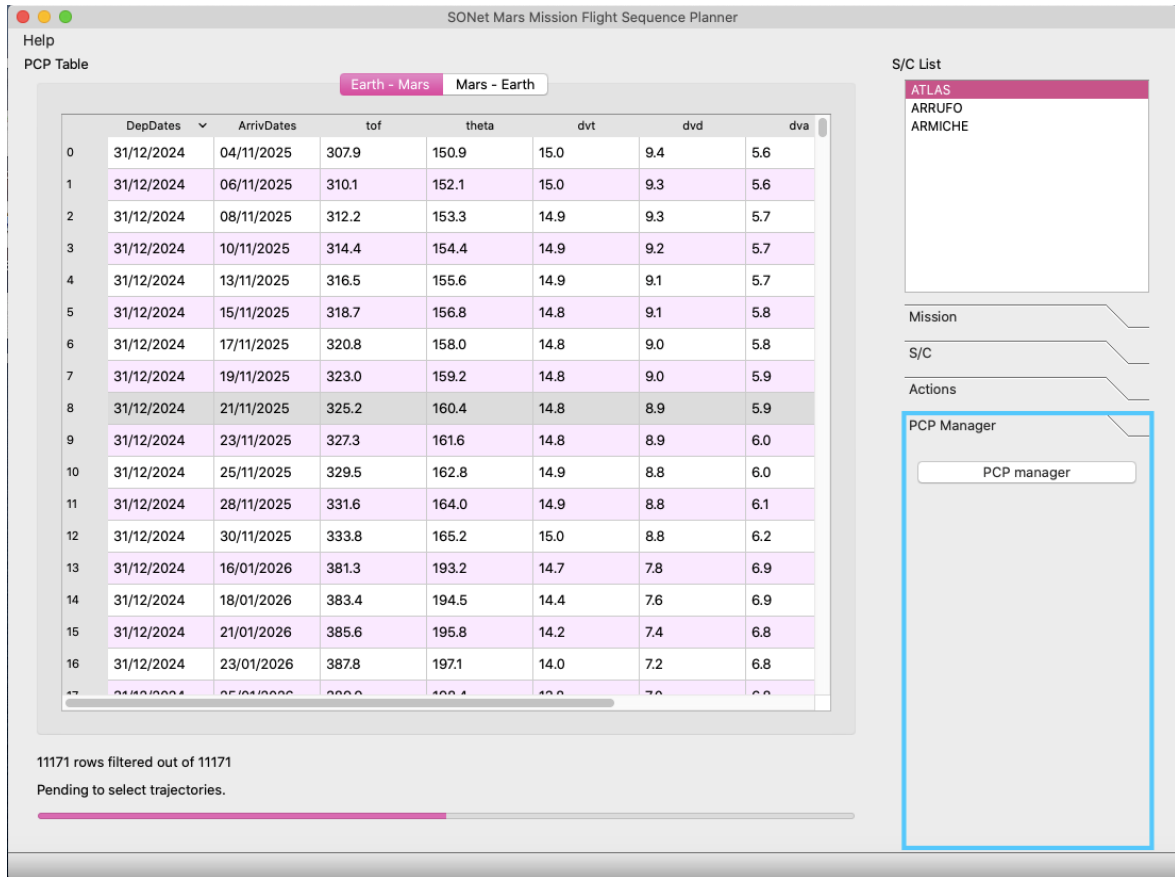


Figure 37. Main window PCP Manager Tab.

S/C Info

It's surrounded by the red rectangle in [Figure 31](#). It displays information of the current selected s/c. In the first row, you see a label with the count of the number of filtered trajectories. If you see something like '300 rows filtered out of 11171', it means that you are currently applying a filter, which only leaves visible 300 rows, out of 11171 which has the database for the selected transit. In the second and third rows, you will see a label and a progress bar, which tell you whether you have chosen all the trajectories for the selected s/c.

Status bar

It's surrounded by the cyan rectangle in [Figure 31](#). It eventually displays messages to the user, depending on which action she's trying to perform, giving additional and (hopefully) useful information.

Keyboard Shortcuts

To improve the user experience while working with the app, some buttons have associated a keyboard shortcut.

Table 5. Main window keyboard shortcuts.

Button	Keyboard shortcut
Add s/c	A, Enter, Return
Remove s/c	R, Backspace, Delete
Edit s/c filter	E
Select trajectory	U



Unselect trajectory

D

However, It's a bit tricky to make use of them, If you see that the don't work, try to left click over a generic area of the main window (e.g. the centre of nowhere), and then do the keyboard shortcut.

Canvas Window

This window's main purpose is to display an overview of a given s/c state. At any time, it will update with the information of the selected s/c within the [S/C List](#).

Figure 38 shows a general view of the canvas window. It can be divided into three differentiated sections:

- [S/C Info](#)(red) Display of s/c name, type, and its relations with other s/c.
- Trajectories Filter (orange) Display of the active filters of the s/c.
- Active Trips (blue) Display the chosen trajectories of the s/c.

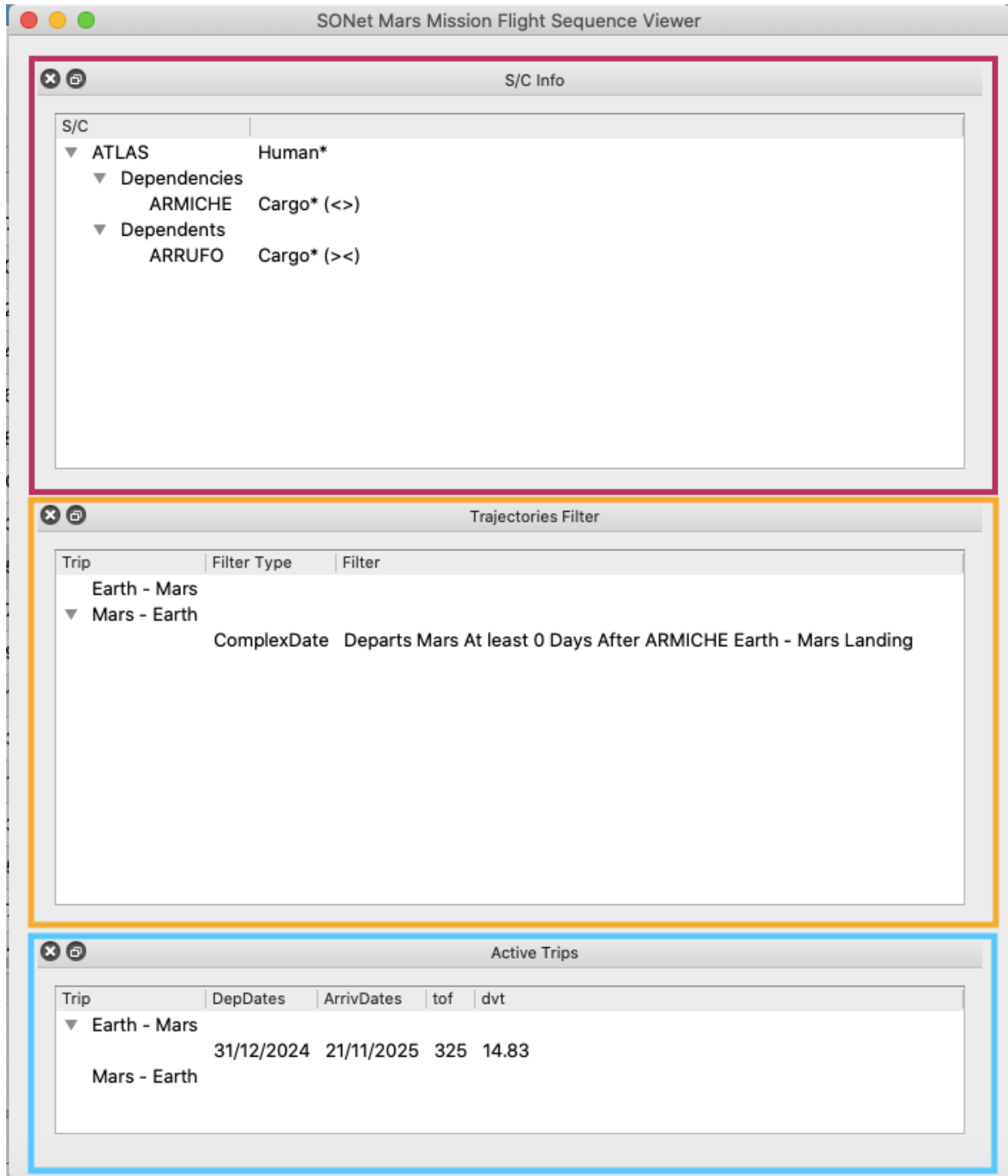


Figure 38. Canvas window general view.

S/C Info

It's surrounded by the red rectangle in Figure 38. The s/c info is arranged in a tree structure. The example from Figure 38 means that the selected s/c is named ATLAS, and it's a CREWED spacecraft. The star (*) next to the s/c name means that it has two transits, one Earth-Mars and another Mars-Earth. This is consistent with the other information seen in the canvas window or other windows.

- If you suspect at any moment that the information displayed isn't correct, click again on the desired s/c within the S/C List to force an update. If the information is still incorrect, report a bug in the GitHub project²⁰.

Following, there are two sections which display dependents/dependencies information for the selected s/c. The example from Figure 38 means:

- **Dependencies** ATLAS's Mars-Earth (represented <) transit depends on ARMICHE's Earth-Mars (represented >) transit. This dependency information is represented in an own compacted notation as (<>). This information is consistent with the one shown in other places (e.g. below, within the Trajectories Filter area).
- **Dependents** Same philosophy. ATLAS Earth-Mars (represented >) transit has a dependent, which is ARRUF0's Mars-Earth (represented <) transit.

Trajectories Filter

It's surrounded by the orange rectangle in Figure 38. The Trajectories Filter information is arranged in a tree structure. The example from Figure 38 means that the s/c ATLAS has two transits. The Earth-Mars transit has no active filters (it may have a filter, but it would be deactivated, see Edit filters Window). The Mars-Earth transit has an active filter, which tells you that ATLAS departs from Mars (this is, transit <) after s/c ARMICHE's Earth-Mars (>) landing. This is why in the above S/C info tree, the symbol (<>) is shown in the Dependencies section.

Active Trips

It's surrounded by the blue rectangle in Figure 38. The Active Trips information is arranged in a tree structure. The example from Figure 38 means that the s/c ATLAS has an Earth-Mars transit defined, but no a Mars-Earth one. To fully define the s/c, the user is pending to choose a Mars-Earth transit. In the example, the Earth-Mars transit of the ATLAS s/c departs Earth on December 2024, and lands on Mars on November 2025, resulting in a time of flight of 325 days, and a total orbit energy of dvt=14.83 km/s.

Edit filters Window

This window's main purpose is to display and edit the filter of a given s/c and transit. The window is modal, which mean you cannot manipulate other windows while this one is opened.

When clicking OK, the following happens:

- Each s/c and transit are analysed.
- If the filter has been modified within the window, the new filter is saved in the s/c object. If the s/c and transit had a trajectory chosen, this will be lost.
- If its filter has been unmodified, no action is performed.

When clicking Cancel, the window is closed, and nothing happens, which mean any modification is lost.

Clicking Reset will return all the window widgets to their default state, losing any user modification.

Clicking Add will add the current activated filters from the tab area to the filters table, if any.

Clicking Delete will delete the current selected filter in the filters table, if any.

Clicking Delete all will remove all the available filters in the filters table.

²⁰ https://github.com/horiaghionoiu/sonet_tfm_horia.

Figure 39 shows a general view of the filter editor window. It can be divided into three differentiated sections:

- **Select S/C and Trip** (red)
- Add a filter (green)
 - **Filter by Dates**
 - **Filter by Time of Flight**
 - **Filter by Energy**
 - **Auto Trajectory Selection**
- **Active Filters View** (blue)

The window workflow is from top to bottom. See **Select S/C and Trip**.

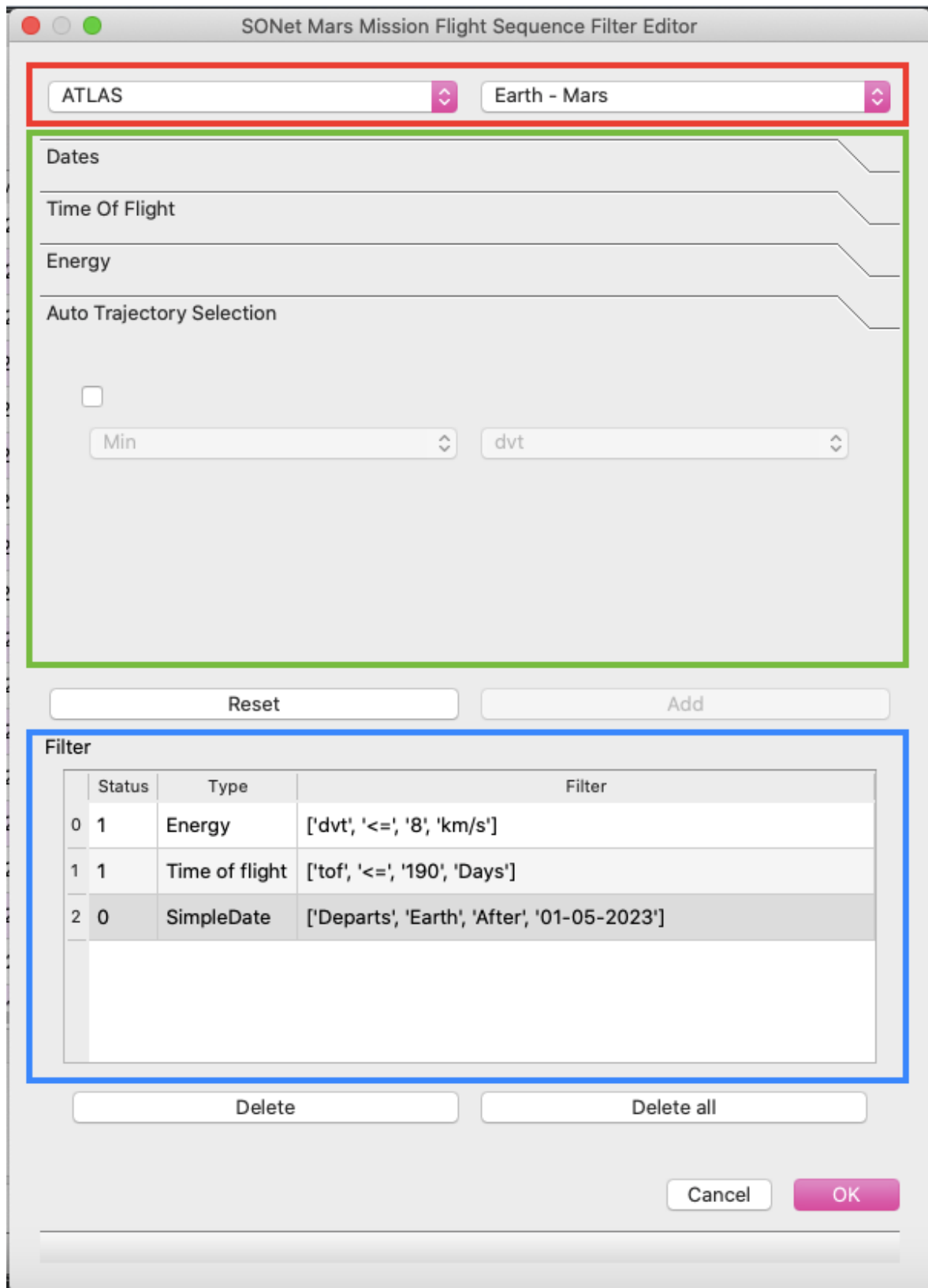


Figure 39. Edit filters window overview.

Select S/C and Trip

As seen in Figure 40, select a s/c from the top left combo, then, one of its available transits from the top right combo. Once this is done, the filter table for this s/c and transit is displayed at the bottom **Active Filters View**. At any time, you need to have a s/c and a transit selected, to be able to operate with the window.

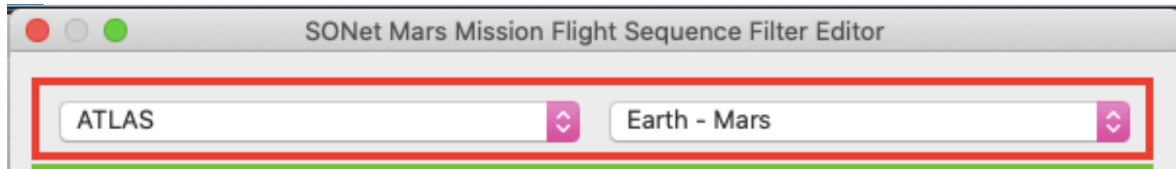


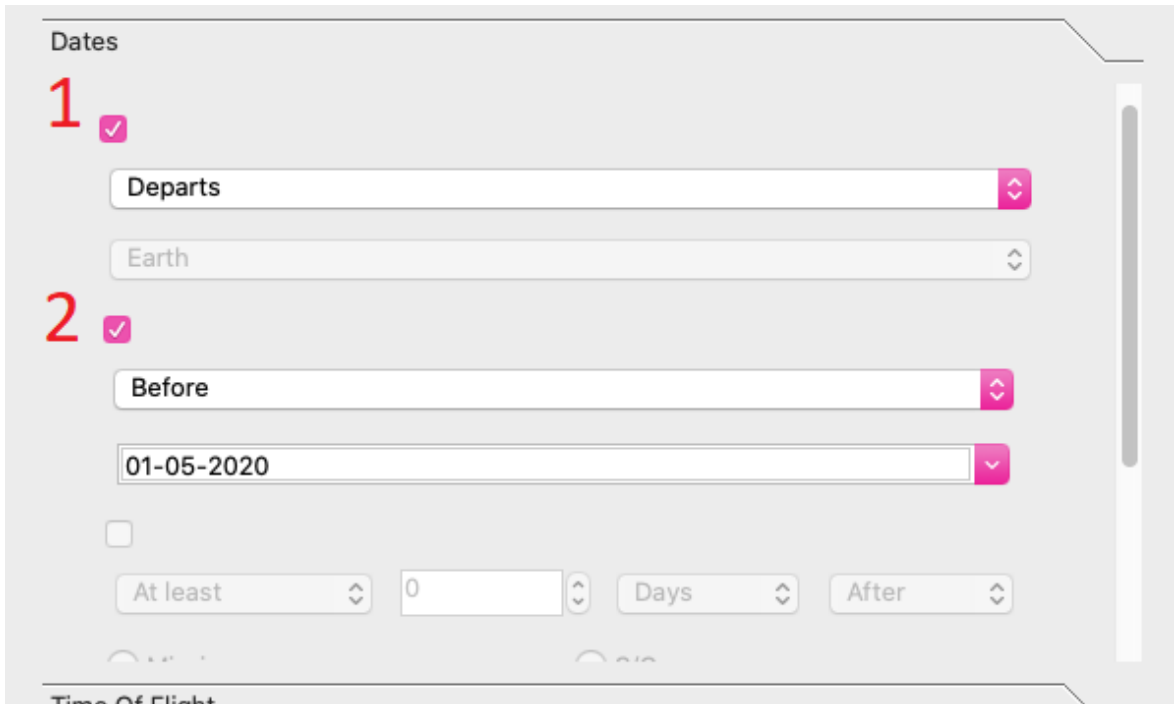
Figure 40. Edit filters window, select s/c and trip combos.

Filter by Dates

This section serves for adding filters related with the launch/landing dates of the s/c. There are two types of dates filter:

- **SimpleDate** Seen in Figure 41. To add a filter of the type 'S/C X departs/arrives Earth/Mars Before/After/On Y date.
- **ComplexDate** Seen in Figure 42. To add a filter of the type 'S/C X departs/arrives Earth/Mars At least/At most/At the same time S/C Y transit Earth-Mars/Mars-Earth launching/landing date. Adding a ComplexDate filter effectively establishes a relation between S/C X and S/C Y. This relation is a dependency for S/C X, and a dependent for S/C Y. For example, in Figure 42, the s/c Mars-Earth transit launching has been related with the same s/c Earth-Mars landing. This is an example of imposing that a s/c has to return from Mars only after it has arrived to Mars, which is logic but must be established manually.

- Check **Active Filters View** section to understand what happen when you relate a s/c with a second one, and this second s/c has no current trajectory chosen.



Dates

1

Departs

Earth

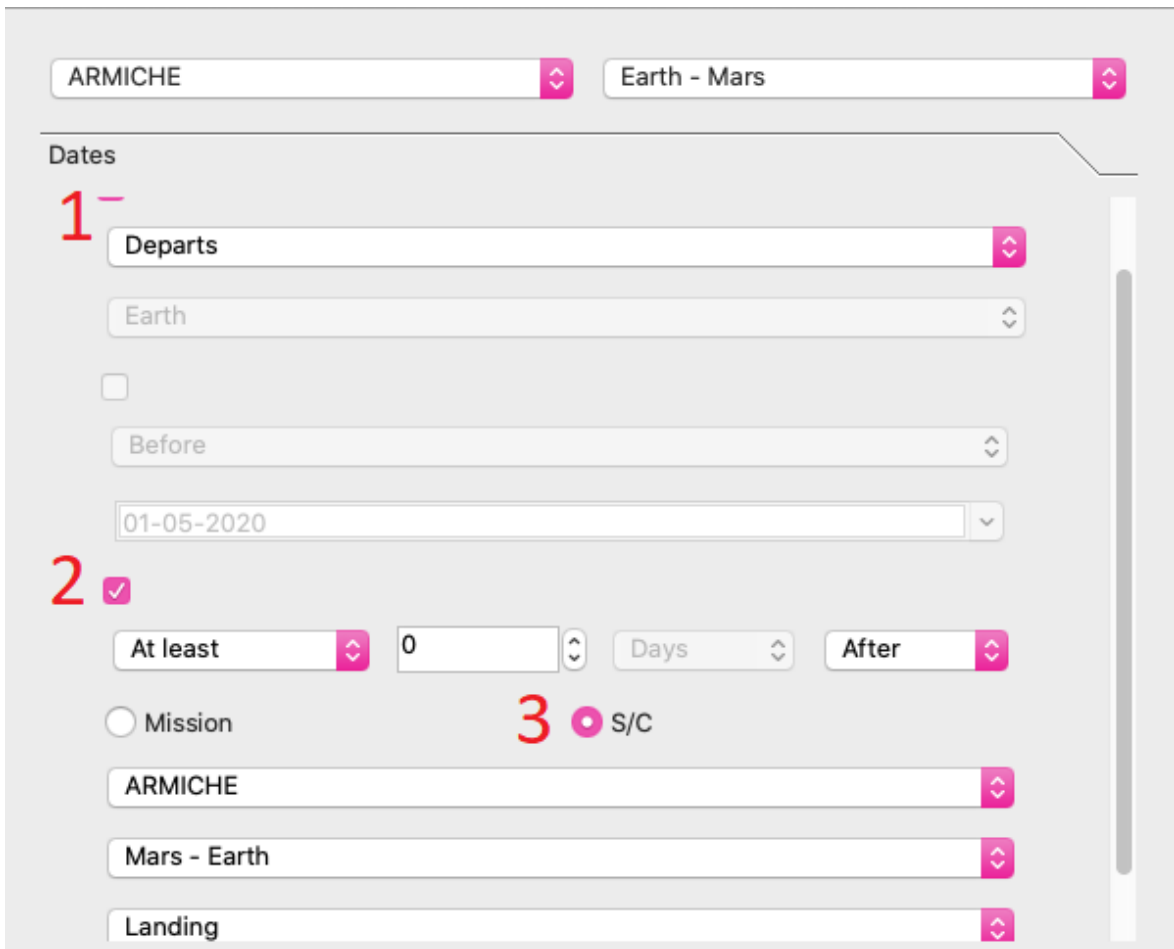
2

Before

01-05-2020

At least 0 Days After

Figure 41. Edit filters window, Date - SimpleDate filter.



ARMICHE Earth - Mars

Dates

1

Departs

Earth

Before

01-05-2020

2

At least 0 Days After

Mission S/C

3

ARMICHE

Mars - Earth

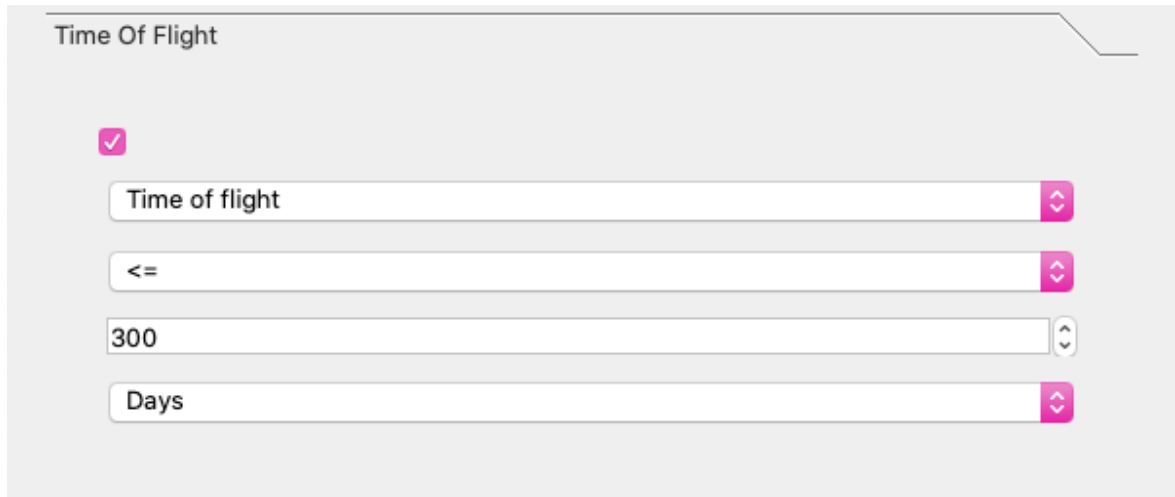
Landing

Figure 42. Edit filters window, Date - ComplexDate filter.

Note: The above figure is wrong, I should set the top right combo to Mars-Earth, and the top fourth combo from the top should write Mars, and the second combo from the bottom should write Earth-Mars.

Filter by Time of Flight

This section serves for adding filters related time of flight. Just set the maximum/minimum TOF for the s/c and transit, as seen in Figure 43.



The screenshot shows a window titled "Time Of Flight" with a checked checkbox. Below it are four input fields: a dropdown menu with "Time of flight", a dropdown menu with "<= ", a text input field with "300", and a dropdown menu with "Days".

Figure 43. Edit filters window, time of flight filter.

Filter by Energy

This section serves for adding filters related to what I called 'Energy' parameters. These are:

- **dvt** Delta Velocity Total. Total impulse (km/s) consumed in the trajectory.
- **dvd** Delta Velocity Departure. Departure impulse (km/s) consumed in the trajectory.
- **dva** Delta Velocity Arrival. Arrival impulse (km/s) consumed in the trajectory.
- **c3d** Departure dvd squared (km²/s²). It represents the kinetic energy consumed for the departure.
- **c3a** Arrival dva squared (km²/s²). It represents the kinetic energy consumed for the arrival.
- **Theta** The total trajectory angle, which tells you if it's a type I (<180°) or II (>180°).

Just set the maximum/minimum value for the chosen parameter, and for a given s/c and transit, as seen in Figure 43.

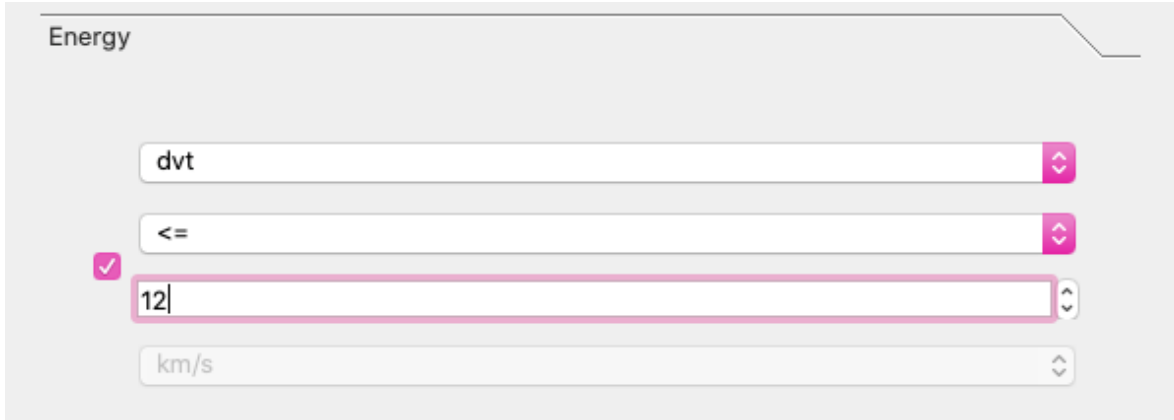


Figure 44. Edit filters window, energy filter.

Auto Trajectory Selection

This section serves for adding filters which will automatically choose a given trajectory, among the ones available. Just select Min/Max and a parameter, as seen in Figure 45. The available parameters are:

- Departure date, Arrival date.
- dvt, dvd, dva.
- c3d, c3a.
- theta.

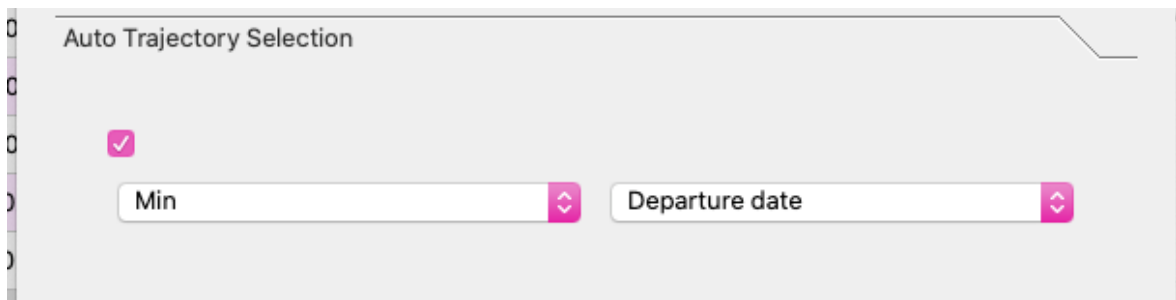


Figure 45. Edit filters window, auto trajectory selection filter.

Active Filters View

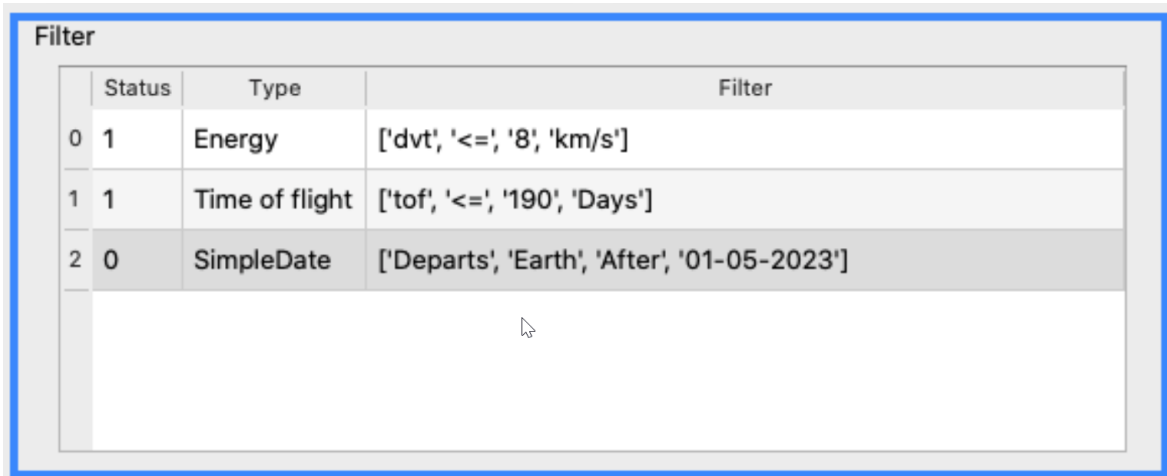
The above window elements described serve for manipulating the filter (which is a table as seen in Figure 46) of a given s/c and transit.

Observing Figure 46 (and recalling for Figure 39) one is supposed to add filters, which will appear activated in the Filter table below. You can change a filter status, between 1/0, to activate/deactivate it. When you are done, accept the window, or cancel it if you want to discard the changes.

It's worth to note that when one adds a ComplexDate filter and accepts the window, this filter will be tried to be applied. If it cannot be applied²¹ it will be automatically deactivated. Once you resolve

²¹ Example: 's/c X depends on s/c Y Earth-Mars transit, but s/c has no chosen Earth-Mars transit, so no relation can be established between s/c X and s/c Y.

the dependency, you can get back to the edit filters window, and activate the filter. **This is the reason why you can apply a filter and end up not seeing it when accepting the edit filters window.**



	Status	Type	Filter
0	1	Energy	['dvt', '<=', '8', 'km/s']
1	1	Time of flight	['tof', '<=', '190', 'Days']
2	0	SimpleDate	['Departs', 'Earth', 'After', '01-05-2023']

Figure 46. Edit filters window, filters table.

Keyboard Shortcuts

To improve the user experience while working with the app, some buttons have associated a keyboard shortcut.

Table 6. Edit filters window keyboard shortcuts.

Button	Keyboard shortcut
Add a filter	Enter, Return
Accept window	W
Cancel window	Q, Escape

However, It's a bit tricky to make use of them, If you see that the don't work, try to left click over a generic area of the window (e.g. the centre of nowhere), and then do the keyboard shortcut.

PCP Manager Window

This window's main purpose is to manage the trajectories database available for working with the app. The window is modal, which mean you cannot manipulate other windows while this one is opened.

By default, you have preconfigured trajectories for Earth-Mars and Mars-Earth transits, but you can change them manipulating the Working PCP Tab. Figure 47 shows a general view of the window. It features three tabs, which are:

- Generate PCP Tab.
- Convert PCP Tab.
- Working PCP Tab.

The intended use of the first two tabs is sporadic, as you will use them the generate PCP databases. Once you have generated enough PCP, you won't use them, instead, you will use the third tab, Working PCP Tab, to change between already generated PCPs.

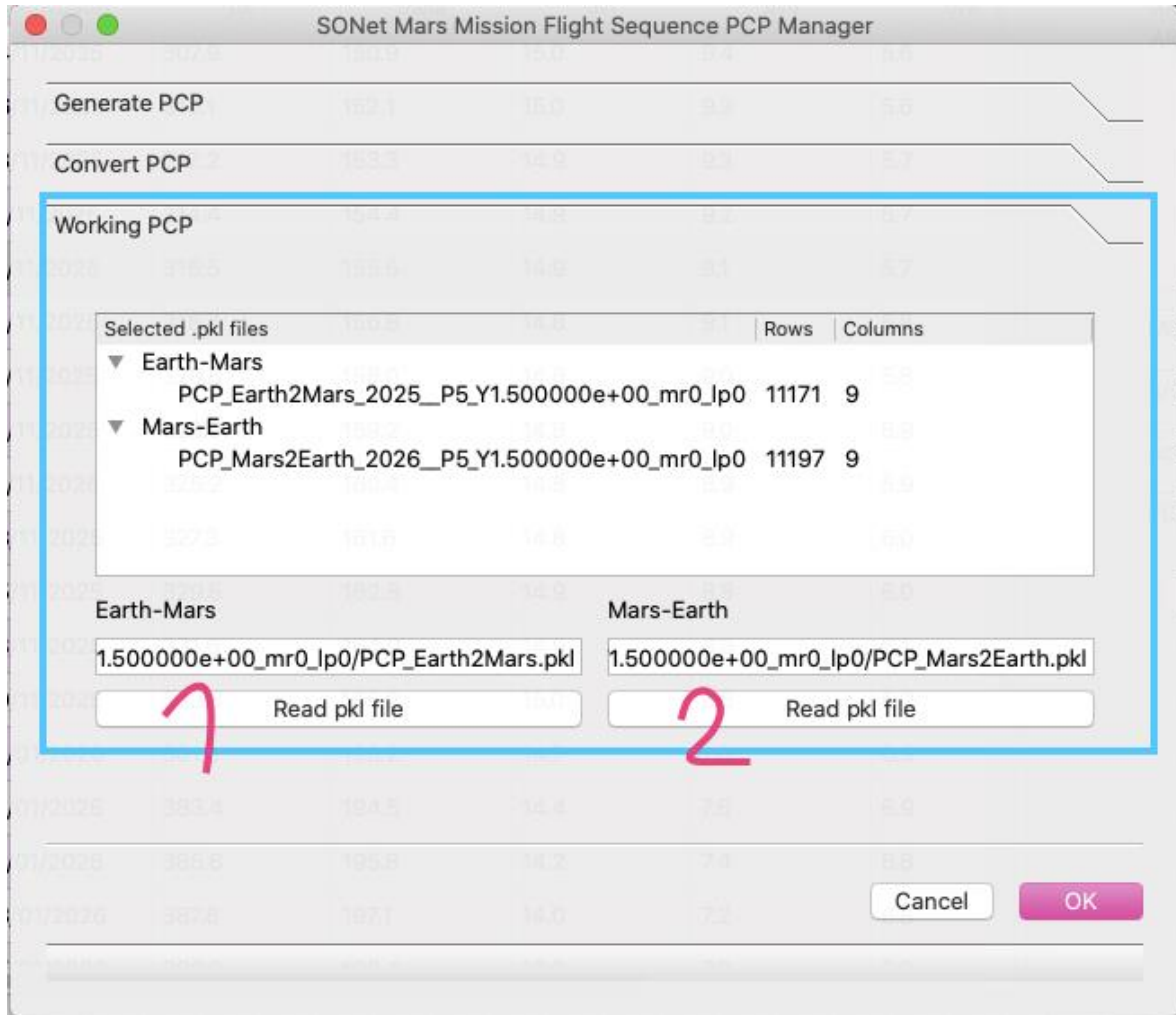


Figure 47. PCP manager window, overview.

Generate PCP Tab

This section, seen in Figure 48 contains the button 'MATLAB PCP generator', which effectively calls to a AstroLib's function in MATLAB, which generates the right hand side window. See Generate PCP Window section for more information.

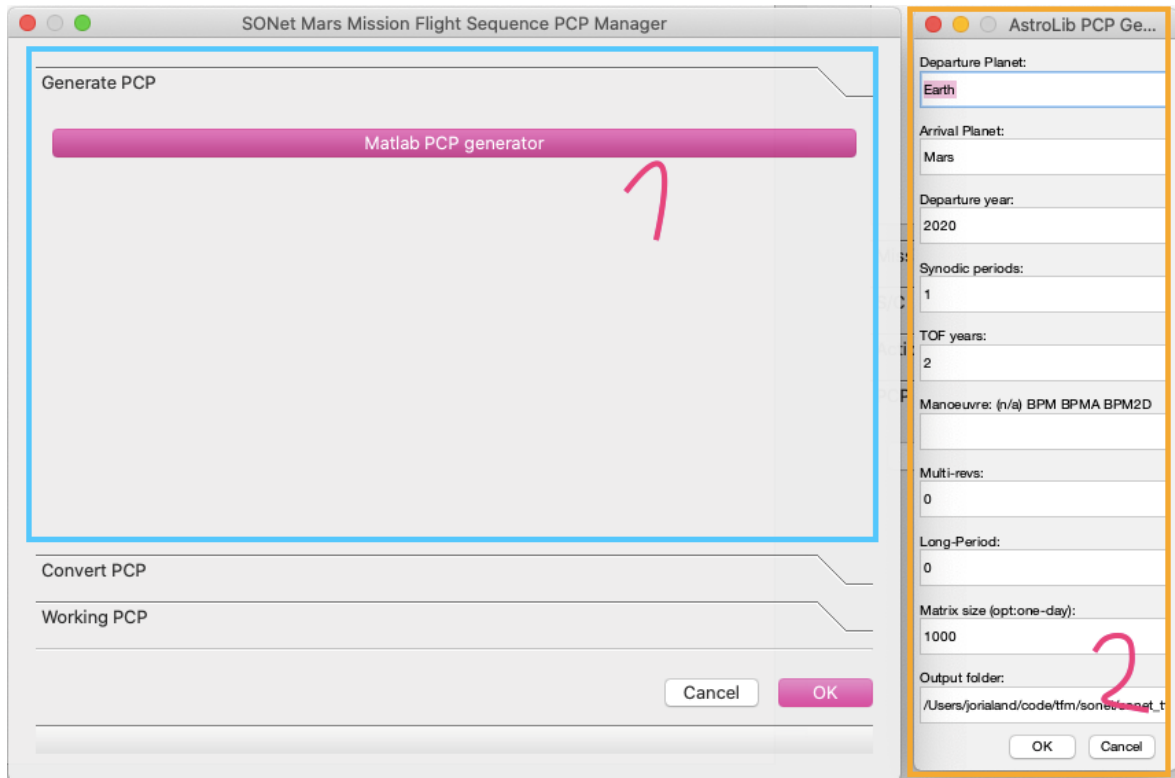


Figure 48. PCP manager window, generate PCP tab.

Convert PCP Tab

This section takes the .mat files generated in Generate PCP Tab section and converts them to Python Pickle (.pkl) files. What we are really doing is converting matrix like data (.mat) to tabular like data (.pkl), because it's easier to filter data when it's in tabular form. The app works with the .pkl files, which are set in Working PCP Tab.

As seen in Figure 49, you can select a .mat file (e.g. for the Earth-Mars transit), then set a limit dvt. This will discard all those trajectories above that dvt value, in this case 20 km/s. Finally click 'Convert PCP to table format' to generate the .pkl file.

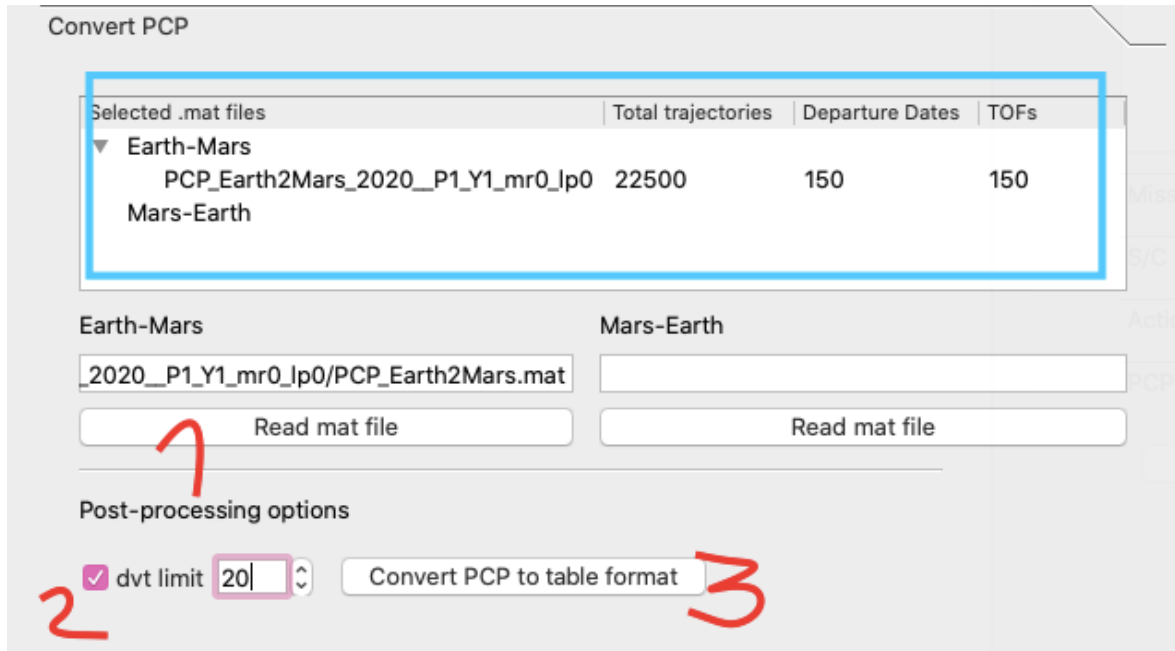


Figure 49. PCP manager window, convert PCP tab.

Working PCP Tab

Seen in Figure 47, left hand-side. You select the .pkl files to be used as the PCP data for the current app session. When changing the working PCP and accepting the window, the chosen trajectories for all the s/c will be lost, if any, as the PCP data they're based on has changed.

View PCP Window

Calls to AstroLib (MATLAB) for the current selected s/c and transit (e.g. ATLAS – Earth/Mars seen in Figure 36), and displays the PCP (e.g. the 11171 trajectories seen in Figure 36). See Figure 50.

You can close the window, which performs no action, or you can select a trajectory (left click in the graph on the left) from the PCP, which will be plotted at the right-hand side.

Once you select the first trajectory, if you close the window, the last selected one will be taken and chosen as the trajectory for the selected s/c and transit. This will effectively change the s/c state, and the associated widgets will update accordingly.

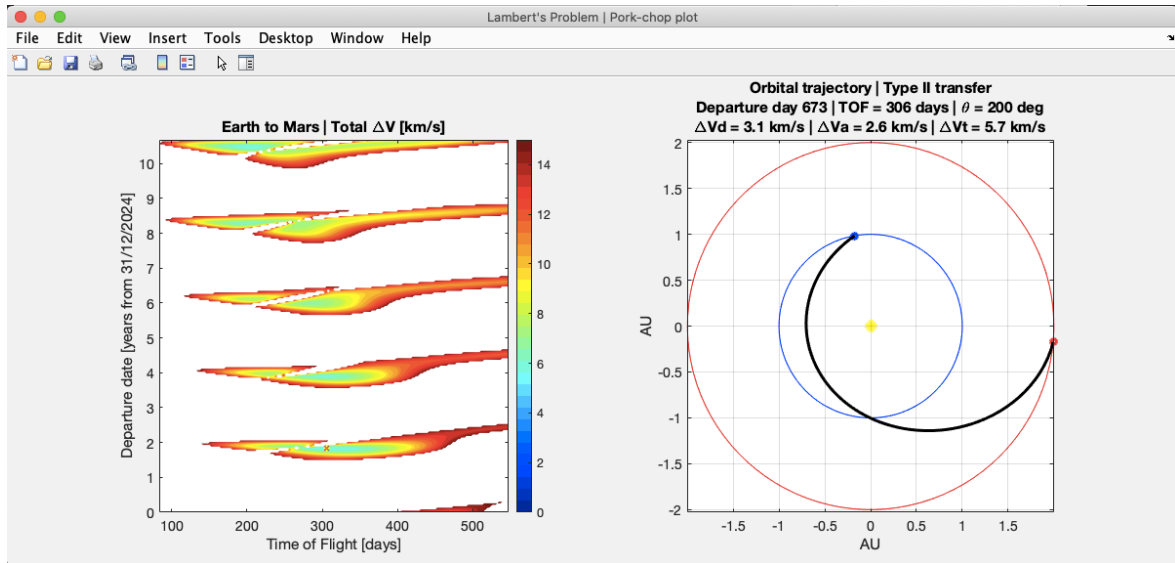


Figure 50. View PCP Window.

Generate PCP Window

This modal popup window can be seen in the right-hand side of Figure 48, it asks to the user for the following parameters:

- Departure Planet: Generally, Earth/Mars.
- Arrival Planet: Generally, Earth/Mars.
- Departure year: first year considered for generating the PCP.
- Synodic periods: the last departure year is the Departure year + Synodic periods.
- TOF years: maximum time of flight considered.
- Manoeuvre: left in blank, not used in the app.
- Multi-revs: left to zero, not used in the app.
- Long-Period: left to zero, not used in the app.
- Matrix size: **WARNING!** This parameter highly influences the resultant .mat file size. If too large, it may be difficult to operate with the app. It means the resultant PCP file matrix size, which is measured as the departure dates vs time of flight, 1000x1000 in the Figure 48 example.
- Output folder: recommended to leave the default one.

When clicking OK, it generates a binary .mat file, which contains the PCP data, according to the parameters set in the previous popup window. This .mat file can be read by Convert PCP Tab and be converted to Pickle (.pkl) Python files, which are finally used by the app to store the trajectories database.

Tutorials

Tutorials on how to use the application capabilities are being uploaded at the project site on https://github.com/horiaghionoiu/sonet_tfm_horia.

Annexes

Download Code From GitHub (zip file)

It's also possible to directly download the code in a compressed (.zip) file to your computer, and then add the project in PyCharm, as seen in next figures.

Using this option (instead of the control version one) you won't be able to update the application with new features developed and pushed to the GitHub repository.

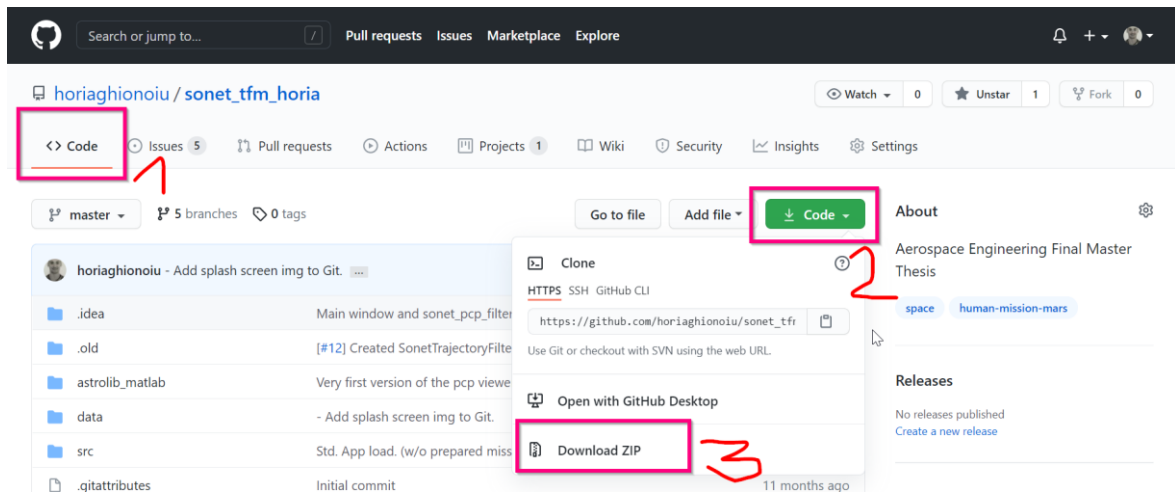


Figure 51. Get the Code directly into a zip file, from GitHub.

And then go to PyCharm and open it by selecting the folder where you extracted the zip file.

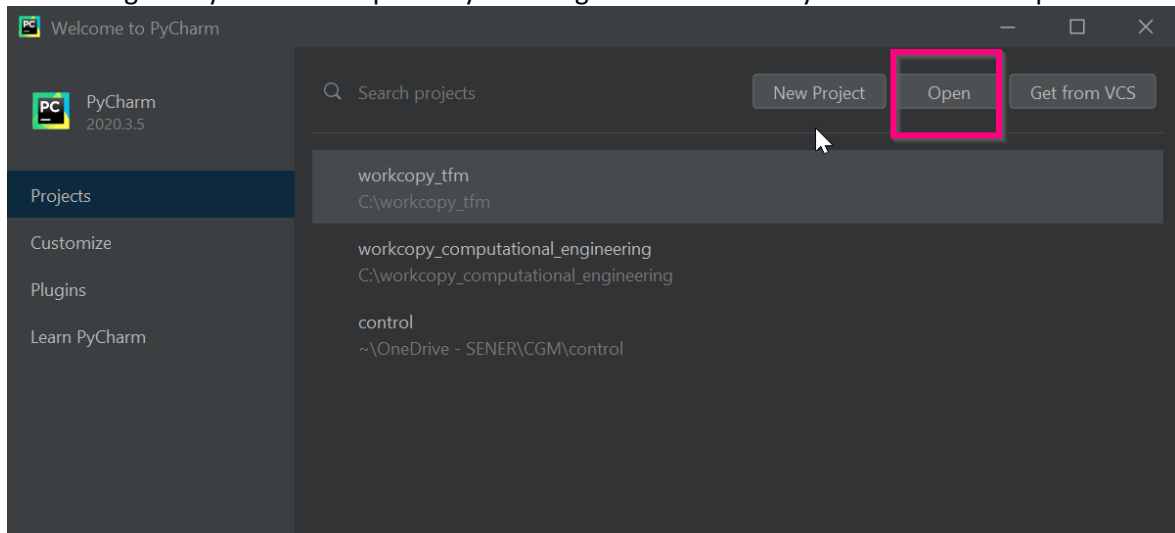


Figure 52. Create a PyCharm project from the downloaded code.