B

Boil-off tool guidelines

In this appendix, some important guidelines on the use of the Boil-off tool are given, together with a detailed description of its program files.
The Boil-off tool is open-source, and it is available upon request to the author or to the Delft University of Technology.

B.1. The Boil-off tool package

The Boil-off tool is written using the Cython 3.0.0a11 and Python 3.9.4 versions. In particular, the tool consists of several Python and Cython files that all contribute to building and analysing the propellant tank thermal model. The Boil-off tool package must contain the following files to properly work:

- *'Import_from_ESATAN.py'*
- *'BO_tool_main_body.pyx'*
- *'setup.py'*
- *'Boiloff_tool.py'*

The function of each of the Boil-off tool package files is explained in the following sections.
In addition, the following Python packages must also be installed:

- pandas
- tqdm
- numpy
- pathlib
- cython
- matplotlib
- CoolProp

B.2. Guidelines to run the program

In this section, the function of each file part of the Boil-off tool package is explained. Furthermore, instructions on how to run the code, the inputs to be given to the tool, and the outputs format are also given.
The files introduced in the following section are also in their execution order in the Boil-off tool.

B.2.1. Import thermal environment data in Python

File:'Import_from_ESATAN.py' Output: 'Radiative_model.npz'
This Pyhton program takes as input the "Radiative results" report file from ESATAN-TMS. The file extension is *.rpt, and needs to be converted in a *.txt format.
Some changes are needed before giving this file as input to the Boil-off tool: first of all, every data that does not directly concern the fluxes on the nodes need to be removed from the file: this includes the radiative model information at the beginning of the file, and fluxes information about total model and inactive faces at the end of the file.
The *'Import_from_ESATAN.py'* file can then be executed in Python, and it fills in a fluxes matrix according to the procedure already explained in subsection 3.1.2, and shown again here with Figure B.1 for completeness.

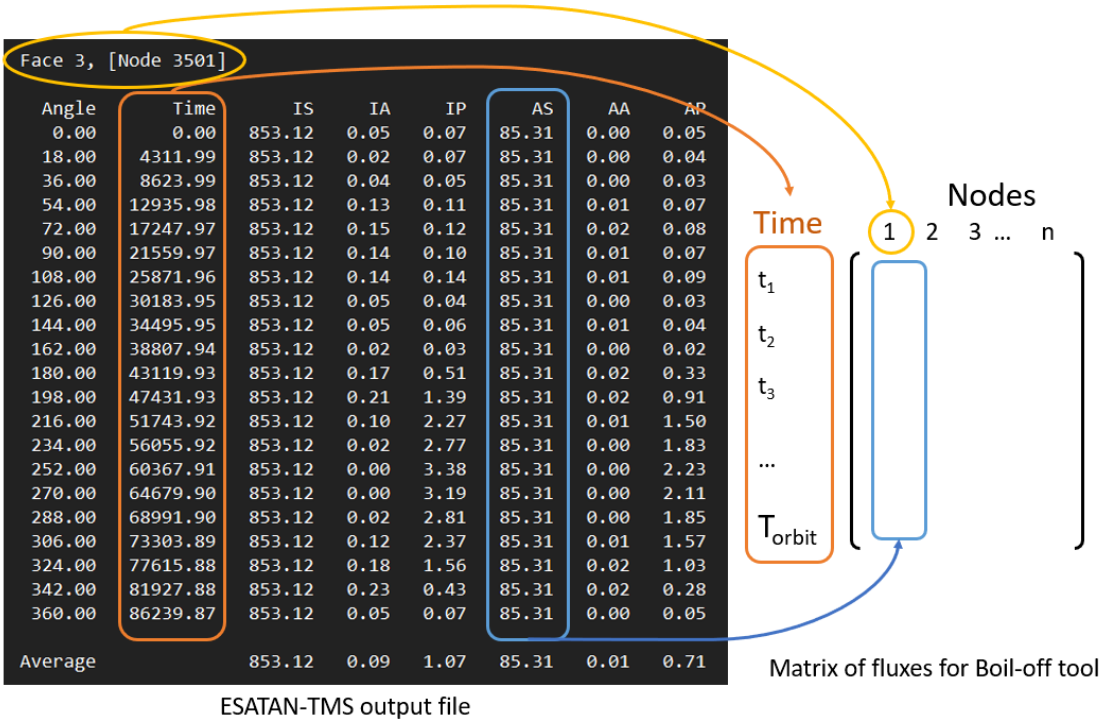

Figure B.1: Python-based ESATAN conversion: fluxes matrix generation. Optical properties used (MLI): $\alpha = 0.10, \varepsilon_{IR} = 0.66$.

In particular, all the heat fluxes will be saved in a total of two matrices: one for the right and the other for the left side of the tank, as shown in Figure B.2.

It has to be noted that the fluxes matrix generation process is dependant on the number of nodes chosen and on how the tank model has been generated in ESATAN-TMS. Therefore, it is up to the user to make sure that the fluxes are acquired correctly in the matrix.
The code developed here is a first draft made only to the scope of acquiring the thermal fluxes to continue with the analysis. Improving the flexibility of this code is indeed a recommendation for future work.

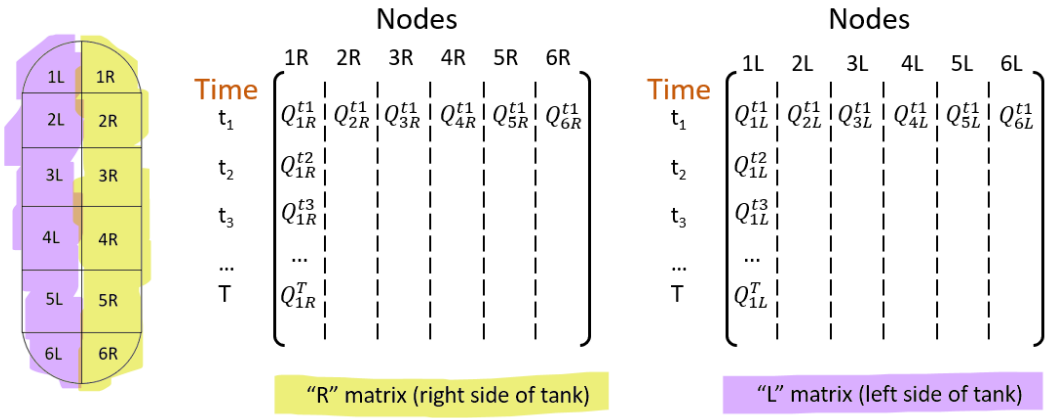

Figure B.2: Matrices of fluxes for right (R) and left (L) side of the tank, generated using ESATAN-TMS data.

B.2.2. *Cythonize* the Boil-off program

The Boil-off tool main body can be found in the **'BO_tool_main_body.pyx'** file, and it is written in Cython . There are little to no differences with the Python language (more details can be found in the Cython documentation [2]).
To execute the Boil-off tool using Cython, the code needs to be first compiled in C (this process is also called "Cythonizing" the code). More details about how to compile Cython programs can be found in [2], however one method would be to execute the following line in the terminal:

python setup.py build_ext –inplace

This requires an additional Python file called *'setup.py'* which is provided with the Boil-off tool package. A message notifies the user when the compilation is complete, generating two additional files with extensions *.c and *.so.

It is important to always make sure that the latest version of the *'BO_tool_main_body.pyx'* file is being used. If changes are made to the code, saving these changes does not mean that they are automatically compiled. Therefore, it is advised to always Cythonize the code before running it.

B.2.3. Run the Boil-off program

File: 'Boiloff_tool.py' Output: npz files and csv files

This is a Python file where all the Boil-off tool inputs are set, and it is executed in Python once the *'BO_tool_main_body.pyx'* has been compiled in C. The complete list of inputs of the boil-off tool can be found hereby:

- Simulation time step
- Storage duration (in months)
- Number of MLI layers
- MLI layer density
- Type of ESATAN-TMS fluxes: absorbed or direct
- VDMLI: ON or OFF
- VDMLI number of sectors
- VCS activity: ON or OFF
- VCS location: middle or inner VDMLI sector
- Insulation structure: MLI only, or MLI and WALL

- Pressurization: autogenous or heterogeneous
- Propellant mass
- Initial temperature and pressure
- Maximum tank pressure
- Tank radius (set this to 0 for spherical tank)
- Tank initial fill level
- Tank wall material (C_p, k, density, thickness)
- Reflector layer specific weight and nominal thickness
- Spacer layer specific weight and nominal thickness
- MLI thermal parameters (C_p, k)
- MLI inner and outer emissivity, and MLI solar absorptivity
- Gravity
- Space temperature ($\approx 3K$)
- Propellant mixture (LH2, LCH4 and LOX possible)
- Tank number of nodes
- Tank end-cap height (spherical or ellipsoidal shape)
- VCS tube diameter and length
- VCS shield material (density, C_p)
- VCS shield thickness
- Time-step for VCS analysis
- Desired pressure after venting
- Save data vectors over time: YES or NO

The Boil-off tool outputs are the following:

- Design options parameters, mass of the systems, and boil-off rate and mass are stored in *.csv files based on the selected mission duration
- Final temperature and heat loads are stored in *.npz files and can only be opened using Python

Once a single simulation is done, the outputs need to be stored on different directories before running the tool again: every new run, any existing data is automatically overwritten.