# AUTONAV

Autonomous Orbital Navigation


## Summary

1. Activity Overview
2. Topic addressed (orbital transfer)
3. Market size
4. HW components overview
5. Method and Constraints, I/Os
6. SW structure, Tools
7. Validation approach
8. Results
9. Lesson learned/Conclusion
10. Future work


# Activity Overview

- **Schedule, Cost: 1y (actual 14 months, 250 k)**
- **Purpose**

  **- Port AUTONAV Algorithm from Matlab to C**

  **- Validate the C version: precision on x86 and performance on LEON CPU**

  **- Get familiar with ESA standards**

- **Consortium**

  **1. *Thales Alenia Space in Italy*: Matlab Algorithm an Test Scenarios**

  **2. *Thales Systems Romania*: C porting, design, testing, validation (Test App and run scenarios)**


**Market Size – Orbit Transfer**

**Current Situation**

- Transfer Takes typically **from 6 to 12 months** (GEO).
- Many contacts with the ground segment are required for the attitude control and the orbital management.
- The ground segment support involves a significant increase in mission costs: for example, a 6 months' orbital transfer can cost about **1-2 M€**

By Using **AUTONAV** (both Geostationary and Constellations) i.e. S4I (Satellite for Italy) and G2G (Galileo Second Generation), it can be expected usage of about **30 completely electric platforms**.

- This development direction can provide a global improvement in competitiveness (50 M€ for the electric platforms)


## Orbital transfer challenge

***Problem:*** during the orbital transfer the **real trajectory** can be quite different with respect to the **optimal one** due to perturbations or PPS underperformances.

***Solution:***

1. Before launch the optimal transfer solution is calculated in the GS and it is stored in the satellites computer memory.
2. During the LEOP the satellite uses the stored database to compute the thrust strategy (thrusters switch on/off and firing direction for optimal maneuvers). During this phase the OBSW stores continuously the local position and velocity from the GPS sensor.
3. When required AUTNAV upgrades the database computing the new optimal transfer trajectory from the local measured position to the nominal target orbit.

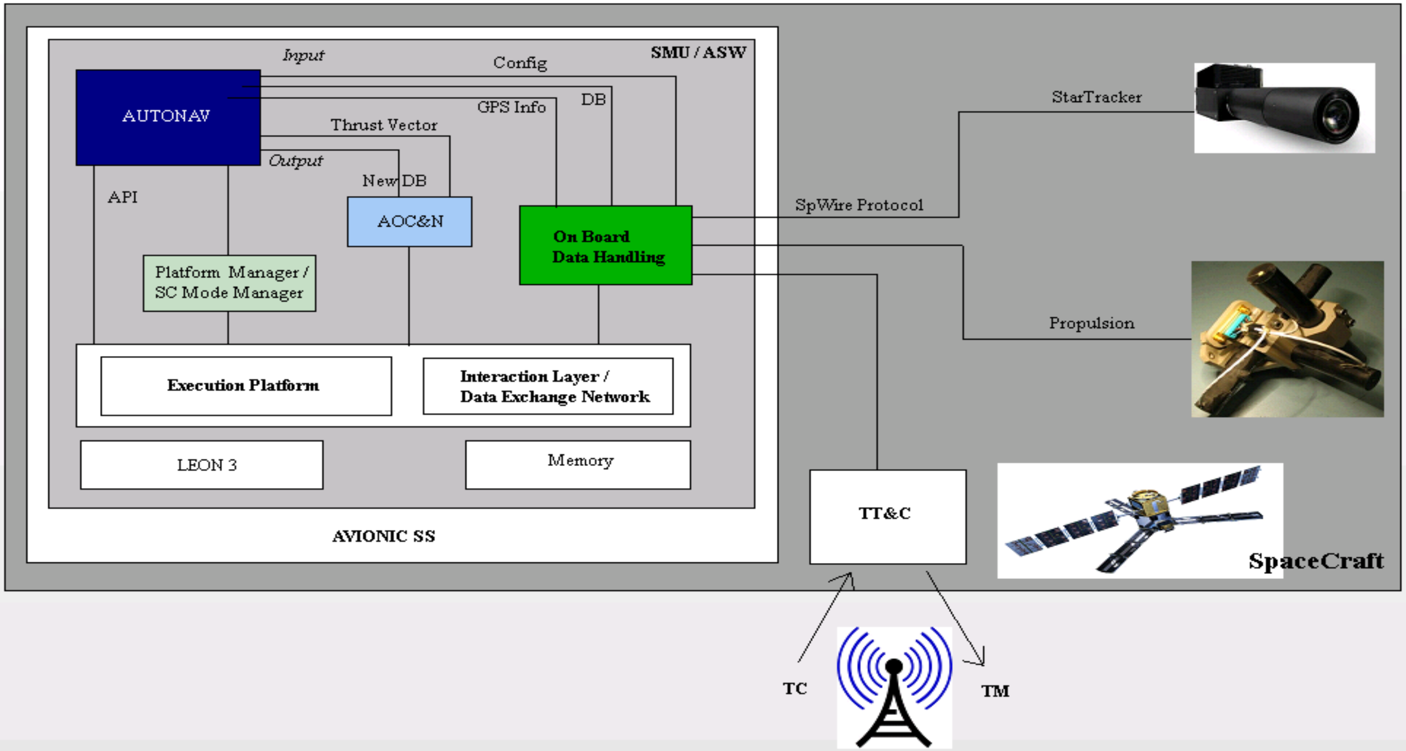

# Generic algorithm requirements

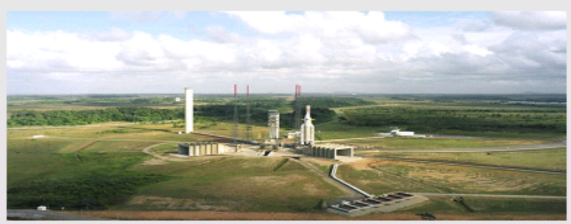
- The orbital transfer trajectory of an electrical satellite, from the **initial** orbit to the **final** one
- The **optimal thrust** strategy in order to minimize the orbital transfer **time**;
- The **optimal thrust** strategy computation for every orbital transfer (changing every orbital parameter: semi-major axis, eccentricity, inclination etc.)
- The **optimal thrust** strategy computation for **every satellite propulsion system** (changing thrust and specific impulse)
- The **optimal thrust strategy** computation for every satellite launch **mass**
- That the **optimal thrust strategy** takes into account the **perturbation** effects (as J2) and **eclipse** phases
- The **optimal thrust strategy** takes into account the **eclipse effects** switching off the thrusters during the eclipse phases
- The **optimal thrust strategy** takes into account **unpredictable** propulsion system underperformances


# Generic SW component requirements

- Functional Requirements (**Orbital transfer computation, propulsion, mass**, etc.)
- Performance Requirements (**Convergence time, Perturbations**)
- External Interfaces (**Use own interfaces**, **Cycle entry point, Component ITFs , Handle Exceptions, etc.**)
- Resources Requirements (**RAM, ROM, CPU**)
- Design requirements and implementation constraints (**ECSS, Criticality, libraries**)
- Portability requirements
- Software Quality requirements (**PA**)
- Software maintainability requirements
- Software Reliability requirements (**Defensive Impl, Err detection/handling** )
- Software Safety requirements

# Autonomous Orbital Navigation (AUTONAV) On Board Software




# Work Performed - TSR

## Development

1. **Build AUTONAV Architecture**
2. **Detailed design (Interfaces, Behavior)**
3. **C code Implementation**
   - **With aid of code generation (in Rhapsody) Started on Matlab version**

## Test/Validation

**Built 2 Test Applications:**

- **For precision on x86 (Linux)**
- **For performance on LEON with TSIM**


# Satellite main components

***Propulsion Boards***

The Propulsion I/O board interfaces the following Spacecraft equipment
7+7 Reaction Control Thrusters + Heaters

***Attitude determination/Control***

Magnetometers
Fine Sun Sensors
Star Tracker

***Attitude Control***

Magnetotorquers
Reaction Wheels
Reaction Wheels (RW)

***Orbit determination***

GPS


# Approach – Mathematical Method

- TAS-I has developed internally the optimization software (SOFTT), based on **indirect** solution techniques

- Pre-existing Research Activity: more than 1000 different scenarios have been studied

- **Maximum Principle** applied where the Hamiltonian is defined along the transfer

SOFTT: Space Optimal Finite Thrust Transfer


# Approach - Constraints

- Convergence criteria as main metric to compute and assess the **optimal thrust strategy**
- Discretization of more solutions around the optimal state and selection of the optimal co-state
- Convergence time

Co-state: (Lagrange multipliers as time-dependent variables)


# Approach Advantages

- Averaging techniuques provides **fast and converging** computing methods for long orbital transfer (computational time compatible with the available CPU)
- **Bigger payload** mass up to 1/5 of the satellite launched mass
- **Optimal trajectory** within minimum transfer time and minimum consumption of propellant
- **Orbital thrust strategy** computation for every orbital transfer, for every satellite launch mass and propultion system

# Algorithm inputs (1/2)

1. Database
   - X – states
   - L – co-states
   - w – particular anomaly for each state

2. Configuration parameters
   - Related to the satellite structure
   - Related to the initial orbit and the final orbit
   - Related to the earth and sun characteristics
   - Related to the boundary conditions
   - Related to the perturbation


# Algorithm inputs (2/2)

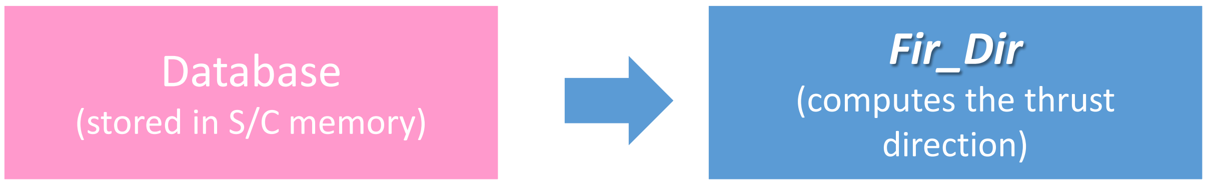
3. Orbital data – provides information on the current orbital status of the satellite
    - Epoch as Julian Date (current time)
    - Position vector components
    - Velocity vector components
    - Actual Mass
    - Progressive anomaly at Epoch (current anomaly )


# Algorithm outputs

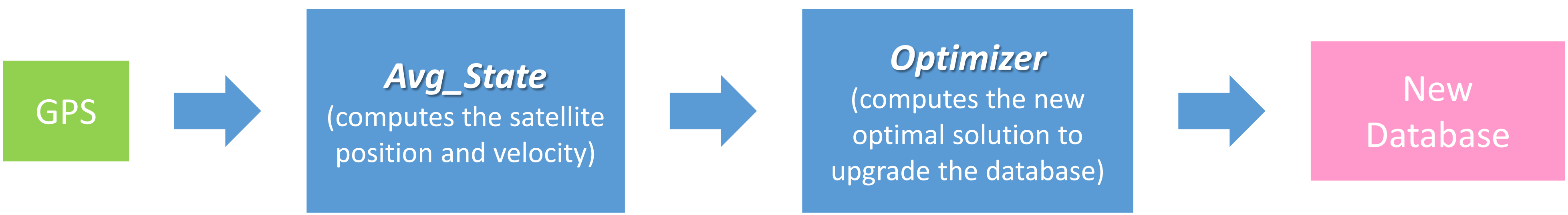
- Database – if the algorithm reached convergency it will update the database
- Firing Direction
    - Anomaly to targeted orbit
    - Thrust unit vector
    - Switching function / control force


***Interfaces:*** the system interface is mainly composed by three subroutines:

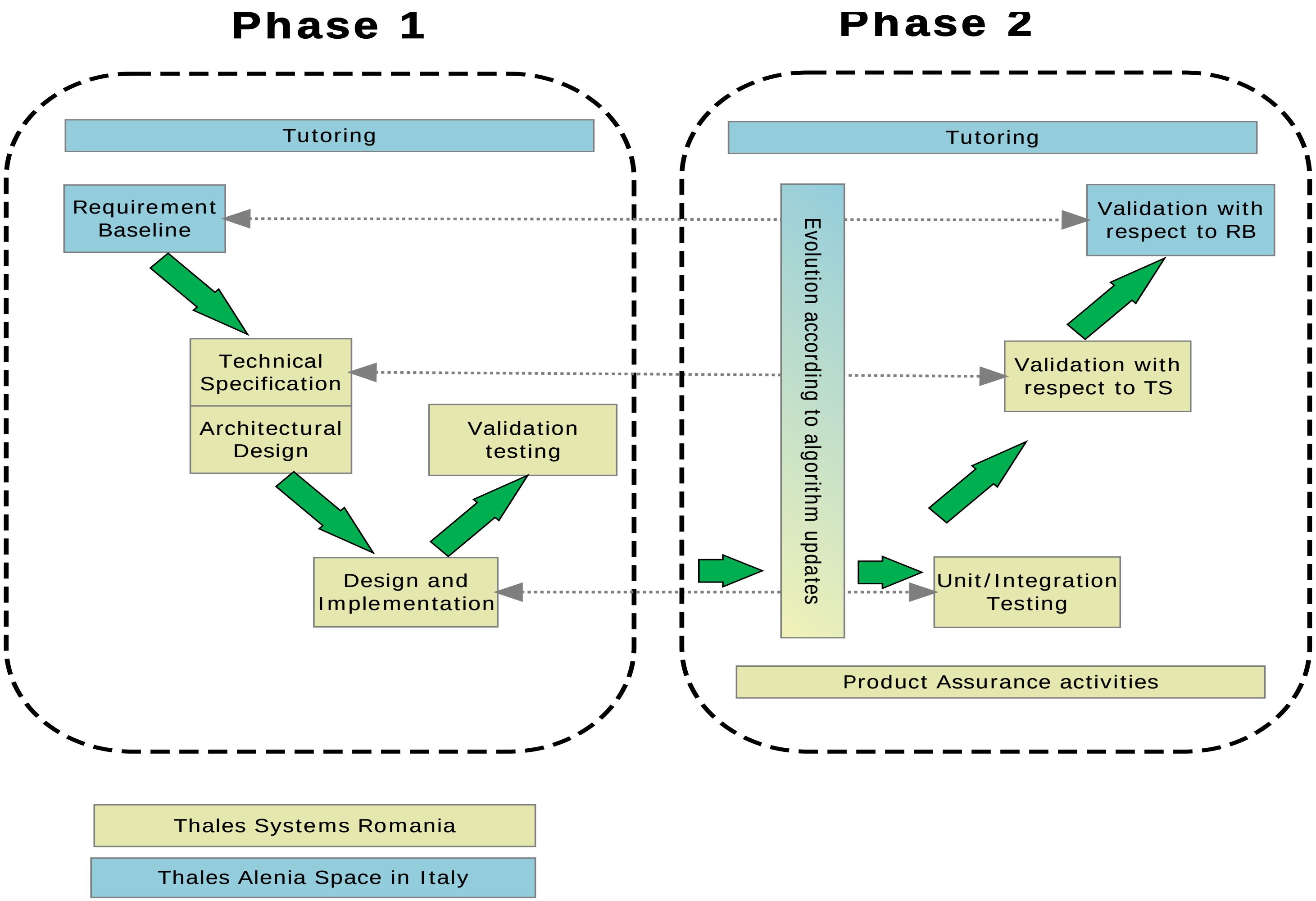
1. ***Fir_Dir***: calculates the thrust direction and the switching function to implement the optimal maneuvers;




2. ***Avg_State***: calculates the average state starting from the measured position on the current time.



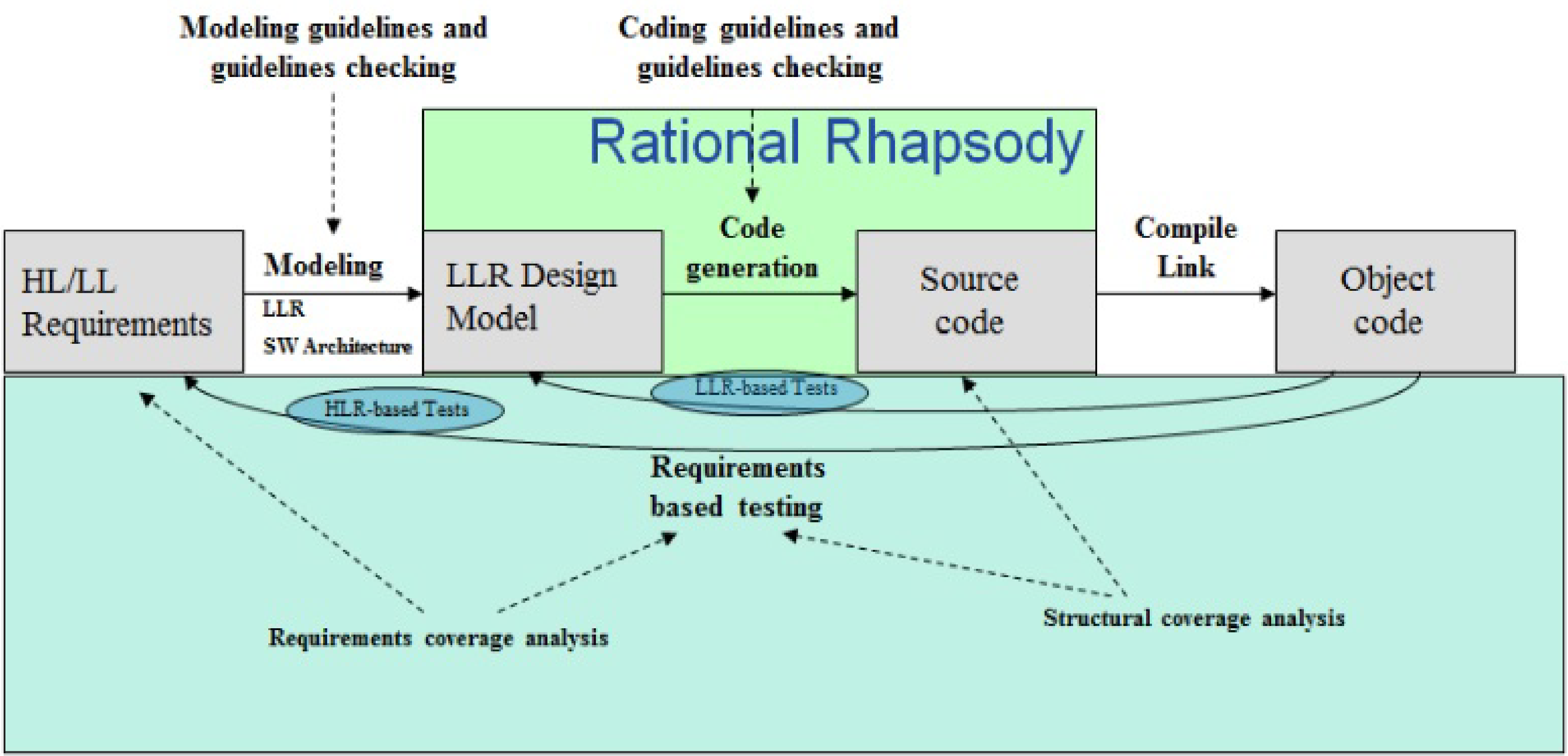

## Development cycle



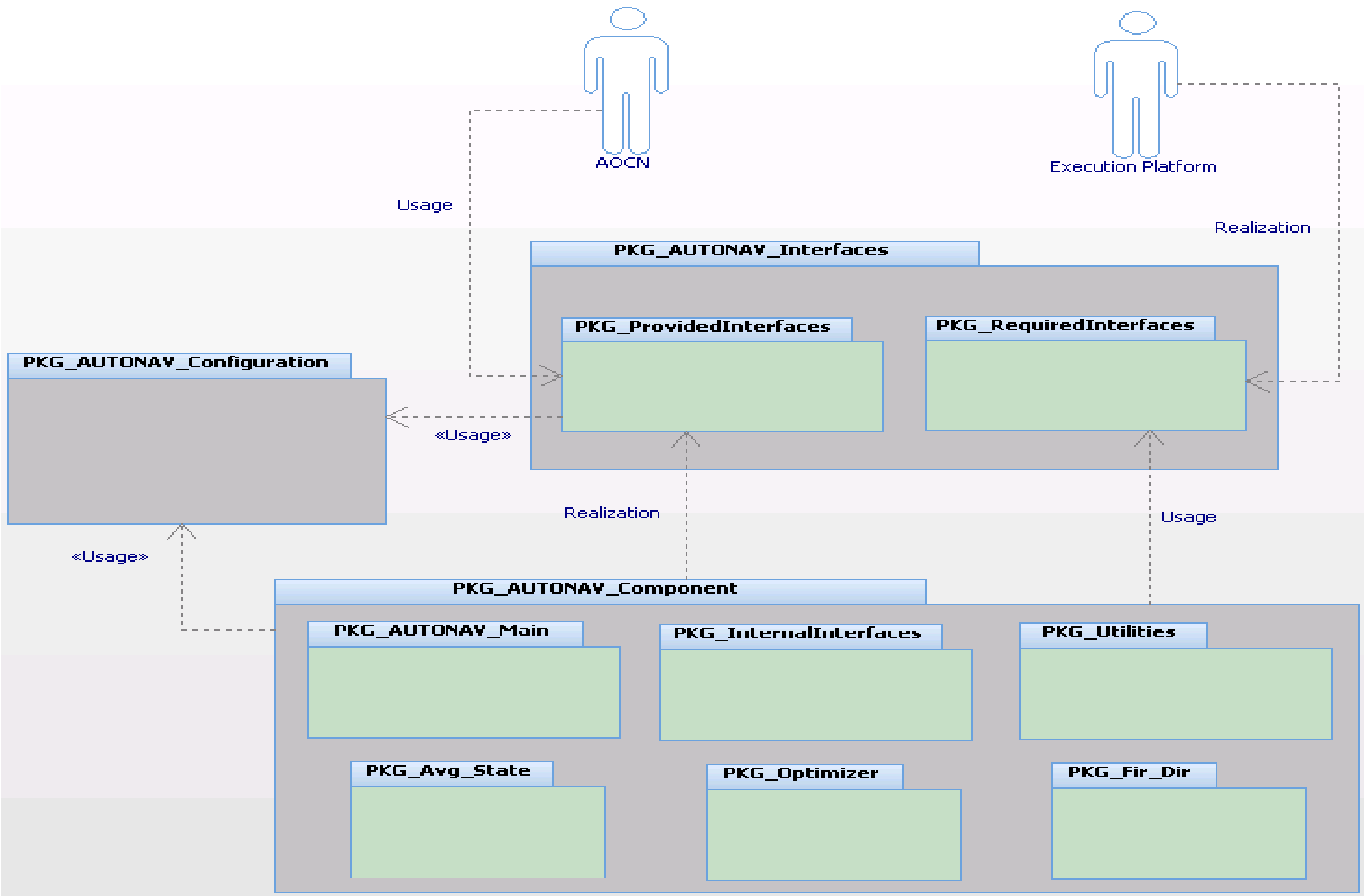

# Used Tools

- ❑ **Rhapsody**
  - ➢ **Modeling and design, traceability of requirements**
  - ➢ **Generating code and documents**
  - ➢ **Software engineering (with UML and SysML)**
  - ➢ **Test applications**
- ❑ **TSIM**
  - ➢ **Emulating LEON-based computer systems (LEON3 and LEON4)**
- ❑ **Matlab**
  - ➢ **Run the simulations, generate benchmark**


## Development with Rhapsody



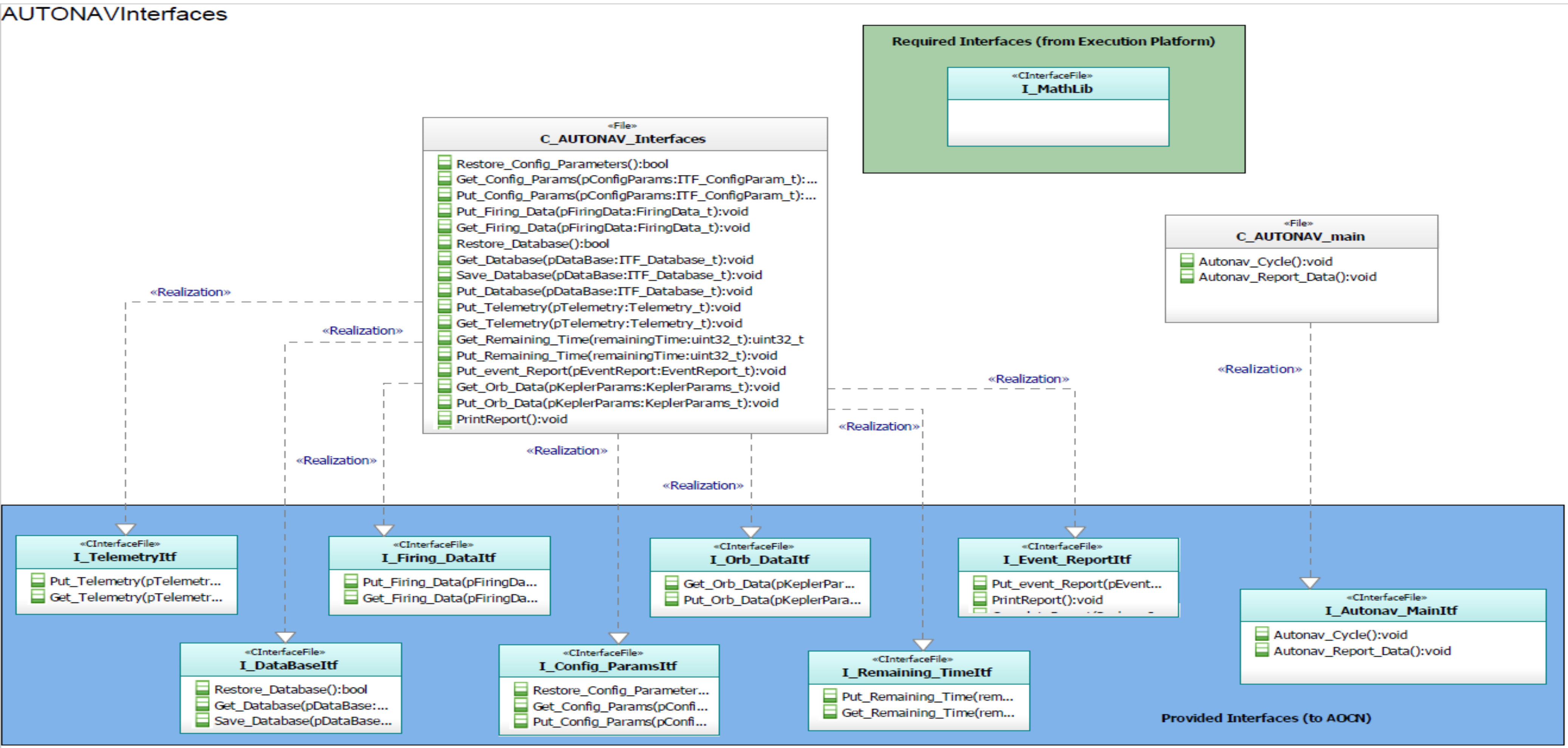

ECSS-E-ST-40C

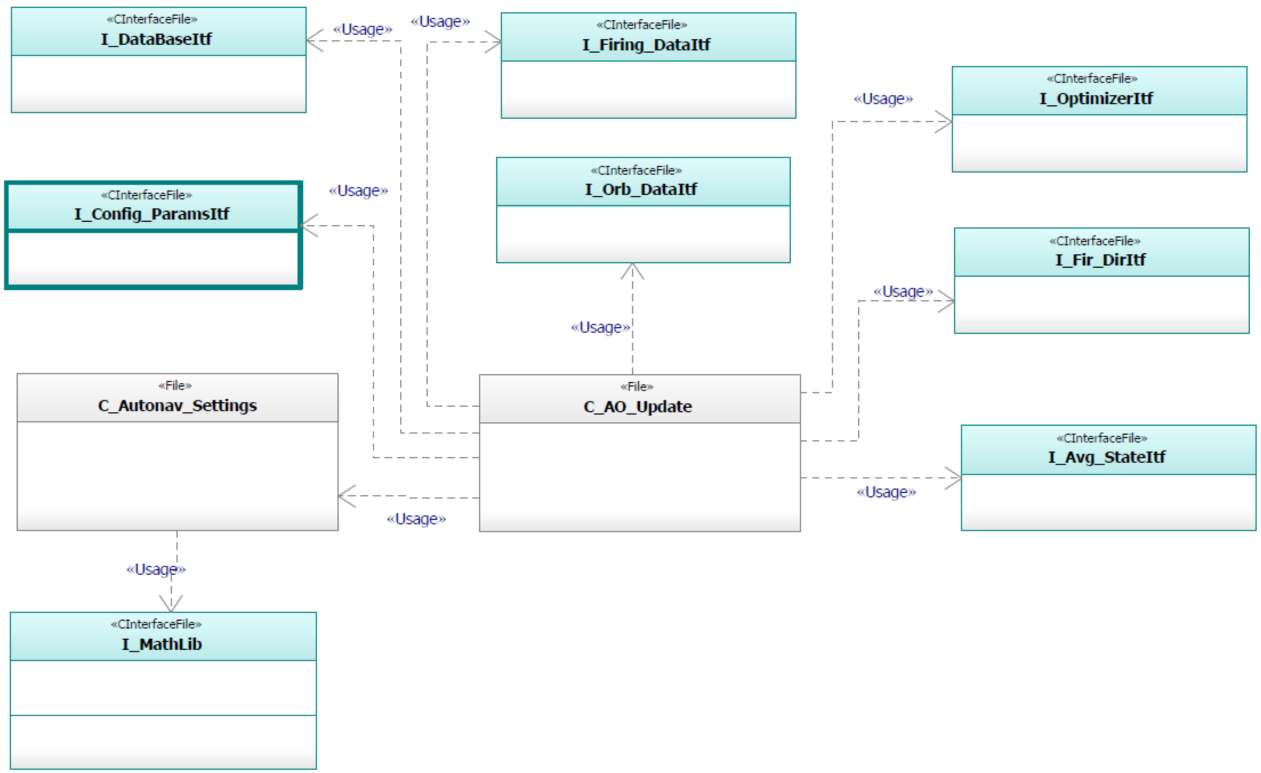

**Provided**:
Implemented by AUTONAV SW module
Explicitly used by host SW

**Required**:
Requested by AUTONAV SW to compile


# SW Modules Used Interfaces

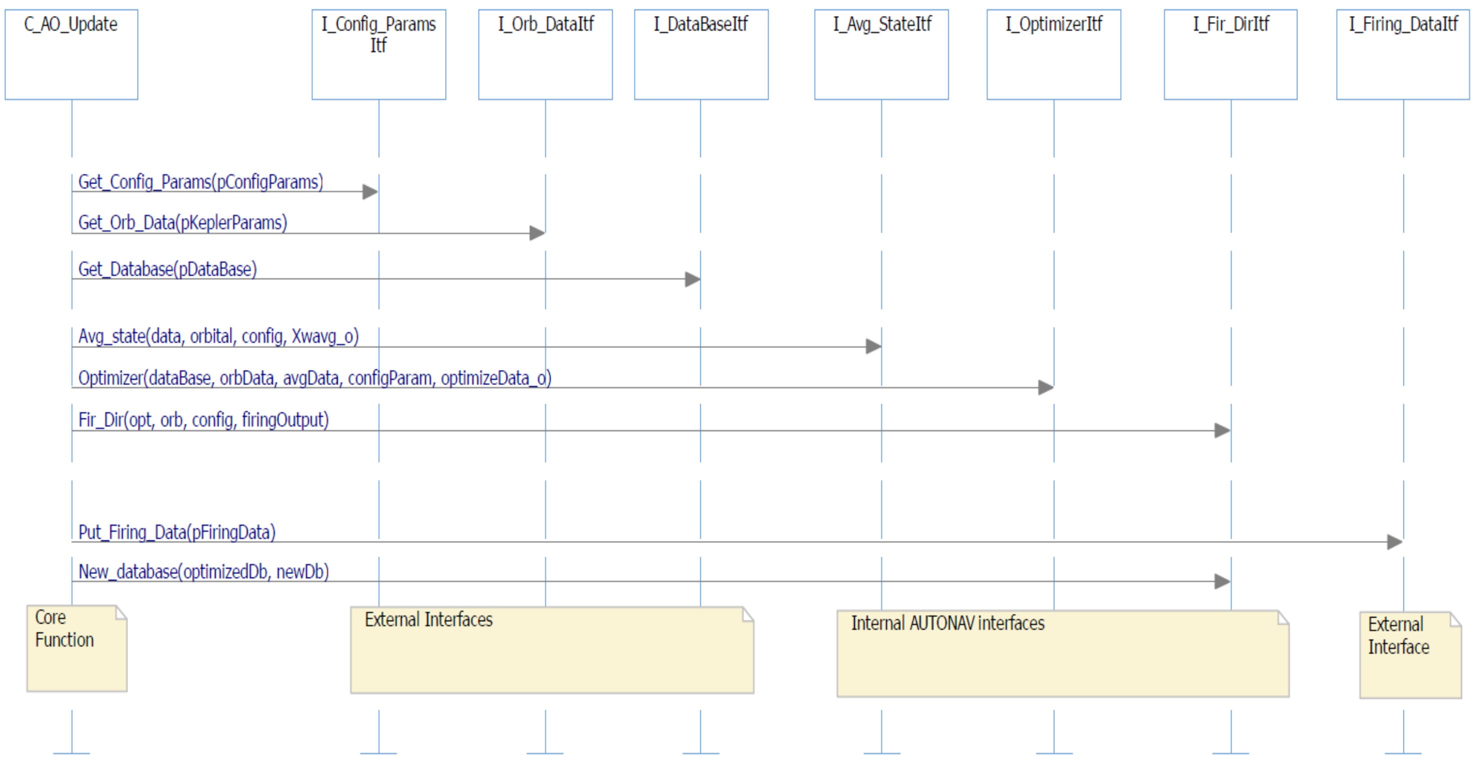



# SW Modules Used Interfaces

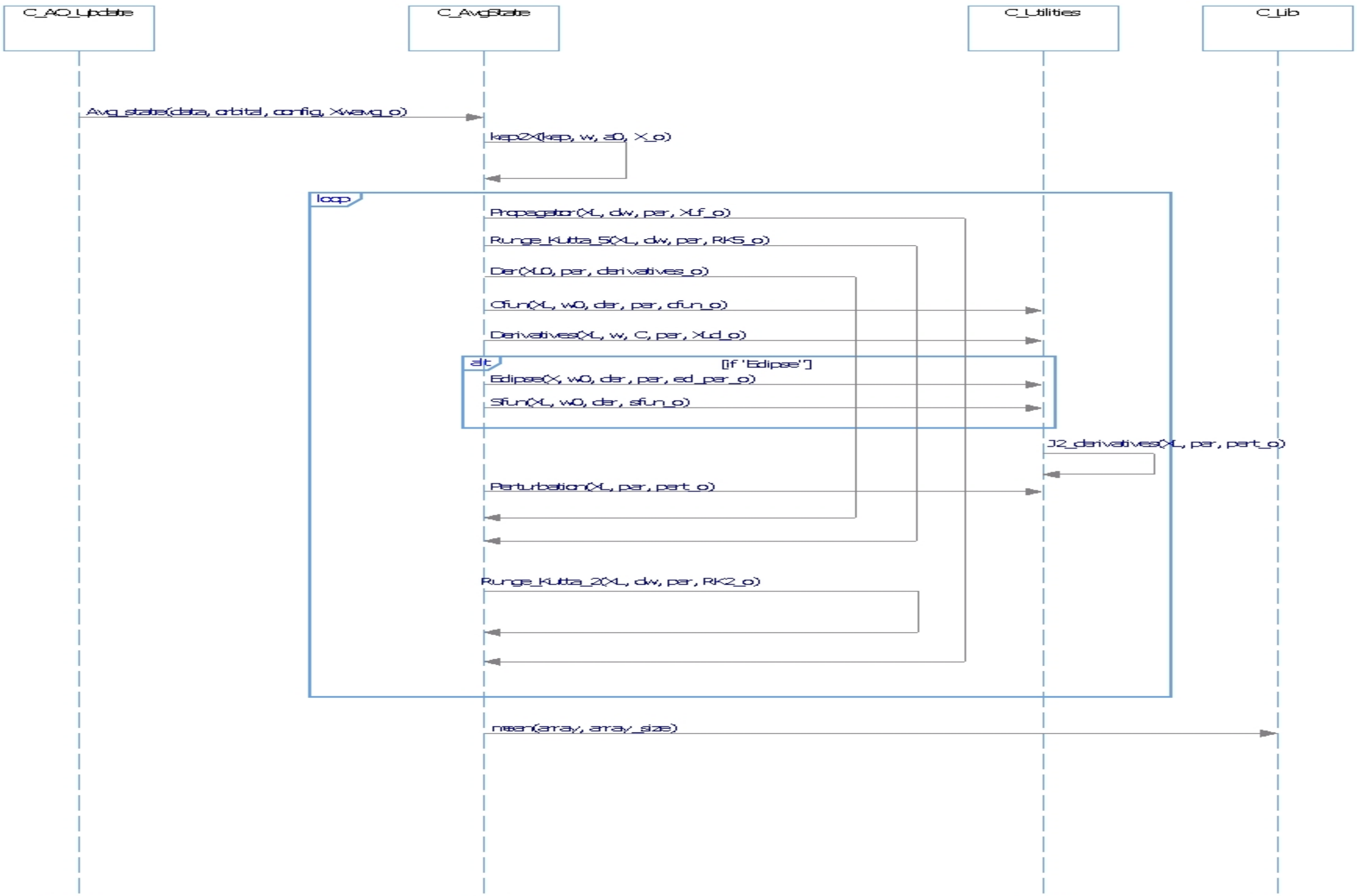



# Generic algorithm Call Sequence

AutonavMain_SeqDiagram



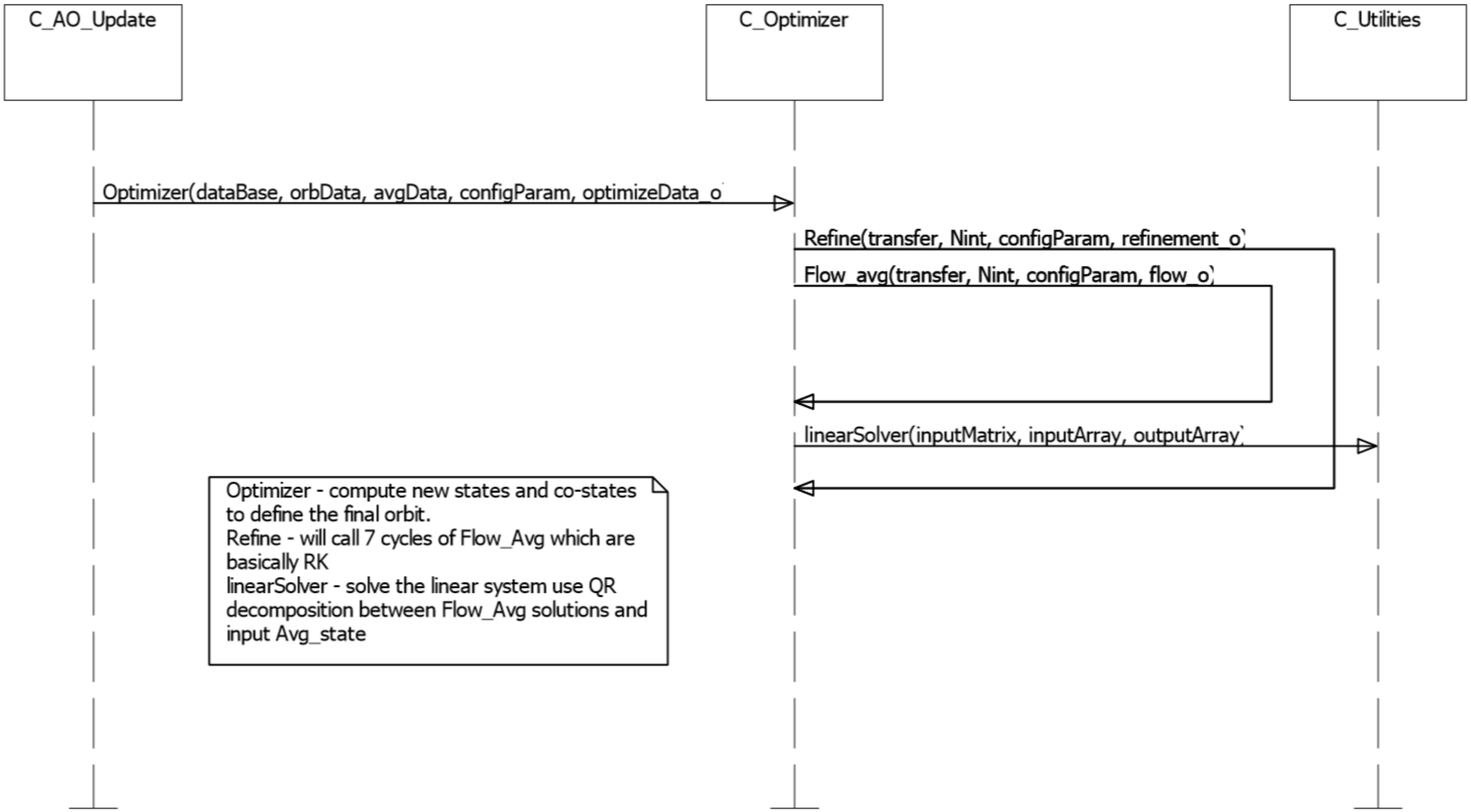

# Component Call Sequence - Average State




Average

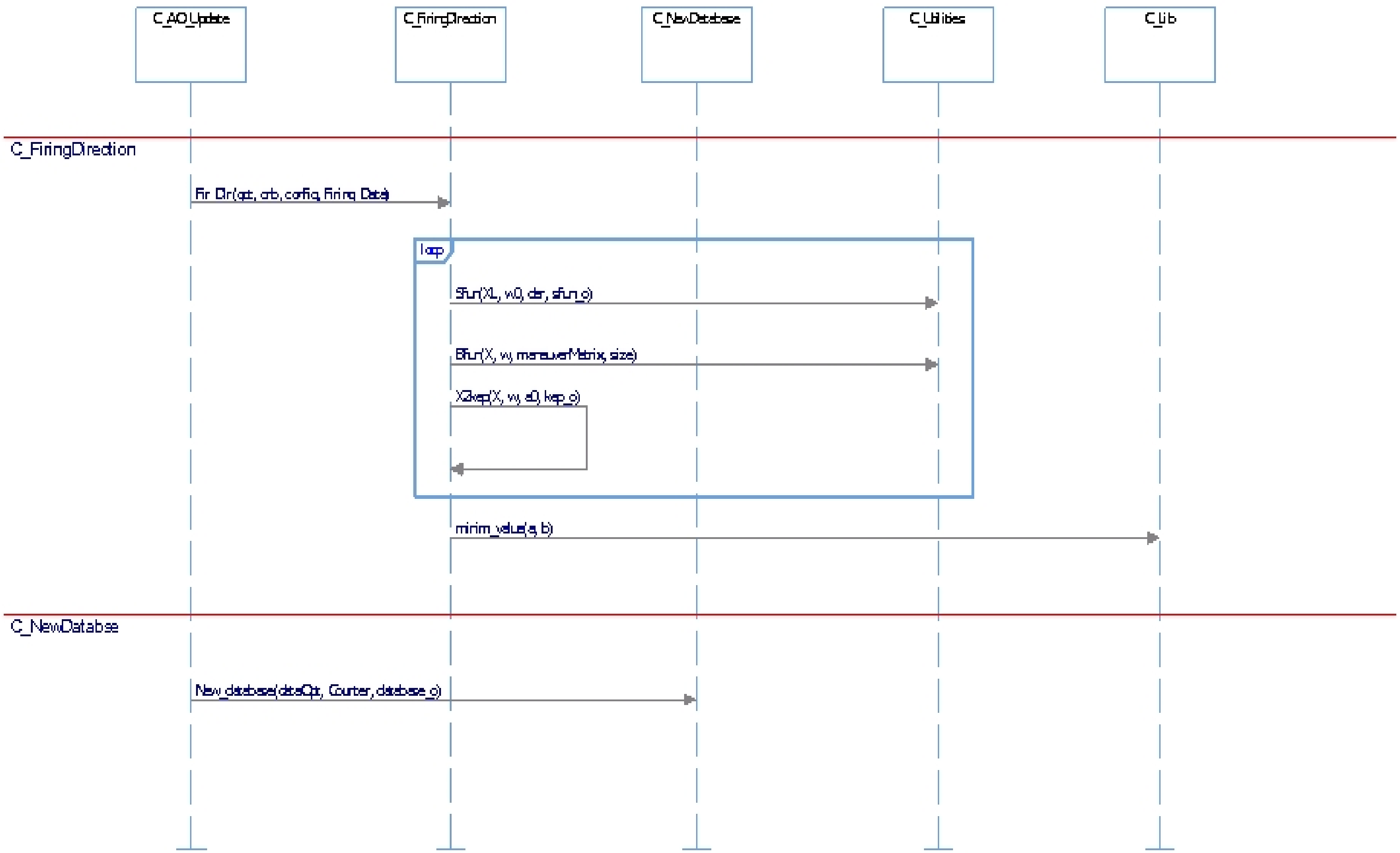
- Math lib calls
- Eclipse exception
- Disturbances


## Component Call Sequence – Optimizer




Convergence criteria: defined by threshold. Euclidian norm of the difference between new state and the old database state < 1e-6


# Component Call Sequence – Firing Direction



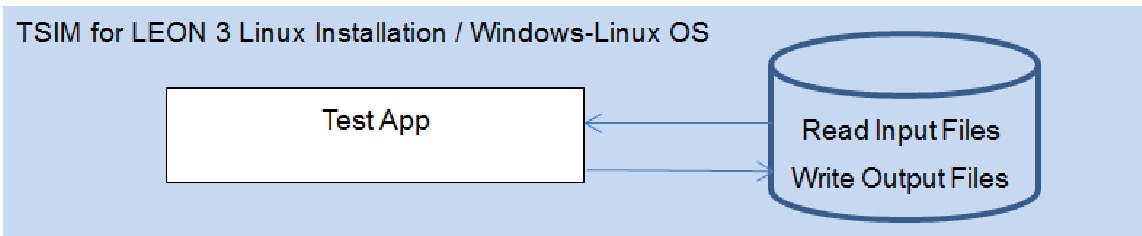

*Calculates direction Vector for* Thrusters at each transfer step Dep. O number of DB entries.

# Validation approach

## Goal

- C Components output vs Matlab reference within specified range
- Implemented interfaces are providing access according to the desired design/behavior

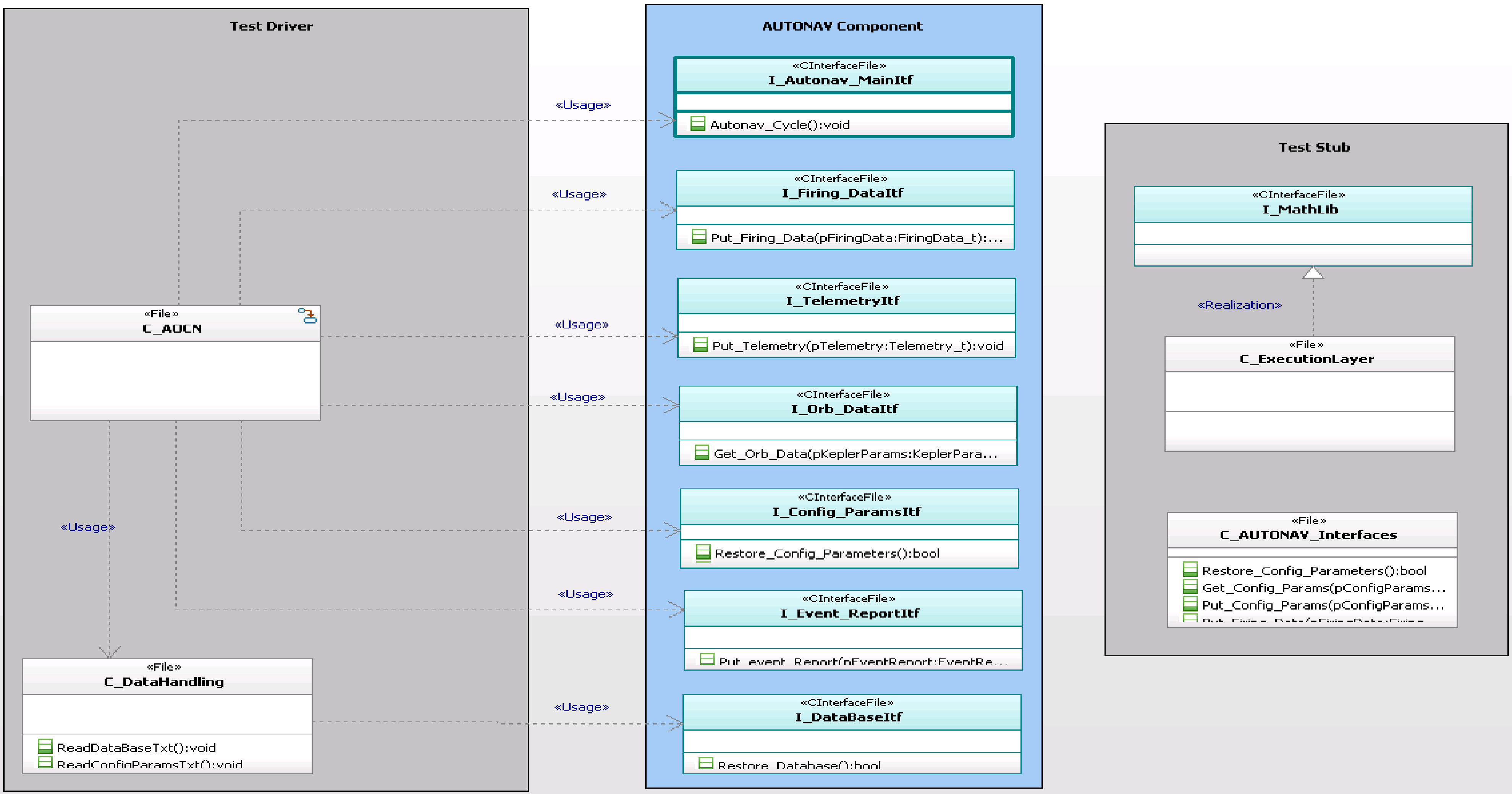

# Validation setup




- A Test Application will be written that loads internally in RAM the database provided as text files, and then the AUTONAV SW will be called
- inputs files represent test scenarios will be used for validating the C implementation regarding **functionality**, **precision** and **performance**

# Autonomous Orbital Navigation (AUTONAV) On Board Software

## Validation – SW architecture




## Validation challenges, adaptations

- Uniformity of Matlab Algorithm and C implementation:
  - Matrix inversion
  - Factorization
  - Linear solver
  - Matrix/Array operations
- Test scenarios execution on x86 and TSIM

**NaN:** Rounding errors lead to ‘zero’ input

**NaN side effects**

- **Numerical issues** within algorithm ***iterations(in matrix inversion, factorization, extrapolation, discontinuity points***)


# Results Precision (1/6)

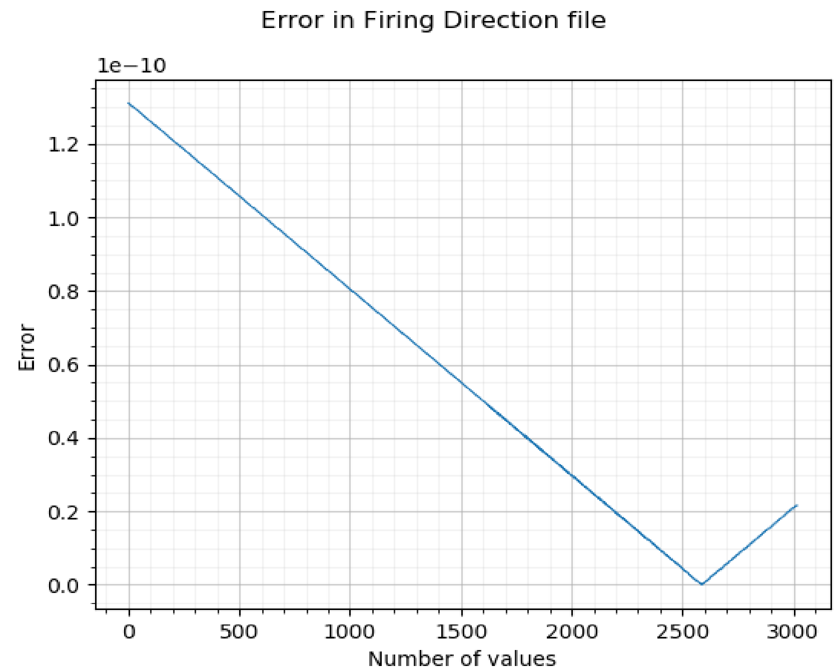
➢ **Absolute errors between Matlab and C execution of the output <u>Database</u>**


- $error_{value} = |val_C - val_{Matlab}|$


# Results Precision (2/6)

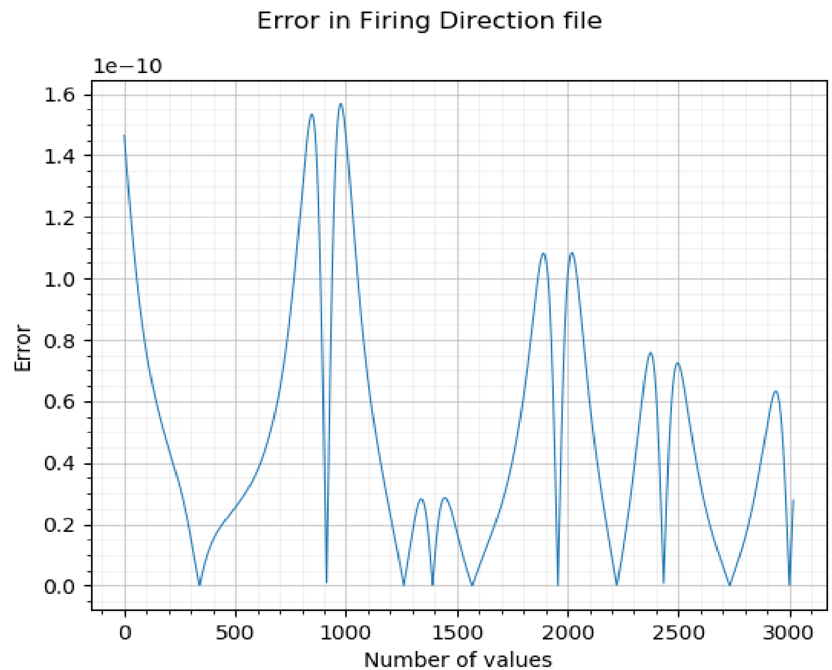
- ➢ **Absolute errors between Matlab and C execution of the output <u>Firing Direction</u>**
- ➢ **This component repesents the <u>anomaly to the targeted orbit</u>**

$$error_{value} = |val_C - val_{Matlab}|$$




# Results Precision (3/6)

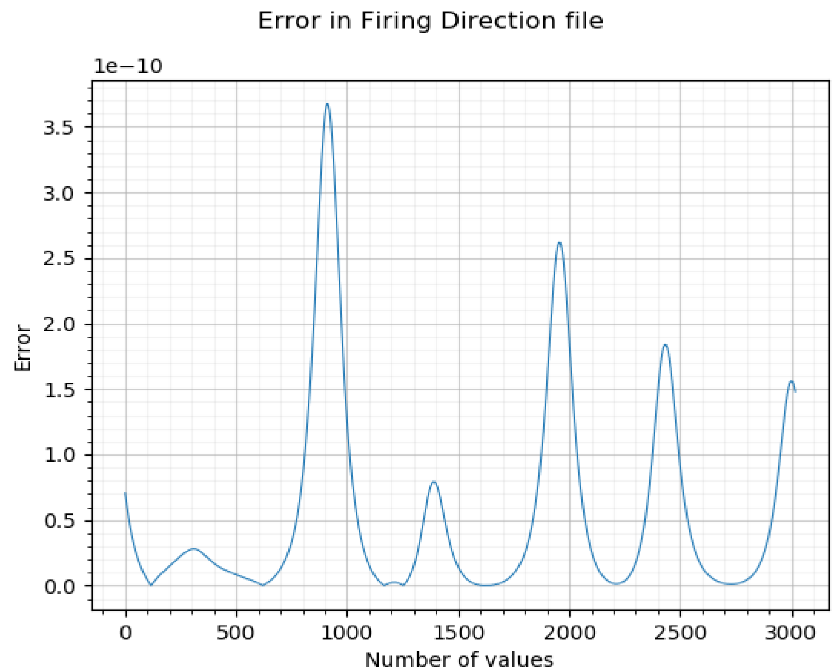
- ➢ **Absolute errors between Matlab and C execution of the output <u>Firing Direction</u>**
- ➢ **This component repesents the <u>control action for thrusters</u> (tri-dimensional component)**




$$error_{value} = |val_C - val_{Matlab}|$$


# Results Precision (4/6)

- Absolute errors between Matlab
- and C execution of the output
- <u>Firing Direction</u>
- This component repesents the
- <u>control action for thrusters</u>
- (tri-dimensional component)

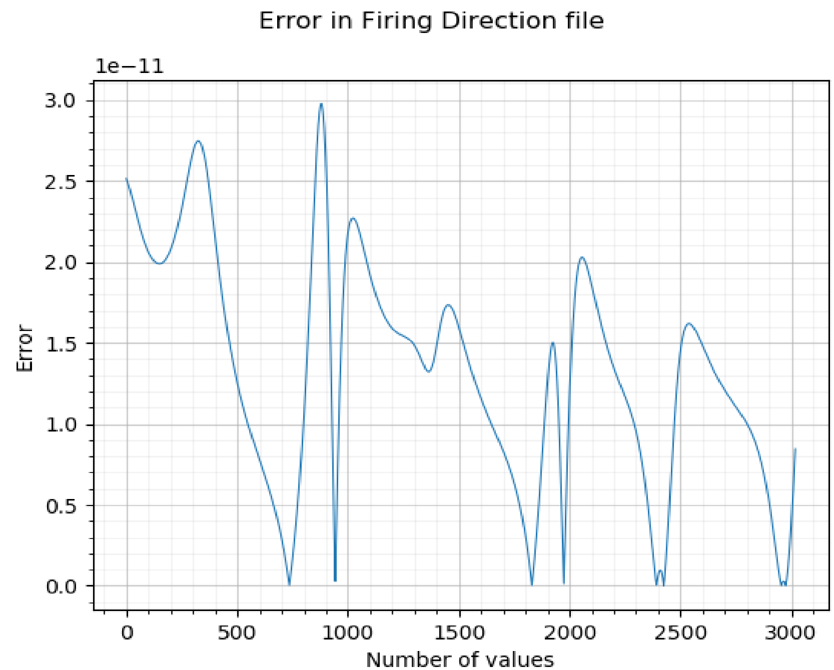



- $error_{value} = |val_C - val_{Matlab}|$


# Results Precision (5/6)

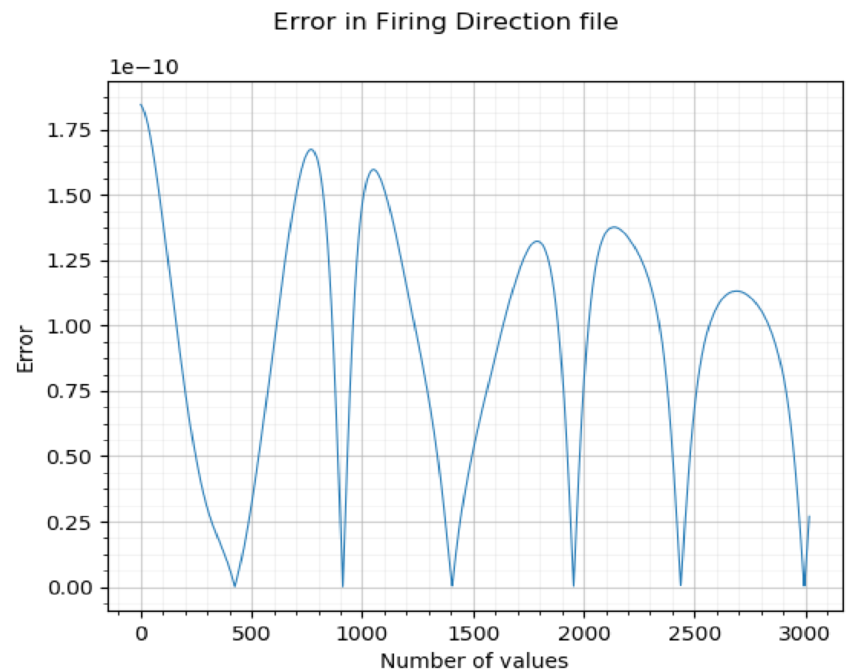
- **Absolute errors between Matlab and C execution of the output <u>Firing direction</u>**
- **This component repesents the <u>control action for thrusters</u> (tri-dimensional component)**

$$error_{value} = |val_C - val_{Matlab}|$$




# Results Precision (6/6)

- **Absolute errors between Matlab and C execution of the output <u>Firing Direction</u>**
- **This component repesents the <u>force control</u> able to turn on/off the engines**




- $error_{value} = |val_C - val_{Matlab}|$


# • Results Performance

- **Leon3 CPU performance was assessed with Dhrystone and Whetstone tests**
- **As the software is split in three components**
  - Average state computation
  - Optimizer (the most CPU intensive)
  - Firing direction
- Average and Firing direction are consuming less than 1% of overall cost
- One iteration of '**optimizer**' take 3h and 20 minutes on LEON3
- In worst case scenario with 50 optimizer iterations in these conditions the test will take nearly 7 days to execute


# Lessons Learned (1/2)

- Thales Romania has learned:
  - AUTNAV algorithm specifics (from orbital parameters to thrusters, precision and convergence related topics)
  - How to develop Flight software compliant with ECSS standards
  - To work with space specific CPUs (LEON3,4 family)
  - To address and trace precision specific problems on C compiler and Matlab framework
    - Matrix inversion
    - Power function mismatch between Matlab and C
    - Matrix and arrays operations from Matlab were translated to C approach


# • Lessons Learned (2/2)

- ❑ To develop SW components that is compatible with On Board SW Platform
- ❑ To tailor ECSS standards depending on the needs and the specifics of the project
- ❑ Thales Romania has managed the project in a novel way by developing architecture, design and development in an UML model which was also used for documentation generation and can support Requirements traceability and Unit Testing

# • Conclusions

- ❑ LEON3 performances are not enough, but some optimization of the algo and compiler options will be explored on Phase 2
- ❑ Very good match between Matlab and C implementations
- ❑ TSR successfully ported the algorithm in C, clearing the way to qualify the AUTONAV SW for space flight
- ❑ TSR knowledge in the navigation and space qualified SW field has consolidated to a level that recommends it for future collaborations


## Future Work – new HW baseline, TRL-7 target

- TAS-I clarifies that the adoption of the more powerful LEON4FT multicore CPU (GR740) is ongoing and implemented by the development of the IPAC (SMU-NG) computer. This is being developed in two versions (HIREL with limited number of “COTS” components and “Full HIREL”. The latter is planned to reach TRL-7 in mid-2021 thanks to Italian National programme (Ital-GovSatCom) which is intended to use the AUTONAV as baseline


## Future Work – qualified SW component (Cat C)

- ❑ Optimization of performances (algorithm options / compiler options)
- ❑ Fulfil Product Assurance activities
- ❑ Run Unit Test with ECSS specified target coverage
- ❑ Perform Integration tests, completion of traceability
- ❑ Provide complete documentation line
- ❑ Analyse adoption of new HW baseline (LEON4FT)


# THANK YOU