TUDAT: the open-source astrodynamics toolbox of Delft University of Technology

Kevin Cowan & Dominic Dirkx

Astrodynamics and Space Missions

Faculty of Aerospace Engineering — Delft University of Technology

ICATT 2018

7th International Conference on Astrodynamics Tools and Techniques

DLR Oberpfaffenhofen, Germany

6–9 November 2018

TUDelft

TUDAT: an introduction

- TUDAT: TU Delft Astrodynamics Toolbox
  - geared towards 'solar-system scale' applications
  - mission design: accurate orbit propagation, optimization
  - natural body evolution over 'short' time scales

- User community
  - historically TU Delft MSc students and PhD researchers

- Used for research projects with various partners
  - JIVE, DLR, JPL, …

- Software written in C++
  - design driver: modularity

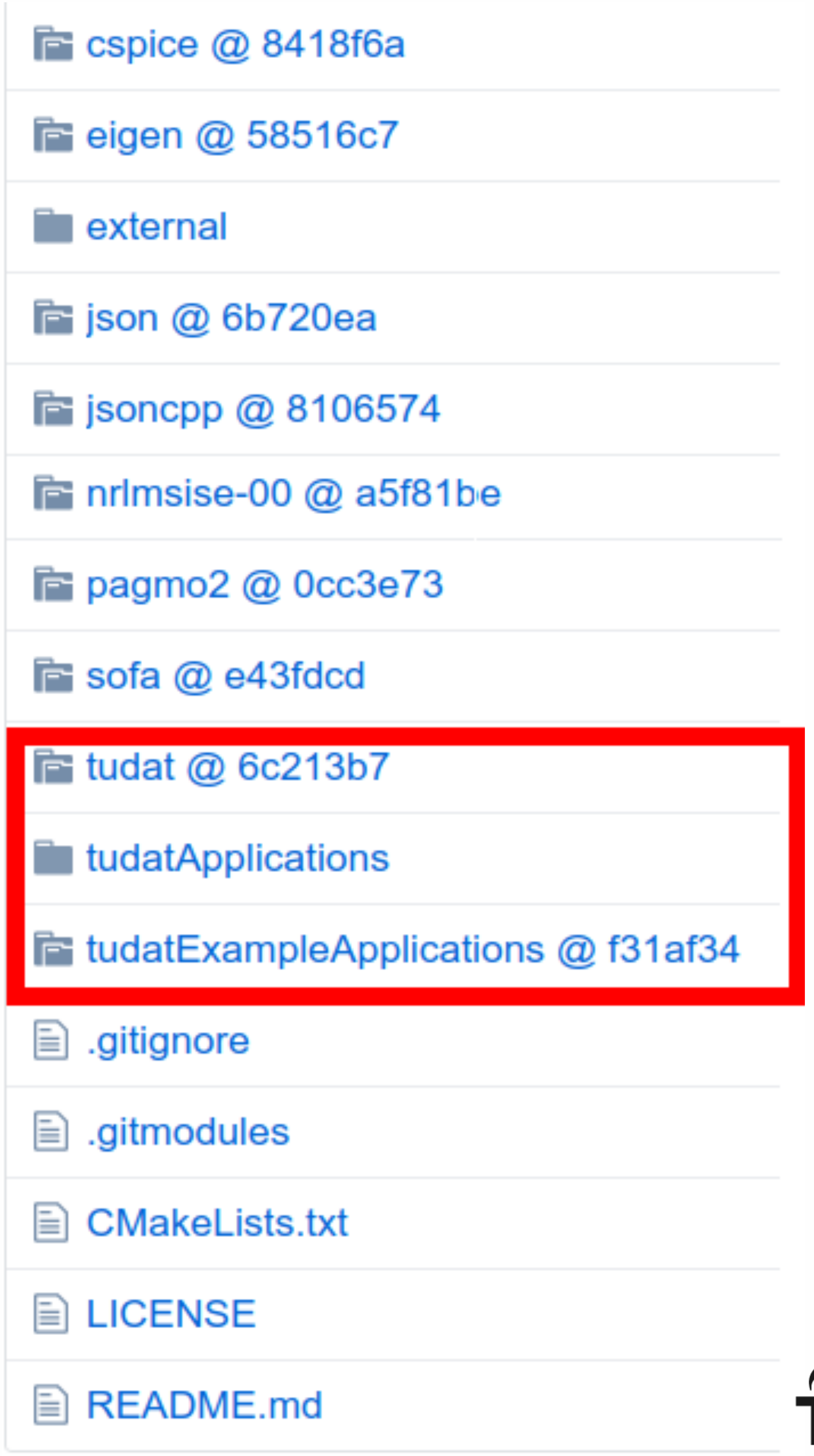
TUDAT: setup

- Wiki: guides, documentation, tutorials
  - tudat.tudelft.nl
- Code hosted on Github
  - github.com/tudat/tudatBundle

- Various external toolbox interfaces with TUDAT
  - Spice (solar system ephemerides)
  - Eigen (linear algebra)
  - Pagmo (global optimization)
  - …

- Code linked together using CMake

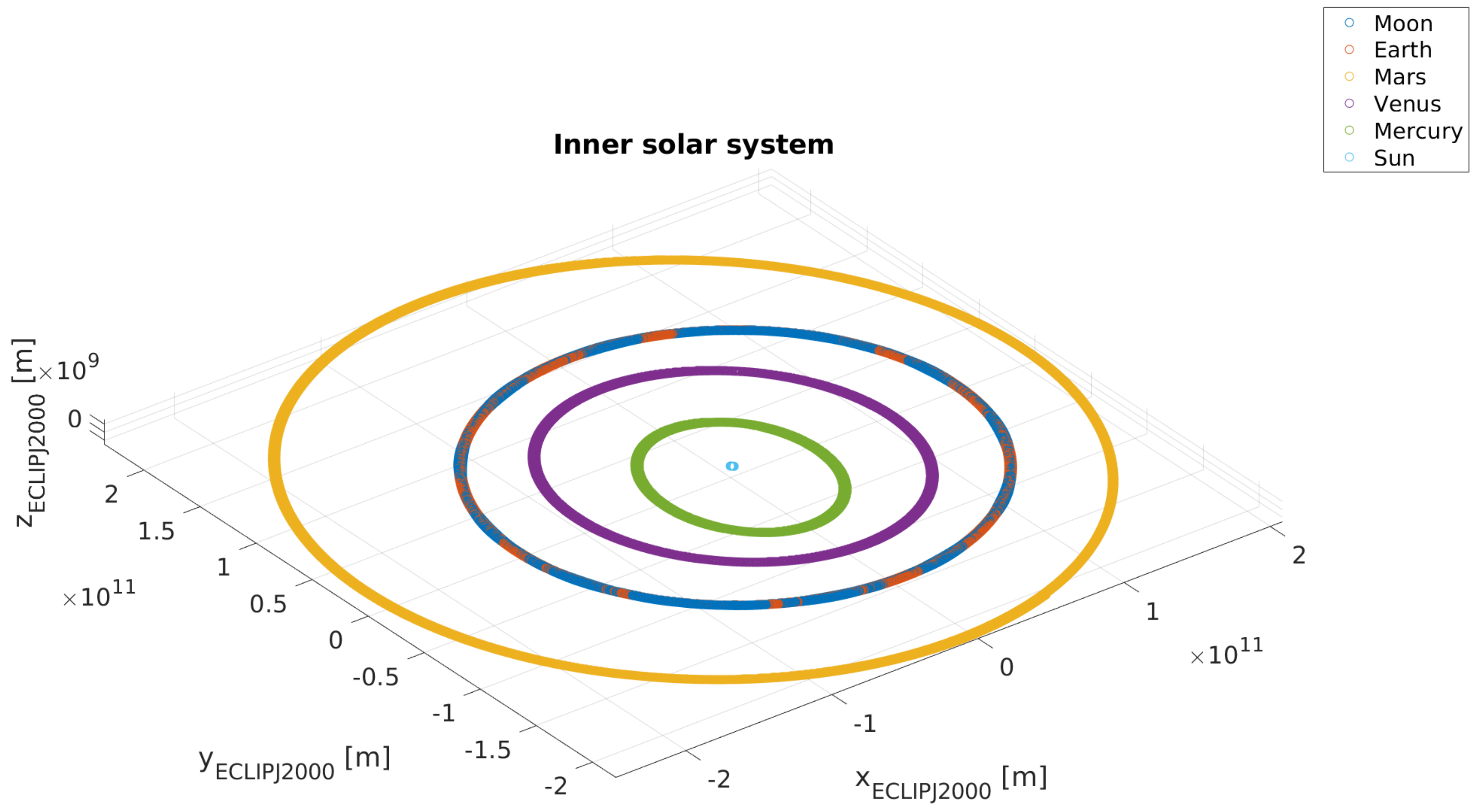
- Extensive unit tests to verify code

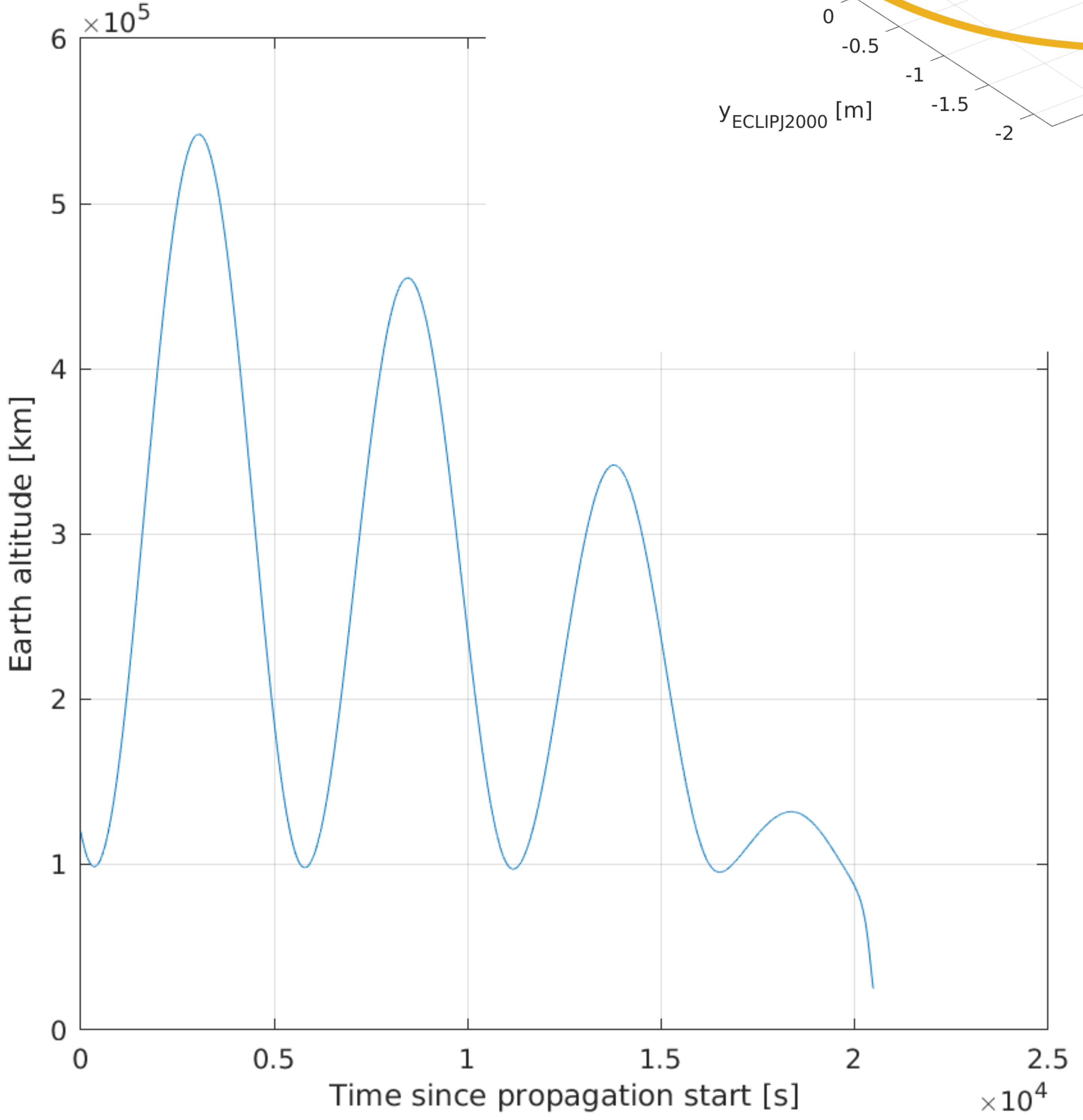


TUDAT: functionality

Accurate numerical state propagation is the core of TUDAT

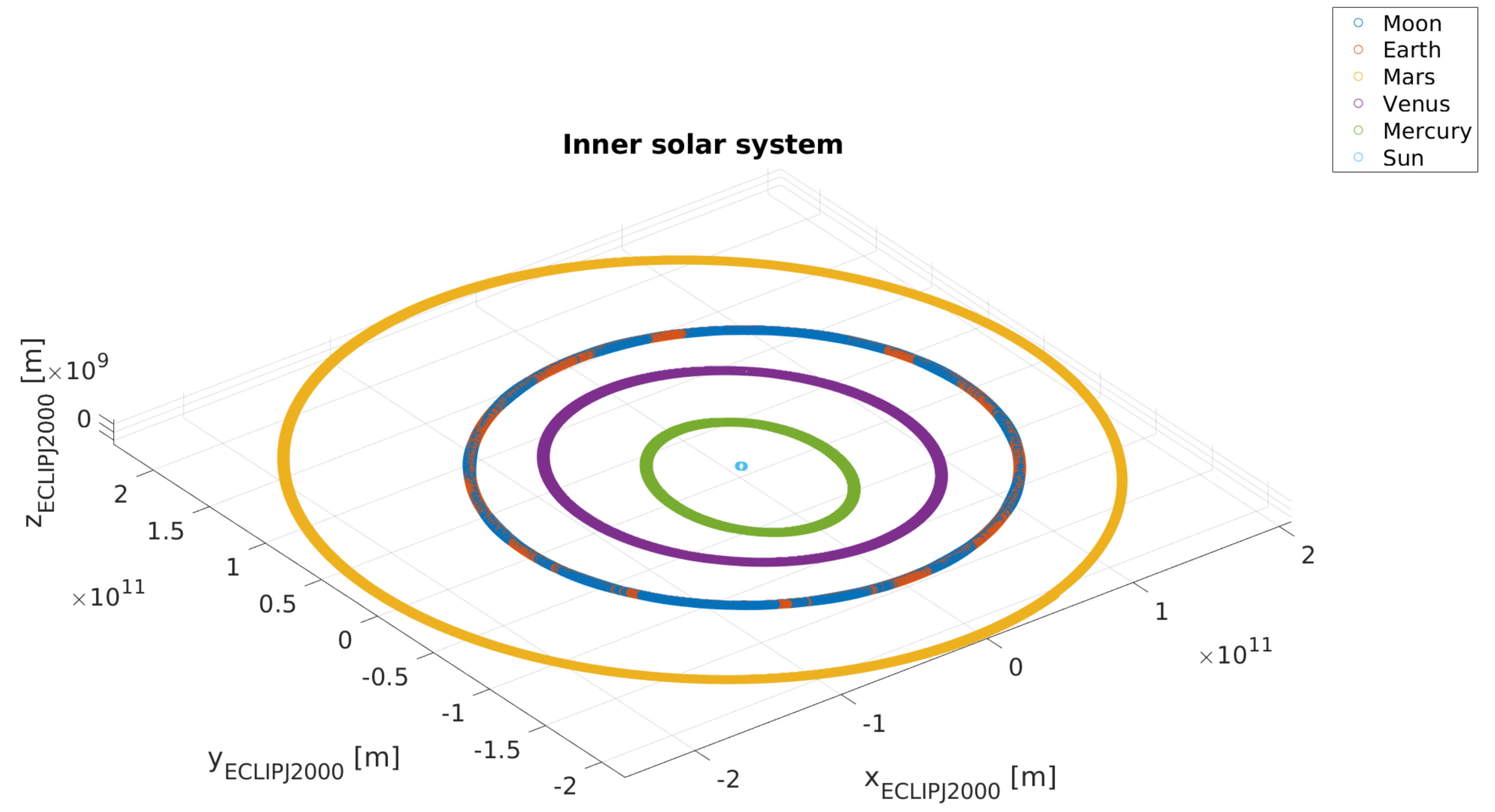
- Propagation
  - translational state
  - rotational state
  - body mass
  - user defined state derivative functions
  - …
  - options for the terminal conditions and dependent variables derived from the state

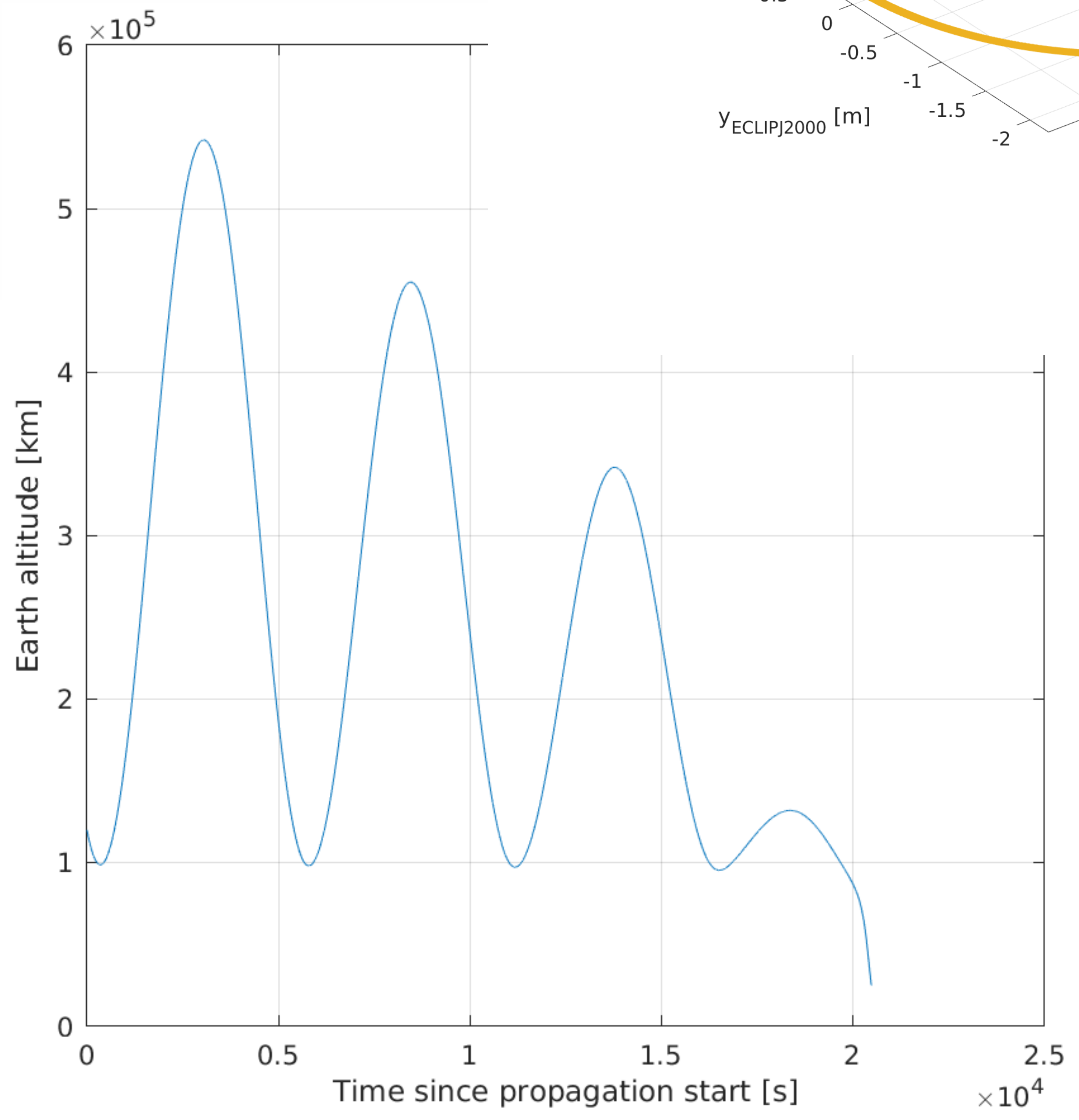




TUDAT: functionality

- Various integrators of fixed and variable step-size
  - Runge-Kutta 4
  - Runge-Kutta variable step-size (various orders)
  - Bulirsch-Stoer
  - Adams-Bashfort-Moulton

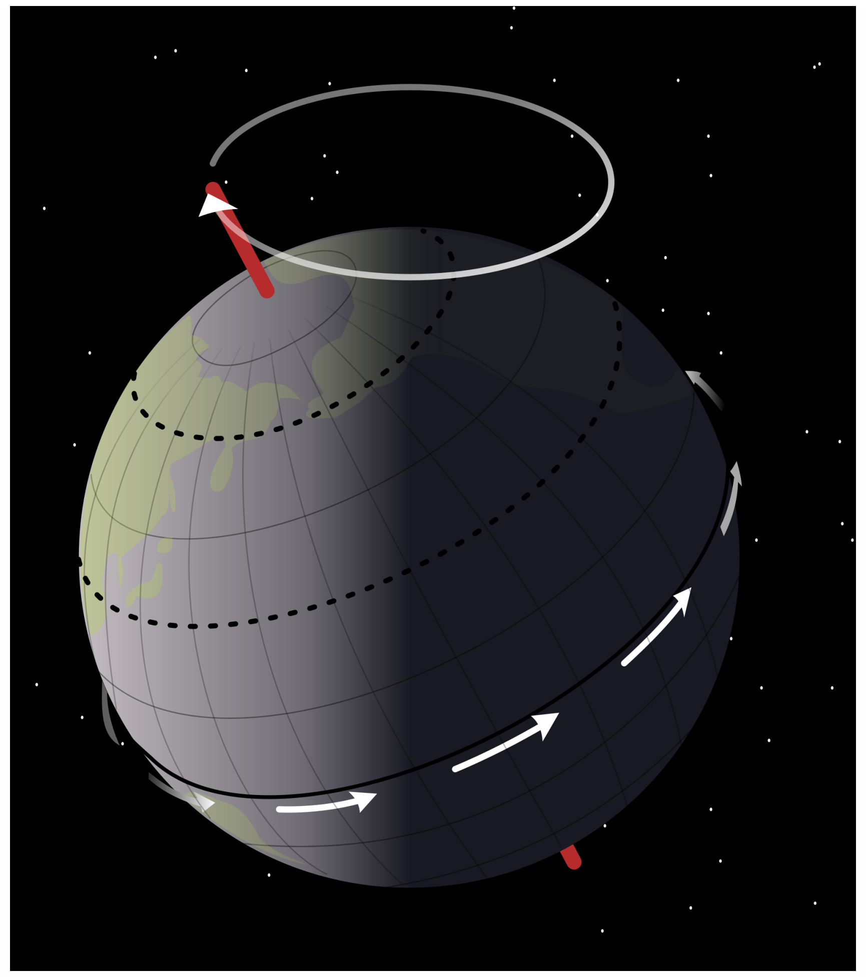


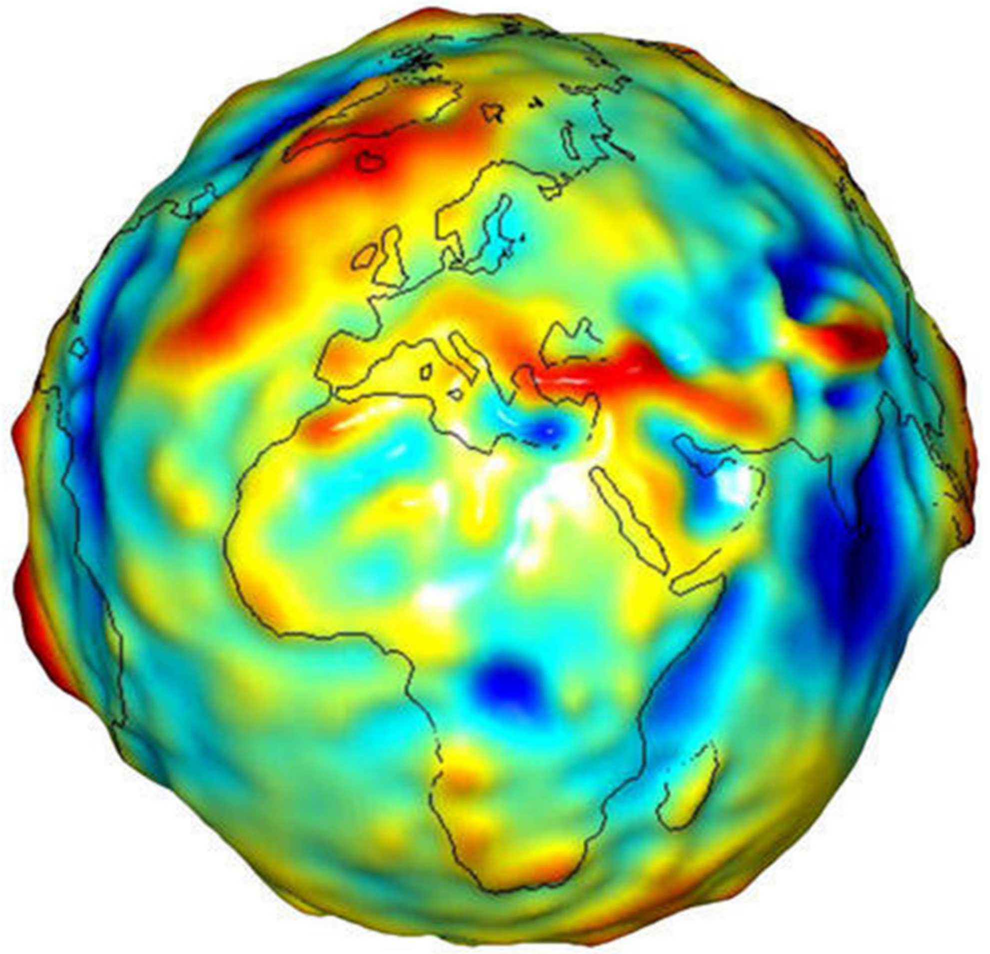


TUDAT: functionality

- A range of environment models
  - gravity fields
  - atmosphere
  - planetary orbits and rotations
  - radiation

- A range of acceleration models
  - point mass gravity
  - spherical harmonics gravity
  - aerodynamics
  - radiation pressure
  - thrust



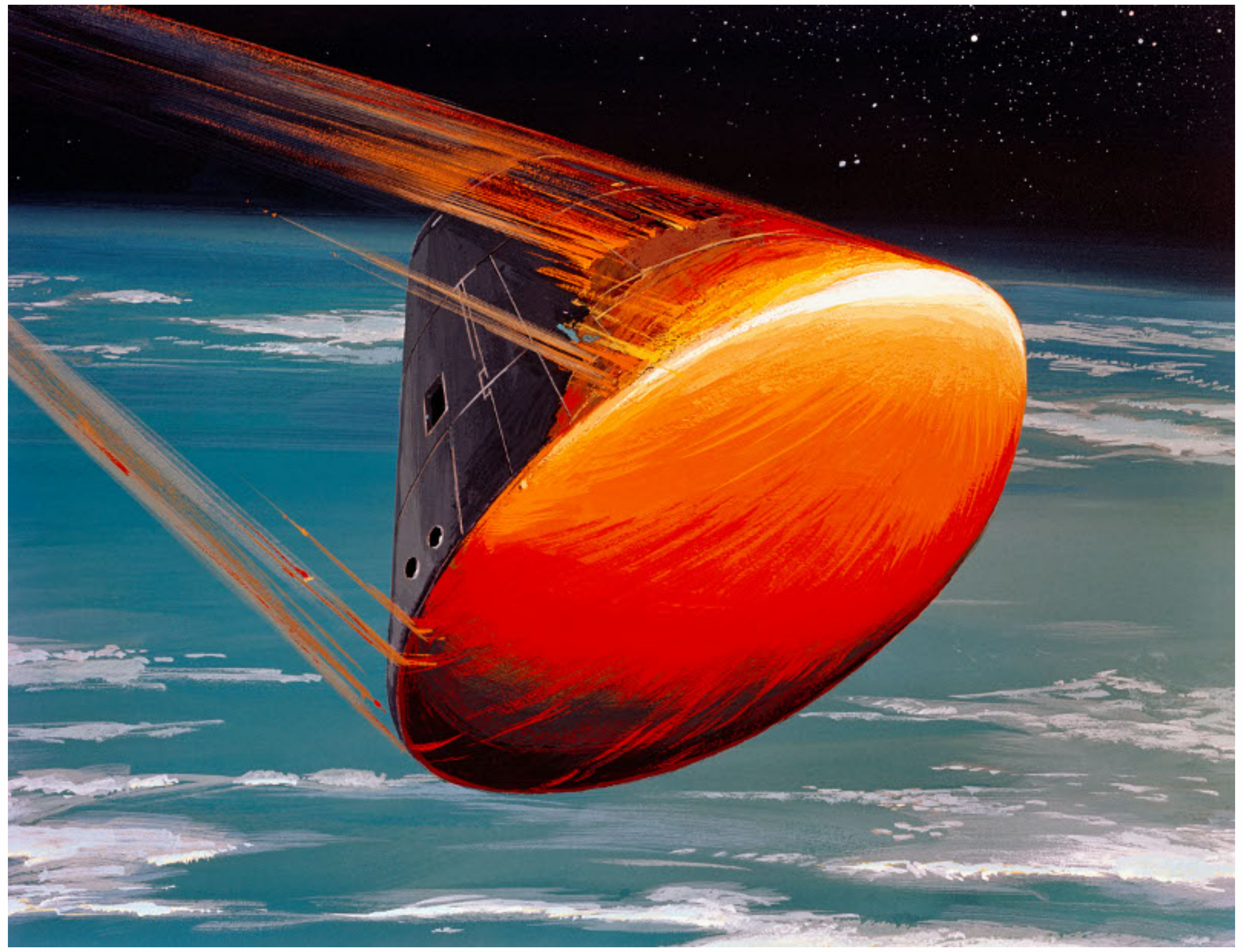


source: NASA

TUDAT: functionality

- Vehicle definitions
  - mass
  - aerodynamic coefficients
  - reference area
  - orientation
  - other parameters

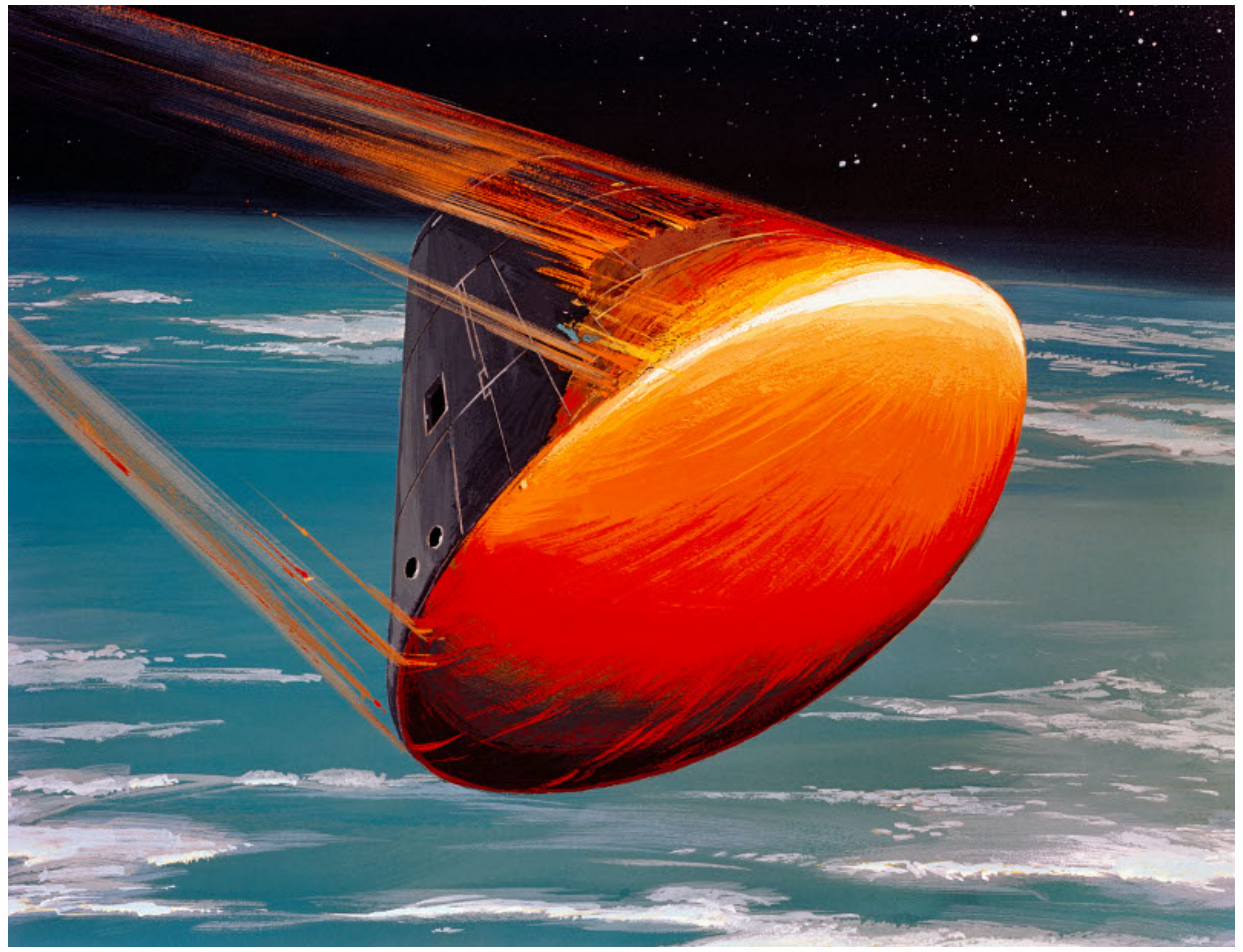
- Guidance models
  - custom made or pre-defined
  - aerodynamic guidance
  - thrust guidance



source: NASA

TUDAT: functionality

- Software architecture does not distinguish between natural and manmade objects
  - difference lies in the properties assigned to the bodies



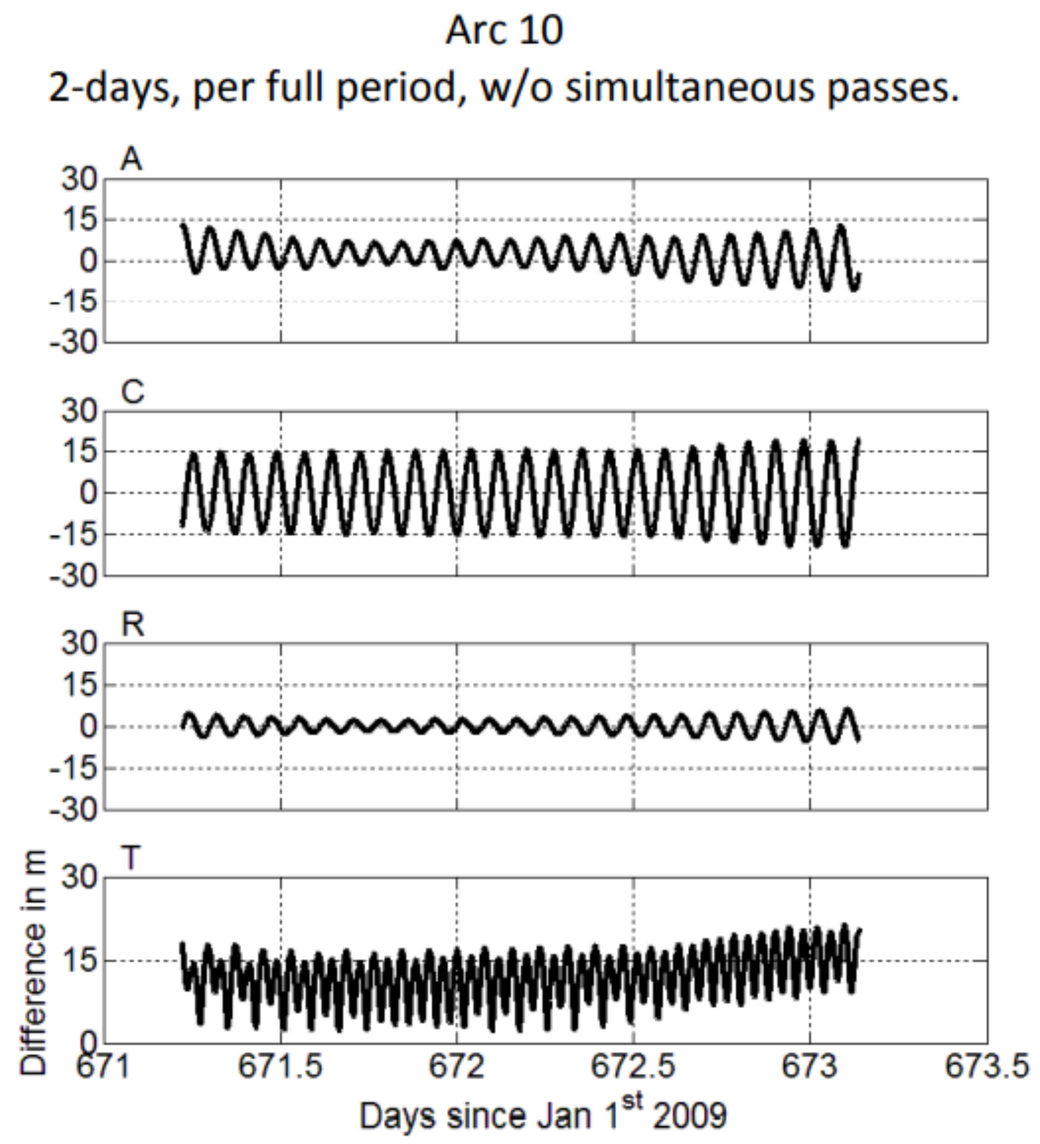
source: NASA

TUDAT: functionality

- Orbit determination
  - variational equations
  - flexible state/parameter estimation
  - observation models
  - …

- Framework is general, but implementation is geared towards planetary missions

- Historically used most for planetary applications
  - to be used for JUICE-PRIDE

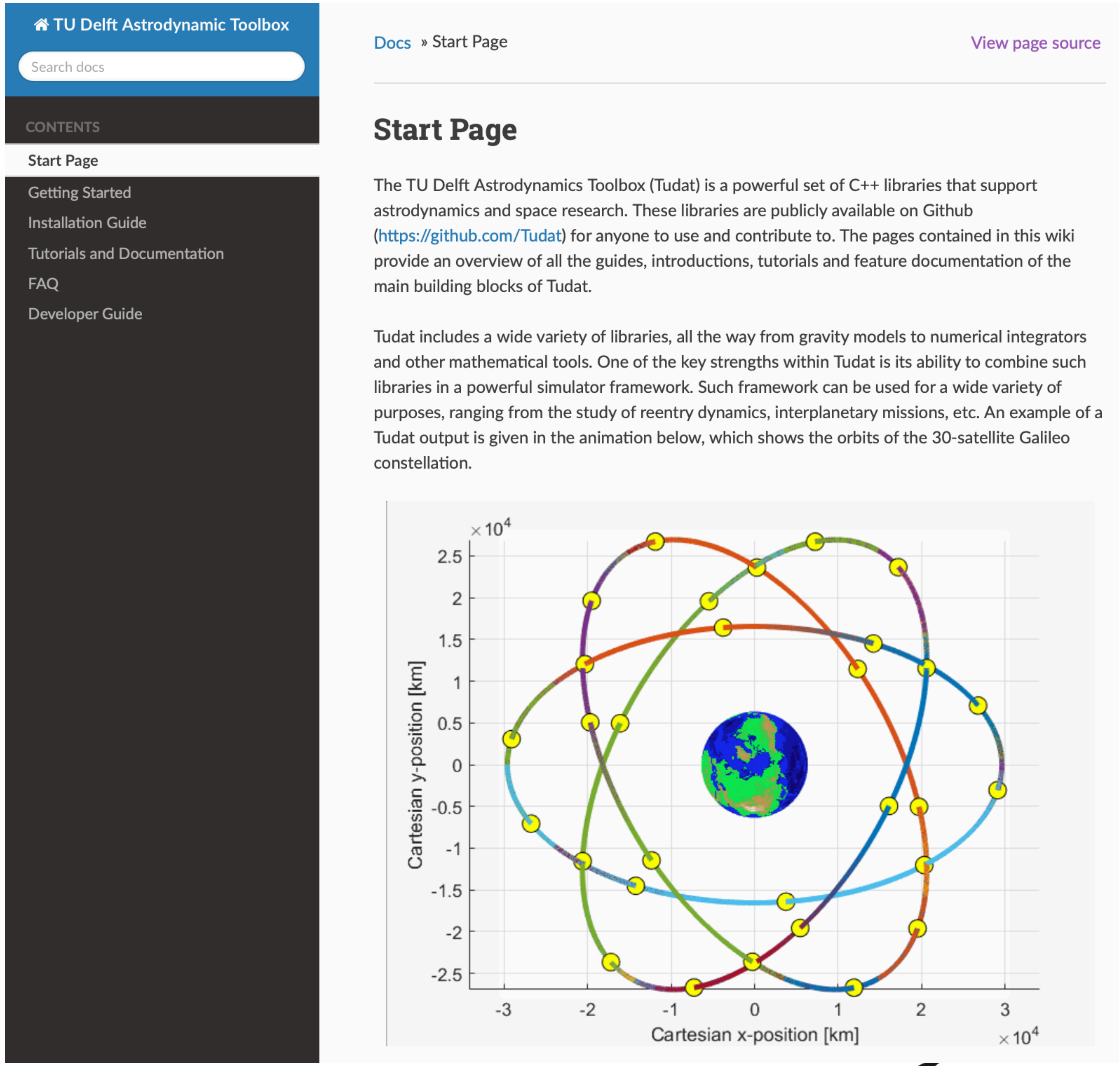


Bauer (2017)

TUDAT: documentation

- Software is documented on our website
  - tudat.tudelft.nl
  - in-code Doxygen documentation

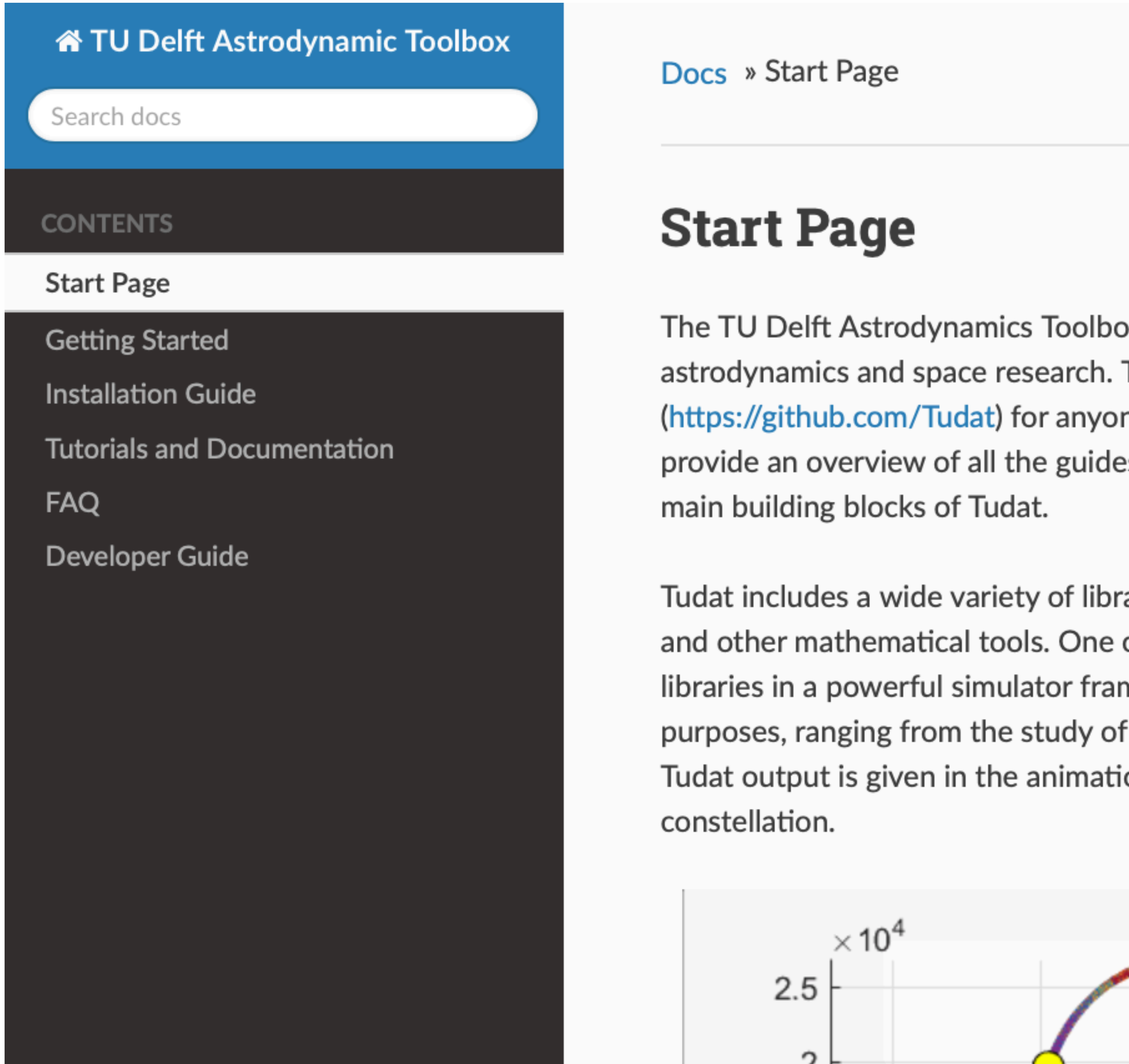
- Starting point for new users: example application tutorials



TUDAT: application tutorials

TUDAT tutorials (ie. walk-throughs)

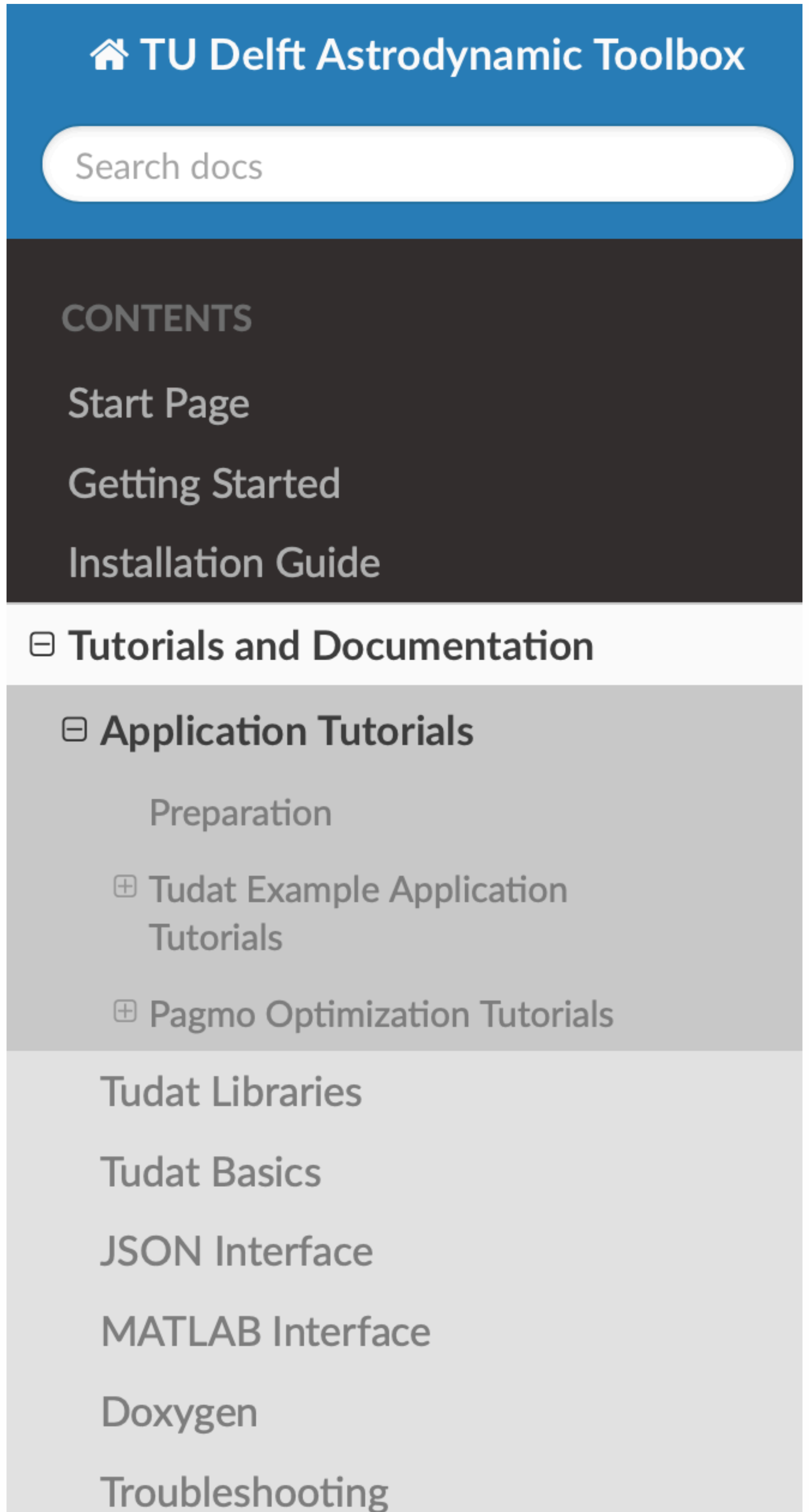
- Unperturbed Earth-orbiting satellite
- Perturbed Earth-orbiting satellite
- Un-guided capsule entry
- Inner solar system propagation
- Use thrust: thrust force along velocity vector
- Use thrust: user-defined thrust vector
- Tabulated atmosphere usage
- Comparison of propagator types
- Kalman filter for state estimation
- Variational equations propagation
- Orbit determination & parameter est.
- MGA trajectory design



TUDAT: application tutorials

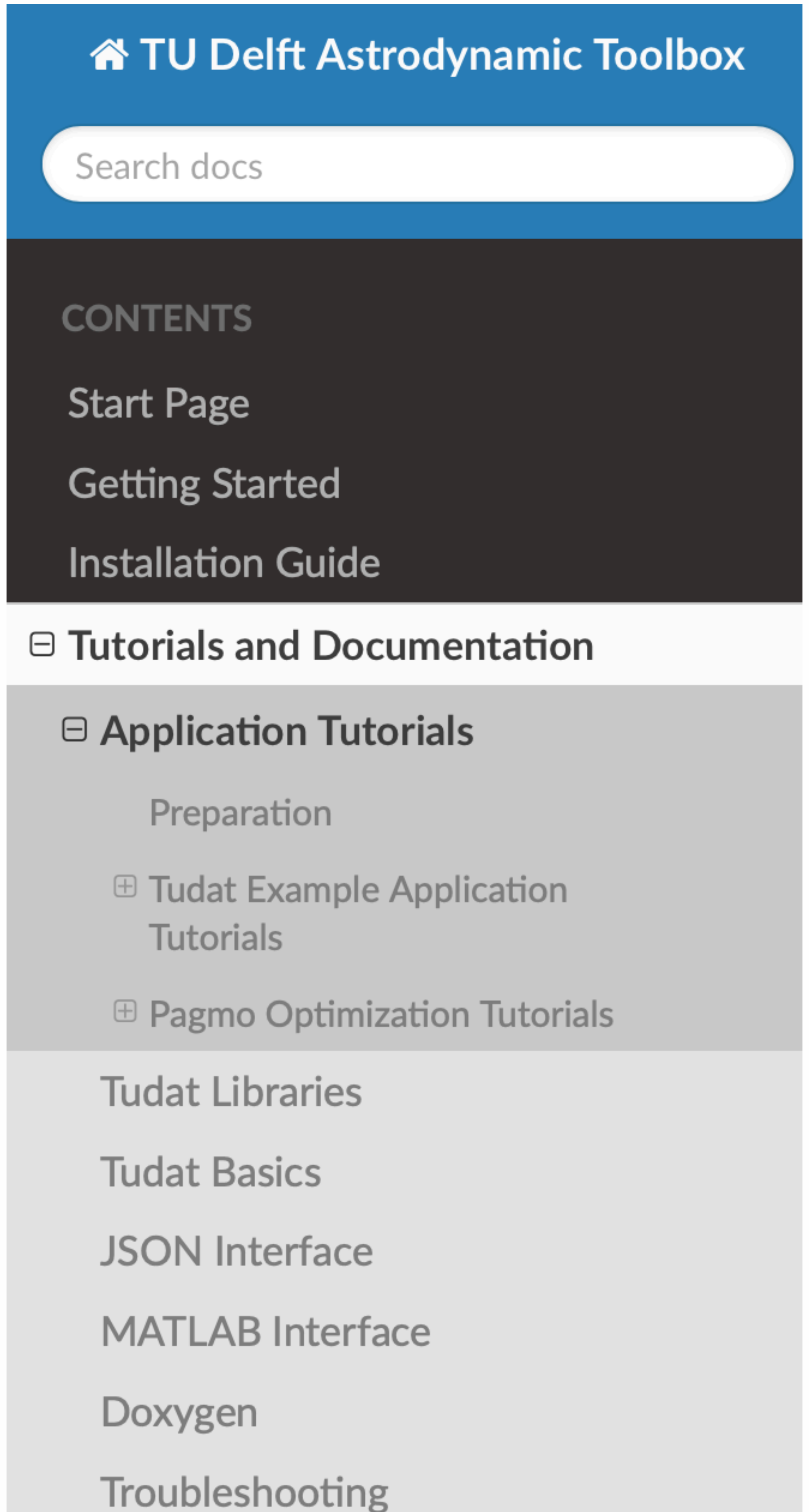
TUDAT tutorials (ie. walk-throughs)

- Unperturbed Earth-orbiting satellite
- Perturbed Earth-orbiting satellite
- Un-guided capsule entry
- Inner solar system propagation
- Use thrust: thrust force along velocity vector
- Use thrust: user-defined thrust vector
- Tabulated atmosphere usage
- Comparison of propagator types
- Kalman filter for state estimation
- Variational equations propagation
- Orbit determination & parameter est.
- MGA trajectory design



PAGMO

- scientific library for massively parallel optimization
  - bio-inspired & evolutionary algorithms
  - optimization algorithms (simplex methods, SQP methods, interior points methods, …)
  - combine to exploit algorithmic cooperation via the asynchronous, generalized island model
- solve constrained, unconstrained, single objective, multiple objective, continuous & integer optimization problems, stochastic & deterministic problems



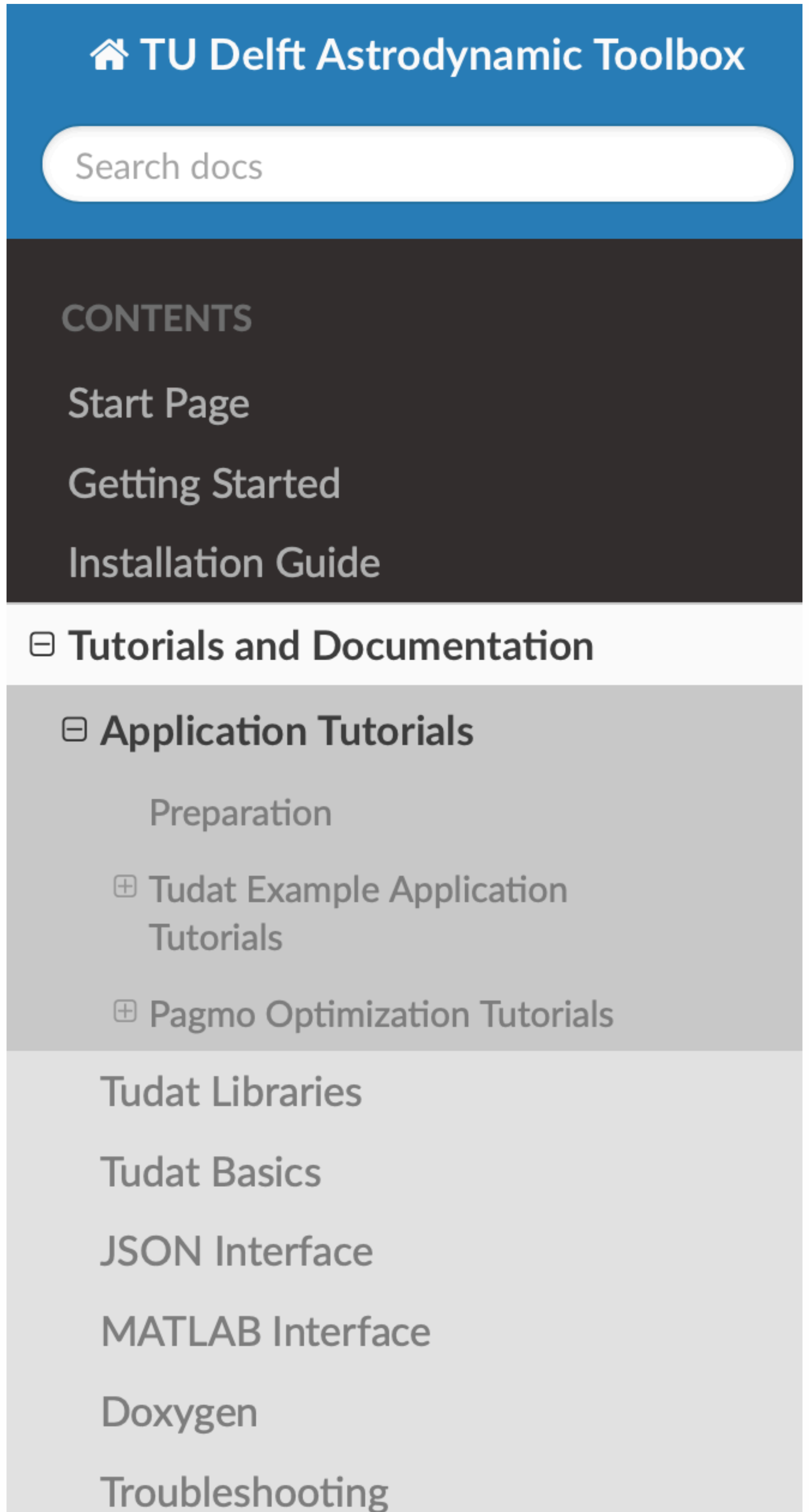
ref: https://www.esa.int/gsp/ACT/open_source/pagmo.html
code: https://esa.github.io/pagmo2/

TU Delft

TUDAT: application tutorials

PAGMO optimization tutorials

- Himmelblau optimization
- CEC 2013 optimizer comparison
- Earth-Mars transfer (multi-objective)
- Multiple gravity assist transfer
- Propagation targeting

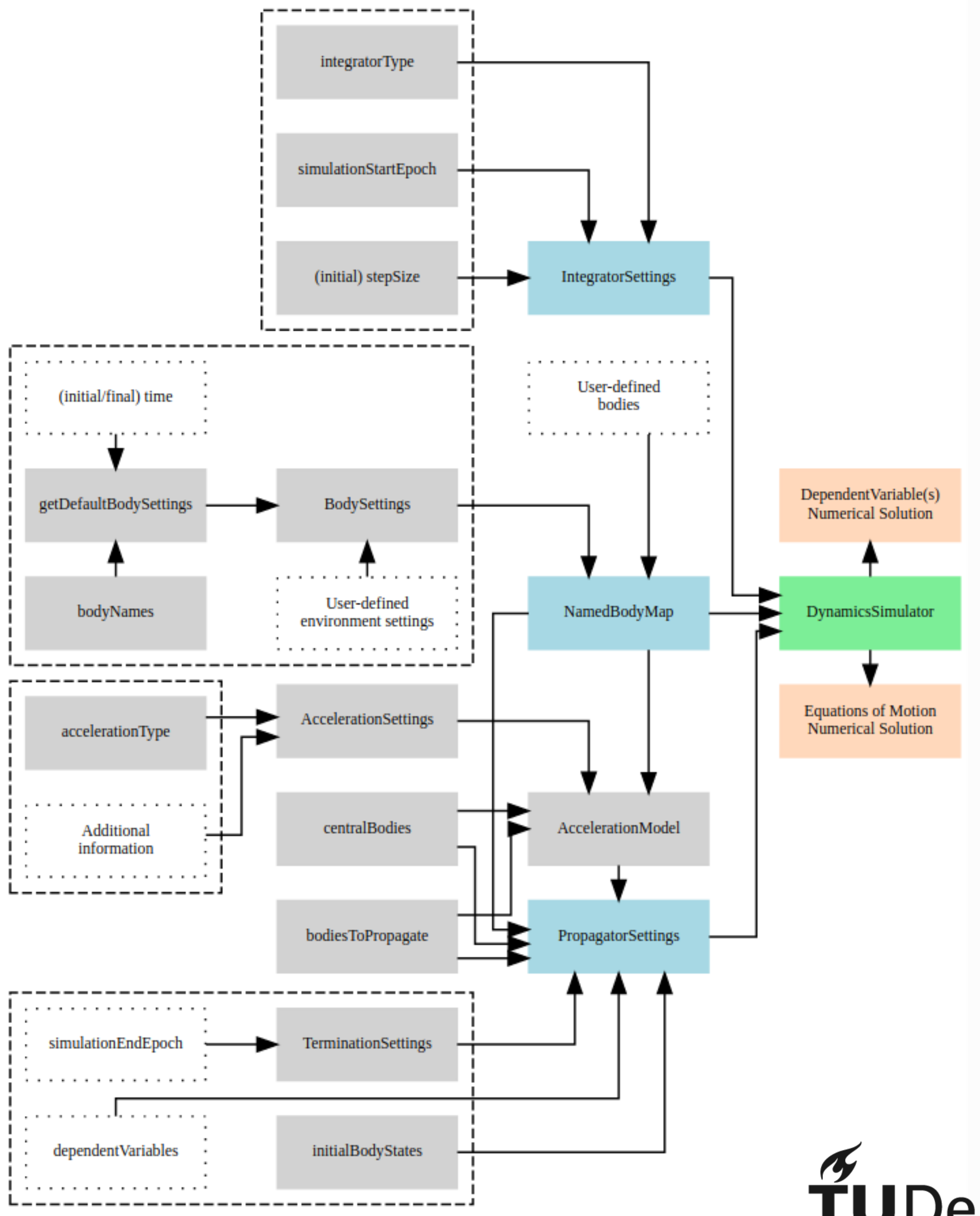


ref: https://www.esa.int/gsp/ACT/open_source/pagmo.html
code: https://esa.github.io/pagmo2/

TUDAT: usage

- TUDAT contains many building blocks
  - blocks set up in a modular fashion
  - combining different blocks (relatively) straightforward

- Users have many options
  - making use of all of TUDAT has a bit of a learning curve



TUDAT: usage

class `IntegratorSettings`

This class is used to define the settings for fixed step-size integration. The constructor for this base class is:

```
IntegratorSettings< TimeType >( integratorType,
                                simulationStartEpoch,
                                fixedStepSize )
```

where:

- `TimeType`

  Template argument used to set the precision of the time, in general `double` is used. For some application where a high precision is required this can be changed to e.g. `long double`.

- `integratorType`

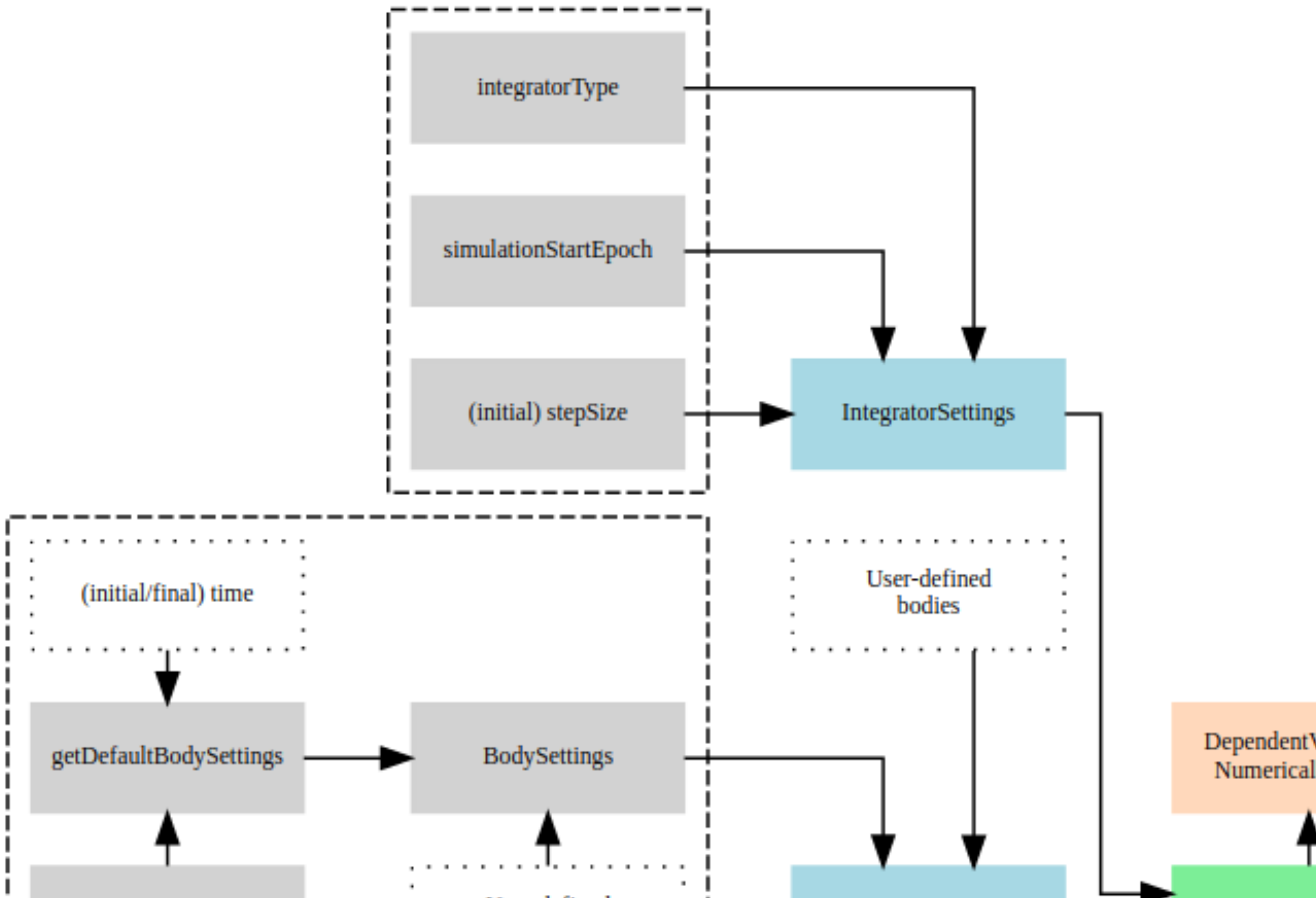
  `AvailableIntegrators` which defines the fixed step-size integrator type to be used. Currently the only options available are `euler` and `rungeKutta4`.

- `simulationStartEpoch`

  `TimeType` that defines the simulation's start epoch.

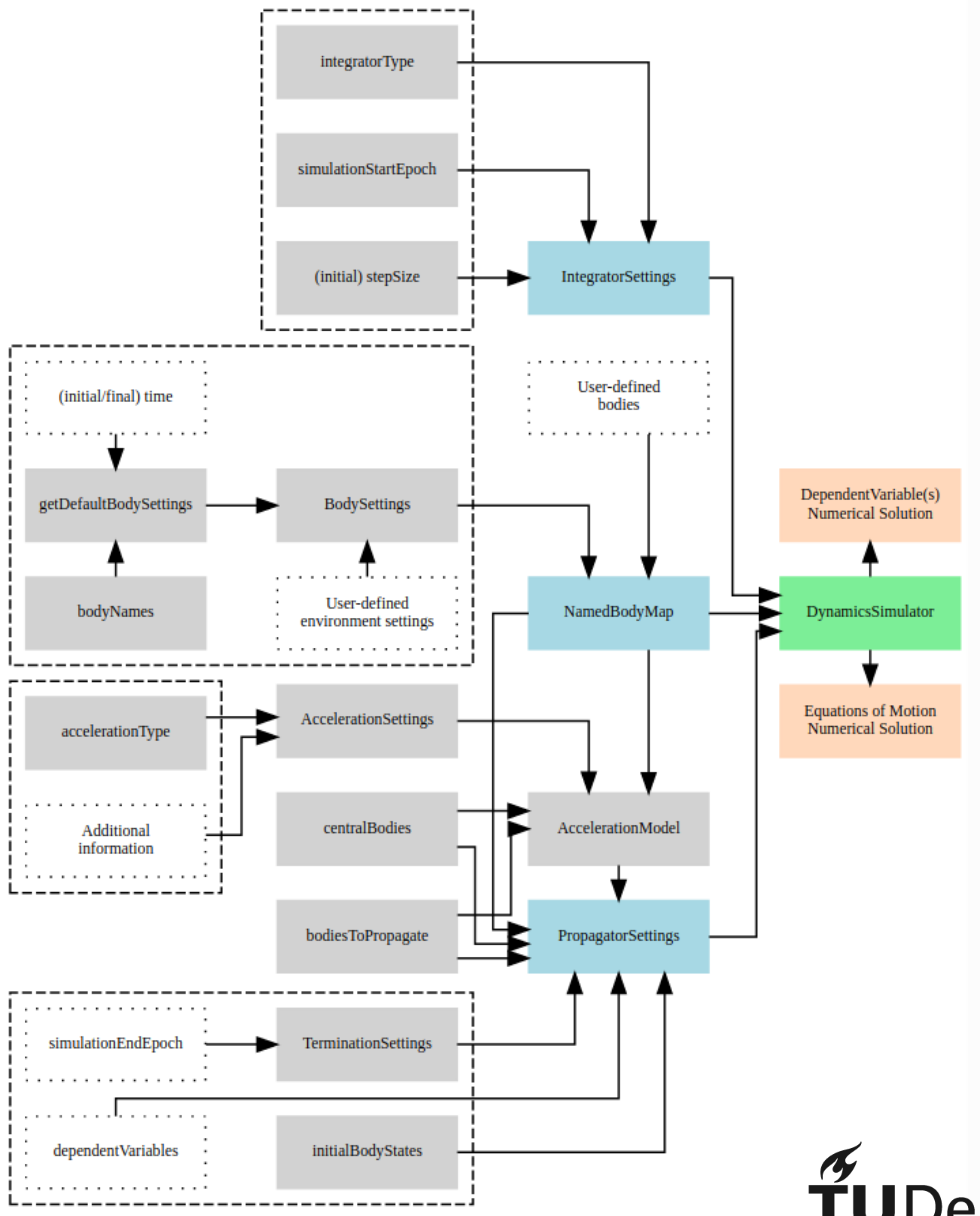
- `fixedStepSize`

  `TimeType` that defines the fixed step-size to be used either by the `euler` or the `rungeKutta4` numerical integrator.



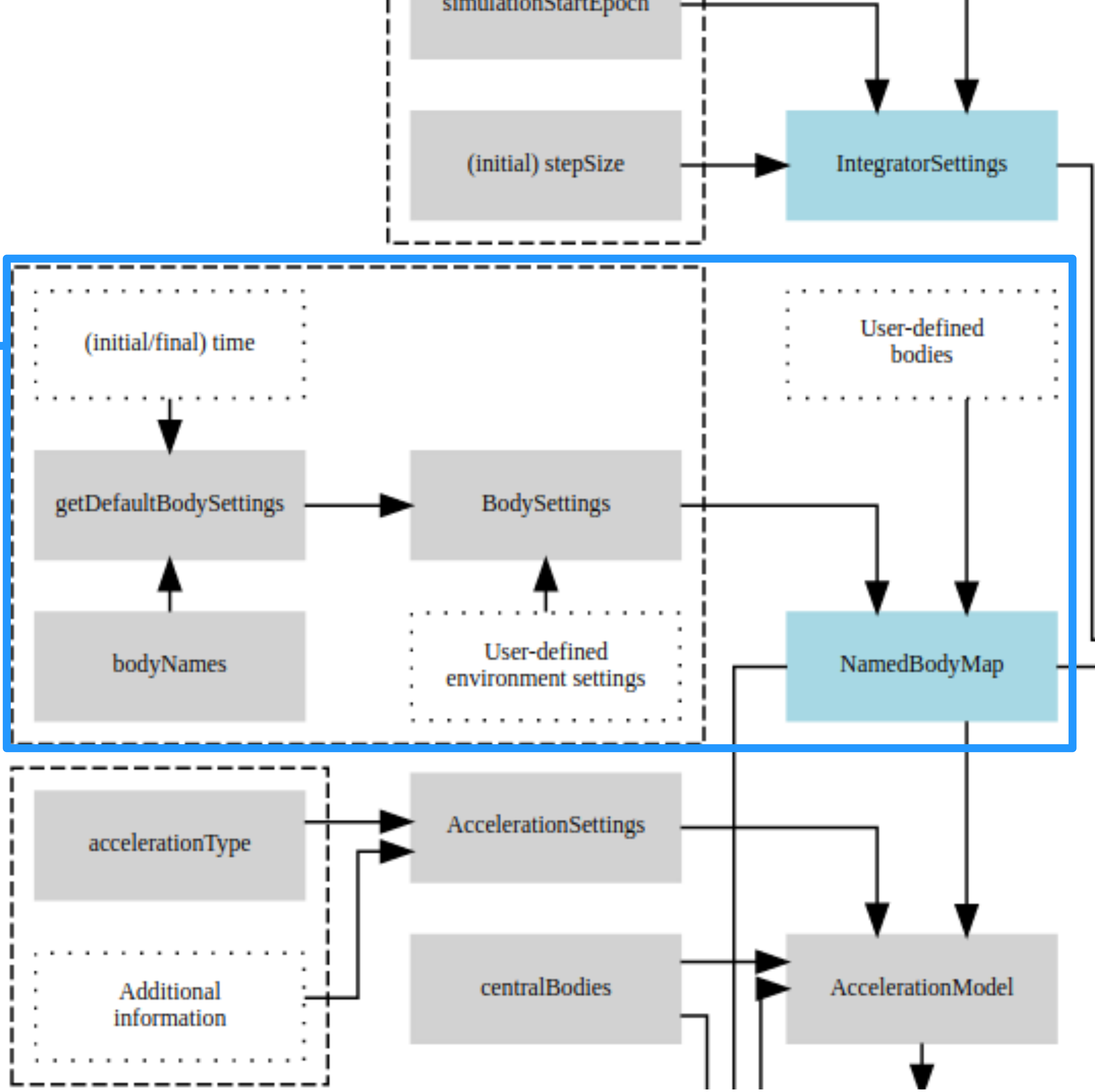
TUDAT: usage

- Input settings for numerical propagation
  - environment
  - accelerations
  - integrator
  - propagator



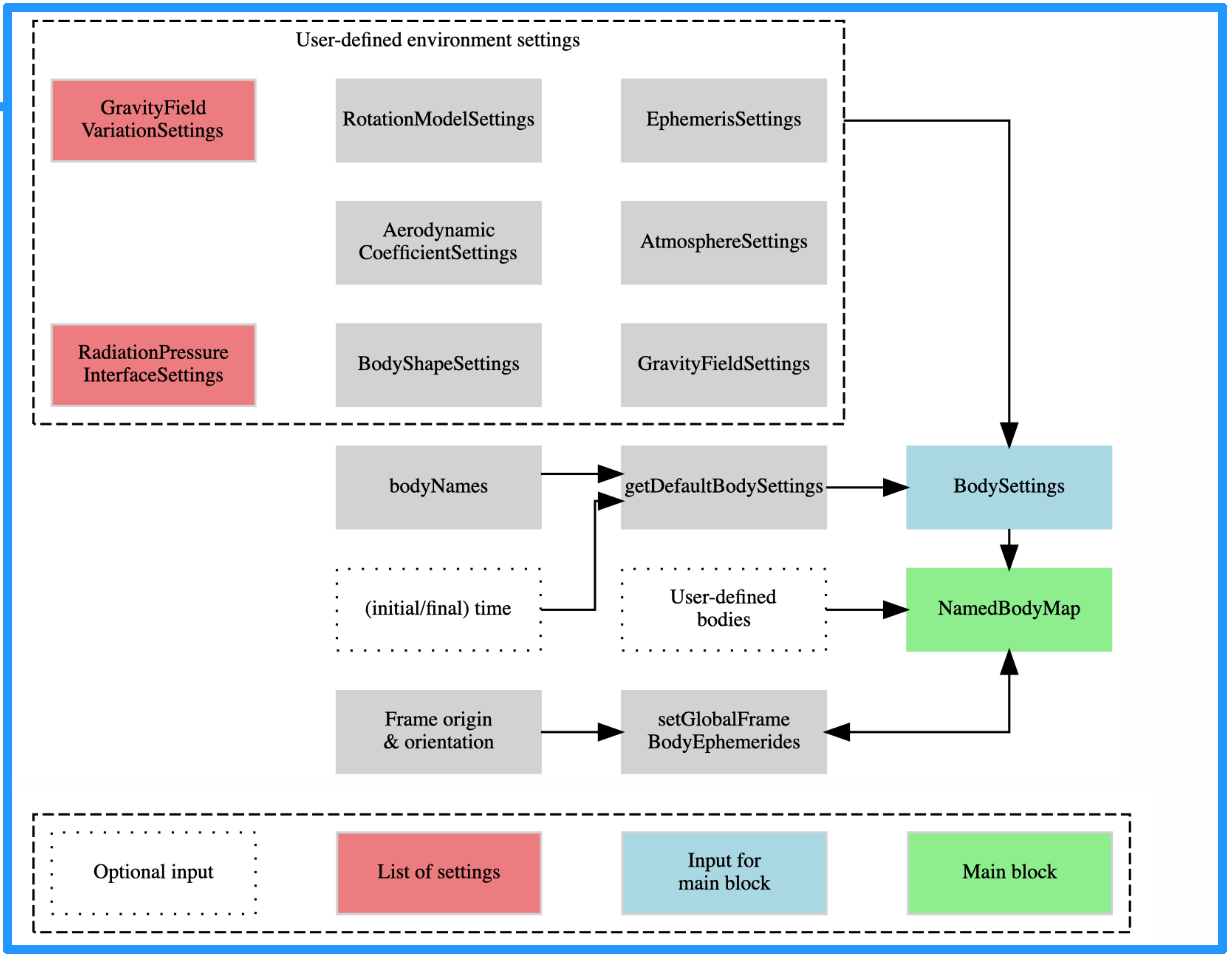
TUDAT: usage

- Input settings for numerical propagation
  - environment
  - accelerations
  - integrator
  - propagator



TUDAT: usage

- Input settings for numerical propagation
  - environment
  - accelerations
  - integrator
  - propagator



TUDAT: usage

TU Delft Astrodynamic Toolbox

Search docs

CONTENTS

Start Page
Getting Started
Installation Guide
⊟ Tutorials and Documentation
Application Tutorials
⊟ Tudat Libraries
⊟ 1. Environment Set-up
1.1. Setting up the Environment
1.2. Available Settings for the Environment Models
1.3. The Environment During Propagation
1.4. Tabulated Atmosphere Settings
1.5. Aerodynamic Coefficients
2. Setting up State Derivative Models
3. Simulator Set-Up
4. Estimation Set-Up
5. Basic Astrodynamics Tools
6. Basic Mathematics Tools
7. Other Libraries
Tudat Basics
JSON Interface
MATLAB Interface
Doxygen
Troubleshooting
FAQ
Developer Guide

1.1.1. Creating the environment from `BodySettings`

Manually creating all objects defining the full environment is possible (see below), but not recommended. In particular, various environment models are interdependent and these dependencies must be fully and consistently defined for the code to function properly. To this end, we provide a `BodySettings` object, which is the easiest way to create a set of (natural or artificial) bodies in Tudat.

class **`BodySettings`**

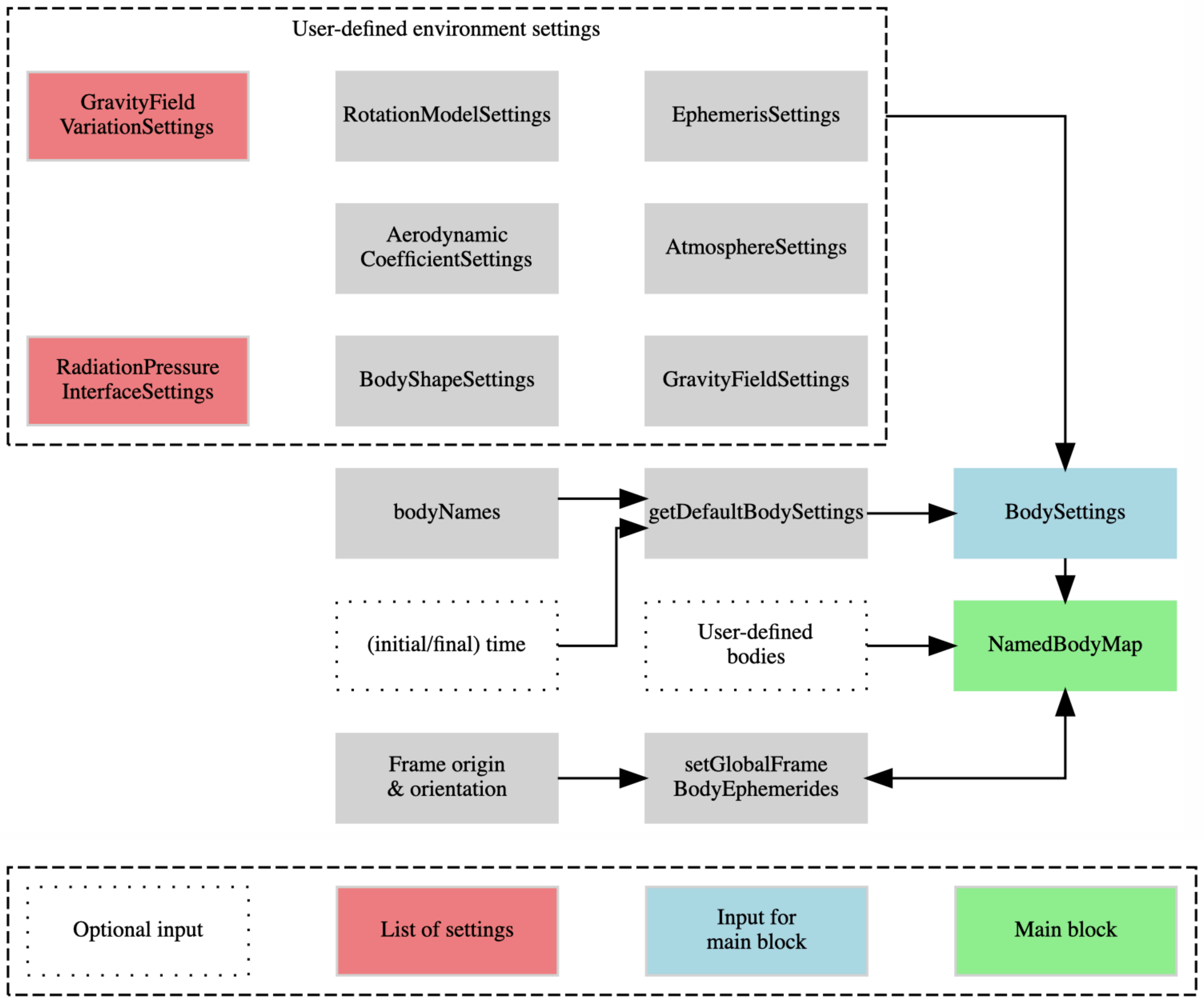
Class in which the general properties of each environment model can be set (see above for the list of the available types of environment models). We note that for `Body` objects that represent vehicles, the manual creation is typically used, as the vehicle conditions may depend on the celestial bodies, but not vice versa.

In many cases, default properties of (celestial) bodies may be used by calling the `getDefaultBodySettings` function, so that the user does not need to define all required properties line-by-line. At present, the following default settings are used (none if not in list):

- **Ephemeris:** Tabulated ephemeris created from Spice (valid in the interval that is specified by the input time-arguments to getDefaultBodySettings).
- **Gravity field models:** Point mass gravity field models, with gravitational parameter from Spice (if available). Exceptions are the Earth and Moon, for which the EGM96 and GGLP spherical harmonic gravity fields are loaded, respectively.
- **Rotation model:** For a given body (if available) the Spice rotation model, with ECLIPJ2000 as base frame, and for a body AAA frame IAU_AAA as target frame (the standard body-fixed frame for each body in Spice).
- **Atmosphere model:** 1976 US Standard Atmosphere for Earth (using pregenerated tables). For other bodies, no default shape model is given.
- **Shape model:** Spherical model with mean radius obtained from Spice (if avaiable).

The default settings for a body are loaded as follows:

```
std::vector< std::string > bodyNames;
bodyNames.push_back( "Earth" );
bodyNames.push_back( "Sun" );
bodyNames.push_back( "Moon" );
bodyNames.push_back( "Mars" );
double initialEphemerisTime = 1.0E7;
double finalEphemerisTime = 2.0E7;
double buffer = 5000.0;
```



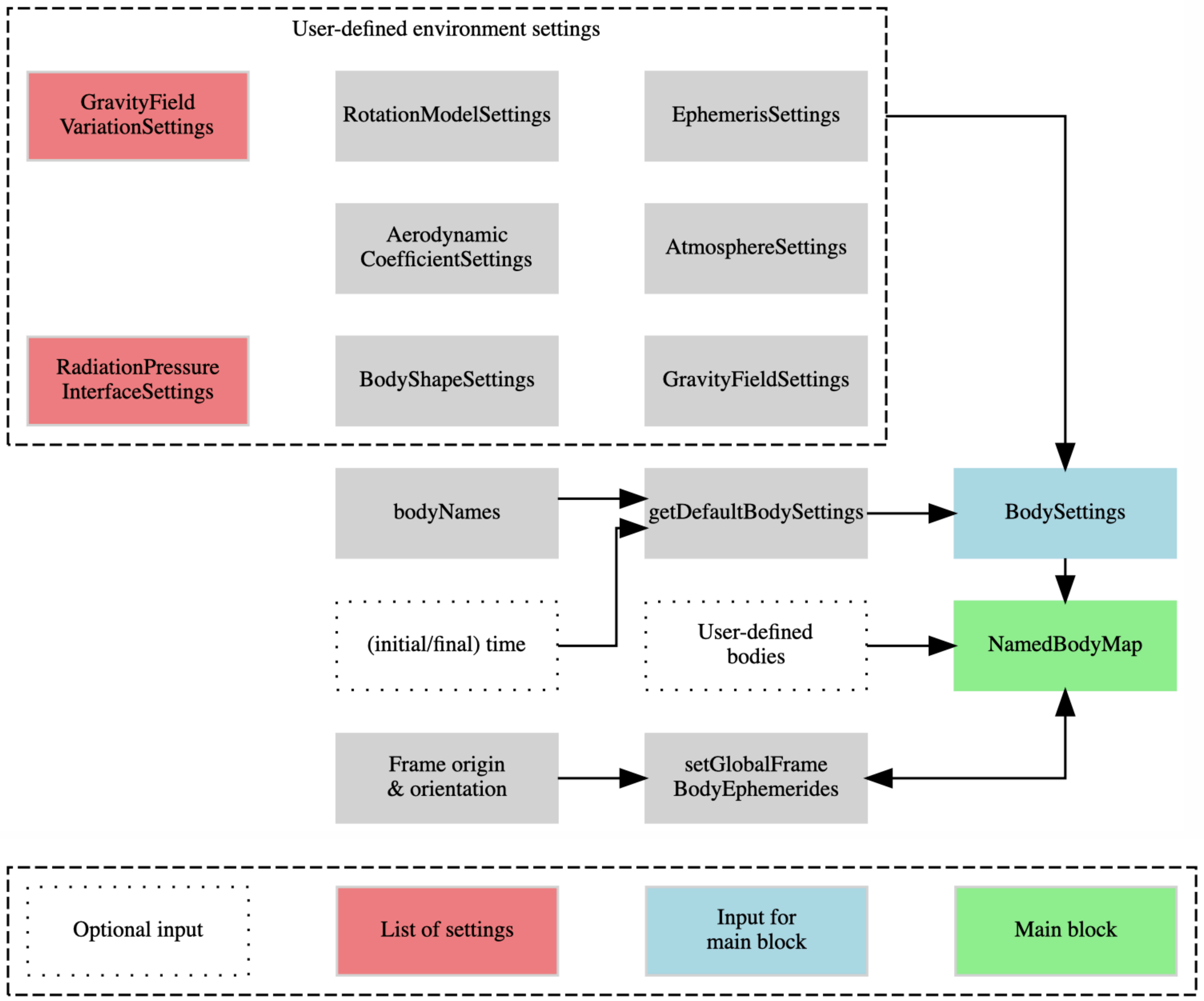
TU Delft

TUDAT: usage

- Each type of environment model can be defined using various representations

- Atmosphere model (example)
  - exponential (conceptual model)
  - tabuated (user-define profile of atmosphere)
  - NRLMSISE-00 (detailed time/location-dependent model)

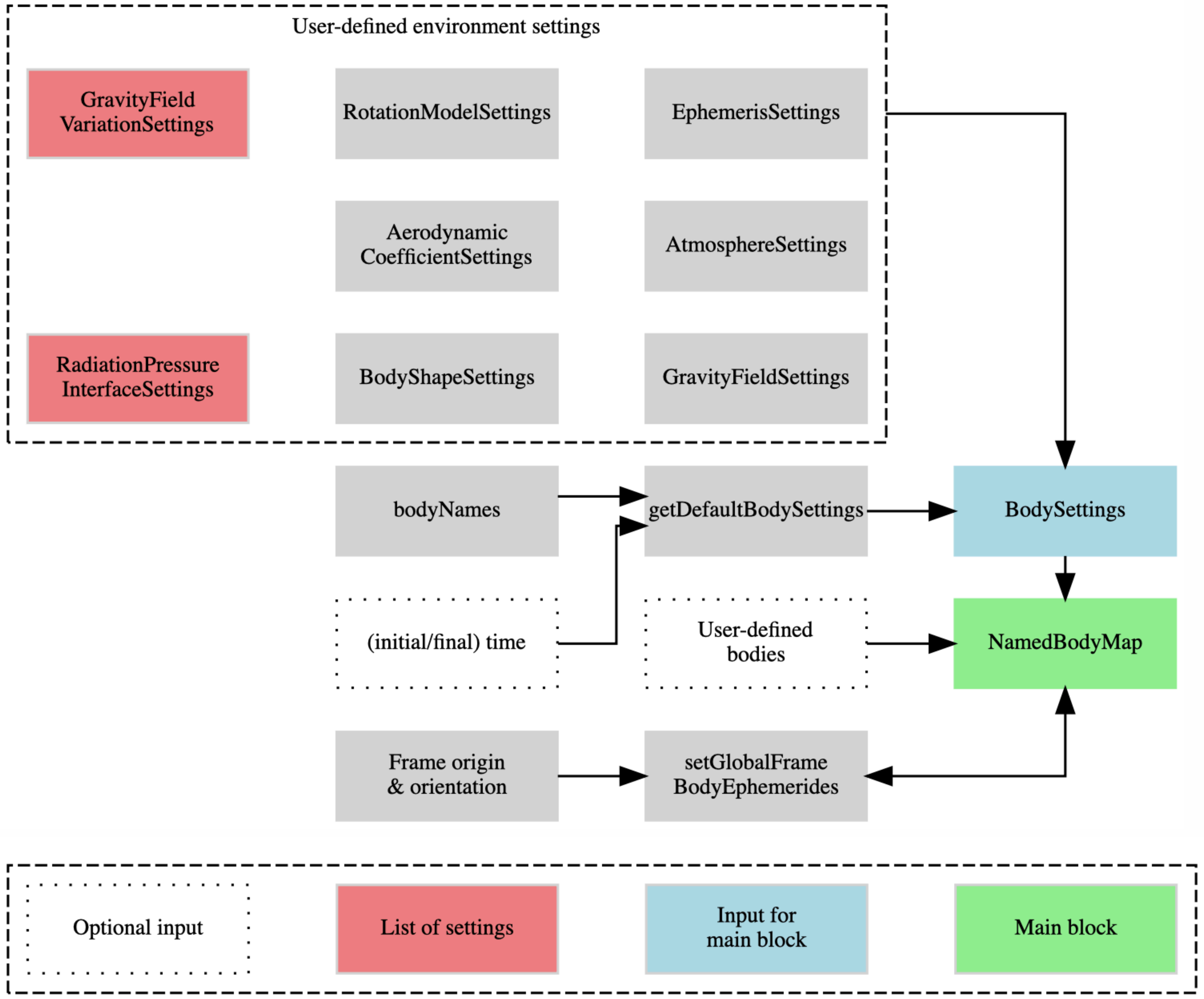
- Environment models are (mostly) independent
  - changes in one model are not used to update the parameters of another model
  - a lot of freedom, but also the possibility to create highly unrealistic combinations!



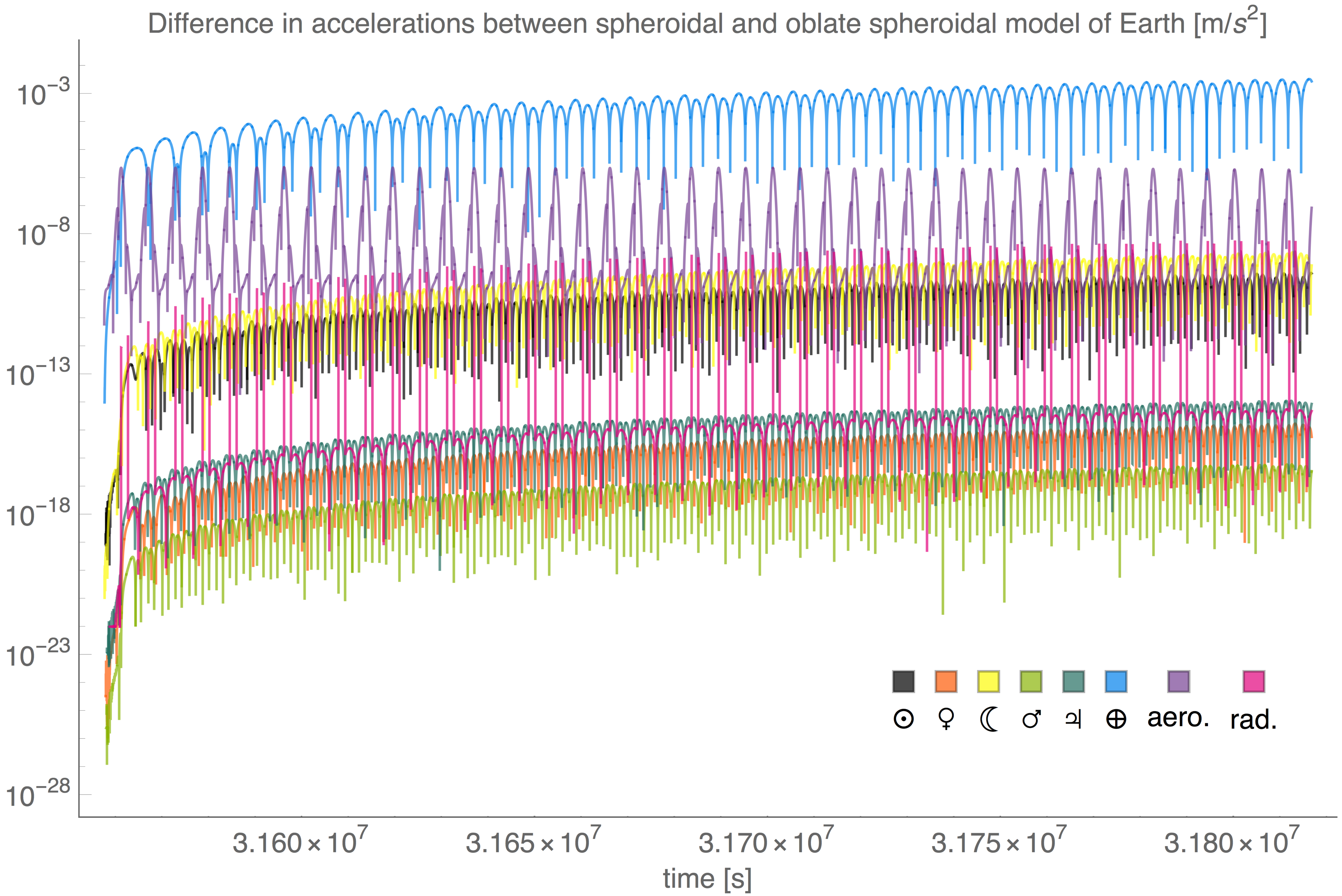
TUDAT: usage

- Environment models are (mostly) independent
  - changes in one model are not used to update the parameters of another model
  - a lot of freedom, but also the possibility to create highly unrealistic combinations!

- eg. change in gravity field (and moments of inertia) does not change rotation model



TUDAT: usage



Difference in accelerations between spheroidal and oblate spheroidal model of Earth [m/s²]
10⁻³
10⁻⁸
10⁻¹³
10⁻¹⁸
10⁻²³
10⁻²⁸
3.160 × 10⁷
3.165 × 10⁷
3.170 × 10⁷
3.175 × 10⁷
3.180 × 10⁷
time [s]
⊙ ♀ ☾ ♂ ♃ ⊕ aero. rad.

TUDAT: usage

- Acceleration models: settings defined by user
  - type of acceleration (and additional information if needed)
  - body undergoing acceleration
  - body exerting acceleration

- Acceleration model (example):
  - spherical harmonic gravity from Earth (maximum degree/order: 7/0)
  - point mass Moon and Sun perturbation
  - Earth aerodynamics
  - Sun radiation pressure

```
"accelerations": {
  "satellite": {
    "Earth": [
      {
        "maximumDegree": 7,
        "maximumOrder": 0,
        "type": "sphericalHarmonicGravity"
      },
      {
        "type": "aerodynamic"
      }
    ],
    "Sun": [
      {
        "type": "pointMassGravity"
      },
      {
        "type": "cannonBallRadiationPressure"
      }
    ],
    "Moon": [
      {
        "type": "pointMassGravity"
      }
    ]
  }
},
```

TUDAT: usage

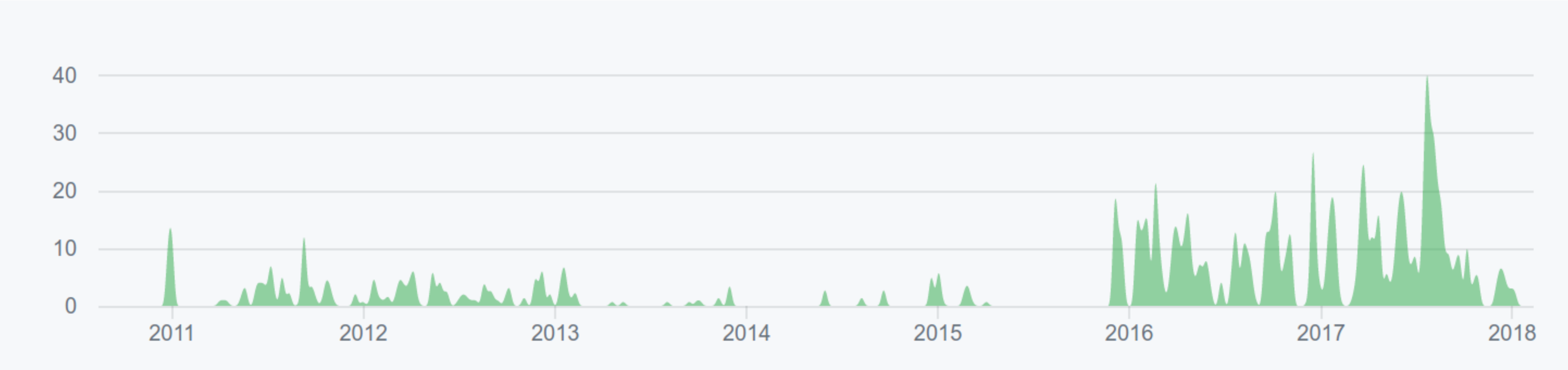
- Acceleration model (example):
  - spherical harmonic gravity from Earth (maximum degree/order: 7/0)
  - point mass Moon and Sun perturbation
  - Earth aerodynamics
  - Sun radiation pressure

- A hierarchical approach is also possible
  - Orbiter propagated wrt. Moon
  - Moon propagated wrt. Earth
  - Earth propagated wrt. Sun

```
"accelerations": {
  "satellite": {
    "Earth": [
      {
        "maximumDegree": 7,
        "maximumOrder": 0,
        "type": "sphericalHarmonicGravity"
      },
      {
        "type": "aerodynamic"
      }
    ],
    "Sun": [
      {
        "type": "pointMassGravity"
      },
      {
        "type": "cannonBallRadiationPressure"
      }
    ],
    "Moon": [
      {
        "type": "pointMassGravity"
      }
    ]
  }
},
```

TUDAT: interfaces – C++

- Traditionally, the project has been fully C++
  - simulation settings written directly into customized (main) function
- Original setup: no end-to-end model. Users put everything together manually
  - around 2015, we realized that the learning curve was too steep
  - a 'Simulation Setup' layer was created, which makes much of the details of the code hidden from users



TUDAT: interfaces – JSON

- Recent addition: JSON interface
  - Input to the program is a JSON file
  - File contains settings of the simulation, defaults may be used if nothing provided


```json
{
  "initialEpoch": 0,
  "finalEpoch": 3600,
  "spice": {
    "preloadEphemeris": false,
    "useStandardKernels": true
  },
  "bodies": {
    "Sun": {
      "useDefaultSettings": true
    },
    "Earth": {
      "useDefaultSettings": true
    },
    "Moon": {
      "useDefaultSettings": true
    },
    "vehicle": {
      "initialState": {
        "x": 8.0E+6,
        "vy": 7500,
        "type": "cartesian"
      },
      "mass": 5000
    }
  },
  "propagators": [
    {
      "centralBodies": [
        "Earth"
      ],
      "accelerations": {
        "vehicle": {
          "Earth": [
            {
              "type": "pointMassGravity"
            }
          ],
          "Sun": [
            {
              "type": "pointMassGravity"
            }
          ],
```

TUDAT: interfaces – JSON

- Recent addition: JSON interface
  - Input to the program is a JSON file
  - File contains settings of the simulation, defaults may be used if nothing provided

- Sufficient for many applications, but does not allow access to full functionality

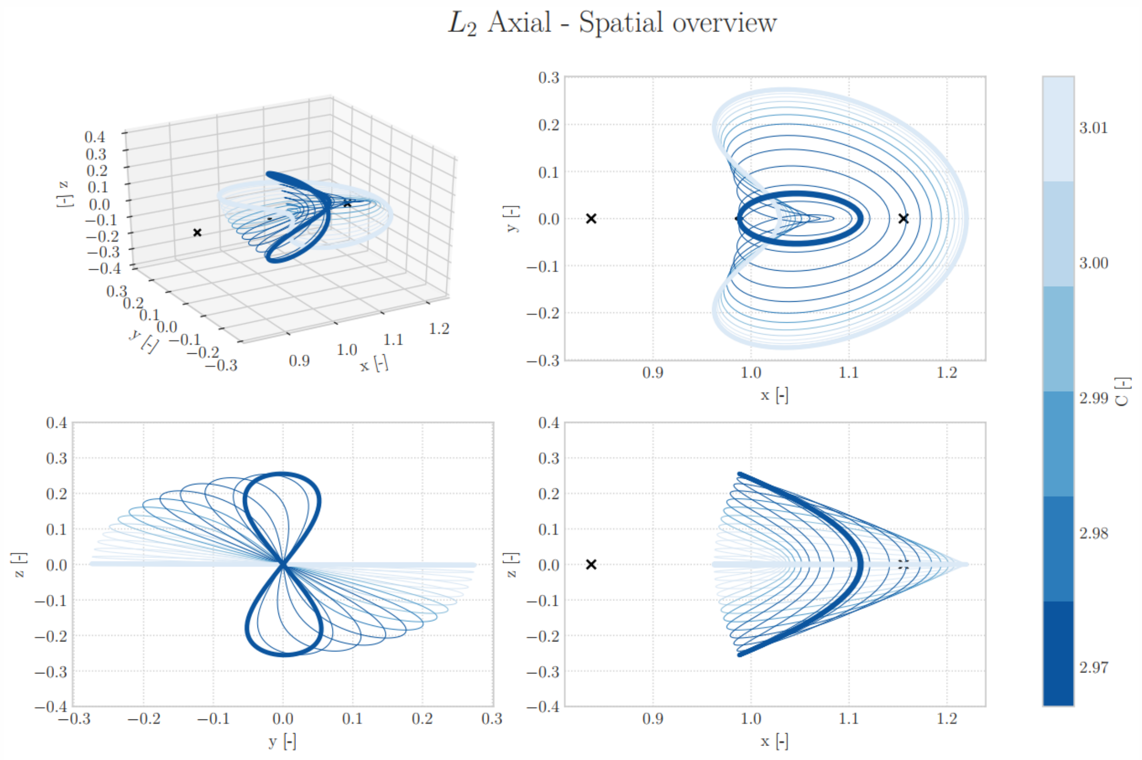
- Developed in context of ESA's SOCIS project (socis.esa.int)

```
},
"propagators": [
  {
    "centralBodies": [
      "Earth"
    ],
    "accelerations": {
      "vehicle": {
        "Earth": [
          {
            "type": "pointMassGravity"
          }
        ],
        "Sun": [
          {
            "type": "pointMassGravity"
          }
        ],
        "Moon": [
          {
            "type": "pointMassGravity"
          }
        ],
        "vehicle": [
          {
```

TUDAT: example projects

- Interplanetary trajectory design and optimization
- Re-entry predictions of space debris
- Space plane ascent optimization
- Re-entry vehicle shape optimization
- Analysis of interplanetary laser ranging
- Analysis of orbital dynamics of Mab
- Orbit design using manifolds and periodic orbits
- Test of Einstein equivalence principle using RadioAstron
- Multidisciplinary launcher optimization (TUD/NLR)
- Orbit determination of LRO satellite (DLR/TUD)
- Dynamics of Galilean moons (JIVE/TUD/JPL)
- Analysis of VLBI tracking of JUICE (TUD/JIVE)

credit: NASA



Langemeier (2018)

TUDAT: example publications

On the contribution of PRIDE-JUICE to Jovian system ephemerides, D. Dirkx, L.I. Gurvits, V. Lainey, G. Lari, A. Milani, G. Cimò, T.M. Bocanegra-Bahamon, P.N.A.M. Visser. Planetary and Space Science, Volume 147, 2017, Pages 14-27, ISSN 0032-0633.

Dynamical modelling of the Galilean moons for the JUICE mission, D. Dirkx, V. Lainey, L.I. Gurvits, P.N.A.M. Visser. Planetary and Space Science, Volume 134, 2016, Pages 82-95, ISSN 0032-0633.

Demonstration of orbit determination for the Lunar Reconnaissance Orbiter using one-way laser ranging data, S. Bauer, H. Hussmann, J. Oberst, D. Dirkx, D. Mao, G.A. Neumann, E. Mazarico, M.H. Torrence, J.F. McGarry, D.E. Smith, M.T. Zuber. Planetary and Space Science, Volume 129, 2016, Pages 32-46, ISSN 0032-0633.

Comparative analysis of one- and two-way planetary laser ranging concepts, D. Dirkx, R. Noomen, P.N.A.M. Visser, S. Bauer, L.L.A. Vermeersen. Planetary and Space Science, Volume 117, 2015, Pages 159-176, ISSN 0032-0633.

TUDAT: example publications

Phobos laser ranging: Numerical Geodesy experiments for Martian system science, D. Dirkx, L.L.A. Vermeersen, R. Noomen, P.N.A.M. Visser. Planetary and Space Science, Volume 99, 2014, Pages 84-102, ISSN 0032-0633.

Mab's orbital motion explained, K. Kumar, I. de Pater, M.R. Showalter. Icarus, Volume 254, 2015, Pages 102-121, ISSN 0019-1035.

Statistical Impact Prediction of Decaying Objects, A. L. A. B. Ronse and E. Mooij. Journal of Spacecraft and Rockets, Vol. 51, No. 6 (2014), pp. 1797-1810.

Node Control and Numerical Optimization of Aerogravity-Assist Trajectories, Jaimy Hess and Erwin Mooij. AIAA Atmospheric Flight Mechanics Conference, AIAA SciTech Forum, (AIAA 2017-0471).

Reachability Analysis to Design Zero-Wait Entry Guidance, Alejandro Gonzalez-Puerta, Erwin Mooij, and Celia Yabar Valles. 2018 AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum, (AIAA 2018-1316).

TUDAT: the open-source astrodynamics toolbox of Delft University of Technology

Kevin Cowan & Dominic Dirkx
Astrodynamics and Space Missions
Faculty of Aerospace Engineering — Delft University of Technology

ICATT 2018
7th International Conference on Astrodynamics Tools and Techniques
DLR Oberpfaffenhofen, Germany
6–9 November 2018

TUDelft