

Studying tidal effects in planetary systems with Posidonius. A N-body simulator written in Rust.

Sergi Blanco-Cuaresma^{1,2} Emeline Bolmont³

¹ Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138, USA

² Observatoire de Genève, Université de Genève, CH-1290 Versoix, Switzerland

³ Laboratoire AIM Paris-Saclay, CEA/DRF-CNRS-Univ. Paris Diderot - IRFU/SAp, Centre de Saclay, F- 91191 Gif-sur-Yvette Cedex, France

Abstract

Planetary systems with several planets in compact orbital configurations such as TRAPPIST-1 are surely affected by tidal effects. Its study provides us with important insight about its evolution. We developed a second generation of a N-body code based on the tidal model used in Mercury-T, re-implementing and improving its functionalities using Rust as programming language (including a Python interface for easy use) and the WHFAST integrator. The new open source code ensures memory safety, reproducibility of numerical N-body experiments, it improves the spin integration compared to Mercury-T and allows to take into account a new prescription for the dissipation of tidal inertial waves in the convective envelope of stars. Posidonius is also suitable for binary system simulations with evolving stars.

1 Introduction

Detection of planetary systems around ultracool dwarfs (i.e., stars with effective temperatures lower than 2 700 K) are expected to increase in the coming years (Gillon *et al.*, 2016) thanks to on-going efforts such as the TRAPPIST survey (Jehin *et al.*, 2011) and future missions such as the ambitious project SPECULOOS (Search for habitable Planets EClipsing ULtra-cOOl Stars) that will be installed at ESO's Paranal Observatory. These systems can have several planets in compact orbital configurations, and they can be in or close to mean motion resonances as already observed in TRAPPIST-1 (Gillon *et al.*, 2017). Planets arranged in this kind of set-up are surely affected by tidal effects, and the use of N-body simulation is necessary to account for the complicated dynamics of the system.

One of the most used open source tools by the community to simulate exoplanetary systems is Mercury-T¹ (Bolmont *et al.*, 2015), which is based on the N-body simulator Mercury (Chambers, 1999) with an added extension to account for tidal effects, general relativity, rotational flattening and the evolution of the central body. The original Mercury is a customizable general-purpose N-body code written in Fortran 77, although there are forks re-written in Fortran 90² that were actually used to create Mercury-T.

Fortran (derived from Formula Translation) is an imperative programming language created by IBM in the 1950s and it is especially suited to scientific computing. The language has gone through several updates since its birth (where version names correlate with the year the update was released), adding support for structured programming (Fortran 77), modular and generic programming (Fortran 90), high

performance capabilities (Fortran 95), object-oriented programming (Fortran 2003) and concurrent programming (Fortran 2008). Nevertheless, most of the scientific community keeps using Fortran 77/90 and it is common than available codes only work properly with specific compilers (e.g., the proprietary Intel Fortran/ift compiler, the deprecated g77 or the modern GNU Fortran/gfortran). Fortran (like the C language) provides to the developer a high level of control for managing the computer memory which can lead to errors and bugs. For instance, Mercury suffered a problem due to an array not being properly initialized, hence its elements may access random values which will affect the simulation results (de Souza Torres & Anderson, 2008) and the scientific conclusions.

Certain Fortran compilers offer validations such as checking for initialized variables or checking out-of-bound access for arrays, which can be enabled/disabled depending on the flags used to compile the code. These verifications, although useful, do not cover all possible problems derived from memory management and, in any case, they are not enforced by the language. Given the wide use of the language in the scientific community, the lack of software engineering training in certain disciplines such as astrophysics and the lack of memory management controls in the language, it can be convenient to look for alternative technologies. A possibility could be to use garbage collected languages such as Python or Java, which are already extensively used by the astronomy community (especially Python). But these solutions have an impact on execution times, and codes such as N-body simulators are computationally intense and they require fast solutions to be useful. It is for these specific cases that the new programming language Rust can be well fitted as exposed in Blanco-Cuaresma & Bolmont (2017).

¹<http://www.emelinebolmont.com/>

²<https://code.google.com/archive/p/mercury-90/>

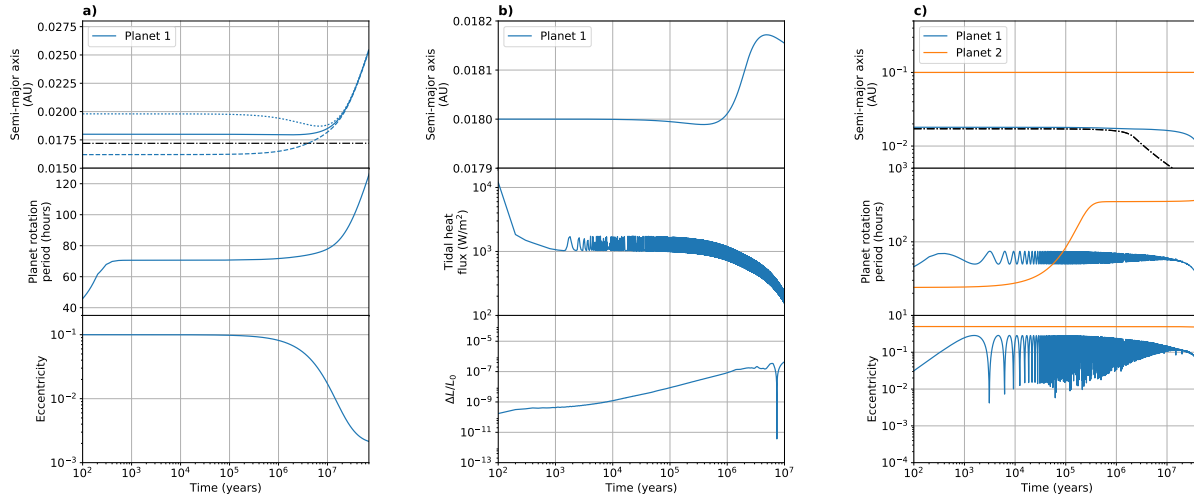


Figure 1: a) Case 3 from Bolmont *et al.* (2015) where the dashed blue lines correspond to the periastron and apostron and the black dashed line is the co-rotation radius in the upper figure. b) Case 4 from Bolmont *et al.* (2015) where the conservation of angular momentum is shown in the lower figure. c) Case exposed in Sect. 5.2.2 from Bolmont (2013) (Fig. 5.27) where the black dashed line is the co-rotation radius in the upper figure.

We developed Posidonius^{3 4} and distributed it under an open source license⁵, a second generation of N-body code based on the tidal model used in Mercury-T (Bolmont *et al.*, 2015), re-implementing and improving its functionalities using Rust as programming language and the WHFAST integrator (Rein & Tamayo, 2015). The new code ensures memory safety, reproducibility of numerical N-body experiments, it improves the spin integration compared to Mercury-T and allows to take into account a new prescription for the dissipation of tidal inertial waves in the convective envelope of stars (Bolmont & Mathis, 2016; Gallet *et al.*, 2017; Bolmont *et al.*, 2017).

2 The design

Posidonius is composed by a N-body simulator written in Rust and a Python package used to define the initial condition of the scientific cases the user wants to simulate. Rust ensures that the simulation is free from typical memory management errors but its learning curve can be steep (Blanco-Cuaresma & Bolmont, 2017), hence with this separation, we provide a user-friendly scripting interface and do not force the user to learn Rust details unless a more advanced use is really necessary (e.g., implementing new physical effects, developing a new integrator). Once the case is defined, the python script generates a JSON format file describing the initial scenario (actually, with a bit of effort, that JSON file could be generated manually or using any other programming language) and Posidonius can read this file to initiate the simulation.

³<http://www.blancocuaresma.com/s/>

⁴Posidonius (c. 135 BCE - c. 51 BCE), was a Greek Stoic philosopher, politician, astronomer, geographer, historian and teacher native to Apamea (Syria). In Hispania, on the Atlantic coast near Cadiz, Posidonius could observe tides much higher than in his native Mediterranean. He wrote that daily tides are related to the Moon's orbit, while tidal heights vary with the cycles of the Moon.

⁵GNU Affero General Public License

An important aspect of N-body simulations (and more in general, for any scientific study) is ensuring that your computations can be reproduced and reverified (Rein & Tamayo, 2017). Posidonius stores the history of the simulation together with frequent recovery snapshots in binary format. The user can stop and resume any simulation at any point or repeat again the same calculations by using an older recovery snapshot. But, more importantly, any study can be completely reproduced by indicating what Posidonius version was used and providing the original JSON files with the initial conditions (or the Python script that generates it).

From a technical point of view, we have taken some decisions to optimize the execution time. We do not use certain Rust capabilities such as vectors (i.e., re-sizable arrays allocated in the heap), all the bodies of a simulation are stored in arrays allocated in the stack for faster access. The array has a fixed size for all the simulations (defined by a constant in the code), it is possible to create cases with a lower number of bodies but not higher (although that can be overcome by changing the constant value and recompiling Posidonius). We do not take full advantage of the possibilities of Rust for memory management due to this decision, but we are open to study and incorporate alternative technical designs that other experts may come up with.

To compute the evolution of positions and velocities from gravity forces, Posidonius implements the symplectic integrator WHFAST (Rein & Tamayo, 2015) without correctors (since tidal forces are velocity dependent). This algorithm has been proved to be fast and accurate for long-term orbit integrations of planetary systems. The code can execute integrations using Jacobi, Democratic-Heliocentric or WHDS coordinates (Hernandez & Dehnen, 2017) as provided in the REBOUND N-body code (Rein & Liu, 2012). In a consistent way, the spin evolution is computed for all the bodies by using a midpoint integrator (Bulirsch, 2013).

Tidal forces, rotational-flattening effects and general relativity corrections are implemented following the model de-

scribed by Bolmont *et al.* (2015). In the case of the general relativity, we also included the prescriptions from Anderson *et al.* (1975) and Newhall *et al.* (1983) (the latter considers effects from all bodies in the system, making Posidonius ready for binary stars too) following the REBOUND implementation. Additionally, evolution models for FGKM stars and gaseous planets have been included. Some of these models were already present in Mercury-T: brown dwarfs models of Leconte *et al.* (2011), 0.1 and 1 M_{\odot} models of Baraffe *et al.* (1998) Jupiter model of Leconte & Chabrier (2013). In Posidonius, we added the evolutionary models used to compute tidal evolution in Bolmont & Mathis (2016). These models are STAREVOL models for FGKM stars (Siess *et al.*, 2000). We also implemented stellar evolution models, coming from a more recent version of the STAREVOL code (Gallet *et al.*, 2017). All these evolution models give the evolution of stellar quantities such as the radius of the star with time. The newest models (Gallet *et al.*, 2017) allow to take into account the evolution of the frequency-averaged dissipation in the convective envelope of FGKM stars. For these last models, the user must keep in mind that the physical validity of the dissipation formulation is for coplanar, circular orbits, which is never the case if there are more than one planet in the system.

3 The assessment

The best approach to assess the quality of Posidonius is to repeat simulations that were previously executed, peer-reviewed and published based on Mercury-T. We have selected the case three and four defined in Table 4 from Bolmont *et al.* (2015) (non-evolving and evolving brown dwarf with one planet and considering tidal effects), case six to six triple prime listed in Table 5 from Bolmont *et al.* (2015) (non-evolving brown dwarf with two planets and considering rotational-fattening using time steps of 0.08, 0.05, 0.01, 0.001 days) and the case investigated in Sect. 5.2.2 from Bolmont (2013) (evolving brown dwarf with two planets considering tidal effects and general relativity corrections). To facilitate the reproducibility of any of these tests, the initial condition for each of them are included in the Posidonius source code.

The resulting semi-major axis evolution for cases three and four are shown in Fig. 1a and Fig. 1b and they match the results presented in Fig. 3 and Fig. 6 from Bolmont *et al.* (2015). We observed that the conservation of angular momentum for case four (1b) is one order of magnitude better than Mercury-T. On the problem of Mercury-T having a numerical shift of the rotation period of the planets when only the rotational flattening effect was on (cases six to six triple prime), Posidonius has a much better behavior (five orders of magnitude better when Fig. 2 is compared to Fig. 7 from Bolmont *et al.* (2015)). Finally, the resulting evolution for the last case with multiple planets is shown in Fig. 1 and it is also very similar to Fig. 5.27 from Bolmont (2013). Although we do not include the results in this work due to space limitations, it is worth mentioning that we successfully reproduced the results of Fig. 5.28 and Fig. 5.30 from Bolmont (2013), which explore the effects of the stellar dissipation factor and resonances.

In parallel, during the execution of these tests, we measured and compared Posidonius speed and we observed that it is six times faster than Mercury-T in average. This is a sig-

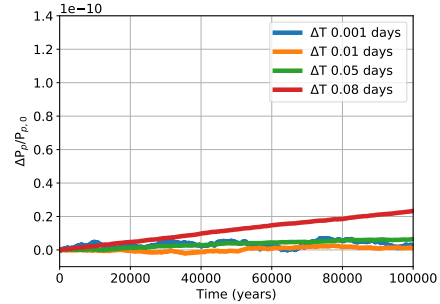


Figure 2: Conservation of planet rotation period (at 10^{-10} scale) for different time steps, equivalent to Fig. 7 (case 6) from Bolmont *et al.* (2015). Difference in rotation period constantly increases for time steps equal or greater than 0.05, for lower values the difference follows a random walk centered around zero when considering longer time spans.

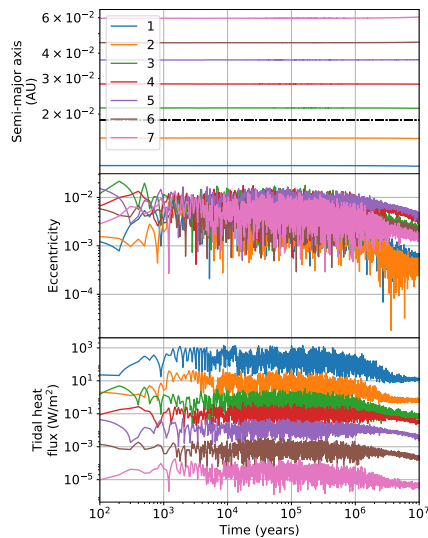


Figure 3: TRAPPIST-1 simulation with parameters from Table 1. The black dashed line in the upper figure corresponds to the co-rotation radius.

nificant improvement that is not strictly linked to the chosen programming language, the new code was written with different design patterns not followed in Mercury-T.

4 The scientific case: TRAPPIST-1

We chose the TRAPPIST-1 system as an example of a useful scientific case to be studied with Posidonius. We run a simulation with a duration of 10^7 years with a time step of 0.08 days, where all the effects were enabled (i.e. tides, rotational flattening and general relativity corrections). The star (just above the brown dwarf limit for this system) was set to have a mass of 0.08 M_{\odot} , a radius of 0.117 R_{\odot} , a rotation period of 3.3 hours, a low dissipation factor (1% of a typical brown dwarf, Bolmont *et al.* 2011), and radius of gyration and love number similar to brown dwarfs. No stellar evolution was

Table 1: TRAPPIST-1 planetary initial conditions used in the simulation shown in Fig. 3.

Planet	Radius (R_{\oplus})	Mass (M_{\odot})	Semi-major axis (AU)	Inclination (deg)	Mean anomaly (deg)
1	1.086	2.5843686×10^{-6}	0.01111	0.35	323.732652895
2	1.056	4.1957984×10^{-6}	0.01521	0.33	96.4925777097
3	0.772	1.2465778×10^{-6}	0.02144	0.25	111.770368348
4	0.918	1.8850689×10^{-6}	0.02817	0.14	165.724187804
5	1.045	2.0674949×10^{-6}	0.03710	0.32	254.117367005
6	1.127	4.0741811×10^{-6}	0.04510	0.29	161.020362506
7	0.755	1.2465778×10^{-6}	0.05960	0.13	134.724813585

enabled because TRAPPIST-1 host star is in an old and stable phase of its life. The parameters for the seven planets in the system are obtained from Gillon *et al.* (2017) and they are listed in Table 1.

The results from the simulation are shown in Fig. 3. Given our initial conditions, the system evolves with 3:2 resonances between planets 3-4, 4-5 and 6-7, a 4:3 resonance between planets 5-6, and a 8:3 resonance between planets 1-2. A more extensive and detailed study of this system will be presented in an upcoming article.

5 Conclusions

Posidonius is a fast and accurate open source N-body simulator for planetary systems that takes into account tidal effects (following the recipes described in Bolmont *et al.* 2015) and adds new updated stellar evolution models. Its high performance and memory safety are ensured thanks to the use of Rust as programming language. To create simulation cases with initial conditions, Posidonius offers a user-friendly scripting interface based on Python. The code was assessed by successfully reproducing a selection of Mercury-T simulations presented in Bolmont (2013) and Bolmont *et al.* (2015). Based on these results, we showed that:

- Posidonius is more than six times faster than Mercury-T;
- It conserves the total angular momentum of the system one order of magnitude better than Mercury-T;
- It conserves the spin to rotational-flattening evolution five orders of magnitude better than Mercury-T.

The simulations can be stopped and resumed at any moment thanks to the automatic creation of recovery snapshots. Additionally, any simulation can be reproduced by just reusing the case definition scripts that contain the initial conditions, the JSON file that the script generates or any intermediary binary snapshot. All the results presented in this work make Posidonius a powerful tool for N-body simulations of planetary systems and, particularly, for the study of tidal effects.

Acknowledgments

This work would not have been possible without the support of Dr. Laurent Eyser (University of Geneva). The authors thank Christophe Cossou for his Python routines to explore

resonances. E.B. acknowledges funding by the European Research Council through ERC grant SPIRE 647383. This research has made use of NASA's Astrophysics Data System. All the software and technology used to build Posidonius were provided by the open source community.

References

- Anderson, J. D., Esposito, P. B., Martin, W., Thornton, C. L., & Muhleman, D. O. 1975, *ApJ*, 200, 221.
- Baraffe, I., Chabrier, G., Allard, F., & Hauschildt, P. H. 1998, *A&A*, 337, 403.
- Blanco-Cuaresma, S. & Bolmont, E. 2017, In *Astroinformatics*, edited by M. Brescia, S. G. Djorgovski, E. D. Feigelson, G. Longo, & S. Cuvuoti, *IAU Symposium*, vol. 325, pp. 341–344.
- Bolmont, E. 2013, *Evolution et habitabilité de systèmes planétaires autour d'étoiles de faible masse et de naines brunes*. Ph.D. thesis. Thèse de doctorat dirigée par Raymond, Sean N. et Selsis, Franck Astrophysique Bordeaux 1 2013.
- Bolmont, E., Gallet, F., Mathis, S., Charbonnel, C., Amard, L., *et al.* 2017, *A&A*, 604, A113.
- Bolmont, E. & Mathis, S. 2016, *Celestial Mechanics and Dynamical Astronomy*, 126, 275.
- Bolmont, E., Raymond, S. N., & Leconte, J. 2011, *A&A*, 535, A94.
- Bolmont, E., Raymond, S. N., Leconte, J., Hersant, F., & Correia, A. C. M. 2015, *A&A*, 583, A116.
- Bulirsch, J. S. 2013, *Introduction to numerical analysis*, vol. 12.
- Chambers, J. E. 1999, *MNRAS*, 304, 793.
- de Souza Torres, K. & Anderson, D. R. 2008, ArXiv e-prints.
- Gallet, F., Bolmont, E., Mathis, S., Charbonnel, C., & Amard, L. 2017, *A&A*, 604, A112.
- Gillon, M., Jehin, E., Lederer, S. M., Delrez, L., de Wit, J., *et al.* 2016, *Nature*, 533, 221.
- Gillon, M., Triaud, A. H. M. J., Demory, B.-O., Jehin, E., Agol, E., *et al.* 2017, *Nature*, 542, 456.
- Hernandez, D. M. & Dehnen, W. 2017, *MNRAS*, 468, 2614.
- Jehin, E., Gillon, M., Queloz, D., Magain, P., Manfroid, J., *et al.* 2011, *The Messenger*, 145, 2.
- Leconte, J. & Chabrier, G. 2013, *Nature Geoscience*, 6, 347.
- Leconte, J., Lai, D., & Chabrier, G. 2011, *A&A*, 528, A41.
- Newhall, X. X., Standish, E. M., & Williams, J. G. 1983, *A&A*, 125, 150.
- Rein, H. & Liu, S.-F. 2012, *A&A*, 537, A128.
- Rein, H. & Tamayo, D. 2015, *MNRAS*, 452, 376.
- Rein, H. & Tamayo, D. 2017, *MNRAS*, 467, 2377.
- Siess, L., Dufour, E., & Forestini, M. 2000, *A&A*, 358, 593.