



# Sorcha: Optimized Solar System Ephemeris Generation

Matthew J. Holman<sup>1</sup> , Pedro H. Bernardinelli<sup>2</sup> , Megan E. Schwamb<sup>3</sup> , Mario Jurić<sup>2</sup> , Drew Oldag<sup>2,4</sup> , Maxine West<sup>2,4</sup> , Kevin J. Napier<sup>1,5,6</sup> , Stephanie R. Merritt<sup>3</sup> , Grigori Fedorets<sup>3,7,8</sup> , Samuel Cornwall<sup>9</sup> , Jacob A. Kurlander<sup>2</sup> , Siegfried Eggl<sup>9,10,11</sup> , Jeremy Kubica<sup>4,12</sup> , Kathleen Kiker<sup>13</sup> , Joseph Murtagh<sup>3</sup> , Shantanu P. Naidu<sup>14</sup> , and Colin Orion Chandler<sup>2,4</sup>

<sup>1</sup> Center for Astrophysics—Harvard & Smithsonian, 60 Garden Street, Cambridge, MA 02138, USA; [mholman@cfa.harvard.edu](mailto:mholman@cfa.harvard.edu)

<sup>2</sup> DiRAC Institute and the Department of Astronomy, University of Washington, 3910 15th Avenue NE., Seattle, WA 98195, USA

<sup>3</sup> Astrophysics Research Centre, School of Mathematics and Physics, Queen's University Belfast, Belfast BT7 1NN, UK

<sup>4</sup> LSST Interdisciplinary Network for Collaboration and Computing Frameworks, 933 N. Cherry Avenue, Tucson, AZ 85721, USA

<sup>5</sup> Michigan Institute for Data and AI in Society, University of Michigan, Ann Arbor, MI 48109, USA

<sup>6</sup> Department of Physics, University of Michigan, Ann Arbor, MI 48109, USA

<sup>7</sup> Finnish Centre for Astronomy with ESO, University of Turku, FI-20014 Turku, Finland

<sup>8</sup> Department of Physics, University of Helsinki, P.O. Box 64, 00014 Helsinki, Finland

<sup>9</sup> Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

<sup>10</sup> Department of Astronomy, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

<sup>11</sup> National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

<sup>12</sup> McWilliams Center for Cosmology, Department of Physics, Carnegie Mellon University, Pittsburgh, PA 15213, USA

<sup>13</sup> Asteroid Institute, 20 Sunnyside Avenue, Suite 427, Mill Valley, CA 94941, USA

<sup>14</sup> Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Received 2025 February 24; revised 2025 May 20; accepted 2025 May 23; published 2025 July 21

## Abstract

*Sorcha* is a solar system survey simulator built for the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST) and future large-scale wide-field surveys. Over the 10 yr survey, the LSST is expected to collect roughly a billion observations of minor planets. The task of a solar system survey simulator is to take a set of input objects (described by orbits and physical properties) and determine what a real or hypothetical survey would have discovered. Existing survey simulators have a computational bottleneck in determining which input objects lie in each survey field, making them infeasible for LSST data scales. *Sorcha* can swiftly, efficiently, and accurately calculate the on-sky positions for sets of millions of input orbits and surveys with millions of visits, identifying which exposures these objects cross, in order for later stages of the software to make detailed estimates of the apparent magnitude and detectability of those input small bodies. In this paper, we provide the full details of the algorithm and software behind *Sorcha*'s ephemeris generator. Like many of *Sorcha*'s components, its ephemeris generator can be easily used for other surveys.

*Unified Astronomy Thesaurus concepts:* [Ephemerides \(464\)](#); [Small Solar System bodies \(1469\)](#); [N-body simulations \(1083\)](#); [Sky surveys \(1464\)](#)

## 1. Introduction

With operations scheduled to start in the second half of 2025, the Vera C. Rubin Observatory's (Rubin) Legacy Survey of Space and Time (LSST; LSST Science Collaboration et al. 2009; Ž. Ivezić et al. 2019; F. B. Bianco et al. 2022) will map  $\sim 18,000$  deg<sup>2</sup> of the sky continuously for 10 yr, with roughly two million total visits expected. The survey will use the LSST Camera (LSSTCam), with a field of view (FOV) of 9.6 deg<sup>2</sup>, a typical expected depth of  $m_r \approx 24.0$  per 30 s visit, and an expected astrometric noise floor of 10 mas. LSST's combination of area, cadence, depth, and astrometric precision will enable it to discover and subsequently track almost 5 to 10 times more objects than are currently known across all solar system populations, from near-Earth objects (NEOs), to main belt asteroids (MBAs), to objects in the distant solar system (R. L. Jones et al. 2009, 2018; LSST Science Collaboration et al. 2009; M. Solonoi et al. 2010; A. Shannon et al. 2015; T. Grav et al. 2016; K. Silsbee & S. Tremaine 2016; P. Vereš & S. R. Chesley 2017;

Ž. Ivezić et al. 2019; G. Fedorets et al. 2020; D. J. Hoover et al. 2022; J. A. Kurlander et al. 2025; J. Murtagh et al. 2025).

A number of excellent survey simulators have been developed for previous solar system surveys (J. M. Petit et al. 2011; S. P. Naidu et al. 2017; K. J. Napier et al. 2021; P. H. Bernardinelli et al. 2022, 2024), but they are not ideally suited to the needs of the LSST; they either do not scale well to the size of LSST or they make approximations that limit their precision in ways that are incompatible with the community's main use cases in the LSST era (M. E. Schwamb et al. 2018). *Sorcha* is a survey simulator designed and built specifically to tackle the challenges of simulating which objects in a model solar system population the LSST would discover (S. R. Merritt et al. 2025). *Sorcha* takes an input solar system population (a set of synthetic objects with orbits, absolute magnitudes, colors, and phase curve parameters) and determines which of the input objects would be detectable in an input survey's pointings. Once there is a set of potential detections per survey pointing, *Sorcha* then replicates the various interlinked observational biases to estimate which of these potential detections can be successfully detected by the survey source detection algorithms and subsequently identified by the survey's moving object



Original content from this work may be used under the terms of the [Creative Commons Attribution 4.0 licence](#). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

discovery algorithms.<sup>15</sup> In this paper, we describe the ephemeris generation algorithm powering *Sorcha*, which quickly and efficiently identifies for each of the exposures which of the input orbits land within a circular approximation of the survey’s camera’s footprint and then calculates their precise on-sky locations at the midpoint of the observation.

As Rubin is expected to discover objects across all solar system populations, approximations such as using Keplerian two-body solutions, or even treating only the giant planets as perturbers (as is common for outer solar system objects), are not sufficient, as such approximations provide enough precision for minor body identification and long-term tracking given Rubin’s expected astrometric performance. Furthermore LSST’s discoveries will be made with data spanning weeks, months, and years. Thus, *Sorcha* needs to propagate a large sample of orbits through the full set of survey exposures in order to simulate the discovery process, which is a significant computational challenge. We confine ourselves to finding an efficient solution to this problem: given a sample of orbits of solar system objects, defined at a reference epoch, and a set of visits (exposure times and telescope pointings), find the sets of objects that appear within the FOV of the camera at each of the visits. Although simulating Rubin’s LSST is our primary motivation, our solution to this problem is widely applicable.

The remainder of this paper is organized as follows. In Section 2, we describe the process of generating an ephemeris for a single solar system body observed at a single time from a single observatory. This serves to illustrate the essential steps in the calculation. We describe how we make those individual steps more efficient. We also compare our ephemeris results to those of other ephemeris generation tools, to ensure that they are correct. In Section 3, we describe our solution to enabling ephemeris generation at LSST scale. Next, in Section 4, we test our optimizations by comparing the results of our solution to those of a brute-force application of the technique described in Section 2. In Section 5, we discuss a number of caveats related to our solution to large-scale ephemeris generation. Finally, we summarize our results and suggest possible future work in Section 6.

## 2. Simple Ephemeris Generation

In this section, we describe each of the steps in generating an ephemeris position for a single small solar system body, observed at a single time. This illustrates the computational cost of each of the steps and motivates the design choices within *Sorcha* that we make to optimize this process for a significant population of objects calculated a large number of times.

For our purposes, an ephemeris position is a unit vector from the observatory in the direction of the solar system body at the time of the observation:

$$\hat{\rho}(t) = \frac{\mathbf{r}(t - \Delta t) - \mathbf{r}_{\text{obs}}(t)}{|\mathbf{r}(t - \Delta t) - \mathbf{r}_{\text{obs}}(t)|}, \quad (1)$$

where  $\hat{\rho}(t)$  is the observed unit vector at time  $t$ ,  $\mathbf{r}(t - \Delta t)$  is the position of the object at time  $t - \Delta t$ , and  $\mathbf{r}_{\text{obs}}(t)$  is the position

of the observatory at time  $t$ . Both the position of the object and the position of the observatory are measured from the same origin (the solar system barycenter). The light time is  $\Delta t = |\mathbf{r}(t - \Delta t) - \mathbf{r}_{\text{obs}}(t)|/c$ , where  $c$  is the speed of light.

There are a few primary calculations needed to generate an ephemeris position with Equation (1): (1) determining the location of the observatory at the time of the observation; (2) converting the orbit to initial conditions suitable for numerical integration; (3) determining the location of the object via numerical integration; and (4) adjusting that location to account for the light time. We describe and validate each of these steps and the overall process in the following subsections.

### 2.1. Observatory Location

Determining the location of the observatory is critical to obtaining accurate ephemeris positions; if the position of the observatory is poorly known, the calculated unit vector toward the object will be as well (Equation (1)). For a ground-based observatory, one calculates the position of the geocenter with respect to the Sun or barycenter, calculates the position of the observatory with respect to the geocenter, and then sums the two.

While the process of calculating the observatory position can be computationally expensive, an observatory’s position only needs to be calculated once per survey exposure. These calculations are done in advance within *Sorcha*. A table of the locations of ground-based observatories is maintained by the Minor Planet Center (MPC).<sup>16</sup> These are geocentric positions with respect to the Earth’s body-fixed frame. One must carry out a rotation, or a sequence of rotations, to get the position of the observatory with respect to the geocenter in an inertial frame. Traditionally, this was done with formulas that expressed the time-dependent direction of the Earth’s pole and its rate of rotation of the Earth (S. E. Urban & P. K. Seidelmann 2014). However, the tidal influence of the Moon and Sun, as well as the time-dependent moments of inertia of the Earth, imply that the rotation state of the Earth can only be predicted approximately. Instead, models of the Earth’s rotation state are fit to observations to precisely determine its past values and to provide accurate predictions for the Earth’s rotation state. The model fits and predictions are updated daily. In addition to the positions of the planets, NASA’s Navigation and Ancillary Information Facility provides Spacecraft Planet Instrument C-matrix Events (SPICE) kernels that tabulate the rotation matrix from the body-fixed frame of the Earth (ITRF93) to the International Celestial Reference Frame (ICRF), as a function of time. Specifically, we use the SPICE function `pxform` to obtain the matrix for this rotation, for each observation at the exposure midtimes. We verify our calculation by comparing the results to those from JPL’s Horizons service<sup>17</sup> and find agreement at the level of  $\sim 10$  m.

### 2.2. Initial Conditions and Coordinate Conversations

As input for numerical integration, we need equatorial barycentric state vectors  $\{\mathbf{r}, \dot{\mathbf{r}}\}$ . However, as an input, we wish to be able to handle sets of (ecliptic-aligned) Keplerian

<sup>15</sup> S. R. Merritt et al. (2025), provides an in-depth introduction to how the simulator is built and works. The source code is written in Python, and the repository and documentation are accessible online. Repository: <https://github.com/dirac-institute/sorcha> and documentation: <https://sorcha.readthedocs.io>.

<sup>16</sup> The MPC provides this information in the observatory codes data file available at <https://www.minorplanetcenter.net/data>.

<sup>17</sup> <https://ssd.jpl.nasa.gov/horizons/app.html>

elements (semimajor axis  $a$ , eccentricity  $e$ , inclination  $i$ , longitude of ascending node  $\Omega$ , argument of pericenter  $\omega$ , and mean anomaly  $\mathcal{M}$ ) or cometary orbital elements (pericenter distance  $q$ , eccentricity  $e$ , inclination  $i$ , longitude of ascending node  $\Omega$ , argument of pericenter  $\omega$ , and time of pericenter passage  $T_p$ ), that can be either heliocentric or barycentric, defined at an arbitrary epoch  $t$ , for multiple object populations. These can range from near-circular ( $e \approx 0$ ) orbits in the case of, for example, MBAs or trans-Neptunian objects (TNOs), parabolic ( $e = 1$ ) for long-period comets (LPCs), or hyperbolic ( $e > 1$ ) for interstellar objects (ISOs). To be able to handle all of these scenarios efficiently, we implement the universal variables method of K. Stumpff (1947), as described in J. M. A. Danby (1992). The primary advantage of this methodology is that a single procedure can be used to handle all transformations, independent of the (physically allowed) value of  $e$ .

We test our implementation in a few different ways. First, we begin with a toy model, in a unitless coordinate system in which the reduced mass  $\mu = 1$ . Any orbit in this system with  $i = \Omega = \omega = 0^\circ$  and is at perihelion ( $T_p = t$  or  $\mathcal{M} = 0^\circ$ ) will have  $x = q$  and  $y = z = 0$ . Such an orbit also has  $v_x = v_z = 0$ , and  $v_y > 0$  by construction. We verify that our transformations match these criteria by choosing a unitless value of  $q = 10$  and numerically computing these transformations for  $0 \leq e \leq 10$ , including  $e = 1$  and small perturbations around this neighborhood, with differences in the predicted positions and the expected values smaller than  $10^{-15}$ . We also verify that by changing  $i$ ,  $\Omega$ , and  $\omega$ , the orbit is rotated accordingly.

We then proceed to test our transformations by comparing our state vectors to those provided by JPL Horizons for three different objects: MBA (3666) Holman, ISO 11/'Oumuamua, and the eccentric TNO 2014 WB<sub>556</sub>. We obtain differences on the order of  $10^{-10}$  au in the derived positions for 11/'Oumuamua, and  $10^{-13}$  au for the other two objects, and, for all three objects, velocity differences are smaller than  $10^{-13}$  au day<sup>-1</sup>.

### 2.3. ASSIST Integrator

For accurate positions of the target objects, we use ASSIST (M. J. Holman et al. 2023), a package for carrying out ephemeris-quality integrations of test particles. By design, the ASSIST integrator almost identically matches the results of the JPL small bodies integrator, the current standard of reference for small body ephemerides. ASSIST is an extension of REBOUND (H. Rein & S. F. Liu 2012) and makes use of its IAS15 integrator (H. Rein & D. S. Spiegel 2015) to integrate test particle trajectories in the field of the Sun, Moon, planets, and 16 massive asteroids, with the positions of the masses coming from the JPL DE440/441 ephemeris and its associated asteroid perturber file (R. S. Park et al. 2021). ASSIST incorporates the most significant gravitational harmonics and general relativistic corrections. It can also account for position- and velocity-dependent nongravitational effects.

Although ASSIST integrations are fast, we exploit a feature of the IAS15 integrator to further optimize ephemeris calculations. An individual IAS15 time step involves iteratively solving for the coefficients of a 15th-order polynomial that accurately approximates the trajectory over the interval from the start of the time step to the end (H. Rein & D. S. Spiegel 2015). After converging, the polynomial is used

to evaluate the location of the object at the end of the step. However, the same polynomial coefficients can be used to evaluate the location of the object at any time within the time step interval. The high order of the polynomial enables each time step to be quite large (several days for MBAs, longer for TNOs, and shorter for NEOs). Evaluating the polynomial is computationally negligible compared to the integration step itself.

REBOUND retains the coefficients of the previously completed time step. ASSIST includes a routine called `integrate_or_interpolate` that accepts a time as an argument. If that time lies within the interval of the previous time step, the routine interpolates using the existing coefficients. Otherwise, it takes the required time step. The nominal LSST observing strategy involves taking three pairs of exposures on distinct nights spanning 10–15 days. The typical interval between successive exposures of a region of sky is often smaller than an IAS15 time step, so most ephemeris calculations can be done with polynomial evaluations. An evaluation of the interpolation polynomials for a single test particle is  $\sim 30,000$  times faster than a single full integration step. The degree to which this optimization speeds up any particular *Sorcha* simulation depends upon the details of the survey. If there are many visits of the same region of sky within a relatively short span of time, the speedup can be significant. However, for surveys with exposures widely separated both spatially and temporally, the speedup from interpolating will be limited. For typical solar system surveys, a speedup of a factor of 2–4 can be expected from using this optimization.

Numerical integration with ASSIST has been extensively tested (M. J. Holman et al. 2023). Nevertheless, we examine a few cases to demonstrate that we are using ASSIST correctly within *Sorcha*. We do this by comparing the state vectors from ASSIST with those from JPL Horizons after a month-long integration (starting at epoch 2017 October 1, midnight Barycentric Dynamical Time), for a few different objects. Our targets include objects in near-circular orbits at a variety of heliocentric distances (including an NEO), objects with high eccentricities (a TNO, 2014 WB<sub>556</sub> with barycentric  $e \approx 0.85$  and an LPC, C/2014 UN<sub>271</sub> with barycentric  $e \approx 0.9975$ ), and an ISO (eccentricity  $e > 1$ ). We present the fractional differences between the reference values for JPL state vectors and astrometry (from Rubin) and those derived by *Sorcha* in Table 1. We also cross validated the ephemeris of (3666) Holman against the independent software package *spacerocks*,<sup>18</sup> and found excellent agreement at the level of tens of microarcseconds.

### 2.4. Light Time Iteration

The time it takes for the light from a solar system object to reach the observer can be significant. During that time, the object will have moved appreciably. Thus, the observer is seeing where the object was when the observed light was reflected by or emitted from the object. In order to calculate the light time, one must know the position of the object, but to calculate the position of the object one must know the light time. Thus, Equation (1) is implicit.

Fortunately, iteration is a simple and reliable approach to solving for the light time. One starts with an initial guess for

<sup>18</sup> *spacerocks* is available online at <http://github.com/kjnapier/spacerocks>.

**Table 1**  
State Vector and Astrometry Differences after Month-long Integrations

Reference Object	Orbit Type	$\ \mathbf{r}_{\text{Sorcha}} - \mathbf{r}_{\text{JPL}}\ /\ \mathbf{r}_{\text{JPL}}\ $	$\ \dot{\mathbf{r}}_{\text{Sorcha}} - \dot{\mathbf{r}}_{\text{JPL}}\ /\ \dot{\mathbf{r}}_{\text{JPL}}\ $	$\Delta\alpha \cos \delta^a$ ( $\mu\text{as}$ )	$\Delta\delta$ ( $\mu\text{as}$ )
2010 TU <sub>149</sub>	MBA	$5.3 \times 10^{-8}$	$1.2 \times 10^{-8}$	-26	-3.2
(3666) Holman	MBA	$5.9 \times 10^{-8}$	$2.0 \times 10^{-8}$	-23	5.8
2011 OB <sub>60</sub>	TNO	$4.3 \times 10^{-8}$	$6.2 \times 10^{-7}$	0.7	0.9
2014 WB <sub>556</sub>	TNO	$4.3 \times 10^{-8}$	$7.8 \times 10^{-7}$	2.2	1.6
1I/'Oumuamua	ISO	$9.6 \times 10^{-5}$	$3.1 \times 10^{-4}$	$-24 \times 10^6$	$-0.89 \times 10^{6b}$
(433) Eros	NEO	$6.8 \times 10^{-8}$	$1.1 \times 10^{-8}$	-23	5.8
C/2014 UN <sub>271</sub> (Bernardinelli-Bernstein)	LPC	$4.3 \times 10^{-8}$	$3.0 \times 10^{-7}$	0.7	1.3

**Notes.**

<sup>a</sup> The R.A./decl. positions were calculated with respect to Rubin.

<sup>b</sup> 1I/'Oumuamua's ephemerides from JPL includes nongravitational acceleration terms that are not captured in our simulations.

$\Delta t$ , such as  $\Delta t = 0$ , finds  $\rho = |\mathbf{r}(t - \Delta t) - \mathbf{r}_{\text{obs}}(t)|$ , and updates  $\Delta t = \rho/c$ . This procedure typically converges within a few iterations. After the light time iteration, the ephemeris position at time  $t$  is known.

Rather than testing for convergence, we set the default number of iterations to three. We verified that changing the number of iterations from three to four does not appreciably change the results, implying convergence. As a final check, we verified that  $\Delta t = |\mathbf{r}(t - \Delta t) - \mathbf{r}_{\text{obs}}(t)|/c$ .

### 2.5. Validation of Simple Ephemeris Generation

We have tested each of the steps separately, and we have verified the overall calculation using the same objects as in Section 2.3. If all previous steps have been done correctly, our predicted on-sky positions from Rubin's location should closely match those produced by JPL. We start with our state vectors on a certain date and produce ephemerides for observations a month later. The results, presented in Table 1, show remarkable agreement (at most tens of microarcseconds for objects with purely gravitational motion) between our integrations and JPL's across a variety of heliocentric distances, thus validating our ephemerides generation procedure.

## 3. Large-scale Ephemeris Generation

Although simple ephemeris generation is efficient enough for many purposes, the brute-force approach becomes a computational bottleneck for survey simulations at LSST scale. As already noted (Section 1), we are considering millions of objects and millions of exposures, with the goal of determining which objects appear within the FOV of the survey camera for each exposure. Calculating the ephemeris positions of all of the objects at the times of all of the visits becomes prohibitively expensive, even with significant computing resources. Fortunately, two simple insights, when combined, lead to a much more efficient method.

1. The ephemeris positions of most objects are far outside the FOV of any particular exposure. Even an approximate position is sufficient to determine that an object cannot plausibly fall within the FOV. The  $9.6 \text{ deg}^2$  FOV of LSSTCam, although enormous by current standards, only encompasses  $\sim 1/4000$  of the sky. Even near the ecliptic, only a tiny fraction of the total sample of objects in a survey simulation will appear in any given exposure.

2. The apparent rates of motion of most solar system objects, as observed from Earth, are slow. Although NEOs can move several degrees across the sky per day, most MBAs traverse only a fraction of a degree in a day. The apparent motion of more distant objects (e.g., TNOs) is even slower. Thus, an object observed on one night will be in roughly the same location the next.

### 3.1. High-level Algorithm

Merging these two ideas yields our basic approach (demonstrated in Figure 1). We build and periodically update a data structure that allows us to efficiently determine which simulated objects are in which regions of the sky, on a given night. We call this data structure, which records the topocentric positions for all the objects, a picket, because we will be computing a sequence of these, like pickets in a fence. Although one could store these pickets, we will maintain one at a time, to limit the memory footprint of the simulation.

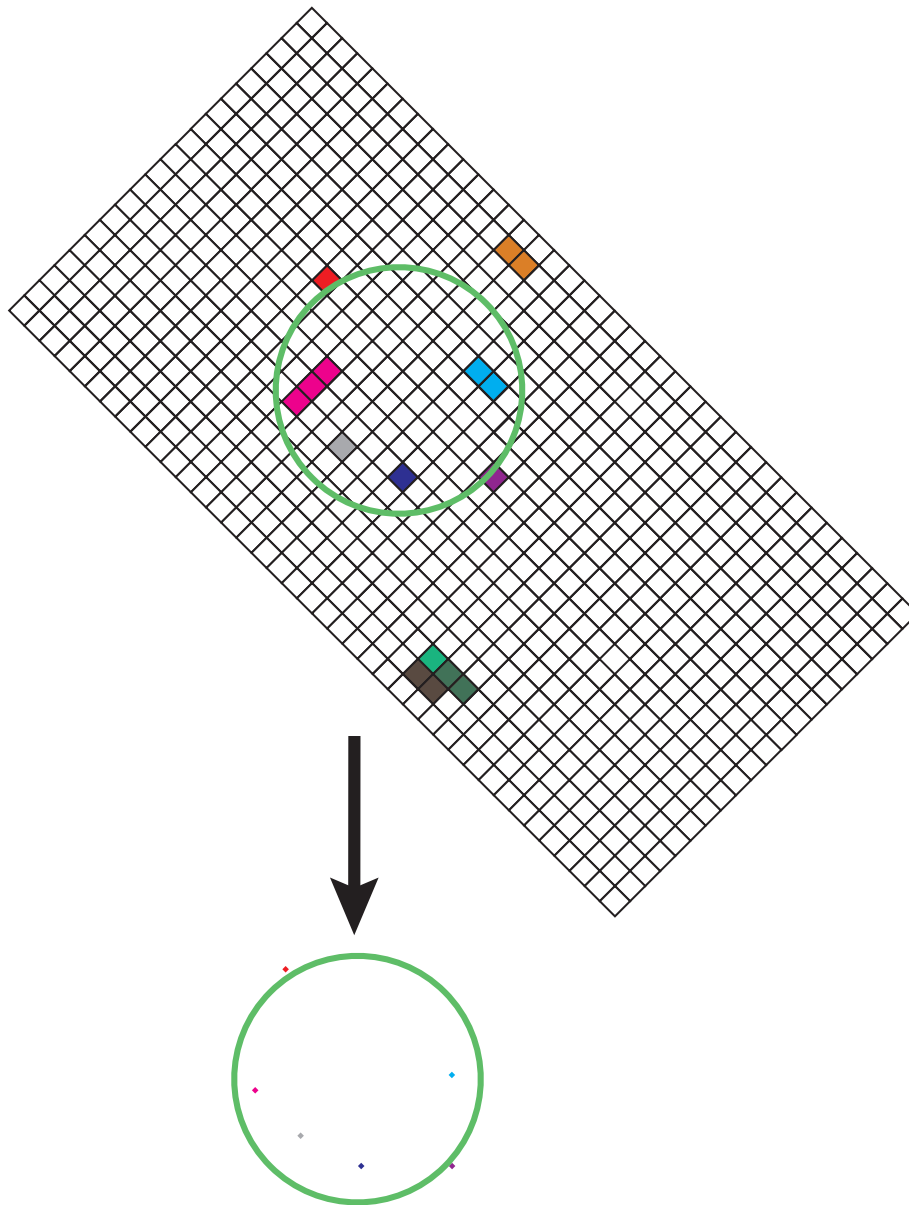
We use the HEALPix tessellation (K. M. Górski & E. Hivon 2011), which divides the sky into equal-area pixels and enumerates them. (We orient the HEALPix tessellation with the ICRF and use the  $n_{\text{side}} = 128$  scale.)<sup>19</sup> For a given night, our data structure maintains a mapping between the HEALPix index and the set of simulated objects that fell within the corresponding region of the sky at any point during the night.

Given the R.A./decl. of a given exposure, we find all the HEALPix tiles overlapped by a circular region with an angular radius that encompasses the camera FOV, including a buffer region to accommodate some uncertainty (discussed below). We then collect the IDs of the simulated objects in those tiles and calculate precise ephemeris positions for the exposure time for these candidate objects. Any objects that actually fall within the angular radius of the FOV are retained.

Other tessellations of the sky would also work. The only requirements are that it is easy to (1) determine the tile index from a sky position and (2) determine which tiles are overlapped by the FOV of an exposure. We selected HEALPix because healpy<sup>20</sup> (K. M. Górski & E. Hivon 2011; A. Zonca et al. 2019) provides efficient Python support for these operations.

<sup>19</sup> The HEALPix  $n_{\text{side}}$  scale value is configurable within Sorcha, but we have found a default of  $n_{\text{side}} = 128$ , which corresponds to a scale of 0.45, worked well for all known solar system populations.

<sup>20</sup> <https://healpy.readthedocs.io/en/latest/>



**Figure 1.** Cartoon schematic of ephemeris generation within *Sorcha*. Each color represents a different moving object on a different orbit. Slower moving objects will cover fewer Hierarchical Equal Area Iso Latitude pixelation (HEALPix) tiles. The green circle represents an area slightly bigger than LSSTCam’s footprint. For the given observation time, any orbits with a HEALPix tile within the circle are integrated to calculate their exact positions at the time of the observation. Those orbits that land within the circle are then identified, and the resulting ephemerides associated with those objects and the observation are saved and passed on to later stages of the simulator that calculate the apparent magnitude and estimate the likelihood of discovery.

There are many ways to implement and exploit the aforementioned concepts. We chose to employ the following procedure, but we also note that further refinements are possible. Consider a toy example in which we have a single exposure, and we wish to determine which of the simulated objects fall within its bounds. The exposure has a time, a topocentric unit vector pointing toward the center of the FOV, and an angular radius that circumscribes the FOV. The FOV of the exposure overlaps a set of HEALPix tiles, which we can determine in advance.

Within *Sorcha*, for each object in our simulated population we:

1. calculate the ephemeris positions on three sequential nights (see Section 3.2), ensuring that the epoch of the exposure is within that time span;
2. finely interpolate those topocentric positions across that 2 day interval;
3. determine which HEALPix tiles the object will traverse in that time interval; and
4. associate the object’s ID with the indices of the HEALPix tiles it crosses (we use a hash table/dictionary for this purpose).

Then we obtain all objects that could possibly appear within the exposure by collecting the set of object IDs in the set of HEALPix tiles that the exposure overlaps. For each of those candidate objects, we determine its precise ephemeris position at the time of the exposure. Only those that actually fall within the FOV are retained.

For a simulation of an LSST survey with millions of exposures, part of the advantage of using this approach is that

steps 1–4 do not necessarily need to be repeated for each exposure. Consider a time-ordered sequence of exposures spanning a number of days. After calculating steps 1–4 for the first exposure, the results can be reused for all the exposures whose epochs land within the 2 day interval spanned by the three points in the picket. If an exposure falls outside that interval, the data must be updated.

### 3.2. Orbits and Ephemerides

The toy example in the previous subsection did not elaborate on how the propagation of orbits is done, but it is computationally intensive and extremely difficult to execute correctly. In order to allow our simulations to scale well to the volume of LSST data, we need to be able to handle millions of orbits. To achieve the fidelity required by a survey that spans a decade and has superb astrometric precision, we need to use an extremely accurate integration scheme that supports high-quality ephemerides. The ephemeris generation tool needs to be able to achieve this with a reasonable number of computing cores and memory in a reasonable amount of time.

To enable this performance and precision, we have taken a counterintuitive approach in which we input the initial orbits one by one and instantiate an individual REBOUND/ASSIST simulation object for each orbit. Since we treat the small bodies as massless test particles (i.e., they do not interact with each other), it would seem more efficient to incorporate a large number of solar system bodies into each numerical integration. For fixed time step integrators, particularly when the orbits of the planets and other perturbing bodies must also be numerically integrated, one can amortize the computational overhead by applying the same calculation to a large number of test particles. However, ASSIST does not numerically integrate the positions of the Sun, planets, and other perturbing bodies. Instead, it efficiently evaluates them through memory-mapped tables of Chebyshev polynomial coefficients. Furthermore, integrating groups of test particles is only efficient if the step size needed is roughly the same for all of the test particles. If a few require particularly small time steps in order for their results to be accurate, integrations of the other objects may be unnecessarily expensive. This is also true for IAS15, despite it being a variable time step method. Each simulation has a single, variable time step that is applied to all of the equations in the simulation. The length of the time step is dominated by the most rapidly changing equation (H. Rein & D. S. Spiegel 2015). Solar system objects have a wide range of orbital periods that correspondingly require a wide range of step sizes. By separating each orbit into its own REBOUND simulation, the integrator will naturally use the time step that is best suited to that object. Furthermore, this means that each object has its own set of polynomial coefficients that can be used to evaluate its position within a time step. In addition, keeping a separate REBOUND/ASSIST object for each orbit allows for easy bookkeeping: as we read in the orbits and instantiate the REBOUND/ASSIST objects, we build a Python dictionary that has the object IDs as keys and the REBOUND/ASSIST objects as values.

### 3.3. Processing Exposures

Next, we process the survey exposures, one by one, roughly following the approach described in the toy example above. Although the exposures do not need to be processed in

chronological order, it is far more efficient to do so. We process the exposures as follows.

1. Read the exposure midpoint time and R.A./decl. of the survey pointing center. Calculate the observatory position for that time, if it is not already available.
2. Calculate which HEALPix tiles the exposure overlaps.
3. If the time of the exposure is outside of the time interval of the current picket, update the picket.
4. Collect the IDs of the objects in the HEALPix tiles that the exposure overlaps.
5. Integrate just those objects to the time of the exposure, including light time.
6. After the set of objects that *might* land in the exposure have been integrated, check which actually fall within the angular radius of the FOV from the center of the exposure. For those objects, calculate and record the rates of motion and any other information that is needed in the postprocessing stages of Sorcha.

This process is essentially implemented as a generator, hiding the details from the user. For a given exposure, the process returns the information for the objects that fall within that exposure.

As noted earlier (Section 2.3), the `integrate_or_interpolate` routine makes this step much more efficient. The time steps can be large, often several days. Thus, most of the time no actual numerical integration needs to be done. In addition, because each of the test particles has its own REBOUND/ASSIST object, the adjustment of the integration time steps is handled simply and automatically.

There will be hundreds to thousands of exposures per night of LSST observations. If the exposures are processed in time order, much of the time the picket does not need to be updated. Sequences of exposures taken within the same night or adjacent nights can use the same picket. Groups of exposures separated by more than a day will typically need different picket information. The interval between pickets is discussed in more detail below.

### 3.4. Design Considerations

The design of the picket data structure and its associated algorithms was motivated by fast-moving objects. NEOs, for example, can traverse several HEALPix tiles between one night and the next, depending upon the topocentric distance to the object, the interval between pickets, and the scale of the HEALPix tiles. We need to allow for this possibility to keep track of such objects. Fortunately, there is a fairly simple solution to this problem. When updating the picket, we record all the HEALPix tiles each object will cross in the two night interval covered by the picket. Then we add the object IDs to the hash map/dictionary for each of those HEALPix tiles. We use moving-window third-order Lagrange polynomials to interpolate the topocentric unit vectors to the objects, across three adjacent nights. We evaluate the position (and HEALPix indices) at 101 interpolation points (this is adjustable via Sorcha’s configuration file). This adds negligible overhead to the overall calculation.

The interval between pickets is an adjustable parameter. We set this to be 1 day to match the natural cadence of ground-based observations, and the amount of motion for the vast majority of objects in a typical simulation is modest on that timescale. Likewise, the HEALPix scale is a tunable

parameter. A smaller scale means that a set of pixels can more closely approximate the shape of the LSSTCam FOV. However, a finer scale results in larger data structures and a larger memory footprint. Although our testing demonstrates that these are reasonable values, we leave a detailed study involving the picket interval, the HEALPix scale, and the number of interpolation points to future work.

#### 4. Validation

We compare the results of *Sorcha*'s optimized ephemeris generation with those of a brute-force application of the method described in Section 2. In particular, we aim to ensure that we are not missing objects. As part of our validation of *Sorcha*, described in detail in S. R. Merritt et al. (2025), we have simulated the observability of three objects (the NEO 433 Eros, the TNO 2011 OB<sub>60</sub>, and the MBA 2010 TU<sub>149</sub>) over a month of LSST observations with a realistic cadence ( $\approx 20,000$  visits), both through *Sorcha* and by directly producing ephemerides by querying JPL. These tests validate not only our ephemerides, but also the HEALPix tiling, the time picket algorithm, and the FOV selection criteria. The JPL-produced ephemerides have a cadence of a minute, and so we linearly interpolated these dense sets of positions to the corresponding exposure midpoint times from the simulated LSST visits, and used a simple independent implementation of the LSST FOV filtering for each visit. We obtain the same astrometric precision shown in Table 1 for these two objects for all detections over the chosen time span, and furthermore, these tests also indicate that all observations are properly identified by our algorithm.

We carried out a number of timing tests. First, we ran 10 MBAs from the *Sorcha* demonstration files. Using *Sorcha* with all the optimizations described in this paper, it takes 12 s, on a Macbook Pro with an M4 processor, for  $\sim 216,000$  visits over the span of a year of LSST observations. Using the same input orbits and the same sequence of visits, but requiring the positions of all the objects to be calculated for each exposure time (i.e., a brute-force calculation), takes 817 s on the same hardware.

Second, we ran 100 MBAs for  $\sim 216,000$  visits, on the same hardware. This took 15 s with all optimizations, but 1600 s as a brute-force calculation. The timing is roughly proportional to the number of objects simulated, with some overhead.

Third, again on a Macbook Pro, with an M4 processor, we ran 15,000 objects for the same  $\sim 216,000$  visits. We found that using `integrate` took 745 s, while using `integrate_or_interpolate` took 278 s. This supports our earlier statement that in typical surveys, the use of `integrate_or_interpolate` can yield a speedup of 2–4.

#### 5. Caveats

There are a few caveats regarding ephemeris generation that the *Sorcha* user should be aware of.

1. Although we have tested the behavior of routines for very fast-moving objects, it is difficult to prove correctness in all cases.
2. If one of the objects listed in Table 2 is included as a test particle in a *Sorcha* simulation, the results can be wildly inaccurate. Without modifying the list of perturbers used by ASSIST, the test particle will

**Table 2**

The 16 Main Belt Asteroids and Dwarf Planet Perturbers Included within the ASSIST Integrations

Asteroid MPC Designation	Asteroid MPC Designation
(107) Camilla	(704) Interamnia
(1) Ceres	(7) Iris
(65) Cybele	(3) Juno
(511) Davida	(2) Pallas
(15) Eunomia	(16) Psyche
(31) Euphrosyne	(87) Sylvia
(52) Europa	(4) Vesta
(10) Hygiea	(88) Thisbe

**Note.** As these objects and their masses are included with the ASSIST integrations, they will gravitationally perturb any orbits on similar or exact orbits. Thus, poor predictions will result if *Sorcha* ephemeris generator attempts to run on the perturber orbits.

experience the gravitational influence of a massive version of itself, at very close range.

3. Currently, ASSIST includes only one formulation of nongravitational acceleration. Likewise, as noted in S. R. Merritt et al. (2025), ASSIST does not include routines for the acceleration due to cometary outgassing, the fragmentation of objects, monitoring for planetary impacts, and other effects that might be of interest to the community. These and other features are feasible to account for, but these would need to be implemented in the software. A primary motivation for making ASSIST and *Sorcha* open-source projects is to enable other members of the community to develop and contribute such improvements.
4. For this version of *Sorcha*, we ignore orbital uncertainties and their resulting positional uncertainties. For a survey simulation, we assume we know the orbits and physical characteristics of the sample of solar system objects being tested. However, in future work, we can examine how positional uncertainties affect discovery, particularly in the early stages of the survey when there are limited data.

#### 6. Summary

The dawn of the petabyte era in astronomical surveys brings great challenges and rewards. The significant increase in photometric and astrometric observations requires more efficient algorithms to accurately predict which orbits will land within past survey images. This is both to identify where the millions of known minor planets recorded in the MPC have serendipitously been imaged by these large surveys, but also to enable simulations that predict the locations of millions to billions of synthetic objects in survey observations in order to compare population models to those discovered by these surveys. We have described the fast ephemeris generation algorithm designed for the *Sorcha* survey simulator (S. R. Merritt et al. 2025) and how we have overcome many of the challenges of scaling up for the LSST. We note that the ephemeris generator within *Sorcha* has been designed and coded with ASSIST and REBOUND in mind, but the algorithm itself is agnostic to the choice of  $N$ -body integrator. Likewise, *Sorcha*'s ephemeris generator can be used separately from the rest of the package. For example, the ephemeris generation

components have been integrated into the known object association software within Rubin’s Solar System Processing pipelines.<sup>21</sup> Our algorithm may also be of particular interest to large astronomical databases such as the Barbara A. Mikulski Archive for Space Telescopes<sup>22</sup> and the Astro Data Lab<sup>23</sup> archive at the National Optical-Infrared Astronomy Research Laboratory.

Future work could include (1) propagating positional uncertainties as part of ephemeris generation, (2) efficiently storing the data at a series of pickets to support the efficient identification of known small bodies for a range of times, (3) a detailed study involving the picket interval and HEALPix scale, as noted in Section 3, and (4) adding the capacity to keep a short list of objects that are always evaluated via brute-force methods, to guarantee that key objects are not missed.

### Acknowledgments

This work was supported by an LSST Discovery Alliance LINCC Frameworks Incubator grant [2023-SFF-LFI-01-Schwamb]. Support was provided by Schmidt Sciences. M.J.H. gratefully acknowledges support from the NSF (grant No. AST2206194), the NASA YORPD Program (grant No. 80NSSC22K0239), and the Smithsonian Scholarly Studies Program (2022, 2023). S.R.M. and M.E.S. acknowledge support in part from UK Science and Technology Facilities Council (STFC) grants ST/V000691/1 and ST/X001253/1. G.F. acknowledges support in part from UK Science and Technology Facilities Council (STFC) grant ST/P000304/1. This project has received funding from the European Union’s Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 101032479. C.O.C., M.J., and P.H.B. acknowledge the support from the University of Washington College of Arts and Sciences, Department of Astronomy, and the DiRAC (Data-intensive Research in Astrophysics and Cosmology) Institute. The DiRAC Institute is supported through generous gifts from the Charles and Lisa Simonyi Fund for Arts and Sciences and the Washington Research Foundation. M.J. wishes to acknowledge the support of the Washington Research Foundation Data Science Term Chair fund, and the University of Washington Provost’s Initiative in Data-Intensive Discovery. J. M. acknowledges support from the Department for the Economy (DfE) Northern Ireland postgraduate studentship scheme and travel support from the STFC for UK participation in LSST through grant ST/S006206/1. J.A.K. and J.M. thank the LSST-DA Data Science Fellowship Program, which is funded by LSST-DA, the Brinson Foundation, and the Moore Foundation; their participation in the program has benefited this work. S.E. and S.C. acknowledge support from the National Science Foundation through the following awards: Collaborative Research: SWIFT-SAT: Minimizing Science Impact on LSST and Observatories Worldwide through Accurate Predictions of Satellite Position and Optical Brightness NSF Award Number: 2332736 and Collaborative Research: Rubin Rocks: Enabling near-Earth asteroid science with LSST NSF Award Number: 2307570. Any opinions, findings, and conclusions or recommendations expressed in

this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

This work was also supported via the Preparing for Astrophysics with LSST Program, funded by the Heising Simons Foundation through grant 2021-2975, and administered by Las Cumbres Observatory. This work was supported in part by the LSST Discovery Alliance Enabling Science grants program, the B612 Foundation, the University of Washington’s DiRAC Institute, the Planetary Society, Karman +, and Adler Planetarium through generous support of the LSST Solar System Readiness Sprints.

This research has made use of NASA’s Astrophysics Data System Bibliographic Services. This research has made use of data and/or services provided by the International Astronomical Union’s Minor Planet Center. The SPICE Resource files used in this work are described in C. H. Acton (1996) and C. Acton et al. (2018). Simulations in this paper made use of the REBOUND *N*-body code (H. Rein & S. F. Liu 2012). The simulations were integrated using IAS15, a 15th-order Gauss–Radau integrator (H. Rein & D. S. Spiegel 2015). Some of the results in this paper have been derived using the `healpy` and `HEALPix` packages. This work made use of `Astropy`:<sup>24</sup> a community-developed core Python package and an ecosystem of tools and resources for astronomy (Astropy Collaboration et al. 2013, 2018, 2022). We are additionally grateful to the members of the Rubin Observatory LSST Solar System Science Collaboration for useful feedback at the LSST Solar System Readiness Sprints. We also thank the contributors to Stack Overflow for their examples and advice on common Python challenges that provided guidance on solving some of the programming challenges we have encountered.

This material or work is supported in part by the National Science Foundation through Cooperative Agreement AST-1258333 and Cooperative Support Agreement AST1836783 managed by the Association of Universities for Research in Astronomy (AURA), and the Department of Energy under Contract No. DE-AC02-76SF00515 with the SLAC National Accelerator Laboratory managed by Stanford University.

We are grateful for use of the computing resources from the Northern Ireland High Performance Computing (NI-HPC) service funded by EPSRC (EP/T022175). We gratefully acknowledge the support of the Center for Advanced Computing and Modeling, University of Rijeka (Croatia), for providing supercomputing resources at HPC (High Performance Computing) Bura.

### Author Contributions

M.J.H. led the development and refinement of `Sorcha`’s ephemeris generator algorithm. He also helped verify the output from the ephemeris generator components and improved the algorithm to better handle fast-moving solar system objects. He also wrote the majority of the text within the manuscript.

P.H.B. contributed to the development of several of the `Sorcha` routines for ephemerides generation and coordinate conversion. He also helped verify the output from the ephemeris generator components and improved the algorithm to better handle fast-moving solar system objects. He also contributed significant portions of text to the manuscript.

<sup>21</sup> <https://dmtn-087.lsst.io/>

<sup>22</sup> <https://archive.stsci.edu>

<sup>23</sup> <https://datalab.noirlab.edu/>

<sup>24</sup> <http://www.astropy.org>

M.E.S. served as principal investigator and project manager of the *Sorcha* team. She also served as PI of the *Sorcha* LINCC Frameworks Incubator Proposal. She contributed to the discussions and decisions about the overall design and implementation of *Sorcha* and the documentation: this included leading the team in-person meetings and calls. She created Figure 1, contributed some text to the paper, and also provided feedback on the manuscript.

M.J. contributed to the original brainstorming sessions about the *Sorcha* ephemeris generator design and implementation. He wrote the original Jupyter notebook that was used as a template by M.J.H to incorporate ASSIST + REBOUND. M.J. also searched through a petabyte of files on disk to provide the notebook to M.J.H.

D.O. led the incubator software engineering team during the LINCC Frameworks Incubator and contributed to integrating the first draft of the ASSIST + REBOUND ephemeris generator within *Sorcha*.

M.W. contributed to the *Sorcha* code base, including efficiency improvements in array handling and integrating the first draft of the ASSIST + REBOUND ephemeris generation.

K.J.N. cross validated the *Sorcha* ephemeris against the ephemeris produced by the independent software *spacerocks* (<https://github.com/kjnapier/spacerocks>). He also helped to review parts of the software, and identified minor algorithmic improvements.

S.R.M. served as the lead developer of *Sorcha* from 2021 June onward, contributing a substantial portion of the current code base and the majority of unit tests. They also led new feature implementations, oversaw ongoing code maintenance and repository management, resolved software issues, and conducted extensive testing on both HPC clusters and local systems.

G.F. was the initial developer of *Sorcha* and contributed significantly to early drafts of the manuscript. G.F. participated in the discussions and decisions about the overall design and implementation of *Sorcha*. He also provided feedback on the overall paper draft.

S.C. contributed to *Sorcha* and assisted in stress testing *Sorcha*'s ephemeris generator and identifying rare cases during these beta tests that required changes that improved the ephemeris generation within *Sorcha*.

J.A.K. provided testing, user feedback, and bug fixing throughout *Sorcha*'s development. He also provided feedback on the paper manuscript.

S.E. contributed to the initial discussions about *Sorcha*'s ephemeris generator as well as providing feedback on the LINCC Frameworks Incubator proposal. He was involved in the discussions about the overall design and functionality of *Sorcha*. He also provided feedback on the paper manuscript.

J.K. contributed to the *Sorcha* code base, including improvements to input file reading of the user generated ephemeris files.

K.K. contributed to beta testing *Sorcha* for the edge case of simulated Earth impactors and very fast-moving potentially hazardous asteroids.

J. M. provided testing, user feedback, and bug fixing throughout *Sorcha*'s development. He also provided feedback on the paper manuscript.

S.P.N. developed the *objectsInField* (OIF) ephemeris generator/simulator that enabled the development of *Sorcha* and its ephemeris generator. The output from OIF was the

input into early iterations of *Sorcha* as the source of ephemeris calculations. OIF's design influenced the overall architecture of the *Sorcha* ephemeris generator.

C.O.C. contributed to discussions about the development and enhancement of *Sorcha*, including the ephemeris generator, during the LINCC Frameworks Incubator. C.O.C. also provided feedback on the paper manuscript.

Data access: the algorithm presented here is available in the *Sorcha* open-source Python package at <https://github.com/dirac-institute/sorcha>.

*Software*: *Sorcha*, ASSIST (M. J. Holman et al. 2023; H. Rein et al. 2023), Astropy (Astropy Collaboration et al. 2013, 2018, 2022), healpy (K. M. Górski & E. Hivon 2011; A. Zonca et al. 2019), Numba (S. K. Lam et al. 2015), Numpy (C. R. Harris et al. 2020), pandas (W. McKinney 2010; The pandas development team 2020), REBOUND (H. Rein & S. F. Liu 2012; H. Rein & D. S. Spiegel 2015), SciPy (P. Virtanen et al. 2020), Spicypy (A. Annex et al. 2020), Black (<https://black.readthedocs.io/en/stable/faq.html>), Jupyter Notebooks (T. Kluyver et al. 2016), *spacerocks* (<https://github.com/kjnapier/spacerocks>).

## ORCID iDs

Matthew J. Holman  <https://orcid.org/0000-0002-1139-4880>  
 Pedro H. Bernardinelli  <https://orcid.org/0000-0003-0743-9422>  
 Megan E. Schwamb  <https://orcid.org/0000-0003-4365-1455>  
 Mario Jurić  <https://orcid.org/0000-0003-1996-9252>  
 Drew Oldag  <https://orcid.org/0000-0001-6984-8411>  
 Maxine West  <https://orcid.org/0009-0003-3171-3118>  
 Kevin J. Napier  <https://orcid.org/0000-0003-4827-5049>  
 Stephanie R. Merritt  <https://orcid.org/0000-0001-5930-2829>  
 Grigori Fedorets  <https://orcid.org/0000-0002-8418-4809>  
 Samuel Cornwall  <https://orcid.org/0000-0002-0672-5104>  
 Jacob A. Kurlander  <https://orcid.org/0009-0005-5452-0671>  
 Siegfried Eggli  <https://orcid.org/0000-0002-1398-6302>  
 Jeremy Kubica  <https://orcid.org/0009-0009-2281-7031>  
 Joseph Murtagh  <https://orcid.org/0000-0001-9505-1131>  
 Shantanu P. Naidu  <https://orcid.org/0000-0003-4439-7014>  
 Colin Orion Chandler  <https://orcid.org/0000-0001-7335-1715>

## References

- Acton, C., Bachman, N., Semenov, B., & Wright, E. 2018, *P&SS*, 150, 9  
 Acton, C. H. 1996, *P&SS*, 44, 65  
 Annex, A., Pearson, B., Seignovert, B., et al. 2020, *JOSS*, 5, 2050  
 Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, *ApJ*, 935, 167  
 Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, *AJ*, 156, 123  
 Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, *A&A*, 558, A33  
 Bernardinelli, P. H., Bernstein, G. M., Sako, M., et al. 2022, *ApJS*, 258, 41  
 Bernardinelli, P. H., Smotherman, H., Langford, Z., et al. 2024, *AJ*, 167, 134  
 Bianco, F. B., Ivezić, Ž., Jones, R. L., et al. 2022, *ApJS*, 258, 1  
 Danby, J. M. A. 1992, *Fundamentals of Celestial Mechanics*,  
 Fedorets, G., Granvik, M., Jones, R. L., Jurić, M., & Jedicke, R. 2020, *Icar*, 338, 113517  
 Górski, K. M., & Hivon, E., 2011 HEALPix: Hierarchical Equal Area IsoLatitude Pixelization of a sphere, *Astrophysics Source Code Library*, ascl:1107.018  
 Grav, T., Mainzer, A. K., & Spahr, T. 2016, *AJ*, 151, 172

- Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, *Natur*, **585**, 357
- Holman, M. J., Akmal, A., Farnocchia, D., et al. 2023, *PSJ*, **4**, 69
- Hoover, D. J., Seligman, D. Z., & Payne, M. J. 2022, *PSJ*, **3**, 71
- Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, *ApJ*, **873**, 111
- Jones, R. L., Chesley, S. R., Connolly, A. J., et al. 2009, *EM&P*, **105**, 101
- Jones, R. L., Slater, C. T., Moeyens, J., et al. 2018, *Icar*, **303**, 181
- Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, Positioning and Power in Academic Publishing: Players, Agents and Agendas (Amsterdam: IOS Press), 87, <https://eprints.soton.ac.uk/403913/>
- Kurlander, J. A., Bernardinelli, P. H., Schwamb, M. E., et al. 2025, *AJ*, **170**, 99
- Lam, S. K., Pitrou, A., & Seibert, S. 2015, in Proc. Second Workshop on the LLVM Compiler Infrastructure in HPC (New York: ACM), **1**
- LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, arXiv:0912.0201
- McKinney, W. 2010, in Proc. 9th Python in Science Conf., ed. S. van der Walt & J. Millman (Austin, TX: SciPy), 51
- Merritt, S. R., Fedorets, G., Schwamb, M. E., et al. 2025, *AJ*, **170**, 100
- Murtagh, J., Schwamb, M. E., Merritt, S. R., et al. 2025, *AJ*, **170**, 98
- Naidu, S. P., Chesley, S. R., & Farnocchia, D. 2017, AAS/DPS Meeting, **49**, 112.04
- Napier, K. J., Gerdes, D. W., Lin, H. W., et al. 2021, *PSJ*, **2**, 59
- Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. 2021, *AJ*, **161**, 105
- Petit, J. M., Kavelaars, J. J., Gladman, B. J., et al. 2011, *AJ*, **142**, 131
- Rein, H., Holman, M., & Akmal, A. 2023, matthewholman/assist: v1.1.1, Zenodo, doi:10.5281/zenodo.7778017
- Rein, H., & Liu, S. F. 2012, *A&A*, **537**, A128
- Rein, H., & Spiegel, D. S. 2015, *MNRAS*, **446**, 1424
- Schwamb, M. E., Jones, R. L., Chesley, S. R., et al. 2018, arXiv:1802.01783
- Shannon, A., Jackson, A. P., Veras, D., & Wyatt, M. 2015, *MNRAS*, **446**, 2059
- Silsbee, K., & Tremaine, S. 2016, *AJ*, **152**, 103
- Solontoi, M., Ivezić, Ž., West, A. A., et al. 2010, *Icar*, **205**, 605
- Stumpff, K. 1947, *AN*, **275**, 108
- The pandas development team 2020, pandas-dev/pandas: Pandas, latest, Zenodo, doi:10.5281/zenodo.3509134
- Urban, S. E., & Seidelmann, P. K. 2014, AS Meeting, **223**, 247.20
- Vereš, P., & Chesley, S. R. 2017, *AJ*, **154**, 13
- Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, *NatMe*, **17**, 261
- Zonca, A., Singer, L., Lenz, D., et al. 2019, *JOSS*, **4**, 1298