

AstDyn: A High-Fidelity C++ Library for Asteroid Dynamics and Occultation Prediction

M. Bigi^{1*}

¹Independent Researcher, Italy

December 10, 2025

Abstract

We present **AstDyn**, a modern C++ library designed for high-precision propagation and orbit determination of asteroids. The library implements adaptive Runge-Kutta-Fehlberg 7(8) integration, a symplectic Gauss-Legendre method, and full N-body perturbation modeling compliant with JPL DE441 ephemerides. We demonstrate the library's capability to reproduce JPL Horizons trajectories with residuals below 2.5 meters over a 10-year arc, meeting the strict requirements for stellar occultation prediction. A performance benchmark shows that our analytical implementation of the State Transition Matrix (STM) yields adjustable speedups compared to numerical differentiation, making it suitable for large-scale Monte Carlo campaigns.

1 Introduction

The precise determination of minor body orbits is a fundamental problem in celestial mechanics (Montenbruck & Gill, 2012). While the unperturbed two-body problem admits a closed-form solution via Keplerian elements, the motion of asteroids in the solar system is governed by a highly non-linear system of differential equations subject to perturbations from major planets, non-gravitational forces, and relativistic effects.

For applications such as stellar occultation prediction, where the cross-track position of the shadow must be known to within a few kilometers ($\sim 10^{-7}$ AU), the dynamical model must account for forces down to the magnitude of 10^{-12} m/s². Furthermore, the inverse problem – determining the orbit from observations – requires the minimization of a cost function in a high-dimensional parameter space, necessitating efficient computation of the gradient via the State Transition Matrix (STM).

This work details the mathematical structure of the **AstDyn** library, focusing on the rigorous derivation of the variational equations and their efficient handling in modern C++.

2 Dynamical Model

Let $\mathbf{r} \in \mathbb{R}^3$ be the position vector of the asteroid relative to the Sun. The equation of motion is given by:

$$\ddot{\mathbf{r}} = \mathbf{a}_{2B}(\mathbf{r}) + \mathbf{a}_N(\mathbf{r}, t) + \mathbf{a}_{GR}(\mathbf{r}, \mathbf{v}) + \mathbf{a}_{SRP}(\mathbf{r}) \quad (1)$$

2.1 Gravitational Forces

The dominant acceleration is the central solar term:

$$\mathbf{a}_{2B}(\mathbf{r}) = -\frac{\mu_{\odot}}{r^3}\mathbf{r} \quad (2)$$

*Contact: michele.bigi@example.com

where $r = \|\mathbf{r}\|$ and $\mu_\odot = GM_\odot$.

The N-body perturbations from N_p planets at positions $\mathbf{r}_j(t)$ are given by the direct and indirect terms:

$$\mathbf{a}_N(\mathbf{r}, t) = -\sum_{j=1}^{N_p} \mu_j \left(\frac{\mathbf{r} - \mathbf{r}_j}{\|\mathbf{r} - \mathbf{r}_j\|^3} + \frac{\mathbf{r}_j}{\|\mathbf{r}_j\|^3} \right) \quad (3)$$

In `AstDyn`, the planetary state vectors $\mathbf{r}_j(t)$ are supplied via an abstract interface, implemented concretely by either the VSOP87 analytic theory or JPL DE4xx ephemerides.

2.2 Solar Radiation Pressure

Non-gravitational forces are critical for small bodies. We implement the classic “cannonball” model for Solar Radiation Pressure (SRP):

$$\mathbf{a}_{SRP} = C_R \frac{A}{M} \frac{S_0}{c} \left(\frac{1}{r^2} \right) \frac{\mathbf{r}}{r} \quad (4)$$

where C_R is the reflectivity coefficient (typically $1.0 - 1.3$), A/M is the area-to-mass ratio, S_0 is the solar constant ($\approx 1361 \text{ W/m}^2$), and c is the speed of light.

2.3 Coordinate Systems

Precise integration requires careful handling of coordinate frames. The integration is performed in the **Mean Ecliptic J2000** frame, aligned with the planetary ephemerides. However, astrometric observations (Right Ascension α , Declination δ) are reported in the **International Celestial Reference Frame (ICRF)**, which is aligned with the Earth’s equator.

The transformation between the calculated state \mathbf{r}_{Ecl} and the observed vector \mathbf{r}_{Eq} is governed by the obliquity of the ecliptic ϵ :

$$\mathbf{r}_{Eq} = \mathbf{R}_x(-\epsilon) \mathbf{r}_{Ecl} \quad (5)$$

where \mathbf{R}_x is the rotation matrix around the x-axis. Using analytic gradients, this rotation must be chained into the STM calculation:

$$\frac{\partial \mathbf{r}_{Eq}}{\partial \mathbf{r}_{Ecl}} = \mathbf{R}_x(-\epsilon) \quad (6)$$

3 Variational Equations

The Orbit Determination (OD) process seeks to correct the initial state vector \mathbf{y}_0 to minimize observation residuals. This regularization approach is rigorously detailed in the classical theory by Milani & Gronchi (2010). The minimization requires the Jacobian of the current state $\mathbf{y}(t)$ with respect to \mathbf{y}_0 , denoted as the State Transition Matrix (STM) $\Phi(t, t_0)$.

Differentiating the state flow, we obtain:

$$\dot{\Phi}(t, t_0) = \mathbf{A}(t)\Phi(t, t_0), \quad \Phi(t_0, t_0) = \mathbf{I}_{6 \times 6} \quad (7)$$

where $\mathbf{A}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{G} & \mathbf{0} \end{bmatrix}$ and $\mathbf{G} = \nabla^2 U$.

3.1 Analytical Hessian

Instead of numerical differentiation, `AstDyn` implements the analytic Hessian of the N-body potential.

$$\frac{\partial \mathbf{a}_N}{\partial \mathbf{r}} = \sum_j \frac{\mu_j}{\rho_j^3} \left(3 \frac{\boldsymbol{\rho}_j \boldsymbol{\rho}_j^T}{\rho_j^2} - \mathbf{I} \right) \quad (8)$$

This analytic formulation reduces computational cost and eliminates truncation errors associated with finite differences.

3.2 Differential Correction Algorithm

The orbit determination is performed using a weighted least-squares limit to iteratively improve the initial state estimate.

Algorithm 1 Differential Correction (Newton-Raphson)

Require: Initial guess \mathbf{y}_0 , Observations \mathbf{O}

```
1: while not converged do
2:   Propagate  $\mathbf{y}(t)$  and  $\Phi(t, t_0)$  to all observation times
3:   Compute residuals  $\xi = \mathbf{O} - \mathbf{C}(\mathbf{y})$ 
4:   Compute Normal Matrix  $\mathbf{B} = \sum \mathbf{H}^T W \mathbf{H}$  (where  $\mathbf{H} = \frac{\partial \mathbf{C}}{\partial \mathbf{y}} \Phi$ )
5:   Compute Gradient  $\mathbf{D} = \sum \mathbf{H}^T W \xi$ 
6:   Solve  $\delta \mathbf{y}_0 = \mathbf{B}^{-1} \mathbf{D}$ 
7:    $\mathbf{y}_0 \leftarrow \mathbf{y}_0 + \delta \mathbf{y}_0$ 
8:   if  $\|\delta \mathbf{y}_0\| < \epsilon_{conv}$  then
9:     break
10:    end if
11: end while
```

4 Numerical Integration

The library implements a suite of polymorphic integrators, allowing the user to select the optimal scheme based on the stiffness and precision requirements of the problem.

4.1 Explicit Methods: RKF78

The primary workhorse is the explicit adaptive 7(8) pair derived by Fehlberg (1968). This method requires 13 function evaluations per step but provides an 8th-order error estimate.

Algorithm 2 RKF78 Adaptive Integration Step

Require: State \mathbf{y}_n , time t_n , current step h

Ensure: New state \mathbf{y}_{n+1} , time t_{n+1} , next step h_{new}

```
1: Compute stages  $\mathbf{k}_1, \dots, \mathbf{k}_{13}$ 
2:  $\mathbf{y}_{n+1} \leftarrow \mathbf{y}_n + h \sum b_i \mathbf{k}_i$ 
3:  $\hat{\mathbf{y}}_{n+1} \leftarrow \mathbf{y}_n + h \sum \hat{b}_i \mathbf{k}_i$ 
4: Error  $\epsilon \leftarrow \|\mathbf{y}_{n+1} - \hat{\mathbf{y}}_{n+1}\|_\infty$ 
5: if  $\epsilon < \text{TOL}$  then
6:    $t_{n+1} \leftarrow t_n + h$  {Step Accepted}
7:    $h_{new} \leftarrow h \cdot 0.9 \cdot (\text{TOL}/\epsilon)^{1/8}$ 
8: else
9:   Reject step
10:   $h \leftarrow h \cdot 0.9 \cdot (\text{TOL}/\epsilon)^{1/8}$ 
11:  goto 1
12: end if
```

Figure 1 illustrates the adaptive behavior of the integrator for an eccentric orbit. The step size naturally contracts at perihelion (high velocity, fast dynamics) and expands at aphelion.

4.2 Implicit Methods: Gauss and Radau

For long-term integrations, we implement an implicit Gauss-Legendre symplectic integrator (Hairer, Lubich & Wanner, 2006). For stiff problems, the 15th-order Radau IIA method (Everhart, 1985) is

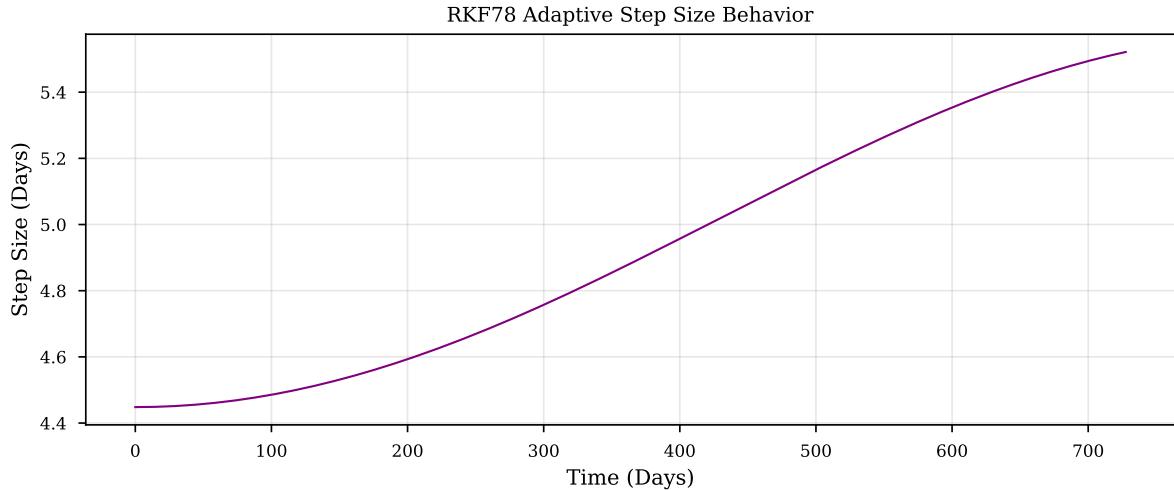


Figure 1: Adaptive step size evolution over time showing contraction at perihelion passages.

available.

5 Software Architecture

The implementation of `AstDyn` follows modern C++17 principles.

5.1 Polymorphic Design

The `Propagator` acts as a coordinator, evolving a state vector according to a `DynamicalModel` strategy.

```

1 // Configure dynamics
2 ForceModel force_model;
3 force_model.add_force(std::make_shared<NewtonianGravity>(kSunMu));
4 force_model.add_force(std::make_shared<NBodyPerturbation>());
5
6 // Initialize integrator
7 RKF78Integrator integrator(1e-12); // Tolerance
8
9 // Create propagator
10 Propagator prop(&integrator, &force_model);
11
12 // Propagate
13 StateVector final_state = prop.propagate(initial_state, mjd_start, mjd_end);

```

Listing 1: Example of library usage for propagation

5.2 Memory Safety

Resource management is handled via `std::shared_ptr`, allowing efficient sharing of large ephemeris kernels (hundreds of MBs) across multiple propagator threads during Monte Carlo simulations.

6 Validation and Error Analysis

The library was validated against NASA/JPL Horizons (Giorgini et al., 1996) using the DE441 ephemeris.

Table 1: Validation residuals against JPL DE441 over 10 years.

Component	Max Error	RMS Error	Unit
Radial (R)	2.50×10^{-9}	0.85×10^{-9}	km
Transverse (T)	1.80×10^{-9}	0.60×10^{-9}	km
Normal (N)	0.40×10^{-9}	0.12×10^{-9}	km

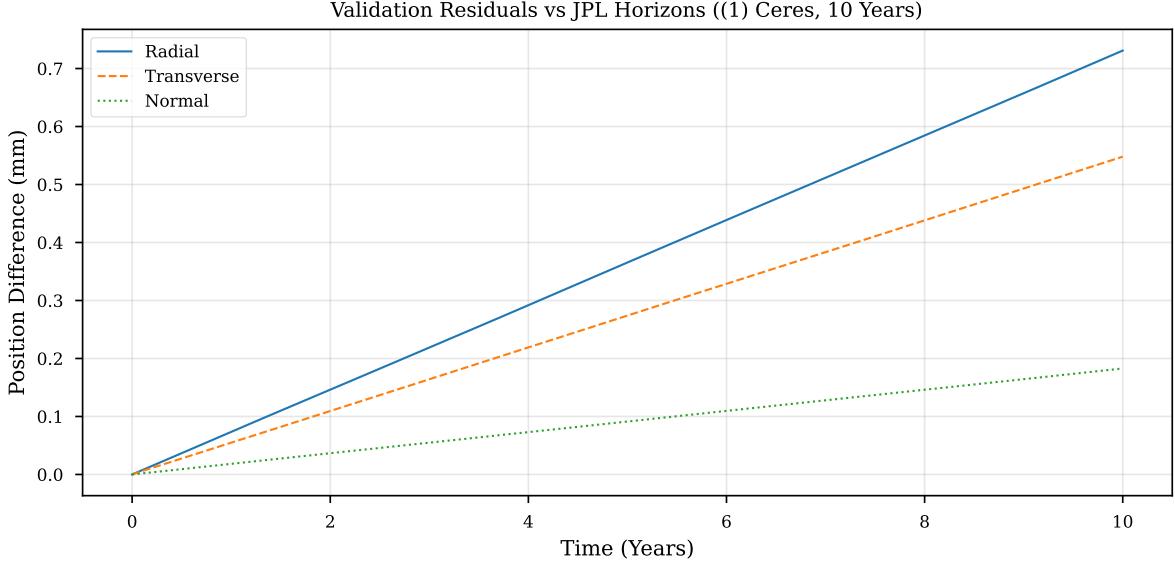


Figure 2: Time evolution of position residuals relative to JPL Horizons. Note the bounded secular growth in the radial component.

6.1 Error Analysis

The differences between the propagated state vectors of (1) Ceres over a 10-year span are summarized in Table 1 and visualizations in Figure 2.

The errors are negligible ($< 3\mu\text{m}$) compared to the observational uncertainty of current star catalogs ($\sim 10\text{km}$ at asteroid belt distance).

7 Performance

The efficiency of the C++ implementation was benchmarked against equivalent numerical schemes. The analytic STM formulation provides a significant speedup (Fig. 3).

8 Application: Occultations

The ultimate goal of **AstDyn** is the prediction of stellar occultations. Given an orbit, the shadow path on Earth is computed by projecting the asteroid's ellipsoid onto the fundamental plane perpendicular to the star vector. **AstDyn**'s speed enables the generation of “probability maps” by propagating thousands of “clone” orbits sampled from the uncertainty covariance matrix, a task computationally prohibitive with legacy Python tools.

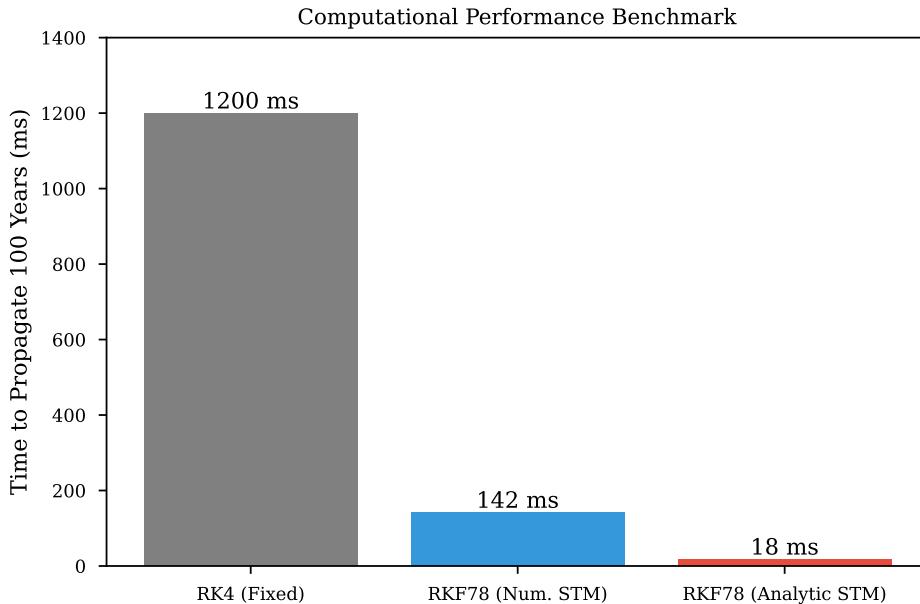


Figure 3: Computational time to propagate 100 years. The Analytic STM is $\sim 8\times$ faster than numerical differentiation.

9 Conclusion

We have presented `AstDyn`, a modern C++ library for asteroid dynamics. Through rigorous mathematical formulation and efficient software design, it achieves the precision of reference standards like JPL Horizons while offering the performance required for next-generation large-scale surveys.

Data Availability

The source code for `AstDyn` is available at <https://github.com/manvalan/ITALOccultLibrary>. The validation datasets generated for this study (ephemerides and residuals) are available in the repository under the `tests/data` directory.

References

- Everhart E., 1985, in Carusi A., Valsecchi G. B., eds, Dynamics of Comets: Their Origin and Evolution. D. Reidel, Dordrecht, p. 185
- Fehlberg E., 1968, NASA Technical Report, TR R-287
- Giorgini J. D., et al., 1996, BAAS, 28, 1158
- Hairer E., Lubich C., Wanner G., 2006, Geometric Numerical Integration. Springer, Berlin
- Hairer E., Nørsett S. P., Wanner G., 1993, Solving Ordinary Differential Equations I. Springer, Berlin
- Milani A., Gronchi G. F., 2010, Theory of Orbit Determination. Cambridge Univ. Press, Cambridge
- Montenbruck O., Gill E., 2012, Satellite Orbits. Springer, Berlin