Examining Propagator's Integration Methods Towards the Development of Autonomous Situational Awareness

Elad Denenberg[*]
Braude Academic College of Engineering, Karmiel, 2161002, Israel

The accurate determination of the Time of Closest Approach (TCA) between satellites is critical for space situational awareness, amongst many other applications. Recently a new method for the computation of the TCA named Conjunction Assessment Through Chebyshev Polynomials (CATCH) was introduced. CATCH has been shown to be both fast and accurate. However, it requires sampling the satellites' states at specific time points, which could require additional computing. This paper examines the computational cost and accuracy of five common integrators. Though the amount of tests is limited, the results suggest that the cost is not high, and therefore CATCH has the potential to be implemented onboard.

I Introduction

In recent years, there has been a push to develop more efficient methods for calculating the TCA between satellites [1,2]. TCA is the instant at which two satellites come closest to each other in their respective orbits, and accurate knowledge of TCA is essential for various applications, including collision avoidance and formation flying.

Two methods for computing the TCA are most common: The first is propagating the orbit of the two objects with a very small time step, and the second is Alfano Negron Close Approach Software (ANCAS) [1]. Propagating with a small time step can be very accurate. However, it is extremely expensive in computation. ANCAS samples the state of the satellite at constant time intervals. Between every four points ANCAS fits a cubic polynomial to the relative velocity. The roots of the polynomial are approximations of the TCA (or time of maximal distance). This method is very fast and computationally cheap. However, it introduces approximation errors.

Recently a novel method for finding TCA, called CATCH (short for Conjunction Assessment Through Chebyshev Polynomials), was proposed [3]. This method uses Chebyshev Proxy Polynomials (CPP) to approximate the relative velocity, then uses Boyd's method to find the roots [4]. To do so it requires sampling the positions of the satellites at specific points: the Gauss-Lobatto nodes (GLN) of a given section size rather than at constant time intervals. The main advantage of this approach is that it allows for more efficient use of computational resources, as the number of samples can be reduced without sacrificing accuracy. However, sampling the required points must be done using a propagator module, and such propagation's performance costs are yet to be examined.

This study focuses on the propagator module used to sample the satellites' positions at the GLN as part of an attempt to implement the CATCH method. Specifically, it

[*]Lecturer, Mechanical Engineering, EladD@braude.ac.il.

examines the runtime and accuracy costs of tasking five common integrators to sample at the required points.

Much work has been done in examining the performance of different integrators over the years (e.g., [5, 6] are but a couple). This paper differs from other works as it aims to establish the accuracy specifically at the time points required for computing the TCA with CATCH, and the additional computational load when requiring high accuracy at these time points.

The main goal of this paper is to examine the possibility that what we gain in accuracy and speed when using CATCH we lose by sampling specifically at the GLN. Therefore, we will examine the computation load required to propagate to these specific points. In addition, we will examine the accuracy around these points when the propagation is done without the GLN sampling requirement. This work is a first step in understanding the performance of a propagator module that is required to work with CATCH in an autonomous system and the performance costs it may incur.

II Background

II.A CATCH

Finding the TCA between two orbiting objects can be described as finding a solution to the following minimization problem:

$$J = \min_{t} \left\| \boldsymbol{\rho}\left(t\right) \right\| \tag{1}$$

where $\boldsymbol{\rho} = \mathbf{r}_1 - \mathbf{r}_2$ is the relative position, $\mathbf{r}_i$ being the position of object i.

Assuming we know the initial state of both objects at $t_0 = 0$: $\mathbf{r}_1\left(0\right)$, $\mathbf{v}_1\left(0\right) = \dot{\mathbf{r}}_1\left(0\right)$, $\mathbf{r}_2\left(0\right)$, $\mathbf{v}_2\left(0\right) = \dot{\mathbf{r}}_2\left(0\right)$. And assuming we have a propagator with which we can compute the state of both objects at any given time t, both CATCH and ANCAS define a distance function thus:

$$f\left(t\right) = \boldsymbol{\rho}\left(t\right) \cdot \boldsymbol{\rho}\left(t\right) \tag{2}$$

The first and second derivatives of the distance function would be:

$$\dot{f}\left(t\right) = 2\dot{\boldsymbol{\rho}} \cdot \boldsymbol{\rho} \tag{3}$$

$$\ddot{f}\left(t\right) = 2\left(\ddot{\boldsymbol{\rho}} \cdot \boldsymbol{\rho} + \dot{\boldsymbol{\rho}} \cdot \dot{\boldsymbol{\rho}}\right) \tag{4}$$

And the solution to Eq. (1) would be found when:

$$\dot{f}\left(t^*\right) = 0 \tag{5a}$$

$$\ddot{f}\left(t^*\right) > 0 \tag{5b}$$

To find the roots of the distance function ANCAS uses cubic polynomials to approximate the first derivative, then find the roots of said polynomials. In contrast, CATCH uses Boyd's CPP method [4]: First, the time in which we seek the minimum is sliced into intervals of size Γ:

$$\Gamma = \frac{T_{min}}{2} \tag{6}$$

Where $Tmin = \min(T_1, T_2)$, and T_i is the orbital period of object i.

Nexy a chebyshev polynomial of this sort is fitted to the first derivative of the distance

$$\dot{f} \approx \dot{f}_N(t) = \sum_{j=0}^{N} a_j T_j \left(\frac{2t - q\Gamma}{\Gamma}\right) \tag{7}$$

Where $q = 0 \ldots {}^{t_{max}}/_{\Gamma}$ is the section number, and T_j is

$$T_j(x) = \cos(j \arccos(x)) \tag{8}$$

To fit the polynomial, we sample at the GLN:

$$t_j = \frac{\Gamma}{2}\left(1 - \cos\left(\pi \frac{j}{N}\right)\right) + q\Gamma \tag{9}$$

Where N is the number of points sampled per section, in [3], it is shown that $N = 16$ is adequate in most practical cases.

Finally, the roots are computed by finding the eigenvalues of the companion matrix. The full description of this stage can be found in [3] and [4]. Using the interval Γ as described here, with $N = 16$, has been proven to generate a companion matrix small enough to be decomposed with a small computational price while achieving great accuracy in the TCA and also in the approximation of the distance.

At the beginning of this subsection, we assumed we have a propagator that enables us to sample the state of the orbiting objects at any time point we desire. In the next subsection, we will discuss the propagator module and its influence on the accuracy and computational load in relation to finding the TCA.

II.B Propagators

An orbital propagator is a mathematical model used to predict the position and velocity of a satellite over time. There are two main types of orbital propagators: analytical and numerical. Analytical propagators are based on simplified mathematical models and are fast and efficient but limited in accuracy. Numerical propagators use numerical methods to simulate the motion of a satellite and are more flexible and accurate but computationally intensive.

In [3], Denenberg uses SGP4 [7] to propagate satellites. SGP4 is an analytic propagator from the Simplified General Perturbations (SGP) family. It solves a simplified and estimated model of the forces that act on the satellite and is accurate to within two weeks of the epoch. If CATCH is to be implemented and used in real life, it must operate with an accurate numerical operator.

In this work, the keplerian orbit was used as a force model. Therefore, the only force operating on the satellite was gravity. The model is given as

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{10}$$

Where μ is the standard gravitational parameter and $\mathbf{r}$ is the position of the satellite.

This Keplerian dynamics can be solved by switching to Keplerian orbital elements. However, the solution will be numerical as it would require an iterative solution to Kepler's equation. In this work, we solved the problem with high-accuracy numerical integrators.

The integrators that are most used in propagators can be divided into three types: Fixed step, variable step, and multi-step. The family of Runge-Kutta (RK) integrators are perhaps the most famous fixed step integrators and most widely used. These methods generally use a fixed time step h. In each step, a Taylor expansion of order p is performed [8].

The variable step methods use two expansions to adjust the time step adaptively. The adaptive step is very efficient in estimating functions that require smaller steps in some areas and larger ones in others. However, these methods require selecting an initial step size. Multi-step integrators use information from previous steps (not only the latest one) to approximate the next step. In this work, we tested a few integrators of each type.

III Methods

A series of simulations using a basic test case was conducted to evaluate the performance of propagator modules. This test case was a satellite in Low Earth Orbit (LEO) orbit that was propagated for 365 days. The force model we chose was the simple Keplerian two-body problem (i.e., no perturbations), as a true baseline can easily be found and compared for such an orbit. The satellite was subject to keplerian dynamics.

We found the baseline using numerical integration with a limited step size and a requirement to reach high accuracy. Indeed the force model chosen is simplistic and can be solved using Kepler's equation. However, Kepler's equation requires a numerical solution and, at certain points, would be highly inaccurate or fail to converge. Thus we used Verner's method [9] (MATLAB's ODE78), with a time step of $T/100$, where T is the satellite's orbital period. The tolerance required was 1e-15km.

We used five different integrators to test the accuracy at the GLN. Two fixed-step integrators: 4th-order Runge-Kutta (RK4) and 8th-order Runge-Kutta (RK8). Two variable step integrators: The Dormand-Prince method [10] (MATLAB's ODE45) and Verner's method [9] (MATLAB's ODE78). The multi-step integrator used was the Adams-Bashforth-Moulton method (MATLAB's ODE113).

We measured the runtime and accuracy around the GLN of each integrator in two use cases. The first was: Integrators were given a fixed or maximal step but were not required to evaluate the state of the satellite at the GLN. After propagation, the satellite state was estimated by interpolation using a weighted average.

For each time point τ in the GLN, two nearest points t_n and t_{n+1}, which the integrator chose to compute, were selected. The estimate of the position was done as follows: First, the weights were calculated thus:

$$w_1 = 1 - \frac{\tau - t_n}{t_{n-1} - t_n} \tag{11}$$

$$w_2 = 1 - w_1 \tag{12}$$

Then the average of the position was computed

$$\tilde{\mathbf{r}(\tau)} = w_1 \mathbf{r}_n + w_2 \mathbf{r}_{n+1} \tag{13}$$

In the second use case, the required points were explicitly evaluated, either by additional interpolation by the variable step integrators or by changing the size of the step h for each iteration for the fixed step integrators. At each step, the next step size was determined using the conversion:

$$t_j = \frac{\Gamma}{2}\left(1 - \cos\left(\pi \frac{j}{N}\right)\right) + q\Gamma \tag{14}$$

Where t is the time, $j = 0, 1, ...N$, N the number of points per interval, Γ the interval, and q the interval number.

The performance in terms of accuracy and runtime of both cases is compared and brought here.

IV Results

The tests were performed on an Intel i7-8550U CPU @ 1.80GHz×8. We used Matlab native ODE solvers (ODE45, ODE78, and ODE113) and implementations of RK4 and RK8. The satellite propagated was ONEWEB-0615; its data was obtained through publicly published Two Line Elements (TLE). The initial position and velocity are given in Table 1.

Table 1: Test-satellite State

	x	y	z
r (km)	4730.65	-1771.29	-4806.24
v (km/s)	-4.64	2.41	-5.46

The runtime results are shown in Table 2. The first column describes the case in which the GLN were explicitly required and defined for the integrator. In case of the RK integrators, this was done by adjusting the step size after each expansion. The adaptable step used variable step size to sample at the required points. The second column is the run time required to compute the state at the GLN using the weighted average. The third column is the time required simply for the computation of the weighted average alone.

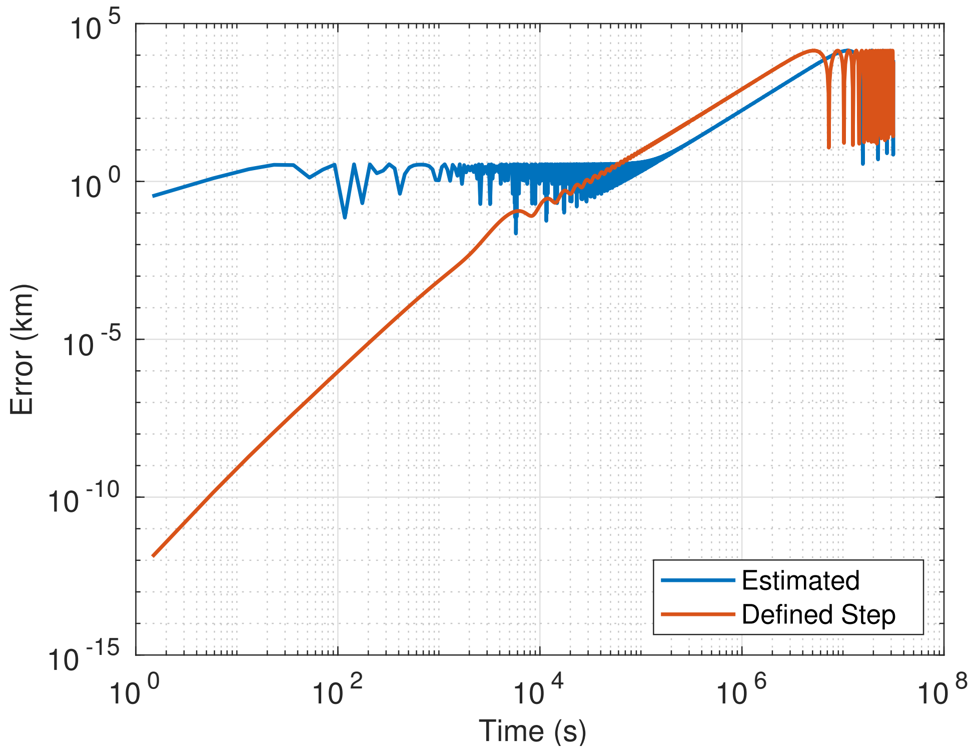
Unsurprisingly, the lower the order, the faster the integrator. Fixed-step integrators are mostly slightly faster. Since the same number of steps were kept, integrating with a fixed step and then estimating the position at the GLN proved to slow the fixed-step integrators significantly.

The time it took to estimate the weighted average is the same for all integrators, allowing the variable step integrators and the multi-step to compute much faster. Requiring the variable step integrators to interpolate the required points may slow them down in some cases, far more than the time required to estimate the points after integration. However, estimating the points in this way carries a cost in accuracy.

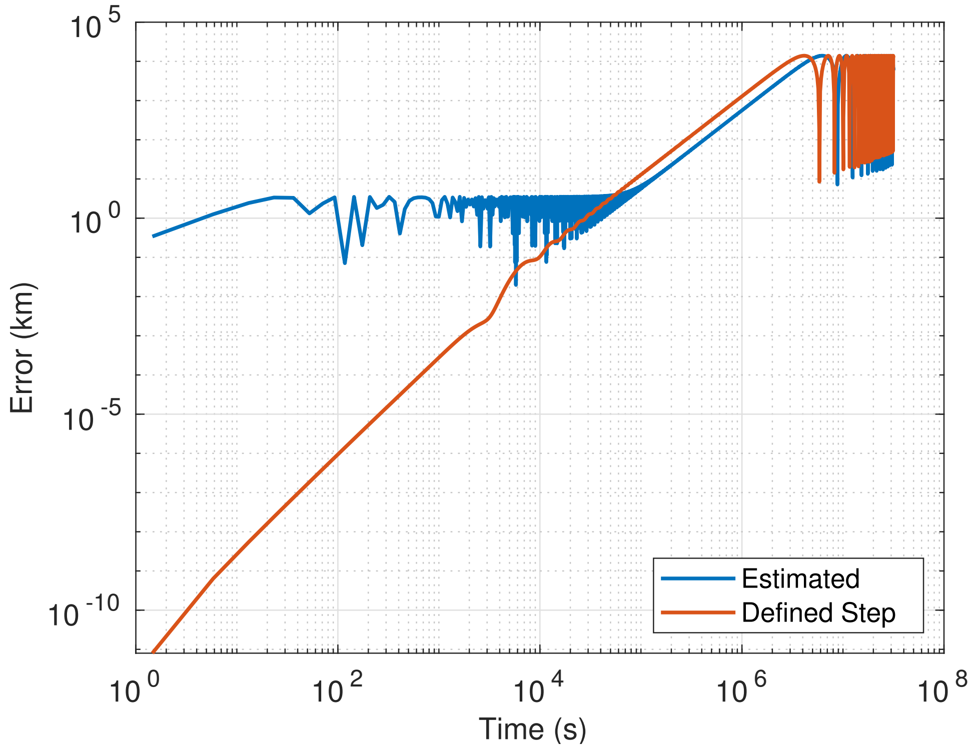
Table 2: Runtime of different methods (s)

Method	Defined Step	Estimated Tot.	Weighted average
RK4	4.33	5.27	1.39
RK8	12.26	11.86	1.36
ODE45	4.97	2.14	1.37
ODE78	8.26	1.81	1.37
ODE113	8.89	1.75	1.37

The comparison of accuracy between the two examined cases is shown in Fig 1, Fig 2,

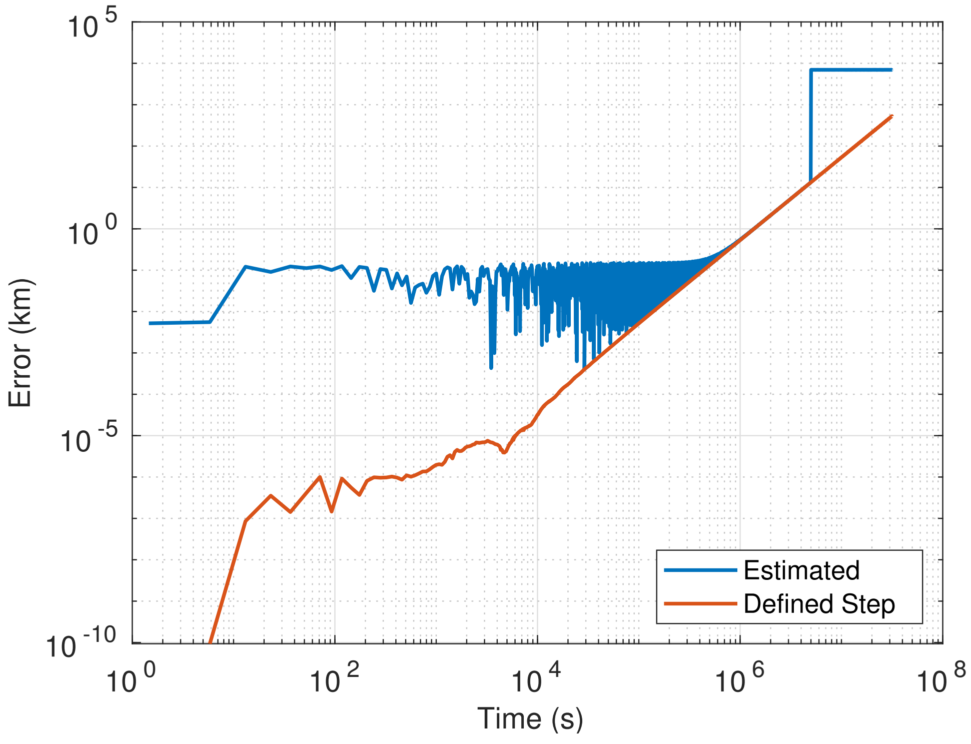


(a) RK4

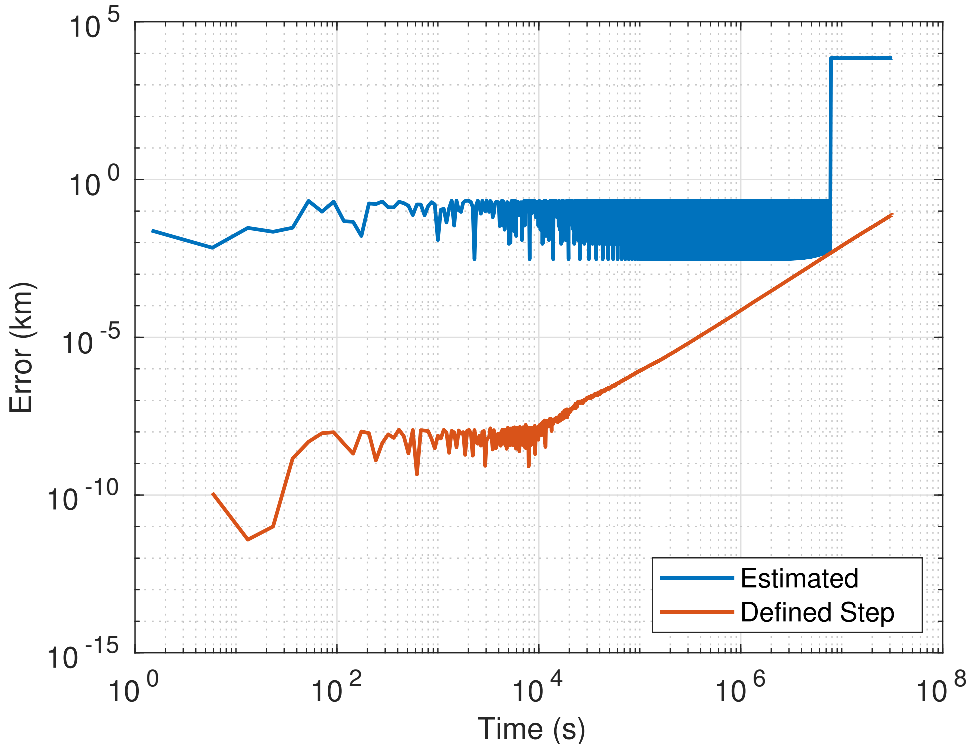


(b) RK8

Figure 1: Position Error of the Fixed Step Integrators



(a) ODE45



(b) ODE78

Figure 2: Position Error of the Variable Step Integrators

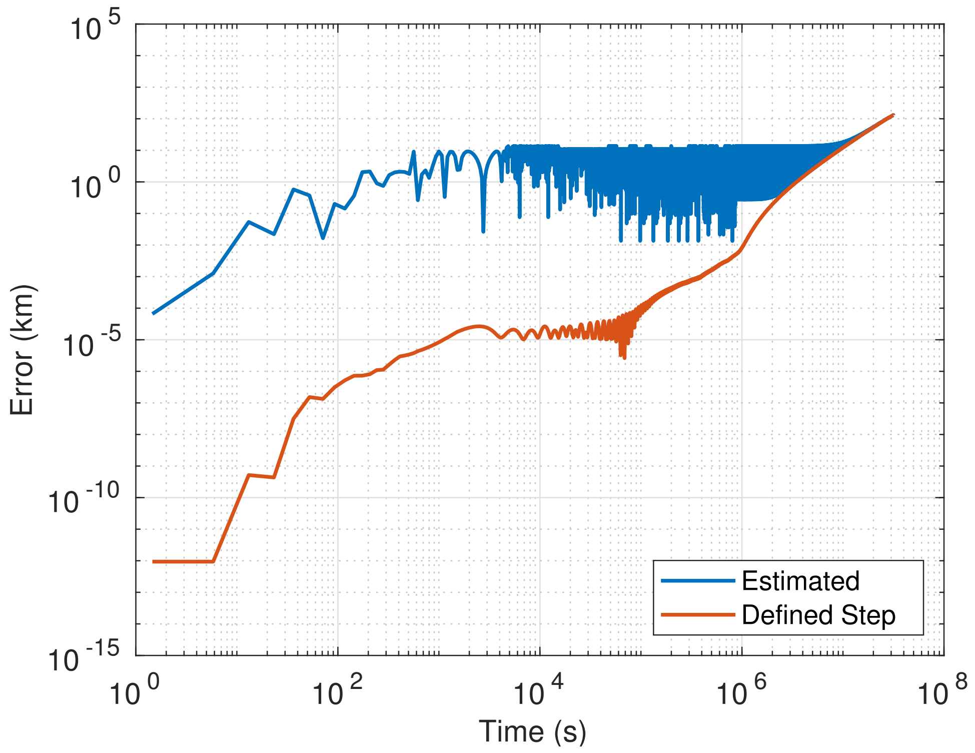


Figure 3: ODE113

Figure 4: Position Error of the Multi-Step Integrator

and Fig 4. The figures show the error between the computed baseline and the integrator's result. specifically at the GLN points. In blue is the weighted average, and in orange is the integration with the points defined explicitly.

In general, estimating the satellite's position after integration is less accurate by several orders of magnitude. The accuracy achieved by using the weighted average was of the same order of magnitude as the position itself: in kilometers. However, the accuracy decays over time when integrating to the GLN points.

The performance of the different integrators is given in Fig 5. The second case is shown: integrating through the desired points. There is not a significant difference between the methods. Typically, variable step methods outperform the fixed step ones, as was to be expected.

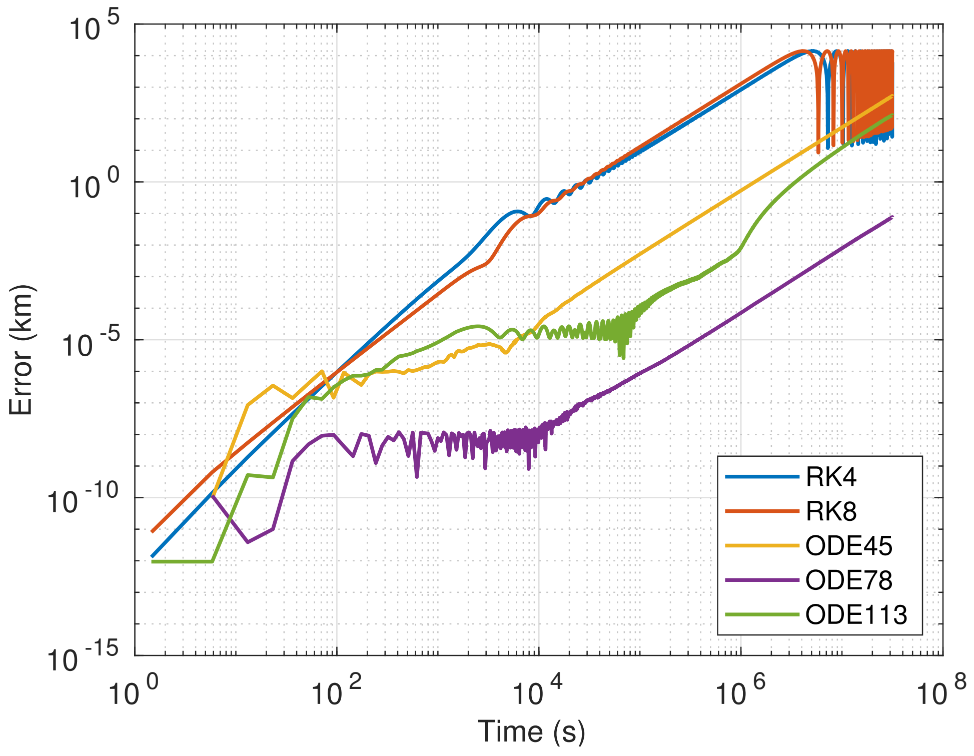


Figure 5: Position Error of all five methods

V Discussion

The accuracy results suggest that propagators require adjustments should they be used with CATCH. If a fixed step is used, the step would have to be stretched to match the GLN. If a variable step integrator is used, it would have to evaluate the state at the desired points explicitly.

The results are promising. The price of this adjustment in terms of CPU time is not great. This indicates that CATCH has the potential to be implemented onboard a satellite and enable autonomous situational awareness.

This work is limited, however. Only one satellite was examined, and only under a simple force model. Future work would include satellites in Geostationary Orbit (GEO), Moderate Elliptic Orbit (MEO), and High Elliptical Orbit (HEO), as well as different force models.

References

[1] Alfano, S., "Determining Satellite Close Approaches, Part II," *The Journal of the Astronautical Sciences*, Vol. 42, No. 2, 1994, pp. 143–152.

[2] Denenberg, E. and Gurfil, P., "Improvements to Time of Closest Approach Calculation," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 9, 2016, pp. 1967–1979.

[3] Denenberg, E., "Satellite Closest Approach Calculation Through Chebyshev Proxy Polynomials," *Acta Astronautica*, Vol. 170, 2020, pp. 55–65.

[4] Boyd, J. P., "Finding the zeros of a univariate equation: proxy rootfinders, Chebyshev interpolation, and the companion matrix," *SIAM review*, Vol. 55, No. 2, 2013, pp. 375–396.

[5] Atallah, A. M., Woollands, R. M., Elgohary, T. A., and Junkins, J. L., "Accuracy and efficiency comparison of six numerical integrators for propagating perturbed orbits," *The Journal of the Astronautical Sciences*, Vol. 67, 2020, pp. 511–538.

[6] Papanikolaou, T. D. and Tsoulis, D., "Assessment of numerical integration methods in the context of low Earth orbits and inter-satellite observation analysis," *Acta Geodaetica et Geophysica*, Vol. 51, No. 4, 2016, pp. 619–641.

[7] Hoots, F. R., Roehrich, R. L., and Kelso, T., "Spacetrack report no. 3," *Project Spacetrack Reports, Office of Astrodynamics, Aerospace Defense Center, ADC/DO6, Peterson AFB, CO*, Vol. 80914, 1980, p. 14.

[8] Butcher, J. C., *Numerical methods for ordinary differential equations*, John Wiley & Sons, pp. 97–110, 3rd ed., 2016.

[9] Verner, J. H., "Numerically optimal Runge–Kutta pairs with interpolants," *Numerical Algorithms*, Vol. 53, No. 2, 2010, pp. 383–396.

[10] Dormand, J. and Prince, P., "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, Vol. 6, No. 1, 1980, pp. 19–26.