Accurate numerical orbit propagation using Polynomial Algebra Computational Engine PACE

Emmanuel Bignon [(1)], Pierre mercier [(1)], Vincent Azzopardi [(1)], Romain Pinède [(1)]

ISSFD 2015 congress, Germany

Dated: September 14, 2015

Keywords: *Taylor differential algebra (TDA), Orbit propagation, Monte-Carlo*

Abstract

Space mechanics problems are extremely sensitive to uncertainties management: small differences on initial conditions can lead to large errors on a predicted phenomenon such as orbit propagation. To cover the whole range of possible trajectories of a given object, one may analyze the evolution of the covariance matrix, which is not very accurate for highly non-linear systems, or use statistical methods like Monte Carlo methods for numerical integrations, which are very time-consuming.

A new way to deal with uncertainties on orbit propagation is the use of Taylor Differential Algebra. Research in this field has been carried out over the past 20 years, including studies on Apophis asteroid close encounters, interplanetary transfers, or particle accelerators [1][2]. A single integration using Taylor Differential Algebra will propagate the reference initial state x_0 along with a full neighborhood of states around x_0 whereas a usual integration will only propagate x_0. This method has several advantages. It reduces the study of uncertainties to a single propagation, decreasing the computation time with respect to Monte-Carlo methods for which many integrations are needed. It also provides an analytical result which possesses some interesting properties and can be manipulated, while usual methods are statistical and only give numerical results.

The purpose of the present study is to implement a full propagator, for any Earth-orbiting objects using Thales Polynomial Algebra Computational Engine PACE. This leads to carefully and accurately model in Taylor Algebra main perturbations such as high order Earth zonal and tesseral potential, Sun/Moon perturbations, Earth atmospheric drag, solar radiation pressure, etc.

The theory requires functions to have specific properties but we show that it is possible to extend the range of available functions to include discontinuities or some piecewise functions.

This propagator has been heavily validated against Thales reference numerical propagator to operate in industrial applications. Even for complex force models, our implementation of a 5^{th} order TDA orbit propagation runs as fast as two hundred classical propagations. Even if some limitations still remain concerning intermediate evaluations, and improvements are still achievable on computation times, the Taylor Differential Algebra is under way to become a classical and powerful tool for space mechanics applications.

References

[1] R. Armellin. Asteroid close encounters characterization using differential algebra: the case of Apophis, 2010.
[2] M. Berz. From Taylor series to Taylor models, 1997.

---

(1) THALES SIX/CIC/CS3 Aerospace & Science Engineering department,
3, avenue de l'Europe 31400 Toulouse, France, 33562887553, pierre.mercier@thalesgroup.com

Introduction

Many space mechanics problems involve dealing with uncertainties since many parameters are often not known accurately. Some examples are the position of non-cooperative debris, spacecraft parameters such as the frontal area or the drag coefficient. As a result, some numerical methods have been set up in order to circumvent these problems and to be able to find the most adequate solution.

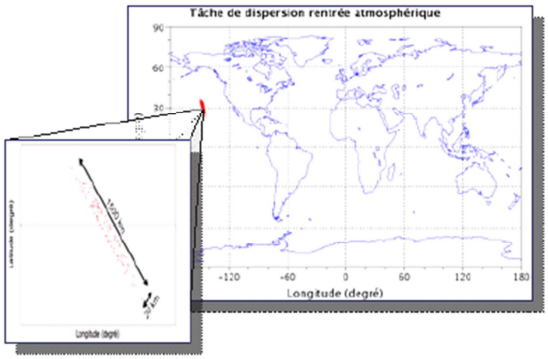


Figure 1: Reentry dispersion area.

The first method usually used is the Monte-Carlo method: a large amount of sets of initial conditions is defined using dispersions representative of the problem (given some standard deviations or covariance matrices for instance) and for each initial condition, the process is performed and the result is computed. In case of orbit propagation, the final state is the result of the propagated initial state.
This method has the advantages of being simple to set up, since the computation core remains unchanged, and accurate, since the resulting set accurately describes the initial set of uncertainties. However this method has the major disadvantages of being quite time-consuming since the process needs to be repeated thousands of times, and being purely numerical since the resulting set is purely statistical.

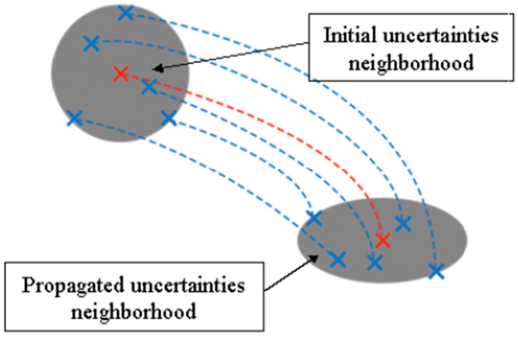


Figure 2: Monte-Carlo process.

The second common method is to compute the covariance matrix – or state transition matrix $dX/dX0$ – of the final state. This matrix describes the behavior of the solution around a reference point (in red in the above picture) to the first order. This method has the advantage of being fast and analytical. It has however several disadvantages: unless computed by finite differences, the computation core of the process needs to be rewritten to account for partial derivatives, and this is only a first order approximation whose accuracy quickly drops off for highly non-linear system which is the case of many space mechanics problems such as low-Earth-orbit propagation or spacecraft reentry.

In the following pages, we describe the use of Taylor Differential Algebra to deal with uncertainties particularly for orbit propagation. A brief summary of the mathematical background of Taylor Algebra is first presented along with its implantation in Thales library PACE. Then we detail its

implementation in Thales orbit propagator JACK, some issues that arose and the solutions we came up with. The results we obtained are presented in the last section of this paper.

1. Taylor Differential Algebra

1.1. Mathematical background

Taylor Algebra and possibilities that came out find their roots in the works of Brook Taylor in 1715. His works led him to the following theorem:

Theorem (Taylor): *Let f be a function* $f:[a,b]\subset\Re\to\Re$ $(n+1)$ *times continuous and* n *times differentiable on* $[a,b]$. *Let* $x_0\in[a,b]$. *Then for any* $x\in[a,b]$, *we have:*

$$f(x)=\sum_{k=0}^{n}\frac{1}{k!}f^{k}(x_0)(x-x_0)^{k}+O\left((x-x_0)^{n+1}\right)$$

In other words, any smooth enough function can be locally approximated around a point x_0 by a polynomial whose coefficients only depend on n^{th} derivatives computed around x_0. This approximation can be as accurate as required, the order of the expansion setting the precision.

This theorem can be generalized to a multivariate functions:

$$f(\vec{x})=\sum_{k=0}^{n}\frac{1}{k!}((\vec{x}-\vec{x}_0).\vec{\nabla})^{k}\,f(\vec{x}_0)+O((\vec{x}-\vec{x}_0)^{n+1}) \quad (1)$$

With the operator $(\vec{h}.\vec{\nabla})^{k}$ defined as:

$$(\vec{h}.\vec{\nabla})^{k}=\sum_{\substack{0\le i_1,\ldots,i_v\le k\\ i_1+\cdots+i_v=k}}\frac{k!}{i_1!\ldots i_v!}h_1^{\,i_1}\ldots h_v^{\,i_v}\frac{\partial^{k}}{\partial x_1^{\,i_1}\ldots\partial x_v^{\,i_{v1}}}$$

Equation (1) splits any smooth function in a polynomial part and a remainder. As a result the polynomial part of is an approximation of the function *f*. This polynomial is entirely defined by:

- Its coefficients,
- Its order *n*,
- The number of variables *v*,
- Its center x_0.

There is no way to define an algebra on the ring of polynomials *K[X]*. However, an algebra can be defined if one considers only polynomials of a given order. Hence, the following algebra called "Taylor Differential Algebra" (TDA) can be defined:

- Elements: polynomials up to order *n* of *v* variables centered around a point x_0 with coefficients in *R*.
- Operations: usual operations on polynomials (addition, multiplication, multiplication by a scalar). The only difference with usual polynomials being that resulting polynomials are truncated to the algebra order *n*.

This algebra is often referred to ${}_nD_v$ in the literature.

Given *n* and *v*, the maximum number of coefficients of a polynomial belonging to ${}_nD_v$ is given by the formula:

$$\frac{(n+v)!}{n!v!}$$

This number also corresponds to the dimension of the algebra. As an example, ${}_5D_6$ is of dimension 462, ${}_{10}D_6$ is of dimension 8008.

Other operations can be defined on the algebra:

- Derivation ∂ and anti-derivation ∂^{-1}, hence Taylor Algebra is Differential,
- Complex functions such as *sin*, *cos*, *atan*, *exp*:

  These functions are simply computed using their Taylor series, which are composed of algebraic operations (+, ×), around x_0. For example with the exponential function:

$$\exp x = \sum_{k=0}^{+\infty} \frac{x^k}{k!}$$

  Given a polynomial P of ${}_nD_v$, we have:

$$\exp P = \sum_{k=0}^{n} \frac{P^k}{k!}$$

  Inverse of polynomial $\frac{1}{P}$ can also be computed using:

$$\frac{1}{1+P} = \sum_{k=0}^{n} (-1)^k P^k$$

Many more properties can be extracted from Taylor Differential Algebra: composition, inversion, algorithms for fixed-point problems, binary relation, etc. Extensive work about Taylor Algebra and rigorous proofs can be found in the literature[1][2][4].

In conclusion, it appears that any complex calculus involving algebraic operations can be performed in Taylor algebra.

1.2. Thales computational engine PACE

Thales has designed and implemented a mathematical library PACE (Polynomial Algebra Computation Engine) based on Taylor Differential Algebra. This library has been written in java and can thus be used in any computer environment. It is also fully generic and can be applied hence for **any physical problem dealing with uncertainties**.

The main constraint on the library is to be as fast as possible since all algebraic operations on polynomials are computationally intensive (see Table 1) and will be performed a huge amount of time.

The library possesses 4 levels, by increasing order of complexity (in a computer library sense):

- Algebra:

  The key to obtain a successful and efficient polynomial computation engine is to make every algebraic operation as efficient as possible. In PACE, the algebra part of the library is devoted to that task. Many articles deal with how to efficiently perform operations in such an algebra[3][5].

  The option we have chosen is to minimize the number of floating point operations when performing algebraic operations. Hence, once the algebra has been defined, some tables are pre-computed so that future algebraic operations on the algebra can be run very efficiently.
  Pre-computed tables are available for all operations: multiplication, derivation, anti-derivation, etc.

Example of precomputed table for multiplication: given two polynomials represented by their sets of coefficients [a_0,…., a_k], [b_0,…., b_l], the pre-computed table lists pairs of indices [k, l, m] involved in the multiplication operation. Hence, the multiplication operation is just a simple loop on this table to pick the right coefficients among the initial polynomials [a_0,…., a_k] and [b_0,…., b_l] to get the resulting polynomial [c_0,…., c_m].

- Polynomial (element of algebra):

For a given algebra, a polynomial is only composed of a list of its coefficients. The other parameters are settled by the algebra (order n, number of variables v and center x_0). All non-obvious operations rely on precomputed tables provided by the algebra to minimize computation cost. All algebraic operations are available as well as math operations (sin, cos, exp, etc.). Many on-the-side functions have been added, such as comparison, truncation, composition, evaluation, partial evaluation, etc.
On-top elements have also been created such as vectors of polynomials, allowing some specific operations such as inversion.

Here are the results obtained in ${}_6D_5$ with a classic 2.4GHz i5 desktop computer:

Operation	Computation time in ${}_5D_6$ (for 100 000 operations)
Addition	120ms
Multiplication by a constant	70ms
Multiplication	1100ms
Derivation	140ms
Polynomial evaluation	700ms

Table 1: computation times for Taylor algebra low-level operations.

One can see how important is to reduce computation cost of multiplication since it is has the biggest computation expense.

- Mathematical tools:

Mathematical tools include algorithms specific to Polynomial Algebra as well as classic mathematical algorithms converted into Taylor Algebra often used to solve physical problems. Two main algorithms are presented here: fixed-step integrators and fixed-point solvers.

Fixed-step integrators, particularly Runge-Kutta integrators are simply a linear combination of terms, as a result it is straightforward to convert it into Taylor Algebra since it involves only algebraic operations. PACE possesses a generic framework for integrating first order differential equations using Runge-Kutta integrators.

Fixed-point solvers require to solve equations of the form $x = f(x)$ and it is usually implemented using a loop and a convergence criterion to stop the loop. The convergence is usually a mere comparison between x and $f(x)$ and loop is stop is x is close enough to $f(x)$. In Taylor Algebra, this convergence criterion is hence replaced by the comparison of two polynomials P and $f(P)$. This comparison can be performed in many ways:

- Term by term comparison: equality is satisfied if all coefficients are close enough to each other (depending on a fixed threshold).
- Term by term comparison using only a subset of the coefficients. This method is faster than the comparison of all coefficients but must be used carefully. Indeed, speed of convergence of coefficients depends on their monomial order. Coefficients of higher order tend to converge more slowly than coefficients of lower order. Numerical quality issues are also at stake since coefficients of high order are partially absorbed by coefficients of lower orders. As a result accurate convergence on high order coefficients

is not required since the last digits in double precision are never taken into account when evaluating polynomials.
- Comparison using norms of polynomials. An example of norm is provided in [2].

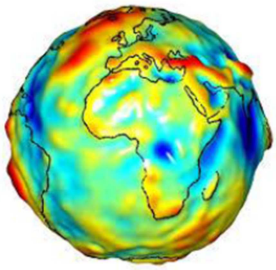
This library is hence generic and can be used for any on-top program dealing with uncertainties. In the following section, PACE is used for orbit propagation.

To use Taylor Algebra in a computer code, original operations (addition, multiplication, sinus, etc.) have to be replaced by their Taylor counterparts, defined in PACE library.

2. Application to orbit propagation

2.1. Orbit propagation in Thales JACK software

A Taylorised version of Thales numerical orbit propagator has been implemented in JACK software. This propagator uses PACE library to perform Taylor polynomial computation. This orbit propagator implements the following models:



- Earth gravity up to any order and degrees,
- Moon, Sun gravity and other planets,
- Atmospheric drag,
- Solar radiation pressure,
- Tides (solid and oceanic),
- Relativistic force.

Included models allow precise orbit propagation to be performed for any kind of orbits (LEO, GEO, GTO, etc.).

Most perturbations are easy to convert into Taylor algebra since it involves only algebraic operations. The trickiest has been solar radiation pressure and particularly to properly manage the transition from enlightened to shadowed part of the orbit since it involves non-algebraic operations. This point is discussed in section 2.2.

Then starting from an initial state vector $[x_0, y_0, z_0, vx_0, vy_0, vz_0]^T$, one propagation is performed in Taylor algebra. The result is a vector of polynomial $\overrightarrow{X_f}$ function of an initial state vector $\overrightarrow{X_i}$:

$$\overrightarrow{X_f} = f\left(\overrightarrow{X_i}\right)$$

Then multiple evaluations of the polynomial *f* are performed for various initial states $\overrightarrow{X_i}$ depending on initial dispersion sets. These polynomials evaluations are very fast since thousands of evaluations can be performed in a fraction of a second with modern computers (see Table 1: computation times for Taylor algebra low-level operationsTable 1).

This is different from classic Monte-Carlo simulations which require as many orbit propagations as initial samples $\overrightarrow{X_i}$. Hereunder is an example with 1000 samples:

- Classic Monte-Carlo requires 1000 propagations,
- Monte-Carlo using Taylor Algebra requires one propagation (which is more computationally intensive) in Taylor Algebra and 1000 very fast polynomial evaluations.

More formally computation time T_{MC} with classic Monte-Carlo method is:

$$T_{MC} = nT_{Real}$$

with *n* the number of samples and T_{Real} the duration of an orbit propagation.

The computation time T_{TA} with Monte-Carlo using Taylor Algebra is:

$$T_{TA} = T_{Taylor} + nt$$

with T_{Taylor} the duration of a Taylor propagation. nt being very small (see Table 1), it yields to:

$$T_{TA} \approx T_{Taylor}$$

Hence using Taylor Algebra is more efficient if:

$$T_{TA} < T_{MC}$$

Or:

$$T_{Taylor} < nT_{Real}$$

Results on threshold number of samples *n* leading to $T_{Taylor} = nT_{Real}$ are provided in section 2.3.

2.2. Issues and solutions

The main issue when translating a computation code into Taylor Algebra is translating non-algebraic operations. In particular conditions such as *if (…condition…) then … else …* are not easy to translate since they involve comparisons.

First case: discontinuities

Such a pattern will occur if there is a discontinuity in the system dynamics. A solution to circumvent this issue is to approximate the condition with an arctangent function with suitably chosen parameters:

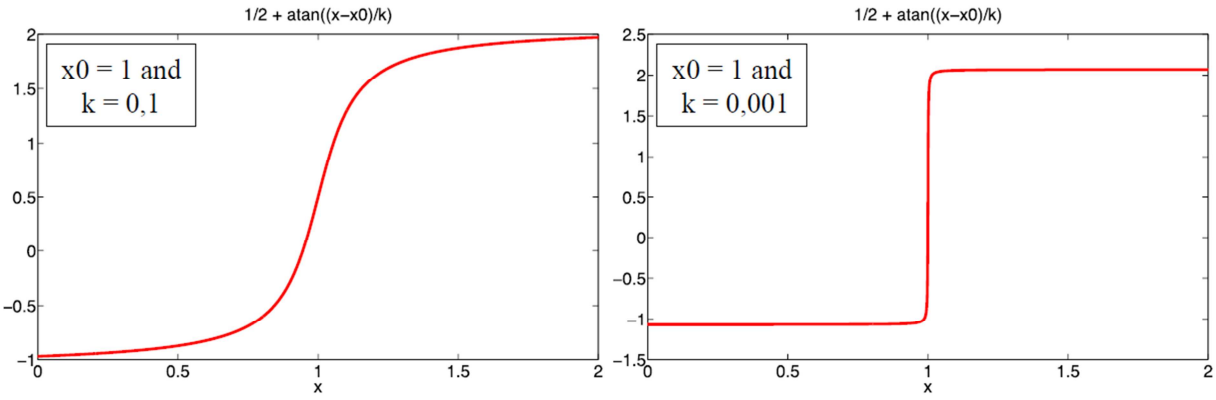


Figure 3: Approximation of a discontinuity by an arctangent function.

One can see in the plots above for a very low parameter *k*, it well fits with the expected discontinuity. The main advantage here is to have a function, *atan*, which is infinitely differentiable and then its Taylor expansion can be obtained.

Second case: the solar radiation pressure shadow function

The main problem met when converting the orbit numerical propagator into Taylor Algebra was to deal with the solar radiation pressure shadow function. This shadow function is continuous, infinitely differentiable but is not easy to translate into Taylor Algebra since it involves comparisons to know if the spacecraft is within the bounds of the shadowed part of the orbit or not. Hence the shadow function looks like this:

$$f(\theta) = \begin{cases} 0 \ \ if\ \theta \in umbra \\ g(\theta) \ \ if\ \theta \in penumbra \\ 1 \ \ if\ \theta \in light \end{cases}$$

Here is the plot of the function:

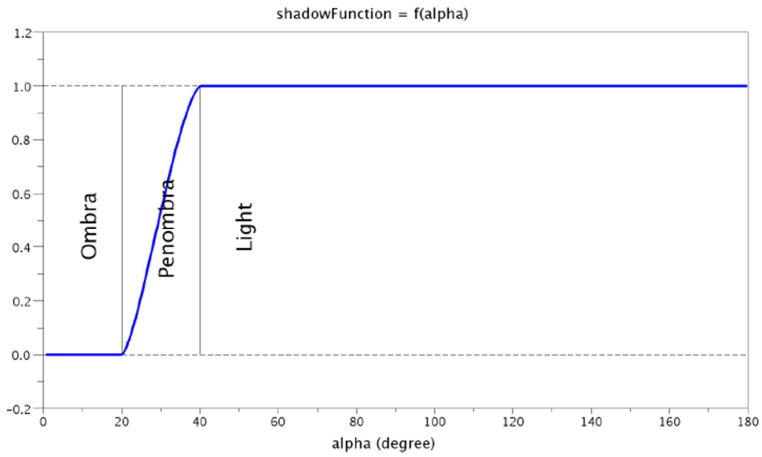


Figure 4: Shadow function of the solar radiation pressure.

The computation of the shadow function involves roots extraction of a second order polynomial. Then determining the spacecraft position with respect to Earth shadow is based on the value of these roots and the comparison issue still remains.

We came with the idea[6] that instead of extracting the real roots of the second order equation, we could keep the complex root and then get the final shadow function with complex coefficients. This required to define polynomials with complex coefficients. Operations on these polynomials are very similar to polynomials with real coefficients and have been included in PACE. As a result the shadow function is now a function with values in C field and does not have any conditions any more:

$$\begin{array}{rccc} g: & \Re & \rightarrow & C \\ & \alpha & \rightarrow & g(\alpha) \end{array}$$

The plot of the real and the imaginary part of the shadow function is presented hereafter:

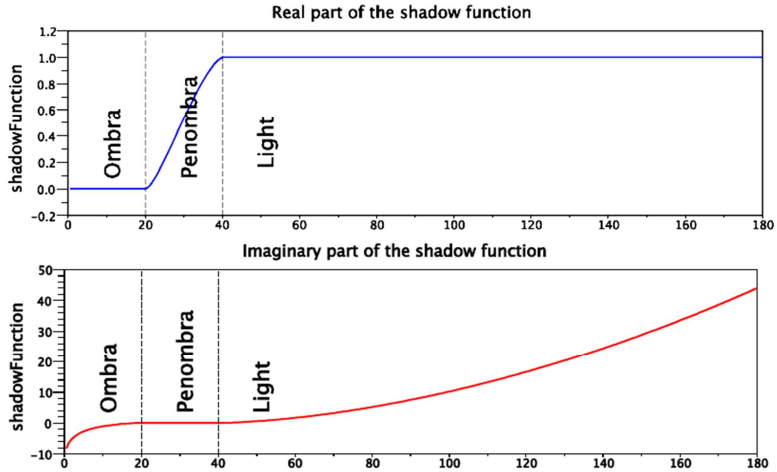


Figure 5: Shadow function of the solar radiation pressure (real and complex parts).

We notice that the real part of the complex shadow function is exactly the shadow function we were expecting for. And this has been obtained without any condition *if (…condition…) then … else* …which cannot be translated into Taylor Algebra. As a result, pieces of code involving conditions on

roots of a n^{th} order polynomial can readily translate into Taylor algebra by computing the complex roots thus removing any conditions on the roots.

2.3. One-day LEO TLE propagation results

Hereunder are presented some orbit propagation results obtained in the following conditions:

- One-day propagation from an object from TLE catalog,
- Set of usual perturbation (Earth gravity 10x10, Sun/Moon, SRP and atmospheric drag).

A Monte-Carlo is then performed with the following parameters:

- 1000 samples,
- Dispersed parameters: position and velocity of initial state.
- Average TLE dispersion (σ = 10km – 10m/s on every component of state vector).

Polynomials are expanded to the 5^{th} order. Hence, computations are performed in the algebra ${}_6D_5$.

Computation time results:

Taylor propagation is as fast as 250 single propagations. This means that if more than 250 samples are required Taylor propagation will be faster. Hence with 1000 samples, Taylor Monte-Carlo is 4 times faster than usual Monte-Carlo.

Accuracy:

Hereafter are presented two plots.

- First plot is the absolute position error on all the samples for various polynomial orders (hence in the algebras ${}_6D_3$, ${}_6D_5$, ${}_6D_7$). The result show that the higher the order, the more accurate the results. However some quality numerical issues will arise for high orders (around 12 in our case).

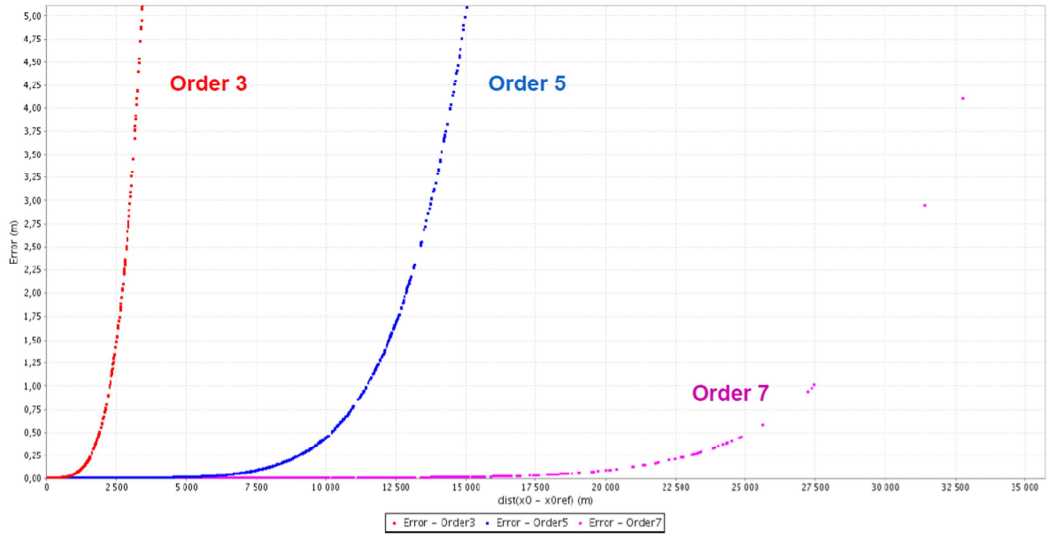


Figure 6: Absolute position error for various algebra orders.

- Second plot shows Monte-Carlo results in the plan (semi-major axis, mean anomaly). The results obtained with usual Monte-Carlo are in red. Plot shows good agreement between usual Monte-Carlo and Taylor method used with 5^{th} order polynomials. Hence 5^{th} order algebra allows the dynamics of the system to be correctly described for LEO one-day propagation on the chosen initial dispersion sets.

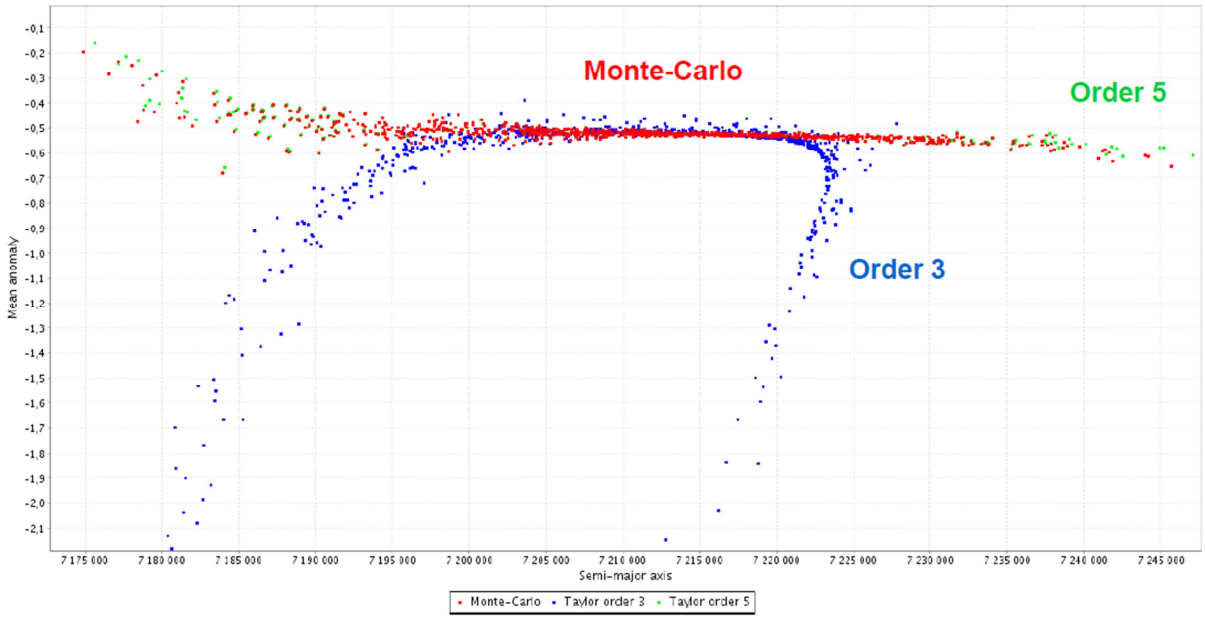


Figure 7: Dispersion error in the plan (semi-major axis, mean anomaly) for various polynomial orders.

One another interesting result is to compute state transition matrix with Taylor Algebra. In that case the computation is performed in ${}_6D_1$ (1st order algebra). Computation is hence very fast and results shows to be twice faster than traditional state transition matrix computation without requiring to write down the variational equations.

3. Ongoing work and future studies

Current and future work includes improvements of computation times since this is a key to Taylor algebra success. It also includes working on a way to convert automatically a code from the real number algebra to the Taylor algebra. Some languages such as C++ allow to overload basic operations (+, ×, /, .) which saves some translation time, but will not allow to overload more complex operations which means partial automatic translation. Taylor algebra also possesses some interesting properties like being a differential algebra. This allows efficient integrators[2] optimized for Taylor Differential Algebra to be defined and is currently being investigated. At last, Thales is currently involved in funded R&D activities regarding operational software using Taylor Algebra.

Conclusion

We showed how to implement Taylor Differential Algebra in a computer program. In particular this has been done in Thales library PACE. This low-level library provides an efficient polynomial computation engine and can be used in any on-top project requiring to deal with uncertainties. In particular we used it for orbit propagation. Even for complex force models, our implementation of a 5th order TDA orbit propagation runs as fast as two hundred classical propagations. Even if some limitations still remain concerning intermediate evaluations, and improvements are still achievable on computation times, Thales is working for Taylor Differential Algebra to become a powerful tool for a wide variety of numerical applications.

References

[1] M. Berz, A new method of TPSA algebra for the description of beam dynamics to high orders, 1986

[2] M. Berz, Modern map methods in particle beam physics, 1999

[3] M. Berz, Differential algebraic description of beam dynamics to very high orders, 1988

[4] M. Berz, Computation and application of Taylor polynomials with interval remainder bounds, 1996

[5] A. Jorba, A methodology for the numerical computation of normal forms, centre manifolds and first integrals of hamiltonian systems, 1997

[6] E. Bignon, JACK: an accurate numerical orbit propagator using Taylor Algebra, Kepassa conference 2014