AAS 19-716

APPLICATION OF DUAL NUMBER THEORY TO STATISTICAL ORBITAL DETERMINATION

Christopher B. Rabotin*

The problem of computing a matrix of partial derivatives with respect to a state to be estimated is shown to be solved using dual number theory. Specifically, hyper-dual spaces are used to compute the state transition matrix and the sensitivity matrix used in Kalman filtering. An implementation of this method has been demonstrated in a programming language called Rust. Benchmarks show similar computation time performance between the analytical method and the hyper-dual method for computing the state transition matrix.

INTRODUCTION

In the field of statistical orbital determination, observations of a spacecraft's range, range rate, or other sources of measurements, are fed into a Kalman filter. The dynamics model of the Kalman filter is determined by an approximation of the orbital dynamics on the spacecraft at the time of the observations. This procedure employs a matrix of partial derivatives. These derivatives are traditionally computed analytically, manually, or via symbolic manipulators. These partial derivatives must be derived anew each time the dynamics of simulation change, e.g. inclusion of higher order gravity models. Moreover, a matrix of partials is needed for the sensitivity matrix, which maps the observations to the estimate spacecraft state. This matrix of partials must also be computed anew for each observations and for each new estimated variable in the Kalman filter state. This paper demonstrates how the use of dual-numbers eases these onuses.

First, the mathematical notions of dual numbers and hyper-dual spaces is recalled, along with their error-free auto-differentiation properties. Subsequently, hyper-dual spaces are elegantly applied to statistical orbital determination problems, both for the computation of linearized dynamics, and for the sensitivity matrix of a Kalman filter. Further, a comparison of the execution speeds of the same statistical orbital determination problem using the hyper-dual space formulation and the traditional manual derivation formulation is provided.

DUAL NUMBERS AND HYPER-DUAL SPACE

Dual number theory

Dual numbers are a type of complex numbers.[1] The ubiquitous set of complex numbers, $\mathbb{C}$, may be defined as follows, where i is the imaginary number:

$$\mathbb{C} = \mathbb{R}[i] = \{z = a + bi \mid (a, b) \in \mathbb{R}^2, i^2 = -1\} \tag{1}$$

Similarly, we may define the set of dual numbers as follows, where ϵ is the dual number:

$$\mathbb{D} = \mathbb{R}[\epsilon] = \{z = a + b\epsilon \mid (a, b) \in \mathbb{R}^2, \epsilon^2 = 0 \text{ and } \epsilon \neq 0\} \tag{2}$$

Moreover, for $z = a + b\epsilon$ where $z \in \mathbb{D},\ (a, b) \in \mathbb{R}$, let us define the $real$ and $dual$ parts of a dual number such as

$$\begin{cases} real(z) = a \\ dual(z) = b \end{cases} \tag{3}$$

*Aerospace and Software Engineer at Advanced Space, 2100 Central Ave., #102, Boulder, CO 80301

An auto-differentiation property emerges from the addition of this nilpotent element, as is obvious from a Taylor series expansion.[2,3] Evidently, this result is only valid for values of a where the function is differentiable.

$$\begin{aligned} f : \mathbb{D} \to \mathbb{D}\,, (a,b) \in \mathbb{R}^2 & \\ f(a+b\varepsilon) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a) b^n \varepsilon^n}{n!} \\ &= f(a) + b\frac{df(a)}{da}\varepsilon \\ &\begin{cases} real(f(a+b\epsilon)) = f(a) \\ dual(f(a+b\epsilon)) = b\frac{df(a)}{da} \end{cases} \end{aligned} \tag{4}$$

By choosing $b = 1$, the first derivative comes out for free by simply evaluating the function f.

Hyper-dual spaces

We can further extend the dual numbers to a hyper-dual space. Let us define a hyper-dual space of size 2 as follows, where ϵ_j is the j-th dual number:

$$\mathbb{D}^2 = \mathbb{R}[\epsilon_x, \epsilon_y] = \{z = a + b\epsilon_x + c\epsilon_y + d\epsilon_x\epsilon_y \mid (a,b,c,d) \in \mathbb{R}^4, \epsilon_\gamma^2 = 0,\ \epsilon_\gamma \neq 0,\ \gamma \in \{x,y\},\ \epsilon_x\epsilon_y \neq 0\} \tag{5}$$

This mathematical tool enables auto-differentiation of multi-variate functions as follows, where $dual_\gamma$ corresponds to the γ-th dual number, i.e. the number associated with ϵ_γ.

$$\begin{aligned} f : \mathbb{D}^2 \to \mathbb{D}^2,\ (x,y) \in \mathbb{R}^2 & \\ &\begin{cases} real(f(x+\epsilon_x,\ y+\epsilon_y)) = f(x,y) \\ dual_x(f(x+\epsilon_x,\ y+\epsilon_y)) = \frac{\partial}{\partial x} f(x,y) \\ dual_y(f(x+\epsilon_x,\ y+\epsilon_y)) = \frac{\partial}{\partial y} f(x,y) \end{cases} \end{aligned} \tag{6}$$

Example

Let us detail a computation example of a smooth multivariate polynomial function defined over all reals.

$$\begin{aligned} f : \mathbb{R} \to \mathbb{R},\ (x,y) \in \mathbb{R}^2 & \\ f(x,y) &= 2x^3 - 0.2y^2 + x \\ \frac{\partial}{\partial x} f(x,y) &= 6x^2 + 1 \\ \frac{\partial}{\partial y} f(x,y) &= -0.4y \end{aligned} \tag{7}$$

Let us extend the definition of this function to $\mathbb{D}^2$.

$$\begin{aligned} g : \mathbb{D} \to \mathbb{D},\ (x,y) \in \mathbb{R}^2 & \\ g(x+\epsilon_x,\ y+\epsilon_y) &= 2(x+\epsilon_x)^3 - 0.2(y+\epsilon_y)^2 + (x+\epsilon_x) \end{aligned} \tag{8}$$

Trivially,

$$\begin{aligned} (x+\epsilon_x)^2 &= x^2 + 2x\epsilon_x \\ (x+\epsilon_x)^3 &= x^3 + 3x^2\epsilon_x \end{aligned} \tag{9}$$

Hence, g may be written as follows:

$$\begin{aligned} g(x+\epsilon_x,\ y+\epsilon_y) &= 2(x+\epsilon_x)^3 - 0.2(y+\epsilon_y)^2 + (x+\epsilon_x) \\ &= 2(x^3+3x^2\epsilon_x) - 0.2(y^2+2y\epsilon_y) + (x+\epsilon_x) \\ &= (2x^3 - 0.2y^2 + x) + (6x^2+1)\epsilon_x - 0.4y\epsilon_y \end{aligned} \tag{10}$$

As expected from equation 6, the $dual_x$ part of g corresponds to the partial of f with respect to x, the $dual_y$ part of g corresponds to the partial of f with respect to y, and the $real$ part of the g corresponds to f.

APPLYING HYPER-DUAL SPACE TO STATISTICAL ORBITAL DETERMINATION USING KALMAN FILTERING

The state transition matrix

The state transition matrix (STM, $\boldsymbol{\Phi}$) is a linearization procedure of a dynamical system. In the field of astrodynamics, it is an approximation of the dynamics over a short period of time. In the case of statistical orbital determination, the STM is computed around a reference point of an orbit and propagated forward in time until the next propagation time step or until the next spacecraft observation. The STM is computed via the partials matrix $\boldsymbol{A}$ of the state $\boldsymbol{X}$ at a time t, as shown in equation 11.[4]

$$\boldsymbol{A}(t) = \frac{d\boldsymbol{X}(t)}{d\boldsymbol{X}_0} \tag{11}$$

If one only estimates the spacecraft state, then the partials matrix corresponds to the gradient of the orbital dynamics applied to the spacecraft. The state $\boldsymbol{X}$ of a spacecraft may be defined as the vector of its position and velocity in a Cartesian frame. Equation 12 shows the relation between the partials matrix and the STM, where t and t_0 correspond to two different times such that $t > t_0$.

$$\frac{d\boldsymbol{\Phi}(t,t_0)}{dt} = \boldsymbol{A}(t)\boldsymbol{\Phi}(t,t_0) \tag{12}$$

The $\boldsymbol{A}$ matrix contains the partial derivatives of the accelerations $(a_x,\ a_y,\ a_z)$ with respect to each of the components of the position of the spacecraft, cf. equation 13.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\partial a_x}{\partial x} & \frac{\partial a_y}{\partial x} & \frac{\partial a_z}{\partial x} & 0 & 0 & 0 \\ \frac{\partial a_x}{\partial y} & \frac{\partial a_y}{\partial y} & \frac{\partial a_z}{\partial y} & 0 & 0 & 0 \\ \frac{\partial a_x}{\partial z} & \frac{\partial a_y}{\partial z} & \frac{\partial a_z}{\partial z} & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

In the case of two-body dynamics, the $\boldsymbol{A}$ matrix is relatively simple to compute. Deriving this partials matrix for higher fidelity dynamics or for a larger state is more complicated. In practice, this may lead orbital estimation software to ignore part of these dynamics. Navigators may then account for smaller perturbations by inflating the covariance matrix with stochastic noise.

Dual numbers, however, provide the partials matrix as part of the computation of the equations of motion (EOM) themselves, as long as these EOMs describe the movement of all of the state variables to be estimated. In practice, this requires defining a hyper-dual space whose size is equal to the number of variables in the state to be estimated. For example, if estimating the Cartesian state a spacecraft and a maneuver magnitude at a reference point during an orbital determination arc, building a seven-dimensional dual space will return the result of the EOMs and the partials matrix computed at that point. An open source example of the building of such hyper-dual space has been implemented as a test case in *dual_num*, a thorough dual number library in a programming language called Rust.[5]

The sensitivity matrix

In a Kalman filter, the sensitivity matrix, noted $\tilde{H}$, relates the filter covariance, the filter gain, the measurements and the noise of the measurement. Like the state transition matrix, the sensitivity matrix is a partials matrix of size $N \times M$, where N is the size of the measurement and M is the size of the state to be estimated. For example, if the measurement is the range ρ and the range-rate $\dot{\rho}$, and the estimated state is the position $\{x, y, z\}$ and velocity $\{\dot{x}, \dot{y}, \dot{z}\}$, then the sensitivity matrix is written as equation 14.

$$\tilde{H} = \begin{bmatrix} \frac{\partial \rho}{\partial x} & \frac{\partial \rho}{\partial y} & \frac{\partial \rho}{\partial z} & \frac{\partial \rho}{\partial \dot{x}} & \frac{\partial \rho}{\partial \dot{y}} & \frac{\partial \rho}{\partial \dot{z}} \\ \frac{\partial \dot{\rho}}{\partial x} & \frac{\partial \dot{\rho}}{\partial y} & \frac{\partial \dot{\rho}}{\partial z} & \frac{\partial \dot{\rho}}{\partial \dot{x}} & \frac{\partial \dot{\rho}}{\partial \dot{y}} & \frac{\partial \dot{\rho}}{\partial \dot{z}} \end{bmatrix} \tag{14}$$

Using the same methodology as previously, it is evident that the sensitivity matrix may be computed by simply defining a hyper-dual space whose size is equal to that of the state to be estimated. The equations which return the range ρ and the range-rate $\dot{\rho}$ from an input state will then automatically also return the components of the sensitivity matrix. An open source example of this hyper-dual space has also been implemented as a testing procedure in the Rust dual numbers library *dual_num*.[6] It should also be noted that auto-differentiation libraries using dual numbers are also available in C++, Python, and Julia.[7–9]

Benchmarks

The formulation presented here has been implemented in the open-source* *nyx* toolkit for astrodynamics, which is validated against NASA GMAT 2018a. This toolkit is programmed in Rust, a systems-level language which provides compile-time guarantees on memory management and thread safety and is mostly portable to embedded systems architectures. Computation speeds in Rust are on par with C and faster than FORTRAN. Rust also allows overloading operators, which allows it to takes advantage of the chain rule, and therefore enables low-level compile-time optimization of the operations. In this code, forward accumulation for the automatic differentiation is performed whereby if the function to differentiate is defined as $f : \mathbb{R}^m \to \mathbb{R}^n$, then m evaluations of the function f are required.

In the following benchmark, the program will propagate the position and velocity of a spacecraft in an Earth centered inertial frame using only two-body dynamics and an RK4 propagator with a fixed-step of 10 seconds for a simulated time indicated in days in table 1. In parallel, the program will estimate the position and velocity of the spacecraft using a classical Kalman filter by generating measurements at each time-step from the three NASA Deep Space Network ground stations, each propagated in a simplified Earth-centered, Earth-fixed frame. The ground stations are simulated with an elevation mask of zero degrees, and a noise level for the range and range-rate measurements respectively set to zero meters and zero meters per second.

The following benchmarks were executed on a Linux openSUSE Leap 15 desktop equipped with a twelve core Intel i7-8700K CPU at 3.70 GHz each and 15.4 GiB of RAM memory. The average duration of five runs, as measured by the operating system, is presented in the table below. In both scenarios, the sensitivity matrix is computed using hyper-dual space as previously described. These benchmarks show that the computational speed is not impacted by the STM method used despite the apriori complexity of the mathematics of hyper-dual spaces. Further, the memory usage is similar in both cases with a maximum of 1.018 MiB for the analytical STM method and 1.006 MiB for the hyper-dual STM method for a simulated time of one day.

It is important to note that the results are identical in either method.

CONCLUSION

This paper describes a novel approach to computing the state transition matrix and the sensitivity matrix as used in the problem of orbital determination. One of the key advantages of the hyper-dual formulation is that it allows for high-fidelity statistical orbital determination regardless of the complexity of the dynamics

*Although open-source, *nyx* (https://github.com/ChristopherRabotin/nyx) is licensed under the Apache License 2.0 subject to the Common Clause Restriction. Refer to the LICENSE file for legal details.

Table 1. Full problem computation time for using either analytical or hyper-dual STM method

Simulated time (days)	Computation time (seconds)	
	Analytical STM	Hyper-dual STM
1.0	0.12	0.12
7.0	0.13	0.14
14.0	0.42	0.51
30.5	0.78	0.96
182.62	4.00	5.16
365.25	7.88	10.19

and without any significant impact in computational time, whereas analytical derivatives and their software implementations are often truncated for highly non-linear regimes such as cislunar space.

ACKNOWLEDGMENTS

Special thanks to Dr. Nathan Parrish for the conversation which sparked this research, and whose own research used dual number theory for low-thrust optimization.[10] I would also like to thank Aaron Trent for the thorough implementation of the dual numbers library in Rust, *dual_num*, and for his availability to update the package with my recommended code changes. I would like to acknowledge Carlos Deccia and Hunter Mellema for reviewing the draft of this paper.

REFERENCES

[1] J. Rooney, "On the Three Types of Complex Number and Planar Transformations," *Environment and Planning B: Planning and Design*, Vol. 5, No. 1, 1978, pp. 89–99, 10.1068/b050089.
[2] F. Messelmi, "Analysis of dual functions," *Annual Review of Chaos Theory, Bifurcations and Dynamical Systems*, Vol. 4, 2013, p. 37.
[3] J. A. Fike and J. J. Alonso, "The Development of Hyper-Dual Numbers for Exact Second-Derivative Calculations," *AIAA paper 2011-886, 49th AIAA Aerospace Sciences Meeting*, 2011.
[4] B. Schutz, B. Tapley, and G. H. Born, *Statistical orbit determination*. Elsevier, 2004.
[5] A. Trent and C. Rabotin, "dual_num/differentials.rs at f90446e7," 2019.
[6] A. Trent and C. Rabotin, "dual_num/lib.rs at f90446e7," 2018.
[7] OSTI.gov, "Implementing automatic differentiation efficiently," 1990.
[8] J. Revels and T. Papamarkou, "DualNumbers.jl," 2018.
[9] M. Bucker, "AD Tools - AutoDiff.org," 2018.
[10] N. L. O. Parrish, "Low Thrust Trajectory Optimization in Cislunar and Translunar Space," 2018.