Advanced SPACE

Delivering innovation to orbit.™

Application of Dual Number Theory to Statistical Orbit Determination

AAS 19-716

2100 Central Avenue, Suite 102
Boulder, CO 80301
720-545-9191

Reminders about Statistical Orbit Determination

- Spacecraft dynamics locally linearized via the State Transition Matrix (STM)
  - Requires the partial derivative of the state parameters, which varies depending on the spacecraft dynamics accounted for
  - Let the acceleration in a Cartesian frame be represented by $(a_x,\ a_y,\ a_z)$ then the simplest partials matrix used to compute the STM may be written as follows:

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

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

Reminders about Statistical Orbit Determination

- Observations are mapped to these linearized dynamics via the sensitivity matrix
  - Requires the partial derivative of the measurement with respect to each of the state parameters to be estimated
  - For example, the sensitivity matrix mapping simultaneous range and Doppler to a spacecraft state in Cartesian may be written as follows:

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

Computing partial derivatives

- Computed analytically:
  - Error prone and labor intensive
- Finite differencing:
  - Reduces the precision to half of the digits of machine precision

    e.g. if 1e-16 is machine precision, finite differencing leads to 1e-8 precision
- Automatic differentiation via dual number theory:
  - Machine precision computation
  - Negligible computational overhead in some languages

Introduction to Dual Number theory

- Flavor of complex numbers
- With a nilpotent element: epsilon

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

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

Introduction to Dual Number theory

- Nilpotent element enables automatic differentiation

$$f : \mathbb{D} \to \mathbb{D}\,, (a,b) \in \mathbb{R}^2$$

$$f(a+b\varepsilon) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a) b^n \varepsilon^n}{n!}$$

$$= f(a) + b\frac{df(a)}{da}\varepsilon$$

$$\begin{cases} real(f(a+b\epsilon)) = f(a) \\ dual(f(a+b\epsilon)) = b\frac{df(a)}{da} \end{cases}$$

(if the real part of the dual number if differentiable)

Introduction to Dual Number theory

- Defining multiple nilpotent elements enables multiple derivatives

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

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.

$$f : \mathbb{D}^2 \to \mathbb{D}^2,\ (x,y) \in \mathbb{R}^2$$

$$\begin{cases} real(f(x+\epsilon_x,\ y+\epsilon_y)) = f(x,y) \\ dual_x(f(x+\epsilon_x,\ y+\epsilon_y)) = \frac{\partial}{\partial x} f(x,y) \\ dual_y(f(x+\epsilon_x,\ y+\epsilon_y)) = \frac{\partial}{\partial y} f(x,y) \end{cases} \quad (6)$$

Hyper-Dual Number example

Assume the following real function

$$f : \mathbb{R} \to \mathbb{R},\ (x, y) \in \mathbb{R}^2$$

$$f(x, y) = 2x^3 - 0.2y^2 + x$$

$$\frac{\partial}{\partial x} f(x, y) = 6x^2 + 1$$

$$\frac{\partial}{\partial y} f(x, y) = -0.4y$$

Extending to hyper-dual space of size 2

$$g : \mathbb{D} \to \mathbb{D},\ (x, y) \in \mathbb{R}^2$$

$$g(x + \epsilon_x,\ y + \epsilon_y) = 2(x + \epsilon_x)^3 - 0.2(y + \epsilon_y)^2 + (x + \epsilon_x)$$

Rewriting this, separating between real and dual parts

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

Software implementations

- Rust
  - Systems programming language (like C/C++) with operator overloading
  - Benefits from compile-time optimizations
  - hyperdual (used in benchmark): https://crates.io/crates/hyperdual
- Julia
  - Scientific language with Matlab-like notation
  - ForwardDiff: https://github.com/JuliaDiff/ForwardDiff.jl
  - HyperDualNumbers: https://github.com/JuliaDiff/HyperDualNumbers.jl
- Other implementations exist too (especially in C++ and Python)

Benchmark in orbital determination software

- Observations generated simultaneously (0.1 Hz)
- Conventional Kalman Filter
- Estimating position and velocity in two body dynamics
- Results identical to machine precision

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

Conclusion and references

- Less labor intensive compared to analytical derivation
- Higher precision computation than finite differencing
- Negligible computational overhead in many cases
- Compile-time optimizations (if applicable)

Main references

- 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*
- F. Messelmi, "Analysis of dual functions," *Annual Review of Chaos Theory, Bifurcations and Dynamical Systems, Vol. 4, 2013*
- B. Schutz, B. Tapley, and G. Born, *Statistical Orbit Determination*, Elsevier, 2004.