Using Multicomplex Variables for Automatic Computation of High-Order Derivatives

Gregory Lantoine

Mission Design Engineer JPL
(previously School of Aerospace Engineering, Georgia Tech)

Ryan P. Russell

Assistant Professor, The University of Texas at Austin
(previously School of Aerospace Engineering, Georgia Tech)

Thierry Dargent *

Research & development directorate, Thales Alenia Space

13rd European Workshop on Automatic Differentiation

Monday 10th and Tuesday 11th June 2013

INRIA Sophia-Antipolis, France

Motivations

- Working on 2nd order optimal control methods… (HDDP- variant of differential dynamic programming)

- Frustrated with expense of derivative calculations…
  - Analytic… tedious to code or not always possible
  - Finite differencing… easy but inaccurate, slow
  - Automatic Differentiation… not straightforward, slow
  - New complex step method (Squire & Trapp 1998, Martens 2003) …accurate for first order derivs only

- **OBJECTIVE**: ***Extend complex method to higher order derivatives***

Motivations

- Sensitivity Analysis
  - Partial Derivatives of outputs w.r.t. inputs

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \longrightarrow \boxed{f} \longrightarrow Y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

Gradient
mxn matrix

$$\nabla_X f = \begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \cdots & \dfrac{\partial y_1}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial y_m}{\partial x_1} & \cdots & \dfrac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Hessian
mxnxn tensor

$$\nabla_{XX} f = \left[\frac{\partial^2 y_k}{\partial x_i \partial x_j} \right]_{\substack{i,j=1\ldots n \\ k=1\ldots m}}$$

- Computations of Sensitivities highly desirable in many fields:
  - Design Optimization: gradient-based on satellite trajectory optimization, and optimal control
  - Inverse Problem (Data assimilation)
  - Curve fitting
  - Parameter identification
  - Nonlinear PDEs

Requirements of Sensitivity Computations

in order of importance (to us):

1. **Accurate**
   - Improvement of algorithm convergence
   - Compute adjoint state dynamic with the same précision as the state dynamic
2. **Fast (enough)**

3. **Easy-to-implement**
   - Low setup time for one problem
   - Generalized for different problems

Existing methods

- **Analytical (Manual) differentiation**
  - ***Can be*** accurate and efficient (depends on the programmer)
  - Knowledge of computer language needed for model implementation
  - Development time is long
  - Error prone
  - Maintaining derivatives an additional burden
  - Difficult on large numerical model

- **Symbolic differentiation**
  - Implies using software for symbolic manipulation such as **Maple or Mathematica**
  - Reduces model development time
  - Reduces errors associated with mathematical manipulations
  - Still requires human efforts for further model implementation
  - Might lead to non-efficient expressions

Existing methods

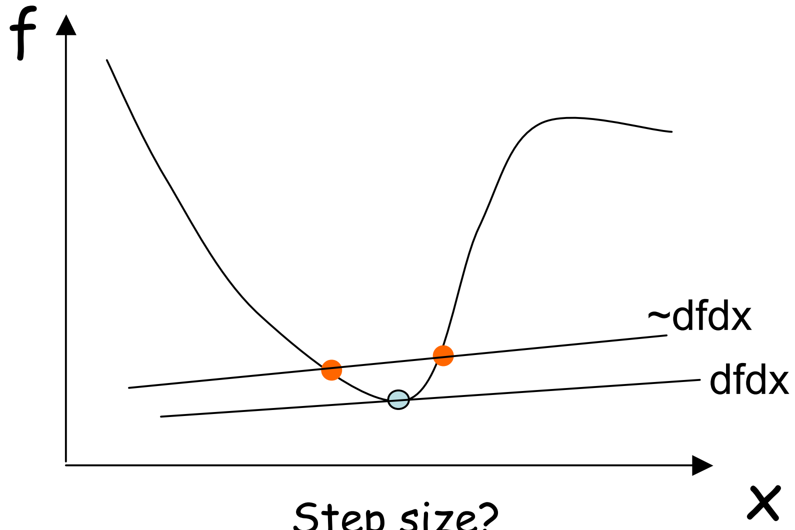
- **Finite Differencing**

Central second-order accurate difference:

$$\left.\frac{\partial F}{\partial x_i}\right|_{x=x_0} \approx \frac{F(x_0 + he_i) - F(x_0 - he_i)}{2h}$$

Error introduced

Step size



Step size?
Large : Averaging
Small : round off error in subtraction

- Easiest method to implement: only original computer program is required → Development time is minimal
- Accuracy is step dependent
- Computationally intensive: N derivatives will require N+1 function evaluations

→ Often inaccurate or inefficient

Existing methods

- **Automatic Differentiation**
  - Analytic differentiation of elementary functions
  - Repetitive application of the **chain rule**
  - Can be implemented in two ways:
    - Source transformation: Produce new code to calculate derivative based on original code of a function (ex: ADIFOR, TAPENADE, …)
    - Operator overloading: Each elementary operation is replaced by a new one, working on pairs of value and its derivative (doublet) (ex: AD02 from the Harwell Subroutine Library)

$$f(\mathbf{x}) = \frac{x_1 + \sin(x_2)}{\left(x_3 - \sqrt{x_4}\right)} - 2x_1$$

Elementary functions

Elementary operations

  - Derivatives of any order
  - Accurate to machine precision: No round-off errors
  - Complete Partial Derivatives with single execution
  - Computationally more efficient than FD
  - Not straightforward method to implement

Existing methods

- **Complex-step method** (Squire & Trapp 1998, Martens 2003)
  - Use complex variables instead of real variables
  - Found from Taylor Series expansion:

$$f(x+ih) = f(x) + \frac{df}{dx} ih + O(h^2) \quad \Longrightarrow \quad \frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{\mathrm{Im}[f(x+ih)]}{h}$$

No subtraction!

  - Avoid the subtraction involved in finite difference
  - Accurate to working precision for VERY small h < 10^{-8}
  - Very easy implementation
  - Separate simulations for each gradient required (like finite differencing)
  - Increased computational time due to complex arithmetic
  - Limited to first-order derivatives
    - (Lai, Crassidis: give tuning methods to find optimal step size for 2nd order approximations, still not machine accurate…)

ThalesAlenia Space

Georgia Tech

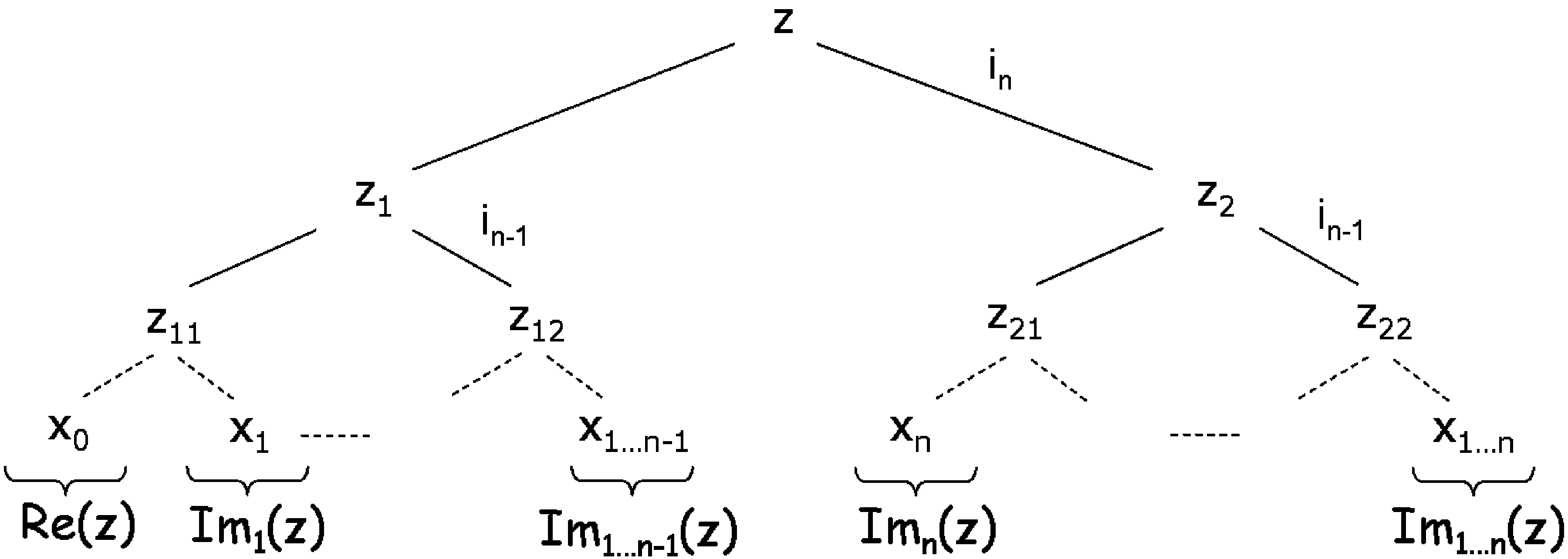
MultiComplex Numbers

- MultiComplex numbers (C^n) are extensions of complex numbers in higher dimensions

cross imaginary terms

$$z = x_0 + x_1 i_1 + \ldots + x_n i_n + \boxed{x_{12} i_1 i_2 + \ldots + x_{n-1n} i_{n-1} i_n + \ldots + x_{1\ldots n} i_1 \ldots i_n}$$

- Same formal representation as complex numbers can be used: $z = z_1 + z_2 i_n$ where $z \in C^n$ and $z_1, z_2 \in C^{n-1}$



MultiComplex Numbers

- Example: bicomplex numbers

$$z = x_0 + x_1 i_1 + x_2 i_2 + x_{12} i_1 i_2$$ where $x_1, x_2, x_3, x_4 \in \mathrm{R}$, $i_1^2=-1$, $i_2^2=-1$, $i_1 i_2 = i_2 i_{1c}$

$$z = z_1 + z_2 i_2$$ where $z_1, z_2 \in \mathrm{C}$, $i_2^2=-1$

- Example: tricomplex numbers

$$z = x_0 + x_1 i_1 + x_2 i_2 + x_3 i_3 + x_{12} i_1 i_2 + x_{13} i_1 i_3 + x_{23} i_2 i_3 + x_{123} i_1 i_2 i_3$$

$$z = z_1 + z_2 i_3$$ where $z_1, z_2 \in \mathrm{C}^2$, $i_3^2=-1$

imaginary terms can be represented as matrix operator example bicomplex i_1:

$$i_1 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Example 2nd Derivative From Taylor Series

$$f\left[x+h\left(i_1+i_2\right)\right]=f\left(x\right)+h\left(i_1+i_2\right)f'\left(x\right)+\frac{h^2}{2}\left(i_1+i_2\right)^2 f''\left(x\right)+H.O.T.\ \text{(ignore...)}$$

$$=f\left(x\right)+h\left(i_1+i_2\right)f'\left(x\right)+\frac{h^2}{2}\left(i_1i_1+2i_1i_2+i_2i_2\right)f''\left(x\right)$$

$$=f\left(x\right)+h\left(i_1+i_2\right)f'\left(x\right)+\frac{h^2}{2}\left(-2+2i_1i_2\right)f''\left(x\right)$$

NOTE: $i_n^2=-1$

$$=f\left(x\right)+h\left(i_1+i_2\right)f'\left(x\right)+h^2\left(i_1i_2\right)f''\left(x\right)-h^2 f''\left(x\right)$$

take now the coefficient of i_1i_2 from both sides, divide by h^2

$$\boxed{f''\left(x\right)=\frac{\mathrm{Im}_{12}\left(f\left[x+h\left(i_1+i_2\right)\right]\right)}{h^2}}+O\left(h^2\right)$$

we choose h extremely small,

say $1e-100$ and there is no subtraction error

$\left(h^2\right.$ must be representable with double precision$\left.\right)$

Lantoine, Russell, Dargent*

MultiComplex-Step Differentiation

- *Fundamental formula:*

$$f^{(n)}(x) = \frac{\mathrm{Im}_{1\ldots n}\left[f(x + hi_1 + \ldots + hi_n)\right]}{h^n} + O(h^2)$$

  - Use multicomplex variables instead of real variables
  - Found from Taylor Series expansion
  - Derivatives up to any order n
  - Same other advantages as complex method
  - See paper for mathematical formalities:
    - Using Multicomplex Variables for Automatic Computation of High-Order Derivatives, Gregory Lantoine, Ryan P. Russell & Thierry Dargent, ACM Transactions on Mathematical Software (TOMS) Volume 38 Issue 3, April 2012, Article No. 16
  - Details:
    - Step with h in each of the imaginary directions
    - Evaluate multi-complex function
    - Resulting derivatives retrieved from the coefficients of the multi-complex function result

Example Third Derivative Calculation

Multiple variables: (x,y,z)

$$\frac{\partial^3 f}{\partial x \partial y \partial z} = \frac{\mathrm{Im}_{123}\left(f\left[x+hi_1, y+hi_2, z+hi_3\right]\right)}{h^3}$$

$$\frac{\partial^3 f}{\partial x \partial y \partial y} = \frac{\mathrm{Im}_{123}\left(f\left[x+hi_1, y+hi_2+hi_3, z\right]\right)}{h^3}$$

$$\frac{\partial^3 f}{\partial y \partial y \partial y} = \frac{\mathrm{Im}_{123}\left(f\left[x, y+hi_1+hi_2+hi_3, z\right]\right)}{h^3}$$

ThalesAlenia Space

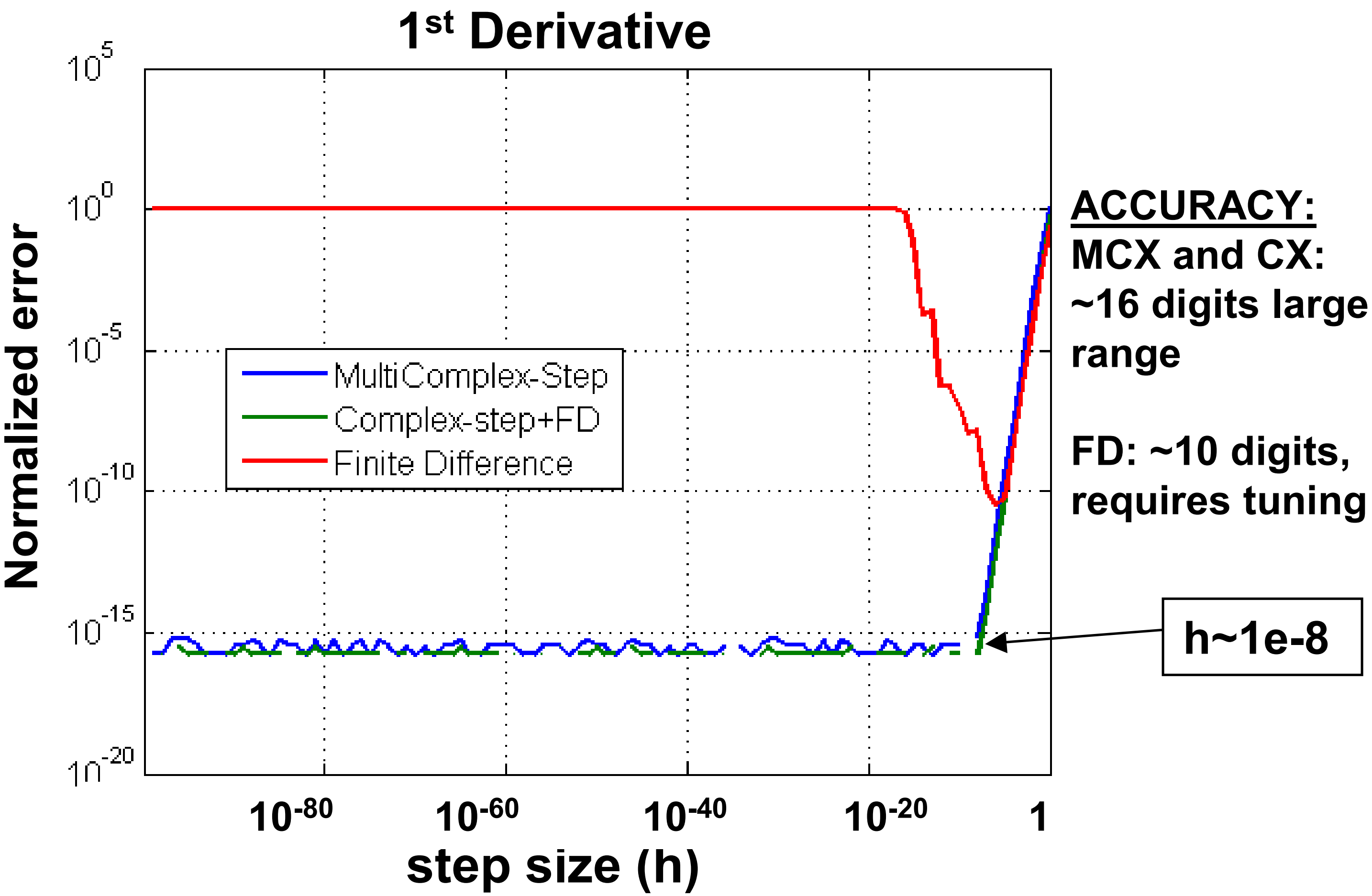
Lantoine, Russell, Dargent*

Georgia Tech

MultiComplex-Step Differentiation

- **simple example:**

$$f(x) = \frac{e^x}{\sqrt{\sin(x)^3 + \cos(x)^3}}$$



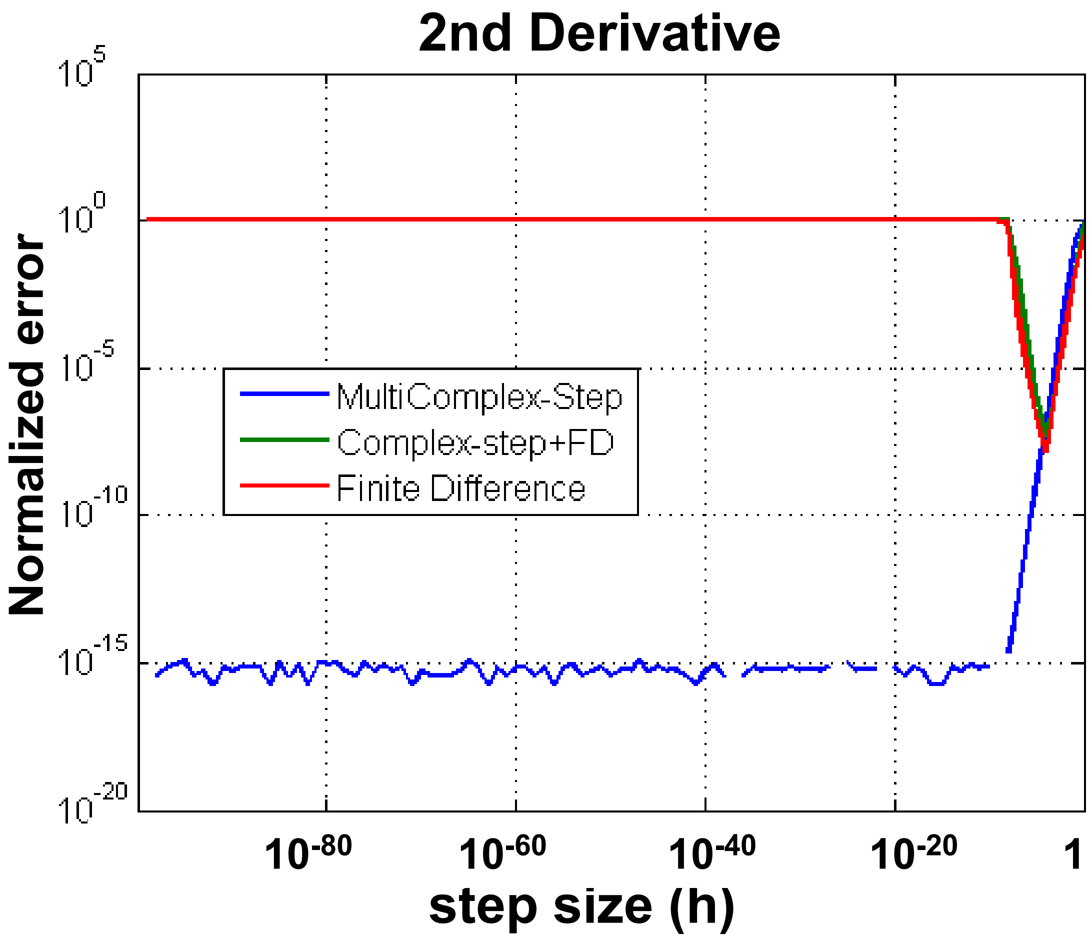
ACCURACY:
MCX and CX: ~16 digits large range

FD: ~10 digits, requires tuning

MultiComplex-Step Differentiation

- **simple example:**

$$f(x) = \frac{e^x}{\sqrt{\sin(x)^3 + \cos(x)^3}}$$

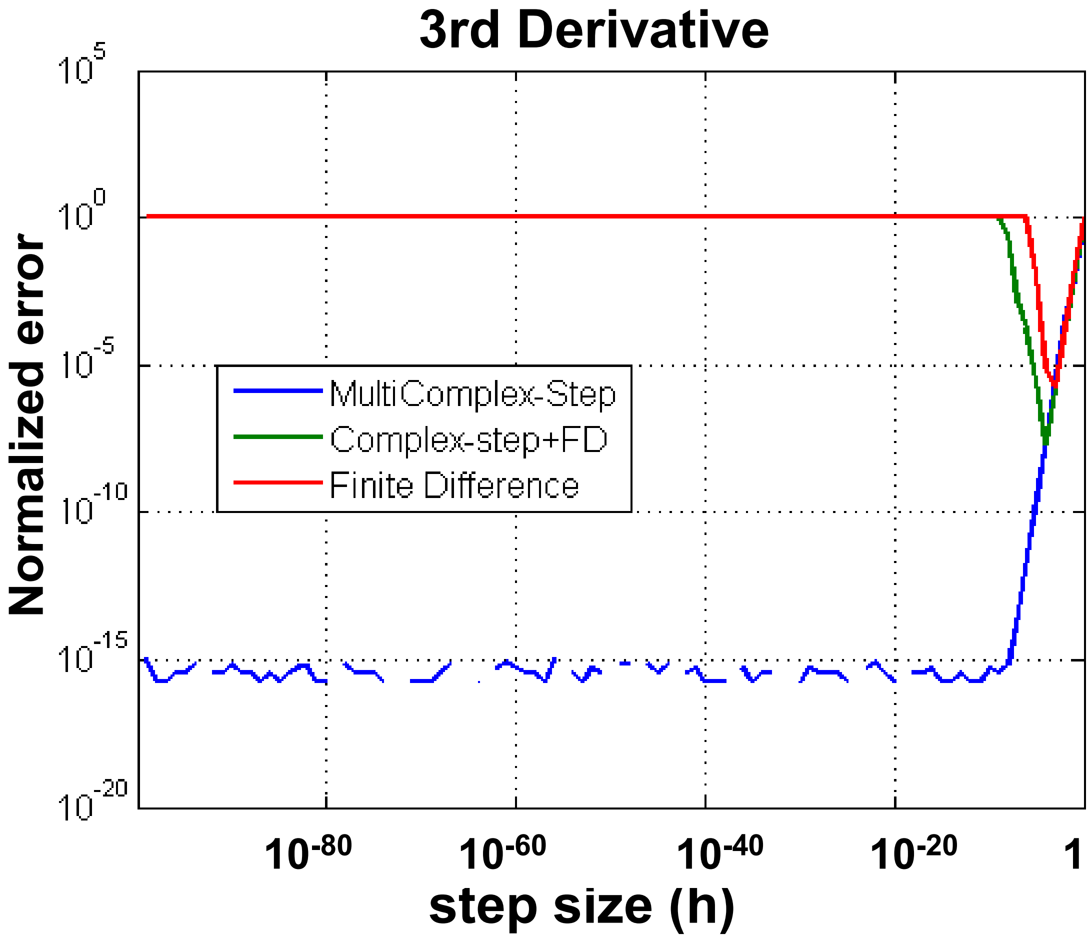


ACCURACY:
MCX: ~16 digits large range

CX and FD: ~7 digits, requires tuning

MultiComplex-Step Differentiation

- **simple example:** $$f(x) = \frac{e^x}{\sqrt{\sin(x)^3 + \cos(x)^3}}$$



ACCURACY:
MCX: ~16 digits large range

CX and FD: ~5-7 digits, requires tuning

Lantoine, Russell, Dargent*

Implementation

Declare all the variables as complex type

Or only the independent variables and the associated intermediate variables

Implementation

Declare all the variables as complex type

↓

Redefine all the functions with complex definitions

Or only the independent variables and the associated intermediate variables

Easy implementation using the same form of operator overloading functions as complex numbers

Implementation

Declare all the variables as complex type

↓

Redefine all the functions with complex definitions

↓

Add complex step (h) to the desired variable

- Or only the independent variables and the associated intermediate variables
- Easy implementation using the same form of operator overloading functions as complex numbers
- $x + hi_1 + \ldots + hi_n$

Implementation

Declare all the variables as complex type

→ Or only the independent variables and the associated intermediate variables

Redefine all the functions with complex definitions

→ Easy implementation using the same form of operator overloading functions as complex numbers

Add complex step (h) to the desired variable

$$x + hi_1 + \ldots + hi_n$$

Partial Derivative Calculations

$$f^{(n)}(x) \approx \frac{\mathrm{Im}_{1\ldots n}\left[f(x + hi_1 + \ldots + hi_n)\right]}{h^n}$$

ThalesAlenia Space

Lantoine, Russell, Dargent*

Georgia Tech

Implementation

Declare all the variables as complex type

Or only the independent variables and the associated intermediate variables

Redefine all the functions with complex definitions

Easy implementation using the same form of operator overloading functions as complex numbers

Add complex step (h) to the desired variable

$$x + hi_1 + \ldots + hi_n$$

Partial Derivative Calculations

$$f^{(n)}(x) \approx \frac{\mathrm{Im}_{1\ldots n}\left[f(x + hi_1 + \ldots + hi_n)\right]}{h^n}$$

Repeat as necessary for more independent variables

Implementation

Declare all the variables as complex type

Or only the independent variables and the associated intermediate variables

↓

Redefine all the functions with complex definitions

Easy implementation using the same form of operator overloading functions as complex numbers

↓

Add complex step (h) to the desired variable

$$x + hi_1 + \ldots + hi_n$$

↓

Partial Derivative Calculations

$$f^{(n)}(x) \approx \frac{\mathrm{Im}_{1\ldots n}\left[f(x + hi_1 + \ldots + hi_n)\right]}{h^n}$$

Repeat as necessary for more independent variables

Currently have working modules in *Matlab and Fortran*

Overloading Sample Code & DEMO

Overloading example in FORTRAN for + operator

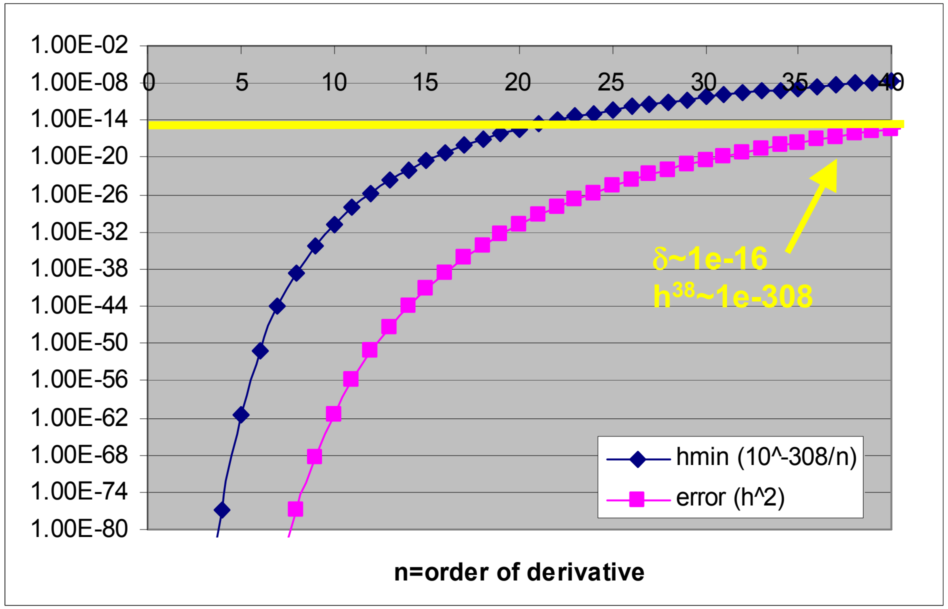
```
! Bicomplex overloading
type bicomplex
     COMPLEX*16 :: a
     COMPLEX*16 :: b
end type

interface operator(+)
   module procedure bicplx_plus_bicplx, bicplx_plus_bicplx_array, bicplx_plus_dble, bicplx_plus_dble_array
end interface


!##########################################################
function bicplx_plus_bicplx(q1,q2) result(q3)

implicit none

type(bicomplex), intent(in) :: q1, q2
type(bicomplex) :: q3

    q3%a = q1%a + q2%a
    q3%b = q1%b + q2%b

    return

end function bicplx_plus_bicplx
```


Lantoine, Russell, Dargent*

Step-size Limits for High order Derivatives

1) Since error is $O(h^2)$, then $h^2 > \delta = 1e\text{-}16$ therefore: $h > \sim 1e\text{-}8$
2) Because h^n appears in derivative approximation: $h^n > \varepsilon = 1e\text{-}308$,
3) therefore:
   - $h > 10^{-308/n}$
   - From above: $1e\text{-}8 > 10^{-308/n}$
   - $n < \sim 38$ for double precision (this is upperbound, in practice should be smaller due to dynamic range of variables, margin on error estimates….)
   - Note that a complex number with n = 35 is represented with $2^{35} > 10^{10}$ real numbers!!!

Min step size and error vs. Derivative order



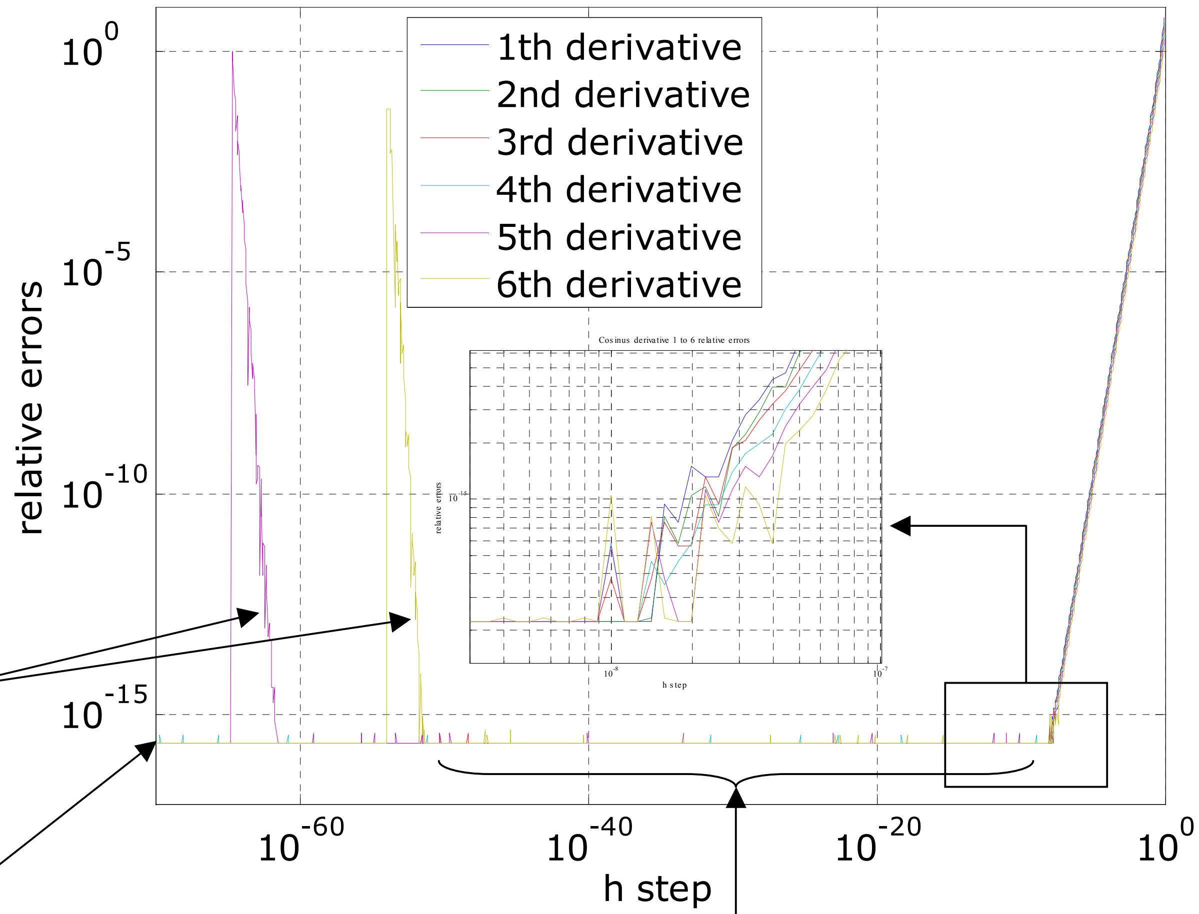
Elementary computation cost comparison to compute a product and its derivative

- MCX: $Z_1 * Z_2 = (x_1 + ih_1)*(x_2 + ih_2) = x_1 x_2 - h_1 h_2 + ih_1 x_2 + ih_2 x_1$
- AD: $\{x_1 x_2 \; ; \; d(x_1 * x_2) = dx_1 x_2 + x_1 \, dx_2 \}$

Over cost of MCX versus AD: the Product $h_1 h_2$

Example: cos(x), … ,d^n(cos x)/d^nx

Relative errors on 1^{st}-6^{th} derivative of cos(x)



h^5 & h^6 under-flow limitation

for double precision,
$(1+\delta)=1$ if $\delta < 2e\text{-}16$

Large choice for step h

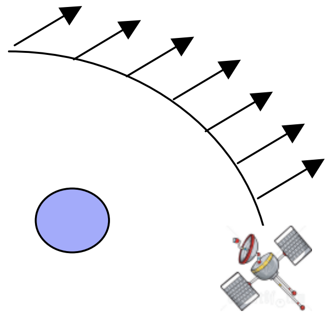
ThalesAlenia Space

Lantoine, Russell, Dargent*

Georgia Tech

Test Case: Trajectory

- Trajectory State Transition Matrix
  - Satellite subject to gravitational force and constant inertial thrust
  - Segment propagated for 6 days
  - 1^{st} and 2^{nd}-order State Transition Matrices useful in trajectory optimization (399 terms)

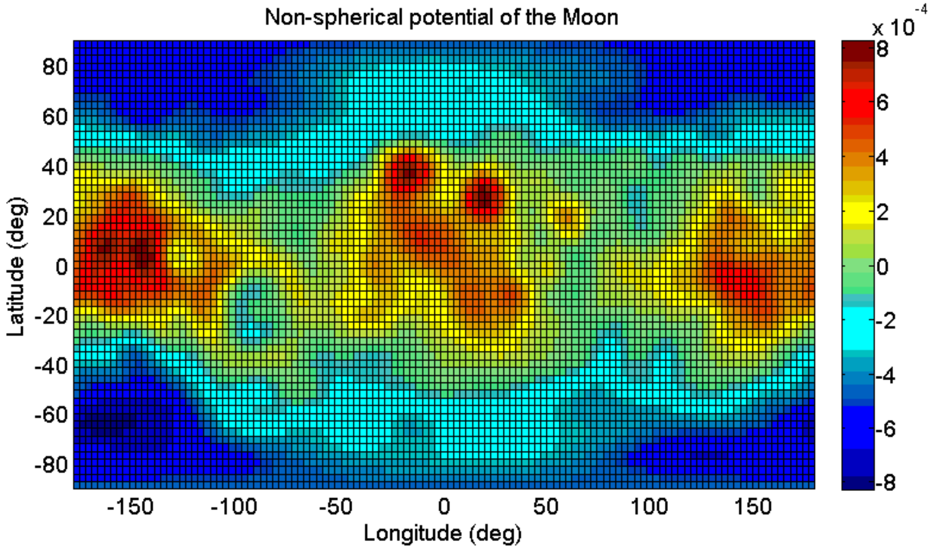


Method	Sample 2^{rd}-order STM	*Accuracy*	Total Relative Compute Time
Analytical	$-2.092290564266828\ 10^{-2}$	NA	1.0*
MultiComplex	*$-2.092290564266829\ 10^{-2}$*	*$5.3\ 10^{-15}$*	*1.7*
TAPENADE	$-2.092290564266826\ 10^{-2}$	$3.7\ 10^{-14}$	2.1
AD02	$-2.092290564266833\ 10^{-2}$	$4.0\ 10^{-14}$	4.4
Finite Differences	$-2.092290785071782\ 10^{-2}$	$2.8\ 10^{-6}$	4.5

*analytic computation of STMs do not take advantage of symmetry, time could likely be reduced in half, but effort is nontrivial

Test Case: Gravity field

- Gravity field derivatives
  - 20x20 Lunar Gravity Field
  - Up to third-order
  - Useful for satellite geodesy and trajectory optimization



Method	Sample 3rd-order Sensitivity	Maximum Relative Difference with Analytic (across all 3rd order terms)	Total Relative Computational Time
Analytical	$-4.239541972305253 \cdot 10^{-12}$	NA	1.0
MultiComplex-Step	$-4.239541972305250 \cdot 10^{-12}$	$6.6 \cdot 10^{-15}$	20.9
TAPENADE	$-4.239541972305257 \cdot 10^{-12}$	$2.6 \cdot 10^{-15}$	30.1
AD02	$-4.239541972305255 \cdot 10^{-12}$	$2.9 \cdot 10^{-15}$	154.9

New Sensitivity Landscape

	FD	Analytical	MCX	AD*
Compute Speed	Slow	Fast	Medium	Medium(1) / slow (2)
Ease of implementation	Easiest	Hardest	Medium	Medium (1)/Hard(2)
Accuracy	Poor	Near Exact**	Near Exact **	Near Exact **
Special requirements	None: function call CAN BE a library or "black box"	Function call and derivatives *must* be consistent. CAN BE a library or "black box"	MCX module +function source	AD module +function source

*only considered TAPENADE (1) & AD02 (2) of the many AD toolboxes available
** subject to round off, order of operations, dynamic variable range errors, but not subtraction error

Conclusion

- MultiComplex-Step method was developed for computation of partial derivatives up to any order: extension of complex step method to any order

- MultiComplex-Step differentiation combines the best of finite difference, complex, and automatic differentiation

- Further increasing in tool flexibility is possible through:
  - Prototype Modules for Fortran and Matlab
  - Developing a script to automatically process source codes
  - Matrix and array operations in Matlab

Thank you !

- We want your feedback !!