AIAA Houston Section Guidance, Navigation & Control: Technical Committee "Lunch-and-Learn"

Emerging Software Tools for Satellite Design

James D. Turner, Ph.D

Director of Operations, Consortium for Autonomous Space Systems

Texas A&M University, Department of Aerospace Engineering, College Station Texas, 77843

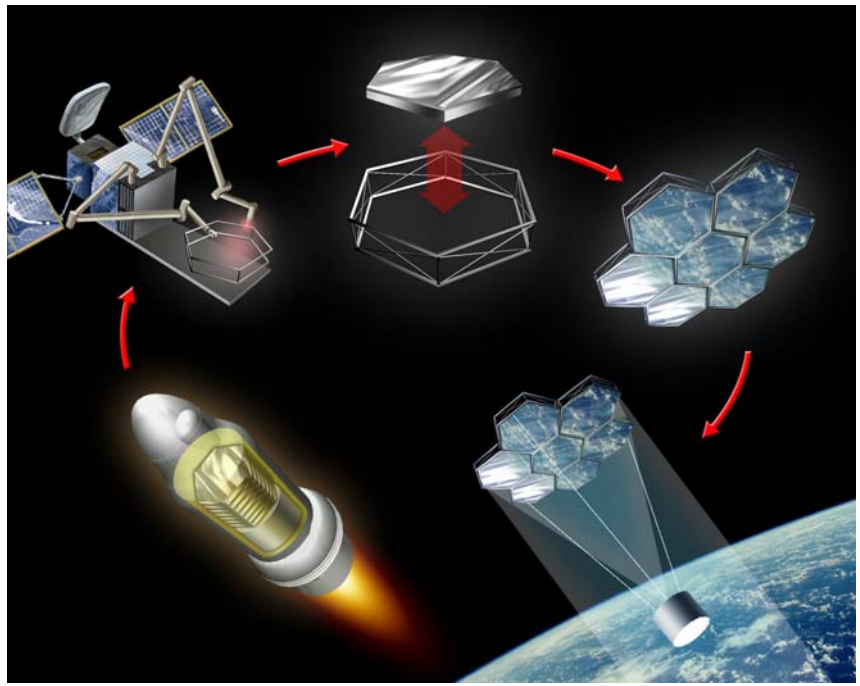
Research Associate Professor, Senior Member AIAA. turner@aero.tamu.edu, 979-458-1429

Principle Features

1. **Fortran 95, *Object Oriented & Operator - Overloading***
2. ***Quaternion Toolbox* for Engineering & Science Applications**
3. ***OCEA – Automatic Differentiation* for Application Development & Generalized Optimization**

Outline

- **Why Fortran 95/2003 for New Tool Development?**
  - Advanced Language Features
- **Quaternion ToolBox**
  - Standard Libraries
  - Linear Equations & Quadratic Polynomials
  - Matrix Capabilities
- **OCEA – AD**
  - Automatic Differentiation – (1 – 4) Order Mixed Partials
  - Scalar, Vector, & Matrix Operations
  - No User Intervention Required for Building Partials
- **Conclusions**



Goal: Design & Optimization of Complex Aero Space Systems

Why Fortran 95/2003 for New Tool Development?

- **Why Not Matlab???**
  - ***Matlab – Like*** Ease & ***Porsche – Like*** Speed
- **Advanced Language Features**
  - User - Defined Operators
  - Application-Specific Language Extensions
    - Operator – Overloading (**Big Deal**)
    - Highly Efficient Exploitation of Math Models
    - Compact & Readable Code
- **Module - Centric Development Environment**
  - Dramatically Minimizes F77- Memory Problems
  - Powerful Debugging for Error Tracking
  - Hide Details from User

Why Go Slow When You Can Go Really FAST!

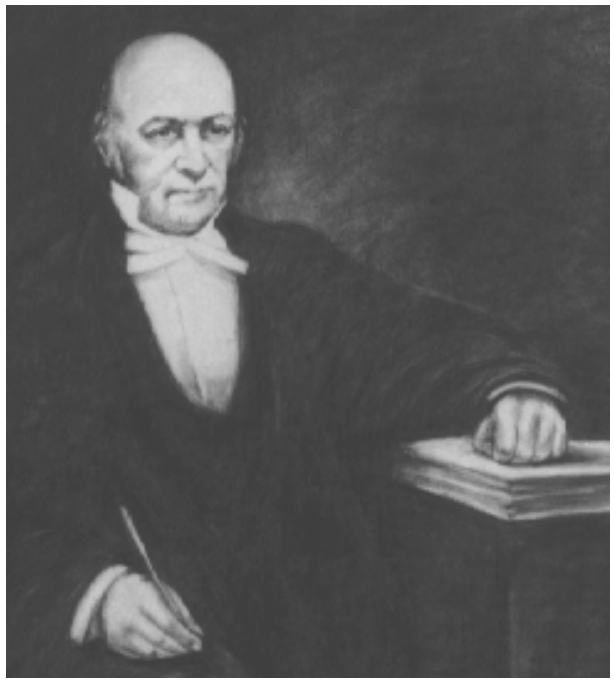
Object – Oriented Data Structures

- Abstract Compound Data Objects
  - Single Variable Embeds All Information about Variable!
    - Size, Type, Dimension(s), String Data Descriptions, Properties, Algebra to be used in Calculations, Multi-Level Derivative Data, Math Properties, Calculational Strategy, etc. ……
  - Hidden Sub-Object Operations Can Be Defined.

$$A := \{A \quad \nabla A \quad "PositiveDefiniteSymmetric" \quad "Sparse" \quad "Cholesky"\}$$

Quaternion ToolBox: Outline

- **Discovery of Quaternions**
- **Math Properties**
  - Intrinsic
  - Math Libraries
- **Algorithms**
  - Linear Scalar Equations
  - Quadratic Equations
  - Matrix Equations
    - Linear Matrix Equations, Inversion, Eigensolutions
- **Quatpack95 "Quaternion Toolbox"**
  - Operator-Overloading
  - Generic Interfaces
- **Applications**
- **Conclusions**



Sir William Rowan Hamilton (1805-1865)

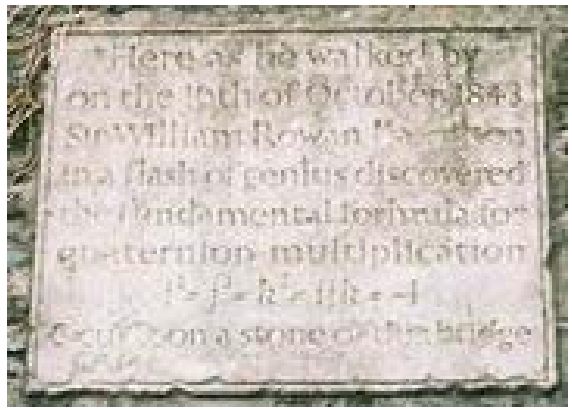
13th November, 1843 he presented a paper, *On a new Species of Imaginary Quantities connected with a theory of Quaternions*,

Quaternions

- Discovered by Hamilton
  - October 16, 1843 While Walking to a meeting with the Irish Academy, he recorded his discovery by carving the following equation on *Brougham Bridge in Dublin*

$$i^2 = j^2 = k^2 = ijk = -1$$

  - Hamilton's Goal
    - Generalize Complex Analysis To 3D



Quaternion plaque on Brougham (Broom) Bridge, Dublin

Quaternions: Who Cares??

- **Aerospace**
  - Control and Attitude Determination
  - Robotics, Orbital Mechanics
- **Computer Science**
  - Graphics, 3D- Rotations
  - Signal Processing
- **Physics**
  - Electro-Magnetic Theory
  - Fluid Mechanics
  - Quantum Mechanics
  - Relativistic Mechanics

Quaternion Math Properties

- Basic Definition

$$\mathbf{a} = \underbrace{a_0}_{Scalar} + \underbrace{a_1\hat{\mathbf{i}} + a_2\hat{\mathbf{j}} + a_3\hat{\mathbf{k}}}_{Vector} = (a_0, \vec{a})$$

$$\mathbf{a} + \mathbf{b} = \left(a_0 + b_0, \quad \vec{a} + \vec{b}\right)$$

$$\mathbf{a} - \mathbf{b} = \left(a_0 - b_0, \quad \vec{a} - \vec{b}\right)$$

$$\mathbf{ab} = \left(a_0, \quad \vec{a}\right)\left(b_0, \quad \vec{b}\right) = \left(a_0 b_0 - \vec{a} \bullet \vec{b}, \quad a_0\vec{b} + b_0\vec{a} + \vec{a}\, x\, \vec{b}\right)$$

Quaternion Math Properties Cont.

- Historical Note
  - Product Rule Originally Found in *Component Form*.
  - *Dot* and *Cross Products* Not Introduced Until 1881 by Willard Gibbs.

- Quaternion Complexity
  - Cross Product Creates *Noncommutivity*

$$\mathbf{ab} - \mathbf{ba} = 2\vec{a} \times \vec{b}$$

All Quaternion **Blessings & Curses** Come From Here

Elementary Operations

$$\mathbf{a}^* = \left(a_0, \quad -\vec{a}\right)$$ Conjugation

$$\left|\mathbf{a}\right| = \mathbf{a}\,\mathbf{a}^* = a_0^2 + a_1^2 + a_2^2 + a_3^2$$ Magnitude

$$\mathbf{a}^{-1} = \frac{\mathbf{a}^*}{\left|\mathbf{a}\right|}$$ Reciprocal

Alternate Representations for Validation Efforts

- Tensor and Matrix Forms

$$\mathsf{a} = q_0 + q_1\hat{\mathbf{i}} + q_2\hat{\mathbf{j}} + q_3\hat{\mathbf{k}}$$

$$= \begin{bmatrix} q_0 + q_1\hat{\mathbf{i}} & q_2 + q_3\hat{\mathbf{i}} \\ -q_1 + q_3\hat{\mathbf{i}} & q_0 - q_1\hat{\mathbf{i}} \end{bmatrix}$$

Useful For Matrix Validation

- Integral Checks

$$\ln(\mathsf{a}) = \int_1^{\mathsf{a}} \frac{\mathrm{d}\,\mathsf{q}}{\mathsf{q}}; \quad \sin^{-1}\mathsf{a} = \int_0^{\mathsf{a}} \frac{\mathrm{d}\,\mathsf{q}}{\sqrt{1-\mathsf{q}^2}}$$

Simpson's Rule Effective

Principal Values Also Apply For Quaternions

Library Functions

- Standard Approach
  - Addition Rules & Power Series
  - Integral Representations

- Exponential Quaternion Example

$$\exp(\mathbf{a}) = \exp(a_0 + \vec{a}) = \exp(a_0)\exp(\vec{a})$$

  - Must Evaluate ***Exponential of a Vector*** to Complete Quaternion Exponential

Exponential of a Vector

- Taylor Expansion

$$\exp(w) = 1 + w + \frac{w^2}{2!} + \frac{w^3}{3!} + \frac{w^4}{4!} + \cdots$$

$$\Rightarrow\ w = \vec{a} = |\vec{a}|\frac{\vec{a}}{|\vec{a}|}$$

Need Unit Vector

$$|\vec{a}| = \sqrt{\left(a_1^2 + a_2^2 + a_3^2\right)}$$

Exponential of a Vector Cont.

- Substituting & Collecting Terms

$$\exp\left(|\vec{a}|\frac{\vec{a}}{|\vec{a}|}\right)=1+|\vec{a}|\frac{\vec{a}}{|\vec{a}|}+\frac{|\vec{a}|^2}{2!}\frac{\vec{a}}{|\vec{a}|}\frac{\vec{a}}{|\vec{a}|}+\frac{|\vec{a}|^3}{3!}\frac{\vec{a}}{|\vec{a}|}\frac{\vec{a}}{|\vec{a}|}\frac{\vec{a}}{|\vec{a}|}+\cdots$$

$$=1+|\vec{a}|\frac{\vec{a}}{|\vec{a}|}-\frac{|\vec{a}|^2}{2!}-\frac{|\vec{a}|^3}{3!}\frac{\vec{a}}{|\vec{a}|}+\cdots; \quad \left(\text{Key Identity: } \frac{\vec{a}}{|\vec{a}|}\frac{\vec{a}}{|\vec{a}|}=-1\right)$$

$$=\left(\left(1-\frac{|\vec{a}|^2}{2!}+\frac{|\vec{a}|^4}{4!}-\cdots\right),\left(|\vec{a}|-\frac{|\vec{a}|^3}{3!}+\frac{|\vec{a}|^5}{5!}-\cdots\right)\frac{\vec{a}}{|\vec{a}|}\right) \quad (\text{Collecting Terms})$$

$$=\left(\cos(|\vec{a}|),\sin(|\vec{a}|)\frac{\vec{a}}{|\vec{a}|}\right)$$

Generalization of Complex Variable Exponential

Generalization of $i^2 = -1$ Identity

- Product of Unit Vectors Identity

$$\frac{\vec{a}}{|\vec{a}|}\frac{\vec{a}}{|\vec{a}|} = \left(0, \quad \frac{\vec{a}}{|\vec{a}|}\right)\left(0, \quad \frac{\vec{a}}{|\vec{a}|}\right) = \left(0 - \frac{\vec{a}}{|\vec{a}|}\bullet\frac{\vec{a}}{|\vec{a}|}, 0 + 0 + \frac{\vec{a}}{|\vec{a}|}x\frac{\vec{a}}{|\vec{a}|}\right) = \left(-1, \vec{0}\right)$$

- Trig Identities for Vector Calculations

$$\cos(\vec{a}) = 1 - \frac{\vec{a}\vec{a}}{2!} + \frac{\vec{a}\vec{a}\vec{a}\vec{a}}{4!} - \cdots$$

$$= 1 + \frac{|a|^2}{2!} + \frac{|a|^4}{4!} \cdots = \cosh(|a|)$$

$$\sin(\vec{a}) = \vec{a} - \frac{\vec{a}\vec{a}\vec{a}}{3!} + \frac{\vec{a}\vec{a}\vec{a}\vec{a}\vec{a}}{5!} - \cdots$$

$$= \left(|\vec{a}| + \frac{|a|^3}{3!} + \frac{|a|^5}{5!} \cdots\right)\frac{\vec{a}}{|\vec{a}|} = \sinh(|a|)\frac{\vec{a}}{|\vec{a}|}$$

Linear Quaternion Equation:

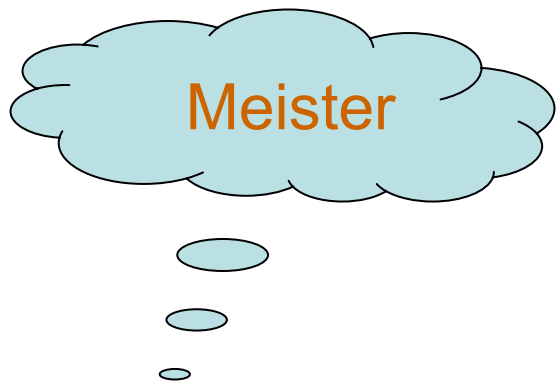
$$a x + x b = c$$

- Two Solution Approaches
  - Matrix – Based **=> Matrix N.C.'s**
  - Nonlinear Transformation **=> Scalar N.C.'s**
- Matrix Algorithm

$$a x + x b = c$$

$$[A]\,vec(x) + [X]\,vec(b) = vec(c)$$

$$[A]\,vec(x) + [B']\,vec(x) = vec(c)$$



Kronecker Product Like solution

$$\underbrace{vec(x)}_{4x1} = \underbrace{\{[A] + [B']\}^{-1}}_{4x4}\,\underbrace{vec(c)}_{4x1}$$

Nonlinear Transformation Method

(To Appear as Tech. Note AIAA JGNC, Turner)

- Scalar N.C.

$$a x + x b = c$$

- Expand Quaternion Scalar & Vector Parts

$$\begin{pmatrix} \left(a_0 + b_0\right) x_0 - \left(\vec{a} + \vec{b}\right) \cdot \vec{x} + c_0 \\ \left(a_0 + b_0\right) \vec{x} + \left(\vec{a} + \vec{b}\right) x_0 + \widetilde{\vec{a} - \vec{b}} \cdot \vec{x} + \vec{c} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{0} \end{pmatrix}$$

Nonlinear Transformation Method Cont.

- Uncouple Scalar & Vector Part

Uncoupling Transformation: (Analytic)

$$\vec{x} = -M^{-1}\left(\vec{c} + \left(\vec{a} + \vec{b}\right)x_0\right)$$

Introduce Scalar Solution to Complete Vector Solution

Closed-Form Soln For Scalar Part

$$x_0 = \frac{-\left\{c_0 + \left(\vec{a} + \vec{b}\right)\cdot M^{-1}\cdot \vec{c}\right\}}{\left(a_0 + b_0\right) + \left(\vec{a} + \vec{b}\right)\cdot M^{-1}\cdot\left(\vec{a} + \vec{b}\right)}$$

$$M = \left[\left(a_0 + b_0\right)I_{3x3} + \widetilde{\vec{a} - \vec{b}}\right]$$

Coefficient Coupling Matrix

Nonlinear Transformation Method Cont.

Analytic Singularities for Solution

- $\mathbf{M^{-1}}$: Check When Matrix is Singular

$$\left[(a_0 + b_0) I_{3x3} + \widetilde{\vec{a} - \vec{b}} \right]^{-1} = \frac{\frac{(\vec{a} - \vec{b})(\vec{a} - \vec{b})^t}{(a_0 + b_0)} + (a_0 + b_0)\ I_{3x3} - \left(\widetilde{\vec{a} - \vec{b}} \right)}{(\vec{a} - \vec{b}) \cdot (\vec{a} - \vec{b}) + (a_0 + b_0)^2}$$

Inverse Only Singular When $a_0 + b_0 = 0$

Special Cases

Case	Coefficient Assumptions	Necessary Conditions	Solution
1	$a=b,\ a_0\neq 0,\ \&\ c\neq 0$	$\begin{pmatrix} 2a_0x_0-2\vec{a}\cdot\vec{x}+c_0 \\ 2a_0\vec{x}+2\vec{a}x_0+\vec{c} \end{pmatrix}=\begin{pmatrix} 0 \\ \vec{0} \end{pmatrix}$	$x_0=-\dfrac{c_0a_0+\vec{a}\cdot\vec{c}}{2\left(a_0^2+\vec{a}\cdot\vec{a}\right)}$ $\vec{x}=\dfrac{-\left(2\vec{a}x_0+\vec{c}\right)}{2a_0}$
2	$a=b,\ a_0\neq 0,\ \&\ c=0$	$\begin{pmatrix} 2a_0x_0-2\vec{a}\cdot\vec{x}, \\ 2a_0\vec{x}+2\vec{a}x_0 \end{pmatrix}=\begin{pmatrix} 0 \\ \vec{0} \end{pmatrix}$	$x_1=\hat{\imath}-\dfrac{a_1}{a_3}\hat{k}$ New Solution $x_2=\hat{\jmath}-\dfrac{a_2}{a_3}\hat{k}$ Meister[7,8]
3	$a=b,\ a_0=0,\ \&\ \vec{a}=-\vec{c}$	$\begin{pmatrix} -2\vec{a}\cdot\vec{x}-c_0 \\ 2x_0\vec{a}+\vec{c} \end{pmatrix}=\begin{pmatrix} 0 \\ \vec{0} \end{pmatrix}$	$x_0=\dfrac{\vec{c}\cdot\vec{x}}{c_0}$ and $\vec{a}=\dfrac{-\vec{c}}{2x_0}$ EG. Let: $x_0=\frac{1}{2};\ \vec{a}=-\vec{c}$ $\Rightarrow x=\left(1-\dfrac{c_0\vec{c}}{\vec{c}\cdot\vec{c}}\right)/2$
4	$b=-a\ \&\ c=0$	$\begin{pmatrix} 0, \\ \sim_{2\vec{a}}\ \cdot\vec{x} \end{pmatrix}=\begin{pmatrix} 0 \\ \vec{0} \end{pmatrix}$	$x=x_0+b\vec{a}$ Meister[7,8]
5	$b=-a\ \&\ c_0=0,\ \vec{c}\neq 0$	$\begin{pmatrix} 0, \\ \sim_{2\vec{a}}\ \cdot\vec{x}+\vec{c} \end{pmatrix}=\begin{pmatrix} 0 \\ \vec{0} \end{pmatrix}$	No Solution Exists

Quadratic Polynomial Equation

$$p^2 + ap + pb + c = 0$$

Case 1: $a = b$ (Complete the Square)

$$p^2 + ap + pa + c = (p + a)(p + a) + c - a^2 = 0$$

$$p^{\pm} = -a \pm \sqrt{a^2 - c}$$

Closed Form Solution: Well Defined Numerics

General Quadratic Equation

Method 1: $p^2 + ap + pb + c = 0$

- **Case:** $a \neq b,$

Necessary Condition

$$\begin{pmatrix} p_0^2 - \vec{p}\cdot\vec{p} + (a_0 + b_0)\, p_0 - (\vec{a} + \vec{b})\cdot\vec{p} + c_0, \\ \\ 2\, p_0 \vec{p} + (a_0 + b_0)\,\vec{p} + (\vec{a} + \vec{b})\, p_0 + \widetilde{\vec{a} - \vec{b}}\cdot\vec{p} + \vec{c} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{0} \end{pmatrix}$$

- **Nonlinear Transformation**

$$\vec{p} = -\left[\left\{ \underbrace{2\, p_0}_{New\ Term} + (a_0 + b_0) \right\} I_{3x3} + \widetilde{\vec{a} - \vec{b}} \right]^{-1} \left(\vec{c} + (\vec{a} + \vec{b})\, p_0 \right)$$

Quadratic Polynomial Equation Cont.

- Single Nonlinear Scalar Equation

$$p_0^2 + \left(a_0 + b_0\right) p_0 - \left(\vec{a} + \vec{b}\right) \cdot \vec{p} - \vec{p} \cdot \vec{p} + c_0 = 0$$

- Quadratic Matrix Singularity Near

$$p_0 = -\left(a_0 + b_0\right) / 2$$

- **Problem**: Desired Roots are in the Neighborhood of the Singularity!!!!!

Quadratic Polynomial Equation

Analytically Eliminate Singularity

- **Approach**: Replace Inverse by Adjoint & Det

$$\vec{p} = -[\mathbf{M}]^{-1}\left(\vec{c} + \left(\vec{a} + \vec{b}\right)p_0\right) = -\frac{Adjoint(\mathbf{M})}{det(\mathbf{M})}\left(\vec{c} + \left(\vec{a} + \vec{b}\right)p_0\right)$$

$$\det(\mathbf{M}) = \left(2p_0 + a_0 + b_0\right)\left\{\left(\vec{a} - \vec{b}\right)\cdot\left(\vec{a} - \vec{b}\right) + \left(2p_0 + a_0 + b_0\right)^2\right\}$$

$$Adjoint(\mathbf{M}) = \left(\vec{a} - \vec{b}\right)\left(\vec{a} - \vec{b}\right)^t + \left(2p_0 + a_0 + b_0\right)^2 I_{3x3} - \left(2p_0 + a_0 + b_0\right)\begin{pmatrix}\sim \\ \vec{a} - \vec{b}\end{pmatrix}$$

Quadratic Polynomial Equation

Analytically Eliminate Singularity

- **Singularity – Free** Scalar Polynomial

$$\det(\mathrm{M})^2\left(p_0^2+\left(a_0+b_0\right)p_0+c_0\right)-\det(\mathrm{M})\left(\vec{a}+\vec{b}\right)\cdot\vec{y}-\vec{y}\cdot\vec{y}=0$$

$$\vec{y}=-\mathrm{Adjoint}(\mathrm{M})\left(\vec{c}+\left(\vec{a}+\vec{b}\right)p_0\right)$$

- **Solution Approach**: Start at $p_0=-\left(a_0+b_0\right)/2$
  - Line Search for Sign Change in Polynomial Value
  - Newton's Method to Polish First Root Value
  - Estimate Other Root Based on Symmetry
  - Newton's Method to Polish Second Root Value

Quadratic Polynomial Equation

Method 2: Homotopy Chain Method

- **Strategy:**
  - Exploit Closed-Form Completing-The-Square Solution
  - Embed Artificial Parameter "s"
  - Assume p = p(s)

$$P^2 + aP + P\left(\underbrace{a + s(b - a)}_{\textit{Embedding Function}}\right) + c = 0$$

Quadratic Polynomial Equation

Homotopy Chain Method: Rate Equation

- **Compute Rate Equation**

$$(P+a)P_{,s}+P_{,s}\left(P+a+s(b-a)\right)+P(b-a)=0$$

- **Solve Rate Equation (Linear Equation)**
- **Integrate Along Homotopy Path**

$$p\big|_{s=1}=p\big|_{s=0}+\int_0^1 p_{,s}\,ds,\quad p\big|_{s=0}=-a\pm\sqrt{a^2-c}$$

- **50 Integration Steps ~ 5-10 digit accuracy**
- **Valid As Long As Linear Equation Accurate**

Quaternion Matrix Equations

- **Matrix Reductions:** Must Respect *Order Dependence Of Operations*
  - Easily Implemented for Gaussian Elimination, Cholesky & LU Decomposition, etc.
- Dimension Doubling Algorithm

$$\underbrace{A}_{nxn}\ \underbrace{x}_{nx1} = \underbrace{b}_{nx1}$$

$$\left[A_0 + A_1\hat{\mathbf{i}} + A_2\hat{\mathbf{j}} + A_3\hat{\mathbf{k}}\right]\left(x_0 + x_1\hat{\mathbf{i}} + x_2\hat{\mathbf{j}} + x_3\hat{\mathbf{k}}\right) = b_0 + b_1\hat{\mathbf{i}} + b_2\hat{\mathbf{j}} + b_3\hat{\mathbf{k}}$$

$$\left[\left(A_0 + A_1\hat{\mathbf{i}}\right) + \left(A_2 + A_3\hat{\mathbf{i}}\right)\hat{\mathbf{j}}\right]\left(\left(x_0 + x_1\hat{\mathbf{i}}\right) + \left(x_2 + x_3\hat{\mathbf{i}}\right)\hat{\mathbf{j}}\right) = \left(b_0 + b_1\hat{\mathbf{i}}\right) + \left(b_2 + b_3\hat{\mathbf{i}}\right)\hat{\mathbf{j}}$$

M_1 & $M_2 \in C^{nxn}$

$$\left[M_1 + M_2\hat{\mathbf{j}}\right]\left(y_1 + y_2\hat{\mathbf{j}}\right) = d_1 + d_2\hat{\mathbf{j}}$$

Still Quaternion !!!

Quaternion Matrix Equations

Double Dimension Reductions

- Simplify Quaternion Operations

$$\left[M_1 + M_2\hat{\mathbf{j}}\right]\left(y_1 + y_2\hat{\mathbf{j}}\right) = d_1 + d_2\hat{\mathbf{j}}$$

$$M_1 y_1 + M_1 y_2\hat{\mathbf{j}} + M_2\hat{\mathbf{j}}y_1 + M_2\hat{\mathbf{j}}y_2\hat{\mathbf{j}} = d_1 + d_2\hat{\mathbf{j}}$$

$$M_1 y_1 + M_1 y_2\hat{\mathbf{j}} + M_2 y_1^*\hat{\mathbf{j}} - M_2 y_2^* = d_1 + d_2\hat{\mathbf{j}}$$

$$M_1 y_1 + M_2\left(-y_2^*\right) - M_2^* y_1\hat{\mathbf{j}} + M_1^*\left(-y_2^*\right)\hat{\mathbf{j}} = d_1 - d_2^*\hat{\mathbf{j}}$$

Pure Complex

$$\underbrace{\begin{bmatrix} M_1 & M_2 \\ -M_2^* & M_1^* \end{bmatrix}}_{2nx2n}\underbrace{\begin{pmatrix} y_1 \\ -y_2^* \end{pmatrix}}_{2nx1} = \underbrace{\begin{pmatrix} d_1 \\ -d_2^* \end{pmatrix}}_{2nx1}$$

Standard S/W Library Works!

Quaternion Eigensolution

- **Two Classes of Eigensolutions**
  - Right $Ax = x\lambda$
  - Left $Ax = \lambda x$ **Extremely Difficult, Not Considered Here**
- **Eigensolution Properties**
  - Solving $Ax = x\lambda \quad \Rightarrow A(xw) = (xw)\{(1/w)\lambda w\}$

$$\Rightarrow xw \text{ is Eigenvector } \& \Rightarrow (1/w)\lambda w \text{ is Eigenvalue}$$

One can choose a preferred unique eigenvalue, say in upper half complex plane

Quaternion Eigensolution Cont.

- Alternate Forms (Reductions Follow Dimension Doubling Algorithm)

$$\begin{bmatrix} A_1 & A_2 \\ -A_2^* & A_1^* \end{bmatrix} \begin{pmatrix} x_1 \\ -x_2^* \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ -x_2^* \end{pmatrix} \quad \text{OR} \quad \begin{bmatrix} A_1 & A_2 \\ -A_2^* & A_1^* \end{bmatrix} \begin{pmatrix} x_2 \\ x_1^* \end{pmatrix} = \lambda^* \begin{pmatrix} x_2 \\ x_1^* \end{pmatrix}$$

$$\mathsf{A} = A_1 + A_2 j \qquad \mathsf{x} = x_1 + x_2 j \qquad \lambda = \lambda_1 + \lambda_2 i$$

$$A_1 \,\&\, A_2 \in \mathbb{C}^{nxn} \qquad x_1 \,\&\, x_2 \in \mathbb{C}^{nx1} \qquad \lambda_1 \,\&\, \lambda_2 \in \mathbb{C}$$

Matlab Routines Successfully Solve This Problem

QuatPack95

- Fortran 95/2003 Object-Oriented Operator Overloaded Tool for Computing Quaternions
- *Abstract Compound Data Objects* for Quaternion Data Structures

$$q = q_0 + \vec{q} = q\%s + q\%v$$

- Supported Intrinsic Operations

$$d = a + b;\ d = a - b;\ d = a * b;\ d = a/b$$

QuatPack95 Cont.

- Supported Library Functions

$b = .conj.a\ (conjugate);\ \ b = \sqrt{a};\ \ b = \log(a)\ or\ \exp(a);\ \ b = a**n$

$b = \sin\ or\ \sin^{-1}(a);\ \ b = \cos\ or\ \cos^{-1}(a);\ \ b = \tan\ or\ \tan^{-1}(a)$

$b = \sinh\ or\ \sinh^{-1}(a);\ \ b = \cosh\ or\ \cosh^{-1}(a);\ \ b = \tanh\ or\ \tanh^{-1}(a)$

$b = a.d.c\ (vector\ dot\ product);\ \ b = a.x.c\ (vector\ cross\ product)$

$b = .tilde.a\ (3x3\ matrix\ representation\ of\ vector\ cross\ product)$

$b = .abs.a\ (absolute\ value, also\ vector\ norm\ when\ a\ is\ a\ vector)$

$b = .inv.a\ (invert\ matrix\ a, where\ a\ is\ DP\ or\ quaternion)$

$c = a.AXeB.b\ \ (solve\ linear\ matrix\ equation, where\ a\ is\ DP\ or\ quaternion)$

- Example: Compute $AA^{-1} - I = 0$

```
check = A *.inv.A - Q_EYE(size(A))
```

QuatPack95 Statement

Identity Matrix

Applications: Linear Equations

- **Assumed Equation Coefficients**

$$\mathbf{a} = 1 - 2\hat{\imath} + \hat{\jmath} + 2\hat{k}; \quad \mathbf{b} = -2 + 3\hat{\imath} + 2\hat{\jmath} + \hat{k}; \quad \mathbf{c} = -1 - 2\hat{\imath} + 3\hat{\jmath} - 4\hat{k}$$

$$\mathbf{a}x + x\mathbf{b} = \mathbf{c}: \quad x = \frac{3}{2} + \frac{1}{2}\hat{\imath} + \frac{1}{2}\hat{\jmath} - \frac{3}{2}\hat{k}$$

$$\mathbf{a}x + x\mathbf{a} = \mathbf{c}: \quad x = -0.6928 - 1.7320\hat{\imath} - 0.8660\hat{\jmath} - 2.0784\hat{k}$$

- **Solution Accuracy**: **~ Double Precision!!!**

Applications: Matrix

- 3x3 Non-Symmetric Matrix
  - Compute $x = .inv.A * b$ & $x = A.AXeB.b$

$$\underset{3x3}{\underline{\underline{A}}} = \begin{bmatrix} (1 & 2 & 3 & 4) & (2 & -1 & 5 & 10) & (-3 & 9 & 4 & -2) \\ (6 & 2 & -3 & 1) & (4 & 4 & 1 & 1) & (-1 & 0 & 2 & 1) \\ (4 & -3 & -1 & 5) & (8 & -6 & 9 & 2) & (-10 & 2 & 4 & 11) \end{bmatrix}$$

$$\underset{3x1}{\underline{\underline{b}}} = \begin{pmatrix} (1 & 1 & 1 & 1) \\ (2 & 4 & -2 & -4) \\ (-5 & -3 & -2 & 1) \end{pmatrix}$$

$$\underset{3x1}{\underline{\underline{x}}} = \begin{pmatrix} (0.59607 & 0.31582 & -1.39729 & 0.23591) \\ (0.28461 & 0.17402 & 0.74776 & -0.50731) \\ (0.18211 & 0.27463 & 0.48752 & 0.33893) \end{pmatrix}$$

Quaternion Gaussian Elimination, Only Quaternion Operations Used

Applications: Eigensolution

- Simple 2x2 Problem

$$A = \begin{bmatrix} 0 & 1-\hat{\mathbf{k}} \\ 1-\hat{\mathbf{k}} & 0 \end{bmatrix}$$

- Double Dimension Companion Matrix

$$C = \begin{bmatrix} 0 & 1 & 0 & -\hat{\mathbf{i}} \\ 1 & 0 & -\hat{\mathbf{i}} & 0 \\ 0 & -\hat{\mathbf{i}} & 0 & 1 \\ -\hat{\mathbf{i}} & 0 & 1 & 0 \end{bmatrix}$$

Applications: Eigensolution

Matlab Generated Solution

- [V,D] Denote Eigenvectors & Eigenvalues

$$V=\begin{bmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix};\quad D=\begin{bmatrix} -1+\hat{\mathbf{i}} & -1-\hat{\mathbf{i}} & 1+\hat{\mathbf{i}} & 1-\hat{\mathbf{i}} \end{bmatrix}$$

- Transforming to Quaternion Variables

$$\mathsf{v}_1=\begin{pmatrix} 1+\hat{\mathbf{j}} \\ 1+\hat{\mathbf{j}} \end{pmatrix};\qquad d_1=1+\hat{\mathbf{i}};\qquad \mathsf{v}_2=\begin{pmatrix} 1-\hat{\mathbf{j}} \\ -1+\hat{\mathbf{j}} \end{pmatrix};\quad d_2=-1+\hat{\mathbf{i}}$$

Which is Easily Checked by Computing $\mathsf{A}\mathsf{v}_i=\mathsf{v}_i d_i;\quad i=1,2$

Quaternion Summary

- Theoretical Underpinning for QuatPack95 Presented.
  - Object – Oriented & Operator – Loaded Quaternion Data Structures
  - Intrinsic and Math Library Functions
- New Results Presented for Scalar Linear and Quadratic Equations.
- Matrix Algorithms Presented and Demonstrated for Linear Equations, Matrix Inversion, and Eigen Solutions.
- QuatPack95 Available for Researchers From Author.
- Current Tool Developed Using Sweat Equity Funding.

Automatic Differentiation

- How the Tools Work
- Data Structures
- Application – Specific Algorithms
- Enabling Capabilities for Generialized Dynamics & Control Problems
- Examples

OCEA Overview

- **OCEA Algorithm**
- **How Does It Work?**
  - Addition & Subtraction
  - Generalized Composite Function
  - Mixed Symbolic / Numeric Development Strategy
- **Linear Matrix Equation**
  - Problem Formulation
  - OCEA Matrix Data Structure During Triangularization
- **Object – Oriented Simulation Environment**

OCEA Algorithm

Transform All Scalars To Abstract Compound Data Objects

$$g := \left(g,\ \underbrace{\nabla g,\ \nabla^2 g,\ \ldots}\right)$$

Differential n-Tuple

Hidden Artificial Dimensions for Building and Storing Partials Enabled with User-Defined Data Types

Initialize Independent Variables (δ_{ik} denotes kronecker delta symbol)

$$g_i := \left(g_i,\ \begin{bmatrix} \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ \vdots \\ \delta_{ni} \end{bmatrix},\ \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix},\ \cdots \right)$$

Operator Algebras

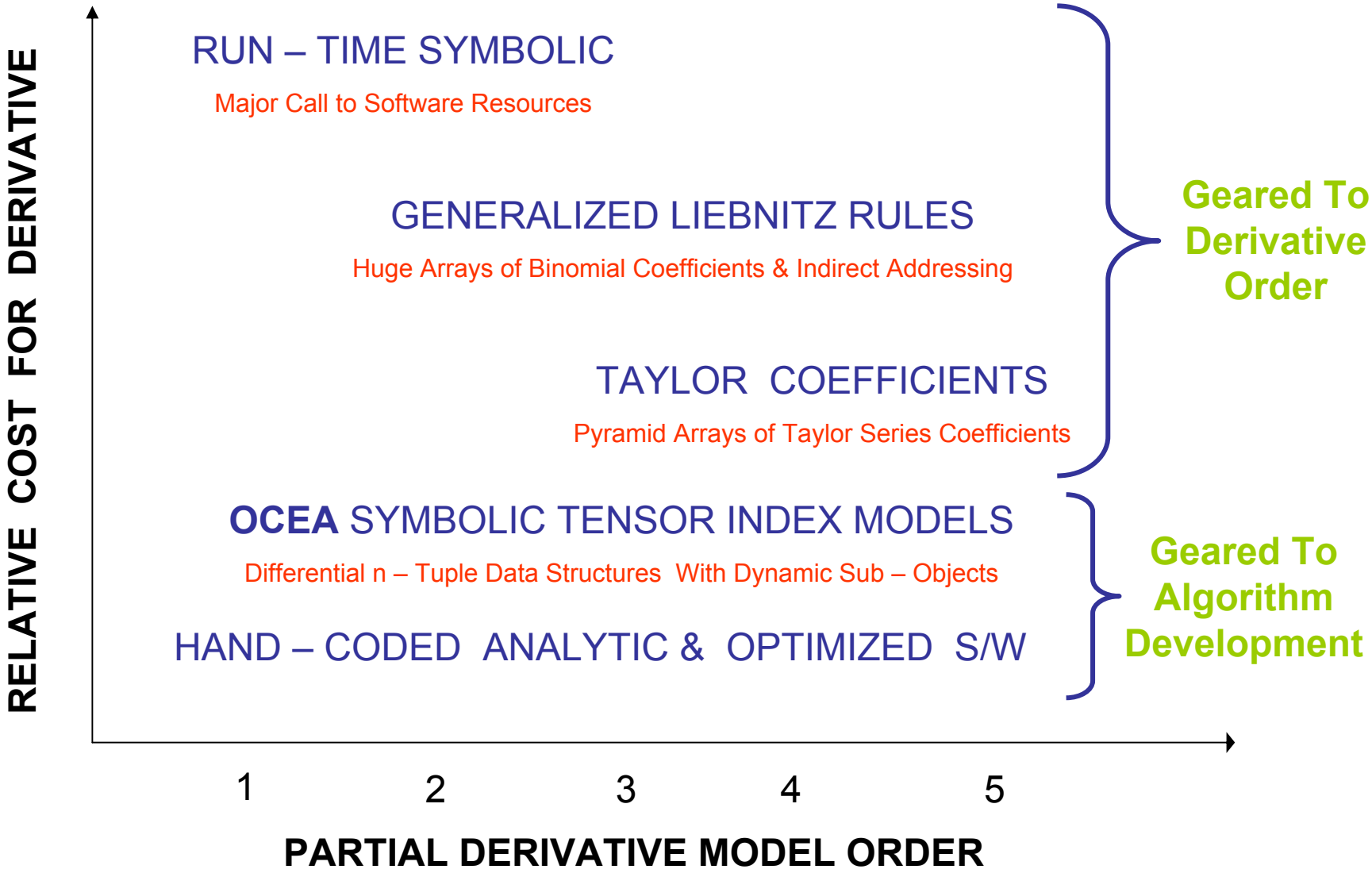
For Computer–Based Modeling

- **Operator – Overloading Strategies**
  - Element – Based => System – Level Operations
  - Matrix – Vector Operators (.inv., .eig., .T., .svd. …)
- **Language Extensions Tailored For Application – Specific Needs**
  - Block Structures, Symmetry, & Algorithm Exploitation
- **Dynamic Sparsity Detection & Computation**
  - Optimized Vector & Tensor Calculations
  - Automatic Memory Management for Data Structures & Sub-Object Data Types

Automated Machine-Based Derivation:
Numerical Partial Derivative Models

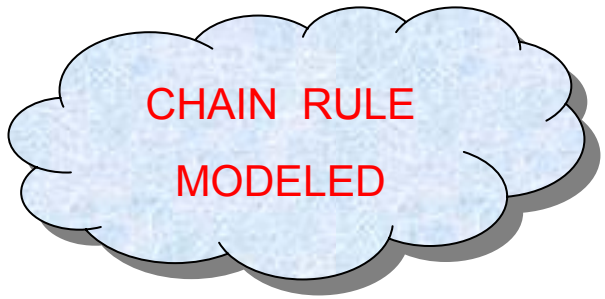
- ***Automatic Differentiation* (AD)** (Research Began ~1980)
  - Intrinsic Operators (+, -, *, **, /)
  - Mathematical Functions (sin(x), tan(x), sqrt(x), …)
  - Partials Accurate To Working Precision of Machine — User Hidden From Details
- ***Approaches***
  - **Source - to - Source** (Large Problems, Limited Data Structure Tools: *1st Order*,)
  - **Operator-Overloading** (Mid –Size Problems, Data Structures: *1st & 2nd Order*)
  - ***Object-Oriented Coordinate Embedding*** (**OCEA**) (Large Problems: *1-4th Order*)
    - Generalized Operator – Overloading (Data Structures & Algorithms)
    - Application – Specific Dynamic Algebras & Operators
    - Differential n-Tuple, Overloaded Sub-Object Operations
    - **Intelligent AD (Future of Automatic Differentiation)**

AD - METHODS


RELATIVE COST FOR DERIVATIVE
RUN – TIME SYMBOLIC
Major Call to Software Resources
GENERALIZED LIEBNITZ RULES
Huge Arrays of Binomial Coefficients & Indirect Addressing
TAYLOR COEFFICIENTS
Pyramid Arrays of Taylor Series Coefficients
Geared To Derivative Order
OCEA SYMBOLIC TENSOR INDEX MODELS
Differential n – Tuple Data Structures With Dynamic Sub – Objects
HAND – CODED ANALYTIC & OPTIMIZED S/W
Geared To Algorithm Development
1
2
3
4
5
PARTIAL DERIVATIVE MODEL ORDER

OCEA Addition & Subtraction

ADDITION: $a + b$



$$= \left[a+b, \quad \nabla a+\nabla b, \quad \nabla\nabla a+\nabla\nabla b, \quad \nabla\nabla\nabla a+\nabla\nabla\nabla b, \quad \nabla\nabla\nabla\nabla a+\nabla\nabla\nabla\nabla b \right]$$

SUBTRACTION: $a - b$

$$= \left[a-b, \quad \nabla a-\nabla b, \quad \nabla\nabla a-\nabla\nabla b, \quad \nabla\nabla\nabla a-\nabla\nabla\nabla b, \quad \nabla\nabla\nabla\nabla a-\nabla\nabla\nabla\nabla b \right]$$

Straightforward Model Development

Generalized Composite Function

Evaluate $b = b(a)$, where $b \in \{\text{sin, cos, tan, exp, ln, cosh, sinh, tanh, asin}, etc.\}$

$$b := \begin{bmatrix} b & \partial_i b & \partial_j \partial_i b & \partial_k \partial_j \partial_i b & \partial_r \partial_k \partial_j \partial_i b \end{bmatrix}, \text{ and}$$

SYMBOLICALLY GENERATED & CODED

SYMMETRIC TENSORS

$$\partial_i b = b' a_{,i}$$

$$\partial_j \partial_i b = b'' a_{,i} a_{,j} + b' a_{,i,j}$$

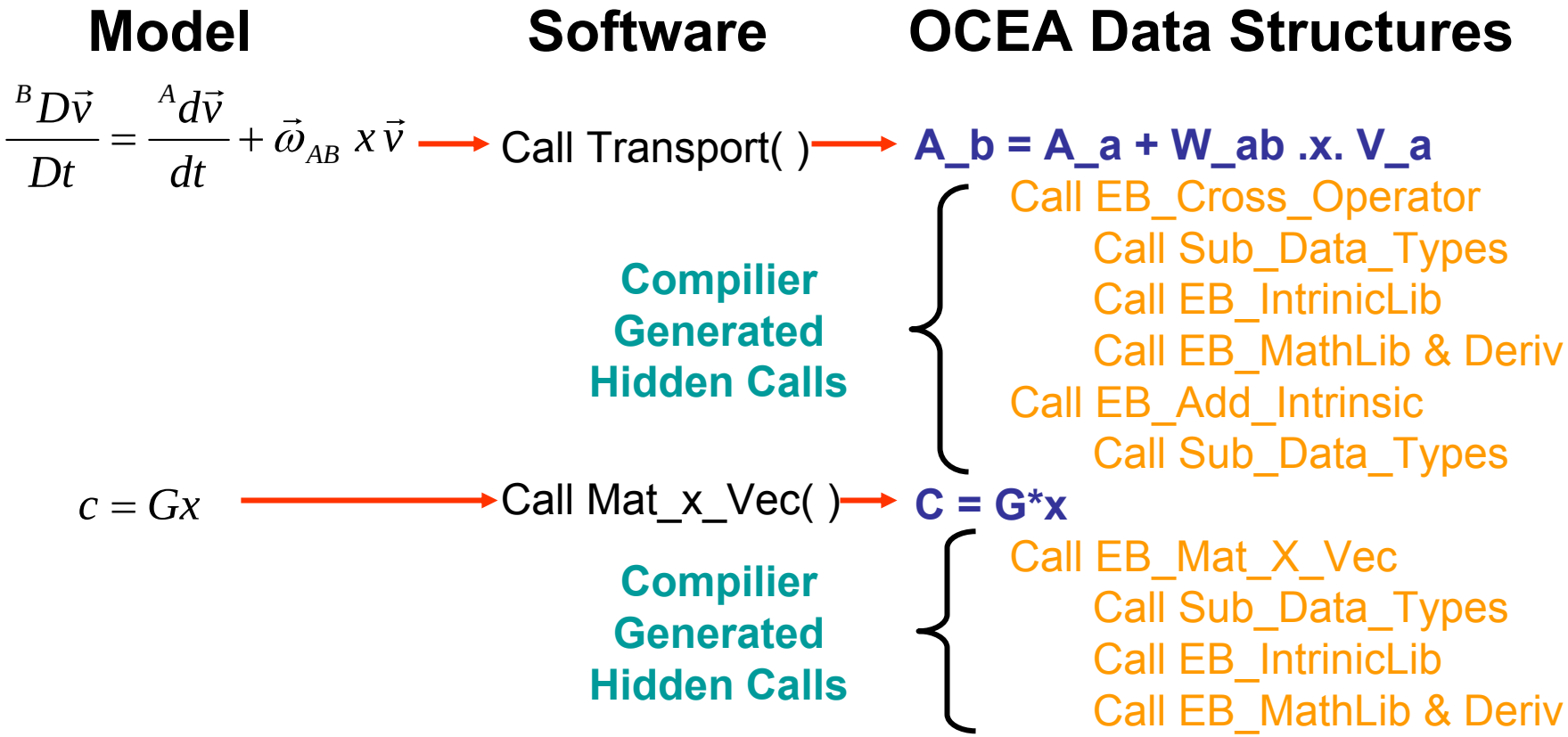
$$\partial_k \partial_j \partial_i b = b''' a_{,i} a_{,j} a_{,k} + b'' \left(a_{,i,k} a_{,j} + a_{,i} a_{,j,k} + a_{,i,j} a_{,k} \right) + b' a_{,i,j,k}$$

$$\partial_r \partial_k \partial_j \partial_i b = b'''' a_{,i} a_{,j} a_{,k} a_{,r} + b''' \left(a_{,i,k} a_{,j} + a_{,i} a_{,j,k} + a_{,i,j} a_{,k} \right) a_{,r} +$$
$$b''' \left(a_{,i,r} a_{,j} a_{,k} + a_{,i} a_{,j,r} a_{,k} + a_{,i} a_{,j} a_{,k,r} \right) +$$
$$b'' \left(a_{,i,k,r} a_{,j} + a_{,i,r} a_{,j,k} + a_{,i,j,r} a_{,k} + a_{,i,k} a_{,j,r} + a_{,i} a_{,j,k,r} + a_{,i,j} a_{,k,r} \right) +$$
$$b'' a_{,i,j,k} a_{,r} + b' a_{,i,j,k,r}$$

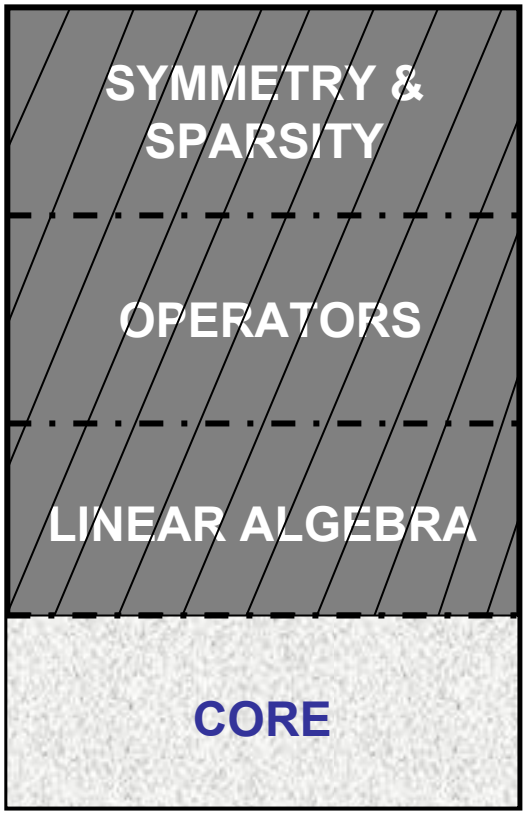
Where $\begin{bmatrix} b', & b'', & b''', & b'''' \end{bmatrix} = \begin{bmatrix} \frac{db}{da}, & \frac{d^2 b}{da^2}, & \frac{d^3 b}{da^3}, & \frac{d^4 b}{da^4} \end{bmatrix}$

Propagates Previous Partials to Current Sensitivity Calculations

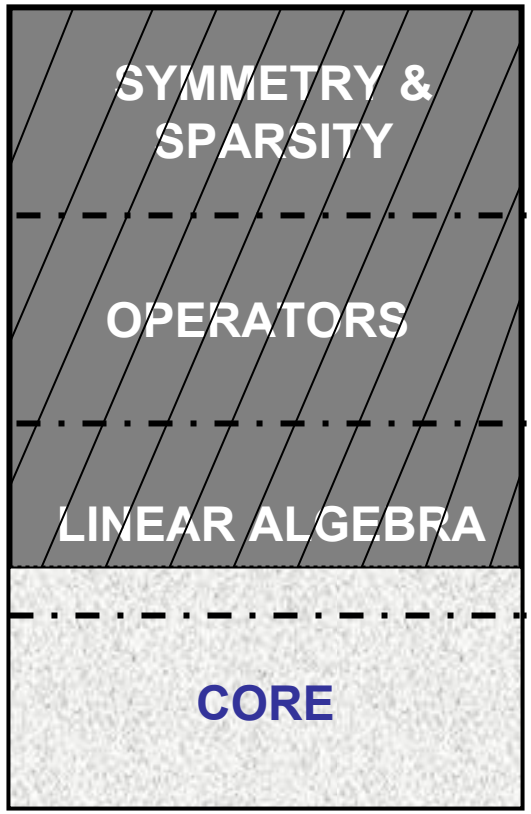
How Does It Work?



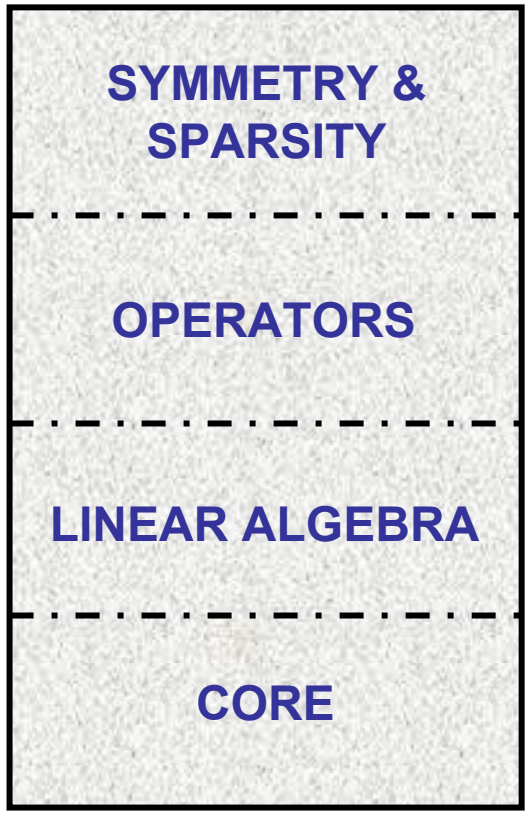
COMPARING AD TOOLS:



OVER - LOADING



SOURCE TO SOURCE



OCEA

Legacy Tools, Sparsity – Processing

Symmetric Storage of Partials

New Tool & Algorithm
Development, Exploit
Symmetry & Sparsity

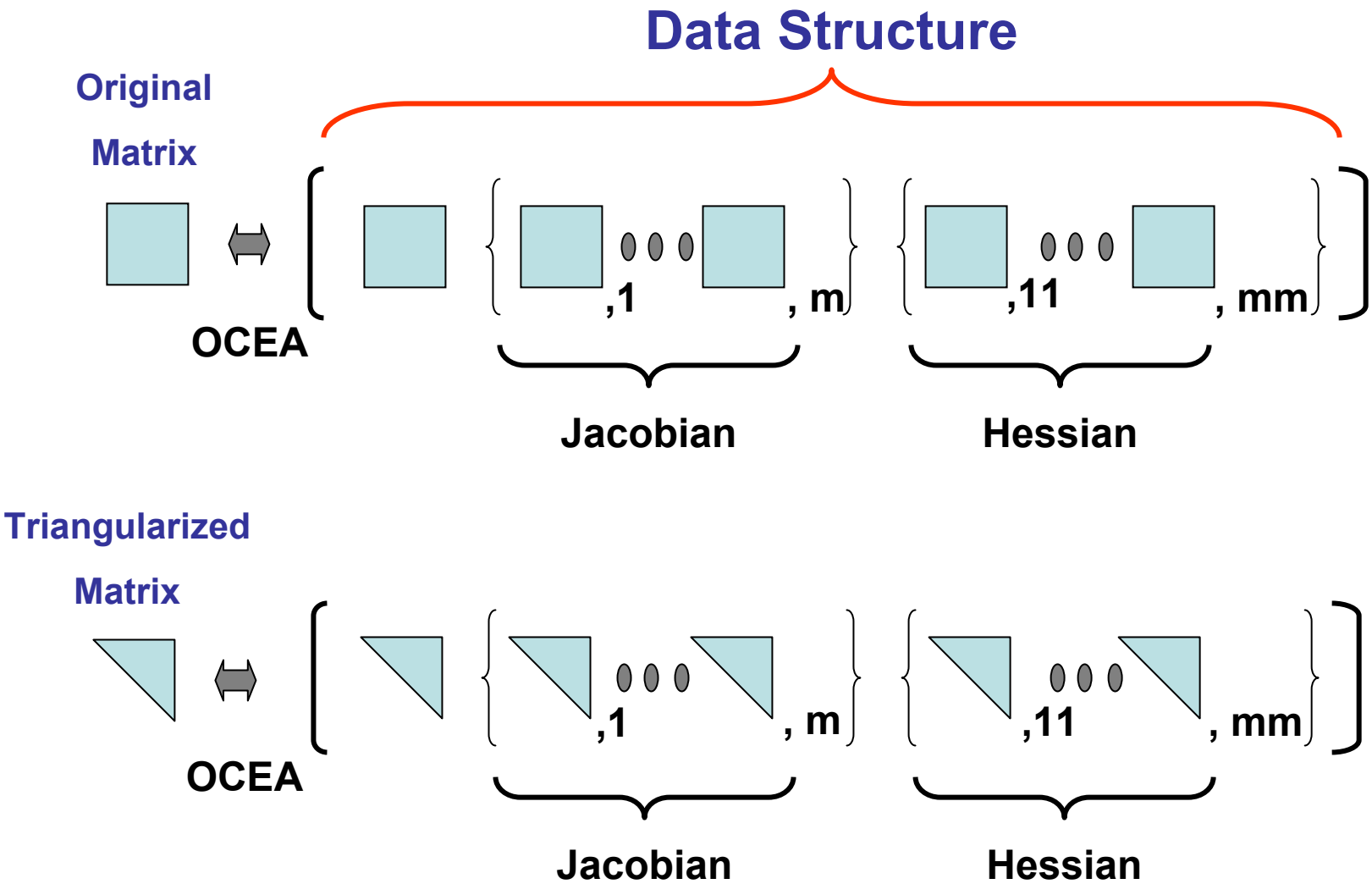
Intelligent AD

OCEA Linear Matrix Eqn

Generalized Linear Model	Generalized Linear Soln
	(E.G., Gaussian Elimin., Cholesky, etc)
$\left[A, A_{,i}, A_{,i,j}\right]\left[x, x_{,i}, x_{,i,j}\right]=\left[b, b_{,i}, b_{,i,j}\right]$	$\left[x, x_{,i}, x_{,i,j}\right]=\left[A, A_{,i}, A_{,i,j}\right]^{-1}\left[b, b_{,i}, b_{,i,j}\right]$
$Ax = b$	$x = A^{-1}b$
$A_{,i}x + Ax_{,i} = b_{,i}$	$x_{,i} = A^{-1}\left(b_{,i} - A_{,i}x\right)$
$A_{,i,j}x + A_{,i}x_{,j} + A_{,j}x_{,i} + Ax_{,i,j} = b_{,i,j}$	$x_{,i,j} = A^{-1}\left(b_{,i,j} - A_{,i,j}x - A_{,i}x_{,j} - A_{,j}x_{,i}\right)$

Calculations and Memory Management Through Object & Sub-Object Operator-Overloading. Linear Algebra *Routines Remain Essentially Unchanged*: Only Data Typing and Use Commands Required

OCEA MATRIX Data Structures During Triangularization

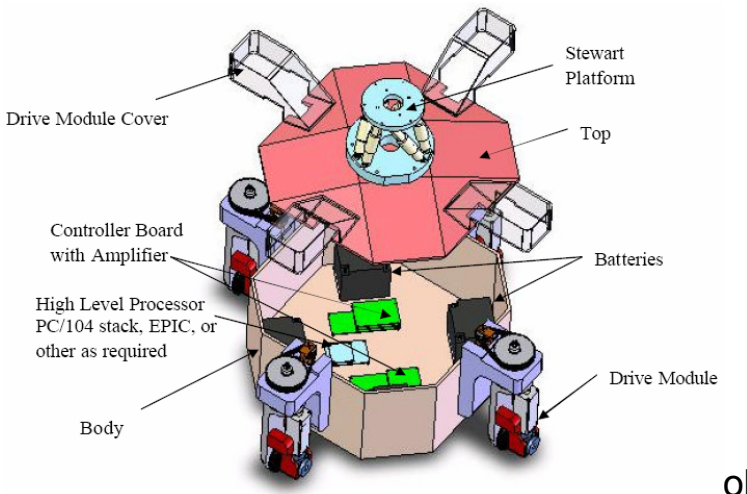


Why Use OCEA–AD For Lagrange's Method?

- **Theoretically Straightforward**
- **Transparent Algorithm Structure & Flow**
- **Model Development Reduced To**:
  - Building Geometric & Kinematic Models
  - OCEA Automates Mechanical Differentiation Of Kinetic Energy
    - Hidden Tools Generate Partial Derivatives
    - Model Building by Generalized Operator – Overloading (Linear Algebra & Vector Operations)
  - OCEA Enables Highly Tuned *Data Structures* For Application – Specific Algorithms

Mobile Stewart Platform: Robotics

- 3 DOF Mobile Base
- 6 DOF multi-body system
- closed-kinematic-chain
- parallel link robot
- high structural rigidity
- high precision positioning
- high force - to - weight ratio



CASS Technology Notional Concept

Hamilton's Principle

Joseph-Louis Lagrange (1736-1813), *Mécanique Analytique* 1788

- **Variational Equation**

$$\delta \int_{t_1}^{t_2} (T - V)\, dt = 0;\ \delta(t_1) = 0;\ \delta(t_2) = 0$$

- **Lagrange's Equation**

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i\ ; \quad i = 1, \cdots, n$$

EOM

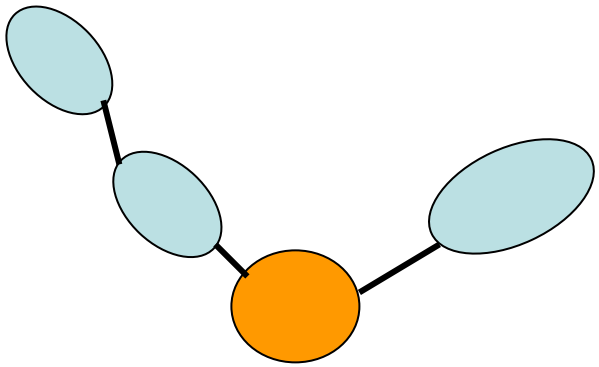
$$Q_i = \sum_{k=1}^{N_B} \left(\vec{F}_k \cdot \frac{\partial \vec{v}_k}{\partial \dot{q}_i} + \vec{\mathrm{T}}_k \cdot \frac{\partial \vec{\omega}_k}{\partial \dot{q}_i} \right)$$

Virtual Work

Subject to: $C\dot{q} = b$

Constraint

Multiple Body Cluster Modeling Issues



Branched Topology

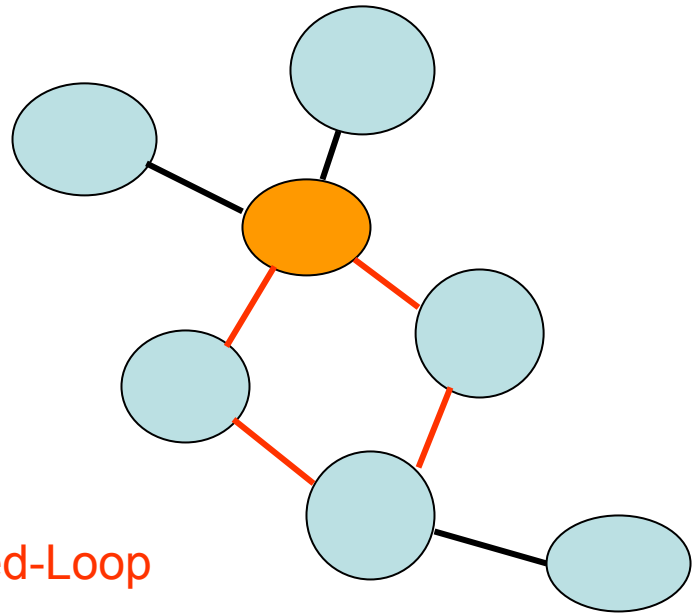
KEY:

Linked Body

Ref. Body

Hinge - Closed-Loop

Hinge - Standard



General Topology

Standard Model:

$(q_1, q_2, \ldots, q_n)$

Independent Coordinates

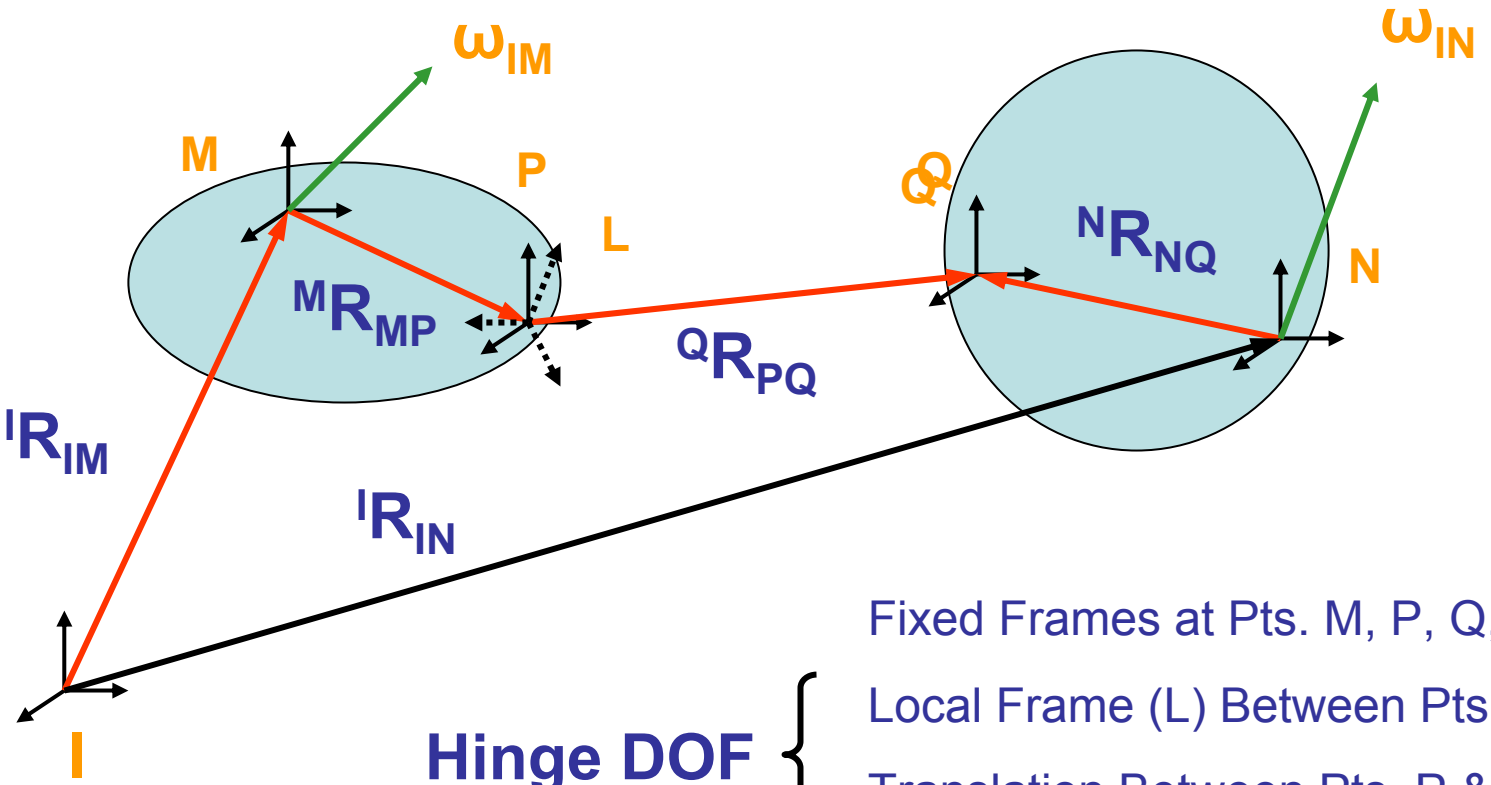
Quasi – Coordinate Model

$(\omega, v, q_7, q_8, \ldots, q_m)$

Independent Cluster Models

Geometric Model

- Generic Hinge Model



Hinge DOF

- Fixed Frames at Pts. M, P, Q, N
- Local Frame (L) Between Pts. P & Q
- Translation Between Pts. P & Q

Kinematic Model (Outboard Body Frame)

- **Position Vector**

$$ {}^{N}R_{IN} = C_{M}^{N}\left({}^{M}R_{IM} + {}^{M}R_{MP}\right) + C_{Q}^{N}\,{}^{Q}R_{PQ} - {}^{N}R_{NQ} $$

- **Velocity Vector**

$$ {}^{N}V_{IN} = C_{M}^{N}\left({}^{M}V_{IM} + {}^{M}\omega_{IM}\,x\,{}^{M}R_{MP}\right) + C_{Q}^{N}\left({}^{Q}V_{PQ} + {}^{Q}\omega_{IQ}\,x\,{}^{Q}R_{PQ}\right) - {}^{N}\omega_{IN}\,x\,{}^{N}R_{NQ} $$

- **Angular Velocity**

$$ {}^{N}\omega_{IN} = C_{M}^{N}\,{}^{M}\omega_{IM} + C_{Q}^{N}\,{}^{Q}\omega_{PQ} $$

Where $C_{M}^{N} = C_{Q}^{N} C_{P}^{Q} C_{M}^{P}$ **Direction Cosine Matrix**

QUASI - COORDINATES

- **Independent Cluster EQM**

$$\frac{d}{dt}\frac{\partial T}{\partial \vec{\omega}} + [\tilde{\omega}]\frac{\partial T}{\partial \vec{\omega}} + \left[\tilde{\mathrm{v}}^{\mathrm{B}}\right]\frac{\partial T}{\partial \vec{\mathrm{v}}} = \Gamma^{-t} Q_{\vec{\theta}}$$

$$\frac{d}{dt}\frac{\partial T}{\partial \vec{\mathrm{v}}} + [\tilde{\omega}]\frac{\partial T}{\partial \vec{\mathrm{v}}} - \frac{\partial T}{\partial \vec{\mathrm{r}}} = \left(\mathrm{C}_{\mathrm{NB}}\right)^{-t} Q_{\vec{\mathrm{r}}^{\mathrm{N}}}$$

Ref. Body

Hinge DOF

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} = Q_i \; ; \quad i = 7, \cdots, n$$

Kinetic Energy & Partials

Automatic Differentiation: q & dq/dt are OCEA Variables

Source-to-Source & Operator Overloading

$$T := \frac{1}{2}\sum_{i=1}^{N_{Bodies}} \begin{bmatrix} \omega \\ V \end{bmatrix}_i^t \begin{bmatrix} J & \widetilde{S} \\ \widetilde{S}^t & mI_{3x3} \end{bmatrix}_i \begin{bmatrix} \omega \\ V \end{bmatrix}_i = \left\{ T, \begin{bmatrix} T_{,q} \\ T_{,\dot{q}} \end{bmatrix}, \begin{bmatrix} T_{,q,q} & T_{,q,\dot{q}} \\ T_{,\dot{q},q} & T_{,\dot{q},\dot{q}} \end{bmatrix} \right\}$$

Kinetic Energy Operator Algebra: Non-Zero Terms, **Boosting Efficiency**.

$$T = \frac{1}{2}\sum_{i=1}^{N_{Bodies}} \begin{bmatrix} \omega \\ V \end{bmatrix}_i^t \begin{bmatrix} J & \widetilde{S} \\ \widetilde{S}^t & mI_{3x3} \end{bmatrix}_i \begin{bmatrix} \omega \\ V \end{bmatrix}_i := \left\{ T, \left\{ \begin{bmatrix} T_{,q} \\ T_{,\dot{q}} \end{bmatrix}_{@t=t_0}, T_{,q}\big|_{@t>t_0} \right\}, T_{,\dot{q},\dot{q}} \right\}$$

Intelligent AD: OCEA

Hidden Sparse Calculations

Second Order Sensitivity

- Fourth Order OCEA Model
  - Mass Matrix has Two Partials

$$M := \left\{ M, \quad \nabla M, \quad \nabla^2 M \right\}$$

  - Extended OCEA State **(Initial Condition & Parameters)**

$$\xi = \left\{ x_0, \quad p \right\}$$

  - Acceleration Solution

$$\dot{x} = M^{-1} f(x,t)$$

OCEA Variables: x, f, M

Second Order Sensitivity Cont.

- **Define Modified Integration State Variable**

$$x(t) = x_0 + \int_{t_0}^{t} f(x(p), p, \tau) d\tau \qquad f := \left\{ f, \begin{bmatrix} \nabla_x f \\ \nabla_p f \end{bmatrix}, \begin{bmatrix} \nabla_{xx} f & \nabla_{xp} f \\ \nabla_{px} f & \nabla_{pp} f \end{bmatrix} \right\}$$

- **Define Modified State Vector**

$$Y := \left(x, \begin{bmatrix} \Phi_x \\ \Phi_p \end{bmatrix}, \begin{bmatrix} \Phi_{xx} & \Phi_{xp} \\ \Phi_{px} & \Phi_{pp} \end{bmatrix} \right); \quad Y_0 = \left(x, \begin{bmatrix} I \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right)$$

$$\dot{Y} := \left(f, \begin{bmatrix} \nabla_x f \cdot \Phi_x \\ \nabla_x f \cdot \Phi_p + \nabla_p f \end{bmatrix}, \begin{bmatrix} \dot{\Phi}_{xx} & \dot{\Phi}_{xp} \\ \dot{\Phi}_{px} & \dot{\Phi}_{pp} \end{bmatrix} \right)$$

$$\dot{\Phi}_{xx} = \nabla_{xx} f \cdot \Phi_x \cdot \Phi_x + \nabla_x f \cdot \Phi_{xx}$$

$$\dot{\Phi}_{xp} = \nabla_{xx} f \cdot \Phi_x \cdot \Phi_p + \nabla_{xp} f \cdot \Phi_x + \nabla_x f \cdot \Phi_{xp}$$

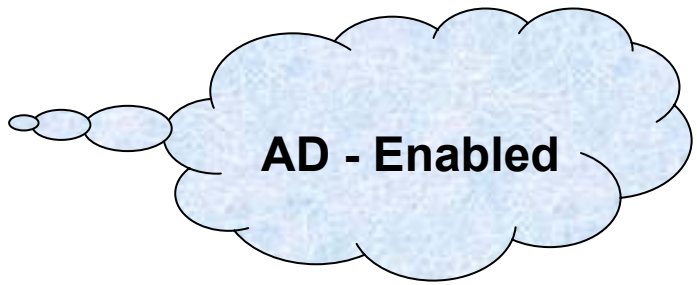
$$\dot{\Phi}_{pp} = \nabla_{xx} f \cdot \Phi_p \cdot \Phi_p + \nabla_{xp} f \cdot \Phi_p + \nabla_x f \cdot \Phi_{pp} + \nabla_{px} f \cdot \Phi_x + \nabla_{pp} f$$

Generalized State Transition Matrices

- **Purpose**: Develop High-Order Taylor Series Relating Initial Conditions and Parameter Sensitivities to Future Times

$$x(t) = \Phi_1(t,t_0)\,x_0 + \frac{1}{2!}\Phi_2(t,t_0)\,x_0x_0 + \cdots + \frac{1}{m!}\Phi_m(t,t_0)\,\underbrace{x_0x_0x_0\cdots x_0}_{m}$$

where $$\Phi_n(t,t_0) = \frac{\partial^n x(t)}{\underbrace{\partial x_0 \partial x_0 \cdots \partial x_0}_{n}}$$



Application: **Model and Error Propagation, Statistical Modeling, Monte-Carlo Simulations, Estimation Theory, Engineering Design & Optimization**

Generalized Newton Method

OCEA – Automatic Differentiation

- Standard Problem Formulation

$$\nabla f\,\delta = f \quad \Rightarrow \quad x_{k+1} = x_k - (\nabla f)^{-1} f = x_k - N$$

- Generalized Problem

$$\nabla f\,\delta = f$$

$$f := \{ f, \quad \nabla f, \quad \cdots \quad \nabla^m f \}$$

$$\nabla f := \{ \nabla f, \quad \nabla^2 f, \quad \cdots \quad \nabla^{m+1} f \}$$

$$\delta := \{ N, \quad \underbrace{\nabla N, \quad \cdots \quad \nabla^m N}_{\text{Newton Method Gradients}} \}$$

Newton Gradients can be Rigorously Proved Using OCEA Algebra

Generalized Newton Method
Analytic Continuation

- Invent an Artificial Independent Variable & Define *Embedding Operator* (s = {1,0}, N = Newton Correction)

$$H = N - s\,N\big|_{s=0} = 0 \Rightarrow N = N\left(x(s)\right)$$

- Compute Partial Derivatives w.r.t "s"

$$H_{,s} = \nabla N\, x_{,s} - N\big|_{s=0} = 0$$

$$\Rightarrow x_{,s} = \left(\nabla N\right)^{-1} N\big|_{s=0}$$

$$H_{,ss} = \nabla^2 N\, x_{,s}^2 + \nabla N\, x_{,ss} = 0$$

$$\Rightarrow x_{,ss} = -\left(\nabla N\right)^{-1} \nabla^2 N\, x_{,s}^2$$

$$H_{,sss} = \nabla^3 N\, x_{,s}^3 + 3\nabla^2 N\, x_{,s} x_{,ss} + \nabla N\, x_{,sss} = 0$$

$$\Rightarrow x_{,sss} = -\left(\nabla N\right)^{-1}\left(\nabla^3 N\, x_{,s}^3 + 3\nabla^2 N\, x_{,s} x_{,ss}\right)$$

Easily Extended to Higher Order

Generalized Newton Method

Analytic Continuation Cont.

- Analytically Continue Initial Guess

$$x = x_0 + x_{,s}(s_f - s_0) + \frac{x_{,ss}}{2!}(s_f - s_0)^2 + \frac{x_{,sss}}{3!}(s_f - s_0)^3 + \cdots$$

- Observe that $s_0 = 1$ & $s_f = 0$, Yielding

$$x = x_0 - x_{,s} + \frac{x_{,ss}}{2!} - \frac{x_{,sss}}{3!} + \cdots$$

Elegantly Simple High – Level Newton Method

- Each term in the Expansion adds 2 digits of accuracy!!!! Gradients Obtained from:

$$\{ n, \nabla n, \cdots \nabla^m n \} = \{ \nabla f, \nabla^2 f, \cdots \nabla^{m+1} f \}^{-1} \{ f, \nabla f, \cdots \nabla^m f \}$$

Gaussian Elimination

Sample Problem

- 5x1 Nonlinear Vector System

$$f_1 = \exp(u) + x^2 \cos(z) - f_{10}$$
$$f_2 = x\,u\,(y\,z)^2 - f_{20}$$
$$f_3 = x\,y\,z^2 - f_{30}$$
$$f_4 = z^3 / u - f_{40}$$
$$f_5 = z\sqrt{w} - f_{50}$$

Reference Model

$f_{i0} = f_{i0}(x,y,z,u,w) = f_{i0}(1,2,3,4,5)$

$i = 1,2,3,4,5$

- 3ed Order Generalized Algorithm

Iteration: 0 1 2

$$x = 1.067 \Rightarrow 1.0003 \Rightarrow 1.0000000000000000004$$
$$y = 1.867 \Rightarrow 1.9997 \Rightarrow 1.9999999999999999996$$
$$z = 3.150 \Rightarrow 2.9997 \Rightarrow 3.0000000000000000000000000000$$
$$u = 3.800 \Rightarrow 3.9997 \Rightarrow 4.0000000000000000000000000000$$
$$w = 5.500 \Rightarrow 5.0002 \Rightarrow 5.0000000000000000000000000000$$

Better Than Modest Convergence

Conclusions & Recommendations

- **QuadPack95 - Quaternions**
  - Core Library
  - Linear & Quadratic Polynomials
  - Matrix Solutions
    - Linear Equations, Inversion
    - Eigensolutions
- **OCEA – Automatic Differentiation**
  - Core Library, Linear Algebra, Operators
  - Applications
    - Multibody
    - Generalized State Transition Matrices
    - Generalized Optimization

Key Features

Fortran 95/2003

Matlab-Like Ease

High-Performance

Object –Oriented

Operator-Overloaded

Available Soon for Research

Funded Research Any Time