

# Accurate Simulation of Rigid Body Rotation and Extensions to Lie Groups

Sam Buss  
Dept. of Mathematics  
U.C. San Diego

Visible Luncheon  
Gwyddor Cyfrifiador  
University of Swansea

March 3, 2011

- Topics:**
- ▶ Algorithms for simulating rotating rigid bodies.
  - ▶ All algorithms preserve angular momentum.
  - ▶ Algorithms can be made energy preserving.
  - ▶ Generalization to Lie group setting.

**Talk outline:**

1. Rigid body rotations. 1st thru 4th order algorithms.  
Unexpected terms.
2. Generalization to Taylor series methods  
over Lie groups/Lie algebras.
3. Energy preservation based on Poincaré ellipsoid.
4. Numerical simulations and efficiency.

## Part I: The simple rotating, rigid body

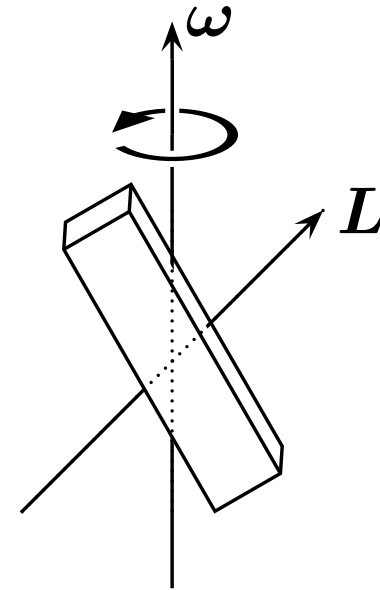
$I$  = Inertia matrix (tensor).

$L$  = Angular momentum.

$\omega$  = Rotation axis & rate,

$L = I\omega$  (Euler's equation)

$\omega = I^{-1}L$



$$\dot{\omega} = I^{-1}(\dot{L} - \omega \times I\omega)$$

$$\ddot{\omega} = \omega \times \dot{\omega} + I^{-1}(\ddot{L} - \dot{\omega} \times L - 2\omega \times \dot{L} + \omega \times (\omega \times L))$$

$$\begin{aligned} \ddot{\omega} = & 2\omega \times \ddot{\omega} - \omega \times (\omega \times \dot{\omega}) + I^{-1}[\ddot{L} - 3\omega \times \ddot{L} - 3\dot{\omega} \times \dot{L} - \ddot{\omega} \times L \\ & + \dot{\omega} \times (\omega \times L) + 2\omega \times (\dot{\omega} \times L) + 3\omega \times (\omega \times \dot{L}) - \omega \times (\omega \times (\omega \times L))] \end{aligned}$$

Wobble:  $\dot{\omega} \neq 0$  even when no applied torque ( $\dot{L} = 0$ ).

$\dot{L} =$  Rate of change of momentum = Applied Torque.

## Framework for simulating rigid body motion

We assume the rigid body has a known angular momentum, and the external torques are completely known. The orientation (and hence the angular velocity) is updated in discrete time steps, at times  $t_0, t_1, t_2, \dots$

**Update Step:** At a given time  $t_i$ , let  $h = \Delta t = t_{i+1} - t_i$ , and assume orientation  $\Omega_i$  at time  $t_i$  is known, and that momentum is known (at all times).

Update step calculates a **net rotation rate vector**,  $\bar{\omega}$ , and sets

$$\Omega_{i+1} = R_{h\bar{\omega}}\Omega_i,$$

where  $R_{\nu}$  performs a rotation around axis  $\nu$  of angle  $\|\nu\|$ .

Nearly every rigid body simulation method fits this framework.

## First-order update method

Use  $\omega = I^{-1}\mathbf{L}$  as the estimate for  $\bar{\omega}$ .

### FIRST-ORDER ALGORITHM:

$$\text{Set } \bar{\omega} := \omega_i = I_i^{-1}\mathbf{L}_i.$$

$$\text{Set } \Omega_{i+1} := R_{h\bar{\omega}}\Omega_i.$$

This first-order method performs poorly. A wobbling, spinning object quickly gains energy and soon ends up spinning on a principal axis.

“Good enough for computer games” (?)

## Second-order update method

Use  $\boldsymbol{\omega}$  and  $\dot{\boldsymbol{\omega}}$  to estimate  $\bar{\boldsymbol{\omega}}$  as  $\bar{\boldsymbol{\omega}} = \boldsymbol{\omega} + \frac{h}{2}\dot{\boldsymbol{\omega}}$ .

### SECOND-ORDER ALGORITHM:

$$\text{Set } \boldsymbol{\omega}_i := I_i^{-1} \mathbf{L}_i.$$

$$\text{Set } \dot{\boldsymbol{\omega}}_i := I_i^{-1} (\dot{\mathbf{L}}_i - \boldsymbol{\omega}_i \times \mathbf{L}_i).$$

$$\text{Set } \bar{\boldsymbol{\omega}} := \boldsymbol{\omega}_i + \frac{h}{2}\dot{\boldsymbol{\omega}}_i.$$

$$\text{Set } \boldsymbol{\Omega}_{i+1} := R_{h\bar{\boldsymbol{\omega}}} \boldsymbol{\Omega}_i.$$

The second-order method performs substantially better. However, a wobbling, spinning object still steadily gains energy and ends up spinning on a principal axis.

## False third-order update method

Try using  $\omega$ ,  $\dot{\omega}$  and  $\ddot{\omega}$  to estimate  $\bar{\omega}$  as  $\bar{\omega} = \omega + \frac{h}{2}\dot{\omega} + \frac{h^2}{3!}\ddot{\omega}$ .

### FALSE THIRD-ORDER ALGORITHM:

$$\text{Set } \omega_i := I_i^{-1} \mathbf{L}_i.$$

$$\text{Set } \dot{\omega}_i := I_i^{-1} (\dot{\mathbf{L}}_i - \omega_i \times \mathbf{L}_i).$$

$$\text{Set } \ddot{\omega}_i := \omega_i \times \dot{\omega}_i + I_i^{-1} (\ddot{\mathbf{L}}_i - \dot{\omega}_i \times \mathbf{L}_i - 2\omega_i \times \dot{\mathbf{L}}_i + \omega_i \times (\omega_i \times \mathbf{L}_i)).$$

$$\text{Set } \bar{\omega} := \omega_i + \frac{1}{2}\dot{\omega}_i h + \frac{1}{6}\ddot{\omega}_i h^2.$$

$$\text{Set } \Omega_{i+1} := R_{h\bar{\omega}} \Omega_i.$$

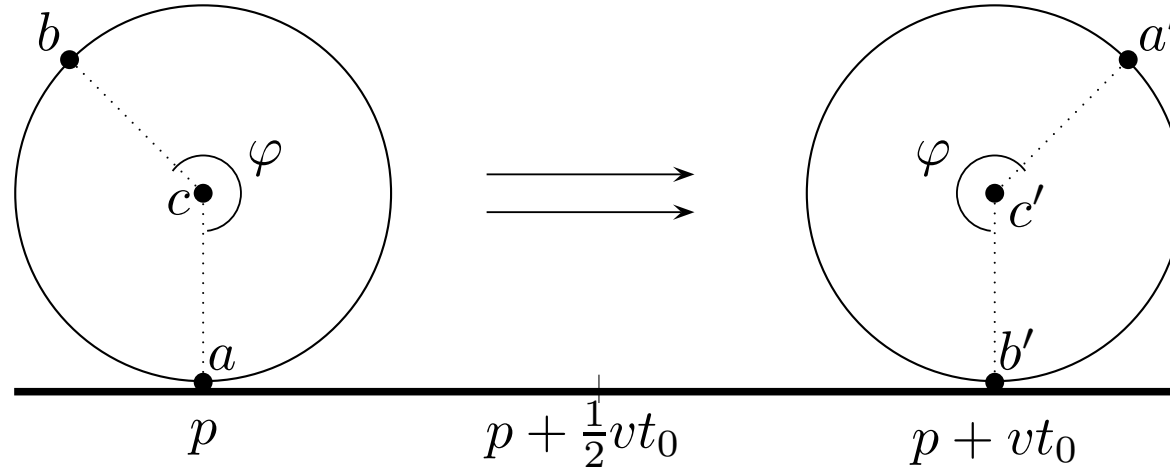
Surprisingly, this however turns out to be slightly worse than the second-order method! In fact, the Taylor series estimate for  $\bar{\omega}$  is not second-order accurate.

## The new third-order term - Motivation

A rolling disk

(or tall skinny cone):

- Total rotation  $\varphi$  in time  $t_0$



$\omega$  - at point  $p$ , pointing into figure, i.e., away from viewer.

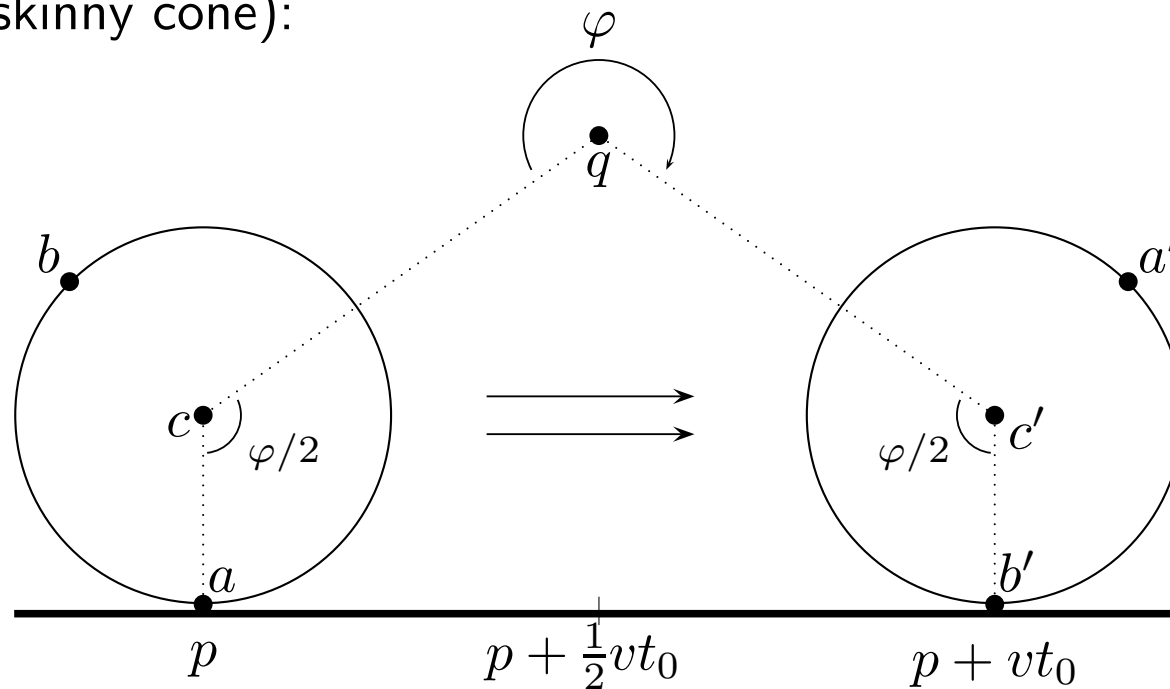
$\dot{\omega}$  - pointing to the right.



## The new third-order term - Motivation

A rolling disk  
(or tall skinny cone):

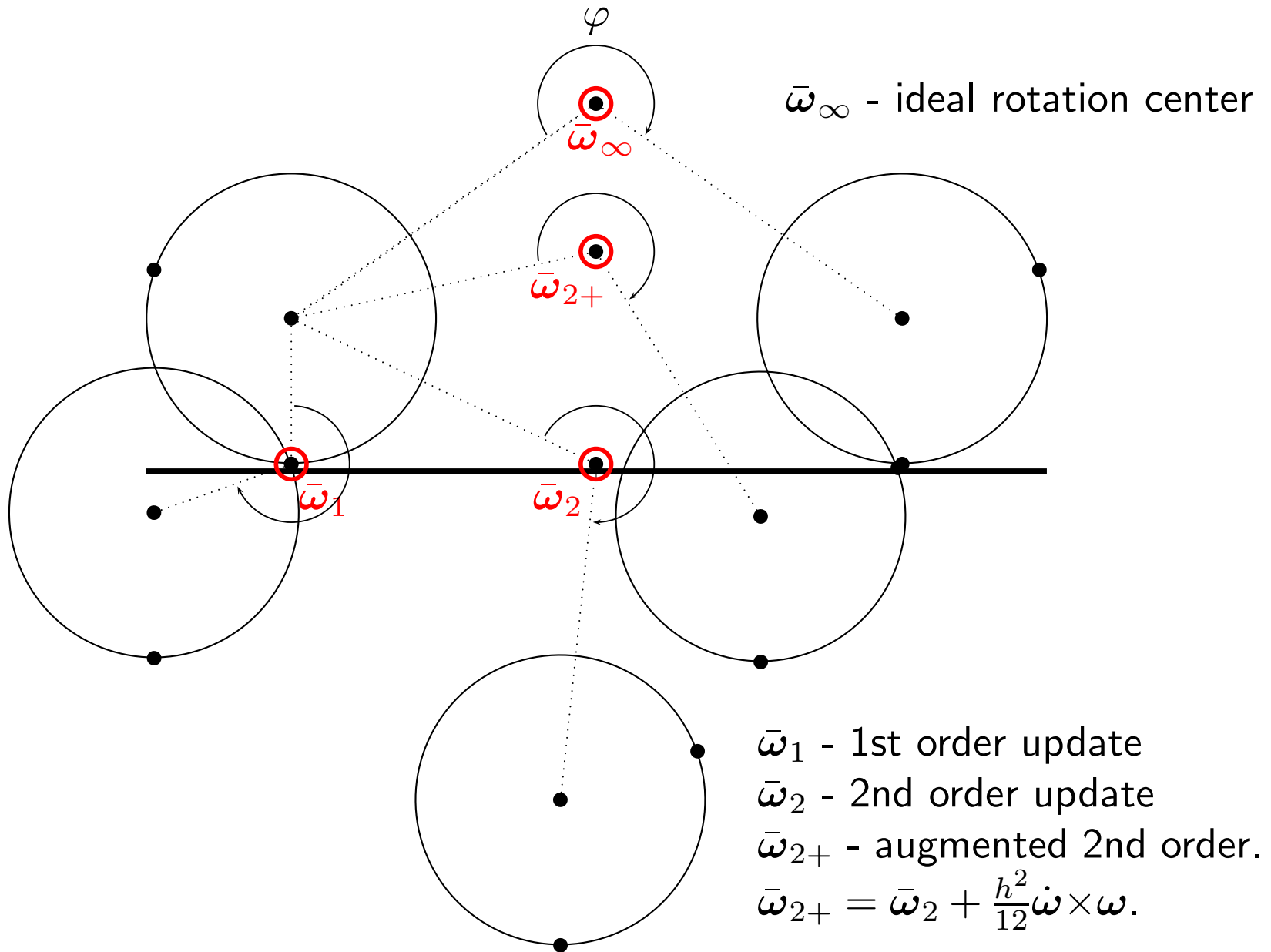
- Total rotation  $\varphi$  in time  $t_0$



$\omega$  - at point  $p$ , pointing into figure, i.e., away from viewer.

$\dot{\omega}$  - pointing to the right.

$\bar{\omega}$  - correct value is the point  $q$ , since a rotation around  $q$  yields the correct net motion.



## Augmented second-order update method

Second-order approximation:  $\bar{\omega}_{2+} = \bar{\omega}_2 + \frac{h^2}{12}\dot{\omega} \times \omega$ .

### AUGMENTED SECOND-ORDER ALGORITHM:

Set  $\omega_i := I_i^{-1} \mathbf{L}_i$ .

Set  $\dot{\omega}_i := I_i^{-1} (\dot{\mathbf{L}}_i - \omega_i \times \mathbf{L}_i)$ .

Set  $\bar{\omega} := \omega_i + \frac{h}{2}\dot{\omega}_i + \frac{h^2}{12}(\dot{\omega}_i \times \omega_i)$ .

Set  $\Omega_{i+1} := R_{h\bar{\omega}} \Omega_i$ .

The augmented second-order method performs substantially better than the second-order method, and has more energy stability, although the energy does drift steadily.

Extra computation cost: only one more cross-product than the second-order method.

## True third-order update method

Now include the new  $\frac{h^2}{12}\dot{\omega} \times \omega$  term in  $\bar{\omega}$ .

### TRUE THIRD-ORDER ALGORITHM:

$$\text{Set } \omega_i := I_i^{-1} \mathbf{L}_i.$$

$$\text{Set } \dot{\omega}_i := I_i^{-1} (\dot{\mathbf{L}}_i - \omega_i \times \mathbf{L}_i).$$

$$\text{Set } \ddot{\omega}_i := \omega_i \times \dot{\omega}_i + I_i^{-1} (\ddot{\mathbf{L}}_i - \dot{\omega}_i \times \mathbf{L}_i - 2\omega_i \times \dot{\mathbf{L}}_i + \omega_i \times (\omega_i \times \mathbf{L}_i)).$$

$$\text{Set } \bar{\omega} := \omega_i + \frac{h}{2}\dot{\omega}_i + \frac{h^2}{6}\ddot{\omega}_i + \frac{h^2}{12}\dot{\omega}_i \times \omega_i.$$

$$\text{Set } \Omega_{i+1} := R_{h\bar{\omega}} \Omega_i.$$

As expected, this is third-order correct, and performs better than the augmented second-order method.

## True fourth-order update method

Additional new term:  $\frac{h^3}{24}\ddot{\omega}_i \times \omega_i$ . (Pattern does not continue)

TRUE FOURTH-ORDER ALGORITHM:

$$\text{Set } \omega_i := I_i^{-1} \mathbf{L}_i.$$

$$\text{Set } \dot{\omega}_i := I_i^{-1} (\dot{\mathbf{L}}_i - \omega_i \times \mathbf{L}_i).$$

$$\text{Set } \ddot{\omega}_i := \omega_i \times \dot{\omega}_i + I_i^{-1} (\ddot{\mathbf{L}}_i - \dot{\omega}_i \times \mathbf{L}_i - 2\omega_i \times \dot{\mathbf{L}}_i + \omega_i \times (\omega_i \times \mathbf{L}_i)).$$

$$\text{Set } \ddot{\omega}_i := (\dots \text{equation on earlier slide \#2} \dots).$$

$$\text{Set } \bar{\omega} := \omega_i + \frac{h}{2}\dot{\omega}_i + \frac{h^2}{6}\ddot{\omega}_i + \frac{h^2}{12}\dot{\omega}_i \times \omega_i + \frac{h^3}{24}\ddot{\omega}_i + \frac{h^3}{24}\ddot{\omega}_i \times \omega_i.$$

$$\text{Set } \Omega_{i+1} := R_{h\bar{\omega}} \Omega_i.$$

Performs better than the true third-order method. Experiments confirm fourth-order accuracy.

## Part II: Generalize to Lie groups/Lie algebras

The extra third- and fourth-order terms can be generalized to the Lie group / Lie algebra setting. This gives Taylor series methods over Lie groups.

**Related to:** Runge-Kutta methods on Lie groups by

Crouch & Grossman '93;

Marthinsen & Owren '98;

Munthe-Kaas '98,'99;

who give higher order corrector terms for Runge-Kutta algorithms.

We write  $[\mathbf{u}, \mathbf{v}]$  for  $\mathbf{u} \times \mathbf{v}$ . Also,  $[\mathbf{u}, \mathbf{v}, \mathbf{w}]$  for  $[\mathbf{u}, [\mathbf{v}, \mathbf{w}]]$ .  $u, v, \dots$  are Lie group elements, and  $[\cdot, \cdot]$  is a Lie group product.

We now use “ $W$ ” instead of “ $\omega$ ”, etc. These are elements of the associated Lie algebra. For  $Z$  is in the Lie algebra,  $\exp(Z)$  is in the Lie group.

$z = \exp(Z)$  is analogous to the rotation operation represented by rotation vector  $Z$ . So,  $Z \sim \bar{\omega}$  and  $\exp(Z) \sim R_{\bar{\omega}}$ .

Suppose  $W(t)$  is a time-varying Lie algebra element. Let  $h > 0$ . We want to find a  $Z = Z(h)$  which is equivalent to applying  $W(t)$  over the time interval 0 to  $h$ :

$$\exp(h \cdot Z) = \lim_{N \rightarrow \infty} \prod_{i=N-1}^0 \exp\left(\frac{h}{N} \cdot W\left(\frac{ih}{N}\right)\right).$$

**Analogy:**  $W(t)$  is time varying instantaneous rotation vector.  $Z$  is  $\bar{\omega}$ .

**Goal:** Find power series for  $Z$  in terms of  $W(0)$ ,  $\dot{W}(0)$ ,  $\ddot{W}(0)$ ,  $\dots$

Let  $Y = Y(h) = h \cdot Z$ . Let  $y(h) = \exp(Y)$ . Now, by defn of  $Z$ ,  $Y$ ,  $y$ ,

$$y'(t) = W(t).$$

Also, taking first derivative of  $y(h) = \exp(Y)$ ,

$$y'(t) = (d \exp)_{Y(t)}(Y'(t)).$$

Power Series expansions: ( $W_0 = W(0)$ ,  $\dot{W}_0 = \dot{W}(0)$ , etc.)

$$(d \exp)_Y = 1 + \frac{1}{2}ad(Y) + \frac{1}{3!}(ad(Y))^2 + \frac{1}{4!}(ad(Y))^3 + \dots$$

(recall  $(ad(A))(B) = [A, B]$ .)

$$W(t) = W_0 + t\dot{W}_0 + \frac{1}{2}t^2\ddot{W}_0 + \frac{1}{3!}t^3\dddot{W}_0 + \dots,$$

$$Y(t) = tY_0 + \frac{1}{2}t^2Y_1 + \frac{1}{3!}t^3Y_2 + \frac{1}{4!}t^4Y_3 + \dots,$$

$$Y'(t) = Y_0 + Y_1t + \frac{1}{2}Y_2t^2 + \frac{1}{3!}Y_3t^3 + \dots.$$

Equating coefficients of powers of  $t$  and solving for  $Y_i$ 's gives:

$$Y_0 = W_0.$$

$$Y_1 = \dot{W}_0.$$

$$Y_2 = \ddot{W}_0 + \frac{1}{2}[\dot{W}_0, W_0]. \quad \text{analogous to the "2+" term.}$$

$$Y_3 = \dddot{W}_0 + [\ddot{W}_0, W_0]. \quad \text{analogous to the "3+" term.}$$



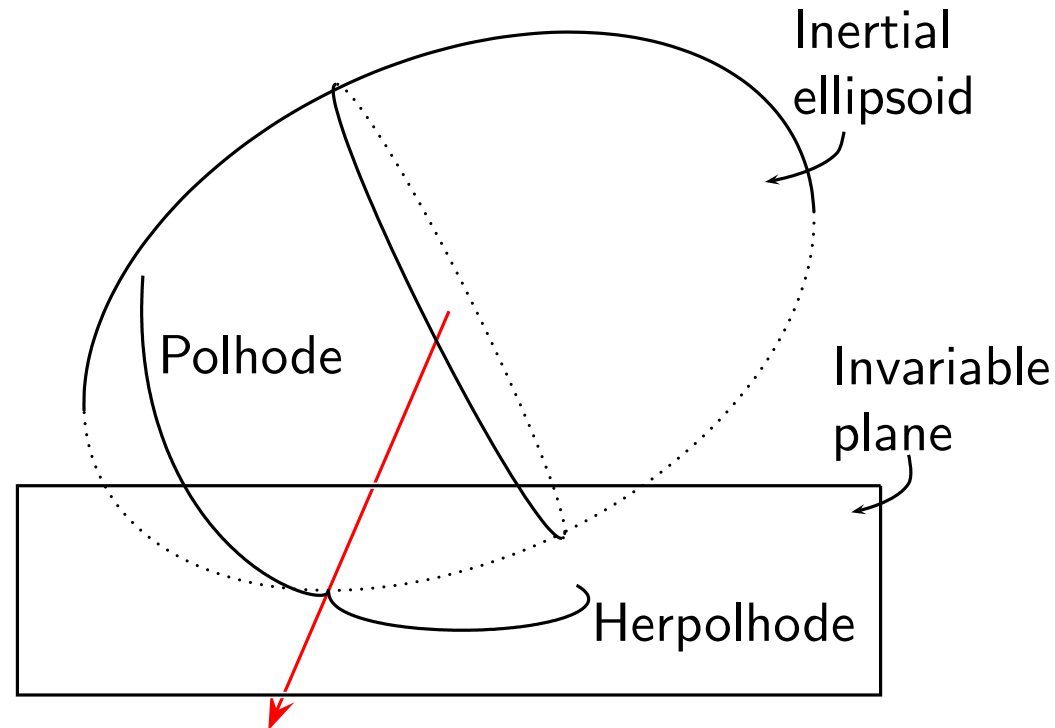
$$Y_4 = \ddot{W}_0 + \frac{3}{2}[\ddot{W}_0, W_0] + [\ddot{W}_0, \dot{W}_0] + \frac{1}{2}[\dot{W}_0, \dot{W}_0, W_0] - \frac{1}{6}[W_0, \ddot{W}_0, W_0] \\ - \frac{1}{6}[W_0, W_0, \dot{W}_0, W_0].$$

$$Y_5 = \ddot{W}_0 + \frac{5}{2}[\ddot{W}_0, \dot{W}_0] + 2[\ddot{W}_0, W_0] + 2[\ddot{W}_0, \dot{W}_0, W_0] + \frac{1}{2}[\dot{W}_0, \ddot{W}_0, W_0] \\ - \frac{1}{2}[W_0, \ddot{W}_0, W_0] - \frac{1}{2}[W_0, W_0, \ddot{W}_0, W_0] - [W_0, \dot{W}_0, \dot{W}_0, W_0].$$

Fifth-order accurate formula for  $Z(h) = h^{-1}Y(h)$ :

$$Z(h) = W_0 + \frac{h}{2}\dot{W}_0 + \frac{h^2}{6}\ddot{W}_0 + \frac{1}{12}h^2[\dot{W}_0, W_0] + \frac{h^3}{24}\ddot{W}_0 + \frac{h^3}{24}[\dot{W}_0, W_0] \\ + \frac{h^4}{120}\ddot{W}_0 + \frac{h^4}{80}[\ddot{W}_0, W_0] + \frac{h^4}{120}[\ddot{W}_0, \dot{W}_0] + \frac{h^4}{240}[\dot{W}_0, \dot{W}_0, W_0] \\ - \frac{h^4}{720}[W_0, \ddot{W}_0, W_0] - \frac{h^4}{720}[W_0, W_0, \dot{W}_0, W_0] \\ + \frac{h^5}{720}\ddot{W}_0 + \frac{h^5}{288}[\ddot{W}_0, \dot{W}_0] + \frac{h^5}{360}[\ddot{W}_0, W_0] + \frac{h^5}{360}[\ddot{W}_0, \dot{W}_0, W_0] \\ + \frac{h^5}{1440}[\dot{W}_0, \ddot{W}_0, W_0] - \frac{h^5}{1440}[W_0, \ddot{W}_0, W_0] - \frac{h^5}{1440}[W_0, W_0, \ddot{W}_0, W_0] \\ - \frac{h^5}{1440}[W_0, W_0, \dot{W}_0, W_0] - \frac{h^5}{720}[W_0, \dot{W}_0, \dot{W}_0, W_0] + O(h^6).$$

## Part III: Back to rigid body: Poinsot inertial ellipsoid



Inertial ellipsoid is attached to the rigid body, and rolls on the plane. Ellipsoid size determined by angular momentum. The plane's height is determined by energy & angular momentum. The polhode is a curve on the ellipsoid: the herpolhode is the curve on the plane. "The polhode rolls without slipping on the herpolhode lying in the invariable plane."

Rotation axis goes through the intersection of the polhode & herpolhode.

## The Poinsot ellipsoid and the polhode

$\rho$  is scaled rotation vector;  $\rho = \omega / \sqrt{I}$ .

$\rho$  depends on orientation, since it is the “lowest” point on the ellipsoid (lowest along the axis of angular momentum).

If no external torques ( $\dot{\mathbf{L}} = 0$ ), then the invariable plane does not vary, and  $\rho$  stays on the same polhode curve.

The polhode is the intersection of two ellipsoids:

1.  $J_{11}\rho_1^2 + J_{22}\rho_2^2 + J_{33}\rho_3^2 = 1$  - the Poinsot ellipsoid.

2.  $J_{11}^2\rho_1^2 + J_{22}^2\rho_2^2 + J_{33}^2\rho_3^2 = \frac{\|\mathbf{L}\|^2}{2E}$ .

where  $J$  is the diagonal inertia matrix.

There is a natural family  $\mathcal{H}$  of hyperboloids which intersects all potential polhodes at right angles.

## Energy Preservation (assuming no external torques)

**Assume no external torques:** If an algorithm exactly conserves angular momentum and energy, then it always produces orientations for  $\rho$  which lies on the polhode. However, any of the earlier algorithms for updating rigid body orientation can give points  $\rho_{i+1}$  which do not lie on the polhode.

Idea of energy preservation: perturb orientation so as to put  $\rho_{i+1}$  back on the polhode. Let  $\rho'$  be close point on polhode and reorient to make it the lowest point.

Algorithm (concept):

Obtain orientation  $\Omega_{i+1}$ , and thence  $\rho_{i+1}$ , by any of the algorithms.

Find hyperboloid from  $\mathcal{H}$  on which  $\rho_{i+1}$  lies.

(Just calculate constant term.)

Find intersection  $\rho'$  of hyperboloid with the two ellipsoids.

(Simple  $3 \times 3$  system of linear equations.)

Reorient ellipsoid by small rotation to make  $\rho'$  the lowest point.

Result is final orientation  $\Omega_{i+1}$ .

## Part IV: Experimental Results

Simulations of rectangular prism of size  $1 \times 4 \times 18$ , for  $\approx 45$  full rotations, no applied torque, updated in fixed time steps. Required number of steps  $N$ , and mean rotation angle  $\theta$  to achieve accuracy of  $\epsilon = 10^{-6}$ .

Algorithm	$N$ (steps)	mean $\theta$ (deg.)
1st order	$> 20480$	failed
1st order EP	16216	1.0
2nd order	14546	1.12
2nd order EP	11505	1.41
False 3rd	15056	1.08
Augmented 2nd (2+)	5573	2.92
Augmented 2nd EP	6616	2.46
3rd order	3661	4.45
3rd order EP	652	25.0
4th order	1304	12.49
4th order EP	688	23.69

“EP” - with energy preservation.

## Required for the Simulation Algorithm

Algorithm	Need to Know	Relative Computational Cost of one update step	
		Without energy preservation	With energy preservation
1 <sup>st</sup> order	$L$	0.30	0.77
2 <sup>nd</sup> order	$\dot{L}$	0.37	0.83
Augmented 2 <sup>nd</sup> order	$\dot{L}$	0.41	0.87
3 <sup>rd</sup> order	$\ddot{L}$	0.54	1.00
4 <sup>th</sup> order	$\dddot{L}$	0.73	1.19

## Comparisons with other methods

Simo-Wong method: similar to 1st order update.

Adams-Bashforth-Moulton Predictor-Corrector: similar to 1st order update.

Traditional 4th order Runge-Kutta: like 2nd order update (slightly better).

McLachlan-Reich-Yoshida Symplectic Algorithms		
Algorithm	$N$ (steps)	mean $\theta$
2nd order 1-2-3	16131	1.01
2nd order 1-2-3 EP	5210	3.12
2nd order 2-3-1	1191	13.68
2nd order 2-3-1 EP	945	17.23
4th order 2-3-1	431	37.85
4th order 2-3-1 EP	196	83.45

Symplectic algorithms very sensitive to axis order.

They are helped considerably by energy conservation, this is odd since energy preservation presumably destroys symplectic property.

## Relative Computational Costs of Single Update Step

Base algorithm	Without energy preservation	With energy preservation
$1^{st}$ order	0.30	0.77
$2^{nd}$ order	0.37	0.83
Augmented $2^{nd}$ order	0.41	0.87
$3^{rd}$ order	0.54	1.00
$4^{th}$ order	0.73	1.19
Symplectic $2^{nd}$ order	0.84	1.20
Symplectic $4^{th}$ order	2.19	2.65



## Part V: Conclusions

### **Recommend algorithms for rigid body update:**

- ▶ If  $\dot{\mathbf{L}}$  is known: augmented 2nd order algorithm.  
(Probably better choice for computer games.)
- ▶ If  $\ddot{\mathbf{L}}$  is known, 3rd order with energy preservation.
- ▶ One symplectic algorithm (4th order 2-3-1 EP) had better performance, but depended on correct axis order, and is presumably no longer symplectic with EP.
- ▶ Our 1st-4th order algorithms all allow arbitrary external torques. (Unlike symplectic algorithms.)

### **Questions:**

- ▶ Do our algorithms work well for rigid multibody systems?
- ▶ Why does energy preservation help symplectic algorithms' long-term accuracy?
- ▶ Why is 3rd order EP unexpectedly accurate?

S. R. Buss, “Accurate and efficient simulations of rigid body rotations”,  
*Journal of Computational Physics* 164 (2000) 377-406.

Web page: <http://math.ucsd.edu/~sbuss/ResearchWeb/accuraterotation>.