Taylor Series Methods for Rigid Body Simulation and Extensions to Lie Groups

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

November 4, 2001

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 Poinsot ellipsoid.
4. Numerical simulations and efficiency.

Part I: The simple rotating, rigid body

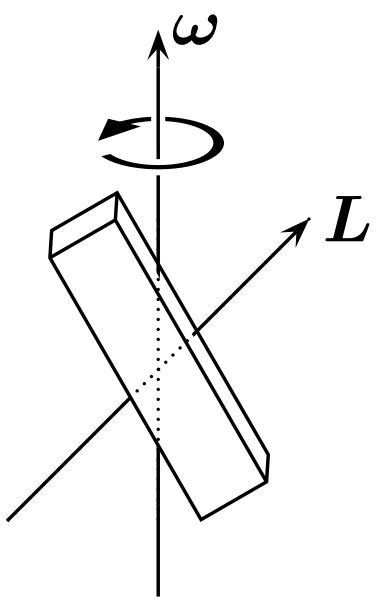
I = Inertia matrix (tensor).

$\boldsymbol{L}$ = Angular momentum.

$\boldsymbol{\omega}$ = Rotation axis & rate,

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

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



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

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

$$\dddot{\boldsymbol{\omega}} = 2\boldsymbol{\omega}\times\ddot{\boldsymbol{\omega}} - \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\dot{\boldsymbol{\omega}}) + I^{-1}[\dddot{\boldsymbol{L}} - 3\boldsymbol{\omega}\times\ddot{\boldsymbol{L}} - 3\dot{\boldsymbol{\omega}}\times\dot{\boldsymbol{L}} - \ddot{\boldsymbol{\omega}}\times\boldsymbol{L} + \dot{\boldsymbol{\omega}}\times(\boldsymbol{\omega}\times\boldsymbol{L}) + 2\boldsymbol{\omega}\times(\dot{\boldsymbol{\omega}}\times\boldsymbol{L}) + 3\boldsymbol{\omega}\times(\boldsymbol{\omega}\times\dot{\boldsymbol{L}}) - \boldsymbol{\omega}\times(\boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{L}))]$$

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

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, $t_2, \ldots$.

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

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

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

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

Nearly every rigid body simulation method fits this framework.

First-order update method

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

First-order algorithm:

Set $\bar{\boldsymbol{\omega}}$:= $\boldsymbol{\omega}_i = I_i^{-1}\boldsymbol{L}_i$.
Set Ω_{i+1} := $R_{h\bar{\boldsymbol{\omega}}}\,\Omega_i$.

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:

Set $\boldsymbol{\omega}_i$:= $I_i^{-1}\boldsymbol{L}_i$.
Set $\dot{\boldsymbol{\omega}}_i$:= $I_i^{-1}(\dot{\boldsymbol{L}}_i - \boldsymbol{\omega}_i \times \boldsymbol{L}_i)$.
Set $\bar{\boldsymbol{\omega}}$:= $\boldsymbol{\omega}_i + \frac{h}{2}\dot{\boldsymbol{\omega}}_i$.
Set Ω_{i+1} := $R_{h\bar{\boldsymbol{\omega}}}\,\Omega_i$.

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

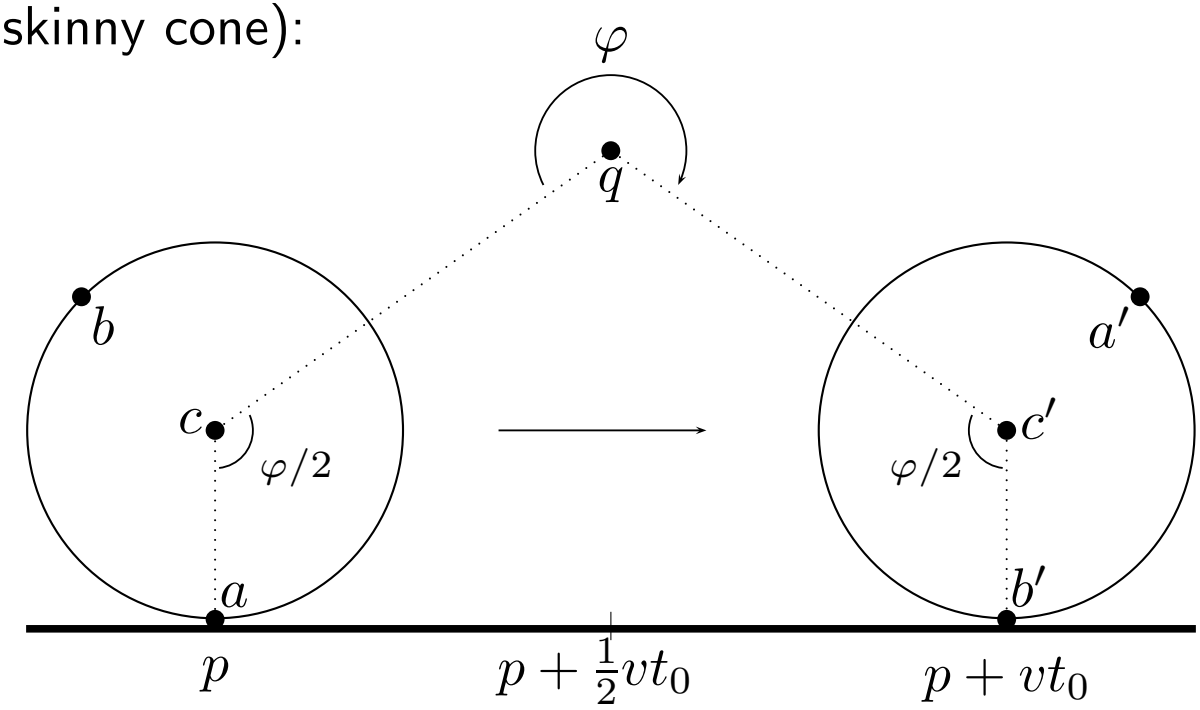
False third-order algorithm:

Set $\boldsymbol{\omega}_i$:= $I_i^{-1}\boldsymbol{L}_i$.
Set $\dot{\boldsymbol{\omega}}_i$:= $I_i^{-1}(\dot{\boldsymbol{L}}_i - \boldsymbol{\omega}_i \times \boldsymbol{L}_i)$.
Set $\ddot{\boldsymbol{\omega}}_i$:= $\boldsymbol{\omega}_i \times \dot{\boldsymbol{\omega}}_i + I_i^{-1}(\ddot{\boldsymbol{L}}_i - \dot{\boldsymbol{\omega}}_i \times \boldsymbol{L}_i - 2\boldsymbol{\omega}_i \times \dot{\boldsymbol{L}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{L}_i))$.
Set $\bar{\boldsymbol{\omega}}$:= $\boldsymbol{\omega}_i + \frac{1}{2}\dot{\boldsymbol{\omega}}_i h + \frac{1}{6}\ddot{\boldsymbol{\omega}} h^2$.
Set Ω_{i+1} := $R_{h\bar{\boldsymbol{\omega}}}\,\Omega_i$.

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

The new third-order term - Motivation

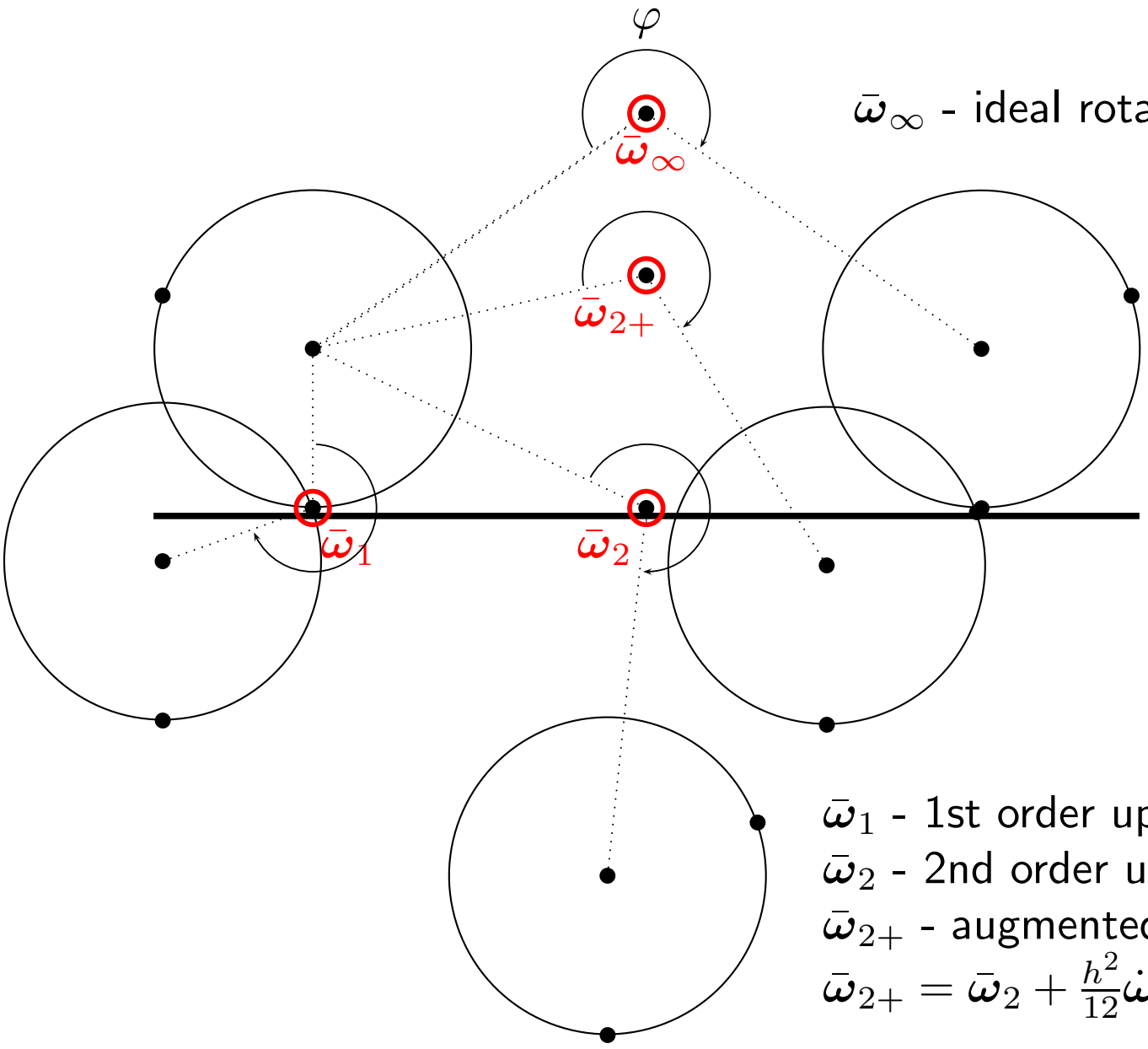
A rolling disk
(or tall skinny cone):



$\boldsymbol{\omega}$ - at point p, pointing up from figure.

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

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



$\bar{\omega}_\infty$ - ideal rotation center

$\bar{\omega}_1$ - 1st order update

$\bar{\omega}_2$ - 2nd order update

$\bar{\omega}_{2+}$ - augmented 2nd order.

$\bar{\omega}_{2+} = \bar{\omega}_2 + \frac{h^2}{12}\dot{\omega}\times\omega$.

Augmented second-order update method

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

AUGMENTED SECOND-ORDER ALGORITHM:

Set $\boldsymbol{\omega}_i$:= $I_i^{-1}\boldsymbol{L}_i$.
Set $\dot{\boldsymbol{\omega}}_i$:= $I_i^{-1}(\dot{\boldsymbol{L}}_i - \boldsymbol{\omega}_i \times \boldsymbol{L}_i)$.
Set $\bar{\boldsymbol{\omega}}$:= $\boldsymbol{\omega}_i + \frac{h}{2}\dot{\boldsymbol{\omega}}_i + \frac{h^2}{12}(\dot{\boldsymbol{\omega}}_i \times \boldsymbol{\omega}_i)$.
Set Ω_{i+1} := $R_{h\bar{\boldsymbol{\omega}}}\,\Omega_i$.

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 $\frac{h^2}{12}\dot{\boldsymbol{\omega}}\times\boldsymbol{\omega}$ term in $\bar{\boldsymbol{\omega}}$.

TRUE THIRD-ORDER ALGORITHM:

$$
\begin{aligned}
&\texttt{Set } \boldsymbol{\omega}_i \texttt{ := } I_i^{-1}\boldsymbol{L}_i\,.\\
&\texttt{Set } \dot{\boldsymbol{\omega}}_i \texttt{ := } I_i^{-1}(\dot{\boldsymbol{L}}_i - \boldsymbol{\omega}_i\times\boldsymbol{L}_i)\,.\\
&\texttt{Set } \ddot{\boldsymbol{\omega}}_i \texttt{ := } \boldsymbol{\omega}_i\times\dot{\boldsymbol{\omega}}_i + I_i^{-1}(\ddot{\boldsymbol{L}}_i - \dot{\boldsymbol{\omega}}_i\times\boldsymbol{L}_i - 2\boldsymbol{\omega}_i\times\dot{\boldsymbol{L}}_i + \boldsymbol{\omega}_i\times(\boldsymbol{\omega}_i\times\boldsymbol{L}_i))\,.\\
&\texttt{Set } \bar{\boldsymbol{\omega}} \texttt{ := } \boldsymbol{\omega}_i + \tfrac{h}{2}\dot{\boldsymbol{\omega}}_i + \tfrac{h^2}{6}\ddot{\boldsymbol{\omega}}_i + \tfrac{h^2}{12}((\dot{\boldsymbol{\omega}}_i + \tfrac{h}{3}\ddot{\boldsymbol{\omega}}_i)\times\boldsymbol{\omega}_i).\\
&\texttt{Set } \Omega_{i+1} \texttt{ := } R_{h\bar{\boldsymbol{\omega}}}\,\Omega_i\,.
\end{aligned}
$$

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

True fourth-order update method

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

TRUE FOURTH-ORDER ALGORITHM:

Set $\boldsymbol{\omega}_i := I_i^{-1}\boldsymbol{L}_i$.
Set $\dot{\boldsymbol{\omega}}_i := I_i^{-1}(\dot{\boldsymbol{L}}_i - \boldsymbol{\omega}_i \times \boldsymbol{L}_i)$.
Set $\ddot{\boldsymbol{\omega}}_i := \boldsymbol{\omega}_i \times \dot{\boldsymbol{\omega}}_i + I_i^{-1}(\ddot{\boldsymbol{L}}_i - \dot{\boldsymbol{\omega}}_i \times \boldsymbol{L}_i - 2\boldsymbol{\omega}_i \times \dot{\boldsymbol{L}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{L}_i))$.
Set $\dddot{\boldsymbol{\omega}}_i :=$ (···*equation on second slide*···).
Set $\bar{\boldsymbol{\omega}} := \boldsymbol{\omega}_i + \frac{h}{2}\dot{\boldsymbol{\omega}}_i + \frac{h^2}{6}\ddot{\boldsymbol{\omega}}_i + \frac{h^2}{12}\dot{\boldsymbol{\omega}}_i \times \boldsymbol{\omega}_i + \frac{h^3}{24}\dddot{\boldsymbol{\omega}}_i + \frac{h^3}{24}\ddot{\boldsymbol{\omega}}_i \times \boldsymbol{\omega}_i$.
Set $\Omega_{i+1} := R_{h\bar{\boldsymbol{\omega}}}\,\Omega_i$.

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 $[\boldsymbol{u}, \boldsymbol{v}]$ for $\boldsymbol{u} \times \boldsymbol{v}$. Also, $[\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}]$ for $[\boldsymbol{u}, [\boldsymbol{v}, \boldsymbol{w}]]$. u, $v, \ldots$ are Lie group elements, and $[\cdot, \cdot]$ is a Lie group product.

We now use "W" instead of "$\boldsymbol{\omega}$", 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.

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:

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

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

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

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: ($W_0 = W(0)$, etc.)

$$
\begin{aligned}
(d\exp)_Y &= 1 + \frac{1}{2}ad(Y) + \frac{1}{3!}(ad(Y))^2 + \frac{1}{4!}(ad(Y))^3 + \cdots \\
&\qquad\qquad \text{(recall } (ad(A))(B) = [A,B].\text{)} \\
W(t) &= W_0 + t\dot{W}_0 + \tfrac{1}{2}t^2\ddot{W}_0 + \tfrac{1}{3!}t^3\dddot{W}_0 + \cdots, \\
Y(t) &= tY_0 + \tfrac{1}{2}t^2Y_1 + \tfrac{1}{3!}t^3Y_2 + \tfrac{1}{4!}t^4Y_3 + \cdots, \\
Y'(t) &= Y_0 + Y_1t + \tfrac{1}{2}Y_2t^2 + \tfrac{1}{3!}Y_3t^3 + \cdots.
\end{aligned}
$$

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

$$
\begin{aligned}
Y_0 &= W_0. \\
Y_1 &= \dot{W}_0. \\
Y_2 &= \ddot{W}_0 + \frac{1}{2}[\dot{W}_0, W_0]. \quad \text{with the "2+" term.} \\
Y_3 &= \dddot{W}_0 + [\ddot{W}_0, W_0].
\end{aligned}
$$

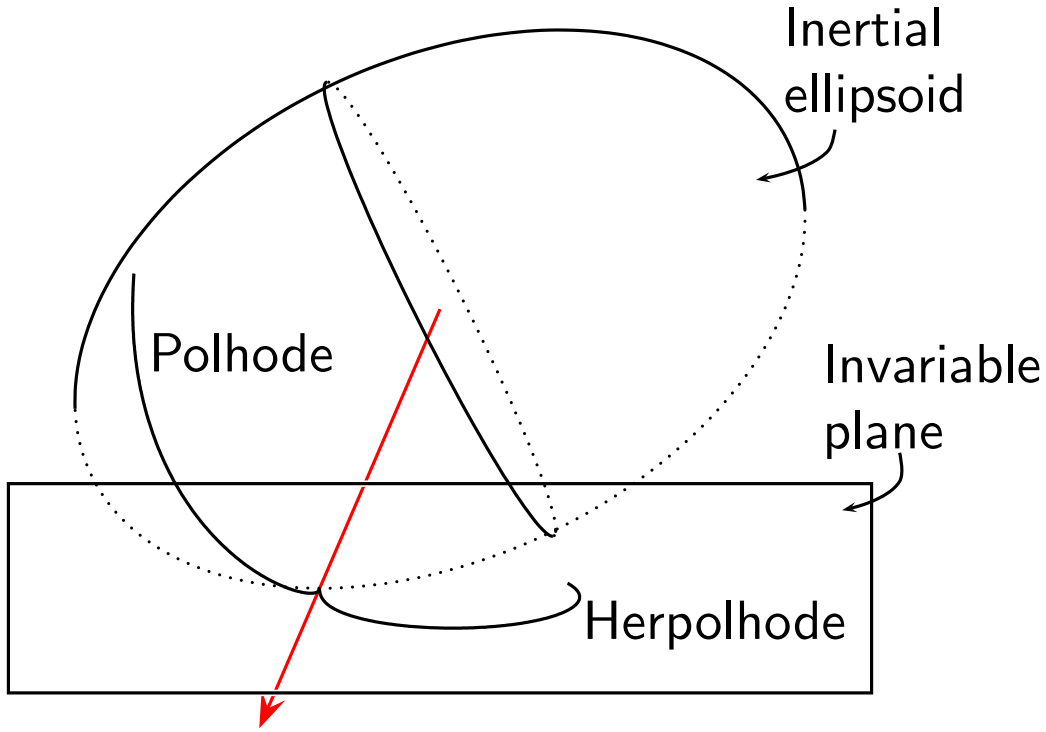
$$
\begin{aligned}
Y_4 &= \dddot{\dot{W}}_0 + \tfrac{3}{2}[\ddot{\dot{W}}_0, W_0] + [\ddot{W}_0, \dot{W}_0] + \tfrac{1}{2}[\dot{W}_0, \dot{W}_0, W_0] - \tfrac{1}{6}[W_0, \ddot{W}_0, W_0] \\
&\quad -\tfrac{1}{6}[W_0, W_0, \dot{W}_0, W_0].
\end{aligned}
$$

$$
\begin{aligned}
Y_5 &= \ddddot{W}_0 + \tfrac{5}{2}[\dddot{W}_0, \dot{W}_0] + 2[\ddddot{W}_0, W_0] + 2[\ddot{W}_0, \dot{W}_0, W_0] + \tfrac{1}{2}[\dot{W}_0, \ddot{W}_0, W_0] \\
&\quad -\tfrac{1}{2}[W_0, \dddot{W}_0, W_0] - \tfrac{1}{2}[W_0, W_0, \ddot{W}_0, W_0] - [W_0, \dot{W}_0, \dot{W}_0, W_0].
\end{aligned}
$$

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

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

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. Polhode is 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 through polhode & plane intersection.

The Poinsot ellipsoid and the polhode

$\boldsymbol{\rho}$ is scaled rotation vector; $\boldsymbol{\rho} = \boldsymbol{\omega}/\sqrt{\mathcal{I}}$.
$\boldsymbol{\rho}$ depends on orientation, since it is the "lowest" point on the ellipsoid (lowest along the axis of angular momentum).

If no external torques ($\dot{\boldsymbol{L}} = 0$), then the invariable plane does not vary, and $\boldsymbol{\rho}$ 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. $J_{11}^2\rho_1^2 + J_{22}^2\rho_2^2 + J_{33}^2\rho_3^2 = \frac{||\boldsymbol{L}||^2}{2E}$.

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 $\boldsymbol{\rho}$ which lies on the polhode. However, any of the earlier algorithms for updating rigid body orientation can give points $\boldsymbol{\rho}_{i+1}$ which do not lie on the polhode.

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

Algorithm (concept):

Obtain orientation Ω_{i+1}, and thence $\boldsymbol{\rho}_{i+1}$, by any of the algorithms.

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

(Just choose constant term.)

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

(Simple 3×3 system of linear equations.)

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

Result is final orientation Ω_{i+1}.

Part IV: Experimental Results

Simulations of rectangular prism of size $1\times4\times18$, updated in fixed time steps. Computed number of steps N, and mean rotation angle θ to achieve accuracy of $\epsilon = 10^{-6}$.

Algorithm	N (steps)	mean θ
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	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.

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 θ
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

Best algorithm for rigid body update:

- If $\dot{\boldsymbol{L}}$ is known: augmented 2nd order algorithm. (Probably better choice for computer games.).
- If $\ddot{\boldsymbol{L}}$ 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.
- 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?