Discrete Variational Symmetrical Top

Brian Beckman, May 2024

Abstract

In *Harmonic Oscillator via Discrete Lagrangian* (https://community.wolfram.com/groups/-/m/t/3161109) and in *Discrete Damped Harmonic Oscillator* (https://community.wolfram.com/groups/-/m/t/3174169), we numerically integrated harmonic oscillators via the discrete Lagrangian, comparing discrete numerical solutions favorably to analytical solutions.

This time, we tackle the classical symmetrical top via the same variational method on a discretized Lagrangian.

rested in these methods at large because they promise far better conservation of momentum and energy and scale to very large systems.

References

1. Marsden and West, 2001, *Discrete Mechanics and Variational Integrators*, http://www.cds.caltech.edu/~marsden/bib/2001/09-MaWe2001/MaWe2001.pdf
2. Matthew West, *Variational Integrators*, Caltech Thesis, May 2004, https://thesis.library.caltech.edu/2492/1/west_thesis.pdf
3. Marin Kobilarov and Jerrold E. Marsden, *Discrete Geometric Optimal Control on Lie Groups*, IEEE Transactions on Robotics, Vol. 27, No. 4, August 2011.
4. https://hepweb.ucsd.edu/ph110b/110b_notes/node36.html

5. https://bingweb.binghamton.edu/~suzuki/Math-Physics/LN-13S_Symmetric_top.pdf

6. William L. Burke, *Applied Differential Geometry*, Cambridge University Press; 1st edition (January 12, 2008)

Free Variables

- Let m be the mass of the top.
- Let the top be an prolate spheroid (shaped like an American football), spinning about its axis of symmetry. Let l, the *polar radius*, be *twice* the length of the long axis (the semi-major axis) and let s, the *equatorial radius*, be the twice the length of the short axis (the semi-minor axis). The inertia matrix in the CG frame with the long axis vertical is, according to Wolfram MathWorld (the cited article does not make it clear that l and s are radii rather than full lengths or diameters, but this supporting article does).

In[5]:=

```
ClearAll[m, a, c, l, s];
```

$$M = \begin{pmatrix} \frac{1}{5} m \left(l^2 + s^2\right) & 0 & 0 \\ 0 & \frac{1}{5} m \left(l^2 + s^2\right) & 0 \\ 0 & 0 & \frac{2}{5} m\, s^2 \end{pmatrix};$$

To reconcile the derivation in this section with the later derivation following Reference 5, we'll need the following numerical solutions for l and s, retaining only the positive solutions:

In[9]:=

```
ClearAll[M5rules];
```

$$\text{M5rules} = \text{N}\left[\text{Solve}\left[\left\{\frac{1}{5} m \left(l^2 + s^2\right) == 2, \frac{2}{5} m\, s^2 == 1\right\}, \{l, s\}\right]\right][\![4]\!]$$

Out[10]=

$$\left\{l \to \frac{2.73861}{\sqrt{m}}, s \to \frac{1.58114}{\sqrt{m}}\right\}$$

- Let l be the height of the center of gravity (CG) of the top from the supporting surface when the top is fully vertical. The height of the CG when the top is inclined by an angle θ from vertical is $l \sin\left(\frac{\pi}{2} - \theta\right) = l \cos\theta$.
- Let g be the acceleration of gravity at the Earth's surface.

Equations of Motion

In[12]:=

```
ClearAll[L, F, V, T, q, ω, ω2, ϕ, θ, ψ, ϕd, θd, ψd];
```

Lagrangian

Though elementary and classical, this problem is not trivial. Let's start from first principles, but check against References 4 and 5.

Generalized Coordinates and Velocities

Our generalized coordinates are roll ϕ, pitch θ, and yaw ψ (mnemonics: ϕ has a longitudinal roll axis, θ has a lateral pitch axis, and ψ reminds us of *y* for *yaw*). The corresponding generalized velocities are $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$, which we'll write as ϕd, θd, ψd because $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ are not valid names for variables. The standard notation $\partial L / \partial \dot{\phi}$ is a cruel pun (e.g., the dedication of Reference 6, "*To all those who, like me, have wondered how in hell you can change $\dot{q}$ without changing q*"). We can't take the derivative of *L* with respect to the time derivative $\dot{\phi} = \phi'[t]$ of a function $\phi[t]$. $\partial L / \partial \dot{\phi}$ really means $\partial L / \partial(\phi\text{d})$, where ϕd is an ordinary parameter variable of the arity-6 Lagrangian function $L[\phi, \theta, \psi, \phi\text{d}, \theta\text{d}, \psi\text{d}]$.

Potential Energy

By inspection.

In[13]:=
```
ClearAll[V];
V[state : {ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] := m l g Cos[θ];
With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
 V[state]]
```

Out[15]=
```
g l m Cos[θ]
```

Manually check units of measure. See dimensions of mass $\text{length}^2 / \text{time}^2$ = energy.

Kinetic Energy

Check the following output cell visually against Reference 4.

In[16]:=

```
(* column vector *)
ω[{ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] := {
    {ϕd Sin[θ] Sin[ψ] + θd Cos[ψ]},
    {ϕd Sin[θ] Cos[ψ] - θd Sin[ψ]},
    {ϕd Cos[θ] + ψd}};
With[{o = ω[{ϕ, θ, ψ, ϕd, θd, ψd}]},
  With[{o12 = (o⟦1, 1⟧^2 + o⟦2, 1⟧^2) // FullSimplify},
   T[state : {ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] :=
    1/2 M⟦1, 1⟧ o12 + 1/2 M⟦3, 3⟧ o⟦3, 1⟧^2]];
With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
 T[state]]
```

Out[18]=

$$\frac{1}{5} m s^2 (\psi d + \phi d \operatorname{Cos}[\theta])^2 + \frac{1}{10} m \left(l^2 + s^2\right) \left(\theta d^2 + \phi d^2 \operatorname{Sin}[\theta]^2\right)$$

Again see dimensions of mass length^2/time^2 = energy.

En passant, we've defined `state` as the sextuple {ϕ, θ, ψ, ϕd, θd, ψd}.

Because $M⟦1, 1⟧ = M⟦2, 2⟧$, it does not matter which we choose, here or anywhere else in this notebook. That's the defining feature of a *symmetrical* top.

Looking forward to the discrete Lagrangian, check this *T* against the simpler expression $\omega^{\mathsf{T}}.M.\omega$

In[19]:=

```
With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
 (T[state] - 1/2 ω[state]ᵀ.M.ω[state]) // FullSimplify]
```

Out[19]=

```
{{0}}
```

Redefine *T* to be this simpler expression, and simultaneously fish out the singleton, scalar element of *T* via `⟦1,1⟧`, then define the Lagrangian:

In[20]:=

```
ClearAll[L, T];
T[state : {ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] := (1/2 ω[state]ᵀ.M.ω[state])⟦1, 1⟧;
L[state : {ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] := T[state] - V[state];
```

Checked

Take a look at it.

In[23]:=
```
With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
  L[state]] // FullSimplify
```

Out[23]=

$$\frac{1}{10} m \left(-10 g l \mathrm{Cos}[\theta] + 2 s^2 (\psi d + \phi d \mathrm{Cos}[\theta])^2 + (l^2 + s^2) (\phi d \mathrm{Cos}[\psi] \mathrm{Sin}[\theta] - \theta d \mathrm{Sin}[\psi])^2 + (l^2 + s^2) (\theta d \mathrm{Cos}[\psi] + \phi d \mathrm{Sin}[\theta] \mathrm{Sin}[\psi])^2\right)$$

Check our L against L hand-copied from Reference 4, remembering that M has dimensions of mass length^2.

In[24]:=

$$\left(\left(\frac{1}{2} M[[1, 1]] \left(\phi d^2 \mathrm{Sin}[\theta]^2 + \theta d^2\right) + \frac{1}{2} M[[3, 3]] (\phi d \mathrm{Cos}[\theta] + \psi d)^2 - m\ l\ g\ \mathrm{Cos}[\theta]\right)\right.$$

```
   - With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
    L[state]]) // FullSimplify
```

Out[24]=
```
0
```

Good to go!

Euler-Lagrange Equations

Writing $\partial L/\partial v$ instead of the pun $\partial L/\partial \dot{q}$, expand out

$$\frac{d}{dt}\frac{\partial L}{\partial v} - \frac{\partial L}{\partial q} = 0 \tag{1}$$

as v ranges over $\{\phi\mathrm{d}, \theta\mathrm{d}, \psi\mathrm{d}\}$ and q ranges over $\{\phi, \theta, \psi\}$. We'll get a column vector (as a flat `List`) of three terms for each of $\frac{d}{dt}\frac{\partial L}{\partial v}$ and $\frac{\partial L}{\partial q}$.

Here are the three expressions for $\frac{\partial L}{\partial q}$ as a column vector.

In[25]:=
```
ClearAll[dLdq];
(dLdq = (With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
       Table[D[L[state], q], {q, {ϕ, θ, ψ}}]] // FullSimplify)) // MatrixForm
```

Out[26]//MatrixForm=

$$\begin{pmatrix} 0 \\ \frac{1}{5} m \left(5 g l - 2 s^2 \phi d \psi d + (l - s) (l + s) \phi d^2 \mathrm{Cos}[\theta]\right) \mathrm{Sin}[\theta] \\ 0 \end{pmatrix}$$

Note that $l > s$ for any prolate spheroid, and both are positive, so $(l - s) > 0$.

See dimensions of mass length^2/time^2 because the angular variables ϕ, θ, and ψ, are dimensionless.

$\frac{\partial L}{\partial \phi}$ and $\frac{\partial L}{\partial \psi}$ are zero. That means that the corresponding terms $\frac{d}{dt}\frac{\partial L}{\partial \dot{\phi}}$ and $\frac{d}{dt}\frac{\partial L}{\partial \dot{\psi}}$ are zero, or that $\frac{\partial L}{\partial \dot{\phi}}$ and $\frac{\partial L}{\partial \dot{\psi}}$ are constant. More about this below.

Here is $\frac{\partial L}{\partial v}$, or $\frac{\partial L}{\partial \dot{q}}$, with homage to the pun:

In[27]:=
```
ClearAll[dLdqDot];
(dLdqDot = (With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
      Table[D[L[state], qd], {qd, {ϕd, θd, ψd}}]] // FullSimplify)) // MatrixForm
```

Out[28]//MatrixForm=

$$\begin{pmatrix} \frac{1}{10} m \left(4 s^2 \psi d \operatorname{Cos}[\theta] + \phi d \left(l^2 + 3 s^2 + \left(-l^2 + s^2\right) \operatorname{Cos}[2\theta]\right)\right) \\ \frac{1}{5} m \left(l^2 + s^2\right) \theta d \\ \frac{2}{5} m s^2 \left(\psi d + \phi d \operatorname{Cos}[\theta]\right) \end{pmatrix}$$

Now see dimensions of mass length^2/time rather than mass length^2/time^2. We'll get the second time dimension back in the denominator when we compute the total time derivatives via d/dt.

Conjugate Momenta

For any Lagrangian, $\frac{\partial L}{\partial \dot{q}}$ are the ***conjugate momenta*** p_q. In our case, the first, p_ϕ, and the third, p_ψ, are ***conserved momenta***, or ***constants of the motion***, via Noether's theorem. In less fancy language, $\frac{\partial L}{\partial \phi}$ and $\frac{\partial L}{\partial \psi}$ are zero, so $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) = \frac{dp_\phi}{dt} = 0$ and $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) = \frac{dp_\psi}{dt} = 0$ by the Euler-Lagrange equations, so p_ϕ and p_ψ are constant.

Manually check these automatically calculated conjugate momenta against Reference 4:

In[29]:=
```
(dLdqDot⟦1⟧ - ((M⟦1, 1⟧ Sin[θ]^2 + M⟦3, 3⟧ Cos[θ]^2) ϕd + M⟦3, 3⟧ ψd Cos[θ])) //
 FullSimplify
```

Out[29]=
```
0
```

In[30]:=
```
(dLdqDot⟦3⟧ - (M⟦3, 3⟧ (ψd + ϕd Cos[θ])))
```

Out[30]=
```
0
```

Define some variables:

In[31]:=
```
ClearAll[pϕ, pψ];
pϕ = dLdqDot⟦1⟧;
pψ = dLdqDot⟦3⟧;
```

Check Reference 4's equations of motion written in conjugate momenta:

In[34]:=
```
(pϕ - pψ Cos[θ])/(M[[1, 1]] Sin[θ]^2) == ϕd // FullSimplify
```
Out[34]=
```
True
```

In[35]:=
```
((pψ/M[[3, 3]] - ((pϕ - pψ Cos[θ]) Cos[θ])/(M[[1, 1]] Sin[θ]^2)) == ψd) // FullSimplify
```
Out[35]=
```
True
```

These are nice, first-order equations, alternatives to the second-order equations we'll find the hard way below. Reference 5 numerically solves these first-order equations. We'll make sure that our numerical solutions of second-order equations matches their numerical solutions of first-order equations.

Save these equations, minus the simplifications that reduce them to useless `True`:

In[36]:=
```
ClearAll[ϕdEqn, ψdEqn];
ϕdEqn = ((pϕ - pψ Cos[θ])/(M[[1, 1]] Sin[θ]^2) == ϕd);
ψdEqn = ((pψ/M[[3, 3]] - ((pϕ - pψ Cos[θ]) Cos[θ])/(M[[1, 1]] Sin[θ]^2)) == ψd);
```

Equations of Motion, the Hard Way

We'll see an easier way when we get to checking Reference 5 below. For now, let's just follow the Euler-Lagrange recipe blindly. The results will be good.

m, *l*, and s are overt constants. Rewrite their total time derivatives to zero (it's interesting in some problems to let them vary with time, but not here)

In[39]:=
```
ClearAll[constRules];
constRules = {Dt[m, t] → 0, Dt[l, t] → 0, Dt[s, t] → 0};
```

$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}}$, total time derivatives of the conjugate momenta, yielding second-order terms:

In[41]:=
```
ClearAll[ddLdqDotdt];
(ddLdqDotdt = (Dt[dLdqDot, t] /. constRules // FullSimplify)) // MatrixForm
```

Out[42]//MatrixForm=
```
( 1/10 m ((l^2 + 3 s^2 + (-l^2 + s^2) Cos[2 θ]) Dt[ϕd, t] + 4 s^2 Cos[θ] Dt[ψd, t] - 4 (s^2 ψd + (-l + s)
                                                 1/5 m (l^2 + s^2) Dt[θd, t]
                                 2/5 m s^2 (Cos[θ] Dt[ϕd, t] + Dt[ψd, t] - ϕd Dt[θ, t] Sin[θ]
```

Finally, the equations of motion, $\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0$:

In[43]:=
```
ClearAll[hardEqns];
(hardEqns = Table[(lhs == 0) // FullSimplify, {lhs, ddLdqDotdt - dLdq}]) //
 MatrixForm
```

Out[44]//MatrixForm=
```
( m ((-l^2 - 3 s^2 + (l - s) (l + s) Cos[2 θ]) Dt[ϕd, t] - 4 s^2 Cos[θ] Dt[ψd, t] + 4 (s^2 ψd + (-l + s
                        m (l^2 + s^2) Dt[θd, t] + m (-5 g l + 2 s^2 ϕd ψd + (-l^2 + s^2) ϕd^2 Cos[
                             m s (Cos[θ] Dt[ϕd, t] + Dt[ψd, t] - ϕd Dt[θ, t] Sin[θ]
```

Here are some rewrites to set up the equations of motion for numerical solution via `NDSolve`:

In[45]:=
```
ClearAll[fTimeRules];
fTimeRules = {
   Dt[ϕd, t] → ϕ''[t], Dt[θd, t] → θ''[t], Dt[ψd, t] → ψ''[t],
   ϕd → ϕ'[t], θd → θ'[t], ψd → ψ'[t],
   ϕ → ϕ[t], θ → θ[t], ψ → ψ[t]};
```

It’s not easy to interpret, or even believe, these equations straight-up, but the numerical solution delivers the goods.

In[47]:=
```
(hardEqns /. fTimeRules) // MatrixForm
```

Out[47]//MatrixForm=
```
( m (4 Sin[θ[t]] θ'[t] ((-l + s) (l + s) Cos[θ[t]] ϕ'[t] + s^2 ψ'[t]) + (-l^2 - 3 s^2 + (l - s) (l +
                     m Sin[θ[t]] (-5 g l + (-l^2 + s^2) Cos[θ[t]] ϕ'[t]^2 + 2 s^2 ϕ'[t] ψ'[t
                             m s (-Sin[θ[t]] θ'[t] ϕ'[t] + Cos[θ[t]] ϕ''[t] + ψ'
```

Numerical Solution

Pick some numerical values for the free variables, values that will make it easier later to reconcile with Reference 5. For now, the “4” denotes rules to help with Reference 4.

In[48]:=
```
ClearAll[numRules4];
numRules4 = (M5rules /. (m → 1)) ~Join~ {g → 9.81, m → 1};
(hardEqns /. fTimeRules /. numRules4) // MatrixForm
```

Out[50]//MatrixForm=
```
( 4 Sin[θ[t]] θ′[t] (-5. Cos[θ[t]] ϕ′[t] + 2.5 ψ′[t]) + (-15. + 5. Cos[2 θ[t]]) ϕ″[t] - 10. (
                    Sin[θ[t]] (-134.329 - 5. Cos[θ[t]] ϕ′[t]^2 + 5. ϕ′[t] ψ′[t]) + 10. θ″[t] =
                       1.58114 (-Sin[θ[t]] θ′[t] ϕ′[t] + Cos[θ[t]] ϕ″[t] + ψ″[t]) == 0
```

Change tN if you want a longer integration.

In[51]:=
```
ClearAll[tN];
tN = 8;
```

Mathematica has no trouble, numerically, with these equations as they stand, algebraic wrestling not needed.

In[53]:=
```
ClearAll[nsoln];
nsoln = NDSolve[(hardEqns /. fTimeRules /. numRules4) ~Join~ {
    ϕ[0] == 0.1, ϕ'[0] == 10.0,
    θ[0] == 0.4, θ'[0] == 0.1,
    ψ[0] == 0.1, ψ'[0] == 0.1},
  {ϕ[t], θ[t], ψ[t]},
  {t, 0, tN}]
```

Out[54]=
```
{{ϕ[t] → InterpolatingFunction[ Domain: {{0., 8.}} Output: scalar ][t],
  θ[t] → InterpolatingFunction[ Domain: {{0., 8.}} Output: scalar ][t],
  ψ[t] → InterpolatingFunction[ Domain: {{0., 8.}} Output: scalar ][t]}}
```

Conservation of Energy

Again, the “4” denotes that we’re currently checking against Reference 4.

In[55]:=
```
ClearAll[E4];
E4[state : {ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] := T[state] + V[state];
```

In[57]:=
```
With[{state = {ϕ[t], θ[t], ψ[t], ϕ'[t], θ'[t], ψ'[t]}},
   E4[state]] /. numRules4 // FullSimplify
```

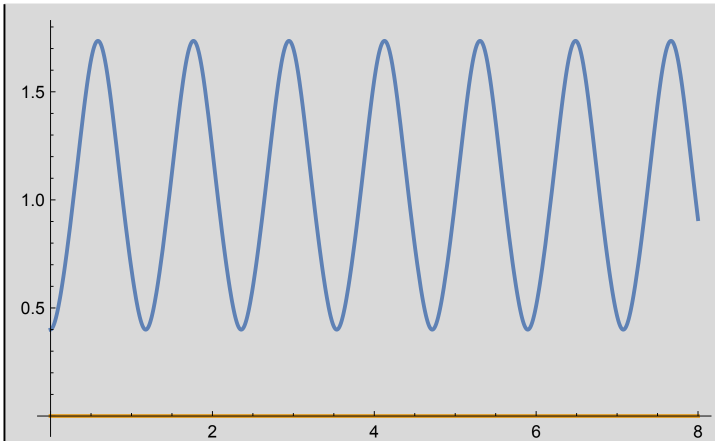
Out[57]=
$$26.8658\,\mathrm{Cos}[\theta[t]] + 1.\,\theta'[t]^2 + (0.75 - 0.25\,\mathrm{Cos}[2\,\theta[t]])\,\phi'[t]^2 + 1.\,\mathrm{Cos}[\theta[t]]\,\phi'[t]\,\psi'[t] + 0.5\,\psi'[t]^2$$

Some Funny Business

It's not immediately straightforward to compute derivatives of the **InterpolatingFunctions** produced by **NDSolve**. The reason is that the solutions are not functions, but expressions depending on the hard-coded parameter *t*, but Mathematica's **Derivative** function requires functions, and not expression. We must convert the expressions to functions and, furthermore, to avoid any use of the **Symbol** *t*, even if apparently safely encapsulated in **Modules** or as dummy variables in **Plot** commands. To wit, the following does not work:

In[58]:=
```
Module[{fθ},
 fθ[u_] := ((θ[t] /. nsoln) /. {t → u});
 Plot[{fθ[t], fθ'[t]}, {t, 0, tN}]]
```

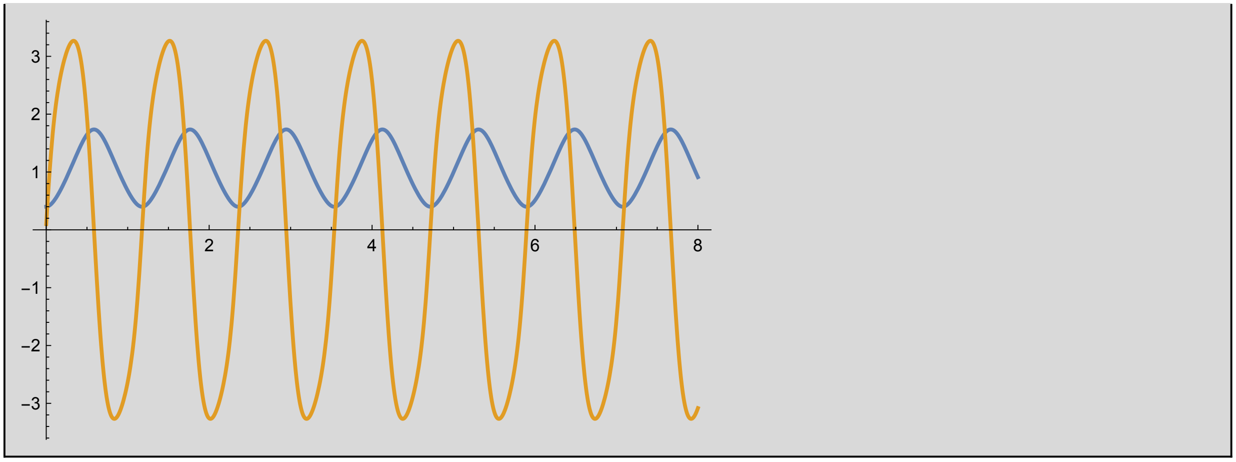
Out[58]=



The following DOES work, so long as the dummy variable in the **Plot** command is anything *except* **t**:

In[59]:=
```
Module[{fθ},
 fθ[u_] := ((θ[t] /. nsoln⟦1⟧) /. {t → u});
 Plot[{fθ[u], fθ'[u]}, {u, 0, tN}]]
```

Out[59]=



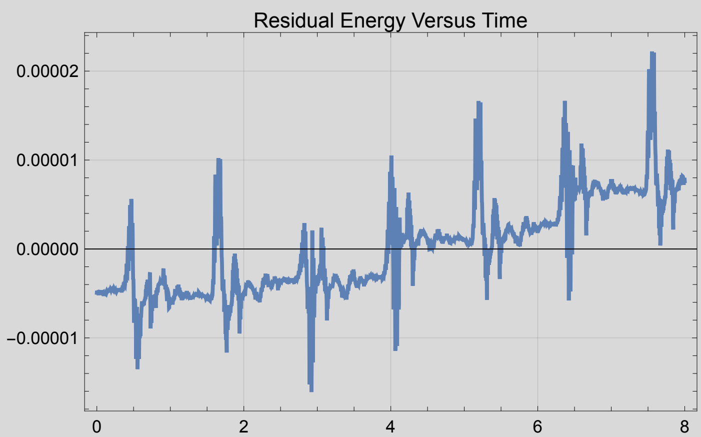
We discovered these limitations by trial-and-error, but we’re now equipped to compute energy from our solution `nsoln`.

Checked

Energy is conserved to five figures. It is, however, trending upwards, certainly due to numerical effects, so we might gain a significant amount for a longer integration. Let’s see below whether the discrete method does better.

In[60]:=
```
Module[{fϕ, fθ, fψ, meanE4},
 fϕ[u_] := ((ϕ[t] /. nsoln⟦1⟧) /. {t → u});
 fθ[u_] := ((θ[t] /. nsoln⟦1⟧) /. {t → u});
 fψ[u_] := ((ψ[t] /. nsoln⟦1⟧) /. {t → u});
 meanE4 =
  Mean[Table[(E4[{fϕ[u], fθ[u], fψ[u], fϕ'[u], fθ'[u], fψ'[u]}] /. numRules4),
    {u, 0, tN, (tN - 0) / 100}]];
 Plot[(E4[{fϕ[u], fθ[u], fψ[u], fϕ'[u], fθ'[u], fψ'[u]}] /. numRules4) - meanE4,
  {u, 0, tN}, PlotLabel → "Residual Energy Versus Time",
  Frame → Automatic, GridLines → Automatic]]
```

Out[60]=



Conservation of Momenta

Likewise, we should check conservation of our momenta, azimuthal $p\phi$ and nodal $p\psi$.

In[61]:=
```
{pϕ, pψ} /. numRules4
```

Out[61]=

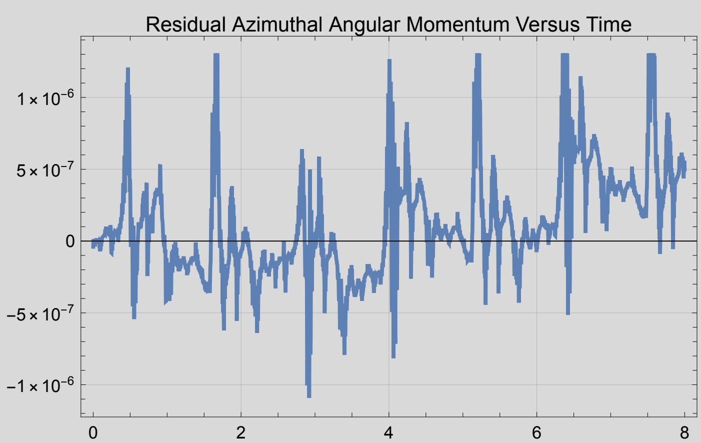
$$\left\{\frac{1}{10}\,(10.\,\psi d\,\mathrm{Cos}[\theta] + \phi d\,(15. - 5.\,\mathrm{Cos}[2\,\theta])),\ 1.\,(\psi d + \phi d\,\mathrm{Cos}[\theta])\right\}$$

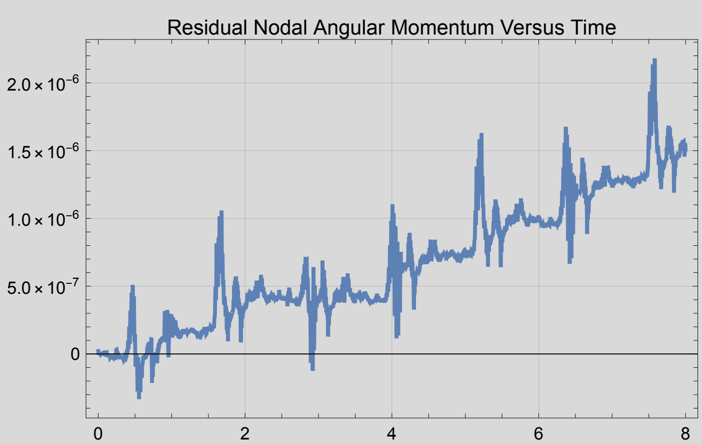
We see conservation within six figures with upward trends.

In[62]:=

```
Module[{fϕ, fθ, fψ, meanPϕ, meanPψ},
 fϕ[u_] := ((ϕ[t] /. nsoln⟦1⟧) /. {t → u});
 fθ[u_] := ((θ[t] /. nsoln⟦1⟧) /. {t → u});
 fψ[u_] := ((ψ[t] /. nsoln⟦1⟧) /. {t → u});
 meanPϕ = Mean[Table[
    pϕ /. numRules4 /. {θ → fθ[u], ψd → fψ'[u], ϕd → fϕ'[u]}, {u, 0, tN, 100}]];
 meanPψ = Mean[Table[
    pψ /. numRules4 /. {θ → fθ[u], ψd → fψ'[u], ϕd → fϕ'[u]}, {u, 0, tN, 100}]];
 Row[{
   Plot[(pϕ /. numRules4 /. {θ → fθ[u], ψd → fψ'[u], ϕd → fϕ'[u]}) - meanPϕ,
    {u, 0, tN}, ImageSize → Medium,
    PlotLabel → "Residual Azimuthal Angular Momentum Versus Time",
    Frame → Automatic, GridLines → Automatic],
   Plot[(pψ /. numRules4 /. {θ → fθ[u], ψd → fψ'[u], ϕd → fϕ'[u]}) - meanPψ,
    {u, 0, tN}, ImageSize → Medium,
    PlotLabel → "Residual Nodal Angular Momentum Versus Time",
    Frame → Automatic, GridLines → Automatic]}]]
```

Out[62]=





Interactive Exploration

`NDSolve` is so fast that we can put it in a `Manipulate` to play around with initial conditions and integration time.

Some graphics helpers:

In[63]:=
```
ClearAll[jack, eulerRotate];
With[{o = {0, 0, 0},
  e1 = {1, 0, 0}, e2 = {0, 1, 0}, e3 = {0, 0, 1},
  tip = {0, 0, 1}, pr = 1.25},
 eulerRotate[obj_, ϕ_, θ_, ψ_] :=
  With[{r3 = RotationMatrix[ϕ, e3]},
   With[{e1Prime = r3.e1},
    With[{e3DoublePrime =
       RotationMatrix[θ, e1Prime].r3.e3},
     Rotate[
      Rotate[
       Rotate[
        obj,
        ϕ, e3],
       θ, e1Prime],
      ψ, e3DoublePrime]]]];
 jack[opacity_ : 0.1, diameter_ : 0.01] := {
   Opacity[opacity],
   {Red, Arrow[Tube[{o, e1}, diameter]]},
   {Darker[Green], Arrow[Tube[{o, e2}, diameter]]},
   {Blue, Arrow[Tube[{o, e3}, diameter]]},
   {Lighter[Cyan], Arrow[Tube[{o, -e1}, diameter]]},
   {Magenta, Arrow[Tube[{o, -e2}, diameter]]},
   {Yellow, Arrow[Tube[{o, -e3}, diameter]]}}]
```

A rendering function:

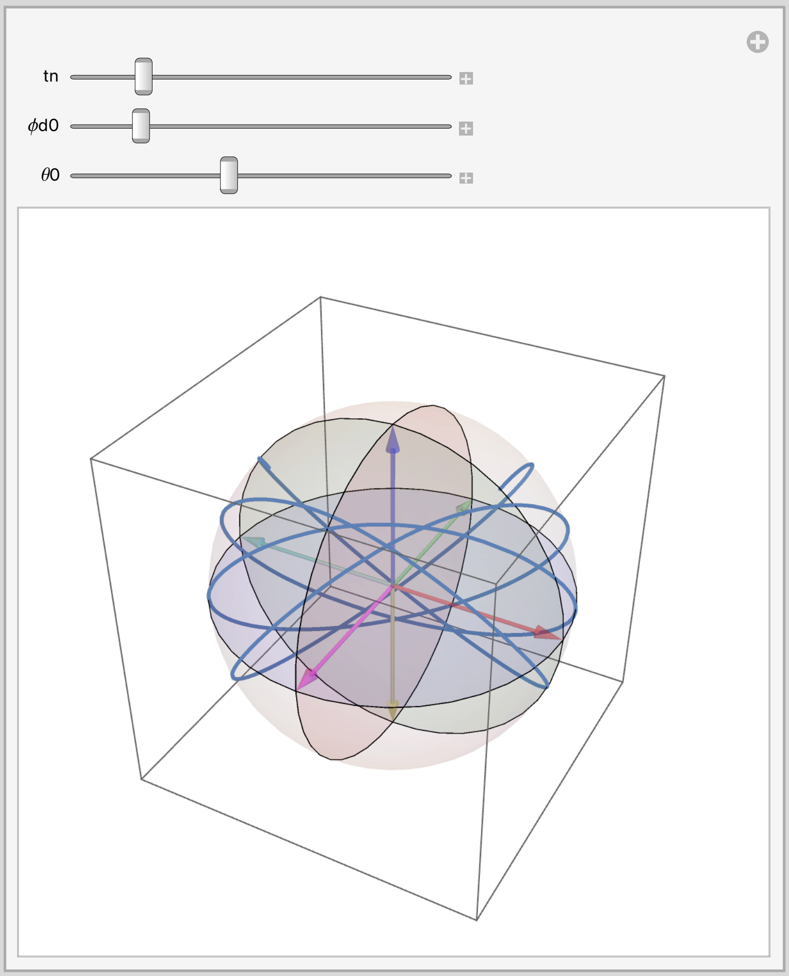
In[65]:=
```
ClearAll[renderSoln];
renderSoln[nsoln_, tn_] :=
  With[{pr = 1.25, e1 = {1, 0, 0}, e2 = {0, 1, 0}, e3 = {0, 0, 1}, squish = 0.0001},
   With[{prl = {{-pr, pr}, {-pr, pr}, {-pr, pr}}},
    Show[
     Graphics3D[{
       jack[.25, 0.0125],
       Opacity[0.1], Sphere[],
       Opacity[0.05],
       Blue, Cylinder[squish {-e3, e3}],
       Red, Cylinder[squish {-e1, e1}],
       Green, Cylinder[squish {-e2, e2}]}],
     ParametricPlot3D[{
        Sin[θ[t]] Cos[ϕ[t]],
        Sin[θ[t]] Sin[ϕ[t]],
        Cos[θ[t]]} /. nsoln,
      {t, 0, tn}, PlotRange → prl, AspectRatio → {1, 1, 1}]]]];
```

A playground:

In[67]:=

```
With[{initials = {θ'[0] == 0, ϕ[0] == 0 °, ψ[0] == 0 °, ψ'[0] == 0},
  eqns = (hardEqns /. fTimeRules /. numRules4)},
 Manipulate[
  Module[{nsoln = NDSolve[
      eqns~Join~initials~Join~{
        ϕ'[0] == ϕd0, θ[0] == θ0},
      {ϕ[t], θ[t], ψ[t]},
      {t, 0, tn}]},
   renderSoln[nsoln, tn]],
  {{tn, 8}, 0.001, 30}, {{ϕd0, 10}, 0.001, 30}, {{θ0, 42 °}, 0.001, π / 2 - 0.001}]]
```

Out[67]=

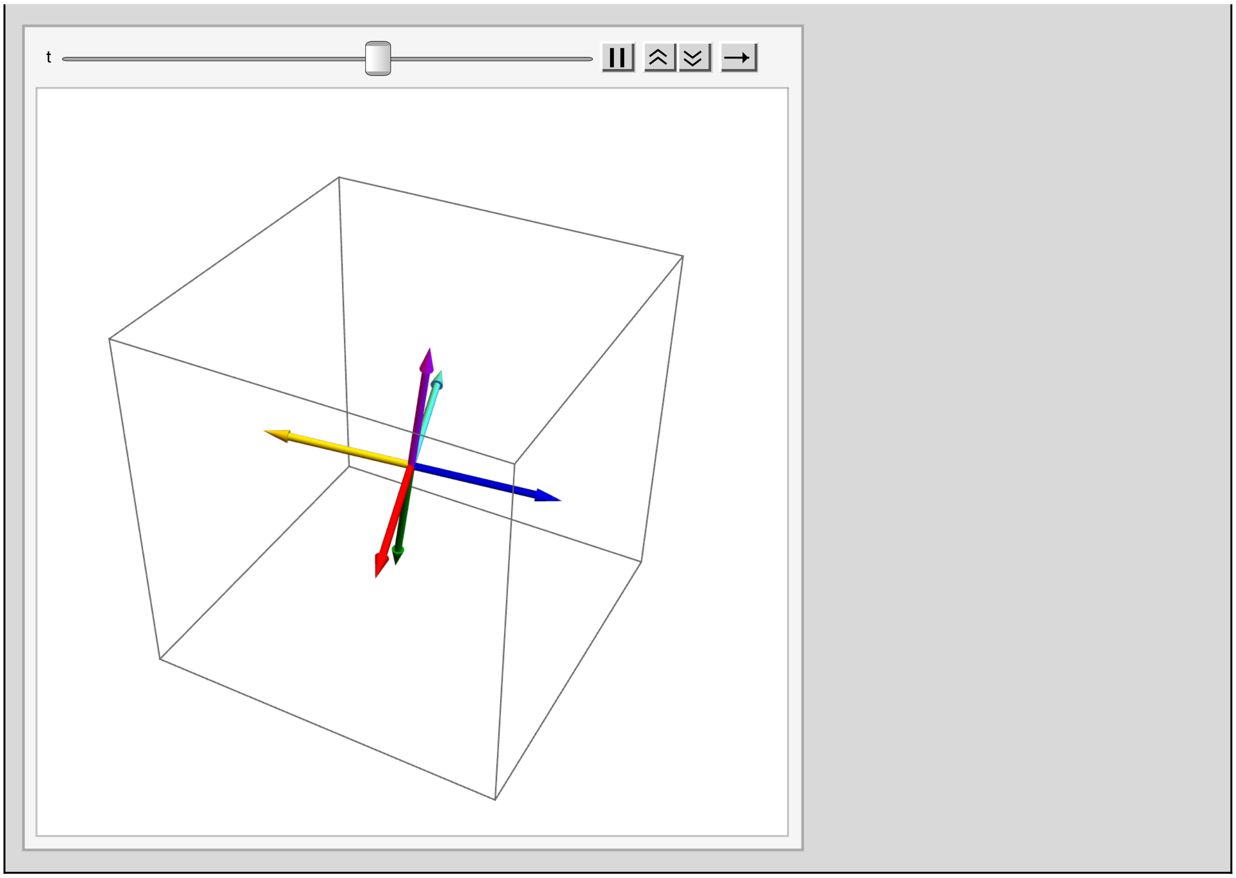


Another playground:

In[68]:=

```
With[{pr = 1.25},
 With[{prl = {{-pr, pr}, {-pr, pr}, {-pr, pr}}},
  With[{ϕ = nsoln⟦1, 1, 2⟧, θ = nsoln⟦1, 2, 2⟧, ψ = nsoln⟦1, 3, 2⟧},
   Animate[
    Evaluate@
     Graphics3D[{
       eulerRotate[
        jack[1.0, 0.025], ϕ, θ, ψ]}, PlotRange → prl],
    {t, 0, tN}]]]]
```

Out[68]=



Once Again, via Reference 5

After writing a Lagrangian equal to ours, Reference 5 takes a different tack to the equations of motion. Their tack is optimized for calculation by hand. It's manifestly shorter, simpler, and easier to read than ours. We'll compare our results, already checked against Reference 4, to theirs.

Lagrangian, L5

Reference 5 makes no assumptions other than lateral symmetry about the moment of inertia. Write our Lagrangian `L5` with their inertia matrix:

In[ ]:=
```
ClearAll[T5, L5];
T5[state : {ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] :=
  (1/2 ω[state]ᵀ.{{I1, 0, 0}, {0, I1, 0}, {0, 0, I3}}.ω[state])[[1, 1]];
L5[state : {ϕ_, θ_, ψ_, ϕd_, θd_, ψd_}] := T5[state] - V[state]

With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
   L5[state]] // Expand // FullSimplify
```

Out[ ]=

$$\frac{1}{4}\left(2\,\text{I1}\,\theta d^2+\text{I1}\,\phi d^2+\text{I3}\,\phi d^2+2\,\text{I3}\,\psi d^2+(-2\,c\,g\,m+4\,\text{I3}\,\phi d\,\psi d)\,\text{Cos}[\theta]+(-\text{I1}+\text{I3})\,\phi d^2\,\text{Cos}[2\,\theta]\right)$$

Compare to a handwritten transcription of their Lagrangian from 13S3:

In[ ]:=
```
((1/2 I1 (θd^2 + ϕd^2 Sin[θ]^2) + 1/2 I3 (ψd + ϕd Cos[θ])^2 - m c g Cos[θ] / 2) -
   With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}}, L5[state]]) // FullSimplify
```

Out[ ]=

0

Equations of Motion

Build up the equations of motion, as before. Reference 5 doesn’t ultimately use these verbatim, so this is an exercise in checking our derivations versus theirs.

Here is $\frac{\partial L}{\partial q}$:

In[ ]:=
```
ClearAll[dLdq5];
(dLdq5 = (With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
       Table[D[L5[state], q], {q, {ϕ, θ, ψ}}]] /. fTimeRules // FullSimplify)) //
 MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ \frac{1}{2}\,\text{Sin}[\theta[t]]\,\left(c\,g\,m+2\,(\text{I1}-\text{I3})\,\text{Cos}[\theta[t]]\,\phi'[t]^2-2\,\text{I3}\,\phi'[t]\,\psi'[t]\right) \\ 0 \end{pmatrix}$$

Here is $\frac{\partial L}{\partial \dot{q}}$:

In[∘]:=
```
ClearAll[dLdqDot5];
(dLdqDot5 = (With[{state = {ϕ, θ, ψ, ϕd, θd, ψd}},
        Table[D[L5[state], qd], {qd, {ϕd, θd, ψd}}]] /. fTimeRules // FullSimplify))
 MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} \frac{1}{2}\,(\text{I1}+\text{I3}+(-\text{I1}+\text{I3})\,\text{Cos}[2\,\theta[t]])\,\phi'[t]+\text{I3}\,\text{Cos}[\theta[t]]\,\psi'[t] \\ \text{I1}\,\theta'[t] \\ \text{I3}\,(\text{Cos}[\theta[t]]\,\phi'[t]+\psi'[t]) \end{pmatrix}$$

Here is $\frac{d}{dt}\frac{\partial L}{\partial \dot{q}}$; we augment our existing constants rules about *m*, *g*, and *l* with two new ones expressing constancy of their inertia matrix:

In[∘]:=
```
ClearAll[ddLdqDotdt5, constRules5];
constRules5 = constRules~Join~{Dt[I1, t] → 0, Dt[I3, t] → 0};
(ddLdqDotdt5 = (Dt[dLdqDot5, t] /. constRules5 // FullSimplify)) // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} \text{Sin}[\theta[t]]\,\theta'[t]\,(2\,(\text{I1}-\text{I3})\,\text{Cos}[\theta[t]]\,\phi'[t]-\text{I3}\,\psi'[t])+\frac{1}{2}\,(\text{I1}+\text{I3}+(-\text{I1}+\text{I3})\,\text{Cos}[2\,\theta[t \\ \text{I1}\,\theta''[t] \\ \text{I3}\,(-\text{Sin}[\theta[t]]\,\theta'[t]\,\phi'[t]+\text{Cos}[\theta[t]]\,\phi''[t]+\psi''[t]) \end{pmatrix}$$

Here are **eqns5**, the equations of motion, $\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0$:

In[∘]:=
```
ClearAll[eqns5];
(eqns5 = Table[(lhs == 0), {lhs, ddLdqDotdt5 - dLdq5}]) // MatrixForm
```

Out[∘]//MatrixForm=

$$\begin{pmatrix} \text{Sin}[\theta[t]]\,\theta'[t]\,(2\,(\text{I1}-\text{I3})\,\text{Cos}[\theta[t]]\,\phi'[t]-\text{I3}\,\psi'[t])+\frac{1}{2}\,(\text{I1}+\text{I3}+(-\text{I1}+\text{I3})\,\text{Cos}[2\,\theta[t \\ -\frac{1}{2}\,\text{Sin}[\theta[t]]\,\left(c\,g\,m+2\,(\text{I1}-\text{I3})\,\text{Cos}[\theta[t]]\,\phi'[t]^2-2\,\text{I3}\,\phi'[t]\,\psi'[t]\right)+ \\ \text{I3}\,(-\text{Sin}[\theta[t]]\,\theta'[t]\,\phi'[t]+\text{Cos}[\theta[t]]\,\phi''[t]+\psi''[t]) == \end{pmatrix}$$

These equations are still second-order, but Reference 5 reduces two of them to first order. Let’s go!

Conjugate Momenta

The first (azimuthal, $p_\phi \overset{\text{def}}{=} \partial L / \partial \dot{\phi}$) and third (nodal, $p_\psi \overset{\text{def}}{=} \partial L / \partial \dot{\psi}$) conjugate momenta should be conserved because $\partial L / \mathrm{d}\phi$ and $\partial L / \mathrm{d}\psi$ vanish.

pϕ5: Azimuthal Momentum

Mathematica rewrites $\partial L / \partial \dot{\phi}$ via the school half-angle formula. Save a rule for checking expressions later.

In[ ]:=
```
(2 Cos[θ]^2 - 1 == Cos[2 θ]) // FullSimplify
ClearAll[cos2rule];
cos2rule = {Cos[2 θ[t]] → 2 Cos[θ[t]]^2 - 1};
```
Out[ ]=
```
True
```

Reference 5 avoids the half-angle formula for a more concise expression:

In[ ]:=
```
ClearAll[pϕ5];
pϕ5 = I3 ψ'[t] Cos[θ[t]] + (I1 Sin[θ[t]]^2 + I3 Cos[θ[t]]^2) ϕ'[t];
```

Mathematica proves that ours equals theirs:

In[ ]:=
```
(dLdqDot5[[1]] == pϕ5) // FullSimplify
```
Out[ ]=
```
True
```

Reference 5 assumes p_ϕ is constant and writes a new convenience constant, *b*:

In[ ]:=
```
ClearAll[b];
b = pϕ5 / I1
```
Out[ ]=
$$\frac{\left(\mathrm{I3}\,\mathrm{Cos}[\theta[t]]^2+\mathrm{I1}\,\mathrm{Sin}[\theta[t]]^2\right)\phi'[t]+\mathrm{I3}\,\mathrm{Cos}[\theta[t]]\,\psi'[t]}{\mathrm{I1}}$$

pψ5: Nodal Momentum

Mathematica and Reference 5 agree verbatim on the expression of p_ψ:

In[ ]:=
```
ClearAll[pψ5];
pψ5 = dLdqDot5[[3]]
```
Out[ ]=
```
I3 (Cos[θ[t]] ϕ'[t] + ψ'[t])
```

Reference 5 assumes p_ψ is constant and writes a new convenience constant, *a*:

In[ ]:=
```
ClearAll[a];
a = pψ5 / I1
```

Out[ ]=

$$\frac{\text{I3}\,(\text{Cos}[\theta[t]]\,\phi'[t] + \psi'[t])}{\text{I1}}$$

Two First-Order Equations of Motion

Reference 5 claims that $\dot{\phi} = \frac{b - a\,\text{Cos}[\theta]}{\text{Sin}[\theta]^2}$. Let's check it:

In[ ]:=
```
(ϕ'[t] == (b - a Cos[θ[t]]) / Sin[θ[t]]^2) // FullSimplify
```

Out[ ]=
```
True
```

Reference 5 claims that $\dot{\psi} = a\,\frac{\text{I1}}{\text{I3}} - \frac{(b - a\,\text{Cos}[\theta])\,\text{Cos}[\theta]}{\text{Sin}[\theta]^2}$. Let's check it:

In[ ]:=
```
(ψ'[t] == a I1/I3 - (b - a Cos[θ[t]]) Cos[θ[t]] / Sin[θ[t]]^2) // FullSimplify
```

Out[ ]=
```
True
```

We notice that $\dot{\phi}$ is hiding in plain sight in that expression, and find the following:

In[ ]:=
```
(a I1/I3 - ϕ'[t] Cos[θ[t]] == ψ'[t]) // FullSimplify
```

Out[ ]=
```
True
```

Clear the constants *a* and *b* so we can assign numerical values later. Package the facts above in two equations, `foEqns`, and in replacement rules for *a* and *b*.

In[•]:=

```
ClearAll[foEqns, a, b, abRules];
foEqns = ({ϕ'[t] == (b - a Cos[θ[t]])/Sin[θ[t]]^2,
    ψ'[t] == a I1/I3 - ((b - a Cos[θ[t]]) Cos[θ[t]])/Sin[θ[t]]^2} // FullSimplify)
abRules = ({a → dLdqDot5[[3]]/I1, b → (dLdqDot5[[1]]/I1 /. cos2rule)} // FullSimplify)
```

Out[•]=

$$\left\{\phi'[t] == (b - a\,\mathrm{Cos}[\theta[t]])\,\mathrm{Csc}[\theta[t]]^2,\ b\,\mathrm{Cot}[\theta[t]]\,\mathrm{Csc}[\theta[t]] + \psi'[t] == a\left(\frac{I1}{I3} + \mathrm{Cot}[\theta[t]]^2\right)\right\}$$

Out[•]=

$$\left\{a \to \frac{I3\,(\mathrm{Cos}[\theta[t]]\,\phi'[t] + \psi'[t])}{I1},\ b \to \frac{\left(I1 + (-I1 + I3)\,\mathrm{Cos}[\theta[t]]^2\right)\phi'[t] + I3\,\mathrm{Cos}[\theta[t]]\,\psi'[t]}{I1}\right\}$$

Check one last time

In[•]:=

```
(pϕ5 == (I1 b /. abRules /. cos2rule)) // FullSimplify
```

Out[•]=

```
True
```

One Second-Order Equation for Nutation

In[•]:=

```
eqns5[[2]]
```

Out[•]=

$$-\frac{1}{2}\mathrm{Sin}[\theta[t]]\left(c\,g\,m + 2\,(I1 - I3)\,\mathrm{Cos}[\theta[t]]\,\phi'[t]^2 - 2\,I3\,\phi'[t]\,\psi'[t]\right) + I1\,\theta''[t] == 0$$

In[•]:=

```
ClearAll[soEqn];
soEqn =
  (I1 θ''[t] == I1 (a^2 + b^2) (Cos[θ[t]]/Sin[θ[t]]^3) - I1 a b ((3 + Cos[2 θ[t]])/(2 Sin[θ[t]]^3)) + m g l Sin[θ[t]])
```

Out[•]=

$$I1\,\theta''[t] == \left(a^2 + b^2\right) I1\,\mathrm{Cot}[\theta[t]]\,\mathrm{Csc}[\theta[t]]^2 - \frac{1}{2}\,a\,b\,I1\,(3 + \mathrm{Cos}[2\,\theta[t]])\,\mathrm{Csc}[\theta[t]]^3 + g\,l\,m\,\mathrm{Sin}[\theta[t]]$$

Three New Constants

α, and β are obviously constant. We'll numerically check the claim that E_1 is also constant.

In[ ]:=
```
ClearAll[E1, α, β, newConstRules];
newConstRules = {E1 → m g l Cos[θ[t]] + 1/2 I1 (θ'[t]^2 + ((b - a Cos[θ[t]])/Sin[θ[t]])^2),
   α → (2 E1)/I1, β → (2 m g l)/I1};
```

New Nutation Equation

In[ ]:=
```
ClearAll[soEqn2];
soEqn2 = (θ''[t] == (a^2 + b^2) (Cos[θ[t]]/Sin[θ[t]]^3) - a b ((3 + Cos[2 θ[t]])/(2 Sin[θ[t]]^3)) + β/2 Sin[θ[t]])
```

Out[ ]=
```
θ''[t] ==
  (a^2 + b^2) Cot[θ[t]] Csc[θ[t]]^2 - 1/2 a b (3 + Cos[2 θ[t]]) Csc[θ[t]]^3 + 1/2 β Sin[θ[t]]
```

Prove the two equations for $\ddot{\theta}$ are equivalent.

In[ ]:=
```
(soEqn[[2]] - I1 soEqn2[[2]]) /. newConstRules // FullSimplify
```

Out[ ]=
```
0
```

Numerical Values for the Constants

The following numerical values are taken directly from Reference 5.

In[ ]:=
```
ClearAll[α, β, a, b];
numRules5 = {α → 1.6, β → 2.0, a → 2.5, b → 1.7, I1 → 2, I3 → 1};
```

For later reconciliation of References 4 and 5, we must back out values of m, g, and l, the only dependencies of our Reference-4 solution, from numRules5.

If $\alpha = 1.6$ and $I_1 = 2$, then $E_1 = \alpha\ I_1/2 \to \alpha \to 1.6$.

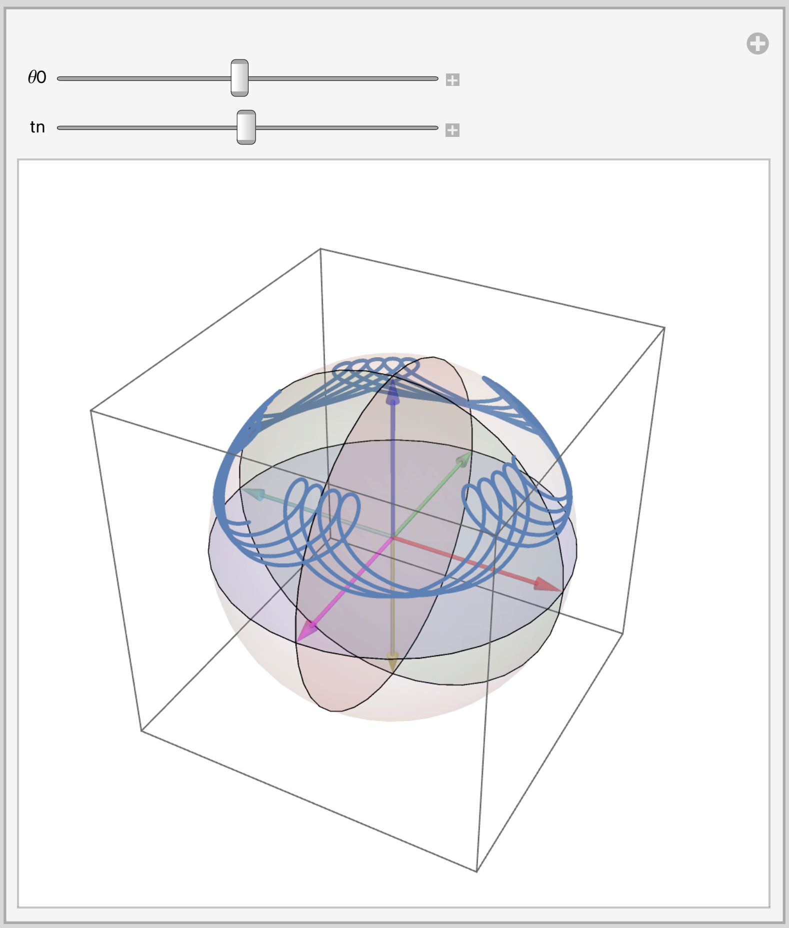
If $\beta = 2.0$ and $I_1 = 2$, then $m\ g\ l = 2$.

Numerical Solution

The following reproduces the solutions exhibited in Reference 5, developed from scratch rather than verbatim. I.e., we've validated Reference 5.

In[ ]:=
```
With[{
  initials = {θ'[0] == 0, ϕ[0] == 0 °, ψ[0] == 0 °},
  eqns5better = {foEqns[[1]], foEqns[[2]], soEqn2} /. numRules5},
 Manipulate[
  Module[{
    eq1 = NDSolve[eqns5better~Join~initials~Join~{θ[0] == θ0 * °},
      {ϕ[t], θ[t], ψ[t]},
      {t, 0, tn}]},
   renderSoln[eq1, tn]],
  {{θ0, 43}, 0, 90}, {{tn, 70}, 0.0001, 140}]]
```

Out[ ]=



Reconciling References 4 and 5

Energy

Energy E5

E is a reserved symbol in Mathematica, so we called our our old energy E4, with homage to Reference 4, and the new energy E5, with homage to Reference 5.

The nutation equation, for *θ*[*t*], is the only second-order equation we'll need to deal with after incorporating the conserved momenta later. Check ours against a hand-written copy from Reference 5, Section 13S3.

Mathematica has a harmless minus sign

In[ ]:=
```
-eqns5[[2, 1]] // Expand // FullSimplify
```
Out[ ]=
```
(g l m - I3 ϕd ψd + (I1 - I3) ϕd^2 Cos[θ]) Sin[θ] - I1 θ''[t]
```

Here is a handwritten copy from Reference 5; by inspection, it perfectly matches ours:

In[ ]:=
```
Sin[θ[t]] (m g l + (I1 - I3) ϕ'[t]^2 Cos[θ[t]]) -
   I3 ϕ'[t] × ψ'[t] Sin[θ[t]] - I1 θ''[t] // Expand // FullSimplify
```
Out[ ]=
```
g l m Sin[θ[t]] + (I1 - I3) Cos[θ[t]] Sin[θ[t]] ϕ'[t]^2 -
 I3 Sin[θ[t]] ϕ'[t] ψ'[t] - I1 θ''[t]
```

Mathematica is not able to cancel out the terms, but it is of no consequence. The following has a leading factor of sin*θ* – sin*θ*, obviously zero:

In[ ]:=
```
(-eqns5[[2, 1]] - (Sin[θ[t]] (m g l + (I1 - I3) ϕ'[t]^2 Cos[θ[t]]) -
      I3 ϕ'[t] × ψ'[t] Sin[θ[t]] - I1 θ''[t])) // FullSimplify
```
Out[ ]=
```
(g l m - I3 ϕd ψd + (I1 - I3) ϕd^2 Cos[θ]) Sin[θ] -
 Sin[θ[t]] (g l m + (I1 - I3) Cos[θ[t]] ϕ'[t]^2) + I3 Sin[θ[t]] ϕ'[t] ψ'[t]
```

Junkyard

Ignore stuff in here.

Damped Solution

Mathematica easily solves the equation and we pick the only pertinent part of the expression returned by `DSolve`:

In[ ]:=

```
dsoln = (DSolve[deqn, x, t])⟦1, 1, 2⟧
```

DSolve : Equation or list of equations expected instead of deqn in the first argument deqn.

Part : Part specification DSolve [deqn, x, t]⟦1, 1, 2 ⟧ is longer than depth of object.

Out[ ]=

```
DSolve[deqn, x, t]⟦1, 1, 2⟧
```

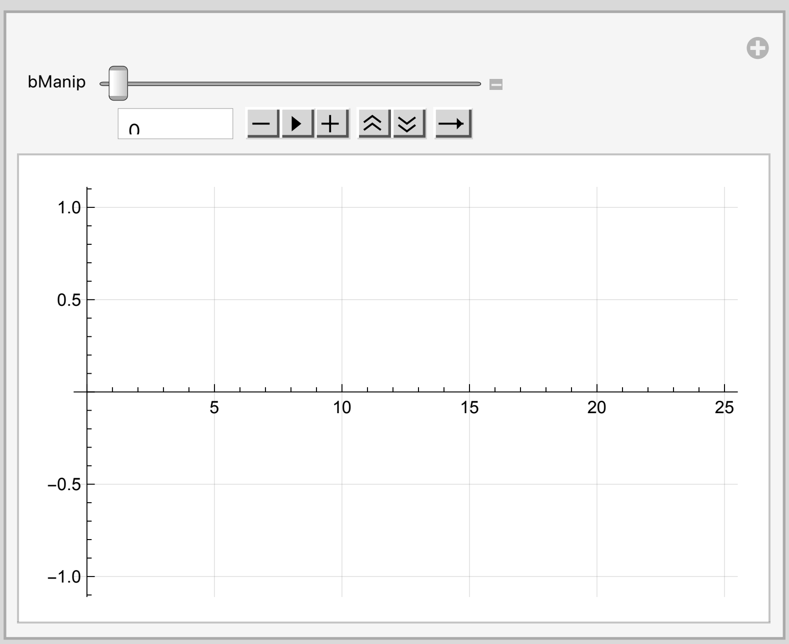
For plotting, we must be a bit careful, because if b^2 is smaller than $4\,m\,\kappa$, we get complex arguments to the exponentials, yielding oscillation, but that's the interesting part! Only the real parts of the solution are physical.

Convince yourself that as b^2 varies from 0 to $4\,m\,\kappa$, the system transitions from undamped, to underdamped (showing some oscillation), to critically damped, to overdamped (showing no oscillation). See this page on damping.

In[ ]:=

```
Manipulate[Plot[Re[dsoln[t] /. {κ → 1, m → 1, b → bManip, C[2] → 1, C[1] → 0}],
  {t, 0, 25}, GridLines → Automatic],
 {bManip, 0, 4, Appearance → "Open"}]
```

Out[ ]=



Discrete Solution

Reference 2 gives us the ***forced discrete Euler-Lagrange Equations***:

where L_d is the discrete Lagrangian and F_d^- and F_d^+ are the left and right discrete forces. These should approximate the continuous forcing so that

$$F_d^-(q_k, q_{k+1}) \cdot \delta q_k + F_d^+(q_k, q_{k+1}) \cdot \delta q_{k+1} \approx \int_{t_k}^{t_{k+1}} F(q(t), \dot{q}(t)) \cdot \delta q \; dt.$$

The equation (1.27) defines an integrator $(q_k, q_{k+1}) \mapsto (q_{k+1}, q_{k+2})$ given implicitly by the ***forced discrete Euler-Lagrange equations***:

$$D_1 L_d(q_{k+1}, q_{k+2}) + D_2 L_d(q_k, q_{k+1}) + F_d^-(q_{k+1}, q_{k+2}) + F_d^+(q_k, q_{k+1}) = 0. \qquad (1.28)$$

The simplest example of discrete forces is to take

In[ ]:=

$$F_d^-(q_k, q_{k+1}) = F(q_k)$$
$$F_d^+(q_k, q_{k+1}) = 0,$$

which, together with the discrete Lagrangian (1.3), gives the forced Euler-Lagrange equations

$$M\left(\frac{q_{k+1} - 2q_k + q_{k-1}}{h^2}\right) = -\nabla V(q_k) + F(q_k).$$

The position-momentum form of a variational integrator with forcing is useful for implementation purposes. This is given by

$$p_k = -D_1 L_d(q_k, q_{k+1}) - F_d^-(q_k, q_{k+1})$$
$$p_{k+1} = D_2 L_d(q_k, q_{k+1}) + F_d^+(q_k, q_{k+1}).$$

References 2 and 3 are very difficult and long. The derivations dive very deeply into differential geometry, topology, and more. No shallow reading can give us a recipe for the force terms F_d^- and F_d^+. However, from our success with the undamped simple harmonic oscillator, we can hope for an easy extension.

Start with quadrature of the unforced discrete Lagrangian according to the third, unnumbered equation on page 643 of Reference 3, and introducing another free variable, the time step h:

Simple Example: Consider a continuous, typical Lagrangian of the form $L(q, \dot{q}) = \frac{1}{2}\dot{q}^T M \dot{q} - V(q)$ (V being a potential function) and define the discrete Lagrangian $L_d(q_k, q_{k+1}) = hL(q_{k+\frac{1}{2}}, (q_{k+1} - q_k)/h)$ by the use of the notation $q_{k+\frac{1}{2}} := (q_k + q_{k+1})/2$. The resulting update equation is

$$M\frac{q_{k+1} - 2q_k + q_{k-1}}{h^2} = -\frac{1}{2}(\nabla V(q_{k-\frac{1}{2}}) + \nabla V(q_{k+\frac{1}{2}}))$$

Guess that

In[∘]:=
```
ClearAll[FMinus, FPlus];
FMinus[q_] := D[F[q, v, t], v] /. {v → (q[k + 1] - q[k])/h};
FPlus[q_] := D[F[q, v, t], v] /. {v → (q[k] - q[k - 1])/h}
```

and write

In[∘]:=
```
discreteEqn =
  m ((q[k + 1] - 2 q[k] + q[k - 1])/h^2) == -1/2 ((D[V[q], q] /. {q → (q[k - 1] + q[k])/2}) +
     (D[V[q], q] /. {q → (q[k] + q[k + 1])/2}) + FMinus[q] + FPlus[q])
```

Out[∘]=

$$\frac{m\,(q[-1+k] - 2\,q[k] + q[1+k])}{h^2} == \frac{1}{2}\left(-V'\left[\frac{1}{2}\,(q[-1+k] + q[k])\right] - V'\left[\frac{1}{2}\,(q[k] + q[1+k])\right] - F^{(0,1,0)}\left[q, \frac{-q[-1+k] + q[k]}{h}, t\right] - F^{(0,1,0)}\left[q, \frac{-q[k] + q[1+k]}{h}, t\right]\right)$$

Algebraic solution of the discrete equation for time step $k + 1$ given solutions for time steps $k - 1$ and k. Do not confuse time step k with spring constant κ. As with most quadratics, we get two solutions.

In[∘]:=
```
discreteSolns = (Solve[discreteEqn, q[1 + k]] // FullSimplify)
```

Solve : This system cannot be solved with the methods available to Solve. Try Reduce or FindInstance instead.

Out[∘]=

$$\text{Solve}\left[\frac{2\,m\,(q[-1+k] - 2\,q[k] + q[1+k])}{h} + h\left(V'\left[\frac{1}{2}\,(q[-1+k] + q[k])\right] + V'\left[\frac{1}{2}\,(q[k] + q[1+k])\right] + F^{(0,1,0)}\left[q, \frac{-q[-1+k] + q[k]}{h}, t\right] + F^{(0,1,0)}\left[q, \frac{-q[k] + q[1+k]}{h}, t\right]\right) == 0,\ q[1+k]\right]$$

Pick the first one to start:

In[ ]:=
```
discreteSoln1 = discreteSolns[[1, 1, 2]] /. {κ → 1, m → 1}
```

Out[ ]=

$$h\left(V'\left[\frac{1}{2}\,(q[-1+k]+q[k])\right]+V'\left[\frac{1}{2}\,(q[k]+q[1+k])\right]+F^{(0,1,0)}\left[q,\,\frac{-q[-1+k]+q[k]}{h},\,t\right]+F^{(0,1,0)}\left[q,\,\frac{-q[k]+q[1+k]}{h},\,t\right]\right)$$

Compare the plot of the analytical solution versus the first discrete solution for various choices of the time step, represented by its negative logarithm base 10, and for the damping parameter *b*.

In[ ]:=

```
ClearAll[experiment];
experiment[bManip_, timeSteps_, h_] :=
  Module[{k, q, x},
   q[k_] := x⟦1 + k⟧; (* Let us start at k == 0 *)
   x = ConstantArray[0, timeSteps + 1];
   (* bootstrap discrete solution from analytical solution *)
   x⟦1⟧ = Re[dsoln[0] /. {b → bManip, ϰ → 1, m → 1, C[2] → 1, C[1] → 0}];
   x⟦2⟧ = Re[dsoln[h] /. {b → bManip, ϰ → 1, m → 1, C[2] → 1, C[1] → 0}];
   For[k = 1, k < timeSteps, k++,
    (* Must manually paste the solution -- reason unknown *)
    x⟦k + 2⟧ = ((-4 + 2 b h - h^2) q[-1 + k] + 2 (4 - h^2) q[k]) / (4 + 2 b h + h^2) /. b → bManip];
   GraphicsRow[{
     ListLinePlot[x, Frame → True,
      FrameLabel → {{"Amplitude", ""}, {"Time Step -- units of h",
         "Discrete Solution"}}, GridLines → Automatic],
     Plot[Re[dsoln[τ] /. {b → bManip, ϰ → 1, m → 1, C[2] → 1, C[1] → 0}],
      {τ, 0, timeSteps h}, Frame → True,
      GridLines → Automatic, FrameLabel → {{"Amplitude", ""},
        {"Time Step -- units of 1/h", "Analytical Solution"}}]}]];
Manipulate[experiment[bManip, 25 / 10^minuslog10h, 10.^minuslog10h],
 {bManip, 0, 4, Appearance → "Open"},
 {minuslog10h, {0, -1, -2, -3, -4}}]
```

Out[ ]=

Total success!