Spinning disk pendulum that swings on top of a rotating table

Initialization Code (optional)

Manipulate

```
Manipulate[

 (
  {solθ, solϕ, solψ} = getSolution[rSmall, hSmall, ρSmall,
    rLarge, hLarge, ρLarge, len, θ0, θ0Speed 2 Pi, ψ0, ψ0Speed 2 Pi, ϕ0, ϕ0Speed 2 Pi ];

  (*set parameters according to the test case to run*)

  If[Not[testCase == 0],
   Which[
    testCase == 1,
    {len = lPost; ρSmall = ρmax; ρLarge = ρmin; rSmall = maxrSmall; rLarge = maxrLarge; hSmall = maxhSmall;
     hLarge = 0.5 maxhLarge; θ0 = 15; θ0Speed = 0.5 maxω; ψ0 = 0; ψ0Speed = 0.2 maxω; ϕ0 = 40; ϕ0Speed = .3 maxω;
     viewPoint = {1.3, -2.4, .5}; angularMomentumOption = 0; showI = True; zoom = 1; animRate = 0.03},

    testCase == 2, {len = lPost; ρSmall = ρmin; ρLarge = ρmin; rSmall = maxrSmall; rLarge = maxrLarge;
     hSmall = minhSmall; hLarge = 0.5 maxhLarge; θ0 = 0; θ0Speed = maxω; ψ0 = 0; ψ0Speed = maxω; ϕ0 = 0; ϕ0Speed = 0;
     viewPoint = {1.3, -2.4, .5}; angularMomentumOption = 1; showI = False; zoom = 1; animRate = 0.01},

    testCase == 3, {len = lPost; ρSmall = ρmax; ρLarge = ρmin; rSmall = maxrSmall; rLarge = 0.6 maxrLarge;
     hSmall = 0.5 maxhSmall; hLarge = maxhLarge; θ0 = 30; θ0Speed = 0.3 maxω; ψ0 = 0; ψ0Speed = 0; ϕ0 = 0;
     ϕ0Speed = 0; viewPoint = {2, -2, 0}; angularMomentumOption = 1; showI = True; zoom = 1; animRate = 0.05},

    testCase == 4, {len = lPost; ρSmall = ρmax; ρLarge = ρmin; rSmall = 0.65 maxrSmall;
     rLarge = 0.6 maxrLarge; hSmall = 0.5 maxhSmall; hLarge = maxhLarge; θ0 = 0; θ0Speed = maxω;
     ψ0 = 0; ψ0Speed = 0.85 maxω; ϕ0 = 0; ϕ0Speed = maxω; viewPoint = {Pi, Pi / 2, 2};
     angularMomentumOption = 0; showI = False; zoom = .73; animRate = 0.02},

    testCase == 5, {len = lPost; ρSmall = 10; ρLarge = ρmin; rSmall = 0.65 maxrSmall; rLarge = 0.6 maxrLarge;
     hSmall = minhSmall; hLarge = maxhLarge; θ0 = 0; θ0Speed = maxω; ψ0 = 0; ψ0Speed = maxω; ϕ0 = 0; ϕ0Speed = maxω;
     viewPoint = {Pi, Pi / 2, 2}; angularMomentumOption = 0; showI = False; zoom = 0.74; animRate = 0.002},

    testCase == 6,
    {len = lPost; ρSmall = 10; ρLarge = ρmax; rSmall = 0.65 maxrSmall; rLarge = minrLarge; hSmall = minhSmall;
     hLarge = minhLarge; θ0 = 133; θ0Speed = 0.5 maxω; ψ0 = 280; ψ0Speed = minω; ϕ0 = 135; ϕ0Speed = maxω;
     viewPoint = {Pi, Pi / 2, 2}; angularMomentumOption = 0; showI = False; zoom = 0.76; animRate = 0.002},

    testCase == 7, {len = lPost; ρSmall = ρmax; ρLarge = ρmin; rSmall = maxrSmall; rLarge = maxrLarge;
     hSmall = minhSmall; hLarge = 0.25 maxhLarge; θ0 = 0; θ0Speed = -0.3; ψ0 = 0; ψ0Speed = 0; ϕ0 = 0; ϕ0Speed = 0;
     viewPoint = {Pi, Pi / 2, 2}; angularMomentumOption = 0; showI = False; zoom = 1; animRate = 0.06},

    testCase == 8,
    {len = lPost; ρSmall = ρmax; ρLarge = ρmin; rSmall = maxrSmall; rLarge = maxrLarge; hSmall = maxhSmall;
     hLarge = 0.5 maxhLarge; θ0 = 15; θ0Speed = 0.5 maxω; ψ0 = 0; ψ0Speed = 0.2 maxω; ϕ0 = 40; ϕ0Speed = .3 maxω;
     viewPoint = {1.3, -2.4, .5}; angularMomentumOption = 9; showI = True; zoom = 1; animRate = 0.03},

    testCase == 9, {len = lPost; ρSmall = ρmax; ρLarge = ρmin; rSmall = .2 maxrSmall; rLarge = .8 maxrLarge;
```



```
    hSmall = maxhSmall; hLarge = 0.5 maxhLarge; θ0 = 0; θ0Speed = maxω; ψ0 = 0; ψ0Speed = 0; ϕ0 = 83; ϕ0Speed = 0;
    viewPoint = {1.3, -2.4, .5}; angularMomentumOption = 0; showI = False; zoom = 1; animRate = 0.018}
  ]
 ];

 Dynamic[
  update[len, ρSmall, ρLarge, rSmall, rLarge, hSmall, hLarge, currentTime, viewPoint, boxIt,
   angularMomentumOption, showI, zoom, testCase, solθ, solϕ, solψ, traceThickness, isTraceOn],
  TrackedSymbols :→ {currentTime, solθ, solϕ, solψ, viewPoint, boxIt,
    angularMomentumOption, showI, zoom, testCase}]

),

(*------------------------*)
(* L E F T   P A N E L   *)
(*------------------------*)
Item[
 Grid[{

   {
    Button[Style["min", 10], len = 0.1 lPost, ImageSize → Tiny],
    Button[Style["max", 10], len = lPost, ImageSize → Tiny],
    Control[{{len, lPost, Text[Style["len", fontSizeForControl]]},
      0.3 lPost, lPost, .1, ImageSize → Tiny, Appearance → "Labeled"}]
   },

   {
    Button[Style["min", 10], ρSmall = ρmin, ImageSize → Tiny],
    Button[Style["max", 10], ρSmall = ρmax, ImageSize → Tiny],
    Control[{{ρSmall, ρmax, Text[Style["ρm", fontSizeForControl]]},
      ρmin, ρmax, .1, ImageSize → Tiny, Appearance → "Labeled"}]
   },

   {
    Button[Style["min", 10], ρLarge = 1, ImageSize → Tiny],
    Button[Style["max", 10], ρLarge = 10, ImageSize → Tiny],
    Control[{{ρLarge, ρmin, Text[Style["ρM", fontSizeForControl]]},
      ρmin, ρmax, .1, ImageSize → Tiny, Appearance → "Labeled"}]
   },

   {
    Button[Style["min", 10], rSmall = minrSmall, ImageSize → Tiny],
    Button[Style["max", 10], rSmall = maxrSmall, ImageSize → Tiny],
    Control[{{rSmall, Mean[{minrSmall, maxrSmall}], Text[Style["r", Italic, fontSizeForControl]]},
      minrSmall, maxrSmall, .1, ImageSize → Tiny, Appearance → "Labeled"}]
   },

   {
    Button[Style["min", 10], rLarge = minrLarge, ImageSize → Tiny],
    Button[Style["max", 10], rLarge = maxrLarge, ImageSize → Tiny],
    Control[{{rLarge, maxrLarge, Text[Style["R", Italic, fontSizeForControl]]},
      minrLarge, maxrLarge, .1, ImageSize → Tiny, Appearance → "Labeled"}]
   },

   {
    Button[Style["min", 10], hSmall = minhSmall, ImageSize → Tiny],
    Button[Style["max", 10], hSmall = maxhSmall, ImageSize → Tiny],
    Control[{{hSmall, maxhSmall, Text[Style["h", Italic, fontSizeForControl]]},
      minhSmall, maxhSmall, .1, ImageSize → Tiny, Appearance → "Labeled"}]
   },

   {
    Button[Style["min", 10], hLarge = minhLarge, ImageSize → Tiny],
    Button[Style["max", 10], hLarge = maxhLarge, ImageSize → Tiny],
```

```
     Control[{{hLarge, 0.5 maxhLarge, Text[Style["H", Italic, fontSizeForControl]]},
       minhLarge, maxhLarge, .1, ImageSize → Tiny, Appearance → "Labeled"}]
    }

   }, Spacings → 0.5, Frame → All, FrameStyle -> Directive[Thickness[.001], Gray]], ControlPlacement → Left
  ],

  Item[Grid[{
    (*Initial conditions*)
    {
     Button[Style["min", 10], θ0 = 0, ImageSize → Tiny],
     Button[Style["mid", 10], θ0 = 180, ImageSize → Tiny],
     Control[{{θ0, 133, Text[Style["θ  ", fontSizeForControl]]},
       0, 360, 1, ImageSize → Tiny, Appearance → "Labeled"}]
    },

    {
     Button[Style["min", 10], ψ0 = 0, ImageSize → Tiny],
     Button[Style["mid", 10], ψ0 = 180, ImageSize → Tiny],
     Control[{{ψ0, 186, Text[Style["ψ  ", fontSizeForControl]]},
       0, 360, 1, ImageSize → Tiny, Appearance → "Labeled"}]
    },

    {
     Button[Style["min", 10], ϕ0 = 0, ImageSize → Tiny],
     Button[Style["mid", 10], ϕ0 = 180, ImageSize → Tiny],
     Control[{{ϕ0, 112, Text[Style["ϕ  ", fontSizeForControl]]},
       0, 360, 1, ImageSize → Tiny, Appearance → "Labeled"}]
    },
    (*initial speeds*)
    {
     Button[Style["min", 10], θ0Speed = 0, ImageSize → Tiny],
     Button[Style["max", 10], θ0Speed = maxω, ImageSize → Tiny],
     Control[{{θ0Speed, 0.4 maxω, Text[Style["θ̇  ", fontSizeForControl]]},
       minω, maxω, .1, ImageSize → Tiny, Appearance → "Labeled"}]
    },

    {
     Button[Style["min", 10], ψ0Speed = 0, ImageSize → Tiny],
     Button[Style["max", 10], ψ0Speed = maxω, ImageSize → Tiny],
     Control[{{ψ0Speed, 0.2 maxω, Text[Style["ψ̇  ", fontSizeForControl]]},
       minω, maxω, .1, ImageSize → Tiny, Appearance → "Labeled"}]
    },

    {
     Button[Style["min", 10], ϕ0Speed = 0, ImageSize → Tiny],
     Button[Style["max", 10], ϕ0Speed = maxω, ImageSize → Tiny],
     Control[{{ϕ0Speed, .3 maxω, Text[Style["ϕ̇  ", fontSizeForControl]]},
       minω, maxω, .1, ImageSize → Tiny, Appearance → "Labeled"}]
    }

   }, Spacings → 0.5, Frame → All, FrameStyle -> Directive[Thickness[.001], Gray]],
   ControlPlacement → Left
  ],
```

```
Item[
 Grid[{

   {
    Control[
      {{animRate, 0.025, Style["step size", 12]}, .001, 0.1, .001, Appearance → "Labeled", ImageSize → Tiny}]
   },

   {
    Control[{{currentTime, 0, Style["run", 12]}, 0, maxSimulationTime , Dynamic[animRate],
       ControlType → Trigger, DisplayAllSteps → True, ImageSize → Tiny, AppearanceElements → {"ProgressSlider",
         "ResetPlayButton", "PauseButton", "StepLeftButton", "StepRightButton", "ResetButton"}}]
   },

   {
    Grid[{
       {
        Control[{{viewPoint, {1.3, -2.4, .5}, Style["viewpoint", 12]}, {{1.3, -2.4, .5} → Style["1", 11],
           {-0.57, -2.7, 0.063} → Style["2", 11], {2, -2, 0} → Style["3", 11], {2, 0, 0} → Style["4", 11],
           {0, 0, 2} → Style["5", 11], {1, -1, 1} → Style["6", 11], {3.96, -0.54, 0.54} → Style["7", 11],
           {Pi, Pi / 2, 2} → Style["8", 11]}, ControlType → SetterBar, ImageSize → Small}]
       }
      }, Spacings → .5, Frame → None]
   },

   {
    Grid[{
       {

        Grid[{{
           Style["display", 11],
           Control[{
              {angularMomentumOption, 0, ""},
              {0 → Text@Style["bob only", 11],
               1 → Text@Style["L", Italic, 11],
               2 → Text@Style["{Lx,Ly,Lz}", Italic, 10],
               3 → Text@
                 Row[{Style["L", Italic, 10], " + ", Style["L", Italic, 10], Style[" components", 10]}],
               4 → Text@Row[{Style["d", 10], Style["L", Italic, 10], Style["/d", 10],
                   Style["t", Italic, 10]}],
               5 → Text@Row[{Style["d", 10], Style["L", Italic, 10], Style["/d", 10],
                   Style["t", Italic, 10], Style[" components", 10]}],
               6 -> Text@Style["ω", 10],
               7 -> Text@Style["{ωx,ωy,ωz}", 10],
               8 -> Text@Row[{Style["ω and ", 10], Style["L", Italic, 10]}],
               9 -> Text@Row[{Style["d{θ,ψ,ϕ}/d", 10], Style["t", Italic, 10]}],
               10 → Text@Row[{Style["d", 10], Style["L", Italic, 10], Style["/d", 10],
                   Style["t", Italic, 10], Style[" and ", 10], Style["L", Italic, 10]}],
               11 → Text@Row[{Style["d", 10], Style["L", Italic, 10], Style["/d", 10],
                   Style["t", Italic, 10], Style[" and ", 10], Style["L", Italic, 10], Style[" and ω", 10]}]
              }, ControlType → PopupMenu, ImageSize → All}
           ]}}, Spacings → -.5, Frame → None, Alignment → Center],

        Grid[{
          {
           Style["test", 11],
           Control[{ {testCase, 0, ""},
              {0 → Style["0", Small],
               1 → Style["1", Small],
               2 → Style["2", Small],
               3 → Style["3", Small],
               4 → Style["4", Small],
               5 → Style["5", Small],
               6 → Style["6", Small],
```

```
            7 → Style["7", Small],
            8 → Style["8", Small],
            9 → Style["9", Small]
           }, ControlType → PopupMenu, ImageSize → All}
         ]}}, Spacings → -.5, Frame → None, Alignment → Center]

      }
     }, Spacings → .5, Frame → None]
    },

    {
     Grid[{
       {
        Style["time (sec)", 11],
        Dynamic[Style[PaddedForm[currentTime, {5, 3},
           NumberSigns → {"-", ""}, NumberPadding → {"0", "0"}, SignPadding → True], 11]]
       }}, Alignment → Center, Spacings → .8
     ]
    }

   }, Alignment → Left, Spacings → .5, Frame → All, FrameStyle -> Directive[Thickness[.001], Gray]],
  ControlPlacement → Left
 ],

 (*---------------------------*)
 (* R I G H T   P A N E L   *)
 (*---------------------------*)
 Item[Grid[{
    {Text[Style["zoom", 10]]},
    {Control[{ {zoom, 1, ""}, .73, 1, .01, ControlType → VerticalSlider, ImageSize → Small}]},
    {""},
    {Text[Style["info", 11]]},
    {Control[{ {showI, True, ""}, {True, False}, ControlType → Checkbox, ImageSize → Tiny}]},
    {Text[Style["box", 11]]},
    {Control[{ {boxIt, False, ""}, {True, False}, ControlType → Checkbox, ImageSize → Tiny}]},
    {""},
    {Grid[{
       {Text[Style["trace", 11]]},
       {Control[{ {isTraceOn, False, ""}, {True, False}, ControlType → Checkbox, ImageSize → Tiny}]},
       {Text@Style["length", 10]},
       {Control[{ {currentMaximumTraceSize, defaultTraceSize, ""},
          1, maxTraceSize, 1, ControlType → VerticalSlider, ImageSize → Tiny}]},
       {Text[Style["thickness", 10]]},
       {Control[{ {traceThickness, defaultTraceThickness, ""},
          0.001, 0.01, 0.001, ControlType → VerticalSlider, ImageSize → Small}]}
      }, Frame → True, FrameStyle → {Thin, Gray}, Spacings → 0]
    }
   }, Alignment → Center, Frame → {{1, 2} → True}, Spacings → .2], ControlPlacement → Right
 ],

 {{solθ, {}}, ControlType → None},
 {{solϕ, {}}, ControlType → None},
 {{solψ, {}}, ControlType → None},

 {{previousTestCaseNumber, 0}, ControlType → None},
 {{maxSimulationTime , 100}, ControlType → None},
 {{lPost, 10}, ControlType → None}, (*length of post below main table*)
 {{rPost, 0.1 lPost}, ControlType → None}, (*radius of post*)
 {{minhLarge, 0.1 lPost}, ControlType → None}, (*minumum height of table*)
 {{maxhLarge, lPost}, ControlType → None}, (*maximum height of table*)
 {{minrLarge, 11 rPost}, ControlType → None}, (*minumum radius of table*)
 {{maxrLarge, 20 rPost}, ControlType → None}, (*maximum radius of table*)
 {{minrSmall , 2 rPost}, ControlType → None}, (*minumum radius of bob disk*)
 {{maxrSmall , 10 rPost}, ControlType → None}, (*maximum radius of bob disk*)
```

```
{{minhSmall, 0.1 lPost}, ControlType → None}, (*minumum height of bob disk*)
{{maxhSmall , 0.5 lPost}, ControlType → None}, (*maximum height of bob disk*)
{{maxω, 1}, ControlType → None}, (*maximum angular velocity in hz*)
{{minω, -1}, ControlType → None}, (*minumum angular velocity in hz*)
{{ρmin, 1}, ControlType → None}, (*minumum density kg/m^3*)
{{ρmax, 10}, ControlType → None}, (*maximum density kg/m^3*)

(*these below is data and variables to track the center of mass of the pendulum*)
(*if trace is selected *)
{{defaultTraceThickness, 0.006}, ControlType → None},
{{maxTraceSize, 1000}, ControlType → None}, (*maximum trace points to keep*)
{{defaultTraceSize, 200}, ControlType → None},

Alignment → Center,
SynchronousUpdating → True,
SynchronousInitialization → True,
FrameMargins → 1,
ImageMargins → 1,
Initialization :→ (

  traceBuffer = Table[0, {maxTraceSize}]; (*where to store the trace coordinates*)
  previousMaxTraceSize = currentMaximumTraceSize;
  isFirstScan = True;
  currentTraceSize = 0;
  isSolutionChanged = False;

  fontSizeForControl = 11;

  (*-------------------------------------------*)
  (* helper function for formatting            *)
  (*-------------------------------------------*)
  padIt1[v_, f_List] :=
   AccountingForm[Chop[v] , f, NumberSigns → {"-", "+"}, NumberPadding → {"0", "0"}, SignPadding → True];

  (*-------------------------------------------*)
  (* helper function for formatting            *)
  (*-------------------------------------------*)
  padIt2[v_, f_List] :=
   AccountingForm[Chop[v] , f, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}, SignPadding → True];

  (*-------------------------------------------------*)
  (* main entry to find the numerical solution              *)
  (*-------------------------------------------------*)
  getSolution[rSmall_, hSmall_, ρSmall_, rLarge_,
    hLarge_, ρLarge_, len_, θ0_, θ0Speed_, ψ0_, ψ0Speed_, ϕ0_, ϕ0Speed_] :=
   Module[{mSmall, mLarge, Id, Icg, Io, kinetic, v, g = 9.8, lagrangian,
     eqs, initialConditions, sol, θ, ϕ, ψ, t},

    {mSmall, mLarge, Id, Icg, Io} =
     findMassesAndMomentsOfInertia[rSmall, hSmall, ρSmall, rLarge, hLarge, ρLarge, len];

    (* find the solution using numerical solver*)
    (*Find kinetic and potential energy and then the Lagrangian*)
    kinetic = 1/2 Id ϕ'[t]^2 + 1/2 mSmall ( (len Sin[θ[t]] ϕ'[t])^2 + (len θ'[t])^2 ) +
      1/2 Icg[[3, 3]] (ψ'[t] + ϕ'[t] Cos[θ[t]])^2 + 1/2 Icg[[2, 2]] (ϕ'[t] Sin[θ[t]])^2 + 1/2 Icg[[1, 1]] θ'[t]^2;

    v = len (1 -  Cos[θ[t]]) mSmall g;
    lagrangian = kinetic - v;
```

```
  (*write down the 3 equations of motion using the above Lagrangian*)
  (*no generalized forces, life is simple *)

  eqs = Apply[D[ D[lagrangian, #1], t] - D[lagrangian, #2] == 0 & ,
    {{θ'[t], θ[t]}, {ψ'[t], ψ[t]}, {ϕ'[t], ϕ[t]}}, 1];

  (*solve using NDSolve with the initial conditions from the user*)
  initialConditions = {θ[0] == θ0 * Pi / 180, θ'[0] == θ0Speed, ψ[0] == ψ0 * Pi / 180,
    ψ'[0] == ψ0Speed, ϕ[0] == ϕ0 * Pi / 180, ϕ'[0] == ϕ0Speed};

  sol = First@NDSolve[Flatten@{eqs, initialConditions}, {θ, ϕ, ψ},
     {t, 0, maxSimulationTime}, MaxSteps → Infinity, PrecisionGoal → 7];

  isSolutionChanged = True;

  {θ /. sol, ϕ /. sol, ψ /. sol}

 ];

(*---------------------------------------------------*)
(* called before numerically solving the system       *)
(* to calculates masses and moments of inertia        *)
(*---------------------------------------------------*)
findMassesAndMomentsOfInertia[rSmall_, hSmall_, ρSmall_, rLarge_, hLarge_, ρLarge_, len_] :=
 Module[{mSmall, mLarge, Id, Icg, Io, Icg1, Icg2, Icg3, Io1, Io2, Io3},

  (*calculate mass of small and large wheel*)
  mSmall = (Pi rSmall^2) hSmall ρSmall;
  mLarge = (Pi rLarge^2) hLarge ρLarge;

  (* moments of inertia of table around its z-axis*)
  Id = (mLarge rLarge^2)/2;

  Icg1 = 1/12 mSmall (3 rSmall^2 + hSmall^2); (*Ix*)
  Icg2 = Icg1; (*Iy*)
  Icg3 = (mSmall rSmall^2)/2; (*Iz*)

  (*apply parallel axis theorem to find I with reference to point o. Point o*)
  (*point o is where the rod of the pendulum is attached to the frame*)
  Io1 = Icg1 + mSmall len^2;
  Io2 = Io1;
  Io3 = Icg3;

  Icg = {{Icg1, 0, 0}, {0, Icg2, 0}, {0, 0, Icg3}};
  Io = {{Io1, 0, 0}, {0, Io2, 0}, {0, 0, Io3}};

  {mSmall, mLarge, Id, Icg, Io}
 ];

(*---------------------------------------*)
(* Generate title grid                    *)
(*---------------------------------------*)
generateTitle[currentθ_, currentϕ_, currentψ_, currentθDer_, ϕDer_, ψDer_, len_, Id_, Icg_, mSmall_] :=
 Module[{currentKE, currentPE, title, totalEnergy, currentKEAsPercentage,
   currentPEAsPercentage, currentKEformattedAsPercentage, currentPEformattedAsPercentage,
   currentKEformattedAsPercentageV1, currentPEformattedAsPercentageV1, g = 9.8},
```



```
currentKE = 1/2 Id ϕDer^2 + 1/2 mSmall (  (len Sin[currentθ] ϕDer)^2 + (len currentθDer)^2 ) + 1/2 Icg⟦3, 3⟧
    (ψDer + ϕDer Cos[currentθ])^2 + 1/2 Icg⟦2, 2⟧ (ϕDer Sin[currentθ])^2 + 1/2 Icg⟦1, 1⟧ currentθDer^2;

currentPE = len (1 -  Cos[currentθ]) mSmall g;
totalEnergy = currentKE + currentPE;

If[totalEnergy ≤ $MachineEpsilon, (*special case, system at rest*)
 {
  currentKEAsPercentage  = 0;
  currentPEAsPercentage  = 0;
 },
 {
  currentKEAsPercentage  = currentKE / totalEnergy 100;
  currentPEAsPercentage  = currentPE / totalEnergy 100;
 }
];

currentKEformattedAsPercentage = Text@Row[{padIt2[currentKEAsPercentage, {2, 1}], " %"}];
currentPEformattedAsPercentage = Text@Row[{padIt2[currentPEAsPercentage, {2, 1}], " %"}];
currentKEformattedAsPercentageV1 = Text@Row[{padIt2[currentKEAsPercentage, {2, 1}], "%"}];
currentPEformattedAsPercentageV1 = Text@Row[{padIt2[currentPEAsPercentage, {2, 1}], "%"}];

title = Text@Style[Grid[{
     {
      "",
      Text["θ"],
      Text["ψ"],
      Text["ϕ"],
      Text@Row[{Style["P.E.", Blue], " (kJ)"}],
      Text@Row[{Style["K.E.", Red], " (kJ)"}]
     },

     { (*angular positions*)
      Text[Style["position (deg)", 9]],
      padIt2[Mod[currentθ 180. / Pi, 360], {6, 3}],
      padIt2[Mod[currentψ 180. / Pi, 360], {6, 3}],
      padIt2[Mod[currentϕ 180. / Pi, 360], {6, 3}],
      padIt2[currentPE / 1000, {8, 0}],
      padIt2[currentKE / 1000, {8, 0}]
     },

     {(*angular velocities*)
      Text[Style["ω (hz)", 9]],
      padIt1[currentθDer / (2. Pi), {5, 3}],
      padIt1[ψDer / (2. Pi), {5, 3}],
      padIt1[ϕDer / (2. Pi), {5, 3}],
      currentPEformattedAsPercentage,
      currentKEformattedAsPercentage
     }

    }, Frame → All,
    FrameStyle → Gray,
    Spacings → 1,
    ItemSize → {{All, 2 ;; -1} → 6},
    Alignment → Center], 11
  ];

{title, currentKE , currentPE, currentKEformattedAsPercentage,
 currentPEformattedAsPercentage, currentPEAsPercentage , currentKEAsPercentage ,
 currentKEformattedAsPercentageV1, currentPEformattedAsPercentageV1}
```

```
];

(*--------------------------------------*)
(* calculate L and L' with reference to pt    *)
(* which is the point where the pendulum rod  *)
(* is attached to the hanger. Also generate   *)
(* grid table containing formatted information*)
(*--------------------------------------*)
calculateAngularMomentum[pt_, ptcg_, Io_, scaleAmount_, θ_, ϕ_, ϕDer_, ψDer_, θDer_, θDerDer_,
  ψDerDer_ , ϕDerDer_] := Module[{Lf, Lx, Ly, Lz, L, norm, inertiaTableDisplay, LDot, LfDot,
   LxDot, LyDot, LzDot, omegaDotVector, ω, ωVector, ωxComp, ωyComp, ωzComp, θVector, ψVector,
   ϕVector, θVectorAnnotation, ψVectorAnnotation, ϕVectorAnnotation, maxVelocityComponent,
   currentptcg, LAnnotation, LDotAnnotation, ωVectorAnnotation, normL, r0, r1},

  (* find coordinates in inertia space of the cg *)
  r0 = RotationTransform[θ, {1, 0, 0}, pt];
  r1 = RotationTransform[ϕ, {0, 0, 1}, {0, 0, 0}];
  currentptcg = r1[r0[ptcg]];

  (* resolve the angular veclocity of the bob along the 3 principal axis*)
  ω = {θDer, ϕDer Sin[θ], ψDer + ϕDer Cos[θ]};

  (* resolve the rate of angular veclocity of the bob along the 3 principal axis*)
  (* by taking derivative w.r.t time of the omega vector, this is the angular acceleration *)
  omegaDotVector = {
    θDerDer,
    ϕDerDer Sin[θ] + ϕDer Cos[θ] θDer ,
    ψDerDer + ϕDerDer Cos[θ] - ϕDer Sin[θ] θDer};

  (* find the angular momentum relative to fixed point 0, this is the fixed point in space*)
  (* that the bob is attached to*)

  L = Chop[Io.ω, 10^-6];

  (* find the rate of angular momentum relative to fixed point 0*)
  LDot = Io.omegaDotVector;

  (*due to rotation of table, to find ABSOLUTE rate of angular momentum      *)
  (*we need to use (d/dt A)_absolute = (d/dt A)_xyz + cross[ϕ,A] formula        *)
  (*where in this case A is L, and the LHS above is absolute rate of change  *)
  (*note: ϕDer is used below, since L is on the fixed point 0, which rotates *)
  (*by ϕDer relative to the ground *)

  LDot = Chop[LDot + Cross[{0, 0, ϕDer }, L], 10^-5];

  inertiaTableDisplay = Text@Grid[{{
       Grid[{
         {
          Row[{Style["I", Italic, 11], " = "}],

          Style[TraditionalForm[{
              {padIt2[Io[[1, 1]], {9, 1}], 0, 0},
              {0, padIt2[Io[[2, 2]], {9, 1}], 0},
              {0, 0, padIt2[Io[[3, 3]], {9, 1}]}}], 11]
         }}, Spacings → 0, Frame → None, Alignment → Left],

       Grid[{

         {Style["‖ω‖ = ", 11],
          Style[TraditionalForm[padIt2[Norm@ω, {11, 1}]], 10]
         },
```



```
{Row[{Style["‖", 11], Style["L", Italic, 11], Style["‖ = ", 11]}],
 Style[TraditionalForm[padIt2[Norm@L, {11, 1}]], 10]},

{Row[{Style["‖", 11], Style["dL/dt", Italic, 11], Style["‖ = ", 11]}],
 Style[TraditionalForm[padIt2[Norm@LDot, {11, 1}]], 10]}
}, Spacings → 0, Frame → None, Alignment → Left]
},

{
 Grid[{
   {
    Style["ω = ", 11],
    Style[TraditionalForm[padIt1[List@ω, {9, 1}]], 11]
   }}, Spacings → 0, Frame → None, Alignment → Left], SpanFromLeft
},

{
 Grid[{
   {
    Row[{Style["L = I", Italic, 11], Style["ω = ", 11]}],
    Style[TraditionalForm[padIt1[List@L, {10, 1}]], 11]
   }}, Spacings → 0, Frame → None, Alignment → Left], SpanFromLeft
},

{
 Grid[{
   {
    Style["dL/dt = ", Italic, 11],
    Style[TraditionalForm[padIt1[List@LDot, {10, 1}]], 11]
   }}, Spacings → 0, Frame → None, Alignment → Left], SpanFromLeft
}

}, Frame → All, Spacings → 1, Alignment → Left, FrameStyle → Gray
];

(* generate the vector representation of θ',ψ',ϕ' from the cg of the bob *)
maxVelocityComponent = Max[Abs[{θDer, ψDer, ϕDer}]];
If[θDer ≤ $MachineEpsilon, θVector = {0, 0, 0}, θVector = ({θDer, 0, 0}/maxVelocityComponent) 1.5 scaleAmount];
θVectorAnnotation = If[θDer ≤ $MachineEpsilon,
  Null, Text[Style["θ̇", Red, 14], ptcg + θVector + 0.005 Norm[θVector], {0, -1}]];
θVector = {ptcg, ptcg + θVector};
θVector = {Blue, Arrowheads[0.04], Arrow[Tube[θVector, .2]]};

If[ψDer ≤ $MachineEpsilon, ψVector = {0, 0, 0}, ψVector = ({0, 0, ψDer}/maxVelocityComponent) 1.5 scaleAmount];
ψVectorAnnotation = If[ψDer ≤ $MachineEpsilon,
  Null, Text[Style["ψ̇", Red, 14], ptcg + ψVector + 0.005 Norm[ψVector], {0, -1}]];
ψVector = {ptcg, ptcg + ψVector};
ψVector = {Blue, Arrowheads[0.04], Arrow[Tube[ψVector, .2]]};
```



```
If[ϕDer ≤ $MachineEpsilon, ϕVector = {0, 0, 0}, ϕVector = ({0, 0, ϕDer}/maxVelocityComponent) 1.5 scaleAmount];

ϕVectorAnnotation = If[ϕDer ≤ $MachineEpsilon, Null,
  Text[Style["ϕ̇", Red, 14], currentptcg + ϕVector + 0.005 Norm[ϕVector], {0, -1}]];

ϕVector = {currentptcg, currentptcg + ϕVector};
ϕVector = {Blue, Arrowheads[0.04], Arrow[Tube[ϕVector, .2]]};

(* generate the vector ω and its components for the angular velocity*)
norm = Norm[ω];
If[norm ≤ 2 $MachineEpsilon, ω = {0, 0, 0}, ω = (ω/norm) 1.5 scaleAmount];
ωVector = {ptcg, ptcg + ω};
ωVectorAnnotation =
 If[norm ≤ $MachineEpsilon, Null, Text[Style["ω", Red, 15], ptcg + ω + 0.005 Norm[ω], {0, -1}]];

ωVector = {Green, Arrowheads[0.04], Arrow[Tube[ωVector, .4]]};
ωxComp = {Blue, Arrowheads[0.03], Arrow[Tube[{ptcg, ptcg + {ω[[1]], 0, 0} }, .1]]};
ωyComp = {Blue, Arrowheads[0.03], Arrow[Tube[{ptcg, ptcg + {0, ω[[2]], 0} }, .1]]};
ωzComp = {Blue, Arrowheads[0.03], Arrow[Tube[{ptcg, ptcg + {0, 0, ω[[3]]} }, .1]]};

(* generate the vector and its components for the angular momentum L*)
normL = Norm[L];
If[normL ≤ 2 $MachineEpsilon, L = {0, 0, 0}, L = (L/normL) scaleAmount];
Lf = {pt, pt + L};

LAnnotation =
 If[normL ≤ $MachineEpsilon, Null, Text[Style["L", Red, 15], pt + L + 0.005 Norm[Lf], {0, -1}]];

Lf = {Red, Arrowheads[0.04], Arrow[Tube[Lf, .2]]};
Lx = {Blue, Arrowheads[0.03], Arrow[Tube[{pt, pt + {L[[1]], 0, 0} }, .1]]};
Ly = {Blue, Arrowheads[0.03], Arrow[Tube[{pt, pt + {0, L[[2]], 0} }, .1]]};
Lz = {Blue, Arrowheads[0.03], Arrow[Tube[{pt, pt + {0, 0, L[[3]]} }, .1]]};

(* generate the vector and its components for the rate of angular momentum dL/dt*)
norm = Norm[LDot];
If[norm < 10^-6 normL, norm = 0]; (* Force norm to zero. Was due to some numerical errors*)

If[norm ≤ 2 $MachineEpsilon, LDot = {0, 0, 0}, LDot = (LDot/norm) * 0.8 scaleAmount];
LfDot = {pt, pt + LDot};
LDotAnnotation =
 If[norm ≤ $MachineEpsilon, Null, Text[Style["L̇", Black, 15], pt + LDot + 0.005 Norm[LfDot], {0, -1}]];

LfDot = {Black, Arrowheads[0.04], Arrow[Tube[LfDot, .2]]};
LxDot = {Blue, Arrowheads[0.03], Arrow[Tube[{pt, pt + {LDot[[1]], 0, 0} }, .1]]};
LyDot = {Blue, Arrowheads[0.03], Arrow[Tube[{pt, pt + {0, LDot[[2]], 0} }, .1]]};
LzDot = {Blue, Arrowheads[0.03], Arrow[Tube[{pt, pt + {0, 0, LDot[[3]]} }, .1]]};

{inertiaTableDisplay, Lf, Lx, Ly, Lz, LfDot, LxDot, LyDot, LzDot, ωVector,
 ωxComp, ωyComp, ωzComp, θVector, ψVector, ϕVector, θVectorAnnotation, ψVectorAnnotation,
 ϕVectorAnnotation, currentptcg, LDotAnnotation, LAnnotation, ωVectorAnnotation}
];

(*----------------------------------------*)
(* Manage trace buffer                    *)
(*----------------------------------------*)
refeshTraceBuffer[tnow_, isTraceOn_] := Module[{},
```



```
  If[(previousMaxTraceSize ≠ currentMaximumTraceSize ||
     isSolutionChanged == True || Length[traceBuffer] == 0 ),
   {
    isSolutionChanged = False;
    traceBuffer = Table[0, {currentMaximumTraceSize}];
    previousMaxTraceSize = currentMaximumTraceSize;
    isFirstScan = True;
    currentTraceSize = 0
   }
  ];

  If[tnow ≤ $MachineEpsilon || Not[isTraceOn], {currentTraceSize = 0; isFirstScan = True}];

 ];

(*--------------------------------------*)
(* Called by Manipulate main expression  *)
(*--------------------------------------*)
update[len_, ρSmall_, ρLarge_, rSmall_, rLarge_, hSmall_, hLarge_, tnow_, viewPoint_, boxIt_,
  angularMomentumOption_, showI_, zoom_, testCase_, solθ_, solϕ_, solψ_, traceThickness_, isTraceOn_] :=
 Module[{Id, mSmall, title, gr, g1, g2, g3, pt0, pt1, pt2, pt3, pt4, pt5, pt6, pt6a, pt7, pt8, pt9,
   pt10, pt11, pt12, pt13, pt14, z0, gextraCylinderOnTopOfHanger, frameRadius = 0.6, currentKE,
   currentPE, peke, totalScale, currentKEformattedAsPercentage, currentPEformattedAsPercentage,
   currentPEAsPercentage, currentKEAsPercentage, a, b, ghangers1, ghangers2, ghangers3,
   gLargeCylinder, line1, gline1 , gPost , gWheel, gLine2, referencePointX, referencePointY,
   gXYZ, labels, currentKEformattedAsPercentageV1, currentPEformattedAsPercentageV1, g0,
   inertiaTableDisplay, LfDot, LxDot, LyDot, LzDot, Lf, Lx, Ly, Lz, imageSize, opacity, Io, Icg,
   ωVector, ωxComp,  ωyComp, ωzComp, θVector, ψVector, ϕVector, θVectorAnnotation, ψVectorAnnotation,
   ϕVectorAnnotation, gextraCylinderOnTopOfHangerSphere, base, currentptcg, p, LDotAnnotation,
   LAnnotation, ωVectorAnnotation, θDer, ψDer, ϕDer, θDerDer, ψDerDer, ϕDerDer, mLarge, θ, ψ, ϕ, t},

  refeshTraceBuffer[tnow, isTraceOn];

  (*this value is the largest vertical value for the overall 3D image. Will use as *)
  (*measuring stick for zooming action and other layout to measure things against*)

  totalScale = 3.2 lPost + hLarge + len + hSmall + rSmall;

  (* The masses and moments of inertia are now calculated from user input parameters *)
  {mSmall, mLarge, Id, Icg, Io} =
   findMassesAndMomentsOfInertia[rSmall, hSmall, ρSmall, rLarge, hLarge, ρLarge, len];

  (* Use the solution passed in, which was allread found *)
  (* Evaluate the solution are the current time*)

  θ = Chop@solθ[tnow];
  ϕ = Chop@solϕ[tnow];
  ψ = Chop@solψ[tnow];
  θDer = Chop@(solθ'[t] /. t -> tnow);
  ψDer = Chop@(solψ'[t] /. t -> tnow);
  ϕDer = Chop@(solϕ'[t] /. t -> tnow);
  θDerDer = Chop@(solθ''[t] /. t -> tnow);
  ψDerDer = Chop@(solψ''[t] /. t -> tnow);
  ϕDerDer = Chop@(solϕ''[t] /. t -> tnow);

  {title, currentKE , currentPE, currentKEformattedAsPercentage, currentPEformattedAsPercentage,
    currentPEAsPercentage , currentKEAsPercentage , currentKEformattedAsPercentageV1,
    currentPEformattedAsPercentageV1} = generateTitle[θ, ϕ, ψ, θDer, ϕDer, ψDer , len, Id, Icg, mSmall];

  (*set the coodinates of the main points to use to draw the 3D graphics*)
  (*these are the coordinates of main markers in the system as things look at*)
  (*rest and all initial conditions are zero*)
```

```
z0 = lPost + hLarge + 2 lPost;
pt0 = {0, 0, 0}; pt1 = {0, 0, lPost}; pt2 = {0, 0, lPost + hLarge};
pt3 = {0, 0, z0 - len - hSmall}; pt4 = {0, 0, z0 - len}; pt5 = {0, 0, z0}; pt6 = {0.95 rLarge, 0, lPost};
pt6a = {rLarge, 0, lPost + hLarge}; pt7 = {0.95 rLarge, 0, z0}; pt8 = {-0.95 rLarge, 0, z0};
pt9 = {- 0.95 rLarge, 0, lPost}; pt10 = {rLarge, 0, lPost}; pt11 = {rSmall, 0, z0 - len};
pt12 = {rSmall, 0, z0 - len - hSmall}; pt13 = {0.1 rLarge, 0, z0}; pt14 = {-0.1 rLarge, 0, z0};

(*only calculate angular momentum L if needed to display*)
If[(Not[angularMomentumOption == 0] || showI || isTraceOn),
 {inertiaTableDisplay, Lf, Lx, Ly, Lz, LfDot, LxDot, LyDot, LzDot, ωVector, ωxComp, ωyComp, ωzComp,
    θVector, ψVector, ϕVector, θVectorAnnotation, ψVectorAnnotation, ϕVectorAnnotation,
    currentptcg, LDotAnnotation, LAnnotation, ωVectorAnnotation} = calculateAngularMomentum[
    pt5, pt4, Io, 2 zoom rSmall, θ, ϕ, ϕDer, ψDer, θDer, θDerDer, ψDerDer , ϕDerDer];
];


If[isTraceOn,
 {
  If[++currentTraceSize > currentMaximumTraceSize, {isFirstScan = False; currentTraceSize = 1}];
  If[DEBUG, Print["isTraceOn True, updated currentTraceSize now=", currentTraceSize]];
  traceBuffer[[currentTraceSize]] = currentptcg;
  If[DEBUG, Print["isFirstScan=", isFirstScan]];
 }
];

(* start making the 3D graphics                    *)
(*make the main post which the large table sits on*)
base = {RGBColor[.1, .8, .8], Cylinder[{pt0, pt0 + {0, 0, rPost}}, 5 rPost]};
gPost = {base, Cylinder[{pt0, pt2}, rPost]};

(*make the large table*)
gLargeCylinder = {Opacity[.8], Cylinder[{pt1, pt2}, rLarge]};

(*line drawn on top of table*)
line1 = {Thickness[0], Red, Line[{pt2, pt6a, pt10}]};

(* draw the hanger to attach the pendulum on*)
opacity = 1;

ghangers1 = {Opacity[.8], Cylinder[{pt6, pt7}, frameRadius]};
ghangers2 =
 {Opacity[opacity], Cylinder[{pt7 + {0.05 pt7[[1]], 0, 0}, pt8 - {0.05 pt8[[1]], 0, 0}}, frameRadius]};
ghangers3 = {Opacity[.8], Cylinder[{pt8, pt9}, frameRadius]};

(*make the little extra pump to show where the pendulum hangs*)
gextraCylinderOnTopOfHanger = {Opacity[opacity], Cylinder[{pt13, pt14}, 3 frameRadius]};

(*make the end small balls at the connection of the frame joints*)
gextraCylinderOnTopOfHangerSphere =
 {Red, Opacity[1], {Sphere[pt7, 2 frameRadius], Sphere[pt8, 2 frameRadius]}};

(*draw the pendulum rod itself*)
gline1 = Cylinder[{pt4, pt5}, .4];

(*draw the pendulum bob, which is a cylinder in this case*)
gWheel = {Yellow, Opacity[.8], Cylinder[{pt3, pt4}, rSmall]};

(*red line on top of the above, to make it easy to see it spining*)
gLine2 = If[angularMomentumOption == 0,
  {Thickness[.01], Red, Line[{pt4, pt11, pt12 }]},
  If[angularMomentumOption == 1 ||
    angularMomentumOption == 2 || angularMomentumOption == 3 || angularMomentumOption == 4 ||
    angularMomentumOption == 9 || angularMomentumOption == 10 || angularMomentumOption == 11,
```

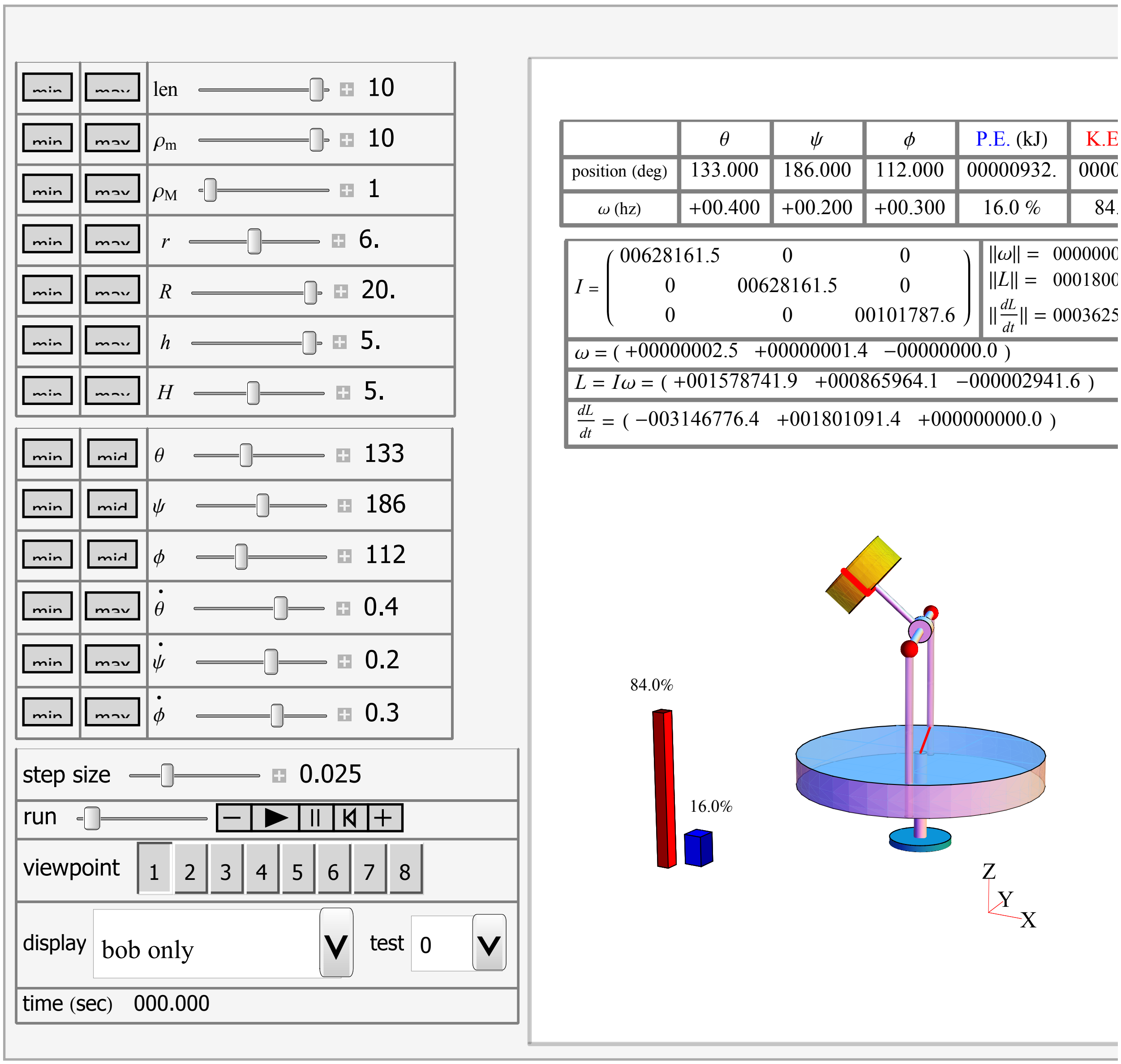


```
    {Thickness[.01], Green, Line[{pt4, pt11, pt12 }]}, Null]
 ];

(*--  Now start applying the solution to rotate items    -*)
(*--  Use Rotate and GeometricTransformation with        -*)
(*--  the RotationTransform based on angles found from   -*)
(*--  the numerical solution above                       -*)

(* start by rotating the pendulum bob itself on its z axis*)
g0 = Rotate[{gLine2, gWheel}, ψ, {0, 0, 1}, pt3];

(* check what display to show.*)
g1 = If[Not[angularMomentumOption == 0],
   Which[
    angularMomentumOption == 1, {Rotate[{Lf}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 2, {Rotate[{Lx, Ly, Lz}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 3, {Rotate[{Lf, Lx, Ly, Lz}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 4, {Rotate[{LfDot}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 5, {Rotate[{LxDot, LyDot, LzDot}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 6, {Rotate[{ωVector}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 7, {Rotate[{ωxComp, ωyComp, ωzComp}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 8,
    {Rotate[{Lf, LAnnotation, ωVector, ωVectorAnnotation}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 9, {Rotate[{ψVector, ψVectorAnnotation}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 10, {Rotate[{LfDot, Lf, LDotAnnotation, LAnnotation}, ψ, {0, 0, 1}, pt5]},
    angularMomentumOption == 11,
    {Rotate[{LfDot, Lf, LDotAnnotation, LAnnotation, ωVector, ωVectorAnnotation}, ψ, {0, 0, 1}, pt5]}
   ],
   Null
 ];

(*transform the whole pendulum with its rod by theta*)
g2 = GeometricTransformation[{gline1, g0, g1, If[angularMomentumOption == 9,
     {θVector, θVectorAnnotation}]}, RotationTransform[θ, {1, 0, 0}, pt5]];

(*now rotate everything by ϕ, the table rotation angle*)
g3 = Rotate[{gLargeCylinder, line1, ghangers1 , ghangers2 , ghangers3,
    gextraCylinderOnTopOfHanger, gextraCylinderOnTopOfHangerSphere, g2 }, ϕ, {0, 0, 1}, pt1];

(*Now check which test case is run. Some test cases does not need PE/KE display*)
If[zoom == 1 && Not[testCase == 3 || testCase == 1 || testCase == 7],
 {
  referencePointX = rLarge + 3 rPost;
  referencePointY = -rLarge - 4 rPost;

  (*make the small XYZ coordinate frame on the side for reference*)
  labels = {Text[Style["X", 12], {referencePointX + 6 rPost, referencePointY, 0}],
    Text[Style["Y", 12], {referencePointX, referencePointY + 6 rPost, 0}],
    Text[Style["Z", 12], {referencePointX, referencePointY, 6 rPost}]};

  gXYZ = {Thin, Red, Line[{
       {referencePointX, referencePointY, 0},
       {referencePointX + 5 rPost, referencePointY, 0},
       {referencePointX, referencePointY, 0},
       {referencePointX, referencePointY + 5 rPost, 0},
       {referencePointX, referencePointY, 0},
       {referencePointX, referencePointY, 5 rPost}}]};

  (*now make PE/KE illustration on the side*)
  a = currentKEAsPercentage / 100 totalScale / 2;
```



```
  b = (currentPEAsPercentage / 100) totalScale / 2;

  referencePointX = -rLarge - 10 rPost;
  referencePointY = -rLarge - 10 rPost;

  (*draw the PE and KE illustrations on the side of the main plot*)
  peke = {
    {Red, Cuboid[{referencePointX, referencePointY, 0},
      {-rLarge - 8 rPost, -rLarge - 8 rPost, (currentKEAsPercentage / 100) totalScale / 2}]
    },

    {Blue, Cuboid[{-rLarge - 7 rPost, -rLarge - 7 rPost, 0},
      {-rLarge - 4 rPost, -rLarge - 4 rPost, (currentPEAsPercentage / 100) totalScale / 2}]
    },
    Text[currentKEformattedAsPercentageV1, {referencePointX, referencePointY, a + 4 rPost}, {0, 0}],
    Text[currentPEformattedAsPercentageV1, {-rLarge - 4 rPost, -rLarge - 4 rPost, b + 4 rPost}, {0, 0}]
   }
 },
 peke = Null;
 gXYZ = Null;
 labels = Null
];

(*-- Done making the graphics parts. now make the final display --*)
imageSize = If[showI, {345, 270}, {345, 480}];

p = If[isTraceOn,
  ListPointPlot3D[If[isFirstScan, traceBuffer[[1 ;; currentTraceSize]],
    traceBuffer[[1 ;; currentMaximumTraceSize]]], PlotStyle → {PointSize[traceThickness], Blue}]
 ];

gr = Graphics3D[
  {gPost, g3, labels, gXYZ, peke, If[angularMomentumOption == 9, {ϕVectorAnnotation, ϕVector}]},
  PlotRange → {
    {-zoom 2 Max[rLarge, len + hSmall] , zoom 1.7 Max[rLarge, len + hSmall]},
    {-zoom 1.9 Max[rLarge, len + hSmall] , zoom 1.4 Max[rLarge, len + hSmall]},
    {If[zoom < 1, lPost, 0], totalScale}},
  ImageSize → imageSize,
  Axes → False,
  Boxed → boxIt,
  AxesOrigin → {0, 0, 0},
  ImageMargins → 2,
  ImagePadding → 2,
  PlotRangePadding → 1,
  ViewPoint -> viewPoint,
  ViewAngle → All
 ];


If[showI,
 Grid[{{title}, {inertiaTableDisplay}, {If[isTraceOn, Show[gr, p], gr]} },
  Spacings → 0,
  Frame → None
 ],
 Grid[{{If[isTraceOn, Show[gr, p], gr]} },
```

```
          Spacings → 0,
          Frame → None
        ]

      ]
    ]

  )
]
```



Caption

Simulation of a pendulum made up of a small spinning rigid cylindrical bob in which the pendulum rod (assumed to have negligible mass) swings attached to a frame fixed on top of a rotating table. The system has 3 degrees of freedom. The angle θ: the pendulum swing angle. The angle ψ: the spin angle of the pendulum bob, and angle ϕ: the rotation angle of the large table.

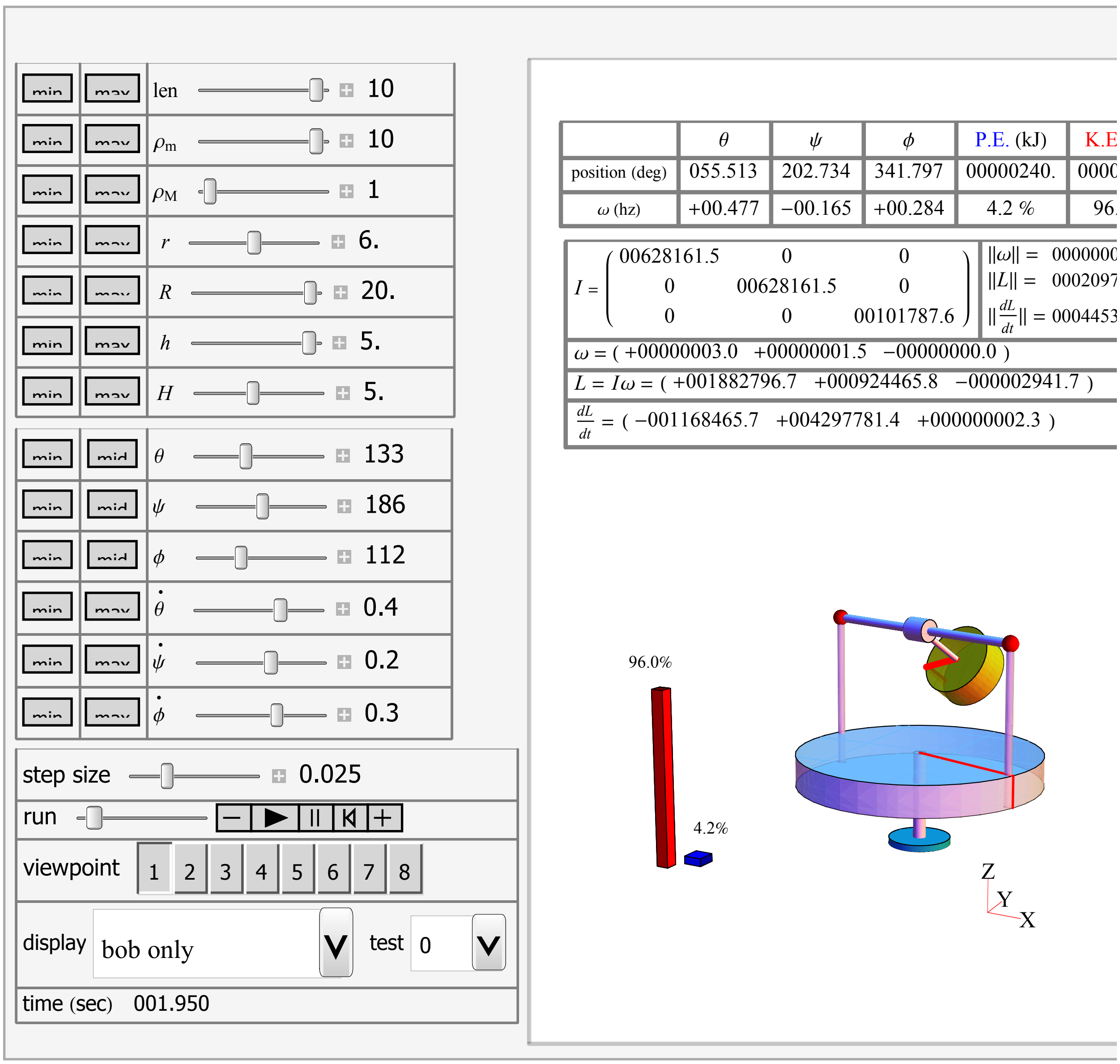
The angular momentum L of the bob with reference to fixed point in space as well as the bob absolute rate of angular momentum $\frac{dL}{dt}$ are calculated and displayed in vector form.

The instantaneous value of the system's kinetic (red bar) and potential energy (blue bar) is illustrated graphically. Other options are available to help study this system in details.

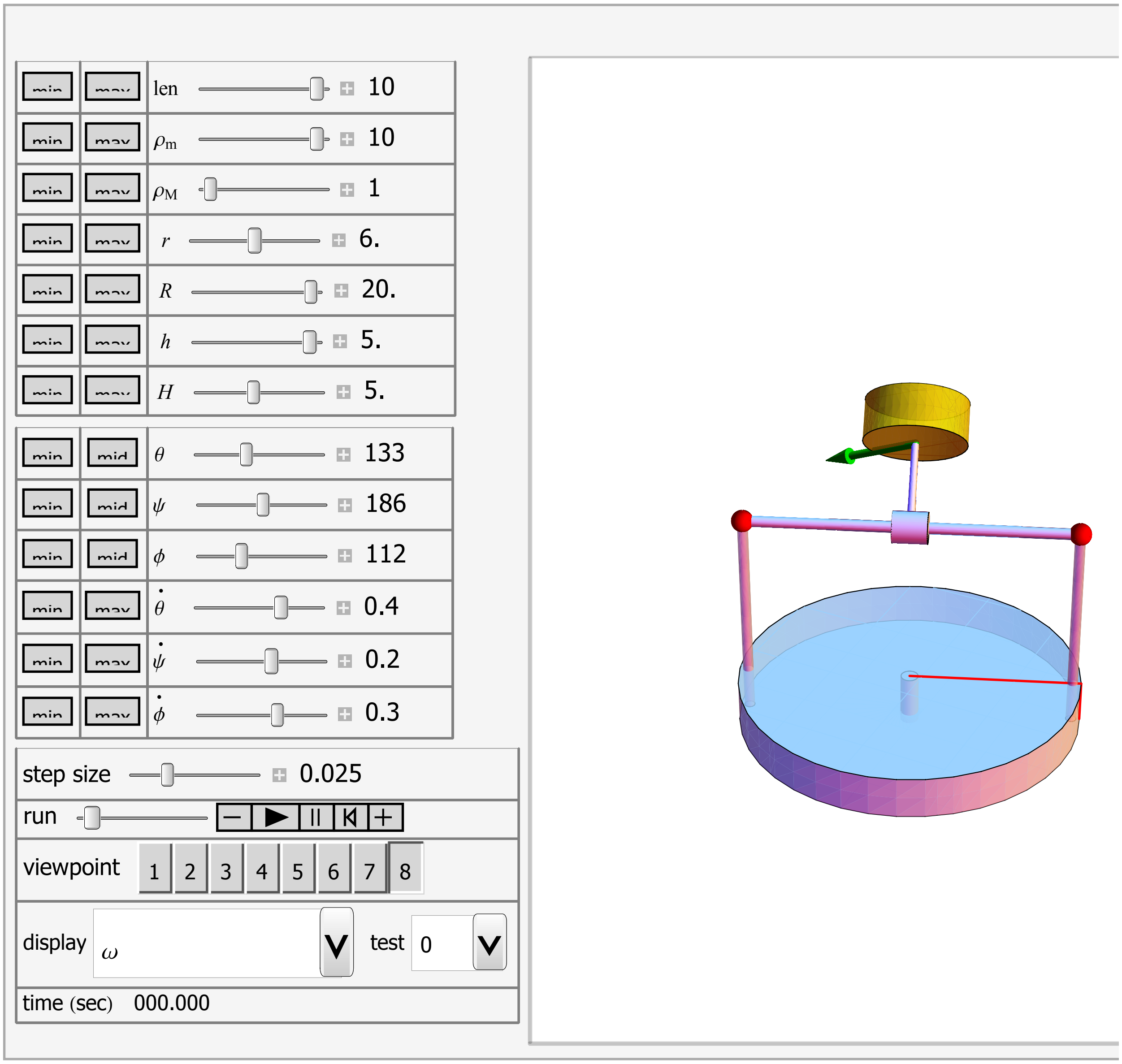
The simulation was performed by finding the Lagrangian and constructing the 3 nonlinear equations of motion and solving them numerically using NDSolve.

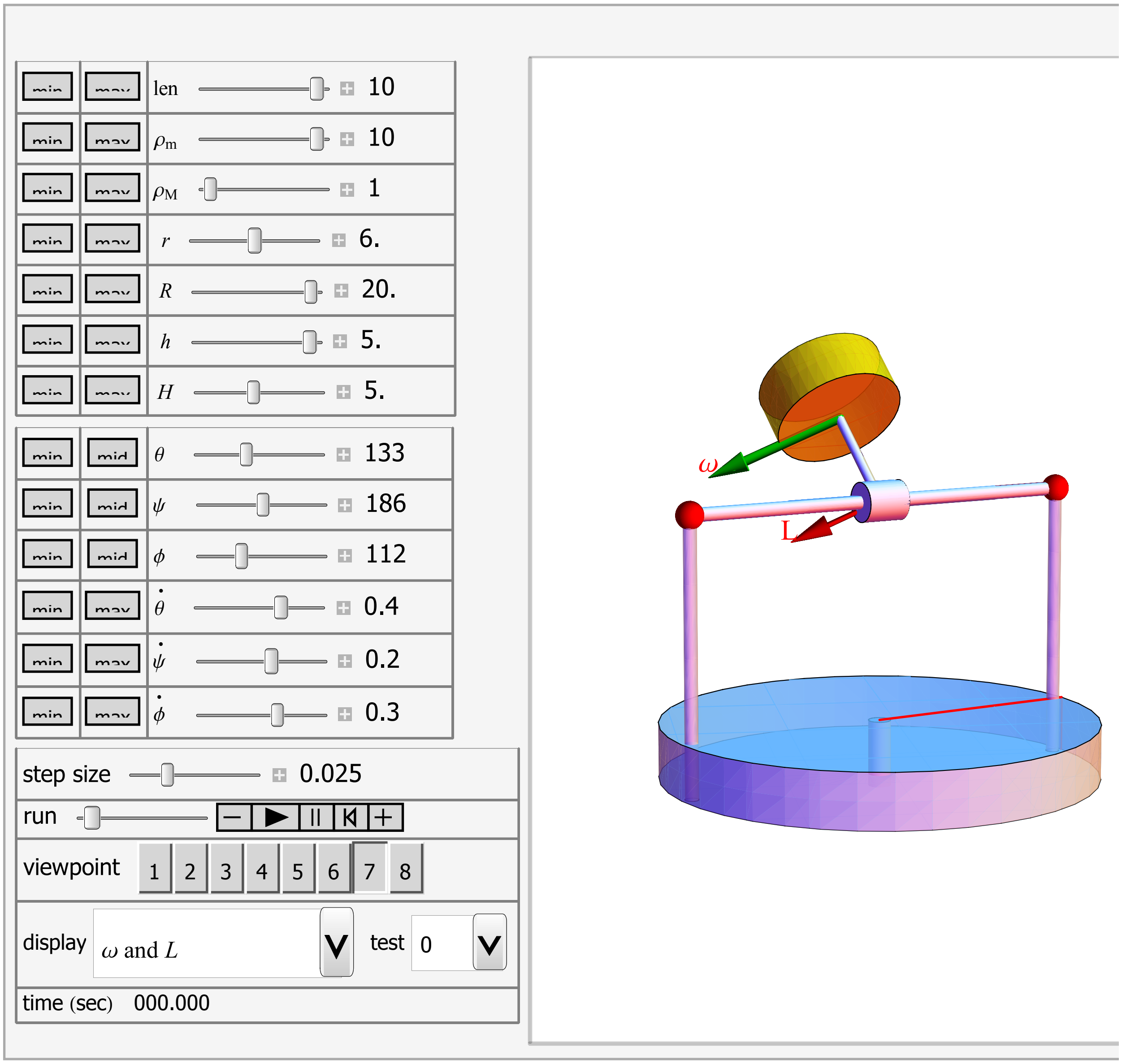
The principal moments of inertia are used for the bob. The fixed point in space that was used to calculate L with respect to is the point where the pendulum rod is attached to the frame.

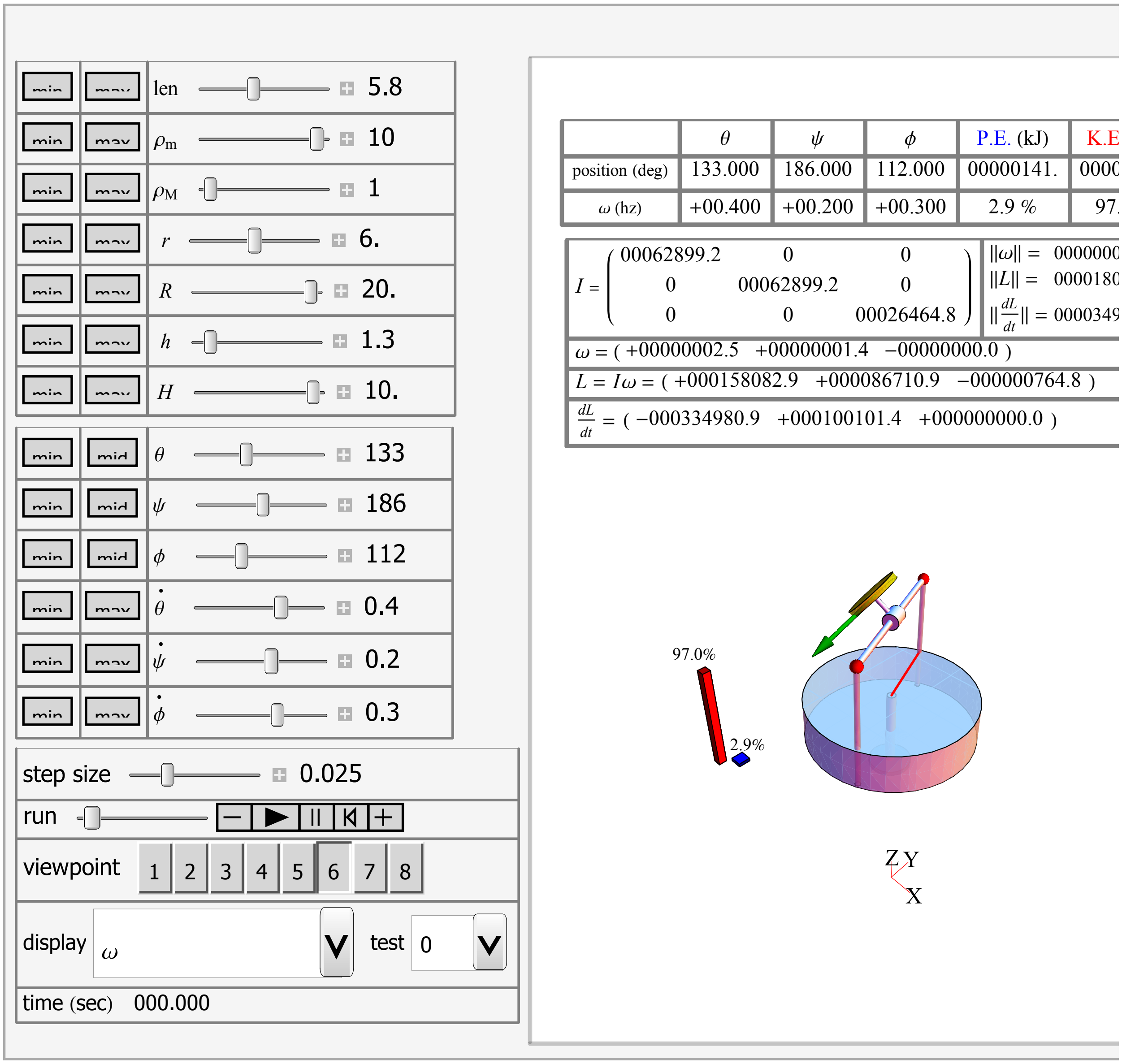
Thumbnail



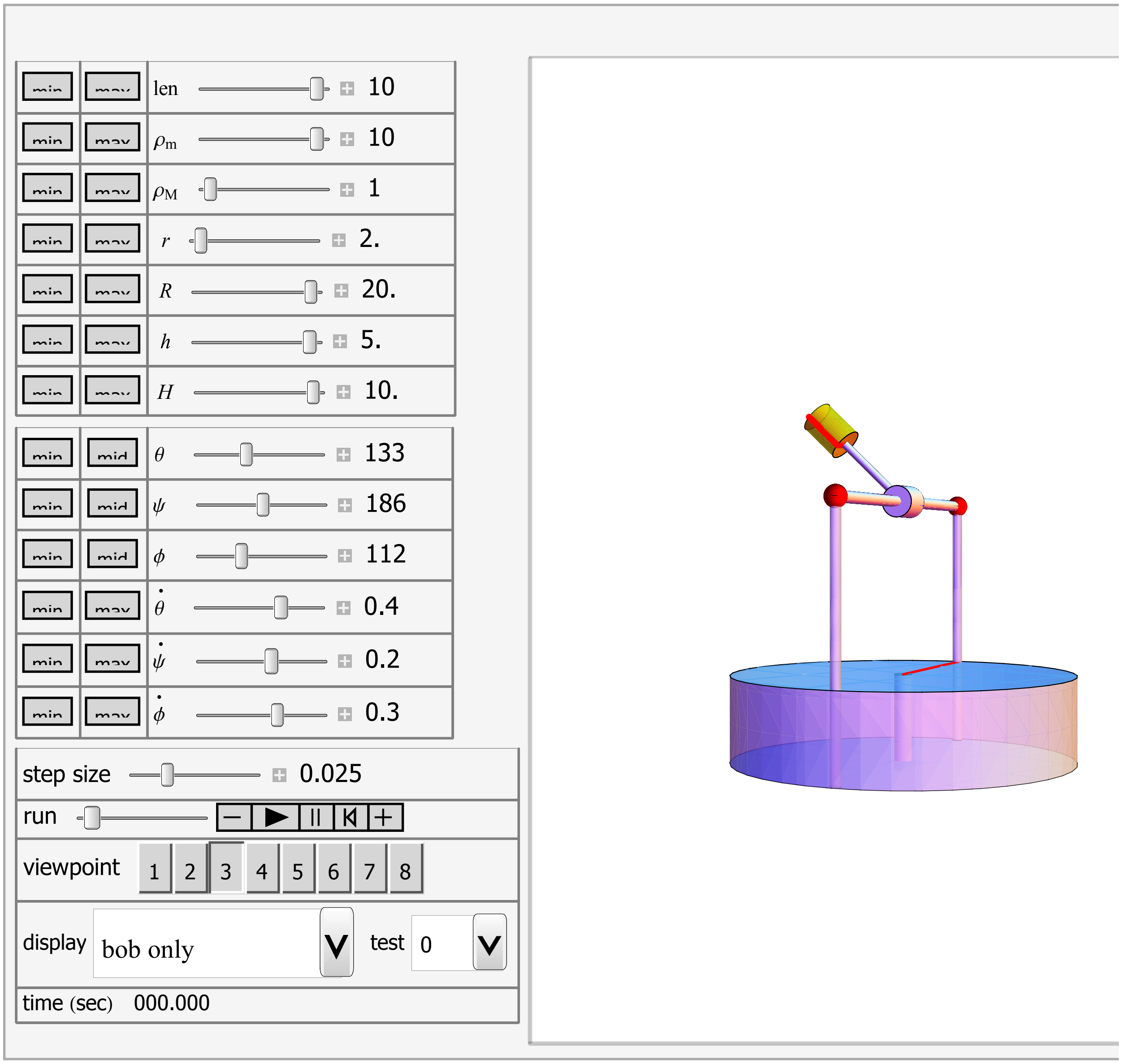
Snapshots

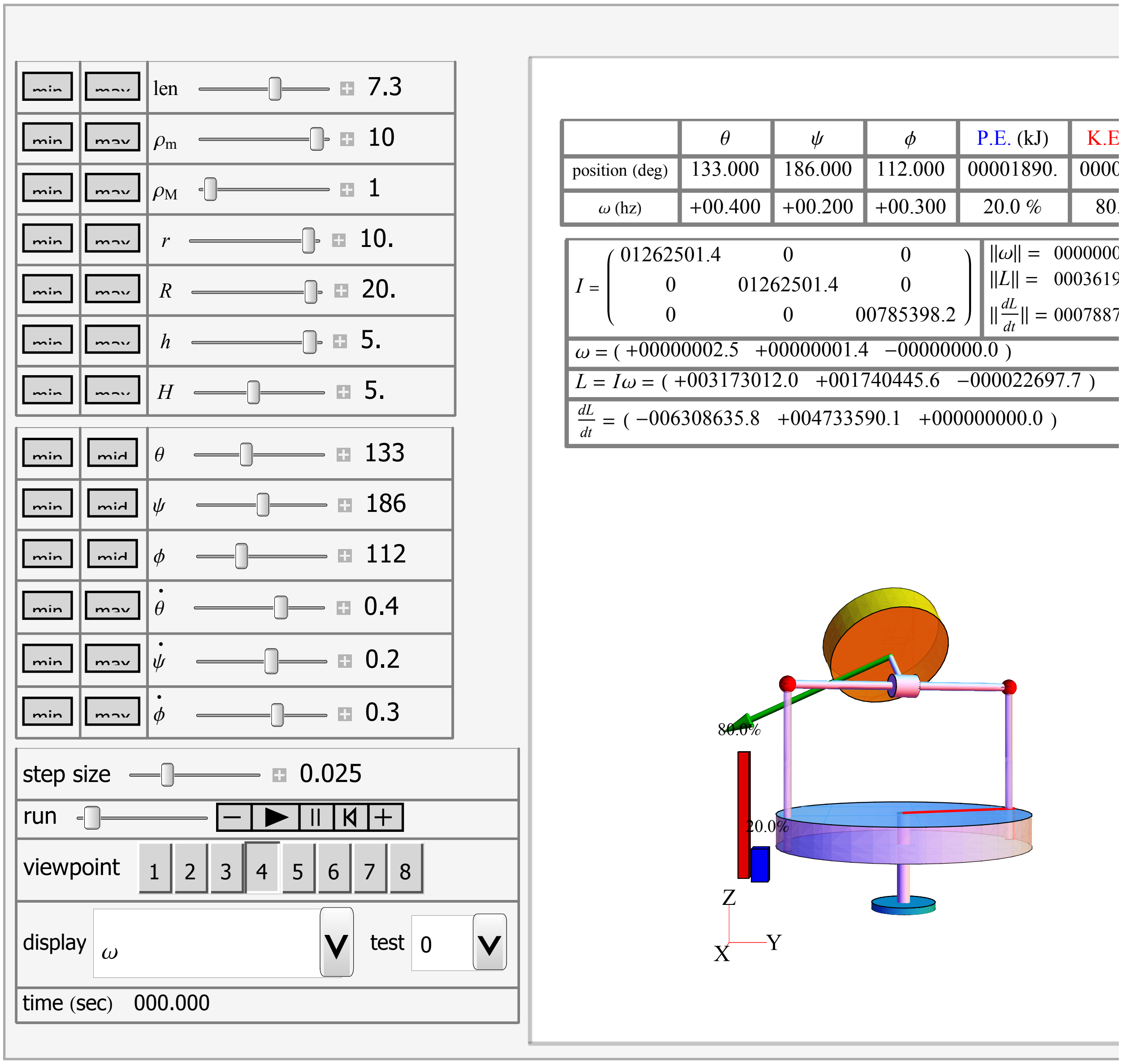

len 10
ρm 10
ρM 1
r 6.
R 20.
h 5.
H 5.
θ 133
ψ 186
ϕ 112
0.4
0.2
0.3
step size 0.025
run
viewpoint 1 2 3 4 5 6 7 8
display ω
test 0
time (sec) 000.000

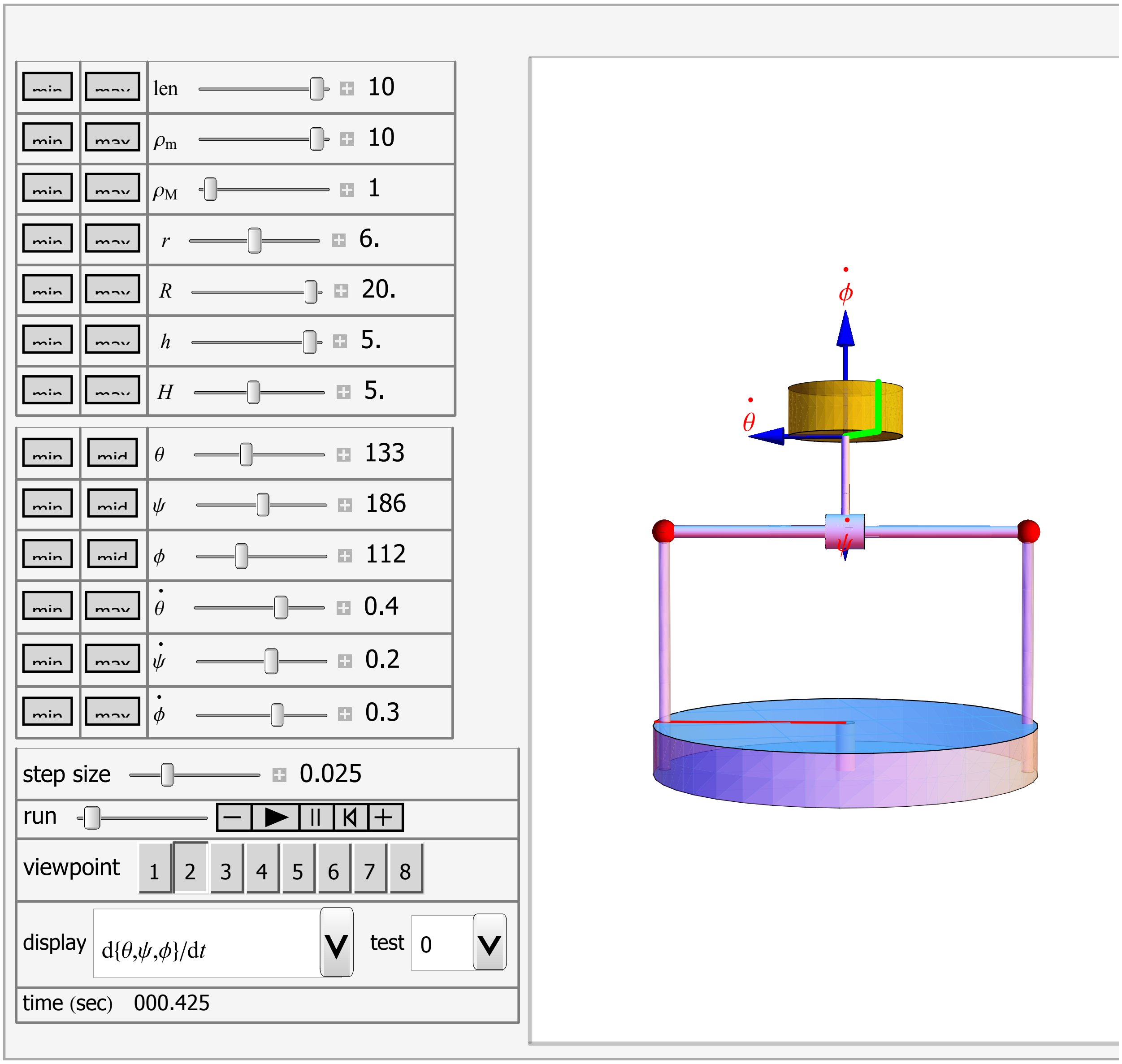

len 10
ρm 10
ρM 1
r 6.
R 20.
h 5.
H 5.
θ 133
ψ 186
ϕ 112
θ̇ 0.4
ψ̇ 0.2
ϕ̇ 0.3
step size 0.025
run
viewpoint 1 2 3 4 5 6 7 8
display ω and L
test 0
time (sec) 000.000
ω
L



min max len 5.8
min max ρm 10
min max ρM 1
min max r 6.
min max R 20.
min max h 1.3
min max H 10.
min mid θ 133
min mid ψ 186
min mid ϕ 112
min max θ̇ 0.4
min max ψ̇ 0.2
min max ϕ̇ 0.3
step size 0.025
run
viewpoint 1 2 3 4 5 6 7 8
display ω
test 0
time (sec) 000.000
θ ψ ϕ P.E. (kJ) K.E
position (deg) 133.000 186.000 112.000 00000141. 0000
ω (hz) +00.400 +00.200 +00.300 2.9 % 97.
I = 00062899.2 0 0 / 0 00062899.2 0 / 0 0 00026464.8
||ω|| = 0000000
||L|| = 0000180
||dL/dt|| = 0000349
ω = (+00000002.5 +00000001.4 −00000000.0)
L = Iω = (+000158082.9 +000086710.9 −000000764.8)
dL/dt = (−000334980.9 +000100101.4 +000000000.0)
97.0%
2.9%
Z Y
X


min
max
mid
len 10
ρm 10
ρM 1
r 2.
R 20.
h 5.
H 10.
θ 133
ψ 186
ϕ 112
θ̇ 0.4
ψ̇ 0.2
ϕ̇ 0.3
step size 0.025
run
viewpoint 1 2 3 4 5 6 7 8
display bob only
test 0
time (sec) 000.000


len 7.3
ρm 10
ρM 1
r 10.
R 20.
h 5.
H 5.
θ 133
ψ 186
ϕ 112
θ̇ 0.4
ψ̇ 0.2
ϕ̇ 0.3
step size 0.025
run
viewpoint 1 2 3 4 5 6 7 8
display ω
test 0
time (sec) 000.000
θ ψ ϕ P.E. (kJ) K.E
position (deg) 133.000 186.000 112.000 00001890. 0000
ω (hz) +00.400 +00.200 +00.300 20.0 % 80.
I = (01262501.4 0 0 / 0 01262501.4 0 / 0 0 00785398.2)
||ω|| = 0000000
||L|| = 0003619
||dL/dt|| = 0007887
ω = (+00000002.5 +00000001.4 −00000000.0)
L = Iω = (+003173012.0 +001740445.6 −000022697.7)
dL/dt = (−006308635.8 +004733590.1 +000000000.0)
80.0%
20.0%
Z
X
Y



Details

(optional)

The kinetic energy of the system is given by $T = \frac{1}{2} I_d \dot{\phi}^2 + \frac{1}{2} m\left(\left(L \sin(\theta)\, \dot{\phi}\right)^2 + \left(L\, \dot{\theta}\right)^2\right) + \frac{1}{2} I_3\left(\dot{\psi} + \dot{\phi} \cos(\theta)\right)^2 + \frac{1}{2} I_2\left(\dot{\phi} \sin(\theta)\right)^2 + \frac{1}{2} I_1\, \dot{\theta}^2$ and the potential energy is given by $V = L(1 - \cos(\theta))\, m\, g$ where $I_d = \frac{M R^2}{2}$ is the moment of inertia of the large table and M is its mass and R is its radius. I_1, I_2, I_3 are the moments of inertia of the bob around its principal axis and due to symmetry $I_1 = I_2 = \frac{1}{12} m\left(3 r^2 + h^2\right)$ and $I_3 = \frac{m r^2}{2}$ where r is the radius of the bob, h is the height of the bob cylinder, and m is its mass. Using parallel axis theorem, the moment of inertia tensor I_o for the bob is found relative to the point where the pendulum rod is attached to the frame. The angular momentum vector L is found relative to this point and not relative to the center of mass of the bob.

The absolute rate of change of the angular momentum vector $\dot{L}_{\text{absolute}}$ is found using $\dot{L}_{\text{absolute}} = \dot{L} + \omega \times L$ where $\dot{L}$ is the rate of change of L relative to the fixed point described above. All terms above are vectors, and the product above is a vector cross product. You can see these vectors and their components visually displayed by selecting a display option from the display popup menu. A special case occurs when the bob is spinning around any one of its principal axes. In this case you will see that L and $\dot{L}$ vectors will always be parallel to each others.

The following is a description of the items on the top half of the left panel of the UI: len is the length of the pendulum rod, ρ_m is the density of the material of the bob cylinder, ρ_M is the density of the material of the large table cylinder, r is the radius of the bob cylinder, R is the radius of the large table cylinder, h is the height of the bob cylinder and H is the height of the large table cylinder.
The panel below that contains the initial conditions for the three rotation angles: Angle θ, which is the pendulum rod swing angle, angle ψ, which is the spin angle of the pendulum bob around its own axis, and angle ϕ which is the spin angle of the large table. The units for the angles are in degree with a range of zero to 360^0. The units for the angular velocities are in hz in the range of -1 hz to $+1$ hz. For convenience, small buttons are placed next to the slider to use to set the an initial condition to its minimum or maximum value or to the middle range value.
The slider labeled "step size" can be used to adjust the animation rate. The smaller the step size, the more accurate the simulation will be, but the longer and slower it will run. You can control the simulation by using the trigger options allowing you to make one step forward or one step backward or to run the simulation to the end or pause it at any time. Before starting new simulation you can reset the trigger.
The popup menu labeled "display" allows you to select which vectors to view while the simulation is running. You can select the angular momentum L or the rate of the angular momentum $\frac{dL}{dt}$of the bob. Or select to view the angular velocity vector of the center of gravity of the bob which is given by $\omega = \left\{\dot{\theta}, \dot{\phi}\sin(\theta), \dot{\psi} + \dot{\phi}\cos(\theta)\right\}$.
A number of test cases are available to choose from. When you select a test case, the control variables will be automatically set to preset values. Then you can click the run button to see the selected test case simulation using those values. When the simulation is running a specific test case, you will not be able adjust any of the controls on the UI other than changing to a different test case. Selecting the special test case 0 releases the UI allowing you to adjust other UI control variables.
If you click on the "info" checkbox on the right side of the display, the following information is displayed while the simulation is running: The moment of inertia tensor I_o, the angular momentum vector L and the rate of change of the angular momentum vector $\frac{dL}{dt}$, and the current values of the 3 angles $\{\theta, \psi, \phi\}$ and the 3 angular velocities $\left\{\dot{\theta}, \dot{\psi}, \dot{\phi}\right\}$. The current value of the system kinetic energy (K.E.) and potential energy (P.E.) are given in units of Kilo Jules (KJ).
The current time in seconds is shown at the lower side of the left part of the display. A zoom slider can be used to zoom into the display while it is running. This provides limited zooming capability. You can use the build-in *Mathematica* support for zoom and panning by clicking on the display then using the ctrl-key. See *Mathematica* documentation for more information. If you decide to use the build-in zooming instead of the zoom slider, then you should pause the simulation first to obtain a better control on the zooming action, then you can resume the simulation.
The trace checkbox on the right side allows you to trace the trajectory of the center of mass of the bob as it moves in the 3D inertial space. You can adjust the length of the trace trajectory using the slider below the trace checkbox and the thickness of the trace line. The trace can be cleared at anytime by clicking on the trace checkbox again.
When the rate of the angular momentum of the bob is not zero, there must exist a torque τ to account for this since $\tau = \frac{dL}{dt}$. This torque is not displayed in this version of the demonstration. It is generated due to support reaction from the ground.
Reference: Donald T. Greenwood, Principles of Dynamics, Prentice-Hall, 1965, chapter 8.

Reference: Richard Feynman, The Feynman lectures on physics, Addison-Wesley, 1963, pp 20-1,20-8.

Control Suggestions (optional)

- [ ] Slider Zoom
- [ ] Drag Locators
- [x] Rotate and Zoom in 3D
- [ ] Automatic Animation
- [ ] Gamepad Controls
- [ ] Resize Images
- [ ] Bookmark Animation

Search Terms (optional)

Lagrangian

rigid body pendulum

angular momentum
pendulum
spinning cylinder
parallel axis theorem

Related Links (optional)

Lagrangian
kinetic energy
potential energy
NDsolve

Authoring Information

Contributed by: Nasser M. Abbasi