Spring Mass System On a Rotating Table

Initialization Code (optional)

Manipulate

```
Manipulate[
 gTick;
 Module[{g1, wasHit = False, r0 = 0.2(*bob radius, fixed*), L0 = R0 / 2.0(*relaxed length of the spring*)},

  If[setIC,
   setIC = False;
   If[initialBobX > R0, initialBobX = R0];
   simulationTime = 0;
   list@setIC[
     {initialBobX, initialBobSpeed, initialTheta, initialThetaDot, 2 m0 initialBobSpeed initialThetaDot}];
   isRelaxedSpring = If[Abs[initialBobX - R0 / 2] ≤ $MachineEpsilon, True, False]
  ];

  If[runningState == "RUNNING" || runningState == "STEP",

   wasHit = solver@step[m0, R0, L0, k, M0, delT, wallType];
   (*this logic below so that we do not animate a hit on the edge unless last time it was no hit*)
   If[wasHit,
    If[wasHitLastTime == True,
     wasHit = False,
     wasHitLastTime = True
    ]
    ,
    wasHitLastTime = False
   ];

   simulationTime = Mod[simulationTime + delT, 1000];
   If[runningState == "STEP", runningState = "STOP", gTick += del]
  ];

  (*all done, display the animation on the screen*)
  g1 = display@makeDisplay[simulationTime, plotType,
     m0, M0, R0, r0, wasHit, k, showPlots, showCounters, showCoriolisForce, trace];
  FinishDynamic[];
  g1
 ],
 (*---------- control layout ------------*)
 Grid[{
   {Grid[{{
       Button[Text[Style["run", 12]], {If[Not[plotType == "velocity/acceleration"],
           runningState = "RUNNING"]; gTick += del}, ImageSize -> {55, 35}],
       Button[Text[Style["stop", 12]], {If[Not[plotType == "velocity/acceleration"],
           runningState = "STOP"]; gTick += del}, ImageSize -> {55, 35}]
      },
      {
       Button[Text[Style["step", 12]], {If[Not[plotType == "velocity/acceleration"], runningState = "STEP"];
         gTick += del}, ImageSize -> {55, 35}],

       Button[Text[Style["reset", 12]], (*bring simulation back to initial conditions*)
        {
```

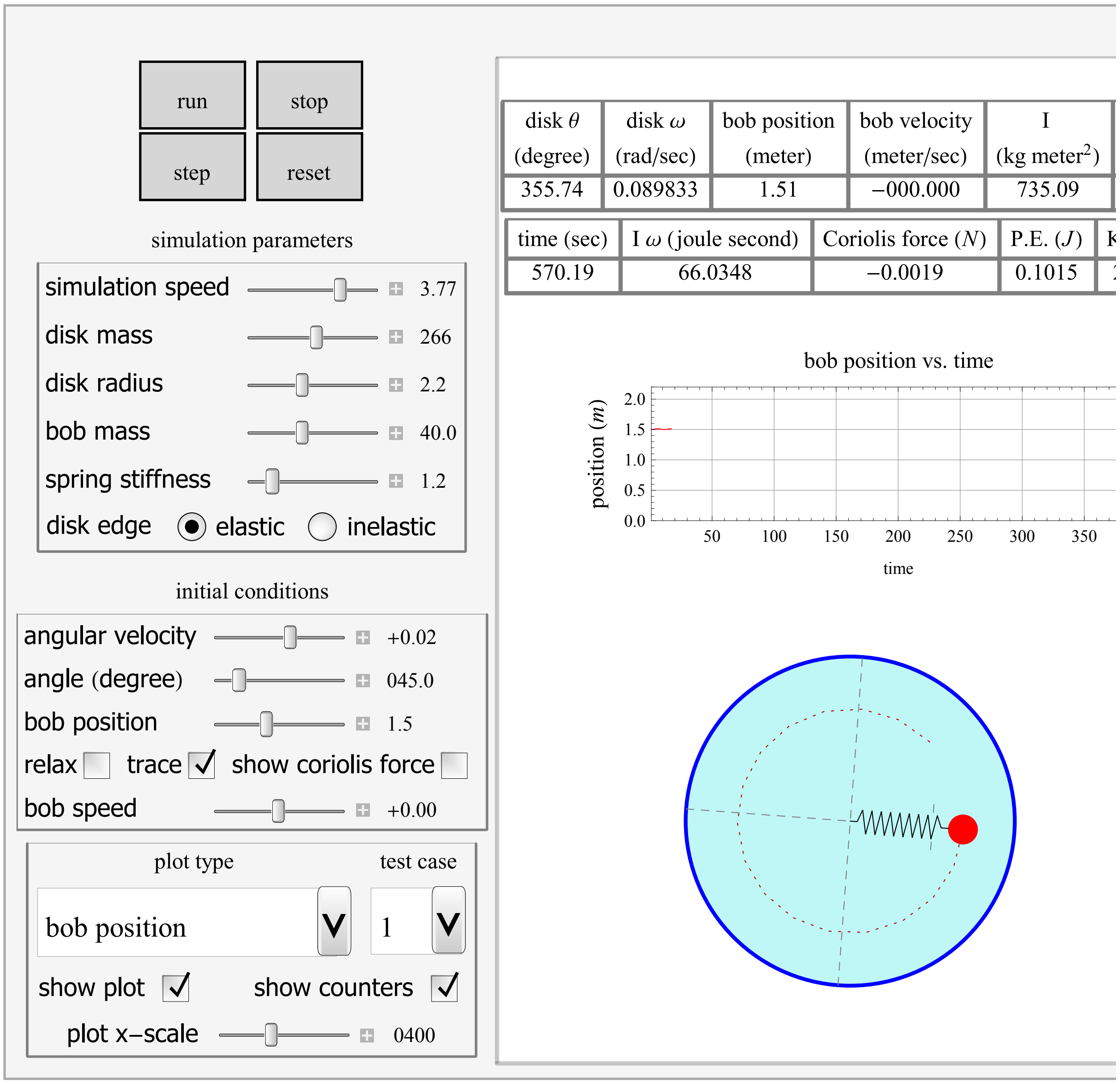
```
        setIC = True;
        runningState = "STOP";
        gTick += del
       }, ImageSize -> {55, 35}]
    }}, Spacings → {0.5, .2}
  ]},

 {Text@Style["simulation parameters", 12]},
 {Grid[
   {
    {"simulation speed",
     Manipulator[Dynamic[delT, {delT = #} &],
      {0.01, 5, 0.01}, ImageSize → Tiny, ContinuousAction -> False],
     Text@Style[Dynamic@padIt2[delT, {3, 2}], 11]
    },

    {"disk mass",
     Manipulator[Dynamic[M0,
       {M0 = #;
         setIC = True;
         gTick += del
        } &], {1, 500, 1}, ImageSize -> Tiny, ContinuousAction -> False],
     Text@Style[Dynamic@padIt2[M0, 3], 11]
    },

    {"disk radius",
     Manipulator[Dynamic[R0,
       {R0 = #;
         setIC = True;
         gTick += del
        } &], {1, 4, 0.1}, ImageSize -> Tiny, ContinuousAction → False],
     Text@Style[Dynamic@padIt2[R0, {2, 1}], 11]
    },

    {"bob mass",
     Manipulator[Dynamic[m0,
       {m0 = #;
         setIC = True;
         gTick += del
        } &], {0, 99, 0.1}, ImageSize -> Tiny, ContinuousAction -> False],
     Text@Style[Dynamic@padIt2[m0, {3, 1}], 11]
    },

    {"spring stiffness",
     Manipulator[Dynamic[k,
       {k = #;
         setIC = True;
         gTick += del
        } &], {0.1, 9, .1}, ImageSize -> Tiny, ContinuousAction -> False],
     Text@Style[Dynamic@padIt2[k, {2, 1}], 11]
    },
    {Grid[{{"disk edge",
        RadioButtonBar[Dynamic[wallType,
          {wallType = #;
            setIC = True;
            gTick += del
           } &], {"elastic" -> "elastic", "inelastic" -> "inelastic"}]
       }}], SpanFromLeft
    }
   }, Spacings → {.6, .5}, Alignment → Left, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
  ]
 },

 {Text@Style["initial conditions", 12]},
 {Grid[{
    {"angular velocity",
     Manipulator[Dynamic[initialThetaDot,
        {initialThetaDot = #;
         setIC = True;
         gTick += del
        } &], {-0.1, 0.1, .01}, ImageSize -> Tiny, ContinuousAction -> False],
     Text@Style[Dynamic@padIt1[initialThetaDot, {2, 2}], 11]
    },
    {"angle (degree)",
     Manipulator[Dynamic[initialTheta,
        {initialTheta = #;
         setIC = True;
         gTick += del
        } &], {0, 2 Pi, 2 Pi / 100.}, ImageSize -> Tiny, ContinuousAction -> False],
     Text@Style[Dynamic@padIt2[initialTheta * 180.0 / Pi, {4, 1}], 11]
    },
    {"bob position",
     Manipulator[Dynamic[initialBobX,
        {initialBobX = #;
         setIC = True;
         gTick += del
        } &], {0, 4, 0.1}, ImageSize -> Tiny, ContinuousAction → False],
     Text@Style[Dynamic@padIt2[initialBobX, {2, 1}], 11]
    },
    {
     Grid[{{
        Row[{"relax", Spacer[2],
       Checkbox[Dynamic[isRelaxedSpring,
         {isRelaxedSpring = #;
           initialBobX = R0 / 2;
           setIC = True;
           gTick += del
          } &]]}],
        Row[{"trace", Spacer[2],
       Checkbox[Dynamic[trace,
         {trace = #;
           If[#, list@clearTrace[]];
           gTick += del
          } &]]}],
        Row[{"show coriolis force", Spacer[2],
       Checkbox[Dynamic[showCoriolisForce,
         {showCoriolisForce = #;
           gTick += del
          } &]]}]
         }}, Spacings → {0.4, 0}], SpanFromLeft
    },
    {"bob speed",
     Manipulator[Dynamic[initialBobSpeed,
        {initialBobSpeed = #;
         setIC = True;
         gTick += del
        } &], {-0.1, 0.1, 0.01}, ImageSize -> Tiny, ContinuousAction -> False],
     Text@Style[Dynamic@padIt1[initialBobSpeed, {2, 2}], 11]
    }
   }, Spacings → {.6, .3}, Alignment → Left, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
  ]
 },
 {Grid[{
    {Text@Style["plot type", 12], Text@Style["test case", 12]},
    {
     PopupMenu[Dynamic[plotType, {plotType = #;
         If[plotType == "velocity/acceleration",
        {
         If[runningState == "RUNNING",
          runningState = "SUSPEND_RUNNING"
         ]
        },
        {
         If[runningState == "SUSPEND_RUNNING", runningState = "RUNNING"]
        }
       ]; gTick += del} &],
  { "disk angular velocity" → Text@Style["disk angular velocity", 12],
    "disk angle" → Text@Style["disk angle", 12],
    "bob position" → Text@Style["bob position", 12],
    "bob velocity" → Text@Style["bob velocity", 12],
   "Coriolis force" → Text@Style["Coriolis force", 12]
  }, ImageSize -> All, ContinuousAction -> False, Enabled → Dynamic[showPlots]]
 ,
 PopupMenu[Dynamic[testCase, {testCase = #;
     runningState = "STOP";
     Which[testCase == 1,
      (
       isRelaxedSpring = False; delT = 1; m0 = 40; k = 1; R0 = 3.9;
       initialThetaDot = 0.09; initialTheta = 206 Degree;
       initialBobX = 0.5; initialBobSpeed = 0; M0 = 10; trace = True;
       setIC = True;
       runningState = "STOP";
       plotType =  "bob position"
      ),
      testCase == 2,
      (
       isRelaxedSpring = False; delT = 5; m0 = 16; k = 3; R0 = 2;
       initialThetaDot = 0.01; initialTheta = 45 Degree;
       initialBobX = 1.4; initialBobSpeed = 0; M0 = 100; trace = True;
       setIC = True;
       runningState = "STOP";
       plotType = "disk angular velocity"
      ),
      testCase == 3,
      (
       isRelaxedSpring = True; delT = 5; m0 = 50; k = 0.5; M0 = 100; R0 = 3; trace = False;
       initialThetaDot = 0; initialTheta = 270 Degree; initialBobX = 1.5; initialBobSpeed = 0.1;
       setIC = True;
       runningState = "STOP";
       plotType = "disk angular velocity"
      )
     ];
     gTick += del} &],
  {
   1 → Text@Style["1", 12], 2 → Text@Style["2", 12], 3 → Text@Style["3", 12]
  }, ImageSize -> All, ContinuousAction -> False]
},

{Grid[{
   {
    Row[{"show plot ", Spacer[4],
      Checkbox[Dynamic[showPlots, {showPlots = #; gTick += del} &]]}], Spacer[12],
    Row[{"show counters ", Spacer[4], Checkbox[Dynamic[showCounters,
         {showCounters = #; gTick += del} &]]}], SpanFromLeft
   }
  }
 ], SpanFromLeft
},

{Grid[{
```

```
         {"plot x-scale",
          Manipulator[Dynamic[xScale,
             {xScale = #;
               list@setSize[xScale];
               setIC = True;
               gTick += del
              } &], {10, 1000, 1},
           ImageSize -> Tiny, ContinuousAction -> False, Enabled → Dynamic[showPlots]],
          Text@Style[Dynamic@padIt2[xScale, 4], 11]
         }
        }], SpanFromLeft
      }

     }, Frame → True, FrameStyle -> Directive[Thickness[.005], Gray]
    ]
   }
  }, Spacings → {0, {2 → 1, 4 → 1, 6 → 0.5, 8 → 0}}, Alignment → Center, Frame → None],
 (*----control variables--*)
 {{m0, 40}, None},
 {{M0, 10}, None},
 {{R0, 3.9}, None},
 {{delT, 1}, None},
 {{k, 1}, None},
 {{initialThetaDot, 0.09}, None},
 {{initialTheta, 45 Degree}, None},
 {{initialBobX, 1.5}, None},
 {{initialBobSpeed, 0}, None},
 {{isRelaxedSpring, False}, None},
 {{plotType, "bob position"}, None},
 {{testCase, 1}, None},
 {{wallType, "elastic"}, None},
 {{showPlots, False}, None},
 {{showCounters, True}, None},
 {{xScale, 400}, None},
 {{showCoriolisForce, False}, None},
 {{trace, True}, None},
 {{setIC, False}, None},

 (*----- refresh control ---*)
 {{gTick, 0}, ControlType → None},
 {{del, $MachineEpsilon}, None},

 (*----- state variables for sim---*)
 {{runningState, "STOP"}, None}, (*"RUNNING","STEP","SUSPEND","STOP"*)
 {{simulationTime, 0}, None},
 {{wasHitLastTime, False}, None},


 TrackedSymbols :→ {gTick},
 ControlPlacement → Left,
 SynchronousUpdating → False,
 SynchronousInitialization → True,
 ContinuousAction → False,
 Alignment -> Center,
 ImageMargins → 0,
 FrameMargins → 0,
 Paneled → True,
 Frame → False,
 Initialization :→
  {
   (*definitions used for parameter checking*)
   integerStrictPositive = (IntegerQ[#] && # > 0 &);
   integerPositive = (IntegerQ[#] && # ≥ 0 &);
```

```
numericStrictPositive = (Element[#, Reals] && # > 0 &);
numericPositive = (Element[#, Reals] && # ≥ 0 &);
numericStrictNegative = (Element[#, Reals] && # < 0 &);
numericNegative = (Element[#, Reals] && # ≤ 0 &);
bool = (Element[#, Booleans] &);
numeric = (Element[#, Reals] &);
integer = (Element[#, Integers] &);

(* This list object is used to store and the manage the time series generated during running
 of the simulation so that display object can use it to make plots and the animation *)

listClass[$size_?integer(*maximum size of the list*)] :=
 Module[{size, lists, self},
  (*lists are in this order x,v,θ,ω,force,trace*)

  self@getSize[] := size;
  self@setSize[n_] := ( size = n; lists = Table[{0, Table[0, {n}]}, {6}] );

  self@add[{x_?numeric, (*bob position*)
      v_?numeric, (*bob speed*)
      θ_?numeric, (*disk angle*)
      ω_?numeric, (*disk angular speed*)
      force_?numeric(* coriolis force*)}
    ] := Module[{},
    MapThread[Function[{e, idx},
      lists[[idx, 1]] ++;
      If[lists[[idx, 1]] > size, lists[[idx, 1]] = 1];
      lists[[ idx, 2, lists[[idx, 1]] ]] = e
     ],
     {{x, v, θ, ω, force, {x*Cos[θ], x*Sin[θ]}}, Range[1, Length[lists]]}
    ]
   ];

  self@setIC[{x_?numeric, v_?numeric, θ_?numeric, ω_?numeric, force_?numeric}] := (
    list@clear[];
    list@add[{x, v, θ, ω, force}]
   );

  self@getCurrent[] := lists[[ #, 2, lists[[#, 1]] ]] & /@ Range[1, Length[lists]];
  self@clear[] := (lists[[#, 1]] = 0) & /@ Range[1, Length[lists]];
  self@clearTrace[] := lists[[6, 1]] = 0;

  self@getPositionList[] := Module[{len = lists[[1, 1]]}, lists[[ 1, 2, 1 ;; len]] ] ;
  self@getSpeedList[] := Module[{len = lists[[2, 1]]}, lists[[ 2, 2, 1 ;; len]] ] ;
  self@getThetaList[] := Module[{len = lists[[3, 1]]}, lists[[ 3, 2, 1 ;; len]] ] ;
  self@getOmegaList[] := Module[{len = lists[[4, 1]]}, lists[[ 4, 2, 1 ;; len]] ] ;
  self@getCoriolisForceList[] := Module[{len = lists[[5, 1]]}, lists[[ 5, 2, 1 ;; len]] ] ;
  self@getTraceData[] := Module[{len = lists[[6, 1]]}, lists[[ 6, 2, 1 ;; len]] ] ;

  (*--- constructor---*)
  self@setSize[$size];
  self
 ];
(*-------------------------------------------------------*)
solverClass[] :=
 Module[{self},

  (* uses NDSolve to integrate equations of motion for given time step*)
  self@step[m0_?numericPositive, (*bob mass*)
     R0_?numericStrictPositive, (*disk radius*)
     L0_?numericStrictPositive, (*relaxed spring length, measured from the origin*)
     k_?numericStrictPositive, (*spring k*)
     M0_?numericStrictPositive, (*disk mass*)
     delT_?numericStrictPositive, (*time length to integrate over for NDSolve*)
     wallType_String(* wall type is either elastic or inelastic*)
    ] := Module[{ic, eq1, eq2, t, x, θ, maxDistance = R0,
     minDistance = 0, currentSolution, currentX, currentV, currentθ, currentOmega,
     wasHit = False, $currentX, $currentV, $currentθ, $currentOmega, $force, trace},

    {$currentX, $currentV, $currentθ, $currentOmega, $force, trace} = list@getCurrent[];

    currentX = $currentX;
    currentV = $currentV;

    If[ Abs[m0] ≤ $MachineEpsilon,
     currentSolution = First@NDSolve[{θ''[t] == 0, θ[0] == Mod[$currentθ, 2 Pi], θ'[0] == $currentOmega},
        {θ, θ'}, {t, 0, delT}, MaxSteps → Infinity];
     currentθ = (θ /. currentSolution)[delT];
     currentOmega = (θ' /. currentSolution)[delT]
     ,
                      -k (x[t] - L0)
     eq1 = x''[t] == ────────────── + x[t] θ'[t]^2;
                           m0
                      -4 m0 x[t] x'[t] θ'[t]
     eq2 = θ''[t] == ──────────────────────────;
                      M0 R0^2 + 2 m0 x[t]^2

     ic = {x[0] == $currentX, x'[0] == $currentV, θ[0] == Mod[$currentθ, 2 Pi], θ'[0] == $currentOmega};

     currentSolution =
      First@Quiet@NDSolve[Flatten@{eq1, eq2, ic}, {x, x', θ, θ'}, {t, 0, delT}, MaxSteps → Infinity];
     currentX = (x /. currentSolution)[delT];
     currentV = (x' /. currentSolution)[delT];
     currentθ = (θ /. currentSolution)[delT];
     currentOmega = (θ' /. currentSolution)[delT];

     (* if bob hits the edge of the disk or the base of the spring, fix speed and reset things*)
     If[currentX > maxDistance || currentX < minDistance,
      currentV = 0;
      If[wallType == "elastic",
       If[currentX > maxDistance, wasHit = True];
       currentV = -$currentV
      ];
      currentX = $currentX;
      currentθ = $currentθ;
      currentOmega = $currentOmega
     ]
    ];

    (*save current solution in the list object so that display can use it*)
    list@add[{currentX, currentV, currentθ, currentOmega, 2 m0 currentOmega currentV}];
    wasHit
   ];

  self
 ];
(*--------- displayClass ---------------------------------------*)
displayClass[] := Module[{makeCounters, makeDiskDiagram, makePlot, self},

  (*--------- private ---------------------------------------*)
  makeCounters[simulationTime_?numericPositive,
    m0_?numericPositive,
    M0_?numericStrictPositive,
    R0_?numericStrictPositive,
    k_?numericStrictPositive] :=
   Module[{h1, h2, currentMomentOfInertia, x, v, θ, ω, currentCoriolisForce, PE, KE, trace},
    {x, v, θ, ω, currentCoriolisForce, trace} = list@getCurrent[];
    currentMomentOfInertia = M0 / 2 * R0 ^ 2 + m0 * x ^ 2;
    PE = 0.5 * k * (x - R0 / 2) ^ 2;
         1                1
    KE = — M0 R0 ^ 2 ω ^ 2 + — m0 (v ^ 2 + x ^ 2 ω ^ 2);
         4                2

    h1 = Text@Style[Grid[{
         {Text["disk θ"],
          Text["disk ω"],
          Text["bob position"],
          Text["bob velocity"],
          Text[Style["I", Italic]],
          Text["energy"]
         },
         {Text["(degree)"],
          Text["(rad/sec)"],
          Text["(meter)"],
          Text["(meter/sec)"],
          Text["(kg meter^2)"],
          Text[Row[{"(", Style["J", Italic], ")"}]]
         },
         { padIt2[180. / Pi * θ, {5, 2}],
          padIt2[ω, {7, 6}],
          padIt2[x, {3, 2}],
          padIt1[v, {6, 3}],
          padIt2[currentMomentOfInertia, {5, 2}],
          padIt2[PE + KE, {4, 3}]
         }
        },
        Frame → {All, None, { {{1, 2}, {1, 6}} → True , {{3, 3}, {1, 6}} → True}},
        FrameStyle → Gray,
        Spacings → 1,
        ItemSize → {{All, 2 ;; -1} → 5},
     Alignment → Center], 12];

    h2 = Text@Style[Grid[{
         {Text["time (sec)"],
          Text["I ω (joule second)"],
          Text@Row[{"Coriolis force (", Style["N", Italic], ")"}],
          Text@Row[{"P.E. (", Style["J", Italic], ")"}],
          Text@Row[{"K.E. (", Style["J", Italic], ")"}]
         },
         { padIt2[simulationTime, {5, 2}],
          padIt2[currentMomentOfInertia * ω, {6, 4}],
          padIt1[currentCoriolisForce, {5, 4}],
          padIt2[PE, {5, 4}],
          padIt2[KE, {5, 4}]
         }},
     Frame → All,
     FrameStyle → Gray,
     Spacings → 1,
     ItemSize → {{All, 2 ;; -1} → 6},
     Alignment → Center], 12];

    Grid[{{h1}, {h2}}, Spacings → {0, .1}]
   ];
  (*---------------private----------------------------------*)
  makeDiskDiagram[m0_?numericPositive,
    R0_?numericStrictPositive,
    r_?numericStrictPositive,
    wasHit_?bool,
    {w_?numericStrictPositive, h_?numericStrictPositive},
    showCoriolisForce_?bool,
    trace_?bool] := Module[{x, y, splash, xx, v, θ, ω, force, traceData, currentTrace},

    {xx, v, θ, ω, force, currentTrace} = list@getCurrent[];
    force = force / (4 * 1.25) + 0.25 * Sign[force];

    x = xx * Cos[θ];
    y = xx * Sin[θ];
    traceData = list@getTraceData[];

    splash = If[wasHit, {
       { Dotted, Red, Rotate[Line[{{R0, 0}, {1.2 R0, 0}}], θ, {0, 0}]},
       {Dotted, Red, Rotate[Line[{{R0, 0}, {1.2 R0, .2 R0}}], θ, {0, 0}]},
       {Dotted, Red, Rotate[Line[{{R0, 0}, {1.2 R0, .1 R0}}], θ, {0, 0}]},
       {Dotted, Red, Rotate[Line[{{R0, 0}, {1.2 R0, -.2 R0}}], θ, {0, 0}]},
       {Dotted, Red, Rotate[Line[{{R0, 0}, {1.2 R0, -.1 R0}}], θ, {0, 0}]}
      },
      Sequence @@ {}
     ];

    Graphics[{
      {EdgeForm[{Thick, Blue}], FaceForm@RGBColor[.75, .97, .97], Disk[{0, 0}, R0 ]},
      If[showCoriolisForce && Abs[force] > $MachineEpsilon,
       {Dotted, Black,
        Rotate[Arrow[{{xx, Sign[force] * r}, {xx, (force + Sign[force] * r)}}], θ, {0, 0}]}, Sequence @@ {}
      ],
      If[trace, {Dotted, Darker@Red, Line[traceData]}, Sequence @@ {}],
      If[m0 ≤ $MachineEpsilon, Sequence @@ {}, { Red, Disk[{x, y}, r]}],
      {Black, Line@makeSpring[0, 0, (xx - r) Cos[θ], (xx - r) Sin[θ], 0.8 * r]},

      (*relaxed position indicator*)
      {Dashed, Gray, Rotate[Line[{{R0 / 2, -r}, {R0 / 2, -2 r}}], θ, {0, 0}]},
      {Dashed, Gray, Rotate[Line[{{R0 / 2, r}, {R0 / 2, 2 r}}], θ, {0, 0}]},

      (* inertial frames*)
      {Gray, Dashed, Rotate[Line[{{-R0, 0}, {0, 0}}], θ, {0, 0}]},
      {Gray, Dashed, Rotate[Line[{{0, -R0}, {0, R0}}], θ, {0, 0}]},
      splash
     },
     ImageSize → {w, h},
     ImagePadding → {{10, 10}, {10, 10}},
     PlotRange → {{-R0 - r, R0 + r}, {-R0 - r, R0 + r}},
     ImageMargins → 1,
     AspectRatio → 1
    ]
   ];
  (*-------------------- private -----------------------------*)
  makePlot[plotType_String,
    R0_?numericStrictPositive] := Module[{plotTitle, data, n = list@getSize[]},

    Which[plotType == "disk angular velocity",
     data = list@getOmegaList[];
     If[Length[data] == 0, data = {0}];
     plotTitle = {{Text@Style["ω (rad/sec)", 12], None},
       {Text@Style["time"], Text@Style["disk angular velocity vs. time", 12]}};
     plotRange = {{1, n}, All}
     ,
     plotType == "disk angle",
     data = 180.0 / Pi * list@getThetaList[];
```

```
    If[Length[data] == 0, data = {0}];
    plotTitle = {{Text@Style["θ (deg)", 12], None},
      {Text@Style["time"], Text@Style["disk angle vs. time", 12]}};
    plotRange = {{1, n}, {0, 360}}
    ,
    plotType == "bob position",
    data = list@getPositionList[];
    If[Length[data] == 0, data = {0}];
    plotTitle = {{Text@Style[Row[{"position (", Style["m", Italic], ")"}], 12], None},
      {Text@Style["time"], Text@Style["bob position vs. time", 12]}};
    plotRange = {{1, n}, {0, R0}}
    ,
    plotType == "bob velocity",
    data = list@getSpeedList[];
    If[Length[data] == 0, data = {0}];
    plotTitle = {{Text@Style[Row[{"velocity (", Style["m/s", Italic], ")"}], 12], None},
      {Text@Style["time"], Text@Style["bob velocity vs. time", 12]}};
    plotRange = {{1, n}, All}
    ,
    plotType == "Coriolis force",
    data = list@getCoriolisForceList[];
    If[Length[data] == 0, data = {0}];
    plotTitle = {{Text@Style[Row[{"Coriolis force (", Style["N", Italic], ")"}], 12], None},
      {Text@Style["time"], Text@Style["Coriolis force vs. time", 12]}};
    plotRange = {{1, n}, All}
   ];

   ListPlot[ data,
    PlotRange → plotRange,
    AxesOrigin → {0, 0},
    Joined → True,
    Frame → True,
    GridLines → Automatic,
    ImageSize → {330, 160},
    ImagePadding → {{60, 10}, {30, 25}},
    ImageMargins -> {{2, 1}, {1, 1}},
    Axes → False,
    FrameLabel → plotTitle,
    PlotStyle → Red,
    AspectRatio → 0.27]
  ];
(*------- public ----------------------------------*)
self@makeDisplay[simulationTime_?numericPositive,
    plotType_String,
    m0_?numericPositive,
    M0_?numericStrictPositive,
    R0_?numericStrictPositive,
    r0_?numericStrictPositive,
    wasHit_?bool,
    k_?numericStrictPositive,
    showPlots_?bool,
    showCounters_?bool,
    showCoriolisForce_?bool,
    trace_?bool] :=
  If[showPlots,
   If[showCounters,
    Grid[{
      {makeCounters[simulationTime, m0, M0, R0, k]},
      {makePlot[plotType, R0]},
      {makeDiskDiagram[m0, R0, r0, wasHit,
        {1.4 ContentSizeW, 0.515 ContentSizeH}, showCoriolisForce, trace]}
     }, Spacings → 0]
    ,
     Grid[{
        {makePlot[plotType, R0]},
        {makeDiskDiagram[m0, R0, r0, wasHit,
           {1.4 ContentSizeW, 0.787 ContentSizeH}, showCoriolisForce, trace]}
       }, Spacings → 0]
    ],
    If[showCounters,
     Grid[{
        {makeCounters[simulationTime, m0, M0, R0, k]},
        {makeDiskDiagram[m0, R0, r0,
           wasHit, {1.4 ContentSizeW, 0.93 ContentSizeH}, showCoriolisForce, trace]}
       }, Spacings → 0]
     ,
     Grid[{
        {makeDiskDiagram[m0, R0, r0,
           wasHit, {1.4 ContentSizeW, 1.2 ContentSizeH}, showCoriolisForce, trace]}
       }, Spacings → 0]
    ]
   ];

  self
 ];
(*-------------------------------------------*)
(* helper function for formatting            *)
(*-------------------------------------------*)
padIt1[v_?numeric, f_List] :=
 AccountingForm[Chop[v] , f, NumberSigns → {"-", "+"}, NumberPadding → {"0", "0"}, SignPadding → True];
(*-------------------------------------------*)
(* helper function for formatting            *)
(*-------------------------------------------*)
padIt2[v_?numeric, f_List] :=
 AccountingForm[Chop[v] , f, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}, SignPadding → True];
padIt2[v_?numeric, f_Integer] := AccountingForm[Chop[v] , f, NumberSigns → {"", ""},
  NumberPadding → {"0", "0"}, SignPadding → True];

(*This function is called to make spring,based on code by Arpad Kosa from WRI demo*)
(*at Wolfram web site modified by me.This returns a Line which is the spring*)
(*szel, larger number means bigger spring width*)
makeSpring[xFirst_?numeric, yFirst_?numeric, xEnd_?numeric, yEnd_?numeric, szel_?numeric] :=
 Module[{hx, veghossz, hossz, hy, dh, tbl},
  hx = xEnd - xFirst;
  If[Abs[hx] ≤ $MachineEpsilon, hx = 10^-6];
  hy = yEnd - yFirst;
  If[Abs[hy] ≤ $MachineEpsilon, hy = 10^-6];

  veghossz = 0.03;
  hossz = Sqrt[hx^2 + hy^2];
  dh = (hossz - 2 * veghossz) / 20;
  tbl = Table[If[OddQ[i], {xFirst + hx * (i * dh + veghossz) / hossz + hy * szel / hossz,
      yFirst + hy * (i * dh + veghossz) / hossz - hx * szel / hossz}, {xFirst + hx * (i * dh + veghossz) / hossz -
       hy * szel / hossz, yFirst + hy * (i * dh + veghossz) / hossz + hx * szel / hossz}], {i, 2, 18}];
  {{xFirst, yFirst}} ~ Join ~ {{xFirst + hx * (dh + veghossz) / hossz, yFirst + hy * (dh + veghossz) / hossz}} ~
   Join ~ tbl ~ Join ~ {{xFirst + hx * (19 * dh + veghossz) / hossz, yFirst + hy * (19 * dh + veghossz) / hossz}} ~
   Join ~ {{xEnd, yEnd}}
 ];

(*--- constant parameters size and width of display ---*)
ContentSizeW = 260;
ContentSizeH = 405;

(* objects used by the simulation. These must be here in the initialization section *)
list = listClass[400];
list@add[{1.5, 0.0, Pi / 4, 0.09, 0.0}];
```

```
    solver = solverClass[];
    display = displayClass[];
  }
]
```



simulation parameters

simulation speed 3.77
disk mass 266
disk radius 2.2
bob mass 40.0
spring stiffness 1.2
disk edge [✓ elastic] inelastic

initial conditions

angular velocity +0.02
angle (degree) 045.0
bob position 1.5
relax ☐ trace ✓ show coriolis force ☐
bob speed +0.00

plot type: bob position
test case: 1
show plot ✓ show counters ✓
plot x–scale 0400

disk θ (degree)	disk ω (rad/sec)	bob position (meter)	bob velocity (meter/sec)	I (kg meter^2)
355.74	0.089833	1.51	−000.000	735.09

time (sec)	I ω (joule second)	Coriolis force (N)	P.E. (J)	K
570.19	66.0348	−0.0019	0.1015	2

Caption

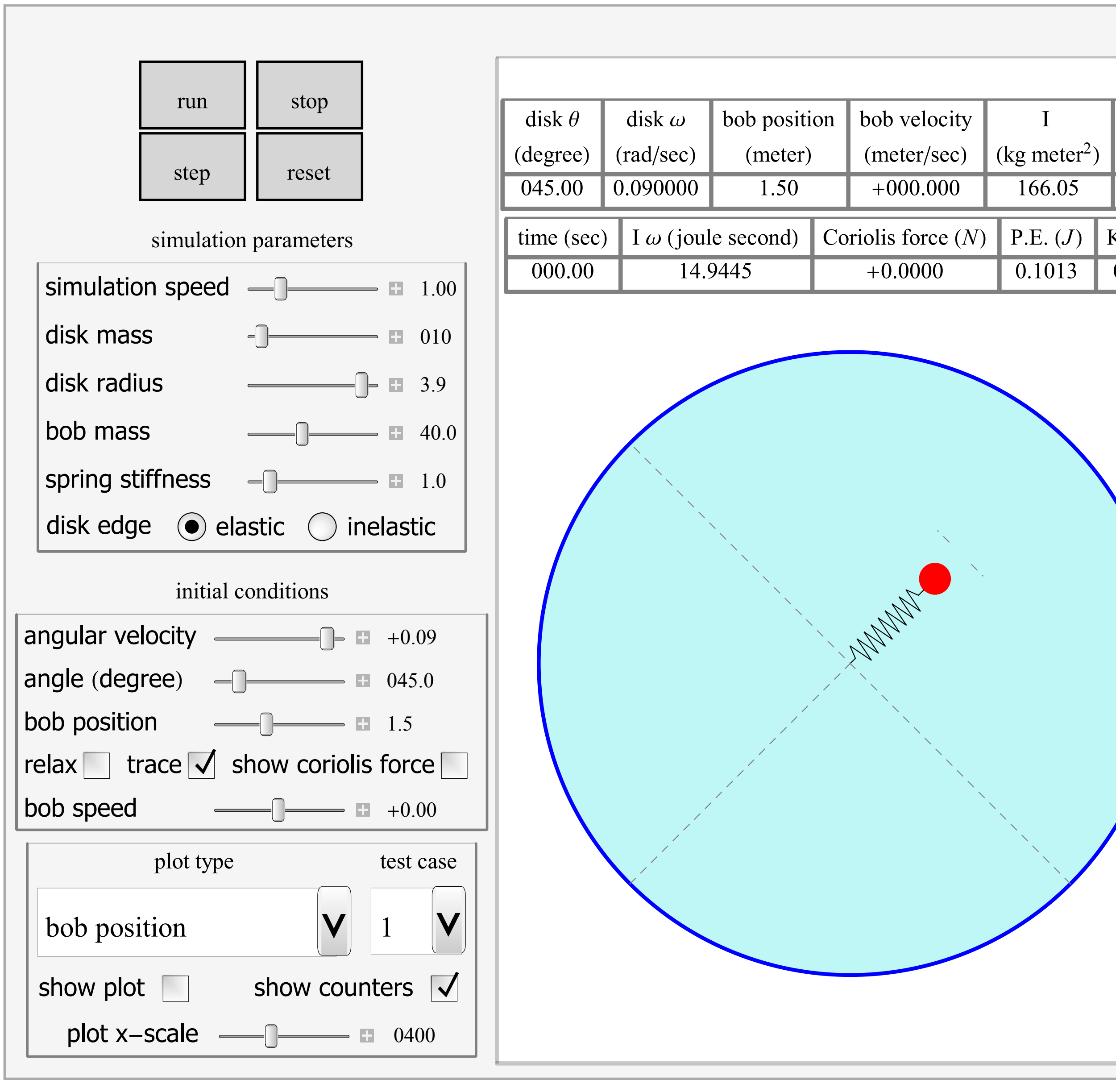
This Demonstration describes the dynamics of a spring-mass system on a rotating disk in the horizontal plane. A mass at the end of a spring moves back and forth along the radius of a spinning disk. The spring is anchored to the center of the disk, which is the origin of an inertial coordinates system. The bob mass is considered part of the overall disk mass for purpose of calculation of the mass moment of inertia; therefore the overall mass moment of inertia around the axis of rotation changes as the bob distance relative to the disk center changes.

There is no friction between the bob and the disk and the spring mass is negligible. The spring is assumed to be infinitely rigid against rotational movement and will remain straight as it vibrates.

In this system, the overall moment of linear momentum, also called the angular momentum, is constant since no external torque and no energy dissipation is present, therefore when the bob is near the center of the disk, the angular speed of the disk increases, since

the mass moment of inertia has decreased. Conversely, when the bob is close to the edge of the disk, the angular speed of the disk decreases, since the mass moment of inertia has increased. This is similar to what happens when ice skaters pull their hands closer to their body in order to increase the speed at which they are spinning.
The table has a raised edge and when the bob hits the edge, its velocity is set to zero if the edge wall is an inelastic type otherwise its velocity reverses direction if the edge is an elastic type. Pure elastic and inelastic material is assumed at the edge.

Thumbnail



simulation parameters

simulation speed	1.00
disk mass	010
disk radius	3.9
bob mass	40.0
spring stiffness	1.0
disk edge	[✓ elastic] inelastic

initial conditions

angular velocity	+0.09
angle (degree)	045.0
bob position	1.5
bob speed	+0.00

relax ☐ trace ✓ show coriolis force ☐

plot type: bob position; test case: 1

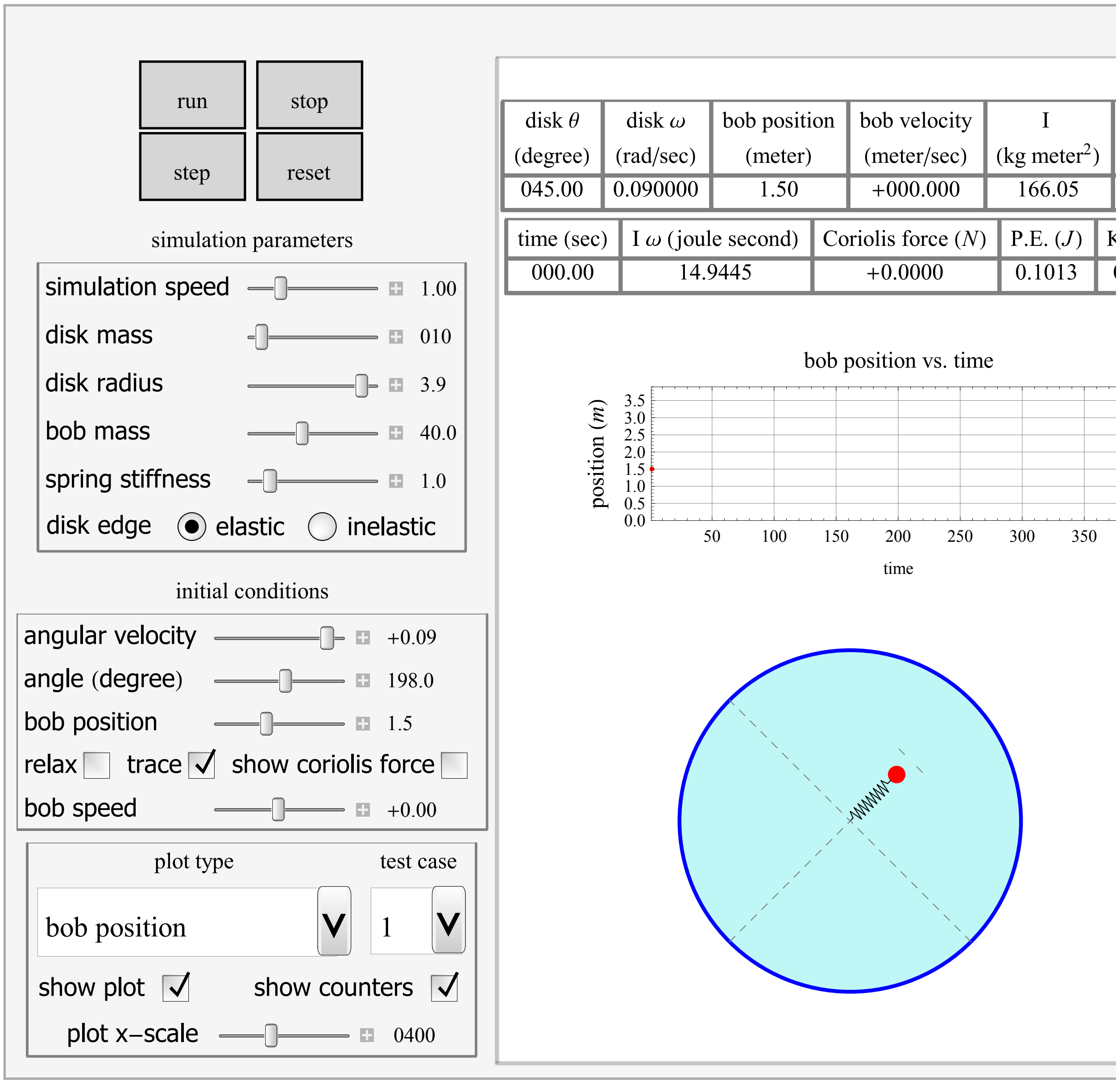
show plot ☐ show counters ✓

plot x–scale 0400

disk θ (degree)	disk ω (rad/sec)	bob position (meter)	bob velocity (meter/sec)	I (kg meter^2)
045.00	0.090000	1.50	+000.000	166.05

time (sec)	I ω (joule second)	Coriolis force (N)	P.E. (J)
000.00	14.9445	+0.0000	0.1013

Snapshots



disk θ (degree)	disk ω (rad/sec)	bob position (meter)	bob velocity (meter/sec)	I (kg meter^2)
045.00	0.090000	1.50	+000.000	166.05

time (sec)	I ω (joule second)	Coriolis force (N)	P.E. (J)	K
000.00	14.9445	+0.0000	0.1013	

simulation parameters

simulation speed	1.00
disk mass	010
disk radius	3.9
bob mass	40.0
spring stiffness	1.0
disk edge	[✓ elastic] inelastic

initial conditions

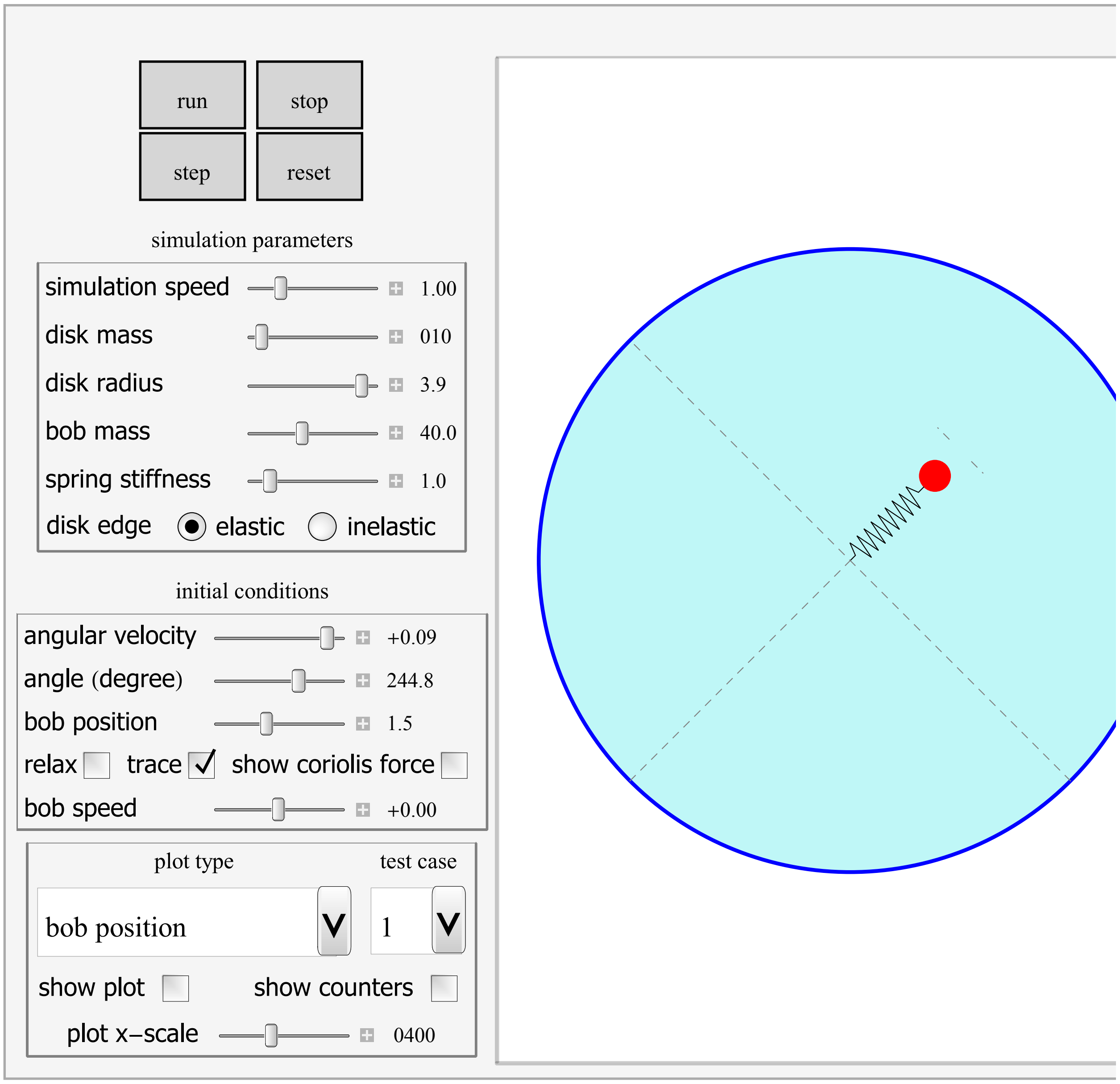
angular velocity	+0.09
angle (degree)	198.0
bob position	1.5
bob speed	+0.00

relax ☐ trace ☑ show coriolis force ☐

plot type	test case
bob position	1

show plot ☑ show counters ☑

plot x–scale 0400



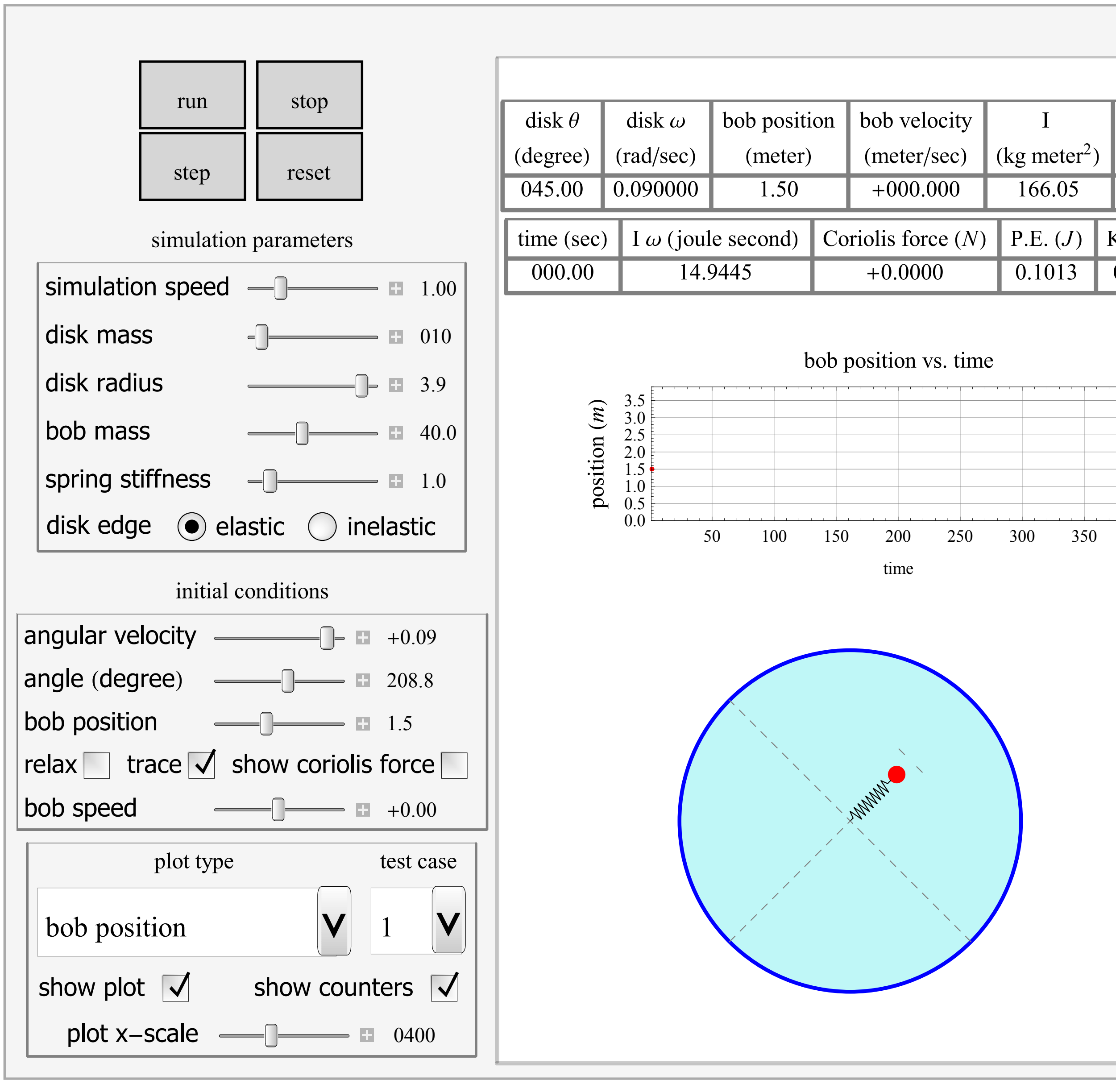
simulation parameters

simulation speed 1.00
disk mass 010
disk radius 3.9
bob mass 40.0
spring stiffness 1.0
disk edge [✓ elastic] inelastic

initial conditions

angular velocity +0.09
angle (degree) 244.8
bob position 1.5
relax ☐ trace ✓ show coriolis force ☐
bob speed +0.00

plot type: bob position
test case: 1
show plot ☐ show counters ☐
plot x–scale 0400



simulation parameters

simulation speed	1.00
disk mass	010
disk radius	3.9
bob mass	40.0
spring stiffness	1.0
disk edge	[✓ elastic] inelastic

initial conditions

angular velocity	+0.09
angle (degree)	208.8
bob position	1.5
relax ☐ trace ✓ show coriolis force ☐	
bob speed	+0.00

plot type	test case
bob position	1
show plot ✓	show counters ✓
plot x–scale	0400

disk θ (degree)	disk ω (rad/sec)	bob position (meter)	bob velocity (meter/sec)	I (kg meter^2)
045.00	0.090000	1.50	+000.000	166.05

time (sec)	I ω (joule second)	Coriolis force (N)	P.E. (J)	K
000.00	14.9445	+0.0000	0.1013	(

Details (optional)

Following is the mathematical model and solution method. The variables are the following:

- M is the disk mass
- m is the mass of the bob
- k is the spring stiffness
- R is the disk radius
- x_0 is the relaxed length of the spring (measured from the center of the disk and taken to be half the length of the disk radius)
- x is the current position of the bob along the radius measured from the center of the disk
- θ is the disk rotation angle
- θ' is the angular velocity of the disk
- I is the total mass moment of inertia around the axis of rotation through the center of the disk and is given by $\frac{1}{2} M R^2 + m x^2$, where

the bob mass m is considered a point mass

- $x(0)$ is the initial position of the bob along the radius
- $x'(0)$ is the initial speed of the bob
- $\theta(0)$ is the initial angle of the disk
- $\theta'(0)$ is the initial speed of the disk.

All units are in SI and counterclockwise is taken as positive.

derivation of equations of motion

The equations of motions are derived using both the Lagrangian and Newton's methods in order to verify the mathematical model.

The bob has one degree of freedom: the distance x along the disk radius. The disk likewise has one degree of freedom: the angle of rotation θ. The equation of motion for the bob in the x direction is $x'' = \frac{-k(x-x_0)}{m} + x(\theta')^2$ and the equation of motion for the disk is $\theta'' = \frac{-2\,m x x'\theta'}{1/2\,M R^2 + m x^2}$ .The torque term in this equation is represented by $2\,m x x'\theta'$ and since there is no external torque present, this torque is due only to the Coriolis force and not due to external forces. These two second-order coupled differential equations are solved using `NDSolve` and the solution simulated. For more information on the derivation of the equations of motion, please see the author's report.

description of the user interface

The top four buttons are used to control the simulation of the solution. The reset button brings the simulation back to the initial conditions. The control variables below these are used to change the initial conditions for the disk and the bob. The "relax" check box is used to put the spring into the relaxed initial position, which is at half the current value of the radius of the disk (this location is shown by two small markers at the disk surface). You can see a trace of the bob motion using the "trace" check box. You can change the length of the disk radius. The bob radius is fixed at 20 *cm* in the simulation therefore as you change the radius of the disk the bob will appear to change in size relative to current disk size.
You can select the behavior of what happens when the bob hits the edge of the table by choosing the edge to be pure elastic or inelastic using the radio button selection.
The main `Manipulate` display has three parts. The top part contains counters that show the current state. The field labeled "I" is the mass moment of inertia of the disk and the bob combined. The field labeled "E" is the sum of the potential energy (P.E.) and the kinetic energy (K.E.). The middle part of the display shows different plots. The plot shown is selected using the pop-up menu labeled "plot type". The third part shows the disk itself as it spins. You can select the parts to see using the check boxes to the left.
A pop-up menu labeled "test case" is used to select one of three preconfigured simulation parameters. Click on the run button after selecting a test case to start the simulation.
The simulation runs with no throttling by using a single dynamic variable called "`gTick`" that is updated or ticked at the end of each `Manipulate` expression evaluation so that the `Manipulate` expression is reevaluated again immediately. This allows the simulation to run at the maximum speed. A slider labeled "simulation speed" can be used to adjust the speed of the simulation as needed.
The simulation runs continuously until you stop it. The simulation time counter rewinds each 1000 simulation seconds.

References

[1] S. Timoshenko, D.H. Young, *Advanced Dynamics,* New York: McGraw-Hill, 1948.

[2] S. Timoshenko, *Vibration Problems In Engineering,* New York: D. Van. Nostrand, 1937.

[3] REA's *Problem Solver's Mechanics static and dynamics,* New Jersey: Research and Education Association, 2002.

[4] D. A. Wells, *Lagrangian Dynamics*, New York: Schaums' Outline, 1967.

Control Suggestions (optional)

☐ Resize Images

✓ Rotate and Zoom in 3D

☐ Drag Locators

☐ Create and Delete Locators

✓ Slider Zoom

☐ Gamepad Controls

✓ Automatic Animation

☐ Bookmark Animation

Search Terms (optional)

angular momentum

mass spring

Coriolis force

vibration

moment of momentum

Related Links (optional)

angular momentum

NDSolve

Authoring Information

Contributed by: Nasser M. Abbasi