Linear Quadratic Regulator Control of an Inverted Pendulum with Friction

Initialization Code (optional)

Manipulate

```
Manipulate[
 tick;
 needToTick = False;

 currentTime = currentTime + delT;
 If[( (currentTime + delT) ≤ simulationTime) && (run || oneStep),
  (
   invertedPendulum@makeStep[delT];

   If[oneStep,
    (
     oneStep = False;
     run = False
    ),
    (
     needToTick = True
    )
   ]
  )];

 {cx, ctheta, cxSpeed, cThetaSpeed} = invertedPendulum@getCurrentPosition[];
 display@addPoint[cx, cxSpeed, ctheta, cThetaSpeed, currentTime];
 finalPlot = display@getPlot[];
 If[needToTick, tick += del];
 FinishDynamic[];
 finalPlot,

 Grid[{
   {Button[Text@Style["run", 12], {run = True; oneStep = False;
      display@reset[simulationTime, delT, ic]; currentTime = 0; tick += del}, ImageSize -> {50, 35}],
    Button[Text@Style["step", 12], {run = True; oneStep = True, tick += del}, ImageSize -> {50, 35}]},
   {
    Button[Text@Style["stop", 12], {run = False; oneStep = False; tick += del}, ImageSize -> {50, 35}],
    Button[Text@Style["reset", 12], {currentTime = -delT; invertedPendulum@reset[]; display@
       reset[simulationTime, delT, ic]; run = False; oneStep = False; tick += del}, ImageSize -> {50, 35}]
   }
  }, Spacings → {0.2, 0.2}, Alignment → Center],

 Grid[{
   {Text@Style["time", 12],
    Manipulator[Dynamic[simulationTime, {simulationTime = #; currentTime = -delT; invertedPendulum@reset[];
        display@reset[simulationTime, delT, ic]; run = False; tick += del} &], {1, 100, 1}, ImageSize → Tiny],
    Text@Style[Dynamic@padIt2[simulationTime, {3, 0}], 11]
   },

   {Text@Style["slow", 12],
    Manipulator[Dynamic[delT, {delT = #; currentTime = -delT; invertedPendulum@reset[];
        display@reset[simulationTime, delT, ic]; run = False} &], {0.01, 0.3, 0.01}, ImageSize → Tiny],
```

```
    Text@Style["fast", 12]
   }
  }, Frame → True, FrameStyle -> Directive[Thickness[.001], Gray]
],

Grid[{
  {Text@Style["θ(0)", 12],
   Manipulator[
    Dynamic[ic, {ic = #; currentTime = -delT; run = False; display@reset[simulationTime, delT, ic];
       invertedPendulum@setInitialAngle[ic * Pi / 180.]; tick += del} &], {70, 110, 1}, ImageSize → Tiny],
   Text[Row[{Style[Dynamic@padIt2[ic, {3, 0}], 11], Degree}]]
  },
  {Text@Style[Row[{Style["x", Italic], "(0)"}], 12],
   Manipulator[
    Dynamic[icx, {icx = #; currentTime = -delT; run = False; display@reset[simulationTime, delT, ic];
       invertedPendulum@setInitialX[icx]; tick += del} &], {-2, 2, 0.1}, ImageSize → Tiny],
   Text[Row[{Style[Dynamic@padIt1[icx, {2, 1}], 11]}]]
  }
 }, Frame → True, FrameStyle -> Directive[Thickness[.001], Gray]],

Grid[{
  {Text@Style["bob mass", 12],
   Manipulator[
    Dynamic[bobMass, {bobMass = #; currentTime = -delT; run = False; display@reset[simulationTime, delT, ic];
       invertedPendulum@setBobMass[bobMass]; tick += del} &], {0.1, 10, 0.1}, ImageSize → Tiny],
   Text@Style[Dynamic@padIt2[bobMass, {3, 1}], 11]
  },

  {Text@Style["cart mass", 12],
   Manipulator[Dynamic[cartMass,
     {cartMass = #; currentTime = -delT; run = False; display@reset[simulationTime, delT, ic];
       invertedPendulum@setCartMass[cartMass]; tick += del} &], {0.1, 10, 0.1}, ImageSize → Tiny],
   Text@Style[Dynamic@padIt2[cartMass, {3, 1}], 11]
  },

  {Text@Style["length", 12],
   Manipulator[Dynamic[pendulumLength, {pendulumLength = #; currentTime = -delT; run = False;
       display@reset[simulationTime, delT, ic]; invertedPendulum@setPendulumLength[pendulumLength];
       tick += del} &], {1, 2, 0.01}, ImageSize → Tiny],
   Text@Style[Dynamic@padIt2[pendulumLength, {2, 1}], 11]
  }

 }, Frame → True, FrameStyle -> Directive[Thickness[.001], Gray]
],

Grid[{
  {Text@Style["Q matrix diagonal", 12], SpanFromLeft},
  {Text@Style[Row[{Style["x", Italic], "(", Style["t", Italic], ")"}], 12],
   Manipulator[Dynamic[positionWeight, {positionWeight = #; currentTime = -delT; run = False;
       display@reset[simulationTime, delT, ic]; invertedPendulum@setPositionWeight[positionWeight];
       tick += del} &], {1, 1000, 1}, ImageSize → Tiny],
   Text@Style[Dynamic@padIt2[positionWeight, {4, 0}], 11]
  },

  {Text@Style[Row[{Style["x'", Italic], "(", Style["t", Italic], ")"}], 12],
   Manipulator[Dynamic[linearVelocityWeight,
     {linearVelocityWeight = #; currentTime = -delT; run = False; display@reset[simulationTime, delT, ic];
       invertedPendulum@setLinearVelocityWeight[linearVelocityWeight];
       tick += del} &], {1, 1000, 1}, ImageSize → Tiny],
   Text@Style[Dynamic@padIt2[linearVelocityWeight, {4, 0}], 12]
  },

  {Text@Style[Row[{"θ(", Style["t", Italic], ")"}], 12],
   Manipulator[Dynamic[angleWeight,
```

```
      {angleWeight = #; currentTime = -delT; run = False; display@reset[simulationTime, delT, ic];
        invertedPendulum@setAngleWeight[angleWeight]; tick += del} &], {1, 1000, 1}, ImageSize → Tiny],
    Text@Style[Dynamic@padIt2[angleWeight, {4, 0}], 11]
   },

   {Text@Style[Row[{"θ'(", Style["t", Italic], ")"}], 12],
    Manipulator[Dynamic[angluarVelocityWeight,
      {angluarVelocityWeight = #; currentTime = -delT; run = False; display@reset[simulationTime, delT, ic];
        invertedPendulum@setAngluarVelocityWeight[angluarVelocityWeight];
        tick += del} &], {1, 1000, 1}, ImageSize → Tiny],
    Text@Style[Dynamic@padIt2[angluarVelocityWeight, {4, 0}], 11]
   }}, Frame → True, FrameStyle -> Directive[Thickness[.001], Gray]
 ],

 Grid[{
   {Text@Style["friction coefficients", 12], SpanFromLeft},
   {Text@Style["static", 11],
    Manipulator[Dynamic[staticFrictionCoefficient, {staticFrictionCoefficient = #; currentTime = -delT;
        run = False; invertedPendulum@setStaticFriction[staticFrictionCoefficient];
        display@reset[simulationTime, delT, ic]; tick += del} &], {0, 0.1, 0.05}, ImageSize → Tiny],
    Text@Style[Dynamic@padIt2[staticFrictionCoefficient, {3, 2}], 11]
   },
   {Text@Style["kinetic", 11],
    Manipulator[Dynamic[kineticFrictionCoefficient, {kineticFrictionCoefficient = #; currentTime = -delT;
        run = False; invertedPendulum@setKineticFriction[kineticFrictionCoefficient];
        display@reset[simulationTime, delT, ic]; tick += del} &], {0, 0.05, 0.01}, ImageSize → Tiny],
    Text@Style[Dynamic@padIt2[kineticFrictionCoefficient, {3, 2}], 11]
   },
   {Text@Style["viscous", 11],
    Manipulator[Dynamic[viscousFrictionCoefficient, {viscousFrictionCoefficient = #; currentTime = -delT;
        run = False; invertedPendulum@setViscousFriction[viscousFrictionCoefficient];
        display@reset[simulationTime, delT, ic]; tick += del} &], {0, 6, 0.05}, ImageSize → Tiny],
    Text@Style[Dynamic@padIt2[viscousFrictionCoefficient, {3, 2}], 11]
   }
  }, Alignment → Center, Frame → True, FrameStyle -> Directive[Thickness[.001], Gray]
 ],

 {{oneStep, False}, None},
 {{currentTime, -0.1}, None},
 {tick, None},
 {{run, False}, None},
 {p, None},
 {cx, None},
 {ctheta, None},
 {cxSpeed, None},
 {cThetaSpeed, None},

 {{ic, 75}, None},
 {{icx, 1}, None},
 {{bobMass, 1}, None},
 {{cartMass, 10}, None},
 {{pendulumLength, 2}, None},
 {{delT, 0.1}, None},
 {{del, $MachineEpsilon}, None},
 {finalPlot, None},

 {{positionWeight, 100}, None},
 {{linearVelocityWeight, 10}, None},
 {{angleWeight, 10}, None},
 {{angluarVelocityWeight, 1}, None},
 {{simulationTime, 10}, None},

 {{staticFrictionCoefficient, 0.05}, None},
 {{kineticFrictionCoefficient, 0.03}, None},
```

```
{{viscousFrictionCoefficient, 3.0}, None},
{needToTick, None},

TrackedSymbols :> {tick},

SynchronousUpdating → False,
ContinuousAction → False,
SynchronousInitialization → True,
Alignment -> Center,
ImageMargins → 0,
FrameMargins → 0,
Paneled → True,
Frame → False,
ControlPlacement → Left,
AutorunSequencing → {1},


Initialization :>
 {
  ContentSizeW = 430;
  ContentSizeH = 520 ;
  (*----------------------------------------*)
  padIt1[v_?(NumericQ[#] && Im[#] == 0 &), f_List] := AccountingForm[Chop[N@v] ,
    f, NumberSigns -> {"-", "+"}, NumberPadding -> {"0", "0"}, SignPadding -> True];
  (*----------------------------------------*)
  padIt2[v_?(NumericQ[#] && Im[#] == 0 &), f_List] :=
   AccountingForm[Chop[N@v] , f, NumberSigns -> {"", ""}, NumberPadding -> {"0", "0"}, SignPadding -> True];

  (*----------------------------------------*)
  invertedPendulumClass[$bobMass_, $cartMass_, $pendulumLen_, $staticFriction_, $kineticFriction_,
    $viscousFriction_, $initialAngle_, $initialX_, $stepSize_, $positionWeight_,
    $linearVelocityWeight_, $angleWeight_, $angluarVelocityWeight_, $x_, $θ_, $t_] := Module[{

     bobMass = $bobMass,
     cartMass = $cartMass,
     pendulumLen = $pendulumLen,
     staticFriction = $staticFriction,
     kineticFriction = $kineticFriction,
     viscousFriction = $viscousFriction,
     initialAngle = $initialAngle,
     initialX = $initialX,
     positionWeight = $positionWeight,
     linearVelocityWeight = $linearVelocityWeight,
     angleWeight = $angleWeight,
     angluarVelocityWeight = $angluarVelocityWeight,
     x = $x,
     θ = $θ,
     t = $t,
     eqns,
     lastAngle = $initialAngle,
     lastX = $initialX,
     lastSpeed = 0,
     lastAngularSpeed = 0,
     lastAppliedForce = 0,
     lastCoulombForce = 0,
     lastViscousForce = 0,
     normalForce,
     stateControlExpression,
     closedLoopEigenvalues,
     gain,
     f,
     model,
     self,
```

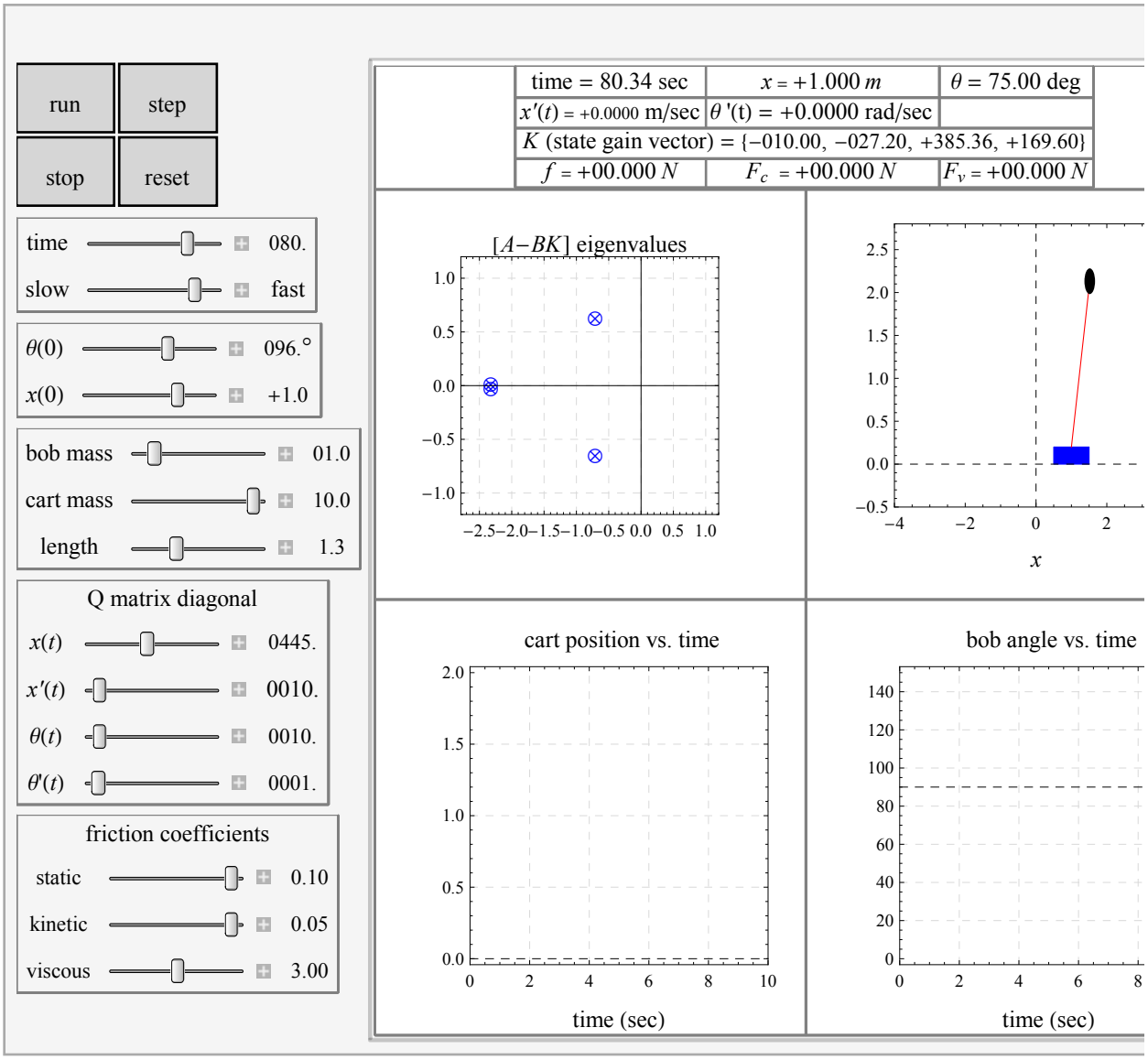
```
 update,
 init,
 calculateFrictionForce},

(*private methods *)
(*--------------------------------------*)
init[] := Module[{ke, pe, lag, eq1, eq2, g = 9.8, currentSolution},

  ke = 1 / 2 cartMass x'[t]^2 +
    1 / 2 bobMass * (x'[t]^2 + pendulumLen^2 *θ'[t]^2 - 2 x'[t] *pendulumLen *θ'[t] Sin[θ[t]]);

  pe = bobMass * g * pendulumLen * Sin[θ[t]];
  lag = ke - pe;
  eq1 = D[D[lag, θ'[t]], t] - D[lag, θ[t]] == 0;
  eq2 = D[D[lag, x'[t]], t] - D[lag, x[t]] == f[t] - viscousFriction * x'[t];

  eqns = {eq1, eq2};

  model = StateSpaceModel[eqns,
    {{x[t], 0}, {x'[t], 0}, {θ[t], Pi / 2}, {θ'[t], 0}}, {{f[t], 0}}, {θ[t], x[t]}, t];

  gain = First@LQRegulatorGains[N[model], {DiagonalMatrix[
       {positionWeight, linearVelocityWeight, angleWeight, angluarVelocityWeight}], {{1}}}];
  stateControlExpression = {x[t], x'[t], θ[t] - Pi / 2, θ'[t]};
  closedLoopEigenvalues =
   {{Eigenvalues[First[Normal[SystemsModelStateFeedbackConnect[N[model], {gain}]]]]}};

  lastAngle = initialAngle;
  lastX = initialX;
  lastSpeed = 0;
  lastAngularSpeed = 0;
  normalForce = (cartMass + bobMass) * 9.8
 ];

(*public methods*)
(*--------------------------------------*)
self@makeStep[delT_] := Module[{currentSolution, initialConditions, effectiveForce},

  initialConditions = {x[0] == lastX, x'[0] == lastSpeed, θ'[0] == lastAngularSpeed, θ[0] == lastAngle};
  lastAppliedForce = -gain. stateControlExpression /.
    {x[t] → lastX, θ[t] -> lastAngle, x'[t] → lastSpeed, θ'[t] → lastAngularSpeed};

  currentSolution =
   First@If[Abs[lastSpeed] ≤ $MachineEpsilon,
     (
      effectiveForce = If[Abs[lastAppliedForce] < normalForce * staticFriction,
        0,
        -(gain.stateControlExpression) - normalForce * staticFriction * Sign[lastAppliedForce]
       ];

      NDSolve[Join[eqns /. f[t] → (effectiveForce * UnitStep[t]),
        initialConditions], {x, θ, x', θ'}, {t, 0, delT}]
     ),
     (
      lastCoulombForce = -kineticFriction * normalForce * Sign[lastSpeed];
      lastViscousForce = -viscousFriction * lastSpeed;

      NDSolve[Join[eqns /. f[t] → (-(gain.stateControlExpression) + lastCoulombForce * UnitStep[t]),
        initialConditions], {x, θ, x', θ'}, {t, 0, delT}]
     )
    ];

  lastSpeed = (x' /. currentSolution)[delT];
  lastAngle = (θ /. currentSolution)[delT];
```

```
    lastX = (x /. currentSolution)[delT];
    lastAngularSpeed = (θ' /. currentSolution)[delT]
   ];

  self@reset[] := (init[]);
  self@setStaticFriction[v_] := (staticFriction = v; init[]);
  self@setKineticFriction[v_] := (kineticFriction = v; init[]);
  self@setViscousFriction[v_] := (viscousFriction = v; init[]);

  self@setCartMass[v_] := (cartMass = v; init[]);
  self@setBobMass[v_] := (bobMass = v; init[]);
  self@setPendulumLength[v_] := (pendulumLen = v; init[]);
  self@setInitialAngle[v_] := (initialAngle = v; init[]);
  self@setInitialX[v_] := (initialX = v; init[]);

  self@setPositionWeight[v_] := (positionWeight = v; init[]);
  self@setLinearVelocityWeight[v_] := (linearVelocityWeight = v; init[]);
  self@setAngleWeight[v_] := (angleWeight = v; init[]);
  self@setAngluarVelocityWeight[v_] := (angluarVelocityWeight = v; init[]);

  (*----------------------------------------*)
  self@getCurrentPosition[] := ({lastX, lastAngle, lastSpeed, lastAngularSpeed});
  self@getGain[] := gain;
  self@getFrictionType[] := (
    Text@Style[If[Abs[lastSpeed] ≤ $MachineEpsilon, "static", "kinetic"], 11]);
  self@getStateSpace[] := model;
  self@getAppliedForce[] := lastAppliedForce;
  self@getCoulombForce[] := lastCoulombForce;
  self@getViscousForce[] := lastViscousForce;
  self@getPendulumLength[] := pendulumLen;
  self@getClosedLoopEigenvalues[] := closedLoopEigenvalues;

  (*constructor*)
  init[];
  self
 ];

(*----------------------------------------*)
displayClass[$simulationTime_, $delT_, $initialAngle_] :=
 Module[{simulationTime = $simulationTime, delT = $delT, initialAngle = $initialAngle, xyData,
   currentIndex, init, makePolesAndZerosCoordinates, polesPlot, getStatistics, self},
  (*----------------------------------------*)
  init[] := (
    xyData = Table[{0, 0, 0, 0, 0}, {Floor[simulationTime / delT] + 1}];
    currentIndex = 0;
    xyData[[1, 4]] = initialAngle * Pi / 180.
   );
  (*----------------------------------------*)
  makePolesAndZerosCoordinates[points_] := Module[{xy},
    xy = Flatten[Replace[points, {Complex[x_, y_] ↦ {x, y}, x_?NumericQ ↦ {x, 0}}, {3}], 1];
    Cases[xy, {_?NumericQ, _?NumericQ}, {2}]
   ];
  (*----------------------------------------*)
  polesPlot[poles_, plotStyle_] := Module[{polesPoints, plotOptions},

    plotOptions = {AxesOrigin → {0, 0},
      ImagePadding → {{20, 20}, {20, 0}},
      Frame → True,
      ImageMargins → 1, AspectRatio → 1};

    polesPoints = makePolesAndZerosCoordinates[poles];
```

```
  ListPlot[polesPoints,
   AxesOrigin → {0, 0},
   PlotMarkers → Style["⊗", plotStyle, 11],
   GridLines → Automatic, GridLinesStyle -> Directive[Dashed, Thickness[.001], LightGray],
   Evaluate@plotOptions,
   PlotRange → {
     { Min[-1.2, 1.2 * Min[polesPoints[[All, 1]] ] ], Max[1.2, 1.2 * Max[polesPoints[[All, 1]]] ]},
     { Min[-1.2, 1.2 * Min[polesPoints[[All, 2]] ] ], Max[1.2, 1.2 * Max[polesPoints[[All, 2]]] ]}
    },
   ImageSize → {0.4 ContentSizeW, 0.4 ContentSizeH},
   ImageMargins → 0,
   PlotLabel -> Text[Row[{"[", Style["A-BK", Italic, 12], Style["] eigenvalues", 12]}]]
  ]
 ];
(*----------------------------------------*)
self@reset[$simulationTime2_, $delT2_, $initialAngle2_] :=
 (
  simulationTime = $simulationTime2;
  delT = $delT2;
  initialAngle = $initialAngle2;
  init[]
 );
(*----------------------------------------*)
self@addPoint[x_, xSpeed_, angle_, angleSpeed_, currentTime_] :=
 (
  currentIndex++;
  xyData[[currentIndex]] = {currentTime, x, xSpeed, angle * 180 / Pi, angleSpeed}

 );
(*----------------------------------------*)
getStatistics[] := Module[{c},
  If[currentIndex == 0, c = 1, c = currentIndex];

  Grid[{
    {Text@Style[Row[{"time = ", padIt2[xyData[[c, 1]], {4, 2}], " sec"}], 12],
     Text[Row[{Style["x", Italic, 12], " = ",
        Style[padIt1[xyData[[c, 2]], {4, 3}], 12], " ", Style["m", Italic, 12]}]],
     Text@Style[Row[{"θ = ", padIt2[xyData[[c, 4]], {4, 2}], " deg"}], 12]
    },
    {
     Text[
      Row[{Style["x'(t)", Italic, 12], " = ", padIt1[xyData[[c, 3]], {5, 4}], Style[" m/sec", 12]}]],
     Text[Style[Row[{"θ '(t) = ", padIt1[xyData[[c, 5]], {5, 4}], " rad/sec"}], 12]],
     ""
    }
    ,
    {Text[Row[{Style["K", Italic, 12], Style[" (state gain vector) = ", 12],
        TraditionalForm@NumberForm[Style[invertedPendulum@getGain[], 11], {5, 2}, SignPadding →
           True, NumberPadding → {"0", "0"}, NumberSigns → {"-", "+"}]}]], SpanFromLeft
    },

    {Text[Row[{Style["f", Italic, 12], " = ", Style[
         padIt1[invertedPendulum@getAppliedForce[], {5, 3}], 12], " ", Style["N", Italic, 12]}]],
     Text[Row[{Style["Fc ", Italic, 12], " = ", Style[padIt1[invertedPendulum@
           getCoulombForce[], {5, 3}], 12], " ", Style["N", Italic, 12]}]],
     Text[Row[{Style["Fv", Italic, 12], " = ", Style[padIt1[invertedPendulum@
           getViscousForce[], {5, 3}], 12], " ", Style["N", Italic, 12]}]]
    }
   }, Frame → All, Spacings → {0.5, .2}, FrameStyle -> Directive[Thickness[.001], Gray]
  ]
 ];
(*----------------------------------------*)
self@getPlot[] := Module[{g0, g1, g2, g3, g4, c, len},
```

```
g0 = polesPlot[invertedPendulum@getClosedLoopEigenvalues[], Blue];
len = invertedPendulum@getPendulumLength[];
If[currentIndex == 0, c = 1, c = currentIndex];
g1 = Graphics[
  {
   {Blue, Rectangle[{xyData[[c, 2]] - 1/2, 0}, {xyData[[c, 2]] + 1/2, 3/15}]},
   {
    Red,
    Line[{
      {xyData[[c, 2]], 3/15},
      {xyData[[c, 2]] + len*Cos[xyData[[c, 4]]*Pi/180], 3/15 + len*Sin[xyData[[c, 4]]*Pi/180]}
     }]
   },

   {Disk[{xyData[[c, 2]] + len*Cos[xyData[[c, 4]]*Pi/180],
      3/15 + len*Sin[xyData[[c, 4]]*Pi/180]}, .15]}
  },
  PlotRange → {{-4, 4}, {-0.5, 1.4*len}},
  ImageSize → {0.4 ContentSizeW, 0.4 ContentSizeH},
  AspectRatio → 1,
  Frame → True,
  Axes → True,
  AxesStyle → Dashed,
  FrameLabel → {{None, None}, {Style["x", Italic, 12], None}},
  ImagePadding → {{22, 5}, {33, 5}},
  ImageMargins → 0
 ];

g2 = ListPlot[xyData[[1 ;; c, 1 ;; 2]],
  PlotRange → {{0, simulationTime}, All},
  Joined → True,
  ImagePadding → {{40, 10}, {36, 35}},
  Frame → True,
  ImageSize → {0.51 ContentSizeW, 0.43 ContentSizeH}, FrameLabel →
   {{None, None}, {Text@Style["time (sec)", 12], Text@Style[" cart position vs. time", 12]}},
  ImageMargins → 0,
  GridLines → Automatic, GridLinesStyle -> Directive[Dashed, Thickness[.001], LightGray],
  PlotStyle → Red,
  AspectRatio → 1,
  Axes → None,
  Epilog → {Dashed, Thin, Line[{{0, 0}, {simulationTime, 0}}]}
 ];

g3 = ListPlot[xyData[[1 ;; c, {1, 4}]],
  PlotRange → {{0, simulationTime}, All},
  Joined → True,
  ImagePadding → {{40, 10}, {36, 35}},
  Frame → True,
  ImageSize → {0.51 ContentSizeW, 0.43 ContentSizeH}, FrameLabel →
   {{None, None}, {Text@Style["time (sec)", 12], Text@Style[" bob angle vs. time", 12]}},
  ImageMargins → 0,
  GridLines → Automatic, GridLinesStyle -> Directive[Dashed, Thickness[.001], LightGray],
  PlotStyle → Red,
  AspectRatio → 1,
  Axes → None,
  Epilog → {Dashed, Thin, Line[{{0, 90}, {simulationTime, 90}}]}
 ];

g4 = getStatistics[];

Grid[{{g4, SpanFromLeft}, {g0, g1}, {g2, g3}}, Spacings → {0, 0},
 Alignment → Center, Frame → All, FrameStyle -> Directive[Thickness[.001], Gray]]
];
```

```
    init[];
    self
   ];

  invertedPendulum =
   invertedPendulumClass[1, 10, 2, 0.1, 0.03, 3, 75*Pi/180., 1, 0.05, 100, 10, 10, 1, x, θ, t];
  display = displayClass[10, 0.05, 5];
 }
]
```

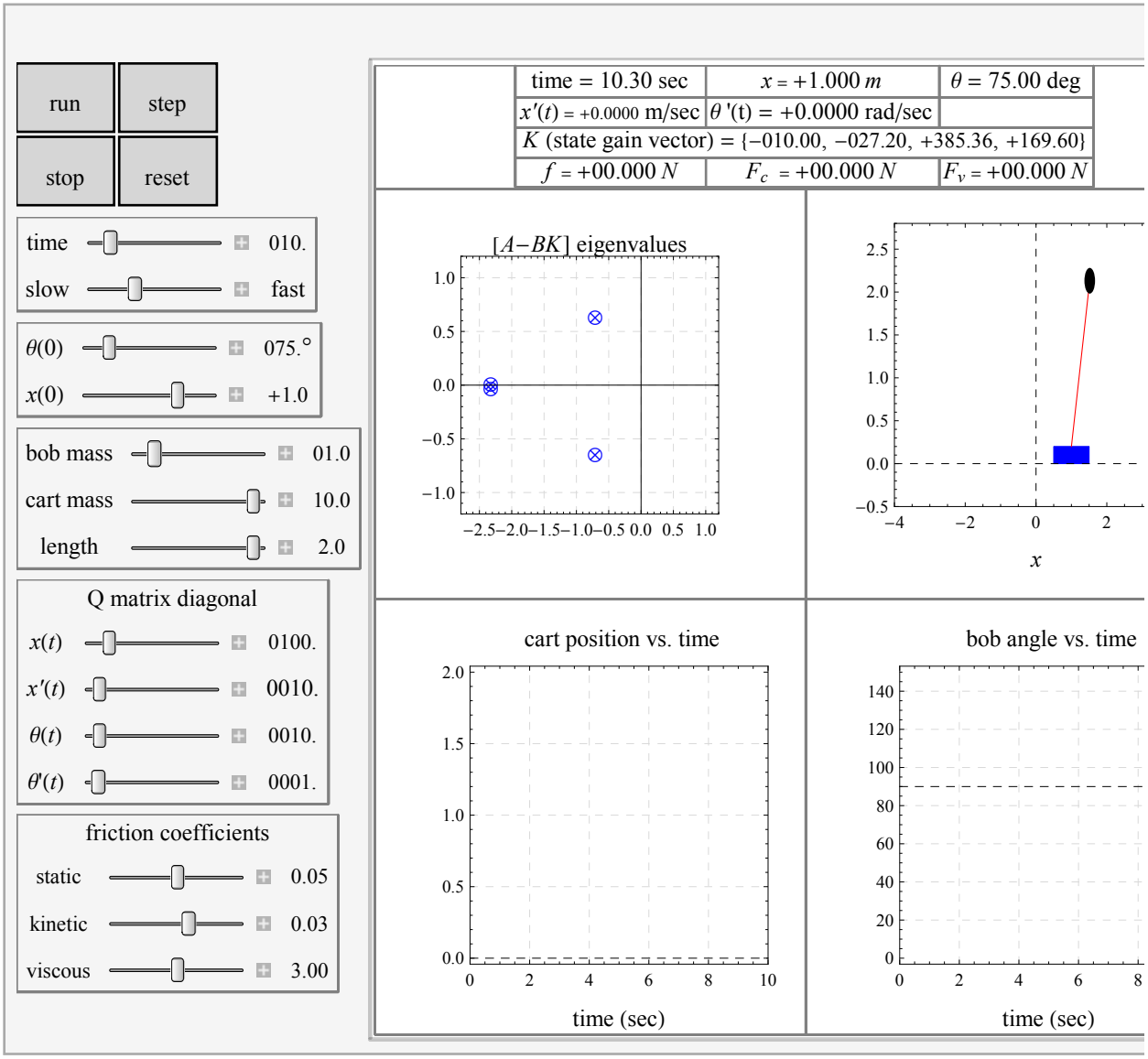


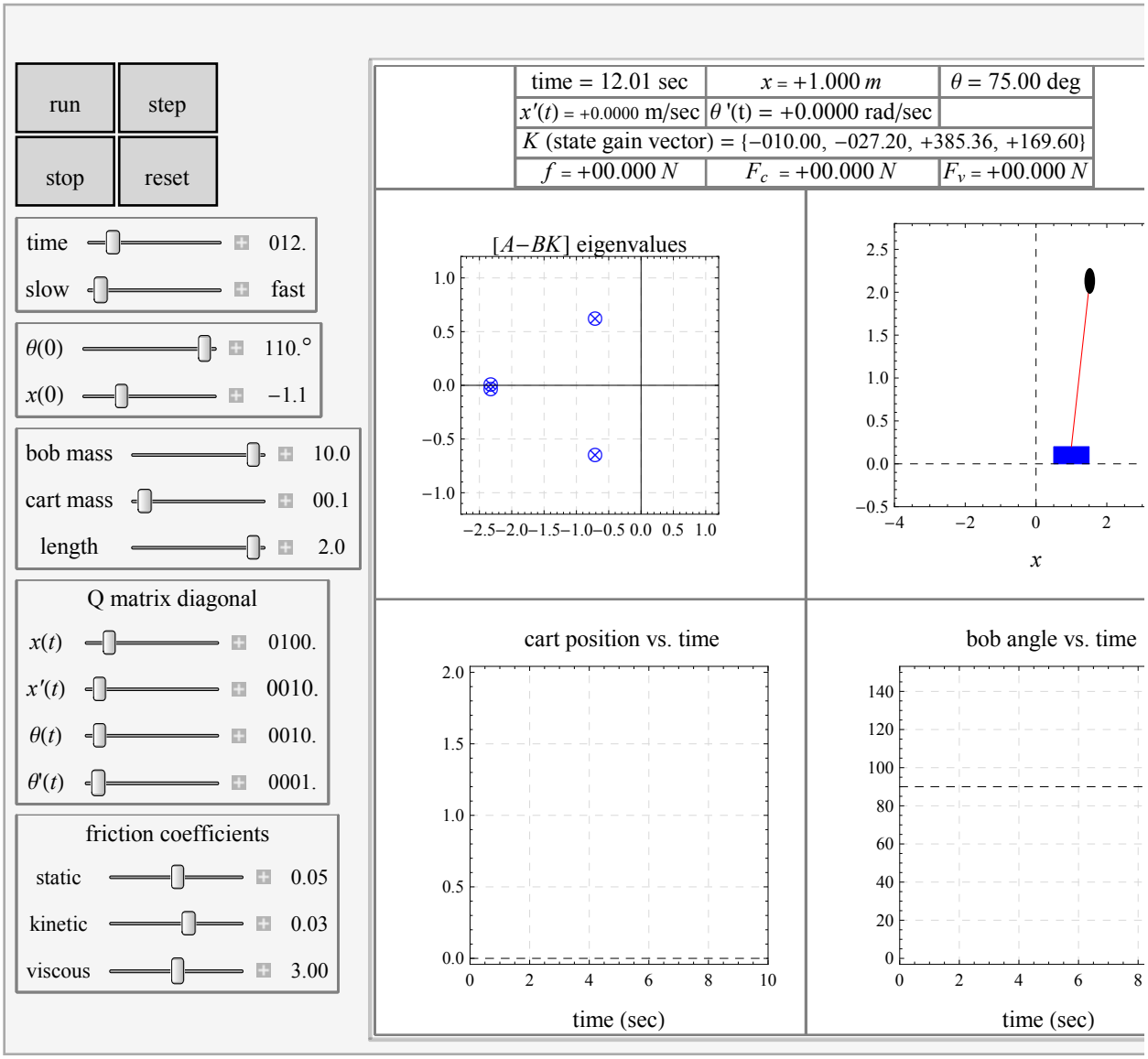
Caption

The linear quadratic regulator (LQR) method is used to generate a control force that brings an inverted pendulum from an initial condition back to the upright position in an optimal way. The state space $x'(t) = A\,x(t) + B\,u(t)$ is used to represent the dynamics of the system. Static and Coulomb friction forces act as external disturbances. Coulomb friction causes an oscillation of the cart position around the equilibrium position ($x = 0$). When only viscous friction is present, LQR brings the pendulum to the upright position since viscous friction is included in the A state matrix, while Coulomb friction is included in neither the A nor the B matrix. A standard friction

model is used and is described below. The eigenvalues of the closed loop state matrix $A - B K$ (where K is the gain vector generated by LQR) are all located in the left side of the complex plane, showing that the resulting system is stable.

Thumbnail





Details (optional)

Let M be the mass of the cart, m the mass of the pendulum bob, and L the length of the pendulum. The kinetic energy of the system is $\mathrm{KE} = \frac{1}{2} M\, x'(t)^2 + \frac{1}{2} m \left(x'(t)^2 - 2\, L\, \theta'(t) \sin\theta(t)\, x'(t) + L^2\, \theta'(t)^2\right)$ and the potential energy is $\mathrm{PE} = m\, g\, L \sin\theta(t)$. Hence the Lagrangian is $\Delta = \mathrm{KE} - \mathrm{PE}$ and the equation of motion for the cart is $\frac{d}{dt}\left(\frac{d\Delta}{dx'(t)}\right) - \frac{d\Delta}{dx} = f(t) - F_f$, where $f(t)$ is the applied force and F_f is the force due to friction. The equation of motion for the bob mass is $\frac{d}{dt}\left(\frac{d\Delta}{d\theta'(t)}\right) - \frac{d\Delta}{d\theta(t)} = 0$.

The friction model used is the following: Let μ_s, μ_c, μ_v be the static, Coulomb (kinetic), and viscous friction coefficients, respectively. Let F_N be the normal force, which is $(M + m)\, g$. When the speed of the cart is zero and $|f(t)| < \mu_s F_N$ then the friction force is $F_f = -f(t)$, and when $|f(t)| \geq \mu_s F_N$, then the friction force $F_f = -\mu_s F_N \operatorname{sign}(f(t))$. When the speed of the cart is not zero, then $F_f = -\mu_c F_N \operatorname{sign}(x'(t)) - \mu_v x'(t)$. The value of μ_c is normally less than μ_s. You can use the sliders to change the values of these coefficients.

The applied force $f(t)$ is found by using the LQR (linear quadratic regulator) method. This force is applied in order to bring the cart to the $x = 0$ position with the pendulum in the upright position.

The table at the top of the display shows simulation information. The field labeled "f" is the applied force (the state feedback control force found by LQR), the field labeled "F_c" is the Coulomb friction force, and the field labeled "F_v" is viscous force, with all units in

Newton. The sign indicates the direction of the force at that moment of the simulation.
You can adjust the weights used by LQR by adjusting the slides that represent the entries in the Q matrix diagonal. The documentation for *Mathematica*'s built-in function `LQRegulatorGains` (link below) explain more about the Q matrix.
You can change the initial angle position of the inverted pendulum and the initial cart position using the sliders.

For more information on the derivation of the equations of motion see the author's report on inverted pendulum.

[1] R. Soutas-Little and D. Inman, *Engineering Mechanics Dynamic*, New Jersey, Prentice–Hall, 1999.

[2] D. Guida, F. Nilvetti, and C. M. Pappalardo, "Dry Friction of Bearings on Dynamics and Control of an Inverted Pendulum," *Journal of Achievements in Materials and Manufacturing Engineering*, **38**(1), January 2010.
[3] S. Campbell, S. Crawford, and K. Morris, "Friction and the Inverted Pendulum Stabilization Problem," *Journal of Dynamic Systems, Measurement, and Control*, **130**(5), September 2008, 054502.

Control Suggestions (optional)

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

Search Terms (optional)

inverted pendulum
friction
viscous
Coulomb
LQRegulatorGains

Related Links (optional)

LQRegulatorGains
stabilized-inverted-pendulum

Authoring Information

Contributed by: Nasser M. Abbasi