Package Description

In[973]:=
```
(*Mathematica Package*)

BeginPackage["cartPendulum`"]

(*Exported symbols added here with SymbolName::usage*)

ffCartPendulumGeneral::usage = "
ffCartPendulumGeneral[n_,τ_,τ1_,A_]
Computes the feedforward state and
    costate trajectories for swinging up the pendulum optimally
{xff,xdotff,θff,θdotff,uff}
";

TestSwingUpGeneral::usage = "
TestSwingUpGeneral[τ_,τ1_,uff0_,A_]
Test the approximate solution on the open-loop dynamics
{xs,θs,xdots,θdots}
";

TestSwingUpGeneralFB::usage = "
TestSwingUpGeneralFB[τ_,τ1_,xff0_,xdotff0_,θff0_,θdotff0_,uff0_,A_]
Add Linear feedback only at the end
{xs,θs,us}
";

CalculateGains::usage = "
CalculateGains[xff0_,xdotff0_,θff0_,θdotff0_,uff0_,A_]
Calculate the gain matrix by solving the quasi stationary Ricatti Equation
K
";

TestSwingUpGeneralFBNumeric::usage = "
TestSwingUpGeneralFBNumeric[τ_,τ1_,xff0_,xdotff0_,θff0_,θdotff0_,uff0_,A_,KTable_,n_]
Simulates the true dynamics with feedback added according to Ktable
{xs,θs,us,K1,K2,K3,K4}
";

SimulateLinearFeedbackEnd::usage = "
SimulateLinearFeedbackEnd[ICs_,n_,τ_,τ1_,A_]
Outputs plots directly using TestSwingUpGeneralFB
Grid[{{p1a,p1b,p1c}}]
";
```

```
SimulateLQRFeedback::usage = "
SimulateLQRFeedback[ICs_,n_,n2_,τ_,τ1_,A_]
Outputs plots directly using TestSwingUpGeneralFBNumeric
Grid[{{p1a,p1b,p1c}}]
";



(*Begin Private Context*)
Begin["`Private`"]

(*ICs - Initial Conditions *)
ffCartPendulumGeneral[ICs_, n_, τ_, τ1_, A_] :=
 Module[{x, xdot, f, θ, θdot, λ1, λ2, λ3, λ4, Δt, bcs, eqns, sv, froot, xff,
   xdotff, xff0, xdotff0, θff0, θdotff0, uff0, θff, θdotff, uff}, Δt = τ/n;
  f[{x_, xdot_, θ_, θdot_, λ1_, λ2_, λ3_, λ4_}] := {xdot,
    1/(1 - A Cos[θ]^2) (A θdot^2 Sin[θ] + 1/(1 - A Cos[θ]^2) (λ4 Cos[θ] - λ2) + A Cos[θ] Sin[θ]), θdot,
    1/(1 - A Cos[θ]^2) (-1/(1 - A Cos[θ]^2) (-λ2 Cos[θ] + λ4 Cos[θ]^2) - Sin[θ] - A θdot^2 Cos[θ] Sin[θ]),
    0, -λ1, 2/(A Cos[2 θ] + A - 2)^3 (Cos[θ] (4 Sin[θ] (A λ4^2 Cos[2 θ] + 4 A λ2^2 + (A + 2) λ4^2) -
          (A Cos[2 θ] - 3 A + 2) (A Cos[2 θ] + A - 2) (A θdot^2 λ2 - λ4)) + A ((A - 2) Cos[2 θ] + A)
        (A Cos[2 θ] + A - 2) (λ2 - θdot^2 λ4) - 4 λ2 λ4 Sin[θ] (3 A Cos[2 θ] + 3 A + 2)),
    4/(A Cos[2 θ] + A - 2) (A θdot Sin[θ] (λ2 - λ4 Cos[θ])) - λ3};
  bcs = {x_0 == ICs[[1]], xdot_0 == ICs[[2]],
    x_n == xdot_n == 0, θ_0 == ICs[[3]], θdot_0 == ICs[[4]], θdot_n == 0, θ_n == π};
  eqns = Flatten[Join[bcs, Table[Thread[{x_i, xdot_i, θ_i, θdot_i, λ1_i, λ2_i, λ3_i, λ4_i} ==
        1/2 Δt (f[{x_(i-1), xdot_(i-1), θ_(i-1), θdot_(i-1), λ1_(i-1), λ2_(i-1), λ3_(i-1), λ4_(i-1)}] +
            f[{x_i, xdot_i, θ_i, θdot_i, λ1_i, λ2_i, λ3_i, λ4_i}]) +
         {x_(i-1), xdot_(i-1), θ_(i-1), θdot_(i-1), λ1_(i-1), λ2_(i-1), λ3_(i-1), λ4_(i-1)}], {i, 1, n}]]];
  sv = Flatten[Table[{{x_i, ICs[[1]]}, {xdot_i, ICs[[2]]}, {θ_i, ICs[[3]]},
      {θdot_i, ICs[[4]]}, {λ1_i, 0}, {λ2_i, 0}, {λ3_i, 0}, {λ4_i, 0}}, {i, 0, n}], 1];
  froot = FindRoot[eqns, sv];
  xff0 = ListInterpolation[Table[x_i, {i, 0, n}] /. froot, {0, τ}, InterpolationOrder → 1];
  xdotff0 =
   ListInterpolation[Table[xdot_i, {i, 0, n}] /. froot, {0, τ}, InterpolationOrder → 1];
  θff0 = ListInterpolation[Table[θ_i, {i, 0, n}] /. froot, {0, τ}, InterpolationOrder → 1];
```

```
  θdotff0 =
   ListInterpolation[Table[θdot_i, {i, 0, n}] /. froot, {0, τ}, InterpolationOrder → 1];
  uff0 = ListInterpolation[Table[1/(1 - A Cos[θ_i]^2) (λ4_i Cos[θ_i] - λ2_i), {i, 0, n}] /. froot,
    {0, τ}, InterpolationOrder → 1];

  xff[t_] := Piecewise[{{xff0[t], 0 ≤ t ≤ τ}}, 0];
  xdotff[t_] := Piecewise[{{xdotff0[t], 0 ≤ t ≤ τ}}, 0];
  θff[t_] := Piecewise[{{θff0[t], 0 ≤ t ≤ τ}}, π];
  θdotff[t_] := Piecewise[{{θdotff0[t], 0 ≤ t ≤ τ}}, 0];
  uff[t_] := Piecewise[{{uff0[t], 0 ≤ t ≤ τ}}, 0];

  {xff, xdotff, θff, θdotff, uff}]

TestSwingUpGeneral[ICs_, τ_, τ1_, uff0_, A_] :=
 Module[{eq, init, x, xdot, θ, θdot, xs, xdots, θs, θdots, t, J},
  eq = {x'[t] == xdot[t],
    xdot'[t] == 1/(1 - A Cos[θ[t]]^2) (uff0[t] + A θdot[t]^2 Sin[θ[t]] + A Cos[θ[t]] Sin[θ[t]]),
    θ'[t] == θdot[t], θdot'[t] ==
     1/(1 - A Cos[θ[t]]^2) (-Sin[θ[t]] - Cos[θ[t]] (uff0[t] + A θdot[t]^2 Sin[θ[t]]))};
  init = {x[0] == ICs⟦1⟧, xdot[0] == ICs⟦2⟧, θ[0] == ICs⟦3⟧, θdot[0] == ICs⟦4⟧};
  {xs, xdots, θs, θdots} = NDSolveValue[{eq, init},
    {x, xdot, θ, θdot}, {t, 0, τ1}, Method → {"DiscontinuityProcessing" → None}];
  J = NIntegrate[uff0[t]^2, {t, 0, τ}];
  {xs, xdots, θs, θdots, uff0, J}]

TestSwingUpGeneralFB[ICs_, τ_, τ1_, xff0_, xdotff0_, θff0_, θdotff0_, uff0_, A_] :=
 Module[{eq, init, θ, θdot, θff, θdotff, x, xdot, xff, xdotff,
   uff, t, κ1, κ2, κ3, κ4, ufb, u, θs, θdots, xs, xdots, us, J},
  κ1 = κ2 = 3;  (* lqr for q=r for balancing pendulum *)
  κ3 = -0.1; κ4 = -0.65;
  xff[t_] := Piecewise[{{xff0[t], 0 ≤ t ≤ τ}}, 0];
  xdotff[t_] := Piecewise[{{xdotff0[t], 0 ≤ t ≤ τ}}, 0];
  θff[t_] := Piecewise[{{θff0[t], 0 ≤ t ≤ τ}}, π];
  θdotff[t_] := Piecewise[{{θdotff0[t], 0 ≤ t ≤ τ}}, 0];
  uff[t_] := Piecewise[{{uff0[t], 0 ≤ t ≤ τ}}, 0];
  ufb[t_]  := Piecewise[{{0, 0 ≤ t ≤ τ}}, κ1 (θff[t] - θ[t]) +
     κ2 (θdotff[t] - θdot[t]) + κ3 (xff[t] - x[t]) + κ4 (xdotff[t] - xdot[t])];
  u[t_] := uff[t] + ufb[t];
  eq = {x'[t] == xdot[t], xdot'[t] ==
```

```
    1/(1 - A Cos[θ[t]]^2) (u[t] + A θdot[t]^2 Sin[θ[t]] + A Cos[θ[t]] Sin[θ[t]]), θ'[t] == θdot[t],
   θdot'[t] == 1/(1 - A Cos[θ[t]]^2) (-Sin[θ[t]] - Cos[θ[t]] (u[t] + A θdot[t]^2 Sin[θ[t]]))};
  init = {x[0] == ICs[[1]], xdot[0] == ICs[[2]], θ[0] == ICs[[3]], θdot[0] == ICs[[4]]};
  {xs, xdots, θs, θdots} = NDSolveValue[{eq, init},
    {x, xdot, θ, θdot}, {t, 0, τ1}, Method → {"DiscontinuityProcessing" → None}];
  us[t_] := uff[t] + Piecewise[{{0, 0 ≤ t ≤ τ}}, κ1 (θff[t] - θs[t]) +
     κ2 (θdotff[t] - θdots[t]) + κ3 (xff[t] - xs[t]) + κ4 (xdotff[t] - xdots[t])];
  J = NIntegrate[us[t]^2, {t, 0, τ}];
  {xs, xdots, θs, θdots, us, J}]

CalculateGains[xff0_, xdotff0_, θff0_, θdotff0_, uff0_, A_] :=
 Module[{x, L, RHS, xdot, θ, θdot, u, K, S, soltn, i, j, s11, s12, s13, s14, s22,
   s23, s24, s33, s34, s44, Af, Bf, Q, fx, xState, ric, R, Mf, x2dot, θ2dot},
  xState = {x, xdot, θ, θdot};
  x2dot = 1/(1 - A Cos[θ]^2) (u + A θdot^2 Sin[θ] + A Cos[θ] Sin[θ]);
  θ2dot = 1/(1 - A Cos[θ]^2) (-Sin[θ] - Cos[θ] (u + A θdot^2 Sin[θ]));
  fx = {xdot, x2dot, θdot, θ2dot};
  L = 1 / 2 * u^2;
  Af = Grad[fx, xState]; (* For nD stuff use Grad*)
  Bf = D[fx, u] ; (*For 1D stuff use D*)
  Q = Grad[Grad[L, xState], xState];
  Mf = Grad[D[L, u], xState];
  R = D[L, {u, 2}];
  S = ( s11 s12 s13 s14
        s12 s22 s23 s24
        s13 s23 s33 s34
        s14 s24 s34 s44 );
  ric = Q +  Af^T.S + S.Af - Outer[Times, S.Bf, Bf^T.S ] ;
  (* This is the Syntax for calculating Outer Products *)(*Q = I, M = 0, R = 1*)
  RHS = Table[0, {i, 4}, {j, 4}];
  x = xff0;
  xdot = xdotff0;
  θ = θff0;
  θdot = θdotff0;
  u = uff0; (* Entering State Values *)
  soltn = NMinimize[{1, ric == RHS}, {s11, s12, s13, s14, s22, s23, s24, s33, s34, s44}][[2]];
  S = S /. soltn;
  K = Bf^T.S ;
  K]
```

```
TestSwingUpGeneralFBNumeric[ICs_, τ_, τ1_,
  xff0_, xdotff0_, θff0_, θdotff0_, uff0_, A_, KTable_, n_] :=
 Module[{K1, K2, K3, K4, eq, init, θ, θdot, θff, θdotff, x, xdot, xff, xdotff,
   uff, t, ufb, u, θs, θdots, xs, xdots, us, κ1, κ2, κ3, κ4, J},
  κ1 = κ2 = 3;  (* lqr for q=r for balancing pendulum *)
  κ3 = -0.1; κ4 = -0.65;
  K1[t_] :=
   Piecewise[Table[{KTable〚i〛〚1〛, (i - 1) * τ / n ≤ t ≤ i * τ / n}, {i, 1, n}], KTable〚n〛〚1〛];
  K2[t_] := Piecewise[Table[{KTable〚i〛〚2〛, (i - 1) * τ / n ≤ t ≤ i * τ / n}, {i, 1, n}],
    KTable〚n〛〚2〛];
  K3[t_] := Piecewise[Table[{KTable〚i〛〚3〛, (i - 1) * τ / n ≤ t ≤ i * τ / n}, {i, 1, n}],
    KTable〚n〛〚3〛];
  K4[t_] := Piecewise[Table[{KTable〚i〛〚4〛, (i - 1) * τ / n ≤ t ≤ i * τ / n}, {i, 1, n}],
    KTable〚n〛〚4〛];
  xff[t_] := Piecewise[{{xff0[t], 0 ≤ t ≤ τ}}, 0];
  xdotff[t_] := Piecewise[{{xdotff0[t], 0 ≤ t ≤ τ}}, 0];
  θff[t_] := Piecewise[{{θff0[t], 0 ≤ t ≤ τ}}, π];
  θdotff[t_] := Piecewise[{{θdotff0[t], 0 ≤ t ≤ τ}}, 0];
  uff[t_] := Piecewise[{{uff0[t], 0 ≤ t ≤ τ}}, 0];
  ufb[t_] := Piecewise[
    {{K3[t] * (θff[t] - θ[t]) + K4[t] * (θdotff[t] - θdot[t]) + K1[t] * (xff[t] - x[t]) +
       K2[t] * (xdotff[t] - xdot[t]), 0 ≤ t ≤ τ}}, κ1 (θff[t] - θ[t]) +
     κ2 (θdotff[t] - θdot[t]) + κ3 (xff[t] - x[t]) + κ4 (xdotff[t] - xdot[t])];
  u[t_] := uff[t] + ufb[t];
  eq = {x'[t] == xdot[t], xdot'[t] ==
     1/(1 - A Cos[θ[t]]^2) (u[t] + A θdot[t]^2 Sin[θ[t]] + A Cos[θ[t]] Sin[θ[t]]), θ'[t] == θdot[t],
    θdot'[t] == 1/(1 - A Cos[θ[t]]^2) (-Sin[θ[t]] - Cos[θ[t]] (u[t] + A θdot[t]^2 Sin[θ[t]]))};
  init = {x[0] == ICs〚1〛, xdot[0] == ICs〚2〛, θ[0] == ICs〚3〛, θdot[0] == ICs〚4〛};
  {xs, xdots, θs, θdots} = NDSolveValue[{eq, init},
    {x, xdot, θ, θdot}, {t, 0, τ1}, Method → {"DiscontinuityProcessing" → None}];
  us[t_] :=
   uff[t] + Piecewise[{{K3[t] * (θff[t] - θs[t]) + K4[t] * (θdotff[t] - θdots[t]) + K1[t] *
          (xff[t] - xs[t]) + K2[t] * (xdotff[t] - xdots[t]), 0 ≤ t ≤ τ}}, κ1 (θff[t] - θs[t]) +
      κ2 (θdotff[t] - θdots[t]) + κ3 (xff[t] - xs[t]) + κ4 (xdotff[t] - xdots[t])];
  J = NIntegrate[us[t]^2, {t, 0, τ}];
  {xs, xdots, θs, θdots, us, K1, K2, K3, K4, J}]

SimulateLinearFeedbackEnd[ICs_, n_, τ_, τ1_, A_] := Module[{x1a, xdot1a, θ1a, θdot1a, u1a,
         x1b, xdot1b, θ1b, θdot1b, u1b,
         x1c, xdot1c, θ1c, θdot1c, u1c,
         p1a, p1b, p1c, J, J1},

  {x1a, xdot1a, θ1a, θdot1a, u1a} = ffCartPendulumGeneral[ICs, n, τ, τ1, A];
```

```
  {x1b, xdot1b, θ1b, θdot1b, u1b, J1} = TestSwingUpGeneral[ICs, τ, τ1, u1a, A];
  {x1c, xdot1c, θ1c, θdot1c, u1c, J} =
   TestSwingUpGeneralFB[ICs, τ, τ1, x1a, xdot1a, θ1a, θdot1a, u1a, A];

  p1a = Plot[{θ1a[t], u1a[t], x1a[t], θdot1a[t], xdot1a[t]},
    {t, 0, τ1}, Filling → {2 → Axis}, PlotRange → {-4, 4}, PlotLegends →
     {"θ1a", "u1a", "x1a", "θdot1a", "xdot1a"}, PlotLabel → "Feedforward solution",
    AspectRatio → 1 / 3, ImageSize → 400, GridLines → {None, {-π, π}}];

  p1b = Plot[{θ1b[t], u1a[t], x1b[t], θdot1b[t], xdot1b[t]},
    {t, 0, τ1}, PlotRange → {-4, 4}, Filling → {2 → Axis}, PlotLegends →
     {"θ1b", "u1b", "x1b", "θdot1b", "xdot1b"}, PlotLabel → "Test on dynamics",
    AspectRatio → 1 / 3, ImageSize → 400, GridLines → {None, {-π, π}}];

  p1c =
   Plot[{θ1c[t], u1c[t], x1c[t], θdot1c[t], xdot1c[t]}, {t, 0, τ1}, PlotRange → {-4, 4},
    Filling → {2 → Axis}, PlotLegends → {"θ1c", "u1c", "x1c", "θdot1c", "xdot1c"},
    PlotLabel → "Linear feedback solution", AspectRatio → 1 / 3,
    ImageSize → 400, GridLines → {None, {-π, π}}];

  {J1, J, p1a, p1b, p1c}]

SimulateLQRFeedback[ICs_, n_, n2_, τ_, τ1_, A_] := Module[{x1a, xdot1a, θ1a, θdot1a, u1a,
        x1b, xdot1b, θ1b, θdot1b, u1b,
        x1c, xdot1c, θ1c, θdot1c, u1c,
        p1a, p1b, p1c, KTable, K1, K2, K3, K4, J},

  {x1a, xdot1a, θ1a, θdot1a, u1a} = ffCartPendulumGeneral[ICs, n, τ, τ1, A];
  KTable = Table[CalculateGains[x1a[τ1 / n2 * i], xdot1a[τ1 / n2 * i],
     θ1a[τ1 / n2 * i], θdot1a[τ1 / n2 * i], u1a[τ1 / n2 * i], A], {i, 0, n2}];
  {x1b, xdot1b, θ1b, θdot1b, u1b, _} = TestSwingUpGeneral[ICs, τ, τ1, u1a, A];
  {x1c, xdot1c, θ1c, θdot1c, u1c, K1, K2, K3, K4, J} =
   TestSwingUpGeneralFBNumeric[ICs, τ, τ1, x1a, xdot1a, θ1a, θdot1a, u1a, A, KTable, n2];;

  p1a = Plot[{θ1a[t], u1a[t], x1a[t], θdot1a[t], xdot1a[t]},
    {t, 0, τ1}, Filling → {2 → Axis}, PlotRange → {-4, 4}, PlotLegends →
     {"θ1a", "u1a", "x1a", "θdot1a", "xdot1a"}, PlotLabel → "Feedforward solution",
    AspectRatio → 1 / 3, ImageSize → 400, GridLines → {None, {-π, π}}];

  p1b = Plot[{θ1b[t], u1a[t], x1b[t], θdot1b[t], xdot1b[t]},
    {t, 0, τ1}, PlotRange → {-4, 4}, Filling → {2 → Axis}, PlotLegends →
     {"θ1b", "u1b", "x1b", "θdot1b", "xdot1b"}, PlotLabel → "Test on dynamics",
    AspectRatio → 1 / 3, ImageSize → 400, GridLines → {None, {-π, π}}];

  p1c = Plot[{θ1c[t], u1c[t], x1c[t], θdot1c[t], xdot1c[t]},
    {t, 0, τ1}, PlotRange → {-4, 4}, Filling → {2 → Axis}, PlotLegends →
     {"θ1c", "u1c", "x1c", "θdot1c", "xdot1c"}, PlotLabel → "LQR feedback solution",
```

```
    AspectRatio → 1 / 3, ImageSize → 400, GridLines → {None, {-π, π}}];

  {J, p1a, p1b, p1c}]


End[] (*End Private Context*)

EndPackage[]
```

Out[973]= cartPendulum`

Out[981]= cartPendulum`Private`

Out[989]= cartPendulum`Private`

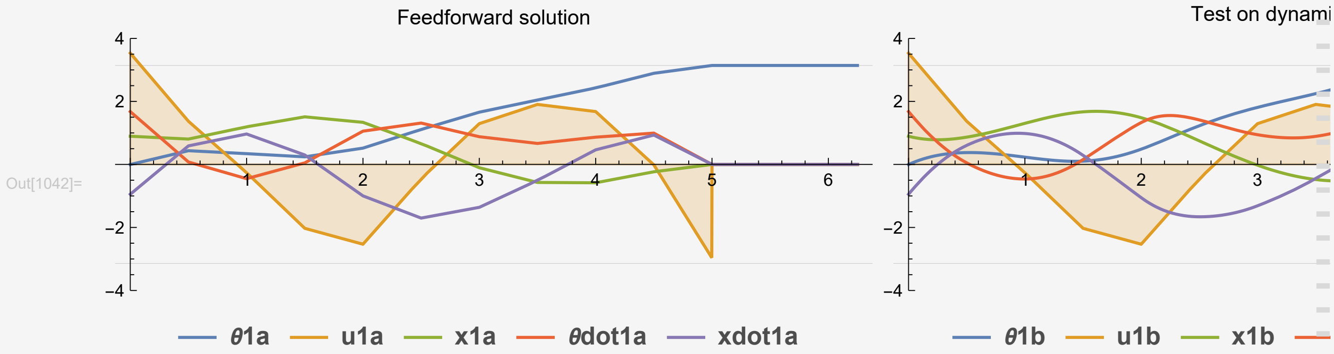
Testing Functions

```
(*
Use this to import the package. And use the following syntax to call the appropriate func
*)
Get["D:\\Dhruv\\MITACS_Summer_22\\Codes\\cartPendulum.m"]
```

In[1039]:=

```
n=10;τ=5;τ1= τ*1.25 ;A=0.2;
ICs = {0.89486028245609,-0.9468360111172656,-0.002994757534002989,1.677668990900959};
(*ICs = {0,0,0,0};*)
{J1,J,p1a,p1b,p1c} = SimulateLinearFeedbackEnd[ICs,n,τ,τ1,A];
Grid[{{p1a,p1b,p1c}}]
J1
J
```

Out[1042]=



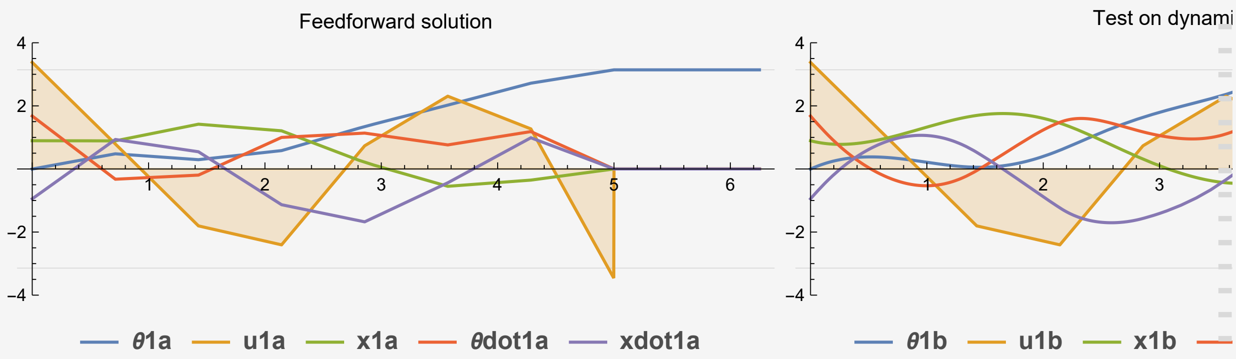
Out[1043]= 13.216

Out[1044]= 13.216

In[1057]:=
```
n=7;τ=5;τ1= τ*1.25 ;n2 = 40;
A=0.2; ICs = {0.89486028245609,-0.9468360111172656,-0.002994757534002989,1.6776689909009
{J,p1a,p1b,p1c} = SimulateLQRFeedback[ICs,n,n2,τ,τ1,A];
Grid[{{p1a,p1b,p1c}}]
J
```

Set::nosym: _ does not contain a symbol to attach a rule to.

Out[1060]=

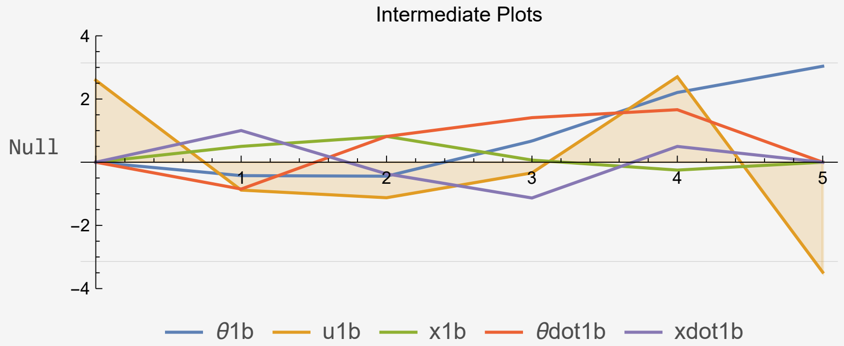


Out[1061]= 15.0371

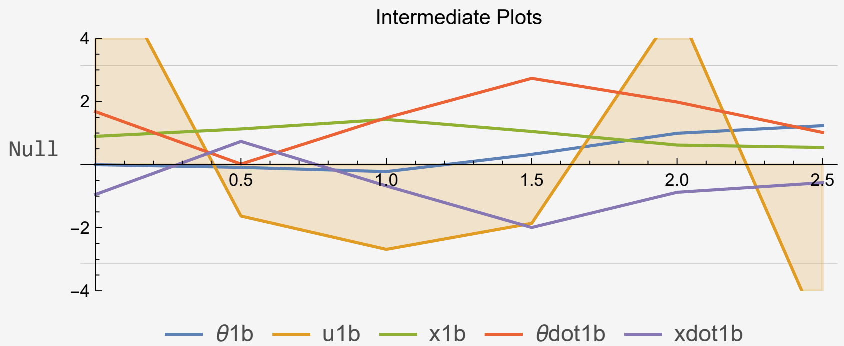
In[1095]:=
```
(* Repeated Computations function*) (*No feedback *)
n=5;τ=5;τ1= τ*1.25 ;n2 = 20; M = 2;
A=0.2; initialConditions = {0,0,0,0};
MyAppend[f1_,f2_,T1_,dT_]:= Module[{f},
f[t_] := Piecewise[{{f1[t],0≤ t ≤ T1},{f2[t-T1],T1≤ t≤ T1 + dT}}];
f]

ICs = initialConditions;
xs[t_] := 0;
xdots[t_] := 0;
θs[t_] := 0;
θdots[t_] := 0;
Js = 0;
For[i = 0, i < M, i++,
    {x1a,xdot1a,θ1a,θdot1a,u1a}=ffCartPendulumGeneral[ICs,n,τ*(1-i/M),τ1,A];
    Plot[{θ1a[t],u1a[t],x1a[t],θdot1a[t],xdot1a[t]},{t,0,τ*(1-i/M)},PlotRange→{-4,4},F
×
    If[i ≠ M-1,
    (*If condition is true*)
        {x1b,xdot1b,θ1b,θdot1b,u1b,J}=TestSwingUpGeneral[ICs,τ*1/M,τ*1/M,u1a,A];
        ICs = {x1b[τ*1/M],xdot1b[τ*1/M],θ1b[τ*1/M],θdot1b[τ*1/M]};
        xs = MyAppend[xs,x1b,τ*i/M,τ*1/M];
        xdots = MyAppend[xdots,xdot1b,τ*i/M,τ*1/M];
        θs = MyAppend[θs,θ1b,τ*i/M,τ*1/M];
        θdots = MyAppend[θdots,θdot1b,τ*i/M,τ*1/M];
        us = MyAppend[us,u1b,τ*i/M,τ*1/M];
        Js = Js + J;
    ,
    (*If condition is false*)
        {x1b,xdot1b,θ1b,θdot1b,u1b,J}=TestSwingUpGeneral[ICs,τ*1/M,τ*1/M + τ1 - τ,u1a,A
        xs = MyAppend[xs,x1b,τ*i/M,τ*1/M + τ1 - τ];
        xdots = MyAppend[xdots,xdot1b,τ*i/M,τ*1/M + τ1 - τ];
        θs = MyAppend[θs,θ1b,τ*i/M,τ*1/M + τ1 - τ];
        θdots = MyAppend[θdots,θdot1b,τ*i/M,τ*1/M + τ1 - τ];
        us = MyAppend[us,u1b,τ*i/M,τ*1/M + τ1 - τ];
    ]
//Print]
p1b=Plot[{θs[t],us[t],xs[t],θdots[t],xdots[t]},{t,0,τ1},PlotRange→{-4,4},Filling→{2→Ax
Js
```

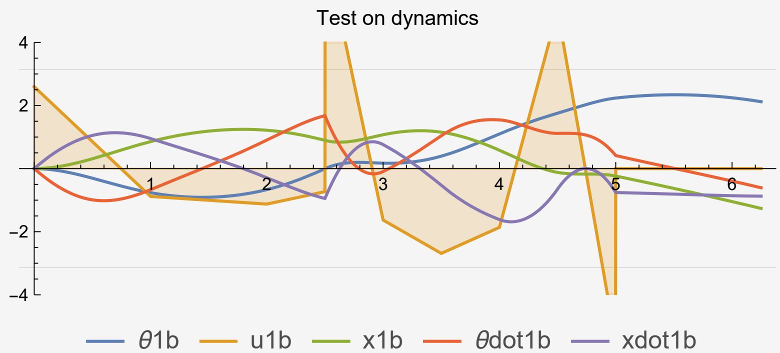
FindRoot::cvmit: Failed to converge to the requested accuracy or precision within 100 iterations.



FindRoot::cvmit: Failed to converge to the requested accuracy or precision within 100 iterations.



Out[1105]=



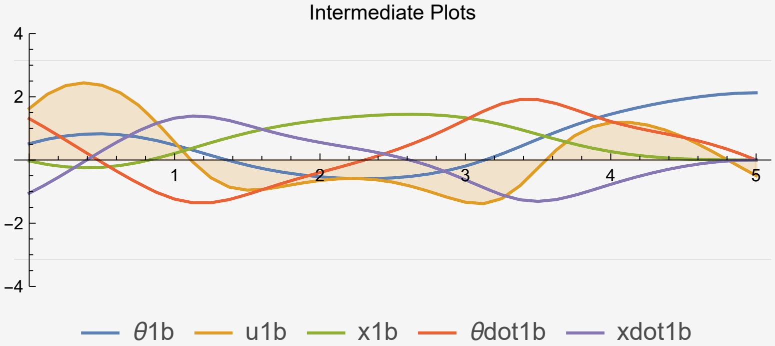
Out[1106]= 3.18855

In[939]:=
```
ICs
{x1a,xdot1a,θ1a,θdot1a,u1a}=ffCartPendulumGeneral[ICs,n,τ,τ1,A];
Plot[{θ1a[t],u1a[t],x1a[t],θdot1a[t],xdot1a[t]},{t,0,τ},PlotRange→{-4,4},Filling→{2→Ax
```

Out[939]= {-0.0458664, -1.55256, 0.761626, 1.93268}

FindRoot::cvmit: Failed to converge to the requested accuracy or precision within 100 iterations.

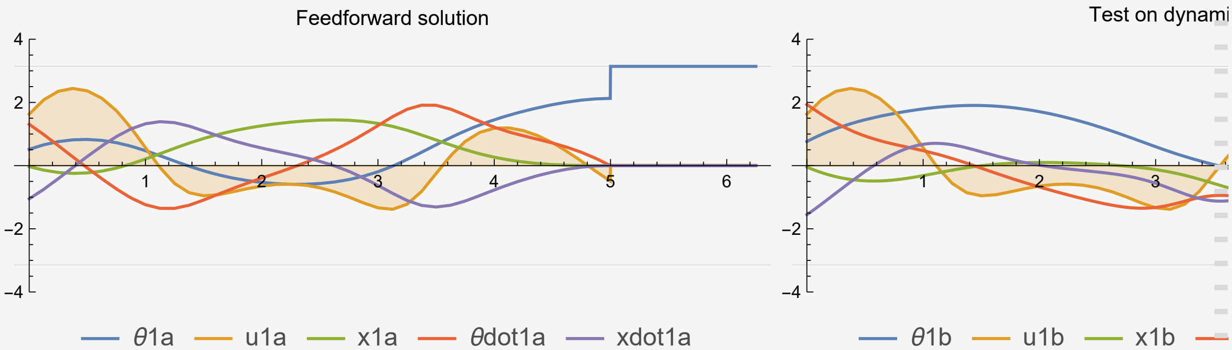
Out[941]=



In[942]:=
```
n=40;τ=5;τ1= τ*1.25 ;n2 = 20; M = 2;
A=0.2;
ICs = {-0.04586643004043421,-1.5525579578695232,0.7616256926010762,1.9326806921803874};
{J1,J,p1a,p1b,p1c} = SimulateLinearFeedbackEnd[ICs,n,τ,τ1,A];
Grid[{{p1a,p1b,p1c}}]
J1
J
```

FindRoot::cvmit: Failed to converge to the requested accuracy or precision within 100 iterations.

Out[946]=



Out[947]= 6.76809

Out[948]= 6.76809