In[592]:=

```
ClearAll["Global`*"];
(*ICs - Initial Conditions *)
ffCartPendulum[ICs_,n_,τ_,τ1_,A_,order_,maxIter_,InitGuess_]:=
Module[{x,xdot,f,θ,θdot,λ1,λ2,λ3,λ4,Δt,bcs,eqns,sv,froot,xff,xdotff,xff0,xdotff0,θff0,θdotff0,uff
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
    4 /(A Cos[2 θ]+A-2) (A θdot Sin[θ] (λ2-λ4 Cos[θ]))-λ3
};
xGuess0= {ICs⟦1⟧,ICs⟦2⟧,ICs⟦3⟧,ICs⟦4⟧,InitGuess⟦1⟧,InitGuess⟦1⟧,InitGuess⟦1⟧,InitGuess⟦1⟧};
 xGuess = Table[If[i ≠ -1,xGuessi+1 = xGuessi +Δt*f[xGuessi],xGuess0 = {ICs⟦1⟧,ICs⟦2⟧,ICs⟦3⟧,ICs⟦4

bcs={Subscript[x, 0]==ICs⟦1⟧,Subscript[xdot, 0]==ICs⟦2⟧,Subscript[x, n]==Subscript[xdot, n]==0,Subs
eqns=Flatten[Join[bcs,Table[Thread[{Subscript[x, i],Subscript[xdot, i],Subscript[θ, i],Subscript
1/2 Δt (f[{Subscript[x, i-1],Subscript[xdot, i-1],Subscript[θ, i-1],Subscript[θdot, i-1],Subscri
f[{Subscript[x, i],Subscript[xdot, i],Subscript[θ, i],Subscript[θdot, i],Subscript[λ1, i],Subscr
{Subscript[x, i-1],Subscript[xdot, i-1],Subscript[θ, i-1],Subscript[θdot, i-1],Subscript[λ1, i-1
sv = Flatten[Table[{{Subscript[x, i],xGuess⟦i+1⟧⟦1⟧},{Subscript[xdot, i],xGuess⟦i+1⟧⟦2⟧},{Subscr
                    {Subscript[λ1, i],xGuess⟦i+1⟧⟦5⟧},{Subscript[λ2, i],xGuess⟦i+1⟧⟦6⟧},{Subscr
froot=FindRoot[eqns,sv,MaxIterations→maxIter];
xff0=ListInterpolation[Table[Subscript[x, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→order];
xdotff0=ListInterpolation[Table[Subscript[xdot, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→ord
θff0=ListInterpolation[Table[Subscript[θ, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→order];
θdotff0=ListInterpolation[Table[Subscript[θdot, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→ord
λ1ff0 = ListInterpolation[Table[Subscript[λ1, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→order
λ2ff0 = ListInterpolation[Table[Subscript[λ2, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→order
λ3ff0 = ListInterpolation[Table[Subscript[λ3, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→order
λ4ff0 = ListInterpolation[Table[Subscript[λ3, i],{i,0,n}]/. froot,{0,τ},InterpolationOrder→order
uff0=ListInterpolation[Table[1/(1-A Cos[Subscript[θ, i]]^2) (Subscript[λ4, i]Cos[Subscript[θ, i]

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,λ1ff0,λ2ff0,λ3ff0,λ4ff0}]

testSwingUp[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
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→{"DiscontinuityProces
J = NIntegrate[uff0[t]^2,{t,0,τ}];
{xs,xdots,θs,θdots,uff0,J}]
```

```
CalculateSMatrix[x1a_,xdot1a_,θ1a_,θdot1a_,u1a_,τ_,A_]:= Module[{x,L,RHS,xdot,θ,θdot,u,K,S,soltn

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]; (* Fix this *)
Q = (1 0 0 0
     0 1 0 0
     0 0 1 0
     0 0 0 1);
Mf = Grad[D[L,u],xState];
R = D[L,{u,2}];
S0 = (0 0 0 0
      0 0 0 0
      0 0 0 0
      0 0 0 0);
RHS[t_] := (IdentityMatrix[4] + Af^T.S[t] + S[t].Af - KroneckerProduct[S[t].Bf,Bf^T.S[t]]) /. {x→
sol2 = S /. NDSolve[{S'[t]== RHS[t],S[0]==S0},S,{t,0,τ }];
S = sol2[[1]]
]
CalculateGains[x1a_,xdot1a_,θ1a_,θdot1a_,u1a_,time_,A_,τ_,S_]:= Module[{x,L,RHS,xdot,θ,θdot,u,K,
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};
Bf = D[fx,u] ;(*For 1D stuff use D*)
K = (Bf^T.S[τ - time])/. {x→ x1a[time], xdot → xdot1a[time], θ →θ1a[time], θdot → θdot1a[time],
K
]
testWithFB[ICs_,τ_,τ1_,xff0_,xdotff0_,θff0_,θdotff0_,uff0_,A_]:=Module[{eq,init,θ,θdot,θff,θdot
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];
S = CalculateSMatrix[xff,xdotff,θff,θdotff,uff,τ,A];
K[t_] := CalculateGains[xff,xdotff,θff,θdotff,uff,t,A,τ,S];
ufb[t_] := Piecewise[{{
 K[t].{xff[t]-x[t],xdotff[t]-xdot[t],θff[t]-θ[t],θdotff[t]-θdot[t]},0≤t≤τ}},κ1(θff[t]-θ[t])+κ2

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[
```

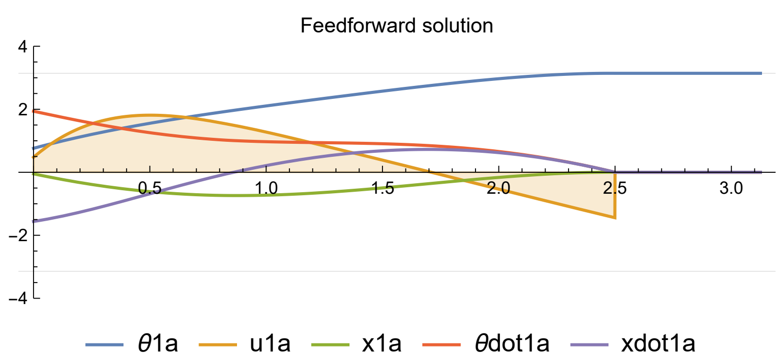
```
init={x[0]==ICs[[1]],xdot[0]==ICs[[2]],θ[0]==ICs[[3]],θdot[0]==ICs[[4]]};

{xs,xdots,θs,θdots}=NDSolveValue[{eq,init},{x,xdot,θ,θdot},{t,0,τ1},Method→{"DiscontinuityProces
us[t_]:=uff[t]+Piecewise[{{K[t].{xff[t]-xs[t],xdotff[t]-xdots[t],θff[t]-θs[t],θdotff[t]-θdots[
J = NIntegrate[us[t]^2,{t,0,τ}];
{xs,xdots,θs,θdots,us,J}]
```

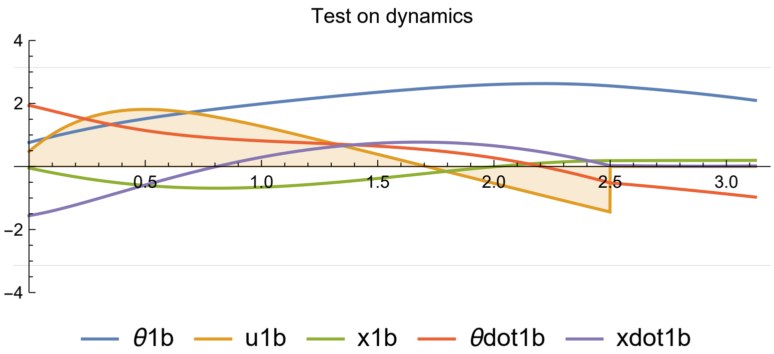
Check the performance of the new initial guess mechanism for the case where n = 40 and greater was required.
We see that n = 6 is only required here which is excellent!

In[654]:=
```
n = 6; τ = 2.5; τ1 = τ * 1.25; A = 0.2; order = 4; maxIter = 100;
ICs = {-0.048555026816794494`,
   -1.560065966757075`, 0.7613152376955525`, 1.9382391342292873`};
InitGuess =
  {RandomReal[{-1, 1}], RandomReal[{-1, 1}], RandomReal[{-1, 1}], RandomReal[{-1, 1}]};
{x1a, xdot1a, θ1a, θdot1a, u1a, λ1ff0, λ2ff0, λ3ff0, λ4ff0} =
  ffCartPendulum[ICs, n, τ, τ1, A, order, maxIter, InitGuess];
{x1b, xdot1b, θ1b, θdot1b, u1b, J1} = testSwingUp[ICs, τ, τ1, u1a, A];
{x1c, xdot1c, θ1c, θdot1c, u1c, J} =
 testWithFB[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 → "LQR feedback solution", AspectRatio → 1 / 3,
  ImageSize → 400, GridLines → {None, {-π, π}}]
```

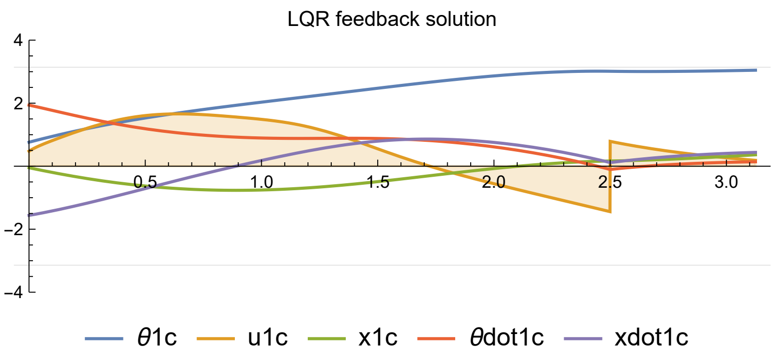
Out[659]=



Out[660]=



Out[661]=

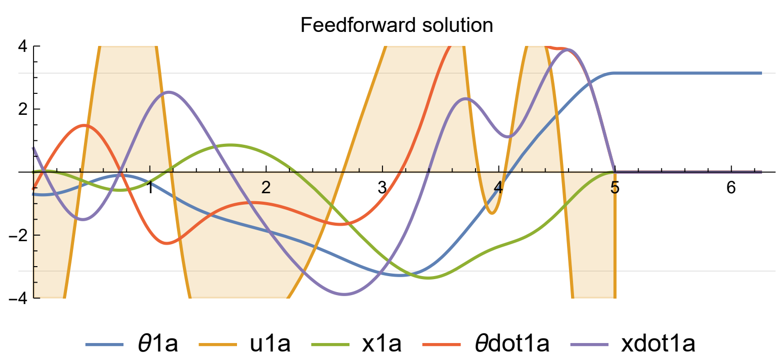


Understand the performance of the functions wrt random initial conditions

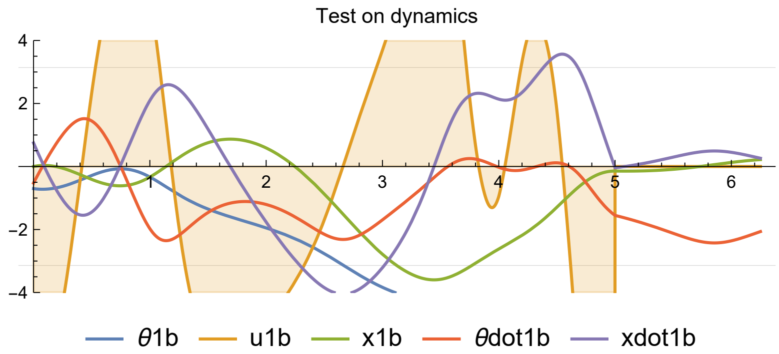
In[882]:=
```
n = 40; τ = 5; τ1 = τ * 1.25; A = 0.2; order = 4; maxIter = 100;
xdotMin = -1;
xdotMax = 1;
θMin = -π;
θMax = π;
θdotMin = -1;
θdotMax = 1;
xdotInit = RandomReal[{xdotMin, xdotMax}];
θInit = RandomReal[{θMin, θMax}];
θdotInit = RandomReal[{θdotMin, θdotMax}];
ICs = {0, xdotInit, θInit, θdotInit}
InitGuess =
  {RandomReal[{-1, 1}], RandomReal[{-1, 1}], RandomReal[{-1, 1}], RandomReal[{-1, 1}]};
{x1a, xdot1a, θ1a, θdot1a, u1a, λ1ff0, λ2ff0, λ3ff0, λ4ff0} =
  ffCartPendulum[ICs, n, τ, τ1, A, order, maxIter, InitGuess];
{x1b, xdot1b, θ1b, θdot1b, u1b, J1} = testSwingUp[ICs, τ, τ1, u1a, A];
{x1c, xdot1c, θ1c, θdot1c, u1c, J} =
 testWithFB[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 → "LQR feedback solution", AspectRatio → 1 / 3,
  ImageSize → 400, GridLines → {None, {-π, π}}]
```

Out[892]= {0, 0.74791, -0.701993, -0.52538}

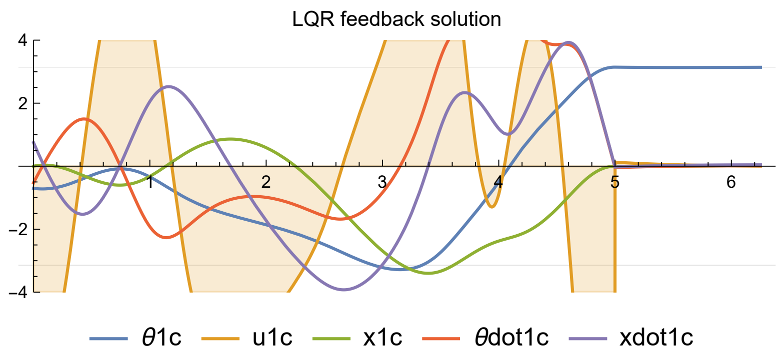
NIntegrate: NIntegrate failed to converge to prescribed accuracy after 9 recursive bisections in t$416882 near {t$416882} = {3.37883}. NIntegrate obtained 135.9613198178056` and 0.0003277794068873446` for the integral and error estimates.

Out[896]=


Out[897]=



Out[898]=



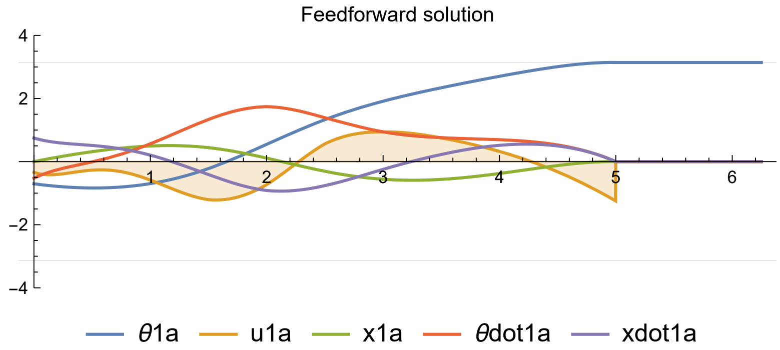
Individual Test Benches

In[1034]:=

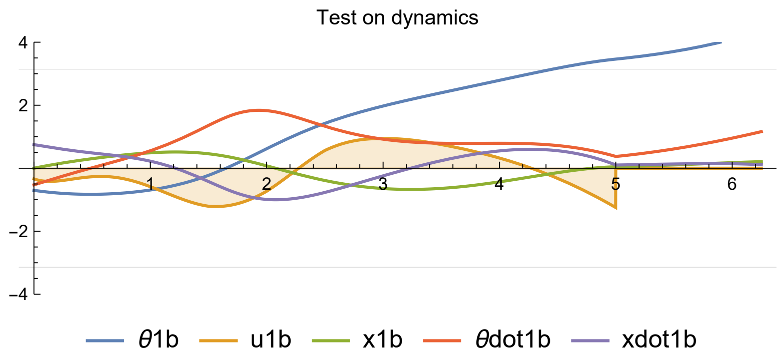
```
n = 10; τ = 5; τ1 = τ * 1.25; A = 0.2; order = 5; maxIter = 100;
ICs = {0, 0.617414376019306`, 2.7724493128186865`, -0.47379838931756035`};
ICs = {0, 0.7479096975521031`, -0.7019932321993547`, -0.5253804568578575`};
InitGuess =
 {RandomReal[{-1, 1}], RandomReal[{-1, 1}], RandomReal[{-1, 1}], RandomReal[{-1, 1}]}
{x1a, xdot1a, θ1a, θdot1a, u1a, λ1ff0, λ2ff0, λ3ff0, λ4ff0} =
  ffCartPendulum[ICs, n, τ, τ1, A, order, maxIter, InitGuess];
{x1b, xdot1b, θ1b, θdot1b, u1b, J1} = testSwingUp[ICs, τ, τ1, u1a, A];
{x1c, xdot1c, θ1c, θdot1c, u1c, J} =
 testWithFB[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 → "LQR feedback solution", AspectRatio → 1 / 3,
  ImageSize → 400, GridLines → {None, {-π, π}}]
```

Out[1037]= {0.506509, -0.161744, 0.591064, 0.321287}

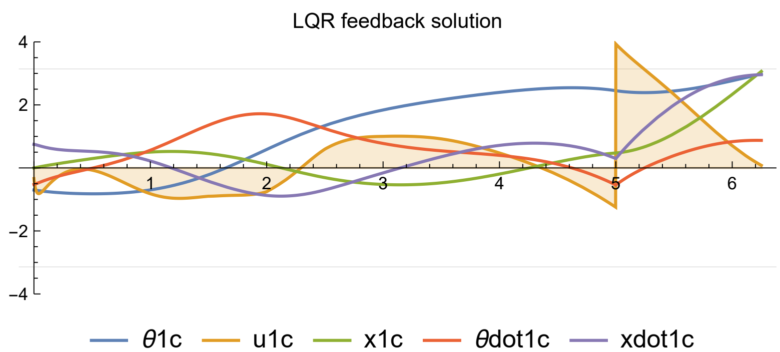
Out[1040]=



Out[1041]=



Out[1042]=



Observations:
The choice of n is critical and would determine the speed of the algorithm. Smaller n has higher speed but may not converge and hence would require a lot of re initializations of the initial guess, while a larger n would not require a lot of re initializations but would take longer to converge. A way to choose the best value of n for a specific system would be extremely useful.