In[58]:=
```
(*============================================================*)
(*0. SATURATION FUNCTION*)
(*============================================================*)
Clear[Sat];
Sat[x_] := If[Abs[x] > 0.1, Sign[x], x / 0.1];
SetAttributes[Sat, Listable];

Plot[Sat[t], {t, -3, 3}];

(*============================================================*)
(*1. SYSTEM SETUP*)
(*============================================================*)

n = 3; (*number of links*)

(*Initial joint positions*)
xj[0] = 0;
yj[0] = 0;

(*Kinematics:position of joint i*)
Do[xj[i] = Sum[-l[j] Sin[θ[j][t]], {j, 1, i}], {i, 1, n}];
Do[yj[i] = Sum[l[j] Cos[θ[j][t]], {j, 1, i}], {i, 1, n}];

(*Center of each link*)



(*Parameter rules*)
ruleL = Table[l[i] → 1, {i, n}];
ruleM = Table[m[i] → 1, {i, n}];

(*Animation demo rule (replace θ with linear motion)*)
RuleTh = Table[θ[i][t] → i t, {i, n}];

(*Moments of inertia*)
Do[Mj[i] = 1 / 12 m[i] × l[i]^2, {i, n}];

(*DISTURBANCE TORQUE MODEL (your line)*)
MoRule = Table[Moj[i] → -9 Derivative[1][θ[i]][t], {i, n}];

(*============================================================*)
(*2. GRAPHICS FOR ANIMATION*)
(*============================================================*)

(*link segments*)
Do[Link[i] = Line[{{xj[i - 1], yj[i - 1]}, {xj[i], yj[i]}}] /. ruleL, {i, 1, n}];
```

```
(*link centers represented by circles*)
Do[Mc[i] = Circle[{xb[i], yb[i]}, 0.1] /. ruleL, {i, 1, n}];

(*Animation*)
Animate[Show[Graphics[Table[{Link[k], Mc[k]}, {k, n}] /. RuleTh /. ruleL /. t → i],
   PlotRange → {{-n, n}, {-n, n}}, Frame → True], {i, 0, 15, .01}];

(*=============================================================*)
(*3. TRACKING CONTROLLER SETUP*)
(*=============================================================*)

q = Table[{θ[i][t]}, {i, n}];
qdes = {{Pi / 3 + Pi / 8 * Sin[t / 3]}, (*Joint 1 oscillates slowly*)
   {Pi / 5 + Pi / 6 * Cos[t / 3]}, (*Joint 2 oscillates at the same speed*)
   {Pi / 2 + Pi / 4 * Sin[t / 3]}  (*Joint 3 oscillates with a different amplitude*)};
(*desired joint angles*)

qtilde = q - qdes;

Λ = DiagonalMatrix[Table[Subscript[λ, i], {i, n}]];
dqr = D[qdes, t] - Λ.qtilde;
s = D[qtilde, t] + Λ.qtilde;        (*sliding variable*)

(*=============================================================*)
(*4. LAGRANGIAN DYNAMICS*)
(*=============================================================*)

T = FullSimplify[
   Sum[1 / 2 m[i] (D[xb[i], t]^2 + D[yb[i], t]^2) + 1 / 2 Mj[i] × D[θ[t][i], t]^2, {i, n}]];

V = FullSimplify[Sum[m[i] g yb[i], {i, n}]];
Lag = T - V;

(*Euler-Lagrange equation:M q''+C q'+G=τ*)
Eqs = Table[D[D[Lag, (θ[i]'[t])], t] - D[Lag, θ[i][t]] - τ[i], {i, n}];

(*=============================================================*)
(*5. M,C,AND G MATRICES*)
(*=============================================================*)

MassM = Table[Coefficient[Eqs[[i]], θ[j]''[t]], {i, n}, {j, n}];

Cc = Table[0, {n}, {n}];

Cc = Simplify[Table[Cc[[i, j]] =
      (1 / 2 Sum[D[MassM[[i, j]], q[[k]]] × D[q[[k]], t], {k, n}] + 1 / 2 Sum[(D[MassM[[i, k]], q[[j]]] -
```

```
D[MassM[[j, k]], q[[i]]]) D[q[[k]], t], {k, n}])[[1]], {i, n}, {j, n}]];

Gg = Table[{Coefficient[Eqs[[i]], g]}, {i, n}] g;

(*=============================================================*)
(*6. CONTROL LAW (Sliding Mode+Saturation)*)
(*=============================================================*)

massRules = Table[m[i] → 1, {i, n}];
lengthRules = Table[l[i] → 1, {i, n}];
StructParams = Join[massRules, lengthRules, {g → 9.81}];

lambdaRules = Table[Subscript[λ, i] → 4, {i, n}];
kRules = Table[k[i] → 1, {i, n}];
ControlParams = Join[lambdaRules, kRules, {η → 7}];

(*estimation errors*)
errorM = ConstantArray[1, {n, n}];
errorC = DiagonalMatrix[ConstantArray[1, n]];
errorG = Table[{1}, {i, n}];

Estimations = {Mhat → MassM + errorM, Chat → Cc + errorC, Ghat → Gg + errorG};

kVector = Table[k[i], {i, n}];

τhat = Simplify[Mhat.D[dqr, {t, 1}] + Chat.dqr + Ghat] /. Estimations /. StructParams /.
   ControlParams;
τc = τhat - kVector * Sat[s];

(*=============================================================*)
(*7. NUMERICAL SIMULATION*)
(*=============================================================*)

EqsN = FullSimplify[Eqs /. Table[τ[i] → τc[[i]], {i, n}]];

eqsToSolve = Table[EqsN[[i, 1]] == 0, {i, n}] /. Estimations /. StructParams /. ControlParams;

initPos = Table[θ[i][0] == 0, {i, n}];
initVel = Table[θ[i]'[0] == 0, {i, n}];

varsToSolve = Table[θ[i][t], {i, n}];
derivsToSolve = Table[θ[i]'[t], {i, n}];

solDE = NDSolve[Join[eqsToSolve, initPos, initVel],
   Join[varsToSolve, derivsToSolve], {t, 0, 150}];

(*=============================================================*)
```

```
(*8. RESULT PLOTS*)
(*============================================================*)

Plot[qtilde /. solDE, {t, 0, 150}, PlotRange → All];
Plot[τc /. Estimations /. StructParams /. ControlParams /. solDE,
  {t, 0, 150}, PlotRange → All];

(*============================================================*)
(*9. END EFFECTOR ANIMATION WITH TRACE*)
(*============================================================*)

endEffectorSymbolic = {xj[n], yj[n]} /. lengthRules;

Animate[
 Show[
  ParametricPlot[endEffectorSymbolic /. solDE, {t, 0, i},
   PlotStyle → {Thick, Red}],
  Graphics[
   Table[
       {
        Link[k], Mc[k]}, {k, n}] /. solDE /. t → i],
  PlotRange → {{-n, n}, {-n, n}},
  Frame → True,
  GridLines → Automatic,
  FrameLabel → {"X Position (m)", "Y Position (m)"}],
 {i, 0.01, 150, 0.05},
 AnimationRate → 1]
```

Out[114]=