```
In[97]:= ClearAll["Global`*"];
         清除全部
(*Suppose the two torques added are linear with parameters listed below*)
tx = 5;
yx1 = 1;
yx2 = 20;
(*Suppose the shot angle can be changed by the finesse*)
shotangle = Pi / 4 - 0.1;
            圆周率
(*Data from the ball*)
rball = 0.123;
m = 0.6;
(*The ground size*)
gdl = 14;
gdw = 15;
gdh = 10;
(*Data from the arm*)
m1 = 1.15;
m2 = 2.18;
l1 = 0.3;
l2 = 0.3;
(*The arm position*)
positionx = 2;
positiony = 12;
positionz = 1.45;

g = 9.8;
positionα = ArcCot[positionx / positiony];
            反余切
positiond = Sqrt[positionx^2 + positiony^2];
            平方根
qx = {{θ1[t]}, {θ2[t]}};
dqx = D[qx, t];
      偏导
ddqx = D[dqx, t];
       偏导
(*LAGRANGIAN*)
PEx = m1 * g * l1 / 2 * Sin[θ1[t]] + m2 * g * (l1 * Sin[θ1[t]] + l2 / 2 * Sin[θ1[t] + θ2[t]]) +
                        正弦                        正弦                   正弦
   m * g * (l1 * Sin[θ1[t]] + l2 * Sin[θ1[t] + θ2[t]]);
                 正弦              正弦
KEx = 1 / 6 * m1 * l1^2 * θ1'[t]^2 + 1 / 2 * m2 *
    ((l1^2 + l2^2 / 4 + l1 * l2 * Cos[θ2[t]]) * θ1'[t]^2 + l2^2 * θ2'[t]^2 / 4 +
                                  余弦
      (l2^2 / 2 + l1 * l2 * Cos[θ2[t]]) * θ2'[t] * θ1'[t]) + m2 * l2^2 / 24 *
                            余弦
    (θ2'[t] + θ1'[t])^2 + m / 2 * (l1^2 * θ1'[t]^2 + l2^2 * (θ2'[t] + θ1'[t])^2 +
```

```
      2 * l1 * l2 * θ1'[t] * (θ1'[t] + θ2'[t]) * Cos[θ2[t]]);
                                                 余弦
k1 = yx1 / tx;
k2 = yx2 / tx;
Fx = {{-k1 * (t - tx) ^2}, {-k2 * (t - tx) ^2}};
Lx = KEx - PEx;
Eqx = (Thread[D[D[Lx, dqxᵀ], t] - D[Lx, qxᵀ] == Flatten[Fxᵀ]]);
       线性作用 …偏导              偏导          压平
ELtempx = Solve[{Eqx[[1]], Eqx[[2]]}, Flatten[ddqx]];
          解方程                        压平
ELx = {θ1''[t] == ELtempx[[1, 1, 2]], θ2''[t] == ELtempx[[1, 2, 2]]};

InitConx = {θ1'[0] == 0, θ2'[0] == 0, θ1[0] == Pi / 6, θ2[0] == 2 * Pi / 3};
                                                 圆周率              圆周率
solx =
  NDSolve[Join[ELx, InitConx], {θ1, θ2}, {t, 0, 5}, Method → {"EventLocator",
  数值求… 连接                                       方法
     "Event" → θ2[t], "EventAction" :→ Throw[tend = t, "StopIntegration"]}];
                                       抛
coord1x[θ1_, θ2_] := {(positiond - l1 * Cos[θ1]) * Cos[positionα],
                                         余弦          余弦
   (positiond - l1 * Cos[θ1]) * Sin[positionα], positionz + l1 * Sin[θ1]};
                     余弦         正弦                             正弦
xtrans[θ1_, θ2_] := (positiond - l1 * Cos[θ1] - l2 * Cos[θ1 + θ2]) * Cos[positionα];
                                      余弦           余弦               余弦
ytrans[θ1_, θ2_] := (positiond - l1 * Cos[θ1] - l2 * Cos[θ1 + θ2]) * Sin[positionα];
                                      余弦           余弦               正弦
ztrans[θ1_, θ2_] := positionz + l1 * Sin[θ1] + l2 * Sin[θ1 + θ2];
                                     正弦          正弦
vtemp1 = θ1'[tend] /. solx[[1]];
vtemp2 = θ2'[tend] /. solx[[1]];
agtemp1 = θ1[tend] /. solx[[1]];
agtemp2 = θ2[tend] /. solx[[1]];
vtemp = Sqrt[
        平方根
   (l1 * vtemp1 * Sin[agtemp1] + l2 * (vtemp1 + vtemp2) * Sin[agtemp1 + agtemp2]) ^2 +
                  正弦                                    正弦
    (l1 * vtemp1 * Cos[agtemp1] + l2 * (vtemp1 + vtemp2) * Cos[agtemp1 + agtemp2]) ^2];
                   余弦                                    余弦
vxtrans[θ1_, θ2_] := (l1 * vtemp1 * Sin[θ1] + l2 * (vtemp1 + vtemp2) * Sin[θ1 + θ2]) *
                                     正弦                               正弦
   Cos[positionα];
   余弦
vytrans[θ1_, θ2_] := (l1 * vtemp1 * Sin[θ1] + l2 * (vtemp1 + vtemp2) * Sin[θ1 + θ2]) *
                                     正弦                               正弦
   Sin[positionα];
   正弦
vztrans[θ1_, θ2_] := l1 * vtemp1 * Cos[θ1] + l2 * (vtemp1 + vtemp2) * Cos[θ1 + θ2];
                                   余弦                                 余弦
```

```
                                   余弦                                               余弦
coord2x[θ1_, θ2_] := {xtrans[θ1, θ2], ytrans[θ1, θ2], ztrans[θ1, θ2]};
xinit = xtrans[θ1[tend] /. solx[[1]], θ2[tend] /. solx[[1]]];
yinit = ytrans[θ1[tend] /. solx[[1]], θ2[tend] /. solx[[1]]];
zinit = ztrans[θ1[tend] /. solx[[1]], θ2[tend] /. solx[[1]]];
vxinit = -vtemp * Cos[shotangle] * Cos[positionα];
                  余弦              余弦
vyinit = -vtemp * Cos[shotangle] * Sin[positionα];
                  余弦              正弦
vzinit = vtemp * Sin[shotangle];
                 正弦
c1x[t_] := coord1x[θ1[t] /. solx[[1]], θ2[t] /. solx[[1]]] /; t ≤ tend;
c1x[t_] := coord1x[θ1[tend] /. solx[[1]], θ2[tend] /. solx[[1]]] /; t > tend;
c2x[t_] := coord2x[θ1[t] /. solx[[1]], θ2[t] /. solx[[1]]] /; t ≤ tend;
c2x[t_] := coord2x[θ1[tend] /. solx[[1]], θ2[tend] /. solx[[1]]] /; t > tend;
cball[t_] := coord2x[θ1[t] /. solx[[1]], θ2[t] /. solx[[1]]] /; 0 ≤ t ≤ tend;
q = {{x[t]}, {y[t]}, {z[t]}};
dq = D[q, t];
     偏导
ddq = D[dq, t];
      偏导
PE = mx * gx * z[t];
KE = 1 / 2 * mx * (x'[t]^2 + y'[t]^2 + z'[t]^2);
L = KE - PE;
p = D[L, dq^T];
    偏导
H = p.dq - L;
Φ = y[t];
Φ1 = (x[t] * (1 - bktr / Sqrt (x[t]^2 + (y[t] - (bktd - bktw + bktr))^2)))^2 +
                        平方根
    ((y[t] - (bktd - bktw + bktr)) * (1 -
         bktr / Sqrt (x[t]^2 + (y[t] - (bktd - bktw + bktr))^2)))^2 + (z[t] - bkth)^2;
                平方根
Eq = (Thread[D[D[L, dq^T], t] - D[L, q^T] == 0]);
      线性作用 ·· 偏导          偏导
ELtemp = Solve[{Eq[[1]], Eq[[2]], Eq[[3]]}, Flatten[ddq]];
         解方程                            压平
EL = {x''[t] == ELtemp[[1, 1, 2]],
   y''[t] == ELtemp[[1, 2, 2]], z''[t] == ELtemp[[1, 3, 2]]};
H0 = H /. {x'[t] → vx1, y'[t] → vy1, z'[t] -> vz1, x[t] → x1, y[t] → y1, z[t] → z1};
Ht =
  H /. {x'[t] -> vtx1, y'[t] -> vty1, z'[t] -> vtz1, x[t] → x1, y[t] → y1, z[t] → z1};
pt = p /. {x'[t] -> vtx1, y'[t] -> vty1, z'[t] -> vtz1};

H0x = H /. {x'[t] → vx2, y'[t] → vy2, z'[t] -> vz2, x[t] → x2, y[t] → y2, z[t] → z2};
Htx =
  H /. {x'[t] -> vtx2, y'[t] -> vty2, z'[t] -> vtz2, x[t] → x2, y[t] → y2, z[t] → z2};
ptx = p /. {x'[t] -> vtx2, y'[t] -> vty2, z'[t] -> vtz2};
```

```
Eqimpact = (Thread[p - pt == λ * D[Φ, qᵀ] /.
            线性作用            偏导
    {x'[t] → vx1, y'[t] → vy1, z'[t] → vz1, x[t] → x1, y[t] → y1, z[t] → z1}]);
equH = First[Ht - H0] == 0;
       第一个
Eqimpacttemp = Solve[{Eqimpact[[1]], Eqimpact[[2]], Eqimpact[[3]], equH, λ ≠ 0},
               解方程
  {vtx1, vty1, vtz1, λ}];
Eqimpactx = (Thread[p - ptx == λx * D[Φ1, qᵀ] /.
             线性作用             偏导
    {x'[t] → vx2, y'[t] → vy2, z'[t] → vz2, x[t] → x2, y[t] → y2, z[t] → z2}]);
equHx = First[Htx - H0x] == 0;
        第一个
Eqimpacttempx = Solve[{Eqimpactx[[1]], Eqimpactx[[2]],
                解方程
   Eqimpactx[[3]], equHx, λx ≠ 0}, {vtx2, vty2, vtz2, λx}];
bkth = 3.05;
bktw = 0.01;
bktr = 0.235;
bktd = 0.15;
InitCon = {x'[0] == vxinit, x[0] == xinit,
  y[0] == yinit, y'[0] == vyinit, z[0] == zinit, z'[0] == vzinit};
mx = 0.6;
gx = 9.8;
sol = NDSolve[Join[EL, InitCon, {WhenEvent[y[t] == 0,
      数值求…  连接               当事件发生时
     {x1 = x[t], y1 = y[t], z1 = z[t], vx1 = x'[t], vy1 = y'[t], vz1 = z'[t],
      x'[t] → (vtx1 /. Eqimpacttemp[[1]]), y'[t] → (vty1 /. Eqimpacttemp[[1]]),
      z'[t] → (vtz1 /. Eqimpacttemp[[1]])}]}, {WhenEvent[
                                               当事件发生时
     (z[t] - bkth == 0) && (x[t]^2 + (y[t] - (bktd - bktw + bktr))^2 - bktr^2 == 0),
     {x2 = x[t], y2 = y[t], z2 = z[t], vx2 = x'[t], vy2 = y'[t], vz2 = z'[t],
      x'[t] → (vtx2 /. Eqimpacttempx[[1]]), y'[t] → (vty2 /. Eqimpacttempx[[1]]),
      z'[t] → (vtz2 /. Eqimpacttempx[[1]])}]},
    {WhenEvent[z[t] == 0, { tendx = t, "StopIntegration"}]}], {x, y, z}, {t, 0, 10}];
     当事件发生时
coord[x_, y_, z_] := {x, y, z};
cball[t_] := coord[(x[t - tend] /. sol)[[1]],
    (y[t - tend] /. sol)[[1]], (z[t - tend] /. sol)[[1]]] /; t > tend;
Animate[Show[Graphics3D[Polygon[{{-gdw/2, 0, 0}, {-gdw/2, gdl, 0},
        显示  三维图形       多边形
    {gdw/2, gdl, 0}, {gdw/2, 0, 0}, {-gdw/2, 0, 0}, {-gdw/2, 0, gdh},
    {gdw/2, 0, gdh}, {gdw/2, 0, 0}}]], Graphics3D[Point[cball[t]]],
                                        三维图形       点
  Graphics3D[{Green, Line[{{positionx, positiony, positionz}, c1x[t]}]}],
             绿色     线
  Graphics3D[{Red, Line[{c1x[t], c2x[t]}]}], Graphics3D[{Black,
             红    线                          三维图形     黑
```

```
                                      红    线                   三维图形      黑
   Line[{{positionx, positiony, 0}, {positionx, positiony, positionz}}]}],
   线
  ParametricPlot3D[{{(bktr + bktw * Cos[v]) Sin[u], bktd - bktw + bktr +
                                    余弦        正弦
     (bktr + bktw * Cos[v]) Cos[u], bkth + bktw * Sin[v]}}, {u, 0, 2 Pi},
                   余弦       余弦                正弦                 圆周率
   {v, 0, 2 Pi}, Mesh → None, Axes → False, Boxed → False, PlotStyle → {Black}],
           …      网格     无     坐标轴   假     边界框    假   绘制样式        黑
  ParametricPlot3D[cball[t], {t, 0, t}], AspectRatio → Automatic,
                                          宽高比         自动
  PlotRange → {{-gdw/2, gdw/2}, {0, gdl}, {0, gdh}}, Frame → True],
                                                     边框     真
 {t, 0, tendx}, AnimationRate → 1, AnimationRunning → False]
                动画速率              动画执行             假
```

Out[189]=