In[1]:=
```
(*Here we create a function that solves the equations of motion
 in discrete sections, defined when each constaint goes to zero*)
solvestep[L_, t0_, θ0_, ϕ0_, α0_, dθ0_, dϕ0_, dα0_, tguess_, tf_, constraints_] := (
  (*Add the constraints*)
  L1 = L;
  For[i = 1, i ≤ Length[constraints], i++,
   L1 = L1 + constraints[[i]][[1]] × constraints[[i]][[2]];];
  sol = {};
  (*NDSolve based on how many constraints*)
  Which[Length[constraints] == 2,
   sol =
    NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*first solve with two constraints*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0,
      D[constraints[[2, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]], constraints[[2, 1]]}, {t, t0, tf}],
   Length[constraints] == 1,
   sol =
    NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*then solve with one constraint*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]]}, {t, t0, tf}],
   Length[constraints] == 0,
   sol = NDSolve[
     {EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*finally solve with no constraints*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0},
     {θ[t], ϕ[t], α[t]}, {t, t0, tf}]];
  (*Pull off solutions*)
```

```
  solnlist = {};
  For[i = 1, i ≤ Length[sol[[1]]], i++,
   AppendTo[solnlist, sol[[1, i, 2]]];
  ];
  (*Obtain first zeros so we can identify when
   each constraint goes to zero and splice them together later*)
  zeros = {};
  For[i = 4, i ≤ Length[solnlist], i++,
   AppendTo[zeros,
     FindRoot[solnlist[[i]], {t, tguess}]];
  ];
  (*Return solutions and zeros*)
  Return[{solnlist, zeros}];
 )
Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L, t, tf,
 EL, soln, xb, yb, xcw, ycw, xcma, ycma, λ, ɣ, G1, G2, x, y, ltot, k]
xbvfinal = {};
tlist = {};
distance = {};
For[k = 2.9, k < 3, k = k + 0.01,
 Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L, t, tf, EL, soln,
  xb, yb, xcw, ycw, xcma, ycma, λ, ɣ, G1, G2, x, y, xbp, ybp, tlaunch];
 Print["current step:", k];

 l1 = 10;
 Print["length of barrel side:", l1];
 l2 = 4;
 Print["length of cw side:", l2];
 mcw = 3000;
 mb = 60;

 (*above four are what we would vary*)
 g = 8.87; (*9.81 for Earth*)(*3.81 for Mars*)(*8.87 for Venus*)
 ma = 1000;
 h = 10;
 r = 6;

 T := 1/2 mb (D[xb, t]^2 + D[yb, t]^2) + 1/2 mcw (D[xcw, t]^2 + D[ycw, t]^2) + 1/2 IIa D[θ[t], t]^2;
 IIa := 1/3 ma ((l1^3 + l2^3)/(l1 + l2));
 Print["moment of inertia:", IIa];
 U := mb g yb + mcw g ycw + ma g ycma;
```

```
xb := l1 Cos[θ[t]] + r Cos[ϕ[t]];
yb := -l1 Sin[θ[t]] - r Sin[ϕ[t]];
xcw := -l2 Cos[θ[t]] + h Cos[α[t]];
ycw := l2 Sin[θ[t]] - h Sin[α[t]];
ycma := -(l1 - l2)/2 Sin[θ[t]];
G1 := θ[t] - α[t] + α[0];
G2 := θ[t] - ϕ[t] + ϕ[0];
L := T - U;
EL[L_, x_] := D[L, x[t]] - D[L, x'[t], t] == 0;
Clear[L1, solnlist, zeros, sol, constraints1, tf];
tf = 30;
constraints1 = {{λ[t], G1}, {γ[t], G2}};
step1 = solvestep[L, 0, 0, Pi, 7 Pi / 4, 0, 0, 0, 1, tf, constraints1];
Clear[L2, solnlist, zeros, constraints2, sol, t2, step2];
constraints2 = {{γ[t], G2}};
t2 = step1[[2, 1, 1, 2]];
step2 = solvestep[L, t2, step1[[1, 1, 0]][t2],
  step1[[1, 2, 0]][t2], step1[[1, 3, 0]][t2], step1[[1, 1, 0]]'[t2],
  step1[[1, 2, 0]]'[t2], step1[[1, 3, 0]]'[t2], 2, tf, constraints2];
Clear[L1, solnlist, zeros, constraints3, sol, t3];
constraints3 = {};
t3 = step2[[2, 1, 1, 2]];
step3 = solvestep[L, t3, step2[[1, 1, 0]][t3],
  step2[[1, 2, 0]][t3], step2[[1, 3, 0]][t3], step2[[1, 1, 0]]'[t3],
  step2[[1, 2, 0]]'[t3], step2[[1, 3, 0]]'[t3], 2, tf, constraints3];
θpw[t_] = Piecewise[{{step1[[1, 1]], 0 ≤ t < t2}, {step2[[1, 1]], t2 ≤ t < t3}, {step3[[1, 1]], t3 ≤ t ≤ tf}}];
ϕpw[t_] = Piecewise[{{step1[[1, 2]], 0 ≤ t < t2}, {step2[[1, 2]], t2 ≤ t < t3}, {step3[[1, 2]], t3 ≤ t ≤ tf}}];
αpw[t_] = Piecewise[{{step1[[1, 3]], 0 ≤ t < t2}, {step2[[1, 3]], t2 ≤ t < t3}, {step3[[1, 3]], t3 ≤ t ≤ tf}}];
λpw[t_] = Piecewise[{{step1[[1, 4]], 0 ≤ t < t2}, {0, t2 ≤ t < t3}, {0, t3 ≤ t ≤ tf}}];
γpw[t_] = Piecewise[{{step1[[1, 5]], 0 ≤ t < t2}, {step2[[1, 4]], t2 ≤ t < t3}, {0, t3 ≤ t ≤ tf}}];
xbp[t_] = l1 Cos[θpw[t]] + r Cos[ϕpw[t]];
ybp[t_] = -l1 Sin[θpw[t]] - r Sin[ϕpw[t]];
Clear[x, y, xdrag, ydrag];
```

```
tlaunch = k;
ρ = 65; (*1.20 for Earth*)(*0.02 for Mars*)(*65 for Venus*)
Cd = 0.2;
R = .5; (*meters*)
A = π * R^2;
B = 1/2 * Cd * ρ * A;
Print["b", B];
dragsoln = NDSolve[{
    xdrag[tlaunch] == xbp[tlaunch],
    ydrag[tlaunch] == ybp[tlaunch],
    xdrag'[tlaunch] == xbp'[tlaunch],
    ydrag'[tlaunch] == ybp'[tlaunch],
    xdrag''[t] == -B/mb Abs[xdrag'[t]] * xdrag'[t],
    ydrag''[t] == -B/mb Abs[ydrag'[t]] * ydrag'[t] - g},
  {xdrag, ydrag}, {t, tlaunch, tf}];
xd = dragsoln[[1, 1, 2]];
yd = dragsoln[[1, 2, 2]];

xbtotal[t_] = {xbp[t]  0 ≤ t ≤ tlaunch
               xd[t]   tlaunch < t ≤ tf};
ybtotal[t_] = {ybp[t]  0 ≤ t ≤ tlaunch
               yd[t]   tlaunch < t ≤ tf};
tfest = FindRoot[ybtotal[t], {t, 25}][[1, 2]];
AppendTo[tlist, tlaunch];
Print["Tfest:", tfest];
Print["x-vel:", xbp'[tlaunch]];
Print["y-vel:", ybp'[tlaunch]];
Print["list of speed:", xbvfinal];
Print["tlaunch:", tlaunch];
Print["distance:", xbtotal[tfest]];
speed = Sqrt[(xbp'[tlaunch])^2 + (ybp'[tlaunch])^2];
AppendTo[distance, xbtotal[tfest]];
Print["speed:", speed];
AppendTo[xbvfinal, speed];

]
current step:2.9

length of barrel side:10

length of cw side:4
```

moment of inertia: $\frac{76\,000}{3}$

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

InterpolatingFunction: Input value {1.24542} lies outside the range of data in the interpolating function. Extrapolation will be used.

b5.10509

Tfest:8.53325

x-vel:75.3106

y-vel:57.0738

list of speed:{}

tlaunch:2.9

distance:32.6011

speed:94.4939

current step:2.91

length of barrel side:10

length of cw side:4

moment of inertia: $\frac{76\,000}{3}$

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

General: Further output of NDSolve::pdord will be suppressed during this calculation.

InterpolatingFunction: Input value {1.24542} lies outside the range of data in the interpolating function. Extrapolation will be used.

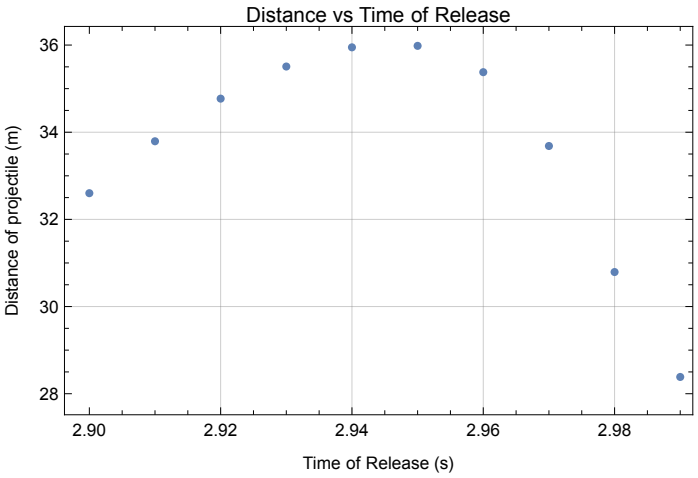
InterpolatingFunction: Input value {1.24542} lies outside the range of data in the interpolating function. Extrapolation will be used.

General: Further output of InterpolatingFunction::dmval will be suppressed during this calculation.

In[7]:=
```
toptimal = Transpose[{tlist, distance}]
ListPlot[toptimal, Frame → True, GridLines -> Automatic,
 FrameLabel → {"Time of Release (s)", "Distance of projectile (m)"},
 PlotLabel → "Distance vs Time of Release"]
```

Out[7]=
```
{{2.9, 32.6011}, {2.91, 33.7926}, {2.92, 34.7707}, {2.93, 35.5075}, {2.94, 35.9478},
 {2.95, 35.9818}, {2.96, 35.3778}, {2.97, 33.6839}, {2.98, 30.7919}, {2.99, 28.3844}}
```

Out[8]=



In[9]:=
```
totalList = Transpose[{toptimal, xbvfinal}]
```

Out[9]=
```
{{{2.9, 32.6011}, 94.4939}, {{2.91, 33.7926}, 94.2298},
 {{2.92, 34.7707}, 93.7761}, {{2.93, 35.5075}, 93.1403},
 {{2.94, 35.9478}, 92.3314}, {{2.95, 35.9818}, 91.3599}, {{2.96, 35.3778}, 90.237},
 {{2.97, 33.6839}, 88.9736}, {{2.98, 30.7919}, 87.5808}, {{2.99, 28.3844}, 86.0689}}
```

In[16]:=
```
ϕpw[2.949999999999999`]
```

Out[16]=
```
10.9101
```

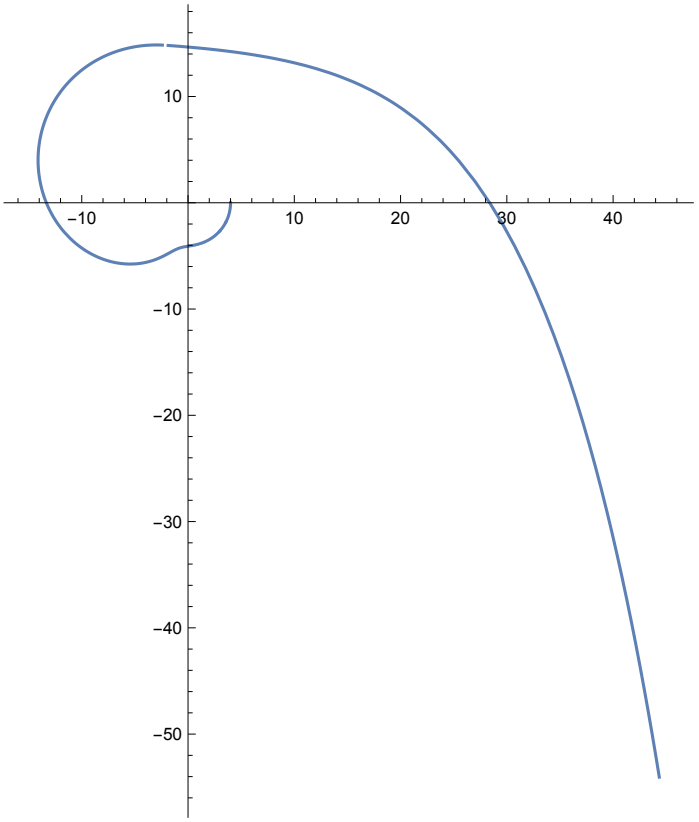
In[17]:=
```
With[{d = N[10.910106144258393` / °]}, Defer[d °]]
```

Out[17]=
```
625.103 °
```

In[12]:=

In[13]:=
```
ParametricPlot[{xbtotal[t], ybtotal[t]}, {t, 0, 10}]
```

Out[13]=



In[14]:=
```
ArcTan[ybtotal'[2.7799999999999985`] / xbtotal'[2.7799999999999985`]]
```

Out[14]=
```
-1.47822
```

NDSolve Module

In[1]:=

```
solvestep[L_, t0_, θ0_, ϕ0_, α0_, dθ0_, dϕ0_, dα0_, tguess_, tf_, constraints_] := (
  (*Add the constraints*)
  L1 = L;
  For[i = 1, i ≤ Length[constraints], i++,
   L1 = L1 + constraints[[i]][[1]] × constraints[[i]][[2]];];
  sol = {};
  (*NDSolve based on how many constraints*)
  Which[Length[constraints] == 2,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0,
      D[constraints[[2, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]], constraints[[2, 1]]}, {t, t0, tf}],
   Length[constraints] == 1,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]]}, {t, t0, tf}],
   Length[constraints] == 0,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0},
     {θ[t], ϕ[t], α[t]}, {t, t0, tf}]];
  (*Pull off solutions*)
  solnlist = {};
  For[i = 1, i ≤ Length[sol[[1]]], i++,
   AppendTo[solnlist, sol[[1, i, 2]]];
  ];
  (*Obtain first zeros*)
  zeros = {};
  For[i = 4, i ≤ Length[solnlist], i++,
   AppendTo[zeros,
```

```
    FindRoot[solnlist[[i]], {t, tguess}]];
 ];
 (*Return solutions and zeros*)
 Return[{solnlist, zeros}];
)
```

Kinetic energy

In[2]:=
```
Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L,
 t, tf, EL, soln, xb, yb, xcw, ycw, xcma, ycma, λ, γ, G1, G2, x, y]
```

In[3]:=
$$T := \frac{1}{2} mb \left(D[xb, t]^2 + D[yb, t]^2\right) + \frac{1}{2} mcw \left(D[xcw, t]^2 + D[ycw, t]^2\right) + \frac{1}{2} IIa\, D[\theta[t], t]^2$$

Note, if you call moment of inertia I you generate errors because this is the imaginary number

In[4]:=
$$IIa := \frac{1}{3} ma \left(\frac{l1^3 + l2^3}{l1 + l2}\right)$$

Potential energy

In[5]:=
```
U := mb g yb + mcw g ycw + ma g ycma
```

Define transformation to generalized coordinates

In[6]:=
```
xb := l1 Cos[θ[t]] + r Cos[ϕ[t]] (*these are barrel coord*)
yb := -l1 Sin[θ[t]] - r Sin[ϕ[t]]
xcw := -l2 Cos[θ[t]] + h Cos[α[t]] (*these are counter-weight coord*)
ycw := l2 Sin[θ[t]] - h Sin[α[t]]
ycma := -(l1 - l2)/2 Sin[θ[t]] (*these are center of mass coord*)
```

Define Constraint Functions:

In[11]:=
```
G1 := θ[t] - α[t] + α[0] (*counter weight*)
```

In[12]:=
```
G2 := θ[t] - ϕ[t] + ϕ[0] (*barrel*)
```

Let' s see what T & U look like ... It looks nice. Doing this in Mathematica saves the pain of differentiating, squaring etc. Try a few lines of this to get a feel for how nasty it is.

In[13]:=
```
Simplify[T]
```

Out[13]=
$$\frac{1}{6}\left(3\,h^2\,mcw\,\alpha'[t]^2-6\,h\,l2\,mcw\,\text{Cos}[\alpha[t]-\theta[t]]\,\alpha'[t]\,\theta'[t]+\left(-l1\,l2\,ma+l1^2\,(ma+3\,mb)+l2^2\,(ma+3\,mcw)\right)\theta'[t]^2+6\,l1\,mb\,r\,\text{Cos}[\theta[t]-\phi[t]]\,\theta'[t]\,\phi'[t]+3\,mb\,r^2\,\phi'[t]^2\right)$$

In[14]:=
```
Simplify[U]
```

Out[14]=
$$-\frac{1}{2}g\,(2\,h\,mcw\,\text{Sin}[\alpha[t]]+(l1\,(ma+2\,mb)-l2\,(ma+2\,mcw))\,\text{Sin}[\theta[t]]+2\,mb\,r\,\text{Sin}[\phi[t]])$$

Define L

In[15]:=
```
L := T - U
```

In[16]:=
```
Simplify[L]
```

Out[16]=
$$\frac{1}{6}\left(3\,h^2\,mcw\,\alpha'[t]^2-6\,h\,l2\,mcw\,\text{Cos}[\alpha[t]-\theta[t]]\,\alpha'[t]\,\theta'[t]+\left(-l1\,l2\,ma+l1^2\,(ma+3\,mb)+l2^2\,(ma+3\,mcw)\right)\theta'[t]^2+6\,l1\,mb\,r\,\text{Cos}[\theta[t]-\phi[t]]\,\theta'[t]\,\phi'[t]+3\left(g\,(2\,h\,mcw\,\text{Sin}[\alpha[t]]+(l1\,(ma+2\,mb)-l2\,(ma+2\,mcw))\,\text{Sin}[\theta[t]]+2\,mb\,r\,\text{Sin}[\phi[t]])+mb\,r^2\,\phi'[t]^2\right)\right)$$

In[17]:=

In[18]:=
```
g := 9.81
mb := 60
mcw := 3000
ma := 1000
l1 := 10
l2 := 4
h := 10
r := 6
```

In[26]:=
```
EL[L_, x_] := D[L, x[t]] - D[L, x'[t], t] == 0;
```

In[27]:=
```
Clear[L1, solnlist, zeros, sol, constraints1, tf];
```

In[28]:=
```
tf = 20;
```

In[29]:=
```
constraints1 = {{λ[t], G1}, {γ[t], G2}};
```

In[30]:=
```
step1 = solvestep[L, 0, 0, Pi, 7 Pi / 4, 0, 0, 0, 1, tf, constraints1]
```

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

Out[30]=
```
{{InterpolatingFunction[  Domain: {{0., 20.}} Output: scalar ][t],
  InterpolatingFunction[  Domain: {{0., 20.}} Output: scalar ][t],
  InterpolatingFunction[  Domain: {{0., 20.}} Output: scalar ][t],
  InterpolatingFunction[  Domain: {{0., 20.}} Output: scalar ][t],
  InterpolatingFunction[  Domain: {{0., 20.}} Output: scalar ][t]}, {{t → 1.19494}, {t → 1.70667}}}
```

In[31]:=
```
Plot[step1[[1, 4]], {t, 0, t2}]
Plot[step1[[1, 5]], {t, 0, t2}]
```

Plot: Limiting value t2 in {t, 0, t2} is not a machine–sized real number.

Out[31]=
```
Plot[step1[[1, 4]], {t, 0, t2}]
```

Plot: Limiting value t2 in {t, 0, t2} is not a machine–sized real number.

Out[32]=
```
Plot[step1[[1, 5]], {t, 0, t2}]
```

In[33]:=
```
Print[step1[[2]]]
```

```
{{t → 1.19494}, {t → 1.70667}}
```

In[34]:=
```
Clear[L2, solnlist, zeros, constraints2, sol, t2, step2];
```

In[35]:=
```
constraints2 = {{γ[t], G2}};
```

In[36]:=
```
t2 = step1[[2, 1, 1, 2]];
Print[t2];
```

```
1.19494
```

In[38]:=
```
step2 = solvestep[L, t2, step1[[1, 1, 0]][t2], step1[[1, 2, 0]][t2], step1[[1, 3, 0]][t2],
  step1[[1, 1, 0]]'[t2], step1[[1, 2, 0]]'[t2], step1[[1, 3, 0]]'[t2], 2, tf, constraints2]
```

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

InterpolatingFunction: Input value {1.11127} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {1.17017} lies outside the range of data in the interpolating function. Extrapolation will be used.

Out[38]=
```
{{InterpolatingFunction[  Domain: {{1.19, 20.}}
                                                                            Output: scalar
                                                                            Data not in notebook; Store now »
                                                                          ][t],

 InterpolatingFunction[  Domain: {{1.19, 20.}}
                                                                            Output: scalar
                                                                            Data not in notebook; Store now »
                                                                          ][t],

 InterpolatingFunction[  Domain: {{1.19, 20.}}
                                                                            Output: scalar
                                                                            Data not in notebook; Store now »
                                                                          ][t],

 InterpolatingFunction[  Domain: {{1.19, 20.}}
                                                                            Output: scalar
                                                                            Data not in notebook; Store now »
                                                                          ][t]}, {{t → 1.43735}}}
```

In[39]:=

In[40]:=
```
Plot[step2[[1, 4]], {t, t2, t3}]
```

Plot: Limiting value t3 in {t, 1.19494, t3} is not a machine-sized real number.

Out[40]=
```
Plot[step2[[1, 4]], {t, t2, t3}]
```

In[41]:=
```
Print[step2[[2]]]
```

```
{{t → 1.43735}}
```

In[42]:=
```
Clear[L1, solnlist, zeros, constraints3, sol, t3];
```

In[43]:=
```
constraints3 = {};
```

In[44]:=
```
t3 = step2[[2, 1, 1, 2]];
Print[t3]
```

```
1.43735
```

In[46]:=
```
step3 = solvestep[L, t3, step2[[1, 1, 0]][t3], step2[[1, 2, 0]][t3], step2[[1, 3, 0]][t3],
  step2[[1, 1, 0]]'[t3], step2[[1, 2, 0]]'[t3], step2[[1, 3, 0]]'[t3], 2, tf, constraints3]
```

Out[46]=
```
{{InterpolatingFunction[ Domain: {{1.44, 20.}} Output: scalar ][t],
  InterpolatingFunction[ Domain: {{1.44, 20.}} Output: scalar ][t],
  InterpolatingFunction[ Domain: {{1.44, 20.}} Output: scalar ][t]}, {}}
```

In[47]:=
```
Clear[L1, solnlist, zeros, constraints4, sol, t4];
```

In[48]:=
```
t4 = FindRoot[Sin[step3[[1, 1]]] == -1, {t, 1.7}][[1, 2]]
```

InterpolatingFunction: Input value {−0.726836} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {−0.726836} lies outside the range of data in the interpolating function. Extrapolation will be used.

FindRoot: The line search decreased the step size to within tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient decrease in the merit function. You may need more than MachinePrecision digits of working precision to meet these tolerances.

Out[48]=
```
-0.726836
```

In[49]:=
```
θstop = step3[[1, 1, 0]][t4]
```

InterpolatingFunction: Input value {−0.726836} lies outside the range of data in the interpolating function. Extrapolation will be used.

Out[49]= -2.77746×10^{14}

In[50]:=
```
G3 := θ[t] - θ[t4];
```

In[51]:=
```
constraints4 = {{β[t], G3}};
```

In[52]:=
```
step4 = solvestep[L, t4, step3[[1, 1, 0]][t4], step3[[1, 2, 0]][t4], step3[[1, 3, 0]][t4],
  0, step3[[1, 2, 0]]'[t4], step3[[1, 3, 0]]'[t4], 2, tf, constraints4]
```

InterpolatingFunction: Input value {−0.726836} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {−0.726836} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {−0.726836} lies outside the range of data in the interpolating function. Extrapolation will be used.

General: Further output of InterpolatingFunction::dmval will be suppressed during this calculation.

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

NDSolve: Unable to find initial conditions that satisfy the residual function within specified tolerances. Try giving initial conditions for both values and derivatives of the functions.

Part: Part 1 of {} does not exist.

Part: Part 1 of {} does not exist.

Part: Part 1 of {} does not exist.

General: Further output of Part::partw will be suppressed during this calculation.

Out[52]=
```
{{{}[[1, 1, 2]], {}[[1, 2, 2]]}, {}}
```

In[53]:=

In[54]:=
```
θpw[t_] = Piecewise[{{step1[[1, 1]], 0 ≤ t ≤ t2}, {step2[[1, 1]], t2 ≤ t ≤ t3}, {step3[[1, 1]], t3 ≤ t ≤ tf}}]
```

In[55]:=
```
ϕpw[t_] = Piecewise[{{step1[[1, 2]], 0 ≤ t ≤ t2}, {step2[[1, 2]], t2 ≤ t ≤ t3}, {step3[[1, 2]], t3 ≤ t ≤ tf}}]
```

In[56]:=
```
αpw[t_] = Piecewise[{{step1[[1, 3]], 0 ≤ t ≤ t2}, {step2[[1, 3]], t2 ≤ t ≤ t3}, {step3[[1, 3]], t3 ≤ t ≤ tf}}]
```

In[57]:=
```
λpw[t_] = Piecewise[{{step1[[1, 4]], 0 ≤ t ≤ t2}, {0, t2 ≤ t ≤ t3}, {0, t3 ≤ t ≤ tf}}]
```

In[58]:=
```
γpw[t_] = Piecewise[{{step1[[1, 5]], 0 ≤ t ≤ t2}, {step2[[1, 4]], t2 ≤ t ≤ t3}, {0, t3 ≤ t ≤ tf}}]
```

In[59]:=
```
θpw[0]
```

Out[59]=
$$-2.77399 \times 10^{-14}$$

In[60]:=
```
θpw[tf]
```

Out[60]=
```
13.4246
```

In[61]:=
```
Plot[θpw[t], {t, 0, tlaunch + 1}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "θ[t]", "Throwing Arm Angle"}]
(*Plot[θpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","θ'[t]","Throwing Arm Angular Velocity"}]*)
```

Plot: Limiting value 1 + tlaunch in {t, 0, 1 + tlaunch} is not a machine-sized real number.

Out[61]=
```
Plot[θpw[t], {t, 0, tlaunch + 1}, PlotRange → All,
 Frame → True, FrameLabel → {t, θ[t], Throwing Arm Angle}]
```

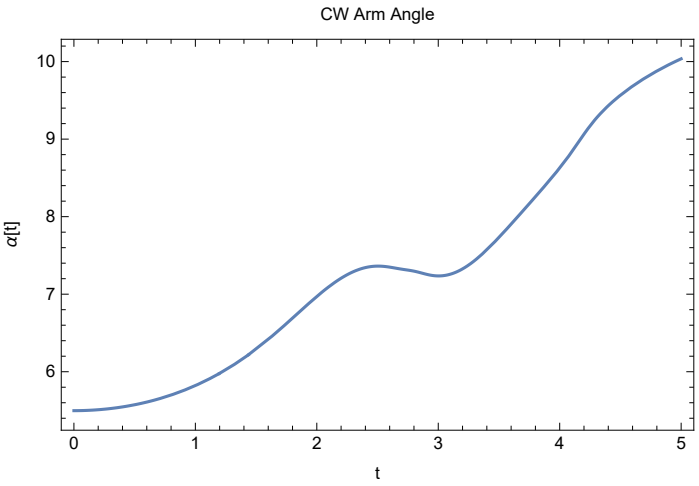
In[62]:=
```
Plot[ϕpw[t], {t, 0, tlaunch + 1}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "ϕ[t]", "Sling Angle"}]
(*Plot[ϕpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","ϕ'[t]","Sling Angular Velocity"}]*)
```

Plot: Limiting value 1 + tlaunch in {t, 0, 1 + tlaunch} is not a machine-sized real number.

Out[62]=
```
Plot[ϕpw[t], {t, 0, tlaunch + 1}, PlotRange → All,
 Frame → True, FrameLabel → {t, ϕ[t], Sling Angle}]
```

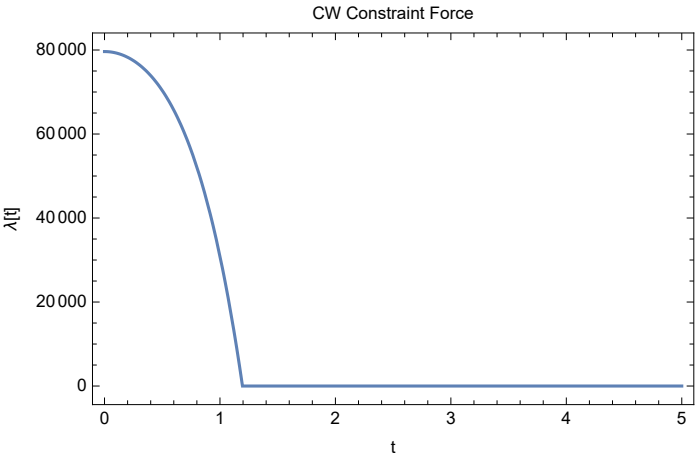
In[63]:=
```
Plot[αpw[t], {t, 0, 5}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "α[t]", "CW Arm Angle"}]
(*Plot[αpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","α'[t]","CW Arm Angular Velocity"}]*)
```

Out[63]=



In[64]:=
```
Plot[λpw[t], {t, 0, 5}, PlotRange → All, Frame → True,
 FrameLabel → {"t", "λ[t]", "CW Constraint Force"}]
```

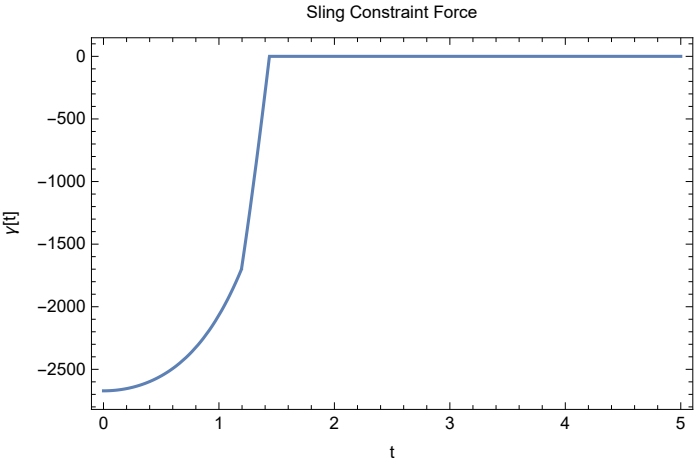
Out[64]=



In[65]:=

In[66]:=
```
Plot[γpw[t], {t, 0, 5}, PlotRange → All, Frame → True,
 FrameLabel → {"t", "γ[t]", "Sling Constraint Force"}]
```

Out[66]=



In[67]:=

In[68]:=

In[69]:=

In[70]:=
```
xtapp[t_] = l1 Cos[θpw[t]];
ytapp[t_] = -l1 Sin[θpw[t]];
xtacp[t_] = -l2 Cos[θpw[t]];
ytacp[t_] = l2 Sin[θpw[t]];
```

In[74]:=
```
xbp[t_] = l1 Cos[θpw[t]] + r Cos[ϕpw[t]];(*these are barrel coord*)
ybp[t_] = -l1 Sin[θpw[t]] - r Sin[ϕpw[t]];
xcwp[t_] = -l2 Cos[θpw[t]] + h Cos[αpw[t]];(*these are counter-weight coord*)
ycwp[t_] = l2 Sin[θpw[t]] - h Sin[αpw[t]];

Clear[x, y, xdrag, ydrag];
tlaunch = FindRoot[xbp'[t] == ybp'[t], {t, 2.74}][[1, 2]];
b = 20;
dragsoln = NDSolve[{
    xdrag[tlaunch] == xbp[tlaunch],
    ydrag[tlaunch] == ybp[tlaunch],
    xdrag'[tlaunch] == xbp'[tlaunch],
    ydrag'[tlaunch] == ybp'[tlaunch],
    xdrag''[t] == (-b/mb) xdrag'[t],
    ydrag''[t] == (-b/mb) ydrag'[t] - g},
   {xdrag, ydrag}, {t, tlaunch, tf}];
xd = dragsoln[[1, 1, 2]];
yd = dragsoln[[1, 2, 2]];

xbtotal[t_] = {xbp[t]  0 ≤ t ≤ tlaunch
               xd[t]   tlaunch < t ≤ tf};
ybtotal[t_] = {ybp[t]  0 ≤ t ≤ tlaunch
               yd[t]   tlaunch < t ≤ tf};
```

In[86]:=

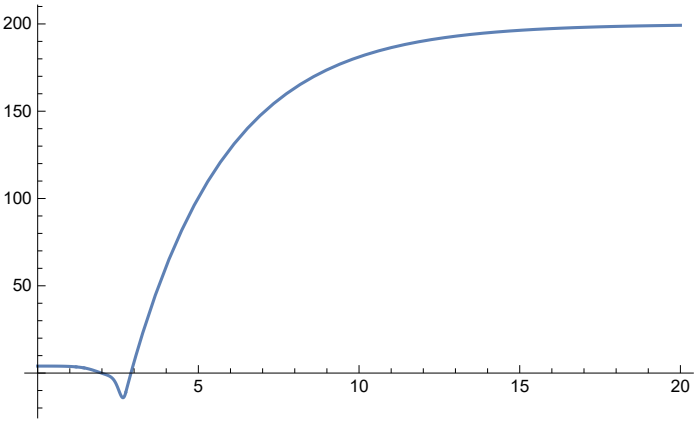
In[87]:=

In[88]:=

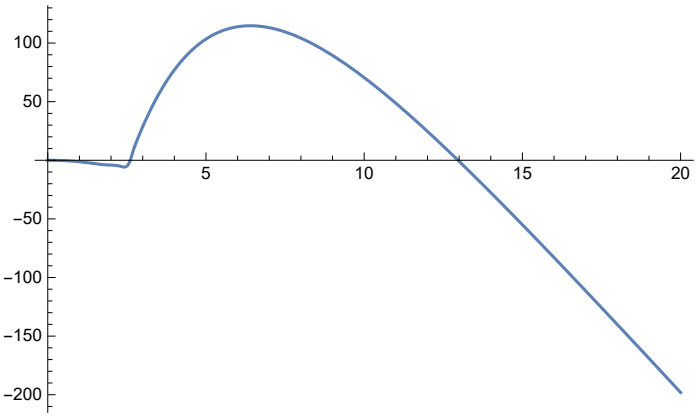
In[89]:=

```
Plot[xbtotal[t], {t, 0, tf}]
Plot[ybtotal[t], {t, 0, tf}]
ParametricPlot[{xbtotal[t], ybtotal[t]}, {t, 0, 14}]
```

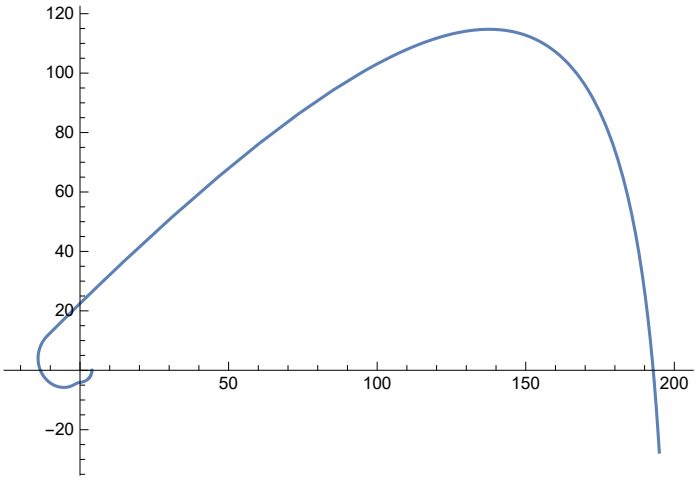
Out[89]=

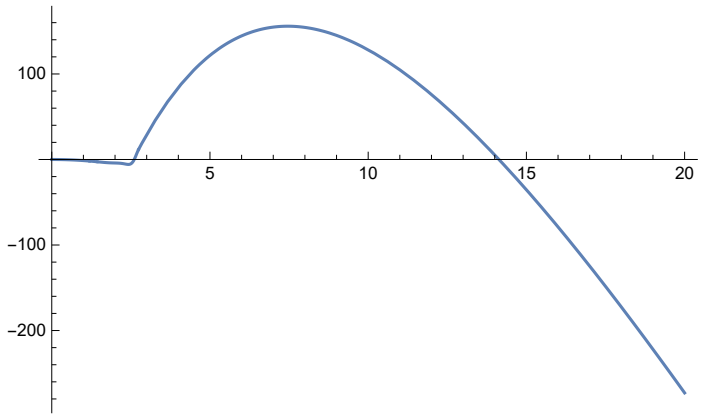


Out[90]=



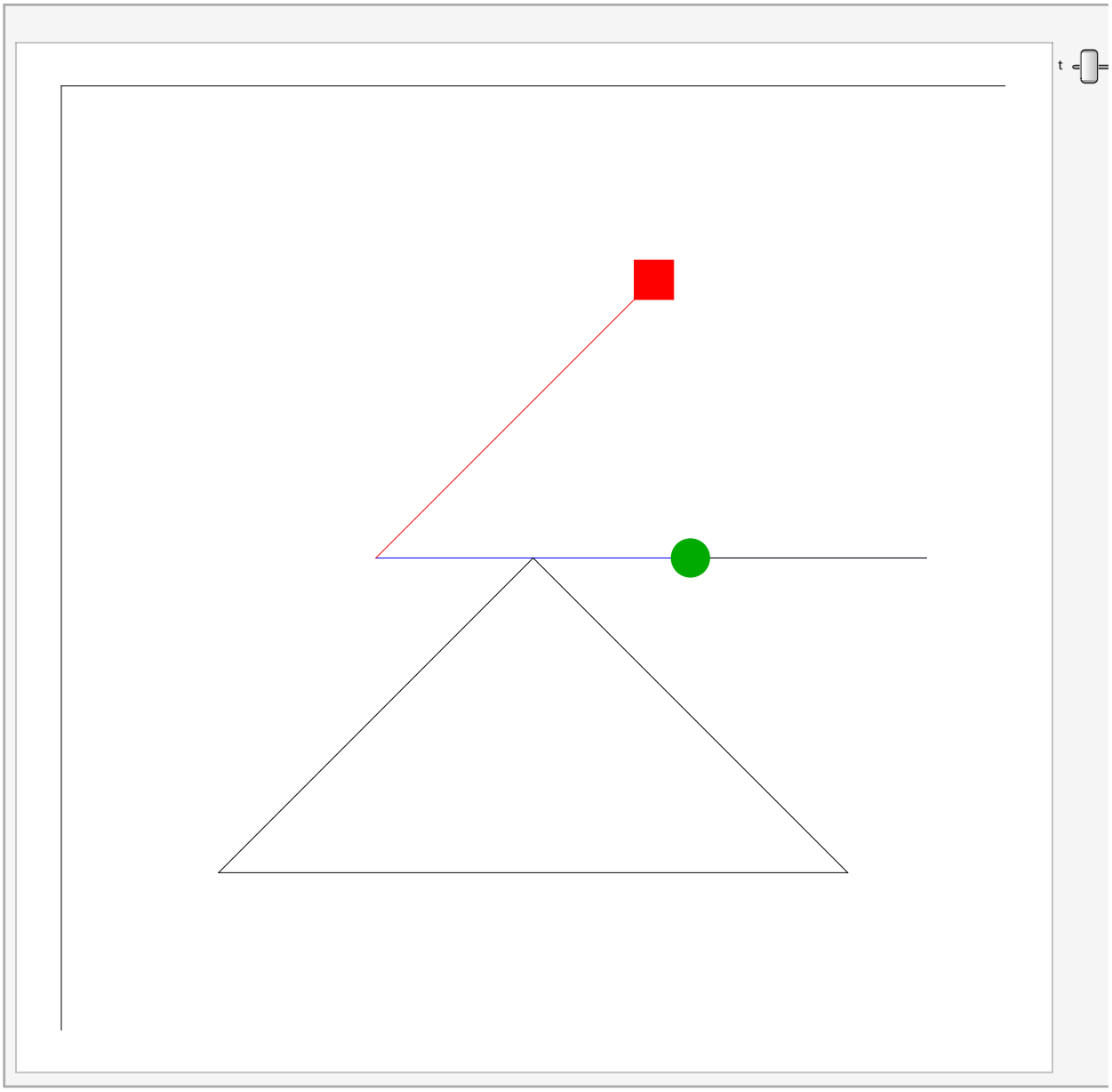
Out[91]=





In[92]:=
```
Manipulate[Graphics[{
   Line[{{-12, -12}, {-12, 12}}],
   Line[{{-12, 12}, {12, 12}}],
   Blue, Line[{{xtapp[t], ytapp[t]}, {xtacp[t], ytacp[t]}}],
   Black, Line[{{xtapp[t], ytapp[t]}, {xbp[t], ybp[t]}}],
   Red, Line[{{xtacp[t], ytacp[t]}, {xcwp[t], ycwp[t]}}],
   Black, Line[{{-8, -8}, {0, 0}}],
   Black, Line[{{0, 0}, {8, -8}}],
   Black, Line[{{-8, -8}, {8, -8}}],
   Red, Rectangle[{xcwp[t] - .5, ycwp[t] - .5}, {xcwp[t] + .5, ycwp[t] + .5}],
   Darker[Green], Disk[{xbtotal[t], ybtotal[t]}, .5]

  }, ImageSize → Large],
 {t, 0.0000, 14}]
```

Out[92]=

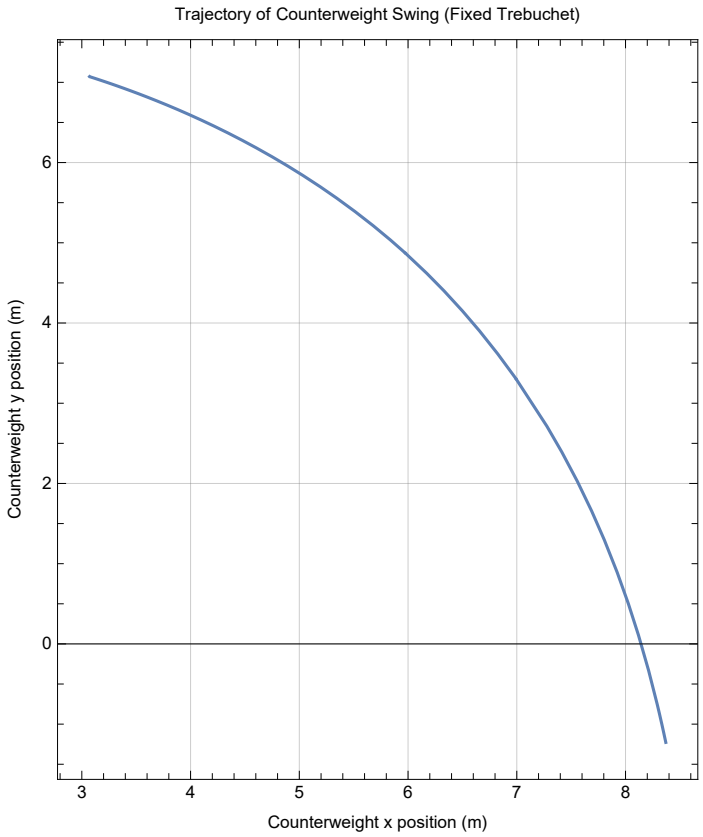


In[93]:=
```
Manipulate[Graphics[{
   Line[{{-6, -6}, {-6, 55}}],
   Line[{{-6, 55}, {350, 55}}],
   Blue, Line[{{xtapp[t], ytapp[t]}, {xtacp[t], ytacp[t]}}],
   Black, Line[{{xtapp[t], ytapp[t]}, {xbp[t], ybp[t]}}],
   Red, Line[{{xtacp[t], ytacp[t]}, {xcwp[t], ycwp[t]}}],
   Black, Line[{{-4, -6}, {0, 0}}],
   Black, Line[{{0, 0}, {4, -6}}],
   Black, Line[{{-4, -6}, {4, -6}}],
   Red, Rectangle[{xcwp[t] - .5, ycwp[t] - .5}, {xcwp[t] + .5, ycwp[t] + .5}],
   Darker[Green], Disk[{xbtotal[t], ybtotal[t]}, .5],

  }, ImageSize → Large],
 {t, 0.0000, 8}]
```

Out[93]=

In[94]:=
```
ParametricPlot[{xcwp[t], ycwp[t]}, {t, 0, 1.9}, AspectRatio → Full, Frame → True,
 FrameLabel → {"Counterweight x position (m)", "Counterweight y position (m)",
   "Trajectory of Counterweight Swing (Fixed Trebuchet)"}, GridLines → Automatic]
```

Out[94]=



In[95]:=
```
Clear[Tpw, Upw, Epw]
```

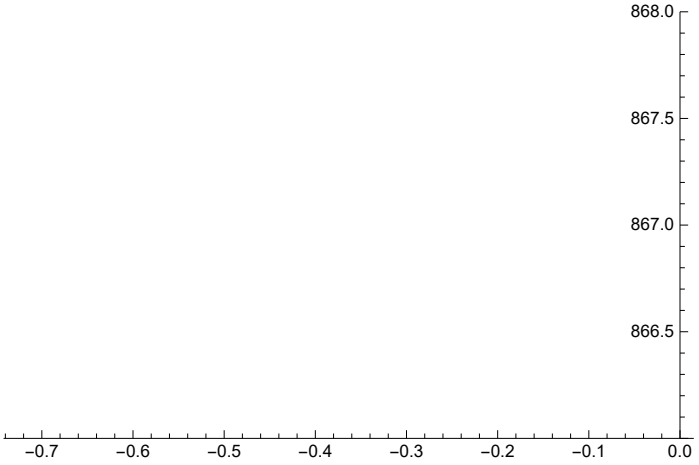
In[96]:=
```
ycmap[t_] := -(l1 - l2)/2 Sin[θpw[t]];
```

In[97]:=
```
Tpw[t_] = 1/2 mb (D[xbp[t], t]^2 + D[ybp[t], t]^2) +
   1/2 mcw (D[xcwp[t], t]^2 + D[ycwp[t], t]^2) + 1/2 IIa D[θpw[t], t]^2;
Upw[t_] = mb g ybp[t] + mcw g ycwp[t] + ma g ycmap[t];
```

In[99]:=
```
Epw[t_] = Tpw[t] + Upw[t];
```

In[100]:=
```
Plot[Epw[t], {t, 0, t4}, PlotRange → {866, 868}]
```

Out[100]=



In[101]:=
```
ttest = FindRoot[xbp'[t] == ybp'[t], {t, 1.63}][[1, 2]]
```

Out[101]= 1.48284

In[102]:=
```
θpw[ttest] + ϕpw[ttest]
```

Out[102]= 4.71262

In[103]:=
```
N[17 Pi / 4]
```

Out[103]= 13.3518

In[104]:=
```
Utoti = Upw[0]
```

Out[104]= 208 102.

In[105]:=
```
Tbf = 1/2 mb (xbp'[ttest]^2 + ybp'[ttest]^2)
```

Out[105]= 816.115

In[106]:=
```
KineticRatio = Tbf / Utoti
```

Out[106]= 0.00392172

In[107]:=

Notebook for the Optimum fulcrum position. This one was used to manually calculate launch velocities for certain lever arm ratios.

In[2433]:=
```
(*Here we create a function that solves the equations of motion in discrete sections,
defined when each constaint goes to zero*)
solvestep[L_, t0_, θ0_, ϕ0_, α0_, dθ0_, dϕ0_, dα0_, tguess_, tf_, constraints_] := (
  (*Add the constraints*)
  L1 = L;
  For[i = 1, i ≤ Length[constraints], i++,
   L1 = L1 + constraints[[i]][[1]] × constraints[[i]][[2]];];
  sol = {};
  (*NDSolve based on how many constraints*)
  Which[Length[constraints] == 2,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*first solve with two constraints*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0,
      D[constraints[[2, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]], constraints[[2, 1]]}, {t, t0, tf}],
   Length[constraints] == 1,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*then solve with one constraint*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]]}, {t, t0, tf}],
   Length[constraints] == 0,
   sol =
    NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*finally solve with no constraints*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0},
     {θ[t], ϕ[t], α[t]}, {t, t0, tf}]];
  (*Pull off solutions*)
  solnlist = {};
  For[i = 1, i ≤ Length[sol[[1]]], i++,
   AppendTo[solnlist, sol[[1, i, 2]]];
  ];
  (*Obtain first zeros so we can identify when
   each constraint goes to zero and splice them together later*)
  zeros = {};
```

```
  For[i = 4, i ≤ Length[solnlist], i++,
   AppendTo[zeros,
     FindRoot[solnlist[[i]], {t, tguess}]];
  ];
  (*Return solutions and zeros*)
  Return[{solnlist, zeros}];
 )
Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L, t, tf,
 EL, soln, xb, yb, xcw, ycw, xcma, ycma, λ, γ, G1, G2, x, y, ltot, k]
(*first testing fulcrum point (ratio between l1 and l2)*)
velslist = {};
lengthslist = {};
j = 9.9;

(*For[j=10, j<13,j=j+0.25,*)
Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L, t, tf, EL,
  soln, xb, yb, xcw, ycw, xcma, ycma, λ, γ, G1, G2, x, y, xbp, ybp, topt];
Print["current L1:", j];
g := 9.81;
mb := 60;
mcw := 3000;
ma := 1000;
l1 := j;
l2 := 14 - j;
h := 10;
r := 6;

T := 1/2 mb (D[xb, t]^2 + D[yb, t]^2) + 1/2 mcw (D[xcw, t]^2 + D[ycw, t]^2) + 1/2 IIa D[θ[t], t]^2;
IIa := 1/3 ma (l1^3 + l2^3)/(l1 + l2);
U := mb g yb + mcw g ycw + ma g ycma;
xb := l1 Cos[θ[t]] + r Cos[ϕ[t]];
yb := -l1 Sin[θ[t]] - r Sin[ϕ[t]];
xcw := -l2 Cos[θ[t]] + h Cos[α[t]];
ycw := l2 Sin[θ[t]] - h Sin[α[t]];
ycma := -(l1 - l2)/2 Sin[θ[t]];
G1 := θ[t] - α[t] + α[0];
G2 := θ[t] - ϕ[t] + ϕ[0];
L := T - U;
EL[L_, x_] := D[L, x[t]] - D[L, x'[t], t] == 0;
Clear[L1, solnlist, zeros, sol, constraints1, tf];
tf = 15;
constraints1 = {{λ[t], G1}, {γ[t], G2}};
step1 = solvestep[L, 0, 0, Pi, 7 Pi/4, 0, 0, 0, 1, tf, constraints1];
Clear[L2, solnlist, zeros, constraints2, sol, t2, step2];
constraints2 = {{γ[t], G2}};
t2 = step1[[2, 1, 1, 2]];
step2 = solvestep[L, t2, step1[[1, 1, 0]][t2],
```

```
   step1[[1, 2, 0]][t2], step1[[1, 3, 0]][t2], step1[[1, 1, 0]]'[t2],
   step1[[1, 2, 0]]'[t2], step1[[1, 3, 0]]'[t2], 2, tf, constraints2];
Clear[L1, solnlist, zeros, constraints3, sol, t3];
constraints3 = {};
t3 = step2[[2, 1, 1, 2]];
step3 = solvestep[L, t3, step2[[1, 1, 0]][t3],
   step2[[1, 2, 0]][t3], step2[[1, 3, 0]][t3], step2[[1, 1, 0]]'[t3],
   step2[[1, 2, 0]]'[t3], step2[[1, 3, 0]]'[t3], 2, tf, constraints3];

θpw[t_] = Piecewise[{{step1[[1, 1]], 0 ≤ t < t2}, {step2[[1, 1]], t2 ≤ t < t3}, {step3[[1, 1]], t3 ≤ t ≤ tf}}]
ϕpw[t_] = Piecewise[{{step1[[1, 2]], 0 ≤ t < t2}, {step2[[1, 2]], t2 ≤ t < t3}, {step3[[1, 2]], t3 ≤ t ≤ tf}}]
αpw[t_] = Piecewise[{{step1[[1, 3]], 0 ≤ t < t2}, {step2[[1, 3]], t2 ≤ t < t3}, {step3[[1, 3]], t3 ≤ t ≤ tf}}]
λpw[t_] = Piecewise[{{step1[[1, 4]], 0 ≤ t < t2}, {0, t2 ≤ t < t3}, {0, t3 ≤ t ≤ tf}}]
γpw[t_] = Piecewise[{{step1[[1, 5]], 0 ≤ t < t2}, {step2[[1, 4]], t2 ≤ t < t3}, {0, t3 ≤ t ≤ tf}}]
xbp[t_] = l1 Cos[θpw[t]] + r Cos[ϕpw[t]];
ybp[t_] = -l1 Sin[θpw[t]] - r Sin[ϕpw[t]];

Plot[{xbp'[t], ybp'[t]}, {t, 0, 2.74}]
FindRoot[xbp'[t] == ybp'[t], {t, 3.1}][[1, 2]]

(*]*)
```

In[2481]:=
```
topt = FindRoot[xbp'[t] == ybp'[t], {t, 2.75}][[1, 2]];
speed = Sqrt[(xbp'[topt])^2 + (ybp'[topt])^2];
Print["toptimal:", topt];
Print["launch velocity:", speed];
AppendTo[velslist, speed];
AppendTo[lengthslist, j];
```

In[2487]:=
```
theList = Transpose[{lengthslist, velslist}]
```

In[2488]:=

In[2489]:=
```
ListPlot[theList, Frame → True, GridLines -> Automatic,
 FrameLabel → {"Length of L1 (m)", "Launch Speed (m/s)"},
 PlotLabel → "Launch Speed vs Length of L1"]
```

Notebook for Optimal Fulcrum placement. Has a loop to run through the NDSolve with many different lever arm ratios.

```
(*Here we create a function that solves the equations of motion in discrete sections,
defined when each constaint goes to zero*)
solvestep[L_, t0_, θ0_, ϕ0_, α0_, dθ0_, dϕ0_, dα0_, tguess_, tf_, constraints_] := (
  (*Add the constraints*)
  L1 = L;
  For[i = 1, i ≤ Length[constraints], i++,
   L1 = L1 + constraints[[i]][[1]] × constraints[[i]][[2]];];
  sol = {};
  (*NDSolve based on how many constraints*)
  Which[Length[constraints] == 2,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*first solve with two constraints*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0,
      D[constraints[[2, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]], constraints[[2, 1]]}, {t, t0, tf}],
   Length[constraints] == 1,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*then solve with one constraint*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]]}, {t, t0, tf}],
   Length[constraints] == 0,
   sol =
    NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], (*finally solve with no constraints*)
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0},
     {θ[t], ϕ[t], α[t]}, {t, t0, tf}]];
  (*Pull off solutions*)
  solnlist = {};
  For[i = 1, i ≤ Length[sol[[1]]], i++,
   AppendTo[solnlist, sol[[1, i, 2]]];
  ];
  (*Obtain first zeros so we can identify when
   each constraint goes to zero and splice them together later*)
  zeros = {};
```

```
  For[i = 4, i ≤ Length[solnlist], i++,
   AppendTo[zeros,
     FindRoot[solnlist[[i]], {t, tguess}]];
  ];
  (*Return solutions and zeros*)
  Return[{solnlist, zeros}];
 )
Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L, t, tf,
 EL, soln, xb, yb, xcw, ycw, xcma, ycma, λ, γ, G1, G2, x, y, ltot, k]
(*first testing fulcrum point (ratio between l1 and l2)*)
velslist = {};
lengthslist = {};

For[j = 9.85, j < 12.85, j = j + 0.05,
 Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L, t, tf, EL,
  soln, xb, yb, xcw, ycw, xcma, ycma, λ, γ, G1, G2, x, y, xbp, ybp, topt];
 Print["current L1:", j];
 g := 9.81;
 mb := 60;
 mcw := 3000;
 ma := 1000;
 l1 := j;
 l2 := 14 - j;
 h := 10;
 r := 6;

 T := 1/2 mb (D[xb, t]^2 + D[yb, t]^2) + 1/2 mcw (D[xcw, t]^2 + D[ycw, t]^2) + 1/2 IIa D[θ[t], t]^2;
 IIa := 1/3 ma ((l1^3 + l2^3)/(l1 + l2));
 U := mb g yb + mcw g ycw + ma g ycma;
 xb := l1 Cos[θ[t]] + r Cos[ϕ[t]];
 yb := -l1 Sin[θ[t]] - r Sin[ϕ[t]];
 xcw := -l2 Cos[θ[t]] + h Cos[α[t]];
 ycw := l2 Sin[θ[t]] - h Sin[α[t]];
 ycma := -(l1 - l2)/2 Sin[θ[t]];
 G1 := θ[t] - α[t] + α[0];
 G2 := θ[t] - ϕ[t] + ϕ[0];
 L := T - U;
 EL[L_, x_] := D[L, x[t]] - D[L, x'[t], t] == 0;
 Clear[L1, solnlist, zeros, sol, constraints1, tf];
 tf = 15;
 constraints1 = {{λ[t], G1}, {γ[t], G2}};
 step1 = solvestep[L, 0, 0, Pi, 7 Pi/4, 0, 0, 0, 1, tf, constraints1];
 Clear[L2, solnlist, zeros, constraints2, sol, t2, step2];
 constraints2 = {{γ[t], G2}};
 t2 = step1[[2, 1, 1, 2]];
 step2 = solvestep[L, t2, step1[[1, 1, 0]][t2],
   step1[[1, 2, 0]][t2], step1[[1, 3, 0]][t2], step1[[1, 1, 0]]'[t2],
```

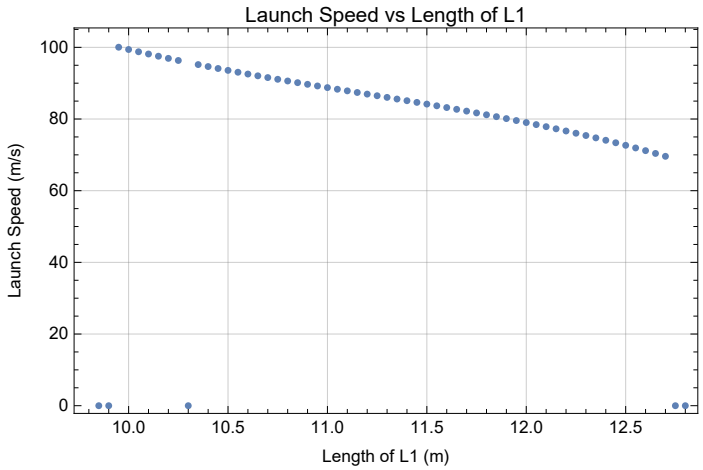
```
  step1[[1, 2, 0]]'[t2], step1[[1, 3, 0]]'[t2], 2, tf, constraints2];
Clear[L1, solnlist, zeros, constraints3, sol, t3];
constraints3 = {};
t3 = step2[[2, 1, 1, 2]];
step3 = solvestep[L, t3, step2[[1, 1, 0]][t3],
  step2[[1, 2, 0]][t3], step2[[1, 3, 0]][t3], step2[[1, 1, 0]]'[t3],
  step2[[1, 2, 0]]'[t3], step2[[1, 3, 0]]'[t3], 2, tf, constraints3];

θpw[t_] = Piecewise[{{step1[[1, 1]], 0 ≤ t < t2}, {step2[[1, 1]], t2 ≤ t < t3}, {step3[[1, 1]], t3 ≤ t ≤ tf}}]

ϕpw[t_] = Piecewise[{{step1[[1, 2]], 0 ≤ t < t2}, {step2[[1, 2]], t2 ≤ t < t3}, {step3[[1, 2]], t3 ≤ t ≤ tf}}]

αpw[t_] = Piecewise[{{step1[[1, 3]], 0 ≤ t < t2}, {step2[[1, 3]], t2 ≤ t < t3}, {step3[[1, 3]], t3 ≤ t ≤ tf}}]

λpw[t_] = Piecewise[{{step1[[1, 4]], 0 ≤ t < t2}, {0, t2 ≤ t < t3}, {0, t3 ≤ t ≤ tf}}]

γpw[t_] = Piecewise[{{step1[[1, 5]], 0 ≤ t < t2}, {step2[[1, 4]], t2 ≤ t < t3}, {0, t3 ≤ t ≤ tf}}]

xbp[t_] = l1 Cos[θpw[t]] + r Cos[ϕpw[t]];
ybp[t_] = -l1 Sin[θpw[t]] - r Sin[ϕpw[t]];

topt = FindRoot[xbp'[t] == ybp'[t], {t, 2.65}][[1, 2]];

speed = Sqrt[(xbp'[topt])^2 + (ybp'[topt])^2];
Print["toptimal:", topt];
Print["launch velocity:", speed];
AppendTo[velslist, speed];
AppendTo[lengthslist, j];
]
```

In[2002]:=
```
theList = Transpose[{lengthslist, velslist}]
```

Out[2002]=
```
{{9.85, 0.}, {9.9, 0.}, {9.95, 100.038}, {10., 99.4005}, {10.05, 98.7682},
 {10.1, 98.1439}, {10.15, 97.5295}, {10.2, 96.9264}, {10.25, 96.3356}, {10.3, 0.},
 {10.35, 95.1925}, {10.4, 94.6404}, {10.45, 94.1007}, {10.5, 93.5731}, {10.55, 93.0567},
 {10.6, 92.5508}, {10.65, 92.0545}, {10.7, 91.5669}, {10.75, 91.0869}, {10.8, 90.6137},
 {10.85, 90.1461}, {10.9, 89.6832}, {10.95, 89.224}, {11., 88.7676}, {11.05, 88.3131},
 {11.1, 87.8595}, {11.15, 87.406}, {11.2, 86.9518}, {11.25, 86.496}, {11.3, 86.0379},
 {11.35, 85.5767}, {11.4, 85.1119}, {11.45, 84.6426}, {11.5, 84.1682}, {11.55, 83.6881},
 {11.6, 83.2017}, {11.65, 82.7085}, {11.7, 82.2077}, {11.75, 81.6989}, {11.8, 81.1814},
 {11.85, 80.6548}, {11.9, 80.1183}, {11.95, 79.5715}, {12., 79.0136}, {12.05, 78.4442},
 {12.1, 77.8623}, {12.15, 77.2674}, {12.2, 76.6587}, {12.25, 76.0352}, {12.3, 75.3962},
 {12.35, 74.7405}, {12.4, 74.0672}, {12.45, 73.3752}, {12.5, 72.6631}, {12.55, 71.9298},
 {12.6, 71.1742}, {12.65, 70.3953}, {12.7, 69.5928}, {12.75, 0.}, {12.8, 0.}}
```

In[2003]:=
```
ListPlot[theList, Frame → True, GridLines -> Automatic,
 FrameLabel → {"Length of L1 (m)", "Launch Speed (m/s)"},
 PlotLabel → "Launch Speed vs Length of L1"]
```

Out[2003]=



NDSolve Module

In[228]:=

```
solvestep[L_, t0_, θ0_, ϕ0_, α0_, xp0_,
  dθ0_, dϕ0_, dα0_, dxp0_, tguess_, tf_, constraints_] := (
  (*Add the constraints*)
  L1 = L;
  For[i = 1, i ≤ Length[constraints], i++,
   L1 = L1 + constraints[[i]][[1]] × constraints[[i]][[2]];];
  sol = {};
  (*NDSolve based on how many constraints*)
  Which[Length[constraints] == 2,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], EL[L1, xp],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      xp[t0] == xp0,
      xp'[t0] == dxp0,
      D[constraints[[1, 2]], t, t] == 0,
      D[constraints[[2, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], xp[t], constraints[[1, 1]], constraints[[2, 1]]}, {t, t0, tf}],
   Length[constraints] == 1,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], EL[L1, xp],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      xp[t0] == xp0,
      xp'[t0] == dxp0,
      D[constraints[[1, 2]], t, t] == 0},

     {θ[t], ϕ[t], α[t], xp[t], constraints[[1, 1]]}, {t, t0, tf}],
   Length[constraints] == 0,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α], EL[L1, xp],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      xp[t0] == xp0,
      xp'[t0] == dxp0},
     {θ[t], ϕ[t], α[t], xp[t]}, {t, t0, tf}]];
  (*Pull off solutions*)
```

```
  solnlist = {};
  For[i = 1, i ≤ Length[sol[[1]]], i++,
   AppendTo[solnlist, sol[[1, i, 2]]];
  ];
  (*Obtain first zeros*)
  zeros = {};
  For[i = 5, i ≤ Length[solnlist], i++,
   AppendTo[zeros,
     FindRoot[solnlist[[i]], {t, tguess}]];
  ];
  (*Return solutions and zeros*)
  Return[{solnlist, zeros}];
 )
```

In[229]:= `ListPlot`

Out[229]= ListPlot

Kinetic energy

In[230]:=
```
Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L4,
 t, tf, EL, soln, xb, yb, xcw, ycw, xcma, ycma, λ, ɣ, G1, G2, x, y]
```

In[231]:=
$$T := \frac{1}{2}\,mb\,\left(D[xb,t]^2 + D[yb,t]^2\right) + \frac{1}{2}\,mcw\,\left(D[xcw,t]^2 + D[ycw,t]^2\right) + \frac{1}{2}\,ma\,\left(D[xa,t]^2 + D[ya,t]^2\right) + \frac{1}{2}\,IIa\,D[\theta[t],t]^2$$

Note, if you call moment of inertia I you generate errors because this is the imaginary number

In[232]:=
$$IIa := \frac{1}{12}\,ma\,(l1+l2)^2$$

Potential energy

In[233]:=
```
U := mb g yb + mcw g ycw + ma g ya
```

Define transformation to generalized coordinates

In[234]:=
```
xb := xp[t] + l1 Cos[θ[t]] + r Cos[ϕ[t]] (*these are barrel coord*)
yb := -l1 Sin[θ[t]] - r Sin[ϕ[t]]
xcw := xp[t] - l2 Cos[θ[t]] + h Cos[α[t]] (*these are counter-weight coord*)
ycw := l2 Sin[θ[t]] - h Sin[α[t]]
xa := xp[t] + (l1 - l2)/2 Cos[θ[t]]
ya := -(l1 - l2)/2 Sin[θ[t]] (*these are center of mass coord*)
```

Define Constraint Functions:

In[240]:=
```
G1 := θ[t] - α[t] + α[0] (*counter weight*)
```

In[241]:=
```
G2 := θ[t] - ϕ[t] + ϕ[0] (*barrel*)
```

Let' s see what T & U look like ... It looks nice. Doing this in Mathematica saves the pain of differentiating, squaring etc. Try a few lines of this to get a feel for how nasty it is.

In[242]:=
```
Simplify[T]
```

Out[242]=
$$\frac{1}{24}\left((l1+l2)^2\,ma\,\theta'[t]^2+3\,ma\left((l1-l2)^2\,\text{Cos}[\theta[t]]^2\,\theta'[t]^2+4\left(xp'[t]-\frac{1}{2}(l1-l2)\,\text{Sin}[\theta[t]]\,\theta'[t]\right)^2\right)+12\,mcw\left((h\,\text{Cos}[\alpha[t]]\,\alpha'[t]-l2\,\text{Cos}[\theta[t]]\,\theta'[t])^2+(xp'[t]-h\,\text{Sin}[\alpha[t]]\,\alpha'[t]+l2\,\text{Sin}[\theta[t]]\,\theta'[t])^2\right)+12\,mb\left((l1\,\text{Cos}[\theta[t]]\,\theta'[t]+r\,\text{Cos}[\phi[t]]\,\phi'[t])^2+(-xp'[t]+l1\,\text{Sin}[\theta[t]]\,\theta'[t]+r\,\text{Sin}[\phi[t]]\,\phi'[t])^2\right)\right)$$

In[243]:=
```
Simplify[U]
```

Out[243]=
$$-\frac{1}{2}g\left(2\,h\,mcw\,\text{Sin}[\alpha[t]]+(l1\,(ma+2\,mb)-l2\,(ma+2\,mcw))\,\text{Sin}[\theta[t]]+2\,mb\,r\,\text{Sin}[\phi[t]]\right)$$

Define L

In[244]:=
```
L4 := T - U
```

In[245]:=
```
Simplify[L4]
```

Out[245]=
$$\frac{1}{24}\left(12\,g\,(l1-l2)\,ma\,\text{Sin}[\theta[t]]+24\,g\,mcw\,(h\,\text{Sin}[\alpha[t]]-l2\,\text{Sin}[\theta[t]])+24\,g\,mb\,(l1\,\text{Sin}[\theta[t]]+r\,\text{Sin}[\phi[t]])+(l1+l2)^2\,ma\,\theta'[t]^2+3\,ma\left((l1-l2)^2\,\text{Cos}[\theta[t]]^2\,\theta'[t]^2+4\left(xp'[t]-\frac{1}{2}(l1-l2)\,\text{Sin}[\theta[t]]\,\theta'[t]\right)^2\right)+12\,mcw\left((h\,\text{Cos}[\alpha[t]]\,\alpha'[t]-l2\,\text{Cos}[\theta[t]]\,\theta'[t])^2+(xp'[t]-h\,\text{Sin}[\alpha[t]]\,\alpha'[t]+l2\,\text{Sin}[\theta[t]]\,\theta'[t])^2\right)+12\,mb\left((l1\,\text{Cos}[\theta[t]]\,\theta'[t]+r\,\text{Cos}[\phi[t]]\,\phi'[t])^2+(-xp'[t]+l1\,\text{Sin}[\theta[t]]\,\theta'[t]+r\,\text{Sin}[\phi[t]]\,\phi'[t])^2\right)\right)$$

In[246]:=

In[247]:=
```
g := 9.81
mb := 60
mcw := 3000
ma := 1000
l1 := 10
l2 := 4
h := 10
r := 6
```

In[255]:=
```
EL[L_, x_] := D[L, x[t]] - D[L, x'[t], t] == 0;
```

In[256]:=
```
Clear[L1, solnlist, zeros, sol, constraints1, tf];
```

In[257]:=
```
tf = 20;
```

In[258]:=
```
constraints1 = {{λ[t], G1}, {ɣ[t], G2}};
```

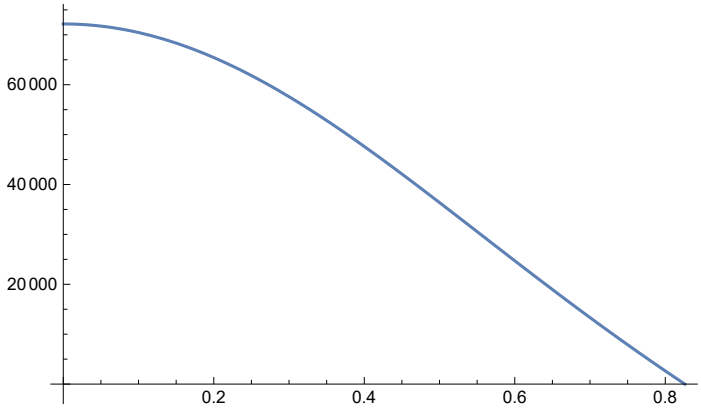
In[259]:=
```
step1 = solvestep[L4, 0, 0, Pi, 7 Pi/4, 0, 0, 0, 0, 0, 2, tf, constraints1]
```

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

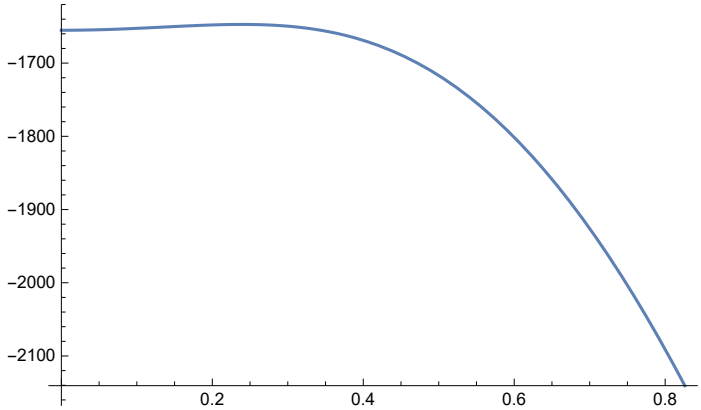
Out[259]=
```
{{InterpolatingFunction[ Domain: {{0., 20.}} Output: scalar  Data not in notebook; Store now » ][t],
  InterpolatingFunction[ Domain: {{0., 20.}} Output: scalar  Data not in notebook; Store now » ][t],
  InterpolatingFunction[ Domain: {{0., 20.}} Output: scalar  Data not in notebook; Store now » ][t],
  InterpolatingFunction[ Domain: {{0., 20.}} Output: scalar  Data not in notebook; Store now » ][t],
  InterpolatingFunction[ Domain: {{0., 20.}} Output: scalar  Data not in notebook; Store now » ][t],
  InterpolatingFunction[ Domain: {{0., 20.}} Output: scalar  Data not in notebook; Store now » ][t]},
 {{t → 0.826024}, {t → 2.00391}}}
```

In[260]:=
```
Plot[step1[[1, 5]], {t, 0, t2}]
Plot[step1[[1, 6]], {t, 0, t2}]
```

Out[260]=



Out[261]=



In[262]:=
```
Print[step1[[2]]]
```

```
{{t → 0.826024}, {t → 2.00391}}
```

In[263]:=
```
Clear[L2, solnlist, zeros, constraints2, sol, t2, step2];
```

In[264]:=
```
constraints2 = {{ϒ[t], G2}};
```

In[265]:=
```
t2 = step1[[2, 1, 1, 2]];
Print[t2];
```

```
0.826024
```

In[267]:=
```
step2 = solvestep[L4, t2, step1[[1, 1, 0]][t2], step1[[1, 2, 0]][t2], step1[[1, 3, 0]][t2],
  step1[[1, 4, 0]][t2], step1[[1, 1, 0]]'[t2], step1[[1, 2, 0]]'[t2],
  step1[[1, 3, 0]]'[t2], step1[[1, 4, 0]]'[t2], 1, tf, constraints2]
```

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

InterpolatingFunction: Input value {−19.} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {−1.} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {0.} lies outside the range of data in the interpolating function. Extrapolation will be used.

General: Further output of InterpolatingFunction::dmval will be suppressed during this calculation.

Out[267]=
```
{{InterpolatingFunction[
```
Domain: {{0.826, 20.}}
Output: scalar

Data not in notebook; Store now »
```
][t],
InterpolatingFunction[
```
Domain: {{0.826, 20.}}
Output: scalar

Data not in notebook; Store now »
```
][t],
InterpolatingFunction[
```
Domain: {{0.826, 20.}}
Output: scalar

Data not in notebook; Store now »
```
][t],
InterpolatingFunction[
```
Domain: {{0.826, 20.}}
Output: scalar

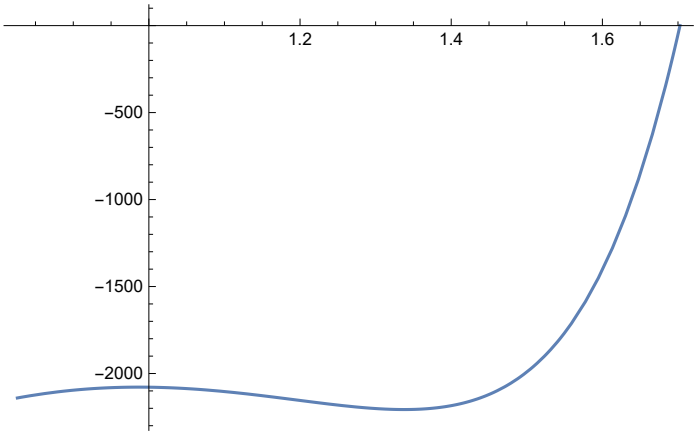
Data not in notebook; Store now »
```
][t],
InterpolatingFunction[
```
Domain: {{0.826, 20.}}
Output: scalar

Data not in notebook; Store now »
```
][t]}, {{t → 1.70266}}}
```

In[268]:=

In[269]:=
```
Plot[step2[[1, 5]], {t, t2, t3}, PlotRange → All]
```

Out[269]=



In[270]:=
```
Print[step2[[2]]]
```

```
{{t → 1.70266}}
```

In[271]:=
```
Clear[L1, solnlist, zeros, constraints3, sol, t3];
```

In[272]:=
```
constraints3 = {};
```

In[273]:=
```
t3 = step2[[2, 1, 1, 2]];
Print[t3]
```

```
1.70266
```

In[275]:=
```
step3 = solvestep[L4, t3, step2[[1, 1, 0]][t3], step2[[1, 2, 0]][t3], step2[[1, 3, 0]][t3],
  step2[[1, 4, 0]][t3], step2[[1, 1, 0]]'[t3], step2[[1, 2, 0]]'[t3],
  step2[[1, 3, 0]]'[t3], step2[[1, 4, 0]]'[t3], 2, tf, constraints3]
```

Out[275]=
```
{{InterpolatingFunction[  Domain: {{1.7, 20.}}
                                                                            Output: scalar ][t],

  InterpolatingFunction[  Domain: {{1.7, 20.}}
                                                                            Output: scalar ][t],

  InterpolatingFunction[  Domain: {{1.7, 20.}}
                                                                            Output: scalar ][t],

  InterpolatingFunction[  Domain: {{1.7, 20.}}
                                                                            Output: scalar ][t]}, {}}
```

In[276]:=
```
θpw[t_] = { step1[[1, 1]]  0 ≤ t ≤ t2
          { step2[[1, 1]]  t2 ≤ t ≤ t3;
          { step3[[1, 1]]  t3 ≤ t ≤ tf
```

In[277]:=
$$\phi pw[t\_] = \begin{cases} step1[[1,2]] & 0 \le t \le t2 \\ step2[[1,2]] & t2 \le t \le t3; \\ step3[[1,2]] & t3 \le t \le tf \end{cases}$$

In[278]:=
$$\alpha pw[t\_] = \begin{cases} step1[[1,3]] & 0 \le t \le t2 \\ step2[[1,3]] & t2 \le t \le t3; \\ step3[[1,3]] & t3 \le t \le tf \end{cases}$$

In[279]:=
$$xppw[t\_] = \begin{cases} step1[[1,4]] & 0 \le t \le t2 \\ step2[[1,4]] & t2 \le t \le t3; \\ step3[[1,4]] & t3 \le t \le tf \end{cases}$$

In[280]:=
$$\lambda pw[t\_] = \begin{cases} step1[[1,5]] & 0 \le t \le t2 \\ 0 & t2 \le t \le t3; \\ 0 & t3 \le t \le tf \end{cases}$$

In[281]:=
$$\gamma pw[t\_] = \begin{cases} step1[[1,6]] & 0 \le t \le t2 \\ step2[[1,5]] & t2 \le t \le t3; \\ 0 & t3 \le t \le tf \end{cases}$$

In[282]:=
```
θpw[0]
```

Out[282]=
$$-7.40926 \times 10^{-13}$$

In[283]:=
```
xppw[0]
```

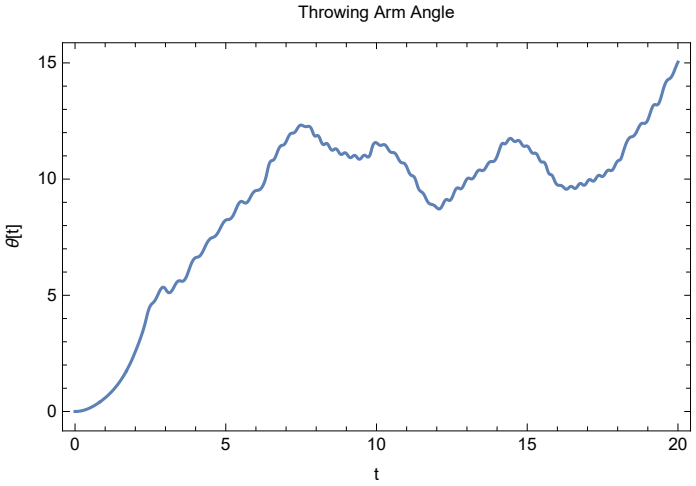
Out[283]=
```
0.
```

In[284]:=
```
xppw[tf]
```

Out[284]=
```
-0.0675632
```

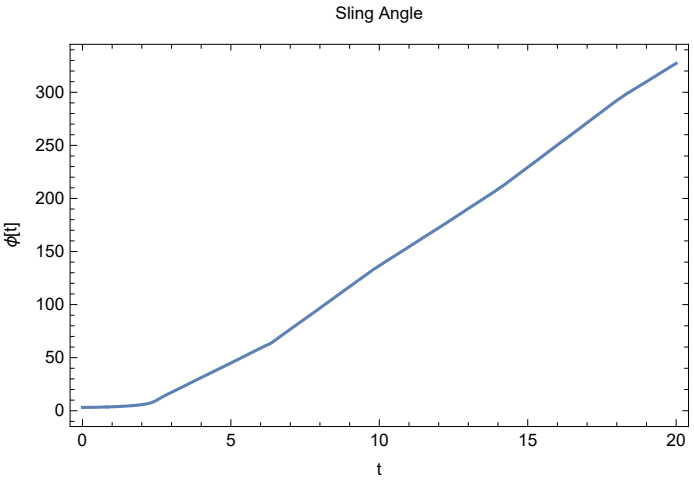
In[285]:=
```
Plot[θpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "θ[t]", "Throwing Arm Angle"}]
(*Plot[θpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","θ'[t]","Throwing Arm Angular Velocity"}]*)
```

Out[285]=



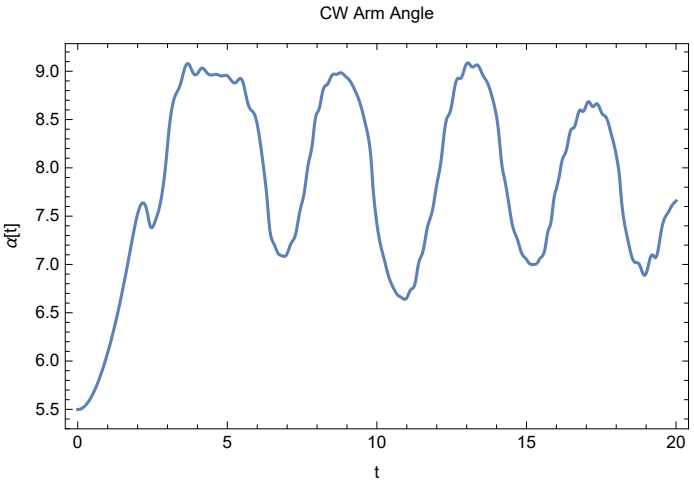
In[286]:=
```
Plot[ϕpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "ϕ[t]", "Sling Angle"}]
(*Plot[ϕpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","ϕ'[t]","Sling Angular Velocity"}]*)
```

Out[286]=



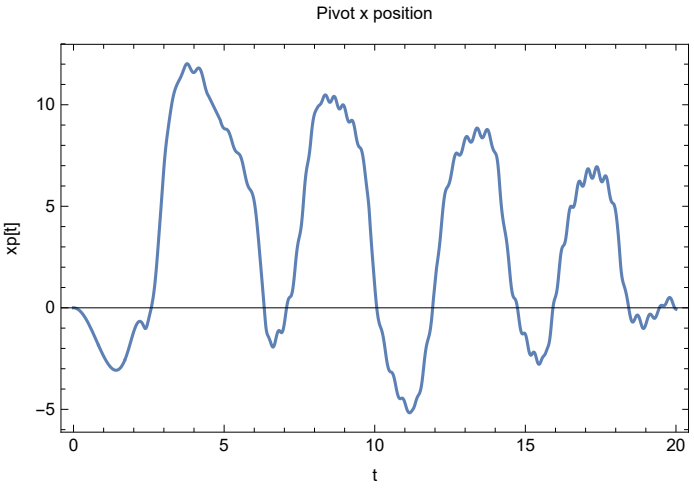
In[287]:=
```
Plot[αpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "α[t]", "CW Arm Angle"}]
(*Plot[αpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","α'[t]","CW Arm Angular Velocity"}]*)
```

Out[287]=



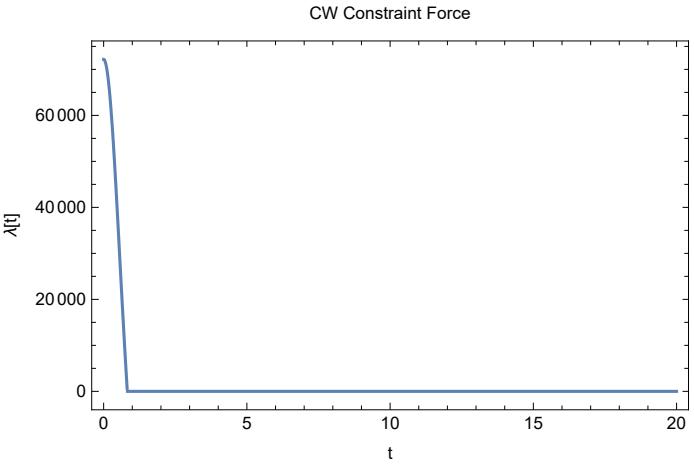
In[288]:=
```
Plot[xppw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "xp[t]", "Pivot x position"}]
(*Plot[xppw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","xp'[t]","Pivit x Velocity"}]*)
```

Out[288]=



In[289]:=
```
Plot[λpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "λ[t]", "CW Constraint Force"}]
```

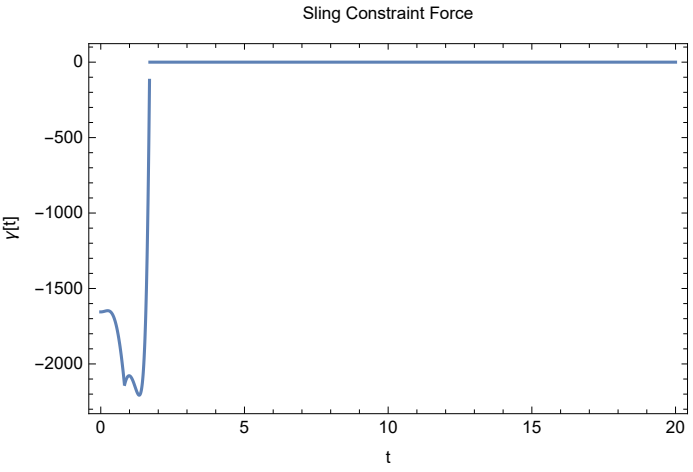
Out[289]=



In[290]:=

In[291]:=
```
Plot[γpw[t], {t, 0, tf}, PlotRange → All, Frame → True,
 FrameLabel → {"t", "γ[t]", "Sling Constraint Force"}]
```

Out[291]=



In[292]:=

In[293]:=
```
Clear[tlaunch]
```

In[294]:=

In[295]:=
```
xtapp[t_] = xppw[t] + l1 Cos[θpw[t]];
ytapp[t_] = -l1 Sin[θpw[t]];
xtacp[t_] = xppw[t] - l2 Cos[θpw[t]];
ytacp[t_] = l2 Sin[θpw[t]];
```

In[299]:=
```
xbp[t_] = xppw[t] + l1 Cos[θpw[t]] + r Cos[ϕpw[t]];(*these are barrel coord*)
ybp[t_] = -l1 Sin[θpw[t]] - r Sin[ϕpw[t]];
xcwp[t_] = xppw[t] - l2 Cos[θpw[t]] + h Cos[αpw[t]];(*these are counter-weight coord*)
ycwp[t_] = l2 Sin[θpw[t]] - h Sin[αpw[t]];
tlaunch = FindRoot[xbp'[t] == ybp'[t], {t, 2.51}][[1, 2]]
xbtotal[t_] = {xbp[t], 0 ≤ t ≤ tlaunch; xbp'[tlaunch] (t - tlaunch) + xbp[tlaunch], tlaunch < t ≤ tf};
ybtotal[t_] = {ybp[t], 0 ≤ t ≤ tlaunch; ybp'[tlaunch] (t - tlaunch) - g/2 (t - tlaunch)^2 + ybp[tlaunch], tlaunch < t ≤ tf};
```

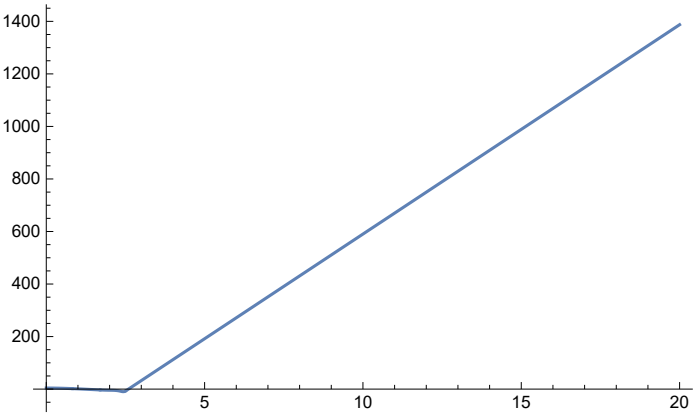
Out[303]=
```
2.50962
```

In[306]:=

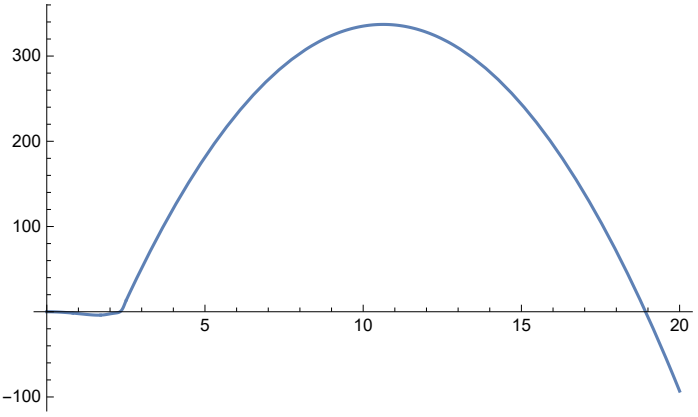
In[307]:=
```
1.65`
Plot[xbtotal[t], {t, 0, tf}]
Plot[ybtotal[t], {t, 0, tf}]
```

Out[307]= 1.65

Out[308]=



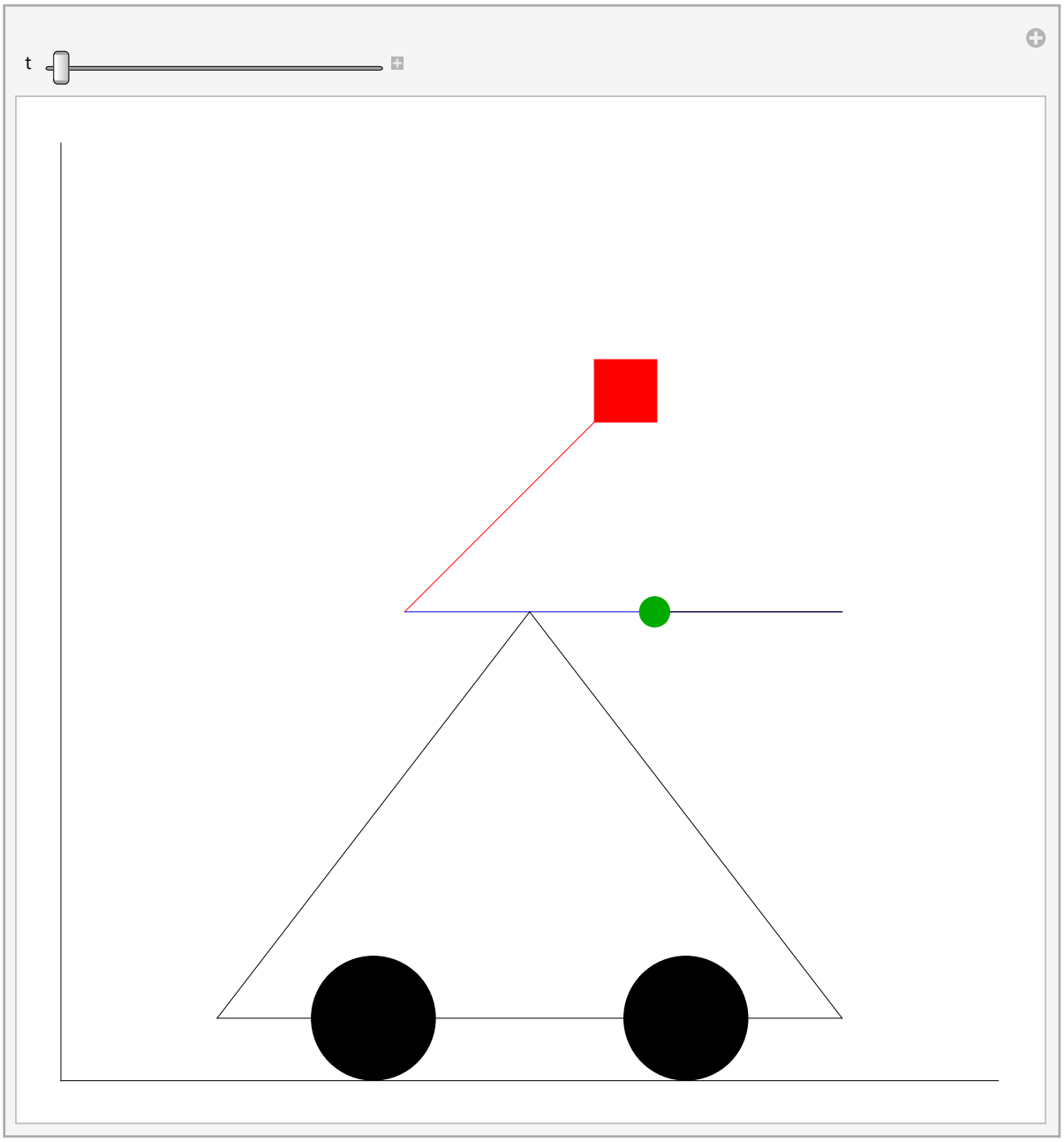
Out[309]=



In[310]:=
```
Clear[slidingtreb]
```

In[311]:=
```
Manipulate[Graphics[{
   Line[{{-15, -15}, {-15, 15}}],
   Line[{{-15, -15}, {15, -15}}],
   Black, Disk[{xppw[t] - 5, -13}, 2],
   Black, Disk[{xppw[t] + 5, -13}, 2],
   Blue, Line[{{xtapp[t], ytapp[t]}, {xtacp[t], ytacp[t]}}],
   Black, Line[{{xtapp[t], ytapp[t]}, {xbp[t], ybp[t]}}],
   Red, Line[{{xtacp[t], ytacp[t]}, {xcwp[t], ycwp[t]}}],
   Black, Line[{{xppw[t] - 10, -13}, {xppw[t], 0}}],
   Black, Line[{{xppw[t], 0}, {xppw[t] + 10, -13}}],
   Black, Line[{{xppw[t] - 10, -13}, {xppw[t] + 10, -13}}],
   Red, Rectangle[{xcwp[t] - 1, ycwp[t] - 1}, {xcwp[t] + 1, ycwp[t] + 1}],
   Darker[Green], Disk[{xbtotal[t], ybtotal[t]}, .5]

  }, ImageSize → Large],
 {t, 0.0000, tf}]
```

Out[311]=



In[312]:=
```
slidingtreb = Table[Graphics[{
      Line[{{-15, -15}, {-15, 15}}],
      Line[{{-15, -15}, {15, -15}}],
      Black, Disk[{xppw[t] - 5, -13}, 2],
      Black, Disk[{xppw[t] + 5, -13}, 2],
      Blue, Line[{{xtapp[t], ytapp[t]}, {xtacp[t], ytacp[t]}}],
      Black, Line[{{xtapp[t], ytapp[t]}, {xbp[t], ybp[t]}}],
      Red, Line[{{xtacp[t], ytacp[t]}, {xcwp[t], ycwp[t]}}],
      Black, Line[{{xppw[t] - 10, -13}, {xppw[t], 0}}],
      Black, Line[{{xppw[t], 0}, {xppw[t] + 10, -13}}],
      Black, Line[{{xppw[t] - 10, -13}, {xppw[t] + 10, -13}}],
      Red, Rectangle[{xcwp[t] - 1, ycwp[t] - 1}, {xcwp[t] + 1, ycwp[t] + 1}],
      Darker[Green], Disk[{xbtotal[t], ybtotal[t]}, .5]

     }, ImageSize → Large],
    {t, 0.0000, tlaunch + .2, .03}];
```

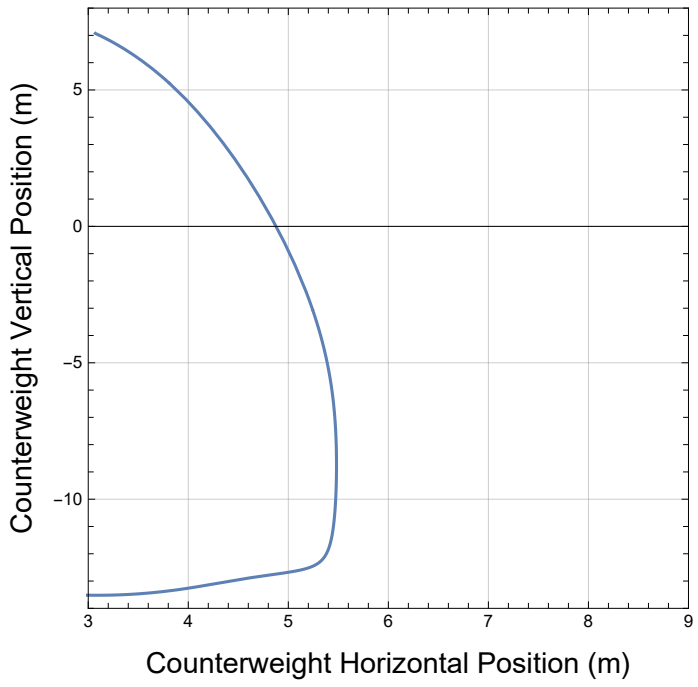
In[313]:=
```
SetDirectory[NotebookDirectory[]]
```

Out[313]=
```
C:\Users\jerem\OneDrive\Desktop\Carleton College\Winter 2021\PHYS 231\Trebuchet Project
```

In[314]:=
```
(*Export["slidingtreb.gif",slidingtreb,"AnimationRepetitions"->Infinity]*)
```

In[315]:=
```
rollingcwswing = ParametricPlot[{xcwp[t], ycwp[t]}, {t, 0, tlaunch + 1},
  AspectRatio → 1, Frame → True, PlotRange → {{3, 9}, {-14, 8}},
  FrameLabel → {Style["Counterweight Horizontal Position (m)", 15],
    Style["Counterweight Vertical Position (m)", 15]}, GridLines → Automatic]
```

Out[315]=



In[316]:=
```
Export["rollingcwswing.png", rollingcwswing]
```

Out[316]=
```
rollingcwswing.png
```

In[317]:=
```
Clear[Tpw, Upw, Epw]
```

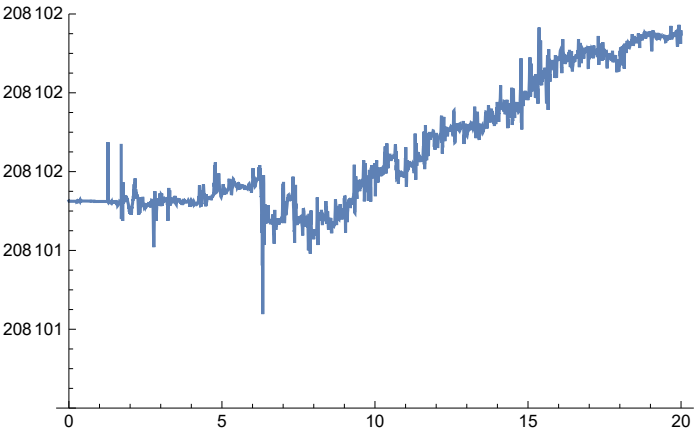
In[318]:=
```
xap[t_] := xppw[t] + (l1 - l2)/2 Cos[θpw[t]]

yap[t_] := -(l1 - l2)/2 Sin[θpw[t]]
```

In[320]:=
```
Tpw[t_] = 1/2 mb (D[xbp[t], t]^2 + D[ybp[t], t]^2) + 1/2 mcw (D[xcwp[t], t]^2 + D[ycwp[t], t]^2) +
    1/2 ma (D[xap[t], t]^2 + D[yap[t], t]^2) + 1/2 IIa D[θpw[t], t]^2;
Upw[t_] = mb g ybp[t] + mcw g ycwp[t] + ma g yap[t];
```

In[322]:=
```
Epw[t_] = Tpw[t] + Upw[t];
```

In[323]:=
```
Plot[Epw[t], {t, 0, tf}, PlotRange → {208101, 208102}]
```

Out[323]=



In[324]:=

In[325]:=
```
Etoti2 = Epw[0]
```

Out[325]= 208102.

In[326]:=
```
Tbf2 = 1/2 mb (xbp'[tlaunch]^2 + ybp'[tlaunch]^2)
```

Out[326]= 381163.

In[327]:=
```
kineticratio = Tbf2/Etoti2;
```

In[328]:=
```
Print["Ratio: ", kineticratio]
```

```
Ratio: 1.83162
```

In[329]:=
```
Clear[Tbpw, Tratio]
```

In[330]:=
```
Tbpw[t_] = 1/2 mb (xbp'[t]^2 + ybp'[t]^2);
```

In[331]:=
```
Tratio[t_] = Tbpw[t]/Tpw[t];
```

In[332]:=
$$\text{Tratiotot}[t\_] = \frac{\text{Tbpw}[t]}{\text{Epw}[t]};$$

In[333]:=
```
Tratio[tlaunch]
```

Out[333]=
```
0.69195
```

In[334]:=
$$\phi\text{launch}[t\_] = \text{ArcTan}\left[\frac{\text{ybp}'[t]}{\text{xbp}'[t]}\right];$$

In[335]:=
```
ϕlaunch[tlaunch + .2]
```

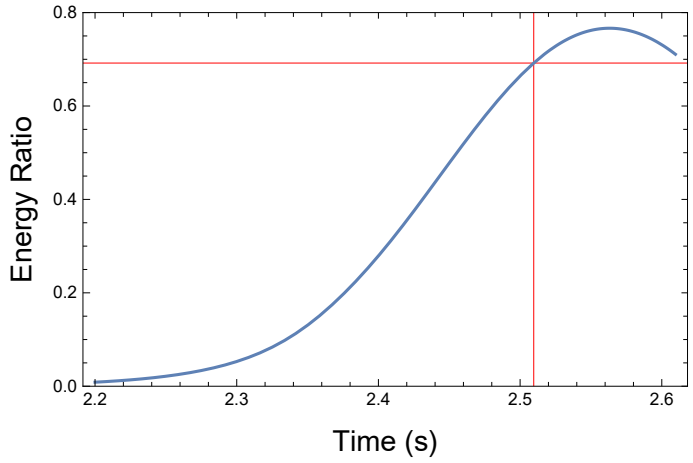
Out[335]=
```
1.3766
```

In[336]:=
```
N[5 Pi / 4]
```

Out[336]=
```
3.92699
```

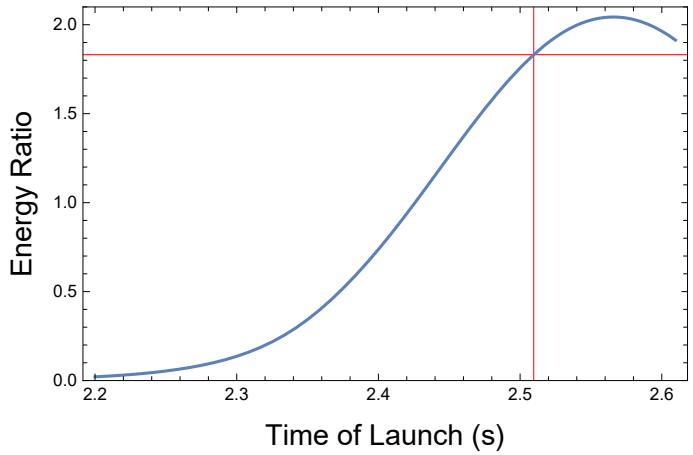
In[337]:=
```
ratioplot = Plot[Tratio[t], {t, 2.2, tlaunch + .1}, PlotRange → {0, 0.8}, Frame → True,
  GridLines → {{tlaunch}, {Tratio[tlaunch]}}, GridLinesStyle → Directive[Red],
  FrameLabel → {Style["Time (s)", 15], Style["Energy Ratio", 15]}]
```

Out[337]=



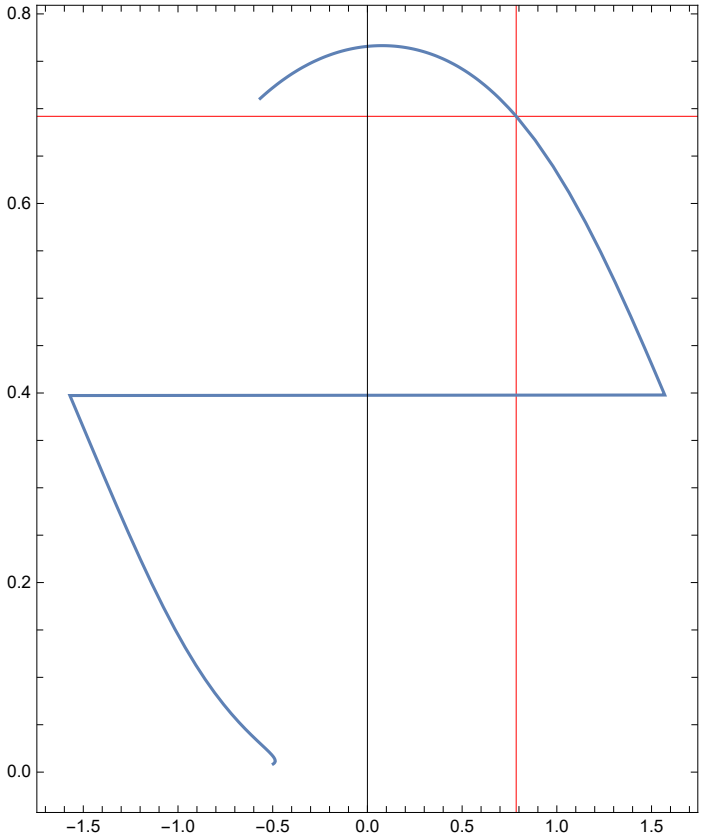
In[338]:=
```
ratioplotpap = Plot[Tratiotot[t], {t, 2.2, tlaunch + .1}, PlotRange → {0, 2.1}, Frame → True,
  GridLines → {{tlaunch}, {Tratiotot[tlaunch]}}, GridLinesStyle → Directive[Red],
  FrameLabel → {Style["Time of Launch (s)", 15], Style[ "Energy Ratio", 15]}]
```

Out[338]=



In[339]:=
```
ParametricPlot[{ϕlaunch[t], Tratio[t]}, {t, 2.2, tlaunch + .1},
 PlotRange → Automatic, AspectRatio → Full, Frame → True,
 GridLines → {{ϕlaunch[tlaunch]}, {Tratio[tlaunch]}}, GridLinesStyle → Directive[Red]]
```

Out[339]=



In[340]:=
```
Export["rollingratiototpap.png", ratioplotpap]
```

Out[340]= rollingratiototpap.png

NDSolve Module

In[231]:=

```
solvestep[L_, t0_, θ0_, ϕ0_, α0_, dθ0_, dϕ0_, dα0_, tguess_, tf_, constraints_] := (
  (*Add the constraints*)
  L1 = L;
  For[i = 1, i ≤ Length[constraints], i++,
   L1 = L1 + constraints[[i]][[1]] × constraints[[i]][[2]];];
  sol = {};
  (*NDSolve based on how many constraints*)
  Which[Length[constraints] == 2,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0,
      D[constraints[[2, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]], constraints[[2, 1]]}, {t, t0, tf}],
   Length[constraints] == 1,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0,
      D[constraints[[1, 2]], t, t] == 0},
     {θ[t], ϕ[t], α[t], constraints[[1, 1]]}, {t, t0, tf}],
   Length[constraints] == 0,
   sol = NDSolve[{EL[L1, θ], EL[L1, ϕ], EL[L1, α],
      α[t0] == α0,
      α'[t0] == dα0,
      θ[t0] == θ0,
      θ'[t0] == dθ0,
      ϕ[t0] == ϕ0,
      ϕ'[t0] == dϕ0},
     {θ[t], ϕ[t], α[t]}, {t, t0, tf}]];
  (*Pull off solutions*)
  solnlist = {};
  For[i = 1, i ≤ Length[sol[[1]]], i++,
   AppendTo[solnlist, sol[[1, i, 2]]];
  ];
  (*Obtain first zeros*)
  zeros = {};
  For[i = 4, i ≤ Length[solnlist], i++,
   AppendTo[zeros,
```

```
    FindRoot[solnlist[[i]], {t, tguess}]];
 ];
 (*Return solutions and zeros*)
 Return[{solnlist, zeros}];
)
```

Kinetic energy

In[232]:=
```
Clear[T, U, mb, mcw, ma, IIa, l1, l2, h, r, g, θ, ϕ, α, L,
 t, tf, EL, soln, xb, yb, xcw, ycw, xcma, ycma, λ, γ, G1, G2, x, y]
```

In[233]:=
$$T := \frac{1}{2}\,mb\,\left(D[xb, t]^2 + D[yb, t]^2\right) + \frac{1}{2}\,mcw\,\left(D[xcw, t]^2 + D[ycw, t]^2\right) + \frac{1}{2}\,IIa\,D[\theta[t], t]^2$$

Note, if you call moment of inertia I you generate errors because this is the imaginary number

In[234]:=
$$IIa := \frac{1}{3}\,ma\,\left(\frac{l1^3 + l2^3}{l1 + l2}\right)$$

Potential energy

In[235]:=
```
U := mb g yb + mcw g ycw + ma g ycma
```

Define transformation to generalized coordinates

In[236]:=
```
xb := l1 Cos[θ[t]] + r Cos[ϕ[t]] (*these are barrel coord*)
yb := -l1 Sin[θ[t]] - r Sin[ϕ[t]]
xcw := -l2 Cos[θ[t]] + h Cos[α[t]] (*these are counter-weight coord*)
ycw := l2 Sin[θ[t]] - h Sin[α[t]]
ycma := -(l1 - l2)/2 Sin[θ[t]] (*these are center of mass coord*)
```

Define Constraint Functions:

In[241]:=
```
G1 := θ[t] - α[t] + α[0] (*counter weight*)
```

In[242]:=
```
G2 := θ[t] - ϕ[t] + ϕ[0] (*barrel*)
```

Let' s see what T & U look like ... It looks nice. Doing this in Mathematica saves the pain of differentiating, squaring etc. Try a few lines of this to get a feel for how nasty it is.

In[243]:=
```
Simplify[T]
```

Out[243]=
$$\frac{1}{6}\left(3\,h^2\,mcw\,\alpha'[t]^2 - 6\,h\,l2\,mcw\,\text{Cos}[\alpha[t] - \theta[t]]\,\alpha'[t]\,\theta'[t] + \left(-l1\,l2\,ma + l1^2\,\left(ma + 3\,mb\right) + l2^2\,\left(ma + 3\,mcw\right)\right)\,\theta'[t]^2 + 6\,l1\,mb\,r\,\text{Cos}[\theta[t] - \phi[t]]\,\theta'[t]\,\phi'[t] + 3\,mb\,r^2\,\phi'[t]^2\right)$$

In[244]:=
```
Simplify[U]
```

Out[244]=
$$-\frac{1}{2} g \left(2\, h\, mcw\, \text{Sin}[\alpha[t]] + \left(l1 \left(ma + 2\, mb\right) - l2 \left(ma + 2\, mcw\right)\right) \text{Sin}[\theta[t]] + 2\, mb\, r\, \text{Sin}[\phi[t]]\right)$$

Define L

In[245]:=
```
L := T - U
```

In[246]:=
```
Simplify[L]
```

Out[246]=
$$\frac{1}{6} \left(3\, h^2\, mcw\, \alpha'[t]^2 - 6\, h\, l2\, mcw\, \text{Cos}[\alpha[t] - \theta[t]]\, \alpha'[t]\, \theta'[t] + \left(-l1\, l2\, ma + l1^2 \left(ma + 3\, mb\right) + l2^2 \left(ma + 3\, mcw\right)\right) \theta'[t]^2 + 6\, l1\, mb\, r\, \text{Cos}[\theta[t] - \phi[t]]\, \theta'[t]\, \phi'[t] + 3 \left(g \left(2\, h\, mcw\, \text{Sin}[\alpha[t]] + \left(l1 \left(ma + 2\, mb\right) - l2 \left(ma + 2\, mcw\right)\right) \text{Sin}[\theta[t]] + 2\, mb\, r\, \text{Sin}[\phi[t]]\right) + mb\, r^2\, \phi'[t]^2\right)\right)$$

In[247]:=

In[248]:=
```
g := 9.81
mb := 60
mcw := 3000
ma := 1000
l1 := 10
l2 := 4
h := 10
r := 6
```

In[256]:=
```
EL[L_, x_] := D[L, x[t]] - D[L, x'[t], t] == 0;
```

In[257]:=
```
Clear[L1, solnlist, zeros, sol, constraints1, tf];
```

In[258]:=
```
tf = 20;
```

In[259]:=
```
constraints1 = {{λ[t], G1}, {γ[t], G2}};
```

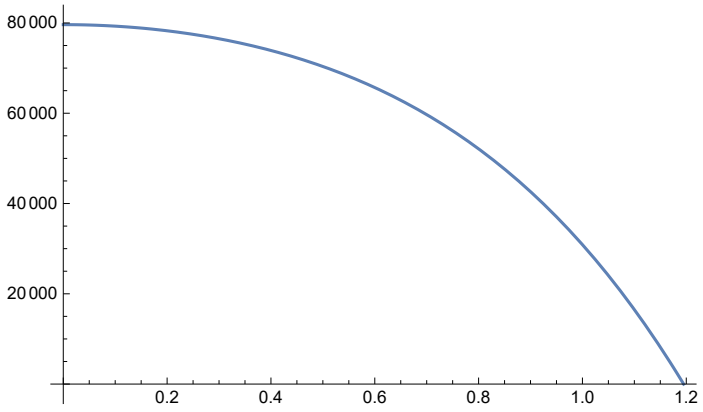
In[260]:=
```
step1 = solvestep[L, 0, 0, Pi, 7 Pi / 4, 0, 0, 0, 1, tf, constraints1]
```

... NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

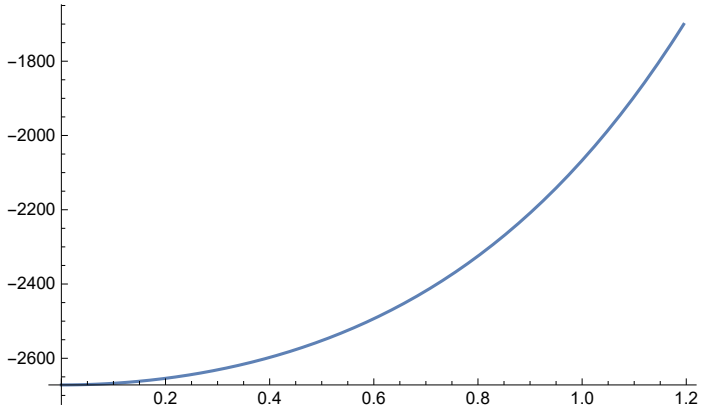
Out[260]=
```
{{InterpolatingFunction[ <Domain: {{0., 20.}} Output: scalar> ][t],
  InterpolatingFunction[ <Domain: {{0., 20.}} Output: scalar> ][t],
  InterpolatingFunction[ <Domain: {{0., 20.}} Output: scalar> ][t],
  InterpolatingFunction[ <Domain: {{0., 20.}} Output: scalar> ][t],
  InterpolatingFunction[ <Domain: {{0., 20.}} Output: scalar> ][t]}, {{t → 1.19494}, {t → 1.70667}}}
```

In[261]:=
```
Plot[step1[[1, 4]], {t, 0, t2}]
Plot[step1[[1, 5]], {t, 0, t2}]
```

Out[261]=



Out[262]=



In[263]:=
```
Print[step1[[2]]]
```

```
{{t → 1.19494}, {t → 1.70667}}
```

In[264]:=
```
Clear[L2, solnlist, zeros, constraints2, sol, t2, step2];
```

In[265]:=
```
constraints2 = {{ϒ[t], G2}};
```

In[266]:=
```
t2 = step1[[2, 1, 1, 2]];
Print[t2];
```

```
1.19494
```

In[268]:=
```
step2 = solvestep[L, t2, step1[[1, 1, 0]][t2], step1[[1, 2, 0]][t2], step1[[1, 3, 0]][t2],
  step1[[1, 1, 0]]'[t2], step1[[1, 2, 0]]'[t2], step1[[1, 3, 0]]'[t2], 2, tf, constraints2]
```

NDSolve: Some of the functions have zero differential order, so the equations will be solved as a system of differential–algebraic equations.

InterpolatingFunction: Input value {1.11127} lies outside the range of data in the interpolating function. Extrapolation will be used.

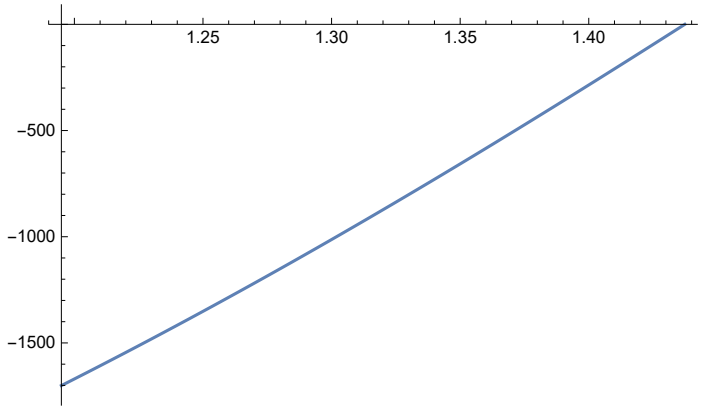
InterpolatingFunction: Input value {1.17017} lies outside the range of data in the interpolating function. Extrapolation will be used.

Out[268]=
```
{{InterpolatingFunction[ ⊞ Domain: {{1.19, 20.}}
                         Output: scalar ][t],
```
Data not in notebook; Store now »
```
  InterpolatingFunction[ ⊞ Domain: {{1.19, 20.}}
                         Output: scalar ][t],
```
Data not in notebook; Store now »
```
  InterpolatingFunction[ ⊞ Domain: {{1.19, 20.}}
                         Output: scalar ][t],
```
Data not in notebook; Store now »
```
  InterpolatingFunction[ ⊞ Domain: {{1.19, 20.}}
                         Output: scalar ][t]}, {{t → 1.43735}}}
```
Data not in notebook; Store now »

In[269]:=

In[270]:=
```
Plot[step2[[1, 4]], {t, t2, t3}]
```

Out[270]=



In[271]:=
```
Print[step2[[2]]]
```

```
{{t → 1.43735}}
```

In[272]:=
```
Clear[L1, solnlist, zeros, constraints3, sol, t3];
```

In[273]:=
```
constraints3 = {};
```

In[274]:=
```
t3 = step2[[2, 1, 1, 2]];
Print[t3]
```

```
1.43735
```

In[276]:=
```
step3 = solvestep[L, t3, step2[[1, 1, 0]][t3], step2[[1, 2, 0]][t3], step2[[1, 3, 0]][t3],
  step2[[1, 1, 0]]'[t3], step2[[1, 2, 0]]'[t3], step2[[1, 3, 0]]'[t3], 2, tf, constraints3]
```

Out[276]=
```
{{InterpolatingFunction[ ⊞ Domain: {{1.44, 20.}}
                         Output: scalar ][t],
  InterpolatingFunction[ ⊞ Domain: {{1.44, 20.}}
                         Output: scalar ][t],
  InterpolatingFunction[ ⊞ Domain: {{1.44, 20.}}
                         Output: scalar ][t]}, {}}
```

In[277]:=

In[278]:=

In[279]:=

In[280]:=
```
θpw[t_] = { step1[[1, 1]]  0 ≤ t ≤ t2
          { step2[[1, 1]]  t2 ≤ t ≤ t3;
          { step3[[1, 1]]  t3 ≤ t ≤ tf
```

In[281]:=
```
ϕpw[t_] = { step1[[1, 2]]  0 ≤ t ≤ t2
          { step2[[1, 2]]  t2 ≤ t ≤ t3;
          { step3[[1, 2]]  t3 ≤ t ≤ tf
```

In[282]:=
```
αpw[t_] = { step1[[1, 3]]  0 ≤ t ≤ t2
          { step2[[1, 3]]  t2 ≤ t ≤ t3;
          { step3[[1, 3]]  t3 ≤ t ≤ tf
```

In[283]:=
```
λpw[t_] = { step1[[1, 4]]  0 ≤ t ≤ t2
          {       0        t2 ≤ t ≤ t3;
          {       0        t3 ≤ t ≤ tf
```

In[284]:=
```
γpw[t_] = { step1[[1, 5]]  0 ≤ t ≤ t2
          { step2[[1, 4]]  t2 ≤ t ≤ t3;
          {       0        t3 ≤ t ≤ tf
```

In[285]:=
```
θpw[0]
```

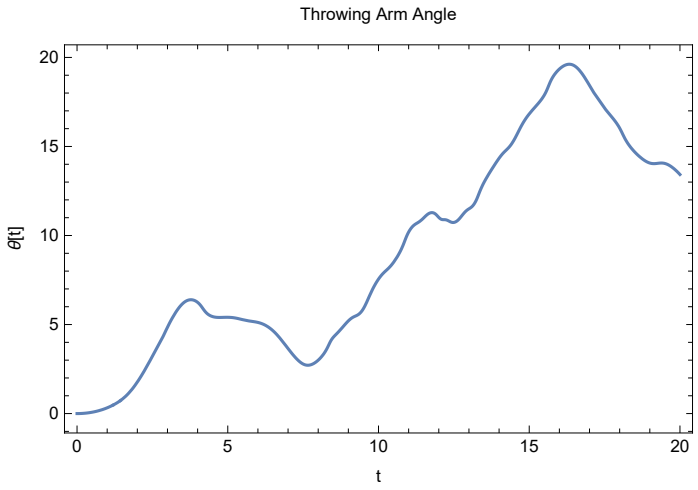
Out[285]= -2.77399×10^{-14}

In[286]:=
```
θpw[tf]
```

Out[286]= 13.4246

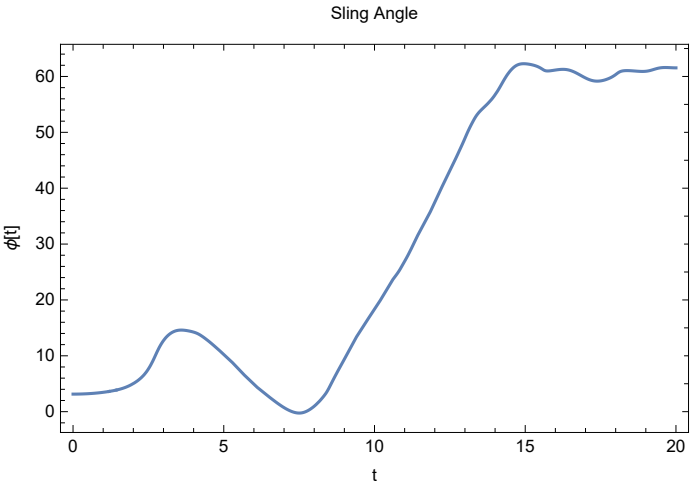
In[287]:=
```
Plot[θpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "θ[t]", "Throwing Arm Angle"}]
(*Plot[θpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","θ'[t]","Throwing Arm Angular Velocity"}]*)
```

Out[287]=



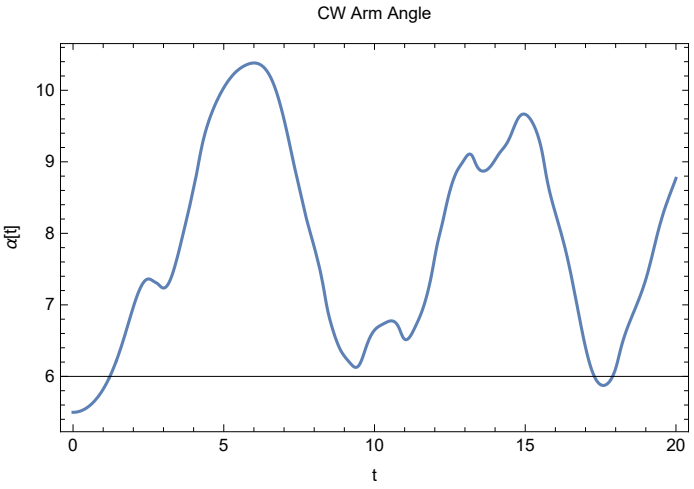
In[288]:=
```
Plot[ϕpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "ϕ[t]", "Sling Angle"}]
(*Plot[ϕpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","ϕ'[t]","Sling Angular Velocity"}]*)
```

Out[288]=



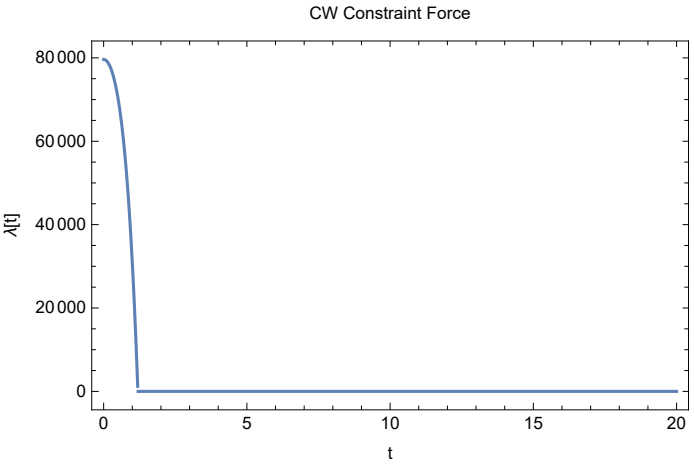
In[289]:=
```
Plot[αpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "α[t]", "CW Arm Angle"}]
(*Plot[αpw'[t],{t,0,tf},PlotRange→All,Frame→True,
 FrameLabel→{"t","α'[t]","CW Arm Angular Velocity"}]*)
```

Out[289]=



In[290]:=
```
Plot[λpw[t], {t, 0, tf}, PlotRange → All,
 Frame → True, FrameLabel → {"t", "λ[t]", "CW Constraint Force"}]
```

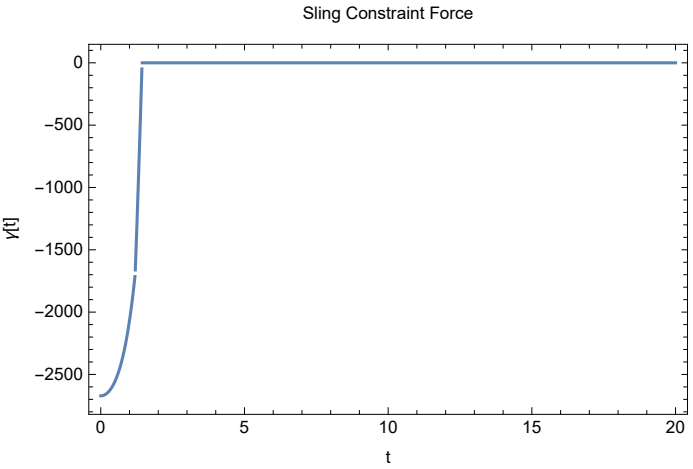
Out[290]=



In[291]:=

In[292]:=
```
Plot[γpw[t], {t, 0, tf}, PlotRange → All, Frame → True,
 FrameLabel → {"t", "γ[t]", "Sling Constraint Force"}]
```

Out[292]=



In[293]:=

In[294]:=

In[295]:=
```
Clear[tlaunch]
```

In[296]:=
```
xtapp[t_] = l1 Cos[θpw[t]];
ytapp[t_] = -l1 Sin[θpw[t]];
xtacp[t_] = -l2 Cos[θpw[t]];
ytacp[t_] = l2 Sin[θpw[t]];
```

In[300]:=
```
xbp[t_] = l1 Cos[θpw[t]] + r Cos[ϕpw[t]];(*these are barrel coord*)
ybp[t_] = -l1 Sin[θpw[t]] - r Sin[ϕpw[t]];
xcwp[t_] = -l2 Cos[θpw[t]] + h Cos[αpw[t]];(*these are counter-weight coord*)
ycwp[t_] = l2 Sin[θpw[t]] - h Sin[αpw[t]];
tlaunch = FindRoot[xbp'[t] == ybp'[t], {t, 2.74}][[1, 2]]
xbtotal[t_] = { xbp[t]                                   0 ≤ t ≤ tlaunch
              { xbp'[tlaunch] (t - tlaunch) + xbp[tlaunch]   tlaunch < t ≤ tf ;
ybtotal[t_] = { ybp[t]                                                  0 ≤ t ≤ tlaunch
              { ybp'[tlaunch] (t - tlaunch) - g/2 (t - tlaunch)^2 + ybp[tlaunch]   tlaunch < t ≤ tf ;
```

Out[304]= 2.74282

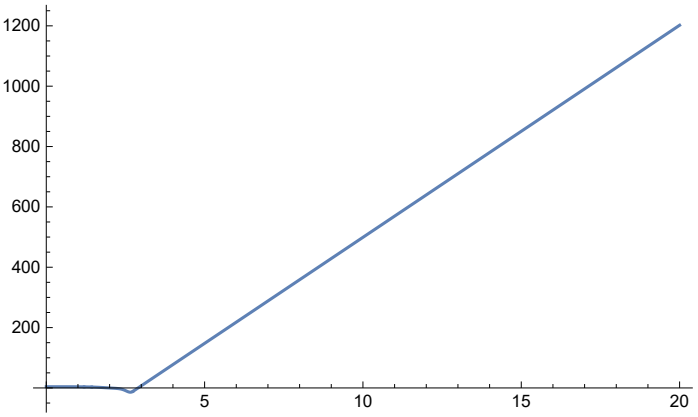
In[307]:=

In[308]:=

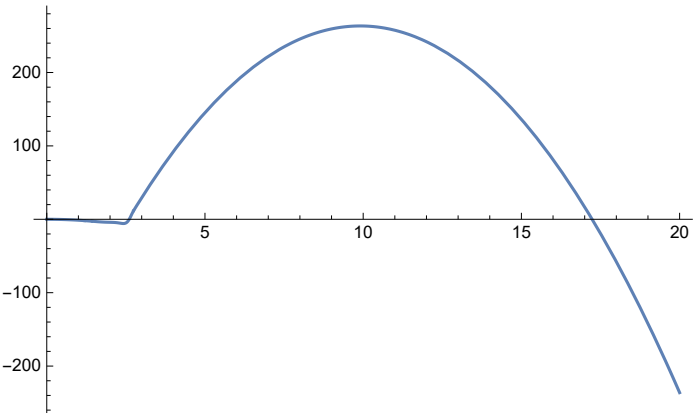
In[309]:=
```
1.65`
Plot[xbtotal[t], {t, 0, tf}]
Plot[ybtotal[t], {t, 0, tf}]
```

Out[309]= 1.65

Out[310]=

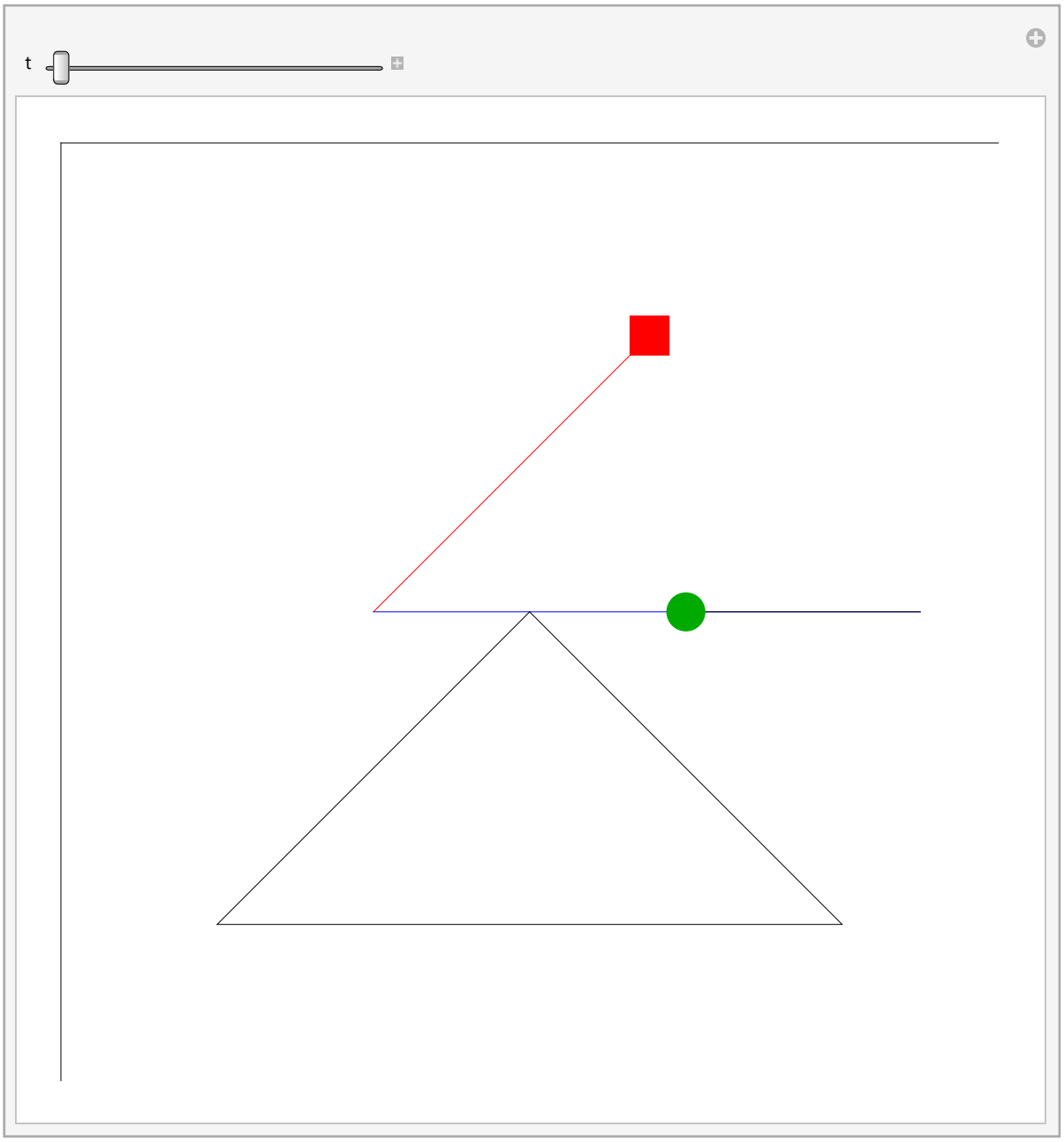


Out[311]=



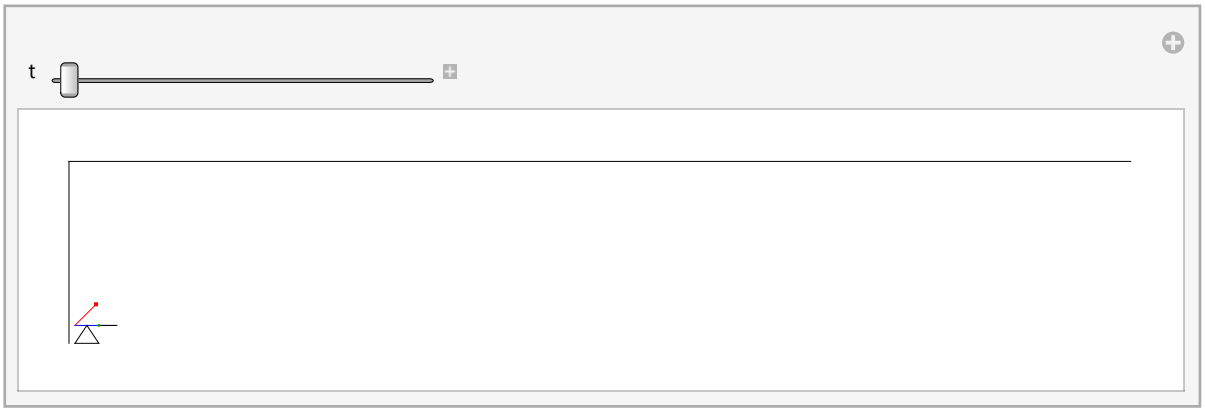
In[312]:=
```
Manipulate[Graphics[{
   Line[{{-12, -12}, {-12, 12}}],
   Line[{{-12, 12}, {12, 12}}],
   Blue, Line[{{xtapp[t], ytapp[t]}, {xtacp[t], ytacp[t]}}],
   Black, Line[{{xtapp[t], ytapp[t]}, {xbp[t], ybp[t]}}],
   Red, Line[{{xtacp[t], ytacp[t]}, {xcwp[t], ycwp[t]}}],
   Black, Line[{{-8, -8}, {0, 0}}],
   Black, Line[{{0, 0}, {8, -8}}],
   Black, Line[{{-8, -8}, {8, -8}}],
   Red, Rectangle[{xcwp[t] - .5, ycwp[t] - .5}, {xcwp[t] + .5, ycwp[t] + .5}],
   Darker[Green], Disk[{xbtotal[t], ybtotal[t]}, .5]

  }, ImageSize → Large],
 {t, 0.0000, tf}]
```

Out[312]=



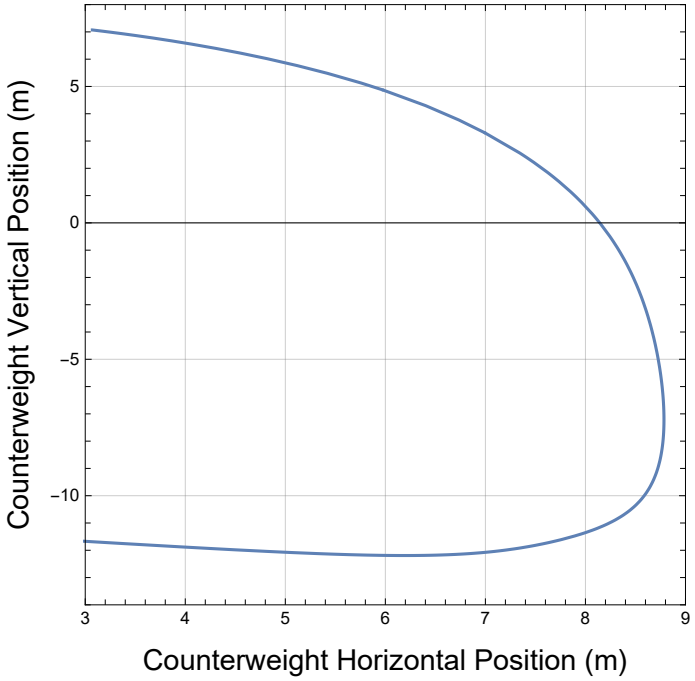
In[313]:=
```
Manipulate[Graphics[{
   Line[{{-6, -6}, {-6, 55}}],
   Line[{{-6, 55}, {350, 55}}],
   Blue, Line[{{xtapp[t], ytapp[t]}, {xtacp[t], ytacp[t]}}],
   Black, Line[{{xtapp[t], ytapp[t]}, {xbp[t], ybp[t]}}],
   Red, Line[{{xtacp[t], ytacp[t]}, {xcwp[t], ycwp[t]}}],
   Black, Line[{{-4, -6}, {0, 0}}],
   Black, Line[{{0, 0}, {4, -6}}],
   Black, Line[{{-4, -6}, {4, -6}}],
   Red, Rectangle[{xcwp[t] - .5, ycwp[t] - .5}, {xcwp[t] + .5, ycwp[t] + .5}],
   Darker[Green], Disk[{xbtotal[t], ybtotal[t]}, .5],

  }, ImageSize → Large],
 {t, 0.0000, 8}]
```

Out[313]=



In[314]:=
```
fixedcwswing = ParametricPlot[{xcwp[t], ycwp[t]}, {t, 0, tlaunch + 1},
  AspectRatio → 1, Frame → True, PlotRange → {{3, 9}, {-14, 8}},
  FrameLabel → {Style["Counterweight Horizontal Position (m)", 15],
    Style["Counterweight Vertical Position (m)", 15]}, GridLines → Automatic]
```

Out[314]=



In[315]:=
```
Export["fixedcwswing.png", fixedcwswing]
```

Out[315]= fixedcwswing.png

In[316]:=
```
Clear[Tpw, Upw, Epw]
```

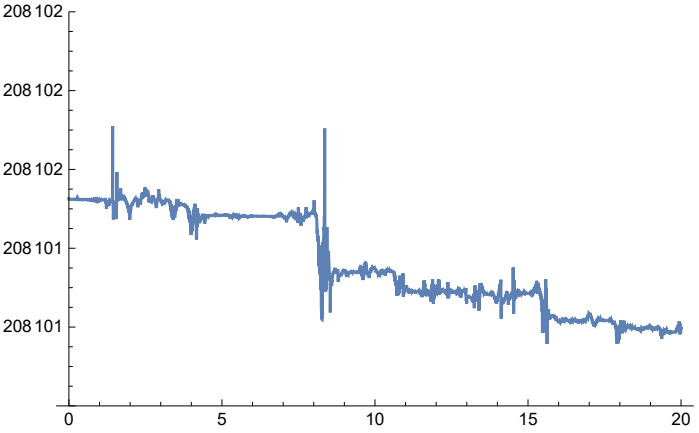
In[317]:=
```
ycmap[t_] := -(l1 - l2)/2 Sin[θpw[t]];
```

In[318]:=
```
Tpw[t_] = 1/2 mb (D[xbp[t], t]^2 + D[ybp[t], t]^2) +
    1/2 mcw (D[xcwp[t], t]^2 + D[ycwp[t], t]^2) + 1/2 IIa D[θpw[t], t]^2;
Upw[t_] = mb g ybp[t] + mcw g ycwp[t] + ma g ycmap[t];
```

In[320]:=
```
Epw[t_] = Tpw[t] + Upw[t];
```

In[321]:=
```
Plot[Epw[t], {t, 0, tf}, PlotRange → {208 101, 208 102}]
```

Out[321]=



In[322]:=
```
FindMaximum[Epw[t], {t, 7}]
```

Out[322]=
```
{208 102., {t → 0.701919}}
```

In[323]:=
```
FindMinimum[Epw[t], t]
```

Out[323]=
```
{208 102., {t → 1.00016}}
```

In[324]:=
```
EError = MaxValue[Epw[t], t] - MinValue[Epw[t], t]
```

Out[324]=
```
208 102.
```

In[325]:=
```
N[17 Pi / 4]
```

Out[325]=
```
13.3518
```

In[326]:=
```
Etoti = Epw[0]
```

Out[326]=
```
208 102.
```

In[327]:=
```
Tbf = 1/2 mb (xbp'[tlaunch]^2 + ybp'[tlaunch]^2)
```

Out[327]=
```
296 414.
```

In[328]:=
```
kineticratio = Tbf/Etoti;
```

In[329]:=
```
Print["Ratio: ", kineticratio]
```

```
Ratio: 1.42437
```

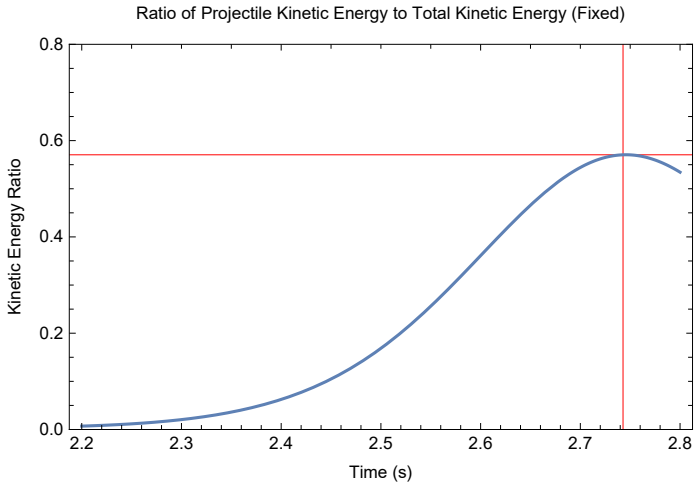
In[330]:=
```
Tbpw[t_] = 1/2 mb (xbp'[t]^2 + ybp'[t]^2);
```

In[331]:=
```
Tratio[t_] = Tbpw[t]/Tpw[t];
```

In[332]:=
```
Tratiotot[t_] = Tbpw[t]/Epw[t];
```

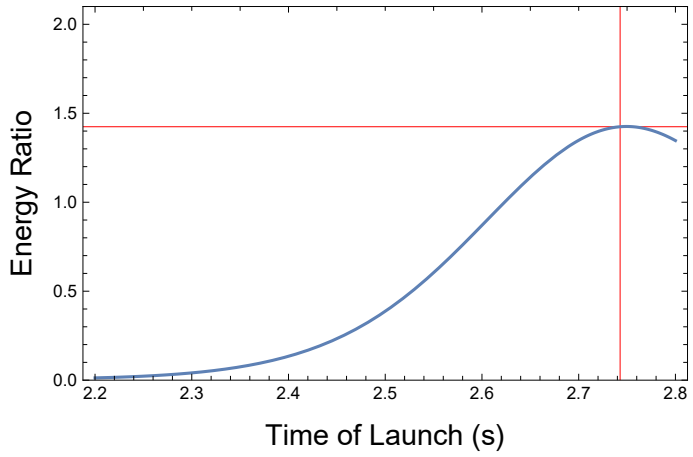
In[333]:=
```
ratioplot = Plot[Tratio[t], {t, 2.2, 2.8}, PlotRange → {0, 0.8},
  Frame → True, GridLines → {{tlaunch}, {Tratio[tlaunch]}},
  GridLinesStyle → Directive[Red], FrameLabel → {"Time (s)", "Kinetic Energy Ratio",
    "Ratio of Projectile Kinetic Energy to Total Kinetic Energy (Fixed)" }]
```

Out[333]=



In[334]:=
```
ratioplottotpaper = Plot[Tratiotot[t], {t, 2.2, 2.8}, PlotRange → {0, 2.1}, Frame → True,
  GridLines → {{tlaunch}, {Tratiotot[tlaunch]}}, GridLinesStyle → Directive[Red],
  FrameLabel → {Style["Time of Launch (s)", 15], Style["Energy Ratio", 15] }]
```

Out[334]=



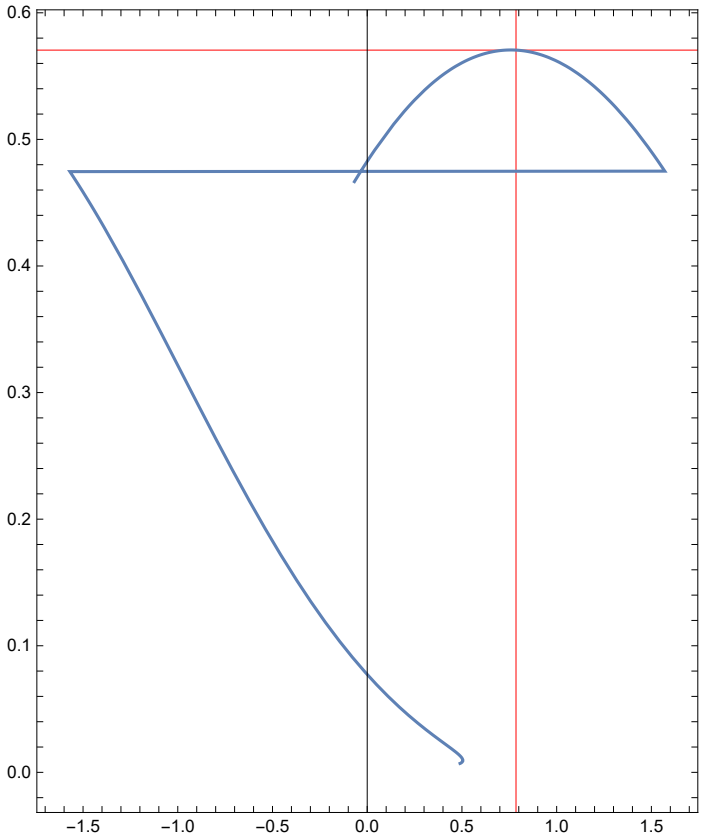
In[335]:=
```
ϕlaunch[t_] = ArcTan[ybp'[t] / xbp'[t]];
```

In[336]:=
```
ϕlaunch[tlaunch]
```

Out[336]=
```
0.785398
```

In[337]:=
```
ParametricPlot[{ϕlaunch[t], Tratio[t]}, {t, 2.2, tlaunch + .1},
 PlotRange → Automatic, AspectRatio → Full, Frame → True,
 GridLines → {{ϕlaunch[tlaunch]}, {Tratio[tlaunch]}}, GridLinesStyle → Directive[Red]]
```

Out[337]=



In[338]:=
```
SetDirectory[NotebookDirectory[]];
```

In[339]:=
```
Export["fixedratiototpap.png", ratioplottotpaper]
```

Out[339]= fixedratiototpap.png