

```

(*impact of a pendulum with a flat rigid surface*)
(* packages that have some differential
equations to solve and define some utility functions *)
Needs["DifferentialEquations`NDSolveProblems`"];
Needs["DifferentialEquations`NDSolveUtilities`"];
Needs["DifferentialEquations`InterpolatingFunctionAnatomy`"];
Needs["GUIKit`"];

ClearAll["Global`*"];
Off[General::spell];
Off[General::spell1];

(*geometry of the pendulum*)
L = 1; (*length of the pendulum [m]*)
g = 9.81; (*gravitational acceleration [m/s^2]*)
ro = 7800; (*density [kg/m^3]*)
R = 0.01; (*radius of the hemisphere end [m]*)
m = Pi R^2 L ro; (*mass [kg]*)
(*angular velocity*)
alpha = {0, 0, theta'[t]};
(*position of CM: C*)
xC = (L/2) * Cos[theta[t]];
yC = (L/2) * Sin[theta[t]];
rC = {xC, yC, 0};
(*position of the tip A*)
xA = L * Cos[theta[t]];
yA = L * Sin[theta[t]];
rA = {xA, yA, 0};
(*velocity of the tip A*)
vA = D[rA, t];
(*gravitational force at C*)
G = {0, mg, 0};
(*impact angle*)
thetai = Pi/6;
(*mass moment of inertia*)
IC = m * L^2 / 12;
IO = IC + m * (L/2)^2;
(*equation of motion for free fall*)
eqI = Simplify[IO * alpha - Cross[rC, G]][[3]];

(*-----*)
sol0 = NDSolve[{eqI == 0, theta[0] == 0, theta'[0] == 0}, theta,
  {t, 0, Infinity},
  Method -> {EventLocator, "Event" -> (theta[t] - thetai)}];

t0 = InterpolatingFunctionDomain[First[theta /. sol0]][[1, -1]];
theta0 = (Evaluate[theta[t] /. sol0] /. t -> t0)[[1]];
omega0 = Chop[(Evaluate[D[theta[t] /. sol0, t] /. t -> t0][[1]]];
vA0 = vA /. {theta[t] -> theta0, theta'[t] -> omega0};
v0x = vA0[[1]];
v0y = vA0[[2]];

Print[" "]
Print["before impact"]
(*Print["t0 = ", t0, " [s]"]*)
Print["theta0 = ", theta0, " [rad] = ", theta0 * 180 / Pi, " [deg]"]
Print["omega0 = ", omega0, " [rad/s]"]
Print["v0 = ", vA0, " [m/s]"];
Print[" "]

(*thetaplot0=Plot[Evaluate[theta[t] /. sol0] * 180 / Pi,
  {t, 0, t0}, AxesLabel -> {"t [s]", "theta [deg]"}, PlotRange -> Automatic]
omegaplot0=Plot[Evaluate[D[theta[t] /. sol0, t]], {t, 0, t0},
  AxesLabel -> {"t [s]", "omega [rad/s]"}, PlotRange -> Automatic]

```

```
AxisLabel→{"t[s]", "Omega[rad/s]"}, PlotRange→Automatic]*)
```

before impact

```
theta0 = 0.523599 [rad] = 30. [deg]
```

```
omega0 = 3.83601 [rad/s]
```

```
v0 = {-1.91801, 3.32209, 0} [m/s]
```

```
Print["impact with a flat surface"]
```

```
(*elastic compression*)
E1 = 200*10^9; (*elastic modulus [Pa]*)
Sy = 1.12*10^9; (*yield strength [Pa]*)
nu = 0.33; (*Poisson's ratio*)
Ep = ((1 - nu^2) / E1 + (1 - nu^2) / E1)^-1; (*equivalent elastic modulus*)
k1 = 2 / (3 (1 - nu^2)) E1 Sqrt[R]; (*elastic constant Hertz*)
CJ = 1.295 E^(0.736 nu); (*critical yield stress coefficient*)
yc = (Pi CJ Sy / (2 Ep))^2 R; (*critical displacement*)
Print["critical displacement yc = (Pi CJ Sy / (2 Ep))^2 R = ", yc, " [m]"]
(*relative displacement of the tip during contact*)
delta = L Sin[theta[t]] - L Sin[theta0];
(*Hertz elastic force*)
P1 = {0, -k1 delta^1.5, 0};
(*equation of motion for elastic compression 0<=delta<=1.9 yc*)
eqe = Simplify[IO*alpha - Cross[rC, G] - Cross[rA, P1]][[3]];
t0 = 0;
sole = NDSolve[{eqe == 0, theta[t0] == theta0, theta'[t0] == omega0},
  theta, {t, t0, Infinity}, MaxSteps → 10 000,
  Method → {EventLocator, "Event" → (delta - 1.9 yc)}];

te = InterpolatingFunctionDomain[First[theta /. sole]][[1, -1]];
thetae = Chop[(Evaluate[theta[t] /. sole] /. t → te)[[1]]];
omegae = Chop[(Evaluate[D[theta[t] /. sole, t] /. t → te)[[1]]];
deltae = delta /. theta[t] -> thetae;
P1e = (P1 /. theta[t] -> thetae)[[2]];

Print[" "]
Print["end of elastic compression: delta = deltae = 1.9 yc"]
Print["te= ", te, " [s]"]
Print["thetae = ", thetae, " [rad] = ", thetae*180/Pi, " [deg]"]
Print["omegae = ", omegae, " [rad/s]"]
Print["deltae = ", deltae, " [m]"]
Print["P1e = ", P1e, " [N]"]
Print[" "]
```

```
thetaplote = Plot[Evaluate[theta[t] /. sole] * 180 / Pi,
  {t, t0, te}, AxesLabel → {"t[s]", "theta[deg]"}, PlotRange → Automatic]
omegaplote = Plot[Evaluate[D[theta[t] /. sole, t]], {t, t0, te},
  AxesLabel → {"t[s]", "omega[rad/s]"}, PlotRange → Automatic]
deltaplote = Plot[Evaluate[delta /. sole], {t, t0, te},
  AxesLabel → {"t[s]", "delta[m]"}, PlotRange → Automatic]
P1plote = Plot[Evaluate[P1[[2]] /. sole], {t, t0, te},
  AxesLabel → {"t[s]", "P[N]"}, PlotRange → Automatic]
```

impact with a flat surface

critical displacement  $y_c = (\text{Pi CJ Sy} / (2 \text{Ep}))^2 R = 6.69933 \times 10^{-6}$  [m]

end of elastic compression:  $\text{delta} = \text{deltae} = 1.9 y_c$

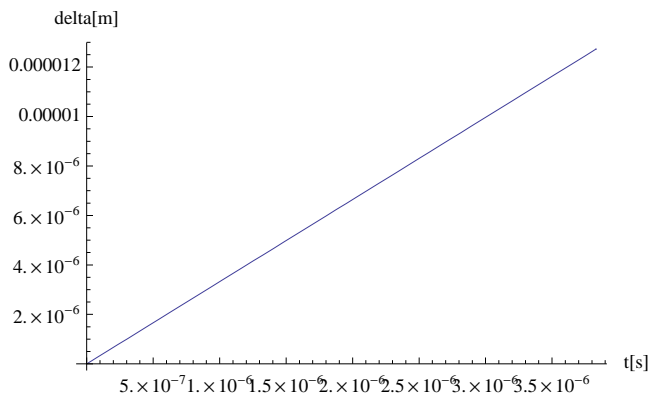
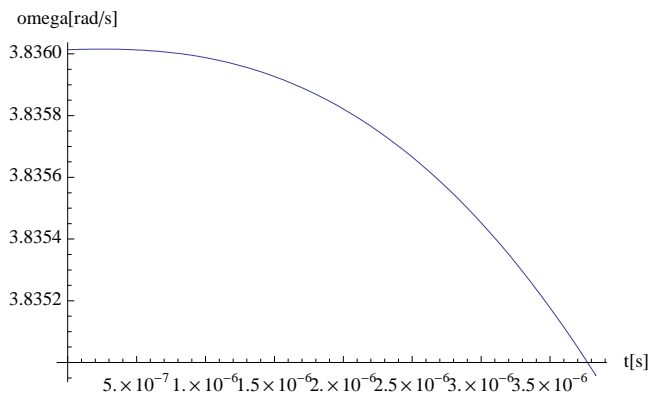
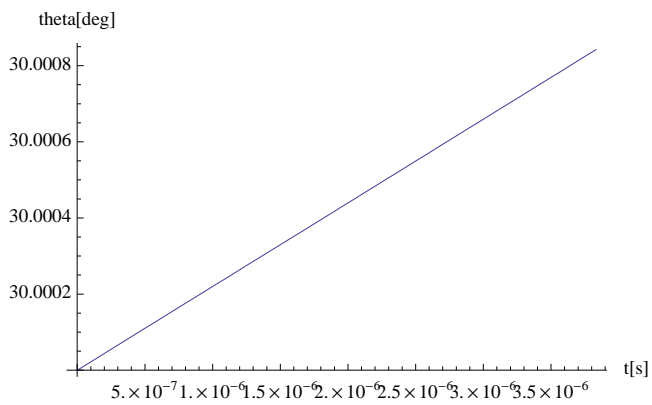
$t_e = 3.83185 \times 10^{-6}$  [s]

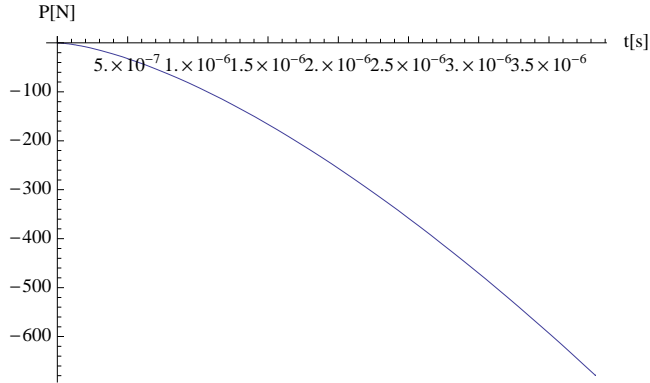
$\text{thetae} = 0.523613$  [rad] = 30.0008 [deg]

$\text{omegae} = 3.83496$  [rad/s]

$\text{deltae} = 0.0000127287$  [m]

$P_{1e} = -679.5$  [N]





```
(* elasto-plastic compression *)
ey = Sy / Ep;
B = 0.14 E ^ (23 ey);
a = Sqrt[R delta (delta / (1.9 yc)) ^ B];
HS = 2.84 - 0.92 (1 - Cos[Pi a / R]);
Pc = 4. / 3 (R / Ep) ^ 2 (Pi CJ Sy / 2) ^ 3;
y = delta / yc;
Pep = Pc (E ^ (-1. / 4 y ^ (5. / 12)) y ^ (3. / 2) + 4 / CJ HS (1 - E ^ (-1. / 25 y ^ (5. / 9))) y);

Print[" "];
Print["P(te-0) = k1 (1.9 delta)^(3./2)/.delta->(1.9 yc) = ", k1 (1.9 yc) ^ (3. / 2), " [N]"];
Print["P(te+0) = Pep/.delta->(1.9 yc) = ", Pep /. delta -> (1.9 yc), " [N]"];
Print[" "];

eqep = Simplify[IO * alpha - Cross[rC, G] - Cross[rA, {0, -Pep, 0}]] [[3]];

solep = NDSolve[{eqep == 0, theta[te] == thetae, theta'[te] == omegae},
  theta, {t, te, Infinity}, MaxSteps -> 10 000,
  Method -> {EventLocator, "Event" -> theta'[t]};

tm = InterpolatingFunctionDomain[First[theta /. solep]] [[1, -1]];
thetam = Chop[(Evaluate[theta[t] /. solep] /. t -> tm) [[1]]];
omegam = Chop[(Evaluate[D[theta[t] /. solep, t]] /. t -> tm) [[1]]];
deltam = delta /. theta[t] -> thetam;
Pm = -Pep /. theta[t] -> thetam;

Print[" "]
Print["end of elasto-plastic compression = maximum compression theta'[t]=omegam=0"]
Print["tm = ", tm, " [s]"]
Print["thetam = ", Chop[thetam], " [rad] = ", Chop[thetam] * 180 / Pi, " [deg]"]
Print["omegam = ", Chop[omegam], " [rad/s]"]
Print["deltam = ", Chop[deltam], " [m]"]
Print["Pm = ", Chop[Pm], " [N]"]
Print[" "]

thetaplotm = Plot[Evaluate[theta[t] /. solep] * 180 / Pi,
  {t, te, tm}, AxesLabel -> {"t[s]", "theta[deg]"}, PlotRange -> Automatic]
omegaplotm = Plot[Evaluate[D[theta[t] /. solep, t]], {t, te, tm},
  AxesLabel -> {"t[s]", "omega[rad/s]"}, PlotRange -> Automatic]
deltaplotm = Plot[Evaluate[delta /. solep], {t, te, tm},
  AxesLabel -> {"t[s]", "delta[m]"}, PlotRange -> Automatic]
Pplotm = Plot[Evaluate[-Pep /. solep], {t, te, tm},
  AxesLabel -> {"t[s]", "P[N]"}, PlotRange -> Automatic]
```

$$P(t_{e-0}) = k_1 (1.9 \text{ delta})^{(3./2)}/.\text{delta}\rightarrow(1.9 \text{ yc}) = 679.5 \text{ [N]}$$

$$P(t_{e+0}) = P_{ep}/.\text{delta}\rightarrow(1.9 \text{ yc}) = 678.134 \text{ [N]}$$

end of elasto-plastic compression = maximum compression  $\theta'[t]=\omega_{gam}=0$

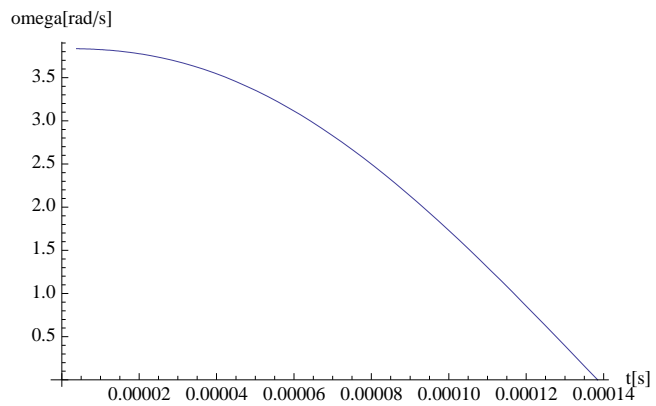
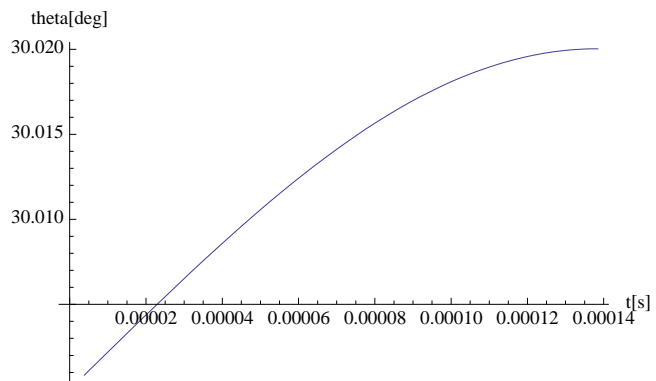
$$t_m = 0.000138444 \text{ [s]}$$

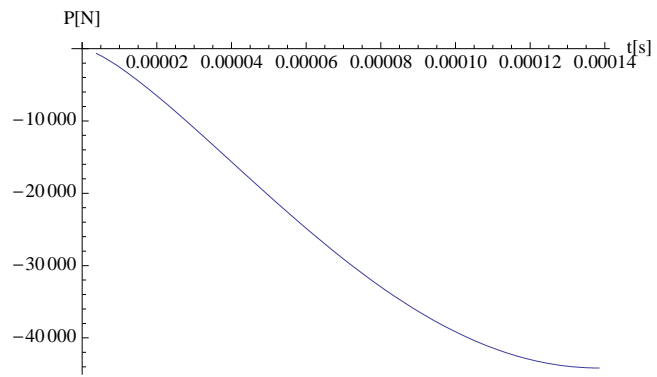
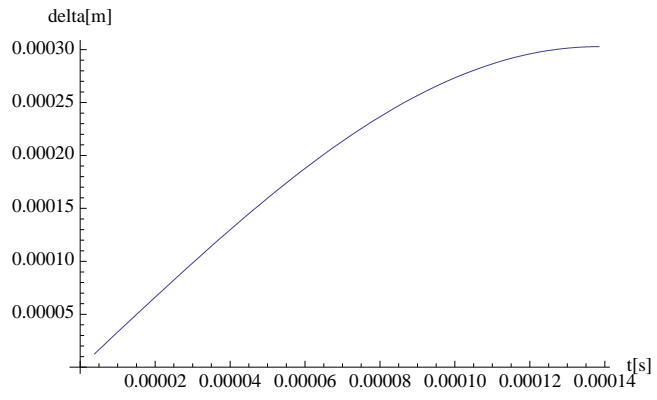
$$\theta_{tam} = 0.523948 \text{ [rad]} = 30.02 \text{ [deg]}$$

$$\omega_{gam} = 5.83227 \times 10^{-9} \text{ [rad/s]}$$

$$\delta_{tam} = 0.000302783 \text{ [m]}$$

$$P_m = -44161. \text{ [N]}$$





```

(*elastic restitution*)
deltams = deltam / yc;
deltar = deltam (1.02 (1 - ((deltams + 5.9) / 6.9) ^ -0.54));
Rr = 1 / (deltam - deltar) ^ 3 (3. / 4 Pm / Ep) ^ 2;
k1r = 2 / (3 (1 - nu ^ 2)) E1 Sqrt[Rr];

Print["Rr = ", Rr, " [m]"];
Print["deltar = ", deltar, " [m]"];
Print["k1r = ", k1r, " "];

Plr = {0, -k1r (delta - deltar) ^ 1.5, 0};

Print[" "];
Print["P(tm-0) = Pm = ", Pm, " [N]"];
Print["P(tm+0) = k1r (deltam-deltar)^1.5 = ", Plr[[2]] /. theta[t] -> thetam, " [N]"];
Print[" "];

eqf = Simplify[IO * alpha - Cross[rC, G] - Cross[rA, Plr]][[3]];

ttf = 0.0002196590448679869 - 0.0000000084448679869;

solf = NDSolve[{eqf == 0, theta[tm] == thetam, theta'[tm] == omegam},
  theta, {t, tm, ttf}, MaxSteps -> 10 000,
  Method -> {EventLocator, "Event" -> (delta - deltar)}];

tf = InterpolatingFunctionDomain[First[theta /. solf]][[1, -1]];
thetaf = Chop[(Evaluate[theta[t] /. solf] /. t -> tf)[[1]]];
omegaf = Chop[(Evaluate[D[theta[t] /. solf, t] /. t -> tf)[[1]]];
deltaf = delta /. theta[t] -> thetaf;
Plf = Plr[[2]] /. theta[t] -> thetaf;

thetaplotf = Plot[Evaluate[theta[t] /. solf] * 180 / Pi,
  {t, tm, tf}, AxesLabel -> {"t[s]", "theta[deg]"}, PlotRange -> Automatic]
omegaplotf = Plot[Evaluate[D[theta[t] /. solf, t]], {t, tm, tf},
  AxesLabel -> {"t[s]", "omega[rad/s]"}, PlotRange -> Automatic]
deltaplotf = Plot[Evaluate[delta /. solf], {t, tm, tf},
  AxesLabel -> {"t[s]", "delta[m]"}, PlotRange -> Automatic]
Pplotf = Plot[Evaluate[Plr[[2]] /. solf], {t, tm, tf},
  AxesLabel -> {"t[s]", "P[N]"}, PlotRange -> Automatic]

Print["end of restitution delta=deltar=", deltar, " [m]"]
Print["tf = ", tf, " [s]"]
Print["thetaf = ", thetaf, " [rad] = ", thetaf * 180 / Pi, " [deg]"]
Print["omegaf = ", omegaf, " [rad/s]"]
Print["deltaf = ", deltax, " [m]"]
Print["Pf = ", Chop[Plf], " [N]"]

ef = - omegaf / omega0;
Print[" "]
Print["e= - omegaf/omega0 = ", ef]

Rr = 0.0905921 [m]

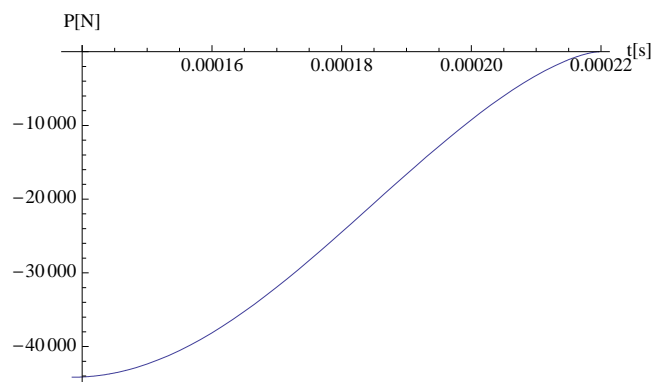
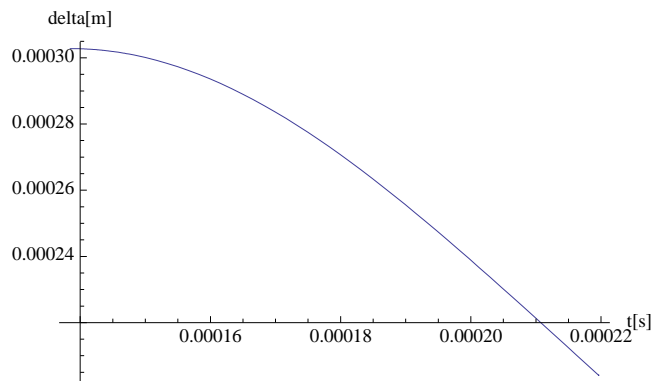
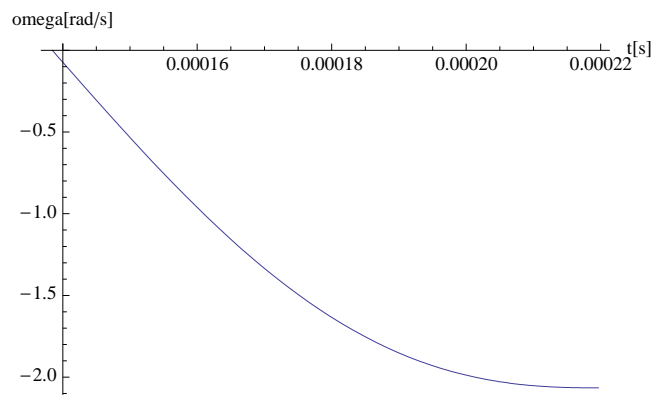
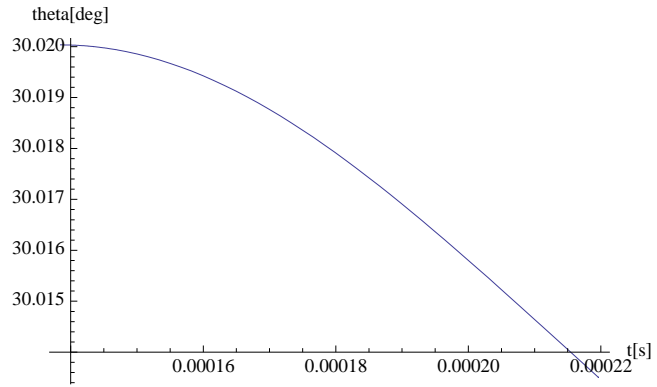
deltar = 0.000204082 [m]

k1r = 4.50358 x 1010

P(tm-0) = Pm = -44 161. [N]

P(tm+0) = k1r (deltam-deltar)^1.5 = -44 161. [N]

```



end of restitution delta=deltar=0.000204082 [m]

tf = 0.000219651 [s]

thetaf = 0.523834 [rad] = 30.0135 [deg]

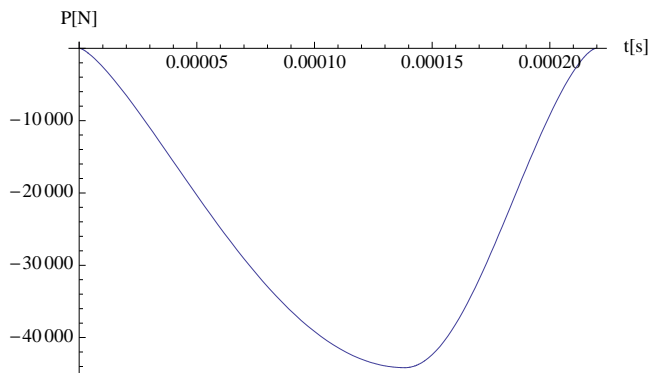
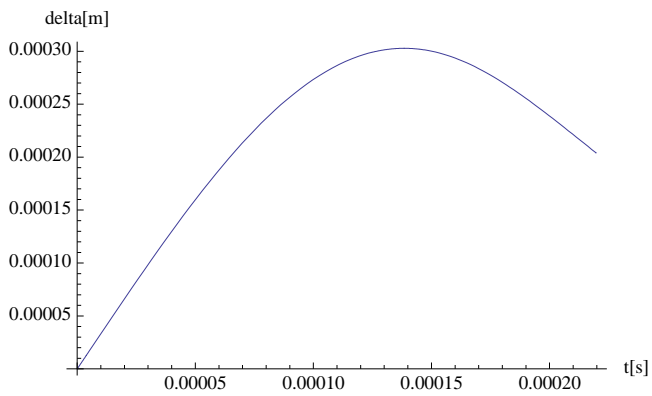
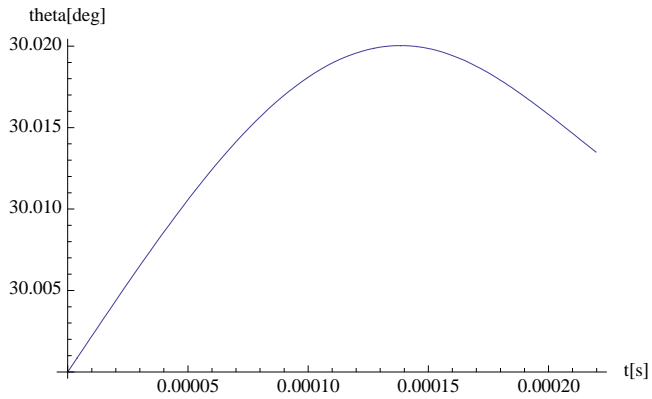
omegaf = -2.06545 [rad/s]

deltaf = 0.000204082 [m]

Pf = -0.000023504 [N]

e= - omegaf/omega0 = 0.538437

```
Show[thetaplot, thetaplotm, thetaplotf, PlotRange → Automatic]
Show[deltaplot, deltaplotm, deltaplotf, PlotRange → Automatic]
Show[Pplot, Pplotm, Pplotf, PlotRange → Automatic]
```



```
Print["Pc = 4./3 (R/Ep)^2 (Pi CJ Sy/2)^3 = ", PcJ, " [N]"];
vc = Sqrt[4 yc Pc / (5 m)];
Print["vc = Sqrt[4 yc Pc / (5 m)] = ", vc, " [m/s]"];
Print[" "];
vs = v0y / vc;
Print["vs = v0y/vc = ", vs];
eJ = 1 - 0.1 Log[vs] ((vs - 1) / 59) ^ .156;
Print["eJ = 1-0.1 Log[vs] ((vs-1)/59)^.156 = ", eJ];
```

$$Pc = 4./3 (R/Ep)^2 (Pi CJ Sy/2)^3 = PcJ [N]$$

$$vc = Sqrt[4 yc Pc / (5 m)] = 0.0238214 [m/s]$$

$$vs = v0y/vc = 139.458$$

$$eJ = 1 - 0.1 \text{ Log}[vs] ((vs-1)/59)^{.156} = 0.435943$$