In[724]:=
```
(* 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];

R = 0.01;
V = 4 Pi R^3 / 3;
m = 0.0327;

E1 = 200 * 10^9;
Sy = 1.12 * 10^9;
nu = 0.33;

 Ep = ((1 - nu^2) / E1 + (1 - nu^2) / E1) ^-1;
```

$$k1 = \frac{2}{3\ (1 - nu\hat{\ }2)}\ E1\ \sqrt{R}\ ;$$

```
g = 9.81;
h = 0.1;
v0 = Sqrt[2 g h];

H = 3 Sy;

(* Chang *)
K = 0.454 + 0.41 nu;
xc = (Pi K H / (2 Ep)) ^2 R;

(* Jackson *)
CJ = 1.295 E^(0.736 nu);
xcJ = (Pi CJ Sy / (2 Ep)) ^2 R;

Print["Chang: xc = ", xc, " [m]"]
Print["Jackson: xc = ", xcJ, " [m]"]

(*  m x''[t]==m g-k1 x[t]^(3./2) *)
eq = m x''[t] == m g - k1 (x[t]) ^ (3. / 2);

Print[" "]
Print["Chang"];

solC = NDSolve[{eq, x[0] == 0, x'[0] == v0}, x, {t, 0, Infinity},
   Method → {EventLocator, "Event" → (x[t] - 1.9 xc)}];

tIC = InterpolatingFunctionDomain[First[x /. solC]][[1, -1]];

xIC = Chop[First[Evaluate[x[t] /. solC] /. t → tIC]];
vIC = Chop[First[Evaluate[x'[t] /. solC] /. t → tIC]];

Print["tI = ", tIC, " [s]"];
Print["xI = ", xIC, " [m]"];
Print["vI = ", vIC, " [m/s]"];

Print[" "];
Print["Jackson"];
Print[" "];
Print["critical values "];
```

```
Print["xc = ", xcJ, " [m]"]
Pc = k1 xcJ ^ (3. / 2);
Print["Pc = k1 xc^(3./2) = ", Pc, " [N]"];
PcJ = 4. / 3 (R / Ep) ^ 2 (Pi CJ Sy / 2) ^ 3;
Print["Pc = 4./3 (R/Ep)^2 (Pi CJ Sy/2)^3 = ", PcJ, " [N]"];
vc = Sqrt[4 xc PcJ / (5 m)];
Print["vc = Sqrt[4 xc PcJ /(5 m)] = ", vc, " [m/s]"];
Print[" "];

(* I elastic compression *)

solJ = NDSolve[{eq, x[0] == 0, x'[0] == v0}, x, {t, 0, Infinity},
   Method → {EventLocator, "Event" → (x[t] - 1.9 xcJ)}];

tIJ = InterpolatingFunctionDomain[First[x /. solJ]][[1, -1]];

xIJ = Chop[First[Evaluate[x[t] /. solJ] /. t → tIJ]];
vIJ = Chop[First[Evaluate[x'[t] /. solJ] /. t → tIJ]];

Print["tI = ", tIJ, " [s]"];
Print["xI = ", xIJ, " [m]"];
Print["vI = ", vIJ, " [m/s]"];

gI = Plot[Evaluate[x[t] /. solJ], {t, 0, tIJ}, AxesLabel → {"t[s]", "x [m]"}, AxesOrigin → {0, 0}]
P = k1 (Evaluate[(x[t] /. solJ)]) ^1.5;
pI = Plot[Evaluate[P], {t, 0, tIJ}, AxesLabel → {"t[s]", "P [N]"}, AxesOrigin → {0, 0}]
```

Chang: xc = 7.68147×10^{-6} [m]

Jackson: xc = 6.69933×10^{-6} [m]

Chang

tI = 0.0000106616 [s]

xI = 0.0000145948 [m]

vI = 1.29012 [m/s]

Jackson

critical values

xc = 6.69933×10^{-6} [m]

Pc = k1 xc^(3./2) = 259.453 [N]

Pc = 4./3 (R/Ep)^2 (Pi CJ Sy/2)^3 = 259.453 [N]

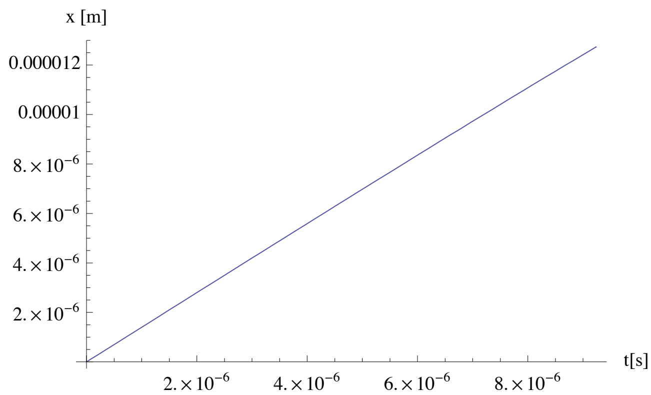
vc = Sqrt[4 xc PcJ /(5 m)] = 0.220812 [m/s]

tI = 9.23406×10^{-6} [s]

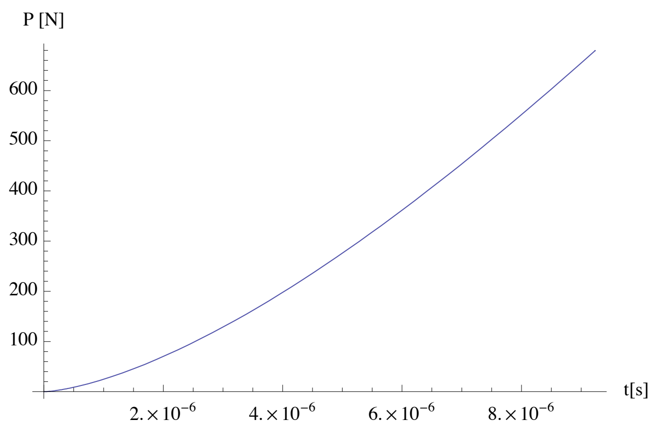
xI = 0.0000127287 [m]

vI = 1.32312 [m/s]

Out[778]=



Out[780]=



In[781]:=

```
(* II elasto-plastic compression *)
ey = Sy / Ep;
B = 0.14 E^(23 ey);
a = Sqrt[R x[t] (x[t] / (1.9 xcJ))^B];
HS = 2.84 - 0.92 (1 - Cos[Pi a / R]);
y = x[t] / xcJ;
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(tI-0)  = k1  (1.9 xc)^(3./2)  = ", k1 (1.9 xcJ)^(3. / 2) , "  [N]"];
Print["P(tI+0)  = Pep/.x[t]→(1.9 xc)  = ", Pep /. x[t] → (1.9 xcJ) , "  [N]"];
Print[" "];

solep = NDSolve[{ m x''[t] == m g - Pep, x[tIJ] == xIJ, x'[tIJ] == vIJ}, x, {t, tIJ, Infinity},
   Method → {EventLocator, "Event" → x'[t]}];

tm = InterpolatingFunctionDomain[First[x /. solep]][[1, -1]];
xm = Chop[First[Evaluate[x[t] /. solep] /. t → tm]];
vm = Chop[First[Evaluate[x'[t] /. solep] /. t → tm]];
Pm = Pep /. x[t] → xm;

Print["tm = ", tm, "  [s]"];
Print["xm = ", xm, "  [m]"];
Print["vm = ", vm, "  [m/s]"];
Print["Pm = ", Pm, "  [N]"];

gII = Plot[Evaluate[x[t] /. solep],
  {t, tIJ, tm}, AxesLabel → {"t[s]", "x  [m]"}, AxesOrigin → {0, 0}]
pII = Plot[Evaluate[Pep /. solep], {t, tIJ, tm},
  AxesLabel → {"t[s]", "P  [N]"}, AxesOrigin → {0, 0}]
```

P(tI-0) = k1 (1.9 xc)^(3./2) = 679.5 [N]

P(tI+0) = Pep/.x[t]→(1.9 xc) = 678.134 [N]

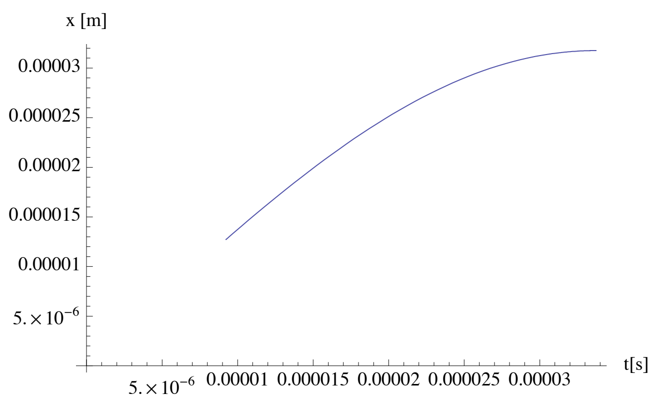
tm = 0.0000337225 [s]

xm = 0.0000317578 [m]

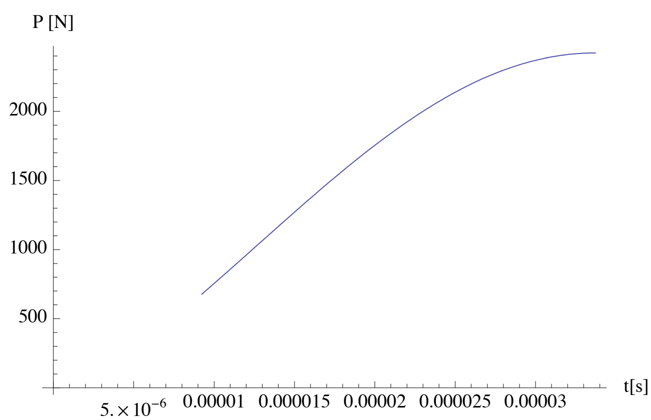
vm = -1.33432×10^{-10} [m/s]

Pm = 2422.18 [N]

Out[800]=



Out[801]=



In[802]:=
```
(* III elastic restitution *)

CJ = 1.295 E^(0.736 nu);
xcJ = (Pi CJ Sy / (2 Ep))^2 R;
xms = xm / xcJ;
xr = xm (1.02 (1 - ((xms + 5.9) / 6.9)^-0.54));
Rr = 1 / (xm - xr)^3 (3. / 4 Pm / Ep)^2;
k1r = 2 / (3 (1 - nu^2)) E1 Sqrt[Rr];

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

Print["P(tII-0) = Pm = ", Pm , " [N]"];
Print["P(tII+0) = k1r (xm-xr)^1.5 = ", k1r (xm - xr)^1.5 , " [N]"];

ttf = 0.00006396239413318553;
solr = NDSolve[{ m x''[t] == m g - k1r (x[t] - xr)^1.5, x[tm] == xm, x'[tm] == 0},
   x, {t, tm, ttf}, MaxSteps → 200000,
   Method → {EventLocator, "Event" → (x[t] - xr)}];

tr = InterpolatingFunctionDomain[First[x /. solr]][[1, -1]] - 3×10^-8;

gIII = Plot[Evaluate[x[t] /. solr],
  {t, tm, tr}, AxesLabel → {"t[s]", "x [m]"}, AxesOrigin → {0, 0}]

pIII = Plot[Evaluate[(k1r (x[t] - xr)^1.5) /. solr],
  {t, tm, tr}, AxesLabel → {"t[s]", "P [N]"}, AxesOrigin → {0, 0}]

xf = Chop[First[Evaluate[x[t] /. solr] /. t → tr]];
vf = Chop[First[Evaluate[x'[t] /. solr] /. t → tr], 10^-9];

Print["tf = ", tr, " [s]"];
Print["xf = ", xf, " [m]"];
Print["xr = ", xr, " [m]"];
Print["vf = ", vf, " [m/s]"];

Print[" "];
Print["v0 = ", v0, " [m/s]"];
Print["e = -vf/v0 = ", -vf / v0]

vs = v0 / vc;
Print["vs = v0/vc = ", vs];
eJ = 1 - 0.1 Log[vs] ((vs - 1) / 59)^.156;
Print["eJ = 1-0.1  Log[vs]((vs-1)/59)^.156 = ", eJ];
```

Rr = 0.0167672 [m]

xr = 6.75574×10^{-6} [m]

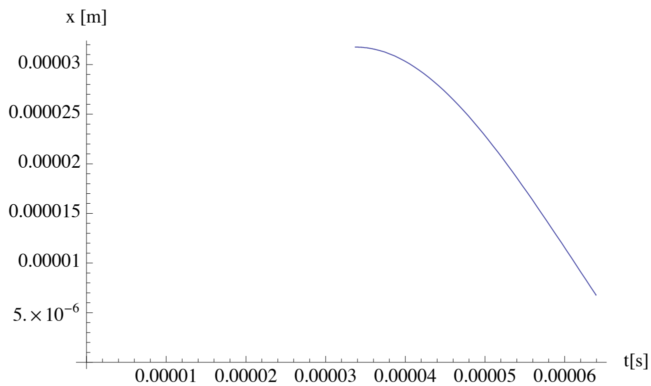
k1r = 1.9375×10^{10}

P(tII-0) = Pm = 2422.18 [N]

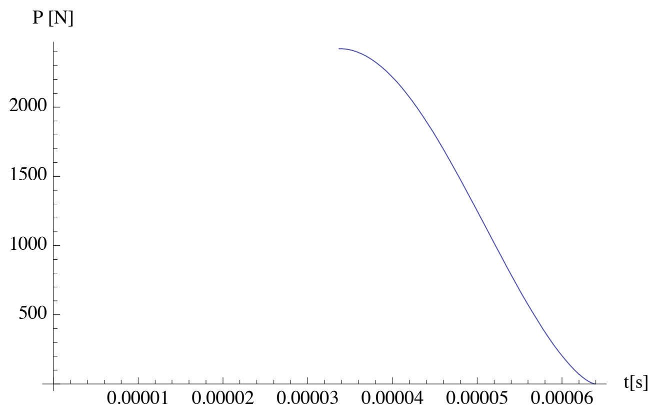
P(tII+0) = k1r (xm-xr)^1.5 = 2422.18 [N]

NDSolve::evre :
The value of the event function at t = 0.00006396239413318553` was not a real number. The event will be ignored in steps where it does not evaluate to real numbers at both ends. ≫

Out[815]=



Out[816]=



tf = 0.0000639324 [s]

xf = 6.78176×10^{-6} [m]

xr = 6.75574×10^{-6} [m]

vf = -1.217 [m/s]

v0 = 1.40071 [m/s]

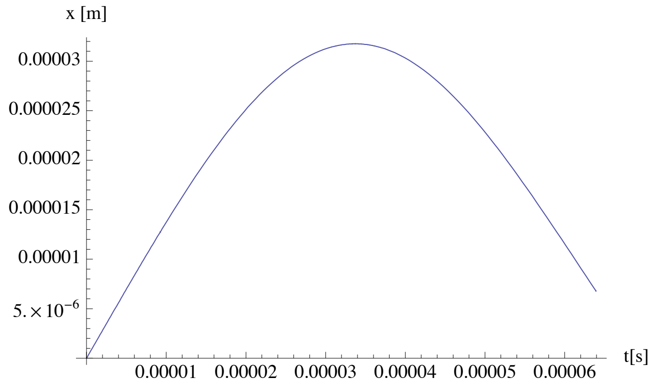
e = -vf/v0 = 0.868841

vs = v0/vc = 6.34347

eJ = 1-0.1 Log[vs]((vs-1)/59)^.156 = 0.872985

In[830]:=
```
Show[gI, gII, gIII, PlotRange → Automatic]
Show[pI, pII, pIII, PlotRange → Automatic]
```

Out[830]=



Out[831]=