Roadster Thrusters

I have attached the charts separately.

These notes are my working notes.

Megan E S Macafee

Set Up and Assumptions

Let us assume that the car's traction management system uses static friction (i.e. no slipping or water) and does not fall-back to kinetic friction - typically 25% less acceleration. So $\mu = 1.0$ (Cast Iron/copper $\mu \sim 1.05$ / ice/ice ~ 0.02).

https://en.wikipedia.org/wiki/Gravitational_acceleration

In[•]:=
```
g = 9.80665;
oneInch = 2.54 × 10^-2; (* m *)
oneMilePerHour = (5280 × 12 oneInch)/60^2 (* m/s *);
sixtyMPH = 60 oneMilePerHour
```

Out[•]= 26.8224

In[•]:=
```
oneHundredMPH = 100 oneMilePerHour
```

Out[•]= 44.704

In[•]:=
```
AvailableAcceleration[carMass_] := Module[{μstatic = 1.0, downForce, forwardForce},
  (* no aerodynamics for the moment *)
  downForce = g carMass;
  forwardForce = μstatic downForce;
  (* in the simple case with no aerodynamics the mass CANCELS OUT *)
  forwardForce/carMass
 ]
```

It is actually a lot more complicated than this http://theracingline.net/2018/race-car-tech/race-tech-explained/-tyres-combined-tyre-forces/

In[•]:=
```
estRoadsterMass = 2300 (* guessed from 2300 kg Model X *)
```

Out[•]= 2300

In[•]:=
```
carLength = 179 oneInch;
carWidth = 82 oneInch;
carArea = carLength carWidth (* Again all of these are guessed using the Model X data*)
```

Out[•]= 9.46966

In[•]:=
```
standardAtmosphere = 101 325 (* Newtons per m^2 *)
```
Out[•]= 101 325

Without Thrusters Normal

v = u + at

u = 0

v = at (in our case) and $t = \frac{v}{a}$

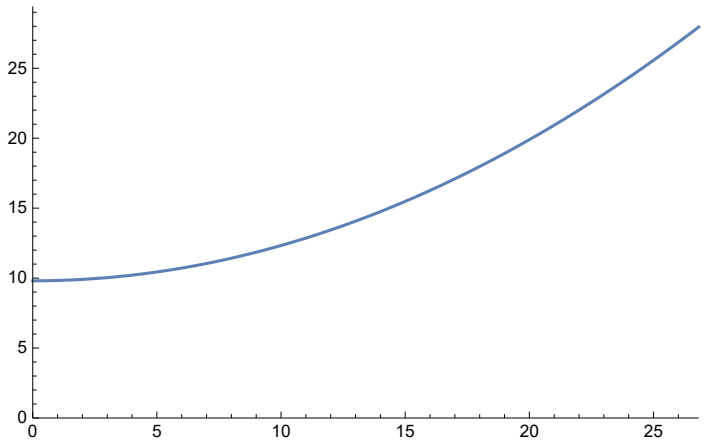
In[•]:=
```
time60 = sixtyMPH / AvailableAcceleration[estRoadsterMass] (* 1.9 seconds quoted *)
```
Out[•]= 2.73512

In[•]:=
```
time100 = oneHundredMPH / AvailableAcceleration[estRoadsterMass] (* 4.2 seconds quoted *)
```
Out[•]= 4.55854

In[•]:=
```
AvailableAcceleration[carMass_, velocity_] :=
 Module[{μstatic = 1.0, downForce, forwardForce, unknown = 58},
  (* no aerodynamics for the moment *)
  downForce = g carMass + unknown velocity^2;
  (* I looked at other aerodynamic structures and made this guess *)
  forwardForce = μstatic downForce;

  forwardForce / carMass
 ]
```

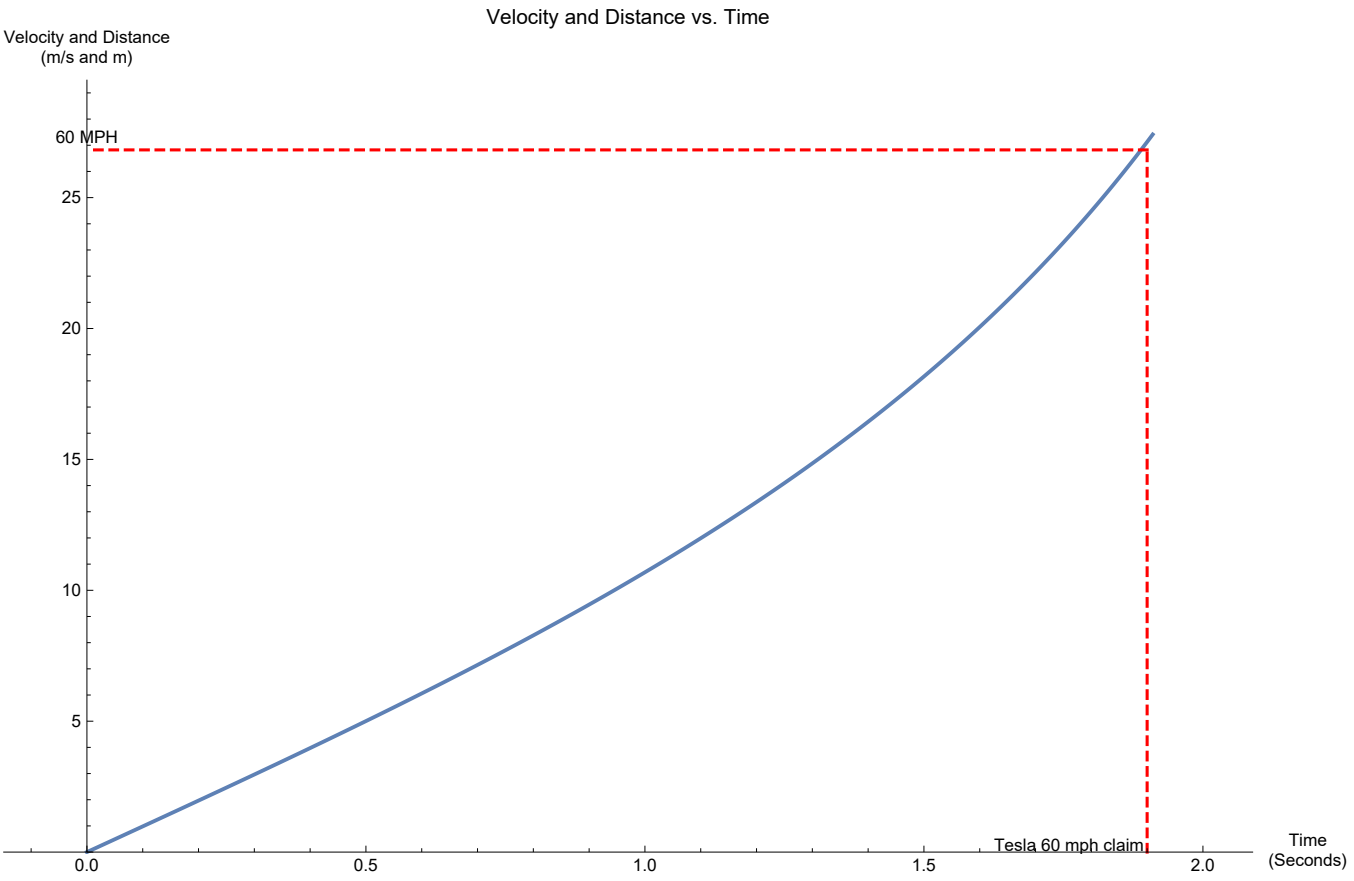
In[•]:=
```
Expand[AvailableAcceleration[estRoadsterMass, vel]]
```
Out[•]= 9.80665 + 0.0252174 vel^2

In[•]:=
```
Plot[AvailableAcceleration[estRoadsterMass, v],
 {v, 0, sixtyMPH}, PlotRange → {{0, sixtyMPH}, {0, All}}]
```
Out[•]=



In[ ]:=
```
time60claim = 1.9;
time100claim = 4.2;

position = 0; velocity = 0; step = 0.01;
history = Append[{}, {0, position, velocity}];

Do[
  position = position + velocity step;
  velocity = velocity + AvailableAcceleration[estRoadsterMass, velocity] step;
  AppendTo[history, {t + step, position, velocity}],
  {t, 0, time60claim, step}
 ];

velocityPlot = ListLinePlot[
   Map[#[[{1, 3}]] &, history],
   PlotLabel → "Velocity and Distance vs. Time",
   PlotStyle → Directive[Thick],
   PlotRange → {{-.15, time60claim 1.1}, {0, sixtyMPH 1.1}},
   AxesLabel → {"Time\n(Seconds)", "Velocity and Distance\n(m/s and m)"},
   ImageSize → 700
  ];

sixtyPlot = ListLinePlot[
   {
    Labeled[{time60claim, 0}, "Tesla 60 mph claim"],
    {time60claim, sixtyMPH}
   },
   PlotStyle → Directive[Red, Dashed]
  ];

speedPlot60 = ListLinePlot[
   {{time60claim, sixtyMPH}, Labeled[{0, sixtyMPH}, "60 MPH", {Top}]},
   PlotStyle → Directive[Red, Dashed]];

Show[{
  velocityPlot, sixtyPlot, speedPlot60
 }]
```

Out[ ]=



In[ ]:= `peakAeroForceGuess = 58 sixtyMPH^2`

Out[ ]= 41 727.6

In[ ]:= `pressureDifference = peakAeroForceGuess / carArea (* Newtons/m^2 *)`

Out[ ]= 4406.45

In[ ]:= `percentOfAtmos = pressureDifference / standardAtmosphere 100 (* % of one atmosphere *)`

Out[ ]= 4.34883

So, we have model that gives the right 0-60 time if the 60 mile per hour downforce is about 4.5 tonnes - which suggest that the suction at 60 mph is 4% of atmospheric.

The 0-100 mph times are depressed by comparison (they do not continue to grow as fast as the curve above) - I assume that the engines are not powerful enough to stay on the upper limit of the sticky-friction curve?

Comparison Data

In[ ]:=
```
MPHPosition[lineList_] := Position[lineList, "mph"][[1, 1]];
```

```
zeroSixtyWebData = Import["https://www.0-60specs.com/0-60-times/"];
```

... $CharacterEncoding: The byte sequence {240} could not be interpreted as a character in the UTF-8 character encoding.

... $CharacterEncoding: The byte sequence {159} could not be interpreted as a character in the UTF-8 character encoding.

... $CharacterEncoding: The byte sequence {153} could not be interpreted as a character in the UTF-8 character encoding.

... General: Further output of $CharacterEncoding::utf8 will be suppressed during this calculation.

In[ ]:=
```
goodLines = Select[StringSplit[zeroSixtyWebData, "\n"], StringContainsQ[#, "@"] &];
```

In[ ]:=
```
wordListInLineList = Map[
    StringSplit,
    goodLines
   ];
plotDataRaw = Map[{
      StringRiffle[#[[1 ;; MPHPosition[#] - 10]]],
      ToExpression[#[[MPHPosition[#] - 9]]],
      ToExpression[#[[MPHPosition[#] - 6]]]
     } &,
    wordListInLineList
   ];

plotData = Select[plotDataRaw, Between[#[[2]], {1949, 2019}] && Between[#[[3]], {0, 500}] &];
```

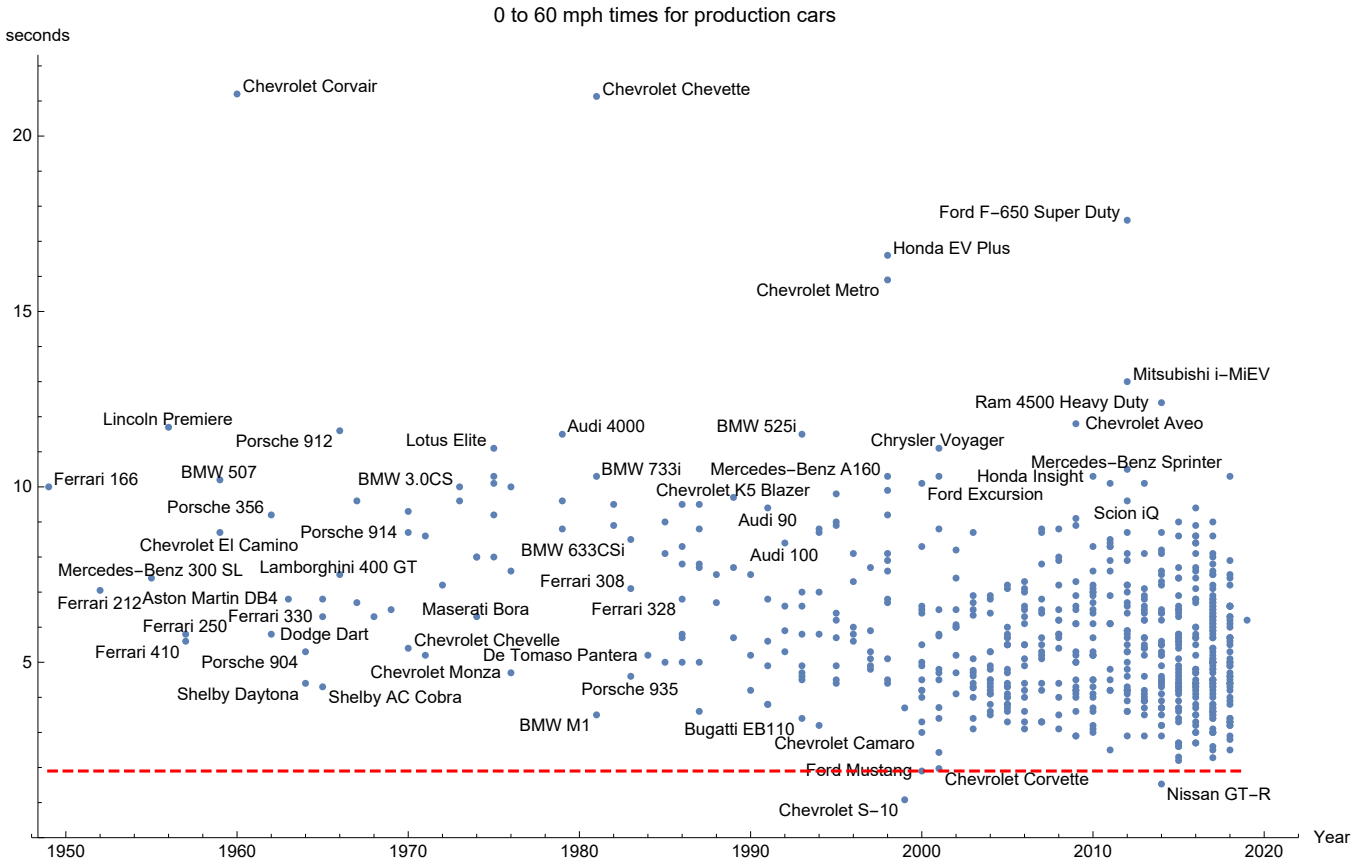
ToExpression::sntxi: Incomplete expression; more input is needed .

In[ ]:=
```
{firstYear, lastYear} = MinMax[Map[#[[2]] &, plotData]]
```

Out[ ]=
```
{1949, 2019}
```

In[ ]:=
```
Show[{
  ListPlot[
   Map[Labeled[#[[2 ;;]], #[[1]]] &, plotData],
   PlotLabel → "0 to 60 mph times for production cars",
   AxesLabel → {"Year", "seconds"},
   PlotRange → {{firstYear - 1, lastYear + 3}, {0, All}},
   ImageSize → 700
  ],
  ListLinePlot[{{firstYear, time60claim}, {lastYear, time60claim}},
   PlotStyle → Directive[{Red , Dashed}]]
 }]
```

Out[ ]=



The very fast cars are getting help with their acceleration probably using aerodynamics.

With Thrusters (In addition to existing aerodynamic forces and friction)

v = u + at

u = 0

v = at (in our case) and $t = \frac{v}{a}$

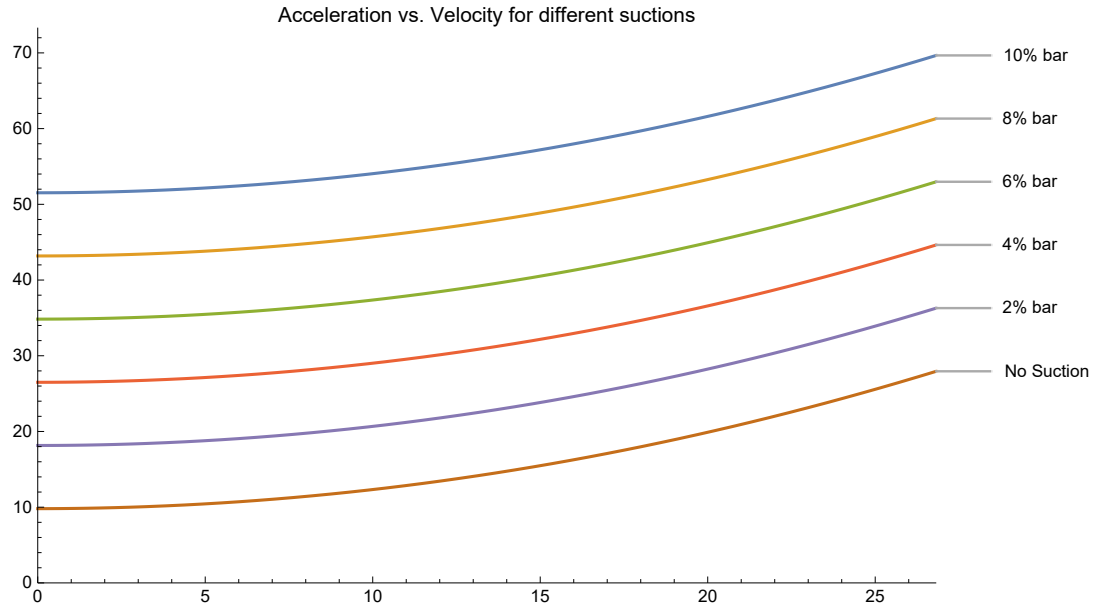
```
In[ ]:= ThrusterAcceleration[carMass_, velocity_, fractionAtmosphere_] :=
 Module[{μstatic = 1.0, downForce, forwardForce, unknown = 58, suction},
  suction = standardAtmosphere carArea fractionAtmosphere (* fractionAtmosphere atmospheres *);
  (* no aerodynamics for the moment *)
  downForce = g carMass + unknown velocity^2 + suction;
  forwardForce = μstatic downForce;

  forwardForce
    carMass
 ]
```

```
In[ ]:= Expand[ThrusterAcceleration[estRoadsterMass, vel, 0.1]]
Out[ ]= 51.5246 + 0.0252174 vel^2
```

```
In[ ]:= Expand[ThrusterAcceleration[estRoadsterMass, vel, 0]]
Out[ ]= 9.80665 + 0.0252174 vel^2
```

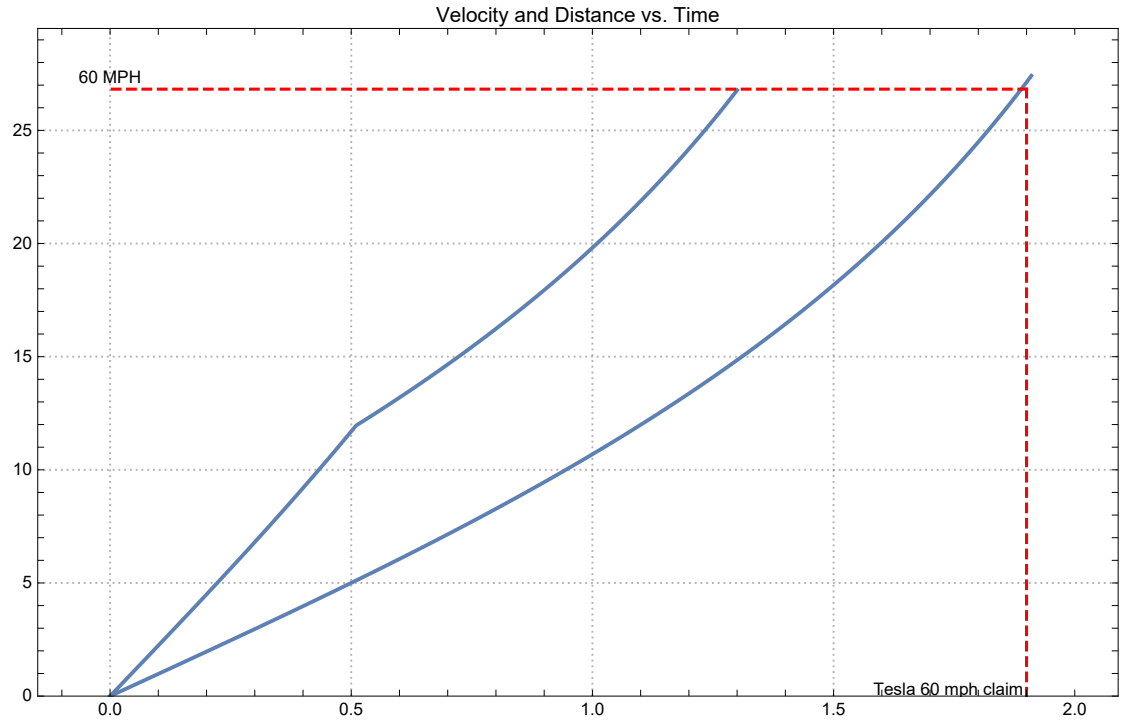
```
In[ ]:= Plot[{
   ThrusterAcceleration[estRoadsterMass, v, .1],
   ThrusterAcceleration[estRoadsterMass, v, .08],
   ThrusterAcceleration[estRoadsterMass, v, .06],
   ThrusterAcceleration[estRoadsterMass, v, .04],
   ThrusterAcceleration[estRoadsterMass, v, .02],
   ThrusterAcceleration[estRoadsterMass, v, 0]
  }, {v, 0, sixtyMPH},
  PlotRange → {{0, sixtyMPH}, {0, All}},
  PlotLabel → "Acceleration vs. Velocity for different suctions",
  ImageSize → Large,
  PlotLabels → {"10% bar", "8% bar", "6% bar", "4% bar", "2% bar", "No Suction"}
 ]
```

Out[ ]=



In[ ]:=
```
position = 0; velocity = 0; step = 0.01;
history = Append[{}, {0, position, velocity}];

suctionGenerated = .03;(* fraction of an atmosphere *)
suctionDuration = .5; (* duration of suction *)

Do[
  position = position + velocity step;
  velocity =
   velocity +
    If[
      t ≤ suctionDuration,
      ThrusterAcceleration[estRoadsterMass, velocity, suctionGenerated],
      ThrusterAcceleration[estRoadsterMass, velocity, 0]
     ] ×
     step;
  AppendTo[history, {t + step, position, velocity}],
  {t, 0, 1.29, step}
 ];

suctionVelocityPlot = ListLinePlot[
   Map[#[[{1, 3}]] &, history],
   PlotLabel → "Velocity and Distance vs. Time",
   PlotStyle → Directive[Thick],
   PlotRange → {{-.15, time60claim 1.1}, {0, sixtyMPH 1.1}},
   AxesLabel → {"Time\n(Seconds)", "Velocity and Distance\n(m/s and m)"},
   PlotTheme → "Detailed",
   ImageSize → Large
  ];

sixtyPlot = ListLinePlot[
   {
    Labeled[{time60claim, 0}, "Tesla 60 mph claim"],
    {time60claim, sixtyMPH}
   },
   PlotStyle → Directive[Red, Dashed]
  ];

speedPlot60 = ListLinePlot[
   {{time60claim, sixtyMPH}, Labeled[{0, sixtyMPH}, "60 MPH", {Top}]},
   PlotStyle → Directive[Red, Dashed]];

Show[{
  suctionVelocityPlot, velocityPlot, sixtyPlot, speedPlot60
 }]
```

Out[ ]=


Velocity and Distance vs. Time
60 MPH
Tesla 60 mph claim
0
5
10
15
20
25
0.0
0.5
1.0
1.5
2.0