Altimeter Data Fitting
&
Apogee Prediction

Tom Mattison
UBC Rocket Lunch
18 February 2018

No-Drag Case

If there were no drag, altitude would be parabolic with time

$$y(t) = vt - \frac{1}{2}gt^2$$

This is maximum at $t_{\max} = v/g$, where the value is $y_{\max} = v^2/2g$

We can then write $y(t - t_{\max}) = y_{\max} - \frac{1}{2}g \cdot (t - t_{\max})^2$

Add Drag

Drag adds to gravity a force proportional to velocity squared

$$F = -gm - cmv^2$$

The factor of *m* in the drag term is just for convenience.

Then the equation of motion is $F = ma \rightarrow -gm - cmv^2 = m\dfrac{dv}{dt}$

Doing the usual tricks we get $\quad -dt = \dfrac{dv}{g + cv^2}$

Integrating both sides gives $\; t = -\dfrac{\tan^{-1}\left(\sqrt{c/g} \cdot v\right)}{\sqrt{cg}} + C$

Add Drag 2

Inverting this gives $v(t) = -\sqrt{\frac{g}{c}}\tan\left(\sqrt{cg}\cdot(t-C)\right)$

At apogee, we will have $v = 0$, which requires the argument of the tangent function to be zero, or $t = C$, so $C = t_{\max}$.

$$v(t-t_{\max}) = -\sqrt{\frac{g}{c}}\tan\left(\sqrt{cg}\cdot(t-t_{\max})\right)$$

To get altitude vs time, we integrate the velocity.

$$y(t-t_{\max}) = \frac{1}{c}\ln\left(\cos\left(\sqrt{cg}\cdot(t-t_{\max})\right)\right)+C$$

Add Drag 3

At $t = t_{max}$, the cosine is 1 so the log is zero, so we just have C, which must be y_{max}. So we have

$$y(t - t_{max}) = y_{max} + \frac{1}{c} \ln\left(\cos\left(\sqrt{cg} \cdot (t - t_{max})\right)\right)$$

If we define $t_{drag} = 1/\sqrt{cg}$ we can write this as

$$y(t - t_{max}) = y_{max} + g \cdot t_{drag}^2 \ln\left(\cos \frac{t - t_{max}}{t_{drag}}\right)$$

This is the functional form for altitude vs time with drag proportional to velocity (reasonable if sub-sonic)

Meanings of Parameters

The meanings of t_{max} and y_{max} are pretty obvious: they are the time and altitude of the apogee.

The g parameter is the acceleration of gravity.

The t_{drag} parameter is the time scale away from the apogee where the velocity is high enough for drag to be important.

Expansion Around Apogee

The Taylor expansion of cos(x) around $x = 0$ is $1-\frac{1}{2}x^2+...$

The Taylor expansion of ln(1+x) around $x = 0$ is x + ...

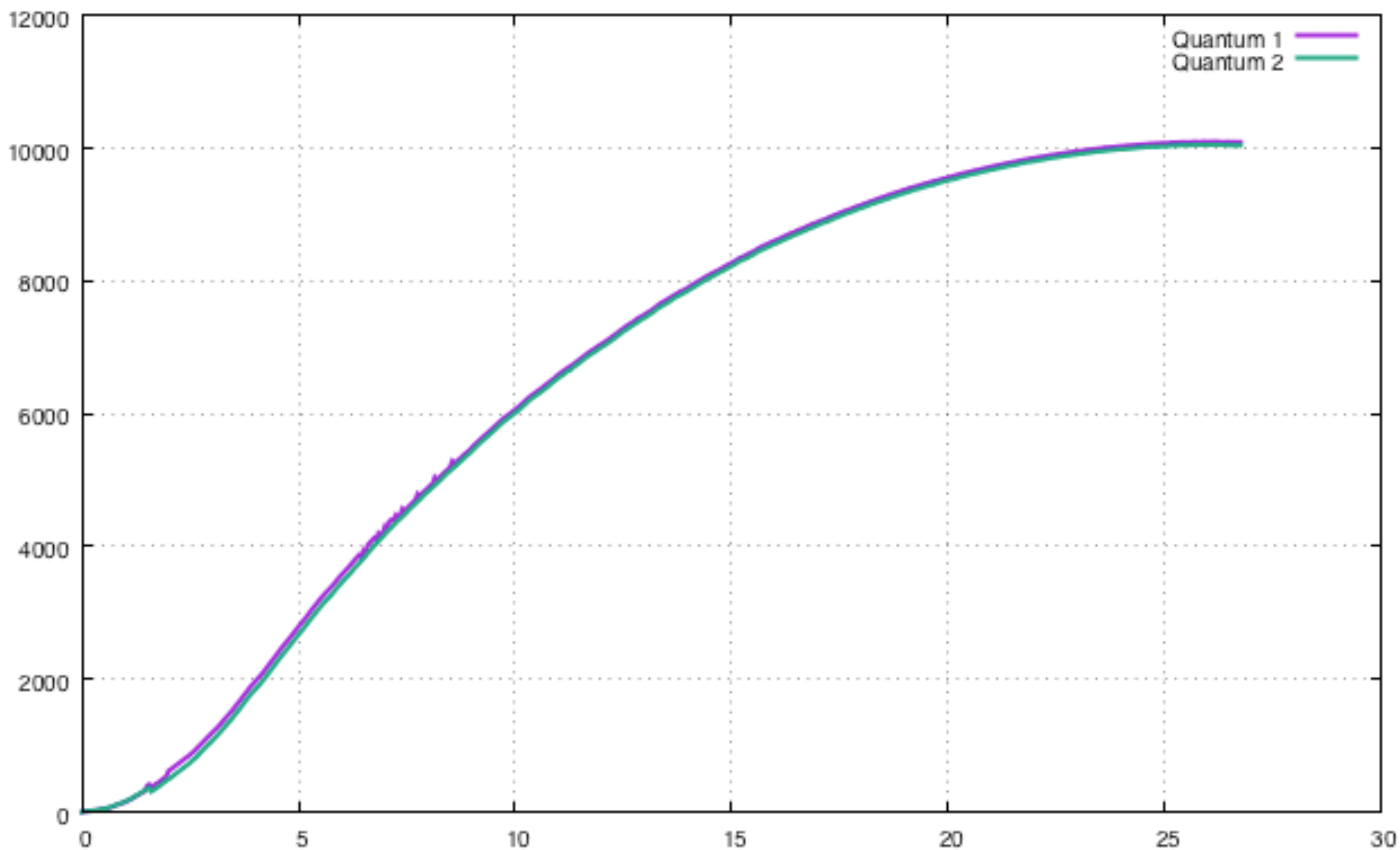
So the expansion of ln(cos(x)) is $-x^2/2$ + ...

Then the expansion of $\ln\left(\cos\frac{t-t_{\text{max}}}{t_{\text{drag}}}\right)$ is $-\frac{1}{2}\left(\frac{t-t_{\text{max}}}{t_{\text{drag}}}\right)^2$

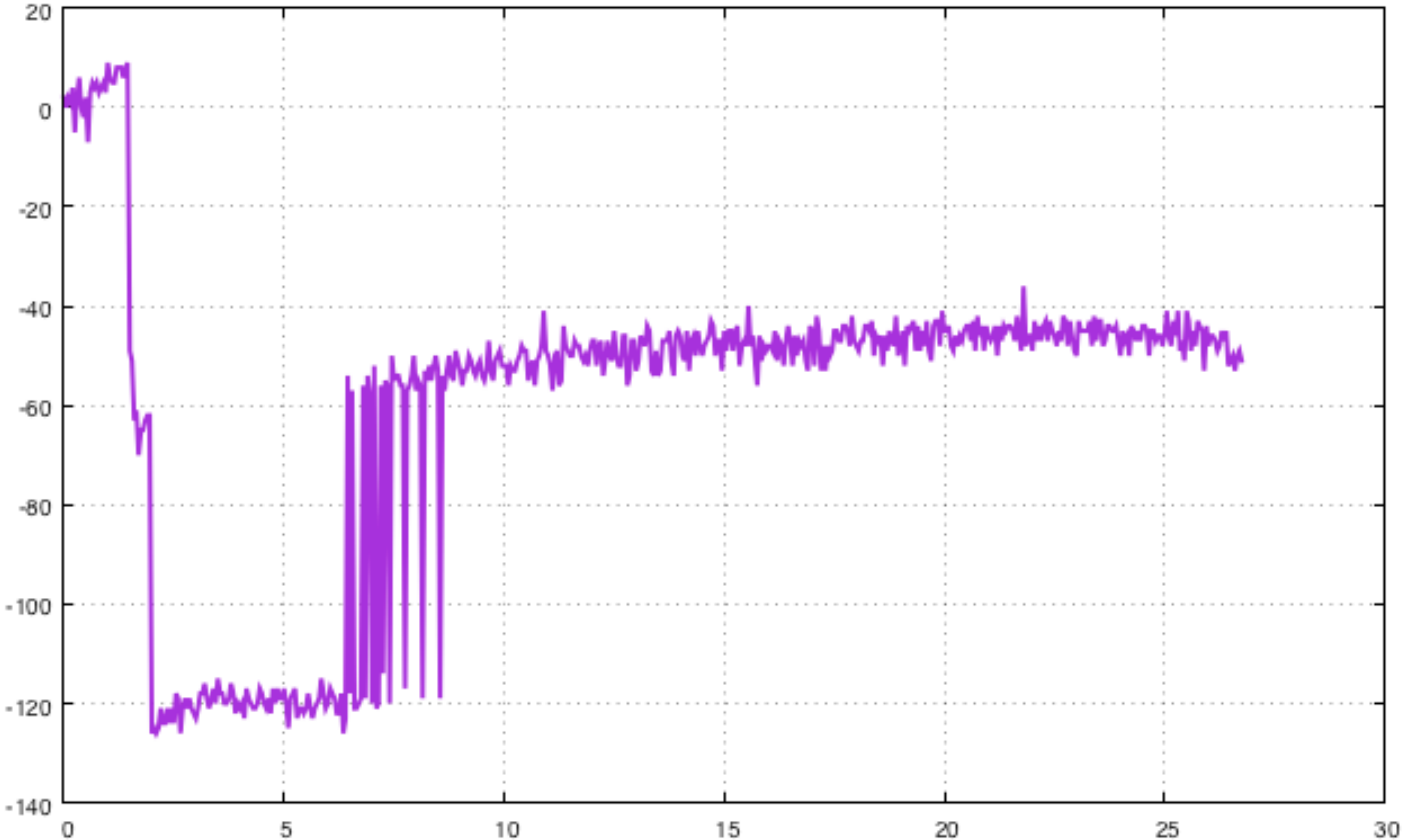
That gives $y\left(t-t_{\text{max}}\right)=y_{\text{max}}-\frac{1}{2}g\cdot\left(\frac{t-t_{\text{max}}}{t_{\text{drag}}}\right)$ as expected.

Competition Data

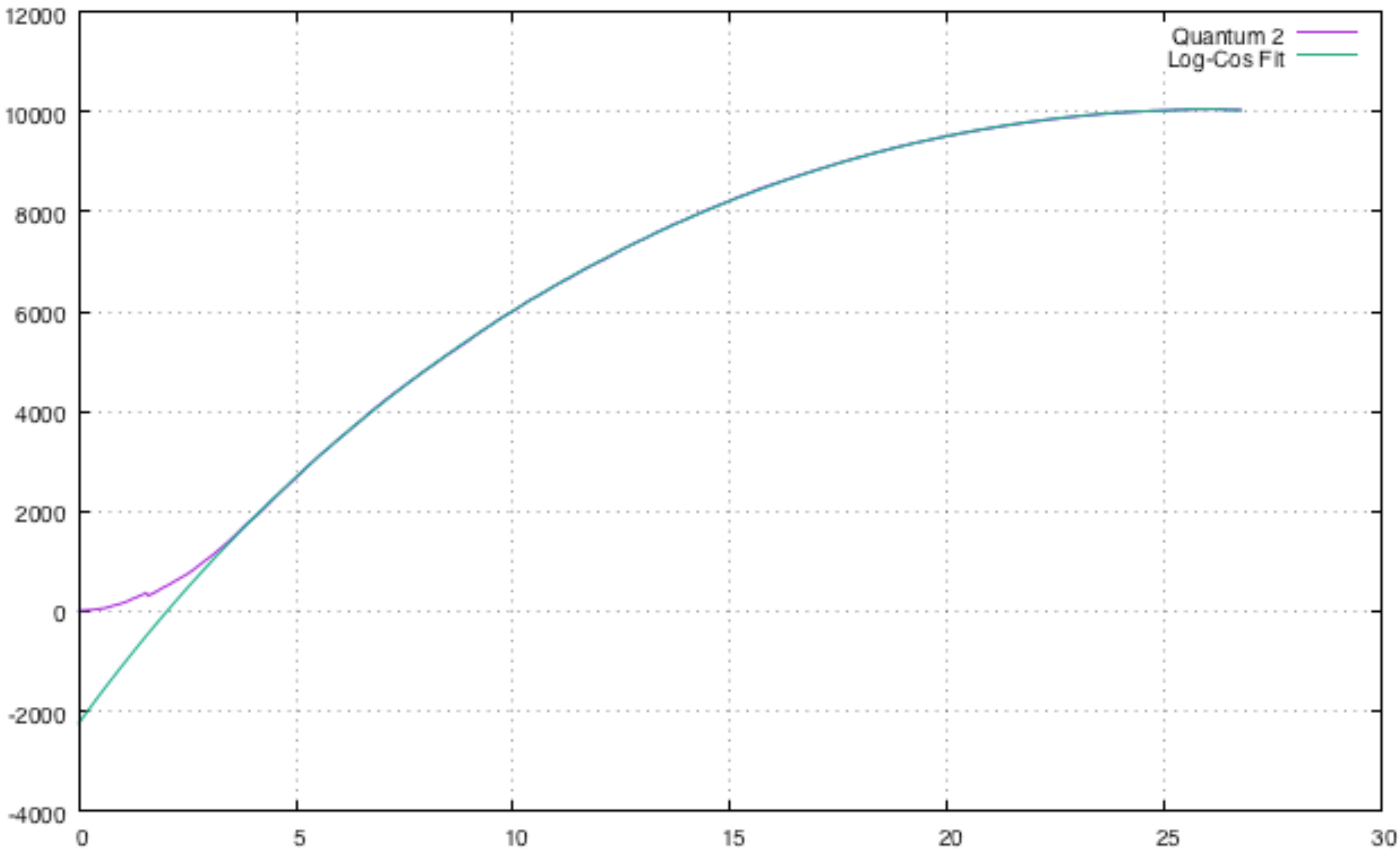
The “Quantum 1” and “Quantum 2” altimeter files, in feet.
“Quantum 2” is shifted by 0.4 seconds to match ejection time.



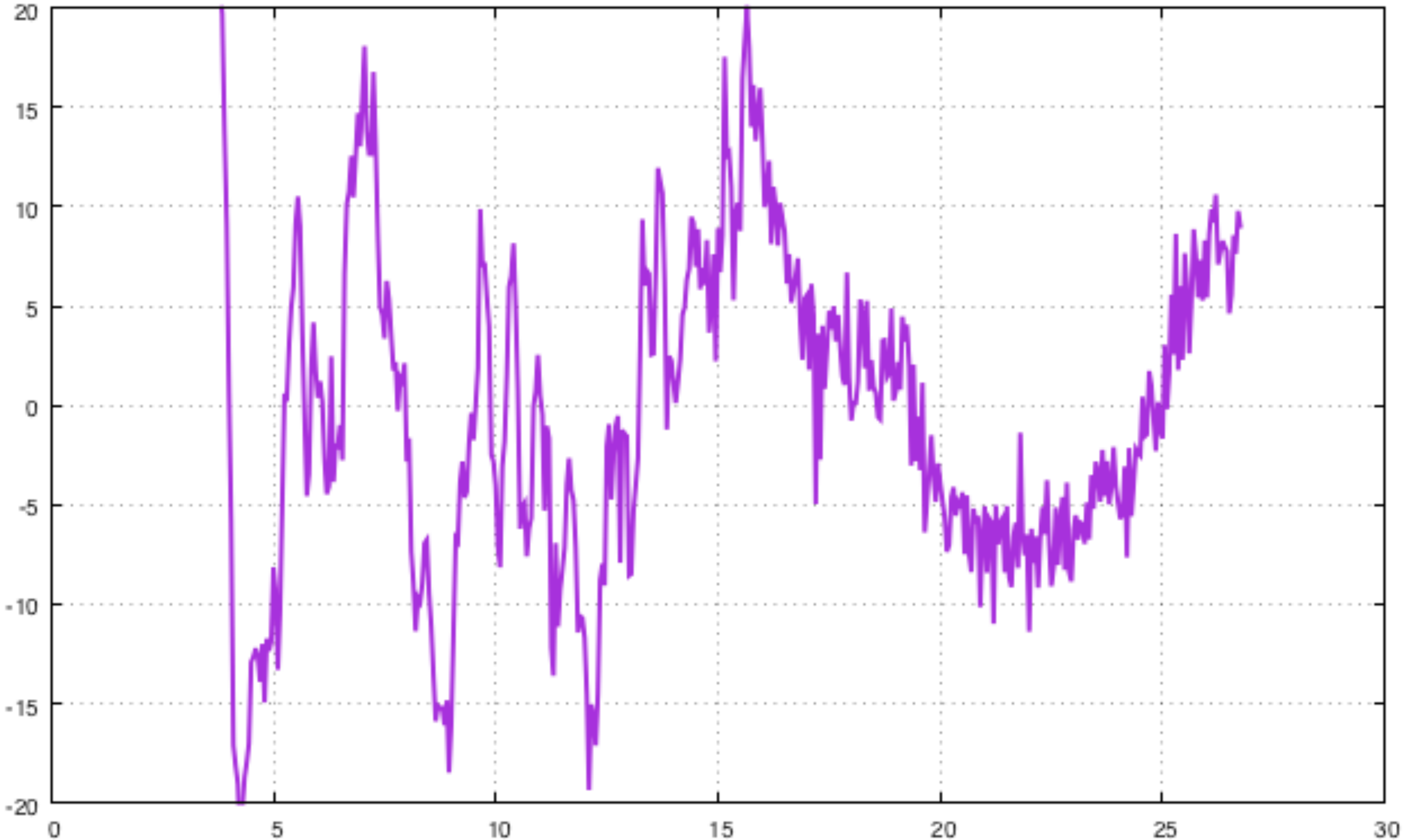
Q2 - Q1



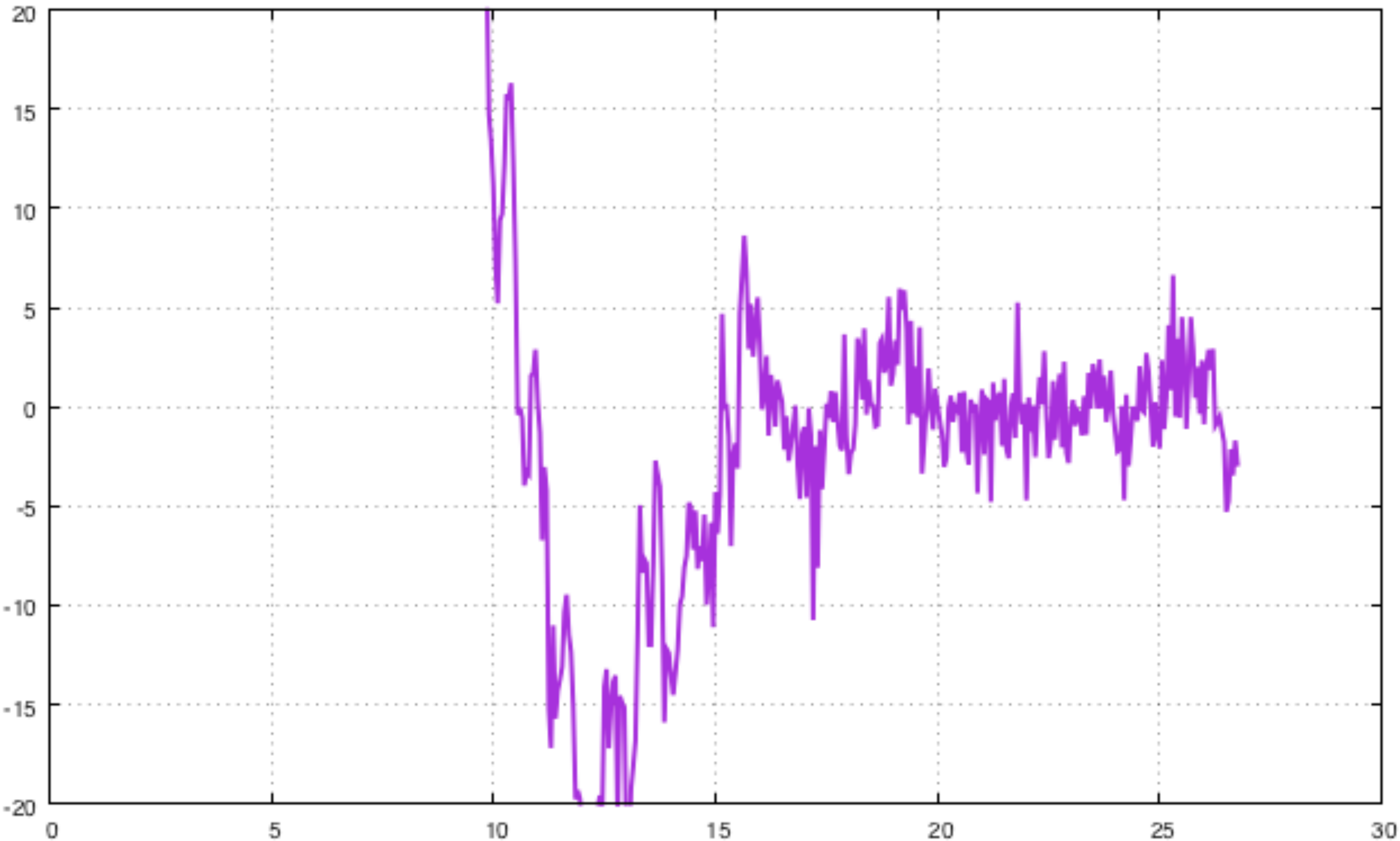
Log-Cos Fit to Quantum 2 ($t > 5$)



Residuals



Residuals for Fit of $t > 15$



Parameter Values

$t > 15$ fit

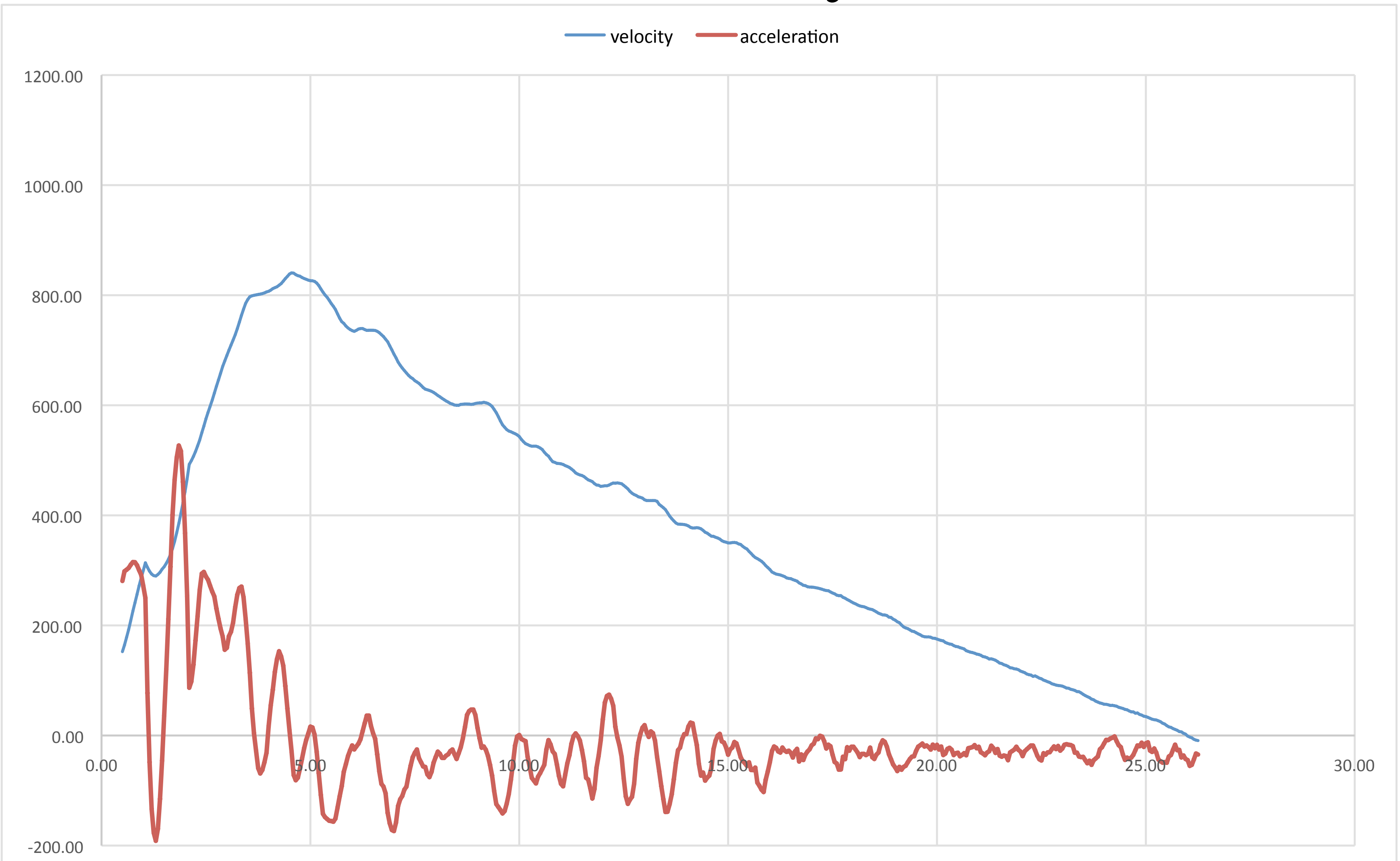
```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
amax            = 10050.1          +/- 0.4612       (0.004589%)
tmax            = 26.1366          +/- 0.0115       (0.04399%)
decel           = 27.9232          +/- 0.1033       (0.3699%)
tdrag           = 21.2891          +/- 0.4431       (2.081%)
```

$t > 5$ fit

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
amax            = 10043.7          +/- 1.014        (0.01009%)
tmax            = 25.9002          +/- 0.01196      (0.04619%)
decel           = 29.776           +/- 0.05792      (0.1945%)
tdrag           = 25.9245          +/- 0.0886       (0.3418%)
```

Excel Analysis of Velocity & Acceleration

Fit a parabola to +/– 0.5 seconds around each point.
Acceleration from t^2 term, velocity from t term

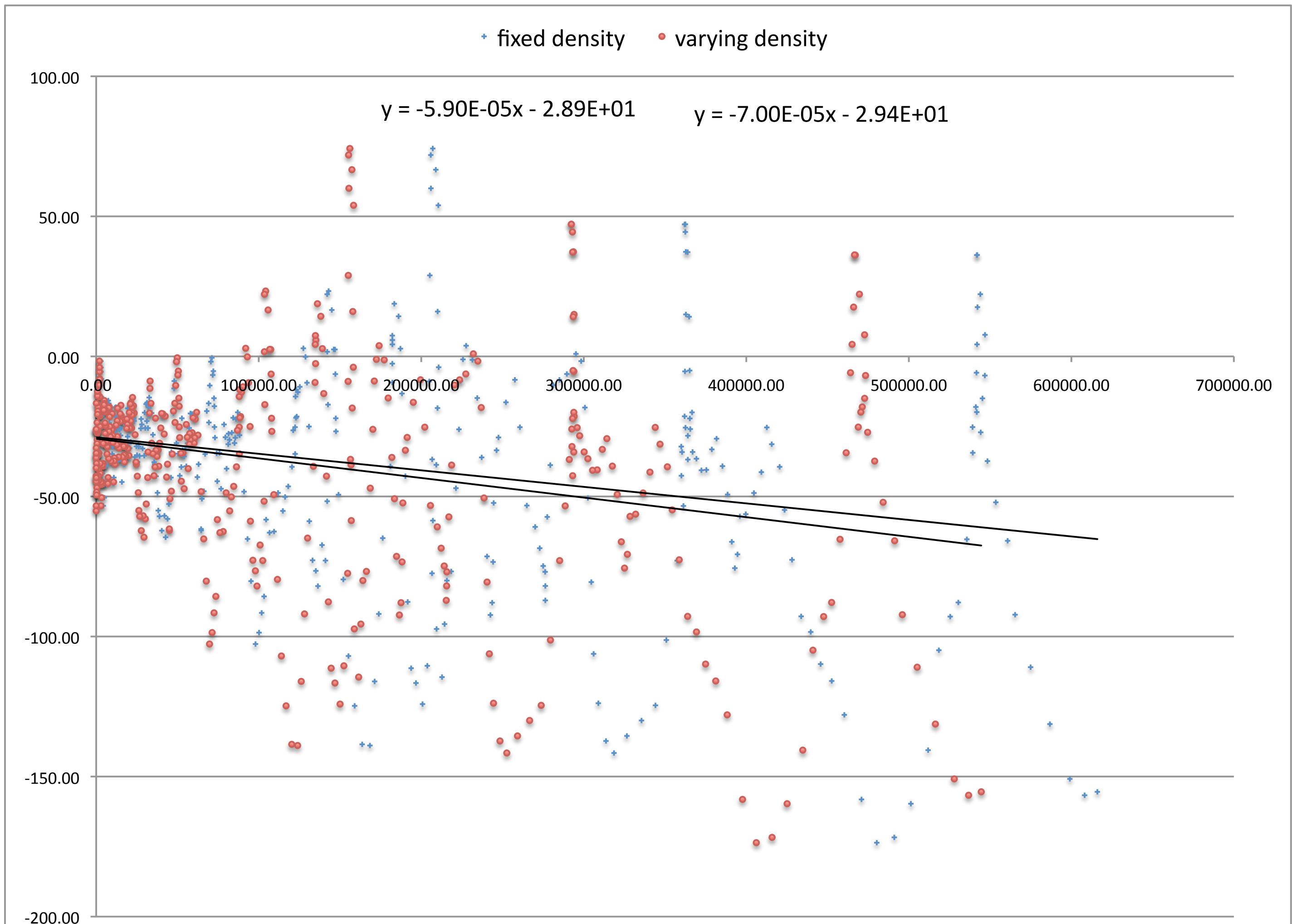


Altitude Dependence of Drag?

Acceleration after burnout should be $-\left(g + c\rho v^2\right)$

Plot accel vs v^2 and against $e^{-y/\lambda} \cdot v^2$

Data too noisy for any conclusion

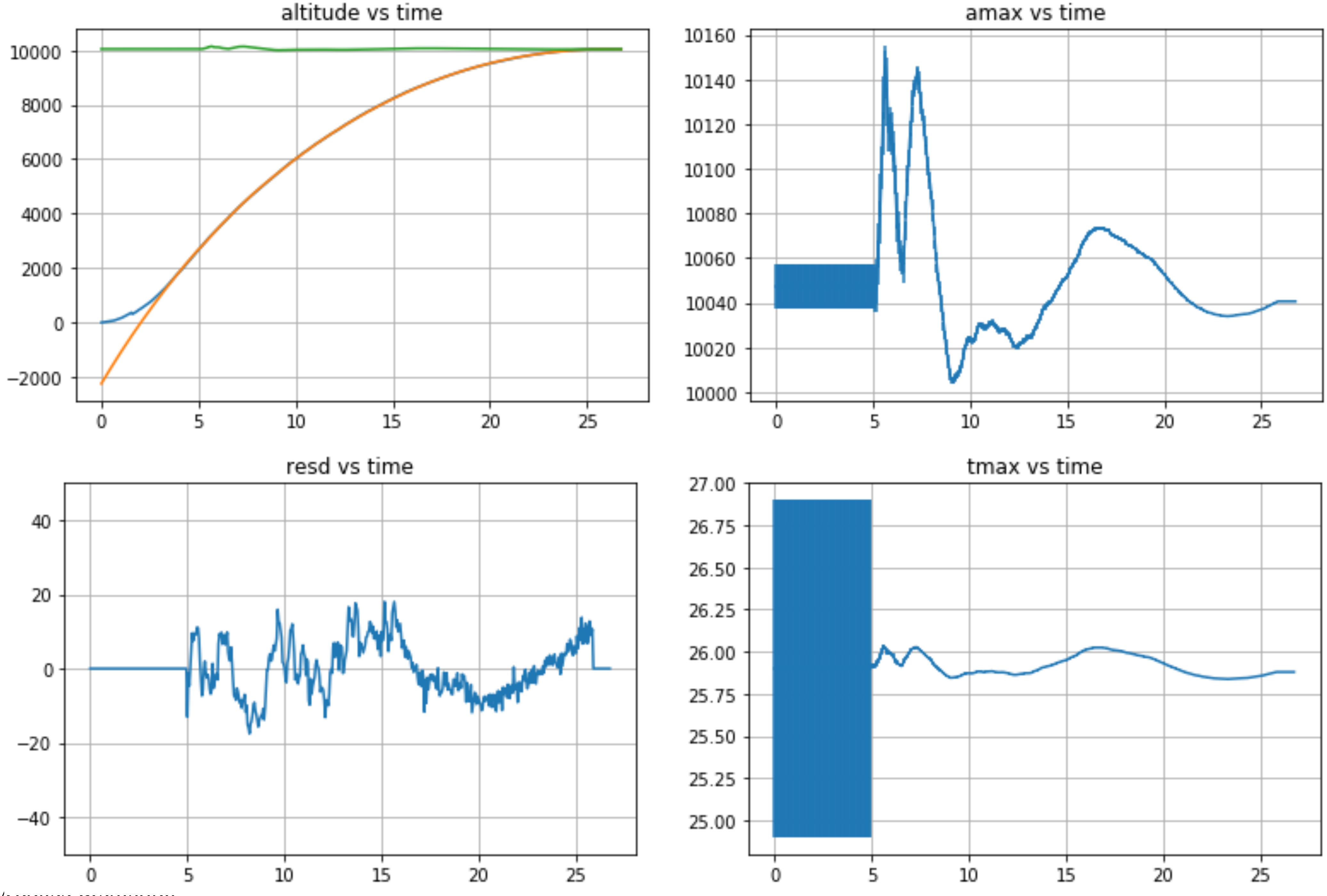


Kalman Filter

Progressive fit to data stream for continuous apogee-prediction, usable for apogee-feedback.

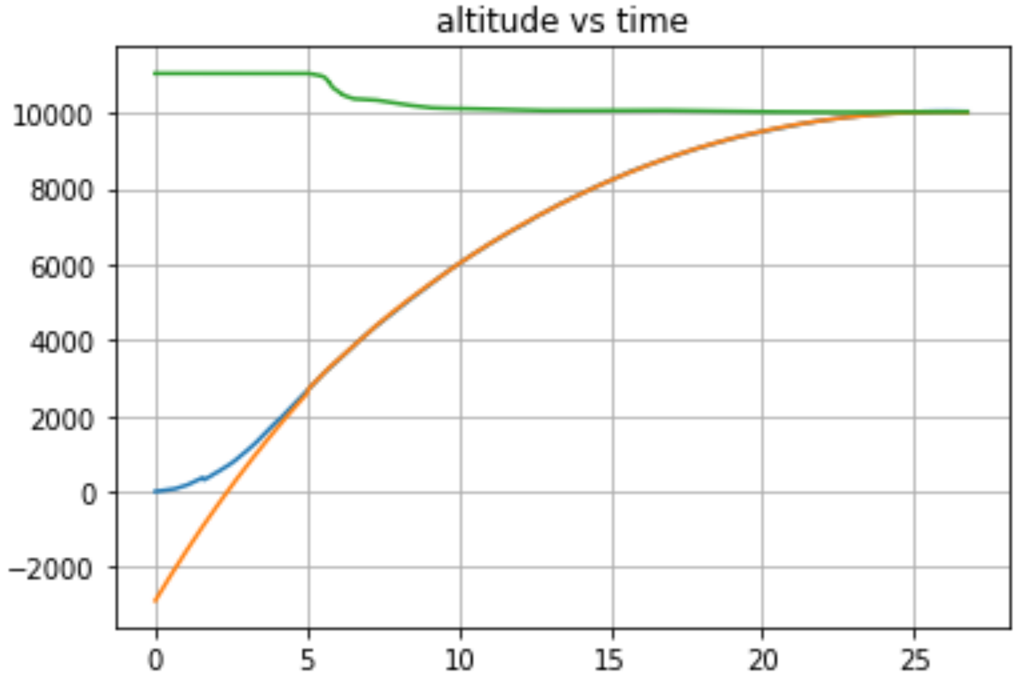
Code written (Python), seems be working.

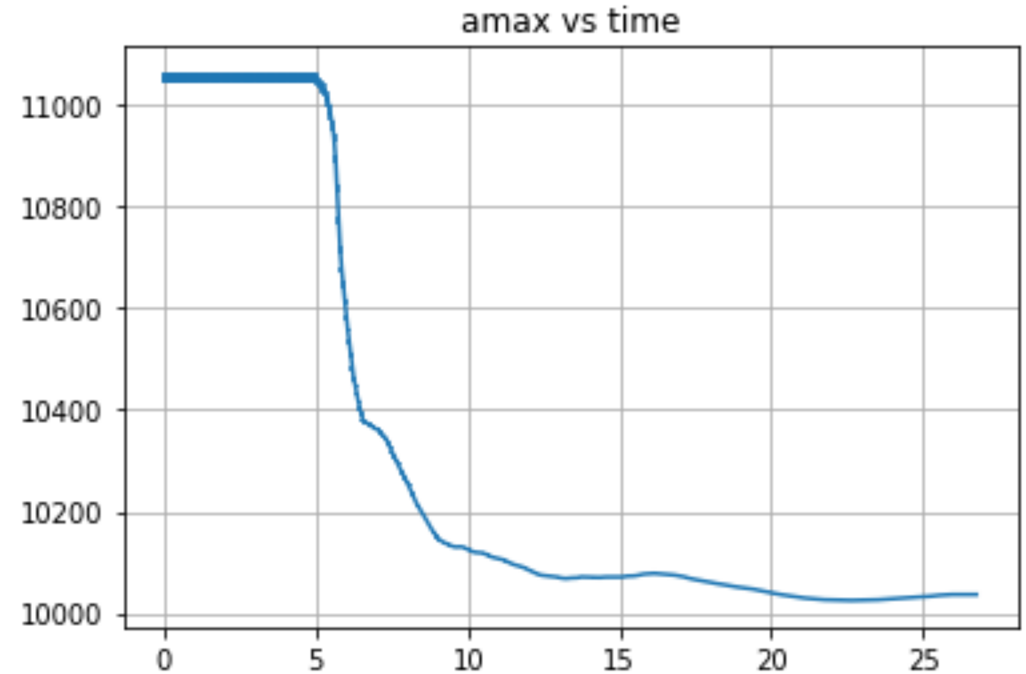
Start with Final Parameters

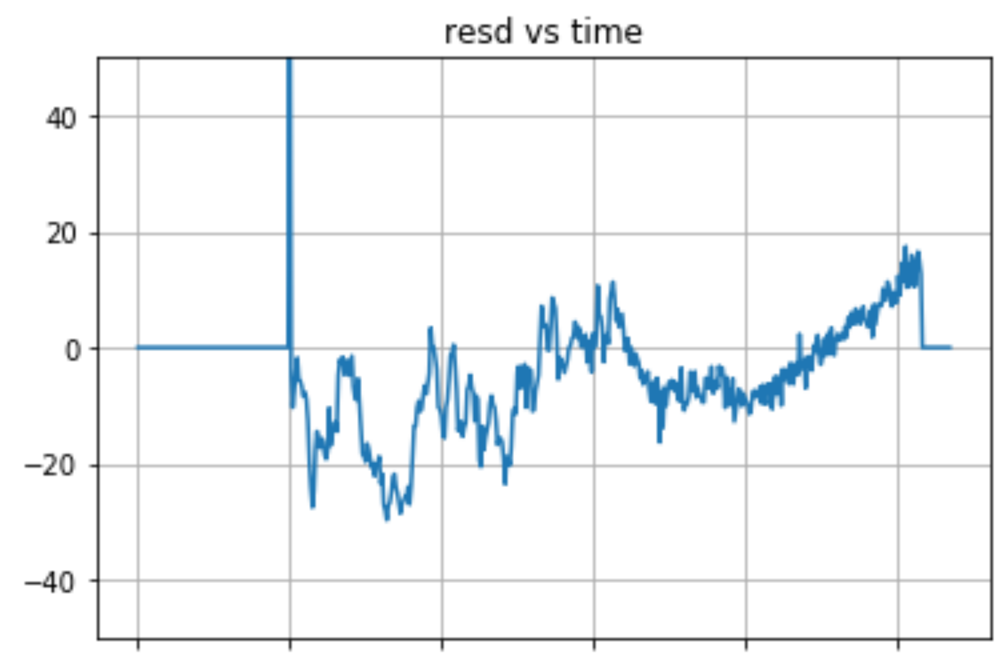


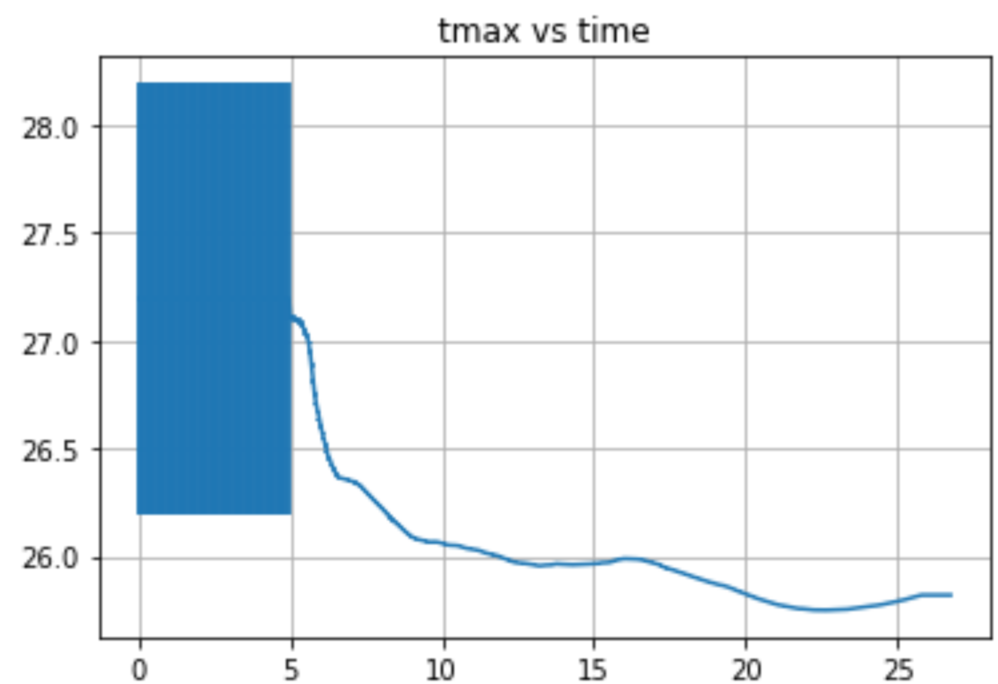
Apogee Prediction

Start with 1.1 Times Apogee

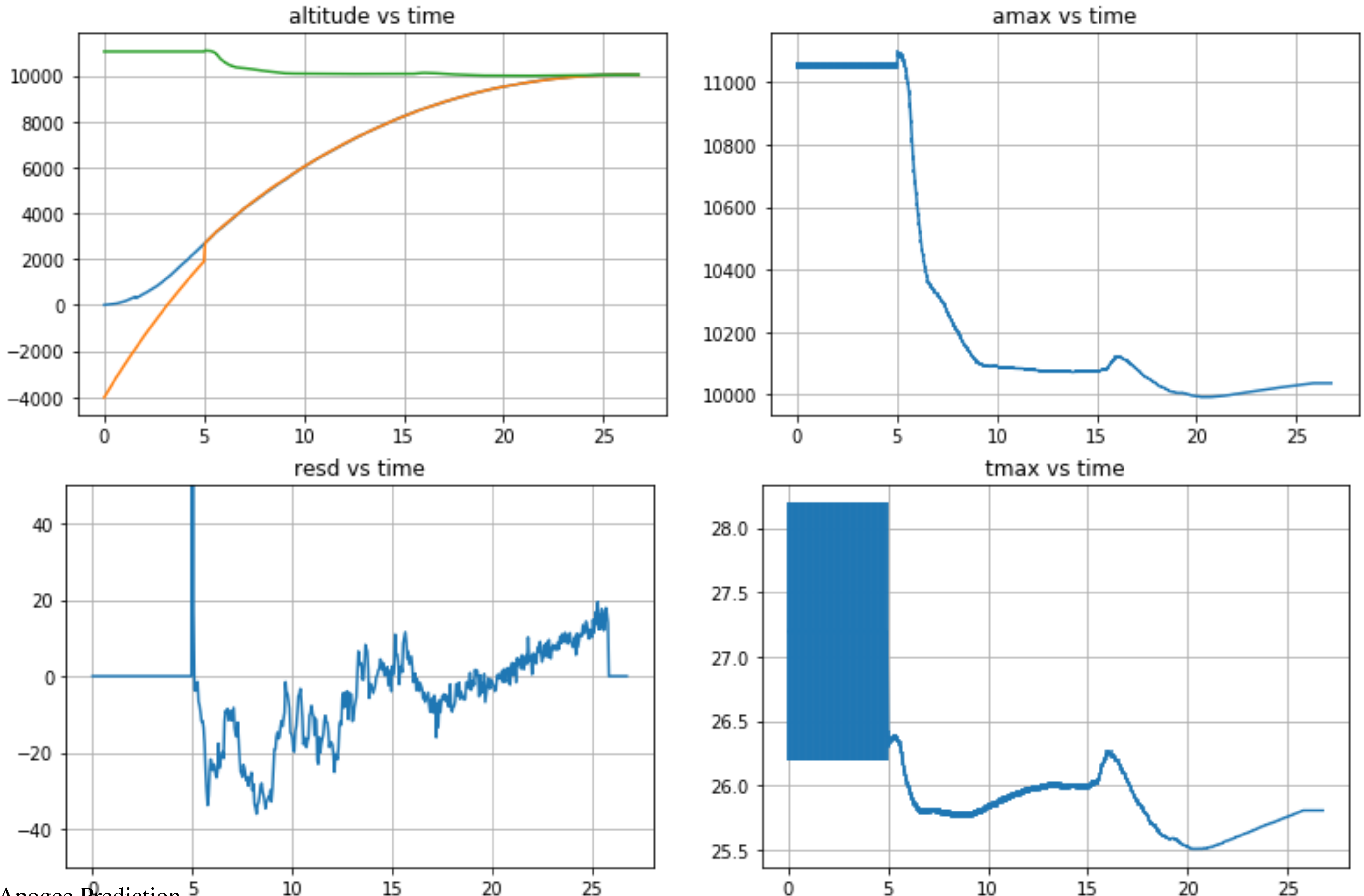


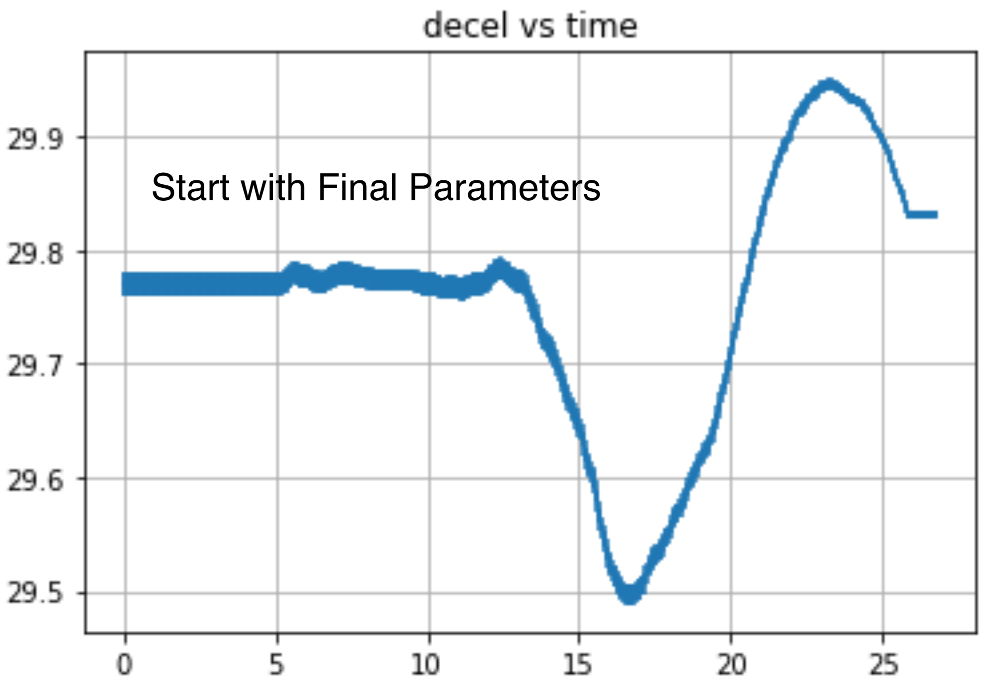


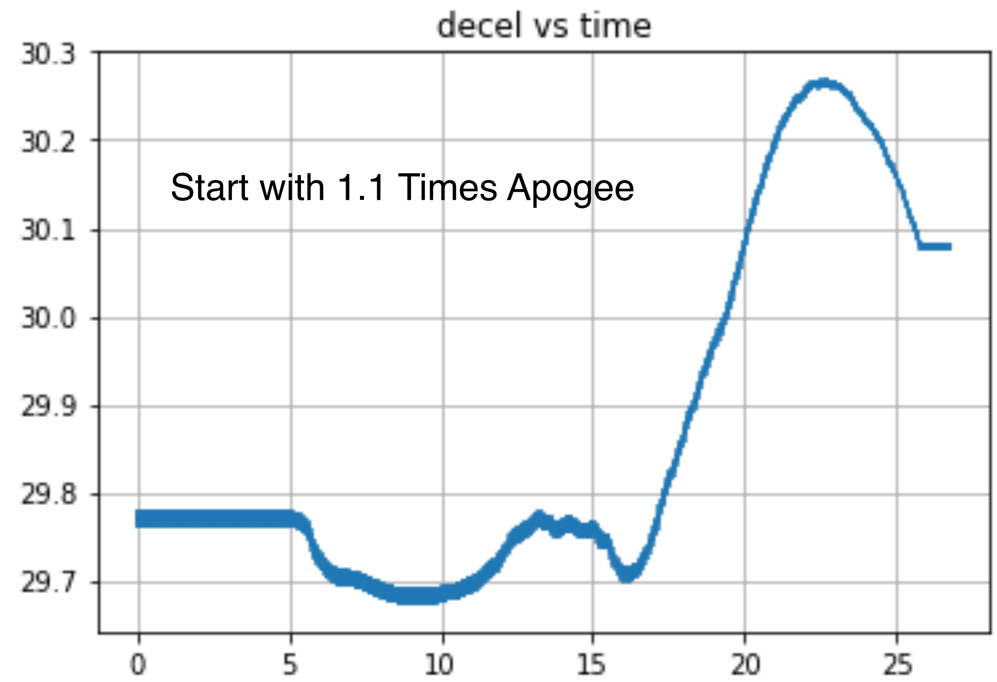


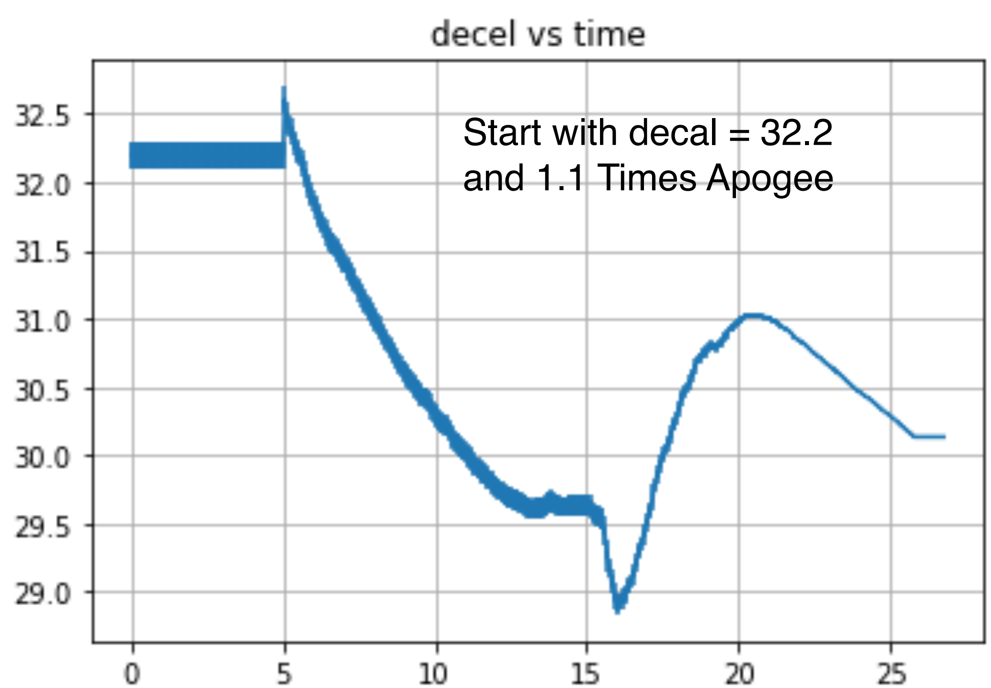


Start with g = 32.2 and 1.1 Times Apogee









Apogee Prediction