Code Details

This python code is an attempt to combine various aspects of the Eagleworks test stand used to evaluate their EM drive prototype.

A major contention is the use of the superposition of two signals: thermal noise and impulse force. This was outlined in their Figure 5:

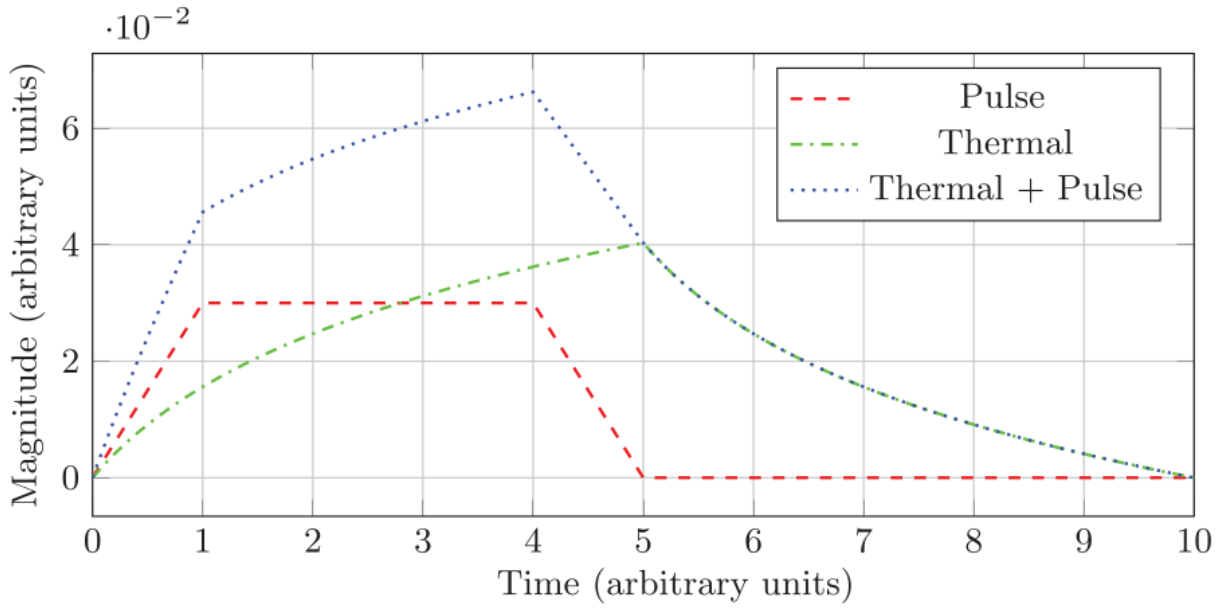

Fig. 5 Superposition of signals: conceptual superposition of an impulsive thrust (red) and thermal drift (green) signal over an on/off power cycle on the torsion pendulum.

There was some difficulty in modeling this response since there was no data presented of what the thermal portion of the curve was or the expected rise/fall times of the modeled signals (note the arbitrary time units in Fig. 5). With only the composite result presented it was difficult to work backwards to recreate the signals.

The calibration pulses were used in an attempt to measure any linear jump in forces against the nominal position of the torsion balance. There was also a thermal signal, an assumed impulse force generated by the EM Drive and some noise, which was unquantified in their experiement.

To replicate these calculations and methods, Figure 8. and the corresponding text was carefully analyzed.

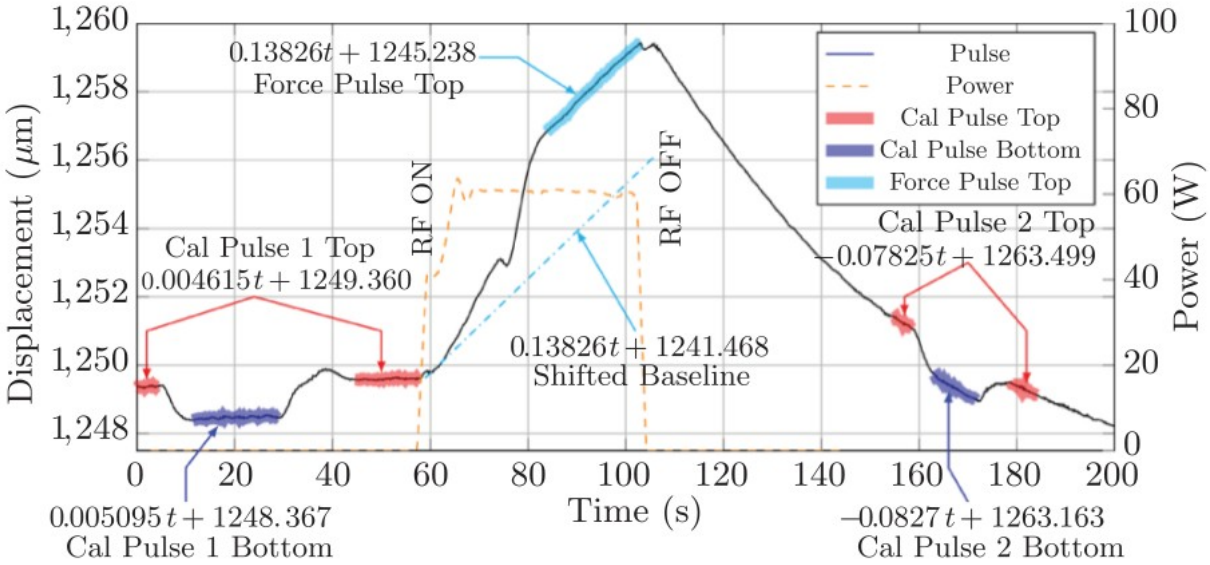

Fig. 8 Force measurement procedure plot: the figure shows one of the 60 W forward thrust runs with the data annotated to indicate the sections used to determine the calibration pulse characteristics and the force pulse characteristics (Cal, calibration).

The resulting simulated signals are shown below with the assumed impulse force, 2 calibration pulses, thermal profile and the total combination of these signals (with some Gaussian noise) scaled to match the values shown in Figure 8.

Thermal Profile Curve Fit

The included EW-data.ods in the repository shows some of the supporting data and formulas used to generate the curve fit and to look at their data quality of Table 1 of the report. Graph paper was laid over Figure 8 to record the rise and fall shapes.

This is the cooling portion of the curve (see the spread sheet EW-data.ods for details).

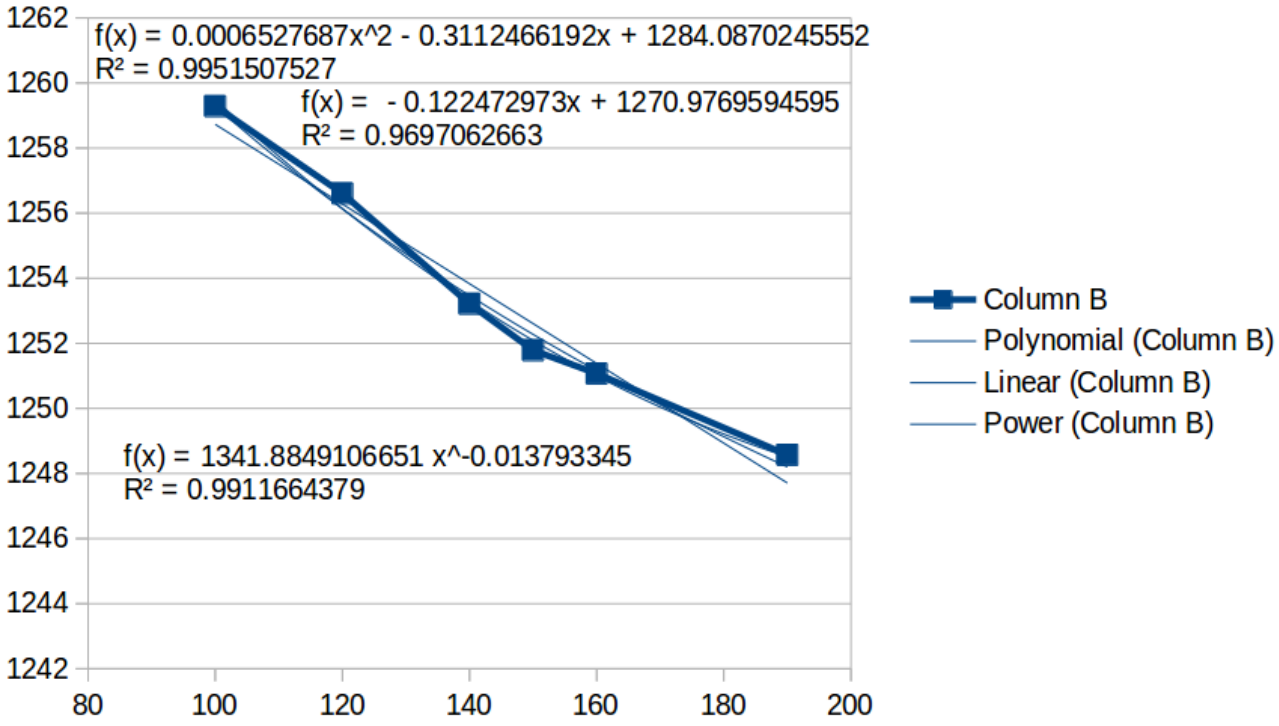

FIGURE A: Thermal Cooling Curve Fit

Since the exponential fit produced the best results, it was used in the code. However in the code you'll notice that there is a curve fit method that is commented out. This was the first pass attempt at just emulating the curve shapes, which worked fairly well, but this curve fit produced results that were closer to Eagleworks' data.

The thermal rise was more complicated because it theoretically should also include the rise due to EM drive force.

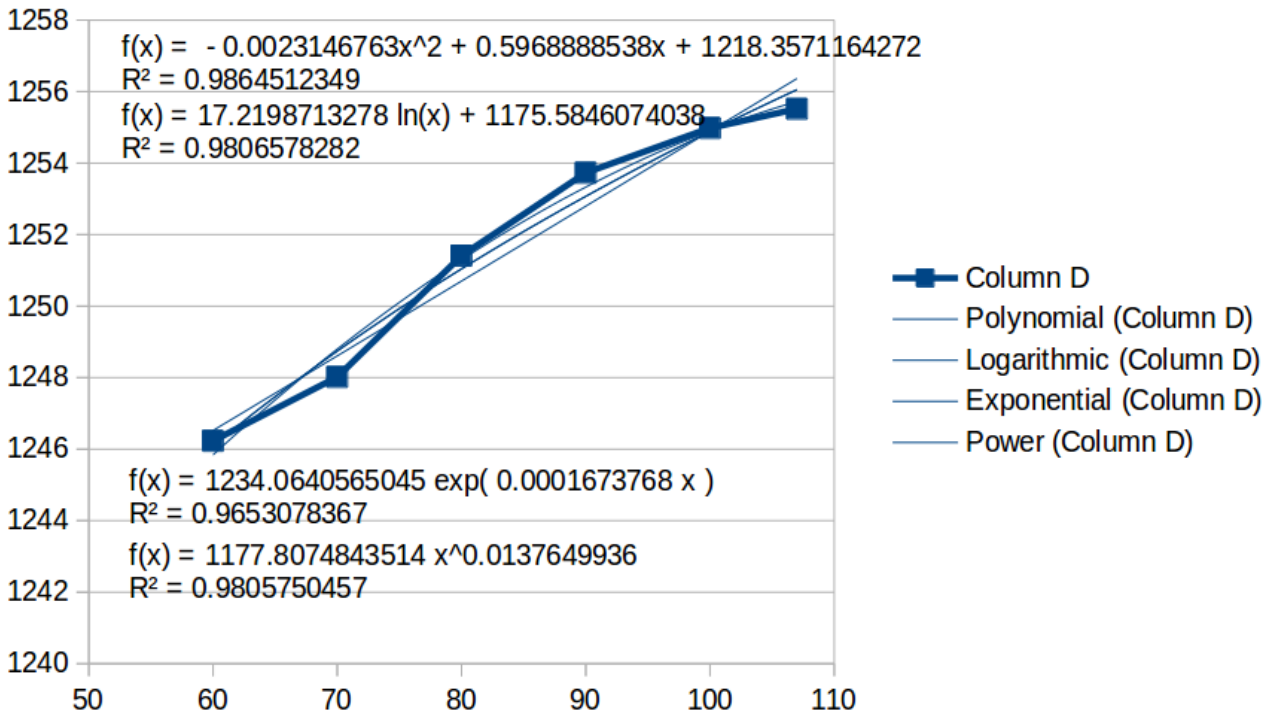

FIGURE B: Thermal Heating Curve Fit

The equation used for the rise was the logarithmic curve with r^2 of 0.9807. When computing the curve fit, I tried removing the 3.77 uM offset due to the pulse. This turned out to be a bad idea because it made the thermal signal discontinuous, so to correct for that I just put the "offset" variable back into the equation as a parameter that could be passed dynamically.

*value = 17.2198713278 * numpy.log(t) + 1175.584607 - 1249.360 + offset*

In order to superimpose these signals the intercept value of 1249.360 was subtracted from this curve to match the nominal offset from Figure. 8. Once all the signals are combined, this 1249.360 is added back into the baseline of the total waveform. This makes it mathematically easier to combine the two calibration pulses because they have a value of 0 when they are off.

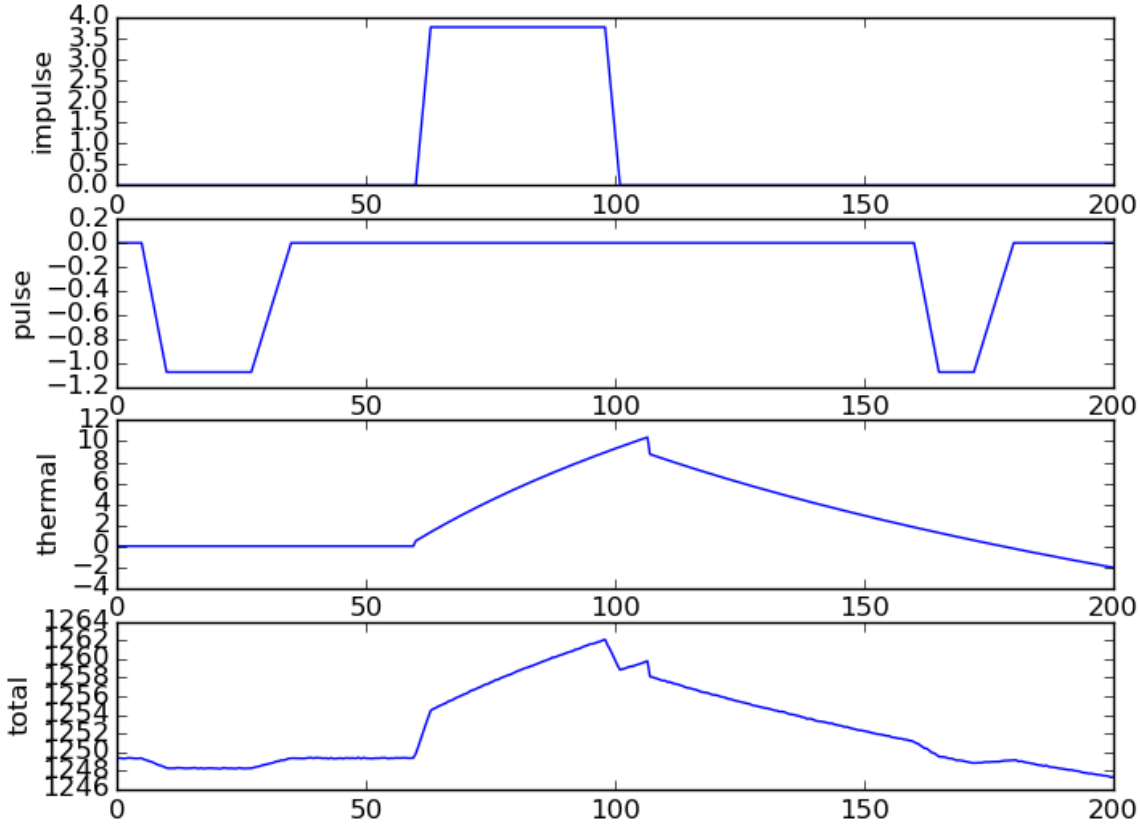

FIGURE C: Signals used for simulation

The same linear regression techniques were used to establish estimates and they are shown on the following diagram with each of their segments highlighted in a different color.

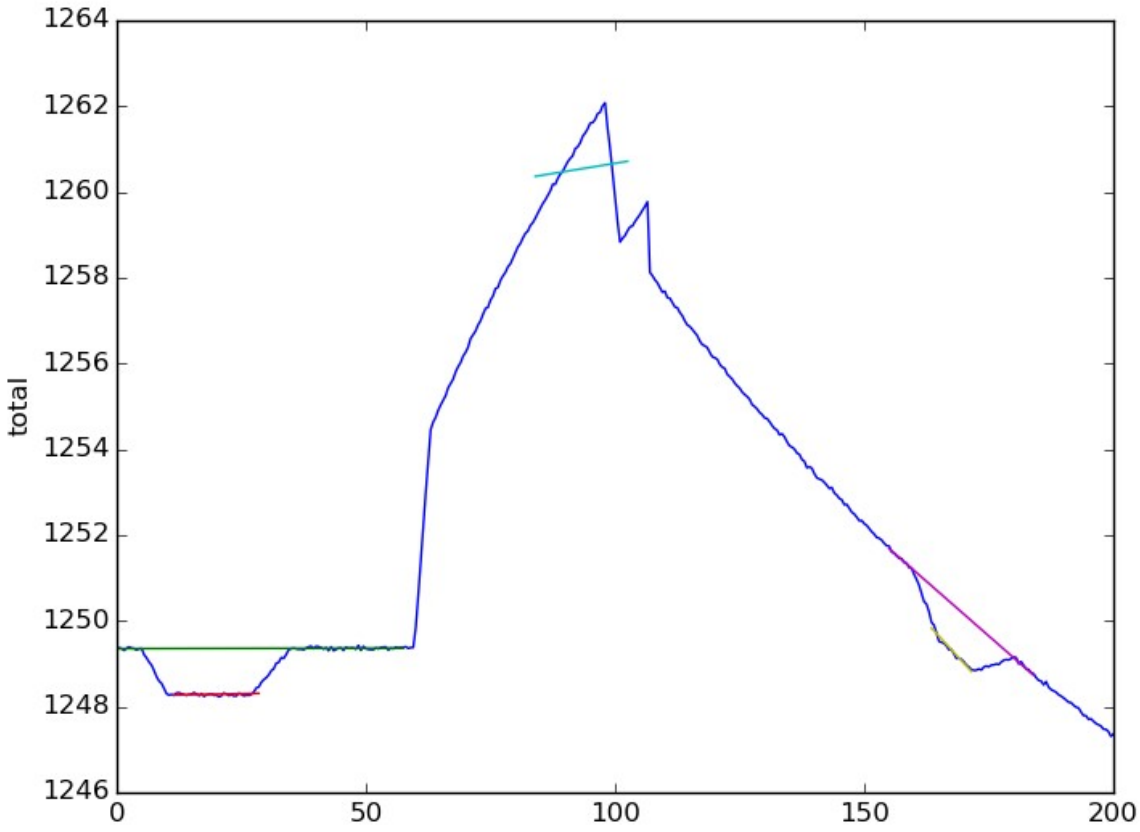

FIGURE D: Analysis of composite signal with linear estimates plotted

The resulting curve fits and force calculations follow:

Pulse 1 Top
m= 0.000199281720898 b= 1249.35454542 r= 0.153395238783 p= 0.378987350365 stderr= 0.000223474536704
Eagleworks: m= 0.004615 b=1249.360

Pulse 1 Bottom
m= 0.00233965835768 b= 1248.24534463 r= 0.252523511584 p= 0.143321722911 stderr= 0.00156057759392
Eagleworks: m= 0.005096 b=1248.367

Pulse 2 Top
m= -0.101838143564 b= 1267.4704713 r= -0.999230888042 p= 2.33817994882e-25 stderr= 0.000969273682573
Eagleworks: m= -0.07825 b=1263.499

Pulse 2 Bottom
m= -0.126786382148 b= 1270.56541839 r= -0.977859188505 p= 1.33548443768e-11 stderr= 0.0070055963548
Eagleworks: m= -0.0827 b=1263.163

Pulse Force
m= 0.0189533282626 b= 1258.77433421 r= 0.111116803606 p= 0.50659437746 stderr= 0.028252488058
Eagleworks: m=0.13826 b=1245.238

CAL1 Pulse Separation:
1.06596518439 um or 31.4475995618 uN force
CAL2 Pulse Separation:
1.07140875332 um or 31.6081931521 uN force
Impulse Force Calculations:
9.41978879458 um or -277.89814368 uN force

The estimate impulse force (time 83.8 to 102.8 minutes) is way off from the expected 106uN and in the wrong direction. This was calculated using the Pulse 1 intercept of b= 1249.35454542 and pulse force intercept of b= 1258.77433421, then using their scale factor of dx/df = 0.0338965517 to get ~ -277.9 uN. Whereas, Eagleworks used 1249.360 (cal1 top intercept) and 1245.238 (pulse intercept). The impulse force curve fit can be improved in simulation by adjusting the time window, however this was not done in order to compare the models as accurately as possible. See the section regarding test2.py with a better curve fit.

test2.py – improving curve fit for pulse force estimate

Since the computed signal curves don't exactly match the Eagleworks data, test2.py was adjusted slightly in just one parameter:

- test2 computes the slope of the impulse force with a slightly narrower window of 83.8 to 95 instead of 83.8 to 102.8 as described in Eagleworks test
- the only modified line of code is:
  - f_pulse = Calc(times, total, 83.8, 95) # adjusted pulse time window

Using a slightly smaller window produces a much better fit. Compare the light blue or teal line in Figure B with the same line here in Figure E below:

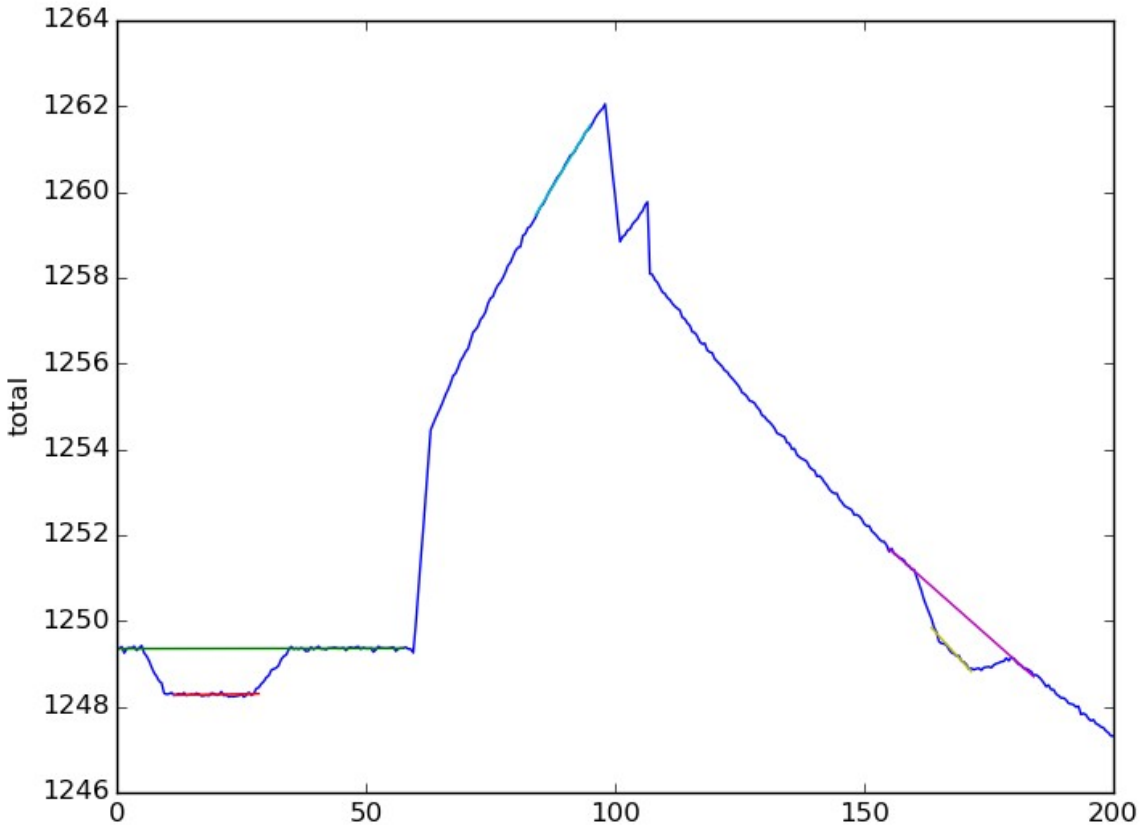

FIGURE E: Reduced calculation window for force pulse at 83.8 to 95 minutes. Fit to curve is so close that it is hard to identify in this plot.

The numerical results are the same in test1.py as shown earlier, except for the pulse force curve fit which is much closer to Eagleworks estimates:

Pulse Force
m= 0.191644024747 b= 1243.35701241 r= 0.998854473792 p= 3.2649144987e-29 stderr= 0.00200344082639
Eagleworks: m=0.13826 b=1245.238

CAL1 Pulse Separation:
1.06902614289 um or 31.5379025085 uN force

CAL2 Pulse Separation:
1.05530046189 um or 31.1329739742 uN force

Impulse Force Calculations:
-5.99602967517 um or 176.892025131 uN force

This also shows that the test1.py curve fit produced a number that resulted in negative force which is corrected with this better curve fit. However the value is sill off significantly from the 106 uN estimate and you'll notice that the slope of 0.1916 vs Eagleworks of 0.13826 is quite different. This implies that perhaps a pulse shape is not the best superposition solution.

No Additional Impulse Force Signal

As an additional experiment, the added impulse force was set to 0 to see if 106uN would be extracted from the curve fit data.

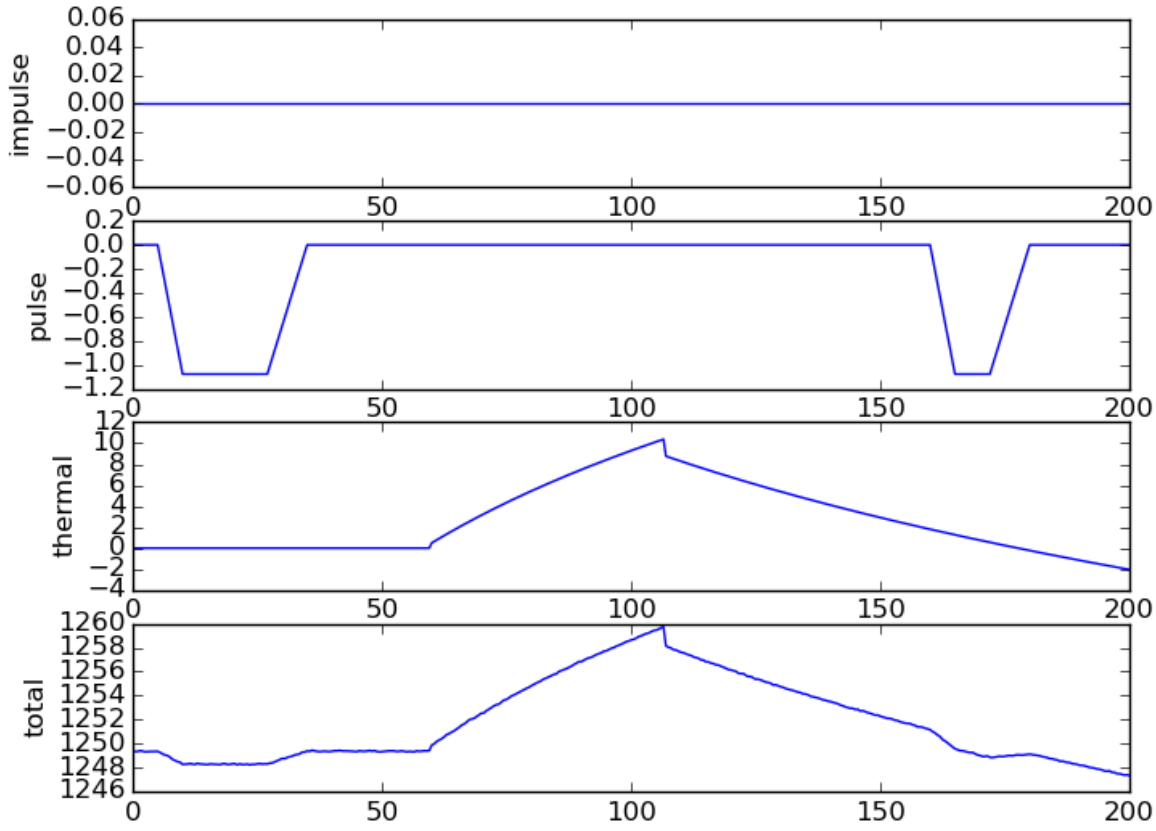

FIGURE F: No Impulse Force was added

This produced the following curve fits (using test2.py time window for the pulse):

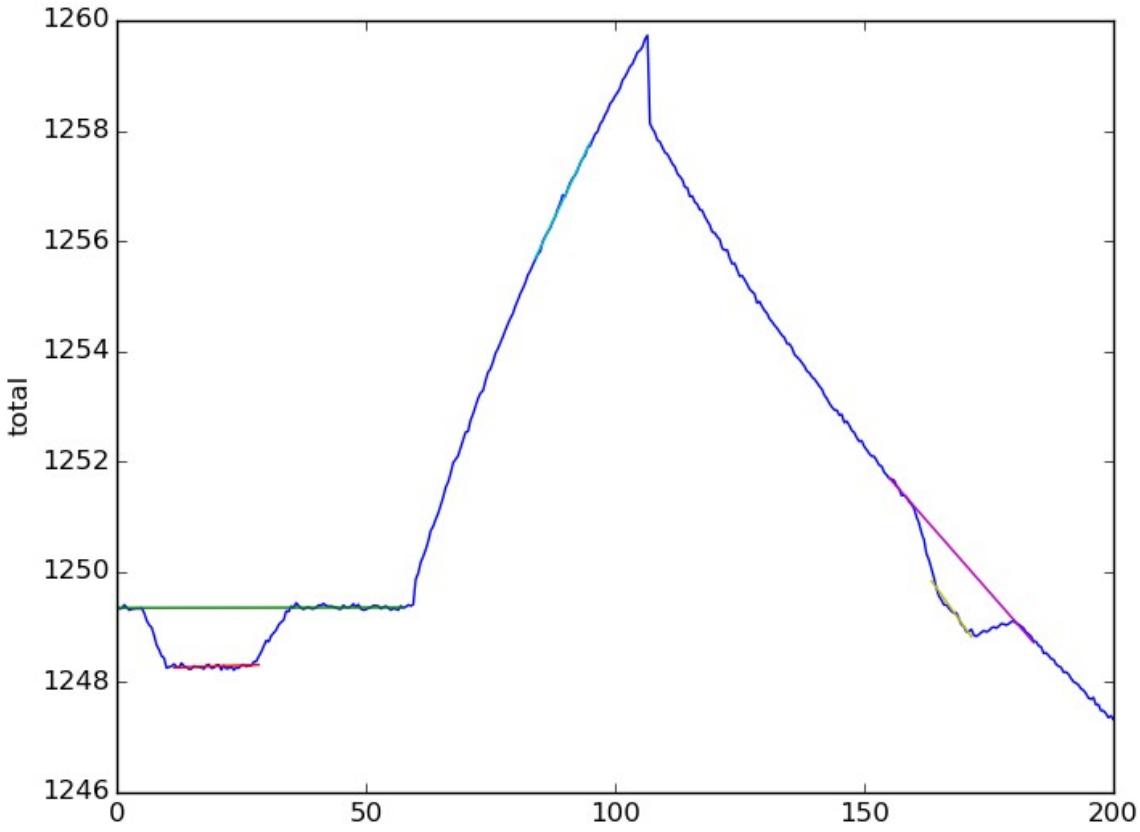

FIGURE G: No addition impulse force was simulated

The numerical results show:

Pulse 1 Top
m= 0.000139664241161 b= 1249.34771452 r= 0.0998534105271 p= 0.568192094514 stderr= 0.000242264274863
Eagleworks: m= 0.004615 b=1249.360

Pulse 1 Bottom
m= 0.00314886248176 b= 1248.22728828 r= 0.330696406164 p= 0.0523360320335 stderr= 0.00156429378437
Eagleworks: m= 0.005096 b=1248.367

Pulse 2 Top
m= -0.102958674697 b= 1267.66474758 r= -0.99903531818 p= 1.60304738336e-24 stderr= 0.00109763999145
Eagleworks: m= -0.07825 b=1263.499

Pulse 2 Bottom
m= -0.126748857786 b= 1270.56259013 r= -0.964577154843 p= 4.36089660093e-10 stderr= 0.00895030258973
Eagleworks: m= -0.0827 b=1263.163

Pulse Force
m= 0.190398344863 *b= 1239.69625897* r= 0.998373721019 p= 1.29110324666e-27 stderr= 0.00237244558335
Eagleworks: m=0.13826 b=1245.238

compare with nominal impulse force added (test2.py):
m= 0.191644024747 b= 1243.35701241 r= 0.998854473792 p= 3.2649144987e-29 stderr= 0.00200344082639

CAL1 Pulse Separation:
1.05964044265 um or 31.2610100291 uN force

CAL2 Pulse Separation:
1.07511803014 um or 31.7176224783 uN force

Impulse Force Calculations:
9.65145555754 um or 284.732666702 uN force

So the impulse force ended up higher because of the intercept value is so different in the curve fits. Obviously there is a problem somewhere in here, either the model, the simulation or the method being used.

Problems:

- Unable to duplicate Eagleworks calculations with this simulation.
- There seems to be little correlation between the “calculated impulse force” and the modeled impulse force, raising concern on the method used in general.
- The thermal curve fit essentially includes the theoretically hidden force pulse signal. The code then adds another pulse of about the same magnitude on top of it, then computes the results. 177 uN (test2.py) and being generous you could just divide by two and say 88.2 uN, vs 106 uN with Eagleworks.
- The curve fit for the force pulse is the biggest source of error. This can be changed if the time window is modified some, however for this example the numbers were kept the same as reported in Eagleworks paper.
- Thermal curve has a problem with the a jump in it when switching directions due to curve fitting one formula to another formula and being able to adjust the center on the time scale. For the most part this is excluded from any of the windows of calculations, but it is an illustration that the data fit is not perfect.
- The thermal + impulse superposition in Eagleworks data does not seem to be pulse like, but rather tapers to a lower value as the amplifier stays on. This could simply be a non-linear heating effect, but putting in a pulse model for the force as shown in Figure C produces a composite response that is much higher than what was measured by Eagleworks.
- Matching the data curves was done by hand laying their plots over fine graph paper. This is not an idea way to generate data for curve fitting and leaves room for errors.
- There was no “thermal only” profile measured by Eagleworks, so separating these signals mathematically is difficult.
- The Eagleworks report states different values for dx/df which makes calculations confusing:
  - From EW paper P. 4, the dx vs df is computed based on their statement:
    " 0.983 um, which corresponds with the calibration pulse magnitude of 29 uN"
    which means dx/df = 0.0338965517 m/N
  - On P.5 "two fitted linear equations is 1.078 um, which corresponds with the calibration pulse magnitude of 29 uN."
    which produces dx/df = 0.0371724137931 m/N
- Lack of noise data reported by Eagleworks required just visually estimating their noise response

Improvements

- Make thermal curve more continuous.

- Test against no impulse force added to see if curve fits Eagleworks estimations.
- Access to raw data would improve modeling estimates
- Various values of pulses and pulse shapes could be simulated to estimate the variation and reliability of this measurement technique
- Statistical bounds on accuracy can be established using a wide range of trial runs and simulated impulse signal values
- Fine tuning the numerical windows might provide slightly better approximations of what was shown in Figure 8, however they were strictly followed for this example in an attempt to duplicate their measurements.
- Thermal model needs some work to mimic their response, however without having a thermal only response to compare it too, this might never be possible.
- Obviously in Figure D vs Eaglework's Figure 8 there is a significant difference in the trailing edge of the pulse drop which changes the linear estimates drastically. As mentioned before the speculation is the lack of change in Figure 8 could be due to the “impulse” force lessening or the thermal heat starting to saturate producing less force before the amp is shut down.
- General code improvements are needed as well – it was just hacked together