# Building a complete calibration pipeline in the front end

Dane Stocks (SURF)

Joseph Betzwieser

LLO

# Outline

- Project overview/objective
- Approach
- Simulink diagrams of new model
- Comparisons with current model
- Obstacles, where we go from here

# Project overview

- For O1 and O2, Advanced LIGO produced calibrated GW strain using a system of disjoint calibration pipelines.
  1. Primary/Online system—CALCS partially calibrates data from the DARM loop and sends it to the GDS pipeline, where $h(t)$ is reconstructed through the use of models of $A$ and $C^{-1}$. This system runs in the front end and uses IIR filtering to produce strain time series. Latency is between 5 and 10 seconds.

  2. Redundant/Offline system—the DCS model is sent $d_{err}$ and $d_{ctrl}$ and reconstructs $h(t)$ using complete FIR filters designed from measurements of the $A$ and $C^{-1}$ transfer functions. Used for recalibration of data sets (posterity).

# Project overview cont.

- We are constructing a self-contained calibration pipeline that will be placed in the front end computers, producing calibrated $h(t)$ as a raw data product.
- It will use FIR filters, identical to the ones used in current DCS flow.
- When implemented, operators in the control room will have access to calibrated strain in extremely low latency and redundancy of current calibration scheme will be removed.

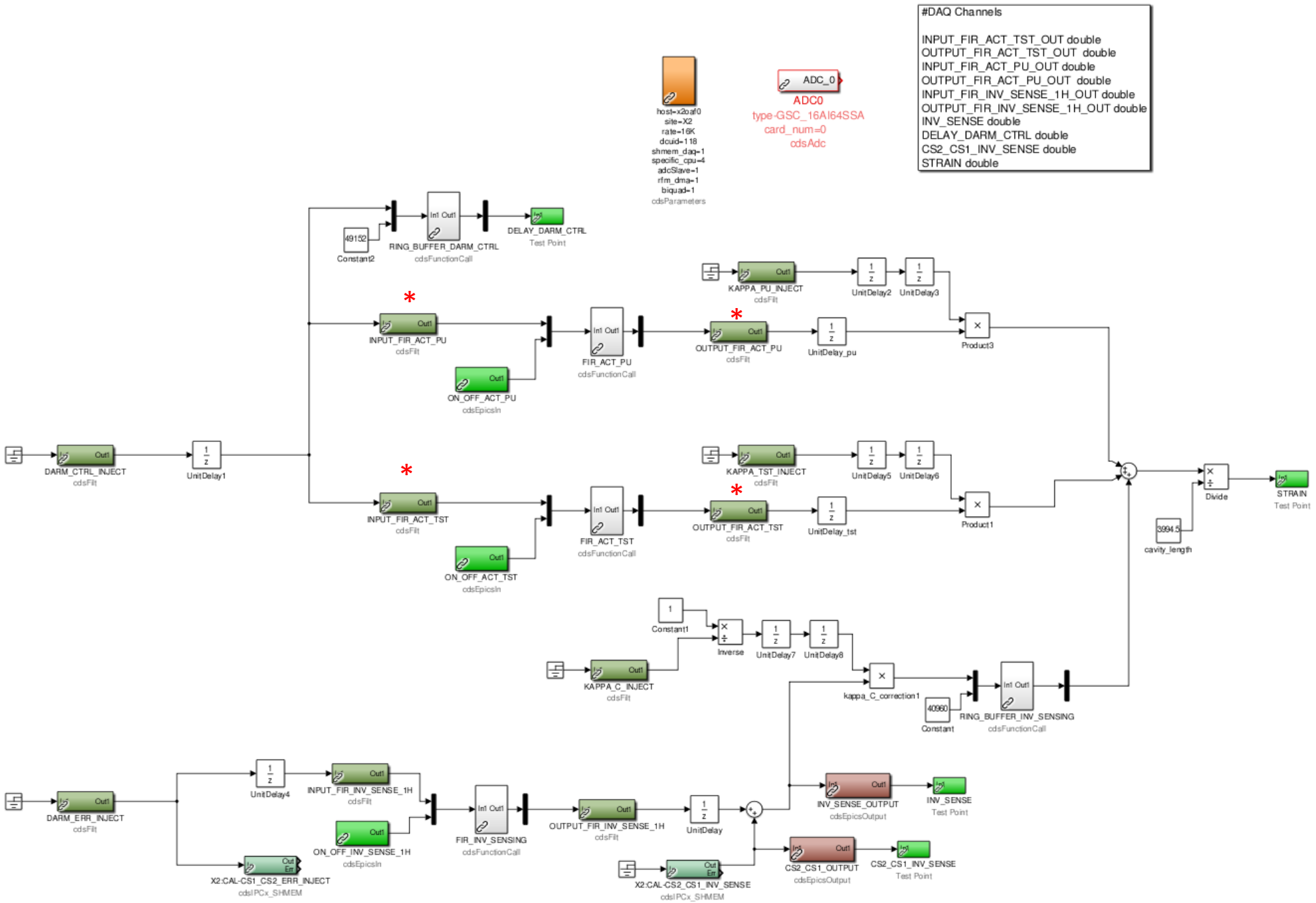
# Approach

- Began by writing C code to be placed in Simulink filter modules. We pull data from “L1DCS_1175961600.npz” and write the coefficients for each FIR filter into a C code block, which performs basic convolution of the input signal and filter.
- The inverse sensing filter (running at 16384 Hz) could not complete its computations in time, so the filter was split “across space,” and half of its computations are done in a separate model.

Filtering C code (inv sensing)

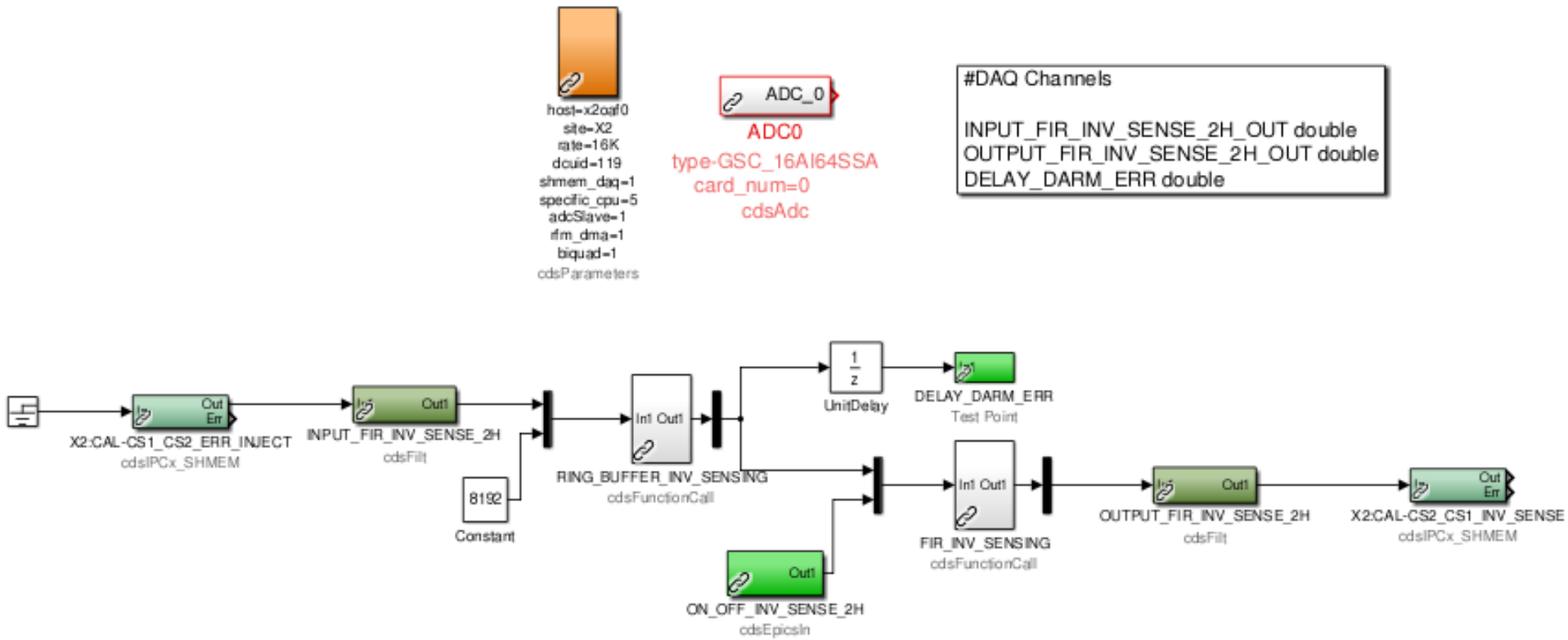
```
void user_fir(double *argin, int nargin, double *argout, int nargout){
    static double history[8192] = {0};

  static double output = 0;
  double *hist_ptr,*hist1_ptr;
  double input;
  double on_off;

  hist_ptr = &history[0];
  hist1_ptr = hist_ptr;

  input = argin[0];

  on_off = argin[1];

  if (on_off > 0) {
    output = 3.4650051394835747e-12 * (*hist_ptr++);
    *hist1_ptr++ = *hist_ptr;
    output += -4.4428923154350198e-13 * (*hist_ptr++); // 2
    *hist1_ptr++ = *hist_ptr;
    output += 9.0096896636020352e-12 * (*hist_ptr++); // 3
    *hist1_ptr++ = *hist_ptr;
    output += -1.2512562357161775e-11 * (*hist_ptr++); // 4
    *hist1_ptr++ = *hist_ptr;
    output += -3.6116262273449539e-12 * (*hist_ptr++); // 5
    *hist1_ptr++ = *hist_ptr;
    output += 1.0934443489968872e-11 * (*hist_ptr++); // 6
    *hist1_ptr++ = *hist_ptr;
    output += -1.248721480391032e-11 * (*hist_ptr++); // 7
    *hist1_ptr++ = *hist_ptr;
    output += 1.1217473100452248e-11 * (*hist_ptr++); // 8
    *hist1_ptr++ = *hist_ptr;
    output += -8.8692767713504332e-12 * (*hist_ptr++); // 9
    *hist1_ptr++ = *hist_ptr;
    output += 6.4218041837870596e-12 * (*hist_ptr++); // 10
    *hist1_ptr++ = *hist_ptr;
    output += -4.34120820850123e-12 * (*hist_ptr++); // 11
    *hist1_ptr++ = *hist_ptr;
    output += 2.801574528933253e-12 * (*hist_ptr++); // 12
    *hist1_ptr++ = *hist_ptr;
    output += -1.7713098376196786e-12 * (*hist_ptr++); // 13
    *hist1_ptr++ = *hist_ptr;
    output += 1.1460910827389522e-12 * (*hist_ptr++); // 14
    *hist1_ptr++ = *hist_ptr;
    output += -7.7747393268475827e-13 * (*hist_ptr++); // 15
    *hist1_ptr++ = *hist_ptr;
    output += 5.5553440128881088e-13 * (*hist_ptr++); // 16
    *hist1_ptr++ = *hist_ptr;
    output += -3.8976878744217215e-13 * (*hist_ptr++); // 17
    *hist1_ptr++ = *hist_ptr;
    output += 2.500141598319365e-13 * (*hist_ptr++); // 18
    *hist1_ptr++ = *hist_ptr;
    output += -1.167673434511809e-13 * (*hist_ptr++); // 19
    *hist1_ptr++ = *hist_ptr;
```

x2calcs1 diagram

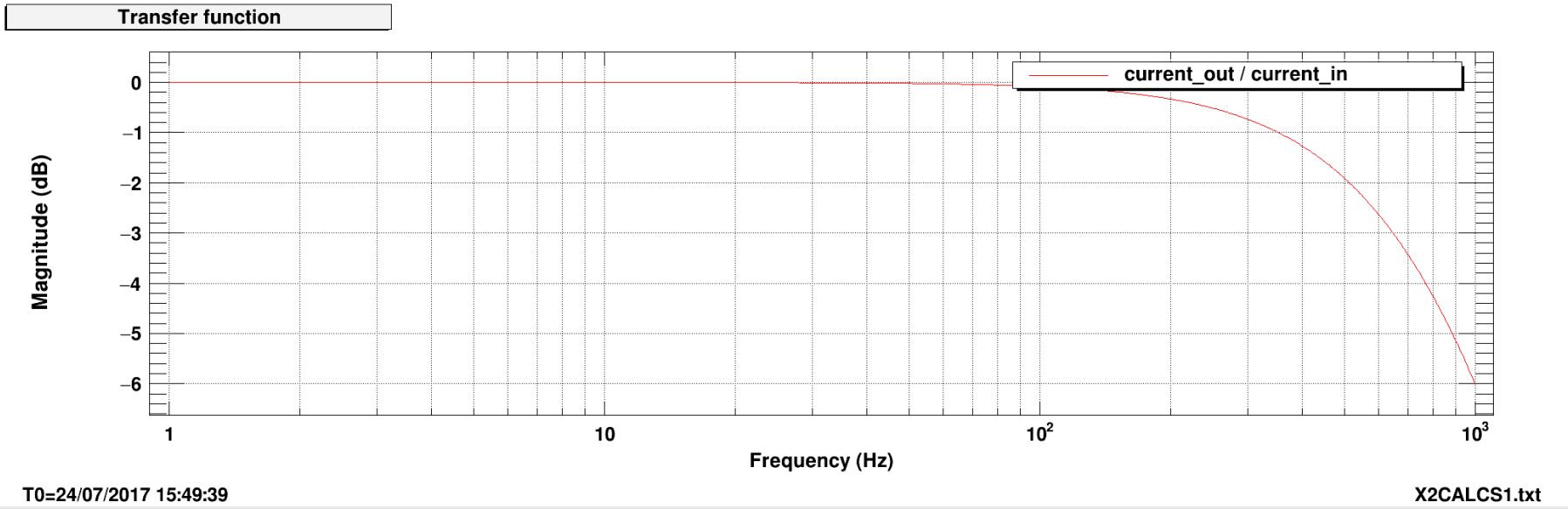


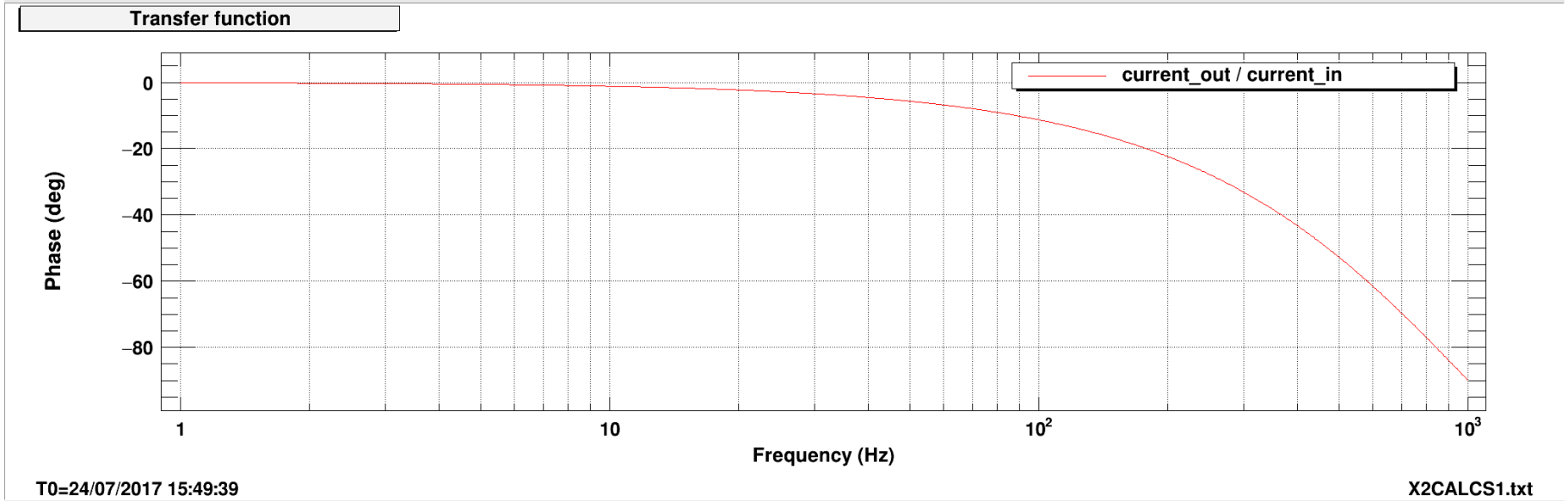

* act lowpass at 1000 Hz

# x2calcs2 diagram

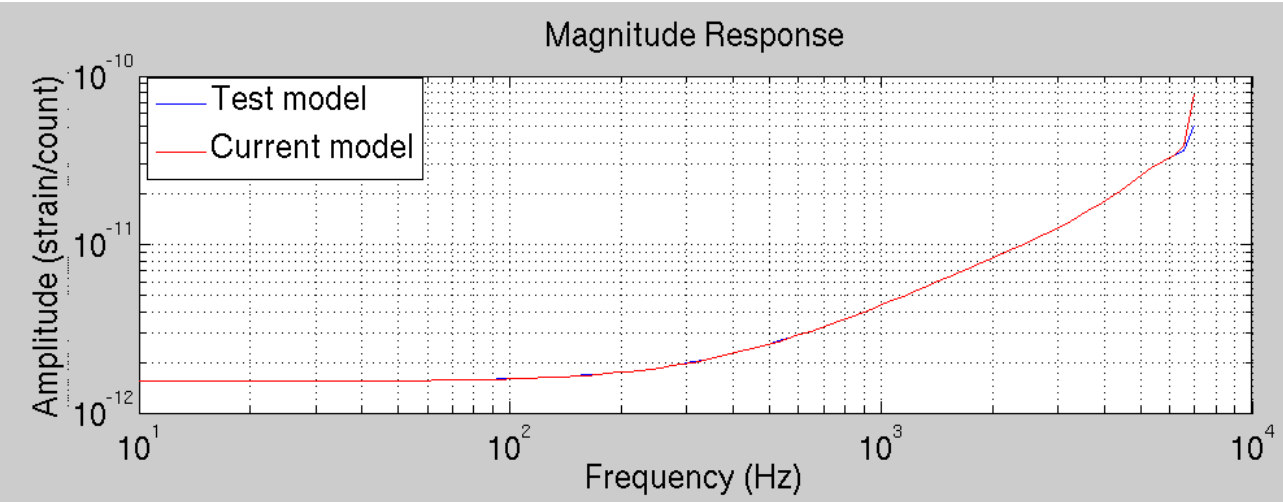


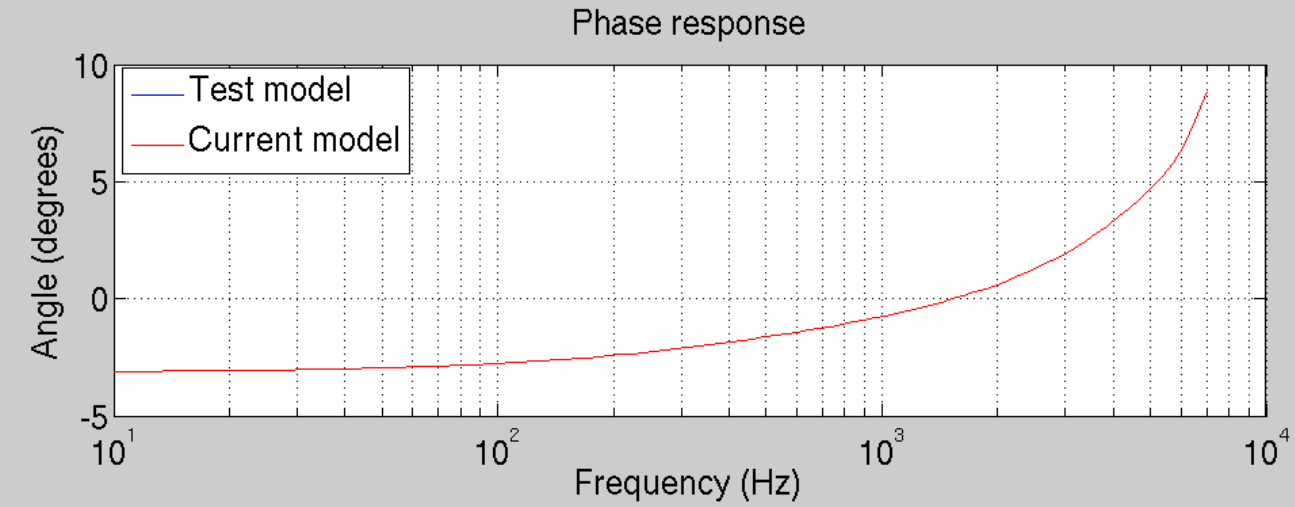
# 1000 Hz actuation lowpass filter

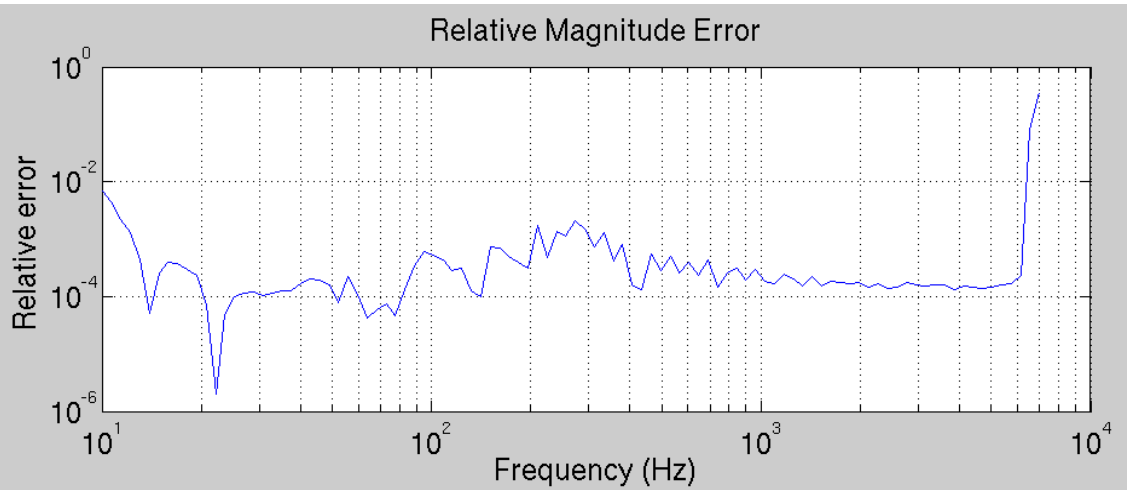


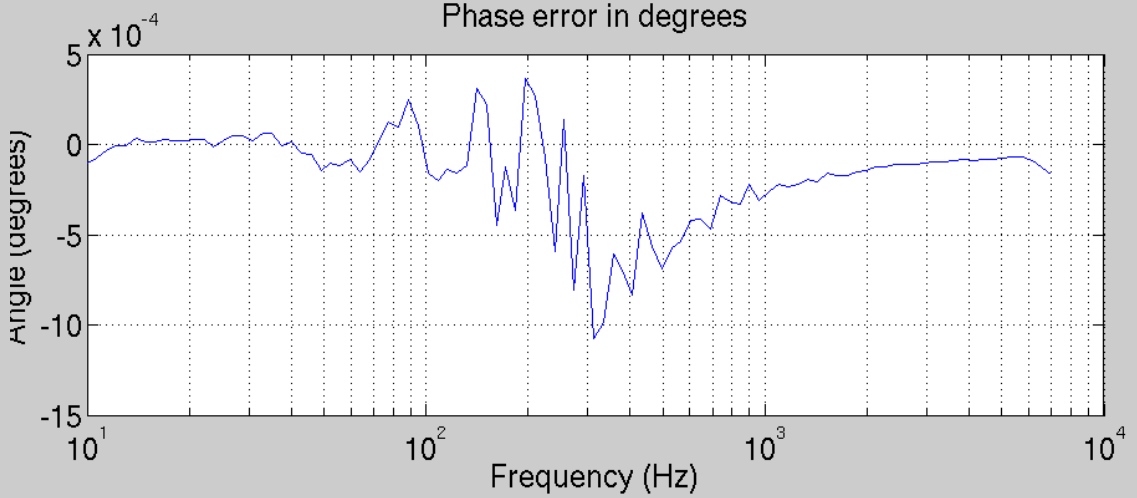



# Inverse sensing filter comparison:
# (our model vs. calibration model i.e. computeDARM in matlab)

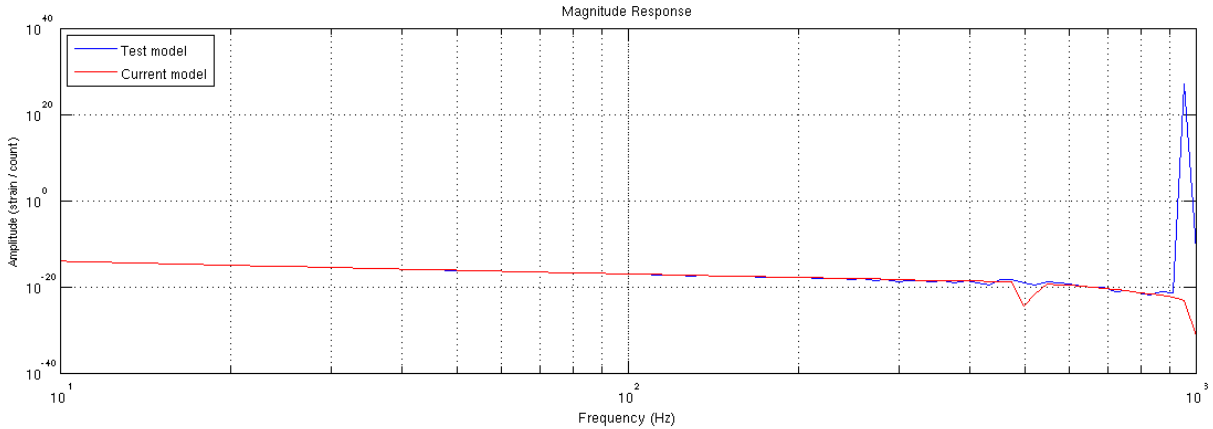


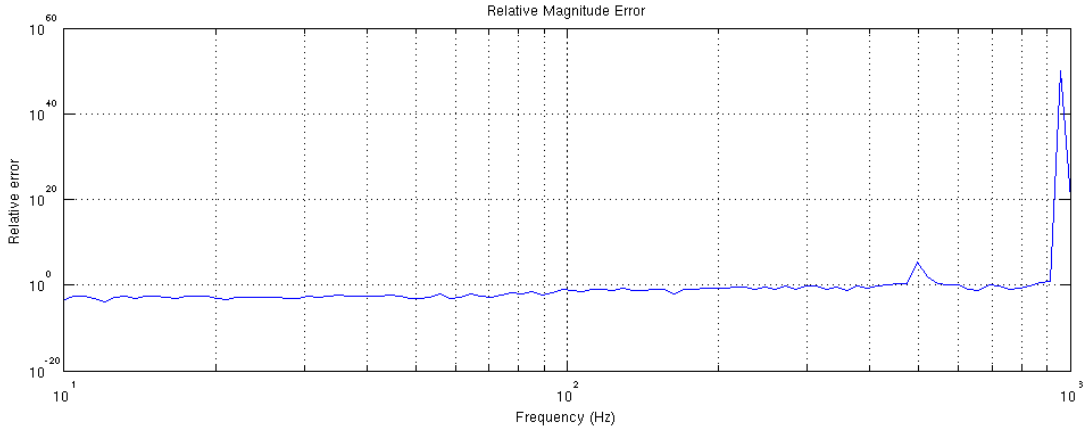



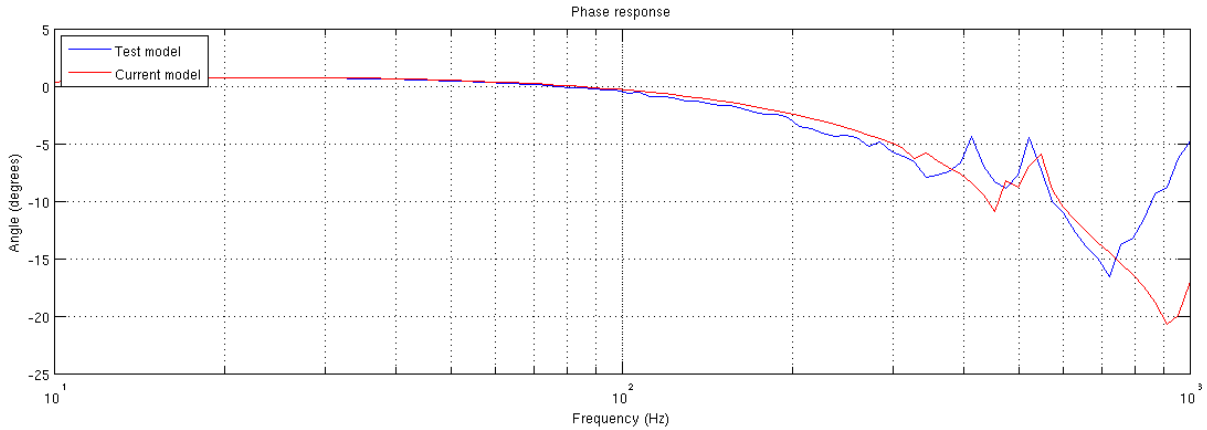



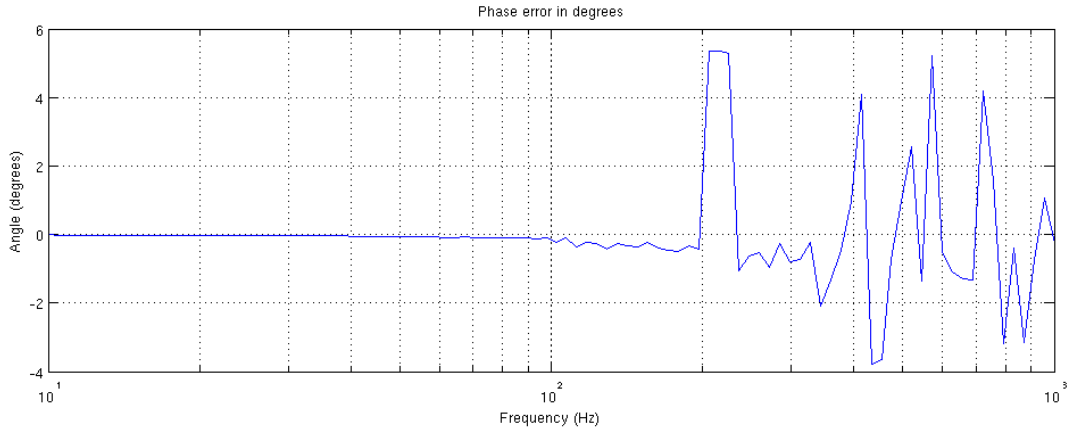



# Actuation pu filter comparison:

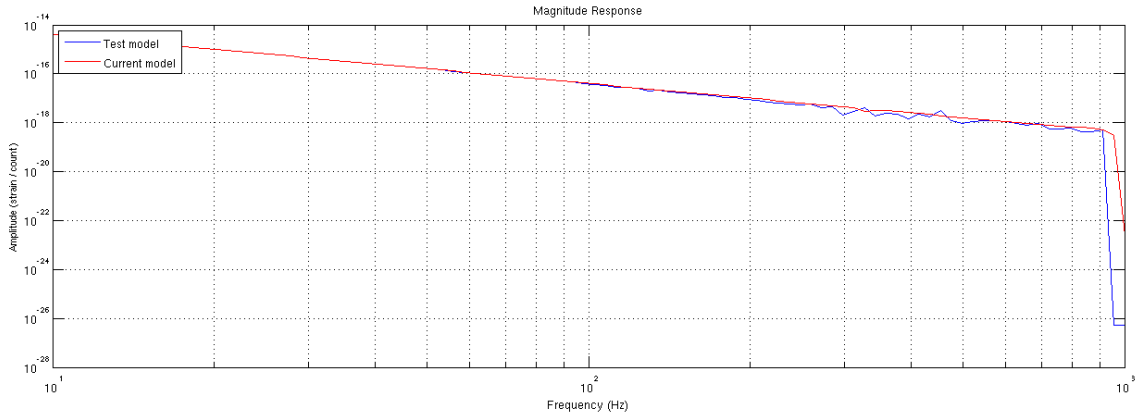


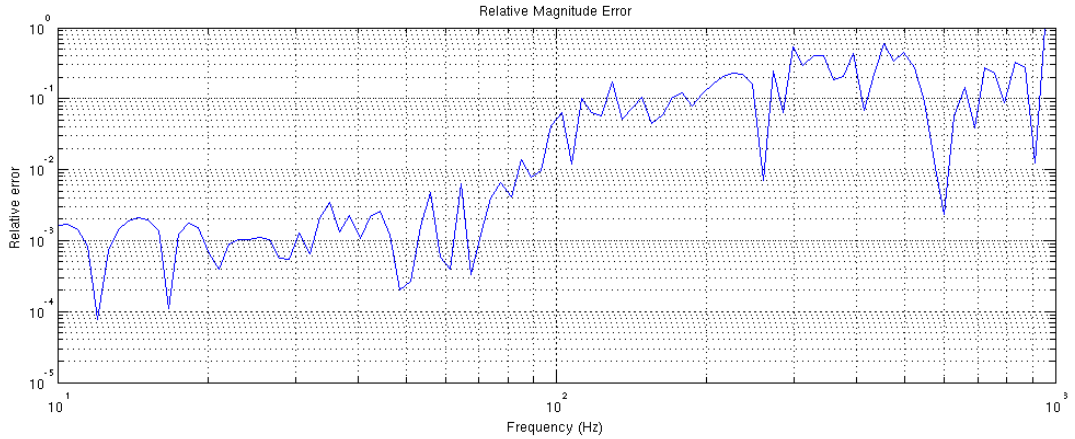



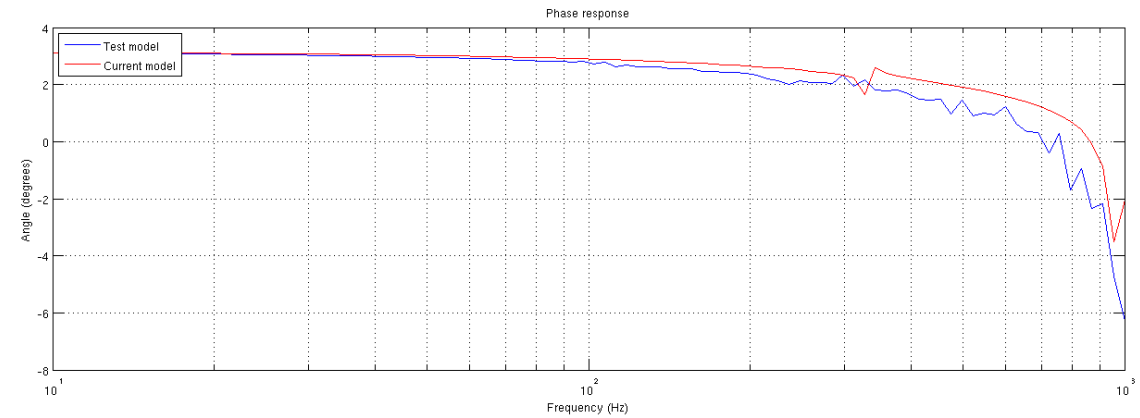



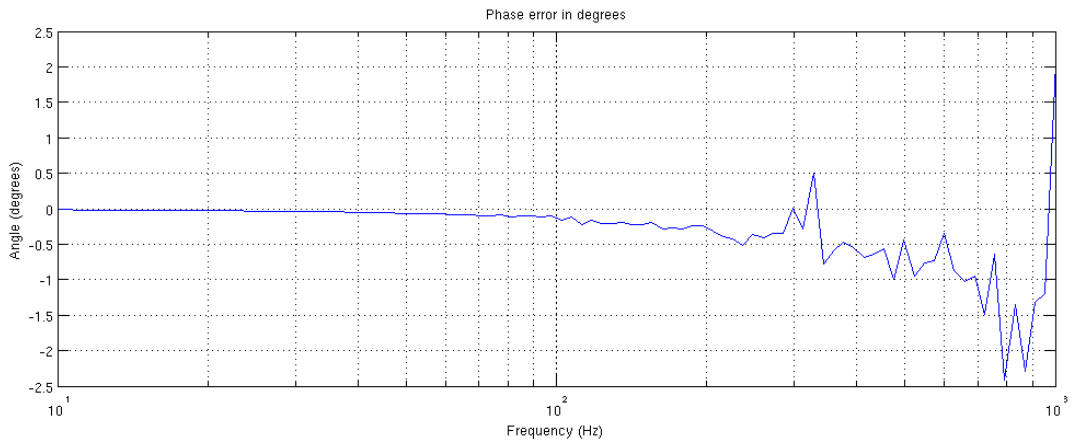



# Actuation tst filter comparison:



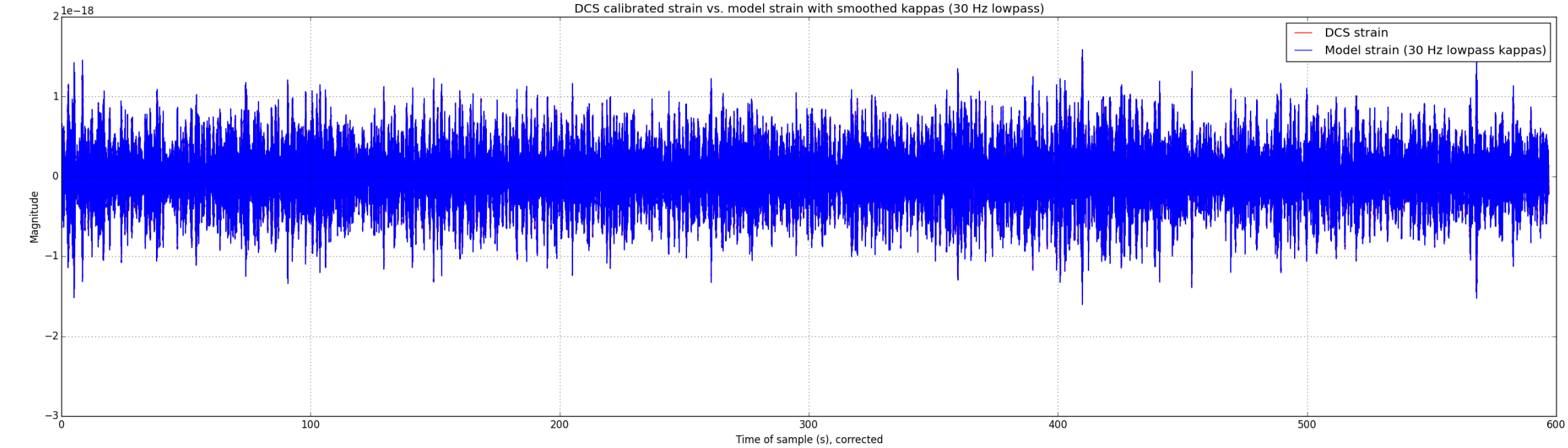



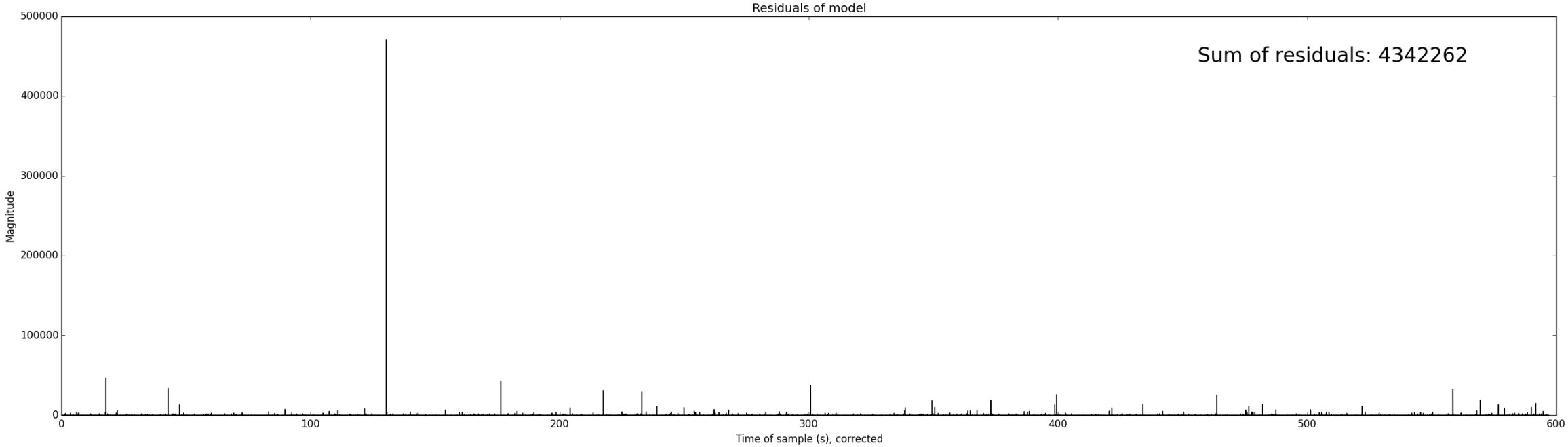



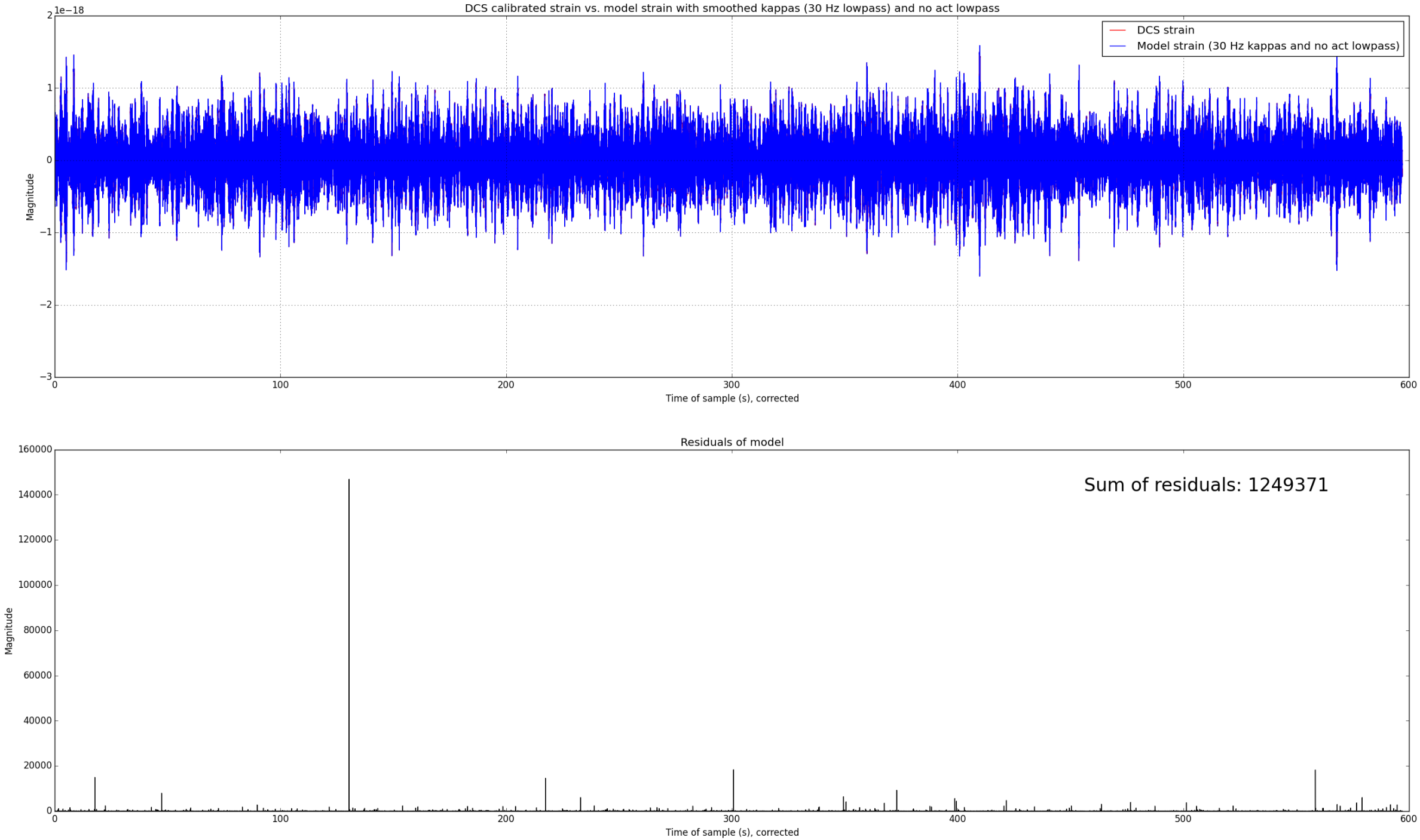
# Calibrated strain comparison between new model and DCS pipeline

Retrieve 600 seconds of data beginning April 28th, 2017 04:00 UTC from the following channels:

- L1:DCS-CALIB_STRAIN_CO1
- L1:DCS-CALIB_KAPPA_TST_REAL
- L1:DCS-CALIB_KAPPA_PU_REAL
- L1:DCS-CALIB_KAPPA_C_REAL
- L1:CAL-DARM_CTRL_WHITEN_OUT_DBL
- L1:CAL-DARM_ERROR_WHITEN_OUT_DBL

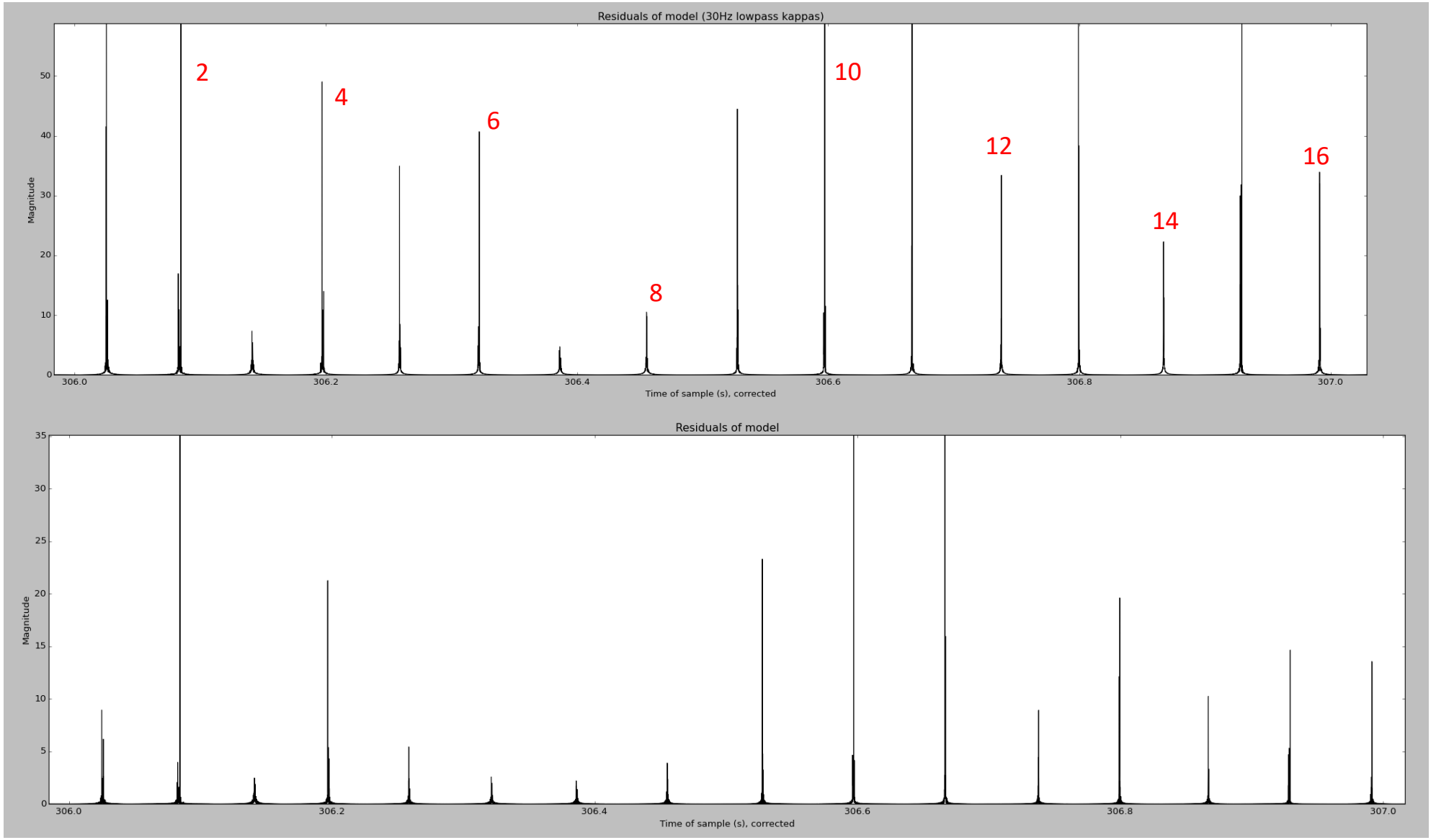

DCS calibrated strain vs. model strain with smoothed kappas (30 Hz lowpass)
1e−18
DCS strain
Model strain (30 Hz lowpass kappas)
Magnitude
Time of sample (s), corrected


Residuals of model
Sum of residuals: 4342262
Magnitude
Time of sample (s), corrected


DCS calibrated strain vs. model strain with smoothed kappas (30 Hz lowpass) and no act lowpass
1e−18
DCS strain
Model strain (30 Hz kappas and no act lowpass)
Magnitude
Time of sample (s), corrected
Residuals of model
Sum of residuals: 1249371
Magnitude
Time of sample (s), corrected

# Discovered a problem with kappas that is still unresolved

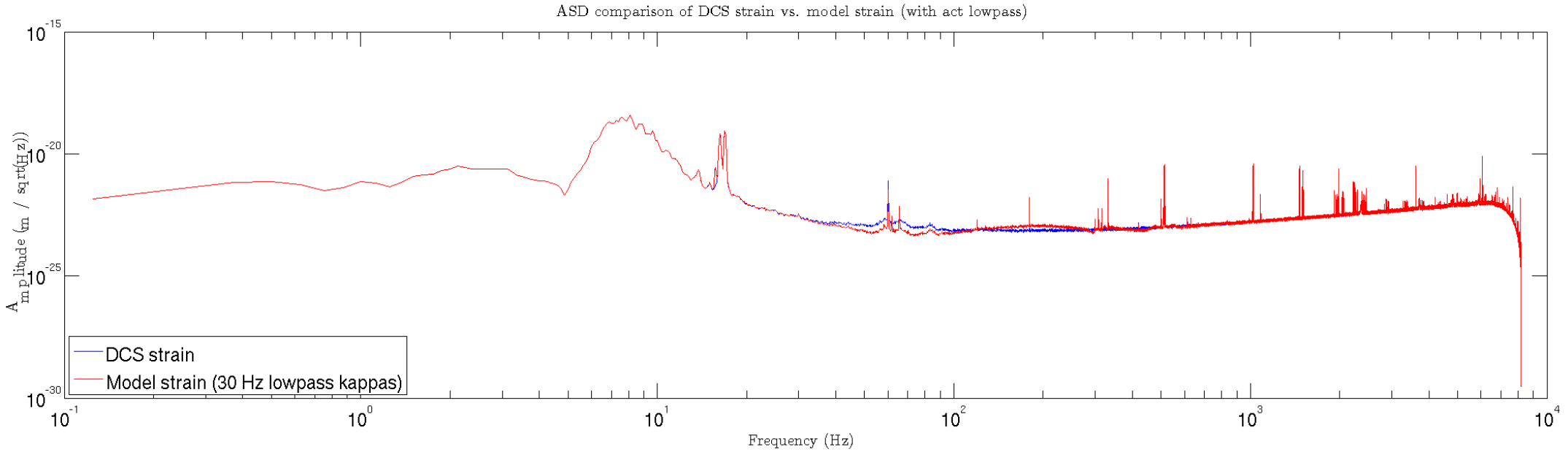
With act lowpass:

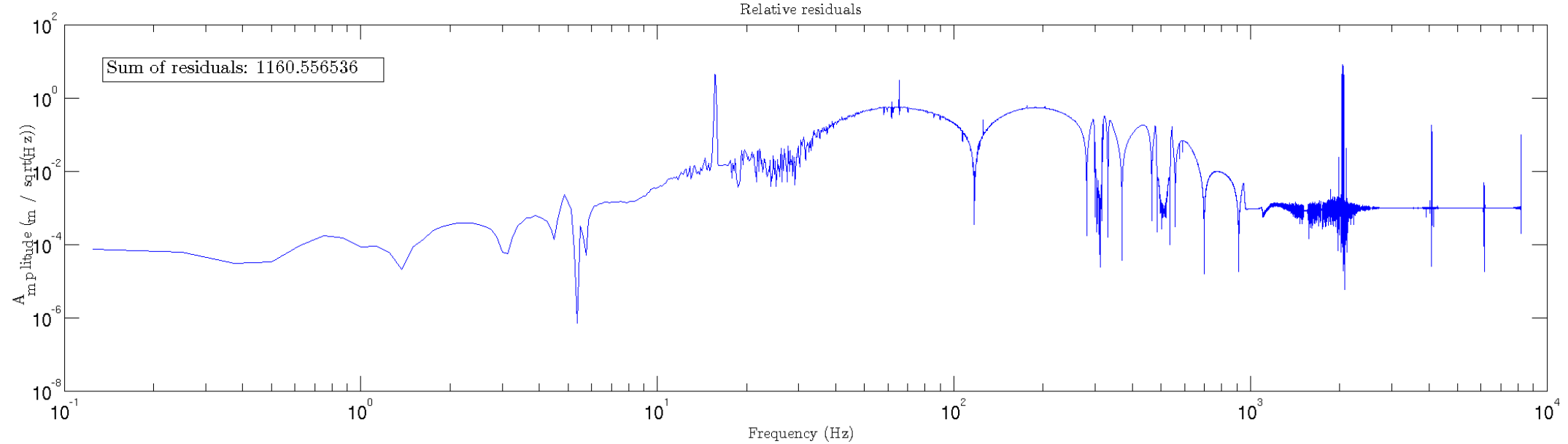


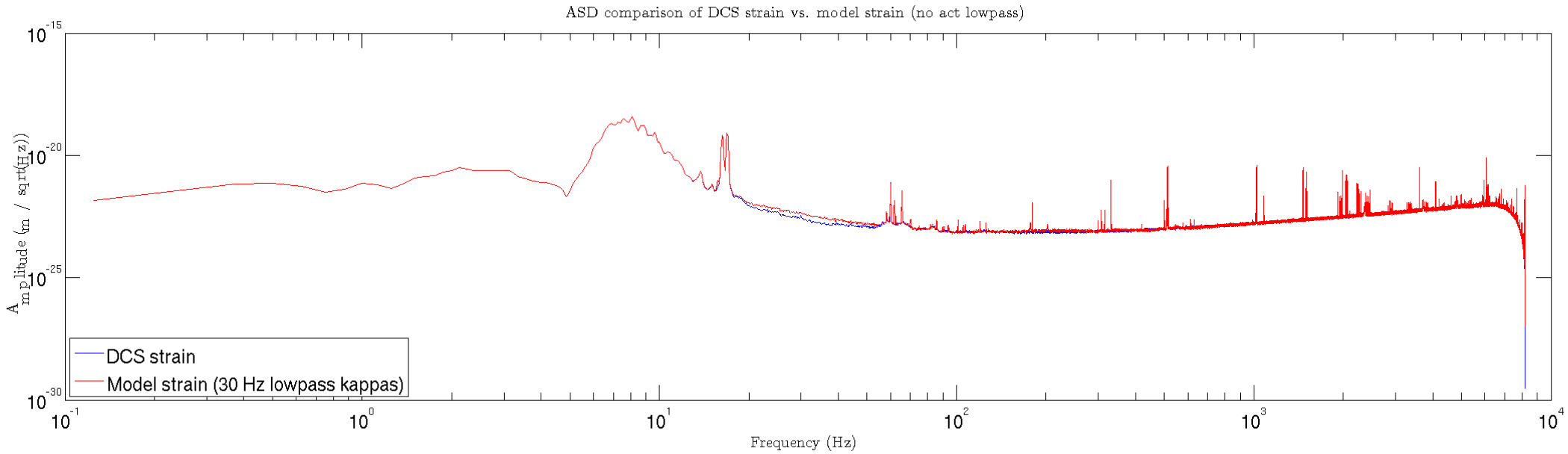

No act lowpass:

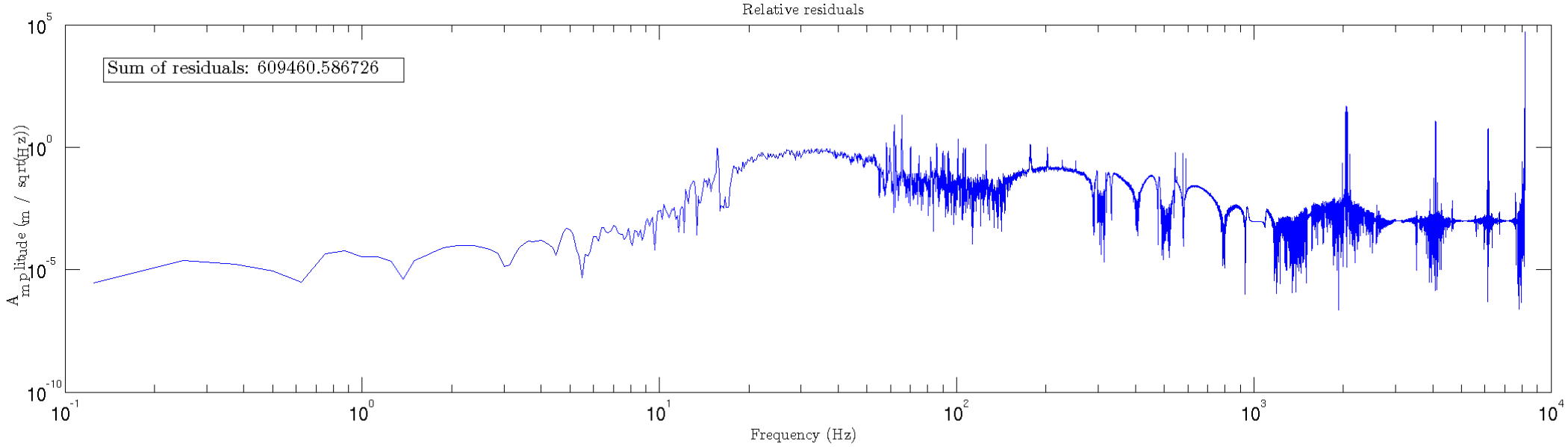
# Amplitude spectral density (ASD) comparison plots of DCS strain and model strain

Compared DCS and model calibrated strain after lowpassing kappas at 30 Hz, tested to see how actuation lowpass filters were affecting residuals between the two.


ASD comparison of DCS strain vs. model strain (with act lowpass)
Amplitude (m / sqrt(Hz))
Frequency (Hz)
DCS strain
Model strain (30 Hz lowpass kappas)


Relative residuals
Sum of residuals: 1160.556536
Amplitude (m / sqrt(Hz))
Frequency (Hz)


ASD comparison of DCS strain vs. model strain (no act lowpass)
Amplitude (m / sqrt(Hz))
Frequency (Hz)
DCS strain
Model strain (30 Hz lowpass kappas)


Relative residuals
Sum of residuals: 609460.586726
Amplitude (m / sqrt(Hz))
Frequency (Hz)

# Moving forward

- Resampling

  We will write C code that implements a sinc table and proper interpolation between data points (as used in DCS pipeline), and then replace our current, simple lowpass filters.

  Then, the output strain of the DCS model and our current model will be tested again (both ASD and TD plots). In addition, the magnitude and phase responses of the actuation FIR filters will be inspected.

- Moving to L1

  If this adjustment leads to a model with acceptable errors, the calibration pipeline will be moved to the real interferometer for testing.