LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	**LIGO-T1600384-v1-D**	2016/10/31

Online State Space Controls of Advanced LIGO Optics

Stephen Quinn, Aidan F Brooks

Distribution of this document:

Detector Group

California Institute of Technology
LIGO Project, MS 100-36
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
PO Box 159
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

1 Abstract

As the usable laser power within the aLIGO interferometer is increasing toward 750kW, one of the dominant problems is that of thermal lensing due to laser absorption heating in the fused silica optics. This project looks to apply a Kalman Filter numerical estimation algorithm to simulated thermal optic models within a finite element modelling software (COMSOL), updated with real data from the Hartmann Wavefront Sensors used in aLIGO, to give an improved state-space estimation of the thermal lensing effect. The intent is that this more modern, state space control system can be utilised by the aLIGO thermal compensation system to better actuate for the thermal lensing effect, improving mode matching and thus, overall signal power.

2 Introduction

2.1 Gravitational Waves

Gravitational Waves (GWs) were predicted in 1916 as a consequence of Einstein's General Relativity, where they are described as a result of changing mass-energy distributions in massive objects [1]. At a distance much greater than the immediate vicinity of the source, spacetime distortions due to massive bodies can be approximated by flat spacetime. In this linearised, weak field approximation, Gravitational Waves present as a small, oscillating perturbation of spacetime. [2] Gravitational waves are due to time variation in the mass quadrupole moment of the source and are seen as transverse waves of spatial strain, predicted to travel at the speed of light. Given any two separated locations, GWs are seen to cause a time varying contraction of the Lorentz length interval between them, this being the subject of LIGO's endeavour to detect these phenomena [3]. The first experimental indication of GWs was made by Hulse and Taylor in 1974. They discovered a binary system containing a pulsar and some other unknown similar mass object. By observing the orbital period of the two objects, it was measured that their separation was decreasing by approximately 3.1mm per period, corresponding to the energy loss predicted by the emission of GWs [4]. Within the last year, aLIGO has made two definitive detections of Gravitational Waves. The first (discovery) detection was of a binary black hole merger of a $36^{+5}_{-4}M_{\odot}$ and $29^{+4}_{-4}M_{\odot}$ black hole system with the detection having a Signal-to-Noise (SNR) of 21, corresponding to over 5.1σ confidence [3]. The second detection was also of a binary black hole merger, this time with masses of $14.2^{+8.3}_{-3.7}M_{\odot}$ and $7.5^{+2.3}_{-2.3}M_{\odot}$, with the detection having a SNR of 13, corresponding to over 5σ significance. This was done using the two aLIGO interferometers, which will be described immediately.

2.2 aLIGO overview

The following information is adapted from the aLIGO overview paper, see [3] for additional detail.

LIGO comprises two detector sites, located in Hanford, WA and Livingstone, LA. These de-

tectors attempt to detect GWs by measuring change in length due to the spacetime distortion occurring as GWs pass the Earth. This is achieved by utilising a dual-recycled Fabry-Perot Michaelson Interferometer (IFO), shown in Fig.1 . The measured length change, corresponding to strain in space time, is given by the length difference of the optical path between the perpendicular arms of the interferometer.

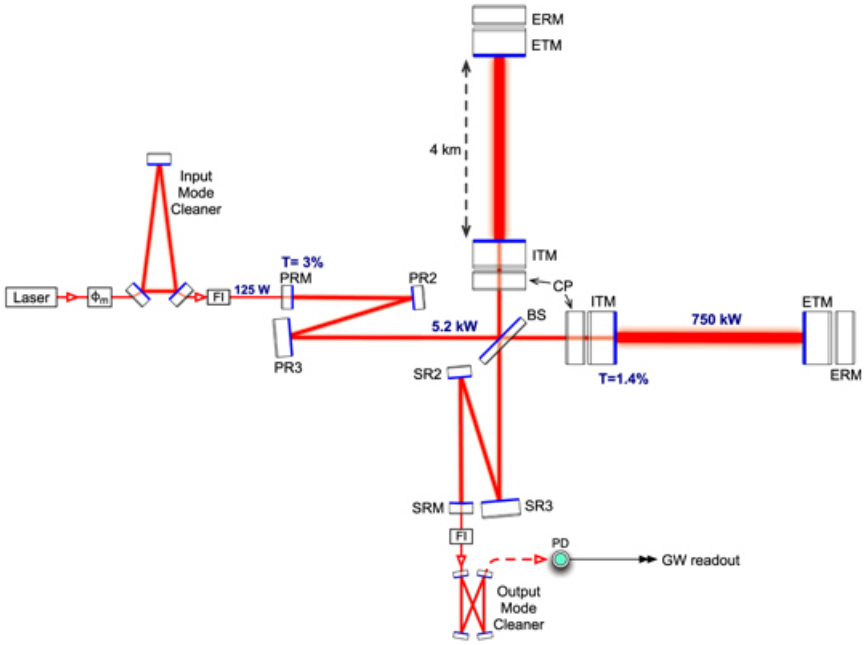

Figure 1: [3]Advanced LIGO optical configuration. ITM: input test mass; ETM: end test mass; ERM: end reaction mass; CP: compensation plate; PRM: power recycling mirror; PR2/PR3: power recycling mirror 2/3; BS: 50/50 beam splitter; SRM: signal recycling mirror; SR2/SR3: signal recycling mirror 2/3. The laser power numbers correspond to full-power operation. All of the components shown, except the laser and phase modulator, are mounted in the LIGO ultra-high vacuum system on seismically isolated platforms.

Before reconverging into a potential GW signal, the laser within aLIGO undergoes a complicated set of steps which will be outlined below, excluding more subtle detail.

The initial laser used is a three stage Nd:YAG laser which can potentially ramp up to 180W of power. Each stage is pumped by laser diodes and at this stage the laser is subject to a control system - the Pre Stabilized Laser (PSL) system - which stabilizes the laser's frequency, intensity and beam direction. The beam passes through a bow-tie shaped set of four mirrors (two flat, two curved) called the pre-mode-cleaner (PMC) (figure 2), reducing higher order modes. Then through the Input Mode Cleaner (IMC), a set of three curved mirrors are used to bring the input laser as close as possible to the lock mode of the IFO (at least as close as 95%). The next significant step is for the laser to pass through three mirrors, PRM (Power Recycling Mirror) and PR2/3 (Power Recycling Mirror 2/3). These mirrors feed the input laser into the main body of the interferometer and also serve to recycle most of the output laser power that doesn't make it to the photo diode (where potential detections are made).

Upon leaving the set of PRM mirrors, the beam is now within the 'core optics' of the IFO.

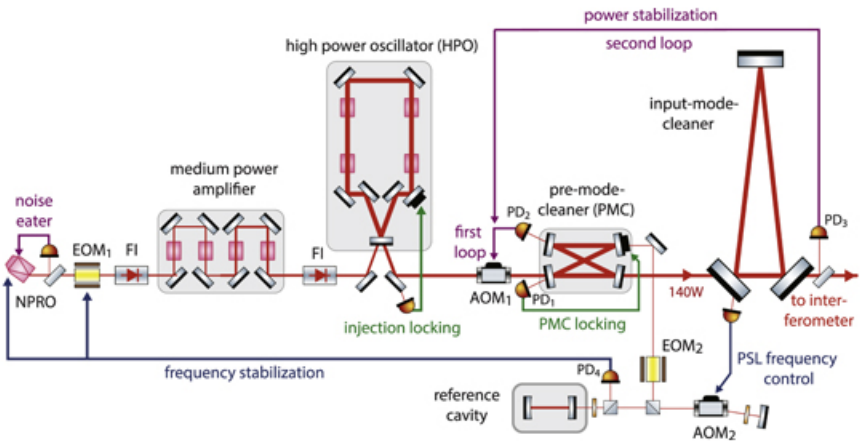

Figure 2: [3] Overview of the Pre-Stabilized Laser (PSL) system. Pre and input mode cleaner discussed above.

The beam first encounters a 50/50 beam splitter which separates the beam into the x and y arms of the detector. Each arm is 3996m in length and consists primarily of a Fabry-Perot cavity with a mirror at each end, known as the Initial Test Mass (ITM) at the mid station and End Test Mass and the end station of each arm. Each test mass consists of a 40kg cylinder of fused silica with radius of 0.17m and width of 0.2m. The ITM has radius of curvature (ROC) of 1934m and has 1.4% transmission. The ETM has ROC 2245m and transmits 5ppm. For Advanced LIGO the arm round-trip loss goal is set at 75 ppm. This would allow a total arm cavity stored power (both arms) of up to: $1/75$ ppm $= 1.3 \times 10^4$ times the input power. The 2020 goal is that the stored power within the arm can be ramped to 750kW. Note that a majority of the 75ppm loss is in the form of absorption within the optic itself, manifesting as increased thermal activity, the subject of this project.

The beam output from the Fabry-Perot arms potentially ends in three ways:

- If no signal is present, the beam is recycled via the aforementioned PRM system.
- If there is a potential signal, the beam is projected to the output signal channel wherein it is most often recycled by a analogous system to the PRM, called the SRM (Signal Recyling mirror).
- Potential signals go through a final mode cleaner called the Output Mode Cleaner then are output into a photodiode, where the potential signal is detected.

In order to detect GWs within a radius that allows for regular detections, aLIGO is required to be sensitive to strain on the order of 10^{-22}. Given the $\approx 10^4$ arm length this requires for measurement of length changes of the order 10^{-19}, i.e. unprecedented precision. To achieve this careful attention is paid to noise sources within the detector, this being the primary concern of a majority of the LIGO collaboration.

3 Thermal Lensing

Since the laser heating is not uniform but instead is focused on the centre of each optic. This produces a thermal lensing effect. In effect, the result is to produce a gradient in the refractive index of the mirrors[5] and possibly a mechanical distortion due to the thermal expansion of the optic along the beam axis. The result of uncontrolled thermal lensing is to reduce mode matching between the mirrors in aLIGO.
The shape of this thermal can be described by the following equations[6]

The Thermooptic Effect (path change ΔS_T due to changing refractive index, n , with temperature,T) is given by

$$\Delta S_T = \int_S \Delta n_T ds = \frac{dn}{dT} \int_S \Delta T$$

There is contribution to the lensing effect by virtue of thermal expansion of the fused silica[**?**] of the optic itself. This is described by surface deformation Δu along with temperature T and thermal expansion coefficient α by

$$\Delta u = \alpha \int_S \Delta T ds$$

Lastly there is the local change in refractive index due to the mechanic strain due to thermal expansion

$$\Delta S_E = \int_S \Delta n_E ds = \alpha p_{11} \int_S \Delta T$$

with p_{11} being the component of the elastooptic tensor along the probe beam polarization axis.

Unfortunately, these equations are not practical for determining the actual thermal lens profile. The solution does not contain measurement of the noise in the system, does not account for the non-uniformity of the beam profile or for the non-ideal conduction of heat through the optic. This calls for a numerical approach to thermal lens determination. The current method of controlling the thermal lensing in aLIGO optics is to take measurements of the lens profile using Hartmann Wavefront Sensors [7] and updating a simulated model of the optics through a FEM (Finite Element Modeller)

4 Thermal Compensation System (TCS)

This section is adapted from a comprehensive paper on the aLIGO Thermal Compensation System (A. Brooks et al. 2016), see [7] for more detail. Each mirror within the aLIGO system is subject to up to 750kW of laser power during operation of the detector. This power propagates in a number of different ways, namely, absorption as heat, transmission/reflection and some scattering. With a nominal coating absorption coefficient of 0.5ppm, there will be

up to 375mW of laser power, with a Gaussian spatial distribution, absorbed in the coatings of the test masses. Absorption within the coatings of these optics results in a negative radial temperature gradient within the optic and, subsequently, thermo-refractive substrate lenses in the recycling cavities and thermo-elastic surface deformation in the Fabry-Perot arms.

The TCS is a system of actuators and sensors to allow compensation for thermal effects in the test masses to reduce the overall distortion on transmission and reflection. The overarching purpose of this whole system is to maintain the interferometer in a given optical configuration so that it may continue to operate reliably. This means being able to measure, with dedicated sensors, the magnitude and spatial distribution of wavefront distortion introduced by self-heating. Compensate for the self-heating induced surface deformation, by actuating on the radius of curvature (ROC) of the test masses and crucially, performing these tasks without injecting additional noise into the GW detector

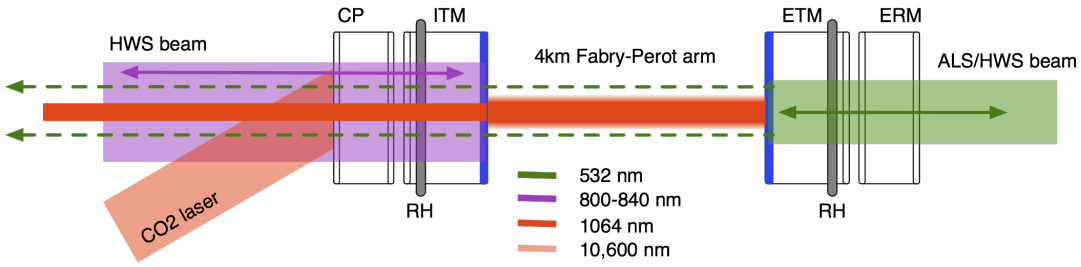

Figure 3: [7]aLIGO TCS overview. Absorption of the main interferometer beam (red) in the test masses induces thermal lenses (and wavefront distortion). Hartmann wavefront sensors (HWS) probe beams, purple and green, measure the thermal lens in the substrates of the ITM(Initial Test Mass)+CP(Compensation Plate) and ETM(End Test Mass)+ERM. Ring heaters (RH), grey, encircle the test masses and radiate heat onto the barrel to induce thermo-elastic and thermo-refractive distortion. CO2 laser beams, pink, heat the CP to induce a spatially tunable thermal lens. Note that the green laser beam at the ETM is also port of the Arm Length Stabilization (ALS) system. A small fraction the green laser leaks through the 4km arm (dashed green) and is used to align the ITM HWS beam to the ITM.

A schematic overview of the TCS sensor beams and actuators used in a single FP arm of aLIGO is shown in Figure 3. The components are

(a) Hartmann wavefront sensors (HWS) measuring the spatial distribution of the substrate thermal lenses in the ITMs and ETMs,

(b) ring heater (RH) actuators that heat the barrel of the ITMs and ETMs, altering the surface curvature and substrate lenses of these optics

(c) CO2 laser projectors that heat the compensation plate (CP) providing spatially tunable lensing actuation in the recycling cavities.

The control of the TCS system is implemented in the frequency domain. A potential means of improving the TCS is to move to time domain control and employ more modern state space control systems. This would allow finite element modeling and real time control of the system to be exploited. This was the primary goal of this project.

5 Finite Element modeling

Solving the differential equations describing the thermal lensing effect, as outlined in Section 3 is difficult. If we include noise terms, non-isotropy, radiation terms etc, analytical methods of solving the system are not achievable. This calls for numerical techniques. A technique which is particularly useful for our purposes is that of the Finite Element Method[8].

The defining concept behind the Finite Element Method is to take the full domain of some Differential Equation(DE) and split it into small intervals. The end points of each interval are called nodes (mesh points) and the grid of intervals and nodes spanning the domain is known as the mesh. This splitting of a large domain allows for a number of numerical techniques be established. Small intervals allow for a better approximation in Taylor series expansions and for small enough intervals there is the effect of local linearity[8]. Using this method, instead of having to solve a very difficult/impossible DE, many much simpler equations can be solved. These solutions are then summed over the domain with a weighting function which decides the contribution from each mesh point, with a corresponding error function which is minimised. This particular method within the Finite Element Method is called the Weak Form Approximation.

This mathematical technique is also powerful in that it corresponds well with computer simulation of a physical model. Applying the Finite Element Method to simulation is known as Finite Element Modeling (FEMing). Now a geometrical model of a physical system can be described by a mesh and using the Weak Form method, almost any required physical rules can be applied to the model.

There are many available software applications for FEMing. The one used for this project was COMSOL Multiphysics®[9]. The model being used primarily had a specified 3D geometric (contrary to 2D axially symmetric). The model had the following specifications:

Radius of Optic	0.17m
Height of Optic	.2m
Emissivity	0.9
Material	Silica
Initial Ambient Temperature	273.15K

Table 1: Parameters of the Thermal Optics model used for state space extraction for Kalman Filtering (See section 6). The model used 3D geometry, with time dependence and thermal model with surface to surface radiation.

A particular pertinent tool available within COMSOL® is LiveLink™. LiveLink is an interface between COMSOL and MATLAB whereby the user can export COMSOL model objects into MATLAB using a particular package. This allows for coding techniques to be applied to the model and also allows more direct extraction of the physical content of the model itself. By running the model within a loop, iterative model calculation becomes possible, an invaluable tool for simulating real time data streams such as those within aLIGO.

6 State Space

6.1 State Space Representations

Within Control System Engineering, the concept of state space is considered to be a space containing discreet sets of inputs, outputs and state variables. These sets are typically functions or functionals representing variables within a differential equation. The defining feature of a state space representation is the consideration of the state variable matrices. A function of state variables is one such that the state of a physical system can be entirely described in terms of the state variables at an arbitrary time. Typical examples of this are consideration of position and velocity in Newtonian Mechanics or Momentum and Position as given by Hamiltonian Mechanics. State space representations are very useful in Control System Engineering, they lend to time domain consideration of a system, application of physical law via the DEs describing progression of your system within state space and also allow large dimensional systems to be considered by one mathematical object.

Given the explicit temperature dependence of all the equations describing the current thermal lens profile within the aLIGO optics, a natural state space representation of the thermal optics is of temperature. Temperature at discreet points in the model can be taken and assembled in a state space vector. Theoretically, the higher the number of considered temperature values, the better the estimate of the current state of the system but in reality there is a limit on the number of points to be taken based on the sensitivity of the sensors being used.

There is an issue with this system in that, given the ΔT dependence of radiation, this particular set of state variables cannot fully describe the optic system since radiation is an important term. This was resolved by Hello et al.[10]. By taking the Stefan-Boltzman Law

$$E = \sigma T^4$$

and differentiating, and assuming that we have heat loss through a surface given by $F = \delta(T^4 - T_{ext}^4)$ we get:

$$P = 4\sigma T^3 dT$$

Hence there is a dependence on dT alone, meaning if we expand our representation to include dT also, we can once more completely describe the system.

For the purposes of this project, the real power of using a state space representation of the system is in that it allows implementation of state estimation techniques that incorporate predictive estimates of the state alongside real data to achieve accurate, real time estimates of the state. The most accurate of these methods was developed in 1960 by R. Kalman[11].

6.2 Kalman Filter

In essence, Kalman Filtering[12] serves as a method of creating a particularly accurate estimate for the state of a system. It does this by combining a predictive estimate of the value of

the state (prediction stage) along with actual measurement of the state (update stage), with each of these two values weighted toward that of least uncertainty. More precisely, given vector input

$$\mathbf{X}, \mathbf{P}$$

representing the initial inputted state of the system, alongside the covariance of the state. These then become the '(k-1)' input for the first prediction of state, of the form:

$$\mathbf{X_k} = \mathbf{F}\mathbf{X_{k-1}} + \mathbf{B}\mathbf{U_{k-1}} + \mathbf{V_{k-1}}$$

:

$$\mathbf{P_k} = \mathbf{F_k}\mathbf{P_{k-1}}\mathbf{F_k^T} + \mathbf{Q_{k-1}}$$

Here, $\mathbf{F}_k$ and $\mathbf{B}_k$ are matrices which transform our input matrices into the correct dimension both in terms of physical units and also in terms of matrix size. $\mathbf{V}_k$ and $\mathbf{Q}_k$ represent noise within the prediction calculation and covariance noise respectively. $\mathbf{U}_k$ is the physical model describing the time evolution of the system, hence, it works in accordance with $\mathbf{X}_{k-1}$ to produce the new prediction of the state of the system.

Next, an object known as the 'Kalman Gain', $\mathbf{K}$ is calculated. It is a function of the uncertainty in the measurement and uncertainty in the prediction. The Kalman gain serves to assign more importance to the value with less uncertainty, a result which will become more clear given the formalism of the update stage. Firstly, given a new variable $\mathbf{R}$ representing the uncertainty in the measured value and $\mathbf{H}$, another dimensional transformation. We have:

$$\mathbf{K} = \frac{\mathbf{P_k}\mathbf{H_k^T}}{(\mathbf{H_k}\mathbf{P_k}\mathbf{H_k^T} + \mathbf{R_k})}$$

So, for clarification, if the uncertainty in measurement is very high, $\mathbf{K} \to 0$ and, conversely if the prediction has large uncertainty, $\mathbf{K} \to 1$. This is a defining feature of Kalman Filtering as will be shown in the formalism for the update stage.

The value for the measurement, $\mathbf{Z}_k$, should be taken in the form:

$$\mathbf{Z}_k = \mathbf{C}\tilde{\mathbf{X}}_\mathbf{k} + \mathbf{Z_{k-1}}$$

Where $\mathbf{C}$ is a transformation matrix, $\mathbf{Z}_k$ is noise within the measurement and $\tilde{\mathbf{X}}_\mathbf{k}$ is the measured value of the state. Finally, we can combine the prediction and measurement, mediated by the Kalman Gain to get our expected value and covariance by:

$$\mathbf{X'_k} = \mathbf{X_k} + \mathbf{K}(\mathbf{Z_k} - \mathbf{H}\mathbf{X_k})$$

$$\mathbf{P'_k} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P_k}$$

These represent the most accurate estimate for the state of the system, which can now be reinputted into the predictive stage and updated again. These iterations, alongside updates with measurement, give rise to an accurate estimation of the system, which converges very quickly.

7 Extracting a state space model from a general FEM

The overall goal of this project is to construct a way to utilise COMSOL to extract the state space variables from a particular model so that Kalman Filtering can then be applied. Ideally, this would be a generalisable method which can be applied to any appropriate system within LIGO. To verify the method itself, the TCS system within LIGO was the main focus. It was found that using LiveLink in particular, a lot of data could be extracted from a particular model.

The main two equations being explored as key to capturing the essence of Kalman Filtering were:

$$\mathbf{X_k} = \mathbf{F}\mathbf{X_{k-1}} + \mathbf{B}\mathbf{U_{k-1}} + \mathbf{V_{k-1}} \tag{1}$$

$$\mathbf{P_k} = \mathbf{F_k}\mathbf{P_{k-1}}\mathbf{F_k^T} + \mathbf{Q_{k-1}} \tag{2}$$

Utilising COMSOL, it is possible to extract whatever state space variables are required, for instance, temperature of arbitrary points within the model can be determined by defining 'cutpoints' at selected points have using the command 'mpheval' for temperature, at these cutpoints. Given this knowledge of the state space, the challenge was how to propagate from $\mathbf{X}_k - 1$ to $\mathbf{X}_k$ i.e how to find the matrix F, the state propagation matrix, for a simple model.For instance, since the noise within the system, $\mathbf{V}_k$ is well understood as being a Gaussian noise distribution [7] it was disregarded for this early stage exploration.

A number of different methods were explored before arriving at a successful means:

- An initial endeavour was to search within the child nodes of the LiveLink script to attempt to extract the precise PDE being solved and then adapting this equation to take the state vector as an input and return the next time step. This was infeasible for a number of reasons. Namely, it was not possible to find the function being solved by the weak form method within COMSOL but only the sum of the equations describing the physics of the model. This sum was not practically convertible into the form required for Kalman Filtering. Also this method relied very heavily on the particular model being examined so was not generalisable as per the intention of the project.

- Another approach was to find F by first finding the covariance matrices $\mathbf{P}_k$ and then solving eqn 2 above, inverted. The intention was to use a method similar to the relaxation method employed in computational solid state physics. An array of points would be defined by the mesh points within the model and temperature measured. A small perturbation would be induced on one point and then the system would be relaxed by one time step. The new value of the temperature at each point would indicate the dependence of each point under small changes, giving the error. This method should work in principal but also failed to be easily generalisable and also the method of converting this information into an accurate measure of the covariance of the system was elusive.

- Potentially, COMSOL itself could be used as a state propagator, since it makes a predictive physical model. COMSOL's predicted value at the next time step could

be updated with real data via Kalman Filtering and this new data used as input for the next iteration, all within a LiveLink loop. The failing of this method was that it requires intense computational power which may be a) unavailable b) not powerful enough to complete the computation within the required time for real time estimation.

- Lastly, the method that was used a combination of the latter two. COMSOL was used to propagate the model forward by one time step, with the temperature of one point raised by 1K. The state matrix extracted before and after was compared to give an 'empirical' value for **F**.

8 Procedure

Firstly, note the form of equation 1 in the instance of the thermal optic model:

$$\begin{pmatrix} T_{k1} \\ T_{k2} \\ \vdots \\ \vdots \\ T_{kn} \end{pmatrix} = \mathbf{F} * \begin{pmatrix} T_{(k-1)1} \\ T_{(k-1)2} \\ \vdots \\ \vdots \\ T_{(k-1)n} \end{pmatrix} + \mathbf{B} * \begin{pmatrix} \Delta T_{k1} \\ \Delta T_{k2} \\ \vdots \\ \vdots \\ \Delta T_{kn} \end{pmatrix} + \mathbf{V}$$

As described previously, **V** was disregarded as it is a previously known quantity which can be added after **F** and **B** are extracted. To eliminate **B** initially, a model was used that neglected radiations terms. Since only radiation has the ΔT dependence. This meant our equation could be rewritten as:

$$\begin{pmatrix} T_{k1} \\ T_{k2} \\ \vdots \\ \vdots \\ T_{kn} \end{pmatrix} = \mathbf{F} * \begin{pmatrix} T_{(k-1)1} \\ T_{(k-1)2} \\ \vdots \\ \vdots \\ T_{(k-1)n} \end{pmatrix}$$

This, though simpler, still contains too many unknowns to be solved. We can determine all the elements of $\mathbf{X_k}$ and $\mathbf{X_k - 1}$ but not the interdependence of one temperature point on another, i.e. this is not a linear system, so we cannot solve for **F**. The solution to this was to determine **F** in a piecewise manner by setting all initial temperature points to zero except one, propagating this within the model, reinitialising and repeating for every point in the mesh. This means only one point is being considered at a time, removing the problem of interdependence This causes each iteration to give a new column of the square matrix **F** in the following way:

$$\begin{pmatrix} T_{k1} \\ T_{k2} \\ \vdots \\ \vdots \\ T_{kn} \end{pmatrix} = \mathbf{F} * \begin{pmatrix} T_{(k-1)1} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}$$

giving, for the first temperature point raised

$$\begin{pmatrix} \frac{T_{k1}}{T_{(k-1)1}} \\ \frac{T_{k2}}{T_{(k-1)1}} \\ \vdots \\ \vdots \\ \frac{T_{kn}}{T_{(k-1)1}} \end{pmatrix} = \begin{pmatrix} F_1 & & \\ F_2 & & \\ \vdots & & \\ \vdots & & \\ & & F_n \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}$$

Hence, our reading for temperature at each point, divided by the initial temperature at the non-zero point, gives us the i$^{\text{th}}$ column of **F**. Iterating n times then gives us every column of **F**.

So as to correspond to the approximate number of data points to be expecting when using state vectors taken from the HWS, 254 points were selected (this is also the number of mesh points produced in a 'coarse mesh' of the model within COMSOL. A 'point', for temperature determination, was defined to be a small region (radius¡distance between coarse mesh points) about these coarse mesh points defined using a top-hat function. Initial results with this configuration were not well defined however. [redacted]: A coarse mesh was defined within the model and the coordinates of each vertex saved. The coarse mesh was then replaced by a fine mesh, then by finding the minimum distance to neighbouring vertices, the nearest neighbour vertex was found (between the saved coarse mesh points and new fine mesh points). These 254 fine mesh points were then given a region around then as outlined above and temperature determination was done at these points. The code used for finding **F** is given in the appendices below.

The validity of F was confirmed by simulating progression of the model for 10 time steps in COMSOL and then comparing the predicted COMSOL values of the temperature against what was predicted by left multiplication of **F**, for instance checking the 4$^{\text{th}}$ time step against the 1$^{\text{st}}$ time step left multiplied by **F** 3 times. Figure 4 shows a plot of temperature at each point (y-axis) against vertex number (x-axis). It was discovered that there was a exact correspondence between predicted and resultant values however these values were diverging with each time step.

The cause of this divergence was the arbitrary definition of the 'regions' about mesh points. To have a well defined state space, it is necessary for the vector space to have a well defined basis. In order to achieve this, the space between the 254 chosen vertex points was defined by an interpolation function such that these functions would span the space without overlap

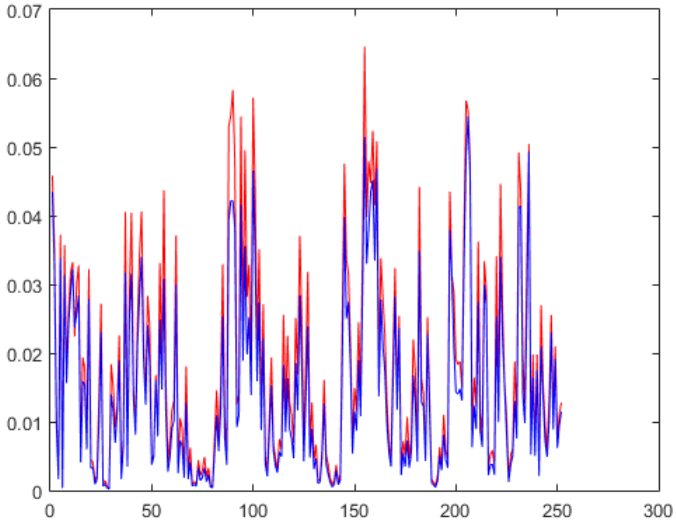

Figure 4: Plot of Temperature (arbitrary scale) on y-axis versus Vertex number (also arbitrary) on the x-axis. This plot only serves as a comparison between the COMSOL prediction in blue and the **F** matrix prediction in red. Note the discrepancy in magnitude and correspondence in shape, this plot strongly suggests linearity, confirming the method used to determine **F**

(this function is inbuilt in COMSOL). This means of defining a basis produced results that had a considerably better correspondence than previously , confirming the basis hypothesis.

The next step was to determine the value of **B**. This was done in a way analogous to finding **F**. In order to eliminate contributions from a general **F**, the temperature of all points was set to 0. Since radiation is a surface only term, one can apply 'forced' radiation (forced in that, since all temperature points were 0, there was no source, not a problem for this type of simulation!). A surface to ambient temperature difference of 1 was in-putted ($\Delta T = 1$). Then **B** was to be computed in a similar fashion to determination of **F**. Unfortunately, radiation terms are computationally expensive and this project ended before the data could be computed and analysed.

9 Conclusion and Discussion

The method used to find **F** was successful. The acquired plots gave evidence of linearity in T for the model and showed that this was a viable means of extracting the State Propagation Matrix from COMSOL. Early results indicate that this is the same for **B**. The theory that the initial results were skewed by a poorly defined basis set was backed up by the better fit produced by an appropriately defined basis.

Save for the lack of completion of the project as per the proposal, this project was successful. A means of applying extracting state space information from COMSOL has been discovered and applied in initial stages to the operation of the Thermal Compensation System. Moreover, the method used is not confined to the TCS but is easily generalizable to whatever system is required, so long as the state vectors being used can be dimensionalized into the correct form.

Looking to the future of the project, **B** needs to be found and the Gaussian noise term added as well. Using this data, the covariance matrix **P** will need to be determined either by clever linear algebra or failing that, employing the method suggested in section. 7. Once this is completed, the predictive part of Kalman Filtering will be fully realised. Next, the data coming in from the HWS needs to be dimensionalised so as to correspond to the form of the state space vectors. At this point, all ingredients required for Kalman Filtering are found. Offline data from the IFO should be introduced as a test and should this step be successful then this new online system is ready for implementation on the IFO itself.

The methods used in this future progress within the TCS will hopefully also form the procedure for generalising this method to any system. Given this, Kalman Filtering should be used in all appropriate systems, since it is the best known estimation technique. This would result in a large improvement to IFO functionality.

10 Acknowledgements

Firstly, I would like to Acknowledge the NSF for funding my project, LIGO for facilitating the project and Caltech for hosting me.

I would like to thank the 'SURF pen' for being such wonderful office mates and putting up with my 'Irish humour' all summer, and organising so many wonderful trips, it would have been a very different summer without you guys.

And the following people for teaching me something, showing me how to code or giving their ear for project discussion: Jamie Rollins, Alistair Heptonstall, Gabrielle Vajente and a particular thank you to Alan Weinstein for organising the LIGO SURF program and more so for being a mentor to us all, it's rare one can get such valuable advice on a weekly basis.

Similarly but for other SURFS: Nic, Wayne, Avi - thanks for showing me how to code/how to be a good student. Nikhil, our chats at the bench were one of the fruits of the summer.

Lastly, I'd like to thank my mentor Aidan, for your patience with me, for staring at white-boards and for showing me what an excellent experimentalist looks like. You thought me more than you realise.

References

[1] Einstein, A., 1916, Sitzungsberichte der Kniglich Preuischen Akademie der Wissenschaften (Berlin) 33, 688.

[2] The generation of gravitational waves. I - Weak-field sources,Thorne, K. S., Kovacs, S. J. http://adsabs.harvard.edu/full/1975ApJ...200..245T

[3] Advanced LIGO B. P. Abbott et al. http://iopscience.iop.org/article/10.1088/0264-9381/32/

[4] Hulse R A and Taylor J H 1975 Discovery of a pulsar in a binary system , Astrophys. J. 195 L51 http://adsabs.harvard.edu/full/1975ApJ...195L..51H

[5] Sheldon, S. J., L. V. Knight, and J. M. Thorne. "Laser-induced thermal lens effect: a new theoretical model." Applied optics 21.9 (1982): 1663-1669 https://www.osapublishing.org/ao/abstract.cfm?uri=ao-21-9-1663

[6] Lawrence, Ryan Christopher. Active wavefront correction in laser interferometric gravitational wave detectors. Diss. Massachusetts Institute of Technology, 2003. http://virgo.roma2.infn.it/lawrence.pdf

[7] Aidan F Brooks et al. Overview of Advanced LIGO adaptive optics (2016) https://www.osapublishing.org/ao/abstract.cfm?uri=ao-55-29-8256

[8] Becker, Eric B., Graham F. Carey, and John Tinsley Oden. "Finite Elements, An Introduction: Volume I." ., 258 (1981): 1981.

[9] https://www.comsol.com/ https://www.comsol.com/

[10] Hello, Patrice, and Jean-Yves Vinet. "Analytical models of thermal aberrations in massive mirrors heated by high power laser beams." Journal de Physique 51.12 (1990): 1267-1282.

[11] Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." Journal of basic Engineering 82.1 (1960): 35-45. http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=14304

[12] Kalman filtering : theory and practice / Mohinder S. Grewal and Angus P. Andrews.

A Code used to find F

```
%function out = model

PointwiseTfieldimprove.m

%Model exported on Aug 9 2016, 11:12 by COMSOL 5.1.0.234.
tic;
['start']
datestr(now, 'HH:MM:SS')

import com.comsol.model.*
import com.comsol.model.util.*

Initalise the model
model = ModelUtil.create('Model');
model.modelPath('C:\Users\squinn\Desktop\COMSOL files');
model.label('HerecomestheF.mph');
model.comments(['Untitled\n\n']);

model.file.clear;

model.modelNode.create('comp1');

model.geom.create('geom1', 3);
model.geom('geom1').create('cyl1', 'Cylinder');
model.geom('geom1').feature('cyl1').set('r', '.17');
model.geom('geom1').feature('cyl1').set('h', '.2');
model.geom('geom1').run;

%This initialises a coarse mesh to get points for testing, after the
%solution is plotted (~line 70), mesh information is taken then a fine
%mesh is made, for better computation
['coarse mesh']
datestr(now, 'HH:MM:SS')
mesh0 = model.mesh.create('mesh1', 'geom1');
mesh0.feature('size').set('hauto','7'); \%Meshing is done with a 1-9 index, 1 being fine
model.mesh('mesh1').run;


%Solve the model the first time, the things being created are self
%explained, they make the solution work then output useful information
['First solution']
datestr(now, 'HH:MM:SS')
model.study.create('std1');
model.study('std1').create('time', 'Transient');
```

```
model.sol.create('sol1');
model.sol('sol1').study('std1');
model.sol('sol1').attach('std1');
model.sol('sol1').create('st1', 'StudyStep');
model.sol('sol1').create('v1', 'Variables');
model.sol('sol1').create('t1', 'Time');
model.sol('sol1').feature('t1').create('fc1', 'FullyCoupled');
model.sol('sol1').feature('t1').create('d1', 'Direct');
model.sol('sol1').feature('t1').feature.remove('fcDef');

model.study('std1').feature('time').set('initstudyhide', 'on');
model.study('std1').feature('time').set('initsolhide', 'on');
model.study('std1').feature('time').set('solnumhide', 'on');
model.study('std1').feature('time').set('notstudyhide', 'on');
model.study('std1').feature('time').set('notsolhide', 'on');
model.study('std1').feature('time').set('notsolnumhide', 'on');
model.study('std1').feature('time').set('tlist', 'range(0,1000,2000)');

model.sol('sol1').attach('std1');
model.sol('sol1').feature('t1').set('maxstepbdf', '200');
model.sol('sol1').feature('t1').set('tlist', 'range(0,1000,2000)');
model.sol('sol1').feature('t1').set('estrat', 'exclude');
model.sol('sol1').feature('t1').set('maxstepbdfactive', true);
model.sol('sol1').feature('t1').set('maxorder', '2');
model.sol('sol1').feature('t1').feature('fc1').set('jtech', 'once');
model.sol('sol1').feature('t1').feature('fc1').set('damp', '0.9');
model.sol('sol1').feature('t1').feature('fc1').set('maxiter', '5');
model.sol('sol1').feature('t1').feature('d1').set('linsolver', 'pardiso');
model.sol('sol1').runAll;

%The solution is just run, vertices of coarse mesh obtained, fine mesh
%created, new vertices retrieved then, using the loop below, the fine
%mesh points, Mf, nearest to coarse mesh points Mc are found and
%output, these points are where temperatures are set and evalutated

%Mc is the vertex points of the coarser '7' mesh
Mc = mesh0.getVertex; %get vertex coords from previously computed coarse mesh

['Making fine mesh']
datestr(now, 'HH:MM:SS')
%Creating a fine '3' mesh
mesh0.feature('size').set('hauto','3'); %redefine mesh density to '3'
model.mesh('mesh1').run;
model.sol('sol1').runAll; %Solution is run again to allow use of '.getVertex' below
```

```
Mf = mesh0.getVertex;

dim = size(Mc); %loop over number of coarse mesh points
A = zeros(1,dim(2));
rmin = 1;

['Nearest mesh points']
datestr(now, 'HH:MM:SS')
%This loop finds the points in Mf nearest to points in Mc and outputs these
%points in 'B'
    for ii = 1:dim(2)

        r0 = Mc(:,ii);

        r = sqrt(((Mf(1,:)-r0(1)).^2)+((Mf(2,:)-r0(2)).^2)+((Mf(3,:)-r0(3)).^2));
        ind0 = find(r == min(r));
        ind0 = ind0(1);

        A(ii) = ind0;

    end


    B = Mf(:,A);  %B contains the coords of the fine mesh points found nearest to coarse
    dim0 = size(B); %To ensure main loop iterated over number of points in question

%Now that the points to be evaluated are set, the model is run for the
%parameters set to increase the temperature at each point
%
Tf = [];
model.file.create('res16');


['Starting model loop']
datestr(now, 'HH:MM:SS')

%This is the 'main loop' it causes the model to be solved for each point in
%the mesh. If you want to run the model just once, change jj.
for jj = 1:dim0(2)

    Progress = sprintf('Iteration no.%d',jj) %To know how many iterations are finished


    %Initalise the model, everytime
    model = ModelUtil.create('Model');
```

```
    model.modelPath('C:\Users\squinn\Desktop\COMSOL files');
    model.label('HerecomestheF.mph');
    model.comments(['Untitled\n\n']);

    str1 = sprintf('pev%d',jj);

    model.file.clear;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Defining interpolation function
    ['Interpolation function']
    datestr(now, 'HH:MM:SS')

    B = B'; %B is the wrong dimension for what we want, this line fixes it
    Interpoltest = [B zeros(size(B,1),1)]; %Make a new array with coords of mesh points
    Interpoltest(jj,4) = 1; %Make one of the zeroes be 1, iterating through evert point
    dlmwrite('Interpoltest.txt', Interpoltest) %Write to text file
    B = B'; %because B is defined once then reused in this loop iteratively, if B=B' was
            %it would be wrong half the time

    model.func.remove('int1');%remove version from last iteration

    model.func.create('int1', 'Interpolation');%Make new one
    model.func('int1').set('source', 'file');
    model.func('int1').set('filename', 'C:\Users\squinn\Desktop\COMSOLfiles\Defining a b
    model.func('int1').importData; %Get the new interpolation function for this iteratio




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ['Solve the Model']
    datestr(now, 'HH:MM:SS')
    model.modelNode.create('comp1');

    model.geom.create('geom1', 3);
    model.geom('geom1').create('cyl1', 'Cylinder');
    model.geom('geom1').feature('cyl1').set('r', '.17');
    model.geom('geom1').feature('cyl1').set('h', '.2');
    model.geom('geom1').run;


    %Working with a fine mesh (should {and does} have same mesh points as
    %we found in B
    mesh0 = model.mesh.create('mesh1', 'geom1');
```

```
mesh0.feature('size').set('hauto','3');
model.mesh('mesh1').run;



%Simulating silica, for more accuracy the properaties below could be
%set to correspond to the actual aLIGO fused silica properties
model.param.set('e', '0.9', 'Surface Emissivity');
model.material.create('mat1', 'Common', 'comp1');
model.material('mat1').propertyGroup.create('Enu', 'Young''s modulus and Poisson''s
model.material('mat1').propertyGroup.create('RefractiveIndex', 'Refractive index');
model.material('mat1').label('Silica glass');
model.material('mat1').set('diffuse', 'custom');
model.material('mat1').set('noise', 'on');
model.material('mat1').set('roughness', '0.02');
model.material('mat1').set('family', 'custom');
model.material('mat1').set('specular', 'custom');
model.material('mat1').set('noisefreq', '1');
model.material('mat1').set('lighting', 'cooktorrance');
model.material('mat1').set('ambient', 'custom');
model.material('mat1').set('fresnel', '0.99');
model.material('mat1').set('roughness', '0.02');
model.material('mat1').set('customambient', {'1' '1' '1'});
model.material('mat1').set('customspecular', {'1' '1' '1'});
model.material('mat1').set('customdiffuse', {'1' '1' '1'});
model.material('mat1').propertyGroup('def').set('relpermeability', {'1' '0' '0' '0'
model.material('mat1').propertyGroup('def').set('electricconductivity', {'1e-14[S/m]
model.material('mat1').propertyGroup('def').set('thermalexpansioncoefficient', {'0.5
model.material('mat1').propertyGroup('def').set('heatcapacity', '703[J/(kg*K)]');
model.material('mat1').propertyGroup('def').set('relpermittivity', {'2.09' '0' '0' '
model.material('mat1').propertyGroup('def').set('density', '2203[kg/m^3]');
model.material('mat1').propertyGroup('def').set('thermalconductivity', {'1.38[W/(m*K
model.material('mat1').propertyGroup('Enu').set('youngsmodulus', '73.1e9[Pa]');
model.material('mat1').propertyGroup('Enu').set('poissonsratio', '0.17');
model.material('mat1').propertyGroup('RefractiveIndex').set('n', '');
model.material('mat1').propertyGroup('RefractiveIndex').set('ki', '');
model.material('mat1').propertyGroup('RefractiveIndex').set('n', {'1.45' '0' '0' '0'
model.material('mat1').propertyGroup('RefractiveIndex').set('ki', {'0' '0' '0' '0' '

 %Creating the appropriate Physics, here we do not have radiation
['Adding Physics']
model.physics.create('ht', 'HeatTransferWithSurfaceToSurface', 'geom1');
model.physics('ht').feature('init1').set('Tinit', '273[K] + int1(x,y,z)');



model.study.create('std1');
```

```
    model.study('std1').create('time', 'Transient');

    model.sol.create('sol1');
    model.sol('sol1').study('std1');
    model.sol('sol1').attach('std1');
    model.sol('sol1').create('st1', 'StudyStep');
    model.sol('sol1').create('v1', 'Variables');
    model.sol('sol1').create('t1', 'Time');
    model.sol('sol1').feature('t1').create('fc1', 'FullyCoupled');
    model.sol('sol1').feature('t1').create('d1', 'Direct');
    model.sol('sol1').feature('t1').feature.remove('fcDef');

    model.study('std1').feature('time').set('initstudyhide', 'on');
    model.study('std1').feature('time').set('initsolhide', 'on');
    model.study('std1').feature('time').set('solnumhide', 'on');
    model.study('std1').feature('time').set('notstudyhide', 'on');
    model.study('std1').feature('time').set('notsolhide', 'on');
    model.study('std1').feature('time').set('notsolnumhide', 'on');

    model.result.dataset.create('dset2', 'Solution');

   %The commented stuff below shows useful plot information which is done
   %standardly for whatever model is made, it is not currently useful for
   %our calculation
%     model.result.create('pg1', 'PlotGroup3D');
%     model.result.create('pg2', 'PlotGroup3D');
%     model.result.create('pg3', 'PlotGroup3D');
%     model.result('pg1').create('surf1', 'Surface');
%     model.result('pg2').create('iso1', 'Isosurface');
%     model.result('pg3').create('surf1', 'Surface');
%     model.result('pg3').create('surf2', 'Surface');
%     model.result('pg3').create('surf3', 'Surface');
%     model.result('pg3').feature('surf2').create('def1', 'Deform');
%     model.result('pg3').feature('surf3').create('def1', 'Deform');


    model.study('std1').feature('time').set('tlist', 'range(0,1000,2000)');

    model.sol('sol1').attach('std1');
    model.sol('sol1').feature('t1').set('maxstepbdf', '200');
    model.sol('sol1').feature('t1').set('tlist', 'range(0,1000,2000)');
    model.sol('sol1').feature('t1').set('estrat', 'exclude');
    model.sol('sol1').feature('t1').set('maxstepbdfactive', true);
    model.sol('sol1').feature('t1').set('maxorder', '2');
    model.sol('sol1').feature('t1').feature('fc1').set('jtech', 'once');
    model.sol('sol1').feature('t1').feature('fc1').set('damp', '0.9');
```

```
    model.sol('sol1').feature('t1').feature('fc1').set('maxiter', '5');
    model.sol('sol1').feature('t1').feature('d1').set('linsolver', 'pardiso');
    model.sol('sol1').runAll;



%Don't need this stuff either
%     model.result('pg1').label('Temperature (ht)');
%     model.result('pg1').set('looplevel', {'1'});
%     model.result('pg1').feature('surf1').set('colortable', 'ThermalLight');
%     model.result('pg2').label('Isothermal Contours (ht)');
%     model.result('pg2').set('looplevel', {'1'});
%     model.result('pg2').feature('iso1').set('number', '10');
%     model.result('pg2').feature('iso1').set('colortable', 'ThermalLight');
%     model.result('pg3').label('Radiosity (ht)');
%     model.result('pg3').set('looplevel', {'1'});
%     model.result('pg3').feature('surf1').label('Radiosity');
%     model.result('pg3').feature('surf1').set('descr', 'J_plot');
%     model.result('pg3').feature('surf1').set('unit', '');
%     model.result('pg3').feature('surf1').set('expr', 'J_plot');
%     model.result('pg3').feature('surf2').label('Upside Radiosity');
%     model.result('pg3').feature('surf2').set('descr', 'Ju_plot');
%     model.result('pg3').feature('surf2').set('unit', '');
%     model.result('pg3').feature('surf2').set('expr', 'Ju_plot');
%     model.result('pg3').feature('surf2').feature('def1').set('expr', {'nx/sqrt(tremetr
%     model.result('pg3').feature('surf2').feature('def1').set('scaleactive', true);
%     model.result('pg3').feature('surf2').feature('def1').set('scale', '0.1');
%     model.result('pg3').feature('surf3').label('Downside Radiosity');
%     model.result('pg3').feature('surf3').set('descr', 'Jd_plot');
%     model.result('pg3').feature('surf3').set('unit', '');
%     model.result('pg3').feature('surf3').set('expr', 'Jd_plot');
%     model.result('pg3').feature('surf3').feature('def1').set('expr', {'-nx/sqrt(tremet
%     model.result('pg3').feature('surf3').feature('def1').set('scaleactive', true);
%     model.result('pg3').feature('surf3').feature('def1').set('scale', '0.1');


    ['Made it to start of cutpoint loop']
     datestr(now, 'HH:MM:SS')
    %%%%%%%%%%Creating Cut points for temperature determination%%%%%%%%%%%%%%


%This loop creates a 'cutpoint' at each mesh point. These are points where
%variables can be read out but which don't affect the geometry or meshing
    for i = 1:dim0(2)
        str = sprintf('cpt%d',i);
        str1 = sprintf('pev%d',i);
```

```
%Create cutpoints and set their coords to those of mesh points
        model.result.dataset.create(str, 'CutPoint3D');
        model.result.dataset(str).set('pointx', num2str(B(1,i)));
        model.result.dataset(str).set('pointy', num2str(B(2,i)));
        model.result.dataset(str).set('pointz', num2str(B(3,i)));

%Make each cutpoint an evaluable point
        model.result.dataset(str).run;
        model.result.numerical.create(str1, 'EvalPoint');
        model.result.numerical(str1).set('data', str);
    end
    %Below finds the temperature at each 'cut point' defined earlier( a cut
    %point is just a named point that does not impact the geometry or
    %meshing
    out = model;
    ['Made it to temperature evaluation']
    datestr(now, 'HH:MM:SS')
    Tj = [];

    for j = 1:dim0(2)


        str = sprintf('cpt%d',j);

        %Reads the temperature value at each cut point, many unique and
        %useful LiveLink commands start with 'mph...' I'd recommend
        %exploring these
        T = mphinterp(model,'T','dataset',str);

        Tj = [Tj T];


    end

%Saves the value of the temperature at each point in 'Tf'
    Tf = [Tf Tj];
    fullResults(jj).Tj = Tj;
    save('SPM_results.mat', 'fullResults', 'jj', 'Tf');

    ['Made it to the end']
    datestr(now, 'HH:MM:SS')
    toc;
end

%Once this code has been run, script '2' will take 'Tf' and automatically
%find F
```