7-1-2009

An Optimization approach to plant-controller co-design

Jared S. Russell

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Accessed from

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritsscholarworks@rit.edu.
An Optimization Approach to Plant-Controller Co-design

by

Jared S. Russell

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Juan C. Cockburn
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
July 2009

Approved By:

Dr. Juan C. Cockburn
Primary Advisor – R.I.T. Dept. of Computer Engineering

Dr. Marcin Lukowiak
Secondary Advisor – R.I.T. Dept. of Computer Engineering

Dr. Marcos Esterman
Secondary Advisor – R.I.T. Dept. of Industrial and Systems Engineering
Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title: An Optimization Approach to Plant-Controller Co-design

I, Jared S. Russell, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

_________________________________
Jared S. Russell

_________________________________
Date
This work is dedicated to the memories of Cliff G. Russell and Dr. James C. H. Russell.
Acknowledgements

This work would not have been possible had it not been for the vast pool of knowledge, guidance, and patience of Dr. Juan C. Cockburn. The author would also like to thank Dr. Marcos Esterman and Alvaro J. Rojas of RIT’s PRISM laboratory for their support with the test bed systems that formed the basis of this research. Additionally, many thanks are owed to Pamela Steinkirchner for helping to facilitate a long-distance research correspondence.
Improving the behavior of a controlled mechanical device is traditionally accomplished by manipulating the parameters of the control system in isolation. If permitted, a better solution can be achieved by including the physical attributes of the mechanical structure as optimization variables. However, this expansion of the search space increases the importance of properly formulating the optimization problem to avoid undesirable behavior. Some modern (e.g. $\mathcal{H}_\infty$) methods can be used to simultaneously optimize dynamic performance and robustness, but they require high levels of understanding and do not handle nonlinearities and arbitrary optimization constraints without additional augmentation. This work proposes and applies a method to add robustness to an optimized stabilizing controller and plant combination using constrained performance index optimization of chirp signal tracking. Using a chirp reference helps to improve the generality of the system response and ensures that resonant modes lay outside the useful range of input frequencies. Moreover, applying constraints on physical optimization parameters and their sensitivities helps to limit the solution space of a potentially high-dimensional problem while ensuring that the resultant system is both realizable and robust.

An experimental platform for studying the process of toner ink fusion was modeled to demonstrate the effectiveness of the proposed method. For this system, combined optimization resulted in a performance index over 45% better than the result of optimizing the controller alone. Meanwhile, a worst-case robustness floor was maintained on several critical and uncertain system qualities.
# Table of Contents

Thesis Release Permission Form ................................................................. ii
Dedication ................................................................................................. iii
Acknowledgements ................................................................................... iv
Abstract ..................................................................................................... v
Table of Contents ..................................................................................... vi
List of Figures ........................................................................................... vii
List of Tables ............................................................................................. viii
Glossary ..................................................................................................... ix
Chapter 1  Introduction and Motivation ................................................... 1
Chapter 2  Supporting Work in Simultaneous Design ............................... 17
Chapter 3  Problem Formulation and Methodology ................................. 22
Chapter 4  Results and Discussion .......................................................... 53
Chapter 5  Conclusion and Future Extensions ......................................... 65
Bibliography .............................................................................................. 74
Appendix A: MATLAB Code .................................................................... 77
List of Figures

Figure 1: A flow chart of the proposed integrated optimization process .............. 3
Figure 2: Chirp signals as reference inputs......................................................... 12
Figure 3: Simplified schematic, render, and photograph of the SFTB ................. 24
Figure 4: Simplified schematic of the RFTB ....................................................... 25
Figure 5: Schematic of the stamp-based fuser testbed (SFTB) ......................... 33
Figure 6: Simulink model of the SFTB and controller ........................................ 50
Figure 7: Final simulation results......................................................................... 61
Figure 8: A vibrational mode is revealed within the range of test frequencies. ... 64
Figure 9: Constraining non-convex solution spaces.......................................... 67
List of Tables

Table 1: Nominal parameter values ................................................................. 36
Table 2: Mapping of parameter values .............................................................. 39
Table 3: Nominal parametric sensitivities ......................................................... 43
Table 4: Minimum and maximum parameter value constraints ....................... 47
Table 5: Parametric sensitivity ceilings for various SFTB parameters ............ 49
Table 6: Major issues encountered with the LFTBs ........................................ 59
Table 7: Parameter values before and after the optimization procedure .......... 63
Table 8: Parametric sensitivity values ............................................................... 64
Glossary

**LFTB**  Laser Fuser Test Bed. The two test platforms in RIT’s PRISM laboratory for testing the properties of the fusion of powdered toner ink to a specimen by applying heat and pressure to the interface between the two.

**RFTB**  Roller-based Fuser Test Bed. An LFTB where pressure is exerted on the specimen by compression between two counter-rotating rubber rollers.

**SFTB**  Stamp-based Fuser Test Bed. An LFTB where pressure is exerted on the specimen by rotating a camshaft, imparting force against a stationary platform.

**$H_2$, $H_\infty$ control**  Particular mathematical formulations of a control system design problem that capture both robustness and performance criteria, designed to achieve the “best” possible worst-case controller configuration.

**Convex**  (in an optimization sense) Optimization problems where local minima are also global minima and that are therefore solvable by a number of efficient mathematical techniques.
Chapter 1  Introduction and Motivation

In traditional systems engineering approaches, control systems are realized only after the completion of the design and construction of the electromechanical system to be controlled. While this offers loose coupling between designers of different disciplines and is often the easiest logistic path towards a working system, there are several benefits (and some caveats) to an integrated design paradigm that simultaneously considers both the control system and the underlying physical system. This work investigates an optimization approach to integrated design that is data-driven (based on simulation), includes both performance and robustness criteria in its formulation, and tries to effectively manage the high dimensionality of the solution space of an integrated optimization problem. By applying the method to the model of the example system used in this work, each of these goals is attained to some degree.

Feedback design in isolation is often a balancing act, seeking to find an acceptable mixture of low sensitivity to parameter variations, low susceptibility to process noise, and the desired transient and steady-state responses. Achieving each of these goals can be detrimental, if not mutually exclusive to, the others. An “optimal” system can only be obtained insofar as particular tradeoffs are accepted. When common low-order controllers – such as the ubiquitous PID (Proportional-Integral-Derivative) controller – are utilized, the designer typically only has a few independent variables with which to obtain the solution. Introducing more complex and higher order controllers gives the designer more free parameters with which to strike this balance. However, the resulting increase in the dimensionality of the problem space makes finding a globally or even locally optimal configuration challenging, and gainful hands-on system tuning becomes
more difficult. By instead giving the controls engineer influence during the mechanical
design phase, physical parameters (constrained by practical cost, manufacturing, and
other limitations) can be exploited. The resulting optimization problem’s search space
can offer a good tradeoff between the over-constrained, low-order controller-only case
and the daunting task of solving a high-dimensional, unconstrained optimization problem.

The design of control systems for a pair of experimental Laser Fuser Test Beds
(LFTBs) in RIT’s PRISM laboratory presented an ideal opportunity for this integrated
paradigm to be explored. These Laser Fuser Test Beds are used to conduct research on
the fusing process of electrophotographic systems where heat and pressure are applied to
permanently bond the powdered toner to paper. Using a variety of metrics, image quality
can then be measured to determine the relationship between the fusing parameters and the
end product. Improving the dynamic response of the fusing process improves the fuser’s
ability to consistently exert the required pressure, which has been shown to improve the
quality and consistency of the printed images [5]. In this work, mathematical models of
the LFTBs were developed, verified, and optimized using the proposed simulation-based
optimization approach. By carefully formulating an optimization problem that included
robustness constraints, the tracking error performance indices were improved by up to
45% over the non-optimized case.

Formulating a sensible controls optimization problem is difficult even when the
controller is the only independent variable. Modern approaches like $H_{\infty}$ and $H_2$
optimization techniques (further explained below) attempt to capture elements of design
robustness and performance in the problem definition, and they have been augmented to
include plant parameters through recent research. One of the most versatile formulations
is the $H\infty$ mixed-signal optimization problem where performance and robustness objectives are cast as a trade-off between different closed loop sensitivity functions. However, the above paradigm cannot include many common real-world problem constraints. Plant nonlinearities such as actuator and sensor saturation, dead bands, and backlash are not easy to include in the classical $H\infty$ optimal control formulation but can be included to some degree in its LMI (Linear Matrix Inequality) formulation [21]. Moreover, restrictions to the configuration of the controller being optimized, such as fixed order and structure, eschew many of the benefits of the technique. As a result, the gap between researchers and practitioners is still relatively wide.

What is needed is a methodology that offers the benefits of modern approaches such as $H\infty$ control and LMI optimization within a data driven (simulation-based) environment that can cope, in a unified manner, with nonlinearities, uncertain plants, and arbitrary controller architectures.

Figure 1: A flow chart of the proposed integrated optimization process. The top text in each block describes the goal of that step, while the subtext describes the specific approach used in this work.
Figure 1 describes the overall approach proposed and executed by this work. A system model is developed and verified, and a controller topology is chosen based on controls requirements. A constrained optimization routine is used to improve both the dynamic performance and robustness criteria. In utilizing any optimization routine (data-driven or otherwise), several choices must be made. An objective function must be chosen. Constraints on parameter values and other system quantities must be created. For simulation-based methods, a particular reference shape that offers sufficient coverage of expected nominal system inputs must be chosen. The exact choices for each of these components is described in detail below, but it is highly likely that any given block in Figure 1 could be replaced by another, equivalent method and the end result would be similar.

The first choice that must be made is to determine what system quantity should be minimized. In this work an integrated approach based on optimization of certain performance indices and identification based on chirp signals is developed. Performance indices are functions valued on tracking error and time; typically, lowering performance index values is indicative of improving transient response. Optimization via performance indices can offer many benefits within a framework that is easier on the implementer at the expense of solving a non convex problem, and therefore, no guarantees for global optimality. However, the lack of global optimality is not a significant disadvantage since the designer is often looking for a better or acceptable solution, not necessarily the best one mathematically possible.

Performance indices capture objectives such as transient and steady-state responses of a system in a function whose value can then be minimized, with respect to a
set of allowable parameters, by solving a nonlinear program (several examples of this are discussed in Chapter 2). These methods are well established, but they typically do not consider robustness with respect to parametric variations or noise. Furthermore, they are particularly sensitive to the choice of reference input signal, especially when mechanical parameters that may affect vibrational modes are being optimized. By choosing a sinusoidal input that is swept across the frequency band of relevant input frequencies, many of these issues can be addressed while allowing free manipulation of the system parameters and an arbitrary (but fixed) controller topology.

The process of performing mechanical and control system design simultaneously offers several benefits to systems engineers. Traditionally, the mechanical portion of a system is first designed in some way, usually to achieve favorable open-loop dynamics, material utilization, manufacturability, and other goals. The mechanical engineering design process can itself be regarded as an optimization problem, albeit one that is often implicit and solved qualitatively. The control system design is likewise an optimization problem where the objective is to achieve certain performance specifications for the given mechanical structure. This consists of finding an acceptable tradeoff between dynamic performance, controller complexity, and sensitivity. In essence, two sequential optimization problems are solved and, assuming that quantifiable objective functions can be established and evaluated, an optimal solution can be found for each of them. However these problems are in general not separable. Given that the system operates as a complete whole (mechanical structure and controller), this approach makes the assumption that the two sequential solutions achieve a globally optimum response in the system domain – an assumption that is often false.
Using optimization theory, it has been shown that certain problems are separable and will indeed yield identical results if solved sequentially or as a whole [6]. However, others – including a large portion of controls problems with nontrivial dynamics or objective functions, including the systems examined by this work – are not separable and will only yield the global optimum when solved in a holistic manner. The aim of this work was to investigate these claims with a real-world system, and to evaluate and weigh the performance benefits of a systems-level approach with the costs of the increased complexity of the optimization problem. More precisely, a pragmatic, simulation-based integrated optimization technique was introduced.

Powerful mathematical techniques for constructing robust, optimal systems do exist but are limited in their applicability to real-world systems. Nonlinearities tend not to be well handled by methods such as $H\infty$ control without additional modifications, and restricting the results of the optimization problem to an arbitrary controller topology further complicates the approach. Moreover, such problems are non-convex, and require techniques that may not be well understood by practitioners. A return to simulation-based parametric optimization alleviates these concerns, but does not consider robustness at all unless the problem is adequately formulated.

In its simplest sense, optimization is the process of minimizing some cost function (or alternatively maximizing some utility function). The main idea is to formulate an optimization problem where the satisfaction of desired performance specifications is achieved when a certain cost function is minimized. For example, if the performance specification requires good tracking, cost can be expressed as a function of the tracking error or any other “error signals”. Given a reference input, the tracking error is the
difference between the controlled variable and the desired reference signal, or in general
the difference between the actual system state and the desired state. Alternatively the
designer might need to minimize settling time, thus driving the error to within an
arbitrary small tolerance as quickly as possible. Other performance specifications such as
controller energy can likewise be minimized by a suitable choice of cost function.

When a nontrivial signal is the reference, it may be desirable to track intermediate
performance measures rather than the time it takes to achieve a certain error ceiling. For
example, the integral of the magnitude of the error signal with respect to time yields a
measure of the “closeness” of the actual output trajectory to the desired trajectory. This
is the classical integral of absolute error (IAE) performance index (and corresponds to the
one norm of the error signal). Other performance indices may use different norms (such
as the integral of squared error, or ISE, or the two norm squared) or penalize error later in
time more heavily (the integral of time multiplied by absolute or squared error, or ITAE
and ITSE, respectively). Anecdotally, minimizing these quantities often achieves an
effective control solution.

The downfall of performance index-based optimization lay in the greedy nature of
the problem formulation. Consider a prototypical system with the trivial plant $G(s)$ and
controller $C(s)$ given by Equation 1.

$$G(s) = \frac{1}{s} \quad \text{(1)}$$

$$C(s) = K$$

The unity feedback, closed-loop transfer function is therefore given by Equation
2.
Using a unit step as the reference, Equation 3 is the resulting output function.

\[
\frac{Y(s)}{X(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{K}{K + s}
\]  

Using a unit step as the reference, Equation 3 is the resulting output function.

\[
Y(s) = \frac{K}{s(K + s)}
\]  

The error is therefore the expression in Equation 4.

\[
Y(s) - X(s) = \frac{K}{s(K + s)} - \frac{1}{s}
\]

It is seen that as \(K\) grows without bound, the error approaches zero at all frequencies. Naively optimizing the value of \(K\) based solely on minimizing a function of error would therefore result in a very large \(K\). However, sensor noise would also be multiplied by \(K\), resulting in terrible noise susceptibility and very large control action. As a result, although the “optimized” controller would perform well in a noise-free environment, it would be lead to a very poor performance since there is always sensor noise in a practical implementation.

The robustness of a system can be measured in different ways and with respect to different properties. A property of a system is called robust if the property is invariant with respect to uncertainty in the system. For example, robust stability is the property of a system to maintain stability despite uncertainty in the plant and the environment. A typical measure of robustness with respect to parametric variations is the differential sensitivity function defined in Equation 5.

\[
S_p^F = \frac{\partial F}{\partial p} \frac{p}{F}
\]

Here \(F\) is a relevant system transfer function and \(p\) is a parameter of the plant or controller. The parameter \(p\) may vary (slowly) over time due to aging, operating
conditions, etc. High parametric sensitivity indicates that small variations in a parameter are reflected in the system’s output, indicating poor robustness in the presence of parametric uncertainty. Since the transfer function $F$ is in the denominator of the sensitivity quantity, high gains at all frequencies (in the forward path of the feedback loop) would therefore minimize the impact of parametric variation. However, these high gains would magnify the transfer function gain between noise and the output, since the noise passes through the controller along with the intended signal. The result is that high gains minimize parametric sensitivity, while low gains minimize input (alternatively called noise) sensitivity. A truly robust system should offer a sensible tradeoff between the two objectives.

This class of problems led to the $H\infty$ optimization problem formulated and solved in the 1980s. $H\infty$ optimization is a “minimax” technique where the controller is designed to minimize the effect of the worst allowable disturbance. The problem reduces to the minimization of the magnitude of the largest singular value of a certain transfer matrix (which is the infinity norm in the Hardy space of stable proper transfer matrices called $H\infty$). Depending on the exact formulation of the problem, this may have different meanings. One of the most common and powerful techniques is mixed sensitivity optimization, whereby performance is optimized and worst-case robustness is considered simultaneously [9]. The sensitivity and input sensitivity matrices of the closed-loop system are optimized in an $H\infty$ sense and the tradeoff between parametric sensitivity and disturbance rejection and tracking is captured by stable weighting transfer functions. The result is a stabilizing controller that exhibits an “optimal” balance of various performance
objectives. The method is also applicable to MIMO (multiple input, multiple output) systems without any modification.

While convenient in that it allows several constraints to be handled simultaneously, posing the $H_\infty$ design problem and choosing the weighting functions can be challenging, particularly when nonlinearities and controller constraints are present. Popular modern techniques include, but are not limited to solving linear matrix inequalities (LMI), transfer matrix factorization, spectral factorization, gradient descent-based search, and exhaustive search [6][11][14].

The question posed by this work was: Does there exist a way to improve performance of an integrated system using standard optimization techniques and simulations while simultaneously considering worst-case robustness? Constrained optimization is one such solution. With this approach, a time-domain simulation is performed using candidate controller and mechanical system parameters. A chosen performance index is minimized while ensuring that certain inequalities, or bounds on system metrics, are maintained. Constrained optimization has the added benefit of substantially reducing the search space but of course leads in general to non-convex problems.

Regardless of the exact problem formulation, a fundamental limitation of time-domain simulation is that only a single reference shape (ex. a step input) is considered. If performance optimization is conceptualized as a curve fitting problem of noisy data, then the main concern would be over fitting; that is, approximating not only the function but the noise as well. When only the controller is being tuned this is not a significant concern, as a first or second order controller will remain fairly general regardless of its
parameter values. In an integrated system, this becomes much more of a concern. If a reference input of a single frequency, such as a sinusoid, is chosen as the reference signal for simulations, problems may arise. It is possible that the optimization procedure would actually move a vibrational mode of the mechanical structure towards the input frequency. The system would resonate with the input and have small tracking error thus actually minimizing the desired tracking error.

However, the same tuned system would not track sinusoids of different frequencies nearly as well. Alternatively, it is possible that the system would track the input in a controlled manner, but a vibrational mode has been moved to a nearby frequency. Once the system is put online, an input of this frequency would cause large oscillations.

The over fitting problem is exacerbated by the expansion of the scope of the optimization problem to include physical parameters. Consider the high-order plant given by Equation 6.

\[
H(s) = \frac{(s - z_0)(s - z_1)...(s - z_n)}{(s - p_0)(s - p_1)...(s - p_m)}, m \geq n \tag{6}
\]

In the nominal case, the zeros \(z\) and poles \(p\) are fixed due to invariant physical constraints. When integrated optimization takes place, \(z\) and \(p\) can vary freely since they are the products of physical system parameters. As \(m\) and \(n\) above grow arbitrarily large, \(H(s)\) becomes an unconstrained function of high order, meaning that it can exactly match any reference input of a lesser degree.
Figure 2: A chirp (top). The bottom two pairs are images of Bode plots for test systems (left two images) and their corresponding magnitude responses to a chirp over the same frequency range as the bode plot. At high energy (peaks) areas of the chirp, the Bode plot is recreated.
Testing the system at multiple input frequencies ensures that such problems do not make themselves manifest. A chirped sinusoidal input, or chirp, is a periodic function with a time-varying frequency, and is one way to capture the performance of the system at multiple frequencies within a single input signal. A linear chirp signal is given by Equation 7. Such signals are of special importance in signal processing and are the basis of frequency modulated (FM) communication. The output magnitude of a system with a chirp input is in essence a portion of the system’s Bode plot at the frequencies of high energies in the chirp signal, as shown in Figure 2. By using a chirp that sweeps through the operational band of system frequencies, simulation will reveal any vibrational modes that have been moved to undesirable locations. The resulting tracking error penalizes the performance index and the optimization solver is forced to find a better solution.

\[
f(t) = \sin(\omega(t) \cdot t + \phi),
\omega(t) = t
\]

(7)

A chirp-based approach need not weigh all frequencies equally. By changing the shape of the frequency sweep, \(\omega(t)\) above, the simulation will spend more time on one particular frequency than at another. For a system that spends the majority of the time in a narrow band of frequencies, this approach may provide better results by weighing these frequencies more heavily after integrating the error. Indeed, any arbitrary continuous function \(\omega(t)\) may be used to map time to frequency.

The performance could also be changed by opting to use a delay-penalizing index such as ITAE (integral of time, absolute error) or ITSE (integral of time, squared error). With these performance indices, the error function is multiplied by the current time step before integrating; later errors have a higher time value and therefore will be penalized.
more heavily. Such performance indices are typically used with reference inputs like steps, where the goal is to attain the desired steady state as quickly as possible. However, when trying to track a “moving target”, such a scheme simply places higher priority on the location of the “target” at later time steps. For a low to high frequency sweep, this would mean that the high frequency region would be more heavily weighted in the objective function. While both approaches offer customizability, this study will focus on changing the input profile, since these are plotted by Simulink and are therefore directly visible to the end user.

The proposed technique uses performance indices to improve transient response and a chirp reference to ensure the generality of the response, but it needs additional augmentation to maintain a worst case robustness floor. Monte Carlo methods ensure robustness by randomly altering each parameter with a given variance distribution and then running the simulation multiple times [4]. As the number of permutations from the nominal system increases, Monte Carlo optimization converges towards a robust solution with high probability. The designer would only need to specify distributions for the error in each physical parameter and for noise rather than solving for potentially complicated partial derivatives. However, modern numerical software such as Maple can perform any complicated maths for a designer. An analytical method that uses robustness property inequalities as constraints on the solver has the added advantage of reducing the search space of the potentially high-dimensional optimization problem.

The task of optimizing a system’s dynamic response or parameter sensitivities is difficult, but doing both simultaneously would require a mixed objective function capturing both performance and robustness aspects. A much simpler approach is to
minimize one quantity while ensuring that the other satisfies a hard inequality. This is the approach used by DK iterations for $\mu$-synthesis in [18], as well as in a more general integrated optimization paradigm in [22]. In [18], a performance floor is enforced while robustness parameters are minimized. What is proposed here is similar to [22] where worst case robustness criteria are maintained while a performance index of the tracking error is minimized (this method is explained in greater detail in Chapter 2).

In order to formulate this type of optimization problem, it is necessary to compute the closed-form expression for each robustness quality and then provide inequality constraints to the solver. An $n$-parameter system implies that $n$ individual sensitivity equations must be determined, which shifts responsibility back to the designer by requiring sensible thresholds for each. In the domain of integrated optimization, this can be powerful. Certain parameters will be known to vary more than others. For example, the masses of rigid bodies don’t typically vary much if at all (unless travelling near the speed of light), so as long as a precise measurement is taken, sensitivity to mass parameters should not be of major concern. Rate parameters physically located near a heating element will be very likely to change with temperature and should have their associated sensitivities closely monitored. Viscous damping parameters (frictional losses) are generally difficult to calculate from first principles and must be loosely estimated; this indicates that low sensitivity to these uncertain quantities would be desirable.

Such intuition is hard to mathematically express using many techniques, but it is modeled by this method. Solving for sensitivity expressions is as simple as taking the appropriate partial derivatives of the open-loop transfer function, which can be automated
by using symbolic manipulation software. The resulting sensitivity functions can have minimax limits imposed over the same frequency range as the simulation. In other words, the maximum absolute value of each sensitivity function over the operable range of input frequencies must not exceed the pre-determined threshold.

The application of a priori system knowledge to the optimization problem is the pillar upon which the claim is made that this method may achieve sensible design tradeoffs. Generalized $H_\infty$ approaches lack this facet, so while they may have desirable generality across a variety of problems, they fail to exploit constraints that may result in a better specific solution.
Chapter 2  
Supporting Work in Simultaneous Design

Much of the existing research in the paradigm of simultaneous control system and mechanical structure design introduces a method for solving the resultant optimization problem by some mathematical process. In some cases, numerical examples are given. Because of the lack of guarantees that come with a simulation-based optimization technique (i.e. non-convexity, a lack of generality, and potential numerical and computational issues) the majority of research focuses on analytic and symbolic optimization. $H\infty$ optimization is particularly well-represented in the modern literature, as it is a powerful way to apply some of the previously discussed robustness constraints under the umbrella of a single problem with a well-understood multivariate objective function. The method used in this work actually harkens back to much earlier studies.

The earliest work in this field comes from the aerospace industry in the 1980s and the robotics industry in the 1990s. High speed positioning systems, such as pick and place machines for PCB manufacturing, represent applications where the performance benefits of an integrated approach are highly desirable. Performance gains in settling time in this case are directly correlated with increased throughput and therefore decreased cost. In [15], a two-link robotic arm’s dynamic performance was optimized by finding optimal parameters from given mechanical topologies and control laws. This study used a somewhat ad-hoc approach, analyzing the symbolic expressions for each controller gain and then moving system poles in the mechanical system to help achieve optimal performance. While this method did not perform mathematical optimization, it did show
by example that integrated design could offer substantial performance benefits to systems designers.

Over time, more studies were devoted to formal methods of integrated optimization as well as the underlying mathematical basis of these problems. [6] examined the problem space of the controls and mechanical optimization problem and determined that simple tests could determine whether or not a simultaneous design approach would yield a better solution than a sequential one. Numerical examples of several types of system yielded results that supported the mathematical distinction. What was most interesting was that a given system topology could be both decomposable (meaning that a sequential approach would provide the same optimum as a simultaneous one) or non-decomposable based on the nature of the optimization problem cost functional.

[20] also considered the optimality conditions of various approaches to simultaneous design. The authors discussed not only the sequential and integrated cases, but also the iterative case where the plant and controller are tuned in turn. Once again, it was revealed that only simultaneous design always achieved the best performance. [16] demonstrated that controllers with feed forward elements are interacting in an optimization sense and the feed forward and feedback elements may not be optimized separately. While this work did not make use of feed forward control, this is a logical addendum to the above distinction.

Numerous other works have examined the nature of the optimization problem that must be solved. Some papers have proposed the use of linear matrix inequalities (LMIs) to solve $H_\infty$ optimization problems. Formulating a problem with LMIs leads to a
semidefinite programming problem for which efficient numerical techniques have been developed. In [11], an iterative LMI-based approach is introduced where “controller effort” was the quantity to be minimized. Numerical examples showed that mechanical redesign could sometimes reduce this value by an order of magnitude or more. In [14] and [21], the system was also solved using LMIs.

Work has also been done on optimizing control under certain pole placement constraints (usually to prevent undesirable vibrational modes in the resulting mechanical system). Such an approach handles directly what this work indirectly considers by the use of a chirp input signal. [1] used an ad-hoc optimization function consisting of a linear sum of various performance and cost metrics with arbitrary weighting factors, which is easily solvable with tools like MATLAB’s Optimization Toolbox.

One very interesting and intuitive approach to simultaneous design heavily utilizes parametric sensitivities. [13] examined the sensitivity of the objective function to be minimized (in this case the $H_2$ norm of certain transfer matrices). This provides something of a “tuning map” for the individual system parameters. Such information allows for manual tuning of the system. Also, this information can be used to help restrict the size of the search space or supply good initial “guesses” in solving the optimization problem numerically. Other works use various parametric sensitivities as constraints on the $H_2$ or $H_\infty$ optimization problems [7]. These approaches are effective since they ensure that robustness is included in the optimized system. Moreover, the sensitivities can help formulate an intuitive interpretation of the results of the optimization process. Parameters with high sensitivities are likely to be most affected by optimization.
Several papers offered approaches similar to the one proposed here, but with slight variations. In [17], a crystallization process and its control system are optimized by using transient response optimization along with metric inequalities. The constraints are considered by restricting the area of the complex plane where system poles are allowed to be placed. This achieves the same effect as constraining linear and nonlinear functions of system parameters as in the proposed method, albeit with a more intuitive and explicit representation in the frequency domain. [21] used parameter constraints directly in the expression of their multivariate optimization problem (which was solved using a homotopy-like LMI method). It also used constraints on parametric and input sensitivities by trying to minimize the effect of perturbations on the minimized multivariate performance index (rather than minimizing their effect on the plant transfer function). Lastly, [22] offered an algebraic approach whereby a performance index is minimized while maintaining a worst-case robustness floor. The only conceptual difference between that study and this one is in how the robustness expression was obtained; in the former case, a multivariate norm was created to represent all of the robustness criteria. In this case, each sensitivity is constrained individually based on knowledge of parameter uncertainty.

Two notable omissions are noticed when examining the current body of research in integrated design and optimization. First, direct comparisons between differing methods are few and far between. Many papers offer ground truth controllers and then show the advantages of using an integrated optimization technique, but don’t consider other optimization techniques. This is most likely due to the fact that direct comparative
metrics between disparate optimization techniques are difficult to formulate. Only in the case of an identical objective function can conclusive comparison be conducted.

Second, most (but not all) studies offer a single numerical example of the technique applied. Or, several studies will be offered, but they will be “toy” problems of a trivial scale. The issue here is that it is hard to make general statements about a controls process that applies to a wide variety of non-trivial plants. Stiff and non-stiff systems often respond better or worse to a given approach and the various types of common nonlinearities are better accounted for by one technique than by another. The implication is that the approach proposed and explored by the work here is valid; since no single optimization technique has been proven to be applicable to all classes of controls problems, a potential lack of generality is not fatal to an effective controls technique.
Chapter 3  Problem Formulation and Methodology

The actual systems with which the integrated design approach was tested were a pair of Laser Fuser Test Beds (LFTBs) that apply heat and pressure to a paper sample to fix and fuse powdered toner ink, as in a laser printer. Fusing is the last step of the laser printing process, but one that has significant effects on the quality of the printed image [12]. Furthermore, because the temperatures inside the fuser can exceed 150°C, the majority of the power consumed by a commercial laser printer is during this stage. The temperature requirements are reflected in the overall cost of ownership of a laser printer both in direct energy consumption as well as in the production considerations that must be taken to ensure that excess heat is dissipated in an appropriate manner. One common strategy for minimizing energy requirements is the use of clever power management schemes. Other strategies include the use of smaller heating chambers, therefore minimizing the volume to be heated by utilizing new structural design techniques. However, for a given toner chemistry, all of these optimizations and more can only approach a certain limit of energy reduction [8].

New types of toner are being developed to try to reduce these requirements, but high-fidelity testing platforms are necessary to establish the empirical properties of the toner fusing process in order to design commercial fusers that are optimal from image quality, robustness, and cost standpoints. The LFTBs in RIT’s PRISM laboratory must provide accurate pressure profile tracking in order to remove this uncertainty from the imaging experiments that will be conducted. A study of the effect of toner fusing parameters on gloss (which is one measurable metric of image quality) found that the maximum pressure applied and duration of pressure application accounted for over 40%
of the gloss variation in the output for a constant toner type [5]. This underscores the importance of the performance of the LFTB pressure application systems.

Two LFTBs were available for testing. The first to be constructed was the Stamp-based Fuser Test Bed (SFTB), which uses a rotating cam profile to apply compressive force downwards into the stamp, where heat is applied and toner is fused. This system allows for different shapes in the pressure profile by varying either the cam profile or the trajectory of the stepper motor driving the cam. However, it has no intrinsic feed mechanism so only small areas of toner may be fused at once. A quadrature shaft encoder provides position feedback of the cam, while a load cell provides measurement of the applied force. The existing motor driving circuitry provides rudimentary position control functionality for the motor, but the goal of the control system is to close the loop between the load cell and the motor. For the most part, the pressure profiles that must be tracked will be sinusoidal or nearly sinusoidal. Relating the sinusoidal profile of the cam back to the motor position, this actually means that a constant motor velocity (or a position ramp) reference of finite duration is the most common input.
Figure 3: Simplified schematic, render, and photograph of the SFTB
The Roller-based Fuser Test Bed (RFTB) more closely resembles a laser printer’s fuser. Here, two linear actuators compress two rollers together. One roller is heated while the other is rotated to act as a feed mechanism. Two load cells – one on each side of the rollers – provide force feedback, so this system is not SISO (single input, single output) (though the left and right assemblies are nearly identical). Potentiometers provide position feedback and control for the linear actuators. In this case, the transient response of the system is not as important as in the SFTB case, as the pressure is generally held constant while the media is passing through the rollers. The major performance goal of this system was therefore disturbance rejection in the steady state.

![Simplified schematic of the RFTB](image)

**Figure 4: Simplified schematic of the RFTB**

The first task for this work was to model the LFTBs. This was accomplished by deriving the systems’ dynamic equations from first principles. Individual parameter values were obtained from either first principles or from empirical measurements, or both. Some quantities, such as the rate constants of commercial springs, could easily be
obtained from product documentation. Others, such as viscous damping coefficients, were impossible to accurately estimate from geometry and physical properties alone, and had to be estimated from specially designed experiments. The models themselves were described in state-space form, so as to allow computational tools and simulators such as MATLAB and Simulink to represent the system in the time domain. The SFTB is a SISO system with a single actuator and single output sensor, so frequency-domain tools were used as well. The RFTB has a pair of actuators, and so was represented by two identical and symmetric transfer functions for simulation and analysis. Interaction between the two sides of the RFTB was not considered (as it was found to be negligible within the normal operating window of the system).

The actual systems are necessarily nonlinear. The SFTB in particular has a periodic, nonlinear function relating motor position to linear displacement because of the cam. The RFTB circumvents this by using linear actuators. However, both systems are subject to parametric variation due to the thermal effects of the heating elements. The magnitudes of such effects were unknown a priori, but were suspected to be minor. The proposed optimization method captured this uncertainty by limiting maximum sensitivity values to ensure that variations of susceptible physical parameters are not strongly reflected in the output.

An accurate model of the transducer used to sense pressure on the sample is just as important as the model of the physical system. Load cells were installed on both the RFTB and SFTB, but they differed in their internal configuration. The SFTB load cell is capacitive and its output slowly decays to zero (i.e. the transfer function has a zero at zero) given a constant input. This affect the observability of the overall system, as the
concept of steady-state is lost given such a sensor. This did not present a major issue, since the SFTB will not be expected to maintain a constant force for more than a few seconds, which proved to be significantly less than the load cell’s time constant. As a result, no steady-state analysis was conducted for the SFTB. The RFTB load cells are resistive and can accurately sense constant force, which will be expected of them for that particular fuser design.

Once the model had been verified by comparing simulated results with those of the real system, the parametric sensitivities could be derived. As previously defined, sensitivity $S$ of a function $F$ with respect to a parameter $p$ is defined by the percentage variation in the function divided by the percentage variation in the parameter. As the percentage change in the parameter asymptotically approaches zero, this reduces to the expression in Equation 5.

Thus, a symbolic partial derivative of the system transfer function was taken with respect to each of the parameters. This information shows some of the robustness properties of the system. Parameters with high sensitivities affect the system response to a high degree when varied even slightly about their nominal point. This information yielded two conclusions. First, it showed which mechanical properties may be most effectively altered in order to change the system response. Second, it showed which parameters’ thermal variations were most important to capture, whether by uncertainty modeling or by adding new terms to the system representation. Finally, the closed-form sensitivity expressions were used to constrain the final optimization problem.

Sensitivity of a final design is usually desired to be low with respect to all parameters so that nominal variations in a physical property do not significantly affect the
output. Closed-loop feedback usually reduces sensitivity. If necessary, the simultaneous
design paradigm also allows sensitivities to be reduced mechanically. The partial
derivative curve can be shifted to a more favorable region simply by modifying physical
properties.

Noise susceptibility was modeled as the transfer function between disturbances
and the output. It was addressed by adding stochastic Gaussian noise (of variance $3 \text{ Pa}^2$) to the sensor output. Although the noise was generated randomly, the same seed was used for each iteration within a given optimization problem so as not to confuse the gradient-based solver (and to provide repeatability to the experiments).

When conducting the search over the parameter space of the system, it was
desirable to constrain the dimensionality of the solution space as much as possible. To this end, sweeps across potential parameter values were not unbounded in both directions. In particular, each of the three types of elements in the physical model – masses, spring constants, and viscous damping coefficients – was constrained uniquely. Rigid masses were not to be larger than the mass of the largest volume of material that would fit within the physical LFTB structure. They were also not to be smaller than that of a volume of material which would undergo plastic deformation under compression. Spring rates were limited to those available commercially in springs that fit the chosen form factors. As the LFTBs are maintained at a well-lubricated state, viscous damping coefficients were forbidden to be decreased from their nominal value, though they were allowed to increase without bound. This stemmed from the assertion that friction can always be increased, but not easily reduced below a certain level.
When considering control systems, it is important to remember that the ultimate goal is not necessarily a perfect mathematic solution but rather one that achieves the desired response of the plant-controller combination. Achieving optimal transient response characteristics will not translate into increased system performance when transient response is not an important system characteristic. Because of this, the RFTB’s control solution was designed using classical controls techniques. This meant that the pressure-actuation loop could be closed more quickly so that (1) testing the effect of fuser parameters on a sample could be done quickly and (2) time was saved for a more in-depth integrated optimization procedure to be used with the SFTB.

Recall that the RFTB’s concept of operations called for the rollers to be compressed together to achieve the desired pressure prior to introducing the sample into the system. Thus, traditional measures of transient response such as rise time and settling time were not critical (as long as the proper pressure could be achieved within a reasonable window of a couple seconds so the operator does not have an excessive wait before he can begin fusing). Rather, it was the elimination of steady-state error and disturbances caused by the sample and roller imperfections that formed the goals of the controller.

Designing a controller to eliminate certain types of steady-state error and disturbances is a matter of ensuring that there are a sufficient number of integrators present in the feedback loop. As integrators are added, higher-order input signals can be tracked and higher-order disturbances can be eliminated with either zero or constant steady state error. Classically, systems are assigned a “type” number to denote how many integrators they possess and therefore what sorts of signals can be tracked without
error. Type 0 systems contain no integrators and achieve finite steady-state error for step inputs, but infinite error for ramp or parabolic inputs. Type 1 systems have a single integrator allowing for the elimination of steady-state error for steps, finite error for ramps, and infinite error for parabolic references. Each additional integrator adds to the maximum order of the reference signal that can be tracked without error. However, additional integrators also hurt stability, as additional poles at the origin of the complex plane narrow the region of stabilizing controllers.

The RFTB’s normal usage scenario involved moving the rollers to achieve a constant pressure level. This is a step input, and requires a type 1 system to operate without steady-state error. Two types of disturbances were predicted. One was the increase in pressure once the sample had been inserted into the roller interface (a step disturbance). The other was small imperfections in the rollers themselves. These disturbances were suspected to be minor, and would be periodic artifacts of the roller rotation. Moreover, the compressible nature of the rubber rollers would act as a dampener for these disturbances. Thus they did not enter into the consideration of system type.

A “napkin sketch” model of the RFTB was developed to determine the system type of the plant. As a linear sequence of masses, springs, and dampers (friction), the RFTB was necessarily a type 0 system. Thus, a single integrator had to be present in the controller. Hardware PID control circuitry wired to the RFTB’s linear actuator would provide this integration (which was highly preferably to the alternative of doing software control via LabView through the National Instruments test bench setup from a delay standpoint). Although the PID controllers were designed to do position control based on
a built-in potentiometer, they could use any properly scaled analog signal as an input. This meant that closing the loop was as simple as wiring the load cell’s analog output (after low-pass filtering and proper scaling) to the circuit’s feedback input. The reference command was generated by the LabView user interface and sent to the circuit over an RS-232 serial connection.

The LabView software offered the user an interface for viewing various system measurements as well as controlling both the pressure and temperature outputs. A “slider bar” UI component allowed the user to select a desired pressure and then send the command to the motor controller. The pressure selected by the user (in PSI) was first scaled and shifted appropriately to represent a desired load cell voltage for the controller. The hardware then took over, driving the motors until the error between reference and input had been eliminated.

Tuning the PID controller was conducted using standard industry practiced techniques. The integral and derivative gains (Ki and Kd, respectively) were set to zero and the proportional gain (Kp) was tuned by hand. Once the proportional-only controller was stable and settling at a finite steady-state error, the integral gain was slowly increased until the steady state error was eliminated. Tuning Kp and Ki then continued iteratively until the desired response had been achieved. The derivative gain remained zero since the system was stable without it (Kd contributes primarily to transient response and stability, so it wasn’t useful here). Once logistical concerns such as wiring, the RS-232 communication protocol, and sign conventions were addressed, tuning took less than 30 minutes for the final result to be produced. Pressure could be controller to within the
nearest .5 PSI, and the remaining error was a result of noise in the sensor and backlash in the linear actuators. This was deemed more than adequate for fusing with the RFTB.

Some additional concerns were discovered and addressed in the process of implementing the RFTB control system. These are further discussed in Chapter 4.

The SFTB’s concept of operations dealt almost exclusively with transient response. In order to achieve a high level of performance in that respect, an integrated parametric optimization approach was utilized. The first step was to obtain a model of the system so that optimization could take place in simulation. The SFTB was modeled in state-space form by examining a simplified rigid body diagram and deriving the necessary dynamic equations. The diagram is given in Fig. 4.
Figure 5: Schematic of the stamp-based fuser testbed (SFTB). Masses m1, m2, and m3 represent the body at the top of the large spring, the stamp, and the platform above the load cell, respectively. Spring rates k0, k12, ks, kc, and kt represent the flex of the cam shaft, the large spring, the small springs supporting the stamp, the load cell, and the compression of the sample, respectively. Damping constants b1, b2, and bt represent the friction of the motion of the mass at the top of the spring, the stamp at the bottom of the spring, and the damping of the sample, respectively. The system transfer function relates the displacement of m1 with the pressure exerted on the sample by kt.

Note that Fig. 4 presents separate bodies attached by rigid connections (welds, bolts, and adhesives) as single masses for the purposes of computation. The initial model of the SFTB included deformation of the masses due to compression as additional spring and damping constants [19]. However, given that the majority of the SFTB is made of aluminum, these constants were much larger than any of the others in the system and resulted in numerical problems in the model. Ignoring rigid body deformation both
simplifies the dynamic equations and results in a non-singular model. Additionally, exact deformation constants are difficult to obtain experimentally and add a high degree of uncertainty to the model when determined from first principles.

The Newtonian dynamic equations of the SFTB follow in Equation 8. Note that positive displacement is downwards for the purposes of this model. Linear force models are used. \( x_1, x_2, \) and \( x_3 \) represent the downwards displacements of the three identified masses, respectively.

\[
\begin{align*}
    m_1 \ddot{x}_1 &= f(t) + m_1 g - k_0 x_1 - b_1 \dot{x}_1 - k_{12} (x_1 - x_2) \\
    m_2 \ddot{x}_2 &= m_2 g - k_1 x_2 - b_2 \dot{x}_2 - k_{12} (x_2 - x_1) - k_2 (x_2 - x_3) - b_1 (\dot{x}_1 - \dot{x}_2) \\
    m_3 \ddot{x}_3 &= m_3 g - k_3 x_3 - k_2 (x_3 - x_2) - b_1 (\dot{x}_1 - \dot{x}_2)
\end{align*}
\]  

Equation 8

This system of equations represents three 2\textsuperscript{nd} order differential equations, or alternatively one 6\textsuperscript{th} order ordinary differential equation. The nominal state vector is therefore given by Equation 9.

\[
\bar{x} = [x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3]^T
\]  

Equation 9

The state matrix (\( A \) in standard state-space nomenclature) is given by Equation 10.
\[
A = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
-\frac{k_0 + k_{12}}{m_1} & \frac{k_{12}}{m_1} & 0 & -\frac{b_1}{m_1} & 0 & 0 \\
\frac{k_{12}}{m_2} & -\frac{k_{12} + k_x + k_r}{m_2} & \frac{k_r}{m_2} & 0 & -\frac{b_2 + b_r}{m_2} & \frac{b_r}{m_2} \\
0 & \frac{k_r}{m_3} & -\frac{k_r + k_r}{m_3} & 0 & \frac{b_r}{m_3} & -\frac{b_r}{m_3}
\end{bmatrix}
\] (10)

The input matrix \((B\) in standard state-space nomenclature) is given by Equation 11.

\[
B = \begin{bmatrix} 0 & 0 & 0 & k_0/m_1 & 0 & 0 \end{bmatrix}^T
\] (11)

The output matrix \((C\) in standard state-space nomenclature) is given by Equation 12.

\[
C = \begin{bmatrix} 0 & 0 & k_r/CS & 0 & 0 & 0 \end{bmatrix}
\] (12)

Note that \(CS\) above indicates the cross-sectional area of the sample within the fuser. It is acknowledged that there is some uncertainty in this quantity, and the area may in fact be a time-varying parameter given the deformation of the sample under pressure.

There is no direct coupling component to this system, so the \(D\) matrix is simply zero. The effect of gravity on the system was not modeled since it is effectively an offset that can be subtracted from the input command.

Since the load cell in the SFTB is capacitive, a model for the sensor was derived as well. Because the cell’s output goes to zero at steady state, it is known that its transfer function has a zero at the origin. The location of the complementary pole was determined
by examining the response of the sensor under a known input load (a 1 kg mass was
added and the decay of the output was observed). The final model of the sensor is given
by Equation 13.

\[ H(s) = K \frac{s}{s + 0.1} \quad (13) \]

\( K \) represents the gain of the sensor, which is unimportant as it can be corrected by
a constant scaling factor in the controller. For simulation purposes, it was set to unity.

To ensure that the model accurately captured the salient behaviors and
characteristics of the SFTB, it was important to establish both that the topology of the
model and the values of its parameters were good representations of the actual system.

Before the topology could be verified, the parameters must be reasonable.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Nominal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1 (kg)</td>
<td>.280</td>
</tr>
<tr>
<td>m2 (kg)</td>
<td>1.700</td>
</tr>
<tr>
<td>m3 (kg)</td>
<td>.4977</td>
</tr>
<tr>
<td>k0 (N/m)</td>
<td>60577</td>
</tr>
<tr>
<td>k12 (N/m)</td>
<td>17512</td>
</tr>
<tr>
<td>ks (N/m)</td>
<td>630</td>
</tr>
<tr>
<td>kt (N/m)</td>
<td>1E9</td>
</tr>
<tr>
<td>kc (N/m)</td>
<td>1.05E9</td>
</tr>
<tr>
<td>b1 (Ns/m)</td>
<td>100</td>
</tr>
<tr>
<td>b2 (Ns/m)</td>
<td>1000</td>
</tr>
<tr>
<td>bt (Ns/m)</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Nominal parameter values
Table 1 shows the nominal parameters established for the SFTB. The masses were measured with a scale in kilograms. The spring constants were taken directly from the spring specifications for $k_{12}$ and $k_s$. $k_0$, the flex of the cam shaft, was obtained from first principles by computing the elastic flex of a prismatic beam made of hardened steel (AISI 1040 Steel) [17]. This yielded the calculations in Equation 14.

$$k_0 = \frac{48EI}{L^3}$$
$$I = \frac{\pi d^4}{64}$$
$$d = 0.5\text{in}$$
$$L = 0.5\text{in}$$
$$E = 200\text{GPa}$$
$$k_0 = 60,577\text{N/m}$$

$k_c$, the spring rate of the load cell, was determined from product documentation. The remaining spring constant, $k_t$, for the sample was estimated to within a power of ten from studies of toner properties [5]. Likewise, the viscous damping coefficients were all estimates, as there are neither straightforward methods for determining them from first principles nor suitable experimental methods to determine them to high precision with the measurement equipment available. As a result, the sensitivities of these parameters were closely monitored during optimization.

With reasonable parameter guesses in place, the overall model could be checked for consistency and accuracy against measured outputs. Actual output from the SFTB was captured and compared side-by-side against the output of the model. It was at this point that one of the issues described later in Chapter 4 was discovered. For a given
collection of test data, the sensor gain in $H(s)$ could be tuned so the model matched the shape of the actual output curve. Actual errors in the dynamics outside of this scaling factor were not present. However, repeating the model verification step with test data gathered from a different run of the SFTB revealed that a completely different gain factor was needed. The gain of the load cell was not changing; rather, because the load mechanism was locked into place by hand, slight variations in vertical alignment along with variable tightness of the threads that locked the vertical position in place resulted in different gains. As a result, verifying actual gain for the system could only happen for a given configuration of the SFTB that would be very difficult to re-obtain. Still, after compensating for this factor, the dynamics of the model closely mimicked those of the actual system once the overall gain constant had been matched.

In order to be able to optimize the LFTB’s parameters while considering parametric robustness, parametric sensitivity had to be derived from the system model. Parametric sensitivity expressions can be calculated by taking the partial derivative of the open-loop transfer function with respect to each parameter (Equation 5). The partial derivative is then multiplied by the parameter’s value and divided by the system transfer function itself. The resulting expression is a function of complex frequency that can be evaluated at any complex operating point $s = j\omega$. Within the scope of the problem formulation used here, the maximum sensitivity within the range of useful input frequencies is the important quantity. One way to find the maximum would be to take the derivative with respect to $s$ and then find the zero crossings. However, this would have to be done for each parameter at for each function evaluation of the optimization procedure, slowing down the process significantly.
A much faster solution was to calculate a numerical frequency response for each sensitivity function and take the maximum absolute value instead. If the frequency sweep is of a sufficiently high resolution, this method provides nearly identical results as the algebraic case. Custom software was written to provide this functionality for the optimization procedure. The code to do this (using MATLAB 7.1 with the Symbolic Math and Control System Toolboxes) is explained below.

The custom MATLAB function `deriveSensitivity(params, fmin, fmax)` was designed to be called given an array of current system parameters as well as a frequency range over which to evaluate the system transfer function. `params` represents a vector of the variable system parameters (with an encoding given by Table 2 below) for the holistically optimized case. `fmin` and `fmax` represent the minimum and maximum system frequencies used in the chirp sweep (for example, the SFTB was designed to operate with profiles between 20 milliseconds and one second long, so this range was set to 1-50 Hz).

<table>
<thead>
<tr>
<th>Index into params</th>
<th>Meaning (see Fig. 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m1 (kg)</td>
</tr>
<tr>
<td>2</td>
<td>m2 (kg)</td>
</tr>
<tr>
<td>3</td>
<td>m3 (kg)</td>
</tr>
<tr>
<td>4</td>
<td>ks (N/m)</td>
</tr>
<tr>
<td>5</td>
<td>k12 (N/m)</td>
</tr>
<tr>
<td>6</td>
<td>b1 (Ns/m)</td>
</tr>
<tr>
<td>7</td>
<td>b2 (Ns/m)</td>
</tr>
</tbody>
</table>

Table 2: Mapping of actual parameter values to the `params` array used throughout the code for the integrated optimization case. Note that MATLAB arrays start at index “1”.

To derive the sensitivity, a MATLAB state-space representation of the LFTB with a given set of parameters must first be created. This was problematic because the Control System Toolbox’s useful state-space functions do not accept symbolic inputs, so certain
necessary functionality had to be re-written to support symbols. The state-space A, B, C, and D matrices were created using the expressions from Equations 10-12, but without substituting values for each of the parameters (e.g. by first declaring `syms m1 m2 m3 ks k0 k12 kc b1 b2 kt bt csa`).

When using normal Control Systems Toolbox data structures, converting between state-space form and transfer function form is as easy as calling the function `tf(Gss)` with the state space model as the argument. Since the four symbolic matrices were not represented as a state-space object, the conversion had to be done explicitly. For a SISO system, the transfer function can be obtained from the state-space matrices through Equation 15.

\[ G(s) = C(sI - A)^{-1}B + D \quad (15) \]

After declaring the `s` operator as a symbol, this was accomplished in code with the line:

```matlab
G=C*(s*eye(size(A))-A)^-1*B + D;
```

The next step is to obtain the partial derivatives with respect to all of the parameters of interest. Since multiple partial derivatives were needed, the transfer function’s Jacobian was computed (the Jacobian is a matrix of every partial derivative of a given function):

```matlab
partials = jacobian(G, [ks k0 k12 kc b1 b2 kt bt csa]);
```

Note that the complete Jacobian was not calculated; this is because under the specific problem formulation further described below, some sensitivities were unconstrained.
Partial derivatives are not sensitivity functions. To complete the calculation, the columns of the Jacobian were each multiplied by the appropriate parameter and then divided by the original transfer function.

```matlab
sens = partials.*[ ks k0 k12 kc b1 b2 kt bt]/G;
sens = simplify(sens);
```

Algebraic simplification was done at this point because by performing reduction and combining like terms, the final sensitivity functions can be evaluated by substitution much quicker within the optimization iterations (by default the Symbolic Math Toolbox is careful never to simplify implicitly in order to avoid possible numerical ramifications). After this step, `sens` is a 1x8 array of symbolic parametric sensitivity functions (each spanning several pages of expressions, even after simplification).

The final step was to evaluate the transfer functions at the operating point and find the maximum responses. This was accomplished once for each of the appropriate parameters (the “j” variable seen below is the index of the parameter into the array specified by Table 2).

```matlab
fullparams = [params(1:4)', 60577, params(5), 1.05E9, params(6:7)', 1E9, 1, .001];
subst = subs(sens(j),[m1 m2 m3 ks k0 k12 kc b1 b2 kt bt csa],fullparams);
```

The `subs` function evaluates the function given by the first argument with the symbols in the second argument substituted by the corresponding values in the third. After this step, `subst` is entirely numeric except for the Laplace operator $s$.

We now wish to perform a frequency sweep over the input range in order to find the maximum response. With proper Control Systems Toolbox objects, this can be
accomplished through the `freqresp` function, which takes a system and an array of frequencies and outputs an array of the corresponding response magnitudes. While this functionality could be duplicated by iteratively calling `subs` with different complex frequencies, symbolic substitution is quite slow using MATLAB’s underlying Maple engine. A better technique would be to convert the symbolic expression `subst` into a proper transfer function object and then using the built-in `freqresp` function. Numerous third-party functions to do this are freely available online, and an implementation by Joerg J. Buchholz of Hochschule Bremen in Germany was leveraged (supplying the `sym2tf` function).

```matlab
wmin = fmin*pi/180;
wmax = fmax*pi/180;
S(j) = freqresp(sym2tf(subst),[wmin:.01:wmax]);
S(j) = max(abs(squeeze(S(j))));
```

The first two lines convert the frequency range into radians per second. Next, the frequency response is taken between the two ends of the range. The “step” factor of 0.01 chosen here was chosen to provide a good tradeoff between accuracy and simulation time. Finally, the maximum magnitude of the response was found. After iterating through all eight (in this case) parametric sensitivity functions, `S` contained a 1x8 matrix of the maximum sensitivity values.

Inspecting the outputs of the `deriveSensitivity` function can help to paint an intuitive picture of the relationships between parameters and the system response. For the nominal system configuration, these maximum sensitivities are given in Table 3.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Nominal Maximum Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ks (Small springs)</td>
<td>1.2300 e-006</td>
</tr>
<tr>
<td>k0 (Cam shaft deformation)</td>
<td>0.2243</td>
</tr>
<tr>
<td>k12 (Large spring)</td>
<td>0.7757</td>
</tr>
<tr>
<td>kc (Load cell deflection)</td>
<td>1.3561 e-005</td>
</tr>
<tr>
<td>b1 (Lever damping)</td>
<td>0.0011</td>
</tr>
<tr>
<td>b2 (Stamp damping)</td>
<td>1.6936 e-006</td>
</tr>
<tr>
<td>kt (Specimen spring constant)</td>
<td>1.4240 e-005</td>
</tr>
<tr>
<td>bt (Specimen damping constant)</td>
<td>1.2352 e-014</td>
</tr>
</tbody>
</table>

Table 3: Nominal parametric sensitivities of important system quantities (maximum value on the frequency range 1-50 Hz)

Several conclusions can be made by analyzing this data. First, the large spring (through which all force from the cam input to the load cell output must be transferred) is by far the most influential parameter to the system response. Changes to this parameter by the optimization routine will likely yield major changes to the final response. Likewise, the deformation of the cam shaft is also a relatively sensitive quantity. It is noticed that the specimen dynamics are of relatively little importance to the response, which is fortunate given the comparative uncertainty in their values. However, these sensitivities must still be monitored during optimization to ensure that they do not get amplified through changes elsewhere.

The `deriveSensitivity` function not only allowed for the derivation of the parametric sensitivity functions for a given configuration of the SFTB; it also doubled as the basis of the nonlinear constraint function used for the optimization procedure detailed below. By wrapping the outputs of this function with a format that MATLAB’s Optimization Toolbox expected and imposing particular constraints on each value of the $S$ array, nonlinear inequalities were created that the solver had to satisfy in order to accept a given solution.
The framework for robustness-constrained performance index optimization has been established, but the exact formulation of the problem must first be developed before it can be solved.

What is desired is an optimization algorithm that can minimize a cost functional while maintaining several hard problem constraints. First, lower and upper bounds are present on many of the parameters. These express the feasibility of affecting the changes of the optimization procedure as well as enforcing physical parameter interpretations for the purely mathematical solver (for example, masses must be constrained to be positive). Second, several parameters’ sensitivities must be constrained through inequalities. For non-linear, constrained optimization, the MATLAB Optimization Toolbox offers the \texttt{fmincon} function.

\texttt{fmincon} is actually capable of solving far more complicated types of problems than the one here, so several of its capabilities will be ignored. Like most of MATLAB’s solvers, \texttt{fmincon} can utilize several different optimization algorithms to achieve a solution. The choice of algorithm is driven by the size of the problem, the convexity of the search space, and the presence or absence of user-supplied gradient information. For this problem, a medium-scale line search algorithm was selected because it did not require any additional inputs (such as Hessian matrices for the nonlinear inequality constraints). This algorithm is based on gradient descent, computing local second derivatives (the Hessian of the Lagrangian) through finite differencing to find the path of steepest descent towards the local minimum. For convex problems, the local minimum is also the global minimum. For non-convex problems, such as the nonlinear optimization problem modeled here, care must be taken to ensure that convergence occurs in a suitable
“basin”. This can be accomplished by first manually tuning the system so that all of its
performance metrics meet some worst case floor. Optimization then occurs from this
starting point and the performance will only get better.

Formally, \texttt{fmincon} attempts to solve a problem of the form given by Equation
16 according to MATLAB Optimization Toolbox documentation.

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad c(x) \leq 0 \\
& \quad \text{ceq}(x) = 0 \\
& \quad A \cdot x \leq b \\
& \quad A_{eq} \cdot x = b_{eq} \\
& \quad lb \leq x \leq ub
\end{align*}
\]  \tag{16}

With \( f(x) \) as the performance index cost functional to be minimized, \( x \) represents
the vector of independent variables (system parameters to be tuned). \texttt{fmincon} accepts
three different types of problem constraints: parameter bounds, linear equalities and
inequalities, and nonlinear equalities and inequalities. For this problem, linear equalities
are unnecessary since the sensitivity constraints are necessarily nonlinear (because they
utilize both maximum and absolute value functions). Parameter bounds to \texttt{fmincon} are
supplied simply as vectors (with \(-\text{Inf}\) and \(\text{Inf}\) indicating no effective lower and upper
bound, respectively). Nonlinear sensitivity constraints are supplied in the form of a
function callback that evaluates the constraint condition at a given operating point and
returns a vector of the result (where non-positive results indicate that the inequality has
been met).

For this optimization problem, the formal formulation is therefore Equation 17.

\[
\begin{align*}
\text{minimize} & \quad \text{IAE}(f(s; \hat{x})) \\
\text{subject to} & \quad \left(\max_s S(s)_{x_i}\right) \leq \zeta_i \\
& \quad x_i \leq T_i \quad \text{for all} \quad x_i \in \hat{x} \\
& \quad x_i \geq t_i
\end{align*}
\]  \tag{17}
Here, $IAE(s; \hat{x})$ represents the integral of absolute error performance index of the frequency-domain system with the parameter vector $\hat{x}$. $t_i$ and $T_i$ represent the lower and upper bounds for each parameter $x_i$ in $\hat{x}$, respectively. $\zeta_i$ represents the vector of corresponding parametric sensitivity thresholds.

For each independent variable, upper and lower boundaries were considered. Of course, some parameters (e.g. controller gains) had no theoretical boundaries. Table 4 summarizes the boundaries given to the solver for this problem (“Inf” indicates a boundary of infinity, so these parameters are essentially unbounded).

The three masses $m1$, $m2$, and $m3$ were constrained based on the geometry of the LFTB. The lower bound represents the smallest possible mass of aluminum that would not deform significantly under load. The upper bound represents the maximum amount of aluminum that would fit in the appropriate space in the LFTB. These values are rough because many factors (exact aluminum alloy, part geometry, etc.) affect them; more precise limits could be instilled once decisions on these factors are made. Note that the post-optimization masses did not significantly approach these boundaries, so exact values aren’t necessary here.

The rate parameters $k_s$ and $k_{12}$ were bounded based on the commercial availability of springs that fit the nominal form factor. A quick perusal of common industrial supplies (McMaster-Carr et. al.) revealed the values in the table representing the minimum and maximum rate constants available in these sizes. In practice, the spring rates are not only bounded but quantized to some discrete set of available parts. For the purposes of this procedure, a continuous range was used with the assumption that the implementer would then buy the “closest” commercially available spring.
The damping constants $b1$ and $b2$ were bounded on the low end by their nominal estimations. This stems from the fact that friction can always be increased, but not decreased below some threshold. The LFTB’s normal, well-lubricated state is assumed to represent the lowest possible friction.

The controller gains are abstract quantities and therefore don’t strictly need boundaries. However, limiting the gains to be non-negative in this structure prevents the solver from trying to use unfeasible positive feedback controllers.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Lower Boundary</th>
<th>Upper Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>.1 kg</td>
<td>10 kg</td>
</tr>
<tr>
<td>m2</td>
<td>.1 kg</td>
<td>10 kg</td>
</tr>
<tr>
<td>m3</td>
<td>.1 kg</td>
<td>10 kg</td>
</tr>
<tr>
<td>ks</td>
<td>70 N/m</td>
<td>42 kN/m</td>
</tr>
<tr>
<td>k12</td>
<td>100 N/m</td>
<td>1300 kN/m</td>
</tr>
<tr>
<td>b1</td>
<td>100 Ns/m</td>
<td>Inf</td>
</tr>
<tr>
<td>b2</td>
<td>1000 Ns/m</td>
<td>Inf</td>
</tr>
<tr>
<td>Kp</td>
<td>eps (smallest representable positive value)</td>
<td>Inf</td>
</tr>
<tr>
<td>Ki</td>
<td>0</td>
<td>Inf</td>
</tr>
</tbody>
</table>

Table 4: Minimum and maximum parameter value constraints

Parametric sensitivity limits were also created. However, since the sensitivity constraint was handled by the nonlinear inequality function of fmincon, these sensitivities did not need to correspond with the independent variables on a one-to-one basis. Indeed, it was deemed most beneficial to closely monitor the sensitivities of many unchanging constants (such as sample parameters $kt$ and $bt$) because these elements were known with high uncertainty, changing due to thermal effects, or both. At the same time, some of the independent variables’ sensitivity functions were irrelevant given the precision to which they were known and their unchanging nature.
A number of different schemes were tried to set sensible sensitivity limits on system parameters, and finding the best expression of the intent and knowledge of the designer was challenging. The first attempt was to group the parameters into three types: most uncertain, moderate uncertainty, and most certain. The masses and controller gains fell into the last category, as they were well-known values that were strictly unchanging. The maximum parametric sensitivity within the simulation range was therefore not even considered. Other parameters were somewhat more uncertain, owing to manufacturing tolerances and possible minor thermal variations. These included most of the spring rate constants of the system: $ks, k12, k0,$ and $kc$. The sensitivities of these values were limited to at most twice to three times the nominal sensitivities (Table 3). This allowed the solver some leeway, but still encouraged an overall reduction or at least limit to the effects of these parameters on the total system response. Lastly, the parameters with the most uncertainty and/or potential thermal variation, $b1, b2, bt,$ and $kt,$ were strictly bounded by their nominal sensitivity values. This ensured that the solver would under no circumstance increase the relative importance of these values to the system transfer function.

While the above scheme was used and arrived at a solution, the transient response performance was only marginally better than the nominal case. This is because the initial sensitivity limits were far too constraining. Recall that parametric sensitivity has an intuitive interpretation besides an equation relating partial derivatives. It is simply the percentage variation of the transfer function divided by the percentage change of a parameter. As such, sensitivity values are actually just percentages that can be easily understood. If the sensitivity of a system $F$ to a parameter $p$ is 0.5, then a 5% change to $p$
would result in a 2.5% change to $F$. Thus, we can use direct knowledge about our measurement and process uncertainty in our sensitivity thresholds. It was decided that no individual parameter’s sensitivity should affect the transfer function by more than 5% (with this value ultimately being an implementer-specific tradeoff between robustness and transient response).

Once again, the masses and controller parameters were ignored because of their well-known and unchanging nature. It was estimated that the spring rate constants were measured to within about +/-5% of their nominal values, thus these sensitivities were constrained to be less than unity (100%). The damping parameters were determined through rough estimation and indirect data analysis, so they contained significant error. They could easily be off by a factor of 20. Thus, their sensitivities were limited to $5\% / 2000\% = .0025$. Lastly, the sample parameters were complete estimates so were considered to vary by a factor of 1000. So $b_t$ and $k_t$ were constrained by having sensitivities under $5\% / 100,000\% = .00005$. Table 5 summarizes these constraints.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Maximum sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_s$</td>
<td>1.0000</td>
</tr>
<tr>
<td>$k_0$</td>
<td>1.0000</td>
</tr>
<tr>
<td>$k_{12}$</td>
<td>1.0000</td>
</tr>
<tr>
<td>$k_c$</td>
<td>1.0000</td>
</tr>
<tr>
<td>$b_1$</td>
<td>0.0025</td>
</tr>
<tr>
<td>$b_2$</td>
<td>0.0025</td>
</tr>
<tr>
<td>$b_t$</td>
<td>5.0000 e-005</td>
</tr>
<tr>
<td>$k_t$</td>
<td>5.0000 e-005</td>
</tr>
</tbody>
</table>

**Table 5: Parametric sensitivity ceilings for various SFTB parameters**

Although the lower and upper parameter bounds were supplied as vectors to `fmincon`, the sensitivity constraints had to be specified in the form of a callback function conforming to a particular call structure. This function then got called at each
iteration of the optimization procedure to tell the solver whether or not a given search path was valid:

```matlab
function [c, ceq] = sensCon(params)
    global maximumSensitivities; % Global vector of maxes
    ceq = []; % No nonlinear equality
    S = deriveSensitivity(params, 1, 50);
    c = S - maximumSensitivities;
end
```

The final step was to implement the derived SFTB model, sensor model, and controller in Simulink. The MATLAB solver was then passed a callback function to run an iteration of the simulation and evaluate the integral of absolute error performance index. A flow chart of the Simulink process is given in Fig. 5.

![Simulink model of the SFTB and controller](image)

**Figure 6: Simulink model of the SFTB and controller**

The SFTB dynamics $G$ was generated at each iteration based on the current set of parameters and the state-space model from Equations 10-12. The sensor model was also from Chapter 6. The controller $Cnt$ was a PID generated with the following MATLAB code:

```matlab
if( Ki > eps && Kp/Ki > eps )
    Ti = Kp/Ki;
```
\[ \text{Cnt} = K_p (1 + tf([0 1], [Ti 0])); \]
else
\[ \text{Cnt} = K_p; \]
end

By first checking for valid $K_p$ and $K_i$, the controller generation avoids numerical problems or singularities (when $K_i$ is too small, a proportional-only controller is substituted). Without this check, the simulation would periodically fail.

In terms of nonlinearities, additive Gaussian noise of variance $3 \text{Pa}^2$ was added to the feedback path to simulate sensor noise. A known random seed was used for the noise generator in simulations; this allowed for repeatability. An actuator saturation block was also added between the controller and plant. This block limited the command to the motor to the range +/- 10V in order to simulate the real-world effect of limited control bandwidth.

During simulation, the attached scope could be viewed during the optimization procedure. This allowed visual insight into the process, and some of the observations are reported in Chapter 9.

Other optimization parameters of interest include the ODE solver used by Simulink (ode45, based on the Runge-Kutta (4,5) formula for integration) and the maximum step size of the simulation (0.0001 seconds). The ode45 solver represents a good tradeoff between simulation speed and accuracy, and is the recommended solver for most non-stiff MATLAB simulations. The maximum step size parameter helped to ensure that the solver remained stable in the presence of the modeled high-frequency process noise.
Convergence criteria of \texttt{fmincon} were set as follows. A maximum of 10,000 objective function evaluations were permitted. The minimum change in a given parameter between iterations was set to 0.01. The minimum change in the objective function in a given iteration was set to 0.001. Note that for numerical purposes, the initial values of the search parameters were normalized. This was to ensure an equal comparison basis for terminating based on the minimum parameter change (otherwise convergence would take an extremely long time). In practice, convergence was almost always because of the minimum parameter change limit.
Chapter 4  Results and Discussion

This work aimed to optimize state-space models of the LFTBs in a specified sense. One ultimate goal of this work was to present a set of mechanical improvements and recommendations to improve the dynamic properties of the physical LFTB implementations. In this spirit, a few anecdotal observations are needed to provide context before the more mathematical improvements suggested by the techniques explored in this work are applied. For the most part, changes to system parameter values were relatively minor. As a result they are sensible to implement only after any other undesirable system behavior is eliminated.

The RFTB’s dynamic performance was deemed to be “good enough” after manual tuning of the system had occurred. Recall that the RFTB was designed to apply a fixed pressure load prior to the introduction of the sample and then maintain said pressure in the presence of disturbances and thermal effects. Detailed transient analysis and optimization was therefore unnecessary. Steady-state analysis showed that the RFTB was a type 0 system and no steady-state error resulted from position commands in the presence of an integrating controller (such as the discrete PID controller currently installed).

The fact that the RFTB used a compressible rubber bottom roller was both beneficial and detrimental to the disturbance rejection properties of the RFTB. The rate constant of the material was low enough that the disturbance of a paper sample passing through the roller interface was not registered by the low-pass filtered load cells. This stood in contrast to the SFTB, where rigid parts resulted in a system that was highly sensitive to the thickness and compressibility of the sample being fused.
However, the soft black rubber used on the roller proved to be highly sensitive to thermal variations. As the temperature of the fusing interface increased, the spring rate constant of the material decreased. Above a certain critical temperature, permanent deformation of the rubber material occurred. It was for this reason that the operator of the RFTB was instructed never to turn on the heating element when the rollers were at rest; by spinning the rollers, the rubber was allowed to heat uniformly and cool as it rotated away from the heating strip. Decreasing the spring rate of the roller meant that the fuser test bed would exert less force on the sample while maintaining a constant linear actuator position. As a result, the feedback loop moved the actuators downwards as the roller was heated. The time constant of the linear actuator control loop was measured in tenths-of-a-second, while the time constant of the heating and cooling action of the roller was measured in minutes. Thus, no special care was needed for this disturbance; the displacement of the linear actuators was changed smoothly and appropriately as the roller came up to its final temperature.

One (previously unconsidered) source of disturbances remained. The bottom roller was directly coupled to the DC motor that spun it. Misalignments and play in the coupling of the small roller shaft to the much larger motor shaft resulted in measurable periodic disturbances in the load cell output when the feedback loop and heating elements were disabled. However, the maximum practical roller speed of about 80 RPM meant that the system had plenty of time to recover from periodic disturbances and the control system visibly reduced their impact. Moreover, once the rollers were heated and the spring constant was correspondingly lowered, the magnitude of the dynamic disturbances was again reduced in magnitude (as more energy was absorbed by the roller instead of by
the rest of the stiff system). As a practical matter, although the coupling between the motor and roller can undoubtedly be improved by machining and proper coupling selection, once the system is running within its operational envelope the impact is minimal.

A couple of defects in the construction of the RFTB were noticed during the process of installing and tuning the control system. The flexible metal housing holding the rubber roller was observed to be bending when pressure was applied. This piece had originally been assumed to be a rigid body, but that was clearly no longer the case. Unintended bending both shortens the life of a component and introduces susceptibility to new vibrational modes. At certain rotational frequencies, resonance was observed in the roller support. Aluminum blocks were installed to prevent this flexing. Revising a model of the RFTB to include the flexing action and then conducting this work’s nominal optimization procedure would have resulted in a similar but quantifiable recommendation. The spring constant of the metal housing would have been increased by the procedure until the vibrational mode’s frequency had been shifted to outside of the 0-80 RPM range of the rotation.

The SFTB had no significantly flexible components (other than the sample itself) located near the heating element, so these concerns were not applicable. However, some major changes will be required before the parametric and controller recommendations put forth here can be applied. Whereas the RFTB used two DC linear actuators with analog voltage inputs for pressure loading, the SFTB is currently configured to use one stepper motor to rotate the cam and a second stepper motor to raise and lower the entire stamp assembly. The driving circuitry currently used for position and velocity control is limited
in terms of bandwidth, configurability, and feedback capabilities. The most critical limitation is that, for unknown reasons, the manufacturer of the driver chose not to allow the position or velocity setpoint to be changed while the system is in the operating mode. The system can move in user-specified steps, ramps, and acceleration curves, but not arbitrary profiles. This means that external feedback is not natively supported. An attempt was made to let the controlling LabView software incrementally alter the setpoint so that it was achieved very quickly and a thus new setpoint could then be sent to the motor control circuitry, but this resulted in cumulative delays of on the order of a second. As a result, any gains made by mathematically optimizing the system would be insignificant compared to this penalty.

The stepper motor driver does advertise a feedback feature, and this was the next option to be explored. A quadrature shaft encoder could be hardwired into the circuitry to provide feedback, but this capability turned out to be far from ideal. First, the driver required a 24 VDC encoder; the one currently installed on the SFTB is rated only up to 12 VDC. Second, even if the voltage levels could be reconciled, the stepper motor driver does not support proportional feedback control. The driver uses open-loop positioning commands to first move the motor to roughly the right position. Only then does the feedback influence the motor command, and there is no documentation to suggest that this is done with anything more than a “bang-bang” on/off controller. The action may be even simpler and may consist only of issuing a new open-loop command. In either case, the feedback capabilities are wholly unsatisfactory.

Assuming that the cam rotation stepper motor is replaced by a suitably powerful DC motor and appropriate driving circuitry, a few more changes should be made. The
second stepper motor that changed the elevation of the entire stamp was originally installed for two reasons. First, it lifted the stamp in order to allow the operator to insert and remove samples. Second, it allowed the overall gain of the pressure system to be changed either prior to or during fusing operation. In practice, this motor proved not to be powerful enough to maintain its position in the presence of the huge disturbance caused by operating the stamp. Manual fasteners were therefore installed to fix the vertical adjustment of the system prior to operation. Given that the entire structure is essentially put into compression during operation, the tightness of the nuts that secured the vertical adjustment were of extreme importance. The act of tightening a thread alters the relative displacement of two materials. Recalling the high overall system gain from input displacement to output displacement derived in Chapter 6, the additional thickness of half of a thread can have an enormous influence on the output pressure.

What is needed is a regulatory mechanism for ensuring uniform gain between cycles. One possibility is to use specialized hardware or tools that allow precise adjustment every time. A better option may be to completely remove the vertical adjustment capability. With a properly designed cam profile, a “dead area” can be added so that the stamp does not contact the bottom support at some point in its travel. When the cam is rotated to the appropriate position the operator may then insert or remove the sample. Adding to the desire to replace the stepper motor with a more servo-like actuator, overall gain control can be accomplished by varying the motor’s trajectory and/or cam profile rather than by operating a secondary mechanism. Both stepper motors are therefore removed from the SFTB, resulting in a simpler, more linear, and stiffer system.
The last item of topological concern is partially addressed by removing the vertical adjustment capability of the SFTB. The shaft coupler between the cam rotation motor and the cam shaft itself is ingenious in that it allows for two degrees of freedom in alignment while still transmitting maximal torque. It was originally intended that by using this coupler in tandem with the vertical positioning stepper motor, the larger rotation motor could remain stationary while the cam moved up and down. However, the large coupler utilizes three levels of linkages and bearings to transmit power, resulting in a large rotational mass in the torque transmission path. This severely limits the maximum angular acceleration of the cam shaft, which in turn limits the minimum amount of time that the sample is in the nip. An artificial ceiling on the range of available system frequencies is imposed as a result. Since its primary function has been made obsolete by removing the capability for vertical adjustment, using a well-aligned direct-drive interface between the new DC rotation motor and the cam shaft will allow for the maximal range of input frequencies.

Table 6 presents a summary of the major architectural problems encountered in the process of designing control system for the LFTBs, along with the proposed and/or implemented solutions.
<table>
<thead>
<tr>
<th>System</th>
<th>Problem</th>
<th>Proposed Solution</th>
<th>Status as of 8/2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTB</td>
<td>Permanent deformation of the rubber roller can occur when heating</td>
<td>Instruct operator to enable heating elements only after roller rotation is enabled</td>
<td>Implemented</td>
</tr>
<tr>
<td>FTB</td>
<td>Shaft misalignment results in disturbances when rollers are turning</td>
<td>Use an integrating controller and only apply significant pressure when the rollers are heated</td>
<td>Implemented</td>
</tr>
<tr>
<td>FTB</td>
<td>Flexing occurred in the metal roller housing under load</td>
<td>Add aluminum blocks as supports to housing to eliminate flexing</td>
<td>Implemented</td>
</tr>
<tr>
<td>FTB</td>
<td>Stepper motor driver does not allow for proper feedback control</td>
<td>Replace cam shaft stepper motor with an appropriately sized DC motor and compatible driver</td>
<td>Pending</td>
</tr>
<tr>
<td>FTB</td>
<td>Vertical adjustment position highly influences overall gain</td>
<td>Eliminate vertical adjustment capability; Alter the cam profile and motor trajectory instead</td>
<td>Pending</td>
</tr>
<tr>
<td>FTB</td>
<td>Flexible shaft coupling introduces high rotational mass to the cam shaft</td>
<td>Flexible coupling no longer needed; replace with direct coupling to DC motor</td>
<td>Pending</td>
</tr>
</tbody>
</table>

Table 6: Major issues encountered with the LFTBs, proposed solutions, and their status.

Two runs of the optimization procedure were conducted. First, the structural parameters were held constant and the sensitivity constraints were ignored. Next, the entire integrated system was optimized holistically. In both cases, initial guesses were normalized against nominal parameter values prior to starting the optimization procedure. This was necessary to numerically condition the problem; MATLAB uses the same directional derivative termination criteria for all of the tuning parameters regardless of magnitude and the solver would take a very long time to converge without this normalization.
The primary comparative metric between the two studies was the IAE performance index after all optimization had been completed. Significant improvement in this sense was the major goal of the work. Secondarily, the comparative robustness of the final systems was considered. The hard constraints on these quantities that were formulated in Chapter 8 had to be met otherwise the solution would be discarded.
Figure 7: Final simulation results for the manually-tuned controller (A), controller-only optimization (B), and integrated holistic optimization (C). The top plot of each is the input superimposed with the actual response. The middle plot is the control action. The bottom plot is the integral of absolute error performance index (IAE).
The controller-only optimization procedure converged after 8 iterations and 106 objective function evaluations. The IAE performance index was reduced from 1418 to 1368 as a result. The solver converged because the minimum difference between parameters of subsequent iterations was not met.

The integrated optimization procedure converged after 8 iterations and 208 objective function evaluations. The IAE performance index was reduced from 1418 to 937, an improvement of 46.0% over the control and 51.3% better than manual tuning by itself. The solver once again converged because of too little change in parameter values between iterations.

Figure 7 shows the final responses of both optimal controllers and the manually-tuned controller. The difference in the response plot is difficult to discern visually, but the plot of IAE demonstrates the improvement in tracking. Moreover, the plot of controller action clearly shows that the holistically optimized system exerts much less control effort than do the others. This is indicative of improved disturbance and noise rejection capabilities.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Nominal Value</th>
<th>Controller Optimization</th>
<th>Integrated Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1 (kg)</td>
<td>.280</td>
<td>N/A</td>
<td>.746</td>
</tr>
<tr>
<td>m2 (kg)</td>
<td>1.700</td>
<td>N/A</td>
<td>.746</td>
</tr>
<tr>
<td>m3 (kg)</td>
<td>.4977</td>
<td>N/A</td>
<td>1.373</td>
</tr>
<tr>
<td>k0 (N/m)</td>
<td>60577</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>k12 (N/m)</td>
<td>17512</td>
<td>N/A</td>
<td>353058</td>
</tr>
<tr>
<td>ks (N/m)</td>
<td>630</td>
<td>N/A</td>
<td>296</td>
</tr>
<tr>
<td>kt (N/m)</td>
<td>1E9</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>kc (N/m)</td>
<td>1.05E9</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>b1 (Ns/m)</td>
<td>100</td>
<td>N/A</td>
<td>100</td>
</tr>
<tr>
<td>b2 (Ns/m)</td>
<td>1000</td>
<td>N/A</td>
<td>1000</td>
</tr>
<tr>
<td>bt (Ns/m)</td>
<td>1</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Kp</td>
<td>1.4E-6</td>
<td>1.465E-6</td>
<td>9.701E-7</td>
</tr>
<tr>
<td>Ki</td>
<td>3.3E-5</td>
<td>3.256E-5</td>
<td>1.407E-5</td>
</tr>
</tbody>
</table>

Table 7: Parameter values before and after the optimization procedure.

Table 7 shows the initial and final parameter values for the test system. Note that the relative limits on parametric sensitivity put forth above seem to be reflected in the results. The damping coefficients $b1$ and $b2$ were not appreciably altered by the optimization technique. Meanwhile, the large spring $k0$ and the three masses $m1$, $m2$, and $m3$ were each altered by a large amount. The intuitive interpretation of these results is that a heavier, stiffer system allows force to be transferred most effectively. As a result, control loop gains can be lowered and noise susceptibility is reduced. Note that re-running the simulation with different random seeds for the simulated additive Gaussian noise produced results that did not significantly differ from those presented here.

The usefulness of the chirp signal was readily apparent by watching the solver conduct its parameter search in real time. In the course of optimization, the solver would move the vibrational modes of the system around in order to minimize undesirable behavior. As a result, the modes would occasionally be moved into the tested region. Figure 8 shows what an inappropriately placed vibrational mode can look like.
Simulation-based optimization approaches that utilize only a single frequency of input would not reveal this behavior until actual realization.

Figure 8: A vibrational mode is revealed within the range of test frequencies.

Table 8 shows the final parametric sensitivities after the completion of the integrated, constrained optimization procedure. Note that none of the constraints proved to lie at the optimized minimum. This does not mean that the constraints did not affect the final solution; rather, they served to force the solver to select the local minimum that it did. Also note that sensitivities were only increased for a handful of parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Original Maximum Sensitivity</th>
<th>Constraint</th>
<th>Final Maximum Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ks</td>
<td>1.2300 e-006</td>
<td>1.0000</td>
<td>5.7787 e-007</td>
</tr>
<tr>
<td>k0</td>
<td>0.2243</td>
<td>1.0000</td>
<td>0.4669</td>
</tr>
<tr>
<td>k12</td>
<td>0.7757</td>
<td>1.0000</td>
<td>0.5331</td>
</tr>
<tr>
<td>kc</td>
<td>1.3561 e-005</td>
<td>1.0000</td>
<td>2.7229 e-005</td>
</tr>
<tr>
<td>b1</td>
<td>0.0011</td>
<td>0.0025</td>
<td>7.6341 e-004</td>
</tr>
<tr>
<td>b2</td>
<td>1.6936 e-006</td>
<td>0.0025</td>
<td>1.6935 e-006</td>
</tr>
<tr>
<td>kt</td>
<td>1.4240 e-005</td>
<td>5.0000 e-005</td>
<td>2.8592 e-005</td>
</tr>
<tr>
<td>bt</td>
<td>1.2352 e-014</td>
<td>5.0000 e-005</td>
<td>2.4802 e-014</td>
</tr>
</tbody>
</table>

Table 8: Parametric sensitivity values
Chapter 5  Conclusion and Future Extensions

The primary goal of this study was to apply knowledge of the nature of mechanical system properties to help formulate a solvable and gainful optimization problem. The problem was posed as a constrained, non-linear optimization exercise with two distinct types of constraints. First, physical parameter values were bounded so as to meet feasibility and manufacturability criteria. Second, parametric sensitivity expressions for particularly important or uncertain parts of the system were constrained by a designer-specified set of weights. The combination of the two allows the designer to express his intuition mathematically, and helps both to ensure that a workable solution is found and to limit the extent of the search of a potentially high-dimensional solution space.

By conducting optimization through an iterative simulation process, more flexibility is permitted within the model of system than with a purely algebraic method. Nonlinearities such as deadbands and actuator saturation can be modeled and handled without any need to augment the solver. Moreover, arbitrary control system topologies can be used because the optimization routine considers parameter values only. Many modern methods only work when the order of the controller matches the order of the plant. The simulation-based approach also allows the designer to shape the final system characteristics by manipulating the reference input to the model. A chirped sinusoid was used in this study in order to test multiple input frequencies in a single run of the simulation. While the chirp is a convenient way to view the system response one frequency at a time (which is especially useful in revealing vibrational modes of the mechanical plant) any suitable signal could be substituted. This helps to ensure that the
optimized plant-controller combination fits the particular demands of a system’s normal operating window instead of a one-size-fits-all response.

For the SFTB simulation, a performance index of the tracking error was minimized while the specified worst-case robustness floor was maintained. A Simulink model of the SFTB was created and optimized using MATLAB’s Optimization Toolbox. The tracking error’s IAE index was improved by more than 45% over the manually tuned case, and more than 30% over the controller generated by a comparable controls-only optimization procedure. Meanwhile, parametric sensitivities for uncertain or changing physical parameters were derived using MATLAB’s Symbolic Math Toolbox and then constrained by a designer-specified weighting scheme. This was accomplished through the use of a nonlinear constraint function callback passed to the minimization routine. Such a scheme allows for any system transfer function to be constrained, at which point the technique emulates the mixed sensitivity problem as various disturbance transfer functions are minimized.

A major benefit of a suitably constrained, simulation-based minimization problem is that good solutions can arise out of non-convex spaces. Modern methods for convex optimization have become extremely effective, but many real-world problems are difficult if not impossible to pose convexly. This includes a large portion of modern L1, H2, and H-infinity problems. Convexity denotes that any local minimum is also the global minimum, so gradient-descent solvers (such as the one used by MATLAB’s \texttt{fmincon}) will always converge at the globally optimal solution. In a non-convex space, “hill climbing” becomes a problem for these solvers.
In this study, two measures were taken to help increase the probability that the solution achieved a favorable local minimum, if not the global minimum. First, the generous application of constraints to the solver helped to prune unfavorable subsets of the solution space. With enough appropriate constraints, the solution space can become convex. Second, even when faced with multiple solution “basins”, a good initial guess effectively selects the basin of the solution. Manual tuning was conducted until a worst-case performance floor had been obtained. This ensured that any optimization would only make performance better as the system’s operating point was moved to the local minimum. Fig. 8 illustrates a 2D projection of a non-convex solution space and the effects of additional problem constraints on such a space.

![Constraints on a Solution Space](image)

**Figure 9:** A non-convex solution space (left). Blue dots represent local minima. On the right, various constraints are added to “mask out” unfeasible regions, increasing the probability that the solution space becomes convex, or at least has the number of local minima reduced.

For the laser fuser test beds considered by this study, achieving optimal, or at least increased, levels of performance directly translates into a more cost-effective printing solution. The toner fusing process consumes much of the total power of a laser printer
and generates most of the heat. Dissipating this wasted energy requires careful design both of the printer control circuitry and software and the physical printer body. Moreover, it has been shown that the heat and pressure profiles of the fuser constitute a major contribution to the final image quality of the fused specimen. By squeezing every last drop of performance from the fuser, printing quality is increased without increasing power consumption and heat generation.

There are two logical “next steps” for this research. The first is to realize the changes suggested by both the intuitive inspection of the LFTBs and the optimization of the SFTB model. In the absence of a rigorous process model for toner fusing, this is the only way to verify the claim that decreasing the tracking error of the LFTB would directly result in improved image quality or consistency. The second step is to continue to develop the mathematical framework of this study’s proposed integrated optimization approach in order to determine how well the concept generalizes to other types of systems.

In order to affect the changes recommended by the optimization procedure, the improvements from Chapter 9 should first be implemented. It should be stressed that the dynamic performance and robustness improvements of the optimized SFTB model are only possible in the presence of a low-latency feedback framework that is not currently installed. The substantial delays in the load cell-stepper motor loop as well as the limited control modes of the stepper motor controller make effective feedback impossible. Moreover, improving the strength of the mechanical structures of both the RFTB and SFTB represents the “lowest hanging fruit” in terms of performance improvements. The optimized plant-controller combination for the SFTB decreased tracking error by 45% in
simulation, but the effect of unmodeled deformation in such a stiff system is several orders of magnitude greater than this improvement.

Once a servo-like control loop has been added to the SFTB and the measurement system has been shown to be repeatable (by addressing the system of locking the stamp into place by hand), the parametric optimization results can begin to be applied. While the optimization procedure was conducted with a total of nine independent variables consisting of seven physical parameters and two controller gains, not all nine need to be tweaked in order to realize performance benefits. Indeed, inspecting the parametric sensitivity table generated in Chapter 7, the relative importance of each parameter to the final response can be determined. This information could be combined with a “cost” model capturing the engineering effort necessary to enact each change. The product of the cost and utility will reveal the most gainful use of engineering time to improve the LFTB.

Manipulating system masses requires replacing various aluminum components of the test bed with new appropriately lighter or heavier pieces. Because the mass components of the system were bounded based on the minimum or maximum amount of material that could occupy the available volume in the LFTB structure without deforming under load, this is as simple as designing new aluminum blocks that provide the same vertical dimension as the nominal ones while varying the width of material so as to meet the specified weight (which is a function of volume for a known density, in this case a particular aluminum alloy). The part can then be milled (with a CNC if necessary) and installed.
Changing spring rate constants should be accomplished by replacing the existing LFTB springs with commercially available models in the same form factor. Industrial suppliers such as McMaster-Carr typically offer several different spring rates for a given uncompressed length, so replacement is as simple as purchasing a new COTS part. Because of this simplicity, as well as the particularly high parametric sensitivities of these components, the rate constants of the LFTB present a good candidate for an implementation starting point.

Viscous damping parameters are difficult to accurately estimate and just as difficult to control. It is therefore fortunate that the sensitivity of the system to frictional losses is very low, and that the optimization procedure did not suggest altering them appreciably. It is therefore recommended that the LFTB remain in its existing well-lubricated state and that no measures should be taken to change the damping coefficients.

Changing controller gains is as simple as setting a constant in software, so the implementation cost is essentially zero. As such, altering Kp and Ki should definitely be part of the implementation plan. Moreover, because of this ease of manipulation, Kp and Ki can together be manually tuned after all of the physical alterations are implemented in order to compensate for the residual impact of the unimplemented changes. This is common practice in applied controls, as various modeling errors and uncertainties often combine to make simulation gains less than optimal (perhaps even unstable) for the tangible plant and controller. The calculated gains should therefore be treated as a starting point for manual “optimization”.

Although the integrated optimization approach outlined in this study seemed to yield realizable and sensible improvements to the SFTB, a single case study is not
enough to come to general conclusions about its merits. The test system in this case had only six states and fewer than twenty physical and controls variables. While this scale is representative of the class of bench top engineering test systems, it is much smaller than that of many of the traditional applications of control theory. Automotive dynamics, chemical processes, aerodynamics, and robotics often deal with systems of hundreds of states. Parametric optimization, like many fundamental computer science problems, suffers from the “curse of dimensionality”, whereby the search space rapidly grows to the limits of computation as more dimensions are added to the problem. In order to handle these larger systems, the technique described herein must be augmented.

In the SFTB case study, sensitivity analysis was conducted in order to paint a picture of the relative importance of each parameter to the total system response. These sensitivity functions not only helped to restrict the search space of the parametric optimization problem, they also gave some insight into which physical changes would be most fruitful to implement. For complex problems, exploiting this knowledge is crucial to limiting the dimensionality of the search space. Every parameter whose alteration offers little to the final plant-controller combination should be excluded from the parameter vector in order to focus on the areas where the most overall improvement can be achieved.

Even if the problem space can be reduced to a feasible scale, the question of convexity needs to be revisited in order to evaluate the applicability of the integrated parametric optimization approach to a more general set of controls problems. Recall that for the purposes of the SFTB optimization procedure, convexity was not rigorously considered. It was acknowledged that the combination of a multivariate cost functional
and particular plant nonlinearities led to what was most likely a non-convex solution space. However, a gradient-descent based solver was still used to solve the system. By exploiting a relatively large number of problem constraints coupled with a good “initial guess” established by manually tuning the system until an acceptable performance baseline was achieved, it was asserted that a small subset of the original problem space was actually considered. By excluding much of the space, many local minima were pruned. The initial guess ensured that whatever “basin” the solver would ultimately choose was viable, if not the globally optimal one.

In order for this technique to be generalized, this intuitive assertion should be rigorously analyzed within an algebraic framework. It gives rise to many questions that should be answered. How many constraints are necessary to prune the solution space to the point of convexity, or at least to the point that an arbitrarily good solution is always found? What kinds of constraints? Can a probabilistic representation of the relationship between these constraints and the convexity of the solution space be expressed? The various attractive properties of simulation-based methods (the seamless handling of plant nonlinearities and arbitrary controller topologies) make finding mathematical answers to these questions extremely challenging.

There are other common non-convex optimization techniques that may be useful in solving a larger-scale version of the problem. For example, Monte Carlo methods could choose random stable “starting points” for the optimization procedure and perform multiple runs, choosing the best solution from among them. Simulated annealing purposefully introduces an entropy step after each iteration, worsening the current solution in the hopes of overcoming local hills and valleys in the solution space.
Performance-index based parametric optimization is a topic that has fallen from the forefront of modern control systems research, but one that was shown by this study to still be a viable path towards a solution for at least one class of system. By using a careful combination of reference signal selection, a good cost functional, and parameter constraints, many of the traditional pitfalls of simulation-based methods can be alleviated. The use of a priori knowledge of the physics of the plant helps to ensure that the solution meets various cost, labor, or robustness criteria. With additional investigation, the technique could be extended to a wider variety of controls problems.
Bibliography


function F = sftb_error(params,params0,complete)
% sftb_error.m
% % Run the SFTB test scenario and return the IAE (integral of absolute error).
% % params = The normalized parameter values (nominally ones)
% params0 = The nominal parameter values
% complete = 1 if integrated optimization is taking place, 0 otherwise
% % Returns: F, the IAE of the current configuration over a 1 sec chirp
% jsr, 2008

if( complete > 0 )
    % 1. Assign parameters and constants
    m1 = params(1)*params0(1);   % [ kg]
    m2 = params(2)*params0(2);   % [ kg]
    m3 = params(3)*params0(3);   % [ kg]

    ks  = params(4)*params0(4);   % [N/m] Small Springs
    k0  = 60577; % [N/m] Flexible Shaft
    k12 = params(5)*params0(5); % [N/m] Orange Spring
    kc  = 1.05E9; % [N/m] Load Cell
    b1  = params(6)*params0(6); % [Ns/m] Lever damping
    b2  = params(7)*params0(7); % [Ns/m] Stamp damping

    kt  = 1E9; % [N/m] Specimen spring constant
    bt  = 1; % [Ns/m] Specimen damping constant
    cs  = .001; % [m^2] Estimate of cross-sectional area

    Kp = params(8)*params0(8);
    Ki = params(9)*params0(9);
else
    % 1. Assign parameters and constants
    m1 = .28;   % [ kg]
    m2 = 1.7;   % [ kg]
    m3 = .4977; % [ kg]

    ks  = 630;   % [N/m] Small Springs
    k0  = 60577; % [N/m] Flexible Shaft
    k12 = 17512; % [N/m] Orange Spring
    kc  = 1.05E9; % [N/m] Load Cell
    b1  = 100; % [Ns/m] Lever damping
    b2  = 1000; % [Ns/m] Stamp damping

    kt  = 1E9; % [N/m] Specimen spring constant
    bt  = 1; % [Ns/m] Specimen damping constant
    cs  = .001; % [m^2] Estimate of cross-sectional area

    Kp = params(1)*params0(1);
    Ki = params(2)*params0(2);
end
% 2. Generate the system

% 2a: The system dynamics
A =
[0   0   0   1   0   0 ;
0   0   0   0   1   0 ;
0   0   0   0   0   1 ;
-(k0+k12)/m1 k12/m1 0  -(b1)/m1 0  0 ;
k12/m2 -(k12+ks+kt)/m2 kt/m2 0  -(bt+b2)/m2 bt/m2 ;
0   kt/m3  -(kt+kc)/m3 0   bt/m3  -bt/m3 ];
B = [0 0 0 k0/m1 0 0 ]';
C = [0 0 kc/cs 0 0 0 ];
D = [0 ];
G=balreal(ss(A,B,C,D));

% 2b: The sensor
H = zpk(0,-.1,1);

% 2c: The controller
if (Ki > eps && Kp/Ki > eps )
    Ti = Kp/Ki;
    Cnt = Kp*(1 + tf([0 1],[Ti 0]));
else
    Cnt = Kp;
end

% 3. Set up and run the simulatoin
opt = simset('solver','ode45','SrcWorkspace','Current','MaxStep',.0001);
[tout,xout,yout] = sim('lftbsim',[0 1],opt);
F = yout(end);
assignin('base','F_LFTB_ERROR',F);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End of sftb_error.m
function p = sftb_optim(complete)

% sftb_optim.m
% Optimizing the LFTB plant and controller, subject to constraints
% complete = 1 if integrated optimization should occur, 0 otherwise
% Returns: p, the optimized plant/controller
% jsr, 2008

global s0 weighting;
if( complete > 0 )
    % Initial guesses
    
    params0 = [
    .28 % m1 (kg)
    1.7 % m2 (kg)
    .4977 % m3 (kg)
    630 % small spring (N/m)
    17512 % orange spring (N/m)
    100 % Lever damping (Ns/m)
    1000 % Stamp damping (Ns/m)
    1.4e-6 % Kp
    3.3e-5 % Ki
    ];

    % Set parametric boundaries
    lb = [
    .1 % m1 (kg) - Minimum mass in volume
    .1 % m2 (kg) - Minimum mass in volume
    .1 % m3 (kg) - Minimum mass in volume
    70 % small spring (N/m) - Smallest available spring
    100 % orange spring (N/m) - Smallest available spring
    100 % Lever damping (Ns/m) - At least nominal
    1000 % Stamp damping (Ns/m) - At least nominal
    eps % Kp - Only positive controllers
    0 % Ki - Only non-negative controllers
    ];
    ub = [
    10.0 % m1 (kg) - Maximum mass in volume
    10.0 % m2 (kg) - Maximum mass in volume
    10.0 % m3 (kg) - Maximum mass in volume
    42000 % small spring (N/m) - Largest available spring
    1300000 % orange spring (N/m) - Largest available spring
    Inf % Lever damping (Ns/m)
    Inf % Stamp damping (Ns/m)
    Inf % Kp (no upper limit)
    Inf % Ki (no upper limit)
    ];

    lb = lb ./ params0;
    ub = ub ./ params0;
else
    % Initial guesses
    
    params0 = [
    1.4e-6 % Kp
    ];
3.3e-5 % Ki
];

% Set parametric boundaries
lb = [eps % Kp - Only positive controllers
0 % Ki - Only non-negative controllers ];

ub = [Inf % Kp (no upper limit)
Inf % Ki (no upper limit ) ];
end

% Set up optimization options
options = optimset('FunValCheck','on',...
'Display','iter',...
'Diagnostics','on',...
'TolX',1e-2,...
'TolFun',1e-1,...
'MaxFunEvals',10000,...
'DiffMinChange',1e-3,...
'OutputFcn',@outfun);

% RUN OPTIMIZATION ALGORITHM!
p = fmincon('sftb_error', ones(size(params0)), [], [], [], [], lb, ub, @sensCon, options, params0, complete);
end

% Output function
function stop = outfun(x,optimValues,state,params0, complete)
stop = false;

switch state
  case 'init'
    %hold on
  case 'iter'
    disp('Current parameter values:');
    disp(x);
  case 'done'
    %hold off
  otherwise
end
end

% Nonlinear (sensitivity) constraints
function [c, ceq] = sensCon(params,params0,complete)
global maximumSensitivities; % Global vector of maxes
ceq = []; % No nonlinear equality

if complete
  % Compute sensitivity for this step
  S = deriveSensitivity(params.*params0, 1, 50);
  c = S - maximumSensitivities;
end
end
End of sfth_optim.m
function S=deriveSensitivity(params, fmin, fmax)
% Derive the sensitivity values for the parameters in the SFTB
% params = the current parameter values
% fmin = min. frequency of sweep
% fmax = max. frequency of sweep
% Returns: S = vector of maximum sensitivites over [fmin, fmax]
% jsr, 2008

tic
% Make the nominal plant
global m1 m2 m3 ks k0 k12 kc b1 b2 kt bt csa s;
syms m1 m2 m3 ks k0 k12 kc b1 b2 kt bt csa s;
wmin = fmin*pi/180;
wmax = fmax*pi/180;

A = [0 0 0 1 0 0 0; 0 0 0 0 1 0 0; 0 0 0 0 0 1; -(k0+k12)/m1 k12/m1 0 -(b1)/m1 0 0; k12/m2 -(k12+ks+kt)/m2 kt/m2 0 -(bt+b2)/m2 bt/m2; 0 kt/m3 -(kt+kc)/m3 0 bt/m3 -bt/m3];
B = [0; 0; 0; k0/m1; 0; 0];
C = [0 0 kc/csa 0 0 0];
D = [0];
G=C*(s*eye(size(A))-A)^-1*B + D;

% Get jacobian (array of partial derivatives that we care about)
partials = jacobian(G, [ks k0 k12 kc b1 b2 kt bt]);

% Make the sensitivity functions
sens = simplify(partials.*[params(4) 60577 params(5 ) 1.05E9, params(6:7)', 1E9, 1]/G);
S = zeros(1,8);

% For each independent parameter...
for j=1:8
  % Plug in our parameters
  subst = funEval(sens(j),params);

  % Now do a frequency sweep
  S(j) = max(abs(squeeze(freqresp(sym2tf(subst),[wmin:.01:wmax]))));
end
toc

function F = funEval(system, params)
global m1 m2 m3 ks k0 k12 kc b1 b2 kt bt csa s;
fullparams = [params(1:4)', 60577, params(5), 1.05E9, params(6:7)', 1E9, 1, .001];
F = subs(system,[m1 m2 m3 ks k0 k12 kc b1 b2 kt bt csa],
fullparams);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End of deriveSensitivity.m