Optimization of Hard Disk Drive Components

by

Matthew Arnold O'Hara

B.S. (University of Maryland, College Park) 1992
M.S. (University of California, Berkeley) 1994

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy
in

Engineering-Mechanical Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY of CALIFORNIA, BERKELEY

Committee in charge:

Professor David B. Bogy, Chair
Professor Albert Pisano
Professor Alberto Sangiovanni-Vincentelli

Spring 1997

Optimization of Hard Disk Drive Components

by

Matthew Arnold O’Hara

Abstract

Optimization of Hard Disk Drive Components

by

Matthew Arnold O'Hara

Doctor of Philosophy in Engineering-Mechanical Engineering

University of California, Berkeley

Professor David B. Bogy, Chair

Computer hard disk drives have been used as large capacity storage devices for better than three decades. Over the course of that time, the amount and density of information held has increased manyfold. Current hard disk drives combine technology from the fields of magnetics, material science, signal processing, mechanics and fluid dynamics. Optimization allows a number of the considerations from these various fields to be addressed simultaneously

This dissertation focuses on the development and application of global optimization techniques to the problem of hard disk drive component design. Genetic algorithms and the Simulated Annealing (SA) algorithm are compared for their applicability to the problem of hard disk drive component optimization. Through empirical and theoretical study SA has been found to be superior for this problem. An accompanying software package that utilizes an implementation of SA has been developed that allows for the multi-objective optimization of air bearing slider and suspension designs. It is demonstrated that through proper application of the guiding principles of global optimization the slider air bearing and suspension design problems can be solved in an efficient and systematic fashion for various design objectives.

In order to understand the underlying physics of the problem, a description of the governing equation of the physical process--the Reynolds' equation--is presented in addition to the method implemented to solve it numerically. Also, a survey of deterministic and stochastic global optimization techniques is given to provide a backdrop for the current study.

The parameters used in the simulated annealing algorithm that allow it to achieve global optima for the problem at hand in a more efficient manner are determined. Separate suggestions are given for the slider design and suspension design problems. The suspension design problem demands an implementation that requires more function evaluations due to the sensitivity of the objective function to the parameters employed. Several examples are given in support of this claim.

Application of SA to the problem of slider altitude sensitivity optimization is presented. It is demonstrated that altitude insensitive designs can, indeed, be achieved through use of SA. The application of SA to the problem of sub-ambient pressure minimization is also presented. It is demonstrated that sub-ambient pressure minimization can be achieved through use of SA

The problem of slider dynamic fly height variation is addressed through optimization. Through modification of static characteristics of the bearing, an attempt was made to modify the dynamic characteristics of the bearing. Although the optimization algorithm optimized the quantities that were included in the objective function, the resulting designs did not minimize the dynamic fly height variation.

Finally, the application of SA to the problem of suspension optimization is shown. This problem is shown to be feasible. It demonstrates that the procedure of structural optimization is possible through application of the SA algorithm using an appropriately generous cooling schedule.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

CHAPTER 9

LIST OF TABLES

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

CHAPTER 9

ACKNOWLEDGMENTS

First of all, I would like to thank Prof. David B. Bogy. Without his guidance and support, graduation would not have been a possibility. I would also like to thank Profs. Pisano and Sangiovanni-Vincentelli for reviewing this thesis. This research was supported primarily by the Computer Mechanics Laboratory with some portion provided by the National Storage Industry Consortium (NSIC).

I would also like to thank those who made material contributions to the completion of this thesis: Ryan Jurgenson, Jim Mahoney, Tom Christienson, and specifically Bob Evans of Hutchinson Technology for help with the suspension study; Nestor Queipo, a former colleague at Berkeley, for help in more thoroughly understanding Genetic Algorithms; Dr. C. Singh Bhatia of IBM for managing the head disk interface group of the NSIC project and for his constant "encouragement"; and Jennifer Chen (趙爽新) for her patience and valuable assistance.

Although there are too many to mention, I would like to acknowledge all the people I have worked with at the Computer Mechanics Laboratory. In particular, I would like to thank Yong Hu and Sha Lu for all of the collaborative efforts we have made. I would also like to recognize these two as the complement of the "three-headed dragon."

Finally, I would like to acknowledge all of the people who have provided support and love. In particular, I would like to thank my parents for providing me the space to grow into adulthood and my girlfriend Yuming He (何予明) for having such an impact on my life.

CHAPTER 1

INTRODUCTION

The current demand for memory of digital information is rapacious. Current hard disk drive technology is continually being pushed to higher levels of density and speed concomitant with downward pressures on cost. The ability to efficiently design, evaluate, improve and subsequently fabricate air bearing sliders and suspensions is deemed crucial for success in keeping up with these demands. The subject of this dissertation is the application of global optimization techniques to slider air bearing and suspension design problems. This novel application facilitates the design and improvement stages listed above. It is demonstrated that through proper application of the guiding principles of global optimization the slider air bearing and suspension design problems can be solved in an efficient and systematic fashion for various design objectives.

Slider air bearings have been used in hard disk drives for over three decades. Over the course of those decades, the fly height of the slider--taken as the spacing between the slider surface and the spinning magnetic disk beneath it--has decreased considerably. The advantage of lower flying sliders is increased data storage per unit

area. The following graph (Figure 1.1) demonstrates the change in fly height over the last decade as reported in relevant literature.

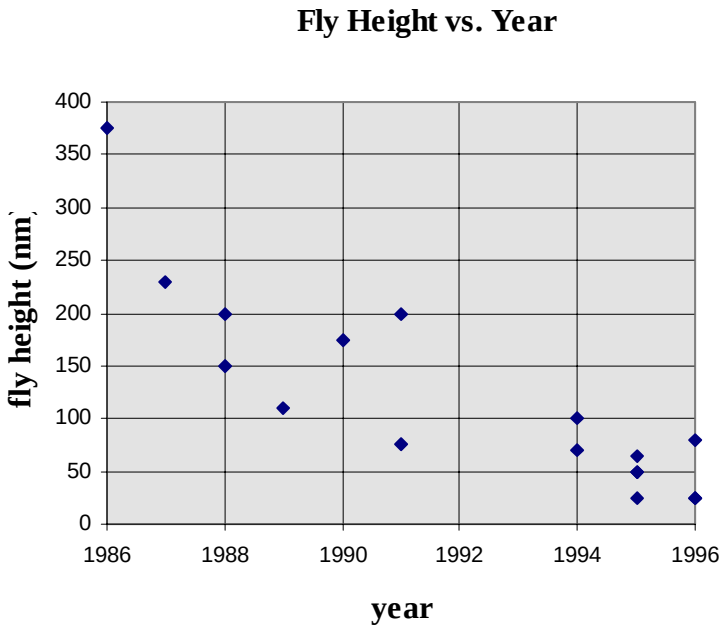

Figure 1.1: Plot of fly height vs. year as reported in relevant literature

This graph (Figure 1.1) clearly demonstrates that the nominal fly height of slider air bearings has steadily decreased and is now at the point where the slider is exceptionally close to the disk. While this provides the advantage of increased data storage density, it creates the problem of a very narrow margin for error in design and fabrication.

The flying characteristics of slider air bearings, both static and dynamic, have been studied numerically for over three decades. Over the course of that time, significant improvements have been made in modeling capability and sophistication. Solution of the Reynolds' lubrication equation is the basis for these simulations. Some of the original work was done by Castelli and Pirvics (1968), Ono (1975) and Gross (1980). Over time, as the fly height of the slider reduced to a level that was less than the mean free path of air, modifications had to be made to the boundary conditions such that Reynolds'

lubrication equation would still be valid. The first modification was the first-order slip correction derived by Burgdorfer (1959) followed by the second-order slip correction derived by Hsia and Domoto (1983) and higher order correction by Gans (1985). Subsequently, an approach was taken to derive the modifications to the Reynolds' equation from the linearized Boltzmann equation. Fukui and Kaneko (1988a, 1988b) derived this relationship and verified it experimentally.

Since derivation of modified boundary conditions to the Reynolds' equation many studies have been performed to further quantify the characteristics of the so-called head disk interface. Ruiz and Bogy (1988) compared the effects of employing different boundary conditions on the air bearing's calculated load capacity. White and Ponnaganti (1988) derived an expression for the slider's dynamic behavior including modeling of contacts between the slider and disk. Bolasna (1990) examined the effects of various parameters on the take-off velocity of sliders. (The take-off velocity of the slider is the speed at which the air bearing beneath the slider is fully developed.) Hsia and Chang (1995) performed a Monte-Carlo statistical analysis of the air bearing's response to variation in manufacturing parameters. Liu and Soh (1996) investigated the effects of seeking velocity on air bearing performance through comparison of commonly found designs. Hu and Bogy (1996) examined the spacing modulation of air bearings as they make the transition from a designated take-off zone to a designated data read/write zone. These are only a sampling of an expansive set of studies that have been performed.

In addition to the above general studies, studies of specific slider designs have been reported. Yoneoka, et. al. (1987) investigated the performance of a so-called fast take-off negative pressure slider. Hardie, et. al. (1994) reported the results of an analysis

of the Seagate Advance Air Bearing Slider. Lu, et. al. (1995) not only performed numerical simulation of a specific slider design but reported results of fabrication and measurement of the design as well.

Naturally following these studies were basic attempts at optimization of the performance of air bearings. Early disk drives moved the slider across the surface of the disk along a single radius. This method is called linear actuation. Later designs moved the slider across the disk's surface by rotating an arm about a single pivot point similar to a phonograph. This method is called rotary actuation. White (1986) reported the performance of a rotary actuated design that achieved uniform flying height. Henze (1989) reported the results of an air bearing that minimized the effects of skew. This design was a modification of the early standard air bearing--that being a design with two straight rails and a taper region at the front of either rail. The modification studied was simply a transverse cut through these straight rails. The study investigated the optimal configuration for various radius and skew combinations. Cooper (1990) took a different tack and investigated optimization of air bearing compliance using constraints supplied by actuator design. Yoon and Choi (1995, 1996) presented work that attempted to address the question of optimization more directly but were limited to relatively simple studies with local optimization techniques.

This thesis will attempt to address the issue of air bearing optimization from a general framework. The range of performance objectives considered includes static characteristics, e.g., fly height attitude of the slider, dynamic characteristics--through sensitivities and crown minimization--and tribological characteristics through sub-ambient pressure optimization. Sensitivity optimization is also a method of increasing

the stiffness of the air bearing which has the capability of minimizing changes in fly height during track accessing (Tokuyama, et. al., 1987 and Matsumoto, et. al., 1994), increasing flying stability (Yoneoka, et. al., 1987) and increasing reliability (Hardie, et. al., 1994). In pursuit of these objectives global optimization techniques are employed, primarily simulated annealing, which are able to find the global minimum of non-linear objective functions. Optimal implementation of the simulated annealing algorithm as it applies to air bearing optimization is also presented. The issue of suspension optimization will also be addressed utilizing the same tools that have been developed for air bearing optimization.

In chapter 2 a description of the governing equation of the physical process--the Reynolds' equation--is presented. This chapter also discusses in detail, the methods implemented in the numerical solution of this equation. Chapter 3 gives a survey of currently used deterministic and stochastic global optimization techniques. Chapter 4 gives a description of the implementation of the simulated annealing algorithm and the genetic algorithm used for verification of applicability to the slider air bearing design is presented. In this chapter, the algorithms are explained and then applied to an example problem whose search space has been mapped out completely. It is this search space mapping that allows for the algorithms to be checked in a very rigorous, albeit empirical, fashion. Chapter 5 explores the parameters used in the simulated annealing algorithm that allow it to achieve global optima and provides values that are optimal for the slider air bearing problem. Several examples throughout this thesis are given in support of this claim. Chapter 6 describes usage of the simulated annealing algorithm to create designs that are insensitive to changes in ambient pressure. Chapter 7 discusses the use of the

simulated annealing algorithm for minimizing sub-ambient pressure while maintaining other desirable flying characteristics for slider air bearings. The purpose of this work is to improve tribological performance. Chapter 8 explores attempts to optimize dynamic performance of slider air bearings through optimization of static design characteristics, specifically, bearing sensitivity to load and pitch and, independently, sensitivity to crown. Chapter 9 explores application of global optimization to suspension design. Finally, Chapter 10 concludes and summarizes all the material presented.

CHAPTER 2

SOLUTION OF THE REYNOLDS EQUATION

INTRODUCTION

Design of sliders used in modern disk drives depends heavily on numerical simulation of the pressure field beneath the slider. This is achieved by numerical solution of the Reynolds' lubrication equation. This type of problem falls generally into the classification of high speed, low spacing gas-lubricated bearing problems (Gross, 1980). A large amount of research has been focused in this area over roughly the past thirty years (Burgdorfer, 1959, Ono, 1975, Gross, 1980, Fukui and Kaneko, 1988a). This corresponds to the existence of the modern hard disk drive. As the demand for storage capacity has increased over the years, the spacing between the slider and the disk on which it flies has decreased commensurately. This has necessitated modification of the Reynolds' Equation to account for slider-disk spacings that are actually less than the mean free path of air at standard temperature and pressure.

The ratio of the mean free path of air to the characteristic spacing in a fluid flow problem is known as the Knudsen number. The Reynolds' equation, when employing a

continuum model, is generally considered accurate when the Knudsen number is on the order of 0.1 (Gross, 1980). The mean free path of air at standard temperature and pressure is 63.4nm. This implies that in order for the Reynolds' Equation to be valid the slider-disk spacing has to be on the order of 600nm. The spacing in current disk drives is on the order of 25nm with some designs actually intended to be in contact with the disk during operational conditions. This has led to necessary modifications of the Reynolds' Equation. The first type of modification was known as the first order slip correction (Burgdorfer, 1959). This attempted to account for the extremely small spacing by modifying the flow and temperature boundary conditions at the fluid/slider and fluid/disk interface. The next major correction that was attempted followed the same logic but extended the principle to a second order correction (Hsia and Domoto, 1983). Finally, higher order slip models were developed as the logical conclusion of the approach (Gans, 1985). The correction that has yielded the closest correlation with experiment, however, was not based on a slip correction approach but instead employed a modification based on the linearized Boltzmann equation (Fukui and Kaneko, 1988) which is the equation that governs, statistically, the behavior of gases.

Due to the geometry of sliders currently employed, a closed form solution to the Reynolds' equation is not feasible (Gross, 1980). This implies that the only option is to solve the problem numerically. To this end, an air bearing solution code has been developed in the Computer Mechanics Laboratory (Lu and Bogy, 1994). This code discretizes the problem through use of control volumes (Patankar, 1980). The resulting discretized problem is then solved iteratively.

Solution of the discretized problem can become expensive if the number of grid points utilized becomes large. It has been found that iteration of a problem on a single computational grid suffers the drawback of not being able to relax all frequencies of the error at the same rate. In order to ameliorate this problem the multi-grid computational method is employed. The central idea in the multi-grid method is to relax different frequency components of the error on different computational grids. Brandt (1977) showed, through use of Fourier analysis, that iterative methods are only efficient at smoothing the error components whose frequency is comparable to the wavelength of the computational grid. By successively interpolating on grids with different spacing, the problem can be solved in a considerably more efficient way than iteration on a single grid would allow. (Shyy and Sun, 1993).

MATHEMATICAL FORMULATION

GENERALIZED REYNOLDS' EQUATION

The Reynolds' lubrication equation which governs the pressure distribution between the slider and disk is a compressible-flow partial differential equation. It is written as

$$\frac{\partial}{\partial x}\left[ph^3 \frac{\partial p}{\partial x}\right] + \frac{\partial}{\partial y}\left[ph^3 \frac{\partial p}{\partial y}\right] = 6U\mu \frac{\partial}{\partial x}[ph] + 6V\mu \frac{\partial}{\partial y}[ph] + 12\mu \frac{\partial}{\partial t}[ph] \qquad (2.1)$$

where p is pressure, h is the slider disk separation as a function of position, μ is the viscosity of the gas and U and V are the sliding velocities in the length (x) and transverse (y) directions of the slider, respectively. This equation is derived from the Navier-Stokes

flow equations by assuming negligible inertial and body forces, laminar flow, Newtonian viscosity, no-slip boundary conditions at the walls and a small film thickness (Gross, 1980, Hamrock, 1991).

As mentioned previously, however, when the film thickness approaches the mean-free-path of the embodying fluid, air in this case, the no-slip boundary condition no longer is valid and one of the modifications listed previously needs to be employed. For convenience, the following non-dimensional variables will be defined

$$X = \frac{x}{L}, Y = \frac{y}{L}, H = \frac{h}{h_m}, P = \frac{p}{p_a}, T = \omega t \tag{2.2}$$

where L, h_m, p_a, ωt are the slider's length, minimum spacing, ambient pressure, and chosen angular frequency respectively. With these definitions, all forms of the Reynolds' equation listed above can be written in a single expression (Ruiz and Bogy, 1990)

$$\frac{\partial}{\partial X}\left[\hat{Q}PH^3\frac{\partial P}{\partial X} - \Lambda_x PH\right] + \frac{\partial}{\partial Y}\left[\hat{Q}PH^3\frac{\partial P}{\partial Y} - \Lambda_y PH\right] = \sigma\frac{\partial}{\partial T}[PH] \tag{2.3}$$

where $\Lambda_x = 6\mu UL/p_a h_m^2$ and $\Lambda_y = 6\mu VL/p_a h_m^2$ and the squeeze number $\sigma = 12\mu\omega L^2/p_a h_m^2$. $\hat{Q}$ is the Poiseuille flow factor. The values substituted for this factor indicate the type of slip-flow modification implemented:

$\hat{Q} = 1$	Continuum model
$\hat{Q} = 1 + 6a\frac{K_n}{PH}$	First order slip model
$\hat{Q} = 1 + 6\frac{K_n}{PH} + 6\left[\frac{K_n}{PH}\right]^2$	Second order slip model
$\hat{Q} = f\left(\frac{K_n}{PH}\right)$	Fukui-Kaneko model

where $K_n = \lambda/h_m$ is the Knudsen number, λ is the mean free path of the interface medium, air in this case, and $a = (2-\alpha)/\alpha$ with α being the accommodation coefficient. The

accommodation coefficient represents the reflectivity of the surface with respect to the incident molecules. It accounts for surface variations. In the final equation above, the functional form of f(K_n / PH) is given by Fukui and Kaneko (1988). In the current implementation, the database form of the Fukui-Kaneko model is employed (Fukui and Kaneko, 1990).

DISCRETIZATION OF THE REYNOLDS' EQUATION

The Reynolds' equation solver that was implemented in concert with the optimization program has gone through many iterations. Previously, when the equation was discretized, linearization was employed (Miu and Bogy, 1986 and Cha, 1993). The current discretization of the Reynolds' equation follows Patankar's (1980) unified control volume formulation for a general class of convection-diffusion equations of the following form:

$$\frac{\partial}{\partial t}(\rho\phi) + \nabla\cdot(\rho\underline{u}\phi) = \nabla\cdot(\Gamma\nabla\phi) + S \tag{2.4}$$

where ϕ is the dependent variable, Γ is the diffusion coefficient, $\underline{u}$ is the flow velocity and S is the source term. This equation can be written in dimensional form as follows:

$$\frac{\partial}{\partial t}(\rho\phi) + \frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} = S \tag{2.5}$$

where J_x and J_y are the convection and diffusion fluxes defined as:

$$\begin{aligned} J_x &\equiv \rho u\phi - \Gamma\frac{\partial\phi}{\partial x}, \\ J_y &\equiv \rho v\phi - \Gamma\frac{\partial\phi}{\partial y} \end{aligned} \tag{2.6}$$

where u and v represent the x and y components of $\underline{u}$.

The generalized Reynolds' equation written above (eq. 2.1) can be cast into this form using the following definitions, $\phi = P$, $\rho = H$, $u = \Lambda_x/\sigma$, $v = \Lambda_y/\sigma$, and $\Gamma = \hat{Q}PH^3/\sigma$.

The next step in the explanation of the numerical solution of the Reynolds' equation is the discretization of the above convection-diffusion equation. An implicit method is employed for the unsteady term. Clearly, if a steady state solution is sought, the unsteady term can be neglected entirely. The above equation (eq. 2.5), when integrated over the following control volume (Figure 2.1), yields

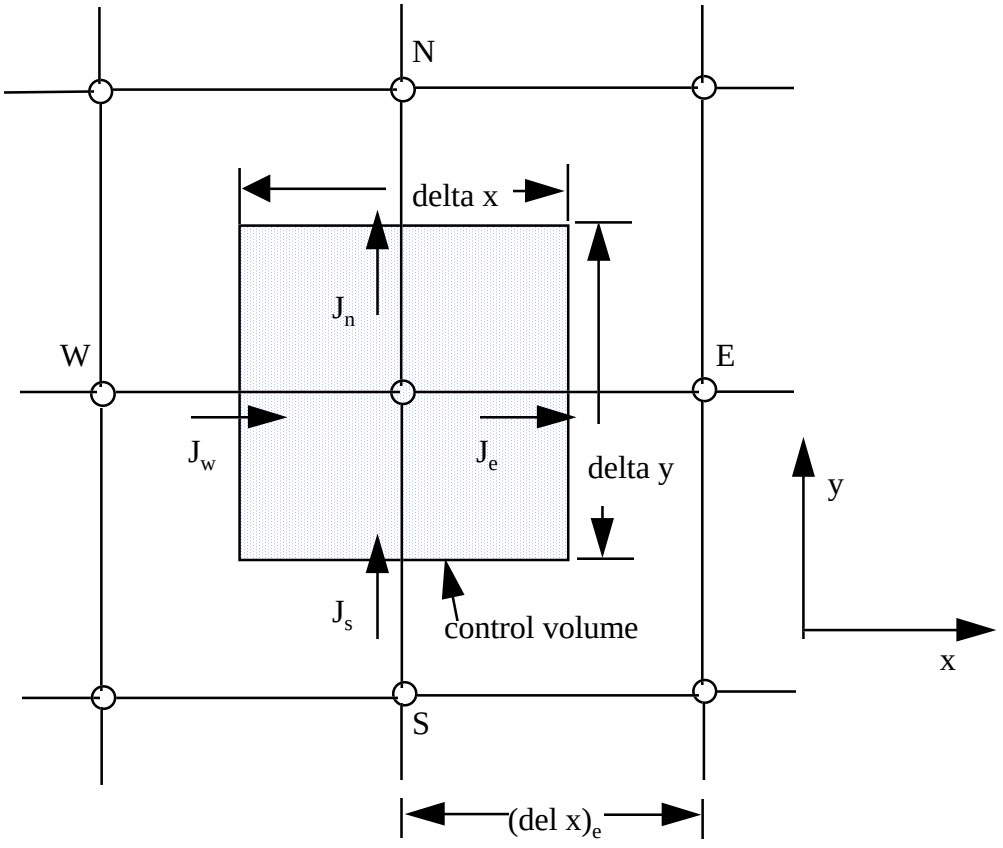

Figure 2.1: Representative control volume

$$\frac{\left(\rho_p\phi_p - \rho_p^0\phi_p^0\right)\Delta x\Delta y}{\Delta t} + J_e - J_w + J_n - J_s = \left(S_c + S_p\phi_p\right)\Delta x\Delta y \qquad (2.7)$$

where the superscript 0 denotes quantities from the previous time step, the source S has been decomposed into a linearized function of ϕ and the quantities J_e, J_w, J_n and J_s represent total integrated fluxes across the control volume boundaries.

Approximation of the total flux employs values of the dependent variable at neighboring points. One method of approximating derivatives that use these neighboring point values is central differences. Central differences can, however, lead to numerical instabilities during the course of iteration. Utilizing the definition for the flux defined above (eq. 2.7), the integrated control volume governing equation can be rewritten as

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b \tag{2.8}$$

with

$$a_E = D_e A\left(\left|P_e\right|\right) + \max\left(-F_e, 0\right)$$

$$a_W = D_w A\left(\left|P_w\right|\right) + \max\left(F_w, 0\right)$$

$$a_N = D_n A\left(\left|P_n\right|\right) + \max\left(-F_n, 0\right)$$

$$a_S = D_s A\left(\left|P_s\right|\right) + \max\left(F_s, 0\right)$$

$$b = S_c \Delta x \Delta y + \frac{\rho_p^0 \Delta x \Delta y \phi_p^0}{\Delta t}$$

$$a_P = a_E + a_W + a_N + a_S + \left(F_e - F_w + F_n - F_s\right) + \frac{\rho_p \Delta x \Delta y}{\Delta t} - S_p \Delta x \Delta y$$

where F ≡ ρu, $D \equiv \frac{\Gamma}{\delta x}$, $P = \frac{\rho u \delta x}{\Gamma}$ at the e and w faces, or $F \equiv \rho v$, $D \equiv \frac{\Gamma}{\delta y}$, and $P = \frac{\rho v \delta y}{\Gamma}$ at the n and s faces. The values are referred to as the convection coefficient,

diffusion coefficient and the cell Peclet number, respectively. The function $A(|P|)$ depends on the numerical scheme implemented for the convective term. They are:

1-0.5|P|	for the central difference scheme,
1	for the upwind scheme,
max(0,1-0.5|P|)	for the hybrid scheme,
$\max(0,(1-0.1\|P\|)^5)$	for the power-law scheme, and
|P|/[exp(|P|)-1]	for the exponential scheme.

Each of these schemes has various advantages and disadvantages. The central difference scheme suffers from inaccurate prediction of pressure in high gradient areas and even diverges when the geometry is relatively complex. Therefore, the central difference scheme is not useful for practical simulations. The upwind scheme is stable in the sense that it converges yet it consistently predicts lower pressure peaks and total bearing forces. The exponential scheme is rarely used because exponentials are considerably expensive to calculate. In practical simulations, the hybrid and power-law schemes are the most suitable due to the facts that they have comparable orders of accuracy to the other schemes and possess good convergence characteristics.

The resulting system of equations is solved using an alternating direction line sweep method (Patankar, 1980) coupled with a multi-grid algorithm.

MULTI-GRID METHOD

As mentioned above, the numerical solution of the Reynolds' equation employs the multi-grid method. The central idea of the multi-grid method is to take advantage of the maximum relaxation of the error in the solution of a linear system as it relates to the

grid size. It is well known that the error in a standard fixed grid problem decreases rapidly at first and then slows considerably in later iterations (Brandt, 1977). This is a function of the fact that frequency components of the error that have wavelengths on the order of the grid spacing are relaxed effectively while others are not. The implementation makes use of successively coarser or smoother grids so that all wavelengths of the error can be relaxed effectively.

The method currently employed is based on the FMG-FAS (Full Multi-Grid-Full Approximation Storage) scheme developed by Shyy and Sun (1993). This technique was developed to effectively deal with non-linear problems in contrast to the simpler correction storage scheme.

The converged solution to the linear system discussed in the previous section satisfies the following equation:

$$[\alpha_k]\underline{\Psi_k} = \underline{\xi_k} \tag{2.9}$$

where the coefficient matrix $[\alpha_k]$ and the source vector $\underline{\xi_k}$ are based on the final solution of $\underline{\Psi_k}$ but are estimated in the interim on the various grid levels.

The multi-grid method calculates the solution on the various grid levels successively. Calculation is performed on a given grid then this result is interpolated to a coarser grid, then again until the coarsest grid is reached, then the process is reversed until the calculation is again made on the finest grid. This process is known as a V-cycle. In the current implementation, V-cycles of various depths are made until the solution converges on the finest grid. The maximum number of grid levels employed is 5.

The multi-grid method has been shown to increase the speed of solution of linear systems considerably. The improvement for typical air bearing simulations is more than one order of magnitude (Hu, 1996a).

It should be noted also that the numerical solution of the Reynolds' equation takes place on a calculation grid whose size is specified at the outset of the calculation. This calculation grid can, however, have variable spacing. In fact, the spacing of the grid is controlled by a mapping of the gradient of the equation at the beginning of the calculation on an evenly spaced grid. By mapping the gradient and subsequently adjusting the spacing of the calculation grid points the solver can maximize use of the computational effort. This also allows the Reynolds' equation to be solved with various geometric boundary conditions without using an inordinate number of grid points (Lu, 1994).

CHAPTER 3

OPTIMIZATION INTRODUCTION AND BACKGROUND

INTRODUCTION

Optimization, generally stated, is the problem of trying to minimize (or maximize) a function subject to conditions on the independent or dependent variables. This function is typically referred to as the objective function. The conditions set on the variables are referred to as constraints. If there are no constraints set on the variables, the problem is referred to as unconstrained optimization and can be stated as:

$$\text{maximize or minimize } \left\{ f(x) \middle| x \in S \right\} \tag{3.1}$$

where S is a set of feasible solutions to the problem known as the search space, x is a single point within that set and f(x) is a real-valued function of x. Practical problems typically have constraints. A set of constraints reduces the size of the set S by limiting the number of feasible solutions. The mathematical formulation remains the same with

constraints except that the maximization or minimization occurs over a smaller set S, which typically complicates the problem.

An extensive body of literature exists on how to solve this problem, a general survey of which will be presented here.

If the objective function and the constraints in a problem are both linear combinations of the independent variables then the problem is referred to as a linear programming problem. Additional restrictions need to be specified on the set of feasible solutions and the function f. For example, S is a given compact set and $f: S \subset \Re^n \to \Re$ is a continuous function. If these conditions are satisfied then the problem can be solved deterministically with assurance that the optimal solution has been achieved. Typical methods employed are the simplex method and interior point methods. (Dantzig, 1963).

The next level of difficulty involves forms of the objective function that are quadratic in nature. The linearity of the constraints must be maintained, however, for the problem to remain under the heading of quadratic programming, which is a method that decomposes the problem into a form amenable to solution by the simplex method mentioned above. Again, when the optimization problem takes this form, its solution is readily found.

Beyond these cases, the optimization problem is generally non-linear. This means that the objective function has no distinct form and the constraints can take any form as well. Posed in this very general way, an optimization problem can quickly become intractable. One of the main difficulties in solving a problem posed in this manner is that of multiple optima. Even if the search space is a compact set and the objective function is

a continuous, smooth real-valued function, the optimum found only has the assurance of being locally optimal. That is to say, if the following conditions are satisfied

$$\frac{df}{d\underline{x}^{*}} = 0, \qquad \frac{d^{2}f}{d\underline{x}^{*2}} > 0 \tag{3.2}$$

the solution is only guaranteed to be the minimum in some neighborhood δ surrounding the point $\underline{x}^{*}$, i.e.:

$$f\left(\underline{x}^{*}\right) \leq f\left(\underline{x}\right) \qquad \forall \underline{x} \in \delta \cap S \tag{3.3}.$$

This presents a severe difficulty when one attempts to find the globally optimal solution of a problem, i.e., the absolute minimum or maximum of the objective function over the given search space. Outside of this neighborhood δ the value of the objective function could vary considerably. Dixon (1978) provided the following argument to demonstrate that any method designed to solve a global optimization problem needs an unbounded number of steps. For any continuously differentiable function f, any point $\underline{x}$ and any neighborhood N surrounding $\underline{x}$, there exists a function g that is continuously differentiable that satisfies the following equation

$$f + g = f \qquad \forall x \notin N \tag{3.4}$$

with the global minimum of $f + g$ being at $\underline{x}$. ($f + g$ is referred to as an indentation of f). Therefore, for any point $\underline{x}$, it cannot be guaranteed that it is not the global minimum without at least evaluating one point in every neighborhood N of $\underline{x}$. Since N can be chosen arbitrarily small, it follows that any method constructed to find the global optimum of a function numerically would require an unbounded number of steps.

Global optimization, as stated above, is the search for the absolute minimum or maximum of the objective function over the given search space. That is to say, for a real-valued objective function $f:\Re^n \to \Re$, the goal is to find a point $\underline{x}^* \in \Re^n$ such that

$$f\left(\underline{x}^*\right) \leq f\left(\underline{x}\right) \quad \forall \underline{x} \in \Re^n \tag{3.5}$$

A review of current methods can be found in Rinnooy Kan and Timmer (1986).

The algorithms employed in global optimization can be divided into two categories, deterministic and stochastic. These methods differ in one fundamental way, the determination of a new point starting from the current point in the search. For deterministic methods, the algorithm for choosing search points contains no random component. Stochastic methods, as the name suggests, includes a random component in the selection of search points. The advantage of deterministic methods is their ability to find optima when the objective function has a very definite structure. Stochastic methods have the advantage of applicability to a wider range of objective function types.

DETERMINISTIC METHODS

A sampling of previously and currently employed deterministic methods will be presented in this section.

The most well-known of these methods is the branch and bound method (Horst, 1986). A generalization of this method is the partition and search method. Both of these are taken from the field of combinatorial optimization.

At any point in the procedure there exists a partition of the search space S into subsets S_a ($S_a \in S$), a lower bound $LB(S_a)$ on $\min_{\underline{x} \in S}\left(f\left(\underline{x}\right)\right) \forall a \in S$, and an upper bound $UB\left(S_a\right) = f\left(\underline{\tilde{x}}\right)\ \left(\underline{\tilde{x}} \in S\right)$ which represents the smallest feasible solution found so far.

It is evident that subsets S_a for which $LB(S_a) \geq UB$ cannot contain the global minimum and can, consequently, be eliminated. Attempts can then be made to improve the value of UB. If after these improvements are made, any subsets S_a remain for which $UB - LB(S_a) > 0$, the partition can then be refined. The typical approach is to divide the subset which satisfies the relation $LB\left(S_a\right) = \min_{a \in S}\left(LB\left(S_a\right)\right)$ into finer partitions and repeat the process. Partition and search methods are costly to implement even in the case where the function can be evaluated analytically because calculation of UB and LB on the partitions is computationally expensive and the number of partitions can grow rapidly for a complicated function. Regarding convergence, the difficulty of finding the lower bounds for the partition chosen is comparable to finding the global minimum itself. Therefore, all algorithms of this nature necessitate additional conditions on the objective function being explored to be feasible to implement, e.g., if the function is convex determination of UB and LB within a single partition is simplified considerably.

COVERING METHODS

Several methods are derived from the assumption that the objective function is Lipschitz continuous and that the Lipschitz constant L is available *a priori*. These types of methods make use of the search information (x_k, $f(x_k)$), k = 1 to N to determine future search points from regions that have a high likelihood of containing the global minimum. These methods fall under the category of *methods of covering*. The goal is to construct a

series of functions that are piece-wise continuous to cover the entire search space. These functions are constructed to be convex such that the search within a single region is simplified considerably. A description of this type of algorithm can be found in Horst and Tuy(1987). An extension of this method to N-dimensional problems was put forth by Mladineo (1986). The approach consists of creating gradually closer piece-wise-differentiable approximations to the objective function. The method clusters around relative optima close to the global optima with some theoretical underpinnings to support this.

TUNNELING METHOD

An algorithmically simple implementation of a global optimization is the tunneling method. Levy, et. al. (1985) proposed this algorithm which consists of two steps: a minimization step and a tunneling step. These steps are performed repeatedly until a stopping criterion is satisfied. The first step is to find a local minimum, $\underline{x}_j^*$, starting from the initial point with any suitable local optimization procedure. The next step is the tunneling step. The purpose of the tunneling step is to find new starting points that will lead to new local minima. These starting points are determined by finding zeros of a tunneling function $T\left(x, f\left(\underline{x}_j^*\right)\right)$. A simple form of this function is

$$T\left(\underline{x}, f\left(\underline{x}_j^*\right)\right) = f\left(\underline{x}\right) - f\left(\underline{x}_j^*\right) \qquad (3.6).$$

Zeros of this function are guaranteed to provide regions that have a minimum that is at least as small or smaller than the previous local minimum. It is evident from the algorithmic description that finding zeros of the tunneling function poses a task that is nearly as difficult as finding the global minimum itself.

INTERVAL ANALYSIS

Originally proposed by Moore (1966), the method of interval analysis consists basically of dividing the search region into sub-regions and then disregarding the regions that do not have the global minimum. The procedure takes advantage of the field of interval analysis to reduce the sub-regions as the algorithm progresses. When the width of the remaining interval becomes sufficiently small, the algorithm terminates, taking the final value as the global optimum.

Interval analysis has been extended to multi-dimensional problems but suffers from some very serious drawbacks. The most severe of these is that the number of intervals necessary to properly evaluate a function grows precipitously with the dimension of the problem. Second, the interval mathematics becomes quite involved as the regions become geometrically complicated. Lastly, the iteration method for reducing the width of the interval may require several iterations at every step of the algorithm, which can add significantly to computational cost.

STOCHASTIC METHODS

From the discussion of the deterministic methods above, it is clear that more robust, computationally less costly methods are desirable. The trade-off made in the pursuit of these is the sacrifice of a convergence proof. Deterministic methods are well suited for objective functions that can be evaluated analytically. If the derivatives are costly to evaluate or the function itself has some stochastic component, stochastic methods are better suited.

The simplest stochastic global optimization technique is the *random search method*. Starting at the initial point $\underline{x}_j \in S \cap \Re^n$ the next point simply adds a random vector $\underline{r} \in \Re^n$ to determine the next search point. This random vector is derived from a probability distribution P_r. The only restriction on P_r for the method to converge in an asymptotic sense is that

$$\prod_{j=i}^{\infty}\left(1-\left(P_r\right)_j\right)=0 \tag{3.7}$$

where $(P_r)_j$ is the probability of reaching a point in the search space starting from the current point. Clearly from the definition if the algorithm is allowed to run indefinitely, any point in the search space can be reached and subsequently evaluated. This is known as asymptotic convergence and is a necessary condition for a random method to have practical value as a finite-time method. Each new point in the algorithm represents an improvement in objective function value over the previous point. Since it is chosen by

$$\underline{x}_{j+1}=\begin{cases}\underline{x}_j+\underline{r} & \text{if } \left(\underline{x}_j+\underline{r}\right)\in S \text{ and } f\left(\underline{x}_j+\underline{r}\right)\le f\left(\underline{x}_j\right)\\ \underline{x}_j & \text{otherwise}\end{cases} \tag{3.8}.$$

Using this algorithm and setting the additional restriction that f be a measurable function on a measurable set $S \subset \Re_n$, one can be show that (Solis, 1981) for $\varepsilon > 0$ and $M < 0$

$$\lim_{j\to\infty} P\left(\underline{x}_j \in O_{\varepsilon,M}\right)=1, \tag{3.9}$$

where

$$O_{\varepsilon,M}=\begin{cases}\left\{\underline{x}\in S \middle| f\left(\underline{x}\right)<\min f\left(\underline{x}\right)+\varepsilon\right\} & \min f\left(\underline{x}\right) \text{ finite}\\ \left\{\underline{x}\in S \middle| f\left(\underline{x}\right)< M\right\} & \min f\left(\underline{x}\right)=-\infty\end{cases} \tag{3.10}.$$

Typically $\underline{r}$ is dependent on the current state of the algorithm, i.e., it is chosen adaptively. Several variations of this method have been implemented, primarily by using different schemes through which $\underline{r}$ is chosen.

Bayesian methods, which will not be discussed at length here, can be viewed as an extension of random search methods with additional information regarding the probability distribution of the objective function over subsets of the search space S. For example, information such as the probability that the f(x), $x \in S$, belongs to a specific interval or sub-region $S_a \subset S$ can be exploited to create a loss function which, in turn, is minimized. Minimization of this loss function is the basis of the Bayesian method.

STOCHASTIC PATH FOLLOWING METHODS

A simple approach to finding the global minimum of a function is to follow the path of the gradient

$$\frac{d\underline{x}(t)}{dt} = -\nabla f\left(\underline{x}(t)\right) \tag{3.11}$$

To create a stochastic equivalent, a diffusive term is added to the right hand side of the equation

$$\frac{d\underline{x}(t)}{dt} = -\nabla f\left(\underline{x}(t)\right) + \varepsilon_0 \frac{d\underline{w}(t)}{dt} \tag{3.12}$$

where w(t) is an n-dimensional Wiener process. (A Wiener process is defined as $R(x_1, x_2) = \sigma^2 \min\{x_1, x_2\}$.) The solution of this diffusive, stochastic differential equation is a probability density function which is asymptotically proportional to

$$p(\underline{x}) = \exp\left(\frac{-2f(\underline{x})}{\varepsilon_0^2}\right) \tag{3.13}$$

as $t \to \infty$. Assuming that $f(x) > 0$, $p(x)$ approaches a Dirac delta function centered at the global minimum as $\varepsilon_0 \to 0$, independent of the initial point $\underline{x}_0$. This is because the volume integral of the probability density function $\int_S \exp\left(-2f(\underline{x})/\varepsilon_0^2\right) < \infty$, due to the fact that the probability is proportional to the probability of the entire space S, which is clearly equal to one.

The distribution arrived at above is the same distribution of probability that is employed in the simulated annealing algorithm. The value $\frac{1}{2}\varepsilon_0^2$ is the equivalent of the 'temperature' in the simulated annealing algorithm and the time, t, represents the annealing time. Therefore, at a fixed value of the 'temperature' with the value of t approaching infinity, the probability of finding the global minimum is proportional to the probability distribution given above. The simulated annealing algorithm and the method of genetic algorithms will be discussed in the following section. In Chapter 5, the simulated annealing algorithm will be discussed in depth as well as the current implementation.

CONCLUSIONS

This section has introduced a wide range of optimization methods. Beginning with simple, linear methods up to non-linear, global methods. The basis for the simulated annealing algorithm--which is used extensively in this treatise--has been laid relative to

other global optimization techniques. Further background and exposition will be made manifest in the next two chapters.

CHAPTER 4

SIMULATED ANNEALING AND GENETIC ALGORITHMS

INTRODUCTION

This section describes Genetic Algorithms (GA) and the Simulated Annealing Algorithm (SAA). A comparison between the two methods is made to determine suitability for the air bearing design problem. An example problem is provided to demonstrate the ability of the algorithms to find the global optimum of a given cost function. From the application of these to examples, a determination is made that the SAA is better suited for slider air bearing optimization than the GA.

BACKGROUND

AIR BEARING SIMULATOR

Developments in plasma etching and similar fabrication techniques have made virtually arbitrary air bearing surface geometries feasible. With this control of geometry,

reduction of slider fly height variation across the radius of a disk drive as well as other performance objectives can be satisfied. This increase in design flexibility brings a new challenge, namely, finding the optimal configuration of the air bearing surface. Two methods are presented here for solving this problem given some basic design constraints. The methods, genetic algorithms and simulated annealing, are iterative optimization techniques that do not rely on the (numerical) calculation of gradients to achieve optimal solutions. The methods are shown to be highly effective.

Design optimization, or the problem of finding an optimal configuration given performance objectives and constraints, is of substantial interest. With the developments in fabrication technology and the increasingly critical requirements of fly height uniformity, among other objectives, optimization has become an important goal for slider design. The difficulties that are presented include the time for evaluation of the objective function (solution of Reynolds' equation for a given configuration), noisy numerical data, which precludes the effective use of traditional gradient-based optimization techniques, and satisfaction of multiple objectives through control of multiple design parameters.

Briefly, genetic algorithms encode the parameters in a problem as “gene strings” and then through two operations, crossover and mutation, in addition to objective function evaluation, achieve a family of optimal strings which are then decoded into the optimal, or nearly optimal, solutions. The original “gene strings” represent a random distribution of the parameters. Adaptive simulated annealing also starts with a random configuration. Its performance is evaluated and then a point in the neighborhood of this

configuration is chosen and evaluated. An acceptance criterion is then invoked to determine whether or not the new point will be kept as the starting point for the next neighborhood operation. If not, the original point is retained and a new point in the neighborhood is found. Ideally, this allows the global optimum to be found.

As described earlier, the underlying equation for calculation of the fly height of sliders is the Reynolds' lubrication equation. It is a simplification of the Navier-Stokes equations based on certain assumptions regarding the nature of the flow field under the slider. The optimization programs presented here implement the Computer Mechanics Laboratory air bearing design program. This simulator solves the Reynolds' equation through use of a multi-grid method as described earlier. In addition, the simulator accounts for arbitrary bearing geometry by using a flow averaging method. These features allow arbitrary configurations to be evaluated relatively quickly, which is essential to the optimization procedure.

GENETIC ALGORITHMS

The idea of evolution was presented by Darwin in the classic *The Origin of Species* in 1859. It was not until the early 1970's that this idea was incorporated effectively into a computer program by Holland (1975). The basic idea of GAs is to represent, or “encode,” parameters in a problem as a string of data, typically binary. These strings are then combined and “mutated” in an attempt to achieve the most fit strings possible based on some performance measure. This idea is taken directly from evolution where human genes combine and mutate to form new genes. The fittest

parameter strings in a problem correspond to optimal solutions of the objective function that is being employed.

GAs are typically employed when other, more standard, methods of optimization are not feasible or simply would not work. Examples include exceedingly high dimensional problems, numerical simulation results, machine learning and noisy data fields.

An example of a generational genetic algorithm is given below (Figure 4.1).

```
Initialize random population
Evaluate population
While (counter < total
generations)
        Increment counter
        Perform selection
        Perform operations
        Evaluate population
Loop
```

Figure 4.1: Outline of generational genetic algorithm

The first step in implementing GAs is the representation of the parameters in the problem as strings. Typically binary strings are used for simplicity. When this is done a continuous problem is reduced to a discrete one. The resolution, which determines the smallest achievable change in a parameter, can be controlled by the user down to a computer-dependent limit. The underlying purpose of this is to allow the optimization algorithm to operate on these strings in a manner similar to the way genes crossover and mutate in biological entities. In the study presented here each range of the parameters is simply divided into equal segments and mapped into binary strings. The main purpose of mapping parameters into binary strings is to ease the crossover and mutation processes.

After the parameters in the problem have been put into a string representation, or "encoded," the optimization algorithm can begin.

In order to initialize the population, a random set of parameters is chosen for a single member of the first generation, and then this process is repeated until the entire first generation has been populated, i.e., until the pre-determined population size has been initialized with random configurations. The generation (population) size and number of generations evaluated ideally will yield the minimum number of function evaluations needed to achieve the global minimum. Since this number cannot be known *a priori*, there are some guidelines for choosing their values. This procedure can be modified if there is a reason for choosing a specific region of the search space in which to concentrate the population originally.

Once the initial population has been determined it is then evaluated using a fitness function. The fitness function assigns a real value to each member of the population. If minimization is being attempted, as in the current investigation, then a fitness function can be defined as

$$u(\underline{x}) = F_{\max} - f(\underline{x}) \tag{4.1}$$

where F_{max} is the greatest value of f(x), the cost function, over a certain number of previous generations. In the first few generations F_{max} is simply drawn from a smaller pool. This type of fitness function implicitly employs a technique called "window" scaling. That is to say, F_{max} is found from a certain number of previous generations called the "window." One disadvantage of window scaling is that if a single member of the population has an extremely high cost (poor fitness) then it will have the effect of making

it difficult to distinguish between the remaining members when it comes to selecting which are the best and how much better they are than the others. This is an important disadvantage in the selection process as we will see below. A way to ameliorate this problem is to use "sigma scaling." In sigma scaling the fitness of a member of the population is defined as

$$u(\underline{x}) = \frac{s + \sigma_{f(\underline{x})}}{s} \tag{4.2}$$

where s is the sigma scaling factor and $\sigma_{f(x)}$ is the cost value measured in standard deviations. A typical value for s is 2. This method has the effect of limiting the severe disadvantage of window scaling while still allowing useful selection to occur. The performance evaluations at the beginning of the algorithm and within the generational (iterative) loop are the same.

The first task in the generational loop is to select members for the current generation from the previous generation based on performance. Given the fitness for a certain chosen string, a probability that this string will be present in the next generation is calculated, i.e., strings that are more fit (more optimal) will have a higher probability of appearing in the next generation than strings that are less fit (less optimal).

The simplest possible method for selecting the new generation is called the "spinning wheel" method. A brief outline of this algorithm is (Figure 4.2):

Sum the fitness' of the current generation
Divide each individual fitness by the sum
Map each of these ratios continuously to the segment [0,1], i.e., the first ratio r_1 would cover $[0,r_1]$, the second $(r_1, (r_1+r_2)]$, etc.
Choose a random number between [0,1]
Select the member that corresponds to that number
Repeat until new generation is fully populated

Figure 4.2: Outline of "spinning wheel" method of selection for genetic algorithms

The advantage of this method is that it has zero bias, where bias is defined as the difference between the actual sampling probability of a population member and its expected value. Therefore, zero bias is achieved when the probability of a member is equal to its expected value. The problem with this technique is that a single member could fully populate the new generation regardless of its expected value. This method is also known as "Stochastic Sampling with Replacement." One way to deal with this problem is to choose the random number and then divide the segment into equal partitions with the random number as an offset from zero. This insures that members with an expected value greater than one partition will have at least one representative in the new generation, as seen in the diagram below (Figure 4.3); where each arrow represents a selection of the next generation.

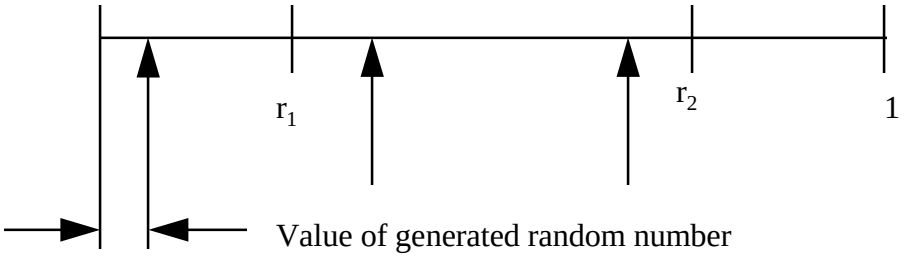

Figure 4.3: Depiction of "stochastic universal sampling" method of selection for genetic algorithms

The above diagram (Figure 4.3) illustrates a simplified selection procedure. Once the random number has been generated, the entire population is selected simultaneously. Since each arrow represents a member of the next generation, we can see that the member of the current generation represented by the line segment between r_1 and r_2 will have two members in the next generation. This is the method employed in the current investigation and it is also known as "Stochastic Universal Sampling." Other methods of selection are available but will not be discussed any further here (Baker, 1987).

In the algorithm sketched above, the most important step is the one entitled "operations." It consists of "crossover" and "mutation." Crossover occurs when two string representations are combined to create two new strings by appending the ending part of one string to the beginning part of another. The point at which the first substring ends and the second begins is constant for our particular implementation of genetic algorithms. For example, in the following illustration (Figure 4.4) the strings on the left are crossed over to obtain the strings on the right.

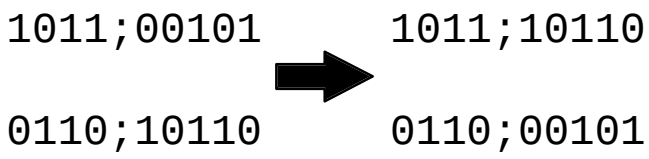

Figure 4.4: Example of GA crossover procedure

In this illustration, the crossover point is between the fourth and fifth bit positions. Ideally, crossover allows for optimal sub-blocks, or "schemata," to be combined into optimal strings that will represent the optimal solution. This implies that GAs are based on the assumption that optimal schemata will lead to optimal strings. If this is not the case then the chances of a GA converging to the optimal solution are severely hampered.

Multi-point crossover schemes and others are discussed in Schraudolph (1994).

SIMULATED ANNEALING

Annealing in material technology is the process of raising the temperature of the temperature, which is a measure of the mean energy, and then slowly cooling it to allow it to reach its lowest energy state. Simulated Annealing (SA) emulates this process in an attempt to find the global optimum of an objective, or "energy," function.

In physical annealing the temperature of the process is controlled. In SA an analog of the temperature is controlled to allow the simulation to progress. For convenience we will call this control parameter "T." We will call the measure, i.e., cost function, of the current state of the parameters $C(p_k)$, where p_k is the current state. The significance of the control parameter T is that it will be used to indirectly decide how much of an increase in the cost will be accepted from one change of state to another. So, we start with a random configuration and measure its cost.

Using a simple equation of the form:

$$p_{k+1}^{i} = p_{k}^{i} + y^{i}\left(B_i - A_i\right) \tag{4.3}$$

where y^i is a random variable between -1 and 1, i.e.,

$$y^{i} \in [-1,1] \tag{4.4}$$

with a probability density equal to

$$g^{i}\left(y^{i};T_i\right) = \frac{1}{2\left(\left|y^{i}\right| + T_i\right)\ln\left(1+\frac{1}{T_i}\right)} \tag{4.5}$$

where T is the parameter temperature, and A_i and B_i are the lower and upper limit for a given parameter p^i, i.e.,

$$p^i \in [A_i, B_i] \tag{4.6}$$

we choose the next state of a given parameter (assuming it satisfies the above constraint, eq. 4.6).

Below (Figure 4.5) is an illustration of the probability density g as a function of y and T. For a given value of T_i, the probability distribution of y_i is shown.

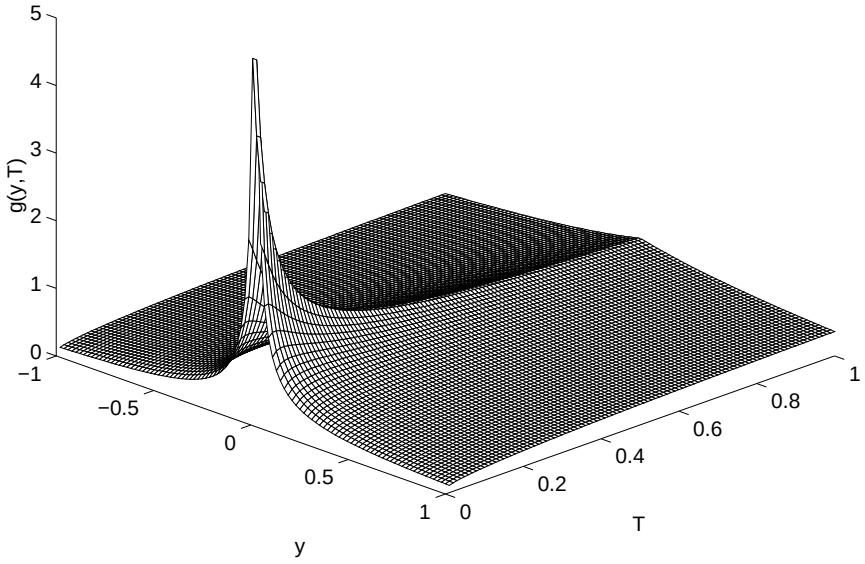

Figure 4.5: Plot of probability distribution for new parameter selection for simulated annealing algorithm

Clearly, as the parameter temperature is reduced, the chances of finding a configuration in the neighborhood closer to the current configuration are increased markedly.

After this process is completed for all of the parameters, the cost of the new current state of the parameters is evaluated, i.e., $C(p_{k+1})$. Once the cost is evaluated, it can be determined if the new state will be the starting point for future states or if the

previous state will again serve as the starting point. This determination is made using the following rules: if the new cost is less than the previous one then it is kept unconditionally, if the new cost is more, meaning it is less optimal, then it is kept if it meets the condition

$$\exp\left[-\frac{\left(C(p_{k+1})-C(p_k)\right)}{T_{cost}}\right] > U \tag{4.7}$$

where T_{cost} is the current value of the control parameter, or temperature, T, and U is a randomly generated number subject to the restriction

$$U \in [0,1) \tag{4.8}$$

So, when the value of the control parameter T is very large then essentially all configurations are retained as the starting point for calculation of new configurations. This allows the configurations to map the space initially. It is analogous to the physical process of melting where all atoms are heated to allow for nearly random motion. As the temperature is reduced, the chances for accepting a configuration with a higher cost decrease commensurately. The final point is chosen as the point at which no further configurations are accepted for a set number of "steps," i.e., a set number of attempted calculations of new configurations is attempted without any of them meeting the above criterion.

COOLING SCHEDULE

Control of the temperature T has the greatest impact on convergence. Boltzmann annealing, where the temperature is reduced according to

$$T(k) = \frac{T_0}{\ln(k)} \tag{4.9}$$

where k is the current step, is expected, statistically, to achieve a global minimum. Fast annealing reduces the temperature according to $T(k)=T_0/k$. Fast annealing, when coupled with the proper generating function, can be shown to achieve the global minimum of an objective function in probability (Ingber, 1994).

EXAMPLE

The algorithms presented above are used to find the optimal configuration of the air bearing surface on a Transverse Pressure Contour type slider. The one-dimensional case has the high pressure air bearing surfaces moving in unison and the two dimensional case allows them to move independently.

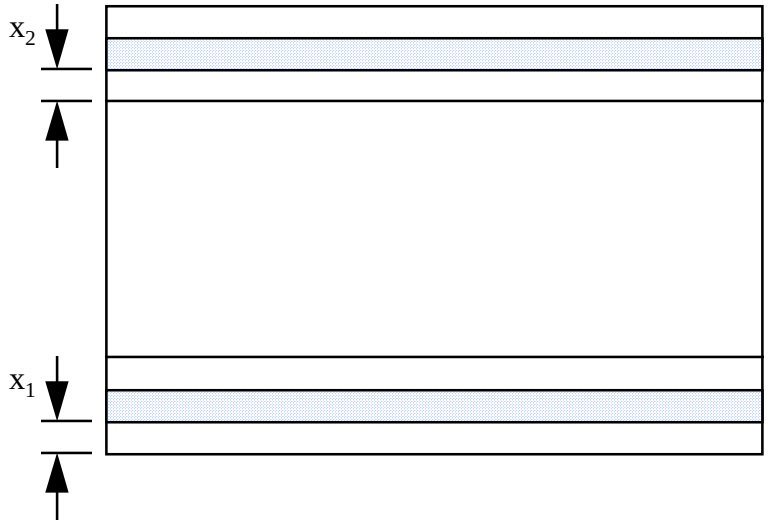

Figure 4.6: Parameters used in optimization

In the above diagram (Figure 4.6) for the one-dimensional case the rails (high pressure air bearing surfaces) are constrained to move in unison, i.e., $x_1=x_2$ and in the two-dimensional case they are allowed to move independently.

The cost of each configuration is defined as the maximum difference in fly height between three points, located at the inner, center and outer radius of the disk, with corresponding skews multiplied by a scaling factor, i.e.,

$$Cost = 10^9 \times \Delta FH \qquad (4.10).$$

For the calculation, the radius of curvature of the head path of travel (suspension arm plus actuator arm) was taken to be 44.72 mm., the radius points and corresponding skews were:

Radius (mm)	Skew (deg)
20	0.0
24	-5.26
28	-9.87

Table 4.1: Table of radius and skew values used for example

The other relevant design parameters used, but not varied, are:

Parameter	Value
slider length	2.05 mm
slider width	1.6 mm
rpm	5400
loading force	3.0 g
taper length	0.205 mm
taper height	4.0 microns
rail recess depth	1.5 microns
fully relieved region recess depth	100 microns
crown, camber, twist	0.0, 0.0, 0.0

Table 4.2: Table of non-varied parameters used in example

NUMERICAL RESULTS

The one and two dimensional cases were optimized using both genetic algorithms and simulated annealing. For the purposes of comparison, the genetic algorithm used was GAUCSD1.4 (Schraudolph, 1994) and the simulated annealing package used was Adaptive Simulated Annealing (Ingber, 1994). In the following chapters, the algorithm developed during the current investigation is explicated and all subsequent results are obtained through its use. Further, in the appendices, a user's manual and example problem are provided.

For the one-dimensional case, the genetic algorithm was implemented with the following control parameters:

Control Parameter	Value
population size	10
total trials	80
crossover rate	0.60
mutation rate	0.052
sigma scaling factor	2.00

Table 4.3: Table of control parameters used with genetic algorithm for example problem

The population size is the number of solutions under consideration for any generation. The total trials is the upper limit of how many separate configurations will be evaluated. The crossover rate is the probability of a solution from the current generation being combined with another member of the current generation to form two new configurations as depicted (Figure 4.4) above. The mutation rate is the expectation of any bit in any member of the population being flipped. The sigma scaling factor is the number used in evaluating the fitness of a given configuration.

In the one-dimensional problem, the binary string employed to represent the parameter was 8 bits in length. That implies that there were 2^8 , or 256, possible discrete values that it could take. The parameter can take on values between 0.0 and 0.096585 so, when it is represented by the 8 bit string, the minimum change in the parameter is 0.773 microns. This is determined by dividing the entire range by the total number of possible values then multiplying by a scaling factor (the slider length) to obtain an actual physical dimension.

The resulting family of optimal solutions, defined as the ten best, is shown below against a plot of objective function value versus parameter value (Figure 4.7). This

clearly shows that even in this very simple case--one that does not take advantage of the power of genetic algorithms--the optimal solution is still achieved.

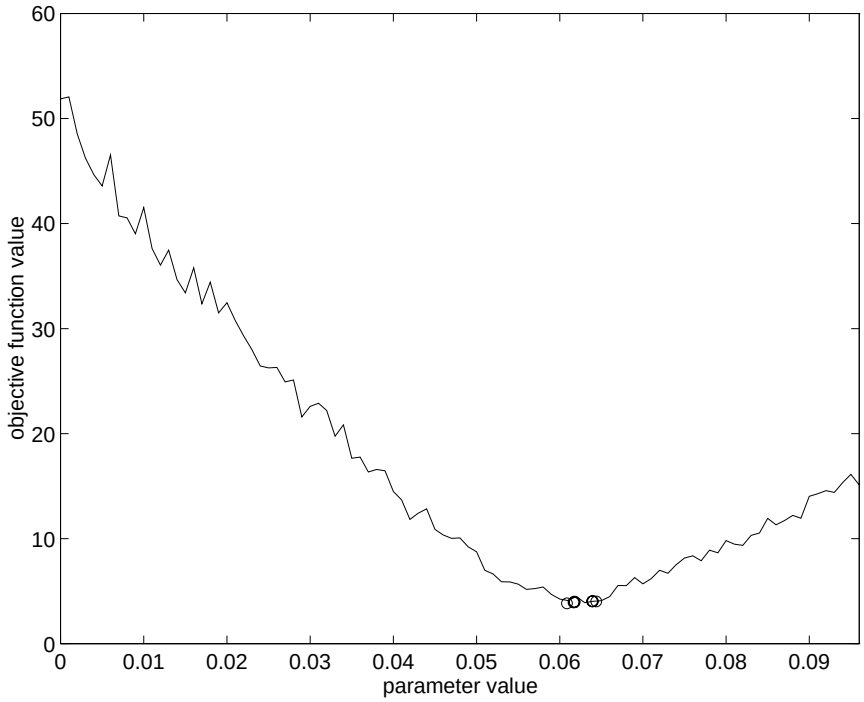

Figure 4.7: Plot of objective function value versus parameter value including ten best configurations obtained from genetic algorithm (denoted by o's)

The 10 best solutions obtained are shown below (Table 4.4):

Parameter 1	Objective function value (nm)
0.0639847	4.0366
0.0617599	3.9424
0.0608742	3.8359
0.0639125	4.0364
0.0616585	3.9396
0.0639027	4.0352
0.0639576	4.0362
0.0617673	3.9424
0.0617462	4.0003
0.0643863	4.0175

Table 4.4: Table of results for one-dimensional problem using GA

These values are the position of the rail, as described in the problem definition section, normalized by the length of the slider.

For the one-dimensional case, the simulated annealing method uses the following cooling schedule for each of the parameters:

$$T_i(k_i) = T_{0i} \exp\left(-c_i k_i^{\frac{1}{D}}\right) \tag{4.11}$$

where k_i is the current step, D is the number of parameters (dimension) and c is the normalizing constant calculated from user supplied values. The cooling schedule for cost acceptance is similar except that the current step is the number of previously accepted configurations, i.e., the total number of generated configurations minus the number of rejected configurations.

The other relevant parameters used in the optimization were (Table 4.5):

Control Parameter	Value	Brief Description
accepted configuration limit	1000	maximum number of configurations accepted
generated configuration limit	99999	maximum number of configurations created
invalid configuration limit	1000	maximum number of invalid configurations allowed
cost precision	1.0×10^{-5}	tolerance for repeated cost values
maximum repeated cost limit	5	maximum number of times neighbor configurations will be evaluated before termination

Table 4.5: Table of control parameters used by GA

The solution obtained is shown against a map of the objective function versus parameter value (Figure 4.8).

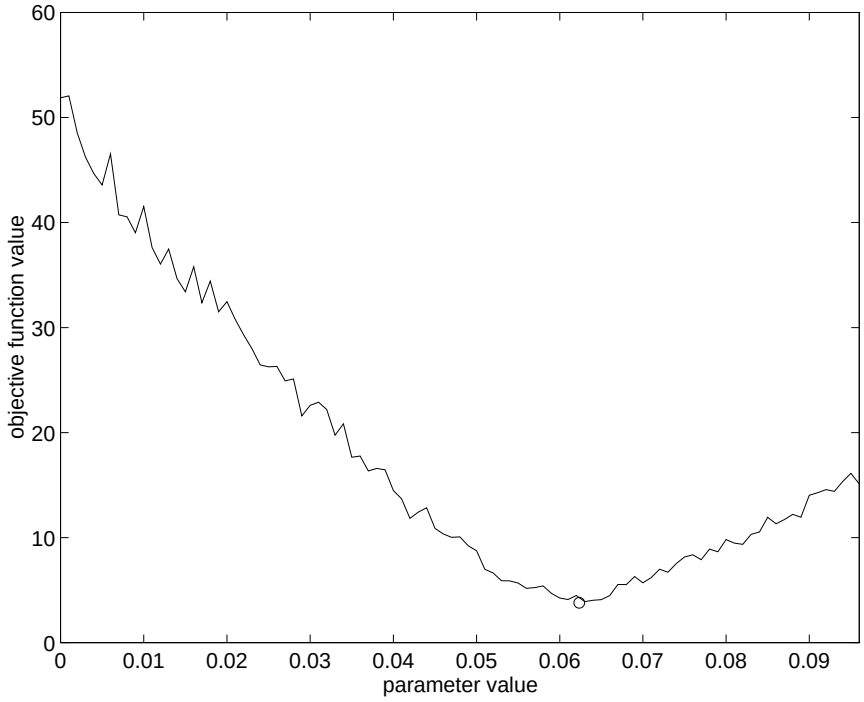

Figure 4.8: Plot of objective function value versus parameter value including best configuration found from simulated annealing algorithm

It is seen that the optimal solution was again obtained. The value obtained by the simulated annealing method was 0.06235251 that had an corresponding objective function value of 3.79.

The two algorithms were also implemented in a similar way to the two-dimensional case. For the GA, the resolution of the second parameter was the same as the first. The simulated annealing method implemented the same cooling schedule. Both algorithms used the same control parameters as in the one dimensional case.

The graph below (Figure 4.9) shows that the genetic algorithm and the simulated annealing method both achieved solutions in the optimal regions.

The values on the contour lines represent the values of the objective function as a function of the two parameters. The solutions for the genetic algorithms (represented by asterisks) were:

Parameter 1	Parameter 2	Objective function value (nm)
0.0435419	0.0750172	3.9863
0.0190679	0.0939901	4.8948
0.0359568	0.0809572	3.9192
0.0212428	0.0936637	4.0895
0.0232145	0.0959719	4.2102
0.0545375	0.068694	4.4822
0.019019	0.0939986	4.8920
0.0545282	0.0638383	4.1667
0.0544756	0.0686282	4.4852
0.0573867	0.0749233	4.8454

Table 4.6: Table of optimal solutions obtained by GA for two-dimensional problem

The solution obtained from the simulated annealing method (represented by the circle) was (0.09609084, 0.05069649) which had a corresponding objective function value of 3.0549. Similar to the one dimensional case, these values are the positions of the rail, as described in the problem definition section above, normalized by the length of the slider.

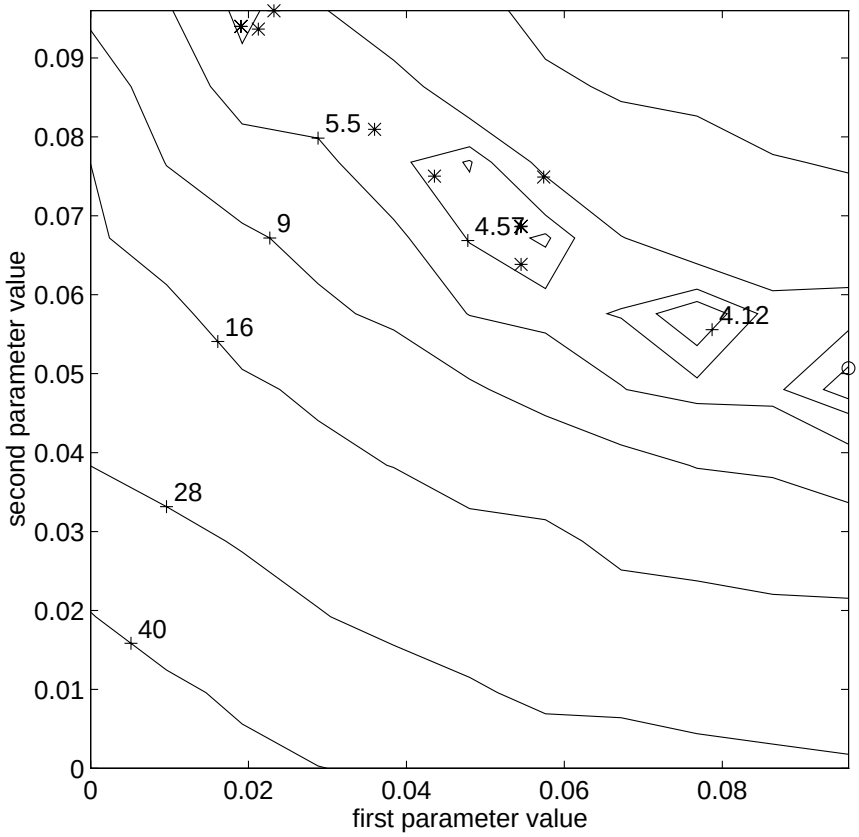

Figure 4.9: Contour plot of objective function versus both parameters employed including optimal solutions found by genetic algorithm (*) and simulated annealing (o)

Note that the GA solutions are concentrated in the basins of attraction (regions where contour lines are closed) and that the simulated annealing solution is on the minimum contour. This result is typical of both algorithms.

A more careful evaluation of the results shows that the genetic algorithm solutions are spread evenly through a nearly optimal region and the simulated annealing algorithm found the global optimum (Figure 4.9). In an attempt to improve the quality of the genetic algorithm solutions, they were allowed to run for 12 and 16 generations, starting from the same point, for comparison to the original 8 generation case. In the 12 generation case, the best ten solutions had three members that were better than three members of the 8 generation case, however, the most optimal solution was not improved.

Parameter 1	Parameter 2	Objective function value (nm)
0.0529843	0.0647321	4.2265
0.0304041	0.0897645	4.1879
0.0362133	0.0807856	4.2742

Table 4.7: Three solutions for the 12 generation genetic algorithm run that exceeded those of the 8 generation case

In the 16 generation case, the ten best solutions had two members that were better than two members of the 12 generation case, however, as in the 12 generation case, the most optimal solution was not improved.

Parameter 1	Parameter 2	Objective function value (nm)
0.0347563	0.0807346	3.9250
0.034794	0.0896046	4.1642

Table 4.8: Two solutions for the 16 generation genetic algorithm run that exceeded those of the 12 generation case

CONCLUSION

Results for the one dimensional and two-dimensional optimization of a Transverse Pressure Contour slider using genetic algorithms and simulated annealing are presented. In these problems both algorithms achieved the goal of finding a globally optimum solution, or nearly globally optimum solutions, based on a comparison with the graphs of the objective function.

The results of the two dimensional case imply that the simulated annealing method is more effective at finding global optimums. We should note, however, that the resolution of the parameters in the genetic algorithms was roughly 0.00038 in the

normalized coordinate system (used for plotting the graphs of the objective function) which may not allow it to capture the very intricate details of the objective function topology that are ostensibly present in the fine grid graphing. In comparison, the simulated annealing technique can take a step of any length, only limited by the parametric constraints above and the machine precision below. Given this, it follows that the simulated annealing algorithm found the global optimum in the solution space, whereas, the genetic algorithm was confined to a minimum region that appeared as a global optimum on the length scale that it could resolve.

Due to this inherent constraint on resolution that exists for genetic algorithms, their combinatoric nature and the fact that the objective function is evaluated by a numeric solver, the simulated annealing algorithm is better suited to the problem of slider air bearing optimization.

CHAPTER 5

SIMULATED ANNNEALING IN DEPTH

INTRODUCTION

In the previous chapter, it was demonstrated that the simulated annealing algorithm is well suited for global optimization of air bearing slider designs. In this chapter a deeper look will be taken at the parameters that affect the optimization procedure and what values are optimal for the slider air bearing problem. Several examples will be tracked in depth in an attempt to elucidate the process that leads to optimal designs. Additionally, in the cooling schedule section, heuristic proofs will be given for three variations of the simulated annealing algorithm.

SIMULATED ANNEALING

As described in the previous chapter, annealing is the process of raising the temperature, which is a measure of the mean energy, of a material and then slowly cooling it to allow it to reach its lowest energy state. Simulated Annealing (SA) emulates

this process in an attempt to find the global optimum of an objective, or “energy," function. Simulated Annealing was originally introduced by Metropolis, N., et. al. (1953) to solve high dimensional path integrals that arise in statistical physics problems through a Monte Carlo technique. Subsequently, it has been used for a variety of non-convex, non-linear optimization problems, e.g., determining the optimal layout for circuits on a computer chip (Kirkpatrick, et. al., 1983).

Initially, the algorithm was used to solve combinatorial optimization problems, i.e., problems where the parameters have discrete values and the possible combinations of them, while large, was countable. In the current implementation, the parameters are allowed to vary continuously. This implementation of the algorithm is entitled Continuous Simulated Annealing. The essential difference is that the algorithm is derived from a stochastic descent algorithm, i.e., a differential equation that includes the gradient direction and random, white decreasing Gaussian noise. It is this last quantity that keeps the algorithm from getting stuck in a local minimum. Gelfand and Mitter (1991) have provided a convergence proof for this sort of algorithm that states for infinite time steps (iterations of the algorithm) the method converges to the global minimum in probability. There are not, however, conclusions for the finite time behavior of the algorithm. The requirement that the algorithm be consistent for infinite time behavior has, at the minimum, been met.

In physical annealing the temperature of the process is controlled. In SA an analog of the temperature is controlled to allow the simulation to progress. For convenience we will call this control parameter “T." We will call the measure, i.e., cost function, of the current state of the parameters $C(p_k)$, where p_k is the current state. The

significance of the control parameter T is that it will indirectly decide how much of an increase in the cost will be accepted from one change of state to another and will determine the probability distribution for the neighborhood function which generates new states starting from the current state. In the implementation used here, the control parameter T has been split into two separate parameters. There is a T associated with the cost (and acceptance of new configurations) and a T associated with parameter generation. This allows for separate cooling schedules to be implemented for each. The functional form of the cooling schedule is, in fact, the same. The only difference is its indexing which is called "annealing time," or the number of steps taken so far in the optimization. For the T associated with the cost (T_{cost}) the index in annealing time is the number of configurations "accepted" so far (a concept which will be made clear below) and the T associated with the current configuration (T_{param}) is indexed by the total number of configurations generated. So, we start with a random configuration and measure its cost.

Restating the algorithm as it appeared in the previous chapter, we begin with a random configuration of the parameters:

$$p_{k+1}{}^{i} = p_{k}{}^{i} + y^{i}\left(B_{i} - A_{i}\right), \tag{5.1}$$

where y^{i} is a random variable between -1 and 1, i.e.,

$$y^{i} \in [-1,1], \tag{5.2}$$

with a probability density equal to

$$g^{i}\left(y^{i}; T_{i}\right) = \frac{1}{2\left(\left|y^{i}\right| + T_{i}\right)\ln\left(1 + \frac{1}{T_{i}}\right)} \tag{5.3}$$

where T is the parameter temperature, and A_i and B_i are the lower and upper limits for a given parameter p^i, i.e.

$$p^i \in [A_i, B_i], \tag{5.4}$$

the next state of a given parameter is chosen randomly (assuming it satisfies the above constraint, eq. (5.4)). The joint probability distribution g given above, also known as the generating function, is based on the functional form derived for many physical systems belonging to the class of Gaussian-Markovian systems. The probability density corresponds to the state space of the parameters that exist within the problem.

An illustration of the probability density g as a function of y and T is provided in the previous chapter. For a given value of T_i, the probability distribution of y^i can be seen. The significance of the given probability distribution can be seen in an analysis of the algorithm's capability of statistically achieving the global optimum.

After this process is completed for all of the parameters, the cost of the new current state of the parameters is evaluated, $C(p_{k+1})$. From this, it can be determined if the new state will be the starting point for future states or if the previous state will again serve as the starting point. This determination is made using the following rules: if the new cost is less than the previous one then it is kept unconditionally, if the new cost is more, meaning it is less optimal, then it is kept if it meets the condition

$$\exp\left[-\frac{\left(C(p_{k+1}) - C(p_k)\right)}{T_{cost}}\right] > U \tag{5.5}$$

where T_{cost} is the current value of the temperature associated with the cost and U is a randomly generated number with uniform distribution subject to the restriction

$$U \in [0,1). \tag{5.6}$$

As mentioned above, this is the concept of acceptance. The current point is the most recently accepted point. It is from here that the algorithm continues. One difficulty specific to the slider air bearing problem is that certain configurations of the parameters in a problem can create designs that do not support the load applied to the back of the slider, i.e., the slider crashes, or equivalently stated, does not fly. This difficulty is handled by keeping track of the worst configuration to date, i.e., the one with the highest objective function value, and assigning its value to this configuration. This is one method of insuring that any configuration that does not fly will not be considered more optimal than the worst configuration so far that can fly. This scheme, however, presents difficulties of its own. If the parameters in a problem are chosen with ranges that create a preponderance of configurations that are not capable of flying, the optimization process will be hindered. This is caused by the inability of the algorithm to choose new configurations to accept and subsequently move from. This in turn leads to a delay in the annealing time sequence. This effects the temperature that is associated with the cost but not the temperature associated with the parameters. So, eventually, when the new configurations that are being generated are sufficiently close to the current configuration--which flies--they, too, should fly. This is at the expense, however, of being able to sufficiently explore the search space near the inception of the algorithm. This effect will be demonstrated later in the section through use of an example problem.

At the beginning of the calculation when the value of the control parameter T is very large then essentially all configurations are retained as the starting point for calculation of new configurations. In the absence of large number of configurations that

do not fly, the algorithm is able to map the space initially. It is in this part of the optimization procedure that "hill climbing" (acceptance of less optimal configurations as the new starting point) is most likely to occur. As the temperature is reduced, the chances for accepting a configuration with a higher cost decrease concomitantly. The final point is chosen as the point at which no further configurations are accepted for a set number of iterations in annealing time or the temperature reaches some pre-set minimum. This is a manifestation of the limitation of global optimization algorithms. The value or location of the global optimum is not known *a priori* and there is not a suitable test to determine if the current configuration is the global minimum. Consequently, some other stopping criterion must be employed.

COOLING SCHEDULE

Control of the temperature T has the most bearing on convergence. Boltzmann annealing, where the temperature is reduced as

$$T(k) = \frac{T_0}{\ln(k)}, \tag{5.7}$$

where k is the current point in annealing time, is expected, statistically, to achieve a global minimum when coupled with the equation defining the appropriate generating function g. The significance of coupling the cooling schedule with the generating function is that they jointly determine the probability of the algorithm reaching any given point in the search space. The generating function employed in Boltzmann annealing is

$$g(\underline{x}) = (2\pi T)^{-D/2} \exp\left(\frac{-\Delta \underline{x}^2}{2T}\right) \tag{5.8}$$

where D is the dimension of the problem, and $\Delta\underline{x} = \underline{x} - \underline{x}_0$ is the distance of the current point from the previous configuration or some other reference configuration.

In order to insure that any point in the search space can be reached in the course of the optimization starting from the current point, k_0, it suffices to show that the probability of not reaching the point in question is zero

$$\prod_{k_0}^{\infty}(1-g_k)=0 \tag{5.9}$$

which is equivalent to

$$\sum_{k_0}^{\infty} g_k = \infty \tag{5.10}.$$

By putting the expression for T above into this equation, we can see that

$$\sum_{k_0}^{\infty} g_k \geq \sum_{k_0}^{\infty} \exp(-\ln k) = \sum_{k_0}^{\infty} \frac{1}{k} = \infty \tag{5.11},$$

which is a sufficient condition to insure that the algorithm converges to the global minimum statistically (Szu and Hartley, 1987).

In Boltzmann annealing it is clear that the procedure proceeds prohibitively slowly because the temperature is lowered such a small amount at each iteration. The challenge is to develop a cooling schedule that is efficient for the problem at hand yet still allows for the global optimum to be found. A variation known as “fast annealing” employs the cooling schedule $T(k)=T_0/k$. The generating function associated with it is

$$g(\underline{x}) = \frac{T}{\left(\Delta\underline{x}^2 + T^2\right)^{(D+1)/2}} \tag{5.12}$$

which is simply the Cauchy distribution. One advantage of the Cauchy distribution is that it has greater probability of testing points farther from the current configuration than the Boltzmann distribution. This is particularly useful in the phase of the optimization procedure that is attempting to map the search space and find the local minima. Making use of the above example, we note that for the generating function and cooling schedule associated with fast annealing, the convergence criterion is satisfied, i.e.,

$$\sum_{k_0}^{\infty} g_k \approx \frac{T_0}{\Delta \underline{x}^{D+1}} \sum_{k_0}^{\infty} \frac{1}{k} = \infty \qquad (5.13).$$

The cooling schedule associated with fast annealing, $T(k) = T_0/k$, is exponentially faster than that of Boltzmann annealing and has been tested on a wide range of problems (Szu and Hartley, 1987).

Although fast annealing is exponentially faster than Boltzmann annealing, it is still unduly slow for practical problems. A cooling schedule that is quicker, coupled with an appropriate generating function, is still sought. The method of very fast annealing employs the cooling schedule

$$T_i(k) = T_{0i} \exp\left(-c_i k^{1/D}\right) \qquad (5.14)$$

where T_i in this case refers to the temperature associated with each individual parameter (indexed by i), k is the annealing time index as before and c_i is a constant that can be used to scale the speed of the cooling schedule. To complete the specification of the algorithm, the generating function and acceptance criterion need to be made specific. The generating function associated with this cooling schedule is

$$g_T(T,y) = \prod_{i=1}^{D} \frac{1}{2\left(\left|y^i\right| + T_i\right)\ln\left(1+1/T_i\right)} \equiv \prod_{i=i}^{D} g_T^i\left(y^i\right) \qquad (5.15).$$

In order for this equation (eq. 5.15) to satisfy the same heuristic proof given previously it has to satisfy the same conditions. We must show that the combination of generating function, acceptance criterion (which is taken to be the same as before) and cooling schedule, will allow for any point in the search space to be reached, statistically, at any state of the optimization algorithm subsequent to the current. Summing the generating probability from the current state to infinity

$$\sum_{k_0}^{\infty} g_k \approx \sum_{k_0}^{\infty} \left[\prod_{i=1}^{D} \frac{1}{2\left|y^i\right| c_i} \right] \frac{1}{k} = \infty \tag{5.16}$$

where c_i is a scaling coefficient, we can see that the this condition has been met. The parameters in the problem are generated from the specification for y^i specified above through the equation

$$x_{k+1}^i = x_k^i + y^i \left(A_i - B_i \right) \tag{5.17}$$

where A_i and B_i are the upper and lower constraints on the parameter, respectively. Clearly, it is possible to generate a value that is outside the parameter constraints using this equation so it is simply iterated until the constraints are satisfied. This is for satisfaction of the linear constraints in the problem which are defined simply as constraints which have a given range and are not part of another constraint equation to be satisfied.

NON-LINEAR CONSTRAINTS

As noted above, the algorithm has a simple method of satisfying the linear constraints in the problem through iteration of the generation routine. In slider design,

however, it may be beneficial to define constraints that are overlapping. That is to say, the resulting design would have a region that is not simply connected. Although this does not present a problem in principle, the Reynolds' equation solver can not deal with this geometric configuration. In order to allow the flexibility of having overlapping constraints without creating configurations that are not simply connected, a non-linear constraint checking algorithm had to be implemented.

The outline of the algorithm is check the line segments connecting each adjacent point that define a slider rail with every other line segment defined between adjacent points to see if they intersect within the adjacent points being checked. The difficulty presented is when the points are used to create equations of lines and the resulting set of equations is solved, numerical inaccuracy can lead to incorrect results. The solution is to parameterize the line segments and then check the value of the parameters to see if an intersection occurs. Each line segment is parameterized by a single parameter and its endpoints, e.g.,

$$\begin{aligned} x(\alpha) &= \alpha x_1 + (1-\alpha)x_2 \\ y(\alpha) &= \alpha y_1 + (1-\alpha)y_2 \end{aligned} \tag{5.18}$$

and similarly for the second line segment. The equations for the two segments are set equal and the linear system specified in the algorithm description results.

Loop over all rails (index i)
  Loop over all rail points on rail i (index j)
    Loop over all rails (index k)
      Loop over all points on rail k (index l)
        Check to be sure lines are not parallel
          Find parameter values by solving the linear system:

$$\begin{bmatrix} x_{ij} - x_{i(j+1)} & x_{k(l+1)} - x_{kl} \\ y_{ij} - y_{i(j+1)} & y_{k(l+1)} - y_{kl} \end{bmatrix} \begin{Bmatrix} \alpha \\ \beta \end{Bmatrix} = \begin{Bmatrix} x_{k(l+1)} - x_{i(j+1)} \\ y_{k(l+1)} - y_{i(j+1)} \end{Bmatrix}$$

          for α and β then check to see if both are
          between 0 and 1, i.e.,
          if $0 < \alpha < 1$ and $0 < \beta < 1$ then an intersection exists
            stop looping

Figure 5.1: Outline of rail intersection algorithm

This method is coupled with the linear constraint checking to check if the configuration generated is feasible, i.e., it satisfies *all* constraints. Further, it does not suffer from the inaccuracy that plagues the method of generating equations of lines for the segments in question.

EXAMPLE

The following example demonstrates some of the principles introduced above. The optimization of the following slider employed various constraint combinations and objective functions in the search for the optimal design. It is important to note that the objective function defined is the quantity that is optimized. It may turn out that the minimum value of the objective function does not possess characteristics deemed necessary to the design being sought. In order to remedy this problem, the objective

function weightings need to be adjusted so that the final optimized design will possess the necessary physical characteristics. Proper weighting of the objective function is a difficulty inherent to multi-goal optimization. Clearly, a simple rule of thumb would be to give the most important terms the heaviest weight. It is possible, however, that if this rule is followed without care, the problem becomes one of single goal optimization, i.e., the remaining goals are neglected. Scaling schemes have been proposed to avoid this difficulty (Yoon and Choi, 1995, 1996) but they suffer from selection of proper scaling parameters--a problem tantamount to that of proper weighting.

The air bearing slider that was used as the subject of this optimization was originally designed with the intent of minimizing contamination (Zhang, 1996). The central idea is to force particles that enter the region under the slider to leave from the sides. By doing this, no large contamination particles will be created at the rear pad of the slider which could potentially cause a problem by breaking off and then re-entering the slider air bearing region. In the original design, however, account was not taken of the sliders fly height profile over the radius of the disk. This is a relatively simple goal to achieve through optimization but it is complicated by the slider's small sub-ambient pressure cavity. The objective function that was used for this optimization was

$$f(\underline{x}) = 50 \times \sum \left(fh_i - 35nm \right) \tag{5.19}$$

where the fh_i are evaluated at the following positions

radius (mm)	skew (deg.)
15	-7.59
23	0.0
31	7.59

Table 5.1: Table of radius and skew values used in evaluation of objective function

The constraints used in the problem are depicted graphically below (Figure 5.2).

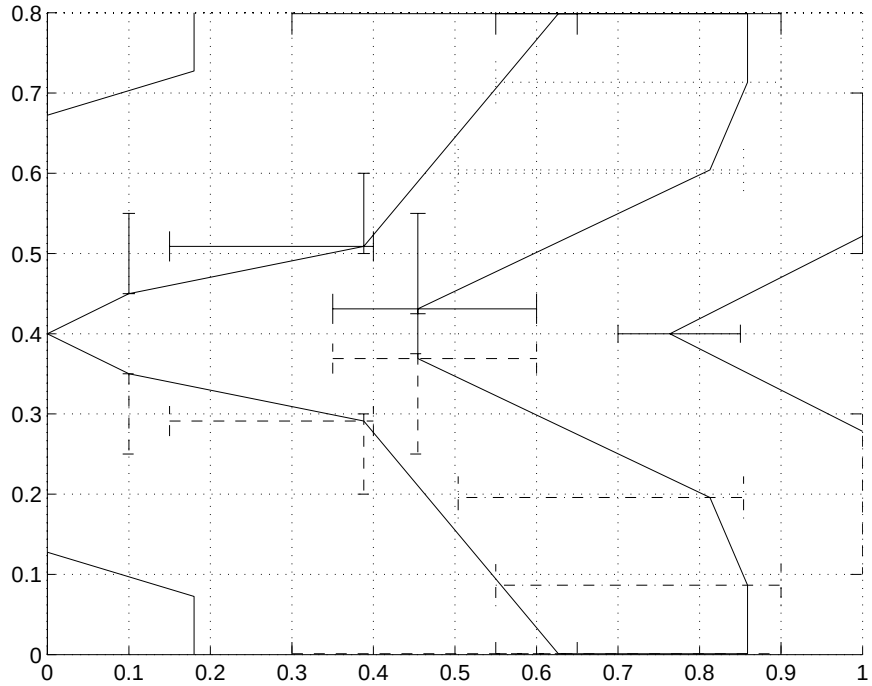

Figure 5.2: Constraints employed in contamination optimization

The constraints are specified by lines centered on specific vertices. Theses lines represent the entire range, in one dimension, that a vertex may move in the course of the optimization. If a vertex has two lines centered on it then it may move in two dimensions. The solid lines represent constraints that are used in the problem directly, i.e., they are calculated as described above and their values represent the basis for a new configuration. The lines represented by large dashes are constraints that move vertices in symmetry with other, specified vertices. The intent in this design is to maintain the symmetric nature of the air bearing throughout the optimization. The lines represented by the short dashes are constraints on vertices that are to move relative to other, specified vertices. The purpose of employing these ‘relative constraints’ is to maintain a specific feature of the air bearing, in this case the geometry of the back section of the main air bearing pad. The final type of constraint depicted in the figure--represented by the

dashed-dotted line--is the ‘relative symmetric constraint’ whose purpose folds the previous two together, i.e., its purpose is to maintain a specific feature symmetric to the original, maintained feature. In addition, the recess depth of the main cavity was allowed to vary from 2 to 15 microns and the position of the loading force acting on the slider was allowed to vary from 0.45 to 0.55--in normalized units--along the direction of flow.

The following figure shows how the optimization procedure progresses. The plot is of the value of the objective function as a function of configuration number.

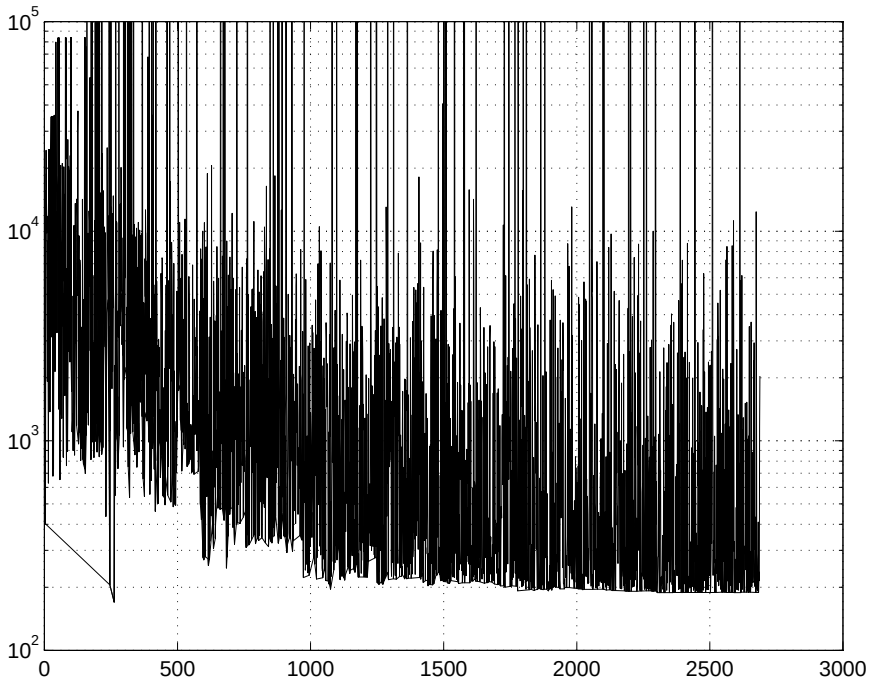

Figure 5.3: Progression of objective function for contamination slider optimization

From the above figure, it is clear that a number of configurations crashed (as defined earlier) and have had their objective function value arbitrarily set to the maximum value. Even near the conclusion of the optimization when the temperature is relatively small, implying that the configurations are not changing considerably from one iteration to the next, the number of configurations that crash is high. Further, for the configurations that do not crash the change in objective function value is large from one iteration to the next

which implies that there is a large sensitivity to one or more of the parameters or some combination of them. It should also be noted that the optimal configuration is found early in the optimization procedure then approached asymptotically as the optimization draws to conclusion. This is empirical evidence of the search ability of the optimization procedure. While mapping the search space in the early stages of the optimization the algorithm finds what turns out to be what is taken as the optimal configuration. Later, as the algorithm concentrates on the global optimum, the objective function value asymptotically approaches the objective function value associated with the previously found optimum configuration. It is possible that the optimal value being approached is not unique and, in fact, uniqueness is not a necessity of the procedure, i.e., for a defined objective function the goal is simply to find a single global optimum regardless of the physical configuration.

The cooling schedule employed is shown in the following plot

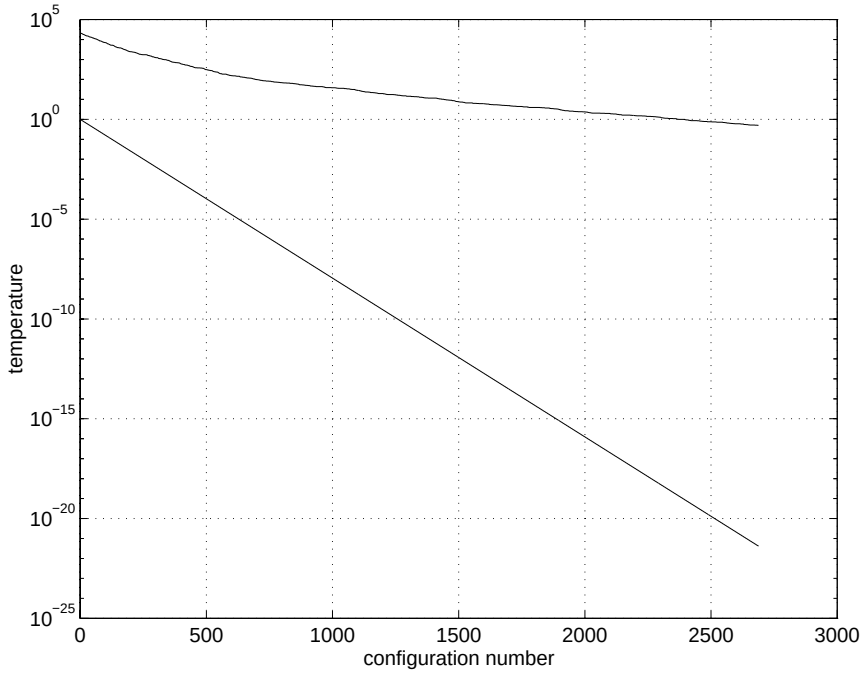

Figure 5.4: Progression of objective function and parameter temperature

Note that there are two lines in the above figure. The lower line represents the temperature associated with the parameters. The upper line represents the temperature associated with the cost, both as described above. The cost temperature starts at a much higher value and then does not decrease uniformly as the parameter temperature does. The reasons for these phenomena are that the cost temperature is scaled initially to the mean value of the objective function and the non-uniform decrease is due to reductions in the cost temperature only coming upon acceptance of new configurations versus reduction in the parameter temperature occurring upon every new instance of a configuration. The mean value of the objective function which is used as the initial cost temperature is determined by sampling random configurations in the search space then taking their arithmetic average. The purpose of setting the initial cost temperature equal to the mean value of the objective function is to allow acceptance of almost any new configuration at the outset of the optimization.

The functional form of the cooling schedule is

$$T = \frac{1}{\exp\left(\alpha \exp\left(-\beta\right) N\right)} \tag{5.20}$$

where, for the cost temperature, α and β are 1.0 and 2.0, respectively and N is the number of configurations accepted so far and, for the parameter temperature, α and β are 1.0 and 4.0, respectively and N is the number of configurations generated so far. Note that this cooling schedule is independent of the number of parameters. This particular combination of parameters was chosen to be generous due to the nature of the problem. The larger the value of β, the more slowly the temperature decreases. For the problem of slider air bearing optimization, employing the cooling schedule given above, it has been

determined that values of 1.0 and 3.0 for α and β respectively, are optimal. The value of 4.0 used for beta here was determined not to significantly improve the algorithm's ability to converge to the optimal solution. It is through search space expansion, i.e., modification of the constraints, that the quality of the final solution is improved. This satisfies one necessary condition of an algorithm that aspires to find global optima, namely, that the quality of a solution for a search space that encompasses a previous search space must be as good or better, i.e.:

if $\min_{\underline{x} \in S} f(\underline{x}) = f^*$ and $\min_{\underline{x} \in S'} f(\underline{x}) = f'^*$ where S $\subseteq$ S' then $f'^* \leq f^*$ must be true.

Therefore, if an algorithm reaches the global minimum of a function for a given search space and the search space is subsequently enlarged so as to encompass the original search space, then the final solution on the larger search space should be as good or better than the solution on the smaller search space. This has been shown to be the case in this problem. Subsequent chapters will also demonstrate this property of the simulated annealing algorithm for more complicated examples.

The final design is depicted below (Figure 5.5).

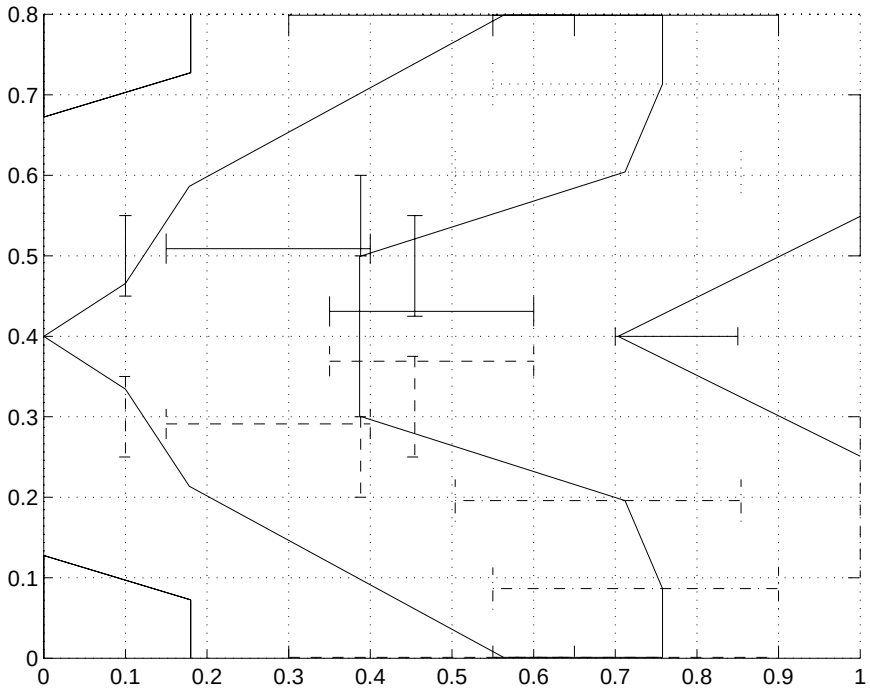

Figure 5.5: Optimized contamination slider

The most distinctive difference between the original and optimal designs is the wide expansion of the subambient pressure cavity. Development of subambient pressure is critical to the bearing's ability to maintain a constant fly height across the radius of the disk. The algorithm develops this feature without guidance towards or away from this goal save the objective function that has been defined. This is indicative of the algorithms ability to effectively search the configuration space provided by the constraint definitions. The change in subambient pressure region geometry was coupled with a change in recess depth from the original value of 3.0 microns to 4.4 microns.

The optimal value of the objective function translates into physical characteristics of the bearing. The only term included in this optimization is the fly height of the bearing over the disk--defined to be the physical space between the transducer point on the slider (0.001 inch from the back of the slider centered transversely) and the disk. The initial and resulting fly height profiles are depicted in the graph below (Figure 5.6).

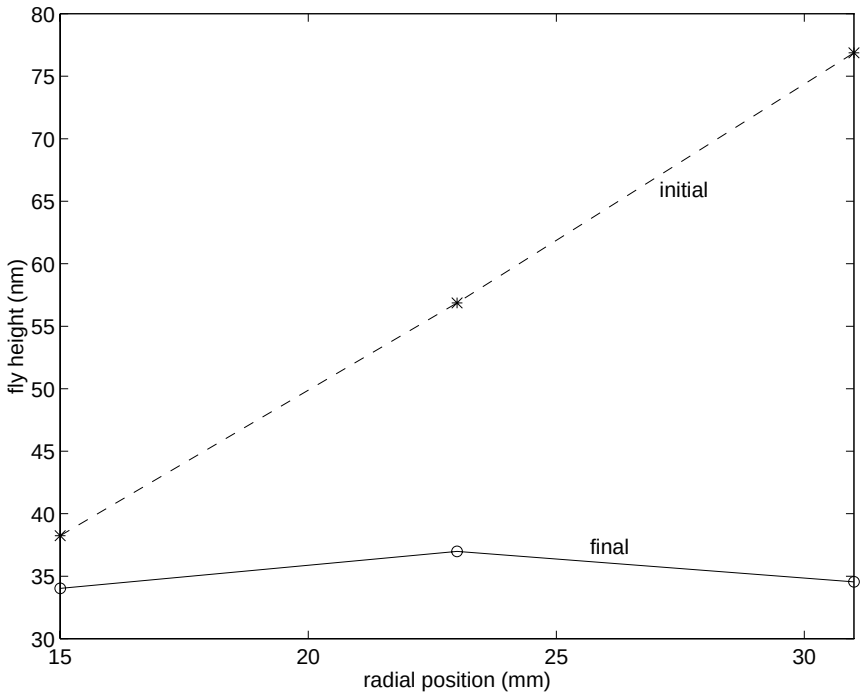

Figure 5.6: Comparison of fly height profiles for original and optimized design

It is clear from this graph that the optimization procedure achieved the goal initially defined starting from a configuration that considerably less than optimal. This points to the utility of the algorithm to transform designs that are created for a specific purpose into useful designs whilst maintaining the original characteristics, in this case contamination control.

CONCLUSIONS

In this section, a fuller description of the simulated annealing algorithm has been presented. In particular, the implementation specific to this study has been illustrated fully. The theoretical basis for the algorithm has been further explicated and its utility made manifest through an example.

CHAPTER 6

ALTITUDE INSENSITIVITY

INTRODUCTION

In magnetic hard disk drives the minimum spacing between the air-bearing slider and disk has been reduced to under 50nm, and some drives now employ so-called proximity sliders that are designed to operate at some level of interference between the slider and the peak asperities on the disk. This ultra-low flying condition brings into play some new interface phenomena and accentuates some of the well known phenomena as well.

The performance of proximity sliders with regard to their sensitivity to altitude changes is investigated. We include altitude sensitivity as an objective in the design optimization scheme and demonstrate that it can yield air-bearing designs much less sensitive to changes in altitude.

Research for future drives targets fly heights that are in the range of 15 to 20nm, and even less. Because the fly height of current and future sliders is so small, the loss of fly height due to reduction in ambient pressure becomes a significant effect. This

situation typically occurs in controlled pressure environments such as airplanes or at high altitude.

It is of interest to find air-bearing surface designs that can minimize the change in fly height that a hard disk drive air bearing experiences when it is subjected to changes in ambient pressure. The basic idea is to balance the change in positive and sub-ambient pressure that the slider develops. The method for achieving this is through a systematic process of optimization. The optimization technique employed is the simulated annealing algorithm. The constraints on the slider are geometric, which constrain the changes in bearing geometry that the slider undergoes, as well as parametric, which constrain the parameter changes that affect the slider's flying attitude.

PROBLEM DESCRIPTION

Several parameters can be varied to optimize specific characteristics of a slider air bearing. For example, there is a strong correlation between the applied load and fly height. However, due to other performance constraints, e.g., tribological, applied load may not be the best parameter for manipulating the fly height. Other parameters include the position of the loading force, the length and height of the taper and the recess depth. In addition, the geometry of the air bearing surface can be modified to optimize the slider bearing's performance. Each vertex (corner) of a slider rail that is varied represents a parameter in the optimization (two if varied in two directions). Due to the relatively high number of parameters that exist in such an optimization problem, the nonlinear nature of the lubrication equation, and the fact that the objective function is evaluated by employing a numerical solver, the simulated annealing method is employed as the

optimization algorithm, because this technique avoids the expense of calculating derivatives that exists in gradient-based optimization algorithms and handles the high number of parameters efficiently. Additionally, it can achieve globally optimal solutions for multi-term objective functions.

MATHEMATICAL FORMULATION

OBJECTIVE FUNCTION

The cost function defined for this problem, where altitude sensitivity is a primary concern, is:

$$f(\underline{x}) = 50 \times \Delta fh(nm) + 50\left(\overline{fh} - 14\right)(nm) + \sum P(pitch - 200)(\mu rad) + \sum\left(\Delta fh_{alt}\right) \tag{6.1}$$

The first term in the objective function represents the maximum difference in fly height over the radius of the disk multiplied by a weighting of 50. The next term represents the difference in average fly height over the radius of the disk minus the target fly height of 14nm multiplied by a weighting of 50. If the sum of these two terms is minimized to zero, the resulting design will fly at all radii of the disk at the targeted fly height of 14nm. The next term sums the pitch at each of the evaluation points and has the form of a penalty function, i.e., it does not contribute to the objective function until the pitch at an evaluation point is greater than 200μrad. The weighting of this term is 1. The final term represents the sum of the maximum changes in fly height due to changes in ambient pressure, at each of the radial evaluation points specified for the objective function. The weighting for this term is also 1.

When positive pressure sliders experience a decrease in ambient pressure, the fly height typically also decreases. However, because sub-ambient pressure bearings balance positive and sub-ambient pressure to achieve their bearing capacity, this is not necessarily the case. In consideration of this, the final term in the objective function calculates the maximum difference in fly height at a single evaluation point on the disk due to changes in ambient pressure and then sums over all evaluation points. For the purposes of this investigation, two ambient pressure conditions are used, one at standard temperature and pressure and one corresponding to the ambient pressure and temperature at 2,000 meters above sea level. The change in environmental conditions is realized in the equation as changes in the mean free path of the air and the viscosity.

CONSTRAINTS

In the optimization example presented here, 14 parameters were allowed to vary. Of these 14, 13 were geometric constraints as depicted in Figure 6.1. Each box represents two constraints because a vertex is allowed to move in two directions. The lines represent one constraint since the vertex in question is allowed to move in only one direction.

The final variable included in the optimization was the recess depth. It was allowed to vary between 1 and 5 microns. There is also the implicit constraint of symmetry which does not add degrees of freedom to the problem. The constraint of symmetry is not necessary for a slider design. Symmetry may be a desirable characteristic of a slider for other drive-related design issues. The symmetry constraint can be implemented on a vertex by vertex basis, i.e., it may be desirable to retain symmetry for some feature of the bearing design but not all.

RESULTS

Figure 6.1 shows the original design as well as the design created by employing the simulated annealing algorithm on the above stated objective function, subject to the defined constraints.

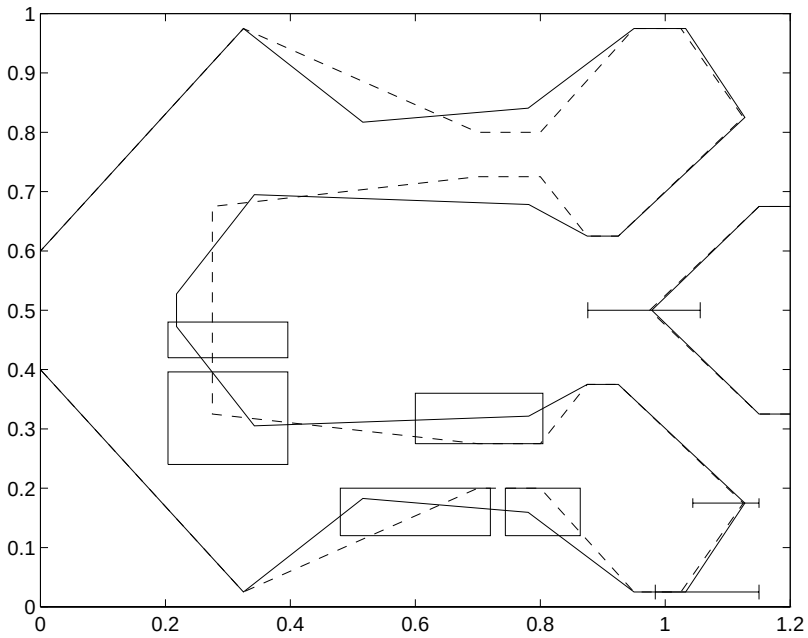

Figure 6.1: Original (dashed) and optimal (solid) designs with constraints (all dimensions in mm)

The most significant change was the reduction of the frontal area of the slider. The rear of the slider underwent almost no variation from the original design although parameters were available to allow changes. The recess depth changed from 3 microns in the original design to 2.35 microns in the optimized design. The quantitative improvements in the objective function are summarized in Table 6.1.

Item	Initial value	Final value	Percent improvement
Total cost	1.16e+02	9.63e+00	91.7
Δfh	9.20e+01	3.00e+00	96.7
Variation from target	1.03e+01	3.33e-01	96.8
Altitude Sensitivity	1.36e+01	6.300e+00	53.8

Table 6.1: Objective function term values including weighting

It is seen that the overall objective function experienced considerable improvement. Due to the relatively high weighting given to the Δfh and variation from target terms, the optimized design had almost no variation from the design fly height value of 14nm at any evaluation point on the disk when flying in standard temperature and pressure conditions. The pitch term was not included in the table because its value never changed from zero, i.e., the pitch at any of the evaluation points never exceeded 200μrad.

Most importantly, the altitude sensitivity term improved considerably. Figure 6.2 and Figure 6.3 show the change in fly height profile from standard temperature and pressure to 2000m for both the original (Figure 6.2) and optimized (Figure 6.3) designs.

These figures demonstrate that the optimized design suffers a much smaller decrease in fly height due to the increase in altitude. In fact, the improvement exceeds what is implied by the improvement in the objective function. The altitude sensitivity term is constructed by summing the maximum difference in fly height at each of the evaluation points for different environmental conditions. Because the original design did not meet the design fly height of 14nm exactly, the altitude sensitivity term did not completely reflect the decrease in fly height due to the increase in altitude.

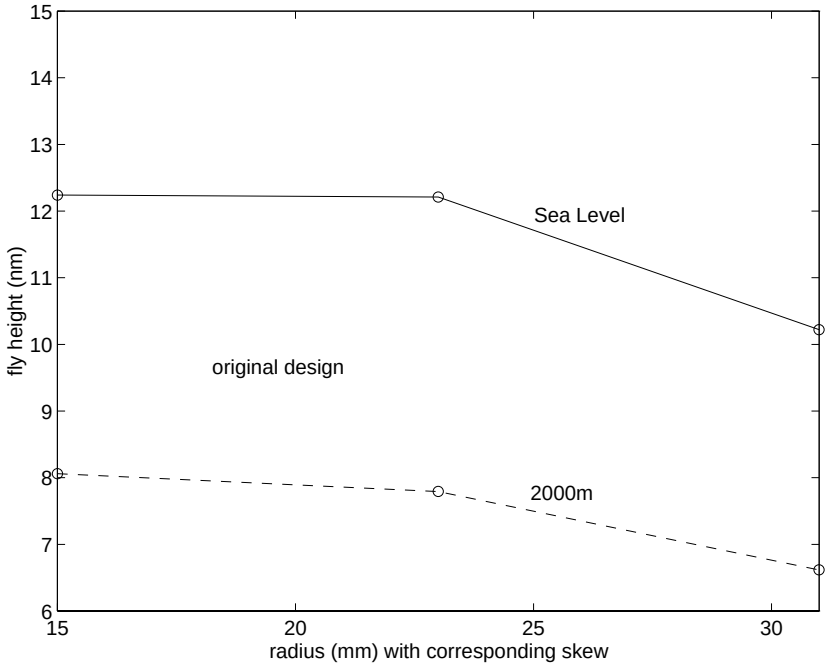

Figure 6.2: Fly height profile of original design at ambient and 2000m equivalent pressure

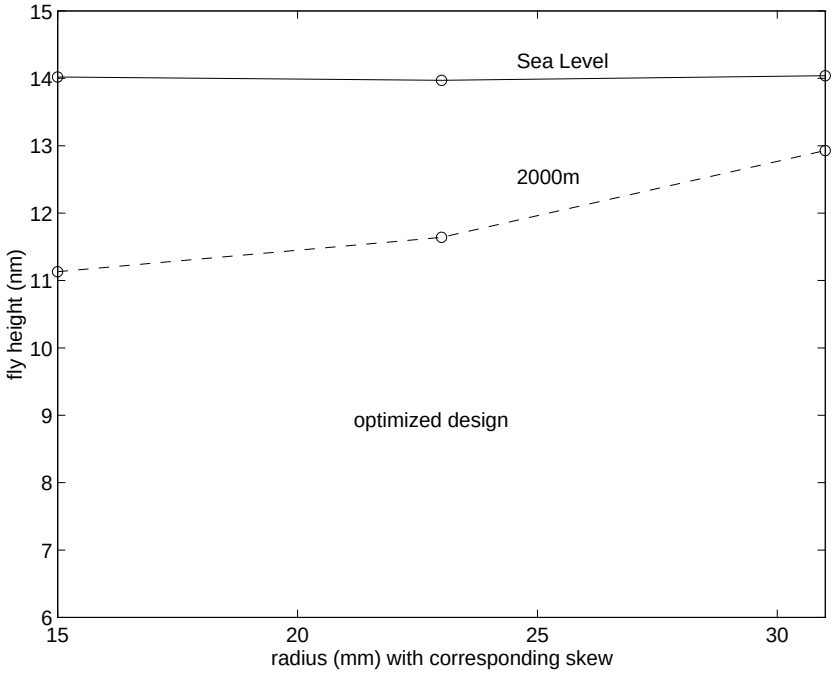

Figure 6.3: Fly height profile of optimized design at ambient and 2000m equivalent pressure

Note that the side rails of the slider expand for a greater bearing area. The net result is a design that does not experience any significant change in pitch (Table 6.2) and has much less of a reduction in fly height due to an altitude change than the original design.

	inner radius	middle radius	outer radius
sea level	95	138	180
2000m above sea level	94	141	186

Table 6.2: Pitch of optimized slider in μrad at ambient and 2000m equivalent pressure

To further improve the quality of the optimal design, the objective function coefficients were changed so that the altitude insensitivity goal was weighted more heavily. The constraints in the problem were widened as well to provide a larger search space over which to seek out the global optimum. The objective function employed in this search is

$$f(\underline{x}) = 20 \times \sum (fh - 14)(nm) + \sum P(pitch - 200)(\mu rad) + 25 \times \sum (fh_{alt} - 14) \tag{6.2}$$

which represents an increase in the weight associated with the altitude sensitivity term. Additionally, the first two terms in the objective function used previously have been collapsed into a single term which sums the distance at each evaluation point from the target fly height of 14nm. The same technique is employed in the altitude sensitivity term. As described above, in the previous case the maximum difference over all altitudes (2 in this case) at each evaluation point was summed. The advantage of measuring the difference in fly height from the target value is that these two terms (the first and last)

will not be directly conflicting, i.e., using the previous scheme, the altitude sensitivity term did not necessarily need to approach the target fly height value to be reduced. Using the previous scheme, if the fly height at standard temperature and pressure deviated from the target value, the altitude sensitivity term could be minimized without ever approaching the target fly height. Employing the new scheme, these goals do not compete directly. They compete implicitly through the nature of the physics that govern the problem.

The constraints that were employed in this case are depicted in the following graph (Figure 6.4)

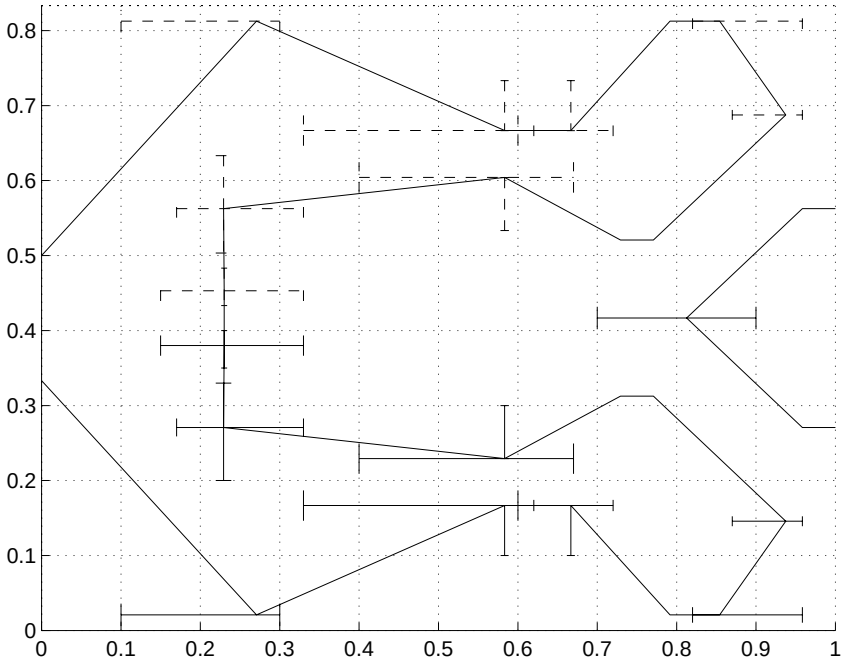

Figure 6.4: Constraints used in second altitude sensitivity optimization

The most significant difference in the constraint configuration between the first and second optimization is that the sub-ambient pressure region and the trailing edge of the bearing are given more freedom.

The resulting optimal configuration is depicted in the following graph (Figure 6.5)

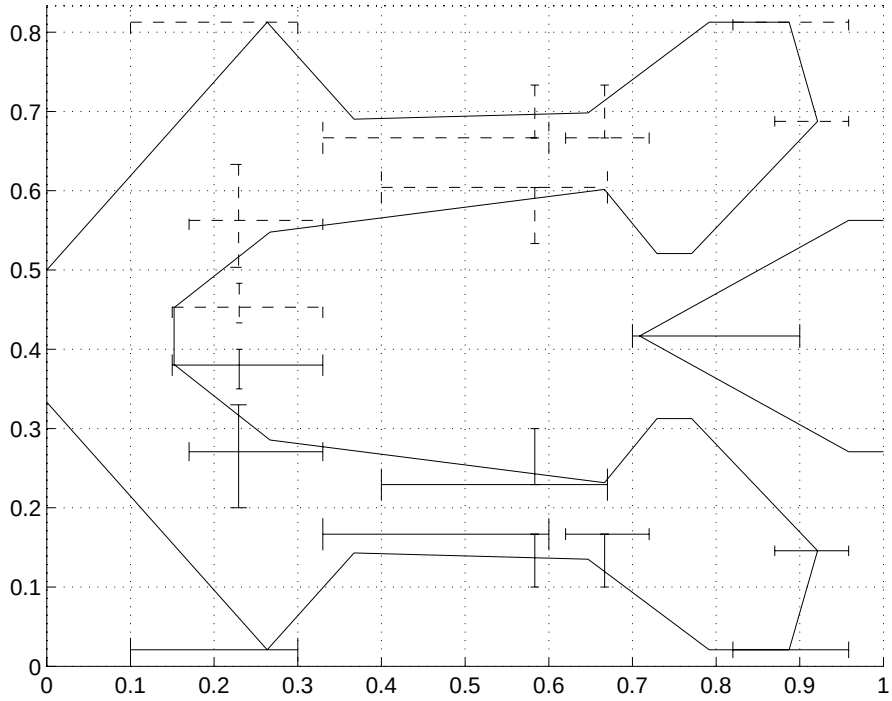

Figure 6.5: Optimal configuration of slider air bearing for second iteration of altitude sensitivity optimization including constraints (recess depth = 2.23 microns)

The most notable difference between the original and optimal configurations is the reshaping of the sub-ambient pressure cavity. Sub-ambient pressure is developed under a slider as a result of the expansion of inlet air through Couette flow due to a steep increase of film thickness (Peng and Hardie, 1995b). By giving the constraints that dictate the geometry of the sub-ambient region greater range, the probability of creating a configuration that balances the two terms appropriately is enhanced. It creates a design that is capable of maintaining the same pitch and fly height independent of ambient pressure up to the equivalent ambient pressure of 2000m. Consistent with this, the trailing part of the air bearing does not change considerably, enlarging only slightly to generate slightly more pressure at the rear of the slider.

The resulting fly height profile of the slider is displayed in the following graph (Figure 6.6)

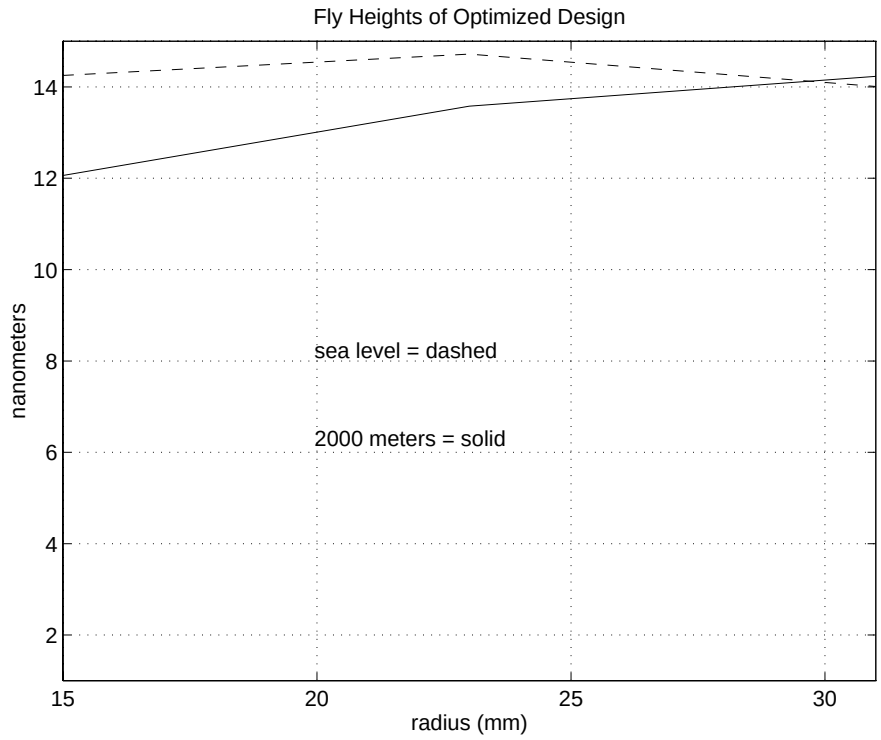

Figure 6.6: Fly height profile of second iteration of optimized design at ambient and 2000m equivalent pressure

From the above graph (Figure 6.6), it is clear that the second iteration of the optimized design came closer to achieving the goal of ambient pressure insensitivity. This is an example of the simulated annealing algorithm satisfying the necessary condition of improving (or at least achieving an equal value of) the optimal value of the objective function for a search space that encompasses a previous search space. It should be noted, however, that the comparison is not direct because the weighting of the objective function was different.

This result also demonstrates that the simulated annealing algorithm is capable of optimizing multiple goals simultaneously. In this specific example, the algorithm found a configuration that maintained the target fly height at standard temperature and pressure

while minimizing the amount of fly height loss due to a change in ambient pressure. In fact, the above figure (Figure 6.6) demonstrates that it is possible for the design to actually gain fly height with a loss in ambient pressure by appropriately balancing the development of positive and sub-ambient pressure developed under the bearing. The following graphs (Figure 6.7, Figure 6.8) depict the pressure profiles under the slider for both ambient pressure values while the slider is at the outer radius of the disk (31mm).

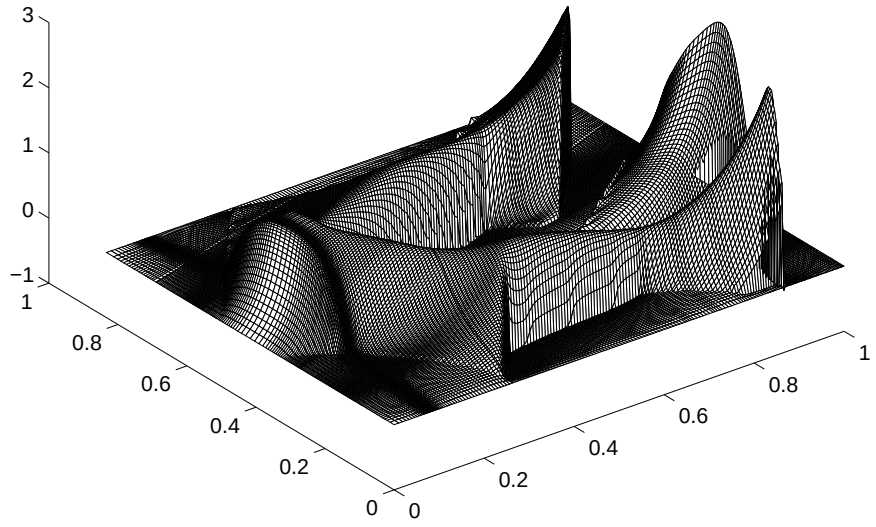

Figure 6.7: Pressure profile for second iteration of altitude insensitive design at ambient pressure equivalent to sea level; calculated at outer radius of disk (31mm, negative 17.39 deg. skew)

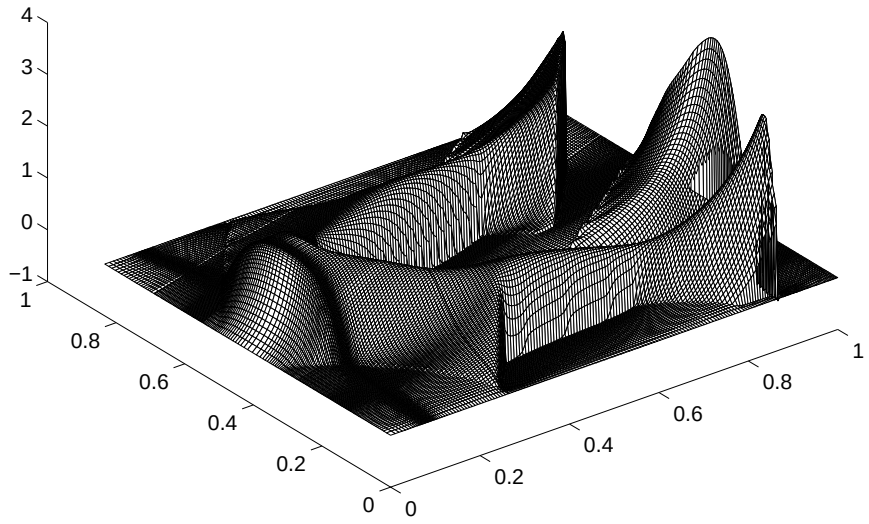

Figure 6.8: Pressure profile for second iteration of altitude insensitive design at ambient pressure equivalent to 2000m; calculated at outer radius of disk (31mm, negative 17.39 deg. skew)

The fly heights, positive pressure and sub-ambient pressure generated by these operating conditions are summarized in the following table (Table 6.3).

operating condition	mean free path of air (nm)	ambient pressure (pa)	fly height (nm)	positive pressure (gf)	sub-ambient pressure (gf)
sea level	63.5	1.01×10^5	14.0	3.77	-1.97
2000m	76.4	0.795×10^5	14.2	3.37	-1.57

Table 6.3: Table of fly heights and pressure values for second iteration of altitude insensitive design

Although the difference is not readily seen in the pressure profiles (Figure 6.7 and Figure 6.8) it is evident from the above table (Table 6.3) that positive and sub-ambient pressure are both lost as the slider undergoes a change in operating conditions from the equivalent of sea level to 2000m ambient pressure. They are, however, balanced in such a way that the fly height can be maintained constant. This demonstrates the feasibility of creating a design that is insensitive to changes in ambient pressure.

CONCLUSIONS

That a slider can fly at a targeted fly height at sea level and not experience significant loss in fly height due to a change in altitude is demonstrated. Although the ambient pressure and mean free path of air change considerably between sea level and 2000m, the design was able to retain most of its fly height by balancing the positive and sub-ambient pressure of the slider. In fact, at the outer radius of the disk, the design actually increased slightly in fly height.

CHAPTER 7

SUB-AMBIENT PRESSURE OPTIMIZATION

INTRODUCTION

Optimization of the sub-ambient pressure that a slider develops is an attempt to address a very practical consideration in the operation of hard disk drives, namely the pickup of lubricant in the sub-ambient pressure cavity in designs that take advantage of this feature. Lubricant pickup in the subambient pressure cavity of air bearing sliders is a genuine concern. Because the flying characteristics of the slider change as the bearing's geometry changes, it is important to consider this effect.

If the sub-ambient pressure is reduced for a given slider with the external force remaining fixed, it is clear that less positive pressure is necessary in order to maintain the force balance. This will likely force a change in the slider's attitude. The point of concern, however, is at the magnetic read-write element of the slider. Due to this, sliders with different attitudes may achieve equal fly heights. This is, in fact, the intent of the current optimization.

The sub-ambient pressure is calculated by taking the pressure output from the CML air bearing design code and integrating it numerically in two dimensions. The integration scheme implemented takes the weighted average of the neighbors for a given point and then normalizes by the area of the control volume under consideration. Through cancellation it turns out, however, that the contribution is simply the nodal value multiplied by the control volume area (Figure 7.1). This is relevant due to the fact that an unevenly spaced grid is employed.

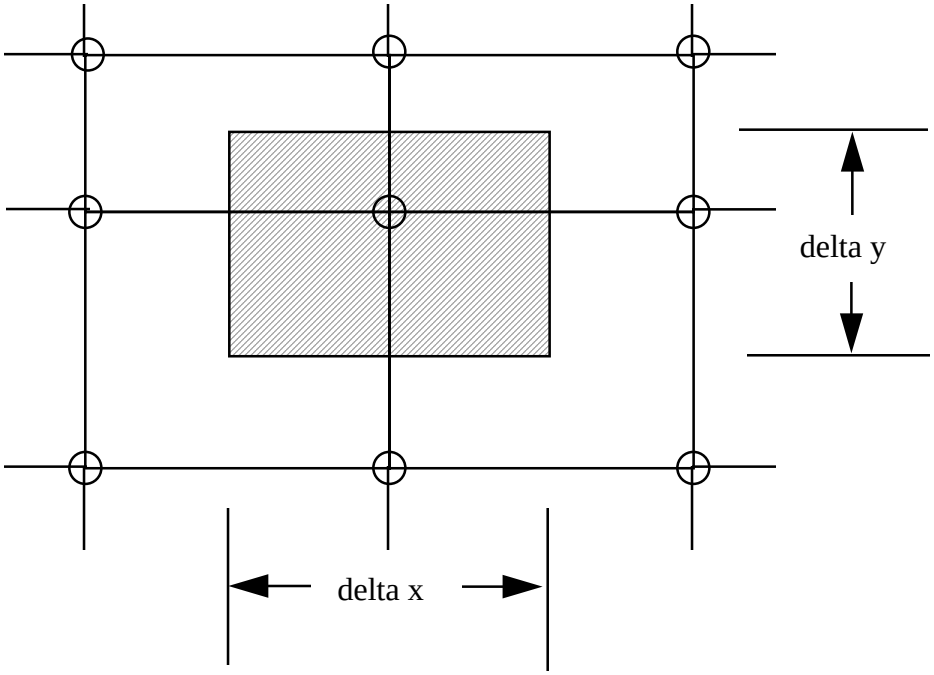

Figure 7.1: Example area used for pressure integration

MATHEMATICAL FORMULATION

OBJECTIVE FUNCTION

The objective function used for the sub-ambient pressure optimization is given below (Eq. 7.1).

$$f(\underline{x}) = \Delta fh(nm) + \left(\overline{fh} - 14\right)(nm) + \sum |roll|(\mu rad) + \sum \left(P_{sub-ambient} - P_t\right) \quad \text{(Eq. 7.1)}$$

The first term in the above objective function represents the maximum change in fly height over the radius of the disk. The second term represents the difference between the average fly height over the radius of the disk and the target fly height of 14nm. The third term represents the sum of the roll values over the evaluation points. The final term represents the sum of the difference between the sub-ambient pressure and the target pressure value over all the evaluation points. If the final term is minimized to zero, the sub-ambient pressure of the slider will be at the target value at every evaluation point. In this optimization, it is important to optimize the value of the roll. If it is not kept in check, it is possible that the final configuration would have a roll value that is unacceptably large because increasing the roll is a simple path to achieving the specified fly height on a design where the fly height is calculated at the side rail.

The objective function was evaluated at the following points (Table 7.1):

radius (mm)	skew (deg.)
15.0	6.27
17.2	0.0
31.0	-17.0

Table 7.1: Table of radius and skew values used in sub-ambient pressure optimization

CONSTRAINTS

There were nine constraints used in this optimization. They are depicted in the following graph (Figure 7.2). The constraint of symmetry is also imposed on the design which is reflected in the final design.

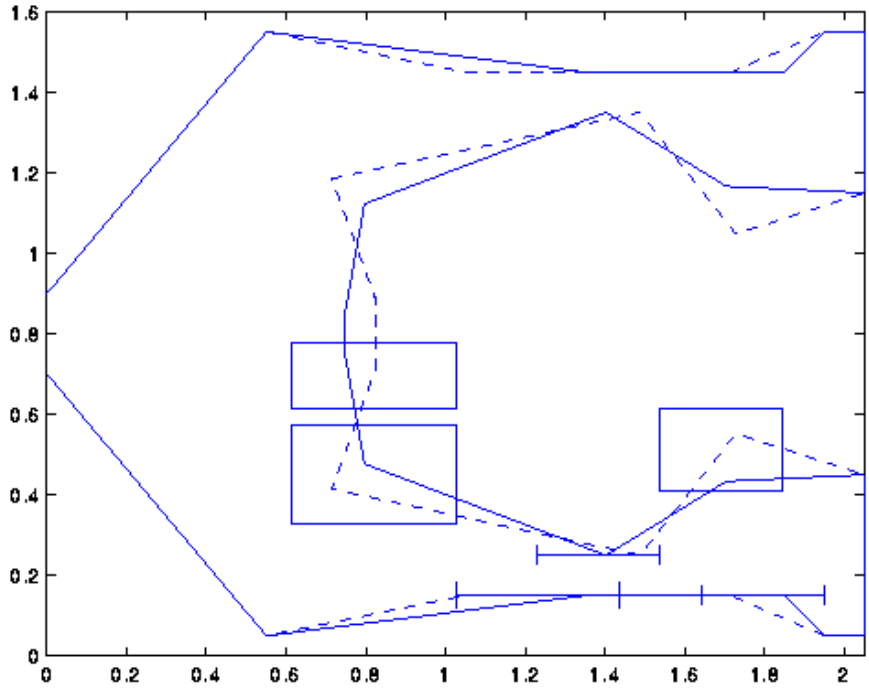

Figure 7.2: Original (dashed) and optimized (solid) designs including constraints for sub-ambient pressure optimization

The load acting on the slider was 3.5gf, located at the geometric center of the back of the slider. Neither the load nor the recess depth was varied in the attempt to minimize the sub-ambient pressure the slider developed. Typically, the sub-ambient pressure has a strong dependence on the recess depth of the slider. The intent in this case, however, is to minimize the sub-ambient pressure only through manipulation of the bearing geometry.

NUMERICAL RESULTS

The initial and final values of the pressure, both positive and sub-ambient, are tabulated as follows (Table 7.2):

radius (mm)	skew (deg.)	Original design pressure (g)		Optimized design pressure (g)		
		positive	sub-ambient	positive	sub-ambient	percent improvement
15.0	6.27	8.49	-4.99	8.26	-4.76	4.61
17.2	0.0	8.94	-5.44	8.71	-5.21	4.23
31.0	-17.0	10.24	-6.74	9.89	-6.39	5.19

Table 7.2: Table of pressure values for original and sub-ambient pressure optimized slider designs

The above table (Table 7.2) indicates that the improvements in the sub-ambient pressure term were slight. It should be noted, however, that the recess depth was not varied in this optimization. The purpose of the study was to determine if sub-ambient pressure could be effectively manipulated strictly through control of geometry.

The pressure profiles for the original and optimized designs, calculated at the outer radius are shown below (Figure 7.3 and Figure 7.4).

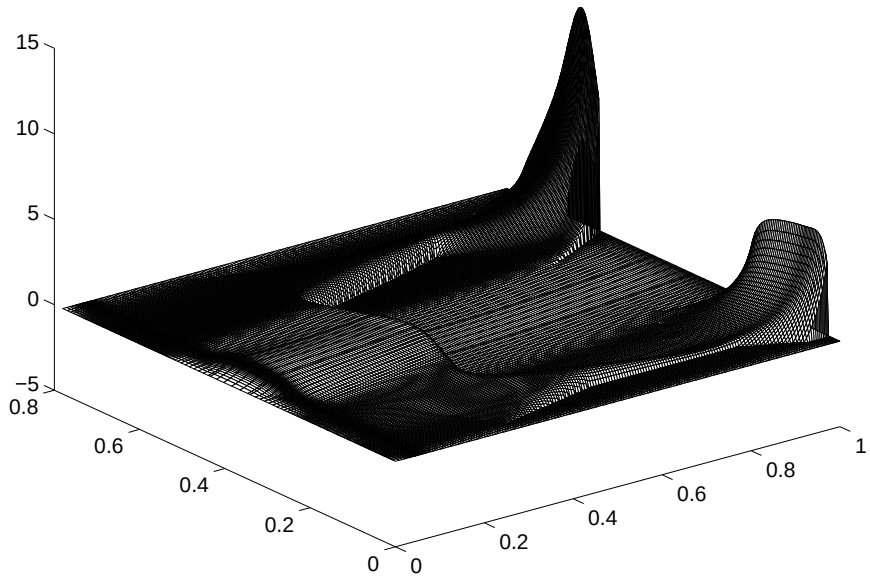

Figure 7.3: Pressure profile of original design at outer radius

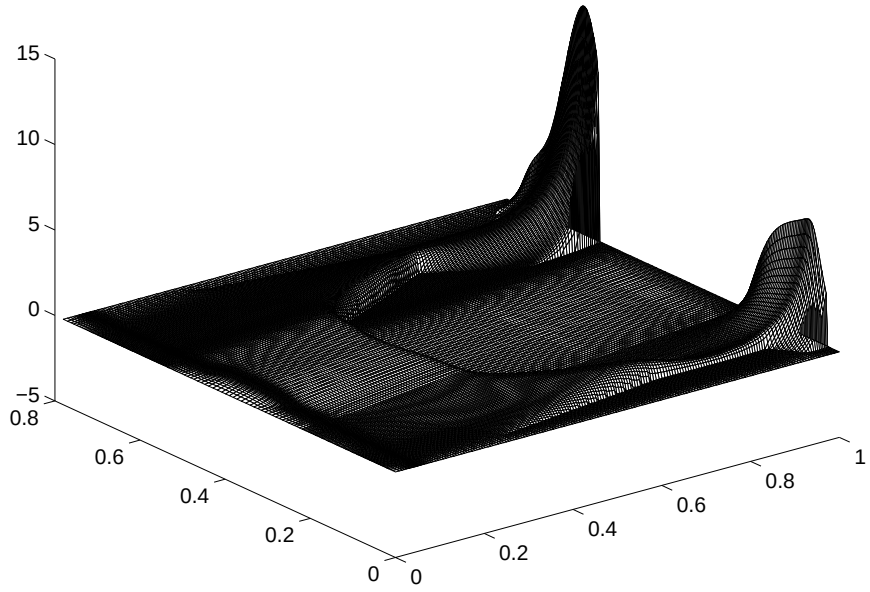

Figure 7.4: Pressure profile at outer radius of sub-ambient pressure optimized design

The above figures (Figure 7.3 and Figure 7.4) show that changes in the pressure profile are, in fact, very subtle. A more dramatic change in the sub-ambient pressure value would require much wider ranges for the constraints placed on the problem and inclusion of the recess depth as a parameter.

CONCLUSIONS

This chapter discusses the use of optimization for the purpose of reducing the sub-ambient pressure that an air bearing develops without changing its flying characteristics. It is shown that the even through control of slider geometry alone, it is possible to achieve a measurable deduction in the amount of sub-ambient pressure the slider develops. Coupling the ability for improvement based on geometric modification with the improvements afforded by recess depth variation, the sub-ambient pressure that a slider develops can be manipulated without changing the flying characteristics.

The ultimate goal of the optimization is to limit the amount of lubricant that the slider will pick up while operating. This facet of the optimization was not verified experimentally. It has been demonstrated, however, that sub-ambient pressure optimization is possible which implies that if a better understanding of the lubricant pickup process were developed, that model could be built into the optimization procedure. For example, if the largest determining factor regarding lubricant pickup was the maximum sub-ambient pressure under the bearing or the integrated sub-ambient pressure in the region directly behind the positive pressure rail then either of these quantities could be minimized in pursuit of a design that had lower lubricant pickup.

CHAPTER 8

OPTIMIZATION OF AIR BEARING DYNAMIC PERFORMANCE

INTRODUCTION

Due to the demand for increased storage density for computer hard disk drives, the minimum fly height for read/write heads and the sliders that house them has necessarily decreased significantly. The fly height for sliders in current commercial drives is in the range of 40 to 50nm and below. Research for future drives targets fly heights that are in the range of 15 to 20nm and below. Since this fly height is on the order of the roughness of many disks, it is reasonable to expect that the slider will periodically contact the disk. Even when direct physical contact does not occur, the roughness of the disk can contribute to modulation of the fly height of the slider. So, in addition to achieving the goal of slider fly heights at the 15 to 20nm range, dynamic stability is also sought. One approach for achieving lower dynamic fly height modulation is to increase the vertical stiffness of the bearing, that is, to increase the applied load necessary to effect a unit change in the fly height. Because the pitch and vertical modes

of the slider are coupled, the pitch stiffness should also be maximized. The stiffness can be cast as a sensitivity by simply taking its inverse which can be subsequently minimized. Adding the goal of sensitivity minimization to the more basic goals of low fly height and fly height uniformity across the disk, among others, typically creates a slider design that has a lower fly height modulation. This work shows that this is not always true, however.

The stiffness, and consequently the sensitivity, that is calculated from the static bearing analysis code does not include the time-varying nature of the Reynolds' equation. Because of this, the stiffness that is predicted is typically low and does not take into account the non-linear stiffening that occurs as a result of transient fly height decrease. The comparison, however, is consistent in that the method employed for calculating the stiffness values is uniform.

In the second section of this chapter, a different approach is taken in an attempt to minimize the dynamic fly height variation. The approach is to minimize the sliders sensitivity to crown. (Crown is defined as the maximum height of a parabola that is pinned to the front and back edges of the slider.) The slider's dynamic response can be characterized as a response to the changing surface heights that it sees as it travels along the disk surface. The conformity of the slider to the disk surface is considered the effective crown of the slider (Stanley, 1993) and affects the spacing variation directly (Zhu, 1988). By minimizing the static sensitivity to crown, the intent is to minimize the sensitivity to effective crown that the slider sees. In effect, the procedure hopes to filter out the frequency in the response of the slider to the disk that corresponds to the slider's length. Although it is a single frequency in a continuous response, it is a frequency that makes a major contribution to the variation of the fly height. The results show that if the

nominal design is not changed considerably then a slight improvement in dynamic fly height variation can be achieved, however, if the design is changed considerably then the improvement in performance is not necessarily realized. The former is essentially finding the partial derivative with respect to the crown sensitivity which would account for the slight performance improvement whereas the former

MATHEMATICAL FORMULATION: SENSITIVITY OPTIMIZATION

OBJECTIVE FUNCTION

The cost function that is defined for this problem is:

$$f(\underline{x}) = 50.0 \times \Delta fh(nm) + 50.0\left(\overline{fh} - 14\right)(nm) + \sum roll(\mu rad) + \sum P(pitch - 200)(\mu rad) + 50.0 \times \sum (VS - 10.0)\left(nm/g\right) + 10.0 \times \sum (PS - 50.0)\left(\mu rad/g - mm\right) \quad (8.1)$$

The first term in the objective function represents the maximum difference in fly height over the radius of the disk multiplied by a weighting of 50. The next term represents the difference in average fly height over the radius of the disk minus the target fly height of 14nm multiplied by a weighting of 50. If the sum of these two terms is minimized to zero, the resulting design will fly flat over the radius of the disk at the targeted fly height of 14nm. The next term represents the sum of the absolute value of the roll of the slider at the three points used to evaluate the objective function (the inner radius, center radius, and outer radius). The next term sums the pitch, similar to the roll term, except that it has the form of a penalty function, i.e., it does not contribute to the objective function until

the pitch is greater than 200μrad. The penultimate term represents the sum of the slider air bearing vertical sensitivity to load at each of the evaluation points minus the target of 10nm/g multiplied by a weighting of 50. The final term represents the sum of the slider's pitch sensitivity at each of the evaluation points minus the target of 50 μrad/g-mm multiplied by a weighting of 10. By minimizing all of these terms simultaneously, an air bearing slider can be obtained that satisfies the basic design criteria, i.e., a flat fly height profile at the target fly height and roll minimization, and, presumably, a more advanced criterion, i.e., minimization of the fly height due to changes in the applied loads. The weights associated with each term are indicative of their relative importance.

CONSTRAINTS

In the optimization, 14 variables were allowed to vary. Of these 14, 13 were geometric constraints which are depicted in Figure 8.1.

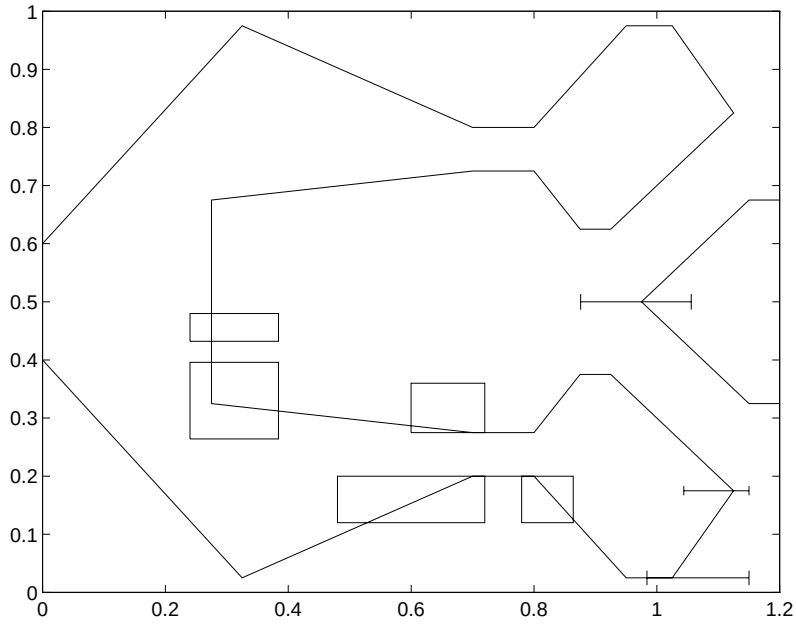

Figure 8.1: Original design with geometric constraints (all dimensions in mm)

The final variable included in the optimization was the recess depth. It was allowed to vary between 3 and 5 microns. All vertices that are symmetric to the vertices that are used as parameters in the optimization are made to vary symmetrically about the center line of the slider.

NUMERICAL RESULTS

After employing the simulated annealing algorithm on the above stated objective function subject to the defined constraints, the following improvements were made (Table 8.1):

Item	Initial value	Final value	Percent improvement
Total cost	1224.0	585.9	52.1
Δfh	101.0	126.1	-19.9
Variation from target	122.2	158.2	-29.5
Roll	5.23	43.3	-727.4
Vert. sens.	665.1	88.8	86.6
Pitch sens.	330.5	169.6	48.7

Table 8.1: Table of objective function values at inception and conclusion of optimization (values include weighting factors)

It is clear from the above table (Table 8.1) that, although the overall objective function made a considerable improvement, certain terms actually worsened. This is an inherent difficulty of multi-objective optimization. Due to the weightings, however, this reduction in optimality actually represents a very small physical change. The weighting of the Δfh term is 50.0 which implies that, although the term worsened by nearly 20%, the increase was only by 0.5nm. Similarly, for the difference from target term (second in the

objective function), the actual increase amounts to about 0.5nm. The roll term, however, due to its relatively small weighting, worsened considerably but its value was not considered crucial in the optimization procedure because the slider fly height is taken at the center of the slider. Its small weighting is indicative of this. The pitch term was not included in the table because its value never changed from zero, i.e., the pitch at any of the evaluation points never climbed above 200μrad.

Due to the small physical change in the first two terms and the relative unimportance of the roll term, our attention shifts to the sensitivity optimization terms. The vertical and pitch sensitivity terms both enjoyed a significant increase in optimality. The weighting of the vertical sensitivity term was 50.0 which implies that the final design came very close, physically, of satisfying the goal of 10nm/g. The original and final vertical stiffness values at the outer radius were 13.1 nm/g and 9.76 nm/g respectively. The values for the pitch sensitivity were 48.2 μrad/g-mm and 37.2 μrad/g-mm respectively.

At the conclusion of the optimization, the design had the form shown in Figure 8.2.

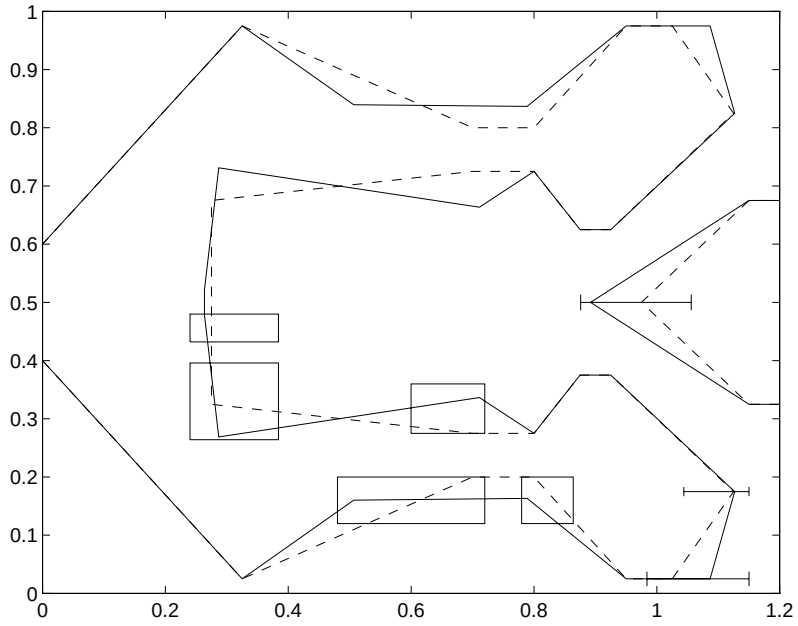

Figure 8.2: Optimized design (solid) with original design (dashed) and constraints (all dimensions in mm)

The recess depth of the final design was 3.1 microns, very close to the lower value of the constraint on the parameter.

At this point in the optimization, the new design shares the same characteristics with regard to the first two terms in the objective function as the original design, a larger roll (2.2 μrad originally vs. 28.6 μrad at the outer radius) and a considerably lower vertical and pitch sensitivity. One might expect this to lead to a smaller dynamic fly height modulation when flown over a non-flat surface. The opposite, however, was found to be the case.

The CML Air Bearing Dynamic Simulator (Hu, 1995) was employed to simulate the fly height of both the original and optimized design. The simulation considered the slider flying over a non-flat surface. The disk track data was taken from a measurement of an actual disk using a Zygo white light interferometer. The measured data had a standard deviation of 15.2 nm from its mean plane. The measured profile after high pass

filtering is shown in Figure 8.3. The simulation results for this track are shown in Figure 8.4 and Figure 8.5. The discrepancy in the mean fly height is accounted for by the fact that the dynamic fly height simulation was done at the outer radius of the disk (31mm) where the original design's static fly height dipped to roughly 10nm. The standard deviation of the dynamic fly height modulation was 0.878 nm and 1.16 nm for the original and optimized designs respectively. The plots (Figure 8.4 and Figure 8.5) for the absolute fly height of the slider at the read-write element point (25 microns forward of the absolute back of the trailing edge) depict graphically that the fly height variation is indeed greater for the optimized design.

The dynamic simulations covered 16ms of real time with one μs time steps. The simulation was done at the outer radius of the disk (31mm) with a rotational speed of 5400rpm. The plots shown for the dynamic fly height modulation start at the 200th time step, or, equivalently, at 200μs.

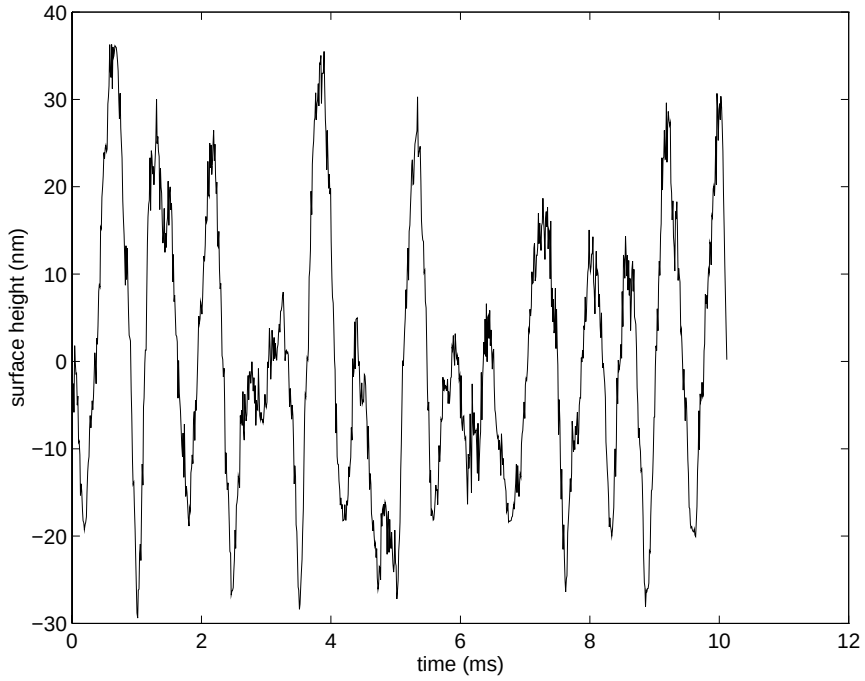

Figure 8.3: Measured, filtered surface used in dynamic fly height simulation

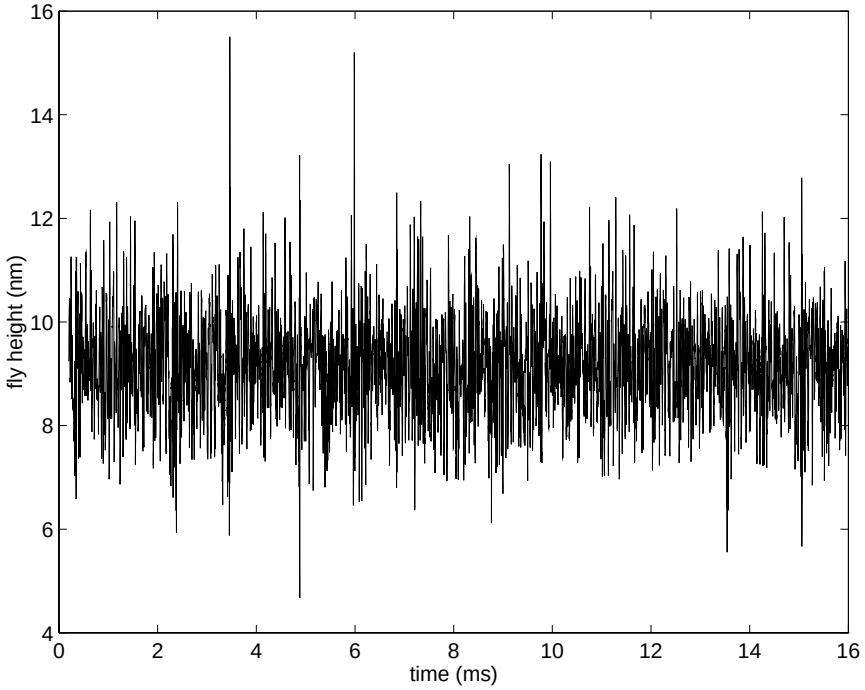

Figure 8.4: Calculated absolute fly height of original design

The reason for this is that the simulation has an initial fly height for the slider that is far above its mean dynamic fly height. In order to compare the deviations in fly height of the

two designs meaningfully, the first two hundred steps were omitted in the calculation so that the true dynamic nature of the bearing would be illustrated.

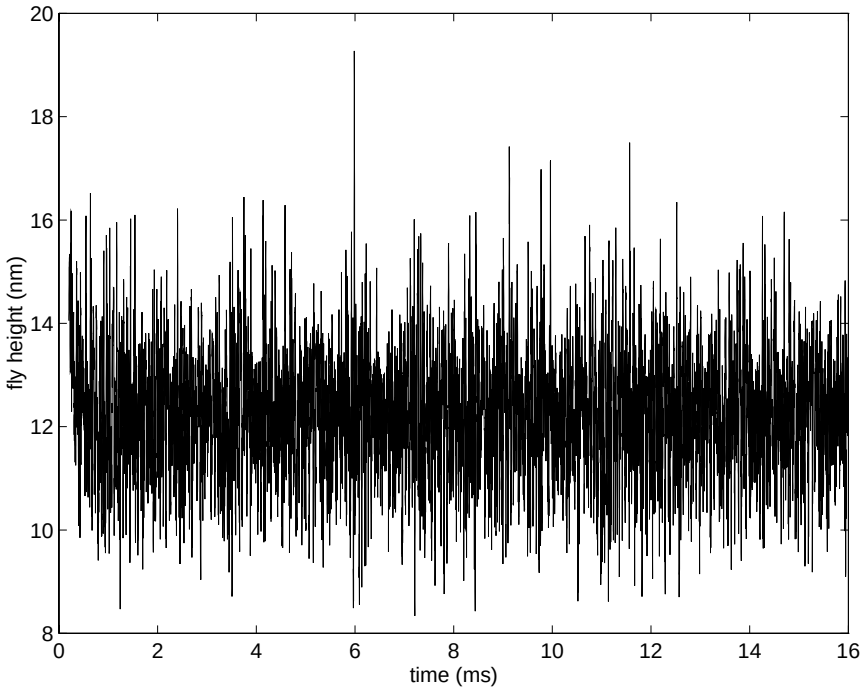

Figure 8.5: Absolute fly height of optimized design

MATHEMATICAL FORMULATION: CROWN SENSITIVITY OPTIMIZATION

OBJECTIVE FUNCTION

The cost function is redefined to represent the desire to minimize the crown sensitivity The function is written as (Eq. 8.2):

$$f(\underline{x}) = 10.0 \times (fh - 14)(nm) + roll(\mu rad) + \\ +10.0 \times P(pitch - 200) + 20.0 \times S_{crown} \quad \textbf{(}8.2)$$

As in chapter 6 above, the objective function (Eq. 8.2) has combined the first two terms from the previous objective function (Eq. 8.1). The objective function also contains

terms that minimize the roll and limit the pitch through use of a penalty function. The last term in this equation represents the crown sensitivity term. The intent, as described above, is to minimize the slider's sensitivity to crown. Its weighting is greater than any other term in the objective function which indicates its relative importance in the optimization. None of the terms is summed due to the fact that only a single evaluation point was used in calculating the objective function. The evaluation point chosen was at the outer radius of the disk (31mm radius, -17.39 deg. skew).

The crown sensitivity term is defined as the change in fly height due to changes in the crown. Calculation of numerical derivatives can, however, lead to wildly inaccurate values. The following graph (Figure 8.6) depicts the fly height of the original design versus crown. The nominal crown value is 10nm and the nominal fly height is 12.6nm. The lighter colored line represents calculation of the fly height calculated by recreating the computational grid at each evaluation. The darker line represents the fly height calculated by creating the computational grid at the nominal crown value and then retaining that for calculations at all the neighboring points. As mentioned above, if the numerical derivative were to be calculated using the former method, it would poorly predict the actual sensitivity. Using the latter method, the derivative calculation is much more accurate (Figure 8.7). The derivative is a simple, two-point backward difference. Figure 8.7 depicts the value of the derivative at each point versus the nominal crown value. Using the method of grid retention in calculating the fly heights yields a linearly varying curve (Figure 8.6) which in turn yields a nearly constant sensitivity value. Through inspection of these graphs, it is clear that when calculating the crown sensitivity,

the change in crown should be at least 2nm. This was the value chosen for the subsequent calculations.

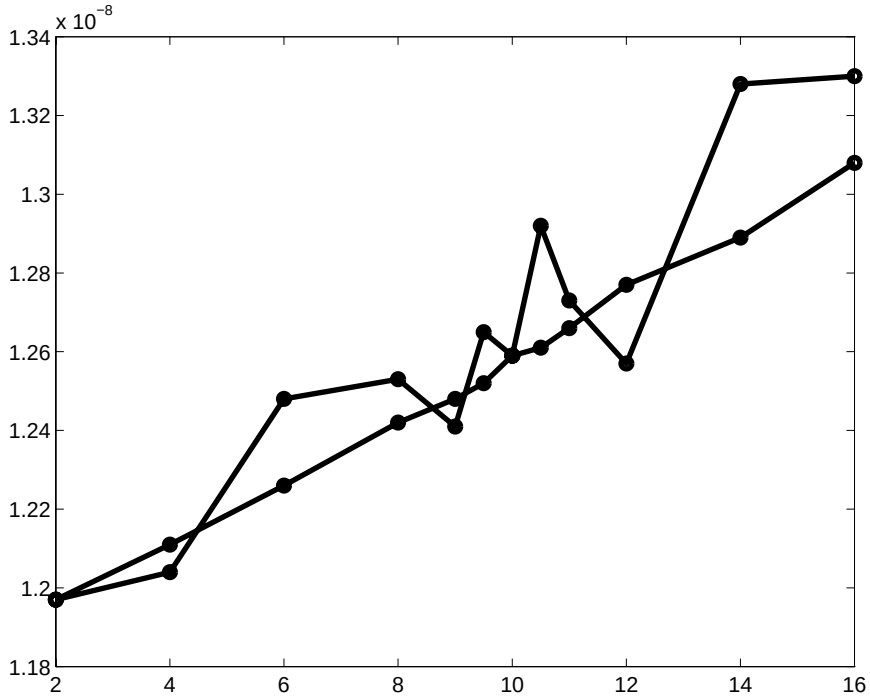

Figure 8.6: Fly height versus crown for original design (light colored line: same computational grid for all points, dark line: new computational grid created at each point)

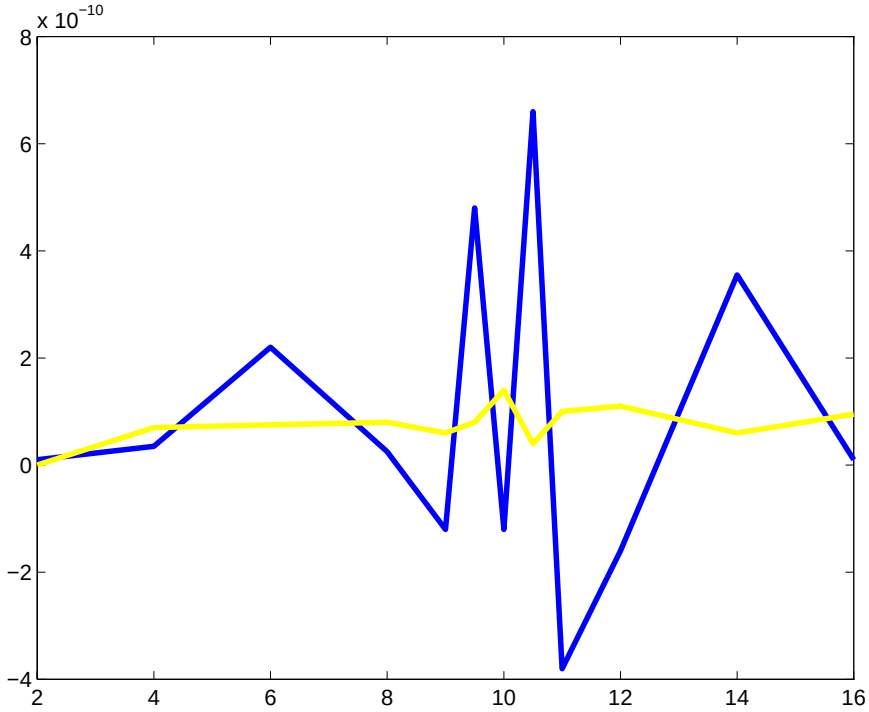

Figure 8.7: Sensitivity calculated from fh vs. crown data (light colored line: same computational grid for all points, dark line: new computational grid created at each point)

CONSTRAINTS

The constraints employed in this optimization were exactly the same as those used in the sensitivity optimization (Figure 8.1). The initial condition was taken to be the same as well.

NUMERICAL RESULTS

The following figure (Figure 8.8) depicts the initial solution (dark line), the optimal solution (light line) and the constraints (as described above). The design has not changed appreciably. The most pronounced difference comes at the back of the main rail where the two “feet” have been extended slightly. The improvement in the objective function, however, was marked for the crown sensitivity term.

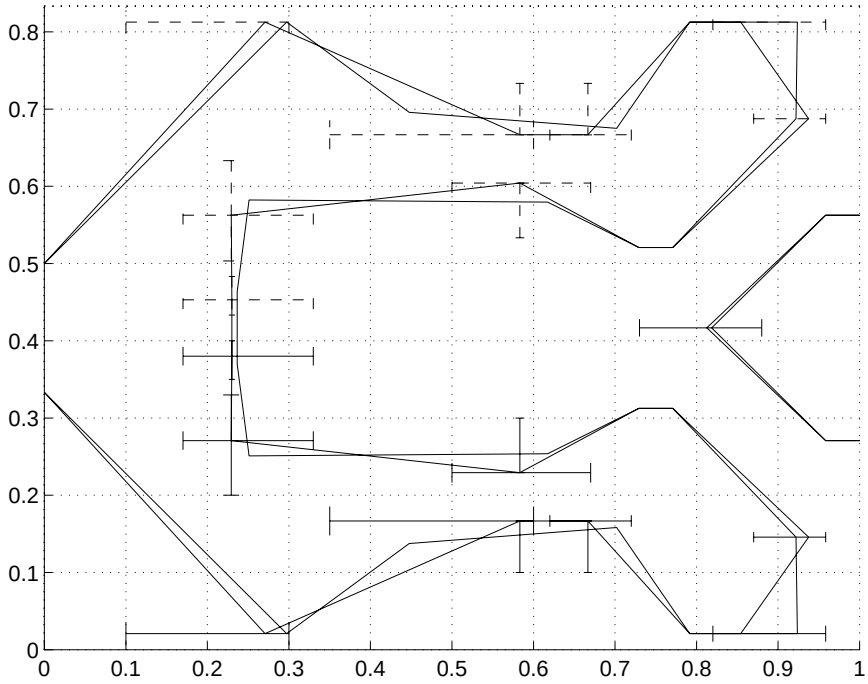

Figure 8.8: Original and optimal design with constraints (normalized coordinates)

The fly height targeted was achieved and the crown sensitivity was reduced from 0.33nm/nm to 0.09nm/nm which led to a reduction in dynamic fly height variation (simulated over a measured disk) from 8.78e-10m for the original design to 8.45e-10m for the optimized design (standard deviation from mean fh). This equates to a 3.8% improvement. Although, the improvement is slight, it is in the standard deviation and not peak to valley.

The progression of the objective function is depicted in the following graph (Figure 8.9):

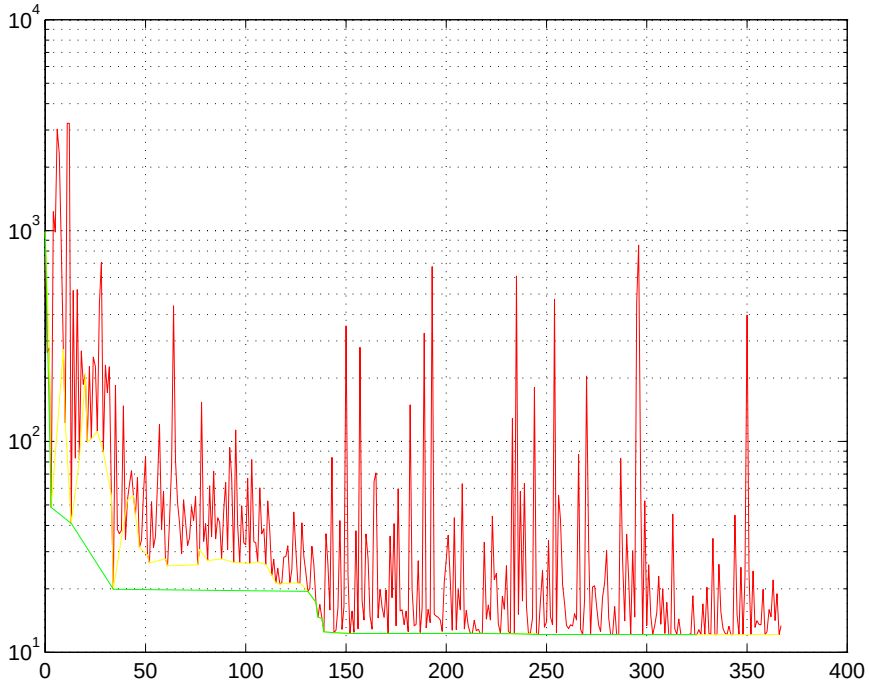

Figure 8.9: Objective function value vs. iteration number for first crown optimization

The progression of the crown term is depicted in the following graph (Figure 8.10):

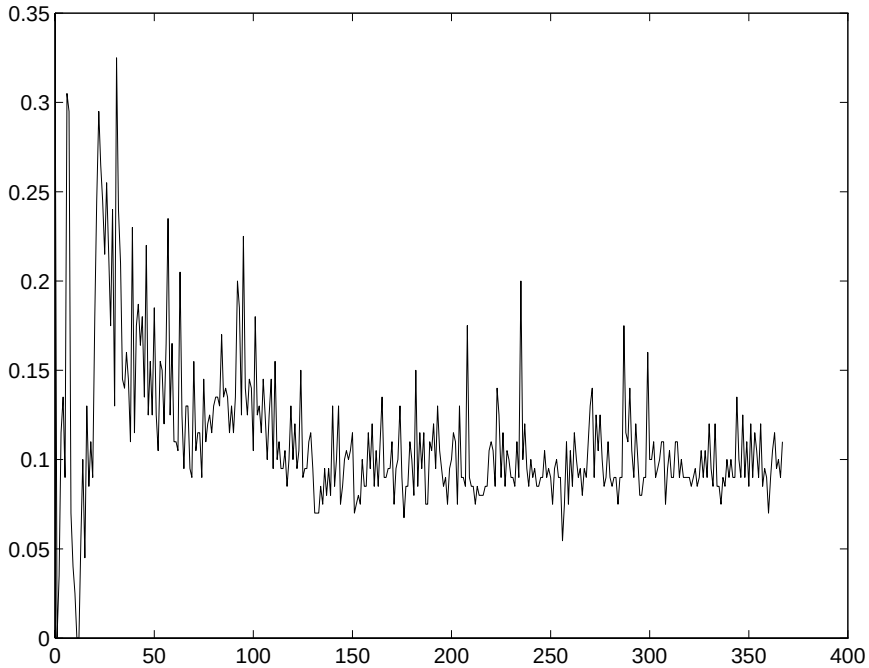

Figure 8.10: Crown term value vs. iteration number for first crown optimization

The progression of the crown term shown in the above graph (Figure 8.10) shows that an even greater improvement is possible. Because the optimization procedure balances all terms that exist in the objective function, the crown term's minimum value occurred previous to the optimal configuration. This implies that if the weighting of the crown sensitivity were greater, it could be minimized further ideally providing an even greater improvement in dynamic fly height variation.

Based on this observation, the objective function was rewritten as (Eq. 8.3):

$$f(\underline{x}) = 10.0 \times (fh - 14)(nm) + roll(\mu rad) + \\ +10.0 \times P(pitch - 200) + 100.0 \times S_{crown} \quad \textbf{(}8.3\textbf{)}$$

The only difference between Eq. 8.2 and Eq. 8.3 is that crown sensitivity term is weighted with a factor 100.0 in the latter as opposed to only 20.0 in the former. The design that resulted from this modification of the objective function is shown below (Figure 8.11).

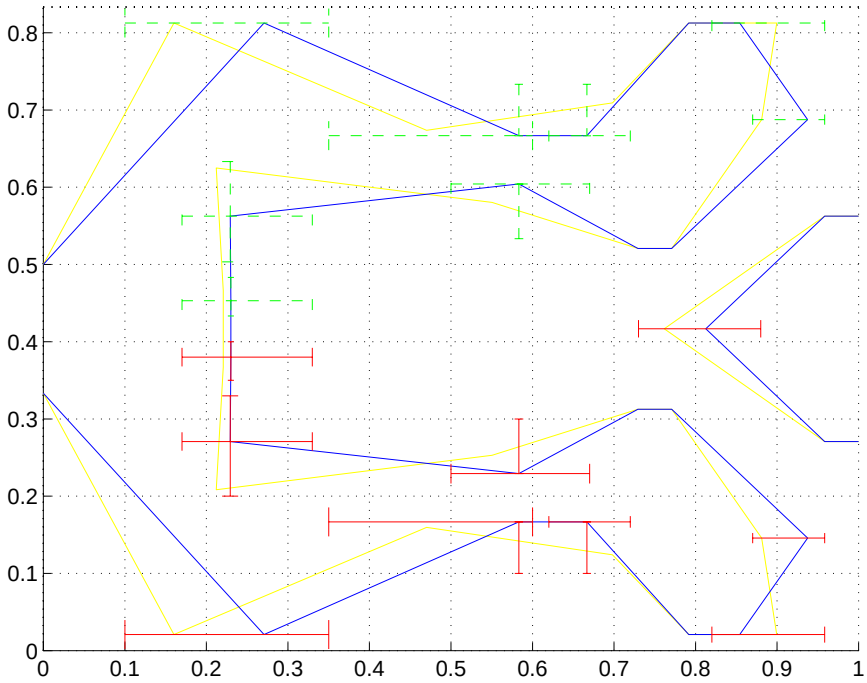

Figure 8.11: Original and optimal designs (including constraints) for second crown sensitivity optimization

The original and optimal design vary significantly in this case. The front of the air bearing has been moved up and flattened, the sub-ambient pressure cavity has been widened, the side rails have been thinned and the trailing edge has been moved up. It is expected that to achieve a reduction in crown sensitivity, the surface area of the bearing would be reduced. It is clear from the above graph (Figure 8.11) that this is, in fact the case.

Although this design also satisfied the fly height objective and further improved on the crown sensitivity--from 0.09nm/nm to 0.035 nm/nm--the dynamic fly height variation worsened--from 8.4967e-10m to 9.5907e-10m (measured in standard deviation from mean fh). This represents a 12.9% increase in fly height variation.

CONCLUSIONS

It is shown above that improvement in the vertical and pitch stiffness of an air bearing does not ensure that the dynamic fly height modulation will decrease. It is also show that strict improvement of the crown sensitivity of a slider does not necessarily improve the dynamic fly height variation. In all optimizations, the goal was ostensibly achieved, i.e., the static characteristic pursued was satisfied. This demonstrates, however, that optimization of static parameters does not necessarily improve dynamic bearing characteristics.

CHAPTER 9

SUSPENSION OPTIMIZATION

INTRODUCTION

In all of the previous examples, the simulated annealing algorithm was applied to air bearing sliders. In this chapter, the algorithm is applied to the problem of suspension design.

In hard disk drives, sliders house the magnetic elements that read and write data to the spinning magnetic disks. The slider's position is controlled to access different sections on the disk. In modern disk drives, this position control is achieved through a rotary voice-coil motor. Connected to this motor is a relatively rigid actuator arm which is in turn connected to a suspension-gimbal assembly. At the end of this assembly is the slider. The suspension is designed to have dynamic characteristics that allow the slider to follow tracks on the spinning magnetic disk as closely as possible. Disks inside the disk drive have some amount of distortion due to manufacturing variance and clamping. This variance is called runout. The slider's ability to follow this runout without varying its spacing from the disk is called slider following capability (Utsunomiya, et. al., 1996).

The difficulty that arises in designing suspensions that maximize slider following capability is that there is a trade-off between stiffness in the seek direction (motion of the slider in the radial direction) and compliance in the roll and pitch direction which decide the slider following capability (Tokuyama, et. al., 1987). Further, the suspension should preserve frequency response to improve actuator servo design. If there are large resonances between 0 and 20kHz, large off-track error can occur (off-track error is defined as deviance in radial position of the slider from its targeted value).

Yang and Tu (1996a) proposed optimization of suspension assemblies (suspension and gimbal) that consisted of raising the natural frequencies of the suspension in order to improve the servo system response. The design variables employed controlled the shape of the suspension. The suspension type optimized, however, was for a linear actuator system (one that moves the slider across the disk along a single radius) which is typically found in earlier drive designs. In the study, the natural frequencies of the suspension were taken as terms in the objective function and sensitivity analysis was performed at the conclusion of the optimization. This optimized design was then subjected to a control analysis where the performance was quantified (Yang and Tu, 1996b). Control system performance will not be discussed in this thesis.

The suspension optimization investigated in this thesis will address the issue of load force sensitivity. The suspension supplies a force that acts on the top of the slider that balances the air bearing force that the slider develops. Of all the parameters that affect the fly height of a slider, typically, the suspension force has the greatest impact. The suspension force is created by initially manufacturing the suspension with a bend angle then subsequently making the suspension flat. The residual stress that exists in the

suspension structure creates the downward force mentioned above. In the course of manufacturing and handling, however, the bend angle can be inadvertently changed. The result of this will be to change the downward force acting on the slider. The amount of change, however, is dependent on the sensitivity of the design. This value, known as the spring rate is the objective function that will be minimized.

The basic structure that is being optimized is shown below (Figure 9.1, Figure 9.2 and Figure 9.3).

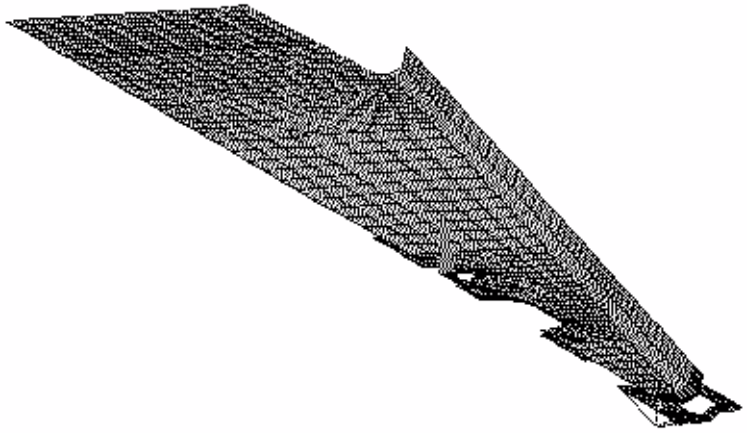

Figure 9.1: Isometric view of half suspension assembly used in optimization

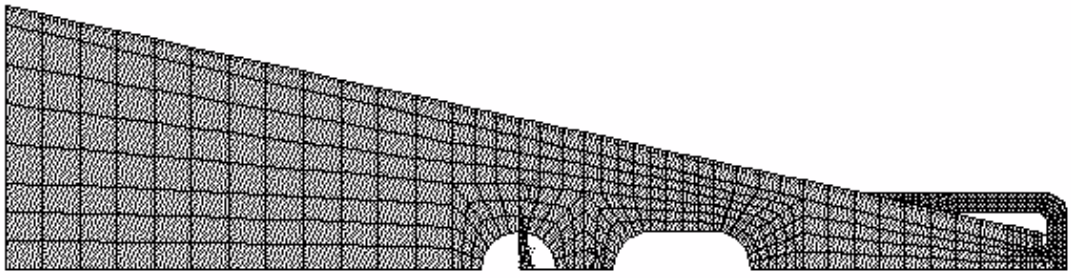

Figure 9.2: Plan view of half suspension assembly used in optimization

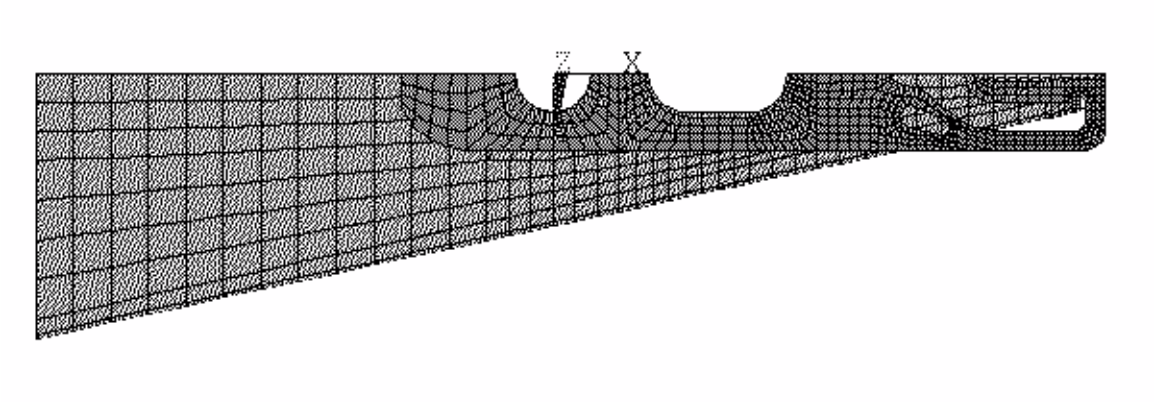

Figure 9.3: Bottom view of half suspension assembly used in optimization

MATHEMATICAL FORMULATION

OBJECTIVE FUNCTION

The objective function to be minimized is simply the spring rate of the suspension. The spring rate is defined as the change in force due to a change in deflection at the cantilevered end of the suspension. It is calculated using the Ansys® finite element analysis package. The parameters are generated by the simulated annealing algorithm and passed to Ansys® which then performs the analysis that determines the spring rate. This information is then read back into the simulated annealing algorithm and the process is repeated until one of the stopping criteria defined in chapter 5 is met. The objective function is simply written as:

$$f(\underline{x}) = k_{rate} \tag{9.1}$$

where $\underline{x}$ is combination of parameters defined below.

Two combinations of cooling schedule and acceptance criteria are used. Specifically, using the cooling schedule defined in Chapter 5 (eq. 5.20), the values for α and β for each combination were (2,2) and (1,2), respectively. The second of these two

combinations lowers the temperature more slowly, i.e., it allows for more configurations to be examined before termination of the algorithm.

CONSTRAINTS

The parameters employed in the optimization control the structure of the base of the suspension. As described above, the suspension attaches to the actuator arm through the suspension base. It is here that the bend angle is defined. Through varying the length of the base and its width, the spring rate can be optimized. An outline of the structure that is being optimized is shown below.

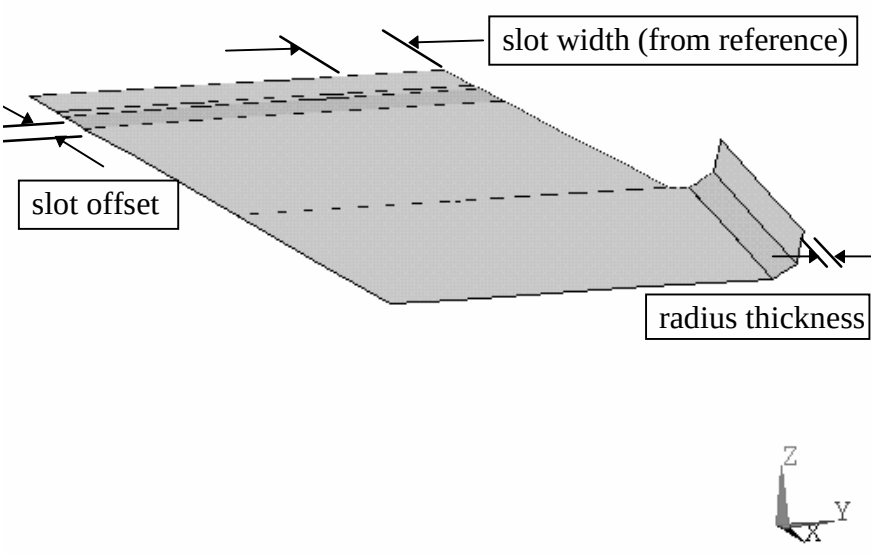

Figure 9.4: Frame of suspension base used in optimization

The parameters that control the shape of the base are the so-called slot offset, slot width and radius thickness. The slot offset is the distance from the actuator arm to the beginning of the suspension assembly shown above (Figure 9.1). The slot width is the thickness of this region when added to a reference value and the radius thickness is the thickness of the flange that runs the length of the suspension assembly. All dimensions

are given in millimeters. The following table (Table 9.1) gives the parameter ranges and initial values.

Parameter	lower value	upper value	initial value
slot offset	0.1	2	0.1
slot width	0.1	0.3	0.3
radius thickness	0.0254	0.0572	0.0381

Table 9.1: Table of parameter ranges and initial values

NUMERICAL RESULTS

The progression of the parameter and cost temperatures for the first cooling schedule employed is shown below (Figure 9.5).

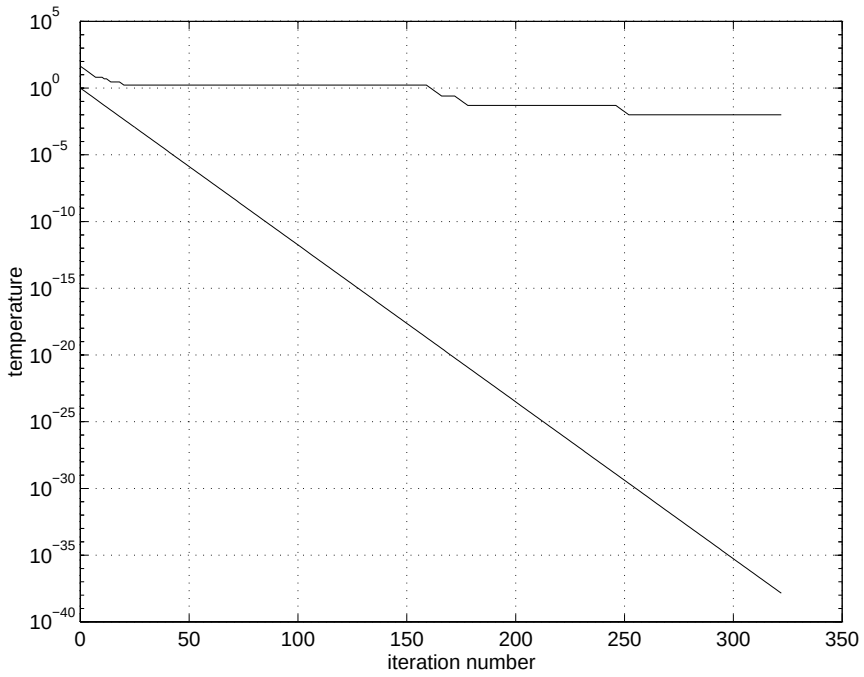

Figure 9.5: Plot of cost (upper) and parameter (lower) temperatures vs. iteration

Similar to the plots shown in chapter 5, there are two curves on the above graph. The upper curve represents the cost temperature which has a non-uniform rate of decrease due to the linkage of temperature decrease to acceptance criterion as described in chapter 5.

The lower curve represents the parameter temperature that decreases uniformly due to the fact that it is only linked to the iteration number.

The objective function progression is shown in the following plot (Figure 9.6).

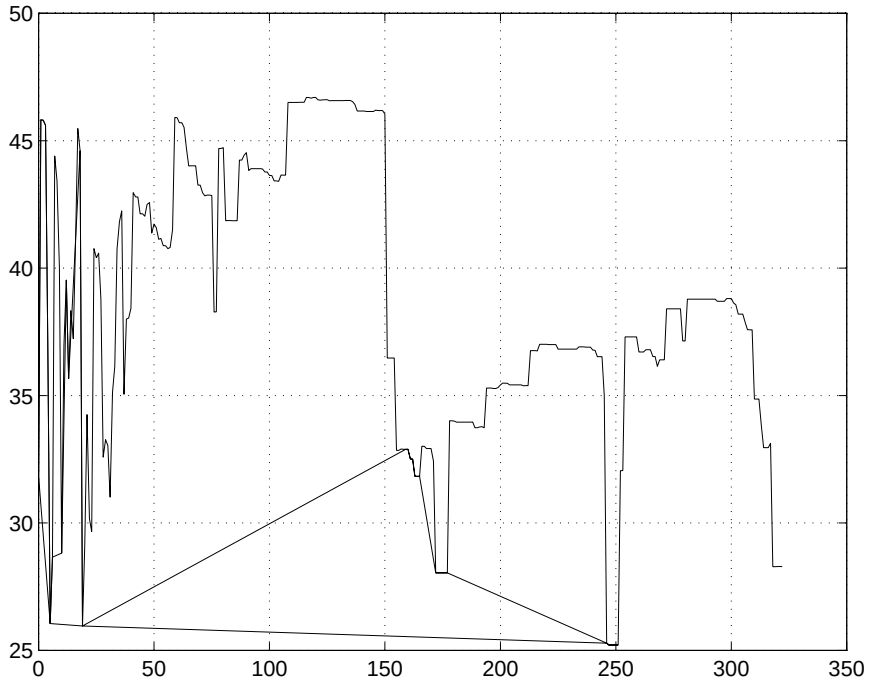

Figure 9.6: Progression of objective function vs. iteration

In the above figure (Figure 9.6), there are three lines. The line that bounds the other two from above represents the cost associated with each configuration generated. The line that bounds the other two from below is the progression of the optimal solution. The line that sits between these two is the progression of the currently accepted state as described in chapter 5. From this graph we can see that the objective function has a high sensitivity to one or more of the parameters in the problem perhaps even a combination of them. The following table (Table 9.2) shows the value of the parameters at the conclusion of the optimization.

Parameter	Final value
slot offset	0.575
slot width	0.179
radius thickness	0.0268

Table 9.2: Table of final parameter values

The initial and final values of the objective function corresponding to the parameter sets given (Table 9.1 and Table 9.2) are 31.85 and 25.21 respectively. The units of the spring rate are mN/mm.

Using the second cooling schedule mentioned above (alpha, beta) = (1,2) yielded the following temperature plot (Figure 9.7).

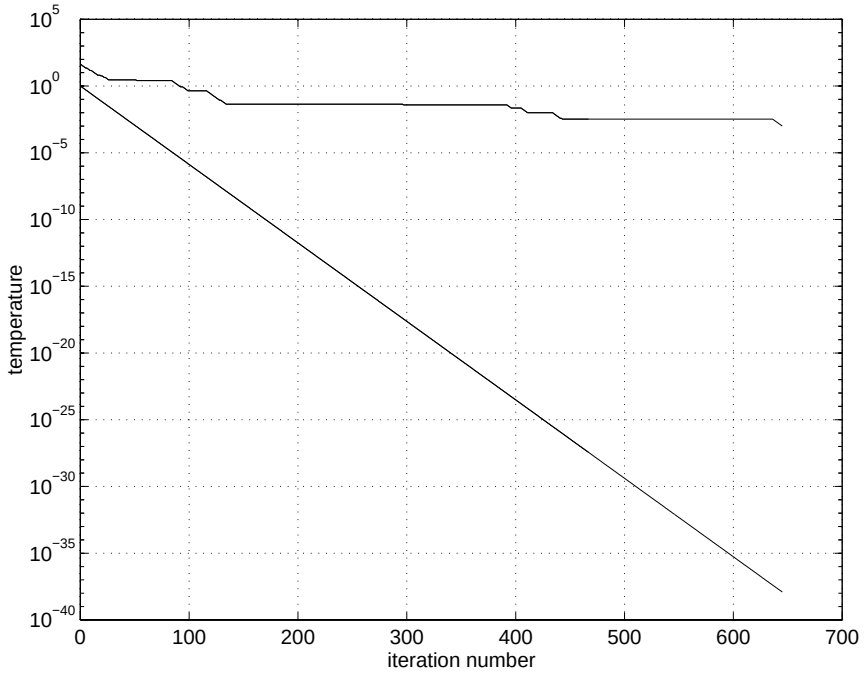

Figure 9.7: Plot of cost (upper) and parameter (lower) temperatures vs. iteration for second cooling schedule

The above plot (Figure 9.7) shows that this cooling schedule reduced the temperatures in a much more gradual fashion than previous one (Figure 9.5). This allowed a greater number of configurations to be considered before termination of the algorithm. Because the cooling schedule lowers the temperature more slowly than the

previous cooling schedule, the result achieved is expected to be better, i.e., the algorithm is coming closer to the asymptotic equalities described in chapter 5 that are necessary for global convergence. A plot of the objective function value versus iteration (Figure 9.8) shows that near the outset of the algorithm, the minimum value was found and that as the algorithm concluded, it was approaching this value asymptotically. This is an example of the balance that has to be achieved in a finite-time algorithm--speed versus quality of solution. Were the algorithm to continue, it would eventually arrive at the absolute minimum it achieved earlier in the solution process.

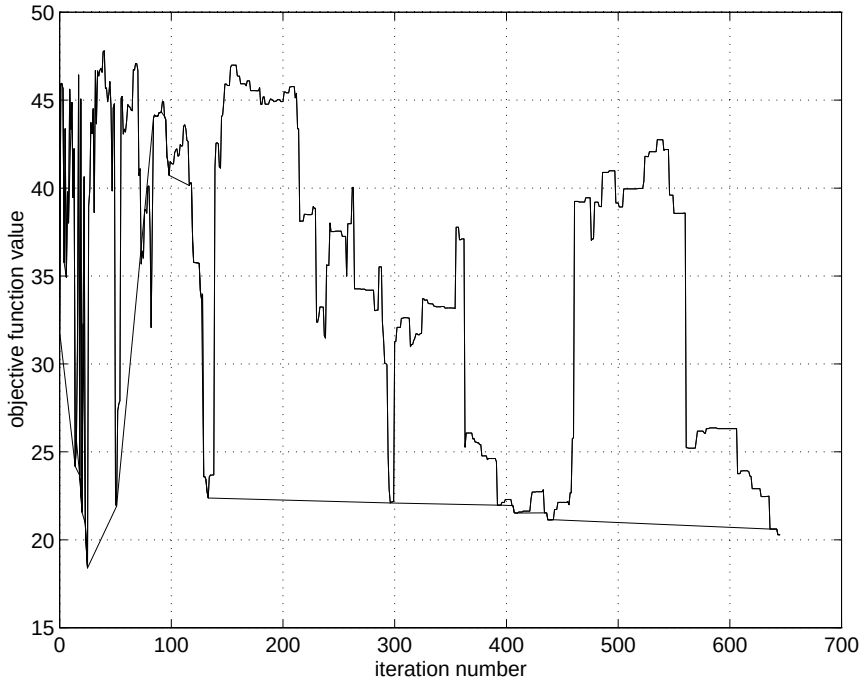

Figure 9.8: Progression of objective function vs. iteration for second cooling schedule

It is evident from the above plots (Figure 9.6 and Figure 9.8) that when the cooling schedule lowers the temperature more slowly, the final solution is more optimal. Even the value that is being approached asymptotically by the algorithm when the second cooling schedule is used is more optimal than the final value achieved when the first

cooling schedule is used. This is consistent with the derivation of the algorithm given in chapter 5.

CONCLUSION

This chapter gives an exposition of the issues related to suspension optimization and provides an example of one such optimization using two different cooling schedules. The objective function minimized was the spring rate of the suspension which has a large impact on the manufacturability of a suspension. It is shown that structural optimization is feasible and that, due to the sensitivity of the parameters involved, a cooling schedule that lowers the temperatures more slowly is necessary.

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

It has been demonstrated throughout the course of this dissertation that the concept of applying global optimization to the problems of air bearing slider and suspension design is tractable and achievable. The results of the many example problems support this claim.

Chapter 1 provided an introduction to the field of hard disk drive research. The relationship between slider disk spacing and storage density is explained. The continuous reduction of this spacing over the years is shown. This chapter also provides a basic introduction to and motivation for optimization as it relates to hard disk drive design. Some of the reasons that make the problem interesting are discussed.

Chapter 2 discusses the solution of the Reynolds' lubrication equation. Starting with the generalized equation, the process of solution is described step by step. The method of control volume discretization is discussed followed by the coupled line-

sweeping multi-grid method of solution. Various details of the implementation are described and their relevance to the optimization procedure is explained.

Chapter 3 provides a thorough introduction to modern day optimization techniques. Both deterministic and stochastic methods are discussed in detail. The mathematical details are also described at length. The strengths and weaknesses of the various methods are described as well as their suitability for various types of problems.

Chapter 4 compares the Simulated Annealing (SA) and Genetic Algorithm (GA) optimization methods. Through comparison of these two stochastic methods it is found that SA is better suited for slider air bearing optimization. This chapter also demonstrates that the SA algorithm is, indeed, capable of finding global minima over a complicated search space. This verification was achieved by mapping the objective function over the entire search space, determining exactly where the global minimum was and correlating it to the result the algorithms obtained.

After determining that SA would be the method of choice, Chapter 5 gives a thorough development of the algorithm. The implementation that is specific to this research is shown through use of an example problem. The description of the algorithm given in this chapter is used throughout the remainder of the work. Chapter 5 also provides the theoretical basis for SA's ability to find global optima.

Chapter 6 discusses the application of SA to the problem of slider altitude sensitivity optimization. This chapter gives background for the importance of this problem and formulates it for solution by SA. The result demonstrates that SA is capable of optimizing a slider design with complex constraints and multi-goal objective function.

Further, it demonstrates that SA satisfies the necessary condition that expanded search spaces yield superior objective function values.

Chapter 7 discusses the application of SA to the problem of sub-ambient pressure minimization. This is an attempt to control tribological characteristics of the slider through characteristics of the air bearing. This chapter demonstrates, again, that problems with complex constraints and a multi-goal objective function can be minimized through the use of SA.

Chapter 8 examines the application of SA to the problem of slider dynamic fly height variation. Through modification of static characteristics of the bearing, an attempt was made to modify the dynamic characteristics of the bearing. Although the optimization algorithm optimized the quantities that were included in the objective function, the resulting designs did not minimize the dynamic fly height variation. If the optimization routine could be effectively coupled with dynamic simulation, the problem could be solved directly. However, due to the prohibitive cost associated with dynamic simulation, the approach was not considered feasible.

Chapter 9 probes the application of SA to the problem of suspension optimization. It demonstrates that the procedure of structural optimization is possible through application of the SA algorithm using an appropriately generous cooling schedule.

In sum, the problem of slider air bearing optimization has been examined in detail using both theoretical and practical approaches and the problem of suspension design has also been explored and has been shown to be feasible.

FUTURE WORK

The problem of slider air bearing optimization has been presented in a very general framework with specific examples provided. The most fruitful direction to follow would be the application of optimization to the problem of dynamic fly height variation. If the controlling characteristics could be determined more exactly, the optimization procedure could improve more consistently, the dynamic fly height variation. The area that has the greatest amount of room for work, however, is suspension optimization. Coupling the work done here with that of Chang and Tu (1996a) to find a suspension design that has both manufacturability and controllability characteristics would be a significant contribution to the field of hard disk drive research.

REFERENCES

Baker, J. E., 1987, “Reducing bias and efficiency in the selection Algorithm,” *Proc. Second International Conference on Genetic Algorithms*, Lawrence Erlbaum, Hillsdale, N.J., pp. 14-21.

Best, G. L., Deckert, K. L., Millman, S. E. and Singh, G. P., 1988, “Effect of Disk Roughness on Slider Dynamics,” *IEEE Transactions on Magnetics*, Vol. 24, No. 6, pp. 2748-2750.

Bolasna, S., 1990, “Air Bearing Parameter Effects on Take-Off Velocity,” *IEEE Transactions on Magnetics*, Vol. 26, No. 6, pp. 3033-3038.

Brandt, A., 1977, “Multi-Level Adaptive Solutions to Boundary-Value Problems,” *Mathematics of Computations*, Vol. 31, No. 138, pp. 333-390.

Burgdorfer, A., 1959, "The Influence of the Molecular Mean Free Path on the Performance of Hydrodynamic Gas Lubricated Bearings," *Transactions of the ASME Journal of Basic Engineering*, Vol. 81, pp. 94-100.

Castelli, V. and Pirvics, J., 1968, “Review of Numerical Methods in Gas Bearing Film Analysis,” *ASME Journal of Lubrication Technology*, Vol. 90, pp. 777-792.

Cha, E. and Bogy, D. B., 1991, “Numerical Simulation and Animation of Head-Disk-Assembly Dynamics,” *IEEE Transactions on Magnetics*, Vol. 27, No. 6, pp. 5136-5138.

Cha, E. T., 1993, “Numerical Analysis of Head-Disk Assembly Dynamics for Shaped Rail Sliders with Sub-ambient Pressure Regions,” *Doctoral Dissertation*, Department of Mechanical Engineering, University of California, Berkeley.

Cha, E. T. and Bogy, D. B., 1995, “A Numerical Scheme for Static and Dynamic Simulation of Subambient Pressure Shaped Rail Sliders,” *ASME Journal of Tribology*, Vol. 117, pp.36-46.

Cha, E., Chiang, C. and Lee, J. J. K., 1995, “Flying Height Change During Seek Operation for TPC Sliders,” *IEEE Transactions on Magnetics*, Vol. 31, No. 6, pp. 2967-2969.

Clifford, G. and Henze, D., 1989, “An Air Bearing Minimizing the Effects of Slider Skew Angle,” *IEEE Transactions on Magnetics*, Vol. 25, No. 5, pp. 3713-3715.

Cooper, E. S., 1990, "Disk File Access-Time Constraints Imposed by Magnetic Air-Bearing Compliance," *IBM Journal of Research and Development*, Vol. 34, No. 5, pp. 668-679.

Dantzig, G. B., 1963, "Linear Programming and Extensions," Princeton University Press, Princeton, NJ.

Dixon, L. C. W., 1978, "Global Optima Without Convexity," *Technical Report*, Numerical Optimization Centre, Hatfield Polytechnic, Hatfield, England.

Fukui, S. and Kaneko, R., 1988a, "Analysis of Ultra-Thin Gas Film Lubrication Based on the Linearized Boltzmann Equation: First Report-Derivation of a Generalized Lubrication Equation Including Thermal Creep Flow," *ASME Journal of Tribology*, Vol. 110, pp. 335-341.

Fukui, S. and Kaneko, R., 1988b, "Analysis of Flying Characteristics of Magnetic Heads with Ultra-Thin Spacings Based on the Boltzmann Equation," *IEEE Transactions on Magnetics*, Vol. 24, No. 6.

Fukui, S. and Kaneko, R., 1990, "A Database for Interpolation of Poiseuille Flow Rates for High Knudsen Number Lubrication Problems," *ASME Journal of Tribology*, Vol. 112, pp. 78-83.

Gans, R. F., 1985, "Lubrication Theory at Arbitrary Knudsen Number," *ASME Journal of Tribology*, Vol. 107, No. 3, pp. 431-433.

Gelfand, S. B. and Mitter, S. K., 1991, "Simulated Annealing Type Algorithms for Multivariate Optimization," *Algorithmica*, Vol. 6, No. 3, pp. 419-436.

Greffenstette, J. J., 1986, "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, SMC-16, No. 1, pp. 122-128.

Gross, W., 1980, *Fluid Film Lubrication*, New York, Wiley.

Hamrock, B. J., 1991, "Fundamentals of Fluid Film Lubrication," *NASA reference publication*, #1255, Washington, D.C.

Hardie, C., Menon, A., Crane, P. and Egbert, D., 1994, "Analysis and Performance Characteristics of the Seagate Advanced Air Bearing Slider," *IEEE Transactions on Magnetics*, Vol. 30, No. 2, pp. 424-432

Holland, J. H., 1975, "Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence," *Doctoral Dissertation*, University of Michigan, Ann Arbor.

Holland, J. H., 1992, "Genetic Algorithms," *Scientific American*, Vol. 267, No. 1, pp. 44-50.

Horst, R., 1986, "A General Class of Branch-and-Bound Methods in Global Optimization with Some New Approaches for Concave Minimization," *Journal of Optimization Theory and Applications*, Vol. 51, No. 2, pp. 271-91.

Horst, R. and Tuy, H., 1987, "On the Convergence of Global Methods in Multiextremal Optimization," *Journal of Optimization Theory and Applications*, Vol. 54, No. 2, pp. 253-271.

Hsia, Y.-T., and Domoto, G. A., 1983, "An Experimental Investigation of Molecular Rarefaction Effects in Gas Lubricated Bearings at Ultra-Low Clearances," *Journal of Tribology*, Vol. 112, pp. 78-83.

Hsia, Y.-T. and Chang, C., 1995, "Monte-Carlo Statistical Analysis of Non-Linear Flying Attitude," *IEEE Transactions on Magnetics*, Vol. 31, No. 6, pp. 2988-2990.

Hu, Y., 1995, "The CML Air Bearing Dynamic Simulator," *CML Report*, No. 95-011, Computer Mechanics Laboratory, University of California, Berkeley.

Hu, Y., 1996, "Head-Disk-Suspension Dynamics," *Doctoral Dissertation*, Department of Mechanical Engineering, University of California, Berkeley.

Hu, Y. and Bogy, D. B., 1996b, "Spacing Modulation of a Slider Flying Across a Transition Between Landing and Data Zones," *IEEE Transactions on Magnetics*, Vol. 32, No. 5, pp. 3714-3716.

Ingber, L., 1994, Adaptive Simulated Annealing, Lester Ingber Research, McLean, VA.

Jain, P., 1989, "A Vector Quantization Multistart Method for Global Optimization," *Doctoral Dissertation*, Department of Mechanical Engineering, University of California, Berkeley.

Kirkpatrick, S., Gelatt, C. D., Jr., Vecchi, M. P., 1983, "Optimization by Simulated Annealing," *Science*, Volume 220, No. 4598, pp. 671-680.

Levy, A. V., Gomez, S., 1985, "The Tunnelling Method Applied to Global Optimization," *Numerical Optimization 1984*, eds. Boggs, P. T., Byrd, R. H. and Schnabel, R. B., pp. 213-244.

Liu, B. and Soh, S. H., 1996, "Effects of Seeking Velocity on Air Bearing Skew Angle, Air Flow Speed and Flying Performance of Sliders with Different ABS Designs," *IEEE Transactions on Magnetics*, Vol. 32, No. 5, pp. 3693-3695.

Lu, S., and Bogy, D. B., 1994, “A Multi-Grid Control Volume Method for the Simulation of Arbitrary Shaped Slider Air Bearings with Multiple Recess Levels”, *CML report*, No. 94-016, Computer Mechanics Laboratory, University of California, Berkeley.

Lu., S., Stanley, H. M., Bogy, D. B., Bhatia, C. S. and Hsia, Y.-T., 1995, “Design, Simulation, Fabrication and Measurement of a 25 nm, 50% Slider,” *IEEE Transactions on Magnetics*, Vol. 31, No. 6, pp. 2952-2954.

Lu; S., Hu; Y., O'Hara, M.; Bogy, D. B., 1996, “Air Bearing Design, Optimization, Stability Analysis and Verification for Sub-25 nm Flying,” *IEEE Transactions on Magnetics*, Vol. 32, No. 1, pp. 103-109.

Matsumoto, M., Takeuchi, Y., Agari, H. and Takahashi, H, 1994, “Design and Performance of Novel Air Bearing Slider,” *IEEE Transactions on Magnetics*, Vol. 30, No. 6, pp. 4158-4160.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E., 1953, “Equation of State Calculations by Fast Computing Machines,” *J. Chem. Phys.*, Vol. 21, No. 6, pp. 1087-1092.

Miu, D. K. and Bogy, D. B., 1986, “Dynamics of Gas-lubricated Slider Bearings in Magnetic Recording Files--Part II: Numerical Simulation,” *ASME Journal of Tribology*, Vol. 108, pp. 589-593.

Mladineo, R. H., 1986, “An Algorithm for Finding the Global Maximum of a Multimodal, Multivariate Function,” *Mathematical Programming*, Vol. 34, No. 2, pp. 188-200.

Moore, R. E., 1966, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ.

O’Hara, M. A. and Bogy, D. B., 1995, "Robust Design Optimization Techniques for Ultra-Low Flying Sliders," *IEEE Transactions on Magnetics*, Vol. 31, No. 6, Pt. 1, pp. 2955-2957.

O'Hara, M. A.; Yong Hu; Bogy, D.B., 1996, “Effects of Slider Sensitivity Optimization,” *IEEE Transactions on Magnetics*, Vol. 32, No. 5, Pt. 1, pp. 3744-3746.

Ono, K., 1975, “Dynamic Characteristics of Air-Lubricated Slider Bearing for Noncontact Magnetic Recording,” *Transactions of the ASME, Series F, Journal of Lubrication Technology*, Vol. 97, No. 2, pp. 250-60.

Patankar, S. V., 1980, *Numerical Heat Transfer and Fluid Flow*, McGraw-Ridge.

Patankar, S. V., 1981, “A Calculation Procedure for Two-Dimensional Elliptic Situations,” *Numerical Heat Transfer*, Vol. 4, pp. 409-425.

Peng, J.-P. and Hardie, C. E., 1995a, "A Finite Element Scheme for Determining the Shaped Rail Slider Flying Characteristics with Experimental Confirmation," *Journal of Tribology*, Vol. 117, pp. 358-364.

Peng, J.-P. and Hardie, C. E., 1995b, "Characteristics of Air Bearing Suction Force in Magnetic Recording Disks," *submitted to STLE/ASME Tribology Conference*, October, 1995, Orlando, FL.

Peng, J.-P. and Harwood, R. F., 1996, "Modeling of a General Crown Shape and the Effects on Air Bearing Steady-State Performance," IEEE Transactions on Magnetics, Vol. 32, No. 5, pp. 3711-3713.

Queipo, N., Devarakonda, R., Humphrey, J. A. C., 1994, "Genetic Algorithms in Thermosciences Research: Application to the Optimized Cooling of Electronic Components," *International Journal of Heat and Mass Transfer*, vol. 37, number 6, pp. 893-908.

Rinnooy Tan, A. H. G. and Timmer, G. T., 1986, "Global Optimization," Report 8612/A, Econometric Institute, Erasmus University, Rotterdam, The Netherlands.

Romeo, F. and Sangiovanni-Vincentelli, A., 1991, "A Theoretical Framework for Simulated Annealing," *Algorithmica*, Vol. 6, No. 3, pp. 302-345.

Ruiz, O. J. and Bogy, D. B., 1988, "A Comparison of Slider Bearing Simulations Using Different Models," *IEEE Transactions on Magnetics*, Vol. 24, No. 6, pp. 2754-2756.

Sangiovanni-Vincentelli, A., Romeo, F. and Mitra, D., 1985, "Convergence and Finite-Time Behavior of Simulated Annealing," *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, FL, pp. 761-767.

Schraudolph, N. S., 1994, GAUCSD1.4, Computer Science and Engineering Department, UCSD, La Jolla, CA, Grefenstette, John J., Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, D.C.

Shyy, W. and Sun, C. S., 1993, "Development of A Pressure-Correction/Staggered-Grid Based Multi-Grid Solver for Incompressible Recirculating Flows", *Computer and Fluids*, Vol. 22, No.1, pp. 51-76.

Solis, F. J. and Wets, R. J-B., 1981, "Minimization by Random Search Techniques," *Mathematics of Operations Research*, Vol. 6, No. 1, pp. 19-30.

Stanley, H. Mark, 1993, "The Effect of Interfacial Conformity on the Contact, Adhesion, and Sliding Friction of Surfaces with Small Roughness," *Doctoral Dissertation*, Department of Mechanical Engineering, University of California, Berkeley.

Stenberg, H., Wahl, M. H. and Talke, F. E., 1995, "Comparison of Numerical and Experimental Flying Characteristics of Tri-Pad Sliders," *IEEE Transactions on Magnetics*, Vol. 31, No. 6, pp. 2970-2972.

Szu, H. and Hartley, R., 1987, "Fast Simulated Annealing," *Phys. Lett. A*, Vol. 122, pp. 157-162.

Tokuyama, M., Yamaguchi, Y., Takeuchi, Y. and Shimizu, I., 1987, "Reducing Flying Height Degradation Due to Track Access Acceleration Acting on a Magnetic Head Slider," *IEEE Transactions on Magnetics*, Vol. 23, No. 5, pp. 3470-3472.

Utsunomiya, M., Hashimoto, M. and Kajitani, H., 1996, "Development of a Magnetic Head Suspension System for High-Speed Seeking Performance," *NEC Research & Development*, Vol. 37, No. 3, pp. 369-381.

White, J. W., "The Transverse Pressure Contour Slider: Flying Characteristics and Comparisons with Taper-Flat and Cross-Cut Type Sliders," *ASME Advances in Information Storage Systems*, Vol. 3, pp. 1-14.

White, J. W., 1986, "A Uniform Flying Height Rotary Actuated Air Bearing Slider," *IEEE Transactions on Magnetics*, Vol. 22, No. 5, pp. 1028-1030.

White, J. W. and Ponnaganti, V., 1988, "Non-Linear Air Bearing Dynamcs of a Six Degrees of Freedom Magnetic Recording Slider with Head-Disk Contact/Impact," *IEEE Transactions on Magnetics*, Vol. 24, No. 6, 2757-2759.

Yang, Y.-P. and Tu, C.-C., 1996a, "Multiobjective Optimization of Hard Disk Suspension Assemblies: Pt. I--Structure Design and Sensitivity Analysis," *Computers & Structures*, Vol. 59, No. 4, pp. 757-770.

Yang, Y.-P. and Tu, C.-C., 1996b, "Multiobjective Optimization of Hard Disk Suspension Assemblies: Part II--Integrated Structure and Control Design," *Computers & Structures*, Vol. 59, No. 4, pp. 771-782.

Yoneoka, S., Yamada, T., Aurga, K., Doe, T., and Takahashi, M., 1987, "Fast Take-Off Negative Pressure Slider," *IEEE Transactions on Magnetics*, Vol. 23, No. 5, pp. 3464-3466.

Yoneoka, S., Katayama, M., Ohwe, T., Mizoshita, Y. and Yamada, T., 1991, "A Negative Pressure Microhead Slider for Ultralow Spacing with Uniform Flying Height," *IEEE Transactions on Magnetics*, Vol. 27, No. 6, pp. 5085-5087.

Yoon, S.-J. and Choi, D-H, 1995, "Design Optimization of the Taper-Flat Slider Positioned by a Rotary Actuator," *Journal of Tribology*, Vol. 117, pp. 588-593.

Yoon, S.-J. and Choi, D.-H., 1996, “An Optimum Design of the Transverse Pressure Contour Slider for Enhanced Flying Characteristics,” submitted to the ASME/STLE Joint Tribology Conference, SF, CA.

Zhang, S., and Bogy, D. B., 1996, “Slider Designs for Controlling Contamination”, accepted to *1996 ASME/STLE International Tribology Conference & Exposition*.

Zhang, S., and Bogy, D. B., 1996, “Slider Designs for Controlling Particle Contamination”, *CML report*, UC Berkeley.

Zhu, L.-Y. and Bogy, D. B., 1988, “Head-Disk Spacing Fluctuation Due to Disk Topography in Magnetic Recording Hard Disk File,” *CMRR Technical Report*, No. 21, Department of Mechanical Engineering, University of California, Berkeley.

Zhu, L.-Y., 1988, “Measurement and Analysis of the Slider-Disk Spacing in Magnetic Recording Disk Files,” *Doctoral Dissertation*, Department of Mechanical Engineering, University of California, Berkeley.

APPENDIX A

The CML Air Bearing Optimization Program Version 1.5

Matthew A. O'Hara and David. B. Bogy
Computer Mechanics Laboratory
Department of Mechanical Engineering
University of California
Berkeley, CA 94720

Abstract

This report presents a detailed description of the air bearing optimization program developed at the Computer Mechanics Laboratory at the University of California at Berkeley. The code provides the tools necessary for the optimization of air bearing designs. Version 1.5 is an update from the previously released optimization program.

The optimization implemented is a variation of the simulated annealing algorithm. The technique does not make use of numerical derivatives in the search for an optimum, rather it makes use of a neighborhood function. Further, it will achieve a global optimum in the presence of multiple local optima.

The program implements the CML Air Bearing Design Program to evaluate the quality of different designs. Therefore, it is necessary for the design program (version quick300) to be available and working in order for this optimization program to be used.

The method is capable of multi–dimensional optimization with multiple goals being simultaneously satisfied.

1. Introduction

The algorithm implemented to optimize air bearing slider designs is a variation of the Simulated Annealing optimization algorithm. This algorithm is a stochastic optimization technique that is modeled after the physical process of annealing. The algorithm does not make use of function derivative information, either analytical or numerical. Instead, it uses a neighborhood operator to find the global optimum. The algorithm also achieves global optimums in the presence of multiple local optima. This has been shown for a specific case of an air bearing slider[1], for which this technique is particularly well suited.

The optimization problem can be divided into two parts: constraints and cost function. A file, entitled "constraint.dat," enables one to define the geometric and parametric constraints for a problem and to control the weighting of the terms in the cost function. To evaluate the cost function for a given configuration, the optimization program makes use of the CML Air Bearing Design Program[2].

2. Input and output

i. Input

The file *constraint.dat* is the only input file to the program (in addition to the files *rail.dat, steady.def, multcase.dat,* and *tol.dat* which are necessary to run the CML Air Bearing Design Program). All variables that are not set explicitly in the *constraint.dat* file are taken from the *rail.dat* and *steady.def* files. A specific example of such data is the wall profile information. The wall profile is defined in the *rail.dat* file as described in the CML Air Bearing Design Program manual. Essentially, a series of linear segments can be connected to approximate a wall angle and/or edge blend that may result from a manufacturing process. If such a profile is defined, it often terminates at a recessed region of the slider. When the recess depth is varied in the optimization procedure, the wall profile must be modified. The modifi-

cation implemented is a simple scaling of both dimensions with the ratio of the current recess depth to the original recess depth (Fig. 1.).

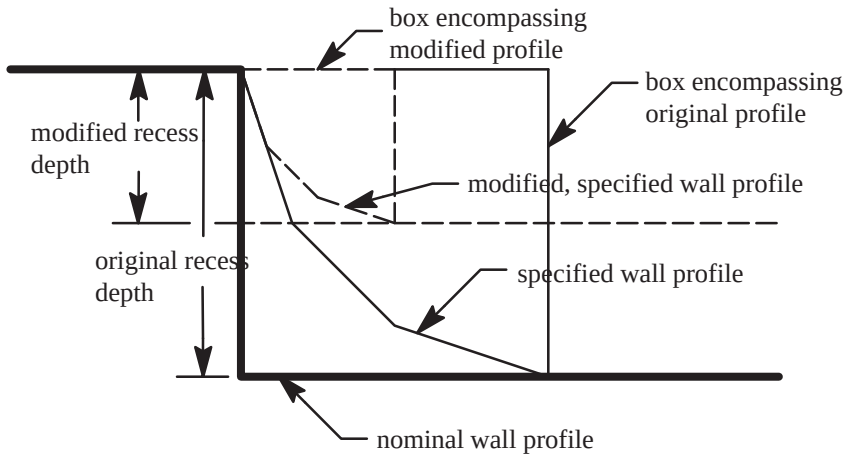

Fig. 1. Modification of wall profile
under recess depth variation.

The first two lines of the *constraint.dat* file describe the format of the six parametric constraints and **should not be edited**. They are:

```
Format for non–geometric constraints:
variable name  lower value  upper value  initial value
```

The following six lines actually define constraints for the problem. If a particular parameter listed in these six lines is not desired for optimization, simply set the upper and lower bounds to be equal. An example of how these lines would be written is:

```
load 2.5 3.5 3.0
x offset 0.4 0.6 0.5
y offset –0.1 0.1 0.0
taper length 0.8e–1 1.2e–1 1.0e–1
taper height 0.5e–6 1.5e–6 1.0e–6
recess depth 4.0e–6 6.0e–6 5.0e–6
```

Note that the taper length (but not the taper height) is given in normalized units, i.e., the desired taper length range should be divided by the length of the slider before being entered in *constraint.dat* which is consistent with the bearing design code.

The next line is a separator line and **should not be edited**.

The following two lines describe how the geometric constraints should be constructed and should also **not be edited**. They are:

```
Format for original geometric constraints:
rail#   vertex#   dir   low val.   up val.   init val.
```

Below these lines is where geometric constraints are defined for the problem. Multiple vertices can be given a range in which to vary. Each constraint takes up one line. The upper limit on the number of geometric constraints is 40. The **rail#** field defines on which rail the vertex resides as defined in the CML Air Bearing Design program. Similarly, the **vertex#** field defines which vertex is to be varied on this rail and is also used as it is defined in the CML Air Bearing Design program. The **dir** field defines in which direction a vertex is to move. The field should read x (or X) if the vertex is to move parallel to the direction of flow or y (Y) if the vertex is to move in the direction transverse to the flow. These conventions are consistent with the CML Air Bearing Design program[2]. The **low val.** field gives the lower value of the geometric constraint. Note that all geometric constraints are given in normalized coordinates, i.e., all physical coordinates are divided by the slider length with the coordinate axes at the inner leading edge. So, the trailing edge would have an x coordinate value of 1, every point not at the back edge would have x coordinate values less than 1, and the outer edge would typically have a y coordinate that is less than 1 due to the rectangular shape of most sliders. The next field, **up val.**, gives the upper limit on the value of the constraint. The following field, **initial val.**, gives the values that are used for the starting point of the optimization. Note that if a parameter range has lower and upper bounds that are equal, i.e., this parameter is not to be included in the optimization, then the **initial val.** field will be ignored and the value read in from the *rail.dat* that already exists in the current directory will be used.

The next two lines describe how the symmetric constraints should be constructed and **should not be edited**. They are:

```
Format for symmetric constraints
rail#   vertex#   dir   to be symmetric with:  rail#  vertex#
```

Below these lines is where symmetric constraints are defined for the problem. Each line consists of five fields. The first two fields, **rail#** and **vertex#**, indicate which rail and vertex will be varied symmetrically. The **dir** field defines in which direction the specified vertex should vary. The final two fields, (to be symmetric with:) **rail#** and **vertex#**, define the vertex that the current vertex will vary symmetrically with.

The next two lines describe how the relative constraints should be constructed and **should not be edited**. They are:

```
Format for relative constraints
rail#   vertex#   to be moved relative to:  rail#   vertex#
```

Below these lines is where relative constraints are defined for the problem. A relative constraint fixes a specified vertex to move with the same relative distance to another specified vertex throughout the optimization. Each line consists of four fields. The first two fields, **rail#** and **vertex#**, define which vertex will be moving relatively. The second two fields, (to be moved relative to:) **rail#** and **vertex#**, define the vertex to move relative to.

The next section defines constraints on recess values for individual rails. The first three lines consist of a separator and two description lines and **should not be edited**. They are:

```
************************************************************
Format for rail height constraints
rail#       low val. (m)      up val. (m)       init val. (m)
```

The first field, **rail#**, lists which rail will have its recess value varied in the optimization. The next three fields, **low val.**, **up val.** and **init val.**, define the lower and upper bound on the constraint and the initial value used to start the optimization, respectively. Directly following this is the relative rail height section. The first two lines are description lines and **should not be edited**. They are:

```
Format for relative rail height constraints
rail#       rail# to be moved relative to
```

The two fields, **rail#** and **rail# to be moved relative to**, define which rail will be varying relatively and which rail will be the reference.

Typically, the optimization program evaluates slider fly heights, roll values, etc. at different places on the disk. Where, and how many of, these evaluations are made are described in this section. Two fields are needed to define exactly where the slider is to be evaluated for purposes of the cost function. The first field, **radius**, determines the radial distance from the center of the disk and the next field, **skew**, determines the corresponding skew. An example of how this section might look:

```
************************************************************
Format for evaluation points (from inner radius out):
radius(meters)          skew(degrees)
0.015       1.22
0.019       -9.1
0.031       -17.39
```

The next section is the altitude sensitivity section. The first two lines are a separator and direction line and **should not be edited**. They are:

```
************************************************************
Altitudes in meters (one entry per line) (STP is always evaluated)
```

If no altitude sensitivity is to be performed, **do not enter any value**. Only enter an altitude value if altitude sensitivity is one of the terms to be included in the objective function. An evaluation is always made at the values provided in the *steady.def* file, therefore, this altitude does not need to be entered.

The remaining section determines the weightings of the objective function. The first two lines of this section are a separator line and a description of the section and **should not be edited**.

```
************************************************************
Weightings for objective function:
```

The following line is an informational line and **should not be edited**. The following line is the weighting for the **maximum difference** term in the objective function. An example of these two lines is:

```
weighting for maximum difference in fly height (nm):
1.0e0
```

The **maximum difference** term is defined to be the maximum difference in fly height at the first fly height of interest for all the points specified in the evaluation points section, given in nanometers. The first fly height of interest is defined in the *rail.dat* file that exists prior to the beginning of the optimization. It is defined in the CML Air Bearing Design Program manual[2], and it is the last two rows of the first column of the *rail.dat* file. The penultimate row is the normalized x value, e.g., 1.0 corresponds to the trailing edge, and the final row corresponds to the normalized y value. The next two lines in the *constraint.dat* file describe and define the weighting for the **difference between average and target fly height**, which is defined as the absolute value of the difference between the average of the first fly heights of interest at the evaluation points and the **target fly height** that is described and defined in the following two lines of the *constraint.dat* file. An example of these lines is:

```
weighting for difference between average and target fly height(nm):
1.0e0
target fly height (nm):
14.0e0
```

This term is also given in nanometers. The following two lines describe and define the weighting for the **sum of |roll values|** term in the objective function. An example of these lines is:

```
weighting for sum of |roll values (urad)|:
1.0e0
```

At each evaluation point, the roll value is determined. This term consists of the sum of the absolute values of these roll values given in μradians. The next four lines define the **pitch – pitch cutoff** term in the objective function. This term has the form of a penalty function.

The term takes on no value until the pitch cutoff, given in μradians, is reached at any evaluation point. Beyond this, the term gains value for every μradian, multiplied by the weighting. An example of these lines is:

```
weighting for sum of (pitch – pitch cutoff) (urad):
1.0e0
pitch cutoff (urad):
175
```

This is graphically depicted in Fig. 2.

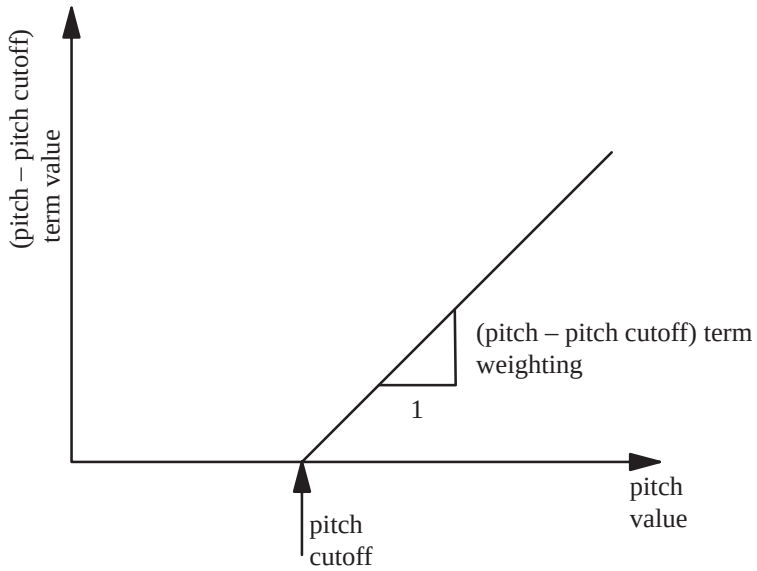

Fig. 2. Graphical depiction of pitch term in objective function

The next weighting is for the **sum of vertical sensitivities** term. An example of the lines that define this term is:

```
weighting for sum of vertical sens. sum(|eval. pt. value – tar-
get(nm/g)|):
1.0
vertical sensitivity target (nm/g):
10.0
```

This term is the sum of the inverse of the vertical stiffness of the air bearing at each of the evaluation points minus the target value specified. The next weighting is for the **sum of pitch sensitivities** term. An example of the lines that define this term is:

```
weight for sum of pitch sensitivities:
1.0
pitch sensitivity target urad/(g–mm):
10.0
```

This term is the sum of the inverse of the pitch stiffness of the air bearing at each of the evaluation points minus the target value specified. The last sensitivity term is the **sum of roll sensitivities** term. An example of the lines that define this term is:

```
weight for sum of roll sensitivities:
1.0
roll sensitivity target urad/(g–mm):
10.0
```

This term is the sum of the inverse of the roll stiffness of the air bearing at each of the evaluation points minus the target value specified.

An entire stiffness matrix is calculated by the CML Air Bearing Design Program (if desired) by comparing the load, the x moment (pitch), and the y moment (roll) against the fly height, pitch and roll of the slider. The reason the stiffness is used is that the entire problem is a minimization problem and by casting the stiffness as sensitivity, it can be added directly to the objective function. If any of the sensitivity weighting is non–zero, the stiffness calculation will be carried out. If sensitivity optimization is not desired in an optimization, these terms should all be set to zero so that the bearing design program will not evaluate the stiffness matrix.

The next section defines the weighting and target for the **sum of sub–ambient pressure** term.

```
weighting for sum of sub–ambient pressures term sum(|neg. press – tar-
get(g)|):
1.0
sub–ambient pressure target (g) (note: give as absolute value)
3.0
```

The target for this term specifies, in grams, what value of integrated sub–ambient pressure is desired at each evaluation point. The motivation for including this term is the reduction of cavity contamination buildup.

The final weighting is for the **altitude sensitivity** term. An example of the lines that define this term are:

```
Altitude insensitivity term weighting (nm)
1.0
```

This term represents the sum of maximum difference in fly height at each evaluation point due to changes in altitude, i.e., if more than two altitudes are specified in the altitude definition section then the fly heights at all altitudes for a given evaluation point are compared to find the highest and lowest values.

ii. Output

Six output files are written by the program. The file that tracks the progress of the optimization is entitled *opti_res*. This file writes the current annealing temperature, for both the parameters and the cost, the number of configurations generated so far, the number of configurations accepted and the current optimal value of all the parameters used in the optimization. Because it is inconvenient to write these values into the appropriate file (either rail.dat and/or steady.def), the current optimal parameters are written into the files *rail.dat.opt* and *steady.def.opt* every time a new optimum is found. So, if the program is terminated prematurely, these files will be available for use in the simulator code. The remaining three files contain information about the code that simplify plotting progress of the optimization. The *cost.dat* file contains information about every feasible configuration that is generated by the optimization program. The *cost.dat* file has 13 fields. The fields are defined as follows:

1. total cost
2. difference in fly height term
3. difference from target term
4. roll term
5. pitch term
6. vertical sensitivity term

7. pitch sensitivity term
8. roll sensitivity term
9. sub–ambient pressure term
10. altitude sensitivity term
11. cost temperature
12. parameter temperature
13. number of feasible designs generated

The files *nacost.dat* and *optcost.dat* have one additional field:

14. number of accepted designs

The file *nacost.dat* keeps a record of all accepted configurations and *optcost.dat* only keeps track of optimal configurations, i.e., it is only updated when a configuration is found that improves on the current best configuration.

Section 3. How to run the program

i. How to compile

Create a new directory that will hold the simulator program. Copy all the files in the distribution to this directory. Usually this will be done by untarring a tar file. The distribution file is entitled: optiv1_5.tar. The command to untar it would be:

tar –xvf optiv1_5.tar

Most UNIX machines come with the “make” utility. If this is the case on your machine, simply type “make” at the UNIX prompt. If not, then you need to enter a command that will compile all of the files. The following is an example:

```
cc –O –o opti anneal.c cost.c cost_status.c config_status.c init_con-
fig.c myrandom.c newconfig.c opti.c readconstraint.c readrails.c read-
steady.c rel_distances.c results.c write_rail.c write_steady.c  –lm
```

This will create the executable file *opti*.

ii. *quick300*

In order for the optimization program to work properly, the CML Air Bearing Design program needs to be running, specifically, the version *quick300*. All the files that are necessary to run *quick300* are necessary for the optimization program as well, specifically, *rail.dat, steady.def, tol.dat, multcase.dat*. The *tol.dat* and *multcase.dat* files should be formatted so that the simulation program will be run only once.

iii. Setting up a problem and running the program

The simplest way to keep track of different optimizations is to create a different subdirectory for each case. Make sure that *quick300* and *opti* are in the search path of the computer so that they can be run from any directory. Copy the files necessary to run *quick300* into the newly created directory and copy a skeleton of the *constraint.dat* file from the directory where the optimization program was created. At this point, the easiest way to create the

constraints is to print out a copy of the rail design from the static design code and draw the constraints. Using this, fill out the *constraint.dat* file including appropriate values for the parametric constraint section, the geometric constraint section, the evaluation points section and the weighting section. At this point, the program is ready to run. To run simply type `opti` at the UNIX prompt.

Section 4. Example problem

This section uses an example problem to display all of the main features of the CML Air Bearing Optimization Program. To complete the first section of *constraint.dat* file, a decision has to be made regarding which parametric constraints to include in the optimization. The example bearing chosen is a 50% slider, i.e., 2.00mm x 1.6mm, with a fixed load of 3 grams. This implies that the upper val. and lower val. of the load constraint line will be equal. The position of the loading force on the slider will also be taken as fixed in the “x” and “y” direction. The taper length is fixed but the taper height will be allowed to vary from 2.0 to 3.0 μm, which implies a range of taper angles between 10.0 and 15.0 mrad. The recess depth will be allowed to vary from a minimum of 4.0 μm, to a maximum of 8.0 μm. Based on this information, the first part of the *constraint.dat* file is:

```
Format for non-geometric constraints:
variable name  lower value  upper value  initial value
load(kg)  3.0e-3 3.0e-3 3.0e-3
x offset 0.5 0.5 0.5
y offset 0.0 0.0 0.0
taper length 1.0e-1 1.0e-1 1.0e-1
taper height 2.0e-6 3.0e-6 2.0e-6
recess depth 4.0e-6 8.0e-6 6.0e-6
```

Now, the geometry of the bearing is considered. Take the following air bearing as the nominal design (Fig. 3.). Note that the air bearing has only symmetric constraints and that it has two independent constraints applied to the same vertex. The first constraints to be considered will be the only two–dimensional original constraint. The combination of these two constraints creates a box about the vertex. The vertices that are directly adjacent to this ver-

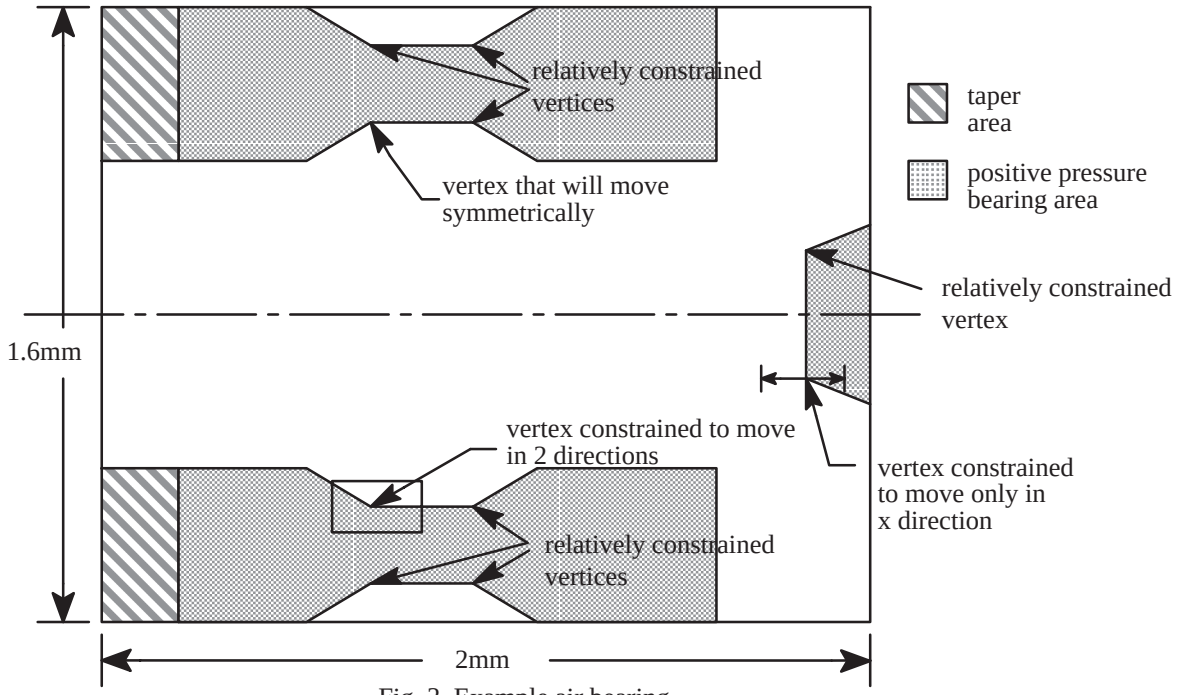

Fig. 3. Example air bearing

tex and form a rectangle with it will be specified as relative constraints, i.e., they will maintain the same relationship throughout the course of the optimization. The vertex that symetrically opposes the first constraint will be specified as a symmetric constraint. The symmetry will be maintained in both the x and y direction (corresponding to the direction of flow and transverse to the direction of flow). The vertices that surround this vertex (symmetric to the vertices described above) will be specified as relative constraints. It is important that these be specified as relative constraints and not symmetric constraints because the order of updating for vertices is:

i. original constraints,

ii. symmetric constraints,

iii. relative constraints.

Therefore, if the vertices on the rail with the symmetric constraint were also specified as symmetric, they would not be updated. This is because the originally constrained vertex would be updated first, then the symmetric constraints would be updated which would not

move these vertices since the relative constraints on the rail with the original constraint have not yet been updated.

The second section of the *constraint.dat* file will have the following format based on the numbering given in Fig. 3. and Fig. 4.

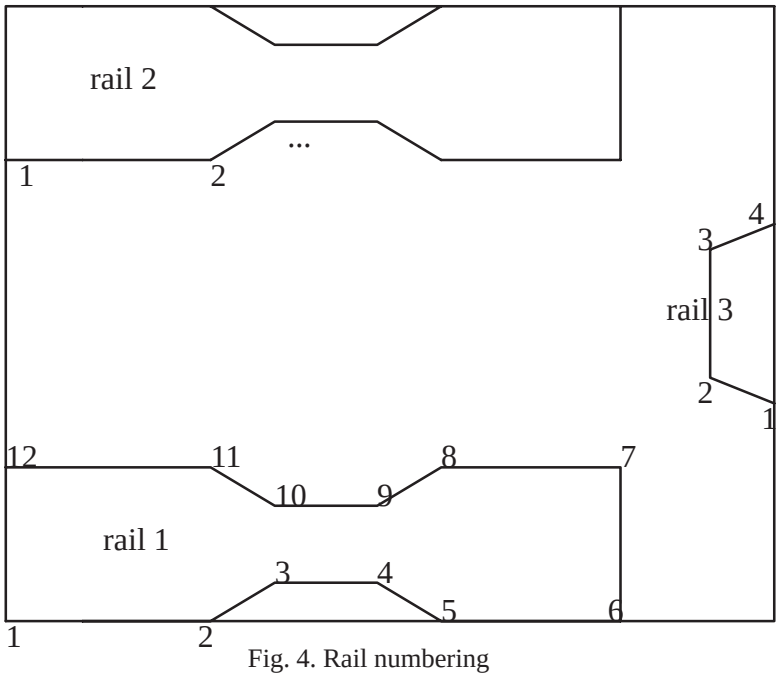

Fig. 4. Rail numbering

```
************************************************************
Format for original geometric constraints:
rail#   vertex#   dir   low val.   up val.   init val.
1       10         x     0.3        0.4       0.35
1       10         y     0.15       0.2       0.175
3       2          x     0.85       0.9       0.9
Format for symmetric constraints
rail#   vertex#   dir   to be symmetric with:  rail#  vertex#
2       3         x                            1      10
2       3         y                            1      10
Format for relative constraints
rail#   vertex#   to be moved relative to:  rail#   vertex#
1       3                                  1       10
1       4                                  1       10
1       9                                  1       10
2       4                                  2       3
2       9                                  2       3
2       10                                 2       3
3       3                                  3       2
```

Note that all of these values are in normalized coordinates.

The third section of the *constraint.dat* file specifies constraints on variation of individual rail heights. The following will be used in this example:

```
************************************************************
Format for rail height constraints
rail#       low val. (m)   up val. (m)     init val. (m)
Format for relative rail height constraints
rail#       rail# to be moved relative to
```

This means that none of the rails will vary in height during the optimization.

The fourth section of the *constraint.dat* file is controlled by the radius of curvature of the suspension and actuator arm and the zero skew point. For this example, three evaluation points with the following values will be used:

```
************************************************************
Format for evaluation points (from inner radius out):
radius(meters)   skew(degrees)
0.015            1.0
0.019            -8.0
0.031            -15.0
```

Note that the radii values are given in meters and that the the positive skew values occur when the slider is at the inner radius. (This skew definition is currently contrary to the IDEMA standard.)

The next section of the *constraint.dat* file deals with the altitude sensitivity term. This optimization will not include the altitude sensitivity term, therefore, no altitudes will be specified in the altitude section.

```
************************************************************
Altitudes in meters (one entry per line) (STP is always evaluated)
```

The final section of the *constraint.dat* file will determine the composition of the objective function. Choosing the default of 1.0 for the maximum difference weighting is a good initial value. The remaining weights can be normalized to this. The initial weighting for the difference from target fly height term should also be 1.0. The target fly height will be taken to be 50.0nm. Note that in the *constraint.dat* file, the target fly height should be entered in nanometers. In this example, the roll minimization term will be included. A weighting of

1.0 is a reasonable initial value so that it will not predominate the calculation. A value of zero will be used for the pitch term. This corresponds to the assumption that the pitch does not have a direct impact on the air bearing's performance. Although this may not necessarily be the case, we will still neglect it in this instance. The weighting for the stiffness terms will each be set to 1.0. Frequently, these terms can predominate a calculation when the fly height terms (maximum difference and difference from target) approach their minimum values of zero. In that instance, the weighting for non–sensitivity terms should be adjusted upwardly or a higher target value should be used. The problem lies in the fact that, feasibly, other terms can go to zero, but clearly the sensitivity can never go to zero. The sub–ambient pressure and altitude sensitivity terms will not be employed in this example.

Using the above weightings, the final section would be:

```
************************************************************
Weightings for objective function:
weighting for maximum difference in fly height (nm):
1.0e0
weighting for difference between average and target fly height(nm):
1.0e0
target fly height (nm):
50.0e0
weighting for sum of |roll values (urad)|:
1.0e0
weighting for sum of (pitch – pitch cutoff) (urad):
0.0e0
pitch cutoff (urad):
175
weighting for sum of vertical sens. sum(|eval. pt. value – tar-
get(nm/g)|):
1.0
vertical sensitivity target (nm/g):
10.0
weight for sum of pitch sensitivities:
1.0
pitch sensitivity target urad/(g–mm):
10.0
weight for sum of roll sensitivities:
1.0
roll sensitivity target urad/(g–mm):
10.0
weighting for sum of sub–ambient pressures term sum(|neg. press – tar-
get(g)|):
0.0
sub–ambient pressure target (g) (note: give as absolute value)
3.0
Altitude sensitivity term weighting (nm)
0.0
```

Section 5. Hints For Successful Optimization

The example problem illustrates one of the subtleties of optimization—choosing the weighting for the terms in the objective function properly. A general rule is to weight the items of greatest concern most heavily. While this may seem obvious, it should be executed with some care. For example, if the most important characteristics of a specific design are that it fly completely flat over the radius of the disk and at the target fly height, then those terms need to be weighted roughly five times as heavily as the other so that in the course of the optimization, the other terms, specifically the sensitivity terms, will not be minimized at the

expense of these terms. This is a difficulty that is inherent in multi–objective optimization and cannot be avoided.

Also be aware that certain configurations that are created by the Optimization Program (within the constraints defined by the user) may not fly. If it turns out that an inordinate number of configurations are not flying, then the limits of the constraints should be narrowed. Typically, this helps to limit the number of configurations that will not fly from being created.

By summarizing the results of the optimization, we can gain further insight into the simulated annealing process.

Figure 5 shows that the value of the objective function starts to level off near the conclusion of the optimization. This figure is a plot of the value of the objective function versus the number of feasible designs generated. There are three lines in this plot. The line that bounds the other two from above is the objective function value of each design that is created. The abscissa of this plot is the number of feasible configurations generated. The line that bounds the other two from below is the current optimum value. As the optimization procedure progresses, the decrease of the objective function can be tracked via this line. The final line is a plot of currently accepted configurations. This is a feature of the simulated annealing algorithm. Notice that the accepted configuration plot actually goes up over the course of the optimization. This allows the algorithm to overcome locally optimal solutions in the search for the global optimum.

We can conclude from this two graph that a nearly optimal value is achieved, normally, when the objective function starts to level off.

Finally, a practical consideration to keep in mind is that when the optimization is performed, the CML Air Bearing Design Program is run multiple times. Each time it is run, it creates a stream of output. If you run the optimization program as a batch job, be aware that if you

147redirect the output of the program to a file, the file created may become extremely large. Two different solutions to this problem are

i. run the job in an xterm or otherwise “detachable” window, or,

ii. run in batch mode but redirect the output to /dev/null.

The first case will allow you to monitor the progress of the optimization more closely, e.g., if you are having problems with configurations that are repeatedly crashing, and the second will dispose of all the output information. Note that it is not necessary to monitor the progress of the optimization normally because all useful information will be written into output files.

The example case that was run for this manual used the command

```
opti > /dev/null &
```

where `opti` is the name of the optimization program and `“> /dev/null”` is the redirection command to dispose of all output.

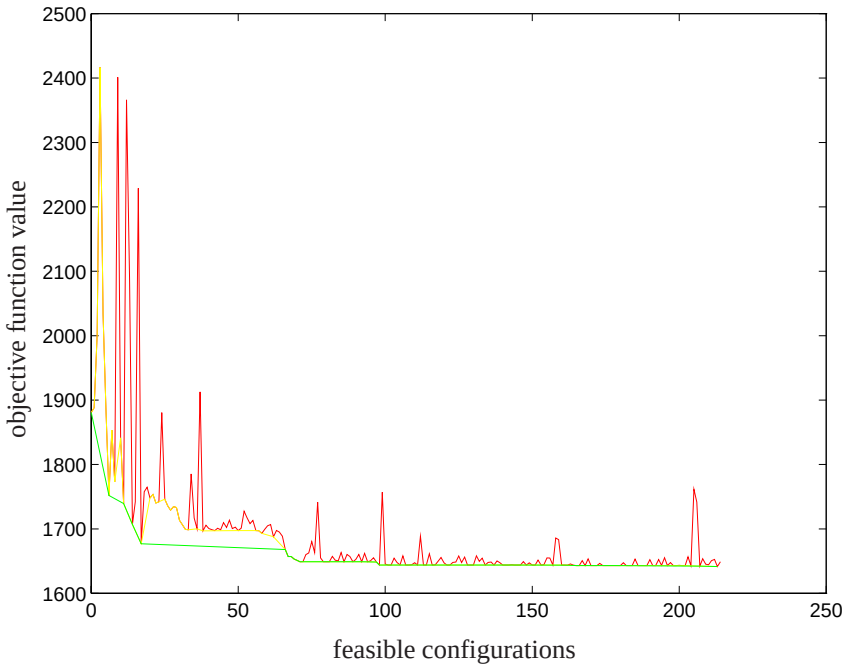

Fig. 5. Objective function vs. current number of feasible configurations

[1]M. A. O'Hara, *Global Optimization of Slider Designs*, Master's Report, UC Berkeley, December, 1994

[2]S. Lu, and D. B. Bogy, "CML Air Bearing Design Program User's Manual," **Technical Report No. 95–003**, Computer Mechanics Laboratory, U.C. Berkeley

APPENDIX B

Example listing of *constraint.dat* file

```
Format for non–geometric constraints:
variable name  lower value  upper value  initial value
load(kg)  3.0e–3 3.0e–3 3.0e–3
x offset 0.5 0.5 0.5
y offset 0.0 0.0 0.0
taper length 1.0e–1 1.0e–1 1.0e–1
taper height 2.0e–6 3.0e–6 2.0e–6
recess depth 4.0e–6 8.0e–6 6.0e–6
************************************************************
Format for original geometric constraints:
rail#   vertex#   dir   low val.   up val.   init val.
1       10         x     0.3        0.4       0.35
1       10         y     0.15        0.2      0.175
3       2          x     0.85       0.90      0.9
Format for symmetric constraints
rail#   vertex#   dir   to be symmetric with:  rail#  vertex#
2       3         x                            1      10
2       3         y                            1      10
Format for relative constraints
rail#   vertex#   to be moved relative to:  rail#   vertex#
1       3                                  1       10
1       4                                  1       10
1       9                                  1       10
2       4                                  2       3
2       9                                  2       3
2       10                                 2       3
3       3                                  3       2
************************************************************
Format for rail height constraints
rail#       low val. (m)            up val. (m)
init val. (m)
Format for relative rail height constraints
rail#       rail# to be moved relative to
************************************************************
Format for evaluation points (from inner radius out):
radius(meters)   skew(degrees)
0.015            1.0
0.019            –8.0
0.031            –15.0
************************************************************
Altitudes in meters (one entry per line) (STP is always evaluated)
************************************************************
Weightings for objective function:
weighting for maximum difference in fly height (nm):
1.0e0
weighting for difference between average and target fly height(nm):
1.0e0
target fly height (nm):
50.0e0
weighting for sum of |roll values (urad)|:
1.0e0
```

```
weighting for sum of (pitch – pitch cutoff) (urad):
0.0e0
pitch cutoff (urad):
175
weighting for sum of vertical sens. sum(|eval. pt. value – tar-
get(nm/g)|):
1.0
vertical sensitivity target (nm/g):
10.0
weight for sum of pitch sensitivities:
1.0
pitch sensitivity target urad/(g–mm):
10.0
weight for sum of roll sensitivities:
1.0
roll sensitivity target urad/(g–mm):
10.0
weighting for sum of sub–ambient pressures term sum(|neg. press – tar-
get(g)|):
0.0
sub–ambient pressure target (g) (note: give as absolute value)
3.0
Altitude sensitivity term weighting (nm)
0.0
```

APPENDIX C

Example listing of original *rail.dat* file

```
   3     1
  12   1   2
  0.0000e+00  0.0000e+00
  2.5000e-01  0.0000e+00
  3.5000e-01  2.5000e-02
  5.0000e-01  2.5000e-02
  6.0000e-01  0.0000e+00
  8.0000e-01  0.0000e+00
  8.0000e-01  2.0000e-01
  6.0000e-01  2.0000e-01
  5.0000e-01  1.7500e-01
  3.5000e-01  1.7500e-01
  2.5000e-01  2.0000e-01
  0.0000e+00  2.0000e-01
  0.0000e+00  3.0000e-03
  0.0000e+00  6.0000e-06
  12   1   2
  0.0000e+00  6.0000e-01
  2.5000e-01  6.0000e-01
  3.5000e-01  6.2500e-01
  5.0000e-01  6.2500e-01
  6.0000e-01  6.0000e-01
  8.0000e-01  6.0000e-01
  8.0000e-01  8.0000e-01
  6.0000e-01  8.0000e-01
  5.0000e-01  7.7500e-01
  3.5000e-01  7.7500e-01
  2.5000e-01  8.0000e-01
  0.0000e+00  8.0000e-01
  0.0000e+00  3.0000e-03
  0.0000e+00  6.0000e-06
  4   1   2
  1.0000e+00  2.8000e-01
  9.0000e-01  3.2000e-01
  9.0000e-01  4.8000e-01
  1.0000e+00  5.2000e-01
  0.0000e+00  3.0000e-03
  0.0000e+00  6.0000e-06
 6.000000e-06  0.000000e+00
1.000000e-01 2.000000e-06
1.000000e-08 0.000000e+00 0.000000e+00
1.000000e+00 9.873000e-01 9.746000e-01 0.000000e+00
4.000000e-01 4.000000e-01 2.000000e-01 0.000000e+00
```

APPENDIX D

Example listing of original *steady.def* file

```
************Problem Definition Menu************
hm         h0(/hm)      hs           xl           yl(/xl)
  1.0000e-07   2.0000e+01   0.0000e+00   2.0000e-03   8.0000e-01
disk     ske         ra          rpm      ityact
  1.0000e+10   1.0000e+00   1.5000e-02   5.4000e+03    0
dact       vact          yact          xact             isolv
  0.0000e+00   0.0000e+00   0.0000e+00   0.0000e+00          1
f0         xf0           yf0           xfs(g-mm)    yfs(g-mm)
  3.0000e-03   5.0000e-01   0.0000e+00    0.0000e+00    0.0000e+00
*************Solution Control Menu************
istiff   akmax      emax          p0          al
  0    1.000000e-07 1.000000e-03 1.013500e+05 6.350000e-08
**************Grid Control Menu **************
iadpt  isymmetry ioldgrid   nx   ny   nsx   nsy   nest
      1       0       0    100    100       1       1      4
xnt(i),i=2,nsx
 0.000000e+00
nxt(i),i=2,nsx
 0.000000e+00
dxr(i),i=1,nsx
 1.000000e+00
ynt(i),i=2,nsy
 0.000000e+00
nyt(i),i=2,nsy
 0.000000e+00
dyr(i),i=1,nsy
 1.000000e+00
************** Non-Newtonian Menu *****************
incom   icnst      vis1      vis2      cindex
  0   0   1.8060e-05   0.0000e+00   1.0000e+00
clambda     gammac     sigma      devol      beta
 1.0000000e-07  9.9999997e-06   138000.0      2.0000000e-02   1000.000

***************** Expert Menu *****************
ievl ipress  itnon idisc
   0    0    1    1
 iqpo    difmax      decay      ipmax
   5   40.000   40.000   0
***************** End of Input Data************
```