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

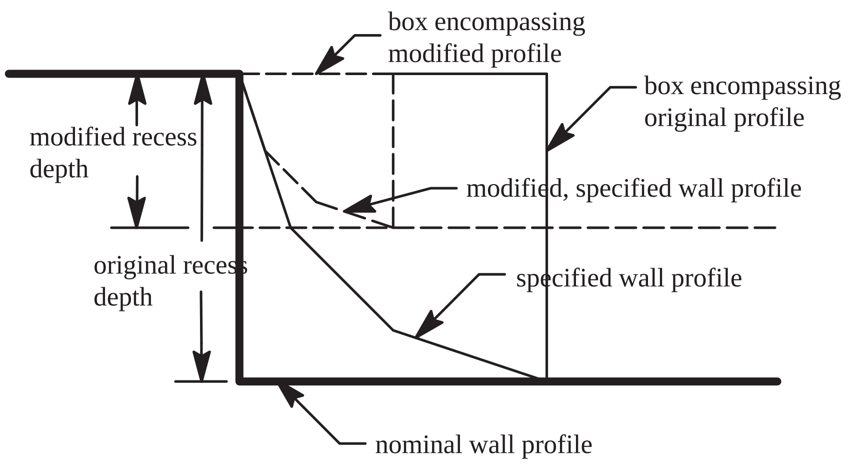
box encompassing modified profile
box encompassing original profile
modified recess depth
modified, specified wall profile
original recess depth
specified wall profile
nominal wall profile

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.

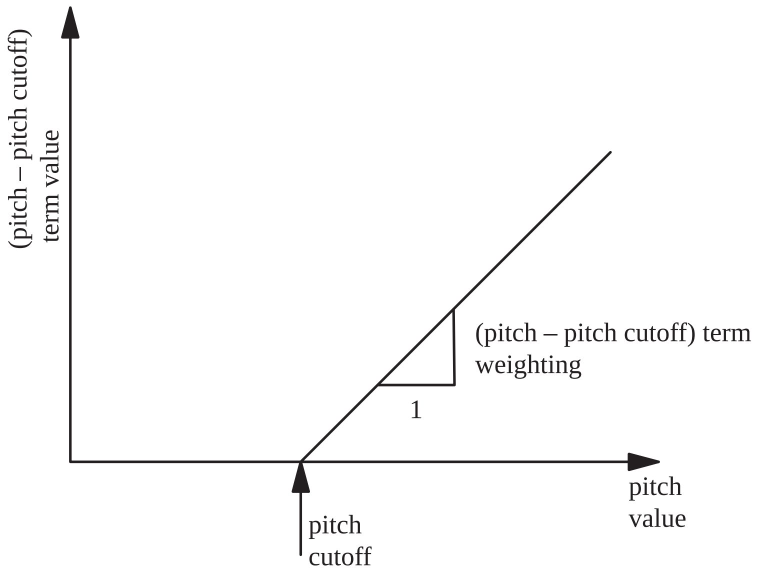

Fig. 2. Graphical depiction of pitch term in objective function

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 – target(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_config.c
myrandom.c  newconfig.c  opti.c  readconstraint.c  readrails.c  readsteady.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 inde-

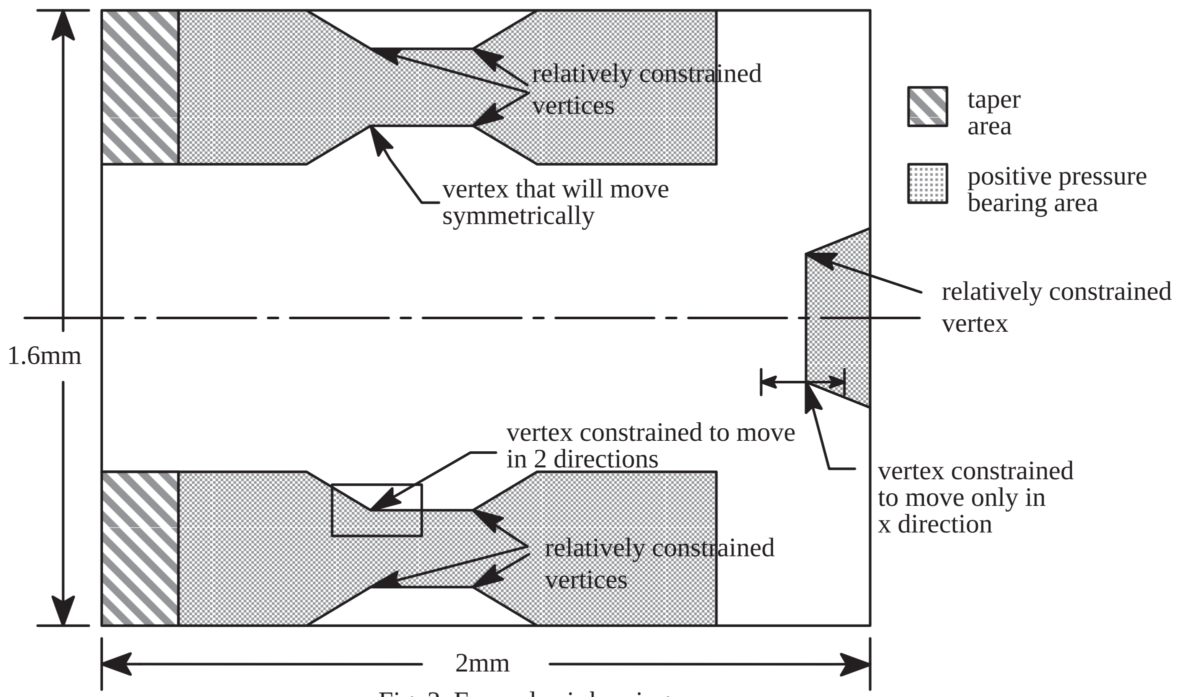
relatively constrained vertices
vertex that will move symmetrically
taper area
positive pressure bearing area
relatively constrained vertex
1.6mm
vertex constrained to move in 2 directions
vertex constrained to move only in x direction
relatively constrained vertices
2mm

Fig. 3. Example air bearing

pendent 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 vertex 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.

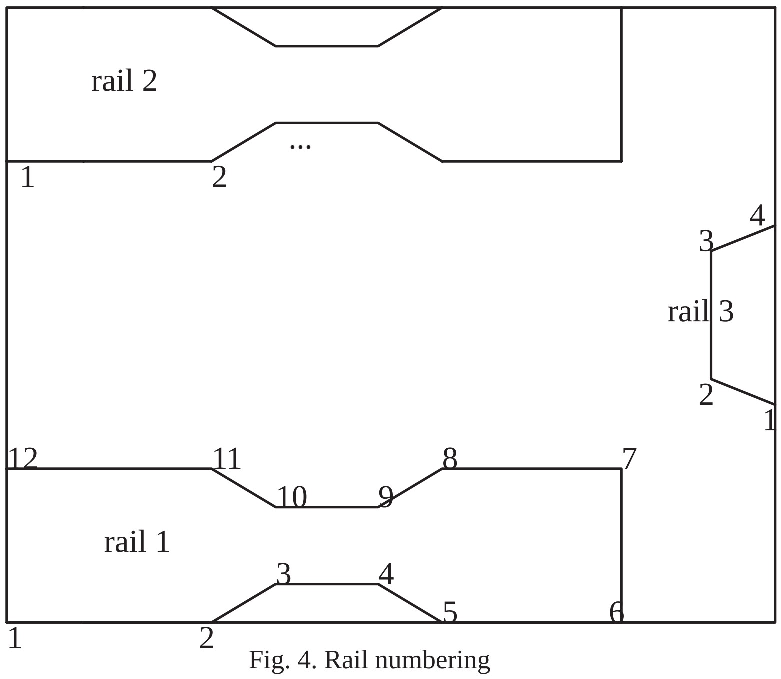

Fig. 4. Rail numbering

```
*************************************************************
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 – target(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 redirect 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.

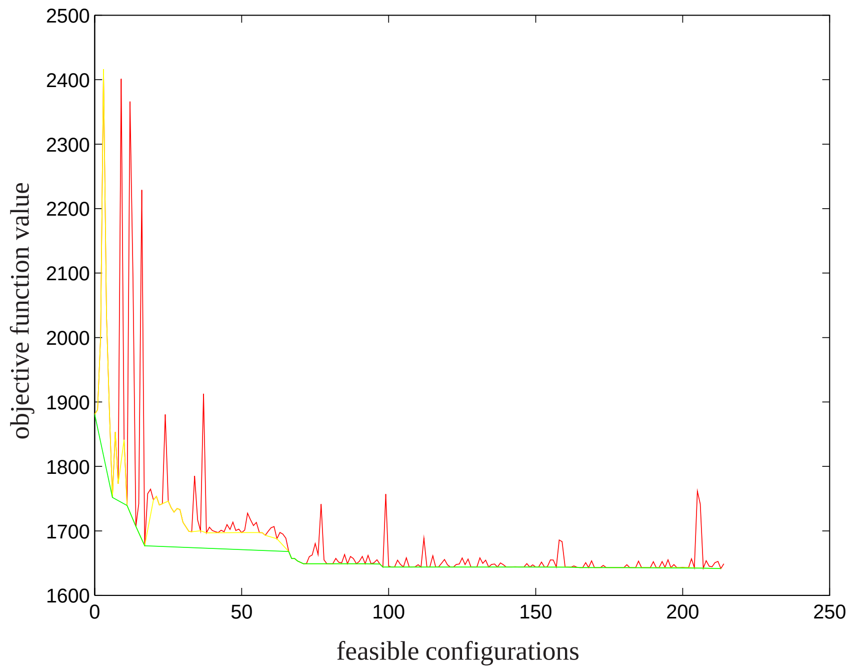

Fig. 5. Objective function vs. current number of feasible configurations

[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 A: 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 – target(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 B: 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 C: 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************
```