# The CML Air Bearing Optimization Program Version 2.0

**Hong Zhu and David B. Bogy**

Computer Mechanics Laboratory

Department of Mechanical Engineering

University of California at Berkeley

Berkeley, CA 94720

## ABSTRACT

This report presents a detailed description of the air bearing optimization program version 2.0 developed at the Computer Mechanics Laboratory at the University of California at Berkeley. The program provides the tools necessary for the optimization of air bearing designs for hard disk drives. Version 2.0 was developed from the previously released optimization program Version 1.5.

Unlike the old version, which has only one kind of simulated annealing algorithm, this new version includes the more powerful Adaptive Simulated Annealing algorithm as well as other variations in that family. Also, new solvers have been implemented to make the code more flexible and more efficient than its previous version. Generally speaking, the

efficiency of the new version of the optimization code is 3 to 4 times higher than the old version.

The program implements the CML Air Bearing Steady Codes to evaluate the designs generated during the optimization process. Therefore, the CML steady codes (rectangular mesh solver Quick419 and triangular mesh solver Quick5) must be available in order for this optimization program to be used.

This optimization program is capable of multi-objective optimization with an arbitrary number of constraints, and can find the optimal ABS design in the search space.

## ***1. INTRODUCTION***

Optimization is the process of trying to minimize a function subject to conditions on the variables. This function is generally referred to as the “objective” or “cost” function. The conditions set on the variables are referred to as constraints.

Figure 1 illustrates the optimization technique types according to the search space, search method and search scale, respectively.

The optimization algorithm implemented here is the Simulated Annealing optimization algorithm. It is an analogy of the physical annealing process wherein a system’s temperature is gradually cooled down in order to reach its lowest energy state. It can be used to find a global minimum of an objective function in parameter space. It is a stochastic global optimization technique for constrained optimization problems. The simulated annealing algorithm has been widely used in many areas such as chemistry, biology, economics, and circuit design, and it has been proven to be a very successful and powerful technique for solving non-linear multi-objective optimization problems.

In this program, we implemented the latest addition to the simulated annealing family, i.e. the Adaptive Simulated Annealing (ASA) algorithm, along with the Standard Simulated Annealing (SA) algorithm, the Fast Simulated Annealing (FA) algorithm, and a self-defined Simulated Annealing algorithm, which was the only algorithm used by the previous version 1.5.

ASA has a much faster cooling schedule and a self-adaptive mechanism to reflect the differing sensitivities of different constraint points. Therefore, it can converge much faster than other simulated annealing algorithms. Also it can effectively avoid the final phase "freezing" phenomenon, which appears in other simulated annealing algorithms. The technical details of these algorithms can be found in CML technical report no. 00-010.

## *2. STRUCTURE OF THE CML OPTIMIZATION PROGRAM*

To perform the optimization, we need two closely integrated parts: the optimization algorithm, and the solver. The optimization algorithm is used to generate different sample designs, which are then sent to the solver for calculation. From these results, the algorithm will evaluate the quality of the current design and, based on that evaluation, will generate a new design.

Here we used the simulated annealing algorithm and the CML steady solvers, including the CML rectangular mesh solver Quick419 and the CML triangular mesh solver Quick5. Figure 2 presents a schematic illustration of the structure of the optimization program version 2.0.

The flow chart of the CML optimization program version 2.0 is shown in Fig. 3.

## 3. COMPARISON BETWEEN THE CURRENT VERSION 2.0 AND THE OLD VERSION 1.5

There are a number of differences between the old and new versions. For clarity, we list only the main differences in Table 1.

## 4. INPUT FILE

The file *constraint.dat* is the only input file to the program (other than the files *rail.dat* and *run.dat* which are necessary to run the CML Air Bearing Design rectangular mesh solver Quick419, or the files *rail.dat*, *run.dat* and *trigrid.dat* which are necessary to run the CML Air Bearing Design triangular mesh solver Quick5). Before running the optimization program, users should copy the file *rail.dat* to *rail.dat.orig* and *rail.dat.opt* and copy the file *run.dat* to *run.dat.orig* and *run.dat.opt*. This is necessary in order to create an initial reference design, because the optimization program will generate new designs during the running process and overwrite these two files. We always regard the initial design as the first optimized design.

All variables that are not set explicitly in the *constraint.dat* file are taken from *rail.dat* and *run.dat*. Please refer to the CML Air Bearing Design Program manual for a detailed explanation of these two files.

Now let's look at the input file *constraint.dat*.

The first two lines of the *constraint.dat* file describe the optimization program information and the way to report bugs. They should not be edited. They are:

```
CML Optimization Code Version 2.0   CONSTRAINT.DAT
REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU
```

The next two lines define the solver you would like to use and they should not be edited. They are:

```
**************************************************************
Select solver (1=rectangular solver  2=triangular solver)
```

The choice (1 or 2) should be entered in the following line.

The next line defines which optimization algorithm you would like to use and it should not be edited. It is:

```
Select optimization algorithm (1=SA 2=FA 3=ASA 4=Self-defined)
```

The choice (1, 2, 3 or 4) should be entered in the following line. Choice 3 (ASA) is recommended.

The next line defines whether constraints for the solver results should be used and it should not be edited. It is:

```
Set constraints on solver results? (0=No 1=Yes)
```

Setting constraints on the solver will accelerate the optimization process. This is because we usually need to evaluate the ABS designs at different radial positions (e.g., OD, MD and ID). The optimization will invoke the CML steady code to calculate results for these different positions. If for a certain design we find that some important parameters are very bad at a certain radial position (e.g., having a very high Roll), the best course is to skip this design, and no longer calculate it for the rest of the radial positions. Essentially, we think this design is bad and should not be accepted. If we set constraints for the solver results, we can let the optimization program skip a design automatically according to the conditions we set, and probably save some time during the optimization process.

The choice (0 or 1) should be entered in the following line. Choice 1 is recommended.

The next two lines describe the format for solver constraints and they should not be edited. They are:

```
Format for solver constraints:
FH_L(nm) FH_U(nm) Roll_L(urad) Roll_U(urad) Pitch_L(urad) Pitch_U(urad)
```

The FH_L and FH_U represent the lower and the upper limits of the flying height (nm), respectively. Similarly, Roll_L and Roll_U represent the lower and the upper limit of the roll (μrad), respectively, and Pitch_L and Pitch_U represent the lower and the upper limit of the pitch (μrad), respectively.

If we set the constraints for the solver, and if the flying height, roll or pitch falls beyond their range of the lower and the upper limit, then the evaluated design will be considered to be unacceptable and will be skipped.

Note: If the range of these parameters is too tight, then very few promising designs will emerge.

The magnitude of these six parameters should be entered on the following line.

The next line defines which screen output mode is preferred and it <u>should not be edited</u>. It is:

```
Screen display mode (1=verbose 2=concise)
```

Verbose mode means the program will show the entire screen output for both the solver and the algorithm. Concise mode means only the most important information will be output to the screen, including the flying height, roll, and pitch for each radial position of a certain design, and a few important parameters for the algorithms. Below is a typical concise screen output:

```
Point#    1 :      FH=    4.8864  ROLL=    2.7764  PITCH=  275.9209
Point#    2 :      FH=    4.2947  ROLL=    5.6050  PITCH=  233.4297
Point#    3 :      FH=    5.3174  ROLL=   -3.4249  PITCH=  182.3231
Quick419 is DONE!
N_gen: 45      N_acc: 15     N_opt: 2
```

```
Cost: 1.568647e+001        Cost_bsf: 1.389004e+001
T_cost = 7.673284e-001     T_param = 4.486467e-002
```

The first three lines give the flying heights (nm), rolls (urad) and pitches (urad) for the three radial positions (OD, MD, ID). The next line shows the solver used by the program (Quick419 in this case). N_gen, N_acc and N_opt represent the number of generated designs, number of accepted designs and number of optimized designs so far, respectively. Cost represents the value of the objective function (also called the cost function) for the current design. Cost_bsf means the best so far cost function value. T_cost represents the temperature for the cost function and T_param represents the temperature for the parameters (or only the first parameter temperature if the algorithm we choose is ASA).

The choice (1 or 2) should be entered on the following line. Choice 1 is recommended.

The next three lines describe the format of the seven parametric constraints and should not be edited. They are:

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

The following seven lines actually define constraints for the problem. To skip optimizing a particular parameter listed here, simply set the upper and lower bounds to be the same. Here is an example:

```
load(kg)            1.5e-3        1.5e-3        1.5e-3
x offset            0.0           0.0           0.0
y offset            0.0           0.0           0.0
taper  length       0.0           0.0           0.0
taper  angle        0.0           0.0           0.0
recess depth        2.5e-6        2.5e-6        2.5e-6
step   depth        0.3e-6        0.3e-6        0.3e-6
```

Note that x offset, y offset, taper length, recess depth, and step depth are all given in unit meters, consistent with the new CML Air Bearing Design program. Taper angle is given in radians.

The next two lines describe the related parameters of recess depth and step depth optimization and should not be edited. They are:

```
***************************************************************
recess index, step index, mid index, property (1=proportional 2=fixed normal dist.)
```

The recess index, step index and mid index represent the wall profile indices for the base recess (cavity depth), whereas step is the segment between them. When the program modifies the recess and step, the wall profiles associated with them should also be changed accordingly. We can change the wall profiles proportionally or with a fixed normal distance. These two options are shown in Figs. 4 and 5.

The three index numbers and one property number should be given in the following line. Please refer to the *rail.dat* file for the corresponding index numbers.

The next three lines describe how the geometric constraints should be constructed and should not be edited. They are:

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

Here we define the geometric constraints, of which there are three different kinds. Original constraints are mutually independent. Symmetric constraints require some vertices to vary symmetrically with the original constraint points. Relative constraints require some vertices to maintain fixed spatial relationships between the original constraint points and the symmetrical constraint points.

Multiple vertices can be given a range in which to vary. Each constraint takes up one line. 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 in the slider length direction, or y (or Y) if the vertex is to move in the slider width direction. These conventions are consistent with the CML Air Bearing Design program. The **low val.** field gives the lower value of the geometric constraint. Note that unlike the previous version 1.5, now all the geometric constraints are given in absolute coordinates in unit meters, which is consistent with the CML Air Bearing Design program. The next field, **up val.**, gives the upper limit on the value of the constraints. 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 the same lower and upper bounds (e.g., this parameter is not to be included in the optimization), then the **initial val.** field will be ignored and the value read in 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#
```

Here we define the symmetric constraints 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 with which the current vertex will vary symmetrically.

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#
```

Here we define the relative constraints 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.

Figure 6 shows the rail shape and some rail indices of the initial ABS design of the example case. Figure 7 shows its 3-D rail geometry. Figure 8 shows the constraints prescribed for the example case.

The corresponding parts for these constraints defined in the *constraint.dat* file are:

```
*************************************************************
Format for original geometric constraints:
rail#   vertex#   dir   low val.   up val.   init val.
5      1      x      0.70e-3      0.85e-3      0.80e-3
5      2      x      0.94e-3      1.09e-3             1.05e-3
5      8      x      0.72e-3      0.86e-3      0.82e-3
Format for symmetric constraints
rail#   vertex#   dir   to be symmetric with:  rail#  vertex#
10     1      x      5      1
10     2      x      5      2
10     8      x      5      8
Format for relative constraints
rail#   vertex#   to be moved relative to:  rail#   vertex#
3      11     5      1
3      12     5      8
5      3      5      2
5      4      5      2
5      5      5      2
5      6      5      2
5      7      5      2
8      11     10     1
8      12     10     8
10     3      10     2
10     4      10     2
10     5      10     2
10     6      10     2
10     7      10     2
```

The next three lines describe the format for evaluation points and should not be edited. They are:

```
**************************************************************
Format for evaluation points (from OD to ID):
radius(meters)   skew(degrees)
```

Typically, the optimization program evaluates slider flying height, roll, pitch, etc. at different radial positions. 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 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. Please note that we have adopted the IDEMA standard regarding positive and negative skews, which is opposite of what we used in our earlier version. Please refer to the latest CML Air Bearing Design program manual to make sure your input is correct.

The final section dictates the weight given to the various terms 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. They are:

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

The next line is an informational line and should not be edited. The line after that is the weight for the **maximum difference in flying height (nm) term**, the 1st term defined in the objective function. Note that all nine terms in the objective function have been normalized, which means that if they have an initial value, it will be 1; otherwise it will be 0. By normalizing the objective function terms we can more easily define their weightings according to our optimization goals, and we can also readily see improvement in different terms. An example of these two lines is:

```
(1)-----Weight for maximum difference in flying height (nm) term:
        1.0
```

This term is defined as:

$$\frac{\left|Maximum\_FH\_difference(FH)\right|}{\left|Maximum\_FH\_difference(FH_0)\right|},$$

where FH means the flying heights of the current design and $FH_0$ means the flying heights of the initial design (parameters with sub-index 0 will be regarded as the parameters of the initial design). We can see that for the initial design, the value of this term is always 1.

An important note about the definition of the flying height: in the CML Air Bearing Design program, there are several different kinds of flying heights in the result file (e.g., nominal flying height, minimum flying height, etc). The flying height we define here is the transducer flying height, or "actual flying height", which is the clearance between the read-write sensor and the disk surface. In this optimization program, we **always** define the read-write sensor point as our first point of interest in the *rail.dat* file and <u>the program will take the flying height at the first point of interest as the actual flying height</u>. So please make sure you define the read-write sensor point as the first point of interest in your *rail.dat* file.

The next two lines in the *constraint.dat* file describe and define the weight for the **flying height term**, which is the 2nd term of the objective function. Note that, as we just mentioned, “flying height” in this case means the transducer flying height. This term is used to check the uniformity of the flying heights around our target flying height. The target flying height is described and defined in the following two lines of the *constraint.dat* file. An example of these lines is:

```
(2)-----Weight for flying height(nm) term:
        9.0
        Target flying height (nm):
        5.0
```

This term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n}(FH_i - FH_{target})^2}}{\sqrt{\sum_{i=1}^{n}(FH_{0i} - FH_{target})^2}},$$

where $FH_{target}$ represents our target flying height and n is the number of the evaluation points. We can see that for the initial design, the value of this term is always 1.

The next two lines in the *constraint.dat* file describe and define the weight for the **roll (mrad) term**, which is the 3rd term of the objective function. This term represents the flatness of the roll profile. An example of these lines is:

```
(3)-----Weight for roll (urad) term:
        1.0
```

This term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n} Roll_i^2}}{\sqrt{\sum_{i=1}^{n} Roll_{0i}^2}}\ .$$

Again we see that for the initial design, the value of this term is always 1.

The next two lines in the *constraint.dat* file describe and define the weight for the **roll – roll cutoff (mrad) term**, which is the 4th term of the objective function. The roll cutoff (μrad) is described and defined in the following two lines; here is an example:

```
(4)-----Weight for roll - roll cutoff (urad) term:
         1.0
         Roll cutoff (urad):
         5.0
```

This term will define the acceptable range for roll (e.g., from –5μrad to +5μrad). Rolls within this range are all acceptable, although smaller is still better. This roll cutoff term acts like a penalty function. If the rolls are all in the range we defined, this term has the value of 0. Otherwise it won't be 0. The more the rolls deviate from our acceptable range, the greater an effect this term will have.

This term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n} Roll\_cutoff_i^2}}{\sqrt{\sum_{i=1}^{n} Roll\_cutoff_{0i}^2}}\ ,$$

where $Roll\_cutoff_i = \begin{cases} 0 & \text{if} \quad |Roll_i| \le Roll\_cutoff \\ |Roll_i| - Roll\_cutoff & \text{if} \quad |Roll_i| > Roll\_cutoff \end{cases}$ .

For this term, if the $\sqrt{\sum_{i=1}^{n} Roll\_cutoff_{0i}^2}$ is equal to 0, then we define the initial value of this term to be 0. For this case, this term will be defined as $\sqrt{\sum_{i=1}^{n} Roll\_cutoff_i^2}$ to avoid dividing by zero.

The next two lines in the *constraint.dat* file describe and define the weight for the **pitch – pitch cutoff (μrad) term**, which is the 5th term of the objective function (the pitch cutoff (μrad) is itself described and defined in the two lines immediately following).

An example of these two lines is:

```
(5)-----Weight for pitch - pitch cutoff (urad) term:
        1.0
        Pitch cutoff (urad):
        300.0
```

The need for this term is quite similar to that for the roll cutoff term. That is, sometimes we don't want the pitch exceed a certain upper limit. Again we have created a penalty function. This term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n} Pitch\_cutoff_i^2}}{\sqrt{\sum_{i=1}^{n} Pitch\_cutoff_{0i}^2}},$$

where $Pitch\_cutoff_i = \begin{cases} 0 & if \quad Pitch_i \leq Pitch\_cutoff \\ Pitch_i - Pitch\_cutoff & if \quad Pitch_i > Pitch\_cutoff \end{cases}$.

For this term, if the $\sqrt{\sum_{i=1}^{n} Pitch\_cutoff_{0i}^2}$ is equal to 0, then we define the initial value of this term as 0. For this case, this term will be defined as $\sqrt{\sum_{i=1}^{n} Pitch\_cutoff_i^2}$ to avoid dividing by zero.

The next six lines define the weights for the three stiffness terms, i.e. **vertical stiffness (g/nm) term**, **pitch stiffness (mN-m/mrad) term** and **roll stiffness (mN-m/mrad) term**. Note that "sensitivity" is simply the inverse of "stiffness". Therefore, increasing the stiffness is equivalent to decreasing the sensitivity. These lines define the 6th, 7th and 8th terms of the objective function. An example of these lines is:

```
(6)-----Weight for vertical sensitivity (nm/g) term:
        1.0
(7)-----Weight for pitch sensitivity (urad/uN-m) term:
        1.0
(8)-----Weight for roll sensitivity (urad/uN-m) term:
        1.0
```

The vertical sensitivity term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n}\left(\frac{1}{Vertical\_stiffness_i}\right)^2}}{\sqrt{\sum_{i=1}^{n}\left(\frac{1}{Vertical\_stiffness_{0i}}\right)^2}}\ .$$

The pitch sensitivity term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n}\left(\frac{1}{Pitch\_stiffness_i}\right)^2}}{\sqrt{\sum_{i=1}^{n}\left(\frac{1}{Pitch\_stiffness_{0i}}\right)^2}}\ .$$

The roll sensitivity term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n}\left(\frac{1}{Roll\_stiffness_i}\right)^2}}{\sqrt{\sum_{i=1}^{n}\left(\frac{1}{Roll\_stiffness_{0i}}\right)^2}}\ .$$

Note that the CML triangular mesh solver Quick5 doesn’t calculate the stiffness matrix. When using Quick5, please set the weights of all these sensitivity terms to 0. Also, if you want to optimize these sensitivities using the CML rectangular mesh solver Quick419, you must set the stiffness matrix flag *istiff* in the *run.dat* file to 1.

The last two lines define the weight for the **negative force (g) term**, the 9th term of the objective function. They describe and define the negative force target (absolute value). An example of these lines is:

```
(9)-----Weight for negative force(g) term :
        1.0
        Negative force target (g) (note: give as absolute value)
        2.0
```

Again, the need for this term is similar to that for the roll cutoff and pitch cutoff terms. When designing a slider ABS, we sometimes want to maintain a high negative force value to achieve certain slider performance targets (e.g., load-unload). Again, this is a penalty function for this purpose. If the absolute value of the negative force is higher than the negative force target, this term is set to 0. If the negative force is lower than the target, it will be non-zero, and will have an effect.

This term is defined as:

$$\frac{\sqrt{\sum_{i=1}^{n} Negative\_cutoff_i^2}}{\sqrt{\sum_{i=1}^{n} Negative\_cutoff_{0i}^2}} ,$$

where

$$Negative\_cutoff_i = \begin{cases} 0 & if \quad \left|N\_force_i\right| \geq N\_force_{target} \\ N\_force_{target} - \left|N\_force_i\right| & if \quad \left|N\_force_i\right| < N\_force_{target} \end{cases} .$$

Here N_force means negative force. For this term, if the $\sqrt{\sum_{i=1}^{n} Negative\_cutoff_{0i}^{2}}$ is equal to 0, then we define the initial value of this term as 0. For this case, this term will be defined as $\sqrt{\sum_{i=1}^{n} Negative\_cutoff_{i}^{2}}$ to avoid dividing by zero.

The total objective function value is the total linear summation of the nine terms (including their weights). It is:

$$Objective\_function\_value = \sum_{i=1}^{9} \left( weight_i \times objecitve\_term_i \right)$$

## 5. *OUTPUT FILE*

In addition to the result files of the CML Air Bearing Design program, there are a total of 10 output files generated by the optimization program. These output files are:

| | |
|---|---|
| *rail.dat.opt* | *opti_res.dat* |
| *run.dat.opt* | *opt.dat* |
| *cost.dat* | *scr_sav.dat* |
| *nacost.dat* | *asa.dat* |
| *optcost.dat* | *asa_res* |

These files are all text files.

The *rail.dat.opt* and *run.dat.opt* files contain all the necessary information about the current optimal design. They have the same structure as the corresponding *rail.dat* and *run.dat* input files used by CML Air Bearing Design program. For detailed information about these two files, please refer to the User's Manual for the CML steady code.

The *cost.dat* file contains information about every configuration or design generated by the optimization program. It has 13 fields:

*total objective function value*

*maximum difference in flying height term*

*flying height term*

*roll term*

*roll cutoff term*

*pitch cutoff term*

*vertical sensitivity term*

*pitch sensitivity term*

*roll sensitivity term*

*negative force term*

*cost temperature*

*parameter temperature (the first parameter temperature for the ASA)*

*number of designs generated*

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

*number of designs accepted*

The file *nacost.dat* keeps a record of all accepted designs and *optcost.dat* keeps track of optimized designs, i.e., it is only updated when a design is found to be the best-so-far design (that is, the objective function value is by far the lowest).

The file *opti_res.dat* keeps track of the optimization progress. This file writes the current annealing temperature for both the parameters and the cost, the number of designs generated so far, the number of designs accepted, and the current optimal value of all the parameters used in the optimization. An example of this part of the file for our example case is as follows:

```
num_generated so far: 41
asa->para_k[ 0 ]  so far: 29.687326
asa->para_k[ 1 ]  so far: 31.119396
asa->para_k[ 2 ]  so far: 30.316471
The current *cost_T: 1.027722e+000
The current asa->cost_t: 1.569854e-001
The current *param_T: 2.341335e-002
The current asa->para_t[ 0 ]: 2.341335e-002
The current asa->para_t[ 1 ]: 2.149885e-002
The current asa->para_t[ 2 ]: 2.254466e-002
rail: 5, vertex: 1, x direction: 7.311987e-004
rail: 5, vertex: 2, x direction: 1.005120e-003
rail: 5, vertex: 8, x direction: 7.549578e-004
rail: 10, vertex: 1, x direction (symmetric to r5, v1): 7.311987e-004
rail: 10, vertex: 2, x direction (symmetric to r5, v2): 1.005120e-003
```

```
rail: 10, vertex: 8, x direction (symmetric to r5, v8): 7.549578e-004
rail: 3, vertex: 11, (relative to r5, v1): 7.311987e-004  9.680000e-004
rail: 3, vertex: 12, (relative to r5, v8): 7.549578e-004  8.200000e-004
rail: 5, vertex: 3, (relative to r5, v2): 1.024120e-003  9.640000e-004
rail: 5, vertex: 4, (relative to r5, v2): 1.040120e-003  9.540000e-004
rail: 5, vertex: 5, (relative to r5, v2): 1.050120e-003  9.380000e-004
rail: 5, vertex: 6, (relative to r5, v2): 1.054320e-003  9.190000e-004
rail: 5, vertex: 7, (relative to r5, v2): 1.054320e-003  8.700000e-004
rail: 8, vertex: 11, (relative to r10, v1): 7.311987e-004  3.200000e-005
rail: 8, vertex: 12, (relative to r10, v8): 7.549578e-004  1.800000e-004
rail: 10, vertex: 3, (relative to r10, v2): 1.024120e-003  3.600000e-005
rail: 10, vertex: 4, (relative to r10, v2): 1.040120e-003  4.600000e-005
rail: 10, vertex: 5, (relative to r10, v2): 1.050120e-003  6.200000e-005
rail: 10, vertex: 6, (relative to r10, v2): 1.054320e-003  8.100000e-005
rail: 10, vertex: 7, (relative to r10, v2): 1.054320e-003  1.300000e-004
new load:   1.500000e-003
new xf0:    0.000000e+000
new yf0:    0.000000e+000
new xt:     0.000000e+000
new ht:     0.000000e+000
new recess: 2.500000e-006
new step:   3.000000e-007
Total cost: 6.616694e+000.
FH max_diff term:    1.685984e+000 * 1.000000e+000 = 1.685984e+000,
FH term:             1.452572e-001 * 9.000000e+000 = 1.307315e+000,
Roll term:           8.615574e-001 * 1.000000e+000 = 8.615574e-001
Roll cutoff term:    0.000000e+000 * 1.000000e+000 = 0.000000e+000
Pitch term:          0.000000e+000 * 1.000000e+000 = 0.000000e+000
Vertical sens. term: 8.071782e-001 * 1.000000e+000 = 8.071782e-001
Pitch sens. term:    7.600304e-001 * 1.000000e+000 = 7.600304e-001,
Roll sens. term:     1.194629e+000 * 1.000000e+000 = 1.194629e+000,
Negative force term: 0.000000e+000 * 1.000000e+000 = 0.000000e+000.
```

The file *opt.dat* saves all the important results for all the optimized designs at different radial positions. It has the following six fields :

*flying heights (nm) (note: actual flying heights)*

*minimum flying heights (nm)*

*nominal flying heights (nm)*

*pitches (urad)*

*rolls (urad)*

*negative forces (g)*

Each field will have N real numbers, where N is the number of evaluation points. Generally we choose evaluation points at OD, MD and ID, so N is equal to 3.

The file *scr_sav.dat* keeps all the screen display information during the whole optimization process if you have chosen the concise display mode (mode 2). If you choose the verbose mode (mode 1) in the *constraint.dat* file, this file will not be generated.

The file *asa.dat* keeps all the temperature parameters and the step information for the algorithm during the optimization process. It has six fields:

*number of designs generated*

*number of designs accepted*

*cost temperature*

*all the parameter temperatures*

*number of time step for cost temperature*

*number of time steps for all the parameter temperatures*

The file *asa_res* keeps all the information similar to those of *asa.dat*. It is used mainly to debug the ASA algorithm.

## 6. MATLAB POST-PROCESS FILE

For post-processing, there are four Matlab files that make use of our input files and output files to provide users with a direct graphic explanation of the optimization results. All of them run under Matlab Version 5 or higher. These files are:

*conrail.m* (together with *xline.m*, *yline.m* and *lplot.m*)

*plotopt.m*

*history.m*

*objterm.m*

To illustrate the use of these Matlab post-process files, let's first look at the example case. The initial slider is a pico slider with a flying height of around 7nm. The rail shape of the slider is shown in Figs. 6 and 7. We wish to optimize it to fly uniformly at 5nm while still maintaining a flat roll profile. The input files *constraint.dat*, *rail.dat* and *run.dat* are listed in Lists 1, 2 and 3, respectively. The constraints defined for this example case are shown in Fig. 8.

The Matlab post-process file *conrail.m* is used to show the differences between the optimized design and the initial design. It also shows all the constraints (original, symmetric and relative) defined by the user. For the example case, *conrail.m* yields results shown in Fig. 9.

In Fig. 9, the gray lines show the rail shape of the initial design whereas the dark lines show the rail shape of the optimized design. The three black dots show the original constraint points. The fine solid lines and fine dashed lines represent the symmetric and relative constraints, respectively. We see from Figure 9 that the optimized ABS has a smaller rail area as compared with the initial design. Thus, its flying heights are lowered.

The Matlab post-process file *plotopt.m* shows the variation of the objective function value throughout the optimization process.

For the example case, *plotopt.m* yields the results shown in Fig. 10. The small hollow squares represent the different designs generated. The small solid squares represent designs that were skipped or ignored due to a breach of the solver constraints we set, or due to a slider crash. The larger squares represent the designs that were accepted. Lastly, the dark circles represent all the "best-so-far" optimized designs we obtained during the optimization process. We see that these best-so-far designs have the lowest objective function values so far.

Figure 10 also shows the change in the total objective function value and the percentage of improvement, which is defined as:

$$Percent_{imp} = \frac{Cost_{ini} - Cost_{opt}}{Cost_{ini}} \times 100\% \ ,$$

where $\text{Cost}_{ini}$ means the initial objective function value, and $\text{Cost}_{opt}$ means the objective function value for the final optimized design. $N_{gen}$, $N_{ign}$, $N_{acc}$, $N_{opt}$ represent the number of the designs generated, ignored, accepted and optimized, respectively.

The Matlab post-process file *history.m* shows six parameters for all the best-so-far optimized designs: actual flying height (nm), minimum flying height (nm), nominal flying height (nm), roll (μrad), pitch (μrad) and negative force (g). By running this post-process file for the example case, we generate Fig. 11.

In each of these six small pictures, the horizontal coordinates represent the index number of the best-so-far optimized designs. In this case, index 1 is our initial design and index 7 is the final optimized design. We see that the optimization program found an optimized design with very uniform flying heights around the target flying height (5nm), and with a flat roll profile.

The Matlab post-process file *objterm.m* shows the variations in all nine objective function terms for all the best-so-far optimized designs, and gives the percentage of improvement for each term (defined similarly to the improvement in the objective function value). By running *objterm.m* under Matlab, we generate Fig. 12.

Similarly to Fig. 11, the horizontal coordinates all represent the index number for the different best-so-far optimized designs. From Fig. 12 we see that, for the 2nd objective function term (i.e., the flying height term), the percentage of improvement is very high:

97.07%. That means the final optimized design has a very constant flying height profile around the target flying height. The roll (as seen in the 3rd objective function term), also improved.

In summary, for this example case a greatly optimized ABS design was obtained by using the CML optimization program version 2.0.

## *7. HOW TO RUN THE PROGRAM*

The optimization program is written in the PC Windows version of “C”. As yet, the optimization program doesn’t have a graphic user interface. To use it, download the version you need from the CML website http://cml.me.berkeley.edu, and install it on your PC.

Download the ZIP file *opti2.ZIP* and unzip it under the desired root directory (e.g., the root directory of drive C). You should see a new set of directories:

*C:/opti/program*

*C:/opti/quick*

*C:/opti/m_files*

*C:/opti/example*

The directory *C:/opti/program* contains the optimization executable program *opti.exe.*

The directory *C:/opti/quick* contains the CML Air Bearing Design executable program *Quick419.exe* (rectangular mesh solver) and *Quick5.exe* (triangular mesh solver).

The directory *C:/opti/m_files* contains the Matlab post-process files *conrail.m*, *xline.m*, *yline.m*, *lplot.m*, *plotopt.m*, *history.m* and *objterm.m*.

The directory *C:/opti/example* contains an example case with all of its input and output files.

When you are ready to do an actual optimization, copy the file *c:/opti/opti.exe* to your current working directory and make sure you have all the necessary input files. Then, run the optimization program under Windows by double-clicking on it.

During any stage of the process, you can use the Matlab post-process files to check the results. To do this, start Matlab (version 5 or higher). Then, in the Matlab console window, enter:

```
>> cd C:/your_current_working_directory
>> path (path, 'C:/opti/m_files')
```

Finally, under the Matlab prompt, enter *conrail*, *plotopt*, *history* or *objterm* to see the related results.

## *8. WHEN TO STOP*

At present, the internal parameters are set to keep the optimization program running until the number of generated designs reaches 1000, or the number of accepted designs reaches 500, or the cost temperature reaches 1.0E-7.

In practice, we suggest that you only run the program until the number of generated designs reaches 300 ~ 400. Because the latest optimized design will always be saved in files *rail.dat.opt* and *run.dat.opt*, it is okay to stop or interrupt the optimization program once you think you have the optimized design you want.

## *9. SOME TIPS FOR SUCCESSFUL OPTIMIZATION*

- **Combine the use of the optimization code with the use of CMLAir32 software.**
  The latest CML Air Bearing Design program CMLAir32 v6.0 (PC Windows version) has a graphic user interface and it's very easy to use. Using CMLAir32 is a good way to prepare your initial design, and will also make it quite easy for you to define the constraints.

- **Try to get a better initial design.**
  A better initial design will always help you to find the optimized design more quickly.

- **Define the constraints reasonably.**

  Although constraint points can be set arbitrarily, and the optimization program will always try to find the optimized design with whatever constraints the user has defined, it helps a lot if you can define them more “wisely.” Generally, the fewer the constraint points defined, the faster the optimization goes. We recommend that users choose no more than 6 or 7 original constraint points for faster optimization. If you have some experience or insight as to which points will affect the performance of the slider most, then those points will be the best candidates for constraint points. You can build such experience by playing with the optimization program.

- **Use the Matlab post-process file** *conrail.m* **to check the constraints you defined.**

  Make sure that all the constraints are defined correctly. Running an optimization is a time-consuming process (at least 2 days in most cases). Although the process is automatic (so you don’t need to take care of it once it starts running), you still don’t want to generate a useless “optimized” design because you defined the constraints incorrectly.

  Before you start the optimization program, it is a good strategy to use the Matlab post-process file conrail.m to check the constraints you defined. To do this, copy the file rail.dat to rail.dat.orig and rail.dat.opt and copy the file run.dat to run.dat.orig and run.dat.opt.

- **Try to make use of the constraints for the solver.**

  This too will save you some time in the optimization process. However, don't be too "greedy" and try to set very tight solver constraints. This will hinder the so-called "hill climbing" effect of the optimization algorithm and might cause the program to become "trapped" at some local minimum point.

  When you define the evaluation points in the file *constraint.dat*, remember to enter the points from OD to ID. By doing so the solver invoked by the optimization program will always evaluate the slider performance at OD first. In our experience, bad performance happens most likely at OD, due to its high linear velocity. If this is the case, the program will only need to evaluate the slider design at the first position and then will skip the rest.

- **Choose the weights for different objective function terms reasonably.**

  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 characteristic of a specific design is that it fly completely flat over the radius of the disk at the target flying height, then the flying height and roll terms will need to be weighted more heavily than the other terms you also want to optimize (e.g., sensitivity). In other words, try to avoid sacrificing uniform flying heights or flat rolls to gain stiffness.

- **Use multiple optimizations to get better designs.**

If you are not satisfied with the optimized design, use it as another initial design. Redefine the constraints and let the program optimize it again. Repeated iterations will gradually bring you closer to your final optimized design.

**List 1**: Example listing of the *constraint.dat* file

```
CML Optimization Code Version 2.0   CONSTRAINT.DAT
REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU
*************************************************************
Select solver (1=rectangular solver  2=triangular solver)
1
Select optimization algorithm (1=SA 2=FA 3=ASA 4=Self-defined)
3
Set constraints on solver results? (0=No 1=Yes)
1
Format for solver constraints:
FH_L(nm)  FH_U(nm)  Roll_L(urad)  Roll_U(urad)  Pitch_L(urad)  Pitch_U(urad)
2  10  -30  30  100  400
Screen display mode (1=verbose 2=concise)
2
*************************************************************
Format for non-geometric constraints:
variable name  lower value  upper value  initial value
load(kg)            1.5e-3       1.5e-3       1.5e-3
x offset            0.0          0.0          0.0
y offset            0.0          0.0          0.0
taper  length       0.0          0.0          0.0
taper  angle        0.0          0.0          0.0
recess depth        2.5e-6       2.5e-6       2.5e-6
step   depth        0.3e-6       0.3e-6       0.3e-6
*************************************************************
recess index, step index, mid index, property (1=proportional 2=fixed normal dist.)
1            3            2            1
*************************************************************
Format for original geometric constraints:
rail#   vertex#   dir   low val.   up val.   init val.
5      1      x      0.70e-3      0.85e-3      0.80e-3
5      2      x      0.94e-3      1.09e-3             1.05e-3
5      8      x      0.72e-3      0.86e-3      0.82e-3
Format for symmetric constraints
rail#   vertex#   dir   to be symmetric with:  rail#  vertex#
10     1      x      5      1
10     2      x      5      2
10     8      x      5      8
Format for relative constraints
```

```
rail#   vertex#   to be moved relative to:  rail#   vertex#
3      11     5      1
3      12     5      8
5      3      5      2
5      4      5      2
5      5      5      2
5      6      5      2
5      7      5      2
8      11     10     1
8      12     10     8
10     3      10     2
10     4      10     2
10     5      10     2
10     6      10     2
10     7      10     2
*************************************************************
Format for evaluation points (from OD to ID):
radius(meters)   skew(degrees)
0.031          17.39
0.023           9.1
0.015         -1.22
*************************************************************
Weightings for objective function:
(1)-----Weight for maximum difference in flying height (nm) term:
       1.0
(2)-----Weight for flying height(nm) term:
       9.0
       Target flying height (nm):
       5.0
(3)-----Weight for roll (urad) term:
       1.0
(4)-----Weight for roll - roll cutoff (urad) term:
       1.0
       Roll cutoff (urad):
       5.0
(5)-----Weight for pitch - pitch cutoff (urad) term:
       1.0
       Pitch cutoff (urad):
       300.0
(6)-----Weight for vertical sensitivity (nm/g) term:
       1.0
(7)-----Weight for pitch sensitivity (urad/uN-m) term:
```

```
            1.0
(8)-----Weight for roll sensitivity (urad/uN-m) term:
            1.0
(9)-----Weight for negative force(g) term :
            1.0
            Negative force target (g) (note: give as absolute value)
            2.0
```

**List 2:** Example listing of the original *rail.dat* file

```
CML Version 4.019  RAIL.DAT
REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU
1.2500E-003      1.0000E-003      3.0000E-004
 10               3
 13               1
1.0609E-003      3.7500E-004      2
1.2100E-003      3.7500E-004      0
1.2100E-003      6.2500E-004      2
1.0609E-003      6.2500E-004      2
8.4772E-004      5.5000E-004      2
8.2872E-004      5.4600E-004      2
8.1272E-004      5.3500E-004      2
8.0172E-004      5.1900E-004      2
7.9772E-004      5.0000E-004      2
8.0172E-004      4.8100E-004      2
8.1272E-004      4.6500E-004      2
8.2872E-004      4.5400E-004      2
8.4772E-004      4.5000E-004      2
3.0000E-007
 13               1
8.8159E-004      5.5000E-004      3
8.6259E-004      5.4600E-004      3
8.4659E-004      5.3500E-004      3
8.3559E-004      5.1900E-004      3
8.3159E-004      5.0000E-004      3
8.3559E-004      4.8100E-004      3
8.4659E-004      4.6500E-004      3
8.6259E-004      4.5400E-004      3
8.8159E-004      4.5000E-004      3
1.1259E-003      3.7500E-004      1
1.2100E-003      3.7500E-004      0
1.2100E-003      6.2500E-004      1
1.1259E-003      6.2500E-004      3
0.0000E+000
 13               1
4.0000E-004      7.8000E-004      2
3.0000E-004      6.7000E-004      2
3.0000E-004      5.0000E-004      0
5.0000E-005      5.0000E-004      2
```

```
5.0000E-005       7.1500E-004       2
5.3000E-005       7.4800E-004       2
6.1000E-005       7.8000E-004       2
7.4000E-005       8.1000E-004       2
9.2000E-005       8.3800E-004       2
2.0000E-004       9.6800E-004       2
8.0221E-004       9.6800E-004       0
8.2348E-004       8.2000E-004       2
6.0000E-004       7.8000E-004       2
3.0000E-007
 12                1
1.0000E-004       5.8000E-004       3
1.0000E-004       7.1500E-004       3
1.0300E-004       7.4800E-004       3
1.1100E-004       7.8000E-004       3
1.2400E-004       8.1000E-004       3
1.4200E-004       8.3800E-004       3
2.5000E-004       9.6800E-004       1
5.1694E-004       9.6800E-004       3
5.1694E-004       8.2000E-004       3
3.8000E-004       8.2000E-004       3
2.7000E-004       7.0000E-004       3
2.7000E-004       5.8000E-004       3
0.0000E+000
 8                 1
8.0221E-004       9.6800E-004       1
1.0491E-003       9.6800E-004       1
1.0681E-003       9.6400E-004       1
1.0841E-003       9.5400E-004       1
1.0941E-003       9.3800E-004       1
1.0983E-003       9.1900E-004       1
1.0983E-003       8.7000E-004       1
8.2348E-004       8.2000E-004       3
0.0000E+000
 4                 1
1.2100E-003       6.2500E-004       0
1.2100E-003       3.7500E-004       0
1.2500E-003       3.7500E-004       0
1.2500E-003       6.2500E-004       0
1.5000E-008
 4                 1
1.2200E-003       5.0025E-004       0
```

```
1.2200E-003      4.9975E-004      0
1.2199E-003      4.9975E-004      0
1.2199E-003      5.0025E-004      0
0.0000E+000
 13               1
4.0000E-004      2.2000E-004      2
3.0000E-004      3.3000E-004      2
3.0000E-004      5.0000E-004      0
5.0000E-005      5.0000E-004      2
5.0000E-005      2.8500E-004      2
5.3000E-005      2.5200E-004      2
6.1000E-005      2.2000E-004      2
7.4000E-005      1.9000E-004      2
9.2000E-005      1.6200E-004      2
2.0000E-004      3.2000E-005      2
8.0221E-004      3.2000E-005      0
8.2348E-004      1.8000E-004      2
6.0000E-004      2.2000E-004      2
3.0000E-007
 12               1
1.0000E-004      4.2000E-004      3
1.0000E-004      2.8500E-004      3
1.0300E-004      2.5200E-004      3
1.1100E-004      2.2000E-004      3
1.2400E-004      1.9000E-004      3
1.4200E-004      1.6200E-004      3
2.5000E-004      3.2000E-005      1
5.1694E-004      3.2000E-005      3
5.1694E-004      1.8000E-004      3
3.8000E-004      1.8000E-004      3
2.7000E-004      3.0000E-004      3
2.7000E-004      4.2000E-004      3
0.0000E+000
 8                1
8.0221E-004      3.2000E-005      1
1.0491E-003      3.2000E-005      1
1.0681E-003      3.6000E-005      1
1.0841E-003      4.6000E-005      1
1.0941E-003      6.2000E-005      1
1.0983E-003      8.1000E-005      1
1.0983E-003      1.3000E-004      1
8.2348E-004      1.8000E-004      3
```

```
0.0000E+000
 10 10 10
  0.0000E+000   1.3822E-006   2.7644E-006   4.1467E-006   5.5289E-006   6.9111E-006
8.2933E-006  9.6756E-006  1.1058E-005  1.2440E-005
  0.0000E+000   5.2469E-007   9.8765E-007   1.3889E-006   1.7284E-006   2.0062E-006
2.2222E-006  2.3765E-006  2.4691E-006  2.5000E-006
  0.0000E+000   1.2222E-006   2.4444E-006   3.6667E-006   4.8889E-006   6.1111E-006
7.3333E-006  8.5556E-006  9.7778E-006  1.1000E-005
  3.0000E-007   7.6173E-007   1.1691E-006   1.5222E-006   1.8210E-006   2.0654E-006
2.2556E-006  2.3914E-006  2.4728E-006  2.5000E-006
  0.0000E+000   1.6933E-007   3.3867E-007   5.0800E-007   6.7733E-007   8.4667E-007
1.0160E-006  1.1853E-006  1.3547E-006  1.5240E-006
  0.0000E+000   6.2963E-008   1.1852E-007   1.6667E-007   2.0741E-007   2.4074E-007
2.6667E-007  2.8519E-007  2.9630E-007  3.0000E-007
0.0000E+000      0.0000E+000      2.5000E-006
2.5400E-008      2.5000E-009      0.0000E+000
 1.2200E-003  1.2200E-003  2.4100E-004  2.4100E-004
 5.0000E-004  6.5000E-004  9.7400E-004  2.6000E-005
```

**List 3:** Example listing of the original *run.dat* file

```
CML Version 4.019  RUN.DAT
REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU
***************Solution Control***************
istiff isolv   ioldg   iadpt   isave
 1      1       0       1       0
***************Initial Attitude***************
hm(m)          pitch(rad)     roll(rad)
7.0000E-009    1.5000E-004    5.0000E-006
***************Runs***************
irad           irpm           ialt
 3             1              0
radii(m)
   3.1000E-002 0.023 0.015
skews(deg)
   1.7390E+001   9.1 -1.22
RPMs
   7.2000E+003
altitudes(m)

***************Air Parameters***************
p0(pa)         al(m)          vis(nsm^-2)
1.0135E+005    6.3500E-008    1.8060E-005
***************Load Parameters***************
f0(kg)         xf0(m)         yf0(m)
1.5000E-003    0.0000E+000    0.0000E+000
xfs(µNM)       yfs(µNM)       emax
0.0000E+000    0.0000E+000    1.0000E-003
***************Grid Control***************
nx      ny
 289     289
nsx      nsy     isymm
 1       1       0
xnt(i), i = 2, nsx

nxt(i), i = 2, nsx

dxr(i), i = 1, nsx
 1
ynt(i), i = 2, nsy
```

```
nyt(i), i = 2, nsy

dyr(i), i = 1, nsy
 1
***************Adaptive Grid***************
difmax         decay          ipmax
 40             40             0
***************Reynolds Equation***************
ischeme        imdoel         akmax
 2              3             1.0000E-007
***************Partial Contact***************
icmodel        stdasp(m)      dnsasp(m^-2)
 0             6.0000E-009    1.0000E+012
rdsasp(m)      eyoung(pa)     yldstr(pa)
1.0000E-008    1.0000E+010    1.0000E+012
frcoe          pratio
 0.3            0.3
***************Sensitivities***************
crowninc(m)    camberinc(m)   twistinc(m)
0.0000E+000    0.0000E+000    0.0000E+000
tlnginc(m)     tanginc(rad)   loadinc(kg)
0.0000E+000    0.0000E+000    0.0000E+000
ptrqinc(µNM)   rtrquinc(µNM)  recessinc(m)
0.0000E+000    0.0000E+000    0.0000E+000
iwscale
 1
***************Comments***************
""
```

| | **Optimization Program Version 1.5** | **Optimization Program Version 2.0** |
|---|---|---|
| Operating Platform | UNIX | Windows |
| Algorithm | Self-defined SA | SA, FA, ASA, Self-define SA |
| Solver | Quick300 rectangular mesh solver | Quick419 rectangular mesh solver, Quick5 triangular mesh solver |

Table 1 Differences between the old and new versions

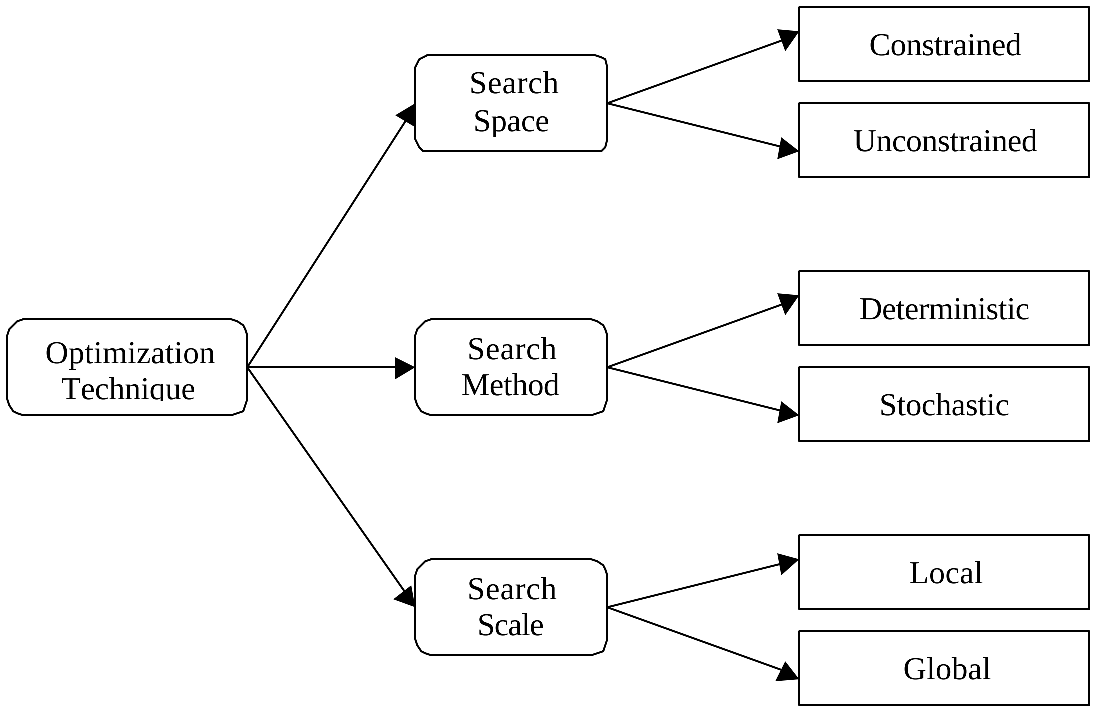

Fig. 1 Optimization technique categories

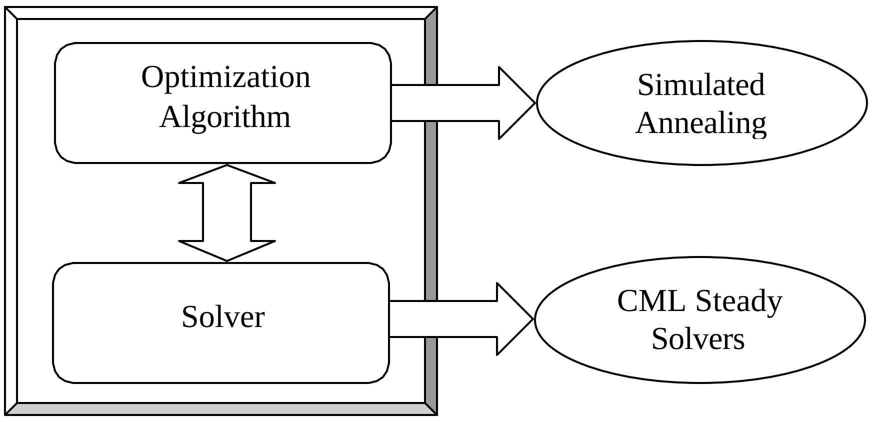

Fig. 2 Structure of optimization program version 2.0

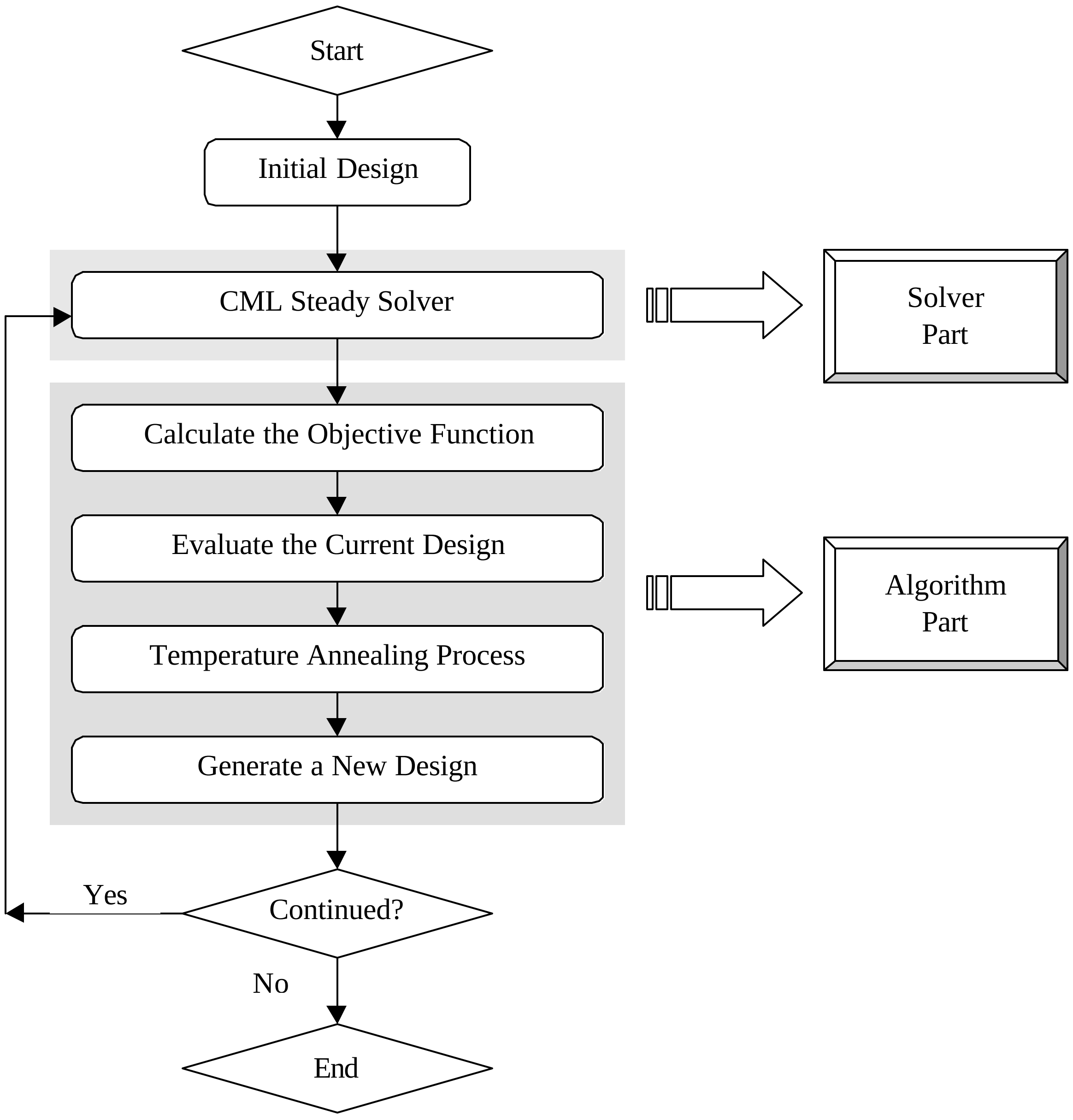

Fig. 3 Flow chart of the CML optimization program version 2.0

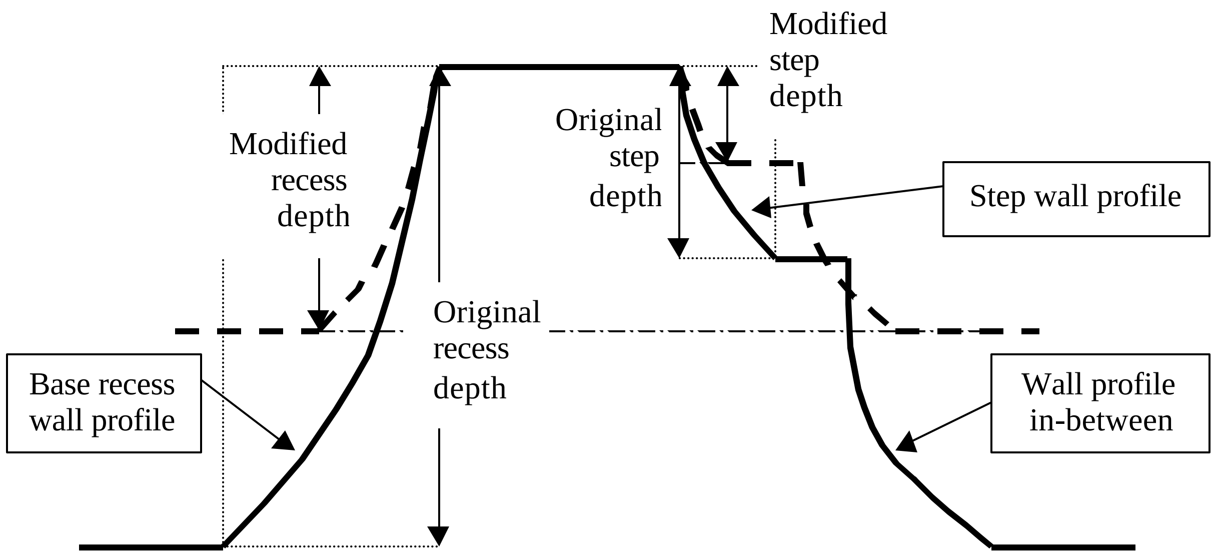

Fig. 4 Modification of wall profiles in proportional mode

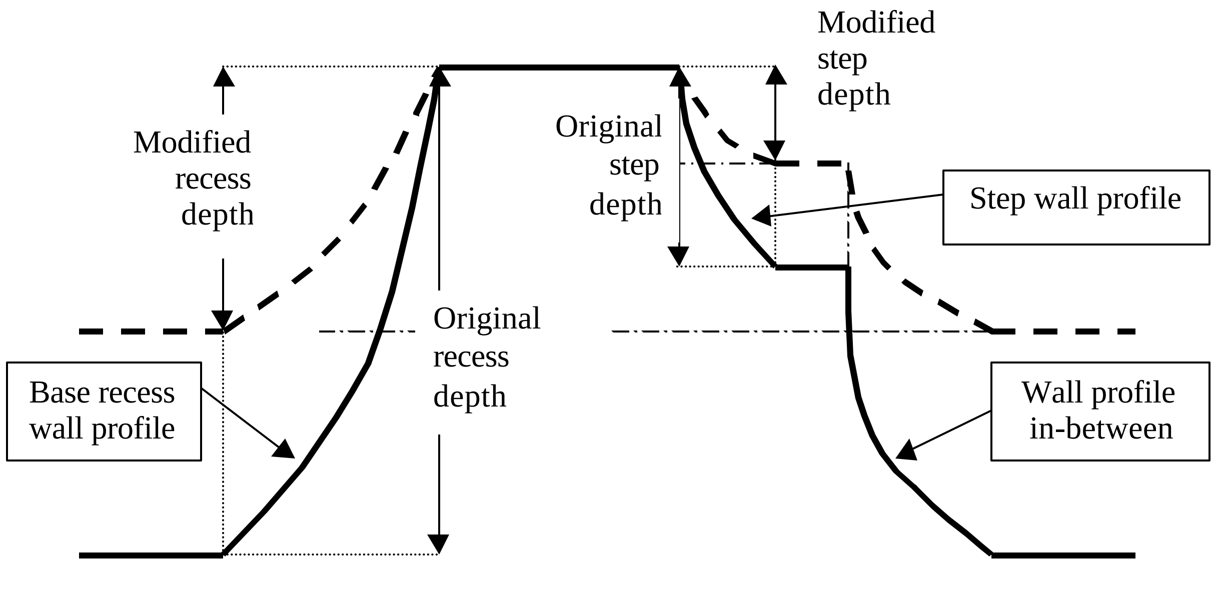

Fig. 5 Modification of wall profiles in fixed normal distance mode

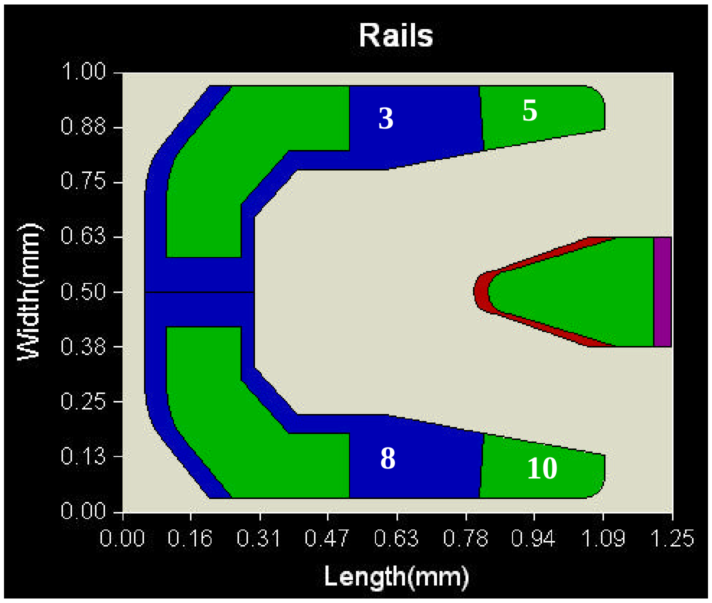

Fig. 6 Rail shape and some rail indices of the initial ABS design

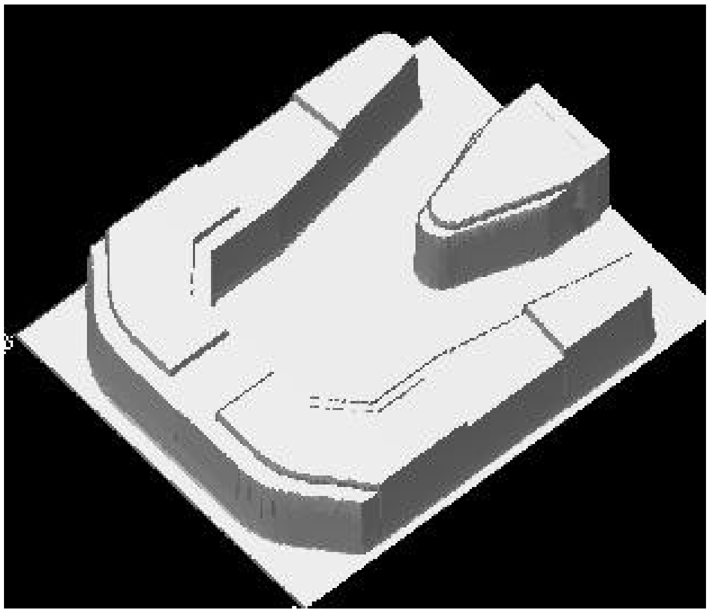

Fig. 7 Three-dimensional rail shape of the initial ABS design

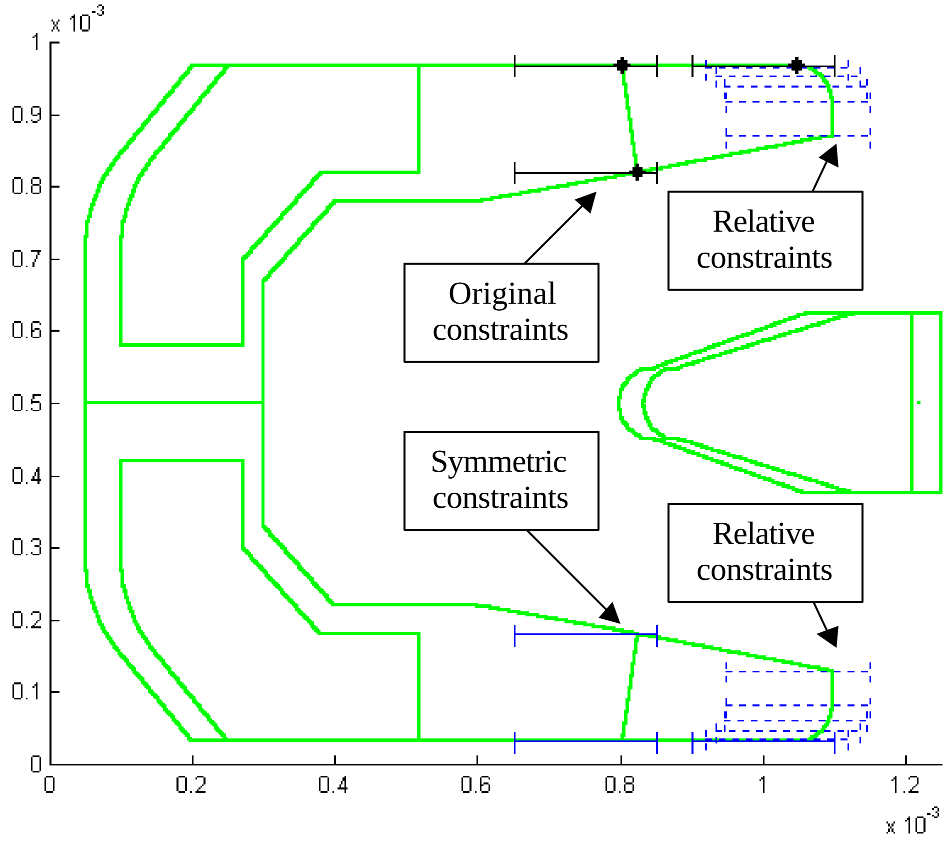

Fig. 8 Constraints defined in the example case

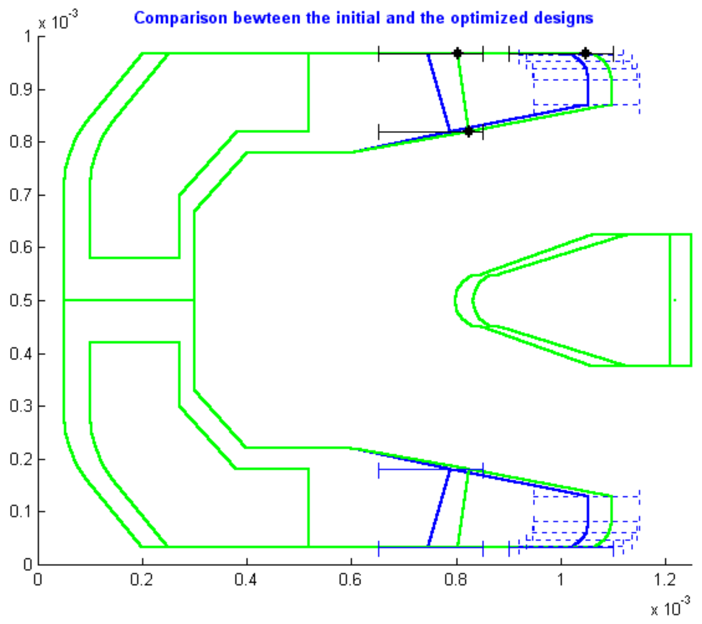

Fig. 9 Comparison between the initial and the optimized designs

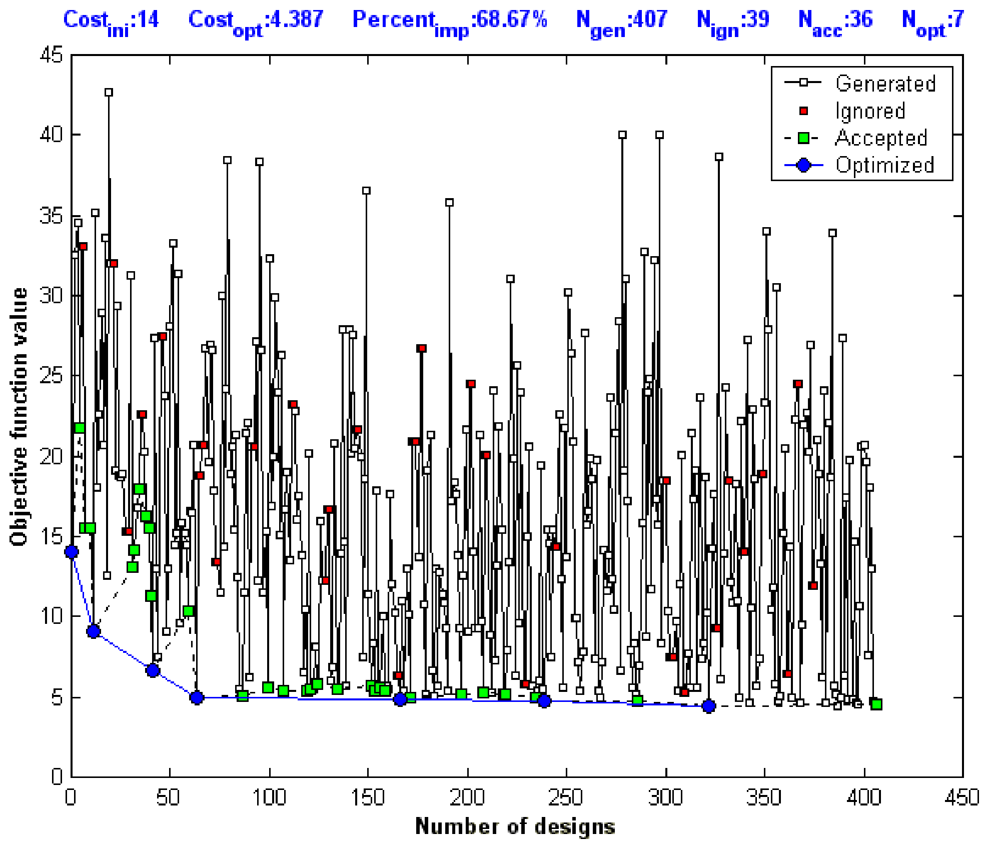

Fig. 10 Variation of the objective function value

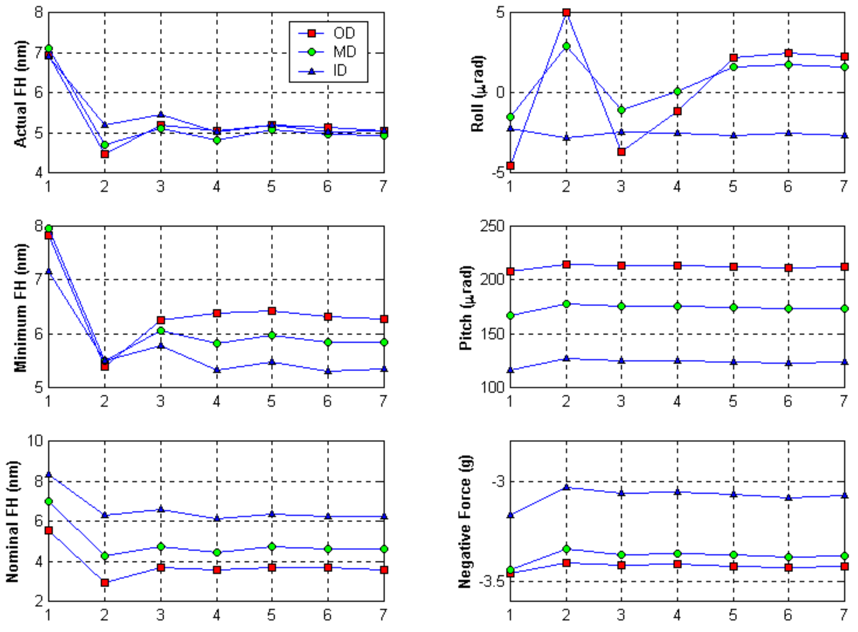

Fig. 11 Variations of the slider performance parameters

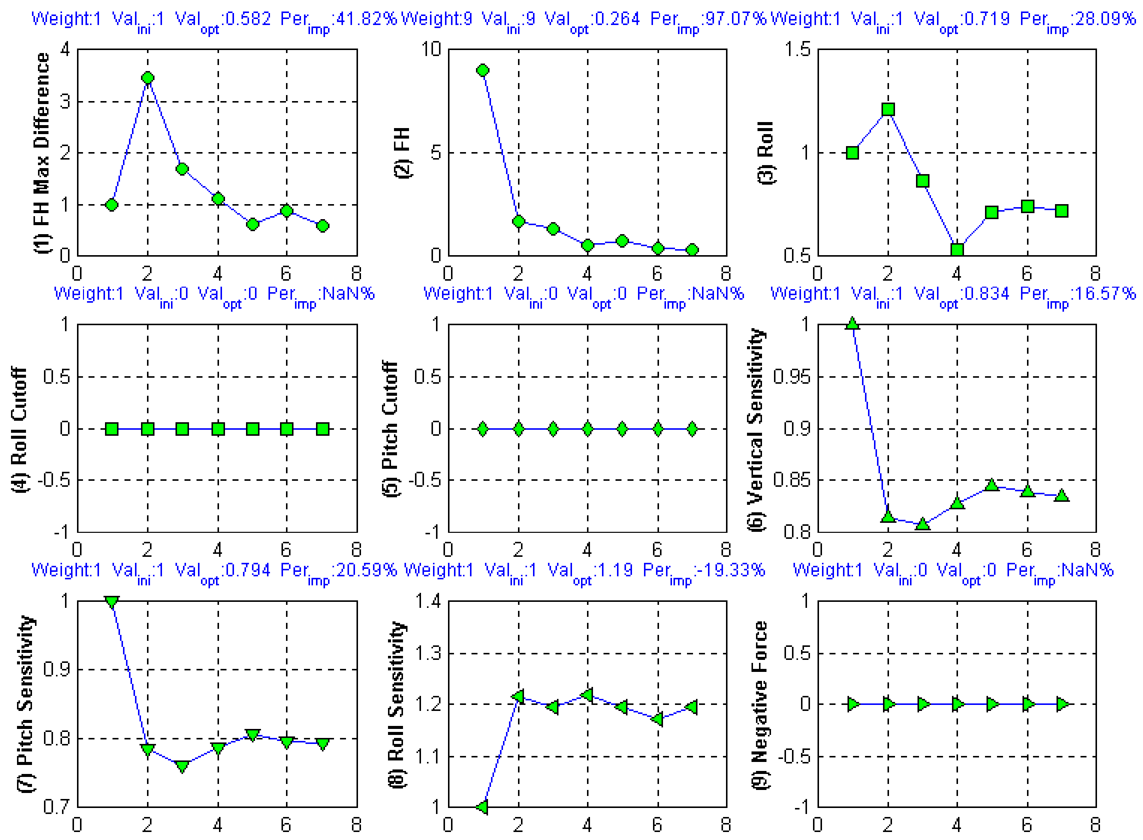

Fig. 12 Variations of the objective function terms