The CML Air Bearing Optimization Program Version 3.0

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 CML Air Bearing Optimization Program Version 3.0, 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 computer hard disk drives.

Unlike Version 2.0, which was developed based on the global stochastic Simulated Annealing algorithm, Version 3.0 was developed based on the more recent global deterministic DIRECT algorithm, which has a much higher convergence rate than does the Simulated Annealing algorithm. Therefore, Version 3.0 can find the optimized ABS design much faster than does Version 2.0.

We also include three locally biased variations of the standard DIRECT algorithm in this program. Generally, these variations have even higher convergence rates than does the standard DIRECT algorithm, and may dramatically reduce the time needed to find the global minimum in some situations.

In addition, two other modifications to the DIRECT algorithm have been included in this program. They are intended to handle manufacturing tolerances and hidden constraints, and they can further improve the efficiency of the DIRECT algorithm.

This program also implements the new geometric constraints. In addition to the original constraint points, constraint rails and lines have also been added. This new feature makes the program much more flexible and convenient to use, and also makes it more powerful for slider ABS optimization.

Like Version 2.0, Version 3.0 also 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. It is more flexible and powerful than Version 2.0 for slider ABS optimization.

1. INTRODUCTION

Optimization is the process of trying to minimize a function subject to certain 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.

The optimization algorithm implemented here is the DIRECT algorithm, which is an acronym for **DI**viding **REC**T**T**angles, a key step in the algorithm. It is a global deterministic algorithm based on the classical one-dimensional Lipschitzian optimization algorithm known as the Shubert algorithm. It is a multi-dimensional Lipschitzian optimization method that can be used without knowing the Lipschitz constant. DIRECT is designed to solve those problems subjected to bounded constraints and it has a very fast convergence rate. Details about the DIRECT algorithm can be found in CML technical report 01-003.

We also include three locally biased variations of the standard DIRECT algorithm in the program. They are: DIRECT-I (having fewer groups), DIRECT-II (having double partitions for the box containing the point with the lowest function value), and DIRECT-III (which combines these two measures). These variations are proposed to further increase the standard DIRECT algorithm's convergence rate and thus improve its efficiency. The three locally biased variations of the DIRECT algorithm generally have higher convergence rates than does the standard DIRECT algorithm. The variations perform especially well in some situations and they may dramatically reduce the time needed to find the global minimum

points. Details of these locally biased variations can be found in CML technical report 01-007.

Two modifications to the DIRECT algorithm, one to handle tolerance (i.e., minimum side lengths) and one to deal with hidden constraints, have also been included in this program. These two modifications to the DIRECT algorithm improve its efficiency and make it more flexible. Details of these modifications can be found in CML technical report 01-013.

In this program we also include the new geometric constraints for slider ABS optimization. For the new geometric constraints, we not only define constraint points, but also define constraint rails and constraint lines.

The constraint rail can be translated in either the length or width directions, rotated with respect to a fixed point, and expanded or shrunk proportionally. The constraint line can be translated in either the length or width directions, rotated with respect to a fixed point, and extended or contracted along its length direction. To maintain a symmetrical slider ABS design and fixed local geometric shapes, we also define symmetrical and relative constraints for constraint rails and constraint lines.

The new geometric constraints can make the slider ABS optimization much easier by enabling users to explore the much wider range of constraints found in practical ABS optimization problems.

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 DIRECT algorithm and the CML steady solvers, including the CML rectangular mesh solver Quick419 and the CML triangular mesh solver Quick5. Figure 1 presents a schematic illustration of the structure of the optimization program version 3.0. The flow chart of the CML optimization program version 3.0 is shown in Fig. 2, where N represents the number of the designs, N_{max} the maximum number of designs prescribed, I the number of iterations and I_{max} the prescribed maximum number of iterations.

3. INPUT FILE

The files *constraint.dat* and *option.dat* are the two input files to the CML slider optimization program version 3.0 (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). The file *constraint.dat* is mainly used to define the

constraints and the objective function. The file *option.dat* is used to define certain control parameters for the DIRECT algorithm.

In Version 2.0, before running the optimization program, users must 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 will create an initial reference design, which is necessary because Version 2.0 of the optimization program generates new designs during the running process and overwrites these two files. In Version 3.0, users are not required to make these copies, because the DIRECT algorithm always picks the midpoint in the search space as the initial sample point. The simulation results from that initial point are used for normalization of the objective function terms throughout the whole optimization process. If desired for comparison purposes, Version 3.0 will allow the user to define the prototype slider ABS design, instead of using the initial design generated by the DIRECT algorithm as the normalization design. In that case, the user should copy the original file *rail.dat* to *rail.dat.ref* and copy the original file *run.dat* to *run.dat.ref*.

All variables that are not set explicitly in the *constraint.dat* and *option.dat* files 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 seven 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 "OPTI341" input file:  CONSTRAINT.DAT  *
*                  Copyright (C) 1998-2002,                   *
*          Computer Mechanics Laboratory, UC Berkeley.        *
*************************************************************
*        PLEASE REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU       *
*************************************************************
```

The next line defines the solver to be used and it should not be edited. It is:

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

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

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

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

Setting hidden constraints 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., a very high Roll), the best course is to skip this design, and do not calculate it for the remaining radial positions. This design will be marked as an

"infeasible design". If we set hidden constraints, on the other hand, we can let the optimization program skip a design automatically according to the conditions we set, and save some calculation 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 are the lower and the upper limit of the roll (μrad), respectively, and Pitch_L and Pitch_U are 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 one of these values, then the evaluated design will be considered to be infeasible 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 should not be edited. It is:

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

Verbose mode means the program will show all the 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=    3.0653  ROLL=    1.7218  PITCH=   89.0202
Point#    2 :      FH=    3.4089  ROLL=    2.1425  PITCH=   90.5169
Point#    3 :      FH=    3.4593  ROLL=   -1.0607  PITCH=   90.7013
Quick419 is DONE!
N_gen: 9   N_opt: 4
Cost: 3.281688e+000  Cost_bsf: 2.455146e+000
```

The first three lines give the flying heights (nm), rolls (μrad) and pitches (μrad) for the three radial positions (OD, MD, ID). The next line shows the solver used by the program (Quick419 in this case). N_gen and N_opt represent the current number of designs generated and the current number of the best-so-far optimized designs found, 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.

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 units of 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, step, mid indexes and WP property (1=proportional 2=fixed)
```

The recess index, step index and mid index represent the wall profile indexes for the base recess (cavity depth), whereas step is the segment between them. When the recess and step are modified by the program, 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. 3 and 4, respectively.

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.

In this program, in addition to the constraint points defined in Version 1.5 and Version 2.0, we include new geometric constraints, i.e., the constraint rails and constraint lines. Figure 5 shows the comparison between the new geometric constraints defined in Version 3.0 and the old geometric constraints defined in Version 1.5 and Version 2.0.

We demonstrate the new constraint rails in Figs. 6 ~ 8. In these figures, the gray lines shows the original slider ABS design, and the dark lines show the shape of the slider ABS design after the rail deformation. Notice that we also defined the symmetrical constraints in order to maintain a symmetrical slider ABS design. In Fig. 6, a rear rail has been translated in

the length direction. In Fig. 7, a rear rail has been rotated with respect to one of its rail points. In Fig. 8, a front rail has been shrunk proportionally.

We demonstrate the new constraint lines in Figs. 9 ~ 11. In these figures, the gray lines shows the original slider ABS design, and the dark lines show the shape of the slider ABS design after the line deformation. We also defined the symmetrical constraints to maintain a symmetrical slider ABS design. In Fig. 9, a line has been translated in the length direction. In Fig. 10, a line has been rotated with respect to one of its endpoints. In Fig. 11, a line has been extended along its length direction.

Next we begin to define the geometric constraints, of which there are three different kinds. Original constraints are mutually independent. Symmetric constraints are defined in order to maintain the symmetry of the geometric shapes. Relative constraints are mostly defined in order to maintain the local geometric shapes.

The next four lines describe the definition of the original rail constraint and should not be edited. They are:

```
************************************************************
RAIL
------------------------------------------------------------
Format for original geometric constraints
```

The next two lines describe how the original rail constraints for translation should be constructed and should not be edited. They are:

```
Translation   (form=1)
rail#  dir                        low_delta     up_delta
```

Here we define the original rail constraints for translation. The first field, **rail#**, defines which rail will be translated. The second field, **dir**, defines in which direction the rail should be translated. The field should read x (or X) if the rail is to move in the slider length direction, or y (or Y) if the rail is to move in the slider width direction. These conventions are consistent with the CML Air Bearing Design program. The third and the fourth fields, **low_delta** and **up_delta**, define the range in which the rail can be moved. Note that these two fields are given by relative coordinates in the units of meters. The **low_delta** must be negative or equal to 0, whereas the **up_delta** must be positive or equal to 0. These four fields must be entered in the following line.

The next two lines describe how the original rail constraints for rotation should be constructed and <u>should not be edited</u>. They are:

```
Rotation      (form=2)
rail#  cent_X       cent_Y       low_ang(deg)  up_ang(deg)
```

Here we define the original rail constraints for rotation. The first field, **rail#**, defines which rail will be rotated. The second and the third fields, **cent_X** and **cent_Y**, define the absolute coordinates of the point with respect to which the rail should rotate. The point can be a vertex of a rail, but it does not have to be. The unit here is meters. The fourth and the fifth fields, **low_ang(deg)** and **up_ang(deg)**, define the angular interval over which the rail

can be rotated. Note that these two fields are given by relative coordinates in the unit of degrees. The counterclockwise direction is defined as positive and the clockwise direction is defined as negative. The **low_ang(deg)** must be negative or equal to 0, while the **up_ang(deg)** must be positive or equal to 0. These five fields must be entered in the following line.

The next two lines describe how the original rail constraints for expansion should be constructed and should not be edited. They are:

```
Expansion     (form=3)
rail#  sign   rail#  pt#    mode   low_delta     up_delta
```

Here we define the original rail constraints for expansion. The first field, **rail#**, defines which rail will be expanded. The second field, **sign**, defines the sign of the rail. If the vertices of the rail are arranged in a counterclockwise direction, the sign of the rail is positive and a plus symbol + should be entered. Otherwise a minus symbol - should be entered for this field. The next three fields describe how the rail should move after the expansion. The third and the fourth fields, **rail#** and **pt#**, define a vertex on this rail. After the expansion, the whole rail will be moved so that the new vertex will coincide with the corresponding initial vertex on that rail. If the **rail#** and **pt#** are both set to 0, the fifth field, **mode**, will define how the expanded rail moves. The **mode** field has five possible values:

- L (or l): the expanded rail will be moved so that its leftmost vertex will have the same X coordinate as the initial rail's leftmost vertex.

- R (or r): the expanded rail will be moved so that its rightmost vertex will have the same X coordinate as the initial rail's rightmost vertex.
- U (or u): the expanded rail will be moved so that its uppermost vertex will have the same Y coordinate as the initial rail's uppermost vertex.
- D (or d): the expanded rail will be moved so that its lowermost vertex will have the same Y value as the initial rail's lowermost vertex.
- *: the expanded rail will not be moved.

The sixth and the seventh fields, **low_delta** and **up_delta**, define the interval over which the rail can be shrunk or expanded. Note that these two fields are given by relative coordinates in the unit of meters. Expansion is defined as positive and shrink is defined as negative. The **low_delta** must be negative or equal to 0, while the **up_delta** must be positive or equal to 0. These seven fields must be entered in the following line.

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

```
------------------------------------------------------------
Format for symmetric constraints
rail#  dir                -->    rail#
```

Here we define the symmetric rail constraints for the problem. Each line consists of three fields. The first field, **rail#**, indicates which rail will be varied symmetrically. The second **dir** field defines in which direction the specified rail should vary. The **dir** field might take the following three possible values:

- X (or x): Only the X coordinates of the vertices on that rail will be varied symmetrically.
- Y (or y): Only the Y coordinates of the vertices on that rail will be varied symmetrically.
- B (or b): Both the X and Y coordinates of the vertices on that rail will be varied symmetrically.

The final field, **rail#**, defines the rail for which the current rail will vary symmetrically.

The next three lines describe how the relative rail constraints should be constructed and <u>should not be edited</u>. They are:

```
------------------------------------------------------------
Format for relative constraints
rail#  form   sign   mode   ==>    rail#
```

Here we define the relative rail constraints for the problem. Each line consists of five fields. The first field, **rail#**, defines which rail will be moving relatively. The second field, **form**, defines by which relative form the current rail will be moving. The form value 1 means translation, 2 means rotation, and 3 means expansion. The third field, **sign**, defines whether the rail's vertices are arranged in counterclockwise direction (+) or clockwise direction (-). The fourth field, **mode**, as before, defines how the current rail moves. The **mode** field has five possible values, i.e., L (or l), R (or r), U (or u), D (or d) and *. The fifth field, **rail#**, defines the rail to move relative to, or the reference rail.

For example, if the form field takes the value 1, the current rail will move the same distance as the reference rail does. If the form field takes the value 2, the current rail will rotate the same angle as the reference rail does. If the form field takes the value 3, the current rail will expand the same distance as the reference rail does.

Multiple constraint rails can be input for all of the above cases.

The next four lines describe how the original line constraint will be defined and should not be edited. They are:

```
*****************************************************************
LINE
-----------------------------------------------------------------
Format for original geometric constraints
```

The next two lines describe how the original line constraints for translation should be constructed and should not be edited. They are:

```
Translation   (form=1)
rail#  pt#    rail#  pt#    dir    low_delta     up_delta
```

Here we define the original line constraints for translation. The first two fields, **rail#** and **pt#**, define the first endpoint of the line. The second two fields, **rail#** and **pt#**, define the second endpoint of the line. The next field, **dir**, defines in which direction the line should be translated. The field should read x (or X) if the line is to move in the slider length direction, or y (or Y) if the line is to move in the slider width direction. These conventions are

consistent with the CML Air Bearing Design program. The last two fields, **low_delta** and **up_delta**, define the interval over which the line can be moved. Note that these two fields are given by relative coordinates in the unit of meters. The **low_delta** must be negative or equal to 0, while the **up_delta** must be positive or equal to 0. These seven fields must be entered in the following line.

The next two lines describe how the original line constraints for rotation should be constructed and <u>should not be edited</u>. They are:

```
Rotation      (form=2)
rail#  pt#    rail#  pt#    cent_X cent_Y low_ang       up_ang
```

Here we define the original line constraints for rotation. The first two fields, **rail#** and **pt#**, define the first endpoint of the line. The second two fields, **rail#** and **pt#**, define the second endpoint of the line. The third two fields, **cent_X** and **cent_Y**, define the absolute coordinates of the point, with respect to which the line should rotate. The point can be a vertex of a rail, but it is not required to be. The unit here is meters. The fourth two fields, **low_ang(deg)** and **up_ang(deg)**, define the angular interval over which the line can be rotated. Note that these two fields are given by relative coordinates in the unit of degrees. The counterclockwise direction is defined as positive and the clockwise direction is defined as negative. The **low_ang(deg)** must be negative or equal to 0, while the **up_ang(deg)** must be positive or equal to 0. These five fields must be entered in the following line.

The next two lines describe how the original line constraints for extension should be constructed and should not be edited. They are:

```
Extension     (form=3)
rail#  pt#    rail#  pt#    mode   low_delta     up_delta
```

Here we define the original line constraints for extension. The first two fields, **rail#** and **pt#**, define the first endpoint of the line. The second two fields, **rail#** and **pt#**, define the second endpoint of the line. The fifth field, **mode**, defines how the line extends. The **mode** field has three possible values:

- 1: Only the first endpoint of the line gets extended along the length direction.
- 2: Only the second endpoint of the line gets extended along the length direction.
- 3: Both endpoints of the line get extended along the length direction.

The sixth and the seventh fields, **low_delta** and **up_delta**, define the interval over which the line can be contracted or extended. Note that these two fields are given by relative coordinates in the unit of meters. Extension is defined as positive and contraction is defined as negative. The **low_delta** must be negative or equal to 0, while the **up_delta** must be positive or equal to 0. These seven fields must be entered in the following line.

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

```
---------------------------------------------------------------
Format for symmetric constraints
rail#  pt#     rail# pt#    dir --> rail#   pt#   rail#   pt#
```

Here we define the symmetric line constraints for the problem. The first two fields, **rail#** and **pt#**, define the first endpoint of the line. The second two fields, **rail#** and **pt#**, define the second endpoint of the line. The fifth **dir** field defines in which direction the specified line should vary. The **dir** field might take any of the following three possible values:

- X (or x): Only the X coordinates of the two endpoints of the line will be varied symmetrically.
- Y (or y): Only the Y coordinates of the two endpoints of the line will be varied symmetrically.
- B (or b): Both the X coordinates and the Y coordinates of the two endpoints of the line will be varied symmetrically.

The third two fields, **rail#** and **pt#**, define the first endpoint of the line for which the current line will vary symmetrically. The fourth two fields, **rail#** and **pt#**, define the second endpoint of the line for which the current line will vary symmetrically.

The next three lines describe how the relative line constraints should be constructed and <u>should not be edited</u>. They are:

```
------------------------------------------------------------
Format for relative constraints
rail#  pt#    rail#  pt#    form ==> rail#   pt#   rail#   pt#
```

Here we define the relative line constraints for the problem. The first two fields, **rail#** and **pt#**, define the first endpoint of the line. The second two fields, **rail#** and **pt#**, define the second endpoint of the line. The fifth field, **form**, defines with which relative form the

current line will be moving. The form value 1 means translation, 2 means rotation, and 3 means extension. The third two fields, **rail#** and **pt#**, define the first endpoint of the line relative to which the current line will vary. The fourth two fields, **rail#** and **pt#**, define the second endpoint of the line relative to which the current line will vary.

For example, if the form field takes the value 1, the current line will move the same distance as the reference line does. If the form field takes the value 2, the current line will rotate the same angle as the reference line does. If the form field takes the value 3, the current line will extend the same distance as the reference line does.

Multiple constraint lines can be input for all above cases.

The next five lines describe how the original constraint points will be defined and should not be edited. They are:

```
*****************************************************************
POINT
-----------------------------------------------------------------
Format for original geometric constraints
rail#   pt#         dir    low_val      up_val       init_val
```

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. Similarily, the **pt#** 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 earlier version 1.5, here all the geometric constraints are given by absolute coordinates in the unit of 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, **init_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 **init_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 three lines describe how the symmetric constraint points should be constructed and should not be edited. They are:

```
---------------------------------------------------------------
Format for symmetric constraints
rail#   pt#         dir    -->    rail#        pt#
```

Here we define the symmetric constraint points for the problem. Each line consists of five fields. The first two fields, **rail#** and **pt#**, indicate which rail and vertex will be varied symmetrically. The **dir** field defines in which direction the specified vertex should vary. This field may take X(or x) or Y(or y) as the value. The final two fields, **rail#** and **pt#**, define the vertex with which the current vertex will vary symmetrically.

The next three lines describe how the relative constraint points should be constructed and should not be edited. They are:

```
---------------------------------------------------------------
Format for relative constraints
rail#   pt#         dir    ==>    rail#   pt#
```

Here we define the relative constraint points for the problem. A relative constraint point fixes a specified vertex to move with the same distance relative to another specified vertex throughout the optimization. Each line consists of four fields. The first two fields, **rail#** and **pt#**, define which vertex gets moved. The **dir** field defines in which direction the specified vertex should vary. The **dir** field might take the following three possible values:

- X (or x): Only the X coordinates of vertex will be varied.
- Y (or y): Only the Y coordinates of the vertex will be varied.
- B (or b): Both the X coordinates and the Y coordinates of the vertex will be varied.

The final two fields, **rail#** and **pt#**, define the vertex relative to which to move.

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(m)    skew(deg)
```

Typically, the optimization program evaluates slider flying height, roll, pitch, etc. at various 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 the convention used in the 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 weights 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 represents the flying height of the current design and FH_0 represents the flying height of the initial design (parameters with sub-index 0 will be regarded as the parameters of the initial design). We 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 flying heights in the result file (e.g., nominal flying height, minimum flying height, etc). The flying height defined 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 the program will take the flying height at the first point of interest as the actual flying height. 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):
    3.5
```

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 evaluation points. 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 (μrad) 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 (μrad) 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 defines 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 $Roll\_cutoff_i = \begin{cases} 0 & if \quad |Roll_i| \le Roll\_cutoff \\ |Roll_i| - Roll\_cutoff & if \quad |Roll_i| > Roll\_cutoff \end{cases}$.

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:
    0.0
    Pitch cutoff (urad):
    300.0
```

This term is defined for reasons similar to those for the roll cutoff term. That is, sometimes we don't want the pitches 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 $Pitch\_cutoff_i = \begin{cases} 0 & if \quad Pitch_i \le Pitch\_cutoff \\ Pitch_i - Pitch\_cutoff & if \quad Pitch_i > Pitch\_cutoff \end{cases}$.

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 (μN-m/μrad) term** and **roll stiffness (μN-m/μrad) 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:
    0.0
(7) Weight for pitch sensitivity (urad/uN-m) term:
    0.0
(8) Weight for roll sensitivity (urad/uN-m) term:
    0.0
```

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 next 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 :
    0.0
    Negative force target (g) (note: 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, we have created a penalty function. 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

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

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 linear summation of the nine terms (including their weights). It is:

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

The last three lines denote the end of the *constraint.dat* file and they should not be edited. They are:

```
*****************************************************************
*                          END OF FILE                          *
*****************************************************************
```

The other input file is *option.dat*. It is used to define some of the control parameters for the DIRECT algorithm.

The first seven lines of the *option.dat* file describe some optimization program information and instructions for reporting bugs. They should not be edited. They are:

```
*****************************************************************
* CML Optimization Code "OPTI341" input file:  OPTION.DAT       *
*                  Copyright (C) 1998-2002,                     *
*          Computer Mechanics Laboratory, UC Berkeley.          *
*****************************************************************
*        PLEASE REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU         *
*****************************************************************
```

The next two lines define the type of DIRECT algorithm to be used and they should not be edited. They are:

```
(1: Standard 2: Fewer Groups 3: Double Partitions 4: Combined)
Choose the DIRECT algorithm [1]:
```

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

The next line defines the maximum number of iterations for the DIRECT algorithm and it should not be edited. It is:

```
Maximum number of iterations for DIRECT [100]:
```

The maximum number of iterations should be entered in the following line.

The next line defines the maximum number of function evaluations for the DIRECT algorithm and it should not be edited. It is:

```
Maximum number of function evaluations for DIRECT [500]:
```

The maximum number of function evaluations should be entered in the following line.

The next line defines whether the manufacturing tolerance should be set for the original constraints and it should not be edited. It is:

```
Set manufacturing tolerance? (1=YES 0=NO)
```

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

The next line defines the manufacturing tolerances for the original constraints and it should not be edited. It is:

```
Manufacturing tolerance for the original constraints:
```

If N original constraints have been defined, then N rows of tolerance values should be entered in the following line. If we do not want to set the manufacturing tolerance, then the value should be set as 0.

The last three lines denote the end of the *option.dat* file and they should not be edited. They are:

```
*************************************************************
*                          END OF FILE                      *
*************************************************************
```

4. OUTPUT FILE

The most important output files generated by the optimization program version 3.0 (in addition to the result files of the CML Air Bearing Design program) are:

rail.dat.opt

run.dat.opt

cost.dat

optcost.dat

opti_res.dat

opt.dat

scr_sav.dat

record.dat

resall.dat

minpoint.dat

These files are all text files.

The *rail.dat.opt* and *run.dat.opt* files contain all the information necessary for the current optimal design. They have the same structure as the corresponding *rail.dat* and *run.dat* input files used by the 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 feasible design generated by the optimization program. It has 11 fields that are defined as follows:

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

number of designs generated

The file *optcost.dat* keeps track of the best-so-far optimized designs, i.e., it is only updated when a design is found to have the so-far-lowest objective function value. Like file *cost.dat*, file *optcost.dat* also has 11 fields. These two files have the same fields except for the last one. For *optcost.dat*, the last field is:

sequence number of the best-so-far optimized designs

The file *opti_res.dat* keeps tracks of the optimization progress. A typical part of this file (for the example case) is as follows:

```
num_generated so far: 4
new load:   1.500000e-003
new xf0:    0.000000e+000
new yf0:    0.000000e+000
new xt:     0.000000e+000
new ht:     1.000000e-002
new recess: 3.000000e-006
new step:   3.000000e-007
Total cost: 4.226780e+000.
FH max_diff term:    6.995107e-001 * 1.000000e+000 = 6.995107e-001,
FH term:             2.416586e-001 * 9.000000e+000 = 2.174927e+000,
Roll term:           1.352342e+000 * 1.000000e+000 = 1.352342e+000
Roll cutoff term:    0.000000e+000 * 1.000000e+000 = 0.000000e+000
```

```
Pitch term:          0.000000e+000 * 0.000000e+000 = 0.000000e+000
Vertical sens. term: 0.000000e+000 * 0.000000e+000 = 0.000000e+000
Pitch sens. term:    0.000000e+000 * 0.000000e+000 = 0.000000e+000,
Roll sens. term:     0.000000e+000 * 0.000000e+000 = 0.000000e+000,
Negative force term: 1.012075e+000 * 0.000000e+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 has N real numbers, where N is the number of evaluation points. Generally we choose evaluation points at OD, MD and ID. Then 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 the verbose mode (mode 1) in the *constraint.dat* file is chosen, this file will not be generated.

The file *record.dat* keeps all the information of the best-so-far optimized ABS design during the optimization process. It has five fields:

iteration number

number of designs generated

current lowest objective function value

normalized coordinates of the best-so-far sample point

normalized dimensions of the box containing the best-so-far sample point

The file *resall.dat* keeps the latest information about all the boxes and sample points in the search space. It has nine fields:

sequence number of the boxes or sample points

partition flag of the box (1 if it is just partitioned, otherwise 0)

infeasible flag of the sample point (1 if it is infeasible, otherwise 0)

partition level of the boxes (the higher the level, the smaller the box)

dimension of the problem

magnitude of the box size

objective function value of the sample points

normalized coordinates of the sample points

normalized dimensions of the boxes containing the sample points

The file *minpoint.dat* keeps information about the minimum point found by the algorithm. It has one line and eight fields:

sequence number of this box or sample point

partition flag of the box (1 if it is just partitioned, otherwise 0)

partition level of the box (the higher the level, the smaller the box)

dimension of the problem

magnitude of the box size

minimum value of the objective function

normalized coordinates of the sample point

normalized dimensions of the box containing the sample point

5. MATLAB PRE-PROCESS GUI

A Matlab pre-process Graphic User Interface (GUI) called the Slider Geometry Interface Visualization GUI has been developed. This GUI has been developed to help the user define the constraints more easily. Its Matlab file is *railgui2.m* (with supporting files and sub-directories) and it runs under Matlab Version 5 or higher.

This pre-process GUI takes the *rail.dat.orig* file as the only input file and it can show the slider 2-D geometry interactively. Figures 12 ~ 15 demonstrate how to use this GUI.

Figure 12 shows the initial appearance of the GUI after loading the example case. The GUI consists of menus, a list box on the left and a plot area on the right. In this state, when the cursor is moved close to a vertex of a rail in the plot area, the rail index and the vertex sequence number will be displayed instantaneously, as shown in Fig. 12.

If we left click on that vertex, then that vertex and the rail it is on will be highlighted and the information about that rail and all of its vertices (vertices sequence number, x and y coordinates, wall profile indexes) will be displayed in the left list box, as shown in Fig. 13. That vertex will also be highlighted in the list box.

If we click on a rail, then the whole rail will be highlighted and all of its information will be displayed in the left list box, as shown in Fig. 14. Additionally, if we select a rail index from the top of that list box, the corresponding rail will also be highlighted in the plot area.

Figure 15 shows the structure of the menus. There are 3 menu items: *Information*, *Option* and *Close*. The *Information* menu shows general information about this GUI. The *Close* menu is used to quit the GUI program. The *Option* menu allows the user to show the Matlab built-in menus in order to better manipulate the graphics, and also to show the current date at the left bottom of the interface, as shown in Fig. 15. The user may also change the background color for the window, the list box and the plot area, as well as the cursor shape. Selecting *Default Settings* will reset all options to their default settings.

6. MATLAB POST-PROCESS FILE

We have also created four Matlab files for post-processing. They use the input and output files, and can provide users with a direct graphic explanation of the optimization results. All of them run under Matlab Version 5 or higher. These files are:

plotopt3.m

conrail3.m (together with *xline.m, yline.m* and *lplot.m*)

history3.m

objterm3.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 FHs around 5nm. The rail shape of the slider is shown in Figs. 16 and 17. We wish to optimize it by lowering its FHs to 3.5nm while still maintaining a flat roll profile. The input files *constraint.dat*, *option.dat*, *rail.dat* and *run.dat* are shown in List 1, 2, 3 and 4, respectively. For the example case we defined two original constraint rails, i.e., the side rails of the slider can move along the length direction and they can also expand or shrink.

The Matlab post-process file *plotopt3.m* is used to show the variation in the objective function value during the optimization process.

For the example case, running *plotopt3.m* yields the results shown in Fig. 18. The small hollow squares represent all the designs generated. The small solid squares represent the infeasible designs that were skipped or ignored due to either a breach in the hidden constraints we set or due to a slider crash. The dark circles represent all the "best-so-far" optimized designs obtained during the optimization process. These best-so-far designs have the lowest objective function values obtained so far.

Figure 18 also shows the change in the total objective function value and the percentage of the improvement, which is defined as:

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

where $Cost_{ini}$ means the initial objective function value, and $Cost_{opt}$ means the objective function value for the final optimized design. N_{gen}, N_{ign}, and N_{opt} in this figure represent the number of the designs generated, ignored and optimized, respectively.

The Matlab post-process file *conrail3.m* is used to show the differences between the optimized design and the initial design. For the example case, running of *conrail3.m* yields the results shown in Fig. 19.

In Fig. 19, the gray lines show the rail shape of the initial design, whereas the dark lines show the rail shape of the optimized design. We see that the side rails have been reduced and moved toward the leading edge of the slider ABS design, thus lowering its FHs.

The Matlab post-process file *history3.m* is used to show 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). Running this post-process file for the example case yields the results shown in Fig. 20.

In Fig. 20, the horizontal coordinates in each of the six small pictures represent the index number of all the best-so-far optimized designs. In this case, index 1 is the initial

design and index 11 is the final optimized design. We see that the optimization program found a design with very uniform flying heights, around the target flying height of 3.5nm, as well as a flat roll profile.

The Matlab post-process file *objterm3.m* shows the variations in all nine objective function terms for all the best-so-far optimized designs, as well as the percentage of improvement for each term. Percentage of improvement here is defined similarly to that for the objective function value. Running *objterm3.m* under Matlab yields the results shown in Fig. 21.

Again, all the horizontal coordinates represent the index number of various best-so-far optimized designs. In Fig. 21 we see that for the 2nd objective function term (i.e., the flying height term), the percentage of improvement is a very high 95.15%. That means that the final optimized design has a very constant flying height profile around the target flying height. However, the roll term (the 3rd objective function term) did not improve. The 4th objective function term (the roll cutoff term) remains 0 for all the best-so-far optimized designs. The remaining objective function terms are all 0 because we defined their weights as 0.

In summary, for this example case, a greatly optimized ABS design was obtained by using the CML optimization program version 3.0.

7. HOW TO RUN THE PROGRAM

The optimization program is written in "C" running under PC Windows. Currently, the optimization program doesn't have a graphic user interface. In order to use it, you must download the version you need from the CML website (http://cml.me.berkeley.edu) and install it on your PC.

Download the ZIP file *opti3.ZIP* to your PC, unzip it under the root directory (e.g., the root directory of drive C). You should see a new set of directories:

C:/opti3/program

C:/opti3/quick

C:/opti3/m_files

C:/opti3/example

The directory *C:/opti3/program* contains the optimization executable program *opti3.exe*.

The directory *C:/opti3/quick* contains the CML Air Bearing Design executable program *Quick419.exe* (rectangular mesh solver) and *Quick5.exe* (triangular mesh solver).

The directory *C:/opti3/m_files* contains both the Matlab pre-process GUI file *railgui2.m* and the Matlab post-process files *conrail3.m*, *xline.m*, *yline.m*, *lplot.m*, *plotopt3.m*, *history3.m* and *objterm3.m*.

The directory *C:/opti3/example* contains an example case with all the input and output files.

When you are ready to perform the optimization, copy the file *c:/opti3/opti3.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. First, start Matlab (version 5 or higher). Then, in the Matlab console window, enter:

>> cd C:/your_current_working_directory

>> path (path, 'C:/opti3/m_files')

Finally, under the Matlab prompt, enter *railgui2, conrail3*, *plotopt3*, *history3* or *objterm3* to see the related results.

8. WHEN TO STOP

The DIRECT algorithm has a much faster convergence rate than does ASA. In practice, for DIRECT defining a maximum number of function evaluations of 200 ~ 300 (in *option.dat*) should be adequate for low-dimensional problems. Higher numbers should be set for higher-dimensional problems. Because the latest optimized design will always be saved in the 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 graphical user interface and it's very easy to use. A good way to prepare your initial design is to use of CMLAir32.

- **Use the Matlab pre-process GUI *railgui2.m* to define the constraints.**

  It is quite easy to use the Matlab pre-process GUI *railgui2.m* to define and double-check the constraints. To do this, you need to copy the file *rail.dat* to *rail.dat.orig*.

- **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 constraints can be set arbitrarily, and the optimization program will always attempt to find an optimized design with whatever constraints the user defines, it helps greatly to define them properly. Also, the fewer the constraints defined, the faster the optimization will be. We recommend that users choose no more than 10 original constraints for faster optimization.

- **Make use of the manufacturing tolerance and hidden constraints.**

  By defining the manufacturing tolerance, the algorithm can avoid wasting time in partitioning boxes with sides smaller than the tolerance. Thus, the algorithm can put more effort into exploring larger boxes in the search space and improve its efficiency.

  Defining the hidden constraints will also save some calculation time in the optimization process. However, one must be careful when using very strict hidden constraints. If the constraint points are not properly defined the algorithm may not be able to yield optimized designs.

  When you define the evaluation points in the file *constraint.dat*, remember to list the points in order from OD to ID. The solver invoked by the optimization program will always evaluate the slider's performance at the OD first. In our experience, bad performance is most likely at the OD, due to its high linear velocity. In this 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.**

In general, 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 at the target flying height, then the flying height and roll terms should have greater weights than other terms you also want to optimize (e.g., sensitivity). In other words, try to avoid sacrificing uniformity of flying heights and flatness of rolls in order to gain the stiffness.

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

  If you are not satisfied with the current optimized design, use it as a new initial design, redefine the constraints, and then optimize it again. Several iterations will eventually yield a final optimized design.

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

```
*************************************************************
* CML Optimization Code "OPTI341" input file:  CONSTRAINT.DAT  *
*                  Copyright (C) 1998-2002,                   *
*          Computer Mechanics Laboratory, UC Berkeley.        *
*************************************************************
*        PLEASE REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU       *
*************************************************************
Select solver (1=rectangular solver  2=triangular solver)
1
Set constraints on solver results? (0=No 1=Yes)
1
Format for solver constraints:
FH_L(nm)   FH_U(nm)   R_L(urad)   R_U(urad)   P_L(urad)   P_U(urad)
2          10         -30         30          50          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, step, mid indexes and WP property (1=proportional 2=fixed)
1      3      2      1
*************************************************************
RAIL
-------------------------------------------------------------
Format for original geometric constraints
Translation   (form=1)
rail#  dir                        low_delta     up_delta
4      x                          -0.15e-3      0.15e-3
Rotation      (form=2)
rail#  cent_X       cent_Y       low_ang(deg)  up_ang(deg)
Expansion     (form=3)
```

```
rail#  sign   rail#  pt#    mode   low_delta     up_delta
4      -      0      0      u      -0.02e-3      0.02e-3
-----------------------------------------------------------------
Format for symmetric constraints
rail#  dir                 -->    rail#
2      b                          4
-----------------------------------------------------------------
Format for relative constraints
rail#  form   sign   mode   ==>    rail#
*****************************************************************
LINE
-----------------------------------------------------------------
Format for original geometric constraints
Translation   (form=1)
rail#  pt#    rail#  pt#    dir     low_delta    up_delta
Rotation      (form=2)
rail#  pt#    rail#  pt#    cent_X  cent_Y      low_ang    up_ang
Extension     (form=3)
rail#  pt#    rail#  pt#    mode    low_delta    up_delta
-----------------------------------------------------------------
Format for symmetric constraints
rail#  pt#     rail#pt#    dir --> rail#   pt#   rail#   pt#
-----------------------------------------------------------------
Format for relative constraints
rail#  pt#    rail#  pt#    form ==> rail#   pt#   rail#   pt#
*****************************************************************
POINT
-----------------------------------------------------------------
Format for original geometric constraints
rail#   pt#         dir    low_val      up_val       init_val
-----------------------------------------------------------------
Format for symmetric constraints
rail#   pt#         dir    -->    rail#        pt#
-----------------------------------------------------------------
Format for relative constraints
rail#   pt#         dir    ==>    rail#   pt#
*****************************************************************
Format for evaluation points (from OD to ID):
radius(m)   skew(deg)
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):
    3.5
(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:
    0.0
    Pitch cutoff (urad):
    300.0
(6) Weight for vertical sensitivity (nm/g) term:
    0.0
(7) Weight for pitch sensitivity (urad/uN-m) term:
    0.0
(8) Weight for roll sensitivity (urad/uN-m) term:
    0.0
(9) Weight for negative force(g) term :
    0.0
    Negative force target (g) (note: absolute value)
    2.0
****************************************************************
*                          END OF FILE                         *
****************************************************************
```

List 2: Example listing of *option.dat* file

```
****************************************************************
*   CML Optimization Code "OPTI341" input file:  OPTION.DAT    *
*                  Copyright (C) 1998-2001,                    *
*          Computer Mechanics Laboratory, UC Berkeley.         *
****************************************************************
*        PLEASE REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU        *
****************************************************************
(1: Standard 2: Fewer Groups 3: Double Partitions 4: Combined)
Choose the DIRECT algorithm [1]:
1
Maximum number of iterations for DIRECT [100]:
100
Maximum number of function evaluations for DIRECT [500]:
400
Set manufacturing tolerance? (1=YES 0=NO)
0
Manufacturing tolerance for the original constraints:
0.0
0.0
****************************************************************
*                          END OF FILE                         *
****************************************************************
```

List 3: Example listing of original *rail.dat* file

```
CML Version 4.018   RAIL.DAT
REPORT BUGS TO INFO@CML.ME.BERKELEY.EDU
1.250000e-003  1.000000e-003  3.000000e-004
4  3
21 1
1.250000e-003  6.500000e-004  2
1.150000e-003  6.500000e-004  2
1.124000e-003  6.477200e-004  2
1.098700e-003  6.409500e-004  2
1.075000e-003  6.299000e-004  2
1.053600e-003  6.149100e-004  2
1.035100e-003  5.964200e-004  2
1.020100e-003  5.750000e-004  2
1.009100e-003  5.513000e-004  2
1.002300e-003  5.260500e-004  2
1.000000e-003  5.000000e-004  2
1.002300e-003  4.739500e-004  2
1.009100e-003  4.487000e-004  2
1.020100e-003  4.250000e-004  2
1.035100e-003  4.035800e-004  2
1.053600e-003  3.850900e-004  2
1.075000e-003  3.701000e-004  2
1.098700e-003  3.590500e-004  2
1.124000e-003  3.522800e-004  2
1.150000e-003  3.500000e-004  2
1.250000e-003  3.500000e-004  2
0.000000e+000
38 1
7.500000e-004  0.000000e+000  2
7.604200e-004  9.100186e-007  2
7.705200e-004  3.620051e-006  2
7.800000e-004  8.039991e-006  2
7.885700e-004  1.404004e-005  2
7.959600e-004  2.143008e-005  2
8.019600e-004  3.000005e-005  2
8.063800e-004  3.948004e-005  2
8.090900e-004  4.958006e-005  2
8.100000e-004  6.000005e-005  2
8.090900e-004  7.042004e-005  2
```

```
8.063800e-004  8.052005e-005  2
8.019600e-004  9.000005e-005  2
7.959600e-004  9.857008e-005  2
7.885700e-004  1.059601e-004  2
7.800000e-004  1.119600e-004  2
7.705200e-004  1.163800e-004  2
7.604200e-004  1.190900e-004  2
7.500000e-004  1.200000e-004  2
4.874998e-004  1.200000e-004  2
4.770800e-004  1.190900e-004  2
4.669800e-004  1.163800e-004  2
4.575000e-004  1.119600e-004  2
4.489300e-004  1.059601e-004  2
4.415400e-004  9.857008e-005  2
4.355400e-004  9.000005e-005  2
4.311200e-004  8.052005e-005  2
4.284100e-004  7.042004e-005  2
4.275000e-004  6.000005e-005  2
4.284100e-004  4.958006e-005  2
4.311200e-004  3.948004e-005  2
4.355400e-004  3.000005e-005  2
4.415400e-004  2.143008e-005  2
4.489300e-004  1.404004e-005  2
4.575000e-004  8.039991e-006  2
4.669800e-004  3.620051e-006  2
4.770800e-004  9.100186e-007  2
4.875000e-004  0.000000e+000  2
0.000000e+000
36 1
2.934100e-004  2.538000e-004  2
3.355100e-004  2.650800e-004  2
3.750000e-004  2.834900e-004  2
4.107000e-004  3.084900e-004  2
4.415100e-004  3.393000e-004  2
4.665100e-004  3.750000e-004  2
4.849200e-004  4.144900e-004  2
4.962000e-004  4.565900e-004  2
5.000000e-004  5.000000e-004  2
4.962000e-004  5.434100e-004  2
4.849200e-004  5.855100e-004  2
4.665100e-004  6.250000e-004  2
4.415100e-004  6.607000e-004  2
```

```
4.107000e-004  6.915100e-004  2
3.750000e-004  7.165100e-004  2
3.355100e-004  7.349200e-004  2
2.934100e-004  7.462000e-004  2
2.500000e-004  7.500000e-004  2
2.065900e-004  7.462000e-004  2
1.644900e-004  7.349200e-004  2
1.250000e-004  7.165100e-004  2
8.930300e-005  6.915100e-004  2
5.848900e-005  6.607000e-004  2
3.349400e-005  6.250000e-004  2
1.507700e-005  5.855100e-004  2
3.798100e-006  5.434100e-004  2
0.000000e+000  5.000000e-004  2
3.798100e-006  4.565900e-004  2
1.507700e-005  4.144900e-004  2
3.349400e-005  3.750000e-004  2
5.848900e-005  3.393000e-004  2
8.930300e-005  3.084900e-004  2
1.250000e-004  2.834900e-004  2
1.644900e-004  2.650800e-004  2
2.065900e-004  2.538000e-004  2
2.500000e-004  2.500000e-004  2
0.000000e+000
38 1
7.500000e-004  1.000000e-003  2
7.604200e-004  9.990900e-004  2
7.705200e-004  9.963800e-004  2
7.800000e-004  9.919601e-004  2
7.885700e-004  9.859600e-004  2
7.959600e-004  9.785700e-004  2
8.019600e-004  9.700000e-004  2
8.063800e-004  9.605200e-004  2
8.090900e-004  9.504200e-004  2
8.100000e-004  9.400000e-004  2
8.090900e-004  9.295800e-004  2
8.063800e-004  9.194800e-004  2
8.019600e-004  9.100000e-004  2
7.959600e-004  9.014300e-004  2
7.885700e-004  8.940400e-004  2
7.800000e-004  8.880400e-004  2
7.705200e-004  8.836200e-004  2
```

```
7.604200e-004  8.809100e-004  2
7.500000e-004  8.800000e-004  2
4.874998e-004  8.800000e-004  2
4.770800e-004  8.809100e-004  2
4.669800e-004  8.836200e-004  2
4.575000e-004  8.880400e-004  2
4.489300e-004  8.940400e-004  2
4.415400e-004  9.014300e-004  2
4.355400e-004  9.100000e-004  2
4.311200e-004  9.194800e-004  2
4.284100e-004  9.295800e-004  2
4.275000e-004  9.400000e-004  2
4.284100e-004  9.504200e-004  2
4.311200e-004  9.605200e-004  2
4.355400e-004  9.700000e-004  2
4.415400e-004  9.785700e-004  2
4.489300e-004  9.859600e-004  2
4.575000e-004  9.919601e-004  2
4.669800e-004  9.963800e-004  2
4.770800e-004  9.990900e-004  2
4.875000e-004  1.000000e-003  2
0.000000e+000
10   10   10
0.000000e+000 1.382200e-006 2.764400e-006 4.146700e-006 5.528900e-006 6.911100e-006
8.293300e-006 9.675600e-006 1.105800e-005 1.244000e-005
0.000000e+000 5.246900e-007 9.876500e-007 1.388900e-006 1.728400e-006 2.006200e-006
2.222200e-006 2.376500e-006 2.469100e-006 2.500000e-006
0.000000e+000 1.222200e-006 2.444400e-006 3.666700e-006 4.888900e-006 6.111100e-006
7.333300e-006 8.555600e-006 9.777800e-006 1.100000e-005
3.000000e-007 7.617300e-007 1.169100e-006 1.522200e-006 1.821000e-006 2.065400e-006
2.255600e-006 2.391400e-006 2.472800e-006 2.500000e-006
0.000000e+000 1.693300e-007 3.386700e-007 5.080000e-007 6.773300e-007 8.466700e-007
1.016000e-006 1.185300e-006 1.354700e-006 1.524000e-006
0.000000e+000 6.296300e-008 1.185200e-007 1.666700e-007 2.074100e-007 2.407400e-007
2.666700e-007 2.851900e-007 2.963000e-007 3.000000e-007
0.000000e+000 1.000000e-002 3.000000e-006
2.740000e-008 2.500000e-009 0.000000e+000
1.250000e-003 0.000000e+000 0.000000e+000 6.250000e-004
5.000000e-004 0.000000e+000 1.000000e-003 5.000000e-004
```

List 4: Example listing of original *run.dat* file

```
CML Version 4.018   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
   5.0000e-008   1.5000e-004   0.0000e+000
***************Runs************
 irad            irpm             ialt
   3   1   0
 radii(m)
  3.1000e-002    2.3000e-002    1.5000e-002
 skews(deg)
  1.7000e+001    9.0000e+000    -1.0000e+000
 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               xf0                yf0
   1.5000e-003   0.0000e+000   0.0000e+000
 xfs              yfs                emax
   0.0000e+000   0.0000e+000   1.0000e-003
 **************Grid Control**************
 nx    ny
   225   225
 nsx   nsy   isymm
 1       1      0
 xnt(i),i=2,nsx

 nxt(i),i=2,nsx

 dxr(i),i=1,nsx
  1.000000e+000
 ynt(i),i=2,nsy
```

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

 dyr(i),i=1,nsy
  1.000000e+000
***************Adaptive Grid************
 difmax      decay      ipmax
40.000000 40.000000 0
***************Reynolds Equation************
 ischeme     imodel     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
0.300000 0.300000
***************Sensitivities************
 crowninc      camberinc     twistinc
0.0000e+000    0.0000e+000    0.0000e+000
 tlnginc       tanginc       loadinc
0.0000e+000    0.0000e+000    0.0000e+000
 ptrqinc       rtrquinc      recessinc
0.0000e+000    0.0000e+000    0.0000e+000
 iwscale
1
*****************Comments*************
"This is a test case"
```

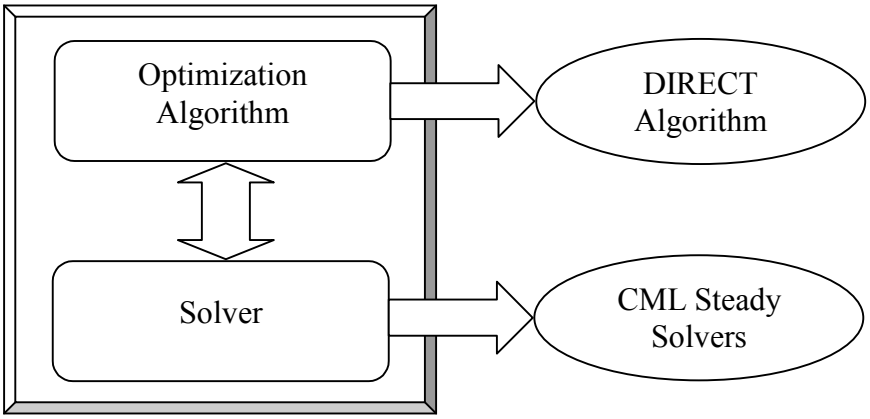

Fig. 1 Structure of optimization program version 3.0

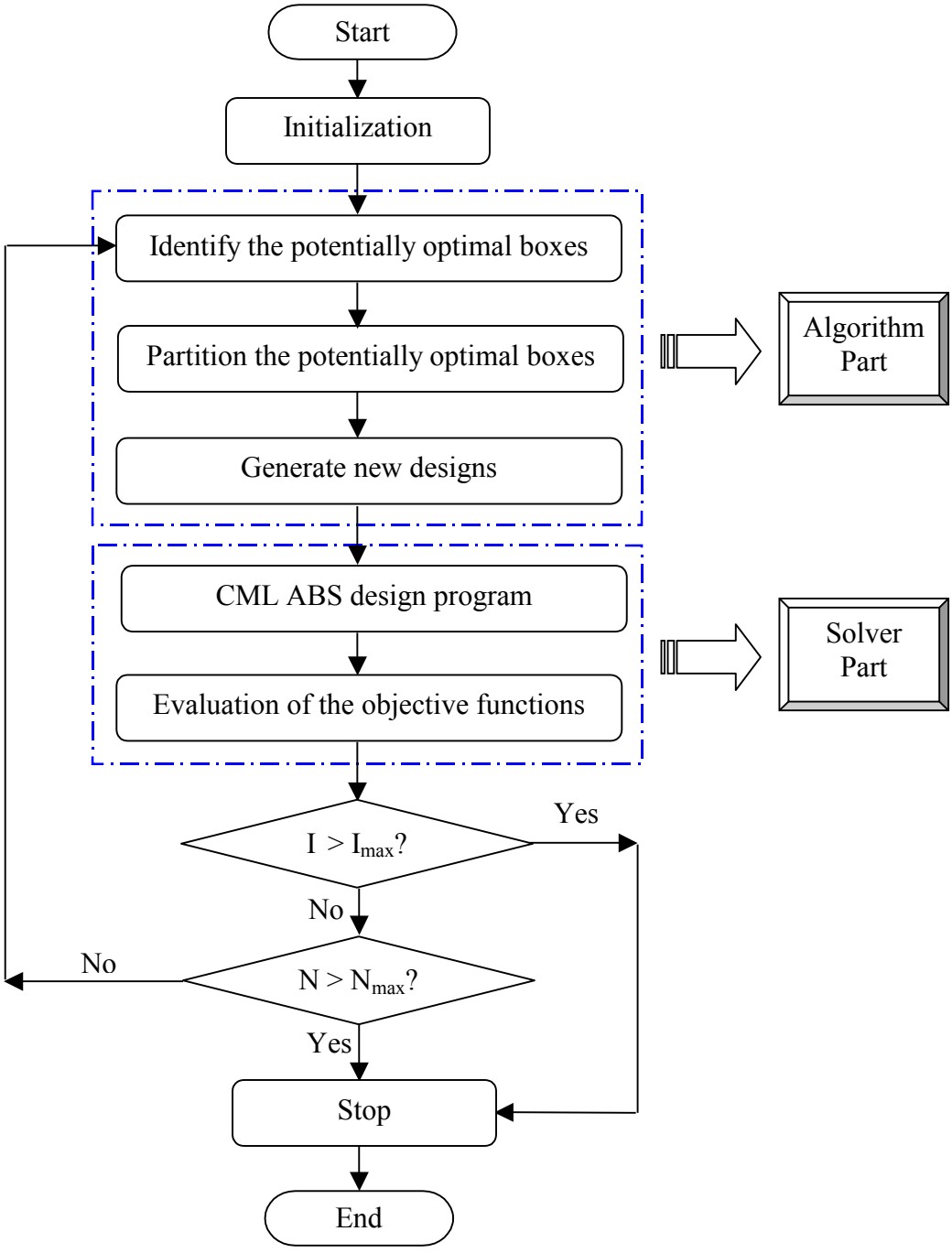

Fig. 2 Flow chart of the CML optimization program version 3.0

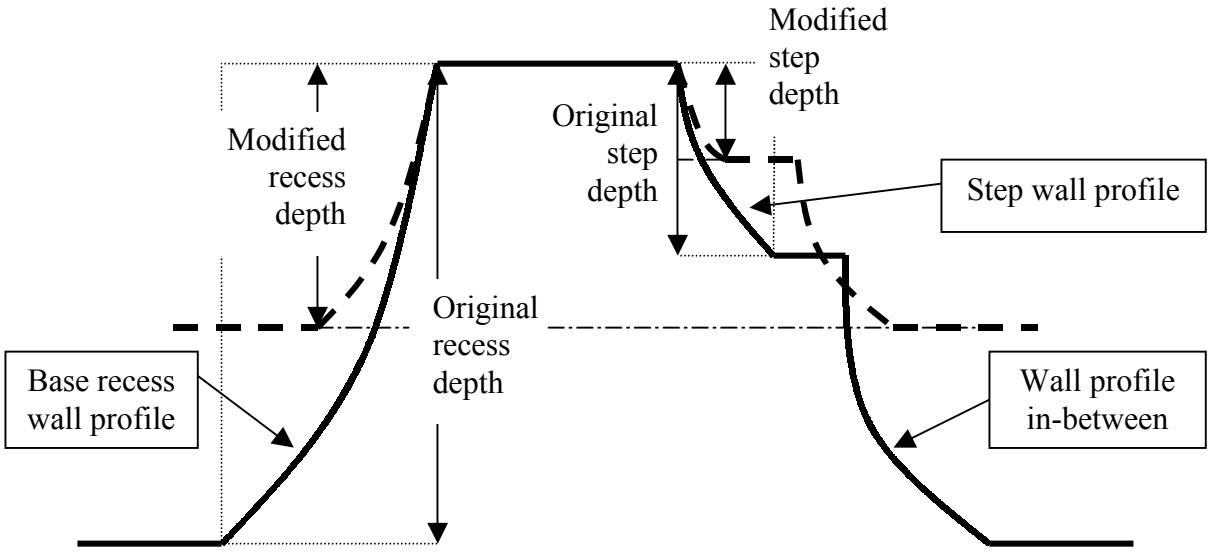

Fig. 3 Modification of wall profiles in proportional mode

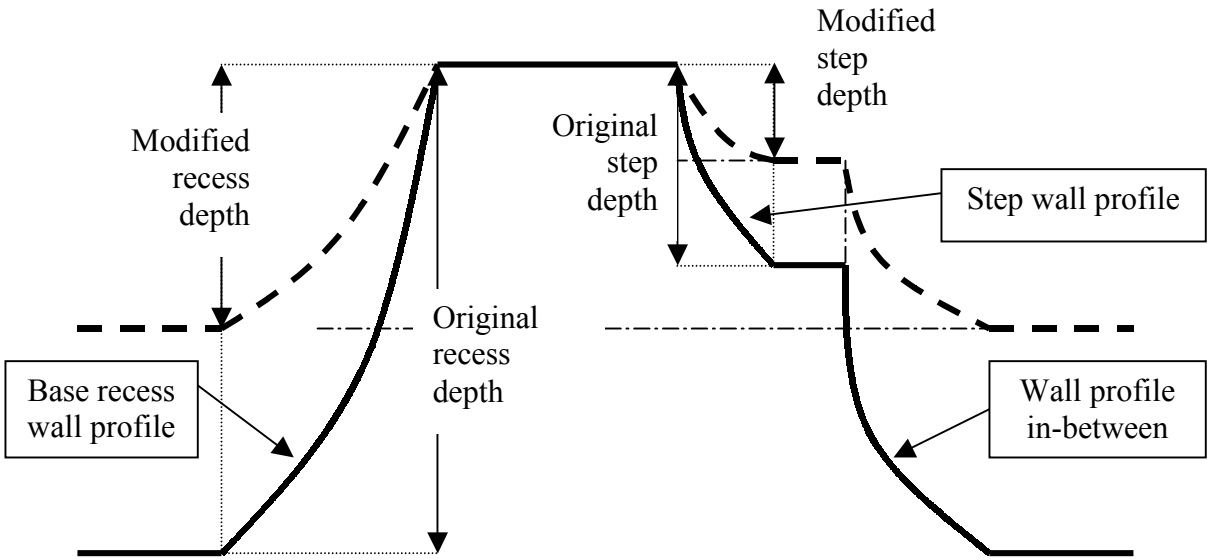

Fig. 4 Modification of wall profiles in fixed normal distance mode

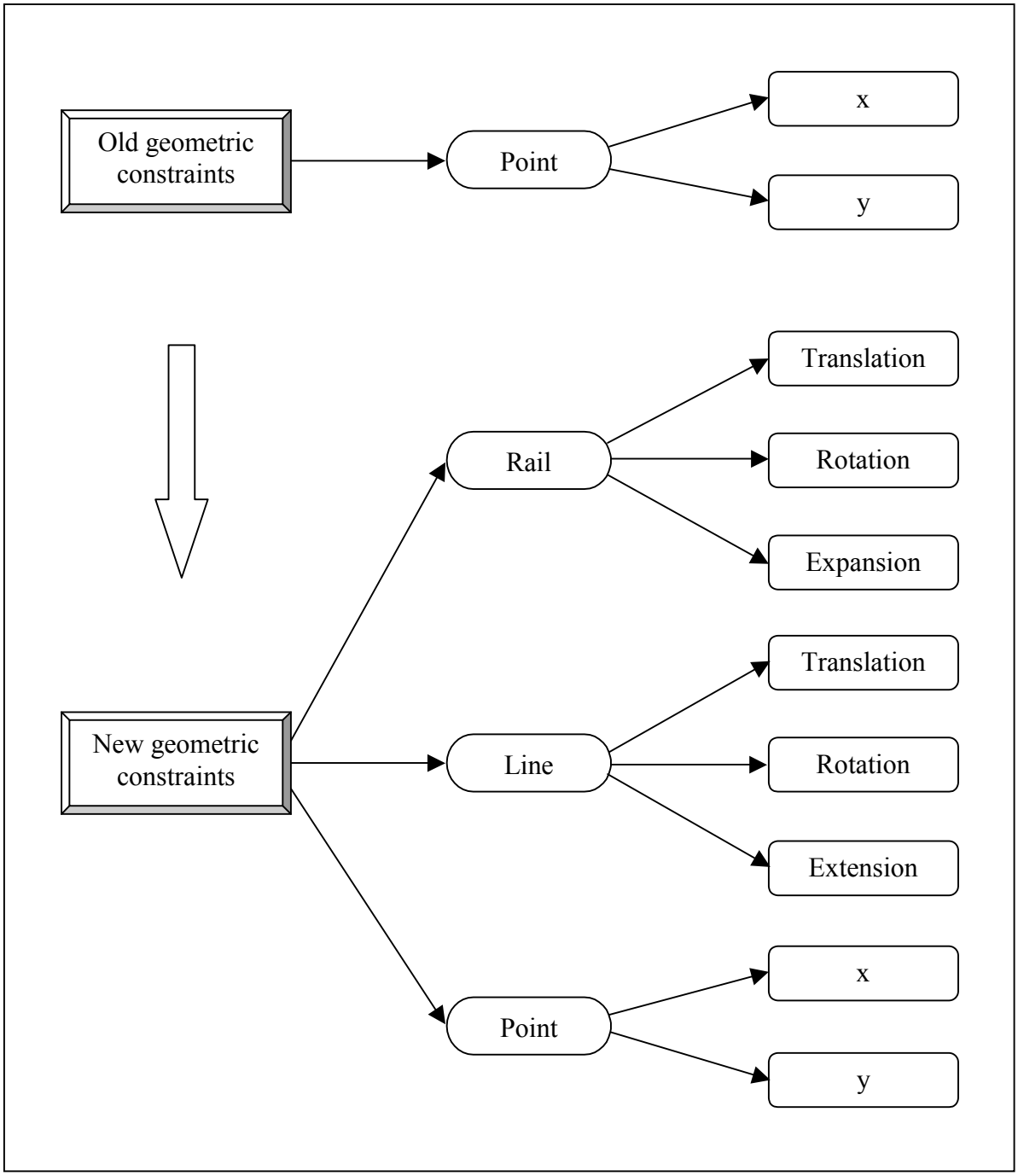

Fig. 5 Comparison between the old and the new geometric constraints

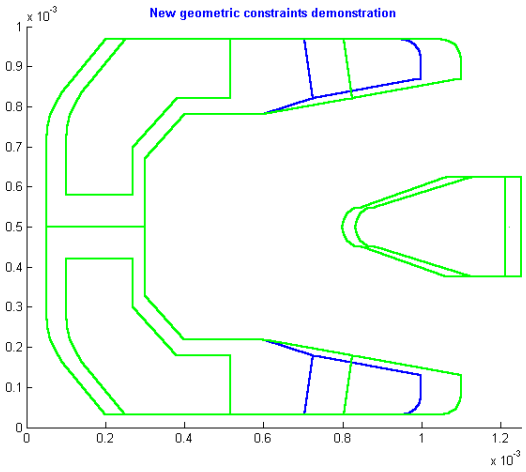

Fig. 6 Rail translation

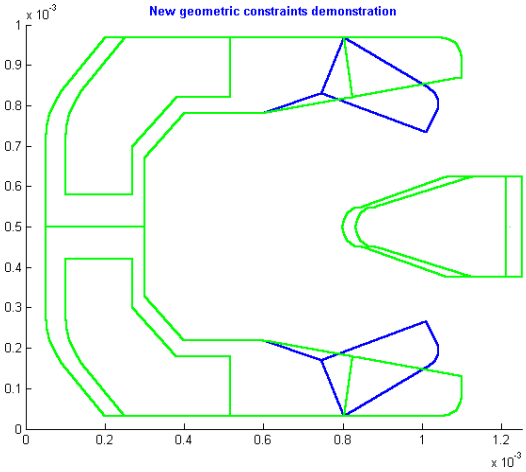

Fig. 7 Rail rotation

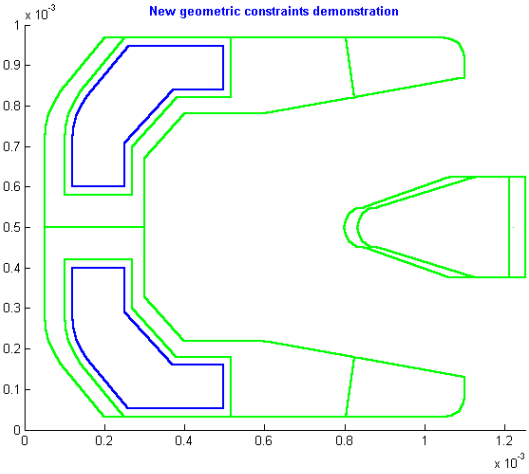

Fig. 8 Rail expansion

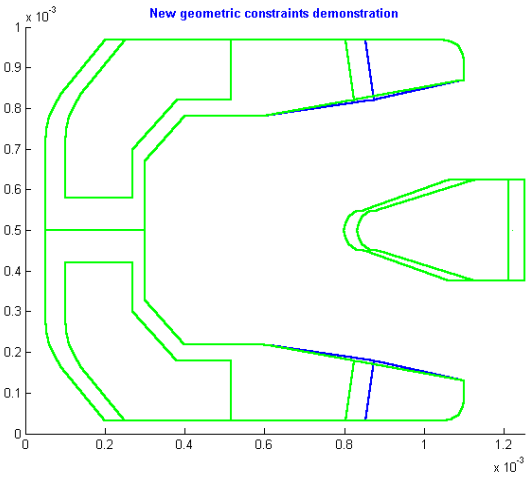

Fig. 9 Line translation

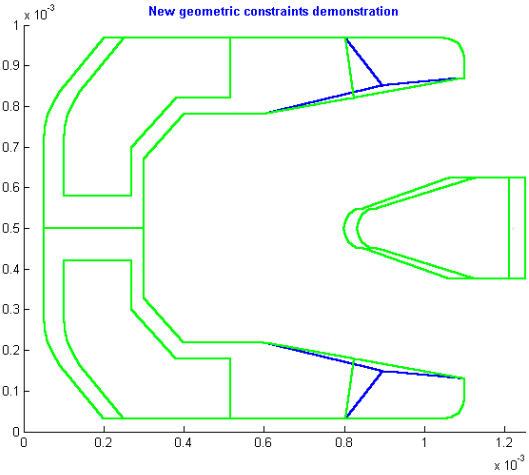

Fig. 10 Line rotation

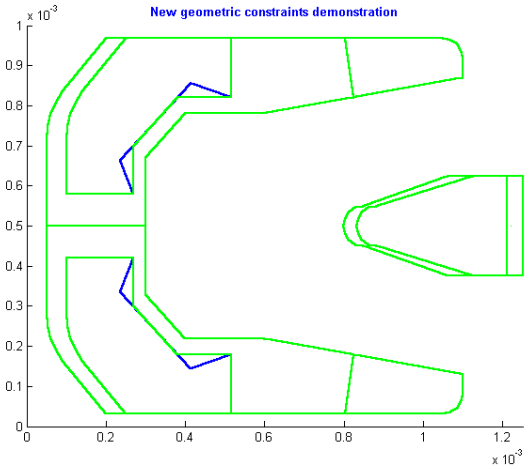

Fig. 11 Line extension

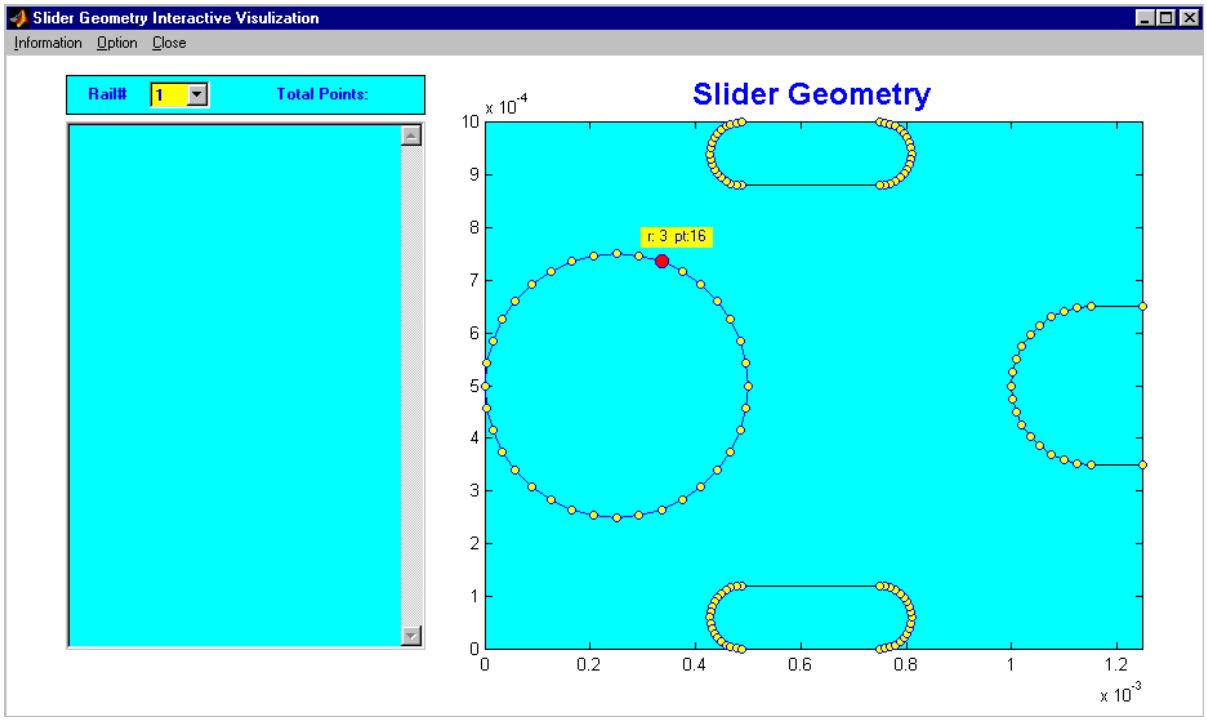

Fig. 12 Demonstration of the Slider Geometry Interactive Visualization GUI (1)

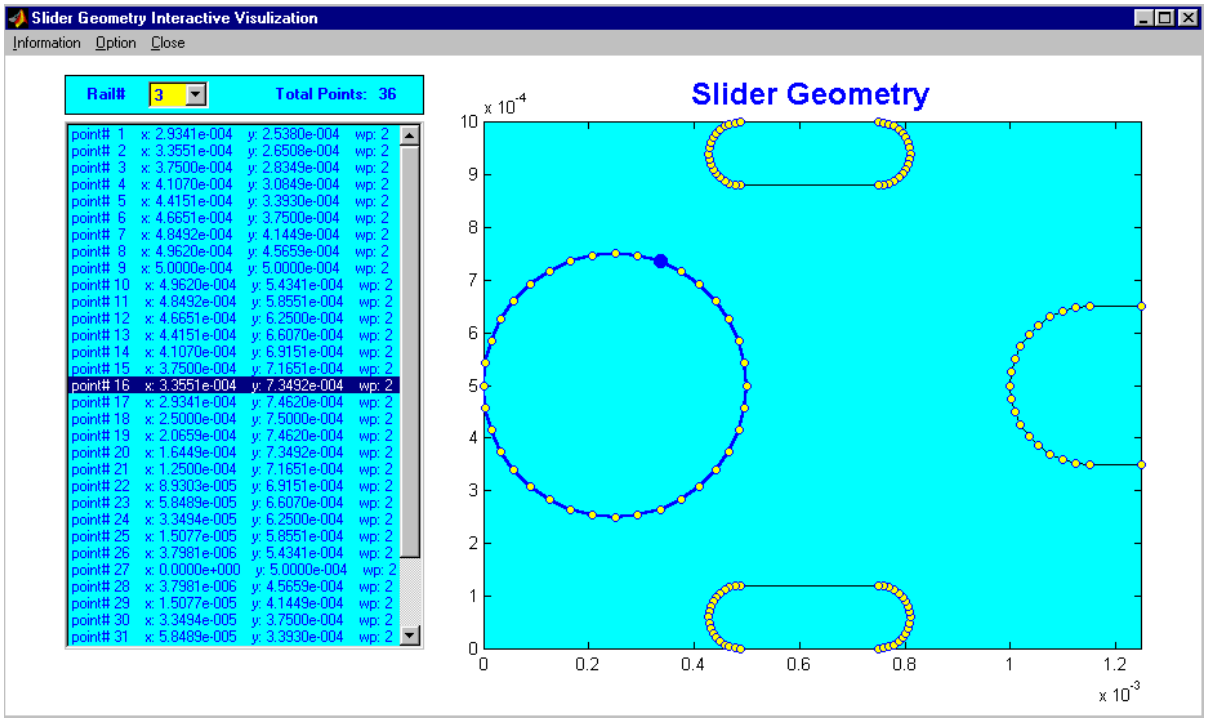

Fig. 13 Demonstration of the Slider Geometry Interactive Visualization GUI (2)

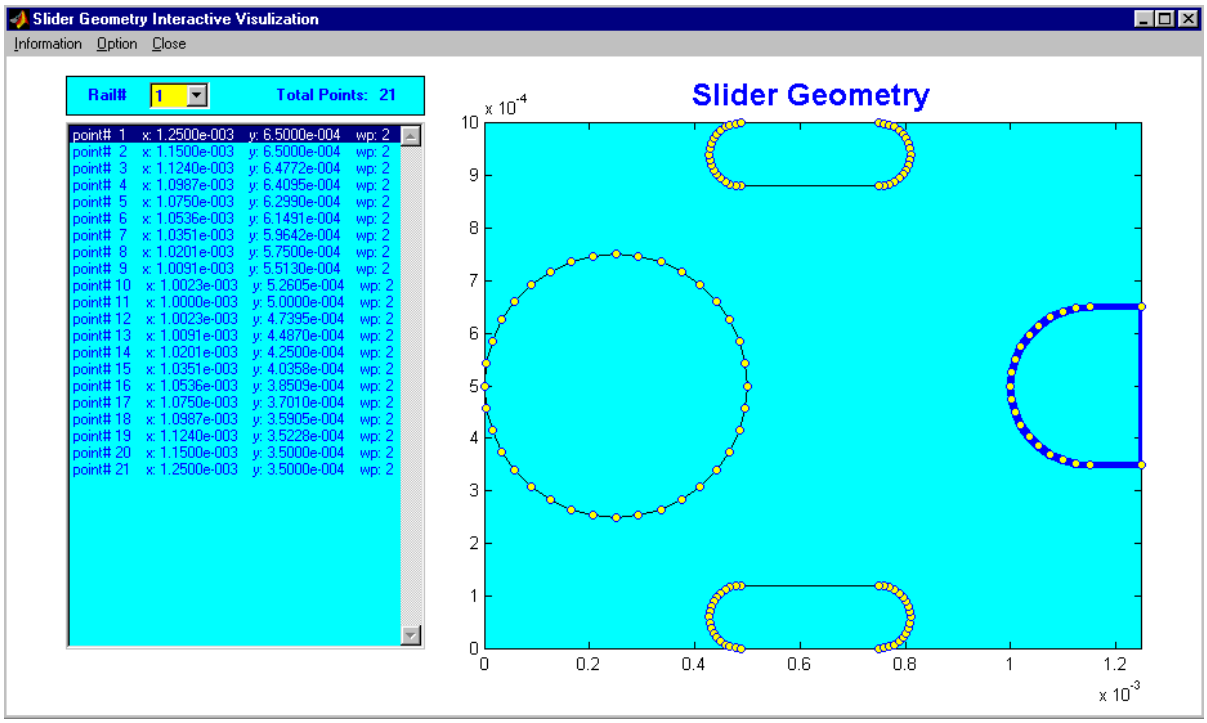

Fig. 14 Demonstration of the Slider Geometry Interactive Visualization GUI (3)

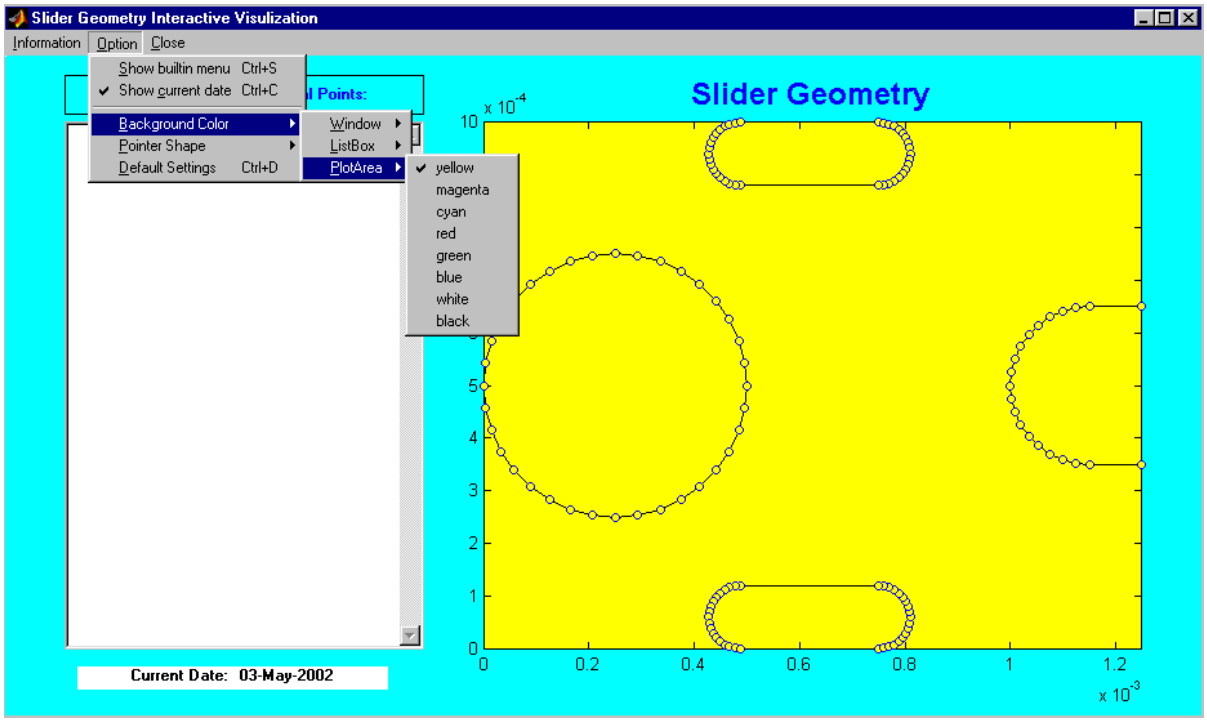

Fig. 15 Demonstration of the Slider Geometry Interactive Visualization GUI (4)

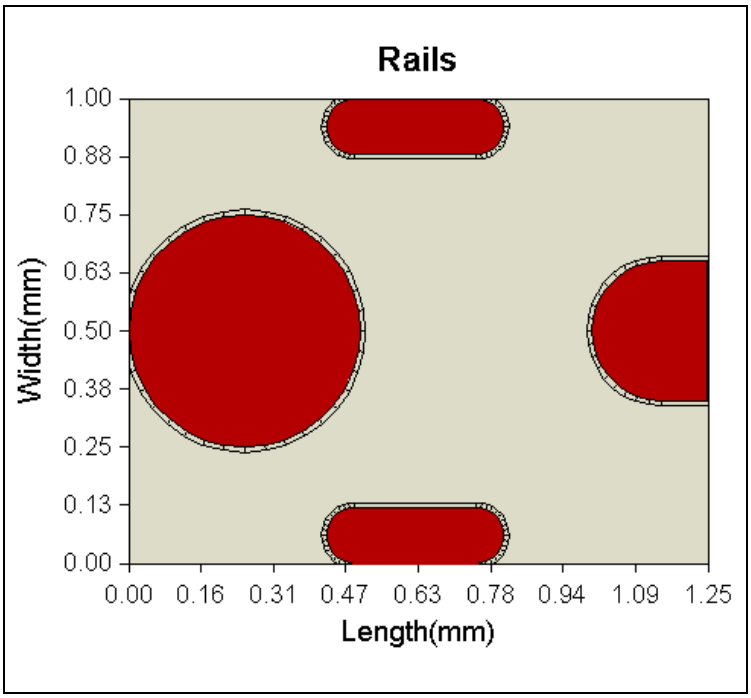

Fig. 16 Rail shape of the initial “Enterprise” slider design

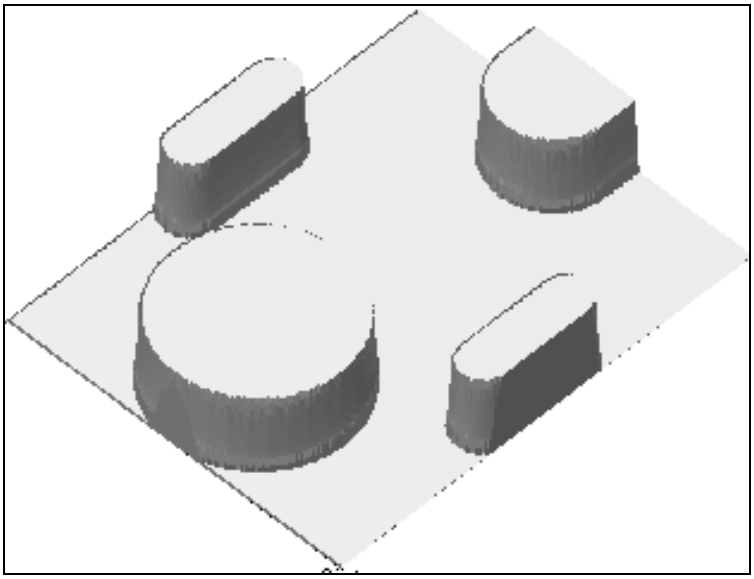

Fig. 17 3-D rail shape of the initial “Enterprise” slider design

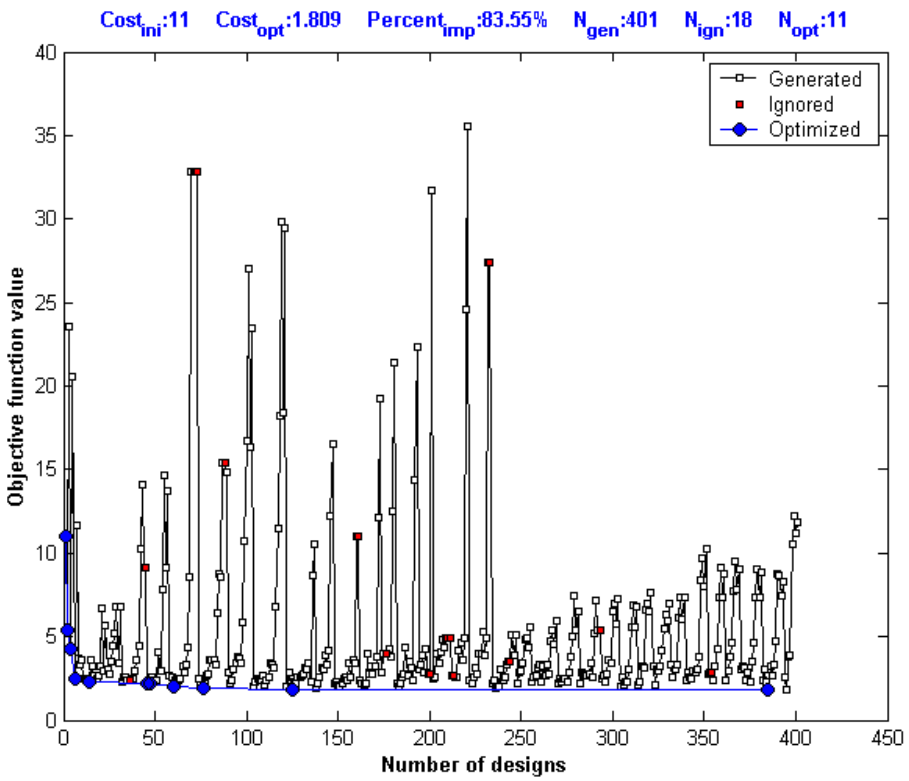

Fig. 18 Variation of the objective function value for the 2-D example case

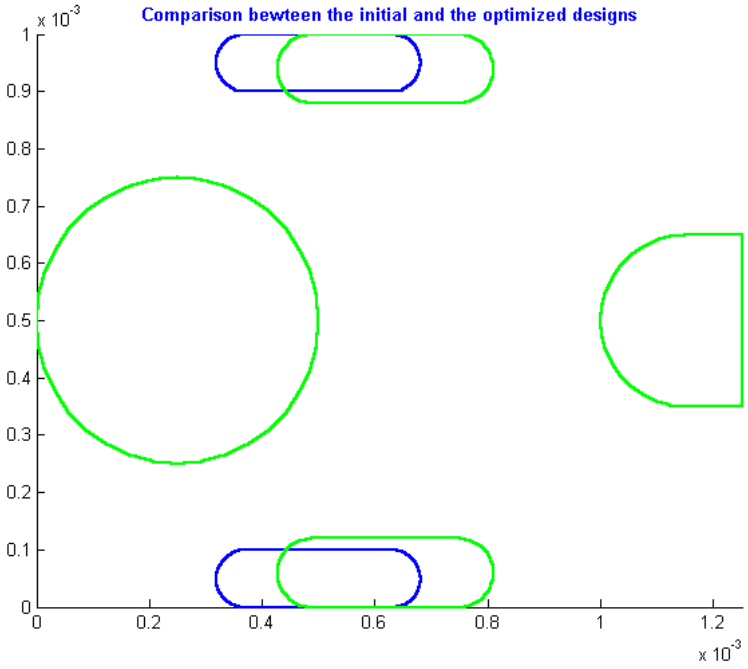

Fig. 19 Comparison of the initial and optimized designs for the 2-D example case

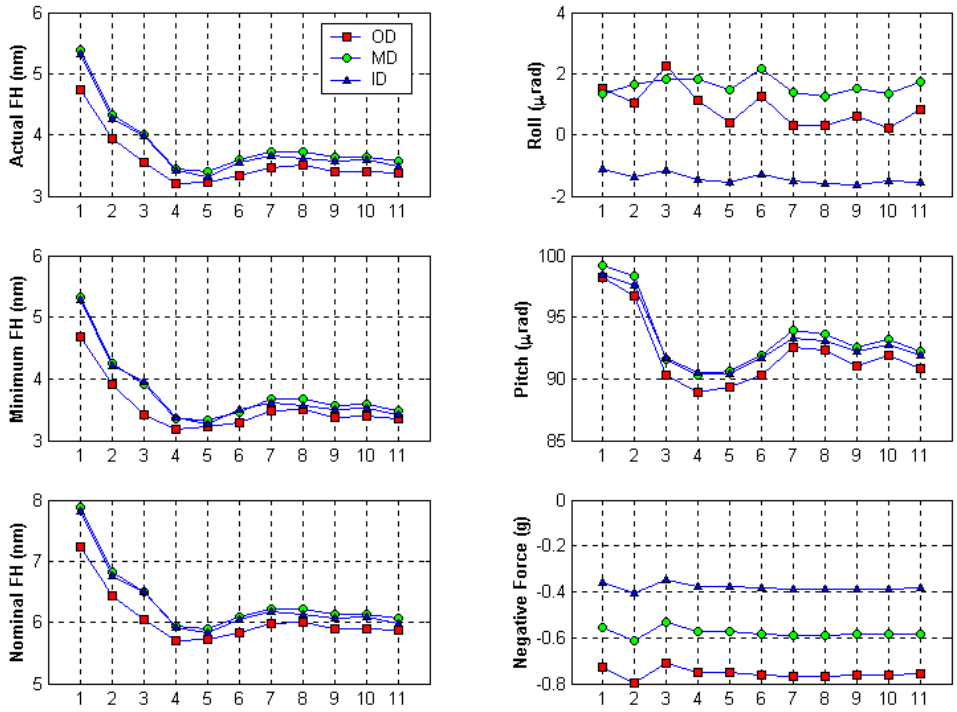

Fig. 20 Variations in the slider performance parameters for the 2-D example case

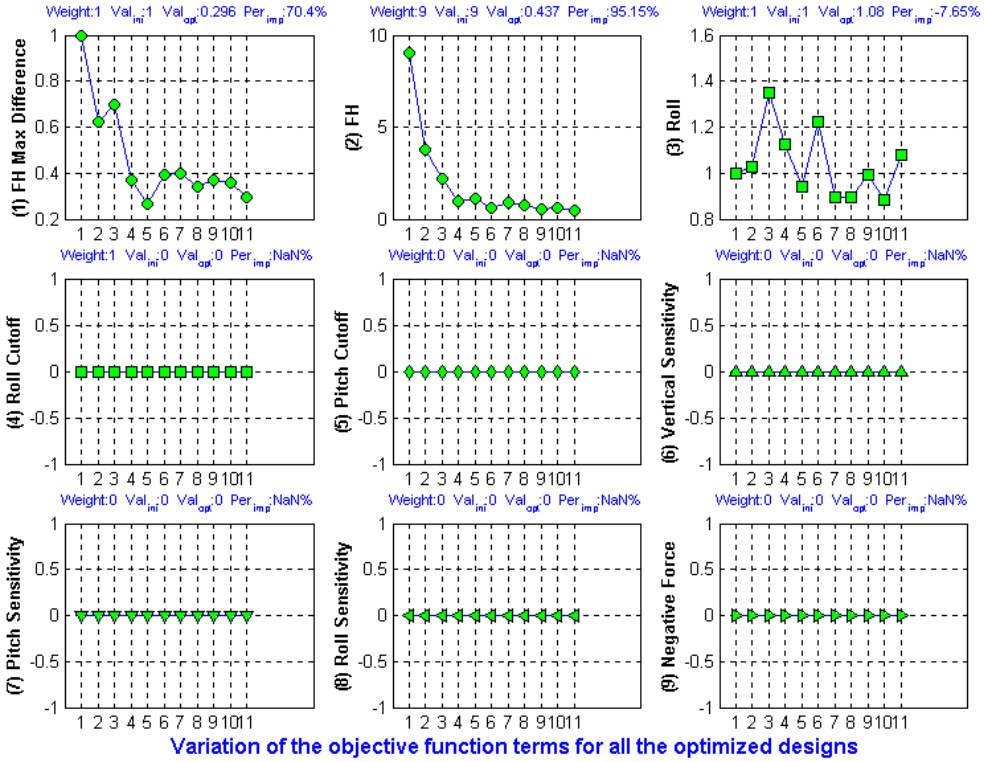

Fig. 21 Variations in the objective function terms for the 2-D example case