Modifications to the DIRECT Algorithm

Hong Zhu and D. B. Bogy

Computer Mechanics Laboratory
Department of Mechanical Engineering
University of California
Berkeley, CA 94720

ABSTRACT

In this report, we discuss two modifications to the DIRECT algorithm: one to handle tolerance (minimum side lengths) and one to deal with hidden constraints. Some numerical experiments were carried out using these modifications and then the modified DIRECT algorithm was applied to slider ABS optimization. The results show that these two modifications can improve the efficiency of the DIRECT algorithm and that they also make the slider ABS optimization program more flexible.

1. INTRODUCTION

The DIRECT algorithm is a global deterministic algorithm developed by Jones et al. in 1993 [5]. DIRECT is guaranteed to converge [4] and has a very fast convergence rate. Thus, it can find the global minimum points very quickly as compared with other algorithms [5] [6].

We presented the details of the DIRECT algorithm, the results of numerical experiments, and its application to the slider Air Bearing Surface (ABS) optimization in a previous CML technical report [1]. The results verify the very fast convergence rate of the DIRECT algorithm and show that it is suitable for slider ABS optimization.

We have also presented three locally biased variations of the standard DIRECT algorithm in another CML technical report [2]. Our investigations show that 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.

For the slider ABS optimization, the evaluation of an ABS design is a time-consuming process. Therefore, it is desirable to further increase the efficiency of the DIRECT algorithm in order to shorten the computational time for the optimization process needed to find the global optimized ABS design.

Here, we report on two modifications to the standard DIRECT algorithm. After reviewing the standard DIRECT algorithm, we present the modifications, and discuss our numerical experimental results. Then we apply these modifications to the slider ABS optimization. Finally, we present results for several slider ABS optimization test cases and draw our conclusions.

2. NUMERICAL METHOD

2.1 Standard DIRECT algorithm

DIRECT is an acronym for **DI**viding **RECT**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 which can be used without knowing the Lipschitz constant. DIRECT is designed to solve problems subjected to bounded constraints.

Without loss of generality, in the DIRECT algorithm we always assume that every variable has a lower bound of 0 and an upper bound of 1, since we can always normalize the variables to this interval. Thus, the search space is an n-dimensional unit hyper-cube. There are two main components in the DIRECT algorithm: one is the dividing strategy for the hyper-cubes and the hyper-rectangles (they are referred to as "boxes" in our reports); the other is the selection of the potentially optimal boxes. We briefly introduce them in sections 2.1.1 and 2.1.2. For more details, please see Ref. [1].

2.1.1 Dividing strategy

The dividing strategy of the DIRECT algorithm for the hyper-cubes and the hyper-rectangles is as follows:

A. Partition of a hyper-cube
Assume m is the center point a hyper-cube. We sample the points $m \pm \delta e_i$, where δ equals 1/3 of the side length of the cube and e_i is the i-th Euclidean base-vector. We define $s_i = \min \{ f(m-\delta e_i), f(m+\delta e_i) \}$, then the partition will be in the order given by s_i, starting with the lowest s_i. This means the hyper-cube is first partitioned along the direction with the lowest s_i, then the remaining field is partitioned along the direction of the second lowest s_i, and so on until the hyper-cube is partitioned in all directions.

B. Partition of a hyper-rectangle
Hyper-rectangles are only partitioned along their longest sides. This partition strategy ensures a reduction in the maximal side length of a hyper-rectangle.

2.1.2 Selection of potentially optimal boxes

Let m_i denote the center point of the i-th hyper-rectangle, and d_i the distance from the center point to the vertices. Then the potentially optimal boxes are defined as follows:

Definition 2.1 *Let $\varepsilon > 0$ be a positive constant and f_{min} be the current lowest function value. A hyper-rectangle (box) j is said to be potentially optimal if there exists some rate-of-change constant $\tilde{K} > 0$ such that*

$$f(m_j) - \tilde{K} d_j \le f(m_i) - \tilde{K} d_i \quad \text{for any } i \tag{2.1}$$

$$f(m_j) - \tilde{K} d_j \le f_{min} - \varepsilon |f_{min}| \tag{2.2}$$

2.2 Modifications of the standard DIRECT algorithm

2.2.1 Tolerance

It is clear from section 2.1 that the standard DIRECT algorithm almost always identifies the box containing the minimum point as the potentially optimal box, as illustrated in Fig. 1.

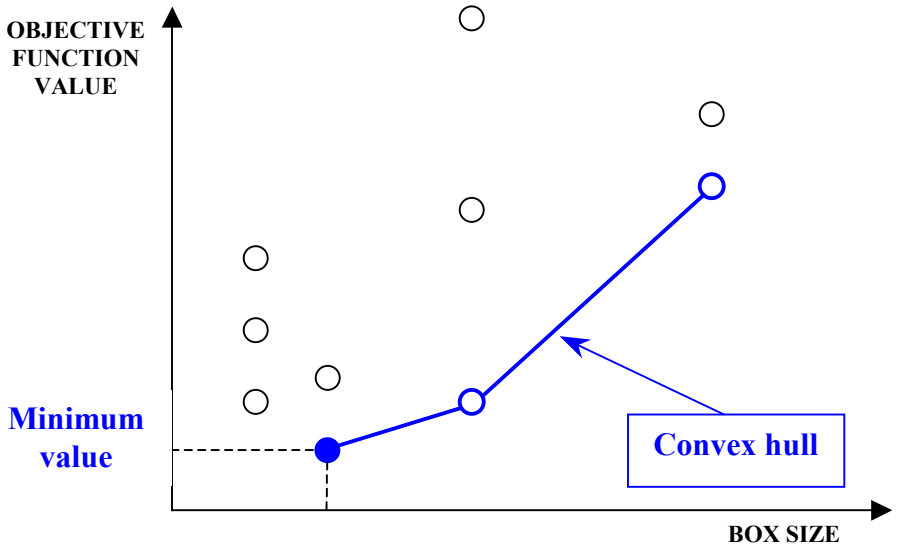


Fig. 1 Illustration of the convex hull and the potentially optimal boxes

In fact, imagine that we draw a line with a positive slope K below all the data points and then move it upward. If K is positive but small enough, the first data point that the line intersects would be the point with the lowest objective function value. [1] The only exception is when there is another data point that has a very similar small value to the actual minimum point and the inequality (2.2) is satisfied. In that case, the minimum point will not be chosen as the potentially optimal point.

Since the DIRECT algorithm usually partitions the box containing the minimum point at each iteration step, it is possible that the box containing the best point will become smaller and smaller as the optimization process goes on. This accounts for the fast convergence property of the DIRECT algorithm.

However, in applications, we do not want the partitioning to continue once the size of the box containing the best point shrinks to a certain level. The main reason is that since the practical head manufacturing process has a limitation on the processing resolution, it cannot differentiate very minor differences among sample designs. Another possible reason is that, instead of pursuing the "perfectly" optimized design, we might just need a certain level of resolution from the engineering point of view. Therefore, we are introducing the concept of "tolerance", that is, the minimal side lengths for all the sides of all the boxes. Each side may have a different tolerance.

When a box is to be partitioned, all of its side lengths will be checked to see if they are greater than the tolerances defined. If a certain side length is greater than the tolerance value prescribed, that side will be partitioned; otherwise, it will not be partitioned.

It follows that the introduction of the tolerance can prevent the DIRECT algorithm from becoming too local around the current best point, wasting valuable function evaluation time. Therefore, the algorithm can search more globally on a fixed number of function evaluations. Consequently, this modification will improve the efficiency of the standard DIRECT algorithm.

2.2.2 Hidden Constraints

As far as the DIRECT algorithm is concerned, the search space is a multi-dimensional unit hyper-cube. If no hidden constraints exist in the search

space, every sample point in the search space can be evaluated and has a definite objective function value.

However, if there are hidden constraints in the search space, things will be quite different. Let's call the sample points that satisfy the hidden constraints the infeasible points. Similarly, the sample points that do not satisfy the hidden constraints are referred as the feasible points.

It is impossible to evaluate the infeasible points such that they return definite objective function values. For the DIRECT algorithm, however, all the sample points must have values so that the algorithm can find the potentially optimal ones. Also notice that we cannot simply discard those infeasible points. The reason is simple: if a midpoint of a large box is an infeasible point, discard that point and its box will result in the loss of all other possible feasible points within that large box.

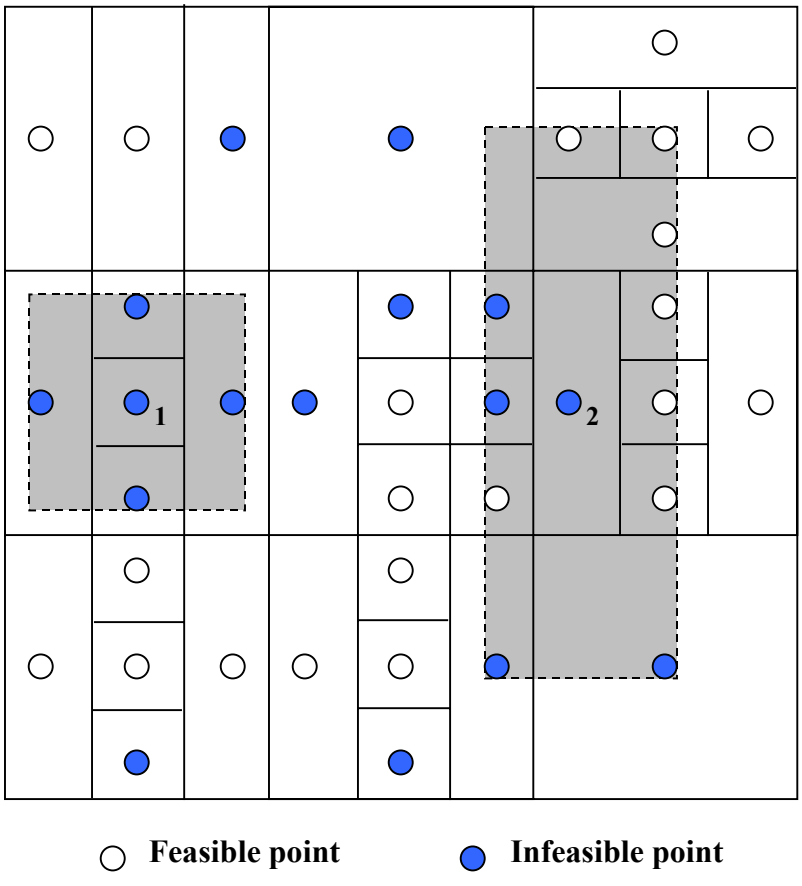


Fig. 2 Illustration of the strategy to handle hidden constraints

In order to handle these hidden constraints, we use the method proposed by Gablonsky[7]. This method provides a pseudo-value for the infeasible point depending on its neighboring points' values.

Figure 2 demonstrates how we deal with the infeasible points. The square area is the search space for a 2-D problem. The solid round dots represent the infeasible points, which satisfy the hidden constraints. The hollow round dots represent the feasible points. All the feasible points have definite objective function values.

To determine the pseudo-value an infeasible point should have, we take the following steps:

1. Double the size of the box which contains the infeasible point. (The double size box is shown as the shadow box with dashed side lines in Fig. 2.)
2. Check the status of all the sample points inside the double size box (including those points on the boundary of the box) except the infeasible point being considered.
3. If all the sample points inside that enlarged box are infeasible points (for example, the case of infeasible point 1 in Fig. 2), then that infeasible point is marked as a "real" infeasible point and is given a pseudo-value as: $f_{max} + 1$, where f_{max} is the current maximum objective function value of all the feasible points.
4. If the sample points inside that enlarged box are not all infeasible points, (for example, the case of infeasible point 2 in Fig. 2), then the infeasible point is given a pseudo-value as: $f_{min} + \varepsilon |f_{min}|$, where f_{min} is the minimum value of all feasible points inside the enlarged box, and ε is a small prescribed value. We set ε as 10^{-6} in our optimizations.

Note that the pseudo-values of the infeasible points may change for every iteration.

The above strategy ensures that, for the boxes of the same size, the ones containing "real" infeasible points will get partitioned last, because they have the highest (pseudo) objective function value. This is good because the boxes containing feasible points are potentially better choices for partition than the ones containing "real" infeasible points.

The above strategy also ensures that near some feasible points, even the boxes containing the infeasible points still have a good possibility of being partitioned very quickly. Thus the algorithm maintains its high convergence rate while dealing with the hidden constraints.

3. NUMERICAL EXPERIMENTS

3.1 Testing function with one minimum point

For a simple demonstration of the numerical experimental results, we chose 2-D testing functions. The testing function used here has only one minimum point. It is defined as follows:

$$F(x_1, x_2) = (x_1 - 0.4)^2 + (x_2 - 0.2)^2 .$$

Where $x_1 \in [0,1]$, $x_2 \in [0,1]$. It has a minimum point at (0.4, 0.2) and the minimum value is 0.

Figure 3 shows contour lines of the 2-D function. Figure 4 shows its surface shape.

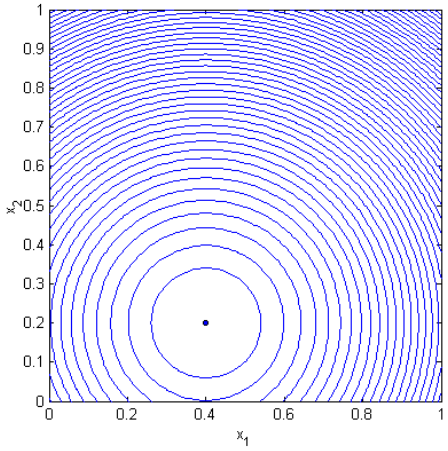


Fig. 3 Contour lines

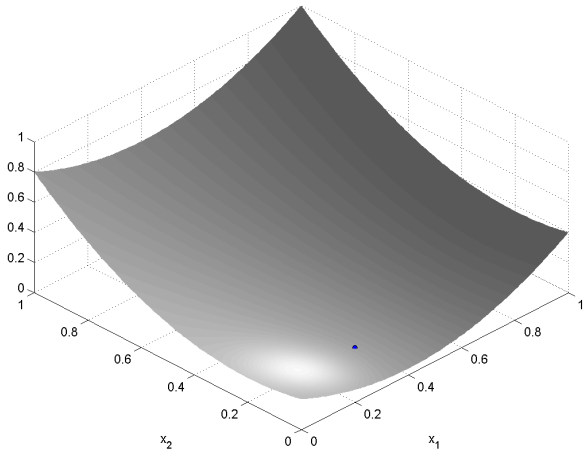


Fig. 4 Surface shape

Figures 5 ~ 8 show the optimization results after 500 function evaluations. In Fig. 5 tolerance is not used, whereas in Fig. 7 the tolerance is set at 0.01

for both of the independent variables x_1 and x_2. Figures 6 and 8 show the local zoom-ins around the best point for Figs. 5 and 7, respectively.

In Figs. 5 ~ 8, X and Y represent the variables x_1 and x_2, respectively. The tiny dots inside the boxes are the sample points generated by the DIRECT algorithm. The circle represents the best point found by the algorithm.

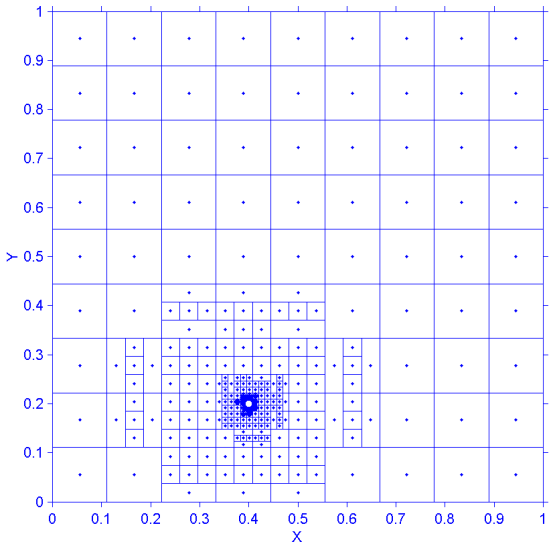


Fig. 5 Results with no-tolerance case

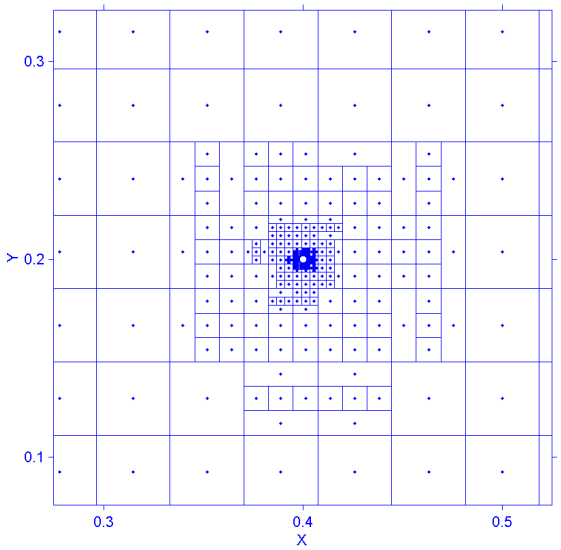


Fig. 6 Local zoom-in

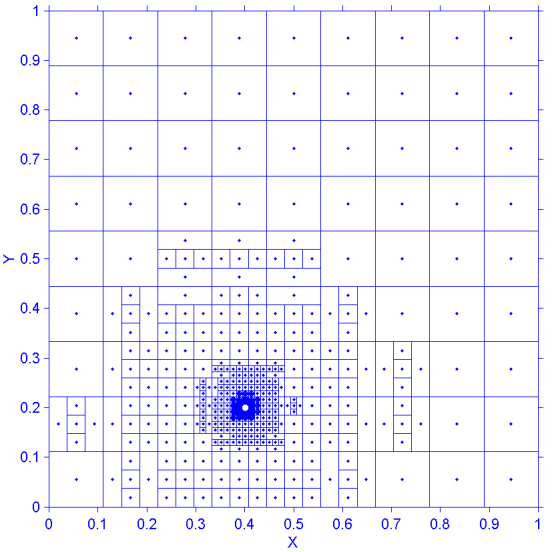


Fig. 7 Results with tolerance case

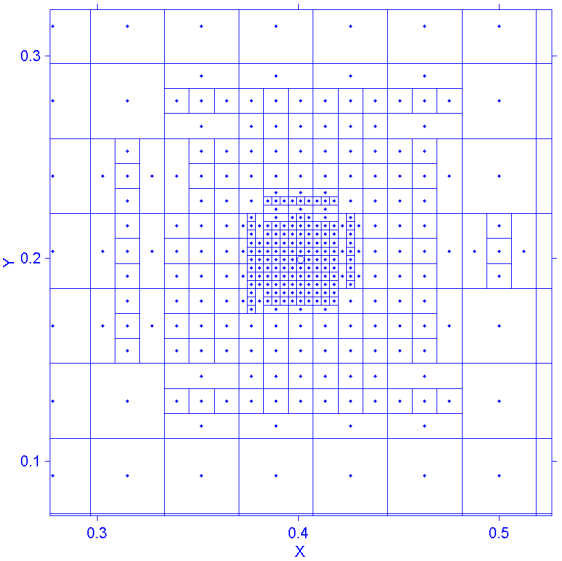


Fig. 8 Local zoom-in

For the no-tolerance case, the best point found after 500 function evaluations is (3.999983E-01, 2.000006E-01). The minimum value is 3.186636E-12. The side lengths of the box containing the best point are L_x=1.693509E-05 and L_y=5.645029E-06. For the case considering tolerance, the best point found after 500 function evaluations is (4.012346E-01, 1.995885E-01). The minimum value is 1.693509E-06. The side lengths of the box containing the best point are L_x=4.115226E-03 and L_y=4.115226E-03. Figure 8 shows that when both sides of a box are smaller than the tolerance prescribed, which is 0.01 in this case, that box will no longer be partitioned.

We can also define different tolerance values for different independent variables. In the following case, we set the tolerance for x_1 as 0.15 and the tolerance for x_2 as 0.05. After 100 function evaluations, i.e., after 100 sample points are generated, the best point found is (3.888889E-01, 2.037037E-01). The side lengths of the box containing the best point are L_x=1.111111E-01 and L_y=3.703704E-02. The optimization results are shown in Fig. 9.

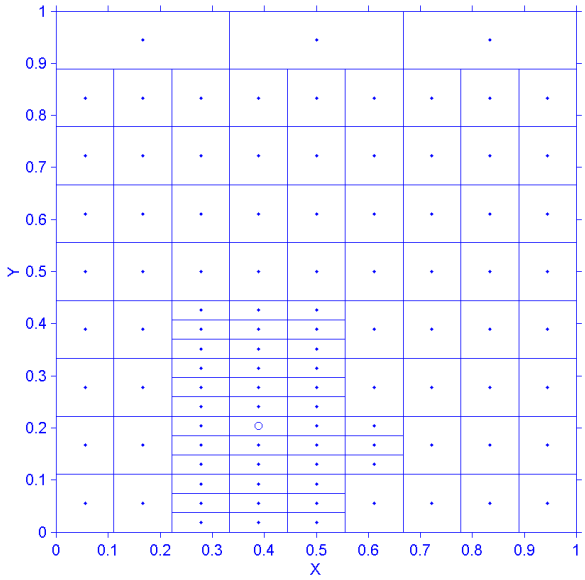


Fig. 9 Optimization results for different tolerance values

It is clear from Fig. 9 that because variable x_2 (Y) has a smaller tolerance value (0.05) than that of the variable x_1 (X), which is 0.15, side x_2 gets more partitioning than does x_1 side. Because the box containing the best point has reached the tolerance limit, it will not be further partitioned in the subsequent optimization process. Because the testing function only has one

minimum point, the best point found at this stage will not change if the optimization process continues.

Next we introduce some hidden constraints for the testing function. The tolerance is set at 0.01 for both x_1 and x_2. The number of function evaluations is 3000. Figure 10 shows the optimization results without hidden constraints. Figures 12, 14, and 16 show the optimization results with various hidden constraints. Figures 11, 13, 15, 17 show local zoom-ins around the best points for Figs. 10, 12, 14, and 16, respectively.

The hidden constraints for Fig. 12 are:

$(x_1 - 0.35)^2 + (x_2 - 0.35)^2 \le 0.15^2$ or $x_2 \le x_1^2$.

The hidden constraints for Fig. 14 are:

$(x_1 - 0.4)^2 + (x_2 - 0.2)^2 \ge 0.1^2$ and $(x_1 - 0.4)^2 + (x_2 - 0.2)^2 \le 0.2^2$.

The hidden constraint for Fig. 16 is:

$(x_1 - 0.4)^2 + (x_2 - 0.2)^2 \le 0.1^2$.

In Fig. 12 the hidden constraint areas are the areas inside the circle and the area below the parabolic line. In Fig. 14 the hidden constraints areas are the areas between the two concentric circles. In Fig. 16 the hidden constraint area is inside the circle that centers at (0.4, 0.2). The sample points that satisfy the hidden constraints are referred as the infeasible points, and are not evaluated. The shadowed boxes are the ones containing infeasible points.

Figures 12 ~ 15 show that inside the hidden constraint areas, the regions that are adjacent to the best point get partitioned more often.

Note that there are infinite minimum points in Fig. 16. These minimum points are all on the circumference of the circle. The optimization results clearly show the clustering of the feasible points around that circle, which means that the algorithm found all the global minimum points. The infeasible points also show a similar clustering pattern around that circle. This is logical, because the algorithm is expected to search more intensively around the best points, for both feasible areas and infeasible areas.

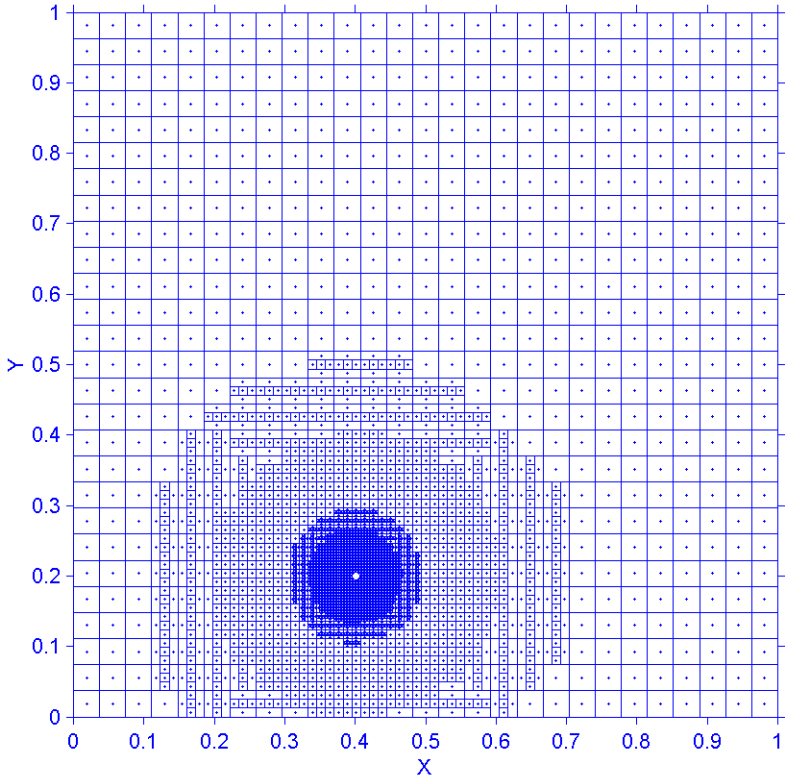


Fig. 10 Optimization results without hidden constraints

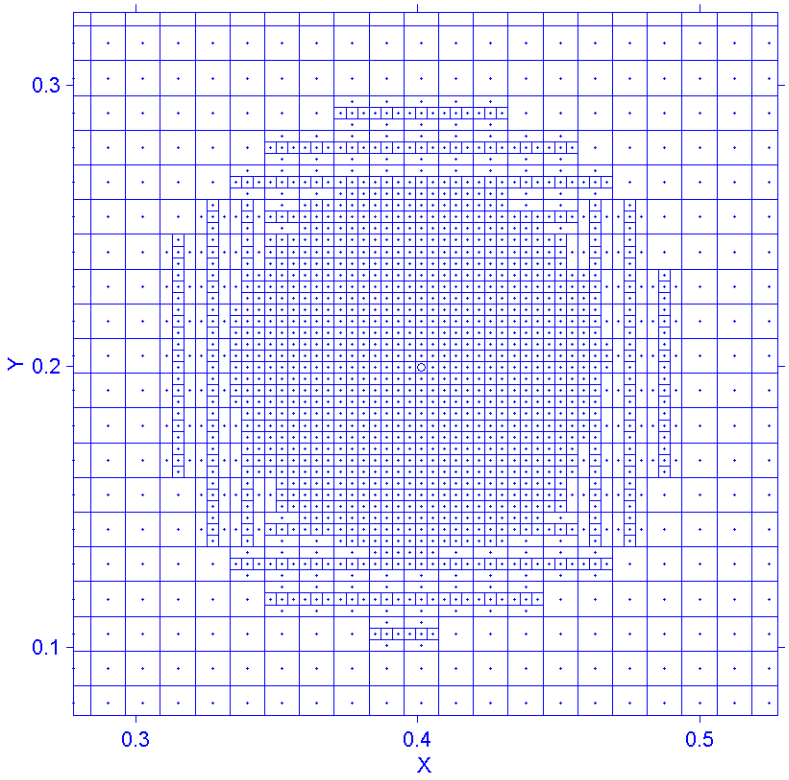


Fig. 11 Local zoom-in around the best point without hidden constraints

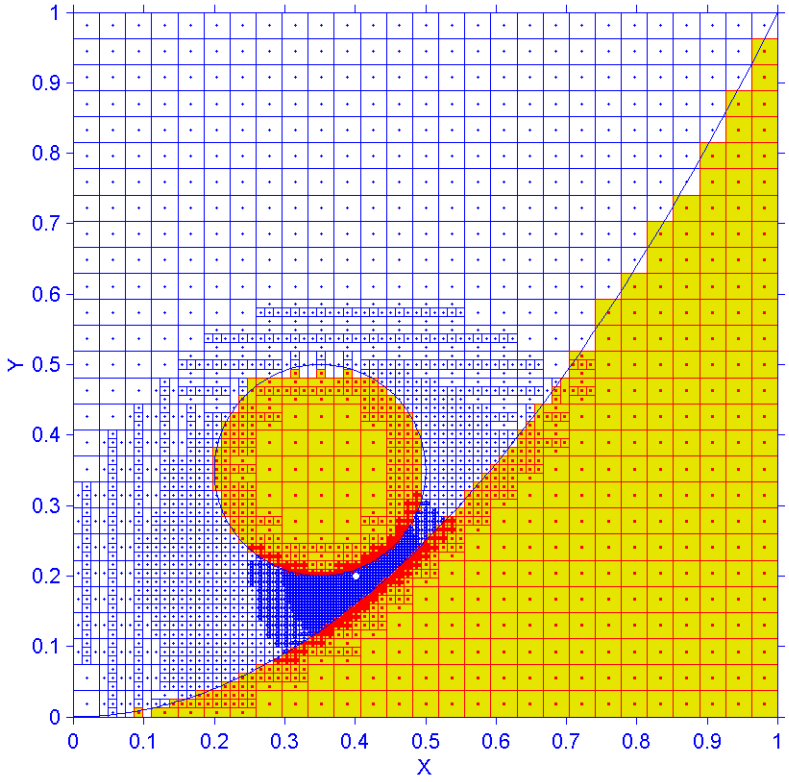


Fig. 12 Optimization results with hidden constraints, case 1

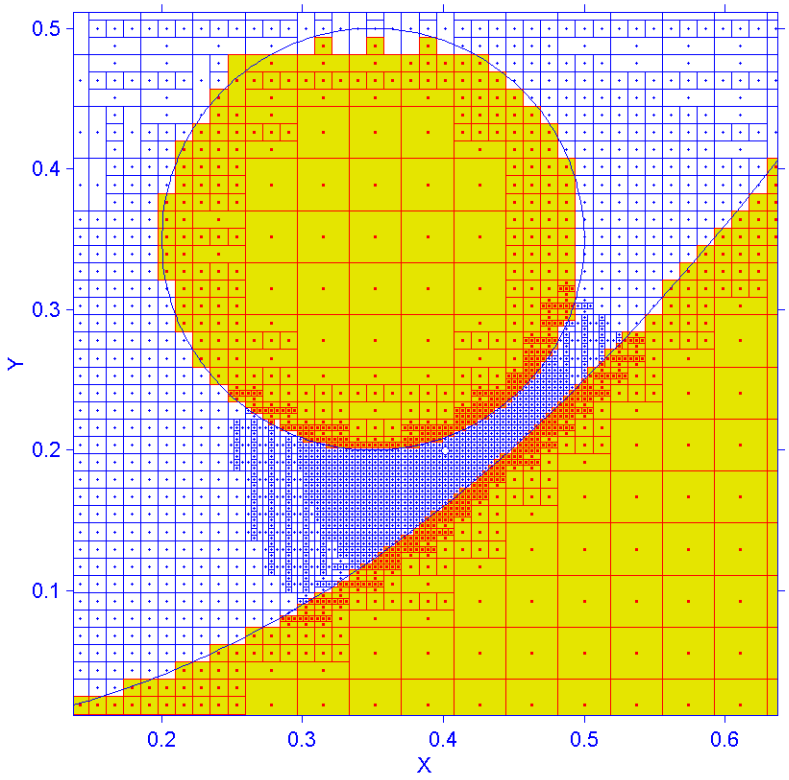


Fig. 13 Local zoom-in around the best point for case 1

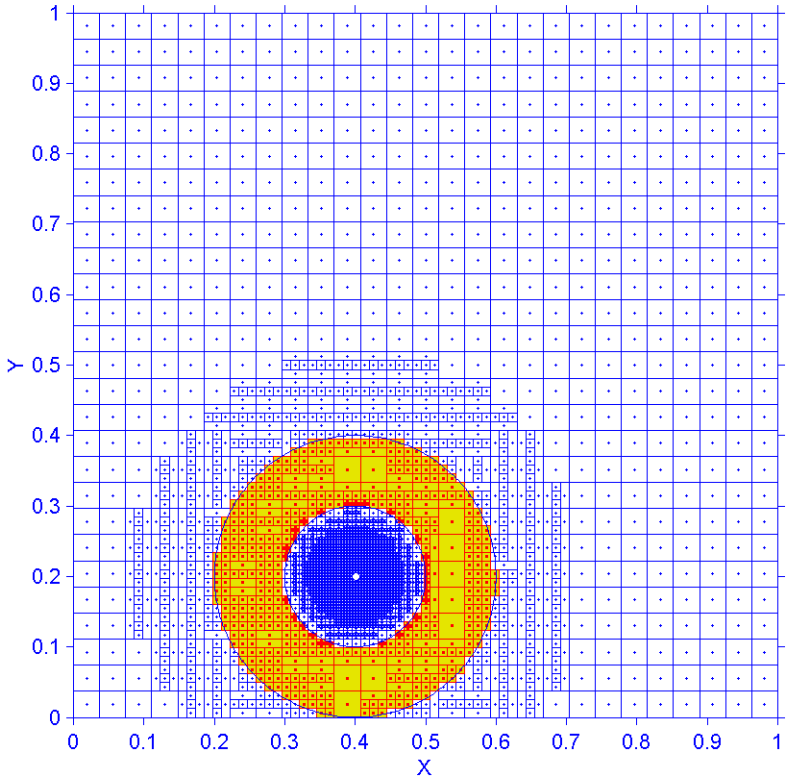


Fig. 14 Optimization results with hidden constraints, case 2

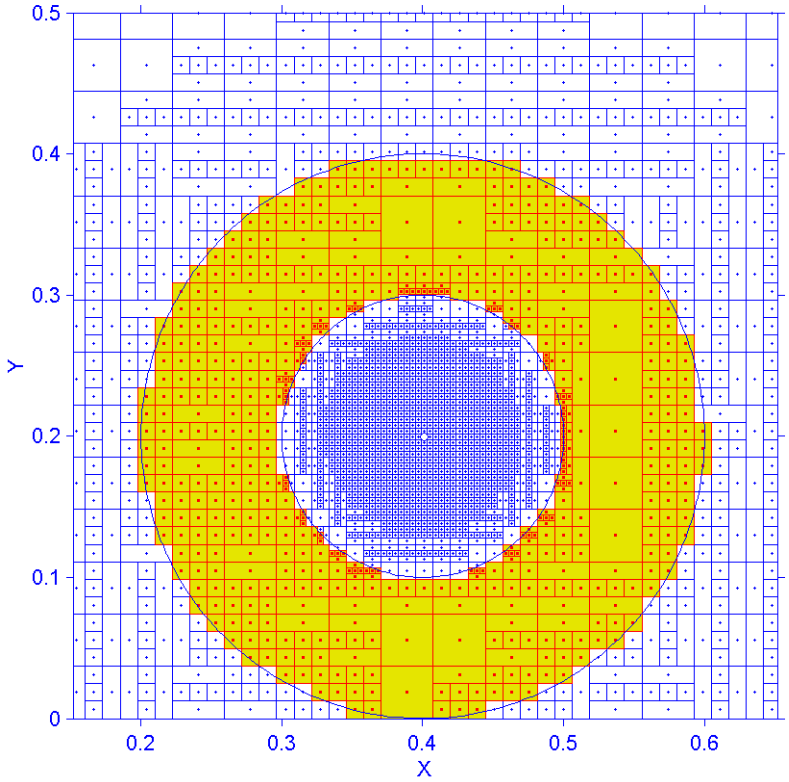


Fig. 15 Local zoom-in around the best point for case 2

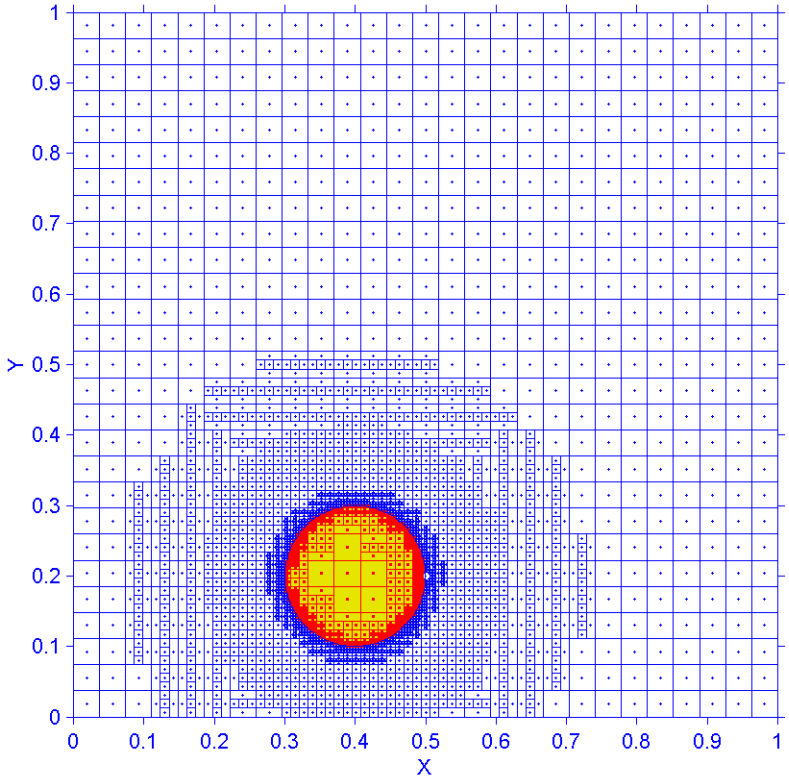


Fig. 16 Optimization results with hidden constraints, case 3

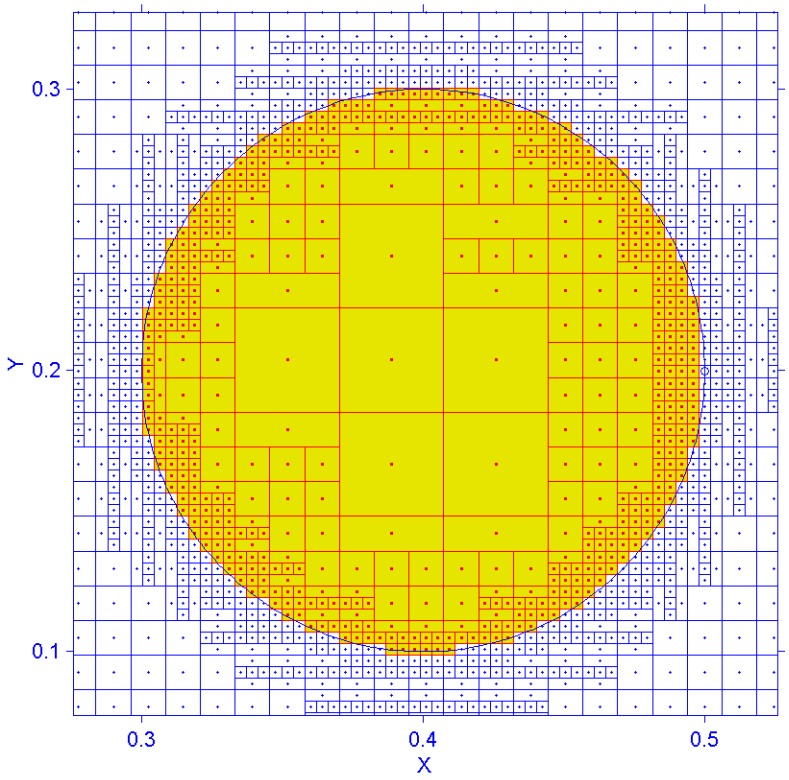


Fig. 17 Local zoom-in around the best point for case 3

3.2 Testing functions with multiple minimum points

The first testing function considered here is called the "six-hump" function, defined as:

$$F(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4,$$

where $x_1 \in [-2, 2]$, $x_2 \in [-1, 1]$. This function has two global minimum points and four other local minimum points. If we normalize the range of variables x_1 and x_2 into $[0,1]$, then its global minimum points are (0.52246, 0.14367) and (0.47754, 0.85633) and its global minimum is -1.03163.

The contour lines and the surface shape of the six-hump function are shown in Figs. 18 and 19, respectively. The round dots in Fig. 18 represent the global minimum points. The six "humps" can be clearly discerned from these two figures.

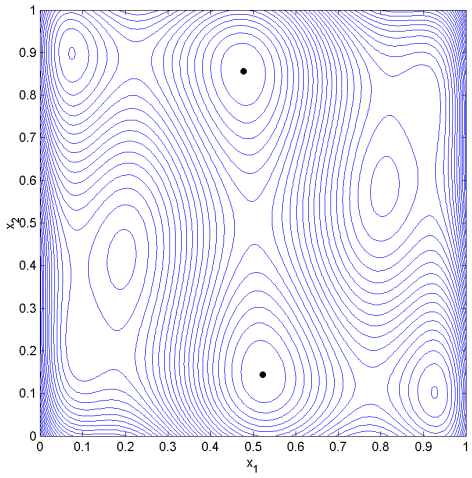


Fig. 18 Contour lines

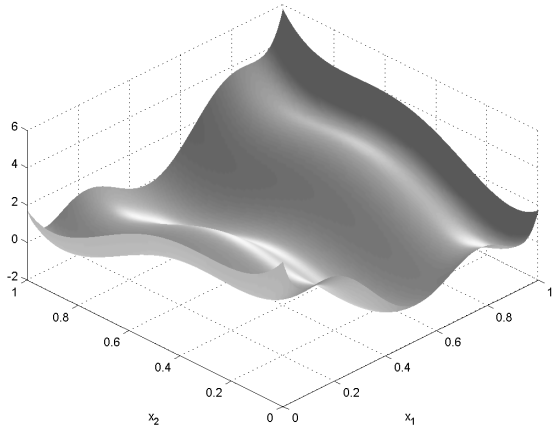


Fig. 19 Surface shape

Figures 20 and 21 show the optimization results with no tolerance and with 0.02 tolerance for both independent variables, respectively. The number of function evaluations for both cases is 500. The tiny dots represent the sample points in the center of the boxes. The centers of the circles represent the position of the global minimum points. We see from Figs. 20 and 21 that when the tolerance is defined, the algorithm will be biased toward a global search for the fixed number of function evaluations. Figure 21 clearly shows that the algorithm explores more "bigger boxes" with tolerance included than it does without tolerance (Fig. 20).

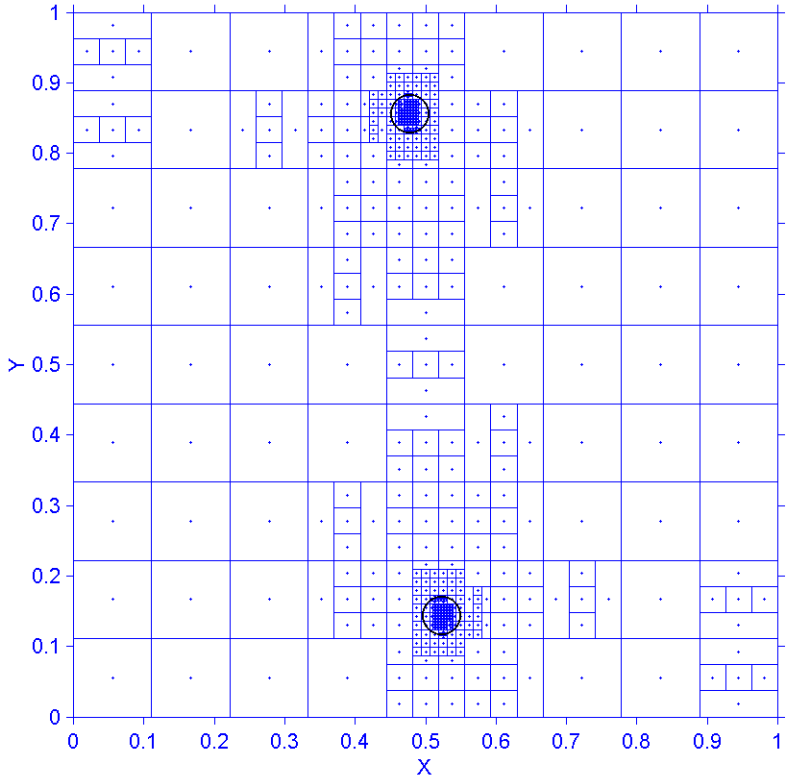


Fig. 20 Optimization results with no tolerance

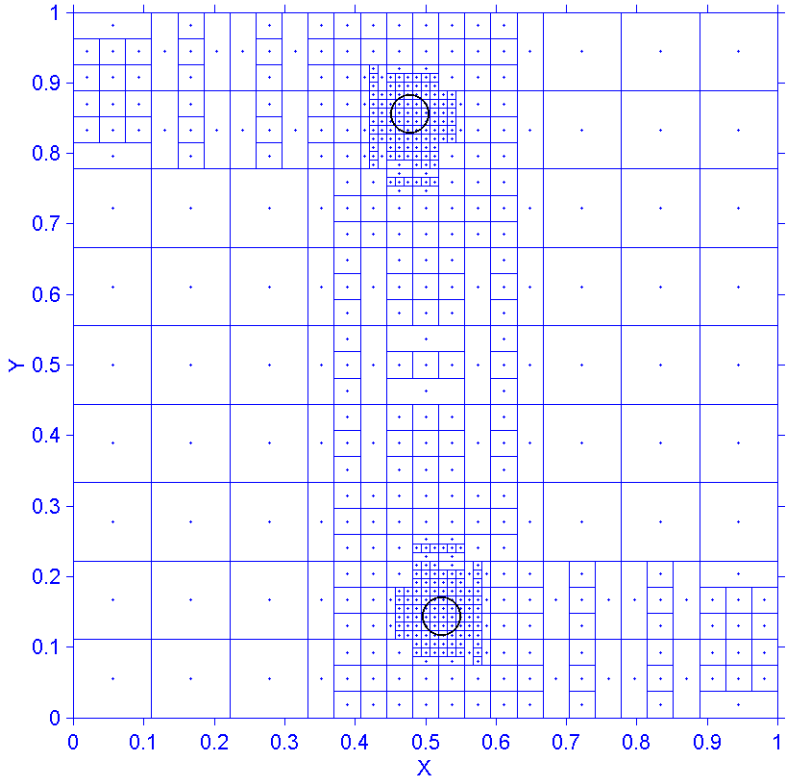


Fig. 21 Optimization results with 0.02 tolerance

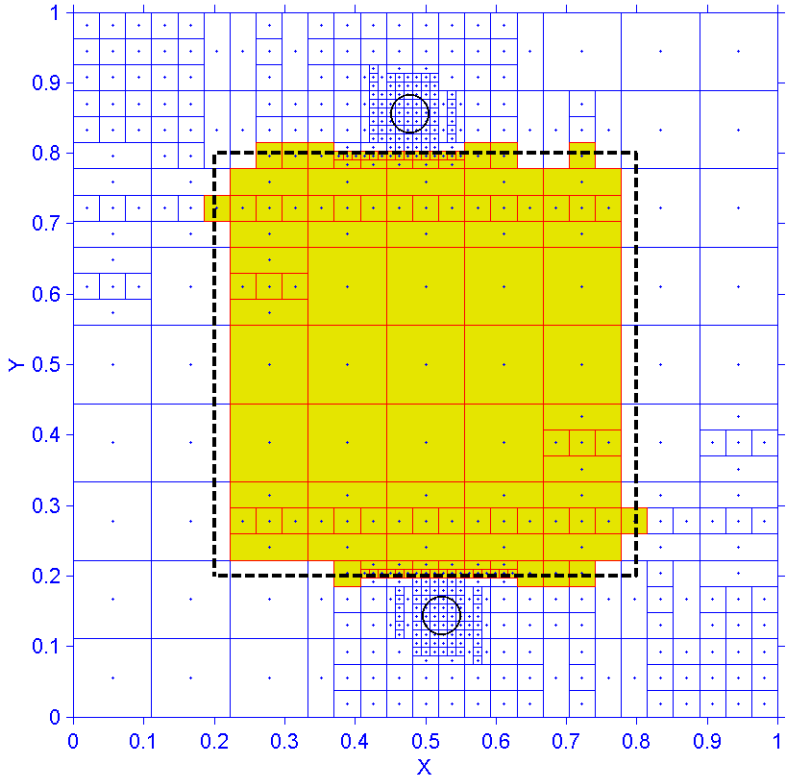


Fig. 22 Optimization results with hidden constraints

Figure 22 shows the optimization results with hidden constraints for the six-hump function. The tolerance is also set at 0.02 for both x_1 and x_2, and the number of function evaluations is still 500.

The hidden constraints for Fig. 22 are:

$0.2 \le x_1 \le 0.8$ *and* $0.2 \le x_2 \le 0.8$.

In Fig. 22 the hidden constraint areas are the areas inside the dashed-line square. The shadowed boxes represent the boxes that contain the infeasible points. Similar to the one minimum point testing function cases, Fig. 22 also shows that, inside the hidden constraint areas, the regions that are adjacent to the minimum points get partitioned more often.

The second testing function considered is the Branin function, defined as:

$$F(x_1, x_2) = [1 - 2x_2 + (1/20)\sin(4\pi x_2) - x_1]^2 + [x_2 - (1/2)\sin(2\pi x_1)]^2 ,$$

where $x_1, x_2 \in [-10,10]$. This function has five global minimum points and the global minimum is 0. If we normalize the range of variables x_1 and x_2 into $[0,1]$, then the five global minimal points are (0.55, 0.5), (0.50743, 0.52010), (0.52013, 0.51437), (0.57987, 0.48563) and (0.59257, 0.47990). The contour lines and the surface shape of the Branin function are shown in Figs. 23 and 24, respectively. The five round dots in Fig. 23 represent the global minimum points.

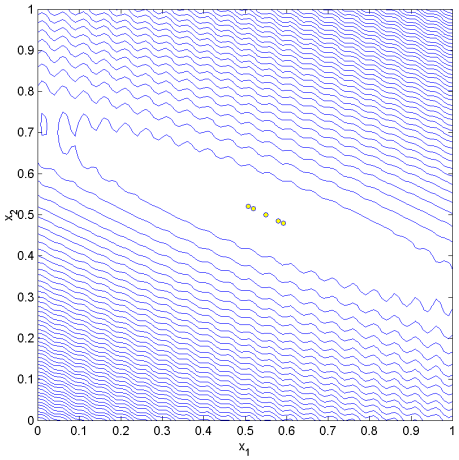


Fig. 23 Contour lines

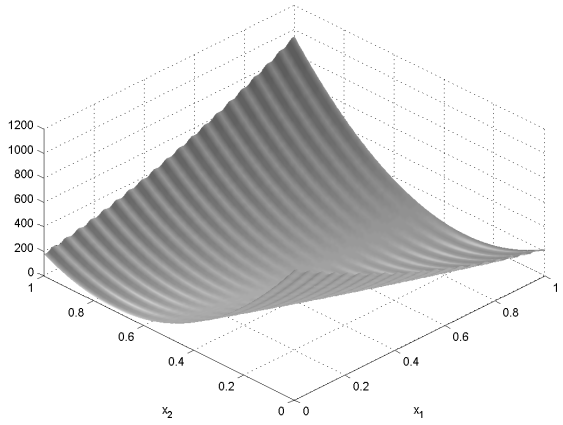
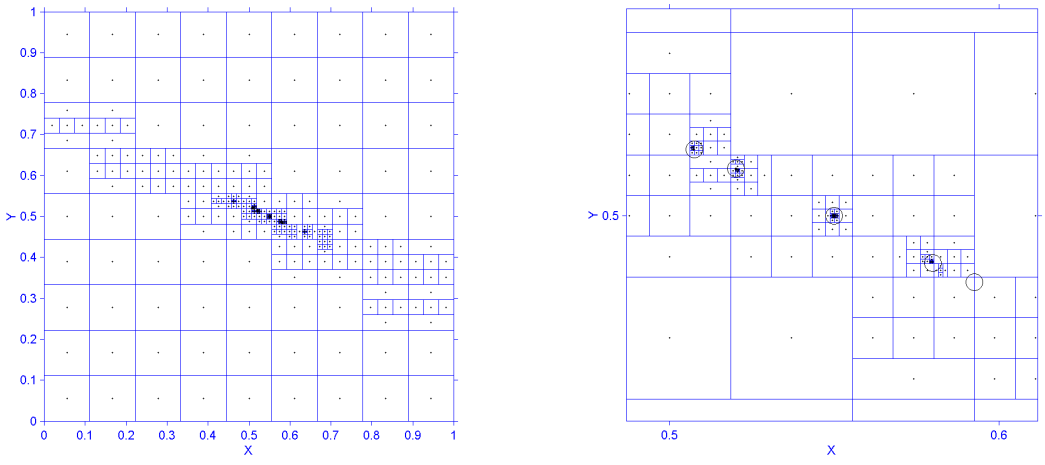
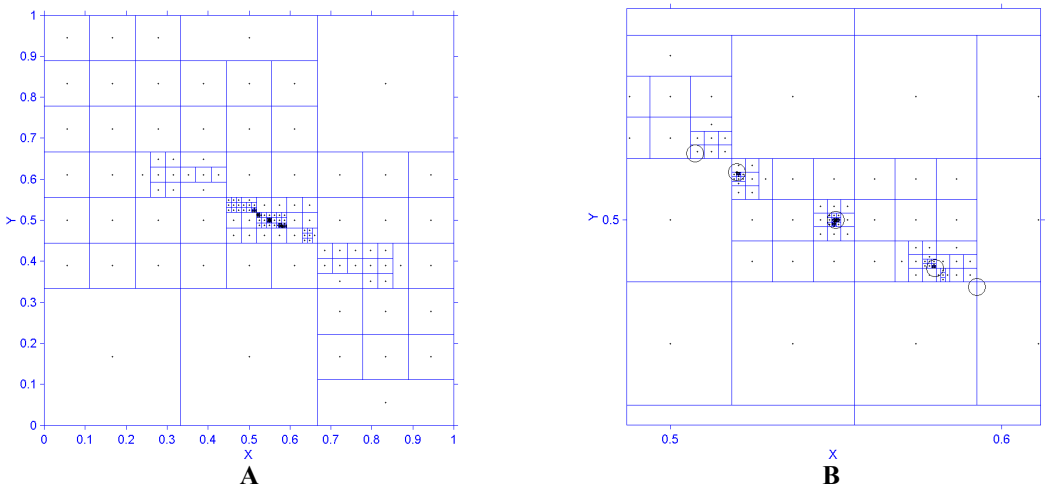


Fig. 24 Surface shape



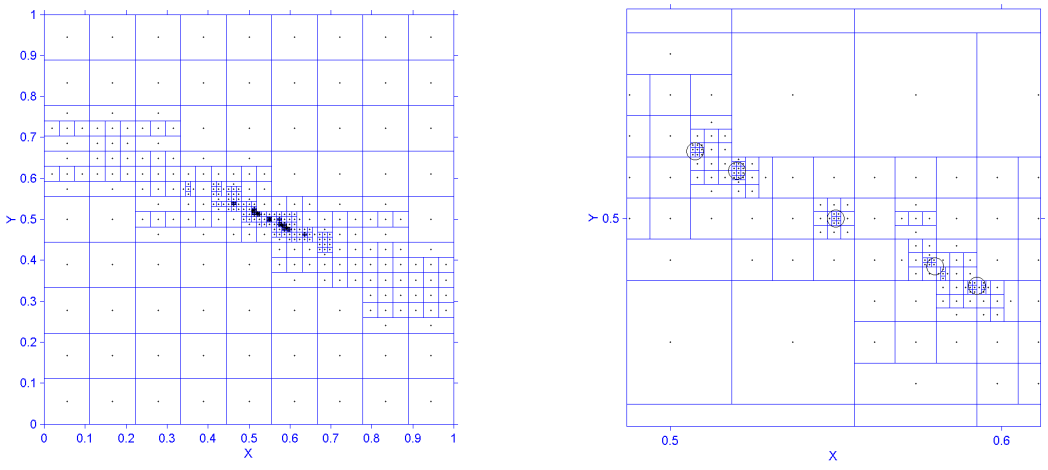
A **B**

Fig. 25 Results of standard DIRECT without tolerance



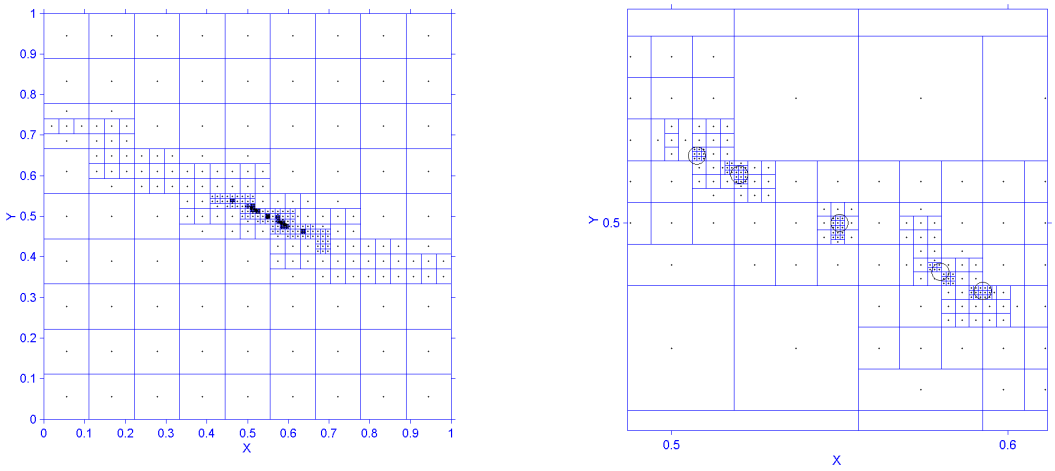
A **B**

Fig. 26 Results of DIRECT-III without tolerance



A **B**

Fig. 27 Results of standard DIRECT with 0.002 tolerances



A **B**

Fig. 28 Results of DIRECT-III with 0.002 tolerances

Figures 25 ~ 28 show the optimization results for different cases of the Branin function after 400 function evaluations. The tiny dots in Figs. 25A ~ 28A (left) on the left represent the sample points. Figures 25B ~ 28B (right) are the respective local zoom-ins around the global minimum points. The centers of the circles denote the locations of the global minimum points.

Figure 25 shows the optimization results using the standard DIRECT algorithm with no tolerance. The sample points cluster around 4 of the 5 global minima after 400 function evaluations. The algorithm found 4 of the 5 global minimum points for this case.

Figure 26 shows the optimization results using the DIRECT-III algorithm with no tolerance. DIRECT-III is a strong locally biased variation of the standard DIRECT algorithm, which combines the features of DIRECT-I and DIRECT-II. Detailed information about DIRECT-III can be found in Ref. [2]. Figure 26 shows that DIRECT-III only found 3 of the 5 global minimum points at this stage. This is because the locally biased property of DIRECT-III results in a more intensive local search around only one of the best points, thus preventing it from finding all global minima as fast as the standard DIRECT algorithm. Notice that the pattern of the optimization results in Figs. 25 and 26 are quite different. There are 3 large unexplored boxes in Fig. 26, another verification of the locally biased property of DIRECT-III.

Figures 27 and 28 illustrate the optimization results using the standard DIRECT algorithm and DIRECT-III respectively, but with 0.002 tolerances for each case. Since the definition of the tolerances ensures that the algorithm will not spend time partitioning the boxes with sizes smaller than the tolerances prescribed, the algorithm can search more unexplored larger boxes. Figure 27 shows that the DIRECT algorithm found all 5 global minima for this case.

The interesting thing about Fig. 28 is that DIRECT-III also found all 5 global minima, with a pattern similar to Figs. 25 and 27. This is because, to some extent, the globally biased property of defining tolerances balances the locally biased property of DIRECT-III.

4. SLIDER AIR BEARING DESIGN OPTIMIZATION CASE

4.1 Air bearing design optimization problem

Given a prototype slider ABS design, we wish to optimize it to obtain uniform flying heights near the target flying height with a flat roll profile, and to increase its air bearing stiffness if possible.

In this case we used the NSIC 7nm flying height slider as the prototype. The rail shape and the 3-dimensional rail geometry are shown in Figs. 29 and 30, respectively.

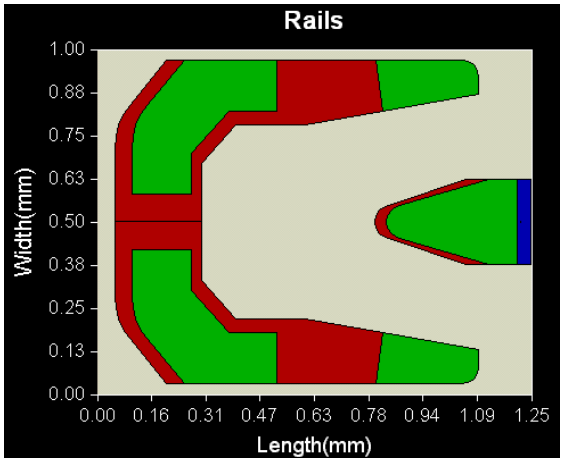


Fig. 29 Rail shape of the initial ABS design

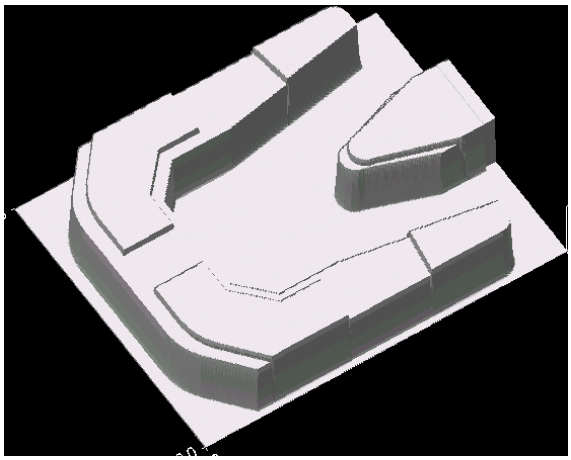


Fig. 30 3-D rail shape of the initial ABS design

The slider is a Pico slider (1.25×1.0mm) that flies over a disk rotating at 7200 RPM. Its flying heights are all around 7nm from OD to ID. In this case we want to lower all the flying heights to the target flying height, i.e. 5nm, and at the same time maintain a flat roll profile at the three different radial positions OD, MD and ID. The objective or cost function is defined as:

$$
\begin{aligned}
&\boldsymbol{1}\times(\textit{FH Max Difference}) + \\
&\boldsymbol{9}\times(\textit{FH}) + \\
&\boldsymbol{1}\times(\textit{Roll}) + \\
&\boldsymbol{1}\times(\textit{Roll Cutoff}) + \\
&\boldsymbol{1}\times(\textit{Pitch Cutoff}) + \\
&\boldsymbol{1}\times(\textit{Vertical Sensitivity}) + \\
&\boldsymbol{1}\times(\textit{Pitch Sensitivity}) + \\
&\boldsymbol{1}\times(\textit{Roll Sensitivity}) + \\
&\boldsymbol{1}\times(\textit{Negative Force}).
\end{aligned}
$$

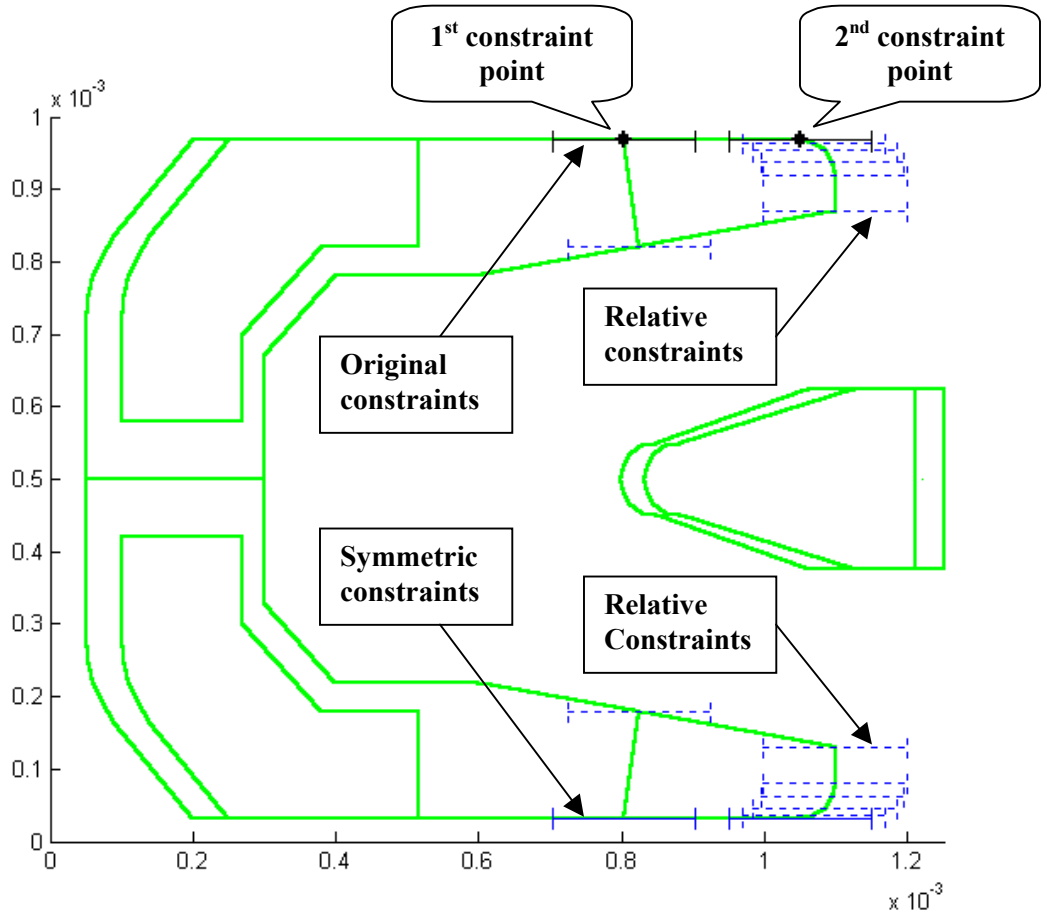


Fig. 31 Constraints defined on the initial design

The goal of the optimization is to minimize this multi-objective function under the given constraints. Note that since we are primarily concerned with the flying heights, we put a heavier weight (9) on that term. All the objective terms are normalized and their definitions can be found in the "CML optimization program version 2.0 user's manual". [3] The constraints are shown in Fig. 31, and the definition of those constraints can also be found in the user's manual.

4.2 Some special features of slider ABS optimization

For slider ABS optimization problems, the tolerance mentioned in 2.2.1 is referred to as the manufacturing tolerance. The actual slider ABS fabricating process determines the magnitude of the manufacturing tolerance, which is generally different from company to company.

Notice that in the previous chapter the hidden constraints are related only to the independent variables. The infeasible points are generated but not evaluated. However, for the slider ABS optimization problems, the hidden constraints are not directly related to the independent variables (i.e., the so called "original constraints points"[3]). They are associated with the numerical results because we cannot judge an ABS design until we know its actual Flying Heights (FH), Rolls, and Pitches etc. from the results. So the question here is: "Can the definition of hidden constraints improve our optimization and thereby improve its efficiency?" The answer is yes.

One special feature in slider ABS optimization is that we not only evaluate the slider performance at a single disc radial position, but instead we evaluate it at multiple radial positions, such as OD (Outer Diameter), MD (Middle Diameter) and ID (Inner Diameter). If a sample ABS design satisfies the hidden constraints we prescribed, then no further calculation is necessary. So if the hidden constraints are satisfied at any radial position, the calculation will not be continued and that ABS design will be marked as infeasible. It will also be given a pseudo-value depending on the status of its neighboring ABS designs. Therefore all the infeasible ABS designs will be at least partially calculated.

4.3 Simulation results

Using the same initial design, constraints, and objective function, we carried out the optimization for three different cases. We used 200 function evaluations for all three cases.

In case 1 we carried out the optimization without defining any manufacturing tolerance or hidden constraints. In this case, all the sample points generated by the algorithm are fully evaluated and have definite objective function values. Figure 32 shows the optimization results. The horizontal axis X and the vertical axis Y represent the 1st and 2nd constraint points defined in Fig. 31, respectively.

Figure 33 shows the contour lines in the search space drawn from the results in Fig. 32. The circular dot represents the best point. The contour map gives us an overall view of the performance property for every ABS sample point in the search space. Interestingly, by looking at the gradient value around the best point, we can evaluate the sensitivity of the optimized ABS design at those two constraint points. If we compare Figs. 33 and 34, it is clear that the pattern of the results generated using the DIRECT algorithm reflects the shape of the contour lines. This again verifies the high efficiency of the search strategy of the DIRECT algorithm, as well as its fast convergence rate.

In cases 2 and 3 we defined both the manufacturing tolerance and the hidden constraints. The manufacturing tolerance is set at 1μm for the two original constraint points for both cases.

In case 2 we used loose hidden constraints that are defined as:

$FH \leq 2$ nm or $FH \geq 10$ nm or $Roll \leq -30$ μrad or $Roll \geq 30$ μrad.

But in case 3 we used strict hidden constraints that are defined as:

$FH \leq 4$ nm or $FH \geq 6$ nm or $Roll \leq -10$ μrad or $Roll \geq 10$ μrad.

Figures 34 and 35 show the optimization results for case 2 and case 3, respectively. In Figs. 32, 34 and 35 the tiny dots represent the sample points generated by the algorithm; the shadowed boxes represent the boxes containing the infeasible points; and the circle represents the best point

found by the algorithm. It is not surprising that strict hidden constraints yield larger infeasible regions than loose hidden constraints.

The infeasible regions in the search space also tell us how we should search for the optimized designs. For example, Fig. 34 shows that we cannot get better ABS designs by moving the 2nd constraint point toward the trailing edge (Y = 0.5 represents the 2nd constraint point's initial position). Empirically, this makes sense because if we move the 2nd constraint point toward the trailing edge we will increase the total areas of the rails. Thus, it will result in higher FHs, which is contrary to our optimization goal of lowering the FHs.

Figures 36 ~ 38 show the variation of the objective function value for cases 1, 2 and 3 respectively. In all these figures, $Cost_{ini}$ means the initial objective function value, and $Cost_{opt}$ means the objective function value for the final optimized design. The $Percent_{imp}$ signifies the percentage of improvement for the cost function value, which is defined as:

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

N_{gen} and N_{opt} represent the number of the ABS designs generated and optimized, respectively. N_{ign} represents the number of the infeasible ABS designs.

The dark circles represent the optimized designs generated during the process. The optimized designs are the ones with the best-so-far objective function values.

For case 1, because no hidden constraints are defined, there are 0 infeasible designs. For the loose hidden constraints defined in case 2, however, there are 22 infeasible designs out of total 203 designs generated. Note that all the designs with an objective function value higher than 40 have been cut off. For the strict hidden constraints defined in case 3, there are 107 infeasible designs out of a total of 205 designs generated. With stricter hidden constraints, all the designs with an objective function value higher than 20 have been cut off.

Figures 36 ~ 38 show that all three cases yield the same optimized ABS design with an objective function value of 5.571. In case 3, more than half of the ABS designs generated are infeasible. These infeasible designs are only partially evaluated. If we assume that every infeasible design takes half of the calculation time of an average feasible design, then case 3 saved ¼ of the total calculation time as compared with case 1. Therefore, using strict hidden constraints costs less calculation time for the fixed number of samples generated.

Figure 39 shows a comparison of the initial ABS design (light-colored) and the optimized ABS design (dark-colored). Table 1 shows the summary of the optimization results, demonstrating that the optimized ABS design has very uniform flying heights around the target 5nm FH, and a reasonably flat roll profile.

Figures 40, 42 and 44 show the variations of the objective function terms for cases 1, 2 and 3, respectively. These show impressive minimization of the Flying Height term, i.e. the 2nd objective function term, on which we put a heavier weight. The roll term also was improved, along with some improvement for the Vertical Sensitivity and the Pitch Sensitivity terms. However, Roll Sensitivity has not been improved. The 1st term, the FH Maximum difference term, has not been improved either. Some objective terms such as the Pitch cutoff term and Negative Force term remain zero for all the optimized designs. The combinatorial effect results in the minimization of the total value of the objective function. By minimizing the multi-objective cost function we obtained our final optimized designs.

Figures 41, 43 and 45 show the variations of the slider performance parameters for all the best-so-far designs for cases 1, 2 and 3 respectively. The optimized design has uniform FHs around the target FH and a flat roll profile.

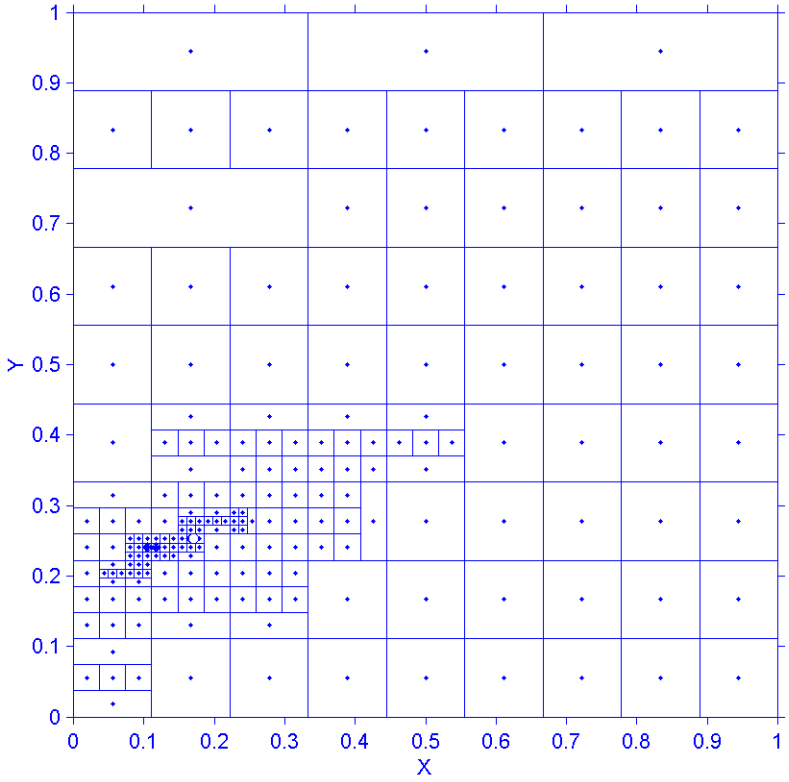


Fig. 32 Results without manufacturing tolerance or hidden constraints

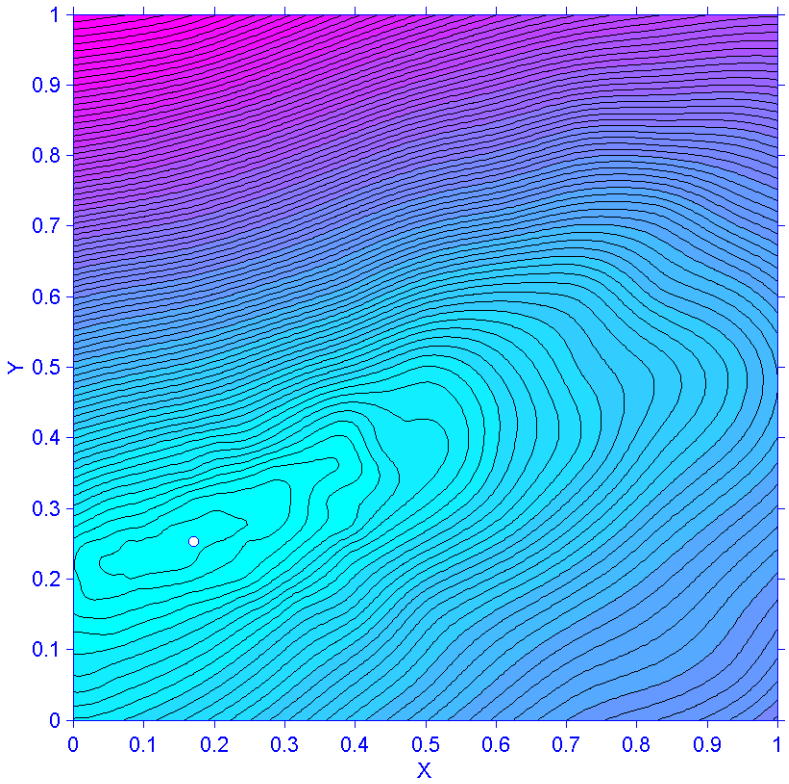


Fig. 33 Contour lines in the search space

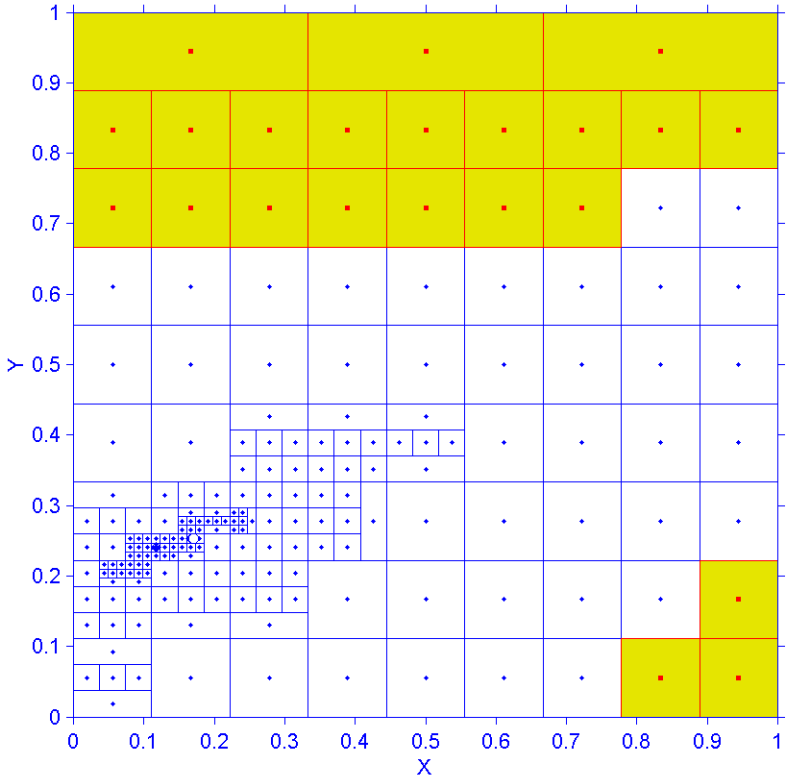


Fig. 34 Results with manufacturing tolerance and loose hidden constraints

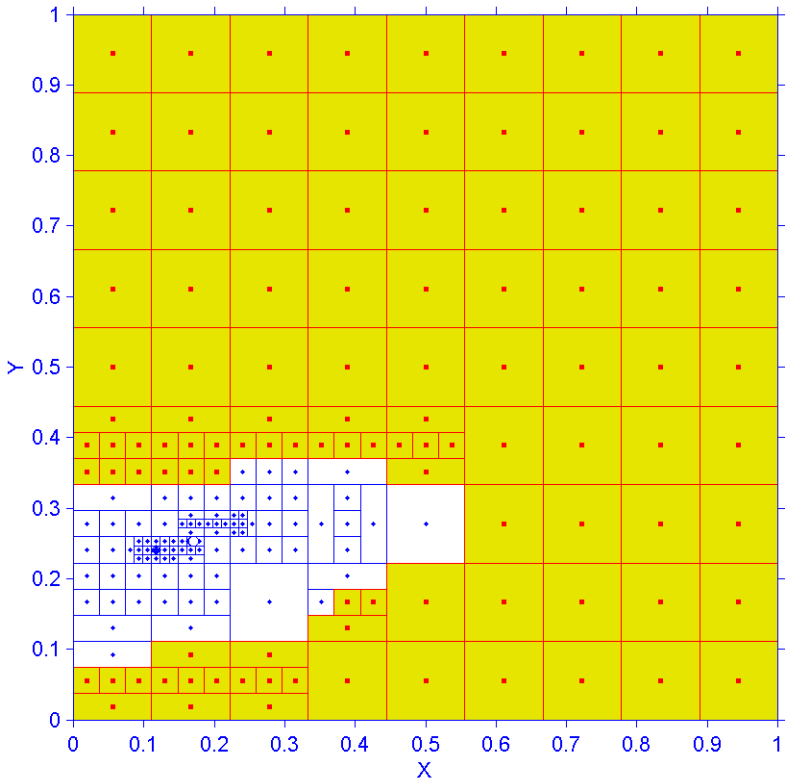


Fig. 35 Results with manufacturing tolerance and strict hidden constraints

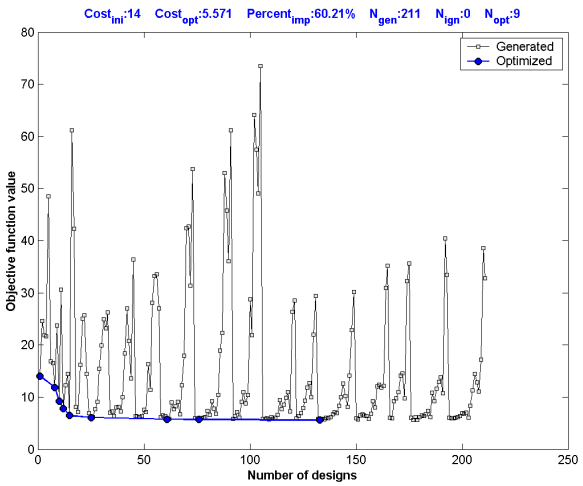


Fig. 36 Variation of the objective function value for case 1

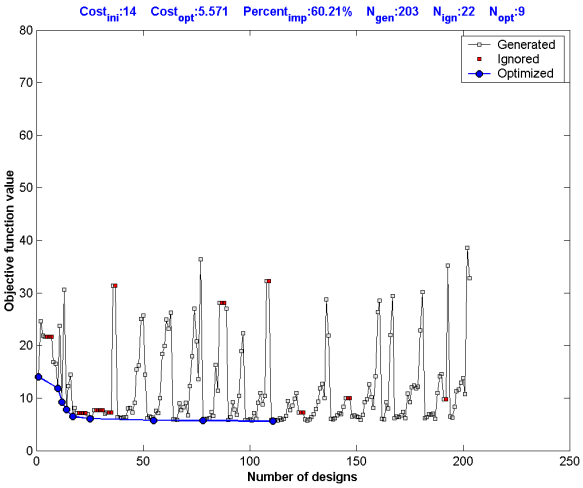


Fig. 37 Variation of the objective function value for case 2

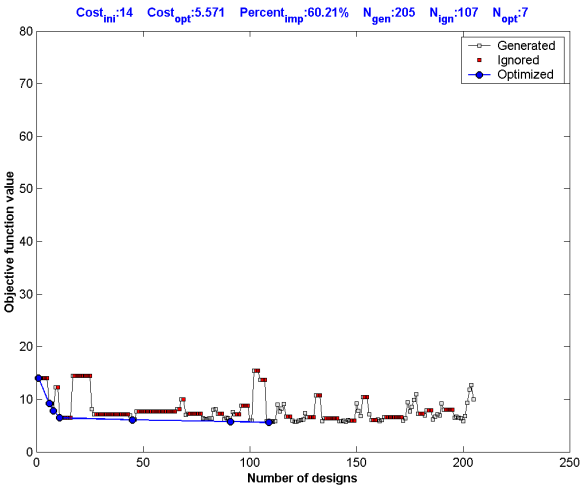


Fig. 38 Variation of the objective function value for case 3

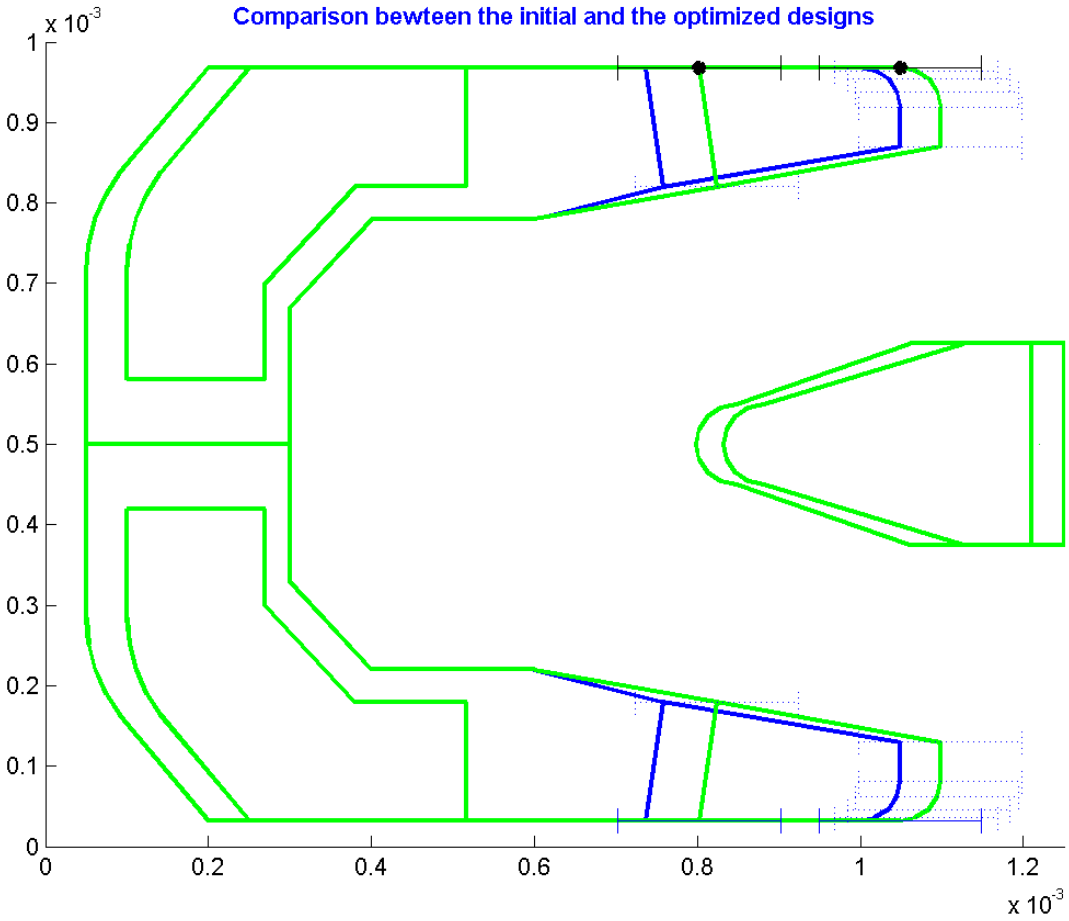


Fig. 39 Comparison of the initial design and the optimized design

	Initial ABS design			Optimized ABS design		
	OD	**MD**	**ID**	**OD**	**MD**	**ID**
FH (nm)	6.91	7.11	6.90	5.13	4.88	5.14
Roll (μrad)	-4.55	-1.54	-2.27	-4.35	-1.36	-2.42
Pitch (μrad)	207.8	167.3	116.2	213.2	175.6	125.2

Table 1 Summary of the optimization results

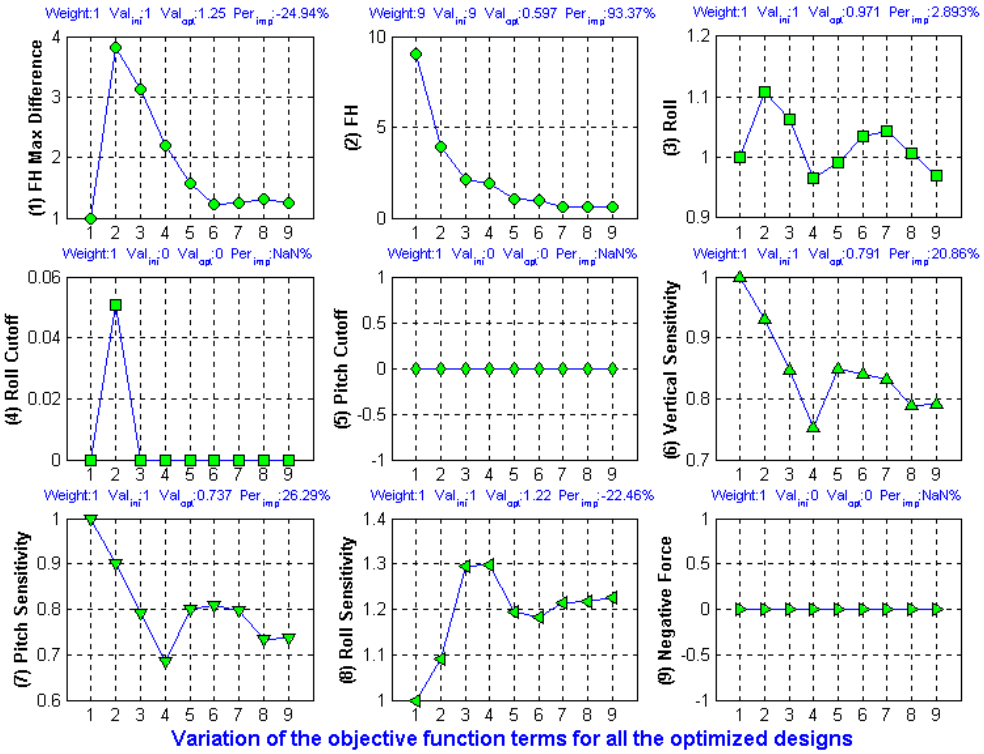


Fig. 40 Variations of the objective function terms (case 1)

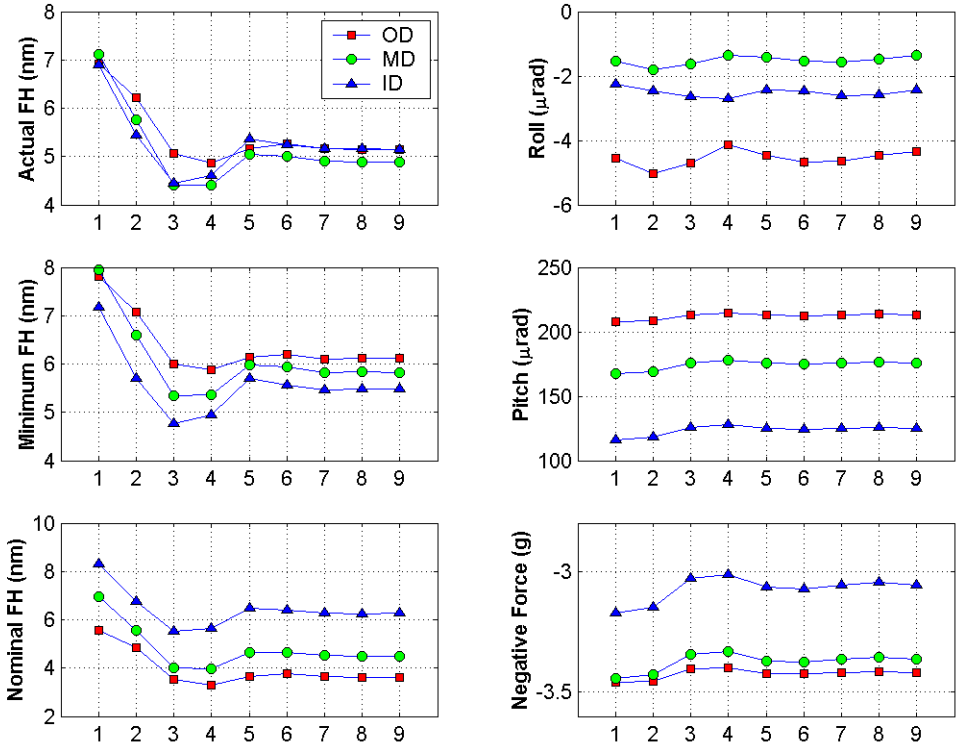


Fig. 41 Variations of the slider performance parameters (case 1)

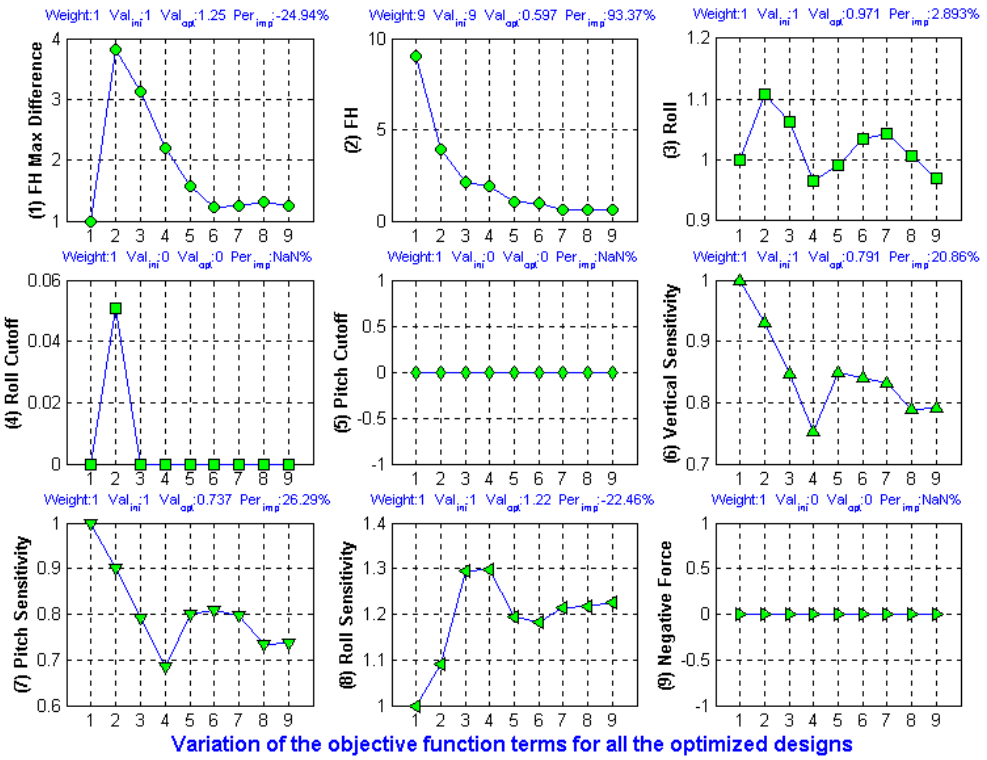


Fig. 42 Variations of the objective function terms (case 2)

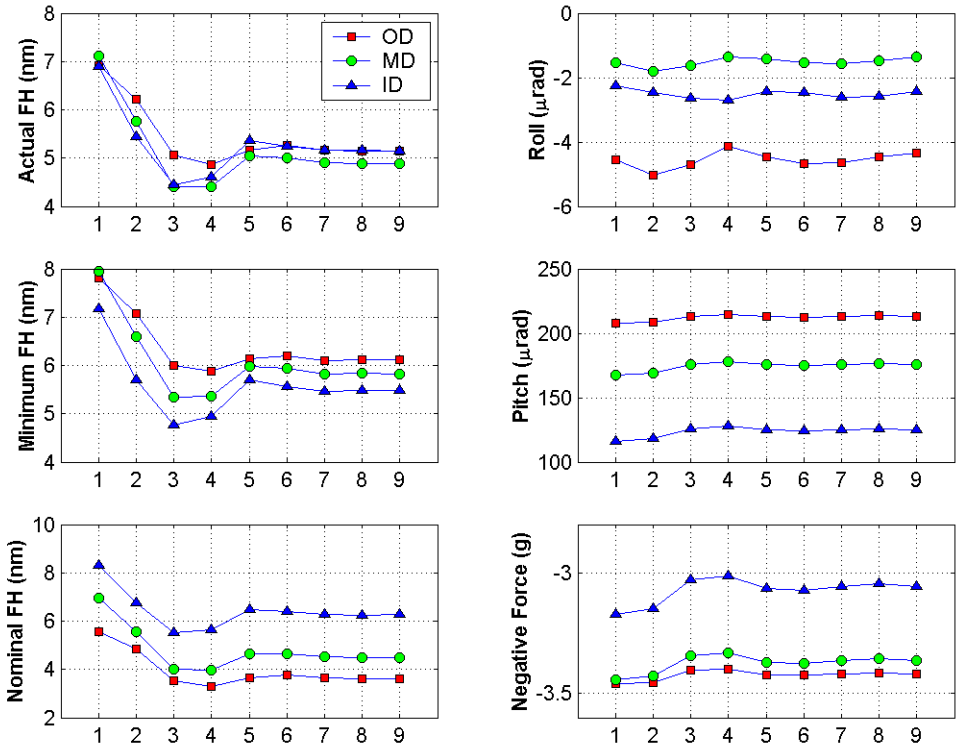


Fig. 43 Variations of the slider performance parameters (case 2)

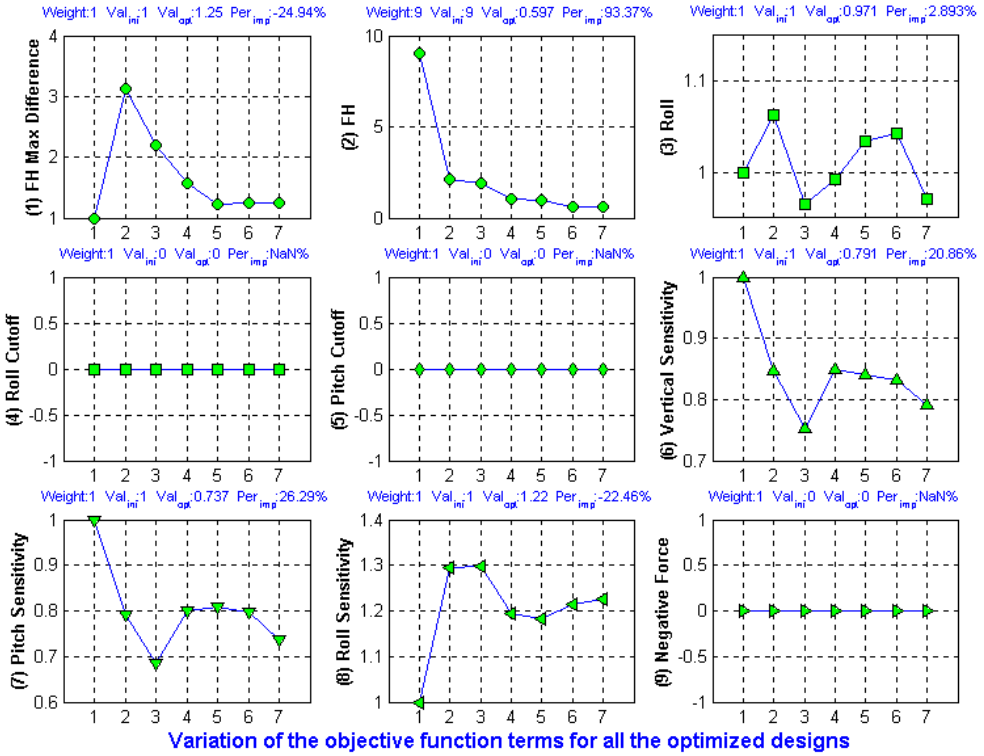


Fig. 44 Variations of the objective function terms (case 3)

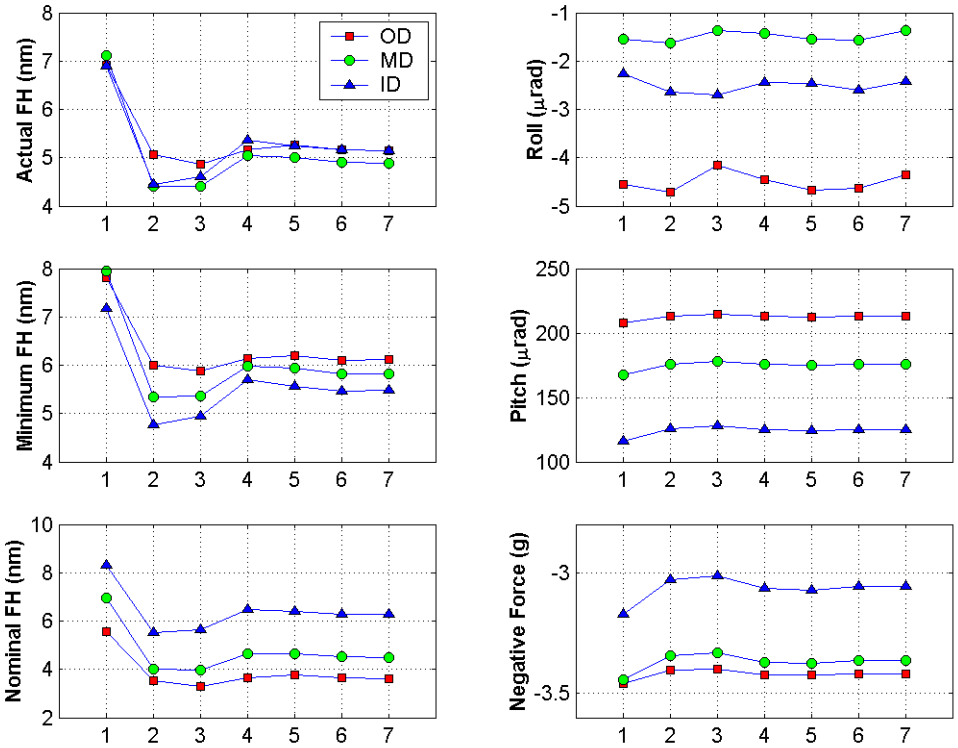


Fig. 45 Variations of the slider performance parameters (case 3)

7. CONCLUSION

The DIRECT algorithm is a deterministic global optimization technique used to find the minimum of a Lipschitz continuous function without knowing the Lipschitz constant.

Here we discussed our investigations of two modifications to the DIRECT algorithm: one to handle the tolerance (minimum side lengths) and one to deal with hidden constraints.

We carried out some numerical experiments using these modifications. The results show that by defining the 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. In other words, the algorithm will be globally biased. The numerical results also show that the strategy we adopted for dealing with hidden constraints is reasonable and effective.

We then applied the modified DIRECT algorithm to the slider ABS optimization and investigated three cases, i.e. case 1, in which no manufacturing tolerance or hidden constraints were defined; case 2, in which manufacturing tolerance and loose hidden constraints were defined; and case 3, in which manufacturing tolerance and strict hidden constraints were defined. The results show that defining the manufacturing tolerance and hidden constraints can save calculation time for the fixed number of designs generated, and thus improve the efficiency of the DIRECT algorithm.

The results also show that defining stricter hidden constraints can save even more calculation time. However, one must be careful when using very strict hidden constraints, since if the constraint points are not properly defined the algorithm may not be able to yield optimized designs.

In summary, these two modifications to the DIRECT algorithm can improve its efficiency and make it more flexible.

ACKNOWLEDGEMENTS

This study is supported by the Computer Mechanics Laboratory (CML) at the University of California at Berkeley and partially supported by the Extremely High Density Recordings (EHDR) project of the National Storage Industry Consortium (NSIC).

REFERENCES

1. Zhu, H. and Bogy, D., 2001, "*DIRECT Algorithm and its Application to Slider Air Bearing Surfaces Optimization*", Technical Report 2001-003, Computer Mechanics Laboratory, University of California at Berkeley.

2. Zhu, H. and Bogy, D., 2001, "*Locally Biased Variations of the DIRECT Algorithm and their application to the Slider Air Bearing Surfaces Optimization*", Technical Report 2001-007, Computer Mechanics Laboratory, University of California at Berkeley.

3. Zhu, H. and Bogy, D., 2001, "*The CML Air Bearing Optimization Program Version 2.0*", Technical Report, Computer Mechanics Laboratory, University of California at Berkeley.

4. Zhu, H. and Bogy, D., 2000, "*Optimization of Slider Air Bearing Shapes using Variations of Simulated Annealing*", Technical Report 2000-010, Computer Mechanics Laboratory, University of California at Berkeley.

5. Jones, D. R., Perttunen, C. D. and Stuckman, B. E., 1993, "*Lipschitzian Optimization Without the Lipschitz Constant* ", Journal of Optimization Theory and Application, Vol. 79, No. 1, pp 157-181.

6. Gablonsky, J. M., 1998, "*An Implementation of the DIRECT algorithm*", Technical Report CRSC-TR98-29, Center for Research in Scientific Computation, North Carolina State University.

7. Gablonsky, J. M., 2001, "*DIRECT Version 2.0 User Guide*", Technical Report CRSC-TR01-08, Center for Research in Scientific Computation, North Carolina State University.