DIRECT Algorithm and Its Application To Slider Air Bearing Surface Optimization

Hong Zhu and D. B. Bogy

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

ABSTRACT

One of the most important concerns in optimization is the calculation time. For global optimization, the goal is to find the global minimum point in the whole search area, not just the local minima. More sample points are usually required to obtain the global minimum, requiring more calculation time. In slider Air Bearing Surface (ABS) optimization, the evaluation of the objective function for every single sample design takes substantial computation time. It is desirable to reduce the number of sample designs evaluated without losing the global property of the optimization algorithm. There are two main parts in this report. In the first part, we give a detailed introduction to a deterministic global optimization technique called **DIRECT** (**DI**viding **RECT**angle), which is used to find the minimum of a Lipschitz continuous function without knowing the Lipschitz constant. We also present the results of extensive numerical experiments performed with different test functions and an analysis of the optimization results. In the second part, we show the application of the DIRECT algorithm to slider ABS optimization. The results of the test case show rapid convergence to the optimal design.

1. INTRODUCTION

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

We can state the optimization problem as:
Minimize $\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in S\}$, where $f(\boldsymbol{x})$ is the objective function, S is a set of feasible solutions to the problem known as the search space and $\boldsymbol{x}$ is a single point within the set.

In this report, we consider the bounded constrained optimization problem

Minimize $\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in [\boldsymbol{u},\boldsymbol{v}]\}$, where $\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}$ are n-dimensional vectors.

There are many global optimization algorithms, and they can be divided into two fundamentally different categories, i.e. deterministic algorithms and stochastic algorithms. For the deterministic algorithms, every new search point is chosen in a definite way so no random processes are involved. For the stochastic algorithms, random elements are introduced to generate the new search points.

The optimization algorithms used in our previous studies are the Simulated Annealing family, including the Standard Boltzmann Annealing (BA), Fast Cauchy Annealing (FA) and the Adaptive Simulated Annealing (ASA) [1]. They are all stochastic algorithms.

It is well known that the critical issue in global optimization is the long calculation time. Because of the need to find the global minimum of the objective function, we must generate and evaluate enough sample points. Theoretically, for either a deterministic or stochastic algorithm, if the number of the sample points is large enough, i.e., the whole search space has been searched exhaustively, the global minimum point will be found. Obviously, we cannot afford to sample every point, especially when the evaluation of each sample point is quite expensive, as in our slider ABS optimization case. Therefore, it is always desirable to use fewer sample points while maintaining the global property of the algorithms.

The main advantages of the stochastic algorithms are that they are quite robust and can be applied to a wider range of objective function types. Also, they are usually easily implemented (at least for the Simulated Annealing algorithm). But the cost is that they usually require a longer running time. The deterministic algorithms can handle definite objective functions very well. Because they use a definite searching strategy and their searching directions are strongly oriented, it is expected that they will require fewer sample points to find the global minimum point.

The DIRECT algorithm is a global deterministic algorithm developed by D. R. Jones et al. in 1993.[3] The DIRECT algorithm has a very fast convergence rate, thus it can find the global minimum very quickly compared with other algorithms.[3][4] Because of the need to reduce the calculation time in our slider ABS optimization when manufacturing tolerance is considered, we were motivated to examine the DIRECT algorithm for this application.

2. NUMERICAL METHOD

2.1 Introduction to DIRECT

The **DIRECT** algorithm is an acronym for **DI**viding **REC**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 without knowing the Lipschitz constant. DIRECT is designed to solve the problems subjected to bounded constraints.

2.2 One dimensional Lipschitzian optimization

A function $f(\boldsymbol{x})$ is said to be a Lipschitz function if

$$|f(\boldsymbol{x}) - f(\boldsymbol{x}')| \leq \boldsymbol{K}\,|\,\boldsymbol{x} - \boldsymbol{x}'\,| \quad \text{for all } \boldsymbol{x}, \boldsymbol{x}' \in [\boldsymbol{u},\boldsymbol{v}] \tag{2.1}$$

Where the positive constant $\boldsymbol{K}$ is referred as the Lipschitz constant, $\boldsymbol{x}, \boldsymbol{x}'$, $\boldsymbol{u}, \boldsymbol{v}$ are n-dimensional vectors.

For the one dimensional Lipschitz function, we have the following inequalities:

$$f(x) \geq f(u) - K(x - u) \quad (2.2)$$
$$f(x) \geq f(v) + K(x - v) \quad (2.3)$$

With the two inequalities we can define a piecewise linear function $g(x)$, which consists of two lines with slopes –K and +K and lies below $f(x)$.

$$g(x) = f(u) - K(x - u) \quad for \quad x \leq X(u,v,f,K) \quad (2.4)$$
$$g(x) = f(v) + K(x - v) \quad for \quad x \geq X(u,v,f,K) \quad (2.5)$$

Where

$$X(u,v,f,K) = [f(u) - f(v)] / (2K) + (u + v) / 2 \quad (2.6)$$

At $x = X(u,v,f,K)$, $g(x)$ has a minimum value $B(u,v,f,K)$.

$$B(u,v,f,K) = [f(u) + f(v)] / 2 - K (v - u) / 2 \quad (2.7)$$

Fig. 1 illustrates this.

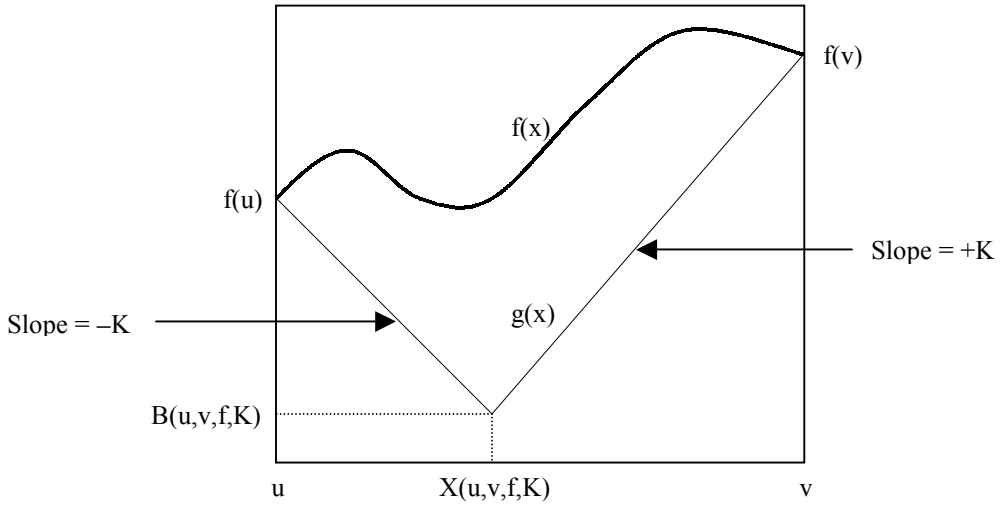


Fig. 1 Example of f(x) and g(x)

The key idea of the Shubert algorithm is to divide the search area into two intervals $I_1 = [u, X(u,v,f,K)]$ and $I_2 = [X(u,v,f,K), v]$ and then calculate the

new values of x and B for each of these two intervals, choosing a new interval with the lowest value of B to divide. The process of the Shubert algorithm is illustrated in Fig. 2.

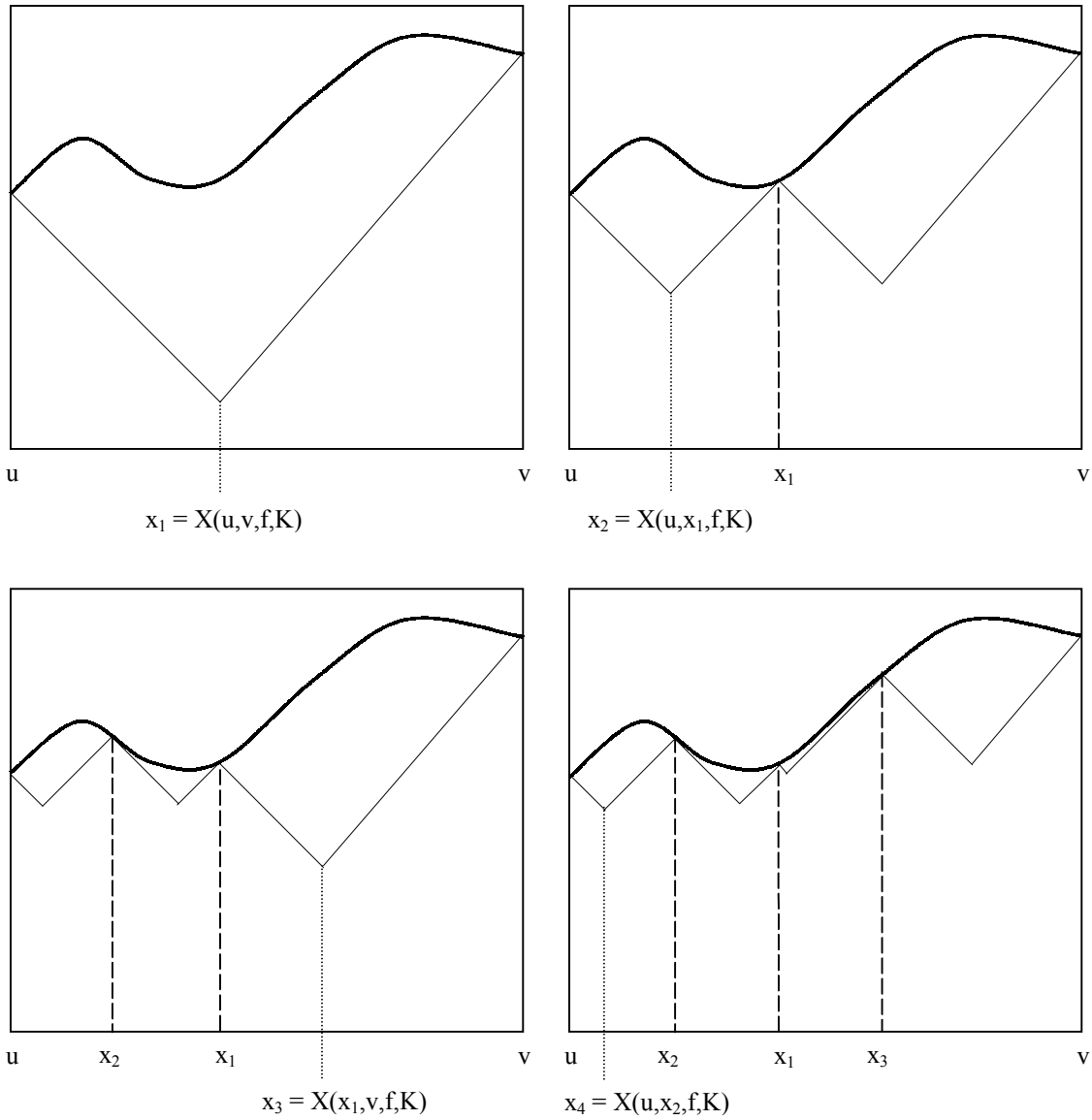


Fig. 2 Process of the Shubert algorithm

From Fig. 2, it is clear that the piece-wise linear function *g(x)* becomes more similar to the exact function *f(x)* with increasing iterations.

However, there are two limitations associated with the Shubert algorithm: First, to extend the Shubert algorithm to n dimensional cases, we would need to evaluate 2^n points at every iteration. The selection of the new points

involves solving several systems of n linear equations in n+1 unknowns, and the number of such systems grows quickly with the number of iterations. [3] That will cause high calculation complexity. Second, in order to make use of the Shubert algorithm, we must know the Lipschitz constant K, which is, of course, normally unknown or extremely hard to find for most realistic situations.

2.3 One dimensional DIRECT algorithm

The DIRECT algorithm developed by D. R. Jones et al. [3] solved the above-mentioned problems associated with the Shubert algorithm.

Again, for a one dimensional Lipschitz problem, if we let $[u,v]$ be an interval with the middle point $m = (u + v) / 2$, then for any $x \in [u,v]$ we have the following inequalities:

$$f(x) \geq f(m) + K(x - m) \quad for \quad x \leq m \tag{2.8}$$
$$f(x) \geq f(m) - K(x - m) \quad for \quad x \geq m \tag{2.9}$$

With the two inequalities we can also define a piecewise linear function $h(x)$, which consists of two lines with slopes +K and –K, and lies below $f(x)$.

$$h(x) = f(m) + K(x - m) \quad for \quad x \leq m \tag{2.10}$$
$$h(x) = f(m) - K(x - m) \quad for \quad x \geq m \tag{2.11}$$

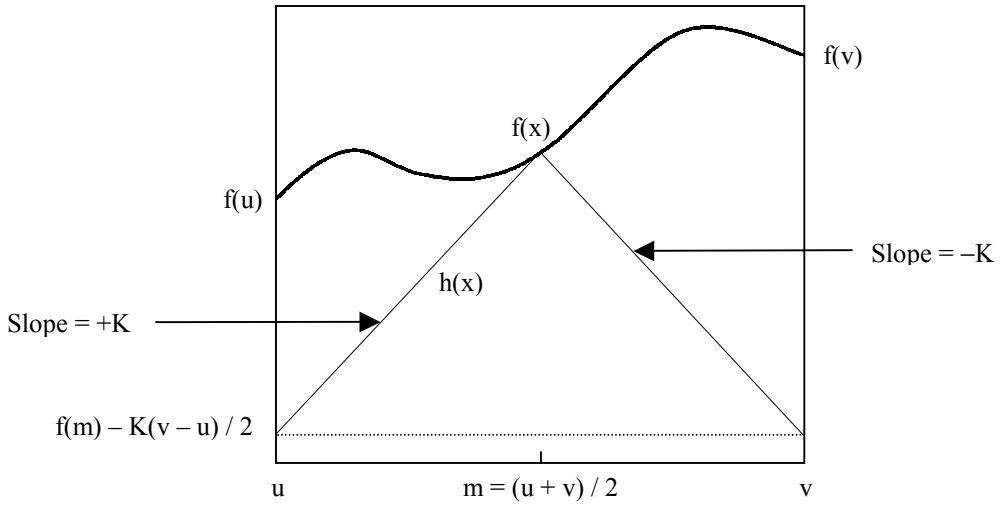


Fig. 3 Example of f(x) and h(x)

This is shown in Fig. 3. The lowest value of $h(x)$ is located at $x = u$ and $x = v$. The lowest value is $f(m) – K(v – u) / 2$.

The DIRECT algorithm consists of two main components: the dividing strategy, which defines how we partition an interval; and the potentially optimal intervals, i.e., how we choose the intervals to be partitioned at each iteration step.

2.3.1 Dividing strategy

The DIRECT algorithm divides the interval into three equal subintervals. The dividing strategy is shown in Fig. 4.

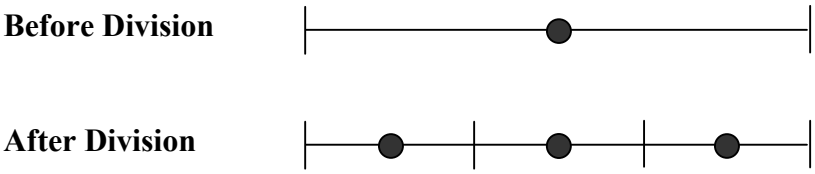


Fig. 4 Dividing strategy of DIRECT

2.3.2 Potentially optimal intervals

Assume that the search area $[u, v]$ has been divided into N intervals $[u_i, v_i]$ with centers m_i. Then create a graph with $(v – u) / 2$ as the x-axis and $f(m)$ as the y-axis, as shown in Fig. 5. The x-axis represents the distance from the intervals' centers to their endpoints, and indicates the amount of unexplored territory in the interval. The y-axis represents the values of the function at the intervals' centers, and indicates the "quality" of the sample point, where low function value means high quality of the sample point.

Next draw a line with slope K through any data point in Fig. 5. The intersection of this line with the y-axis is $(0, D(u_i, v_i))$, where $D(u_i, v_i))$ is a lower bound for the function in the interval $[u_i, v_i]$.

$$D(u_i, v_i) = f(m_i) – K(v_i – u_i) / 2. \tag{2.12}$$

Then the interval with the lowest value of $D(u_i, v_i)$ is selected as the one to be partitioned next. Imagine that we draw a line with slope K below all

the data points and then move it upward. The first data point that the line intersects would be the sample point of the interval that is to be dividied in the next step.

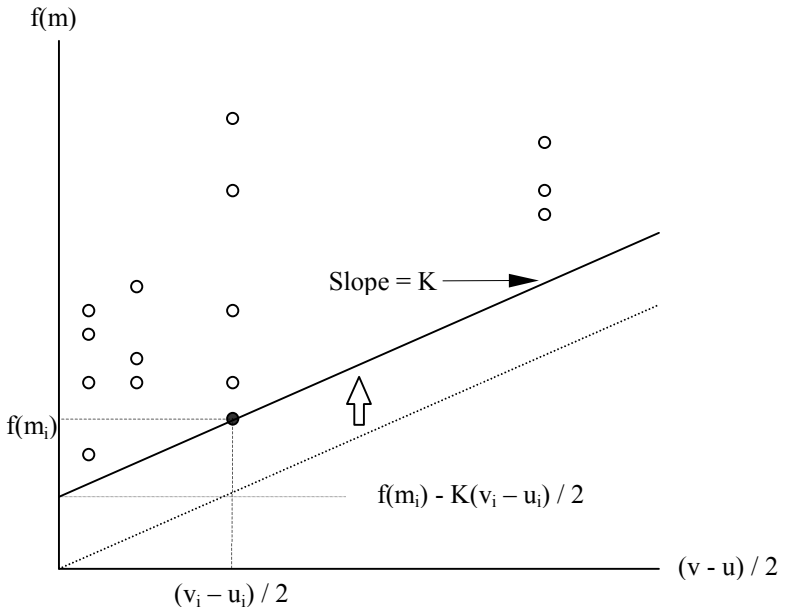


Fig. 5 Interval selection

However as we mentioned before, in many situations the Lipschitz constant K is unknown. So we need to estimate K based on our known data points. This would correspond to identifying the set of intervals that could be chosen using a line with some positive slope. These intervals are called potentially optimal intervals. This is done in DIRECT by finding the "convex hull" of the known data points. The algorithm used here to find the convex hull is the Graham's scan [5] (see **Appendix A** for details).

An example of choosing potentially optimal intervals is shown in Fig. 6. For all the data points with the same x-coordinate (i.e., the distance from the center to the endpoints), only the point with the lowest function value is eligible to be selected.

In the DIRECT algorithm, the formal definition of the potentially optimal interval is given as follows:

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

$$f(m_j) - \widetilde{K}\,(v_j - u_j)/2 \le f(m_i) - \widetilde{K}\,(v_i - u_i)/2 \quad \text{for any } i \tag{2.13}$$

$$f(m_j) - \widetilde{K}\,(v_j - u_j)/2 \le f_{min} - \varepsilon\,|f_{min}| \tag{2.14}$$

The inequality (2.13) represents the property of the data points on the convex hull. The inequality (2.14) ensures that the lower bound for the interval, based on the rate-of-change constant $\widetilde{K}$, exceeds the current best solution by a small amount. This condition is needed to prevent the algorithm from becoming too local in its orientation, wasting valuable function evaluation time in search of an extremely small improvement. In this report, ε is set to 10^{-2}, which means that the lower bound for the interval should exceed the current best solution by more than 1%. Also note that $\widetilde{K}$ is a rate-of-change constant, not a Lipschitz constant *K* in the normal sense.

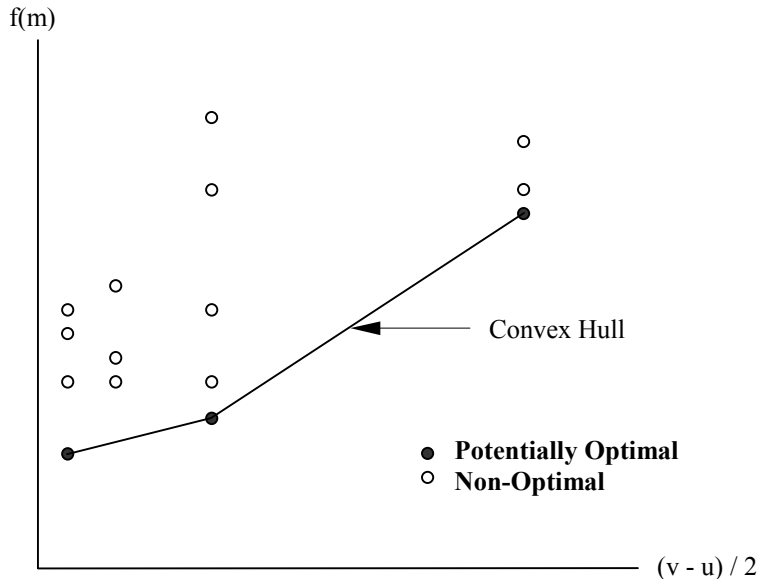


Fig. 6 Selection of potentially optimal intervals

2.4 Multi-dimensional DIRECT algorithm

The multi-dimensional DIRECT algorithm is an extension of the one-dimensional case. 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. The main difference between the multi-dimensional DIRECT algorithm and the one-dimensional case is the partitioning of the search space.

2.4.1 Dividing strategy

We now explain the dividing strategy of the multi-dimensional DIRECT algorithm for the hyper-cubes and for the hyper-rectangles:

A. Partition of a hyper-cube
Assume m is the center point a hyper-cube. We will 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 $s_i = \min \{ f(m - \delta e_i), f(m+\delta e_i) \}$, which means that the partition will be in the order given by s_i, starting with the lowest s_i. Therefore, the hyper-cube is first partitioned along the direction with the lowest s_i, and 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 that we obtain a reduction in the maximal side length of a hyper-rectangle.

2.4.2 Potentially optimal hyper-rectangles

The definition of potentially optimal hyper-rectangles is very similar to Definition 2.1. 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 we define the potentially optimal hyper-rectangle as:

Definition 2.2 *Let $\varepsilon > 0$ be a positive constant and f_{min} be the current lowest function value. A hyper-rectangle j is said to be potentially optimal if there exists some rate-of-change constant $\widetilde{K} > 0$ such that*

$$f(m_j) - \widetilde{K} d_j \leq f(m_i) - \widetilde{K} d_i \quad \text{for any } i \tag{2.15}$$

$$f(m_j) - \widetilde{K} d_j \leq f_{min} - \varepsilon |f_{min}| \tag{2.16}$$

2.4.3 2-D and 3-D examples

Now let's use the first few iterations for a 2-D and a 3-D examples to demonstrate the process of the DIRECT algorithm.

For the 2-D case, the function used here is:

$F(x_1, x_2)=(x_1 - 0.4)^2 + (x_2 - 0.2)^2 \quad where\ x_1, x_2 \in [0,1]$

Figs. 7a ~ 7f show the first 5 iterations for this 2-D case.

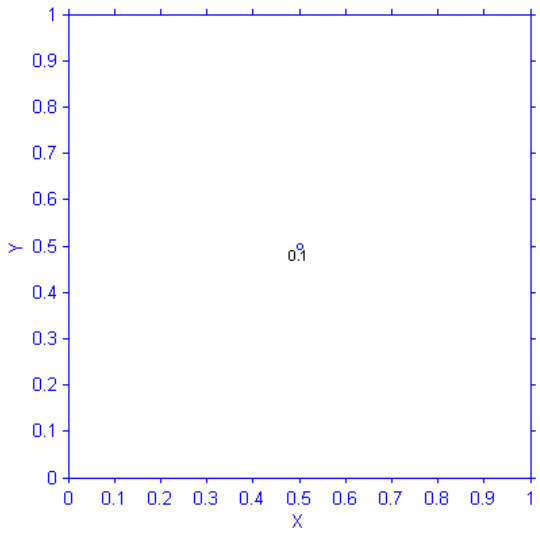


Fig. 7a Initial state

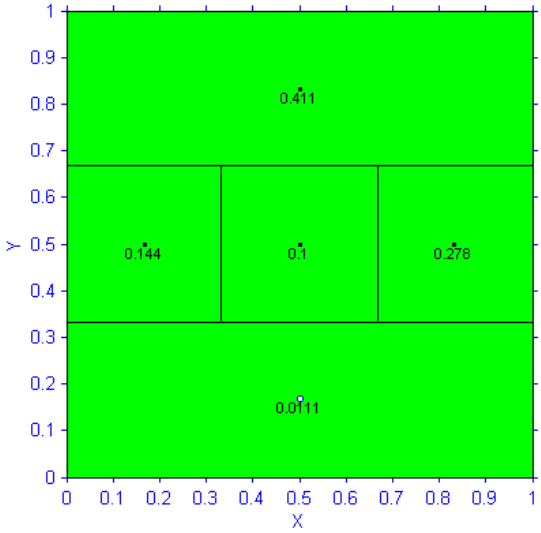


Fig. 7b Iteration 1

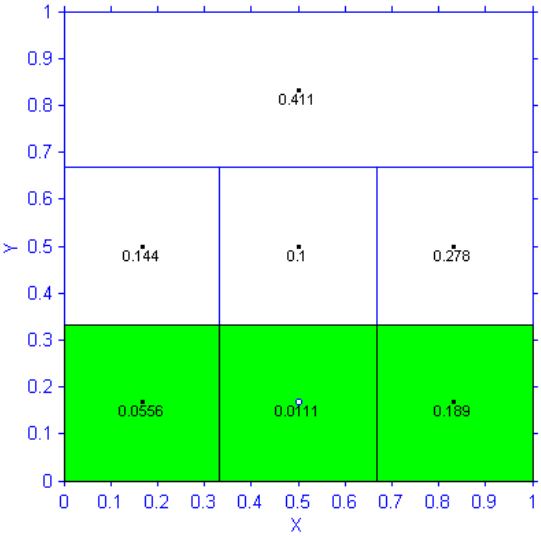


Fig. 7c Iteration 2

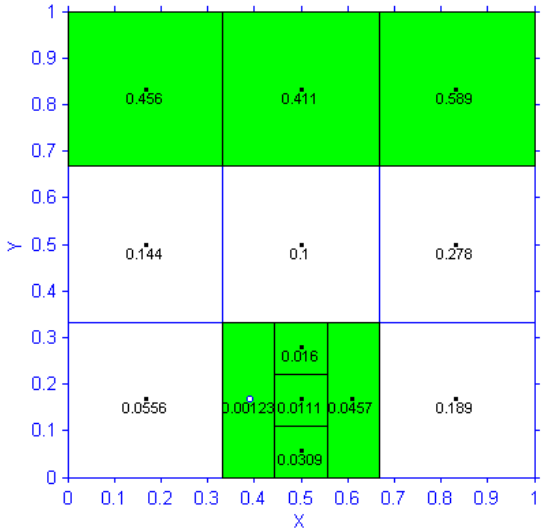


Fig. 7d Iteration 3

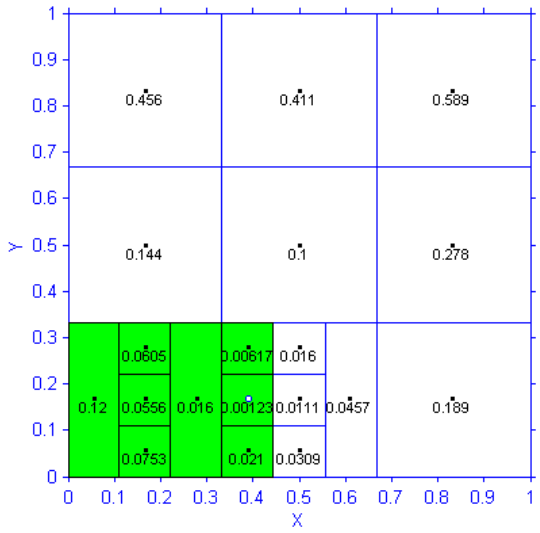


Fig. 7e Iteration 4

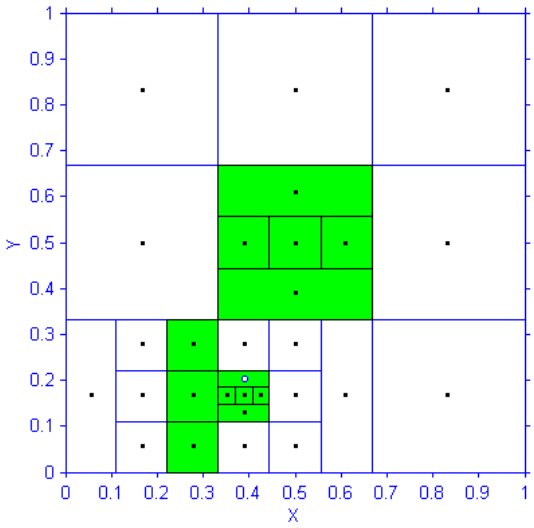


Fig. 7f Iteration 5

In these pictures, the x-axis stands for variable x_1 and the y-axis stands for variable x_2. The unit square is the search space. The shadowed areas are the boxes (can be squares or rectangles) just partitioned. The boxes chosen are the potentially optimal ones. The dots represent the center points of the boxes. The circular dot shows the sample point with the lowest function value. The numbers under those dots are the function values at those center points.

From Fig. 7b we see that

$s_1 = \min \{0.144, 0.278\} = 0.144$
$s_2 = \min \{0.0111, 0.411\} = 0.0111$

So the x_2 direction (y) gets partitioned first, and then the x_1 direction (x) gets partitioned.

From Fig. 7c we see that the rectangles are only partitioned along their longest side.

For the 3-D case, consider the function:

$$F(x_1, x_2, x_3)=(x_1 - 0.2)^2 + (x_2 - 0.3)^2 + (x_3 - 0.4)^2 \quad \text{where } x_1, x_2, x_3 \in [0,1]$$

Figs. 8a ~ 8f show the first 5 iterations for this 3-D case.

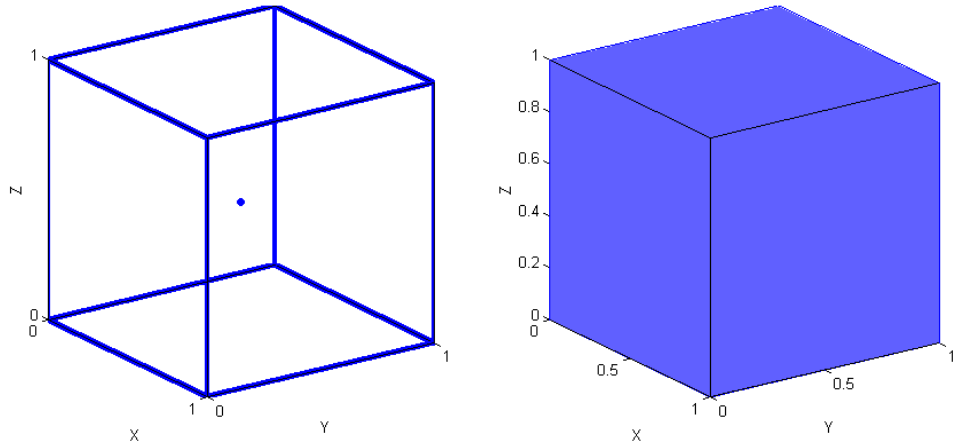


Fig. 8a Initial state

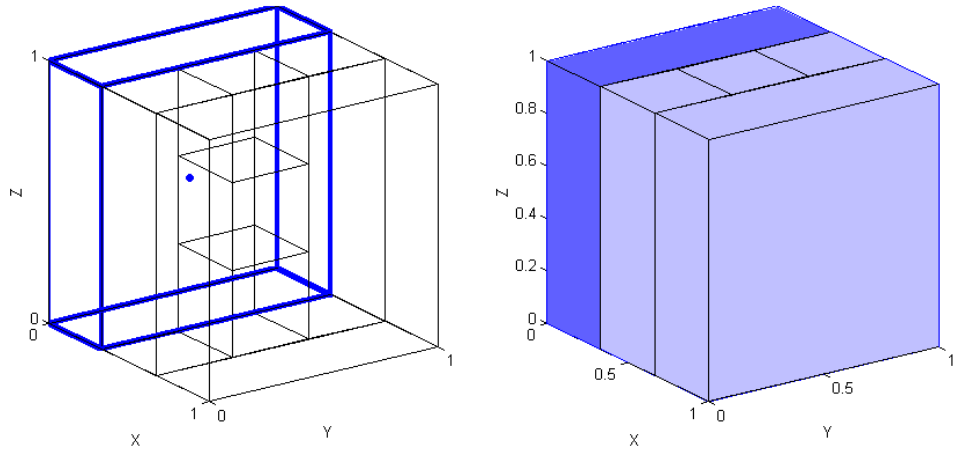


Fig. 8b Iteration 1

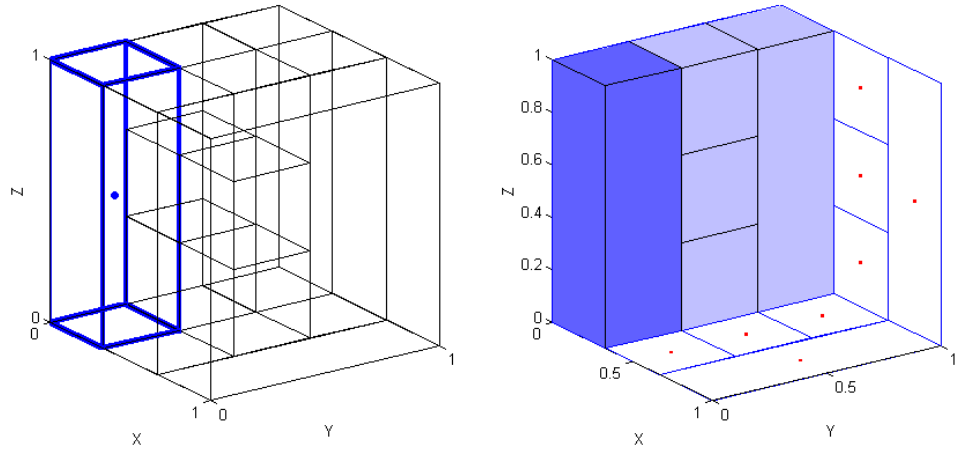


Fig. 8c Iteration 2

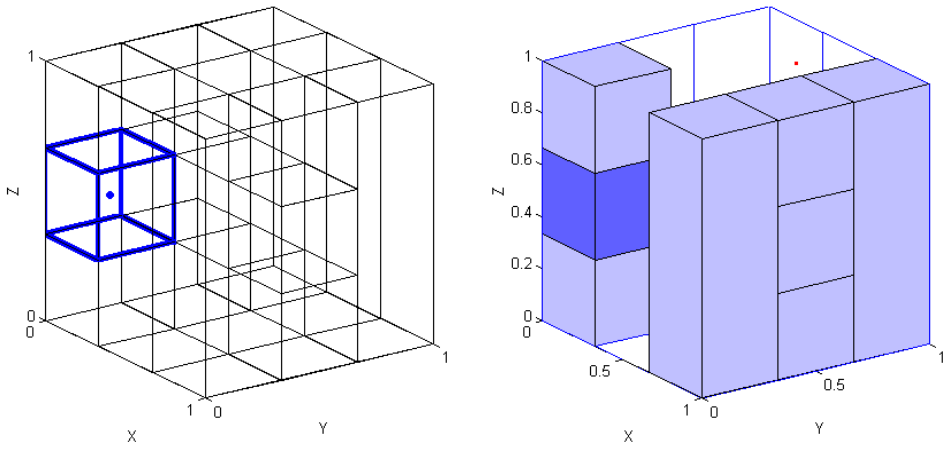


Fig. 8d Iteration 3

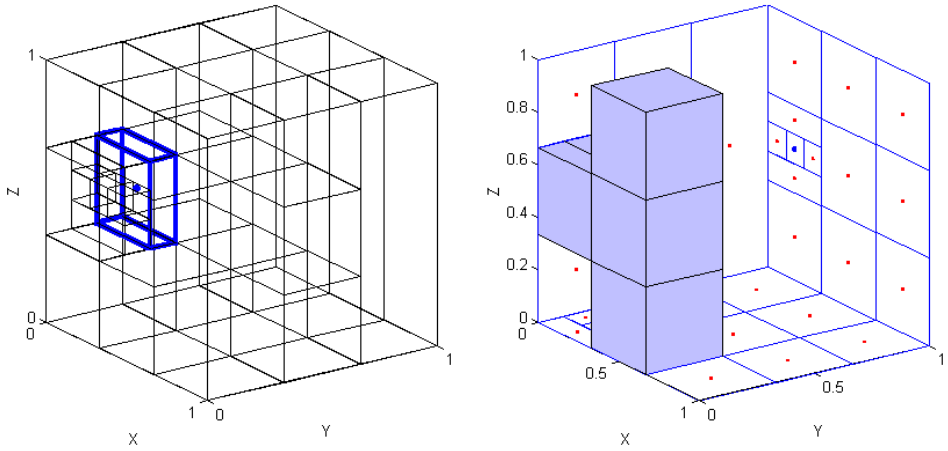


Fig. 8e Iteration 4

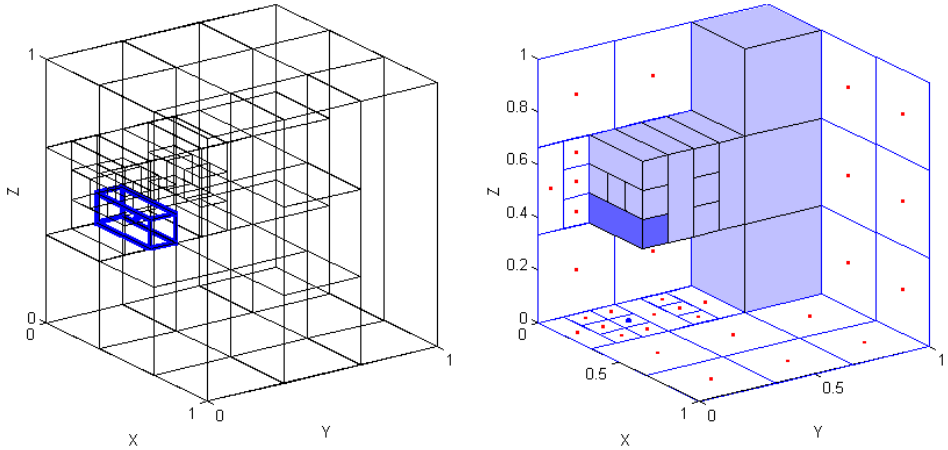


Fig. 8f Iteration 5

In these figures, the x-axis stands for variable x_1, the y-axis stands for variable x_2 and the z-axis stands for variable x_3,. The unit cube is the search space.

The graphs on the left in Figs. 8a ~ 8f show the frames of all the boxes. The box with the thick lines is the one where the sample point with the lowest function value is located. This sample point is represented by a circular dot.

The graphs on the right in Figs. 8a ~ 8f show the partition status corresponding to each of the figures on the left. The shadowed boxes are the cubes or cuboids that were just partitioned. The boxes chosen are the potentially optimal ones. The dark-shadowed box represents the box that contains the sample point with the lowest function value. All the boxes and the sample points are projected to the XY, YZ and ZX planes.

3. NUMERICAL EXPERIMENTS WITH THE DIRECT ALGORITHM

3.1 General testing function cases

The testing functions used here include 2-D, 3-D, 5-D and 10-D functions. These functions have only one global minimum point, and the minimum values of these functions are zero. These functions are defined as follows:

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

3-D: $F(x_1, x_2, x_3) = (x_1 - 0.2)^2 + (x_2 - 0.3)^2 + (x_3 - 0.4)^2$.

5-D: $F(x_1, x_2, x_3, x_4, x_5) = (x_1 - 0.1)^2 + (x_2 - 0.3)^2 + (x_3 - 0.5)^2 + (x_4 - 0.7)^2 + (x_5 - 0.9)^2$.

10-D: $F(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (x_1 - 0.1)^2 + (x_2 - 0.2)^2 + (x_3 - 0.3)^2 + (x_4 - 0.4)^2 + (x_5 - 0.5)^2 + (x_6 - 0.6)^2 + (x_7 - 0.7)^2 + (x_8 - 0.8)^2 + (x_9 - 0.9)^2 + (x_{10} - 1.0)^2$.

For all these cases, $x_i \in [0,1]$, $i = 1,...10$.

The results for 2-D case are shown in Figs. 9 ~ 13.

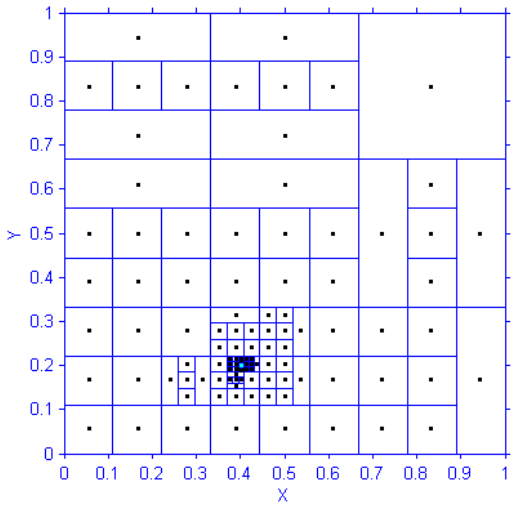


Fig. 9 Results for the 2-D case

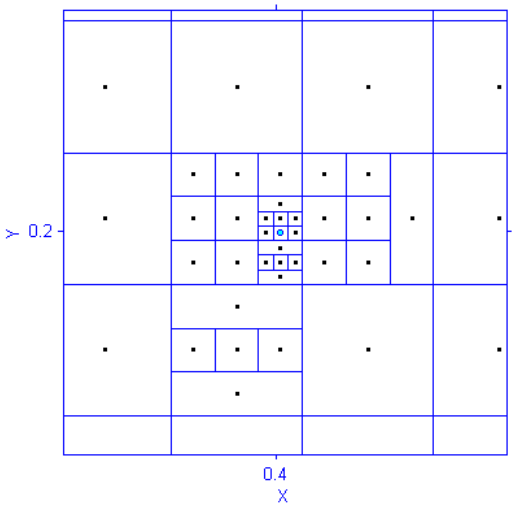


Fig. 10 Local zoom-in near minimum

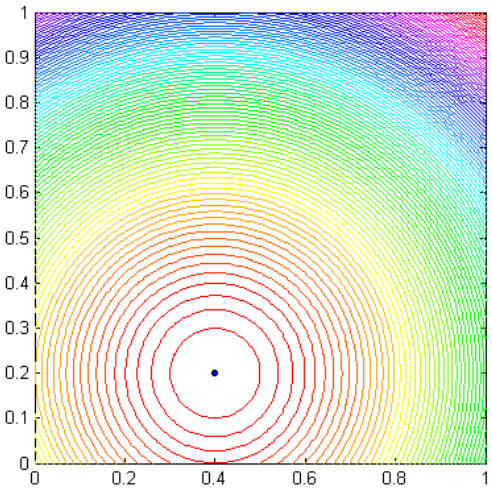


Fig. 11 Contour lines of the 2-D testing function

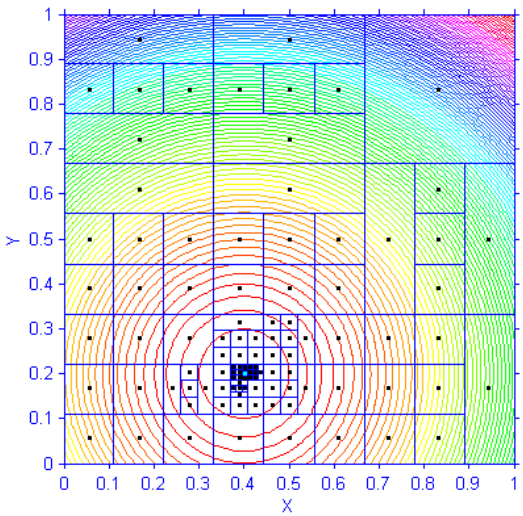


Fig. 12 Combination of contour lines and results

Figure 9 shows the results of the optimization after only 10 iterations (113 function evaluations). The dots represent the sample points in the center of the boxes. Figure 10 shows the local zoom-in of Fig. 9. The global minimum point found by DIRECT at this stage is (0.4012346, 0.1995885), which is denoted by the circular dot in Fig. 10, and the value at the minimum point is 1.693509E-06. The exact minimum point for this 2-D function is (0.4, 0.2) and the minimum value is 0. So the optimization results are very close to the exact solution.

Figure 11 shows the contour lines of the 2-D function, and the dot represents the exact minimum point. We have overlaid the optimization results and the contour lines in Fig. 12. It is seen that the sample points generated by the DIRECT algorithm become more and more clustered around the exact solution. Figure 13 shows the DIRECT algorithm's convergence property for this case with a very fast convergence rate.

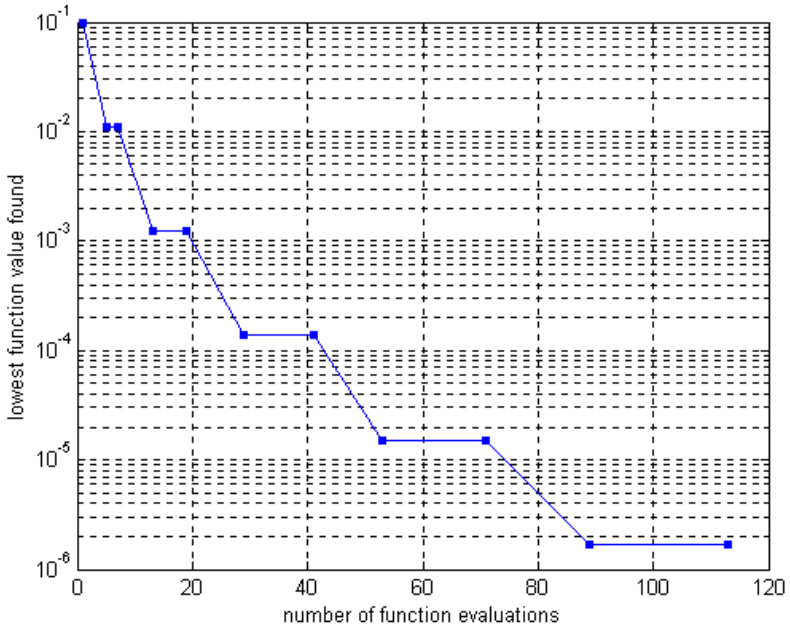


Fig. 13 Convergence property for the 2-D case

Figures 14 and 15 show the results for the 3-D case.

The left picture in Fig. 14 shows the results of optimization after 14 iterations (223 function evaluations). All the sample points and frames of all the boxes are projected to the XY, YZ and ZX planes. The right picture of Fig. 14 shows the local zoom-in of the left one. The shadowed box contains the sample point with the lowest function value. The three dashed lines point to the projection of the point with the lowest function value on the XY, YZ and ZX planes respectively. The global minimum point found by DIRECT at this stage is (0.1995885, 0.2983539, 0.4012346), and the value at the minimum point is 4.403123E-06. The exact minimum point for this 3-D function is (0.2, 0.3, 0.4) and the minimum value is 0. So again the optimization results are very close to the exact solution.

The convergence curve shown in Fig. 15 confirms that the DIRECT algorithm has a very fast convergence rate.

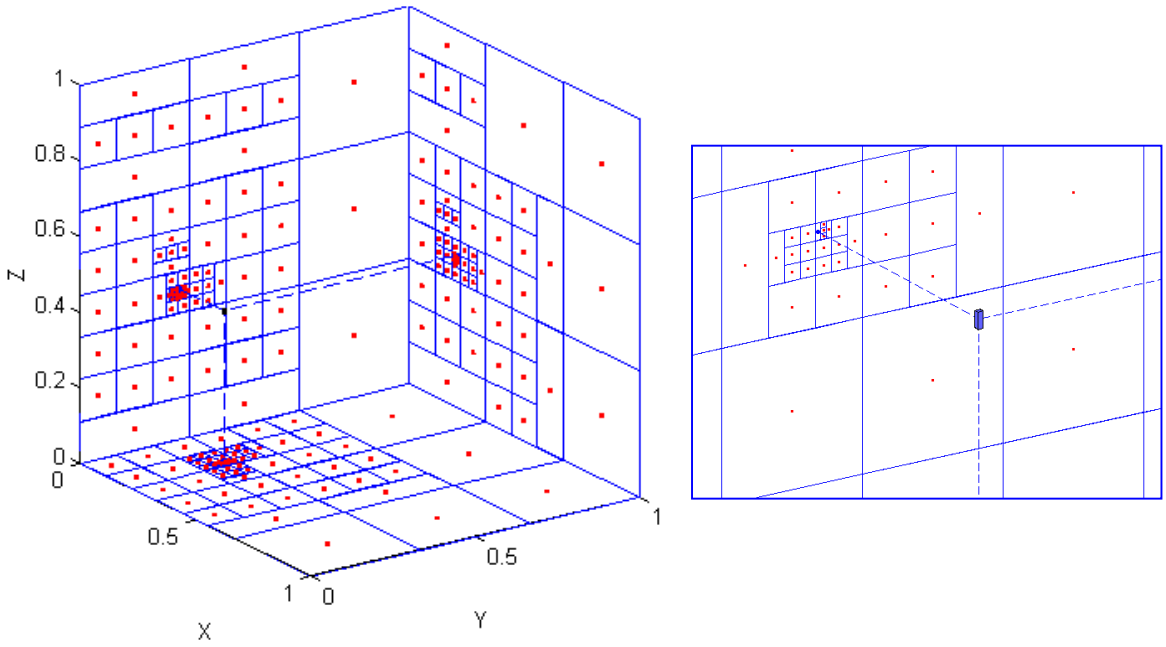


Fig. 14 Results of 3-D case

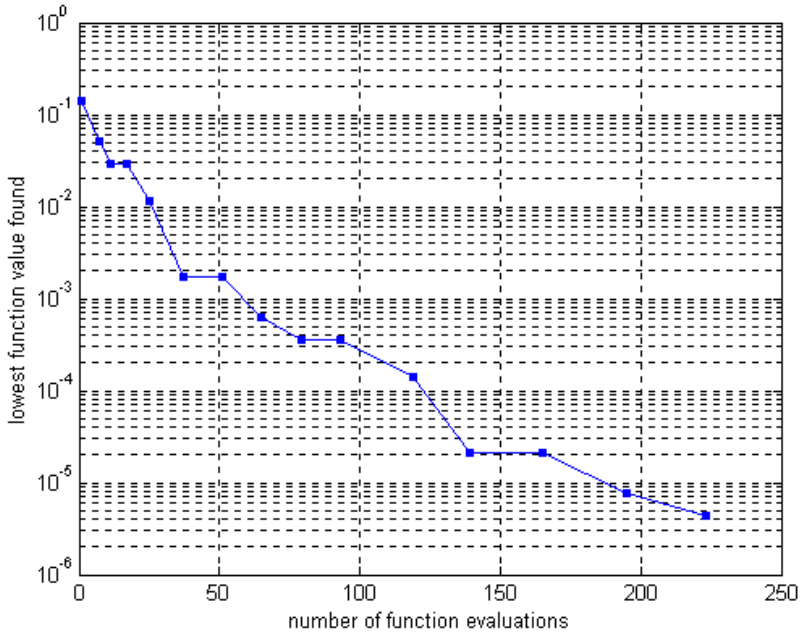


Fig. 15 Convergence property for 3-D case

Figures 16 and 17 show the results for the 5-D case. Figure 16 shows the variations of the five variables of the sample point with the lowest function value during the 21 iterations (535 function evaluations). The global minimum point found by DIRECT at the final stage is (0.1049383,

0.3024961, 0.5, 0.6975309, 0.8991770), and the value at the minimum point is 3.725719E-05. The exact minimum point for this 3-D function is (0.1, 0.3, 0.5, 0.7, 0.9) and the minimum value is 0. Figure 17 shows the convergence property of DIRECT for this 5-D case.

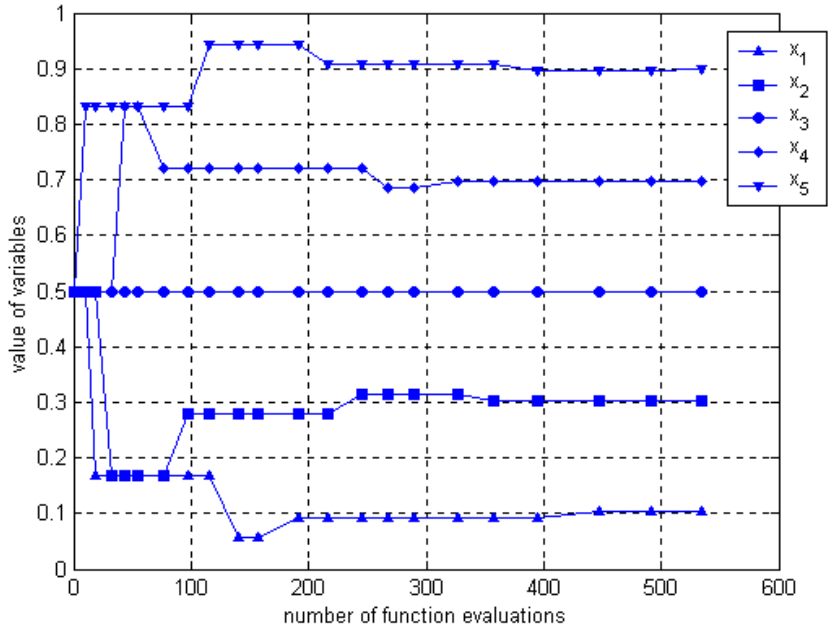


Fig. 16 Variations of variables for 5-D case

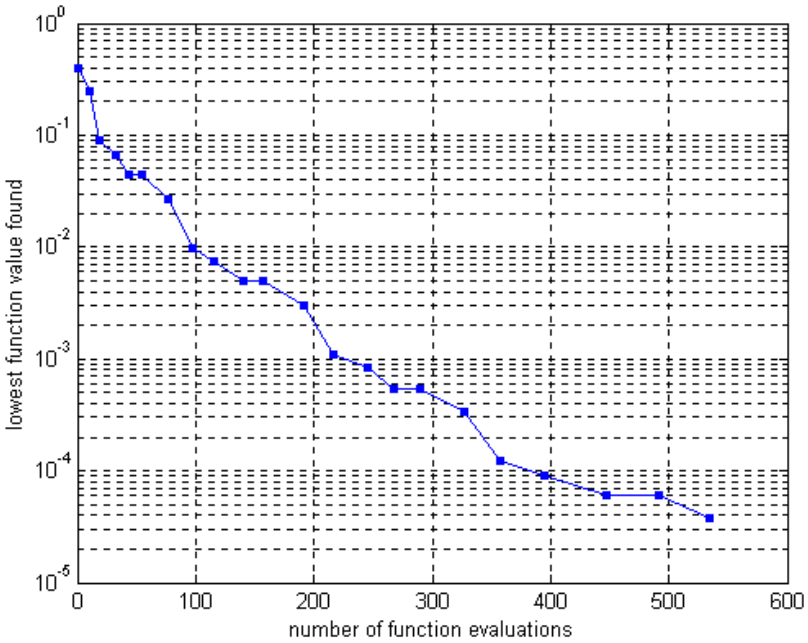


Fig. 17 Convergence property for 5-D case

Similarly, we show the results for the 10-D case in Figs. 18 and 19.

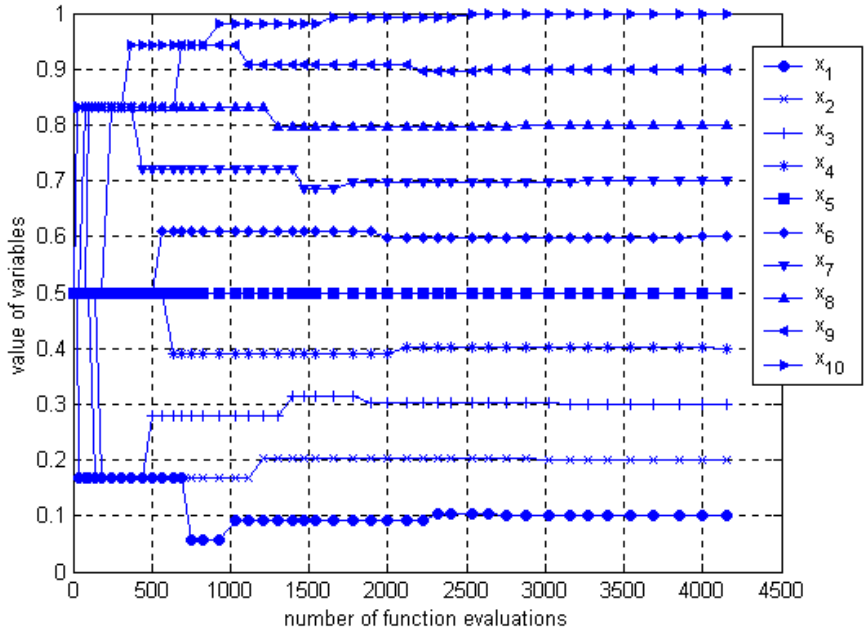


Fig. 18 Variations of variables for 10-D case

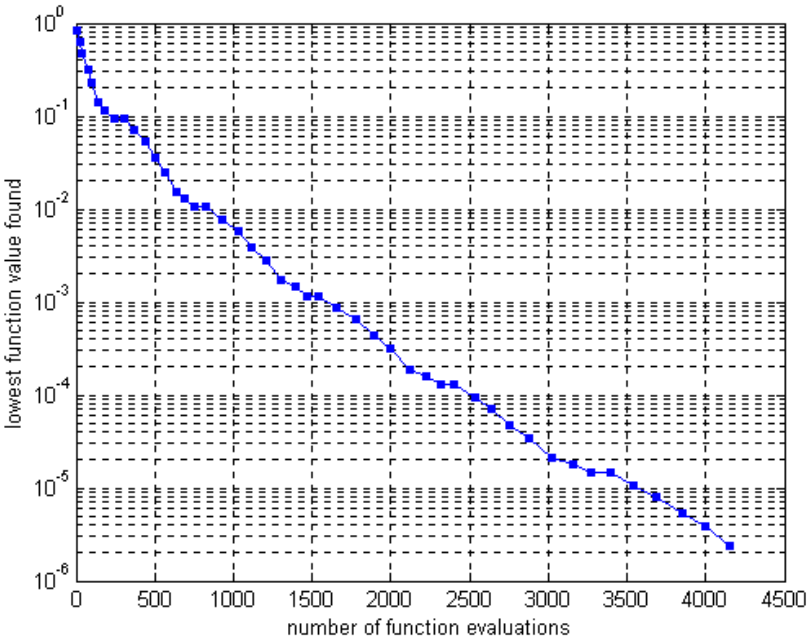


Fig. 19 Convergence property for 10-D case

Figure 18 shows the variations of the ten variables for the sample point with the lowest function value during the 45 iterations (4157 function

evaluations). The global minimum point found by DIRECT at the final stage is (0.1008230, 0.1995885, 0.2997257, 0.3998628, 0.5, 0.6001372, 0.7002743, 0.8004115, 0.8991770, 0.9993141), and the value at the minimum point is 2.35s096E-06. The exact minimum point for this 3-D function is (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0) and the minimum value is 0. Figure 19 shows the convergence property of DIRECT for this 10-D case.

3.2 Special testing function cases

We investigated two special cases here. The first one is a 2-D constant function $F(x_1, x_2) = 100$, where $x_1, x_2 \in [0,1]$. Figure 20 shows the results after 30 iterations (81 function evaluations).

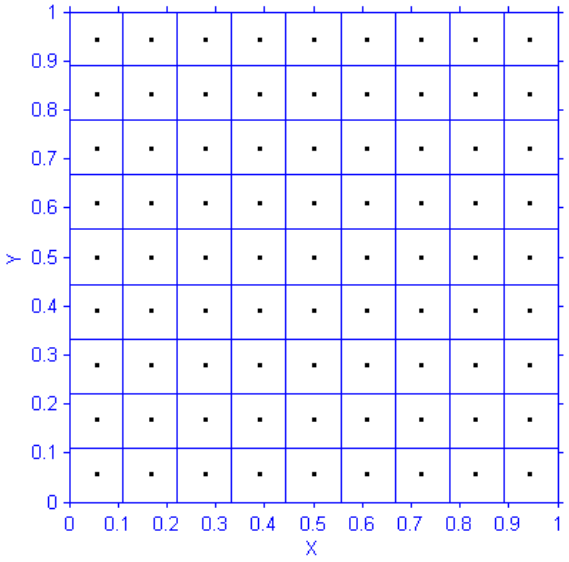


Fig. 20 Results for 2-D constant functions

The uniform distribution of sample points in Fig. 20 reflects the global search property of the DIRECT algorithm.

The second case is one with multiple global minima. The function we considered here 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 minima. If we normalize the range of variables x_1 and x_2 into $[0,1]$, then the five global minima are (0.55, 0.5), (0.50743, 0.52010), (0.52013, 0.51437), (0.57987, 0.48563) and (0.59257, 0.47990).

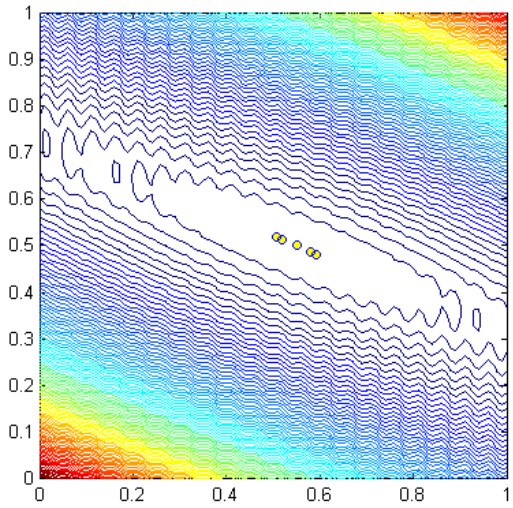


Fig. 21 Contour lines of Branin function

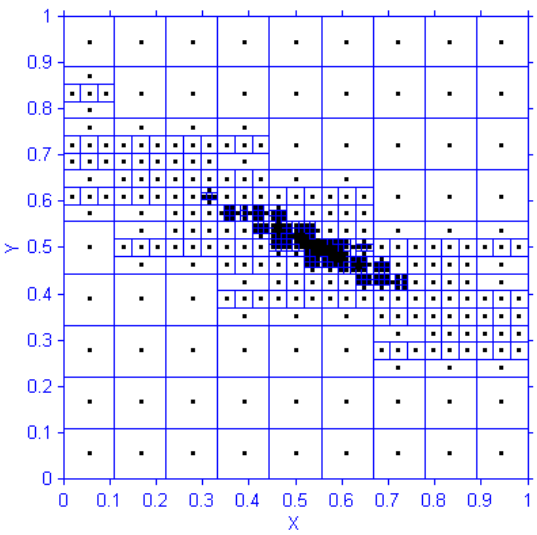


Fig. 22 Optimization results

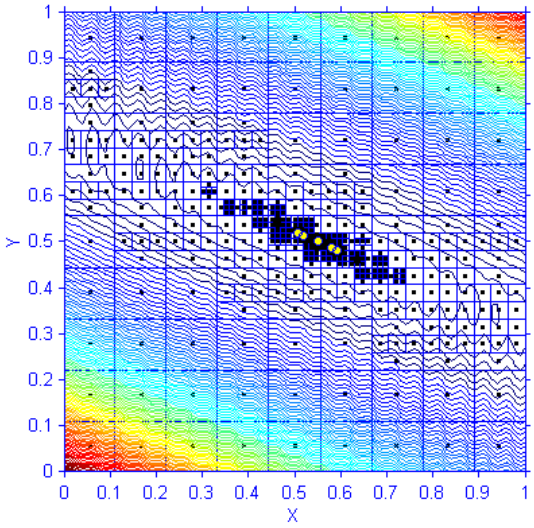


Fig. 23 Combination of contour lines and optimization results

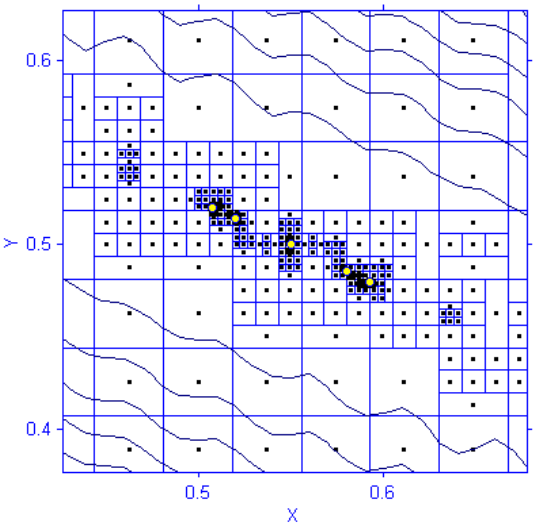


Fig. 24 Local zoom-in around the global minima points

Figure 21 shows the contour lines of the Branin function. The five circular dots represent the five global minima. Figure 22 shows the optimization results after 32 iterations (1029 function evaluations). We

combine Fig. 21 and 22 in Fig. 23. Figure 24 shows the local zoom-in of Fig. 23. From Fig. 24 we see that the sample points cluster around all five global minimal points. So the DIRECT algorithm is capable of finding multiple global minima.

3.3 Tough testing function cases

The so-called "tough" functions are the ones whose global minima are difficult for the optimization technique to find. This is mostly caused by either multiple local minima or a wide "flat" area around the global minimum point. These features will make the optimization difficult since it's easy to be trapped at a local minimum, or, conversely, because it's hard to reach the global minimum point.

We investigated two functions here. The first function is the Rosenbrock function, a standard test function in optimization theory. The Rosenbrock function is defined as: $F(x_1, x_2) = 100\,(x_1 - x_2^2)^2 + (1 - x_1)^2$, where $x_1, x_2 \in [-2.048, 2.048]$.

If we normalize the range of variables x_1 and x_2 into $[0,1]$, then its global minimum point is (0.74414, 0.74414) and the global minimum is 0. It's hard to find the global minimum point of this function because the global minimum point is located at a long narrow flat valley.

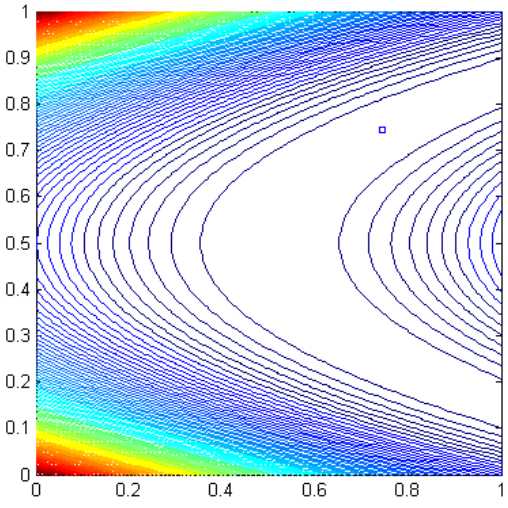


Fig. 25 Contour lines of function

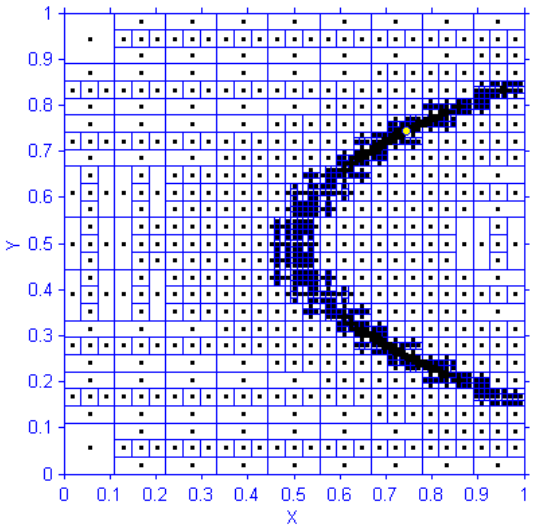


Fig. 26 Optimization results

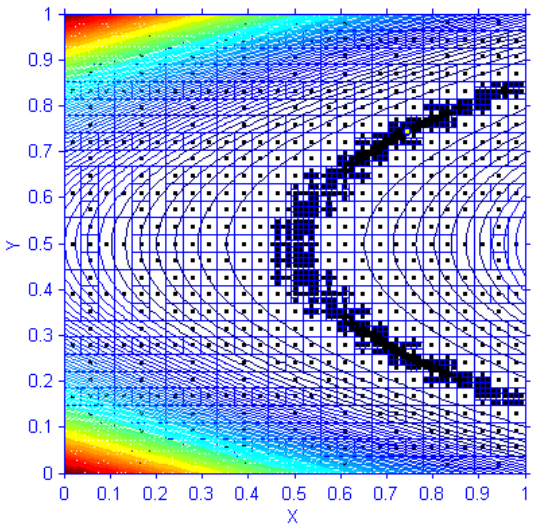


Fig. 27 Combination of contour lines and optimization results

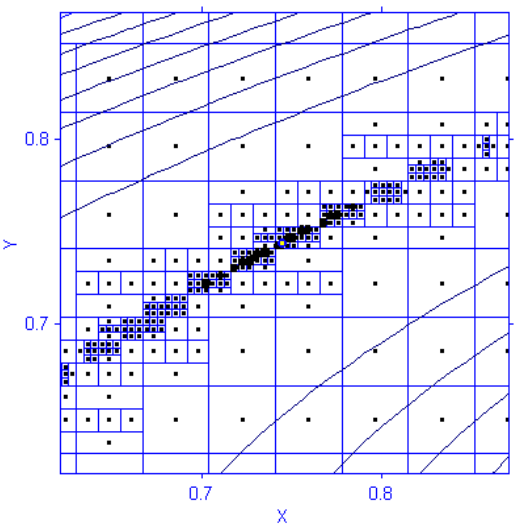


Fig. 28 Local zoom-in around the global minimum point

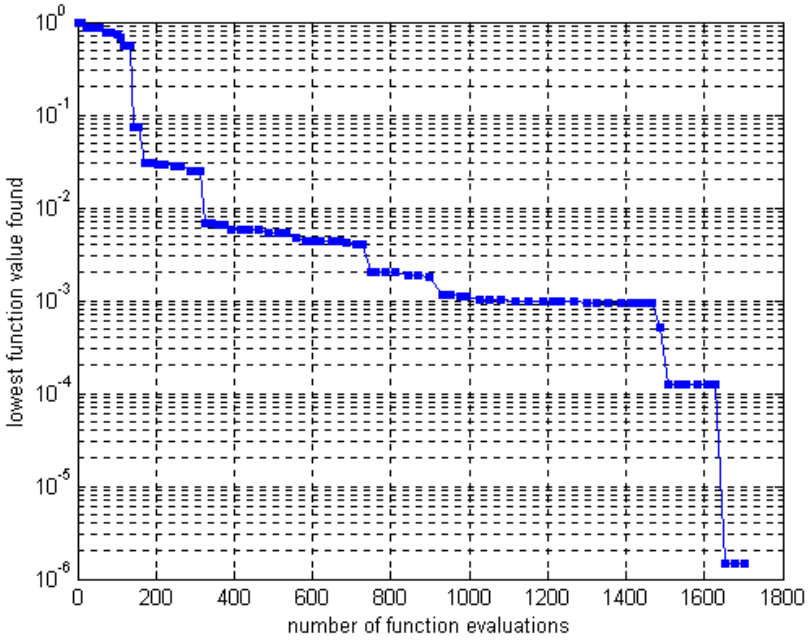


Fig. 29 Convergence property for Rosenbrock function case

Figure 25 shows the contour lines of the Rosenbrock function. The square dot represents the global minimum point. Figure 26 shows the results of optimization after 83 iterations (1701 function evaluations). The global minimum point found by DIRECT at the final stage is (0.7441710, 0.7441701), and the value at the minimum point is 1.472094E-06.

We combined Fig. 25 and Fig. 26 in Fig. 27. Figure 27 clearly shows that the sample points are clustered in the valley. Figure 28 shows the local zoom-in of Fig. 27 around the global minimum point. The minimum point found by DIRECT (circular dot) almost coincides with the exact global minimum point (square dot). The convergence property of DIRECT for this function is shown in Fig. 29.

The second function we considered here is an extremely "nasty" function called the Shubert function. This function not only has 9 global minima, but it also has a total number of 400 local minimum points! The Shubert function is defined as follows:

$$F(x_1, x_2) = -(\sum_{i=1}^{5} i \sin((i+1)x_1 + i) + \sum_{j=1}^{5} j \sin((j+1)x_2 + j)) ,$$

where $x_1, x_2 \in [-10, 10]$. If we normalize the range of variables x_1 and x_2 into $[0,1]$, then its 9 global minimum points are:

(0.1612712, 0.1612712),
(0.1612712, 0.4754305),
(0.1612712, 0.7895897),
(0.4754305, 0.1612712),
(0.4754305, 0.4754305),
(0.4754305, 0.7895897),
(0.7895897, 0.1612712),
(0.7895897, 0.4754305),
(0.7895897, 0.7895897).

The global minimum is -24.062499. The 3-D surface and 2-D contour of the Shubert function are shown in Figs. 30 and 31, respectively. The nine solid dots in Fig. 31 denote the nine global minimum points.

Fig. 32 shows the optimization results after 327 iterations (2505 function evaluations). The global minimum value found by DIRECT at the final stage is -24.06146. The centers of the nine circles in Fig. 32 represent the positions of the nine global minimum points. The tiny dots represent the

sample points. Clearly, the sample points cluster around all nine global minimum points. That means that the DIRECT algorithm found all the global minima of the Shubert function.

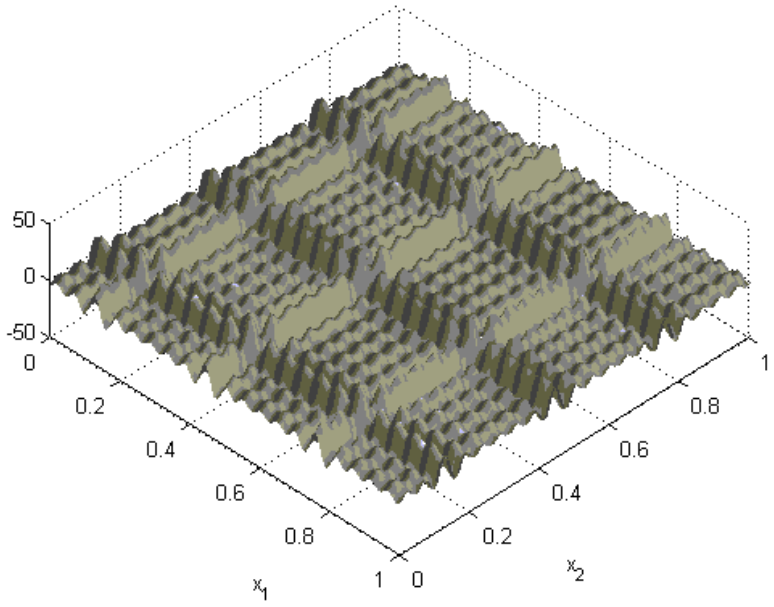


Fig. 30 3-D surface of the Shubert function

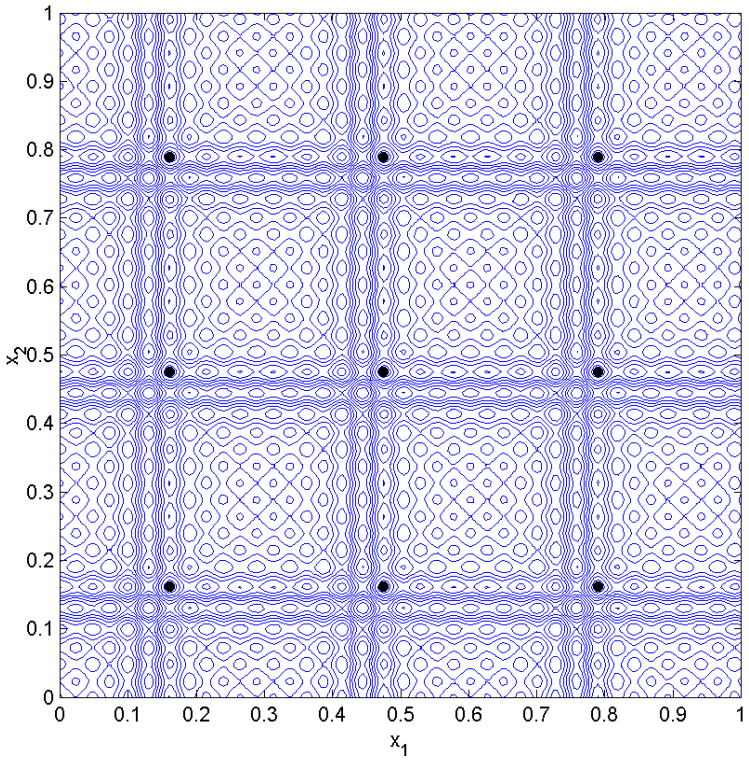


Fig. 31 Contour lines of the Shubert function

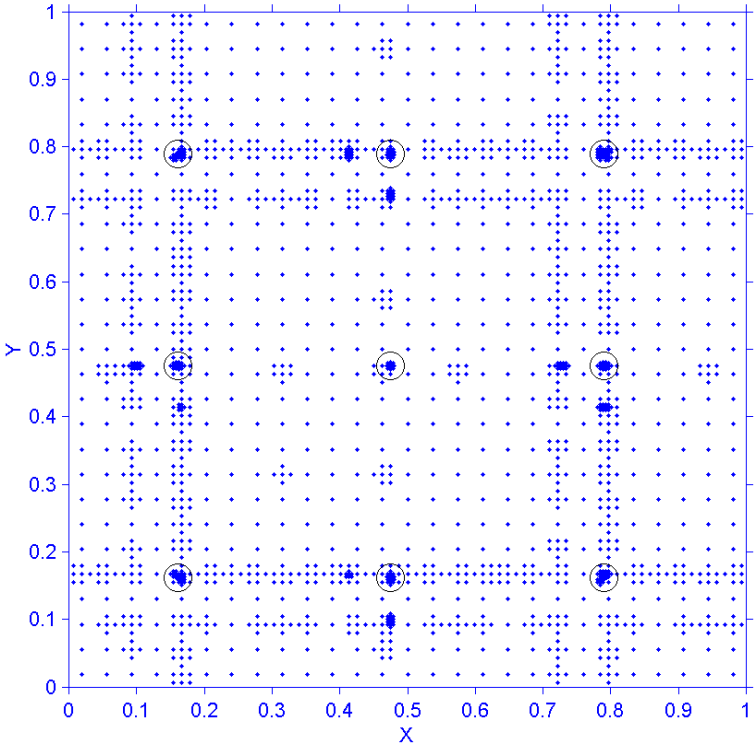


Fig. 32 Optimization results for Shubert function

3.4 Summary of the numerical experiments

We performed extensive numerical experiments with general, special, and "tough" test functions. The DIRECT algorithm found the global minimum points for all the test functions, and it is capable of finding multiple global minima, even for the extremely tough functions like the Shubert function.

As proved by D. R. Jones et. al.[3], the DIRECT algorithm is guaranteed to converge to the globally optimal function value if the objective function is continuous or at least continuous in the neighborhood of a global optimum. This property results from the fact that, as the number of iterations goes to infinity, the points sampled by DIRECT form a dense subset of the unit hypercube. For more details please refer to Ref. 3.

The numerical experiments also show that the DIRECT algorithm has a very fast convergence rate. In other words, to obtain the same low objective function value, DIRECT uses far fewer sample points than would be requested by other algorithms. The fast convergence rate of DIRECT is the primary motivation for us to try to apply it to our slider ABS optimization.

4. IMPLEMENTATION OF DIRECT INTO ABS OPTIMIZATION

4.1 Structure of the optimization program

To implement the optimization, two closely integrated parts are needed. One is the optimization algorithm, and the other is the solver. The optimization algorithm is used to generate different sample designs, which are then sent to the solver for calculation of the parameters. The algorithm generates new sample points based on the results of the function evaluations.

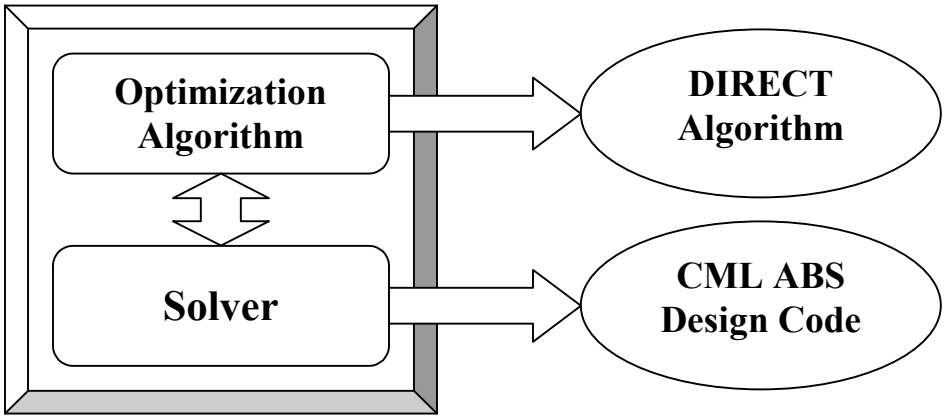


Fig. 33 Structure of the optimization program

The optimization algorithm used here is the DIRECT algorithm. The solvers are the CML slider ABS design programs, which were developed by the Computer Mechanics Laboratory at the University of California at Berkeley, including the CML rectangular mesh solver Quick419 and the CML triangular mesh solver Quick5.

4.2 Flow chart of the optimization program

The flow chart of the optimization program is shown in Fig. 34, where **N** represents the number of the designs, $\mathbf{N_{max}}$ the maximum number of designs prescribed, **I** the number of iterations and $\mathbf{I_{max}}$ the prescribed maximum number of iterations.

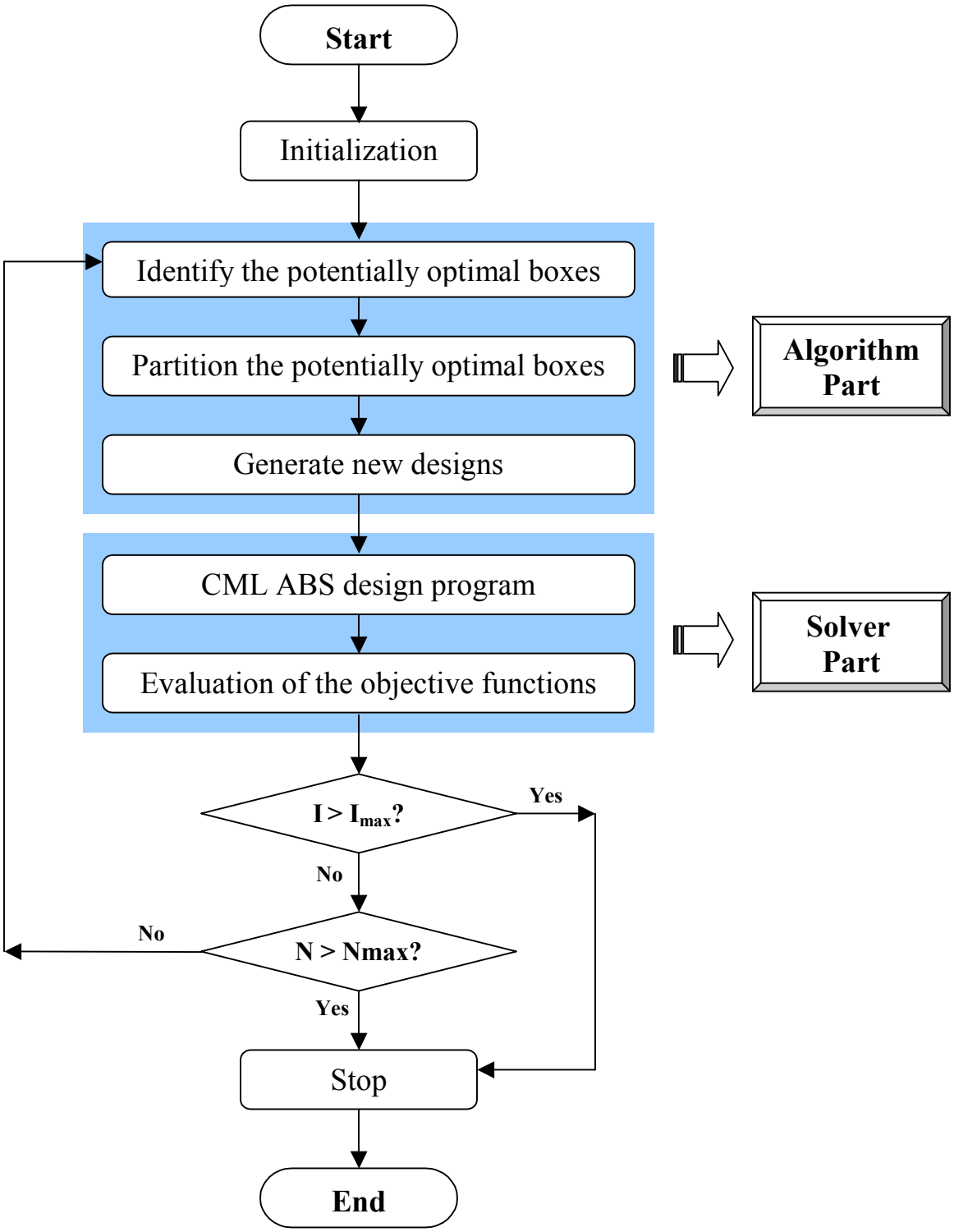


Fig. 34 Flow chart of the CML optimization program using DIRECT

5. AIR BEARING DESIGN OPTIMIZATION PROBLEM

We define the optimization problem to be: given a prototype slider ABS design, optimize it to get uniform flying heights near the target flying height and at a flat roll profile, and if possible, increase its air bearing stiffness.

Here the NSIC 7nm flying height slider is used as the prototype slider. The rail shape and the 3-dimensional rail geometry are shown in Figs. 35 and 36, respectively.

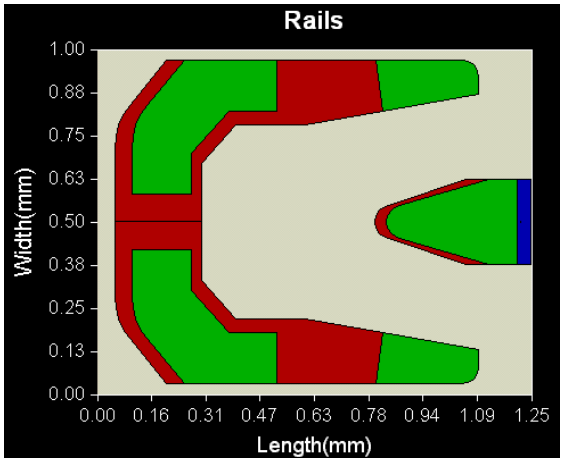


Fig. 35 Rail shape of the initial ABS design

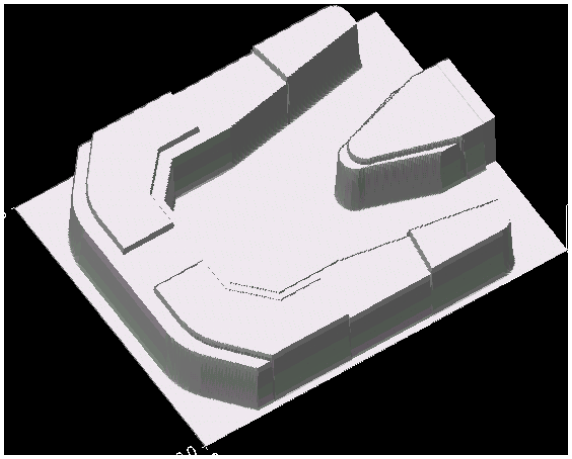


Fig. 36 3-D rail shape of the initial ABS design

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

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

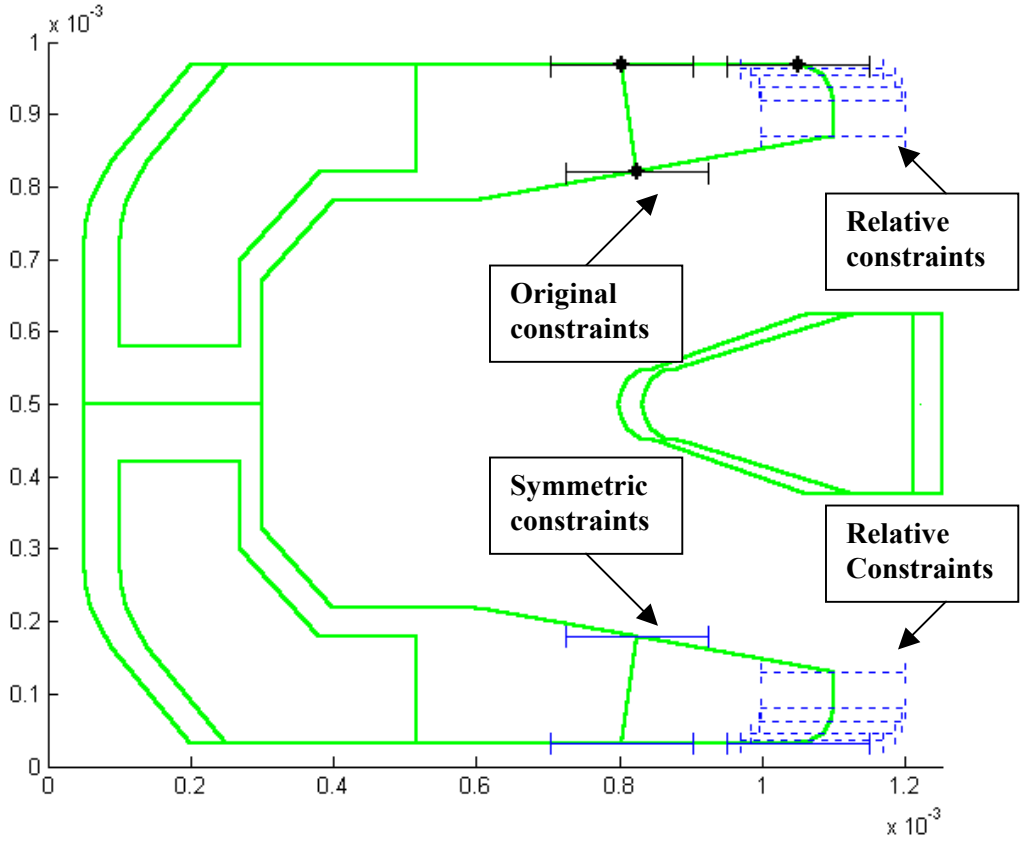


Fig. 37 Constraints defined on the initial design

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"[2]. The constraints are shown in Fig. 37, and can also be found in the user's manual.

6. SIMULATION RESULTS

Using the initial design, constraints, and objective function, we carried out the optimization using the DIRECT algorithm. Figure 38 shows the variation of the objective function values during the optimization process.

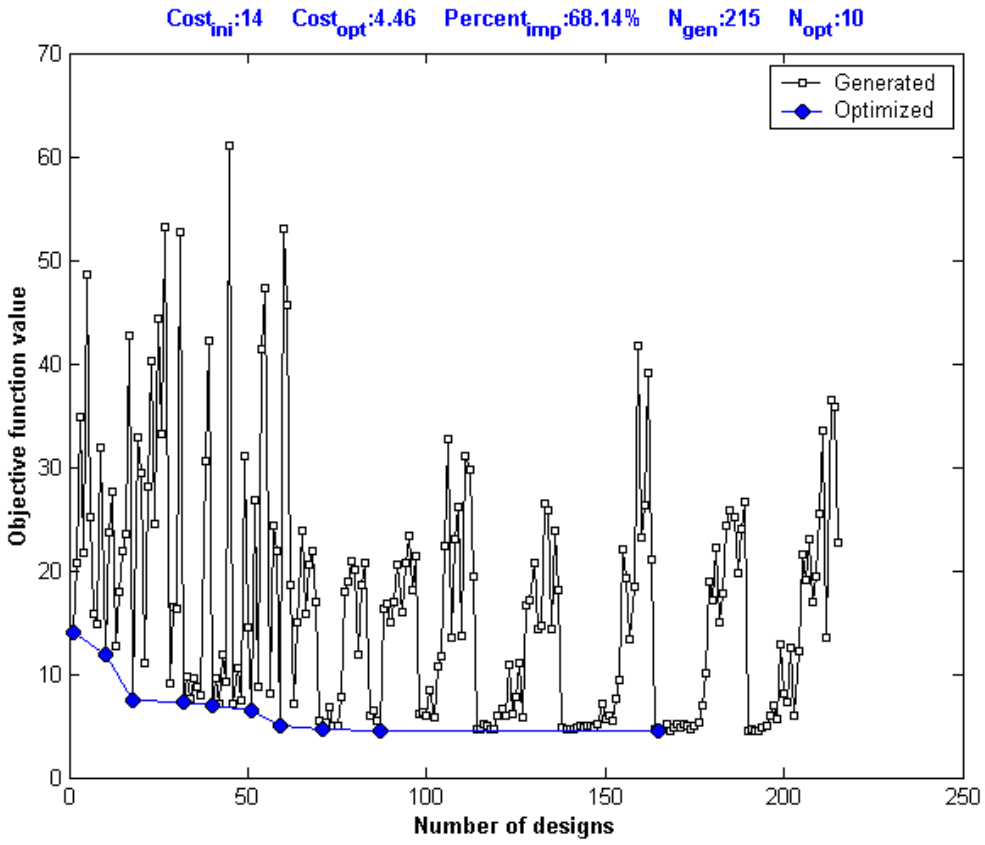


Fig. 38 Variation of the objective function value

In the above figure, $Cost_{ini}$ is the initial objective function value, and $Cost_{opt}$ is 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 designs generated and optimized respectively.

The small squares represent the sample designs generated during the process. The dark circles represent the optimized designs. The optimized designs are the ones with the best-so-far objective function values. The lower the objective function value, the better the design.

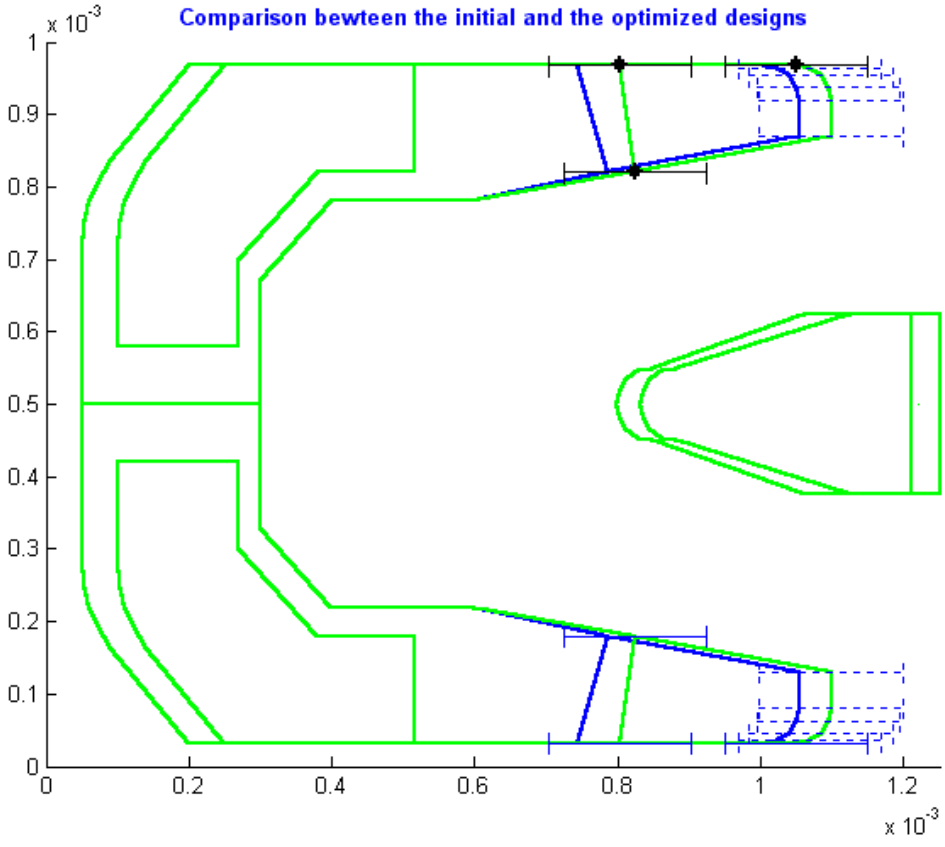


Fig. 39 Optimization results

The comparison between the initial and optimized designs is shown in Fig. 39, in which the light lines show the rail shape of the initial design and the dark lines show the rail shape of the optimized design.

Figure 40 shows the variation of the objective function terms for all of the best-so-far designs generated during the optimization process.

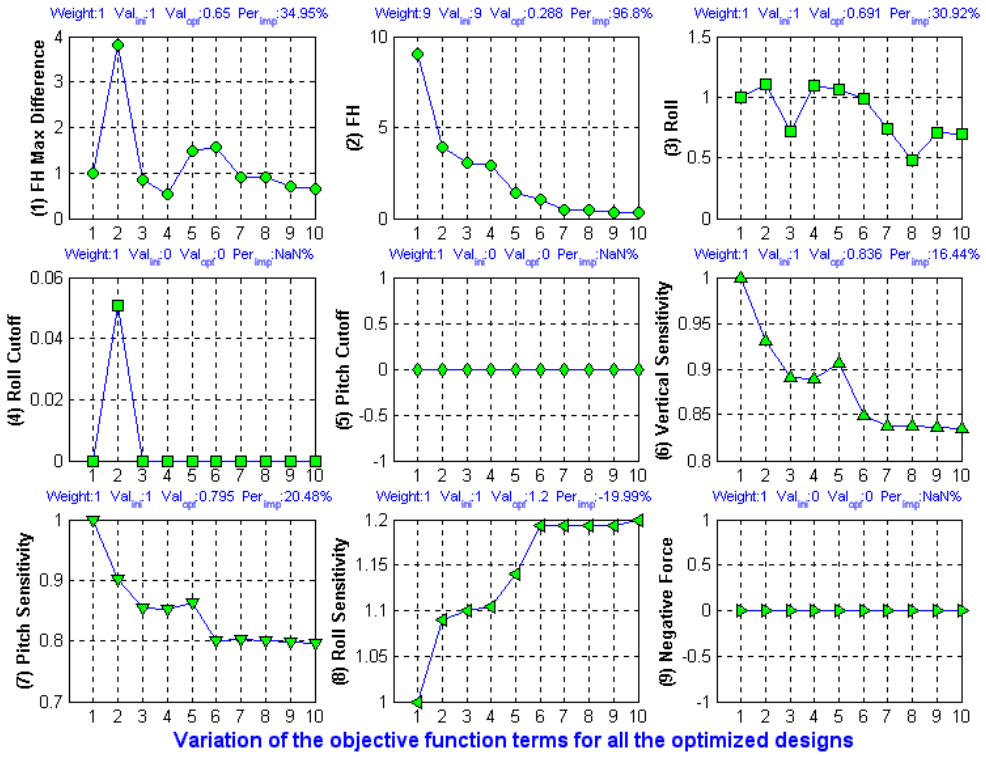


Fig. 40 Variation of the objective function terms

From Fig.40 we see that the DIRECT algorithm provides impressive minimization in the Flying Height term, i.e. the 2nd objective function term, which was weighted more heavily. There was also improvement in the roll term as well as some improvement of the Vertical Sensitivity and the Pitch Sensitivity terms. However, the Roll Sensitivity did not improve. Some objective terms such as the Pitch cutoff term and Negative Force term remained zero for all of the optimized designs. The combinatorial effects are the minimization of the total value of the objective function. By minimizing the multi-objective cost function we obtained the final optimized designs.

The variations of the slider performance parameters for all the best-so-far designs are shown in Fig. 41.

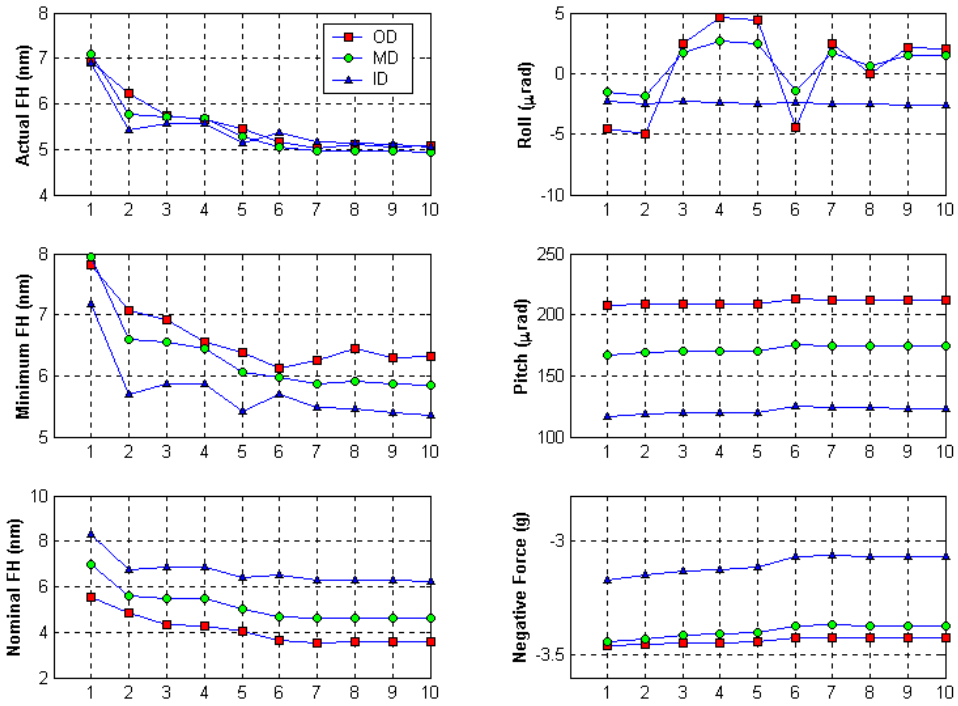


Fig. 41 Variations of the slider performance parameters

It is clear that the optimized ABS design has fairly constant flying heights around the target flying height of 5nm. Also, it maintains a reasonably flat roll profile.

In addition to using the DIRECT algorithm, we also used the Adaptive Simulated Annealing (ASA) algorithm to carry out the optimization for the same problem. Figure 42 shows the convergence comparison between ASA and DIRECT. It shows that the DIRECT algorithm clearly has a much higher convergence rate than the ASA algorithm. In this case, it only takes about 100 sampling designs for DIRECT to converge to the optimal design, while it takes more than 600 sampling designs for ASA!

Also note that the objective function value of the final optimized design obtained by using DIRECT is 4.46. For ASA the final optimized design's objective function value is 4.74. Since a smaller objective function value means a better design, the DIRECT algorithm obtained a better optimized design than ASA algorithm.

Obviously the DIRECT algorithm outperforms the ASA algorithm in this case. Here we presented only the convergence comparison results for the

ASA and DIRECT. More details about the comparisons between these two algorithms will be given in another CML technical report.

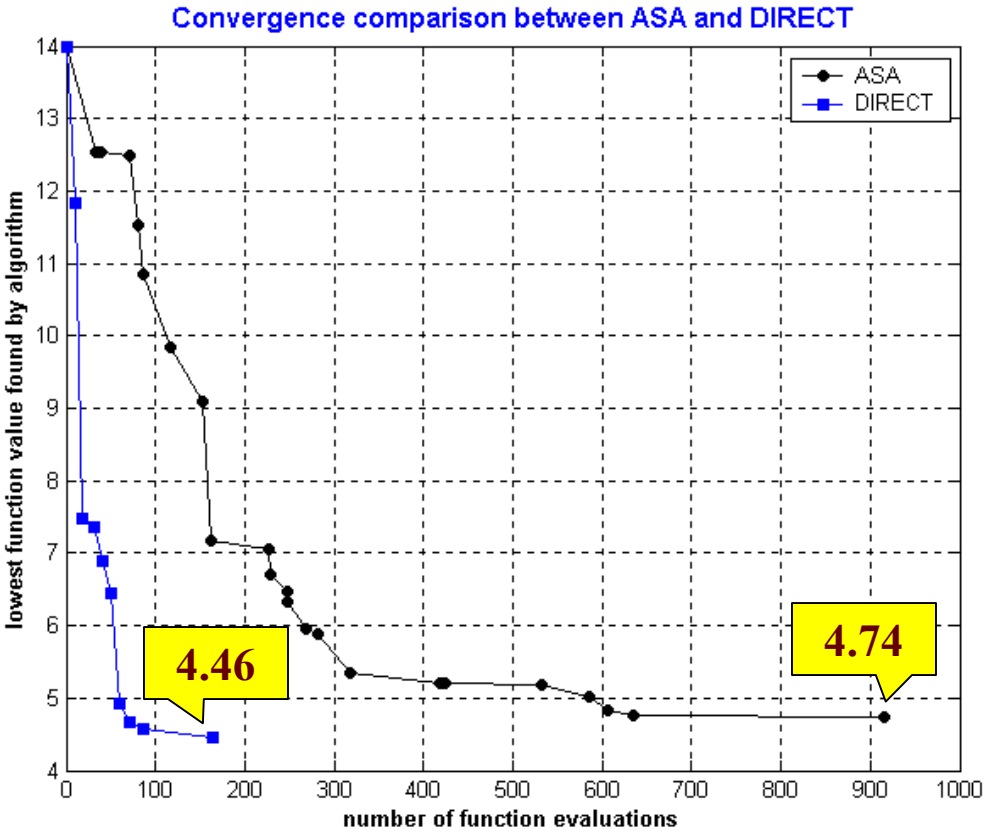


Fig. 42 Convergence comparison between ASA and DIRECT

7. CONCLUSION

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

We did extensive numerical experiments for the DIRECT algorithm with general, special, and "tough" test functions. The DIRECT algorithm found the global minimum points for all the testing functions, and it is also capable of finding multiple global minima, even for some extremely tough functions.

D. R. Jones et. al.[3] proved that the DIRECT algorithm is guaranteed to converge to the globally optimal function value if the objective function is continuous or at least continuous in the neighborhood of a global optimum. Our numerical experiments also verify that conclusion, and show that the DIRECT algorithm has a very fast convergence rate.

Slider ABS designs that satisfy very strict multi-objective goals are of great importance for magnetic hard disk drives. Finding such optimal designs is a strongly non-linear problem. Use of the DIRECT optimization technique, which is a global deterministic optimization method, provides the optimized designs automatically for a given initial design and constraints. When different weights are put on different objective function terms, the objective function steers the designs to its goals.

The DIRECT algorithm was shown to produce an optimized ABS design with greatly improved performance, i.e., uniform flying heights around the target flying height, flat rolls and improved stiffness. This illustrates that the DIRECT algorithm is quite suitable for the optimization of ABS designs.

The convergence comparison between the DIRECT algorithm and the ASA algorithm, which is a global stochastic optimization technique used previously for the slider ABS optimization, shows that the DIRECT algorithm clearly has a much higher convergence rate than does the ASA algorithm. In other words, DIRECT obtains the global optimal design in many fewer sample designs than does ASA. In the case examined about 100 sampling designs were needed for DIRECT to converge to the optimal design, while it takes more than 600 sampling designs for ASA.

Also note that the objective function value of the final optimized design obtained by using DIRECT is smaller than the one obtained by using ASA. Since smaller objective function value means a better design, the optimized design obtained by using the DIRECT algorithm has the better overall performance.

In summary, the DIRECT algorithm clearly outperforms the ASA algorithm in our test case. More details about the comparison between these two algorithms will be given in another CML technical report.

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., 2000, "*Optimization of Slider Air Bearing Shapes using Variations of Simulated Annealing*", Technical Report 2000-10, Computer Mechanics Laboratory, University of California at Berkeley.

2. 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.

3. 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.

4. 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.

5. Preparata, F. P. and Shamos M. I., 1985, "*Computational Geometry: An Introduction*", Springer-Verlag, New York, New York.

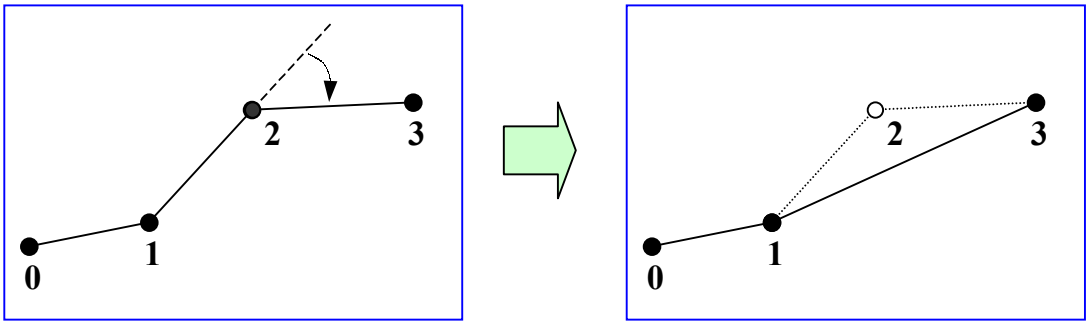
Appendix A

Graham's scan [5]

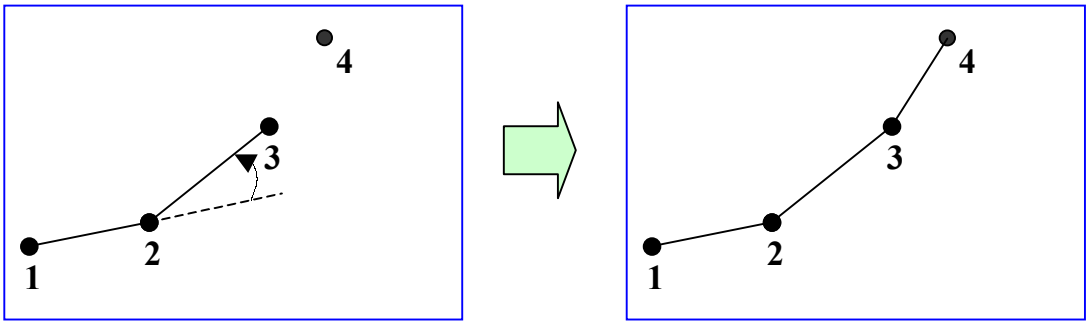
Graham's scan is an algorithm which can find the convex hull of a set of *m* arbitrary points in $O(m\log_2 m)$ times. If the points are already sorted by their abscissas, it will only require $O(m)$ times.

The basic procedure for a Graham's scan in our case is:

1. Sort all the data points according to their abscissas.
2. Find a starting point, which is the sample point with the lowest function value.
3. Start with that point, pick up three continuously neighboring points, judge if they form a "left-turn" or "right-turn", and then decide what action to take. This is shown in Fig. A1.



Right turn: (1 2 3) → (0 1 3) and point 2 eliminated



Left turn: (1 2 3) → (2 3 4)

Fig. A1 Illustration of "Right turn" and "Left turn"

4. Repeat this whole process until the algorithm finishes scanning all the data points.

The following pictures demonstrate the basic processes to obtain the convex hull for a given set of data points.

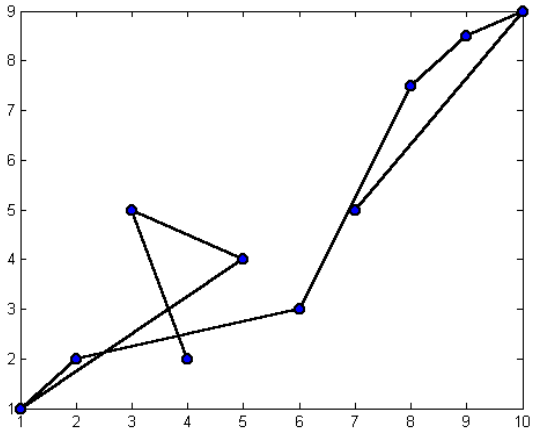


Fig. A2 Initial set of data points

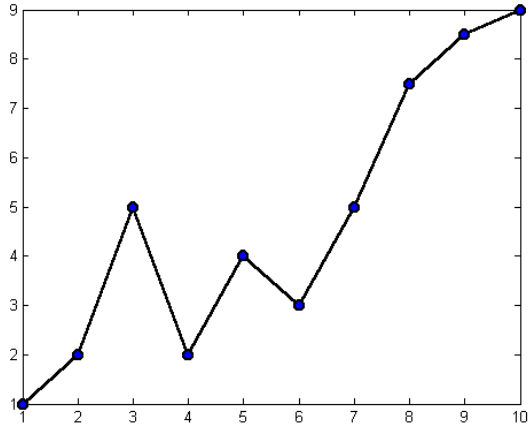


Fig. A3 Results after sorting

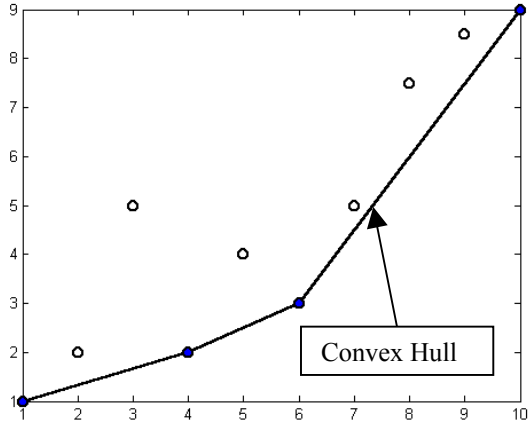


Fig. A4 Results of Graham's scan