Global Optimization of Slider Air Bearing Design

By

Hong Zhu

B. E. (Chengdu University of Science and Technology) 1990

M. Eng. (University of Science and Technology of China) 1993

M. S. (University of California, Berkeley) 2001

A dissertation submitted in partial satisfaction of the

Requirements for the degree of

Doctor of Philosophy

In

Engineering-Mechanical Engineering

In the

GRADUATE DIVISION

Of the

UNIVERSITY of CALIFORNIA, BERKELEY

Committee in charge:

Professor David B. Bogy, Chair
Professor Kyriakos Komvopoulos
Professor John Strain

Fall 2002

The dissertation of Hong Zhu is approved:

Chair	Date
	Date
	Date

University of California, Berkeley

Fall 2002

Global Optimization of Slider Air Bearing Design

Abstract

Global Optimization of Slider Air Bearing Design

by

Hong Zhu

Doctor of Philosophy in Engineering-Mechanical Engineering

University of California, Berkeley

Professor David B. Bogy, Chair

Hard disk drives continue to increase in areal data density. Densities as high as 1 Tbit/in^2 are now being considered. The extremely high areal density requires air bearing sliders with ultra-low flying height (FH), less than 10nm. At this very low FH, slider air bearing surface (ABS) designs must satisfy very strict performance goals, such as uniform FHs and roll profiles across the disk. By using modern optimization techniques, it is possible to optimize slider ABS designs according to multiple design goals.

This dissertation focuses on the development and application of global optimization techniques to the problem of hard disk drive slider air bearing design. Both the stochastic global optimization techniques (Simulated Annealing algorithm family) and the deterministic global optimization techniques (DIRECT algorithm and its locally biased variations as well as some modified versions) are investigated and applied to the slider air bearing surface (ABS) design optimization problem.

We first give a detailed description of the Simulated Annealing family, including the Standard Simulated Annealing (BA) algorithm, the Fast Simulated Annealing (FA) algorithm and the more powerful Adaptive Simulated Annealing (ASA) algorithm. These Simulated Annealing algorithms are then applied to slider ABS optimization. These three main members of the simulated annealing family are shown to produce similar optimized ABS designs with greatly improved performance, i.e. uniform flying heights around the target flying height, flat rolls and improved stiffness. This illustrates that the simulated annealing algorithm is quite suitable for the optimization of ABS designs. Among them, the ASA was found to be the most efficient and robust scheme due to its fastest cooling schedule and its unique adaptive re-annealing mechanism.

An introduction of the new deterministic DIRECT algorithm is then presented through various numerical experiments and slider ABS optimization case studies. The comparison between ASA and DIRECT shows that DIRECT has a much faster convergence rate than ASA. Thus DIRECT can find the global minimum more quickly. It is shown that the DIRECT algorithm outperforms the ASA algorithm, so it is considered to be more suitable for the slider ABS optimization than ASA. Therefore, this dissertation focuses primarily on the DIRECT algorithm.

To further improve the efficiency of the DIRECT algorithm, we then propose three locally biased variations of the standard DIRECT algorithm. These variations generally have faster convergence rates than the standard DIRECT algorithm, and they may dramatically reduce the time needed to find the global minimum in some situations.

This dissertation also reports on two modifications to the DIRECT algorithm: one to handle tolerance (minimum side lengths) and one to deal with hidden constraints. The results

show that defining the manufacturing tolerance and hidden constraints can save calculation time for a fixed number of designs generated, and thus improve the efficiency of the DIRECT algorithm.

To make the slider ABS optimization program more flexible, new geometric constraints are introduced. The slider ABS sensitivity optimization issue is also discussed.

Two new versions of the CML Air Bearing Optimization Program based on the Simulated Annealing algorithm and the DIRECT algorithm have been developed and they have been successfully applied to the ultra-low FH slider design and optimization problem for the Extremely High Density Recordings (EHDR) project of the National Storage Industry Consortium (NSIC).

Prof. David B. Bogy, Chair

To my parents

Zhiye Zhu and Yongfang Ni

and my wife Chuanping

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEGMENTS

First of all, I would like to express my genuine gratitude to my advisor, Professor David B. Bogy, for his constant guidance and support throughout my Ph.D. research. I consider this experience unforgettable and invaluable. Professor Bogy is a role model for me not only for his earnest attitude toward scientific research; for his generosity and consideration toward other people; but also for his dignified personality. I would also like to thank Professors Kyriakos Komvopoulos and John Strain for reviewing this thesis. This research was supported by the Computer Mechanics Laboratory (CML) and partially supported by the Extremely High Density Recordings (EHDR) project of the National Storage Industry Consortium (NSIC).

I am grateful to Dr. Matthew Arnold O'Hara, for his helpful suggestions and advice when I continued this research project. The great job he did previously in this project was a real help to me. I also want to thank Dr. C. Singh Bhatia of IBM for managing the Tribology group of the NSIC project and for his encouragement and assistance. Mr. Thomas Pitchford of Seagate gave me many valuable suggestions on the design of sliders during my participation in the NSIC project. In addition, the NSIC slider ABS designs we designed and optimized were finally fabricated at Seagate through his assistance and help.

I would also like to thank my colleagues in CML for their friendship and support. In particular, I am thankful to Ryan Grisso for his friendship and great help on teaching me how to become a system administrator. I also appreciate many useful discussions with Walton Fong, Brian Thornton and Ling Huang. In addition, I treasure the friendship with Takeshi

Watanabe and Toru Watanabe. I learned a lot about Japanese culture and language from them. My stay in CML has been a really enjoyable experience.

Any acknowledgments would be incomplete without mentioning my parents. They have always tried their best to provide me with the best education even during very hard times. Any of my successes is impossible without their great love and encouragement. I also want to deeply thank my wife, Chuanping He for her love, patience and support at all times.

Chapter 1

INTRODUCTION

1.1 SLIDER DESIGN AND THE DEVELOPMENT OF THE HARD DRIVE

Since the first commercial hard disk drive, which was the IBM product RAMAC (Random Access Method of Accounting and Control) with an areal density of 2000 bits/in.^{2} or a linear density of 100 bits/in. (BPI) and 20 tracks/in. (TPI) was delivered in 1957, the areal density increased at an average annual growth rate of about 39% from 1957 to 1991. The rate increased to 65% from 1991 to 1997 due to the use of many new technologies such as magnetoresistive (MR) read heads, smaller diameter disks and smoother disk surfaces which allow lower flying heights. Figure 1.1 shows the areal density growth for the hard disk. IBM has achieved an areal density of 35.3Gb/in.^{2} in a laboratory demonstration, and areal densities of 100Gb/in.^{2} have been demonstrated by some companies at the end of year 2001. Recently the hard disk industry has begun discussion of HDD areal density of 1Tb/in.^{2}.

The increase of the areal density is of great economic and technical interest, and it has a huge impact on the price of the hard disk drive. Along with the increase of the areal density, the price per megabyte has been lowered from more than $200 per megabyte in 1980 to the present cost of about .13 cents per megabyte. Figure 1.2 shows the price history over the past 20 years, showing a precipitous drop in the last few years.

To obtain higher areal density, the head-to-media spacing or flying height (FH) must be lowered. Figure 1.3 shows the relationship between the head-to-media spacing and the areal density.

In order to obtain the areal density as high as 100 Gb/in.^{2}, the flying height of the slider will need to be below 10 nanometers. At these very low flying heights, the performance of the slider becomes quite crucial and requires tighter control of flying height uniformity, roll profile flatness across the disk, etc.

To meet the increasingly rigorous multi-objective slider performance criteria, modern optimization techniques can be used to solve this strongly nonlinear problem. The advantages of using numerical optimization are:

- Reduction of design time.
- Optimization provides a systematized logical design procedure.
- Many design variables and constraints, which are not easy to visualize, can be handled by the optimization program.
- Generally some design improvement can be realized by using optimization.
- The optimization process is not biased by intuition or experience in engineering. Therefore it has a higher probability of obtaining improved nontraditional designs.
- Optimization requires a minimal amount of human-machine interaction.

In summary, high efficiency, simplicity and automaticity are the reasons to use optimization techniques when designing slider air bearing for HDD.

1.2 SURVEY OF THE OPTIMIZATION TECHNIQUES

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

If there are no constraints set on the variables, the problem is said to be unconstrained. Otherwise it is called a constrained problem. The constraints reduce the size of the set S by limiting the number of feasible solutions. For both the unconstrained and constrained problems, the mathematical description is basically the same. The only difference is the size of the set S. The constrained problem has a smaller set S, but the constraints also make the problem much more complicated.

If the objective function f(x) and the constraints in a problem are both linear combinations of the independent variables, the problem is referred to as a linear programming problem. For this kind of problem, the solution can easily be found by the

Simplex method or the interior point method. If the objective function is quadratic in nature, while its constraints are of the linear form, we can decompose the problem and then find its solution by the Simplex method. If the objective function f(x) has a definite form, we can always find its minimum in a predictable way. Unfortunately, many problems of interest are nonlinear. Air bearing design optimization problems have many objective functions with no distinct forms and the constraints can take many possible forms as well.

For nonlinear problems, the most difficult issue is multiple optima. The objective function may have many minima and the one found might not be the absolute minimum point. Instead, we only have the assurance that it is a local optimum, i.e. a feasible point x* that is an optimal solution to the problem whose feasible region is the intersection of the original region and some neighborhood of x*. It is important to avoid finding local optima. That's why we adopt global optimization techniques, which search for the absolute minimum point of the objective function over the given search space.

There are many global optimization algorithms, which can be divided into two fundamentally different categories, i.e. <u>deterministic</u> algorithms and <u>stochastic</u> algorithms. In deterministic algorithms every new sample point is chosen in a definite way and no random components are involved. In stochastic algorithms random elements are introduced to generate the new sample points. Deterministic algorithms can handle definite objective functions very well. The stochastic algorithms can be applied to a wider range of objective function types, but usually with slower convergence rate.

O'Hara (1997) compared two types of the stochastic algorithms: the Genetic algorithm and Simulated Annealing (SA) algorithm, for their applicability to the problem of hard drive component optimization. Through empirical and theoretical study he found SA is superior for this problem. He developed an accompanying software package (The CML Air Bearing Optimization Program Version 1.5) that utilizes an implementation of SA. This software package allows for the multi-objective optimization of air bearing slider and suspension designs.

1.3 OVERVIEW

This dissertation focuses on the development and application of global optimization techniques to the problems of hard disk drive slider air bearing design. Both stochastic global optimization techniques (the Simulated Annealing algorithm family) and deterministic global optimization techniques (the DIRECT algorithm and its locally biased variations as well as some modified versions) have been investigated and applied to the slider air bearing surface (ABS) design optimization problems.

Through substantial numerical experiments and ABS optimization case study, we show that the new deterministic DIRECT algorithm clearly outperforms the stochastic SA algorithm. Therefore, between these two algorithms, our investigation focuses primarily on the DIRECT algorithm.

Two new versions of the CML Air Bearing Optimization Program have been developed. Version 2.0 is developed for the Simulated Annealing algorithms (including the Standard Boltzmann Simulated Annealing (BA), Fast Simulated Annealing (FA) and the more powerful Adaptive Simulated Annealing (ASA) algorithms). Version 3.0 is developed for the DIRECT algorithm. It includes three locally biased variations and the modified versions of the standard DIRECT algorithm. It also features new geometric constraints.

These two new CML Air Bearing Optimization Programs have been successfully applied to the ultra-low FH slider design and optimization problem for the Extremely High Density Recordings (EHDR) project of the National Storage Industry Consortium (NSIC). Figures 1.4 and 1.5 show the pictures of the actual NSIC 7nm and 5nm FH sliders fabricated by the Seagate Corporation based on two optimized slider ABS designs obtained by using the new ABS optimization programs. Table 1.1 shows the numerical results of the two optimized slider ABS designs.

In this dissertation, after this general introduction we present details of the Simulated Annealing algorithm family in Chapter 2. Chapter 2 is an extension to O'Hara's previous work. Here we introduce the more efficient ASA algorithm to the slider ABS optimization problem. In Chapter 3, we introduce the new deterministic DIRECT (acronym for DIviding RECTangles) algorithm through various numerical experiments and slider ABS optimization case studies. We present three locally biased variations of the standard DIRECT algorithm in Chapter 4. Those variations generally have higher convergence rates than the standard DIRECT algorithm, and they may dramatically reduce the time needed to find the global minimum in some situations. In Chapter 5 we address two modifications made to the DIRECT algorithm. They can further improve the efficiency of the DIRECT algorithm and they are especially useful in the slider ABS optimization case. We present a detailed comparison between ASA and DIRECT in Chapter 6. The results clearly show that DIRECT is a much better choice than ASA in slider ABS optimization. We introduce some new geometric constraints in Chapter 7, which can make the slider ABS optimization program more flexible. We also discuss the slider ABS sensitivity optimization issue in Chapter 7. Finally, Chapter 8 concludes and summarizes all the material presented.

	7nm FH NSIC slider ABS design			5nm FH NSIC slider ABS design		
	OD	**MD**	**ID**	**OD**	**MD**	**ID**
FH (nm)	6.91	7.11	6.90	5.12	4.74	5.12
Roll (μrad)	-4.55	-1.54	-2.27	0.79	0.39	-3.15
Pitch (μrad)	207.8	167.3	116.2	265.0	220.9	162.3

Table 1.1 Summary of the simulation results for the NSIC 7nm and 5nm FH sliders

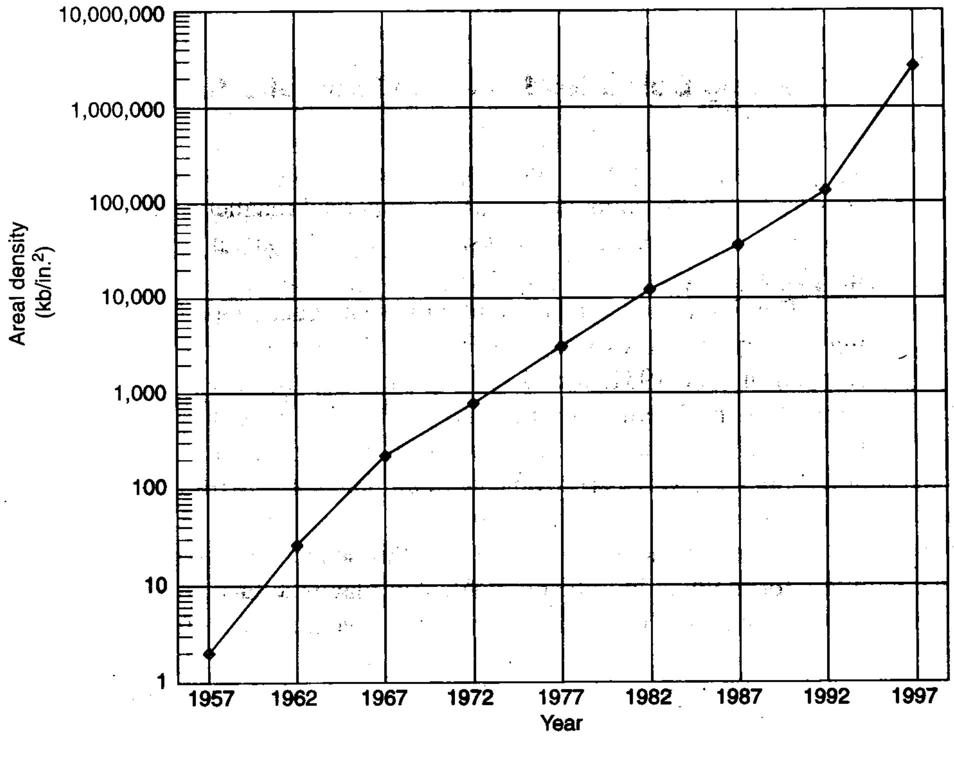


Fig.1.1 Hard magnetic disk areal density growth (Daniel and Clark, 1999)

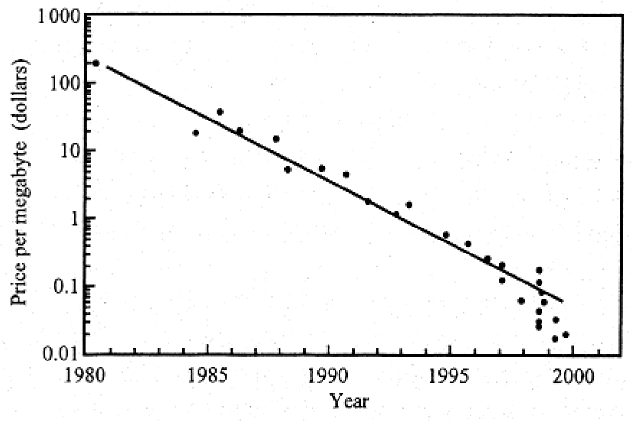


Fig. 1.2 Price history of hard disk product (Thompson, 2000)

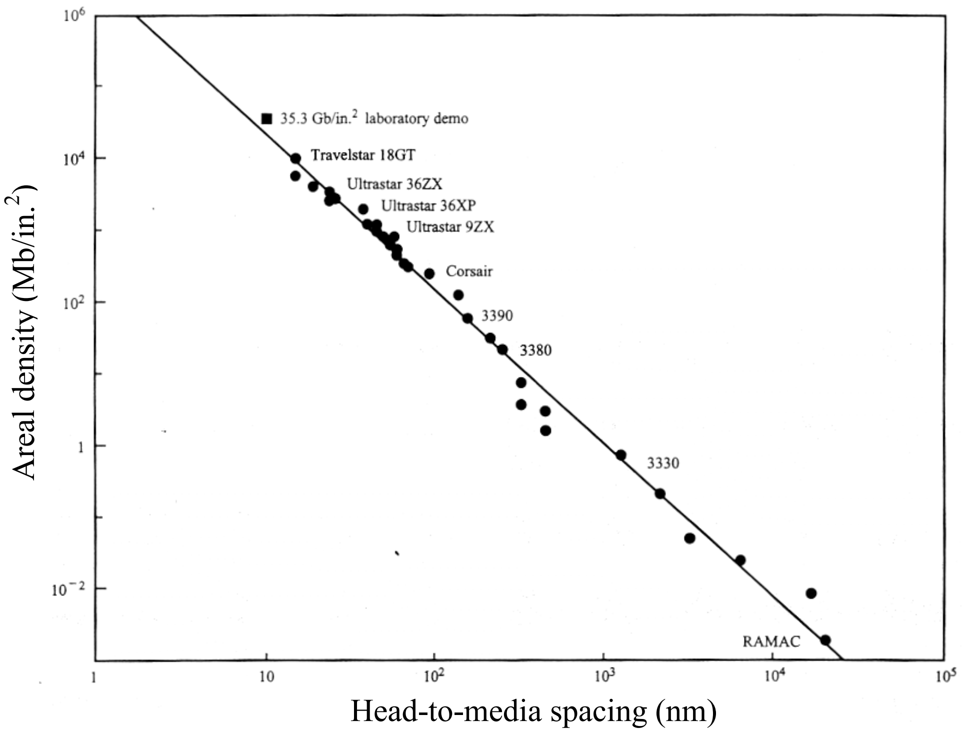


Fig. 1.3 Head-to-media spacing vs. areal density for IBM hard drives (Thompson, 2000)

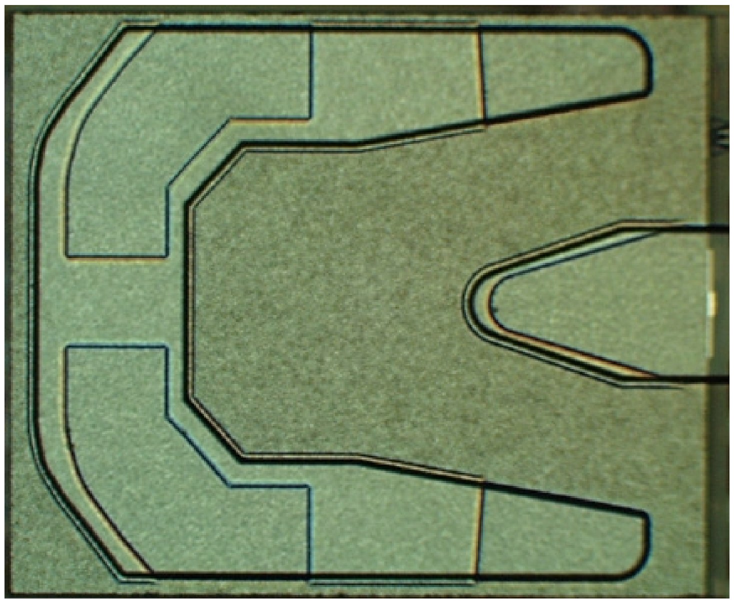


Fig. 1.4 7nm FH NSIC slider fabricated by Seagate

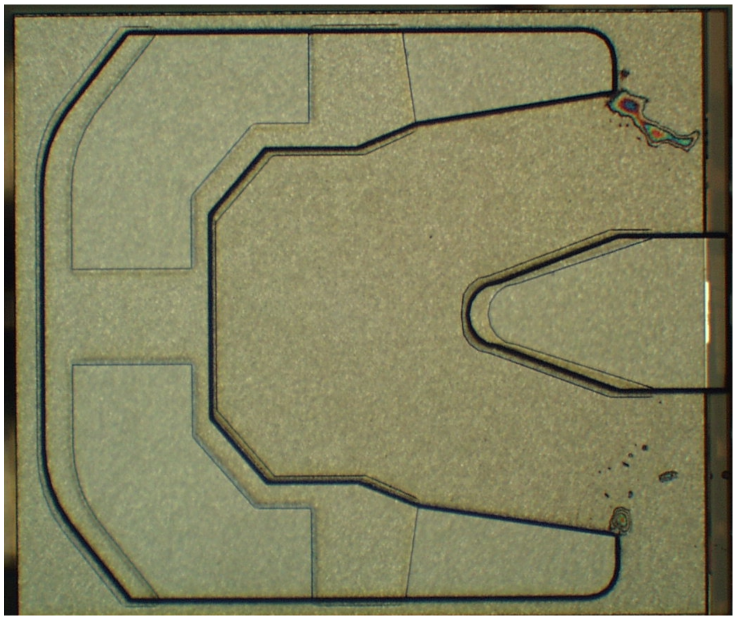


Fig. 1.5 5nm FH NSIC slider fabricated by Seagate

Chapter 2

SIMULATED ANNEALING ALGORITHM

2.1 INTRODUCTION

O'Hara (1997) applied the simulated annealing optimization technique to the problems of air bearing slider and suspension design. He also compared the Simulated Annealing (SA) algorithm and the Genetic Algorithm (GA) algorithm. Through comparison of these two stochastic methods he found that SA is better suited for slider air bearing optimization.

O'Hara investigated the Standard Simulated Annealing algorithm. Here we continue investigations of the Simulated Annealing family in this chapter. Besides the Standard Simulated Annealing (BA) algorithm, we also investigate the Fast Simulated Annealing (FA) algorithm and the more powerful Adaptive Simulated Annealing (ASA) algorithm. In this chapter, we describe the Simulated Annealing optimization technique and we also compare the performance of BA, FA and ASA through the optimization of a slider ABS design.

2.2 NUMERICAL METHOD

The simulated annealing algorithm is a global optimization technique that is based on the concept of the physical annealing process where the temperature of a system is gradually lowered in order to obtain its lowest energy state.

Simulated annealing, which is a stochastic technique, was developed to handle nonlinear problems that are extremely difficult to minimize. It is known to be a powerful and useful tool for a wide variety of minimization problems of large nonlinear systems. It has also been widely applied in many areas, such as circuit design, chemistry, economics, biology, image processing, statistics etc. Research on the simulated annealing technique has also become intensified in recent years.

One of the most important elements of all simulated annealing algorithms is the Metropolis rule, which was developed by Metropolis et al. in 1953. We now briefly describe this important rule.

Recall that optimization is a process for minimizing an objective or cost function $E(\underline{\mathbf{x}}^{\mathbf{i}})$. Here E is a function of the vector $\underline{\mathbf{x}}$ where its components x_n are derived from a certain set in a search space. The superscript i represents different states or designs generated during the process. The Metropolis rule incorporates the following three stages:

1) Given a starting design $\underline{\mathbf{x}}^{i}$ with cost function $E(\underline{\mathbf{x}}^{i})$, a small perturbation to $\underline{\mathbf{x}}^{i}$ is made to obtain a new design $\underline{\mathbf{x}}^{j}$ according to a probability function $g_T(\underline{\mathbf{x}}^{i})$.

2) Computation of the cost function difference between the two designs by $\Delta E = E(\underline{\mathbf{x}}^j) - E(\underline{\mathbf{x}}^i)$.

3) Decide whether or not to accept the new design. There are two cases:

   a. If $\Delta E \le 0$, the new design is always accepted.

   b. If $\Delta E > 0$, the new design is accepted with the probability $h(\Delta E) = e^{-\frac{\Delta E}{T}}$, where T is the cost function temperature.

The procedure is repeated while the temperature T is gradually lowered. If the annealing procedure is carried out properly, the system is expected to converge to the global minimum state. By accepting states with relatively higher cost function values according to some probability, which is called the "hill-climbing" technique shown in the Fig. 2.1, the Metropolis rule can help the process avoid getting trapped in a local minimum point.

Several simulated annealing algorithms have developed, such as the Standard Boltzmann Annealing (BA) (Pincus, 1970; Cerny, 1982), Fast Cauchy Annealing (FA) (Szu and Hartley, 1987) and Adaptive Simulated Annealing (ASA) (Ingber, 1989). The basic idea of all these algorithms is the same. The main differences are the selection of the probability functions $g_T(\underline{\mathbf{x}})$ and $h(\Delta E)$, and the different cooling schedules resulting from those two probability functions. There are three components in all these simulated annealing algorithms:

- $g_T(\underline{\mathbf{x}})$: The probability density function in the state space of D parameters $x=\{x_n, n=1,D\}$, where the subscript T represents the temperature.
- $h(\Delta E)$: The probability function for acceptance of new cost functions given the most recent prior value.
- T(k): The annealing "temperature" for step k, which is also referred to as the cooling schedule. It includes the parameter temperature and the cost temperature, which affect the perturbation of the previous state and the acceptance probability of the new state respectively. These two temperatures comply with the same cooling schedule. The cooling schedule is actually a result of the probability functions $g_T(\underline{\mathbf{x}})$ and $h(\Delta E)$.

2.2.1 Standard Boltzmann Annealing (BA)

The BA was first introduced as a stochastic method for implementing large dimension path integrals for statistical physics (Metropolis et al, 1953). The method was developed for handling general minimization problems. The kernel of the algorithm is based on the probability density derived from Gaussian Markovian systems. The Boltzmann distribution is given by

$$g_T(x) = (2\pi T)^{\frac{-D}{2}} e^{\frac{-x^2}{2T}},$$

where $x \equiv \frac{\eta - \eta_0}{b-a} \in [-1,1]$. The range $[a, b]$ is the constraint set on certain points. η_0 represents the previous point and η is the new point. So x can be viewed as the normalized deviation of η from the previous point η_0. T is the parameter temperature, which is the measure of the perturbation of the Boltzmann distribution g in the D-dimensional space η.

The 3-D and 2-D views of the Boltzmann distribution are shown in Figs. 2.2 and 2.3 respectively. These two figures show the change of the distribution as the temperature is lowered from 1 to 0.1.

The acceptance probability of the BA is:

$$h(\Delta E) = \frac{1}{1 + e^{\frac{\Delta E}{T}}},$$

where ΔE signifies the "energy" difference between the present and previous values of the energies (considered here as cost functions), i.e., $\Delta E = E_{k+1} - E_k$. T is the cost temperature. The lower the cost temperature, the lower the acceptance probability.

Given $g_T(\underline{\mathbf{x}})$, it has been proved (Geman et al, 1984) that a global minimum of E(x) will be obtained if T is selected to be not faster than

$$T_k = \frac{T_0}{\ln(k)}.$$

The strict proof will not be reproduced here. Instead, we only present a heuristic demonstration to show that T_k will give a global minimum of E(x). In order to statistically assure that any point in x-space can be sampled "Infinitely Often in annealing Time" (IOT), it suffices to prove that the products of probabilities of not generating a state x IOT for all annealing-times successive to time k_0 yield zero,

$$\prod_{k=k_0}^{\infty}(1-g_k)=0 \quad ,$$

which is equivalent to:

$$\sum_{k=k_0}^{\infty} g_k = \infty \quad .$$

If we put the expression of T_k into $g_T(\underline{\mathbf{x}})$, then we obtain

$$\sum_{k=k_0}^{\infty} g_k \geq \sum_{k=k_0}^{\infty} e^{-\ln k} = \sum_{k=k_0}^{\infty} \frac{1}{k} = \infty \quad .$$

2.2.2 Fast Cauchy Annealing (FA)

The methodology of BA can be extended for use with any function *g* that satisfies $\prod_{k=k_0}^{\infty}(1-g_k)=0$ in conjunction with a proper cooling schedule T(k).

It may be desirable for the function *g* to enable a faster convergence rate. The use of a Cauchy distribution is a good example of how this may be achieved. The Cauchy distribution is defined by the following equation:

$$g_T(x) = \frac{T}{\left(x^2 + T^2\right)^{\frac{D+1}{2}}} \quad .$$

The 3-D and 2-D views of the Cauchy distribution are shown in Figs. 2.4 and 2.5 respectively. These two figures show the change of the distribution as the temperature is lowered from 1 to 0.1.

The simulated annealing procedure using the Cauchy distribution converges to a global minimum with temperature declining not faster than

$$T(k) = \frac{T_0}{k} \quad .$$

Then

$$\sum_{k=k_0}^{\infty} g_k \approx \frac{T_0}{x^{D+1}} \sum_{k=k_0}^{\infty} \frac{1}{k} = \infty \quad .$$

Thus the FA method statistically has an annealing schedule exponentially faster than the method of BA.

2.2.3 Adaptive Simulated Annealing (ASA)

Many physical problems have a D-dimensional parameter space. Different parameters have different finite ranges, given by physical considerations, and different annealing-time-dependent sensitivities, measured by the curvature of the cost function at local minima. BA and FA have g distributions that sample infinite ranges, and there is no provision for considering differences in each parameter-dimension, e.g., different sensitivities might require different cooling rates. Adaptive Simulated Annealing (ASA) was developed to meet the following goals:

- Obtain a solution from a bounded parameter space instead of an unbounded space.
- Use a faster cooling schedule so as to get faster convergence.
- Consider the sensitivities of for each parameter-dimension, i.e., different constraint points should have different cooling rates.

Each of the parameters α^i at annealing time k is bounded within the range

$$\alpha_k^i \in [A_i, B_i] \ .$$

The parameters are generated at each new step by a random variable $x^i \in [-1,1]$ as

$$\alpha_{k+1}^i = \alpha_k^i + x^i (B_i - A_i) \ .$$

Define the generating function as

$$g_T(x) = \prod_{i=1}^{D} g_T^i(x^i) = \prod_{i=1}^{D} \frac{1}{2(|x^i| + T_i)\ln(1 + \frac{1}{T_i})} \ .$$

The 3-D and 2-D views of this distribution are shown in Figs. 2.6 and 2.7 respectively. These two figures show the change of the distribution as the temperature is lowered from 1 to 0.1.

Then the cumulative probability distribution is

$$G_T(x) = \int_{-1}^{x^1}\int_{-1}^{x^2}\cdots\int_{-1}^{x^D}\prod_{i=1}^{D} g_T^i(x^i)dx^1dx^2\cdots dx^D \equiv \prod_{i=1}^{D} G_T^i(x^i) \quad ,$$

where

$$G_T^i(x^i) = \frac{1}{2} + \frac{\operatorname{sgn}(x^i)}{2}\frac{\ln(1+\frac{|x^i|}{T_i})}{\ln(1+\frac{1}{T_i})} \quad .$$

x^i is generated from the uniform distribution U[0,1] through a parameter $u^i \in U[0,1]$:

$$x^i = \operatorname{sgn}(u^i - \tfrac{1}{2})T_i[(1+\frac{1}{T_i})^{|2u^i-1|} - 1] \quad .$$

For the cooling schedule

$$T_i(k) = T_{0i}e^{-c_i k^{\frac{1}{D}}} \quad ,$$

a global minimum can be obtained, i.e.,

$$\sum_{k=k_0}^{\infty} g_k \approx \sum_{k=k_0}^{\infty}\left(\prod_{i=1}^{D}\frac{1}{2|x^i|c_i}\right)\frac{1}{k} = \infty \quad .$$

The following relations control the parameter c_i:

$$k_{fi} = e^{n_i}$$

$$T_{fi} = T_{0i} e^{-m_i}$$

$$c_i = m_i e^{\frac{-n_i}{D}} \quad ,$$

where k_{fi} and T_{fi} are the final time step and the final parameter temperature.

The acceptance probability function is defined as:

$$h(\Delta E) = e^{\frac{-\Delta E}{T}} \quad .$$

For a multi-dimensional search the cost function value generally has different sensitivities with respect to different parameters. So at any annealing time, it is sensible to attempt to “stretch out” the ranges over which the relatively insensitive parameters are being searched, as compared to the ranges of the more sensitive parameters. In the algorithm, that is equivalent to resetting the annealing time k for the different parameters. This mechanism is referred to as “Re-annealing” or “Adaptation” and it is the reason why the algorithm is called Adaptive Simulated Annealing. The adaptation is accomplished by calculating the energy sensitivities with respect to the different parameters at the most current minimum value of the cost function:

$$s_i = (B_i - A_i) \frac{\partial E}{\partial \alpha^i} \quad .$$

The annealing time k_i is rescaled for each parameter α^i by making use of the maximum sensitivity $s_{\max}=\max(s_1, s_2, \ldots s_D)$:

$$k_i' = \left(\frac{\ln\left(\frac{T_{i0}}{T_{ik}} \frac{s_i}{s_{\max}} \right)}{c_i} \right)^D .$$

T_{i0} is set to unity at the beginning of the search, which is ample to span each parameter dimension.

2.2.4 Summary

Table 2.1 gives a comparison among these three simulated annealing algorithms. Among these three algorithms, ASA has the fastest cooling rate since the temperature is decreased exponentially. That means it has the fastest convergence rate. Also, ASA features an adaptive re-annealing mechanism. This unique feature enables the ASA to set different cooling rates for parameters with different sensitivities. These properties make ASA the most efficient and robust algorithm among the three.

2.3 IMPLEMENTATION OF THE SIMULATED ANNEALING ALGORITHM

2.3.1 Structure of the optimization program

The structure of the optimization program is shown in Fig. 2.8. 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. From the results the algorithm evaluates the quality of the current design and generates a new design based on the result.

The optimization algorithm used here is the simulated annealing algorithm, including the Standard Boltzmann Annealing (BA), Fast Cauchy Annealing (FA) and the Adaptive Simulated Annealing (ASA). The solvers are the CML slider ABS design programs, which were developed by the Computer Mechanics Laboratory of University of California at Berkeley, including the CML rectangular mesh solver Quick419 and the CML triangular mesh solver Quick5.

2.3.2 Flow chart of the optimization program

Figure 2.9 presents a schematic overview of the structure of the optimization program, where **N** represents the number of the designs, $\mathbf{N_{max}}$ the maximum number of designs prescribed, **T** the annealing temperature and $\mathbf{T_{min}}$ the prescribed minimum annealing temperature.

2.4 AIR BEARING DESIGN OPTIMIZATION PROBLEM

The optimization problem defined here is: given a prototype slider ABS design, optimize it to get uniform flying heights near the target flying height and a flat roll profile across the disk. Also increase its air bearing stiffness if possible.

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. 2.10 and 2.11, respectively.

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. Now we want to lower its 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 function or cost function is defined as:

1× *(FH Max Difference term)* + ***9*** × *(FH term)* + ***1*** × *(Roll term)* +

1 × *(Roll Cutoff term)* + ***1*** × *(Pitch Cutoff term)* + ***1*** × *(Vertical Sensitivity term)* +

1 × *(Pitch Sensitivity term)* + ***1*** × *(Roll Sensitivity term)* + ***1*** × *(Negative Force term)* .

So 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 are described in the following.

The *FH Max Difference term* is defined as:

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

where *FH* means the flying heights of the current design and *FH₀* means the flying heights of the initial design (parameters with sub-index 0 are regarded as the parameters of the initial design).

The *FH term* is defined as:

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

where FH_{target} represents the target flying height and n is the number of the evaluation points.

The *Roll term* is defined as:

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

The *Roll Cutoff term* is defined as:

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

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

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

The *Pitch Cutoff term* is defined as:

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

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

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

The *Vertical Sensitivity term* is defined as:

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

The *Pitch Sensitivity term* is defined as:

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

The *Roll Sensitivity term* is defined as:

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

The *Negative Force term* is defined as:

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

where

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

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

Figure 2.12 shows that 3 original constraint points are defined for this case. Original constraints are mutually independent. These 3 constraint points can move along the length direction within the intervals prescribed. To maintain a symmetric ABS design and the fixed local rail shape we also defined the symmetric constraints and the relative constraints. Symmetric constraints require some vertexes to vary symmetrically with the original constraint points. Relative constraints require some vertexes to maintain fixed spatial relationships between the original constraint points and the symmetrical constraint points.

2.5 OPTIMIZATION RESULTS

With the same initial design, constraints and objective function, we carried out the optimization using the BA, FA and ASA respectively.

Figures 2.13, 2.14 and 2.15 show the variation of the objective function values during the optimization process for BA, FA and ASA respectively.

In all of the above figures, Cost_{ini} means the initial objective function value, and Cost_{opt} means the objective function value for the final optimized design. The $\text{Percent}_{\text{imp}}$ 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}, N_{ign}, N_{acc}, N_{opt} in these figures represent the number of the designs generated, ignored, accepted and optimized, respectively.

The dark circles represent the "best-so-far" optimized designs generated during the process. The optimized designs are the ones corresponding to the best-so-far objective function values. We know that the lower the objective function value, the better the design.

The objective function values for the three final optimized designs by using BA, FA and ASA, respectively, are 5.301, 4.574 and 4.387. That means ASA obtained the best-optimized design in this case. Also, ASA had the fastest convergence rate. This can be verified by reference to Fig. 2.16, which shows the cost temperature variation during the optimization process. ASA had the fastest cooling rate while BA had the slowest one.

The so-called "freezing" phenomenon for FA can be observed in Fig. 2.14. After certain stages, in this case, after generating about 200 designs, the fluctuation of the cost function value becomes much smaller. This indicates the parameter temperature is so low that the newly generated designs are very small perturbations from the previously accepted design. Thus the new designs have been "frozen" there. This is due to the fast cooling rate of FA. (This phenomenon is not observed for BA at this stage, since its cooling rate is quite slow) The question arises: if ASA has an even faster cooling rate than FA, why was there no freezing phenomenon for ASA? The answer is that ASA has an adaptive re-annealing process. Different parameters will have different cooling rates according to their different sensitivities. For those parameters with lower sensitivities, their temperatures get raised to let them vary more freely in the following stages. So this mechanism actually lets ASA avoid the "freezing" phenomenon while maintaining its fast cooling rate. Therefore ASA is more efficient and more robust than BA or FA.

The comparison between the initial and optimized designs produced by BA, FA and ASA are shown in the Figs. 2.17 ~ 2.19, respectively, in which the gray lines show the rail shape of the initial design and the dark lines show the rail shape of the optimized design. All these algorithms obtained similar results.

We also show the variation of the objective function terms for the three optimized designs by using BA, FA and ASA in the Figs. 2.20 ~ 2.22, respectively.

All three algorithms provide impressive minimization in the Flying Height term, i.e. the 2nd objective function term, which was weighted more heavily. There was also improvement for the roll term as well as some improvement on the Vertical Sensitivity term and the Pitch Sensitivity term. But the Roll Sensitivity was not improved. 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 comparison of the performance parameters of most concern, i.e. the flying heights and the rolls, are given in the Figs. 2.23 and 2.24 respectively.

It is clear that all of the optimized ABS designs have quite constant flying heights around the target flying height, which is 5nm. Also they all maintain a reasonably flat roll profile. The optimized design obtained by using ASA has the most uniform flying height profile and the best overall performance.

2.6 CONCLUSION

Slider ABS designs that satisfy very strict multi-objective goals are of great importance for the performance of magnetic hard disk drives. This is a strongly non-linear problem.

Use of the simulated annealing optimization technique, which is a global stochastic optimization method, provides the optimized designs automatically for a given initial design and constraints.

By putting different weights on different objective function terms, the objective function steers the design to its goals.

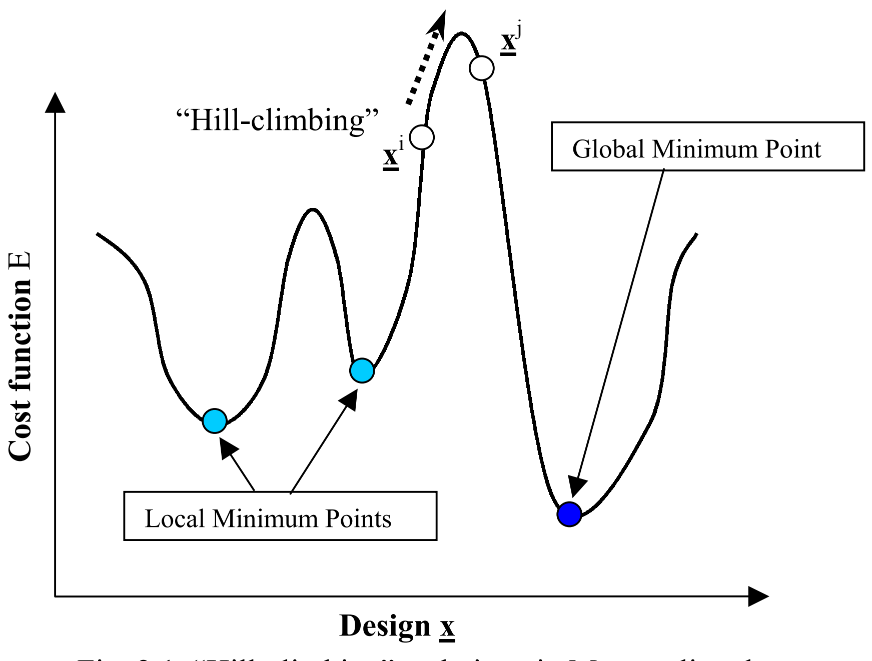
Three main members of the simulated annealing family, namely the Standard Boltzmann Annealing (BA), the Fast Cauchy Annealing (FA) and the Adaptive Simulated Annealing (ASA), were shown to produce similar optimized ABS designs with greatly improved performance, i.e. uniform flying heights around the target flying height, flat rolls and improved stiffness. This illustrates that the simulated annealing algorithm is quite suitable for the optimization of the ABS designs.

The ASA was found to be the most efficient and robust scheme due to its fastest cooling schedule and its unique adaptive re-annealing mechanism. These features gave it the fastest convergence rate and let it effectively avoid the “freezing” phenomenon, which is generally a side-effect of the fast cooling rate. FA also had a faster convergence rate than BA.

Among the optimized designs obtained by using BA, FA and ASA, the one obtained by ASA had the most uniform flying height profiles and the smallest objective function value, which means its design had the best overall performance.

	BA	**FA**	**ASA**
Generating Probability $g_T(x)$	$(2\pi T)^{\frac{-D}{2}} e^{\frac{-x^2}{2T}}$	$\frac{T}{\left(x^2+T^2\right)^{\frac{D+1}{2}}}$	$\prod_{i=1}^{D} \frac{1}{2(\lvert x^i \rvert + T_i)\ln(1+\frac{1}{T_i})}$
Acceptance Probability $h(\Delta E)$	$\frac{1}{1+e^{\frac{\Delta E}{T}}}$	$\frac{1}{1+e^{\frac{\Delta E}{T}}}$	$e^{\frac{-\Delta E}{T}}$
Cooling Schedule $T(k)$	$\frac{T_0}{\ln(k)}$	$\frac{T_0}{k}$	$T_{0i}e^{-c_i k^{\frac{1}{D}}}$

Table 2.1 Comparisons among BA, FA and ASA



$\underline{\mathbf{x}}^j$
"Hill-climbing"
$\underline{\mathbf{x}}^i$
Global Minimum Point
Cost function E
Local Minimum Points
Design $\underline{\mathbf{x}}$

Fig. 2.1 "Hill-climbing" technique in Metropolis rule

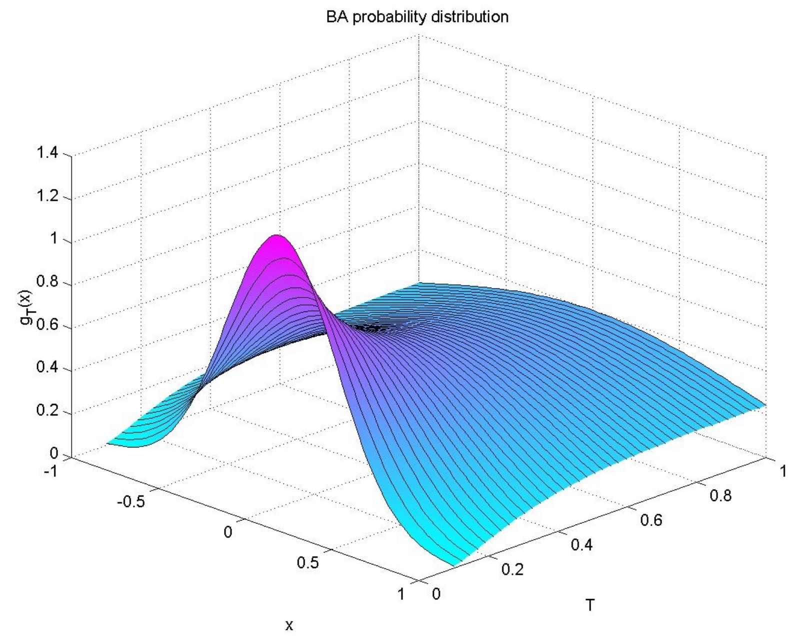


Fig. 2.2 3-D view of the BA probability function $g_T(x)$

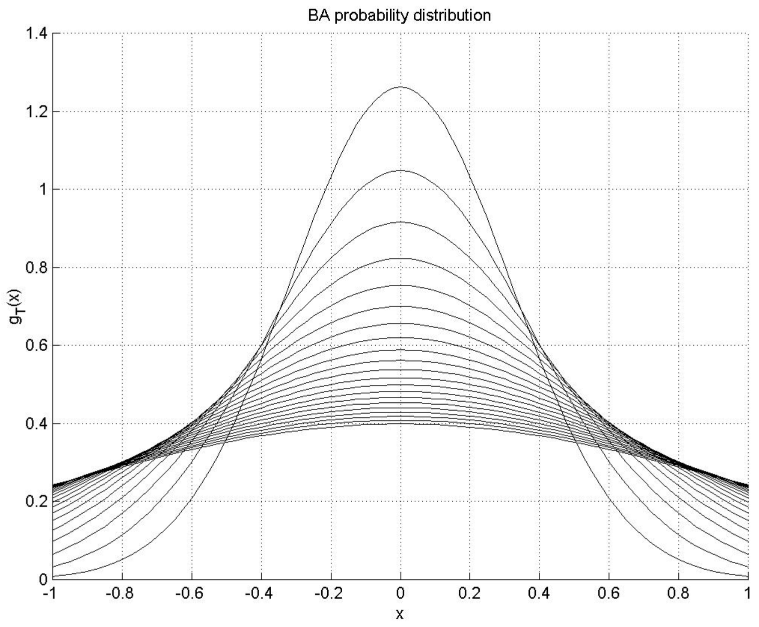


Fig. 2.3 2-D view of the BA probability function $g_T(x)$

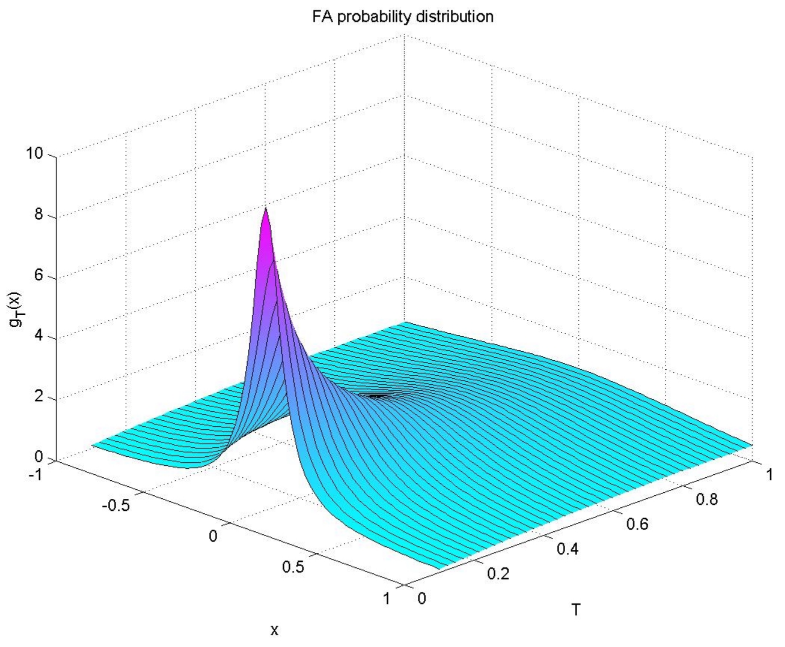


Fig. 2.4 3-D view of the FA probability function $g_T(x)$

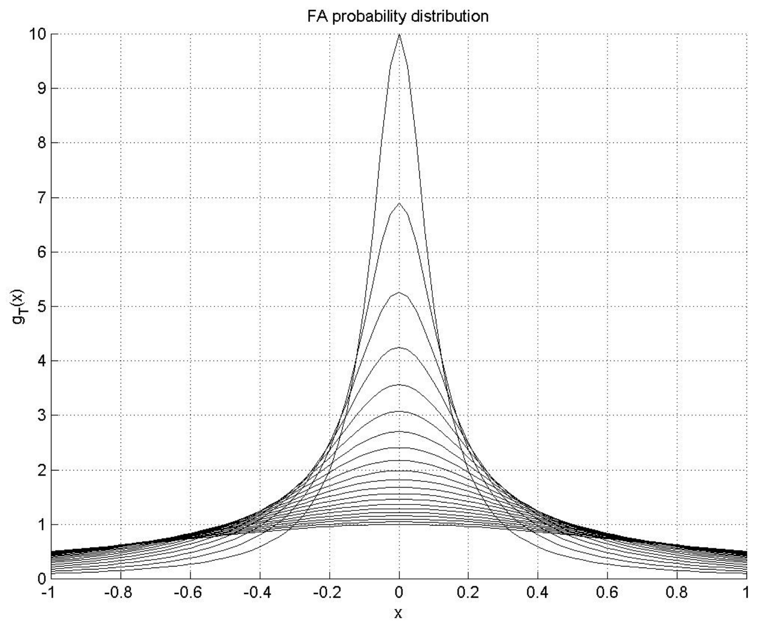


Fig. 2.5 2-D view of the FA probability function $g_T(x)$

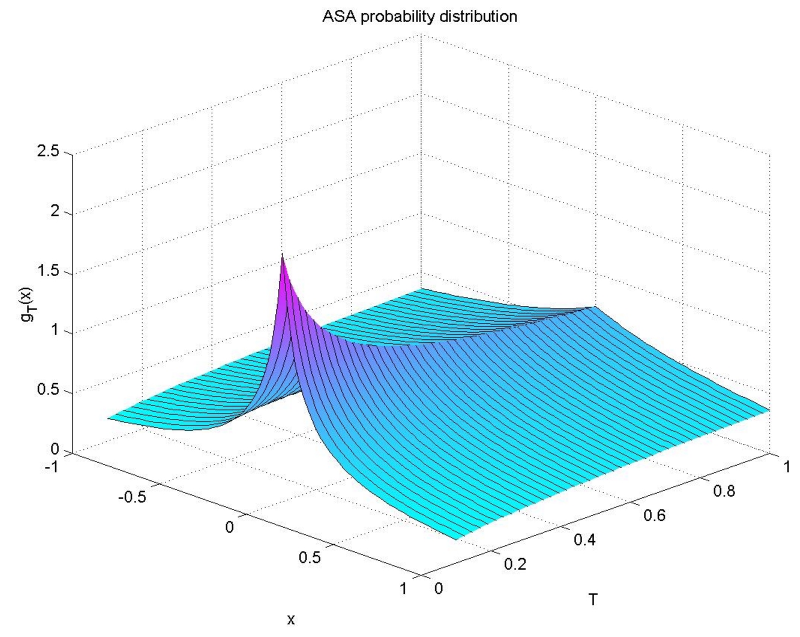


Fig. 2.6 3-D view of the ASA probability function $g_T(x)$

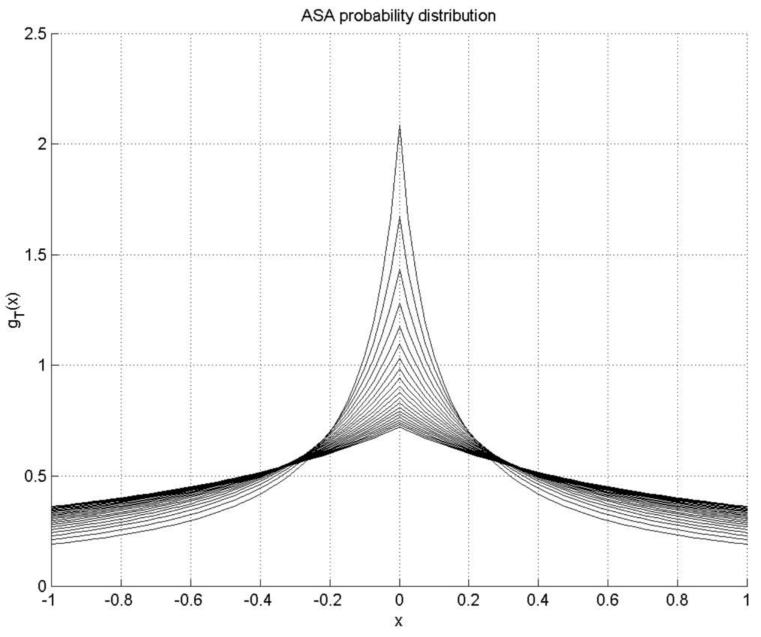


Fig. 2.7 2-D view of the ASA probability function $g_T(x)$

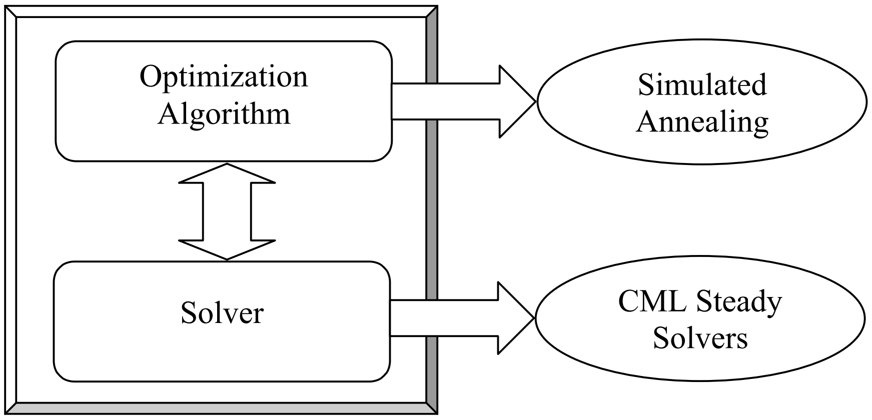


Fig. 2.8 Structure of the optimization program

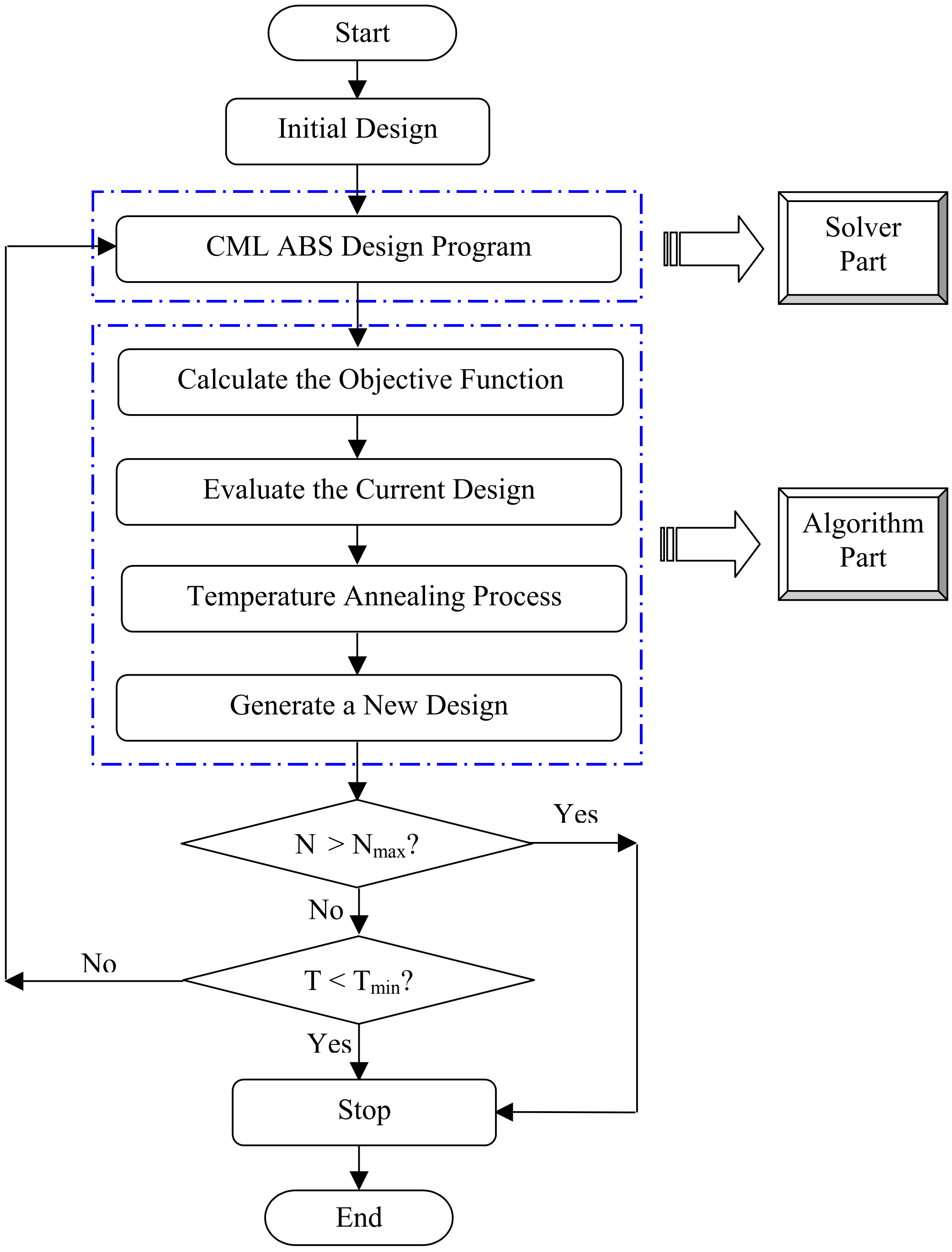


Fig. 2.9 Flow chart of the CML optimization program

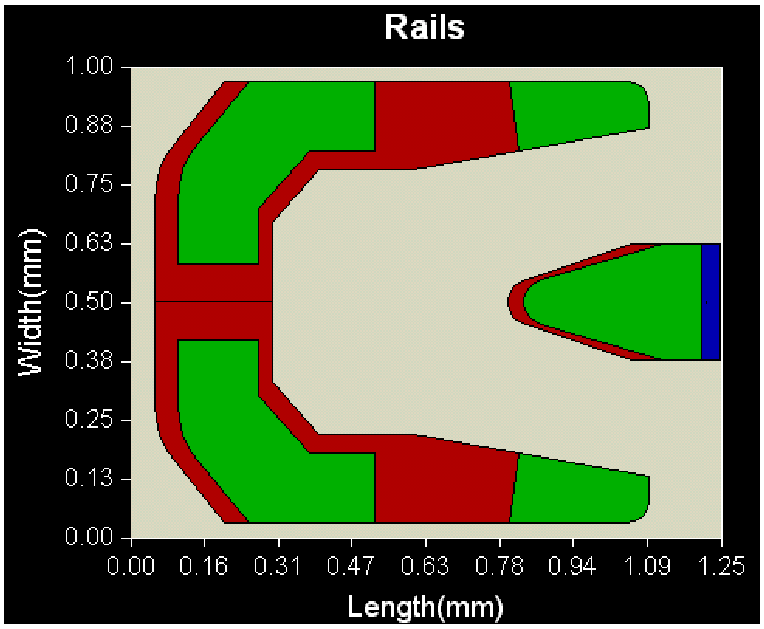


Fig. 2.10 Rail shape of the initial ABS design

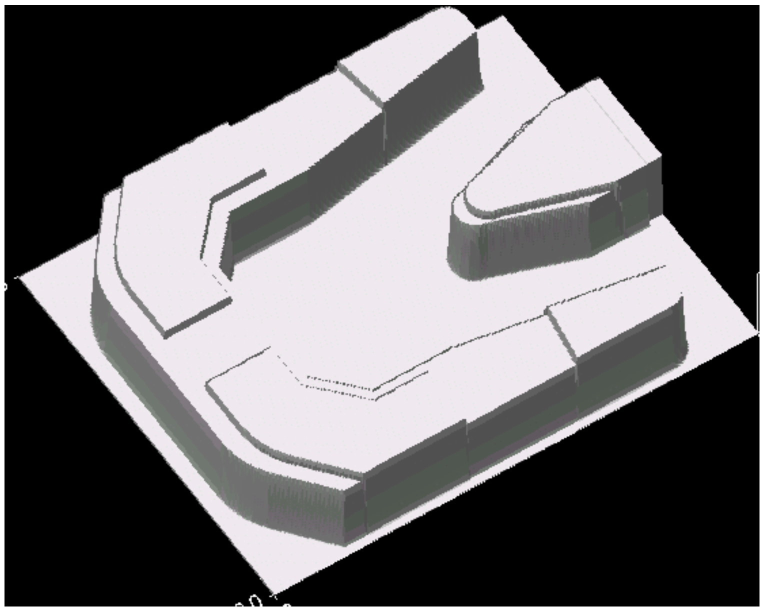


Fig. 2.11 3-D rail shape of the initial ABS design

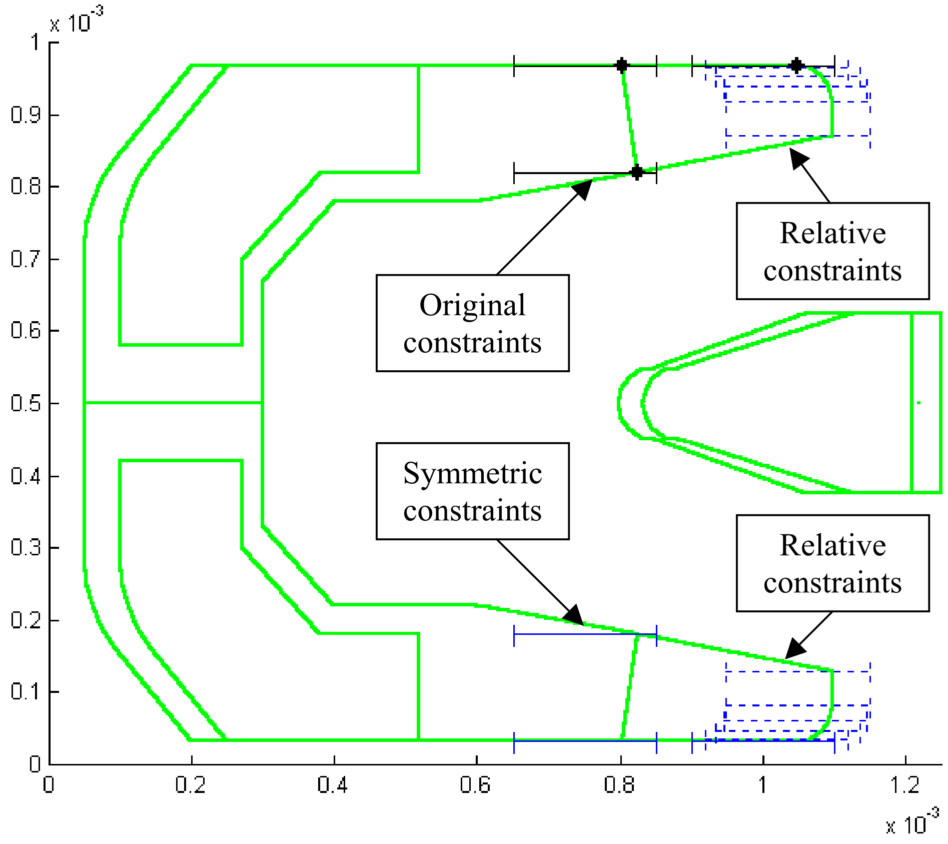


Fig. 2.12 Constraints defined on the initial design

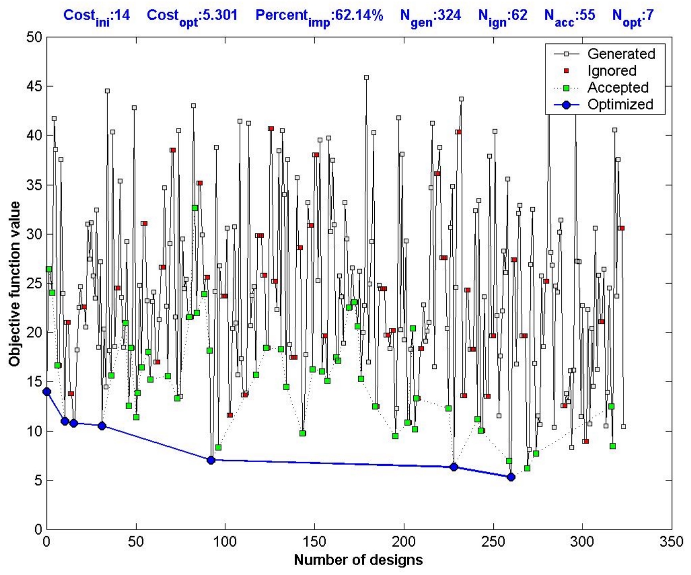


Fig. 2.13 Variation of the objective function value for BA

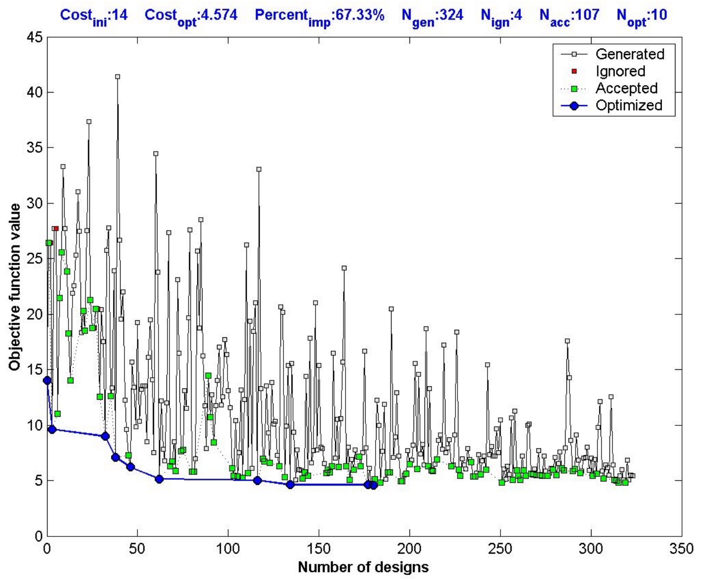


Fig. 2.14 Variation of the objective function value for FA

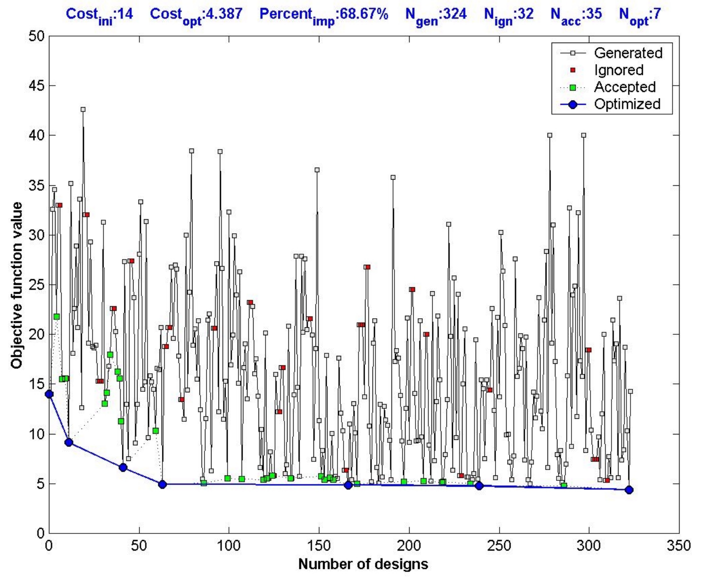


Fig. 2.15 Variation of the objective function value for ASA

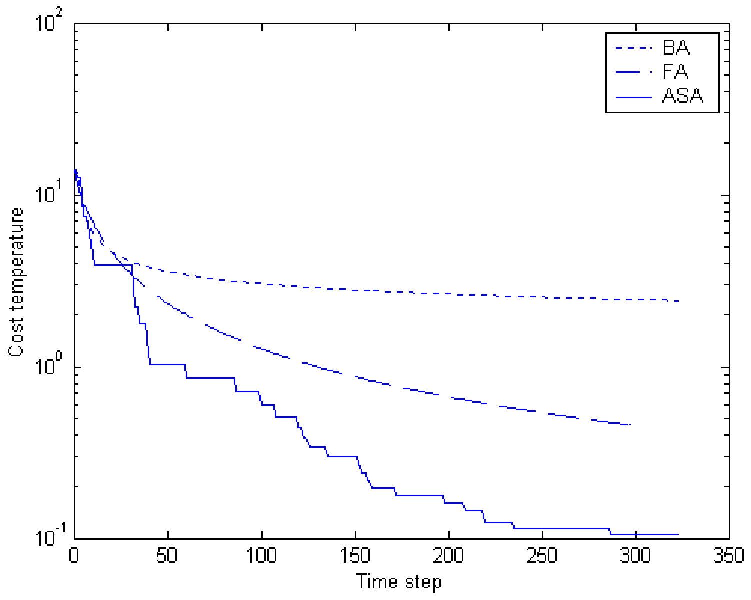


Fig. 2.16 Cost temperature variations for BA, FA and ASA

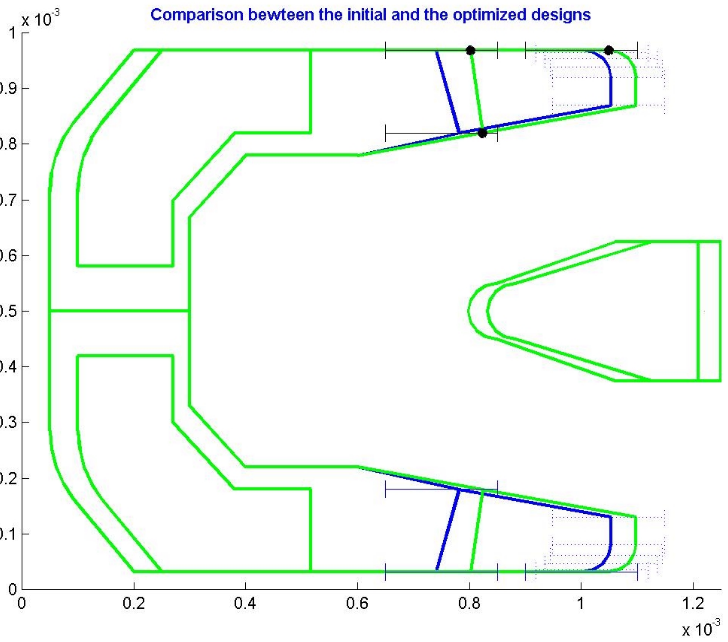


Fig. 2.17 Optimization results by using BA

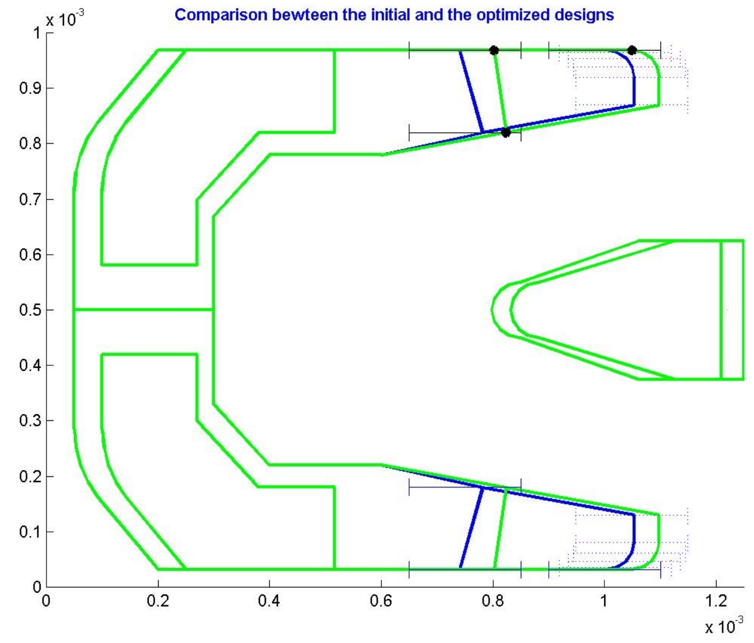


Fig. 2.18 Optimization results by using FA

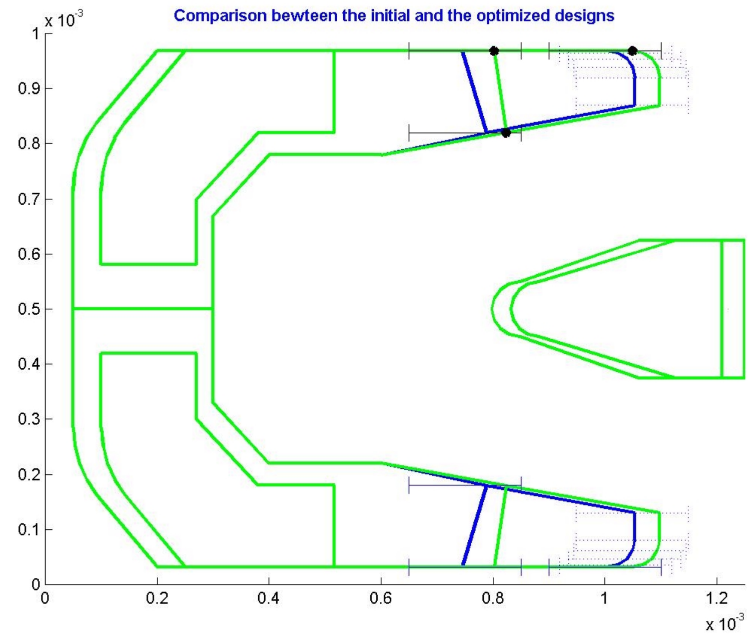


Fig. 2.19 Optimization results by using ASA

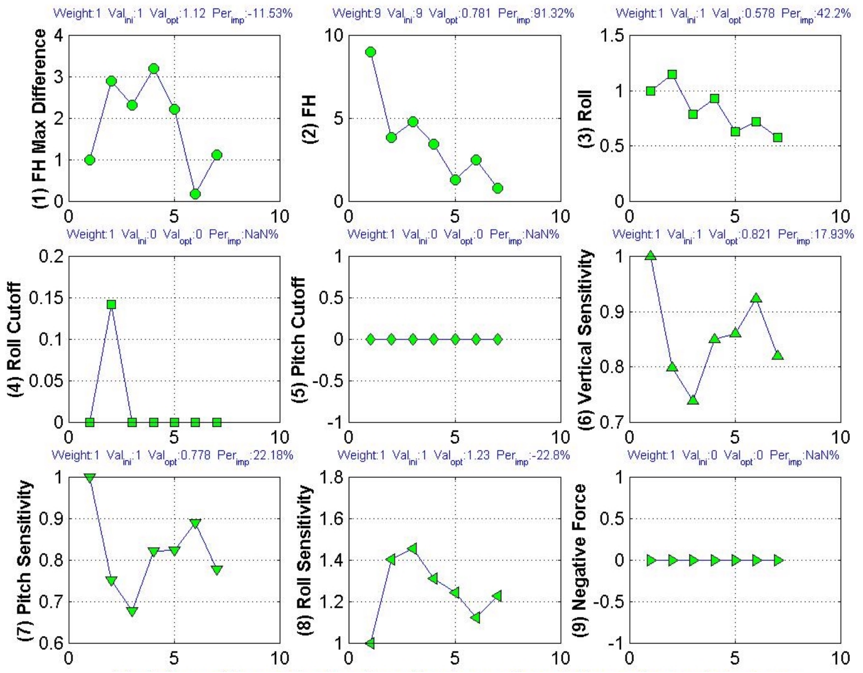


Fig. 2.20 Variation of the objective function terms by using BA

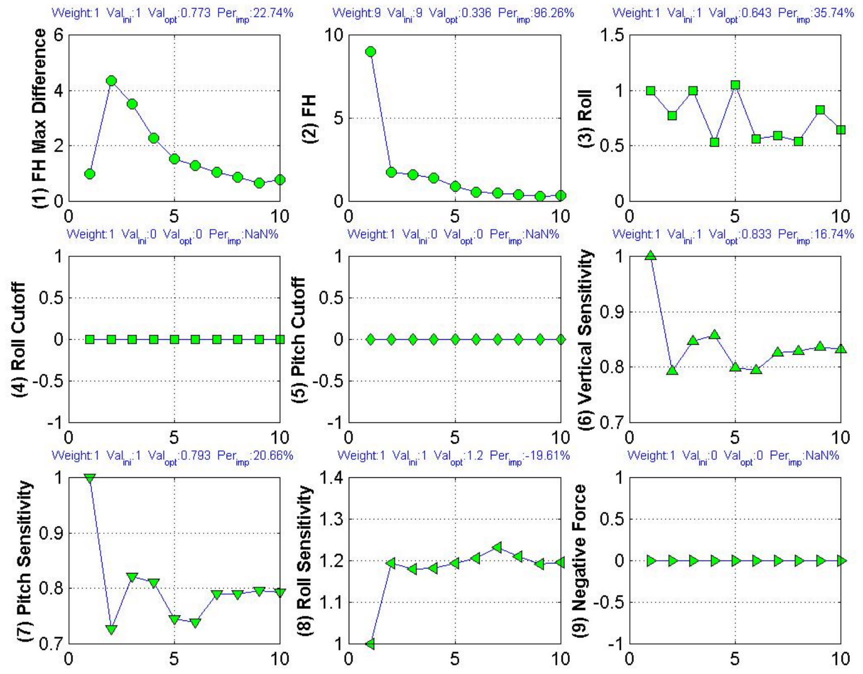


Fig. 2.21 Variation of the objective function terms by using FA

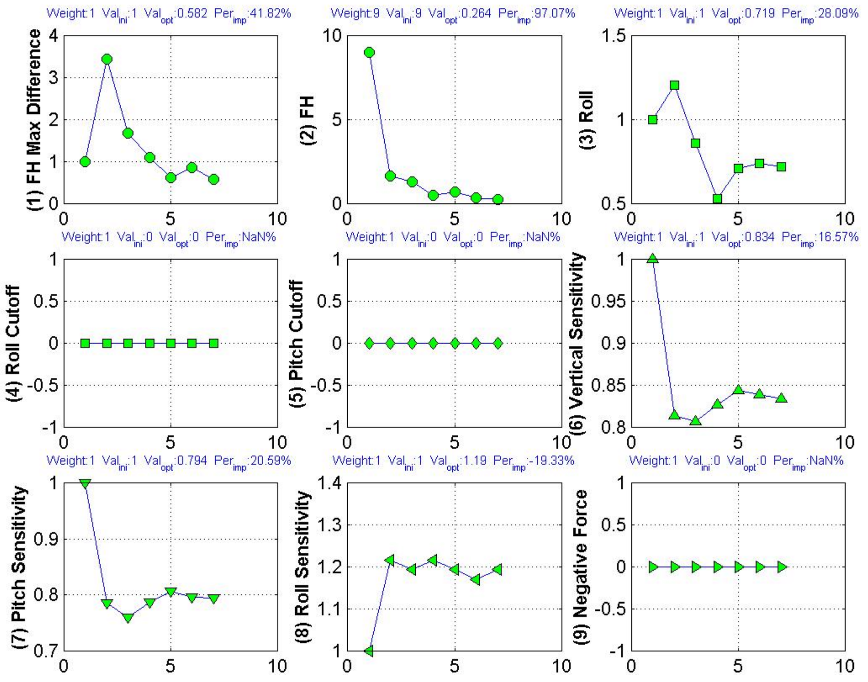
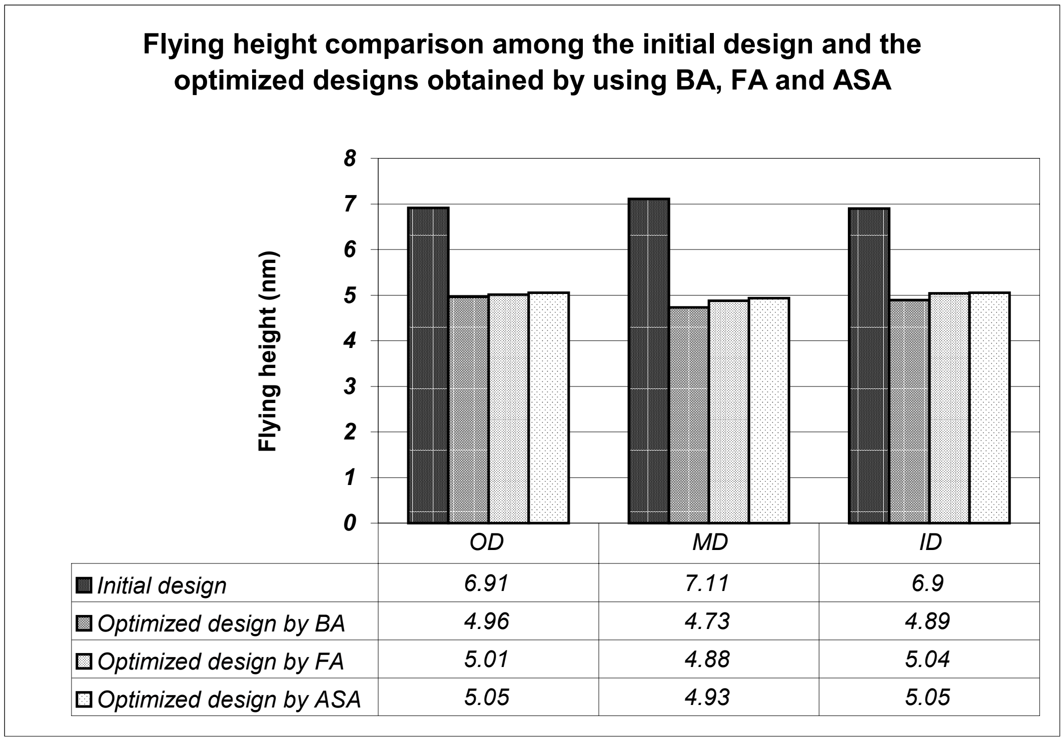
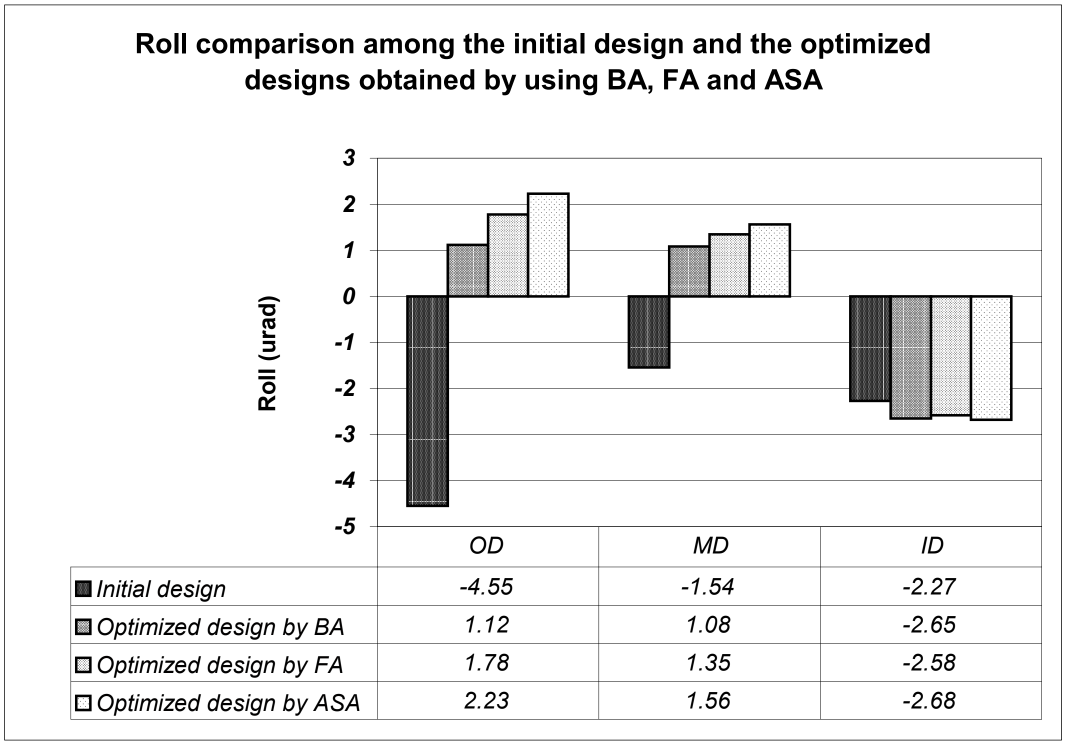


Fig. 2.22 Variation of the objective function terms by using ASA



	OD	*MD*	*ID*
■ *Initial design*	*6.91*	*7.11*	*6.9*
▨ *Optimized design by BA*	*4.96*	*4.73*	*4.89*
▧ *Optimized design by FA*	*5.01*	*4.88*	*5.04*
□ *Optimized design by ASA*	*5.05*	*4.93*	*5.05*

Fig. 2.23 Flying height distribution from OD to ID for different designs



	OD	MD	ID
■ *Initial design*	*-4.55*	*-1.54*	*-2.27*
▨ *Optimized design by BA*	*1.12*	*1.08*	*-2.65*
▨ *Optimized design by FA*	*1.78*	*1.35*	*-2.58*
□ *Optimized design by ASA*	*2.23*	*1.56*	*-2.68*

Fig. 2.24 Roll distribution from OD to ID for different designs

Chapter 3

DIRECT ALGORITHM

3.1 INTRODUCTION

The optimization algorithms used in the previous chapter were of the Simulated Annealing family, including the Standard Boltzmann Annealing (BA), Fast Cauchy Annealing (FA) and the Adaptive Simulated Annealing (ASA). 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 wide range of objective function types. Also, they are usually easily implemented (at least for the Simulated Annealing algorithm). But the disadvantage is that they usually require a long running time. Deterministic algorithms can handle definite objective functions very well. Because they use a specific searching strategy and their searching directions are strongly oriented, it is expected that they should require fewer sample points to find the global minimum point.

The DIRECT algorithm is a global deterministic algorithm developed by Jones et al. in 1993. The DIRECT algorithm has a very fast convergence rate, thus it should be able to find the global minimum very quickly compared with other algorithms (Jones et al, 1993; Gablonsky, 1998). 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.

3.2 NUMERICAL METHOD

3.2.1 Introduction to DIRECT

The **DIRECT** algorithm 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 that does not require knowledge of the Lipschitz constant. DIRECT is designed to solve problems subjected to bounded constraints.

3.2.2 One dimensional Lipschitzian optimization

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

$$| f(\boldsymbol{x}) - f(\boldsymbol{x'}) | \leq \boldsymbol{K} \, | \boldsymbol{x} - \boldsymbol{x'} | \quad \text{for all } \boldsymbol{x}, \boldsymbol{x'} \in [\boldsymbol{u},\boldsymbol{v}] \tag{3.1}$$

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:

$$f(x) \geq f(u) - K(x - u) \tag{3.2}$$

$$f(x) \geq f(v) + K(x - v) \tag{3.3}$$

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

$$g(x) = f(u) - K(x - u) \quad \text{for} \quad x \leq X(u, v, f, K) \tag{3.4}$$

$$g(x) = f(v) + K(x - v) \quad \text{for} \quad x \geq X(u, v, f, K)\,, \tag{3.5}$$

where

$$X(u, v, f, K) = [\, f(u) - f(v) \,] / (2K) + (u + v) / 2 \tag{3.6}$$

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

$$B(u, v, f, K) = [\, f(u) + f(v) \,] / 2 - K (v - u) / 2\,. \tag{3.7}$$

Figure 3.1 illustrates this.

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. 3.2, in which, it is clear that the piece-wise linear function *g(x)*, is approaching the 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 (Jones et al, 1993). 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.

3.2.3 One dimensional DIRECT algorithm

The DIRECT algorithm developed by Jones et al. 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:

$$f(x) \geq f(m) + K(x-m) \quad for \ \ x \leq m \tag{3.8}$$

$$f(x) \geq f(m) - K(x-m) \quad for \ \ x \geq m, \tag{3.9}$$

where, for now *K* is assumed known.

With these 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 $f(x)$.

$$h(x) = f(m) + K(x - m) \quad for \ \ x \leq m \tag{3.10}$$

$$h(x) = f(m) - K(x - m) \quad for \ \ x \geq m \tag{3.11}$$

This is shown in Fig. 3.3. 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 an interval is partitioned; and determining the potentially optimal intervals, i.e., the choice of intervals to be partitioned at each iteration step.

3.2.3.1 Dividing strategy

The DIRECT algorithm divides the interval into three equal subintervals. The dividing strategy is shown in Fig. 3.4.

3.2.3.2 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. 3.5. The x-axis represents the distances from the intervals' centers to their endpoints, and indicates the amount of unexplored territory in the intervals. 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. 3.5. 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 $[u_i, v_i]$.

$$D(u_i, v_i) = f(m_i) - K(v_i - u_i) / 2. \tag{3.12}$$

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 divided in the next step.

However as mentioned before, in many situations the Lipschitz constant K is unknown. So we need to estimate K based on the values at known data points. This corresponds 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 **(**Preparata and Shamos, 1985) (see **Appendix B** for details).

An example illustrating the process of choosing potentially optimal intervals is shown in Fig. 3.6. For all the data points with the same x-coordinate (i.e., the same 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 a potentially optimal interval is given as follows:

Definition 3.1 *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*

$$f(m_j) - \widetilde{K}(v_j - u_j)/2 \le f(m_i) - \widetilde{K}(v_i - u_i)/2 \quad \text{for any } i \tag{3.13}$$

$$f(m_j) - \widetilde{K}(v_j - u_j)/2 \le f_{min} - \varepsilon\,|f_{min}| \tag{3.14}$$

The inequality (3.13) represents a property of the data points on the convex hull. The inequality (3.14) 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. Here we set ε as 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.

3.2.4 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.

3.2.4.1 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 $s_i = \min \{ f(m - \delta e_i), f(m+\delta e_i) \}$, and partition in the order given by s_j, starting with the lowest s_j. 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 partitioned only along their longest sides. This partition strategy ensures that we obtain a reduction in the maximal side length of a hyper-rectangle.

Figure 3.7 presents a schematic illustration of the dividing strategy for 2-D case.

3.2.4.2 Potentially optimal hyper-rectangles

The definition of potentially optimal hyper-rectangles is very similar to Definition 3.1. 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 3.2 *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*

$$f(m_j) - \widetilde{K} d_j \leq f(m_i) - \widetilde{K} d_i \quad \text{for any } i \tag{3.15}$$

$$f(m_j) - \widetilde{K} d_j \leq f_{min} - \varepsilon |f_{min}| \tag{3.16}$$

3.2.4.3 2-D and 3-D examples

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

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

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

Figures 3.8a ~ 3.8f show the first 5 iterations for this 2-D case. 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. 3.8b we see that

$s_1 = \min\{0.144, 0.278\} \quad = 0.144$

$s_2 = \min\{0.0111, 0.411\} = 0.0111$

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

From Fig. 3.8c we see that the rectangles are only partitioned along their longest side.

For the 3-D case, consider the function:

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

Figures 3.9a ~ 3.9f show the first 5 iterations for this 3-D case. 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. 3.9a ~ 3.9f 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. 3.9a ~ 3.9f 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 of the boxes and the sample points are projected to the XY, YZ and ZX planes.

3.3 NUMERICAL EXPERIMENTS WITH THE DIRECT ALGORITHM

3.3.1 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, $x_i \in [0,1],\ \ i = 1,\ldots 10.$

The results for 2-D case are shown in Figs. 3.10 ~ 3.14.

Figure 3.10 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 3.11 shows the local zoom-in of Fig. 3.10. The global minimum point found by DIRECT at this stage is (0.4012346, 0.1995885), which is denoted by the circular dot in Fig. 3.12, 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 3.12 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. 3.13. It is seen that the sample points generated by the DIRECT algorithm become more and more clustered around the exact solution. Figure 3.14 shows the DIRECT algorithm's convergence property for this case with a very fast convergence rate.

Figures 3.15 and 3.16 show the results for the 3-D case.

The left picture in Fig. 3.15 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. 3.15 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. 3.16 confirms that the DIRECT algorithm has a very fast convergence rate.

Figures 3.17 and 3.18 show the results for the 5-D case. Figure 3.17 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 3.18 shows the convergence property of DIRECT for this 5-D case.

Similarly, we show the results for the 10-D case in Figs. 3.19 and 3.20.

Figure 3.19 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 3.20 shows the convergence property of DIRECT for this 10-D case.

3.3.2 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 3.21 shows the results after 30 iterations (81 function evaluations).

The uniform distribution of sample points in Fig. 3.21 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).

Figure 3.22 shows the contour lines of the Branin function. The five circular dots represent the five global minima. Figure 3.23 shows the optimization results after 32 iterations (1029 function evaluations). We combine Figs. 3.22 and 3.23 in Fig. 3.24. Figure 3.25 shows the local zoom-in of Fig. 3.24. From Fig. 3.25 we see that the sample points cluster around all five global minimal points. This illustrates how the DIRECT algorithm is capable of finding multiple global minima.

3.3.3 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 make the optimization difficult since it's easy for the process to get 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:

$F(x_1, x_2) = 100\,(x_1 - x_2^2)^2 + (1 - x_2)^2,$ 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.

Figure 3.26 shows the contour lines of the Rosenbrock function. The round dot represents the global minimum point. Figure 3.27 shows the 3-D surface shape of the Rosenbrock function.

Figure 3.28 shows the results of optimization after 81 iterations (2011 function evaluations). The global minimum point found by DIRECT at the final stage is (0.7440967, 0.7441193), which is represented by the circular dot, and the value at the minimum point is 1.024812E-08. Figure 3.28 clearly shows that the sample points are clustered in the valley. Figure 3.29 shows the local zoom-in of Fig. 3.28 around the global minimum point. The convergence property of DIRECT for this function is shown in Fig. 3.30.

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. 3.31 and 3.32, respectively. The nine solid dots in Fig. 3.32 denote the nine global minimum points.

Figure 3.33 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. 3.33 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 of the global minima of the Shubert function.

3.3.4 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 Jones et al (1993), 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.

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 apply it to the problem of slider ABS optimization.

3.4 IMPLEMENTATION OF DIRECT TO ABS OPTIMIZATION

3.4.1 Structure of the optimization program

The structure of the optimization program is shown in Fig. 3.34. It is similar to Fig. 2.8 in Section 2.3.1 except that the algorithm now has been changed from Simulated Annealing to DIRECT.

3.4.2 Flow chart of the optimization program

The flow chart of the optimization program is shown in Fig. 3.35, 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.

3.5 AIR BEARING DESIGN OPTIMIZATION PROBLEM

We choose the same 7nm FH Pico slider defined in Section 2.4 as the prototype slider. Its rail shape and the 3-dimensional rail geometry are shown in Figs. 2.10 and 2.11, respectively. Our optimization goals are also the same, i.e., to lower its flying heights to the target flying height, i.e. 5nm, and at the same time maintain a uniform roll profile at the three different radial positions OD, MD and ID. And we define the same objective function as in Section 2.4.

We defined slightly different constraint points in this case. The constraints are shown in Fig. 3.36.

3.6 OPTIMIZATION RESULTS

Using the initial design, constraints, and objective function, we carried out the optimization using the DIRECT algorithm. Figure 3.37 shows the variation of the objective function values during the optimization process. In this figure, Cost_{ini} is the initial objective function value, and Cost_{opt} is the objective function value for the final optimized design. The $\text{Percent}_{\text{imp}}$ 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.

The comparison between the initial and optimized designs is shown in Fig. 3.38, in which the gray lines show the rail shape of the initial design and the dark lines show the rail shape of the optimized design. Figure 3.39 shows the variation of the objective function terms for all of the best-so-far designs generated during the optimization process.

From Fig. 3.39 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 cutoff 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. 3.40.

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. The results show that the DIRECT algorithm has a much higher convergence rate than the ASA algorithm. And the DIRECT algorithm obtained a better-optimized design than ASA algorithm. Details about the comparisons between these two algorithms will be given in Chapter 6.

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

Jones et al (1993) 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.

By comparing the results obtained by using DIRECT and ASA, we found DIRECT clearly outperforms ASA. Details will be discussed in Chapter 6.

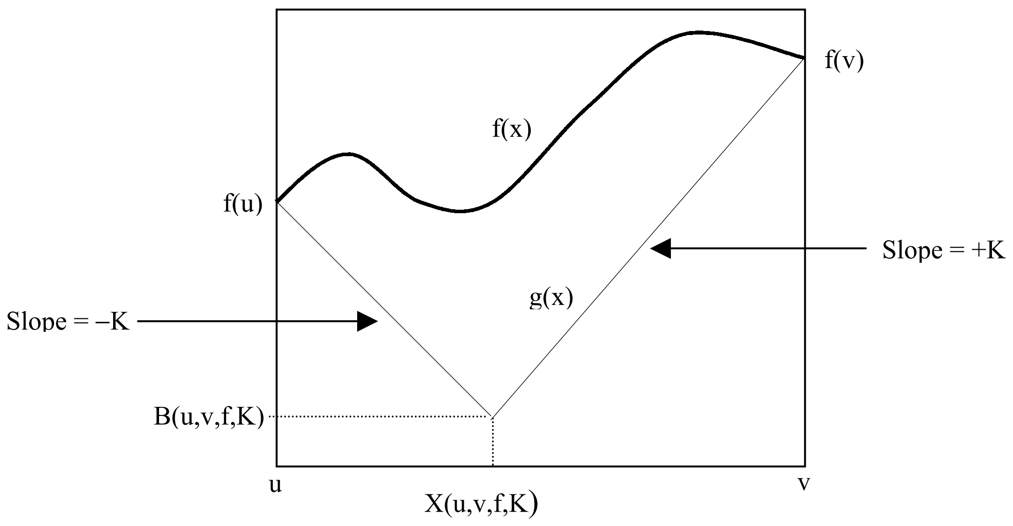


Fig. 3.1 Examples of f(x) and g(x)

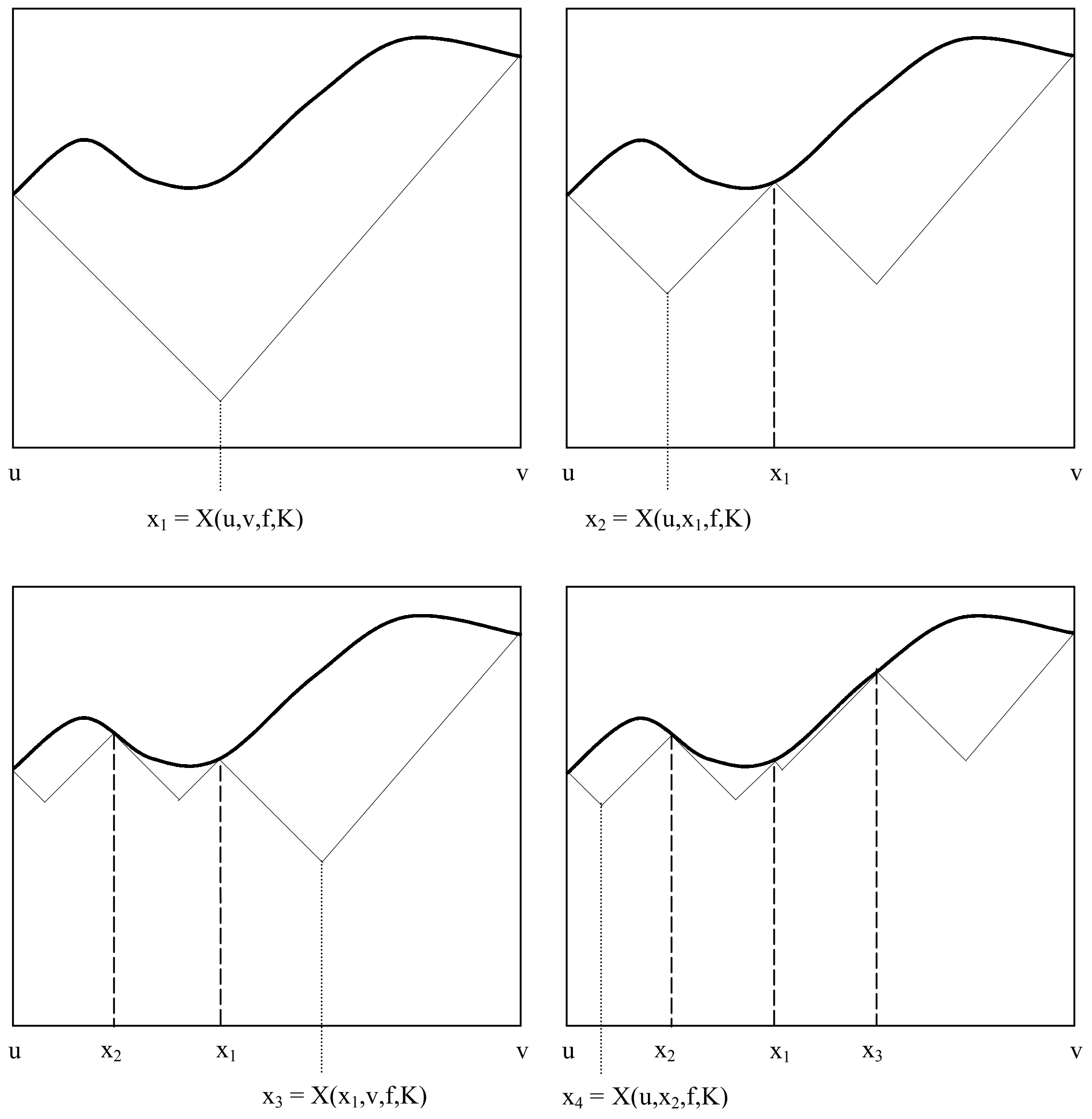


Fig. 3.2 Process of the Shubert algorithm

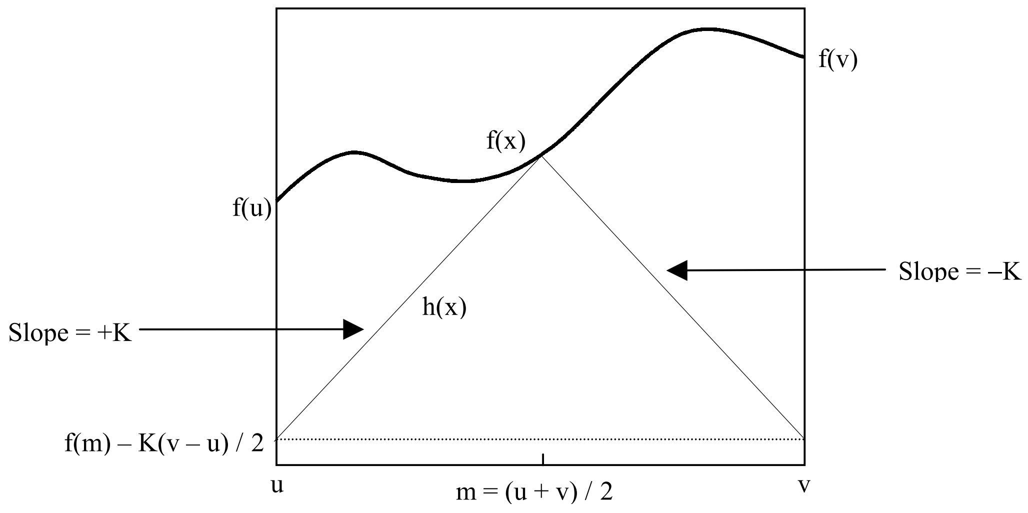


Fig. 3.3 Example of f(x) and h(x)

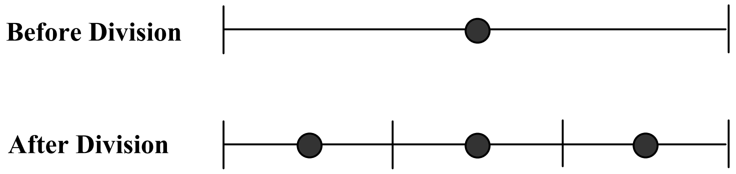


Fig. 3.4 Dividing strategy of DIRECT

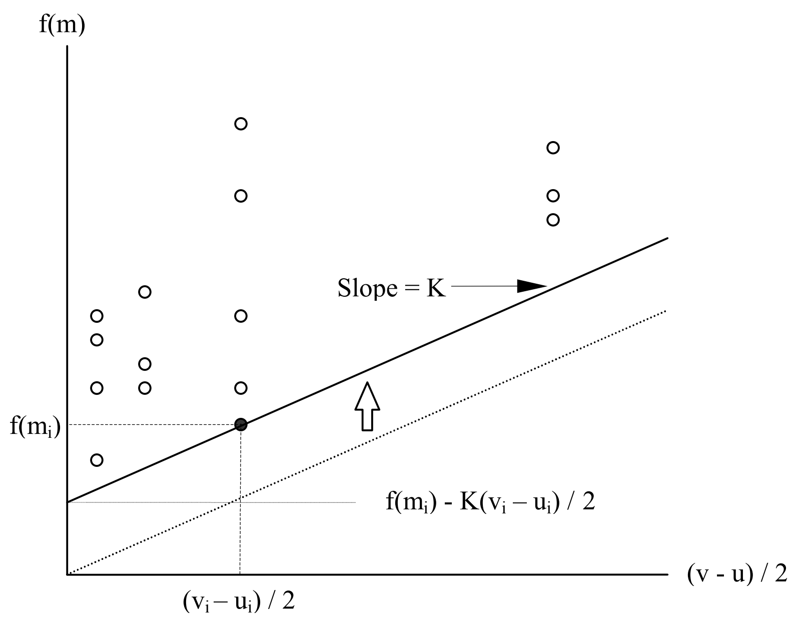


Fig. 3.5 Interval selection

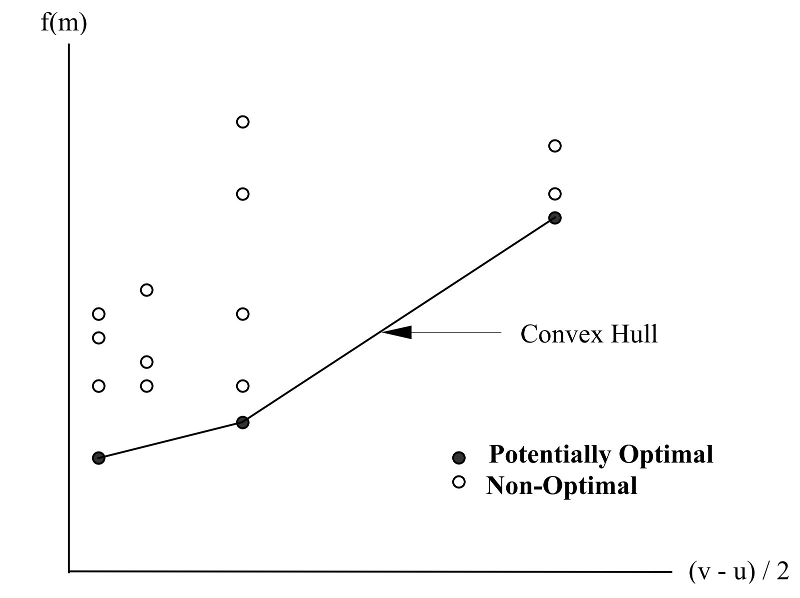


Fig. 3.6 Selection of potentially optimal intervals

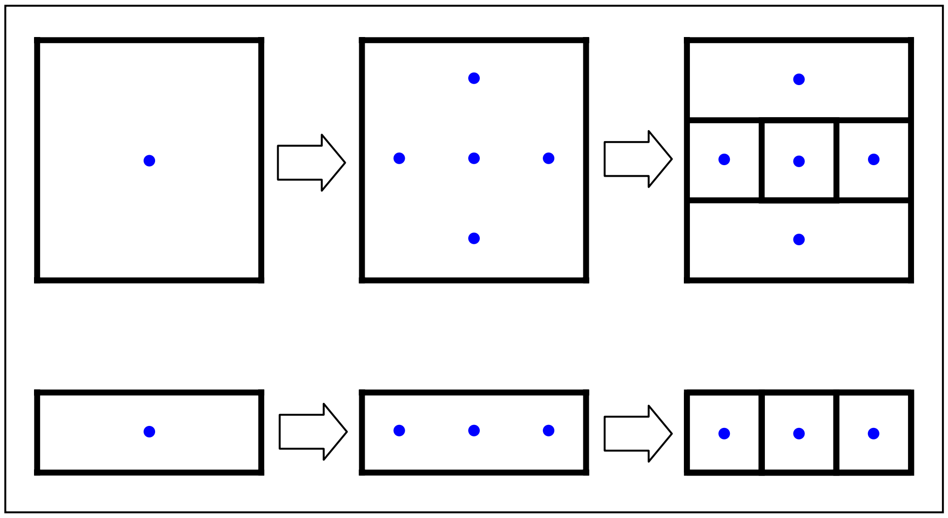
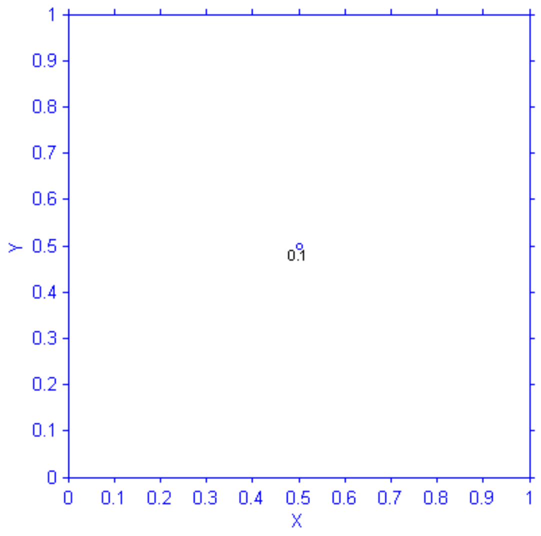


Fig. 3.7 Dividing strategy for 2-D case



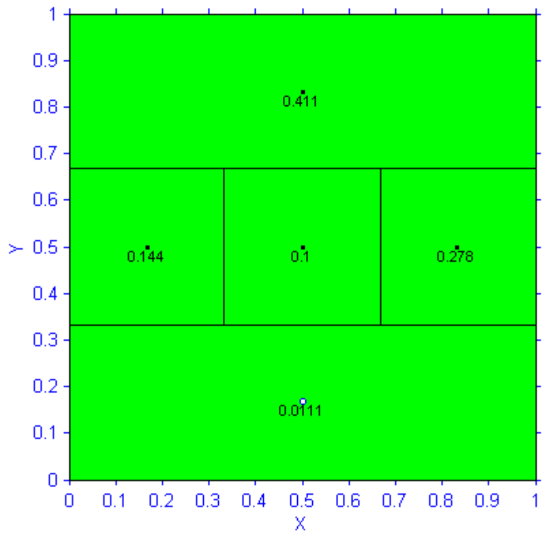


Fig. 3.8a Initial state

Fig. 3.8b Iteration 1

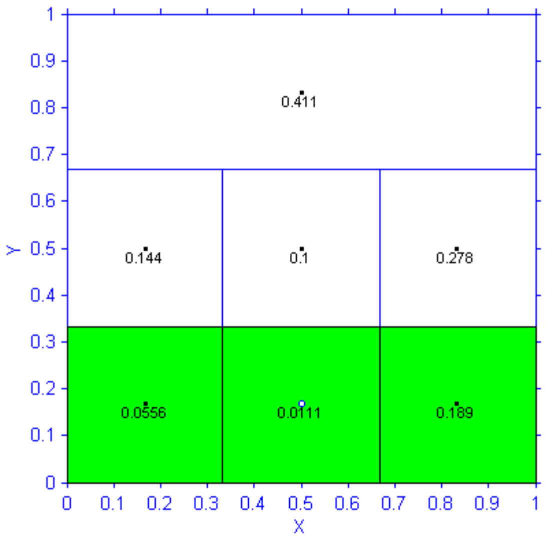


Fig. 3.8c Iteration 2

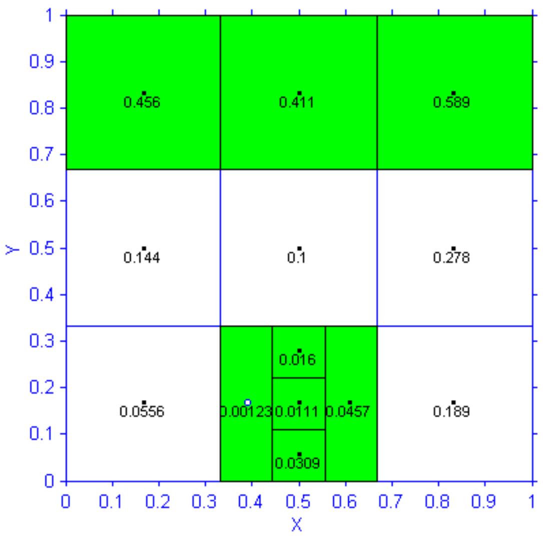


Fig. 3.8d Iteration 3

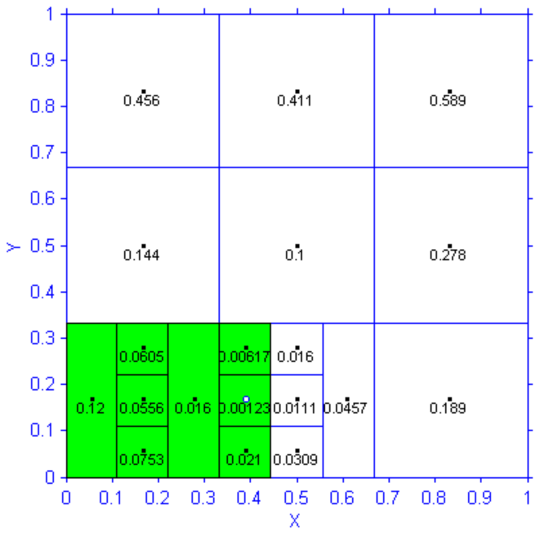


Fig. 3.8e Iteration 4

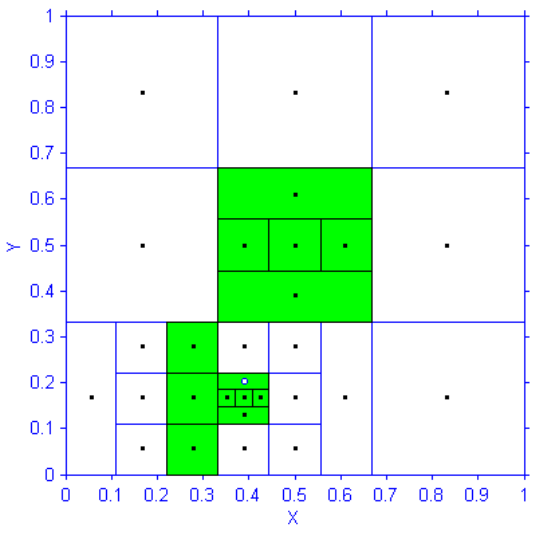


Fig. 3.8f Iteration 5

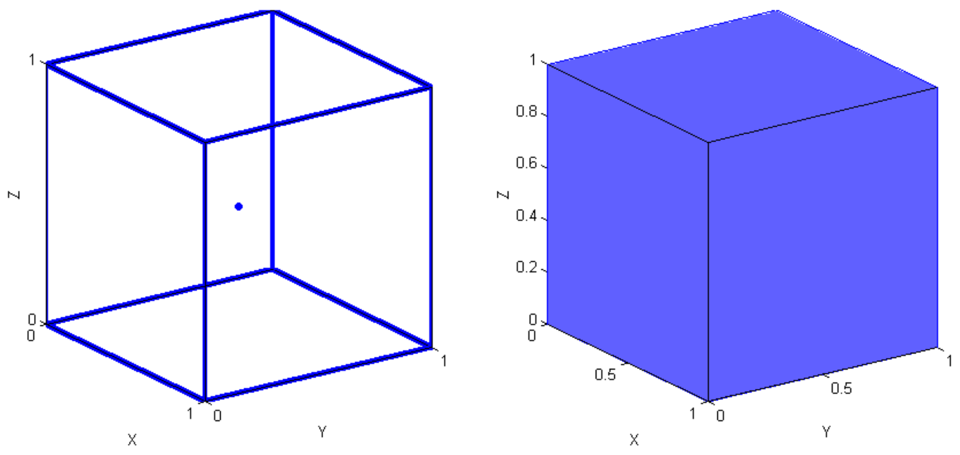


Fig. 3.9a Initial state

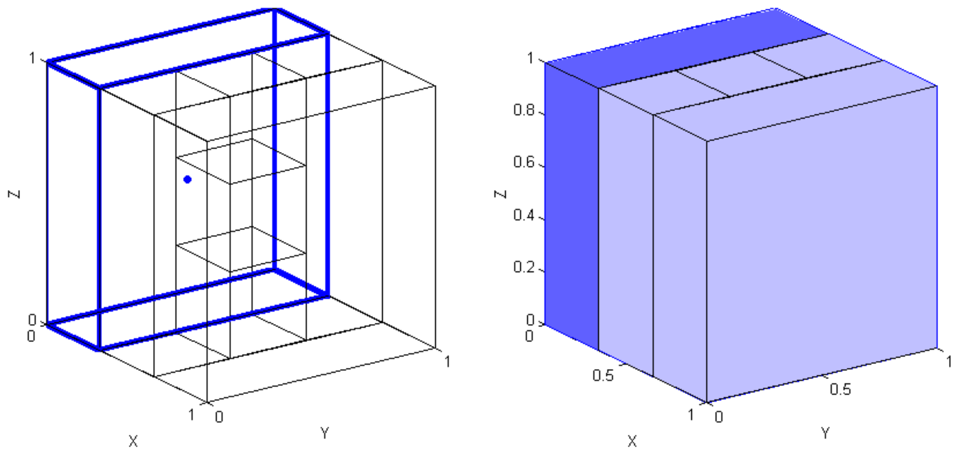


Fig. 3.9b Iteration 1

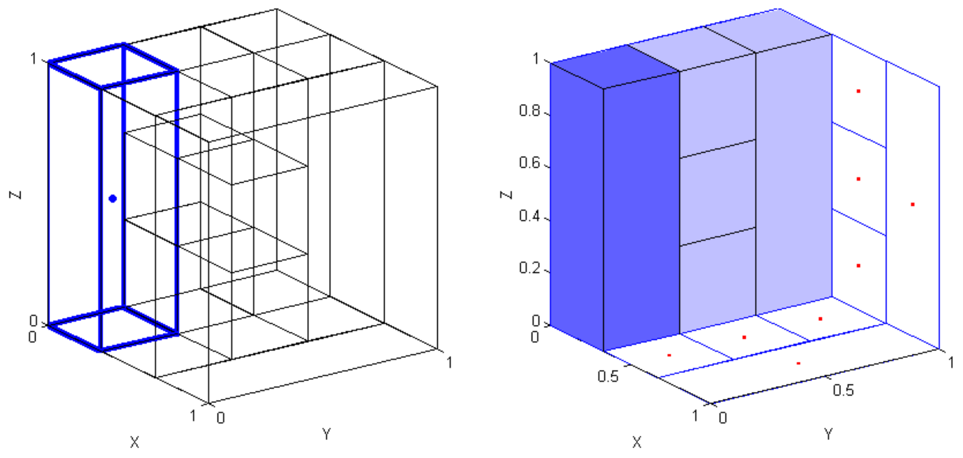


Fig. 3.9c Iteration 2

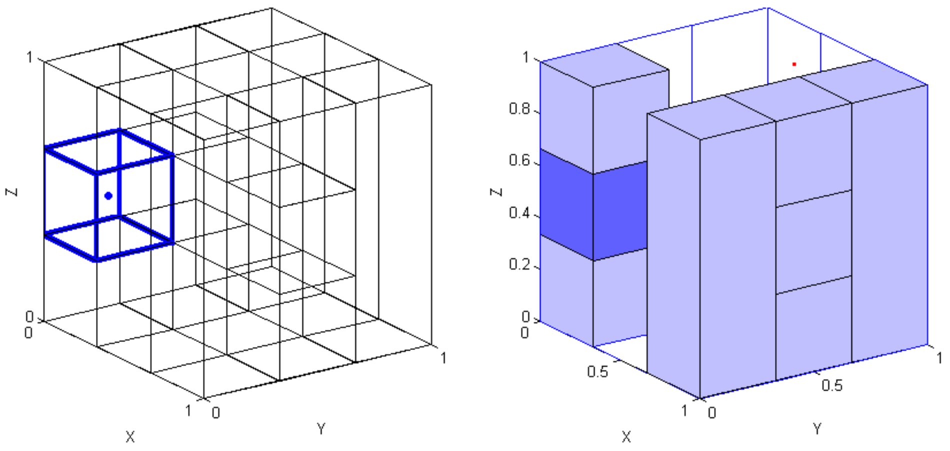


Fig. 3.9d Iteration 3

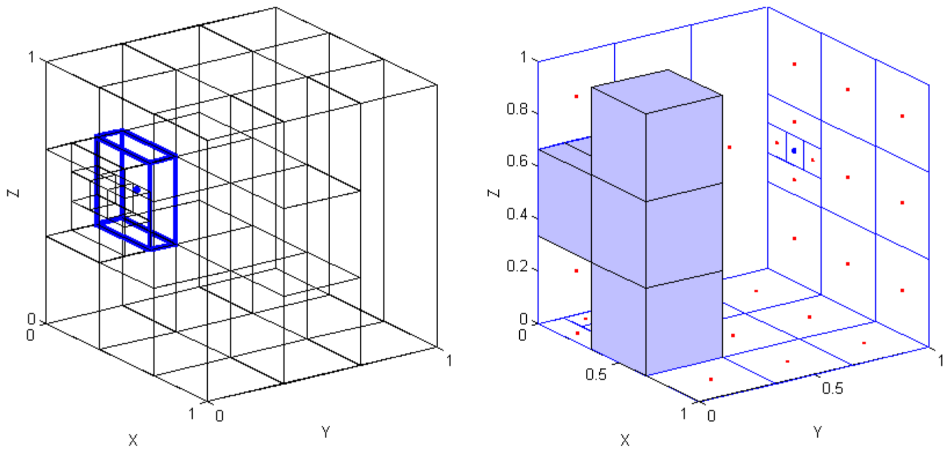


Fig. 3.9e Iteration 4

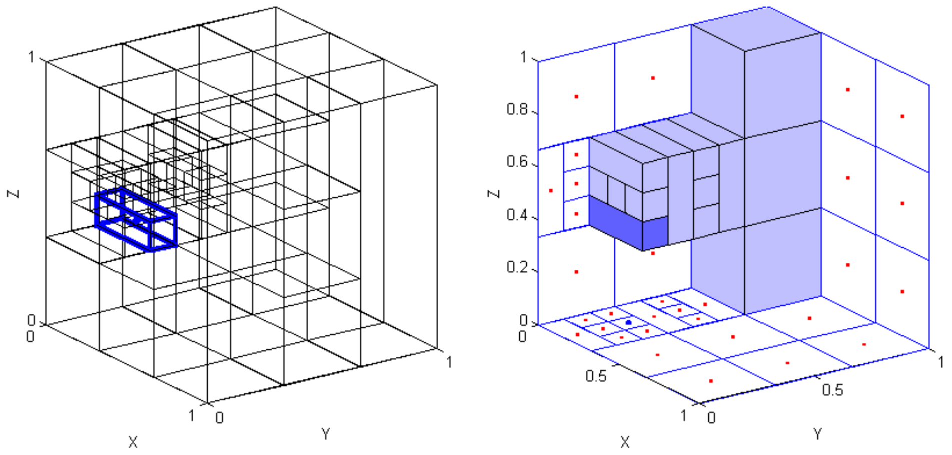


Fig. 3.9f Iteration 5

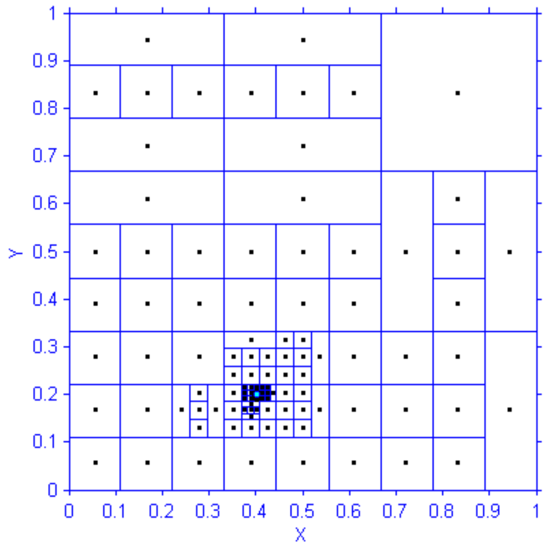


Fig. 3.10 Results for the 2-D case

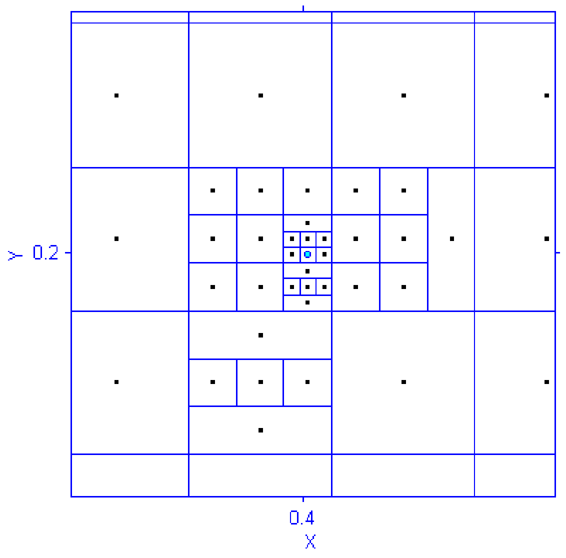


Fig. 3.11 Local zoom-in near minimum

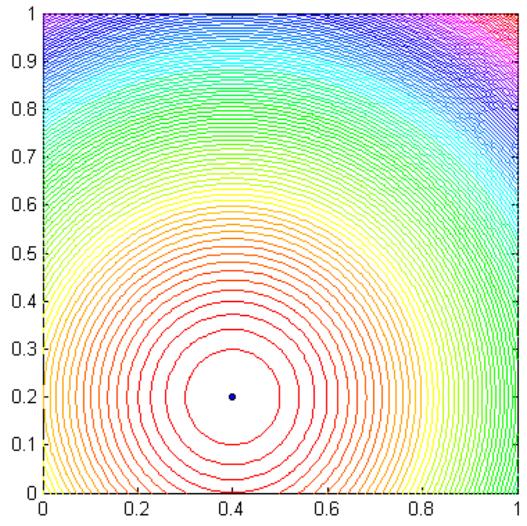


Fig. 3.12 Contour lines of the 2-D testing function

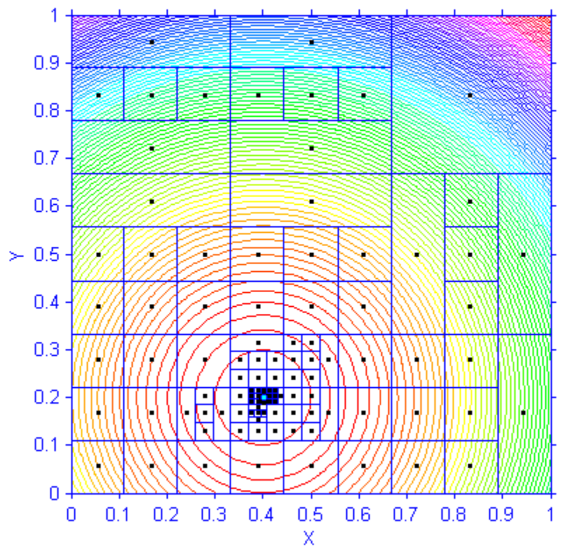


Fig. 3.13 Combination of contour lines and results

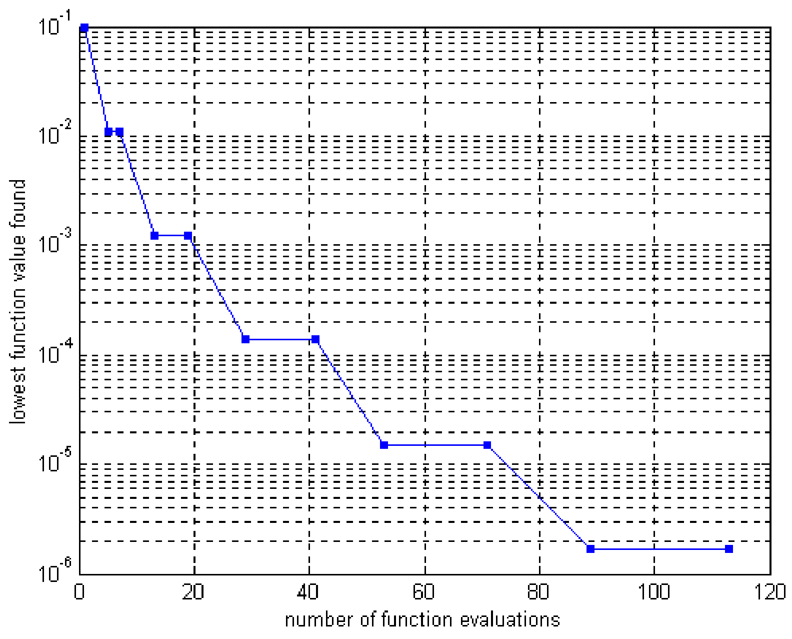


Fig. 3.14 Convergence property for the 2-D case

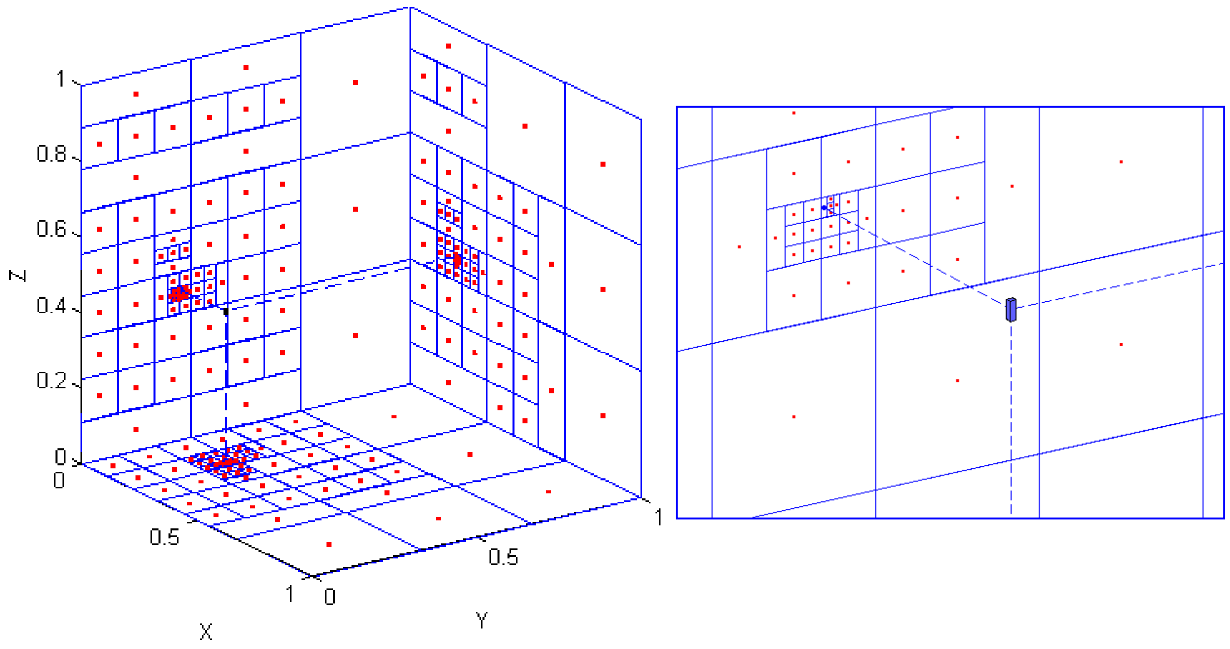


Fig. 3.15 Results of 3-D case

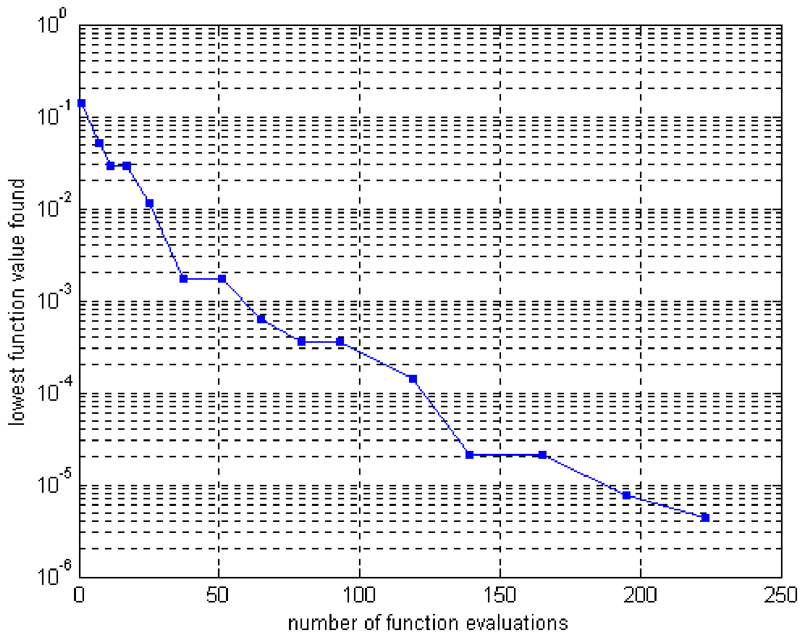


Fig. 3.16 Convergence property for 3-D case

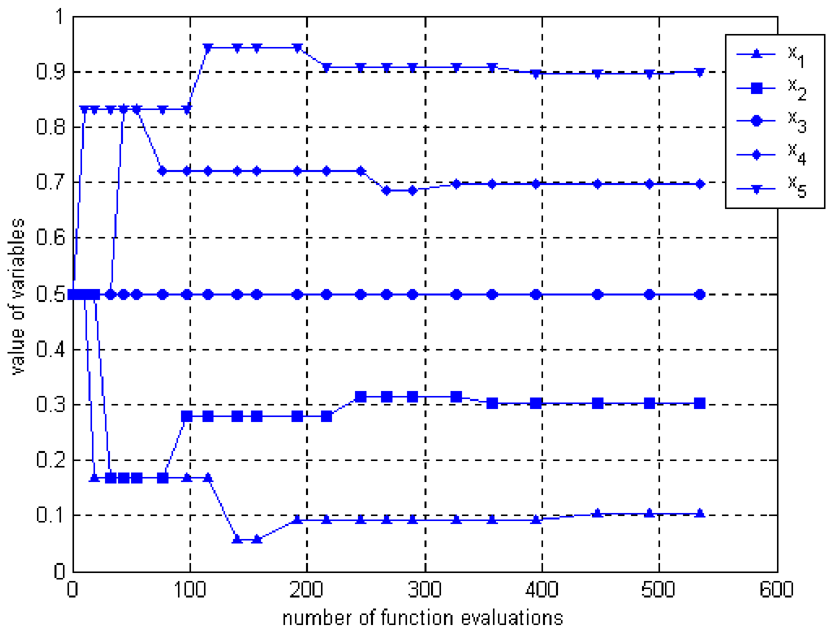


Fig. 3.17 Variations of variables for 5-D case

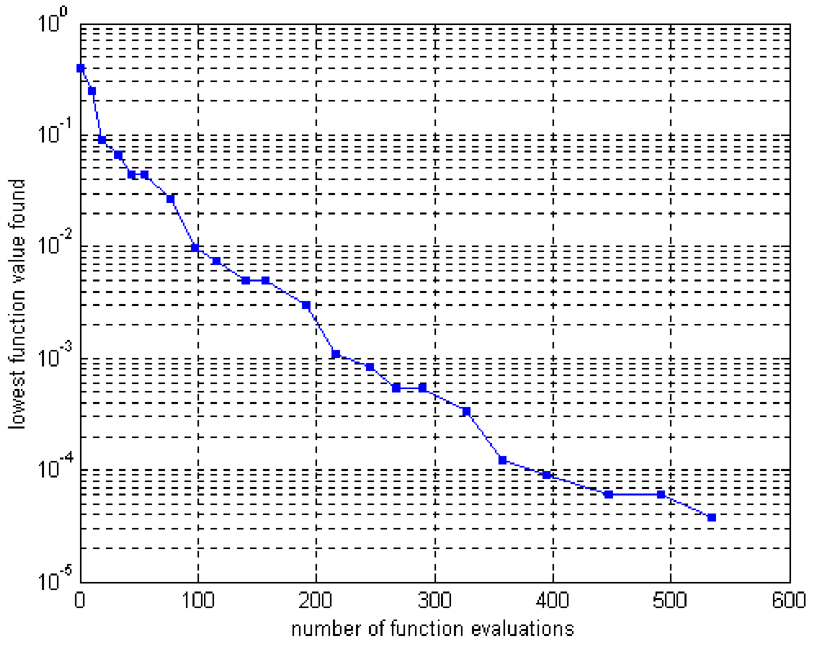


Fig. 3.18 Convergence property for 5-D case

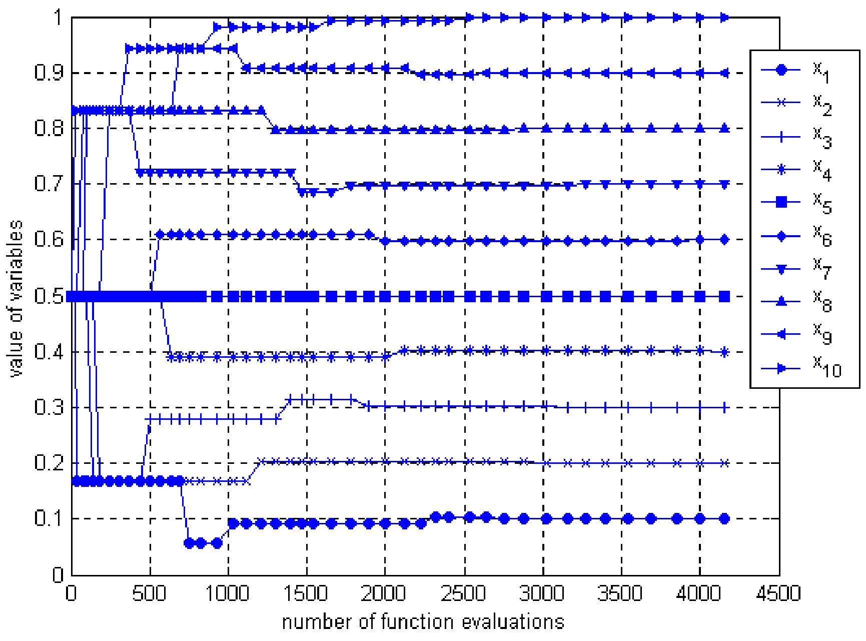


Fig. 3.19 Variations of variables for 10-D case

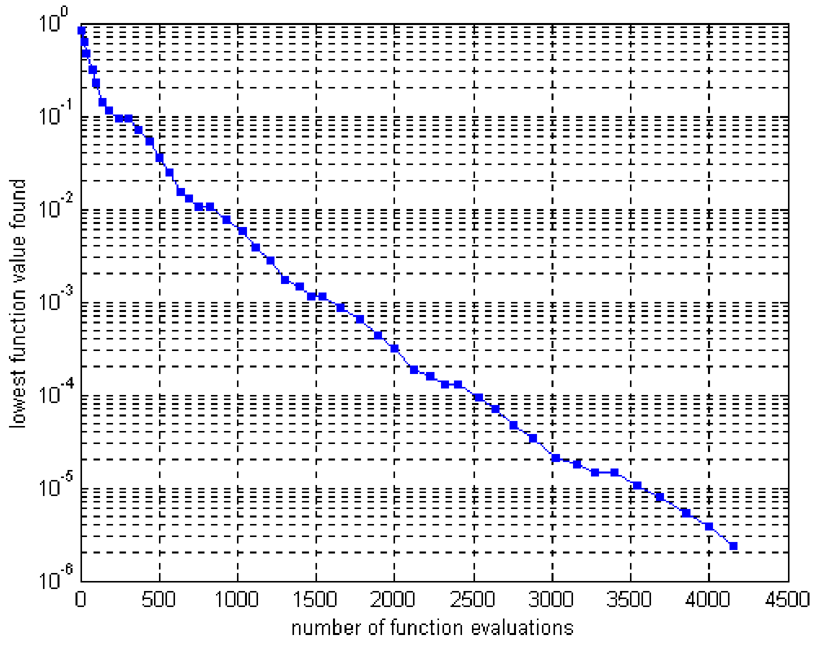


Fig. 3.20 Convergence property for 10-D case

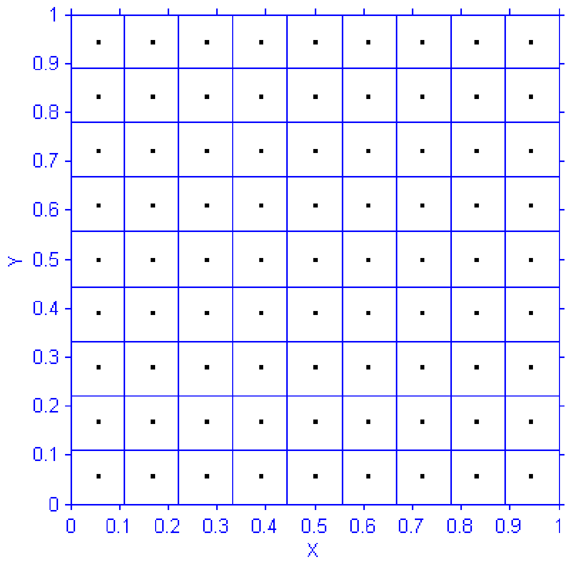


Fig. 3.21 Results for 2-D constant functions

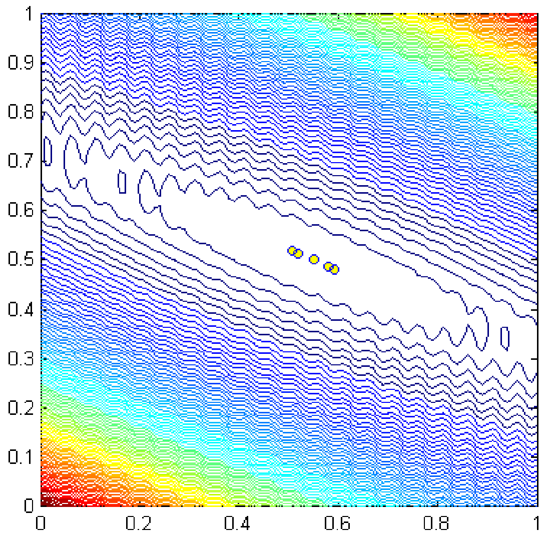


Fig. 3.22 Contour lines of Branin function

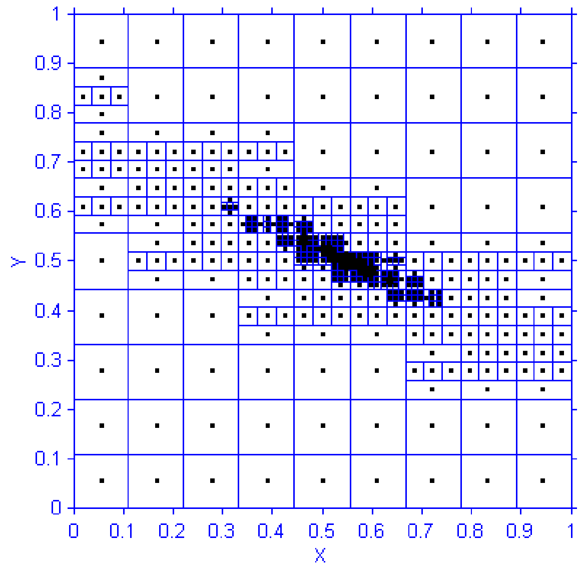


Fig. 3.23 Optimization results

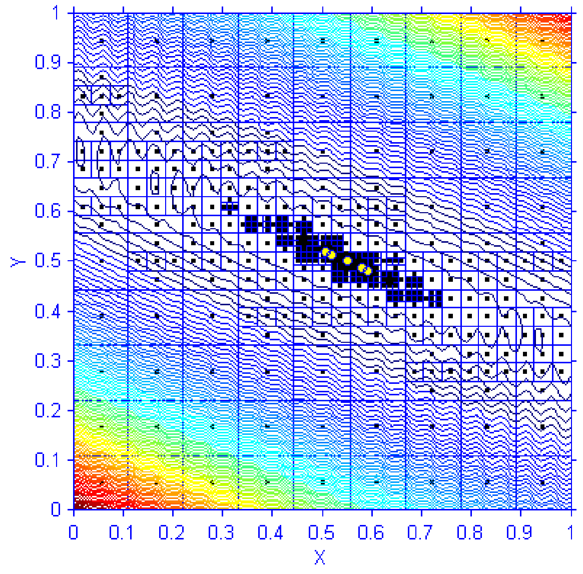


Fig. 3.24 Combination of contour lines and optimization results

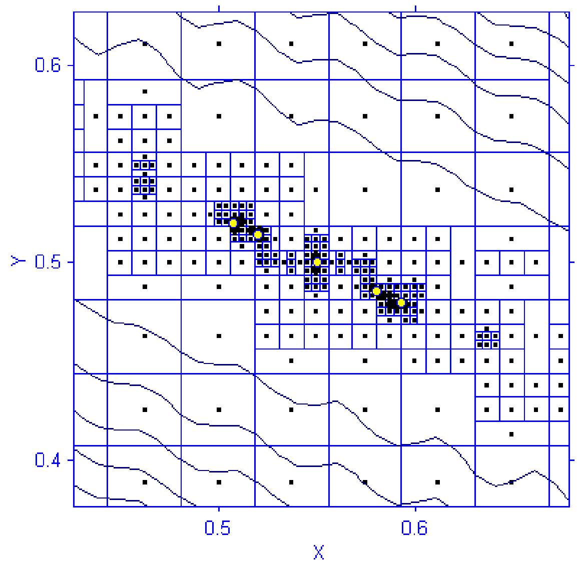


Fig. 3.25 Local zoom-in around the global minima points

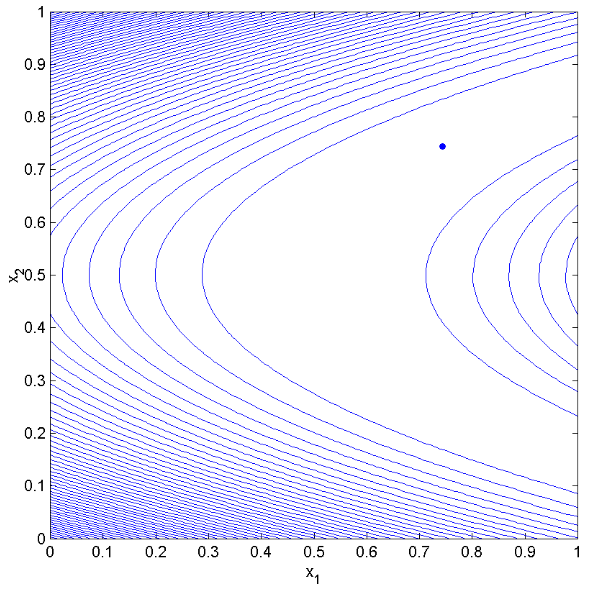


Fig. 3.26 Contour lines of Rosenbrock function

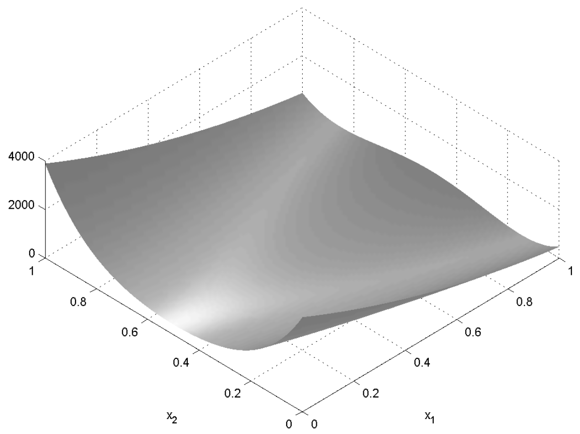


Fig. 3.27 3-D surface of function

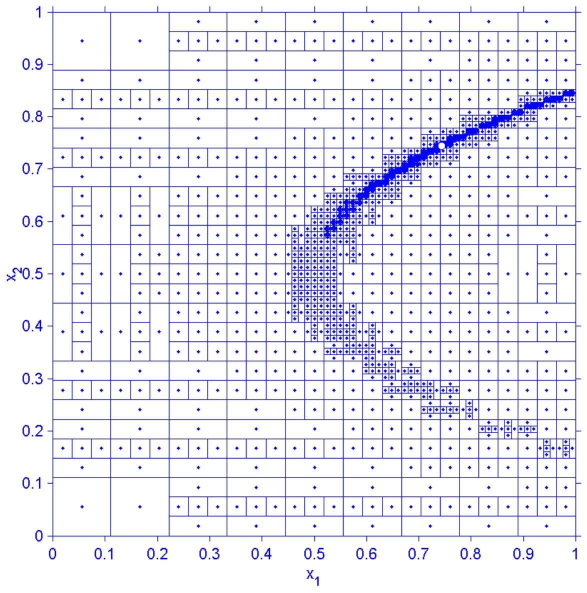


Fig. 3.28 Optimization results

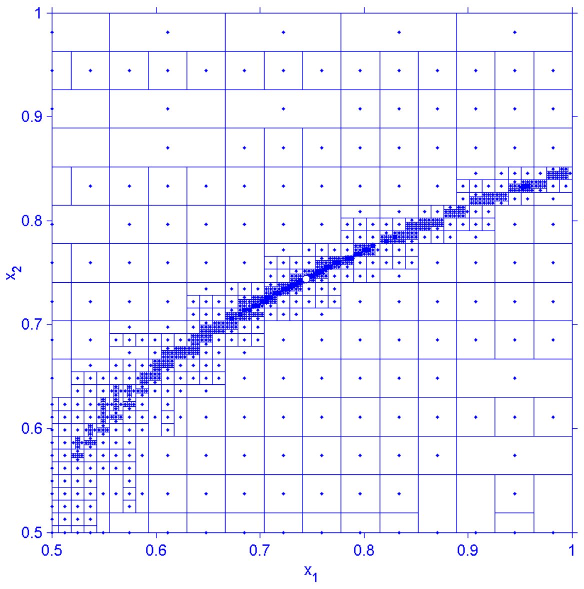


Fig. 3.29 Local zoom-in around the global minimum point

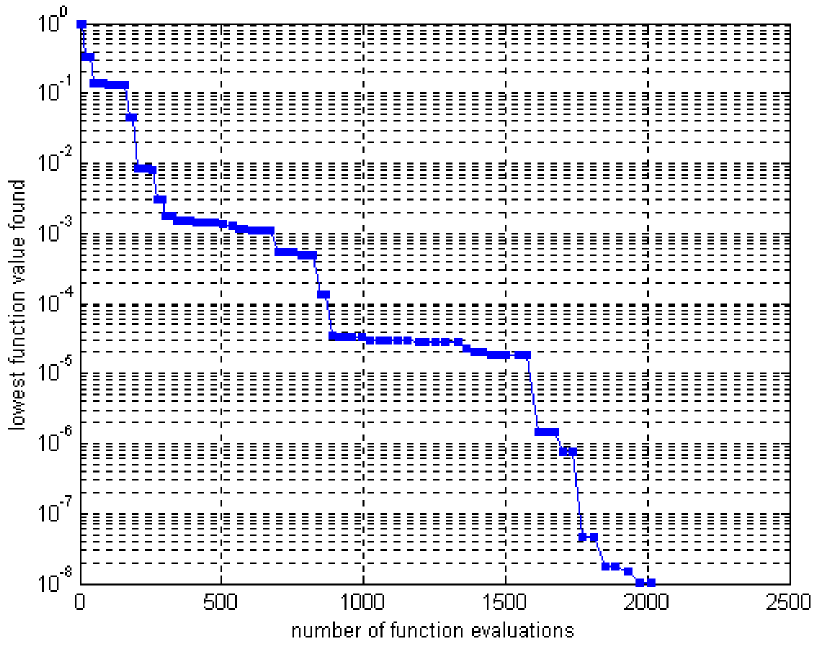


Fig. 3.30 Convergence property for Rosenbrock function case

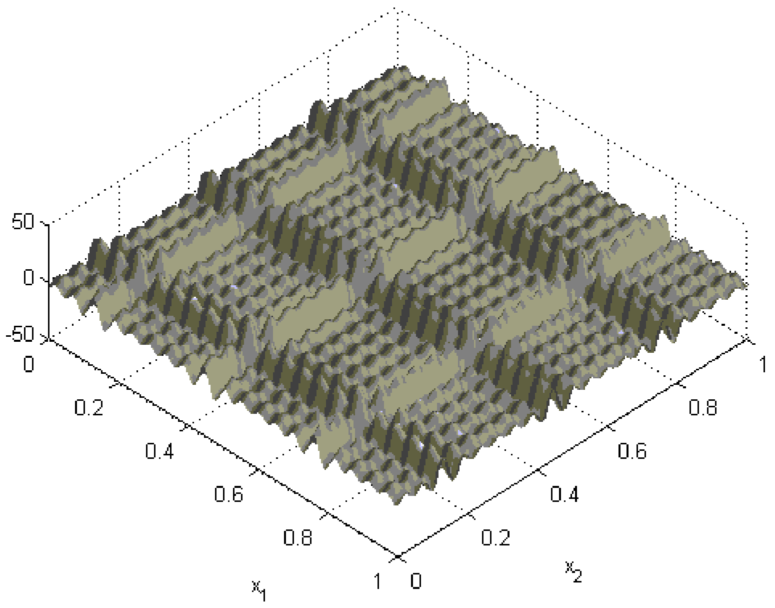


Fig. 3.31 3-D surface of the Shubert function

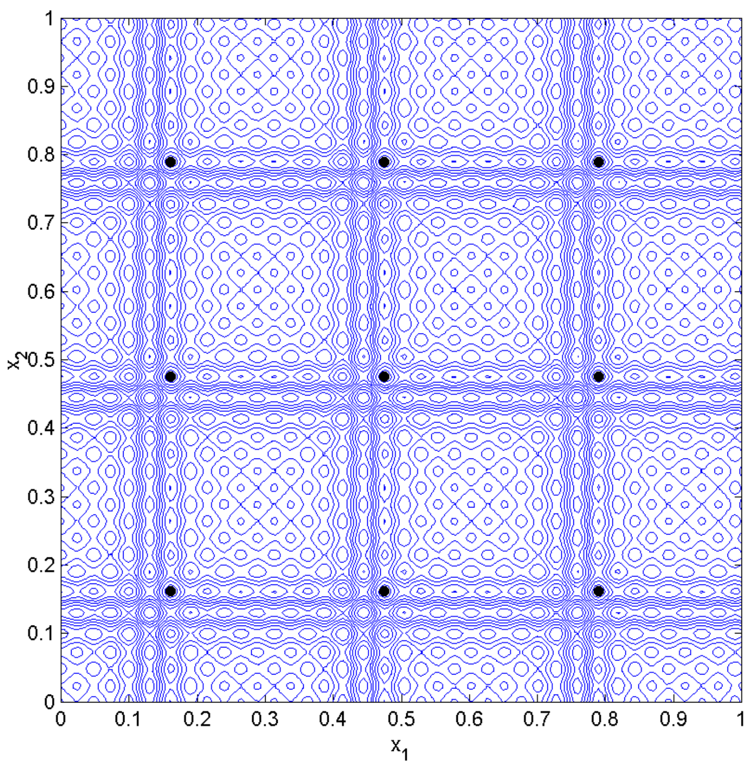


Fig. 3.32 Contour lines of the Shubert function

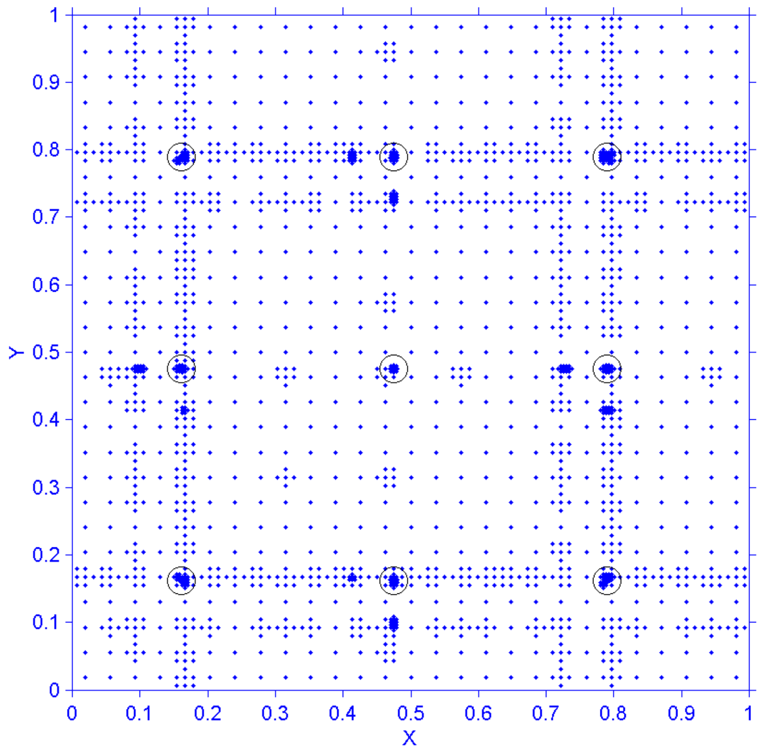


Fig. 3.33 Optimization results for Shubert function

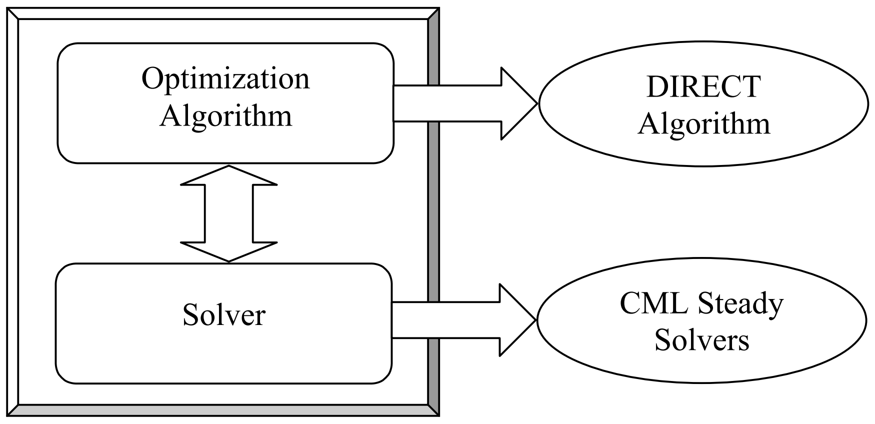


Fig. 3.34 Structure of the optimization program

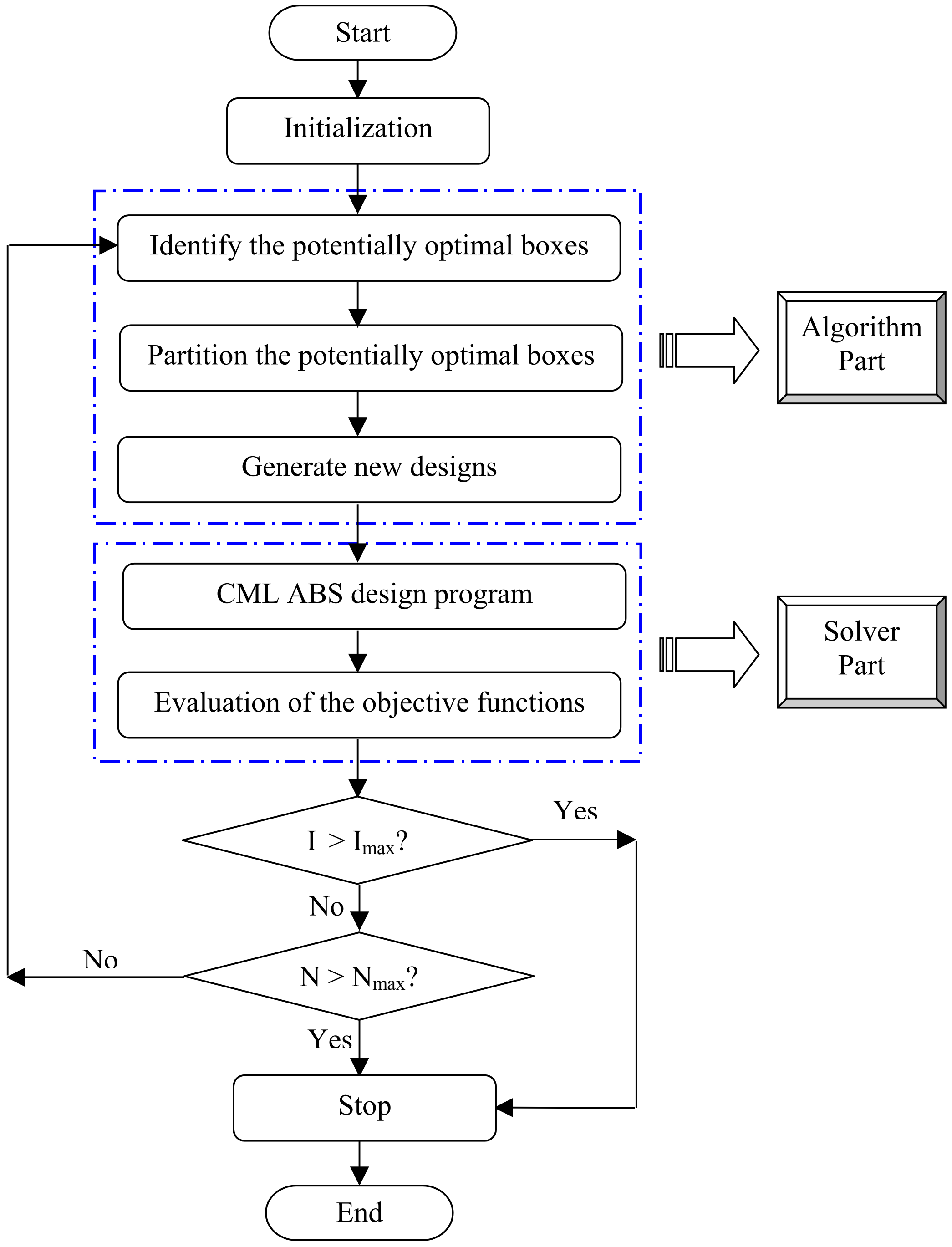


Fig. 3.35 Flow chart of the CML optimization program

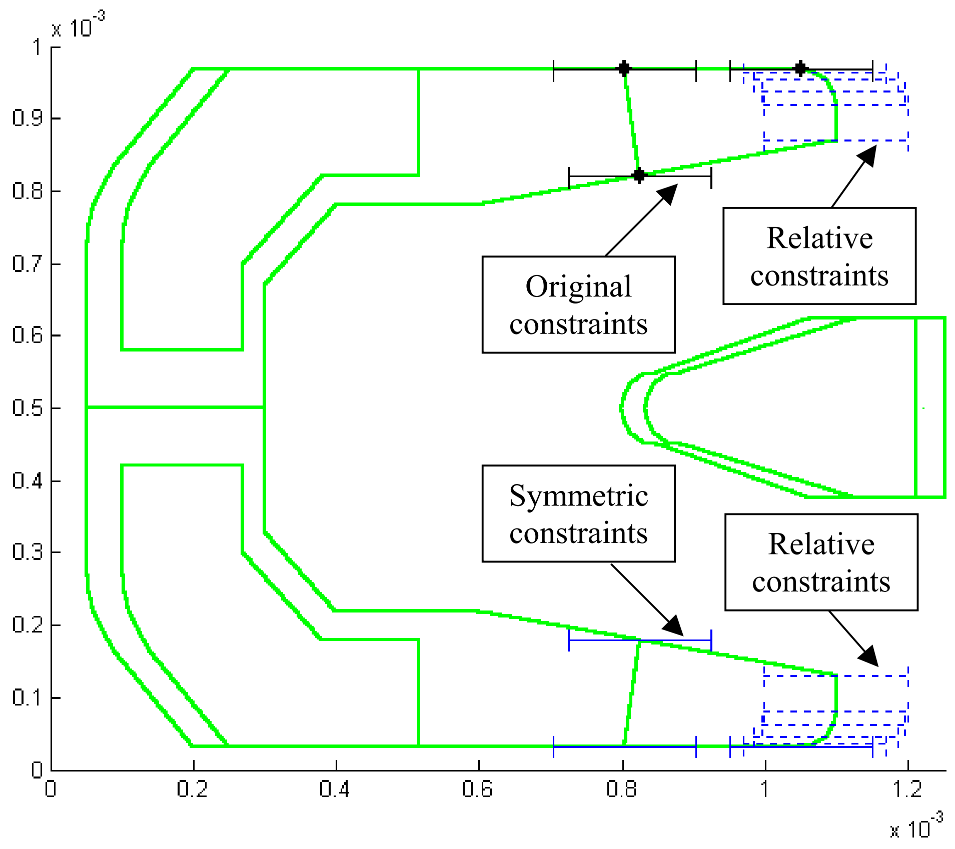


Fig. 3.36 Constraints defined on the initial design

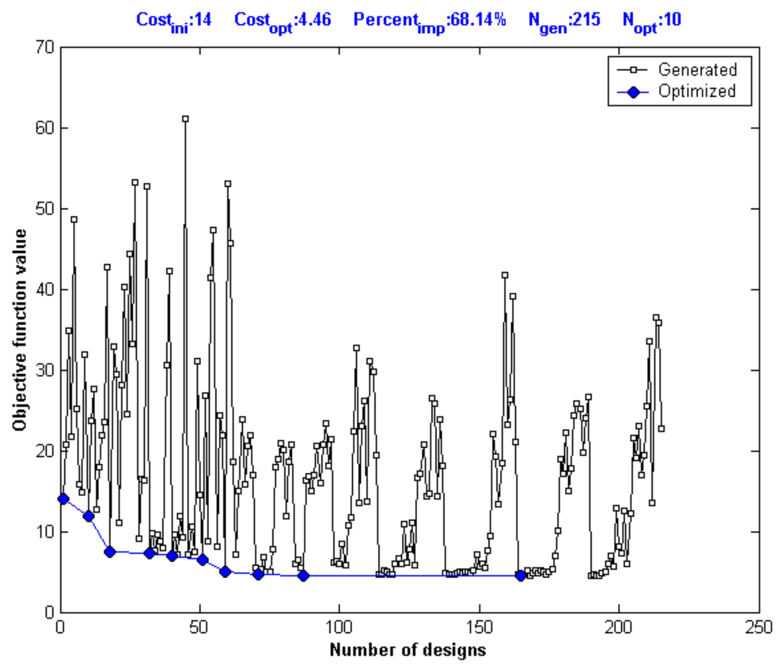


Fig. 3.37 Variation of the objective function value

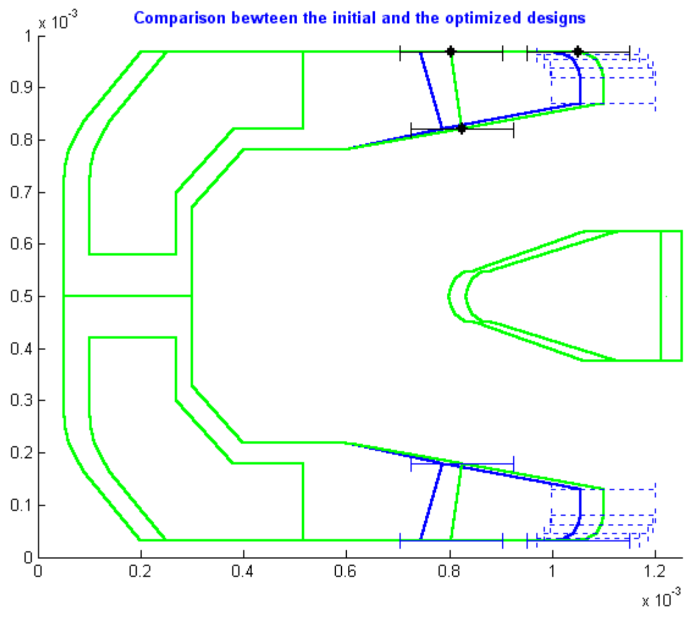


Fig. 3.38 Optimization results

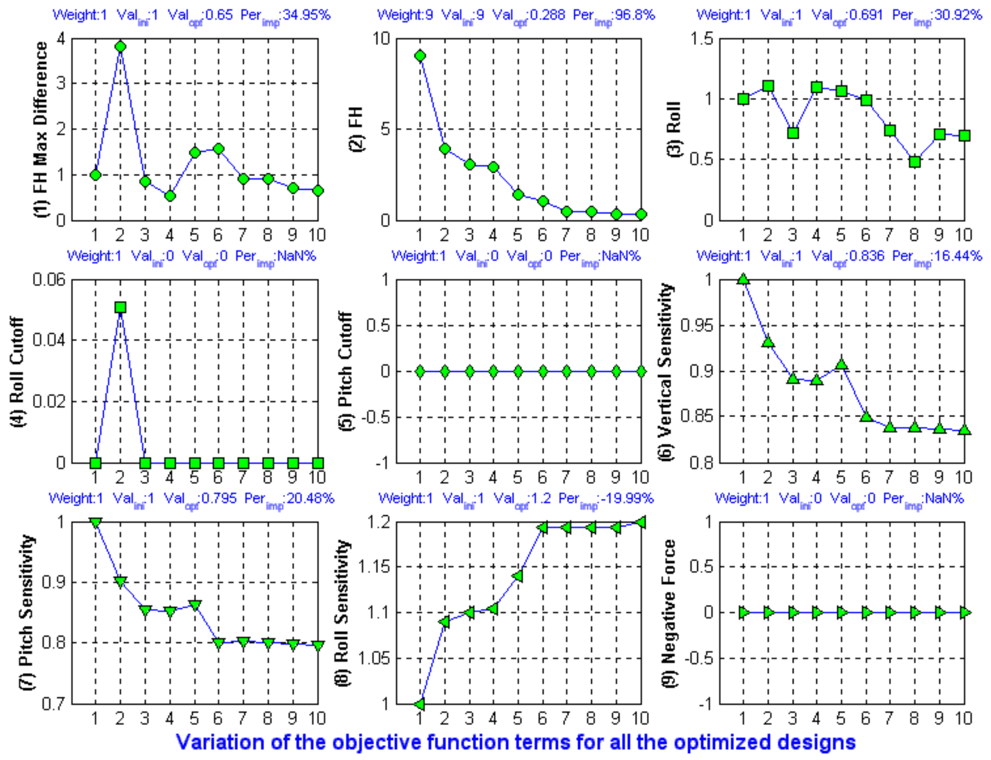


Fig. 3.39 Variations of the objective function terms

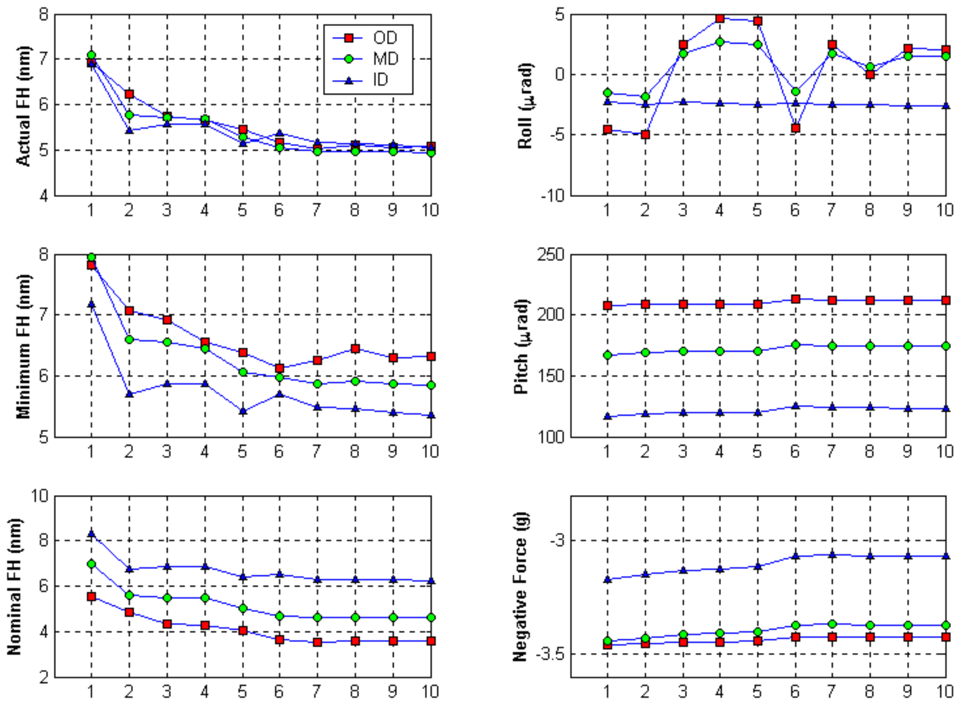


Fig. 3.40 Variations of the slider performance parameters

Chapter 4

LOCALLY BIASED VARIATIONS OF THE DIRECT ALGORITHM

4.1 INTRODUCTION

In the previous chapter, we presented the details of the DIRECT algorithm and the results of numerical experiments as well as its application to the slider Air Bearing Surface (ABS) optimization.

In this chapter, we report on three locally biased variations of the standard DIRECT algorithm. These variations are proposed to further increase the standard DIRECT algorithm's convergence rate and thus improve its efficiency. We first introduce the three variations of the standard DIRECT algorithm. Subsequently, we discuss the results from experimentation using combinations of all forms of these variations. Finally we present results for a test case of slider ABS optimization.

4.2 NUMERICAL METHOD

4.2.1 Locally biased variations of the standard DIRECT algorithm

4.2.1.1 DIRECT algorithm with fewer groups

The first variation to be discussed was developed by Gablonsky and Kelley in 2000, and uses fewer groups. In this thesis, we also refer to it as DIRECT-I.

The only difference between the standard DIRECT algorithm and this variation is the definition of the measure of the groups. For the standard DIRECT algorithm, the group measure is defined as the distance from the center point of a box to its vertices, which is illustrated in Fig. 4.1. For DIRECT-I, the group measure is defined as the length of the longest side of a box, which is illustrated in Fig. 4.2.

From Figs. 4.1 and 4.2 it is clear that by changing the definition of the group measure, DIRECT-I has relatively fewer groups than the standard DIRECT does. Since only the point with the lowest value in a group is eligible to be potentially optimal, DIRECT-I will bias the search toward the local minima.

4.2.1.2 DIRECT algorithm with double partitions

We proposed a second locally biased variation of the standard DIRECT algorithm, also referred to here as DIRECT-II. The purpose of this variation is to partition the box containing the point of the lowest function value twice during each iteration. By doing this, we double the weight on the search around the point with the lowest function value. Thus, the algorithm searches more intensively around the point with the lowest function value.

4.2.1.3 DIRECT algorithm with both features

Based on the above two variations, we propose a third variation which combines their features, i.e. fewer groups and double partitions. This third variation is also referred as DIRECT-III. Since the DIRECT-III algorithm combines the two locally biased features, it is expected that its search will be heavily biased toward the local minima.

4.2.2 Process demonstration of the algorithms

Here we use a 2-D example to demonstrate the process of the DIRECT algorithm and its three locally biased variations. The function used is:

$$F(x_1, x_2) = 10\,(\,|x_1 - 0.4|^{1/2}\,) + 50\,(\,|x_2 - 0.2|^{3/2}\,) \quad \text{where } x_1, x_2 \in [0,1].$$

Figures 4.3a ~ 4.3f, Figs. 4.4a ~ 4.4f, Figs. 4.5a ~ 4.5f and Figs. 4.6a ~ 4.6f show the first 5 iterations for the standard DIRECT algorithm, DIRECT-I, DIRECT-II and DIRECT-III, respectively.

For those pictures on the left-hand side, the x-axis stands for variable x_1 and the y-axis stands for variable x_2, so the unit square is the search space. The shadowed areas are the potentially optimal boxes (can be squares or rectangles) just partitioned. 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.

In the pictures on the right-hand side, the horizontal axis stands for the group measure and the vertical axis stands for the function value. The circular points represent the center points of the non-optimal boxes. The solid round points represent the center points of the potentially optimal boxes. The lines connecting these potentially optimal points form the convex hull of all the data points.

Figure 4.3a shows the initial state of the standard DIRECT algorithm. In this state only one central point is evaluated and it is designated as the potentially optimal point. The box located in this way is partitioned as shown in Fig. 4.3b.

$s_1 = \min \{13, 14.8\} \quad = \ 13$

$s_2 = \min \{3.47, 28.4\} \ = \ 3.47$

The x_2 direction (y) gets partitioned first, followed by the x_1 direction (x). Because only one potentially optimal point is chosen in Fig. 4.3b, only one box containing that point is partitioned in Fig. 4.3c. The rectangle is only partitioned along its longest side.

For the DIRECT-I algorithm, which defines the longest side length of a box as its group measure, fewer groups will be used. For the standard DIRECT algorithm, the numbers of groups from the initial state to the fifth iteration are: 1, 2, 2, 3, 3, 5; for DIRECT-I, the numbers are: 1, 2, 2, 2, 2, 3. Since DIRECT-I uses fewer groups, it also has fewer potentially optimal points at each iteration and the search will be more locally focused.

For the DIRECT-II algorithm, the box containing the lowest function value is partitioned twice during each iteration. Because its definition of the group measure is the same as that of the standard DIRECT algorithm, and because more boxes will be partitioned during each iteration, the result should be that more groups in each iteration are obtained compared with the standard DIRECT algorithm. From Figs. 4.5a ~ 4.5f, we see that the numbers of groups from the initial state to the fifth iteration for DIRECT-II are: 1, 2, 3, 5, 7, 8.

Because the DIRECT-III algorithm combines the above two locally biased measures, it has a strong locally orientated search strategy.

If we compare Figs. 4.3f, 4.4f, 4.5f and 4.6f, which are the results at iteration five for DIRECT, DIRECT-I, DIRECT-II and DIRECT-III respectively, it is clear that DIRECT-II and DIRECT-III generate more sample points around the local minimum points for a given number of iterations than does DIRECT. Since DIRECT-I generates fewer sample points in each iteration as compared with the standard DIRECT algorithm, it follows that, if the number of function evaluations is fixed, all three locally biased variations of the standard DIRECT algorithm will have more sample points around the local minima than would the standard DIRECT algorithm.

4.3 NUMERICAL EXPERIMENTS

4.3.1 Testing functions with one global and local minimum

The testing functions used here include 2-D, 3-D, 5-D, 10-D and 20-D functions. These functions have only one global and local minimum point, and the minimum values of these functions are zero. They 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: $F(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (x_1 - 0.1)^2 + (x_2 - 0.2)^2 + (x_3 - 0.3)^2 + (x_4 - 0.4)^2 + (x_5 - 0.5)^2 + (x_6 - 0.6)^2 + (x_7 - 0.7)^2 + (x_8 - 0.8)^2 + (x_9 - 0.9)^2 + (x_{10} - 1.0)^2.$

20-D: $F(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}, x_{20}) =$

$$(x_1 - 0.05)^2 + (x_2 - 0.1)^2 + (x_3 - 0.15)^2 + (x_4 - 0.2)^2 +$$
$$(x_5 - 0.25)^2 + (x_6 - 0.3)^2 + (x_7 - 0.35)^2 + (x_8 - 0.4)^2 +$$
$$(x_9 - 0.45)^2 + (x_{10} - 0.5)^2 + (x_{11} - 0.55)^2 + (x_{12} - 0.6)^2 +$$
$$(x_{13} - 0.65)^2 + (x_{14} - 0.7)^2 + (x_{15} - 0.75)^2 + (x_{16} - 0.8)^2 +$$
$$(x_{17} - 0.85)^2 + (x_{18} - 0.9)^2 + (x_{19} - 0.95)^2 + (x_{20} - 1.0)^2.$$

For all these cases, $x_i \in [0,1]$, $i = 1,\ldots 20$. The results for the 2-D case are shown in Figs. 4.7 ~ 4.11.

Figures 4.7 ~ 4.10 show the optimization results obtained by using DIRECT, DIRECT-I, DIRECT-II and DIRECT-III respectively. From Figs. 4.7 ~ 4.11 it's clear that all four algorithms converge to the global minimum point. However, the locally biased variations focus more on a local search, thus leaving a relatively larger unexplored area. Using DIRECT, we have only one unexplored box (the largest one, with side length of 1/3). However, we have 5, 3 and 6 such boxes for DIRECT-I, DIRECT-II and DIRECT-III, respectively. Figure 4.11 shows the convergence comparison among these four algorithms.

Figures 4.12 ~ 4.15 show the convergence comparison for the 3-D, 5-D, 10-D and 20-D cases, respectively. In these figures the locally biased variations generally have a faster convergence rate than does the standard DIRECT algorithm. DIRECT-I and DIRECT-II have similar convergence rates. DIRECT-III, which combines those two locally biased measures, has the fastest convergence rate. Since higher dimension problems require a larger search space, a fairly fast convergence rate is of great importance in obtaining the global minimum within the limited number of function evaluations. Since DIRECT-III has the fastest convergence rate, it shows superior performance for higher dimension problems such as the 10-D and the 20-D problems.

As an example, let's take a look at the 20-D case, whose optimization results are shown by Figure 4.16. Table 4.1 shows the values of the best points found by the three locally biased variations within 2000 function evaluations and the exact solution. Table 4.1 also shows the relative errors. The results obtained by using DIRECT-III have the smallest relative errors among the standard DIRECT algorithm and its three variations. From Fig. 4.16 and Table 4.1, it is clear that, after 2000 function evaluations, the results obtained by using DIRECT-III almost match the exact solution precisely, far better than the results obtained by using DIRECT, DIRECT-I and DIRECT-II.

4.3.2 Testing functions with one global minimum and multiple local minima

Here we consider two functions, the Rosenbrock function and the local Shubert function.

The Rosenbrock function is a standard test function in optimization theory, and is defined as: $F(x_1, x_2) = 100\ (x_1 - x_2^2)^2 + (1 - x_2)^2$, 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 the global minimum point is (0.74414, 0.74414) and the global minimum is 0. The global minimum point is located in a long narrow flat valley with lots of local minima. The contour lines and the surface shape of the Rosenbrock function are shown in Figs. 3.26 and 3.27. The convergence comparison in Fig. 4.17 shows that DIRECT-I and DIRECT-III have faster convergence rates in the early stage. However, all four algorithms show similar convergence rates in the final phase.

The second function is the local Shubert function, which 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 $x_1, x_2 \in [-5, 5]$. It has one global minimum point and 100 local minimum points. If we normalize the range of variables x_1 and x_2 into $[0,1]$, then its global minimum point is (0.4508609, 0.4508609) and the global minimum is -24.062499. The 2-D contour and 3-D surface of the local Shubert function are shown in Figs. 4.18 and 4.19, respectively. The round dot in Fig. 4.18 denotes the global minimum point.

The convergence comparison given in Fig. 4.20 shows that, for this case, DIRECT has the fastest convergence rate and DIRECT-III has the slowest. However, all the algorithms converge to the same global minimum point. It is easy to imagine that, because the local Shubert function has many local minimum points, it would be likely for the locally biased variations of the DIRECT algorithm to spend too much time on the local searches. These locally biased variations get to the global minimum point slower than does the standard DIRECT algorithm.

4.3.3 Testing functions with multiple global and local minima

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 4 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. 4.21 and 4.22. The round dots in Fig. 4.21 represent the global minimum points. We can clearly discern the six "humps" from these two figures. Figures 4.23 ~ 4.26 show the optimization results obtained by using DIRECT, DIRECT-I, DIRECT-II and DIRECT-III, respectively. 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 can observe the strongly biased property of the DIRECT-III by looking at its large unexplored area.

It's clear from Figs. 4.23 ~ 4.26 that sample points cluster around the two global minimum points for both the standard DIRECT algorithm and its variations. Thus, all algorithms found the two global minimum points. Figure 4.27 shows the convergence comparison. The DIRECT algorithm and its variations show very similar fast convergence rates.

The second testing function we 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. 4.28 and 4.29. The five round dots in Fig. 4.28 represent the global minimum points.

Figures 4.30 ~ 4.33 show the optimization results after 500 function evaluations using DIRECT, DIRECT-I, DIRECT-II and DIRECT-III, respectively. The tiny dots in the figures on the left represent the sample points. The figures on the right are the local zoom-ins of the ones on the left, around the global minimum points. The centers of the circles denote the locations of the global minimum points.

Figure 4.30 shows that DIRECT found all of the five global minimum points at this stage, while Figs. 4.31, 4.32 and 4.33 show that only four global minimum points were found by the three locally biased variations of DIRECT.

Figure 4.34 shows the convergence comparison. The locally biased DIRECT algorithms clearly have higher convergence rates than does the standard DIRECT algorithm with DIRECT-III having the fastest convergence rate among them.

Therefore, for this testing case the locally biased DIRECT algorithms have faster convergence rate, but the standard DIRECT algorithm found all the global minimum points more quickly.

Next we will discuss the performance of the standard DIRECT algorithm and its three locally biased variations in the slider Air Bearing Surface (ABS) optimization.

4.4 SLIDER AIR BEARING DESIGN OPTIMIZATION CASE

4.4.1 Air bearing design optimization problem

We choose the same 7nm FH Pico slider defined in Section 2.4 as the prototype slider. Its rail shape and the 3-dimensional rail geometry are shown in Figs. 2.10 and 2.11, respectively. The optimization goals are also the same, i.e., to lower its 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. And we define the same objective function as in Section 2.4. The constraints defined are as same as the ones defined in Section 3.5, which are shown in Fig. 3.36.

4.4.2 Optimization results

Using the initial design, constraints and objective function, we carried out the optimization using the DIRECT algorithm and its three locally biased variations. Figure 4.35 shows the convergence comparison. For this testing case, all four algorithms show a similarly fast convergence rate. The best objective function values obtained by using DIRECT, DIRECT-I, DIRECT-II and DIRECT-III are 4.46, 4.46, 4.43 and 4.43. Figures 4.36 ~ 4.39 show the optimized ABS designs obtained after 200 function evaluations by using DIRECT, DIRECT-I, DIRECT-II and DIRECT-III, respectively. In these figures, the gray lines show the rail shape of the initial design and the dark lines show the rail shape of the optimized design. The four optimized ABS designs are almost the same. All four optimized ABS designs have very uniform flying heights around the target 5nm FH, and a reasonably flat roll profile.

4.5 CONCLUSION

We carried out extensive numerical experiments using the DIRECT algorithm and its three locally biased variations, i.e., DIRECT-I (having fewer groups), DIRECT-II (having double partitions for the box containing the point with the lowest function value), and DIRECT-III (which combines these two measures).

For test functions with only one global and local minimum point, all the locally biased variations have faster convergence rates than does the standard DIRECT algorithm. DIRECT-I and DIRECT-II have similar convergence rates, whereas DIRECT-III has a faster convergence rate. For higher dimension problems, DIRECT-III is superior to the other three and can find the global minimum point far more quickly.

For test functions with one global minimum point and multiple local minimum points, it's hard to tell which algorithm is best. Though they might show different convergence properties at some stages, they all show a similar convergence trend in the long run.

For test functions with multiple global and local minimum points, the locally biased variations have a similar or higher convergence rate than does the standard DIRECT algorithm. However, the standard DIRECT finds all of the global minimum points earlier.

The slider ABS optimization problem is a strongly nonlinear problem. The results of the test case show very similar performance for DIRECT and its three variations.

In summary, 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.

	Best Point					**Relative Error (%)**			
	DIRECT	**DIRECT -I**	**DIRECT -II**	**DIRECT -III**	**Exact Solution**	**DIRECT**	**DIRECT -I**	**DIRECT -II**	**DIRECT -III**
x_1	0.167	0.056	0.167	0.051	0.05	234.00	12.00	234.00	2.00
x_2	0.167	0.093	0.167	0.105	0.10	67.00	-7.00	67.00	5.00
x_3	0.167	0.167	0.167	0.154	0.15	11.33	11.33	11.33	2.67
x_4	0.167	0.204	0.167	0.204	0.20	-16.50	2.00	-16.50	2.00
x_5	0.167	0.241	0.167	0.253	0.25	-33.20	-3.60	-33.20	1.20
x_6	0.167	0.315	0.278	0.302	0.30	-44.33	5.00	-7.33	0.67
x_7	0.500	0.352	0.389	0.352	0.35	42.86	0.57	11.14	0.57
x_8	0.500	0.389	0.500	0.401	0.40	25.00	-2.75	25.00	0.25
x_9	0.500	0.463	0.500	0.451	0.45	11.11	2.89	11.11	0.22
x_{10}	0.500	0.500	0.500	0.500	0.50	0.00	0.00	0.00	0.00
x_{11}	0.500	0.500	0.500	0.549	0.55	-9.09	-9.09	-9.09	-0.18
x_{12}	0.500	0.537	0.500	0.599	0.60	-16.67	-10.50	-16.67	-0.17
x_{13}	0.500	0.537	0.611	0.648	0.65	-23.08	-17.38	-6.00	-0.31
x_{14}	0.833	0.796	0.722	0.698	0.70	19.00	13.71	3.14	-0.29
x_{15}	0.833	0.648	0.833	0.747	0.75	11.07	-13.60	11.07	-0.40
x_{16}	0.833	0.796	0.833	0.796	0.80	4.13	-0.50	4.13	-0.50
x_{17}	0.833	0.648	0.833	0.846	0.85	-2.00	-23.76	-2.00	-0.47
x_{18}	0.833	0.944	0.833	0.895	0.90	-7.44	4.89	-7.44	-0.56
x_{19}	0.833	0.648	0.944	0.944	0.95	-12.32	-31.79	-0.63	-0.63
x_{20}	0.944	0.944	0.944	0.998	1.00	-5.60	-5.60	-5.60	-0.20

Table 4.1 Summary of the optimization results for the 20-D case

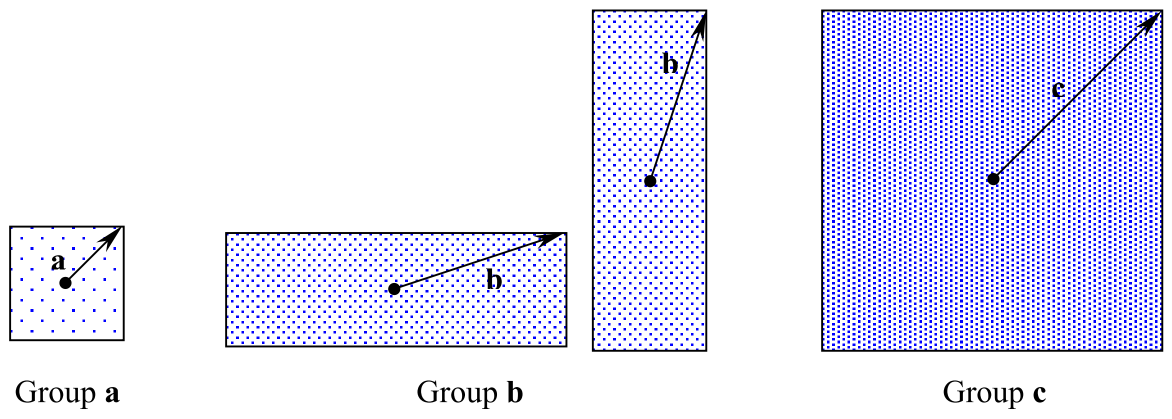


Fig. 4.1 Example of different groups in standard DIRECT

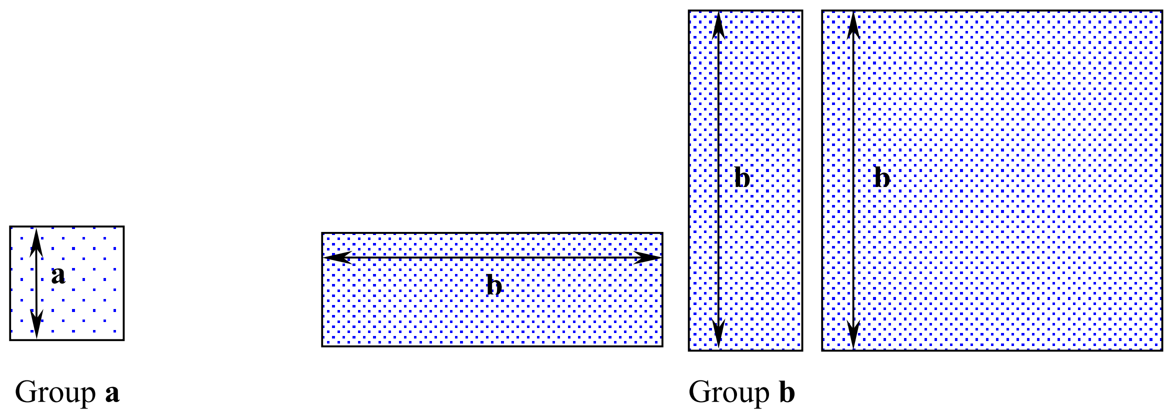


Fig. 4.2 Example of different groups in DIRECT-I

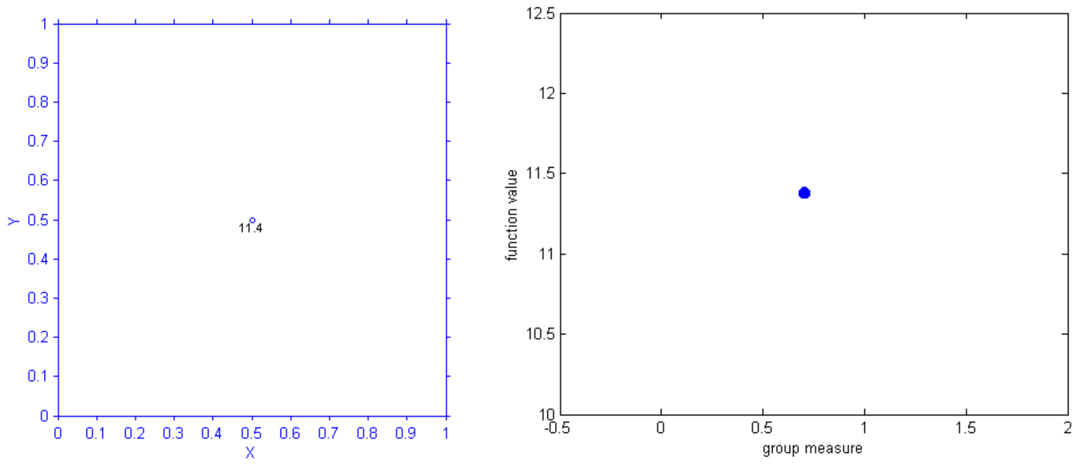


Fig. 4.3a DIRECT initial state

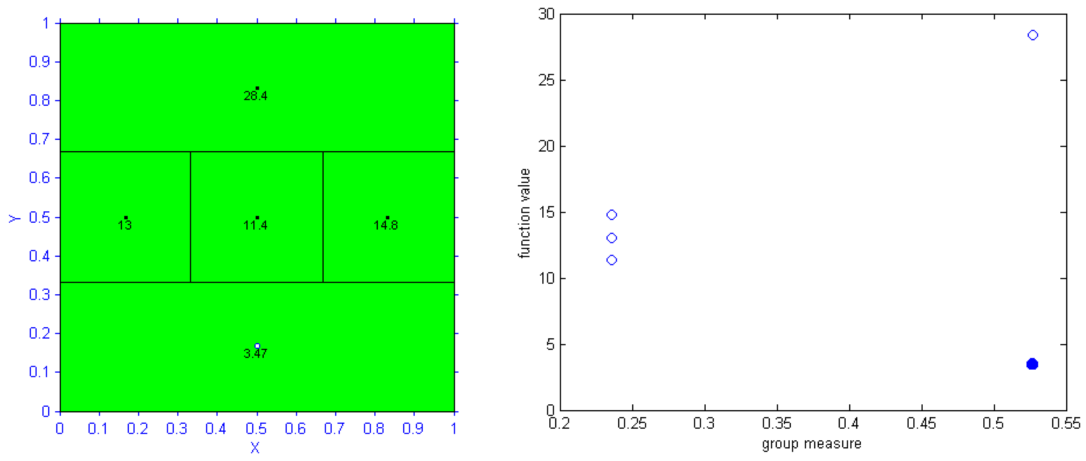


Fig. 4.3b DIRECT iteration 1

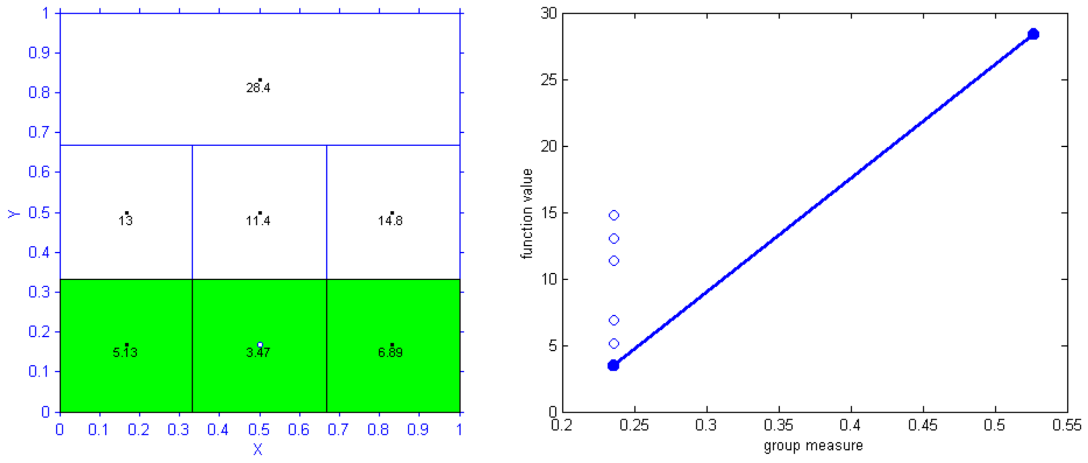


Fig. 4.3c DIRECT iteration 2

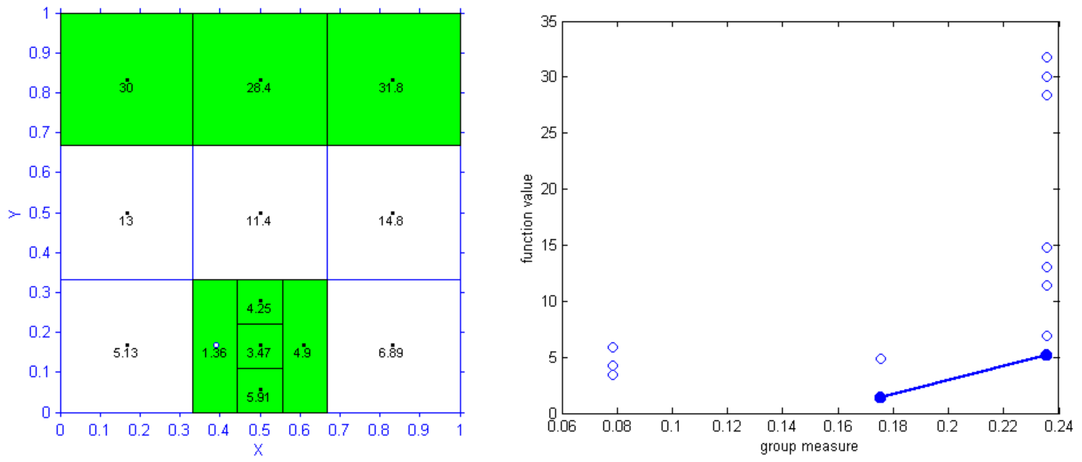


Fig. 4.3d DIRECT iteration 3

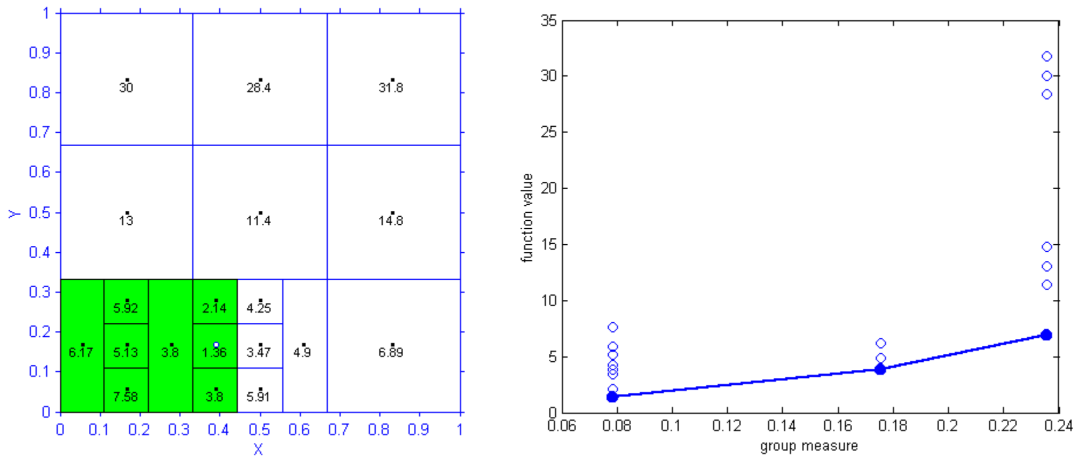


Fig. 4.3e DIRECT iteration 4

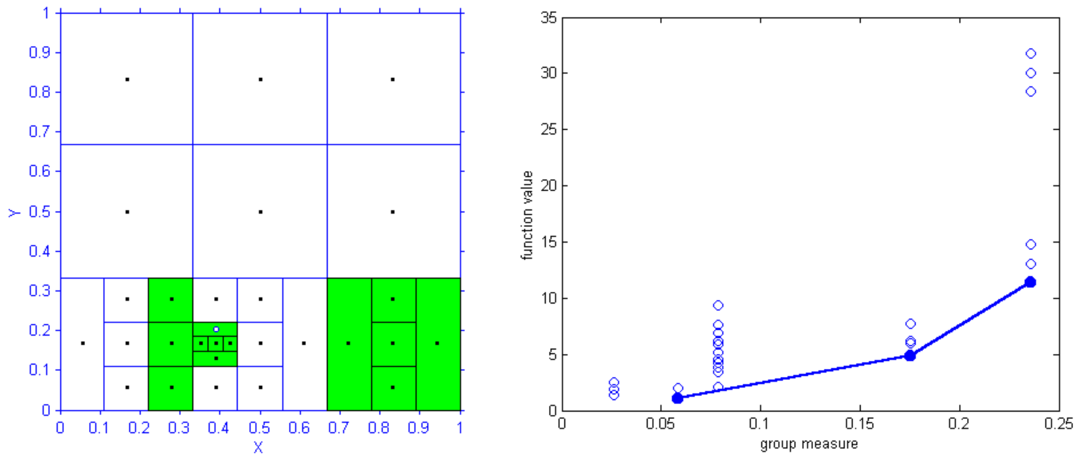


Fig. 4.3f DIRECT iteration 5

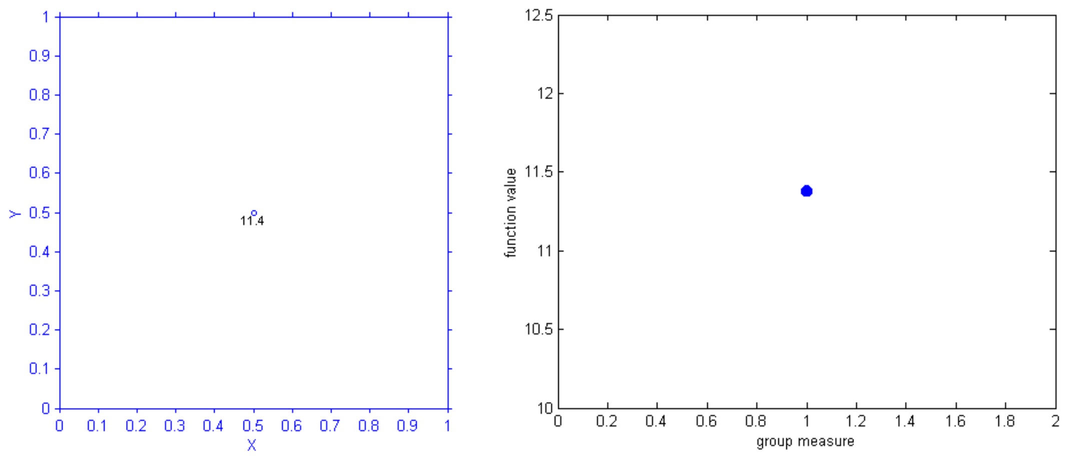


Fig. 4.4a DIRECT-I initial state

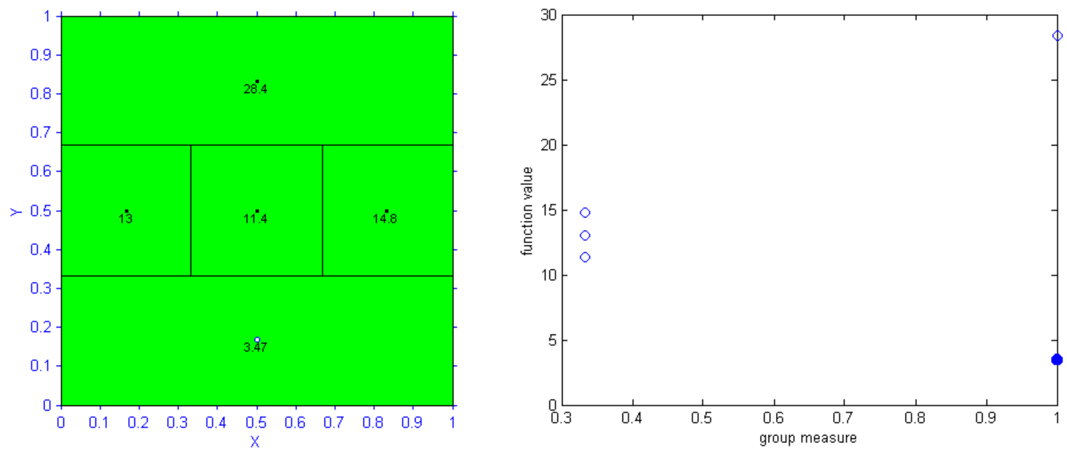


Fig. 4.4b DIRECT-I iteration 1

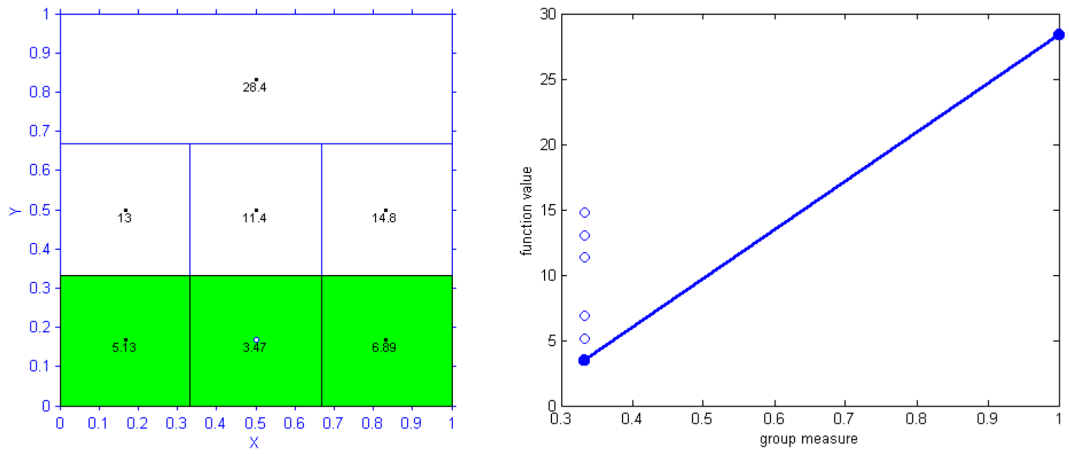


Fig. 4.4c DIRECT-I iteration 2

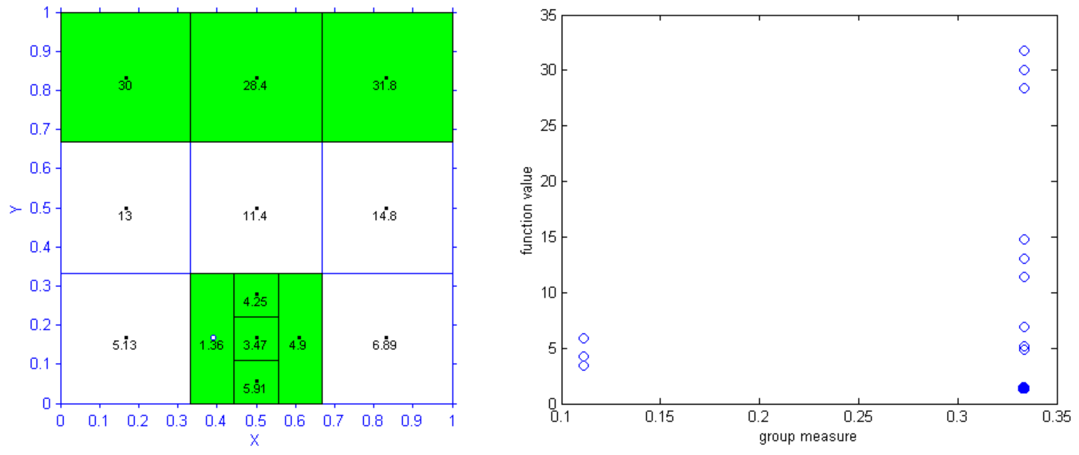


Fig. 4.4d DIRECT-I iteration 3

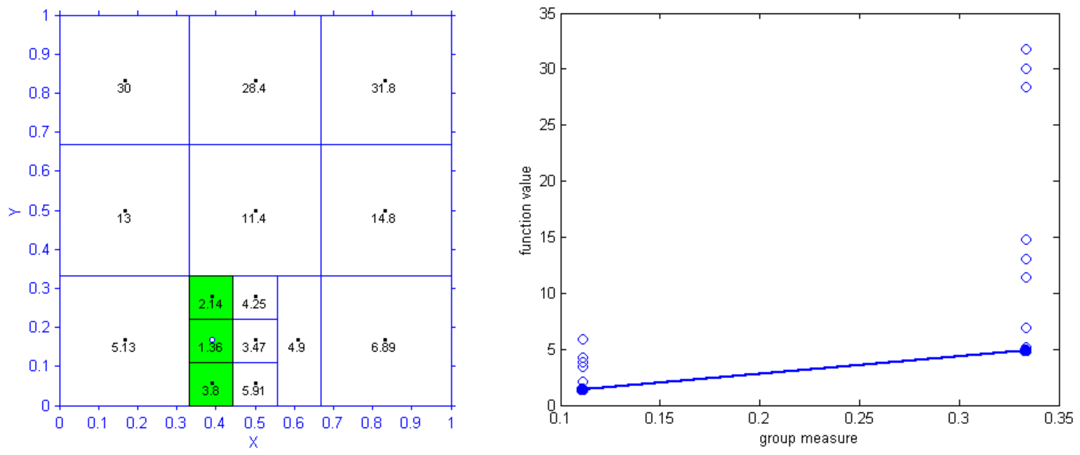


Fig. 4.4e DIRECT-I iteration 4

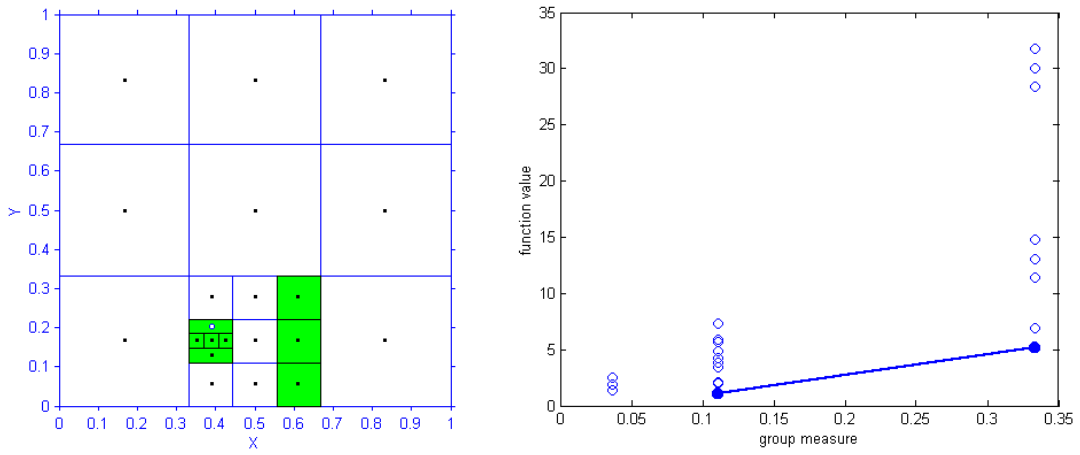


Fig. 4.4f DIRECT-I iteration 5

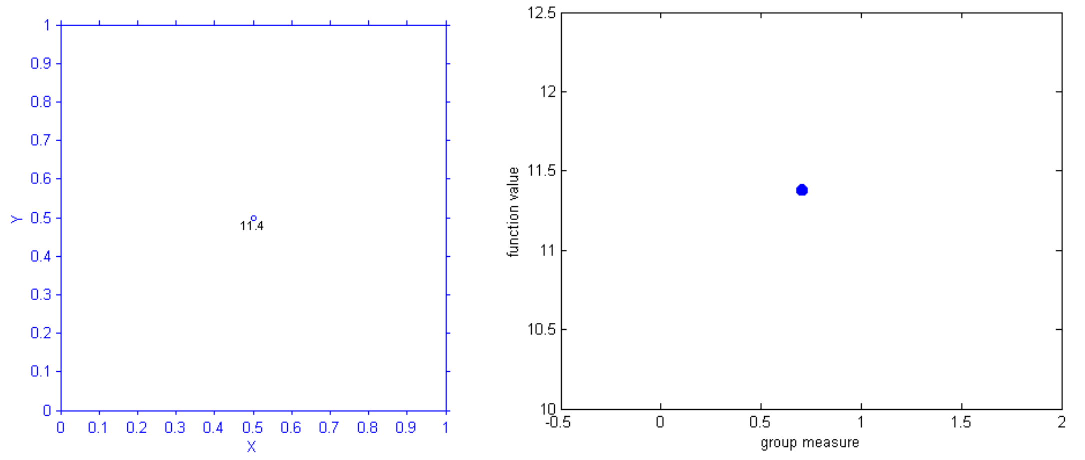


Fig. 4.5a DIRECT-II initial state

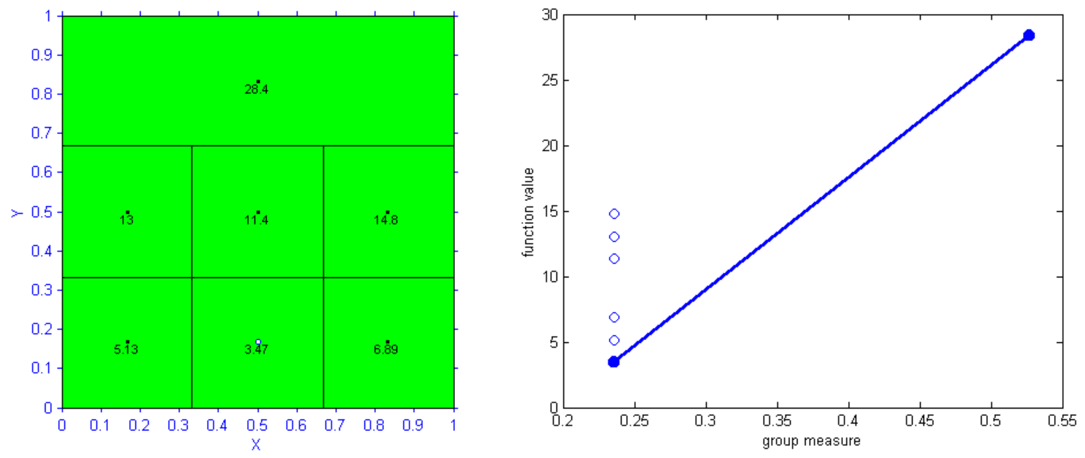


Fig. 4.5b DIRECT-II iteration 1

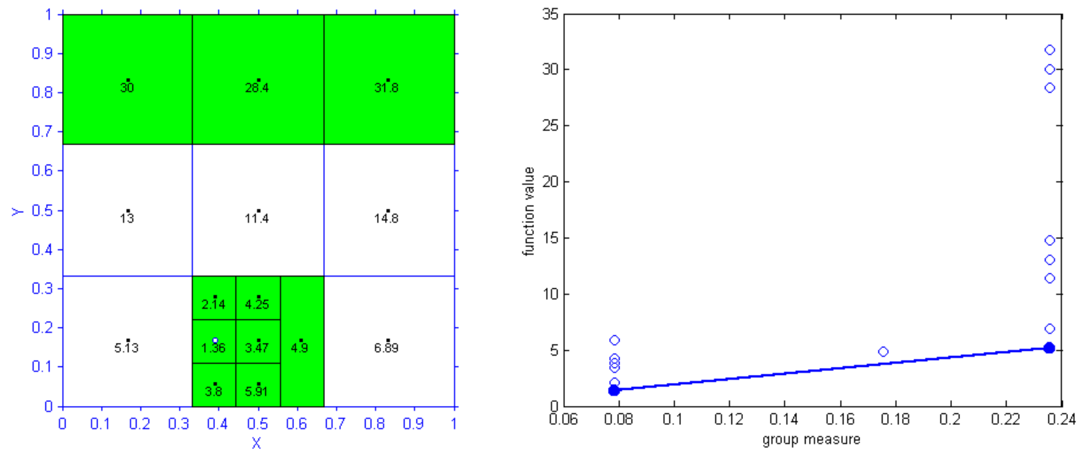


Fig. 4.5c DIRECT-II iteration 2

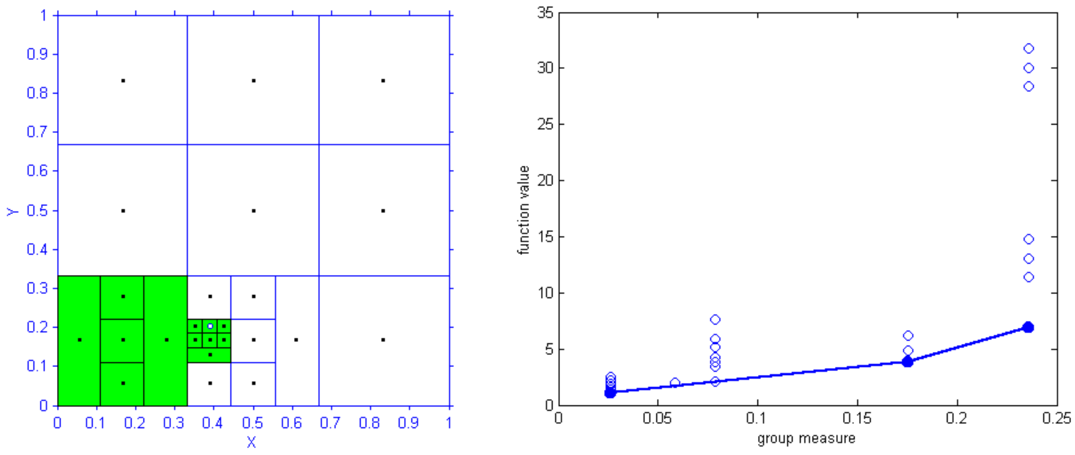


Fig. 4.5d DIRECT-II iteration 3

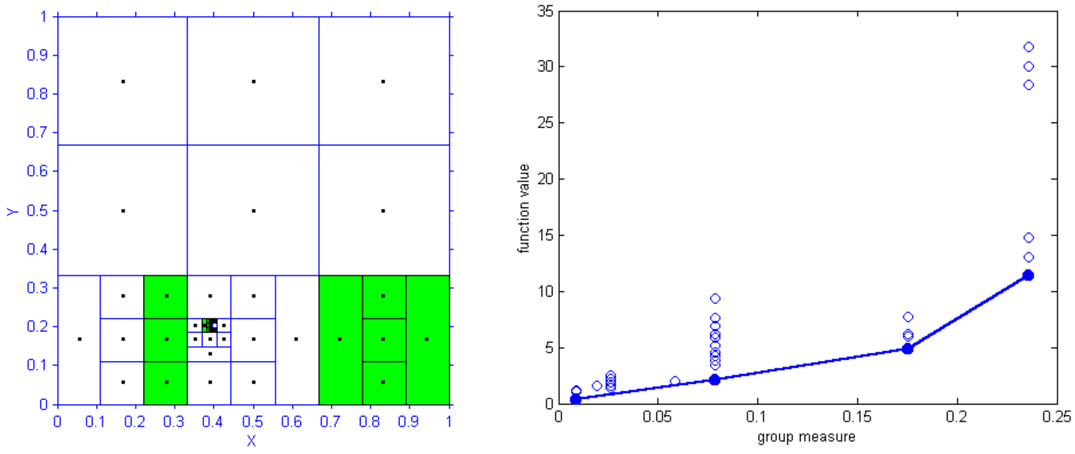


Fig. 4.5e DIRECT-II iteration 4

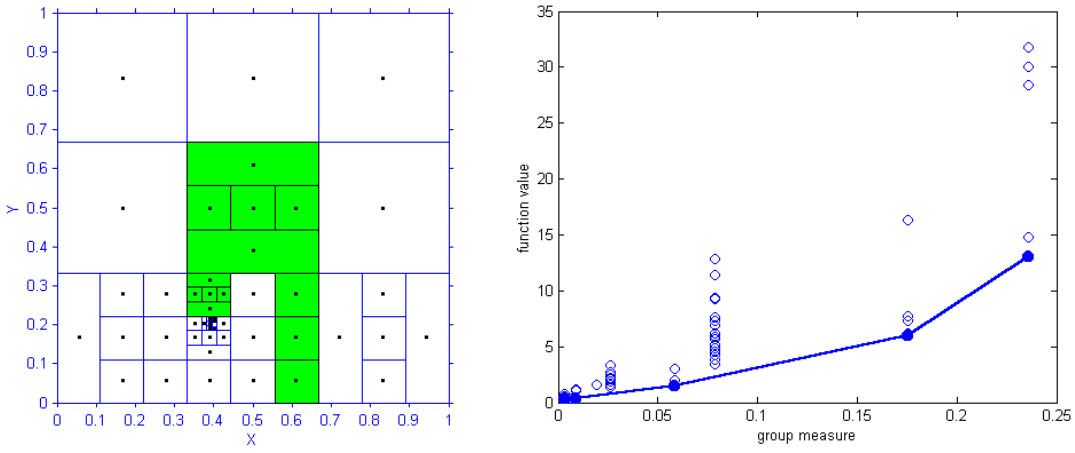


Fig. 4.5f DIRECT-II iteration 5

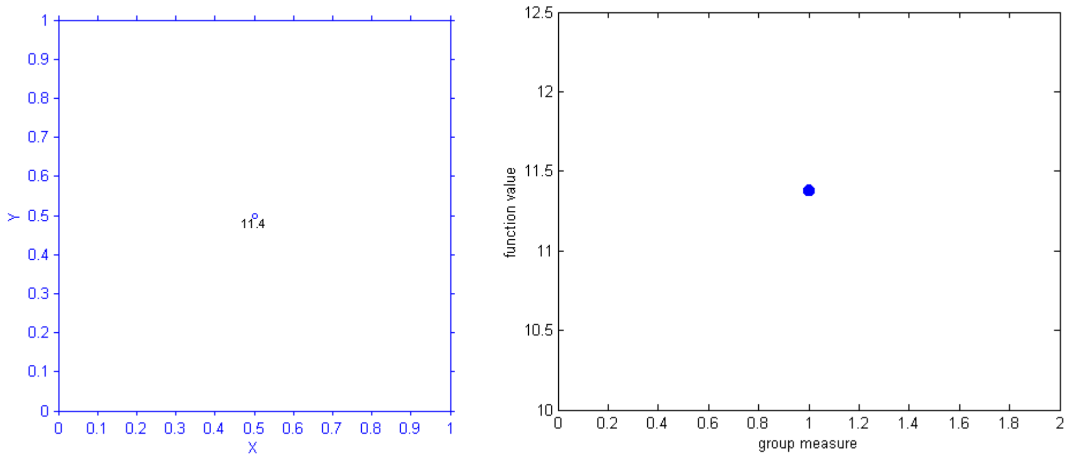


Fig. 4.6a DIRECT-III initial state

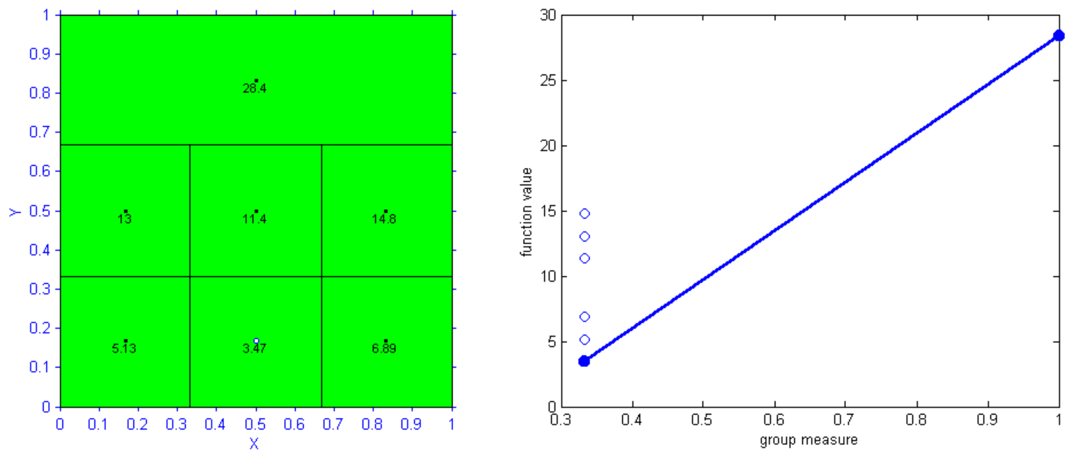


Fig. 4.6b DIRECT-III iteration 1

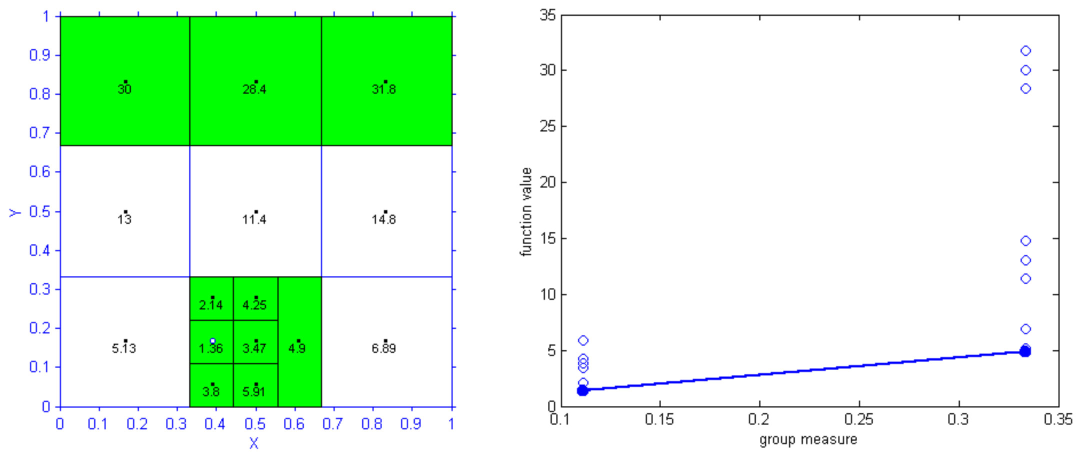


Fig. 4.6c DIRECT-III iteration 2

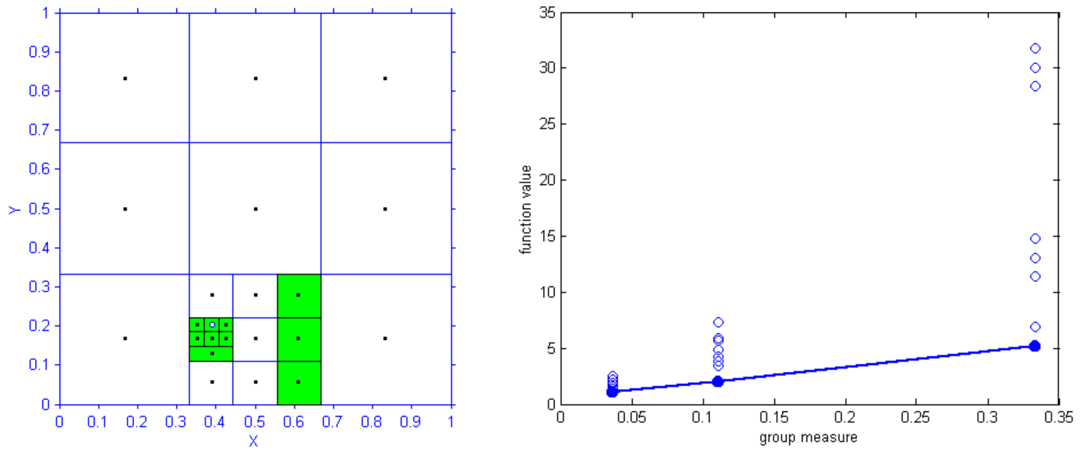


Fig. 4.6d DIRECT-III iteration 3

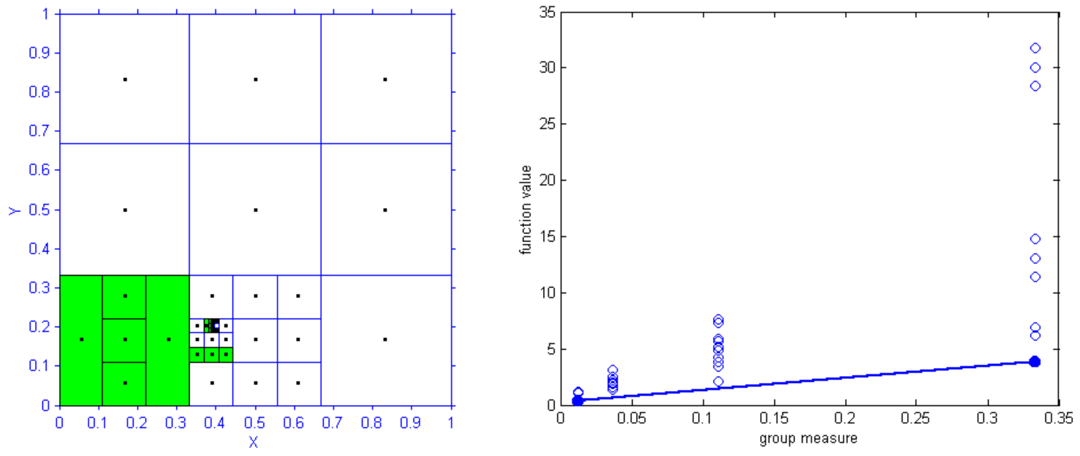


Fig. 4.6e DIRECT-III iteration 4

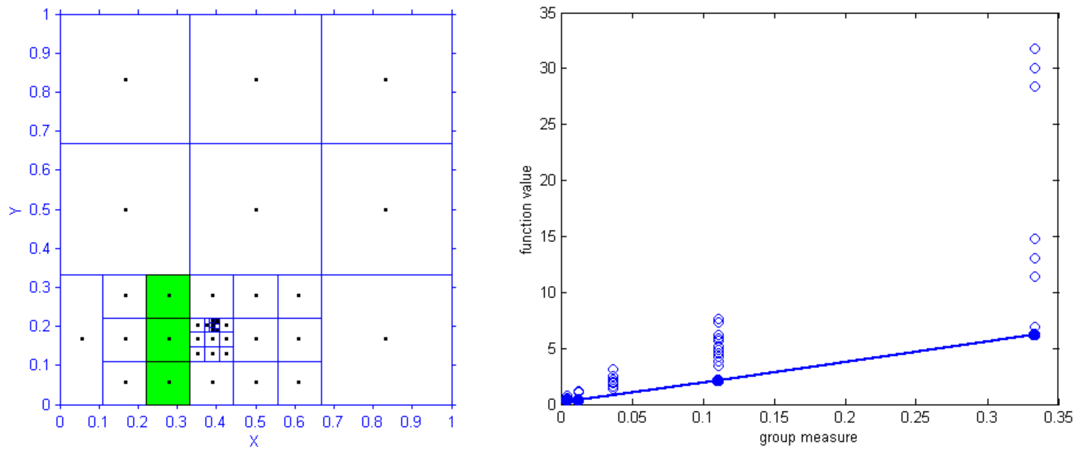


Fig. 4.6f DIRECT-III iteration 5

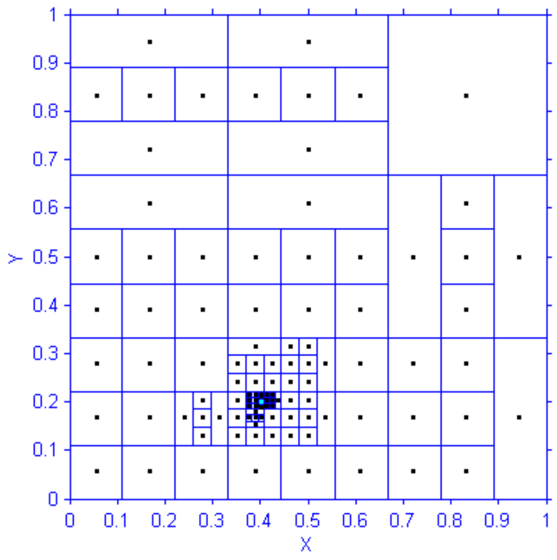


Fig. 4.7 Results of DIRECT

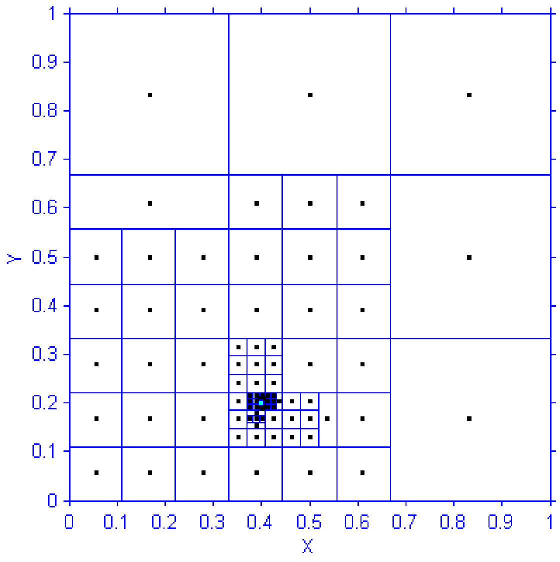


Fig. 4.8 Results of DIRECT-I

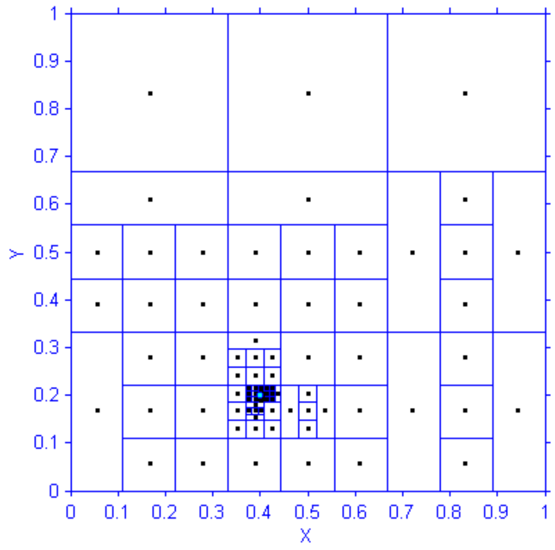


Fig. 4.9 Results of DIRECT-II

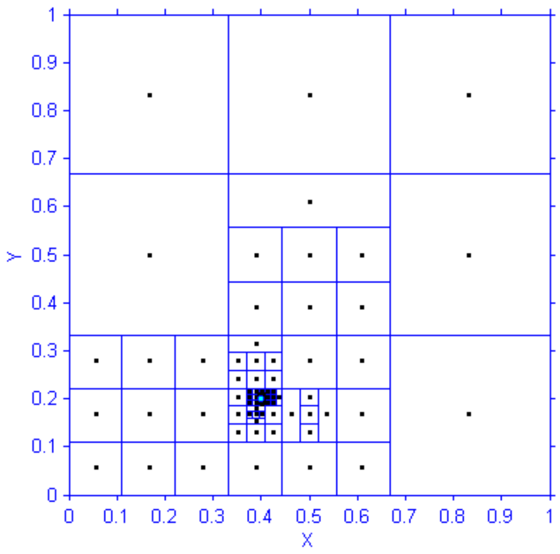


Fig. 4.10 Results of DIRECT-III

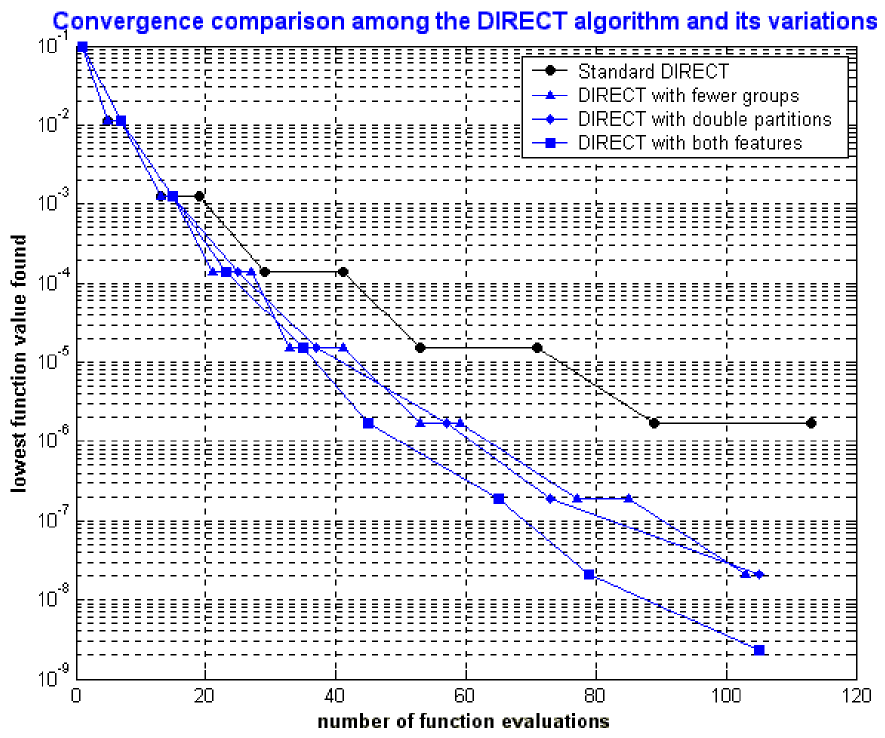


Fig. 4.11 Convergence comparison for the 2-D case

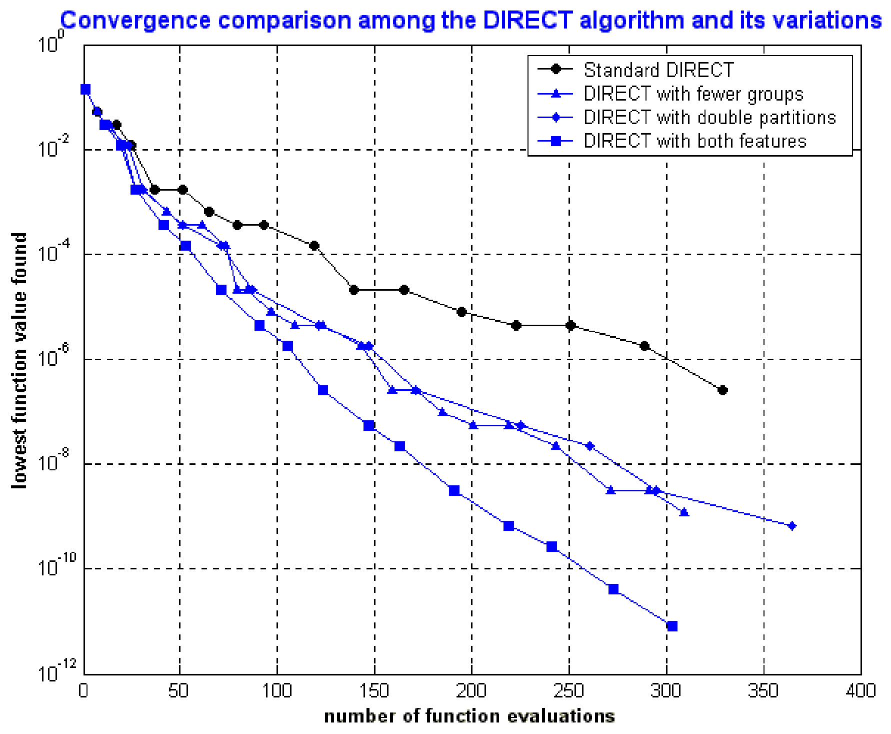


Fig. 4.12 Convergence comparison for the 3-D case

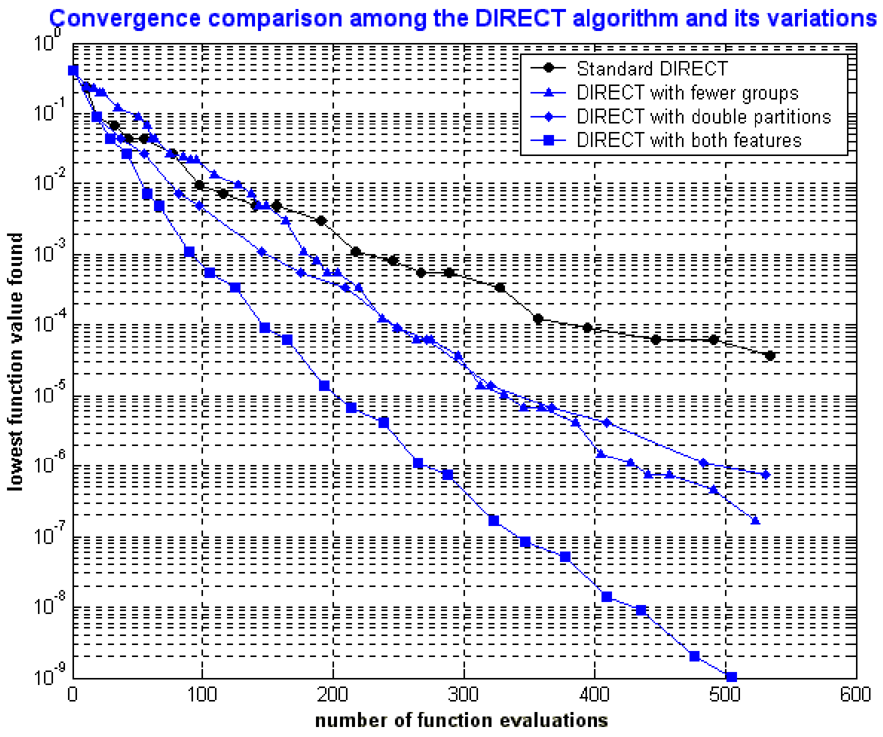


Fig. 4.13 Convergence comparison for the 5-D case

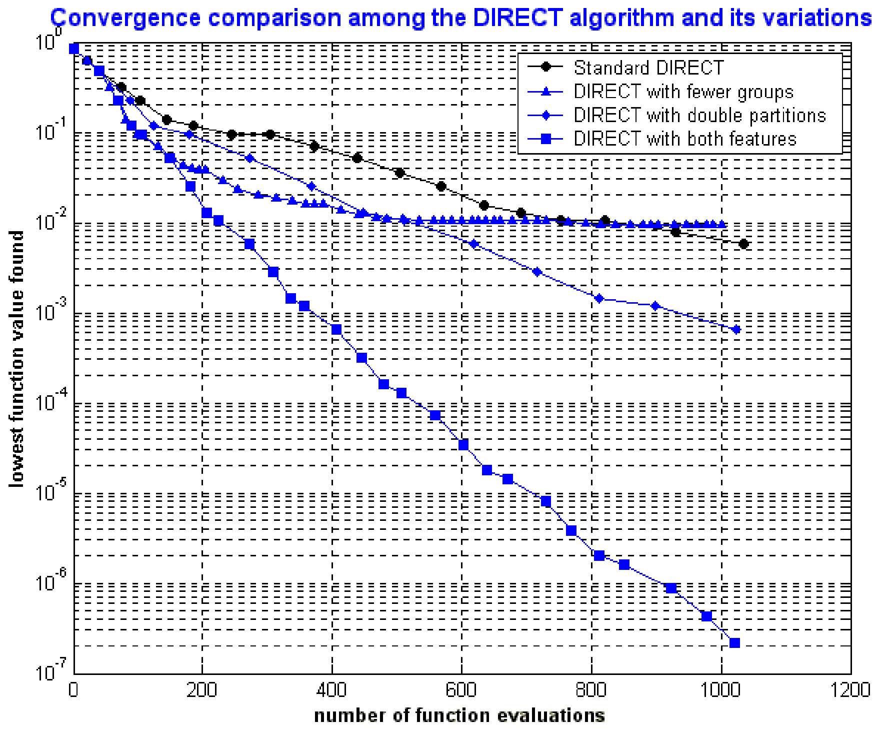


Fig. 4.14 Convergence comparison for the 10-D case

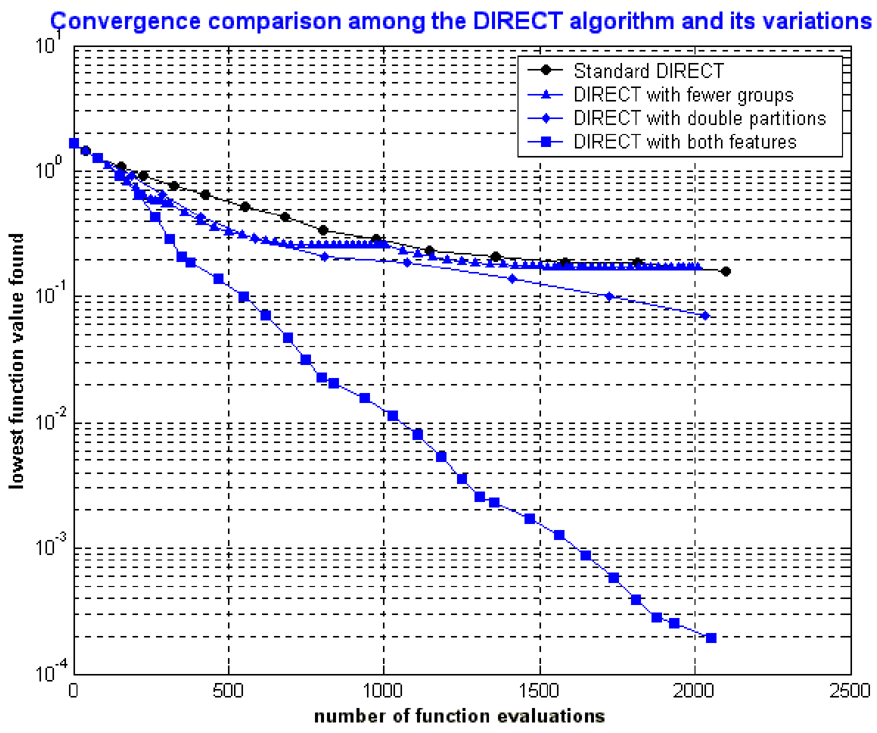


Fig. 4.15 Convergence comparison for the 20-D case

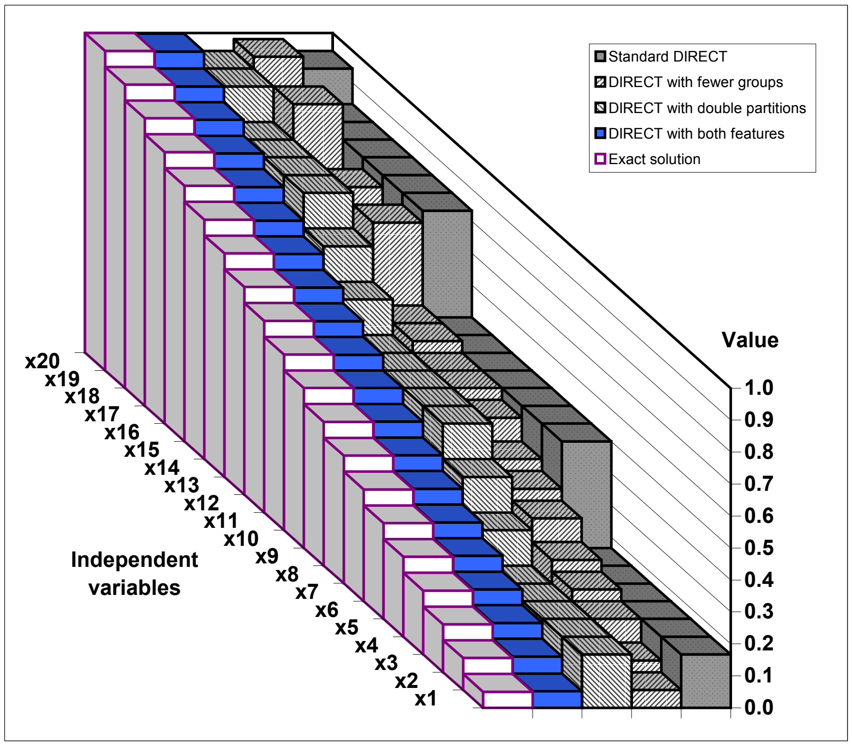


Fig. 4.16 Comparison of the final optimization results for the 20-D case

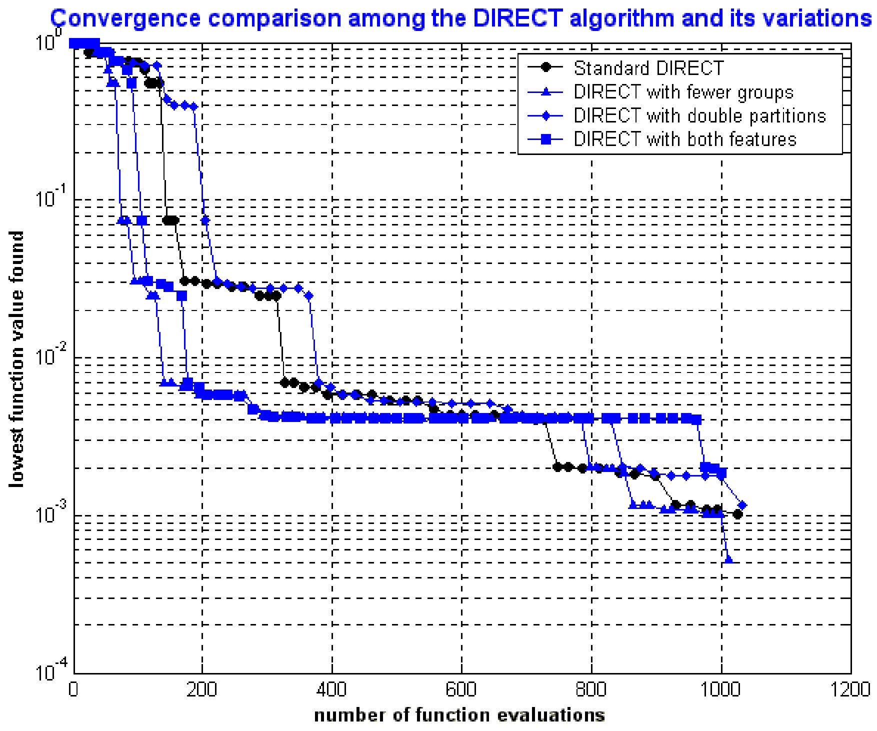
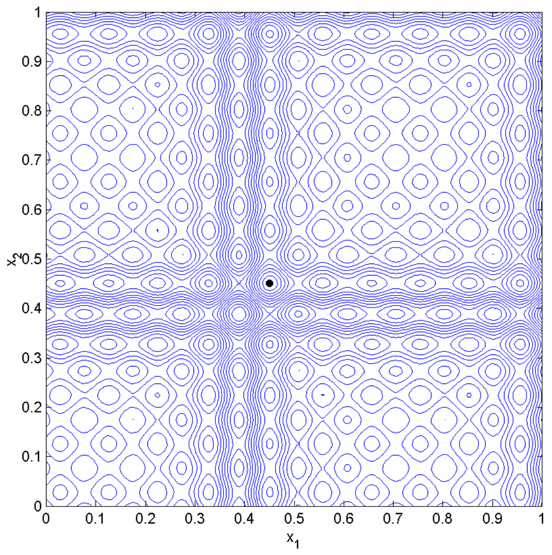


Fig. 4.17 Convergence comparison for the Rosenbrock function



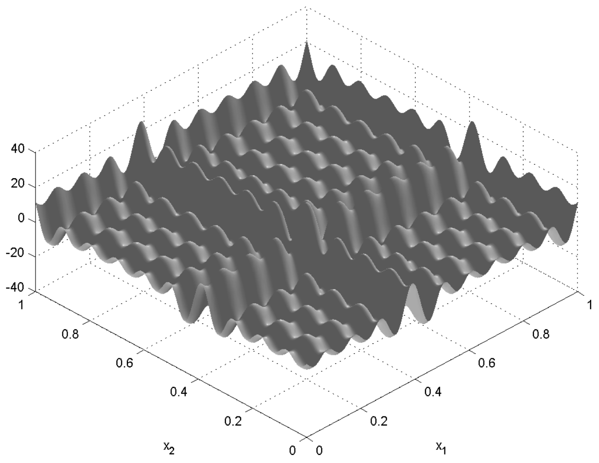


Fig. 4.18 Contour lines

Fig. 4.19 Surface shape

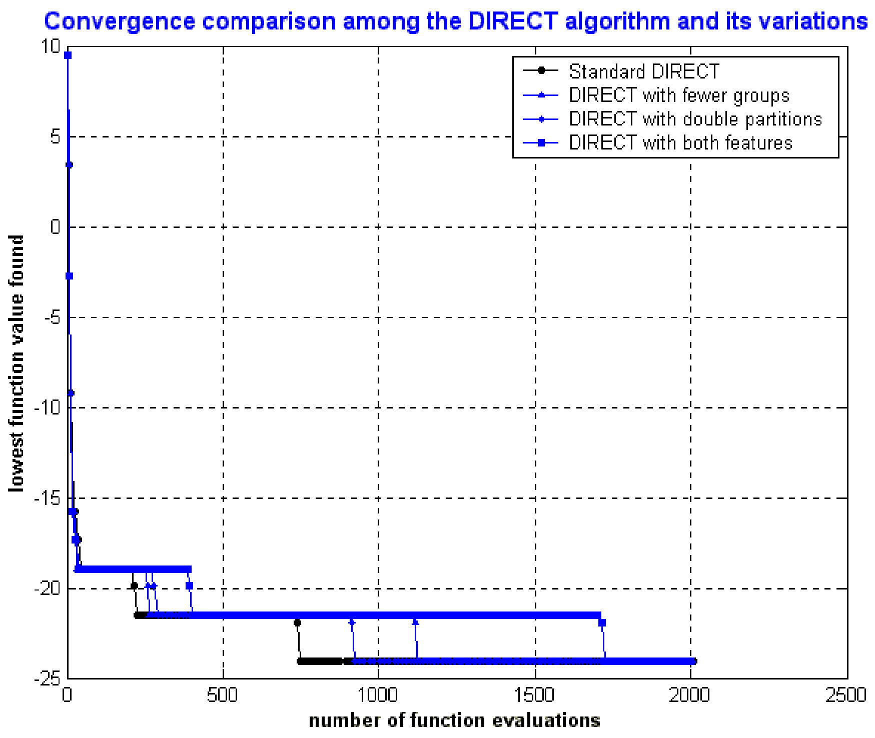
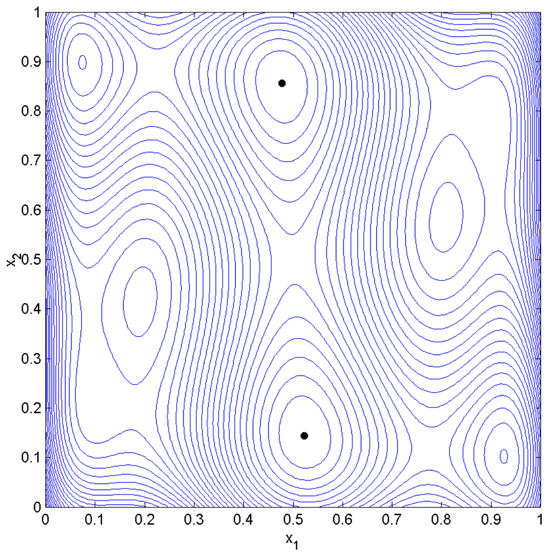


Fig. 4.20 Convergence comparison for the local Shubert function



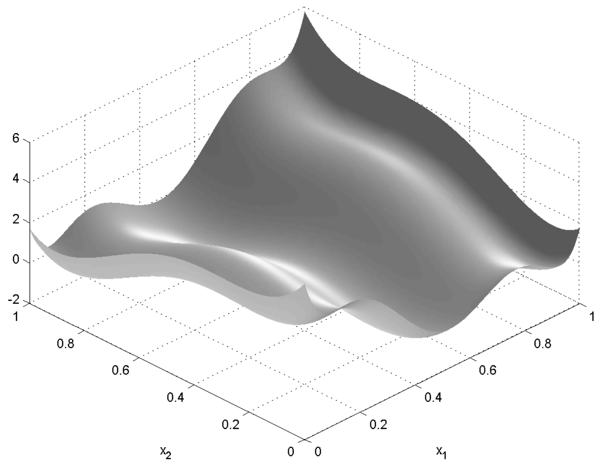


Fig. 4.21 Contour lines

Fig. 4.22 Surface shape

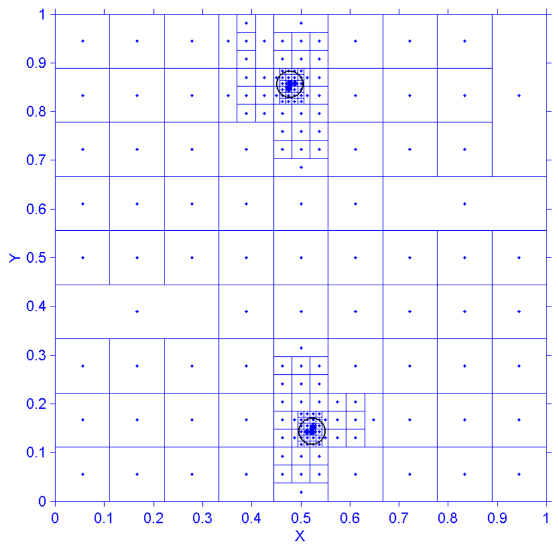


Fig. 4.23 Results of DIRECT

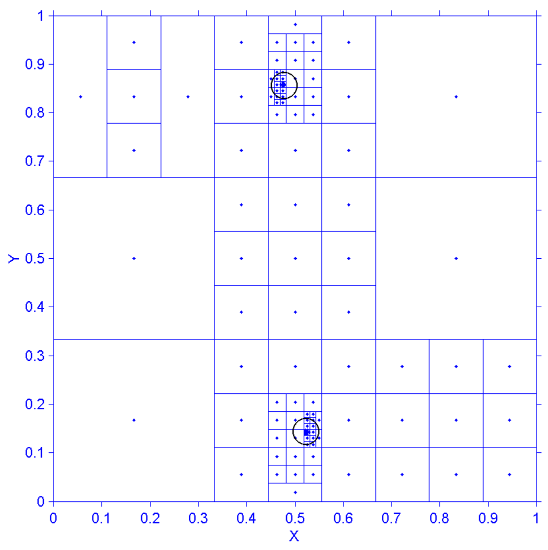


Fig. 4.24 Results of DIRECT-I

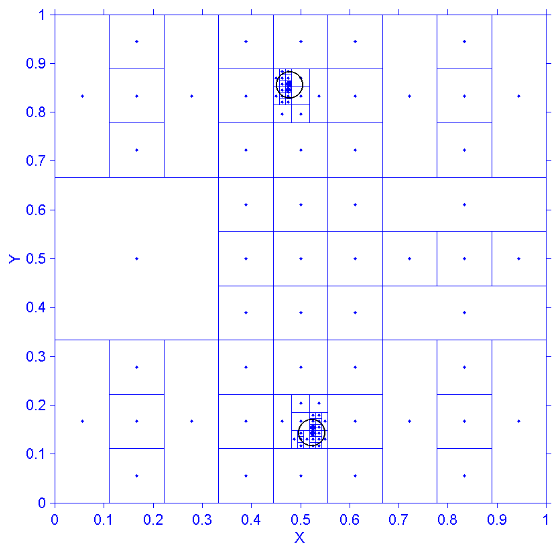


Fig. 4.25 Results of DIRECT-II

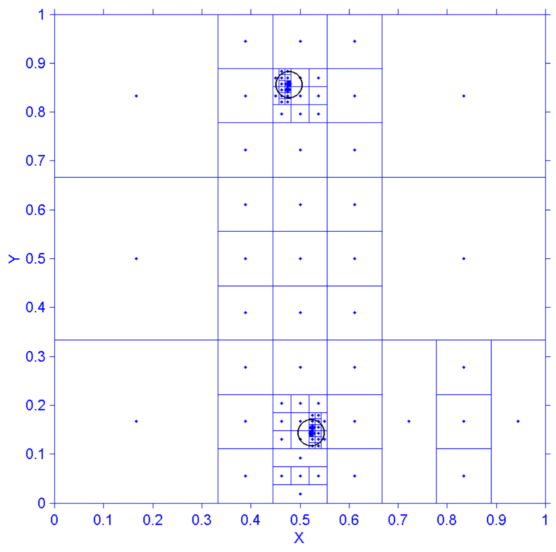


Fig. 4.26 Results of DIRECT-III

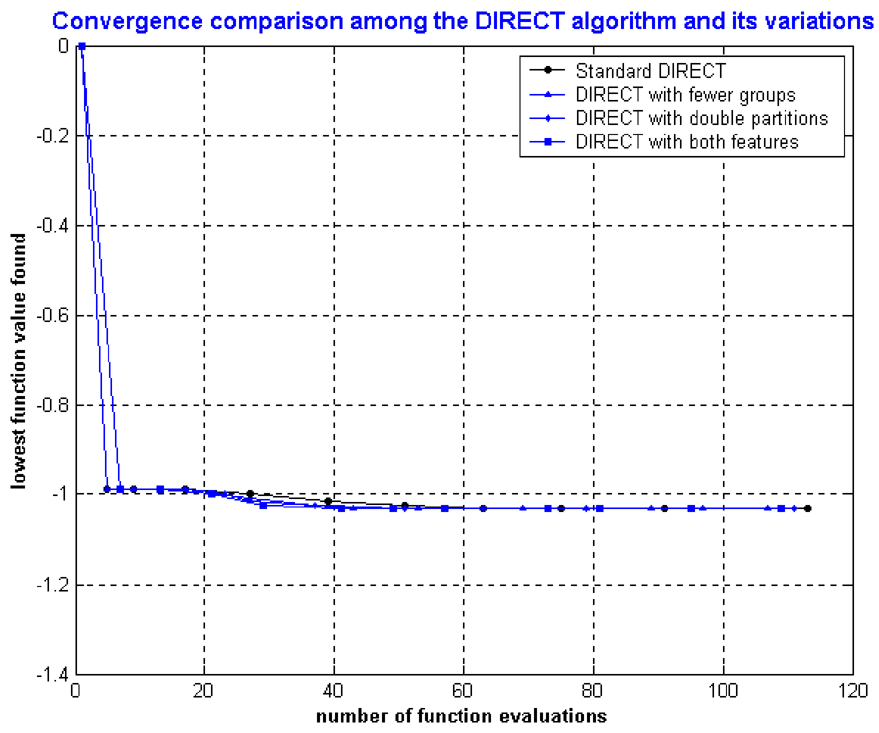
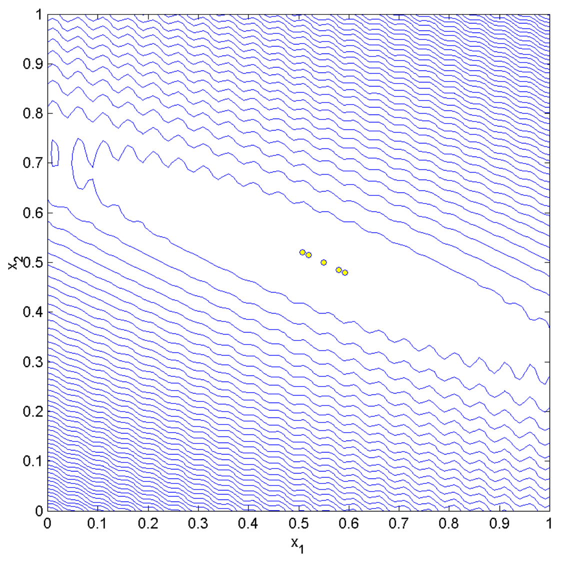


Fig. 4.27 Convergence comparison for the six-hump function



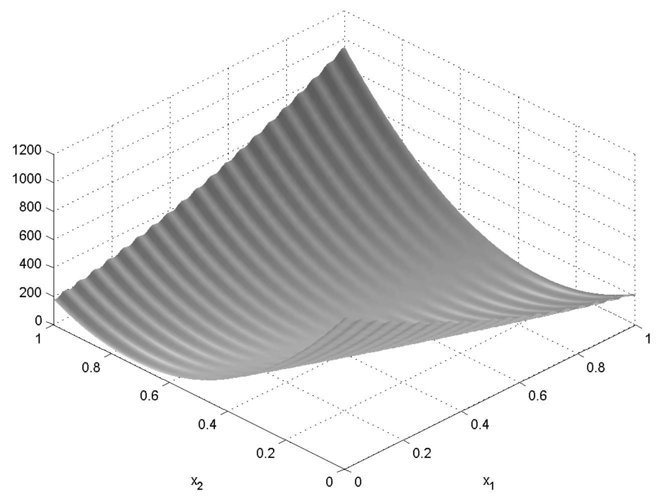


Fig. 4.28 Contour lines

Fig. 4.29 Surface shape

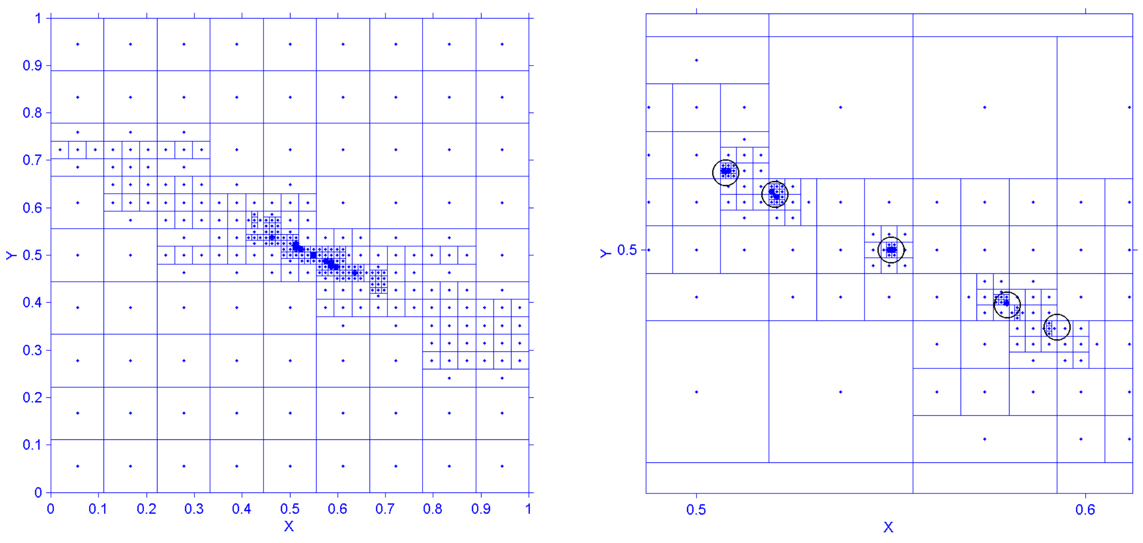


Fig. 4.30 Results of DIRECT

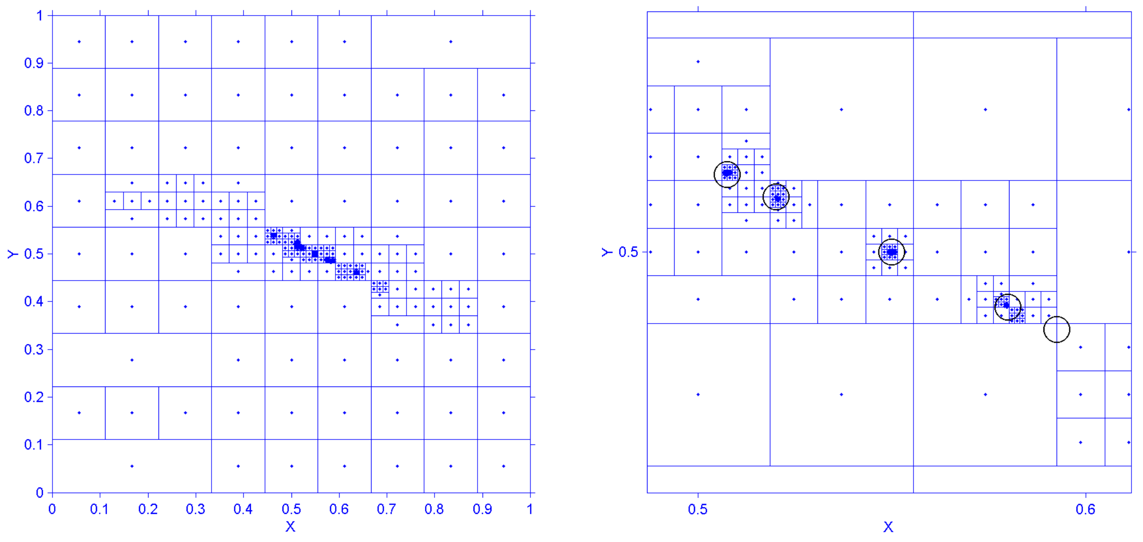


Fig. 4.31 Results of DIRECT-I

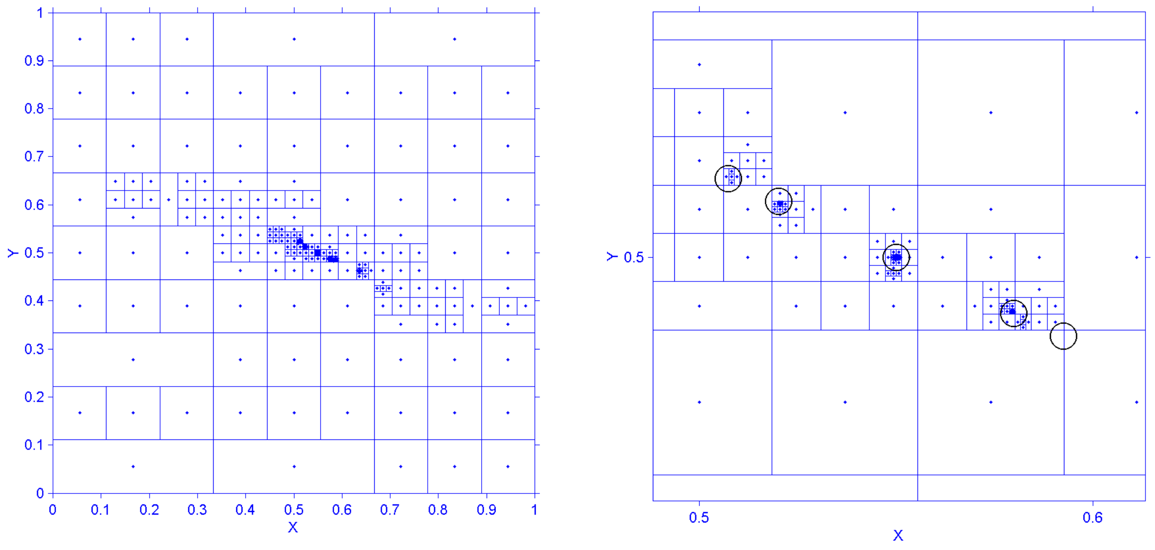


Fig. 4.32 Results of DIRECT-II

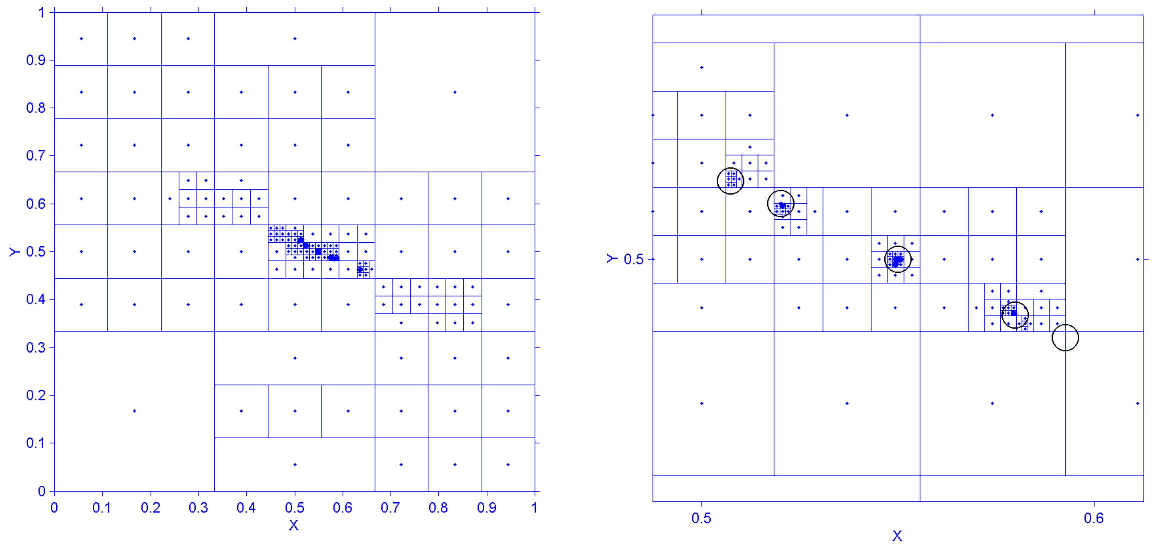


Fig. 4.33 Results of DIRECT-III

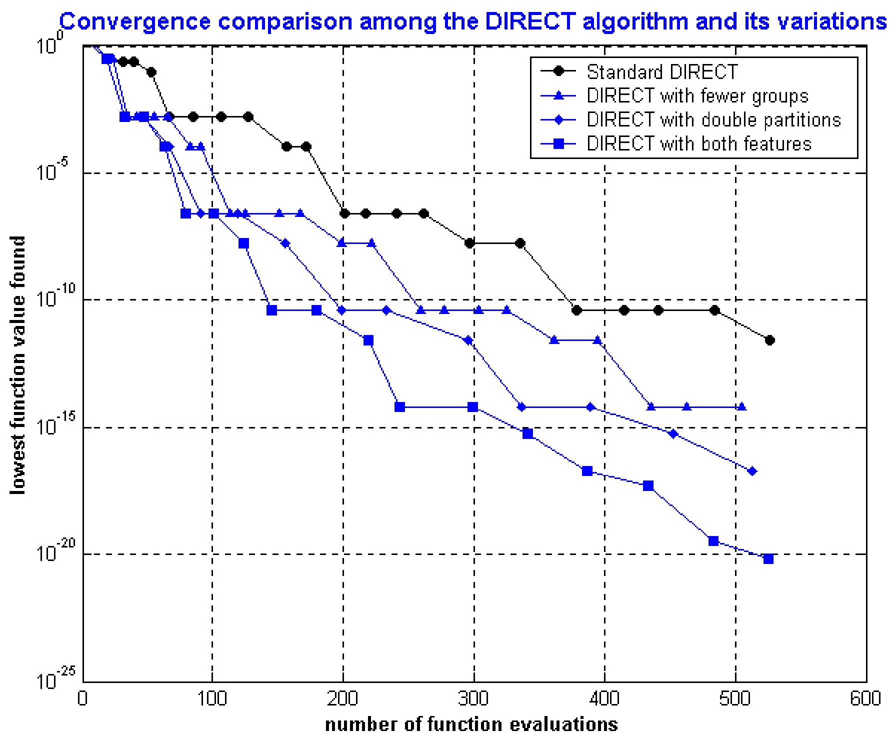


Fig. 4.34 Convergence comparison for the Branin function

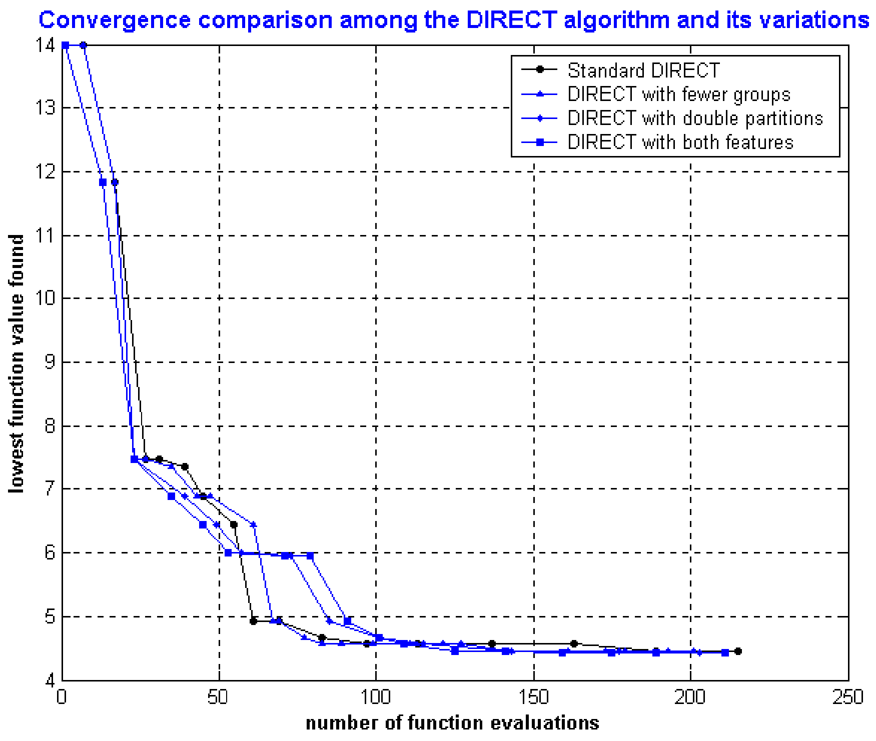


Fig. 4.35 Convergence comparison for the ABS optimization case

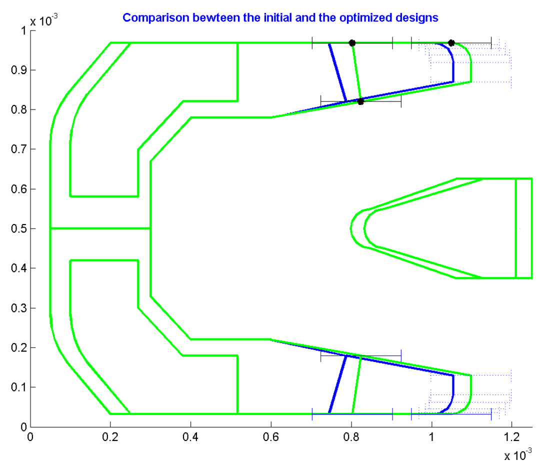


Fig. 4.36 Results of DIRECT

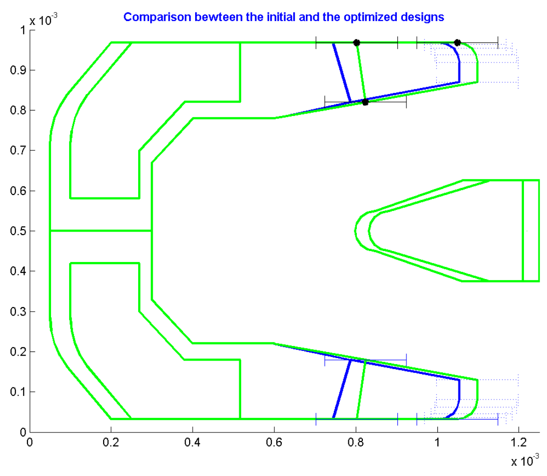


Fig. 4.37 Results of DIRECT-I

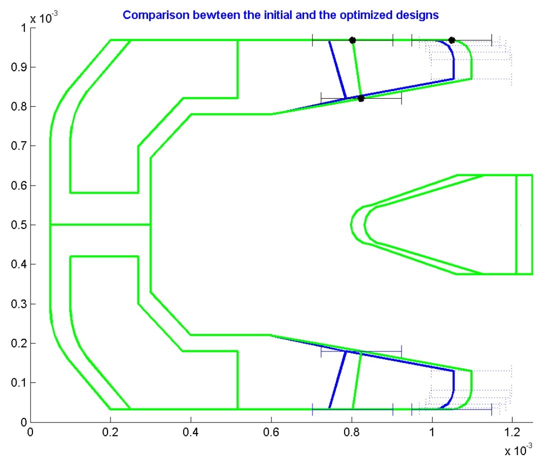


Fig. 4.38 Results of DIRECT-II

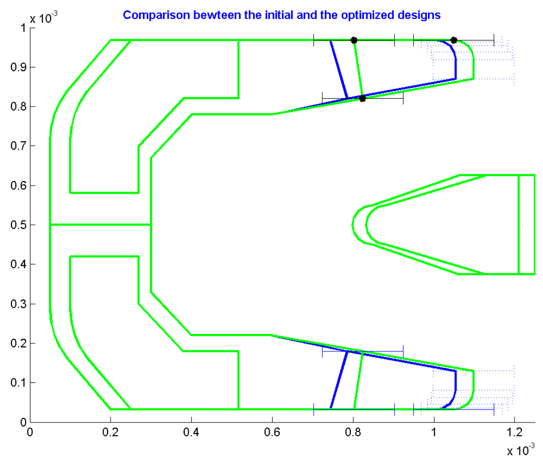


Fig. 4.39 Results of DIRECT-III

Chapter 5

MODIFICATIONS TO THE DIRECT ALGORITHM

5.1 INTRODUCTION

In slider ABS optimization, the evaluation of the ABS designs 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 needed to find the globally optimized ABS design.

In the previous chapter we have presented three locally biased variations of the standard DIRECT algorithm. 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.

In this chapter, we report on two other modifications to the DIRECT algorithm. After presenting the modifications, we discuss some numerical experiments. Then we apply these modifications to slider ABS optimization. Finally, we present results for several slider ABS optimization test cases and draw our conclusions.

5.2 MODIFICATIONS OF THE ALGORITHM

5.2.1 Tolerance

It was shown in Section 3.2.4 that the standard DIRECT algorithm almost always identifies the box containing the minimum point as the potentially optimal box, as illustrated in Fig. 5.1. In fact, in that figure imagine that we draw a line with a positive slope K below all of the data points and then move it upward. If K is positive but small enough, the first data point that the line intersects will be the point with the lowest objective function value. 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 (3.16) 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 expected that the box containing this point will become smaller as the optimization process goes on. This accounts for the fast convergence property of the DIRECT algorithm.

However, in applications to ABS design we do not want the partitioning to continue once the size of the box containing the minimum point shrinks to a certain level. The main reason is that the practical head manufacturing process has a limitation on the processing resolution, and therefore it cannot differentiate between 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 introduce here the concept of "tolerance", that is, the minimal side lengths of all 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 such tolerances can prevent the DIRECT algorithm from wasting valuable function evaluation time in local refinement around the current best point. Therefore, the algorithm can search more globally with a fixed number of function evaluations. Consequently, this modification should improve the efficiency of the standard DIRECT algorithm.

5.2.2 Hidden constraints

The search space of the DIRECT algorithm is a multi-dimensional unit hyper-cube. If no hidden constraints exist in the search space, every sample point can be evaluated, and it yields a definite objective function value. However, if there are hidden constraints in the search space, this is not the case. Let's call the sample points that satisfy the hidden constraints <u>infeasible points</u>. Similarly, the sample points that do not satisfy the hidden constraints are referred as the <u>feasible points</u>.

We want to avoid evaluating the objective function at the infeasible points. However, for the DIRECT algorithm, however, all of the sample points must have values so that the algorithm can find the potentially optimal ones. Therefore we cannot simply discard those infeasible points. The reason is simple: if a midpoint of a large box is an infeasible point, discarding that point and its box will result in the loss of all other possible feasible points within that large box. In order to handle these hidden constraints, we use the method proposed by Gablonsky (2001), which provides a pseudo-value for an infeasible point depending on its neighboring points' values.

Figure 5.2 illustrates how we deal with the infeasible points. The square area is the search space for a 2-D problem. The hollow round dots with x's represent infeasible points, i.e., points which satisfy the hidden constraints. The solid round dots represent feasible points. All of the feasible points have calculated 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 an infeasible point. (Two such double size boxes are shown as the shadow boxes with dashed sidelines in Fig. 5.2.)

2. Check the status of all 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 of the sample points inside that enlarged box are infeasible points (for example, the case of the infeasible point in the square box in Fig. 5.2), then that infeasible point is marked as a "real" infeasible point and is given the pseudo-value

of f_{max} + 1, where f_{max} is the current maximum objective function value of all feasible points.

4. If the sample points inside that enlarged box are not all infeasible points, (for example, the case of the infeasible point in the rectangular box in Fig. 5.2), then the infeasible point is given the 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 choose 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 the boxes of the same size containing "real" infeasible points will get partitioned last, because they have the highest (pseudo) objective function value. This is reasonable 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, even the boxes containing the infeasible points near some feasible 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.

5.3 NUMERICAL EXPERIMENTS

5.3.1 Test function with one minimum point

For simple demonstrations of the numerical experimental results we chose 2-D test functions. The test 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 5.3 shows contour lines of this 2-D function. Figure 5.4 shows its surface shape.

Figures 5.5 ~ 5.8 show the optimization results after 500 function evaluations. In Fig. 5.5 tolerance is not used, whereas in Fig. 5.7 the tolerance is set at 0.01 for both of the independent variables x_1 and x_2. Figures 5.6 and 5.8 show the local zoom-ins around the best point for Figs. 5.5 and 5.7, respectively.

In Figs. 5.5 ~ 5.8, X and Y represent the variables x_1 and x_2, respectively. The small dots inside the boxes are the sample points generated by the DIRECT algorithm. The circle represents the minimum point found by the algorithm.

For the non-tolerance case, the minimum 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 5.8 shows that when both sides of a box are smaller than the prescribed tolerance, 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 this 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. 5.9. It is clear from this figure 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 the side x_1. Because the box containing the best point has reached the tolerance limit, it will not be further partitioned in the subsequent optimization process, and because the test function has only 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 search space. The sample points that satisfy the hidden constraints are referred as the infeasible points, and are not evaluated. The tolerance is set at 0.01 for both x_1 and x_2. The number of function evaluations is 3000. Figure 5.10 shows the optimization results without hidden constraints. Figures 5.12, 5.14, and 5.16 show the optimization results with various hidden constraints. Figures 5.11, 5.13, 5.15 and 5.17 show local zoom-ins around the best points for Figs. 5.10, 5.12, 5.14, and 5.16, respectively.

The hidden constraints for Fig. 5.12 are:

$(x_1 - 0.35)^2 + (x_2 - 0.35)^2 \leq 0.15^2$ and $x_2 \leq x_1^2$.

The hidden constraint for Fig. 5.14 is:

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

The hidden constraint for Fig. 5.16 is:

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

In Fig. 5.12 the hidden constraint areas are the areas inside the circle and the area below the parabolic line. In Fig. 5.14 the hidden constraint area is the area between the two concentric circles. In Fig. 5.16 the hidden constraint area is inside the circle that centers at (0.4, 0.2). The shadowed boxes are the ones containing infeasible points.

Figures 5.12 ~ 5.15 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. 5.16. 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.

5.3.2 Test functions with multiple minimum points

The first test 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. 4.21 and 4.22, respectively. The round dots in Fig. 4.21 represent the global minimum points.

Figures 5.18 and 5.19 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. 5.18 and 5.19 that when the tolerance is imposed, the algorithm is biased toward a global search for a fixed number of function evaluations. Figures 5.18 and 5.19 clearly show that the algorithm explores more "bigger boxes" with tolerance included than it does without tolerance.

Figure 5.20 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:

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

In Fig. 5.20 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. 5.20 also shows that, inside the hidden constraint areas, the regions that are adjacent to the minimum points get partitioned more often.

The second test 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. 4.28 and 4.29, respectively. The five round dots in Fig. 4.28 represent the global minimum points.

Figures 5.21 ~ 5.24 show the optimization results for different cases of the Branin function after 400 function evaluations. The tiny dots in Figs. 5.21a ~ 5.24a (left) represent the sample points. Figures 5.21b ~ 5.28b (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 5.21 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 5.22 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. Figure 5.22 shows that DIRECT-III found only 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. 5.21 and 5.22 are quite different. There are 3 large unexplored boxes in Fig. 5.22, another verification of the locally biased property of DIRECT-III.

Figures 5.23 and 5.24 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 5.23 shows that the DIRECT algorithm found all 5 global minima for this case.

The interesting thing about Fig. 5.24 is that DIRECT-III also found all 5 global minima, with a pattern similar to Figs. 5.21 and 5.23. This is because, to some extent, the globally biased property of defining tolerances balances the locally biased property of DIRECT-III.

5.4 SLIDER AIR BEARING DESIGN OPTIMIZATION CASE

5.4.1 Air bearing design optimization problem

We choose the same 7nm FH Pico slider defined in Section 2.4 as the prototype slider. Its rail shape and the 3-dimensional rail geometry are shown in Figs. 2.10 and 2.11, respectively. The optimization goals are also the same, i.e., to lower its flying heights to the target flying height, i.e. 5nm and at the same time maintain a uniform roll profile across the three different radial positions OD, MD and ID. We also define the same objective function as in Section 2.4. For easier demonstration of the optimization results, only two original constraint points are defined, which are shown in Fig. 5.25.

5.4.2 Some special features of slider ABS optimization

For slider ABS optimization problems, the tolerance mentioned in Section 5.2.1 is referred to as the manufacturing tolerance. The actual slider ABS fabricating process determines the magnitude of the manufacturing tolerance, which is generally dependent on the manufacturer.

In the previous section 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"). 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 disk radial position, but we evaluate it at multiple radial positions, such as the OD (Outer Diameter), MD (Middle Diameter) and ID (Inner Diameter). If a sample ABS design satisfies the prescribed hidden constraints, then no further calculation is necessary. So if the hidden constraints are satisfied at any one 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 most partially calculated.

5.4.3 Optimization 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 the optimization was performed without defining any manufacturing tolerance or hidden constraints. In this case, all the sample points generated by the algorithm were fully evaluated and have definite objective function values. Figure 5.26 shows the optimization results. The horizontal axis X and the vertical axis Y represent the 1st and 2nd constraint points defined in Fig. 5.25, respectively.

Figure 5.27 shows the contour lines in the search space drawn from the results in Fig. 5.26. The circular dot represents the best point. The contour map gives 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. 5.26 and 5.27, 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 a manufacturing tolerance and 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 were defined as:

FH ≤ 2 nm or FH ≥ 10 nm or Roll ≤ -30 μrad or Roll ≥ 30 μrad.

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

FH ≤ 4 nm or FH ≥ 6 nm or Roll ≤ -10 μrad or Roll ≥ 10 μrad.

Figures 5.28 and 5.29 show the optimization results for case 2 and case 3, respectively. In Figs. 5.26, 5.28 and 5.29 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. 5.28 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 it will increase the total areas of the rails. Thus, it will result in higher FHs, which is contrary to the optimization goal of lowering the FHs.

Figures 5.30 ~ 5.32 show the variations of the objective function values 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 $\text{Percent}_{\text{imp}}$ 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 the 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 5.30 ~ 5.32 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 5.33 shows a comparison of the initial ABS design (gray) and the optimized ABS design (dark). Table 5.1 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 5.34, 5.36 and 5.38 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 5.35, 5.37 and 5.39 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.

5.5 CONCLUSION

In this chapter we discussed two modifications to the DIRECT algorithm: one to handle 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 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 a manufacturing tolerance and loose hidden constraints were defined; and case 3, in which a 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 improve its efficiency and make it more flexible.

	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 5.1 Summary of the optimization results

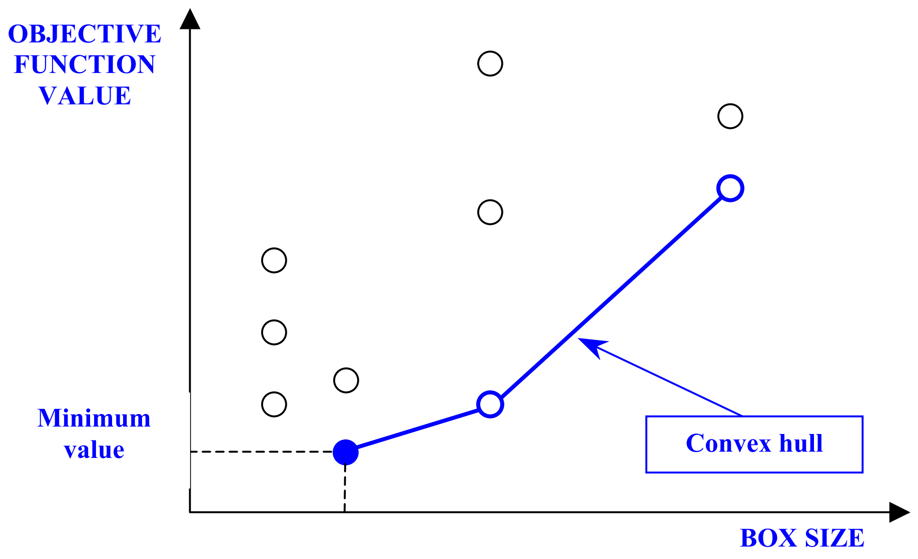
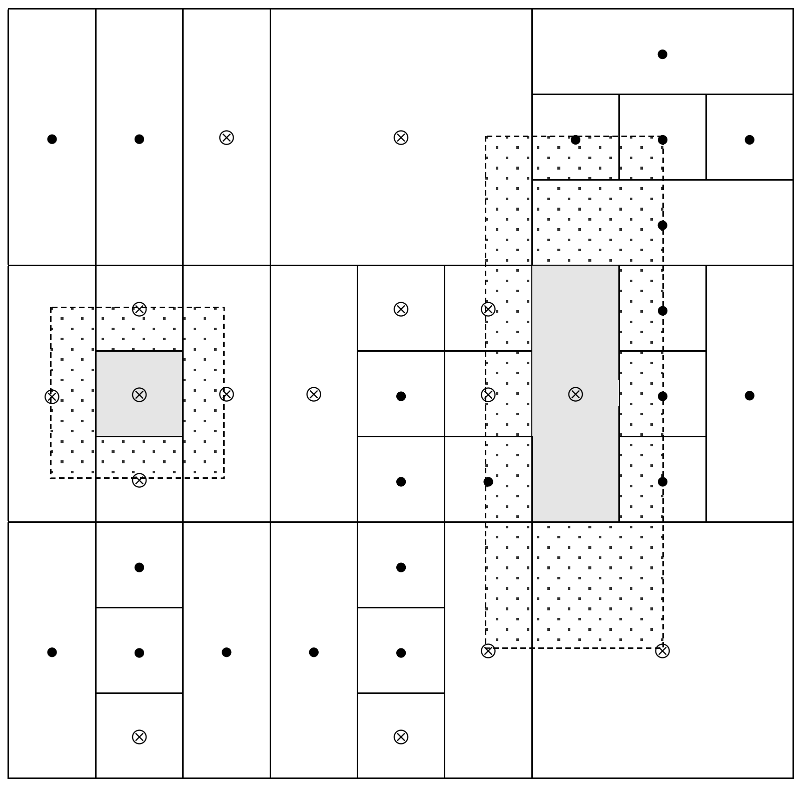
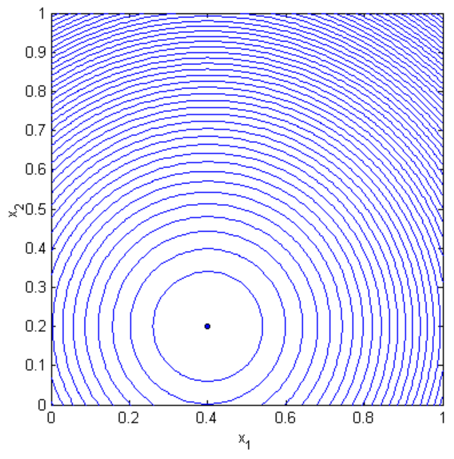


Fig. 5.1 Illustration of the convex hull and the potentially optimal boxes



● Feasible point ⊗ Infeasible point

Fig. 5.2 Illustration of the strategy to handle hidden constraints



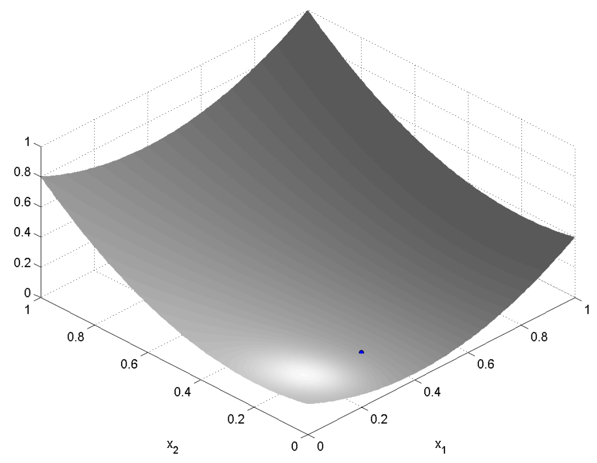


Fig. 5.3 Contour lines

Fig. 5.4 Surface shape

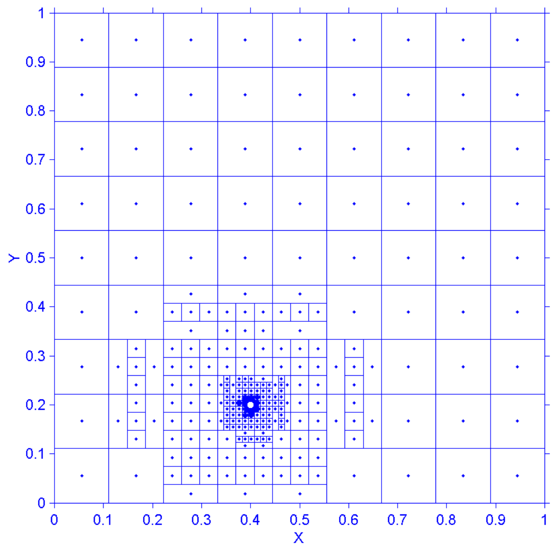


Fig. 5.5 Results with no-tolerance case

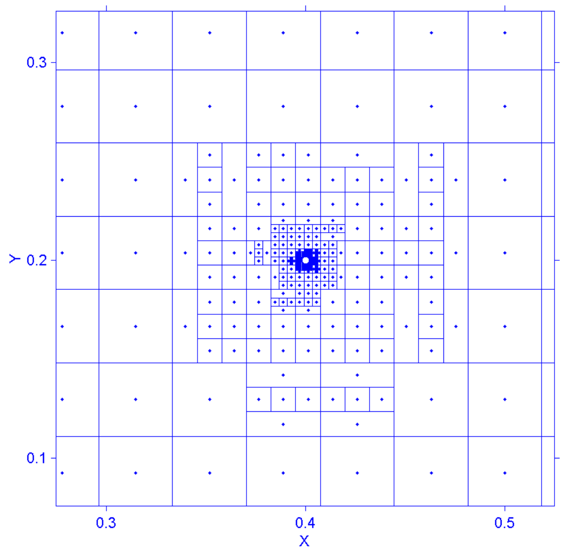


Fig. 5.6 Local zoom-in

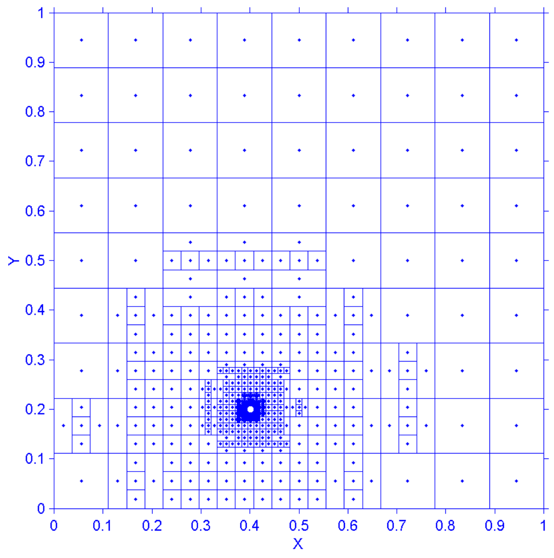


Fig. 5.7 Results with tolerance case

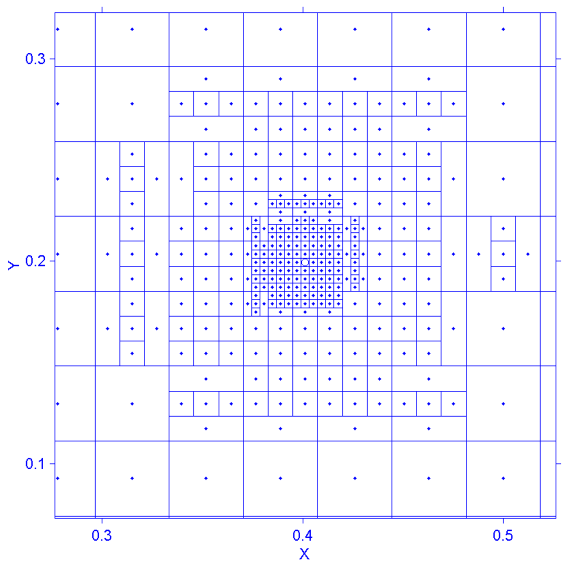


Fig. 5.8 Local zoom-in

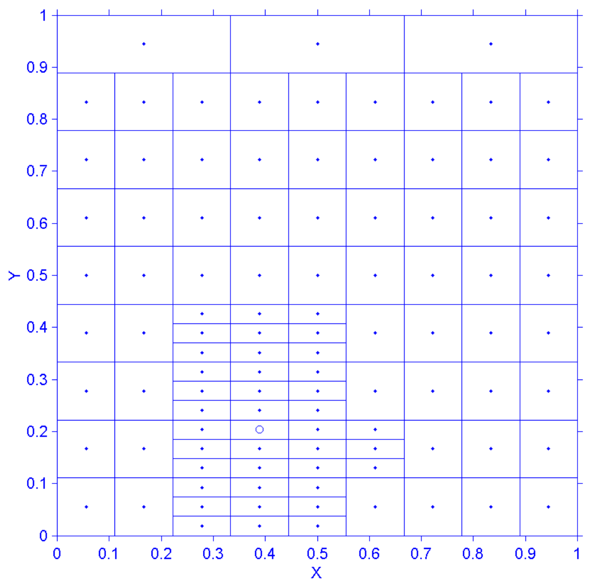


Fig. 5.9 Optimization results for different tolerance values

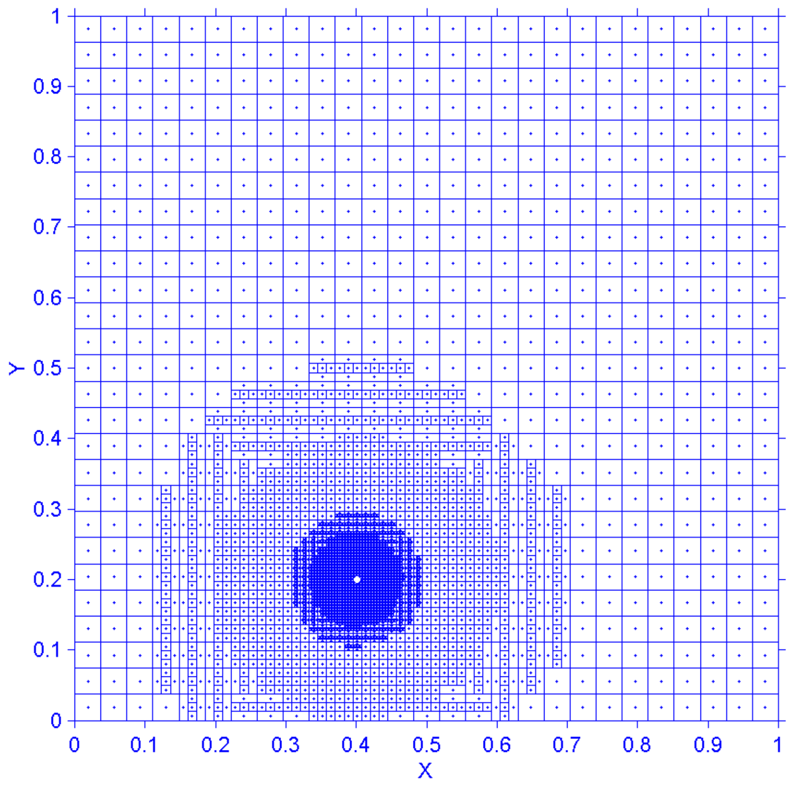


Fig. 5.10 Optimization results without hidden constraints

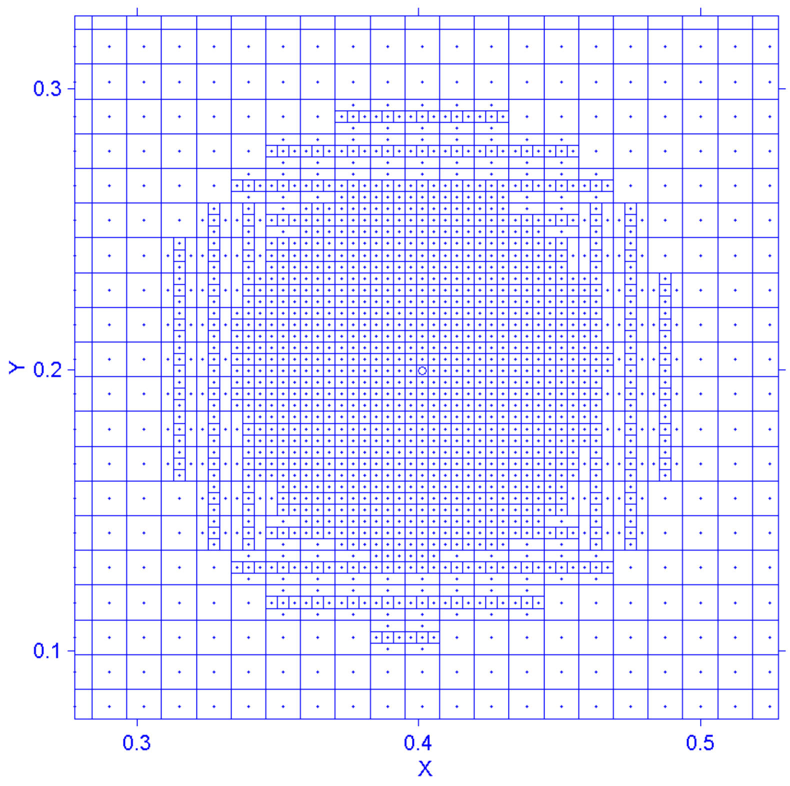


Fig. 5.11 Local zoom-in around the best point without hidden constraints

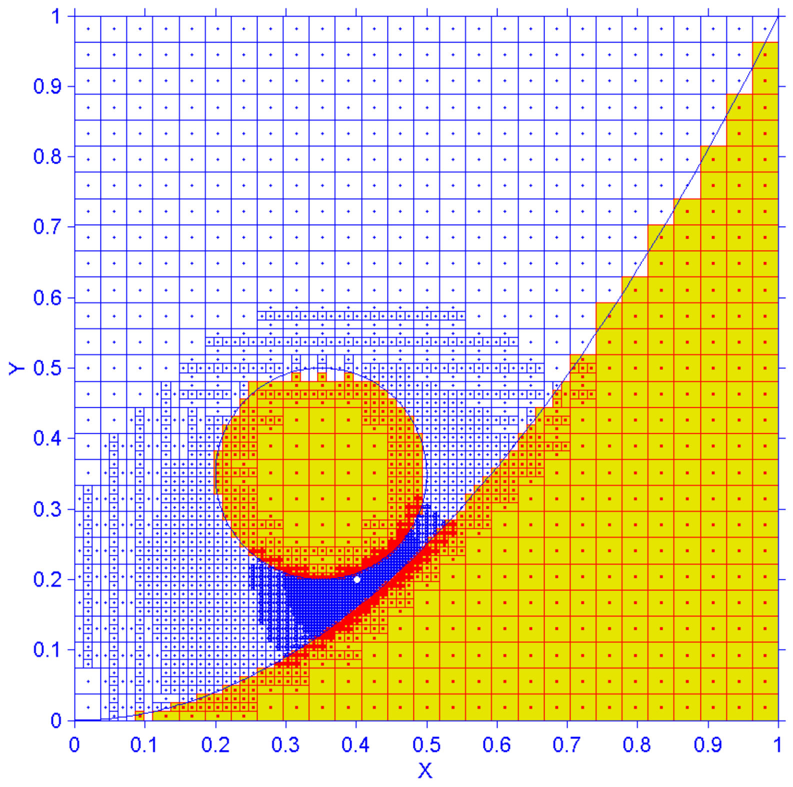


Fig. 5.12 Optimization results with hidden constraints, case 1

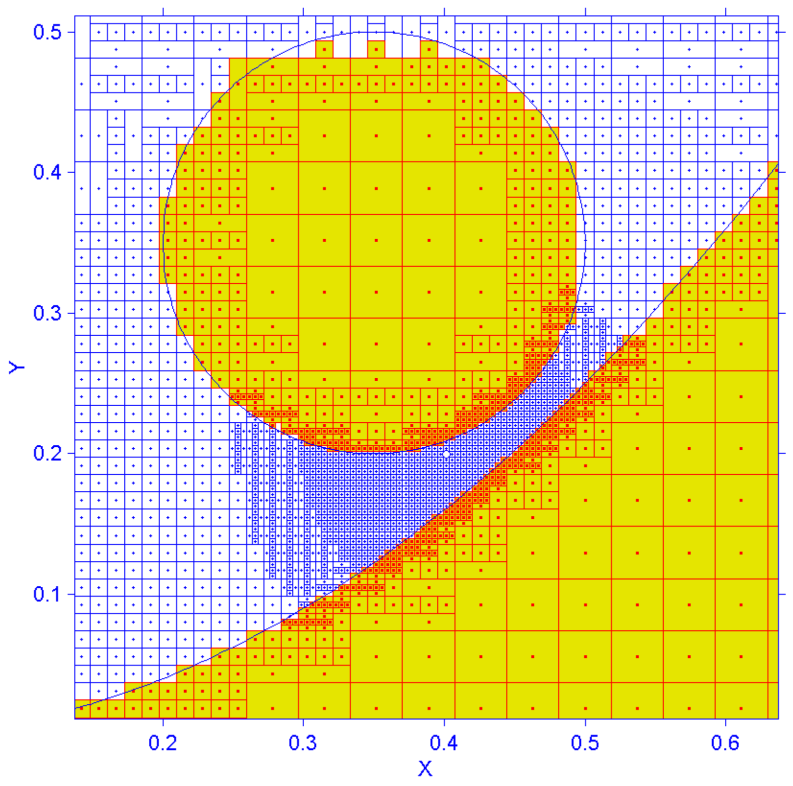


Fig. 5.13 Local zoom-in around the best point for case 1

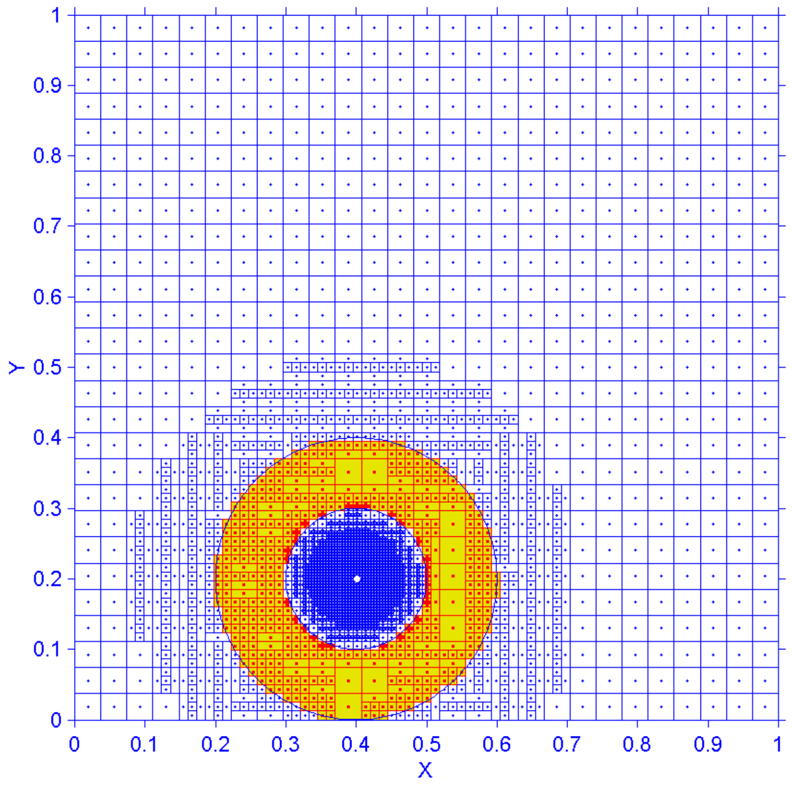


Fig. 5.14 Optimization results with hidden constraints, case 2

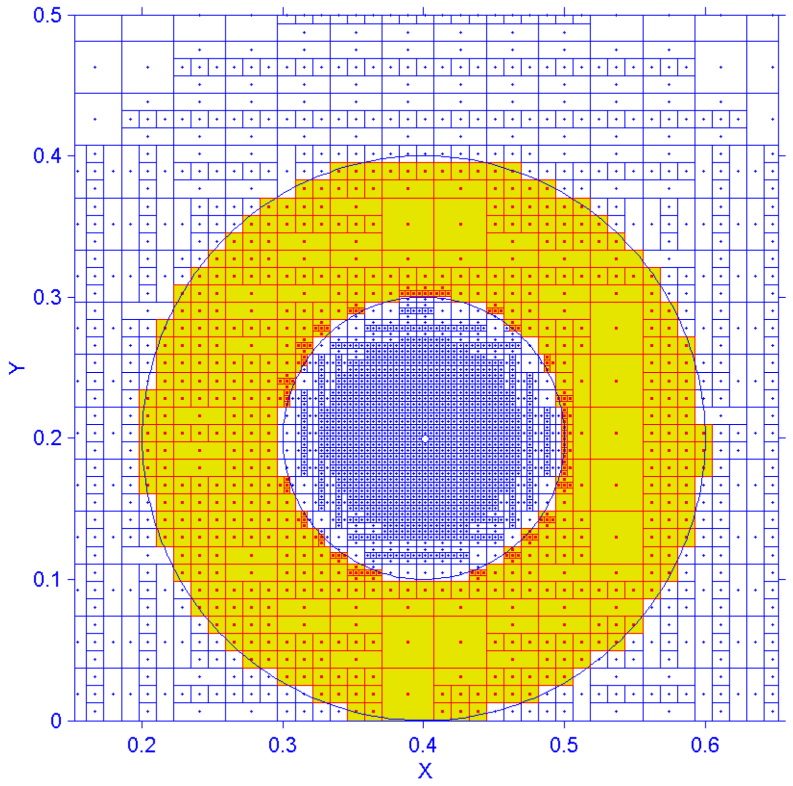


Fig. 5.15 Local zoom-in around the best point for case 2

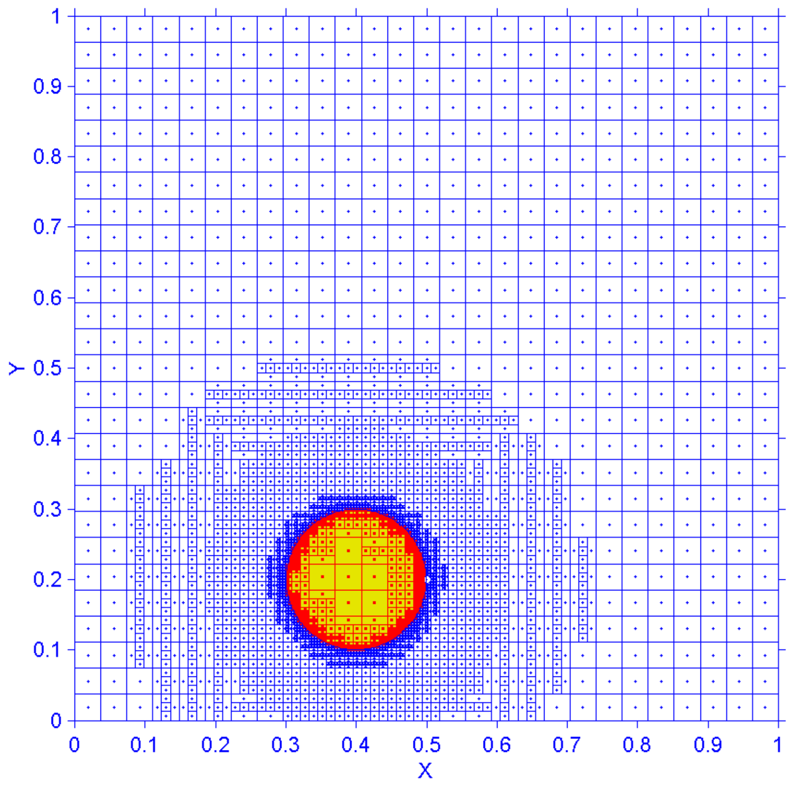


Fig. 5.16 Optimization results with hidden constraints, case 3

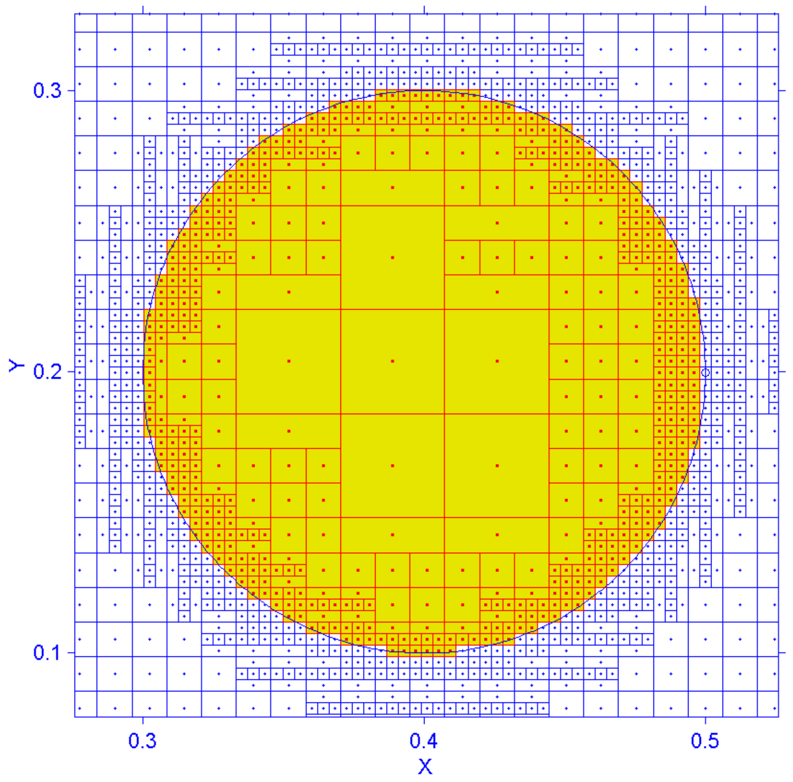


Fig. 5.17 Local zoom-in around the best point for case 3

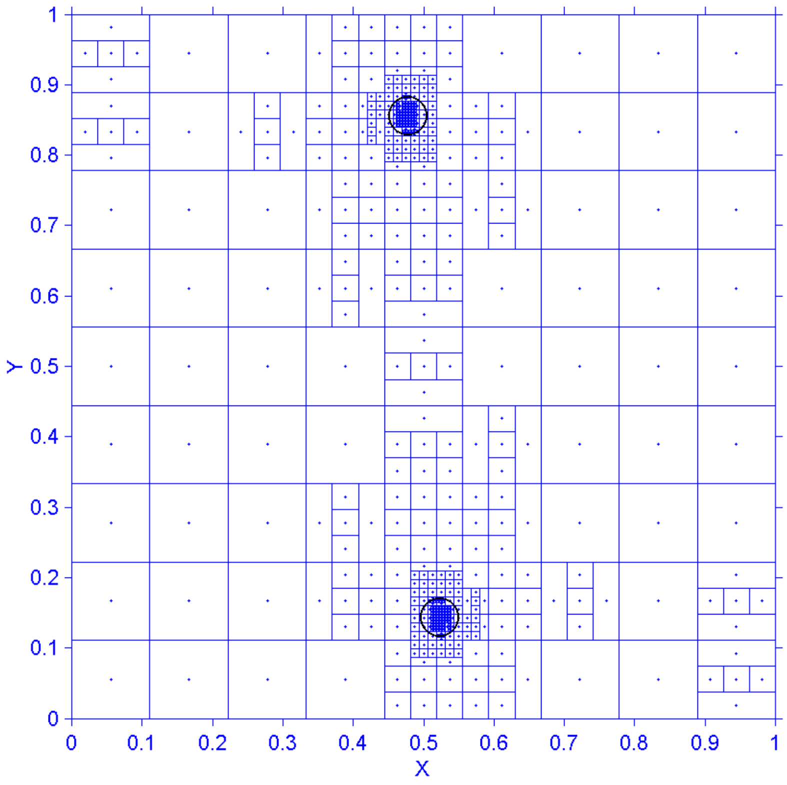


Fig. 5.18 Optimization results with no tolerance

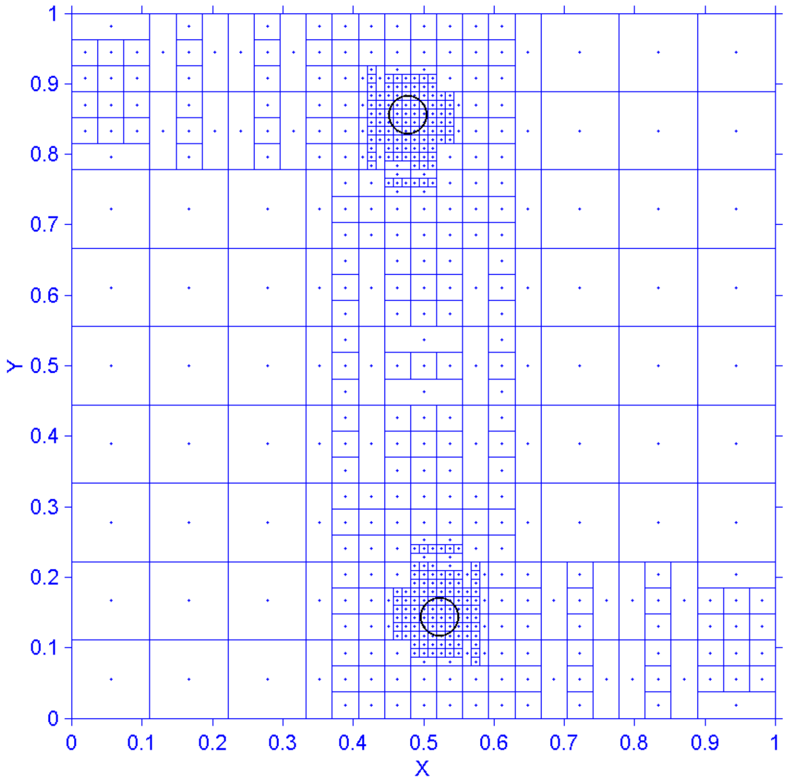


Fig. 5.19 Optimization results with 0.02 tolerance

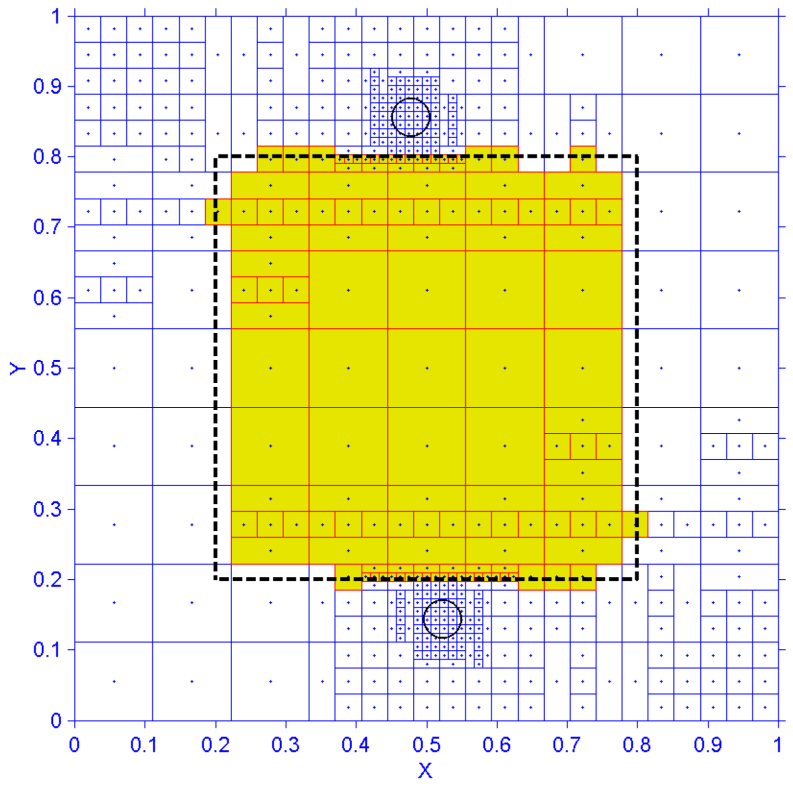
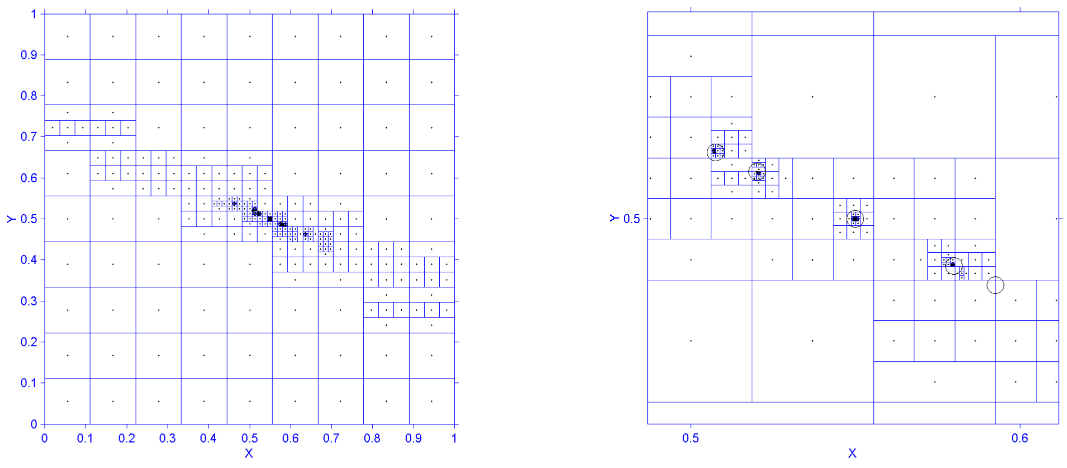


Fig. 5.20 Optimization results with hidden constraints



a **b**

Fig. 5.21 Results of standard DIRECT without tolerance

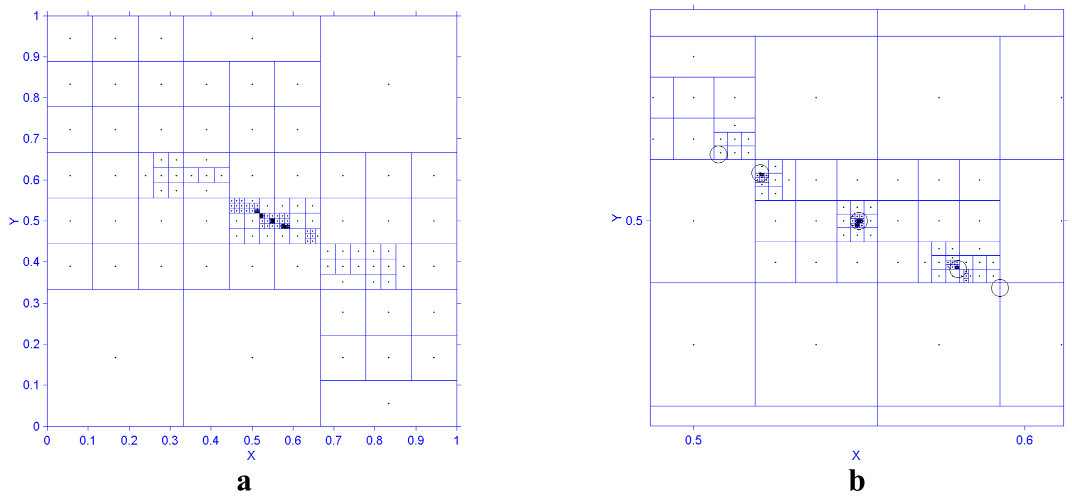


Fig. 5.22 Results of DIRECT-III without tolerance

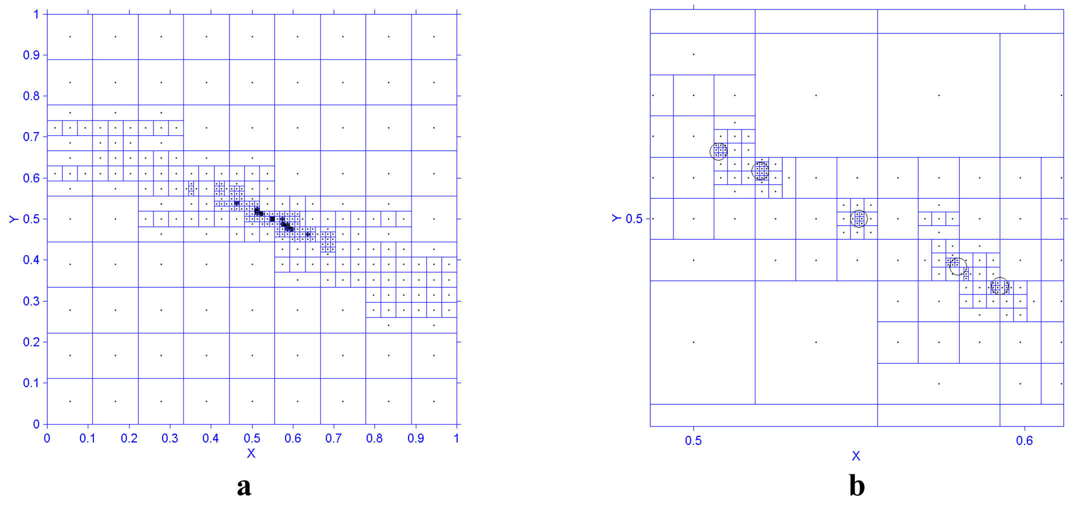


Fig. 5.23 Results of standard DIRECT with 0.002 tolerances

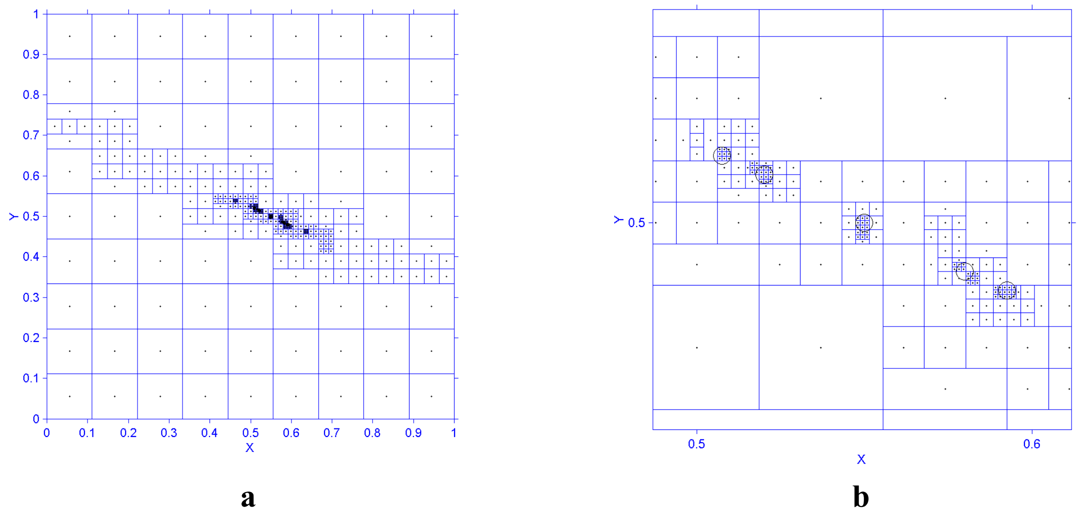


Fig. 5.24 Results of DIRECT-III with 0.002 tolerances

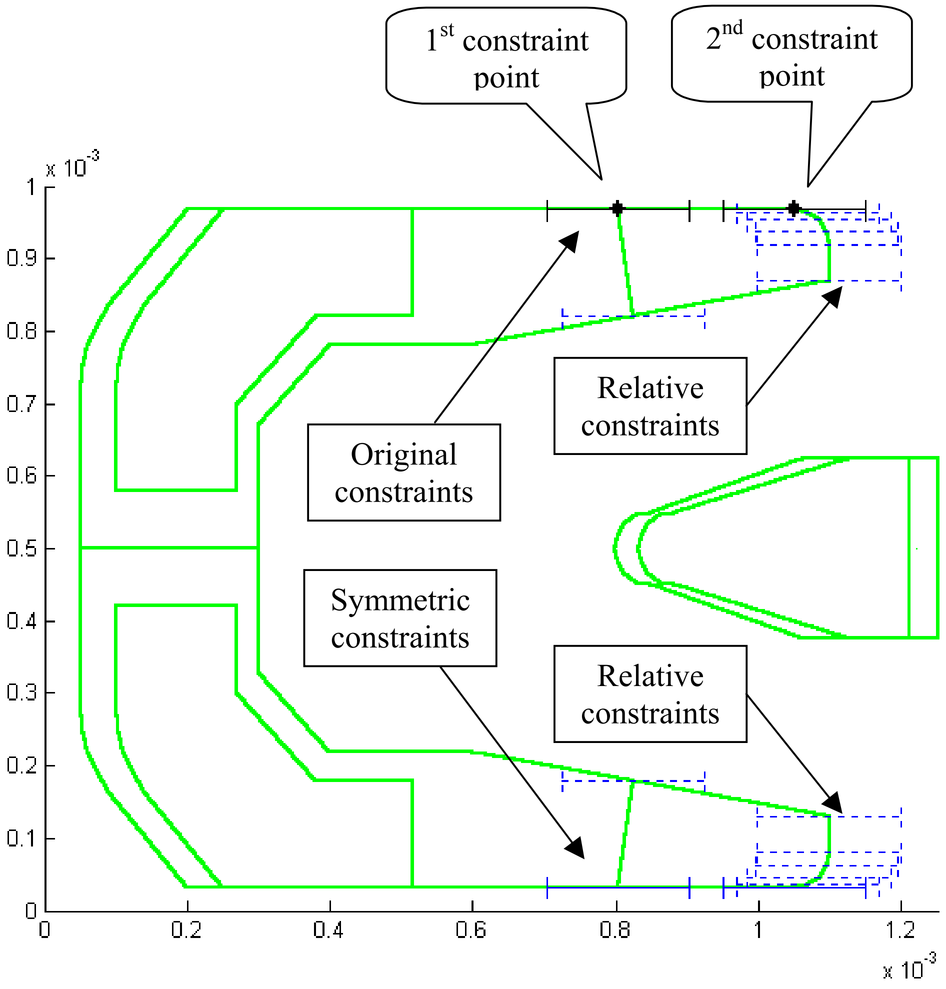


Fig. 5.25 Constraints defined on the initial design

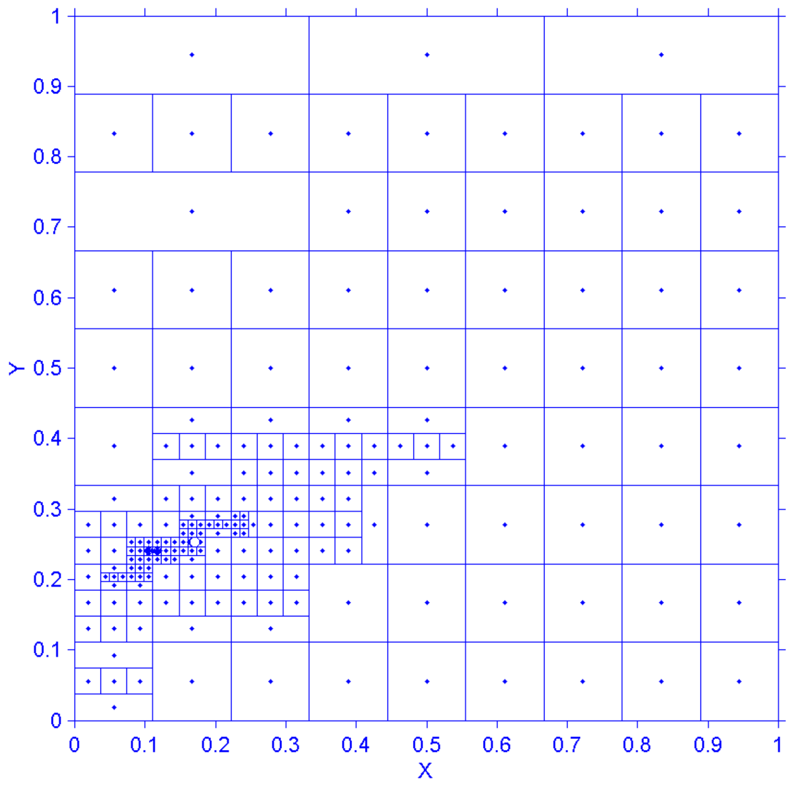


Fig. 5.26 Results without manufacturing tolerance or hidden constraints

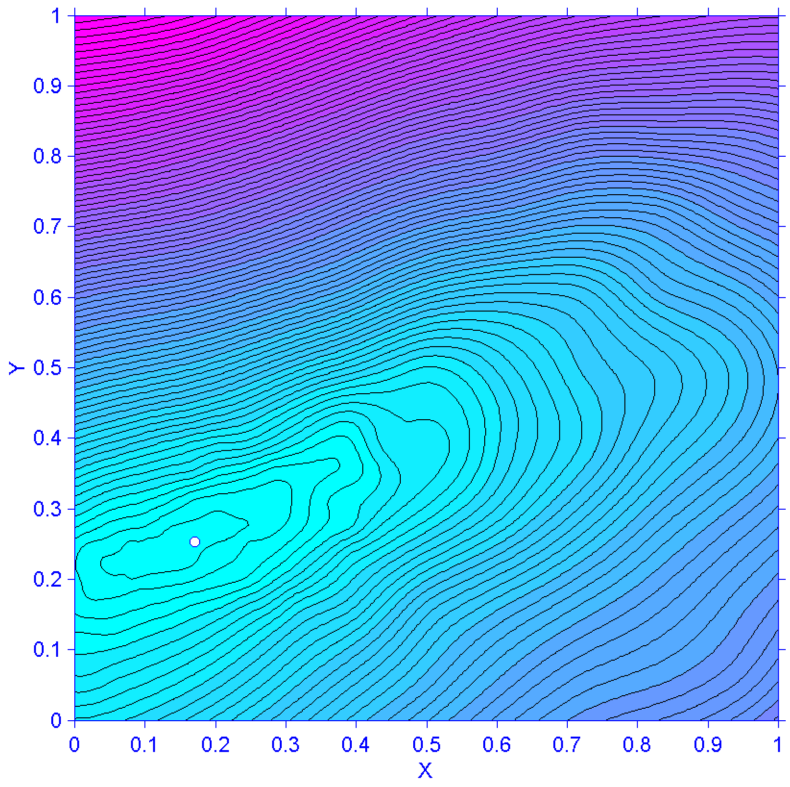


Fig. 5.27 Contour lines in the search space

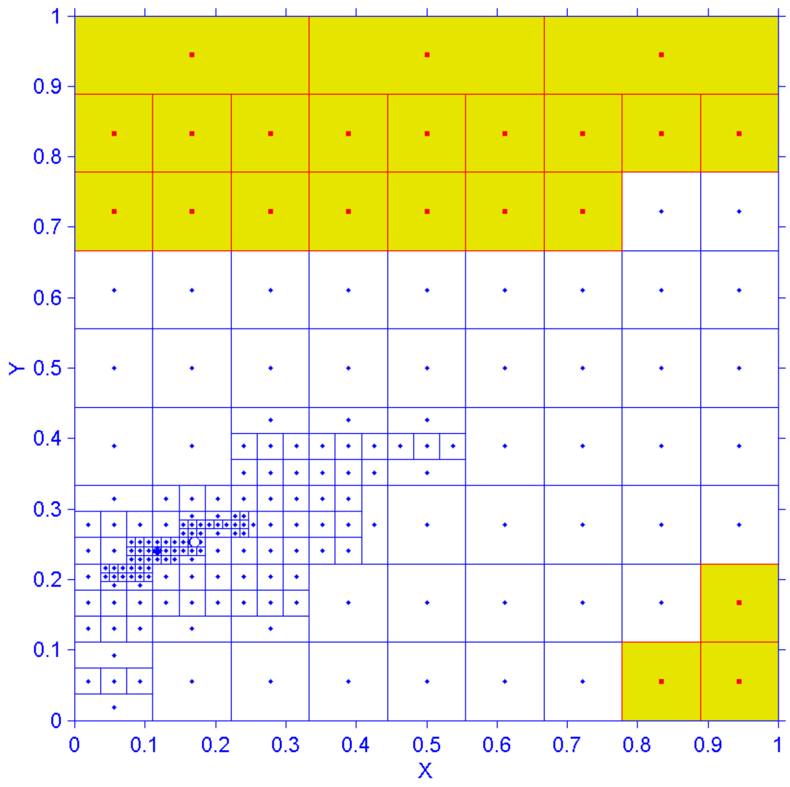


Fig. 5.28 Results with manufacturing tolerance and loose hidden constraints

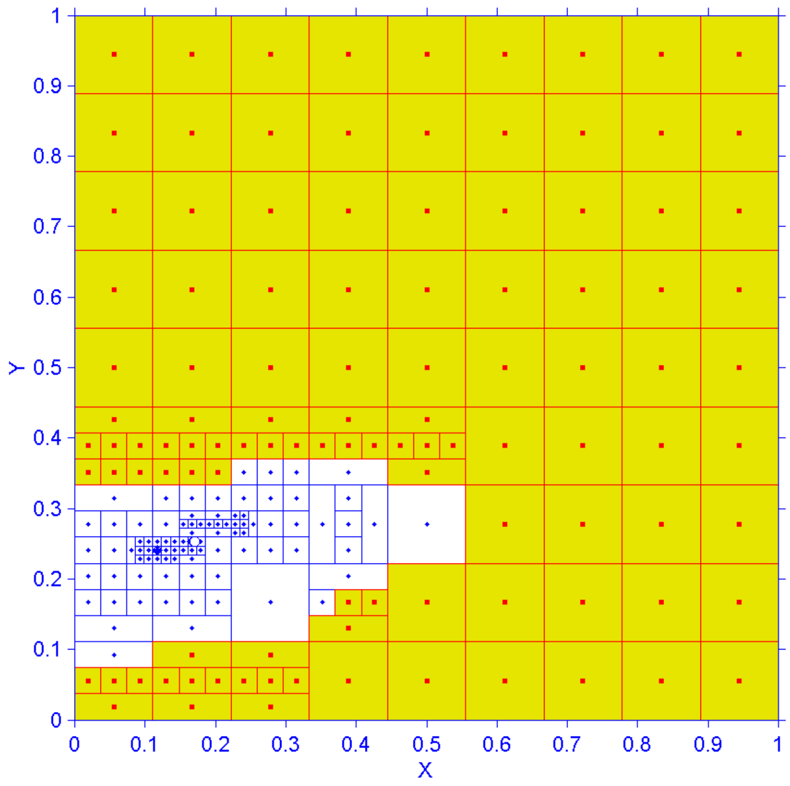


Fig. 5.29 Results with manufacturing tolerance and strict hidden constraints

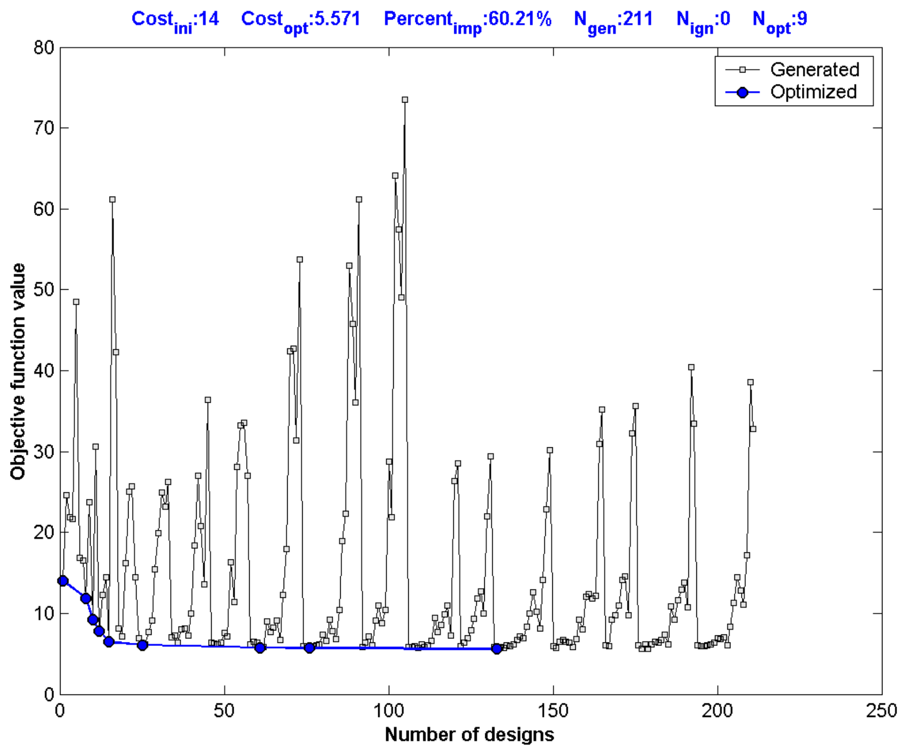


Fig. 5.30 Variation of the objective function value for case 1

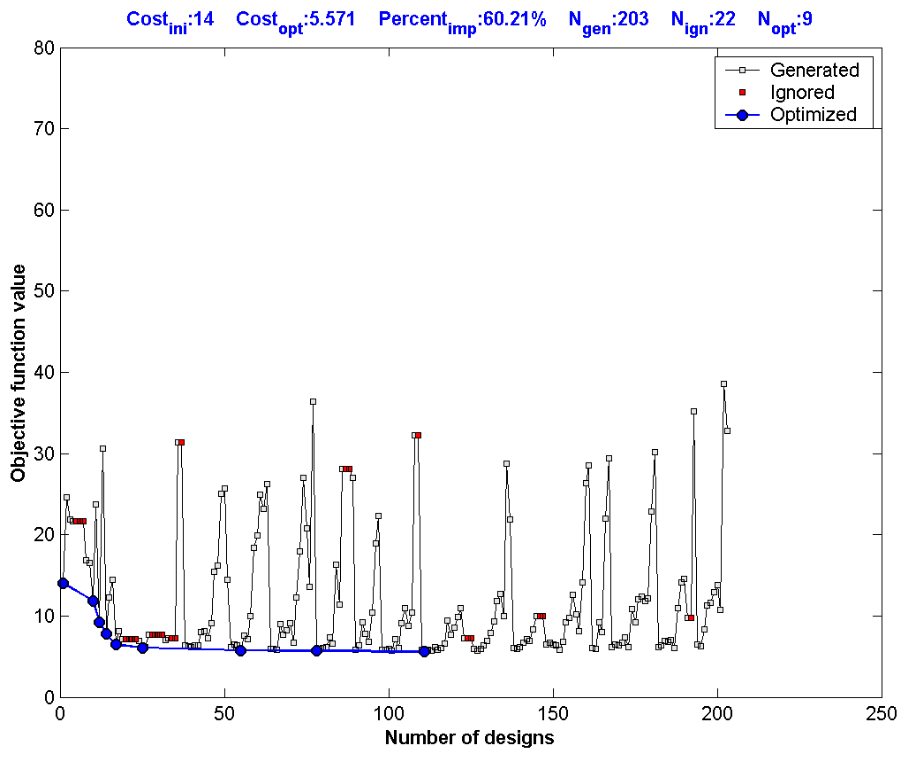


Fig. 5.31 Variation of the objective function value for case 2

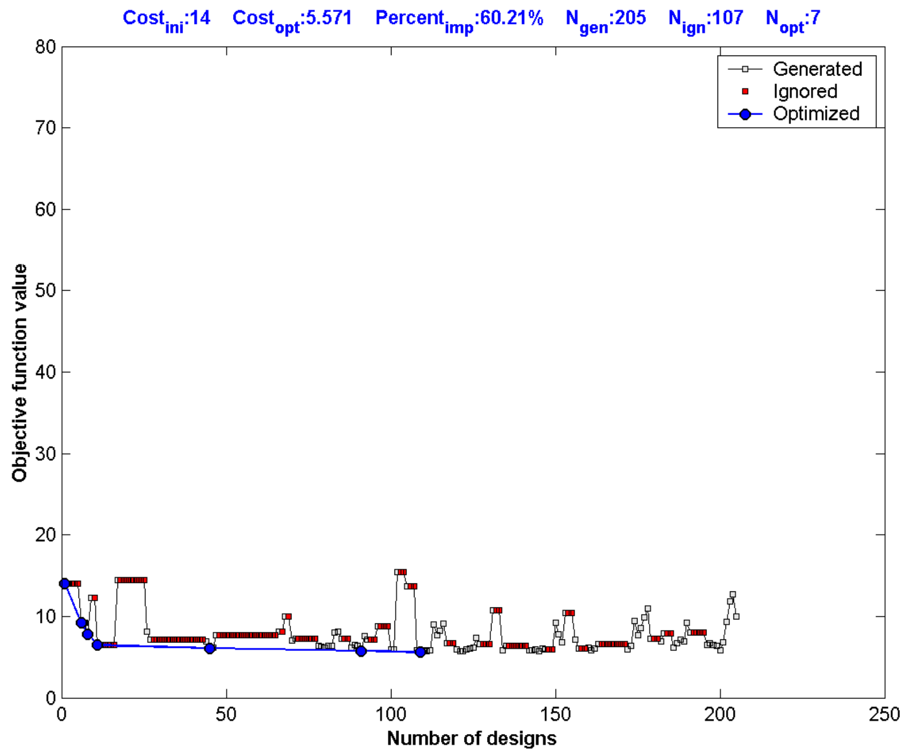


Fig. 5.32 Variation of the objective function value for case 3

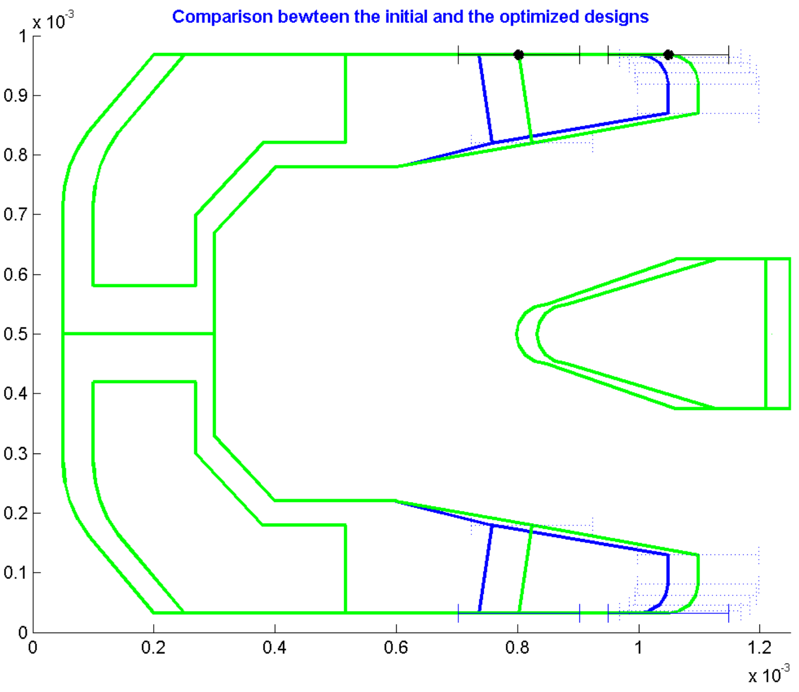


Fig. 5.33 Comparison of the initial design and the optimized design

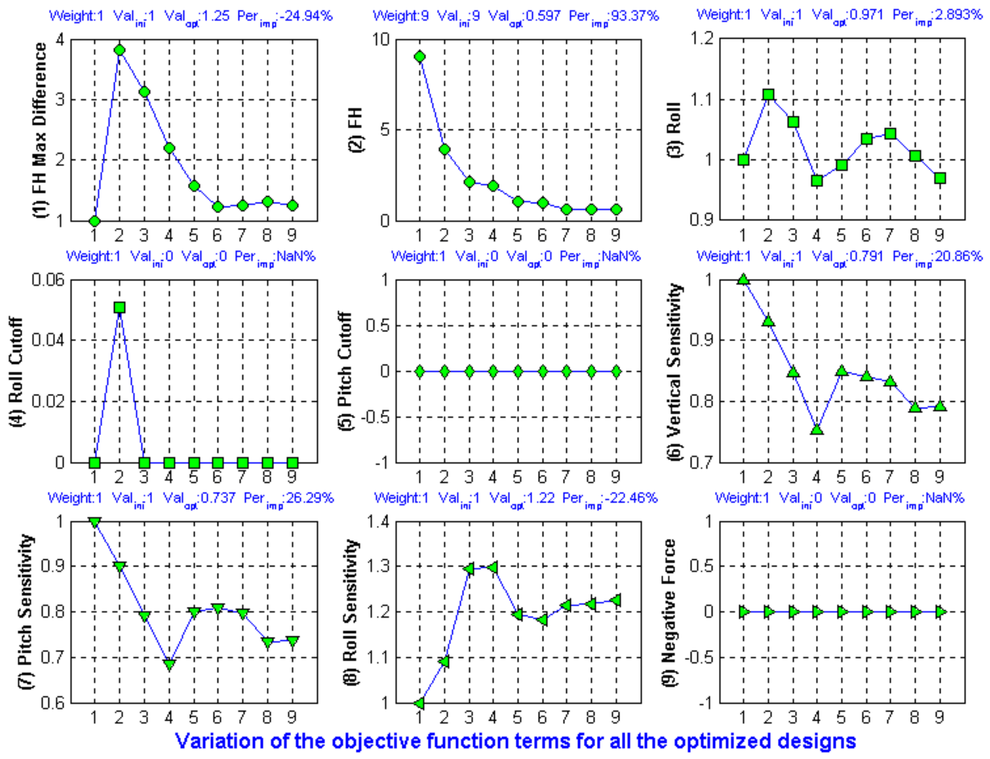


Fig. 5.34 Variations of the objective function terms (case 1)

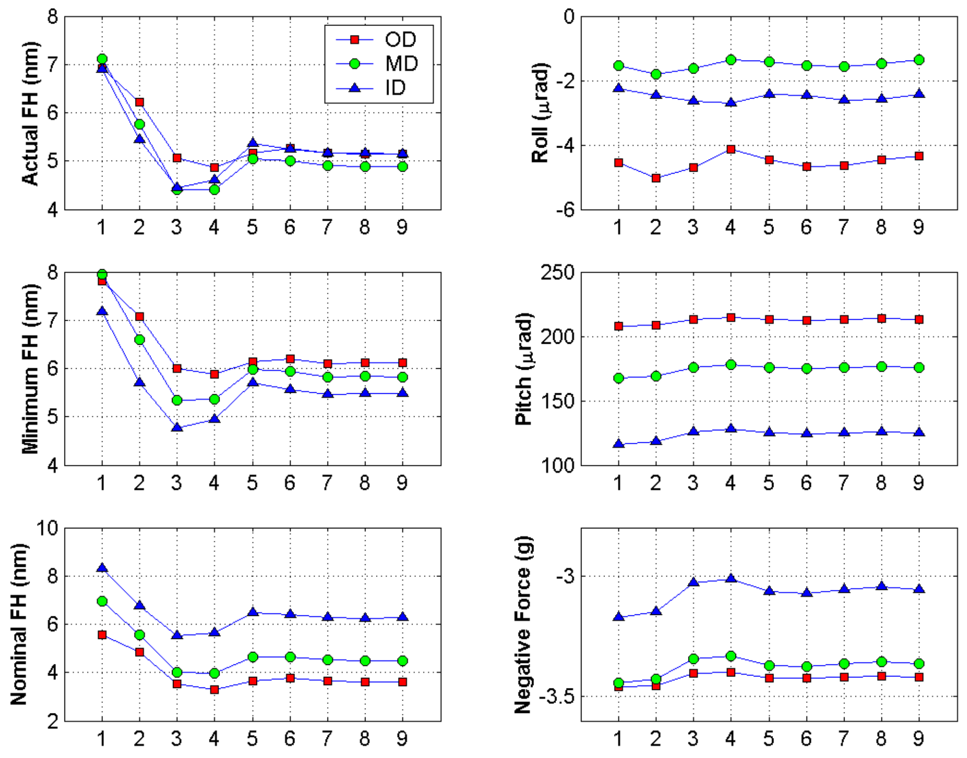


Fig. 5.35 Variations of the slider performance parameters (case 1)

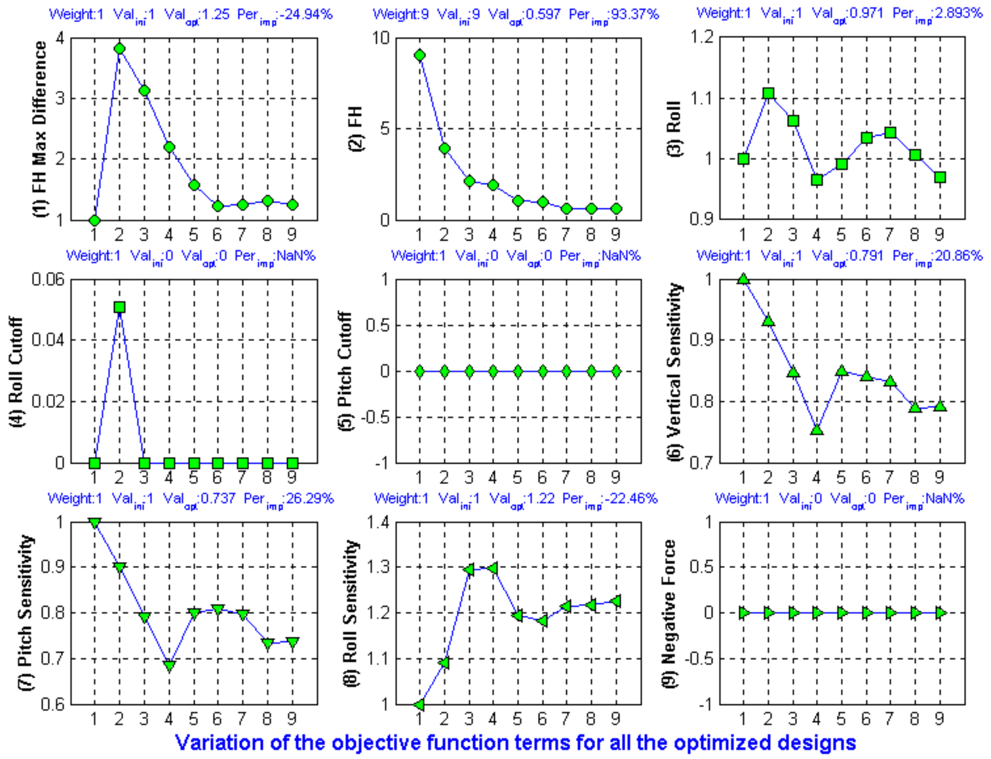


Fig. 5.36 Variations of the objective function terms (case 2)

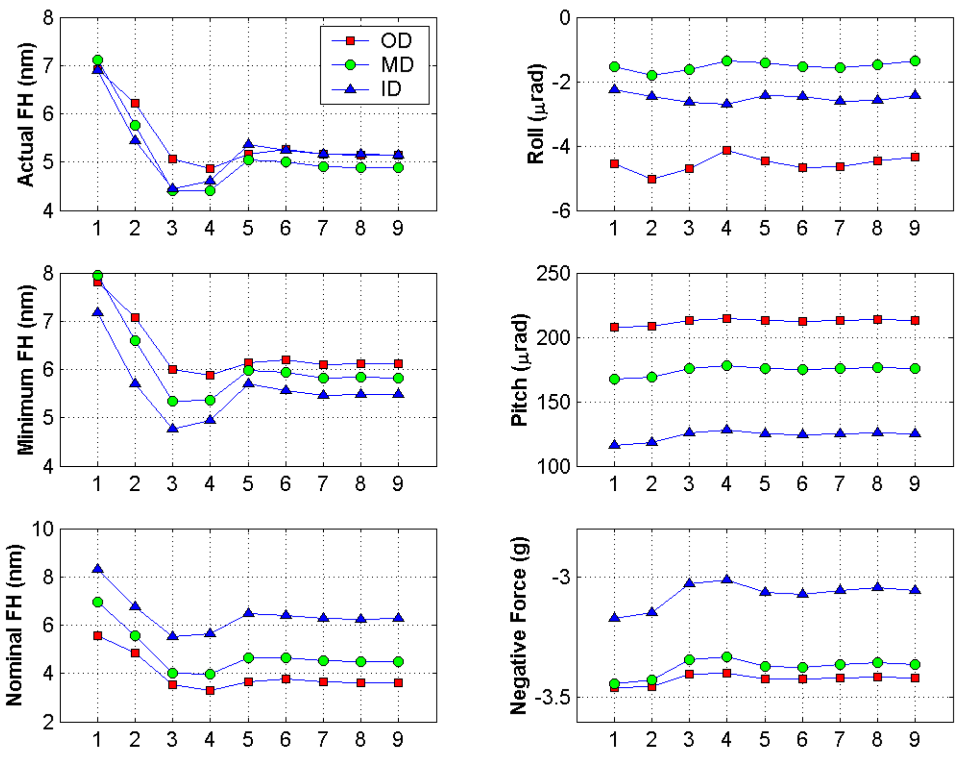


Fig. 5.37 Variations of the slider performance parameters (case 2)

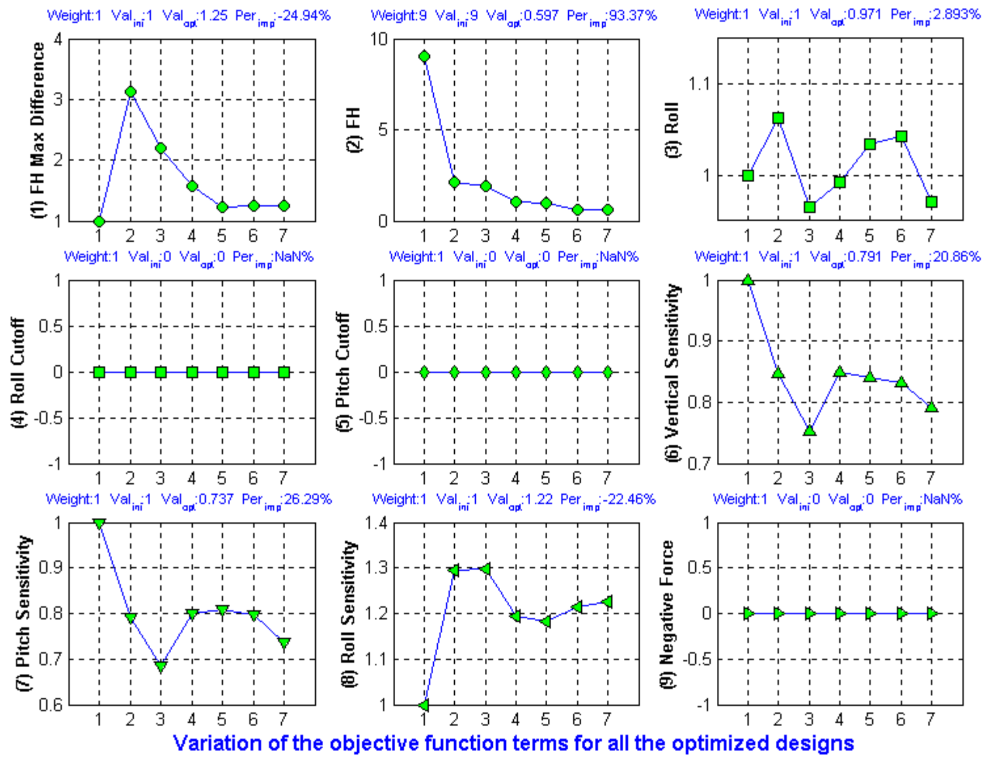


Fig. 5.38 Variations of the objective function terms (case 3)

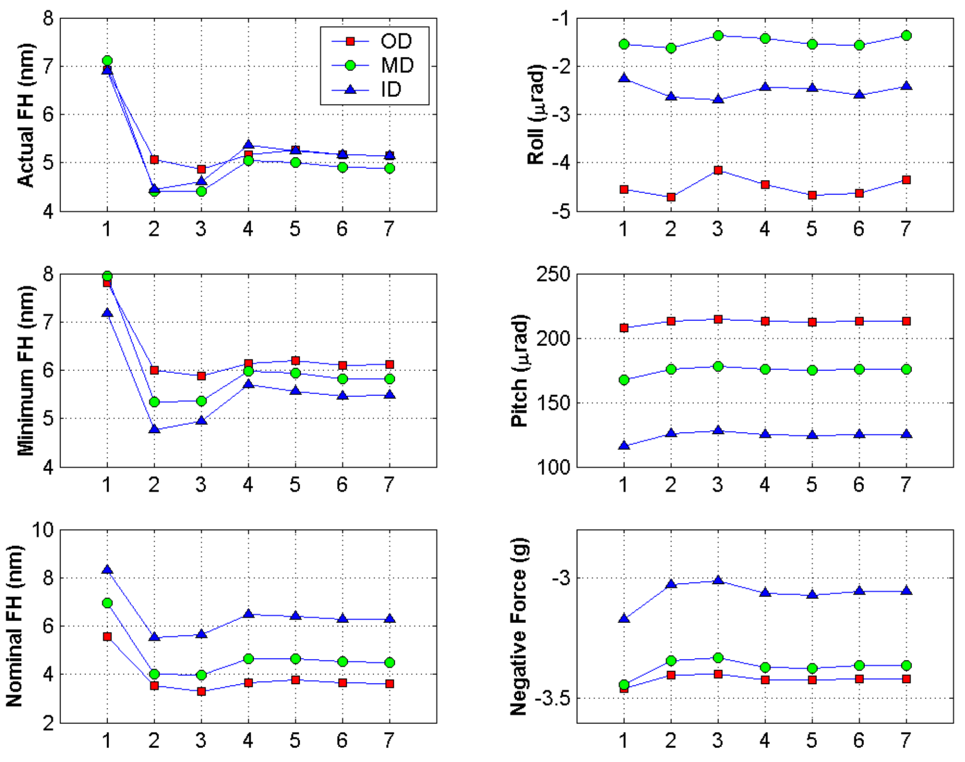


Fig. 5.39 Variations of the slider performance parameters (case 3)

Chapter 6

PERFORMANCE COMPARISON BETWEEN ASA AND DIRECT

6.1 INTRODUCTION

We introduced the Adaptive Simulated Annealing (ASA) algorithm and the DIRECT algorithm in Chapters 2 and 3, respectively. As discussed previously, the ASA algorithm and the DIRECT algorithm are both global optimization techniques. However, these two algorithms belong to two fundamentally different categories. ASA is a stochastic algorithm, which means a random process is introduced within the algorithm. DIRECT is a deterministic algorithm, and thus no random process is involved.

The ASA algorithm, developed by Ingber in 1989, is one of the latest developments of the Simulated Annealing (SA) family, and it has been applied to many areas. Research on the Simulated Annealing technique has also become intensified in recent years. On the other hand, the DIRECT algorithm, developed by Jones in 1993, is a relatively new algorithm and has not been widely used.

Both algorithms are powerful tools for dealing with strongly non-linear problems. Neither require a definite form of the objective function, but instead treat the solver and the objective function as "black boxes". After generating the samples (input) and sending them to the solver, only the values of the objective function given by the solver (output) are needed. For both algorithms, only continuity is required for the optimization problems, i.e., the objective function must be continuous within the search space.

Both algorithms are quite robust. Although ASA has the highest convergence rate within the SA family, it is still relatively slow. It generally needs to be “tuned up” (Ingber, 1996) before it can be applied successfully to a specific problem. Also, since ASA is a stochastic algorithm, it’s reasonable to expect different optimization results for a limited number of samples generated if there are any changes in initial conditions or algorithm parameters. However, statistically, ASA will give the same results if a large number of samples are generated. DIRECT is a deterministic algorithm. Therefore, there are no random factors involved in its optimization process. Interestingly, DIRECT does not need to be “tuned up” for a specific problem, because there are almost no parameters that need to be changed. Jones (1993) and Gablonsky et al. (1998, 2000) showed that DIRECT has a very fast convergence rate. Thus, it can find the global minimum very quickly as compared with other algorithms.

Another distinction between these two algorithms is their relative difficulty of implementation. We found that ASA is much easier to implement than DIRECT. However, once the more difficult implementation of DIRECT is finished, the effort will be rewarded by its amazingly high efficiency.

Next we will compare the relative performance of ASA and DIRECT using numerical simulation and the slider ABS optimization case study.

6.2 PERFORMANCE COMPARISON BY NUMERICAL EXPERIMENTS

6.2.1 Easy test function cases

The test functions used here include 2-D, 10-D and 20-D functions. These functions have only one global and local minimum point, and the minimum values of these functions are zero. They are defined as follows:

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

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

20-D: $$F(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}, x_{20}) = (x_1 - 0.05)^2 + (x_2 - 0.1)^2 + (x_3 - 0.15)^2 + (x_4 - 0.2)^2 + (x_5 - 0.25)^2 + (x_6 - 0.3)^2 + (x_7 - 0.35)^2 + (x_8 - 0.4)^2 + (x_9 - 0.45)^2 + (x_{10} - 0.5)^2 + (x_{11} - 0.55)^2 + (x_{12} - 0.6)^2 + (x_{13} - 0.65)^2 + (x_{14} - 0.7)^2 + (x_{15} - 0.75)^2 + (x_{16} - 0.8)^2 + (x_{17} - 0.85)^2 + (x_{18} - 0.9)^2 + (x_{19} - 0.95)^2 + (x_{20} - 1.0)^2.$$

For all these cases, $x_i \in [0,1], \ i = 1, \ldots 20.$

As the dimensions of a problem increase, the search space becomes much larger, and the difference between the convergence rates of ASA and DIRECT become more obvious. Figures 6.1, 6.2 and 6.3 show the comparison between convergence rate for ASA and DIRECT for the 2-D, 10-D and 20-D cases, respectively.

Figure 6.1 shows that, for the 2-D case, ASA and DIRECT demonstrate similar convergence curves. For the 10-D case shown in Fig. 6.2, we see that ASA and DIRECT reveal different convergence rates. ASA shows a fast convergence rate at the initial stage, but slows thereafter. However, DIRECT shows a steady and very fast convergence rate throughout the whole optimization process. This phenomenon can be observed more clearly in Fig. 6.3 for the 20-D case. Figure 6.3 shows that ASA converges quite quickly when the number of function evaluations is less than 10000, but after that it converges very slowly. ASA required more than 80000 function evaluations to find the best value of 10^{-3}, whereas DIRECT took fewer than 15000 function evaluations to accomplish the same task. Also, the convergence rate of DIRECT remains steady throughout the entire process. The best value found by DIRECT is lower than 10^{-4} when the number of function evaluations reaches 20000. In Fig. 6.3, we also plot the convergence curve for DIRECT-III, a strongly locally biased variation of the standard DIRECT algorithm introduced in Chapter 4. It is clear that DIRECT-III converges even faster than DIRECT in this case.

Consider another 2-D test function that is defined as:

$F(x_1, x_2) = x_1^{\,2} + x_2^{2} + 0.3\, z \sin(13\pi z)^2,$

where $z = x_1^2 + x_2^2 + 0.0001$ and $x_1, x_2 \in [-5,4]$. If we normalize the range of variables x_1 and x_2 into $[0,1]$, then its global minimum is 5.0039E-10 at (5/9, 5/9). The 2-D contour lines and 3-D surface shape of this function are shown in Figs. 6.4 and 6.5, respectively. In Fig. 6.4 the round dot denotes the location of the global minimum point. From Figs. 6.4 and 6.5 it is observed that this function has many local minima due to the perturbation term $0.3\ z\ sin(13\pi z)^2$. However, since the perturbation term contributes very little to the function value, it is still considered here to be an "easy" test function.

Figure 6.6 shows the convergence comparison between ASA and DIRECT for this 2-D test function. ASA and DIRECT show similar convergence rates in this case.

Figure 6.7 shows the convergence comparison for ASA for five different initial conditions: (0.5, 0.5), (0, 0), (0, 1), (1, 1) and (1, 0). Although the convergence curves for these five different initial conditions are different, statistically they are similar to each other. In Fig. 6.7 we see that different initial conditions result in different optimization processes for ASA. However, in this case all of them exhibit similar convergence rates and all of the curves converge to the global minimum.

6.2.2 Tough testing function cases

As we mentioned in Section 3.3.3, the so-called "tough" functions are the ones whose global minima are difficult for the optimization technique to find. Mostly this is caused by either multiple local minima or a wide "flat" area around the global minimum point. These features make the optimization difficult since it's easy for the process to get trapped at a local minimum.

We investigated two tough test functions with one global minimum and multiple local minima. The first function is the 4-D Colville function. It is defined as:

$$\begin{aligned} F(x_1, x_2, x_3, x_4) = {} & 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2 + \\ & 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + \\ & 10.1\,((x_2 - 1)^2 + (x_4 - 1)^2) + \\ & 19.8(x_2 - 1)(x_4 - 1), \end{aligned}$$

where $x_1, x_2, x_3, x_4 \in [-10, 10]$. If we normalize the range of variables x_1, x_2, x_3 and x_4 into $[0,1]$, then its global minimum point is (0.55, 0.55, 0.55, 0.55) and the global minimum is 0.

Figure 6.8 shows the convergence comparison between ASA and DIRECT for the 4-D Colville function. ASA shows a very slow convergence rate after 1000 function evaluations, whereas DIRECT shows a very steady and fast convergence rate.

The second function is the local Shubert function, which 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 $x_1, x_2 \in [-5, 5]$. It has one global minimum point and 100 local minimum points. If we normalize the range of variables x_1 and x_2 into $[0,1]$, then its global minimum point is (0.4508609, 0.4508609) and the global minimum is -24.062499. The 2-D contour and 3-D surface of the local Shubert function are shown in Figs. 4.18 and 4.19, respectively. The round dot in Fig. 4.18 denotes the global minimum point.

Figure 6.9 shows the convergence comparison between ASA and DIRECT when ASA takes the midpoint (0.5, 0.5) as the initial point. We see that, for this case, although both algorithms did find the global minimum, ASA found it faster than DIRECT.

However, if we change the initial point for ASA, the results are quite different. To do that, we divided the square search space into an 8 x 8 grid, and used each grid point as the initial point for the optimization using ASA. Consequently, we tested 81 different initial points for ASA, and we show the results in Fig. 6.10. We combined the contour lines of the local Shubert function with the 8 x 8 grid. The triangular symbols represent the initial points for which ASA failed to find the global minimum. For 18 out of the total 81 initial points ASA failed to find the global minimum. This verifies that the optimization results of ASA are dependent on the initial conditions, which in this case were the initial points. This dependence is caused by the combined effects of the ASA algorithm itself and the embedded random process. The DIRECT algorithm always chooses the midpoint of the search space as its initial point and there is no embedded random process. Therefore, there is no such dependence for DIRECT.

Even thought ASA fails to find the global minimum in Fig. 6.10, this does not mean that ASA is not a global optimization technique. Remember that ASA has many parameters and generally needs to be “tuned up” for each specific problem. In this case, we used the default settings for ASA and did not tune it up. For all those initial points for which ASA failed to find the global minimum, if we adjust some of the parameters (e.g., setting a higher initial temperature, or simply changing the random number generator), ASA will be able to find the global minimum point.

In fact, leaving many parameters to be “tuned up” by users for a specific problem is one of the disadvantages of ASA, especially for problems with expensive function evaluations, because users usually do not have the luxury of “tuning up” the ASA until it performs well. In contrast, the DIRECT algorithm, being deterministic and not needing parameters to be changed by the user, is much more convenient for practical usage.

Note that all of the test functions we chose only have one global minimum. In practical terms it is very difficult for ASA to handle problems with multiple global minima, especially those problems with multiple global minima and many local minima (for example, the Shubert function). We have shown in Chapter 3 that DIRECT can handle the multiple global minima very well, even for some very “nasty” test functions.

6.3 PERFORMANCE COMPARISON USING SLIDER ABS OPTIMIZATION

6.3.1 Air bearing design optimization problems

Here we investigate a 2-D and a 3-D ABS optimization problem. For both cases, we choose the same 7nm FH pico slider defined in Section 2.4 as the prototype slider. Its rail shape and the 3-dimensional rail geometry are shown in Figs. 2.10 and 2.11, respectively. The optimization goals are also the same in both cases: to lower the flying heights to the target flying height (i.e., 5nm), and at the same time maintain a flat roll profile across the three different radial positions OD, MD and ID. We also define the same objective function for both cases as we did in Section 2.4. For the 2-D ABS optimization problem, only two original constraint points are defined, shown in Fig. 5.25. For the 3-D ABS optimization problem, three original constraint points are defined, shown in Fig. 3.36.

6.3.2 Optimization results

Using the same initial design, constraints, and objective function, we carried out the optimization for the 2-D and 3-D ABS optimization problem using both ASA and DIRECT. Notice that for the purpose of comparison, we only use the standard DIRECT algorithm without manufacturing tolerance or hidden constraints. We use 200 function evaluations for DIRECT in both the 2-D and 3-D cases.

6.3.2.1 2-D ABS optimization results

Figure 6.11 shows the convergence comparison between ASA and DIRECT for the 2-D ABS optimization case. Here, the convergence curves of ASA and DIRECT are quite similar. The optimized ABS design found by ASA has the objective function value of 5.561, whereas the optimized ABS design found by DIRECT has the objective function value of

5.571. ASA found a slightly better ABS design in this case. Figures 6.12 and 6.13 show the optimized designs found by ASA and DIRECT, respectively. The gray lines in those two figures represent the initial ABS design, and the dark lines represent the optimized ABS design.

Figures 6.14 and 6.15 show the optimization results for the 2-D case using DIRECT and ASA, respectively. Here the small dots represent the sample ABS designs generated by the algorithms, and the circular dots represent the best-so-far ABS designs found by the algorithms during the optimization process.

It can be observed from Figs. 6.14 and 6.15 that while the optimization results obtained by using the deterministic DIRECT algorithm show a very regular pattern, the optimization results obtained using the stochastic ASA algorithm also show an interesting "band" pattern. That is, most of the sample points generated by ASA for this 2-D case were located in two "bands", which are shown in Fig. 6.16 by the two rectangles ($a \times b1$ and $b \times a1$) surrounded by the dashed lines.

This "band" pattern can be explained by the following:

Let $a2 = a - a1$ and $b2 = b - b1$, and suppose that for the whole optimization process there is a probability of 70% for the new sample points to be generated within the $a1$ and $b1$ intervals for the first and second original constraint points, as shown in Fig. 6.17. Then, for the whole search space, the probability distribution for all the sample points is:

$$((70\%)^{a1} + (30\%)^{a2}) \times ((70\%)^{b1} + (30\%)^{b2})$$
$$= (49\%)^{a1+b1} + (21\%)^{a1+b2} + (21\%)^{a2+b1} + (9\%)^{a2+b2}.$$

We can also verify this by looking at the intersection of the two “bands”, which is *(a1+b1)* and is shown in the shaded area of Fig. 6.17, the section with the greatest density of points.

6.3.2.2 3-D ABS optimization results

Figure 6.18 shows the convergence comparison between ASA and DIRECT for the 3-D ABS optimization case. Here we see that the DIRECT algorithm clearly has a much higher convergence rate than the ASA algorithm. In this case, it takes only about 100 sampling designs for DIRECT to converge on the optimal design, whereas 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.

Figures 6.19 and 6.20 show the optimized designs found by ASA and DIRECT, respectively. The gray lines in those two figures represent the initial ABS designs, and the dark lines represent the optimized ABS designs. Figures 6.21 and 6.22 show the comparison between the FH and the Roll of the optimized ABS designs found by ASA and DIRECT, respectively. Figures 6.21 and 6.22 show that both algorithms found greatly optimized ABS designs, but the optimized ABS design found by DIRECT has even more uniform FHs around the 5nm target FH and flatter rolls. Moreover, DIRECT found the optimized design much faster than did ASA, due to its much higher convergence rate.

6.4 CONCLUSION

The ASA algorithm is one of the latest developments of the Simulated Annealing (SA) family, and it has been applied to many areas. The DIRECT algorithm is a relatively new algorithm and has not yet been widely used.

The ASA algorithm and the DIRECT algorithm are both global optimization techniques. Both algorithms are powerful tools for dealing with strongly non-linear problems, and neither require a definite form of the objective function. The only requirement for both algorithms is that the objective function must be continuous within the search space.

However, these two algorithms belong to two fundamentally different categories. ASA is a stochastic algorithm, which means it introduces a random process within it. DIRECT is a deterministic algorithm and no random process is involved.

Both ASA and DIRECT are quite robust. ASA generally needs to be “tuned up” in order to be successfully applied to a specific problem. Also, since ASA is a stochastic algorithm, it is reasonable to expect different optimization results for a limited number of samples generated for any changes in initial conditions or algorithm parameters However, statistically, ASA gives the same results if a large number of samples are generated. DIRECT is a deterministic algorithm, and therefore there are no random factors involved in the optimization process. DIRECT does not need to be "tuned up". Thus, DIRECT is more convenient for practical usage.

Although DIRECT is initially more difficult to implement than is ASA, it also has a much higher convergence rate than dose ASA. Thus, DIRECT can find the global minimum more quickly than ASA. This property is observed more clearly for higher dimensional problems. We have verified this by numerical simulations as well as in slider ABS optimization case studies.

Also notice that it is quite easy for DIRECT to handle the manufacturing tolerance issue (refer to Chapter 5 for details). This, together with its high convergence rate, makes DIRECT more suitable for the slider ABS optimization.

In summary, the DIRECT algorithm clearly outperforms the ASA algorithm in our test cases, and we consider it more suitable for the slider ABS optimization than is ASA.

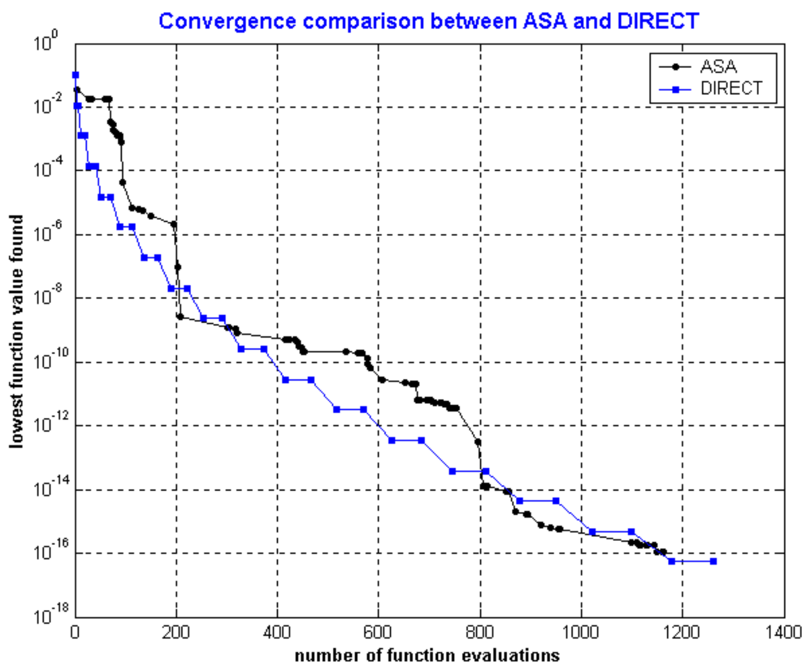


Fig. 6.1 Convergence comparison for 2-D case

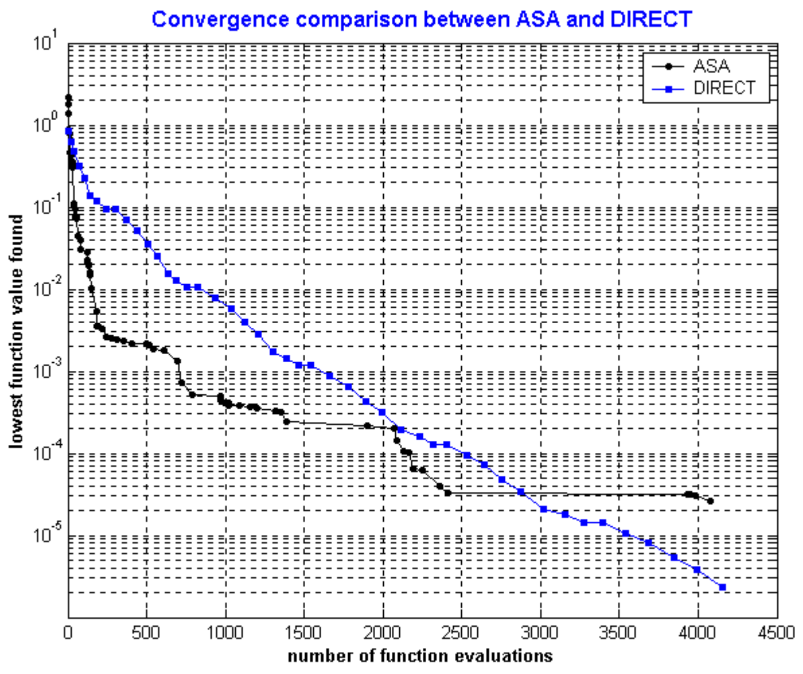


Fig. 6.2 Convergence comparison for 10-D case

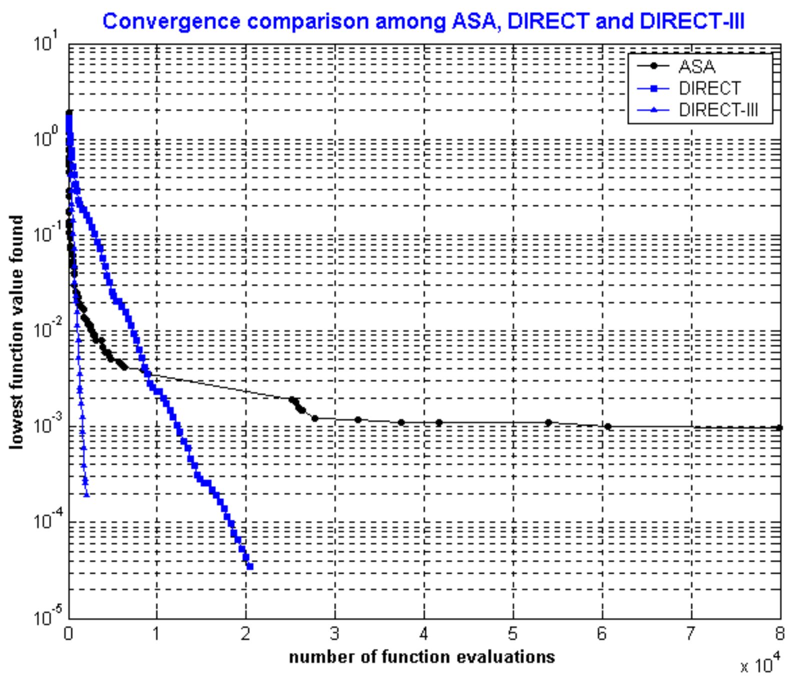
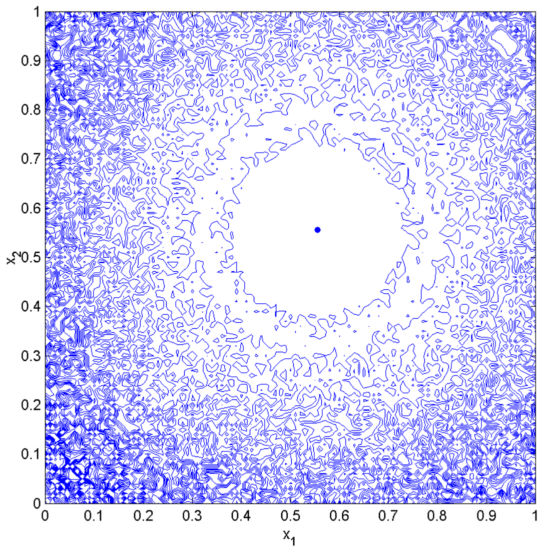


Fig. 6.3 Convergence comparison for 20-D case



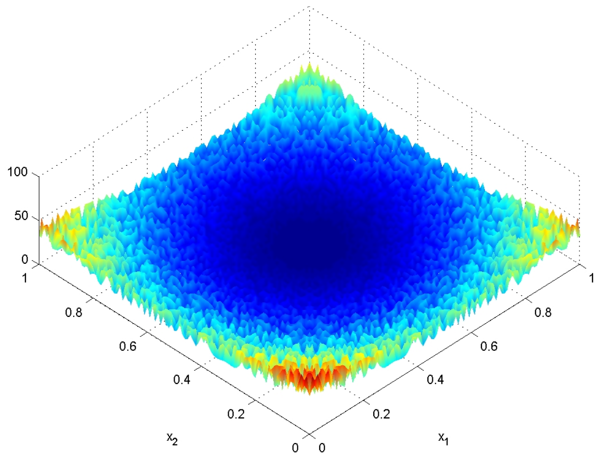


Fig. 6.4 Contour lines

Fig. 6.5 Surface shape

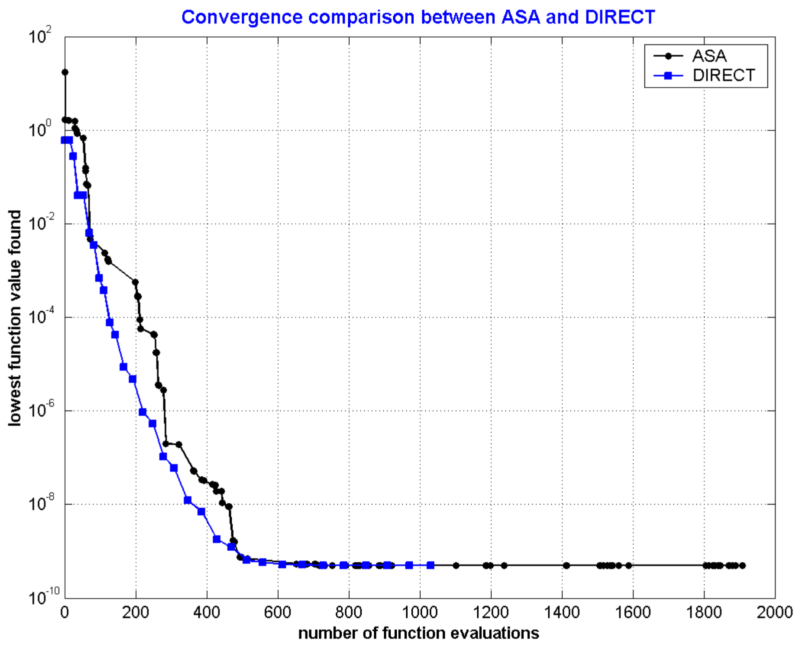


Fig. 6.6 Convergence comparison

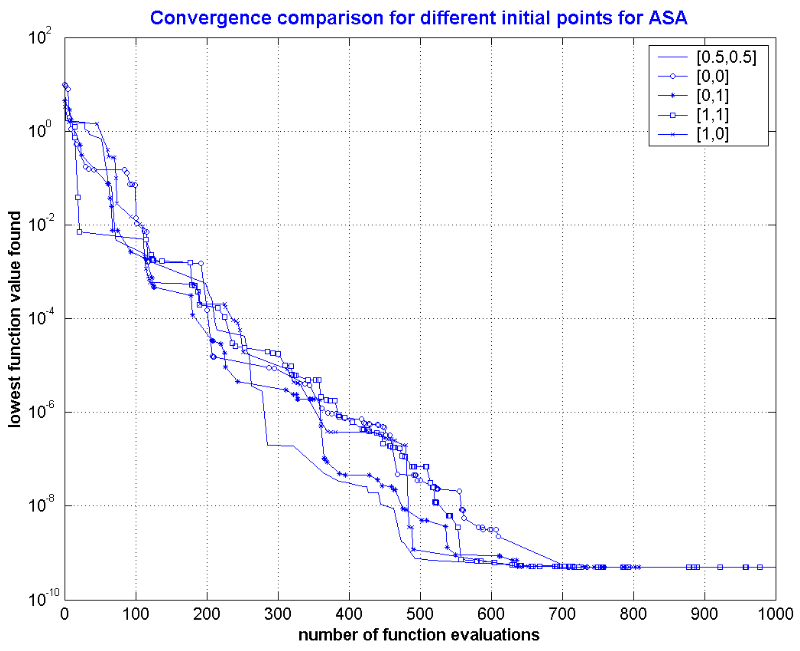


Fig. 6.7 Convergence comparison for different initial points using ASA

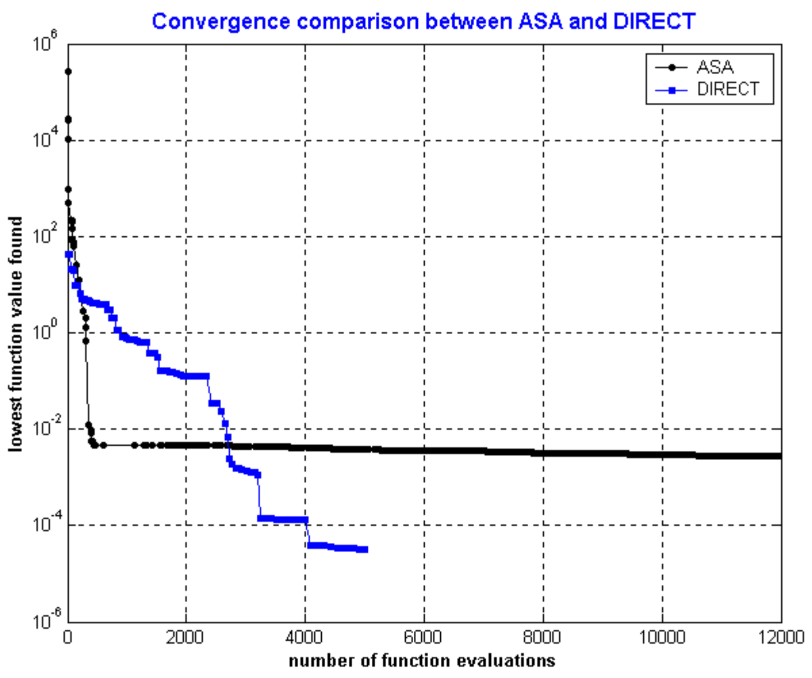


Fig. 6.8 Convergence comparison for 4-D Colville function

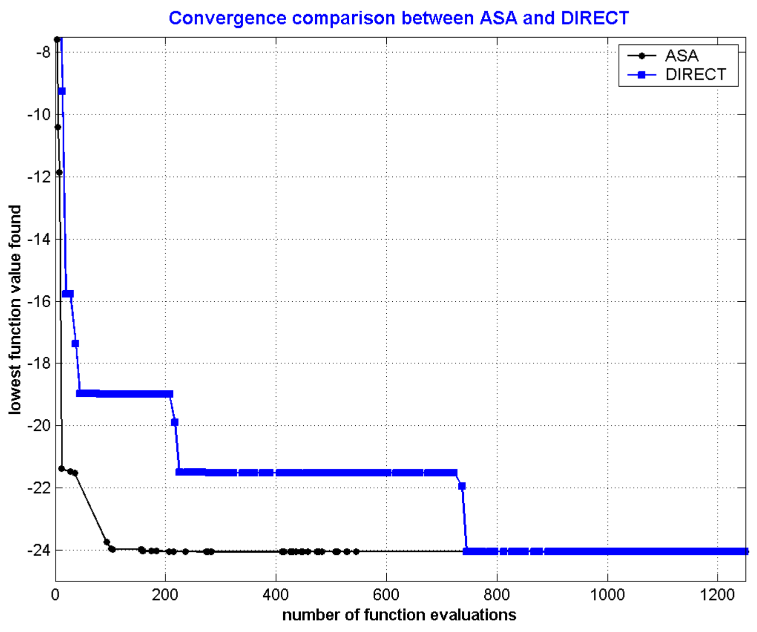


Fig. 6.9 Convergence comparison for local Shubert function

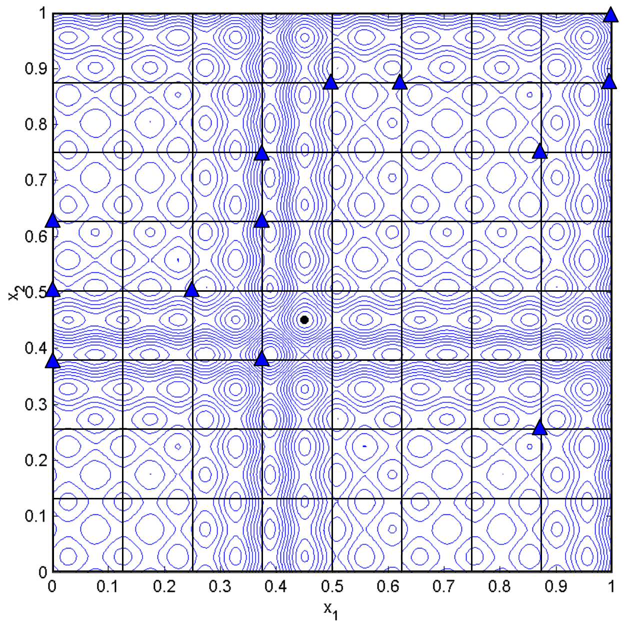


Fig. 6.10 Optimization results for different initial points for ASA

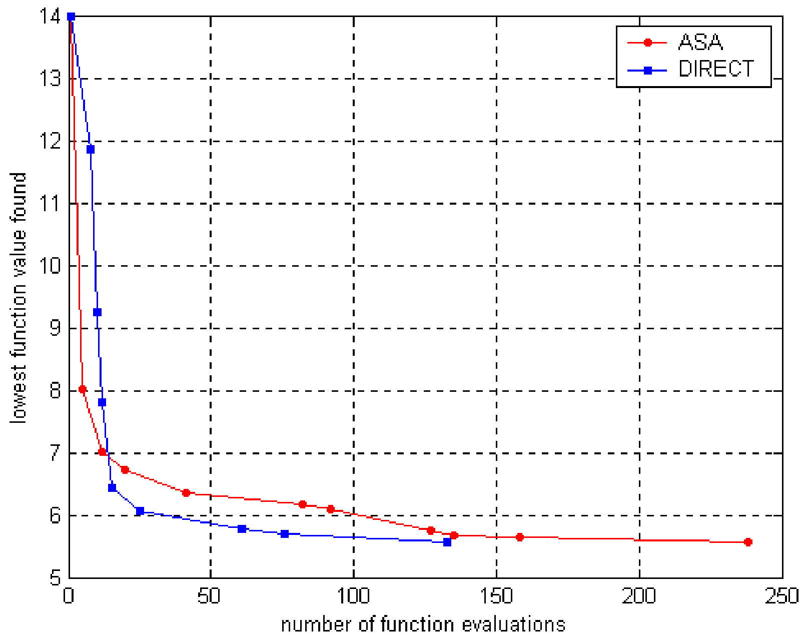


Fig. 6.11 Convergence comparison for the 2-D ABS optimization problem

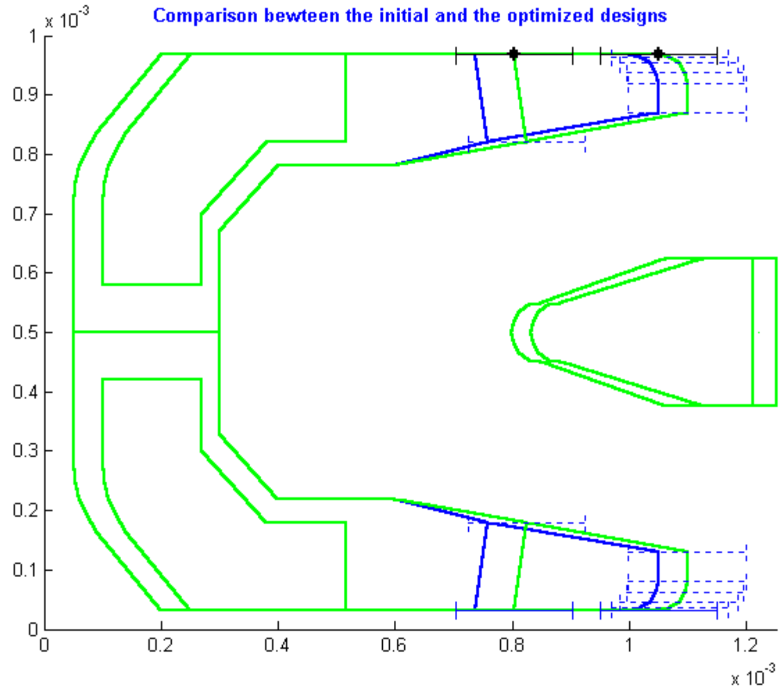


Fig. 6.12 Optimized design obtained by DIRECT

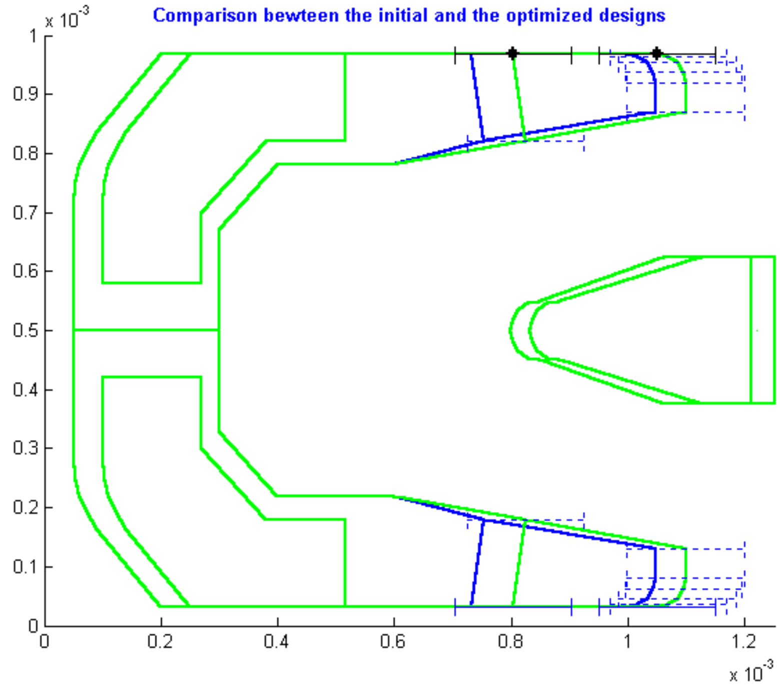


Fig. 6.13 Optimized design obtained by ASA

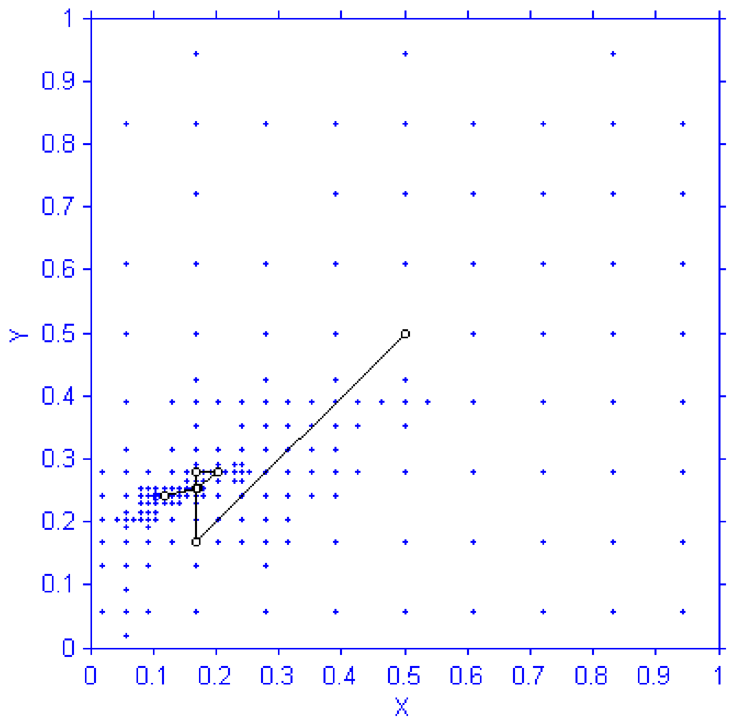


Fig. 6.14 Optimization results using DIRECT

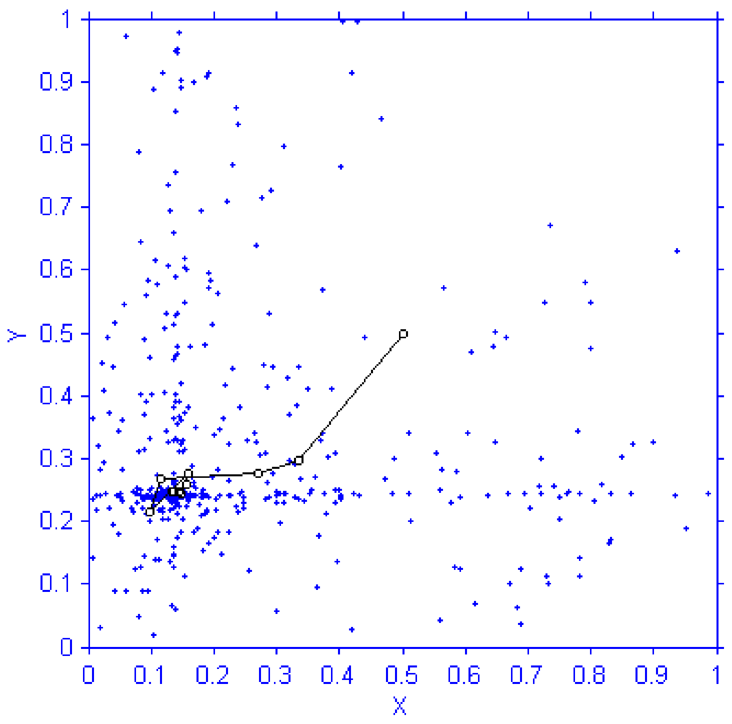


Fig. 6.15 Optimization results using ASA

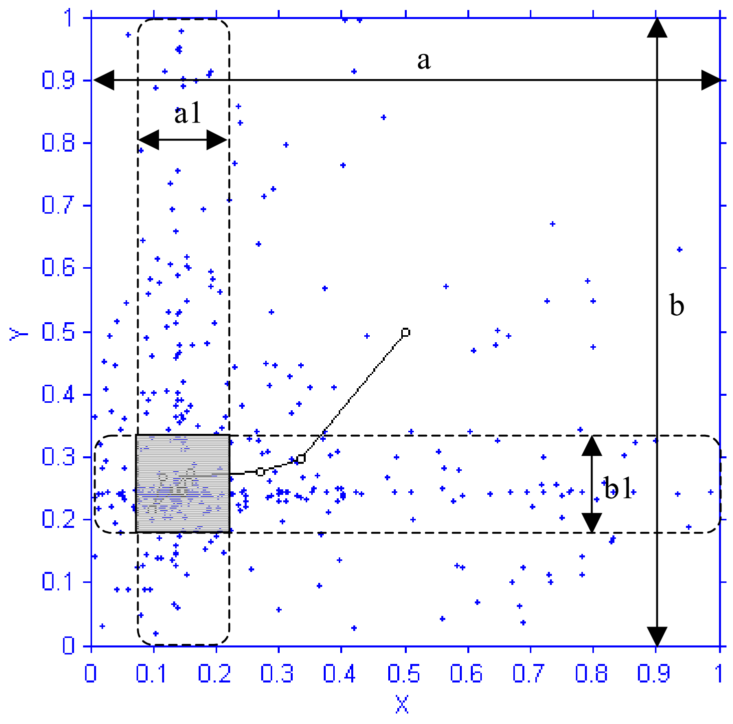


Fig. 6.16 Illustration of the “band” pattern of the optimization results using ASA

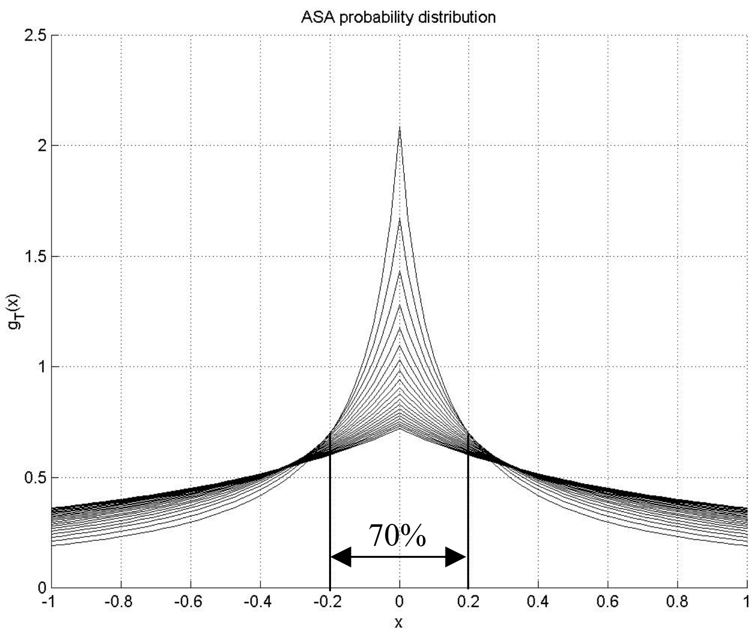


Fig. 6.17 ASA probability function

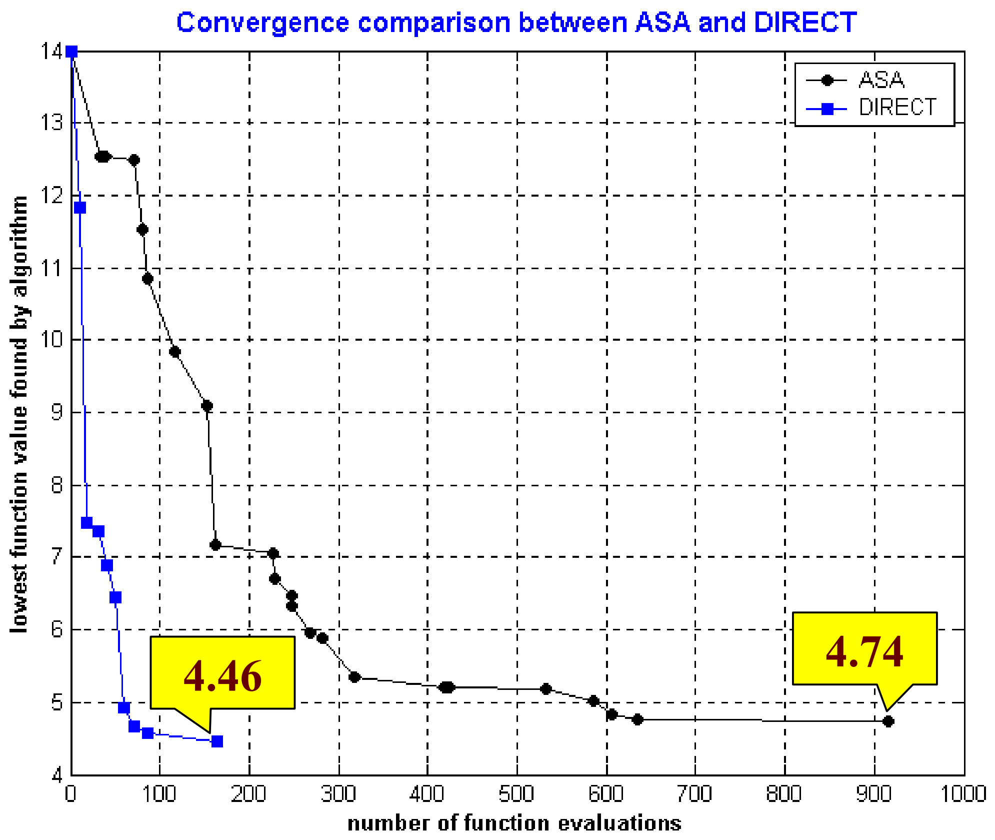


Fig. 6.18 Convergence comparison for the 3-D ABS optimization problem

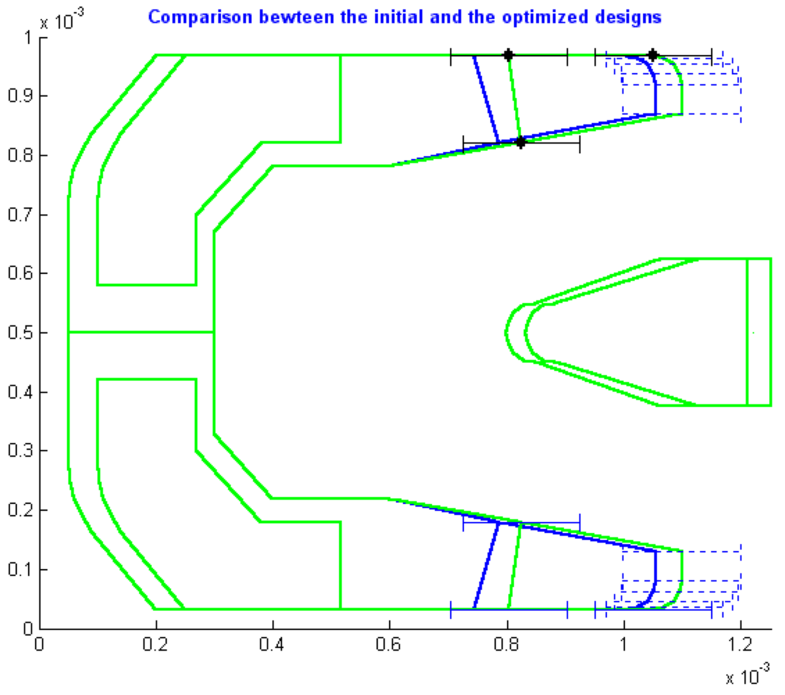


Fig. 6.19 Optimized design obtained by DIRECT

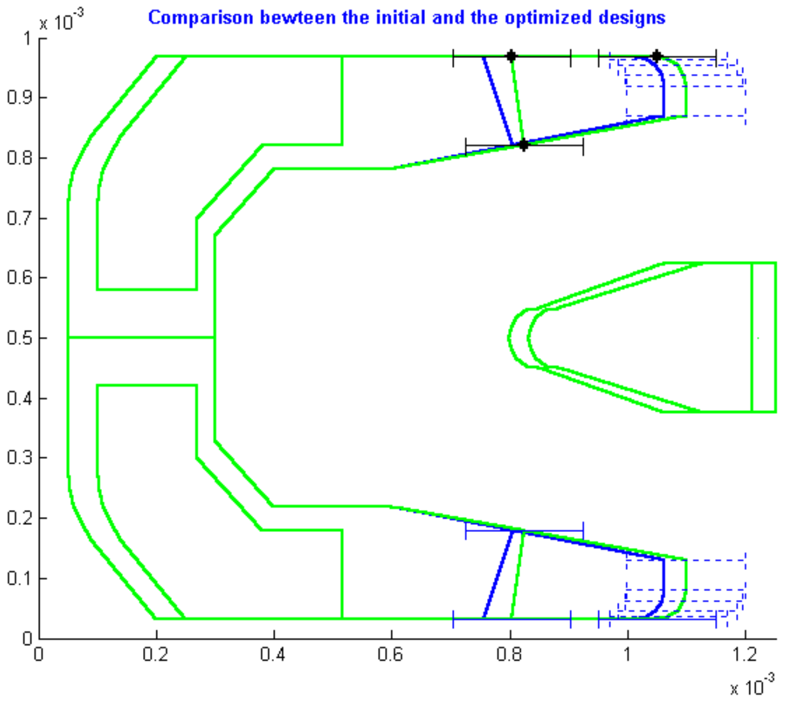
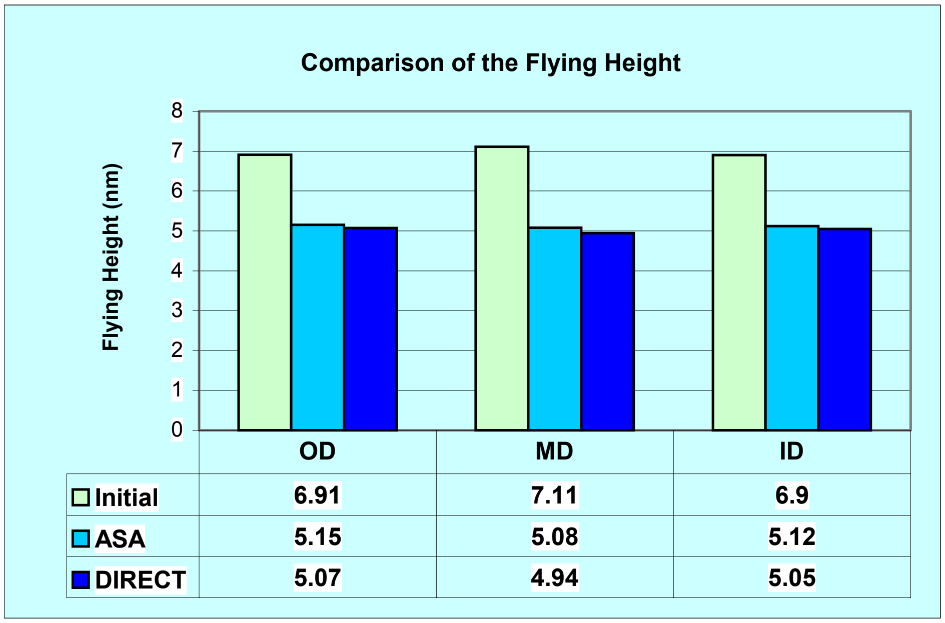
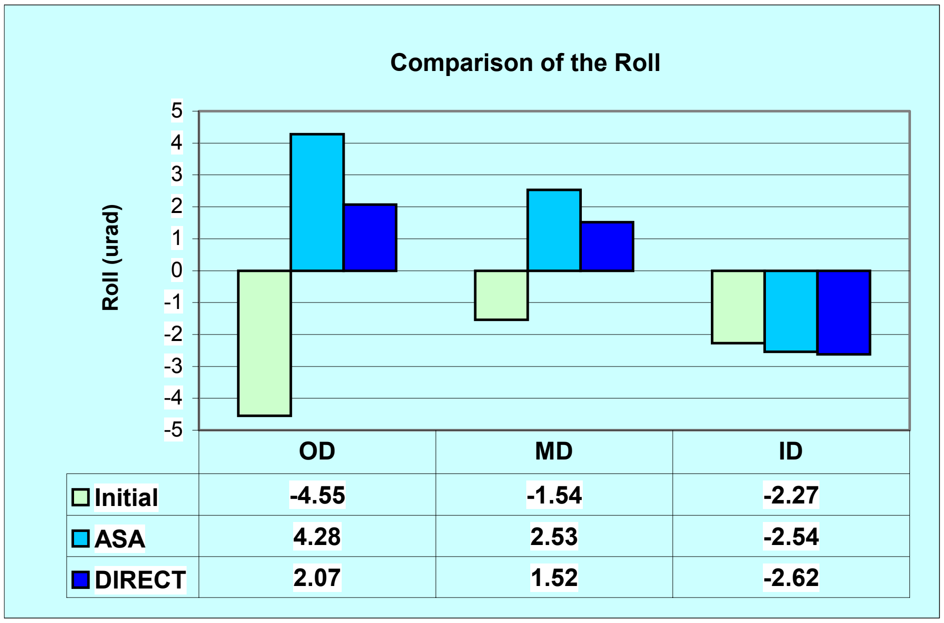


Fig. 6.20 Optimized design obtained by ASA



	OD	MD	ID
🟩 Initial	6.91	7.11	6.9
🟦 ASA	5.15	5.08	5.12
🟦 DIRECT	5.07	4.94	5.05

Fig. 6.21 Comparison of the FH for the 3-D ABS optimization case



	OD	MD	ID
🟩 Initial	-4.55	-1.54	-2.27
🟦 ASA	4.28	2.53	-2.54
🟦 DIRECT	2.07	1.52	-2.62

Fig. 6.22 Comparison of the Roll for the 3-D ABS optimization case

Chapter 7

NEW GEOMETRIC CONSTRAINTS

7.1 INTRODUCTION

In the previous CML air bearing surface design optimization program (version 1.5 developed by O'Hara (1997)), the only geometric constraints that could be defined in each optimization problem were the constraint points themselves. This meant that, for instance, we could only alter the points defining the rails to move in the length or width direction within certain intervals. As designs have gotten more complex the need for greater flexibility in constraint definition has developed. For example, if we need to rotate an entire rail in order to find the optimized ABS design, we cannot do it by such a limitation on constraints. Therefore, in order to extend the capability of the optimization program, we need to introduce new geometric constraints.

In this chapter, we first introduce the definition of the new geometric constraints, then we apply these new geometric constraints to slider ABS optimization. We also discuss the concept of slider ABS design sensitivity optimization in this chapter.

7.2 DEFINITION OF NEW GEOMETRIC CONSTRAINTS

Figure 7.1 shows a comparison between the new geometric constraints and the old geometric constraints. In the new geometric constraints, we not only define constraint points, we also define constraint rails and constraint lines. The constraint rails can be translated in either the length or width direction, rotated with respect to a fixed point, and expanded or shrunk proportionally. The constraint lines can be translated in either the length or width direction, rotated with respect to a fixed point, and extended or contracted along the length direction. To maintain a symmetrical slider ABS design and fixed local geometric shapes, we also define symmetrical and relative constraints for constraint rails and constraint lines. Please refer to CML technical report "The CML Air Bearing Optimization Program Version 3.0" for more details.

We demonstrate the new rail constraints in Figs. 7.2 ~ 7.4. In these figures, the gray lines show the original slider ABS design, and the dark lines show the shape of the slider ABS design after the rail deformation. Notice that we also define symmetrical constraints in order to maintain a symmetrical slider ABS design. In Fig. 7.2, a rear rail is translated in the length direction. In Fig. 7.3, a rear rail is rotated with respect to one of its rail points. In Fig. 7.4, a front rail is shrunk proportionally.

We demonstrate the new line constraints in Figs. 7.5 ~ 7.7. In these figures, the gray lines show the original slider ABS design, and the dark lines show the shape of the slider ABS design after the line deformation. We also define the symmetrical constraints in order to maintain a symmetrical slider ABS design. In Fig. 7.5, a line is translated in the length direction. In Fig. 7.6, a line is rotated with respect to one of its endpoints. In Fig. 7.7, a line is extended along its length direction.

Next we apply those new geometric constraints to slider ABS optimization problems.

7.3 APPLICATION TO SLIDER ABS OPTIMIZATION

7.3.1 Air bearing design optimization problems

The optimization problem defined here is: given a prototype slider ABS design, optimize it to obtain uniform flying heights near the target flying height and a flat roll profile across the disk.

To demonstrate the application of the new geometric constraints, we use a so-called "Enterprise" slider as the prototype slider. This is a pico slider that flies over a disk rotating at 7200 RPM with FHs around 5nm. Its rail shape and its 3-dimensional rail geometry are shown in Figs. 7.8 and 7.9, respectively.

We wish to lower the flying heights of this slider to the target flying height (i.e., 3.5nm), and at the same time maintain a flat roll profile at the three different radial positions OD, MD and ID. The objective function or cost function is defined as:

1$\times$ *(FH Max Difference term)* + ***9*** $\times$ *(FH term)* + ***1*** $\times$ *(Roll term)* +

1 $\times$ *(Roll Cutoff term)* + ***0*** $\times$ *(Pitch Cutoff term)* + ***0*** $\times$ *(Vertical Sensitivity term)* +

$$\mathbf{0} \times (\textit{Pitch Sensitivity term}) + \mathbf{0} \times (\textit{Roll Sensitivity term}) + \mathbf{0} \times (\textit{Negative Force term})\,.$$

Note that since we are primarily concerned with the flying heights, we put a heavier weight (9) on that term. Because our only objective is to lower the FHs and maintain a flat roll profile, we put a 0 weight on all the sensitivity terms and the negative force term.

We did not define a manufacturing tolerance, but we did impose the hidden constraints:

FH ≤ 2 nm or FH ≥ 10 nm or Roll ≤ -30 μrad or Roll ≥ 30 μrad.

7.3.2 Air bearing design optimization results

For the same prototype slider ABS design, the same objective function and the same hidden constraints, we carried out optimizations for two cases: a 4-D case and a 2-D case.

7.3.2.1 4-D ABS optimization results

For the 4-D ABS optimization case, we defined four original constraint rails. The front rail and the rear rail can be expanded or shrunk. The side rails can be expanded or shrunk and also translated in the length direction and. Figure 7.10 shows the four original constraint rails. Notice that we also defined constraints to maintain the symmetrical configuration of the “Enterprise” slider design.

Figure 7.11 shows the variation of the objective function value during the optimization process. It shows that, although we defined relatively “loose” hidden constraints, there are still many infeasible samples. We see that there are 160 infeasible samples out of the 224 samples generated. This means that the design is quite sensitive to some of the constraint rails, i.e., small changes may have large effects on the slider performance. The objective function value was lowered from the initial value 11 to the final value of 2.455.

Figure 7.12 shows the comparison between the initial design and the optimized design. The gray lines represent the initial “Enterprise” design and the dark lines represent the final optimized design. From Fig. 7.12 we see that for the optimized design, the front rail and the rear rail remain unchanged. Only the side rails have been shrunk and moved toward the leading edge.

Figure 7.13 shows the variations of the slider’s performance parameters for all the best-so-far ABS designs found during the optimization process. Table 7.1 summarizes the optimization results by comparing the FHs, Rolls and Pitches of the initial and the optimized designs. We observe that the optimized ABS design has uniform FHs around the 3.5nm target and maintains a flat roll profile.

7.3.2.2 2-D ABS optimization results

In optimization care must be taken to define the constraints properly. Improperly defined constraints might yield only slightly optimized results or even no optimization at all. We also want to keep the dimensions of an optimization problem as low as possible, because lower dimensions mean a smaller search space, and a smaller search space means the optimized design is found more rapidly.

The previous 4-D ABS optimization results show that in order to lower the FHs of the "Enterprise" slider from 5nm to 3.5nm, we need only change the shape and location of the side rails. Therefore, in our next example we define only two original constraints for the side rails, i.e., they can be moved in the length direction and expanded or shrunk. The front rail and the rear rail remain fixed in this case.

Figure 7.14 shows the variation of the objective function value during the optimization process. After we removed the two constraints for the front rail and the rear rail, which are quite sensitive, the number of infeasible sample points has been reduced substantially. In this case there are only 18 infeasible samples out of the 401 samples generated. The objective function value has been lowered from the initial value of 11 to the final value of 1.809, which is smaller than the value of 2.455 in the 4-D case. This indicates that the 2-D case found a better-optimized slider ABS design than did the previous 4-D case.

Figure 7.15 shows the comparison between the initial design and the optimized design. If we compare Fig. 7.15 with Fig. 7.12, we see that for the better-optimized ABS design found in the 2-D case, the side rails have been shrunk less than in the 4-D case, but they have been moved further toward the leading edge.

Figure 7.16 shows the variations of the slider performance parameters for all the best-so-far ABS designs found during the optimization process. Figure 7.17 shows the variations of the objective function terms for all the best-so-far designs. We see that the second objective function term, on which we put a heavier weight, has been reduced very effectively. Table 7.2 summarizes the optimization results by comparing the FHs, Rolls and Pitches of the initial and the optimized designs. If we compare Table 7.2 with Table 7.1, we see that the optimized ABS design found in the 2-D case has more uniform FHs around the 3.5nm target FH and that it also maintains a slightly flatter roll profile.

Figure 7.18 shows the distribution of the sample points within the search space for the 2-D slider ABS optimization case. Figure 7.19 shows a local zoom-in around the best point. The small dots in these two figures represent the sample points generated by the algorithm; the shadowed boxes represent the boxes containing the infeasible points; and the circular dot represents the best point found by the algorithm. The scatter pattern of the infeasible points inside the search space reflects to some extent the strong nonlinear property of this slider ABS optimization problem. Figure 7.20 shows the objective function contour lines according to the results obtained. The round dots represent the best-so-far sample points found by the algorithm during the optimization process.

7.4 SLIDER ABS SENSITIVITY OPTIMIZATION

One of the important issues of slider design and optimization is the slider ABS sensitivity. Due to the manufacturing tolerance, there are always minor differences between the actual fabricated slider designs and the numerically optimized ones.

If the optimized slider ABS design is too sensitive for the manufacturing tolerance, that is, if minor manufacturing tolerance differences have large effects on the slider's performance, then this optimized design will be unsuitable for fabrication. With high sensitivity, a large fraction of the fabricated sliders will fail to satisfy the performance requirements. Therefore, slider ABS sensitivity optimization is of great importance. In this section, we only present some preliminary thoughts on how to perform slider ABS sensitivity optimization. To carry out slider ABS sensitivity optimization, we need to include a sensitivity term in the objective function. That sensitivity term should involve slider performance parameters such as FHs, Rolls and Pitches. We also need to prescribe the sensitivity constraints. With use of the new geometric constraints, the definition of the sensitivity constraints becomes much easier. After each new sample ABS design is generated and evaluated, it is perturbed with the manufacturing tolerance according to the sensitivity constraints prescribed. After a set of perturbed ABS designs are generated and evaluated, the objective function is calculated. The optimization process continues. Figure 7.21 presents a schematic illustration for this process. The symbol Δ represents the manufacturing tolerance. We see that the new objective function consists of two parts: the first part contains the current objective function terms, and the second part is the sensitivity term.

As can be observed from Fig. 7.21, for every sample ABS design generated, at least two perturbed samples must be evaluated ($\pm\Delta$). For example, if we define two sensitivity constraints, for every new sample ABS design generated, four perturbed sample designs need to be evaluated to get the value of the sensitivity term. It follows that sensitivity optimization requires much longer simulation time. So naturally, we should also think of ways to improve the efficiency of sensitivity optimization. We proposed two ways here to improve the efficiency when performing sensitivity optimization.

The first one is to make use of the hidden constraints during the sensitivity optimization. There are two cases. The first case is shown in Fig. 7.22. The idea is, if the sample design generated satisfies the hidden constraints and is identified as an infeasible design, then it will not be perturbed and the sensitivity term will not be evaluated. The second case is shown in Fig. 7.23. The idea is, if any of the perturbed designs satisfies the hidden constraints, the rest of the perturbed designs will not be generated and evaluated. Again the sensitivity term will not be evaluated. The sample design will be marked as infeasible and later it will be given a pseudo value.

The second way is to make use of the first part of the objective function, as shown in Fig. 7.24. The idea is, after we have evaluated the sample design generated and thus attained the value of the first part of the objective function, if that value is larger (by a weight ≥ 1) than the value of the objective function of the best-so-far optimized design, then we will not perturb it. The reason is obvious: that sample design will not become an optimized design because even if its sensitivity term is 0 its objective function value is still higher than that of the best-so-far optimized design.

The two methods we proposed here are expected to improve the efficiency of the sensitivity optimization by greatly reducing the slider ABS samples need to be evaluated.

7.5 CONCLUSION

In this chapter we defined new geometric constraints for slider ABS optimization, including not only constraint points, but also constraint rails and constraint lines.

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

We applied the new geometric constraints to slider ABS optimization problems, investigating a 4-D case and a 2-D case. Although we found satisfactory optimization results in both cases, a better-optimized slider ABS design was found in the 2-D case by redefining the constraints. This verifies the importance of properly defining constraints.

In conclusion, the new geometric constraints make the slider ABS optimization much easier by enabling users to explore a much wider range of constraints among those found in practical ABS optimization problems.

We also discussed the issue of the slider ABS sensitivity optimization. The definition of the new geometric constraints will make the sensitivity optimization much easier. We presented the basic idea for performing the sensitivity optimization. Some possible ways to improve the efficiency for the sensitivity optimization have also been proposed.

	Initial ABS design			**Optimized ABS design**		
	OD	**MD**	**ID**	**OD**	**MD**	**ID**
FH (nm)	4.74	5.38	5.31	3.21	3.44	3.42
Roll (μrad)	1.52	1.34	-1.10	1.13	1.84	-1.46
Pitch (μrad)	98.3	99.2	98.5	88.9	90.2	90.5

Table 7.1 Summary of the optimization results for the 4-D case

	Initial ABS design			**Optimized ABS design**		
	OD	**MD**	**ID**	**OD**	**MD**	**ID**
FH (nm)	4.74	5.38	5.31	3.38	3.57	3.49
Roll (μrad)	1.52	1.34	-1.10	0.82	1.75	-1.56
Pitch (μrad)	98.3	99.2	98.5	90.9	92.3	91.9

Table 7.2 Summary of the optimization results for the 2-D case

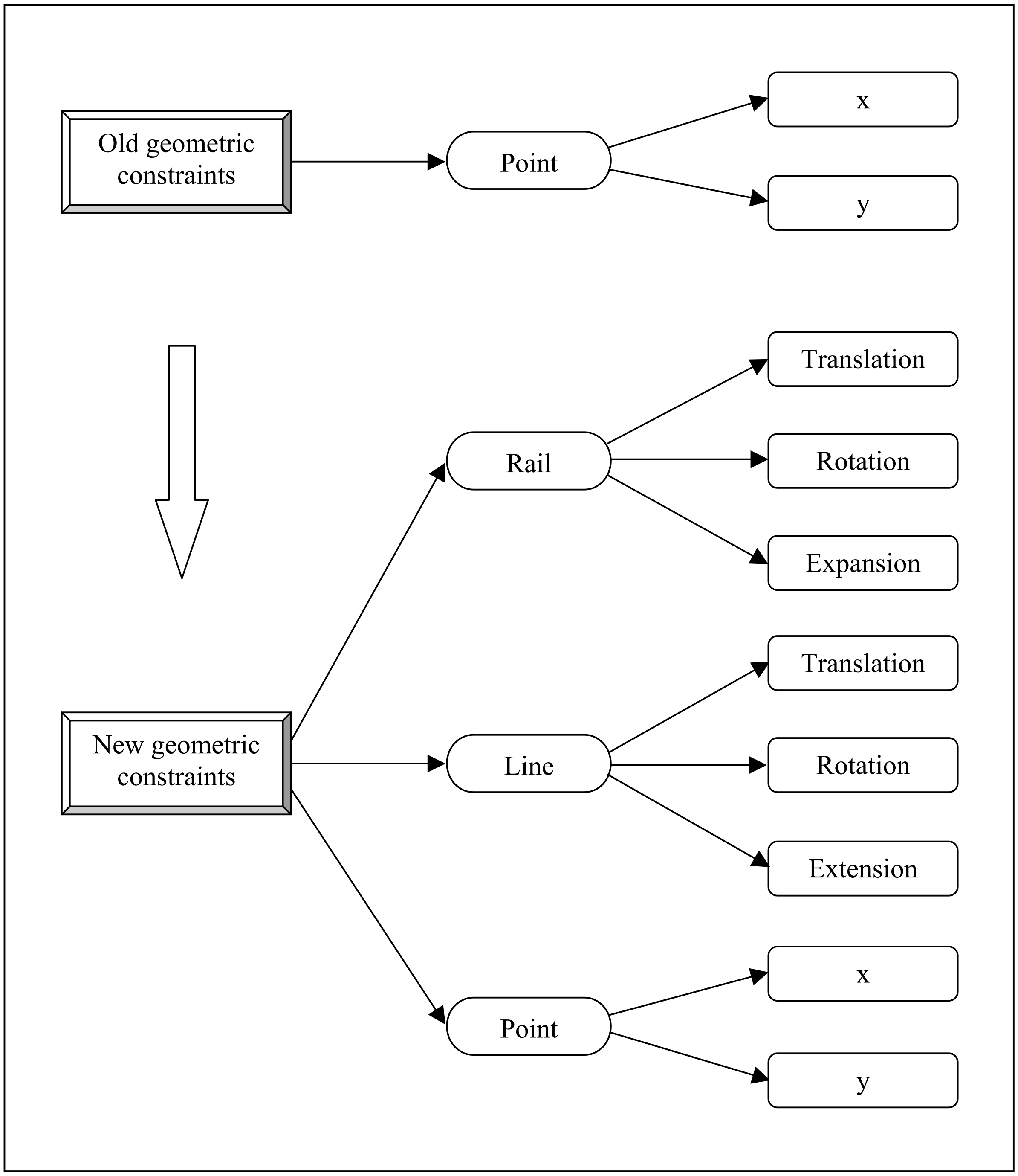


Fig. 7.1 Comparison between the old and the new geometric constraints

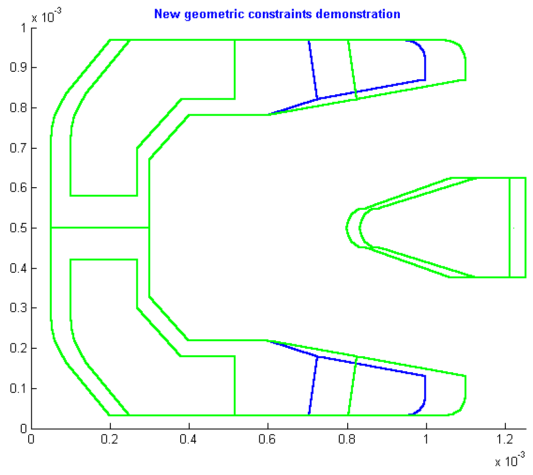


Fig. 7.2 Rail translation

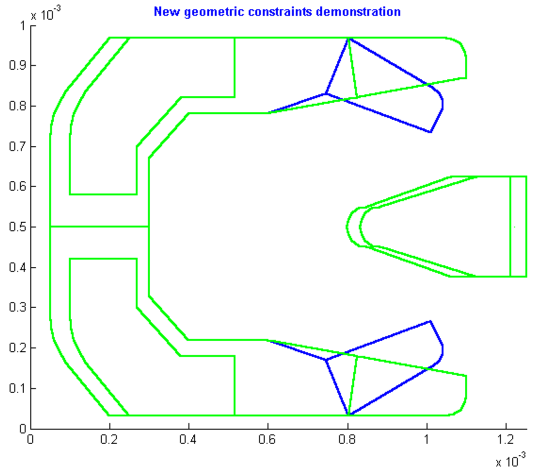


Fig. 7.3 Rail rotation

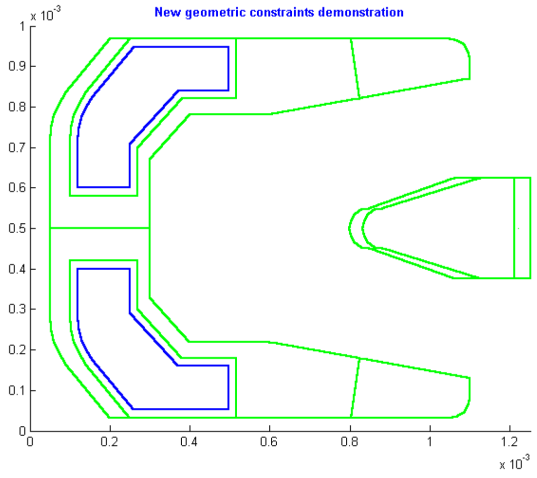


Fig. 7.4 Rail expansion

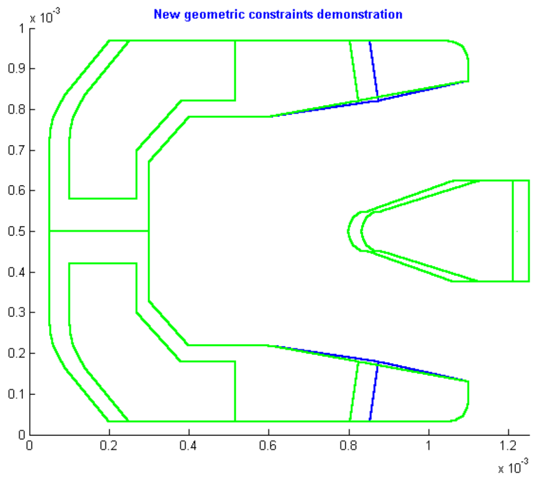


Fig. 7.5 Line translation

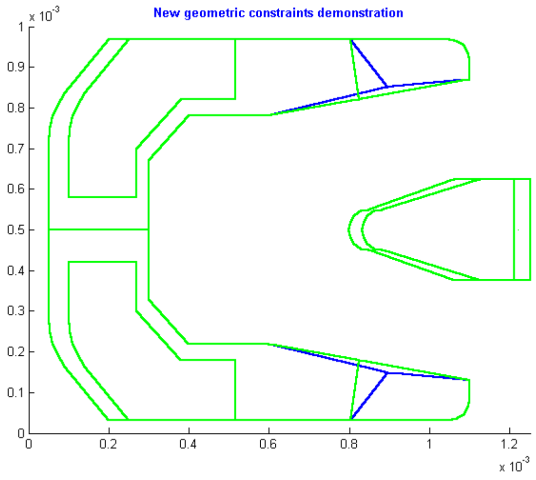


Fig. 7.6 Line rotation

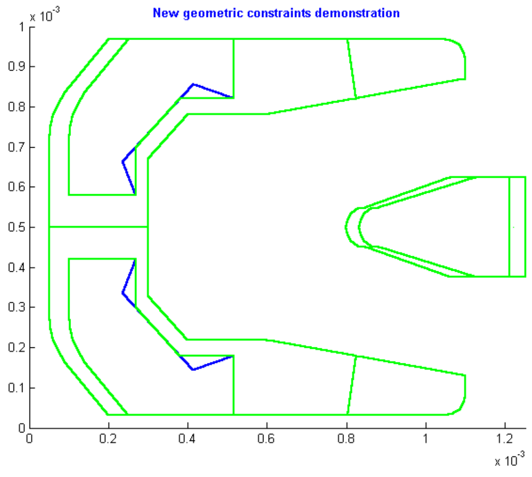


Fig. 7.7 Line extension

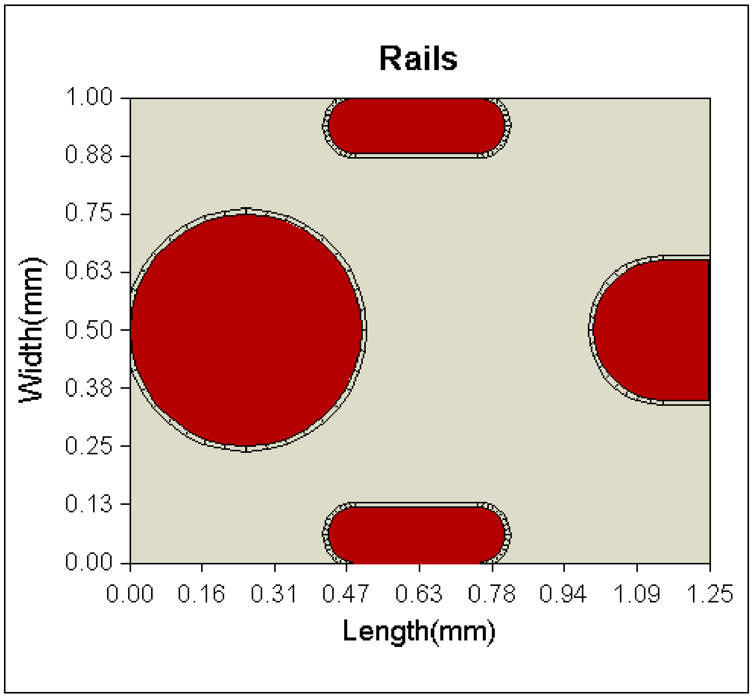


Fig. 7.8 Rail shape of the initial “Enterprise” slider design

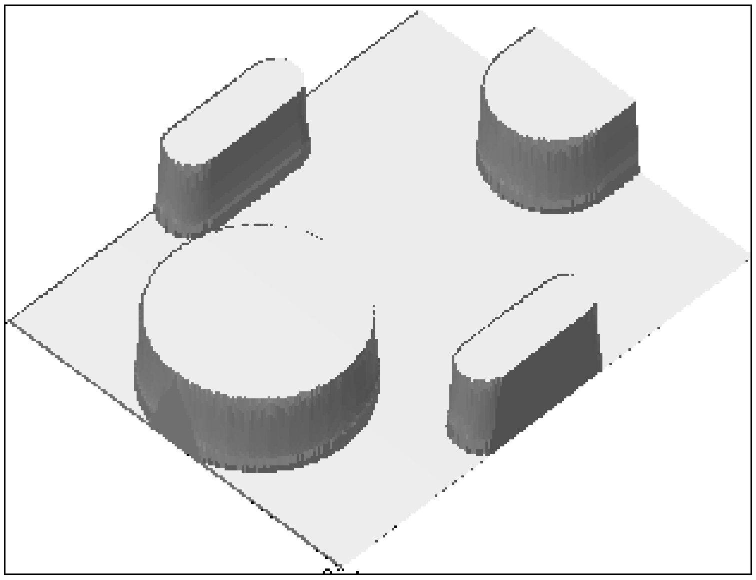


Fig. 7.9 3-D rail shape of the initial “Enterprise” slider design

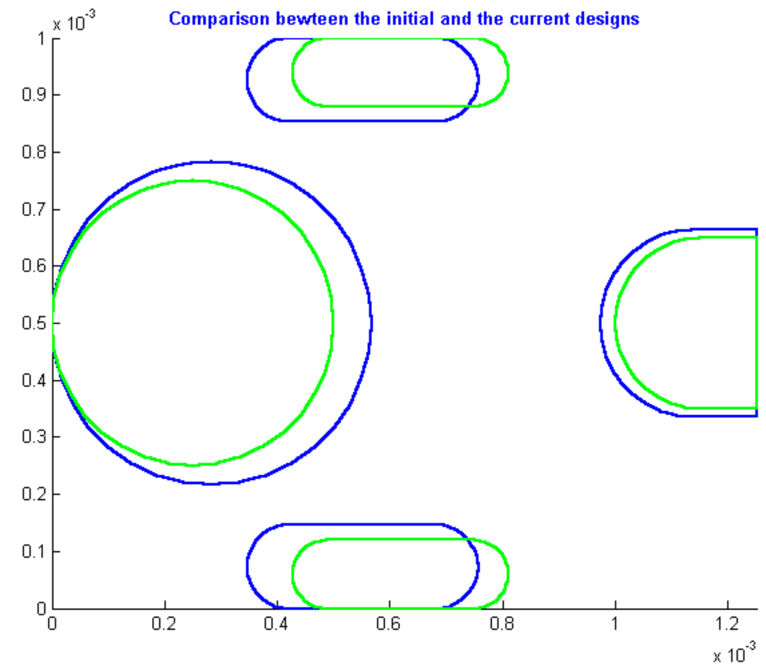


Fig. 7.10 Demonstration of the constraint rails prescribed for the 4-D case

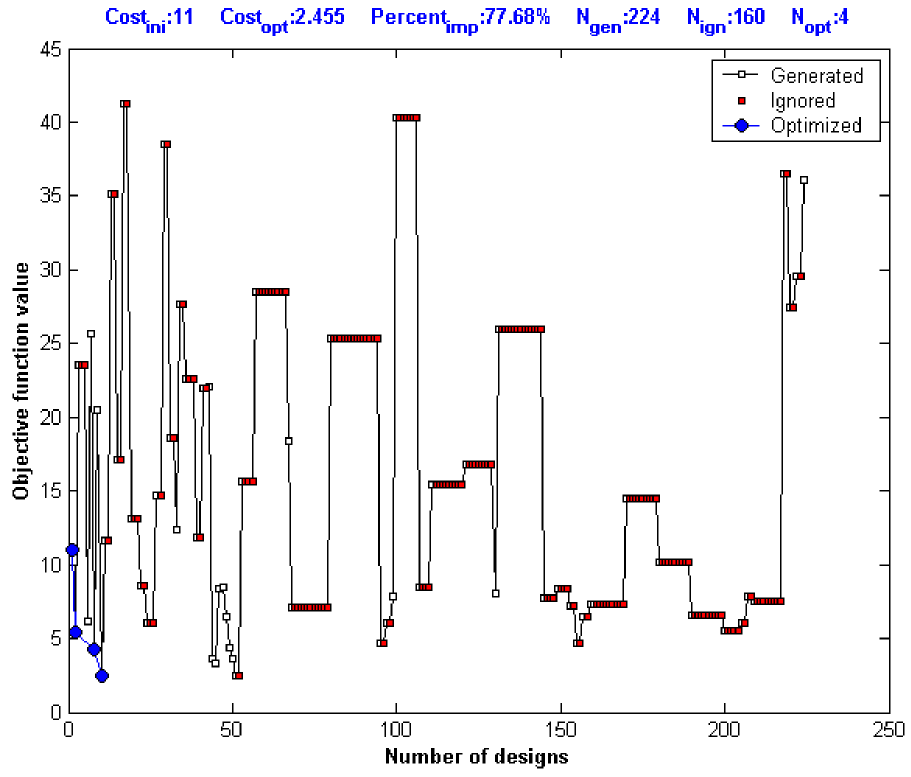


Fig. 7.11 Variation of the objective function value for the 4-D case

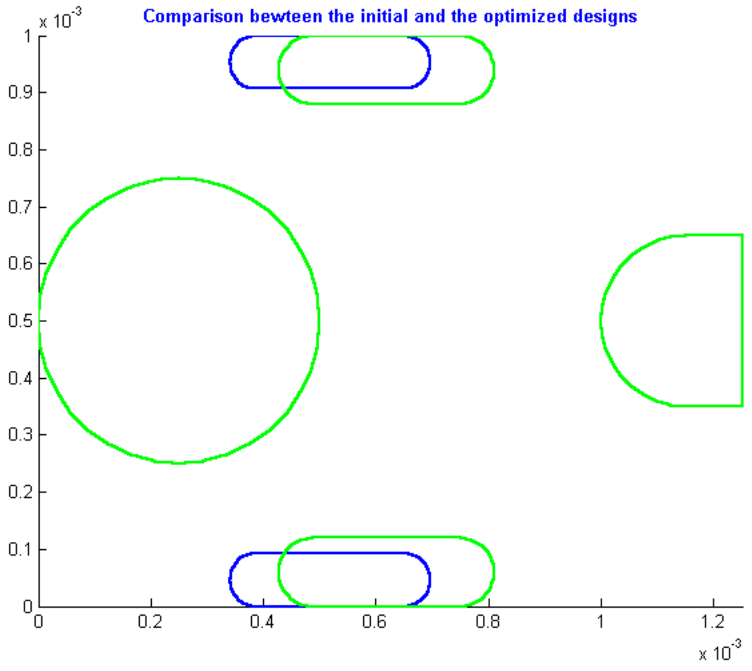


Fig. 7.12 Comparison between the initial design and the optimized design for the 4-D case

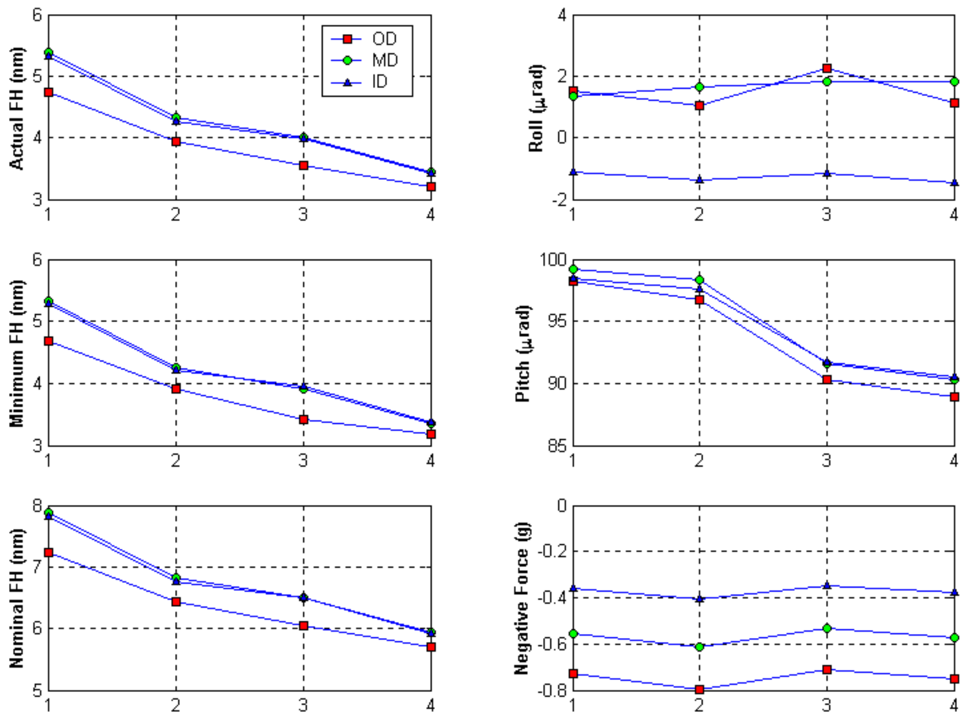


Fig. 7.13 Variations of the slider performance parameters for the 4-D case

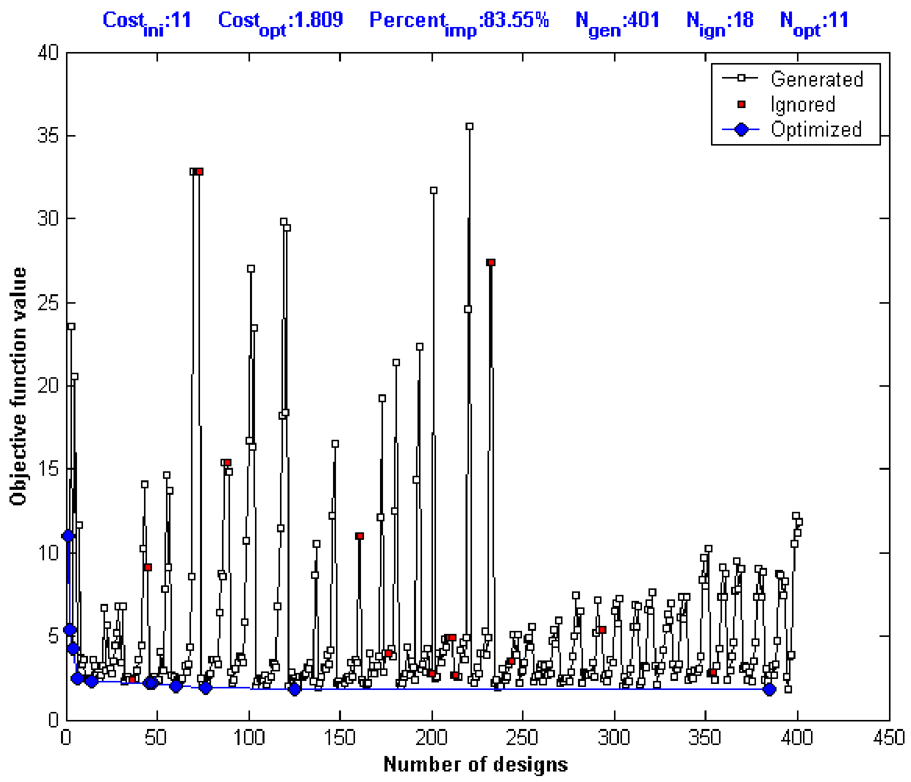


Fig. 7.14 Variation of the objective function value for the 2-D case

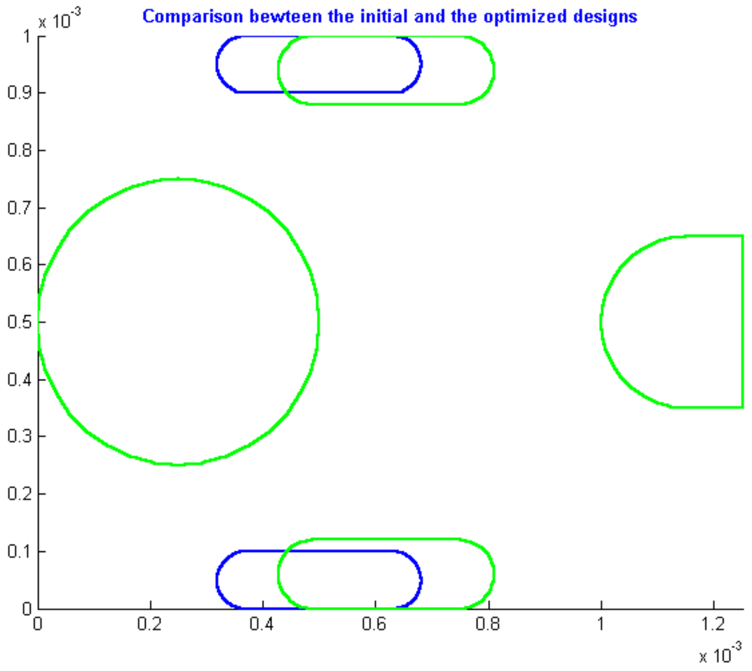


Fig. 7.15 Comparison between the initial design and the optimized design for the 2-D case

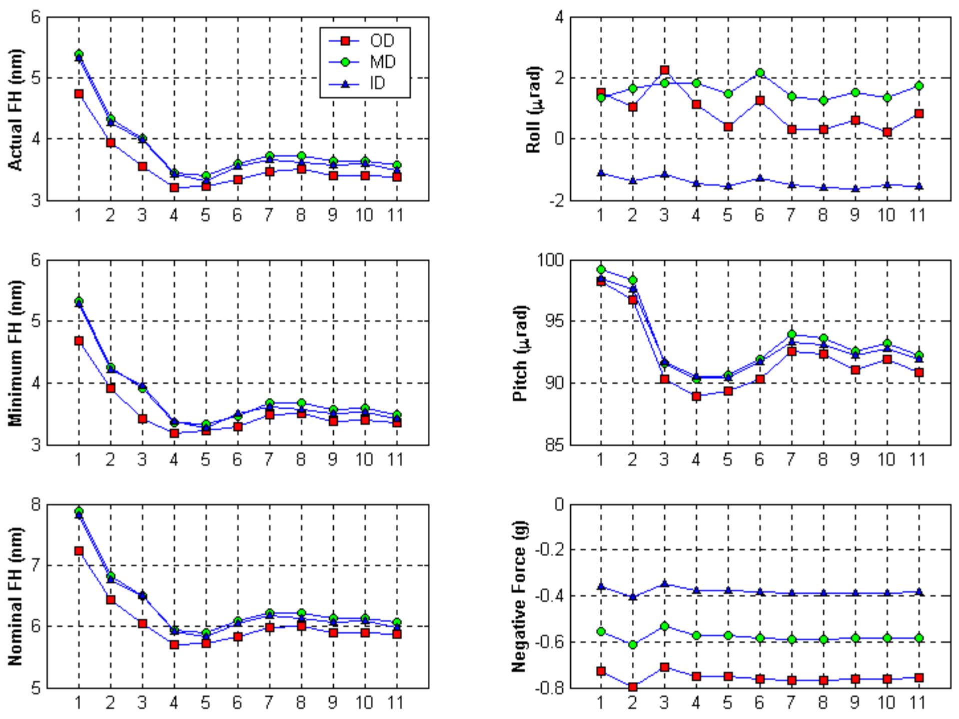


Fig. 7.16 Variations of the slider performance parameters for the 2-D case

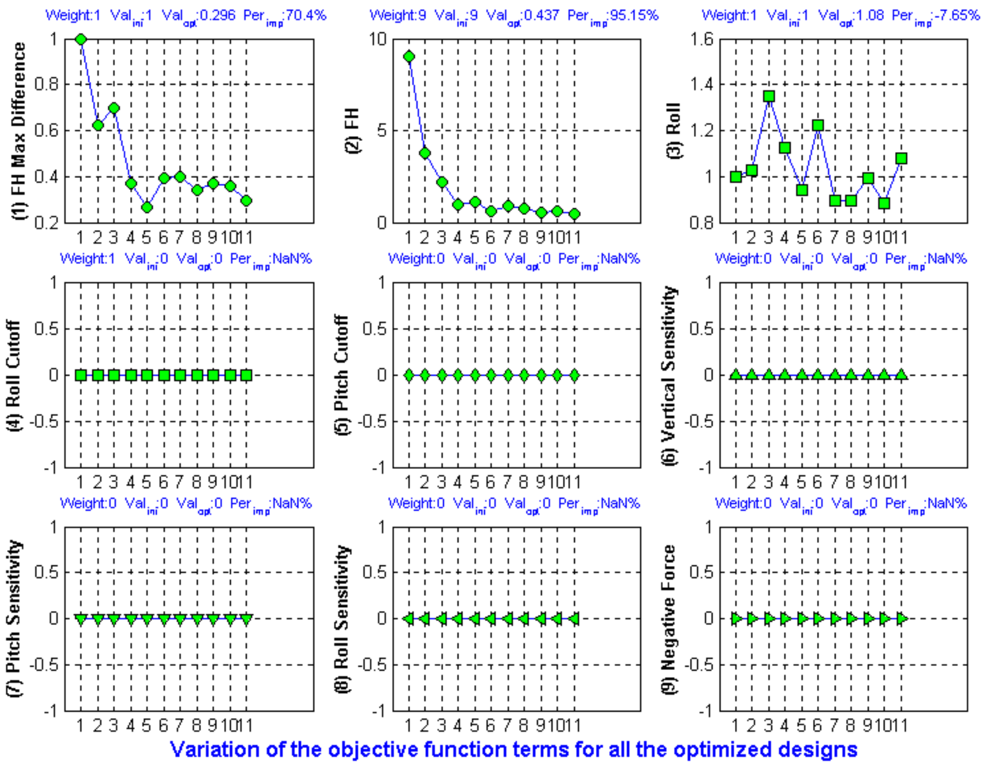


Fig. 7.17 Variations of the objective function terms for the 2-D case

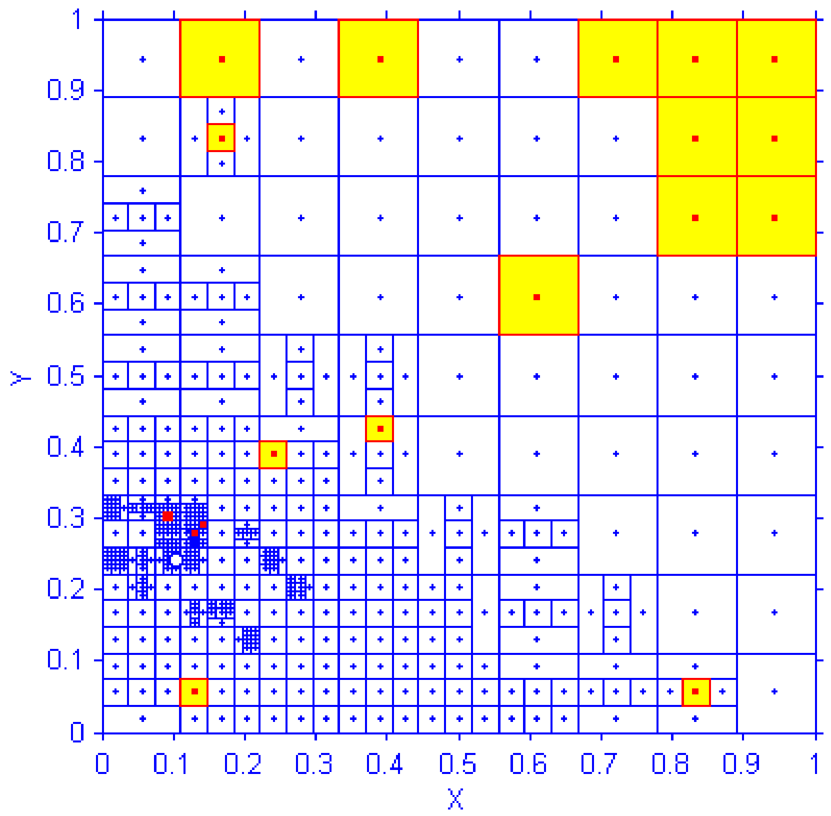


Fig. 7.18 Optimization results for the 2-D case

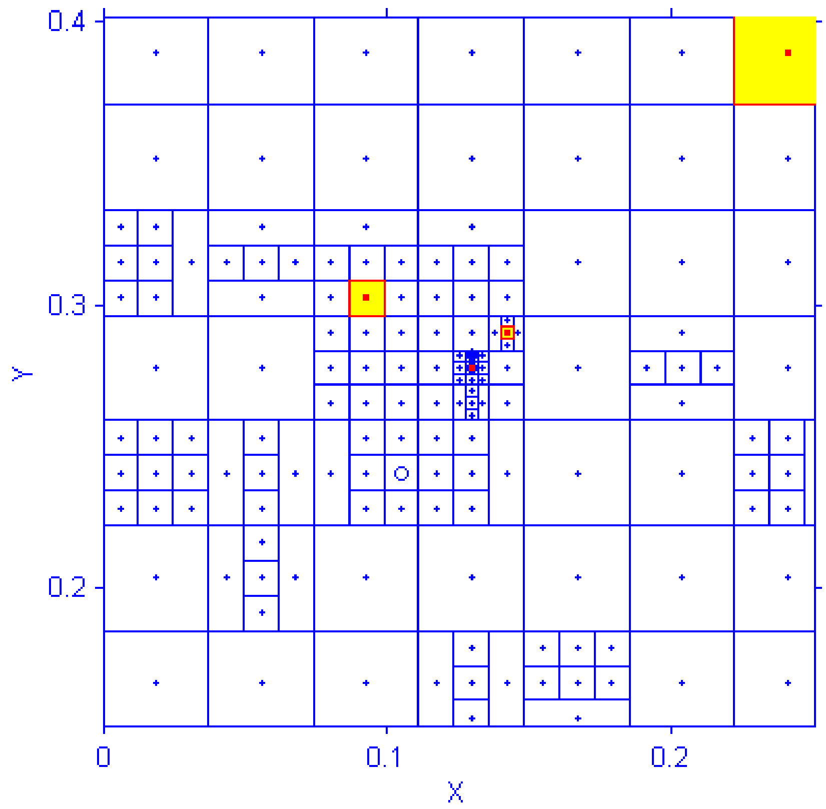


Fig. 7.19 Local zoom-in around the best point

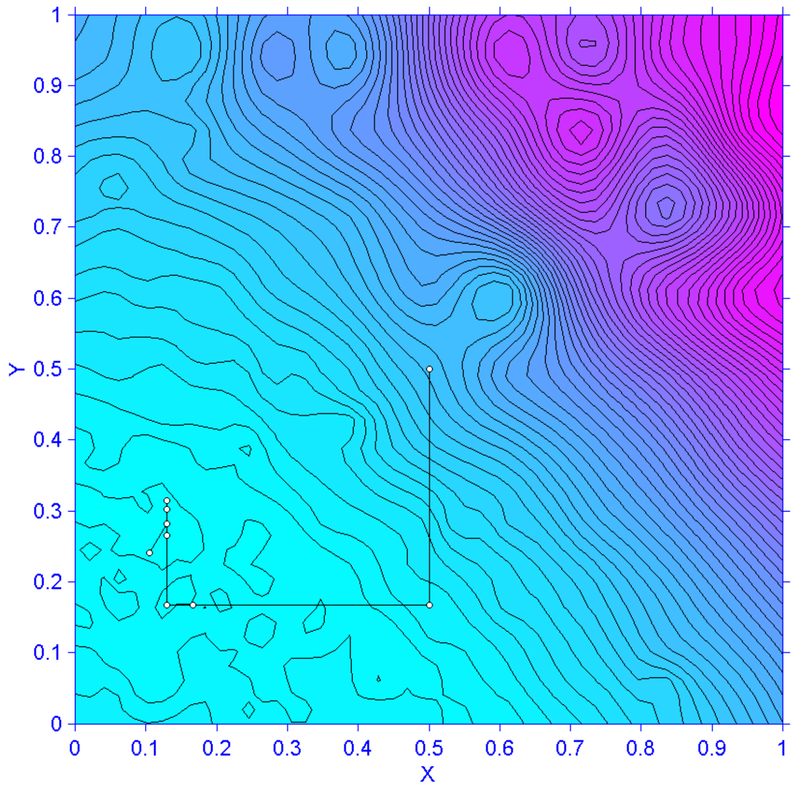


Fig. 7.20 Contour lines in the search space for the 2-D case

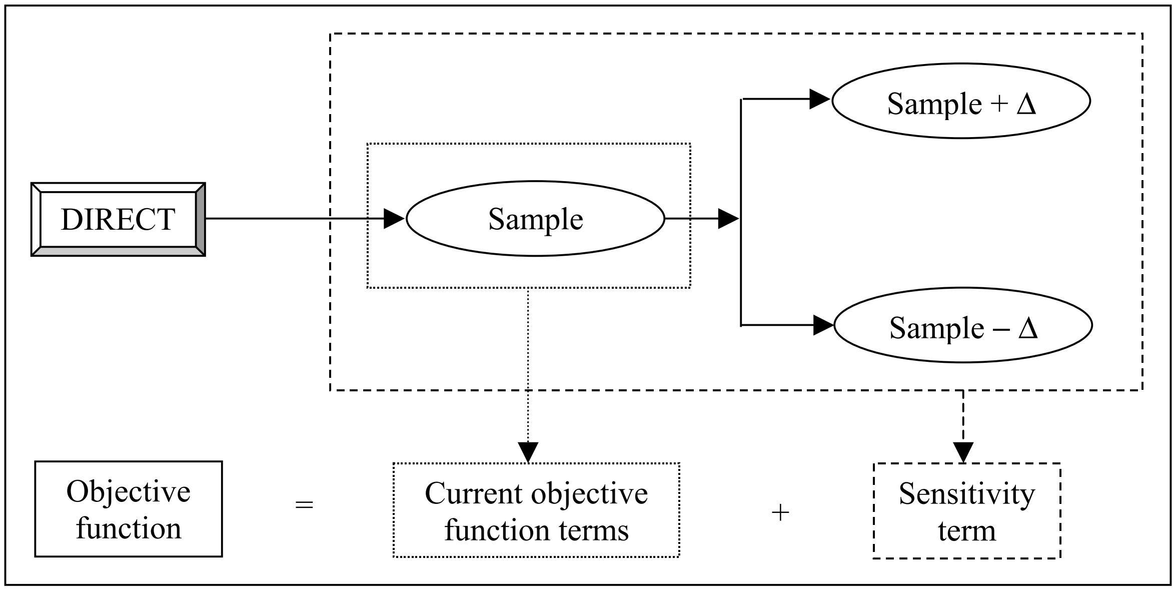


Fig. 7.21 Illustration of the sensitivity optimization process

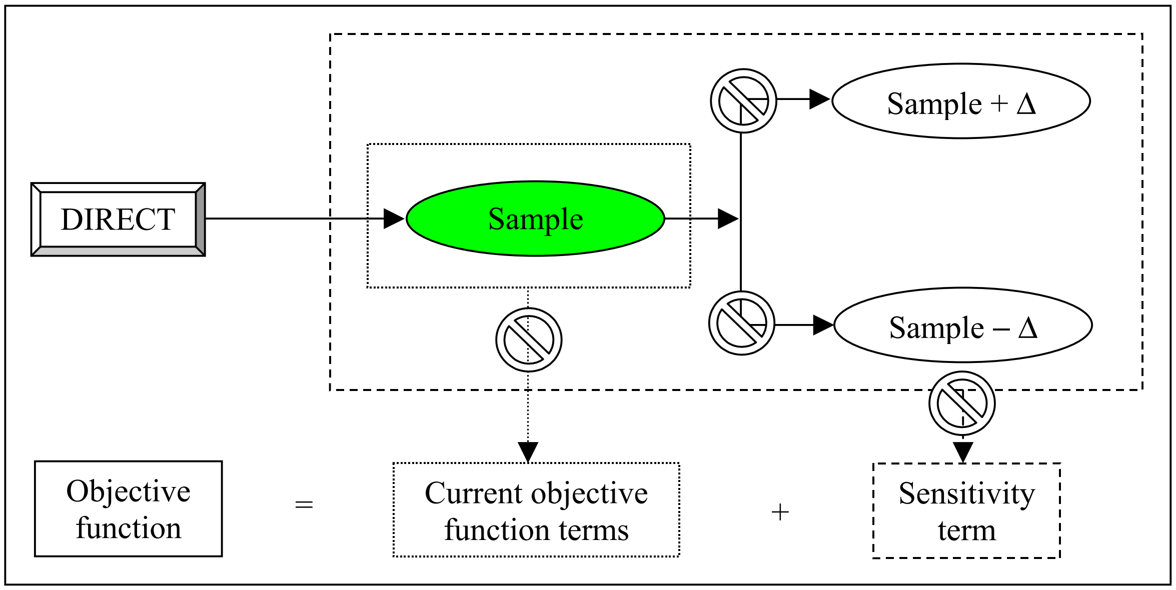


Fig. 7.22 Ways to improve the efficiency: Hidden constraints (1)

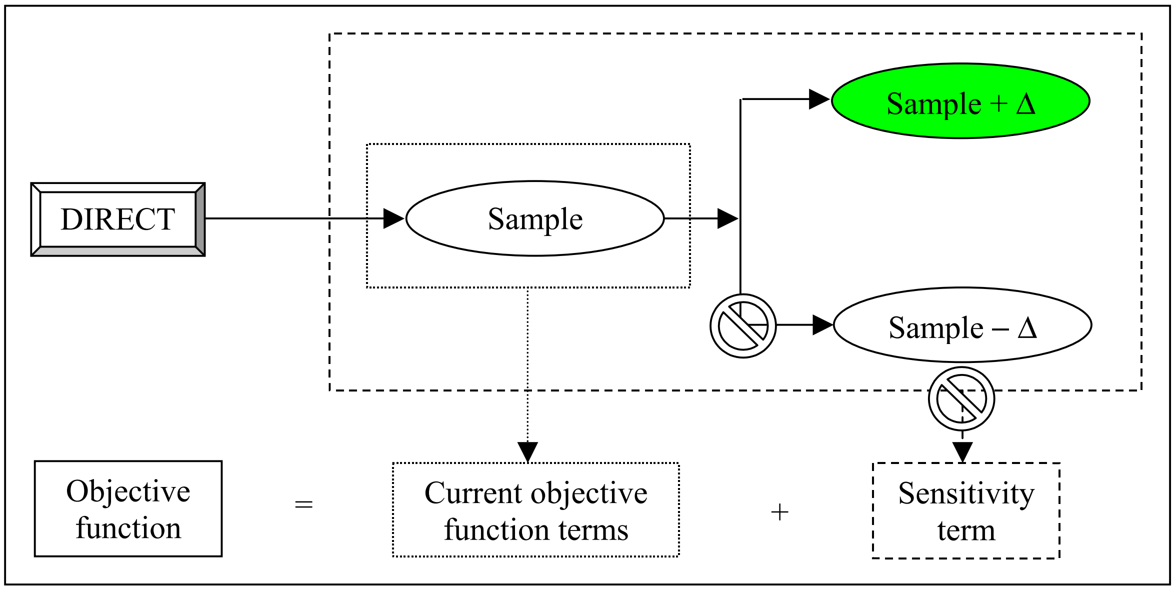


Fig. 7.23 Ways to improve the efficiency: Hidden constraints (2)

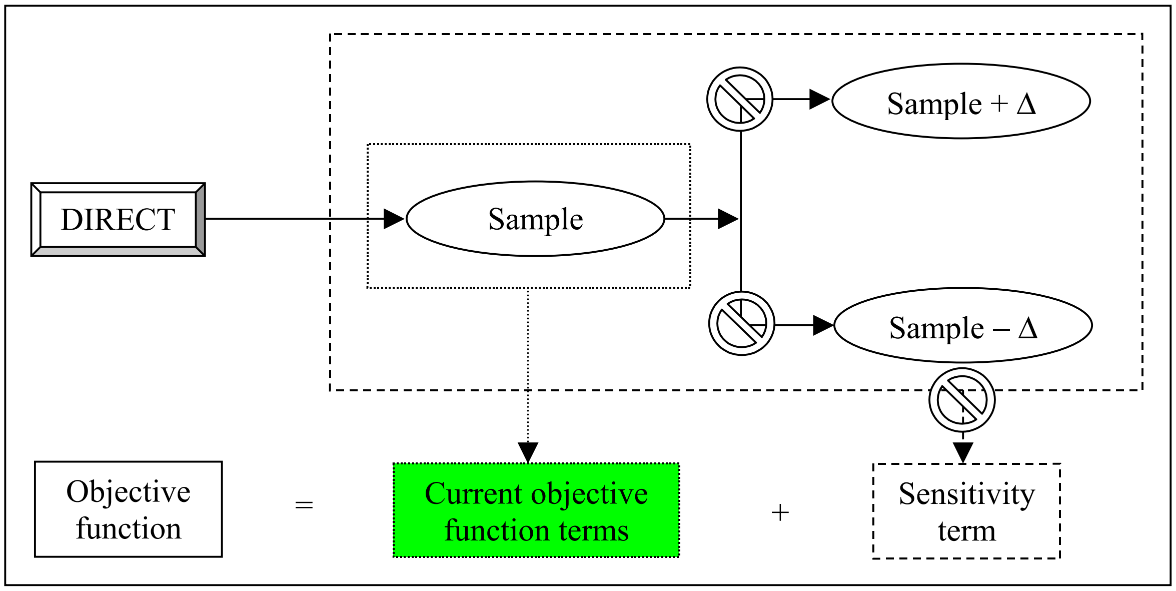


Fig. 7.24 Ways to improve the efficiency: First part of the objective function

Chapter 8

SUMMARY AND FUTURE WORK

8.1 SUMMARY

This dissertation focuses on the development and application of global optimization techniques to the problem of hard disk drive slider air bearing design.

Chapter 1 provides an introduction to slider design and some comments on the historical development of the hard drive. The continuous increase of the areal density is shown. And the relationship between the head-to-media spacing and the areal density is demonstrated; showing that the performance of the slider becomes quite critical and requires tighter control of flying height uniformity, roll profile flatness, etc. To meet the increasingly rigorous multi-objective slider performance criteria modern optimization techniques are used to solve this strongly nonlinear problem. This chapter shows that high efficiency, simplicity and automaticity are the reasons to use optimization techniques when designing slider air bearing for HDD. This chapter also provides a general survey for optimization techniques. Some previous work on slider ABS optimization is also reviewed. Finally this chapter presents an overview of the dissertation.

Chapter 2 gives a detailed description of the Simulated Annealing optimization family, including the Standard Simulated Annealing (BA) algorithm, the Fast Simulated

Annealing (FA) algorithm and the more powerful Adaptive Simulated Annealing (ASA) algorithm. This chapter compares the performance of BA, FA and ASA by optimizing a certain slider ABS design. These three main members of the simulated annealing family are shown to produce similar optimized ABS designs with greatly improved performance, i.e. uniform flying heights around the target flying height, flat rolls and improved stiffness. This illustrates that simulated annealing algorithms are quite suitable for the optimization of the ABS designs. Among them, the ASA was found to be the most efficient and robust scheme due to its fastest cooling schedule and its unique adaptive re-annealing mechanism.

Chapter 3 provides an introduction to the DIRECT algorithm, which is a deterministic global optimization technique that is used to find the minimum of a Lipschitz continuous function without knowing the Lipschitz constant. During the 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 test functions, and it is also capable of finding multiple global minima, even for some extremely tough functions. The numerical experiments also verify that the DIRECT algorithm has a very fast convergence rate. For the slider ABS optimization case, the DIRECT algorithm was shown to produce an optimized ABS design with improved performance, i.e., uniform flying heights around the target flying height, flat rolls and improved stiffness, illustrating that the DIRECT algorithm is quite suitable for the optimization of ABS designs.

Chapter 4 reports on three locally biased variations of the standard DIRECT algorithm. These variations are proposed to further increase the standard DIRECT algorithm's convergence rate and thus improve its efficiency. We carried out numerical experiments using the DIRECT algorithm and its three locally biased variations, i.e., DIRECT-I (having fewer groups), DIRECT-II (having double partitions for the box containing the point with the lowest function value), and DIRECT-III (which combines these two measures). We also applied these three variations to slider ABS optimization and found 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 substantially reduce the time needed to find the global minimum points.

Chapter 5 reports on two modifications to the DIRECT algorithm: one to handle tolerance (minimum side lengths) and one to deal with hidden constraints. We also 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 is globally biased. The numerical results also show that the strategy adopted for dealing with hidden constraints is reasonable and effective. We then applied the modified DIRECT algorithm to 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 a fixed number of designs generated, and thus improve the efficiency of the DIRECT algorithm. In summary, these two modifications to the DIRECT algorithm improve its efficiency and make it more flexible.

Chapter 6 compares the two fundamentally different global optimization techniques: the stochastic ASA algorithm and the deterministic DIRECT algorithm. Both algorithms are powerful tools for dealing with strongly non-linear problems and they are quite robust. However, ASA generally needs to be “tuned up” in order to be successfully applied to a specific problem. Also, since ASA is a stochastic algorithm, different optimization results can be expected for a limited number of samples generated for any changes in initial conditions or algorithm parameters. DIRECT is a deterministic algorithm, and therefore there are no random factors involved in the optimization process. DIRECT does not need to be “tuned up”. Thus, DIRECT is more convenient for practical usage. DIRECT also has a much higher convergence rate than does ASA. Therefore, DIRECT can find the global minimum more quickly than ASA. This property is observed more clearly for higher dimensional problems and it has been verified by numerical simulations as well as in slider ABS optimization case studies. Also notice that it is quite easy for DIRECT to handle the manufacturing tolerance issue. In summary, we found the DIRECT algorithm clearly outperforms the ASA algorithm in our test cases, and we consider DIRECT more suitable for the slider ABS optimization than is ASA.

Chapter 7 addresses the definition of new geometric constraints, in which we define not only constraint points, but also constraint rails and constraint lines. The constraint rails can be translated in either the length or width direction, rotated with respect to a fixed point and expanded or shrunk proportionally. The constraint line can be translated in either the length or width direction, rotated with respect to a fixed point and extended or contracted along its length direction. To maintain a symmetrical slider ABS design and fixed local geometric shapes, we also define symmetrical and relative constraints for constraint rails and constraint lines. We then applied those new geometric constraints to slider ABS optimization problems, investigating a 4-D case and a 2-D case. Although we found satisfactory optimization results in both cases, a better-optimized slider ABS design was found in the 2-D case by redefining the constraints. This verifies the importance of properly defining constraints. In conclusion, the new geometric constraints make the slider ABS optimization much easier by enabling users to explore a much wider range of constraints in practical ABS optimization problems. We also discussed the issue of slider ABS sensitivity optimization. The definition of the new geometric constraints will make sensitivity optimization much easier. We proposed the basic idea of how to perform the sensitivity optimization. Some possible ways to improve the efficiency for the sensitivity optimization have also been proposed.

Two new versions of the CML Air Bearing Optimization Program have been developed. Version 2.0 was developed for the Simulated Annealing algorithms (including the Standard Boltzmann Simulated Annealing (BA), Fast Simulated Annealing (FA) and the more powerful Adaptive Simulated Annealing (ASA) algorithms). Version 3.0 was developed for the DIRECT algorithm. It includes three locally biased variations and the modified versions of the standard DIRECT algorithm. It also features the new geometric constraints. These two new CML Air Bearing Optimization Programs have been successfully applied to the ultra-low FH slider design and optimization problem for the Extremely High Density Recordings (EHDR) project of the National Storage Industry Consortium (NSIC).

In summary, two categories of global optimization techniques, i.e., the stochastic Simulated Annealing (SA) algorithm and the deterministic DIRECT (DIviding RECTangles) algorithm, have been investigated and applied to the problem of slider Air Bearing Surface (ABS) optimization. By using these global optimization techniques, we can design and optimize the slider ABS automatically according to the optimization goals. Through substantial numerical experiments and slider ABS optimization case studies, we found the DIRECT algorithm has a very fast convergence rate and it clearly outperforms the SA algorithm. We consider it more suitable for the slider ABS optimization problems.

8.2 FUTURE WORK

One very important future work is to develop a graphic user interface (GUI) for the CML air bearing optimization program version 2.0 and 3.0. It will make the ABS optimization an easier task for general users.

The future work might also include the slider dynamic performance optimization. As it is well known, the simulation of slider dynamic performance takes a much longer time than the simulation of the slider steady performance. This makes the task quite challenging. One way to reduce the optimization time is to implement parallel calculation.

The slider ABS sensitivity optimization is still ongoing, and much more work can be done in this area also.

Also, the new DIRECT optimization technique can be applied to other areas of the HDD such as the thermal and contamination area, and it can be applied to the optimization of other components of HDD such as the suspension.

Finally, it is always interesting to investigate new global optimization techniques and apply them to the slider ABS optimization problems.

REFERENCES

Bird, G., 1976, *Molecular Gas Dynamics*, Oxford: Clarendon Press.

Bogy, D., Wu, L., Zeng, Q-H and Zhu, H., 1999, "Air Bearing Designs for 100 Gbit/in^2", TRIB-Vol. 9, *Proceedings of the Symposium on Interface Technology Towards 100 Gbit/in^2*, ASME, pp. 11-16

Bogy, D., Fong, W., Thornton, B., Zhu, H., Gross, H. and Bhatia, C., 2002, "Some Tribology and Mechanics Issues for 100 Gbit/in^2 HDD", *Intermag 2002 Conference*, Amsterdam, The Netherlands.

Chen, S., Luk, B. L. and Liu, Y., 1998, "Application of adaptive simulated annealing to blind channel identification with HOC fitting", *Electronics Letters,* Vol. 34, No. 3, pp. 234-235.

Cohen, B., 1994, "Training synaptic delays in a recurrent neural network", M.S. Thesis, Tel-Aviv, Israel, Tel-Aviv University.

Daniel, E., Mee, C. and Clark, M., 1999, *Magnetic Recording, The First 100 Years*, IEEE, New York.

Desai, R. and Patil, R, 1996, “SALO: Combining Simulated Annealing and Local Optimization for Efficient Global Optimization”, *Proceedings of the 9th Florida AI Research Symposium (FLAIRS-’96)*, pp. 233-237.

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.

Gablonsky, J. M. and Kelley, C. T., 2000, “A locally-biased form of the DIRECT algorithm”, Technical Report CRSC-TR00-31, Center for Research in Scientific Computation, North Carolina State University.

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.

Geman, S. and Geman, D., 1984, “Stochastic relaxation, Gibbs distribution and the Bayesian restoration in images”, *IEEE Trans. Patt. Anal. Mac. Int.*, 6, pp. 721-741.

Huyer, W., Neumaier, A., 1999, “Global Optimization by Multilevel Coordinate Search”, *Journal of Global Optimization*, 14, pp. 331-355.

Ingber, L., 1989, “Very Fast Simulated Re-Annealing”, *J Math. Comput. Modeling*, Vol. 12, pp. 967-973.

Ingber, L., 1993, “Simulated annealing: Practice versus theory”, *J Math. Comput. Modeling*, 18, 11, pp. 29-57.

Ingber, L., 1996, “Adaptive Simulated Annealing (ASA): Lessons learned”, *Journal of Control and Cybernetics*, Vol. 25, pp. 33-54.

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.

Mee, C. and Daniel, E., 1990, *Magnetic Recording Handbook, Technology & Applications*, McGraw-Hill Publishing Company, New York.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E., 1953, “Equation of state calculations by fast computing machines”, *J. Chem. Phys.*, 21, pp. 1087-1092.

O’Hara, M., 1997, “Optimization of Hard Disk Drive Components”, Ph.D. Thesis, University of California at Berkeley.

O’Hara, M. and Bogy, D., 1997, “The CML Air Bearing Optimization Program Version 1.5”, Technical Report, Computer Mechanics Laboratory, University of California at Berkeley.

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

Rosen, B., 1992, “Function optimization based on advanced simulated annealing”, *IEEE Workshop on Physics and Computation –PhysComp ’92*, pp. 289-293.

Sen, M. and Stoffa, P., 1995, *Global Optimization Methods in Geophysical Inversion*, ELSEVIER, Amsterdam – Lausanne – New York – Oxford – Shannon – Tokyo.

Szu, H. and Hartley, R., 1987, “Fast simulated annealing”, *Phys. Lett.* A 122, 3-4, pp. 157-162.

Thompson, D., 2000, “The future of magnetic data storage technology”, *IBM J. Res. Develop.*, Vol. 44, No. 3, pp. 311-321.

Vanderplaats, G., 1984, *Numerical Optimization Techniques for Engineering Design with Applications*, McGraw – Hill Book Company.

Zhu, H. and Bogy, D., 2000, “Optimization of Slider Air Bearing Shapes using Variations of Simulated Annealing”, Technical Report 00-010, Computer Mechanics Laboratory, University of California at Berkeley.

Zhu, H. and Bogy, D., 2001, “DIRECT Algorithm and its Application to Slider Air Bearing Surfaces Optimization”, Technical Report 01-003, Computer Mechanics Laboratory, University of California at Berkeley.

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 01-007, Computer Mechanics Laboratory, University of California at Berkeley.

Zhu, H. and Bogy, D., 2001, “Modifications to the DIRECT Algorithm”, Technical Report 01-013, Computer Mechanics Laboratory, University of California at Berkeley.

Zhu, H. and Bogy, D., 2001, “The CML Air Bearing Optimization Program Version 2.0”, Technical Report 01-016, Computer Mechanics Laboratory, University of California at Berkeley.

Zhu, H. and Bogy, D., 2002, “DIRECT Algorithm and Its Application To Slider Air Bearing Surface Optimization”, *Intermag 2002 Conference*, Amsterdam, The Netherlands.

Zhu, H. and Bogy, D., 2002, “Hard disk drive air bearing design: Modified DIRECT algorithm and its application to slider Air Bearing Surface optimization”, abstract accepted by *6th International Tribology Conference (AUSTRIB ’02)*, Western Australia.

Appendix A

SAMPLING FROM A PRESCRIBED DISTRIBUTION

During the process of optimization, the generation of new design requires the generation of representative values of variables that are distributed in a prescribed manner. This is done through random numbers and is a key step in the implementation of simulated annealing algorithms. We now introduce the ways suggested by Bird (1976) to do the sampling from a prescribed distribution.

Here we assume *u* is a random number which is uniformly distributed between 0 and 1, i.e. $u \in U[0,1]$.

The distribution of the variable *x* can be described by a normalized probability density function *g(x)* such that the probability of a value of *x* lying between *x* and *x+dx* is given by *g(x)dx*.

If $x \in [a, b]$, then the total probability is

$$\int_a^b g(x)dx = 1 \quad .$$

Now let's define the cumulative distribution function *G(x)* as

$$G(x) = \int_a^x g(x)dx \quad .$$

Then we may invert the cumulative distribution function *G(x)* and get

$$x = x(G) \quad .$$

Next we can generate a random number *u* and set it equal to *G*. Then the representative value of *x* is given by

$$x = x(u) \quad .$$

This method is therefore referred as the *inverse-cumulative method*. The operation of this method is shown graphically in Fig. 1.

First let's consider a trivial example in which the *x* is uniformly distributed between *a* and *b*. For this case *g(x)* is a constant and we have

$$g(x) = 1/(b-a) \quad .$$

So, from the above we get

$$G(x) = \int_a^x g(x)dx = \frac{x-a}{b-a} \quad .$$

Inverting *G(x)*, we have

$$x = a + G \times (b-a) \quad .$$

Then let *G* be equal to a random number $u \in U[0,1]$ to get

$$x = a + u \times (b - a) \ .$$

For the case of the Adaptive Simulated Annealing algorithm (ASA), the probability density function is defined as

$$g_T(x) = \prod_{i=1}^{D} g_T^i(x^i) = \prod_{i=1}^{D} \frac{1}{2(|x^i| + T_i)\ln(1 + \frac{1}{T_i})} \quad ,$$

where T stands for the annealing temperature and D the dimension of parameter space. We can easily verify that

$$\int_{-1}^{1}\int_{-1}^{1}\cdots\int_{-1}^{1}\prod_{i=1}^{D} g_T^i(x^i)dx^1dx^2\cdots dx^D = 1 \ .$$

Its cumulative probability function is

$$G_T(x) = \int_{-1}^{x^1}\int_{-1}^{x^2}\cdots\int_{-1}^{x^D}\prod_{i=1}^{D} g_T^i(x^i)dx^1dx^2\cdots dx^D \equiv \prod_{i=1}^{D} G_T^i(x^i) \ ,$$

where

$$G_T^i(x^i) = \frac{1}{2} + \frac{\operatorname{sgn}(x^i)}{2}\frac{\ln(1 + \frac{|x^i|}{T_i})}{\ln(1 + \frac{1}{T_i})} \quad .$$

Then as before, we can generate a set of random numbers u^i from the uniform distribution $u^i \in U[0,1]$. After inverting the cumulative distribution function and letting

$$G_T^i(x^i) = u^i \ ,$$

we get

$$x^i = \operatorname{sgn}(u^i - \tfrac{1}{2})T_i[(1+\frac{1}{T_i})^{|2u^i-1|} - 1] \quad .$$

So far we have discussed the inverse-cumulative method and its applications. However, this method can be used only when it is possible to invert the cumulative distribution function *G(x)* to obtain an explicit function for *x*. But sometimes it's impossible to obtain the inverse cumulative distribution function, if for example, the probability density function is

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \quad ,$$

then

$$G(x) = \int_{-\infty}^{x} g(x)dx = \frac{1}{2} + \frac{1}{2} erf\left(\frac{x}{\sqrt{2}}\right) \quad .$$

This expression can't be inverted to give *x* in terms of *G* and thus the inverse-cumulative method fails.

The general alternative is to apply the *acceptance-rejection method*. In order to make direct use of the random number *u*, the probability function is normalized by dividing it by its maximum value g_{max} to give

$$\hat{g}(x) = \frac{g(x)}{g_{\max}} \quad .$$

A value of x is then chosen at random on the basis of x being uniformly distributed between its limits, i.e.

$$x = a + u \times (b - a) \quad .$$

The function $\hat{g}(x)$ is then calculated for this value of x and a second random number is generated. The value of x is then either accepted or rejected according to whether $\hat{g}(x)$ is greater or less than this second random number. This procedure is repeated until a value of x is accepted. Since the random number u is uniformly distributed between 0 and 1, the probability of a particular value of x being accepted is clearly proportional to $\hat{g}(x)$ and the accepted values conform to this distribution.

For the Boltzmann Annealing (BA) and Fast Annealing (FA) of the simulated annealing algorithm family, when the inverse-cumulative method fails because we can't invert their cumulative distribution functions, we can use the acceptance-rejection method.

For the Boltzmann Annealing, the probability density function is

$$g_T(x) = (2\pi T)^{\frac{-D}{2}} e^{\frac{-x^2}{2T}} \quad .$$

So

$$\hat{g}_T(x) = \frac{g_T(x)}{g_{T\max}} = e^{\frac{-x^2}{2T}} \quad .$$

Then the first random number $u_1 \in U[0,1]$ can be generated. If we denote η_0 as the previous point and η as the new point, and define the range of this point as *[a, b]*, then

$$\eta = a + u_1(b-a) \quad .$$

From this value of η we can obtain *x* since *x* is defined as

$$x \equiv \frac{\eta - \eta_0}{b-a} \in [-1,1] \quad .$$

Then the value of $\hat{g}_T(x)$ follows from *x*. Next the second random number $u_2 \in U[0,1]$ is generated and compared with the value of $\hat{g}_T(x)$. If $\hat{g}_T(x)$ is greater than u_2 , the *x* value gets accepted. Otherwise the *x* value is rejected and the above procedure is repeated until a value of *x* is accepted.

For Fast Annealing, the probability density function is

$$g_T(x) = \frac{T}{\left(x^2 + T^2\right)^{\frac{D+1}{2}}} \quad .$$

By following the above procedure we again get the value of *x* according to this prescribed probability density distribution.

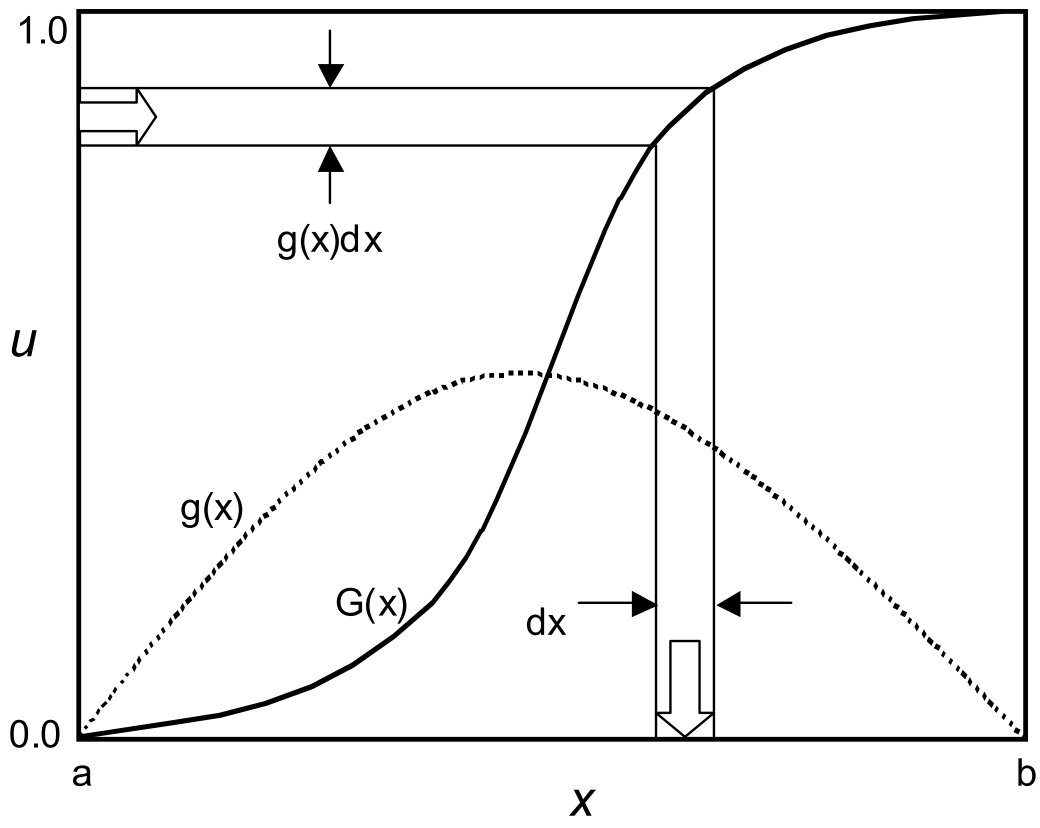


Fig. 1 Relationship between the typical normalized probability density function g(x) and the cumulative distribution function G(x)

Appendix B

GRAHAM'S SCAN

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. 1.

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

Figures 2 ~ 4 demonstrate the basic processes to obtain the convex hull for a given set of data points.

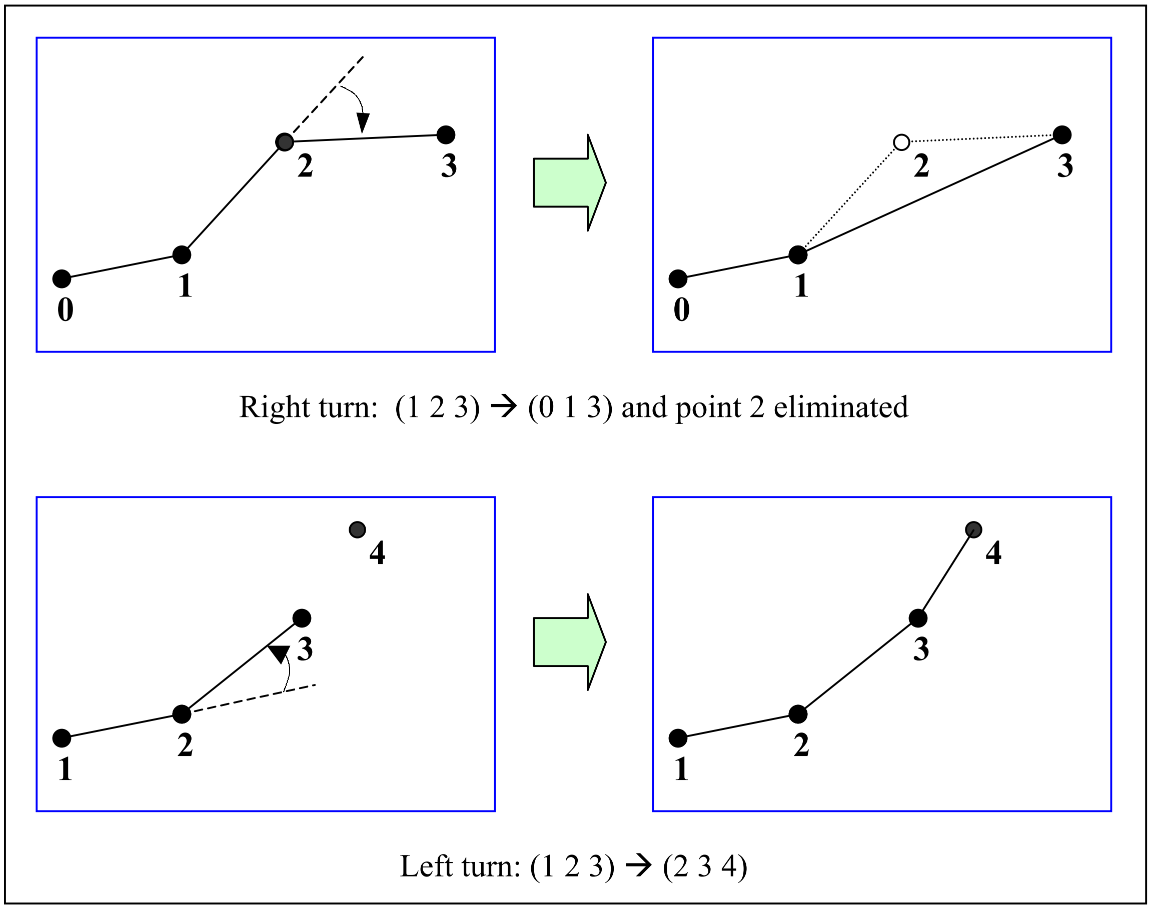

0
1
2
3
0
1
2
3
Right turn: (1 2 3) → (0 1 3) and point 2 eliminated
4
3
1
2
4
3
2
1
Left turn: (1 2 3) → (2 3 4)

Fig. 1 Illustration of "Right turn" and "Left turn"

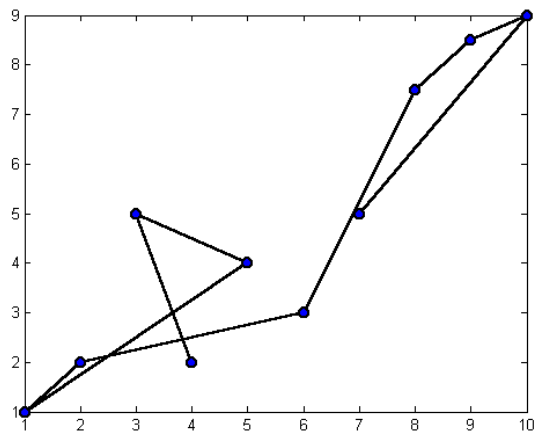


Fig. 2 Initial set of data points

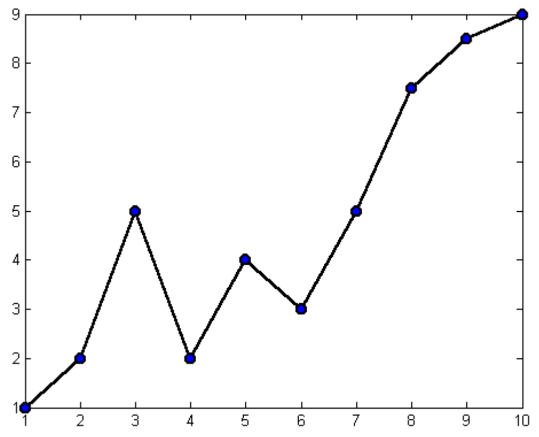


Fig. 3 Results after sorting

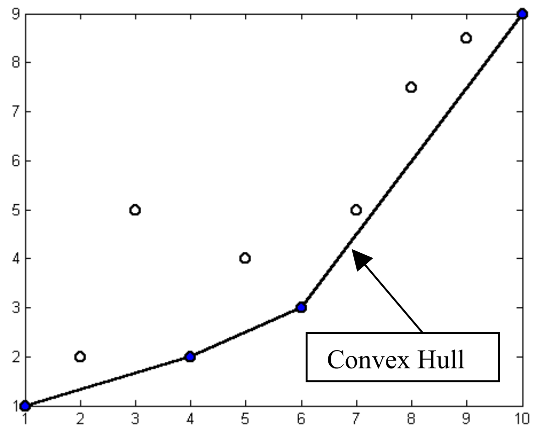


Fig. 4 Results of Graham's scan