A Parallel Response Surface Method

by
Kuang-Yen Liu
Advisors
Weichung Wang

Institute of Statistics,
National University of Kaohsiung
Kaohsiung, Taiwan 811 R.O.C.
July 2005
平行反應曲面法

指導教授: 王偉仲 博士
國立高雄大學應用數學系

學生: 劉光晏
國立高雄大學統計學研究所

摘要

本論文考慮一個在電腦模擬實驗上具有多重極值的最佳化問題。為了要解決這個
問題，在這提出了一個新的演算法。基本上這個新方法是利用在實驗區域中任意選
取數個初始值後，同時進行數個反應曲面法，並輔以平行計算的技巧來節省計算成
本。

關鍵字: 反應曲面法、平行計算、電腦模擬實驗
A Parallel Response Surface Method

Advisor: Dr. Weichung Wang  
Department of Applied Mathematics  
National University of Kaohsiung

Student: Kuang-Yen Liu  
Institute of Statistics  
National University of Kaohsiung

Abstract

In this thesis an optimization problems for finding multiple extremes in computer experiments (Sacks et al., 1989, and Koehler and Owen, 1996) are considered. To solve this kind of problems, a novel method is proposed. Basically the idea is to run multiple Response Surface Method (RSM) in the parallel environment at the same time.

Keywords. Response surface methods, parallel computing, computational experiment.
1 Introduction

In this thesis, we consider the problems of finding extreme points for unknown response function $f$ in computer experiments. In Sacks et al. (1998), and Koehler and Owen (1996), these optimization problems can be formulated as

$$\max y = f(x_1, \ldots, x_k),$$

s.t. $x = (x_1, \ldots, x_k) \in X,$

where $y \in \mathbb{R}$ is response variable, $x = (x_1, \ldots, x_k)$ is the vector of $k$ factors in $\mathbb{R}^k$, and $X \subset \mathbb{R}^k$ is the experimental region. Here $y = f(x)$ represents the result of the corresponding computer experiment at the experiment point $x$, and the response function $f$ may contain multiple extremes. Here we define the response surface as

$$S_X = \{(x, f(x)) | x \in X\}.$$  

When response function is unknown on very complicated, response surface method (RSM), proposed by Box and Wilson (1951), is a popular and powerful statistical tool for solving this kind of optimization problem with only one extreme. Even though in RSM the response value, $y$, is modeled as

$$y = f(x) + \epsilon,$$

RSM is still concerned on functional relationship between $y$ and $x$, i.e. $E(y) = f(x)$. In RSM, the central composite designs are used for sampling the experimental points, and the first- and the second-order polynomial models are applied for surface approximation. However, in computer experiment, there is no replication. Hence the model of $y$ should be noiseless model, i.e. $y = f(x)$. In Chen et al. (2006), they modified RSM for computer experiments, and their modification is called RSM-CE. Basically in stead of using lack-of-fit statistics for detecting the surface curvature in RSM, the curvature of response surface is checked by using the division of coefficients of a partial second-order polynomial model.

Since RSM is an simple iterative procedure to find a local extreme point, for the case of multiple extremes, an intuitive idea is to run several RSMs from different initial points in $X$. To run several RSMs efficiently, we propose an method to handle these RSMs simultaneously. Basically, by sharing the information with other RSMs, our new algorithm should provide a more efficient way to handle RSMs at the same time.
Hence in this thesis, we propose an algorithm for running multiple RSMs with parallel computing techniques.

This thesis is organized as follows. In Section 2, we give a brief introduction of RSM and RSM-CE. Then our algorithm is proposed in Section 3.1 and the details of our algorithm are described in Section 3.2 to 3.4. In Section 4, using the information we propose several methods to get a better sketch of response surface. Then the numerical results will be shown in Section 5. Finally, Section 6 concludes the thesis.

2 Response surface method

Response surface methodology (RSM) is a combination of mathematical and statistical techniques that are useful for analysis of problems in which a response is influenced by several variables and the goal of RSM is to optimize the response. For example, assume there is an response function \( f \) that has \( k \) input variables \( x_i, \ldots, x_k \), and an output variable \( y \) is modeled as

\[
y = f(x) + \epsilon,
\]

where \( \epsilon \) is a white noise. We denote \( S \) as the surface formed by function without noise called response surface, i.e., \( S = E(y) = f(x) \). In most RSM problems the form of relationship between input variables and output variables is unknown. Thus, the first step in RSM is to find a suitable approximation for the true function \( f \). Usually we use low-order polynomial models. If the response is well modeled by a linear function of input variables, then this approximating function is called the first-order polynomial model,

\[
y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \epsilon.
\]

If there is a curvature, then a polynomial of higher degree must be used, such as the second-order polynomial model,

\[
y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i<j}^{k} \beta_{ij} x_i x_j + \epsilon.
\]

Of course, it is not always that a polynomial model can be a reasonable approximation of true function, but for a relatively small area they usually work fine.

RSM is a sequential procedure. When we start at a point on response surface that may be remote from optimum, our goal is to lead to the experimenter rapidly and
efficiency along a path of improvement toward the optimum. By using a full $2^k$ design and repeating measures on center point, the first-order polynomial model is used for fitting the true function, i.e. $\hat{y} = \hat{\beta}_0 + \sum_{i=1}^{k} \hat{\beta}_i x_i$, where $\hat{\beta}_i$ is least square estimation of $\beta_i$, and $\hat{y}$ is the prediction of $y$. The contour of fitted surface is a series of parallel lines. Thus, the direction of steepest ascent is the direction in which output $y$ increases most rapidly. Then the direction of steepest ascent is $(\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_k)^T$. Once we had the direction of steepest ascent, we can process step operation. It forms a path of steepest ascent by the direction of steepest ascent and a given step size. Then we compute the point along the path until no further improvement of output. At this time, a new first-order polynomial model will be fitted, and a new path of steepest ascent will be determined. Repeating above operation, RSM will lead us to vicinity of extreme point. Before exploring along the path of steepest ascent, the adequacy of first-order polynomial model should be investigated. In ordinary experiment design, usually the lack-of-fit statistic is used to determine if we should use the second-order polynomial model to replace the first-order polynomial model. When the second-order polynomial model is chosen, we need more runs to estimate the coefficients in the second-order polynomial model. Thus we add axial runs for estimating these coefficients.

In this thesis, the computer experiment is considered. That is no repeat measure for each experiment point. Thus, the lack-of-fit statistic is unable to compute. So instead of RSM Chen et al. (2006) proposed a RSM for computer experiments and it is called RSM-CE. In face, RSM-CE can be treated as a modified RSM for computer experiments. In RSM-CE, the lack-of-fit statistic is replaced by the division of coefficients of the partial second-order polynomial model. For example, when $k = 2$ based on $2^k$ factorial design and one center point, the partial second-order polynomial model is chosen for fitting the surface, i.e. 

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_{11} x_1^2.$$ 

Then the surface curvature is checked by the value of 

$$SC = \frac{|\hat{\beta}_1| + |\hat{\beta}_2|}{|\hat{\beta}_{11}| + |\hat{\beta}_{12}|},$$

where the $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_{11},$ and $\hat{\beta}_{12}$ are the estimations of corresponding coefficients of the partial second-order polynomial model. When SC is large, the surface curvature should do not exist, and then only the first-order polynomial model is used here. Otherwise, the second-order polynomial model will be employed for local surface approximation.
RSM-CE works quite well in the examples of Chen et al. (2006).

3 Parallel response surface method

RSM have many successful applications in many fields, and it works well on finding a extreme point. One of successful applications of RSM in computer experiments is presented in Steffen (2002). However, RSM also has its limits. One of them is that RSM can only guarantee to find a local extreme point. But when a response surface has multiple extremes, RSM only can find one of these extreme points for us. To solve this problem, running multiple RSMs on different initial points should be a good way to do. But it may cause the waste of the computing resources. For example, sometimes the paths of different RSMs may overlap and join into the same path. Another problem for RSM is the saddle point. When RSM reaches the saddle point, it would be stuck and unable to move again.

3.1 Master-client parallel model

Figure 1: A response surface with two extreme points.

The response surfaces considered contain multiple extremes. Figure 1. is one example of the response surface with two extremes. Since RSM is efficient to identify one
local extreme, if we want to allocate all the extreme points, then one intuitive idea is to repeat RSM for several times with different initial points. Of course, we can run those RSMs sequentially. But if we have more computing resources, we might apply parallel computing method on RSM. Because we believe that parallel computing techniques might help us to process those RSMs simultaneously and save our computing cost. Here we choose to use master-client architecture to handle multiple RSMs. It means that one computer is the master for controlling whole process and recording the result of each RSM, and the other computers are the clients for running RSMs. We hope that based on this architecture, the clients could share the information to each others, and could use information to improve the efficiency for finding multiple extremes.

![Master-client architecture](image)

Figure 2: Master-client architecture.

### 3.2 PRSM algorithm

Here we propose a Parallel Response Surface Method (PRSM) algorithm, which is able to parallelly run several RSMs. Generally, the outline of PRSM algorithm is in the following:

**If** is master **then**

1) send initial parameter to client.

**repeat** following step 2 and 3 until all client finish and no more initial points

2) receive information from any client.

**Select case:**

2a) function value request.

2b) RSM modeling point return, direction of RSM model.
2c) PRSM procedure request return (steepest ascent request,...etc)
3) analysis the information from clients.
   3a) check function value been computed of not. if already computed send
   the value back, else ask client to compute function value and record.
   3b) record the path information (direction,..etc ).
   3c) check path should perform PRSM operation (join, fork). if need send
   the corresponding echo to client.
   4) send PRSM stop message to clients
Else
   repeat until master send a PRSM stop message.
   1) get initial parameter from master.
   2) run RSM and send real time information to master.
      2a) when compute function, send function value request to master.
      2b) when enter model operation, send the direction and a request for steep-
      est ascent.
      3c) when RSM end, send master RSM finish message.
   4) receive the echo from master.
   5) perform the instruction from master.
End

3.3 Information Sharing

According to above PRSM algorithm, how to share information is important for
improving the computing efficiency. There are two kind of information we are interested
in. One is the function values of experiment points and the other one is the paths of
RSMs.

3.3.1 Sharing the function value

When PRSM is processing, each client computes response value for some experiment
points. Sometimes if the response values of experiment points have been computed by
other clients, then it is not necessary to recompute this response value again. Figure.
3 is an example. To avoid computing a response twice, we need to share information
of explored points between client. In practice, the probability that two RSM paths
meet at the same experiment point is very low. So we relax the restriction to be that
when the next experiment point is close to the explored point, we will not compute this response. Please see Figure 4. But with more and more explored points, the cost of computing distances between all points will also be large. To reduce this computing cost, we divide the search area into many small blocks and define the points have the same function value when they are in the same block. Based on this idea, those blocks can form a grid for the experimented region. So it should be useful to solve the problem of some experiments that the input variables might be discrete. In those cases, we can consider the experiment points are the center of block.

Due to the grid of experimented region, there could be two way to select the path of steepest ascent: 1) Choosing a closest grid direction to replace the true direction of steepest ascent and 2) Choosing a closest grid point to true direction of steepest
ascent. Please see Figure 5 and 6 for illustration. These two methods have their own advantage. By choosing the suitable step size, the first strategy can produce a path that all points are on the grid, and the second strategy can make the path direction close to the steepest ascent direction. For smooth response surface choosing closest grid point will provide the better performance.

3.3.2 Share the path information

Besides the information of response values, the paths of RSMs also provide the information of response surface. To use these information, we have two strategies: join and fork. When some RSM procedures might go to the same extreme point, we should
join them into single path to save computing cost. When the path might lead two or more extreme points, we should fork it into different paths which are toward to different extreme point.

3.2.2.1 Join

Starting multiple RSMs from different initial points, some of them may be toward to the same extreme. To save computing resource, we should join those paths into one single path, and let only one client to continue that path. By that way we can free up other client to explore more area.

We first consider the case that two paths meet at same experiment point. Since the computer experiment considered here, both of them will get the same direction. Thus the second path will joint with the first path.

![Diagram of path joining](image)

Figure 7: Join two paths if two paths do model operation at the same point.

Following the same ideas we discuss in Section 3.3.1, we join two paths when the corresponding RSMs explore the next experiment points which are in a small neighborhood, i.e. distance between two points $< r_{\text{join}}$. Figure 8 is an illustration. However, it might exist the risk for this operation. To reduce the risk, we can still fit the steepest ascent direction for each RSM procedure, if directions are close to each other then we would join paths. In practice, we usually skip this kind of check because of smooth assumption of $f$. Similarly we can also apply the idea of block to reduce the cost for
computing distances between experiment points.

The another situation we would join two path is that two paths cross by with a small angle. But this operation might exist the risk because two path could lead to different local extreme point, even if they cross by with small angle. We will show an example in section 5.1.

3.2.2.1 Fork

Here we want to run the multiple RSMs to reach as many extreme points as possible. So we use fork operation to make a RSM can have the secondary possible direction. For solving the problem that the second-order polynomial model is unable to produce a steepest ascent direction, we use the direct search (Kolda et al., 2003) at those points. Based on this idea, we add the fork operation when we decide to choose the second-order polynomial model as our surrogate model. Here we choose grid directions had largest function value be possible direction to fork. If the angle between this direction and steepest ascent direction are large enough, we will do fork operation. Please see Figure 9 for illustration.
3.4 Adding new initial point

We can add more initial points to improve the efficiency of searching optimum, because we might use those new initial point to help in searching new extreme point. Usually we use uniform design to sample initial points which is proposed by Fang et al. (2002a, b). Because of the limit number of clients, we can only choose few initial points. To solve this problem, we let a client to start a new RSM after it finishes current job. By this way, there is no limit on the number of the initial points. We will discuss three kind of strategies to add new initial points: I. Choose uniform design with more points. II. Use directions of paths to locate area that might have extreme points. III. Explore the area that we have less information.

3.4.1 Adding uniform design point

At the beginning of our algorithm, we would choose more initial points according to a uniform design. Once a client finish its current job, this client would start a new RSM procedure from a new initial point.

3.4.2 Using paths’ directions

Since we run multiple RSMs simultaneously, sometimes some RSM procedures will be toward to the same area. If we can put some initial points in this area, it would reach the extreme point faster. Figure. 10 is an illustration. To apply this idea, we need to check the directions of each RSM, and to see if they toward to the same area. Of course, it will take a lot of computing cost. So we still can use the skill that we used before. That is by counting how may paths’ directions extended line thought a block,
Figure 10: Paths’ directions are pointing toward the top.

we can find which block might be a good choice for adding new initial point. Please see Figure. 11.

Figure 11: Counting how many extended line passing through a block (the number is how many line pass though current block).

3.4.3 Exploring unknown area

We hope our algorithm to find as more extreme points as possible, but each RSM can only guarantee to find a local extreme point. So we should make the initial point can fill the whole experimental region. It is also the reason why we choose uniform design.
for sampling initial point. If we have additional resources, it might be good to put some extra initial point on those unexplored area. For this resource, it is reasonable to choice new initial points which are far from the current experiment points unexplored. A way to avoid computing distance between points is dividing the search area into many little blocks, and then randomly choose a initial point in blocks which have less experiment points.

4 Estimation of trend

From PRSM process, we explore a lot experiment points in the experimental region. Since response function is assumed to be unknown, we would like to use these response values to give us a rough idea about this unknown function. Hence the key point is how to link those response values together. Three methods are proposed.

4.1 Direct fitting

Since we usually assume that the true response function is smooth then the first approach is to fit these response value by a well-known statistical method. For example, the polynomial model fitting or kriging method are possible choices.

4.2 Fitting on modified data

The data we collected is not randomly spreading over the region. RSM makes the points form many paths over the region. How to use the path’s information is what we considered. Here we add extra point to show path’s information. We use two way to add extra points.

First we consider a single path. A path is made by a series of points, but how the surface between one path point to the next path point. We assumed the surface between points and neighbors on the same path is a straight line. So we can add extra point between on path, and we called those points ”path points”. The function value of path points is given by interpolation. And then use we can use any surface fitting method on modified data. Because of the extra point, the fitted surface will tend to approach the path line.

Secondly we can add extra points orthogonal to path’s direction. Because the steepest ascent path and contour line are tending orthogonal. So we can assume a
Figure 12: Add the "path points".

short contour line on the points and the value is given by the corresponding path point, and we call this kind of extra points "contour points".

Figure 13: Add the "contour points".

4.3 Extrapolation

Another way to approximate the true surface is to interpolate each grid point. Basically the response values are interpolated by weighted sum of other well-known response value. There are many ways for choosing the weights. Here we choose these weights are chose to be proportional to the inverse of the distance between the experiment points, i.e.

\[ \hat{y}_j = \sum_{i=1}^{n} \frac{1/r_{ij}}{\sum_{k=1}^{n} 1/r_{kj}} y_i, \]

where \( r_{ij} \) is the distance between \( y_i \) and \( y_j \). Usually we believe that is reasonable to only interpolate in a local neighborhood. That is

\[ \hat{y}_j = \sum_{i \in I} \frac{1/r_{ij}}{\sum_{k \in I} 1/r_{kj}} y_i \]

, where \( I \) is the index set that distance between \( y_i \) and \( y_j \) less than \( r \).
5 Numerical experiment

Here we demonstrate several experiments for showing the performance of PRSM. Those experiments are processed on parallel computer (using one master and eight clients), and the grid version of RSM-CE algorithm is used here for PRSM. First we display the join and fork operations in PRSM, and then PRSM is applied to solve three optimization problems. Finally the trend experiments for three problems are displayed in the last subsection.

5.1 Functional demonstration

First we show how the join operation work in our algorithm. The response function \( f_{test1} \) is a smooth function with single extreme and is defined as

\[
f_{test1}(x_1, x_2) = \frac{1}{10((x_1 + 1)^2 - (x_2 + 1)^2) + x_1^2 + 4}.
\]

This function has been used in Balkin and Lin (2000). Here the experimental region is \([-0.5, 2.0] \times [-1.5, 1.0] \) and the corresponding optimization problem is

Problem \( P_{test1} : \max f_{test1}(x) = \frac{1}{10((x_1 + 1)^2 - (x_2 + 1)^2) + x_1^2 + 4} \)

s.t. \(-0.5 \leq x_1 \leq 2.0, -1.5 \leq x_2 \leq 1.0\).

First the grid of experimental region is \( \{(x_1, x_2) | x_1 \in \{-0.5, -0.5 + \frac{2.5}{160}, -0.5 + \frac{2.5 \times 2}{160}, \ldots, 2.0\} \) and \( x_2 \in \{-1.5, -1.5 + \frac{2.5}{160}, -1.5 + \frac{2.5 \times 2}{160}, \ldots, 1.0\}\). The response surface of \( f_{test1} \) on grid is shown in Figure 14. We choose two grid points (124,116) and (4,68), as the
initial points, and define $r_{\text{join}} = 1$. Then PRSM is applied on these two initial points. The result is shown in Figure 15. Here the square ($\square$) are used to denote the initial points and the end point of PRSM is denoted by triangle ($\triangle$). It is easily to see that those two RSM paths are joint into one path.

Next we demonstrate fork operation. The test function $f_{\text{test}2}$ is a smooth function with two extremes and is defined as $f_{\text{test}2}(x_1, x_2) = \frac{-x_1^4 + 4.5x^2 + 2}{e^2x_2^2}$. Here the optimization problem is

$$\text{Problem } P_{\text{test}2}: \max f_{\text{test}2}(x_1, x_2) = \frac{-x_1^4 + 4.5x^2 + 2}{e^2x_2^2}$$

$$\text{s.t. } -4 \leq x_1 \leq 4, -4 \leq x_2 \leq 4,$$

and grid is chosen as $\{(x_1, x_2) | x_1 \in \{-4.0, -4.0 + \frac{8}{160}, -4.0 + \frac{8}{160}, \ldots, 4.0\} \text{ and } x_2 \in \{-4.0, -4.0 + \frac{8}{160}, -4.0 + \frac{8}{160}, \ldots, 4.0\}\}$. This function had been used in Chen et al. (2006). The response surface of $f_{\text{test}2}$ on the grid is shown in Figure 16. We choose the grid point, $(80,4)$, as the initial point. Then we process PRSM from this grid point. The result is shown in Figure 17. In this figure the square ($\square$) are used to denote the initial points and the extreme point found by PRSM is denoted by triangle ($\triangle$). It shows that when the PRSM have two possible steepest ascent directions, the PRSM does fork operation.
5.2 The performance of PRSM

In this subsection, we apply PRSM to three optimization problems for showing its performance.

- Case 1. In first case we apply PRSM to problem $P_{test1}$. Here 20 initial points are sampled by a two-factor uniform design with 20 level for each factor. Based
on the 20 initial points, first we run the RSM sequentially, and record the all paths. Here we call this method as sequential RSM (SRSM in short). In SRSM, it takes 1446 path points to finish 20 RSM’s. The result of SRSM is shown in Figure 18. Then we run PRSM with \( r_{\text{join}} = 0 \) and 1.5. They only need 1072 and 877 path points to complete PRSM respectively. The results of SRSM and two PRSM with different \( r_{\text{join}} \) are shown in Figure 19 and 20. In those figures, the numbers ,1 to 20, are used to denote the initial points, and the end points of PRSM paths are denoted by triangle (\( \triangle \)).

![Figure 18: SRSM result on problem \( P_{test1} \).](image)

![Figure 19: PRSM result on problem \( P_{test1} \). \( (r_{\text{join}} = 0) \)](image)
• Case 2. The second case, we choose $f_{test2}$ to be true response function, and consider the following optimization problem, defined as

$$\text{Problem } P'_{test2} : \max f_{test2}(x_1, x_2) = \frac{-x_1^4 + 4.5x_2^2 + 2}{e^{2x_2^2}}$$

s.t. $-2.25 \leq x_1 \leq 2.25, -2.25 \leq x_2 \leq 2.25$.

Here the grid is $\{(x_1, x_2) | x_1 \in \{-2.25 + \frac{4.5}{160}, -2.25 + \frac{4.5x_1^2}{160}, \cdots, 2.25\}$ and $x_2 \in \{-2.25 + \frac{4.5}{160}, -4.0 + \frac{4.5x_2^2}{160}, \cdots, 2.25\}$}. Then 20 initial points which is chosen by a uniform design, and we run SRSM and two PRSM with different $r_{join} = 0$ and 1.5. The response surface is shown in Figure 21. The result of SRSM and two PRSM with different $r_{join}$ is shown in Figure 22, 23, and 24. The numbers of total path points of three methods are 635, 631, and 623 respectively.

Figure 20: PRSM result on problem $P_{test1}$. ($r_{join} = 1.5$)

Figure 21: The response surface of problem $P'_{test2}$. 
Figure 22: SRSM result on problem $P_{test2}'$.

Figure 23: PRSM result on problem $P_{test2}'$. ($r_{join} = 0$)

Figure 24: PRSM result on problem $P_{test2}'$. ($r_{join} = 1.5$)
• Case 3. In third case, we consider a real example, and the response value are the Lyapunov exponents. For detail about Lyapunov exponents, please see Parker and Chua (1989), and further information regarding the problem can be found in Wang et al. (2001). Here the optimization problem is

\[
\text{Problem } P_{\text{test}3} : f_{\text{test}3}(x_1, x_2) = \text{experiment outcome on the grid point}(x_1, x_2)
\]

\[
st. \quad 20 \leq x_1 \leq 30, \quad 5 \leq x_2 \leq 15
\]

Here the grid, \(\{(x_1, x_2)|x_1 \in \{20, 20.5, \cdots, 30\} \text{ and } x_2 \in \{5, 5.5, \cdots, 15\}\}\) is chosen. Then 21 initial points is sampled by a uniform design for running SRSM and two PRSM with different \(r_{\text{join}} = 0\) and 1.5 are processed. The response surface on the grid is shown in Figure 25. The results of SRSM and two PRSM are shown in Figure 26, 27, and 28. The number of path points used in experiments are 93, 93, and 80 respectively.

Figure 25: Response surface of problem \(P_{\text{test}3}\) on grid.
Figure 26: SRSM result of problem $P'_{test3}$.

Figure 27: PRSM result of problem $P_{test3}$. ($r_{join} = 0$)

Figure 28: PRSM result of problem $P_{test3}$. ($r_{join} = 1.5$)
5.3 Trend estimation

In this subsection we apply the approaches we proposed in Section 4 for constructing the true yet unknown response surfaces. There three approaches are applied into the first case, case 1. in Section 5.2. All results are shown in Figures 29, 30 and 31. For the first approach, the surrogate model is the second- or the third-order polynomial model.

Figure 29: Trend estimation of SRSM result of $P_{test1}$. The figure show the result of (a)real response surface, (b)extrapolation, (c)2-order polynomial lsq estimation of raw data, (d)3-order polynomial lsq estimation of raw data, (e)2-order polynomial lsq estimation of data with path points added, (f)3-order polynomial lsq estimation of data with path points added, (g)2-order polynomial lsq estimation of data with contour points added, (h)3-order polynomial lsq estimation of data contour points added ($r_{join} = 1.5$)
Figure 30: Tread estimation of PRSM ($r_{\text{join}} = 0$) result of $P_{\text{test1}}$. The figure show the result of (a) real response surface, (b) extrapolation, (c) 2-order polynomial lsq estimation of raw data, (d) 3-order polynomial lsq estimation of raw data, (e) 2-order polynomial lsq estimation of data with path points added, (f) 3-order polynomial lsq estimation of data with path points added, (g) 2-order polynomial lsq estimation of data with contour points added, (h) 3-order polynomial lsq estimation of data contour points added ($r_{\text{join}} = 1.5$)
Figure 31: Tread estimation of PRSM $(r_{\text{join}} = 0)$ result of $P_{\text{test1}}$. The figure show the result of (a) real response surface, (b) extrapolation, (c) 2-order polynomial lsq estimation of raw data, (d) 3-order polynomial lsq estimation of raw data, (e) 2-order polynomial lsq estimation of data with path points added, (f) 3-order polynomial lsq estimation of data with path points added, (g) 2-order polynomial lsq estimation of data with contour points added, (h) 3-order polynomial lsq estimation of data contour points added ($r_{\text{join}} = 1.5$)
5.4 Discussions

Here we highlight some points for the performance of PRSM.

- Form these cases, it seems that PRSM can find all local extremes, and is an efficient algorithm. The following table shows the number of path points for these three cases. From this table, PRSM explore less points than those of SRSM.

<table>
<thead>
<tr>
<th></th>
<th>Case.1</th>
<th>Case.2</th>
<th>Case.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRSM</td>
<td>1446</td>
<td>635</td>
<td>93</td>
</tr>
<tr>
<td>PRSM($r_{join} = 0$)</td>
<td>1072 (74%)</td>
<td>631 (99%)</td>
<td>93 (100%)</td>
</tr>
<tr>
<td>PRSM($r_{join} = 1.5$)</td>
<td>877 (60%)</td>
<td>623 (98%)</td>
<td>80 (86%)</td>
</tr>
</tbody>
</table>

- The paths of PRSM would identify the hot spots. Then we could focus on these hot spots for searching extremes.

- Besides searching the extremes the path information is useful to detect the trend of response surface, and it would give us a rough idea about the response function.

6 Conclusion

In this thesis, we propose a novel algorithm, PRSM, for solving optimization problems with multiple extremes in computer experiments. Basically PRSM is to run a series of RSM’s parallelly. From our numerical experiments on three different types of response surfaces, PRSM shows a better performance on searching the extremes than that of SRSM. There is an extra advantage of PRSM. From the explored experiment points and corresponding response values, the trend of true yet unknown response surface would be also approximated. Then it would give us some ideas about this unknown function $f$. 
References


