The implementation and comparison of optimization methods for extreme value

ABSTRACT. In this project, we implemented three optimization methods for extreme value in python, including the golden section Method, Powell's Method and Down-Hill Simplex Method. We test functions of different dimensions in these three algorithms. It is found that in one-dimensional function, the golden section Method is the most efficient, and in multi-dimensional function, Powell's method is faster. The advantage of the Down-Hill Simplex Method is that it can find the global extreme value without manually setting the initial point.

1. Overview

Optimization problem is aim to find the target function $f(x)$, $x \in R_n$ extreme point under given constraints. Extremum can be divided into global extremum or local extremum. Global extremum is the maximum/minimum value of a function, while local extremum is the maximum/minimum value of a function in a range (Press, Teukolsky, Vetterling, & Flannery, 2007). It is usually hoped that the global extreme value of a function can be obtained, but generally speaking, it is very difficult to implement. This is because the local minimum of the objective function is related to the positive quality of the Hessian matrix when its gradient is 0.

Suppose that there is a real function of n elements which is $c(x_1, x_2, \dots, x_n)$; $x_i \in R_n, \forall i \in [1, \dots, n]$. If the function f is differentiable in the second order of any dimension, then its Hessian matrix is:

$$H(f) = \begin{bmatrix} \frac{d^2 f}{dx_1^2} & \frac{d^2 f}{dx_1 dx_2} & \cdots & \frac{d^2 f}{dx_1 dx_n} \\ \frac{d^2 f}{dx_2 dx_1} & \frac{d^2 f}{dx_2^2} & \cdots & \frac{d^2 f}{dx_2 dx_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{d^2 f}{dx_n dx_1} & \frac{d^2 f}{dx_n dx_2} & \cdots & \frac{d^2 f}{dx_n^2} \end{bmatrix}$$

For the judgment of the extreme value of the objective function, there are the following rules (on the premise that the gradient is 0):

(1) When the Hessian matrix of the function is a positive definite matrix (all eigenvalues are positive), this point is the minimum of the function.

(2) When the Hessian matrix of the function is a negative definite matrix, this point is the maximum point of the function.

(3) When the Hessian matrix of the function is an indefinite matrix, this point is the saddle point of the function.

(4) When the Hessian matrix of the function is semi-positive definite or semi-negative, it is the possible extreme point, which needs to be further determined by other methods.

When the output of the objective function is N dimension, its Hessian matrix has N eigenvalues. We first assume that Hessian matrix satisfies Gaussian distribution. According to stochastic matrix theory, the probability that any eigenvalue of Gaussian random matrix is positive or negative is 0.5. Therefore, the probability that this matrix is positive definite is 0.5^N. When N of the target function is a very large number, the saddle points of the target function will be much more than the minimum points.

Fortunately, there are some heuristic algorithms that can deal with the calculation of global extreme value to some extent currently, but it is not guaranteed that the global extreme value can be obtained every time. In addition, a class of convex optimization problems can be easily computed to obtain the global extremum.

2. Statement

Depending on the number of parameters and the smoothness of the function, the problem can be complex and sometimes very slow, and one should adopt appropriate methods to get the best and fastest results. The purpose of this paper is to do an implementation and comparison of some optimization algorithms. First of all,

according to the different dimension n of the independent variable x, the problem can be divided into one and multidimensional cases. For one-dimensional optimization method, there is only one dimension, so there is no problem of choosing the "search direction", and it only needs to calculate the extreme value iteratively in this direction. Golden section method is a very classic example. For multi-dimensional optimization methods, the situation is much more complicated. In this project, we mainly discuss Powell's Method and Down-Hill Method. The purpose of this project is not only to implement the three methods in python, but also to compare the speed, accuracy and application breadth of different functions in the application of the three optimization methods.

3. Descriptions

3.1 golden section Search

The idea of the golden section method to find the extreme value of a function is similar to the dichotomy method in the root algorithm. In the dichotomy, it is necessary to first determine the interval (a, b) where a and b satisfy $a, b < 0$ to ensure that there is at least one root in the interval. Then gradually reduce the range until the preset accuracy is reached. In the golden section, it is also necessary to determine the range of the extreme value. This can be represented by a triple (a, b, c) where $a < b < c$ and $f(b) < \min\{f(a), f(c)\}$. Thus, there is an extreme value in the interval (a, c).

Next, you need to gradually try to narrow the interval (a, c). If a new point x is selected within the interval (a, c), x falls within the interval (a, b) or (b, c). Taking $x \in (b, c)$ as an example, we calculate the value of $f(x)$ and compare it with $f(b)$. If $f(b) < f(x)$, then take the new triple (a, b, x), if $f(b) > f(x)$, then the new triples would be (b, x, c). In this way, it is always guaranteed that the length of the interval determined by the new triplet is smaller than the original interval. This process is repeated until the interval range is less than the preset precision to obtain an extreme point. The trial point x is selected as the 0.38197 split point in the longer interval. The

initial triplet does not need to meet the golden ratio, and the algorithm will automatically reach a stable repetition ratio in the iterative process. Obviously, the golden section algorithm is a linear convergence algorithm, and the interval range can be reduced to the original 0.618 in each step of calculation.

Even though the Golden Section Search is a one-dimensional method, we can still use it for more than one parameter. By optimizing the first with respect to the first variable, then with respect to the second and so on. This procedure has to be repeated many times and may eventually converge to a local minimum in the multi-dimensional space. Note that the convergence of this method is linear which is fairly slow.

3.2 Powell's Method

Powell's method is a search method based on the property that the conjugate direction can accelerate the rate of convergence. This method does not need to differentiate the objective function, and can be applied when the derivative of the objective function is discontinuous. Therefore, Powell's algorithm is a very effective direct search method.

The Powell's method can be used to solve general unconstrained optimization problems. This method is an effective conjugate direction method, which can find the minimum point of quadratic function in a finite number of steps. It is also effective for non-quadratic function as long as it has a continuous second derivative. This method does not need to calculate the derivative of the objective function. It can also get good results when the dimension $n < 20$, or the first derivative of the objective function is discontinuous. It is a more effective method for unconstrained optimization problems. Different from other direct methods, Powell's method has a complete theoretical system, so the computational efficiency is higher than other direct methods. The method uses a one-dimensional search instead of jumping probe steps. At the same time, the search direction of Powell's method is not always the direction of descent.

The optimization thought of Powell's method is actually an improvement of the conjugate direction method. Firstly, as shown on Figure 3.2.1, the initial point $X_0^{(1)}$ is

selected and started from it. According to the search direction of the coordinate rotation method, one-dimensional search is carried out along directions $\vec{e}_1$ and $\vec{e}_2$ successively to obtain the one-dimensional minimum points $X_1^{(1)}$ and $X_1^{(2)}$ in their respective directions. Connect the initial point $X_0^{(1)}$ to the last one-dimensional minimum point $X_1^{(2)}$, resulting in a new direction $\vec{S}_1 = X_1^{(2)} - X_0^{(1)}$. Then taking the end point of the first iteration $X^{(1)}$ as the starting point of the second iteration $X_0^{(2)}$. In the direction group of the first loop $(\vec{e}_1\vec{e}_2)$, after discarding the first direction $\vec{e}_1$, the vector $\vec{e}_2$ and the new direction $\vec{S}_1$ form the basic search direction group of the second loop $(\vec{e}_1,\vec{S}_1)$. The one-dimensional minimum point $X_1^{(2)}$, $X_2^{(2)}$ is obtained along the two directions in turn. By connecting $X_0^{(2)}$ and $X_2^{(2)}$, the new direction $\vec{S}_2$ of the second loop is obtained $(\vec{S}_2 = X_2^{(2)} - X_0^{(2)})$. The minimal point $X^{(2)}$ obtained by a one-dimensional search along $\vec{S}_2$ is the final iteration point of the second loop. When minimizing the positive definite quadratic function, if the first n directions in each iteration are linearly independent, then Powell's method reaches the minimum point after n iterations at most.

However, Powell's method is not an appropriate method for high-dimensional optimization problems $(n > 20)$. Because the algorithm does not measure whether the subsequent search directions become linearly dependent. Once there is a correlation between the search directions, conjugate directions cannot be formed, thus the dimensional space cannot be constructed. As a result, the subsequent iterations are carried out in the space of dimensionality reduction, degradation occurs, and the minimum point is not obtained.

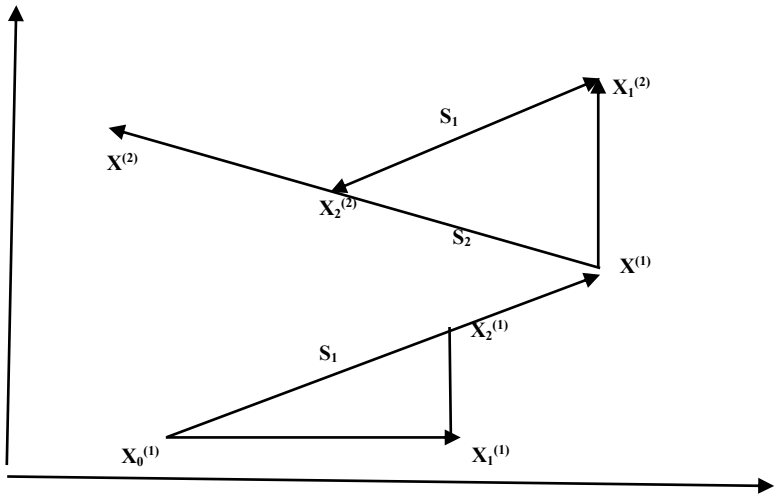

Figure 3.2.1. Search Process of Powell's Method

3.3 Down-Hill Simplex Method

The main idea of Down-Hill simplex method is to construct a non-degenerate initial simplex in n-dimensional space, carry out a series of geometric operations, such as reflection, expansion, contraction, and so on, and gradually move the simplex to the extreme point.

The main idea of Down-Hill simplex method is to construct a non-degenerate initial simplex in n-dimensional space, carry out a series of geometric operations, such as reflection, expansion, contraction, and so on, and gradually move the simplex to the extreme point. Down-Hill simplex method has been widely used in multidimensional minimization regions (Nelder & Mead, 1965). This method is a widely used "derivative free" optimization algorithm. In general, it's not as efficient as Powell's method, but it wins by being able to pre-select initial guesses and escape local minima. For many studies, the initial guess to choose depends on the knowledge and experience of the researcher. Sometimes it's hard to get a good initial guess and trying a lot of initial guesswork manually can be time-consuming (Huang, et al., 1998).

For the implementation of the Down-Hill simplex method, first, we assume that the

function to be optimized is $f(X)$, and the $N+1$ vertices of the simplex Z in N-dimensional space are arranged in order of function values from smallest to largest $x_0, x_2, \cdots, x_n$. We define $\bar{x} = \sum_{i=0}^{N-1} x_i$ is the center point of the vertices in Z except for the vertex x_N. $\bar{x}$ and x_n linear equation can be written as: $\bar{x}(t) = (1-t)\bar{x} + t \cdot x_n$. The Down-Hill simplex method is to find the alternative point of x_N from several special steps along the direction of the line $\bar{x}(t)$, so that the function value at the alternative point is smaller than that of x_N. If it is not found, then we move the rest of the points to x_0.

If we want to find the minimum value of the function, we can consider the smaller the value of the corresponding function as better, and the larger the value of the function as worse. Taking 2D as an example, the iterative process of each step of the Down-Hill simplex method is described as follows:

(1) We start with three points x_1, x_2, x_3, and make it to a triangle.

(2) Order the three points with $f(x_1) < f(x_2) < f(x_3)$, $x_0 = (x_1 + x_2)/2$.

(3) We have four options to achieve the goal in the algorithm, which including reflection, Expansion, Contraction and Shrinkage (Figure 3.3.1).

i. Reflection:

a. $x_r = x_0 + (x_0 - x_3)$

b. If $f(x_1) \leq f(x_r) < f(x_2)$, then replace x_3 by x_r and go back to (2).

ii. Expansion:

a. If $f(x_r) < f(x_1)$, $x_e = x_0 + 2(x_0 - x_3)$.

b. If $f(x_1) \leq f(x_e) < f(x_2)$ then replace x_3 by x_e and go back to (2).

iii. Contraction:

a. If $f(x_r) \geq f(x_2)$, $x_c = x_0 - {}^1/_2 (x_0 - x_3)$.

b. If $f(x_c) \geq f(x_3)$ then replace x_3 by x_c and go back to (2).

iv. Shrinkage: if none of the above works, replace $x_i = x_1 + {}^1/_2 (x_i - x_1)$

We continue in this way as we get closer to the target minimum point. Eventually the triangles get so close.

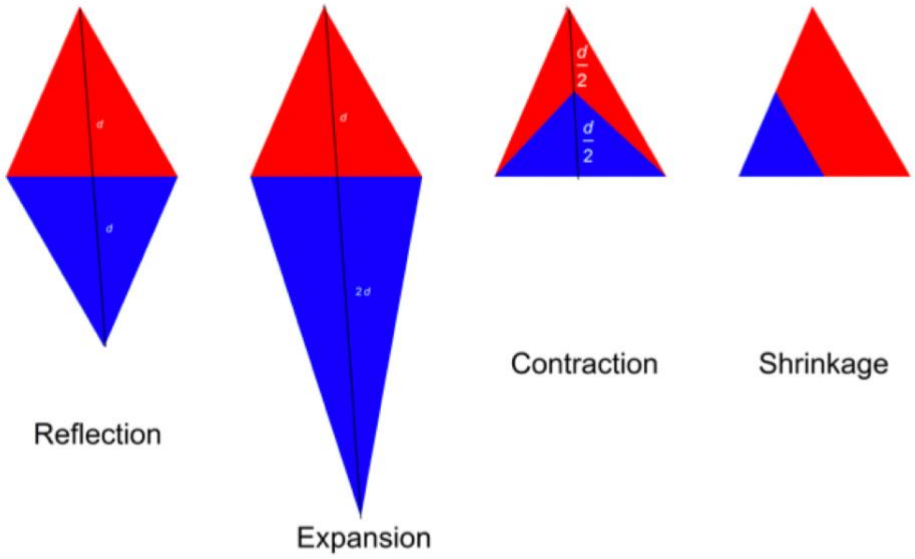

Figure 3.3.1. Elementary move of Down-Hill Method

4. Results

In this project, we mainly use three examples to test the speed, accuracy and range of application of the three methods.

4.1 One Dimensional smooth Function

Firstly, we use a simple one-dimension function $f = x^2$ to test the validation and compare the speed and precision with which the three methods run this function.

As we have checked, all three methods work on a one-dimensional smooth function.

For the time-consuming test, we use the *time* module to perform the experiments. We can see that the Golden Section Search method gives the fastest in practice with nearly 0.785 ms for 21 iterations. When we focus on the rest two methods, Powell's method and Down-Hill method, we have the time consuming on Down-Hill method is 1.624 ms which is almost twice faster as that on Powell's method at 2.643 ms. We can also see that the Down-Hill only iterate 6 times while Powell iterates 54 times (see Fig.4.1.1).

We also compare the results with the implementation in *scipy.optimize* package. We obtain the fastest method in *opt.minimize (method='Powell')* with 1.28 ms which is twice faster than the result for the Down-Hill method(2.46 ms). This is contrary to what we have in our practice implementation. The Powell's method has 2 iterations while Down-Hill have 21 iterations in the *scipy.optimize* package. We can see, more or less illustrates the speed for implementing those methods is corresponding to the number of iterations (see Fig.4.1.1).

All the method compared to those using the *scipy.optimize* package is much more precise. The Powell's method gives the best estimate answer with 4.55749e-12 in the difference between the real value which is slightly more accurate than other two methods. The Down-Hill method shows the least precise estimator with 1.68366e-10 in the error. All the errors are similar and above 0.5 in our practice which is considerable. But when we use the *scipy.optimize* package, we can see that the *opt.golden* and *opt.downhill* have the error is approx. equal to 0 that cannot be identified by the computer. Even The *opt.powell* has 3.55e-15 precision error (see Fig.4.1.1).

	# Iterations	Time (ms)	Precision error
Golden Search section	21	0.478	3.3053e-05
Opt.golden		7.713	0.0
Powell's method	54	2.643	4.55749e-12
Opt.powell	2	1.282	3.55e-15
Down Hill method	6	1.624	1.68366e-10
Opt.downhill	21	2.463	0.0

Figure 4.1.1. The number of iterations, times, and precision result for one-dimensional function in Project.ipynb

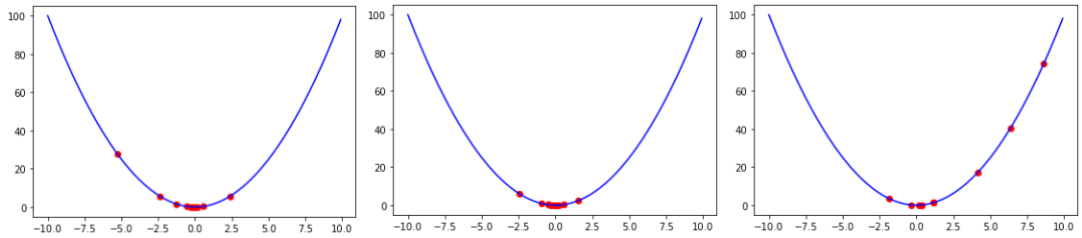

Figure 4.1.2 Iterations moves of the Golden Search Section method, Powell's method and Down Hill method. We can see the convergence route at each iteration. Eventually the points get so close to the real value it is hard to resolve there are they anymore.

4.2 Two-Dimensional smooth Function

Now we will do the same test of validation and compare the speed and precision for a two-dimensional function $f = x^2 + y^2$.

As shown in figure 4.2.1, the golden search method implements the 2-D function but still treats it as a 1d function, which corresponds to what we assume that Golden Section Search is a one-dimensional method. Even though we can still achieve the minimum point by finding the first with respect to the first variable, then with respect to the second and so on. This means that we need is to repeat the golden search section method of the function value until eventually converges to a local minimum in the multi-dimensional space. Even though this method is workable, it is obviously time consuming.

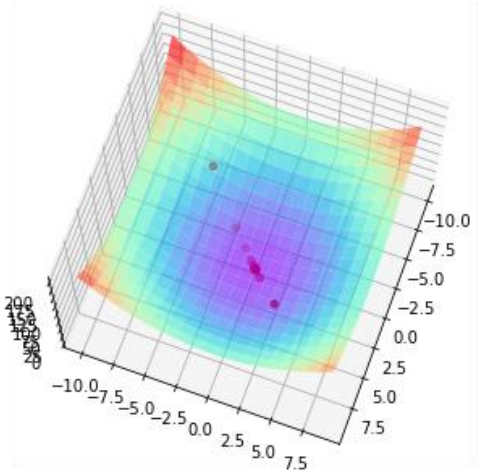

Figure 4.2.1. Iterations route for Golden search section method on two-dimensional function

Now we only have to compare Powell's method and Down Hill method. The data in Table 4.2.2 shows the same pattern as the result we obtained on a one-dimensional function. In our practical implementation, the Powell's method gives the slowest calculation for 9.56ms of 1888 iterations. Down Hill method shows a 3.14ms running time of 15 iterations. On the contrary, the methods using *scipy.optimize* package illustrate that the Powell's method speed (1.845 ms) is again more than twice faster than the speed of Down Hill method with 5.092 ms. It iterates 2 times and 46 times respectively. We observe the positive relationship between the number of iterations and the running speed again. And the opposite comparison between the speed of Powell's method and Down Hill method in practice coding and *scipy.optimize* package (Table 4.2.2).

We can see precision error in Powell's method are smaller than Down-Hill method. But if we run the *scipy.optimize* package in both methods, there is a so minor error especially compared to what we have in our practical coding. Thus, there may be some room for us to improve the functions we defined in Solver.py when it gets closer and closer to the optimized point. But this one is not like what we are going to talk about below.

	# Iterations	**Time (ms)**	**Precision error**
Powell's method	1888	9.56	00.000169
Opt.powell	2	1.845	5.799e-14
Down Hill method	687	3.14	0.0375

Opt.downhill	46	5.092	3.696e-5

Table 4.2.2. The number of iterations, times, and precision result for two-dimensional function in Project.ipynb

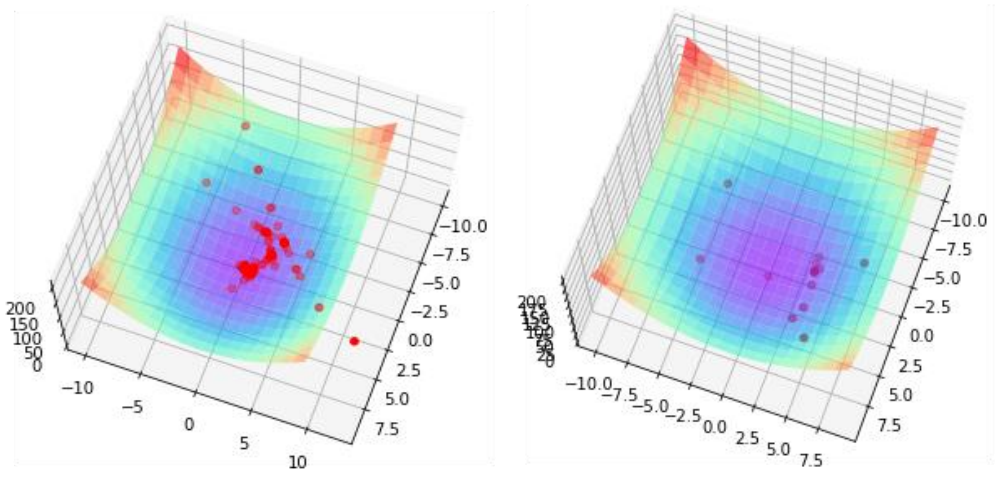

Figure 4.2.3. Iterations moves of the Powell's method and Down Hill method in two-dimensional. We can see the convergence route at each iteration. Eventually the points get so close to the real value it is hard to resolve there are they anymore.

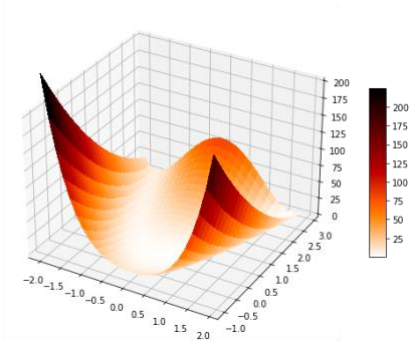

Figure 4.2.4. The Rosenbrock function in 2-dimensions with a = 1, b = 10

We also do the same test on the Rosenbrock function. This gives a similar result as we stated earlier. Except we claimed that there may be some improvement we can do in our code as the iterate close to the real value. In the Rosenbrock function, we can see that the *scipy.optimization* package does not give a better result than our implementation. This may result from the property of the Rosenbrock function. As we can see in Fig 4.2.4 the Rosenbrock function has a wide valley, with a small gradient.

Although the valley is easy to find, it is difficult to converge to the minimum. This creates a challenge for optimization algorithms.

	# Iterations	Time (ms)	Precision error
Powell's method	98	10.67	0.700639
Opt.powell	14	13.6	9.93824
Down Hill method	27	4.28	0.8549
Opt.downhill	129	13.6	1.99994

Table 4.2.5. The number of iterations, times, and precision result for Rosenbrock function in Project.ipynb

We may also say that the Powell's method and Down-Hill method running time and precision are relatively stable in 2-Dimensions.

4.3 Two-Dimensional non-smooth Function

Now we are trying to test on multidimensional non-smooth functions with our methods, we focus on two-dimension for simplicity.

The Schwefel Function

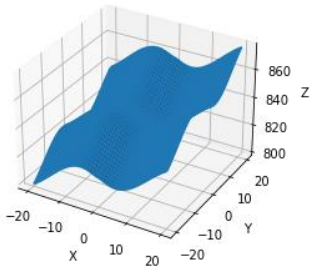

Figure 4.2.6 The Schwefel function in 2-dimensions

From Table 4.2.7, we can see that the speed of Powell's method is 35.766ms which is much faster than that of Down Hill method, also the accuracy of Powell's method on the Schwefel function is also much better than Down Hill method. The number of iterations of both methods are significant, which may result from the properties of the Schwefel function.

	# Iterations	Time (ms)	Precision error
Powell's method	11432	35.766	3.2488e-05
Down Hill method	1578	94.247	0.149088

Table 4.2.7. The number of iterations, times, and precision result for Schwefel function.

The Rastrigin function

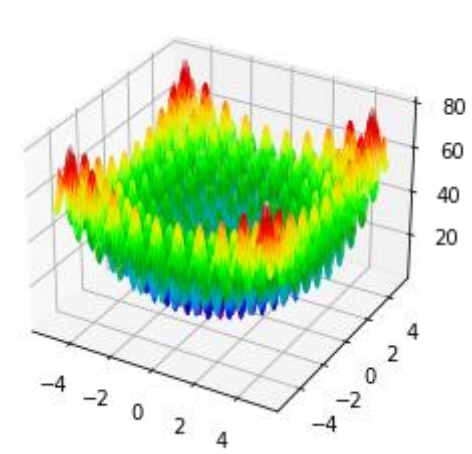

Figure 4.2.8 The Rastrigin function in 2-dimensions

As we can see in figure 4.2.8 on the following page, this function has many local minima. It is highly multimodal, but the location of the minimum is regularly distributed.

The data from Table 4.2.9 gives us consistency with the data from Schwefel function on the Powell's method performs faster. But the accuracy of Down Hill method is better than Powell's method.

	# Iterations	Time (ms)	Precision error
Powell's method	1277	20.138	10.24312
Down Hill method	1967	52.536	0.994958

Table 4.2.9. The number of iterations, times, and precision result for Rastrigin function.

5. Conclusions and Future Work

In conclusion, for one-dimensional functions, although all three methods can be applied, the golden section method is the most efficient. The golden section method cannot be applied to two - dimensional functions. After comparison, Powell's Method has advantages over Down-Hill in accuracy and time.

Based on that, we led to the discussion that in optimization techniques, there is no algorithm that is highly efficient for all problems or for any initial value for some problems, and a good algorithm can only be solved successfully for most optimization problems with an acceptable solution speed. The future work is to continue to explore how to use the combination of several appropriate algorithms to further improve the efficiency of solving unconstrained optimization problems.

Works Cited

Huang, M., Aine, C., Supek, S., Best, E., Ranken, D., & Flynn, E. (1998). Multi-start downhill simplex method for spatio-temporal source localization in magnetoencephalography. *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section, 108*(1), 32-44.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal, 7*(4), 308-313.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Minimization or Maximization of Functions. In *Numerical recipes the art of scientific computing* (p. 487). Cambridge University Press.