21374389 牛鹏军 210711

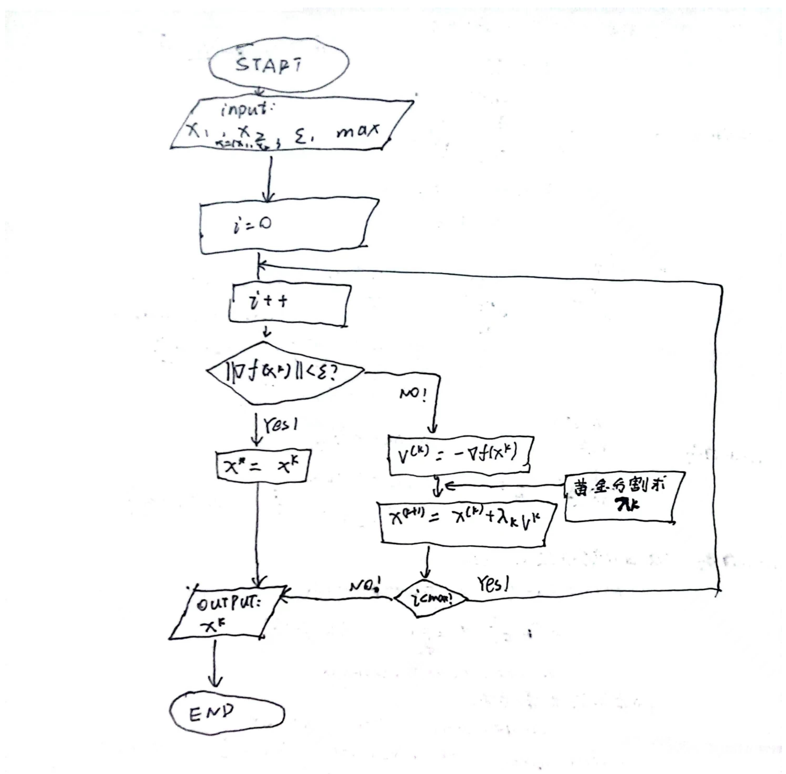
理论分析

3. 在第 2 题算法基础上，用 C 语言实现二元函数的最速下降算法，并用 Rosenbrock 函数 $f(x,y)=(1-x)^2+100(y-x^2)^2$ 为实例验证算法的应用效果。

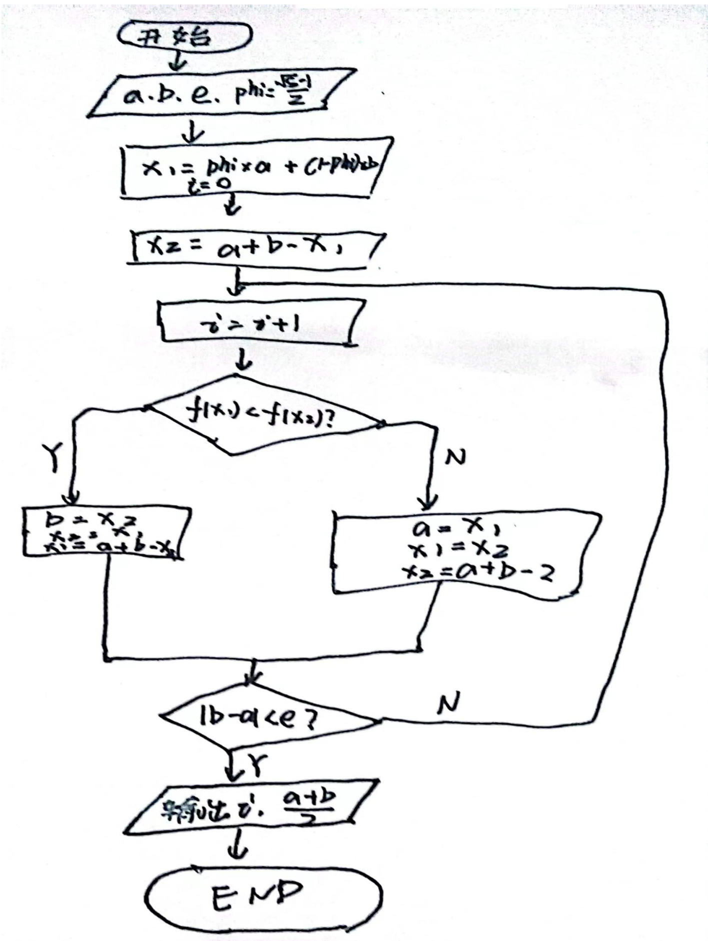
正常参照教材算法 7.1 最速下降法将伪代码实现，其中求 lamda 部分使用黄金分割法将 lamda 作为变量。

算法设计

此为最速下降求全局最小值的框图



此为黄金分割求 lamda 的框图



编程实现

```
#include <stdio.h>

#include <math.h>



#define EPSILON 0.001

#define MAX 66666



double f(double x, double y) {

    return pow(1 - x, 2) + 100 * pow(y - x * x, 2);

}
```

```
void gradient(double x, double y, double* grad_x, double* grad_y) {
	*grad_x = 2 * (x - 1) + 200 * (y - x * x) * (-2 * x);
	*grad_y = 200 * (y - x * x);
}

double goldenSectionSearch(double a, double b, double x1, double x2) {
	const double phi = (sqrt(5) - 1) / 2;

	double c = a + (1 - phi) * (b - a);
	double d = a + phi * (b - a);

	double f_c = f(x1 + c, x2);
	double f_d = f(x1 + d, x2);

	while (fabs(b - a) > EPSILON) {
		if (f_c < f_d) {
			b = d;
			d = c;
			c = a + (1 - phi) * (b - a);
			f_d = f_c;
			f_c = f(x1 + c, x2);
```

```
            } else {
                a = c;
                c = d;
                d = a + phi * (b - a);
                f_c = f_d;
                f_d = f(x1 + d, x2);
            }
        }

        return (a + b) / 2;
}
double golden_section_search(double a0, double b0,double x,double y) {
       double b=b0;
       double a=a0;
        const double phi = (sqrt(5)-1) / 2; //  黄金分割比例

        double x1 = phi*a+(1-phi)*b; //  计算内点 1
        double x2 = a+b-x1; //  计算内点 2
       int i=0;
//   f(x1 + c, x2)
        while (fabs(b - a) > EPSILON) {
           i++;
```

```
        double *p,*q;
        gradient(x,y,p,q);
          if(f(x+x1*(*p),y+x2*(*q))<f(x+x1*(*p),y+x2*(*q))){
            b=x2;
            x2=x1;
            x1=a+b-x2;
          }else{
            a=x1;
            x1=x2;
            x2=a+b-x2;
          }
    }
         return (a + b) / 2; //  返回极值点的估计值
}
void steepestDescent(double start_x, double start_y, double alpha) {
    double x = start_x;
    double y = start_y;
    double grad_x, grad_y;
    int i = 0;

    for (i = 0; i < MAX; i++) {
        gradient(x, y, &grad_x, &grad_y);
```

```
        double lambda = goldenSectionSearch(0.0, 1.0, x,y);
        x = x - lambda * grad_x;
        y = y - lambda * grad_y;
        if (sqrt(grad_x * grad_x + grad_y * grad_y) < EPSILON) {
            break;
        }
    }

    printf("Optimal point: x = %.2lf, y = %.2lf\n", x, y);
    printf("Number of iterations: %d\n", i);
}

int main() {
    double start_x = 1.0;
    double start_y = 1.1;
    double alpha = 0.75;

    steepestDescent(start_x, start_y, alpha);

    return 0;
}
```

测试分析

采用 EPSILON 为 0.001 时，迭代 36178 次得到解为（1.00，1.00），迭代次数很多，这与函数的性质有关，其全局最小值点（1，1）在一条狭小的“山谷”内。

```
Optimal point: x = 1.00, y = 1.00
Number of iterations: 36178

--------------------------------
Process exited after 1.747 seconds with return value 0
请按任意键继续. . .
```

结论

最速下降法求二元函数最小值，其中采用黄金分割法求 lamda，是一种比较可靠的方法，得到的结果比较精准，但在遇到函数全局最小值在狭小的“山谷”内情况时，需要迭代很多次数。