**21374389 牛鹏军 210711**

**理论分析**

|  |
| --- |
| 5ca1482909bfa29dddd512e4b184df8 |

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  3aa907bacf84fd6029e73d1d21fea5a |
| 此为黄金分割求lamda的框图  906897d4db91746265f8edea0e773d5 |

**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。

|  |
| --- |
|  |

**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。