Report on the Experiment

No. 2

Subject 勾配法と探索法の比較と離散化による誤差の測定

Date 2023. 1. 30

Weather　Temp ℃ Wet ％

Class　S1
Group
Chief
Partner

No　17
Name　小畠 一泰

Kure National College of Technology

1　目的

関数の最小・最大化問題のような最適化問題の解法は, 数学的に関数が微分可能であれば容易に解を得ることができる. しかし, 関数の微分が困難な場合や不可能な場合は, 関数の最小・最大値を得ることができない. このような場合, 数値微分を使った勾配法や遺伝的アルゴリズムを応用した探索法などにより解を得る方法がある. 勾配法は傾斜に沿って少ない計算で比較的早く解を発見することができるが多くの局所的な最適解が存在する場合は広域的な最適解を発見することが困難な場合がある. また, 遺伝的アルゴリズムのような探索法を用いた場合, 広域的な解を発見する確率は上がるが解を得るために多くの計算時間を必要とする場合がある.

本演習は, 勾配法による解の発見や探索法による解の発見を行いその比較を行う. また, 勾配法に用いる数値微分や遺伝的アルゴリズムを用いる際の探索範囲の離散化による誤差の測定を行う.

2　関数 g1 の確認

コード 1 テストプログラム：関数 g1 の確認処理 (g1.c)

```c
#include <stdio.h>

#include "gene.h"

int main() {
  FILE *fp;
  double x;
  int t;

  fp = fopen("g1.csv", "wt");

  for (t = -100; t <= 100; t++) {
    x = (double)t / 10.e0;
    fprintf(fp, "%lf,%lf\n", x, g1(x));
  }
  fclose(fp);
  return 0;
}
```

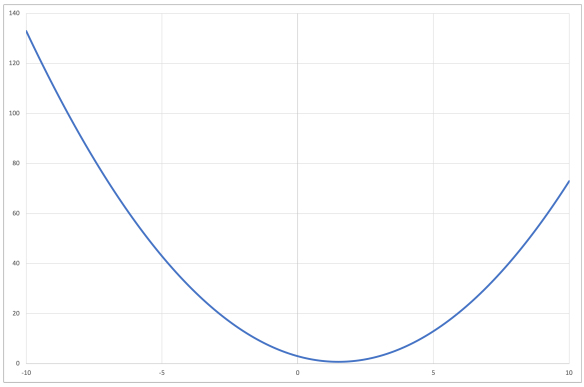

図 1: 関数 g1 の値

3　勾配法による解の発見や探索法による解の発見を行いその比較を行う

コード 2 勾配法による関数 g1 の最小値を求める処理 (g1g.c)

```c
#include <math.h>
#include <stdio.h>

#include "gene.h"

int main() {
  int c = 0;
  double a = 0.6, x0, x1 = -10.e0;

  do {
    x0 = x1;
    x1 = x0 - a * (g1(x0 + 0.5e-3) - g1(x0 - 0.5e-3)) / 1.e-3;
    c++;
  } while (fabs(x1 - x0) > 1.e-8);

  printf("%1f %d\n", x1, c);
  return 0;
}
```

コード 3 g1 の最小値を求めた結果

```
$ ./main
1.500000 15
```

コード 4 g1g の初期値 a = 0.9 により探索を行った結果

```
$ ./main
1.500000 98
```

コード 5 g1g の初期値 a = 0.01 により探索を行った結果

```
$ ./main
1.500000 841
```

4　複数の解を持つ関数の勾配法による解の発見

コード 6 勾配法による関数 g2 解の探索 (g2.c)

```c
#include <stdio.h>

#include "gene.h"

int main() {
  FILE *fp;
  double x;
  int t;

  fp = fopen("g2.csv", "wt");

  for (t = -100; t <= 100; t++) {
    x = (double)t / 10.e0;
    fprintf(fp, "%lf,%lf\n", x, g2(x));
  }
  fclose(fp);
  return 0;
}
```

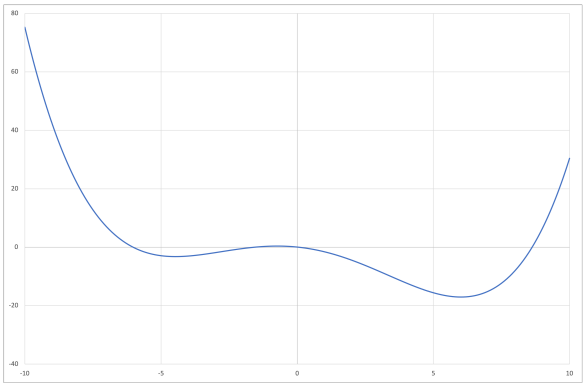

図 2: 関数 g2 の値

コード 7 勾配法による関数 g2 解の探索 (g2g.c)

```c
#include <stdio.h>

#include "gene.h"

int main() {
  int c = 0;
  double a = 0.2, x0, x1 = -10.e0;

  do {
    x0 = x1;
    x1 = x0 - a * (g2(x0 + 0.5e-3) - g2(x0 - 0.5e-3)) / 1.e-3;
    c++;
  } while (fabs(x1 - x0) > 1.e-8);

  printf("%1f %d\n", x1, c);
  return 0;
}
```

コード 8 初期値に x1=-10 を選び回を探索した結果

```
$ ./main
-4.469109 46
```

コード 9 初期値に x1=10 を選び回を探索した結果

```
$ ./main
6.012712 20
```

コード 10 初期値に x1=0 を選び回を探索した結果

```
$ ./main
6.012712 28
```

5　多元関数の最小値の探索問題

コード 11 テストプログラム：関数 g3 の確認処理 (g3.c)

```c
#include <stdio.h>

#include "gene.h"

int main() {
  FILE *fp;
  double x0, x1;
  int t0, t1;

  fp = fopen("g3.csv", "wt");

  for (t1 = -100; t1 <= 100; t1++) {
    x1 = (double)t1 / 10.e0;
    fprintf(fp, ",%lf", x1);
  }
  fprintf(fp, "\n");

  for (t0 = -100; t0 <= 100; t0++) {
    x0 = (double)t0 / 10.e0;
    fprintf(fp, "%lf", x0);
    for (t1 = -100; t1 <= 100; t1++) {
      x1 = (double)t1 / 10.e0;
      fprintf(fp, ",%lf", g3(x0, x1));
    }

    fprintf(fp, "\n");
  }

  fclose(fp);
  return 0;
}
```

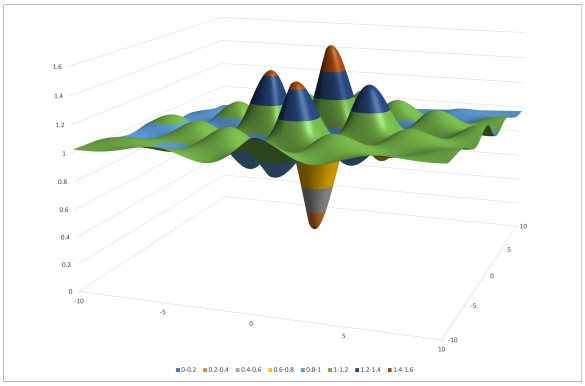

図 3: 関数 g3 の値

コード 12 勾配法による関数 g3 の最小値を求める処理 (g3g.c)

```c
#include <stdio.h>

#include "gene.h"

int main() {
  double x00, x01 = -10.e0, x10, x11 = -10.e0, a = 0.3;
  int c = 0;

  do {
    x00 = x01;
    x10 = x11;

    x01 = x00 - a * (g3(x00 + 0.5e-3, x10) - g3(x00 - 0.5e-3, x10)) / 1.e-3;
    x11 = x10 - a * (g3(x00, x10 + 0.5e-3) - g3(x00, x10 - 0.5e-3)) / 1.e-3;

    if (x01 >  10.e0) x01 =  10.e0;
    if (x01 < -10.e0) x01 = -10.e0;
    if (x11 >  10.e0) x11 =  10.e0;
    if (x11 < -10.e0) x11 = -10.e0;
    c++;
  } while (fabs(x01 - x00) > 1.e-8 || fabs(x11 - x10) > 1.e-8);

  printf("x0 = %lf, x1 = %lf, g3() = %lf, %d\n", x01, x11, g3(x01, x11), c);
  return 0;
}
```

コード 13 g3g.c の初期値 x0=-10, x1=-10 により探索を行った結果

```
$ ./main
x0 = -3.753734, x1 = -8.814267, g3() = 0.930826, 3720
```

コード 14 g3g.c の初期値 x0=-0.8, x1=0.2 により探索を行った結果

```
$ ./main
x0 = -0.789987, x1 = 0.246784, g3() = 0.249455, 51
```

6　GA による探索を用いた解法

コード 15 GA による解の探索

```
#include <stdio.h>
#include <stdlib.h>
#include "gene.h"
double vg(int a) { return (double)(a - 4096) / 409.6e0; }
int main() {
  int g[50][2] = {{0, 0}, {8191, 8191}, {4096, 4096}}, bv, c, d, gn;
  double g3v[50], gv;
  for (c = 3; c < 50; c++) { // generate initial genes
  rst0:
    g[c][0] = rand() % 8192;
    g[c][1] = rand() % 8192;
    for (d = 0; d < c; d++) {
      if (g[c][0] == g[d][0] && g[c][1] == g[d][1]) {
        goto rst0;
      }
    }
  }

  for (gn = 0; gn <= 2000; gn++) {
    for (c = 0; c < 50; c++) { // get value
      g3v[c] = g3(vg(g[c][0]), vg(g[c][1]));
    }
    for (c = 0; c < 50 - 1; c++) { // sort
      for (d = c + 1; d < 50; d++) {
        if (g3v[c] > g3v[d]) {
          bv = g[c][0];
          g[c][0] = g[d][0];
          g[d][0] = bv;

          bv = g[c][1];
          g[c][1] = g[d][1];
          g[d][1] = bv;

          gv = g3v[c];
          g3v[c] = g3v[d];
          g3v[d] = gv;
        }
      }
    }
```

コード 16 GA による解の探索

```
40      for (c = 30; c < 31; c++) {
41      rst1:
42        g[c][0] = g[rand() % 30][0];
43        g[c][1] = g[rand() % 30][1];
44        for (d = 0; d < c; d++) {
45          if (g[c][0] == g[d][0] && g[c][1] == g[d][1]) {
46            goto rst1;
47          }
48        }
49      }
50
51      for (c = 31; c < 40; c++) {
52      rst2:
53        g[c][0] = g[c][0] ^ (rand() % 512);
54        g[c][1] = g[c][1] ^ (rand() % 512);
55        for (d = 0; d < c; d++) {
56          if (g[c][0] == g[d][0] && g[c][1] == g[d][1]) {
57            goto rst2;
58          }
59        }
60      }
61
62      for (c = 40; c < 50; c++) {
63      rst3:
64        g[c][0] = rand() % 8192;
65        g[c][1] = rand() % 8192;
66        for (d = 0; d < c; d++) {
67          if (g[c][0] == g[d][0] && g[c][1] == g[d][1]) {
68            goto rst3;
69          }
70        }
71      }
72
73      printf("\n%d %1f %1f %1f\n\n", gn, vg(g[0][0]), vg(g[0][1]),
74             g3(vg(g[0][0]), vg(g[0][1])));
75    }
76
77    return 0;
78  }
```

コード 17 GA により探索を行った結果

```
$ ./main
2000 -0.791016 0.246582 0.249455
```

7　α 値の変化による収束の様子や初期値の与え方による収束の変化について

コード 3, コード 4, コード 5 より α を大きくすると最小値を飛び越してしまい計算量の増加を, 逆に小さくしても徐々にしか近づけないため計算の増加を確認できた. またより大きくすると計算量が増加したためか, 発散したためか不明だが試行時間内に最小値を見つけることができなかった. このことから計算量を抑え有限時間内に終了させるには α 並びに初期値は適切に設定する必要があることがわかった.

同様にコード 8, コード 9, コード 10 から複数の解を有する場合, 初期値の与え方によって局所的解か広域的解を見つけられるかが変わることがわかる.

8　GA による探索の収束

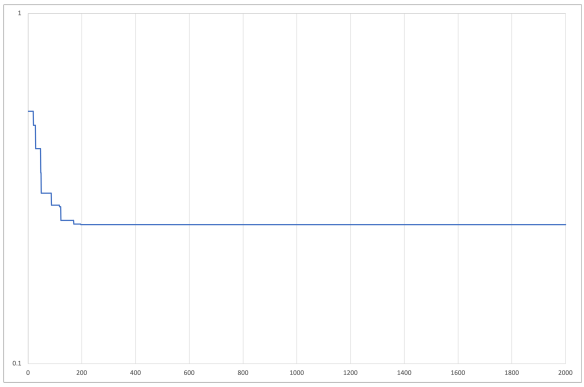

図 4: GA により最小値が収束していく様子

図 4 より突然変異を組み込むことでかなり早い世代で収束していることがわかり, GA は正確な初期値を与えることなく解が求められている.

9　本演習で用いた勾配法や探索法以外の解法について

次のような解法が考えられる

- ブルームフィルター法
  - 探索範囲を狭めることができるため, 探索空間を小さくすることができる.
  - 単純なアルゴリズムのため, 計算時間がかかることがある.
- ダイクストラ法
  - 最短距離を求めるために有効であり, 高速に計算することができる.
  - グラフが大きすぎる場合, メモリを圧迫することがある.
- ヒープソート法
  - スケジューリングアルゴリズムに使用されることが多く, 高速な探索が可能である.
  - 非常に大きなデータセットには適していない.
- スタートアップ法
  - 初期状態からの最短距離を求めるために有効であり, 高速に計算することができる.
  - 不安定なアルゴリズムであり, 探索空間が大きい場合は適していない.

- スイープ法
  - 線形探索アルゴリズムの一つであり, 高速に計算することができる.
  - 探索空間が大きい場合は適していない.
- ベイズ最適化法
  - 不確かな関数の最小値を求めるために有効であり, 高精度な結果を得ることができる.
  - 計算量が大きく, 処理に時間がかかることがある.