数値積分の課題

学籍番号 2120029, 氏名 政野玄空

2023 年 7 月 21 日

1　台形公式

1.1　$S = \int_1^2 \frac{2}{x^2} dx$ を台形公式を用いて解くプログラムとその実行結果

$S = \int_1^2 \frac{2}{x^2} dx$ を台形公式を用いて解くプログラムを作成した.

ソースコード 1: kadai4-1-1

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <unistd.h>
4
5  double sekibun(double a, double b, double n);
6  /* 関数の定義 */
7  double f(double x);
8
9  int main(void)
10 {
11   printf("S=%lf",sekibun(1,2,100));
12   return (0);
13 }
14
15 double sekibun(double a, double b, double n)
16 {
17   double h;
18     h = (b - a) / n;
19
20     double x, sum;
21     int i;
22
23     sum = 0;
24     sum = 1/2 * f(a);
25     for (i = 1; i < n; i++)
26     {
27         x = a + h * i;
28         sum += f(x);
29     }
30
31     sum = sum + 1/2 * f(b);
```

```
32        return h * sum;
33    }
34
35    /* 関数の定義 */
36    double f(double x)
37    {
38      return 2/pow(x,2);
39    }
```

実行結果はこの様になった.

```
genku@gen-mba output % ./"kadai4-1-1"
S=0.987529%
```

1.2 $S = \int_0^1 \frac{4}{1+x^2} dx$ を台形公式を用いて解くプログラムとその実行結果

$S = \int_0^1 \frac{4}{1+x^2} dx$ を台形公式を用いて解くプログラムを作成した.

ソースコード 2: kadai4-1-2.c

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <unistd.h>
4
5  double sekibun(double a, double b, double n);
6  /* 関数の定義 */
7  double f(double x);
8
9  int main(void)
10 {
11   printf("S=%lf",sekibun(0,1,100));
12   return (0);
13 }
14
15 double sekibun(double a, double b, double n)
16 {
17   double h;
18   h = (b - a) / n;
19
20   double x, sum;
21   int i;
22
23   sum = 0;
24   sum = 1/2 * f(a);
25   for (i = 1; i < n; i++)
26   {
27       x = a + h * i;
28       sum += f(x);
29   }
30
```

```
31   sum = sum + 1/2 * f(b);
32   return h * sum;
33 }
34
35 /* 関数の定義 */
36 double f(double x)
37 {
38   return 4/(1+pow(x,2)); /* 穴埋め箇所 5 */
39 }
```

実行結果はこの様になった.

```
genku@gen-mba output % ./"kadai4-1-2"
S=3.111576%
```

2 シンプソン公式

2.1 $S = \int_1^2 \frac{2}{x^2} dx$ をシンプソン公式を用いて解くプログラムとその実行結果

$S = \int_1^2 \frac{2}{x^2} dx$ をシンプソン公式を用いて解くプログラムを作成した.

ソースコード 3: kadai4-2-1.c

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <unistd.h>
4
5  double simpson(double a, double b, double n);
6  /* 関数の定義 */
7  double f(double x);
8
9  int main(void)
10 {
11   printf("S=%lf",simpson(1,2,50));
12   return (0);
13 }
14
15 double simpson(double a, double b, double n)
16 {
17   double S,h;
18         int i;
19
20         h=(b-a)/(2.0*n);
21
22         S=(f(a)+f(b));
23         for (i=1;i<n;i++){
24                 S += 4.0*f(a+(2.0*i-1.0)*h)+2.0*f(a+2.0*i*h);
25         }
26         S += 4.0*f(a+(2.0*n-1.0)*h);
27         S *= h/3.0;
```

```
28
29          return S;
30  }
31
32  /* 関数の定義 */
33  double f(double x)
34  {
35    return 2/pow(x,2); /* 穴埋め箇所 5 */
36  }
```

実行結果はこの様になった.

```
genku@gen-mba output % ./"kadai4-2-1"
S=1.000000%
```

ソースコード 4: kadai4-2-2.c

```
1   #include <stdio.h>
2   #include <math.h>
3   #include <unistd.h>
4
5   double simpson(double a, double b, double n);
6   /* 関数の定義 */
7   double f(double x);
8
9   int main(void)
10  {
11    printf("S=%lf",simpson(0,1,50));
12    return (0);
13  }
14
15  double simpson(double a, double b, double n)
16  {
17    double S,h;
18          int i;
19
20          h=(b-a)/(2.0*n);
21
22          S=(f(a)+f(b));
23          for (i=1;i<n;i++){
24                  S += 4.0*f(a+(2.0*i-1.0)*h)+2.0*f(a+2.0*i*h);
25          }
26          S += 4.0*f(a+(2.0*n-1.0)*h);
27          S *= h/3.0;
28
29          return S;
30  }
31
32  /* 関数の定義 */
33  double f(double x)
34  {
```

```
35    return 4/(1+pow(x,2)); /* 穴埋め箇所 5 */
36 }
```

実行結果はこの様になった.

```
genku@gen-mba output % ./"kadai4-2-2"
S=3.141593%
```

3 $S = \int_1^2 \frac{2}{x^2} dx$,$S = \int_0^1 \frac{4}{1+x^2} dx$ を手計算で解きプログラムとの誤差を示す.

手計算の結果を示す.

$$S = \int_1^2 \frac{2}{x^2}\,dx$$

$$= \left[-\frac{2}{x}\right]_1^2 = -(1-2) = 1$$

$$S = \int_0^1 \frac{4}{1+x^2}\,dx$$

$x = \tan\theta \left(-\frac{\pi}{2} < 0 < \frac{\pi}{2}\right)$ と置く。

$$\frac{dx}{d\theta} = \frac{1}{\cos^2\theta} \text{ より}$$

$$\int_0^{\frac{\pi}{4}} \frac{4}{\tan^2\theta + 1} \cdot \frac{1}{\cos^2\theta}\,d\theta$$

$$= \int_0^{\frac{\pi}{4}} \frac{4}{\frac{1}{\cos^2\theta}} \cdot \frac{1}{\cos^2\theta}\,d\theta$$

$$= \int_0^{\frac{\pi}{4}} 4\,d\theta = \left[4\theta\right]_0^{\frac{\pi}{4}} = \pi$$

定積分の結果は,$S = \int_1^2 \frac{2}{x^2} dx = 1$,$S = \int_0^1 \frac{4}{1+x^2} dx = \pi$ となった. プログラムとの誤差を示すためそれぞれまとめる以下の表のようになった.

表 1: 計算結果

定積分	台形公式	シンプソン公式	手計算
$S = \int_1^2 \frac{2}{x^2} dx$	0.987529	1	1
$S = \int_0^1 \frac{4}{1+x^2} dx$	3.11157	3.141593	π

4 各公式において、分割数と結果の精度について考察

結果は台形公式のほうが誤差があることになったが分割数を増やすとどうなるのか試してみる.
$S = \int_1^2 \frac{2}{x^2} dx$ の分割数 n を 10000 にした場合.

```
genku@gen-mba output % ./"kadai4-1-1"
S=0.999875%
```

$S = \int_1^2 \frac{2}{x^2} dx$ の分割数 n を 100000 にした場合.

```
genku@gen-mba output % ./"kadai4-1-1"
S=0.999988%
```

$S = \int_1^2 \frac{2}{x^2} dx$ の分割数 n を 1000000 にした場合.

```
genku@gen-mba output % ./"kadai4-1-1"
S=0.999999%
```

$S = \int_1^2 \frac{2}{x^2} dx$ の分割数 n を 10000000 にした場合.

```
genku@gen-mba output % ./"kadai4-1-1"
S=1.000000%
```

このようにシンプソン公式と比べて台形公式でより正確な値を出すには多くの分割数が必要だということがわかる. 分割数が増えるたびに計算量が増えるので今回書いたプログラムではシンプソン公式のほうを使うのが効率がよいだろう.