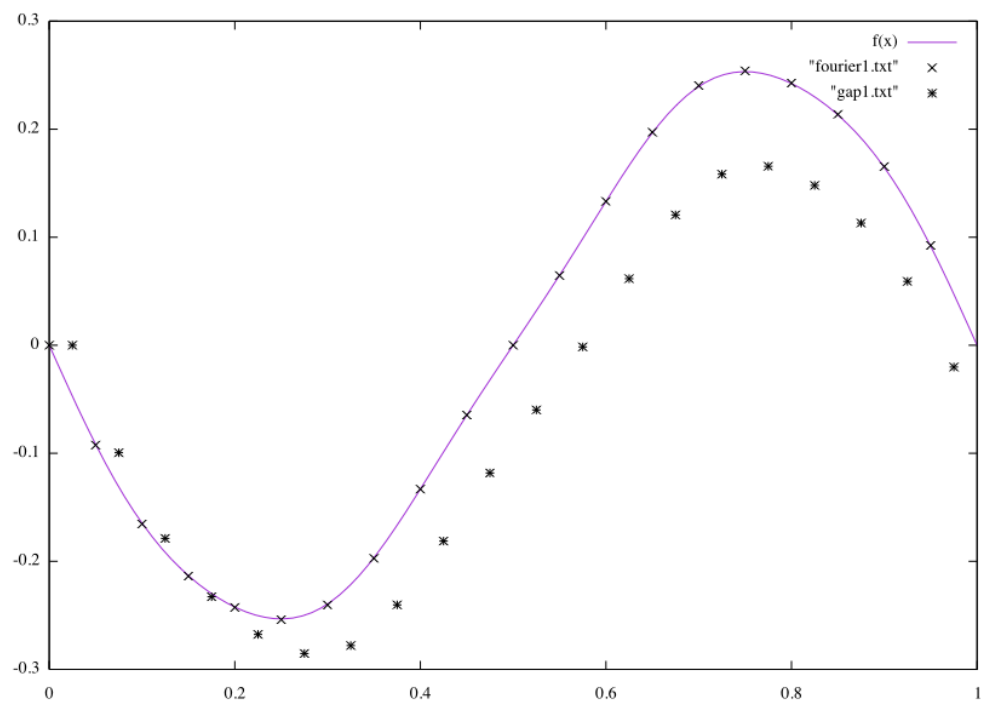


08-153031

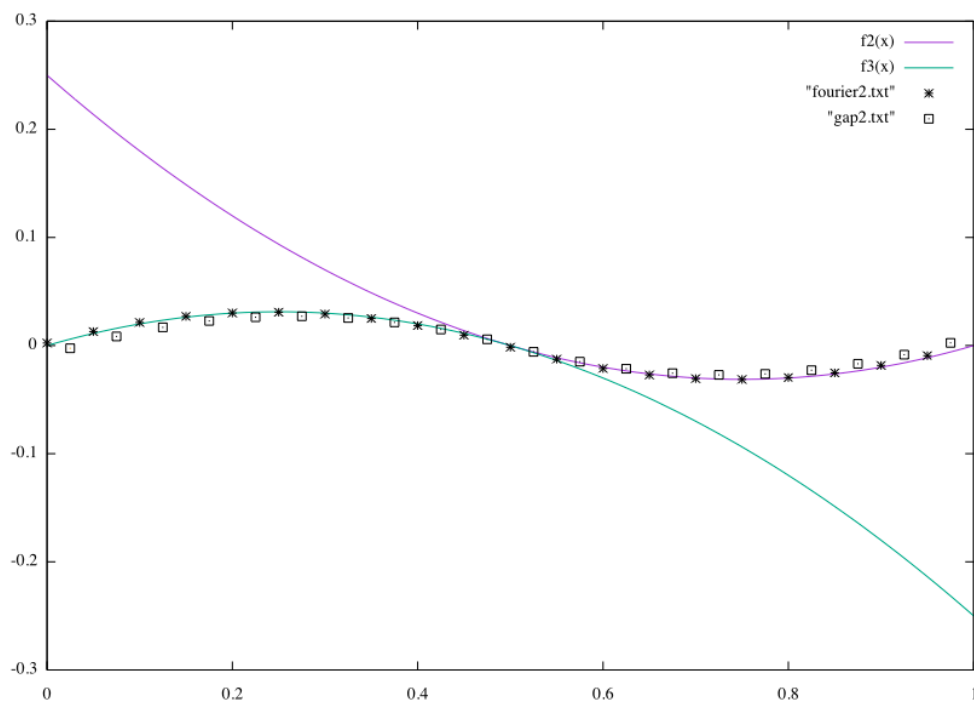
教養学部統合自然科学科統合生命コース3年 木戸口 航

# 結果

①



②



## ソースコード

### ① フーリエ変換

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <complex.h>
4  #define PI 3.141592653589793238
5
6  double g1 (double t) {
7      double result = 10 * sin(2*PI*t) + 2 * sin(4*PI*t) + 4 * sin(8*PI*t);
8      return result;
9  }
10
11 double g2 (int i) {
12     if (0 <= i && i < 10) {
13         return 1;
14     } else if (i == 1) {
15         return 0;
16     } else if (i > 10) {
17         return -1;
18     }
19 }
20
21 int main(void) {
22     double N = 20;
23     double time = 1;
24     double dlt = time / N;
25     double t = 0;
26     double n = 0;
27
28     double _Complex G[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
29     double _Complex rev[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
30
31     double _Complex tmp = cexp(I*2*PI/N);
32
33     for (int i=0; i<N; i++){
34         for (int k=0; k<N; k++) {
35             G[i] += g1(t) * cpow(tmp, i*k);
36             t += dlt;
37         }
38     }
39
40     for (int i=0; i<N; i++) {
41         for (int k=1; k<N; k++)
42         {
43             // printf("%d %f\n", k, cimag((-pow(time/(2*PI*k), 2)) * G[k]));
44             rev[i] += (-pow(time/(2*PI*(k-N)), 2)) * G[k] * cpow(tmp, -i*k);
45             // rev[i] += G[k] * cpow(tmp, -i*k);
46         }
47         rev[i] /= N;
48     }
49
50     //f(=rev)を出力
51     t = 0;
52     for (int i=0; i<N; i++) {
53         printf("%f %f\n", t, creal(rev[i]*2));
54         t += dlt;
55     }
56 }
```

## ① 差分法

```
3  #include <math.h>
4  #define PI 3.141592653589793238
5
6  double g1 (double t) {
7      double result = 10 * sin(2*PI*t) + 2 * sin(4*PI*t) + 4 * sin(8*PI*t);
8      return result;
9  }
10
11  int main(void) {
12
13      int i, j, k, n;
14      double x = -0.475;
15
16      // 拡大係数行列 M
17      double M[N][N + 1];
18
19      for (k = 0; k < N; k++) {
20
21          if (k == 0) { //1行目にセット
22              M[k][0] = 1;
23              for (n = 1; n < N; n++) {
24                  M[k][n] = 0;
25              }
26          } else if (k == N-1) { //N+1行目にセット
27              M[k][N-1] = 1;
28              for (n = 0; n < N-1; n++) {
29                  M[k][n] = 0;
30              }
31          } else {
32              for (n = 0; n < N; n++) {
33                  if (k == n) {
34                      M[k][n] = -2;
35                  } else if (k == n-1 || k == n+1) {
36                      M[k][n] = 1;
37                  } else {
38                      M[k][n] = 0;
39                  }
40              }
41          }
42      }
43
44      for (k = 0; k < N; k++) {
45          M[k][N] = g1((k*1.0)/N);
46      }
47
48      double pivot, mul;
49
50      // 対角成分が1で正規化された階段行列を作る(前進消去)
51      for (i = 0; i < N; ++i)
52      {
53          // 対角成分の選択、この値で行成分を正規化
54          pivot = M[i][i];
55          for (j = 0; j < N + 1; ++j)
56          {
57              M[i][j] = (1 / pivot) * M[i][j];
58          }
59
60          // 階段行列を作る為に、現在の行より下の行について
61          // 1列目の成分が0になるような基本変形をする
62          for (k = i + 1; k < N; ++k)
63          {
64              mul = M[k][i];
65              for (n = 1; n < N + 1; ++n)
66              {
67                  M[k][n] = M[k][n] - mul * M[i][n];
68              }
69          }
70      }
71
72      // 下から上に向かって変数に代入して、独立した解の形にする(後進代入)
73      // このとき一番下の行はすでに独立した解を得ている
74      for (i = N - 1; i > 0; --i)
75      {
76          for (k = i - 1; k >= 0; --k)
77          {
78              mul = M[k][i];
79              for (n = 1; n < N + 1; ++n)
80              {
81                  M[k][n] = M[k][n] - mul * M[i][n];
82              }
83          }
84      }
85
86      for (k = 0; k < N; ++k) {
87          printf("%f %f\n", x+0.5, M[k][N]*(0.05)*(0.05));
88          x+=0.05;
89      }
90  }
```

## ② フーリエ変換

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <complex.h>
4  #define PI 3.141592653589793238
5
6  double g2 (int i) {
7      if (0 <= i && i < 10) {
8          return -1;
9      } else if (i == 1) {
10         return 0;
11     } else if (i > 10) {
12         return 1;
13     }
14 }
15
16 int main(void) {
17     double N = 20;
18     double time = 1;
19     double dlt = time / N;
20     double t = 0;
21     double n = 0;
22
23     double _Complex G[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
24     double _Complex rev[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
25
26     double _Complex tmp = cexp(I*2*PI/N);
27
28     for (int i=0; i<N; i++){
29         for (int k=0; k<N; k++) {
30             G[i] += g2(k) * cpow(tmp, i*k);
31         }
32     }
33
34     for (int i=0; i<N; i++) {
35         for (int k=1; k<N; k++)
36         {
37             rev[i] += (-pow(time/(2*PI*(k-N)), 2)) * G[k] * cpow(tmp, -i*k);
38         }
39         rev[i] /= N;
40     }
41
42     //f(=rev)を出力
43     t = 0;
44     for (int i=0; i<N; i++) {
45         printf("%f %f\n", t, creal(rev[i]*2));
46         t += dlt;
47     }
48 }
```

## ② 差分法

```
3  #include <math.h>
4  #define PI 3.141592653589793238
5
6  double g1(double t) {
7      double result = 10 * sin(2*PI*t) + 2 * sin(4*PI*t) + 4 * sin(8*PI*t);
8      return result;
9  }
10
11 int main(void) {
12
13     int i, j, k, n;
14     double x = -0.475;
15
16     // 拡大係数行列 M
17     double M[N][N + 1];
18
19     for (k = 0; k < N; k++) {
20
21         if (k == 0) { //1行目にセット
22             M[k][0] = 1;
23             for (n = 1; n < N; n++) {
24                 M[k][n] = 0;
25             }
26         } else if (k == N-1) { //N+1行目にセット
27             M[k][N-1] = 1;
28             for (n = 0; n < N-1; n++) {
29                 M[k][n] = 0;
30             }
31         } else {
32             for (n = 0; n < N; n++) {
33                 if (k == n) {
34                     M[k][n] = -2;
35                 } else if (k == n-1 || k == n+1) {
36                     M[k][n] = 1;
37                 } else {
38                     M[k][n] = 0;
39                 }
40             }
41         }
42     }
43
44     for (k = 0; k < N; k++) {
45         M[k][N] = g1((k+1.0)/N);
46     }
47
48     double pivot, mul;
49
50     // 対角成分が1で正規化された階段行列を作る(前進消去)
51     for (i = 0; i < N; ++i)
52     {
53         // 対角成分の選択、この値で行成分を正規化
54         pivot = M[i][i];
55         for (j = 0; j < N + 1; ++j)
56         {
57             M[i][j] = (1 / pivot) * M[i][j];
58         }
59
60         // 階段行列を作る為に、現在の行より下の行について
61         // 1列目の成分が0になるような基本変形をする
62         for (k = i + 1; k < N; ++k)
63         {
64             mul = M[k][i];
65             for (n = i; n < N + 1; ++n)
66             {
67                 M[k][n] = M[k][n] - mul * M[i][n];
68             }
69         }
70     }
71
72     // 下から上に向かって変数に代入して、独立した解の形にする(後進代入)
73     // このとき一番下の行はすでに独立した解を得ている
74     for (i = N - 1; i > 0; --i)
75     {
76         for (k = i - 1; k >= 0; --k)
77         {
78             mul = M[k][i];
79             for (n = i; n < N + 1; ++n)
80             {
81                 M[k][n] = M[k][n] - mul * M[i][n];
82             }
83         }
84     }
85
86     for (k = 0; k < N; ++k) {
87         printf("%f %f\n", x+0.5, M[k][N]*(0.05)*(0.05));
88         x+=0.05;
89     }
90 }
```