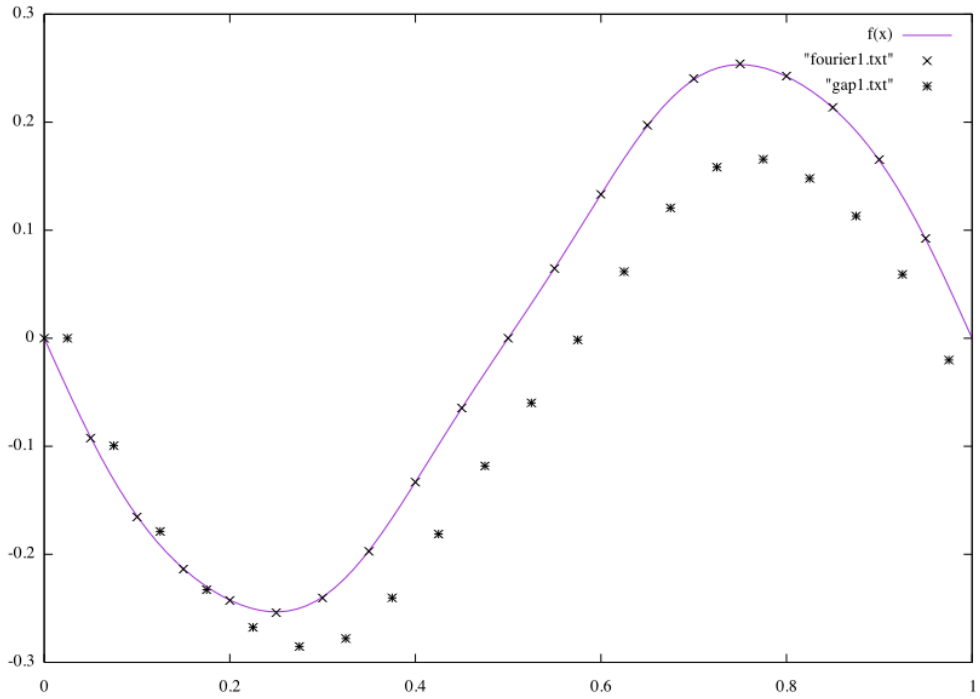
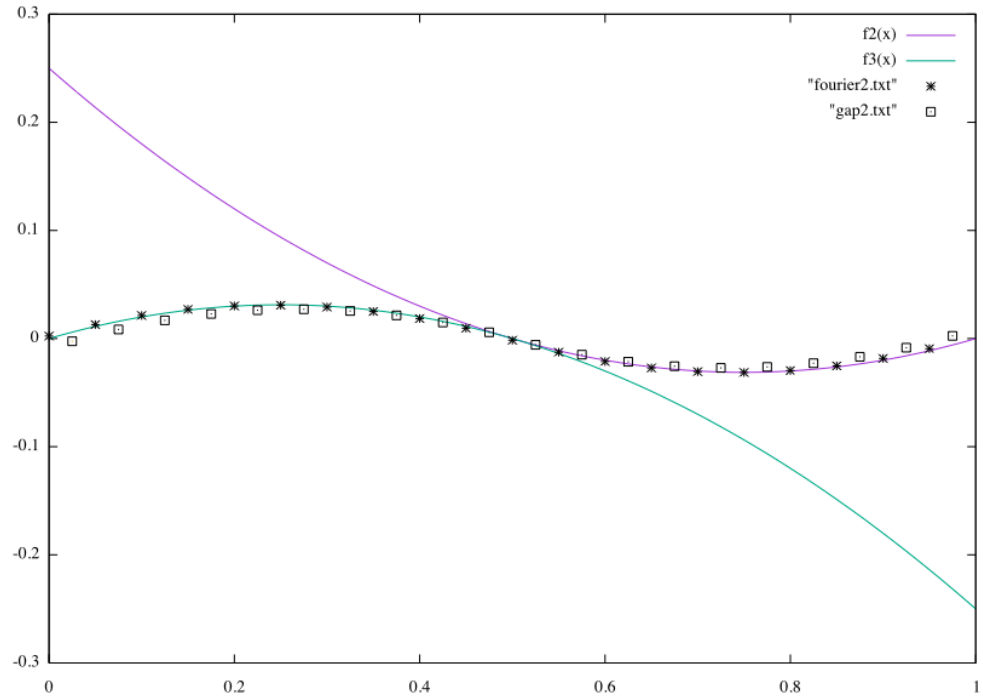
08-153031
教養学部統合自然科学科統合生命コース３年　木戸口　航

結果

①



②



ソースコード

① フーリエ変換

```c
#include <stdio.h>
#include <math.h>
#include <complex.h>
#define PI 3.141592653589793238

double g1 (double t) {
    double result = 10 * sin(2*PI*t) + 2 * sin(4*PI*t) + 4 * sin(8*PI*t);
    return result;
}

double g2 (int i) {
    if (0 <= i && i < 10) {
        return 1;
    } else if (i == 1) {
        return 0;
    } else if (i > 10) {
        return -1;
    }
}

int main(void) {
    double N = 20;
    double time = 1;
    double dlt = time / N;
    double t = 0;
    double n = 0;

    double _Complex G[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    double _Complex rev[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

    double _Complex tmp = cexp(I*2*PI/N);

    for (int i=0; i<N; i++){
        for (int k=0; k<N; k++) {
            G[i] += g1(t) * cpow(tmp, i*k);
            t += dlt;
        }
    }

    for (int i=0; i<N; i++) {
        for (int k=1; k<N; k++)
        {
            // printf("%d %f\n", k, cimag((-pow(time/(2*PI*k), 2)) * G[k]));
            rev[i] += (-pow(time/(2*PI*(k-N)), 2)) * G[k] * cpow(tmp, -i*k);
            // rev[i] += G[k] * cpow(tmp, -i*k);
        }
        rev[i] /= N;
    }

    //f(=rev)を出力
    t = 0;
    for (int i=0; i<N; i++) {
        printf("%f %f\n", t, creal(rev[i]*2));
        t += dlt;
    }
}
```

① 差分法

```c
#include <math.h>
#define PI 3.141592653589793238

double g1 (double t) {
    double result = 10 * sin(2*PI*t) + 2 * sin(4*PI*t) + 4 * sin(8*PI*t);
    return result;
}

int main(void) {

    int i, j, k, n;
    double x = -0.475;

    // 拡大係数行列 M
    double M[N][N + 1];

    for (k = 0; k < N; k++) {

        if (k == 0) { //1行目にセット
            M[k][0] = 1;
            for (n = 1; n < N; n++) {
                M[k][n] = 0;
            }
        } else if (k == N-1) { //N+1行目にセット
            M[k][N-1] = 1;
            for (n = 0; n < N-1; n++) {
                M[k][n] = 0;
            }
        } else {
            for (n = 0; n < N; n++) {
                if (k == n) {
                    M[k][n] = -2;
                } else if (k == n-1 || k == n+1) {
                    M[k][n] = 1;
                } else {
                M[k][n] = 0;
                }
            }
        }
    }

    for (k = 0; k < N; k++) {
            M[k][N] = g1((k*1.0)/N);
    }

    double pivot, mul;

    // 対角成分が1で正規化された階段行列を作る(前進消去)
    for (i = 0; i < N; ++i)
    {
        // 対角成分の選択、この値で行成分を正規化
        pivot = M[i][i];
        for (j = 0; j < N + 1; ++j)
        {
            M[i][j] = (1 / pivot) * M[i][j];
        }

        // 階段行列を作る為に、現在の行より下の行について
        // i列目の成分が0になるような基本変形をする
        for (k = i + 1; k < N; ++k)
        {
            mul = M[k][i];
            for (n = i; n < N + 1; ++n)
            {
                M[k][n] = M[k][n] -  mul * M[i][n];
            }
        }
    }

    // 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)
    // このとき一番下の行はすでに独立した解を得ている
    for (i = N - 1; i > 0; --i)
    {
        for (k = i - 1; k >= 0; --k)
        {
            mul = M[k][i];
            for (n = i; n < N + 1; ++n)
            {
                M[k][n] = M[k][n] - mul * M[i][n];
            }
        }
    }

    for (k = 0; k < N; ++k) {
        printf("%f %f\n", x+0.5, M[k][N]*(0.05)*(0.05));
        x+=0.05;
    }
}
```

② フーリエ変換

```c
#include <stdio.h>
#include <math.h>
#include <complex.h>
#define PI 3.141592653589793238

double g2 (int i) {
    if (0 <= i && i < 10) {
        return -1;
    } else if (i == 1) {
        return 0;
    } else if (i > 10) {
        return 1;
    }
}

int main(void) {
    double N = 20;
    double time = 1;
    double dlt = time / N;
    double t = 0;
    double n = 0;

    double _Complex G[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    double _Complex rev[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

    double _Complex tmp = cexp(I*2*PI/N);

    for (int i=0; i<N; i++){
        for (int k=0; k<N; k++) {
            G[i] += g2(k) * cpow(tmp, i*k);
        }
    }

    for (int i=0; i<N; i++) {
        for (int k=1; k<N; k++)
        {
            rev[i] += (-pow(time/(2*PI*(k-N)), 2)) * G[k] * cpow(tmp, -i*k);
        }
        rev[i] /= N;
    }

    //f(=rev)を出力
    t = 0;
    for (int i=0; i<N; i++) {
        printf("%f %f\n", t, creal(rev[i]*2));
        t += dlt;
    }
}
```

② 差分法

```c
#include <math.h>
#define PI 3.141592653589793238

double g1 (double t) {
    double result = 10 * sin(2*PI*t) + 2 * sin(4*PI*t) + 4 * sin(8*PI*t);
    return result;
}

int main(void) {

    int i, j, k, n;
    double x = -0.475;

    // 拡大係数行列 M
    double M[N][N + 1];

    for (k = 0; k < N; k++) {

        if (k == 0) { //1行目にセット
            M[k][0] = 1;
            for (n = 1; n < N; n++) {
                M[k][n] = 0;
            }
        } else if (k == N-1) { //N+1行目にセット
            M[k][N-1] = 1;
            for (n = 0; n < N-1; n++) {
                M[k][n] = 0;
            }
        } else {
            for (n = 0; n < N; n++) {
                if (k == n) {
                    M[k][n] = -2;
                } else if (k == n-1 || k == n+1) {
                    M[k][n] = 1;
                } else {
                M[k][n] = 0;
                }
            }
        }
    }

    for (k = 0; k < N; k++) {
            M[k][N] = g1((k*1.0)/N);
    }

    double pivot, mul;

    // 対角成分が1で正規化された階段行列を作る(前進消去)
    for (i = 0; i < N; ++i)
    {
        // 対角成分の選択、この値で行成分を正規化
        pivot = M[i][i];
        for (j = 0; j < N + 1; ++j)
        {
            M[i][j] = (1 / pivot) * M[i][j];
        }

        // 階段行列を作る為に、現在の行より下の行について
        // 1列目の成分が0になるような基本変形をする
        for (k = i + 1; k < N; ++k)
        {
            mul = M[k][i];
            for (n = i; n < N + 1; ++n)
            {
                M[k][n] = M[k][n] -  mul * M[i][n];
            }
        }
    }

    // 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)
    // このとき一番下の行はすでに独立した解を得ている
    for (i = N - 1; i > 0; --i)
    {
        for (k = i - 1; k >= 0; --k)
        {
            mul = M[k][i];
            for (n = i; n < N + 1; ++n)
            {
                M[k][n] = M[k][n] - mul * M[i][n];
            }
        }
    }

    for (k = 0; k < N; ++k) {
        printf("%f %f\n", x+0.5, M[k][N]*(0.05)*(0.05));
        x+=0.05;
    }
}
```