Maciej Kozub

Laboratorium 5 – aproksymacja

A. Aproksymacja średniokwadratowa wielomianami algebraicznymi

1. Funkcje testowe i struktury

Jako funkcje testowe wybrane zostały f(x) = $e^{\cos x(x)}$ oraz $f(x) = sin(x) \cdot e^{\frac{-x}{\pi}}$, przedział którym będziemy się zajmować to [0, 10]. Stworzona została również struktura reprezentująca punkt (argument i wartość funkcji dla tego argumentu). Wielomiany będą dopasowywane na podstawie kolejno 5, 10, 20, 50, 100 punktów, a stopnie wielomianów to kolejno: 4, 7, 10, 20.

```
const double START = 0;
const double END = 10.0;
const int degrees[4] = {4, 7, 10, 20};
const int points[5] = {5, 10, 20, 50, 100};
const std::string files[2] = {"f1", "f2"};

struct point {
    double x;
    double y;
};

double function(double x) {
    return exp(cos(x));
}
double function2(double x) {
    return sin(x)*exp( x: -x/M_PI);
}
```

W kolejnym kroku zaimplementowano funkcję zwracającą dokładną wartość funkcji testowej w podanej liczbie punktów, tak aby można ją również było wykorzystać do wyznaczenia wartości (x, y) dla punktów służących do aproksymacji.

```
std::vector<point> getPoints(double function(double x), int pointsNumber) {
    std::vector<point> values;
    double interval = (END - START) / (pointsNumber - 1);
    for (int i = 0; i < pointsNumber; i++) {
        point point{};
        point.x = START + interval * i;
        point.y = function(point.x);
        values.push_back(point);
    }
    return values;
}
```

2. Funkcja wyznaczająca współczynniki wielomianu

Aproksymacja średniokwadratowa wielomianami algebraicznymi jest metodą przybliżania funkcji tak aby suma kwadratów różnic wartości funkcji w danym punkcie oraz wartości znalezionego wielomianu powinna być jak najmniejsza. W tej metodzie funkcja błędu jest zdefiniowana następująco:

$$H(a_0, a_1, \ldots, a_m) = \sum_{j=0}^{n} w(x_j)(y_j - \sum_{i=0}^{m} a_i x_j^i)^2$$

Współczynnik w(x) jest ustaloną funkcją wagową, w naszym przypadku równą stale 1. Funkcja błędu osiąga minimum w punkcie, w którym pochodne cząstkowe względem współczynników a_i są równe zero. W celu znalezienia tego minimum należy rozwiązać zatem układ równań. Kod znajduje się poniżej:

```
std::vector<double> polynomial(const std::vector<point>& values, int degree){
    std::vector<double> sigmaX( n: 2*degree+1,  value: 0);
    std::vector<double> sigmaY( n: degree+1,  value: 0);
    std::vector<double> coefficients( n: degree + 1,  value: 0);
    double normalMatrix[degree + 1][degree + 2];

    for (int i = 0; i < sigmaX.size(); i++)
        for (point value : values)
            sigmaX[i] += pow(value.x, i);
    for (int i = 0; i < sigmaY.size(); i++)
        for (point value : values)
            sigmaY[i] += pow(value.x, i) * value.y;
    for (int i = 0; i < degree+1; i++) {
        for (int j = 0; j < degree + 1; j++)
            normalMatrix[i][j] = sigmaX[i + j];
        normalMatrix[i][degree + 1] = sigmaY[i];
    }
    for (int i = 0; i < degree; i++)
        for (int k = i+1; k < degree + 1; k++){
            double tmp = normalMatrix[k][i] / normalMatrix[i][i];
            for (int j = 0; j <= degree + 1; j++)
                normalMatrix[k][j] -= tmp * normalMatrix[i][j];
        }
    for (int i = degree; i >= 0; i--){
        coefficients.at(i) = normalMatrix[i][degree+1];
        for (int j = 0; j < degree + 1; j++)
            if (j != i)
                coefficients.at(i) -= normalMatrix[i][j] * coefficients.at(j);
        coefficients.at(i) /= normalMatrix[i][i];
    }

    return coefficients;
}
```

Tworzymy układ równań:

$$
\begin{array}{ccccccc}
a_0(n+1) & + & a_1 \sum_{j=0}^{n} x_j & + & \dots & + & a_m \sum_{j=0}^{n} x_j^m & = & \sum_{j=0}^{n} y_j \\
a_0 \sum_{j=0}^{n} x_j & + & a_1 \sum_{j=0}^{n} x_j^2 & + & \dots & + & a_m \sum_{j=0}^{n} x_j^{m+1} & = & \sum_{j=0}^{n} y_j x_j \\
\dots & & \dots & & \dots & & \dots & & \dots \\
a_0 \sum_{j=0}^{n} x_j^m & + & a_1 \sum_{j=0}^{n} x_j^{m+1} & + & \dots & + & a_m \sum_{j=0}^{n} x_j^{2m} & = & \sum_{j=0}^{n} y_j x_j^m
\end{array}
$$

A następnie po zbudowaniu macierzy „normalMatrix” rozwiązujemy ten układ metodą eliminacji Gaussa i wyznaczamy szukane współczynniki wielomianu.

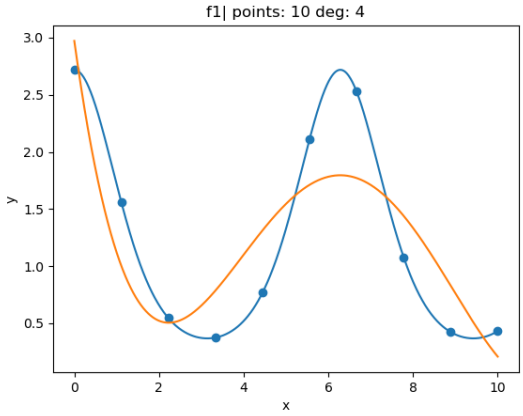
3. Funkcja aproksymująca

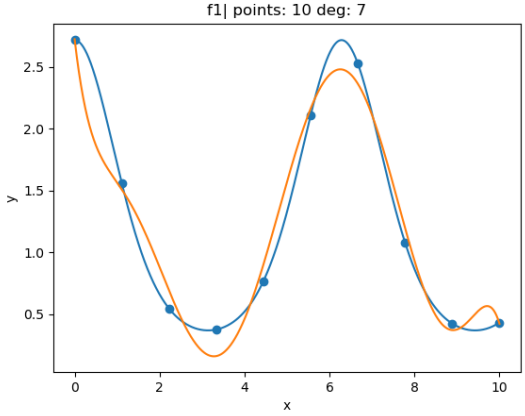
Funkcja wraz z funkcją pomocniczą wyznacza wartości wielomianu na podstawie jego współczynników i zwraca listę punktów (x, y), tak aby możliwe było na przykład narysowanie danego wielomianu i wyznaczenie dokładności aproksymacji.

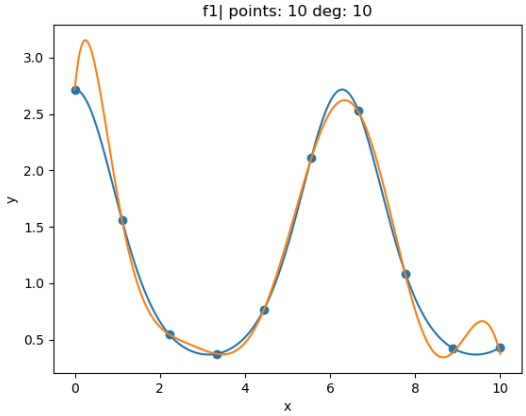
```
double approxHelper(double x, const std::vector<double>& coefficient){
    double result = 0;
    for (int i = 0; i < coefficient.size(); i++)
        result += coefficient[i]*std::pow(x, i);
    return result;
}

std::vector<point> approximation(const std::vector<point>& values, const std::vector<double>& coefficients)
{
    std::vector<point> result;
    for(point item : values){
        point point{};
        point.x = item.x;
        point.y = approxHelper(item.x, coefficients);
        result.push_back(point);
    }
    return result;
}
```

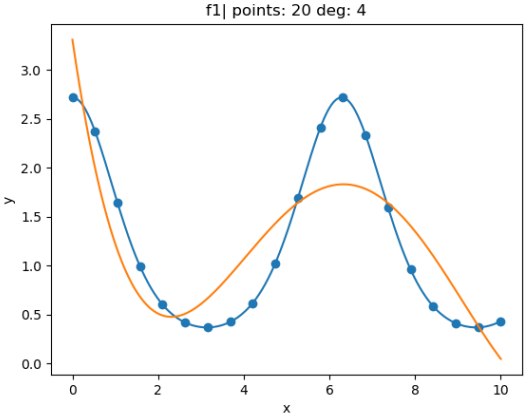
4. Wyniki

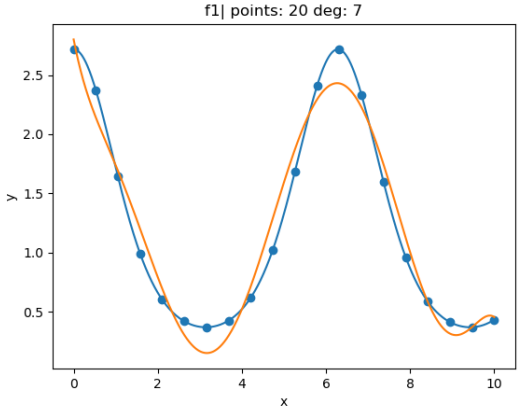
Poniżej znajduje się porównanie (niektórych) wyników, tj. wykresu funkcji aproksymowanej (kolor niebieski), węzły służące do aproksymacji (niebieskie punkty) i wykres wielomianu aproksymującego (kolor pomarańczowy) dla różnej liczby węzłów i różnych stopni wielomianu.

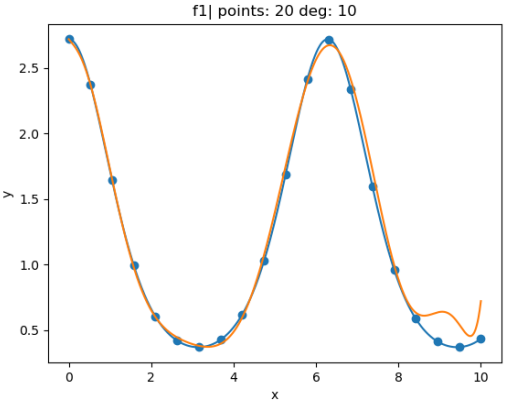

f1| points: 10 deg: 4
x
y

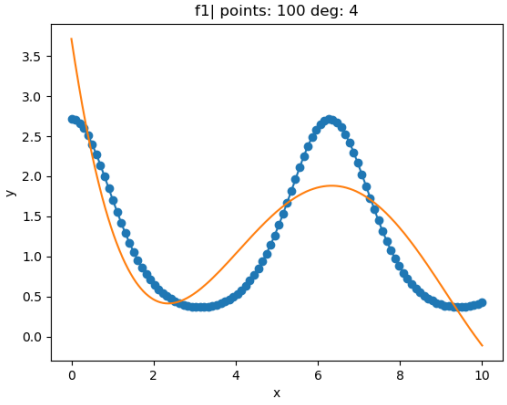

f1| points: 10 deg: 7
x
y

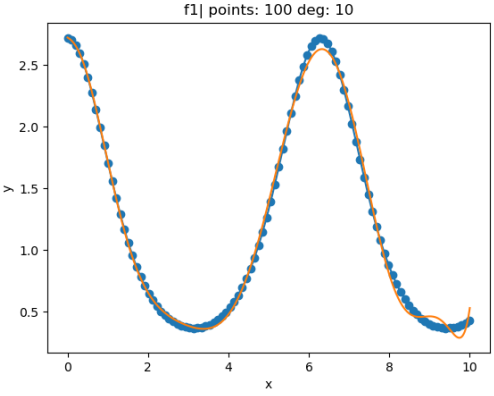

f1| points: 10 deg: 10
x
y

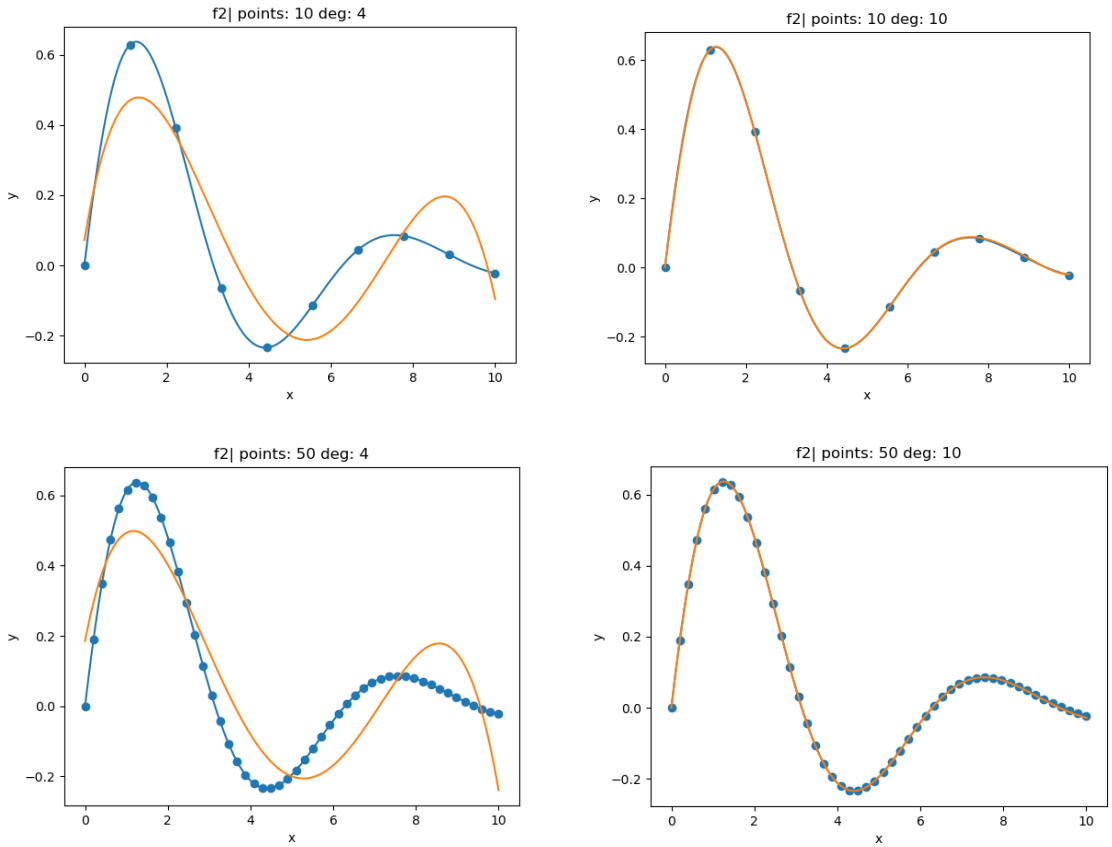

f1| points: 20 deg: 4
x
y


f1| points: 20 deg: 7
x
y


f1| points: 20 deg: 10
x
y


f1| points: 100 deg: 4
x
y


f1| points: 100 deg: 10
x
y



5. Błędy

Funkcja obliczająca błąd aproksymacji wyznacza całkowitą różnicę pomiędzy wartościami funkcji aproksymowanej w punktach wykorzystanych do aproksymacji oraz wartościami otrzymanego wielomianu w tych punktach.

```
double totalError(double function(double x), const std::vector<point>& result) {
    double error = 0.0;
    for(point item : result){
        error += std::abs( x: function(item.x) - item.y);
    }
    return error;
}
```

Wyniki znajdują się poniżej.

P f1: points number: 5, degree: 4, error: 4.79616e-14
P f1: points number: 5, degree: 7, error: 5.82312e-14
P f1: points number: 5, degree: 10, error: 1.97398e-13
P f1: points number: 5, degree: 20, error: 3.63043e-14
P f1: points number: 10, degree: 4, error: 3.95321
P f1: points number: 10, degree: 7, error: 0.94071
P f1: points number: 10, degree: 10, error: 3.527e-05
P f1: points number: 10, degree: 20, error: 3.16017e-07
P f1: points number: 20, degree: 4, error: 7.55603

```
P f1: points number: 20, degree: 7, error: 2.62474
P f1: points number: 20, degree: 10, error: 0.646711
P f1: points number: 20, degree: 20, error: 0.141472
P f1: points number: 50, degree: 4, error: 18.1202
P f1: points number: 50, degree: 7, error: 6.65039
P f1: points number: 50, degree: 10, error: 1.82454
P f1: points number: 50, degree: 20, error: 0.492825
P f1: points number: 100, degree: 4, error: 35.5706
P f1: points number: 100, degree: 7, error: 13.0741
P f1: points number: 100, degree: 10, error: 3.60744
P f1: points number: 100, degree: 20, error: 1.27936
P f2: points number: 5, degree: 4, error: 3.10661e-14
P f2: points number: 5, degree: 7, error: 1.33206e-14
P f2: points number: 5, degree: 10, error: 9.88411e-14
P f2: points number: 5, degree: 20, error: 1.38924e-14
P f2: points number: 10, degree: 4, error: 1.00026
P f2: points number: 10, degree: 7, error: 0.0294551
P f2: points number: 10, degree: 10, error: 7.58059e-07
P f2: points number: 10, degree: 20, error: 2.39209e-08
P f2: points number: 20, degree: 4, error: 2.01037
P f2: points number: 20, degree: 7, error: 0.104308
P f2: points number: 20, degree: 10, error: 0.000655278
P f2: points number: 20, degree: 20, error: 3.16252e-05
P f2: points number: 50, degree: 4, error: 4.61614
P f2: points number: 50, degree: 7, error: 0.25938
P f2: points number: 50, degree: 10, error: 0.00212461
P f2: points number: 50, degree: 20, error: 0.000611649
P f2: points number: 100, degree: 4, error: 9.00845
P f2: points number: 100, degree: 7, error: 0.490863
P f2: points number: 100, degree: 10, error: 0.00417295
P f2: points number: 100, degree: 20, error: 0.000659409
```

Na podstawie otrzymanych wartości można zauważyć że dla danej liczby punktów dokładność aproksymacji wzrasta wraz ze wzrostem stopnia wielomianu. Widać również, że przy zwiększeniu liczby punktów wzrasta również błąd aproksymacji przy zachowaniu poprzednio opisanej tendencji. Zauważalna jest również, że dokładniejszą aproksymację otrzymaliśmy dla drugiej funkcji.

B. Aproksymacja średniokwadratowa trygonometryczna

1. Funkcje testowe i struktury

Zmianie uległ jedynie rozpatrywany przedział, teraz to: [0,10].

2. Funkcja wyznaczająca współczynniki wielomianu

W metodzie aproksymacji wielomianami trygonometrycznymi poszukujemy wielomianu postaci:

$$F(x) = \frac{1}{2}a_0 + \sum_{j=1}^{n}(a_j cos(jx) + b_j sin(jx))$$

Gdzie czynniki a_j, b_j wyznacza się z warunku minimalizacji wyrażenia:

$$\sum_{i=0}^{2L-1}[f(x_i) - F(x_i)]^2 = min$$

Gdzie 2L to ilość punktów aproksymacji. Kod przestawiony poniżej:

```
std::vector<double> trigonometric(const std::vector<point>& values, int degree, bool ifSIN){
    std::vector<double> coefficients( n: degree + 1,  value: 0);

    for(int i = 0; i <= degree; i++){
        double sum = 0;
        for (auto value : values) {
            if (ifSIN)
                sum += value.y * sin( _X: i * value.x);
            else
                sum += value.y * cos( _X: i * value.x);
        }
        coefficients[i] = 2 * sum / values.size();
    }
    return coefficients;
}
```

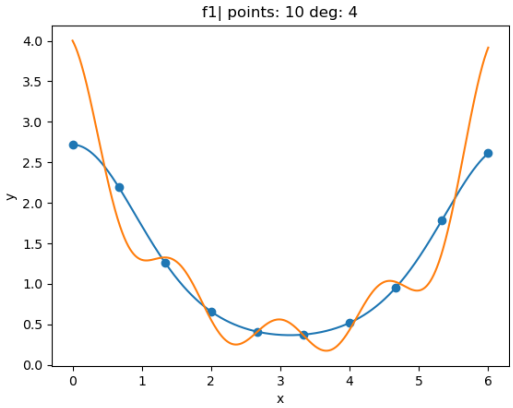
3. Funkcja aproksymująca

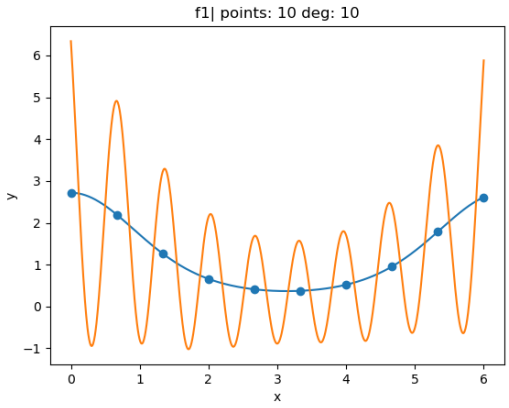
Zmianie uległa funkcja pomocnicza, która oblicza wartość wielomianu w punkcie na podstawie danych współczynników tego wielomianu.

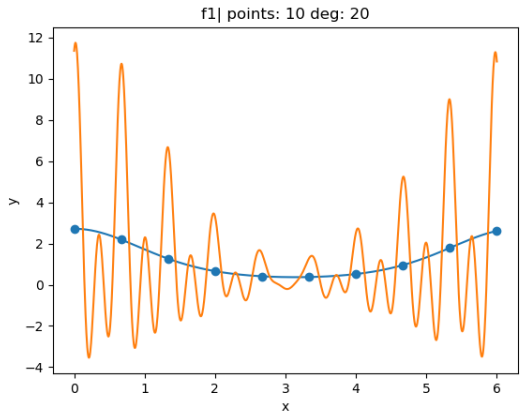
```
double approxHelper(double x, const std::vector<double>& coeffCOS, const std::vector<double>& coeffSIN){
    double result = coeffCOS[0] / 2;
    for (int i = 1; i < coeffCOS.size(); i++)
        result += (coeffCOS[i]*cos( _X: i*x) +  coeffSIN[i]*sin( _X: i*x));
    return result;
}
```

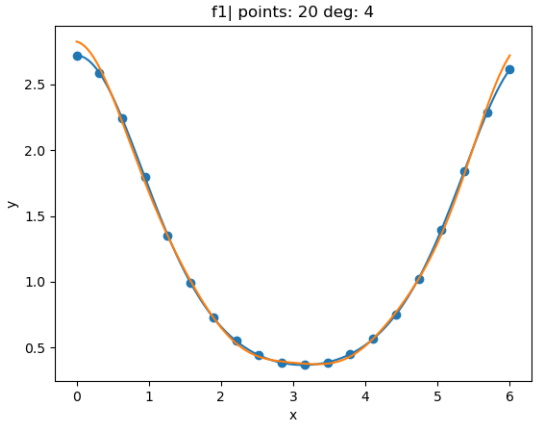
4. Wyniki

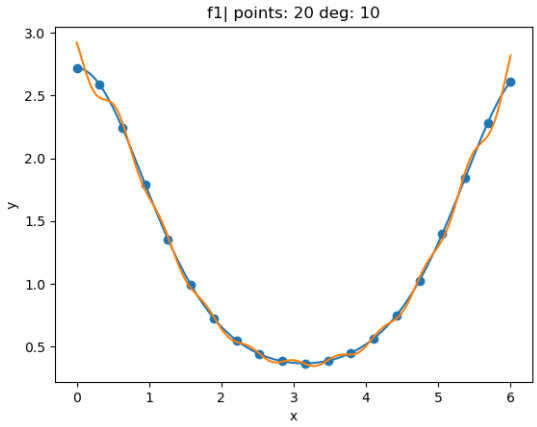
Poniżej znajduje się porównanie (niektórych) wyników, tj. wykresu funkcji aproksymowanej (kolor niebieski), węzły służące do aproksymacji (niebieskie punkty) i wykres wielomianu aproksymującego (kolor pomarańczowy) dla różnej liczby węzłów i różnych stopni wielomianu.

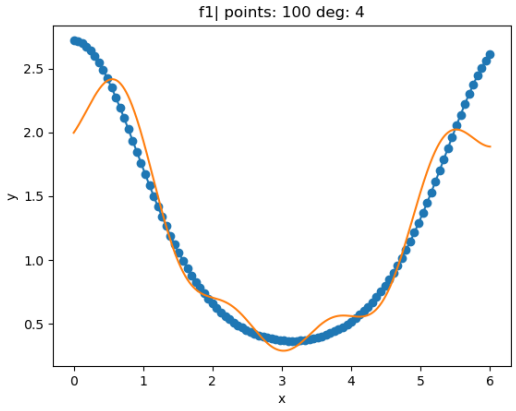


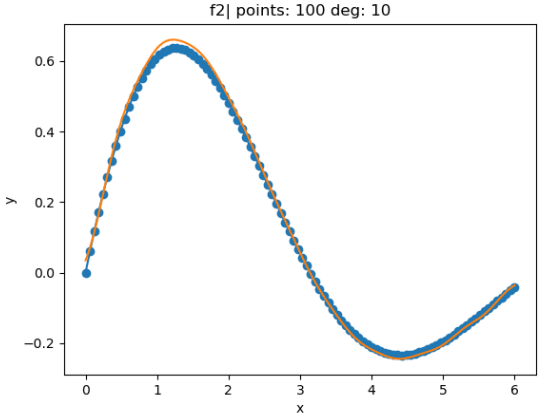


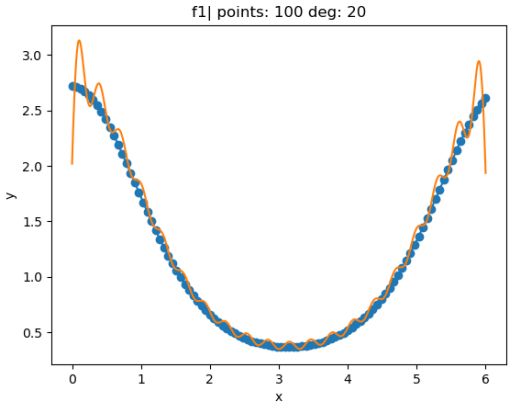


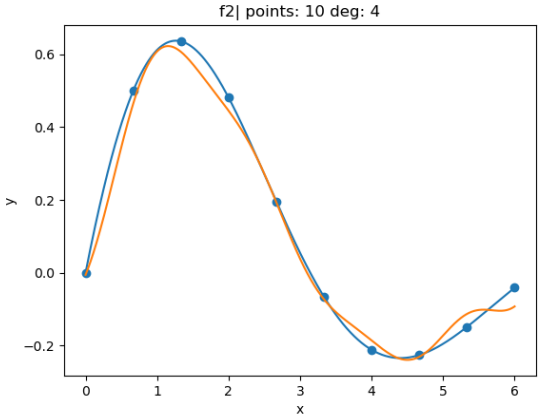


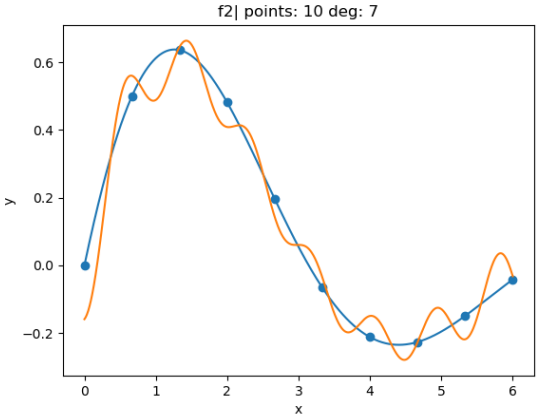


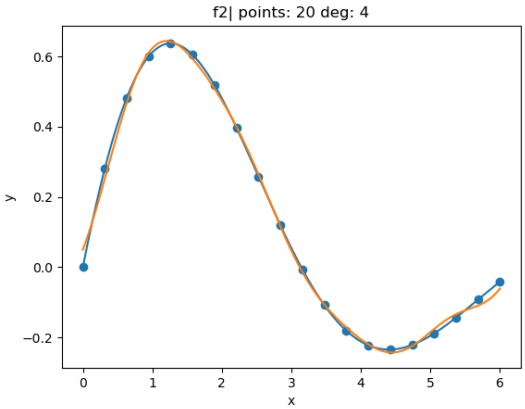


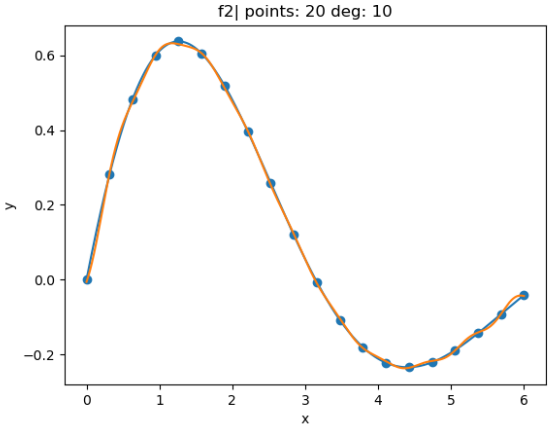

f2| points: 100 deg: 10
x
y

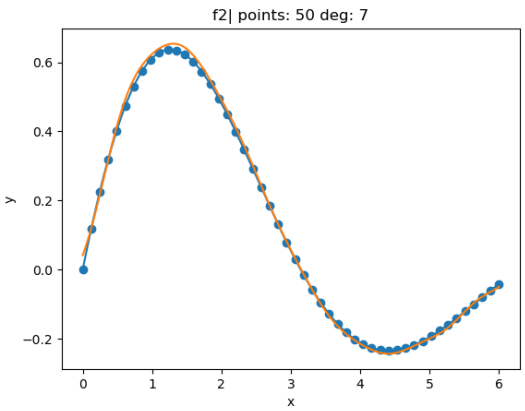

f1| points: 100 deg: 20
x
y

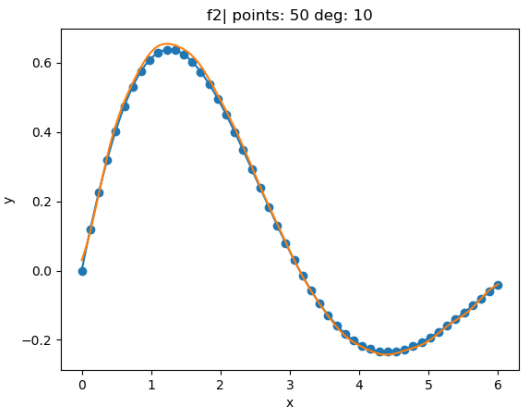

f2| points: 10 deg: 4
x
y

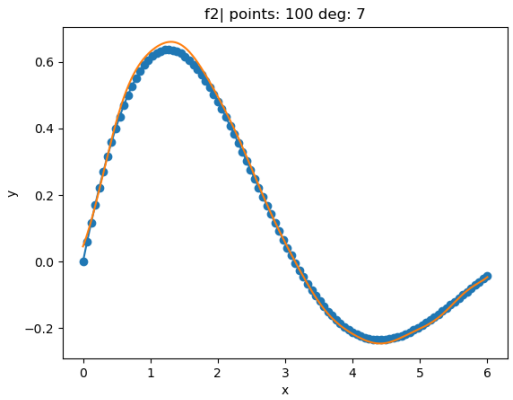

f2| points: 10 deg: 7
x
y

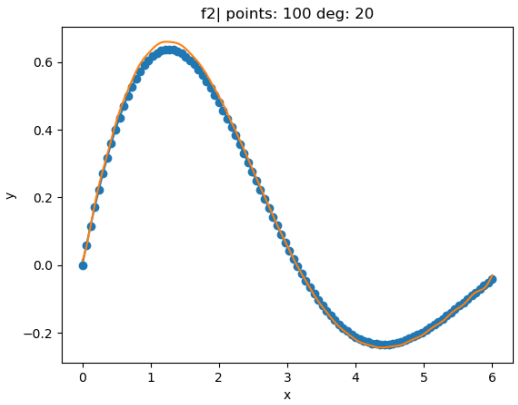

f2| points: 20 deg: 4
x
y


f2| points: 20 deg: 10
x
y


f2| points: 50 deg: 7
x
y


f2| points: 50 deg: 10
x
y





5. Błędy

Wyniki znajdują się poniżej.

T f1: points number: 5, degree: 4, error: 17.7022
T f1: points number: 5, degree: 7, error: 19.9915
T f1: points number: 5, degree: 10, error: 25.3165
T f1: points number: 5, degree: 20, error: 46.118
T f1: points number: 10, degree: 4, error: 3.74668
T f1: points number: 10, degree: 7, error: 8.22552
T f1: points number: 10, degree: 10, error: 20.1562
T f1: points number: 10, degree: 20, error: 48.8832
T f1: points number: 20, degree: 4, error: 0.542384
T f1: points number: 20, degree: 7, error: 0.700572
T f1: points number: 20, degree: 10, error: 0.957602
T f1: points number: 20, degree: 20, error: 47.4174
T f1: points number: 50, degree: 4, error: 5.69283
T f1: points number: 50, degree: 7, error: 5.96739
T f1: points number: 50, degree: 10, error: 5.86462
T f1: points number: 50, degree: 20, error: 4.30755
T f1: points number: 100, degree: 4, error: 15.1215
T f1: points number: 100, degree: 7, error: 15.0638
T f1: points number: 100, degree: 10, error: 13.7137
T f1: points number: 100, degree: 20, error: 7.28167
T f2: points number: 5, degree: 4, error: 0.753388
T f2: points number: 5, degree: 7, error: 2.13876
T f2: points number: 5, degree: 10, error: 3.03939
T f2: points number: 5, degree: 20, error: 7.08523
T f2: points number: 10, degree: 4, error: 0.232518
T f2: points number: 10, degree: 7, error: 0.52587
T f2: points number: 10, degree: 10, error: 4.62574
T f2: points number: 10, degree: 20, error: 9.32761
T f2: points number: 20, degree: 4, error: 0.234114
T f2: points number: 20, degree: 7, error: 0.127911
T f2: points number: 20, degree: 10, error: 0.0670117

```
T f2: points number: 20, degree: 20, error: 6.46342
T f2: points number: 50, degree: 4, error: 0.558873
T f2: points number: 50, degree: 7, error: 0.423275
T f2: points number: 50, degree: 10, error: 0.406109
T f2: points number: 50, degree: 20, error: 0.38698
T f2: points number: 100, degree: 4, error: 1.26887
T f2: points number: 100, degree: 7, error: 1.09106
T f2: points number: 100, degree: 10, error: 1.08793
T f2: points number: 100, degree: 20, error: 1.05777
```

Z uzyskanych danych wynika, że aproksymacja trygonometryczna jest mniej dokładna oraz nie radzi sobie dobrze z funkcjami nieokresowymi. Zwiększenie stopnia wielomianu nie prowadzi do znaczącej poprawy dokładności, a wręcz czasem pogarsza dopasowanie wielomianu do funkcji. Natomiast zauważalne zmniejszenie błędu jest spowodowane przez wzrost liczby punktów wykorzystanych do aproksymacji.