AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Metody Obliczeniowe w Nauce i Technice:

Aproksymacja

Laboratorium 5

Przemysław Lechowicz

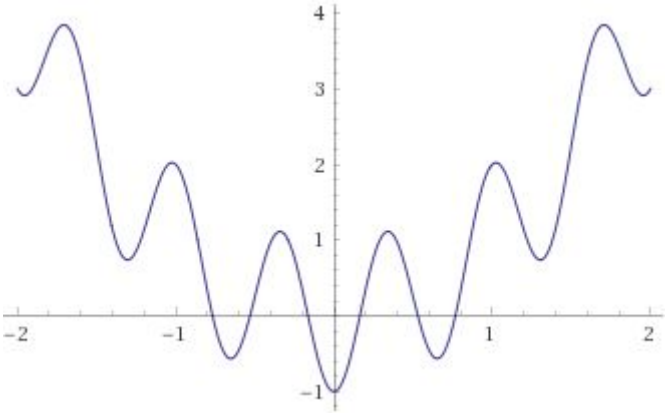
1. Funkcja testowa:

Wybrane przeze mnie funkcje testowe to:

a.

$$x^2 - \cos(3\pi x)$$

Jej wykres w przedziale $<-2;2>$ wygląda następująco:

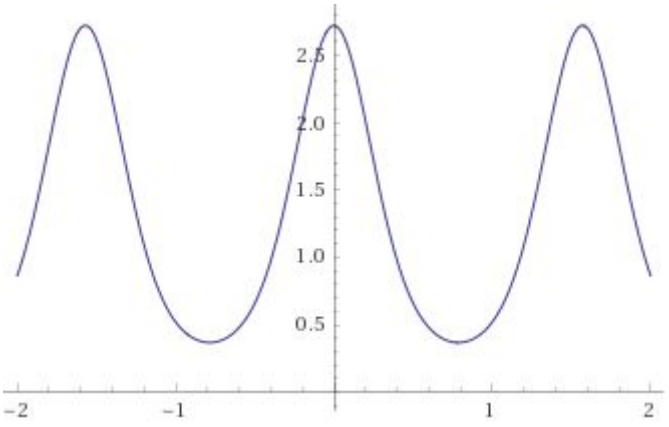



```
double f1(double x) {
	return x * x - cos(3 * M_PI * x);
}
```

b.

$$e^{\cos(4x)}$$

Jej wykres w przedziale $<-2;2>$ wygląda następująco:




```
double f2(double x) {
	return pow(M_E, cos(4 * x));
}
```

2. Aproksymacja średniokwadratowa wielomianami algebraicznymi

W metodzie aproksymacji średniokwadratowej staramy się znaleźć taki wielomian n-tego stopnia, aby jego suma kwadratów różnicy wartości aproksymowanej od aproksymującej w punktach, których wartości znamy, była jak najmniejsza.

a. Implementacja

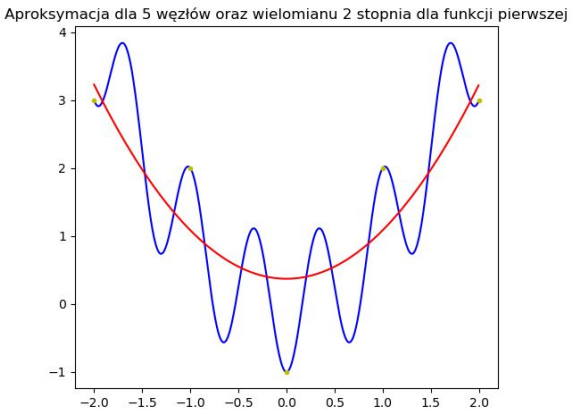
W celu ułatwienia obliczeń macierzowych wykorzystałem bibliotekę NumCpp.

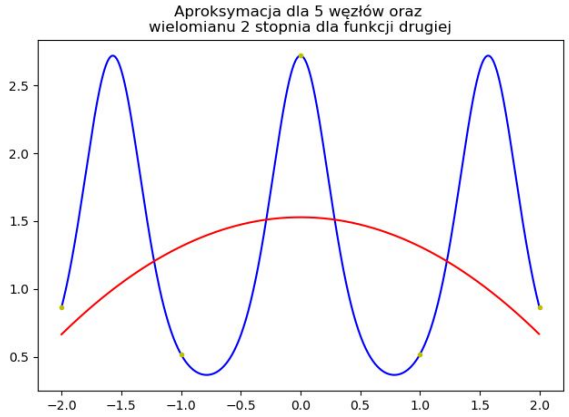
```
std::vector<double> equidistant(double x0, double x1, int n) {
	std::vector<double>result;
	for (int i = 0; i < n; i++) {
		result.push_back(x0 + i * (x1 - x0) / ((double)n - 1));
	}
	return result;
}

nc::NdArray<double> regress(double f(double x), std::vector<double> nodes, int degree) {
	int n = nodes.size();
	nc::NdArray<double> values=nc::zeros<double>(n,1);
	for (int i = 0; i < n; i++) {
		values[i] = f1(nodes[i]);
	}
	nc::NdArray<double> matrix=nc::zeros<double>(n,degree+1);
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < degree + 1; j++) {
			matrix(i,j) += pow(nodes[i], j);
		}
	}

	return nc::linalg::inv(nc::transpose(matrix).dot(matrix))
.dot(nc::transpose(matrix)).dot(values);
}

double poly(nc::NdArray<double> vec, double x) {
	double result = 0;
	for (int i = 0; i < vec.size(); i++) {
		result += vec[i] * pow(x, i);
	}
	return result;
}

int main() {
	int n = 10;
	int degree = 8;
	if (n <= degree) {
		std::cout << "n must be bigger than degree";
		return 0;
	}
```

```
        std::string name = std::to_string(n) + " " + std::to_string(degree) + ".txt";
        int amount = 1000;
        double x0 = -2;
        double x1 = 2;

        std::vector<double> points;
        for (int i = 0; i < amount; i++) {
                points.push_back((x0 + i * (x1 - x0) / amount));
        }
        std::vector<double> values1;
        std::vector<double> values2;
        for (int i = 0; i < amount; i++) {
                values1.push_back(f1(points[i]));
                values2.push_back(f2(points[i]));
        }
        std::vector<double> nodes = equidistant(x0, x1, n);

        std::vector<double> regressed1;
        std::vector<double> regressed2;
        for (int i = 0; i < amount; i++) {
                regressed1.push_back(poly(regress(f1, nodes, degree), points[i]));
                regressed2.push_back(poly(regress(f2, nodes, degree), points[i]));
        }
```

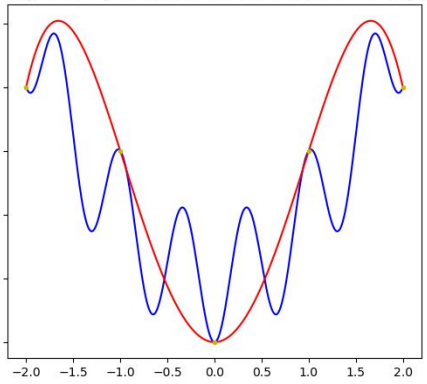
b. Wykresy uzyskane dla różnej liczby węzłów. Na każdym z wykresów wykres niebieski to wykres funkcji aproksymowanej, czerwony to wykres funkcji aproksymującej, a żółte punkty to węzły aproksymacji. Po lewej stronie znajdują się wykresy dla pierwszej funkcji, po prawej dla drugiej.

Wykresy były rysowane na podstawie 1000 równoodległych punktów w podanym przedziale, dla których liczyłem wartość aproksymowaną.

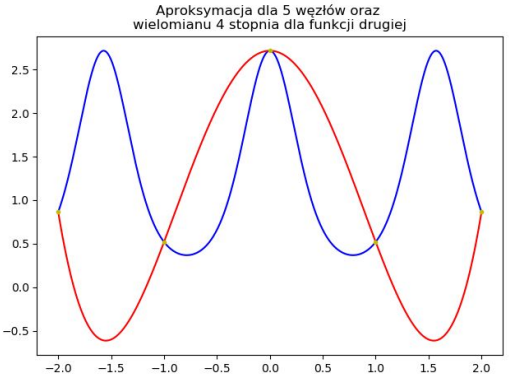




Aproksymacja dla 5 węzłów oraz wielomianu 4 stopnia dla funkcji pierwszej

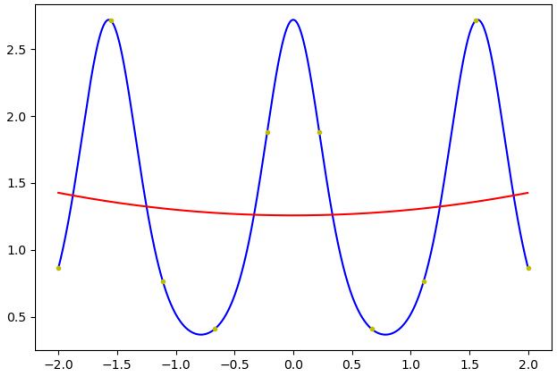


Aproksymacja dla 5 węzłów oraz
wielomianu 4 stopnia dla funkcji drugiej

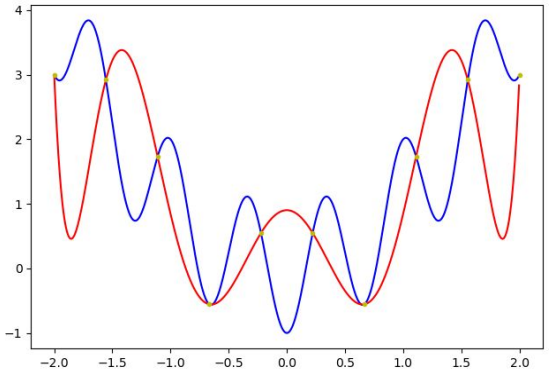


Aproksymacja dla 10 węzłów oraz
wielomianu 2 stopnia dla funkcji pierwszej

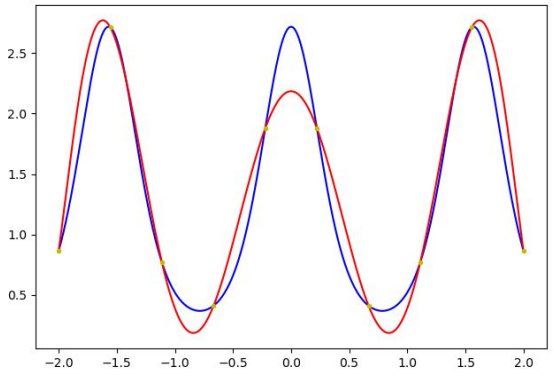
Aproksymacja dla 10 węzłów oraz
wielomianu 2 stopnia dla funkcji drugiej



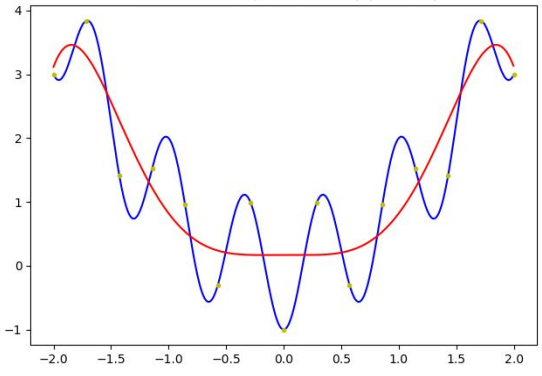
Aproksymacja dla 10 węzłów oraz
wielomianu 9 stopnia dla funkcji pierwszej



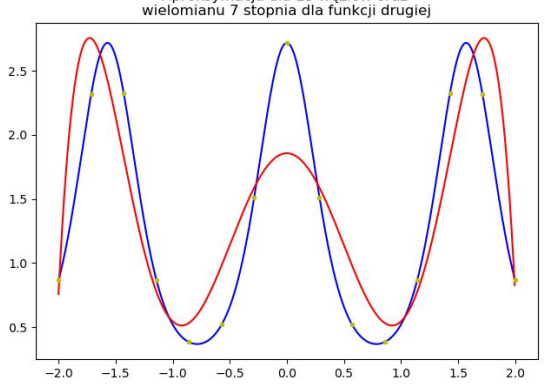
Aproksymacja dla 10 węzłów oraz
wielomianu 9 stopnia dla funkcji drugiej

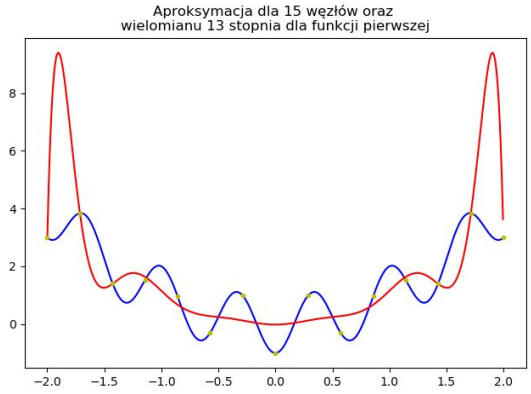


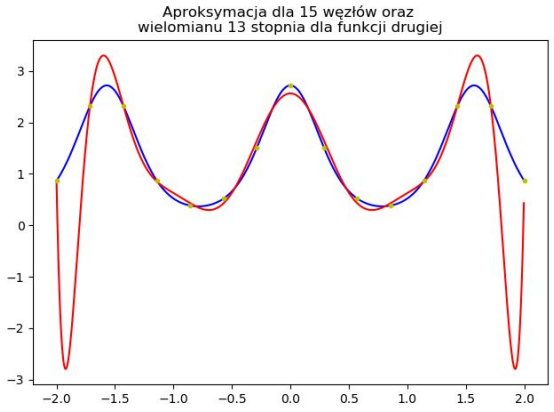
Aproksymacja dla 15 węzłów oraz
wielomianu 7 stopnia dla funkcji pierwszej

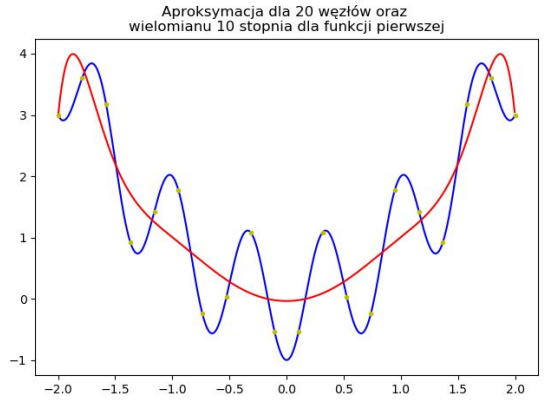


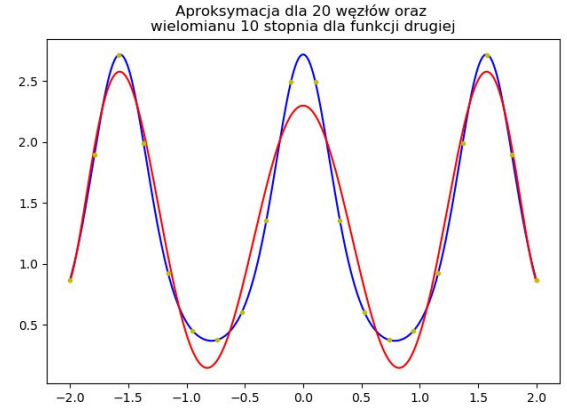
Aproksymacja dla 15 węzłów oraz
wielomianu 7 stopnia dla funkcji drugiej

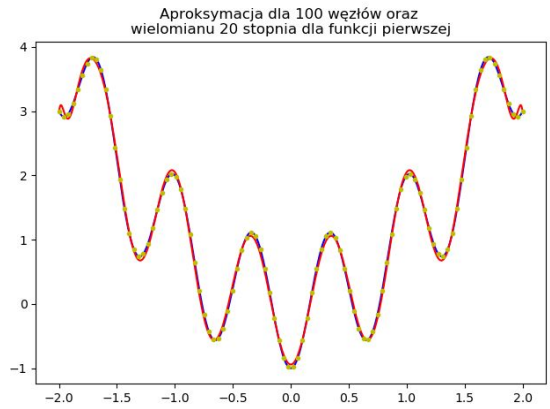


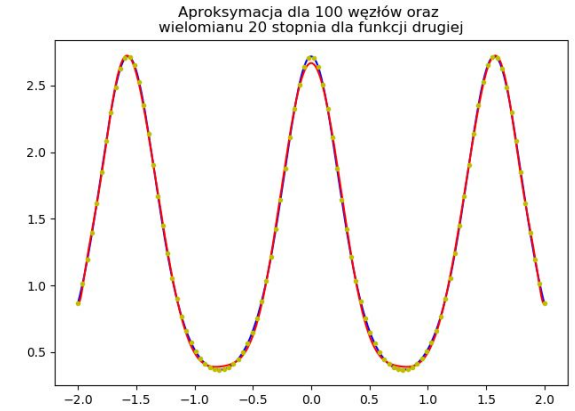












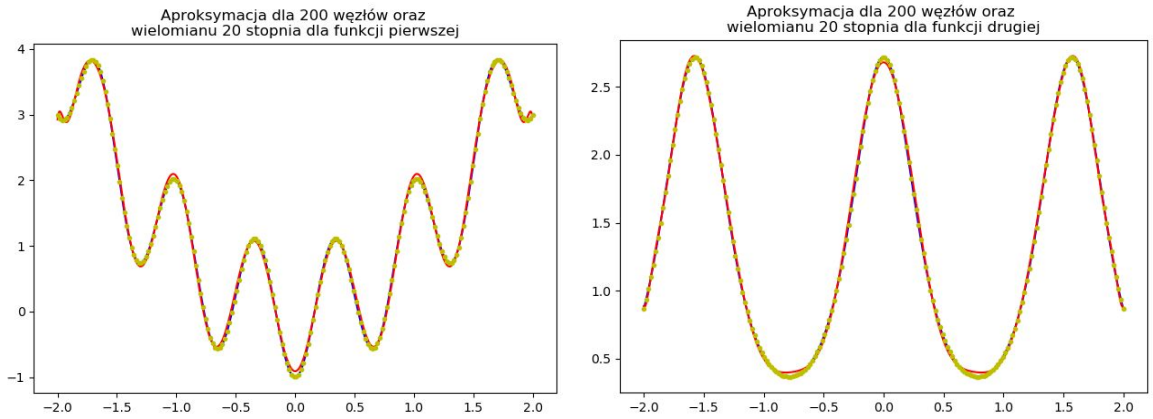
c. Błędy były wyliczane jako pierwiastek średniej z sumy kwadratów różnicy wartości funkcji aproksymującej i aproksymującej w 1000 punktów.

```
double error(std::vector<double> x, std::vector<double> y) {
	double error = 0;
	for (int i = 0; i < x.size(); i++) {
		error += ((x[i] - y[i]) * (x[i] - y[i]));
	}
	error = error / x.size();
	return sqrt(error);
}
```

Liczba węzłów	Liczba funkcji bazowych	Błąd pierwszej funkcji	Błąd drugiej funkcji
3	2	1,22474	1,32662
5	2	0,776739	0,949021
5	3	0,776739	0,949021
5	4	1,19546	1,67562
8	2	0,889551	0,836203
8	3	0,889551	0,836203
8	4	1,00829	0,833931
8	5	1,00829	0,833931
10	2	0,732787	0,812102
10	8	1,44845	0,226982
15	2	0,717521	0,797107
15	3	0,717521	0,797107
15	4	0,70908	0,8012
15	10	0,951209	0,190383
15	12	1,79635	1,01288
20	2	0,713442	0,790159
20	3	0,713442	0,790159
20	4	0,706738	0,794833
20	5	0,706738	0,794833
20	15	2,39415	0,30499
25	2	0,711374	0,786723
25	20	4,49955	2,14712
50	2	0,708174	0,781798
50	3	0,708174	0,781798

50	20	0,0774388	0,0270203
100	2	0,707213	0,780467
100	15	0,566675	0,071209
100	20	0,0514118	0,0210595
200	20	0,0501307	0,0209095

Najmniejszy błąd zarówno dla pierwszej jak i drugiej funkcji występuje dla 200 węzłów i 20 funkcji bazowych. Ich wykresy wyglądają wtedy następująco:



Wykresy są bardzo zbliżone do pierwowzorów, jednak nie są idealne, co widać w okolicach ekstremów.

Warto również zauważyć, że dla małej liczby funkcji bazowych funkcja pierwsza posiadała mniejszy błąd od funkcji drugiej, im jednak liczba funkcji bazowych rosła tym lepsza stawała się dokładność aproksymacji funkcji drugiej w porównaniu do pierwszej.

Po przedstawionych wyżej wykresach i tabeli można zauważyć, że zarówno zwiększenie liczby funkcji bazowych jak i liczby węzłów aproksymacji zmniejsza wartość błędu, jednak zwiększenie liczby węzłów nie ma aż tak znacznego wpływu na obniżenie wartości błędu jak zwiększenie liczby funkcji bazowych.

3. Aproksymacja średniokwadratowa trygonometryczna

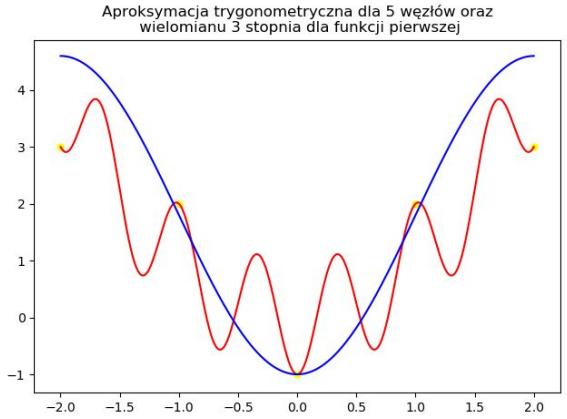
W tym przypadku szukamy takiej kombinacji funkcji sin(nx) i cos(nx) aby aby jej suma kwadratów różnicy wartości aproksymowanej od aproksymującej w punktach, których wartości znamy, była jak najmniejsza.

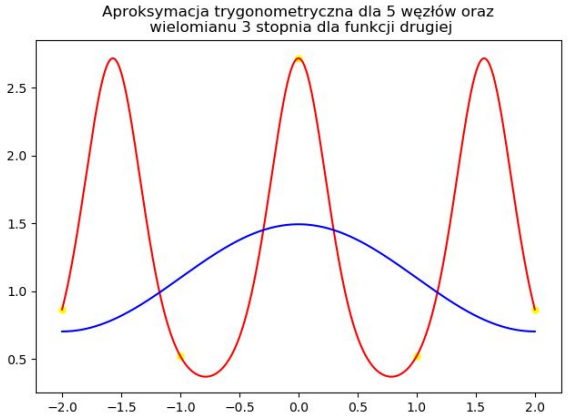
a. Implementacja:

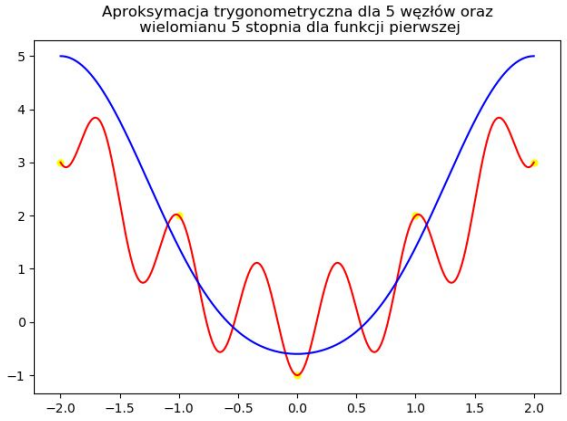
```
double approximate(int degree, std::vector <double> nodes, std::vector <double> yNodes,
double f(double x), double x) {
	int n = yNodes.size();

	double a0 = 0.0;
	for (int i = 0; i < n; i++) {
		a0 += yNodes[i];
	}
	a0 /= n;

	std::vector <double> a_coefficients;
	std::vector <double> b_coefficients;
	double t = 2 * M_PI / (n - 1);
	for (int j = 1; j < degree + 1; j++) {
		double a = 0.0;
		double b = 0.0;

		for (int i = 0; i < n; i++) {
			a += yNodes[i] * cos(t * (double)j * (double)i);
			b += yNodes[i] * sin(t * (double)j * (double)i);
		}
		a = a * (2 / (double)n);
		b = b * (2 / (double)n);

		a_coefficients.push_back(a);
		b_coefficients.push_back(b);
	}
	double mapping_a = 2 * M_PI / 4.0;
	double mapping_b = 0;

	x = mapping_a * x  + M_PI;
	double result = a0;
	for (int k = 0; k < a_coefficients.size(); k++) {
		result += a_coefficients[k] * cos((k + 1) * x) + b_coefficients[k] *
sin((k + 1) * x);
	}
	return result;
}
```

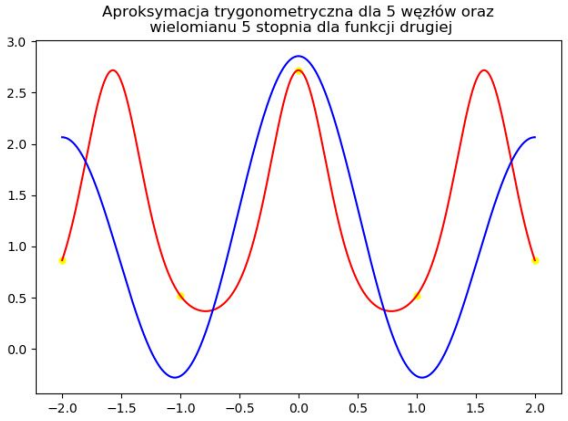
c. Wykresy uzyskane dla różnej liczby węzłów. Na każdym z wykresów wykres niebieski to wykres funkcji aproksymowanej, czerwony to wykres funkcji aproksymującej, a żółte punkty to węzły aproksymacji. Po lewej stronie znajdują się wykresy dla pierwszej funkcji, po prawej dla drugiej.

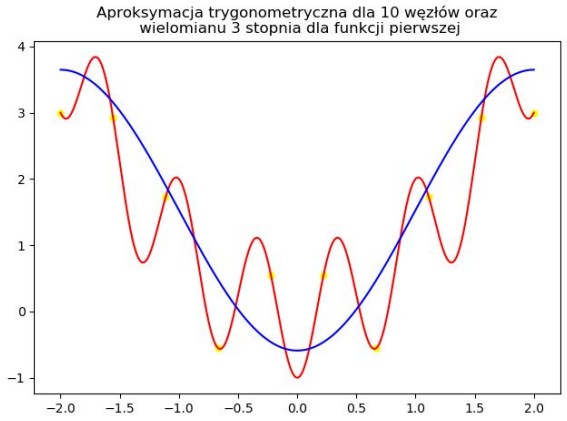
Wykresy były rysowane na podstawie 1000 równoodległych punktów w podanym przedziale, dla których liczyłem wartość aproksymowaną.

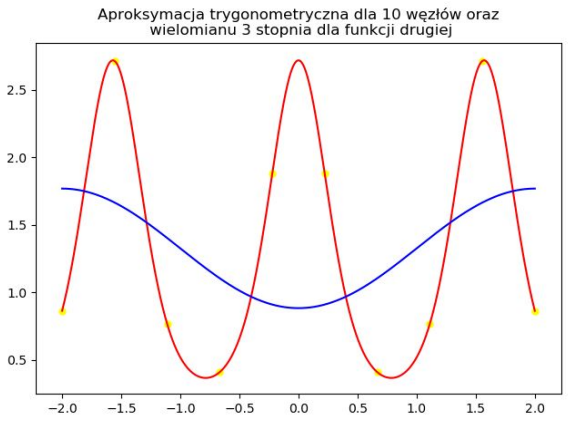


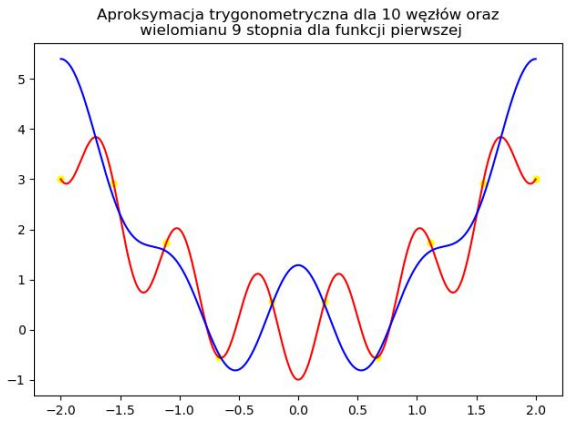


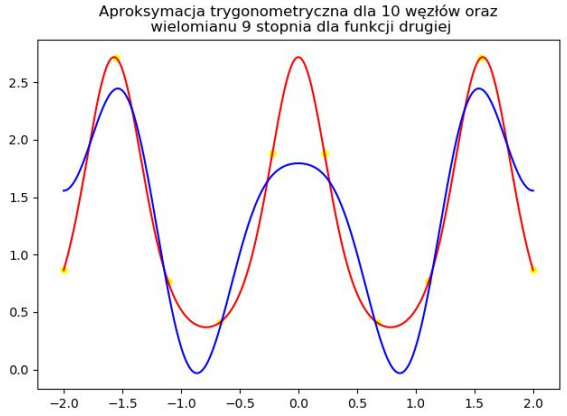


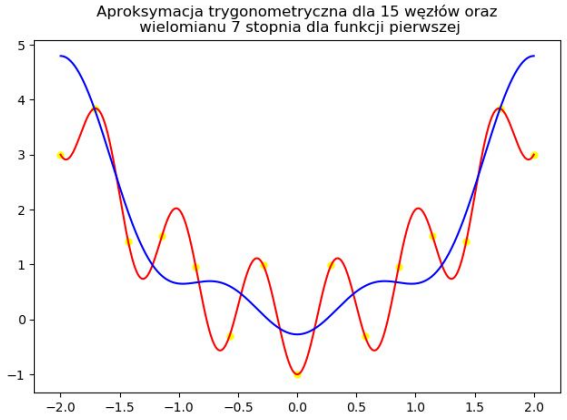


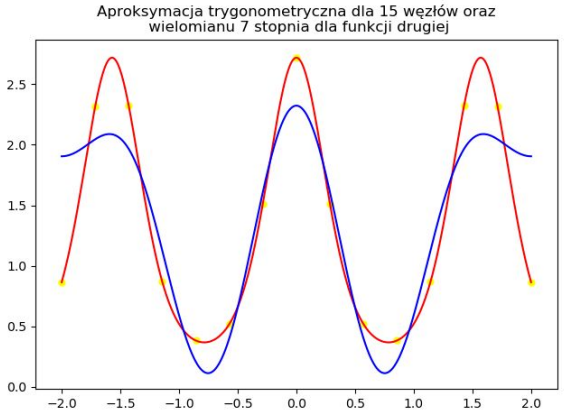


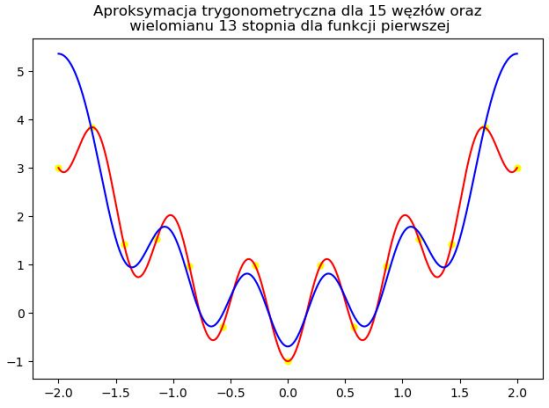


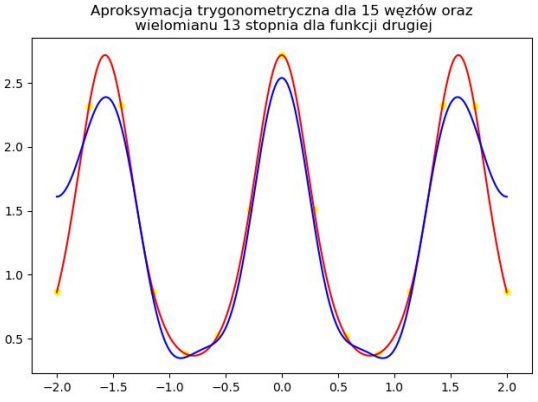

Aproksymacja trygonometryczna dla 10 węzłów oraz
wielomianu 9 stopnia dla funkcji pierwszej

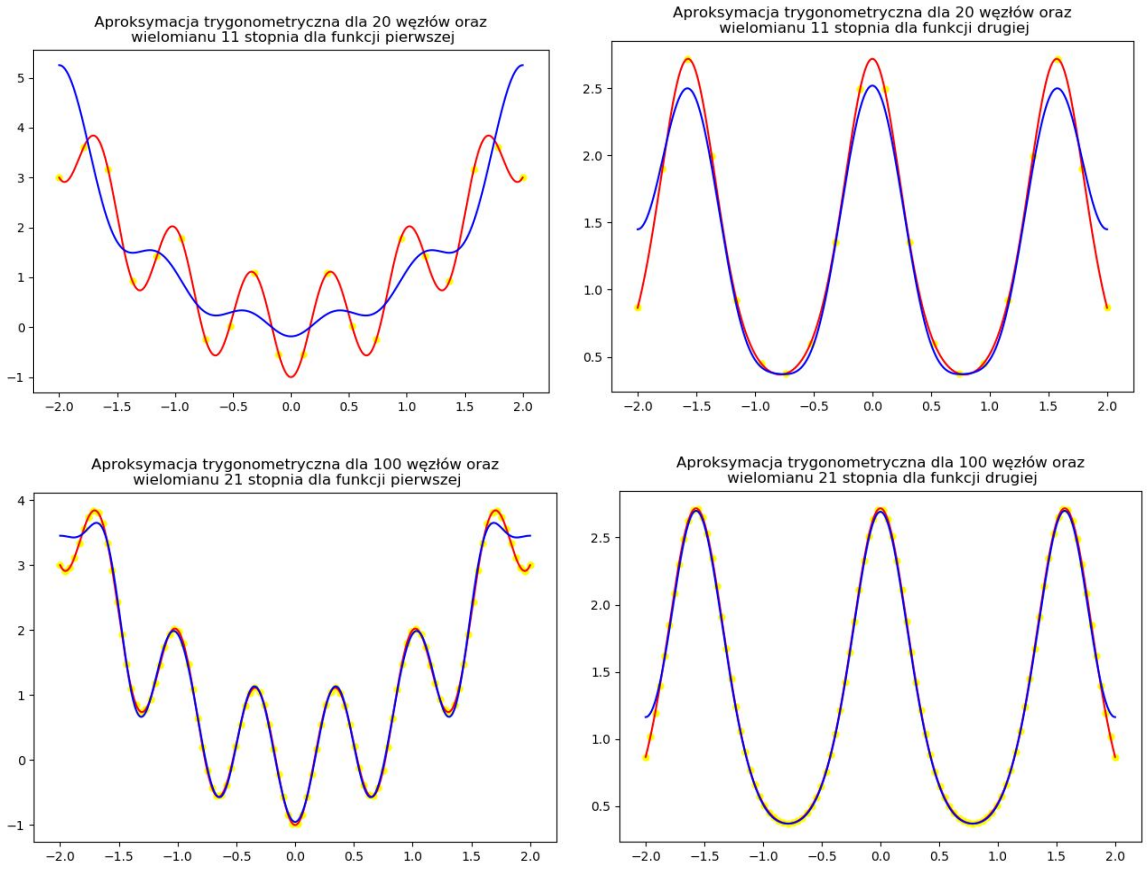

Aproksymacja trygonometryczna dla 10 węzłów oraz
wielomianu 9 stopnia dla funkcji drugiej


Aproksymacja trygonometryczna dla 15 węzłów oraz
wielomianu 7 stopnia dla funkcji pierwszej


Aproksymacja trygonometryczna dla 15 węzłów oraz
wielomianu 7 stopnia dla funkcji drugiej


Aproksymacja trygonometryczna dla 15 węzłów oraz
wielomianu 13 stopnia dla funkcji pierwszej


Aproksymacja trygonometryczna dla 15 węzłów oraz
wielomianu 13 stopnia dla funkcji drugiej



d. Błędy były wyliczane jako pierwiastek średniej z sumy kwadratów różnicy wartości funkcji aproksymującej i aproksymującej w 1000 punktów.

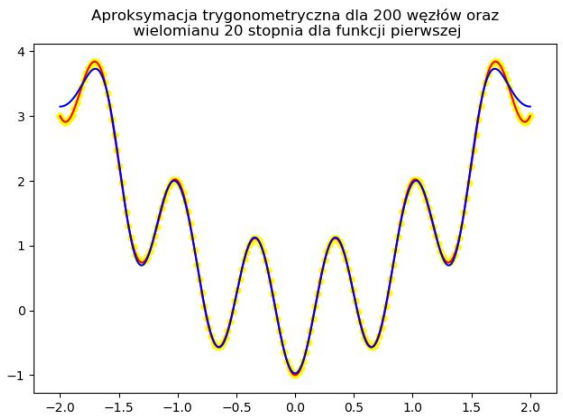
Błędy były wyliczane tą samą funkcją co w przypadku aproksymacji wielomianami.

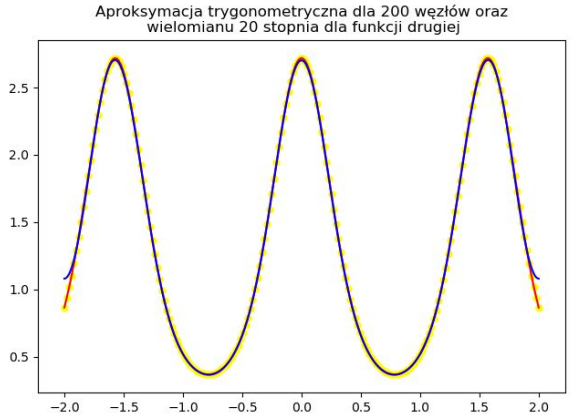
Liczba węzłów	Liczba funkcji bazowych	Błąd pierwszej funkcji	Błąd drugiej funkcji
3	2	3.09053	1.84144
5	2	1.18163	0.962242
5	3	2.19716	0.843242
5	4	3.31423	1.89988
8	2	0.950227	0.650839
8	3	1.12642	0.390047
8	4	1.31135	0.474579
8	5	1.54616	0.716978
10	2	0.917135	0.621211

10	8	2.00067	0.764495
15	2	0.792058	0.61505
15	3	0.836253	0.336971
15	4	0.886488	0.308706
15	10	1.00232	0.273671
15	12	1.22475	0.736813
20	2	0.748312	0.613191
20	3	0.769182	0.327732
20	4	0.797481	0.294027
20	5	0.827254	0.143147
20	15	0.97585	0.286821
25	2	0.728215	0.612225
25	20	0.946746	0.293989
50	2	0.701854	0.61076
50	3	0.695602	0.315172
50	20	0.391924	0.108569
100	2	0.69543	0.610326
100	15	0.166608	0.0499553
100	20	0.193584	0.056599
200	20	0.096297	0.0194202

W tym przypadku za każdym razem dokładniejsza okazywała się aproksymacja dla funkcji drugiej. Stało się tak prawdopodobnie dlatego, ponieważ funkcja druga jest okresowa, a kształtem przypomina funkcje trygonometryczne takie jak sinus i cosinus. Prawdopodobnie dlatego aproksymowanie jej wielomianami trygonometrycznymi dało lepszy rezultat niż dla funkcji pierwszej.

Ponownie najlepszą dokładność osiągnąłem dla 200 węzłów i 20 stopnia wielomianu. Wykresy dla tych danych wyglądają następująco:





W porównaniu do aproksymacji wielomianami algebraicznymi w wielu przypadkach dla tej samej liczby węzłów i stopnia wielomianu błędy są podobne, czasem błąd jest niewiele mniejszy na korzyść wielomianów algebraicznych, w większości przypadków jednak błędy są mniejsze dla wielomianów trygonometrycznych. Dla przykładu dla 25 węzłów i 20 stopnia dla funkcji drugiej błąd aproksymacji wielomianami algebraicznymi wyniósł 2.14, kiedy dla wielomianów trygonometrycznych wyniósł 0.29.