AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Metody Obliczeniowe w Nauce i Technice:

Interpolacja

Laboratorium 4

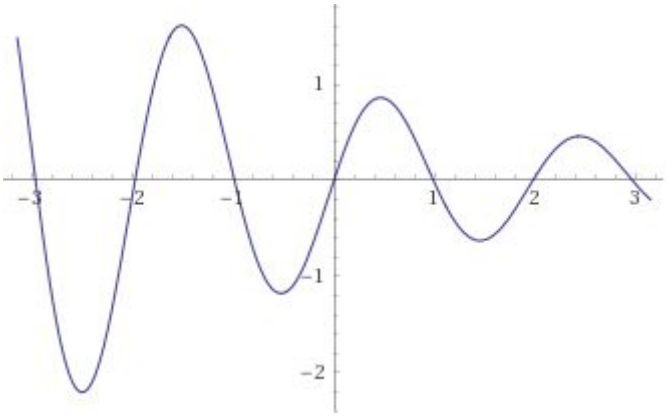
Przemysław Lechowicz

1. Funkcja testowa:

Wybrana przeze mnie funkcja testowa to:

$$f(x) = \sin\left(\frac{10x}{\pi}\right) \cdot e^{\frac{-x}{\pi}}$$

Jej wykres w przedziale $<-\pi, \pi>$ wygląda następująco:




```
double f1(double x) {
        return sin((10 * x) / M_PI) * pow(M_E, (-x) / M_PI);
}
```

2. Interpolacja Lagrange'a

Interpolacja Lagrange'a jest to metoda interpolacji wielomianem Lagrange'a stopnia n przyjmującym w n+1 punktach, zwanych węzłami interpolacji, wartości takie same jak przybliżana funkcja.

a. Implementacja

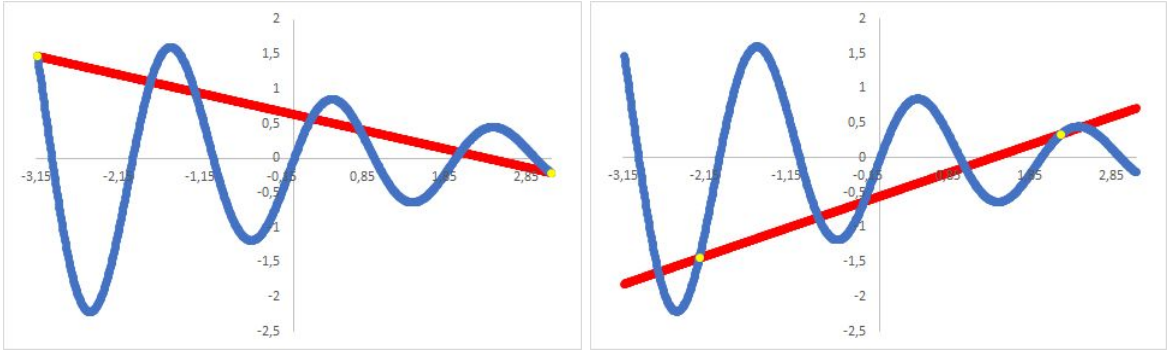
```
std::vector < Point > lagrange(int pointsNumber, int steps, bool chebyshew) {
  std::vector < Point > results;
  std::vector < Point > values;
  if (chebyshew) {
    for (int k = pointsNumber - 1; k >= 0; k--) {
      Point p;
      p.setX(M_PI * cos((M_PI * (2 * (double) k + 1)) / (2 * ((double) pointsNumber))));
      p.setY(f1(p.getX()));
      values.push_back(p);
    }
  } else {
    double interval = (2 * M_PI) / ((double)(pointsNumber - 1));
    for (int i = 0; i < pointsNumber; i++) {
      Point p;
      p.setX(-M_PI + interval * i);
      p.setY(f1(p.getX()));
```

```
      values.push_back(p);
      plik << p.getX() << " " << p.getY() << std::endl;
    }
  }
  std::cout << values.size();
  plik.close();
  double interval = 2 * M_PI / ((double)(steps - 1));
  int n = values.size();
  for (int i = 0; i < steps; i++) {
    Point p;
    p.setX(interval * i - M_PI);
    double result = 0;
    for (int i = 0; i < n; i++) {
      double yi = values[i].getY();
      for (int j = 0; j < n; j++) {
        if (i != j) {
          yi = yi * (p.getX() - values[j].getX()) / (values[i].getX() -
values[j].getX());
        }
      }
      result += yi;
    }
    p.setY(result);
    results.push_back(p);
  }
  return results;
}
```

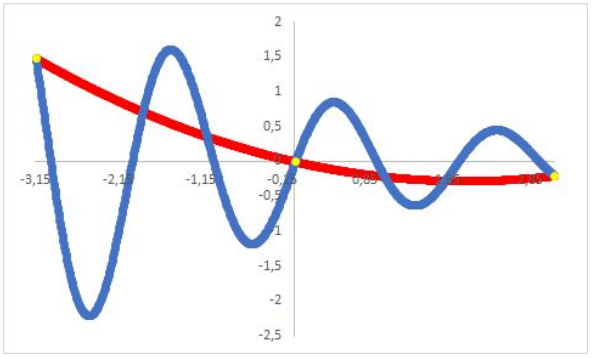
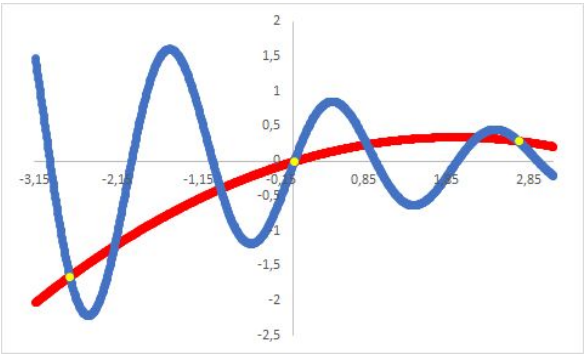
b. Wykresy uzyskane dla różnej liczby węzłów, po lewej dla węzłów równoodległych, po prawej dla węzłów Czebyszewa. Na każdym z wykresów wykres niebieski to wykres funkcji interpolowanej, czerwony to wykres funkcji interpolującej, a żółte punkty to węzły interpolacji.

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

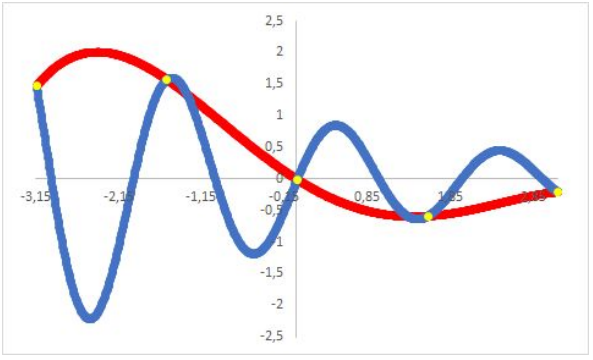
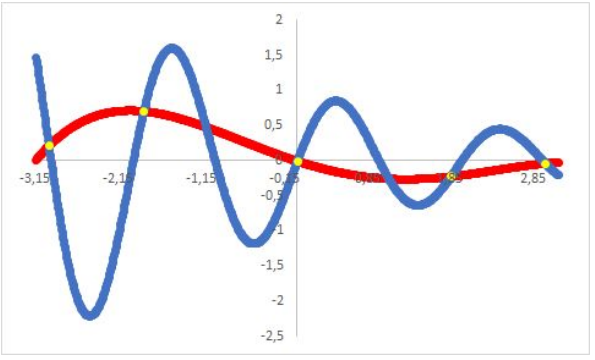
Dla 2 węzłów:



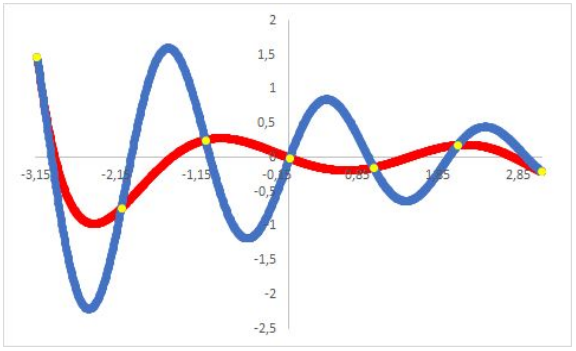
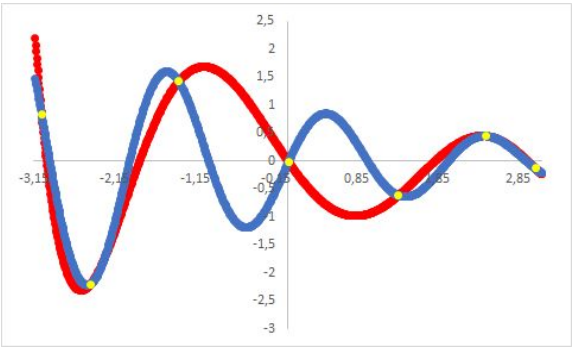
Dla 3 węzłów:

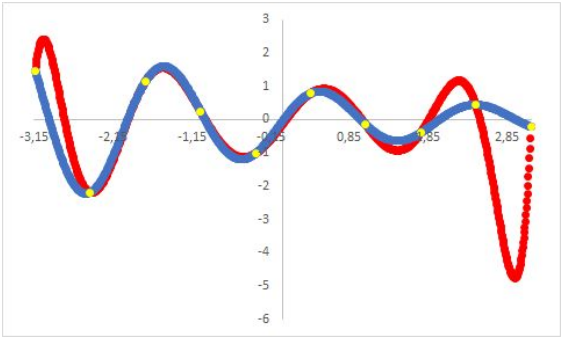
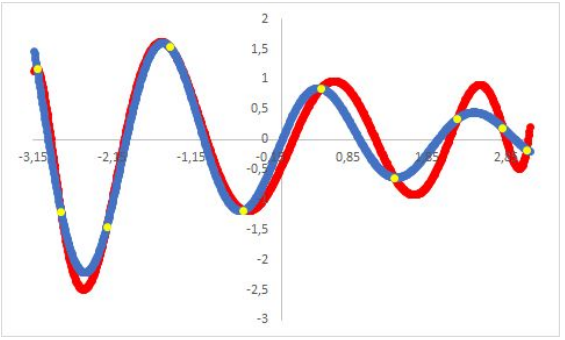
Dla 5 węzłów:

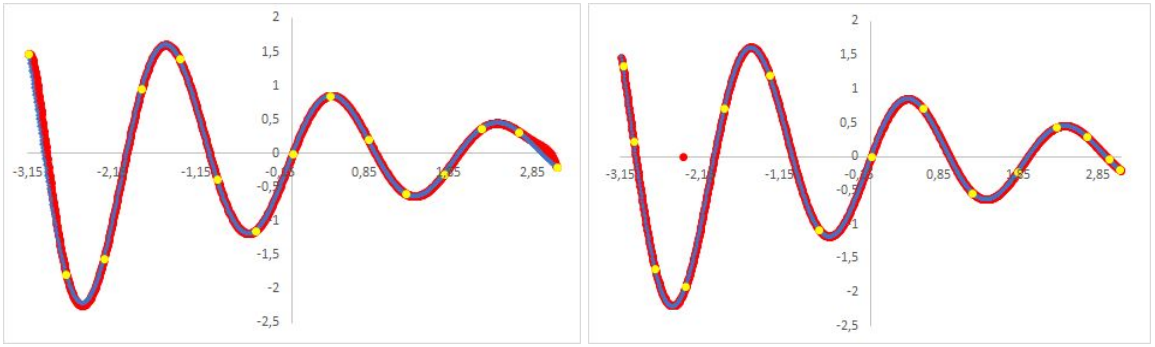
Dla 7 węzłów:

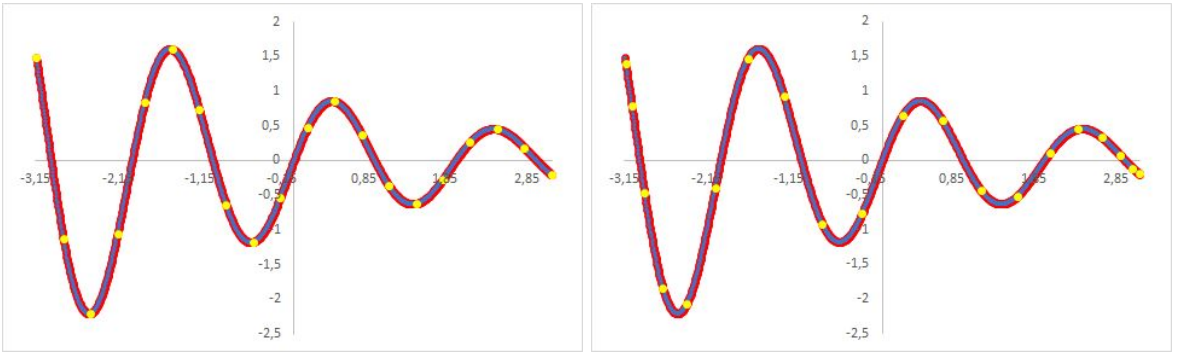
Dla 10 węzłów:

Dla 15 węzłów:



Dla 20 węzłów:



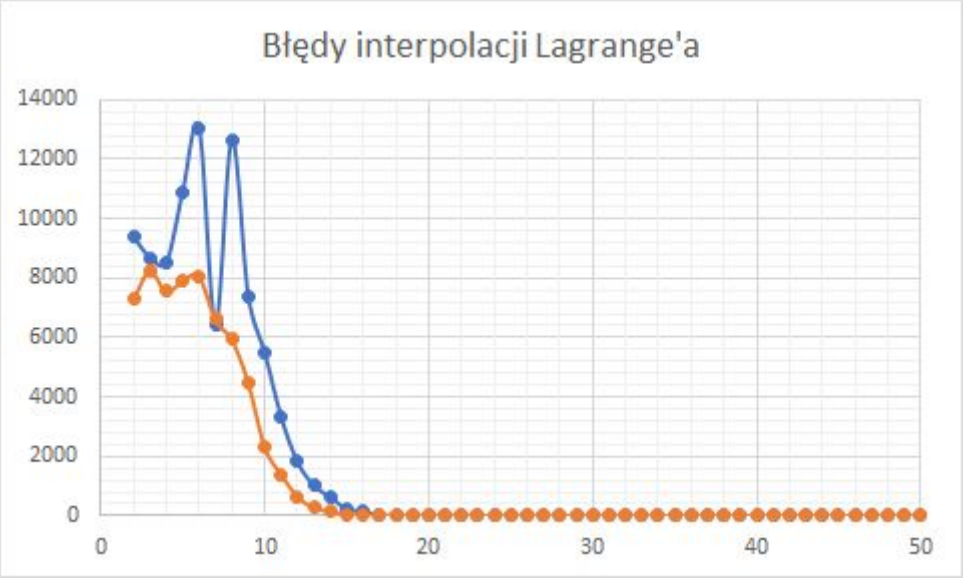
c. Błędy uzyskane przy użyciu metody Lagrange'a, po lewej dla węzłów równoodległych, po prawej dla węzłów Czebyszewa.

Funkcja znajdująca błąd porównuje 1000 punktów funkcji interpolowanej i interpolującej:

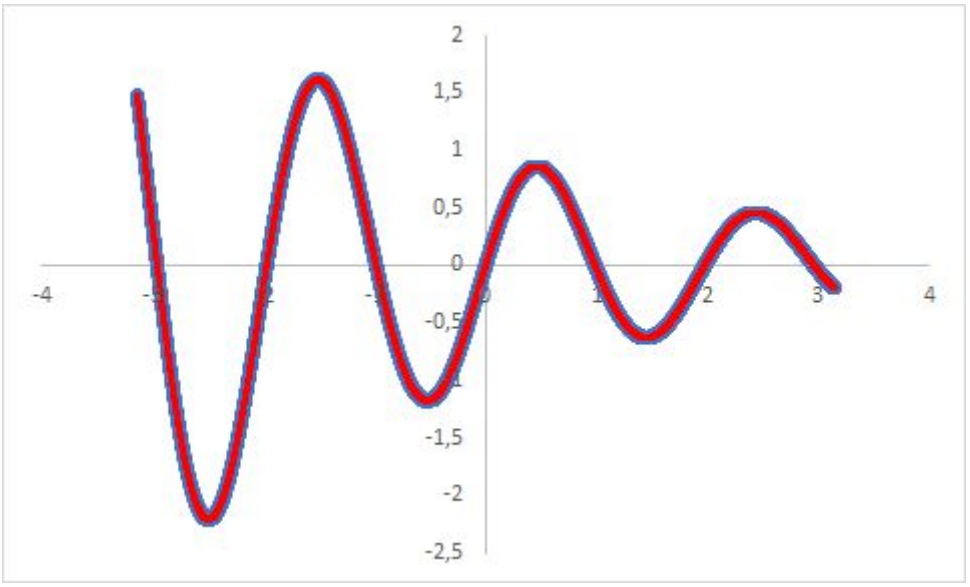
```
double lagrangeError(int n, bool chebyshev) {
    std::vector < Point > lag = lagrange(n, 1000, chebyshev);
    double error = 0.0;
    double compartment = 2 * M_PI / 1000;
    double value = -M_PI;
    for (int i = 0; i < 1000; i++) {
      error += std::abs(f1(value - lag[i].getY()); value += compartment;
      }
      return error;
    }
```

Liczba węzłów	Wartość błędu dla węzłów równoodległych	Wartość błędu dla węzłów Czebyszewa
2	9395,08	7288,47
3	8680,06	8270,24
5	10893,1	7942,46
6	13053,1	8067,12
7	6450,21	6608,53
8	12655,2	5937,19
10	5497,23	2313,33
15	231,174	41,8622
20	3,25058	0,354071
30	3,08E-06	1,34E-07
40	4,24E-05	5,49E-10
45	0,00105052	6,34E-09
50	0,0276196	1,58E-08

Poniżej przedstawiam wykres przedstawiający wartość błędu dla kolejnych węzłów. Kolorem pomarańczowym są oznaczone wartości błędu dla węzłów Czebyszewa, niebieskim dla węzłów równoodległych.

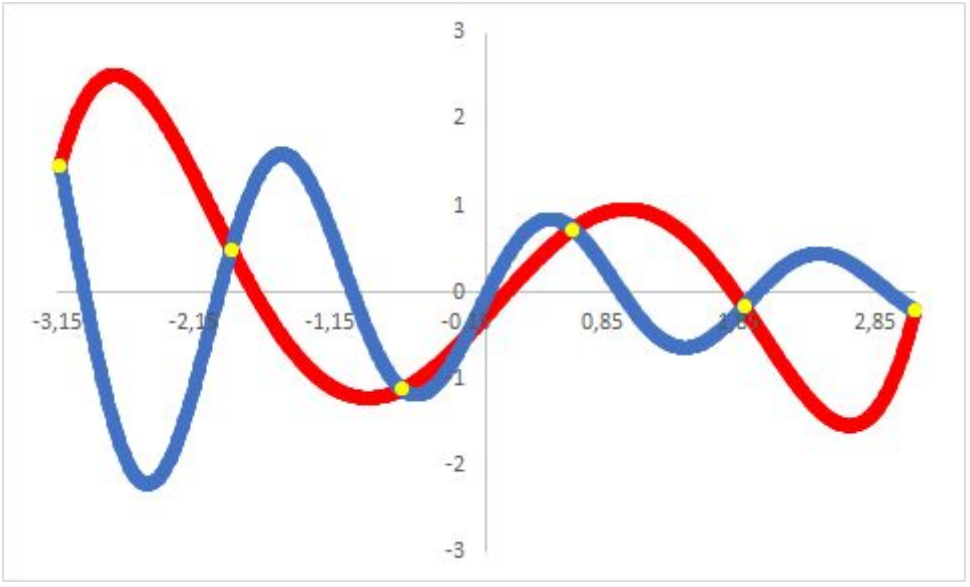


Najmniejszy błąd występuje dla 50 węzłów Czebyszewa (błędy były liczone dla liczby węzłów od 2 do 50 zarówno węzłów równoodległych jak i Czebyszewa). Jego wykres wygląda następująco:



Czerwona linia funkcji interpolującej pokrywa on się prawie idealnie z niebieską linią funkcji interpolowanej.

Możemy zauważyć, że w prawie każdym przypadku wartość błędu była mniejsza dla węzłów Czebyszewa. Największy błąd wystąpił dla 6 równoodległych węzłów. Tak wygląda jego wykres:



Dla węzłów równoodległych efekt Rungego jest widoczny i znacznie zmienia dokładność uzyskanych wyników, podczas gdy dla węzłów Czebyszewa wystąpił rzadziej i nie powodował aż tak wysokich skoków obliczonego błędu jak dla węzłów równoodległych.

3. Interpolacja Newtona

Jest to metoda interpolacji wykorzystująca wzór interpolacyjny Newtona.

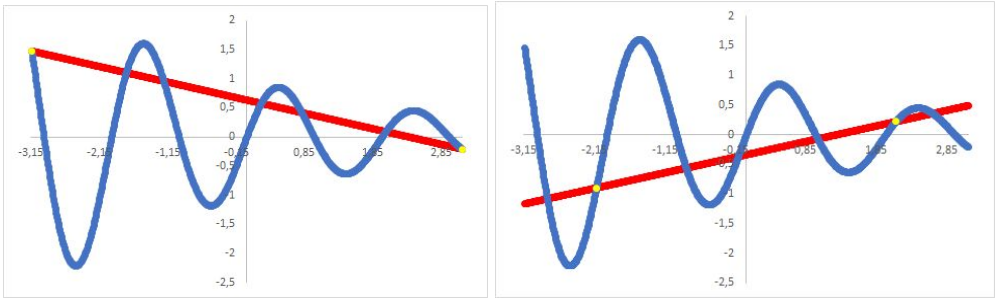
a. Implementacja

```
std::vector < Point > NewtonInterpolation(int pointsNumber, bool chebyshew) {
  std::vector < Point > results;
  std::vector < Point > values;

  if (chebyshew) {
    for (int i = pointsNumber - 1; i >= 0; i--) {
      Point p;
      p.setX(M_PI * cos((M_PI * (2 * i + 1)) / (2 * (pointsNumber))));
      p.setY(f1(p.getX()));
      values.push_back(p);
    }
  } else {
    double interval = 2 * M_PI / (pointsNumber - 1);
    for (int i = 0; i < pointsNumber; i++) {
      Point point;
      point.setX(-M_PI + interval * i);
      point.setY(f1(point.getX()));
      values.push_back(point);
    }
  }
  double interval = 2 * M_PI / (1000);
  for (int i = 0; i < 1001; i++) {
    Point p;
    p.setX(interval * i - M_PI);
    std::vector < std::vector < double >> result;
    std::vector < double > empty_vec;
    empty_vec.resize(pointsNumber, -1.0);
    result.resize(pointsNumber, empty_vec);

    for (int j = 0; j < pointsNumber; j++) {
      result[j][0] = values[j].getY();
    }

    for (int j = 1; j < pointsNumber; j++) {
      for (int k = 0; k < pointsNumber - j; k++) {
        result[k][j] = 0.0;
        result[k][j] = (result[k][j - 1] - result[k + 1][j - 1]) /
          (values[k].getX() - values[j + k].getX());
      }
    }
    std::vector < std::vector < double >> y = result;
    double sum = y[0][0];
    for (int j = 1; j < pointsNumber; j++) {
      double result = 1;
      for (int k = 0; k < j; k++) {
```

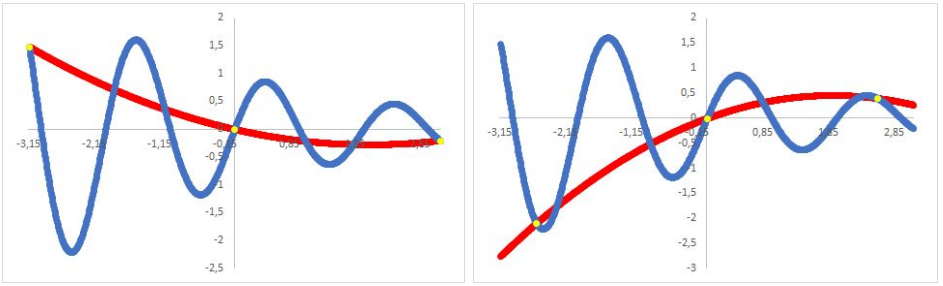
```
        result = result * (p.getX() - values[k].getX());
      }
      sum = sum + (result * y[0][j]);
    }
    p.setY(sum);
    results.push_back(p);
  }
  return results;
}
```

b. Wykresy uzyskane dla różnej liczby węzłów, po lewej dla węzłów równoodległych, po prawej dla węzłów Czebyszewa. Na każdym z wykresów wykres niebieski to wykres funkcji interpolowanej, czerwony to wykres funkcji interpolującej, a żółte punkty to węzły interpolacji.

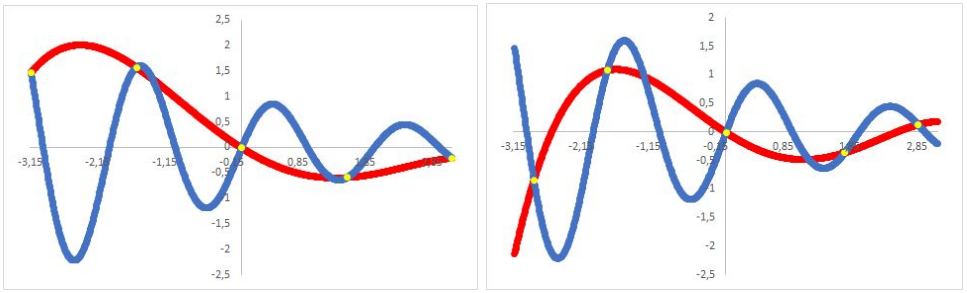
Dla 2 węzłów:



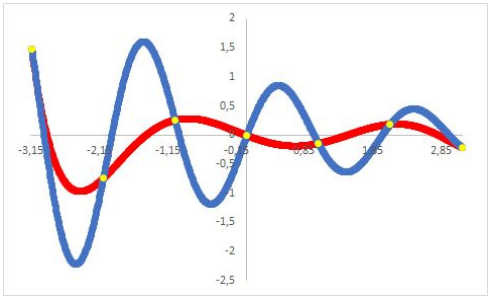
Dla 3 węzłów:

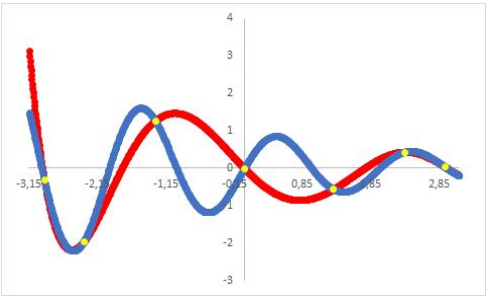


Dla 5 węzłów:

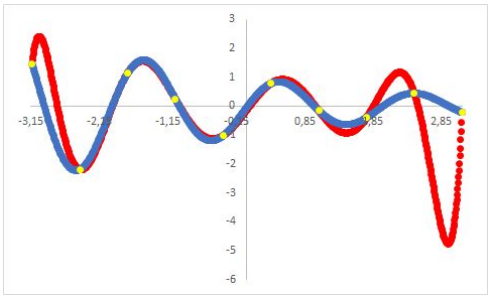


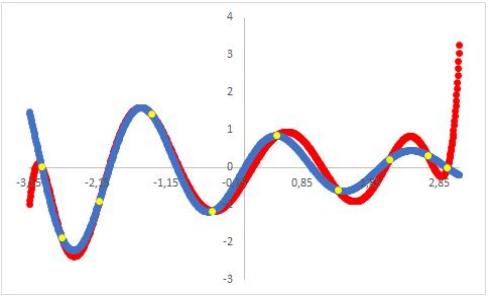
Dla 7 węzłów:



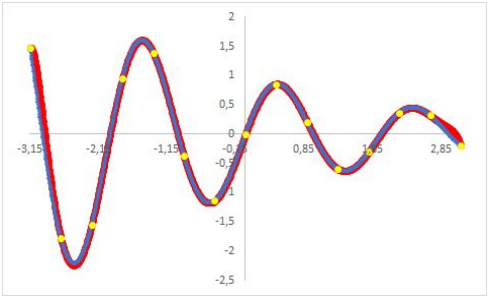


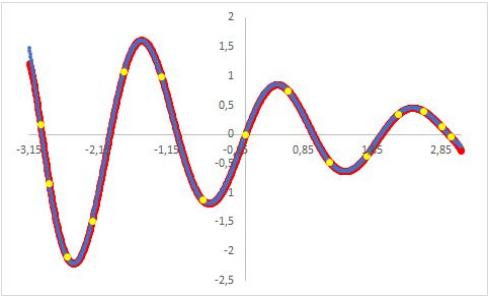
Dla 10 węzłów:



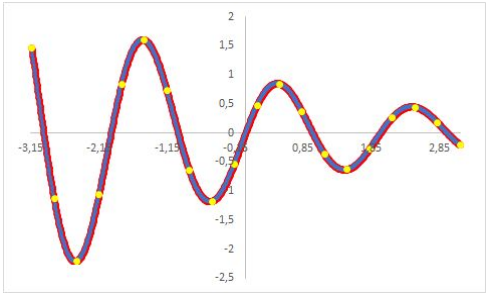


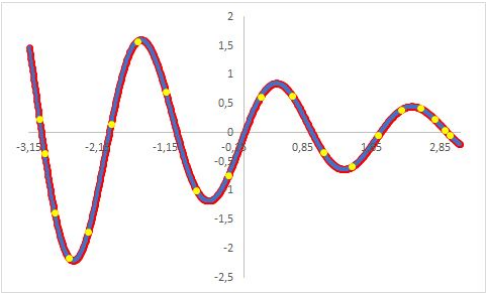
Dla 15 węzłów:





Dla 20 węzłów:



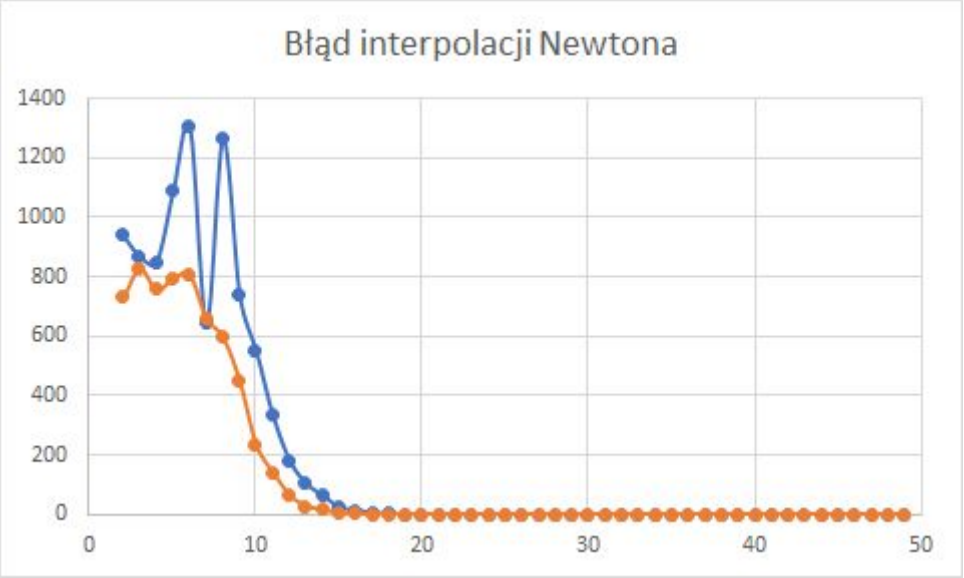


c. Błędy uzyskane przy użyciu metody Newton'a, po lewej dla węzłów równoodległych, po prawej dla węzłów Czebyszewa.

```
double newtonError(int n, bool chebyshev) {
  double error = 0.0;
  std::vector < Point > results;
  results = NewtonInterpolation(n, chebyshev);
  for (int i = 0; i < 1001; i++) {
    error += std::abs(f1(i * ((2 * M_PI) / 1000) - M_PI) - results[i].getY());
  }
  return error;
}
```

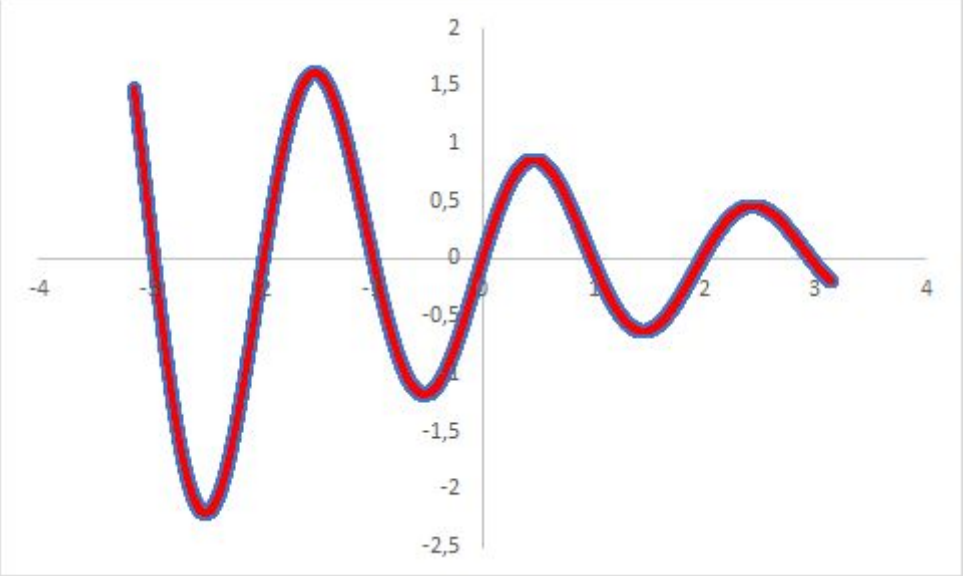
Liczba węzłów	Wartość błędu dla węzłów równoodległych	Wartość błędu dla węzłów Czebyszewa
2	939,504	730,362
3	867,994	829,143
5	1089,29	796,044
6	1305,28	807,881
7	645,021	661,608
10	549,677	234,046
15	23,1118	4,34001
20	0,324895	0,0374904
25	0,00063301	4,38E-05
30	3,10E-07	2,69E-08
40	1,41E-06	3,78E-08
50	0,00029546	1,04E-07

Poniżej przedstawiam wykres przedstawiający wartość błędu dla kolejnych węzłów. Kolorem pomarańczowym są oznaczone wartości błędu dla węzłów Czebyszewa, niebieskim dla węzłów równoodległych.

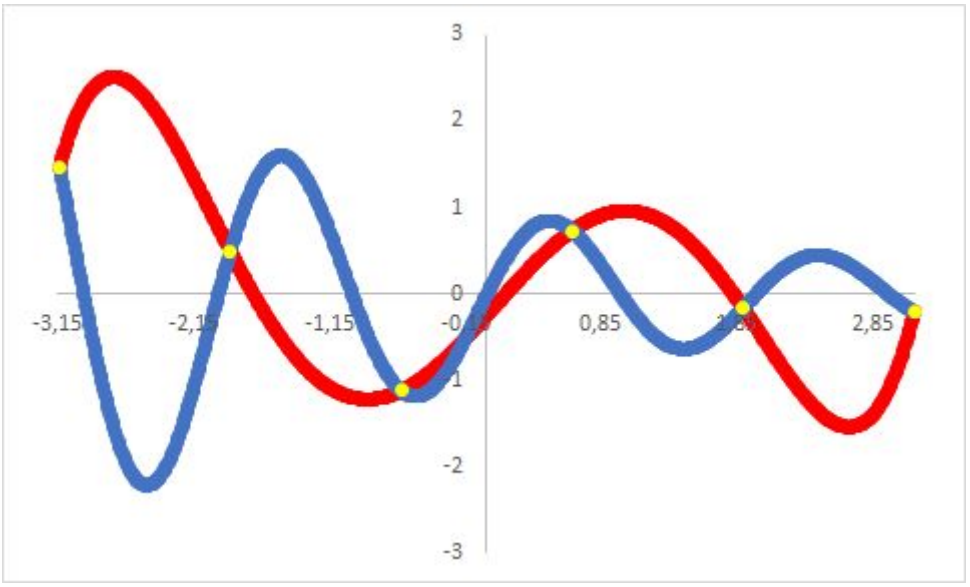


Kształt wykresu jest niemal identyczny jak w przypadku interpolacji Lagrange'a - jednak wszystkie wartości błędów są około 10 razy mniejsze.

Najmniejszy błąd wystąpił dla 34 równoodległych węzłów - wyniósł on 1,12E-08. Jego wykres wygląda następująco:



Natomiast największy błąd wystąpił - podobnie jak przy interpolacji Lagrange'a - dla 6 równoodległych węzłów. Jego wykres wygląda następująco:



Efekt Rungego występuje podobnie jak dla interpolacji Lagrange'a, tzn. jest bardziej widoczny dla węzłów równoodległych, prawie niewidoczny dla węzłów Czebyszewa.

4. Interpolacja Hermite’a

Jest to metoda interpolacji wykorzystująca pochodną funkcji:

```
double f1der(double x) {
	return (pow(M_E, (-x) / M_PI) * (10 * cos(10 * x / M_PI) - sin(10 * x / M_PI))) /
M_PI;
}
```

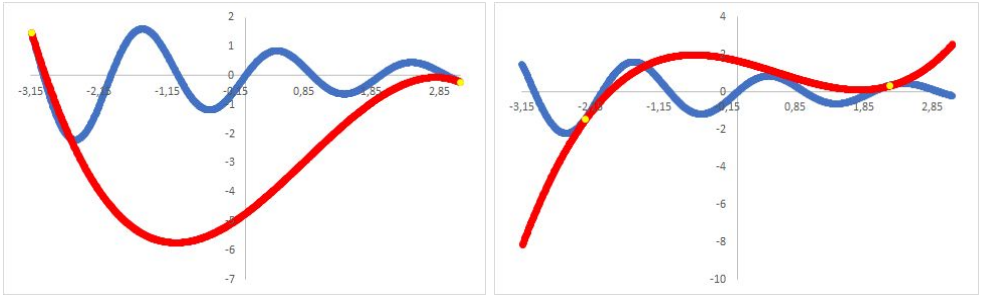
a. Implementacja

```
double hermiteInterpolation(double x, double x0, double x1, int k, bool chebyshew) {
	std::vector<double> nodes;
	if (chebyshew) {
		for (int i = 0; i < k; i++) {
			nodes.push_back(x0 + i * (x1 - x0) / ((double)k - 1));
		}
	}
	else {
		for (int i = 1; i <= k; i++) {
		nodes.push_back((M_PI * cos((2 * (double)i - 1) * M_PI / (2*(double)k))));
		}
	}
	int n = nodes.size();
	std::vector<double> z;
	for (int i = 0; i < n; i++) {
		z.push_back(nodes[i]);
		z.push_back(nodes[i]);
	}
	int n2 = n * 2;
	std::vector<std::vector<double>> matrix;
	matrix.resize(n2);
	for (int i = 0; i < n2; i++) {
		matrix[i].resize(n2, 0.0);
	}
	for (int i = 0; i < n2; i++) {
		for (int j = 0; j < i + 1; j++) {
			if (j == 0) {
				matrix[i][j] = f1(z[i]);
			}
			else if (j == 1 && i % 2 == 1) {
				matrix[i][j] = f1der(z[i]);
			}
			else {
				matrix[i][j] = matrix[i][j - 1] - matrix[i - 1][j - 1];
				matrix[i][j] = matrix[i][j] / (z[i] - z[i - j]);
			}
		}
	}
	double result = 0;
	double helper = 1;
	for (int i = 0; i < n2; i++) {
```

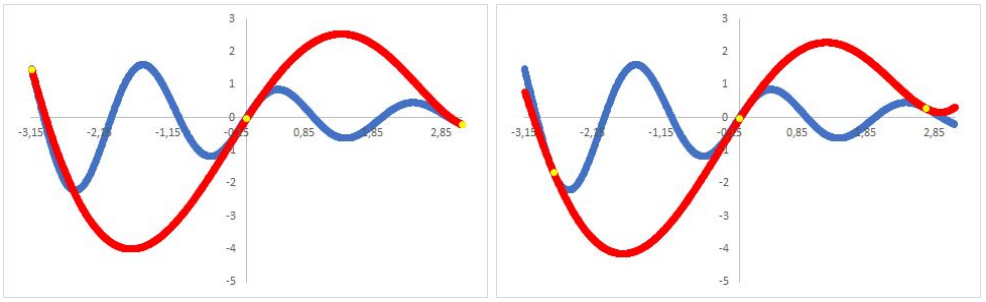
```
                result = result + matrix[i][i] * helper;
                helper = helper * (x - z[i]);
        }
        return result;
}
```

b. Wykresy uzyskane dla różnej liczby węzłów, po lewej dla węzłów równoodległych, po prawej dla węzłów Czebyszewa. Na każdym z wykresów wykres niebieski to wykres funkcji interpolowanej, czerwony to wykres funkcji interpolującej, a żółte punkty to węzły interpolacji.

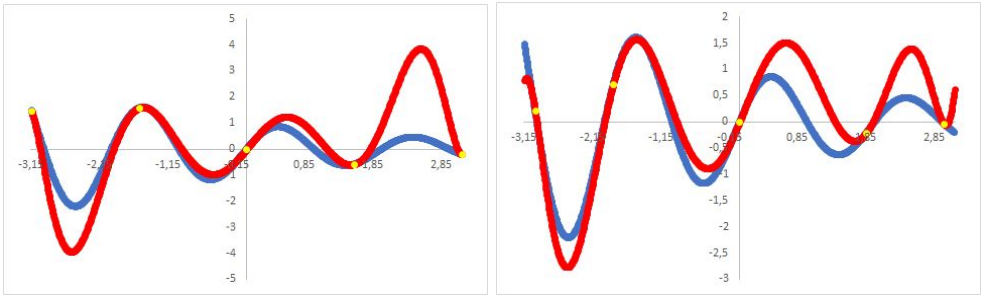
Dla 2 węzłów:



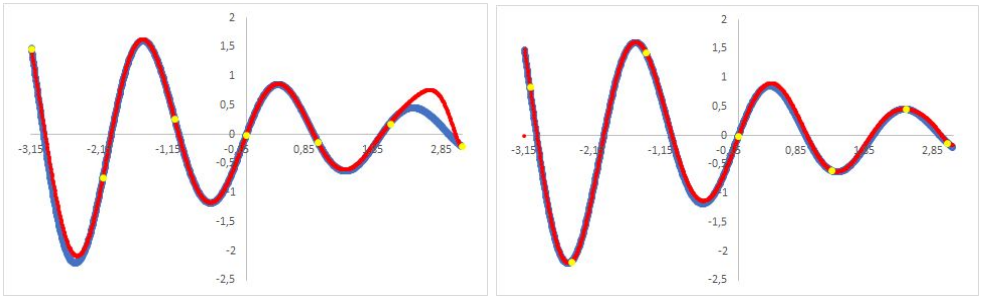
Dla 3 węzłów:



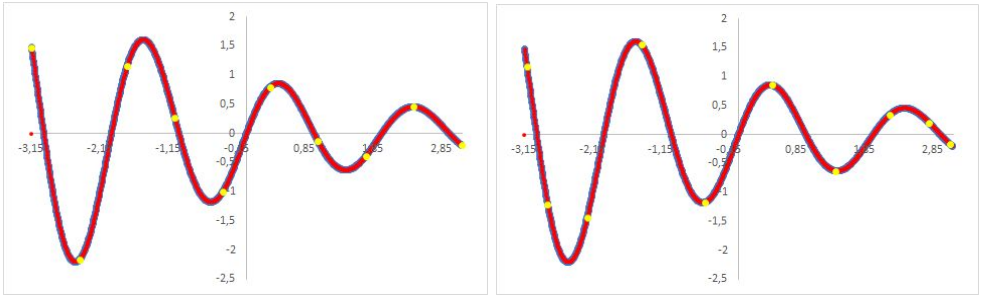
Dla 5 węzłów:



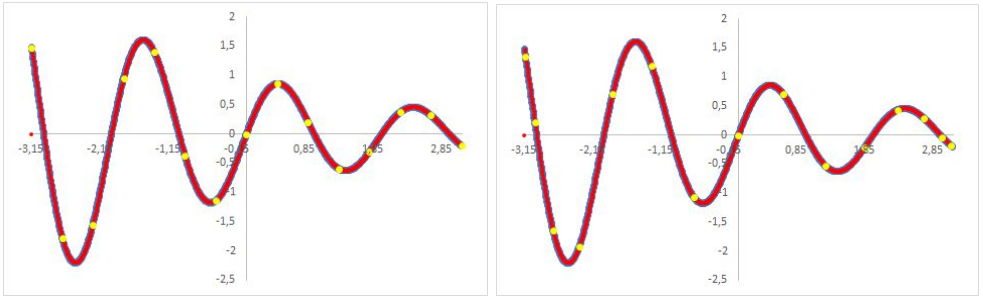
Dla 7 węzłów:



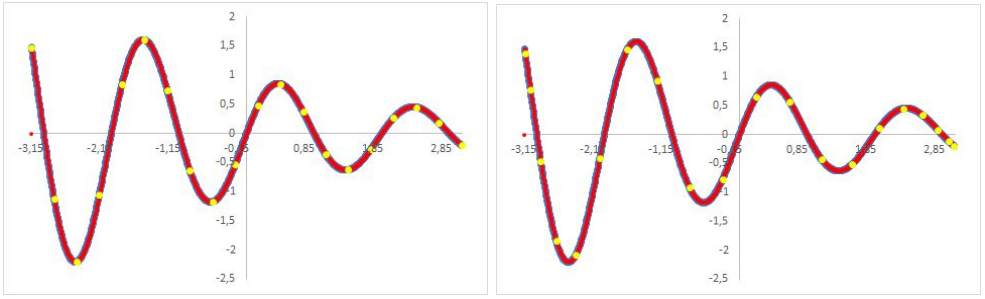
Dla 10 węzłów:



Dla 15 węzłów:



Dla 20 węzłów:

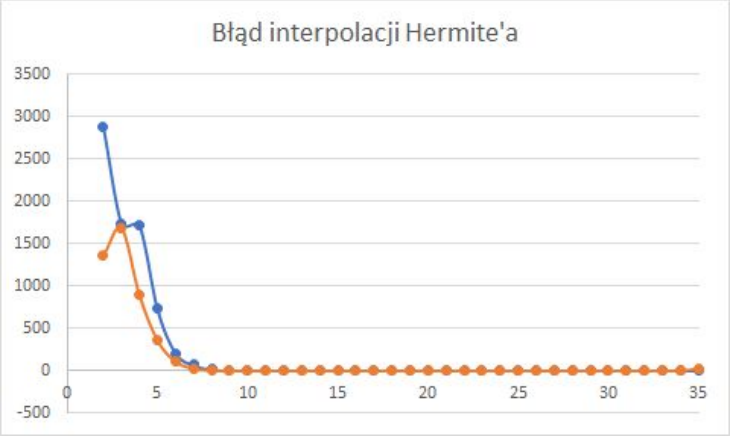


c. Błędy uzyskane przy użyciu metody Hermite'a, po lewej dla węzłów równoodległych, po prawej dla węzłów Czebyszewa.

```
double hermiteError(std::vector<double> points, int pointsNumber, bool chebyshew) {
	double error = 0.0;
	for (int i = 0; i < points.size(); i++) {
		error += abs(hermiteInterpolation(points[i], -M_PI, M_PI, pointsNumber,
chebyshew) - f1(points[i]));
	}
	return error;
}
```

Liczba węzłów	Wartość błędu dla węzłów równoodległych	Wartość błędu dla węzłów Czebyszewa
2	2870,52	1361,08
3	1735,91	1685,34
5	725,346	357,05
7	64,507	24,486
10	0,241132	0,02638
15	1,62E-07	4,52E-09
16	1,29E-08	5,78E-09
17	9,76E-09	6,29E-09
18	1,04E-08	7,15E-09
19	4,95E-08	1,16E-08
20	5,06E-08	9,12E-09
21	4,83E-07	1,18E-08
22	5,88E-07	1,47E-08
23	1,30E-06	1,45E-08
24	1,36E-06	1,77E-08
25	2,94E-05	2,38E-08
30	0,024902	0,000916
40	40819,7	2,39E+06
50	3,32E+14	6,81E+16

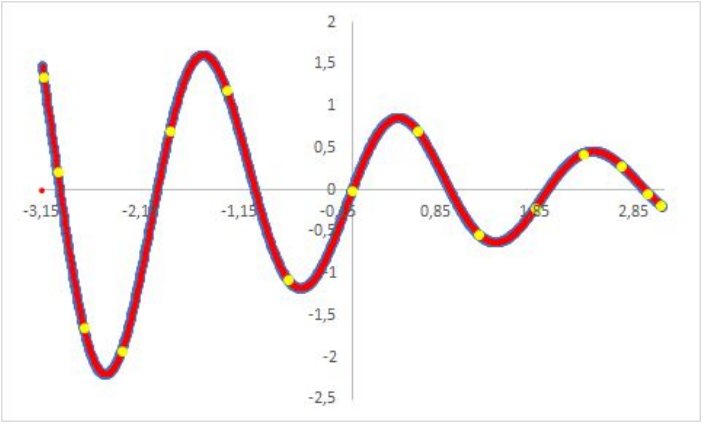
Poniżej przedstawiam wykres przedstawiający wartość błędu dla kolejnych węzłów. Kolorem pomarańczowym są oznaczone wartości błędu dla węzłów Czebyszewa, niebieskim dla węzłów równoodległych.



Przy tej metodzie efekt Runge'go nie jest tak łatwo zauważalny, jednak także występuje, przede wszystkim dla większej ilości węzłów, dla których błędy są naprawdę duże, co wynika z bardzo wysokiego stopnia wielomianu interpolującego, dla 40 węzłów, jest to wielomian 78 stopnia, a dla 50 węzłów, jest to wielomian 98 stopnia.

Można zauważyć że zastosowanie węzłów Czebyszewa w metodzie Hermit'a zwiększa dokładność dla takiej samej liczby węzłów, oraz zapobiega występowaniu efektu Runge'go, ulepszając dopasowanie funkcji interpolującej do funkcji interpolowanej.

Najlepszym dopasowaniem przy metodzie Hermit'a było dla 15 węzłów Czebyszewa. Błąd wynosił zaledwie 4,52E-09. Jego wykres wygląda następująco:



Największy błąd wystąpił dla 50 węzłów Czebyszewa, wykres jest wtedy zupełnie nieczytelny.

Biorąc pod uwagę wszystkie 3 sposoby, najlepiej dopasowaną funkcję (gdzie różnica z wartości funkcji interpolowanej i interpolującej w 1000 punktach była najmniejsza) dawała właśnie metoda Hermite'a dla 15 węzłów Czebyszewa.