

Metody numeryczne

Wykład 4



Dr inż. Michał Łanczont

Instytut Elektrotechniki i Elektrotechnologii

E419, tel. 4293, m.lanczont@pollub.pl, <http://m.lanczont.pollub.pl>

Zakres wykładu

Różniczkowanie

Całkowanie

Interpolacja

- Interpolacja wielomianowa
 - Lagrange'a
 - Newton'a
 - Interpolacja trygonometryczna
- Szybka transformata Fouriera – FFT
- Interpolujące funkcje sklejjane

Aproksymacja

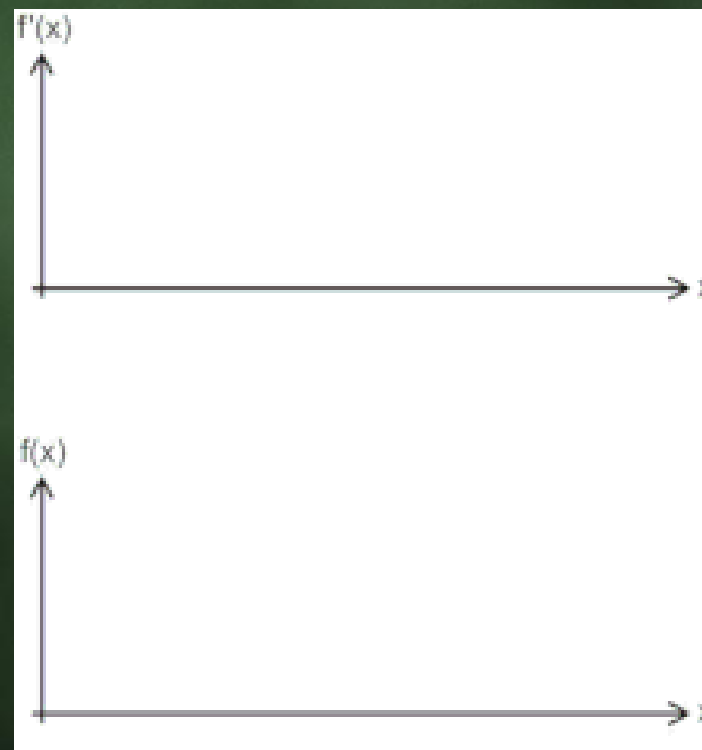
Różniczkowanie

Pochodna funkcji (różniczka) – miara szybkości funkcji, tempo zmian jej wartości względem zmian jej argumentów.

$$\lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

jeżeli $\rightarrow x = x_0 + \Delta x$

$$\lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0}$$



Różniczkowanie - zastosowania

Analiza danych eksperymentalnych – w przypadku eksperymentów, w których dane są zbierane w formie punktów dyskretnych, różniczkowanie numeryczne pozwala na obliczanie pochodnych, co może być istotne w analizie wyników eksperymentalnych.

Różniczkowanie - zastosowania

Symulacje numeryczne – w modelowaniu matematycznym i symulacjach numerycznych często używa się różniczkowania numerycznego do obliczania pochodnych i gradientów.

Jest to istotne w dziedzinach takich jak dynamika płynów, fizyka komputerowa, analiza wytrzymałości materiałów, etc.

Różniczkowanie - zastosowania

Optymalizacja – metody optymalizacji często wykorzystują pochodne w celu znajdowania minimum lub maksimum funkcji celu.

Rozwiązywanie równań różniczkowych - w numerycznym rozwiązywaniu równań różniczkowych często pojawia się konieczność obliczania pochodnych.

Różniczkowanie

1. Różnice skończone, dzielą się na dwie główne kategorie:

a) Różnice w przód obliczają przybliżoną pochodną na podstawie wartości funkcji w punktach x i $x + \Delta x$, gdzie Δx jest niewielką wartością kroku.

b) Różnice wsteczne korzystają z wartości funkcji w punktach x i $x - \Delta x$ do obliczenia przybliżonej pochodnej.

2. Różnice centralne, są bardziej dokładne niż różnice skończone. Obliczają przybliżoną pochodną, korzystając z wartości funkcji w punktach $x + \Delta x$ i $x - \Delta x$.

Różniczkowanie

3. Metoda różnic skończonych, łączy różnice skończone z użyciem wielu punktów, aby uzyskać dokładniejsze przybliżenie pochodnej.
4. Metoda różnic skończonych centralnych, rozwinięcie metody różnic skończonych, używające wielu punktów i obliczające przybliżoną pochodną za pomocą średnich różnic.
5. Metoda różnic dzielonych, opiera się na interpolacji wielomianowej, a następnie oblicza pochodną tego wielomianu.

Różniczkowanie

1. Funkcji danej za pomocą równania analitycznego
2. Funkcji opisanej tablica danych

Różniczkowanie numeryczne

Różnic skończonych

- w przód
- w tył
- centralnych

```
function y=central(t,f)
    n=size(t)(2);
    for i=2:n-1
        a=f(i+1)-f(i-1)
        b=2*(t(i+1)-t(i));
        y(i-1)=a/b;
    end
```

```
function y=backwards(t,f)
    n=size(t)(2);
    for i=2:n
        y(i-1)=(f(i)-f(i-1))/(t(i)-t(i-1));
    end
endfunction
```


Funkcja różniczkująca

Funkcja obliczająca przyrosty elementów tablicy danych

```
y=diff(f,n,"typ");
```

- **y** → wektor lub macierz wyjściowa danych zróżniczkowanych
- **f** → wektor lub macierz wejściowa, może to być także funkcja zwracająca wektor lub macierz danych
- **n** → rząd różniczkowania (stopień liczenia przyrostu), jeżeli atrybut nie zostanie podany funkcja zwraca pochodną pierwszego rzędu
- **"typ"** → atrybut tekstowy określający sposób różniczkowania danych przekazanych w postaci macierzy, może przyjmować trzy wartości:
 - **"r"** - różniczkowanie wierszowe,
 - **"c"** - różniczkowanie kolumnowe,
 - **"*"** - różniczkowania standardowe

Funkcja różniczkująca

$$y = f'(x) = \frac{\Delta f(x)}{\Delta x}$$

`y=diff (wektor) ./diff (x) ;`

`y=diff (wektor) ./dx ;`

`y=diff (wektor) /dx ;`

dx – wektor przyrostów lub wartość przyrostu

Całkowanie numeryczne

Metody numerycznego całkowania polegają na wyznaczeniu wartości przybliżonej całki oznaczonej.

W środowisku *Scilab*'a obliczenia całki oznaczonej realizuje funkcja **integrate()**. Składnia podstawowego wywołania zapisana poniżej wymaga podania czterech atrybutów.

```
W = integrate('f','z',start,koniec);
```

gdzie:

'f' → jest kodem lub odwołaniem do funkcji podcałkowej

'z' → jest zmienną funkcji podcałkowej względem której realizowane jest całkowanie

start i **koniec** → definiują przedział całkowania

Całkowanie numeryczne

1. Metoda prostokątów

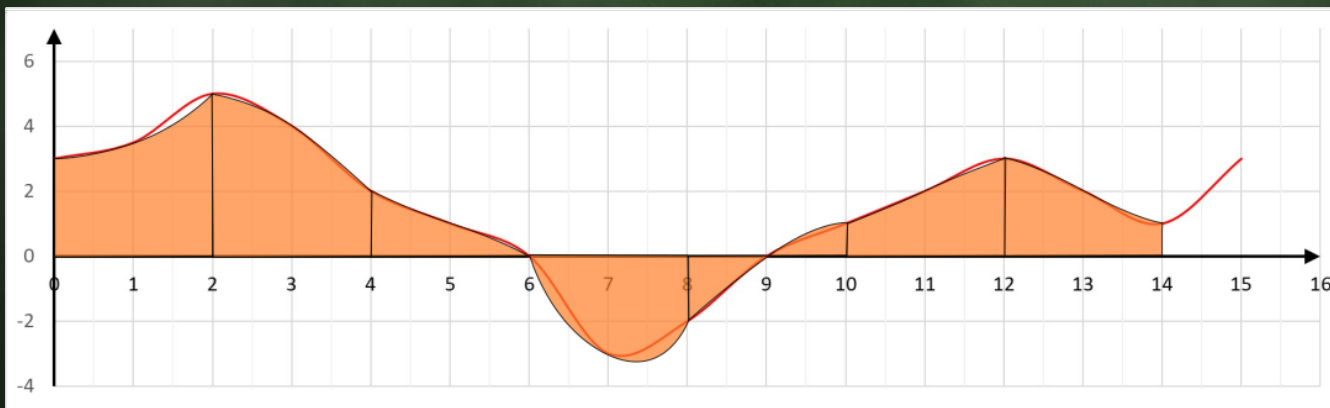
2. Metoda trapezów

3. Metoda Simpsona

a. wielomian II rzędu

b. wielomian III rzędu

- lewych
- prawych
- środkowych



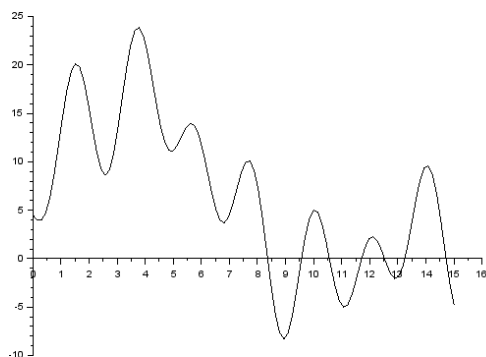
Całkowanie numeryczne

1. Metoda prostokątów
2. Metoda trapezów
3. Metoda Simpsona
 - a. wielomian II rzędu
 - b. wielomian III rzędu

$$\int_a^b f(x)dx = \sum_{n=a}^{b-\Delta x} \frac{3\Delta x}{8} (d(n) + 3d(n+1) + 3d(n+2) + d(n+3))$$

Całkowanie numeryczne

$$f(x) = 8 \sin(0.5x) - 5 \sin(3x) - 3 \cos\left(2x + \frac{\pi}{3}\right) - 0.15x^2 + 1.5x + 6$$



Lp.	Metoda	Δt	Wynik	Błąd
1	INTEGRATE()		101.50239	
2	Prostokątów lewych	1	96.307788	-5.1%
3		0.1	101.03132	-0.46%
4	Prostokątów prawych	1	105.60666	4
5		0.1	101.96121	0.46
6	Prostokątów środkowych	1	101.03132	0.25
7		0.1	101.50545	0.003
8	Trapezów	1	100.95723	-0.5%
9		0.1	101.49627	-0.006%
10	Simsona II rzędu	1	101.48416	-0.018%
11		0.1	101.50239	-0.00000%
12	Simsona III rzędu	1	101.49463	-0.008%
13		0.1	101.50239	-0.0000005%
14	Simsona II rzędu	31 kroków	101.48416	-0.018
15		151 kroków	101.50237	-0.00002%
16	Simsona III rzędu	31 kroków	101.42795	-0.07%
17		151 kroków	101.50234	-0.00004%

Interpolacja

- Wiele obserwowanych zjawisk jest opisanych nieokreślonymi (nieznanymi) funkcjami.
- Możliwe jest jednak zmierzenie wartości tych funkcji, oraz ich pochodnych przy określonych wartościach argumentów.
- Zebrane dane można w sposób graficzny zobrazować za pomocą wykresów

Interpolacja

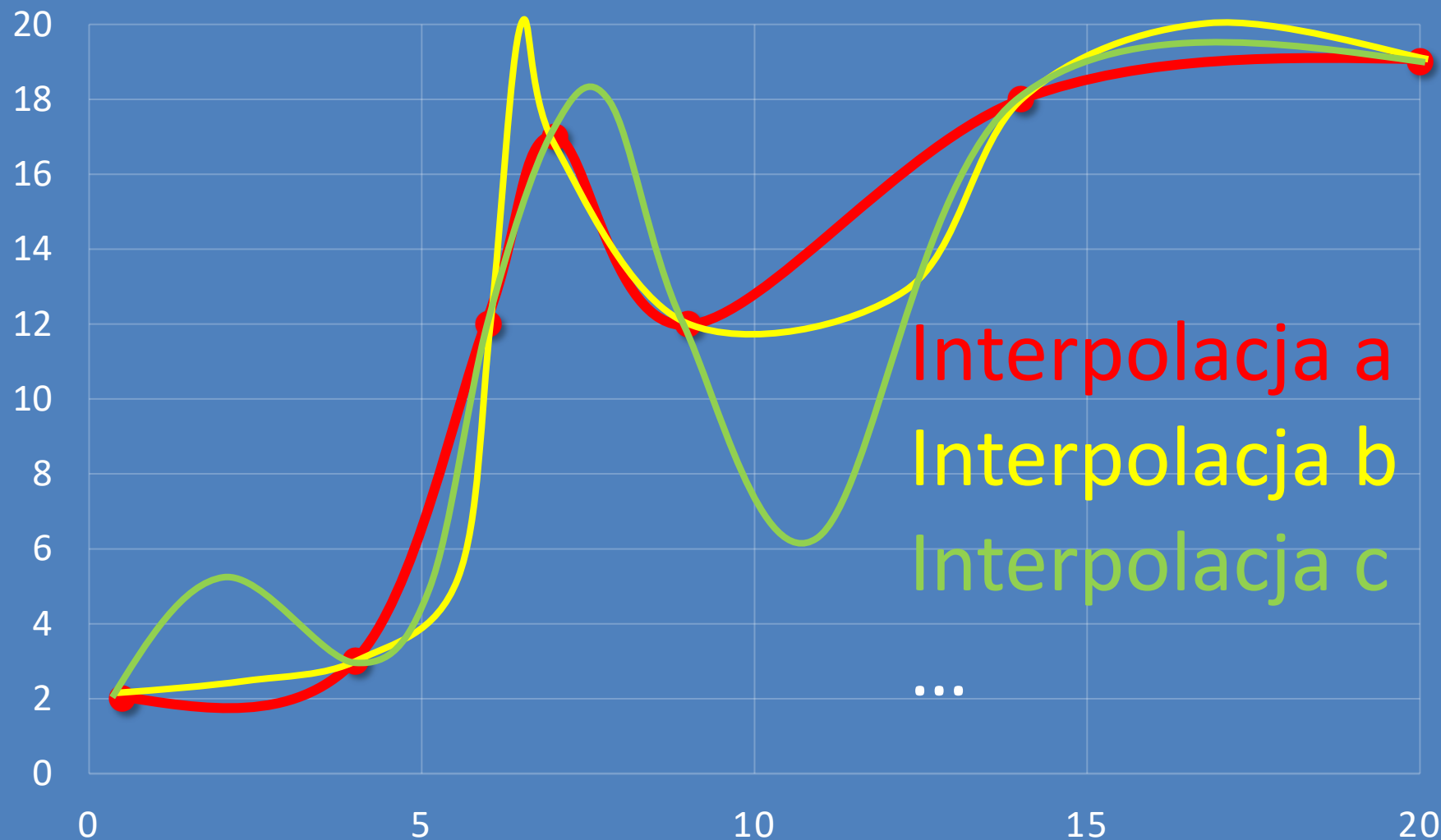
- **Interpolacją** nazywamy procedurę prowadzącą do znalezienia wartości pewnej funkcji $f(x)$ w dowolnym punkcie przedziału (x_0, x_n) na podstawie znanych wartości tej funkcji w punktach $x_0, x_1, \dots, x_n$, zwanych węzłami interpolacji.
- Postępowanie prowadzące do znalezienia wartości funkcji $f(x)$ w punkcie leżącym poza przedziałem (x_0, x_n) nazywamy **ekstrapolacją**.

Interpolacje

Interpolację lub **ekstrapolację** stosuje się gdy:

- Równanie funkcji **$f(x)$** nie jest znane, a posiadane są tylko informacje o jej wartościach w punktach przedziału
- Obliczanie wartości funkcji **$F(x)$** jest skomplikowane (czasochłonne), wygodniej jest interpolować prostszą obliczeniowo funkcją **$f(x)$** zwracającą w zadanych punktach takie same wartości jak funkcja **$F(x)$**

Interpolacja



Interpolacja

Funkcji **interpolującej** i **ekstrapolującej** poszukuje się zazwyczaj w pewnej określonej postaci:

- **Wielomianowa**
 - **Algebraiczna**
 - **Trygonometryczna**
- **Funkcjami sklejanymi**

Interpolacja wielomianowa

Interpolacja wielomianowa opiera się na wielomianie stopnia co najmniej n , gdzie n jest liczbą węzłów (punktów) interpolacji.

$$W(x_i) = y_i$$

gdzie:

$$W(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n$$

Interpolacja wielomianowa

Dla **n** węzłów interpolacyjnych można więc zapisać równanie macierzowe opisujące warunki jakie musi spełnić wielomian interpolacyjny: **$Xa=y$**

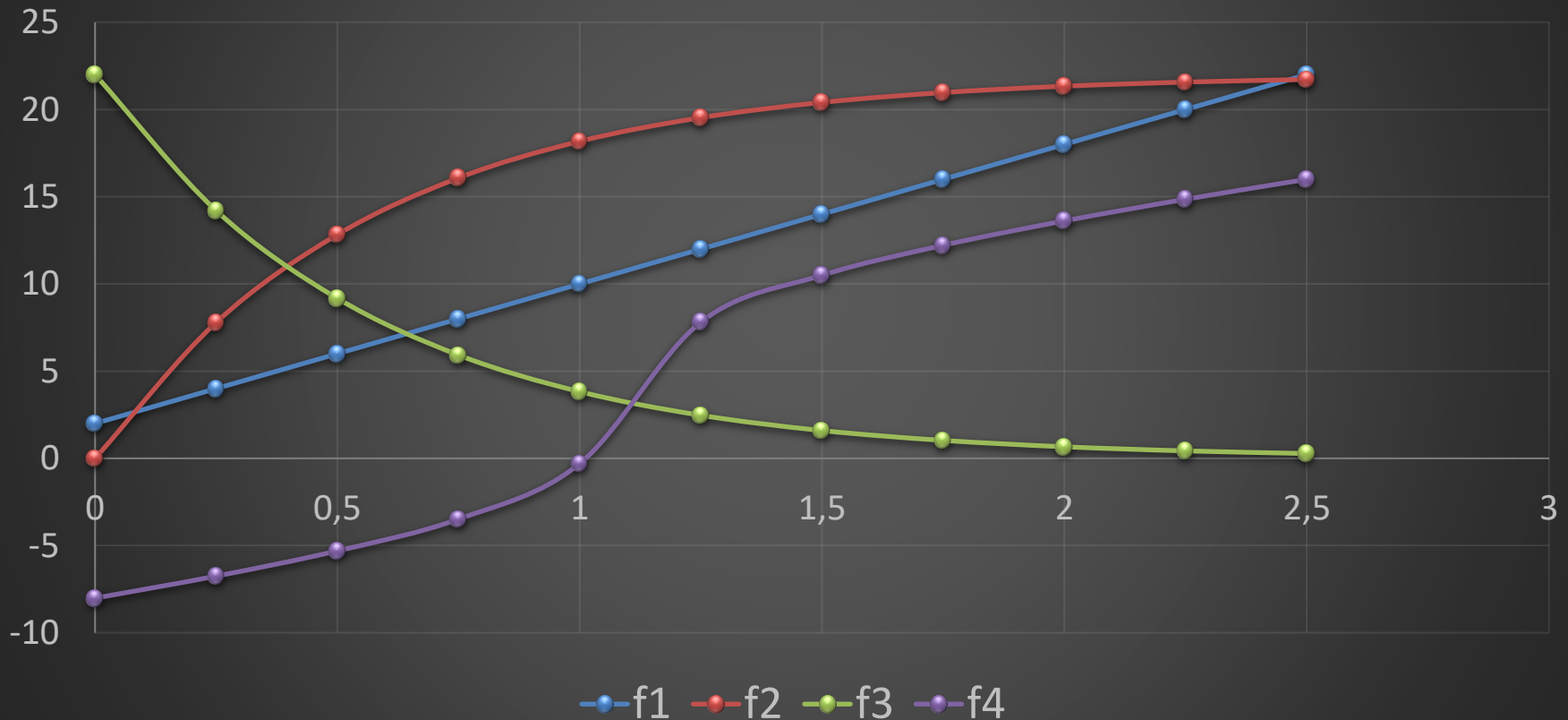
$$\begin{vmatrix} x_0^0 & x_0^1 & x_0^2 & \cdots & x_0^n \\ x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^n \\ x_2^0 & x_2^1 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n^0 & x_n^1 & x_n^2 & \cdots & x_n^n \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{vmatrix} = \begin{vmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{vmatrix}$$

Interpolacja wielomianowa

- Poszukiwane wartości współczynników można policzyć z zależności $a = X^{-1}y$
- Wyznaczenie współczynników można także zrealizować rozwiązując układ równań liniowych, funkcja `linsolve(X, -y)`
- Warunkiem uzyskania jednego rozwiązania jest $\det X \neq 0$

Interpolacja – przykład 1

Przebieg danych testowych



Interpolacja - dane

x	f1	f2	f3	f4
0	2	0	22	-8,02
0,25	4	7,796	14,204	-6,75
0,5	6	12,829	9,171	-5,30
0,75	8	16,079	5,921	-3,48
1	10	18,177	3,823	-0,29
1,25	12	19,532	2,468	7,81
1,5	14	20,406	1,594	10,50
1,75	16	20,971	1,029	12,21
2	18	21,336	0,664	13,61
2,25	20	21,571	0,429	14,85
2,5	22	21,723	0,277	16,00

Interpolacja – przykład 1

```
x=[0,0.25,0.5,0.75,1,1.25,1.5,1.75,2,2.25,2.5];
f1=[2,4,6,8,10,12,14,16,18,20,22];
f2=[0,7.796,12.829,16.079,18.177,19.532,20.406,20.971,21.336,21.571,21.723];
f3=[22,14.204,9.171,5.921,3.823,2.468,1.594,1.029,0.664,0.429,0.277];
f4=[-8.02,-6.75,-5.3,-3.48,-0.29,7.81,10.5,12.21,13.61,14.85,16];
```

```
m=500;
```

```
wielomian(x,f1,1,m)
```

```
plot2d(x,f1,-1)
```

```
wielomian(x,f2,2,m)
```

```
plot2d(x,f2,-2)
```

```
wielomian(x,f3,3,m)
```

```
plot2d(x,f3,-3)
```

```
wielomian(x,f4,4,m)
```

```
function wielomian(x,f,c,m)
```

```
.... A=tablica(x);
```

```
.... y=linsolve(A,-f');
```

```
.... n=size(x,'c');
```

```
.... t=linspace(x(1),x(n),m);
```

```
.... fun=[0];
```

```
.... for i=1:m
```

```
.... .. fun(i)=0
```

```
.... .. for j=1:n
```

```
.... .. .. fun(i)=fun(i)+y(j)*t(i)^(j-1)
```

```
.... .. end
```

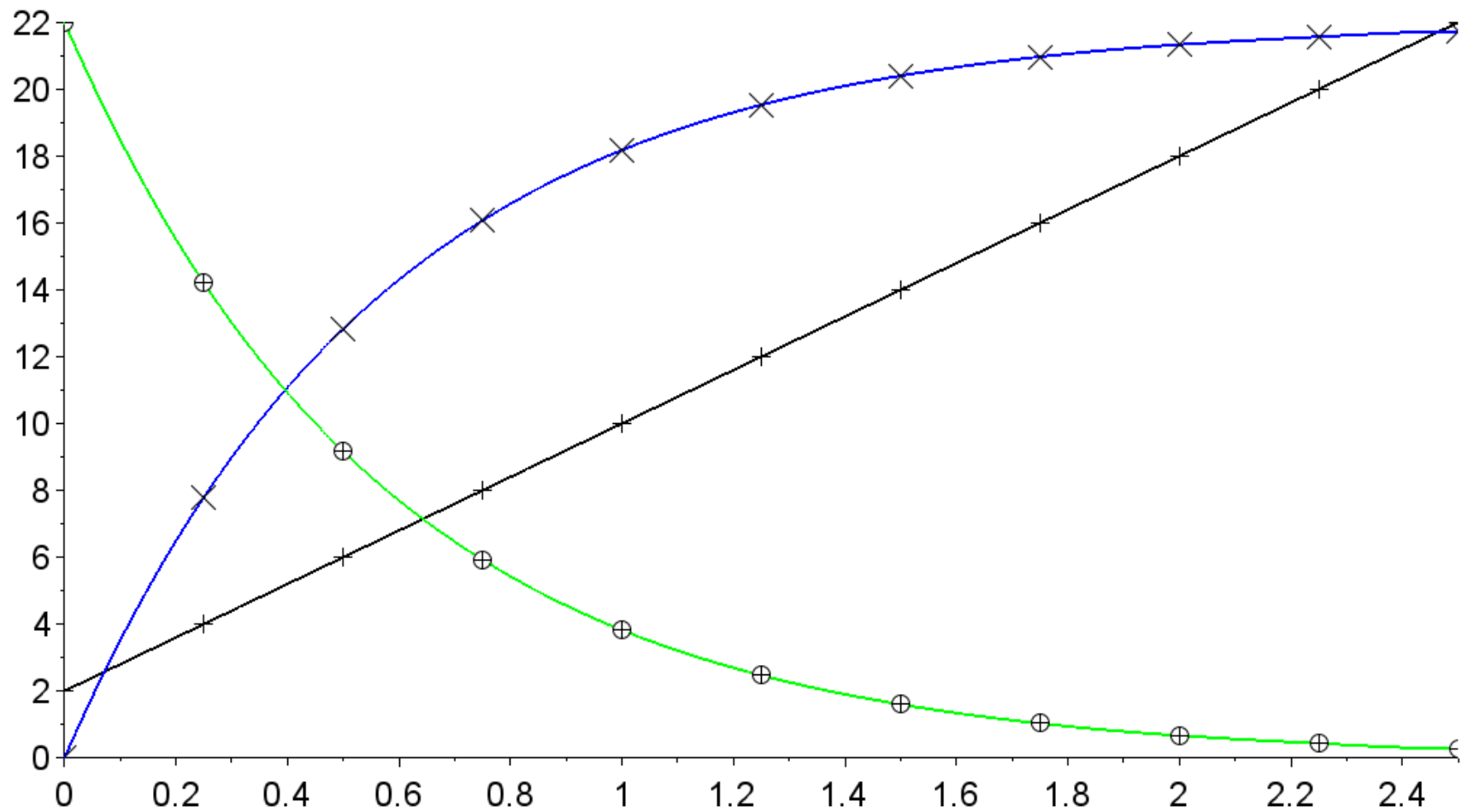
```
.... end
```

```
.... plot2d(t,fun,c)
```

```
endfunction
```

$$\begin{vmatrix} x_0^0 & x_0^1 & x_0^2 & \dots \\ x_1^0 & x_1^1 & x_1^2 & \dots \\ x_2^1 & x_2^1 & x_2^2 & \dots \\ \vdots & \vdots & \vdots & \ddots \\ x_n^1 & x_n^1 & x_n^2 & \dots \end{vmatrix}$$

Interpolacja – przykład 1

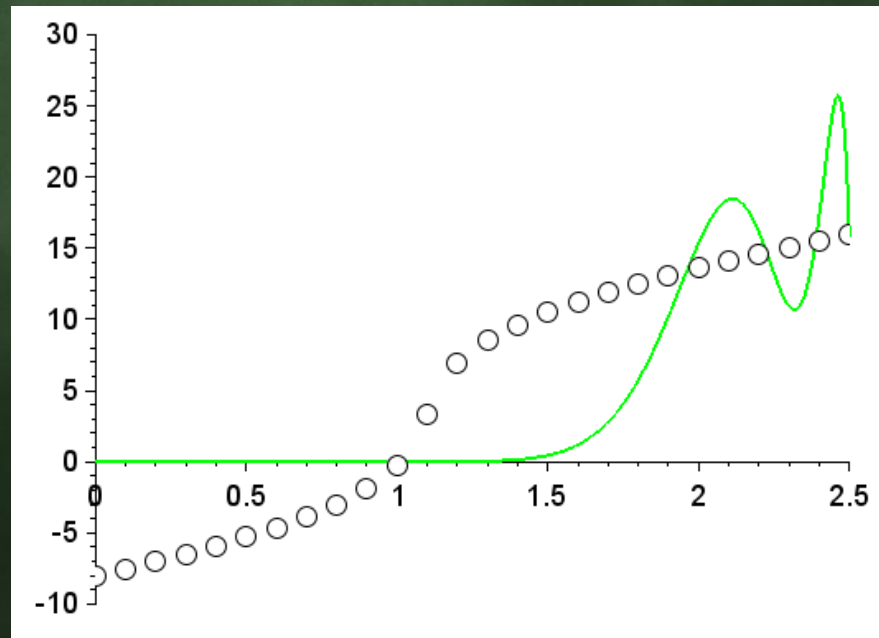


Interpolacja wielomianowa

Wpływ liczby węzłów interpolacyjnych

Duża liczba węzłów pogarsza
dokładność odwzorowania przebiegu
przez wielomian interpolacyjny

- 7
- 11
- 26



Interpolacja wielomianowa

```
function wielomian(x, f, c, m, k)
    A=tablica(x, k);
    y=pinv(A) * (f')
    n=size(x, 'c');
    t=linspace(x(1), x(n), m);
    fun=[0];
    for i=1:m
        fun(i)=0
        for j=1:k
            fun(i)=fun(i)+y(j)*t(i)^(j-1)
        end
    end
    plot2d(t, fun, c)
endfunction
```

erpolacji

onia

owanie

cji

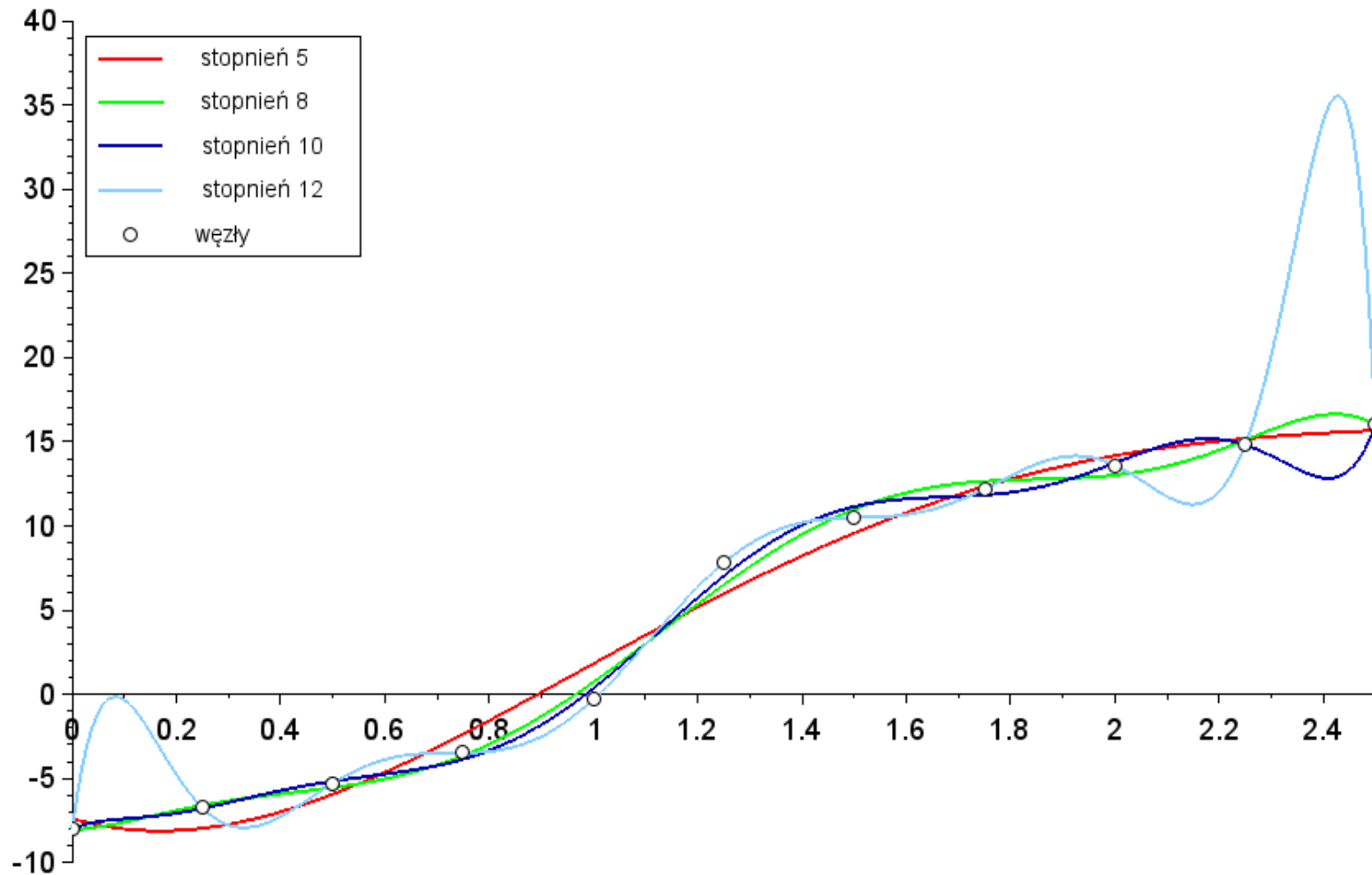
yjnego

owoduje

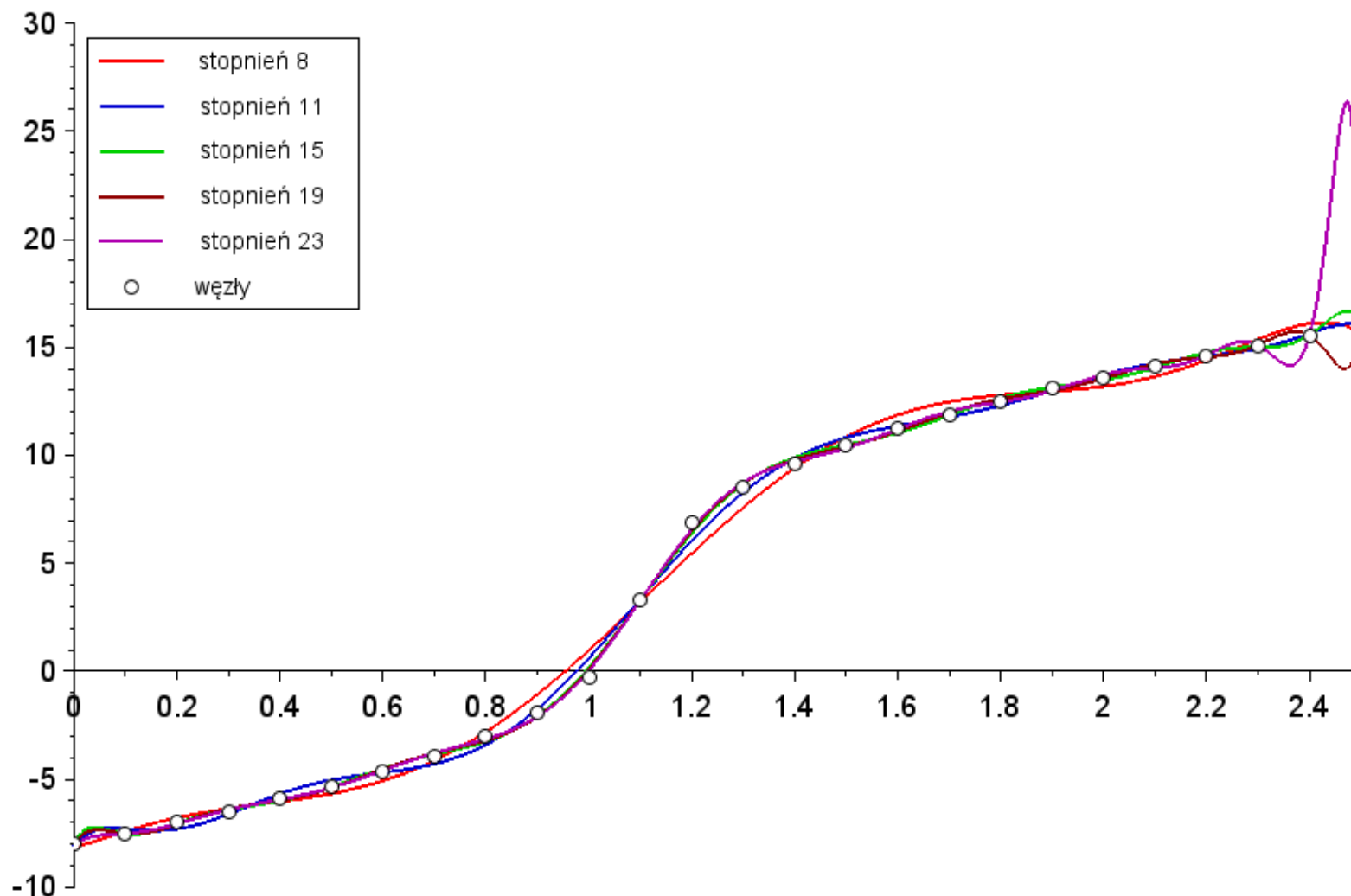
erz

aczenia

Interpolacja – przykład 1'



Interpolacja – przykład 1''



Interpolacja Lagrang'a

Interpolacja wielomianem Lagrang'a jest odmianą interpolacji wielomianowej w której nieznaną funkcję interpoluje się do wielomianu iteracyjnym w postaci:

$$L(x) = \sum_{i=1}^n \left(y_i \cdot \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \right)$$

Interpolacja Lagrang'a

Rozwijając wielomian $L(x)$ otrzymujemy:

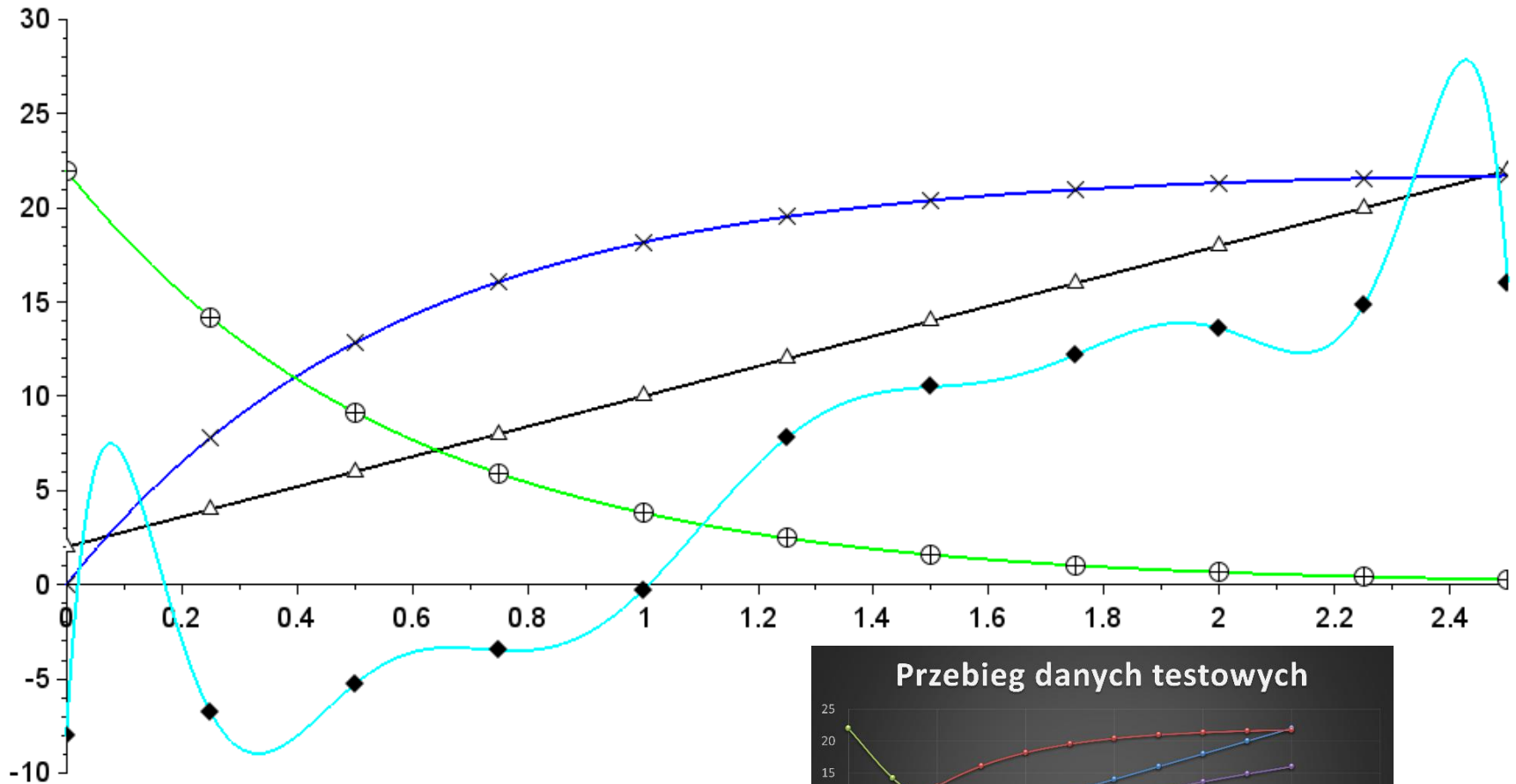
$$\begin{aligned} L(x) = & y_0 \frac{(x - x_1)(x - x_2) \cdots (x - x_n)}{(x_0 - x_1)(x_0 - x_2) \cdots (x_0 - x_n)} \\ & + y_1 \frac{(x - x_0)(x - x_2) \cdots (x - x_n)}{(x_1 - x_0)(x_1 - x_2) \cdots (x_1 - x_n)} + \cdots + \\ & + y_n \frac{(x - x_0)(x - x_1) \cdots (x - x_{n-1})}{(x_n - x_0)(x_n - x_1) \cdots (x_n - x_{n-1})} \end{aligned}$$

Lagrang– przykład 2

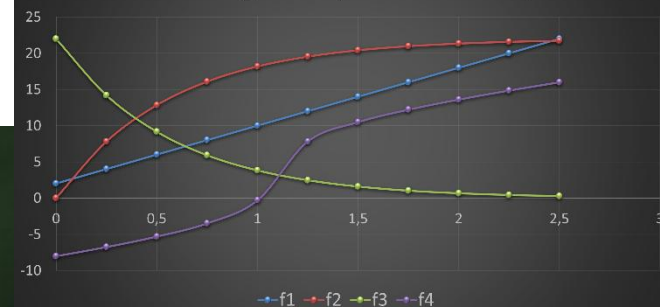
```
function lagrange(x,f,c,m)
... n=size(x,'c');
... t=linspace(x(1),x(n),m);
... fun=[0];
... for i=1:m
...     fun(i)=0
...     for j=1:n
...         suma=1
...         for k=1:n
...             if(k<>j)
...                 suma=suma*(t(i)-x(k))/(x(j)-x(k))
...             end
...         end
...     end
...     fun(i)=fun(i)+f(j)*suma;
... end
... plot2d(t,fun,c)
endfunction
```

$$L(x) = \sum_{i=1}^n \left(y_i \cdot \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \right)$$

Lagrang– przykład 2



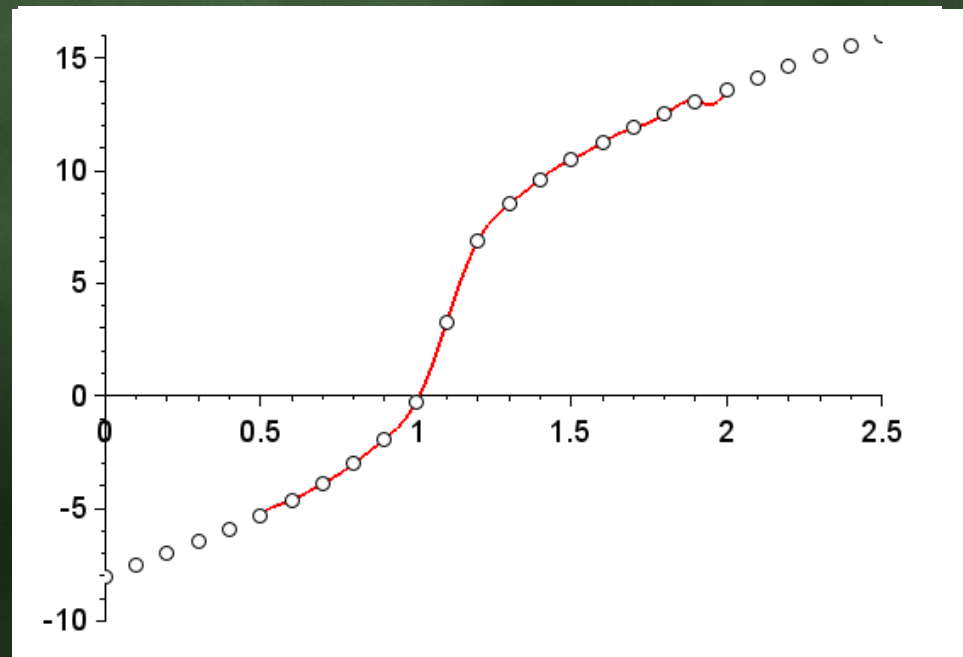
Przebieg danych testowych



Interpolacja Lagrang'a

Analogicznie jak w przypadku interpolacji wielomianowej liczba węzłów interpolacji niekoniecznie wpływa pozytywnie na dokładność interpolacji

- 7
- 11
- 26
- $\langle 0.5, 2 \rangle$



Interpolacja Newtona

15:20

Ilorazów różnicowych

Metoda Newtona opiera się na wykorzystaniu wyrażeń zwanych ilorazami różnicowymi.

Wyrazy te określa się dla $n-1$ rzędów gdzie n jest liczbą punktów pomiarowych (węzłów interpolacyjnych).

Na podstawie wyznaczonych ilorazów różnicowych tworzy się zazwyczaj tablicę.

Interpolacja Newtona

1. Pierwszego rzędu

$$[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}$$

$$[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1}$$

...

$$[x_{n-1}, x_n] = \frac{y_n - y_{n-1}}{x_n - x_{n-1}}$$

2. Drugiego rzędu

$$[x_0, x_1, x_2] = \frac{[x_1, x_2] - [x_0, x_1]}{x_2 - x_0}$$

$$[x_1, x_2, x_3] = \frac{[x_2, x_3] - [x_1, x_2]}{x_3 - x_1}$$

...

$$[x_{n-2}, x_{n-1}, x_n] = \frac{[x_{n-1}, x_n] - [x_{n-2}, x_{n-1}]}{x_n - x_{n-2}}$$

3. k-tego rzędu

$$[x_i, x_{i+1}, \dots, x_{i+k}] = \frac{[x_{i+1}, x_{i+2}, \dots, x_{i+k}] - [x_i, x_{i+1}, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Interpolacja Newtona

Wzór ogólny iteracyjny na iloraz różnicowy można zapisać w postaci:

```
for i=1 to n-1  
  for j=2 to n  
    if j<i+1 then  
      continue
```

$$R_{j,i+1} = \frac{R_{j,i} - R_{j-1,i}}{x_j - x_{j-i}}$$

Interpolacja Newtona

Wzór interpolacyjny Newtona z różnicami ilorazowymi przyjmuje postać:

$$N_n(x) = y_0 + \sum_{i=0}^n \left(R_{i,i} \prod_{j=0}^{i-1} (x - x_j) \right)$$

Interpolacja Newtona

Na podstawie zapisanego równania iteracyjnego wielomianu można zapisać algorytm komputerowy dla interpolacji Newtona.

```
for i=1 to n
```

```
    for j=0 to i-1
```

```
        iloczyn=iloczyn * (x-xj)
```

```
        suma=suma+Ri,i * iloczyn
```

```
fun=y0+suma
```


Interpolacja Newtona

```
x=[0,0.25,0.5,0.75,1,1.25,1.5,1.75,2,2.25,2.5];  
f1=[2,4,6,8,10,12,14,16,18,20,22];  
f2=[0,7.796,12.829,16.079,18.177,19.532,20.406,20.971,21.336,21.571,21.723];  
f3=[22,14.204,9.171,5.921,3.823,2.468,1.594,1.029,0.664,0.429,0.277];  
f4=[-8.02,-6.75,-5.3,-3.48,-0.29,7.81,10.5,12.21,13.61,14.85,16];  
m=500;  
newton(x,f1,1,m)  
plot2d(x,f1,-1)  
newton(x,f2,2,m)  
plot2d(x,f2,-2)  
newton(x,f3,3,m)  
plot2d(x,f3,-3)  
newton(x,f4,4,m)  
plot2d(x,f4,-4)  
osie=gca();  
osie.x_location="origin";  
osie.y_location="origin";
```

Interpolacja Newtona

```
function R=tablica(x,f)
```

```
    n=size(x,'c');
```

```
    R=[0];
```

```
    for i=1:n
```

```
        R(i,1)=f(i);
```

```
    end
```

```
    for i=1:n-1
```

```
        for j=2:n
```

```
            if i+1>j
```

```
                continue;
```

```
            end
```

```
            R(j,i+1)=(R(j,i)-R(j-1,i))/(x(j)-x(j-i));
```

```
        end
```

```
    end
```

```
endfunction
```

```
for i=1 to n-1
```

```
    for j=2 to n
```

```
        if j<i+1 then
```

```
            continue
```

$$R_{j,i+1} = \frac{R_{j,i} - R_{j-1,i}}{x_j - x_{j-i}}$$

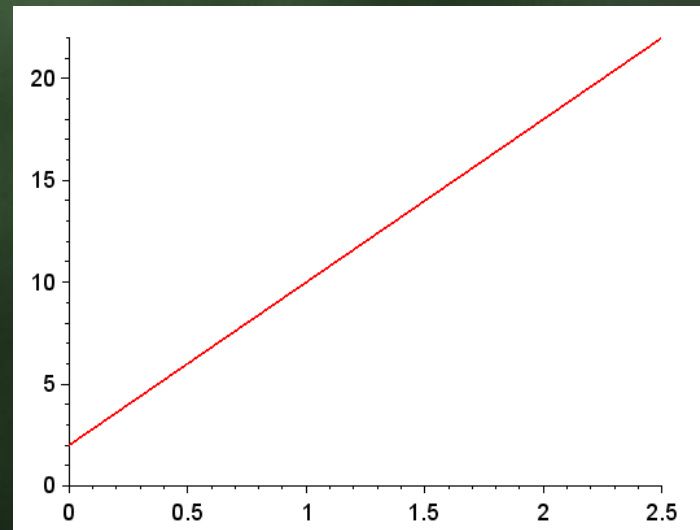
Interpolacja Newtona

```
function newton(x,f,c,m)
... n=size(x,'c');
... t=linspace(x(1),x(n),m);
... R=tablica(x,f)
... fun=[0];
... for i=1:m
...     suma = 0;
...     for j=2:n
...         iloczyn = 1;
...         for k=1:j-1
...             iloczyn = iloczyn * (t(i)-x(k));
...         end
...         suma = suma + R(j,j)*iloczyn;
...     end
...     fun(i)=R(1,1)+suma;
... end
... plot2d(t,fun,c);
endfunction
```

$$N_n(x) = y_0 + \sum_{i=0}^n \left(I_n \prod_{j=0}^{i-1} (x - x_j) \right)$$

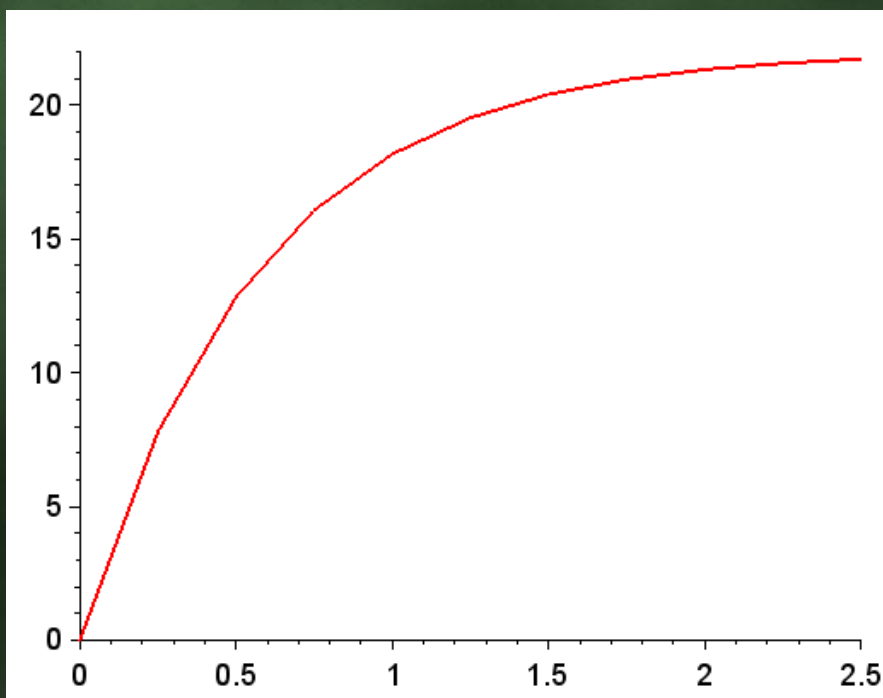
Newtona – przykład

2.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
4.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
6.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
8.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
10.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
12.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
14.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
16.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
18.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
20.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.
22.	8.	0.	0.	0.	0.	0.	0.	0.	0.	0.



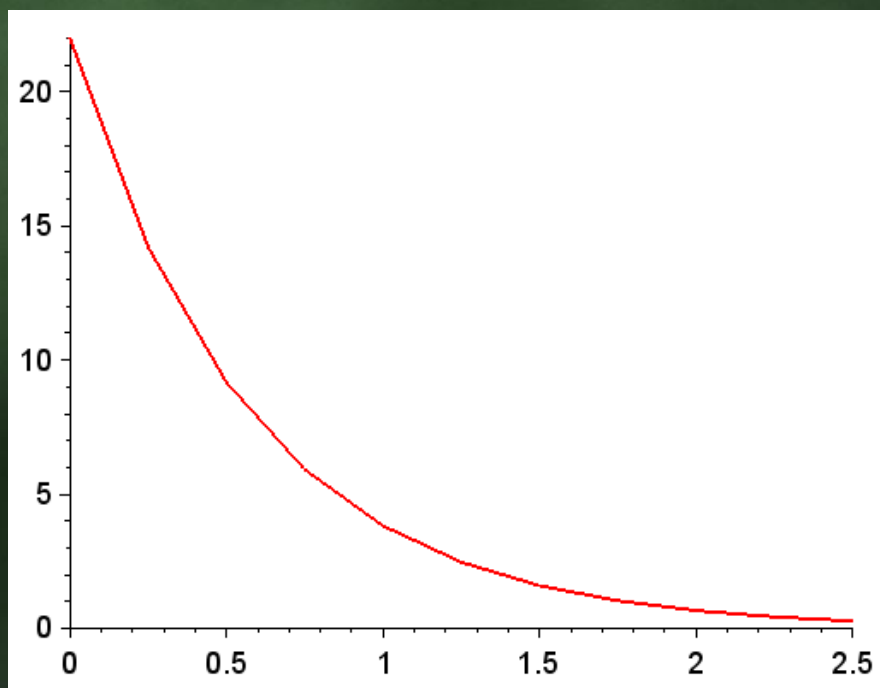
Newtona – przykład

0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
7.796	31.184	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
12.829	20.132	-22.104	0.	0.	0.	0.	0.	0.	0.	0.	0.
16.079	13.	-14.264	10.453333	0.	0.	0.	0.	0.	0.	0.	0.
18.177	8.392	-9.216	6.7306667	-3.7226667	0.	0.	0.	0.	0.	0.	0.
19.532	5.42	-5.944	4.3626667	-2.368	1.0837333	0.	0.	0.	0.	0.	0.
20.406	3.496	-3.848	2.7946667	-1.568	0.64	-0.2958222	0.	0.	0.	0.	0.
20.971	2.26	-2.472	1.8346667	-0.96	0.4864	-0.1024	0.110527	0.	0.	0.	0.
21.336	1.46	-1.6	1.1626667	-0.672	0.2304	-0.1706667	-0.0390095	-0.0747683	0.	0.	0.
21.571	0.94	-1.04	0.7466667	-0.416	0.2048	-0.0170667	0.0877714	0.0633905	0.0614039	0.	0.
21.723	0.608	-0.664	0.5013333	-0.2453333	0.1365333	-0.0455111	-0.016254	-0.0520127	-0.0512903	-0.0450777	0.



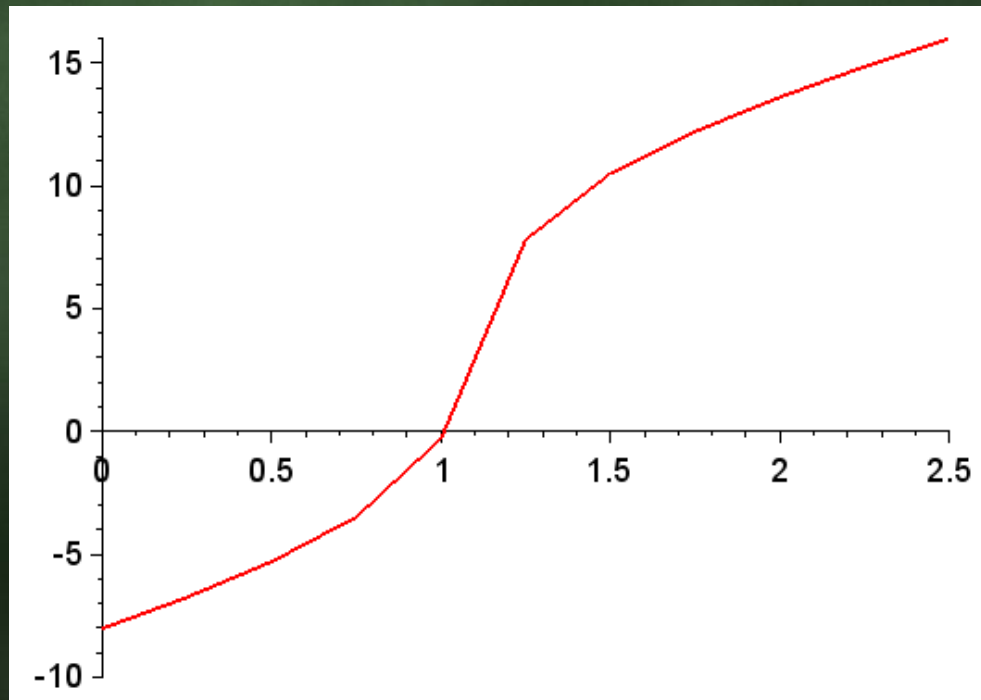
Newtona – przykład

22.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
14.204	-31.184	0.	0.	0.	0.	0.	0.	0.	0.	0.
9.171	-20.132	22.104	0.	0.	0.	0.	0.	0.	0.	0.
5.921	-13.	14.264	-10.453333	0.	0.	0.	0.	0.	0.	0.
3.823	-8.392	9.216	-6.7306667	3.7226667	0.	0.	0.	0.	0.	0.
2.468	-5.42	5.944	-4.3626667	2.368	-1.0837333	0.	0.	0.	0.	0.
1.594	-3.496	3.848	-2.7946667	1.568	-0.64	0.2958222	0.	0.	0.	0.
1.029	-2.26	2.472	-1.8346667	0.96	-0.4864	0.1024	-0.110527	0.	0.	0.
0.664	-1.46	1.6	-1.1626667	0.672	-0.2304	0.1706667	0.0390095	0.0747683	0.	0.
0.429	-0.94	1.04	-0.7466667	0.416	-0.2048	0.0170667	-0.0877714	-0.0633905	-0.0614039	0.
0.277	-0.608	0.664	-0.5013333	0.2453333	-0.1365333	0.0455111	0.016254	0.0520127	0.0512903	0.0450777

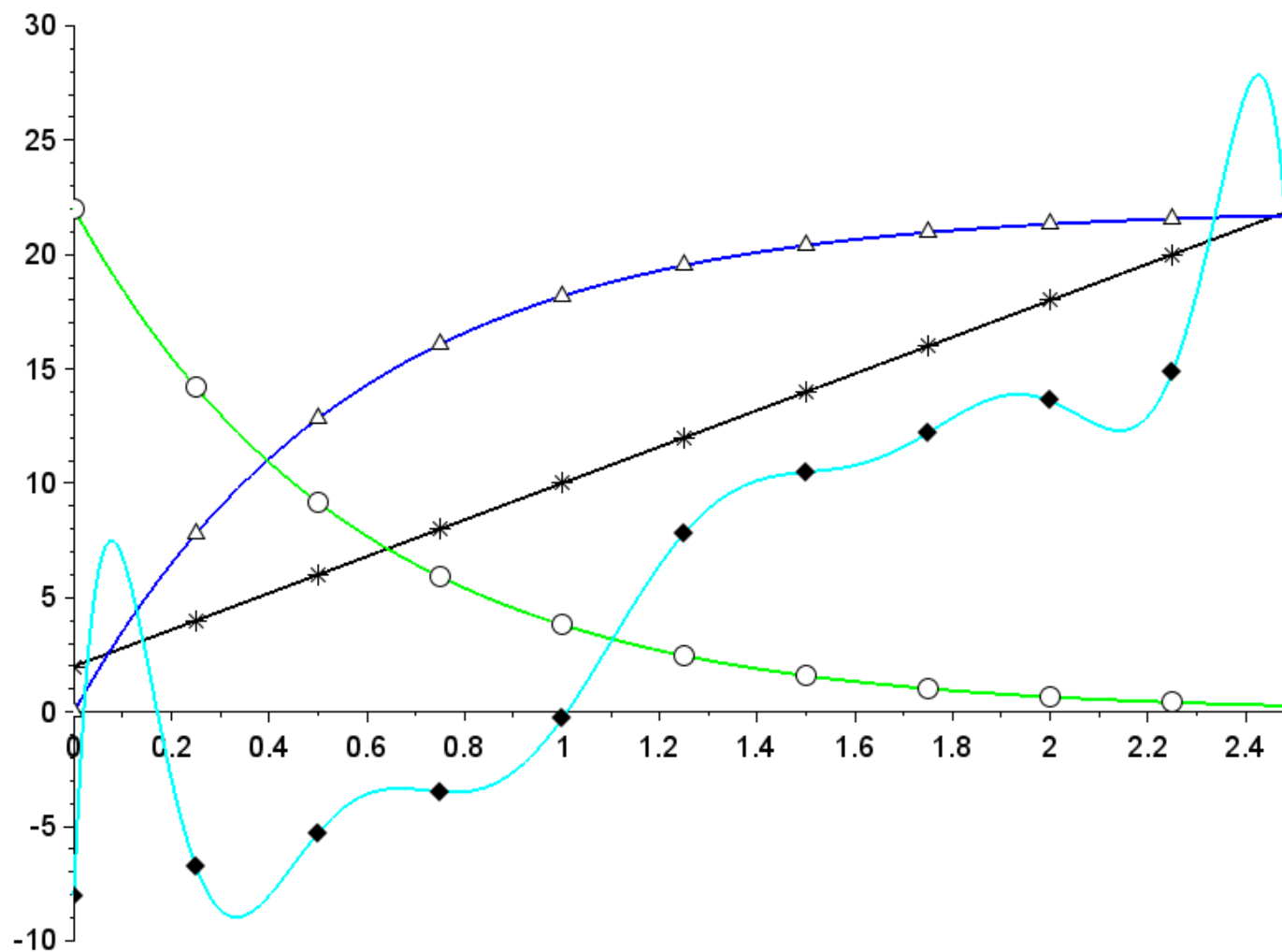


Newtona – przykład

-8.02	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
-6.75	5.08	0.	0.	0.	0.	0.	0.	0.	0.	0.
-5.3	5.8	1.44	0.	0.	0.	0.	0.	0.	0.	0.
-3.48	7.28	2.96	2.0266667	0.	0.	0.	0.	0.	0.	0.
-0.29	12.76	10.96	10.666667	8.64	0.	0.	0.	0.	0.	0.
7.81	32.4	39.28	37.76	27.093333	14.762667	0.	0.	0.	0.	0.
10.5	10.76	-43.28	-110.08	-147.84	-139.94667	-103.13956	0.	0.	0.	0.
12.21	6.84	-7.84	47.253333	157.33333	244.13867	256.05689	205.25511	0.	0.	0.
13.61	5.6	-2.48	7.1466667	-40.106667	-157.952	-268.06044	-299.49562	-252.37537	0.	0.
14.85	4.96	-1.28	1.6	-5.5466667	27.648	123.73333	223.88216	261.68889	228.473	0.
16.	4.6	-0.72	0.7466667	-0.8533333	3.7546667	-15.928889	-79.806984	-151.84457	-183.79265	-164.90626



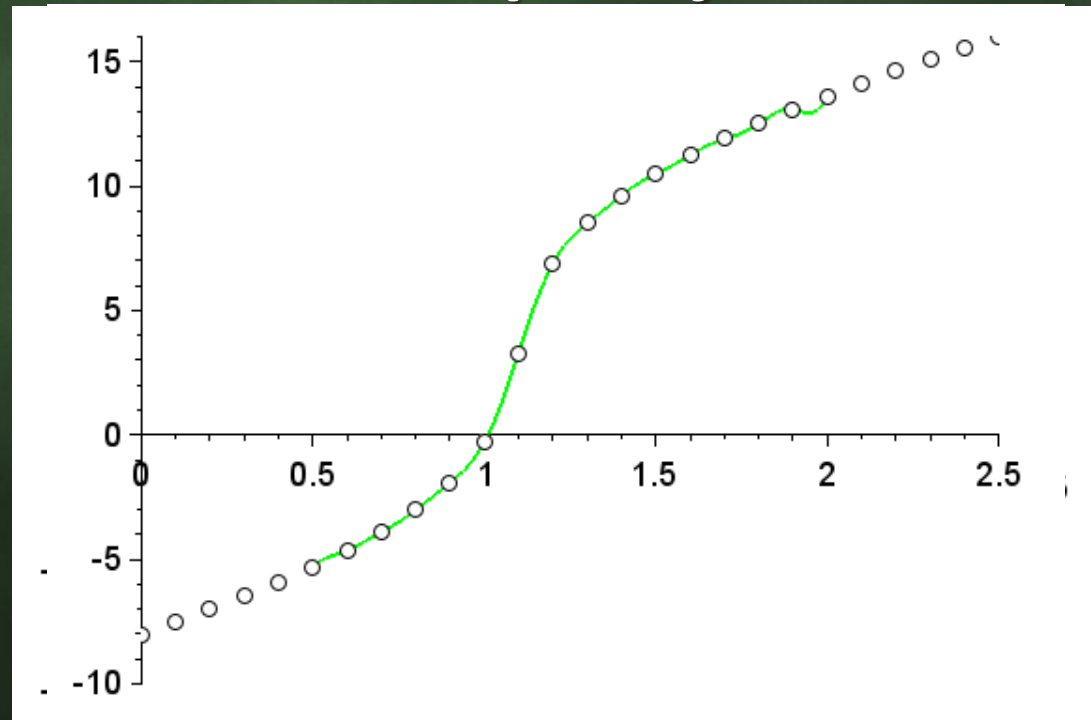
Newtona – przykład



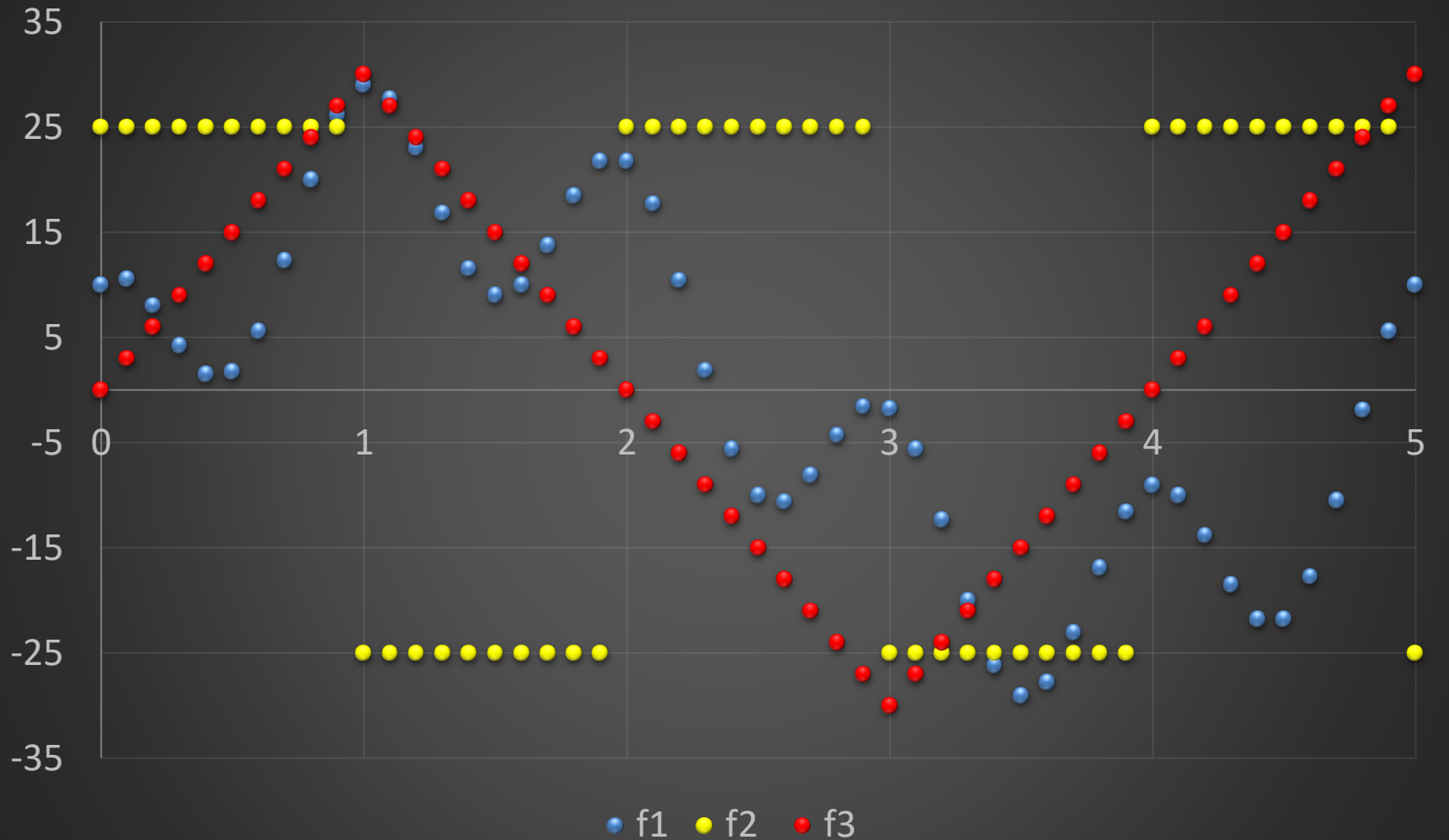
Interpolacja Newtona

Analogicznie jak w przypadku omówionych wcześniej metod interpolacji zwiększenie liczby węzłów interpolacyjnych może niekorzystnie wpłynąć na dokładność interpolacji

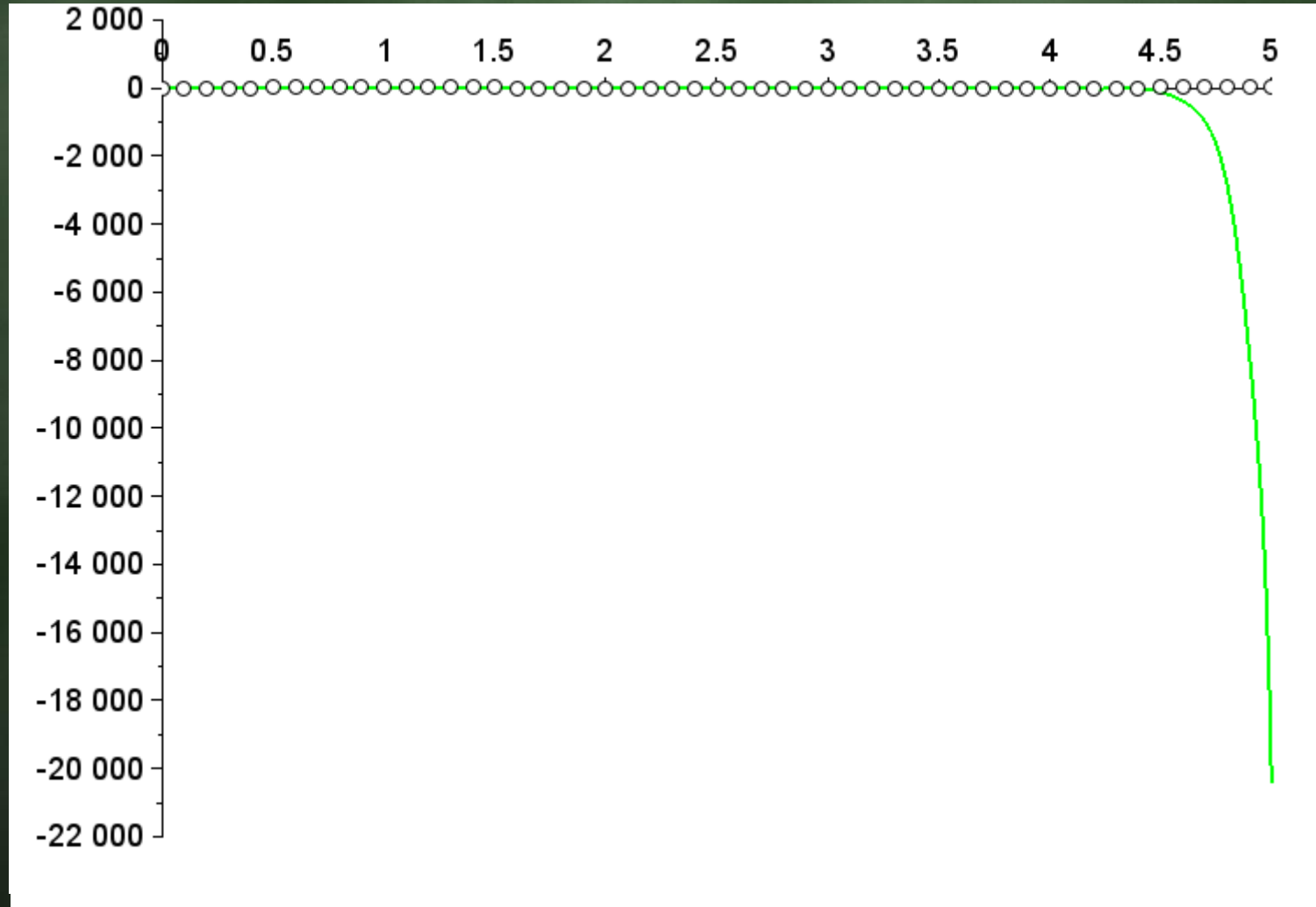
- 7
- 11
- 26
- $\langle 0.5, 2 \rangle$



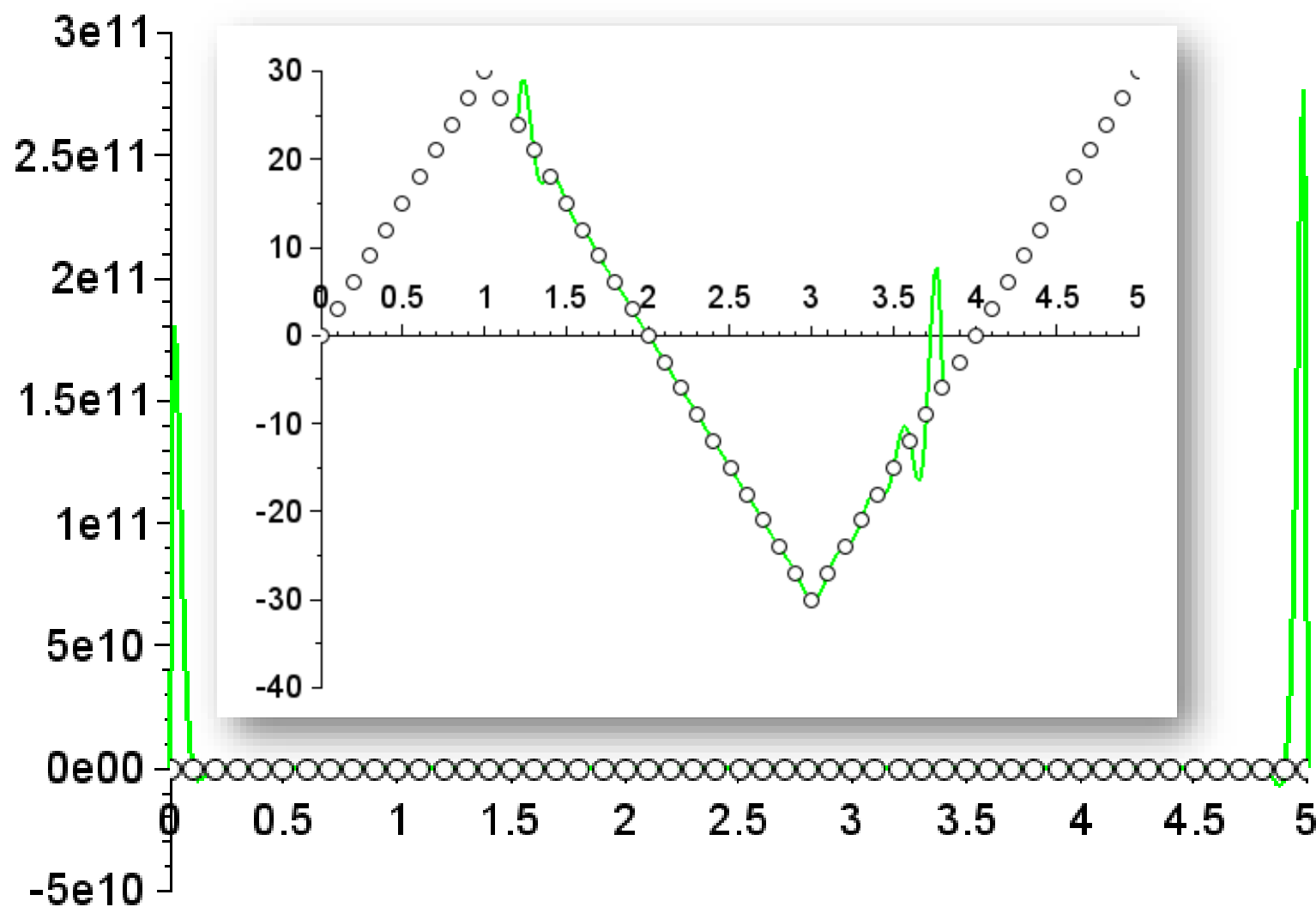
Interpolacja – sygnały okresowe



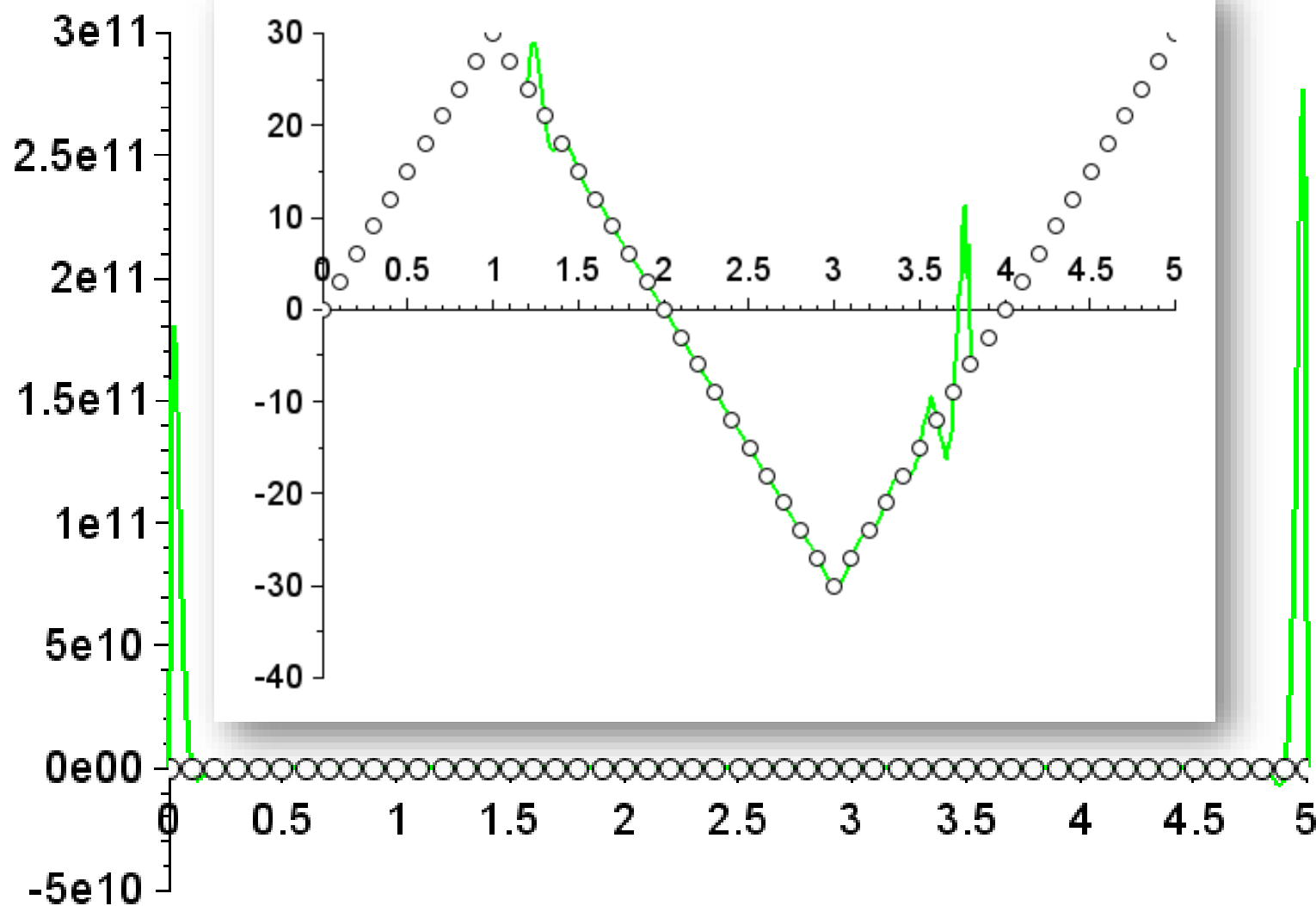
Interpolacja – Wielomianowa



Interpolacja – Lagrang'a

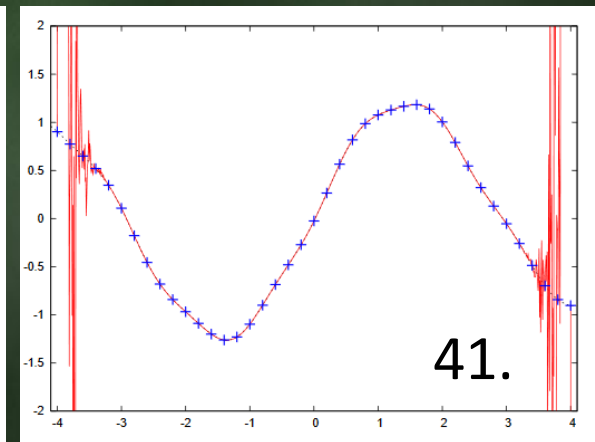
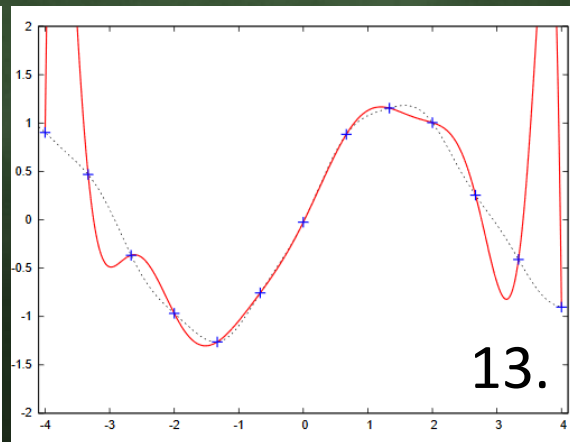
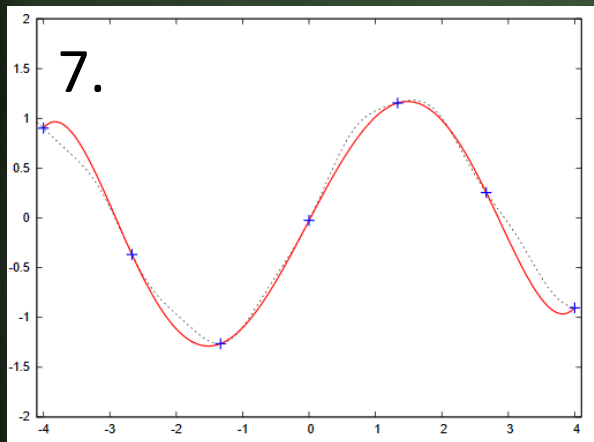


Interpolacja – Newton'a



Interpolacje

- Interpolacje wielomianowe algebraiczne cechują się pogorszeniem jakości, szczególnie na końcach przedziału interpolacji.
- Zjawisko nasila się wraz ze wzrostem stopnia wielomianu, a więc liczby węzłów interpolacji.
- Zjawisko to nazywa się Oscylacjami Rungego



Interpolacja trygonometryczna

- Interpolacje algebraiczne wielomianowe z racji swoich właściwości nie bardzo nadają się do opisu sygnałów okresowych
- Interpolację trygonometryczną stosuje się do wyznaczania funkcji interpolującej dla funkcji okresowych (np. sinusoidalnych)
- Dobrze nadaje się do interpolacji sygnałów elektrycznych czy drgań mechanicznych

Interpolacja trygonometryczna

Dana jest funkcja okresowa f o okresie 2π , zmiennej rzeczywistej o wartościach zespolonych:

$$f(x+2\pi)=f(x)$$

Dowolną funkcję okresową o okresie t można poprzez podstawienie $x = \frac{2\pi}{t}y$ sprowadzić do funkcji okresowej 2π

$$g(y+t) = g(y)$$

$$f(x) = g\left(\frac{xt}{2\pi}\right)$$

Można więc przy takim uogólnieniu rozpatrywać tylko funkcje o okresie 2π .

Interpolacja trygonometryczna

Interpolacja trygonometryczna sprowadza się do określenia dla nieznanej funkcji f wielomianu trygonometrycznego F przyjmującej dla danych węzłów interpolacyjnych te same wartości.

$$F_n(x) = \sum_{m=0}^n c_m e^{j \cdot m \cdot x} = \sum_{m=0}^n c_m (\sin(m \cdot x) + j \cos(m \cdot x))$$

$$F_n(x_k) = f(x_k)$$

Interpolacja trygonometryczna

- Warunki interpolacyjne można zapisać w postaci $n+1$ równań niewiadomych $c_0, c_1, \dots, c_n$

$$\sum_{m=0}^n c_m e^{jmx_k} = f(x_k)$$

- Macierz tego układu jest nieosobliwą macierzą Vandermode'a, układ ma jedno rozwiązanie.
- Wyznacznik macierzy jest różny od 0, przy założeniu że węzły x_k są niepowtarzalne.

Interpolacja trygonometryczna

Jako kryterium upraszczające do dalszych rozważań przyjmuje się że węzły interpolujące są równoodległe.

$$x_k = \frac{2k\pi}{n+1}, k = 0, 1, \dots, n$$

Interpolacja trygonometryczna

Funkcje e^{jmx} , $m=0, 1, 2, \dots, n$ tworzą układ ortogonalny, dla iloczynu skalarnego:

$$(f, g) = \sum_{k=0}^n f(x_k) g(x_k)$$

stąd

$$(e^{jlx}, e^{jmx}) = \sum_{k=0}^n e^{jlx_k} e^{-jmx_k} = \sum_{k=0}^n e^{j(l-m)\frac{2k\pi}{n+1}} =$$

$$= \begin{cases} \frac{e^{j(l-m)2\pi} - 1}{e^{j(l-m)\frac{2\pi}{n+1}} - 1} = 0 & \text{dla } l \neq m \\ n+1 & \text{dla } l = m \end{cases}$$

Interpolacja trygonometryczna

- Współczynniki wielomianu trygonometrycznego c_m są równe:

$$c_m = \frac{(f, e^{jmx})}{n+1} = \frac{1}{n+1} \sum_{k=0}^n f(x_k) e^{-jmx_k}$$

- Współczynniki c_m są równe współczynnikom rozwinięcia Fouriera funkcji f .
- Wyznaczanie współczynników wielomianu trygonometrycznego funkcji f określa się mianem **dyskretnej transformaty Fouriera**.

Interpolacja trygonometryczna

Trygonometryczny wielomian
interpolacyjny funkcji f oparty na węzłach

interpolacyjnych $x_k = \frac{2k\pi}{n+1}, k = 0, 1, \dots, n$

można przedstawić w postaci:

$$F_n(x) = \frac{1}{2}a_0 + \sum_{k=1}^m (a_k \cos(kx) + b_k \sin(kx)) + \frac{\delta}{2}a_{m+1} \cos((m+1)x)$$

gdzie:

$$\begin{cases} \delta = 0 & m = \frac{n}{2} & \text{gdy } n \text{ parzyste} \\ \delta = 1 & m = \frac{n-1}{2} & \text{gdy } n \text{ nieparzyste} \end{cases}$$

Interpolacja trygonometryczna

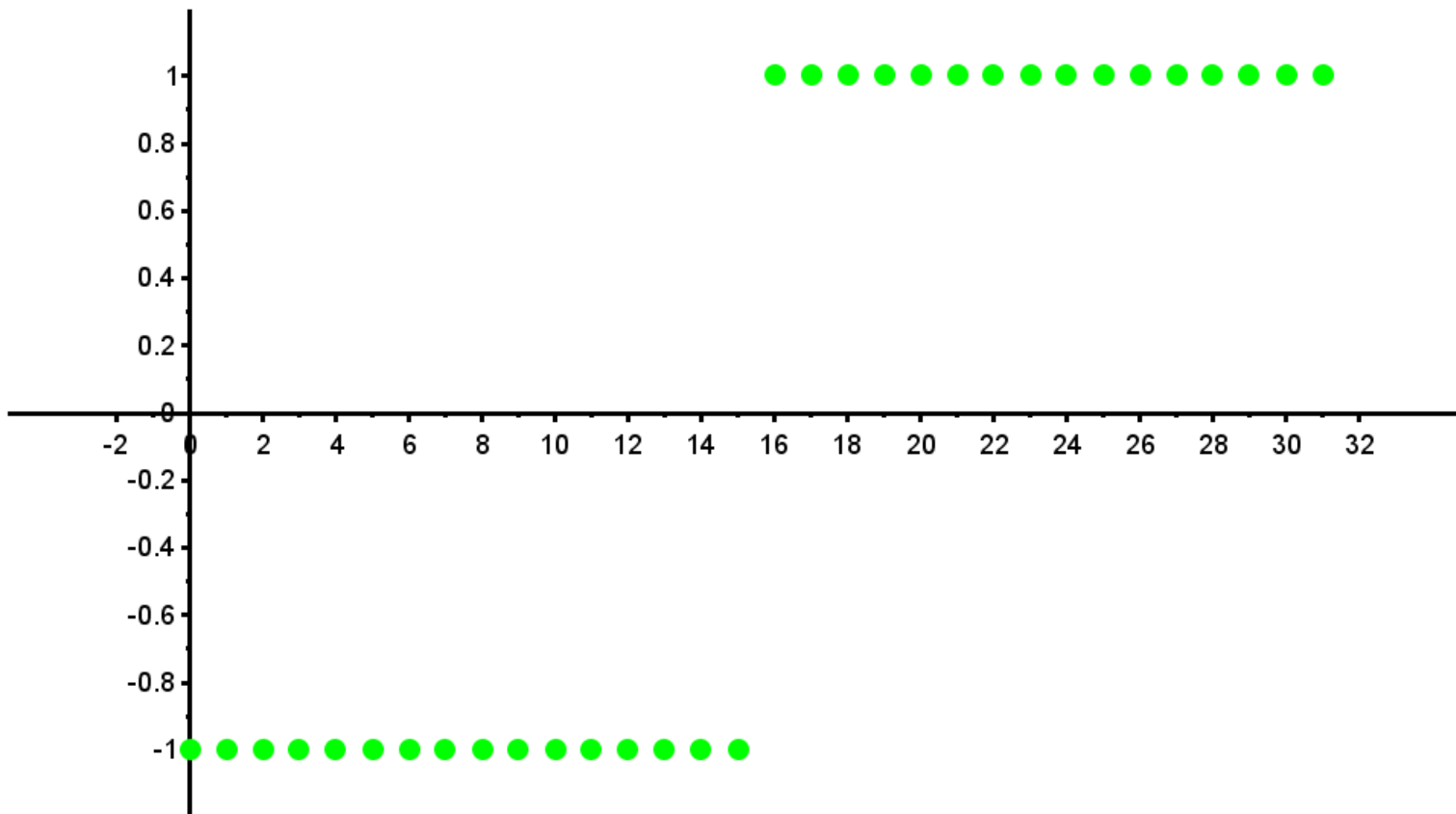
Współczynniki wielomianu wyznaczone są z zależności:

$$a_i = \frac{2}{n+1} \sum_{k=0}^n f(x_k) \cos(ix_k), \text{ dla } i = 0, 1, 2, \dots, m$$

$$b_i = \frac{2}{n+1} \sum_{k=0}^n f(x_k) \sin(ix_k), \text{ dla } i = 1, 2, \dots, m$$

Przykład 4

Określić algorytm interpolacyjny funkcji f dla podanych węzłów interpolacyjnych.



Przykład 4

$n=31$ – nieparzyste, więc $m=(n-1)/2=15$ i $\delta=1$

Na podstawie zależności: $x = \frac{2\pi}{t}y$ wyznaczane są kolejne węzły interpolacyjne.

$$a_i = \frac{2}{32} \sum_{k=0}^{31} f(x_k) \cos(ix_k), \text{ dla } i = 0, 1, 2, \dots, 16$$

$$b_i = \frac{2}{32} \sum_{k=0}^{31} f(x_k) \sin(ix_k), \text{ dla } i = 1, 2, \dots, 15$$

Przykład 4

Na podstawie wyznaczonego wielomianu trygonometrycznego wyznaczane są interpolowane wartości funkcji.

$$F_{31}(x) = \frac{a_0}{2} + \sum_{k=1}^{15} (a_k \cos(kx) + b_k \sin(kx)) + \frac{a_{16}}{2} \cos(16x)$$

Przykład 4

```

1  clc;clear;
2  n=32;
3  M=2000;
4  x0=[0;1;2;3;4;5;6;7;8;9;
5  17;18;19;20;21;22;23;24;
6  y=[-1;-1;-1;-1;-1;-1;-1;
7  -1;-1;-1;-1;1;1;1;1;1;1;
8  f=zeros(1,M);
9  xk=zeros(1,n);
10 t=zeros(1,M);
11 t2=zeros(1,M);
12 for i=1:n
13     xk(i)=2*pi/31*x0(i)
14 end
15 for i=2:M
16     t(i)=xk(n)/M*i;
17     t2(i)=31/M*i;
18 end
19 m=15;

```

```

column 1 to 5
0.    0.2026834    0.4053668    0.6080502    0.8107336

column 6 to 10
1.013417    1.2161004    1.4187838    1.6214672    1.8241506

column 11 to 15
2.026834    2.2295174    2.4322008    2.6348842    2.8375676

column 16 to 20
3.040251    3.2429344    3.4456177    3.6483011    3.8509845

column 21 to 24
4.0536679    4.2563513    4.4590347    4.6617181

column 25 to 28
4.8644015    5.0670849    5.2697683    5.4724517

column 29 to 32
5.6751351    5.8778185    6.0805019    6.2831853

```

Przykład 4

```
20 for i=1:m+2
21     for j=1:n
22         ak(i)=ak(i)+y(j)*cos((i-1)*xk(j));
23     end
24     ak(i)=2/n*ak(i);
25 end
26 for i=1:m
27     for j=1:n
28         bk(i)=bk(i)+y(j)*sin(i*xk(j));
29     end
30     bk(i)=2/n*bk(i);
31 end
32 for k=1:M
33     for i=1:m
34         f(k)=f(k)+(ak(i+1)*cos(i*t(k))+bk(i)*sin(i*t(k)));
35     end
36     f(k)=f(k)+ak(1)/2+(ak(m+2)/2)*cos((n+1)*t(k));
37 end
38 plot2d(x0,y,3);
39 plot2d(t2,f,2);
```


Przykład 4

$$F_{31}(x) = \frac{a_0}{2} + \sum_{k=1}^{15} (a_k \cos(kx) + b_k \sin(kx)) + \frac{a_{16}}{2} \cos(16x)$$

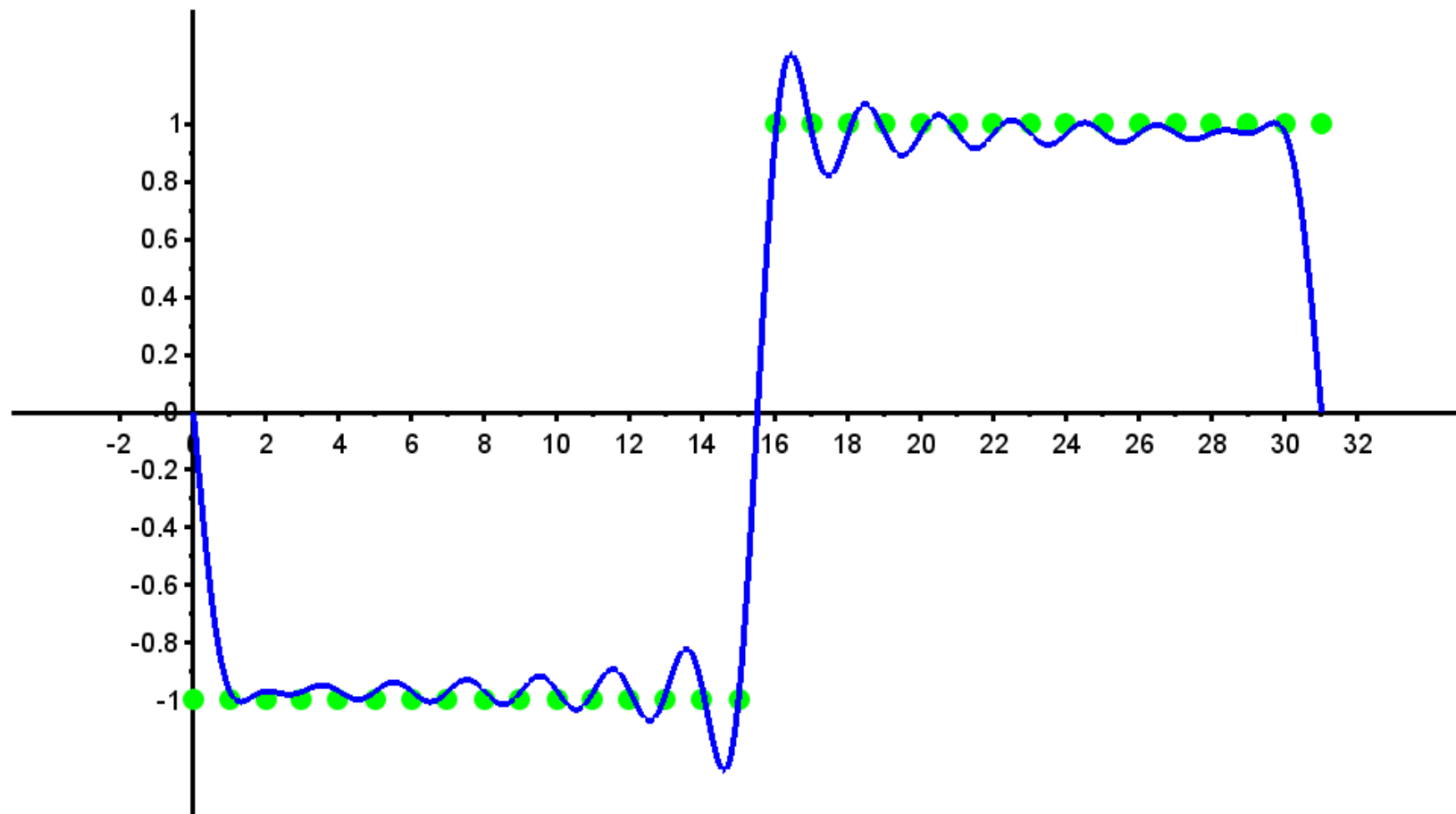
Na podstawie danych wejściowych wyznaczono współczynniki wielomianu: $a_i \approx 0$, dla $i = 0, 1, 2, \dots, 16$

Wielomian interpolacji trygonometrycznej przyjmuje postać:

$$\begin{aligned} F(x) \approx & -1.23 \sin(x) + 0.006 \sin(2x) - 0.41 \sin(3x) \\ & + 0.013 \sin(4x) - 0.24 \sin(5x) + 0.02 \sin(6x) \\ & - 0.17 \sin(7x) + 0.027 \sin(8x) - 0.13 \sin(9x) \\ & + 0.035 \sin(10x) - 0.1 \sin(11x) + 0.043 \sin(12x) \\ & - 0.081 \sin(13x) + 0.054 \sin(14x) - 0.066 \sin(15x) \end{aligned}$$

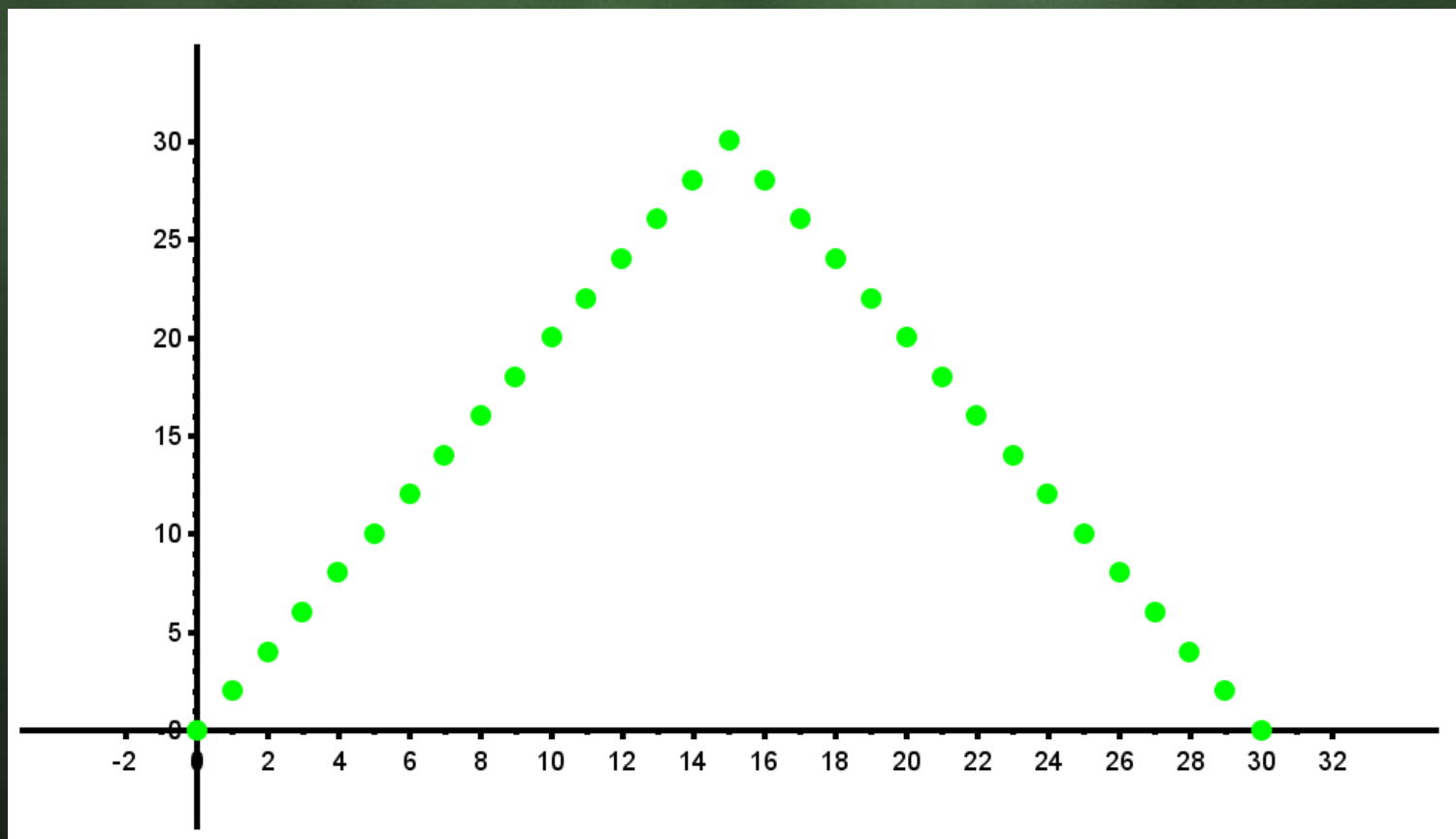
Przykład 4

15:20



Przykład 5

Wyznaczyć wielomian interpolacji trygonometrycznej zadanego sygnału:



Przykład 4

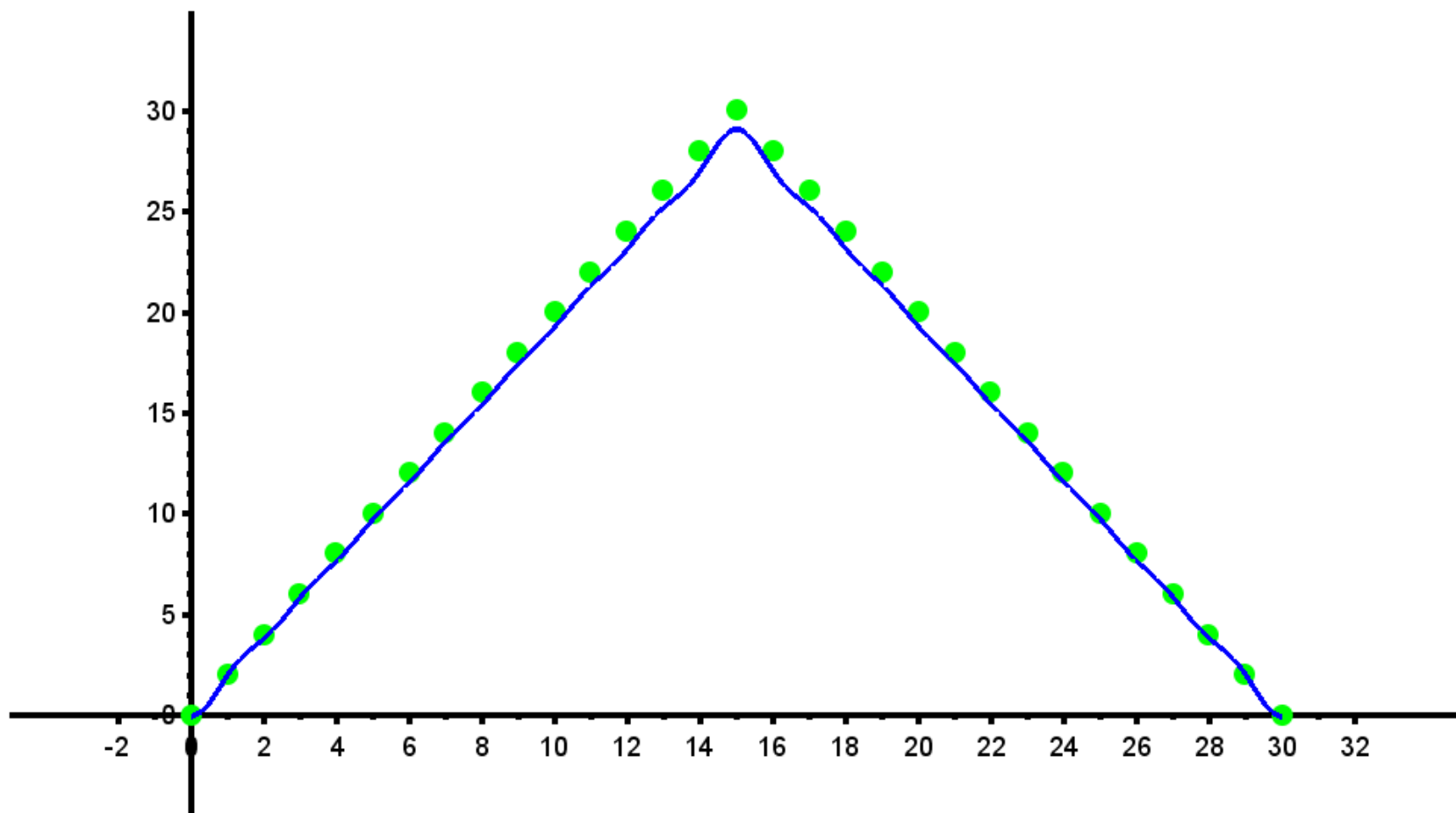
$$F_{31}(x) = \frac{a_0}{2} + \sum_{k=1}^{15} (a_k \cos(kx) + b_k \sin(kx))$$

Na podstawie danych wejściowych wyznaczono współczynniki wielomianu: $b_i \approx 0$, dla $i = 1, 2, \dots, 15$

Wielomian interpolacji trygonometrycznej przyjmuje postać:

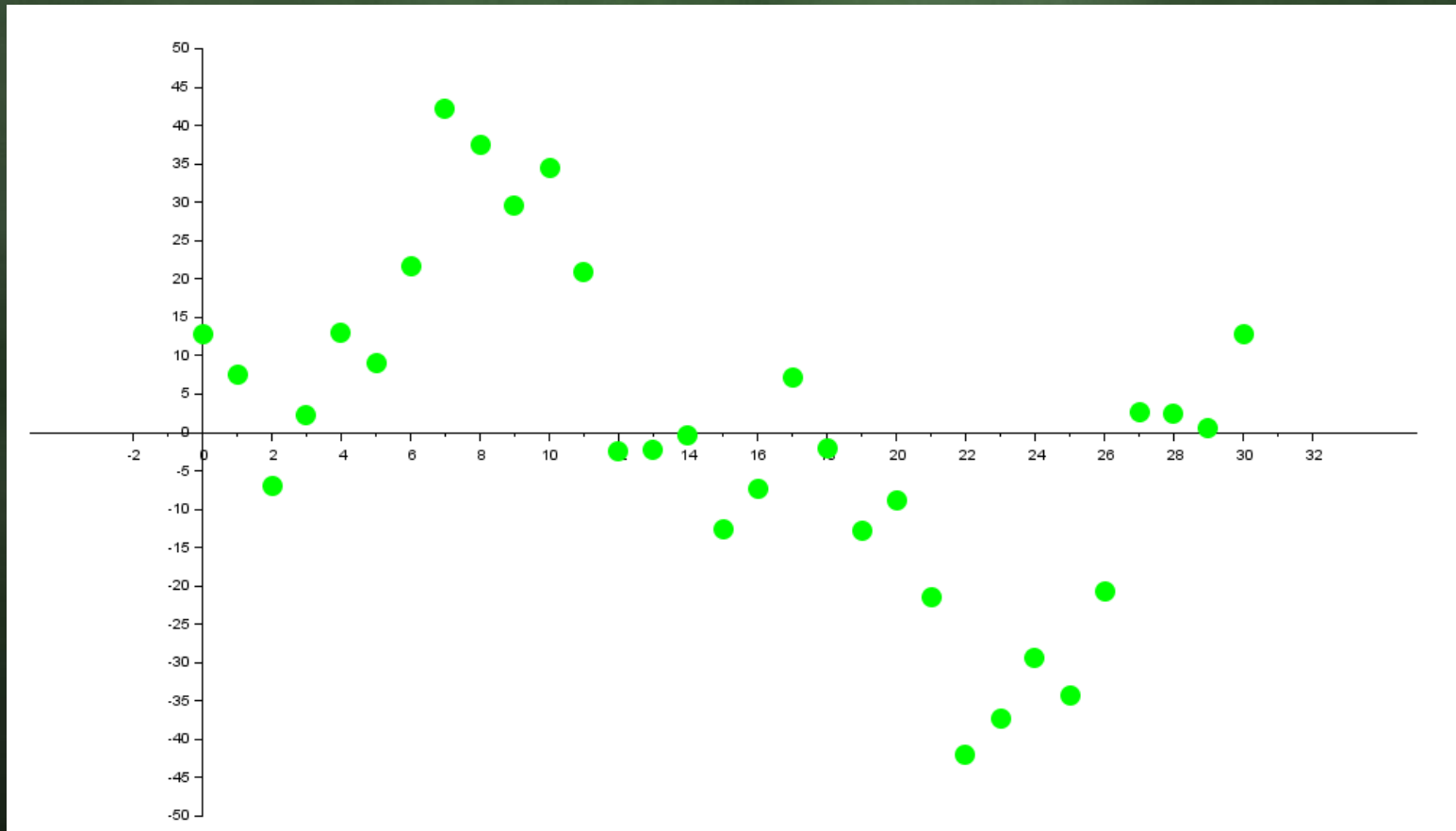
$$\begin{aligned} F(x) \approx & 14.5 - 11.8 \cos(x) - 1.35 \cos(3x) - 0.52 \cos(5x) \\ & - 0.29 \cos(7x) - 0.2 \cos(9x) - 0.15 \cos(11x) \\ & - 0.13 \cos(13x) - 0.13 \cos(15x) \end{aligned}$$

Przykład 5



Przykład 5

Wyznaczyć interpolacyjny wielomian trygonometryczny dla sygnału



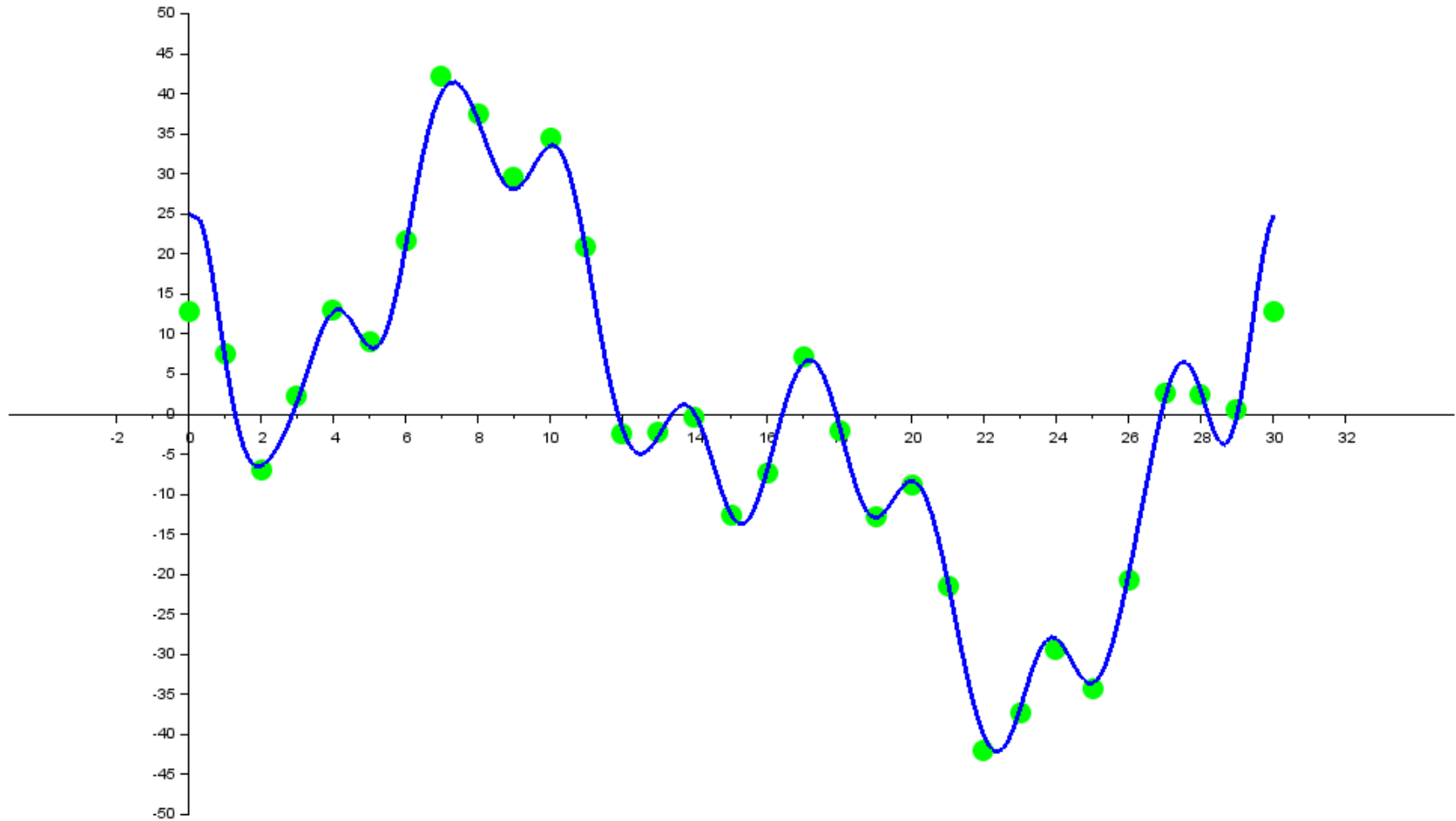
Przykład 5

$$F_{31}(x) = \frac{a_0}{2} + \sum_{k=1}^{15} (a_k \cos(kx) + b_k \sin(kx))$$

Na podstawie danych wejściowych wyznaczono współczynniki wielomianu. Wielomian interpolacji trygonometrycznej przyjmuje postać:

$$F(x) \approx 0.41 + 24 \sin(x) + 7.59 \cos(3x) - 11.7 \sin(3x) + 6.29 \cos(9x) + 5.47 \sin(9x)$$

Przykład 5



Szybka transformata Fouriera

- Przekształcenie Fouriera (transformacja) pozwala na rozkład analizowanego sygnału na częstotliwości składowe.
- Algorytm szybkiej transformaty Fouriera **FFT** służy do efektywnego wyznaczania współczynników szeregu trygonometrycznego

$$F_n(x) = \sum_{m=0}^n c_m e^{j \cdot m \cdot x}$$

Szybka transformata Fouriera

Dekompozycja sygnału na składowe w postaci szeregu sygnałów sinusoidalnych umożliwia interpolację sygnału w postaci szeregu trygonometrycznego poprzez identyfikację parametrów tych sygnałów:

- częstotliwości,
- amplitudy,
- fazy.

FFT znajduje szerokie zastosowanie w wielu dziedzinach: przetwarzaniu sygnałów cyfrowych, obrazowaniu medycznym, rozwiązywaniu równań różniczkowych, poszukiwaniu złóż ropy i gazu, Monitorowaniu pracy maszyn, sterowaniu samolotów i statków kosmicznych.

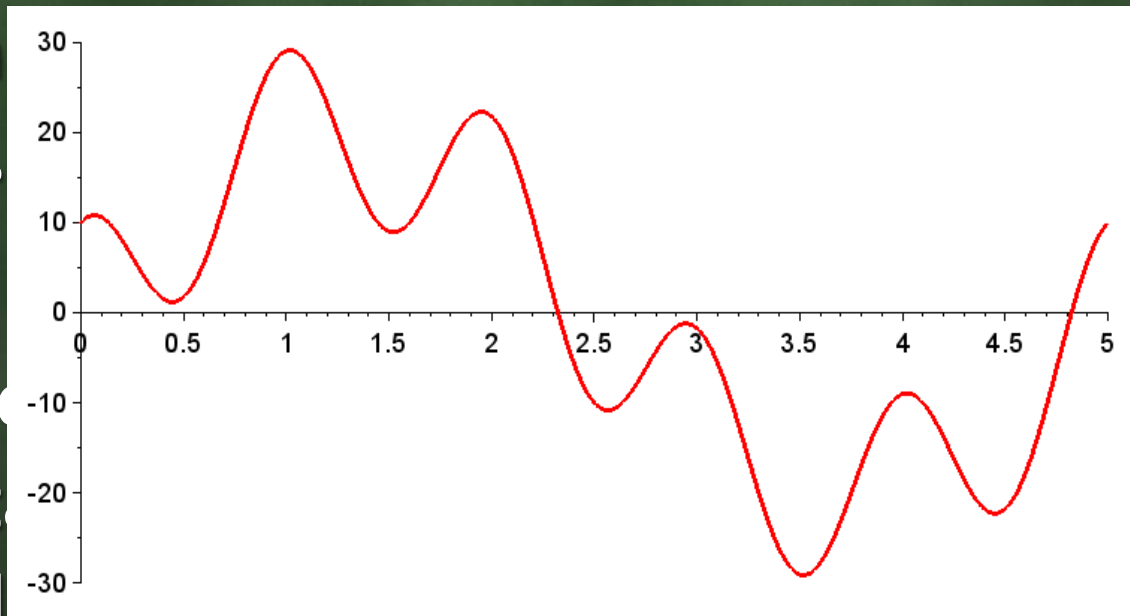
Szybka transformata Fouriera

- Funkcja **FFT** jest dostępna w większości programów obliczeniowych.
- W środowisku Scilaba szybką transformatę Fouriera można przeprowadzać dla wektora lub macierzy danych.
- Podstawowa składnia funkcji **fft()** wymaga podania jedynie na wejście analizowanego sygnału, na wyjściu otrzymujemy odpowiednio wektor lub macierz wartości zespolonych

$$X = \text{fft}(A) ;$$

Przykład 6 - FFT

- Przeprowadzić analizę sygnału, określić parametry interpolacyjnego wielomianu trygonometrycznego (amplitudę i fazę).



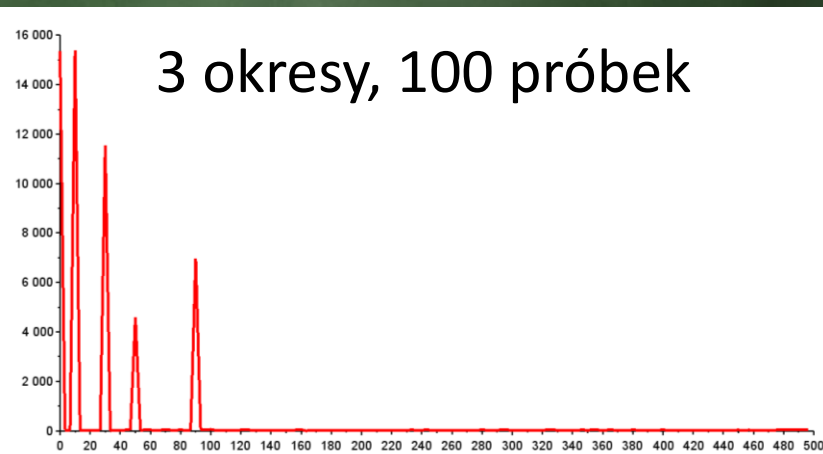
- Przeprowadzić analizę sygnału, określić parametry interpolacyjnego wielomianu trygonometrycznego (amplitudę i fazę).

$$F(t) = 20\sin(0.2t) + 10\cos(t)$$

Przykład 6 - FFT

- Poprawna analiza wymaga posiadania informacji o liczbie okresów analizowanego sygnału w próbce wejściowej
- Im większa częstotliwość próbkowania tym dokładniejsze wyniki obliczeń, ostrzejsze piki
- Liczba okresów powinna być liczbą całkowitą
- Im więcej okresów lub próbek tym amplituda większa
- Zwracany przez `fft()` wektor ma symetryczną budowę względem centralnego elementu tablicy wyników

Przykład 6 - FFT



Przykład 6 - FFT

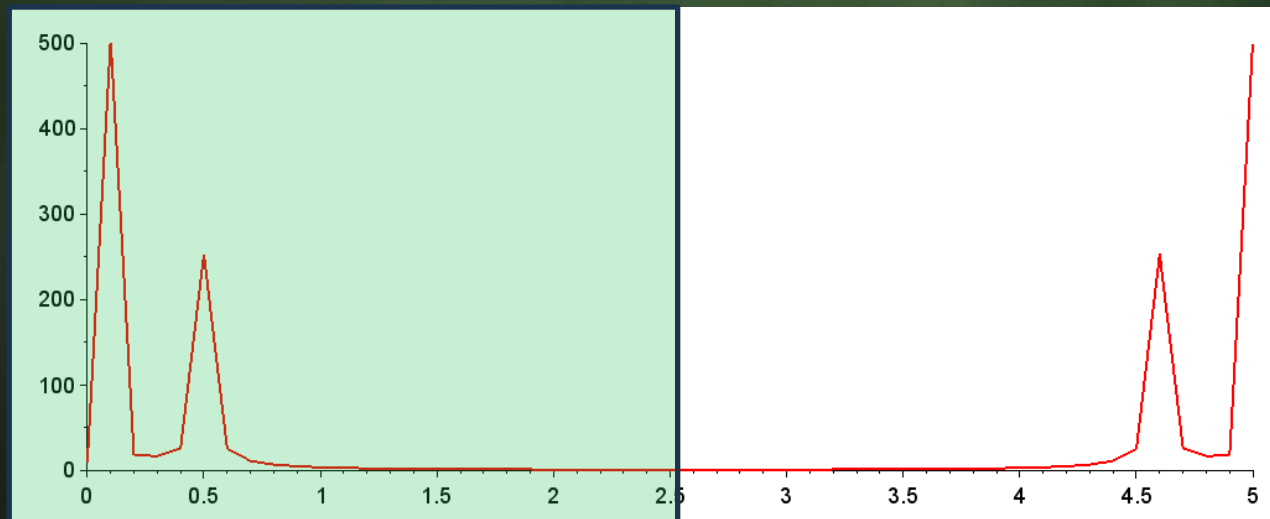
- Każda wartość zwracana przez funkcję `fft()` jest w postaci liczby zespolonej, niosącej informacje o amplitudzie i fazie analizowanego sygnału
- Każda wartość jest powiązana z inną częstotliwością składowej sygnału wejściowego
- Informacja o częstotliwości próbkowania pozwala na wygenerowanie wektora częstotliwości
- Wymagana jest informacja o liczbie próbek

Przykład 8 - FFT

- Konieczna jest identyfikacja ekstremów które odpowiadają poszukiwanym składowym wielomianu interpolacyjnego
- Ze względu na symetrie ograniczyć się można do połowy wartości wektora zwróconego przez FFT
- Odfiltrowanie szumu można zrealizować poprzez wyszukanie wartości wektora FFT większych od średniej

Przykład 6 - FFT

```
fft_y=fft(F);  
for x=1:N2  
    modulFFT(x)=abs(fft_y(x));  
end  
k1=find(modulFFT>mean(modulFFT));
```

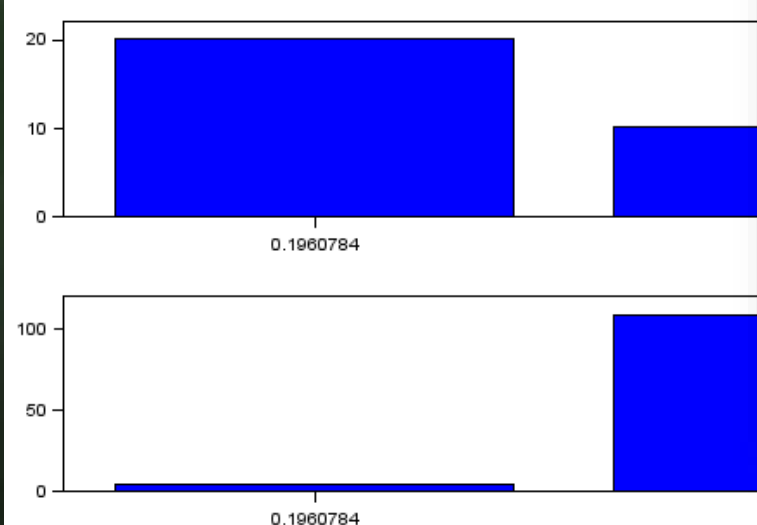


Przykład 6 - FFT

- Amplituda składowych wielomianu trygonometrycznego równe są ilorazowi modułu wartości transformaty Fouriera i iloczynu połowy liczby próbek i liczby okresów, wyznaczanych dla określonych ekstremów
- Częstotliwości określa poprzez iloraz częstotliwości próbkowania i iloczynu liczby próbek i okresów
- Kąty fazowe poszczególnych składowych wielomianu wyznacza się jako argument z liczby zespolonej uzyskanej z funkcji FFT dla każdej z wyznaczonych częstotliwości

Przykład 6 - FFT

```
sample_rate=1/(t(2)-t(1));
czestotliwosc=(sample_rate/N)*(0:N2);
f=czestotliwosc(k1);
A=modulFFT(k1)/N2;
fi=atan(imag(fft_y),real(fft_y));
//fi=angle(fft_y);
faza=fi(k1)+%pi/2;
kat=faza*180/%pi;
```



"Częstotliwości harmoniczných"

0.1960784 0.9803922

"Amplitudy harmoniczných"

20.188497 10.140626

"Fazy harmoniczných"

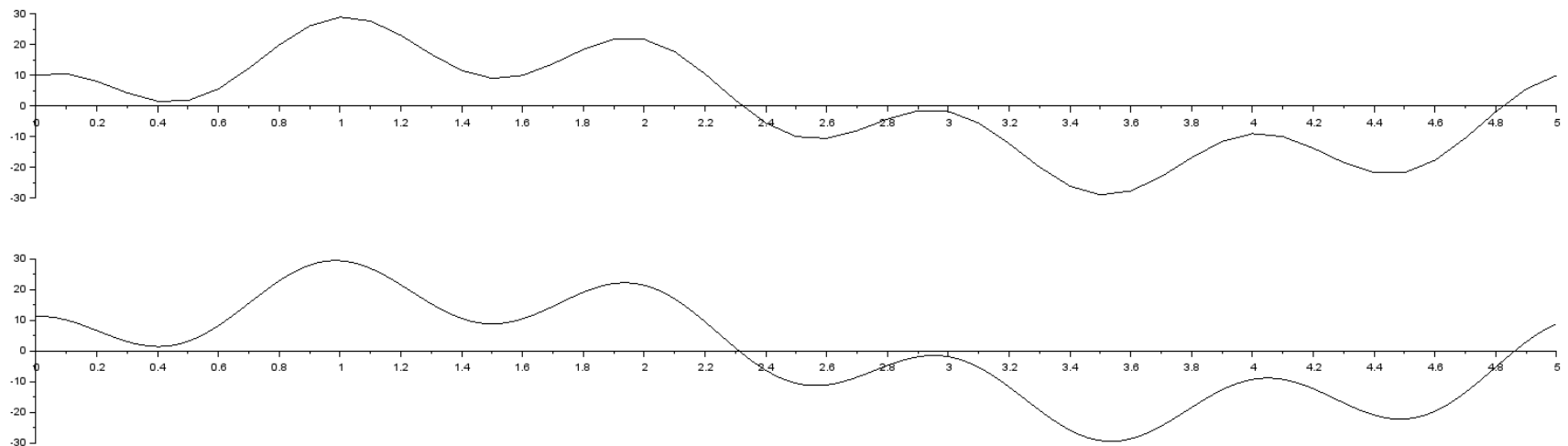
4.7093843 108.62337

Przykład 6 - FFT

```

liczba=5000;
U=zeros(1,liczba);
for y=1:liczba
    t2(y)=(y-1)*(t(N)/liczba);
    for x=1:length(A)
        U(y)=U(y)+A(x)*sin(2*pi*f(x)*t2(y)+faza(x));
    end
end
end

```

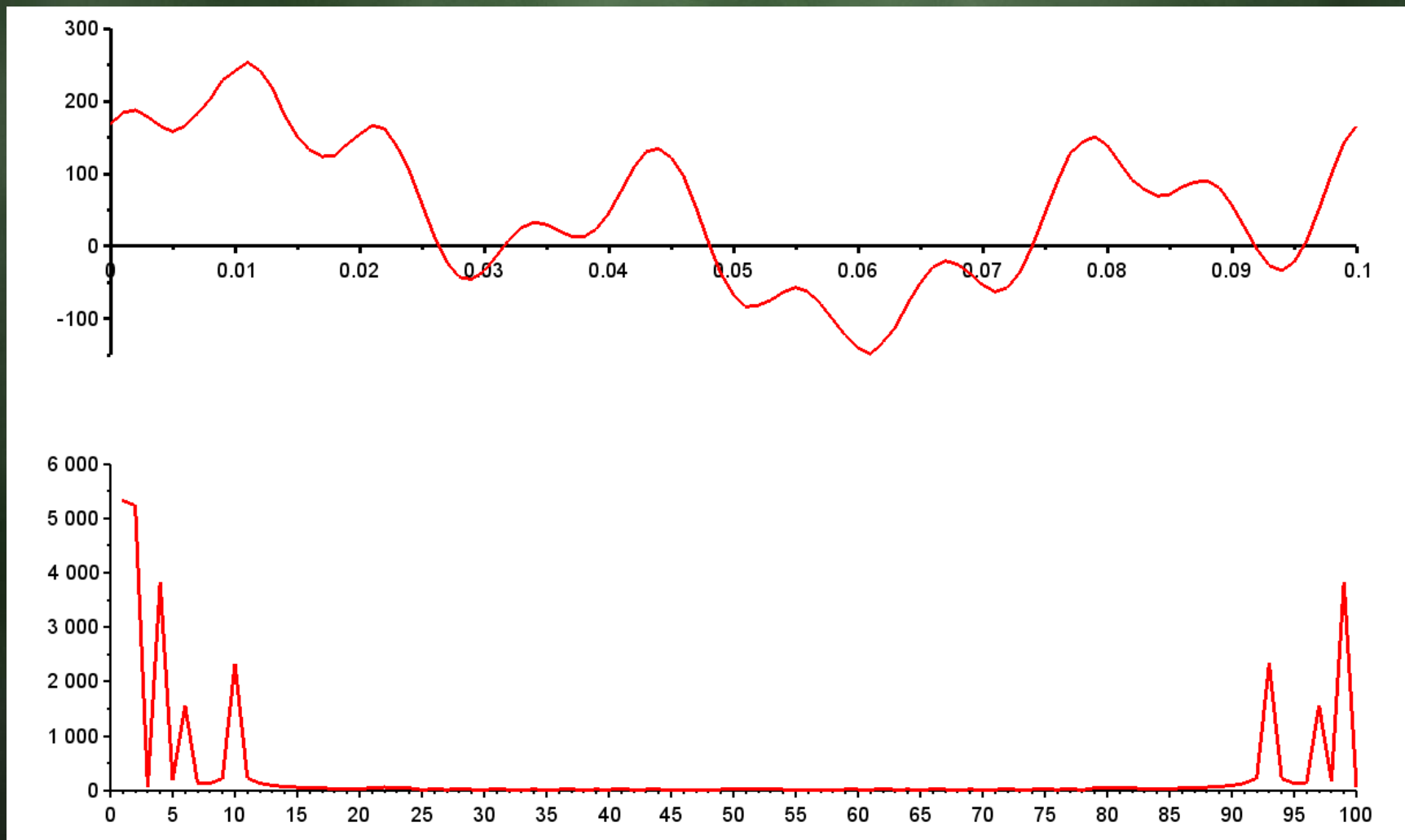


Przykład 7 - FFT

Dane wejściowe dla przykładu

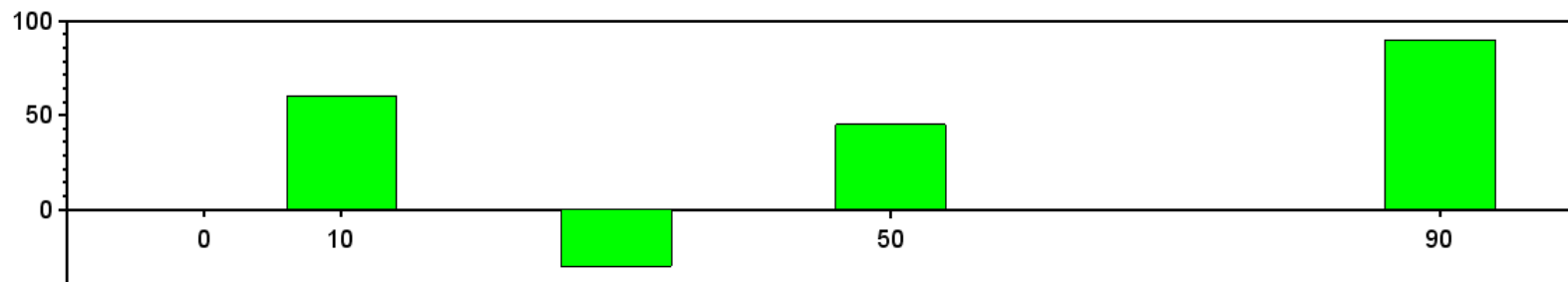
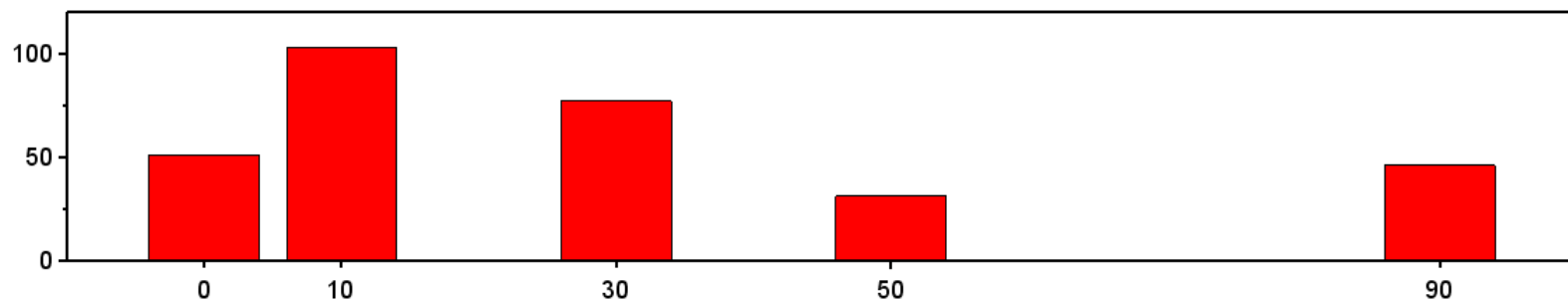
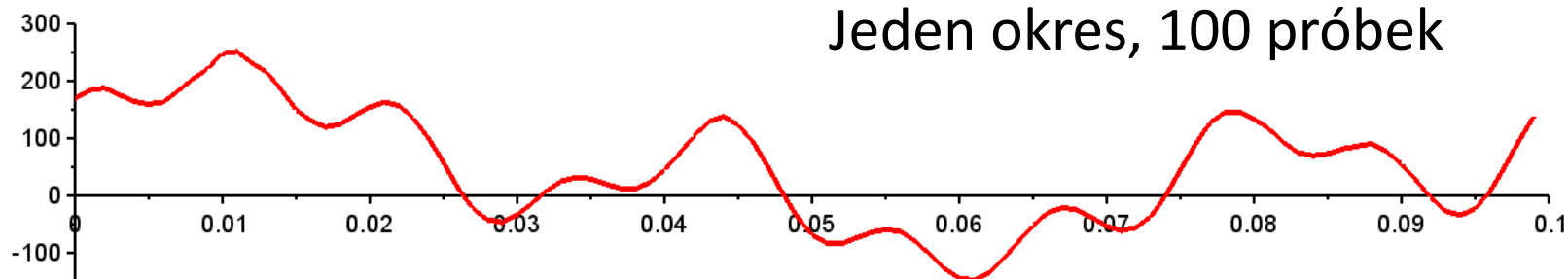
```
1 clc;clear;
2 f=10;//częstotliwość podstawowa
3 n=300;//liczba próbek
4 w=2*%pi*f;
5 O=1;
6 T=1/f;//okres
7 Um=[100;75;30;45];
8 H=[1;3;5;9];
9 fi=[60;-30;45;90];
10 delta=T/n;
11 df=1/delta//częstotliwość próbkowania
12 for i=1:(n)*O;
13     t(i)=(i-1)*T/n;
14     u(i)=Um(1)/2;
15     for j=1:4
16         u(i)=u(i)+Um(j)*sin(H(j)*w*t(i)+%pi/180*fi(j));
17     end;
18     u(i)=u(i)*(1+0.05*rand());//błędy pomiarowe
19 end
```

Przykład 7 - FFT



Przykład 7 FFT

Jeden okres, 100 próbek



Przykład 7 FFT

Częstotliwości wyznaczonych składowych wielomianu [Hz]

0. 10. 30. 50. 90.

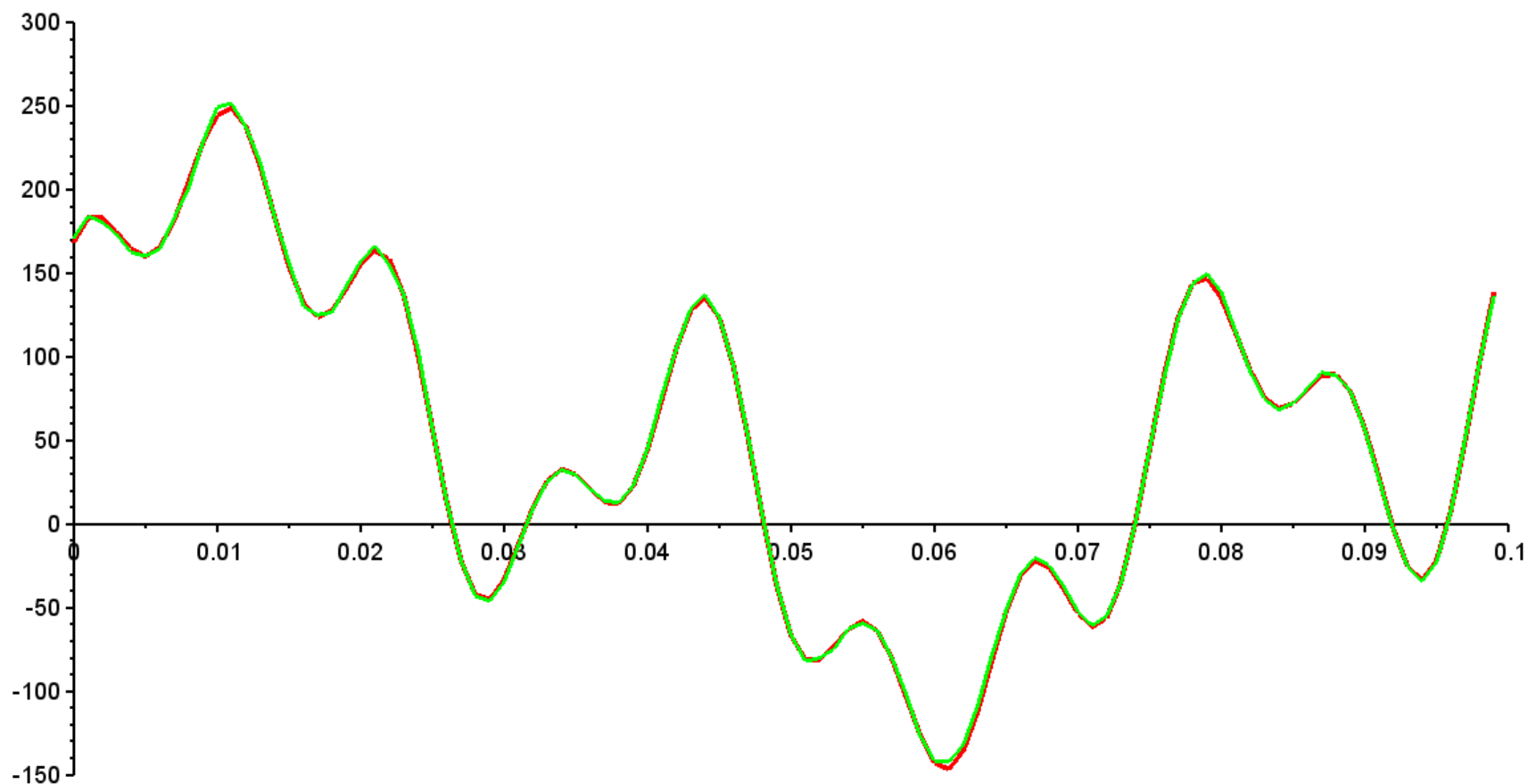
Amplituda wyznaczonych składowych wielomianu [V]

51. 102. 77. 31. 46.

Fazy wyznaczonych składowych wielomianu

0. 60. -30. 45. 90.

Przykład 7 - FFT



Przykład 7 -FFT

Częstotliwości wyznaczonych składowych wielomianu [Hz]

column 1 to 13

0. 7. 13. 20. 27. 33. 40. 47. 53. 60. 87. 93. 100.

column 14 to 15

107. 113.

Amplituda wyznaczonych składowych wielomianu [V]

column 1 to 13

68. 69. 68. 33. 59. 41. 8. 15. 23. 8. 25. 33. 13.

column 14 to 15

9. 7.

Fazy wyznaczonych składowych wielomianu

column 1 to 12

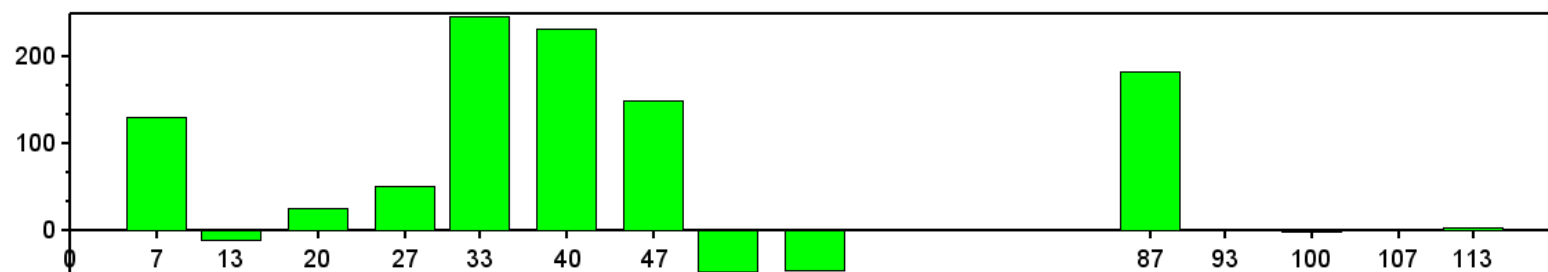
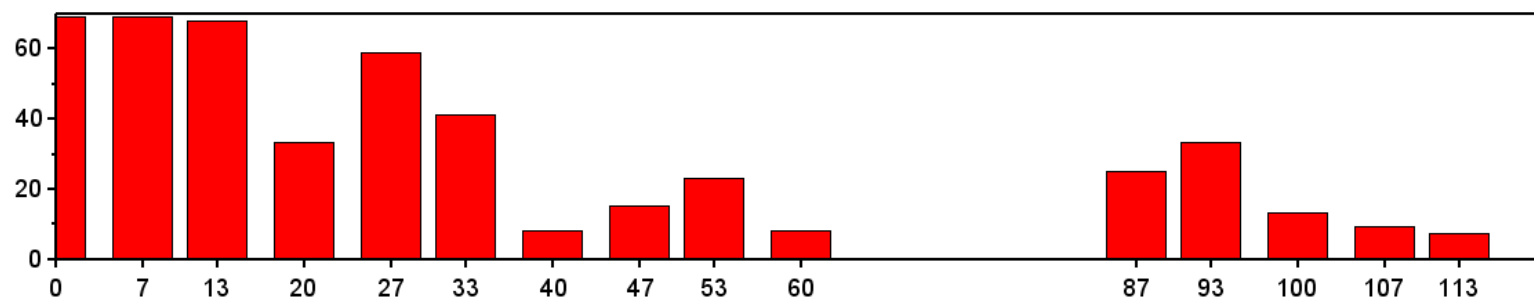
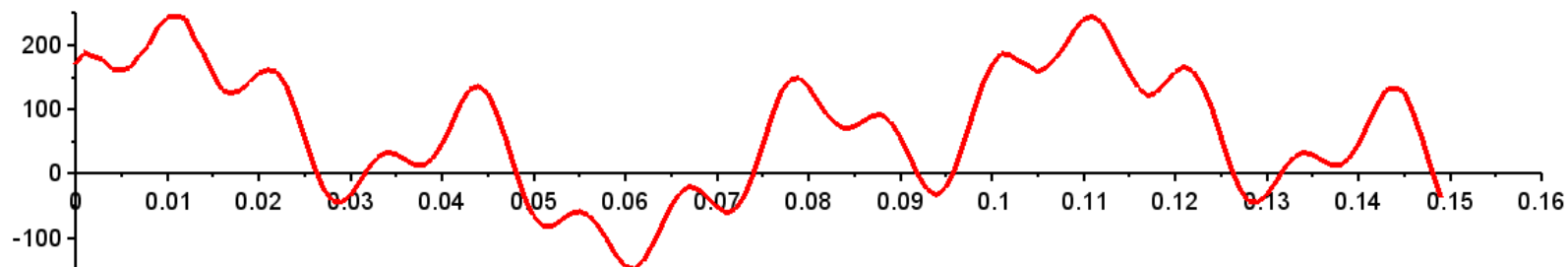
0. 130. -11. 24. 51. 246. 237. 148. -49. -52. 183. -1.

column 13 to 15

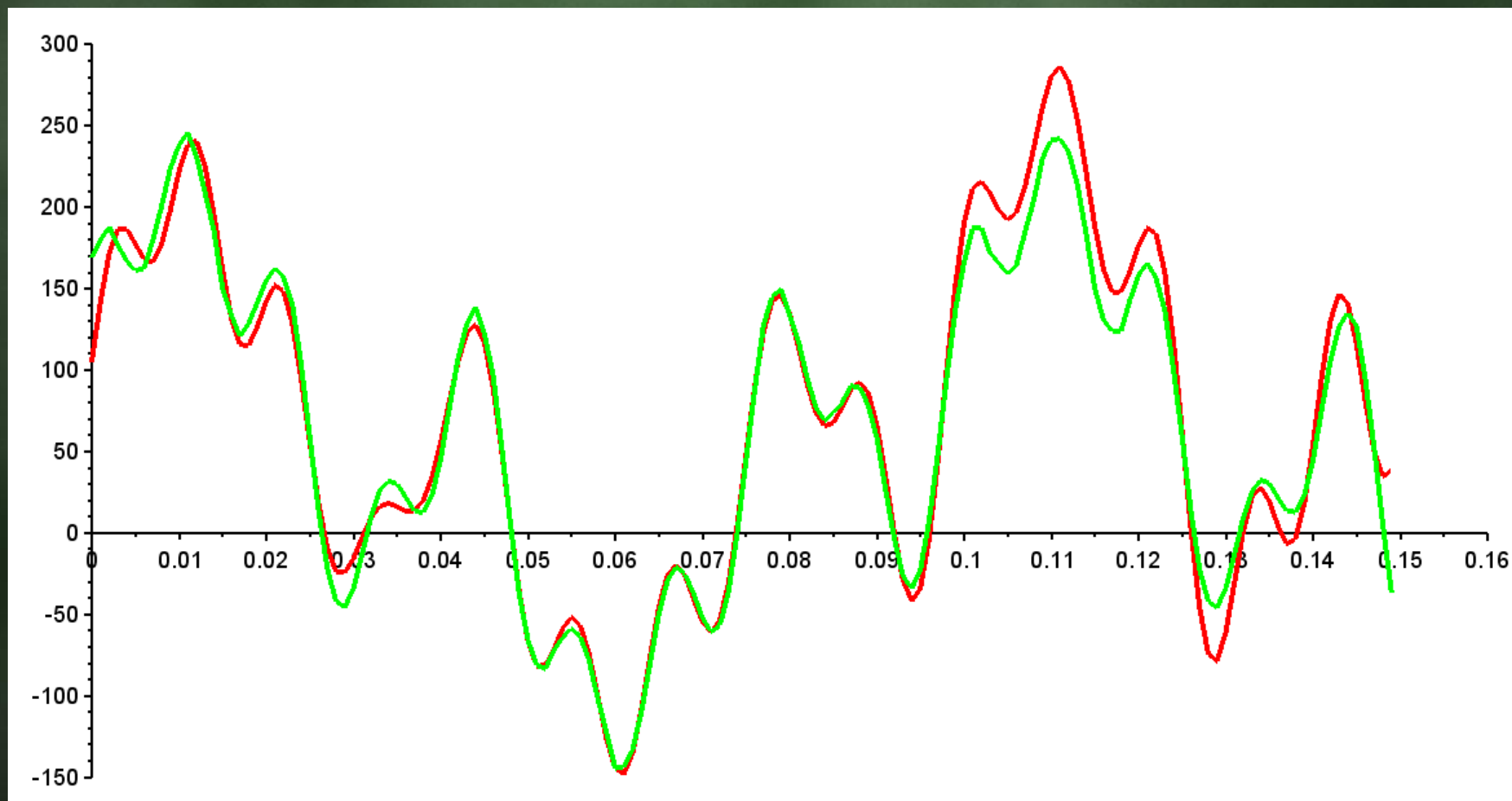
-2. 1. 1.

1.5 okresu, 100 próbek

Przykład 7 FFT



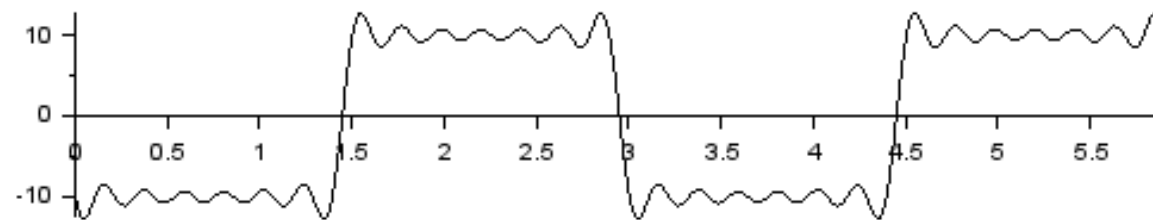
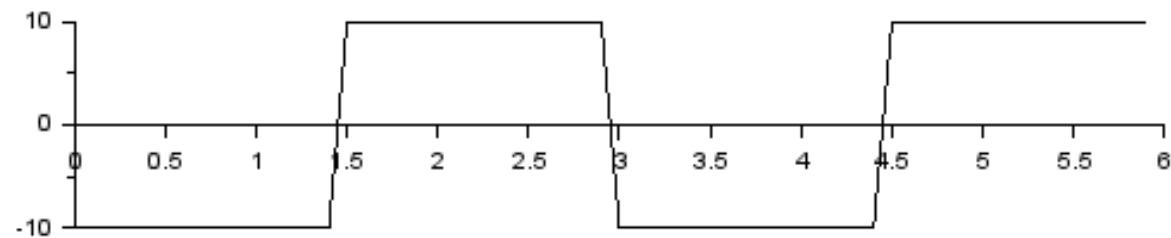
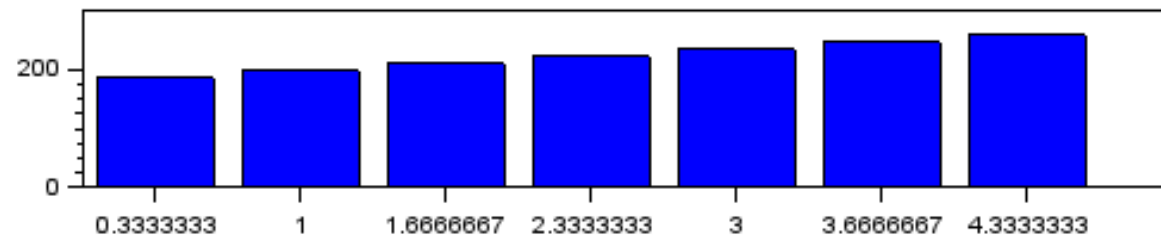
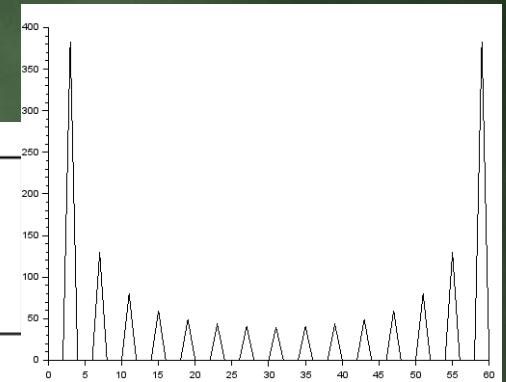
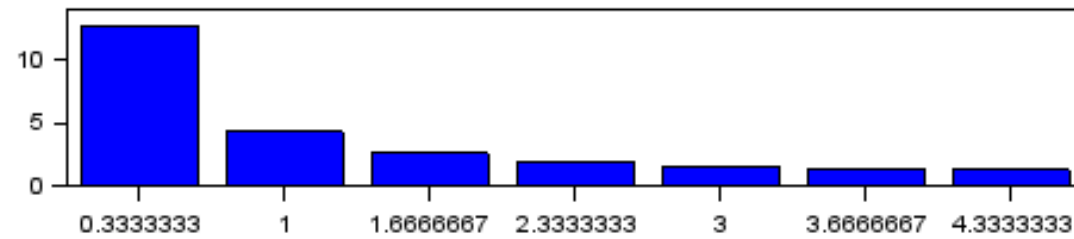
Przykład 7 FFT



FFT

- Zwiększenie częstotliwości próbkowania przy całkowitej liczbie okresów sygnału badanego w próbce wpływa nieznacznie na poprawę dokładności interpolacji
- Zbyt duża liczba próbek może doprowadzić do błędów obliczeniowych
- Jeżeli liczba okresów nie jest całkowita, zwiększenie liczby próbek drastycznie zwiększa liczbę wyznaczonych składowych wielomianu interpolującego, nie poprawiając jednocześnie dokładności interpolacji

FFT



()

Interpolujące funkcje sklejane

- Problemy związane z interpolacjami wielomianowymi (oscylacje Rungego) powodują że wygodniejsze może być interpolowanie za pomocą funkcji sklejanych
- Funkcją sklejaną (splajnem – ang. spline) nazywamy funkcję która:
 - lokalnie jest wielomianem rzędu k
 - jest $(k-1)$ - krotnie różniczkowalna w węzłach (z czego wynika że jej pochodne rzędu $k-2$ i niższych są ciągłe).
- Najczęściej stosuje się funkcje sklejane 3 stopnia (funkcje kubiczne)

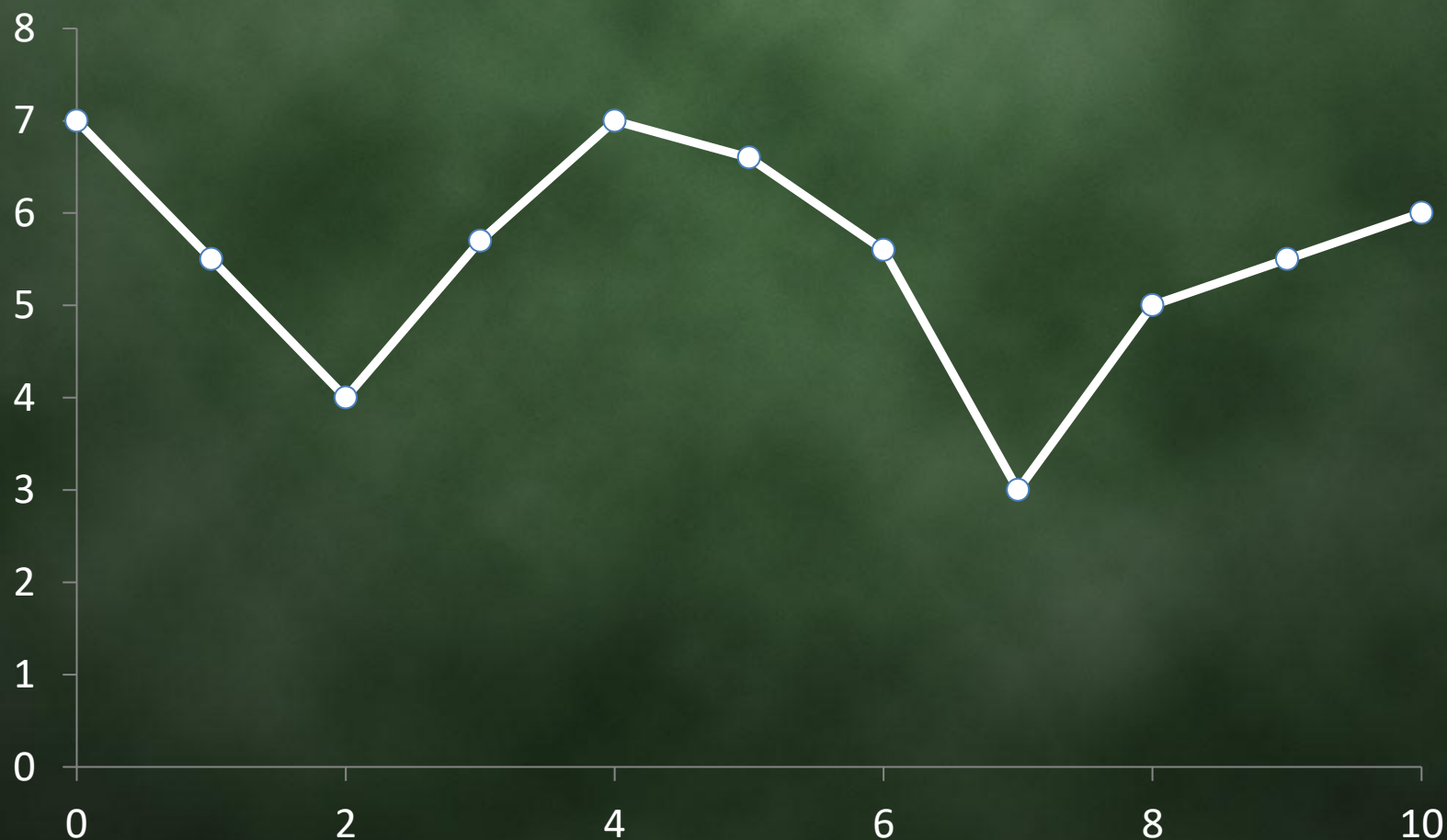
Interpolujące funkcje sklejane

Funkcja sklejana stopnia 0



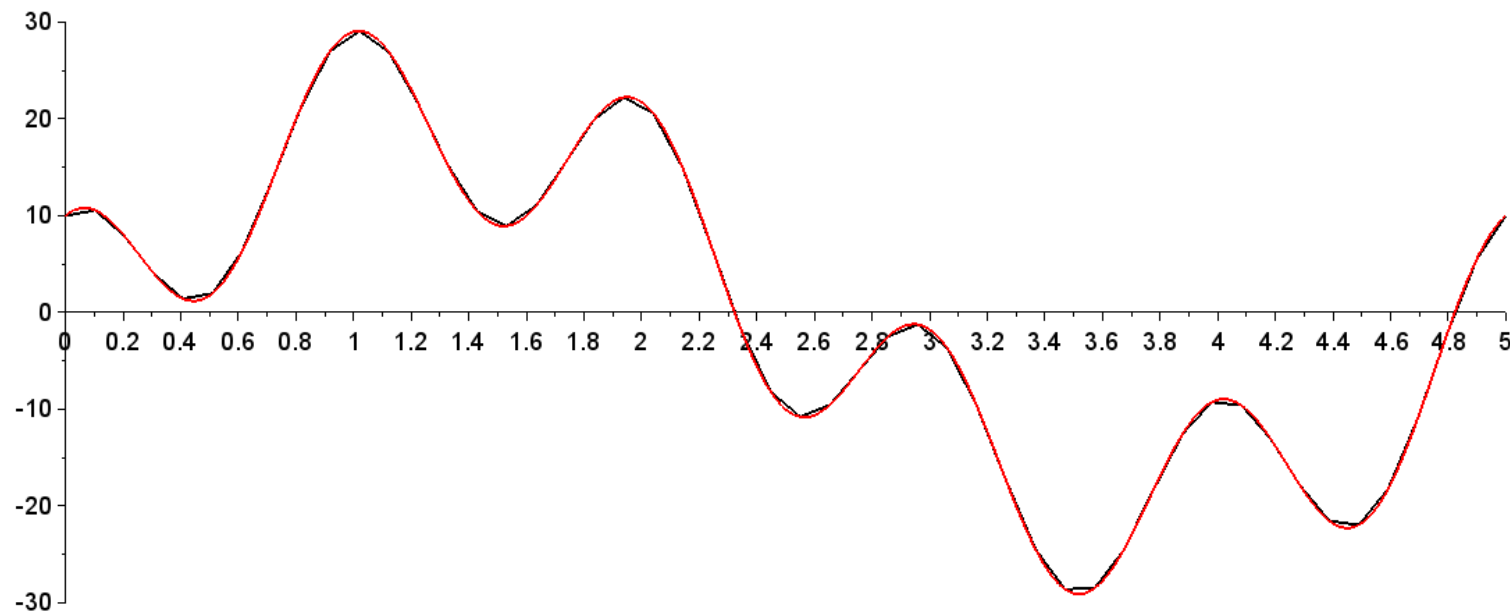
Interpolujące funkcje sklejane

Funkcja sklejana stopnia 1



Interpolujące funkcje sklejane

```
x=linspace(t(1),t(m),n);  
fun=[0];  
for i=1:n  
    k=max(find(t<=x(i)) );  
    fun(i)=A(k)*x(i)+B(k);  
end
```



Interpolujące funkcje sklepane

- Interpolację funkcjami sklejanymi 3 stopnia można przeprowadzić dwoma sposobami:
- Analizowany sygnał dzieli się na tyle podzbiorów, aby każdy z nich składał się z trzech węzłów interpolacyjnych
- Określenie równania wielomianu 3 stopnia jest więc zagadnieniem łatwym
- Metoda ta sprawia że mogą pojawić się na krańcach przedziałów „kanty”

Interpolujące funkcje sklepane

Zakładając, że znane są nie tylko wartości funkcji w węzłach interpolacyjnych, ale również wartości drugiej pochodnej w tych punktach można wyznaczyć interpolację funkcją sklepaną 3 stopnia bez efektu kantów.

Do dalszych rozważań przyjmujemy:

| $\mathbf{x_i}$ | $\mathbf{x_1}$ | $\mathbf{x_2}$ | $\mathbf{x_3}$ | ... | $\mathbf{x_n}$ |
|---------------------------|------------------|------------------|------------------|-----|------------------|
| $\mathbf{f_i=f(x_i)}$ | $\mathbf{f_1}$ | $\mathbf{f_2}$ | $\mathbf{f_3}$ | ... | $\mathbf{f_n}$ |
| $\mathbf{f_i''=f''(x_i)}$ | $\mathbf{f_1''}$ | $\mathbf{f_2''}$ | $\mathbf{f_3''}$ | ... | $\mathbf{f_n''}$ |

Interpolujące funkcje sklepane

Dla każdego przedziału interpolacyjnego zdefiniowanego przez $[x_i; x_{i+1}]$ dla $i=1, 2, \dots, n-1$ zapisujemy wielomian trzeciego stopnia:

$$y_i(x) = af_i + bf_{i+1} + cf'_i + df'_{i+1}$$

gdzie: $a = \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad b = \frac{x - x_i}{x_{i+1} - x_i}$

$$c = \frac{1}{6}(a^3 - a)(x_{i+1} - x_i)^2 \quad d = \frac{1}{6}(b^3 - b)(x_{i+1} - x_i)^2$$

Interpolujące funkcje sklepane

- Druga pochodna funkcji jest jednak nieznaną.
- Problem można rozwiązać zakładając ciągłość pierwszej pochodnej w węzłach
- Przyjmuje się, że pochodna lewym krańcu przedziału $y_i(x)$ jest równa pochodnej obliczanej dla prawego krańca przedziału $y_{i+1}(x)$

$$\begin{aligned} \frac{x_i - x_{i+1}}{6} f_{i-1}'' + \frac{x_{i+1} - x_{i-1}}{3} f_i'' + \frac{x_{i+1} - x_i}{6} f_{i+1}'' \\ = \frac{f_{i+1} - f_i}{x_{i+1} - x_i} - \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \end{aligned}$$

Interpolujące funkcje sklepane

- W efekcie dany jest układ równań liniowych z niewiadomymi $\{f_i''\}$.
- Problem stanowi to, że dla n węzłów interpolacji, danych jest $n-2$ węzłów wewnętrznych i w efekcie można zapisać $n-2$ równań dla n niewiadomych
- Konieczne jest więc przyjęcie dodatkowych dwóch warunków
- Najczęściej zakłada się, że $f_1''=f_2''=0$, ale można narzucić inne warunki brzegowe

Interpolujące funkcje sklepane

Przy założeniu, że odstęp między węzłami interpolacyjnymi są równe $\mathbf{x}_{i+1} - \mathbf{x}_i = h$ można zapisać równanie w postaci

$$\begin{bmatrix} 4 & 1 & 0 & \dots & 0 & 0 \\ 1 & 4 & 1 & \dots & 0 & 0 \\ 0 & 1 & 4 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} f_2'' \\ f_3'' \\ f_4'' \\ \vdots \\ f_{n-2}'' \\ f_{n-1}'' \end{bmatrix} = \begin{bmatrix} 6 \cdot \frac{f_1 - 2f_2 + f_3}{h^2} \\ 6 \cdot \frac{f_2 - 2f_3 + f_4}{h^2} \\ 6 \cdot \frac{f_3 - 2f_4 + f_5}{h^2} \\ \vdots \\ 6 \cdot \frac{f_{n-3} - 2f_{n-2} + f_{n-1}}{h^2} \\ 6 \cdot \frac{f_{n-2} - 2f_{n-1} + f_n}{h^2} \end{bmatrix}$$

Interpolujące funkcje sklepane

Algorytm zastosowania interpolacji funkcjami sklejanymi 3 stopnia składa się z dwóch etapów

1. Rozwiązania układu równań w celu wyznaczenia nieznanych wartości drugich pochodnych sygnału w przedziale $i = [2 ; n-1]$
2. Korzystając z rodziny wielomianów $y_i(x)$ określonych dla każdego przedziału $[x_i ; x_{i+1}]$, w zadanym przedziale można wyznaczyć dowolną liczbę punktów interpolujących

Interpolujące funkcje sklejane

```
function splajan(t,F,n)
... m=size(t,'r');
... step=t(2)-t(1);
... Z=4*eye(m-2,m-2);
... for i=1:m-2
...     for j=1:m-2
...         if j==i+1
...             Z(i,j)=1;
...         end
...         if i==j+1
...             Z(i,j)=1;
...         end
...     end
... end
... end
```

Interpolujące funkcje sklejane

```
Y=zeros(m-2,1);  
for i=1:m-2  
    Y(i)=(6/step^2)*(F(i)-2*F(i+1)+F(i+2));  
end  
F2=linsolve(Z,-Y);  
f=zeros(m,1);  
for i=2:m-1  
    f(i)=F2(i-1);  
end
```

Interpolujące funkcje sklejane

```
x=linspace(t(1),t(m),n);  
kj=[1];  
for i=2:n-1  
    kj(i)=max(find(t<x(i+1)) );  
end  
kj(n)=m-1;
```


Interpolujące funkcje sklepane

```

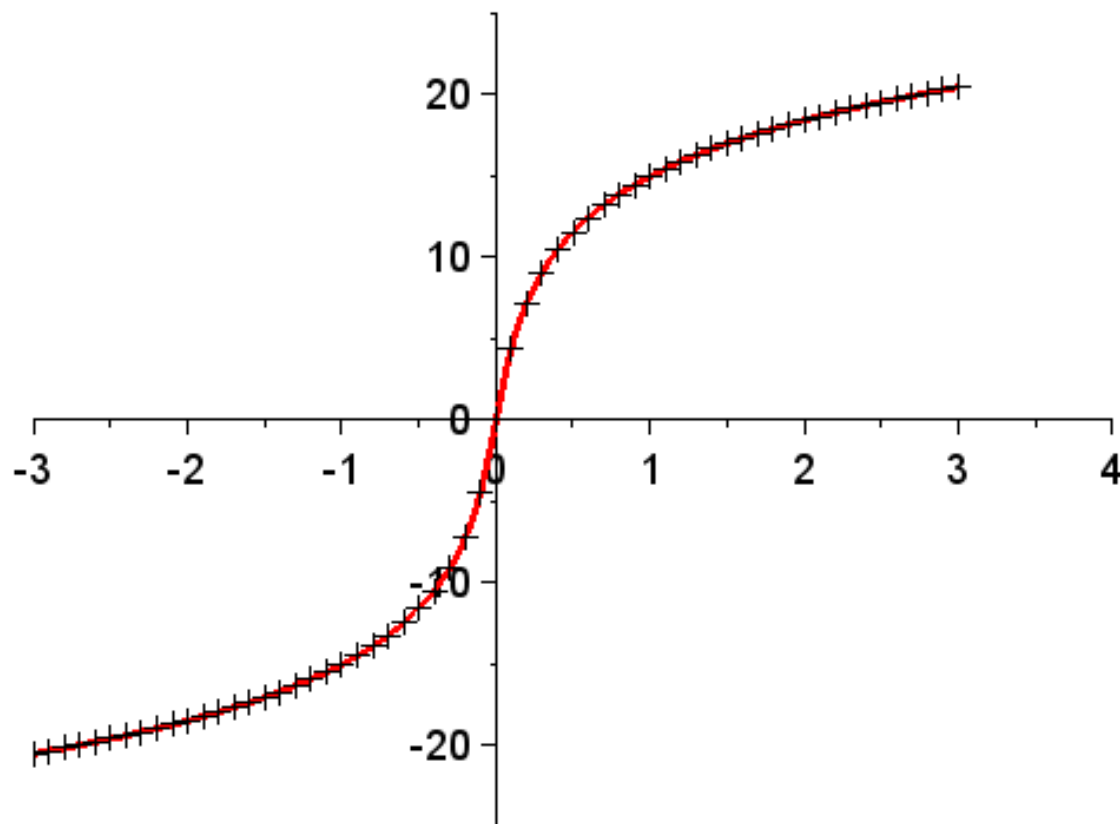
fun=[0];
for i=1:n
     $y_i(x) = af_i + bf_{i+1} + cf'_i + df'_{i+1}$ 
    k=kj(i);
    a=(t(k+1)-x(i))/(t(k+1)-t(k));
    b=(x(i)-t(k))/(t(k+1)-t(k));
    c=(1/6)*(a^3-a)*(t(k+1)-t(k))^2;
    d=(1/6)*(b^3-b)*(t(k+1)-t(k))^2;
    fun(i)=a*F(k)+b*F(k+1)+c*f(k)+d*f(k+1);
end

```

$$a = \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad b = \frac{x - x_i}{x_{i+1} - x_i}$$

$$c = \frac{1}{6}(a^3 - a)(x_{i+1} - x_i)^2 \quad d = \frac{1}{6}(b^3 - b)(x_{i+1} - x_i)^2$$

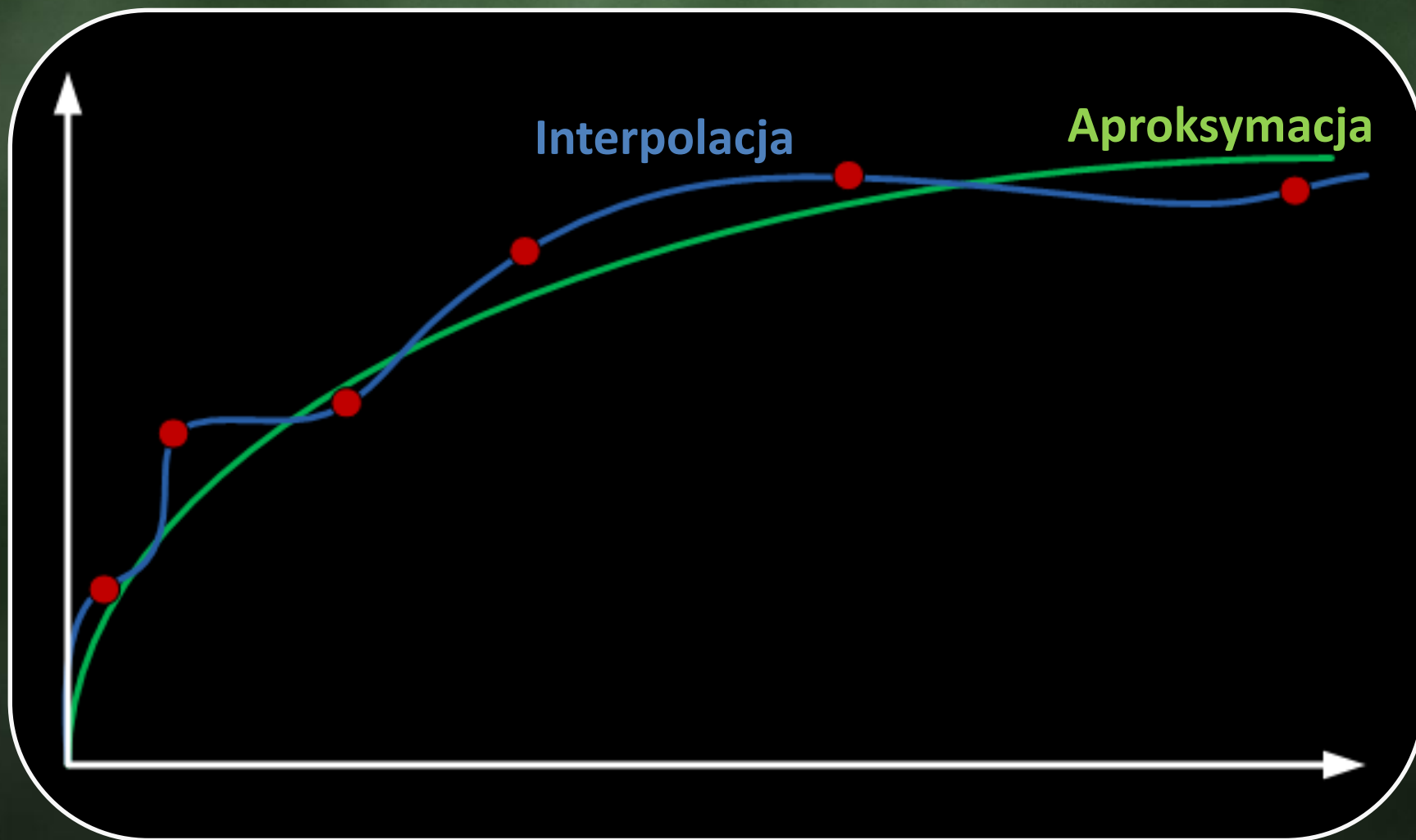
Przykład 8



Aproksymacja

- Aproksymacja polega na wyznaczeniu, możliwie niskim kosztem, funkcji F przybliżającej zadaną funkcję f z żadaną dokładnością
- W odróżnieniu od interpolacji funkcja aproksymująca nie musi przechodzić przez zadane punkty funkcji aproksymowanej.
- Funkcja aproksymująca $f(x)$ ma przybliżać funkcję aproksymowaną $F(x)$ z żadaną dokładnością.

Aproksymacja



Aproksymacja

Aproksymacja funkcji $f(x)$ polega na wyznaczeniu współczynników wielomianu $F(x)$

$$F(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \dots + a_n\varphi_n(x)$$

przy warunku osiągnięcia minimalnej wartości:

$$\|f(x) - F(x)\|$$

$\phi_0, \phi_1, \dots, \phi_n$ są funkcjami bazowymi, których właściwy dobór ma decydujące znaczenie dla prawidłowości i efektywności przeprowadzenia aproksymacji

Aproksymacja

Zadanie aproksymacyjne przy wybranych funkcjach bazowych $\phi_k(\mathbf{x})$ sprowadza się do wyznaczenia współczynników a_k , zapewniających osiągnięcie minimum normy:

$$\|f(x) - F(x)\|$$

Najczęściej stosowane normy to:

1. Norma jednostajna (Czebyszewa)
2. Normą średniokwadratową
3. Normą L_2

Wybór normy determinuje rodzaj aproksymacji

Aproksymacja

Funkcje bazowe $\phi(x)$ wybierane są zazwyczaj z podprzestrzenie:

a) Funkcji trygonometrycznych z bazą:

$$1, \sin(x), \cos(x), \sin(2x), \cos(2x), \dots, \sin(nx), \cos(nx)$$

Dla sygnałów okresowych

b) Wielomianów co najwyżej stopnia n :

$$1, x, x^2, x^3, \dots, x^n$$

Pomimo prostoty działań na wielomianach, cechują się dużymi błędami przy działaniach na małych lub niewiele różniących się liczbach

c) Wielomianów Czebyszewa

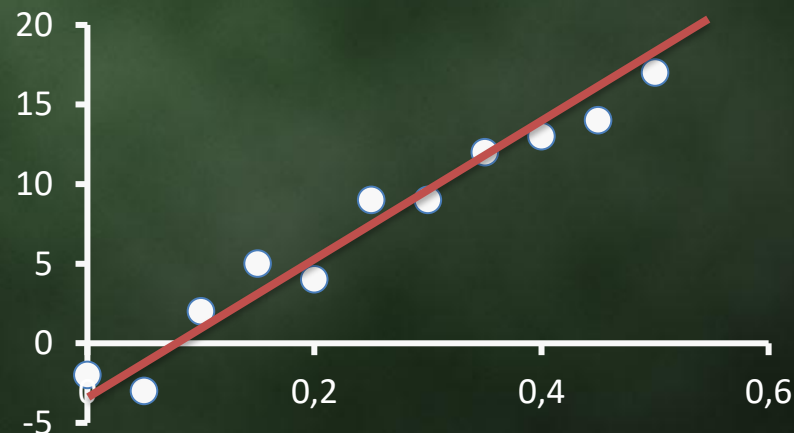
d) Wielomianów Legendre'a

Aproksymacja

- Na podstawie analizy danych pomiarowych można określić charakter – kształt funkcji wiążącej punkty pomiarowe.
- Aproksymacja średniokwadratowa znacznie lepiej „eliminuje” błędy pomiarowe od aproksymacji jednostajnej

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|----|----|---|---|---|---|---|----|----|----|----|
| y | -2 | -3 | 2 | 5 | 4 | 9 | 9 | 12 | 13 | 14 | 17 |

$$y = a_1 x + a_0$$



Aproksymacja

○ Zadanie sprowadza się do określenia współczynników a_0 i a_1 takich aby spełnione było najmniejsze maksimum między $F(x)$ i $f(x)$.

○ Aproksymacji średniokwadratowej

$$S(a_0, a_1) = \sum_{i=1}^n (y_i - (a_1 x_i + a_0))^2$$

Aproksymacja

- Określenie wartości współczynników a_0 i a_1 spełniających warunek osiągnięcia minimum funkcji sprowadza się do rozwiązania równania dla zerowania pochodnych równania.
- Zakładając aproksymację liniową opisaną równaniem $y=a_1x+a_0$ mówi się o regresji liniowej.

Aproksymacja

Poszukując minimum funkcji należy rozwiązać układ równań

$$\frac{\partial S(a_0, a_1)}{\partial a_0} = 2 \sum_{i=1}^n (y_i - a_0 - a_1 x_i)(-1) = 0$$

$$\frac{\partial S(a_0, a_1)}{\partial a_1} = 2 \sum_{i=1}^n (y_i - a_0 - a_1 x_i)(-x_i) = 0$$

Aproksymacja

Przekształcając wzory można równanie zapisać w postaci:

$$a_0 n + a_1 \sum_{i=1}^n x_i = \sum_{i=1}^n y_i$$

$$a_0 \sum_{i=1}^n x_i + a_1 \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i$$

Aproksymacja

W zapisie macierzowym układ równań przyjmuje postać:

$$2 \begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{bmatrix}$$

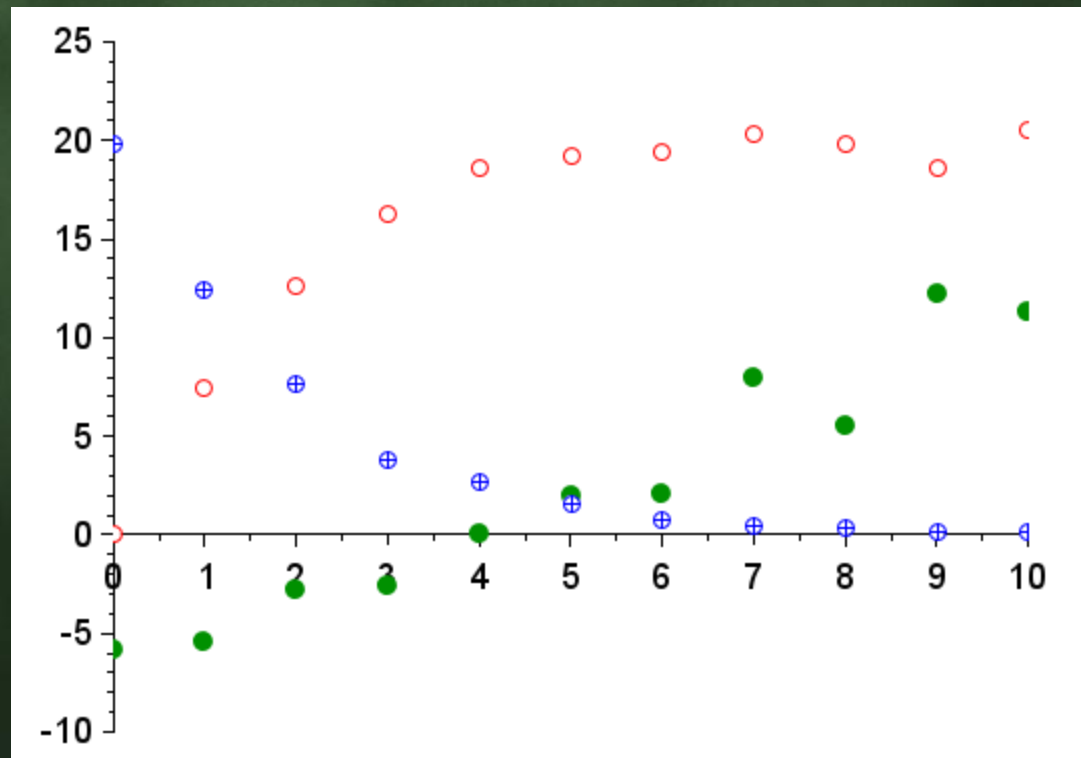
Aproksymacja

Ogólną postać dla wielomianu rzędu **k** można zapisać w postaci równania macierzowego:

$$(k+1) \begin{bmatrix} n & \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^1 & \dots & \sum_{i=1}^n x_i^k \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \dots & \sum_{i=1}^n x_i^{k+1} \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 & \dots & \sum_{i=1}^n x_i^{k+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^n x_i^k & \sum_{i=1}^n x_i^{k+1} & \sum_{i=1}^n x_i^{k+2} & \dots & \sum_{i=1}^n x_i^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n x_i^2 y_i \\ \vdots \\ \sum_{i=1}^n x_i^k y_i \end{bmatrix}$$

Przykład 11

Dla zadanych sygnałów pomiarowych przeprowadzić aproksymację wielomianową:



Przykład 11

- Każde zadanie aproksymacyjne rozwiązywane niezależnie.
- Przy aproksymacji krzywej nieliniowej można aproksymować do wielomianu n stopnia

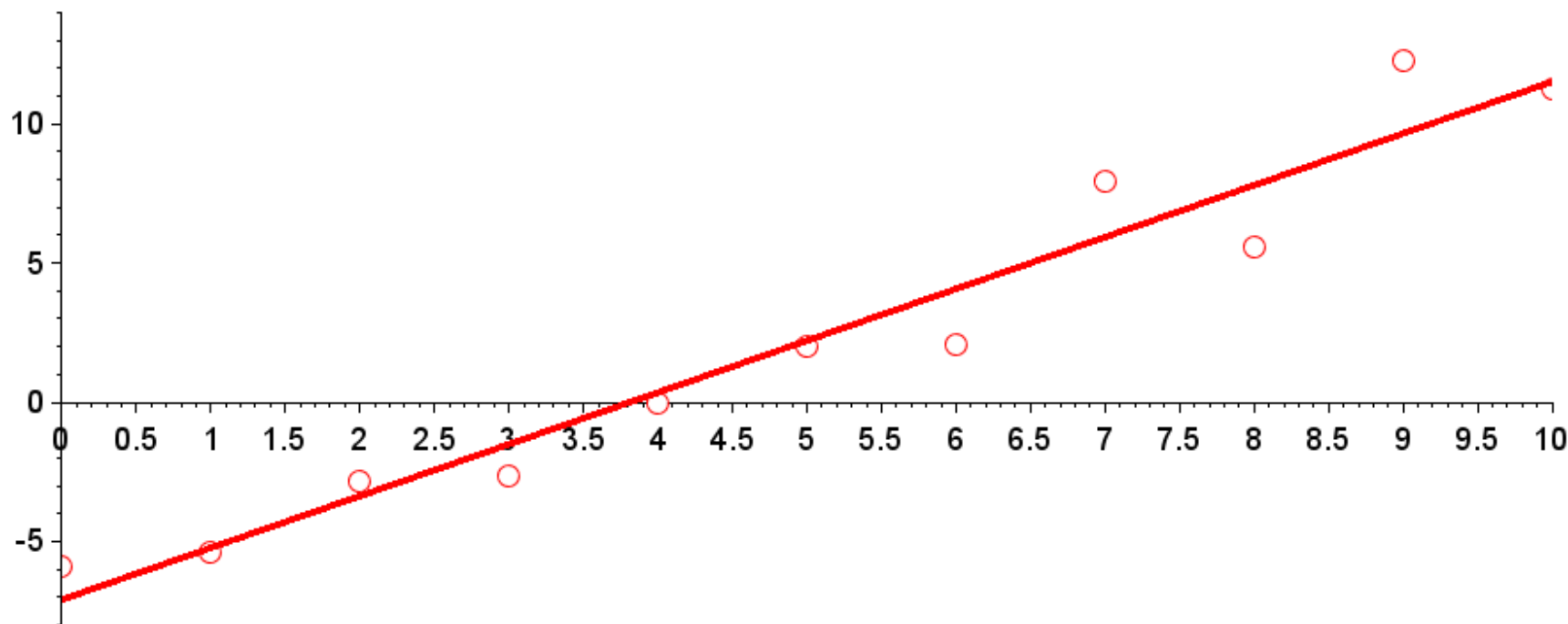
Przykład 11

Wyznaczanie współczynników wielomianu aproksymacyjnego

```
function liniowa(s,t,F,m)
    x=linspace(t(1),t(n),m)
    y=zeros(1,m);
    for i=1:m
        for j=1:s+1
            y(i)=y(i)+a(j)*x(i)^(j-1);
        end
    end
    plot2d(x,y,5);
    disp(a);
endfunction
a=linsolve(A,-B');
```

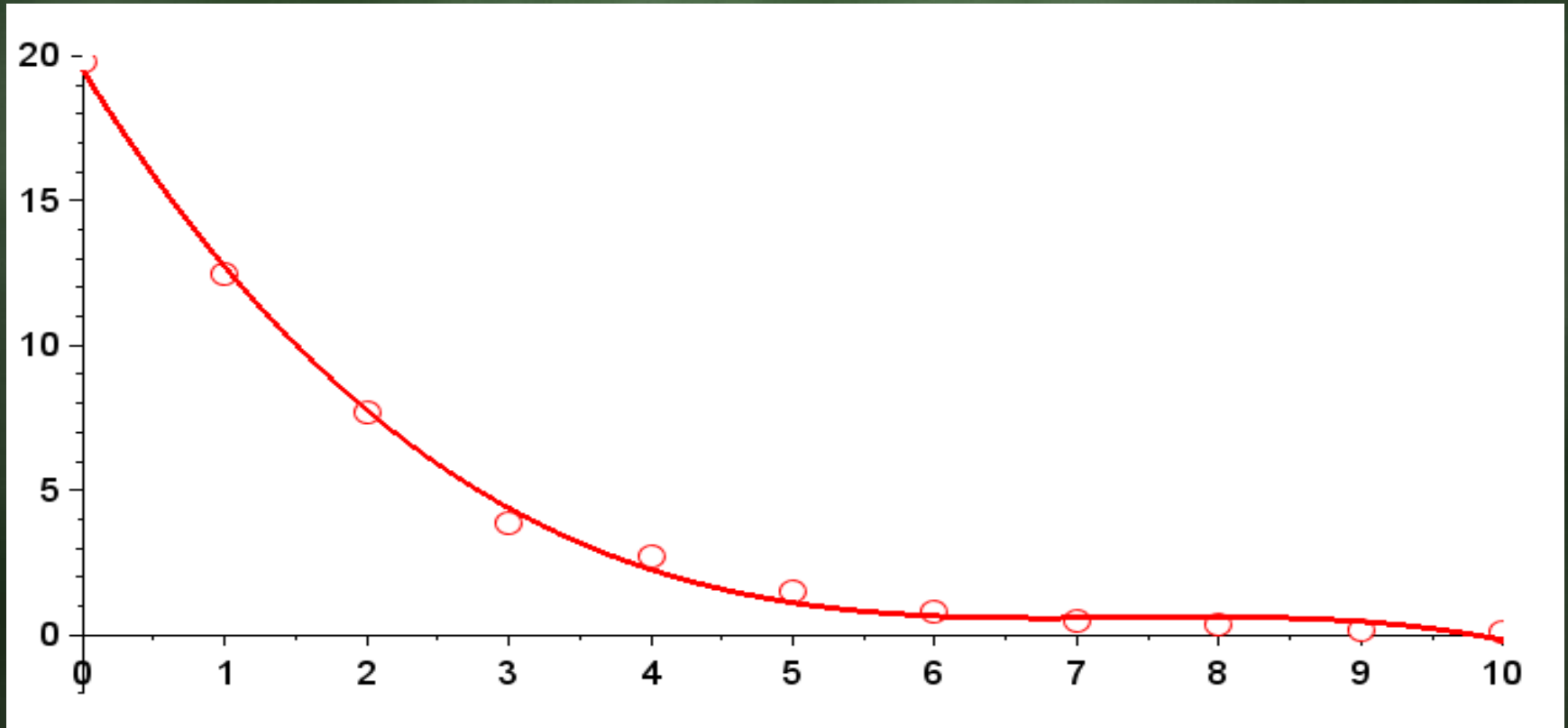
$$\begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{bmatrix}$$

Przykład 11



$$a_1 \approx 1.86 \quad a_0 \approx -7.1$$
$$f(t) = 1.86t - 7.1$$

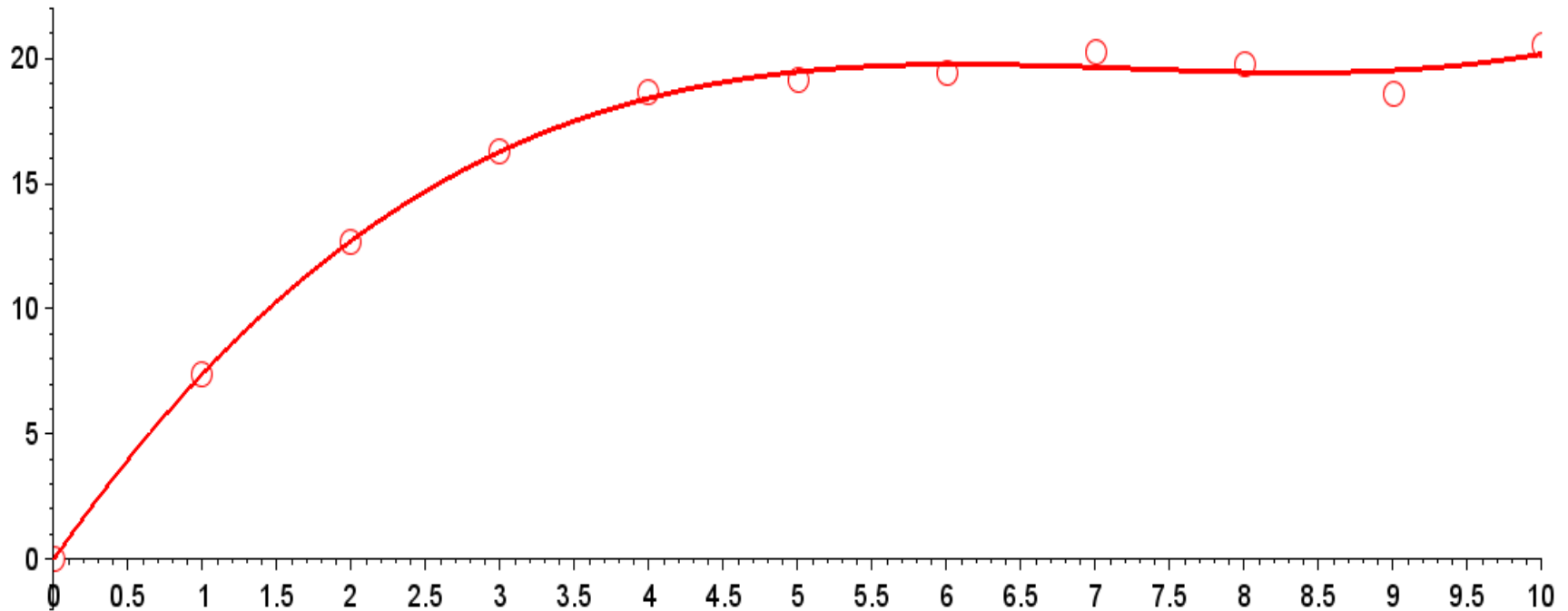
Przykład 11



$$a_0 \approx 19.53 \quad a_1 \approx -7.83 \quad a_2 \approx 1.07 \quad a_3 \approx -0.05$$

$$f(t) = -0.05t^3 + 1.07t^2 - 7.83t + 19.53$$

Przykład 11



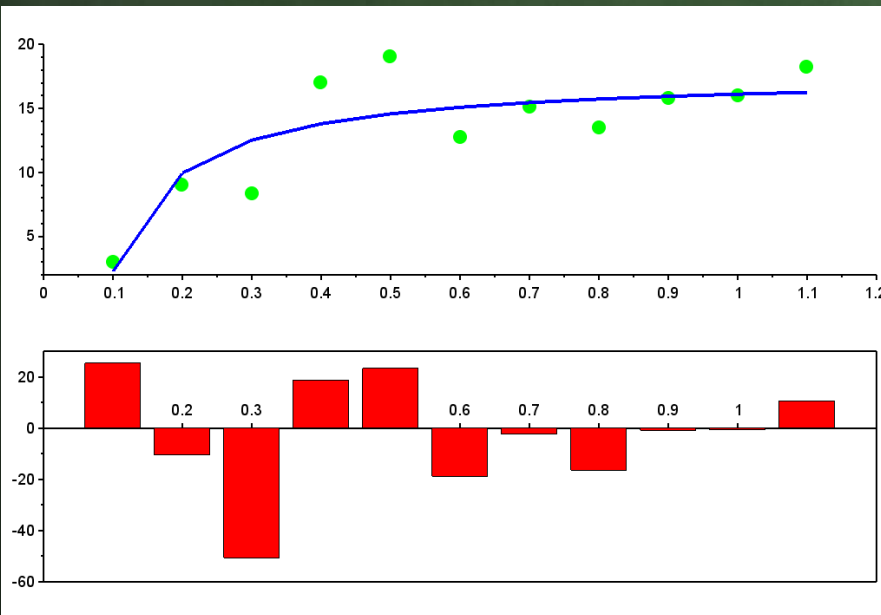
$$a_0 \approx -0.009 \quad a_1 \approx 8.57 \quad a_2 \approx -1.22 \quad a_3 \approx 0.06$$

$$f(t) = 0.06t^3 - 1.22t^2 + 8.57t - 0.009$$

Aproksymacja

W wielu przypadkach można problem nieliniowy sprowadzić do zadania liniowego, stosując odpowiednie podstawienia i przekształcenia.

$$y = \frac{a_1}{x} + a_0 = a_1 \tilde{x} + a_0 \rightarrow \tilde{x} = \frac{1}{x}$$



$$y = -\frac{1.54}{x} + 17.65$$

Aproksymacja

$$y = a_0 e^{a_1 x}$$

$$\ln(y) = \ln(a_0) + \ln(e^{a_1 x})$$

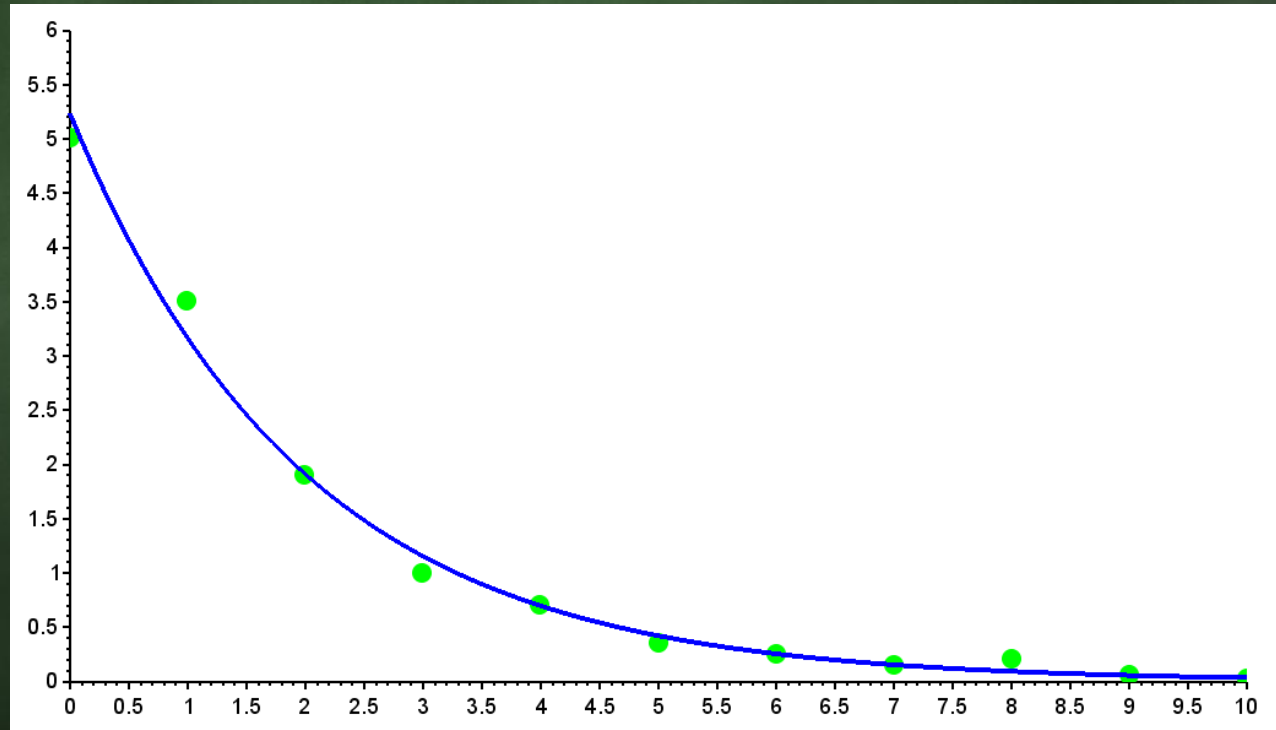
$$\ln(y) = \ln(a_0) + a_1 x$$

$$\ln(y) = Y$$

$$\ln(a_0) = A$$

$$Y = A + a_1 x$$

$$y \approx 5.23e^{-0.5x}$$



Aproksymacja

Dane pomiarowe aproksymuje się do funkcji najbardziej zbliżonej kształtem.

$$y = a_2 x^{a_1} + a_0$$

$$y = e^{a_2 x^2 + a_1 x + a_0}$$

$$y = a_2 e^{a_1 x} + a_0$$

$$y = a_2 x^2 + a_1 x + a_0$$

$$y = \sqrt{a_2 x^2 + a_1 x + a_0}$$

$$y = \frac{a_3 x + a_2}{a_1 x + a_0}$$

$$y = \frac{1}{a_2 x^2 + a_1 x + a_0}$$

$$y = \frac{x}{a_2 x^2 + a_1 x + a_0}$$

$$y = a_2 + \frac{a_1}{x} + \frac{a_0}{x^2}$$