

1.

a)

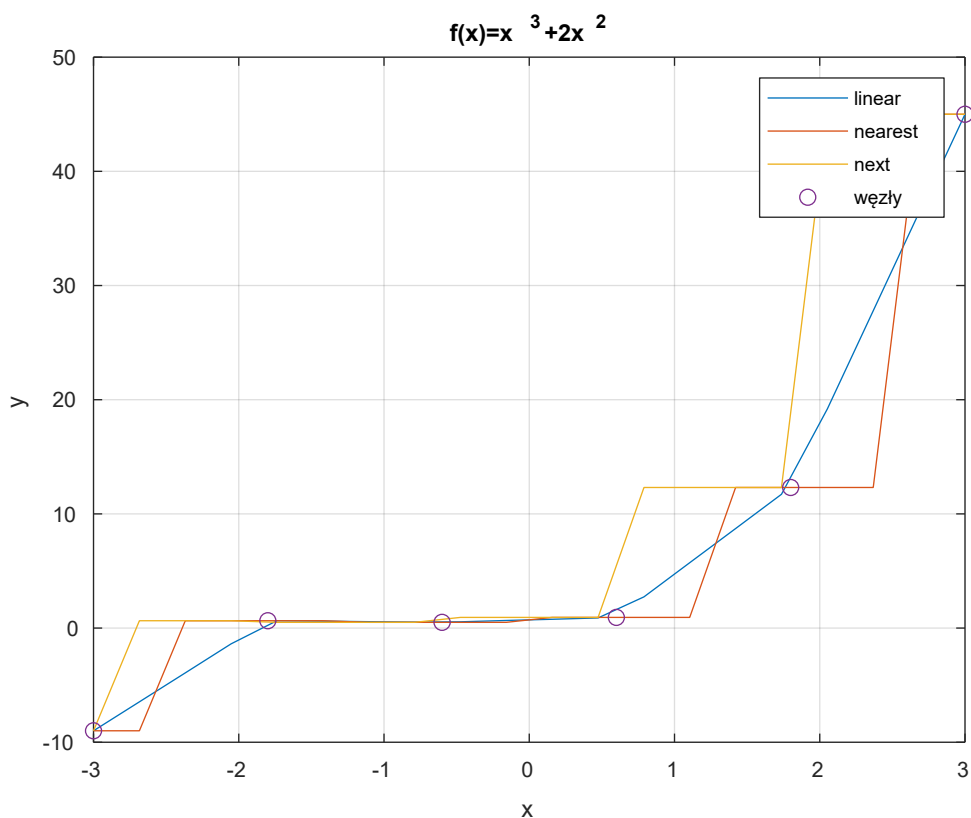
Wykonujemy interpolację 3 wbudowanymi funkcjami matlaba:

linear -przyporządkowuje interpolowany punkt do odpowiedniego wielomianu liniowego pomiędzy każdym znanym węzłem danych

nearest -przyporządkowuje interpolowany punkt do najbliższego węzła danych

next -przyporządkowuje interpolowany punkt do kolejnego węzła danych

```
zad1a.m x zad1b.m x zad2.m x zad3.m x +
2 clear all
3 close all
4
5 x=linspace(-3,3,6);
6 y=x.^3+2*x.^2;
7
8 xi=linspace(-3,3,20);
9 disp('czas interpolacji metodą linear:');
10 tic
11 yi1=interp1(x,y,xi);
12 toc
13 fprintf('\nczas interpolacji metodą nearest:\n');
14 tic
15 yi2=interp1(x,y,xi,'nearest');
16 toc
17 fprintf('\nczas interpolacji metodą next:\n');
18 tic
19 yi3=interp1(x,y,xi,'next');
20 toc
21
22 plot(xi,yi1);
23 title('f(x)=x^3+2x^2');
24 xlabel('x');
25 ylabel('y');
26 hold on
27 plot(xi,yi2);
28 plot(xi,yi3);
29 grid on;
30 plot(x,y,'o');
31 legend('linear','nearest','next','węzły');
```



Jak widać na grafice, metoda linear przybliża najdokładniej do pożądanej funkcji. Jednak jest to okupione zwiększonymi zasobami i czasem trwania procesu. Funkcja wykonuje się wolniej o cały rząd wielkości.

```
czas interpolacji metodą linear:
Elapsed time is 0.011144 seconds.

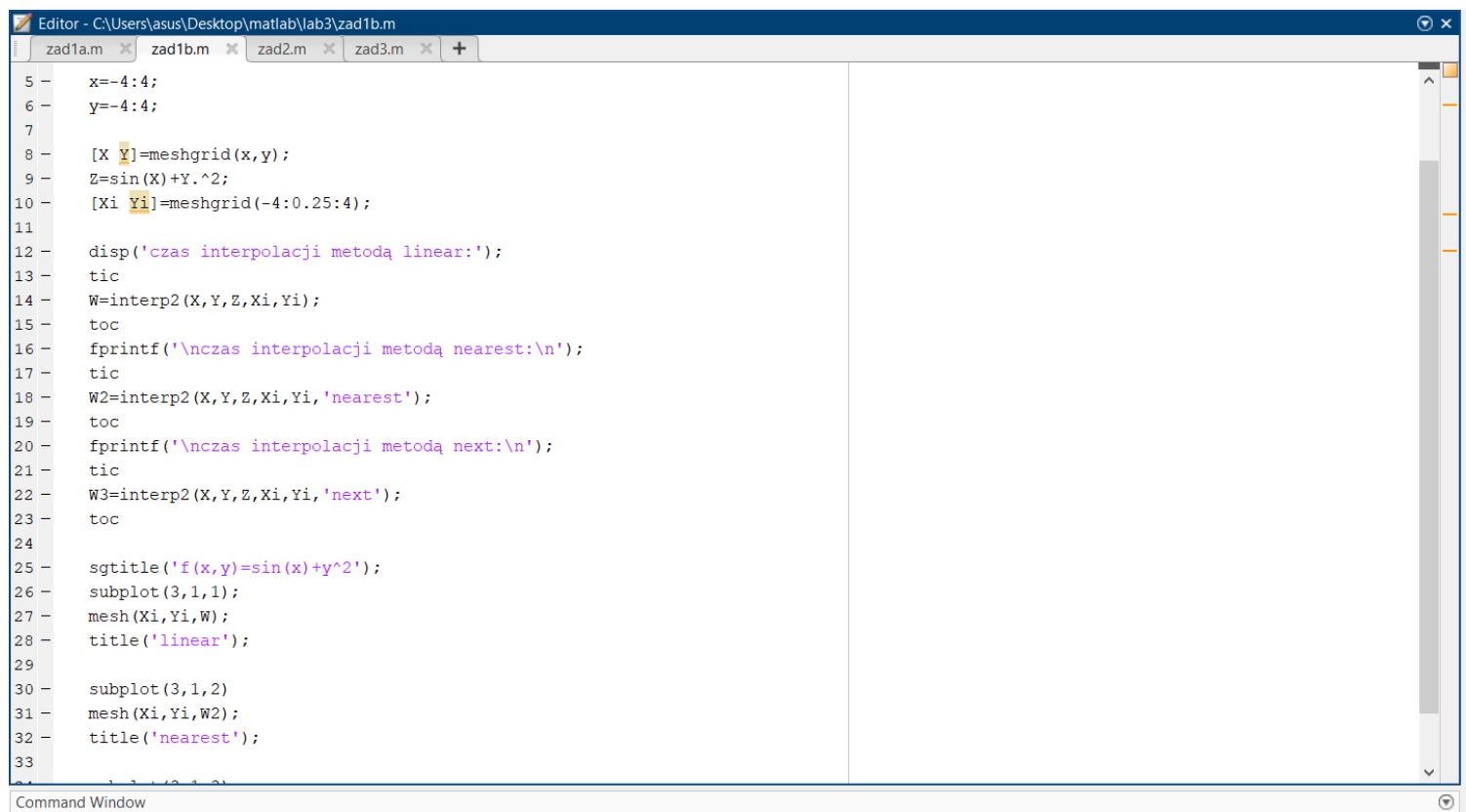
czas interpolacji metodą nearest:
Elapsed time is 0.001688 seconds.

czas interpolacji metodą next:
Elapsed time is 0.001551 seconds.
>>
```

Metody nearest i next są szybsze jednak przybliżenie jest bardzo niedokładne. Metoda nie tworzy żadnych nowych punktów danych, jedynie przybliża wygląd funkcji.

b)

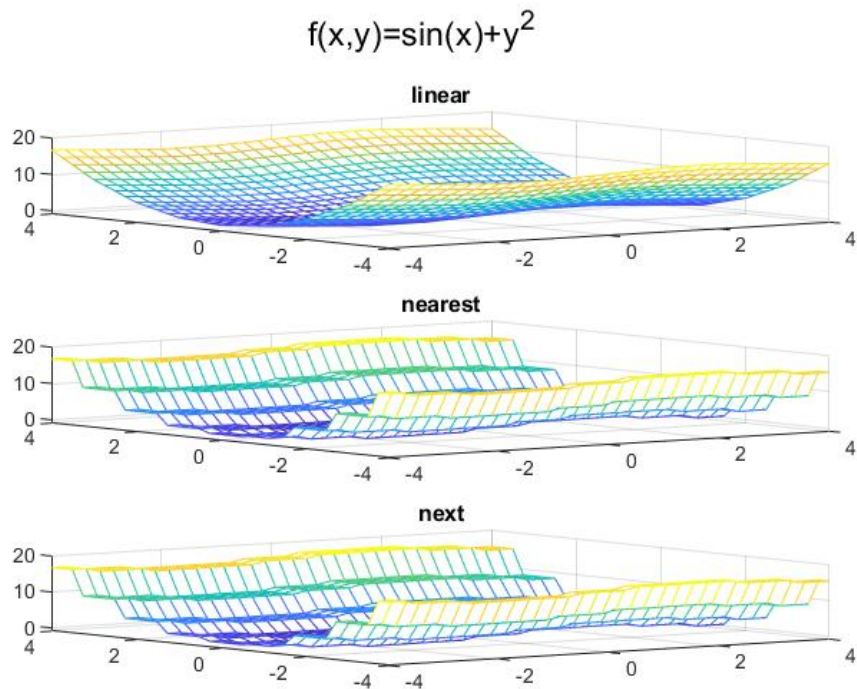
Wykonujemy to samo dla funkcji 2 zmiennych



```
Editor - C:\Users\asus\Desktop\matlab\lab3\zad1b.m
zad1a.m  zad1b.m  zad2.m  zad3.m  +
5  x=-4:4;
6  y=-4:4;
7
8  [X Y]=meshgrid(x,y);
9  Z=sin(X)+Y.^2;
10 [Xi Yi]=meshgrid(-4:0.25:4);
11
12 disp('czas interpolacji metodą linear:');
13 tic
14 W=interp2(X,Y,Z,Xi,Yi);
15 toc
16 fprintf('\nczas interpolacji metodą nearest:\n');
17 tic
18 W2=interp2(X,Y,Z,Xi,Yi,'nearest');
19 toc
20 fprintf('\nczas interpolacji metodą next:\n');
21 tic
22 W3=interp2(X,Y,Z,Xi,Yi,'next');
23 toc
24
25 sgtitle('f(x,y)=sin(x)+y^2');
26 subplot(3,1,1);
27 mesh(Xi,Yi,W);
28 title('linear');
29
30 subplot(3,1,2);
31 mesh(Xi,Yi,W2);
32 title('nearest');
33
```

Command Window

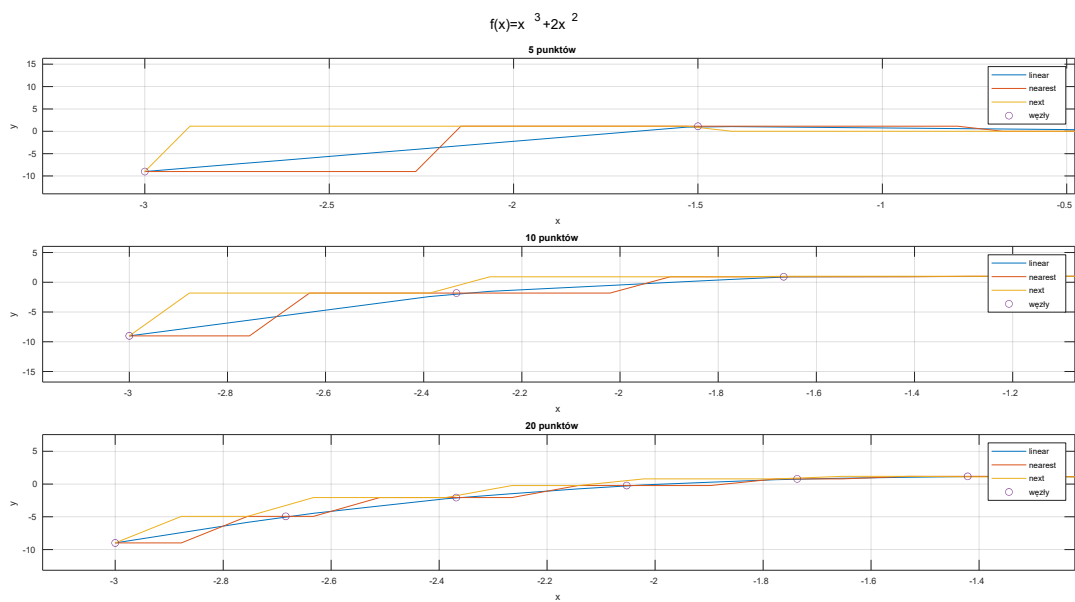
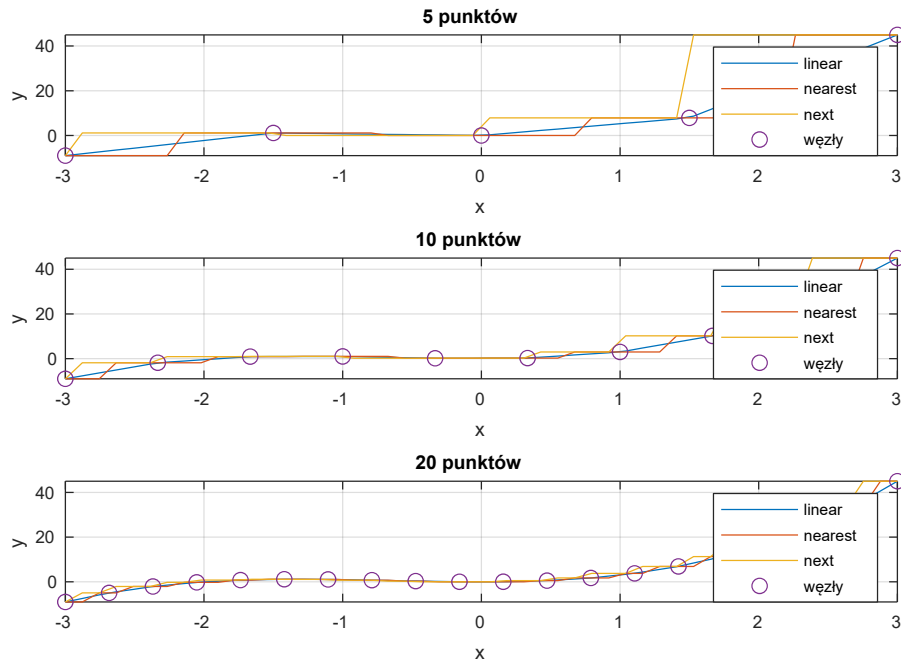
Podobnie jak dla funkcji 1 zmiennej metoda linear jest najdokładniejsza i najwolniejsza.



2. Będziemy porównywać interpolację dla różnej ilości węzłów danych

```
5 - x=linspace(-3,3,5);
6 - y=x.^3+2*x.^2;
7
8 - xi=linspace(-3,3,50);
9 - y1=interp1(x,y,xi);
10 - y12=interp1(x,y,xi,'nearest');
11 - y13=interp1(x,y,xi,'next');
12
13 - sgtitle('f(x)=x^3+2x^2');
14
15 - subplot(3,1,1);
16 - plot(xi,y1);
17 - title('5 punktów');
18 - xlabel('x');
19 - ylabel('y');
20 - hold on
21 - plot(xi,y12);
22 - plot(xi,y13);
23 - grid on;
24 - plot(x,y,'o');
25 - legend('linear','nearest','next','węzły');
26
27 - x=linspace(-3,3,10);
28 - y=x.^3+2*x.^2;
29
30 - xi=linspace(-3,3,50);
31 - y1=interp1(x,y,xi);
32 - y12=interp1(x,y,xi,'nearest');
33 - y13=interp1(x,y,xi,'next');
34
35 - subplot(3,1,2);
36 - plot(xi,y1);
37 - title('10 punktów');
38 - xlabel('x');
39 - ylabel('y');
40 - hold on
41 - plot(xi,y12);
42 - plot(xi,y13);
```

$$f(x)=x^3+2x^2$$

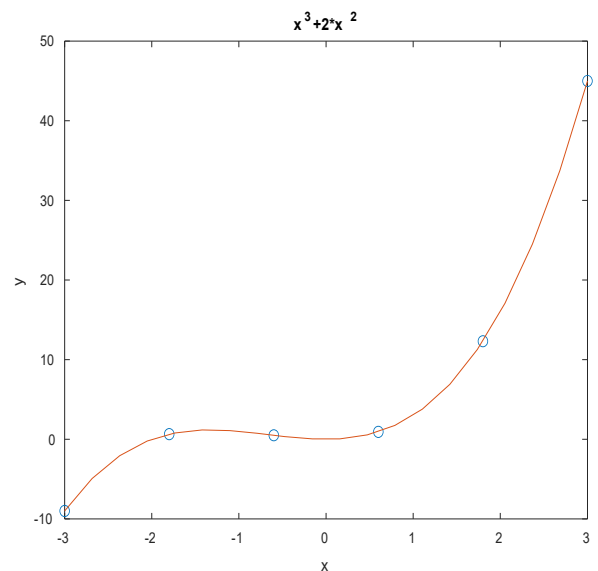


Jak widać zwiększenie ilości węzłów powoduje lepszą aproksymację interpolacji. Jednak jak widzimy na lineara nie wpływa to aż tak bardzo, a na nearest i next znacząco, ponieważ te metody opierają się na ilości węzłów danych. Przy ilości węzłów zmierzającej do nieskończoności metoda nearest i next odwzorowuje funkcję w sposób idealny.

3.

algorytm lagrange'a dla tych samych danych co w zadaniu 1

```
1 - clear all;
2 - clc;
3 - close all;
4
5 - x=linspace(-3,3,6);
6 - y=x.^3+2*x.^2;
7 - xi=linspace(-3,3,20);
8
9 - disp('czas trwania naszego algorytmu interpolacji:');
10 - tic
11 - yi = lagrange(x,y,xi);
12 - toc
13 - plot(x,y,'o',xi,yi);
14 - title('x^3+2*x^2');
15 - xlabel('x');
16 - ylabel('y');
17
18 - function yi = lagrange(x,y,xi)
19 -     yi = zeros(size(xi));
20 -     for i = 1:length(x)
21 -         u = ones(size(xi));
22 -         for j = [1:i-1 i+1:length(x)]
23 -             u = (xi-x(j))./(x(i)-x(j)).*u;
24 -         end
25 -         yi = yi + u*y(i);
26 -     end
27 - end
```



```
czas trwania naszego algorytmu interpolacji:
Elapsed time is 0.005074 seconds.
fx >>
```

Nasz algorytm jest dość szybki, szybszy od metody linear ale wolniejszy od nearest i next, przy tym charakteryzuje się podobnym stopniem aproksymacji co metoda linear.