Sprawozdanie

Marcin Ścisłowski, Tomasz Tkaczyk 23.01.2022

1. Opis teoretyczny zagadnienia

Celem projektu, było usprawnienie działania programu wyznaczającego aproksymację danych pomiarowych zapisanych w pliku o postaci:

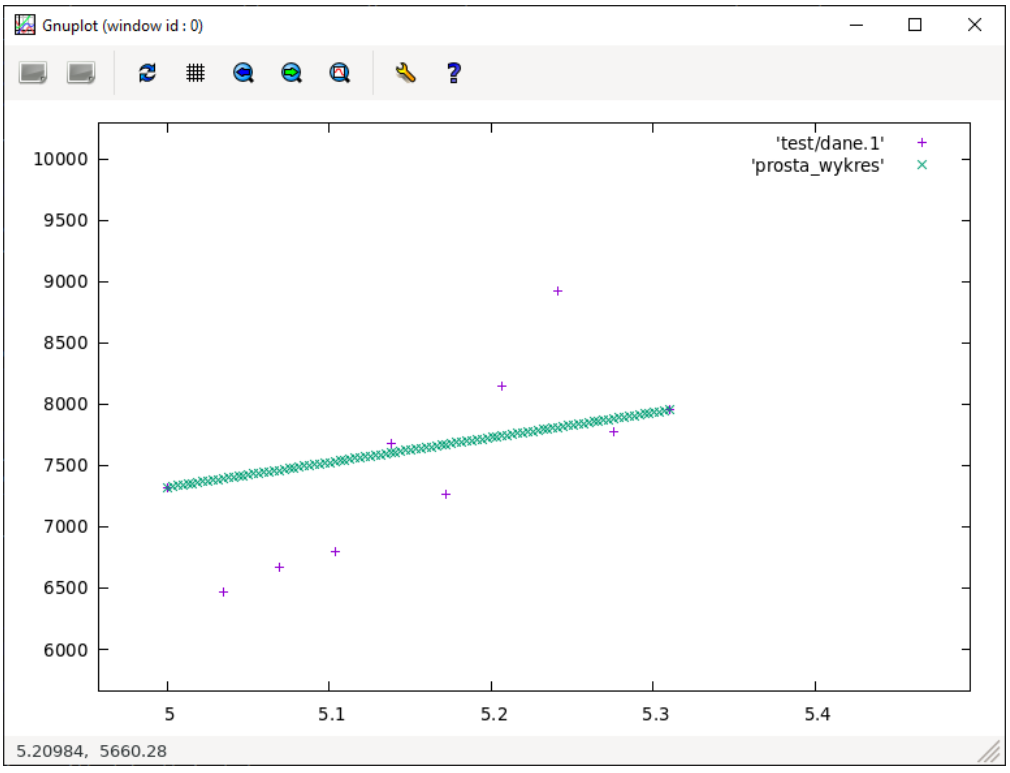
x_0	y_0
x_1	y_1
.	.
x_{n-1}	y_{n-1}

Do wyznaczenia aproksymacji, można używać różnego rodzaju funkcji aproksymujących. Najprostszą funckją aproksymującą, która dobrze ilustruje istote problemu będzie funkcja liniowa wyznaczona przez 2 skrajne punkty. Dla zestawu danych:

```
1   5 7321.46
2   5.03448 6462.77
3   5.06897 6671.45
4   5.10345 6799.14
5   5.13793 7678.74
6   5.17241 7267.03
7   5.2069 8149.58
8   5.24138 8925.87
9   5.27586 7778.01
10  5.31034 7951.56
```

Utworzymy (domyślnie) 100 puntków korzystajac z funckji liniowej o współczynniku a= $\frac{y_9}{y_0}$.

Aproksymacja korzystająca z funkcji liniowej będzie miała następującą reprezentację graficzną:



Zadaniem naszego zespołu było zaimplementowanie do programu funckji, która korzystając z aproksymacji średnio kwadratowej, aproksymowałaby nasze dane, w oparciu o bazę wielomianów Czebyszewa.

W naszym przypadku bazę, będą stanowić wielomiany Czebyszewa, które mają następującą postać:

$T_0(x)= 1$

$T_1(x)=x$

$T_k(x)=2x * T_{k-1}(x) - T_{k-2}(x)$

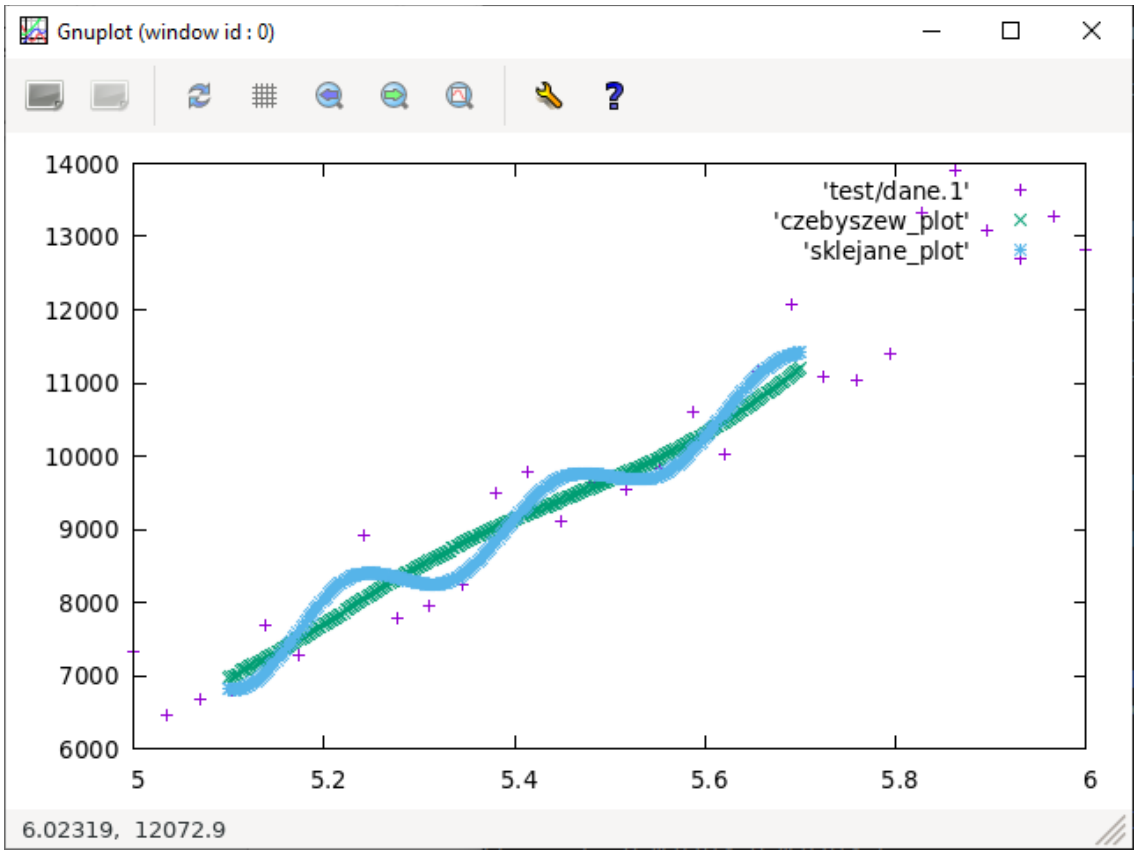
O implementacji tego zagadnienia w języku C napisaliśmy więcej w punkcie 3.

2. Wywoływanie programu

Nasz program dla przykładowych danych: test/dane.1 można wywołać za pomocą napisanego w bashu skyptu o nazwie „comp_and_run". To wywołanie spowoduje uwtorzenie plików „sklejane_plot", „czebyszew_plot", „sklejane_spl" oraz „czebyszew_spl". Po wywołaniu pliku ./comp_and_run program skompiluje wszystkie niezbedne pliki oraz za pomoca gnuplota przedstawi graficznie dane uzyskane w plikach wynikowych: „czebyszew_plot" oraz „sklejane_plot".

```
ehpop@DESKTOP-M456RNS:~/testy/PROJEKT/Projekt/lmp10$ ./comp_and_run.sh
cc     -c -o matrix.o matrix.c
cc     -c -o pivot.o pivot.c
cc     -c -o piv_ge_solver.o piv_ge_solver.c
ar rvs libge.a matrix.o pivot.o piv_ge_solver.o
r - matrix.o
r - pivot.o
r - piv_ge_solver.o
cc     -c -o main.o main.c
cc     -c -o splines.o splines.c
cc     -c -o points.o points.c
cc     -c -o baza_wielomianow_czebyszewa.o baza_wielomianow_czebyszewa.c
cc -Wall -Wextra -pedantic -I gaus -c aproksymator_na_bazie.c
cc     -c -o wielomiany.o wielomiany.c
cc -g -Wall -Wextra -pedantic -o czebyszew main.o splines.o points.o baza_wielomianow_czebyszewa.o aproksymator_na_bazie.o wielomiany.o
cc     -c -o main.o main.c
cc     -c -o splines.o splines.c
cc     -c -o points.o points.c
cc     -c -o baza_funkcji_sklejanych.o baza_funkcji_sklejanych.c
cc -Wall -Wextra -pedantic -I gaus -c aproksymator_na_bazie.c
cc -Wall -Wextra -pedantic -o sklejane  main.o splines.o points.o baza_funkcji_sklejanych.o aproksymator_na_bazie.o -L gaus -l ge -lm
```

Wywołanie skryptu w folderze z plikami



Graficzna interpretacja plików czebyszew_plot oraz sklejane_plot

W skrypcie generowane wykresy mają przykładowe argumenty -f = 5.1 -t = 5.7 oraz -n = 300.

Plik z danymi sklejane_plot tworzony jest poprzez aproksymację średniokwadratową w oparciu o bazę funkcji sklejanych, która została już zaimplementowana w programie. Program można również kompilować korzystając z makefile. Aby uzyskać plik wykonywalny ./czebyszew należy urchomić program make z argumentem „czebyszew”.

```
ehpop@DESKTOP-M456RNS:~/testy/PROJEKT/Projekt/lmp10$ make czebyszew
cc -g -Wall -Wextra -pedantic -o czebyszew main.o splines.o points.o baza_wielomianow_czebyszewa.o aproksymator_na_bazie.o wielomiany.o
ehpop@DESKTOP-M456RNS:~/testy/PROJEKT/Projekt/lmp10$
```

Plik wywołujemy w następujący sposób:

./czebyszew -s <plik_spline> [-p <points_file> [-g <gnuplot_file>[-n -f -t]]]

W przypadku nie podania argumentow z flagą -g program wypisze wynik na stdout, n oznaczajace ilosc punktów zostanie ustawione na 100, a punktami początkowymi i końcowymi będą punkty z pliku points o indeksach 0 i n-1.

3. Implementacja założeń projektowych

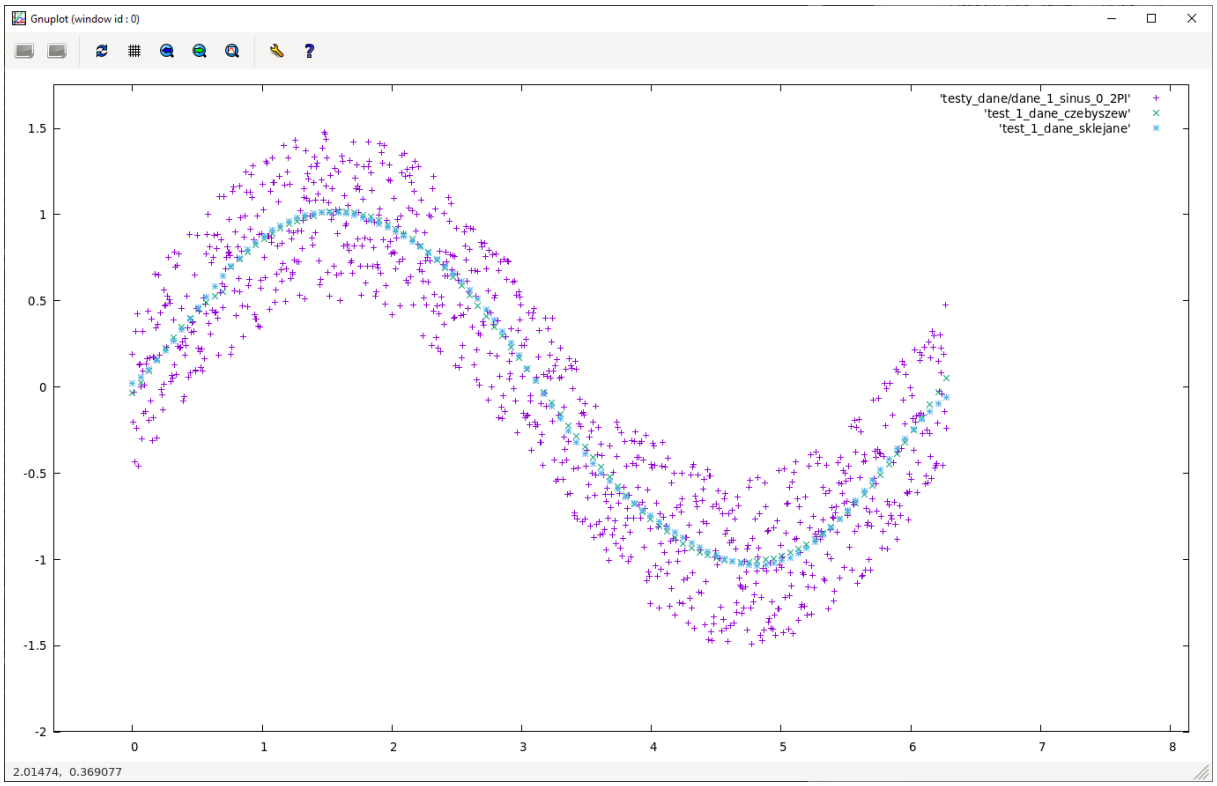
Naszym zadaniem było rozszerzenie programu o aproksymacje śr. Kw. na bazie wielomianów Czebyszewa. Posłużyliśmy się w tym celu dostarczoną nam z programem aproksymacją śr. Kw. Na bazie funkcji sklejanych. Rozdzieliliśmy ją na 2 komponenty, aproksymator śr. Kw. Na bazie i bazę funkcji sklejanych, w ten sposób rozszerzenie programu o jakąkolwiek bazę aproksymacji śr. Kw. Ograniczyło się do napisania funkcji bazy i jej 3 pochodnych, oraz stworzeniu reguły makefile. Wielomiany Czebyszewa dane są rekurencyjnym wzorem T_0(x)=1, T_1(x)=x, T_x(x)=2xT_(k-1)(x)-T_(k-2)(x), postanowiliśmy więc napisać strukturę reprezentującą wielomian w dziedzinie liczb rzeczywistych, i kilka funkcji do operacji na wielomianach.

4. Testy programu dla różnych danych wejsciowych oraz powrównanie z aproksymacją średnio-kwadratową w oparciu o bazę funkcji sklejanych

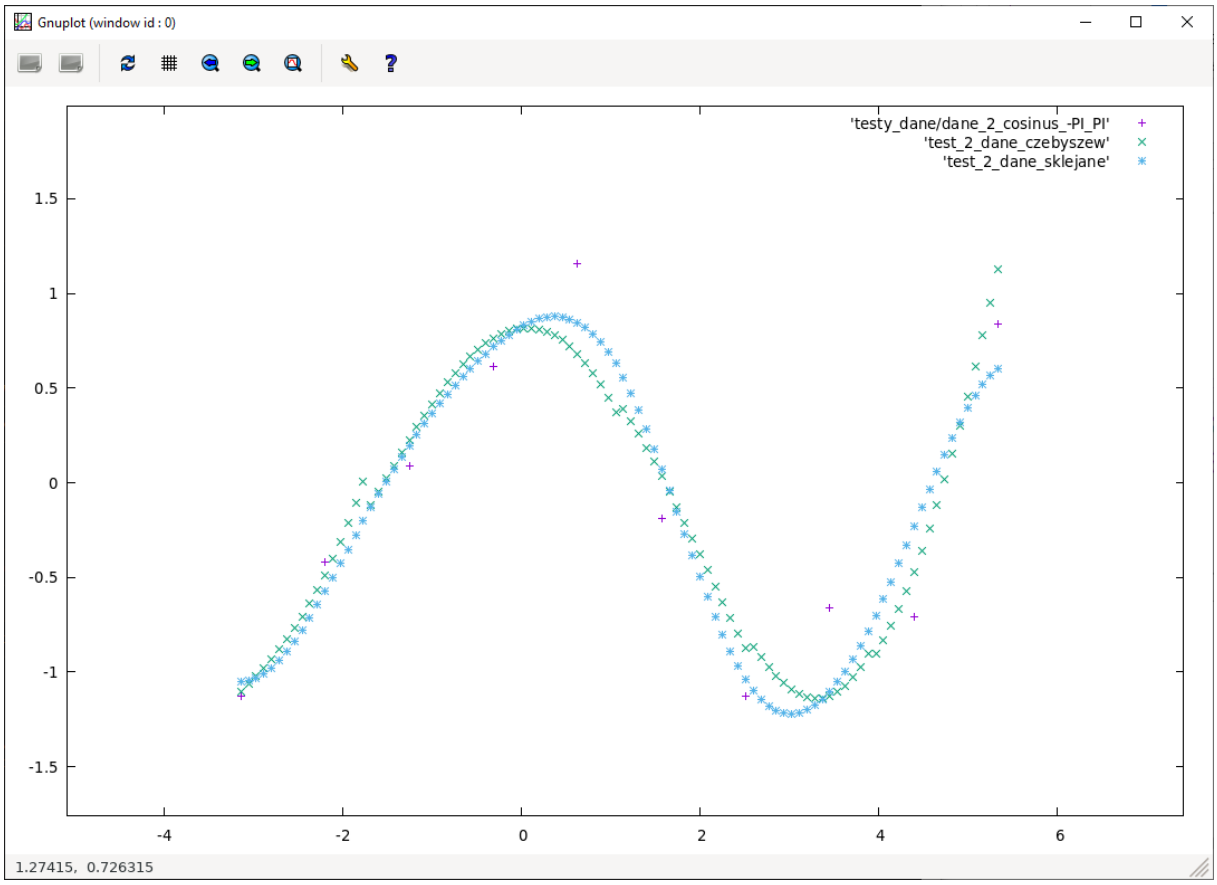
Do generowania danych testowych dla naszego programu użyliśmy własnego programu napisanego w języku c, o nazwie gen_mat(kod do tego programu jest załączony w projekcie). Program generuje n punktów w przedziale <a , b> za pomocą 1 z 6 dostępnych funkcji: sinus, cosinus, tangens, cotangens, arctan, prosta (a = 0,56), prosta (a = -2). Dodatkowo do każdego punktu dodawana jest losowa wartość z przedziału $<-\frac{1}{2}, \frac{49}{100}>$.

Przykładowe aproksymacje

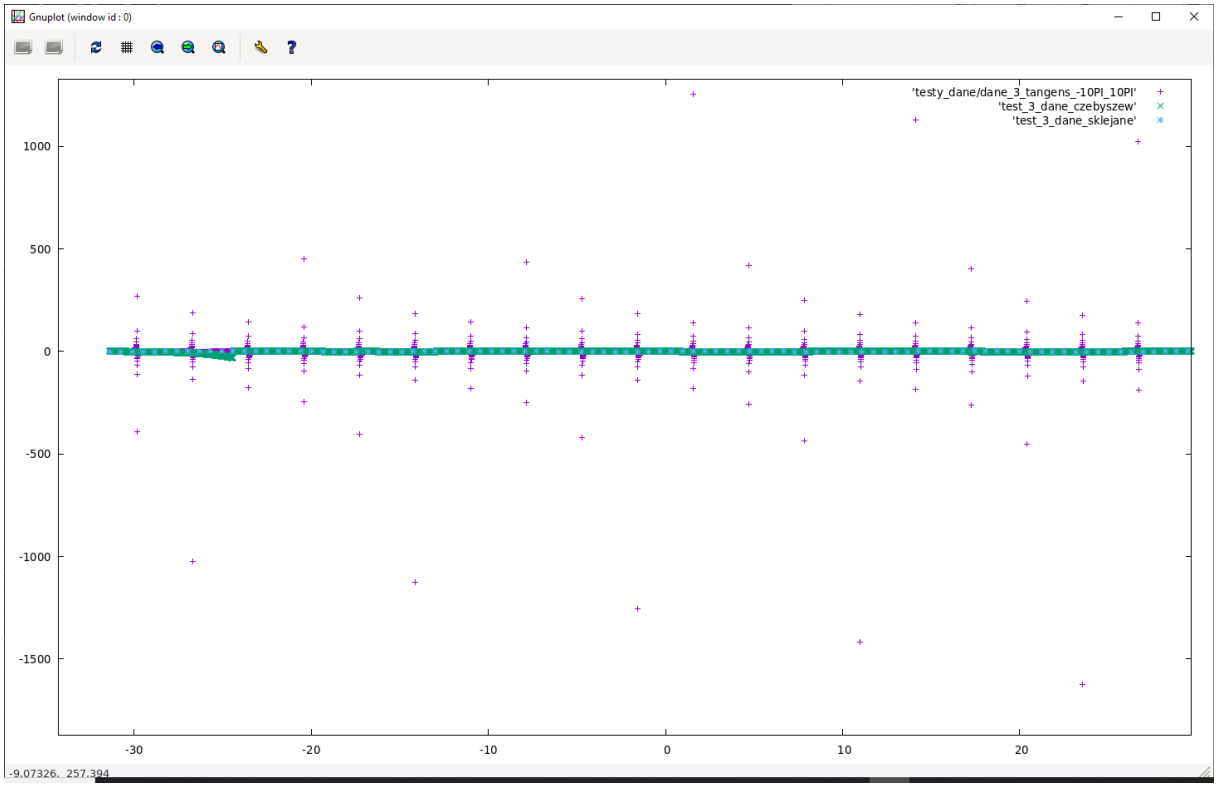
a) Plik z danymi zawiera zawiera 1000 punktów wygenrowane przez nasz program w zakresie <0;2π> bazowanych na funkcji sinus.



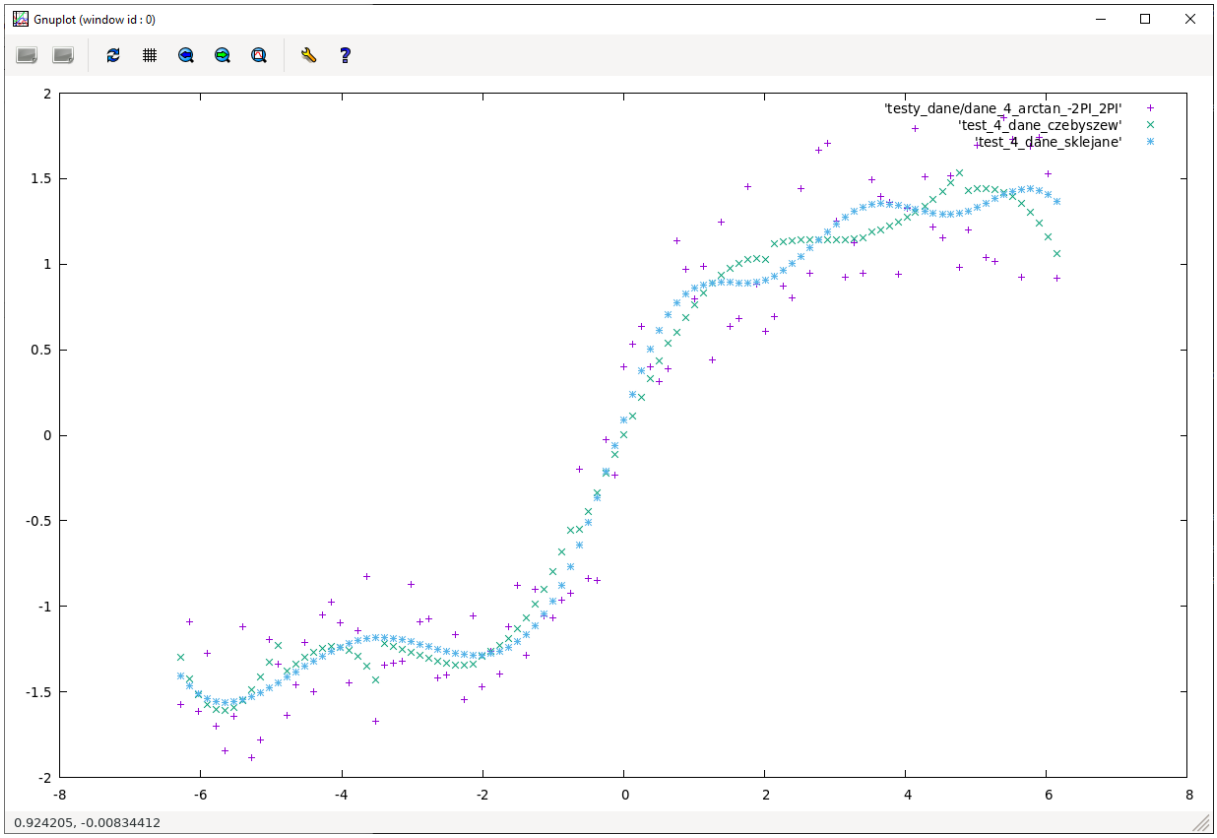
b) Plik z danymi zawiera zawiera 10 punktów wygenrowane przez nasz program w zakresie <-π; π> bazowanych na funkcji cosinus.



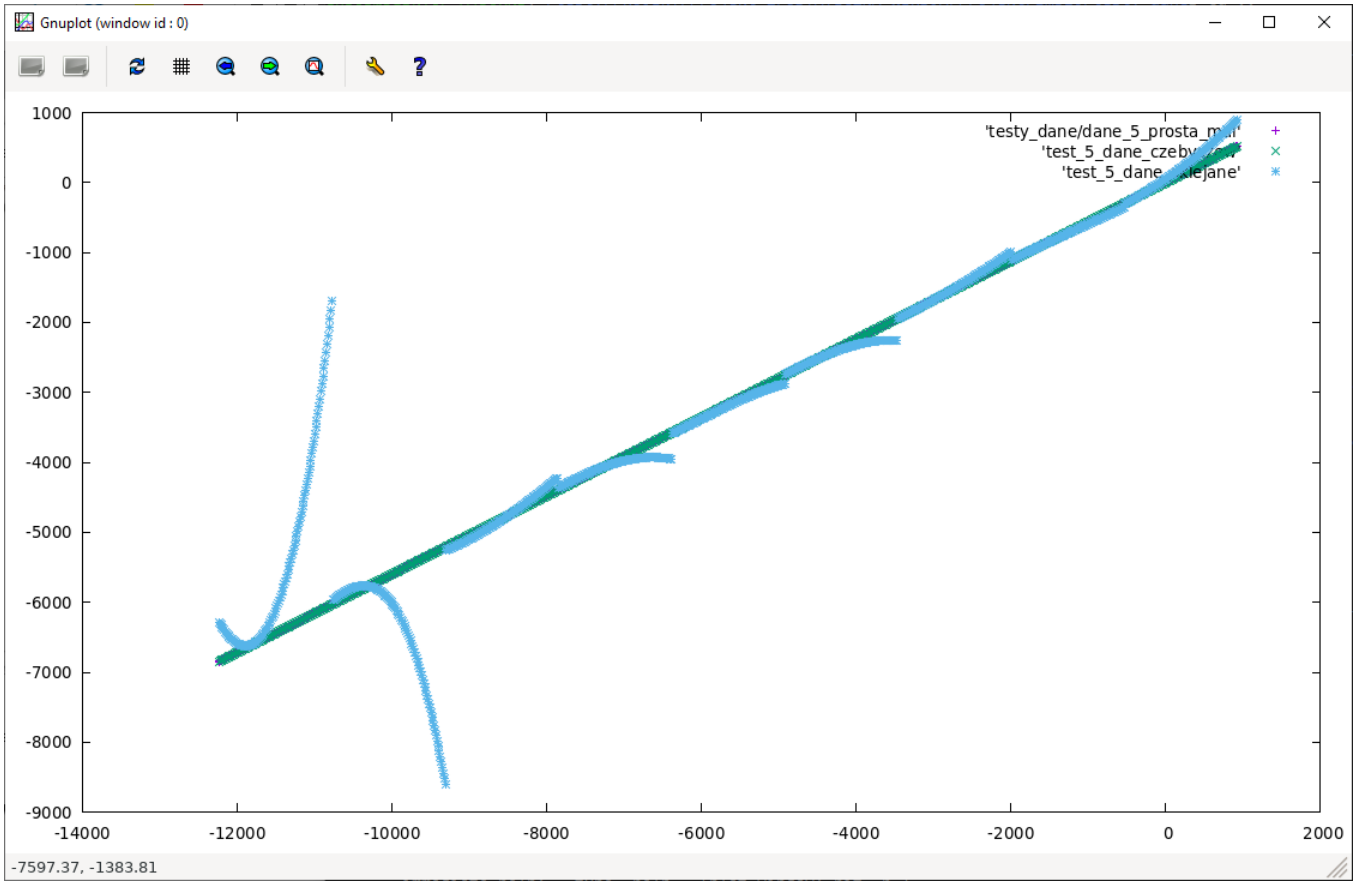
c) Plik z danymi zawiera zawiera 10000 punktów wygenrowane przez nasz program w zakresie <-10π;10π> bazowanych na funkcji tangens



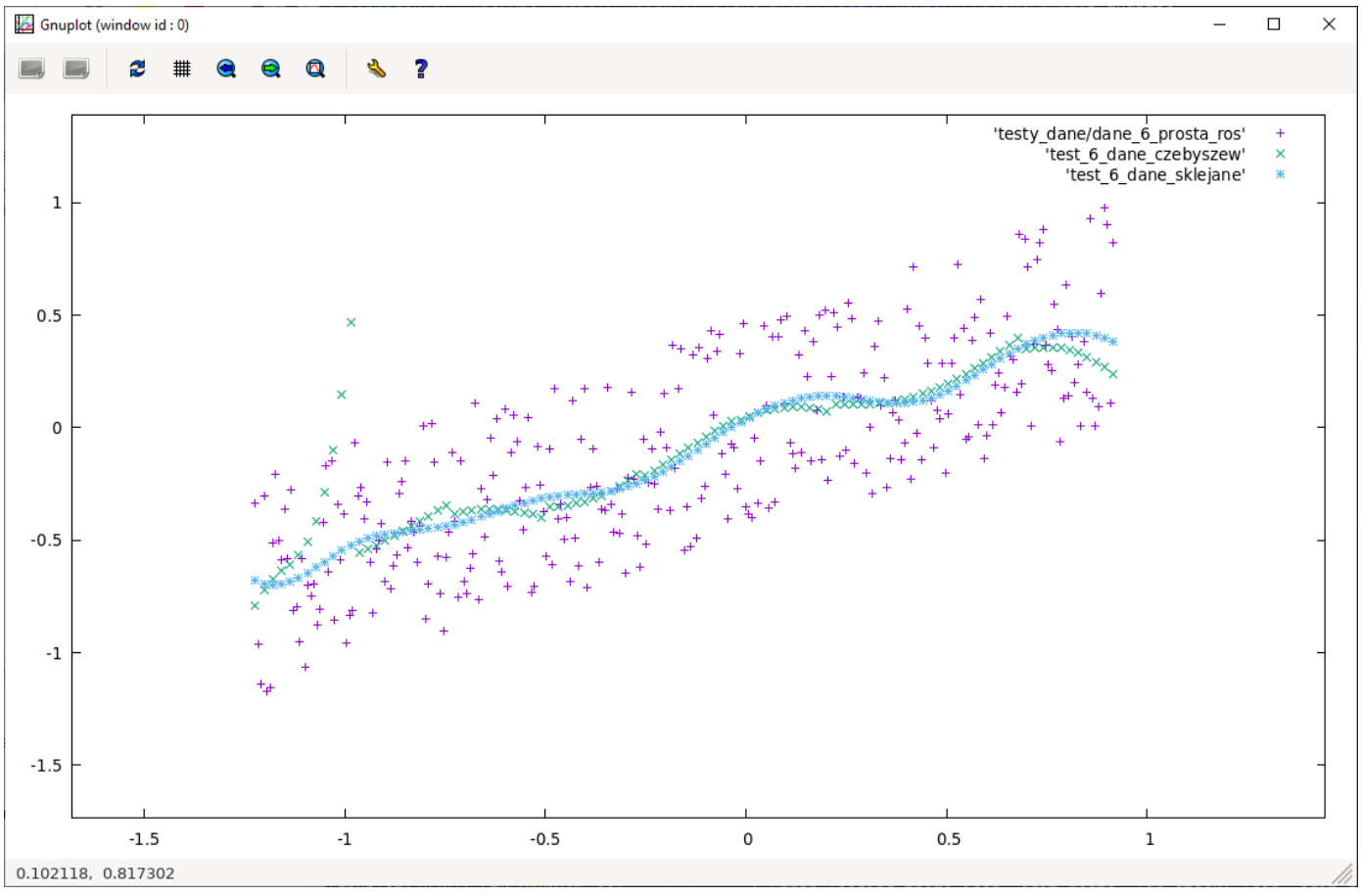
d) Plik z danymi zawiera zawiera 100 punktów wygenrowane przez nasz program w zakresie <-2π;2π> bazowanych na funkcji arcus tangens



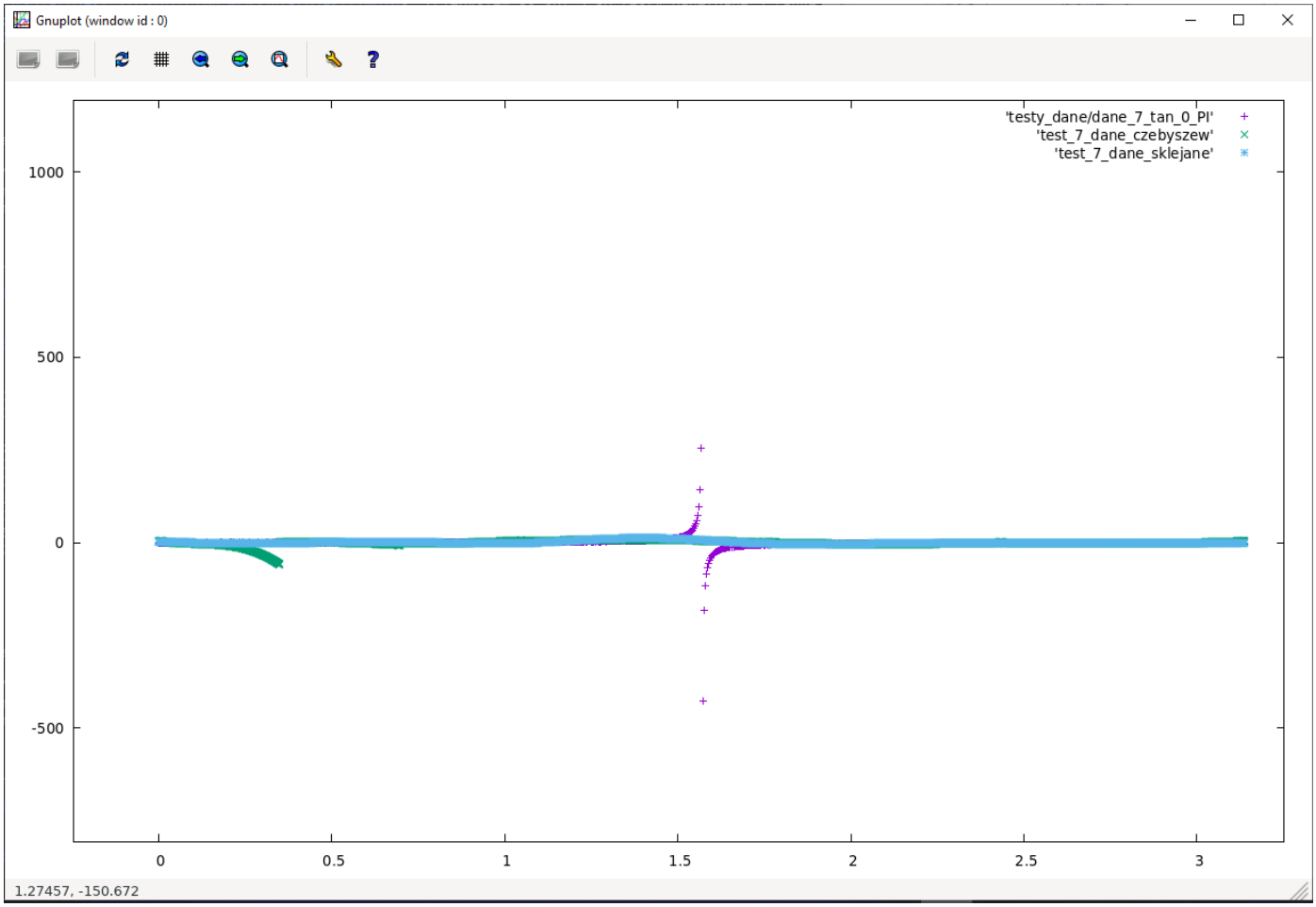
e) Plik z danymi zawiera zawiera 82382 punktów wygenrowane przez nasz program w zakresie <-12232, 516.87> bazowanych na prostej o wspolczynniku a=0.56.



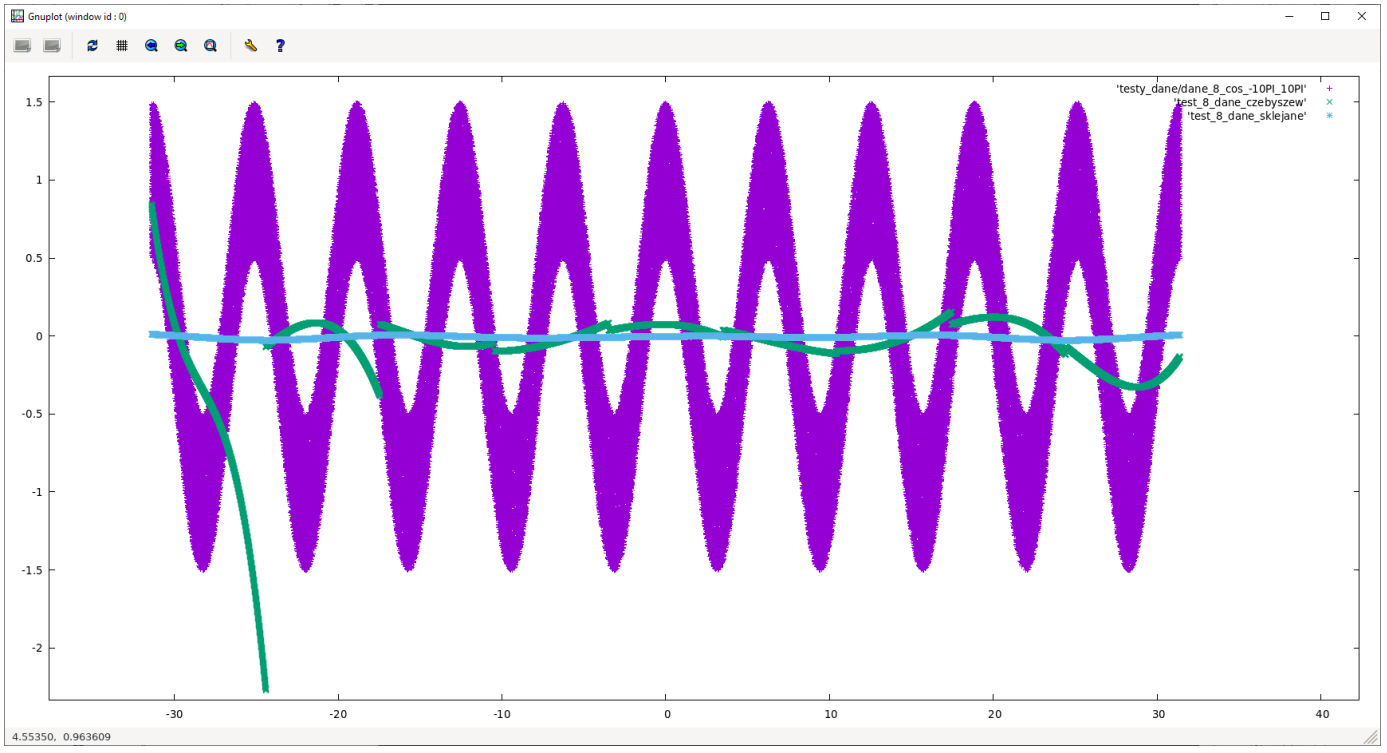
f) Plik z danymi zawiera zawiera 293 punktów wygenrowane przez nasz program w zakresie <-1.22, 0.92> bazowanych na prostej o wspolczynniku a=0.56.



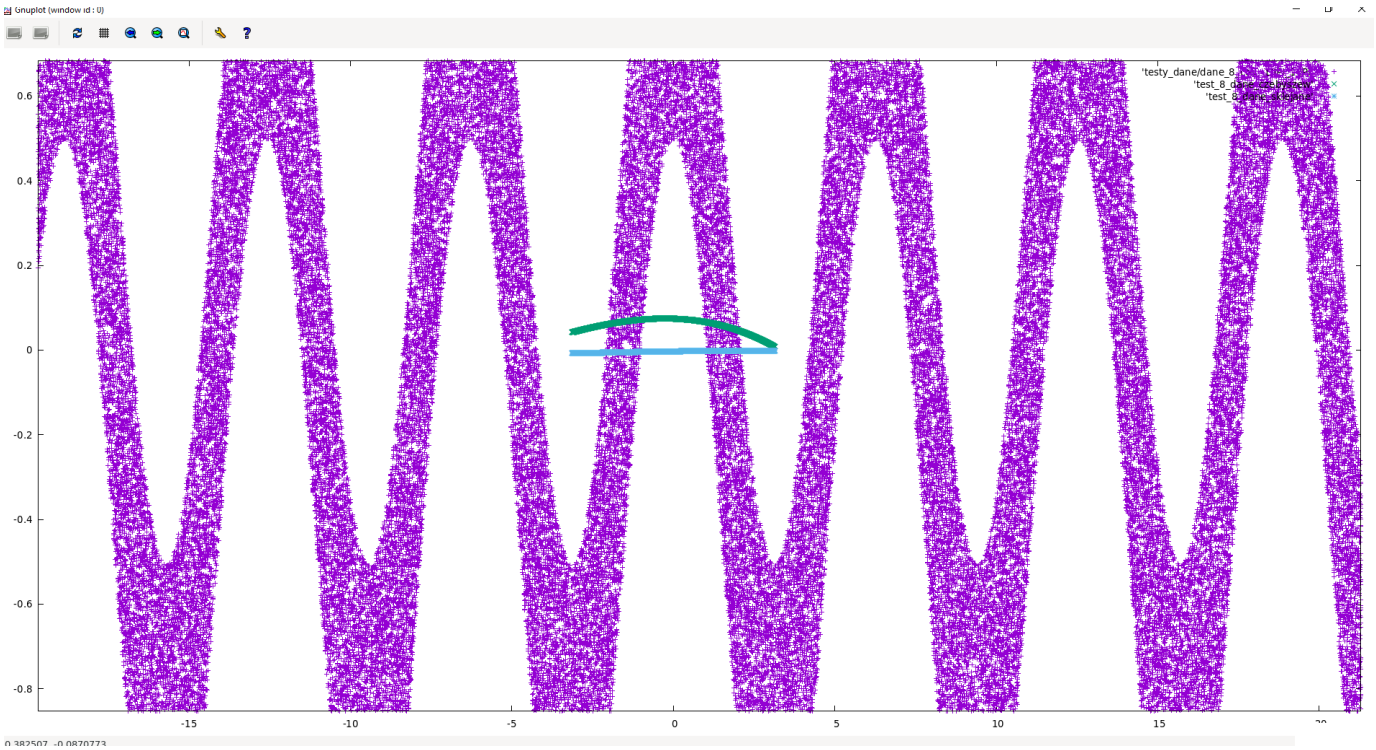
g) Plik z danymi zawiera zawiera 1000 punktów wygenrowane przez nasz program w zakresie <0, π> bazowanych na funkcji tangens.



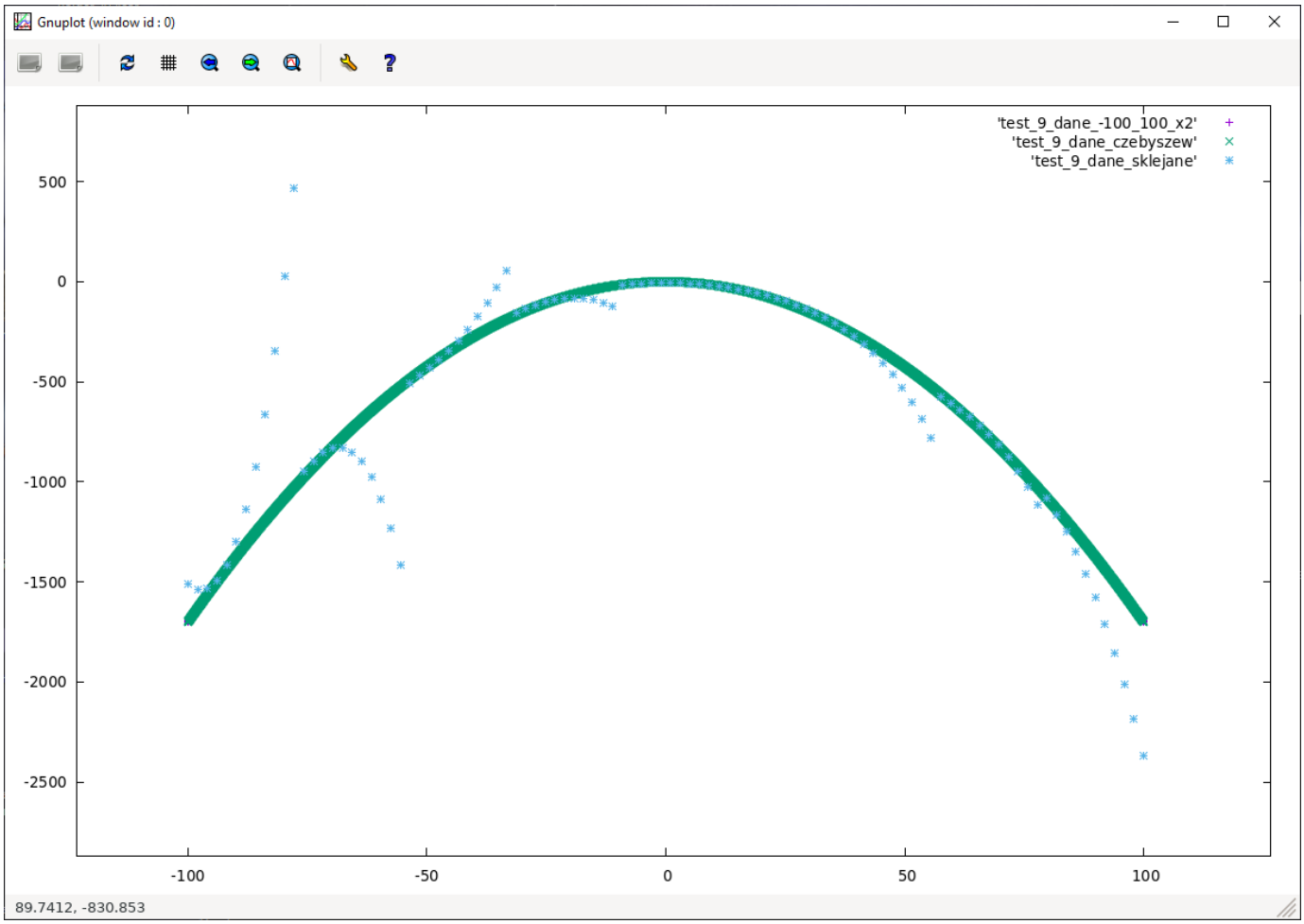
h) Plik z danymi zawiera zawiera 150000 punktów wygenrowane przez nasz program w zakresie < -10π, 10π> bazowanych na funckji cosinus.



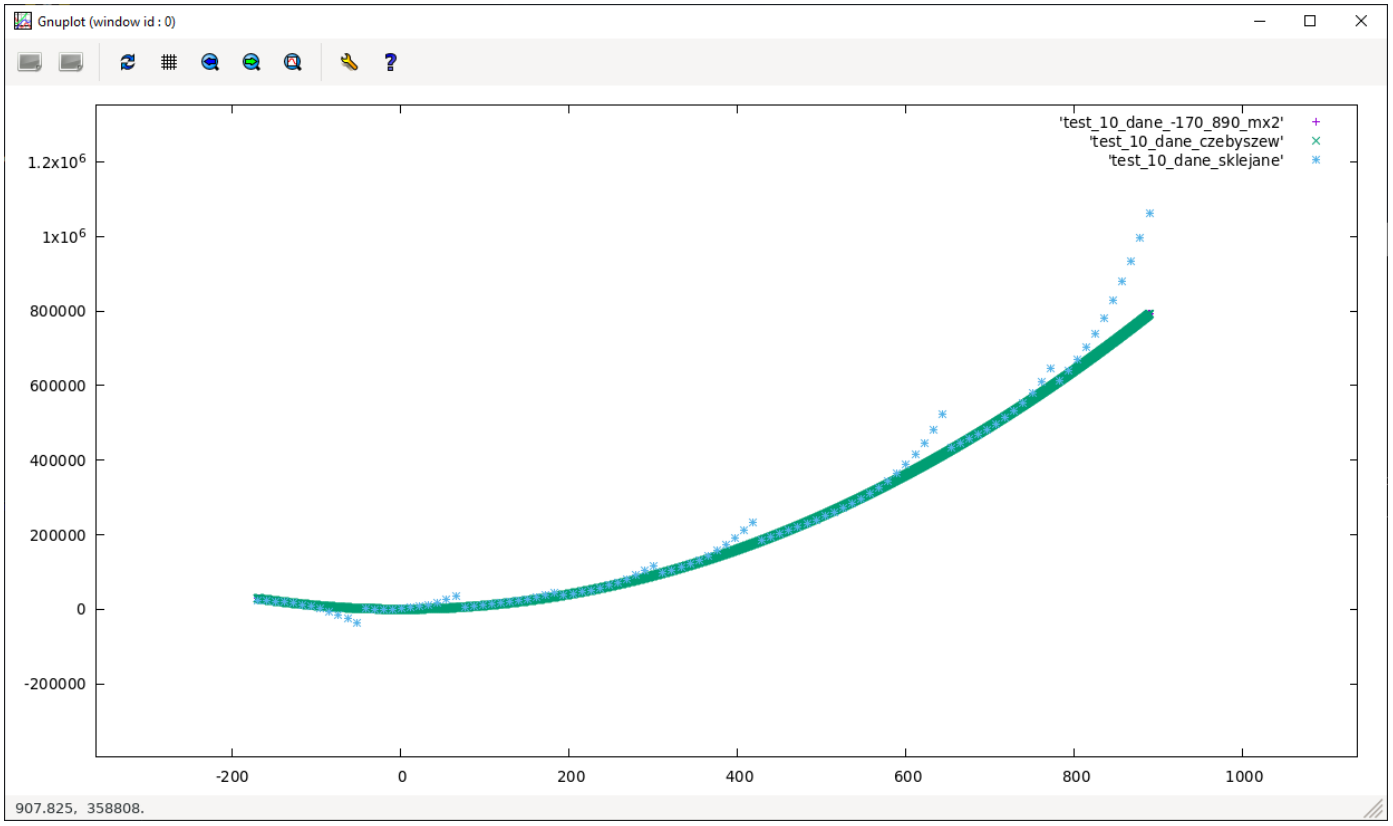
i) Plik z danymi zawiera zawiera 150000 punktów wygenrowane przez nasz program w zakresie <-10π, 10π> bazowanych na funckji cosinus. Funkcje aproksymujące zostały jednak utworzone na przedziale <-π, π>



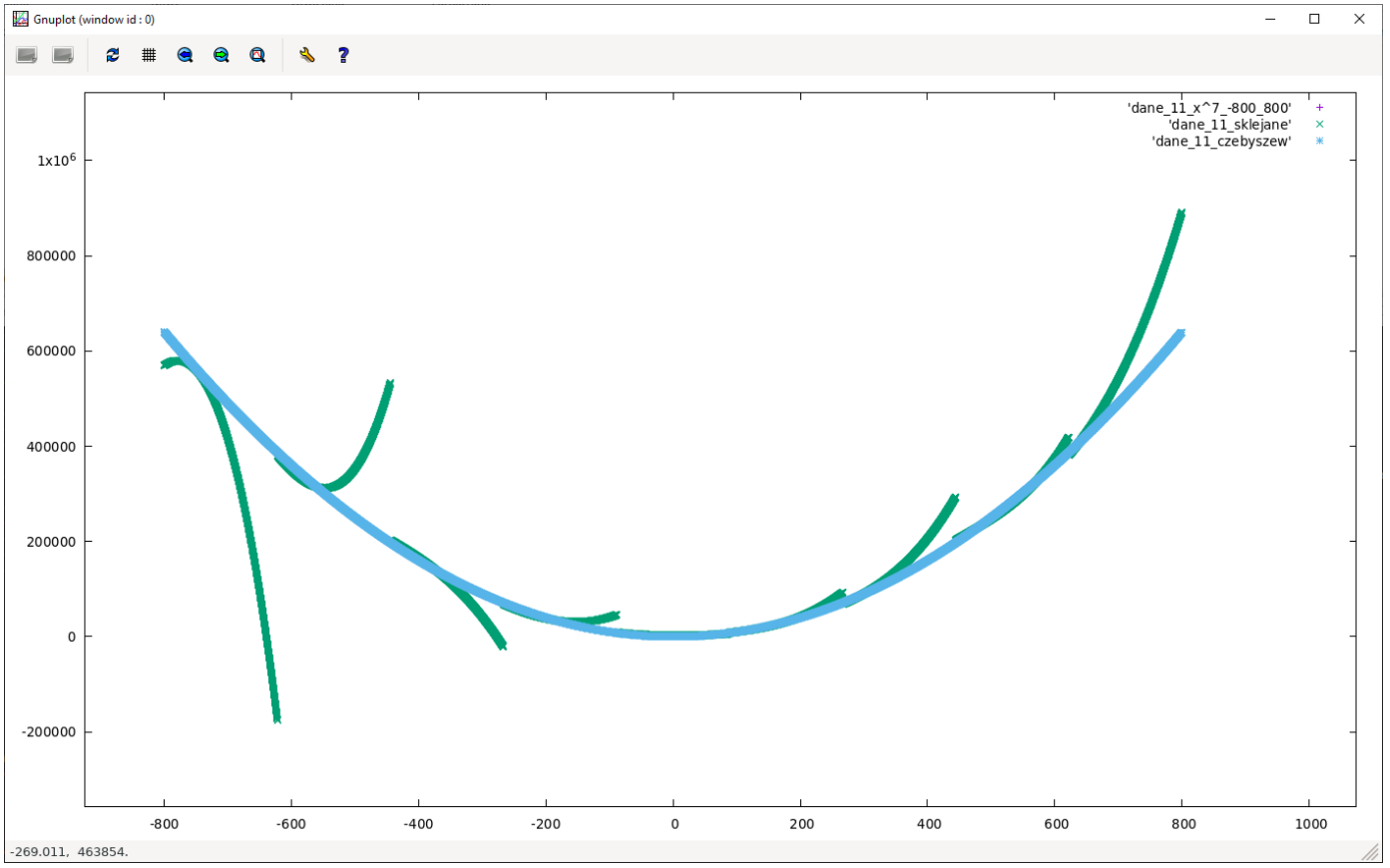
j) Plik z danymi zawiera zawiera 10000 punktów wygenrowane przez nasz program w zakresie <-100, 100> bazowanych na funkcji - x^2.



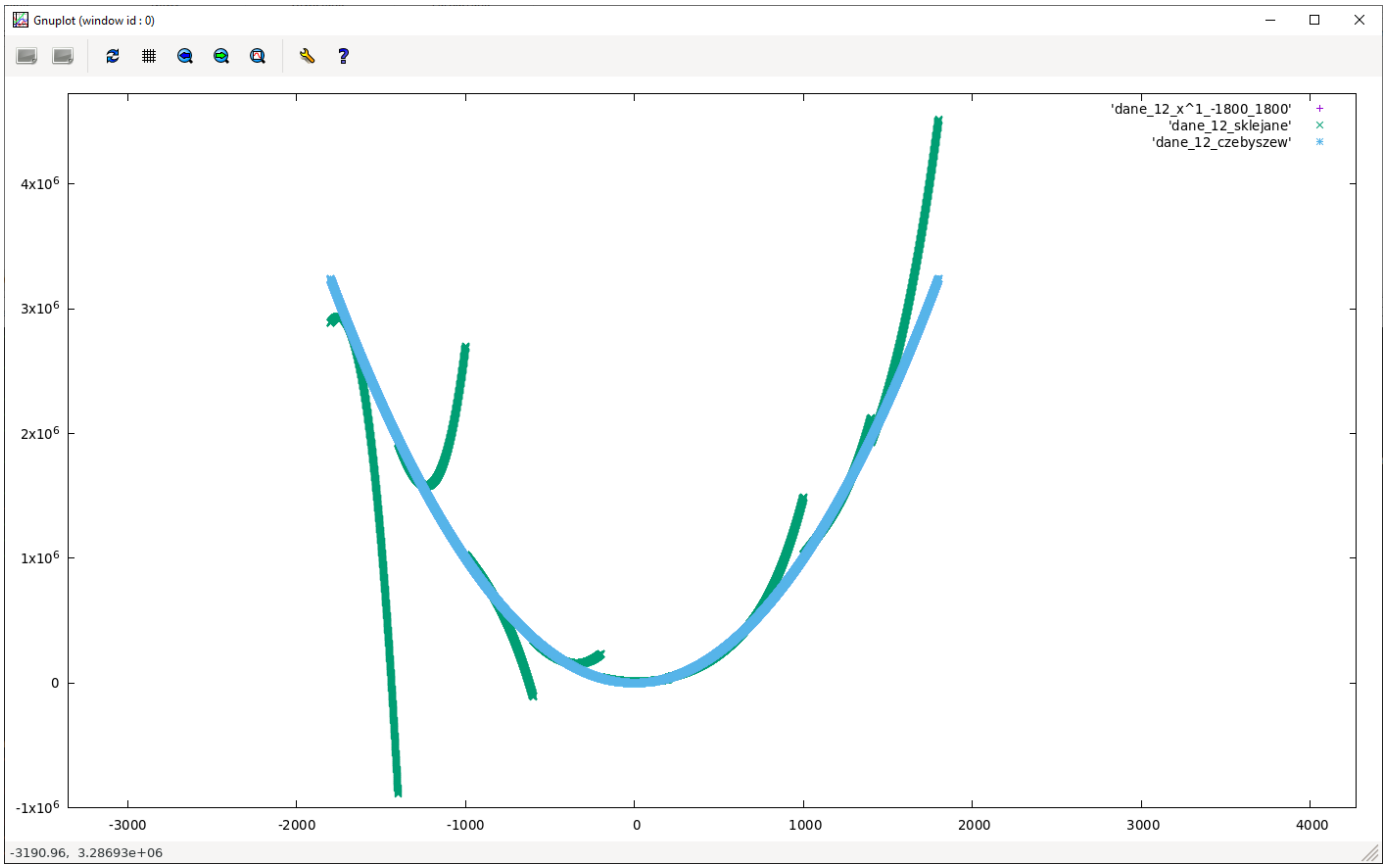
k) Plik z danymi zawiera zawiera 100000 punktów wygenrowane przez nasz program w zakresie <-170, 890> bazowanych na funkcji x^2.



l) Plik z danymi zawiera zawiera 237238 punktów wygenrowane przez nasz program w zakresie <-800, 800> bazowanych na funkcji wielomianowej z najwyższą potengą 7.



m) Plik z danymi zawiera zawiera 100000 punktów wygenrowane przez nasz program w zakresie <-1800, 1800> bazowanych na funkcji wielomianowej z najwyższą potengą 11.



5. Wnioski

Aproksymacja na tej bazie dobrze radzi sobie z przybliżaniem funkcjami monotonicznymi oraz z niewielkimi odchyleniami.

Podobnie jak baza funkcji sklejanych bardzo dokładnie przybliża funkcje liniowe dla niewielkiej ilości punktów, jednak zdecydowanie lepiej przybliża funkcje liniowe kiedy dane jest dużo więcej punktów (podpunkty e, f).

Problemem są dla niej funkcje okresowe, kiedy rozważamy większy okres. Jednak dla tangensa nawet przy pojedynczym okresie funkcja nie przybliża dobrze tej funkcji, co związane jest z gwałtownie rosnącymi wartościami tangensa przy małym wzroście argumentów. (podpunkty c, g, h, i)

Funkcje kwadratowe o dodatnim jak i ujemnym współczynniku zostają dokładnie przybliżone, w odróżnieniu od bazy funckji sklejanych. (podpunkty j, k)

Dla funkcji wielomianowych wyższych stopnii o różnych współczynnikach baza wielomianów Czebyszewa nie odwzorowuje dokładnie wartości, w odróżnieniu od bazy funkcji sklejanych. (podpunkty l, m)

6. Błędy znalezione w programie

a) Wycieki pamięci w aproksymacji na bazie funkcji sklejanych

W pierowtnym programie kompilując kod dla przykładowych danych mogliśmy zauważyć poniższe wycieki pamięci przy użyciu programu valgrind:

```
==530== LEAK SUMMARY:
==530==    definitely lost: 16,816 bytes in 8 blocks
==530==    indirectly lost: 3,360 bytes in 1 blocks
==530==      possibly lost: 0 bytes in 0 blocks
==530==    still reachable: 0 bytes in 0 blocks
==530==         suppressed: 0 bytes in 0 blocks
==530==
==530== ERROR SUMMARY: 8 errors from 8 contexts (suppressed: 0 from 0)
ehpop@DESKTOP-M456RNS:~/testy/PROJEKT/Projekt/lmp10$
```

Aby naprawić wycieki pamięci w kodzie zmodyfikowaliśmy go w podanych miejscach:

1.

```c
static int
realloc_pts_failed(points_t *pts, int size)
{
  double *temp_x= realloc(pts->x, size * sizeof(double));
  double *temp_y= realloc(pts->y, size * sizeof(double));
  if(temp_x == NULL || temp_y == NULL){
    free(pts->x);
    free(pts->y);
    return 1;
  }
  pts->x= temp_x; free(temp_x);
  pts->y= temp_y; free(temp_y);
  return 0;

}
```

W funckji realloc_points_failed() przed przypisaniem nowego adresu do wskazników pts->x oraz pts->y upewniamy się, że nie są one puste (różne od NULL).

2.

```c
void free_splines(spline_t *spl)
{
  if (spl != NULL)
  {
    if (spl->x != NULL)
      free(spl->x);
    if (spl->f != NULL)
      free(spl->f);
    if (spl->f1 != NULL)
      free(spl->f1);
    if (spl->f2 != NULL)
      free(spl->f2);
    if (spl->f3 != NULL)
      free(spl->f3);
  }
}
```

```c
void free_points(points_t *pts)
{
  if (pts != NULL)
  {
    if (pts->x != NULL)
      free(pts->x);
    if (pts->y != NULL)
      free(pts->y);
  }
}
```

```c
void free_wielomian(struct wielomian *ptr)
{
    if (ptr->wsp != NULL)
        free(ptr->wsp);
}
```

Zaimplementowalismy również 3 funkcje, które zwalniają pamięć po strukturach points, splines oraz wielomiany. W efekcie wywoływanie naszego programu dla bazy funkcji sklejanych nie powoduje wycieków pamięci.

```
HEAP SUMMARY:
    in use at exit: 0 bytes in 0 blocks
  total heap usage: 25 allocs, 25 frees, 88,600 bytes allocated

All heap blocks were freed -- no leaks are possible

ERROR SUMMARY: 1143 errors from 13 contexts (suppressed: 0 from 0)
```

b) Wycieki na bazie wielomianów Czebyszewa

Aby naprawić wycieki pamięci w naszej wersji programu korzystając z funkcji free() zwolniliśmy wskaźniki w strukturach w pliku wielomiany.c.

```
    struct wielomian *kontener, *wiel_2x = init_wielomian(1);
    wiel_2x->wsp[1] = 2;
    for (uint8_t x = 0; x < stopien - 1; x++)
    {
        struct wielomian *pom = mul_wielomian(Tpp, wiel_2x);
        kontener = sub_wielomian(pom, T);
        free(pom->wsp);
        free(pom);
        free(T->wsp);
        free(T);
        T = Tpp;
        Tpp = kontener;
    }
    free(T->wsp);
    free(T);
    free(wiel_2x->wsp);
    free(wiel_2x);
    return Tpp;
```

W rezultacie wycieki pamięci zostały zatrzymane.

```
==2669== HEAP SUMMARY:
==2669==     in use at exit: 0 bytes in 0 blocks
==2669==   total heap usage: 6,246,145 allocs, 6,246,145 frees, 218,651,000 bytes allocated
==2669==
==2669== All heap blocks were freed -- no leaks are possible
==2669==
==2669== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Autorzy Projektu: Marcin Ścisłowski, Tomasz Tkaczyk

Data: 23.01.2022r.