Sprawozdanie z zadania NUM8

1. Problematyka:

1. Opis zadania: Naszym zadaniem jest wyznaczenie składowych wektora C=(a,b,c,d) współczynników funkcji posiadając listę stu zaburzonych punktów należących do jej wykresu. Funkcja ma wzór

$$F(x,C)=a*x^2+b*\sin(x)+c*\cos(5x)+d*e^{-x}$$

Przez fakt nie znania przez nas wartości współczynników a,b,c,d musimy użyć aproksymacji. Wzór funkcji zależnej od wektora C ma więc postać:

$$F(x,C)=\sum_i^m a_i\phi_i(x)$$

(przepraszam za formatowanie, ale MS Word słabo sobie radzi)

Zgodnie z poleceniem, w celu znalezienia wartości tego wektora użyjemy metody aproksymacyjnej średniokwadratowej.

2. Teoria:

1. Aproksymacja średniokwadratowa to metoda bazująca na minimalizacjach błędu zdefiniowanej jako (gdzie C jest wektorem):

$$S(a)=\sum_i^m (y_i - F(x_i, C))^2$$

Szukany przez nas wektor C ma za zadanie minimalizację różnicy między kwadratami odległości między punktem dokładnym a przybliżonym, stąd nazwa metody.
Zgodnie z teorią znalezioną w Internecie, musimy rozwiązać układ równań $D^TDC=D^Tf$, w celu obliczenia najlepiej dopasowanego wektora C.

$$D=\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \dots & \phi_m(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_m(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_m(x_2) \\ \dots & \dots & \dots & \dots \\ \phi_0(x_n) & \dots & \dots & \phi_m(x_n) \end{bmatrix} \qquad f=\begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \dots \\ f(x_n) \end{bmatrix}$$

3. Rozwiązanie:

1. Struktura programu: W pliku main.py znajduje się rozwiązanie, w jego celu posługuję się tam wieloma funkcjami, które opisze w poniższym fragmencie:

W klasie FunctionApproximator znajdują się pola, które odpowiadają za wyrazy przy naszych szukanych współczynnikach, wszystkie są funkcjami pakietu numpy lub jego elementami, jak na przykład: numpy.e

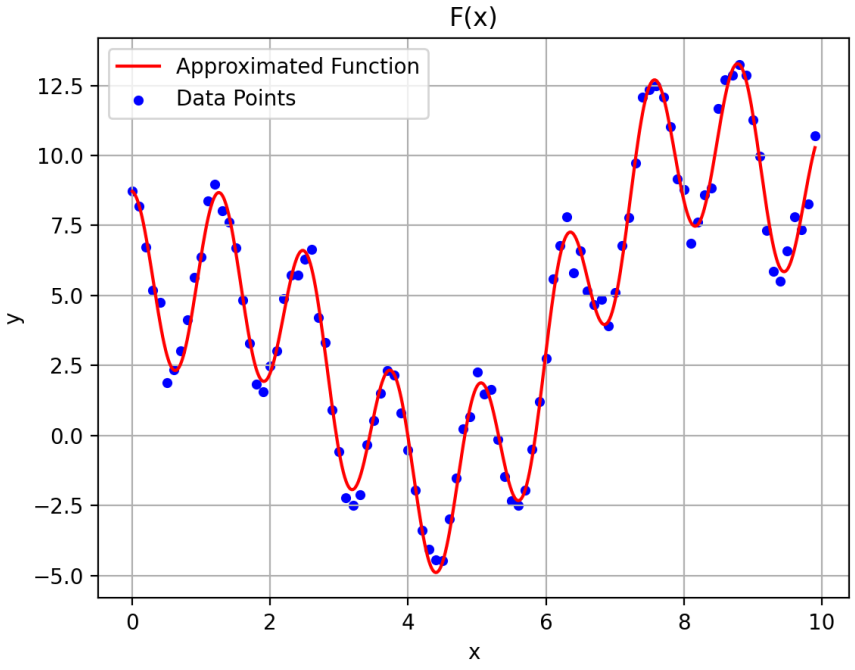
generate_points(self, count: int, components: List[Callable], exact_params: List[float], max_disturbance: float) - zwraca listę punktów pobierając tym samym wiele argumentów, takie jak ilość punktów do wygenerowania, Listę wyrazów(komponenty funkcji), listę parametrów funkcji, które będziemy aproksymować oraz zaburzenie.

solve_and_graph(self,title,components)-używa funkcji generującej macierz używając komponentów funkcji, a następnie rozwiązuje aproksymuje współczynniki, potem wywołuje funkcję graph.

Większość funkcji jest podobnie intuicyjnie nazwana w programie, więc pokrótce opisze ich znaczenie

-graph – rysuje graf.
-create_noisy_point-zaburza punkt,używany w generate_points.
-load_csv- ładuje plik z danymi w formacie .csv.
-plot_data_points- wypisuje na wykres punkty z pliku.
-plot_approximation- wykreśla funkcję z aproksymacji.

2. Wyniki:
Wektor C dla funkcji F(x,C) ma przybliżone wartości=[a: 0.10093369426913555, b: 4.023059455796112, c: 3.0887432722599537, d: 5.632839737726501]



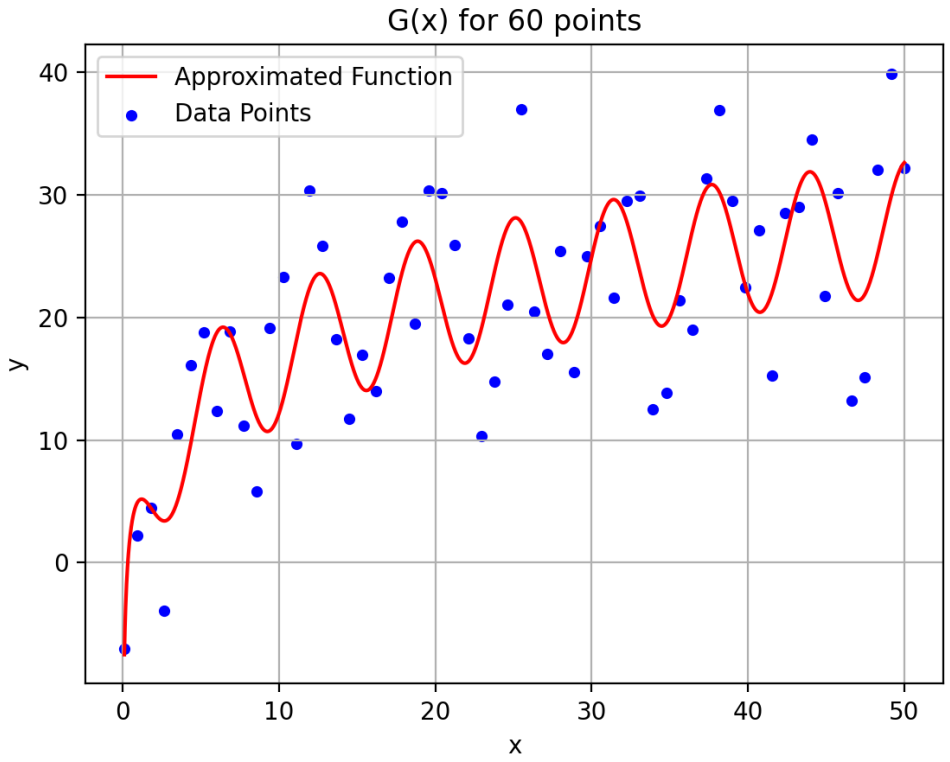
Skupmy się teraz na drugiej części zadania numerycznego- stwórzmy inną funkcję, U mnie G(x) ma postać:

$$G(x)=7 * \ln(x) + 5 * \frac{1}{x+e} - \cos(x) + 6 * \sin(x)$$

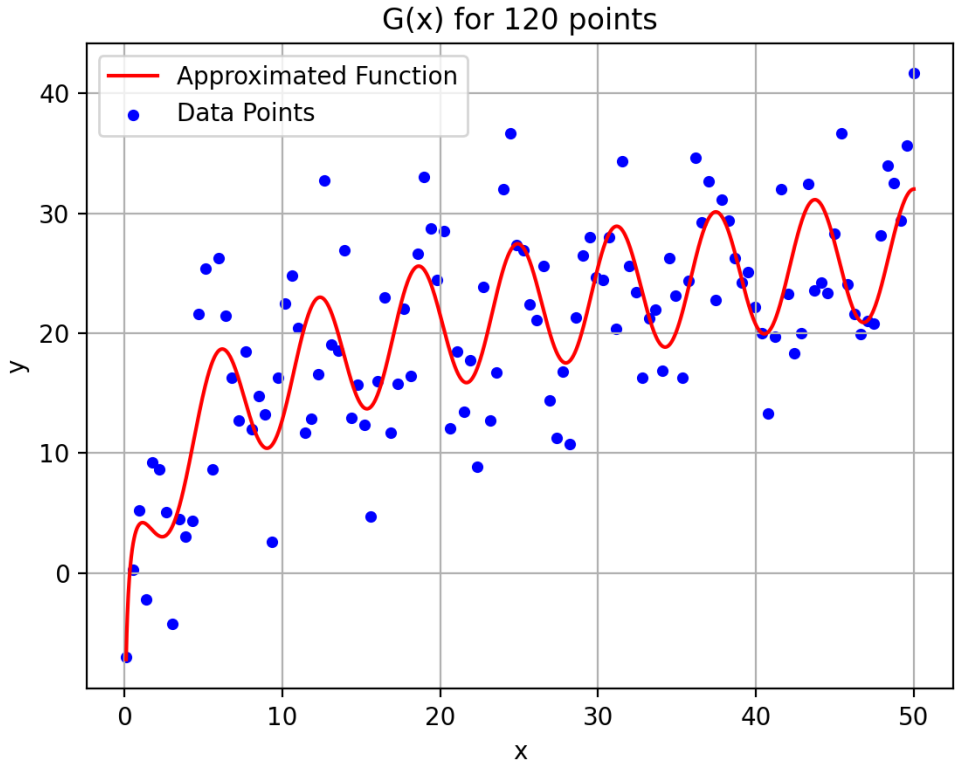
Sprawdźmy jak blisko tymi samymi funkcjami aproksymacji wyznaczymy wartości współczynników dla różnych ilości punktów:

$$G(x)=a * \ln(x) + b * \frac{1}{x+e} + c * \cos(x) + d * \sin(x)$$

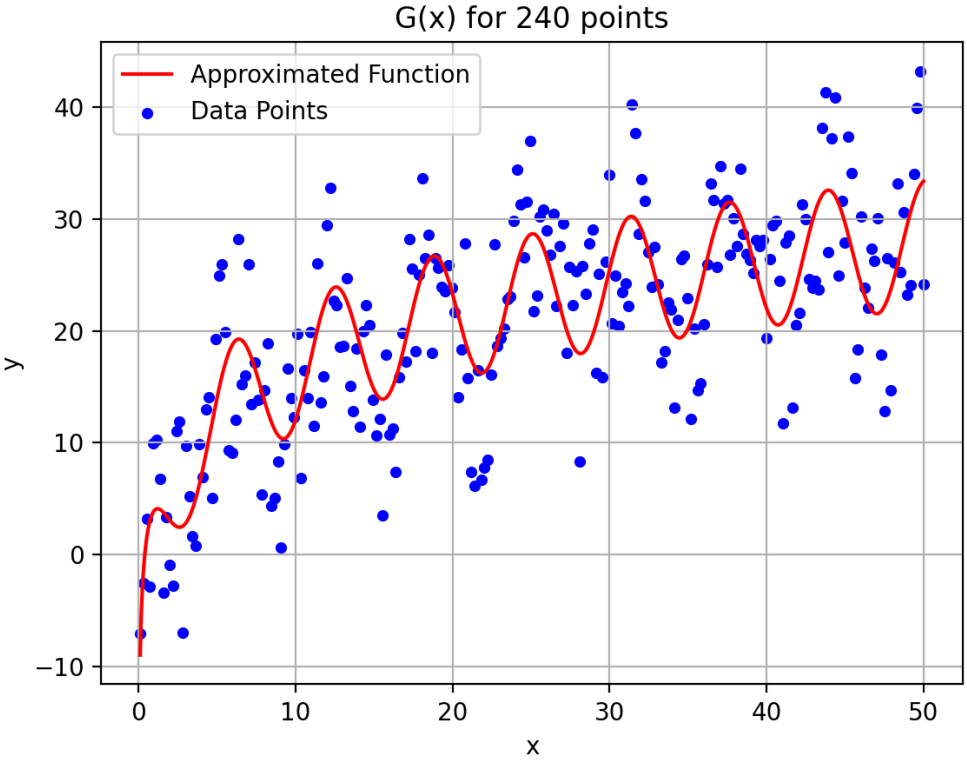
Po wygenerowaniu 60 punktów, a następnie zaburzeniu ich nasze współczynniki mają wartości: [a: 6.929544117825168, b: 8.588829826263208, c: -0.2705099044085893, d: 5.467700532220522]

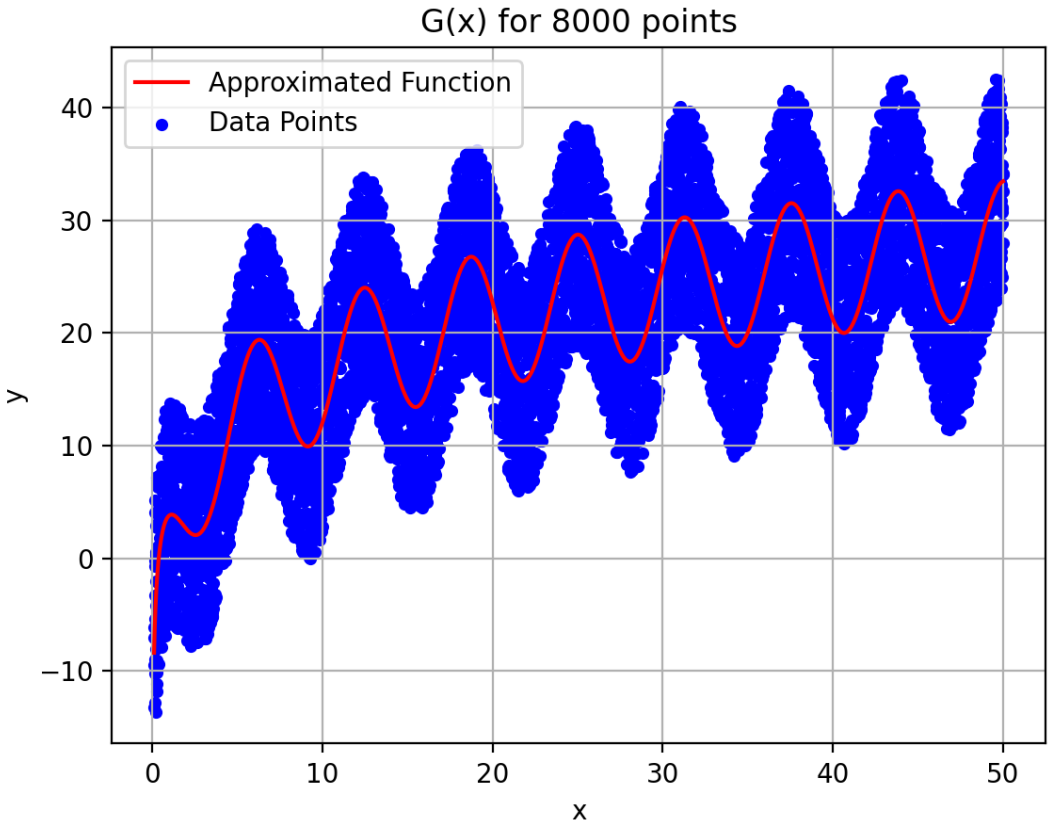


Po wygenerowaniu 120 punktów, a następnie zaburzeniu ich nasze współczynniki mają wartości: [a: 6.772977552964046, b: 9.57801160940573, c: -1.4512709400484274, d: 5.150824059697307]



Po wygenerowaniu 240 punktów, a następnie zaburzeniu ich nasze współczynniki mają wartości: [a: 7.068553572695152, b: 4.503718400134863, c: -0.4626525765623977, d: 5.747620785820249]





Postanowiłem wygenerować jeszcze wykres dla 8 tysięcy punktów, jak widać aproksymacja działa i wyznacza funkcję z dobrą dokładnością, podobną do zamierzonej funkcji.

4.Wnioski:

Aproksymacja jest coraz dokładniejsza wraz ze wzrostem danych, lecz zależy od ich rozrzutu.