**Sprawozdanie z listy 3. – Modele systemów dynamicznych L**

Filip Antoniak (279929)

1. **Wstęp**

Celem tego sprawozdania jest wykorzystanie języka python do porównania dwóch metod numerycznych stosowanych do rozwiązywania równań różniczkowych: metody Eulera oraz funkcji *odeint* z pakietu *scipy*.

Zestawienie metody Eulera, będącej jedną z najprostszych metod rozwiązywania równań różniczkowych z rozwiązaniami uzyskanymi przy pomocy funkcji *odeint*, pozwoli na określenie błędu aproksymacji oraz analizę stabilności metody Eulera.

W badaniu będącym zestawieniem i analizą wykresów przebiegu równań zostaną wykorzystane równanie Lotki-Volterry oraz układ Lorenza. Model Lotki-Volterry to nieliniowy układ równań różniczkowych stosowany do symulacji zachowań populacji ofiar i drapieżników w pewnym ekosystemie.

Obraz zawierający Czcionka, tekst, biały, pismo odręczne

Opis wygenerowany automatycznie

Tabela 1: Legenda oznaczeń parametrów dla równania Lotki-Volterry.

|  |  |
| --- | --- |
| **Ofiary** | **Drapieżniki** |
| x- populacja ofiar | y- populacja drapieżników |
| a -częstość narodzin ofiar | c- częstość narodzin drapieżników |
| b- częstość umierania ofiar | d- częstość umierania drapieżników |

Układ Lorenza to układ trzech nieliniowych równań różniczkowych opisujący procesy związane z przemianą ciepła w atmosferze. Układ ten powiązany jest z teorią chaosu deterministycznego, charakteryzuje go duża wrażliwość na warunki początkowe dla pewnych parametrów.[[1]](#footnote-1)

Obraz zawierający tekst, Czcionka, pismo odręczne, diagram

Opis wygenerowany automatycznie

Tabela 2: Legenda oznaczeń parametrów dla układu Lorenza.

|  |  |  |
| --- | --- | --- |
| σ - liczba Prandtla | β | ρ - liczba Rayleigha |
| lepkość ośrodka | rozmiary obszaru | przewodnictwo cieplne |

1. **Metoda Eulera**

Metoda Eulera pozwala na rozwiązywanie równań różniczkowych. Została ona przedstawiona w 1768 roku w pracy Leonharda Eulera pt. *Institutiones calculi differentialis*. Jej zasada działania jest następująca:

* przyjmuje się stały krok czasowy
* z definicji wynika, że
* fakt ten, jest wykorzystany do obliczenia kolejnych wartości y(t + dt) ze wzoru:

Przedstawiony sposób działania, jest stosunkowo prosty, co ułatwia jego implementacje. Nie mniej jest to jednocześnie wada tego podejścia, ponieważ metoda Eulera charakteryzuje się niską dokładnością, oraz problemami ze stabilnością, które rosną razem ze wzrostem kroku czasowego. Co zostanie wykazane na wykresach oraz poparte błędem aproksymacyjnym, względem funkcji odeint.

1. **Funkcja odeint**

Jest to funkcja z [pakietu scipy](https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html), w pythonie, służąca do rozwiązywania równań różniczkowych. Jej działanie jest wysokopoziomowe, korzysta z programu lsoda z pakietu odepack biblioteki FORTRAN. FORTRAN automatycznie wybiera algorytm, metode Adamsa lub metodę Backward Differention Formula, w zależności od tego jak duże są róznice w szybkości zmian poszczególnych składników.

Jej działanie jest bardziej enigmatyczne i proporcjonalnie bardziej wydajne i dokładne, dlatego na potrzeby tego sprawozdania zostanie wykorzystane jako punkt odniesienia do obliczenia średniego błędu aproksymacji dla metody Eulera.

1. **Model Lotki-Volterry**

* **Metoda Eulera**

**Obraz zawierający tekst, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznie**

Rysunek 1: Rezultat symulacji modelu Lotki-Volterry, metodą Eulera x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 1 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

Na Rysunku 1. krok jest ustawiony na wartość dt = 1, co sprawia, że wykres, pomimo zadanego przedziału czasowego [0, 25] zatrzymuje się w połowie. Obrazuje to ograniczenia metody Eulera dla dużych kroków czasowych. Wykres zatrzymał się w tym miejscu, ponieważ wartości jakie osiągały x(t) / y(t) zbliżały się do + / - nieskończoności.

Tabela 3: Wartości jakie przyjmowały x(t) oraz y(t) dla symulacji metodą Eulera z Rysunku 1.

|  |  |  |
| --- | --- | --- |
| Czas (t) | Ofiary (x) | Drapieżnicy (y) |
| 0 | 2.0 | 1.0 |
| 1 | 3.2 | 2.3 |
| 2 | 2.624 | 7.498 |
| 3 | -6.032 | 20.988 |
| 4 | 62.691 | -86.591 |
| 5 | 3395.011 | -4403.400 |
| 6 | 8,977,222 | -11,962,750 |
| 7 | 64,435,390,000,000 | -85,913,840,000,000 |
| 8 | 3.321535e+27 | -4.428713e+27 |

**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie**

Rysunek 2: Rezultat symulacji modelu Lotki-Volterry, metodą Eulera x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 0.33 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

**Obraz zawierający tekst, Wykres, linia, diagram

Opis wygenerowany automatycznie**

Rysunek 3: Rezultat symulacji modelu Lotki-Volterry, metodą Eulera x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 0.1 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie**

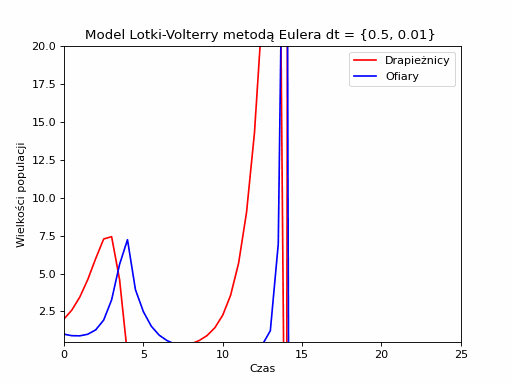
Rysunek 4: Rezultat symulacji modelu Lotki-Volterry, metodą Eulera x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 0.01 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie**

Rysunek 5: Rezultat symulacji modelu Lotki-Volterry, metodą Eulera x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 0.0001 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

Analiza wykresów pokazuje, że im niższy krok czasowy dt, tym bardziej dokładne i zgodne z rzeczywistością są rezultaty metody Eulera. Na Rysunku 6. Widać, jak zmienia się wykres równania Lotki-Volterry, obliczany metodą Eulera, gdy zmienia się skok czasowy.



Rysunek 6: Rezultat symulacji modelu Lotki-Volterry, metodą Eulera x(t), y(t) dla czasu t ∈ [0, 25] przy krokach [0.5, 0.01] ze zmianą o 0.01 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

* **Z wykorzystaniem funkcji odeint**

**Obraz zawierający linia, tekst, Wykres, diagram

Opis wygenerowany automatycznie**

Rysunek 7: Rezultat symulacji modelu Lotki-Volterry, funkcją odeint x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 1 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

.

**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie**

Rysunek 8: Rezultat symulacji modelu Lotki-Volterry, funkcją odeint x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 0.33 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie**

Rysunek 9: Rezultat symulacji modelu Lotki-Volterry, funkcją odeint x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 0.01 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie**

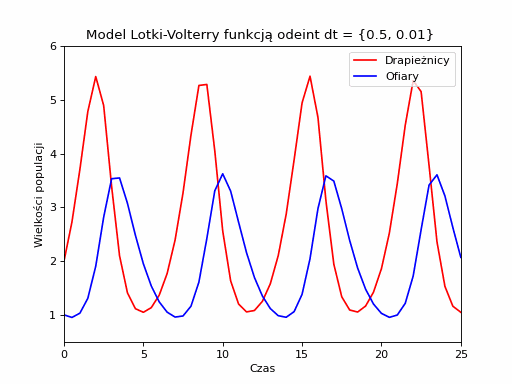
Rysunek 10: Rezultat symulacji modelu Lotki-Volterry, funkcją odeint x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 0.0001 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

**Obraz zawierający linia, Wykres, diagram, tekst

Opis wygenerowany automatycznie**

Rysunek 11: Rezultat symulacji modelu Lotki-Volterry, funkcją odeint x(t), y(t) dla czasu t ∈ [0, 25] przy kroku dt = 3 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

Wykresy te pokazują, że funkcja odeint, sprawdza się niż metoda Eulera, a różnicę te widać najbardziej przy stosunkowo dużych krokach czasu, np. dt = 1 (Rysunek 7) oraz dt = 3 (Rysunek 11).

****

Rysunek 12: Rezultat symulacji modelu Lotki-Volterry, funkcją odeint x(t), y(t) dla czasu t ∈ [0, 25] przy krokach [0.5, 0.01] ze zmianą o 0.01 dla parametrów a = 1.2, b = 0.6, c = 0.3, d = 0.8 oraz populacjach początkowych x0 = 2 y0 = 1.

Na Rysunku 12. widać, jak zmienia się wykres równania Lotki-Volterry utworzony przy użyciu funkcji odeint dla różnych kroków czasowych od *dt=0.5* do *dt=0.01* ze zmianą *0.01*. Przypadek ten pokazuje, że funkcja odeint nie jest już tak uzależniona od wielkości kroku jak w przypadku metody Eulera.

1. **Układ Lorenza**

* **Metoda Eulera**

**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznieObraz zawierający tekst, linia, diagram, Wykres

Opis wygenerowany automatycznie**Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie

Rysunek 12: Rezultat symulacji za pomocą metody Eulera przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.03

**Obraz zawierający diagram, szkic, Wykres, tekst

Opis wygenerowany automatycznie** **Obraz zawierający diagram, rysowanie, szkic, linia

Opis wygenerowany automatycznie** Obraz zawierający tekst, diagram, krąg, linia

Opis wygenerowany automatycznie

Rysunek 13: Rezultat symulacji za pomocą metody Eulera przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.0249

Gdy krok jest ustawiony na wartość 0.025 wykresy wyglądają jak na Rysunku 12. Sytuacja zmienia się od wartości dt < 0.025 np. 0.0249 na rysunku 13. Gdzie wykresy zaczynają przypominać ten właściwy.

**Obraz zawierający diagram, Wykres, tekst, linia

Opis wygenerowany automatycznie** **Obraz zawierający rysowanie, diagram, szkic, linia

Opis wygenerowany automatycznie** Obraz zawierający krąg, diagram, linia, tekst

Opis wygenerowany automatycznie

Rysunek 14: Rezultat symulacji za pomocą metody Eulera przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.02

**Obraz zawierający diagram, Wykres, tekst, zrzut ekranu

Opis wygenerowany automatycznie** **Obraz zawierający diagram, rysowanie, szkic, linia

Opis wygenerowany automatycznie** Obraz zawierający krąg, tekst, zrzut ekranu, diagram

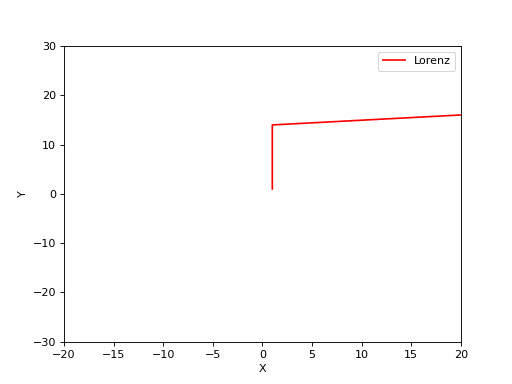
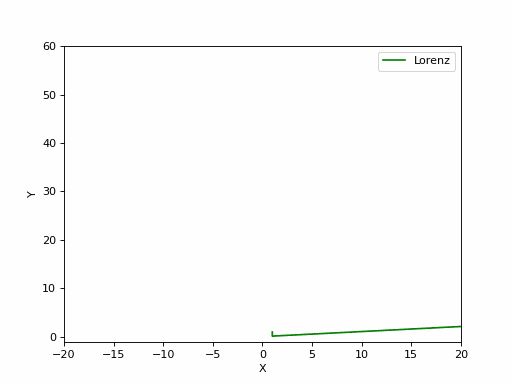
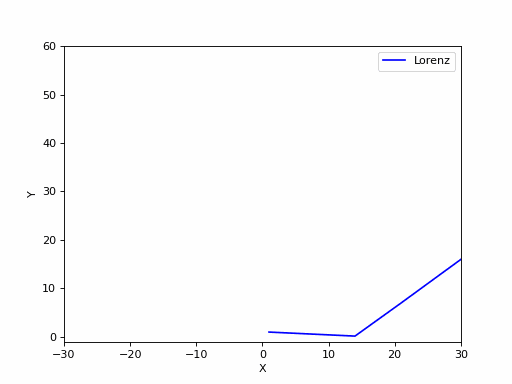
Opis wygenerowany automatycznie

Rysunek 15: Rezultat symulacji za pomocą metody Eulera przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.001

Obraz zawierający szkic, rysowanie, diagram, krąg

Opis wygenerowany automatycznie

Rysunek 16: Rezultat symulacji za pomocą metody Eulera przedstawiający układ Lorenza z(x, y) w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.0001

**** **** 

Rysunek 17: Rezultat symulacji układu Lorenza, metodą Eulera dla czasu t ∈ [0, 25] przy krokach [0.5, 0.01] ze zmianą o 0.01, dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1

* **Z wykorzystaniem funkcji odeint**

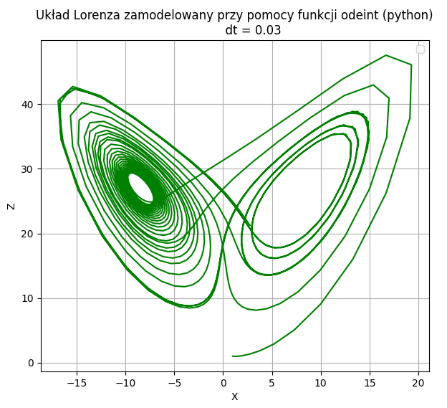
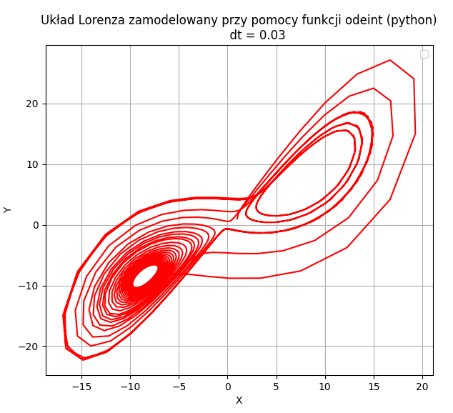
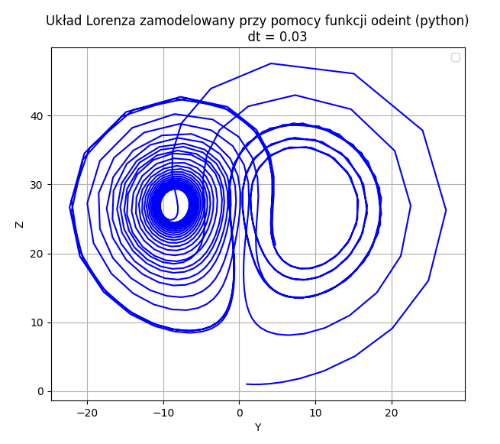
**Obraz zawierający linia, Wykres, diagram, tekst

Opis wygenerowany automatycznieObraz zawierający diagram, linia, Wykres, origami

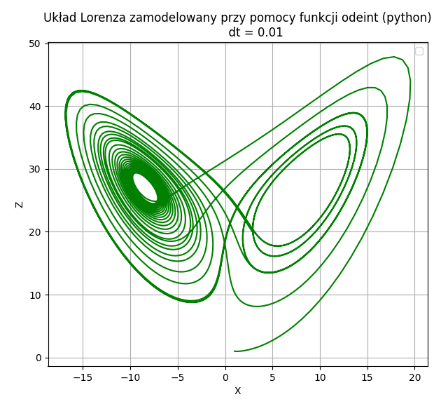
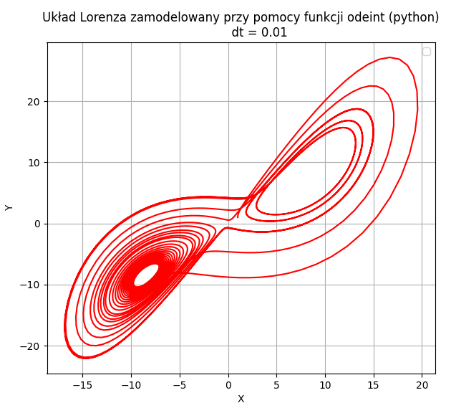
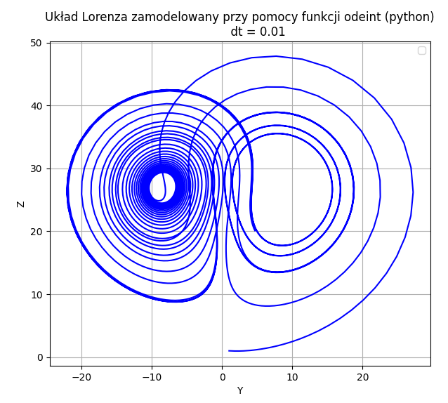
Opis wygenerowany automatycznie**Obraz zawierający linia, diagram, Wykres, origami

Opis wygenerowany automatycznie

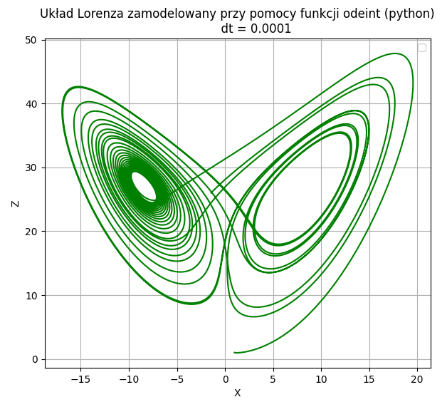
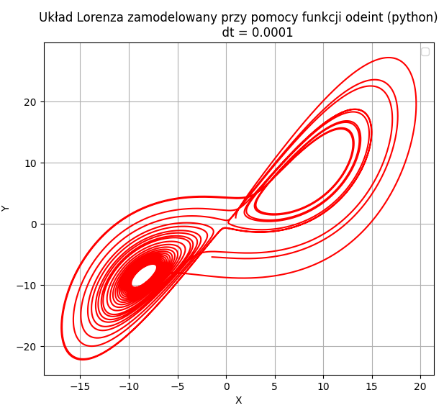
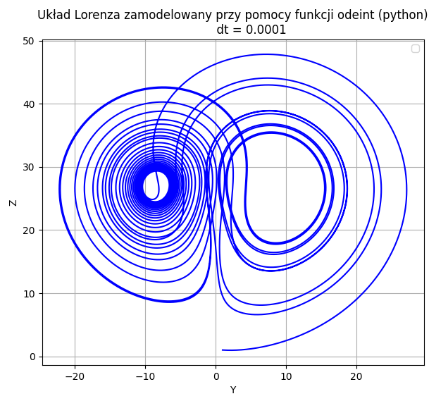
Rysunek 18: Rezultat symulacji za pomocą pakietu scipy przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 1

****

Rysunek 19: Rezultat symulacji za pomocą pakietu scipy przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.03

****

Rysunek 20: Rezultat symulacji za pomocą pakietu scipy przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.01

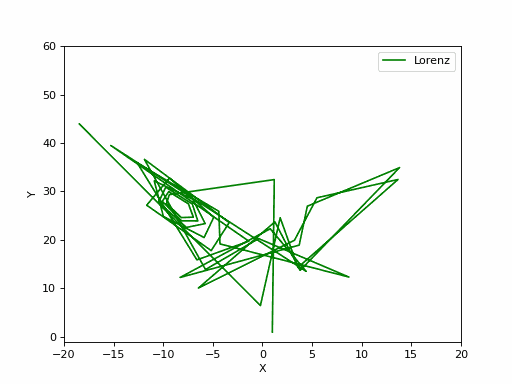
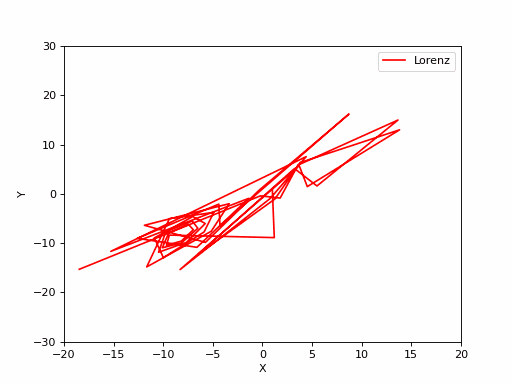
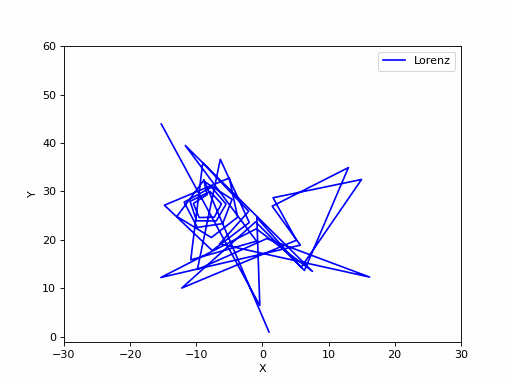
****

Rysunek 21: Rezultat symulacji za pomocą pakietu scipy przedstawiający układ Lorenza w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.0001

Obraz zawierający szkic, rysowanie, diagram, krąg

Opis wygenerowany automatycznie

Rysunek 22: Rezultat symulacji za pomocą funkcji odeint przedstawiający układ Lorenza z(x, y) w czasie t∈[0,25] dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 oraz kroku dt= 0.0001

****

Rysunek 23: Rezultat symulacji układu Lorenza, funkcją odeint dla czasu t ∈ [0, 25] przy krokach [0.5, 0.01] ze zmianą o 0.01, dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1

1. **Średni błąd aproksymacji**

Dla modelu Lotki-Volterry przy a= 1.2, b= 0.6, c= 0.3, d= 0.8 oraz populacjach początkowych x0 = 2 i y0 = 1: Średni błąd aproksymacji metody Eulera względem odeint dla kroku dt=0.1 obliczony ze wzoru:

*MAEx = 1.67, MAEy = 0.97*

Więc dla dt = 0.1 **MAE = 1.32**

Natomiast gdy dla identycznych warunków początkowych zbadamy MAE gdy dt = 0.01,

*MAEx = 0.1, MAEy =0.07*

Wtedy dla dt = 0.01 **MAE = 0.085**

Obliczając średni błąd aproksymacji dla układu Lorenza, wykorzystane zostaną podobne wzory, rozszerzone o MAEz, dla parametrów σ= 10, β=83, ρ= 28, x(0)=y(0)=z(0)=1 dla dt=0.01

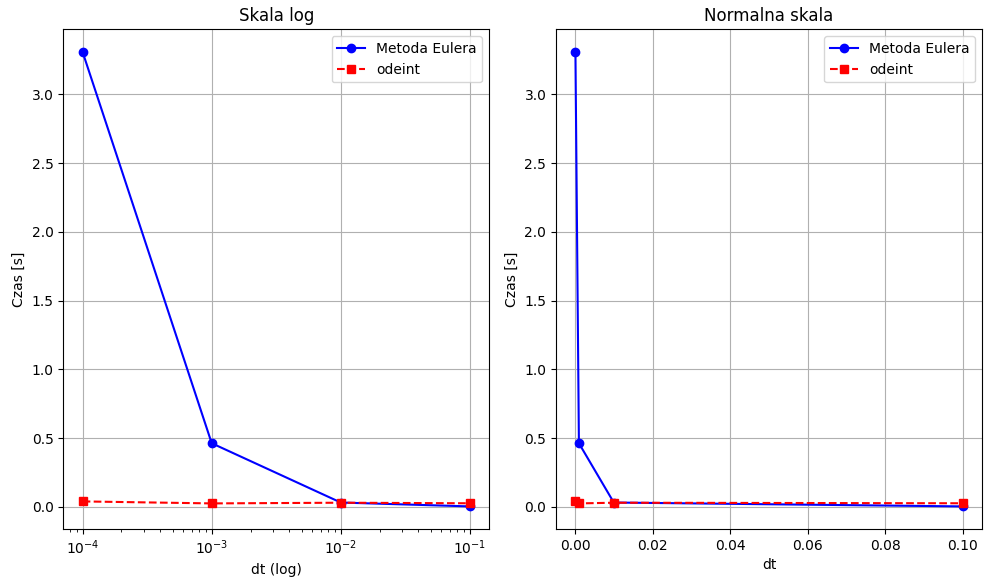
*MAEx = 10.3, MAEy = 11.2, MAEz = 7.8*

Wtedy dla dt = 0.01 **MAE = 9.8**

Na podstawie tych obliczeń można zaobserwować zależność między dokładnością metody Eulera a wielkością kroku dt. Im większy krok, tym większy średni błąd aproksymacji – tym obliczenia dalsze od prawdy.

Większy błąd w przypadku układu Lorenza może być spowodowany naturą układu, który charakteryzuje się deterministyczną chaotycznością, ponad to wyróżniona jest tam również zmienna Z, różnica może wskazywać też na konieczność wykonania normalizacji.

1. **Złożoność obliczeniowa**

****

Rysunek 24: Zestawienie czasu potrzebnego na wywołanie funkcji odeint, oraz metody Eulera dla kroków dt {0.1, 0.01, 0.001, 0.0001}

Rysunek 24. pokazuje, że metoda Eulera, pomimo tego, że działa lepiej, gdy krok jest mniejszy, działa też znacznie wolniej. Do przeprowadzonego testu użyto podejścia pętlowego, podejście rekurencyjne, które wydawałoby się tutaj intuicyjne nie sprawdza się, ze względu na to, że przez osiąganą głębokość rekurencji python wywołuje błąd. Czas wywołania metody Eulera mógłby się nieznacznie (w porównaniu do metody odeint) skrócić, jeśli kod zostałby bardziej zoptymalizowany.

1. **Wnioski**

Analiza i symulacja modeli Lotki-Volterry i układu Lorenza z wykorzystaniem metody Eulera i funkcji odeint z pakiety scipy.integrate prowadzi do następujących wniosków

* Metoda Eulera poza swoją łatwością implementacji i zrozumienia wykazuje wolniejsze oraz mniej dokładnie działanie, niż funkcja odeint. Problem metody Eulera polega na tym, że na przedziale czasowym jakim jest dt traktuje pochodną funkcji jaka stałą. Fakt ten potwierdza sytuacje widoczne na wykresach, gdzie metoda Eulera okazała się być mniej skuteczna, gdy rósł krok dt. Chcąc poprawić wykresy uzyskane tą metodą, należy zmniejszyć krok dt, czego wynikiem jest zwiększenie czasu potrzebnego na jej wywołanie.
* Błąd metody Eulera, można obliczyć używając szeregu Taylora i jest on kwadratowo proporcjonalny do dt
* Funkcja odeint, jest lepszym wyborem, do rozwiązywania równań różniczkowych, cechuje się mniejszą zależnością, zarówno czasową jak i w pewnym stopniu jakościową od kroku dt.

1. [Busłowicz, M. (2012). Analiza układu Lorenza niecałkowitego rzędu. Pomiary, Automatyka, Robotyka, 2(2012), 306.](https://www.par.pl/2012/303_306.pdf) [↑](#footnote-ref-1)