

Modele systemów dynamicznych

Oscylator Van der Pola

Przemysław Sipa

Maj 2022

1 Wstęp

Van der Pol opisał eksperyment w którym poprzez zmianę rezysora na element o nielinowej charakterystyce prądowo-napięciowej w neonowej żarówce, której źródło prądu było dostarczane w postaci sygnału sinusoidalnego, pojemność danego kondensatora powiększyła się. Owy system częstotliwości z uwagi na dostarczany sygnał tworzył dyskretne skoki o nieregularnym kształcie. Van der Pol porównał je z nieregularnym szumem słyszalnym w ówczesnych telefonach, podczas odbierania połączeń.[4] Uniwersalność modelu sprawia, że używany jest on również w medycynie - modelowanie leczenia choroby Parkinsona oraz nieprawidłowości układu krwionośnego. Dzięki zastosowaniu omawianego modelu możemy wiedzieć, czy leczenie zmniejsza negatywne skutki choroby, czy tylko je nasila. [2] Wynika to z układu równań opisujących oscylator, który jest zależny od czasu - tak jak działanie leków. Równania różniczkowe są dobrym elementem do analizy i przewidywania ww. zależności. Jednakże przed etapem implementacji, trzeba sprawdzić ich stabilność oraz współczynnik bifurkacji, czyli zmiany stanów danego układu.[5] Oscylator Van der Pola jest przykładem nielinearnego systemu chaotycznego

opisanego danym równaniem:

$$\frac{\partial^2 x}{\partial t \partial t} + \mu(x^2 - 1) \frac{\partial x}{\partial t} + x = 0, \mu > 0 \quad (1)$$

Po przekształceniu równania (1) modelu oscylatora Van der Pola można przedstawić go jako układ dwóch równań nieliniowych:

$$\begin{cases} \frac{\partial x}{\partial t} = y \\ \frac{\partial y}{\partial t} = -x + \mu \times (1 - x^2) \times y \end{cases} \quad (2)$$

gdzie:

- x, y - współrzędna położenia
- μ - parametr wskazujący na nieliniowość i siłę tłumienia[1]

Wartości początkowe[3] dla danego układu prezentują się następująco:

- $x_0 = 1$
- $y_0 = 1$
- $\mu_0 = 0.001$
- $dt = 0.0001$
- $T = 25$

Podany system dynamiczny został modelowany przy użyciu pakietów: SymPy, Scipy, NumPy, Pandas oraz Matplotlib. Porównanie rozwiązań dokładnych oraz wartości przybliżonych w tym modelu przedstawiono za pomocą metod obliczeniowych wykonyanych przez SymPy i Scipy oraz numerycznej metody Eulera, opierającej się na interpretacji geometrycznej równania różniczkowego. Metoda ta jest najprostrzym sposobem rozwiązywania równań różniczkowych. Formuła wyprowadzana jest z rozwinięcia Taylora, danego wzorem dla każdego punktu z przedziału (a,b):

$$\sum_{k=0}^n \left(\frac{(x-a)^k}{k!} f^{(k)}(a) \right) + R_n(x, b) \quad (3)$$

Aby skorzystać z tej metody przybliżamy pochodną czasową do ilorazu różnicowego

$$\frac{\partial x(t)}{\partial t} \approx \frac{x(t+h) - x(t)}{h} \quad (4)$$

Przekształcając równanie stronami otrzymujemy

$$x(t+h) \approx x(t) + h \times \frac{\partial x(t)}{\partial t} \quad (5)$$

$$x(t+h) \approx x(t) + h \times f(t, x(t)) \quad (6)$$

Całość można zapisać jako:

$$\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + h \times (f(t_i, y_i)) \end{cases} \quad (7)$$

Gdzie:

- h - krok całkowania
- y_{i+1} - rozwiązanie
- y_i - rozwiązanie w kroku poprzednim
- f - funkcja równania różniczkowego

2 Modele

2.1 Model oscylatora dla wartości początkowych

Rozwiązanie układu dla wartości początkowych

Kod przedstawiający implementację układu:

```
1 x0=1
2 y0=1
3 mu=0.001
4 time = np.arange(0,25,0.0001)
5 wart_poczatkowe = [x0,y0]
```

```

6 def oscylator_differential_equation(wart_poczatkowe,t):
7     dxdt = wart_poczatkowe[1]
8     dydt = -wart_poczatkowe[0]+mu*(1-wart_poczatkowe[0]**2)*wart_poczatkowe[1]
9     return dxdt,dydt
10 solution = odeint(oscylator_differential_equation,wart_poczatkowe,time)

```

2.2 Model oscylatora ze zmiennym parametrem x

Model zgodny z wartościami początkowymi, parametru $mu = 3$ (większy parametr mu został wybrany w celu ukazania zmienności systemu) Ustalamy wartość parametru x :

(a) $x = -1$

(b) $x = 0$

(c) $x = 2$

(d) $x = 3$

(e) $x = 4$

(f) $x = 5$

2.3 Model oscylatora ze zmiennym parametrem y

Ustalamy wartość parametru y

(a) $y = -1$

(b) $y = 0$

(c) $y = 4$

(d) $y = 5$

2.4 Model oscylatora ze zmiennym parametrem μ

Ustalamy wartość parametru μ :

(a) $\mu = 0.01$

(b) $\mu = 0.1$

(c) $\mu = 1$

(d) $\mu = 3$

2.5 Rozwiążanie modelu dla zmiennego parametru kroku symulacji

Ustalamy wartość parametry kroku symulacji:

(a) $dt = 0.3$

(b) $dt = 0.9$

(c) $dt = 2$

(d) $dt = 5$

2.6 Model oscylatora na przestrzeni danych T

(a) $T < t_0$ dla $\mu = [0.001, 3]$

(b) $T > t_0$ dla $\mu = [0.001, 0.1, 3]$

3 Wyniki

3.1 Wyniki dla wartości początkowych

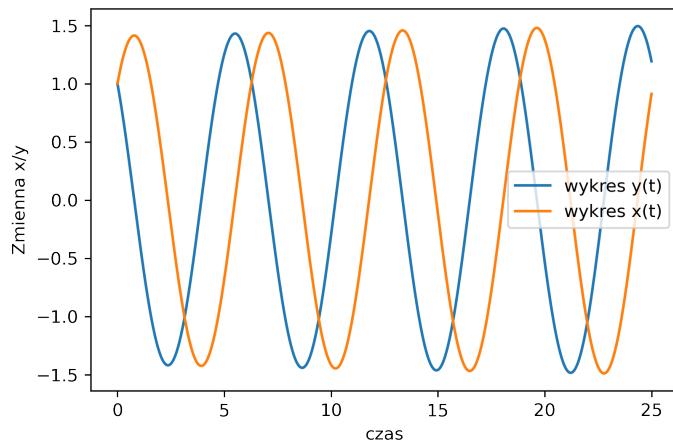


Figure 1: Wykres przebiegu funkcji $y(t)$ oraz $x(t)$ dla wartości początkowych

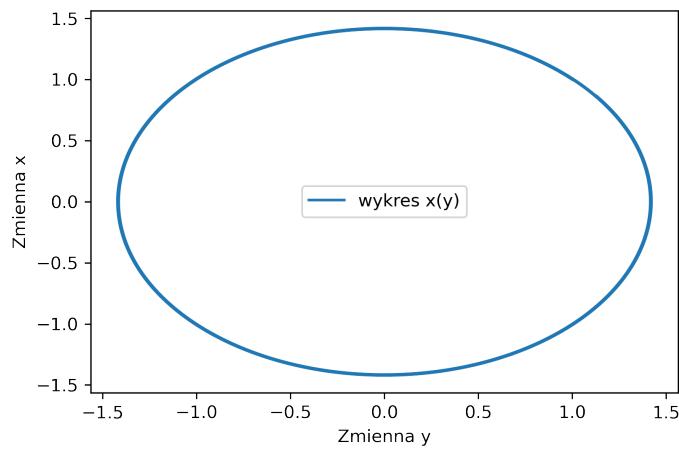


Figure 2: Wykres przedstawiajcy przebieg funkcji $x(y)$ dla wartosci poczatkowych

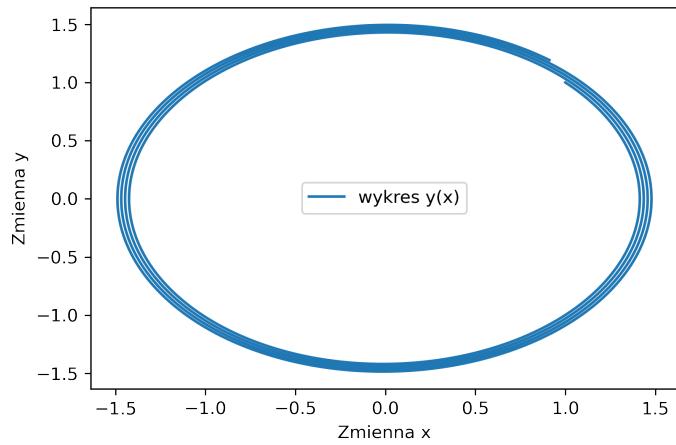


Figure 3: Wykres przedstawiajcy przebieg funkcji $y(x)$ dla wartosci poczatkowych

3.2 Wyniki dla zmiennego parametru x

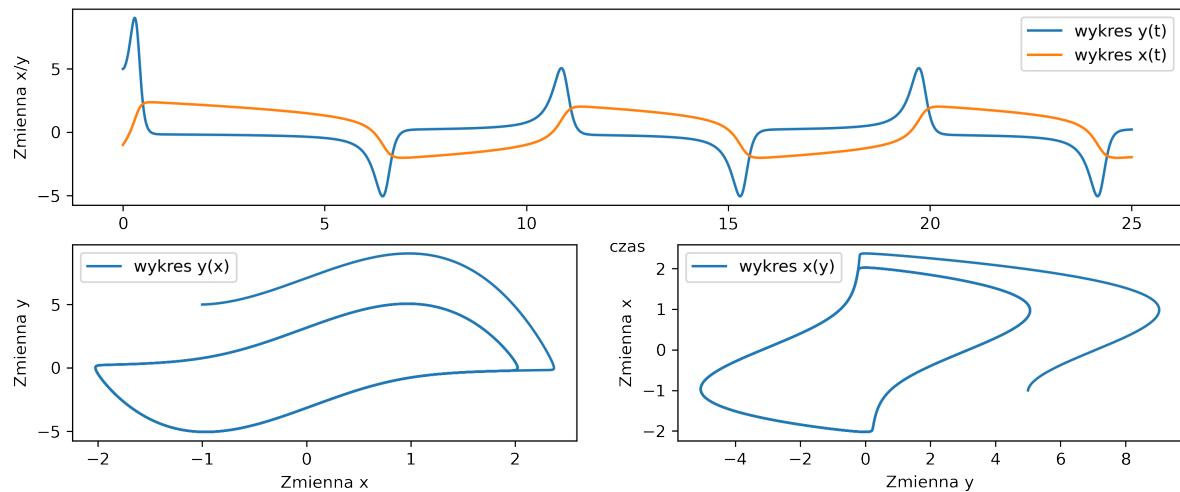


Figure 4: Wykres funkcji dla parametru $x = -1$

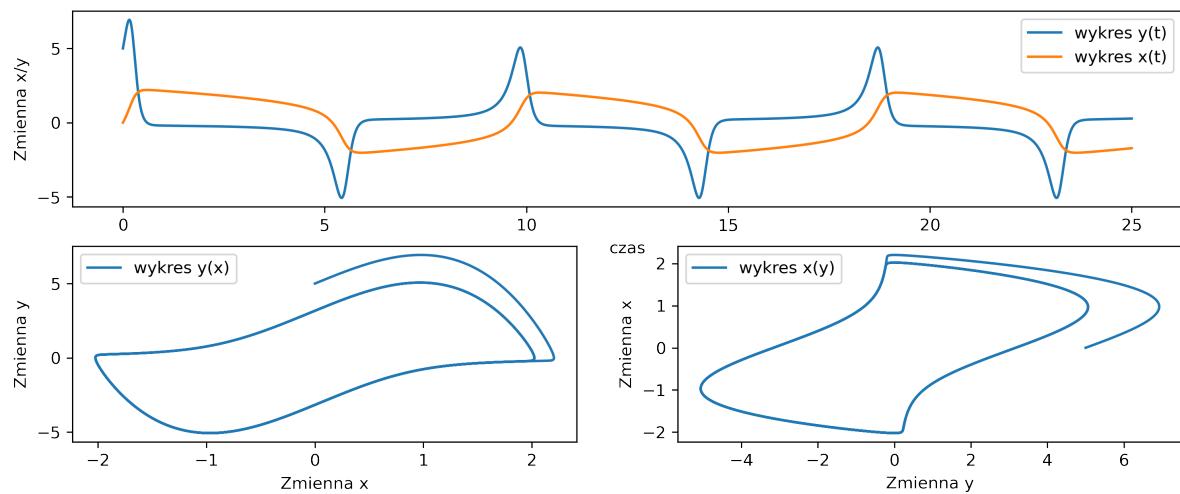


Figure 5: Wykres funkcji dla parametru $x = 0$

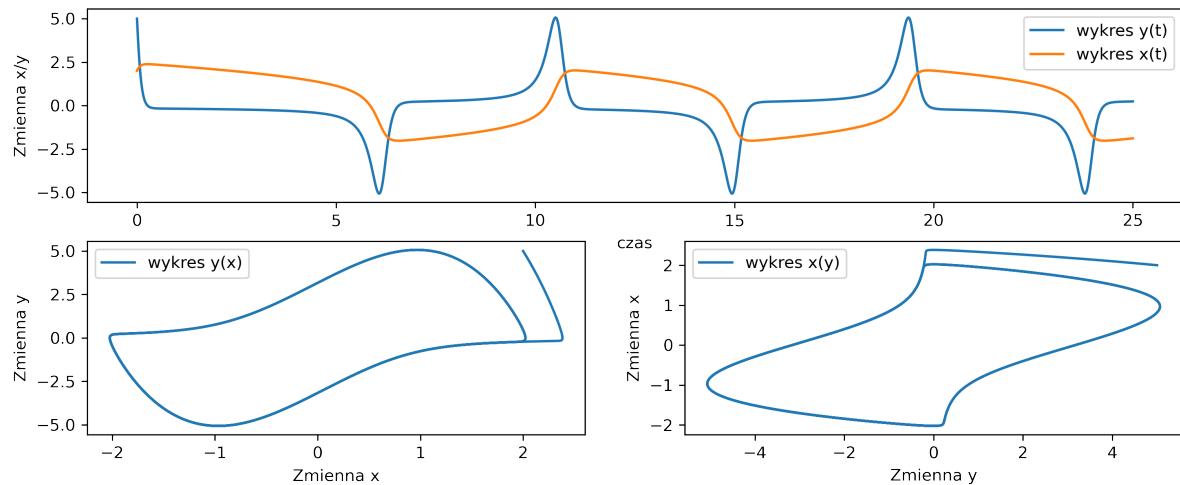


Figure 6: Wykres funkcji dla parametru $x = 2$

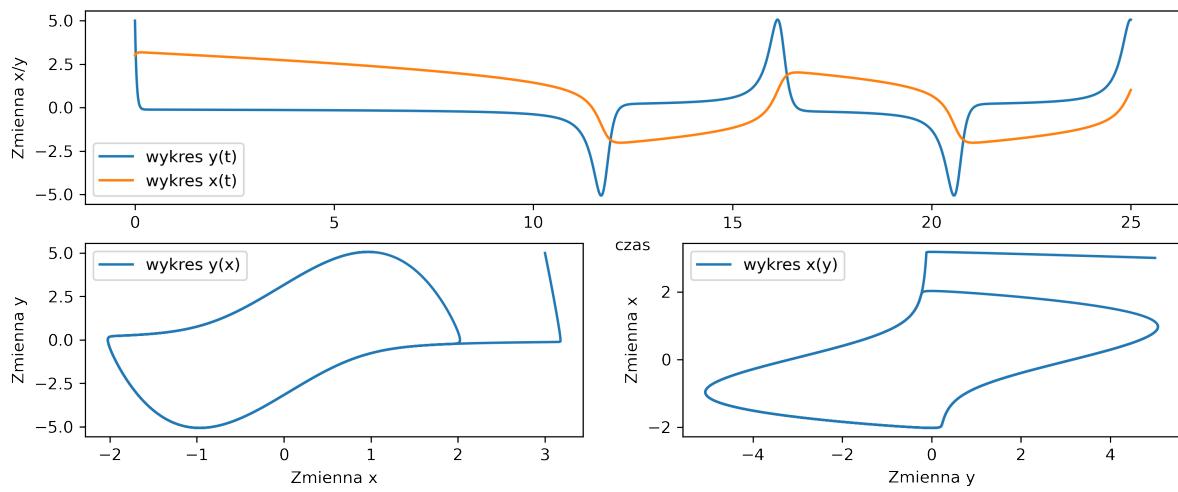


Figure 7: Wykres funkcji dla parametru $x = 3$

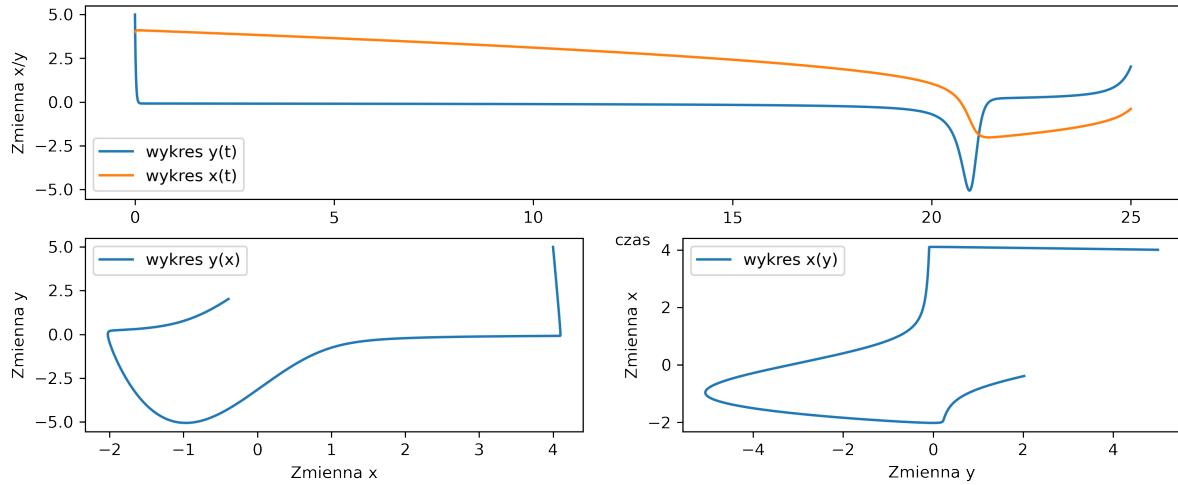


Figure 8: Wykres funkcji dla parametru $x = 4$

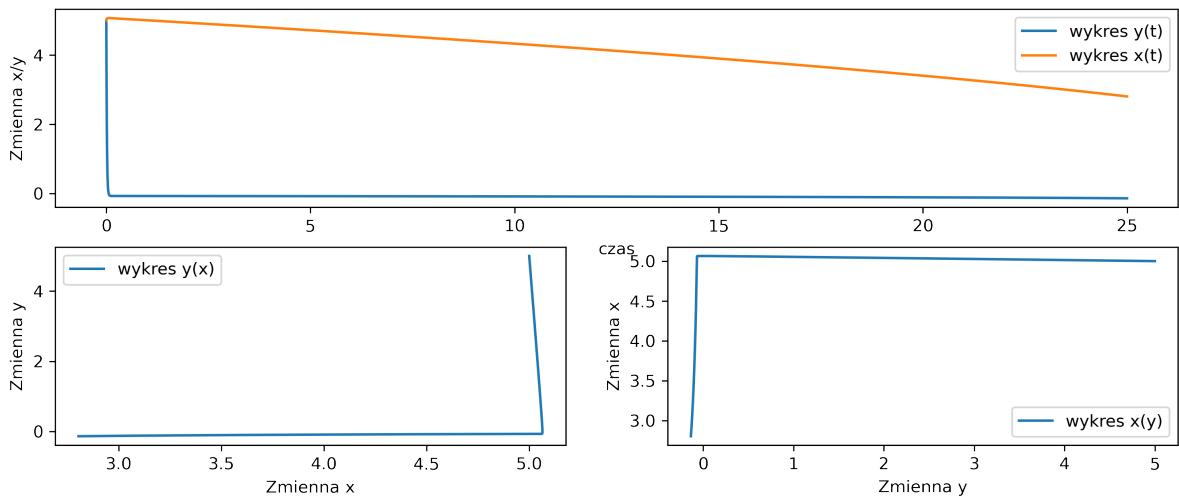


Figure 9: Wykres funkcji dla parametru $x = 5$

3.3 Wyniki dla zmiennego parametru y

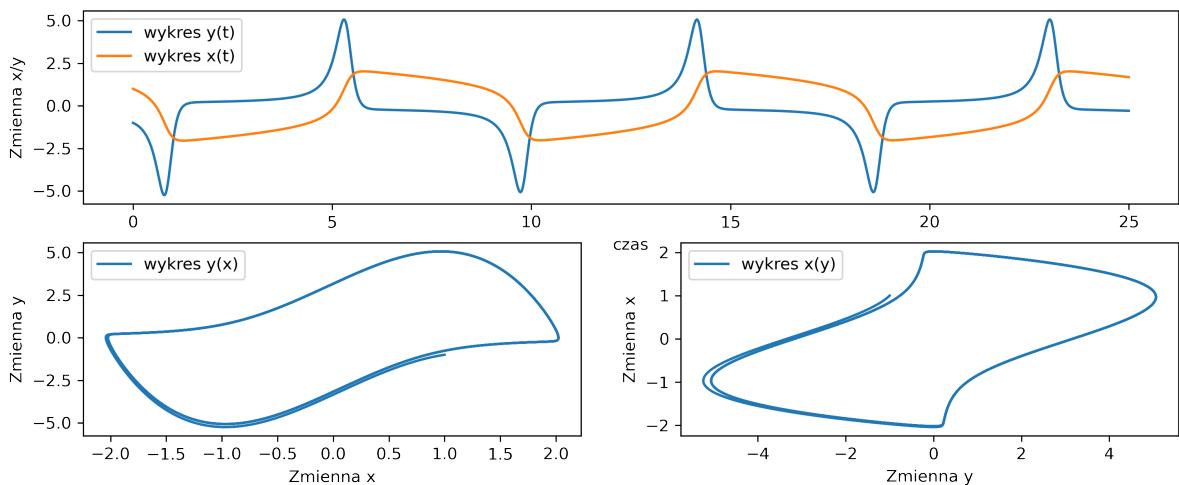


Figure 10: Wykres funkcji dla parametru $y = -1$

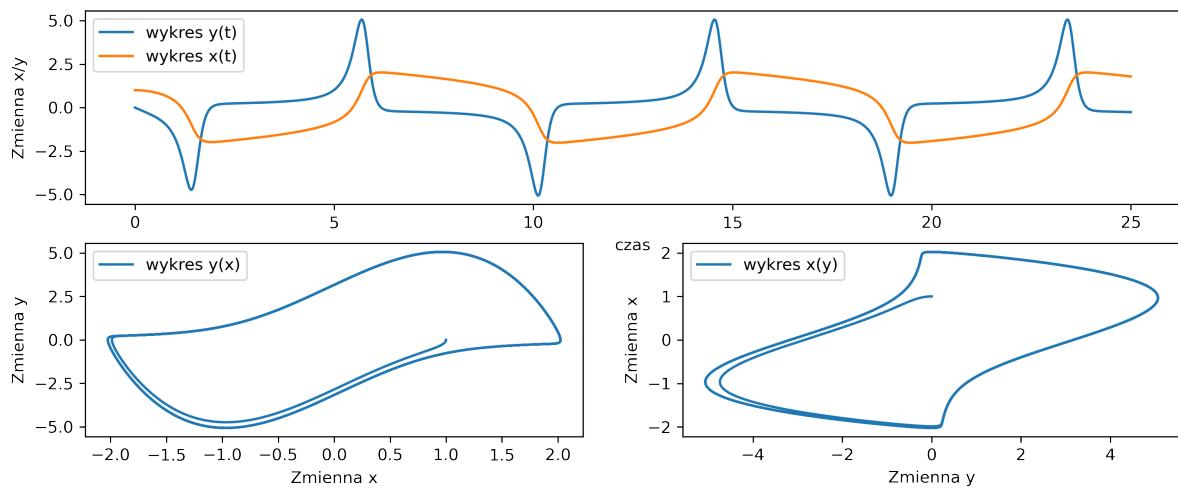


Figure 11: Wykres funkcji dla parametru $y = 0$

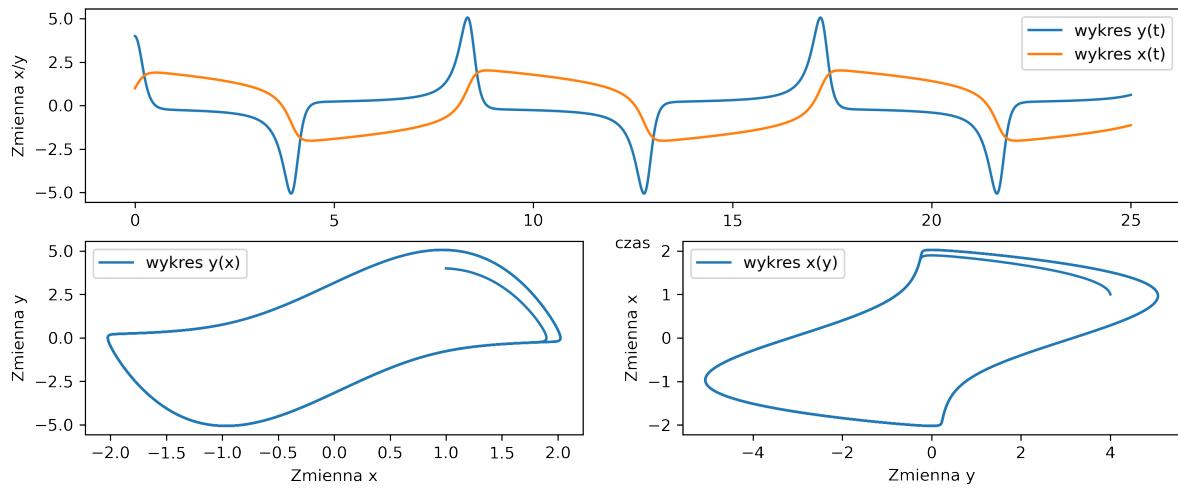


Figure 12: Wykres funkcji dla parametru $y = 4$

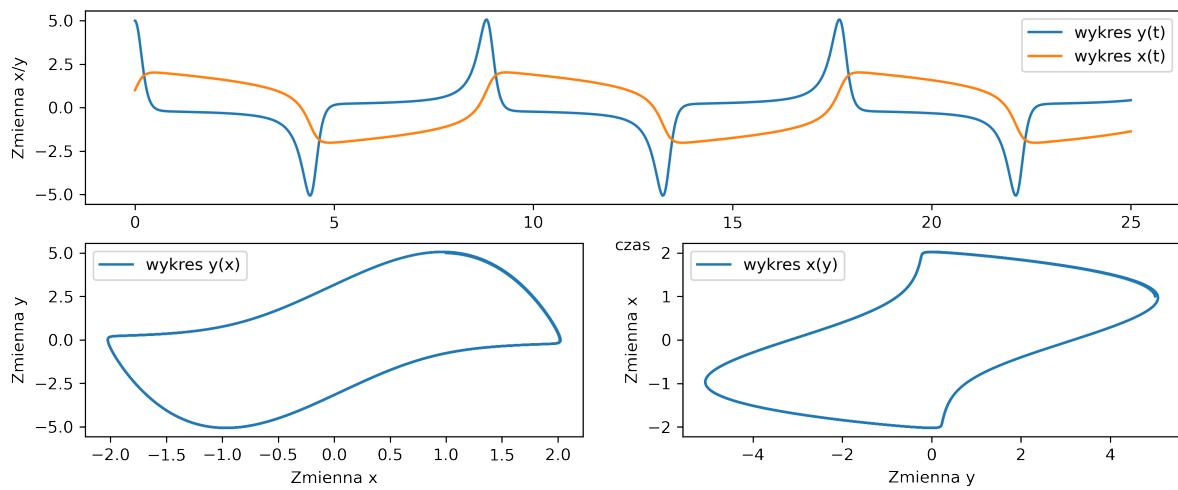


Figure 13: Wykres funkcji dla parametru $y = 5$

3.4 Wyniki dla zmiennego parametru μ

Wykres funkcji dla parametru $\mu = 0.01$

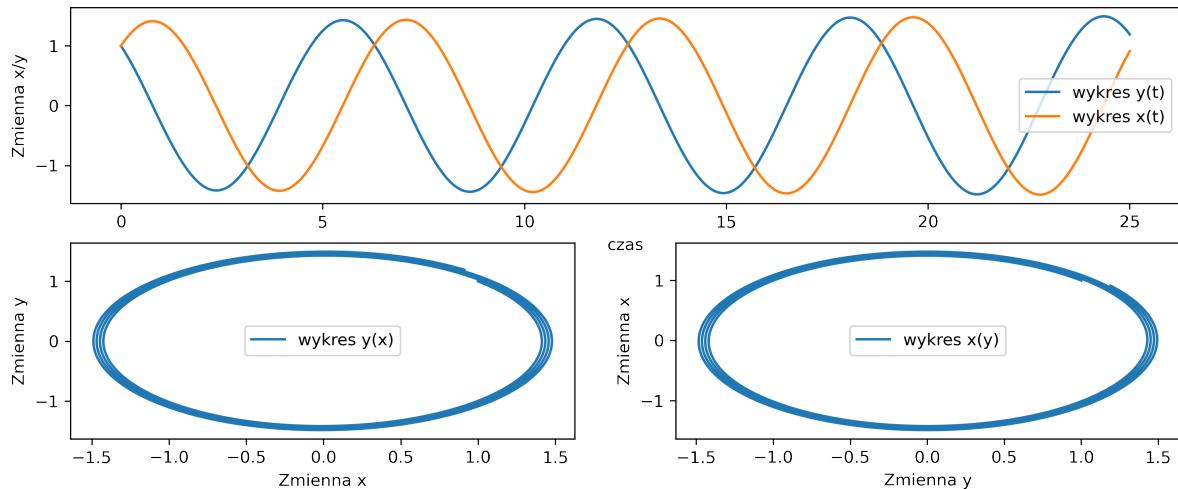


Figure 14: Wykres funkcji dla parametru $\mu = 0.01$

Wykres funkcji dla parametru $\mu = 0.1$

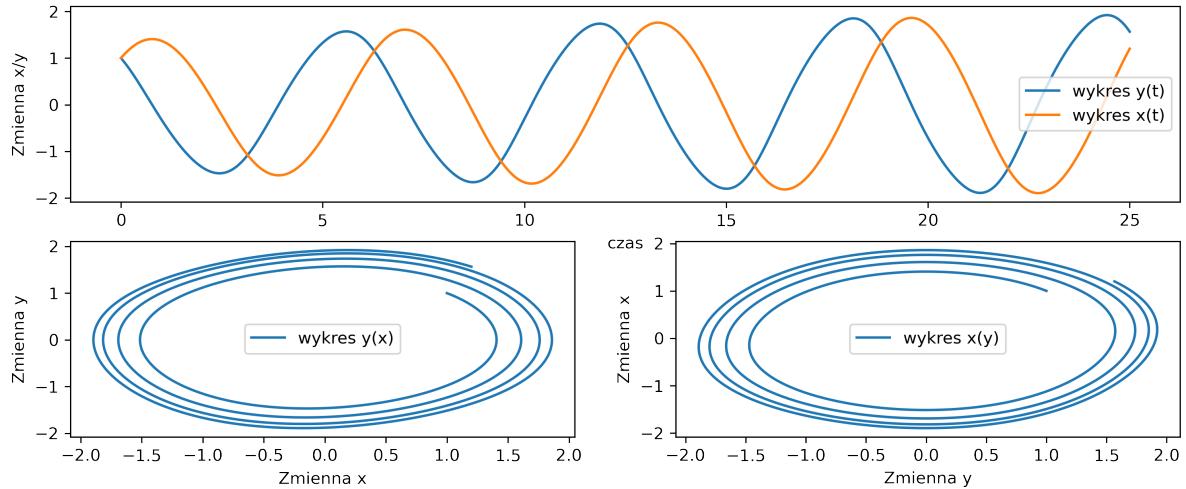


Figure 15: Wykres funkcji dla parametru $\mu = 0.1$

Wykres funkcji dla parametru $\mu = 1$

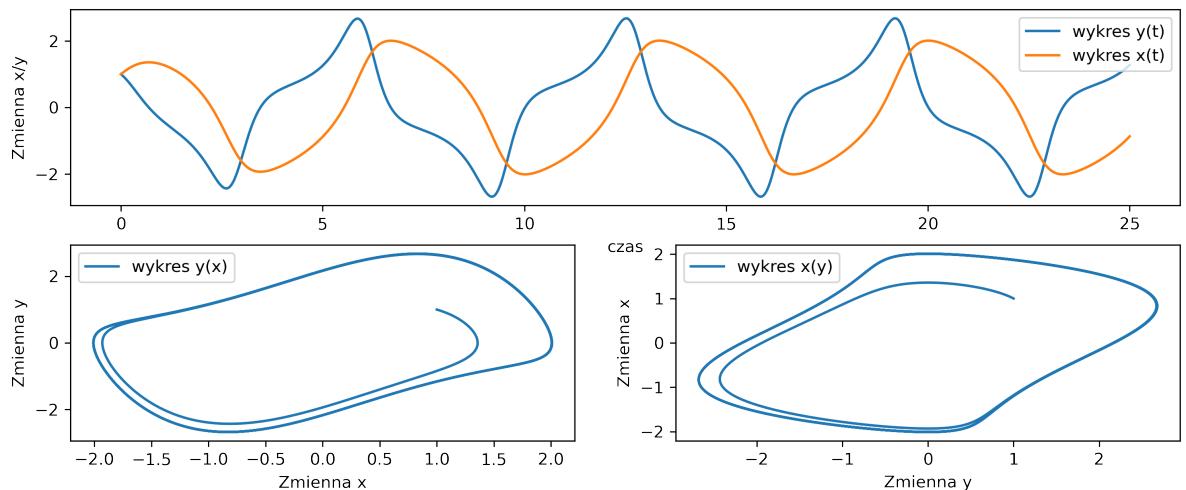


Figure 16: Wykres funkcji dla parametru $\mu = 1$

Wykres funkcji dla parametru $\mu = 3$

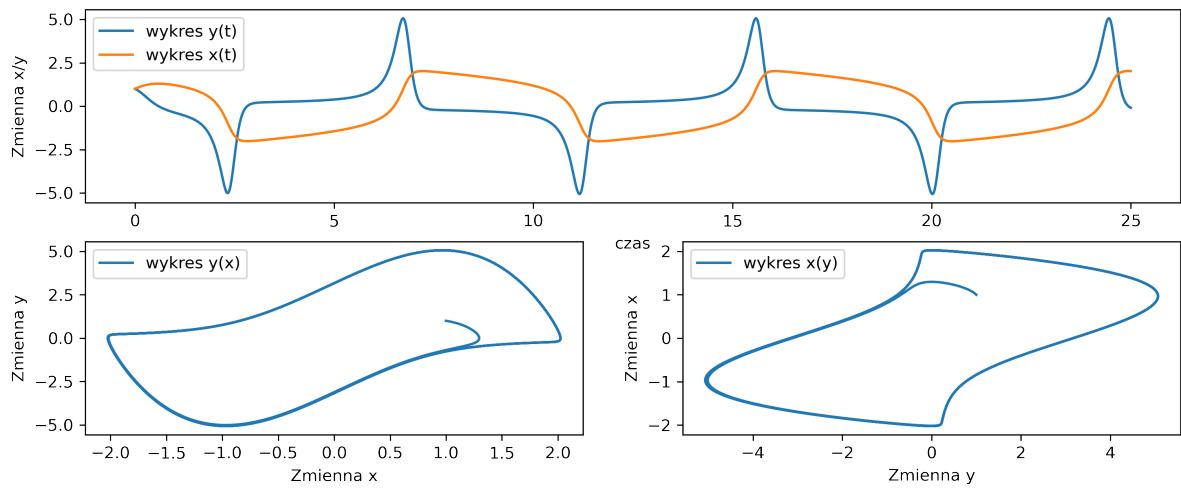


Figure 17: Wykres funkcji dla parametru $\mu = 3$

3.5 Wyniki dla zmiennego parametru kroku symulacji

3.5.1 Rozwiążanie dla solvera ODEINT

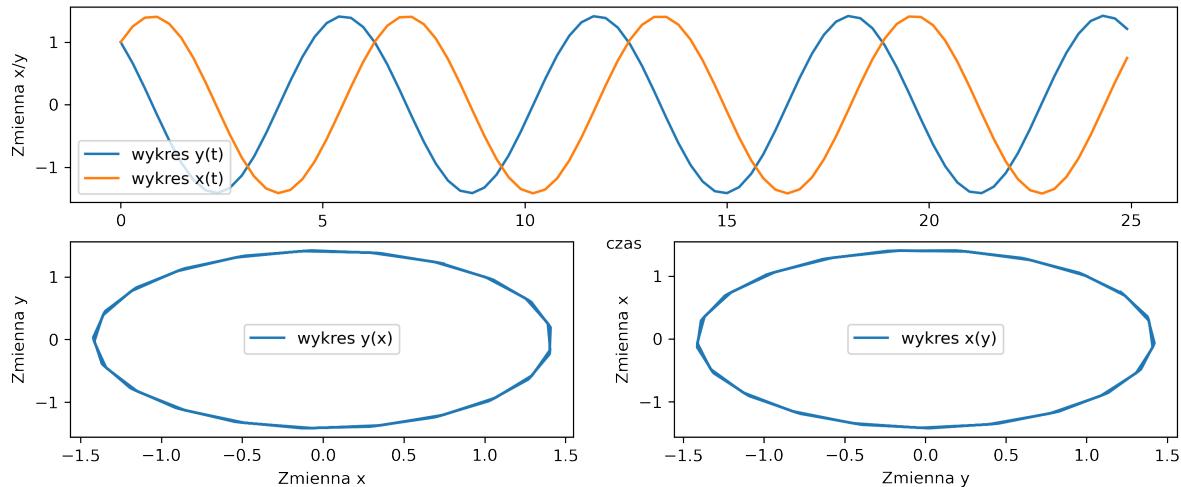


Figure 18: Wykres funkcji dla parametru $dt = 0.3$

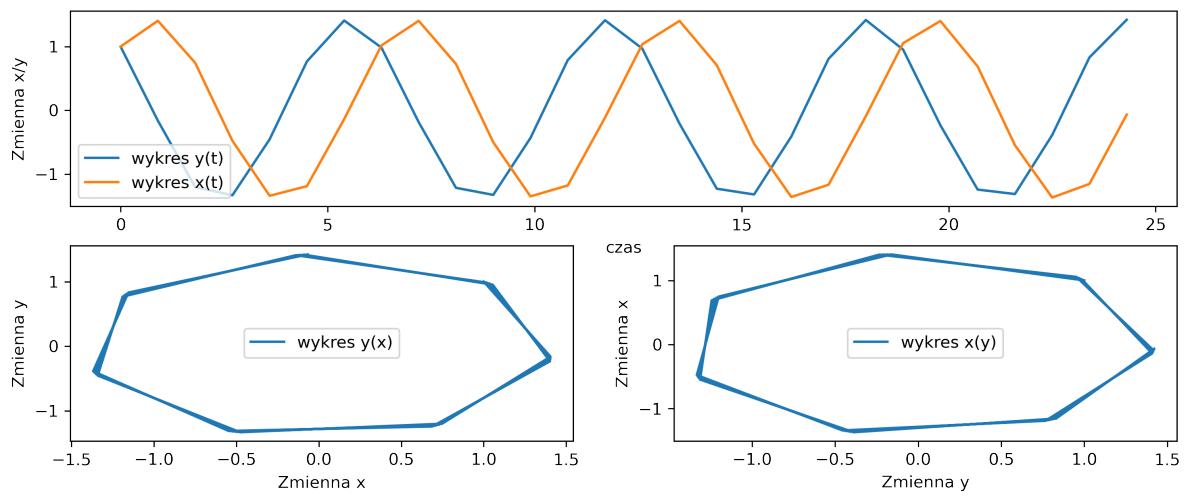


Figure 19: Wykres funkcji dla parametru $dt = 0.9$

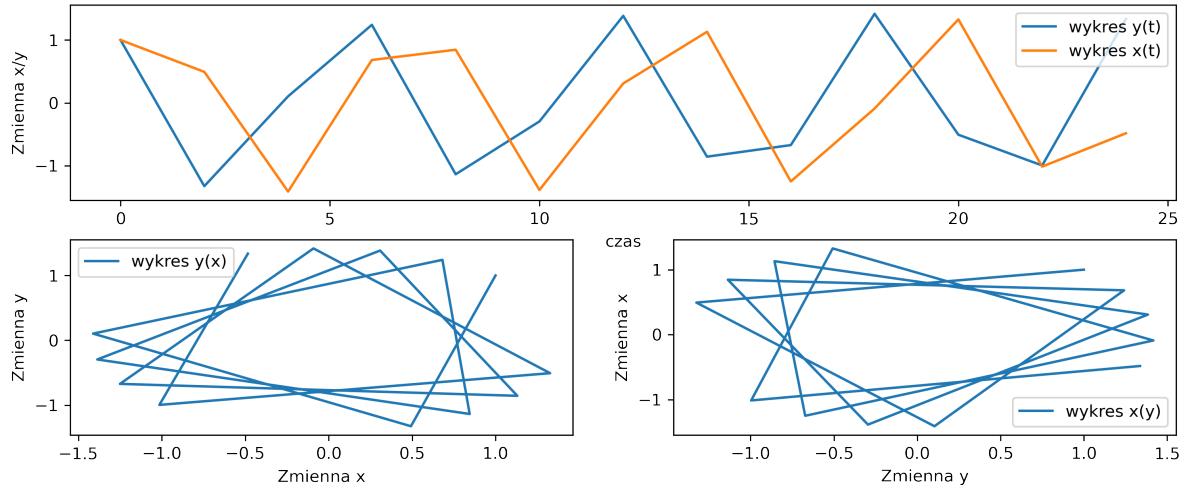


Figure 20: Wykres funkcji dla parametru $dt = 2$

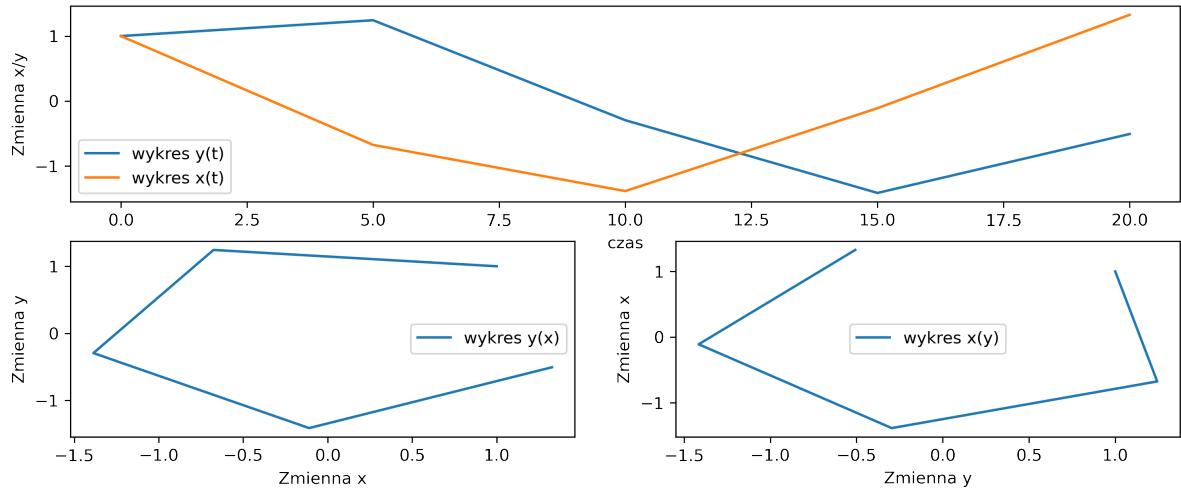


Figure 21: Wykres funkcji dla parametru $dt = 5$

3.5.2 Rozwiążanie dla metody Eulera

Kod bazowy dla metody Eulera

```

1 x0=1
2 y0=1
3 mu=0.001
4 time = np.arange(0,25,0.1)
5 wart_poczatkowe = [x0,y0]
6 model = oscylator_differential_equation(wart_poczatkowe,time)
7
8 def metoda_eulera_model(x0,y0,mu,time):
9     x = np.zeros(len(time))
10    y= np.zeros(len(time))
11    x[0]=x0
12    y[0]=y0
13
14    for i in range(0,len(time)-1):
15        dt = time[i+1]-time[i]
16        cur_model = oscylator_differential_equation([x[i],y[i]],dt)
17        x[i+1]= x[i]+cur_model[0]*dt
18        y[i+1]= y[i]+cur_model[1]*dt
19    return x,y

```

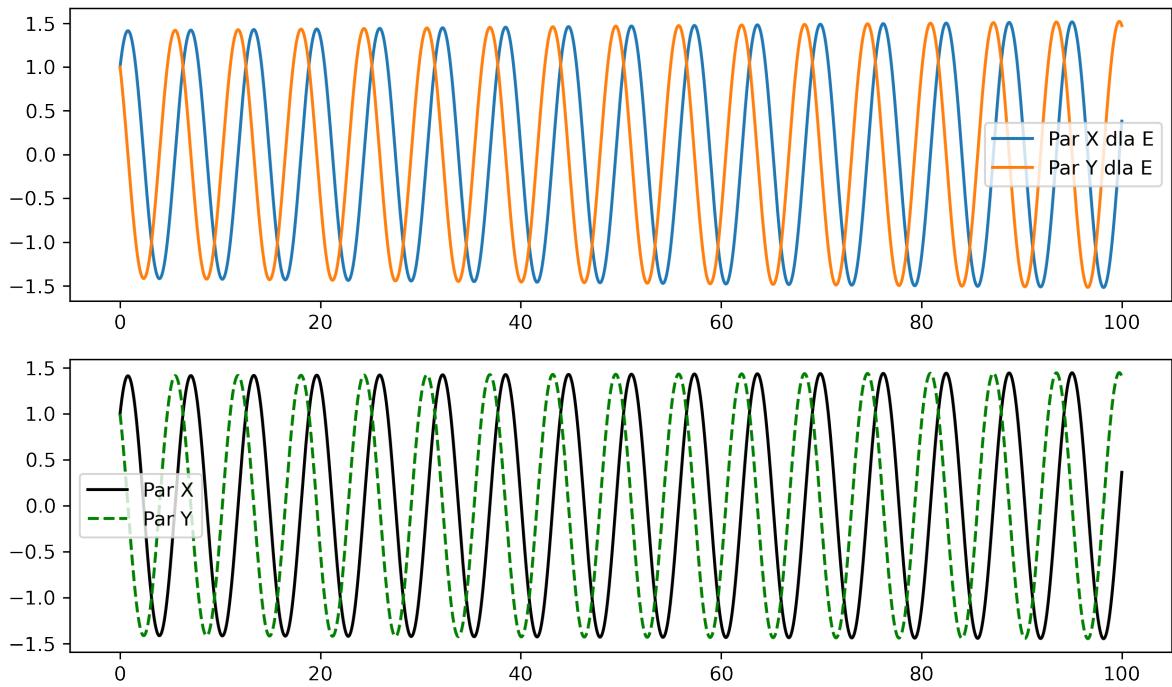


Figure 22: Funkcje dla parametru $dt = 0.001$ dla metody ODEINT oraz metody Eulera

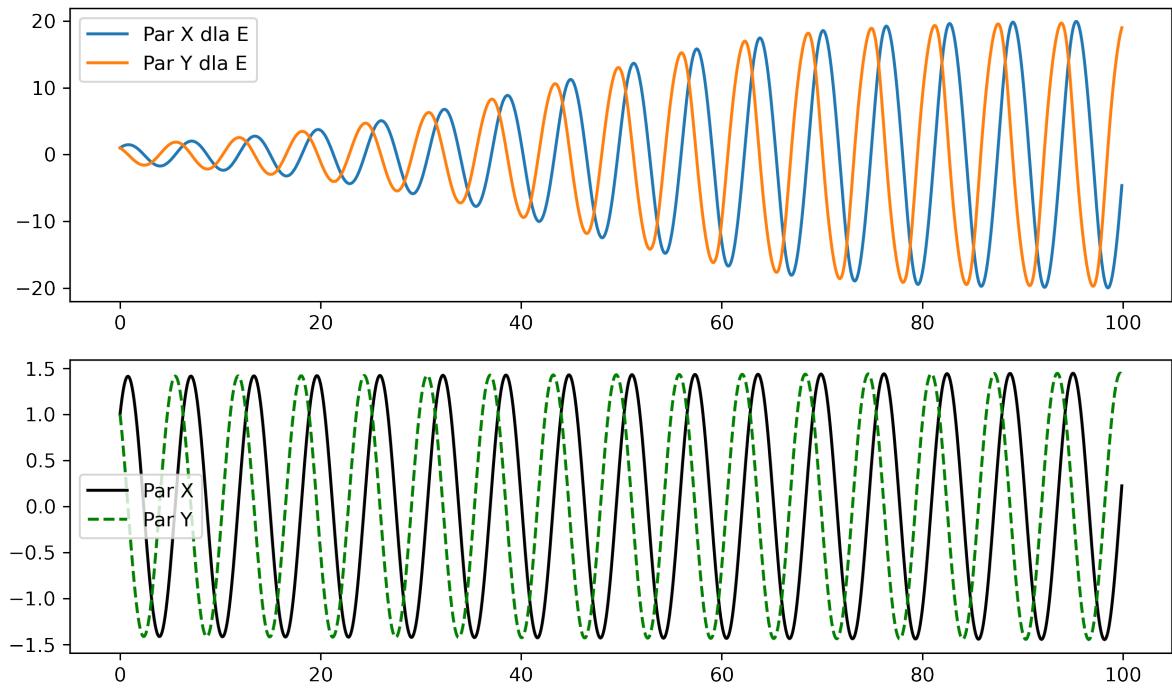


Figure 23: Funkcje dla parametru $dt = 0.1$ dla metody ODEINT oraz metody Eulera

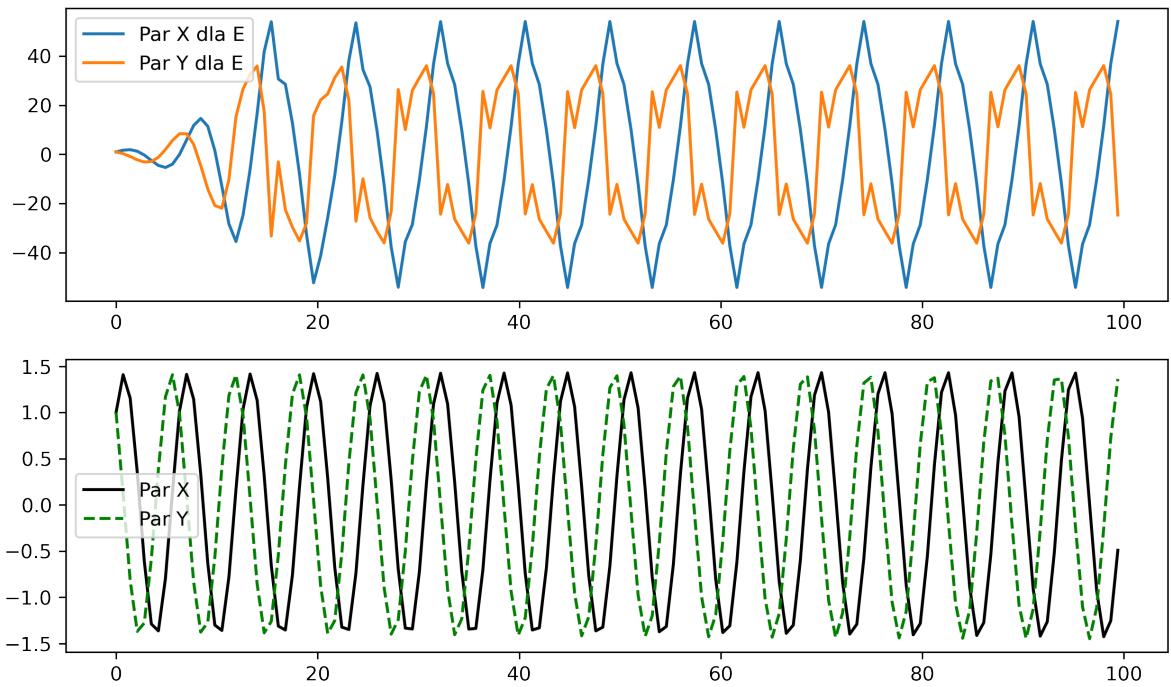


Figure 24: Funkcje dla parametru $dt = 0.7$ dla metody ODEINT oraz metody Eulera

3.5.3 Porównanie wyników metody ODEINT oraz metody Eulera

Porównanie wyników końcowych zostanie przedstawione przy użyciu średniego błędu kwadratowego oraz średniego błędu bezwzględnego.

Średni błąd kwadratowy definiowany jest jako:

$$\frac{1}{n} \sum_{i=1}^n (p(x_i) - f_i)^2 \quad (8)$$

gdzie:

- x - argument położenia
- $p(x)$ - wartość funkcji aproksymującej
- $f(x)$ - wartość funkcji aproksymowanej
- n - ilość elementów w przedziale

Średni błąd bezwzględny definiowany jest jako

$$\frac{\sum_{i=1}^n |(p(x_i) - f(x_i)|}{n} \quad (9)$$

gdzie parametry są zgodne z równaniem (8)

Statystyki dla metod obliczeniowych

Kod w Pythonie:

```

1 tab = [0.001,0.1,0.3,0.4,0.7]
2 for j in tab:
3     x0=1
4     y0=1
5     mu=0.001
6     time = np.arange(0,100,j)
7     wart_poczatkowe = [x0,y0]
8     model = oscylator_differential_equation(wart_poczatkowe,time)
9     odeint_model = odeint(oscylator_differential_equation,wart_poczatkowe,time).T
10    metoda_eulera_wartosci_x = metoda_eulera_model(x0,y0,mu,time)[0].mean()
11    metoda_eulera_wartosci_y = metoda_eulera_model(x0,y0,mu,time)[1].mean()
12    odeint_wartosci_x = odeint_model[0].mean()
13    odeint_wartosci_y = odeint_model[1].mean()
14    blad_kwadratowy = (odeint_wartosci_x - metoda_eulera_wartosci_x)**2
15    blad_bezwzgledny = abs((odeint_wartosci_x - metoda_eulera_wartosci_x))

```

Table 1: Tabela porównująca średnie błędy danych metod obliczeniowych

Krok symulacji	Średni błąd kwadratowy	Średni błąd bezwzględny
0.001	0.00	0.00
0.1	0.03	0.18
0.3	0.09	0.29
0.4	0.06	0.25
0.7	0.06	0.25

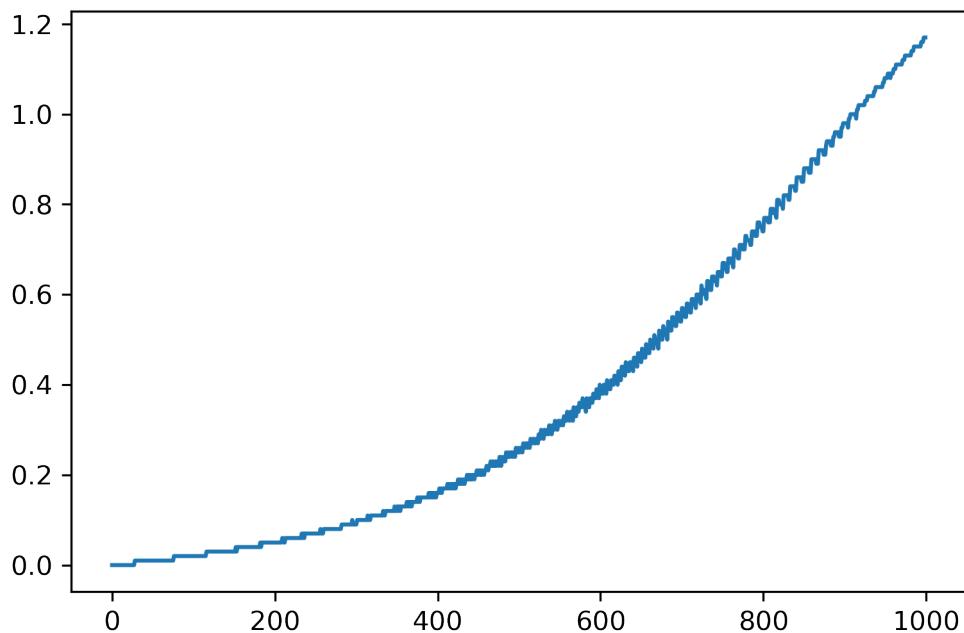


Figure 25: Wykres błędu aproksymacji między metodami dla 1000 prób, przy $dt = (0.001; 0, 7)$ oraz $T = [0; 1000]$

3.6 Wyniki dla zmienionej wartości parametru T

(a) $T < t_0$ dla $\mu = [0.001, 3]$

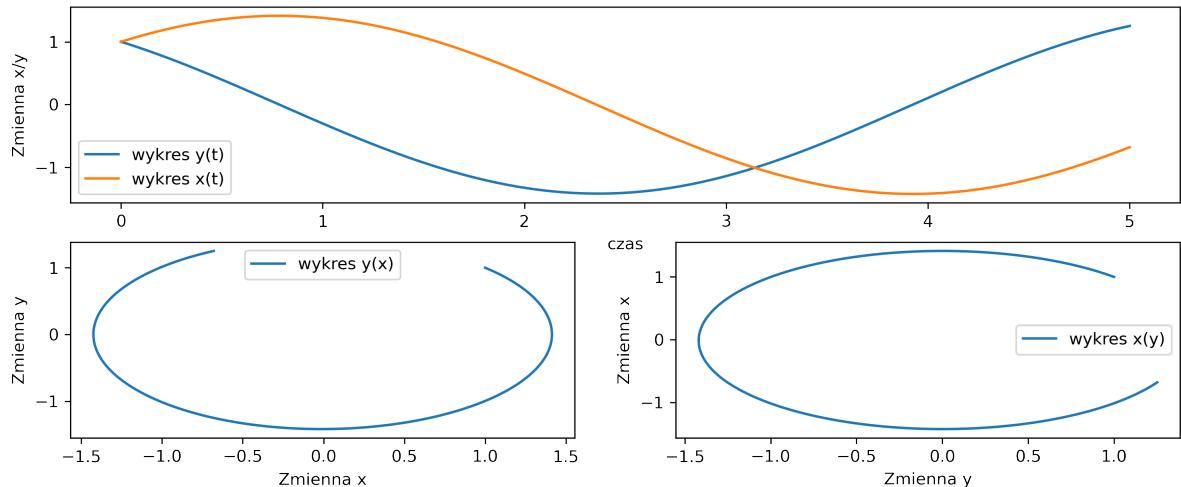


Figure 26: Wykres funkcji dla parametru $T = 5$

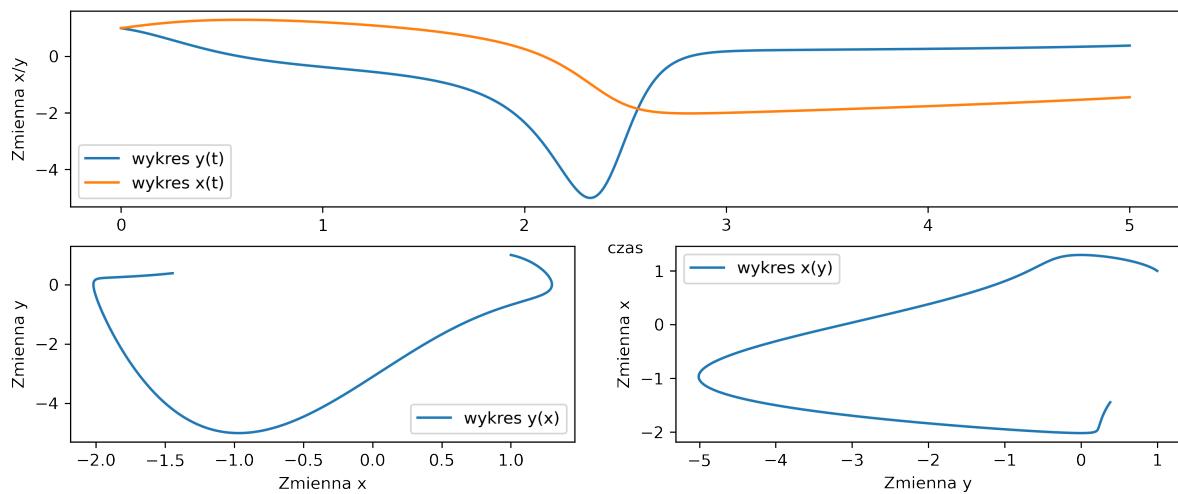


Figure 27: Wykres funkcji dla parametru $T = 5$ oraz $\mu = 3$

(b) $T > t_0$ dla $\mu = [0.001, 3]$

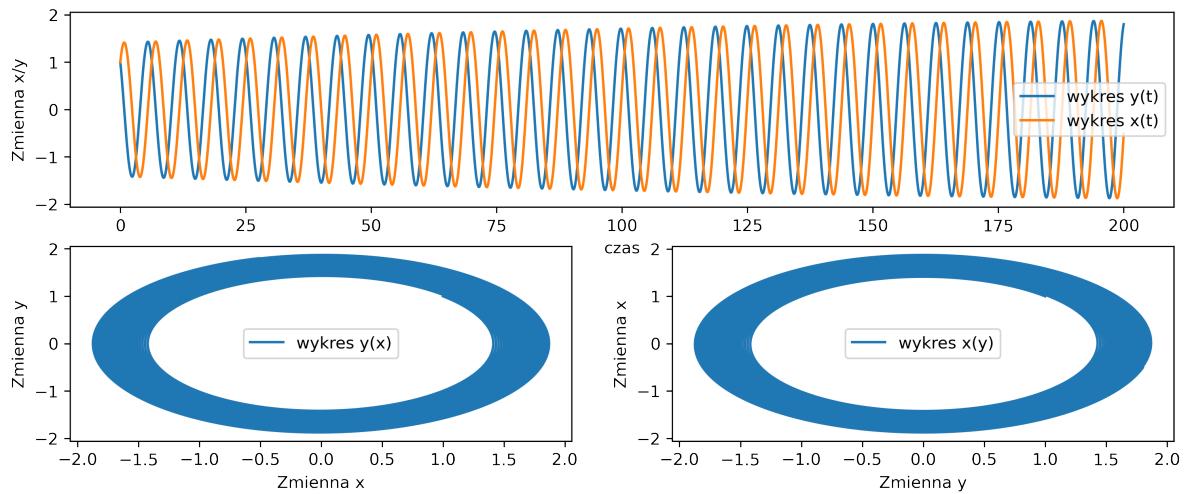


Figure 28: Wykres funkcji dla parametru $T = 200$

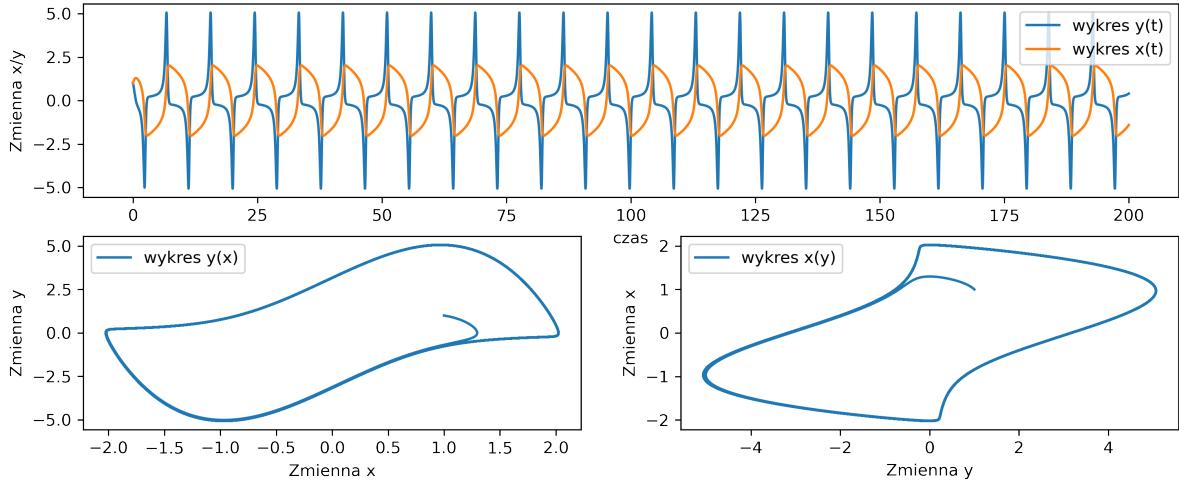


Figure 29: Wykres funkcji dla parametru $T = 200$ oraz $\mu = 3$

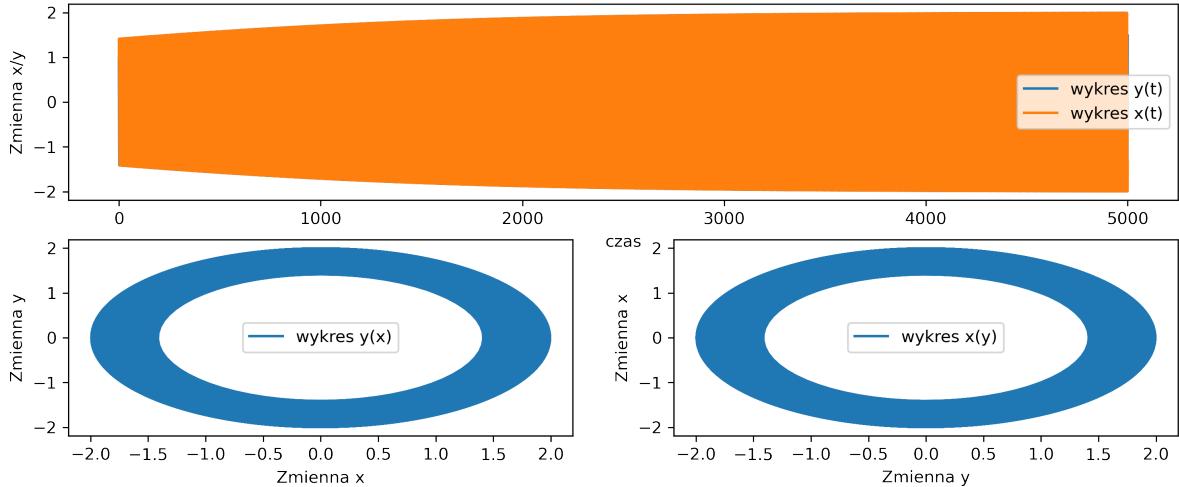


Figure 30: Wykres funkcji dla parametru $T = 5000$ oraz $\mu = 0.1$

4 Wnioski

System jest stabilny, jeśli przy niewielkiej zmianie parametrów zachodzą niewielkie zmiany w trajektoriach. Im większy współczynnik tłumienia μ , tym jest większe prawdopodobieństwo wystąpienia zjawiska bifurkacji. Przedstawione elementy wraz z upływem czasu mogą tracić swoje pierwotne właściwości, przez co przebieg wykresu się

zmienia. Przy większej zmianie parametrów, czyli poprzez zmianę wartości początkowych w części jedności, zauważymy drastyczne zmiany w przebiegu omawianego systemu. Obiekt wtedy staje się wrażliwy na jakiekolwiek zmiany w obrębie jego parametrów. Początkowo wykresy $x(t)$ oraz $y(t)$ przyjmowały wartości powtarzalne, przypominały rozciągniętą wersję wykresu sinusa i cosinusa, zaś wykres $x(y)$ oraz $y(x)$ przypominał oval.

Zmieniając parametr x w równaniu różniczkowym zauważymy drastyczną zmianę w przebiegu wykresu funkcji, funkcje zależne od czasu, nadal mają stałą powtarzalność, okresowość, jednakże nie przypominają wykresu funkcji sinus, lecz ukazują przebieg pewnych funkcji niemalejących, które co dany okres się powtarzają. Dla pozostałych wykresów, zmiany były również dotkliwe, wykres $x(y)$ staje się symetryczny względem wykresu $y(x)$, gdy punktem symetri będzie początek układu współrzędnych, punktu $(0,0)$. Zwiększając wartość parametru x zauważymy zjawisko linearyzacji wykresu, przebieg staje się bardziej spokojny. Dla parametrów $x > 5$ wykresy dążą do liniowości, gdzie dla $x = 100$ wszystkie trzy wykresy osiągają wartości stałe. Zmiany dotyczące parametru y opisują podobne zależności, zauważone w powyższym przypadku, jednakże, zmiany te zachodzą powoli, a zjawisko linearyzacji występuje jedynie dla funkcji powiązanych z czasem. Pozostałe funkcje dynamicznie zmieniają swoje przebiegi, dając do funkcji rosnących/malejących.

Parametr μ jest najwrażliwszą częścią omawianego systemu, niewielkie zmiany tego parametru powodowały zauważalne zmiany w obrębie wykresu. Wraz ze wzrostem danego parametru, dynamika zmian malała, uśredniony przebieg funkcji dla zmiennego parametru μ ukazany jest dla wartości 3, której to wykres podobny jest do wszystkich innych wykresów, w których zmieniono parametr $\mu \in [0.01; 12]$. Powyżej danego parametru funkcja linearyzuje się.

Zmiana parametru kroku symulacji została obliczona przy użyciu solvera ODEINT oraz metody Eulera dla czasu z przedziału $[0; 100]$ Ogólnym wnioskiem wynikającym z przeprowadzonych symulacji jest fakt, że zwiększenie parametru dt powoduje utworze-

nie mniej dokładnego rozwiązania danego układu. Zastosowanie metody Eulera skutkuje powstaniem niedokładnych wykresów układu oscylatora, wynika to z niskiej stabilności danej metody, przez co generuje błędy. Dla małego kroku symulacji $dt = 0.001$, metoda Eulera oraz metoda ODEINT, identycznie obliczyły przebieg omawianych wykresów. Zwiększając krok symulacji, na początku obliczeń metoda jest wysoce niestabilna, zauważalne jest to na wykresie, gdzie dla $T \in (0, 60)$ funkcja systematycznie rośnie, aż dla $T > 60$ przyjmuje przebieg zbliżony do przebiegu wygenerowanego metodą ODEINT. Opisywana niestabilność nasila się wraz z wyborem większego kroku symulacji, przebiegi wykresu zmieniają się diametralnie. Powyżej parametru $dt > 0.7$ metoda Eulera nie radzi sobie z obliczeniami dla danego układu, gdyż wartości obliczanych parametrów dążą do $+\infty$.

Porównanie działania obu metod zostało ograniczone do małych kroków symulacji $dt \leq 0.7$, w celu wyeliminowania błędów obliczeniowych. Dla stworzonej próby 1000 rozwiązań utworzyłem wykres przedstawiający wzrost błędu aproksymacji. Początkowo można powiedzieć, że wykres jest eksponencjalny, jednakże przedstawiają on przebieg funkcji niemalejącej na przedziale $T \in [0; 25]$.

Zmiana wartości czasu, ukazuje większy/mniejszy zakres przebiegu funkcji w zależności od dobranego parametru T . Wybranie większego parametru czasu, powoduje zwiększenie czasu potrzebnego do obliczenia danego rozwiązania. Dla parametru $T = 5000$ zauważymy pewną zależność, która nie jest dostrzegana dla mniejszej jednostki czasu - funkcja z każdym przebiegiem rośnie, dopiero przy użyciu kilkuset razy większego czasu, widzimy dynamikę wzrostu danego systemu.

References

- [1] Martha L. Abell and James P. Braselton. Chapter 6 - systems of differential equations. In Martha L. Abell and James P. Braselton, editors, *Introductory Differential Equations (Fourth Edition)*, pages 277–364. Academic Press, Boston, fourth edition edition, 2014.

- [2] M. A. Elfouly and M. Sohaly. Van der pol model in two-delay differential equation representation. *Scientific Reports*, 12:2925, 02 2022.
- [3] P. Jendykiewicz. Równanie van der pola. page 5, 2009.
- [4] M. Kennedy and L. Chua. Van der pol and chaos. *IEEE Transactions on Circuits and Systems*, 33(10):974–980, 1986.
- [5] Fathalla A Rihan, Cemil Tunc, SH Saker, Shanmugam Lakshmanan, and R Rakkiyappan. Applications of delay differential equations in biological systems, 2018.