Odwzorowanie logistyczne

K. K.,[*] P. S.,[†] M. B.[‡]

20 listopada 2020

Spis treści

[*]Politechnika Krakowska im. Tadeusza Kościuszki w Krakowie

[†]Politechnika Krakowska im. Tadeusza Kościuszki w Krakowie

[‡]Politechnika Krakowska im. Tadeusza Kościuszki w Krakowie

1 Wstęp

Odwzorowanie logistyczne stosowane jest do tworzenia modeli demograficznych. Spopularyzowane zostało w roku 1976 przez biologa Roberta Maya, jako model demograficzny w sposób analogiczny do równania logistycznego Pierre'a Fracois Verhulsta.

Zapis matematyczny odwzorowwania logistycznego przedstawiamy w następujący sposób:

X(N+1) = (KXn)(1-Xn)

W tym równaniu x_n jest liczbą w przedziale (0,1) jako reprezentacja stosunku istniejącej populacji do jej maksymalnej wartości. Parametr k mieści się w przedziale (0,4). To równanie nieliniowej różnicy ma na celu ukazanie dwóch efektów:

- Pierwszym jest efekt reprodukcji. Zachodzi kiedy populacja będzie rosnąć w proporcjonalnym tempie do jej obecnej ilości w przypadku kiedy jest ona niewielka.

- Drugim efektem jest wymieranie (zależne od gęstości). W przypadku tego efektu, tempo wzrostu będzie zmniejszać się proporcjonalnie do wartości uwzględnionej teoretycznym maksimum populacji pomniejszonej o jej aktualny stan.

2 Cel pracy

Głównym celem tej pracy jest przedstawienie pojęcia odwzorowania logicznego oraz zbadanie zależności od parametru k za pomocą algorytmów zaimplementowanych w języku Python.

3 Odwzorowania logistyczne - zastosowanie

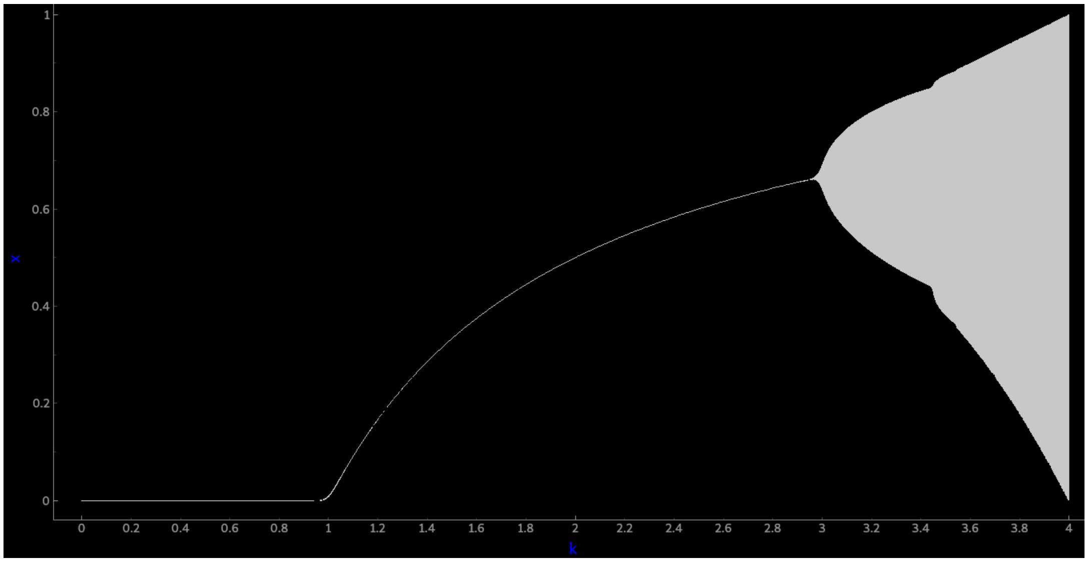
Względna prostota odwzorowania logistycznego sprawia, że jest powszechnie używana jako punkt wyjścia do rozważań nad koncepcją chaosu. Z opisu dowiadujemy się, że systemy chaotyczne wykazują dużą wrażliwość na warunki początkowe (typową właściwość odwzorowwania logistycznego dla większości **k** których wartości wahają się pomiędzy **3,57 a 4).** Częstym źródłem owych warunków jest to, że odwzorowanie przedstawia powtarzające się **zaginanie i rozciąganie przestrzeni**, na której jest zdefiniowana. To rozciąganie i zwijanie prowadzi nie tylko do stopniowej dywergencji sekwencji iteracji, ale także do wykładniczej dywergencji, której dowodem jest również złożoność i nieprzewidywalność chaotycznego odwzorowania logistycznego.

W rzeczywistości wykładnicza rozbieżność sekwencji iteracji wyjaśnia związek między chaosem a nieprzewidywalnością, czyli mały błąd stanu początkowego systemu, który będzie odpowiadał dużemu błędowi stanu późniejszego. **W związku z tym, prognozy dotyczące przyszłych stanów stają się stopniowo gorsze gdy istnieją nawet bardzo małe błędy w naszej wiedzy o stanie początkowym.** Ta cecha nieprzewidywalności i pozornej losowości sprawiła, że równanie odwzorowania logistycznego zostało **użyte jako generator liczb pseudolosowych we wczesnych komputerach oraz szyfrowaniu danych.**

Pomimo chaotycznej natury, możliwe jest sformułowanie precyzyjnych stwierdzeń dotyczących prawdopodobieństwa przyszłego stanu w tym systemie. Jeśli (prawdopodobnie chaotyczny) układ dynamiczny ma **atraktor** (może nim być punkt, zamknięta krzywa lub fraktal),istnieje wówczas miara prawdopodobieństwa, która określa długoterminową proporcję czasu spędzanego przez system w różnych obszarach atraktora. W przypadku odwzorowania logistycznego z parametrem $\mathbf{k} = \mathbf{4}$ i stanem początkowym w **(0,1)** atraktorem jest również przedział (0,1), a miara prawdopodobieństwa odpowiada rozkładowi beta o parametrach $\mathbf{a} = \mathbf{0{,}5}$ **i** $\mathbf{b} = \mathbf{0{,}5}$**.**

4 Parę słów o programie

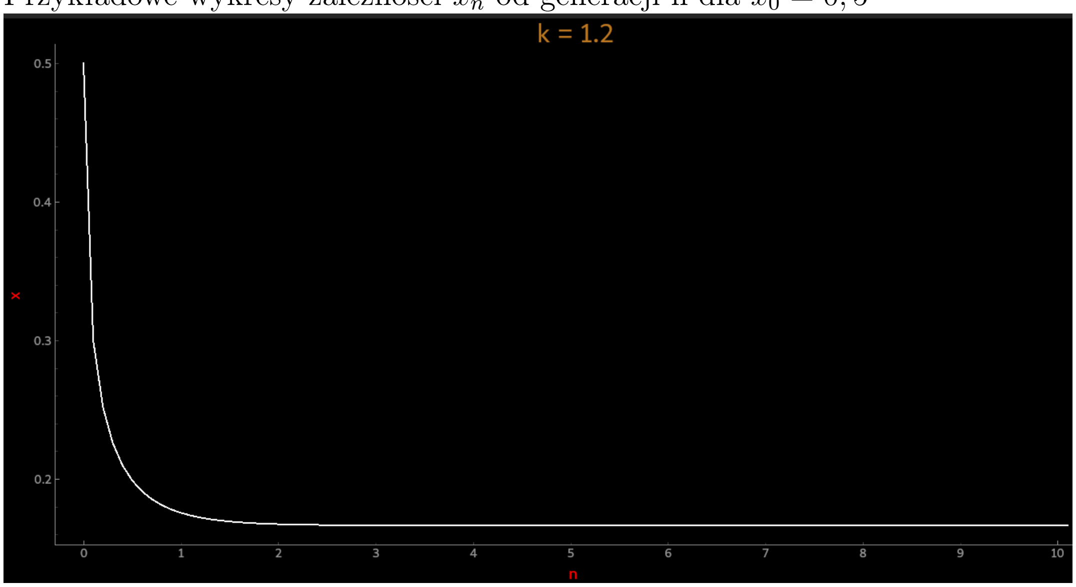
Do rozwiązania problemu stworzyliśmy dwa trywialne skrypty w języku Python, z użyciem bibliotek PyQt5 oraz PyQtGraph.

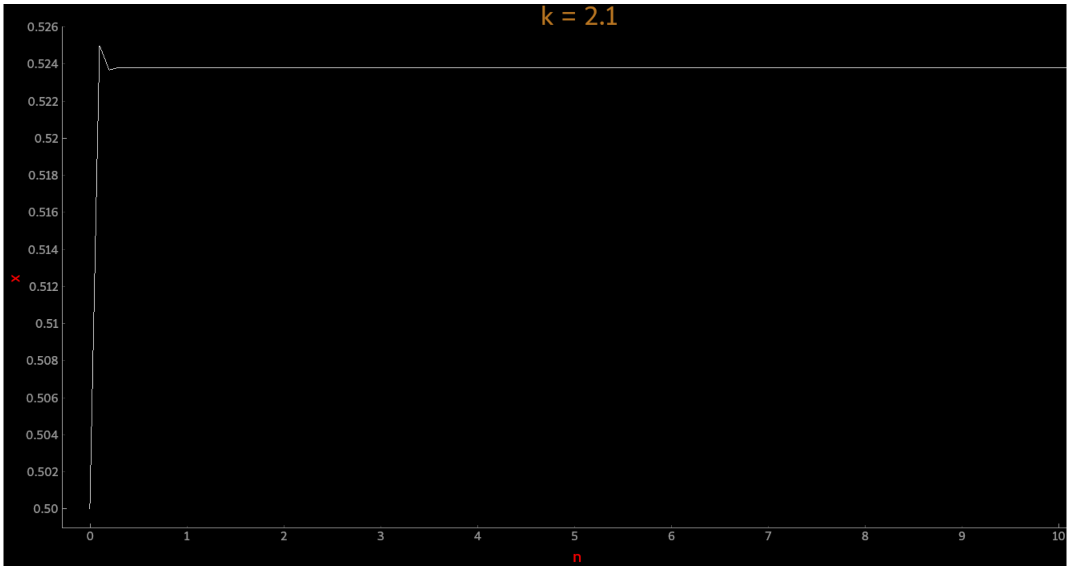


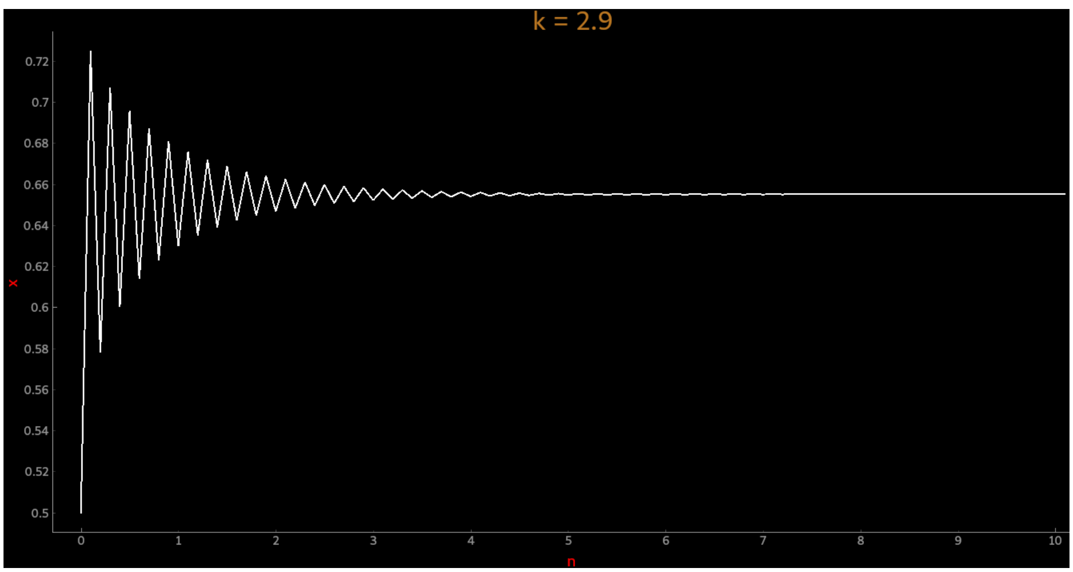
Rysunek 1: Diagram bifurkacyjny wygenerowany przez skrypt

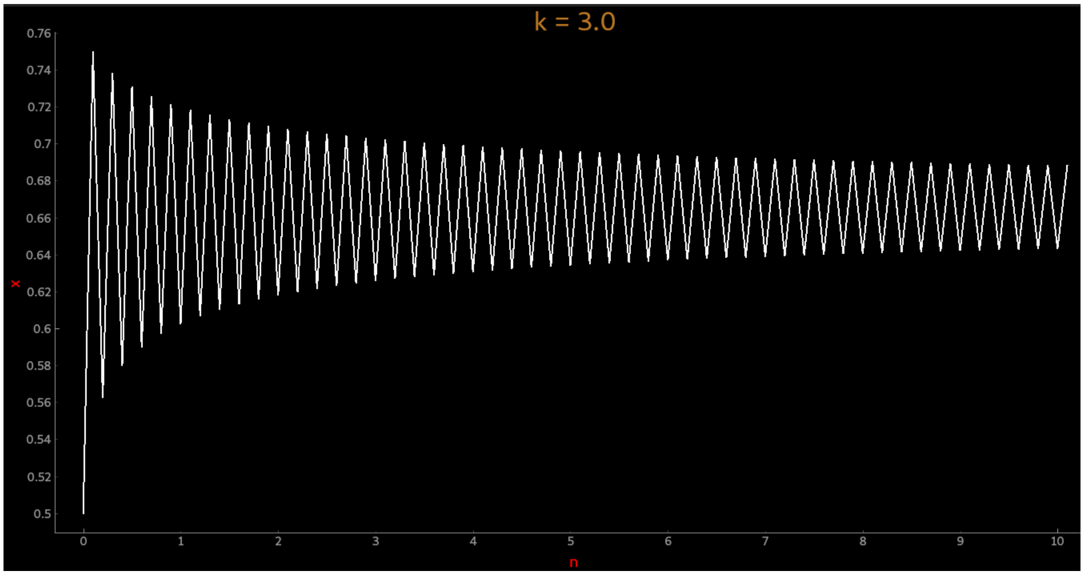
5 Podgląd wyników

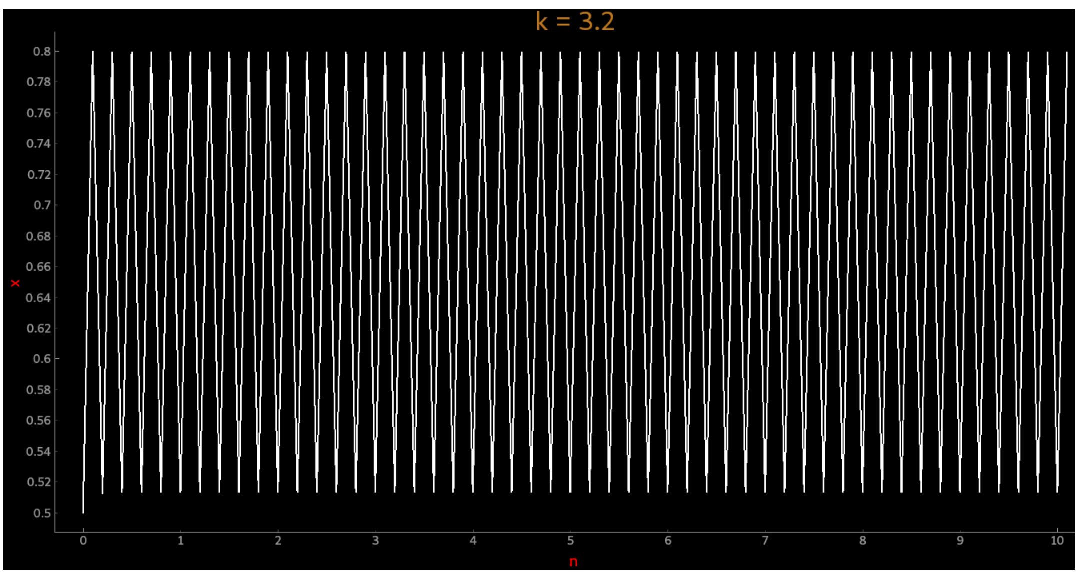
Przykładowe wykresy zależności x_n od generacji n dla $x_0 = 0,5$

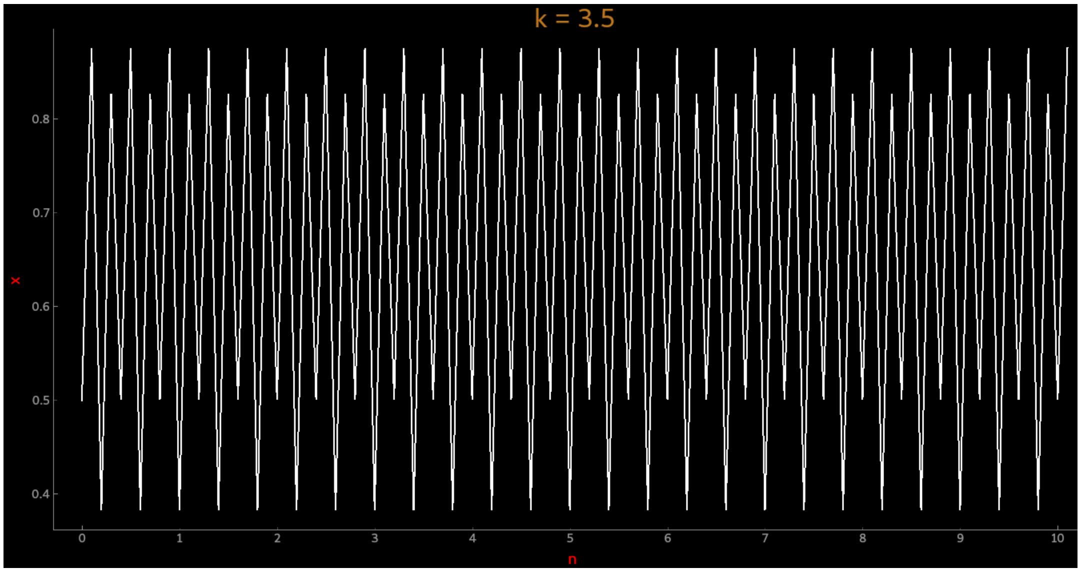


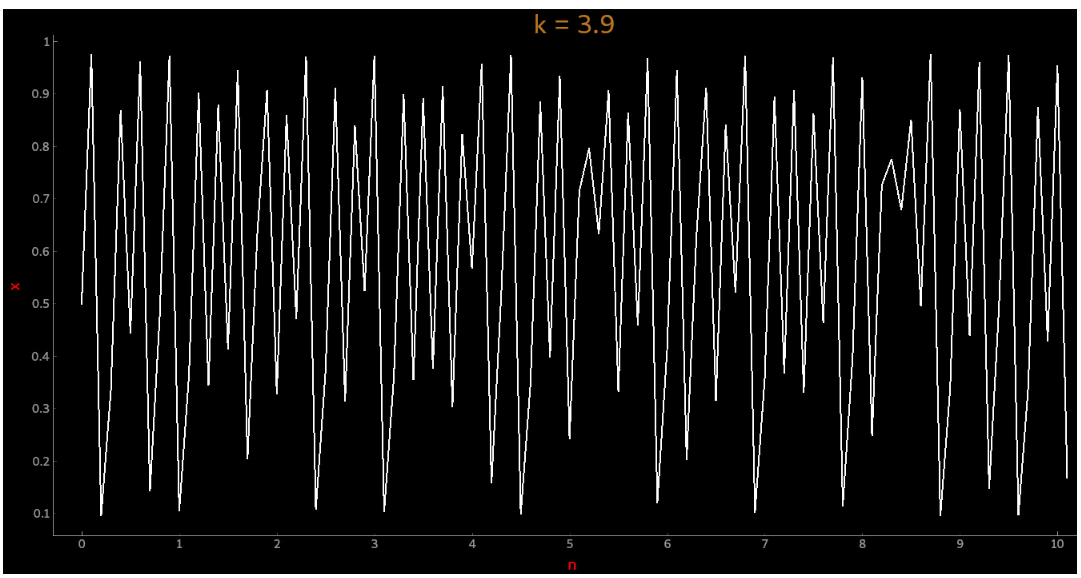

k = 2.1
0.526
0.524
0.522
0.52
0.518
0.516
0.514
0.512
0.51
0.508
0.506
0.504
0.502
0.50
x
0
1
2
3
4
5
6
7
8
9
10
n

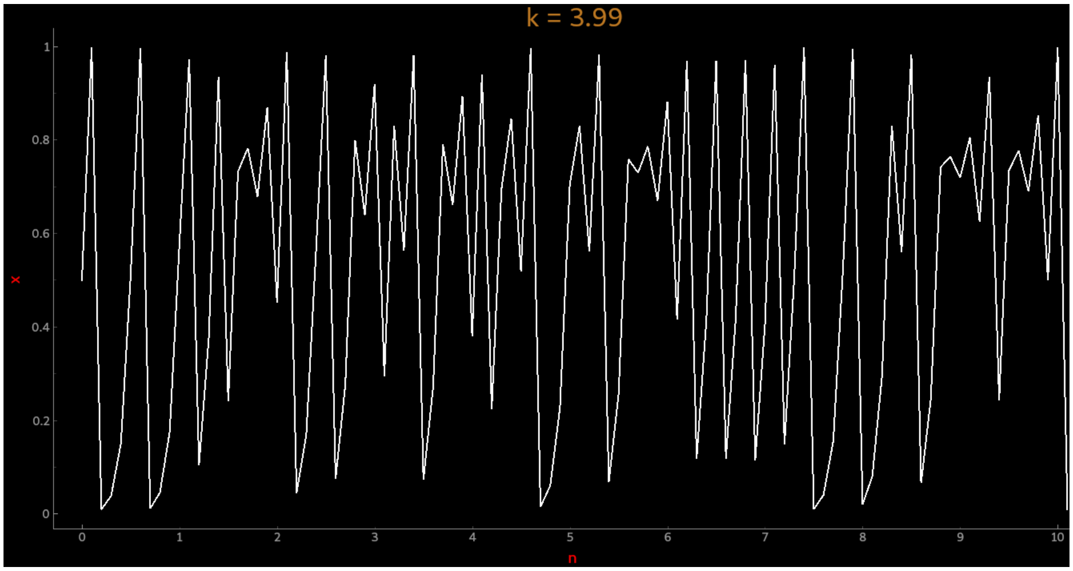

k = 2.9
0.72
0.7
0.68
0.66
0.64
0.62
0.6
0.58
0.56
0.54
0.52
0.5
x
0
1
2
3
4
5
6
7
8
9
10
n


k = 3.0
x
n


k = 3.2
x
n


k = 3.5
x
n
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
5
6
7
8
9
10


k = 3.9
x
n
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6
7
8
9
10



6 Podsumowanie

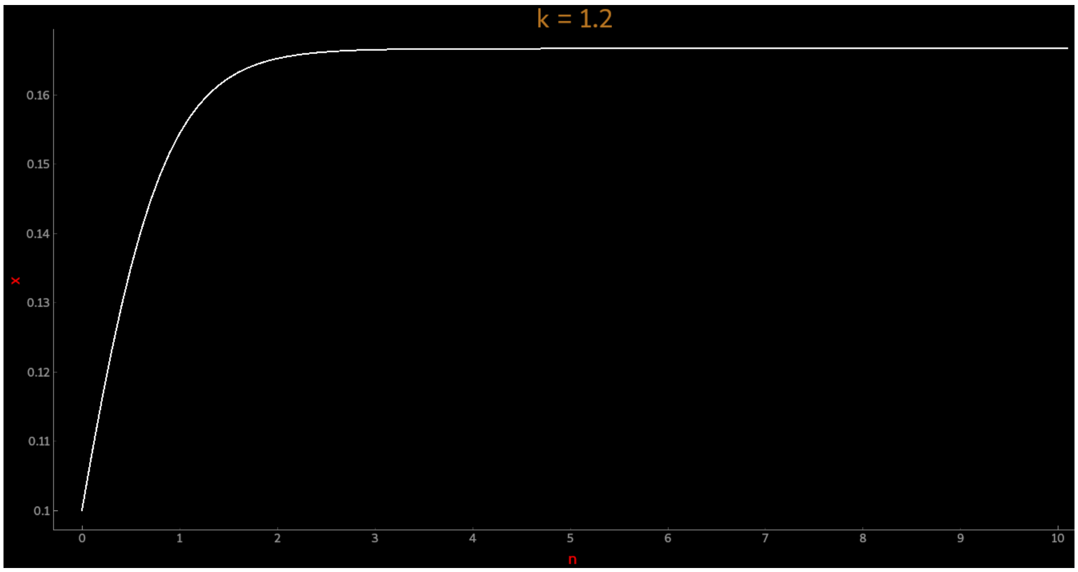
Programy zachowują się zgodnie z przewidywaniami, widać dużą wrażliwość na warunki początkowe. Widzimy, że współczynnik k odgrywa dużo większą rolę niż x_0.

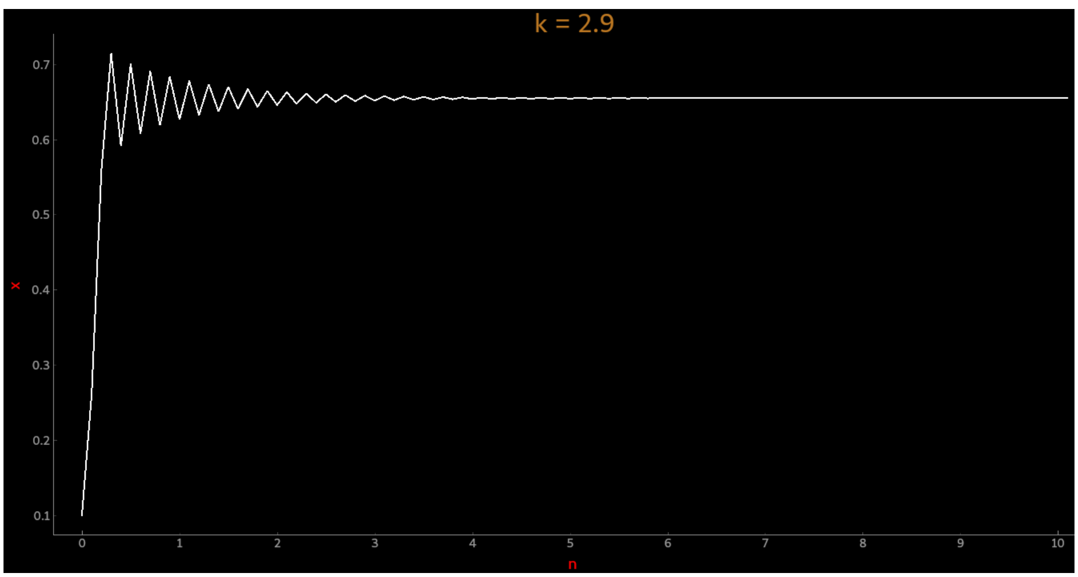
7 Bibliografia

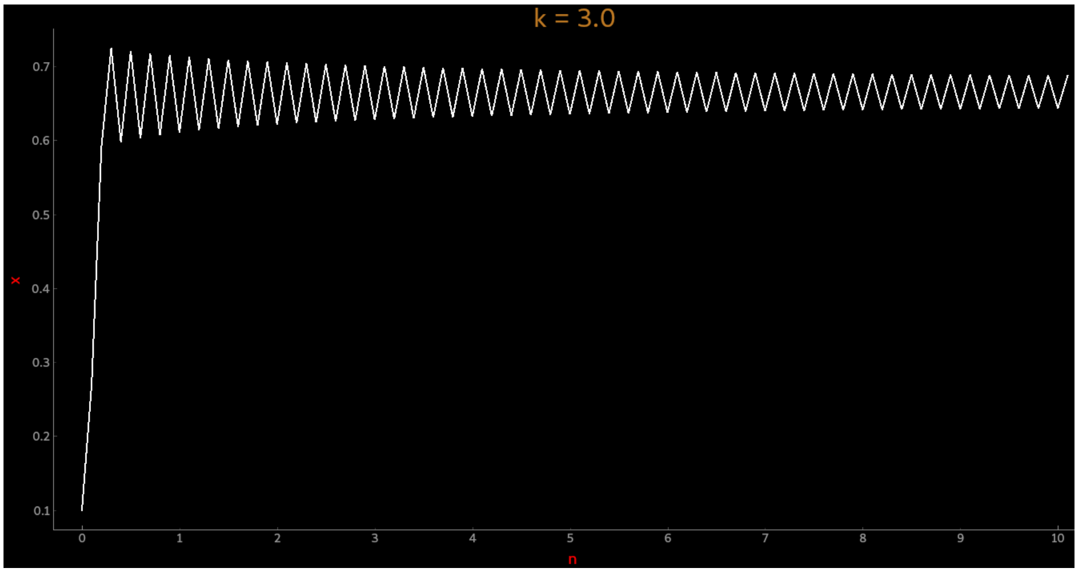
- `https://en.wikipedia.org/wiki/Logistic_map`
- `https://pl.wikipedia.org/wiki/Atraktor`
- `https://www.youtube.com/watch?v=ovJcsL7vyrk`
- `https://geoffboeing.com/2015/03/chaos-theory-logistic-map`

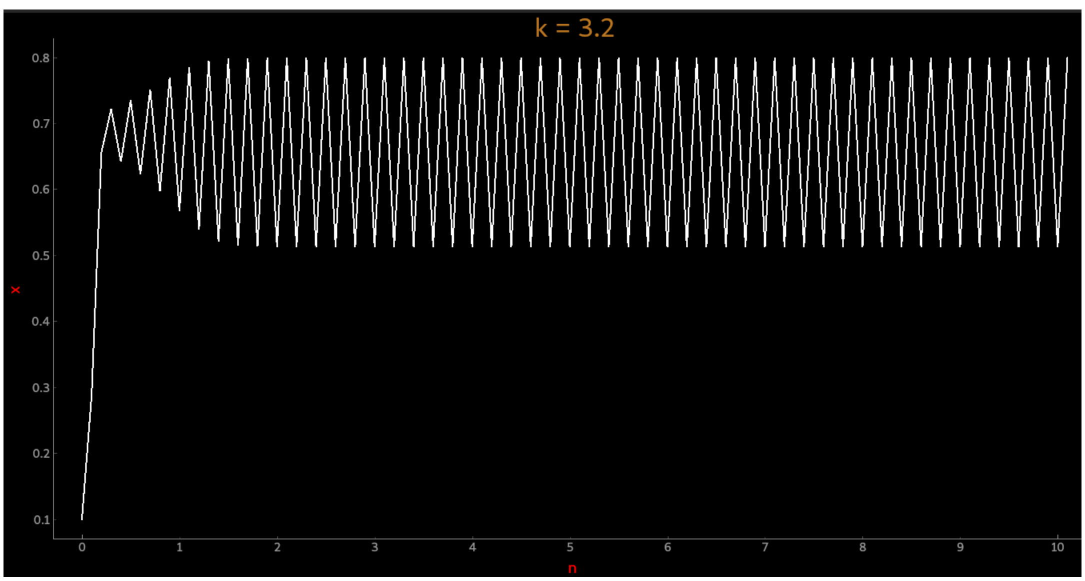
8 Appendix

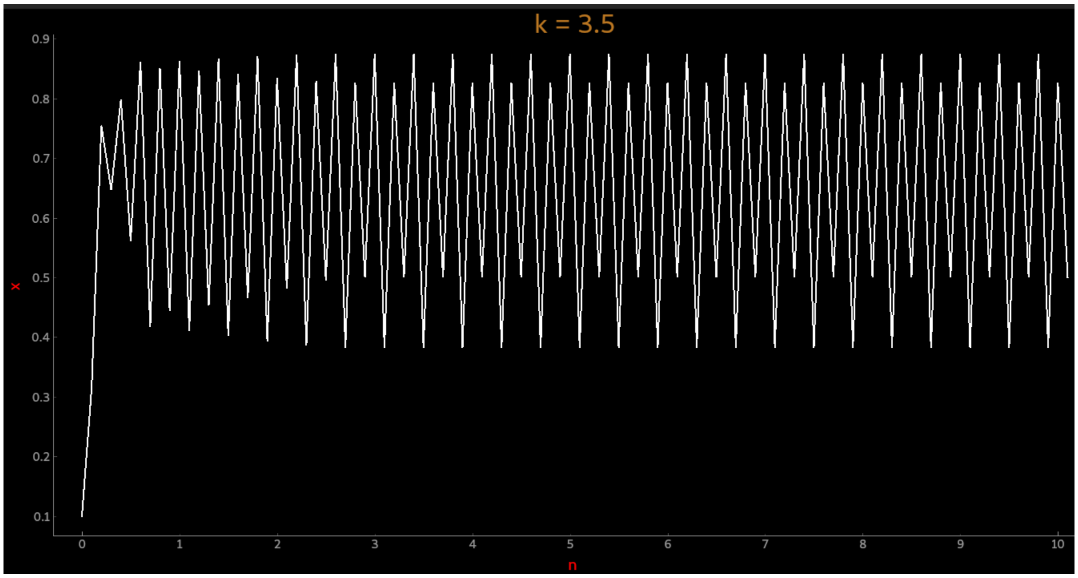
Wykresy dla $x_0 = 0.1$.

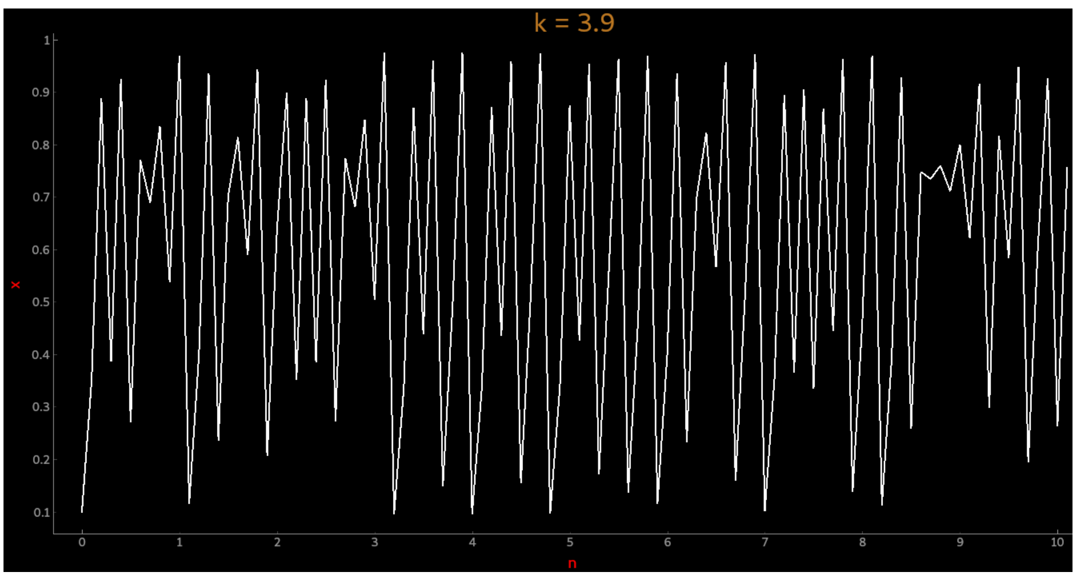


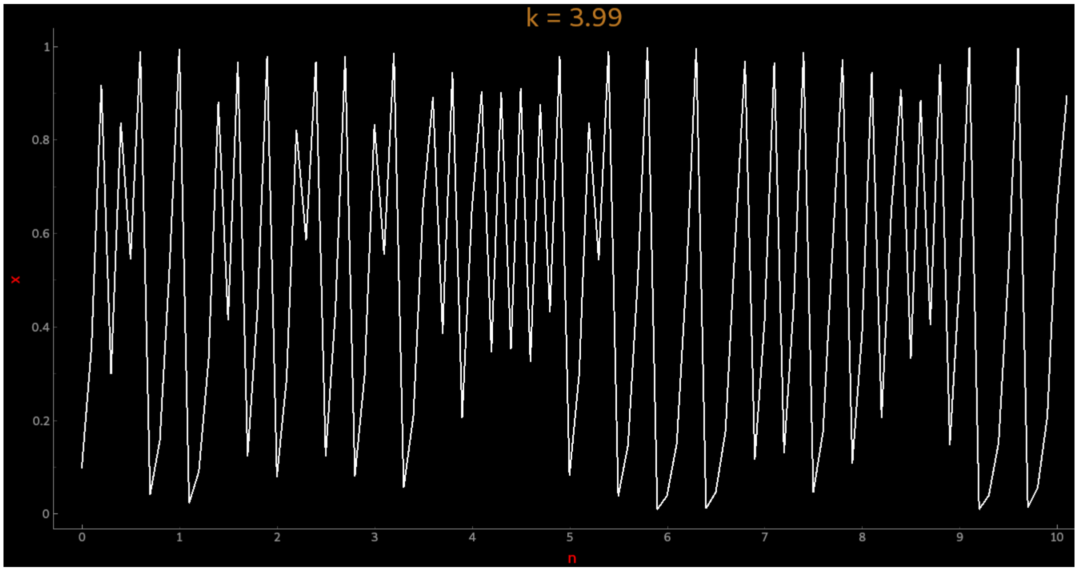



k = 3.0
x
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
1
2
3
4
5
6
7
8
9
10
n


k = 3.2
x
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
5
6
7
8
9
10
n


k = 3.5
x
n
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10


k = 3.9
x
n
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10


k = 3.99
x
n
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4
5
6
7
8
9
10