

Chaos od zera

Jak matematyka opisuje świat
i gdzie przestaje go przewidywać

Konrad Cinkusz

Każda liczba sprawdzona kodem uruchomionym na Pythonie 3.14.7, [numpy](#) 2.5.3 i [matplotlib](#) 3.11.2, wrzesień 2026.

Każdy listing to plik w katalogu code/ repozytorium [konradcinkusz/chaos-theory-for-dummies](#).

Tekst: CC BY-NC-SA 4.0 · Kod: MIT

Wydanie 0.1

Chaos od zera. Jak matematyka opisuje świat i gdzie przestaje go przewidywać. Wydanie 0.1.

Copyright © 2026 Konrad Cinkusz. Wszelkie prawa zastrzeżone, z wyjątkiem udzielonych w poniższych licencjach.

Tekst jest udostępniony na licencji CC BY-NC-SA 4.0. Kod – każdy plik w katalogu code/ i każdy listing, który książka z niego drukuje – na licencji MIT.

Wydanie własne autora. Książka nie ma wydawcy, dystrybutora ani numeru ISBN.

Wydana w dwóch językach, polskim i angielskim, z jednego repozytorium. Dostępna także po angielsku jako *Chaos from Zero*.

Wersje ustalone: wrzesień 2026. Każdą liczbę w tej książce, której nie da się policzyć w pamięci, wyprodukował program z repozytorium, a proces budowania uruchamia wszystkie te programy ponownie przy każdej zmianie i odmawia zbudowania książki, jeśli choć jedna wydrukowana cyfra przestała się zgadzać.

Ta książka nie należy do żadnej serii poradników dla początkujących, niezależnie od tego, jak nazywa się repozytorium, w którym powstała.

Spis treści

Jak korzystać z tej książki	viii
Wprowadzenie	xi
I Mechanizm zegara: jak matematyka opisuje świat	1
1 Model to maszyna do przewidywania	2
1.1 Trzy części każdego modelu	2
1.2 Twoje laboratorium	3
1.3 Model, który robi kroki	5
1.4 Błędny i użyteczny	7
1.5 Co dzieje się z małym błędem?	8
1.6 Mechanizm zegara	9
Czy potrafisz?	11
Punkt kontrolny	11
2 Reguła następnego kroku	12
2.1 Dziś, jutro, pojutrze	12
2.2 Jedna prosta, trzy losy	14
2.3 Powolne eksplozje i liczenie kroków	16
2.4 Gdzie reguła stoi w miejscu	18
2.5 Zostać na miejscu albo uciec	20
2.6 Nachylenie: ile razy mnoży się pchnięcie	21
2.7 Zobaczyć to: diagram pajęczynowy	22
Czy potrafisz?	24
Punkt kontrolny	25
3 Sprężenie zwrotne: dlaczego nic nie rośnie wiecznie	26
3.1 Podwajanie i dlaczego nie może trwać	26
3.2 Sprężenie zwrotne	27
3.3 Wzrost zwalnia, gdy brakuje miejsca	28
3.4 Dwa pchnięcia, które się nie sumują	31
3.5 Gdzie populacja się ustala, i nachylenie	31
3.6 Przestrzelenie	34
3.7 Kiedy populacja przestaje się ustalać	37
Czy potrafisz?	39
Punkt kontrolny	39

4	Ruch: reguły zmiany i mapa wszystkich stanów	40
4.1	Reguła, która mówi, jak szybko	40
4.2	Małe kroki	41
4.3	Spójrz naprzód i uśrednij	43
4.4	Mapa wszystkich stanów	45
4.5	Zatrzymać się, krążyć albo uciec	47
4.6	Dwa ciała: mechanizm zegara w najlepszym wydaniu	50
	Czy potrafisz?	52
	Punkt kontrolny	52
II	Gdzie kończy się przewidywanie	53
5	Jedna linijka algebry, każde możliwe zachowanie	54
5.1	Reguła i jej jedyne pokrętko	54
5.2	Kręcenie pokrętkiem	55
5.3	Stan ustalony i chwila, w której puszcza	57
5.4	Pajęczyna na krzywej	59
5.5	Szukanie cyklu, także daremne	60
5.6	Wygląda na losowe, a nie jest	63
	Czy potrafisz?	65
	Punkt kontrolny	66
6	Efekt motyla, zmierzony	67
6.1	Dwa punkty startowe, jedna dziesięciomiliardowa	67
6.2	Różnica się podwaja	69
6.3	Skąd bierze się podwajanie	71
6.4	Ile jest warta cyfra	73
6.5	Wydruk Lorenza	74
6.6	Deterministyczny, a mimo to nieprzewidywalny	76
	Czy potrafisz?	78
	Punkt kontrolny	78
7	Liczba na chaos: wykładnik Lapunowa	80
7.1	Krok po kroku	80
7.2	Logarytm zamienia mnożenie w dodawanie	82
7.3	Wykładnik Lapunowa	83
7.4	Jak daleko naprzód: horyzont przewidywania	86
7.5	Chaos przychodzi i odchodzi	89
	Czy potrafisz?	91
	Punkt kontrolny	92
8	Droga do chaosu i liczba, której nikt się nie spodziewał	93
8.1	Jeden obraz dla każdego ustawienia pokrętła	93
8.2	Rozwidlenia, które się zagęszczają	96
8.3	Szczyt garbu	97
8.4	Liczba, która należy do drogi	100
8.5	Porządek wewnątrz chaosu	103

Czy potrafisz?	105
Punkt kontrolny	105

III Kształty chaosu 107

9 Trzy równania na pogodę: atraktor Lorenza 108

9.1 Trzy liczby na garnek wody	108
9.2 Za przepływem	110
9.3 Dwa starty o włos od siebie	113
9.4 Nieprzewidywalny, ale nie gdziekolwiek	115
9.5 Co wciąż da się przewidzieć	117
Czy potrafisz?	119
Punkt kontrolny	120

10 Rozciągnij i złóż: dziwne atraktory 121

10.1 Rozciągnij, potem złóż	121
10.2 Dwuwierszowe odwzorowanie Hénona	123
10.3 Jeden kształt dla każdego startu	125
10.4 Ściśnięty, nie zatrzymany	127
10.5 Warstwy w warstwach	130
10.6 Przepływ pocięty na odwzorowanie	131
Czy potrafisz?	132
Punkt kontrolny	133

11 Fraktale: mierzenie czegoś, co nie ma długości 134

11.1 Wybrzeże i linijka	134
11.2 Krzywa, która się nie ustala	136
11.3 Liczenie kopii	138
11.4 Liczenie pudełek	141
11.5 Kształt, który zostawia chaos	142
11.6 Fraktale w świecie	144
Czy potrafisz?	146
Punkt kontrolny	146

12 Iteracja na płaszczyźnie: Julia i Mandelbrot 147

12.1 Liczby, które są punktami	147
12.2 Kwadrat, dodawanie i pytanie o ucieczkę	149
12.3 Zbiór Mandelbrota	152
12.4 Zbiory Julii: ustal c , zmieniaj start	155
12.5 Odwzorowanie logistyczne w przebraniu	156
Czy potrafisz?	160
Punkt kontrolny	160

IV	Chaos w świecie	161
13	Wahadła i planety	162
13.1	Wahadło na wahadle	162
13.2	Liczba, która nie może drgnąć	164
13.3	Milionowa część radiana	165
13.4	Łagodne kołysania i dzikie	168
13.5	Dwa ciała, trzy ciała	170
13.6	Czy Układ Słoneczny jest zegarem?	171
	Czy potrafisz?	172
	Punkt kontrolny	173
14	Pogoda, klimat i prognoza zespołowa	174
14.1	Zabawkowa atmosfera	174
14.2	Dwadzieścia prognoz zamiast jednej	175
14.3	Szansa na deszcz	179
14.4	Jak daleko naprzód?	181
14.5	Pogoda i klimat	183
	Czy potrafisz?	186
	Punkt kontrolny	187
15	Życie, rynki i szum: jak odróżnić chaos od przypadku	188
15.1	Dwa szeregi, jeden zestaw liczb	188
15.2	Każda wartość względem następnej	190
15.3	Prognoza z przeszłości	193
15.4	Prawdziwe dane to mieszanina	195
15.5	Populacje, epidemie, serca	196
15.6	Rynki i czego wymaga twierdzenie o chaosie	198
	Czy potrafisz?	200
	Punkt kontrolny	200
16	Chaos wewnątrz komputera	202
16.1	Co zapamiętuje komputer	202
16.2	Odwzorowanie, którego komputer nie umie policzyć	204
16.3	Jedna reguła, dwa programy	206
16.4	Dlaczego więcej cyfr cię nie uratuje	208
16.5	Zła droga, dobry obraz	210
16.6	Maszyny, które muszą się powtarzać	212
	Czy potrafisz?	214
	Punkt kontrolny	214
V	Życie z chaosem	215
17	Okiełznać chaos: sterowanie i synchronizacja	216
17.1	Punkt, którego orbita nie utrzyma	216
17.2	Pchnięcie we właściwej chwili	217
17.3	Drobne, a nie siła	219

17.4	Dwa układy chaotyczne, które się zgadzają	222
17.5	Co już z tym zrobiono, a czego nie	225
	Czy potrafisz?	226
	Punkt kontrolny	227
18	Po co ci chaos: przewodnik terenowy po przewidywaniu	228
18.1	Pytanie pierwsze: czy mała zmiana rośnie?	228
18.2	Pytanie drugie: jak daleko jest horyzont?	229
18.3	Pytanie trzecie: co przetrwa horyzont?	232
18.4	Pytanie czwarte: chaos, szum czy mała przyczyna?	234
18.5	Pytanie piąte: czy da się tym sterować?	235
18.6	Przewodnik terenowy na jednej stronie	236
18.7	Dlaczego każdy tego potrzebuje	237
18.8	Demon Laplace’a po raz ostatni	238
	Czy potrafisz?	239
	Punkt kontrolny	239
VI	Dodatki	241
A	Odpowiedzi	242
B	Błędne odczytania chaosu	249
C	Matematyka, której potrzebujesz, na czterech stronach	255
C.1	Bardzo duże i bardzo małe liczby	255
C.2	Reguły, kroki i orbity	255
C.3	Potęgi, wzrost i połowienie	255
C.4	Logarytmy	255
C.5	Nachylenie krzywej	256
C.6	Tempo zmian	256
C.7	Sumy i średnie	256
C.8	Reszty	256
C.9	Liczby zespolone	257
C.10	Litery greckie	257
D	Dalsza lektura	258
D.1	Gdzie idee pojawiły się po raz pierwszy	258
D.2	Książki na później	259
E	Manifest	260
	Manifest laboratoriów	260
	Manifest wykresów	261
	Manifest diagramów	262

Jak korzystać z tej książki

Ta książka zakłada, że umiesz dodawać, mnożyć, policzyć procent i odczytać wykres. Nie zakłada, że pamiętasz jakąkolwiek algebrę, i nigdy nie używa symbolu, którego wcześniej nie wprowadziła. Każdy wzór jest też powiedziany słowami, a tam, gdzie wzór tylko by przeszkadzał, wzoru nie ma. Jeśli skończyłeś z matematyką w szkole i zawsze chciałeś wiedzieć, co ludzie mają na myśli, kiedy mówią „chaos”, to jesteś czytelnikiem, dla którego ją napisano.

Prosi cię za to o jedno: o udział. Książka jest zbudowana z małych kroków, a na końcu wielu z nich pada pytanie.

Pętla

Każdy rozdział czyta się tak samo:

1. **Przeczytaj krótki fragment.** Każdy podrozdział to kilka akapitów.
2. **Zatrzymaj się i pomyśl.** Fioletowa ramka zadaje ci pytanie. Odpowiedz na nie – na kartce, na głos albo na zasadzie zgadywania – zanim pójdziesz dalej. Odpowiedź jest tuż pod spodem, więc zasłoń ją dłonią.
3. **Uruchom kod**, jeśli masz obok komputer. Każdy listing to mały plik, a uruchomienie go to jedno polecenie.
4. **Zrób laboratorium**, jeśli chcesz, żeby pomysł stał się twój: laboratorium to napisany do połowy program i test, który mówi ci, kiedy działa.
5. **Rozwiąż sprawdzian** na końcu rozdziału i porównaj odpowiedzi z dodatkiem [A](#).

Dlaczego książka wciąż zadaje ci pytania

Pytanie zadane, zanim poznasz odpowiedź, jest niewygodne. Tak ma być. Odpowiedź, po którą sięgnąłeś sam, nawet błędna, zostaje w głowie dużo dłużej niż taka, którą tylko przeczytałeś, a błędny strzał, który potem zostaje poprawiony, to najszybszy sposób, żeby odkryć, które twoje intuicje cię zawiodą. W tym temacie jest ich pełno. Pierwszy strzał większości ludzi co do tego, jak szybko rośnie mały błąd, albo co potrafi prosta reguła, jest błędny, a książka jest ułożona tak, żebyś oddał ten strzał, zanim zobaczysz dlaczego.

Więc kiedy ramka prosi, żebyś odpowiedział, zanim pójdziesz dalej, zrób to, proszę. Nikt tego nie ocenia.

Laboratorium

Wszystko w tej książce, co może sprawdzić komputer, zostało przez komputer sprawdzone, a ty możesz to sprawdzić ponownie. Programy są w repozytorium książki, w katalogu

code/. Są napisane w Pythonie i potrzebują jednego narzędzia, `uv`, które samo instaluje właściwą wersję Pythona i wszystkie biblioteki. Kiedy już je masz, z katalogu `code/` repozytorium:

```
uv sync
uv run python ch01/falling_stone.py
uv run pytest -k l01_01
```

Pierwsza linia przygotowuje laboratorium, raz. Druga uruchamia listing z rozdziału 1. Trzecia uruchamia test pierwszego laboratorium z tego rozdziału. Nie musisz znać Pythona, żeby zacząć: pierwsze rozdziały tłumaczą każdą linię, którą drukują, a programy przez całą książkę pozostają krótkie.

Ramki

Poza pytaniami tekst przerywają ramki, każda z jednym zadaniem:

Pułapka. Błąd, który popełnia prawie każdy, pokazany po tym, jak miałeś okazję popełnić go sam.

Gdzie to spotkasz. Prawdziwe miejsce, w którym pomysł się pojawia: dziedzina, urządzenie, praktyka.

Czego tu nie dowodzimy. Co książka stwierdza bez dowodu i gdzie ten dowód można znaleźć.

Uwaga, Ostrożnie. Dygresja i przestroga.

chaoslab. Kawałek małej biblioteki chaosu tej książki, z której korzystają listingi.

Laboratorium. Coś do zrobienia przy klawiaturze, z testem, który mówi ci, kiedy to zrobiłeś.

Liczby, którym można ufać

Każdą liczbę w tej książce, której nie da się policzyć w pamięci, wyprodukował program, a książka drukuje to, co wypisał program, a nie liczbę, którą ktoś przepisał. Proces budowania sprawdza każdą z nich ponownie przy każdej zmianie. Tutaj ma to większe znaczenie niż zwykle, z powodu, któremu książka poświęca cały rozdział: w układzie chaotycznym obliczenie może pomylić się na ostatniej cyfrze i kilkadziesiąt kroków później być zupełnie błędne. Rozdział 16 wyjaśnia, jak książka się przed tym chroni i dlaczego niektóre jej liczby są celowo zaokrąglone.

Czytanie bez komputera

Da się. Wynik każdego listingu jest wydrukowany obok niego, każdy wykres jest na stronie, a pytania działają na papierze. Ominą cię tylko laboratoria, a one są dla czytelników, którzy chcą pójść dalej, a nie po to, żeby zamykać drogę do wyводу.

Dwa języki, jedna książka

Książka ukazuje się po angielsku i po polsku z jednego źródła. Oba wydania zawierają te same rozdziały, podrozdziały, wykresy, liczby i pytania w tej samej kolejności, a proces budowania porównuje je, żeby tego dopilnować.

Wprowadzenie

Całkowite zaćmienie Słońca, które przeszło nad Hiszpanią 12 sierpnia 2026 roku, było w tablicach astronomów na długo przed urodzeniem kogokolwiek, kto je oglądał. Jego pas, jego datę i minutę, w której się zaczęło, wyliczono z ruchów Ziemi i Księżyca, a tablice zaćmień, liczone w ten sam sposób, sięgają tysięcy lat naprzód.

Nikt nie potrafiłby powiedzieć ludziom czekającym w tym pasie, na miesiąc przed zaćmieniem, czy niebo będzie na tyle czyste, żeby je zobaczyć.

Obie prognozy powstają z tego samego rodzaju matematyki. W obu przypadkach komputer bierze obecny stan kawałka świata, stosuje reguły opisujące, jak się on zmienia, i krok po kroku idzie naprzód w czasie. Reguły dla powietrza nie są tajemnicą: to te same prawa ruchu i ciepła, które opisują Księżyc. A jednak jedna prognoza jest dobra na tysiąclecie, a druga na mniej więcej tydzień. Różnica nie polega na tym, że meteorolodzy są mniej staranni od astronomów, ani na tym, że powietrze jest zbyt skomplikowane, by je zapisać. Różnica ma nazwę, pomiar i dokładny opis matematyczny. Nazywa się chaos i o nim jest ta książka.

Czym jest chaos

W mowie potocznej chaos oznacza bałagan. W matematyce oznacza coś dużo ciekawszego i niemal przeciwnego: układ, który podlega dokładnej regule, bez żadnego przypadku, a mimo to zachowuje się w sposób, którego nie da się przewidzieć na długo naprzód. Muszą być spełnione trzy warunki naraz:

- **reguła jest dokładna** – ten sam stan początkowy zawsze prowadzi do tej samej przyszłości, bez żadnej domieszki losowości;
- **małe różnice rosną** – dwa stany początkowe różniące się o mniej, niż cokolwiek da się zmierzyć, po pewnym czasie prowadzą do zupełnie różnych skutków;
- **ruch pozostaje w granicach** – nie ucieka do nieskończoności, więc rosnące różnice są wciąż zawracane do tego samego obszaru i nigdy się nie uspokajają.

Każdy z tych warunków jest prosty. Naukowców, którzy znaleźli je razem, zaskoczyło to, jak często ta kombinacja występuje. Nie potrzebuje skomplikowanej reguły: pierwszy układ chaotyczny w tej książce to jedna linijka szkolnej algebry, która mieści się na kalkulatorze. Nie potrzebuje losowości, tłumy ani tajemnicy. Pojawia się w kapiących kranach, wahadłach, populacjach zwierząt, rytmie serca, orbitach planet i, najsłynniej, w pogodzie.

Dlaczego warto poświęcić na to czas

Możesz rozsądnie zapytać, po co ktoś, kto nie jest naukowcem, miałby się tego uczyć. Oto powody, które przekonały mnie.

Żyjesz prognoząmi. Aplikacje pogodowe, projekcje emerytalne, krzywe epidemii, prognozy gospodarcze, szacunki korków i pewne siebie odpowiedzi programów komputerowych to przewidywania tworzone przez modele. Teoria chaosu mówi, którym z nich można w zasadzie ufać daleko naprzód, którym tylko przez krótki czas i jak w każdym przypadku wygląda uczciwa postać prognozy.

Mierzy granicę, a ta granica jest dziwna. W układzie chaotycznym błąd stanu początkowego rośnie w każdym kroku mniej więcej o ten sam czynnik. To znaczy, że tysiąckrotne ulepszenie pomiarów nie daje tysiąc razy dłuższej prognozy; daje stały dodatkowy odcinek czasu, po którym mgła zapada tak samo jak wcześniej. Kiedy raz zobaczysz to przeliczone na liczbach – a zobaczysz, w części II – już nigdy nie usłyszysz zdania „potrzebujemy tylko lepszych danych” w ten sam sposób.

Wyjaśnia, dlaczego dobre prognozy podają szansę. Jeśli dalekiej przyszłości nie da się policzyć jako jednej odpowiedzi, to da się policzyć rozrzut odpowiedzi i to, jak prawdopodobna jest każda z nich. „70 procent szans na deszcz” to nie asekuracja prognosty. To poprawna odpowiedź na pytanie, na które chaos nie pozwala odpowiedzieć „tak” albo „nie”.

Rozdziela rzeczy, które zwykle wrzuca się do jednego worka. „Nie da się tego przewidzieć” może znaczyć, że nikt nie zna reguł, że reguły zawierają przypadek, albo że reguły są znane dokładnie, a przyszłość i tak jest poza zasięgiem. To różne sytuacje, wymagające różnych reakcji, a chaos to ta trzecia.

Jest piękny i uniwersalny. Kształty, które rysuje chaos, to fraktale, struktury pełne szczegółów przy każdym powiększeniu, a niektóre jego liczby okazują się takie same w układach, które nie mają ze sobą nic wspólnego. Ciecz, obwód elektroniczny i linijka algebry mogą dzielić stałą, której nikt tam nie włożył.

I jest użyteczny. Ta sama wrażliwość, która czyni chaos nieprzewidywalnym, pozwala nim sterować bardzo małymi pchnięciami, a układy chaotyczne można zmusić, żeby szły ze sobą w parze. Część V pokazuje jedno i drugie.

Czym ta książka jest, a czym nie jest

Nie jest podręcznikiem i nie robi z ciebie matematyka. Nie jest też książką z obrazkami, z której usunięto matematykę. Każdy pomysł w niej to prawdziwy pomysł, rozwinięty od arytmetyki w górę, a każde twierdzenie, które może sprawdzić komputer, zostało przez komputer sprawdzone; programy są wydrukowane w książce i możesz je uruchomić sam. Tam, gdzie książka coś stwierdza bez dowodu, mówi to otwarcie i mówi, gdzie ten dowód jest.

Zaczyna od zera. Część I (rozdziały od 1 do 4) pokazuje, jak matematyka w ogóle opisuje świat: czym jest model, jak regułę stosuje się krok po kroku w czasie, dlaczego sprzężenie zwrotne nie pozwala niczemu rosnać w nieskończoność i jak opisuje się ruch. Część II (rozdziały od 5 do 8) znajduje chaos w jednej linijce algebry, mierzy, jak szybko rosną małe błędy, zamienia to w jedną liczbę i horyzont prognozy, i idzie drogą od porządku do chaosu. Część III (rozdziały od 9 do 12) przygląda się kształtom, jakie tworzy chaos: atraktorowi Lorenza, atraktorom dziwnym, fraktalom i zbiorowi Mandelbrota. Część IV (rozdziały od 13 do 16) wyprowadza go w świat: wahadła i planety, prognoza pogody, układy żywe i rynki, i sam komputer. Część V (rozdziały 17 i 18) pokazuje, jak chaos można kontrolować, i kończy się przewodnikiem terenowym do oceniania każdej

prognozy, jaką ktoś ci przedstawi.

Pytanie

W 1814 roku matematyk Pierre-Simon Laplace wyobraził sobie inteligencję, która zna każdą siłę w przyrodzie i dokładny stan wszystkiego w jednej chwili. Dla takiej inteligencji, pisał, nic nie byłoby niepewne: przyszłość byłaby równie jasna jak przeszłość. To najbardziej pewne siebie zdanie, jakie kiedykolwiek napisano o przewidywaniu, i przez sto lat wyglądało na naturalny wniosek z fizyki.

Ta książka pyta, czy miał rację, i odpowiada liczbami, które możesz policzyć sam. Rozdział 1 zaczyna się od kamienia i kubka herbaty.

Część I

Mechanizm zegara: jak matematyka opisuje świat

Model to maszyna do przewidywania

Korzystasz z modeli matematycznych codziennie, nawet jeśli tak ich nie nazywasz. Aplikacja, która mówi, że pociąg przyjedzie za siedem minut, liczy jakiś model. Tak samo minutnik w piekarniku, tablica pływów, dawkovanie na ulotce leku i cichy głos, który podpowiada, że trzeba wyjść dziesięć po ósmej, żeby zdążyć na dziewiątą. Każdy z nich bierze kilka liczb o tym, co jest teraz, stosuje regułę i podaje ci liczbę o tym, co będzie później.

Ta książka opowiada o tym, co dzieje się z tą obietnicą. Pierwsze cztery rozdziały pokazują, jak dobrze działa – jak linijka algebry potrafi powiedzieć, gdzie będzie kamień, jak gorąca będzie twoja herbata i kiedy wróci kometa. Reszta książki pokazuje, gdzie się łamie, dlaczego łamie się nawet wtedy, gdy reguła jest doskonała, i co wciąż da się przewidzieć, gdy to się stanie. To ostatnie jest powodem, by ją przeczytać: to pęknięcie nie jest porażką matematyki, a znajomość jego kształtu zmienia to, w co powinieneś wierzyć przy każdej prognozie, jaką ktoś ci poda.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- nazwać stan, regułę i przewidywanie każdego modelu, który ktoś ci pokaże
- uruchomić model napisany w Pythonie i przeczytać to, co wypisuje
- powiedzieć, co znaczy, że model jest błędny, i dlaczego błędny model wciąż może być użyteczny
- powiedzieć, co dzieje się z małym błędem stanu początkowego w dwóch modelach z tego rozdziału
- przytoczyć zegarmistrzowskie twierdzenie Laplace’a z 1814 roku i pytanie, które reszta książki mu zadaje

1.1 Trzy części każdego modelu

Upuść kamień z mostu. Ile przeleci w ciągu trzech sekund?

Fizyk odpowie wzorem, który możesz pamiętać ze szkoły: kamień spadający swobodnie przy powierzchni Ziemi pokonuje drogę

$$d = \frac{1}{2} g t^2$$

w ciągu t sekund, gdzie g to mniej więcej 9,81 metra na sekundę na sekundę. Ta jedna linijka to kompletny model i ma trzy części, które znajdziesz w każdym modelu w tej książce.

- **Stan:** liczby, które mówią, jak jest teraz. Tutaj to jedna liczba – czas t od chwili puszczenia kamienia.
- **Reguła:** jak te liczby zamieniają się w to, co chcesz wiedzieć. Tutaj to wzór.
- **Przewidywanie:** to, co mówi reguła. Tutaj – droga.

Zatrzymaj się i pomyśl 1.1

Podstaw do wzoru $t = 3$. Mniej więcej ile przeleciał kamień: około 10 metrów, około 30, czy około 45?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około 44,1 metra: połowa z 9,81, razy $3 \times 3 = 9$.

Jeśli odpowiedziałeś dziesięć albo trzydzieści, jesteś w dobrym towarzystwie: większość ludzi zaniża, bo kamień nie spada ze stałą prędkością. Ciągłe przyspiesza, więc droga rośnie z *kwadratem* czasu. Trzy razy dłuższy lot to dziewięć razy dłuższa droga.

To pierwsza rzecz, którą daje ci model: łapie intuicję, która miała cię zmylić. Nikt nie ma wyczucia dla t^2 . Wzór go nie potrzebuje.

Pułapka

„**Model to kopia rzeczywistości.**” Nie jest, i nic w powyższym wzorze nie przypomina kamienia. Model to maszyna do odpowiadania na pytania: wkładasz liczby, dostajesz liczby, a jedyny uczciwy test polega na tym, czy te liczby zgadzają się z tym, co potem zmierzysz. Mapa nie jest terenem, a model nie jest nawet mapą – to mapa, która dodatkowo mówi ci, o której dojedziesz.

1.2 Twoje laboratorium

Każdą liczbę w tej książce, której nie dało się policzyć w pamięci, policzył mały program, a każdy z tych programów leży w repozytorium książki w katalogu `code/`. Nie musisz być programistą, żeby je czytać. Są krótkie, napisane tak, żeby dało się je czytać, i każdy jest objaśniony przy pierwszym pojawieniu się. Oto kamień.

`code/ch01/falling_stone.py`

```
9 G = 9.81 # metres per second, per second, near the Earth's surface
10
11
12 def distance(t: float) -> float:
13     """How far a dropped stone has fallen after t seconds, air ignored."""
14     return 0.5 * G * t * t
```

Listing 1.1: Spadający kamień jako model: jedna stała, jedna reguła

Czytaj go tak, jak czytasz wzór. Wiersz zaczynający się od `G =` ustala stałą. Wiersze zaczynające się od `def distance` definiują regułę: podaj jej czas t , a zwróci $0.5 * G * t * t$

t, czyli wzór z rozpisanymi mnożeniami. Tekst w potrójnych cudzysłowach to notatka dla czytelnika, a wszystko po znaku # to komentarz. Potem program zadaje regule pytanie dla każdej pełnej sekundy od 0 do 5:

code/ch01/falling_stone.py

```
20 print(" t (s)   fallen (m)")
21 for t in range(6):
22     print(f"{t:5d}   {distance(t):10.2f}")
```

Listing 1.2: Sześć pytań zadanych modelowi

for t in range(6) znaczy „wykonaj następny wiersz z t równym 0, potem 1 i tak dalej aż do 5”. Uruchom go z katalogu code/ poleceniem `uv run python ch01/falling_stone.py`, a wypisze:

t (s)	fallen (m)
0	0.00
1	4.91
2	19.62
3	44.14
4	78.48
5	122.63

Spójrz na prawą kolumnę. W pierwszej sekundzie kamień spada około pięciu metrów, a w pięć sekund 122,6 metra. Jeśli kiedykolwiek upuściłeś coś z wysokości i liczyłeś, ta druga liczba wydaje się za duża, a wcale taka nie jest.

Uwaga

Materiał wstępny mówi, jak zainstalować laboratorium: jedno narzędzie, `uv`, które instaluje dokładnie tę wersję Pythona, na której sprawdzono książkę (3.14.7), i każdy pakiet w dokładnie tej wersji, którą podaje strona redakcyjna. Potem `uv run python` z nazwą pliku uruchamia dowolny listing z książki. Jeśli wolisz niczego nie uruchamiać, wynik każdego listingu jest wydrukowany obok; uruchamianie ich to sposób, żeby przekonać się, że książka nie prosi cię o wiarę na słowo.

Laboratorium 1.1 – Kamień w obie strony

Model odpowiada na pytania w więcej niż jednym kierunku. Podaj mu czas, a poda drogę; podaj drogę, a poda czas; podaj zmierzony spadek, a powie, ile musi wynosić grawitacja. Plik startowy zawiera trzy funkcje z opisami i bez treści. Uzupełnij je, aż test przejdzie. Druga potrzebuje pierwiastka kwadratowego, który w Pythonie nazywa się `math.sqrt`.

Plik startowy: code/labs/ch01/l01_01_stone.py

Test: code/labs/ch01/test_l01_01_stone.py

Uruchom: `cd code && uv run pytest -k l01_01_stone`

1.3 Model, który robi kroki

Wzór na kamień pozwala przeskoczyć od razu do dowolnej chwili. Większość modeli na to nie pozwala. Nalej herbatę o temperaturze 90 °C w pokoju, w którym jest 20 °C. Isaac Newton opisał w 1701 roku, jak stygnie, a w postaci, której używa ta książka, jego reguła mówi: *co minutę herbata traci stały ułamek różnicy między sobą a pokojem*. Niech ten ułamek wynosi 0,1, czyli jedną dziesiątą.

Zatrzymaj się i pomyśl 1.2

Na początku różnica wynosi $90 - 20 = 70$ stopni. Jaka będzie temperatura minutę później?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. 83 °C. Jedna dziesiąta różnicy to 7, a $90 - 7 = 83$.

Żeby znać temperaturę po dwóch minutach, robisz to samo jeszcze raz, z 83: różnica wynosi teraz 63, jedna dziesiąta z niej to 6,3, a herbata ma 76,7. Nie ma skrótu, który powinieneś znać. Reguła mówi, jak przejść od jednej minuty do następnej, a żeby dojść gdziekolwiek indziej, robisz krok po kroku.

code/ch01/cooling_cup.py

```
9 ROOM = 20.0    # degrees Celsius
10 K = 0.1        # the fraction of the gap to the room lost each minute
11
12
13 def next_minute(temp: float) -> float:
14     """The temperature one minute later."""
15     return temp - K * (temp - ROOM)
```

Listing 1.3: Reguła następnej minuty

Funkcja `next_minute` to cały model. Dostaje temperaturę teraz i zwraca temperaturę minutę później. Żeby zobaczyć przyszłość, wywołujesz ją raz za razem, podając jej z powrotem każdą odpowiedź:

code/ch01/cooling_cup.py

```
21 temp = 90.0
22 for minute in range(31):
23     if minute % 5 == 0:
24         print(f"minute {minute:2d}: {temp:5.1f} C")
25     temp = next_minute(temp)
```

Listing 1.4: Każda odpowiedź wraca na wejście, trzydzieści razy

```
minute 0: 90.0 C
minute 5: 61.3 C
minute 10: 44.4 C
```

```
minute 15: 34.4 C
minute 20: 28.5 C
minute 25: 25.0 C
minute 30: 23.0 C
```

Wiersz `temp = next_minute(temp)` to najważniejszy wiersz tej książki i zobaczysz go w każdym rozdziale pod inną nazwą. Zastępuje stan następnym stanem. Robienie tego raz za razem nazywa się *iteracją*, a rozdział 2 jest wyłącznie o niej.

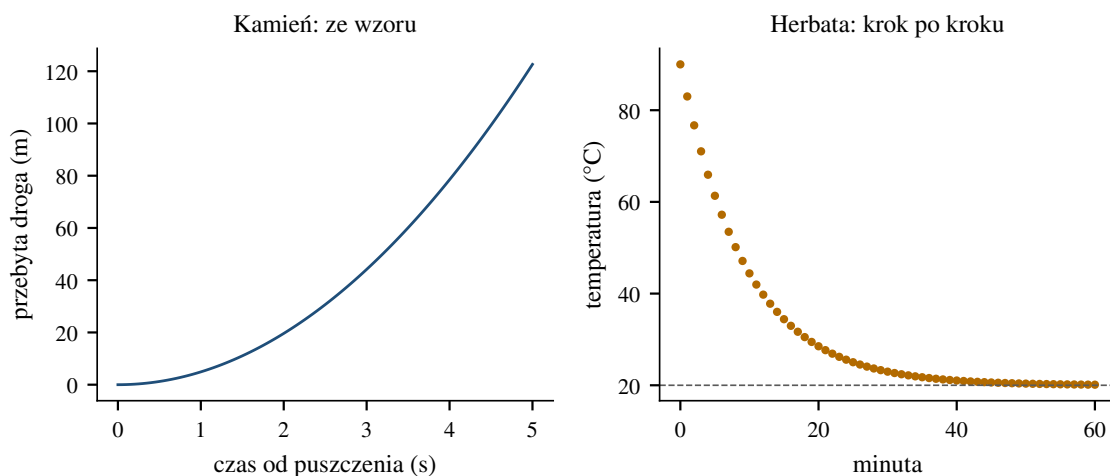
Zatrzymaj się i pomyśl 1.3

Z wydruku wynika, że herbata ma więcej niż 60 °C w minucie 5. Czy ma mniej niż 60 w minucie 6? Co trzeba zrobić, żeby mieć pewność?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tak: pierwszy raz spada poniżej 60 w minucie 6. Żeby mieć pewność, robisz regułą jeszcze jeden krok albo prosisz program o każdą minutę zamiast co piątą.

Tak zadaje się pytanie modelowi krokowemu: robisz kroki, aż pojawi się odpowiedź. Dlatego też modele krokowe były rzadkie przed komputerami, a teraz są wszędzie. Maszynie nie przeszkadza milion kroków.



Rysunek 1.1: Dwa modele tego samego rodzaju pytania. Wzór na kamień podaje dowolną chwilę od razu; reguła herbaty podaje następną minutę, a krzywa to jej kroki połączone ze sobą.

Laboratorium 1.2 – Kiedy da się to wypić?

Napisz `cool`, które stosuje regułę przez zadaną liczbę minut i zapamiętuje każdą temperaturę, oraz `minutes_until`, które odpowiada na pytanie, jakie człowiek naprawdę zadaje. Twoja odpowiedź dla kubka o 90 °C, pokoju o 20 i ułamka 0,1 powinna się zgadzać z tą powyżej.

```
Plik startowy: code/labs/ch01/l01_02_cooling.py  
Test: code/labs/ch01/test_l01_02_cooling.py  
Uruchom: cd code && uv run pytest -k l01_02_cooling
```

1.4 Błędny i użyteczny

Żaden z tych modeli nie jest prawdziwy. Wzór na kamień pomija powietrze, a piórko upuszczone obok kamienia pokazuje, jak bardzo może to mieć znaczenie. Reguła herbaty pomija kubek, spodek, parę i twoją łyżeczkę. Zmierz dokładnie, a przekonasz się, że oba są błędne.

Zatrzymaj się i pomyśl 1.4

Skoro wzór na kamień pomija powietrze, a powietrze oczywiście ma znaczenie, to czy ten wzór jest bezużyteczny?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Dla kamienia spadającego kilka sekund jest dokładny co do szerokości dłoni; dla piórka myli się bardzo. Model jest użyteczny do pytań, na które odpowiada wystarczająco dobrze, a twoim zadaniem jest wiedzieć, które to pytania.

Pułapka

„Model, który jest błędny, jest bezużyteczny.” Każdy model jest błędny, bo każdy coś pomija – właśnie dlatego jest dość mały, żeby go używać. Często cytowane zdanie statystyka George’a Boxa mówi to samo krócej: wszystkie modele są błędne, ale niektóre są użyteczne. Użyteczne pytanie nigdy nie brzmi „czy jest prawdziwy?”, tylko „czy jest wystarczająco dobry do tego pytania, w tym zakresie?”. Zapamiętaj to pytanie. Część II tej książki dotyczy zakresu, w którym model bez żadnego błędu w regule przestaje być wystarczająco dobry, a powód nie jest tym, czego byś się spodziewał.

Model można sprawdzić pomiarem, i to w obie strony. Upuść kamień z dziesięciometrowego muru, a wzór powie, że wyląduje po 1,43 sekundy. Zmierz prawdziwy spadek, dostań trochę inną liczbę, a możesz odwrócić wzór i zapytać, ile musiałoby wynosić g . To trzecia funkcja z pierwszego laboratorium powyżej i tak model zdobywa zaufanie: formułując przewidywania na tyle ostre, że mogłyby okazać się błędne – i nie okazując się.

1.5 Co dzieje się z małym błędem?

Oto pytanie, na którym opiera się cała książka, zadane najłagodniejszemu z jej modeli. Nigdy nie znasz stanu początkowego dokładnie. Termometr pokazuje 90 °C, a herbata ma naprawdę 91. O ile pomylisz się pół godziny później?

Zatrzymaj się i pomyśl 1.5

Dwa kubki zaczynają z różnicą jednego stopnia i stygną według tej samej reguły. Po trzydziestu minutach różnica między nimi jest większa niż jeden stopień, wciąż równa jednemu stopniowi, czy mniejsza?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniejsza: około 0,042 °C. Co minutę różnica jest mnożona przez 0,9, bo każdy kubek traci jedną dziesiątą własnej odległości od temperatury pokoju.

code/ch01/two_cups.py

```
13 a, b = 90.0, 91.0
14 for minute in range(31):
15     if minute % 10 == 0:
16         print(f"minute {minute:2d}: gap {b - a:.4f} C")
17     a, b = next_minute(a), next_minute(b)
```

Listing 1.5: Dwa kubki różniące się o jeden stopień

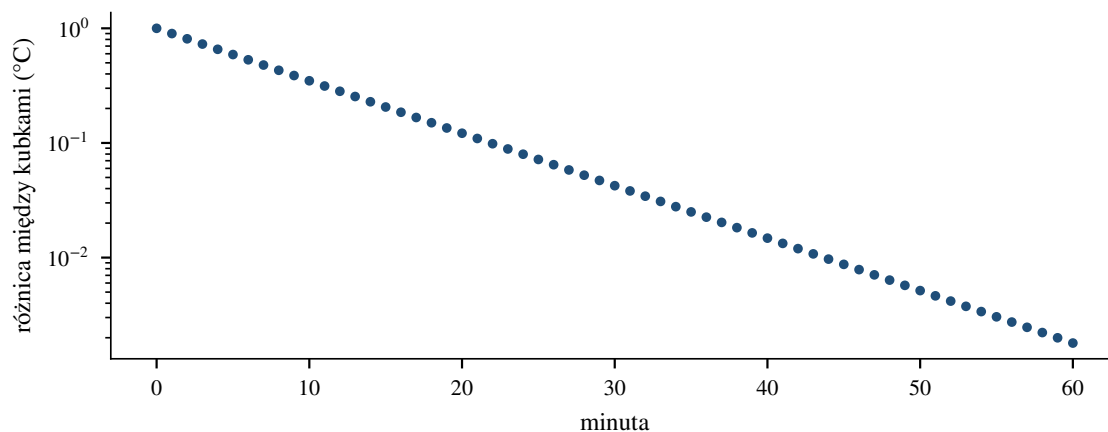
```
minute 0: gap 1.0000 C
minute 10: gap 0.3487 C
minute 20: gap 0.1216 C
minute 30: gap 0.0424 C
```

Błąd nie tylko pozostaje mały; znika. Co minutę jest mnożony przez 0,9, więc po dziesięciu minutach wynosi mniej więcej trzecią część stopnia, a po 44 minutach mniej niż setną część. W skali logarytmicznej, w której każdy krok w dół to dziesięciokrotność, różnica spada wzdłuż linii prostej:

Zapamiętaj tę prostą. Model, którego błędy maleją, wybacz ci, że nie znasz stanu początkowego. Model kamienia jest prawie równie łaskawy: pomył się o jedną dziesiątą sekundy co do chwili puszczenia, a twoje przewidywanie będzie przesunięte o stały czas na zawsze, nigdy o więcej. Większość matematyki, którą poznałeś w szkole, opisuje takie układy i dlatego myśl, że przyroda jest przewidywalna, wydaje się zdrowym rozsądkiem.

Gdzie to spotkasz

Dawkowanie na ulotce leku opiera się na modelu tego rodzaju. Większość leków jest usuwana z krwi w mniej więcej stałym ułamku na godzinę, dokładnie tak, jak herbata traci stały ułamek swojej różnicy (alkohol to słynny wyjątek), więc dawka



Rysunek 1.2: Różnica między dwoma kubkami, które zaczęły z różnicą jednego stopnia. W skali logarytmicznej spada wzdłuż linii prostej: co minutę znika ten sam ułamek.

przyjęta trochę za późno albo trochę za duża zostawia błąd, który maleje, a nie rośnie. Ta wyrozumiałość sprawia, że schemat dawkowania w ogóle da się wydrukować na opakowaniu.

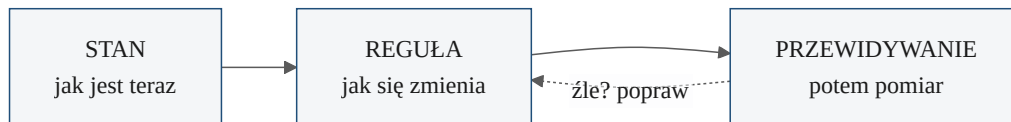
1.6 Mechanizm zegara

W XVIII wieku modele takie jak te dokonały czegoś zdumiewającego. Edmond Halley, korzystając z Newtonowskich praw ruchu i grawitacji, dowodził, że komety widziane w latach 1531, 1607 i 1682 to jeden obiekt na jednej orbicie, i przewidywał, że wróci około 1758 roku. Zmarł w 1742. Kometę zobaczono ponownie w noc Bożego Narodzenia 1758 roku i od tamtej pory nosi jego imię.

Reguła na papierze sięgnęła siedemdziesiąt sześć lat w przyszłość i miała rację. Pierre-Simon Laplace wyciągnął z tego wniosek w 1814 roku, we fragmencie, który w wolnym przekładzie brzmi:

Umysł, który w danej chwili znałby wszystkie siły ożywiające przyrodę i wzajemne położenie wszystkich tworzących ją bytów – umysł dość rozległy, by poddać te dane analizie – objąłby jednym wzorem ruchy największych ciał wszechświata i najlżejszego atomu; nic nie byłoby dla niego niepewne, a przyszłość, tak jak przeszłość, stałaby mu przed oczami.

Ten wyobrażony umysł nazywa się *demonem Laplace’a*, a obraz, który za nim stoi, *wszechświatem jak mechanizm zegara*: znaj regułę i teraźniejszość, a przyszłość wynika z nich tak pewnie jak ruch wskazówek zegara. W 1960 roku fizyk Eugene Wigner pisał o „niepojętej skuteczności matematyki w naukach przyrodniczych” i miał na myśli to samo zdumienie: reguły spisane przez ludzi przy biurkach wciąż okazują się mieć rację co do świata.



Rysunek 1.3: Każdy model w tej książce ma ten kształt. Stan wchodzi, reguła zamienia go w przewidywanie, a pomiar decyduje, czy mu ufać.

Zatrzymaj się i pomyśl 1.6

Załóżmy, że reguła jest dokładna – nie w przybliżeniu poprawna, tylko idealnie poprawna – i że ją znasz. Czy możesz wtedy przewidywać przyszłość dowolnie daleko?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tylko jeśli dokładnie znasz też stan początkowy, a nigdy go nie znasz. Dla herbaty i kamienia nie ma to prawie znaczenia, bo błąd stanu początkowego maleje albo pozostaje taki sam. Czy zawsze zachowuje się tak łaskawie – na to pytanie odpowiada reszta tej książki.

Większość ludzi odpowiada „tak”, i tak samo odpowiadała większość naukowców przez dwieście lat po Laplace’u. Zapamiętaj swoją odpowiedź. W rozdziale 5 poznasz regułę długości jednej linijki, prostszą niż reguła herbaty, której błędy wcale nie maleją; w rozdziale 6 zmierzysz, jak szybko rosną; a pod koniec książki będziesz umieć powiedzieć, liczbami, które sam policzysz, dlaczego niektóre rzeczy da się przewidzieć na sto lat naprzód, a innych nawet na dwa tygodnie. To jest teoria chaosu, i nie jest to teoria nieładu. To matematyka dokładnie tego pytania.

Podsumowanie

- Każdy model ma trzy części: **stan** (liczby, które mówią, jak jest teraz), **regułę** (jak się zmieniają albo jak zamieniają się w odpowiedź) i **przewidywanie** (co mówi reguła).
- Niektóre modele to wzory, które da się obliczyć dla dowolnej chwili. Większość to **reguły krokowe**: mówią, jak przejść od teraz do następnej chwili, a do przyszłości dochodzisz, stosując je raz za razem. To jest iteracja.
- Każdy model jest błędny, bo każdy coś pomija. Użyteczne pytanie brzmi: czy jest wystarczająco dobry do tego pytania, w tym zakresie.
- W przypadku stygnącej herbaty błąd stanu początkowego **maleje** co minutę o ten sam czynnik; w przypadku kamienia pozostaje taki sam. Takie modele wybaczą ci, że nie znasz stanu początkowego dokładnie.
- Demon Laplace’a to zegarmistrzowskie twierdzenie: znaj dokładnie regułę i teraźniejszość, a przyszłość z nich wynika. Reszta książki je sprawdza.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

nazwać stan, regułę i przewidywanie każdego modelu, który ktoś ci pokaże	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
uruchomić model napisany w Pythonie i przeczytać to, co wypisuje	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co znaczy, że model jest błędny, i dlaczego błędny model wciąż może być użyteczny	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co dzieje się z małym błędem stanu początkowego w dwóch modelach z tego rozdziału	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
przytoczyć zegarmistrzowskie twierdzenie Laplace'a z 1814 roku i pytanie, które reszta książki mu zadaje	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Nazwij stan, regułę i przewidywanie rozkładu jazdy autobusu.
2. Ile przeleci kamień w ciągu pierwszych dwóch sekund, bez oporu powietrza? A w ciągu czterech?
3. Herbata o 80°C stygnie w pokoju o 20°C , tracąc co minutę jedną dziesiątą różnicy. Jaką ma temperaturę po jednej minucie, a jaką po dwóch?
4. Dwa kubki zaczynają z różnicą dwóch stopni w tym samym pokoju i stygną według tej reguły. O ile różnią się po jednej minucie?
5. Dlaczego model, o którym wiadomo, że jest błędny, wciąż może być użyteczny?
6. Co musiałby znać demon Laplace'a, żeby przewidzieć przyszłość, i której z tych rzeczy według reszty książki nigdy nie da się znać dokładnie?

Reguła następnego kroku

Lekarz przepisuje ci jedną tabletkę każdego ranka. Każda tabletkę wprowadza do krwi 100 miligramów leku, a do następnego ranka organizm usuwa połowę tego, co w niej było. Bierzesz jedną dziennie przez miesiąc, rok, dziesięć lat: czy ilość leku we krwi rośnie tak długo, jak długo bierzesz tabletki?

Zapamiętaj swoją odpowiedź. To pytanie dotyczy reguły stosowanej raz za razem – połowa, tabletkę; połowa, tabletkę – a reguły tego samego rodzaju rządzą kontami oszczędnościowymi, stygnącą herbatą, populacjami, kredytami i hotelowym prysznicem. Rozdział 1 nazwał wiersz `temp = next_minute(temp)` najważniejszym wierszem tej książki. Ten rozdział mówi o tym, co ten wiersz robi, gdy puścisz go w nieskończoność, a kończy się tym, że potrafisz powiedzieć, co cała rodzina reguł robi na dłuższą metę, wcale ich nie uruchamiając.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- zapisać regułę następnego kroku w notacji tej książki i wyznaczyć jej orbitę, ręcznie i w Pythonie
- rozpoznać regułę liniową, która stoi za oszczędnościami, lekami i stygnięciem, i odczytać jej zachowanie na dłuższą metę z jednej liczby
- odpowiedzieć logarytmem na pytanie „ile kroków, zanim. . .”
- znaleźć stan, w którym reguła stoi w miejscu, i rozstrzygnąć, czy zostaje w nim po lekkim pchnięciu
- zmierzyć nachylenie reguły w punkcie i odczytać diagram pajęczynowy

2.1 Dziś, jutro, pojutrze

Wpłać 1000 złotych na konto oprocentowane na 5 procent rocznie i nie ruszaj ich. Po roku bank dopisze 50, czyli 5 procent z 1000, i będziesz mieć 1050. Po drugim roku dopisze 5 procent od 1050 i będziesz mieć 1102,5; po trzecim 1157,625. Co roku to samo dzieje się z tym, co jest na koncie.

Oznacz stan konta po n latach przez x_n , co czytamy „ x z indeksem n ”: małe n mówi, który to rok, i niczego nie mnoży. Tak więc $x_0 = 1000$ to punkt wyjścia, $x_1 = 1050$ to stan rok później, a reguła banku to

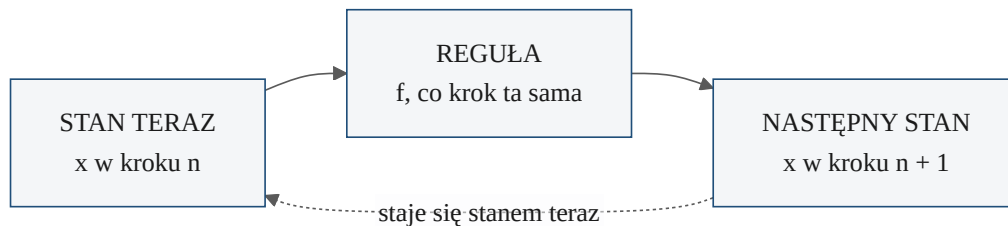
$$x_{n+1} = 1,05 x_n,$$

czyli stan w przyszłym roku to 1,05 razy stan w tym roku, co roku.

Każda reguła krokowa w tej książce ma taki kształt. Jest *stan*, liczba x mówiąca, jak jest teraz; jest *reguła* f , która oznacza „cokolwiek robisz ze stanem, żeby dostać następny”; i jest następny stan, czyli f zastosowane do obecnego:

$$x_{n+1} = f(x_n).$$

Gdy numer kroku nie ma znaczenia, regułę zapisuje się ze strzałką, $x \mapsto 1,05x$, co czytamy „ x przechodzi w 1,05 razy x ”. Lista $x_0, x_1, x_2, \dots$ to *orbita* punktu x_0 , a wytwarzanie jej, przez podawanie każdej odpowiedzi z powrotem jako następnego pytania, to *iteracja*.



Rysunek 2.1: Iteracja. Reguła zamienia obecny stan w następny, który staje się obecnym stanem; na każdym kroku działa ta sama reguła.

Zatrzymaj się i pomyśl 2.1

Weź regułę $x \mapsto 2x + 1$: podwój, potem dodaj jeden. Zaczynając od $x_0 = 1$, ile wynoszą x_1, x_2 i x_3 ?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. 3, 7 i 15. Dwa razy 1 plus jeden to 3; dwa razy 3 plus jeden to 7; dwa razy 7 plus jeden to 15.

Do x_3 nie da się dojść inaczej niż przez x_1 i x_2 : sto kroków naprzód to sto kroków, żmudnych na papierze i bez wysiłku dla maszyny.

chaoslab

Biblioteka książki, [chaoslab](#), robi te kroki za ciebie. Jej orbit to pętla z rozdziału 1 z rachunkowością załatwioną raz na zawsze:

code/src/chaoslab/iterate.py

```

0 def orbit(f: Rule, x0: float, n: int) -> list[float]:
1     """Apply the rule `f` n times to x0 and keep every state.
2
3     The list has n + 1 entries: the start, then each step after it.
4     """
5     xs = [x0]
6     for _ in range(n):
7         xs.append(f(xs[-1]))
8     return xs

```

Listing 2.1: Zastosuj regułę n razy i zachowaj każdy stan

Jej partnerka `linear(a, b)` zwraca regułę $x \mapsto ax + b$, gotową do podania funkcji `orbit`.

2.2 Jedna prosta, trzy losy

Reguła oszczędności tylko mnoży. Wiele reguł mnoży, a potem dodaje:

- **Herbata** z rozdziału 1 traci co minutę jedną dziesiątą swojej różnicy do pokoju o temperaturze 20 stopni: z x przechodzi w $x - 0,1(x - 20)$, co po uporządkowaniu daje $0,9x + 2$.
- **Tabletki**: połowa, potem dodaj 100, czyli $x \mapsto 0,5x + 100$.
- **Populacja**, w której urodzenia co roku przewyższają zgony o jedną piątą: $x \mapsto 1,2x$.
- **Plotka**, którą każdy, kto ją zna, przekazuje dziennie jednej nowej osobie: $x \mapsto 2x$.

Wszystkie mają jeden kształt, $x \mapsto ax + b$: pomnóż przez stałą liczbę a , potem dodaj stałą liczbę b . Jej wykres, z obecnym stanem na osi poziomej i następnym na pionowej, jest linią prostą, dlatego nazywa się ją regułą *liniową* – choć, jak zaraz zobaczysz, to, co robi w czasie, jest przeważnie krzywe. Uruchom cztery takie reguły z tego samego miejsca:

code/ch02/fates.py

```

13 RULES = {
14     "2x": linear(2.0, 0.0),          # multiply by 2
15     "0.5x": linear(0.5, 0.0),        # multiply by 0.5
16     "0.5x+5": linear(0.5, 5.0),      # multiply by 0.5, add 5
17     "-0.5x+15": linear(-0.5, 15.0),  # multiply by -0.5, add 15
18 }
19
20
21 def main() -> None:
22     runs = [orbit(rule, 1.0, 8) for rule in RULES.values()]
23     print(" n" + "".join(f"{name:>11}" for name in RULES))

```

```

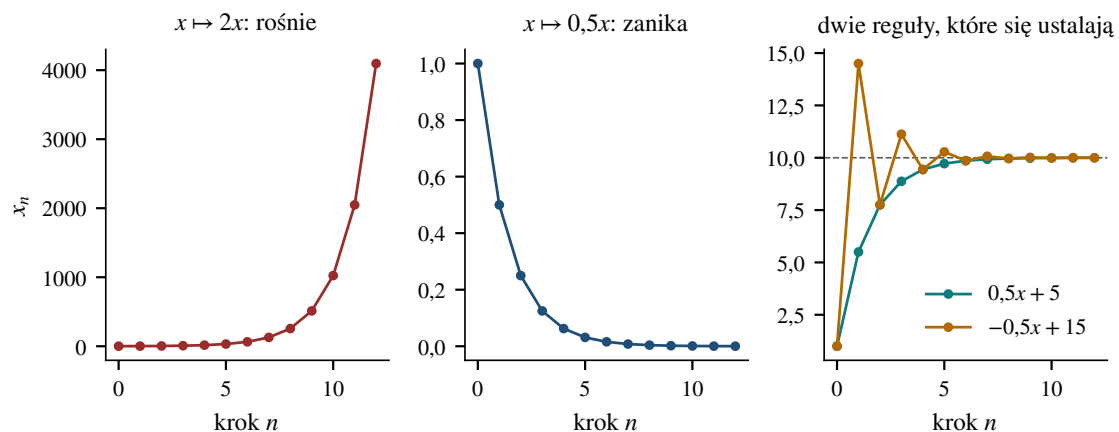
24 for n in range(9):
25     print(f"{n:2d}" + " ".join(f"{run[n]:11.4f}" for run in runs))

```

Listing 2.2: Cztery reguły liniowe iterowane obok siebie z tego samego startu

n	2x	0.5x	0.5x+5	-0.5x+15
0	1.0000	1.0000	1.0000	1.0000
1	2.0000	0.5000	5.5000	14.5000
2	4.0000	0.2500	7.7500	7.7500
3	8.0000	0.1250	8.8750	11.1250
4	16.0000	0.0625	9.4375	9.4375
5	32.0000	0.0312	9.7188	10.2812
6	64.0000	0.0156	9.8594	9.8594
7	128.0000	0.0078	9.9297	10.0703
8	256.0000	0.0039	9.9648	9.9648

Pierwsza kolumna podwaja się w nieskończoność, a druga maleje o połowę ku zeru. Trzecia skrada się do 10 od dołu. Czwarta też ustala się na 10, ale skacząc z jednej strony na drugą, a każdy skok jest mniejszy od poprzedniego: to trzecia kolumna w lustrzanym odbiciu. Oto trzy losy reguły liniowej – rosnąć bez ograniczeń, zanikać do zera albo się ustalić – a ujemny mnożnik dodaje naprzemiennność, ale nie czwarty los.



Rysunek 2.2: Trzy losy reguły liniowej przy tym samym starcie. Jedna rośnie bez ograniczeń, jedna zanika, a dwie ustalają się na tej samej liczbie: jedna się do niej skrada, druga skacze z boku na bok.

Zatrzymaj się i pomyśl 2.2

Wróćmy do konta oszczędnościowego. Chcesz znać stan po dziesięciu latach, nie robiąc dziesięciu kroków. Jakie jedno obliczenie go daje?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Pomnóż 1000 przez 1,05 dziesięć razy z rzędu:

$$x_{10} = 1,05^{10} \times 1000, \text{ co daje } 1628,89.$$

Małe uniesione 10 to *potęga*: $1,05^{10}$ to 1,05 pomnożone przez siebie dziesięć razy. Gdy reguła tylko mnoży, $x \mapsto ax$, jej orbita ma wzór, $x_n = a^n x_0$, i można przeskoczyć do dowolnego kroku. Wzrost, w którym każdy krok mnoży przez tę samą liczbę, nazywa się *wykładniczym*; jego malejący bliźniak, z a między 0 a 1, to *zanik wykładniczy*.

2.3 Powolne eksplozje i liczenie kroków

Słowo „wykładniczy” stało się synonimem słowa „szybki”. Oto test tego przekonania.

Zatrzymaj się i pomyśl 2.3

Dwa konta oszczędnościowe zaczynają od 1000. Konto A co roku dopisuje 50. Konto B rośnie o 2 procent rocznie, czyli wykładniczo. Które jest do przodu po trzydziestu latach?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Konto A, z $1000 + 30 \times 50 = 2500$ wobec 1811 na koncie B. B wyprzedza A dopiero w roku 83.

Jeśli wybrałeś B, rozumowałeś na podstawie słowa, a nie reguły: 2 procent z 1000 to 20, mniej niż 50 konta A, a B potrzebuje dziesięcioleci wzrostu na własnym wzroście, żeby je dogonić.

Pułapka

„Wzrost wykładniczy to szybki wzrost.” To wzrost proporcjonalny do tego, co już jest, a mnożnik może być bardzo bliski jedności. Przy 1 procencie rocznie suma podwaja się po mniej więcej 70 latach. Wzrost wykładniczy gwarantuje nie szybkość, tylko nieuchronność: jakkolwiek mała byłaby stopa, prześcignie każdy wzrost, który dodaje stałą kwotę – w końcu, a to „w końcu” może trwać całe życie.

Pożyteczne pytanie o wzrost wykładniczy brzmi „ile kroków, zanim się podwoi?”, a odpowiedź nie zależy od punktu startu.

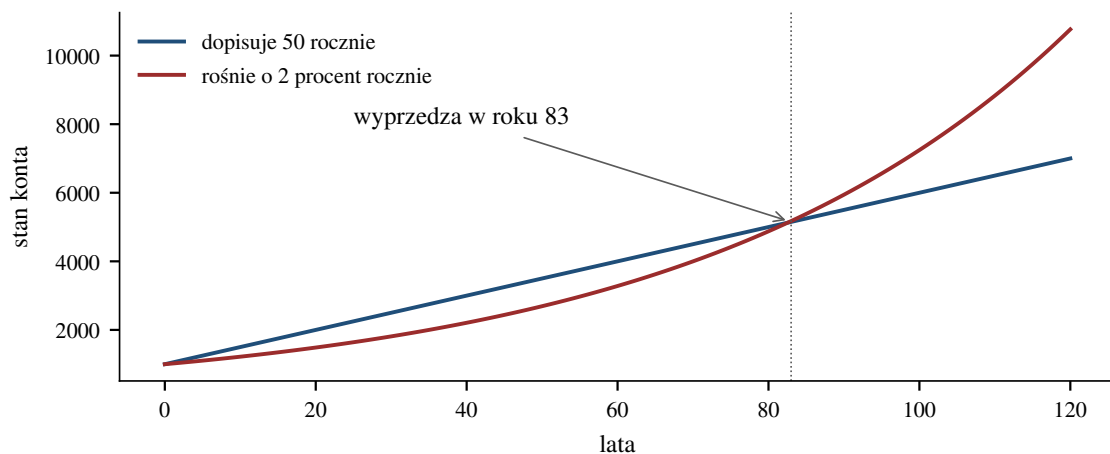
Zatrzymaj się i pomyśl 2.4

Przy 7 procentach rocznie po mniej więcej ilu latach suma się podwaja: po około 5, około 10 czy około 14?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Po około 10. Licząc pełne lata, 1000 po raz pierwszy przekracza 2000 w roku 11; dokładna odpowiedź to 10,24 roku.

Jest na to przycisk. Pytanie brzmi: dla jakiego n zachodzi $1,07^n = 2$? Na dokładnie takie pytania odpowiada *logarytm*. Kalkulator ma przycisk oznaczony $\ln$, logarytm naturalny, a



Rysunek 2.3: Dodawanie stałej kwoty wobec wzrostu o stały ułamek. Konto wykładnicze jest z tyłu przez większą część stulecia, a potem już nie.

jedynie, co musisz o nim wiedzieć, to co robi z potęgą:

$$\ln(a^n) = n \ln a.$$

Zamienia „pomnóż przez a , n razy” na „dodaj $\ln a$, n razy”. Zlogarytmuj obie strony równania $a^n = 2$, a dostaniesz $n \ln a = \ln 2$, czyli

$$n = \frac{\ln 2}{\ln a} :$$

liczba kroków to logarytm tego, jak daleko chcesz dojść, podzielony przez logarytm jednego kroku. Skoro $\ln 2 = 0,693$, a $\ln 1,07 = 0,0677$, wychodzi 10,24.

code/ch02/doubling.py

```

14 def steps_to_double(rate: float) -> int:
15     """Whole steps of x -> (1 + rate) * x until x has at least doubled."""
16     x, steps = 1.0, 0
17     while x < 2.0:
18         x = (1.0 + rate) * x
19         steps += 1
20     return steps
21
22
23 def doubling_time(rate: float) -> float:
24     """The same question asked of the logarithm: ln 2 / ln(1 + rate)."""
25     return math.log(2.0) / math.log(1.0 + rate)
26
27
28 def main() -> None:
29     print("rate    counted    logarithm    rule of 70")
30     for percent in (1, 2, 3, 5, 7, 10):
31         rate = percent / 100
32         print(f"{percent:3d}%    {steps_to_double(rate):7d}"
33               f"    {doubling_time(rate):9.2f}    {70 / percent:10.1f}")

```

Listing 2.3: Po ilu latach stan konta się podwoi: liczone i z logarytmu

rate	counted	logarithm	rule of 70
1%	70	69.66	70.0
2%	36	35.00	35.0
3%	24	23.45	23.3
5%	15	14.21	14.0
7%	11	10.24	10.0
10%	8	7.27	7.0

Liczba kroków to logarytm zaokrąglony w górę. Przy 2 procentach to 35,00 wobec 36: po 35 latach na koncie jest 1,9999 raza tyle, ile wpłacono, o włos mniej niż podwojenie. Ostatnia kolumna, *reguła 70*, dzieli 70 przez liczbę procent; działa, bo logarytm jednego kroku jest bliski samej małej stopie ($\ln 1,07$ to prawie 0,07), a $100 \times \ln 2$ to mniej więcej 70.

Z maleniem jest tak samo. Lek, z którego organizm co godzinę usuwa jedną piątą, zachowuje co godzinę 0,8, a czas, po którym zostaje połowa, jego *okres półtrwania*, wynosi $\ln 0,5 / \ln 0,8$, czyli około 3,1 godziny.

Laboratorium 2.1 – Ile kroków do celu?

Napisz `steps_until`, która liczy kroki reguły $x \mapsto ax$ od 1, aż x osiągnie cel, oraz `time_until`, która zadaje to samo pytanie logarytmowi, dla wzrostu i dla zaniku. Potem napisz `half_life`. Logarytm naturalny w Pythonie to `math.log`.

Plik startowy: `code/labs/ch02/l02_01_until.py`

Test: `code/labs/ch02/test_l02_01_until.py`

Uruchom: `cd code && uv run pytest -k l02_01_until`

2.4 Gdzie reguła stoi w miejscu

Teraz pytanie z początku rozdziału. Stanem jest ilość leku we krwi zaraz po porannej tabletki, w miligramach: $x \mapsto 0,5x + 100$, od $x_0 = 100$.

Zatrzymaj się i pomyśl 2.5

Policz kilka pierwszych poranków: 100, potem połowa tego plus 100 i tak dalej. Czy ilość leku rośnie tak długo, jak długo bierzesz tabletki?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Idzie 100, 150, 175, 187,5, 193,75 i dalej, zbliżając się do 200, ale nigdy go nie osiągając.

Dlaczego 200? Bo reguła zostawia 200 bez zmian: połowa z 200 to 100, a tabletki od razu dokłada 100 z powrotem. Stan, którego reguła nie zmienia, to *punkt stały*, a znajdujesz go, pytając, które x spełnia

$$x = f(x).$$

Dla tabletek to $x = 0,5x + 100$; odejmij $0,5x$ od obu stron, a dostaniesz $0,5x = 100$, czyli

$x = 200$. Dla każdej reguły liniowej te same kroki dają

$$x^* = \frac{b}{1-a},$$

gdzie gwiazdka oznacza punkt stały. (Jeśli a wynosi dokładnie 1, reguła tylko dodaje i nigdzie nie stoi w miejscu.) Punkt stały reguły herbaty leży w $2/0,1 = 20$ stopniach: w temperaturze pokoju.

code/ch02/drug.py

```

12 KEPT = 0.5      # the fraction of yesterday's drug still in the blood
13 DOSE = 100.0    # milligrams in one tablet
14 tablets = linear(KEPT, DOSE)
15 steady = DOSE / (1.0 - KEPT)    # the fixed point: solve x = 0.5x + 100
16
17
18 def main() -> None:
19     print("day    in blood    short of the fixed point")
20     for day, x in enumerate(orbit(tablets, DOSE, 8)):
21         print(f"{day:3d}    {x:8.2f}    {steady - x:8.2f}")

```

Listing 2.4: Jedna tabletka co rano i odległość od punktu stałego

day	in blood	short of the fixed point
0	100.00	100.00
1	150.00	50.00
2	175.00	25.00
3	187.50	12.50
4	193.75	6.25
5	196.88	3.12
6	198.44	1.56
7	199.22	0.78
8	199.61	0.39

Niedobór w prawej kolumnie co rano maleje o połowę, bo organizm usuwa połowę tego, co jest, a tabletka zawsze dokłada tych samych 100. Po 7 dniach od 200 dzieli cię mniej niż jeden miligram, a dokładnie tam nie dotrzesz nigdy. Ilość leku nie rośnie w nieskończoność; wyrównuje się w punkcie stałym.

Gdzie to spotkasz

Farmakolodzy nazywają ten punkt stały *stanem stacjonarnym*, a praktyczna reguła, której uczy się lekarzy i farmaceutów, mówi, że lek przyjmowany w regularnych odstępach osiąga go po czterech lub pięciu okresach półtrwania. To tabela powyżej: po czterech okresach półtrwania niedobór zmalał o połowę cztery razy, co daje poziom 93,8 procent stanu stacjonarnego, a po pięciu 96,9. Gdy lek jest potrzebny od razu w pełnej sile, lekarz może podać większą pierwszą dawkę, *dawkę nasycającą*, która zaczyna orbitę w punkcie stałym, a nie poniżej niego.

2.5 Zostać na miejscu albo uciec

Oto punkt stały innego charakteru. Masz 10 000 długu oprocentowanego na 1 procent miesięcznie i spłacasz 100 miesięcznie, więc co miesiąc dług przechodzi w $1,01x - 100$.

Zatrzymaj się i pomyśl 2.6

Co dzieje się z długiem przez lata? A co, jeśli jesteś winien 11 000, przy tym samym oprocentowaniu i tej samej racie?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Przy 10 000 nic, nigdy: odsetki wynoszą dokładnie 100, a twoje 100 je znosi. Przy 11 000 odsetki wynoszą 110, twoje 100 ich nie pokrywa i po dziesięciu latach jesteś winien 13 300.

Punktem stałym tej reguły jest 10 000, ale reguła wcale cię do niego nie przyciąga. Zaczynij tysiąc powyżej, a dług ucieka w górę; zacznij tysiąc poniżej, od 9000, a ucieka w dół, do zera, spłacony po 232 miesiącach, czyli po mniej więcej 19 latach.

Pułapka

„**Punkt stały to miejsce, w którym układ w końcu się znajdzie.**” To miejsce, w którym układ pozostaje, jeśli jest w nim dokładnie. Czy układ w pobliżu zbliża się do niego, czy się oddala, to osobne pytanie, a punkt stały kredytu odpycha każdy inny stan długu. Punkt stały, który przyciąga pobliskie stany, jest *stabilny*; taki, który je odpycha, jest *niestabilny*. Znalezienie punktu stałego mówi ci, gdzie szukać; dopiero jego stabilność mówi, czy cokolwiek tam będzie.

Różnicę robi jedna liczba. Prześledź przez jeden krok odległość między stanem x a punktem stałym x^* :

$$f(x) - x^* = (ax + b) - (ax^* + b) = a(x - x^*).$$

Pierwszy znak równości korzysta z tego, że x^* jest stały, więc $ax^* + b$ równa się x^* . Wyraz b się skraca, a to, co zostaje, mówi: **każdy krok mnoży odległość przez a** . Tabletki ją połowią; kredyt powiększa ją o 1 procent miesięcznie; herbata, z $a = 0,9$, to powód, dla którego dwa kubki z rozdziału 1 zbliżały się do siebie co minutę o jedną dziesiątą.

Liczy się wielkość a , zapisywana $|a|$, która pomija znak: $|-0,5| = 0,5$. Poniżej 1 odległość maleje i punkt stały jest stabilny; powyżej 1 rośnie i punkt stały jest niestabilny. Znak mówi tylko, czy odległość zostaje po tej samej stronie, czy przeskakuje na drugą.

Ujemne a czułeś pod hotelowym prysznicem. Chcesz 38 stopni, woda ma 30, a ty przesadzasz: każdy obrót kurka zmienia temperaturę o półtora raza tyle, ile wynosi błąd, w przeciwną stronę. Reguła to $x \mapsto x - 1,5(x - 38)$, czyli $x \mapsto -0,5x + 57$. Z 30 przechodzisz do 42, 36, 39, 37,5: za gorąco, za zimno, za gorąco, a każde pudło jest o połowę mniejsze od poprzedniego. Szarpnij kurkiem o dwa i pół raza tyle, ile wynosi błąd, a $a = -1,5$: 30, 50, 20, 65, za każdym razem gorzej.

mnożnik a	pchnięcie od x^*	orbita
powyżej 1	rośnie	ucieka, po jednej stronie
między 0 a 1	maleje	zbliża się, z jednej strony
między -1 a 0	maleje, zmieniając stronę	ustala się, przeskakując
poniżej -1	rośnie, zmieniając stronę	ucieka, w coraz szerszych waha- niach

Laboratorium 2.2 – Gdzie się zatrzyma i czy tam zostanie?

Dla reguły $x \mapsto ax + b$ napisz `fixed_point`; `is_stable`, która rozstrzyga, czy pchnięcie od punktu stałego maleje (spróbuj: zacznij trochę obok, zrób jeden krok, porównaj odległości); oraz `steps_to_settle`.

Plik startowy: `code/labs/ch02/l02_02_fixed.py`

Test: `code/labs/ch02/test_l02_02_fixed.py`

Uruchom: `cd code && uv run pytest -k l02_02_fixed`

2.6 Nachylenie: ile razy mnoży się pchnięcie

Mnożnik a ma drugą nazwę, której używa reszta książki. Narysuj regułę tabletek jako wykres, z obecnym stanem na osi poziomej i następnym na pionowej: to prosta $y = 0,5x + 100$. Przesuń się wzdłuż osi poziomej ze 150 do 151, o jedną jednostkę, a prosta wznie się ze 175 do 175,5, o pół jednostki. *Nachylenie* wykresu w punkcie to właśnie to: **o ile przesuwa się wynik na każdą jednostkę przesunięcia wejścia**. Prosta ma wszędzie to samo nachylenie, a dla $x \mapsto ax + b$ to nachylenie wynosi a . Zatem odkrycie z poprzedniego podrozdziału, w słowach, których używa reszta książki, brzmi: *mały błąd jest na każdym kroku mnożony przez nachylenie*.

Krzywa też ma nachylenie, ale zmienia się ono od punktu do punktu, tak jak zbocze jest w jednym miejscu strome, a w innym prawie płaskie. Mierzysz je w punkcie, pchając: przesun wejście o małą wielkość h w obie strony od x , zobacz, o ile przesunął się wynik, i podziel,

$$\text{nachylenie w } x \approx \frac{f(x+h) - f(x-h)}{2h},$$

gdzie $\approx$ znaczy „prawie równe”. W funkcji `slope` z `chaoslab` pchnięcie h to jedna milionowa: przyrost w górę podzielony przez przyrost w bok, na odcinku zbyt krótkim, by krzywa zdążyła się wygiąć.

`code/ch02/slope.py`

```

12 tablets = linear(0.5, 100.0)    # a straight line
13
14
15 def square(x: float) -> float:
16     """A curved rule: x times itself."""
17     return x * x
18
19
20 def main() -> None:
21     for x in (0.0, 150.0, 400.0):

```

```

22 print(f"line   at x = {x:5.1f}: slope {slope(tablets, x):.4f}")
23 for x in (0.5, 1.5, 3.0):
24     print(f"square at x = {x:5.1f}: slope {slope(square, x):.4f}")

```

Listing 2.5: Nachylenie zmierzone pchnięciem, na prostej i na krzywej

```

line   at x =   0.0: slope 0.5000
line   at x = 150.0: slope 0.5000
line   at x = 400.0: slope 0.5000
square at x =   0.5: slope 1.0000
square at x =   1.5: slope 3.0000
square at x =   3.0: slope 6.0000

```

Nachylenie prostej wynosi 0,5, gdziekolwiek je zmierzysz; krzywa reguła $x \mapsto x \times x$ jest tym bardziej stroma, im dalej w prawo.

Zatrzymaj się i pomyśl 2.7

Reguła kwadratu ma nachylenie 3 w punkcie 1,5. Chciałeś podać jej 1,5, a podałeś 1,501, z błędem 0,001. O ile mniej więcej pomyli się wynik?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. O mniej więcej 0,003: błąd mnoży się przez nachylenie.
Dokładnie: $1,501 \times 1,501 - 2,25 = 0,003001$.

To fakt, który stoi za każdą odległością w tym rozdziale, teraz na krzywej: błąd mnoży się przez nachylenie w miejscu, w którym akurat jest. Na prostej ten czynnik się nie zmienia i dlatego ten rozdział może wszystko przewidzieć. W regule kwadratu się zmienia: między 0 a 0,5 nachylenie jest mniejsze niż 1 i błędy maleją, powyżej 0,5 jest większe niż 1 i rosną. Oto więc pytanie, na które ten rozdział nie odpowie. Co robi orbita, jeśli jej reguła jest krzywa, tak że orbita wciąż przechodzi z miejsc, które zmniejszają błędy, do miejsc, które je rozciągają? Rozdział 3 wygina prostą, a rozdział 5 śledzi orbitę, która robi dokładnie to.

2.7 Zobaczyc to: diagram pajęczynowy

Iterację można oglądać na wykresie samej reguły, z ołówkiem i bez rachunków. Narysuj prostą reguły i przekątną $y = x$, której wysokość zawsze równa się położeniu na osi poziomej. Potem:

1. Od x_0 na osi poziomej idź w górę lub w dół do prostej. Wysokość, na którą dojdiesz, to x_1 .
2. Idź poziomo do przekątnej, gdzie wysokość równa się położeniu: stoisz teraz nad x_1 , gotowym do użycia jako następne wejście.
3. W górę lub w dół do prostej daję x_2 ; w bok do przekątnej i tak dalej.

Narożniki tej ścieżki to narysowana orbita. Funkcja `cobweb` z `chaoslab` je wypisuje; dla tabletek, zaczynając od pustego krwiobiegu:

code/ch02/cobweb_steps.py

```

13 tablets = linear(0.5, 100.0)
14 corners = cobweb(tablets, 0.0, 3)
15 print(f"start          ({corners[0][0]:5.1f}, {corners[0][1]:5.1f})")
16 for k, (x, y) in enumerate(corners[1:]):
17     move = "to the line" if k % 2 == 0 else "to the diagonal"
18     print(f"{move:15} ({x:5.1f}, {y:5.1f})")

```

Listing 2.6: Naróżniki pajęczyny dla tabletek

```

start          ( 0.0,  0.0)
to the line    ( 0.0, 100.0)
to the diagonal (100.0, 100.0)
to the line    (100.0, 150.0)
to the diagonal (150.0, 150.0)
to the line    (150.0, 175.0)
to the diagonal (175.0, 175.0)

```

Naróżniki na przekątnej niosą orbitę, każdą liczbę dwa razy: 0, 100, 150, 175. Ścieżka może się zatrzymać tylko tam, gdzie prosta przecina przekątną, bo tam $x = f(x)$: przecięcie to punkt stały, który widać.

Zatrzymaj się i pomyśl 2.8

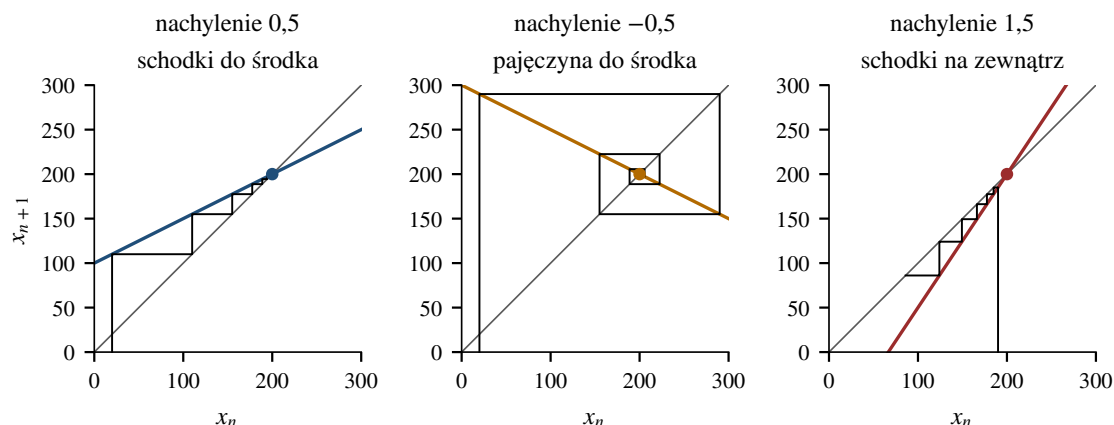
Prosta kredytu, $y = 1,01x - 100$, jest bardziej stroma niż przekątna. Czy ścieżka, która zaczyna trochę z boku przecięcia, wspina się ku przecięciu, czy się od niego oddala?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Oddala się. Każdy krok mnoży odległość od przecięcia przez 1,01, więc ścieżka ląduje za każdym razem trochę dalej, po tej stronie, po której zaczęła.

Gdy nachylenie jest ujemne, ścieżka owija się wokół przecięcia i to od tej pajęczyny diagram wziął nazwę. Trzy panele to narysowana tabela z podrozdziału 2.5.

Masz więc coś rzadkiego w tej książce: pełną zdolność przewidywania. Mając $x \mapsto ax + b$ i punkt startu, możesz bez robienia choćby jednego kroku powiedzieć, co reguła będzie robić zawsze – ustali się w $b/(1-a)$, jeśli $|a| < 1$, ucieknie, jeśli $|a| > 1$, będzie się wahać naprzemiennie, jeśli a jest ujemne – i jak długo to potrwa. Tę pewność odbiera reszta książki: wygnij prostą, a reguła długości jednej linijki potrafi robić rzeczy, których nie przewidzi żaden wzór z tego rozdziału.



Rysunek 2.4: Pajączyny na trzech prostych, które przecinają przekątną w 200. Nachylenie o wielkości mniejszej niż jeden wciąga ścieżkę do przecięcia – jako schodki, gdy nachylenie jest dodatnie, jako pajączynę, gdy ujemne – a nachylenie większe niż jeden ją wypycha.

Podsumowanie

- **Reguła następnego kroku** zamienia obecny stan w następny, $x_{n+1} = f(x_n)$. Powtarzanie jej to **iteracja**; lista stanów to **orbita**.
- Oszczędności, stygnięcie i dawkovanie leków podlegają **regule liniowej**, $x \mapsto ax + b$: rośnie bez ograniczeń, zanika do zera albo się ustala; ujemne a sprawia, że **oscyluje naprzemiennie**.
- Wzrost **wykładniczy** jest proporcjonalny, niekoniecznie szybki. **Logarytm** mówi, ile kroków zajmuje zmiana o czynnik r , $\ln r / \ln a$: stąd czas podwojenia i okres półtrwania.
- **Punkt stały** to stan, którego reguła nie zmienia: rozwiąż $x = f(x)$, co dla reguły liniowej daje $b/(1 - a)$.
- Każdy krok mnoży odległość od punktu stałego przez a : punkt jest **stabilny**, gdy $|a| < 1$, a **niestabilny**, gdy $|a| > 1$, i w niestabilnym nic nie kończy.
- **Nachylenie**, czyli o ile przesuwa się wynik na jednostkę przesunięcia wejścia, to czynnik, przez który mały błąd mnoży się w jednym kroku. Nachylenie krzywej zmienia się od punktu do punktu.
- **Diagram pajączynowy** rysuje orbitę na wykresie reguły; na prostej zbliża się do przecięcia z przekątną, gdy nachylenie ma wielkość mniejszą niż jeden.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

zapisać regułę następnego kroku w notacji tej książki i wyznaczyć jej orbitę, ręcznie i w Pythonie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
rozpoznać regułę liniową, która stoi za oszczędnościami, lekami i stygnięciem, i odczytać jej zachowanie na dłuższą metę z jednej liczby	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
odpowiedzieć logarytmem na pytanie „ile kroków, zanim. . .”	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
znaleźć stan, w którym reguła stoi w miejscu, i rozstrzygnąć, czy zostaje w nim po lekkim pchnięciu	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zmierzyć nachylenie reguły w punkcie i odczytać diagram pajęczynowy	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dla reguły $x \mapsto 3x - 2$ i startu $x_0 = 2$ ile wynoszą x_1 , x_2 i x_3 ?
2. Herbata traci co minutę jedną piątą różnicy do pokoju o temperaturze 20 stopni. Zapisz jej regułę w postaci $x \mapsto ax + b$.
3. Miasto rośnie o 3 procent rocznie. Oszacuj czas podwojenia regułą 70, a potem powiedz, co daje liczenie pełnych lat.
4. Znajdź punkt stały reguły $x \mapsto 0,8x + 4$. Czy jest stabilny?
5. Reguła $x \mapsto 3x - 2$ ma punkt stały w 1. Czy jest stabilny? Co pokazuje pierwsze pytanie?
6. Reguła prysznic $x \mapsto -0,5x + 57$ ma nachylenie $-0,5$. Jest ci o 2 stopnie za zimno. Gdzie jesteś po jednym obrocie kurka?

Sprężenie zwrotne: dlaczego nic nie rośnie wiecznie

W ciepłej szalce z mnóstwem pożywienia bakteria *E. coli* dzieli się na dwie mniej więcej co dwadzieścia minut. Zaczynij od jednej; każda nowa komórka robi to samo, a potem każda z jej potomnych. Gdyby nic nie stało na przeszkodzie, po jakim czasie kolonia ważyłaby więcej niż Ziemia?

Nic takiego się nie dzieje, a ten rozdział mówi, dlaczego. Coś zawsze stawia opór: kończy się pożywienie, zapełnia się przestrzeń, gromadzą się odpady. Gdy wielkość czegoś zmienia tempo, w jakim to rośnie, mamy do czynienia ze sprzężeniem zwrotnym, a sprzężenie zwrotne wygina proste reguły z rozdziału 2 w krzywe. Krzywą trudniej przewidzieć niż prostą, a reszta tej książki opowiada o tym, o ile trudniej. Tutaj spotkasz pierwszą niespodziankę: populacja, której kończy się miejsce, nie zawsze łagodnie się zatrzymuje.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- odróżnić dodatnie sprzężenie zwrotne od ujemnego w układzie opisanym słowami
- zamienić zdanie „wzrost zwalnia, gdy brakuje miejsca” w regułę i ją uruchomić
- sprawdzić, czy skutki dwóch pchnięć danej reguły się sumują
- znaleźć punkty stałe reguły krzywoliniowej i ocenić każdy z nich po zmierzonym nachyleniu
- przewidzieć na podstawie tego nachylenia, co zrobi populacja w pobliżu swojego poziomu ustalonego

3.1 Podwajanie i dlaczego nie może trwać

Rozdział 2 nazwał wykładniczym wzrost, który na każdym kroku mnoży przez tę samą liczbę, i policzył logarytmem, ile kroków trwa podwojenie. Bakteria to właśnie taki wzrost, z mnożnikiem dwa co dwadzieścia minut.

Zatrzymaj się i pomyśl 3.1

Mniej więcej po jakim czasie jej kolonia będzie ważyć więcej niż Ziemia: po stu latach, po roku, po miesiącu czy po dwóch dniach?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Po niecałych dwóch dniach: po 133 podwojeniach, czyli po 44,3 godziny. Jedna komórka waży około jednej bilionowej grama.

Większość ludzi zgaduje rok albo dłużej. Logarytm z rozdziału 2 podałby tę liczbę; listing zamiast tego liczy na liczbach całkowitych, a wiersz `cells = 2 * cells` to cała reguła wzrostu.

code/ch03/doubling.py

```

10 CELL = 1e-12      # grams: one E. coli bacterium weighs about a picogram
11 EARTH = 5.97e27   # grams: the mass of the Earth
12 MINUTES = 20      # one doubling, in a warm dish with food to spare
13
14
15 def doublings_to_outweigh(target: float) -> int:
16     """How many doublings until a colony grown from a single cell weighs
17     more than `target` grams."""
18     cells, n = 1, 0
19     while cells * CELL <= target:
20         cells = 2 * cells
21         n = n + 1
22     return n

```

Listing 3.1: Liczenie podwojeń, aż kolonia przeważy Ziemię

Reszta programu wypisuje stan kolonii co osiem godzin:

hours	cells	grams
0	1.0e+00	1.0e-12
8	1.7e+07	1.7e-05
16	2.8e+14	2.8e+02
24	4.7e+21	4.7e+09
32	7.9e+28	7.9e+16
40	1.3e+36	1.3e+24

heavier than the Earth after 133 doublings, 44.3 hours

W skrócie, którym Python zapisuje bardzo duże i bardzo małe liczby, 4.7e+09 to 4,7 z przecinkiem przesuniętym o dziewięć miejsc w prawo (Python stawia kropkę tam, gdzie po polsku piszemy przecinek), a 1.0e-12 to przecinek przesunięty o dwanaście miejsc w lewo. Po jednej dobie kolonia ważyłaby około 4700 ton; niecałą dobę później więcej niż planeta. Podwajanie myli oko, bo każdy krok dodaje więcej niż wszystkie wcześniejsze razem: połóż jedno ziarnko ryżu na pierwszym polu szachownicy, dwa na drugim i cztery na trzecim, a ostatnie pole zmieści więcej niż pozostałe sześćdziesiąt trzy razem.

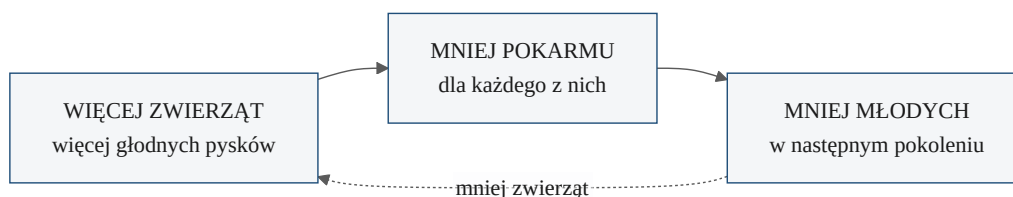
Nic nie rośnie tak długo. W kolbie z bulionem bakterie zatrzymują się na kilku miliardach komórek w każdym mililitrze, gdy kończy się pożywienie i zbierają się odpady. Wzrost, który czymś się żywi, natrafia na granicę tego czegoś.

3.2 Sprzężenie zwrotne

Spójrz jeszcze raz na regułę podwajania. Im więcej jest bakterii, tym więcej podziałów zachodzi w ciągu następnych dwudziestu minut: wielkość kolonii ustala jej własne tempo

wzrostu. Gdy wielkość czegoś zmienia jego własne tempo zmian, mówimy o *sprzężeniu zwrotnym*: skutek wraca i działa na to, co go wywołało.

Sprzężenie zwrotne bywa dwojakie. *Dodatnie sprzężenie zwrotne* pcha dalej w kierunku, w którym sprawy już zmierzają. Zbliź mikrofon do głośnika, do którego jest podłączony, a cichy dźwięk zacznie krążyć w pętli, za każdym razem głośniejszy, aż wycie osiągnie największą głośność, na jaką stać wzmacniacz. *Ujemne sprzężenie zwrotne* stawia opór. Termostat wyłącza ogrzewanie, gdy w pokoju jest za ciepło, i włącza je, gdy jest za zimno. Zatłoczenie to także ujemne sprzężenie zwrotne: im więcej zwierząt, tym mniej pożywienia dla każdego i tym mniej młodych w następnym pokoleniu. Drapieżniki i ofiary tworzą podwójną pętlę: więcej królików wykarmi więcej lisów, więcej lisów zje więcej królików, a mniej królików zagłodzi lisy.



Rysunek 3.1: Zatłoczenie jako ujemne sprzężenie zwrotne. Im więcej jest zwierząt, tym mniej młodych zostawia każde z nich, więc pętla ściąga populację z powrotem.

Zatrzymaj się i pomyśl 3.2

Plotka głosi, że bank może upaść, i część klientów wypłaca pieniądze. Czy to popycha bank ku upadkowi, czy od niego oddala – a więc czy to sprzężenie zwrotne dodatnie, czy ujemne?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Dodatnie. Każda wypłata osłabia bank, a to sprawia, że następny klient chętniej też wypłaci pieniądze. Run na bank sam się napędza.

Run na bank Northern Rock we wrześniu 2007 roku, pierwszy na brytyjski bank od ponad stu lat, skończył się dopiero wtedy, gdy rząd zagwarantował depozyty: ujemne sprzężenie zwrotne, dodane z zewnątrz. Dodatnie sprzężenie kończy się właśnie tak albo na jakiejś granicy, a od granicy zaczyna się ciekawa matematyka.

3.3 Wzrost zwalnia, gdy brakuje miejsca

Oto najprostszy sposób, by wpisać zatłoczenie w regułę. Mierz populację nie w zwierzętach, lecz jako ułamek tego, ile to miejsce mogłoby kiedykolwiek pomieścić. Nazwij go x : $x = 0$ to miejsce puste, a $x = 1$ to takie, w którym zajęty jest każdy okrucuch pożywienia i przestrzeni. Wtedy $1 - x$ to część miejsca, która jest jeszcze wolna.

Gdy miejsca jest pod dostatkiem, przyjmij, że każde zwierzę zostawia r młodych,

które dożyją rozrodu, więc następne pokolenie jest r razy liczniejsze od obecnego: $x \mapsto rx$. Dla $r = 2$ to reguła podwajania. Teraz niech zatłoczenie ją spowolni: pomnóż przez część miejsca, która jest jeszcze wolna:

$$x \mapsto rx(1 - x).$$

Słowami: następne pokolenie to tyle, ile obecne wydałoby na świat, mając miejsca pod dostatkiem, pomniejszone w proporcji do miejsca, które zostało. Dopóki x jest maleńkie, $1 - x$ jest bliskie 1 i reguła jest prawie regułą podwajania. Gdy x rośnie, rośnie z nim hamulec, a przy $x = 1$ następnego pokolenia nie ma wcale. Belgijski matematyk Pierre-François Verhulst zaproponował prawo wzrostu, które zwalnia w ten sposób, w 1838 roku; powyższa reguła to jego pomysł rozpisany na kolejne pokolenia.

code/ch03/crowding.py

```
11 def next_gen(x: float, r: float) -> float:
12     """The next generation, as a fraction of the room: r * x is what the
13     population would breed with room to spare, and (1 - x) is the share
14     of the room still free."""
15     return r * x * (1 - x)
```

Listing 3.2: Reguła zatłoczenia, pokolenie po pokoleniu

Uruchom ją obok zwykłego podwajania, obie od jednej tysięcznej miejsca, przy $r = 2$:

code/ch03/crowding.py

```
21 r = 2.0
22 free = crowded = 0.001
23 print("gen doubling crowded")
24 for gen in range(15):
25     print(f"{gen:3d} {free:8.4f} {crowded:7.4f}")
26     free = r * free
27     crowded = next_gen(crowded, r)
```

Listing 3.3: Podwajanie oraz podwajanie, gdy brakuje miejsca

gen	doubling	crowded
0	0.0010	0.0010
1	0.0020	0.0020
2	0.0040	0.0040
3	0.0080	0.0079
4	0.0160	0.0158
5	0.0320	0.0310
6	0.0640	0.0601
7	0.1280	0.1130
8	0.2560	0.2005
9	0.5120	0.3206
10	1.0240	0.4356
11	2.0480	0.4917
12	4.0960	0.4999
13	8.1920	0.5000
14	16.3840	0.5000

Przez sześć pokoleń kolumn prawie nie da się odróżnić; miejsce zapełnione w kilku procentach nie wydaje się zatłoczone. Potem się rozchodzą. Kolumna podwajania przekracza całe miejsce w pokoleniu 10, co jest niemożliwe, i rośnie dalej, jakby nigdy nic. Kolumna zatłoczenia zwalnia, ustala się na 0,5 i tam zostaje.

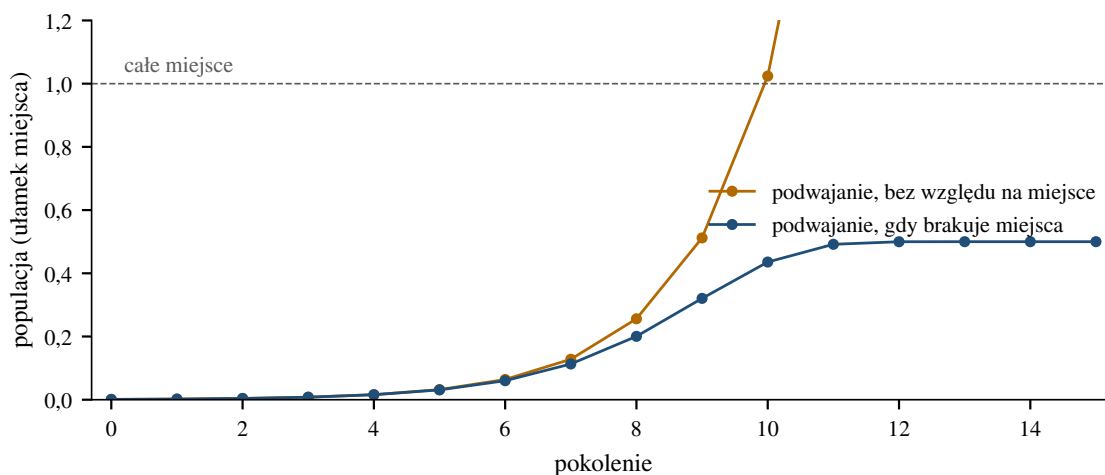
Zatrzymaj się i pomyśl 3.3

Przyjrzyj się kolumnie zatłoczenia. W których pokoleniach populacja przybiera najwięcej: gdy jest mała, gdy jest mniej więcej w połowie drogi do miejsca, w którym się zatrzyma, czy gdy jest już prawie u celu?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniej więcej w połowie drogi. Największy skok jest z pokolenia 8 do pokolenia 9: populacja przybiera wtedy 0,12 miejsca i mija 0,25, połowę poziomu 0,5, na którym się ustala.

Mała populacja ma niewiele zwierząt, które mogłyby się rozmnażać; prawie pełna nie ma miejsca, żeby się rozmnażać. Sprawdzenie każdego poziomu od 0 do 1 co jedną stutysięczną potwierdza, że przyrost w jednym pokoleniu jest największy dokładnie w połowie poziomu ustalonego. Na wykresie kolumna zatłoczenia układa się w krzywą w kształcie litery S, krzywą wzrostu, który natrafia na granicę.



Rysunek 3.2: Podwajanie oraz podwajanie, gdy brakuje miejsca, oba od jednej tysięcznej miejsca. Zgadniają się, dopóki miejsce jest prawie puste; potem jedno ucieka poza wykres, a drugie wygina się w S.

Gdzie to spotkasz

Nauka o rybołówstwie używa krzywych wzrostu tego kształtu od połowy XX wieku. Przełowiona populacja ryb odbudowuje się najszybciej, gdy ma mniej więcej połowę liczebności sprzed połowów, więc największy połów, jaki można brać rok po roku bez uszczuplania populacji, *maksymalny podtrzymywalny połów*, to ten, który utrzymuje ją blisko tej połowy. Ta idea jest zapisana w Konwencji Narodów Zjednoczonych

o prawie morza z 1982 roku. To wciąż tylko model, a prawdziwe populacje ryb upadały przy limitach, które na papierze wyglądały bezpiecznie.

3.4 Dwa pchnięcia, które się nie sumują

Każda reguła z rozdziału 2 była prosta: narysuj następną wartość w zależności od obecnej, a dostaniesz linię prostą. Reguły prostoliniowe mają własność tak naturalną, że pewnie przypisujesz ją każdej regule: skutki pchnięć się sumują. Weź regułę podwajania i populację 0,2. Dorzuć dodatkowe 0,1, a następne pokolenie wzrośnie o 0,2; dorzuć 0,2, a wzrośnie o 0,4. Dwa pchnięcia naraz robią to, co każde z osobna, zsumowane. Nazywa się to *superpozycją* i to ona sprawia, że reguły prostoliniowe są łatwe: problem można rozłożyć na części, rozwiązać każdą osobno i dodać odpowiedzi.

Teraz reguła zatłoczenia, z $r = 2$ i tą samą populacją 0,2. Bez pchnięcia następne pokolenie to $2 \times 0,2 \times 0,8 = 0,32$. Dorzuć 0,1, a będzie to $2 \times 0,3 \times 0,7 = 0,42$: pchnięcie podniosło je o 0,1.

Zatrzymaj się i pomyśl 3.4

Teraz dorzuć 0,2, dwa takie pchnięcia naraz. O ile wzrośnie następne pokolenie? Najpierw zgadnij, potem policz $2 \times 0,4 \times 0,6$.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. O 0,16, nie o 0,2: następne pokolenie to 0,48. Drugie pchnięcie było warte tylko 0,06, bo trafiło na bardziej zatłoczone miejsce niż pierwsze.

Wyżej robi się jeszcze dziwniej. Przy populacji 0,6 następne pokolenie to $2 \times 0,6 \times 0,4 = 0,48$; dorzuć 0,1, a będzie to $2 \times 0,7 \times 0,3 = 0,42$. Dodanie zwierząt sprawia teraz, że następne pokolenie jest *mniej*. To samo pchnięcie pomaga na dole i szkodzi na górze: skutek zmiany zależy od tego, gdzie już jesteś. To właśnie znaczy, że reguła jest *nieliniowa*, i to właśnie sprzężenie zwrotne robi z regułą.

Pułapka

„Skutek dwóch pchnięć to suma ich skutków.” Tylko dla reguły prostoliniowej. Na krzywej skutek pchnięcia zależy od tego, gdzie układ już się znajduje, i od wszystkiego, co jeszcze go pcha, więc dwie przyczyny, z których każda robi niewiele, mogą razem zrobić znacznie więcej, znacznie mniej albo coś przeciwnego. Każda reguła ze sprzężeniem zwrotnym jest krzywa.

3.5 Gdzie populacja się ustala, i nachylenie

Populacja w zatłoczeniu ustaliła się na 0,5. Dlaczego właśnie tam? Populacja stoi w miejscu, gdy następne pokolenie jest tej samej wielkości co obecne. Jednym takim

poziomem jest puste miejsce: z niczego nic się nie urodzi. Drugim jest poziom, na którym każde zwierzę zostawia średnio dokładnie jedno młode na swoje miejsce. Gdy wolna jest część $1 - x$ miejsca, każde zwierzę zostawia $r(1 - x)$ młodych, a to jest jeden, gdy $1 - x$ wynosi $1/r$, czyli przy

$$x = 1 - \frac{1}{r}.$$

Dla $r = 2$ to $1 - \frac{1}{2} = \frac{1}{2}$, tam, gdzie ustaliła się kolumna. Poziom, którego reguła nie zmienia, to *punkt stały*, jak w rozdziale 2.

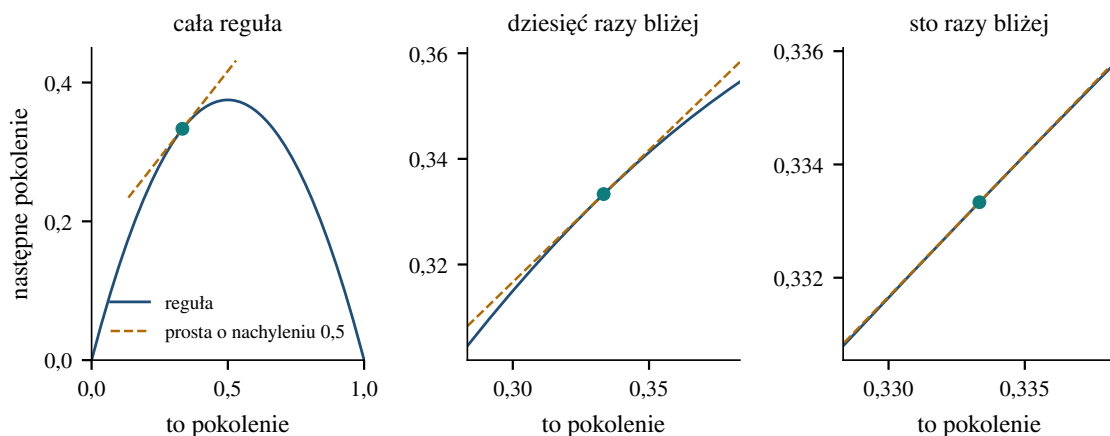
Zatrzymaj się i pomyśl 3.5

Gdzie ustali się populacja przy $r = 1,5$?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Na jednej trzeciej: $1/1,5$ to dwie trzecie, a $1 - \frac{2}{3} = \frac{1}{3}$.

Punkt stały to tylko połowa odpowiedzi; druga połowa to pytanie, czy populacja w jego pobliżu jest do niego przyciągana, czy od niego odpychana. Rozdział 2 odpowiedział na nie dla prostej: szturchnięcie od punktu stałego jest na każdym kroku mnożone przez *nachylenie* reguły, a punkt stały trzyma, gdy nachylenie co do wielkości jest mniejsze od jedności. Stwierdził też, że nachylenie krzywej zmienia się od miejsca do miejsca. To nie psuje sprawdzianu, bo dość małe szturchnięcie czuje tylko ten kawałek krzywej, na którym jest: przybliż ją, a w pobliżu dowolnego punktu wygląda jak prosta.



Rysunek 3.3: Reguła zatłoczenia przy $r = 1,5$, przybliżona wokół punktu stałego w jednej trzeciej. Z bliska krzywej nie da się odróżnić od prostej, a stromość tej prostej rozstrzyga, co stanie się z małym szturchnięciem.

Zmierz więc nachylenie w każdym punkcie stałym, szturchając, funkcją `slope` z `chaoslab`, tak jak w rozdziale 2:

code/ch03/settle.py

```

16 def slope_at(x: float, r: float) -> float:
17     """The slope of the crowding rule at x, measured by nudging."""
18     return slope(lambda y: next_gen(y, r), x)
19
20
21 def main() -> None:
22     for r in (1.5, 2.0):
23         for x in (0.0, 1 - 1 / r):
24             print(f"r = {r}: at x = {x:.4f} the slope is"
25                   f" {slope_at(x, r):6.3f}")

```

Listing 3.4: Nachylenie reguły zatłoczenia w obu punktach stałych, dla dwóch wartości tempa wzrostu

```

r = 1.5: at x = 0.0000 the slope is  1.500
r = 1.5: at x = 0.3333 the slope is  0.500
r = 2.0: at x = 0.0000 the slope is  2.000
r = 2.0: at x = 0.5000 the slope is  0.000

```

W punkcie stałym równym jednej trzeciej nachylenie wynosi 0,500: małe szturchnięcie co pokolenie maleje o połowę, więc populacja jest przyciągana z powrotem, a punkt stały jest *stabilny*. W pustym miejscu nachylenie wynosi 1,500: kilka zwierząt w pustym miejscu się mnoży, czyli rośnie, i ten punkt stały jest *niestabilny*. Przy $r = 2$ nachylenie w jednej drugiej wynosi 0,000. Ten punkt stały leży na płaskim szczycie garbu, gdzie szturchnięcie znika prawie całkowicie w jednym pokoleniu, i dlatego kolumna zatłoczenia tak szybko wskoczyła na 0,5.

Na tej idei opiera się reszta książki: **w punkcie stałym krzywej nachylenie robi to, co dla prostej robił mnożnik**, i rozstrzyga, czy małe szturchnięcie wygaśnie, czy urośnie.

Czego tu nie dowodzimy

Przyjęte tu na wiarę: że nachylenie rozstrzyga dla krzywej to, co mnożnik rozstrzygał dla prostej. Jest tak dla szturchnięć tak małych, że krzywa wygląda na prostą, co przybliżenie pokazuje, ale czego nie dowodzi; dowód jest w każdym pierwszym kursie układów dynamicznych. Pomiar szacuje to, co rachunek różniczkowy nazywa *pochodną*, i tutaj jest dokładny co do każdej wydrukowanej cyfry.

chaoslab

`chaoslab` ma regułę zatłoczenia jako `logistic`: podaj jej r , a zwróci regułę gotową dla `slope` i `orbit`. Ta nazwa pochodzi z rozdziału 5, w którym zaczynają się dziwniejsze zwyczaje tej reguły.

Laboratorium 3.1 – Odczytaj los punktu stałego z nachylenia

Napisz `fixed_points`, które podaje oba punkty stałe reguły zatłoczenia; `level_slope`, które mierzy nachylenie w poziomie ustalonym; oraz `verdict`, które z tego nachylenia odczytuje, czy populacja szturchnięta z poziomu podpełza z powrotem, wraca wahając się, czy zostaje odepchnięta. Testy uruchamiają regułę na dwa tysiące pokoleń i sprawdzają twój werdykt z tym, co się dzieje.

Plik startowy: `code/labs/ch03/l03_01_verdict.py`

Test: `code/labs/ch03/test_l03_01_verdict.py`

Uruchom: `cd code && uv run pytest -k l03_01_verdict`

3.6 Przestrzelenie

Podkreć tempo wzrostu do $r = 2,8$. Poziom ustalony to $1 - 1/2,8$, około 0,643.

Zatrzymaj się i pomyśl 3.6

Zacznij od małej populacji, jak wcześniej. Czy podpełźnie do 0,643 i się zatrzyma, tak jak populacje przy $r = 1,5$ i $r = 2$?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Wznosi się ponad ten poziom, do 0,690 w pokoleniu 5, spada poniżej, do 0,599, i waha się w tę i z powrotem, a każde wahnięcie jest mniejsze od poprzedniego, zanim się ustali.

Pułapka

„Granica wzrostu oznacza, że wzrost łagodnie się zatrzymuje.” Może tak być i przy łagodnym tempie wzrostu tak jest. Przy silnym populacja przestrzeliwuje poziom, który otoczenie jest w stanie utrzymać, zapada się poniżej niego i oscyluje. Granica wciąż istnieje, ale dochodzenie do niej to seria wahań, a po dalszym podkręceniu wahania wcale nie muszą wygasać.

Nachylenie mówi dlaczego. W poziomie ustalonym przy $r = 2,8$ wynosi ono $-0,800$. Co do wielkości jest mniejsze od jedności, więc szturchnięcie maleje i poziom jest stabilny. Ale jest ujemne, a rozdział 2 pokazał na prostej, co robi ujemny mnożnik: populacja trochę powyżej poziomu zostaje posłana trochę poniżej niego, potem znów trochę powyżej. Ustalanie się przez coraz mniejsze wahania nazywa się *oscylacją tłumioną*.

Przestrzelenie bierze się z opóźnienia. Reguła wylicza zmianę na całe pokolenie z tego, jak zatłoczone jest miejsce teraz, i patrzy znowu dopiero pokolenie później. Przy 0,44, poniżej swojego poziomu, populacja widzi mnóstwo miejsca i wydaje na świat liczne następne pokolenie; zanim te młode się pojawiają, miejsca już nie ma, a populacja jest ponad poziomem. Hotelowy prysznic z rozdziału 2 przestrzeliwał z tego samego powodu: woda, którą czujesz, wypłynęła z kurka kilka sekund temu, więc przekręcasz

kurek dalej, niż trzeba – a potem cię parzy.

Żeby sprawdzić to wyjaśnienie, usuń opóźnienie. Zachowaj to samo prawo zmiany, ale niech populacja sprawdza swoje zatłoczenie dziesięć razy na pokolenie: podziel zmianę w każdym pokoleniu na dziesięć rat, każdą wyliczaną z populacji takiej, jaka jest po poprzedniej racie.

code/ch03/overshoot.py

```
14 def in_instalments(x: float, r: float, k: int = 10) -> float:
15     """One generation of the crowding rule, paid in k instalments, each
16     worked out from the population as it stands after the one before."""
17     for _ in range(k):
18         x = x + (next_gen(x, r) - x) / k
19     return x
```

Listing 3.5: To samo prawo zmiany, spłacane w dziesięciu ratach

code/ch03/overshoot.py

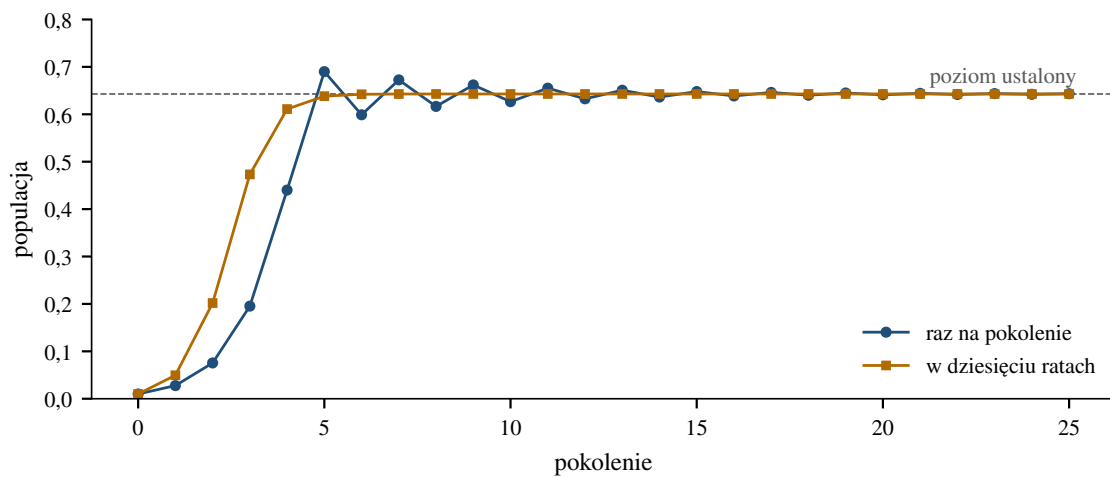
```
27 once = paid = 0.01
28 print(f"r = {r}, settled level {1 - 1 / r:.4f}")
29 print("gen    once    in ten")
30 for gen in range(generations.stop):
31     if gen in generations:
32         print(f"{gen:3d} {once:.4f} {paid:.4f}")
33     once = next_gen(once, r)
34     paid = in_instalments(paid, r)
```

Listing 3.6: Raz na pokolenie wobec dziesięciu razy na pokolenie

```
r = 2.8, settled level 0.6429
gen    once    in ten
0  0.0100  0.0100
1  0.0277  0.0496
2  0.0755  0.2018
3  0.1954  0.4732
4  0.4401  0.6110
5  0.6900  0.6382
6  0.5990  0.6422
7  0.6726  0.6428
8  0.6166  0.6428
9  0.6619  0.6429
10 0.6266  0.6429
11 0.6551  0.6429
12 0.6326  0.6429
13 0.6508  0.6429
14 0.6364  0.6429
```

Obie zmierzają do tego samego poziomu, bo prawo zmiany jest to samo i znika w tym samym miejscu, ale przestrzeliwuje tylko populacja liczona raz na pokolenie. Ta spłacana w ratach na początku rośnie nawet szybciej, bo jej młode zaczynają się rozmnażać jeszcze w tym samym pokoleniu, a mimo to wsuwa się w swój poziom, nie mijając go. Przestrzelenie powoduje nie szybkość wzrostu, lecz to, jak późno działają hamulce.

Własne prawo Verhulsta to to, czym stają się raty, gdy robi się je coraz mniejsze, i ono nigdy nie przestrzeliwuje.



Rysunek 3.4: To samo prawo wzrostu przy $r = 2,8$. Liczone raz na pokolenie przestrzeliwuje swój poziom i się waha; liczone dziesięć razy na pokolenie wsuwa się w ten sam poziom.

Laboratorium 3.2 – Późne hamulce i wczesne hamulce

Napisz regułę zatłoczenia, tę samą regułę spłacaną w k ratach, funkcję, która uruchamia każdą z nich na wiele pokoleń, oraz taką, która mówi, czy populacja kiedykolwiek mija swój poziom ustalony. Potem wypróbuj własne wartości tempa wzrostu.

Plik startowy: `code/labs/ch03/l03_02_instalments.py`

Test: `code/labs/ch03/test_l03_02_instalments.py`

Uruchom: `cd code && uv run pytest -k l03_02_instalments`

3.7 Kiedy populacja przestaje się ustalać

Podkręć tempo wzrostu jeszcze raz, do $r = 3,1$. Poziom ustalony wciąż istnieje: $1 - 1/3,1$ to około 0,677, a populacja umieszczona dokładnie na nim by na nim została. Ale nachylenie w tym miejscu, zmierzone jak wcześniej, wynosi $-1,100$.

Zatrzymaj się i pomyśl 3.7

Co nachylenie $-1,100$ robi z małym szturchnięciem od poziomu ustalonego, pokolenie po pokoleniu?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. W każdym pokoleniu przerzuca szturchnięcie na drugą stronę i powiększa je o jedną dziesiątą. Szturchnięcie rośnie, więc poziom ustalony nie jest już w stanie utrzymać populacji.

`code/ch03/two_cycle.py`

```

15  r = 3.1
16  level = 1 - 1 / r
17  x = level + 0.001
18  for gen in range(6):
19      print(f"gen {gen}: {x - level:+.5f} from the settled level")
20      x = next_gen(x, r)

```

Listing 3.7: Szturchnięcie o jedną tysięczną, pokolenie po pokoleniu

`code/ch03/two_cycle.py`

```

24  compare(3.1, range(60, 66))

```

Listing 3.8: Gdzie populacja ląduje

```

gen 0: +0.00100 from the settled level
gen 1: -0.00110 from the settled level
gen 2: +0.00121 from the settled level
gen 3: -0.00134 from the settled level

```

```
gen 4: +0.00146 from the settled level
gen 5: -0.00162 from the settled level
```

```
r = 3.1, settled level 0.6774
```

```
gen  once  in ten
60  0.5580  0.6774
61  0.7646  0.6774
62  0.5580  0.6774
63  0.7646  0.6774
64  0.5580  0.6774
65  0.7646  0.6774
```

Szturchnięcie rośnie, co pokolenie zmieniając stronę, a populacja ląduje daleko od poziomu ustalonego. Waha się między dwiema wartościami, 0,558 i 0,765, w jednym pokoleniu nisko, w następnym wysoko, bez końca. Nie pojawiło się nic losowego; reguła to wciąż ta sama jedna linijka. Ustalanie się ustało, a jego miejsce zajął cykl, który powtarza się co dwa pokolenia. Populacja spłacana w ratach wciąż ustala się na 0,677: usuń opóźnienie, a wahania znikną razem z nim.

Gdzieś między $r = 2,8$ a $r = 3,1$ poziom ustalony stracił swoją moc przyciągania. Gdzie dokładnie i co robi reguła, gdy tempo wzrostu podkreca się jeszcze dalej, to temat rozdziału 5. Reguła ma jedną linijkę, dziecko policzyłoby ją ołówkiem, a jeszcze nie skończyła cię zaskakiwać.

Podsumowanie

- **Wzrost wykładniczy**, poznany w rozdziale 2, przy każdym podwojeniu dodaje więcej niż wszystkie wcześniejsze podwojenia, i nic rzeczywistego nie rośnie tak długo.
- **Sprzężenie zwrotne** to wielkość, która zmienia własne tempo zmian. **Dodatnie sprzężenie zwrotne** pcha dalej (wycie głośnika, run na bank), aż natrafi na granicę; **ujemne sprzężenie zwrotne** stawia opór (termostat, zatłoczenie).
- Zatłoczenie wpisane w podwajanie daje $x \mapsto rx(1-x)$, gdzie x to zajęta część miejsca: **krzywą S**, rosnącą najszybciej w połowie poziomu ustalonego.
- Dla reguły prostoliniowej skutki pchnięć się sumują (**superpozycja**); dla krzywej, **nieliniowej** reguły nie, a pchnięcie może pomóc albo zaszkodzić, zależnie od tego, gdzie jest układ.
- Reguła ustala się tam, gdzie każde zwierzę jest dokładnie zastępowane, w **punkcie stałym** $1 - 1/r$. **Nachylenie** w nim, zmierzone szturchaniem jak w rozdziale 2, mnoży małe szturchnięcia; wielkość mniejsza od jedności oznacza stabilność.
- Ujemne nachylenie oznacza **przestrzelenie** i **oscylację tłumioną**. Przyczyną jest opóźnienie: to samo prawo w małych ratach nie przestrzeliwuje.
- Po dalszym podkreśleniu wielkość nachylenia przekracza jeden, a populacja bez końca waha się między dwiema wartościami – to wstęp do rozdziału 5.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

odróżnić dodatnie sprzężenie zwrotne od ujemnego w układzie opisanym słowami	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zamienić zdanie „wzrost zwalnia, gdy brakuje miejsca” w regułę i ją uruchomić	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
sprawdzić, czy skutki dwóch pchnięć danej reguły się sumują	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
znaleźć punkty stałe reguły krzywoliniowej i ocenić każdy z nich po zmierzonym nachyleniu	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
przewidzieć na podstawie tego nachylenia, co zrobi populacja w pobliżu swojego poziomu ustalonego	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Ile ziarenek leży na jedenastym polu szachownicy, a ile na pierwszych dziesięciu polach razem?
2. Dodatnie czy ujemne sprzężenie zwrotne: termostaty; oklaski, które stają się głośniejsze, gdy dołączają kolejne osoby; cena, która rośnie, gdy towaru brakuje, więc mniej ludzi go kupuje?
3. Co w regule zatłoczenia przy $r = 2$, zaczynając od 0,5, robią z następnym pokoleniem pchnięcia o 0,1 i o 0,2?
4. Gdzie ustala się populacja przy $r = 2,5$? A przy $r = 1,25$?
5. Nachylenie w punkcie stałym wynosi $-0,5$, a populacja startuje 0,01 nad nim. Gdzie jest po jednym pokoleniu, a gdzie po dwóch?
6. Dlaczego populacja, która sprawdza swoje zatłoczenie dziesięć razy na pokolenie, nie przestrzeliwuje, skoro ta, która sprawdza je raz na pokolenie, przestrzeliwuje?

Ruch: reguły zmiany i mapa wszystkich stanów

Herbata z rozdziału 1 żyła w pełnych minutach: jej reguła przeskakiwała od jednej minuty do następnej. Wahadło nie przeskakuje; w każdej chwili gdzieś jest i dokądś zmierza. Prawa ruchu Newtona, opublikowane w 1687 roku, nie mówią, gdzie będzie planeta w przyszły wtorek. Mówią, jak szybko zmienia się jej prędkość w tej chwili. Jak wydobyć przyszłość z takiej reguły?

Ten rozdział odpowiada sztuczce z rozdziału 2: rób małe kroki. Potem rysuje obraz, w którym narysowana jest reszta książki – mapę, na której każdy możliwy stan jest punktem, a każda przyszłość drogą – i kończy się twierdzeniem, które mówi, dlaczego model pogody z rozdziału 9 potrzebuje trzech liczb.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- czytać regułę zmiany jako informację o tym, jak szybko zmienia się w tej chwili każda liczba stanu
- zamienić regułę zmiany na drogę w kodzie, metodą Eulera i metodą RK4
- użyć energii, żeby sprawdzić, czy symulacja mówi prawdę
- czytać portret fazowy: jego punkty stałe, jego pętle i kierunek ruchu
- powiedzieć, co ruch na płaszczyźnie może, a czego nie może, i ile liczb potrzebuje chaos w czasie ciągłym

4.1 Reguła, która mówi, jak szybko

Prędkościomierz w samochodzie to reguła zmiany. Nie mówi, gdzie jesteś; mówi, jak szybko zmienia się to, gdzie jesteś.

Zatrzymaj się i pomyśl 4.1

Prędkościomierz pokazuje 60 kilometrów na godzinę. Mniej więcej gdzie będzie samochód za minutę? A za godzinę?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Za minutę mniej więcej kilometr dalej: 60 kilometrów na godzinę to kilometr na minutę. Za godzinę nie da się tego powiedzieć, bo prędkość wiele razy się zmieni.

W krótkim czasie to, jak szybko coś się zmienia, mówi, jak daleko zajdzie; w długim mówi niewiele. Więc przejeżdżasz kawałek, znowu patrzysz na prędkościomierz i jedziesz dalej.

Oto pierwszy przykład. Ciężarek wisi na sprężynie; odciągasz go o jeden metr od położenia spoczynku i puszczasz. Jego stan to dwie liczby: *położenie* x , czyli jak daleko jest od spoczynku, i *prędkość* v , czyli jak szybko zmienia się x . Reguła mówi, jak szybko zmienia się każda z nich:

- położenie zmienia się w tempie v , bo tym właśnie jest prędkość;
- prędkość zmienia się w tempie $-k x$: sprężyna ciągnie z powrotem ku środkowi, tym mocniej, im bardziej jest rozciągnięta. Liczba k mówi, jak sztywna jest sprężyna; przyjmij $k = 1$. Znak minus mówi, że siła ciągnie wstecz.

Inne książki zapisują tę samą regułę jako

$$\frac{dx}{dt} = v, \quad \frac{dv}{dt} = -k x,$$

gdzie $\frac{dx}{dt}$ czyta się „jak szybko zmienia się x ”. Reguła tego rodzaju to *równanie różniczkowe*. Ta książka nigdy nie rozwiązuje go ręcznie. Śledzi je małymi krokami, tak jak robi to komputer prognozujący pogodę.

4.2 Małe kroki

Najprostszy sposób śledzenia reguły zmiany nosi imię Leonharda Eulera, który opisał go w XVIII wieku. Wybierz krótki odcinek czasu, *krok*, i przesun każdą liczbę o jeden krok w tempie, jakie ma teraz:

$$\text{nowe } x = x + \text{krok} \times v, \quad \text{nowe } v = v + \text{krok} \times (-k x).$$

Potem zapytaj regułę znowu w nowym stanie i powtarzaj. Reguła zmiany stała się regułą następnego kroku i dalej działa iteracja.

code/ch04/spring.py

```

11 K = 1.0    # how stiff the spring is, for each kilogram of mass
12
13
14 def rates(x: float, v: float) -> tuple[float, float]:
15     """How fast position and velocity are changing right now."""
16     return v, -K * x    # moves at v; the spring pulls back towards 0
17
18
19 def euler_step(x: float, v: float, dt: float) -> tuple[float, float]:
20     """Pretend the rates stay the same for dt, and move."""
21     dx, dv = rates(x, v)

```

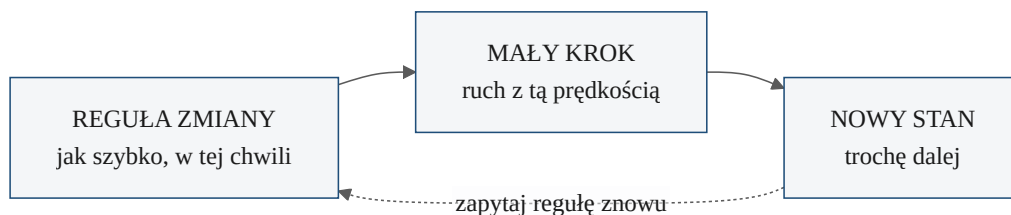
```

22     return x + dt * dx, v + dt * dv
23
24
25 def energy(x: float, v: float) -> float:
26     """Energy of motion plus energy stored in the stretched spring."""
27     return 0.5 * v * v + 0.5 * K * x * x
28
29
30 def main() -> None:
31     x, v, dt = 1.0, 0.0, 0.1      # pulled out 1 metre, let go
32     for step in range(101):      # 100 steps of 0.1 s: ten seconds
33         if step % 20 == 0:
34             print(f"t = {step * dt:4.1f} s   x = {x:6.3f}   "
35                   f"v = {v:6.3f}   energy = {energy(x, v):.4f}")
36             x, v = euler_step(x, v, dt)

```

Listing 4.1: Sprężyna śledzona metodą Eulera

rates to reguła, a euler_step to metoda Eulera, dwa wiersze. energy dodaje energię ruchu, połowę $v \times v$, do energii zmagazynowanej w sprężynie, połowy $k \times x \times x$. Ciężarek startuje z $x = 1$, w spoczynku, a kroki trwają 0,1 sekundy.



Rysunek 4.1: Metoda Eulera. Reguła podaje templa w tej chwili; mały krok przesuwa tak, jakby się nie zmieniały; w nowym stanie znowu pyta się regułę.

Zatrzymaj się i pomyśl 4.2

Prawdziwa sprężyna bez tarcia drga w nieskończoność, za każdym razem dochodząc tak samo daleko: nic nie odbiera jej energii i nic jej nie dodaje. Co powinna pokazywać kolumna energii, wiersz po wierszu?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Za każdym razem tę samą liczbę: 0,5, połowę 1×1 .

t = 0.0 s	x = 1.000	v = 0.000	energy = 0.5000
t = 2.0 s	x = -0.453	v = -1.007	energy = 0.6101
t = 4.0 s	x = -0.810	v = 0.913	energy = 0.7444
t = 6.0 s	x = 1.286	v = 0.402	energy = 0.9083
t = 8.0 s	x = -0.178	v = -1.478	energy = 1.1084
t = 10.0 s	x = -1.409	v = 0.849	energy = 1.3524

Nie pokazuje. Wychylenia rosną, a energia wspina się od 0,5 do 1,35 w dziesięć sekund. Nic w regule nie dodało tej energii. Zrobiła to symulacja.

Powód widać na obrazku. Narysuj stan jako punkt: położenie w poziomie, prędkość w pionie. Przy $k = 1$ energia to połowa kwadratu odległości punktu od środka, więc prawdziwa sprężyna krąży po okręgu. Krok Eulera idzie wzdłuż prostej, którą w tej chwili wskazuje reguła, a prosta styczna do okręgu wychodzi z niego na zewnątrz. Każdy krok ląduje trochę na zewnątrz, i to dokładnie, z twierdzenia Pitagorasa: każdy krok mnoży energię przez $1 + \text{krok} \times \text{krok}$, czyli 1,01 przy kroku 0,1. Jeden procent na krok, 100 razy w dziesięć sekund.

Zatrzymaj się i pomyśl 4.3

Weź kroki dziesięć razy mniejsze, 0,01 sekundy. Czy symulacja jest teraz poprawna?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Poprawna dłużej, a nie poprawna. W dziesięć sekund energia rośnie o 10,5 procent zamiast o 170; po stu sekundach urosła o 172 procent, mniej więcej tyle, ile większy krok zyskał w dziesięć.

Każdy mały krok zyskuje sto razy mniej, a w każdej sekundzie jest ich dziesięć razy więcej. Dziesięć razy mniejsze kroki kupiły dziesięć razy dłuższy czas, za dziesięć razy więcej pracy.

Pułapka

„Mniejszy krok sprawia, że symulacja jest poprawna.” Sprawia, że jest poprawna dłużej. Błąd metody krokowej powstaje na nowo przy każdym kroku i się piętrzy, więc mniejszy krok kupuje czas, zanim ta sterta urośnie za bardzo. Uczciwe pytanie do symulacji nie brzmi, jak mały miała krok, tylko jak długo ją sprawdzano. Część II spotyka surowszy horyzont, którego nie przesunie żaden rozmiar kroku.

4.3 Spójrz naprzód i uśrednij

Euler pyta regułę raz, na początku kroku, i wierzy odpowiedzi przez cały krok. Tempa zmieniają się w trakcie kroku, a Euler nigdy się o tym nie dowiaduje. Metoda, której używa ta książka, opracowana przez Carla Rungego i Martina Kuttę na przełomie XIX i XX wieku i znana jako RK4, pyta cztery razy:

1. o tempa na początku;
2. o tempa pół kroku dalej, idąc w pierwszych tempach;
3. o tempa pół kroku dalej, idąc w drugich tempach;
4. o tempa cały krok dalej, idąc w trzecich.

Potem robi prawdziwy krok ze średnią tych czterech, licząc dwie środkowe dwukrotnie.

code/src/chaoslab/flows.py

```

34 def rk4_step(field: Field, state: State, dt: float) -> State:
35     """One Runge-Kutta step: sample the velocity four times, average them.
36
37     k1 is the velocity at the start; k2 and k3 are velocities half a step
38     ahead; k4 is a full step ahead. The weights 1, 2, 2, 1 are what make the
39     error in the position shrink sixteen-fold when dt is halved, where
40     Euler's only halves. (The slow drift in a spring's energy shrinks faster
41     still, about thirty-two-fold; Chapter 4 measures it.)
42     """
43     k1 = field(state)
44     k2 = field(_add(state, k1, dt / 2))
45     k3 = field(_add(state, k2, dt / 2))
46     k4 = field(_add(state, k3, dt))
47     return tuple(
48         s + dt * (a + 2 * b + 2 * c + d) / 6
49         for s, a, b, c, d in zip(state, k1, k2, k3, k4, strict=True)
50     )

```

Listing 4.2: Przepis RK4 z biblioteki książki

field to reguła zmiany, k1 do k4 to cztery próbki, a _add wylicza, gdzie byłbyś po przejściu kawałka w danych tempach. Wagi 1, 2, 2, 1 są dobrane tak, żeby błędy próbek znosiły się nawzajem, jak tylko się da; algebrę tu pomijamy. Ta sama funkcja, obok `euler_step` i `integrate`, poniesie model pogody z rozdziału 9.

Laboratorium 4.1 – Jeden krok RK4

Napisz `rk4_step` według przepisu. Test porównuje go z biblioteką na sprężynie i na wahadle, a także na regule „ x rośnie w tempie równym x ”: jeden krok długości 1 z 1 musi wylądować na $1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24}$, blisko prawdziwej odpowiedzi $e \approx 2,718$.

Plik startowy: code/labs/ch04/l04_01_rk4.py

Test: code/labs/ch04/test_l04_01_rk4.py

Uruchom: `cd code && uv run pytest -k l04_01_rk4`

Zatrzymaj się i pomyśl 4.4

RK4 pyta regułę cztery razy na krok, więc za tę samą pracę Euler może robić kroki cztery razy mniejsze. Co wolisz: Eulera z krokami 0,025 sekundy czy RK4 z krokami 0,1?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. RK4, z ogromną przewagą. W dziesięć sekund Euler zyskuje 28 procent energii; RK4 traci 1,4 części na milion. Za tę samą pracę jego błąd jest około 200 000 razy mniejszy.

code/ch04/energy_drift.py

```

12 FIELD = spring(1.0)          # the rule of change from the chapter
13 START = (1.0, 0.0)           # pulled out 1 metre, at rest
14
15
16 def energy(state: tuple[float, ...]) -> float:
17     x, v = state
18     return 0.5 * v * v + 0.5 * x * x
19
20
21 def drift stepper, dt: float, seconds: float -> float:
22     """Fractional change in energy after following the rule for seconds."""
23     state = START
24     for _ in range(round(seconds / dt)):
25         state = stepper(FIELD, state, dt)
26     return energy(state) / energy(START) - 1.0

```

Listing 4.3: Pomiar dryfu energii symulowanej sprężyny

method	step	seconds	rule calls	energy change
Euler	0.100	10	100	+170 %
Euler	0.010	10	1000	+10.5 %
Euler	0.010	100	10000	+172 %
Euler	0.025	10	400	+28.4 %
RK4	0.100	10	400	-0.000139 %
RK4	0.100	1000	40000	-0.0139 %

rule calls liczy pracę. Ostatni wiersz mówi, że RK4 też nie jest dokładne: w tysiąc sekund traci 139 części na milion. Ono również jest poprawne dłużej, a nie poprawne, tylko o wiele dłużej.

Laboratorium 4.2 – Zmierz dryf

Napisz energię sprężyny i funkcję, która śledzi sprężynę daną metodą przez dany czas i podaje, o ile zmieniła się jej energia. Potem odpowiedz na pytanie, które rozdział zostawia otwarte: zmniejsz krok o połowę – ile razy mniejszy stanie się dryf energii w każdej z metod? (Komentarz w listingu RK4 dotyczy błędu położenia, a to inna miara.) Przewiduj, zanim uruchomisz.

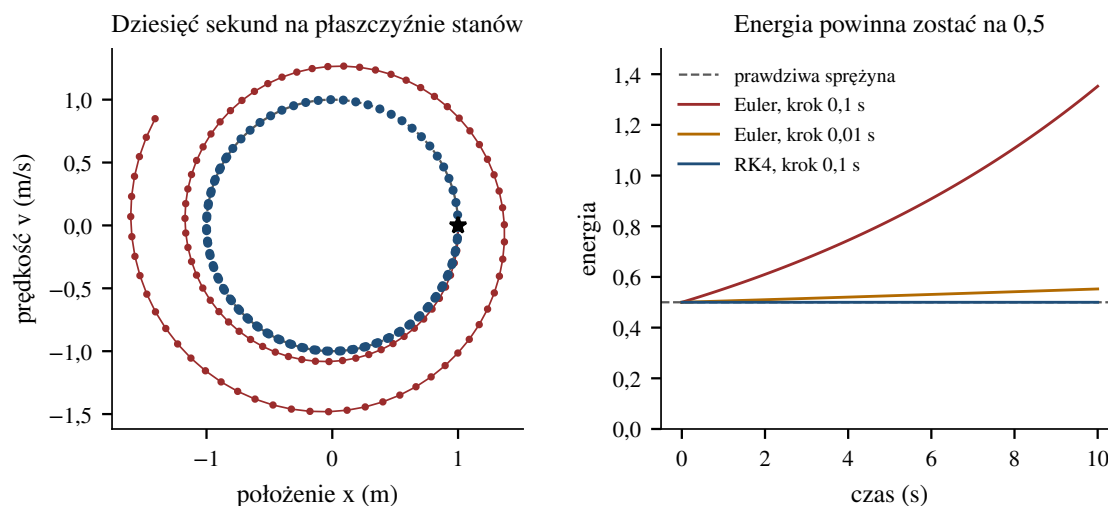
Plik startowy: code/labs/ch04/l04_02_drift.py

Test: code/labs/ch04/test_l04_02_drift.py

Uruchom: `cd code && uv run pytest -k l04_02_drift`

4.4 Mapa wszystkich stanów

Lewa połowa rysunku 4.2 nie pokazuje niczego w funkcji czasu. Każdy jej punkt to cały stan, położenie i prędkość naraz, a ruch to droga od punktu do punktu. Płaszczyzna wszystkich możliwych stanów to *przestrzeń fazowa*, a rysunek dróg na niej to *portret fazowy*. Każdy układ chaotyczny w tej książce jest narysowany na takim portrecie.



Rysunek 4.2: Ta sama sprężyna śledzona przez dziesięć sekund dwiema metodami z tym samym krokiem. Po lewej: na płaszczyźnie stanów Euler (czerwony) oddala się spiralą od prawdziwego okręgu, a RK4 (niebieski) zostaje na nim. Po prawej: energia, którą podaje każda z nich.

Bogatszym przykładem jest wahadło. Jego stan to kąt od pionu w dół i to, jak szybko ten kąt się zmienia. Jego reguła to reguła sprężyny z jedną różnicą: siła ciągnąca z powrotem jest największa, gdy wahadło sterczy w bok, a znika, gdy wskazuje prosto w dół albo prosto w górę. (To sinus, którego woła reguła wahadła z chaoslab.)

Zatrzymaj się i pomyśl 4.5

Wahadło puszczone dalej ma dłuższą drogę, ale też porusza się szybciej. Czy wahadło puszczone z 90 stopni wychyla się tam i z powrotem dłużej niż puszczone z 10°?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Dłużej: 2,368 sekundy wobec 2,010, o około 18 procent więcej. Puszczone ze 170 stopni potrzebuje 4,894 sekundy, 2,4 razy więcej.

code/ch04/pendulum.py

```

15 G_OVER_L = 9.81          # gravity over length: a one-metre pendulum
16 FIELD = pendulum(G_OVER_L) # (angle, angular velocity) -> their rates
17
18
19 def period(swing_degrees: float, dt: float = 0.001) -> float:
20     """Seconds for one full swing, out and back, released at rest."""
21     state = (math.radians(swing_degrees), 0.0)
22     steps = 0
23     while True:
24         new = rk4_step(FIELD, state, dt)
25         if new[1] > 0.0: # it has stopped at the far side, turned round
26             share = -state[1] / (new[1] - state[1])
27             return 2 * (steps + share) * dt # out and back is twice this

```

28

```
state, steps = new, steps + 1
```

Listing 4.4: Mierzenie czasu wahnięcia wahadła

released at	one swing out and back
5 deg	2.007 s
10 deg	2.010 s
45 deg	2.086 s
90 deg	2.368 s
170 deg	4.894 s

Szkolna reguła, że czas wahnięcia nie zależy od wychylenia, jest prawdziwa dla małych wychyleń: czasy zbliżają się do 2,006 sekundy, gdy wychylenie maleje. Dlatego precyzyjny zegar wahadłowy wychyla się ledwie o kilka stopni.

Teraz narysuj regułę jako mapę. W każdym punkcie płaszczyzny reguła podaje tempa, a więc kierunek, w którym stan zaraz się ruszy. Postaw tam małą strzałkę. Ruch to podążanie za strzałkami.

Zatrzymaj się i pomyśl 4.6

Gdzieś na mapie strzałka ma zerową długość: nic się nie zmienia. Jakie to stany, jako położenia prawdziwego wahadła? Co się stanie, gdy każdy z nich lekko trącisz?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Zwisanie prosto w dół w spoczynku i balansowanie dokładnie do góry nogami w spoczynku. Trącone, pierwsze łagodnie się wokół niego waha; drugie odpada. To dwa punkty stałe, jeden stabilny, drugi niestabilny, jak w rozdziale 2.

Blisko środka drogi są zamkniętymi pętlami: wahadło wychyla się tam i z powrotem, w nieskończoność po tej samej pętli. Droga, która zamyka się sama w sobie, to *cykl*, a cykl jest doskonale przewidywalny: jedno okrążenie i znasz wszystkie następne. Daleko u góry i u dołu drogi biegają przez całą szerokość: wahadło przechodzi przez górę i obraca się dalej.

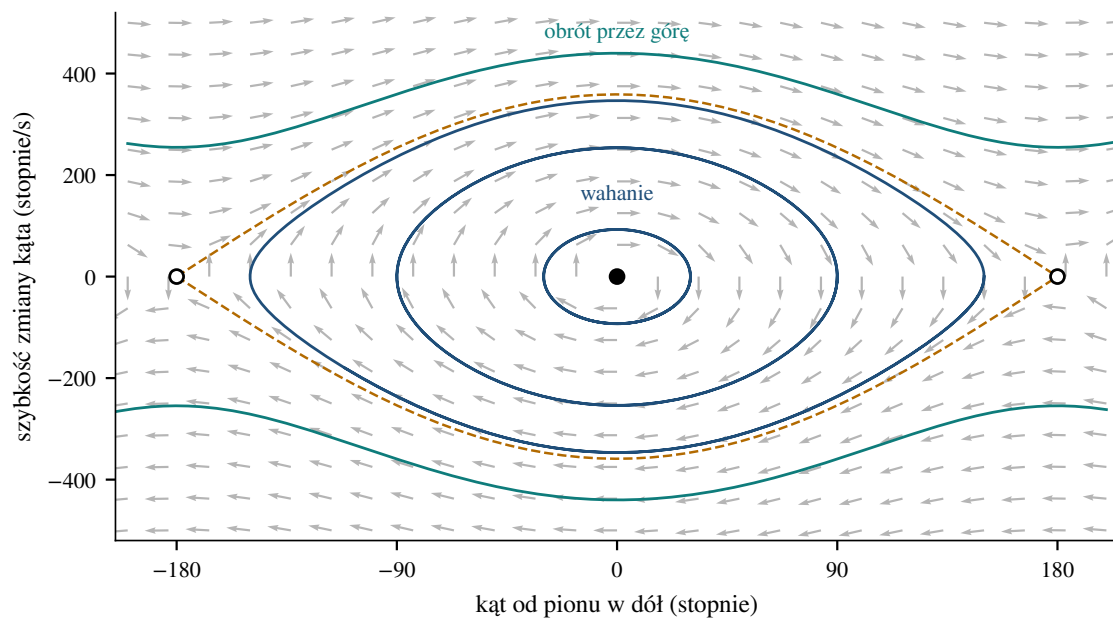
4.5 Zatrzymać się, krążyć albo uciec

Prawdziwe wahadło ma tarcie, które z każdym wahnięciem odbiera trochę energii.

Zatrzymaj się i pomyśl 4.7

Co dzieje się na mapie z drogą wahadła z tarcie puszczonego ze 150 stopni i gdzie się ona kończy?

Odpowiedz, zanim pójdziesz dalej.



Rysunek 4.3: Mapa wszystkich stanów wahadła o długości jednego metra. Szare strzałki pokazują, w którą stronę rusza się stan. Niebieskie pętle to wahania puszczane z 30, 90 i 150 stopni; zielone drogi przechodzą przez górę; linia przerywana oddziela jedno od drugich. Pełna kropka to zwisanie w spoczynku; dwa puste kółka to to samo położenie do góry nogami.

Odpowiedź. Każde wahanie jest mniejsze od poprzedniego, więc droga nie może się zamknąć. Zwija się spiralą do środka i kończy na pełnej kropce: wahadło zwisa prosto w dół, w spoczynku.

Stan, do którego wpadają wszystkie pobliskie drogi, to *atraktor*, a punkt spoczynku wahadła z tarcie to jego najprostszy rodzaj. Przyciągać może też pętla. Holenderski inżynier Balthasar van der Pol zapisał klasyczny przykład mniej więcej sto lat temu, dla obwodów z lampami radiowymi:

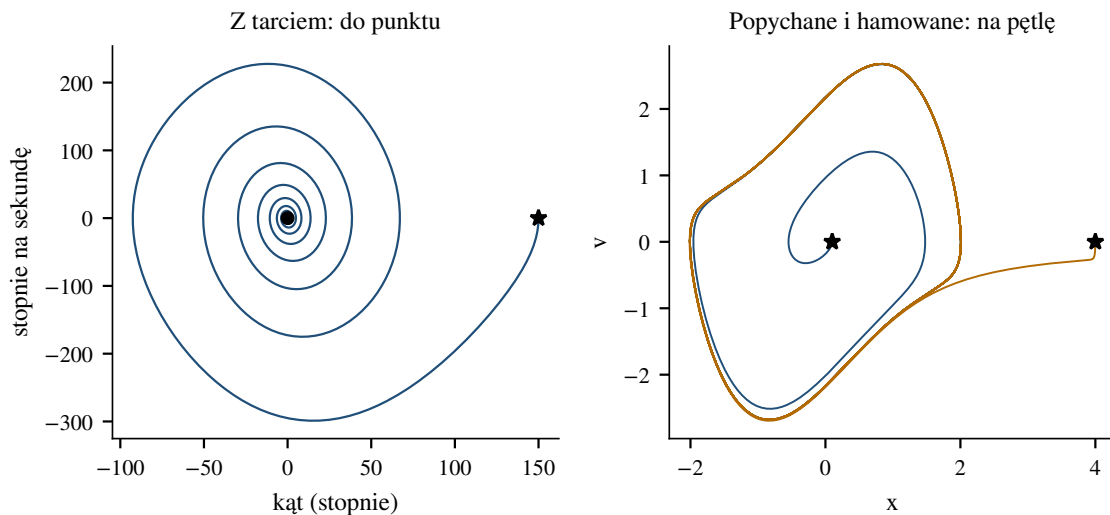
$$x \text{ zmienia się w tempie } v, \quad v \text{ zmienia się w tempie } -x + (1 - x \times x) v.$$

Dopóki x leży między -1 a 1 , nowy człon przyspiesza ruch; poza tym przedziałem hamuje. Zaczynij blisko środka, od $0,1$, a wychylenia rosną; zaczynij daleko, od 4 , a maleją. Po 60 sekundach oba wychylają się do $2,01$ i $2,01$: jedna pętla, osiągnięta od środka i od zewnątrz. To *cykl graniczny*, a zegar wahadłowy jest jednym z nich: małe pchnięcie przy każdym wahanie, tarcie odbierające energię i wychylenie, które ustala się na jednej wielkości, jakkolwiek je zacząłeś.

Zatrzymaj się i pomyśl 4.8

Dlaczego dwie różne drogi na takiej mapie nigdy się nie przecinają?

Odpowiedz, zanim pójdziesz dalej.



Rysunek 4.4: Dwa sposoby, w jakie ruch na płaszczyźnie może się ustalić. Po lewej: wahadło z tarcieniem puszczane ze 150 stopni zwija się spiralą do punktu, w którym zwisa w spoczynku. Po prawej: reguła van der Pola ze startu wewnątrz i ze startu na zewnątrz, nawijająca się na tę samą pętlę.

Odpowiedź. W punkcie przecięcia stan miałby dwie przyszłości, a reguła daje jedną strzałkę: jeden stan, jedna przyszłość.

Na płaszczyźnie ten jeden fakt ma wielką siłę. Droga, która nie może przeciąć ani siebie, ani żadnej innej drogi, i nie opuszcza ograniczonego obszaru, jest ogrodzona własną przeszłością. Może zwijać się do punktu stałego albo do pętli, albo uciec do nieskończoności. Nie może błąkać się bez końca.

Czego tu nie dowodzimy

To *twierdzenie Poincarégo–Bendixsona*, od nazwisk Henriego Poincarégo i Szweda Ivara Bendixsona. Wymaga gładkiej reguły, a jego dokładne sformułowanie dopuszcza jeszcze jedno zakończenie, łańcuch punktów stałych połączonych drogami. Dowód, na kilka stron, jest w każdym podręczniku układów dynamicznych. Dotyczy ruchu w czasie ciągłym: reguła skacząca krokami może przeskoczyć własną drogę, a rozdział 5 znajduje dzikie zachowanie w regule krokowej z jedną liczbą. I wymaga płaszczyzny: w trzech wymiarach droga może przejść nad sobą i pod sobą.

Chaos w czasie ciągłym potrzebuje więc co najmniej trzech liczb stanu. Dlatego model pogody z rozdziału 9, który Edward Lorenz okroił w 1963 roku z większego modelu konwekcji, ma trzy liczby, a nie dwie.

Zatrzymaj się i pomyśl 4.9

Zawieś drugie wahadło u dołu pierwszego. Ilu liczb potrzebuje teraz stan i czy twierdzenie nadal obiecuje prosty ruch?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Czterech: kąta i prędkości dla każdego wahadła. Twierdzenie nic nie mówi o czterech liczbach, a rozdział 13 pokazuje, że ruch wahadła podwójnego jest chaotyczny.

Pułapka

„Wahadło to prosty układ, więc jego ruch musi być prosty.” Dla jednego wahadła tak jest: dwie liczby stanu, płaszczyzna i twierdzenie, które pozwala tylko zatrzymać się, krążyć albo uciec. Zawieś na nim drugie, a stan ma cztery liczby, płot znika i ruch może stać się chaotyczny. To, jak prosto wygląda układ, mówi niewiele o tym, jak prosty jest jego ruch. Więcej mówi policzenie liczb w jego stanie.

4.6 Dwa ciała: mechanizm zegara w najlepszym wydaniu

Cztery liczby stanu umożliwiają chaos; nie sprawiają, że on następuje. Najlepszym dowodem jest układ, na którym zbudowano obraz świata jak zegara: jedna planeta krążąca wokół Słońca. Grawitacja Newtona to reguła zmiany. Położenie planety zmienia się w tempie jej prędkości, a prędkość zmienia się w stronę Słońca o wielkość, która maleje z kwadratem odległości: dwa razy dalej, cztery razy słabiej. Słońce jest tak dużo cięższe, że może stać w miejscu.

code/ch04/two_bodies.py

```

17 def gravity(s: tuple[float, ...]) -> tuple[float, ...]:
18     """Position changes at the velocity; velocity changes towards the sun,
19     by one over the distance squared."""
20     x, y, vx, vy = s
21     r = math.sqrt(x * x + y * y)
22     pull = 1.0 / (r * r * r)          # 1/r^2, and x/r, y/r point home
23     return (vx, vy, -pull * x, -pull * y)

```

Listing 4.5: Grawitacja Newtona jako reguła zmiany

Stan to cztery liczby: gdzie jest planeta, w poziomie i w pionie, i jak szybko się porusza, w poziomie i w pionie. Listing wystrzeliwuje ją w bok z 3 różnymi prędkościami, śledzi każdą orbitę dziesięć razy dookoła metodą RK4 i mierzy czas każdego powrotu na linię startu. size orbity to połowa jej największej szerokości.

launch	first lap	average of 10	size	lap^2/size^3
1.0	6.283185	6.283185	1.0000	39.4784
1.2	14.993321	14.993321	1.7857	39.4784
1.3	36.403013	36.403013	3.2258	39.4784

Średnia z dziesięciu okrążeń to pierwsze okrążenie, co do każdej wydrukowanej cyfry: planeta wraca po swojej elipsie zgodnie z rozkładem, raz za razem. Ostatnia kolumna jest taka sama dla wszystkich trzech orbit, 39,48: kwadrat czasu okrążenia podzielony przez sześćcian rozmiaru. Johannes Kepler znalazł tę regułę w położeniach planet w 1619 roku,

a Newton pokazał w 1687, że wynika ona z grawitacji; twój laptop znalazł ją ponownie z samej reguły zmiany.

Dlaczego przy czterech liczbach nie ma chaosu? Dwie wielkości nigdy się nie zmieniają: energia i tempo, w jakim odcinek od Słońca do planety zakreśla pole. Każda z nich mocniej przyszpila stan, a przy dwóch przyszpilonych ruch nie ma już miejsca, żeby być czymkolwiek poza regularnym. Dodaj trzecie ciało, a to przestaje być prawdą; od tego zaczyna się rozdział 13.

Gdzie to spotkasz

Tablice położenia planet, dużych księżyców i wielu planetoid, według których sterują nawigatorzy sond kosmicznych, publikuje Jet Propulsion Laboratory NASA. Powstają tak, jak ten listing wyznaczył swoje orbity: Newtonowska reguła zmiany dla całego Układu Słonecznego śledzona naprzód małymi krokami i sprawdzana z pomiarami radarowymi i śledzeniem sond.

To mechanizm zegara w najlepszym wydaniu: reguła zmiany, małe kroki i przewidywanie, które wraca zgodnie z rozkładem co do każdej wydrukowanej cyfry. Część II pyta, co się dzieje, gdy reguła jest równie dokładna, a przyszłość nie.

Podsumowanie

- **Reguła zmiany**, czyli **równanie różniczkowe**, mówi, jak szybko zmienia się w tej chwili każda liczba stanu, a nie jaka będzie za chwilę.
- **Metoda Eulera** przesuwa każdą liczbę o jeden krótki krok w tempie, jakie ma teraz. Na sprężynie zyskuje energię przy każdym kroku, a mniejszy krok czyni ją **poprawną dłużej**, a nie poprawną.
- **RK4** próbkuje tempa cztery razy na krok i je uśrednia; za tę samą pracę jej błąd jest dużo mniejszy, ale ona też dryfuje. Sprawdzianem jest energia.
- **Przestrzeń fazowa** to płaszczyzna wszystkich możliwych stanów; reguła to strzałka w każdym punkcie. Wahadło rysuje **cykle** wokół swojego stabilnego **punktu stałego**, a duże wahnięcia trwają dłużej niż małe.
- Tarcie zwija drogę spiralą do **atraktora**; przyciągać może też pętla. Drogi się nie przecinają, więc na płaszczyźnie ruch może się tylko zatrzymać, krążyć albo uciec (**Poincaré–Bendixson**): chaos w czasie ciągłym potrzebuje co najmniej trzech liczb.
- Dwa ciała mają cztery liczby stanu, a mimo to kreślą elipsy zgodnie z rozkładem, jak mówi trzecie prawo Keplera: trzy liczby są potrzebne do chaosu, ale nie wystarczają.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

czytać regułę zmiany jako informację o tym, jak szybko zmienia się w tej chwili każda liczba stanu	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zamienić regułę zmiany na drogę w kodzie, metodą Eulera i metodą RK4	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
użyć energii, żeby sprawdzić, czy symulacja mówi prawdę	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
czytać portret fazowy: jego punkty stałe, jego pętle i kierunek ruchu	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co ruch na płaszczyźnie może, a czego nie może, i ilu liczb potrzebuje chaos w czasie ciągłym	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Prędkościomierz pokazuje 90 kilometrów na godzinę. Mniej więcej jak daleko samochód zajedzie w ciągu następnych 20 sekund?
2. Sprężyna z $k = 1$ jest w $x = 0$ i porusza się z $v = 1$, więc jej energia wynosi 0,5. Zrób jeden krok Eulera o długości 0,1. Jakie są nowe x , v i energia?
3. Euler z krokiem 0,1 zyskuje w dziesięć sekund 170 procent energii sprężyny. Mniej więcej jak długo może trwać przebieg z krokiem 0,001, zanim zyska tyle samo?
4. Opisz cztery próbki, które RK4 pobiera w jednym kroku, i jak je łączy.
5. Co na portrecie fazowym wahadła oznacza dla prawdziwego wahadła zamknięta pętla, droga przez całą szerokość i spirala?
6. Dlaczego gładka reguła zmiany z dwiema liczbami stanu nigdy nie jest chaotyczna i ilu liczb potrzebuje chaos w czasie ciągłym?

Część II

Gdzie kończy się przewidywanie

Jedna linijka algebry, każde możliwe zachowanie

Oto reguła, której można by nauczyć dziecko. Weź liczbę między 0 a 1. Pomnóż ją przez jeden minus ona sama, a wynik pomnóż przez ustaloną liczbę r . To jest następna liczba. Zrób z nią to samo, i jeszcze raz, i jeszcze. Żadnych kości do gry, żadnych pomiarów, żadnego świata zewnętrznego: każdy krok jest dokładnie wyznaczony przez poprzedni.

Zanim przeczytasz dalej, zgadnij. Jeśli będziesz powoli podkreślać jedyne pokrętko tej reguły, r , ile różnych rodzajów zachowania spodziewasz się zobaczyć: jeden, dwa, garść? Ten rozdział kręci pokrętkiem i patrzy. Pod koniec zobaczysz listę liczb, która nigdy się nie ustala i nigdy nie powtarza, wygląda, jakby wypadła z ruletki, a powstała wyłącznie z tej jednej linijki.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- iterować odwzorowanie logistyczne ręcznie przez krok lub dwa, a w Pythonie przez tyle kroków, ile zechcesz
- znaleźć stan ustalony odwzorowania dla danego r i na podstawie nachylenia w tym punkcie powiedzieć, czy się utrzymuje
- narysować diagram pajęczynowy dla zakrzywionej reguły i odczytać z niego, co robi orbita
- odróżnić stan ustalony, cykl i orbitę, która nigdy się nie powtarza, oraz powiedzieć, co program szukający cykli może, a czego nie może stwierdzić
- powiedzieć, co znaczy *deterministyczny* i *chaotyczny* i dlaczego wyglądać na losowe to nie to samo, co być losowym

5.1 Reguła i jej jedyne pokrętko

Rozdział 3 skończył się na populacji, której wzrost zwalnia w pobliżu granicy: przy łagodnym tempie się ustalała, a przy szybszym przestrzeliwała i się wahała. Ten rozdział zapisuje regułę tego rodzaju w postaci, która przyniosła jej sławę. Niech x oznacza populację jako ułamek największej możliwej, czyli liczbę między 0 a 1. Przez x_n , czytane „ x z indeksem n ”, oznaczaj jej wartość po n krokach, tak że x_0 to stan początkowy. Reguła brzmi

$$x_{n+1} = r x_n (1 - x_n).$$

Słowami: następna wartość to r razy obecna wartość razy jeden minus obecna wartość. Czynniki x_n mówi, że więcej rodziców to więcej młodych; czynnik $1 - x_n$ to sprzężenie

zwrotne, pozostałe miejsce, które maleje, gdy populacja rośnie. Liczba r to pokrętko wzrostu. Regułę, która zamienia liczbę na następną, nazywa się *odwzorowaniem*, a ta jest *odwzorowaniem logistycznym*. W porównaniu z regułami z rozdziału 2, $x_{n+1} = a x_n + b$, ma jedno dodatkowe mnożenie, przez $1 - x_n$, i wszystko w tym rozdziale bierze się z tego jednego czynnika.

Iloczyn $x(1 - x)$ jest największy dla $x = \frac{1}{2}$, gdzie wynosi $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$, więc następna wartość nigdy nie przekracza $r/4$. Dla r nie większego niż 4 liczba x nigdy więc nie opuszcza przedziału od 0 do 1.

Zatrzymaj się i pomyśl 5.1

Weź $r = 2,8$ i zacznij od $x_0 = 0,5$. Ile wynosi x_1 ? A x_2 ? Pierwsze policzysz w pamięci; do drugiego wystarczy co najwyżej ołówek.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $x_1 = 2,8 \times 0,5 \times 0,5 = 0,7$, a $x_2 = 2,8 \times 0,7 \times 0,3 = 0,588$.

To wszystko, a komputer robi miliony takich kroków na sekundę. Oto reguła w postaci, w jakiej zapisuje ją kod książki.

code/ch05/four_rates.py

```
11 def step(r: float, x: float) -> float:
12     """One step of the logistic map: the next value from this one."""
13     return r * x * (1.0 - x)
```

Listing 5.1: Odwzorowanie logistyczne: jedna linijka arytmetyki

Każdy komputer zaokrągla mnożenie i odejmowanie dokładnie tak samo, więc liczby w tym rozdziale wychodzą identyczne co do ostatniej cyfry na każdej maszynie. Przy regule, której cały sens, jak zaraz zobaczysz, tkwi w cyfrach, to ważne.

5.2 Kręcenie pokrętkiem

Każde uruchomienie zaczynaj od $x_0 = 0,2$. Pozwól regule zrobić 200 kroków bez patrzenia, żeby *orbita* – lista wartości, które reguła wytwarza z danego początku – miała czas wejść w to, co będzie robić, a potem zapisz osiem następnych wartości. Zrób to dla czterech ustawień pokrętła: $r = 2,8$, $3,2$, $3,5$ i $3,9$.

Zatrzymaj się i pomyśl 5.2

Rozdział 3 pokazał, że przy łagodnym tempie wzrostu populacja się ustala, a przy szybszym przestrzeliwuje i się waha. Te cztery ustawienia sięgają od umiarkowanego do szybkiego. Czego spodziewasz się po 200 krokach przy $r = 3,9$: jednej powtarzanej wartości, wahaniami między kilkoma wartościami, czy czegoś innego?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Czegoś innego. Przy $r = 3,9$ wartości nigdy się nie ustalają i nigdy nie układają się we wzór.

code/ch05/four_rates.py

```

18 def after(r: float, x0: float, skip: int, keep: int) -> list[float]:
19     """Take `skip` steps from x0 without looking, then keep `keep` values."""
20     x = x0
21     for _ in range(skip):
22         x = step(r, x)
23     values = []
24     for _ in range(keep):
25         values.append(x)
26         x = step(r, x)
27     return values
28
29
30 def main() -> None:
31     print("after 200 steps from 0.2, the next eight values:")
32     for r in (2.8, 3.2, 3.5, 3.9):
33         values = after(r, 0.2, 200, 8)
34         print(f"r = {r}: " + " ".join(f"{v:.3f}" for v in values))

```

Listing 5.2: Dwieście kroków bez patrzenia, potem osiem kolejnych, przy czterech ustawieniach

```

after 200 steps from 0.2, the next eight values:
r = 2.8: 0.643 0.643 0.643 0.643 0.643 0.643 0.643 0.643
r = 3.2: 0.799 0.513 0.799 0.513 0.799 0.513 0.799 0.513
r = 3.5: 0.827 0.501 0.875 0.383 0.827 0.501 0.875 0.383
r = 3.9: 0.400 0.936 0.233 0.697 0.824 0.566 0.958 0.157

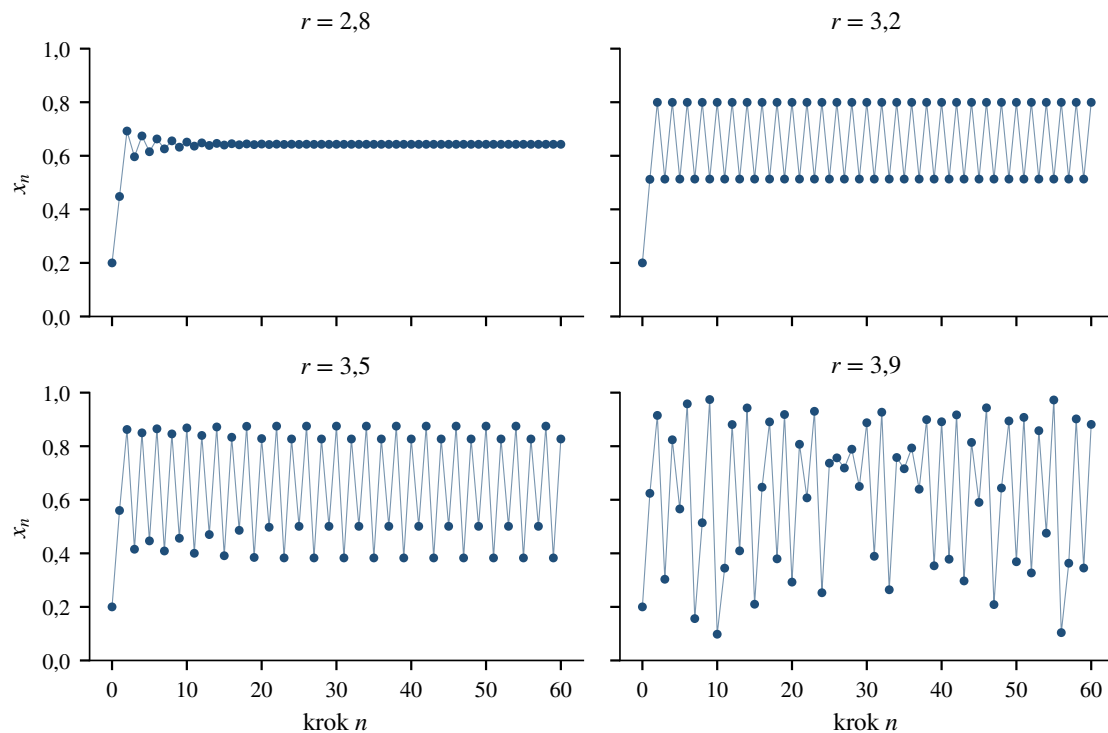
```

Każdy wiersz to inny rodzaj zachowania. Przy $r = 2,8$ populacja spoczęła na 0,643: to *stan ustalony*. Przy $r = 3,2$ bez końca przeskakuje między 0,513 a 0,799, rok wysoki, rok niski: to *cykl o okresie dwa*. Przy $r = 3,5$ obiega cztery wartości, od 0,383 do 0,875, i znowu te same cztery: cykl o okresie cztery. Przy $r = 3,9$ liczby skaczą bez żadnego widocznego wzoru i, jak sprawdzisz niżej, żaden wzór się nie pojawia, choćbyś czekał dowolnie długo.

Na rysunku 5.1 każdy panel zaczyna się od zamieszania, a trzy z czterech szybko łapią swój rytm. Czwarty nigdy.

Pułapka

„Prosta reguła daje proste zachowanie.” Wszystkie cztery panele pochodzą z tej samej linijki arytmetyki; zmieniło się tylko r , z 2,8 na 3,9. Zachowanie przeszło od jednej spoczynkowej wartości do liczb, które nigdy się nie powtarzają. To, jak skomplikowane wydaje się zachowanie układu, bardzo mało mówi o tym, jak skomplikowana jest jego reguła.



Rysunek 5.1: Cztery ustawienia przez pierwsze sześćdziesiąt kroków, wszystkie od $x_0 = 0,2$. Zwróć uwagę, jak kończy się każdy panel: spoczynkiem, przeskakiwaniem, obieganiem czterech wartości i ciągłym skakaniem.

5.3 Stan ustalony i chwila, w której puszcza

Punkt stały to wartość, której reguła nie zmienia, a *stan ustalony* to punkt stały, na którym orbita spoczęła. Odwzorowanie logistyczne zawsze ma jeden w $x = 0$: brak populacji pozostaje brakiem populacji. Żeby znaleźć drugi, zapytaj, kiedy $r x (1 - x)$ równa się x . Podziel obie strony przez x , a dostaniesz $r (1 - x) = 1$, czyli $1 - x = 1/r$, i

$$x^* = 1 - \frac{1}{r}.$$

Gwiazdka to zwyczajowe oznaczenie punktu stałego.

Zatrzymaj się i pomyśl 5.3

Dla $r = 2$ wzór daje $x^* = \frac{1}{2}$. Sprawdź, że reguła naprawdę nie zmienia $\frac{1}{2}$. Co wzór daje dla $r = 4$ i czy to też się zgadza?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $2 \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{2}$. Dla $r = 4$ daje $\frac{3}{4}$, a $4 \times \frac{3}{4} \times \frac{1}{4} = \frac{3}{4}$.

Przy $r = 2,8$ wzór daje 0,643, dokładnie tam, gdzie spoczął wydruk. Przy $r = 3,2$ daje 0,688 – a orbita zamiast tego krąży wokół tej wartości, od 0,513 do 0,799. Punkt stały istnieje; orbita nie chce na nim zostać.

Rozdział 3 wyjaśnił to pomiarem. Przesuń stan odrobinę od punktu stałego i zrób jeden krok: przesunięcie wraca pomnożone przez *nachylenie* reguły w tym miejscu. Jeśli leży ono między -1 a 1 , przesunięcia maleją i punkt stały się utrzymuje; poza tym zakresem rosną i orbita zostaje odepchnięta. Ujemne nachylenie dodatkowo przerzuca przesunięcie na drugą stronę przy każdym kroku. Funkcja biblioteki `slope` mierzy je, przesuwając x o jedną milionową w każdą stronę.

code/ch05/fixed_point.py

```

10 from chaoslab import logistic, slope
11
12
13 def fixed_point(r: float) -> float:
14     """The value, other than zero, that the rule leaves unchanged."""
15     return 1.0 - 1.0 / r
16
17
18 def main() -> None:
19     print(" r    fixed point    slope there    a nudge")
20     for r in (2.8, 3.2, 3.5, 3.9):
21         x = fixed_point(r)
22         s = slope(logistic(r), x)      # nudge x both ways, measure the move
23         fate = "shrinks" if abs(s) < 1 else "grows"
24         print(f"{r:.1f}    {x:11.3f}    {s:11.2f}    {fate}")

```

Listing 5.3: Punkt stały i nachylenie reguły w tym miejscu

r	fixed point	slope there	a nudge
2.8	0.643	-0.80	shrinks
3.2	0.688	-1.20	grows
3.5	0.714	-1.50	grows
3.9	0.744	-1.90	grows

Każde nachylenie wynosi $2 - r$: $2 - 2,8 = -0,8$, $2 - 3,2 = -1,2$ i tak dalej. (Rachunek różniczkowy mówi, że nachylenie $r x (1 - x)$ to $r (1 - 2x)$, a w $x = 1 - 1/r$ jest to dokładnie $2 - r$.) Przy $r = 2,8$ przesunięcie przy każdym kroku zmienia stronę i maleje, więc orbita się zbliża. Przy $3,2$ zmienia stronę i rośnie, więc orbita zostaje odrzucona.

Zatrzymaj się i pomyśl 5.4

Gdy podkręcasz r od $2,8$ w górę, przy jakiej wartości nachylenie $2 - r$ osiąga -1 , tak że punkt stały przestaje się utrzymywać?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Przy $r = 3$, bo $2 - 3 = -1$. Poniżej 3 stan ustalony się utrzymuje; powyżej 3 puszcza.

To pierwsze miejsce, w którym pokrętko zmienia rodzaj zachowania, i znalazłeś je ręcznie. Przejdźcie od dwóch wartości do czterech następuje wyżej; rozdział 8 znajduje, gdzie, i wzór w tym, gdzie.

Laboratorium 5.1 – Gdzie stan ustalony puszcza

Napisz `fixed_point`; `slope_at`, która mierzy nachylenie w punkcie, przesuwając go w obie strony; oraz `holds`, która mówi, czy przesunięcie w punkcie stałym maleje. Potem napisz `lets_go`, która znajduje, gdzie `holds` zmienia się na fałsz między 2,5 a 3,5, dzieląc przedział na pół raz za razem. Wynik powinien się zgadzać z odpowiedzią powyżej.

Plik startowy: `code/labs/ch05/l05_01_stability.py`

Test: `code/labs/ch05/test_l05_01_stability.py`

Uruchom: `cd code && uv run pytest -k l05_01_stability`

5.4 Pajęczyna na krzywej

Rozdział 2 narysował iterację jako pajęczynę na linii prostej. Ten sam rysunek działa dla każdej reguły. Narysuj wykres reguły, wysokość $r \cdot x(1-x)$ dla każdego x od 0 do 1: garb, najwyższy w $x = \frac{1}{2}$. Narysuj przekątną, na której wysokość równa się x . Zaczynij od x_0 na dolnej krawędzi i idź w górę do garbu: wysokość, na którą dojdiesz, to x_1 . Idź poziomo do przekątnej, co stawia cię nad x_1 ; w górę albo w dół do garbu po x_2 ; znów poziomo; i tak dalej. Garb przecina przekątną w punkcie stałym, gdzie następna wartość równa się obecnej. Funkcja biblioteki `cobweb` wypisuje narożniki tej ścieżki.

`code/ch05/cobweb_data.py`

```

10 from chaoslab import cobweb, logistic
11
12
13 def main() -> None:
14     corners = cobweb(logistic(2.8), 0.2, 3)          # three steps from 0.2
15     moves = ["start"] + ["up", "across"] * 3
16     for move, (x, y) in zip(moves, corners, strict=True):
17         print(f"{move:<7} x = {x:.3f} height = {y:.3f}")

```

Listing 5.4: Narożniki pajęczyny, trzy kroki

start	x = 0.200	height = 0.000
up	x = 0.200	height = 0.448
across	x = 0.448	height = 0.448
up	x = 0.448	height = 0.692
across	x = 0.692	height = 0.692
up	x = 0.692	height = 0.596
across	x = 0.596	height = 0.596

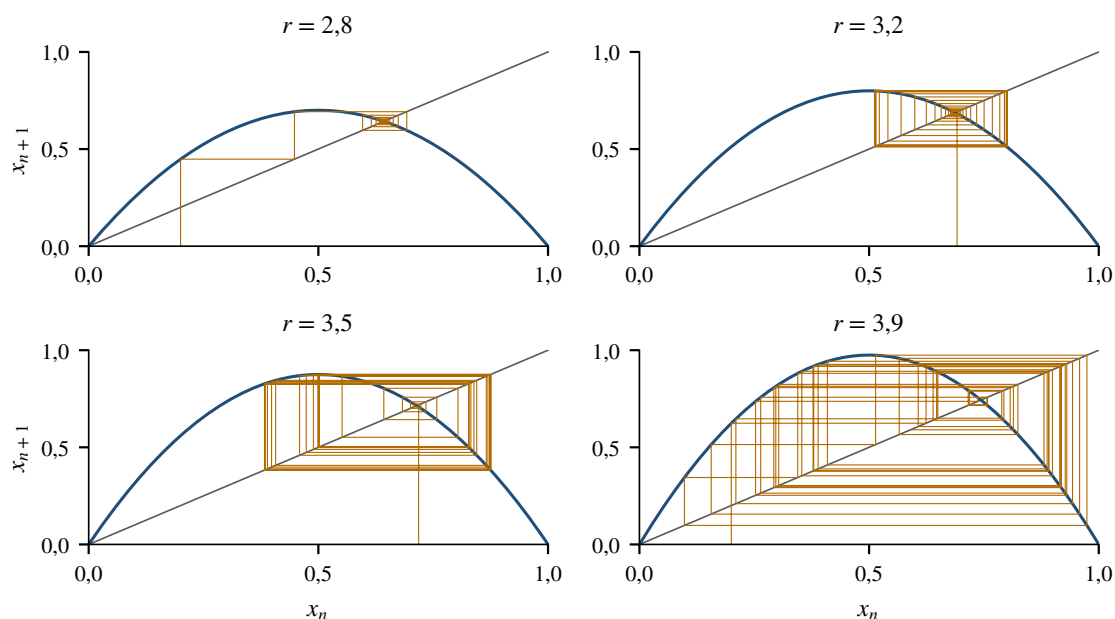
Każdy ruch up (w górę) ląduje na garbie w następnej wartości orbity, dla pierwszego $2,8 \times 0,2 \times 0,8 = 0,448$; każdy ruch across (w poprzek) ląduje na przekątnej na tej samej wysokości.

Zatrzymaj się i pomyśl 5.5

Przy $r = 2,8$ pajęczyna zwija się spiralnie do punktu przecięcia. Przy $r = 3,2$ orbita w końcu bez końca przeskakuje między dwiema wartościami. W jaki kształt układa się jej pajęczyna?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. W zamknięty prostokąt z dwoma narożnikami na garbie: w górę do jednej wartości, poziomo, w dół do drugiej, poziomo i z powrotem do początku.



Rysunek 5.2: Pajęczyny dla czterech ustawień, każda na własnym garbie z przekątną. Przy $r = 2,8$ ścieżka zwija się do środka; przy 3,2 i 3,5 zaczyna tuż obok przecięcia, zostaje odepchnięta i układa się w zamkniętą pętlę; przy 3,9 nigdy się nie zamyka.

Spójrz, gdzie garb przecina przekątną na rysunku 5.2. Przy $r = 2,8$ opada przez punkt przecięcia łagodniej, niż wznosi się przekątna, i spirala się zaciska. Przy 3,2 opada bardziej stromo i ścieżka zaczęta obok przecięcia rozkręca się na zewnątrz, aż zatrzyma się na prostokącie. Krzywa opadająca dokładnie tak stromo, jak wznosi się przekątna, ma nachylenie -1 : rysunek i pomiar się zgadzają. Przy 3,5 pętla ma cztery narożniki na garbie. Przy 3,9 ścieżka nigdy się nie zamyka i powoli zapełnia cały pas.

5.5 Szukanie cyklu, także daremne

Przy tysiącach liczb niech cykl odczyta komputer. Funkcja biblioteki `period` szuka w ostatnich 128 wartościach orbity najmniejszej liczby całkowitej p takiej, że każda wartość równa się tej o p kroków późniejszej, z dokładnością do jednej miliardowej. Zwraca to p albo pythonowe słowo oznaczające nic, `None`, gdy żadne p do 64 nie pasuje.

code/ch05/periods.py

```

11 from chaoslab import logistic, orbit, period
12
13
14 def main() -> None:
15     for r in (2.8, 3.2, 3.5, 3.9, 4.0):
16         xs = orbit(logistic(r), 0.2, 2000) # the start and 2000 steps
17         print(f"r = {r}: period {period(xs)}")

```

Listing 5.5: Pytanie o okres po dwóch tysiącach kroków

```

r = 2.8: period 1
r = 3.2: period 2
r = 3.5: period 4
r = 3.9: period None
r = 4.0: period None

```

Detektor zgadza się z wydrukiem: 1, potem 2 i 4. Przy $r = 3,9$ i przy $r = 4$, górnym końcu zakresu, nie znajduje nic.

Zatrzymaj się i pomyśl 5.6

Przy $r = 3,9$ detektor nie znalazł na końcu dwóch tysięcy kroków żadnego cyklu o długości do 64 kroków. Przypuśćmy, że puścisz regułę na sto tysięcy kroków. Czy powtórzenie w końcu się pojawi?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. W stu tysiącach kroków nie wraca ani jedna wartość.

code/ch05/never_repeats.py

```

11 from chaoslab import logistic, orbit
12
13
14 def main() -> None:
15     for r in (3.9, 4.0):
16         xs = orbit(logistic(r), 0.2, 100_000)
17         again = orbit(logistic(r), 0.2, 100_000)
18         print(f"r = {r}: {len(xs)} values, {len(set(xs))} different")
19         print(f"same start again, same list: {xs == again}")

```

Listing 5.6: Sto tysięcy kroków w poszukiwaniu wartości, która już była

```

r = 3.9: 100001 values, 100001 different
        same start again, same list: True
r = 4.0: 100001 values, 100001 different
        same start again, same list: True

```

Zbiór set w Pythonie przechowuje po jednej kopii każdej różnej wartości, więc wartość, która by wróciła, zmniejszyłaby drugą liczbę. Tak się nie dzieje: wszystkie 100 001 wartości są różne. To rozstrzyga sprawę dla tych 100 000 kroków, bo gdyby jakaś wartość kiedyś wróciła, wróciłaby też następna – ta sama liczba zawsze daje tę samą następną – i wszystko, co po niej: powstałby cykl. Skoro żadna wartość nie wraca, żaden cykl się nie zaczął. Drugi wiersz każdej pary przyda się później: ten sam początek, uruchomiony ponownie, daje dokładnie te same liczby.

Pułapka

„Skoro nie powtórzyło się po tysiącu kroków, to zaraz się powtórzy.” Długie czekanie sprawia, że powtórzenie wydaje się spóźnione, tak jak moneta, która pięć razy pokazała orła, zdaje się być winna reszkę. Orbita nic ci nie jest winna. Żaden cykl jej nie przyciąga, a dłuższe liczenie daje tylko więcej liczb, które nigdy wcześniej się nie pojawiły.

Czego tu nie dowodzimy

Przebieg programu to świadectwo, a nie dowód. Dla $r = 4$ można udowodnić, że z niemal każdego początku orbita nigdy się nie powtarza; dla $r = 3,9$ świadectwem są obliczenia takie jak te. A komputer mieści tylko skończenie wiele liczb między 0 a 1, więc obliczona orbita musi w końcu trafić na wartość, którą już miała, i wejść w cykl – po większej liczbie kroków, niż zrobił ten przebieg. Rozdział 16 mówi, co to znaczy dla symulacji.

Laboratorium 5.2 – Własny detektor cykli

Napisz `settle`, która puszcza regułę bez patrzenia, a potem zachowuje kilka następnych wartości, oraz `period`, która znajduje najmniejsze p , dla którego każda wartość na liście równa się tej o p miejsc dalej, i zwraca `None`, gdy żadne p do ustalonej granicy nie pasuje. Testy używają wymyślonych list i orbit z tego podrozdziału.

Plik startowy: `code/labs/ch05/l05_02_period.py`

Test: `code/labs/ch05/test_l05_02_period.py`

Uruchom: `cd code && uv run pytest -k l05_02_period`

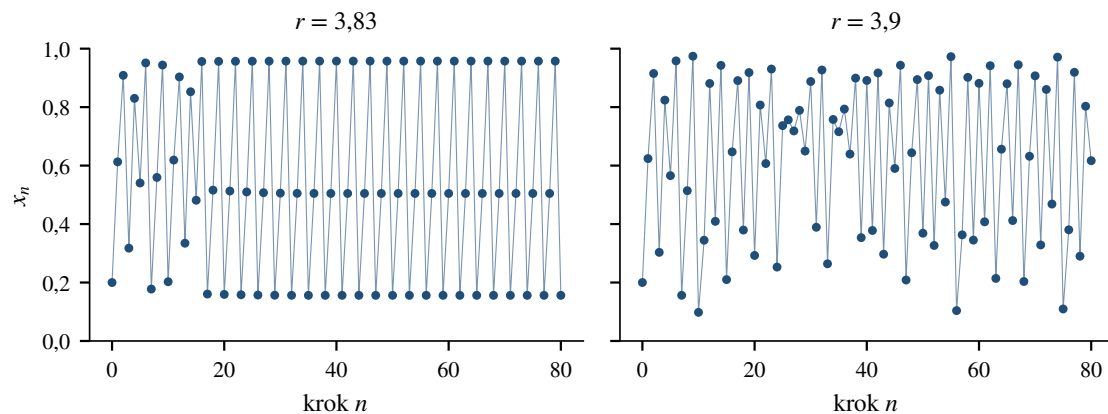
Zatrzymaj się i pomyśl 5.7

Detektor znalazł cykl długości cztery przy $r = 3,5$ i nic przy 3,9. Czego spodziewasz się przy $r = 3,83$, trochę poniżej 3,9?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Cyklu o okresie trzy. Po krótkim zamieszaniu orbita bez końca obiega 0,156, 0,505 i 0,957.

Zaskakuje to niemal każdego i słusznie. Podkręcanie pokrętła nie czyni zachowania coraz

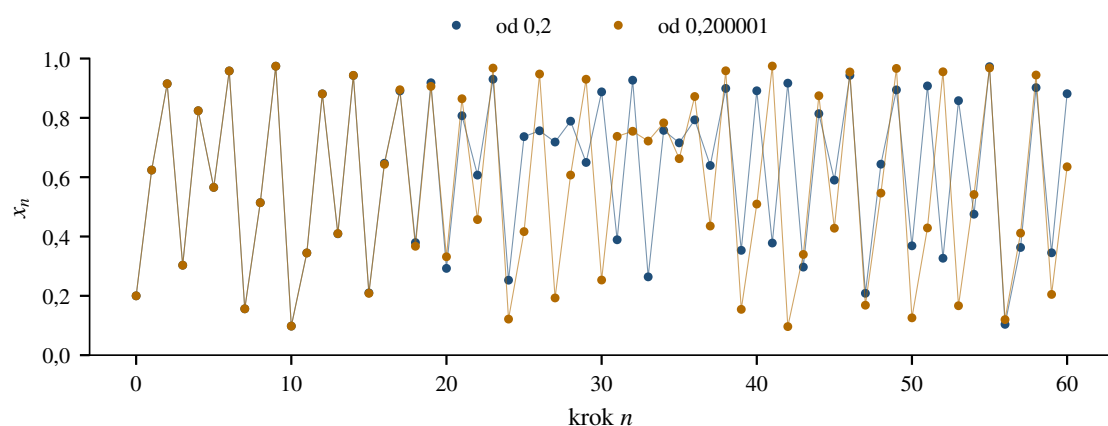


Rysunek 5.3: Porządek wewnątrz skakania. Przy $r = 3,83$ orbita przez chwilę się miota, a potem wpada w cykl o długości trzy; przy pobliskim $r = 3,9$ nigdy się nie ustala.

dzijszym w równym tempie: wśród ustawień, przy których orbity skaczą bez wzoru, są wąskie okna, w których porządek wraca. Gdzie leżą i co oznacza cykl o długości trzy, to część pełnego obrazu, który rysuje rozdział 8.

5.6 Wygląda na losowe, a nie jest

Czas na nazwy. Reguła jest *deterministyczna*, gdy teraźniejszość całkowicie wyznacza następny stan i nic nie zostaje przypadkowi, więc ten sam początek zawsze daje tę samą orbitę. Widziałeś, że odwzorowanie logistyczne takie jest: uruchomione dwa razy od 0,2, dało te same sto tysięcy liczb co do ostatniej cyfry. Orbita jest *chaotyczna*, gdy pochodzi z reguły deterministycznej, pozostaje w ograniczonym zakresie, nigdy nie wychodzi w spoczynek ani w cykl – i, czego jeszcze nie zmierzyłeś, jest tak wrażliwa na początek, że każda różnica w nim, choćby najmniejsza, w końcu rośnie, aż dwie przyszłości nie mają ze sobą nic wspólnego.



Rysunek 5.4: Dwa przebiegi przy $r = 3,9$, od 0,2 i od 0,200001, o jedną milionową wyżej. Przez chwilę linie leżą jedna na drugiej; potem rozchodzą się na dobre.

Ta ostatnia część to pomiar z rozdziału 6; rysunek 5.4 jest tylko zapowiedzią. Razem dają najdziwniejszy jak dotąd fakt w tej książce. Orbita przy $r = 3,9$ wygląda na losową, a nie

jest: każda liczba w niej została dokładnie wyznaczona przez poprzednią. A jednak nie przewidzisz jej daleko naprzód, jeśli nie znasz jej początku dokładnie, niż ktokolwiek zna cokolwiek.

W 1976 roku Robert May, fizyk, który przeszedł do ekologii, opublikował w *Nature* artykuł przeglądowy, którego tytuł mówi to, co pokazał ten rozdział: *Simple mathematical models with very complicated dynamics*, czyli proste modele matematyczne o bardzo skomplikowanej dynamice. Napisał go dla biologów.

Zatrzymaj się i pomyśl 5.8

Ekolog co lato przez dwadzieścia lat liczy owady na polu. Liczebności skaczą w górę i w dół bez wzoru, którego ktokolwiek potrafiłby się dopatrzeć. Co to mówi o przyczynie?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniej, niż się wydaje. Może to sprawiać pogoda albo błędy w liczeniu – albo wyłącznie własna reguła wzrostu populacji.

Ta trzecia możliwość była sednem argumentu Maya. Nieregularne liczby z terenu zwykle odczytywano jako ślad niespokojnego środowiska albo niedokładnego liczenia; odwzorowanie logistyczne pokazuje, że jedno pokolenie na rok i jedna prosta reguła wzrostu mogą dać taką samą nieregularność bez żadnego zakłócenia. May zakończył apelem, który sięga poza ekologie: żeby uczniowie poznawali to równanie wcześniej, bo intuicja wyćwiczona wyłącznie na regułach prostoliniowych zostawia ludzi nieprzygotowanymi na to, co robią reguły zakrzywione. Ty już je poznałeś.

Pułapka

„Nieregularny wynik wymaga przypadkowej albo skomplikowanej przyczyny.”

Wiersz dla $r = 3,9$ w podrozdziale 5.2 powstał z tej samej linijki arytmetyki co wiersze nad nim, bez żadnej losowości. Nieregularny wynik może pochodzić z przypadku, ze skomplikowanego mechanizmu albo z prostej reguły deterministycznej, a sam wynik nie mówi, z którego. Rozdział 15 uczy, jak je od siebie odróżnić.

Gdzie to spotkasz

Nauka o rybołówstwie spotyka to samo zachowanie w innym wzorze. Model zależności stado–rekrutacja Williama Rickera z 1954 roku, wiążący liczbę ryb, które odbyły tarło w danym roku, z liczbą młodych, jakie wydały, to jednolinijkowa reguła tego samego rodzaju: wzrost, który zwalnia, gdy liczebność rośnie, pokolenie po pokoleniu. Podkreśl w nim tempo wzrostu, a przejdzie, w tej samej kolejności co odwzorowanie logistyczne, od stałego stada przez naprzemienne lata obfitości i nędzy do nieregularnych wahań. Połów, który waha się z roku na rok, sam w sobie nie dowodzi, że winna jest pogoda.

Jedna linijka, jedno pokrętko: stan ustalony, cykle długości dwa i cztery, orbity, które

nigdy się nie powtarzają, i wyspy porządku między nimi. Brakuje jeszcze miary tego, jak nieprzewidywalny jest ten chaos. Rozdział 6 liczy kroki, po których dwa bliskie początki się rozchodzą, rozdział 7 zamienia to w jedną liczbę, a rozdział 8 rysuje każde ustawienie pokręta na jednym obrazie.

Podsumowanie

- **Odwzorowanie logistyczne**, $x_{n+1} = r x_n (1 - x_n)$, to reguła prostoliniowa z jednym dodatkowym mnożeniem, sprzężeniem zwrotnym $1 - x_n$. Dla r nie większego niż 4 liczba x pozostaje między 0 a 1.
- Kręcenie r daje **stan ustalony** przy 2,8, **cykl o okresie dwa** przy 3,2, cykl o okresie cztery przy 3,5, a przy 3,9 orbitę, która nigdy się nie ustala ani nie powtarza.
- **Punkt stały** to $x^* = 1 - 1/r$, a **nachylenie** w nim wynosi $2 - r$, więc stan ustalony utrzymuje się dla r poniżej 3.
- **Pajęczyna** zwija się do punktu stałego, który się utrzymuje, zostaje odepchnięta od takiego, który się nie utrzymuje, układa się w zamkniętą pętlę dla cyklu, a przy 3,9 nigdy się nie zamyka.
- **Detektor cykli** podaje 1, 2, 4, a potem None, co znaczy „nie znaleziono”, nigdy „udowodniono chaos”. W stu tysiącach kroków nie wraca żadna wartość; przy 3,83 **okno** kryje cykl o długości trzy.
- **Deterministyczny**: ten sam początek, ta sama orbita, za każdym razem. **Chaotyczny**: deterministyczny, ograniczony, nigdy się nieustalający, wrażliwy na początek. Wygląda na losowy, a nie jest – przestroga Maya z 1976 roku.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

iterować odwzorowanie logistyczne ręcznie przez krok lub dwa, a w Pythonie przez tyle kroków, ile zechcesz	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
znaleźć stan ustalony odwzorowania dla danego r i na podstawie nachylenia w tym punkcie powiedzieć, czy się utrzymuje	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
narysować diagram pajęczynowy dla zakrzywionej reguły i odczytać z niego, co zrobi orbita	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
odróżnić stan ustalony, cykl i orbitę, która nigdy się nie powtarza, oraz powiedzieć, co program szukający cykli może, a czego nie może stwierdzić	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co znaczy <i>deterministyczny</i> i <i>chaotyczny</i> i dlaczego wyglądać na losowe to nie to samo, co być losowym	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dla $r = 3$ i $x_0 = 0,5$ ile wynoszą x_1 i x_2 ?
2. Gdzie leży niezerowy punkt stały dla $r = 2,5$ i czy się utrzymuje?
3. Przy $r = 3,2$ punkt stały leży w 0,688, a jednak orbita w nim nie zostaje. Dlaczego?
4. Pajęczyna układa się w zamkniętą pętlę z czterema narożnikami na garbie. Co robi orbita?
5. Detektor cykli zwraca None. Co możesz stwierdzić, a czego nie?
6. Roczne liczebności populacji zwierząt skaczą bez wzoru. Wymień trzy możliwe przyczyny.
7. Co znaczy, że odwzorowanie logistyczne jest deterministyczne, i jak jego wynik może mimo to wyglądać na losowy?

Efekt motyla, zmierzony

Rozdział 1 zakończył się pytaniem, na które odpowiedział tylko dla łagodnych modeli. Jeśli reguła jest dokładna, to czy przyszłość jest znana? W przypadku stygnącej herbaty odpowiedź brzmiała: prawie tak. Dwa kubki, które zaczynały z różnicą jednego stopnia, z każdą minutą zbliżały się do siebie, więc termometr, który na początku trochę się mylił, później prawie nie miał znaczenia.

Ten rozdział zadaje to samo pytanie jednolinkowej regule z rozdziału 5 i mierzy odpowiedź. Uruchom regułę dwa razy, z dwóch liczb, które różnią się na dziesiątym miejscu po przecinku – są sobie bliższe, niż jakikolwiek termometr, linijka czy waga kuchenna mogłyby kiedykolwiek odróżnić – i pozwól obu biec. Jak długo pozostaną razem? Odpowiedź to liczba kroków, jest taka sama na każdym komputerze i jest krótka.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- uruchomić jedną regułę z dwóch bliskich punktów startowych i odczytać oba przebiegi obok siebie
- zmierzyć różnicę między dwoma przebiegami krok po kroku i odczytać ją w skali logarytmicznej
- powiedzieć, co znaczy wrażliwość na warunki początkowe, w kategoriach czegoś, co da się zmierzyć
- wyjaśnić za pomocą odwzorowania podwajającego, dlaczego prognoza się kończy
- powiedzieć, dla danego błędu punktu startowego, mniej więcej przez ile kroków przewidywanie pozostaje użyteczne
- odróżnić determinizm od przewidywalności i wiernie opowiedzieć o odkryciu Lorenza

6.1 Dwa punkty startowe, jedna dziesięciomiliardowa

Reguła z rozdziału 5 to $x_{n+1} = r x_n (1 - x_n)$. Słowami: weź liczbę x_n , którą masz po n krokach, pomnóż ją przez $1 - x_n$ i przez pokrętło r , a wynikiem jest następna liczba, x_{n+1} . W tym rozdziale pokrętło ustawiamy na $r = 4$, największą wartość, przy której każde x pozostaje między 0 a 1.

Weź dwa punkty startowe: 0,2 oraz 0,2000000001. Różnią się o jedną dziesięciomiliardową, 10^{-10} : przecinek, dziewięć zer i jedynek. Przebiegi, które z nich wyjdą, nazwijmy bliźniakami.

Zatrzymaj się i pomyśl 6.1

Uruchom regułę z obu punktów. Mniej więcej przez ile kroków oba przebiegi pozostaną od siebie bliżej niż o 0,1: około dziesięciu, około trzydziestu, około trzech tysięcy czy zawsze?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około trzydziestu. Oba przebiegi po raz pierwszy różnią się o więcej niż 0,1 w kroku 30.

Większość ludzi zgaduje tysiące albo „zawsze”, bo różnica na dziesiątym miejscu po przecinku wydaje się żadną różnicą. Dla herbaty rzeczywiście by nią nie była. Oto program, który to rozstrzyga.

code/ch06/twins.py

```

12 START = 0.2
13 NUDGE = 1e-10          # one part in ten billion
14 rule = logistic(4.0)   # x -> 4 * x * (1 - x)
15
16
17 def twin_runs(steps: int) -> tuple[list[float], list[float]]:
18     """Run the rule from START and from START + NUDGE; keep every step."""
19     a, b = [START], [START + NUDGE]
20     for _ in range(steps):
21         a.append(rule(a[-1]))
22         b.append(rule(b[-1]))
23     return a, b

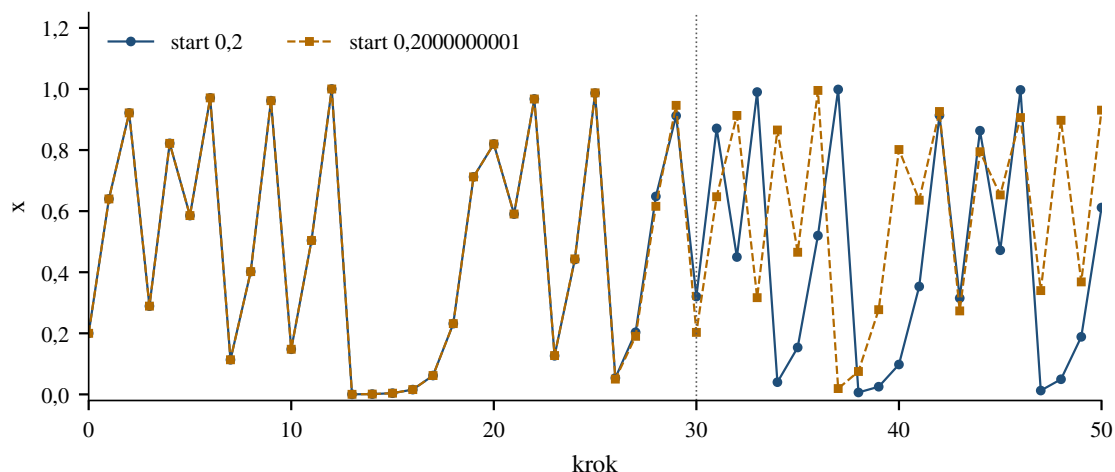
```

Listing 6.1: Dwa przebiegi jednej reguły z dwóch bliskich punktów startowych

`logistic(4.0)` to reguła z rozdziału 5 z pokrętle ustawionym na 4, wzięta z `chaoslab`, małej biblioteki, z której korzystają programy wszystkich rozdziałów. `twin_runs` prowadzi dwie listy, `a` i `b`, i w każdym kroku podaje ostatnią liczbę każdej z nich z powrotem tej samej regule. Potem program wypisuje obie listy co piąty krok:

step 0:	0.2000000000	0.2000000001
step 5:	0.5854205387	0.5854205348
step 10:	0.1478365599	0.1478366508
step 15:	0.0039360251	0.0039365381
step 20:	0.8200138734	0.8199131530
step 25:	0.9863789723	0.9873341848
step 30:	0.3203424285	0.2030211838
step 35:	0.1535471506	0.4653612556
step 40:	0.0978744041	0.8017408721

Czytaj obie kolumny od góry. Aż do kroku 16 przebiegi różnią się o mniej niż jedną milionową i każdy, kto przestałby czytać w tym miejscu, uznałby je za ten sam przebieg. W kroku 20 różnią się mniej więcej o jedną dziesięciotysięczną. W kroku 30 nie zgadzają się nawet na pierwszym miejscu po przecinku, a od tej chwili są dwiema niezwiązanymi ze sobą listami liczb.



Rysunek 6.1: Dwa przebiegi z listingu, krok po kroku. Biegają jedną ścieżką mniej więcej do kroku 30 (linia kropkowana), a po nim dwiema niezwiązanymi.

6.2 Różnica się podwaja

W rozdziale 1 obserwowałeś różnicę między dwoma kubkami. Zrób to samo tutaj. Poniższy program uruchamia oba punkty startowe, zapisuje odległość między przebiegami po każdym kroku i wypisuje obok, ile wynosiłaby ta różnica, gdyby podwajała się dokładnie w każdym kroku. Najpierw pytanie, na które odpowiesz arytmetyką podwajania z rozdziału 2.

Zatrzymaj się i pomyśl 6.2

Jeśli różnica podwaja się w każdym kroku, to ile kroków potrzeba, żeby wzrosła tysiąc razy? A miliard razy?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Dziesięć kroków na wzrost tysiąckrotny, bo $2^{10} = 1024$.
Trzydzieści na miliardkrotny, bo miliard to tysiąc razy tysiąc razy tysiąc.

Od jednej dziesięciomiliardowej do jednej dziesiątej jest dokładnie miliard razy. Jeśli więc różnica podwaja się w każdym kroku, bliźniaki powinny się rozejść mniej więcej w trzydziestym kroku i dokładnie tak się stało. Sprawdź to krok po kroku:

code/ch06/gap.py

```
14 gaps = separation(rule, START, NUDGE, 45)
15
16 print("step      gap      if it doubled")
17 for n in range(0, 46, 5):
18     print(f"{n:4d}    {gaps[n]:9.2e}    {NUDGE * 2**n:14.2e}")
```

Listing 6.2: Różnica między bliźniakami obok różnicy, która podwaja się dokładnie

step	gap	if it doubled
------	-----	---------------

0	1.00e-10	1.00e-10
5	3.94e-09	3.20e-09
10	9.09e-08	1.02e-07
15	5.13e-07	3.28e-06
20	1.01e-04	1.05e-04
25	9.55e-04	3.36e-03
30	1.17e-01	1.07e-01
35	3.12e-01	3.44e+00
40	7.04e-01	1.10e+02
45	1.81e-01	3.52e+03

Aż do kroku 30 obie kolumny dotrzymują sobie kroku. Krok po kroku zmierzona różnica jest poszarpana – niektóre kroki mnożą ją prawie przez cztery, kilka ją zmniejsza – ale na przestrzeni wielu kroków średnio się podwaja. Wzrost o ten sam czynnik w każdym kroku to wzrost wykładniczy z rozdziału 2, prawo procentu składanego. Tam rosły twoje oszczędności. Tu rośnie twój błąd.

To jest właściwość, od której rozdział bierze nazwę. Reguła ma *wrażliwość na warunki początkowe*, gdy odpycha od siebie bliskie punkty startowe, a różnica między nimi rośnie mniej więcej o ten sam czynnik w każdym kroku, aż oba przebiegi nie mają ze sobą nic wspólnego. Im bliżej siebie startują, tym dłużej to trwa, ale nigdy nie trwa wiecznie. Herbata z rozdziału 1 mnożyła swoją różnicę przez 0,9 co minutę; odwzorowanie logistyczne przy $r = 4$ mnoży ją średnio przez dwa. Jeden czynnik mniejszy od jedności, drugi większy: na tym polega cała różnica między modelem, który ci wybacz, a takim, który nie wybacz.

Zatrzymaj się i pomyśl 6.3

Kolumna podwajania rośnie dalej po kroku 35, przekracza jedynkę, a potem setkę. Dlaczego zmierzona różnica przestaje za nią nadążać?

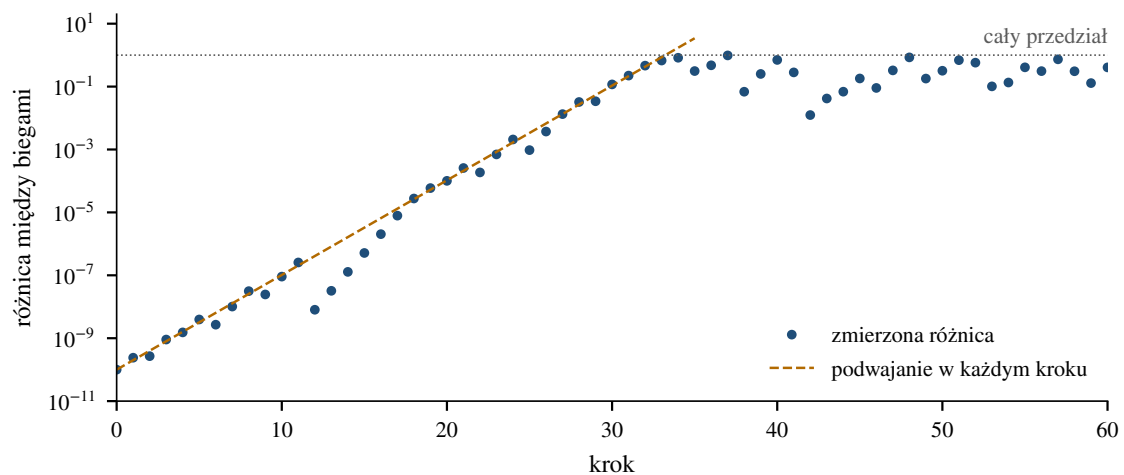
Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Każde x leży między 0 a 1, więc dwa z nich nigdy nie mogą być od siebie dalej niż o 1. Gdy różnica osiągnie rozmiar całego przedziału, nie może już rosnąć: błąka się, średnio wynosząc około 0,4.

W skali logarytmicznej, w której każdy krok w górę to dziesięciokrotność, cała historia to linia prosta i sufit. Porównaj to z rysunkiem herbaty z rozdziału 1: tam prosta opadała, tu się wznosi.

chaoslab

`chaoslab` ma dwie funkcje dla tego rozdziału. `separation` uruchamia regułę z dwóch punktów startowych i zwraca ich odległość po każdym kroku, czyli to, co wypisał powyższy listing; `steps_until_apart` zwraca pierwszy krok, w którym ta odległość przekracza wybraną przez ciebie tolerancję. Laboratorium poniżej prosi cię, żebyś napisał obie sam, zanim im zaufasz.



Rysunek 6.2: Różnica między dwoma przebiegami w skali logarytmicznej. Wspina się, nierówno, wzdłuż przerywanej linii podwajania, aż osiągnie rozmiar całego przedziału i nie może wspiąć się wyżej.

Laboratorium 6.1 – Kiedy dwa przebiegi się rozchodzą?

Napisz gaps, które uruchamia regułę z dwóch punktów startowych i zapamiętuje odległość między nimi w każdym kroku, oraz `first_apart`, które zwraca pierwszy krok, w którym ta odległość przekracza tolerancję, albo `None`, jeśli nigdy jej nie przekracza. Obie przyjmują regułę jako funkcję, więc test sprawdza je także na herbacie z rozdziału 1: jej dwa kubki nigdy się nie rozchodzą. Dla bliźniaków, z tolerancją 0,1, twoja odpowiedź musi brzmieć: krok 30.

Plik startowy: `code/labs/ch06/l06_01_apart.py`

Test: `code/labs/ch06/test_l06_01_apart.py`

Uruchom: `cd code && uv run pytest -k l06_01_apart`

6.3 Skąd bierze się podwajanie

Dlaczego różnica miałaby się podwajać? Dla odwzorowania logistycznego odpowiedź wymaga trochę algebry i daje ją rozdział 7. Jest prostsza reguła, w której odpowiedź widać od razu: *odwzorowanie podwajające*. Podwój liczbę, a jeśli wynik wynosi 1 lub więcej, odejmij 1. Zapisuje się je $x_{n+1} = 2x_n \bmod 1$, gdzie $\bmod 1$ znaczy „zostaw tylko część po przecinku”.

Zatrzymaj się i pomyśl 6.4

Zacznij od 0,625. Zapisz 1, jeśli liczba leży w górnej połowie (0,5 lub więcej), albo 0, jeśli w dolnej, potem zastosuj odwzorowanie podwajające i powtarzaj, aż zapiszesz trzy cyfry.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź.

0,625 leży w górnej połowie: 1. Po podwojeniu daje 1,25, a po odjęciu 1 zostaje 0,25, w dolnej połowie: 0. Podwojone jeszcze raz daje 0,5, w górnej połowie: 1. Zapisales 1, 0, 1.

Teraz zapisz 0,625 w systemie *dwójkowym*. W ułamkach dziesiętnych, których uczyłeś się w szkole, cyfry po przecinku liczą dziesiąte, setne i tysięczne. W systemie dwójkowym liczą połówki, ćwiartki i ósemki, a każda cyfra to 0 albo 1. Ponieważ $0,625$ to $\frac{1}{2} + \frac{1}{8}$, w systemie dwójkowym zapisuje się ją jako 0,101: dokładnie te trzy cyfry, które przed chwilą zapisales.

To nie przypadek. Podwojenie liczby dwójkowej przesuwają każdą cyfrę o jedno miejsce w lewo, tak jak mnożenie przez dziesięć przesuwają cyfry dziesiętne, a odjęcie 1 usuwa cyfrę, która przekroczyła przecinek. Pierwsza cyfra po przecinku mówi, w której połowie leży liczba. Odwzorowanie podwajające odczytuje więc cyfry dwójkowe swojego punktu startowego, po jednej na krok: cała jego przyszłość jest z góry zapisana w coraz dalszych cyfrach teraźniejszości. Poniższy program sprawdza to dla dwóch punktów startowych z tego rozdziału.

code/ch06/digits.py

```

16 def binary_digits(x: float, n: int) -> str:
17     """The first n binary digits of x after the point.
18
19     x * 2**n moves the point n places right, int() drops what is left
20     over, and format(..., "b") writes the whole number in binary.
21     """
22     return format(int(x * 2**n), f"0{n}b")
23
24
25 def sides(x: float, n: int) -> str:
26     """Run the doubling map n steps: 1 in the upper half, 0 in the lower."""
27     out = ""
28     for _ in range(n):
29         out += "1" if x >= 0.5 else "0"
30         x = doubling(x)
31     return out

```

Listing 6.3: Cyfry dwójkowe policzone wprost i odczytane przez odwzorowanie podwajające

```

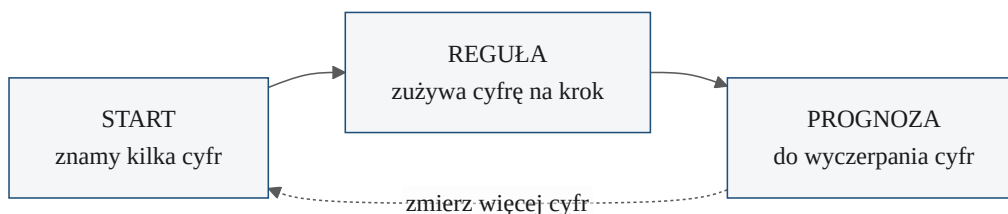
digits of 0.2000000000: 00110011001100110011001100110011
doubling map sides: 00110011001100110011001100110011
digits of 0.2000000001: 001100110011001100110011001110100001
doubling map sides: 001100110011001100110011001110100001

```

Dla każdego punktu startowego cyfry i zapis odwzorowania podwajającego – górna czy dolna połowa – to ten sam ciąg. Oba punkty startowe mają wspólne pierwsze 32 cyfry dwójkowe, więc odwzorowanie trzyma je po tej samej stronie przez dokładnie 32 kroki, a potem zaczyna odczytywać cyfry, którymi się różnią.

To jest wrażliwość na warunki początkowe bez żadnych tajemnic. Pomiar podaje ci

pierwsze tyle a tyle cyfr stanu; reguła zużywa jedną cyfrę na krok; gdy zmierzone cyfry się wyczerpią, prognoza zależy od cyfr, których nikt nie zmierzył. Dziesięć to trochę więcej niż $2 \times 2 \times 2 = 8$, więc jedna cyfra dziesiętna niesie trochę więcej niż trzy dwójkowe, a dziesięć miejsc po przecinku – trochę ponad trzydzieści. Komputer też przechowuje tylko tyle a tyle cyfr dwójkowych każdej liczby; co to robi z obliczeniami chaotycznymi, jest tematem rozdziału 16.



Rysunek 6.3: Dlaczego prognoza chaotyczna się kończy. Punkt startowy znamy do kilku cyfr, a reguła zużywa je krok po kroku; gdy się wyczerpią, prognoza opiera się na cyfrach, których nikt nie zmierzył.

Laboratorium 6.2 – Znane cyfry, przewidziane kroki

Napisz `shared_digits`, które liczy, ile początkowych cyfr dwójkowych mają wspólnych dwie liczby, oraz `steps_together`, które uruchamia odwzorowanie podwajające na obu i liczy kroki, zanim znajdą się po różnych stronach połowy. Test sprawdza, że obie liczby zgadzają się dla wielu par, a dla dwóch punktów startowych z tego rozdziału obie wynoszą 32.

Plik startowy: `code/labs/ch06/l06_02_digits.py`

Test: `code/labs/ch06/test_l06_02_digits.py`

Uruchom: `cd code && uv run pytest -k l06_02_digits`

6.4 Ile jest warta cyfra

Odwzorowanie podwajające mówi, że każda poprawna cyfra dwójkowa punktu startowego daje o jeden krok prognozy więcej. Odwzorowanie logistyczne przy $r = 4$ nie jest odwzorowaniem podwajającym, ale widziałeś, że jego różnica średnio się podwaja, więc ta sama wymiana powinna dla niego obowiązywać, w przybliżeniu. To da się zmierzyć.

Zatrzymaj się i pomyśl 6.5

Bliźniaki, których punkty startowe zgadzały się do dziesięciu miejsc po przecinku, pozostawały bliżej niż o 0,1 przez mniej więcej trzydzieści kroków. Przyrząd sto razy dokładniejszy dałby ci dwanaście miejsc po przecinku. Mniej więcej ile kroków wtedy: około trzydziestu siedmiu czy około sześćdziesięciu?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około trzydziestu siedmiu. Sto to mniej więcej $2^7 = 128$, więc błąd sto razy mniejszy potrzebuje mniej więcej siedmiu podwojeń więcej, żeby nadrobić różnicę.

Dwa razy więcej cyfr nie daje dwa razy dłuższej prognozy. Poniższy program sprawdza tę arytmetykę na tysiącu punktów startowych rozłożonych równo między 0 a 1, bo każdy pojedynczy punkt może mieć szczęście albo pecha: dla każdej wielkości błędu liczy kroki, zanim prognoza pomyli się o więcej niż 0,1, i uśrednia.

code/ch06/worth.py

```

15 TOL = 0.1                                     # "no longer useful"
16 STARTS = [(i + 0.5) / 1000 for i in range(1000)]
17 ERRORS = {2: 1e-2, 4: 1e-4, 6: 1e-6, 8: 1e-8, 10: 1e-10, 12: 1e-12}
18
19
20 def average_steps(error: float) -> float:
21     """Steps a forecast stays within TOL, averaged over every start."""
22     total = 0
23     for x0 in STARTS:
24         total += steps_until_apart(rule, x0, error, TOL)
25     return total / len(STARTS)

```

Listing 6.4: Kroki, które przetrwa prognoza, uśrednione po tysiącu punktów startowych

```

start known to 2 decimal places: 4.1 steps
start known to 4 decimal places: 10.8 steps
start known to 6 decimal places: 17.4 steps
start known to 8 decimal places: 24.1 steps
start known to 10 decimal places: 30.6 steps
start known to 12 decimal places: 37.3 steps

```

Każde dwa miejsca po przecinku dodają mniej więcej siedem kroków, więc każde jedno dodaje około 3,3, i to jest arytmetyka odwzorowania podwajającego: cyfra dziesiętna jest warta trochę więcej niż trzy dwójkowe. Przeczytaj tabelę w drugą stronę, a odpowie na pytanie, które zadaje każdy, kto prognozuje. Zmierz punkt startowy do dwóch miejsc po przecinku, a prognoza wytrzyma około 4,1 kroku; do dziesięciu – około 30,6; heroiczna dokładność, dwanaście miejsc, daje 37,3. Rozdział 7 zamienia tę tabelę w jedną liczbę dla dowolnego układu chaotycznego i we wzór na to, jak daleko naprzód da się go przewidzieć. Tu wystarczy, że umiesz to zmierzyć.

6.5 Wydruk Lorenza

W 1961 roku meteorolog Edward Lorenz z Massachusetts Institute of Technology uruchamiał na komputerze mały model pogody. Chcąc jeszcze raz przyjrzeć się jednemu fragmentowi obliczeń, wznowił je od środka, wpisując liczby z wydruku wcześniejszego przebiegu. Wydruk pokazywał trzy miejsca po przecinku, 0,506, podczas gdy maszyna przechowywała sześć, 0,506127. Nowy przebieg przez jakiś czas szedł za starym, potem się od niego oddalił, aż obie prognozy przestały mieć ze sobą cokolwiek wspólnego.

Różnica między punktami startowymi, mniej więcej jedna część na cztery tysiące, była błędem tego rodzaju, o którym zakładał, że nie ma znaczenia.

Odzworowanie logistyczne potrafi odtworzyć ten przypadek. Uruchom je z pełnej liczby, jeszcze raz z pełnej liczby i raz z liczby z wydruku.

Zatrzymaj się i pomyśl 6.6

Przypuśćmy, że Lorenz wpisałby wszystkie sześć miejsc po przecinku, 0,506127, dokładnie tę liczbę, którą przechowywała maszyna. Czy jego drugi przebieg zgodziłby się z pierwszym?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tak, cyfra po cyfrze i krok po kroku. W regule nie ma przypadku: ten sam punkt startowy zawsze daje tę samą przyszłość.

code/ch06/printout.py

```
14 FULL = 0.506127    # what the machine held
15 PRINTED = 0.506    # what the printout showed: three decimal places
16
17 first = orbit(rule, FULL, 20)
18 again = orbit(rule, FULL, 20)      # the same start, typed in full
19 rounded = orbit(rule, PRINTED, 20) # the start copied off the printout
```

Listing 6.5: Wydruk Lorenza odtworzony na odzworowaniu logistycznym

step	first	again	rounded
0	0.506127	0.506127	0.506000
2	0.000601	0.000601	0.000576
4	0.009580	0.009580	0.009188
6	0.146050	0.146050	0.140356
8	0.999995	0.999995	0.998792
10	0.000081	0.000081	0.019207
12	0.001290	0.001290	0.278701
14	0.020504	0.020504	0.630075
16	0.295517	0.295517	0.252392
18	0.557118	0.557118	0.740388
20	0.051519	0.051519	0.710870

again == first, every digit of every step: True

Kolumna again to powtórzony pierwszy przebieg, a program porównuje każdą jej liczbę z pierwszym przebiegiem i wypisuje True. Kolumna rounded zaczyna 0,000127 dalej i różni się o więcej niż 0,1 w kroku 12, wcześniej niż bliźniaki, bo jej błąd był na starcie znacznie większy.

Pułapka

„Jeśli uruchomię model chaotyczny jeszcze raz, dostanę coś innego.” Nie z tego samego punktu startowego. Reguła chaotyczna to maszyna jak każda inna: to samo wejście daje to samo wyjście, co do cyfry, jak pokazuje kolumna again. Przyszłość zmienia zmieniony punkt startowy, choćby najmniejszy. Jeśli ponownie uruchamiasz model komputerowy i dostajesz inną odpowiedź, coś się zmieniło – punkt startowy, reguła albo arytmetyka – a rozdział 16 jest o tej ostatniej.

W 1972 roku Lorenz wygłosił wykład, którego tytuł pytał, czy trzepot skrzydeł motyla w Brazylii może wywołać tornado w Teksasie. Sformułowanie się przyjęło i od tamtej pory wrażliwość na warunki początkowe nazywa się *efektem motyla*. Jego własne równania i kształt, który rysują, należą do rozdziału 9.

Zatrzymaj się i pomyśl 6.7

Tornado uderza w Teksas. Przypuśćmy, że możesz uruchomić atmosferę jeszcze raz od chwili sprzed dwóch tygodni, usuwając jeden trzepot skrzydeł jednego motyla. Czy tornado by zniknęło?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Może tak, może nie, i nikt nie potrafiłby powiedzieć tego z góry. Ponowny przebieg to inna atmosfera: może nie mieć tornada, mieć tornado gdzie indziej albo mimo wszystko mieć je w Teksasie.

Pułapka

„Motyl wywołuje tornado.” Pytanie Lorenza dotyczyło przewidywania, nie siły. Energia burzy pochodzi z atmosfery, nie od owada. Usunąć trzepot, a pogoda pójdzie inną ścieżką, na której tornado może być albo nie; każdy inny trzepot, podmuch i oddech robi to samo, więc żaden z nich z osobna nie jest przyczyną. W odtworzeniu powyżej trzy odrzucone cyfry nie sprawiły, że zaokrąglony przebieg stał się tym, czym się stał: sprawiła to reguła działająca na cały punkt startowy. Motyl pokazuje, że pogoda za dwa tygodnie zależy od szczegółów, których nikt nie potrafi zmierzyć.

6.6 Deterministyczny, a mimo to nieprzewidywalny

W tym rozdziale dwa słowa wykonywały dwie różne prace i trzeba je nazwać. Reguła jest *deterministyczna*, gdy ten sam punkt startowy zawsze daje tę samą przyszłość: żadnych kości, żadnego przypadku. Układ jest *przewidywalny* w pewnym zakresie, gdy punkt startowy, który naprawdę da się zmierzyć, wyznacza jego przyszłość w tym zakresie. Herbata jest i taka, i taka. Odwzorowanie logistyczne przy $r = 4$ jest deterministyczne – sprawdziliśmy to powyżej, cyfra po cyfrze – a mimo to nieprzewidywalne dalej niż na kilkadziesiąt kroków, bo żaden zmierzony punkt startowy nie ma dość cyfr.

Zatrzymaj się i pomyśl 6.8

Odwzorowanie logistyczne przy $r = 4$ skacze bez żadnego wzoru, który oko mogłoby dostrzec, i nikt nie potrafi go długo przewidywać. Czy jego wyniki są losowe?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Każda liczba wynika dokładnie z poprzedniej i możesz policzyć całą listę jeszcze raz, odzyskując każdą cyfrę. Nie da się zdobyć punktu startowego, a nie reguły.

Pułapka

„Chaos oznacza losowość.” Chaos jest przeciwieństwem przypadku. Układ chaotyczny w każdym kroku dokładnie trzyma się swojej reguły; to, co wygląda na losowość, to reguła odczytująca, krok po kroku, cyfry punktu startowego, których nikt nie zna. Ciąg losowy w ogóle nie ma reguły. Ciąg chaotyczny ma regułę, której nie da się wyprzedzić. Jak odróżnić jedno od drugiego, gdy masz tylko dane, to pytanie rozdziału 15.

Gdzie to spotkasz

Rzut monetą rozstrzyga spory właśnie dlatego, że nikt nie potrafi go przewidzieć, a przecież wirująca moneta podlega prawom Newtona tak samo jak kamień. Statystyk Persi Diaconis i jego współpracownicy zlecili budowę maszyny, która za każdym razem wyrzuca monetę tak samo, i moneta za każdym razem upada tą samą stroną do góry. Kciuk nie potrafi powtórzyć rzutu aż tak dokładnie, więc wynik prawdziwego rzutu zależy od różnic zbyt małych, żeby nad nimi zapanować. Moneta jest deterministyczna, a w twojej dłoni – nieprzewidywalna.

Demon Laplace’a z rozdziału 1 potrzebował dokładnie znać regułę i teraźniejszość. Ten rozdział wycenił słowo *dokładnie*. W przypadku odwzorowania logistycznego każdy krok naprzód kosztuje jedną cyfrę dwójkową teraźniejszości więcej, więc widzenie bez końca wymaga nieskończenie wielu cyfr, a przyrząd, który jest zaledwie znakomity, widzi na kilkadziesiąt kroków, a potem nic. Mechanizm zegara jest nietknięty. To tarczy zegara nie da się odczytać dość dokładnie.

Podsumowanie

- Dwa przebiegi odwzorowania logistycznego przy $r = 4$, które startują o jedną dziesięciomiliardową od siebie, zgadzają się co do jednej milionowej przez 16 kroków i różnią się o więcej niż 0,1 od kroku 30: policzone dokładnie, tak samo na każdym komputerze.
- Ich różnica w przybliżeniu **podwaja się w każdym kroku**, co jest wykładniczym wzrostem błędu, aż osiągnie rozmiar całego przedziału. To jest

wrażliwość na warunki początkowe: w skali logarytmicznej prosta w górę, a potem sufit.

- **Odwzorowanie podwajające** odczytuje cyfry dwójkowe swojego punktu startowego, po jednej na krok, więc jego przyszłość jest zapisana w cyfrach teraźniejszości, których nie ma żaden pomiar.
- Każda poprawna cyfra dziesiętna punktu startowego daje tylko około 3,3 kroku prognozy więcej: dwa razy więcej cyfr to nie dwa razy dłuższa prognoza.
- Zaokrąglony restart Lorenza odtworzony na odwzorowaniu logistycznym: pełny punkt startowy daje identyczny przebieg, zaokrąglony się rozchodzi. **Efekt motyla** dotyczy przewidywania, nie przyczyn.
- Reguła może być **deterministyczna**, a mimo to **nieprzewidywalna**: chaos to nie przypadek, a ponowne uruchomienie z tego samego punktu startowego daje tę samą przyszłość.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

uruchomić jedną regułę z dwóch bliskich punktów startowych i odczytać oba przebiegi obok siebie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zmierzyć różnicę między dwoma przebiegami krok po kroku i odczytać ją w skali logarytmicznej	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co znaczy wrażliwość na warunki początkowe, w kategoriach czegoś, co da się zmierzyć	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
wyjaśnić za pomocą odwzorowania podwajającego, dlaczego prognoza się kończy	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, dla danego błędu punktu startowego, mniej więcej przez ile kroków przewidywanie pozostaje użyteczne	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
odróżnić determinizm od przewidywalności i wiernie opowiedzieć o odkryciu Lorenza	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dwa przebiegi odwzorowania logistycznego przy $r = 4$ startują z 0,2 i 0,2000000001. Mniej więcej kiedy po raz pierwszy różnią się o więcej niż 0,1?
2. Różnica równa jednej milionowej podwaja się w każdym kroku. Po ilu krokach jest większa niż 1?

3. Dlaczego różnica między dwoma przebiegami odwzorowania logistycznego przestaje rosnąć?
4. Zapisz 0,75 w systemie dwójkowym i zapisz, w której połowie leży przez trzy kroki odwzorowania podwajającego.
5. Prognoza z punktu startowego znanego do dziesięciu miejsc po przecinku wytrzymała około 30,6 kroku. Mniej więcej ile przy czternastu miejscach po przecinku?
6. Lorenz wznowił obliczenia od 0,506 zamiast od 0,506127. Co zobaczyłby, gdyby wpisał wszystkie sześć cyfr, i dlaczego?
7. Czy odwzorowanie logistyczne przy $r = 4$ jest losowe? Czym różni się „deterministyczny” od „przewidywalnego”?

Liczba na chaos: wykładnik Lapunowa

Rozdział 6 policzył dwie rzeczy o odwzorowaniu logistycznym przy $r = 4$: odstęp między dwoma bliskimi przebiegami mniej więcej podwaja się w każdym kroku, a każde poprawne miejsce po przecinku stanu początkowego daje tylko kilka kroków prognozy więcej. Oba fakty znaleziono przez liczenie, dla jednego odwzorowania. Pytanie, które za nimi stoi, zadasz każdemu układowi w tej książce: jak szybko rosną małe błędy i jak daleko naprzód pozwala to widzieć?

Ten rozdział zamienia liczenie w jedną liczbę, wykładnik Lapunowa, którą da się zmierzyć dla każdego układu, a tę liczbę w horyzont: liczbę kroków, na które przewidywanie jest dobre. Podczas lektury trzymaj się tego pytania: gdybyś zmierzył stan początkowy układu chaotycznego tysiąc razy dokładniej, o ile dalej mógłbyś go przewidywać?

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- ustalić, jak bardzo jeden krok reguły rozciąga mały błąd, i sprawdzić to, uruchamiając dwie kopie programu
- używać logarytmu naturalnego, żeby zamienić łańcuch mnożeń w sumę i z powrotem
- obliczyć wykładnik Lapunowa odwzorowania logistycznego z orbity i odczytać jego znak
- porównać obliczony wykładnik z wartością znaną dokładnie
- zamienić wykładnik, błąd pomiaru i tolerancję w horyzont przewidywania
- powiedzieć, co daje lepszy pomiar, licząc w krokach

7.1 Krok po kroku

Rozdział 6 zauważył, że odstęp między dwoma przebiegami rośnie nierówno: niektóre kroki prawie go czterokrotnie zwiększają, kilka go zmniejsza, a podwaja się dopiero średnio. Zaczynij od tej nierówności. Uruchom odwzorowanie logistyczne przy $r = 4$ od $x = 0,3$ i od wartości o 10^{-10} wyższej i patrz, co każdy pojedynczy krok robi z odstępem: podziel odstęp po kroku przez odstęp przed nim. Ten iloraz nazwijmy *rozciągnięciem kroku*.

code/ch07/stretch.py

```

11 def step(r: float, x: float) -> float:
12     """One step of the logistic map."""
13     return r * x * (1.0 - x)
14
15
16 def stretch(r: float, x: float) -> float:
17     """How much one step multiplies a tiny nudge at x: the slope there."""
18     return r * (1.0 - 2.0 * x)

```

Listing 7.1: Odwzorowanie logistyczne i jego rozciągnięcie

Funkcja `step` to odwzorowanie logistyczne z rozdziału 5. Funkcja `stretch` to jego nachylenie, $r(1 - 2x)$. Rozdział 3 mierzył nachylenie, lekko przesuwając x i patrząc, jak daleko przesunie się wynik; dla tej reguły da się je zapisać dokładnie, a pierwsze laboratorium sprawdza ten wzór przesuwaniem. Reszta programu mierzy rozciągnięcie pierwszych ośmiu kroków na podstawie dwóch przebiegów, obok nachylenia w miejscu, w którym jest pierwszy przebieg:

code/ch07/stretch.py

```

36 r, a, b = 4.0, 0.3, 0.3 + 1e-10
37 print("step      x      gap grew by      slope there")
38 for n in range(1, 9):
39     grew = abs(step(r, b) - step(r, a)) / abs(b - a)
40     print(f"{n:4d}{a:9.3f}{grew:15.3f}{abs(stretch(r, a)):15.3f}")
41     a, b = step(r, a), step(r, b)

```

Listing 7.2: Pomiar rozciągnięcia ośmiu kroków

step	x	gap grew by	slope there
1	0.300	1.600	1.600
2	0.840	2.720	2.720
3	0.538	0.301	0.301
4	0.994	3.955	3.955
5	0.022	3.820	3.820
6	0.088	3.296	3.296
7	0.321	1.433	1.433
8	0.872	2.973	2.973

Dwie prawe kolumny się zgadzają. W jednym kroku mały odstęp jest mnożony przez nachylenie reguły w miejscu, w którym jesteś, więc rozciągnięcie można odczytać z reguły, zamiast mierzyć je drugim przebiegiem. Ujemne nachylenie dodatkowo przerzuca odstęp na drugą stronę, co nie zmienia jego wielkości; tabela podaje wielkości.

A rozciągnięcia wcale nie przypominają stałej 2. Krok 3 zmniejsza odstęp do mniej niż jednej trzeciej, a kroki 4 i 5 prawie go czterokrotnie zwiększają. W kroku 3 przebieg jest blisko 0,5, gdzie parabola jest płaska na szczycie; w krokach 4 i 5 jest blisko 1 i blisko 0, gdzie parabola jest stroma. Chaos nie rozciąga równomiernie: w jednym miejscu mocno, w innym łagodnie, a czasem ściska.

Zatrzymaj się i pomyśl 7.1

Załóżmy, że jeden krok mnoży odstęp przez 4, a następny przez $\frac{1}{4}$. Co stało się z odstępem w ciągu tych dwóch kroków? I jaki jeden czynnik, użyty w obu krokach, dałby to samo?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nic się nie stało: $4 \times \frac{1}{4} = 1$, i odstęp wrócił tam, gdzie był. Tym jednym czynnikiem jest 1, a nie 2,125, które dostajesz, uśredniając 4 i $\frac{1}{4}$.

Wzrost się składa: każdy krok mnoży to, co zostawił poprzedni. Typowe rozciągnięcie nie jest więc zwykłą średnią rozciągnięć. To czynnik, który użyty w każdym kroku daje ten sam wynik łączny.

Pułapka

„**Żeby poznać typowe rozciągnięcie na krok, uśrednij rozciągnięcia.**” Sprawdź to na tabeli. Średnia ośmiu rozciągnięć wynosi 2,51, a odstęp pomnożony przez nią osiem razy urósłby mniej więcej 1587-krotnie. Pomnóż zamiast tego osiem prawdziwych rozciągnięć, a okaże się, że odstęp urósł 278-krotnie. Kiedy coś się mnoży, dodawanie odpowiada na pytanie, którego nikt nie zadał; średnia, której szukasz, też musi mnożyć.

7.2 Logarytm zamienia mnożenie w dodawanie

Rozdział 2 użył logarytmu, żeby odpowiedzieć na pytanie „ile kroków, zanim...?”. Ten rozdział potrzebuje jednego konkretnego logarytmu, logarytmu *naturalnego*, zapisywanego $\ln$, i jednej rzeczy, którą on robi. Oto kilka jego wartości:

liczba	$\frac{1}{2}$	1	2	4	10	e
jej logarytm naturalny	-0,69	0	0,69	1,39	2,30	1

Spójrz na 2 i 4. Ponieważ $4 = 2 \times 2$, logarytm czwórki jest dwa razy większy od logarytmu 2. To wszystko, czego ten rozdział potrzebuje:

$$\ln(a \times b) = \ln a + \ln b.$$

Słowami: logarytm iloczynu jest sumą logarytmów, więc mnożenie dwóch liczb zamienia się w dodawanie ich logarytmów. Liczby większe od 1 mają logarytmy dodatnie, a liczby między 0 a 1 ujemne: $\frac{1}{2} \times 2 = 1$, więc $\ln \frac{1}{2}$ i $\ln 2$ sumują się do $\ln 1$, czyli do 0. Ostatnia kolumna zawiera e , mniej więcej 2,718, liczbę, której logarytm naturalny wynosi dokładnie 1; odgrywa ona rolę, jaką 10 odgrywa dla logarytmu dziesiętnego. W Pythonie logarytm naturalny to `math.log`, a `math.log(2)` zaczyna się od 0,6931.

Zatrzymaj się i pomyśl 7.2

Korzystając tylko z $\ln 2 \approx 0,69$, oblicz $\ln 8$. A ile wynosi $\ln \frac{1}{4}$?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $8 = 2 \times 2 \times 2$, więc $\ln 8 = \ln 2 + \ln 2 + \ln 2$, czyli mniej więcej 2,08. A $\frac{1}{4} \times 4 = 1$, więc $\ln \frac{1}{4}$ wynosi $-1,39$.

Jeśli napisałeś 2,07, zrobiłeś to dobrze: zaokrąglenie $\ln 2$ policzyłeś trzy razy.

Tak logarytm naprawia średnią. Jeśli odstęp jest w kroku mnożony przez g , jego logarytm rośnie o $\ln g$: *czynnik* na krok staje się *tempem* na krok, a tempa się dodają, więc można je uśredniać zwyczajnie. Dla dwóch kroków z pytania tempa $\ln 4$ i $\ln \frac{1}{4}$ sumują się do 0, średnie tempo wynosi 0, a czynnik, którego logarytm to 0, wynosi 1: właściwa odpowiedź, znaleziona dodawaniem.

Żeby wrócić od tempa do czynnika, potrzebujesz liczby, której logarytmem jest to tempo; zapisuje się ją e^λ , „ e do potęgi λ ”; λ to grecka litera lambda i oznacza tempo. Zatem $e^0 = 1$, $e^1 = e$ i $e^{\ln 2} = 2$. Teraz osiem kroków z tabeli. Logarytmy ich rozciągnięć mają średnią 0,70, a $e^{0,70}$ to mniej więcej 2,02: średnio odstęp rósł o czynnik około 2 na krok, a 2,02 pomnożone przez siebie osiem razy daje 278, które wyszło w tabeli. „Mniej więcej się podwaja” stało się pomiarem.

7.3 Wykładnik Lapunowa

Osiem kroków to próbka. Prześledź zamiast tego orbitę przez sto tysięcy kroków, weź logarytm naturalny wielkości rozciągnięcia w każdym punkcie, który odwiedza, i uśrednij. Wynik to *wykładnik Lapunowa*, nazwany na cześć rosyjskiego matematyka Aleksandra Lapunowa, którego praca o stabilności ruchu ukazała się w 1892 roku. Jeśli $s_1, s_2, \dots, s_N$ to rozciągnięcia wzdłuż orbity,

$$\lambda = \frac{\ln |s_1| + \ln |s_2| + \dots + \ln |s_N|}{N}.$$

Słowami: λ to średnia logarytmów rozciągnięć. Kreski oznaczają „wielkość”, bo przerzucenie na drugą stronę nie czyni odstępu ani większym, ani mniejszym.

Czytaj λ jako tempo. W ciągu N kroków mały błąd jest mnożony przez mniej więcej $e^{\lambda N}$, więc znak mówi, z jakim układem masz do czynienia:

- **λ powyżej zera:** błędy rosną wykładniczo, a układ jest chaotyczny;
- **λ poniżej zera:** błędy maleją, a układ zapomina o swoich zaburzeniach;
- **λ równe zero:** granica, na której błędy średnio ani nie rosną, ani nie maleją.

Zatrzymaj się i pomyśl 7.3

Przy $r = 4$ odstęp mniej więcej podwaja się w każdym kroku. Jakiej wartości λ się spodziewasz?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniej więcej $\ln 2$, czyli 0,69: odstęp mnożony w każdym kroku przez 2 rośnie w tempie $\ln 2$ na krok.

Zatrzymaj się i pomyśl 7.4

Przy $r = 2,8$ orbita zatrzymuje się w punkcie stałym $1 - 1/r$ z rozdziału 5. Tam nachylenie $r(1 - 2x)$ wynosi $2 - r = -0,8$, a $\ln 0,8$ to mniej więcej $-0,22$. Ile wynosi λ ?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniej więcej $-0,22$. Kiedy orbita siedzi w punkcie stałym, każdy składnik średniej to $\ln 0,8$, więc średnia też. Jest ujemna: zaburzenie maleje w każdym kroku o jedną piątą.

code/ch07/exponent.py

```

17 def exponent(r: float, x0: float = 0.3, steps: int = 100_000,
18             burn: int = 1000) -> float:
19     """The average of  $\ln|\text{stretch}|$  along the orbit of the logistic map."""
20     x = x0
21     for _ in range(burn):          # let the orbit settle first
22         x = step(r, x)
23     total = 0.0
24     for _ in range(steps):
25         total += math.log(abs(stretch(r, x)))
26         x = step(r, x)
27     return total / steps

```

Listing 7.3: Wykładnik Lapunowa przez uśrednianie

Pierwsza pętla wykonuje tysiąc kroków odwzorowania i je wyrzuca, żeby orbita się ustaliła, zanim cokolwiek zostanie uśrednione. Druga dodaje logarytm wielkości rozciągnięcia w każdym punkcie, a potem robi krok; ostatni wiersz dzieli. To jest wzór powyżej i nic więcej. Uruchom go przy czterech ustawieniach pokrętki r :

code/ch07/exponent.py

```

33 for r in (2.8, 3.2, 3.5, 4.0):
34     lam = exponent(r)
35     verdict = "errors grow" if lam > 0 else "errors shrink"
36     print(f"r = {r}: exponent {lam:+.2f} {verdict}")
37     print(f"exact value at r = 4: ln 2 = {math.log(2):.4f}")

```

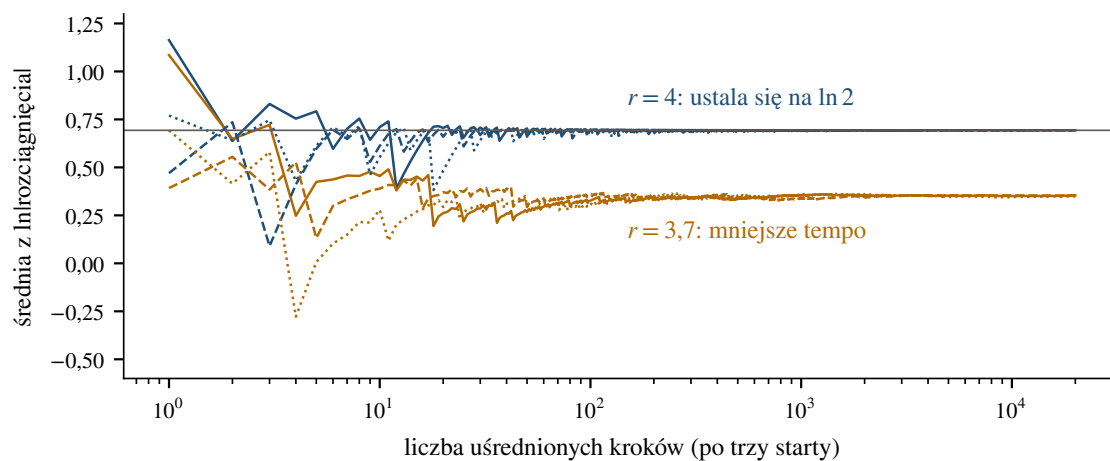
Listing 7.4: Wykładnik przy czterech ustawieniach r

```

r = 2.8: exponent -0.22  errors shrink
r = 3.2: exponent -0.92  errors shrink
r = 3.5: exponent -0.87  errors shrink
r = 4.0: exponent +0.69  errors grow
exact value at r = 4: ln 2 = 0.6931

```

Oba przewidywania się sprawdzają. Przy $r = 2,8$ program znajduje $-0,22$. Przy $r = 4$ znajduje $0,69$, a dokładna wartość to $\ln 2 = 0,6931$: średnia stu tysięcy bardzo różnych logarytmów zgadza się z nią z dokładnością do $0,001$. Przy $r = 3,2$, gdzie orbita przeskakuje między dwiema wartościami, i przy $r = 3,5$, gdzie krąży między czterema, wykładnik też jest ujemny: $-0,92$ i $-0,87$. Cykl, tak jak punkt stały, zapomina o swoich zaburzeniach.



Rysunek 7.1: Średnia bieżąca logarytmu rozciągłości wzdłuż sześciu orbit: trzech startów przy $r = 4$ i trzech przy $r = 3,7$. Każda waha się przez kilkaset kroków, a potem się ustala, a trzy starty przy każdym ustawieniu ustalają się na tej samej liczbie.

Czego tu nie dowodzimy

Dwie rzeczy są tu używane, ale nie dowodzone. Pierwsza: że średnia ustala się na tej samej liczbie przy prawie każdym starcie. To twierdzenie, a „prawie każdym” ma znaczenie: start dokładnie w 0 zostaje w 0 na zawsze i daje średnią $\ln 4$. Przebieg na komputerze może to twierdzenie zilustrować, jak robi to rysunek, ale nie może go dowieść. Druga: dlaczego $r = 4$ daje dokładnie $\ln 2$. Zapisz x jako $\sin^2 \theta$ dla pewnego kąta θ , a jedna linijka szkolnej trygonometrii zamieni $4x(1-x)$ w $\sin^2(2\theta)$: przy $r = 4$ odwzorowanie to przebrane podwajanie kąta, czyli odwzorowanie podwajające z rozdziału 6, z kątem liczonym w półobrotach. Podwajanie rozciąga każdy błąd kąta dokładnie 2 razy, a zmiana przebrania nie zmienia średniego tempa.

Laboratorium 7.1 – Wykładnik z orbity

Napisz `stretch`, czyli nachylenie $r(1 - 2x)$, oraz `exponent`, która pozwala orbicie się ustalić, a potem uśrednia wzdłuż niej logarytm wielkości rozciągnięcia. Testy sprawdzają nachylenie przesuwaniem, tak jak robił to rozdział 3, i porównują twój wykładnik z trzema wartościami znanymi dokładnie: $\ln 2$ przy $r = 4$, $\ln 0,8$ przy $r = 2,8$ i cyklem dwuokresowym przy $r = 3,2$.

Plik startowy: `code/labs/ch07/l07_01_exponent.py`

Test: `code/labs/ch07/test_l07_01_exponent.py`

Uruchom: `cd code && uv run pytest -k l07_01_exponent`

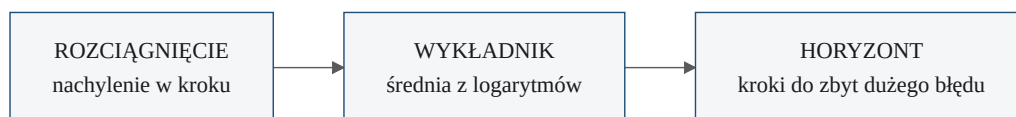
7.4 Jak daleko naprzód: horyzont przewidywania

Po n krokach mały błąd o wielkości δ (grecka litera delta) urasta do mniej więcej $\delta e^{\lambda n}$. Przestajesz ufać przewidywaniu, gdy jego błąd osiąga tolerancję T , największy błąd, z którym możesz żyć. Horyzont to więc liczba kroków, przy której $\delta e^{\lambda n} = T$. Zlogarytmuj obie strony: iloczyn staje się sumą, a logarytm $e^{\lambda n}$ to λn , bo e ma logarytm 1 i jest mnożone λn razy. Zostaje $\ln \delta + \lambda n = \ln T$, a stąd

$$n \approx \frac{1}{\lambda} \ln \frac{T}{\delta}.$$

Słowami: horyzont to logarytm tego, ile razy błąd może urosnąć, podzielony przez tempo, w jakim rośnie.

Liczba $1/\lambda$ nazywa się *czasem Lapunowa*: to liczba kroków, w ciągu których błędy rosną e razy. Przy $r = 4$ wynosi $1/\ln 2$, mniej więcej 1,44 kroku, a wzór mówi, że horyzont to czas Lapunowa pomnożony przez $\ln(T/\delta)$.



Rysunek 7.2: Od reguły do horyzontu. Nachylenie daje rozciągnięcie w każdym kroku, średnia jego logarytmu daje wykładnik, a wykładnik razem z błędem i tolerancją daje horyzont.

Zatrzymaj się i pomyśl 7.5

Założmy, że model prognozy ma czas Lapunowa równy dwóm dniom, a jego prognozy są dobre przez sześć. Nowa sieć przyrządów zmniejsza błąd jego stanu początkowego dziesięciokrotnie. Na jak długo prognozy są dobre teraz: na mniej więcej sześćdziesiąt dni, na mniej więcej dwanaście, czy krócej?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniej więcej 10,6 dnia. Dziesięciokrotnie mniejszy błąd dodaje $\ln 10$ do $\ln(T/\delta)$, więc dodaje $\ln 10$ czasów Lapunowa: $2 \times 2,30$, czyli mniej więcej 4,6 dnia.

Pułapka

„Dziesięć razy lepsze dane dają dziesięć razy dłuższe prognozy.” Dają stałą liczbę dodatkowych kroków, $\ln 10/\lambda$, bez względu na to, jaki był wcześniej błąd. Przy $r = 4$ to 3,3, liczba, którą rozdział 6 policzył dla każdego miejsca po przecinku, teraz wyprowadzona i prawdziwa dla każdego układu chaotycznego z jego własnym λ . Horyzont rośnie z logarytmem twojej dokładności, a nie z samą dokładnością: dwudzieste poprawne miejsce po przecinku daje nie więcej niż pierwsze.

code/ch07/horizon.py

```
16 def horizon(lam: float, delta: float, tol: float) -> float:
17     """Steps until an error delta grows to tol, growing at lam per step."""
18     return math.log(tol / delta) / lam
```

Listing 7.5: Wzór na horyzont

Program zestawia ten wiersz z liczeniem, które wykonał rozdział 6: funkcja counts śledzi z tysiąca startów dwa przebiegi oddalone o δ , aż różnią się o więcej niż T . W wydruku $1e-03$ to pythonowy zapis 10^{-3} , czyli jednej tysięcznej.

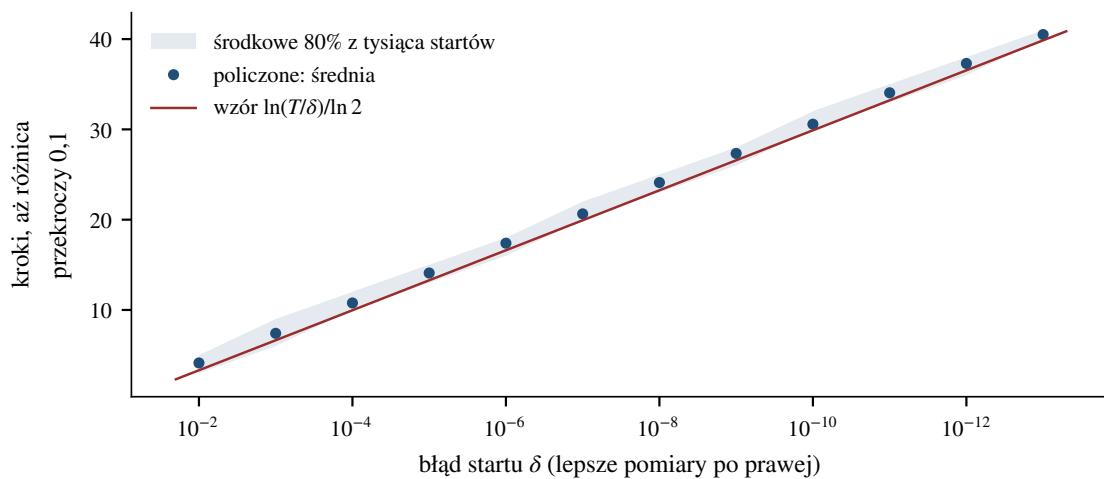
code/ch07/horizon.py

```
31 lam, tol = math.log(2), 0.1
32 print(" error formula counted: mean fastest slowest")
33 for delta in (1e-3, 1e-6, 1e-9, 1e-12):
34     c = counts(4.0, delta, tol)
35     mean = sum(c) / len(c)
36     print(f"{delta:7.0e}{horizon(lam, delta, tol):10.1f}")
37     f"{mean:16.1f}{min(c):10d}{max(c):10d}")
```

Listing 7.6: Wzór kontra liczenie, dla czterech błędów stanu początkowego

error	formula	counted: mean	fastest	slowest
1e-03	6.6	7.4	4	43
1e-06	16.6	17.4	13	24
1e-09	26.6	27.3	24	34
1e-12	36.5	37.3	33	44

Wzór i średnia z liczenia zgadzają się z dokładnością do kroku; liczenie wychodzi odrobinę później, bo odstęp sprawdza się tylko po pełnych krokach, a gdy przestaje być mały, rośnie wolniej. Dwie ostatnie kolumny są ostrzeżeniem: poszczególne starty się różnią, niektóre bardzo, bo horyzont, tak jak wykładnik, jest średnią. Ale spójrz w dół wierszy. Każda tysięczna poprawa dodaje te same dziesięć kroków; w pomiarze wiersze dzieli średnio 10,0 kroku, wobec $\ln 1000/\ln 2$ ze wzoru, czyli 6,91 podzielone przez 0,69: 10.



Rysunek 7.3: Horyzont w zależności od błędu stanu początkowego, w skali logarytmicznej, z lepszymi pomiarami po prawej. Liczenie (kropki i pasmo, w którym mieści się większość startów) idzie wzdłuż prostej ze wzoru: każdy czynnik dziesięć w dokładności daje tę samą liczbę dodatkowych kroków.

Zatrzymaj się i pomyśl 7.6

Komputer przechowuje liczbę z dokładnością do mniej więcej szesnastu cyfr znaczących. Załóżmy, że potrafisz tak dobrze zmierzyć stan początkowy odwzorowania przy $r = 4$, więc $\delta = 10^{-16}$. Przy $T = 0,1$ jak daleko naprzód mógłbyś mniej więcej przewidywać? I ilu miejsc po przecinku stanu początkowego wymagałoby tysiąc kroków?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniej więcej 50 kroków: piętnaście czynników dziesięć, po 3,3 kroku każdy. Tysiąc kroków wymagałoby stanu początkowego z dokładnością do 303 miejsc po przecinku; nawet sto kroków wymaga 32.

Pułapka

„Większy komputer w końcu uczyni każdy układ przewidywalnym.” Szybszy komputer nie zna stanu początkowego ani trochę lepiej, a horyzont zależy od stanu początkowego tylko przez logarytm jego błędu. Żeby podwoić horyzont, musisz podwoić liczbę poprawnych cyfr, a najdokładniejsze pomiary, jakich kiedykolwiek dokonano, za pomocą zegarów atomowych, sięgają mniej niż dwudziestu. Komputer w dodatku zaokrągla w każdym kroku, co jest nowym błędem, jego własnym; rozdział 16 mierzy, ile to kosztuje.

Dlatego prognozy pogody poprawiają się powoli. Satelity, gęstsze obserwacje i szybsze komputery wielokrotnie zmniejszyły błąd zmierzonego stanu atmosfery, a każdy taki czynnik kupił tę samą stałą porcję wyprzedzenia; zwykle przytacza się poprawę o mniej więcej jeden dzień użytecznej prognozy na dekadę. Atmosfera nie jest odwzorowaniem

logistycznym, a rozdział 14 mówi, co jest w niej inne, ale arytmetyka logarytmu stosuje się do niej tak samo.

Gdzie to spotkasz

Czasem Lapunowa astronomowie opisują, jak daleko naprzód da się śledzić orbitę. W 1988 roku Gerald Sussman i Jack Wisdom prześledzili Plutona na komputerze zbudowanym specjalnie w tym celu, stwierdzili, że jego ruch jest chaotyczny, i oszacowali jego czas Lapunowa na mniej więcej dwadzieścia milionów lat; historia samych planet należy do rozdziału 13. Sama ta liczba nie jest horyzontem. Horyzont to czas Lapunowa pomnożony przez $\ln(T/\delta)$, a ten logarytm wynosi tylko mniej więcej 23 nawet wtedy, gdy błąd stanu początkowego jest dziesięć miliardów razy mniejszy od tolerancji, więc horyzont to co najwyżej kilkadziesiąt czasów Lapunowa.

Laboratorium 7.2 – Jak daleko naprzód i jakim kosztem

Napisz horizon, czyli wzór z tego podrozdziału; extra_steps, czyli liczbę kroków zyskanych dzięki zmierzeniu stanu początkowego zadaną liczbę razy dokładniej; oraz digits_needed, czyli liczbę miejsc po przecinku, jakiej wymaga prognoza zadanej długości. Testy sprawdzają, że tysiąckrotna poprawa daje tę samą liczbę kroków bez względu na to, od jakiego błędu zaczynasz.

Plik startowy: code/labs/ch07/l07_02_horizon.py

Test: code/labs/ch07/test_l07_02_horizon.py

Uruchom: `cd code && uv run pytest -k l07_02_horizon`

7.5 Chaos przychodzi i odchodzi

Mając jedną liczbę na każde ustawienie pokrętki, możesz kręcić pokrętką i ją obserwować. Rozdział 5 znalazł punkt stały przy $r = 2,8$ i cykl dwuokresowy przy $r = 3,2$; między nimi, przy $r = 3$, punkt stały ustępuje.

Zatrzymaj się i pomyśl 7.7

W punkcie stałym nachylenie wynosi $2 - r$. Ile wynosi przy $r = 3$ i co to oznacza dla λ ?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $2 - 3 = -1$, a $\ln 1 = 0$, więc $\lambda = 0$: zaburzenie ani się nie rozciąga, ani nie kurczy, tylko przerzuca. Uruchomiony przy $r = 3$ program znajduje wartość odległą od zera o mniej niż 0,001.

Zero to granica i wskazuje miejsca, gdzie jeden rodzaj zachowania przechodzi w inny. Żeby takie miejsca znaleźć, policz wykładnik przy wielu ustawieniach r . Listing używa funkcji `lyapunov` z biblioteki, która jest pętlą z `exponent`, z regułą i jej rozciągnięciem przekazany jako argumenty, więc działa dla każdego odwzorowania:

code/ch07/sweep.py

```

14 print("    r    exponent")
15 for k in range(16):
16     r = (340 + 4 * k) / 100          # 3.40, 3.44, ..., 4.00
17     lam = lyapunov(logistic(r), logistic_slope(r), 0.3, 20_000)
18     verdict = "errors grow" if lam > 0 else "errors shrink"
19     print(f"{r:5.2f}{lam:+11.2f}  {verdict}")

```

Listing 7.7: Wykładnik przy szesnastu ustawieniach r

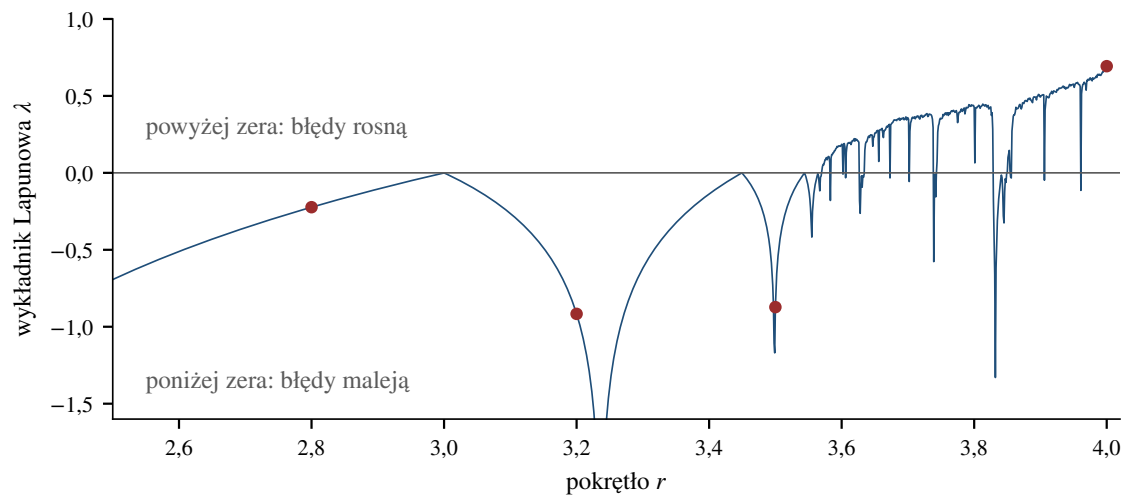
r	exponent	
3.40	-0.14	errors shrink
3.44	-0.02	errors shrink
3.48	-0.24	errors shrink
3.52	-0.19	errors shrink
3.56	-0.08	errors shrink
3.60	+0.18	errors grow
3.64	+0.23	errors grow
3.68	+0.34	errors grow
3.72	+0.36	errors grow
3.76	+0.39	errors grow
3.80	+0.43	errors grow
3.84	-0.04	errors shrink
3.88	+0.46	errors grow
3.92	+0.53	errors grow
3.96	+0.54	errors grow
4.00	+0.69	errors grow

Czytaj środkową kolumnę od góry. Jest ujemna aż do $r = 3,56$ i dodatnia od $r = 3,60$, więc chaos pojawia się gdzieś pomiędzy. Potem rośnie ku $\ln 2$ przy $r = 4$, z wyjątkiem $r = 3,84$, gdzie znów jest ujemna: to wąski zakres pokręta wewnątrz chaosu, w którym orbita znalazła cykl i znów zapomina o zaburzeniach. Tak rzadka tabela przeskakuje większość takich zakresów; rysunek nie.

Przed początkiem chaosu krzywa nigdy nie wznosi się ponad zero. Dotyka zera przy $r = 3$ i jeszcze raz trochę poniżej $r = 3,5$, za każdym razem, gdy cykl przechodzi w cykl dwa razy dłuższy. Dalej jest przeważnie dodatnia i poprzębiana skierowanymi w dół szpilkami: oknami, w których wraca porządek. Gdzie dokładnie zaczyna się chaos, dlaczego przejścia się zagęszczają i które okno jest najszersze, to pytania rozdziału 8. Odpowiedzią tego rozdziału jest pomiar, który przy każdym ustawieniu odróżnia porządek od chaosu: jedna liczba i jej znak.

Podsumowanie

- Jeden krok mnoży malutki błąd przez **rozciągnięcie**, czyli nachylenie reguły w miejscu, w którym jesteś. Wzdłuż chaotycznej orbity rozciągnięcia bardzo się różnią, a ich zwykła średnia jest złym podsumowaniem.
- Logarytm naturalny** zamienia mnożenie w dodawanie: $\ln(a \times b) = \ln a + \ln b$, $\ln 1 = 0$, $\ln 2 \approx 0,69$ i $\ln e = 1$. Czynn timer wzrostu g na krok staje się tempem



Rysunek 7.4: Wykładnik Lapunowa w zależności od pokrętła r . Jest ujemny wszędzie tam, gdzie orbita się ustala, dotyka zera w każdym miejscu przejścia, jest dodatni tam, gdzie odwzorowanie jest chaotyczne, i znów spada poniżej zera w wąskich oknach. Czerwone kropki to cztery wartości z listingu wykładnika.

$\ln g$, a tempa można uśredniać.

- **Wykładnik Lapunowa** λ to średnia z $\ln |\text{rozciągnięcie}|$ wzdłuż orbity: dodatni dla chaosu, ujemny dla porządku, zero na granicy. Dla odwzorowania logistycznego przy $r = 4$ wynosi $\ln 2$, i obliczenie się z tym zgadza.
- **Horyzont przewidywania** to mniej więcej $\frac{1}{\lambda} \ln(T/\delta)$, a $1/\lambda$ to **czas Lapunowa**. Stan początkowy zmierzony tysiąc razy dokładniej daje $\ln 1000/\lambda$ dodatkowych kroków, przy $r = 4$ mniej więcej dziesięć: stałą porcję, a nie tysiąc razy więcej.
- W zależności od r wykładnik pokazuje, jak **chaos przychodzi i odchodzi**: jest ujemny tam, gdzie orbita się ustala, zerowy w każdym miejscu przejścia, dodatni w chaosie i znów ujemny w wąskich oknach.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

ustalić, jak bardzo jeden krok reguły rozciąga mały błąd, i sprawdzić to, uruchamiając dwie kopie programu	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
używać logarytmu naturalnego, żeby zamienić łańcuch mnożeń w sumę i z powrotem	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
obliczyć wykładnik Lapunowa odwzorowania logistycznego z orbity i odczytać jego znak	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
porównać obliczony wykładnik z wartością znaną dokładnie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zamienić wykładnik, błąd pomiaru i tolerancję w horyzont przewidywania	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co daje lepszy pomiar, licząc w krokach	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Odstęp jest mnożony przez 3, potem przez $\frac{1}{3}$, potem przez 3, potem przez $\frac{1}{3}$. O ile w sumie urosł i ile wynosi zwykła średnia tych czterech czynników?
2. Korzystając z $\ln 2 \approx 0,69$ i $\ln 10 \approx 2,30$, oblicz $\ln 20$ i $\ln 0,2$.
3. Orbita ustala się w punkcie stałym, w którym nachylenie wynosi $-0,5$. Ile wynosi wykładnik Lapunowa i co mówi jego znak?
4. Układ ma $\lambda = 0,5$ na krok. Błąd 0,001 musi pozostać poniżej 0,1. Na mniej więcej ile kroków przewidywanie jest dobre?
5. W tym samym układzie mierzysz stan początkowy sto razy dokładniej. Ile kroków zyskujesz? A jeśli zrobisz to jeszcze raz?
6. Dlaczego szybszy komputer sam w sobie prawie nie wydłuży prognozy układu chaotycznego?
7. Przy $r = 3,84$ wykładnik jest ujemny, choć przy $r = 3,80$ i $r = 3,88$ jest dodatni. Co się tam dzieje?

Droga do chaosu i liczba, której nikt się nie spodziewał

W rozdziale 5 ustawiłeś jedno pokrętło, r , w czterech pozycjach i zobaczyłeś cztery zachowania. Przy $r = 2,8$ odwzorowanie logistyczne ustaliło się na jednej wartości; przy 3,2 przeskakiwało między dwiema; przy 3,5 krążyło po czterech; przy 3,9 nigdy się nie powtórzyło. Cztery zdjęcia jednej maszyny. Co dzieje się pomiędzy nimi? Czy porządek zamienia się w chaos przy jednym ustawieniu pokrętła, tak jak woda wrze w stu stopniach, czy po drodze?

Odpowiedzią jest jeden obraz, jeden z najczęściej reprodukowanych w nauce, a ukryta jest w nim liczba, której nikt tam nie umieścił. Wychodzi też z innych równań, zmierzono ją w ciekłym helu, a pod koniec rozdziału sam obliczysz ją z dokładnością do pięciu cyfr, używając wyłącznie dodawania, odejmowania, mnożenia i dzielenia.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- odczytać diagram bifurkacyjny i powiedzieć, co robi odwzorowanie przy każdym pokazanym na nim ustawieniu pokrętła
- zlokalizować podwojenie okresu, obserwując przebiegi odwzorowania, i powiedzieć, co ogranicza dokładność takiego pomiaru
- znaleźć parametr superstabilny metodą bisekcji, używając wyłącznie $+$, $-$, $\times$ i $\div$
- obliczyć stosunek kolejnych odstępów między podwojeniami i oszacować, gdzie podwojenia się kończą
- przeprowadzić to samo poszukiwanie dla drugiej reguły i powiedzieć, co pokazuje porównanie
- znaleźć okno porządku w zakresie chaotycznym i przytoczyć, co Li i Yorke udowodnili o okresie trzy

8.1 Jeden obraz dla każdego ustawienia pokrętła

Żeby zobaczyć wszystkie r naraz, potrzebujesz przepisu, który zamienia cały przebieg w kilka kropek, a potem rysujesz kropki tysięcy przebiegów obok siebie.

1. Wybierz wartość r i zacznij odwzorowanie od $x = 0,5$.
2. Wykonaj tysiąc kroków i wyrzuć je. Ten odcinek to *stan przejściowy*: orbita szuka drogi tam, dokąd zmierza.

3. Wykonaj jeszcze trzysta i nad r narysuj kropkę na wysokości każdej wartości.

code/ch08/bifurcation.py

```

13 BURN = 1000    # steps thrown away while the orbit settles
14 KEEP = 300    # steps kept: where the orbit lives
15
16
17 def long_run(r: float, burn: int = BURN, keep: int = KEEP) -> list[float]:
18     """The values the logistic map visits at r once it has settled."""
19     xs = orbit(logistic(r), 0.5, burn + keep)
20     return xs[burn + 1:]

```

Listing 8.1: Przepis: pozwól orbicie się ustalić, potem zachowaj to, co odwiedza

orbit to funkcja z biblioteki książki, która stosuje regułę raz za razem i zachowuje każdą wartość; xs[burn + 1:] zostawia te po stanie przejściowym.

Zatrzymaj się i pomyśl 8.1

Przy $r = 2,8$ odwzorowanie ustala się w punkcie stałym $1 - 1/r$, który znalazłeś w rozdziale 5. Ile różnych kropek narysuje przepis nad $r = 2,8$, gdy stan przejściowy zostanie wyrzucony?

Odpowiedz, zanim pójdziesz dalej.

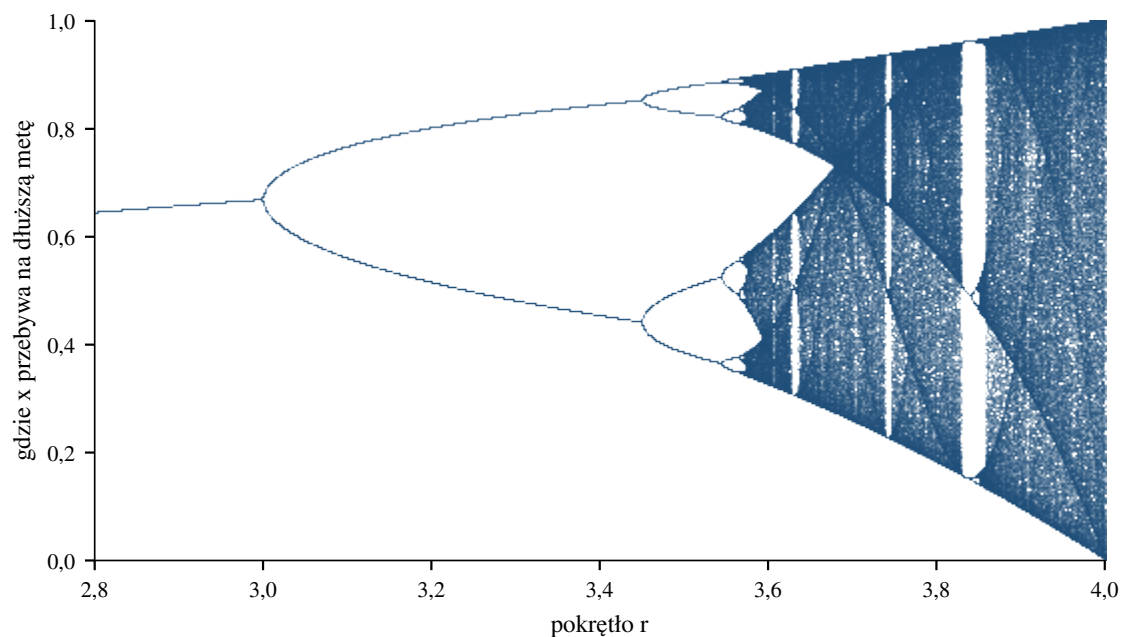
Odpowiedź. Jedną. Wszystkie trzysta wartości to 0,6429 z dokładnością do czterech miejsc po przecinku, czyli $1 - 1/2,8$, więc kropki lądują jedna na drugiej.

Cykl dwóch wartości rysuje dwie kropki, cykl czterech – cztery, a orbita, która nigdy się nie powtarza, rysuje trzysta różnych: krótką pionową smugę. Reszta listingu pyta o sześć wartości r :

r	period	where the orbit lives
2.8	1	0.6429
3.2	2	0.5130 0.7995
3.5	4	0.3828 0.5009 0.8269 0.8750
3.56	8	8 values from 0.35 to 0.89
3.567	16	16 values from 0.34 to 0.89
3.9	-	anywhere from 0.10 to 0.97

Dwa wiersze są nowe: przy 3,56 orbita krąży po ośmiu wartościach, a przy 3,567 po szesnastu. Teraz wykonaj przepis dla tysięcy wartości r między 2,8 a 4 i narysuj wszystkie kropki naraz.

To jest *diagram bifurkacyjny*, cały rozdział 5 w jednym kadrze. Czytaj go od lewej do prawej, jak podróż wzdłuż pokręta. Jedna linia aż do $r = 3$; tam rozdziela się na dwie; w pobliżu 3,45 każda gałąź rozdziela się znowu; potem wszystko rozplywa się w smugę poprzecinaną jasnymi pasami. *Bifurkacja* (od łacińskiego słowa oznaczającego dwuzębne widły) to wartość pokręta, przy której zmienia się rodzaj długoterminowego zachowania.



Rysunek 8.1: Diagram bifurkacyjny odwzorowania logistycznego. Nad każdą wartością r kropki pokazują, gdzie x spędza czas po zakończeniu stanu przejściowego: jedna linia tam, gdzie się ustala, dwie tam, gdzie przeskakuje, smuga tam, gdzie nigdy się nie powtarza.

Każde rozwidlenie tutaj to *podwojenie okresu*: każda odwiedzana wartość staje się dwiema, więc cykl potrzebuje dwa razy więcej czasu, żeby się zamknąć.

Zatrzymaj się i pomyśl 8.2

Spójrz na obraz między $r = 3,5$, gdzie orbita ma okres cztery, a smugą. Czy porządek ustępuje chaosowi przy jednej wartości r , jak po przestawieniu przełącznika?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Wygląda jak przełącznik, ale nim nie jest. Powiększ odcinek przed smugą, a każda gałąź rozdzieli się znowu i znowu: okresy 8, 16, 32 i tak dalej bez końca, wszystkie mieszczące się przed $r = 3,5699$. Tabela złapała już dwa z nich.

Pułapka

„Przejście od porządku do chaosu to nagłe przełączenie.” Nie na tej drodze. Porządek rozpada się w nieskończonej *kaskadzie* podwojeń, z których każde przychodzi szybciej niż poprzednie, a chaos zaczyna się tam, gdzie kaskada się kończy. Smuga wygląda na nagłą tylko dlatego, że jej ostatnie rozwidlenia leżą zbyt blisko siebie, by je narysować. Inne układy dochodzą do chaosu innymi drogami; ta, *kaskada podwojeń okresu*, to droga, którą idzie pojedynczy garb.

8.2 Rozwidlenia, które się zagęszczają

Gdzie dokładnie leży każde rozwidlenie? Tabela wyznacza dla nich przedziały: okres jeden przy 2,8, okres dwa przy 3,2, więc pierwsze rozwidlenie leży pomiędzy. Wypróbuj środek przedziału, sprawdź, na jakim okresie odwzorowanie się tam ustala, zatrzymaj połowę, w której wciąż jest zmiana, i powtórz. Dwadzieścia rund zmniejsza przedział milion razy. To jest *bisekcja*, a potrzebuje ona tylko sposobu, żeby zapytać „czy tutaj to wciąż okres p ?”

code/ch08/doublings.py

```

19 def fork(p: int, lo: float, hi: float, rounds: int = 20) -> float:
20     """The r between lo and hi at which period p gives way to 2p."""
21     for _ in range(rounds):
22         mid = (lo + hi) / 2
23         q = settled_period(mid)
24         if q is not None and q <= p:
25             lo = mid          # still period p here: the fork is further on
26         else:
27             hi = mid          # already doubled: the fork is behind us
28     return (lo + hi) / 2

```

Listing 8.2: Namierzanie rozwidlenia przez połowienie przedziału

`settled_period` wykonuje dwadzieścia tysięcy kroków odwzorowania i pyta biblioteczną funkcję `period`, co się powtarza. Listing uruchamia `fork` na czterech przedziałach, które daje tabela.

Zatrzymaj się i pomyśl 8.3

Okazuje się, że pierwsze dwa rozwidlenia dzieli 0,4498 w r , a kolejne dwa 0,0947. Ile razy mniejszy jest drugi odstęp? Jeśli następny zmaleje o ten sam czynnik, ile będzie wynosił?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około 4,75 razy mniejszy: $0,4498 \div 0,0947$. Ten sam czynnik jeszcze raz dałby następny odstęp około 0,02.

fork	r	gap	ratio
1 -> 2	2.9995		
2 -> 4	3.4493	0.4498	
4 -> 8	3.5440	0.0947	4.75
8 -> 16	3.5644	0.0204	4.64

Trzeci odstęp wynosi 0,0204; drugi podzielony przez trzeci daje 4,64. Każdy odstęp jest więc nieco mniej niż pięć razy większy od następnego, a pozostałe odstępy (ostatni podzielony przez 4,7, potem jeszcze raz przez 4,7 i tak dalej) dają razem ostatni odstęp podzielony przez 4,7 – 1, czyli nieco ponad jego ćwierć. Nieskończenie wiele rozwidleń mieści się na skończonym odcinku pokrętła: kaskada kończy się niedaleko za 3,5644.

Zatrzymaj się i pomyśl 8.4

Pierwsze rozwidlenie znamy dokładnie. W rozdziale 5 okazało się, że punkt stały traci stabilność przy $r = 3$, gdzie jego nachylenie, $2 - r$, przekracza -1 . Tabela podaje 2,9995. Co poszło nie tak?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nic w regule: pomiarowi zabrakło cierpliwości. Przy $r = 2,9998$ każdy krok mnoży małe zaburzenie przez $2 - r = -0,9998$, więc zaburzenie maleje o jedną pięćdziesiątą procenta na krok, a po dwudziestu tysiącach kroków orbita wciąż wyraźnie przeskakuje z boku na bok. `settled_period` uznaje to za cykl o okresie dwa.

Każde rozwidlenie mierzone przez obserwację wypada za wcześnie z tego samego powodu: blisko rozwidlenia cykl, który traci stabilność, ustala się bardzo powoli.

Uwaga

Drugie rozwidlenie da się znaleźć dokładnie. Przy jednym obiegu cyklu o okresie dwa zaburzenie jest mnożone przez nachylenie w każdym z jego dwóch punktów, a strona szkolnej algebry pokazuje, że iloczyn wynosi $4 + 2r - r^2$. Osiąga on -1 , gdy $r^2 - 2r - 5 = 0$, czyli przy $r = 1 + \sqrt{6} = 3,4495$; 3,4493 z tabeli wypada za wcześnie, zgodnie z oczekiwaniem. Późniejsze rozwidlenia nie mają tak prostego wzoru.

8.3 Szczyt garbu

W 1975 roku Mitchell Feigenbaum, fizyk z laboratorium w Los Alamos w Nowym Meksyku, wyznaczał położenie tych rozwidleń na programowalnym kalkulatorze kieszonkowym i zauważył wzór, który właśnie zobaczyłeś: każdy odstęp to stała część poprzedniego. Żeby zmierzyć tę część dokładnie, użyj lepiej położonego punktu orientacyjnego niż rozwidlenie, na którym cykl trzyma się resztkami sił: punktu na każdym odcinku, w którym cykl trzyma się najmocniej.

Nachylenie odwzorowania logistycznego w punkcie x wynosi $r(1 - 2x)$ – to współczynnik rozciągnięcia, który nauczyłeś się mierzyć w rozdziale 3. Jest równe zero przy $x = 0,5$, na szczycie garbu. Cykl przechodzący przez 0,5 mnoży każde zaburzenie przez to zero raz na obieg i je wymazuje. Taki cykl nazywa się *superstabilnym*, a dla każdego okresu 1, 2, 4, 8, ... jest na drodze dokładnie jedna wartość r , przy której to się dzieje: w tym rozdziale nazywamy te wartości *punktami orientacyjnymi*. Każdy z nich to rozwiązanie jednego równania: zacznij od 0,5, zrób tyle kroków, ile wynosi okres, i wróć do 0,5.

Zatrzymaj się i pomyśl 8.5

Dla okresu jeden: przy jakim r jeden krok przenosi 0,5 z powrotem prosto do 0,5? Krok to $r \times 0,5 \times 0,5$.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $r = 2$. Połowa połowy to ćwierć, a $2 \times 0,25 = 0,5$.

Dla dłuższych okresów nie ma skrótu w pamięci, a bisekcja żadnego nie potrzebuje. `from_top` podaje, jak daleko od 0,5 kończy się orbita; ta odległość jest dodatnia po jednej stronie punktu orientacyjnego i ujemna po drugiej:

code/ch08/superstable.py

```

20 def from_top(f: Rule, steps: int) -> float:
21     """Start at the top of the hump; how far from it are you after steps?"""
22     x = 0.5
23     for _ in range(steps):
24         x = f(x)
25     return x - 0.5
26
27
28 def bisect(g: Callable[[float], float], lo: float, hi: float,
29           rounds: int = 60) -> float:
30     """A root of g between lo and hi, where g changes sign: halve and keep
31     the half in which the sign still changes."""
32     below = g(lo) < 0
33     if g(hi) < 0 == below:
34         raise ValueError("g must change sign between lo and hi")
35     for _ in range(rounds):
36         mid = (lo + hi) / 2
37         if g(mid) < 0 == below:
38             lo = mid
39         else:
40             hi = mid
41     return (lo + hi) / 2

```

Listing 8.3: Zaczynij na szczycie i połowiaj przedział, aż orbita do niego wróci

Pierwsze dwa punkty orientacyjne otaczamy przedziałami na oko, z obrazu. Potem każdy odstęp jest mniejszy od poprzedniego, więc następnego punktu szukamy między jedną dziesiątą a połową ostatniego odstepu dalej – to hojny przedział dla odstępów malejących o czynnik bliski pięciu. *Rodzina* to funkcja, która dla danego r buduje regułę.

code/ch08/superstable.py

```

46 def returns(family: Family, steps: int) -> Callable[[float], float]:
47     """The function of r whose root is the landmark with this period."""
48     return lambda r: from_top(family(r), steps)
49
50
51 def landmarks(family: Family, first: tuple[float, float],
52              second: tuple[float, float], count: int,
53              base: int = 1) -> list[float]:
54     """The landmarks for periods base, 2*base, 4*base, ... `first` and
55     `second` bracket the first two; each later one is sought between a
56     tenth and a half of the last gap beyond the last landmark found."""
57     found = [bisect(returns(family, base), *first),
58             bisect(returns(family, 2 * base), *second)]
59     for n in range(2, count):
60         gap = found[-1] - found[-2]
61         found.append(bisect(returns(family, base * 2 ** n),

```

```

62         found[-1] + gap / 10, found[-1] + gap / 2))
63     return found
64
65
66 LOGISTIC = (logistic, (1.5, 2.5), (3.0, 3.4))

```

Listing 8.4: Punkty orientacyjne dla okresów 1, 2, 4, 8, ..., każdy otoczony na podstawie poprzedniego

Listing znajduje pierwszych 11 punktów orientacyjnych, dla okresów od 1 do 1024, używając wyłącznie +, −, × i ÷, więc każda cyfra jest taka sama na każdym komputerze:

n	period	landmark	R_n	gap	ratio
0	1	2.0000000000			
1	2	3.2360679775	1.2360679775		
2	4	3.4985616993	0.2624937218	4.7089	
3	8	3.5546408628	0.0560791634	4.6808	
4	16	3.5666673799	0.0120265171	4.6630	
5	32	3.5692435316	0.0025761518	4.6684	
6	64	3.5697952937	0.0005517621	4.6690	
7	128	3.5699134654	0.0001181717	4.6692	
8	256	3.5699387742	0.0000253088	4.6692	
9	512	3.5699441946	0.0000054204	4.6692	
10	1024	3.5699453555	0.0000011609	4.6692	

Spójrz w dół ostatniej kolumny: każdy odstęp podzielony przez następny. Przez kilka wierszy się waha, a potem przestaje się zmieniać: 4,6692. To *stała Feigenbauma*, oznaczana δ (grecka litera delta). Punkty orientacyjne dają ją znacznie dokładniej niż rozwidlenia, bo w punkcie orientacyjnym orbita ustala się od razu, a nie ledwie wcale. Rozwidlenia i punkty orientacyjne występują na drodze na zmianę, po jednym z każdego na okres, więc jedno i drugie zagęszczają się w tym samym tempie ku temu samemu końcowi, $r = 3,5699$. Punkt orientacyjny okresu dwa, 3,2361, to dokładnie $1 + \sqrt{5}$, gdzie iloczyn z notatki, $4 + 2r - r^2$, jest równy zeru.

Feigenbaum opublikował wyniki w 1978 roku; Pierre Couillet i Charles Tresser we Francji znaleźli tę samą liczbę niezależnie, mniej więcej w tym samym czasie. To, co zrobił potem, sprawiło, że ta liczba jest sławna.

Laboratorium 8.1 – Znajdź punkt orientacyjny sam

Napisz `from_top` i `bisect`, a potem połącz je w `superstable`. Sprawdź swoją odpowiedź dla okresu dwa, porównując ją z $1 + \sqrt{5}$. Następnie znajdź r między 3,82 a 3,84, przy którym 0,5 wraca do siebie po *trzech* krokach. Jeszcze spotkasz tę liczbę.

Plik startowy: `code/labs/ch08/l08_01_superstable.py`

Test: `code/labs/ch08/test_l08_01_superstable.py`

Uruchom: `cd code && uv run pytest -k l08_01_superstable`

8.4 Liczba, która należy do drogi

Następnym krokiem Feigenbauma było wypróbowanie innej reguły, *odwzorowania sinusowego*

$$x_{n+1} = s \sin(\pi x_n),$$

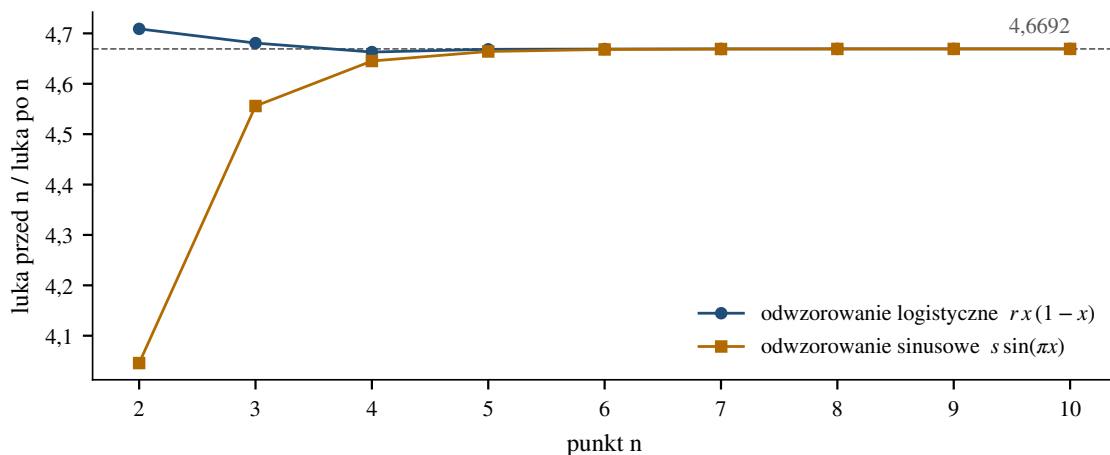
które zwraca s razy sinus z πx . Nie potrzebujesz trygonometrii, żeby za nim nadażyć: dla x między 0 a 1 sinus z πx to gładki garb, który rośnie od 0 do 1 przy $x = 0,5$ i opada z powrotem do 0. Odwzorowanie sinusowe ma więc kształt odwzorowania logistycznego – jeden garb ze szczytem w 0,5 – i nic z jego wzoru: żadnego $x(1-x)$, inne pokrętko s , krzywą, która nie jest parabolą.

Zatrzymaj się i pomyśl 8.6

Przeprowadź to samo poszukiwanie punktów orientacyjnych dla odwzorowania sinusowego. Czy spodziewasz się, że jego odstępów będą maleć znowu o 4,6692 z odwzorowania logistycznego, czy o jakąś inną liczbę, należącą do sinusa?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. O tę samą liczbę. Odstępy odwzorowania sinusowego maleją najpierw o 4,05, a potem, punkt po punkcie, zbliżają się do 4,67, na której zatrzymują się też stosunki odwzorowania logistycznego.



Rysunek 8.2: Każdy odstęp między punktami orientacyjnymi podzielony przez następny, dla dwóch reguł, które łączy tylko pojedynczy gładki garb. Zaczynają daleko od siebie i kończą na tej samej liczbie.

Listing nie wie o odwzorowaniu sinusowym nic poza jego rodziną i dwoma przedziałami odczytanymi z jego diagramu:

code/ch08/two_maps.py

```

17 SINE = (sine_map, (0.4, 0.6), (0.72, 0.84)) # brackets read off its diagram
18
19
20 def main() -> None:
21     a = landmarks(*LOGISTIC, count=11)
22     b = landmarks(*SINE, count=11)
23     print(" n    logistic    sine map")
24     for n, (x, y) in enumerate(zip(ratios(a), ratios(b), strict=True), 2):
25         print(f"{n:2d}   {x:9.2f}   {y:9.2f}")
26     print(f"last{a[-1]:9.4f}   {b[-1]:9.4f}")

```

Listing 8.5: To samo poszukiwanie dla dwóch niepowiązanych reguł

n	logistic	sine map
2	4.71	4.05
3	4.68	4.56
4	4.66	4.65
5	4.67	4.66
6	4.67	4.67
7	4.67	4.67
8	4.67	4.67
9	4.67	4.67
10	4.67	4.67
last	3.5699	0.8656

Obie drogi kończą się w różnych miejscach (ostatni wiersz: 3,5699 i 0,8656) i inaczej się zaczynają, a jednak zatrzymują się na tym samym stosunku, co do każdej wydrukowanej cyfry. Funkcja `ratios` dzieli każdy odstęp przez następny; drugie laboratorium prosi cię, żebyś ją napisał.

Pułapka

„Inne równanie daje inną drogę.” Daje inne drogowskazy – rozwidlenia leżą przy innych wartościach pokręta – ale droga ma ten sam kształt, a jej rozwidlenia zagęszczają się w tym samym tempie. Ta liczba należy do drogi, nie do równania. To jest *uniwersalność* i dlatego stała przyrody może wyjść z zabawkowego modelu.

Dlaczego? O każdym kolejnym rozwidleniu decyduje coraz krótszy odcinek odwzorowania w pobliżu szczytu garbu, oglądany przez coraz silniejsze szkło powiększające. Powiększ dowolny gładki, zaokrąglony garb dostatecznie mocno wokół szczytu, a wygląda jak parabola; wszystko, co czyniło odwzorowanie sinusowe sinusem, zostało powiększone poza pole widzenia. Stałe szkolnej geometrii, takie jak π , należą do kształtów. Ta należy do procesu.

Był to szok, bo przewidywał coś o układach, których równań nikt nie potrafił zapisać: jeśli któryś dochodzi do chaosu przez podwajanie okresu, jego rozwidlenia powinny się zagęszczać w tempie Feigenbauma, bez względu na to, z czego jest zrobiony.



Rysunek 8.3: Uniwersalność w jednym zdaniu: każda reguła z jednym gładkim garbem idzie tą samą drogą, a droga niesie własny stosunek.

Gdzie to spotkasz

Około 1980 roku Albert Libchaber i Jean Maurer w Paryżu podgrzewali od dołu maleńkie pudełko z ciekłym helem i rejestrowali temperaturę w jednym punkcie, gdy płyn się kotłował. Gdy zwiększali grzanie, rytm temperatury podwoił okres, a potem podwoił go znowu. Późniejsze doświadczenia z konwekcją w cieczach i z obwodami elektronicznymi mierzyły stosunek kolejnych odstępów i znajdowały liczby bliskie liczbie Feigenbauma: nie z dokładnością do czterech miejsc po przecinku, bo w laboratorium da się rozróżnić tylko kilka pierwszych podwojeń, ale na tyle bliskie, że nikt nie uważa tego za przypadek. W 1986 roku Feigenbaum i Libchaber otrzymali wspólnie Nagrodę Wolfa w dziedzinie fizyki.

Czego tu nie dowodzimy

Nie dowodzimy tutaj, dlaczego każdy gładki garb daje tę samą stałą δ . Feigenbaum wyjaśnił to metodą z fizyki, *renormalizacją*, która porównuje odwzorowanie po każdym podwojeniu z przeskalowaną kopią samego siebie; Oscar Lanford podał dowód wspomagany komputerowo w 1982 roku. *Gładkość* ma znaczenie: garb o szczytynie bardziej płaskim niż u paraboli daje inną stałą.

Laboratorium 8.2 – Stosunek Feigenbauma i miejsce, w którym kończą się podwojenia

Napisz `ratios`, które dzieli każdy odstęp przez następny, oraz `where_it_ends`, które sumuje wszystkie pozostałe odstępów, zakładając, że każdy to poprzedni podzielony przez ostatni stosunek. Test podaje twoim funkcjom punkty orientacyjne obu odwzorowań: jeden stosunek, dwa różne końce.

Plik startowy: `code/labs/ch08/l08_02_ratio.py`

Test: `code/labs/ch08/test_l08_02_ratio.py`

Uruchom: `cd code && uv run pytest -k l08_02_ratio`

8.5 Porządek wewnątrz chaosu

Za $r = 3,5699$ podwojenia się wyczerpują, a diagram zamienia się w smugę.

Zatrzymaj się i pomyśl 8.7

Czy od $r = 3,5699$ do $r = 4$ odwzorowanie logistyczne jest chaotyczne przy każdym ustawieniu pokrętki, tak jak przy 3,9?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Spójrz jeszcze raz na obraz: smugę przecinają jasne pasy, a w każdym z nich orbita wraca do krótkiego cyklu. Najszerszy, w pobliżu 3,83, to okno okresu trzy.

code/ch08/windows.py przeprowadza r przez ten pas z tą samą funkcją `settled_period`, a potem szuka punktów orientacyjnych w jego wnętrzu:

```

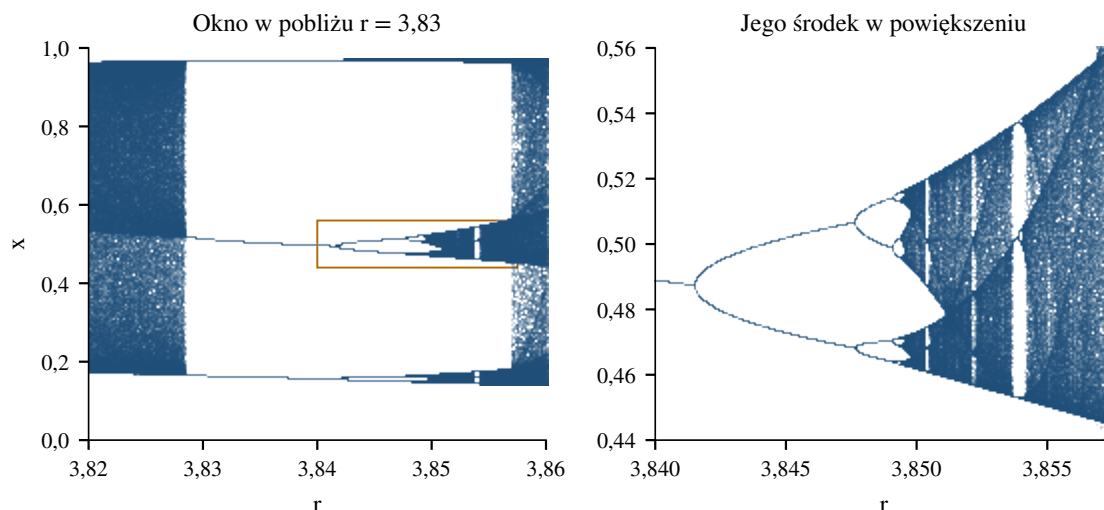
r    period
3.82    -
3.8284  -
3.8285   3
3.83     3
3.84     3
3.845    6
3.848   12
3.85     -
the window opens at 1 + sqrt(8) = 3.82843
periods 3, 6, 12, ...: last ratio 4.669

```

Przy 3,8284 orbita się nie powtarza; przy 3,8285 ustaliła się już w cyklu o okresie trzy. Okno otwiera się pomiędzy nimi, dokładnie przy $r = 1 + \sqrt{8} = 3,82843$. Cykl o okresie trzy podwaja się potem do sześciu i dwunastu, a przy 3,85 znów rozplywa się w smugę. Cała droga powtarza się w miniaturze. Pierwszy punkt orientacyjny okna to r , które znalazłeś w pierwszym laboratorium, 3,8319, a jego punkty orientacyjne zagęszczają się w stosunku 4,669: znowu liczba Feigenbauma, tym razem dla okresów 3, 6, 12, ... zamiast 1, 2, 4, ...

Pułapka

„Gdy układ raz osiągnie chaos, jest chaotyczny wszędzie dalej.” Między $r = 3,5699$ a 4 odwzorowanie logistyczne wraca do porządku nieskończenie wiele razy, w oknach o każdym okresie. Przekręć pokrętkę odrobinę, a chaotyczny układ może stać się idealnie okresowy; przekręć je odrobinę dalej, a znów jest chaotyczny. Wykładnik z rozdziału 7 spada poniżej zera dokładnie w tych pasach.



Rysunek 8.4: Po lewej: okno okresu trzy. Po prawej: środkowa gałąź okna w powiększeniu: mała kopia całego diagramu bifurkacyjnego, z własną kaskadą, własną smugą i własnymi oknami.

Uwaga

W 1975 roku Tien-Yien Li i James Yorke opublikowali pracę, której tytuł był jej wynikiem: *Period three implies chaos* (okres trzy pociąga za sobą chaos). W skrócie: jeśli ciągła reguła przenosi przedział w siebie, tak jak odwzorowanie logistyczne przenosi przedział od 0 do 1, i jakiś punkt startowy wraca do siebie po dokładnie trzech krokach, to reguła ma cykle każdej długości oraz punkty startowe, których orbity nigdy nie ustalają się w żadnym cyklu. Praca nadała słowu *chaos* jego matematyczne znaczenie; silniejszy wynik udowodnił w 1964 roku Ołeksandr Szarkowski w Kijowie, ale na Zachodzie był mało znany aż do czasu tej pracy. W oknie prawie każdy punkt startowy kończy na cyklu o okresie trzy, ale twierdzenie nadal obowiązuje: pozostałe cykle są na miejscu, niestabilne, a orbita może długo błąkać się między nimi, zanim się ustali.

Diagram ma jeszcze jedną niespodziankę, a należy ona do rozdziału 12: cały ten obraz leży wzdłuż linii przecinającej najstojniejszy obraz matematyki, a rozwidlenia i okno są dokładnie tam, gdzie umieszcza je jednoliniżkowe przeliczenie.

Podsumowanie

- **Diagram bifurkacyjny** pokazuje dla każdej wartości pokrętła, gdzie orbita przebywa po zakończeniu stanu przejściowego. **Bifurkacja** to wartość, przy której zmienia się rodzaj długoterminowego zachowania; na tej drodze każda jest **podwojeniem okresu**.
- Rozwidlenia się zagęszczają, każdy odstęp jest nieco mniej niż pięć razy większy od następnego, więc nieskończenie wiele z nich mieści się przed $r = 3,5699$. Zmierzone przez obserwację rozwidlenie wypada za wcześnie, bo w jego pobliżu orbita ustala się bardzo powoli.

- W **superstabilnym** punkcie orientacyjnym cykl przechodzi przez szczyt garbu, gdzie nachylenie jest zerowe. Każdy taki punkt to jedno równanie, rozwiązywane **bisekcją** za pomocą $+$, $-$, $\times$ i $\div$, a stosunek odstępów między nimi zatrzymuje się na **stałej Feigenbauma**, 4,6692.
- Odwzorowanie sinusowe daje ten sam stosunek: liczba należy do drogi, którą idzie każda gładka reguła z jednym garbem, a nie do wzoru. To jest **uniwersalność**, zmierzona w cieczech i obwodach.
- Za kaskadą chaos przerywają **okna** porządku. Najszerze, okresu trzy, otwiera się przy $1 + \sqrt{8}$ i zawiera miniaturę całej drogi; Li i Yorke udowodnili, że okres trzy pociąga za sobą chaos.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrödziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdiesz dalej.

odczytać diagram bifurkacyjny i powiedzieć, co robi odwzorowanie przy każdym pokazanym na nim ustawieniu pokręćła	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zlokalizować podwojenie okresu, obserwując przebiegi odwzorowania, i powiedzieć, co ogranicza dokładność takiego pomiaru	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
znaleźć parametr superstabilny metodą bisekcji, używając wyłącznie $+$, $-$, $\times$ i $\div$	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
obliczyć stosunek kolejnych odstępów między podwojeniami i oszacować, gdzie podwojenia się kończą	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
przeprowadzić to samo poszukiwanie dla drugiej reguły i powiedzieć, co pokazuje porównanie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
znaleźć okno porządku w zakresie chaotycznym i przytoczyć, co Li i Yorke udowodnili o okresie trzy	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Co pokazuje kolumna kropek nad jedną wartością r na diagramie bifurkacyjnym i jak wygląda dla punktu stałego, cyklu o okresie dwa i chaosu?
2. Dwa kolejne odstępy między rozwidleniami wynoszą 0,08 i 0,017. Mniej więcej jaki jest ich stosunek i ile powinien wynosić następny odstęp?
3. Dlaczego rozwidlenie zlokalizowane przez obserwację orbity wypada nieco za wcześnie?

4. Czym jest superstabilny punkt orientacyjny i dlaczego da się go znaleźć z dokładnością do wielu cyfr za pomocą samych $+$, $-$, $\times$ i $\div$?
5. Podwojenia odwzorowania sinusowego kończą się około 0,8656, a logistycznego około 3,5699. Czy to przeczy uniwersalności?
6. Czy odwzorowanie logistyczne jest chaotyczne przy $r = 3,83$? Co Li i Yorke udowodnili o okresie trzy?

Część III

Kształty chaosu

Trzy równania na pogodę: atraktor Lorenza

Postaw szeroki garnek z wodą na małym ogniu. Ciepła woda przy dnie chce się unieść, a chłodna przy powierzchni opaść, i powyżej pewnej temperatury woda zaczyna się toczyć: w górę z jednej strony, w dół z drugiej, jak powolne koło. Powietrze robi to samo nad nasłonecznionym polem, a w większej skali to toczenie się jest jednym z silników pogody.

Opisz ten wir trzema liczbami i trzema krótkimi regułami, dokładnymi, bez żadnej przypadkowości. Czy będzie się kręcił równo, jak koło? Czy też trzy liczby mogą być tak nieregularne jak niebo? Część II znalazła chaos w regułach, które robią jeden krok na raz. Ten rozdział znajduje go w czymś, co płynie – a potem znajduje wewnątrz chaosu coś, czego nikt się nie spodziewał: kształt.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- przeczytać trzy reguły zmiany Lorenza wyraz po wyrazie i powiedzieć, co każdy wyraz robi z wirem
- prowadzić reguły metodą RK4 i odczytać wynik z dwóch widoków: jedna liczba względem drugiej oraz jedna liczba względem czasu
- zmierzyć, jak szybko rozchodzą się dwa prawie identyczne przebiegi, i zamienić to tempo na horyzont przewidywania
- powiedzieć, czym jest atraktor i dlaczego układ chaotyczny nie może pójść, gdzie zechce
- policzyć statystykę przebiegu i sprawdzić, że nie zależy od tego, skąd przebieg wystartował

9.1 Trzy liczby na garnek wody

W 1963 roku Edward Lorenz, meteorolog z Massachusetts Institute of Technology, opublikował pracę zatytułowaną „Deterministic Nonperiodic Flow”, czyli „Deterministyczny nieokresowy przepływ”. Ten tytuł to cały rozdział w pigułce. *Deterministyczny*: w regułach nie ma przypadku. *Nieokresowy*: ruch nigdy się nie powtarza. *Przepływ*: czas płynie gładko, więc reguła podaje prędkość, z jaką zmienia się stan, a nie jego następną wartość, jak w rozdziale 4. Model Lorenza to drastycznie okrojony opis płynu podgrzewanego od dołu i zachowuje trzy liczby:

- x : jak szybko kręci się wir i w którą stronę (dodatnie w jedną, ujemne w drugą);

- y : o ile cieplejsza jest strona, którą woda się wznosi, od strony, którą opada;
- z : o ile temperatura, od dołu do góry, została odgięta od równomiernego spadku przez mieszanie, które robi wir.

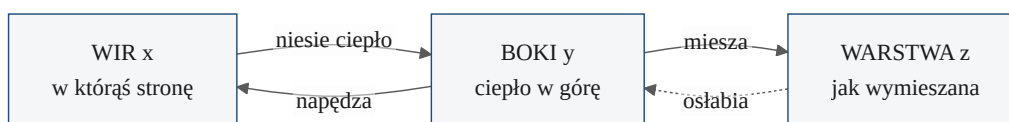
Zapisuj dx/dt i czytaj „tempo x ”: to, jak szybko x zmienia się w tej chwili; to nazwa prędkości, a nie ułamek do skracania. Reguły Lorenza to

$$\frac{dx}{dt} = \sigma (y - x), \quad \frac{dy}{dt} = x (\rho - z) - y, \quad \frac{dz}{dt} = x y - \beta z,$$

z trzema stałymi liczbami nazwanymi greckimi literami: σ (sigma) to 10, ρ (ro) to 28, a β (beta) to $\frac{8}{3}$. Słowami:

- **Wir goni różnicę temperatur.** Gdy y jest większe od x , wir przyspiesza, a gdy mniejsze, zwalnia; σ mówi, jak żwawo.
- **Wir buduje różnicę, a różnica ucieka.** Po wymnożeniu druga reguła to $\rho x - xz - y$. Kręcący się wir wynosi ciepłą wodę w górę z jednej strony i sprowadza chłodną w dół z drugiej, budując y w tempie ρx , gdzie ρ mówi, jak mocno grzejemy garnek ($\rho = 1$ to najsłabsze grzanie, przy którym wir w ogóle rusza). Wyraz $-y$ to różnica uciekająca na boki; wyraz $-xz$ to mieszanie, które się odgryza.
- **Wir miesza warstwę, a warstwa się uspokaja.** Wir wynoszący ciepłą wodę w górę (x i y tego samego znaku) wygina profil temperatury w tempie xy ; pozostawiony sam sobie profil wraca w tempie βz , gdzie β wynika z kształtu wirów.

Tylko dwa wyrazy, xz i xy , mnożą przez siebie dwie z trzech liczb; każdy inny wyraz to stała liczba razy jedna z nich. Te dwa iloczyny to cała nieliniowość modelu w sensie z rozdziału 3 i okaże się, że wystarczą.



Rysunek 9.1: Trzy liczby popychają się nawzajem w kółko. Wir buduje różnicę temperatur, różnica napędza wir, a razem mieszają warstwę, która osłabia różnicę.

Zatrzymaj się i pomyśl 9.1

Załóżmy, że woda stoi zupełnie nieruchomo: $x = 0$, $y = 0$ i $z = 0$. Co mówią trzy reguły?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Każde tempo jest zerem: $10 \times (0 - 0) = 0$, potem $0 - 0 = 0$, potem $0 - 0 = 0$. Nic się nie zmienia.

To garnek, który po cichu przekazuje ciepło w górę, bez żadnego wiru. Stan, w którym każde tempo jest zerem, to *punkt stały* przepływu, w języku rozdziału 2: *zaczynij w nim, a w nim zostaniesz*. Nie jest jedyny.

Zatrzymaj się i pomyśl 9.2

Sprawdź $x = y = \sqrt{72}$, czyli około 8,49, oraz $z = 27$. Czy wszystkie trzy tempa są zerem? Potrzebujesz tylko $\frac{8}{3} \times 27 = 72$.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tak. Pierwsze tempo to $10 \times 0 = 0$. Drugie to $x \times (28 - 27) - y = x - y = 0$. Trzecie to $x \times x - 72 = 72 - 72 = 0$.

To *ustalony wir*: kręci się ze stałą prędkością przy stałej różnicy temperatur. Z minusami, $x = y = -\sqrt{72}$ i $z = 27$, to ten sam wir kręcący się w drugą stronę. Dla każdego grzania ρ większego od 1 oba ustalone wiry leżą w $x = y = \pm\sqrt{\beta(\rho - 1)}$ i $z = \rho - 1$, gdzie $\pm$ znaczy „plus albo minus”.

Laboratorium 9.1 – Reguły Lorenza własną ręką

Napisz `rates`, która dostaje stan (x, y, z) i zwraca trzy tempa, oraz `fixed_points`, która zwraca każdy stan, w którym wszystkie trzy są zerem. Test porównuje twoje tempa z biblioteką książki i sprawdza, że naprawdę są zerem w każdym punkcie, który zwrócisz.

Plik startowy: `code/labs/ch09/l09_01_rates.py`

Test: `code/labs/ch09/test_l09_01_rates.py`

Uruchom: `cd code && uv run pytest -k l09_01_rates`

9.2 Za przepływem

Oto te same reguły w wersji z biblioteki książki. Ostatnie trzy wiersze to trzy reguły; `lorenz` zwraca funkcję, która dostaje stan i oddaje jego trzy tempa, czyli dokładnie to, o co prosiło twoje laboratorium.

`code/src/chaoslab/systems.py`

```

25 def lorenz(sigma: float = 10.0, rho: float = 28.0,
26         beta: float = 8.0 / 3.0) -> Field:
27     """Lorenz's 1963 convection model: three numbers, three rules."""
28
29     def field(s: State) -> State:
30         x, y, z = s
31         return (sigma * (y - x),
32                 x * (rho - z) - y,
33                 x * y - beta * z)
34
35     return field

```

Listing 9.1: Reguły Lorenza z 1963 roku w chaoslab

Żeby za nimi podążać, użyj metody RK4 z rozdziału 4: `integrate` robi kroki długości `DT`, czyli jednej setnej jednostki czasu, i zapamiętuje każdy stan. Jednostka czasu jest własną jednostką modelu, a nie sekundą ani godziną.

code/ch09/lorenz_run.py

```
13 DT = 0.01                                # one step is a hundredth of a time unit
14 field = lorenz(sigma=10.0, rho=28.0, beta=8.0 / 3.0)
15 path = integrate(field, (1.0, 1.0, 1.0), DT, 4000) # forty time units
```

Listing 9.2: Czterdzieści jednostek czasu przepływu Lorenza, od stanu (1, 1, 1)

Zatrzymaj się i pomyśl 9.3

Wahadło z tarcie wchodzi spiralą w swój punkt spoczynku. Garnek Lorenza ma trzy punkty stałe. Czy przebieg rozpoczęty w (1, 1, 1), blisko nieruchomej wody, osiadnie w jednym z nich?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Zostaje odepchnięty od nieruchomej wody, krąży wokół jednego ustalonego wiru coraz szerszymi pętlami, zostaje przerzucony, by krążyć wokół drugiego, i nigdy się nie uspokaja.

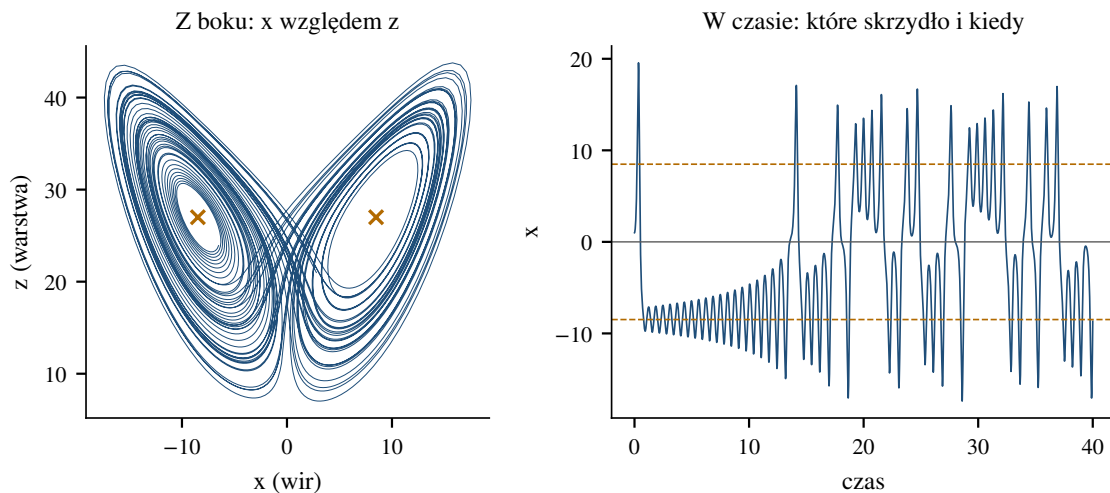
Przy tym grzaniu żaden ustalony wir nie potrafi zatrzymać stanu; potrafią tylko wtedy, gdy ρ jest mniejsze od $\frac{470}{19}$, czyli około 24,74 – tę liczbę daje z reguł algebra, której ta książka nie przeprowadza. Zmniejsz grzanie do 20, a ten sam start rzeczywiście się uspokoi: ostatni wiersz wydruku to ustalony wir w $x = y = -7,12$, $z = 19$.

time	x	y	z	wing
0	1.00	1.00	1.00	R
2	-8.17	-9.56	24.62	L
4	-10.09	-10.93	27.97	L
6	-9.74	-7.71	30.89	L
8	-7.01	-4.59	28.39	L
10	-4.90	-3.74	24.69	L
12	-3.34	-1.98	23.60	L
14	8.75	16.25	12.39	R
16	-6.58	-10.81	15.86	L
18	0.84	-0.85	22.29	R
20	13.47	12.69	34.39	R

loops: LLLLLLLLLLLLLLLLLLRLRRRLRRLLRLRRRRLLRLRRLL
rho = 20, after 100 time units: -7.12, -7.12, 19.00

Wiersz oznaczony `loops` wypisuje jedną literę na pętlę: L dla pętli po stronie, gdzie x jest ujemne, R dla drugiej. Przebieg robi 19 pętli po lewej, zanim zrobi pierwszą po prawej, a potem litery idą seriami od jednej do pięciu, bez żadnego wzoru. Nie ma go czego szukać: to jest *nieokresowość* z tytułu Lorenza.

Lewa połowa rysunku 9.2 zasłużyła na nazwę *motyla Lorenza*. Każde skrzydło to wir kręcący się w jedną stronę; przejście między nimi to wir zmieniający kierunek.



Rysunek 9.2: Przebieg od $(1, 1, 1)$. Z lewej: x względem z od chwili 5 do chwili 60 – pętla wokół dwóch ustalonych wirów (krzyżyki) rysują dwa skrzydła. Z prawej: x względem czasu przez pierwsze czterdzieści jednostek czasu; linie przerywane to ustalone wiry.

Zatrzymaj się i pomyśl 9.4

Służby pogodowe liczą dziś modele niosące wiele milionów liczb. Model Lorenza niesie trzy. Czy trzy wystarczą na prawdziwy chaos, czy tylko na jego szkic?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Trzy wystarczą, i trzy to najmniej, ile potrzebuje przepływ.

Rozdział 4 podał tę dolną granicę; oto powód. Na płaszczyźnie droga nigdy nie może przeciąć samej siebie, bo dwie drogi przez jeden punkt miałyby tę samą przyszłość; przepływ z dwiema liczbami może więc tylko osiąść, zwijać się spiralą albo krążyć po stałej pętli. Z trzecią liczbą droga może przechodzić nad sobą i pod sobą, i to wystarcza.

Pułapka

„Chaos potrzebuje wielu zmiennych.” Potrzebuje trzech liczb i dwóch mnożeń. Pogodę trudno przewidzieć nie tylko dlatego, że atmosfera jest wielka: garnek wody okrojony do trzech liczb jest już nieprzewidywalny. Czego chaos naprawdę potrzebuje, to temat rozdziału 10.

Gdzie to spotkasz

Chaotyczne koło wodne to garnek Lorenza zbudowany z drewna i wody: ciekące kubki wiszą na obwodzie koła, a woda wlewa się do kubków na górze. Willem Malkus i Louis Howard zbudowali takie koło w MIT, a jego równania, po przemianowaniu wielkości, są równaniami Lorenza: koło kręci się, zwalnia i zawraca w chwilach, których nikt nie potrafi zapowiedzieć. Równania pewnego rodzaju lasera też są równaniami Lorenza, co pokazał Hermann Haken w 1975 roku.

9.3 Dwa starty o włos od siebie

Rozdział 6 zmierzył wrażliwość na odwzorowaniu. Oto ten sam pomiar na przepływie. Przebieg najpierw idzie przez pięćdziesiąt jednostek czasu, żeby zostawić start za sobą; potem zostaje skopiowany, a kopia przesunięta o jedną stumilionową w x .

code/ch09/two_runs.py

```

14 DT = 0.01
15 field = lorenz()
16 a = integrate(field, (1.0, 1.0, 1.0), DT, 5000)[-1] # fifty time units
17 b = (a[0] + 1e-8, a[1], a[2]) # the copy, 0.00000001 further in x

```

Listing 9.3: Dwa przebiegi o jedną stumilionową od siebie

Oba są prowadzone tą samą regułą, a program co dwie jednostki czasu wypisuje x każdego z nich i odległość między nimi:

time	x of A	x of B	apart
0	-3.66	-3.66	1.0e-08
2	-2.90	-2.90	6.9e-08
4	-5.28	-5.28	1.1e-06
6	-0.62	-0.62	1.4e-06
8	-15.34	-15.34	4.8e-05
10	2.80	2.80	2.0e-04
12	2.39	2.39	4.6e-04
14	-11.84	-11.84	5.8e-03
16	-2.13	-2.11	3.1e-02
18	3.74	3.62	2.5e-01
20	9.63	9.01	1.2e+00
22	-1.57	-0.08	4.2e+00
24	-12.07	8.35	2.6e+01

first more than 1 apart at time 19.2

Do chwili 14 oba przebiegi zgadzają się co do dwóch miejsc po przecinku; w chwili 24 są na różnych skrzydłach. Odległość rośnie mniej więcej dziesięciokrotnie co dwie, trzy jednostki czasu, choć nie gładko: puchnie i kurczy się, gdy para obiega pętlę. Pierwszy raz przekracza 1 w chwili 19,2.

Rozdział 7 nazwał to tempo wykładnikiem Lapunowa, λ (lambda): to średnia logarytmu tego, jak bardzo rozciąga się mała szczelina. Dla przepływu średnia liczona jest na jednostkę czasu. Biblioteka mierzy go tak:

code/src/chaoslab/measure.py

```

77 def lyapunov_flow(field: Field, state: State, dt: float, steps: int,
78                  delta: float = 1e-8, burn: int = 1000,
79                  stepper: Stepper = rk4_step) -> float:
80     """The largest Lyapunov exponent of a flow, per unit of time.
81
82     Follow two runs delta apart; after every step measure how far apart
83     they have drifted, add the log of the growth, and pull the second run
84     back to distance delta along the same direction. Pulling back is what
85     keeps the gap small enough to measure the stretching rather than the
86     size of the attractor.

```

```

87
88     for _ in range(burn):
89         state = stepper(field, state, dt)
90     other = (state[0] + delta,) + tuple(state[1:])
91     total = 0.0
92     for _ in range(steps):
93         state = stepper(field, state, dt)
94         other = stepper(field, other, dt)
95         d = _distance(state, other)
96         total += math.log(d / delta)
97         other = tuple(s + (o - s) * delta / d
98                       for s, o in zip(state, other, strict=True))
99     return total / (steps * dt)

```

Listing 9.4: Wykładnik Lapunowa przepływu

Prowadź dwa przebiegi o włos od siebie. Po każdym kroku dodaj logarytm tego, o ile urosła szczelina między nimi, a potem ściągnij drugi przebieg z powrotem na pierwotną małą odległość, w tym samym kierunku, żeby szczelina nigdy nie urosła do rozmiarów motyla, gdzie przestałaby mierzyć rozciąganie. Suma podzielona przez upływ czasu to λ . Oto on z trzech startów:

code/ch09/exponent.py

```

16     field = lorenz()
17     for start in STARTS:
18         lam = lyapunov_flow(field, start, 0.01, 50_000) # 500 time units
19         print(f"from {start}: {lam:.2f} per time unit")

```

Listing 9.5: Wykładnik przepływu Lorenza z trzech startów

```

from (1.0, 1.0, 1.0): 0.92 per time unit
from (-5.0, 3.0, 20.0): 0.91 per time unit
from (10.0, -10.0, 30.0): 0.91 per time unit

```

Od 0,91 do 0,92 na jednostkę czasu, z trzech zupełnie różnych startów: przyjmijmy 0,9. Średnio szczelina mnoży się przez e do potęgi 0,9, czyli około 2,5, na każdą jednostkę czasu.

Zatrzymaj się i pomyśl 9.5

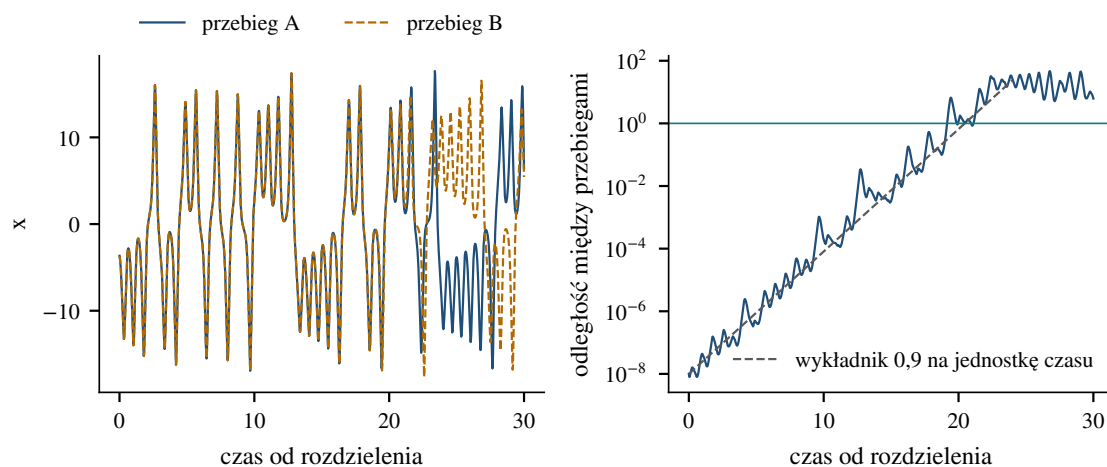
Horyzont z rozdziału 7: błąd rośnie do tolerancji w czasie około $\ln(\text{tolerancja}/\text{błąd})/\lambda$. Przy błędzie 10^{-8} , tolerancji 1 i $\lambda = 0,9$: jak długo dwa przebiegi powinny pozostać bliżej niż 1 od siebie? Potrzebujesz $\ln(10^8)$, czyli około 18,4.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około $18,4/0,9$, czyli mniej więcej 20 jednostek czasu.
Para powyżej potrzebowała 19,2.

Jedna para to jedna próbka. Rozdziel przebieg w 50 różnych miejscach wzdłuż motyla, a

pierwsze przekroczenie 1 wypada gdziekolwiek od chwili 16 do chwili 25, średnio w 20. Wzór przewiduje średnią; rozrzut bierze się stąd, że rozciąganie nie jest takie samo na całych skrzydłach.



Rysunek 9.3: Dwa przebiegi, które startują o jedną stumilionową od siebie. Z lewej: ich x zgadzają się, a potem się rozchodzą. Z prawej: ich odległość w skali logarytmicznej wspina się wzdłuż przerywanej linii wykładnika, aż osiągnie rozmiar motyla.

W regule nie ma nic niepewnego, a mimo to prognoza od startu znanego z dokładnością do ośmiu miejsc po przecinku wystarcza na mniej więcej dwadzieścia jednostek czasu. (Dwa przebiegi, które się rozeszły, można znowu zmusić do zgody, pozwalając jednemu czuć drugi: to opowieść rozdziału 17.)

9.4 Nieprzewidywalny, ale nie gdziekolwiek

Jeśli przebiegu nie da się prognozować dalej niż na dwadzieścia jednostek czasu, co w ogóle można o nim powiedzieć?

Zatrzymaj się i pomyśl 9.6

W chwili 24 dwa przebiegi powyżej nie mają ze sobą nic wspólnego: jeden ma x w pobliżu -12 , drugi w pobliżu 8 . Puść je jeszcze na tysiąc jednostek czasu. Czy któryś z nich może się znaleźć w $x = 100$ albo w $x = -50$?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Przez 20 000 jednostek czasu x nie oddala się od zera bardziej niż o 20 w żadną stronę, a z pozostaje między 1 a 48. I tak jest niezależnie od tego, skąd wystartujesz.

Sprawdź to „niezależnie od tego, skąd” na ośmiu startach w narożnikach pudła o boku osiemdziesiąt, daleko poza wszystkim, co widać na rysunkach powyżej:

code/ch09/cloud.py

```

12 field = lorenz()
13 corners = [(x, y, z) for x in (-40.0, 40.0) for y in (-40.0, 40.0)
14             for z in (-40.0, 80.0)]

```

Listing 9.6: Osiem startów daleko poza motylem

Program prowadzi każdy przez pięćdziesiąt jednostek czasu, pomija pierwsze pięć i wypisuje pokój, w którym żył każdy przebieg:

start	largest x	lowest z	highest z
(-40, -40, -40)	18.7	3.8	46.0
(-40, -40, 80)	18.2	5.2	45.0
(-40, 40, -40)	18.4	4.7	45.4
(-40, 40, 80)	17.7	6.5	44.1
(40, -40, -40)	18.4	4.7	45.4
(40, -40, 80)	17.7	6.5	44.1
(40, 40, -40)	18.7	3.8	46.0
(40, 40, 80)	18.2	5.2	45.0

Każdy przebieg zostaje ściągnięty ze swojego narożnika i potem mieszka w tym samym pokoju co długi przebieg. Zauważ też, że wiersze idą w identycznych parach: $(-40, -40, -40)$ i $(40, 40, -40)$ dają te same liczby. Następny podrozdział mówi dlaczego.

Pułapka

„Układ chaotyczny może pójść gdziekolwiek.” Nieprzewidywalny to nie to samo co nieograniczony. Nie powiesz, w którym miejscu motyla będzie przebieg Lorenza za sto jednostek czasu, ale powiesz, że będzie na motylu, i to jest prawdziwe przewidywanie: cienki zbiór stanów spośród wszystkich, jakie mogłyby przyjąć trzy liczby.

Zbiór stanów, na który przebiegi zostają ściągnięte i którego potem nigdy nie opuszczają, to *atraktor*. Wahadło z tarciami ma za atraktor jeden punkt; wahadło zegara, podtrzymywane w ruchu, ma pętlę. Atraktorem Lorenza jest motyl, ten sam z niemal każdego startu, a przebieg na nim nigdy się nie powtarza i nigdy go nie opuszcza. Jak droga może bez końca krążyć w ograniczonym obszarze, nie przecinając się i nie wracając po własnych śladach, to zagadka dotycząca samego kształtu – dlatego nazywa się go *atraktorem dziwnym*. Rozdział 10 otwiera ten kształt, a rozdział 11 mierzy kształty takie jak on.

Czego tu nie dowodzimy

Dwie rzeczy są tu pokazane przez uruchomienie, a nie udowodnione. To, że żaden przebieg nie ucieknie, da się udowodnić na stronie algebry o wielkości, która maleje zawsze, gdy stan jest daleko. To, że motyl narysowany przez komputer naprawdę istnieje i naprawdę jest chaotyczny, a nie jest wytworem kroków i zaokrągleń, pozostawało otwarte przez ponad trzydzieści lat, aż Warwick Tucker rozstrzygnął to dowodem, w którym komputer liczył tak, że każdy błąd zaokrąglenia był wzięty pod

uwagę.

9.5 Co wciąż da się przewidzieć

Motyl mówi, gdzie przebieg mieszka. Następne pytanie brzmi: jak spędza tam czas.

Zatrzymaj się i pomyśl 9.7

Nie powiesz, na którym skrzydle będzie przebieg w chwili 50. Czy powiesz, jaką część długiego przebiegu spędza on na prawym skrzydle, gdzie x jest dodatnie?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tak: połowę. Zmień jednocześnie znaki x i y : pierwsze dwa tempa zmieniają znak razem z nimi, a trzecie, $xy - \beta z$, wcale się nie zmienia, bo $(-x)(-y) = xy$.

Reguły nie dbają więc o to, w którą stronę kręci się wir. Każdy przebieg na jednym skrzydle ma lustrzane odbicie na drugim – dlatego wiersze chmury szły parami – i długi przebieg dzieli swój czas między skrzydła po równo.

Pułapka

„Jeśli droga jest nieprzewidywalna, nic w niej nie da się przewidzieć.” Droga jest nieprzewidywalna dalej niż na mniej więcej dwadzieścia jednostek czasu. Kształt, na którym mieszka, już nie, podobnie jak to, ile czasu spędza w każdej jego części. To są *statystyki atraktora* i z każdego startu wychodzą takie same.

Laboratorium 9.2 – Dwie liczby, o których start nie decyduje

Napisz `fraction_right`, czyli udział stanów przebiegu z x większym od zera, oraz `mean_height`, czyli średnią jego z . Test prowadzi reguły Lorenza od $(1, 1, 1)$ i od dalekiego narożnika chmury i sprawdza, że twoje dwa podsumowania zgadzają się między przebiegami.

Plik startowy: `code/labs/ch09/l09_02_wings.py`

Test: `code/labs/ch09/test_l09_02_wings.py`

Uruchom: `cd code && uv run pytest -k l09_02_wings`

Oto wersje z książki:

code/ch09/averages.py

```

17 def fraction_right(path: list[tuple[float, ...]]) -> float:
18     """The share of the states with x positive: time on the right wing."""
19     return sum(1 for x, _, _ in path if x > 0) / len(path)
20
21
22 def mean_height(path: list[tuple[float, ...]]) -> float:
23     """The average of z over the run: how high the state lives."""
24     return sum(z for _, _, z in path) / len(path)

```

Listing 9.7: Dwie statystyki przebiegu

Listing prowadzi pięć startów, każdy przez 500 jednostek czasu, odrzuca pierwsze dziesięć, żeby start został zapomniany, i dla każdego przebiegu wypisuje, na którym skrzydle jest w chwili 50, jaką część czasu spędził na prawym skrzydle i jaka jest jego średnia wysokość:

start	wing at 50	time on right	mean height
(1, 1, 1)	L	0.50	23.48
(-5, 3, 20)	L	0.49	23.52
(10, -10, 30)	R	0.50	23.50
(0, -1, 50)	L	0.49	23.51
(30, -30, 5)	R	0.50	23.57

Środkowa kolumna to część nieprzewidywalna: lewe, lewe, prawe, lewe, prawe, i nic, co mógłbyś wiedzieć zawczasu, tego nie wybiera. Ostatnie dwie kolumny to część przewidywalna. Udział prawego skrzydła wynosi od 0,49 do 0,50 z każdego startu, tak jak zapowiadało lustro. Średnia wysokość wynosi od 23,48 do 23,57, a tej liczby nie podpowiedziała żadna symetria: po prostu jest taka sama z każdego startu.

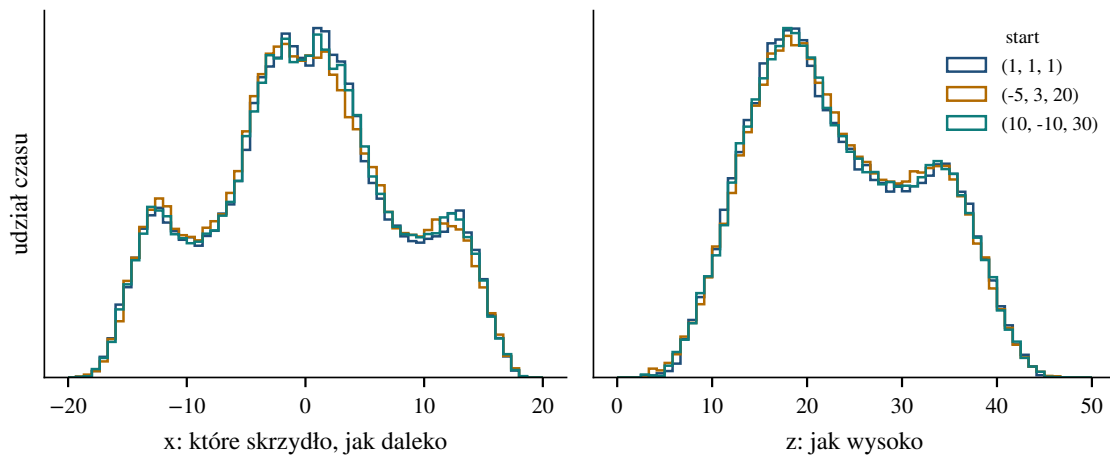
Statystyka skończonego przebiegu trochę się chwieje. Potnij jeden przebieg długości 20 000 jednostek czasu na kawałki: w kawałkach po 100 jednostek czasu udział prawego skrzydła odbiega zwykle od połowy o około 0,05; w kawałkach po 2000 – o około 0,01; w całym przebiegu wynosi 0,50. Im dłużej patrzysz, tym ostrzejsza statystyka – odwrotnie niż droga, której prognoza jest tym gorsza, im dalej sięgasz.

Cały histogram tego, gdzie przebieg spędza czas, wychodzi taki sam z trzech startów. To oś tego rozdziału i druga wielka myśl tej książki: **indywidualna przyszłość układu chaotycznego jest nieprzewidywalna, ale to, gdzie układ mieszka i jak często odwiedza każdą część, da się przewidzieć**. Nawet λ jest liczbą tego rodzaju.

Rozdział 14 buduje właśnie na tym rozróżnienie między pytaniem o pogodę na przyszły tydzień a pytaniem o klimat i pokazuje, jak synoptycy liczą wiele kopii naraz. Rozdział 16 pyta, dlaczego można ufać statystyce, choć własne zaokrąglenia komputera szybko czynią jego drogę inną niż prawdziwa.

Podsumowanie

- Model Lorenza z 1963 roku, opisujący podgrzewany garnek, zachowuje trzy liczby: prędkość wiru x , różnicę temperatur y i wymieszanie z . Tylko dwa wyrazy jego trzech **reguł zmiany**, xy i xz , są **nieliniowe**. Jego **punkty stałe** to nieruchomy garnek i dwa **ustalone wiry**.



Rysunek 9.4: Ile czasu przebieg spędza przy każdej wartości x i z , z trzech startów, po 500 jednostek czasu każdy. Trzy obrysy leżą jeden na drugim: histogram nie dba o to, skąd przebieg wystartował.

- Przy $\rho = 28$ przebieg nie osiada w żadnym punkcie stałym: krąży wokół jednego ustalonego wiru, potem wokół drugiego, przeskakując nieregularnie i nigdy się nie powtarzając. Trzy liczby to najmniej, ile **przeptyw** potrzebuje do chaosu, i wystarczą.
- Dwa przebiegi o jedną stumilionową od siebie rozchodzą się w mniej więcej dwadzieścia jednostek czasu. **Wykładnik Lapunowa** przepływu to około 0,9 na jednostkę czasu, a wzór na horyzont z rozdziału 7 przewiduje średni czas rozejścia.
- Każdy przebieg zostaje ściągnięty na ten sam ograniczony zbiór w kształcie motyla: **atraktor**. Układ chaotyczny nie może pójść, gdzie zechce.
- **Statystyki atraktora**, takie jak część czasu na każdym skrzydle i średnia wysokość, są takie same z każdego startu. Droga jest nieprzewidywalna; to, gdzie układ mieszka, nie.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

przeczytać trzy reguły zmiany Lorenza wyraz po wyrazie i powiedzieć, co każdy wyraz robi z wirem	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
prowadzić reguły metodą RK4 i odczytać wynik z dwóch widoków: jedna liczba względem drugiej oraz jedna liczba względem czasu	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zmierzyć, jak szybko rozchodzą się dwa prawie identyczne przebiegi, i zamienić to tempo na horyzont przewidywania	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, czym jest atraktor i dlaczego układ chaotyczny nie może pójść, gdzie zechce	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
policzyć statystykę przebiegu i sprawdzić, że nie zależy od tego, skąd przebieg wystartował	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Co słowami robi z wirem reguła na tempo x , $\sigma(y - x)$, gdy y jest większe od x ?
2. Gdzie przy $\rho = 28$ leżą trzy punkty stałe reguł Lorenza?
3. Dlaczego przepływ z dwiema liczbami nie może być chaotyczny i ile liczb używa model Lorenza?
4. Wykładnik to około 0,9 na jednostkę czasu. Gdyby start był znany tysiąc razy dokładniej, o ile dłużej dwa przebiegi pozostałyby bliżej niż 1 od siebie?
5. Ktoś twierdzi, że przebieg Lorenza, skoro jest chaotyczny, może kiedyś dojść do $z = 200$. Co mu odpowiesz?
6. Jaką część długiego przebiegu zajmuje prawe skrzydło i skąd możesz to wiedzieć, niczego nie uruchamiając?
7. Podaj liczbę opisującą przebieg Lorenza, której nie przewiduje żadna symetria, a która z każdego startu wychodzi taka sama.

Rozciągnij i złóż: dziwne atraktory

W rozdziale 9 dwa przebiegi równań Lorenza, które zaczęły się o włos od siebie, rozjechały się tak, że nie miały już ze sobą nic wspólnego, a mimo to żaden z nich nigdy nie opuścił obszaru w kształcie motyla, w którym się zaczął. Postaw te dwa fakty obok siebie, a wydadzą się niemożliwe. Jeśli sąsiednie stany są w każdym kroku od siebie odpychane, dlaczego nie lądują na przeciwnych krańcach wszechświata? A jeśli wszystko zostaje w małym obszarze, gdzie podziwiewa się całe to oddalanie?

Odpowiedź każdy piekarz ma w rękach. Ten rozdział pokazuje ją najpierw na pasku ciasta, potem na dwuwierszowej regule, którą astronom Michel Hénon zapisał w 1976 roku, a na końcu prowadzi do kształtu bez pola, bez końca długości i z warstwami wewnątrz warstw, jakkolwiek blisko byś się przyglądał.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- wyjaśnić, jak ruch może na zawsze pozostawać w pewnym obszarze, a mimo to być wrażliwy na stan początkowy, używając słów rozciąganie i składanie
- iterować odwzorowanie namiotowe i odwzorowanie Hénona w Pythonie i czytać to, co wypisują
- policzyć, przez jaki czynnik jeden krok odwzorowania Hénona mnoży pole, i sprawdzić to pomiarem
- powiedzieć, czym są atraktor, jego basen przyciągania i atraktor dziwny
- opisać, co pokazuje powiększanie atraktora dziwnego i czym różni się to od powiększania gładkiej krzywej
- opisać, jak przekrój Poincarégo zamienia przepływ w odwzorowanie

10.1 Rozciągnij, potem złóż

Połóż kawałek ciasta na stolnicy i wciśnij w nie obok siebie dwie rodzyнки. Teraz zagniataj: rozwałkuj ciasto do podwójnej długości, a potem przełóż dalszą połowę na bliższą. I jeszcze raz, i jeszcze. Po minucie rodzyńki są daleko od siebie, a żadna nie opuściła stolnicy.

Jedno zagniecenie to dwa ruchy i to one są treścią całego rozdziału. *Rozciąganie* oddala od siebie każdą parę sąsiadów. *Składanie* umieszcza rozciągnięte ciasto z powrotem w miejscu, z którego przyszło. Samo rozciąganie wyniosłoby rodzyńki poza stolnicę; samo składanie nic by im nie zrobiło. Razem robią to, czego żadne z nich nie potrafi.

Żeby zamienić to w arytmetykę, spłaszcz ciasto w pasek biegnący od 0 do 1 i niech x

oznacza położenie rodzyнки na nim. Rozciąganie przenosi x w $2x$. Wszystko, co wystaje teraz poza 1, zostaje złożone z powrotem, więc punkt, który wystawał do $2x$, ląduje w $2 - 2x$, tyle przed końcem, ile przedtem był za nim:

$$x_{n+1} = 2x_n \text{ jeśli } 2x_n \leq 1, \quad x_{n+1} = 2 - 2x_n \text{ w przeciwnym razie,}$$

gdzie x_n to położenie rodzyнки po n zagnieceniach. Wykres tej reguły wspina się prostą linią od 0 do 1 i schodzi prosto w dół jak namiot widziany z boku, dlatego nazywa się ją *odwzorowaniem namiotowym*.

Zatrzymaj się i pomyśl 10.1

Dwie rodzyнки zaczynają w 0,2345 i 0,2355, jedną tysięczną paska od siebie. Dopóki obie są po tej samej stronie zgięcia, każde zagniecenie podwaja odstęp między nimi. Gdyby podwajał się dalej, po ilu zagnieceniach byłby dłuższy niż cały pasek? Co musi się stać zamiast tego?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Po dziesięciu: $0,001 \times 2^{10} = 1,024$, więcej niż długość paska. To niemożliwe: dwie rodzyнки na pasku długości jeden nigdy nie są od siebie dalej niż o jeden, więc wcześniej zgięcie musi złapać jedną rodzyńkę, a drugiej nie.

code/ch10/kneading.py

```

12 def knead(x: float) -> float:
13     """One knead: stretch the strip to twice its length, then fold."""
14     stretched = 2.0 * x
15     if stretched <= 1.0:
16         return stretched          # still on the strip: nothing to fold
17     return 2.0 - stretched        # past the end: fold it back
18
19
20 def main() -> None:
21     a, b = 0.2345, 0.2355          # two raisins a thousandth apart
22     print("knead  raisin A  raisin B    gap")
23     for n in range(14):
24         print(f"{n:5d}  {a:8.4f}  {b:8.4f}  {abs(b - a):6.4f}")
25         a, b = knead(a), knead(b)

```

Listing 10.1: Zagniatanie paska ciasta: rozciągaj, potem złóż

knead	raisin A	raisin B	gap
0	0.2345	0.2355	0.0010
1	0.4690	0.4710	0.0020
2	0.9380	0.9420	0.0040
3	0.1240	0.1160	0.0080
4	0.2480	0.2320	0.0160
5	0.4960	0.4640	0.0320
6	0.9920	0.9280	0.0640
7	0.0160	0.1440	0.1280
8	0.0320	0.2880	0.2560
9	0.0640	0.5760	0.5120
10	0.1280	0.8480	0.7200
11	0.2560	0.3040	0.0480
12	0.5120	0.6080	0.0960
13	0.9760	0.7840	0.1920

Odstęp podwaja się dziewięć razy z rzędu, do 0,512. Przy dziesiątym zagnieceniu jedna rodzynka zostaje złożona, a druga nie, i odstęp wynosi 0,72. Zagniecenie później dzieli je tylko 0,048, przypadkiem: rozrzucone rodzynki mogą być w dowolnej odległości, a od tej chwili to, gdzie jest jedna, nic ci nie mówi o drugiej. Każde zagniecenie mnoży każdy krótki odstęp dokładnie przez 2, więc wykładnik Lapunowa tej reguły, w sensie rozdziału 7, wynosi $\ln 2 \approx 0,69$ na zagniecenie.

Pułapka

„Jeśli układ nigdy nie może opuścić małego obszaru, nie może być bardzo wrażliwy na stan początkowy.” Obszar ogranicza to, jak *duży* może być błąd, a nie to, jak szybko rośnie. Małe błędy podwaja się raz za razem, aż dorówna obszarowi, a od tej chwili dwa przebiegi są tak niezależne jak dwie rodzynki rzucone na stolnicę na chybił trafił. Pozostawanie w misce i przewidywalność to dwie różne obietnice, a składanie pozwala układowi dotrzymać pierwszej bez drugiej.

Laboratorium 10.1 – Pasek ciasta

Napisz knead, jedno rozciągnięcie i jedno złożenie. Potem napisz kneads_until_apart, które liczy zagniecenia, zanim dwie rodzynki startujące w małym odstępnie znajdują się dalej od siebie niż zadana tolerancja. Dla rodzynek oddalonych o jedną tysięczną i tolerancji 0,1 odpowiedź brzmi siedem. Zaczynij je dziesięć razy bliżej i zobacz, jak niewiele dodatkowych zagnieceń to daje.

Plik startowy: code/labs/ch10/l10_01_knead.py

Test: code/labs/ch10/test_l10_01_knead.py

Uruchom: `cd code && uv run pytest -k l10_01_knead`

10.2 Dwuwierszowe odwzorowanie Hénona

Pasek to jedna liczba. Układ Lorenza z rozdziału 9 ma trzy i rozciąga oraz składa, gdy trajektoria okrąży jeden z płatów, odrywa się od sąsiadów i zostaje rzucona z powrotem

do środka. W 1976 roku Michel Hénon, astronom z obserwatorium w Nicei, opublikował regułę niemal tak prostą, jak to tylko możliwe, która robi to samo z płaszczyzną – tak prostą, że komputer może wykonać miliony jej kroków. Przenosi ona punkt (x, y) , czyli stan złożony z dwóch liczb, w nowy punkt:

$$x_{n+1} = 1 - a x_n^2 + y_n, \quad y_{n+1} = b x_n,$$

gdzie $a = 1,4$, a $b = 0,3$. Słowami: nowe x to jeden minus a razy kwadrat starego x plus stare y ; nowe y to stare x zmniejszone do trzech dziesiątych.

Hénon złożył ją z trzech ruchów. *Wygięcie* przesuwają każdy punkt w górę lub w dół o $1 - ax^2$, garb, który wysyła x i $-x$ na tę samą wysokość, więc prosta biegnąca przez płaszczyznę zostaje zagięta w łuk: to jest składanie. *Ścisnięcie* spłaszcza płaszczyznę na boki do trzech dziesiątych jej szerokości. *Zamiana* zamienia miejscami obie liczby, więc następne wygięcie działa w poprzek poprzedniego.

Zatrzymaj się i pomyśl 10.2

Zacznij w początku układu współrzędnych, $(0, 0)$. Gdzie jest punkt po jednym kroku odwzorowania Hénona, a gdzie po dwóch?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Po jednym kroku $(1, 0)$. Po dwóch

$x = 1 - 1,4 \times 1 + 0 = -0,4$ i $y = 0,3 \times 1$, więc punkt to $(-0,4, 0,3)$.

code/ch10/henon.py

```
11 A = 1.4 # how hard each step bends
12 B = 0.3 # how hard each step squeezes
13
14
15 def henon(x: float, y: float) -> tuple[float, float]:
16     """One step of Henon's map."""
17     return 1.0 - A * x * x + y, B * x
```

Listing 10.2: Odwzorowanie Hénona: dwie liczby, po jednym wierszu arytmetyki na każdą

step	x	y
0	0.0000	0.0000
1	1.0000	0.0000
2	-0.4000	0.3000
3	1.0760	-0.1200
4	-0.7409	0.3228
5	0.5543	-0.2223
6	0.3476	0.1663
7	0.9972	0.1043
8	-0.2879	0.2992

Listing ciągnie dalej tam, gdzie zatrzymała się twoja ręka. Punkt skacze, nie uspokajając się, nie powtarzając i nie uciekając: jakkolwiek długo go puścisz, x pozostaje między $-1,28$ a $1,27$.

10.3 Jeden kształt dla każdego startu

Jeden start to jedna historia. Weź ich wiele: siatkę czterdzieści na czterdzieści, czyli 1600 punktów startowych rozłożonych równo w kwadracie, w którym zarówno x , jak i y biegają od $-0,5$ do $0,5$.

Zatrzymaj się i pomyśl 10.3

Przepuść każdy z tych punktów przez trzydzieści kroków odwzorowania Hénona. Gdzie spodziewasz się zastać chmurę: wciąż rozłożoną po kwadracie, zebraną w jednym punkcie, czy gdzie indziej?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Gdzie indziej. Kilka punktów ucieka na zawsze, a wszystkie pozostałe leżą na jednym cienkim, wygiętym zbiorze – tym samym, który rysuje pojedynczy start, jeśli puścić go dość długo. Nie zbierają się w punkcie, bo rozciąganie trzyma je z dala od siebie wzdłuż tego zbioru.

code/ch10/cloud.py

```

24 def marked_squares() -> set[tuple[int, int]]:
25     """Every little square that one long run of the map visits."""
26     x, y = 0.0, 0.0
27     for _ in range(1000):                # let it settle first
28         x, y = henon(x, y)
29     marked = set()
30     for _ in range(REFERENCE):
31         x, y = henon(x, y)
32         marked.add(square_of(x, y))
33     return marked
34
35
36 def main() -> None:
37     marked = marked_squares()
38     cloud = [(-0.5 + (i + 0.5) / 40, -0.5 + (j + 0.5) / 40)
39             for i in range(40) for j in range(40)]
40     start = len(cloud)
41     print("step left for good on the attractor")
42     for step in range(31):
43         if step in (0, 1, 2, 3, 5, 10, 20, 30):
44             on = sum(square_of(x, y) in marked for x, y in cloud)
45             print(f"{step:4d} {start - len(cloud):13d} "
46                   f"{on:6d} of {len(cloud)}")
47         cloud = [henon(x, y) for x, y in cloud]
48         cloud = [(x, y) for x, y in cloud if abs(x) + abs(y) < 10]

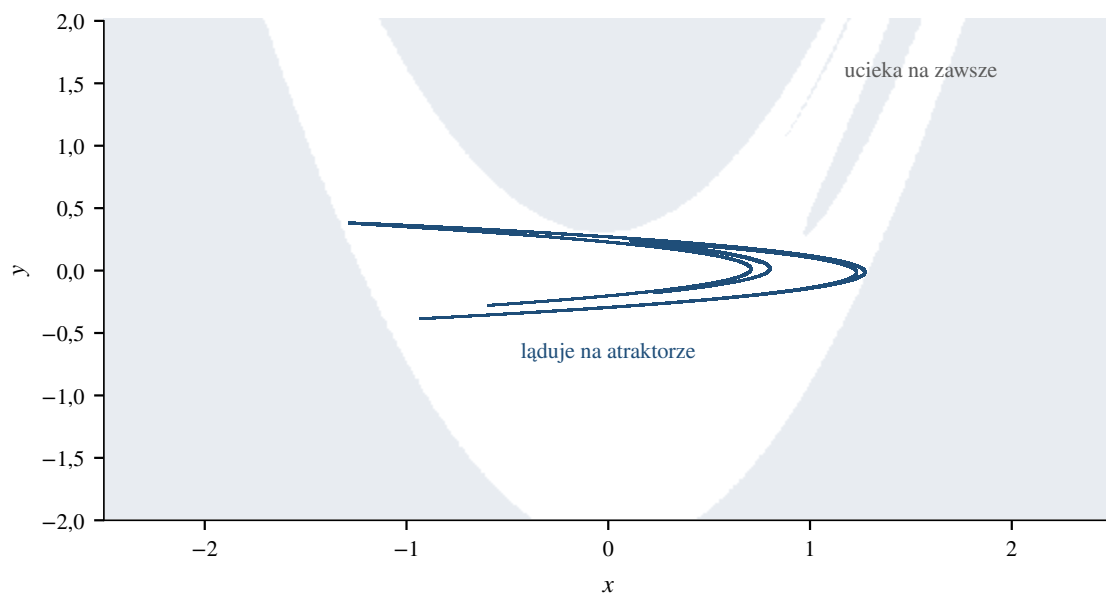
```

Listing 10.3: Chmura startów i to, czy każdy ląduje na atraktorze

step	left for good	on the attractor
0	0	104 of 1600
1	0	237 of 1600
2	0	811 of 1600
3	0	1297 of 1600
5	139	1409 of 1461
10	172	1428 of 1428
20	172	1428 of 1428
30	172	1428 of 1428

Żeby rozstrzygnąć, czy punkt leży na tym zbiorze, program najpierw puszcza jeden start na 500 000 kroków i zaznacza każdy kwadracik o boku 0,01, który ten start odwiedzi. Na początku tylko 104 z 1600 punktów leży w zaznaczonym kwadraciku. Od kroku 10 leżą w nich wszystkie 1428, które zostały, a 172 uciekło na zawsze, poza dziesiątkę i dalej w nieskończoność.

To właśnie znaczy słowo *atraktor*: zbiór stanów, na którym ruch osiada i którego już nie opuszcza, a który przyciąga stany dookoła. Punkt spoczynku tłumionego wahadła z rozdziału 4 to najprostszy atraktor; motyl z rozdziału 9 to kolejny. Każdy start, który kończy na atraktorze, należy do jego *basenu przyciągania*, tak jak każda kropla deszczu, która dociera do jeziora, spada w dorzeczu tego jeziora. Atraktor Hénona ma basen z brzegiem, a starty spoza niego uciekają w nieskończoność.



Rysunek 10.1: Atraktor Hénona (niebieski) na tle swojego basenu. Każdy start z białego obszaru kończy na atraktorze; każdy start z szarego obszaru ucieka na zawsze. Górna krawędź kwadratu z listingu sięga w szarość i stamtąd pochodzą starty, które uciekły.

10.4 Ściśnięty, nie zatrzymany

Chmura zaczęła jako kwadrat o pewnym polu, a skończyła na zbiorze cienkim jak linia. Reguła Hénona mówi dokładnie, jak szybko pole znika. Weź mały kwadrat o boku h z wierzchołkiem w (x, y) . Jeden krok przenosi jego dwa boki, strzałki $(h, 0)$ i $(0, h)$, w dwie nowe strzałki. Przesunięcie x o h zmienia nowe x o mniej więcej $-2axh$, bo $(x+h)^2 = x^2 + 2xh + h^2$, a h^2 jest maleńkie w porównaniu z h , i zmienia nowe y o bh . Przesunięcie y o h zmienia nowe x o h , a nowego y wcale. Kwadrat staje się więc pochylonym równoległobokiem o bokach

$$(p, q) = (-2axh, bh), \quad (r, s) = (h, 0).$$

Pole równoległoboku o bokach (p, q) i (r, s) to wartość bezwzględna liczby $ps - qr$, zwanej *wyznacznikiem*; dla prostokąta o bokach $(2, 0)$ i $(0, 3)$ daje ona $2 \times 3 - 0 \times 0 = 6$. Tutaj wynosi $(-2axh) \times 0 - (bh) \times h = -bh^2$, więc pole h^2 zamienia się w bh^2 . Każdy krok mnoży każde pole przez $b = 0,3$, gdziekolwiek był kwadrat: położenie x , od którego zależy, jak mocno kwadrat się rozciąga, zostaje pomnożone przez zero i znika z wyniku.

Zatrzymaj się i pomyśl 10.4

Plama punktów startowych ma pole 1. Jakie ma pole po dziesięciu krokach? Czy to znaczy, że jej punkty zwalniają i stają, jak tłumione wahadło z rozdziału 4?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około 0,0000059, czyli $0,3^{10}$: sześć milionowych tego, co było. I nie, punkty wcale nie zwalniają. Z plamy wyciskane jest pole, nie ruch.

Pułapka

„Skoro pole wciąż maleje, ruch musi zamierać.” Tłumione wahadło traci pole, bo wszystko jest ściągane do jednego punktu spoczynku. Odwzorowanie Hénona mocno ściska w jednym kierunku, a rozciąga w innym, więc plama staje się coraz dłuższą i cieńszą wstęgą, której punkty poruszają się bez końca. Malejące pole mówi, że ruch kończy na czymś cienkim, a nie że się zatrzymuje.

Teraz zmierz to. Kwadrat o boku jednej dziesiątej zamienia się w szesnaście tysięcy punktów na jego brzegu; odwzorowanie przesuwa każdy z nich, a wzór sznurowadłowy, znany geodetom, mierzy pole wewnątrz nowego obrysu.

code/ch10/area.py

```

45     edge = outline(0.1, 4000)                # a square a tenth on a side
46     area0, length0 = area(edge), length(edge)
47     print("step  area / start  0.3 ** step  outline / start")
48     for step in range(1, 7):
49         edge = [henon(x, y) for x, y in edge]
50         print(f"{step:4d} {area(edge) / area0:13.5f} ")

```

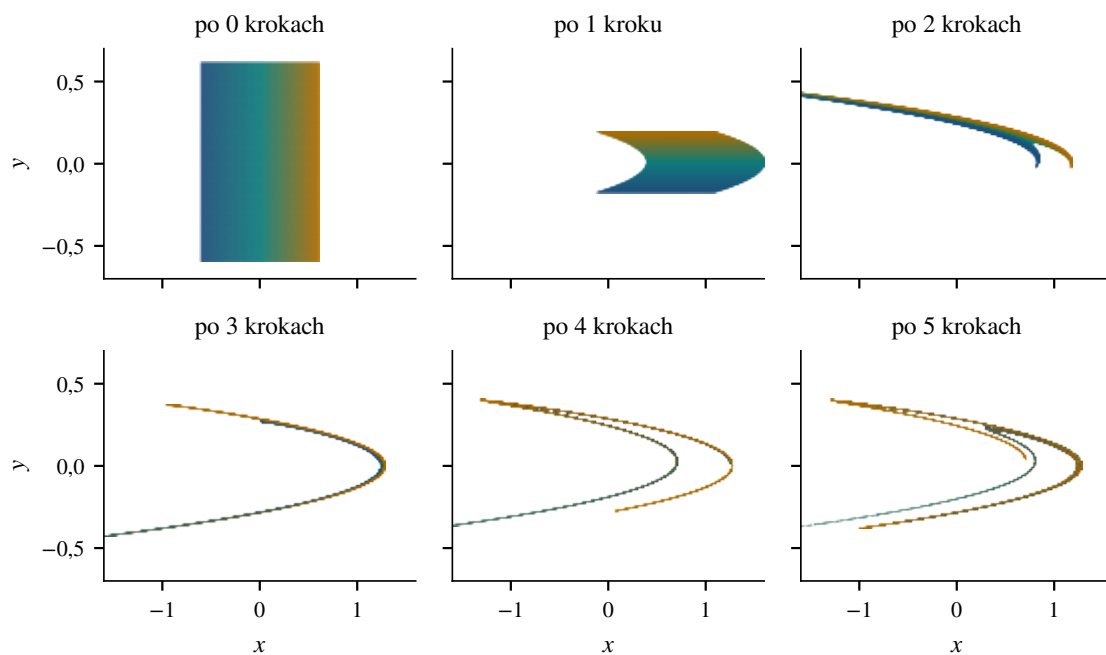
51

```
f"{B ** step:12.5f} {length(edge) / length0:16.2f}")
```

Listing 10.4: Pomiar tego, co odwzorowanie robi z polem

step	area / start	0.3 ** step	outline / start
1	0.30000	0.30000	0.66
2	0.09000	0.09000	1.56
3	0.02700	0.02700	1.63
4	0.00810	0.00810	4.13
5	0.00243	0.00243	7.58
6	0.00073	0.00073	9.81

Pole maleje o czynnik zapowiadany przez wyznacznik, z dokładnością do pięciu miejsc po przecinku, aż do 0,00073 wartości początkowej po sześciu krokach. Obrys robi coś przeciwnego: najpierw się skraca, bo ściśnięcie spłaszcza kwadrat, a potem wciąż rośnie i po sześciu krokach jest 9,8 razy dłuższy, owinięty wokół mniej niż tysięcznej części pola.



Rysunek 10.2: Kwadrat punktów startowych, pokolorowanych według tego, gdzie na osi x każdy się zaczął, po nie więcej niż pięciu krokach. Jeden krok wygina go w półksiężyc, następny rozciąga w cienki pasek, a po trzecim pasek jest złożony sam na siebie i jego kolory leżą warstwami.

Jak mocne jest rozciąganie? Nieś małą strzałkę wzdłuż jednego długiego przebiegu, przesuwając ją w każdym kroku tak, jak przesuwali się boki kwadratu. Zmierz, o ile się wydłużyła, dodaj logarytm tego wydłużenia do bieżącej sumy i skurcz ją z powrotem do długości jeden. Średnia z logarytmów to wykładnik Lapunowa z rozdziału 7, tym razem dla odwzorowania płaszczyzny.

code/ch10/stretch.py

```

38 def exponent(steps: int = 200_000) -> float:
39     """The average of log(stretch) of a tiny arrow along one long run."""
40     x, y = settle()
41     dx, dy = 1.0, 0.0                # an arrow of length one
42     total = 0.0
43     for _ in range(steps):
44         dx, dy = -2.0 * A * x * dx + dy, B * dx    # one step moves it
45         x, y = henon(x, y)
46         grow = math.sqrt(dx * dx + dy * dy)        # how much it stretched
47         total += math.log(grow)
48         dx, dy = dx / grow, dy / grow              # back to length one
49     return total / steps

```

Listing 10.5: Średnie rozciągnięcie maleńkiej strzałki

```

step 0: gap 1.0e-10
step 10: gap 3.0e-08
step 20: gap 1.5e-07
step 30: gap 4.0e-06
step 40: gap 3.5e-05
step 50: gap 3.6e-04
step 60: gap 7.4e-02
step 70: gap 2.1e+00
step 80: gap 3.1e-01
Lyapunov exponent: 0.42 per step
stretch per step: 1.52
squeeze per step: 0.20

```

Pierwsze wiersze pochodzą z drugiej połowy pliku: dwa starty oddalone o 10^{-10} , co Python zapisuje jako $1.0e-10$. Ich odstęp rośnie, aż w kroku 70 wynosi mniej więcej 2, a potem nigdy nie przekracza 2,6, czyli mniej więcej szerokości atraktora. Ograniczony, a mimo to wrażliwy, jak ciasto. Wykładnik wynosi 0,42 na krok: wzdłuż atraktora maleńka strzałka rośnie średnio o czynnik 1,52 na krok, a w poprzek odwzorowanie ściska ją o czynnik 0,20. Pomnóż jedno przez drugie, a dostaniesz z powrotem mniej więcej 0,3, czynnik pola. Rozciągnij w jedną stronę, ściśnij w drugą, złóż, żeby się zmieściło: to cały przepis.

Laboratorium 10.2 – Jeden krok i jego pole

Napisz `step`, jeden krok odwzorowania Hénona, oraz `area_factor`, które przesuwa o jeden krok trzy wierzchołki maleńkiego kwadratu, odczytuje dwa nowe boki jako strzałki (p, q) i (r, s) i zwraca wartość bezwzględną $ps - qr$ podzieloną przez stare pole. Sprawdź, że daje 0,3, gdziekolwiek postawisz kwadrat, i że zmiana b je zmienia, a zmiana a nie.

Plik startowy: code/labs/ch10/l10_02_henon.py

Test: code/labs/ch10/test_l10_02_henon.py

Uruchom: `cd code && uv run pytest -k l10_02_henon`

10.5 Warstwy w warstwach

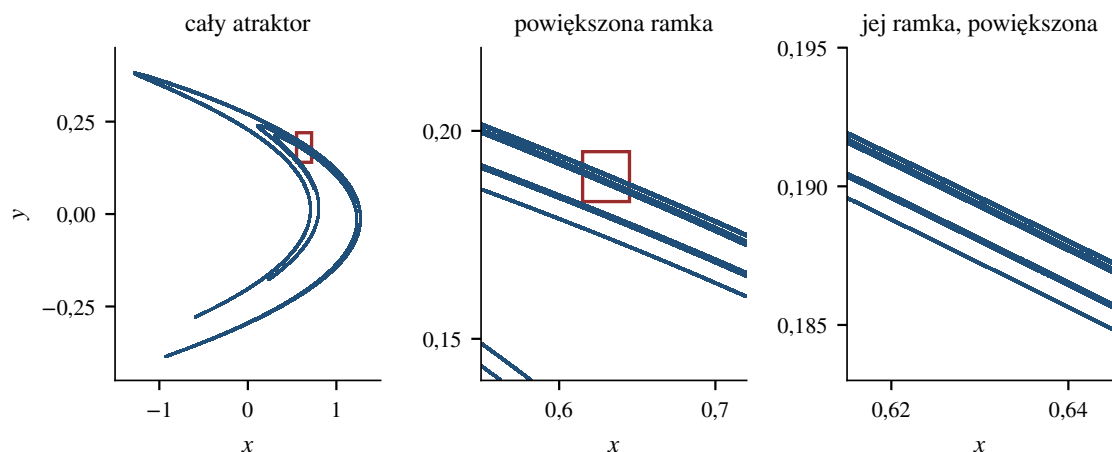
Narysowany na całą stronę atraktor Hénona wygląda jak kilka wygiętych linii. Przyjrzyj się bliżej.

Zatrzymaj się i pomyśl 10.5

Narysuj piórem dowolną gładką krzywą i powiększaj jej mały kawałek, raz za razem. Jak ten kawałek wygląda na końcu? A teraz zgadnij, co zobaczysz, gdy powiększysz jedną z linii atraktora.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Gładka krzywa coraz bardziej przypomina prostą. Linia atraktora nie: pod lupą rozpada się na kilka równoległych linii, a każda z nich, powiększona znowu, znowu się rozpada.



Rysunek 10.3: Powiększanie atraktora Hénona. Każda czerwona ramka jest pokazana w powiększeniu na następnym panelu. To, co wyglądało na jedną linię, jest wiązką linii, a każda linia tej wiązki sama jest wiązką.

Warstwy wynikają wprost z przepisu. Każdy krok rozciąga zbiór, składa go i ścisza, żeby się zmieścił, więc każde złożenie kładzie obok tego, co było, cieńszą kopię wszystkiego, a ta kopia niesie w sobie własne warstwy. Dwa skutki brzmią absurdalnie, dopóki nie zobaczysz ich na własne oczy. Atraktor nie ma pola, bo każde pole jest mnożone przez 0,3 w każdym kroku, bez końca. A jego długość nie ma końca, bo kreśli go krzywa rozciągana w każdym kroku o czynnik mniej więcej 1,52, bez końca.

Pułapka

„Wystarczy dość mocno powiększyć dowolną krzywą, a stanie się linią prostą.”

To prawda dla gładkich krzywych rysowanych piórem i dla wykresów wzorów, które poznałeś w szkole. Tutaj to nieprawda. Jakkolwiek mocno powiększysz atraktor Hénona, każda linia rozpada się na wiązkę: ta struktura nie ma najmniejszej skali.

Atraktor o takiej warstwowej budowie, na którym ruch jest chaotyczny, nazywa się *atraktorem dziwnym*; tę nazwę nadali mu David Ruelle i Floris Takens w 1971 roku. Motyl Lorenza jest takim atraktorem, a atraktor Hénona też. Taki zbiór jest czymś więcej niż linia, która miałaby skończoną długość, i czymś mniej niż pole, bo żadnego pola nie ma: jego wymiar leży między jeden a dwa, a rozdział 11 mierzy, gdzie dokładnie.

Gdzie to spotkasz

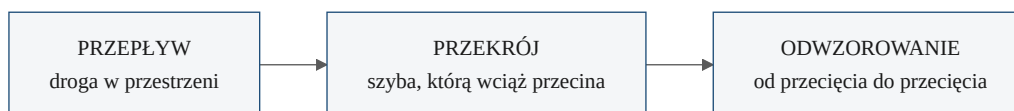
Inżynierowie chemicy mieszają gęste płyny – żywice, uszczelniacze, pasty spożywcze – w *mieszalniku statycznym*: rurze z rzędem krótkich skręconych łopatek i bez żadnych ruchomych części. Każda łopatka dzieli strumień na dwa, obraca połówki i układa je jedna na drugiej: rozciągnięcie i złożenie, wykonane przez sam kształt rury. Każda łopatka podwaja liczbę warstw, więc dwadzieścia łopatek daje ich mniej więcej milion, bo dziesięć podwojeń daje około tysiąca.

Czego tu nie dowodzimy

Stephen Smale przełożył zagniatanie ciasta na matematykę mniej więcej dekadę przed odwzorowaniem Hénona. Jego *podkowa* rozciąga kwadrat w długi pasek, wygina go w podkowę i kładzie z powrotem w poprzek kwadratu. Punkty, które zostają w kwadracie na zawsze, tworzą zbiór z lukami w lukach, a Smale udowodnił, że ruch na nim jest tak nieprzewidywalny jak ciąg rzutów monetą. To, co ten rozdział pokazuje o odwzorowaniu Hénona dla $a = 1,4$ i $b = 0,3$, jest zmierzone, a nie udowodnione: nikt nie udowodnił, że zbiór, który rysuje komputer, nie jest, powiedzmy, niezwykle długim cyklem. Michael Benedicks i Lennart Carleson udowodnili w 1991 roku, że odwzorowania postaci Hénona naprawdę mają atraktory dziwne dla wielu innych wyborów tych dwóch liczb.

10.6 Przepływ pocięty na odwzorowanie

Dlaczego reguła z dwiema liczbami miałaby cokolwiek mówić o układzie Lorenza, przepływie z trzema? Wyobraź sobie motyla z taflą szkła wsuniętą w jedno z jego skrzydeł. Trajektoria przebija taflę raz za razem i za każdym razem zostawia kropkę. Zapomnij o drodze między przebiciami i zachowaj tylko kropki. Gdy znasz jedną kropkę, przepływ rozstrzyga, gdzie będzie następna, więc kropki podlegają własnej regule: odwzorowaniu tafli w nią samą, ze stanem złożonym z dwóch liczb. To jest *przekrój Poincarégo*, nazwany na cześć Henriego Poincarégo, który użył go w pracy nad trzema ciałami z rozdziału 13.



Rysunek 10.4: Przekrój Poincarégo. Droga przepływu raz za razem przecina taflę; reguła, która prowadzi od jednego przecięcia do następnego, jest odwzorowaniem tafli.

Odwzorowanie Hénona nie jest przekrojem samego przepływu Lorenza, ale zbudowano je tak, żeby zachowywało się jak taki przekrój: między dwoma przebiciami przepływ

podobny do Lorenzowskiego rozciąga kawałek taflí, ściska go i składa, a to właśnie robią wygięcie, ściśnięcie i zamiana.

Zatrzymaj się i pomyśl 10.6

Rozdział 4 mówił, że gładki przepływ potrzebuje co najmniej trzech liczb w stanie, żeby być chaotycznym. Odwzorowanie Hénona jest chaotyczne przy dwóch, a odwzorowanie logistyczne z rozdziału 5 przy jednej. Czy to sprzeczność?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Tamta reguła dotyczy przepływów, których drogi biegną w sposób ciągły i na płaszczyźnie nigdy nie mogą się przeciąć. Odwzorowanie przeskakuje od jednego stanu do następnego, więc nie przeszkadza mu składać. Przecięcie przepływu z trzema liczbami taflą zostawia odwzorowanie z dwiema, a chaos przeżywa to cięcie.

Podsumowanie

- Chaos w ograniczonym układzie to dwa powtarzane ruchy: **rozciąganie**, które oddala sąsiadów, i **składanie**, które odkłada wszystko z powrotem. **Odwzorowanie namiotowe** zagniata pasek i podwaja każdy mały odstęp.
- **Odwzorowanie Hénona**, $x_{n+1} = 1 - ax_n^2 + y_n$ i $y_{n+1} = bx_n$, wygina, ściska i zamienia miejscami współrzędne płaszczyzny.
- Chmura startów kończy na jednym zbiorze, **atraktorze**; starty, które tam trafiają, tworzą jego **basen przyciągania**.
- Każdy krok mnoży każde pole przez b , **wyznacznik** kroku, a mimo to ruch nigdy nie ustaje: odwzorowanie **ściska** w jedną stronę, a **rozciąga** w drugą.
- Po powiększeniu każda linia atraktora rozpada się na wiązkę, raz za razem. Atraktor taki jak ten, z chaotycznym ruchem na nim, to **atraktor dziwny**: bez pola i bez końca długości.
- **Przekrój Poincarégo** zamienia przepływ w odwzorowanie, dlatego odwzorowanie z dwiema liczbami może być chaotyczne, choć przepływ z dwiema nie może.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

wyjaśnić, jak ruch może na zawsze pozostawać w pewnym obszarze, a mimo to być wrażliwy na stan początkowy, używając słów rozciąganie i składanie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
iterować odwzorowanie namiotowe i odwzorowanie Hénona w Pythonie i czytać to, co wypisują	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
policzyć, przez jaki czynnik jeden krok odwzorowania Hénona mnoży pole, i sprawdzić to pomiarem	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, czym są atraktor, jego basen przyciągania i atraktor dziwny	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
opisać, co pokazuje powiększanie atraktora dziwnego i czym różni się to od powiększania gładkiej krzywej	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
opisać, jak przekrój Poincarégo zamienia przepływ w odwzorowanie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Rodzynka leży w punkcie 0,3 paska. Gdzie jest po jednym zagnieceniu, a gdzie po dwóch?
2. Dwie rodzynki startują w odstępnie 0,01 i pozostają po tej samej stronie zgięcia. Jak daleko są od siebie po trzech zagnieceniach i dlaczego podwajanie nie może trwać w nieskończoność?
3. Wykonaj jeden krok odwzorowania Hénona z punktu (0,5, 0).
4. Plama punktów startowych ma pole 2. Jakie ma pole po trzech krokach odwzorowania Hénona i czy jej punkty przestały się ruszać?
5. Czym różni się atraktor od swojego basenu przyciągania?
6. Co widzisz, gdy powiększasz gładką krzywą, a co, gdy powiększasz atraktor Hénona?
7. Czym jest przekrój Poincarégo i dlaczego pozwala, by odwzorowanie ze stanem złożonym z dwóch liczb było chaotyczne?

Fraktale: mierzenie czegoś, co nie ma długości

Jak długie jest wybrzeże Wielkiej Brytanii? Brzmi to jak pytanie do atlasu. Sprawdź jednak w dwóch źródłach, a możesz znaleźć dwie liczby tak od siebie odległe, że żadna nie jest zaokrągleniem drugiej, i żadna nie jest błędem drukarskim. Dlaczego?

Ten rozdział na to odpowiada, a potem zastępuje długość liczbą, która naprawdę opisuje kształty takie jak wybrzeże. Zbudujesz dwa takie kształty, powtarzając jedną regułę, zmierzysz nową liczbę, licząc pudełka, a potem tak samo zmierzysz kształt, który chaos zostawił po sobie w rozdziale 10.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- zmierzyć krzywą linijkami różnej długości i powiedzieć, czy jej długość się ustala
- zbudować krzywą Kocha i zbiór Cantora, powtarzając regułę, i ustalić, co każda runda robi z ich długością
- wyznaczyć wymiar z tego, z ilu kopii składa się kształt i ile razy każda kopia jest pomniejszona
- obliczyć wymiar pudełkowy z listy punktów, odczytać go z wykresu w skali logarytmicznej i powiedzieć, na ile mu ufać
- powiedzieć, co twierdzi wymiar między jeden a dwa, i w jakim zakresie rozmiarów prawdziwy obiekt może na niego zasłużyć

11.1 Wybrzeże i linijka

Pierwszym, kto potraktował to pytanie poważnie, był Lewis Fry Richardson, angielski fizyk. Pod koniec życia badał wojny i chciał wiedzieć, czy kraje o dłuższej wspólnej granicy częściej ze sobą walczą. Odkrył, że sąsiedzi potrafią podawać dla tej samej granicy wyraźnie różne długości. Jego pomiary opublikowano w 1961 roku, po jego śmierci.

Krzywą na mapie mierzy się cyrklem: dwiema ostrymi nóżkami rozstawionymi na stałą odległość. Przetawiasz je wzdłuż wybrzeża, liczysz kroki i mnożysz przez długość kroku. Ta stała odległość to twoja linijka.

Zatrzymaj się i pomyśl 11.1

Mierzysz ten sam odcinek wybrzeża dwa razy na tej samej dokładnej mapie: raz z cyrklem rozstawionym na 10 kilometrów, raz na 1 kilometr. Który pomiar jest dłuższy, czy też się zgadzają?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Pomiar z 1-kilometrową linijką. Długa linijka przeskakuje prosto nad każdą zatoką węższą od siebie; krótka wchodzi w zatokę i z niej wychodzi, a każdy taki objazd dodaje długości.

W 1967 roku Benoit Mandelbrot oparł na pomiarach Richardsona artykuł w czasopiśmie *Science* zatytułowany „How long is the coast of Britain?”, czyli „Jak długie jest wybrzeże Wielkiej Brytanii?”. Jego odpowiedź, do której prowadzi ten rozdział, brzmiała: to pytanie nie ma odpowiedzi, ale lepsze pytanie ją ma.

Najpierw zobacz, co ten eksperyment robi z krzywą bez ukrytych szczegółów: półokręgiem o promieniu jeden, którego długość to π , około 3,14159 (połowa z 2π obwodu całego okręgu). Postaw na nim punkty, przejdź od każdego do następnego po prostej i dodaj kroki. Oto cała linijka:

code/ch11/_ruler.py

```

17 def length(points: list[Point]) -> float:
18     """Add up the straight steps from each point to the next."""
19     total = 0.0
20     for (x0, y0), (x1, y1) in pairwise(points):
21         dx, dy = x1 - x0, y1 - y0
22         total += math.sqrt(dx * dx + dy * dy) # Pythagoras
23     return total

```

Listing 11.1: Linijka: dodaj proste kroki

`pairwise(points)` podaje pętli każdy punkt razem z następnym, a wiersz oznaczony `Pythagoras` to odległość między nimi: kwadrat przesunięcia w poziomie plus kwadrat przesunięcia w pionie, spierwiastkowane. Następny listing zaczyna od trzech punktów na półokręgu i w każdej rundzie dokłada punkt na okręgu w połowie między każdymi dwoma sąsiadami, co skraca linijkę o połowę:

code/ch11/semicircle.py

```

40 print(" steps    ruler    measured length")
41 for pts in rounds(8):
42     steps = len(pts) - 1
43     ruler = length(pts[:2]) # one step, from the first point
44     print(f"{steps:6d} {ruler:.4f} {length(pts):.5f}")

```

Listing 11.2: Półokrąg mierzony coraz krótszymi linijkami

steps	ruler	measured length
2	1.4142	2.82843
4	0.7654	3.06147
8	0.3902	3.12145
16	0.1960	3.13655
32	0.0981	3.14033
64	0.0491	3.14128
128	0.0245	3.14151
256	0.0123	3.14157
512	0.0061	3.14159

Każda krótsza linijka mierzy więcej, jak na wybrzeżu. Ale każda runda dodaje mniej więcej 0,25 tego, co dodała poprzednia, więc przyrosty wygasają, a pomiary ustalają się na π . To ustalanie się jest tym, czym długość *jest*: liczbą, do której zbliżają się pomiary, gdy linijka się skraca. Gładka krzywa ją ma. Wybrzeża Richardsona jej nie miały: dla każdego wybrzeża i każdej granicy, które sprawdził, zmierzona długość rosła dalej, gdy linijka się skracała.

11.2 Krzywa, która się nie ustala

Nie da się rozumować o szczegółach, których jeszcze nie naniesiono na mapę, więc zbuduj wybrzeże z reguły, w której każda skala szczegółów jest znana z góry. Helge von Koch opisał taką w 1904 roku. Zaczynij od odcinka o długości jeden. Podziel go na trzy równe części i zastąp środkową dwoma górnymi bokami trójkąta, każdym tak długim jak zastępowana część, jak namiot rozbity na środkowej trzeciej. To pierwsza runda. Druga robi to samo z każdą z czterech części i tak dalej, bez końca.

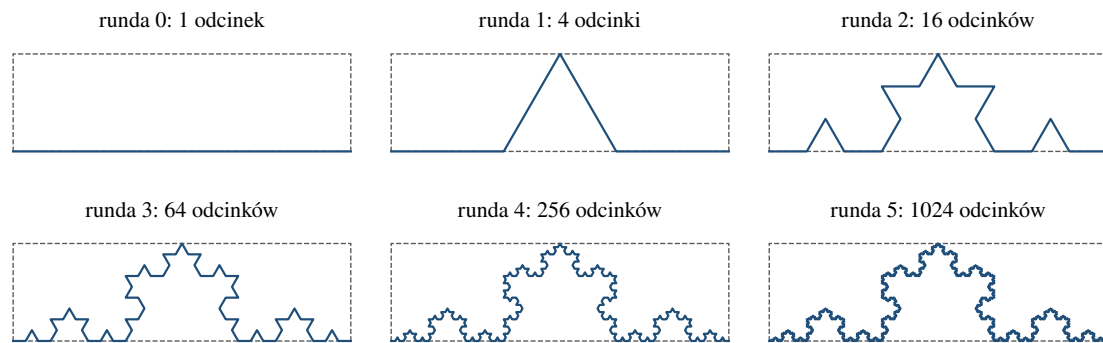
Zatrzymaj się i pomyśl 11.2

Każda runda zastępuje każdy prosty odcinek czterema odcinkami, każdym trzy razy krótszym. Przez jaką liczbę mnoży się całkowita długość w każdej rundzie?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Przez $\frac{4}{3}$: cztery odcinki, każdy o długości $\frac{1}{3}$ zastępowanego, dają razem $4 \times \frac{1}{3}$ jego długości.

Spójrz na rundę 5: krzywa składa się z czterech mniejszych kopii samej siebie, a każda z nich znów z czterech mniejszych kopii. To *samopodobieństwo*, reguła prześwitująca w każdej skali. Listing buduje rundy od 0 do 6 i mierzy każdą linijką z listingu 11.1:



Rysunek 11.1: Krzywa Kocha po rundach od 0 do 5. Każda runda rozbija namiot na środkowej trzeciej każdego prostego odcinka; przerywany prostokąt, wysoki jak pierwszy namiot, mieści każdą rundę.

code/ch11/koch_length.py

```

35 print("round  pieces   ruler  length  height   area")
36 for n in range(7):
37     pts = koch(n)
38     pieces = len(pts) - 1
39     ruler = 1 / 3**n
40     height = max(y for x, y in pts)
41     print(f"{n:5d} {pieces:6d} {ruler:7.5f} {length(pts):7.3f}"
42           f" {height:7.4f} {area(pts):7.4f}")

```

Listing 11.3: Krzywa Kocha mierzona runda po rundzie

round	pieces	ruler	length	height	area
0	1	1.00000	1.000	0.0000	0.0000
1	4	0.33333	1.333	0.2887	0.0481
2	16	0.11111	1.778	0.2887	0.0695
3	64	0.03704	2.370	0.2887	0.0790
4	256	0.01235	3.160	0.2887	0.0832
5	1024	0.00412	4.214	0.2887	0.0851
6	4096	0.00137	5.619	0.2887	0.0859

Po rundzie n każdy odcinek ma długość $\frac{1}{3^n}$, więc mierzenie tej rundy to mierzenie krzywej Kocha linijką tej długości. Kolumna length to za każdym razem $\frac{4}{3}$ wiersza wyżej: po rundzie 6 wynosi 5,619, a po rundzie 20 wynosiłaby 315. Wysokość, height, stoi na 0,2887, wysokości pierwszego namiotu. Pole, area, między krzywą a jej podstawą (funkcja wyżej w tym samym pliku) pełnie ku 0,0866 i dalej nie idzie: każda runda rozbija cztery razy więcej namiotów, każdy o polu dziewięć razy mniejszym, więc dodaje $\frac{4}{9}$ tego, co dodała poprzednia, a te przyrosty wygasają jak przyrosty półokręgu.

Sama reguła to `koch` w `chaoslab`. Jedyny nieznan wiersz to wierzchołek namiotu: pół odcinka wzdłuż środkowej trzeciej, potem w bok o h razy odcinek, czyli o wysokość trójkąta równobocznego.

code/src/chaoslab/fractals.py

```

31 def koch(level: int) -> list[Point]:
32     """The corners of the Koch curve from (0, 0) to (1, 0) after `level`
33     rounds of replacing every segment's middle third with a tent."""
34     h = math.sqrt(3.0) / 2.0 # height of a unit equilateral triangle
35     pts = [(0.0, 0.0), (1.0, 0.0)]
36     for _ in range(level):
37         nxt = [pts[0]]
38         for (x0, y0), (x1, y1) in zip(pts, pts[1:], strict=False):
39             dx, dy = (x1 - x0) / 3, (y1 - y0) / 3
40             a = (x0 + dx, y0 + dy)
41             b = (x0 + 2 * dx, y0 + 2 * dy)
42             # the tip: the middle third turned 60 degrees to the left
43             tip = (a[0] + dx / 2 - h * dy, a[1] + dy / 2 + h * dx)
44             nxt += [a, tip, b, (x1, y1)]
45         pts = nxt
46     return pts

```

Listing 11.4: Reguła Kocha w chaoslab

Zatrzymaj się i pomyśl 11.3

Załóżmy, że potrafisz narysować każdą rundę, bez końca, i zmierzyć gotową krzywą liniijką tak krótką, jak zechcesz. Jaka długość dostaniesz?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Żadnej. Za każdym razem, gdy linijka skraca się trzykrotnie, pomiar rośnie $\frac{4}{3}$ razy i nigdy się nie ustala. Gotowa krzywa jest dłuższa od każdej liczby, jaką zechcesz wymienić, a wciąż mieści się w przerywanym prostokącie.

Pułapka

„Wybrzeże ma długość, tylko trudno ją zmierzyć.” Długość to wartość, na której ustalają się pomiary, gdy linijka się skraca, a pomiary wybrzeża się nie ustalają: każda dokładniejsza mapa znajduje zatoki w zatokach. Długość wybrzeża to fakt o wybrzeżu *i o linijce* naraz, więc liczba podana bez linijki albo bez skali mapy nie jest w ogóle faktem o wybrzeżu.

11.3 Liczenie kopii

Georg Cantor opisał drugi kształt w 1883 roku i ten idzie w przeciwną stronę. Weź odcinek od 0 do 1, podziel go na trzy części i wyrzuć środkową, zostawiając jej końce. Zrób to samo z każdym kawałkiem, który został, bez końca.

Zatrzymaj się i pomyśl 11.4

Wyrzucaj środkowe trzecie bez końca. Ile z długości odcinka zostanie na końcu? I czy w ogóle coś zostanie?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź.

Długości nie zostanie: każda runda zachowuje $\frac{2}{3}$ tego, co było, więc po 20 rundach 1 048 576 kawałków mieści 0,0003 długości, a całość dąży do zera. Ale zostają punkty. Końce każdego kawałka, takie jak $\frac{1}{3}$ i $\frac{2}{3}$, nigdy nie są wyrzucane. Nie jest też wyrzucana $\frac{1}{4}$, która nigdy nie jest końcem: leży w lewej trzeciej; rozciągnij tę trzecią trzykrotnie, a $\frac{1}{4}$ trafi na $\frac{3}{4}$, do prawej trzeciej; rozciągnij tę, a wróci na $\frac{1}{4}$. Omija każdą środkową trzecią bez końca.

To, co zostaje, to *zbiór Cantora*: pył nieskończenie wielu punktów, więcej, niż dałoby się kiedykolwiek wypisać na liście, bez żadnej długości. Długość to złe pytanie dla obu kształtów: jeden ma jej za dużo, drugi za mało. Oto lepsze.

Pomniejsz odcinek trzykrotnie, a trzy kopie odbudują oryginał. Pomniejsz kwadrat trzykrotnie, a potrzeba dziewięciu; sześciąt potrzebuje 27. To 3^1 , 3^2 i 3^3 , a wykładnik to wymiar. Kształt złożony z kopii samego siebie, każdej pomniejszonej tyle samo razy, ma więc wymiar D , dla którego

$$\text{pomniejszenie}^D = \text{kopie}.$$

Zatrzymaj się i pomyśl 11.5

Krzywa Kocha to cztery kopie samej siebie, każda pomniejszona trzykrotnie. Jeśli $3^D = 4$, to czy D wynosi 1, 2, czy coś pomiędzy?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Coś pomiędzy: $3^1 = 3$ kopie to za mało, a $3^2 = 9$ to za dużo. Potęga 3, która daje 4, to około 1,26.

Znalezienie tej potęgi to zadanie logarytmu, który w rozdziale 2 odpowiadał na pytanie „ile kroków, zanim”, a tu odpowiada na pytanie „ile czynników trzy daje cztery”:

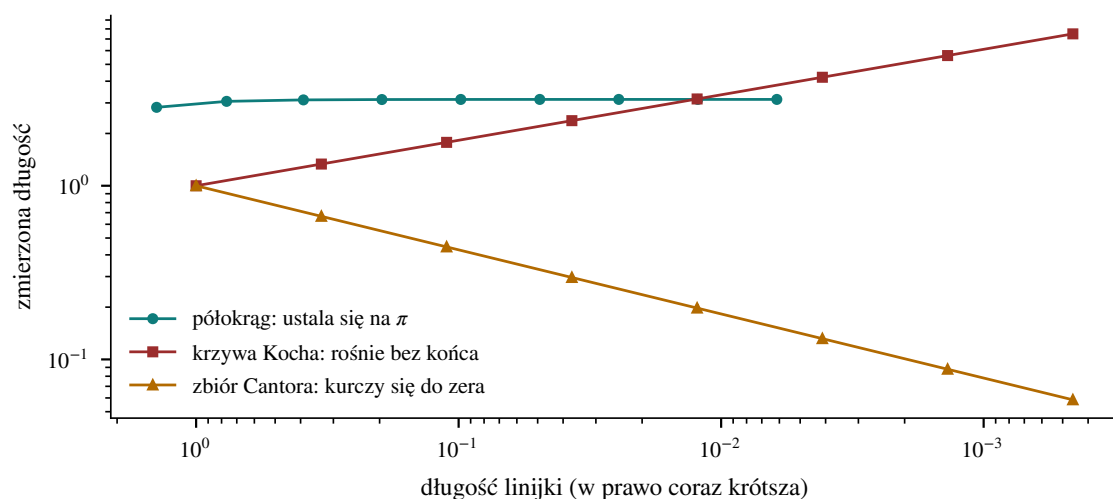
$$D = \frac{\ln 4}{\ln 3} \approx 1,26,$$

gdzie $\ln$ to logarytm naturalny z każdego kalkulatora. Zbiór Cantora to dwie kopie samego siebie, każda pomniejszona trzykrotnie, więc jego wymiar to $\ln 2 / \ln 3$, około 0,63: więcej niż osobne punkty, które mają wymiar 0, i mniej niż odcinek.

Pułapka

„Wymiar musi być liczbą całkowitą.” Jest liczbą całkowitą dla kształtów, które poznałeś w szkole, bo są gładkie: przybliż gładką krzywą, a wygląda jak odcinek. Licz kopie zamiast kierunków, a wymiar staje się pomiarem, który może wyjść jakikolwiek. Wymiar krzywej Kocha, 1,26, mówi coś precyzyjnego: za długa, żeby mieć długość, za cienka, żeby mieć pole. Kształt, którego wymiar nie jest liczbą całkowitą, Mandelbrot nazwał w 1975 roku *fraktalem*, od łacińskiego *fractus*, złamany.

Wymiar wyjaśnia też linijki Richardsona. Skróć linijkę trzykrotnie, a zmierzona długość pomnoży się przez 3^{D-1} : przez $\frac{4}{3}$ dla krzywej Kocha, przez 1 dla wszystkiego, co gładkie, przez $\frac{2}{3}$ dla zbioru Cantora. Na osiach, na których każdy krok to dziesięciokrotność, każdy z tych przypadków jest prostą, a rysunek 11.2 pokazuje wszystkie trzy. Wybrzeża Richardsona dawały właśnie takie proste, a Mandelbrot odczytał z ich nachyleń wymiary: około 1,25 dla zachodniego wybrzeża Wielkiej Brytanii.



Rysunek 11.2: Co dzieje się ze zmierzoną długością, gdy linijka się skraca (w prawo). Gładka krzywa się ustala; krzywa Kocha rośnie bez końca, a zbiór Cantora kurczy się do zera, każda wzdłuż prostej, której nachylenie to jej wymiar minus jeden.

Laboratorium 11.1 – Zbiór Cantora z reguły

Napisz `cantor`, który zwraca kawałki pozostałe po danej liczbie rund, oraz `length_left`, który je sumuje. Potem napisz `similarity_dimension`, który zamienia „tyle kopii, każda pomniejszona tyle razy” na wymiar za pomocą `math.log`. Test sprawdza odcinek, kwadrat, krzywą Kocha i zbiór Cantora oraz to, że $\frac{1}{4}$ przetrwa każdą rundę.

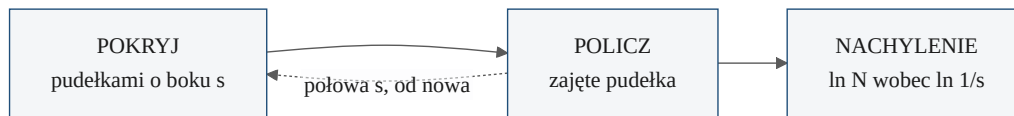
Plik startowy: `code/labs/ch11/111_01_cantor.py`

Test: `code/labs/ch11/test_111_01_cantor.py`

Uruchom: `cd code && uv run pytest -k 111_01_cantor`

11.4 Liczenie pudełek

Liczenie kopii wymaga reguły, a żadna reguła nie zbudowała wybrzeża. Na wszystko, co da się narysować, działa *zliczanie pudełek*: nałóż na kształt siatkę kwadratów o boku s , policz kwadraty, których dotyka, nazwij tę liczbę N , zmniejsz s o połowę i policz znowu.



Rysunek 11.3: Zliczanie pudełek w trzech krokach. Przepis potrzebuje tylko listy punktów, więc działa na kształtach, których nie zbudowała żadna reguła.

Dla odcinka zmniejszenie pudełka o połowę podwaja liczbę pudełek; dla wypełnionego kwadratu mnoży ją przez cztery. W ogólności zmniejszenie pudełka o połowę mnoży N przez 2^D i tak odczytuje się D .

Zatrzymaj się i pomyśl 11.6

Jeśli wymiar krzywej Kocha naprawdę wynosi 1,26, to mniej więcej ile razy więcej pudełek powinna potrzebować po każdym zmniejszeniu pudełka o połowę?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około $2^{1,26}$, czyli 2,4: więcej niż 2 dla odcinka i mniej niż 4 dla kwadratu.

Następny listing pokrywa 65 537 wierzchołków krzywej z rundy 8 pudełkami od $\frac{1}{2}$ do $\frac{1}{1024}$, celowo połówkami: krzywa jest zbudowana z trzecich, a pomiar, który by to wiedział, byłby słabszym sprawdzianem.

code/ch11/koch_boxes.py

```

35 ns = counts(POINTS, SIZES)
36 print(" box side   boxes   times as many as the row above")
37 for k, (s, n) in enumerate(zip(SIZES, ns, strict=True)):
38     line = f" 1/{round(1 / s):<6d} {n:8d}"
39     if k:
40         line += f"   {n / ns[k - 1]:.2f}"
41     print(line)
42 print(f"slope of ln N against ln(1/s): {dimension(SIZES, ns):.3f}")
43 print(f"ln 4 / ln 3:                      {math.log(4) / math.log(3):.3f}")
  
```

Listing 11.5: Zliczanie pudełek na krzywej Kocha

box side	boxes	times as many as the row above
1/2	3	
1/4	7	2.33
1/8	15	2.14
1/16	33	2.20
1/32	91	2.76
1/64	201	2.21
1/128	539	2.68
1/256	1255	2.33
1/512	3169	2.53
1/1024	7325	2.31
slope of $\ln N$ against $\ln(1/s)$: 1.262		
$\ln 4 / \ln 3$: 1.262		

Ostatnia kolumna waha się wokół 2,4, bo trzecie i połówki do siebie nie pasują. Żeby uśrednić to wahanie, narysuj logarytm każdej liczby pudełek względem logarytmu każdego $1/s$. Jeśli N rośnie jak $(1/s)^D$, logarytm zamienia tę potęgę w prostą o nachyleniu D , a `fit_slope` prowadzi prostą, która przechodzi jak najbliżej wszystkich dziesięciu punktów. Jej nachylenie to 1,26; $\ln 4 / \ln 3$ to 1,26.

Nie ufaj tej zgodności bardziej, niż na to zasługuje. Dopasuj prostą do dowolnych pięciu sąsiednich rozmiarów zamiast wszystkich dziesięciu, a nachylenie wyląduje gdziekolwiek od 1,21 do 1,31. Ten rozrzut to uczciwa wielkość błędu i dlatego wymiar pudełkowy dopasowuje się w tak szerokim zakresie rozmiarów, na jaki pozwalają dane.

Laboratorium 11.2 – Policz pudełka sam

Napisz `count_boxes`, który mówi, ile kwadratów o danym boku zawiera co najmniej jeden punkt, oraz `box_dimension`, który dopasowuje nachylenie dla listy rozmiarów. Test sprawdza odcinek, wypełniony kwadrat, krzywą Kocha i zbiór Cantora, którego wymiar pudełkowy powinien zgodzić się z wymiarem z kopii z pierwszego laboratorium.

Plik startowy: `code/labs/ch11/l11_02_boxes.py`

Test: `code/labs/ch11/test_l11_02_boxes.py`

Uruchom: `cd code && uv run pytest -k l11_02_boxes`

11.5 Kształt, który zostawia chaos

W rozdziale 10 punkty odwzorowania Hénona, które przenosi punkt (x, y) płaszczyzny w $(1 - 1,4x^2 + y, 0,3x)$, osiadały na atraktorze bez żadnego pola, któremu rozciąganie odbiera też skończoną długość. Nikt nie zbudował tego kształtu kopia po kopii. To dokładnie ten rodzaj obiektu, do którego służy zliczanie pudełek.

Zatrzymaj się i pomyśl 11.7

Bez pola i bez skończonej długości. Zanim policzysz choć jedno pudełko: czy spodziewasz się, że wymiar atraktora Hénona wynosi 1, 2, czy coś pomiędzy?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Coś pomiędzy 1 a 2: brak skończonej długości czyni go czymś więcej niż krzywą, a brak pola czymś mniej niż powierzchnią. Listing mierzy około 1,23.

code/ch11/henon_boxes.py

```

37 print("boxes of side 1/4, 1/8, ... 1/1024 touched by the attractor")
38 for n in (100_000, 1_000_000):
39     ns = counts(attractor(n), SIZES)
40     fives = [dimension(SIZES[i:i + 5], ns[i:i + 5]) for i in range(5)]
41     print(f"{n:9d} points:" + "".join(f"{c:6d}" for c in ns))
42     print(f"    slope through all nine sizes:    "
43           f"{dimension(SIZES, ns):.3f}")
44     print(f"    through any five in a row:         "
45           f"{min(fives):.3f} to {max(fives):.3f}")

```

Listing 11.6: Zliczanie pudełek na atraktorze Hénona

```

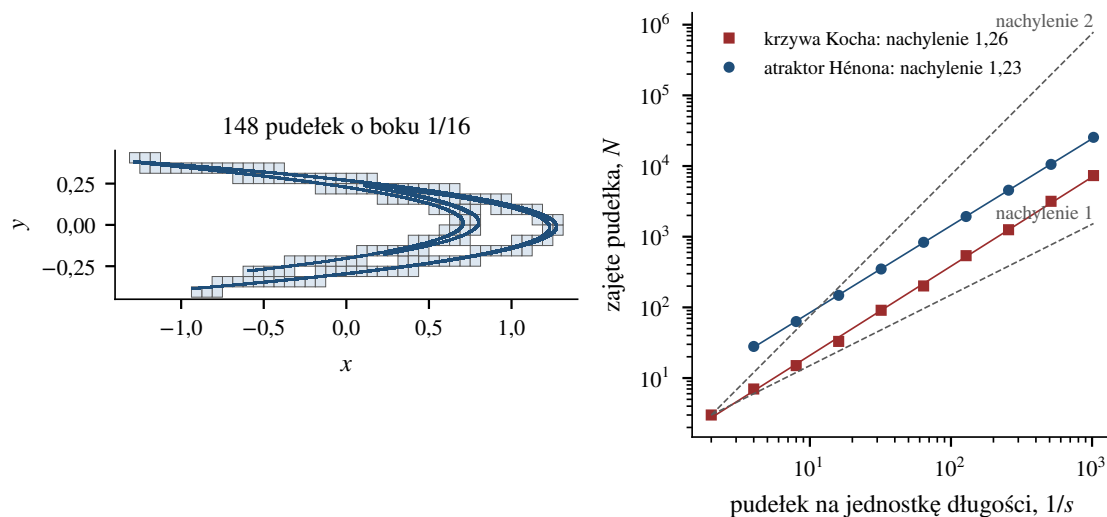
boxes of side 1/4, 1/8, ... 1/1024 touched by the attractor
100000 points:    28    63   148   349   830  1917  4455 10081 22362
slope through all nine sizes:    1.214
through any five in a row:        1.190 to 1.234
1000000 points:   28    63   148   349   833  1930  4548 10551 25457
slope through all nine sizes:    1.230
through any five in a row:        1.226 to 1.237

```

Funkcja `attractor` uruchamia odwzorowanie z $(0, 0)$ i zachowuje każdy punkt po pierwszym tysiącu kroków. Dla 1 000 000 punktów nachylenie przez wszystkie dziewięć rozmiarów pudełek wynosi 1,23, a przez dowolne pięć sąsiednich leży między 1,23 a 1,24. Dla 100 000 punktów spada do 1,21: najmniejsze pudełka potrzebują wielu punktów, zanim każda cienka warstwa atraktora zostanie odwiedzona, więc za mało punktów zostawia pudełka niepolczone i spłaszcza prostą.

Staranne opublikowane pomiary, z dużo większą liczbą punktów i mniejszymi pudełkami, dają wymiar około 1,26. Nasz jest nieco niższy, ma rozrzut i ma znany powód, by być zaniżony: tak wygląda uczciwy pomiar wymiaru. To, że opublikowana wartość zgadza się z wymiarem krzywej Kocha, jest zbiegiem okoliczności: te dwa kształty nic o sobie nie wiedzą.

W tym właśnie sensie atraktory tej części książki są dziwne. Rozciąganie czyni je długimi, składanie trzyma je w pudełku, a kształt za długi, by mieć długość, i za cienki, by mieć pole, jest fraktalem.



Rysunek 11.4: Z lewej: atraktor Hénona pokryty pudełkami o boku $\frac{1}{16}$. Z prawej: liczba zajętych pudełek względem liczby pudełek na jednostkę długości, na osiach, na których każdy krok to dziesięciokrotność. Oba kształty leżą na prostych między przerywanymi prostymi o nachyleniu 1 i 2.

11.6 Fraktale w świecie

Kiedy już znasz ten kształt, widzisz go wszędzie. Liść paproci składa się z mniejszych listków. Rzeka zbiera strumienie, które zbierają mniejsze strumienie. Kalafor romanesco to spirala stożków, z których każdy jest spiralą mniejszych stożków. Naczynia krwionośne rozgałęziają się i rozgałęziają znowu.

Zatrzymaj się i pomyśl 11.8

Twoje płuca muszą przepuszczać tlen do krwi przez cienką powierzchnię, a cała ta powierzchnia musi zmieścić się w klatce piersiowej. Dlaczego płuco mogłoby być zbudowane jak fraktal, rozgałęziając się raz za razem?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Bo fraktal mieści ogromną powierzchnię w małej przestrzeni, tak jak krzywa Kocha mieści nieograniczoną długość w swoim przerywanym prostokącie. Każda runda rozgałęzień mnoży powierzchnię, a klatka piersiowa zostaje tej samej wielkości.

Pułapka

„**Fraktale to ozdoba: ładne obrazki, żadnego pożytku.**” Rozgałęzienia w płucach, naczyniach krwionośnych i sieciach rzecznych wykonują pracę: docierają do każdego punktu objętości albo krajobrazu i mieszczą dużą powierzchnię albo długi brzeg w ograniczonej przestrzeni. Kształt jest rozwiązaniem problemu, a jego wymiar to liczba, którą można zmierzyć i porównać.

Gdzie to spotkasz

W twoich płucach drogi oddechowe dzielą się na dwie mniej więcej dwadzieścia trzy razy między tchawicą a maleńkimi pęcherzykami płucnymi, w których gaz przechodzi do krwi, i tak mieszczą w klatce piersiowej powierzchnię kilkudziesięciu metrów kwadratowych. Fizjologowie opisują drzewo oskrzelowe pojęciami z tego rozdziału: każde pokolenie rozgałęzień to mniejsza kopia poprzedniego.

Każdy z tych kształtów jest fraktalem tylko w pewnym zakresie rozmiarów. Krzywa Kocha ma namioty w namiotach aż do samego dołu, bo reguła się nie męczy; wybrzeże przestaje pokazywać nowe szczegóły mniej więcej na poziomie kamyczków i ziaren piasku, paproć na swoich najmniejszych listkach, płuco na pęcherzykach o średnicy ułamka milimetra. Kiedy więc ktoś nazywa prawdziwy obiekt fraktalem, zapytaj: w jakim zakresie rozmiarów i jak zmierzony? Zliczanie pudełek w zakresie tysiąckrotnym to wynik; w zakresie trzykrotnym to zgadywanie.

Najsłynniejszy fraktal ze wszystkich powstaje z iterowania reguły na punktach płaszczyzny zamiast prostej i jest tematem rozdziału 12.

Podsumowanie

- **Długość** krzywej to wartość, na której ustalają się jej pomiary, gdy linijka się skraca. Gładka krzywa się ustala; wybrzeże rośnie dalej, co odkrył Richardson i o co Mandelbrot zapytał w 1967 roku.
- **Krzywa Kocha** mnoży swoją długość przez $\frac{4}{3}$ w każdej rundzie, więc nie ma skończonej długości, a jej wysokość i pole, które ogranicza, pozostają stałe. Jest **samopodobna**: złożona z mniejszych kopii samej siebie.
- **Zbiór Cantora** zachowuje $\frac{2}{3}$ długości w każdej rundzie, więc długości nie zostaje, ale nieskończenie wiele punktów tak. Kształt złożony z kopii samego siebie, każdej pomniejszonej tyle samo razy, ma **wymiar** D , dla którego pomniejszenie ^{D} = kopie: około 1,26 dla Kocha i 0,63 dla Cantora. Kształt, którego wymiar nie jest liczbą całkowitą, to **fraktal**.
- **Zliczanie pudełek** mierzy wymiar z samych punktów: to nachylenie $\ln N$ względem $\ln(1/s)$, gdzie N pudełek o boku s dotyka kształtu. Rozrzut dla różnych zakresów rozmiarów to uczciwy błąd.
- **Atraktor Hénona** daje w pomiarze około 1,23 (w publikacjach: około 1,26): atraktory dziwne są fraktalami.

- Prawdziwe obiekty są **fraktalami w pewnym zakresie rozmiarów**, nigdy aż do samego dołu, a fraktalne rozgałęzienia mieszczą dużą powierzchnię w małej przestrzeni.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

zmierzyć krzywą linijkami różnej długości i powiedzieć, czy jej długość się ustala	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zbudować krzywą Kocha i zbiór Cantora, powtarzając regułę, i ustalić, co każda runda robi z ich długością	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
wyznaczyć wymiar z tego, z ilu kopii składa się kształt i ile razy każda kopia jest pomniejszona	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
obliczyć wymiar pudełkowy z listy punktów, odczytać go z wykresu w skali logarytmicznej i powiedzieć, na ile mu ufać	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co twierdzi wymiar między jeden a dwa, i w jakim zakresie rozmiarów prawdziwy obiekt może na niego zasłużyć	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dwa atlasy podają długość tego samego wybrzeża jako 800 i 1100 kilometrów. Czy oba mogą mieć rację i co każdy powinien być ci powiedzieć?
2. Krzywa Kocha zaczyna jako odcinek o długości 1. Jak długa jest po trzech rundach?
3. Inna reguła dzieli każdy kawałek odcinka na pięć równych części i wyrzuca drugą i czwartą. Ile długości zostaje po jednej rundzie i jaki wymiar ma to, co zostaje na zawsze?
4. Kształt składa się z 8 kopii samego siebie, każda pomniejszona 2 razy. Jaki ma wymiar i jaki znany kształt go ma?
5. Zmniejszenie pudełka o połowę zmienia liczbę pudełek ze 100 na 283. Jaki jest w przybliżeniu wymiar?
6. Dlaczego zmierzony wymiar atraktora Hénona wzrósł, gdy zwiększyła się liczba punktów?
7. Ktoś mówi ci, że główka brokułu to fraktal. O co powinieneś zapytać?

Iteracja na płaszczyźnie: Julia i Mandelbrot

Weź liczbę, podnieś ją do kwadratu i dodaj 1. Zrób to samo z wynikiem, i jeszcze raz. Zaczynając od 0, dostajesz 0, 1, 2, 5, 26 i liczby znikają, pędząc do nieskończoności. Dodaj -1 zamiast 1, a ten sam przepis daje 0, -1 , 0, -1 bez końca. Oto więc pytanie z odpowiedzią tak albo nie dla każdej liczby c , którą możesz dodawać: czy przepis, zaczęty od 0, ucieka?

Na osi liczbowej odpowiedź jest nudna: te c , które nie uciekają, tworzą jeden nieprzerwany odcinek, od -2 do $\frac{1}{4}$. Ten rozdział zadaje to samo pytanie punktom płaskiej płaszczyzny, niczego poza tym nie zmieniając, a odpowiedzią jest najsłynniejszy obraz w matematyce. To także droga do chaosu z rozdziału 8 oglądana z boku i ostatnia rzecz, którą część III ma ci do pokazania o kształtach chaosu.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- dodawać i mnożyć dwa punkty płaszczyzny, ręcznie i za pomocą liczb zespolonych Pythona
- stosować regułę $z \rightarrow z^2 + c$ od punktu startowego i znajdować krok, w którym orbita ucieka
- powiedzieć, co zbiór Julii, a co zbiór Mandelbrota ustala, a co pozwala zmieniać
- przeliczać r odwzorowania logistycznego na c zbioru Mandelbrota i z powrotem oraz odnaleźć kaskadę z rozdziału 8 na osi rzeczywistej
- odróżnić części zbioru Mandelbrota, w których orbita się ustala, od tych, w których nigdy się nie ustala

12.1 Liczby, które są punktami

Wszystkie liczby w tej książce żyły dotąd na prostej. Punkt płaskiej płaszczyzny potrzebuje dwóch liczb: jak daleko w poziomie i jak daleko w pionie. Punkt trzy w prawo i cztery w górę zapisz jako $3 + 4i$. Na razie i to tylko etykieta znacząca „w górę”: liczba przy niej to część pionowa, a druga to część pozioma. Liczby zapisane w ten sposób nazywają się *liczbami zespolonymi*; część pozioma nazywa się *rzeczywistą*, a pionowa *urojoną* – do końca tego podrozdziału będziesz mieć powody, żeby tym nazwom nie ufać.

Dodawanie dwóch takich liczb to przesuwanie. Dodaj osobno części poziome i osobno pionowe, $(3 + 4i) + (1 + 2i) = 4 + 6i$, dokładnie tak, jak dwa ruchy na mapie składają się w jeden.

Mnożenie wymaga reguły; oto ona w słowach. Każdy punkt ma *długość*, czyli odległość od 0, i *kąt*, czyli to, o ile jest obrócony przeciwnie do ruchu wskazówek zegara od dodatniej połowy osi poziomej. Dla $3 + 4i$ długość wynosi 5, z twierdzenia Pitagorasa, a kąt to 53,13 stopnia. Reguła: **żeby pomnożyć dwa punkty, pomnóż ich długości i dodaj ich kąty**. Mnożenie przez 2, które ma długość 2 i kąt 0, podwaja odległość punktu od 0 i nie zmienia jego kierunku. Mnożenie przez i , punkt o jeden krok w górę, zachowuje długość i obraca punkt o ćwierć obrotu przeciwnie do ruchu wskazówek zegara.

Zatrzymaj się i pomyśl 12.1

Punkt i ma długość 1 i kąt 90 stopni. Ile wynosi $i \times i$?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. -1 . Długości mnożą się do $1 \times 1 = 1$, a kąty dodają do 180 stopni: pół obrotu od 1 prowadzi do -1 .

Szkoła uczy, że żadna liczba podniesiona do kwadratu nie daje wyniku ujemnego, i na prostej to prawda. Poza prostą już nie, i nie stało się nic tajemniczego: dwa ćwierćobroty dają pół obrotu. Mając $i \times i = -1$, możesz wymnażać nawiasy jak zawsze i na końcu uporządkować wynik:

$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i.$$

Słowami: część pozioma iloczynu to iloczyn obu części poziomych minus iloczyn obu części pionowych; część pionowa to suma iloczynów części poziomej każdej liczby przez część pionową drugiej. Tej reguły używa komputer i zgadza się ona z regułą długości i kątów dla każdej pary punktów. Python ma ją wbudowaną: typ nazywa się `complex`, a i zapisuje się jako `1j`.

code/ch12/complex_numbers.py

```
13 def times(a: float, b: float, c: float, d: float) -> tuple[float, float]:
14     """(a + b i) times (c + d i): multiply out, then use i * i = -1."""
15     return a * c - b * d, a * d + b * c
```

Listing 12.1: Reguła mnożenia, ręcznie

code/ch12/complex_numbers.py

```
26 print("i times i, by hand: ", times(0, 1, 0, 1))
27 print("i times i, by Python:", 1j * 1j)
28 z = 3 + 4j
29 print("(3+4i)^2, by hand: ", times(3, 4, 3, 4))
30 print("(3+4i)^2, by Python: ", z * z)
31 print(f"lengths: {abs(z)} and {abs(z * z)}")
32 print(f"angles: {degrees(z):.2f} and {degrees(z * z):.2f} degrees")
```

Listing 12.2: Ręcznie, w Pythonie i przez długości i kąty

```
i times i, by hand: (-1, 0)
i times i, by Python: (-1+0j)
(3+4i)^2, by hand: (-7, 24)
(3+4i)^2, by Python: (-7+24j)
lengths: 5.0 and 25.0
angles: 53.13 and 106.26 degrees
```

Podnieś $3 + 4i$ do kwadratu na oba sposoby. Algebra daje $9 - 16 = -7$ w poziomie i $12 + 12 = 24$ w pionie. Obrazek daje długość $5 \times 5 = 25$ i kąt $106,26$ stopnia, dwa razy $53,13$, a punkt $-7 + 24i$ ma dokładnie tę długość i ten kąt. Dwa opisy, jedna odpowiedź.

Zatrzymaj się i pomyśl 12.2

Liczby z i nazywa się urojonymi. Czy cokolwiek na świecie naprawdę zachowuje się jak i ?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tak: wszystko, co ma wielkość i kierunek i daje się obracać. Mnożenie przez i to ćwierć obrotu, a obrót jest tak samo rzeczywisty jak wszystko, co da się zmierzyć.

Pułapka

„Liczby zespolone są urojone, więc nie opisują niczego rzeczywistego.” Ta nazwa to historia, nie opis. Liczba zespolona to punkt płaszczyzny z regułą mnożenia, a ta reguła mówi, jak rozciągać i jak obracać. Wszystko, co się rozciąga i obraca – falę, obrót, prąd przemienny – opisują liczby zespolone, bo właśnie do tego służą, a nie dlatego, że ktoś udaje.

Gdzie to spotkasz

Elektrycy opisują napięcie przemienne jako punkt krążący po okręgu: jego długość to wielkość napięcia, a kąt mówi, w którym miejscu swojego cyklu napięcie jest. Obwód mnoży ten punkt przez stałą liczbę zespoloną, swoją *impedancję*, która zmienia wielkość i przesuwa fazę – długości się mnożą, kąty dodają. Inżynierowie piszą j zamiast i , bo litera i była już zajęta przez natężenie prądu, a Python idzie w ich ślady: dlatego pisze $1j$.

12.2 Kwadrat, dodawanie i pytanie o ucieczkę

Oto cała reguła tego rozdziału. Wybierz liczbę zespoloną c . Zaczynij od $z = 0$ i rób kroki

$$z \rightarrow z^2 + c,$$

gdzie strzałka znaczy *staje się*: podnieś punkt do kwadratu, a potem przesuń go o c . To iteracja z rozdziału 2, tylko z płaszczyzną zamiast prostej. Gdy c leży na osi poziomej, to

arytmetyka z początku rozdziału: ucieczka do nieskończoności dla $c = 1$, cykl długości dwa dla $c = -1$.

Zatrzymaj się i pomyśl 12.3

Zacznij od 0 z $c = -2$ i zapisz kilka pierwszych kroków. Czy orbita ucieka?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Idzie 0, -2 , 2, a potem $2 \times 2 - 2 = 2$ bez końca: stoi dokładnie na 2 i nigdy nie oddala się bardziej.

code/ch12/orbits.py

```
10 def orbit_of_zero(c: complex, steps: int) -> list[complex]:
11     """0, then z*z + c applied again and again: steps + 1 values."""
12     zs = [0j]
13     for _ in range(steps):
14         z = zs[-1]
15         zs.append(z * z + c)
16     return zs
```

Listing 12.3: Orbita 0 pod działaniem reguły

c = 1:	0	1	2	5	26	677
c = -1:	0	-1	0	-1	0	-1
c = 0.25:	0	0.25	0.3125	0.3477	0.3709	0.3875
c = -2:	0	-2	2	2	2	2
c = i:	0	1i	-1+1i	-1i	-1+1i	-1i

Ostatni wiersz to płaszczyzna w działaniu. Dla $c = i$ orbita od razu opuszcza prostą, obiega małą pętlę, a potem przeskakuje między $-1 + i$ a $-i$ bez końca. Wiersz dla $\frac{1}{4}$ pełnie w górę i nigdy nie przekracza $\frac{1}{2}$.

Nie da się patrzeć w nieskończoność, więc potrzebny jest test, który rozstrzyga sprawę w skończenie wielu krokach, i taki test istnieje: **gdy długość orbity przekroczy 2, orbita już nie wraca**. Liczba kroków potrzebna do przekroczenia 2 to *czas ucieczki* punktu, a biblioteka książki liczy go w pięciu wierszach.

Czego tu nie dowodzimy

Dlaczego 2 wystarcza – słowami, a nie jako dowód. Jeśli c jest dłuższe niż 2, już pierwszy krok przekracza 2, więc załóż, że nie jest, a z jest dłuższe niż 2. Podniesienie do kwadratu zamienia długość z w jej kwadrat, a dodanie c cofa punkt co najwyżej o długość c , która jest mniejsza od długości z . Nowa długość wynosi więc co najmniej starą długość razy (stara długość minus jeden), a to więcej niż stara długość: punkt się oddala, i to tym szybciej, im jest dalej. „Dodanie c cofa co najwyżej o jego długość” to nierówność trójkąta, udowodniana na każdym pierwszym kursie liczb zespolonych.

code/src/chaoslab/fractals.py

```

51 def escape_time(c: complex, limit: int = 100, z: complex = 0j) -> int:
52     """Iterate z -> z*z + c from z; return the step at which |z| exceeds 2,
53     or `limit` if it never does. Once |z| > 2 it is gone for good."""
54     for n in range(limit):
55         if z.real * z.real + z.imag * z.imag > 4.0:
56             return n
57         z = z * z + c
58     return limit

```

Listing 12.4: Czas ucieczki, z biblioteki książki

Test dodaje kwadraty obu części i porównuje sumę z 4, co jest twierdzeniem Pitagorasa bez pierwiastka: długość przekracza 2 dokładnie wtedy, gdy jej kwadrat przekracza 4. Dla $c = 1$ odpowiedź to 3, krok, w którym orbita dochodzi do 5. Dla c , którego orbita nigdy nie ucieka, funkcja liczy aż do limit i zwraca tę wartość.

Zatrzymaj się i pomyśl 12.4

$c = \frac{1}{4}$ nigdy nie ucieka: to pełnący wiersz tabeli. Punkt $c = 0,26$ leży o jedną setną dalej w prawo. Mniej więcej ile kroków potrzebuje jego orbita, żeby przekroczyć 2: około 5, około 30 czy około 300?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około 30: przekracza 2 w kroku 30. Pełnie w stronę $\frac{1}{2}$ jak jego sąsiad, długo tam zwleka i dopiero potem się prześlizguje.

code/ch12/slow_escape.py

```

15 for c in (0.3, 0.26, 0.251, 0.2501, 0.25):
16     n = escape_time(c, LIMIT)
17     if n < LIMIT:
18         print(f"c = {c:<6} passes 2 at step {n}")
19     else:
20         print(f"c = {c:<6} still inside after {LIMIT} steps")

```

Listing 12.5: Punkty blisko brzegu zwlekają

```

c = 0.3      passes 2 at step 12
c = 0.26     passes 2 at step 30
c = 0.251    passes 2 at step 97
c = 0.2501   passes 2 at step 312
c = 0.25     still inside after 10000 steps

```

Trochę dalej orbita ucieka szybko: dla $c = 0,3$ znika w kroku 12. Ale za każdym razem, gdy c zbliża się do $\frac{1}{4}$ dziesięć razy bardziej, orbita zwleka mniej więcej trzy razy dłużej: 30, 97, 312 kroków. Nie ma więc liczby kroków, po której można mieć pewność: punkt, który nie uciekł po tysiącu kroków, może jeszcze uciec po dziesięciu tysiącach. Każdy

dalszy obraz jest rysowany z limitem i blisko brzegu jest uczciwym przybliżeniem, a nie wyrokiem.

Laboratorium 12.1 – Czas ucieczki bez liczb zespolonych

Zapisz punkt jako parę zwykłych liczb i napisz jeden krok reguły za pomocą reguły mnożenia z podrozdziału 12.1; potem czas ucieczki, korzystający z tego kroku; potem dwa pytania z odpowiedzią tak albo nie: czy dla tego c orbita 0 zostaje i czy dla ustalonego c zostaje orbita tego punktu startowego? Test porównuje twoje czasy ucieczki z czasami z biblioteki w 480 punktach płaszczyzny.

Plik startowy: code/labs/ch12/l12_01_escape.py

Test: code/labs/ch12/test_l12_01_escape.py

Uruchom: `cd code && uv run pytest -k l12_01_escape`

12.3 Zbiór Mandelbrota

Teraz pokoloruj każde c na płaszczyźnie: na ciemno, jeśli orbita 0 nigdy nie ucieka, tym jaśniej, im szybciej ucieka. Ciemne punkty tworzą *zbiór Mandelbrota*. Program poniżej rysuje go znakami, po jednym dla każdego c , i uznaje punkt za należący do zbioru, jeśli po 50 krokach wciąż jest w środku.

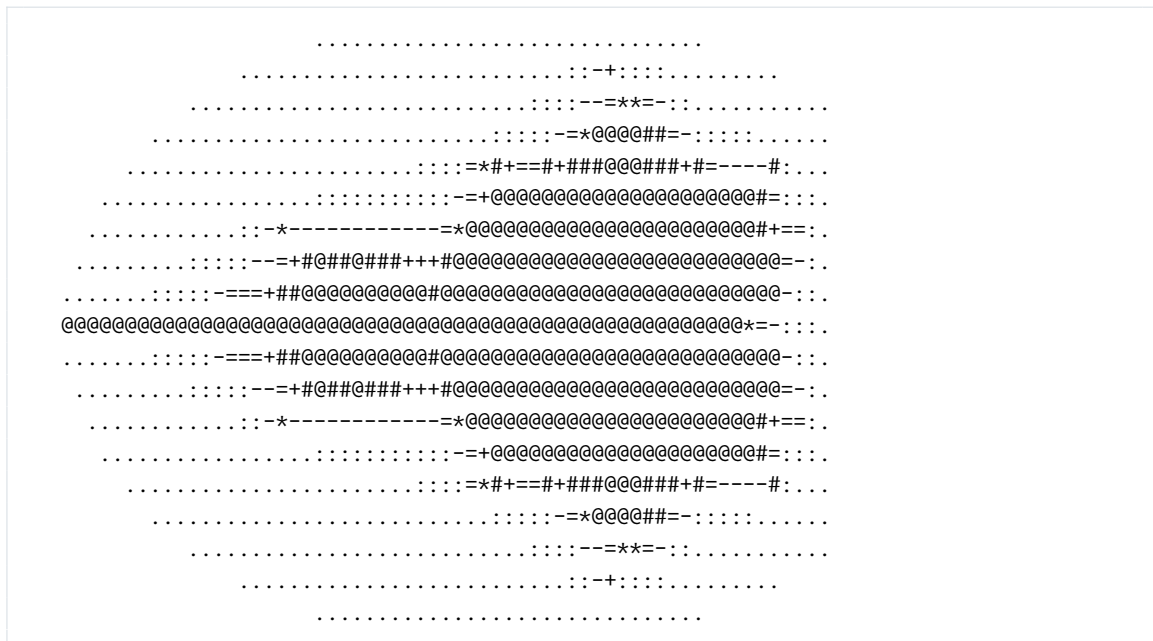
code/ch12/ascii_mandelbrot.py

```

13 LIMIT = 50
14 MARKS = " .,:--==+*##" # the later it escapes, the denser the mark
15
16
17 def mark(c: complex) -> str:
18     n = escape_time(c, LIMIT)
19     return "@" if n == LIMIT else MARKS[min(n, len(MARKS) - 1)]
20
21
22 def main() -> None:
23     for row in range(19): # top to bottom
24         y = (9 - row) / 7.5 # from 1.2 down to -1.2
25         line = "".join(mark(complex(-2.1 + col * 0.042, y))
26                         for col in range(64))
27         print(line.rstrip())

```

Listing 12.6: Zbiór Mandelbrota narysowany znakami



Środkowy wiersz to oś pozioma: stoją w nim same @ od -2 do $\frac{1}{4}$, jak obiecywał początek rozdziału. Poza osią zbiór to ciało w kształcie serca z okrągłym bąblem po lewej, mniejszymi bąblami na obu i cienką igłą sięgającą do -2 . Rysunek 12.1 pokazuje go porządnie, z dwoma powiększonymi kwadratami na brzegu.

Pierre Fatou i Gaston Julia badali reguły takie jak ta w latach 1918–1920, piórem na papierze i zupełnie bez obrazków. Komputerowe obrazy Benoit Mandelbrota, opublikowane w 1980 roku, rozślawiły ten zbiór, a reguła, którą iterował w tamtej pracy, była samą regułą logistyczną, zapisaną liczbami zespolonymi – co wyjaśnia ostatni podrozdział tego rozdziału.

Zatrzymaj się i pomyśl 12.5

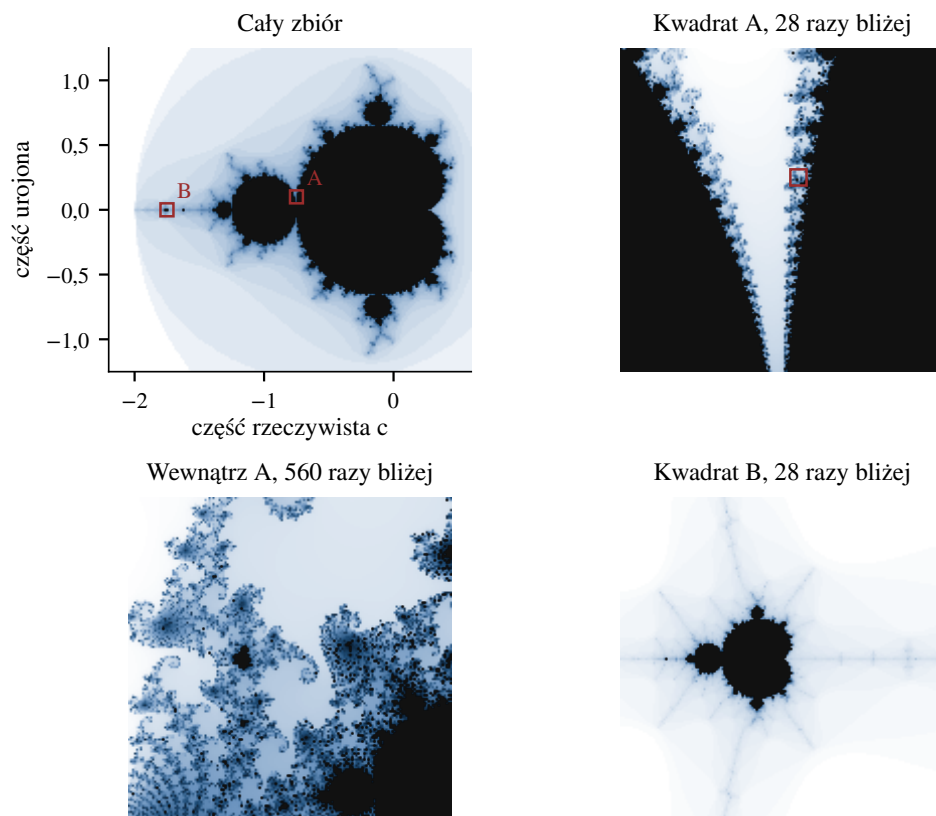
Powiększ brzeg tysiąc razy, milion razy, a wciąż pojawiają się nowe szczegóły: spirale, włoski, maleńkie kopie. Gdzie przechowywane są wszystkie te szczegóły?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nigdzie. Każdy punkt jest liczony od nowa z reguły i testu ucieczki. Program, który narysował powyższe znaki, ma 11 wierszy.

Pułapka

„Nieskończenie wiele szczegółów to nieskończenie wiele informacji.” Brzeg ma szczegóły w każdej skali, do której sięgniesz, i wszystkie wynikają z jednej linijki algebry i jednego pytania. Szczegóły są *wytwarzane*, a nie przechowywane – to znowu lekcja z rozdziału 5, gdzie jedna linijka dawała każdy rodzaj zachowania. Żeby przekazać komuś cały zbiór dokładnie, wystarczy podać mu regułę i pytanie;



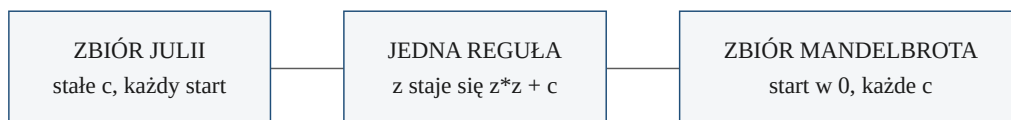
Rysunek 12.1: Zbiór Mandelbrota: czarny tam, gdzie orbita 0 nigdy nie ucieka, tym ciemniej niebieski, im dłużej punkt zwleka przed ucieczką. Kwadrat A leży na brzegu, tam gdzie serce styka się ze swoim największym bąblem, i jest pokazany w dwóch powiększeniach; kwadrat B leży na igle.

krótki opis może mieć nieskończone konsekwencje.

Niektóre proste fakty o zbiorze wciąż są nieznane, na przykład jego pole. Policz kwadraty siatki, których środek po 50 krokach wciąż należy do zbioru, a dostaniesz około 1,59; po 200 krokach 1,53; po 1000 1,51. Wynik wciąż spada, w miarę jak przy brzegu wychodzą na jaw powolni uciekinierzy, i nikt nie zna wzoru na to, gdzie się zatrzyma. Sam brzeg jest tak pomięty, że jego wymiar w sensie z rozdziału 11 wynosi 2, najwięcej, ile może mieć jakiegokolwiek kształt narysowany na płaszczyźnie: udowodnił to Mitsuhiro Shishikura.

12.4 Zbiory Julii: ustal c , zmieniaj start

Ta sama reguła odpowiada na drugie pytanie. Ustal c i zmieniaj *punkt startowy*: czy dla każdego punktu z płaszczyzny orbita, która się w nim zaczyna, ucieka? Punkty startowe, które zostają, wypełniają *zbiór Julii* dla tego c , nazwany na cześć Gastona Julii. (Ściśle biorąc, zbiorem Julii jest brzeg tego obszaru.)



Rysunek 12.2: Jedna reguła, dwa pytania. Ustal c i pytaj o każdy punkt startowy, a narysujesz zbiór Julii; zaczynaj od 0 i pytaj o każde c , a narysujesz zbiór Mandelbrota.

Zatrzymaj się i pomyśl 12.6

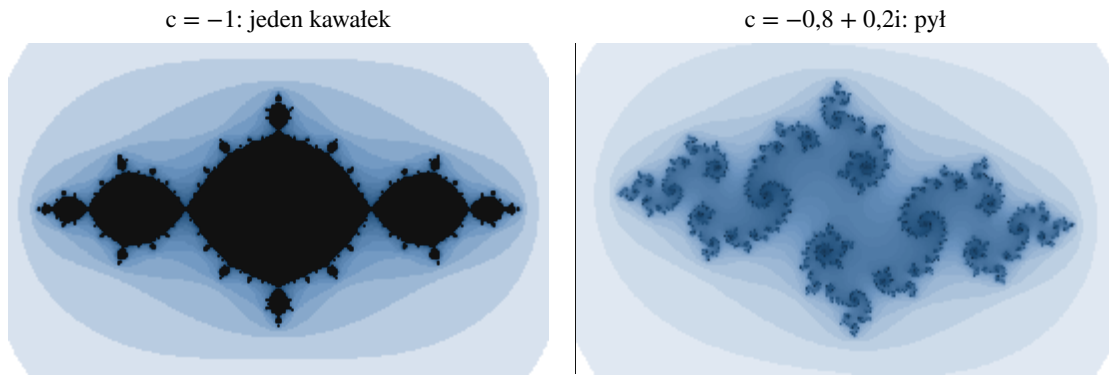
Weź $c = 0$, więc reguła to tylko „podnieś punkt do kwadratu”. Które punkty startowe uciekają?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Te, które leżą dalej niż 1 od środka. Podnoszenie do kwadratu podnosi do kwadratu długość: długość większa od 1 rośnie bez końca, mniejsza od 1 maleje do 0, a długość równa dokładnie 1 zostaje równa 1. Punkty startowe, które zostają, wypełniają koło o promieniu 1, a jego brzegiem jest okrąg.

Zmień c , a okrąg się marszczy. Rysunek 12.3 pokazuje dwa przykłady. Dla $c = -1$ zbiór jest w jednym kawałku: to łańcuch ściśniętych ze sobą bąbli. Dla $c = -0,8 + 0,2i$ rozpadł się w pył: nieskończenie wiele drobinek, z których żadne dwie się nie łączą, jak zbiór Cantora z rozdziału 11.

Co o tym decyduje? Orbita 0, czyli pytanie Mandelbrota. Dla $c = -1$ przeskakuje między 0 a -1 i zostaje; dla $c = -0,8 + 0,2i$ przekracza 2 w kroku 15. Twierdzenie, którego nie dowodzimy tutaj, ale które jest w każdej książce o dynamice zespolonej, mówi, że to zawsze rozstrzyga sprawę: zbiór Julii jest w jednym kawałku dokładnie wtedy, gdy



Rysunek 12.3: Dwa zbiory Julii, czarne tam, gdzie punkt startowy nigdy nie ucieka. Dla $c = -1$ zbiór jest jednym kawałkiem; dla $c = -0,8 + 0,2i$ zostaje tylko pył, zbyt drobny, by wyszedł czarny, widoczny dzięki ciemnym spiralom, na których punkty zwlekają najdłużej.

c należy do zbioru Mandelbrota, a jest pyłem dokładnie wtedy, gdy nie należy. Zbiór Mandelbrota jest więc mapą wszystkich zbiorów Julii naraz, a każdy jego czarny punkt to zbiór Julii, który trzyma się w całości.

12.5 Odwzorowanie logistyczne w przebraniu

Reguła tego rozdziału i odwzorowanie logistyczne z rozdziału 5, $x \rightarrow rx(1-x)$, wyglądają inaczej, a są tą samą regułą. Przenumeruj prostą: mierz każde x od $\frac{1}{2}$ zamiast od 0, odwróć je i rozciągnij r razy, czyli dokonaj zamiany

$$z = r \left(\frac{1}{2} - x \right).$$

Wtedy $x(1-x) = \frac{1}{4} - z^2/r^2$, a następne z , czyli $\frac{r}{2} - r^2x(1-x)$, wychodzi równe $z^2 + c$, gdzie

$$c = \frac{r}{2} - \frac{r^2}{4}.$$

Słowami: weź połowę r i odejmij ćwierć jego kwadratu. W tym przekładzie nic nie ginie. Każda orbita jednej reguły jest orbitą drugiej na przenumerowanej prostej, a $x = \frac{1}{2}$ staje się $z = 0$: orbita 0, o którą pyta zbiór Mandelbrota, to orbita logistyczna startująca z $\frac{1}{2}$.

Zatrzymaj się i pomyśl 12.7

Punkt stały z rozdziału 5 traci stabilność przy $r = 3$ i władzę przejmuje cykl długości dwa. Jakie to c ?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $c = \frac{3}{2} - \frac{9}{4} = -0,75$: dokładnie tam, gdzie serce styka się z okrągłym bąblem po swojej lewej.

Serce to więc punkt stały, a ten bąbel to cykl długości dwa. Gdy r biegnie od 1 do 3, c biegnie od $\frac{1}{4}$ w dół do $-0,75$ wzdłuż odcinka prostej należącego do serca. Listing uruchamia obie reguły obok siebie dla wartości r z rozdziału 5, od punktów startowych, które zamiana ze sobą łączy, i podaje długość cyklu, w który każda z nich się układa.

code/ch12/same_rule.py

```

15 def square_plus(c: float) -> Callable[[float], float]:
16     """The rule  $z \rightarrow z^2 + c$ , on the real line."""
17     return lambda z: z * z + c
18
19
20 def both_periods(r: float, x0: float = 0.3) -> tuple[int | None, ...]:
21     """The cycle length of each rule, after 3000 steps from matching starts.
22     None means no cycle up to 64 steps long was found."""
23     xs = orbit(logistic(r), x0, 3000)
24     zs = orbit(square_plus(r_to_c(r)), r * (0.5 - x0), 3000)
25     return period(xs), period(zs)

```

Listing 12.7: Ta sama reguła, uruchomiona dwa razy

r	c	cycle of x	cycle of z
2.80	-0.5600	1	1
3.20	-0.9600	2	2
3.50	-1.3125	4	4
3.56	-1.3884	8	8
3.83	-1.7522	3	3
3.90	-1.8525	none	none

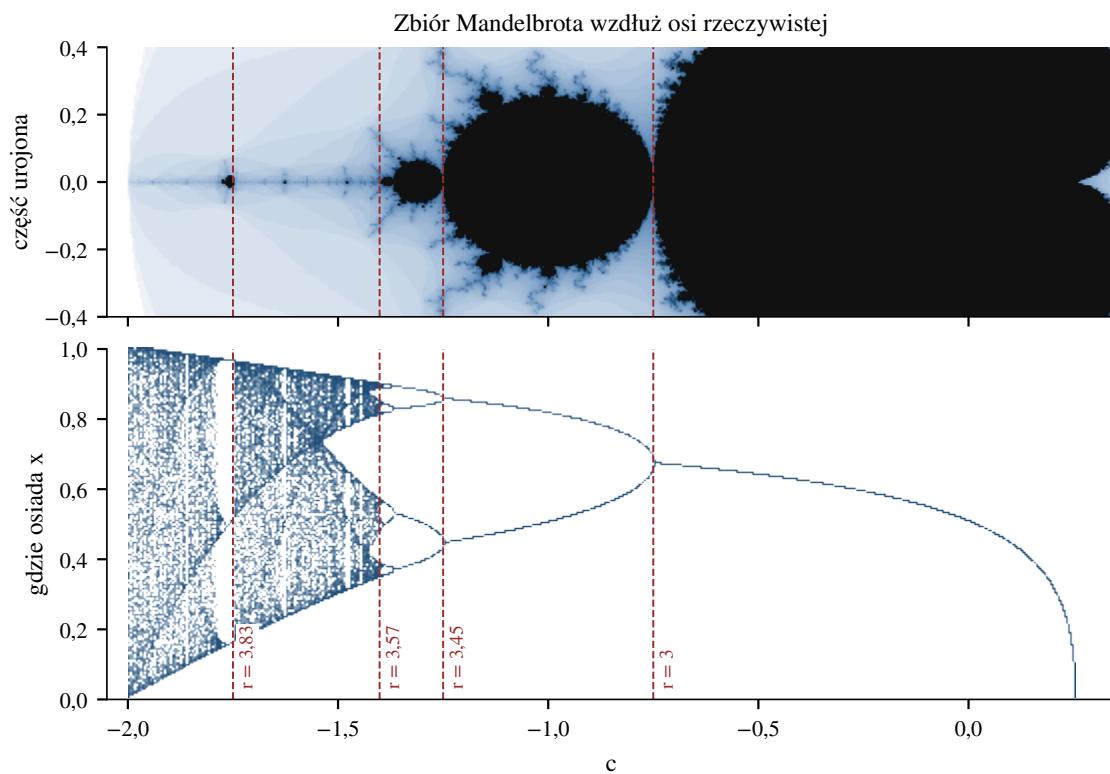
Cykle zgadzają się przy każdym r – jeden, dwa, cztery, osiem, trzy i żaden przy 3,9 – a to jest diagram bifurkacyjny z rozdziału 8, zapisany wzdłuż osi rzeczywistej. Rysunek 12.4 stawia jeden obraz nad drugim.

Drugie rozszczepienie, przy $r = 1 + \sqrt{6}$, przelicza się na $c = -1,25$, gdzie pierwszy bąbel styka się z następnym. Każdy bąbel ma też środek, takie c , przy którym orbita 0 wraca dokładnie do 0: -1 , potem $-1,3107$, potem $-1,3815$ i tak dalej. Przeliczone na r dają 3,2361, 3,4986 i 3,5546: wartości, przy których cykle z rozdziału 8 przechodzą przez $\frac{1}{2}$, czyli jego parametry superstabilne. Odstępy między środkami maleją o czynnik, który ustala się na 4,669 – to znowu liczba Feigenbauma – a bąble gromadzą się przy $c = -1,401$, czyli $r = 3,5699$: w punkcie Feigenbauma. Dalej leży igła. Jest tam nawet okno okresu trzy. Otwiera się przy $c = -1,75$, a kwadrat B na rysunku 12.1 pokazuje, co siedzi na igle w tym miejscu: maleńką kopię całego zbioru.

Zatrzymaj się i pomyśl 12.8

Orbity z rozdziału 5, które nigdy się nie ustalają, żyją przy $r = 3,9$. Tabela powyżej ją przelicza. Czy odpowiadające jej c leży w sercu, w bąblu, czy na igle?

Odpowiedz, zanim pójdziesz dalej.



Rysunek 12.4: U góry zbiór Mandelbrota wzdłuż osi rzeczywistej; na dole diagram bifurkacyjny z rozdziału 8, w którym każde r narysowano w miejscu jego c . Każde rozszczepienie cyklu zachodzi tam, gdzie jeden bąbel styka się z następnym. Linie przerywane oznaczają $r = 3$, $r = 1 + \sqrt{6}$, punkt Feigenbauma i okno okresu trzy.

Odpowiedź. Na igle: c wynosi około $-1,85$, na lewo od punktu Feigenbauma, za ostatnim z bąbli, które się tam gromadzą.

Pułapka

„**Zbiór Mandelbrota to obraz chaosu.**” Większość tego, co czarne, to porządek. W sercu orbita 0 ustala się w jednym punkcie, w dużym bąblu w cyklu długości dwa, a w każdym bąblu w cyklu, którego długość ten bąbel wyznacza. Nieustalające się orbity z rozdziału 5 leżą na cienkim brzegu, a na osi rzeczywistej oznaczają to igłę, między jej oknami. Słynny obraz jest głównie mapą porządku, a chaos żyje wzdłuż jego granicy.

Laboratorium 12.2 – Od r do c i z powrotem

Napisz przeliczenie w obie strony, r na c i c z powrotem na r , oraz samą zamianę $z = r(\frac{1}{2} - x)$. Test uruchamia obok siebie odwzorowanie logistyczne i $z \rightarrow z^2 + c$ i sprawdza, czy twoja zamiana utrzymuje je w zgodzie. Potem przenieś na drugą stronę wartości r podwojenia okresu z pliku startowego i zobacz, gdzie na osi rzeczywistej wylądują.

Plik startowy: `code/labs/ch12/l12_02_convert.py`

Test: `code/labs/ch12/test_l12_02_convert.py`

Uruchom: `cd code && uv run pytest -k l12_02_convert`

Na tym kończy się część III: chaos ma kształt, a jej ostatni obraz zawiera w sobie cały rozdział 8. Część IV wynosi te idee w świat wahadeł, planet, pogody i żywych istot.

Podsumowanie

- **Liczba zespolona** to punkt płaszczyzny, zapisywany jako $a + bi$. Dodawanie to przesuwanie; mnożenie mnoży **długości** i dodaje **kąty**, co daje to samo, co wymnożenie nawiasów z i $i \times i = -1$.
- Reguła $z \rightarrow z^2 + c$, zaczęta od 0, albo zostaje, albo ucieka, a gdy długość przekroczy 2, ucieczka jest ostateczna. **Czas ucieczki** liczy kroki; blisko brzegu rośnie bez ograniczeń.
- **Zbiór Mandelbrota** to wszystkie c , dla których orbita 0 nigdy nie ucieka: serce, bąble, igła do -2 i brzeg ze szczegółami w każdej skali, wytworzonymi przez regułę. Jego pole nie jest dokładnie znane.
- **Zbiór Julii** ustala c i pyta o każdy punkt startowy. Jest w jednym kawałku, gdy c należy do zbioru Mandelbrota, a pyłem, gdy nie należy.
- Zamiana $z = r(\frac{1}{2} - x)$ zmienia **odwzorowanie logistyczne** w $z \rightarrow z^2 + c$ z $c = \frac{r}{2} - \frac{r^2}{4}$, więc oś rzeczywista zbioru to diagram bifurkacyjny: bąble to cykle, gromadzą się w punkcie Feigenbauma, a chaos żyje na igle.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

1	2	3	4	5
☐	☐	☐	☐	☐
dodawanie i mnożenie dwóch punktów płaszczyzny, ręcznie i za pomocą liczb zespolonych Pythona				
1	2	3	4	5
☐	☐	☐	☐	☐
stosowanie reguły $z \rightarrow z^2 + c$ od punktu startowego i znajdowanie kroku, w którym orbita ucieka				
1	2	3	4	5
☐	☐	☐	☐	☐
powiedzieć, co zbiór Julii, a co zbiór Mandelbrota ustala, a co pozwala zmieniać				
1	2	3	4	5
☐	☐	☐	☐	☐
przeliczać r odwzorowania logistycznego na c zbioru Mandelbrota i z powrotem oraz odnaleźć kaskadę z rozdziału 8 na osi rzeczywistej				
1	2	3	4	5
☐	☐	☐	☐	☐
odróżnić części zbioru Mandelbrota, w których orbita się ustala, od tych, w których nigdy się nie ustala				

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Oblicz $(2 + 3i) + (1 - i)$ oraz $(1 + i)(1 - i)$.
2. Znajdź $(1 + i)^2$ za pomocą długości i kątów, a potem sprawdź wynik, wymnażając nawiasy.
3. Zaczynij od 0 z $c = 2$. W którym kroku orbita przekracza 2?
4. Czy $c = -1,5$ należy do zbioru Mandelbrota?
5. Czy dla $c = 0$ punkt startowy $0,6 + 0,9i$ ucieka?
6. Przelicz $r = 3,2$ na c . Czy wypada w sercu, czy w dużym bąblu, i czy zgadza się to z rozdziałem 5?
7. Dlaczego zbiór Mandelbrota nie jest obrazem chaosu?

Część IV

Chaos w świecie

Wahadła i planety

Zegar stojący odmierza czas, bo wahadło to najbardziej regularna rzecz, jaką większość ludzi kiedykolwiek będzie mieć w domu: puść je, a będzie się kołysać w tym samym rytmie, dopóki zegar jest nakręcony. Rozdział 4 pokazał dlaczego: stan wahadła to dwie liczby, a bez tarcia krążą one po tej samej zamkniętej pętli bez końca. Wszechświat jak mechanizm zegara z rozdziału 1 zbudowano właśnie na takiej mechanice.

Teraz zawieś drugie wahadło u dołu pierwszego, na zawiasie w miejscu, gdzie się łączą, i puść je z wysoka. Reguła wciąż pochodzi od Newtona, wciąż jest dokładna i nie ma w niej nic losowego. Co ono robi? Ten rozdział odpowiada pomiarem, a potem idzie za tym samym pytaniem od wahadła na biurku aż do planet krążących wokół Słońca, gdzie zadano je po raz pierwszy.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- przeprowadzić wahadło podwójne krok po kroku metodą RK4 i odczytać jego stan z wydruku
- sprawdzić symulację układu bez tarcia, obserwując jego energię, i powiedzieć, czego ten sprawdzian dowodzi, a czego nie
- zmierzyć, jak daleko od siebie odchodzą dwa niemal identyczne puszczenia, i sprawdzić, czy wynik zależy od długości kroku
- odróżnić łagodne puszczenie od dzikiego i powiedzieć, co zachowanie energii przesądza o przyszłości układu, a czego nie
- oddzielić to, co zmierzono na temat chaosu w Układzie Słonecznym, od tego, co jest legendą

13.1 Wahadło na wahadle

Weź dwa pręty o długości metra i przymocuj drugi zawiasem do końca pierwszego. Na zawiasie umieść ciężarek o masie jednego kilograma, a drugi taki sam na wolnym końcu; całą masę niosą ciężarki, a pręty traktujemy jak nieważkie. Unieś oba ramiona tak, by były odchylone o 120° od pionu, i puść.

Zatrzymaj się i pomyśl 13.1

Jedno wahadło kołysze się z doskonałą regularnością. Czy spodziewasz się, że dwa, jedno zawieszone na drugim, ułożą się w jakiś powtarzający się wzór – bardziej skomplikowany niż u jednego wahadła, ale mimo to regularny?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Puszczane z wysoka wahadło podwójne nie układa się w żaden powtarzający się wzór, jaki dałoby się znaleźć: dolne ramię śmiga dookoła, czasem przelatując nad samą górą, a ruch wciąż się zmienia. Nic losowego tu nie działa.

Różnica tkwi w tym, ile jest liczb. Stan jednego wahadła to dwie liczby: kąt i to, jak szybko ten kąt się zmienia. Stan wahadła podwójnego to cztery liczby, dwa kąty i dwie prędkości ich zmian, a rozdział 4 mówił, że gładka reguła potrzebuje co najmniej trzech liczb, żeby mogła być chaotyczna.

Prawa Newtona dają regułę zmiany dla tych czterech liczb: znając stan teraz, mówi ona, jak szybko zmienia się każda z nich. To mniej więcej piętnaście wierszy arytmetyki z sinusami i cosinusami i nie trzeba jej czytać, żeby jej używać; to `double_pendulum` z biblioteki książki, `chaoslab`. RK4 z rozdziału 4 stosuje ją krok po kroku, po tysięcznej części sekundy naraz:

code/ch13/release.py

```

16 DT = 0.001                # seconds per step
17 RULE = double_pendulum()  # g = 9.81, arms of 1 m, bobs of 1 kg
18
19
20 def release(upper: float, lower: float) -> tuple[float, ...]:
21     """A pendulum let go from rest, with its arms at these angles (degrees).
22
23     The state is four numbers: the two angles (in radians, measured from
24     straight down) and how fast each angle is changing (radians a second).
25     """
26     return (math.radians(upper), math.radians(lower), 0.0, 0.0)

```

Listing 13.1: Puszczanie: cztery liczby i długość jednego kroku

`math.radians` zamienia stopnie na radiany, jednostkę, której oczekują sinus i cosinus w Pythonie. Pełny obrót to 2π radianów, nieco ponad sześć, więc jeden radian to około 57° . Potem listing przesuwa stan naprzód przez dziesięć sekund i wypisuje go co dwie:

code/ch13/release.py

```

32 state = release(120, 120)
33 print(" t (s)   upper arm   lower arm   energy (J)")
34 for step in range(10_001):
35     if step % 2000 == 0:
36         upper = math.degrees(state[0])
37         lower = math.degrees(state[1])
38         energy = double_pendulum_energy(state)
39         print(f"{step * DT:5.1f} {upper:9.1f} deg {lower:7.1f} deg"
40               f" {energy:13.8f}")
41     state = rk4_step(RULE, state, DT)

```

Listing 13.2: Dziesięć sekund wahadła podwójnego, wypisane co dwie

t (s)	upper arm	lower arm	energy (J)
0.0	120.0 deg	120.0 deg	14.71500000
2.0	-109.5 deg	-108.3 deg	14.71500000
4.0	83.2 deg	76.8 deg	14.71500000
6.0	-22.0 deg	-52.7 deg	14.71500000
8.0	-85.5 deg	414.2 deg	14.71499999
10.0	178.5 deg	758.5 deg	14.71499999

Najpierw przeczytaj ramiona. Kąt ujemny to wychylenie na drugą stronę, a kąt powyżej 360° oznacza, że ramię obróciło się dookoła: 758° po dziesięciu sekundach to dwa pełne obroty i jeszcze 38° . Dolne ramię wiruje. Teraz przeczytaj ostatnią kolumnę.

13.2 Liczba, która nie może drgnąć

Wahadło bez tarcia nie może zyskać ani stracić energii: nic go nie popycha i nic nie spowalnia. Jego energia ma dwie części. Energia wysokości: każdy ciężarek o masie jednego kilograma wnosi $g \times h$, gdzie g to 9,81, a wysokość h mierzy się w górę od punktu zawieszenia. Energia ruchu: połowa kwadratu prędkości każdego ciężarka. Gdy wahadło opada, wysokość zamienia się w prędkość, i z powrotem; suma pozostaje dokładnie taka sama.

Zatrzymaj się i pomyśl 13.2

Załóżmy, że puszczasz wahadło z obydwoma ramionami ustawionymi poziomo, w tę samą stronę, pod kątem 90° . Jaka jest jego energia, jeśli wysokości mierzymy od punktu zawieszenia?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Zero. Oba ciężarki są na wysokości punktu zawieszenia, więc obie wysokości są zerowe, a oba stoją w miejscu, więc nie ma też energii ruchu.

Wysokości poniżej punktu zawieszenia liczą się jako ujemne, więc wahadło wiszące w spoczynku ma energię ujemną; znaczenie mają tylko zmiany energii. Dla puszczenia z wydruku odrobina geometrii umieszcza górny ciężarek pół metra nad punktem zawieszenia, a dolny cały metr, więc energia wynosi $9,81 \times (0,5 + 1)$, czyli około 14,7 dżula – od tego zaczyna się kolumna energii. Przeczytaj ją w dół: przez dziesięć sekund wirowania zmienia się o jedną jednostkę na ósmym miejscu po przecinku, o stumilionową część dżula.

To jest sprawdzian i to najpożyteczniejszy nawyk w symulowaniu mechaniki. Komputer nie rozwiązuje reguły; robi małe kroki, każdy odrobinę błędny. Prawdziwe wahadło zachowuje energię dokładnie, więc każda zmiana energii w symulacji to zmierzony błąd samych kroków.

Laboratorium 13.1 – Liczba, która nie może drgnąć

Napisz energię wahadła podwójnego sam: `heights` podaje wysokość każdego ciężarka nad punktem zawieszenia, a `energy` – całość. Plik startowy objaśnia jedyny krok, który wymaga uwagi: dolny ciężarek jest niesiony przez górny, więc jego prędkość składa się z dwóch ruchów. Test porównuje twoją energię z energią z biblioteki, a potem obserwuje ją podczas kołysania, w którym nie może się zmienić.

Plik startowy: `code/labs/ch13/l13_01_energy.py`

Test: `code/labs/ch13/test_l13_01_energy.py`

Uruchom: `cd code && uv run pytest -k l13_01_energy`

13.3 Milionowa część radiana

Teraz puść dwa wahadła podwójne. Drugie jest dokładnie takie jak pierwsze, z wyjątkiem tego, że jego górne ramię zaczyna o jedną milionową część radiana dalej: na końcu metrowego ramienia to tysięczna część milimetra, dużo mniej niż grubość włosa. Oba słuchają tej samej dokładnej reguły, z tym samym krokiem.

Zatrzymaj się i pomyśl 13.3

Oba wahadła zachowują energię, jak pokazał sprawdzian, i nic losowego ich nie dotyka. Czy pod koniec dziesięciu sekund z wydruku powyżej wciąż będą się kołysać razem? Po minucie? Czy się rozejdą, a jeśli tak, to mniej więcej kiedy?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Rozchodzą się. Po 10,2 sekundy różnią się o więcej niż jedną dziesiątą radiana, około sześciu stopni, co zobaczyłby każdy obserwator. Kilka sekund później robią już zupełnie co innego.

`code/ch13/two_releases.py`

```
20 def gap(a: tuple[float, ...], b: tuple[float, ...]) -> float:
21     """How far apart two pendulums are: the larger of the two angle gaps."""
22     return max(abs(a[0] - b[0]), abs(a[1] - b[1]))
```

Listing 13.3: Jak daleko od siebie są dwa wahadła

`code/ch13/two_releases.py`

```
28 a = release(120, 120)
29 b = (a[0] + NUDGE, a[1], a[2], a[3])
30 step = 0
31 while gap(a, b) <= APART:
32     if step % 1000 == 0:
33         print(f"t = {step * DT:.1f} s    gap {gap(a, b):.0e} rad")
34         a, b = rk4_step(RULE, a, DT), rk4_step(RULE, b, DT)
35         step += 1
36 print(f"more than {APART} rad apart after {step * DT:.1f} s")
```

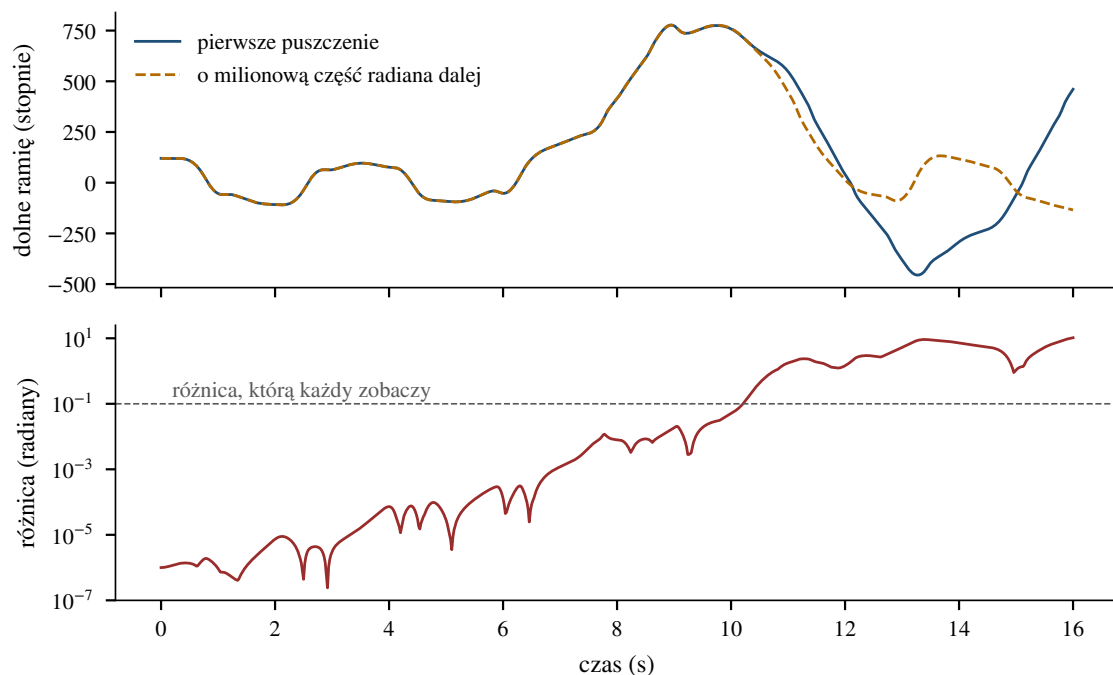
Listing 13.4: Dwa puszczenia różniące się o milionową część radiana

```

t = 0.0 s   gap 1e-06 rad
t = 1.0 s   gap 9e-07 rad
t = 2.0 s   gap 7e-06 rad
t = 3.0 s   gap 3e-06 rad
t = 4.0 s   gap 7e-05 rad
t = 5.0 s   gap 3e-05 rad
t = 6.0 s   gap 1e-04 rad
t = 7.0 s   gap 1e-03 rad
t = 8.0 s   gap 8e-03 rad
t = 9.0 s   gap 2e-02 rad
t = 10.0 s  gap 5e-02 rad
more than 0.1 rad apart after 10.2 s

```

$7e-05$ to w zapisie Pythona 7×10^{-5} , czyli siedem setysięcznych. Różnica nie rośnie gładko: wznosi się i opada w rytm kołysania ramion. Ale tendencja jest nieubłagana. Od milionowej do dziesiątej części to pięć mnożeń przez dziesięć w mniej więcej dziesięć sekund, więc średnio różnica mnoży się przez dziesięć co dwie sekundy: to efekt motyla z rozdziału 6, w maszynie, którą dałoby się zbudować na stole warsztatowym.



Rysunek 13.1: Dwa wahadła podwójne puszczone z różnicą milionowej części radiana. U góry: dolne ramię każdego z nich, nie do odróżnienia przez około dziesięć sekund. U dołu: różnica między nimi w skali logarytmicznej, rosnąca nierówno, aż staje się dość duża, by ją zobaczyć.

Oto wątpliwość, która powinna ci przyjść do głowy. Każdy krok jest odrobinę błędny: a co, jeśli rozejście wywołują te błędy, a nie wahadło? Wtedy dokładniejsze kroki by

je przesunęły. Uruchom więc te same dwa puszczenia z czterema długościami kroku, obserwując przy tym energię:

code/ch13/step_check.py

```

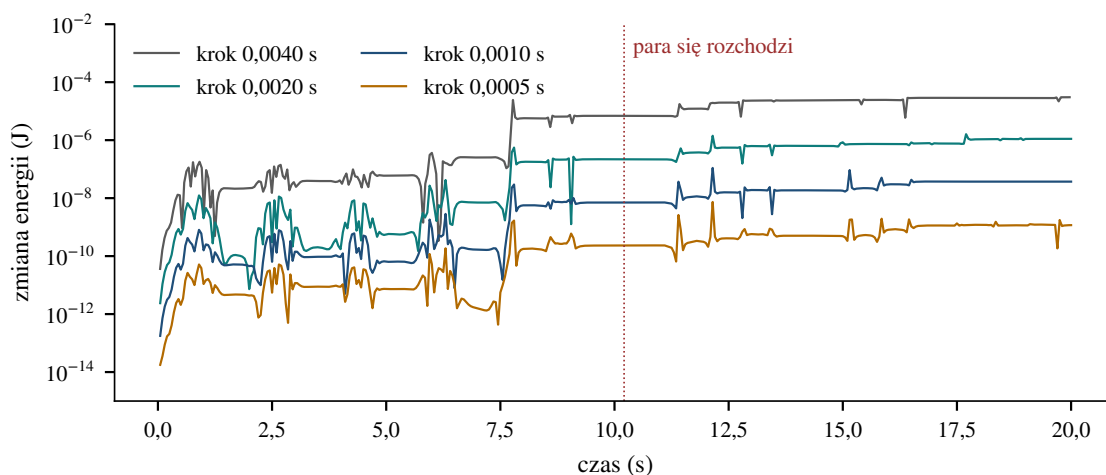
16 def parting(dt: float) -> tuple[float, float]:
17     """Seconds until the pair is APART, and how far the energy moved."""
18     a = release(120, 120)
19     b = (a[0] + NUDGE, a[1], a[2], a[3])
20     start = double_pendulum_energy(a)
21     worst, step = 0.0, 0
22     while gap(a, b) <= APART:
23         a, b = rk4_step(RULE, a, dt), rk4_step(RULE, b, dt)
24         step += 1
25         worst = max(worst, abs(double_pendulum_energy(a) - start))
26     return step * dt, worst

```

Listing 13.5: Ta sama para, z dowolną długością kroku

step 0.0040 s	apart after 10.21 s	energy moved 2.5e-05 J
step 0.0020 s	apart after 10.21 s	energy moved 1.7e-06 J
step 0.0010 s	apart after 10.21 s	energy moved 1.1e-07 J
step 0.0005 s	apart after 10.21 s	energy moved 6.8e-09 J

Każde skrócenie kroku o połowę dzieli błąd energii mniej więcej przez 16, tak jak obiecywał przepis RK4 z rozdziału 4, a chwila, w której para się rozchodzi, nie przesuwają się nawet o setną część sekundy. Przy kroku używanym w książce, 0,001 sekundy, energia zmienia się, zanim para się rozejdzie, o około 10^{-7} dżula: mniej więcej o jedną część na 10^8 . Rozejście należy do wahadła, nie do komputera.



Rysunek 13.2: Jak daleko energia w symulacji odchodzi od swojej prawdziwej, stałej wartości, dla czterech długości kroku. Każde skrócenie kroku o połowę obniża całą krzywą mniej więcej o ten sam czynnik, a linia kropkowana zaznacza chwilę, w której dwa puszczenia się rozchodzą, taką samą dla wszystkich czterech.

Uwaga

Ten sprawdzian ma granicę. Policz jedno puszczenie dwa razy, z krokiem książki i z jego połową, a te dwa obliczenia też się rozejdą, po około 15 sekundach. Drobnym błąd każdego kroku jest pchnięciem, a chaos powiększa je jak każde inne. Zaczyna się jednak od czegoś dużo mniejszego niż milionowa część radiana, dlatego nasza para rozchodzi się pierwsza, a każda długość kroku zgadza się co do tego, kiedy. Rozdział 16 pyta, czemu jeszcze można ufać potem.

Pułapka

„Wahadło, które zachowuje energię, bez tarcia i bez niczego losowego, nie może być chaotyczne.” Nie potrzebuje ani tarcia, ani przypadku. Zachowanie energii tylko częściowo trzyma wahadło w ryzach. Stan jednego wahadła to dwie liczby; ustalona energia zostawia mu jeden kierunek ruchu, wokół jednej zamkniętej pętli, i dlatego się powtarza. Stan wahadła podwójnego to cztery liczby; ustalona energia zostawia trzy kierunki, a trzy wystarczą do chaosu. Sprawdzian energii dowodzi, że symulacja jest uczciwa. Nie sprawia, że ruch staje się przewidywalny.

Laboratorium 13.2 – Kiedy dwa puszczenia się rozchodzą?

Zamień listing w funkcję. `seconds_until_apart` prowadzi krok po kroku dwa wahadła obok siebie i zwraca pierwszą chwilę, w której różnią się o więcej niż zadana tolerancja, albo `None`, jeśli wciąż są razem, gdy skończy się czas. Potem jej użyj: o ile dłużej rozchodzą się puszczenia różniące się o miliardową część radiana niż takie, które różnią się o milionową? Porównaj odpowiedź z arytmetyką horyzontów z rozdziału 7.

Plik startowy: `code/labs/ch13/l13_02_apart.py`

Test: `code/labs/ch13/test_l13_02_apart.py`

Uruchom: `cd code && uv run pytest -k l13_02_apart`

13.4 Łagodne kołysania i dzikie

Wszystko dotąd było puszczeniem z wysoka, z dużą energią. Wahadło podwójne można też puścić łagodnie.

Zatrzymaj się i pomyśl 13.4

Puść to samo wahadło z obydwoma ramionami pod kątem 10° zamiast 120° : ta sama reguła, ta sama maszyna, to samo pchnięcie o milionową część radiana. Czy różnica wciąż urośnie w ciągu dwudziestu sekund do czegoś, co każdy by zobaczył?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Przy 10° oba pozostają w odległości kilku milionowych części radiana od siebie: pchnięcie prawie nie rośnie. Ta sama maszyna jest spokojna przy małych wychyleniach i dzika przy dużych.

code/ch13/gentle_wild.py

```

19 def fate(angle: float) -> tuple[float | None, float]:
20     """When a pair let go at `angle` degrees parts (None if it does not),
21     and the largest gap it reached before then."""
22     a = release(angle, angle)
23     b = (a[0] + NUDGE, a[1], a[2], a[3])
24     largest = gap(a, b)
25     for step in range(1, round(SECONDS / DT) + 1):
26         a, b = rk4_step(RULE, a, DT), rk4_step(RULE, b, DT)
27         if gap(a, b) > APART:
28             return step * DT, largest
29     largest = max(largest, gap(a, b))
30     return None, largest

```

Listing 13.6: Co robi para puszczona pod jednym kątem

```

10 deg  together; largest gap 1e-06 rad
30 deg  together; largest gap 2e-06 rad
50 deg  together; largest gap 4e-06 rad
70 deg  together; largest gap 8e-06 rad
90 deg  apart after 16.9 s
110 deg apart after 12.1 s
130 deg apart after 8.2 s
150 deg apart after 10.1 s
170 deg apart after 5.4 s

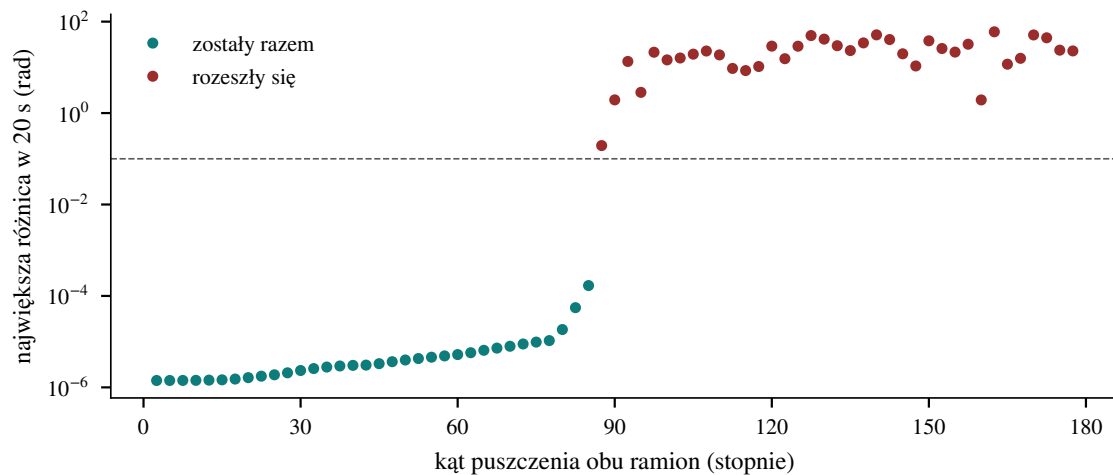
```

Aż do 70° każda para jest wciąż razem po 20 sekundach, a nawet przy 70° największa różnica wynosi 8×10^{-6} radiana: pchnięcie urosło kilka razy, a nie sto tysięcy razy. Od 90° w górę każda para się rozchodzi. Wykres wypełnia kąty pomiędzy.

Tak właśnie żyje chaos w układzie bez tarcia. Przy małej energii wahadło podwójne jest prawie dwoma prostymi kołysaniami, z ramionami poruszającymi się razem albo przeciwnie, i jego ruch jest regularny. Puść je z większej wysokości, a pojawia się chaos; z jeszcze większej, a chaos przejmuje władzę, a ruch regularny przetrwa tylko w zakamarkach: chaotyczne morze z wyspami ruchu regularnego.

Czego tu nie dowodzimy

Ta książka nie dowodzi, że wahadło podwójne jest chaotyczne; mierzy wrażliwość dla kilku puszcz. Twierdzenie stojące za morzem i jego wyspami nazywa się KAM, od Andrieja Kołmogorowa, który ogłosił je w 1954 roku, oraz Jürgena Mosera i Władimira Arnolda, którzy udowodnili jego wersje w latach 1962 i 1963. Mówi ono, że gdy regularny układ bez tarcia zostanie lekko zaburzony, większość jego regularnych ruchów przetrwa, nieco wygięta, a chaos zostaje zamknięty w cieniach



Rysunek 13.3: Każda kropka to puszczenie obu ramion pod jednym kątem, razem z kopią przesuniętą o milionową część radiana. Małe wychylenia pozostają razem przez dwadzieścia sekund; duże się rozchodzą, a zmiana następuje w przedziale kilku stopni.

warstwach między nimi; gdy zaburzenie rośnie, warstwy się poszerzają. Jego dowód to matematyka na poziomie wyższych lat studiów, do znalezienia w książkach o mechanice takich układów.

Jeszcze jedna różnica względem rozdziału 9. Równania Lorenza zawierają tarcie, a tarcie sprowadza każdy start na jeden atraktor. Wahadło podwójne nie ma tarcia: każdy start zachowuje swoją energię na zawsze i nie ma atraktora. Jest za to płot. Energia mówi, które stany w ogóle da się osiągnąć, a ruch, choćby najdzikszy, pozostaje w środku.

13.5 Dwa ciała, trzy ciała

Mechanizmem zegara z rozdziału 1 był Układ Słoneczny, a jego triumfem problem dwóch ciał: jednej planety i Słońca. Johannes Kepler odkrył na początku XVII wieku, że planety poruszają się po elipsach, a Isaac Newton pokazał w 1687 roku, że jego prawo grawitacji każe dwóm ciałom robić dokładnie to. Dwa ciała mają wzór: elipsę, zataczaną bez końca. Na każde pytanie o nie istnieje odpowiedź, którą da się zapisać.

Dodaj trzecie ciało, a wzoru takiego rodzaju w ogóle nie ma. W 1889 roku Henri Poincaré zdobył nagrodę ufundowaną w imieniu króla Szwecji Oskara II za pracę nad problemem kilku ciał poruszających się pod wpływem wzajemnej grawitacji. Gdy jego nagrodzoną rozprawę przygotowywano do druku, znaleziono w niej błąd. Poprawiając go, Poincaré odkrył, że orbity w pobliżu prostego, powtarzającego się ruchu trzech ciał mogą owijać się wokół niego w płataninę tak zawiłą, że później napisał, iż nawet nie będzie próbował jej narysować. Poprawiona rozprawa ukazała się w 1890 roku, a to, co opisuje, uznaje się dziś za pierwszy przebłysk chaosu: regułę równie dokładną jak reguła Newtona, a za nią ruch, za którym nie nadąży żaden wzór.

W 1908 roku, pisząc dla szerokiej publiczności, Poincaré ujął tę lekcję słowami, które ta książka właśnie mierzy: przyczyna zbyt mała, by ją zauważyć, może zdecydować o skutku, którego nie da się przeoczyć, i wtedy przewidywanie staje się niemożliwe.



Rysunek 13.4: Dwa ciała mają wzór; trzy nie mają żadnego; planety są chaotyczne w skali dziesiątek milionów lat, a mimo to pozostają na swoich miejscach.

13.6 Czy Układ Słoneczny jest zegarem?

Przez dwa stulecia Układ Słoneczny był wzorem doskonałego zegara. Pod koniec XVIII wieku Joseph-Louis Lagrange i Pierre-Simon Laplace obliczyli, z dokładnością, na jaką pozwalały ich metody, że rozmiary orbit planet się chwieją, ale nie dryfują. Demon Laplace’a z rozdziału 1 miał planety za dowód.

W 1989 roku Jacques Laskar obliczył powolne wahania orbit planet na przestrzeni dwustu milionów lat i stwierdził, że ruch planet wewnętrznych – Merkurego, Wenus, Ziemi i Marsa – jest chaotyczny. Czas Lapunowa, który znalazł, w sensie z rozdziału 7, wynosił około pięciu milionów lat: błąd położenia planet mnoży się przez e , około 2,7, co pięć milionów lat.

Zatrzymaj się i pomyśl 13.5

Sto milionów lat to dwadzieścia takich czasów Lapunowa. Mniej więcej ile razy powiększy się błąd w ciągu dwudziestu takich okresów: sto razy, milion razy czy miliard razy?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około $4,9 \times 10^8$ razy: bliżej miliarda niż miliona. W tym tempie błąd jednego metra w dzisiejszym położeniu Ziemi urósłby do 490 000 kilometrów, więcej niż odległość do Księżyca.

Pułapka

„Układ Słoneczny to doskonały zegar.” W skali kilku tysięcy lat jest zegarem tak dobrym jak cokolwiek, co mamy: zaćmienia przewiduje się na stulecia naprzód. W skali dziesiątek milionów lat nie da się go odczytać. Żaden pomiar i żaden komputer nie powie, gdzie na swojej orbicie będzie Ziemia za sto milionów lat. Reguła jest dokładna; stan początkowy nie.

Nieprzewidywalny to nie to samo co niestabilny. Wynik Laskara nie mówi, że planety się rozlecą, a późniejsze obliczenia, sięgające miliardów lat, w niemal każdym przebiegu pokazują planety na orbitach o mniej więcej tych samych rozmiarach i kształtach. Nie umiemy powiedzieć, gdzie na orbicie będzie Ziemia za sto milionów lat; umiemy powiedzieć, że najpewniej będzie na orbicie bardzo podobnej do obecnej. To lekcja z

rozdziału 9 w nowym miejscu: drogi nie da się przewidzieć, a obszar, po którym się porusza – tak.

Chaos ujawnia się szybciej w niektórych zakątkach Układu Słonecznego. Księżyc Saturna Hyperion to nieforemna bryła na wydłużonej orbicie, a obliczenia przewidują, że jego obrót nie może się ustalić, tylko koziółkuje chaotycznie, szarpany przez Saturna raz w jedną, raz w drugą stronę. Obserwacje z sond Voyager i Cassini zgadzają się z tym przewidywaniem: Hyperion nie ma stałej doby.

Gdzie to spotkasz

Geolodzy datują warstwy osadów według rytmów klimatu w nich zapisanych. Powolne cykle kształtu orbity Ziemi i nachylenia jej osi, nazwane od nazwiska Milutina Milankovicia, zmieniają to, jak światło słoneczne pada na planetę, a klimat zostawia w skałach pasujące do nich pasma. Żeby zamienić pasma w daty, trzeba znać orbitę Ziemi daleko w przeszłość, i geolodzy biorą ją z obliczeń zespołu Laskara. Tym obliczeniom ufa się, cykl po cyklu, wstecz do mniej więcej pięćdziesięciu milionów lat i nie dalej – z powodu chaosu opisanego w tym podrozdziale.

Podsumowanie

- Stan **wahadła podwójnego** to cztery liczby, dwa kąty i dwie prędkości ich zmian. Jego reguła pochodzi od Newtona, jest dokładna i nie ma w niej przypadku, a RK4 stosuje ją krok po kroku.
- Bez tarcia **energia** nie może się zmienić, więc energia symulacji mierzy błąd kroków: przy kroku książki około jednej części na 10^8 .
- Puszczania różniące się o milionową część radiana rozchodzą się po 10,2 sekundy przy każdej długości kroku: **rozejście to fizyka**. Zachowanie energii nie zapobiega chaosowi.
- Łagodne puszczenia są **regularne**, a dzikie chaotyczne: morze z wyspami, jak opisuje twierdzenie KAM. Bez tarcia nie ma atraktora; energia ogradza ruch.
- Dwa ciała mają wzór, a trzy nie. Poprawiona nagrodzona rozprawa **Poincarégo** z 1890 roku była pierwszym przebłyskiem chaosu.
- Laskar stwierdził, że planety wewnętrzne są chaotyczne, a ich **czas Lapunowa** wynosi około pięciu milionów lat: ich położenia nie da się przewidzieć na sto milionów lat naprzód, choć ich orbity pozostają mniej więcej takie, jakie są.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

przeprowadzić wahadło podwójne krok po kroku metodą RK4 i odczytać jego stan z wydruku	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
sprawdzić symulację układu bez tarcia, obserwując jego energię, i powiedzieć, czego ten sprawdzian dowodzi, a czego nie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zmierzyć, jak daleko od siebie odchodzą dwa niemal identyczne puszczenia, i sprawdzić, czy wynik zależy od długości kroku	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
odróżnić łagodne puszczenie od dzikiego i powiedzieć, co zachowanie energii przesądza o przyszłości układu, a czego nie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
oddzielić to, co zmierzono na temat chaosu w Układzie Słonecznym, od tego, co jest legendą	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dlaczego wahadło podwójne może być chaotyczne, a pojedyncze nie, choć żadne z nich nie traci energii?
2. Wahadło podwójne puszczone ze spoczynku z obydwoma ramionami skierowanymi pionowo w górę. Jaka jest jego energia, jeśli wysokości mierzymy od punktu zawieszenia?
3. Twoja symulacja maszyny bez tarcia traci w ciągu minuty jedną dziesiątą energii. Co z tego wnioskujesz?
4. Dwa puszczenia rozchodzą się po dziesięciu sekundach. Skracasz krok o połowę i teraz rozchodzą się po trzech. Co ci to mówi?
5. Wahadło podwójne nie ma atraktora. Co powstrzymuje jego ruch przed pójściem gdziekolwiek?
6. Przy czasie Lapunowa wynoszącym pięć milionów lat mniej więcej przez ile mnoży się błąd w ciągu dziesięciu milionów?
7. Czy chaos Układu Słonecznego oznacza, że planety się rozlecą?

Pogoda, klimat i prognoza zespołowa

Prognoza w twoim telefonie mówi, że jutro po południu szansa na deszcz wynosi 70 procent. Siedemdziesiąt procent czego? Nie nieba i nie popołudnia: nikt nie spodziewa się deszczu nad siedmioma dziesiątymi miasta przez siedem dziesiątych czasu. Ta liczba to wynik liczenia, a do końca tego rozdziału sam zrobisz taką prognozę na małym modelu atmosfery i sprawdzisz, czy można jej ufać.

Tu złe wieści z części II zamieniają się w metodę. Atmosfera jest chaotyczna, więc pojedyncza prognoza ma horyzont, a rozdział 7 pokazał, że lepsze pomiary przesuwają go tylko trochę. Synoptycy nie mogli zmienić żadnego z tych faktów. Zmienili pytanie, na które odpowiadają – z jednej prognozy na wiele – i jest to najbardziej udane praktyczne zastosowanie, jakie znalazła teoria chaosu.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- uruchomić prognozę zespołową na zabawkowej atmosferze i odczytać jej rozrzut jako miarę pewności
- zamienić zespół w prawdopodobieństwo i powiedzieć, jak ocenia się prognozę probabilistyczną
- zmierzyć, jak daleko naprzód prognoza wygrywa z podawaniem średniej klimatycznej
- użyć reguły z rozdziału 7, żeby powiedzieć, co lepszy start daje prognozie
- odróżnić pytanie o pogodę od pytania o klimat

14.1 Zabawkowa atmosfera

Model pogody dzieli powietrze na pudełka, każdemu przypisuje kilka liczb – temperaturę, ciśnienie, wiatr, wilgotność – i za pomocą fizyki płynów opisuje, jak każda z nich się zmienia. Pierwszą prognozę zrobioną w ten sposób na komputerze elektronicznym policzono na ENIAC-u w 1950 roku; dzisiejsze modele globalne niosą wiele milionów liczb.

W 1996 roku Edward Lorenz, którego trzy równania poznałeś w rozdziale 9, zaproponował model dość mały, żeby dało się o nim myśleć, i dość duży, żeby zachowywał się jak pogoda. Rozmieść 40 liczb, od x_1 do x_{40} , wokół równoleżnika tak, żeby każda oznaczała pogodę w jednym miejscu, a sąsiadem ostatniej była pierwsza. Lewą stronę czytaj tak, jak nauczył cię rozdział 4: jako to, jak szybko zmienia się x_i :

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F.$$

Słowami: każdą liczbę niosą jej sąsiedzi (iloczyn, który przesuwą pogodę wokół koła tak jak wiatr), każda wycieka (to $-x_i$, które gra rolę tarcia) i każdą pcha stały napęd F , *wymuszenie*, które gra rolę słońca. Standardowe wymuszenie Lorenza to $F = 8$, a w `chaoslab` cała reguła to jedna linijka:

code/src/chaoslab/systems.py

```

124 def lorenz96(forcing: float = 8.0) -> Field:
125     """Lorenz's 1996 toy atmosphere: n numbers round a circle of latitude.
126
127     Each one is pushed by its neighbours, damped, and driven by `forcing`.
128     The number of variables is the length of the state you pass in.
129     """
130
131     def field(s: State) -> State:
132         n = len(s)
133         return tuple((s[(i + 1) % n] - s[i - 2]) * s[i - 1] - s[i] + forcing
134                     for i in range(n))
135
136     return field

```

Listing 14.1: Czterdziestoliczbowa zabawkowa atmosfera Lorenza

Koło zamyka się samo: gdy i wynosi 0, Python czyta $s[i - 2]$ jako $s[-2]$, czyli drugą liczbę od końca.

Zatrzymaj się i pomyśl 14.1

Ustaw każdą liczbę na 8, czyli wartość wymuszenia. Co według reguły każda z nich zrobi dalej?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nic. Sąsiedzi się znoszą, $(8 - 8) \times 8 = 0$, a wyciek o 8 znosi pchnięcie o 8, więc każde tempo zmian wynosi zero.

Idealnie spokojna atmosfera pozostaje spokojna. Spokój jest punktem stałym i, jak w rozdziale 3, pytanie brzmi, czy jest stabilny. Nie jest: listingi poniżej trącą jedną liczbę o jedną setną i czekają, a spokój rozpada się na fale, które wędrują wokół koła i nigdy się nie uspokajają.

Lorenz dobrał jednostki tak, że jedna jednostka czasu to mniej więcej pięć dni prawdziwej pogody, więc krok listingów, 0,05, to około sześciu godzin. Zmierzony funkcją `lyapunov_flow` z `chaoslab` wykładnik z rozdziału 7 wynosi 1,70 na jednostkę czasu: mały błąd podwaja się co 0,41 jednostki, czyli mniej więcej co dwa dni.

14.2 Dwadzieścia prognoz zamiast jednej

Prognoza zaczyna się od *analizy*: najlepszego oszacowania obecnego stanu atmosfery, zbudowanego z pomiarów, które nigdy nie są dokładne. Powiedzmy, że analiza jest trafna z dokładnością do 0,1 dla każdej z czterdziestu liczb. Z rozdziału 6 wiesz, co robi

pojedyncza prognoza z niej wystartowana: przez jakiś czas będzie podążać za prawdziwą pogodą, a potem przestanie.

Służba meteorologiczna nie zna jutrzejszej prawdy w chwili, gdy prognozuje. My ją znamy, bo sami gramy rolę natury. Uruchom model raz i nazwij ten przebieg *prawdą*; zrób analizę, dodając do prawdy błąd; prognozuj z analizy; oceń prognozę, porównując ją z prawdą. Model i prawda podlegają tej samej regule, więc każdy błąd poniżej pochodzi ze startu, a żaden z modelu: zabawka ma wszystkie przewagi, których brakuje prawdziwej prognozie.

Teraz nowy pomysł. Zamiast jednej prognozy zrób ich 20. Każdy *członek* startuje z analizy trąconej kolejnym błędem tej samej wielkości, więc członkowie różnią się od siebie mniej więcej tak, jak analiza mogłaby się różnić od prawdy. Razem tworzą *zespół*.

code/ch14/ensemble.py

```

20 N = 40          # numbers round the circle of latitude
21 FORCING = 8.0   # the drive: Lorenz's standard value
22 DT = 0.05       # one step; about six hours in Lorenz's units
23 ERROR = 0.1     # the analysis is right to within 0.1 on every number
24 MEMBERS = 20    # copies in the ensemble
25
26
27 def run(state: State, steps: int, forcing: float = FORCING) -> State:
28     """Follow the toy atmosphere for a number of steps."""
29     field = lorenz96(forcing)
30     for _ in range(steps):
31         state = rk4_step(field, state, DT)
32     return state
33
34
35 def spin_up(forcing: float = FORCING, nudge: float = 0.01) -> State:
36     """A calm atmosphere with one number nudged, left to make weather."""
37     calm = (forcing + nudge,) + (forcing,) * (N - 1)
38     return run(calm, 2000, forcing)
39
40
41 def perturb(state: State, size: float, rng: random.Random) -> State:
42     """Add an error no bigger than `size` to every number."""
43     return tuple(x + rng.uniform(-size, size) for x in state)

```

Listing 14.2: Zabawkowa atmosfera, spokojny start i błąd zadanej wielkości

`rng.uniform(-size, size)` losuje liczbę między `-size` a `size`, każdą wartość z tą samą szansą, więc `perturb` dodaje do każdej liczby inny mały błąd. Resztę robią trzy miary. *Średnia zespołu* uśrednia każdą liczbę po członkach. *Rozrzut* to typowa odległość członków od tej średniej: podnieś każdą różnicę do kwadratu, uśrednij kwadraty, wyciągnij pierwiastek. *Błąd* to ten sam przepis zastosowany do prognozy i prawdy. Rozrzut i błąd są więc liczbami tego samego rodzaju, mierzonymi względem siebie nawzajem albo względem prawdy.

Zatrzymaj się i pomyśl 14.2

Członkowie startują w odległości kilku setnych od siebie. Czy w miarę biegu prognozy ich rozrzut rośnie bez końca?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Rośnie szybko, a potem się wyrównuje: członkowie nie mogą rozejść się bardziej niż dwa niezwiązane ze sobą dni pogody, a atraktor ogranicza, jak bardzo te mogą się różnić.

code/ch14/ensemble.py

```

73 def ensemble(truth: State, rng: random.Random) -> tuple[State, list[State]]:
74     """An analysis of `truth`, and MEMBERS copies of it, each nudged."""
75     analysis = perturb(truth, ERROR, rng)
76     return analysis, [perturb(analysis, ERROR, rng) for _ in range(MEMBERS)]
77
78
79 def forecast(truth: State, rng: random.Random,
80             leads: range) -> list[tuple[float, float, float, float]]:
81     """Score an ensemble forecast at each lead (counted in steps): the
82     spread, the error of the ensemble mean, and the error of one run
83     from the analysis alone."""
84     single, members = ensemble(truth, rng)
85     done, rows = 0, []
86     for lead in leads:
87         gap = lead - done
88         truth = run(truth, gap)
89         single = run(single, gap)
90         members = [run(m, gap) for m in members]
91         done = lead
92         rows.append((lead * DT, spread(members),
93                     error(mean(members), truth), error(single, truth)))
94     return rows

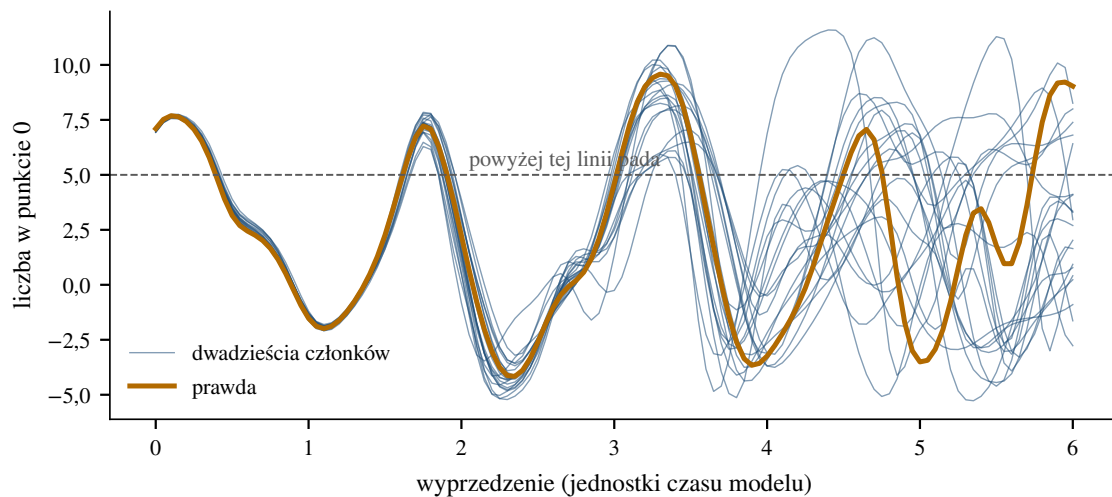
```

Listing 14.3: Prognoza zespołowa oceniana przy każdym wyprzedzeniu

lead	spread	error of the mean	error of one run
0.0	0.06	0.06	0.06
1.0	0.42	0.35	0.31
2.0	1.53	1.35	1.03
3.0	2.49	2.38	3.78
4.0	3.10	2.89	3.93
5.0	3.32	3.58	5.01
6.0	3.44	3.69	4.74
7.0	3.54	3.76	5.34
8.0	3.54	3.18	4.49

Wyprzedzenie, w tabeli „lead”, to to, jak daleko naprzód patrzy prognoza, w jednostkach czasu. Przez pierwszą jednostkę czy dwie rozrzut jest mały, a średnia zespołu blisko

prawdy; do szóstej rozrzut przestaje rosnać, bo członkowie zapomnieli, skąd wystartowali. Pojedynczy przebieg z analizy, w ostatniej kolumnie, jest na początku tak dobry jak średnia, a później dużo gorszy. Oto punkt 0 dla każdego członka, narysowany na tle prawdy:



Rysunek 14.1: Dwudziestu członków powyższej prognozy w punkcie 0, z prawdą na pomarańczowo. Przez prawie trzy jednostki czasu poruszają się niemal jak jeden, potem się rozchodzą; do piątej są rozrzućeni po całym zakresie pogody.

Zatrzymaj się i pomyśl 14.3

W prawdziwej prognozie nie widzisz prawdy, więc nie policzysz dwóch kolumn z błędami. Spójrz na tabelę: czy rozrzut, który policzyć możesz, mógłby ci powiedzieć, jak duży jest błąd średniej?

Odpowiedz, zanim pójdziesz dalej.

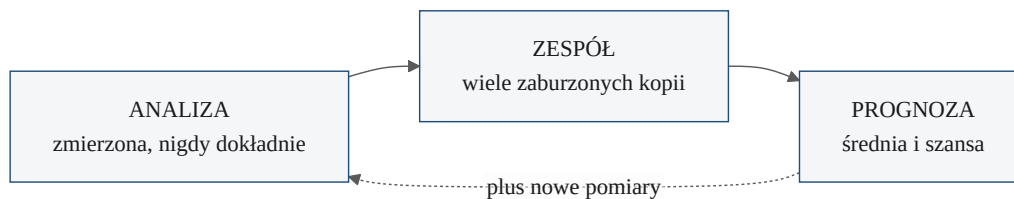
Odpowiedź. Tak, w przybliżeniu. Uśrednione po 40 prognozach z różnych dni rozrzut i błąd średniej wynoszą 0,47 i 0,45 przy wyprzedzeniu jednej jednostki czasu oraz 2,84 i 2,92 przy trzech.

Prawda jest, na ile ktokolwiek może to stwierdzić, jeszcze jednym członkiem: analiza różni się od niej błędem tej samej wielkości co trącenia członków. Leży więc mniej więcej tak daleko od średniej zespołu jak oni, a szerokość wachlarza mierzy wielkość pomyłki, choć nikt nie widzi prawdy.

Pułapka

„Prognoza to jedna liczba.” Jedna liczba to najmniej, co prognoza może ci powiedzieć. Zespół podaje liczbę razem z ostrzeżeniem: w dniu, w którym członkowie się zgadzają, ufaj średniej; w dniu, w którym się rozchodzą, ta sama liczba zasługuje na wzruszenie ramion. Szerokość wachlarza zmienia się z dnia na dzień razem z samą pogodą i mówi, jak daleko naprzód prognozę warto czytać.

Ośrodek prognoz robi to codziennie i zamyka pętlę: dzisiejszą analizę tworzy się, łącząc nowe pomiary z wczorajszą prognozą.



Rysunek 14.2: Cykl prognozy. Niedokładna analiza staje się wieloma zaburzonymi kopiami; ich rozrzut i ich liczenie stają się prognozą; jutro prognoza spotyka nowe pomiary i staje się następną analizą.

14.3 Szansa na deszcz

Teraz te siedemdziesiąt procent. Nazwij punkt 0 swoim miastem i przyjmij, że pada tam zawsze, gdy jego liczba przekracza 5. Liczby Lorenza nie są deszczem; to reguła zamiany liczby na pogodę i nic poniżej nie zależy od tego wyboru. W długim przebiegu pada w 25 procentach przypadków: to klimat twojego miasta, zmierzony w ostatnim podrozdziale.

Zatrzymaj się i pomyśl 14.4

Przy pewnym wyprzedzeniu 15 z 20 członków pokazuje deszcz w twoim mieście. Jaka jest prognozowana szansa na deszcz?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. 75 procent: 15 z 20 to trzy czwarte.

Tym właśnie jest szansa na deszcz: ułamkiem jednakowo prawdopodobnych przyszłości, w których pada. Oto ona dla prognozy, którą śledzisz.

code/ch14/chance_of_rain.py

```

15 RAIN = 5.0 # it rains at a site whenever its number is above this
16
17
18 def chance(members: list[State], site: int = 0) -> float:
19     """The fraction of the members in which it rains at `site`."""
20     wet = sum(1 for state in members if state[site] > RAIN)
21     return wet / len(members)
  
```

Listing 14.4: Szansa na deszcz to liczba członków

lead	members	raining	chance	did it rain?
0.5	0	of 20	0%	no
1.0	0	of 20	0%	no
1.5	0	of 20	0%	no
2.0	0	of 20	0%	no
2.5	0	of 20	0%	no
3.0	3	of 20	15%	no
3.5	15	of 20	75%	yes
4.0	1	of 20	5%	no
4.5	5	of 20	25%	yes
5.0	7	of 20	35%	no
5.5	6	of 20	30%	no
6.0	5	of 20	25%	yes

Przez kilka pierwszych jednostek czasu każdy członek mówi „sucho” i jest sucho. Potem się dzieli. Przy jednym wyprzedzeniu trzech członków na czterech mówi „deszcz” i pada; później mówi tak tylko jeden na czterech, a pada i wtedy. Pod koniec szanse krążą wokół klimatu twojego miasta: prognoza zapomniała swój start.

Laboratorium 14.1 – Szansa na deszcz i rozliczanie prognoz

Napisz `chance`, która zamienia listę członków w prawdopodobieństwo, oraz `hit_rate`, która ocenia wiele prognoz naraz: jaka część tych, które dały szansę z danego przedziału, zakończyła się deszczem? Jeden z testów uruchamia prognozę z tego rozdziału i oczekuje, że twoja `chance` zgodzi się z tabelą powyżej.

Plik startowy: `code/labs/ch14/l14_01_chance.py`

Test: `code/labs/ch14/test_l14_01_chance.py`

Uruchom: `cd code && uv run pytest -k l14_01_chance`

Zatrzymaj się i pomyśl 14.5

Pod koniec tabeli zespół dał deszczowi jedną szansę na cztery i spadł deszcz. Czy prognoza była błędna?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Z jednego dnia tego nie rozstrzygniesz. Szansa jeden na cztery mówi, że w mniej więcej ćwierci takich dni pada, a ten może być jednym z nich.

Pułapka

„Prognoza 30 procent szansy na deszcz, po której spadł deszcz, była błędna.”

Dobra prognoza tego rodzaju sprawdza się mniej więcej trzy razy na dziesięć i to był jeden z tych razów. Błędną uczyniłaby ją dopiero historia: gdyby na setki prognoz 30 procent padało po 10 procentach z nich albo po 60. Synoptycy nazywają to sprawdzenie *wiarygodnością* i można je przeprowadzić tylko na wielu prognozach, nigdy na jednej.

Prowadź więc rachunek. Pomiar z tego rozdziału robi 6000 prognoz, w każdym punkcie i w wielu różnych dniach, każdą na dwie jednostki czasu naprzód. Spośród 380 przypadków, w których zespół dał deszczowi szansę od 25 do 35 procent, padało w 31 procentach. Spośród 243, w których dał od 65 do 75 procent, padało w 68 procentach. Procenty znaczą to, co mówią.

Gdzie to spotkasz

Każdego dnia Europejskie Centrum Prognoz Średnioterminowych z siedzibą w Reading uruchamia zespół około pięćdziesięciu członków swojego modelu globalnego, każdy z nieco innej analizy, na piętnaście dni naprzód. Służby meteorologiczne na całym świecie zamieniają takie zespoły, często po statystycznej korekcie, w procenty w twoim telefonie i w ostrzeżenia przed burzami i powodzią. Zespoły liczy się codziennie od ponad trzydziestu lat, a szansa na deszcz w twoim telefonie jest za kulisami liczeniem bardzo podobnym do tego z listingu powyżej.

14.4 Jak daleko naprzód?

Jedna prognoza nie potrzebuje żadnego modelu: zawsze podawaj klimat. Jutro w dowolnym punkcie liczba wyniesie około 2,3, plus minus 3,6 – to długoterminowa średnia i typowe wahanie, które mierzy ostatni podrozdział. Taka prognoza nigdy nie jest dobra, ale jej typowy błąd nigdy nie jest większy niż to wahanie. Prognoza ma *sprawdzalność* tylko dopóty, dopóki z nią wygrywa.

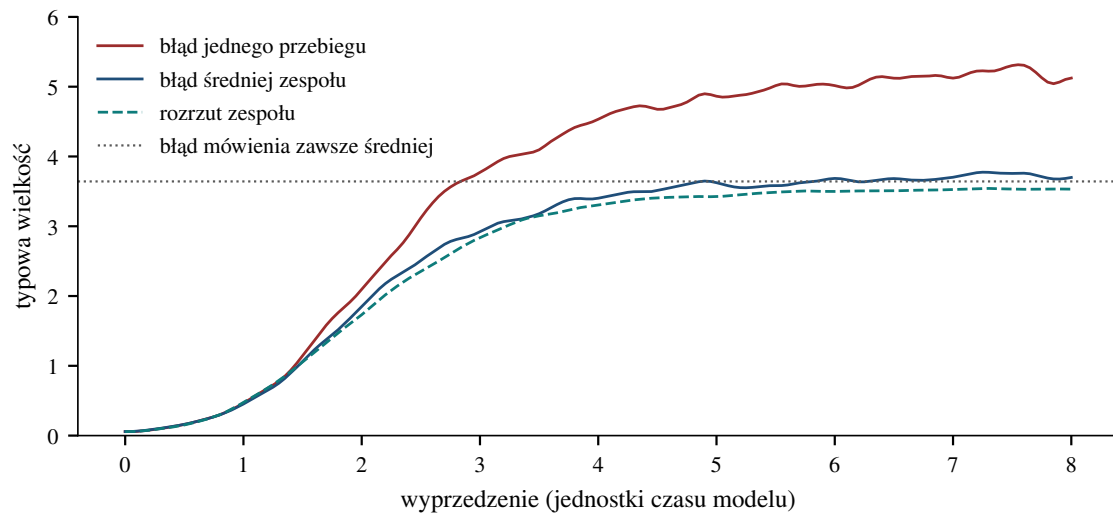
Zatrzymaj się i pomyśl 14.6

Długo po horyzoncie pojedynczy przebieg z analizy jest jeszcze jednym dniem pogody bez związku z prawdą. Która prognoza jest wtedy lepsza: ten przebieg czy średnia klimatyczna?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Średnia klimatyczna. Dwa niezwiązane dni pogody różnią się bardziej, niż każdy z nich różni się od średniej: o mniej więcej $\sqrt{2}$, czyli 1,4, typowego wahan.

Zmierzony błąd pojedynczego przebiegu ustala się na 5,1, a gorszy od średniej klimatycznej staje się przy wyprzedzeniu 2,85 jednostki czasu – w skali Lorenza po mniej więcej



Rysunek 14.3: Rozrzut i błędy w zależności od wyprzedzenia, każde uśrednione po czterdziestu prognozach. Rozrzut (linia przerywana) nadąża za błędem średniej zespołu; pojedynczy przebieg kończy wyraźnie powyżej linii kropkowanej, czyli błędowi mówienia zawsze średniej.

dwóch tygodniach. Prognoza pewna siebie i błędna jest gorsza od takiej, która nic nie wie. Średnia zespołu nigdy nie przekracza tej linii: gdy członkowie zapominają swój start, sama dryfuje ku średniej klimatycznej, a jej błąd ustala się na 3,7, odrobinę powyżej wahania, bo dwudziestu członków to nie nieskończenie wielu.

Laboratorium 14.2 – Kiedy zespół przestaje się rozchodzić?

Napisz `spread` dla całego zespołu, a potem `saturation_lead`: pierwsze wyprzedzenie, przy którym rozrzut osiąga dziewięć dziesiątych poziomu, na którym się ustala, a który uśredniony po czterdziestu prognozach wynosi 3,5. Test przeprowadza prognozę z tego rozdziału przez twój kod. Jedna prognoza to jeden rzut kostką, więc spodziewaj się, że twoje wyprzedzenie będzie się różnić od średniej z czterdziestu prognoz, 3,60 jednostki czasu.

Plik startowy: `code/labs/ch14/l14_02_saturation.py`

Test: `code/labs/ch14/test_l14_02_saturation.py`

Uruchom: `cd code && uv run pytest -k l14_02_saturation`

Zatrzymaj się i pomyśl 14.7

Przypuśćmy, że analiza jest dziesięć razy lepsza, trafna z dokładnością do jednej setnej zamiast jednej dziesiątej. Korzystając z reguły z rozdziału 7, powiedz, o ile mniej więcej później pojedynczy przebieg przegra z klimatem.

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniej więcej o $\ln 10$ podzielone przez wykładnik, niezależnie od błędu startowego: $\ln 10 \approx 2,3$, a po podzieleniu przez 1,70 daje to 1,35 jednostki czasu.

W pomiarze horyzont przesuwają się z 2,85 na 4,25, a przy analizie jeszcze dziesięć razy lepszej na 5,75: zyski 1,40 i 1,50, bliskie regule. Każda dziesięciokrotna poprawa kupuje ten sam dodatkowy odcinek, nigdy dziesięć razy dłuższy.

Prawdziwa atmosfera jest mniej hojna niż zabawka, a Lorenz wyjaśnił dlaczego w 1969 roku. Ma ruchy w wielu skalach, od szerokości oceanu aż po podmuch; małe psują się najszybciej, a ich błędy rozchodzą się w górę, do dużych. Lepsze pomiary opóźniają szkodę, ale jej nie zatrzymują. Jego wniosek, wciąż powszechnie przytaczany, brzmi: szczegółowej pogody nie da się przewidzieć dalej niż mniej więcej dwa tygodnie naprzód, choćby dane były najlepsze. Prognozy mimo to stale się poprawiały – lepsze obserwacje, także z satelitów, lepsze modele, większe komputery – ale zysk ocenia się zwykle na mniej więcej jeden dzień użytecznego wyprzedzenia na dekadę: to logarytm z reguły z rozdziału 7, splecany dekadami.

Czego tu nie dowodzimy

Liczba dwóch tygodni nie jest w tej książce wyprowadzona, a horyzont zabawki, który przesuwają się za każdym razem, gdy zmienia się jej błąd analizy, nie jest jej pomiarem. Pochodzi z oszacowań, jak szybko rosną błędy w każdej skali prawdziwej atmosfery, zapoczątkowanych przez Lorenza w 1969 roku i poprawianych od tamtej pory przez porównywanie prognoz z tym, co potem się wydarzyło. Zabawka pokazuje, dlaczego horyzont istnieje.

14.5 Pogoda i klimat

Skoro nikt nie potrafi powiedzieć, czy za trzy tygodnie będzie padać w twoim mieście, to czy ktokolwiek potrafi powiedzieć, jak często będzie tam padać w przyszłym roku?

Zatrzymaj się i pomyśl 14.8

Uruchom model dwa razy z różnych startów, za każdym razem na 500 jednostek czasu, czyli kilka lat pogody Lorenza. Po kilku pierwszych jednostkach ich pogoda nie ma ze sobą nic wspólnego. Czy ich długoterminowe średnie, typowe wahania i to, jak często pada, też będą się różnić?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Oba przebiegi dają średnią 2,3, wahanie 3,6 i deszcz w 25 procentach przypadków, z dokładnością do wydrukowanych cyfr, podczas gdy w ostatnim kroku punkt 0 pokazuje zupełnie różne liczby.

code/ch14/climate.py

```

22 def climate(forcing: float, nudge: float,
23             steps: int = 10000) -> tuple[float, float, float, float]:
24     """Average, typical swing and rain frequency over a long run, and
25     the number at site 0 when the run ends."""
26     state = spin_up(forcing, nudge)
27     field = lorenz96(forcing)
28     total = squares = 0.0
29     wet = count = 0
30     for _ in range(steps):
31         state = rk4_step(field, state, DT)
32         for x in state:
33             total += x
34             squares += x * x
35             if x > RAIN:
36                 wet += 1
37             count += 1
38     average = total / count
39     swing = math.sqrt(squares / count - average * average)
40     return average, swing, wet / count, state[0]

```

Listing 14.5: Długi przebieg sprowadzony do trzech statystyk

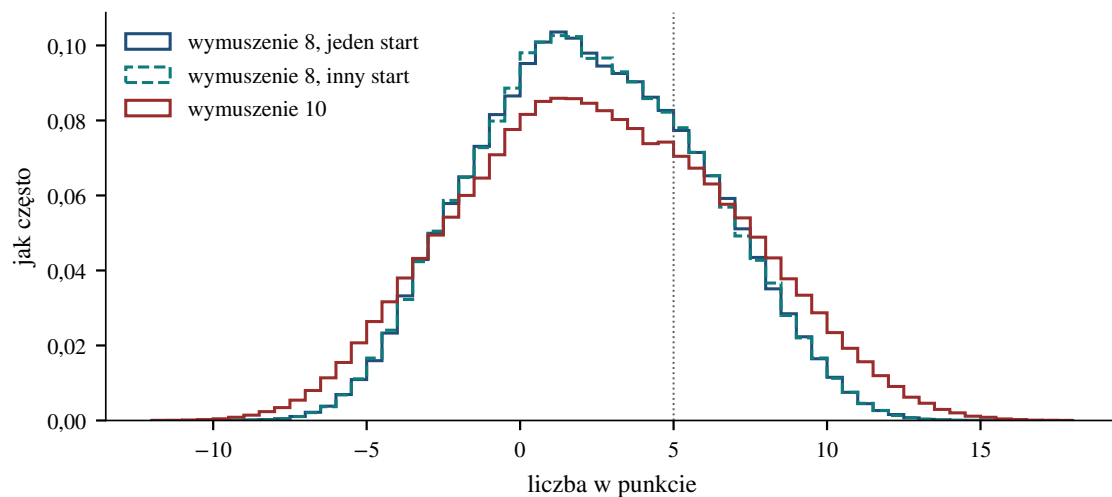
forcing	nudge	average	swing	rain	last x[0]
8	0.01	2.3	3.6	25%	1.44
8	1.00	2.3	3.6	25%	-2.06
10	0.01	2.6	4.4	30%	10.70

To odkrycie z rozdziału 9 na większym modelu: trajektoria jest nieprzewidywalna, atraktor nie, a to, jak często trajektoria odwiedza każdy obszar, należy do reguły, a nie do startu. Pogoda to trajektoria. Klimat to statystyka atraktora. Prognozowanie pogody to *zagadnienie początkowe*: odpowiedź zależy od tego, skąd startujesz, a chaos daje jej horyzont. Klimat od startu nie zależy wcale.

Pułapka

„Skoro nie da się przewidzieć pogody, nie da się też przewidzieć klimatu.” To odpowiedzi na różne pytania. Chaos ogranicza pytania o trajektorię: czy tego dnia będzie padać? Pytania o statystykę – jak często pada, jak szeroko waha się pogoda – mają stabilne odpowiedzi, a dwa przebiegi powyżej nie zgadzają się co do żadnego dnia, zgadzając się co do klimatu.

Klimat zmienia natomiast reguła. Trzeci wiersz wydruku podkreśla wymuszenie z 8 do 10, czyli daje zabawce więcej słońca, i statystyki się przesuwają: średnia wynosi teraz 2,6, wahanie 4,4, a pada w 30 procentach przypadków zamiast w 25. Pogoda w dowolnym pojedynczym dniu jest dokładnie tak samo nieprzewidywalna jak przedtem. Tak wygląda projekcja klimatyczna. Model klimatu uruchamia się nie od dzisiejszej pogody, lecz przy zmienionym wymuszeniu – w prawdziwym przypadku przy zmienionym składzie atmosfery – i pyta się go, jak przesuwają się statystyki. Nazywa się to *projekcją*,



Rysunek 14.4: Jak często każda wartość pojawia się w długim przebiegu. Dwa przebiegi z wymuszeniem 8, z różnych startów, leżą jeden na drugim; przy wymuszeniu 10 rozkład się poszerza i więcej go leży powyżej linii deszczu przy 5.

a nie przewidywaniem, bo wynik zależy od przyjętego wymuszenia. Wymuszenie jest *warunkiem brzegowym*: należy do reguły, a nie do dzisiejszego stanu, więc horyzont pogody nie wyklucza projekcji. Czy konkretny model klimatu dobrze oddaje to przesunięcie, to inne pytanie, rozstrzygane przez sprawdzanie go na klimatach przeszłości, i ta zabawka go nie rozstrzygnie. Pokazuje za to logikę: warunek początkowy decyduje o pogodzie, a warunek brzegowy o klimacie.

Podsumowanie

- **Zabawkowa atmosfera** Lorenza z 1996 roku rozmieszcza czterdzieści liczb wokół koła; pchają je sąsiedzi, tłumi wyciek, a napędza **wymuszenie**. Jej stan spokoju jest niestabilny, a błędy podwajają się w niej mniej więcej co 0,41 jednostki czasu.
- **Zespół** to wiele prognoz z trąconych kopii **analizy**. Jego **rozrzut** rośnie z wyprzedzeniem, wyrównuje się i nadaża za błędem średniej zespołu, choć nikt nie widzi prawdy.
- **Szansa na deszcz** to ułamek członków, w których pada. Prognozę probabilistyczną ocenia się po **wiarygodności**, na wielu prognozach, nigdy na jednym dniu.
- Prognoza ma **sprawdzalność**, dopóki wygrywa ze średnią klimatyczną. Pojedynczy przebieg przegrywa z nią na **horyzoncie** i kończy mniej więcej $\sqrt{2}$ razy gorzej; każda dziesięciokrotnie lepsza analiza przesuwu horyzont o tyle samo, a powszechnie przytaczana granica dla prawdziwej pogody to około dwóch tygodni.
- **Pogoda** to trajektoria i zależy od startu; **klimat** to statystyka atraktora i od startu nie zależy. Zmiana wymuszenia, **warunku brzegowego**, przesuwu statystyki i właśnie o to pyta **projekcja klimatyczna**.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

uruchomić prognozę zespołową na zabawkowej atmosferze i odczytać jej rozrzut jako miarę pewności	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zamienić zespół w prawdopodobieństwo i powiedzieć, jak ocenia się prognozę probabilistyczną	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zmierzyć, jak daleko naprzód prognoza wygrywa z podawaniem średniej klimatycznej	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
użyć reguły z rozdziału 7, żeby powiedzieć, co lepszy start daje prognozie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
odróżnić pytanie o pogodę od pytania o klimat	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Czym jest analiza i dlaczego prognoza z niej psuje się nawet przy doskonałym modelu?
2. W zespole liczącym 50 członków 12 pokazuje deszcz. Jaka jest szansa na deszcz?
3. W ciągu roku synoptyk powiedział „80 procent szansy na deszcz” 100 razy, a padało 55 razy. Czy te prognozy były wiarygodne?
4. Dlaczego długo po horyzoncie pojedyncza prognoza jest gorsza od podawania średniej klimatycznej i mniej więcej o ile?
5. W zabawkowej atmosferze błąd analizy zmniejszono sto razy. O ile mniej więcej przesuwa się horyzont?
6. Pogoda czy klimat: czy w przyszłe Boże Narodzenie spadnie śnieg w Warszawie? Ile śnieżnych grudniowych dni będzie miała Warszawa około 2050 roku przy zadanym wymuszeniu?

Życie, rynki i szum: jak odróżnić chaos od przypadku

Kardiolog patrzy na odstępy między uderzeniami serca, epidemiolog na sto lat comiesięcznych zachorowań na odrę, inwestor na cenę akcji. Każdy widzi poszarpaną linię, która nigdy dokładnie się nie powtarza, i chce wiedzieć, co nią szarpie: reguła, w której kilka liczb popycha się nawzajem, czy przypadek, czyli niezliczone drobne przyczyny, których żaden model nigdy nie wymieni. Najczęściej jedno i drugie.

Rozdział 5 pokazał, że reguła długości jednej linijki potrafi dawać wyniki nieregularne jak losowanie totolotka. Ten rozdział zadaje to pytanie od drugiej strony: gdy ktoś poda ci poszarpany szereg liczb bez żadnej reguły, czy umiesz stwierdzić, czy zrobiła go reguła? Zbudujesz dwa testy, zmierzysz, jak dobrze działają na danych zaszumionych i krótkich, a potem odwiedzisz miejsca, w których ogłaszano odkrycie chaosu. Przesłanie nie brzmi: chaos jest wszędzie, ani: nie ma go nigdzie. Brzmi: twierdzenie o chaosie da się sprawdzić.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- zrobić szereg chaotyczny i szereg losowy z dokładnie tych samych liczb i powiedzieć, dlaczego żaden histogram ich nie rozróżni
- narysować odwzorowanie powrotu, policzyć kratki, w które trafia, i odczytać ten wynik
- prognozować szereg z jego własnej przeszłości metodą analogów i zmierzyć, jak błąd rośnie z liczbą kroków naprzód
- powiedzieć, jak szum pomiarowy i krótki zapis osłabiają oba testy
- ocenić twierdzenie o chaosie w populacjach, epidemiach, sercach czy na rynkach na podstawie przedstawionych dowodów

15.1 Dwa szeregi, jeden zestaw liczb

Zacznij od zagadki. Dwa szeregi zawierają po 2000 liczb z przedziału od 0 do 1. Jeden zrobiła reguła logistyczna z rozdziału 5 z pokrętle ustawionym na $r = 4$: jeśli x_n to wartość w kroku n , to następna wynosi

$$x_{n+1} = 4x_n(1 - x_n),$$

czyli cztery razy wartość razy jeden minus wartość. Drugi szereg jest losowy. Poniżej oba są wydrukowane z dokładnością do czterech miejsc po przecinku; program, który je zrobił, przechowuje około szesnastu cyfr znaczących.

Zatrzymaj się i pomyśl 15.1

Szereg A zaczyna się od 0,0169, 0,8488, 0,6610, 0,6086. Szereg B zaczyna się od 0,2000, 0,6400, 0,9216, 0,2890. Który zrobiła reguła? Czy da się to poznać na oko, a czy da się to policzyć?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. B. Na oko się nie da. Rachunkiem tak:

$4 \times 0,2 \times 0,8 = 0,64$, czyli druga wartość B, a $4 \times 0,64 \times 0,36$ to 0,9216, jej trzecia. Dla A reguła daje z pierwszej wartości około 0,07, a druga wartość wynosi 0,8488.

Zwróć uwagę, czego użył ten test. Nie wielkości liczb ani tego, jak często pojawia się każda wielkość, tylko ich *kolejności*: czy każda wartość wynika z poprzedniej. O tym jest cały ten rozdział.

Pułapka

„Ciąg, który wygląda na losowy, musi pochodzić z losowości.” Szereg B wyglądał na równie losowy jak A, a powstał z jednej linijki arytmetyki bez grama przypadku. To, że coś wygląda losowo, jest faktem o wyglądzie. Nie mówi nic o przyczynie, dopóki nie sprawdzisz, czy jest w nim porządek, jaki zostawia po sobie reguła.

Szereg A nie powstał z rzutów kostką. Powstał z szeregu B przez przetasowanie:

code/ch15/twins.py

```

20 def rule_series(n: int = N, x0: float = X0) -> list[float]:
21     """n values of the logistic map x -> 4x(1 - x), starting at x0."""
22     return orbit(logistic(4.0), x0, n - 1)
23
24
25 def shuffled(xs: list[float], seed: int = SEED) -> list[float]:
26     """The same numbers in a random order: a surrogate."""
27     copy = list(xs)
28     random.Random(seed).shuffle(copy)
29     return copy

```

Listing 15.1: Dwa szeregi z jednego: reguła i te same liczby przetasowane

Pierwsza funkcja iteruje regułę; druga tasuje kopię listy generatorem liczb losowych Pythona, uruchomionym od ustalonego *ziarna*, dzięki czemu tasowanie jest takie samo na każdym komputerze, a twój wynik zgadza się ze stroną. Szereg zbudowany z innego przez pomieszanie kolejności nazywa się *surogatem*: to losowy szereg najtrudniejszy do odróżnienia od oryginału, bo ma dokładnie te same liczby.

Zatrzymaj się i pomyśl 15.2

Jak histogram przetasowanego szeregu – ile jego wartości wpada do każdej piątej części przedziału od 0 do 1 – wypadnie w porównaniu z histogramem szeregu reguły?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Identycznie, słupek w słupek. Histogram liczy wartości i pomija ich kolejność, a tasowanie zmienia tylko kolejność.

```
rule shuffled
0.2000 0.0169
0.6400 0.8488
0.9216 0.6610
0.2890 0.6086
0.8219 0.6066
counts in five slices of width 0.2:
rule      [582, 282, 250, 289, 597]
shuffled [582, 282, 250, 289, 597]
```

Spójrz na kształt: 582 spośród 2000 wartości leży w najniższej piątej części, a tylko 250 w środkowej, bo reguła lubi przebywać blisko 0 i 1. Przetasowany szereg ma ten sam przekrzywiony histogram, a także tę samą średnią, ten sam rozrzut, tę samą największą i najmniejszą wartość. Żadne podsumowanie, które pomija kolejność, nie odróżni tych bliźniaków.

15.2 Każda wartość względem następnej

W zagadce mogłeś sprawdzać każdą wartość względem poprzedniej, bo znałeś regułę. Przy prawdziwych danych nikt ci jej nie poda, więc niech dane same ją narysują. Postaw punkt w (x_n, x_{n+1}) dla każdego kroku n : w poziomie ta wartość, w pionie następna. Taki rysunek nazywa się *odwzorowaniem powrotu*.

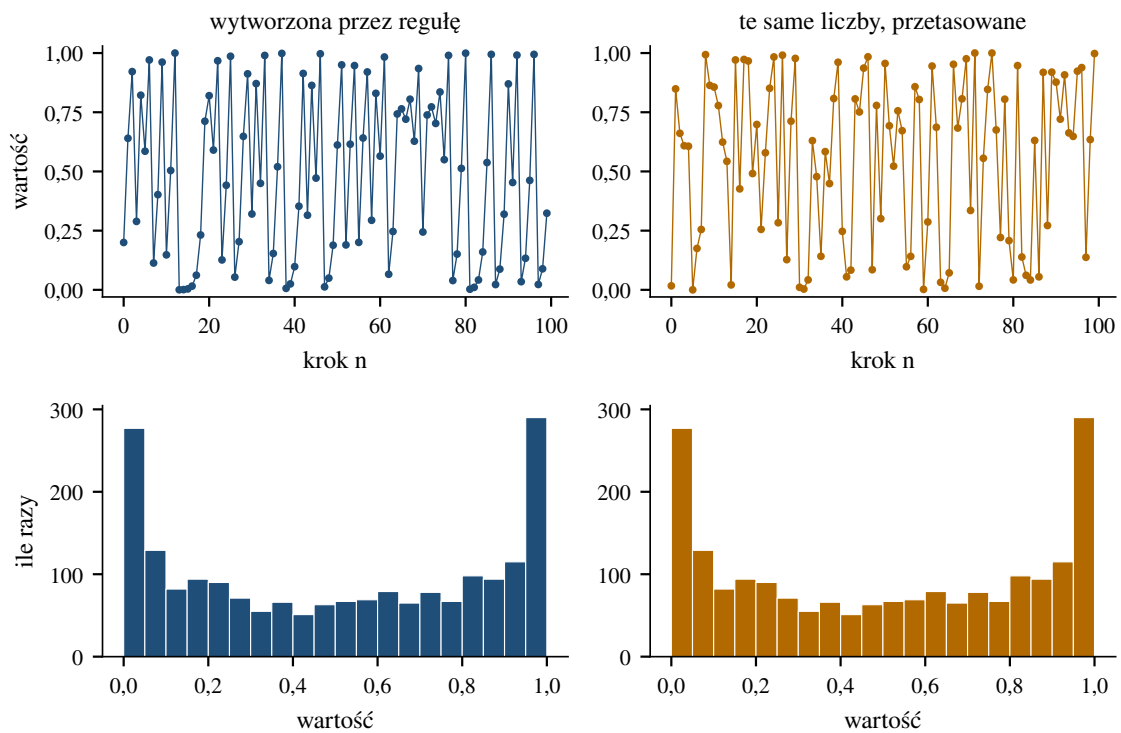
Zatrzymaj się i pomyśl 15.3

Gdzie leżą punkty odwzorowania powrotu szeregu reguły? A przetasowanego?

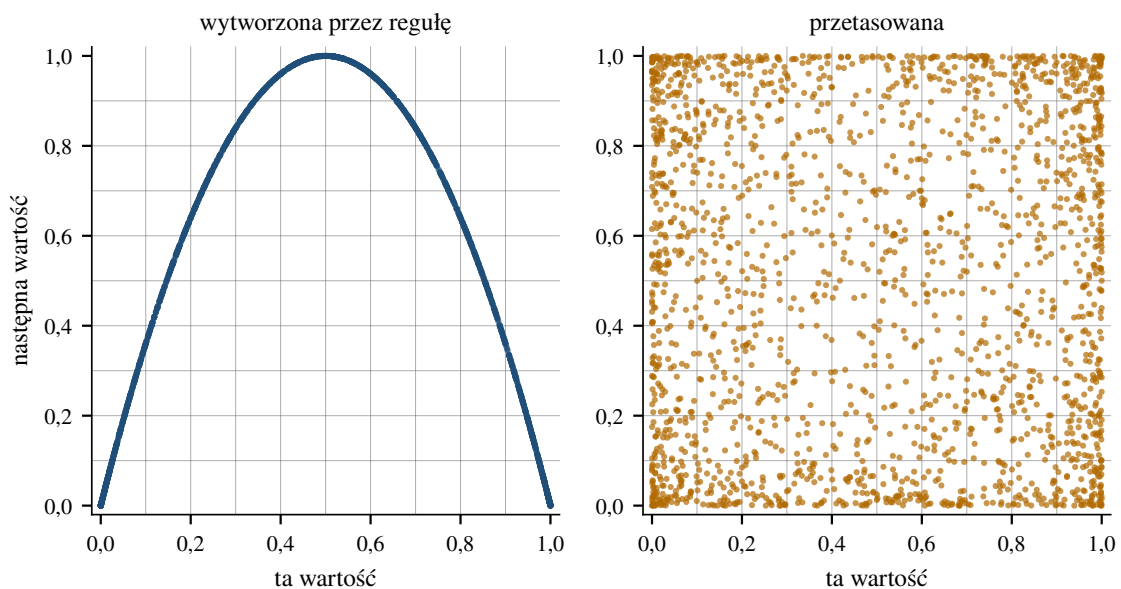
Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Punkty reguły leżą na jednej krzywej, garbie $y = 4x(1 - x)$, bo każda następna wartość *jest* regułą zastosowaną do poprzedniej. Punkty przetasowania rozchodzą się po całym kwadracie, bo po przetasowaniu następna wartość nie ma nic wspólnego z obecną.

Rysunek to ocena; liczba to pomiar. Podziel kwadrat na sto małych kratek i policz, w ile z nich trafia odwzorowanie powrotu:



Rysunek 15.1: Szereg reguły i jego przetasowanie. U góry: pierwsze sto kroków każdego, równie poszarpane. U dołu: ich histogramy, identyczne.



Rysunek 15.2: Każda wartość względem następnej. Szereg reguły rysuje własną regułę; przetasowany szereg wypełnia kwadrat. Cienkie linie dzielą kwadrat na sto małych kratak.

code/ch15/return_map.py

```

14 def return_pairs(xs: list[float]) -> list[tuple[float, float]]:
15     """Each value paired with the one after it."""
16     return list(zip(xs[:-1], xs[1:], strict=True))
17
18
19 def squares_touched(xs: list[float], k: int = 10) -> int:
20     """Cut the unit square into k by k little squares; count how many
21     of them the return map lands in."""
22
23     def box(v: float) -> int:
24         return min(k - 1, max(0, int(v * k)))
25
26     return len({(box(a), box(b)) for a, b in return_pairs(xs)})

```

Listing 15.2: Pary sąsiadów i kratki, w które trafiają

values	rule	shuffled	(of 100 squares)
100	24	64	
500	28	93	
2000	28	100	

Krzywa może przejść tylko przez ograniczoną liczbę krutek. 2000 wartości reguły trafia w 28, a dłuższy zapis nie zwiększyłby tej liczby, bo każdy nowy punkt ląduje na tej samej krzywej. Przetasowanie wciąż znajduje nowe kratki, aż zajmie wszystkie 100.

Laboratorium 15.1 – Krzywa czy chmura?

Napisz test odwzorowania powrotu samodzielnie: połącz każdą wartość w parę z następną, policz kratki, w które trafiają pary, i zwróć "curve", gdy zajętych jest mniej niż połowa krutek, a "cloud" w przeciwnym razie. Potem wypróbuj siatkę dwa na dwa na szeregu, który krąży między trzema wartościami, i zobacz, jak zbyt gruba siatka wprowadza w błąd.

Plik startowy: code/labs/ch15/l15_01_return.py

Test: code/labs/ch15/test_l15_01_return.py

Uruchom: cd code && uv run pytest -k l15_01_return

Gdzie to spotkasz

Robert Shaw, fizyk z Uniwersytetu Kalifornijskiego w Santa Cruz, mierzył czas między kroplami spadającymi z ciekącego kranu. Przy niektórych szybkościach przepływu krople spadały w nieregularnych odstępach, a gdy zaznaczył każdy odstęp względem następnego, punkty nie wypełniły chmury: ułożyły się wzdłuż cienkiego zakrzywionego kształtu. Kuchenny kran stał się jednym z najsłynniejszych pokazów chaosu odczytanego z jednej mierzonej wielkości.

Shaw mierzył jedną rzecz, odstęp między kroplami, choć w wodzie dzieje się znacznie więcej, niż opisze jedna liczba. To, że w ogóle zobaczył kształt, nie jest szczęściem. W

1980 roku Norman Packard, James Crutchfield, Doyne Farmer i sam Shaw pokazali, jak odbudować kształt ruchu układu z jednej mierzonej wielkości, używając jej opóźnionych kopii – odczytu teraz, chwilę temu i jeszcze chwilę wcześniej – w zastępstwie liczb, których nikt nie zmierzył. W 1981 roku Floris Takens udowodnił, że to działa. Odwzorowanie powrotu to przypadek najprostszy: jedna wielkość, jedno opóźnienie.

Czego tu nie dowodzimy

Twierdzenie Takensa słowami: dla gładkiego układu deterministycznego, którego stan ma d liczb, $2d + 1$ opóźnionych kopii niemal dowolnego pojedynczego pomiaru daje wierną kopię ruchu, wygiętą i rozciągniętą, ale o tym samym kształcie. Dowód, w jego pracy z 1981 roku *Detecting strange attractors in turbulence*, korzysta z topologii różniczkowej.

15.3 Prognoza z przeszłości

Drugi test zadaje pytanie, które obchodzi użytkownika danych: czy da się to przewidzieć? Bez reguły możesz przeszukać zapis w poszukiwaniu chwili najbardziej podobnej do obecnej i założyć, że to, co wtedy nastąpiło, nastąpi i teraz. To *metoda analogów*. W 1969 roku Edward Lorenz wypróbował ją na mapach pogody i nie znalazł dnia z przeszłości dość podobnego, by się przydał: przy tak wielu liczbach w stanie atmosfera nie wróciła w pobliże miejsc, w których już była. Dla szeregu pojedynczych liczb metoda działa:

code/ch15/analogues.py

```

15 def forecast(library: list[float], x: float, h: int) -> float:
16     """Predict h steps after x: find the value in the library most like
17     x, and return what came h steps after it."""
18     best = min(range(len(library) - h),
19               key=lambda j: abs(library[j] - x))
20     return library[best + h]
21
22
23 def mean_error(xs: list[float], h: int, split: int = SPLIT) -> float:
24     """The average size of the forecast's miss, h steps ahead: learn from
25     xs[:split], test on every value after it."""
26     library, test = xs[:split], xs[split:]
27     misses = [abs(forecast(library, test[t], h) - test[t + h])
28               for t in range(len(test) - h)]
29     return sum(misses) / len(misses)

```

Listing 15.3: Prognoza według najbliższego analogu z przeszłości

forecast znajduje w przeszłości wartość najbliższą x , jej *najbliższego sąsiada*, i zwraca to, co nastąpiło h kroków po nim. mean_error uczy się na pierwszym tysiącu wartości, prognozuje każdą wartość po nich i uśrednia wielkości chybień.

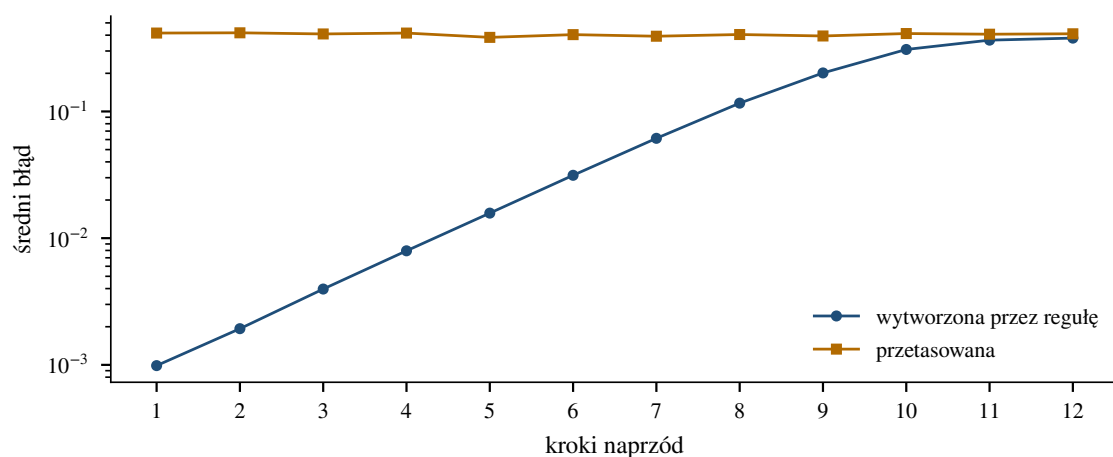
Zatrzymaj się i pomyśl 15.4

Dla szeregu reguły średnie chybiecie o krok naprzód wynosi około 0,00099, czyli jedną tysięczną. Strzał na ślepo – dowolna wartość wzięta z histogramu – chybia średnio o około 0,42. Rozdział 7 zmierzył, że przy $r = 4$ mały błąd mniej więcej podwaja się w każdym kroku. O ile kroków naprzód może sięgnąć prognoza, zanim jej chybiecia dojdą do połowy chybień strzału na ślepo?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Około dziewięciu: jedna tysięczna podwojona dziewięć razy to mniej więcej połowa, bo $2^9 = 512$. Pomiar się zgadza: chybiecia po raz pierwszy przekraczają połowę chybień przetasowania w kroku 9.

steps ahead	rule	shuffled
1	0.00099	0.42
2	0.0019	0.42
3	0.004	0.41
4	0.0079	0.42
5	0.016	0.38
6	0.031	0.4
7	0.061	0.39
8	0.12	0.4
9	0.2	0.39
10	0.31	0.41
11	0.37	0.41
12	0.38	0.41



Rysunek 15.3: Średnie chybiecie prognozy metodą analogów w zależności od liczby kroków naprzód, w skali logarytmicznej. Przetasowanie to strzał na ślepo przy każdym horyzoncie; szereg reguły prognozuje się dobrze i z każdym krokiem gorzej.

Dla przetasowanego szeregu prognoza chybia o około 0,42 przy każdym horyzoncie: jego przeszłość nie niesie żadnej informacji o następnej wartości. Dla szeregu reguły chybiecie o krok naprzód jest około 420 razy mniejsze i rośnie mniej więcej 2,0 razy na krok – to

rozciąganie, które rozdział 7 zmierzył od środka, widziane z zewnątrz – aż po kilkunastu krokach też staje się strzałem na ślepo.

Chaos i szum różnią się więc w obie strony. Szum jest równie nieprzewidywalny przy każdym horyzoncie. Chaos jest przewidywalny na kilka kroków naprzód, gdzie szum nie jest, i traci tę przewidywalność w stałym, mierzalnym tempie: to znacznie mocniejszy odcisk palca chaosu niż jakakolwiek nieregularność.

15.4 Prawdziwe dane to mieszanina

Nic nie jest mierzone dokładnie. Waga, termometr czy urzędnik przepisujący rejestr przesuwają każdą wartość trochę, losowo. Żeby zobaczyć, co to robi, przesun każdą wartość szeregu reguły o losową wielkość nie większą niż pewien rozmiar:

code/ch15/noisy.py

```
21 def with_noise(xs: list[float], size: float,
22               seed: int = NOISE_SEED) -> list[float]:
23     """Move each value by a random amount between -size and +size."""
24     rng = random.Random(seed)
25     return [x + rng.uniform(-size, size) for x in xs]
```

Listing 15.4: Szum pomiarowy: każda wartość przesunięta losowo

Zatrzymaj się i pomyśl 15.5

Dodaj do szeregu reguły szum o rozmiarze do 0,01: jedną setną, mniej niż grubość krzywej na wydrukowanym odwzorowaniu powrotu. Co się stanie ze średnim chybeniem o krok naprzód, które wynosiło 0,00099?

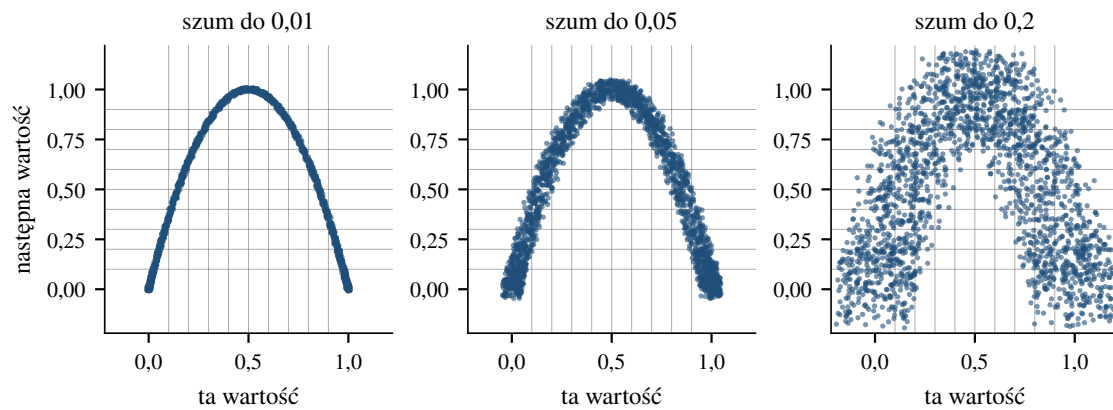
Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Wzrośnie do 0,018, wielokrotnie. Analog, który znajduje prognoza, sam jest przesunięty nawet o jedną setną, podobnie jak prognozowana wartość, i to zanim podwajanie reguły w ogóle się zacznie.

noise	squares	1-step error
0.00	28	0.00099
0.01	28	0.018
0.05	44	0.076
0.20	84	0.26

right answers out of 100 short records:

length	forecast test		squares test	
	clean	noisy	clean	noisy
10	79	74	56	38
20	98	89	95	73
40	100	95	100	92
100	100	99	100	99



Rysunek 15.4: Odwzorowania powrotu szeregu reguły oglądane przez szum pomiarowy trzech rozmiarów. Krzywa grubieje w pas, a przy największym rozmiarze pas prawie wypełnia kwadrat.

W górnej tabeli, dla całego zapisu, krzywa w miarę wzrostu szumu grubieje w pas: przy szumie do 0,2, jednej piątej całego zakresu, zajmuje 84 kratki. Mimo to przy 2000 wartościach prognoza wciąż jest lepsza od strzału na ślepo: 0,26 wobec 0,42.

Dolna tabela to część okrutna. Tnie zapis na krótkie kawałki i liczy, na sto kawałków, jak często każdy test uznaje kawałek reguły za mniej losowy od jego własnego przetasowania. Przy czterdziestu zaszumionych wartościach test prognozy ma rację 95 razy. Przy dziesięciu ma rację 74 razy, a test kratek tylko 38 – gorzej niż rzut monetą. Krótkie, zaszumione dane potrafią sprawić, że porządek wygląda jak przypadek, a przypadek jak porządek. A to najłatwiejszy chaos, jaki istnieje, z jedną liczbą na krok; układ z wieloma liczbami w stanie potrzebuje znacznie dłuższego zapisu.

Laboratorium 15.2 – Zapytaj kilka analogów

Gdy w zapisie jest szum, najbliższy analog może być bliski przypadkiem. Napisz prognozę, która znajduje kilka najbliższych wartości z przeszłości i uśrednia to, co po nich nastąpiło. Test sprawdza ją na małej bibliotece, którą przeliczysz ręcznie, a potem sprawdza, że uśrednienie pięciu analogów jest lepsze od zaufania jednemu na zaszumionym szeregu.

Plik startowy: `code/labs/ch15/l15_02_forecast.py`

Test: `code/labs/ch15/test_l15_02_forecast.py`

Uruchom: `cd code && uv run pytest -k l15_02_forecast`

15.5 Populacje, epidemie, serca

Praca Roberta Maya z 1976 roku o odwzorowaniu logistycznym była pisana dla ekologów. Jej lekcja brzmiała: populacja, której liczebność skacze nieregularnie z roku na rok, nie musi być popychana przez pogodę czy szczęście; może to robić jej własna reguła rozmnażania.

Zatrzymaj się i pomyśl 15.6

Wiele prawdziwych populacji owadów rzeczywiście rośnie i maleje nieregularnie z roku na rok. Czy to czyni je chaotycznymi?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Nieregularne liczby dają chaos, ale daje je też populacja w równowadze szarpana przez pogodę, choroby i przypadek. Nieregularność nie odróżni tych dwóch; mogą to zrobić tylko testy porządku.

Pułapka

„Nieregularne dane dowodzą chaosu.” Każdy szereg w tym rozdziale był nieregularny, a połowa z nich to przetasowania. Chaos to twierdzenie, że nieregularność tworzy reguła z niewieloma liczbami, i wymaga dowodów na tę regułę: struktury w odwzorowaniu powrotu, prognozy, która jest dobra na początku i pogarsza się równomiernie, oraz zapisu dość długiego i czystego, by jedno i drugie coś znaczyło.

Najmocniejsze dowody przyszły z laboratorium, gdzie regułę można zmieniać celowo. W 1997 roku Robert Costantino ze współpracownikami opisali w *Science* trojszyki, drobne chrząszcze żyjące w mące, hodowane w słojach, którym eksperymenciatorzy ręcznie ustalali tempo śmierci dorosłych i dołączania nowych dorosłych. Gdy te tempa się zmieniały, kolonie przechodziły od stałej liczebności przez cykle do nieregularnych wahań, tam gdzie przewidywał ich model. Dzika populacja liczona raz do roku przez kilka dziesięcioleci daje mniej więcej tak mało wartości jak dolna tabela powyżej.

Drugi klasyczny przykład to epidemie. *Model SIR* dzieli ludzi na podatnych, którzy mogą się zarazić, zakaźnych, którzy chorują, i ozdrowieńców. Nowe zakażenia każdego dnia są proporcjonalne do tego, ilu jest podatnych i zakaźnych, którzy mogą się spotkać, chory zaraża średnio przez 13 dni, a narodziny dostarczają nowych podatnych. Weź chorobę podobną do odry, w której jeden chory wśród ludzi bez odporności zaraża około 17 innych, i długość życia 50 lat. Potem dodaj rok szkolny: dzieci stykają się częściej w czasie nauki niż w wakacje, więc niech tempo kontaktów rośnie i maleje raz w roku o ułamek nazwany *sezonowością*.

code/ch15/epidemic.py

```

23 def contact(day: int, season: float) -> float:
24     """How much above or below average the contact rate is on this day:
25     1 + season on New Year's Day, 1 - season at midsummer, in straight
26     lines between."""
27     return 1.0 + season * (4.0 * abs(day / 365.0 - 0.5) - 1.0)
28
29
30 def one_year(s: float, i: float,
31             season: float) -> tuple[float, float, float]:
32     """Step through one year; return the new s and i, and the fraction
33     of people infected during the year."""

```

```

34 births, recover, dt = 1.0 / LIFE, 1.0 / DAYS_ILL, 1.0 / STEPS
35 beta0 = R0 * (recover + births)
36 cases = 0.0
37 for day in range(365):
38     beta = beta0 * contact(day, season)
39     for _ in range(STEPS):
40         new = beta * s * i
41         s = s + dt * (births - new - births * s)
42         i = i + dt * (new - (recover + births) * i)
43         cases = cases + dt * new
44 return s, i, cases

```

Listing 15.5: Epidemia z rokiem szkolnym: reguła SIR krokami po ćwierć doby

season	cases per thousand people, eight years running							
0.0	19	19	19	19	19	19	19	19
0.1	24	13	24	13	24	13	24	13
0.3	33	7	32	4	33	7	32	4
0.4	0	38	44	0	0	0	75	9

the same, starting one part in a billion more susceptible:

0.1	24	13	24	13	24	13	24	13
0.4	0	39	29	0	0	56	25	0

Bez roku szkolnego model się ustala: 19 zachorowań na tysiąc osób, co roku tyle samo. Przy sezonowości 0,1 lata duże i małe występują na przemian, jak odra w Anglii i Walii, zanim w 1968 roku wprowadzono szczepienia. Przy 0,3 wzór powtarza się co cztery lata, a przy 0,4 nie powtarza się wcale, a pchnięcie o jedną miliardową na starcie posyła lata w inną stronę. To droga do chaosu z rozdziału 8, przebyta przez epidemię.

Tu też potrzebna jest uczciwość. W chaotycznym ustawieniu epidemie modelu wypalają się tak dokładnie, że w najniższym punkcie zakaźna jest mniej niż jedna osoba na 10^{12} . Żadne miasto nie ma milionowej części człowieka: prawdziwa choroba by wygasła. Czy odra kiedykolwiek była chaotyczna, wciąż jest przedmiotem sporu. W 1990 roku George Sugihara i Robert May zastosowali prognozę metodą analogów do comiesięcznych zachorowań na odrę w Nowym Jorku i znaleźli podpis z tego rozdziału, dobrą prognozę, która równomiernie pogarsza się z horyzontem, podczas gdy ospa wietrzna w tym samym mieście wyglądała jak roczny cykl z szumem na wierzchu.

Z sercem jest jeszcze trudniej. W doświadczeniach laboratoryjnych opisanych w 1981 roku skupiska komórek serca pobudzone regularnymi impulsami elektrycznymi przechodziły przez podwojenie okresu do nieregularnego bicia. Zdrowe serce też nie bije jak metronom, ale niegdyś popularne twierdzenie, że jego zmienność to chaos, opiera się na testach z tego rozdziału zastosowanych do krótkich, zaszumionych zapisów i pozostaje sporne. Zmienność to nie to samo co chaos.

15.6 Rynki i czego wymaga twierdzenie o chaosie

Ceny to najpilniej obserwowane poszarpane linie na świecie i przez pewien czas wielu miało nadzieję, że to chaos jest wzorem ukrytym pod nimi.

Zatrzymaj się i pomyśl 15.7

Przypuśćmy, że ktoś udowodnił, że cenę akcji tworzy chaotyczna reguła z zaledwie kilkoma liczbami w stanie. Czy mógłbyś się wzbogacić na jej prognozowaniu?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Niewiele. Chaos da się prognozować tylko na kilka kroków naprzód, tylko tak dobrze, jak pozwalają pomiary, i wcale poza swoim horyzontem. A każdy inwestor, który znalazłby tę samą regułę, grałby według niej i zmieniał ceny, które ona opisuje.

Pułapka

„Chaos na rynkach oznacza, że da się je przewidzieć.” Chaotyczne ceny obiecywałyby krótki horyzont z twardą granicą za nim, a nie szklaną kulę. W rzeczywistości ekonomiści, którzy szukali, znaleźli stopy zwrotu trudne do przewidzenia oraz okresy spokoju i burzy skupione razem, ale dowody na chaotyczną regułę z niewieloma liczbami w stanie są słabe i sporne. Nieregularne, skokowe i trudne do przewidzenia to razem jeszcze nie chaotyczne.

Zbierz cały rozdział, a twierdzenie o chaosie w prawdziwych danych wymaga:

- **długiego zapisu**, długiego w porównaniu z liczbą wielkości, które mają znaczenie;
- **małego szumu** albo testów, które go uwzględniają;
- **uczciwego porównania**: szereg musi wygrać ze swoimi surogatami, także z takimi, które zachowują jego powolne wzrosty i spadki, bo szereg z pamięcią, jak dobową temperatura, daje się prognozować na dzień naprzód bez żadnego chaosu;
- **właściwego kształtu błędu prognozy**: małego o krok naprzód i rosnącego w stałym tempie;
- a najlepiej **mechanizmu**, najlepiej z pokrętle, które da się przekręcić, jak u chrząszczy.

Niewiele twierzeń o sercach, mózgach czy gospodarkach spełnia wszystkie pięć. To teoria chaosu wykonująca swoją pracę: mówi ci, do jakich wniosków poszarpana linia cię uprawnia, a do jakich nie.

Podsumowanie

- **Surogat** zrobiony przez tasowanie ma te same liczby, więc ten sam **histogram**. Tylko **kolejność** odróżni regułę od przypadku.
- **Odzworowanie powrotu** przedstawia każdą wartość względem następnej: reguła rysuje krzywą, szum wypełnia kwadrat. **Zanurzenie z opóźnieniem** (Takens, 1981) odbudowuje kształt układu z jednej mierzonej wielkości.

- **Metoda analogów** prognozuje według najbliższej wartości z przeszłości. Chaos prognozuje się dobrze o krok naprzód i gorzej w stałym tempie; szum jest strzałem na ślepo przy każdym horyzoncie.
- **Szum pomiarowy i krótkie zapisy** osłabiają oba testy; przy dziesięciu zaszumionych wartościach test kratek wypada gorzej niż moneta.
- Chaos przekonująco pokazano w populacjach laboratoryjnych; dla odry, serc i rynków dowody są mieszane, sporne albo słabe. **Nieregularne to nie chaotyczne.**
- Twierdzenie o chaosie wymaga długiego, czystego zapisu, uczciwego porównania z surogatami, właściwego kształtu błędu prognozy, a najlepiej mechanizmu.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

zrobić szereg chaotyczny i szereg losowy z dokładnie tych samych liczb i powiedzieć, dlaczego żaden histogram ich nie rozróżni	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
narysować odwzorowanie powrotu, policzyć kratki, w które trafia, i odczytać ten wynik	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
prognozować szereg z jego własnej przeszłości metodą analogów i zmierzyć, jak błąd rośnie z liczbą kroków naprzód	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, jak szum pomiarowy i krótki zapis osłabiają oba testy	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
ocenić twierdzenie o chaosie w populacjach, epidemiach, sercach czy na rynkach na podstawie przedstawionych dowodów	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Co tasowanie szeregu zachowuje, a co niszczy?
2. Odwzorowanie powrotu pewnego szeregu leży na jednej cienkiej krzywej. Co to mówi o tym, jak powstał szereg?
3. Jeden szereg zajmuje 30 ze stu kratek przy 500 wartościach i nadal 30 przy 5000. Inny zajmuje 80, a potem 100. Który zachowuje się jak szereg reguły?
4. Prognoza metodą analogów chybia o około 0,001 o krok naprzód, a jej chybień podwajają się w każdym kroku. Po ilu mniej więcej krokach chybień dojdzie do 0,25?

5. Dlaczego zapis z dziesięciu zaszumionych wartości jest słabym dowodem za chaosem albo przeciw niemu?
6. Co dzieje się z modelową epidemią, gdy rok szkolny staje się silniejszy, i co czyni jej chaotyczne ustawienie wątpliwym obrazem prawdziwej odry?
7. Ktoś pokazuje ci, że cena akcji jest nieregularna. Czy to dowód chaosu? Co byłoby dowodem?

Chaos wewnątrz komputera

Każdy obraz chaosu w tej książce narysował komputer i tak samo policzył każdą liczbę. Oto niewygodny fakt o maszynie, która je narysowała. Nie umie zapamiętać liczby 0,1. Zapamiętuje liczbę bardzo do niej zbliżoną, a za każdym razem, gdy dodaje albo mnoży dwie liczby, zaokrągla wynik do takiej, którą umie zapamiętać.

Przez kilka rozdziałów uczyłeś się, że chaos zamienia najmniejszy błąd w duży. Skoro więc komputer robi drobny błąd w każdym kroku, jak którykolwiek z tych obrazów może być poprawny? Ten rozdział mierzy odpowiedź, a jest ona lepsza, niż mógłbyś się obawiać, i gorsza, niż mógłbyś mieć nadzieję.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- powiedzieć, co zapamiętuje komputer, gdy wpisujesz liczbę, i dlaczego $0,1 + 0,2$ nie daje 0,3
- przewidzieć krok, w którym odwzorowanie podwajające ginie w komputerze
- oszacować, przez ile kroków chaotyczne obliczenie pozostaje poprawne, na podstawie liczby cyfr dwójkowych, które zachowuje
- powiedzieć, którym wynikom chaotycznej symulacji możesz ufać, i dlaczego
- opisać generator liczb losowych jako regułę deterministyczną i powiedzieć, dlaczego musi się powtarzać

16.1 Co zapamiętuje komputer

Zapisz jedną trzecią jako ułamek dziesiętny, a nigdy nie skończysz: $0,333 \dots$, trójki ciągną się bez końca. Zapis dziesiętny oddaje dokładnie tylko te ułamki, których mianownik składa się z dwójek i piątek, czynników dziesiątki. Komputer liczy zamiast tego dwójkami. W *systemie dwójkowym* miejsca po przecinku są warte połowę, ćwierć, jedną ósmą i tak dalej, więc $\frac{3}{4}$ zapisuje się jako 0,11: połowa plus ćwierć.

Zatrzymaj się i pomyśl 16.1

Które z tych liczb system dwójkowy zapisze dokładnie, stałą liczbą miejsc: $\frac{1}{2}$, $\frac{3}{8}$, $\frac{1}{10}$?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $\frac{1}{2}$ i $\frac{3}{8}$, ale nie $\frac{1}{10}$. System dwójkowy zapisuje dokładnie ułamki, których mianownik jest potęgą dwójki, a dziesięć ma czynnik pięć.

Dwójkowo jedna dziesiąta to 0,000110011..., a blok 0011 powtarza się bez końca, więc komputer musi ją gdzieś uciąć. Python, jak prawie każdy program, którego używasz, przechowuje liczbę jako *double*, liczbę podwójnej precyzji: 53 cyfry dwójkowe znaczące, czyli około 16 dziesiętnych, plus potęgę dwójki, która mówi, gdzie stoi przecinek. Dla jednej dziesiątej zapamiętuje najbliższą liczbę podwójnej precyzji, za dużą o mniej więcej $5,55 \times 10^{-18}$. Każda suma i każdy iloczyn są potem także *zaokrąglane*: dokładny wynik zastępuje najbliższa liczba podwójnej precyzji.

Zatrzymaj się i pomyśl 16.2

Zarówno 0,1, jak i 0,2 są zapamiętane odrobinę błędnie, a ich suma jest jeszcze raz zaokrąglana. Czy komputer zgodzi się, że $0,1 + 0,2$ to 0,3?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Suma drukuje się jako liczba o włos większa od 0,3, a porównanie odpowiada False.

code/ch16/floats.py

```

14 print(0.1 + 0.2)
15 print(0.1 + 0.2 == 0.3)
16 print(Decimal(0.1))           # the exact value stored for 0.1
17 print((0.1).as_integer_ratio()) # the same number as a fraction
18 print(2**55)                   # the bottom of that fraction
19 print((0.1 + 0.2) + 0.3, 0.1 + (0.2 + 0.3))
20 big = float(2**53)
21 print(big + 1.0 == big)        # 2**53 + 1 has no room to be stored

```

Listing 16.1: Co zapamiętuje komputer i co z tym robi

```

0.30000000000000004
False
0.1000000000000000055511151231257827021181583404541015625
(3602879701896397, 36028797018963968)
36028797018963968
0.6000000000000001 0.6
True

```

Trzeci wiersz to dokładna wartość zapamiętana dla 0,1; czwarty zapisuje ją jako ułamek, którego mianownik to 2^{55} . Każda liczba podwójnej precyzji to liczba całkowita podzielona przez potęgę dwójki. Szósty wiersz dodaje trzy liczby w dwóch kolejnościach i dostaje dwa wyniki: gdy każdy krok zaokrągla, grupowanie, o którym szkoła mówiła, że nie ma znaczenia, ma znaczenie. Ostatni pokazuje, że brakuje miejsca: powyżej 2^{53} liczba

podwójnej precyzji nie mieści już każdej liczby całkowitej.

Uwaga

Te reguły zaokrąglania ustalono w 1985 roku w standardzie IEEE 754, którego przestrzega prawie każdy komputer. Wymaga on, by $+$, $-$, $\times$, $\div$ i pierwiastek kwadratowy dawały dokładny wynik zaokrąglony do najbliższej liczby podwójnej precyzji. Dlatego twój komputer powinien się zgadzać z każdą liczbą wydrukowaną w tej książce: jej własne pomiary używają tylko tych działań.

16.2 Odwzorowanie, którego komputer nie umie policzyć

Najprostsza chaotyczna reguła, jaka istnieje, mieści się w jednym wierszu: podwój liczbę, a jeśli wynik wynosi 1 albo więcej, odejmij 1. Symbolicznie $x \rightarrow 2x \bmod 1$, czyli „ $2x$ bez części całkowitej”. To *odwzorowanie podwajające* w każdym kroku podwaja odstęp między dowolnymi dwoma bliskimi punktami startowymi. W systemie dwójkowym widać, jak działa: podwojenie przesuną każdą cyfrę o jedno miejsce w lewo, a odjęcie 1 wyrzuci cyfrę, która przeszła przez przecinek, więc 0,1011 staje się 0,011, potem 0,11, potem 0,1. Odwzorowanie odczytuje swój start po jednej cyfrze dwójkowej naraz.

Zatrzymaj się i pomyśl 16.3

Policz odwzorowanie podwajające ręcznie od $\frac{1}{10}$, na ułamkach, przez pięć kroków. Co jego orbita robi potem na zawsze?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $\frac{1}{5}, \frac{2}{5}, \frac{4}{5}, \frac{3}{5}, \frac{1}{5}$: krąży po tych samych czterech wartościach bez końca i nigdy nie dochodzi do 0.

Tymczasem komputerowe 0,1 to liczba całkowita podzielona przez 2^{55} : ma ostatnią cyfrę dwójkową, a każdy krok wypycha jedną cyfrę za przecinek.

Zatrzymaj się i pomyśl 16.4

Jaka będzie komputerowa orbita 0,1 po sześćdziesięciu krokach?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Dokładnie 0. Gdy ostatnia cyfra dwójkowa odejdzie, w kroku 55, nic nie zostaje, a 0 podwojone to 0 na zawsze.

code/ch16/doubling.py

```

22 x = 0.1                # what the computer stores for one tenth
23 t = Fraction(1, 10)    # one tenth itself
24 for n in range(61):
25     if n <= 5 or n in (40, 45, 50, 53, 54, 55, 60):
26         print(f"step {n:2d}: computer {x:.6f} truth {float(t):.6f}")
27     x, t = doubling(x), exact_doubling(t)

```

Listing 16.2: Odwzorowanie podwajające od jednej dziesiątej: w komputerze i na dokładnych ułamkach

step 0:	computer 0.100000	truth 0.100000
step 1:	computer 0.200000	truth 0.200000
step 2:	computer 0.400000	truth 0.400000
step 3:	computer 0.800000	truth 0.800000
step 4:	computer 0.600000	truth 0.600000
step 5:	computer 0.200000	truth 0.200000
step 40:	computer 0.600006	truth 0.600000
step 45:	computer 0.200195	truth 0.200000
step 50:	computer 0.406250	truth 0.400000
step 53:	computer 0.250000	truth 0.200000
step 54:	computer 0.500000	truth 0.400000
step 55:	computer 0.000000	truth 0.800000
step 60:	computer 0.000000	truth 0.600000

Przez czterdzieści kroków obie orbity zgadzają się do sześciu miejsc po przecinku; potem komputer odpyływa i ginie. Obliczona orbita może się mylić w całym swoim charakterze: pętla zamieniła się w upadek do zera.

Najdziwniejsze jest to, że arytmetyka komputera nie popełniła żadnego błędu. Podwojenie liczby podwójnej precyzji i odjęcie 1 nigdy nie wymagają zaokrąglania, a skrypt pomiarowy rozdziału sprawdza krok po kroku, że orbita komputera jest dokładnie prawdziwą orbitą liczby, którą zapamiętał. Błędny jest start. Liczba, którą da się zapamiętać, ma skończenie wiele cyfr dwójkowych, więc każdy taki start zostaje pochłonięty przez 0, a prawie każdy prawdziwy start ma nieskończenie wiele cyfr i nie zostaje pochłonięty nigdy. Z każdego z 996 startów $\frac{1}{997}$, $\frac{2}{997}$ i tak dalej orbita komputera zginęła po liczbie kroków od 45 do 62, najczęściej po 53.

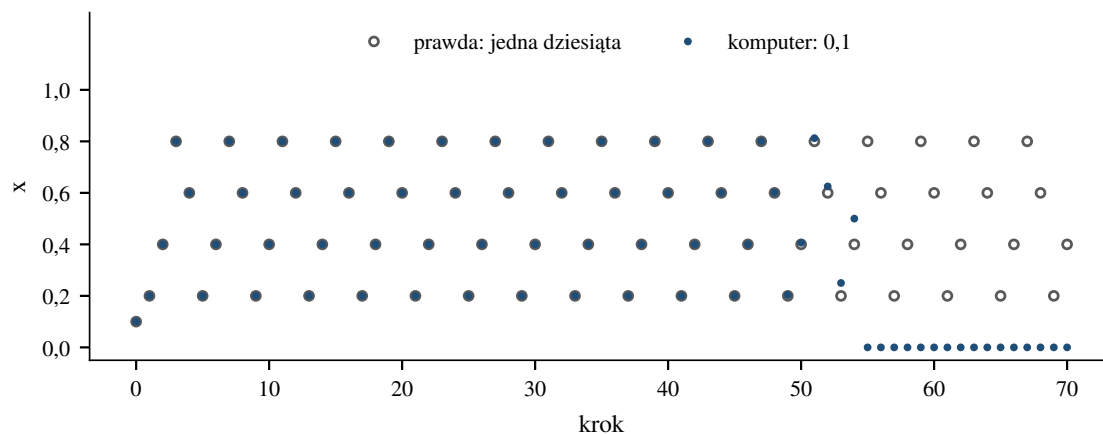
Laboratorium 16.1 – Kiedy ginie odwzorowanie podwajające?

Napisz odwzorowanie podwajające, funkcję, która liczy kroki, aż orbita będzie dokładnie 0, i funkcję, która przewiduje tę liczbę bez uruchamiania czegokolwiek, na podstawie `x.as_integer_ratio()`, które podaje zapamiętaną liczbę jako liczbę całkowitą przez potęgę dwójki. Test sprawdza, że obie zawsze się zgadzają.

Plik startowy: code/labs/ch16/l16_01_collapse.py

Test: code/labs/ch16/test_l16_01_collapse.py

Uruchom: `cd code && uv run pytest -k l16_01_collapse`



Rysunek 16.1: Odwzorowanie podwajające od jednej dziesiątej. Prawda (puste kółka) krąży w pętli bez końca; komputer (kropki) zgadza się z nią przez mniej więcej czterdzieści kroków, odpływa, a w kroku 55 spada do 0 na dobre.

16.3 Jedna reguła, dwa programy

Odwzorowanie podwajające nigdy nie zaokrągla. Odwzorowanie logistyczne z rozdziału 5 jest bardziej typowe: zaokrągla prawie każdy krok. Oto ono przy $r = 4$, zapisane na dwa sposoby.

code/ch16/two_formulas.py

```

14 def rule_a(x: float) -> float:
15     return 4.0 * x * (1.0 - x)
16
17
18 def rule_b(x: float) -> float:
19     return 4.0 * x - 4.0 * x * x      # the same, multiplied out

```

Listing 16.3: Odwzorowanie logistyczne przy $r = 4$ zapisane na dwa sposoby

Wymnóż nawias w pierwszym, a dostaniesz drugie: $4x(1 - x) = 4x - 4x^2$. Na papierze to jedna reguła.

Zatrzymaj się i pomyśl 16.5

Uruchom oba od 0,1 na tym samym komputerze i pozwól im działać przez siedemdziesiąt kroków. Czy oba programy wydrukują te same liczby?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Różnią się na ostatniej cyfrze dwójkowej od kroku 1, a w kroku 53 dzieli je już więcej niż 0,1.

Pierwszy wzór zaokrągla $1 - x$, a potem iloczyn; drugi zaokrągla $4x^2$, a potem różnicę. Każdy błąd to mniej więcej jedna część na 10^{16} , a odwzorowanie podwaja go w każdym kroku. Prawa kolumna poniżej to prawda: orbita dokładnie jednej dziesiątej, liczona ze

stu cyframi znaczącymi.

step	$4x(1-x)$	$4x-4xx$	truth
0	0.100000	0.100000	0.100000
10	0.147837	0.147837	0.147837
20	0.820014	0.820014	0.820014
30	0.320342	0.320342	0.320342
40	0.097905	0.097917	0.097917
45	0.473588	0.474255	0.474241
50	0.560037	0.538755	0.539206
55	0.879163	0.895376	0.904061
60	0.872399	0.806533	0.368811
70	0.039478	0.972805	0.554893

Który wzór ma rację? Oba, przez czterdzieści kroków. W kroku 53 pierwszy odszedł od prawdy, a w kroku 58 także drugi. Żaden wzór nie jest błędny; każdy jest poprawny przez mniej więcej pięćdziesiąt kroków, a potem podąża już za inną orbitą.

Pułapka

„Program deterministyczny zawsze daje ten sam wynik.” Uruchom ten sam program na tej samej maszynie, a da, co do ostatniej cyfry. Ale zmień komputer, kompilator, bibliotekę matematyczną albo kolejność dodawania długiej sumy, a zmienisz to, które zaokrąglenia zachodzą – dokładnie tak jak dwa wzory powyżej. Kompilatory mogą dla szybkości przedstawiać działania albo łączyć mnożenie i dodawanie w jeden krok z jednym zaokrągleniem. W większości programów różnica zostaje na ostatniej cyfrze. W chaotycznym dociera do pierwszej.

Pięćdziesiąt to nie przypadek i da się je przewidzieć.

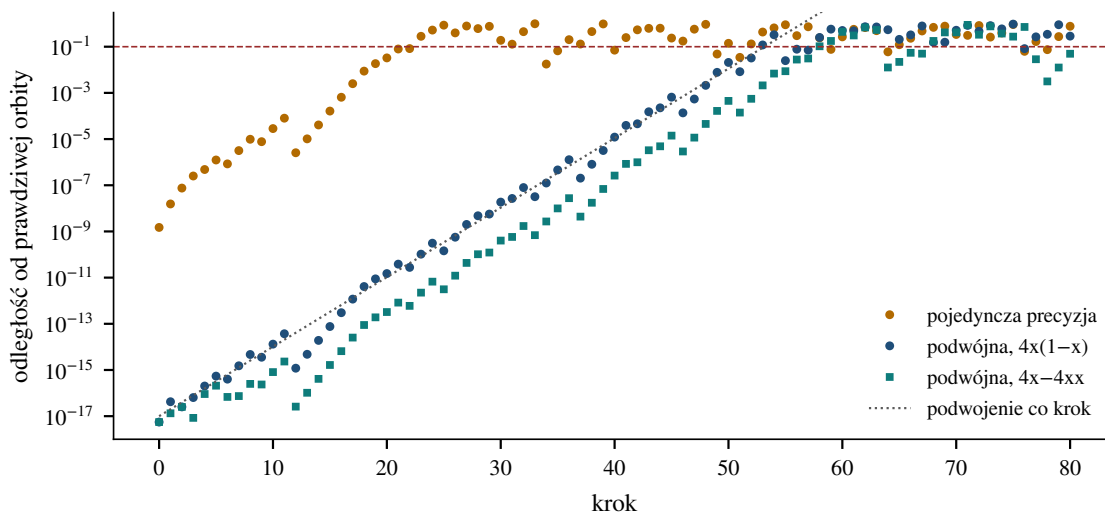
Zatrzymaj się i pomyśl 16.6

Przy $r = 4$ błąd mniej więcej podwaja się w każdym kroku, co zmierzył rozdział 6. Błąd zaokrąglenia liczby podwójnej precyzji to około 10^{-16} , a dziesięć podwojeń mnoży błąd mniej więcej przez tysiąc. Po mniej więcej ilu krokach błąd 10^{-16} urośnie do 0,1?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Po około pięćdziesięciu. Od 10^{-16} do 10^{-1} to piętnaście czynników dziesięć, a każde trzy czynniki dziesięć to mniej więcej dziesięć podwojeń. Horyzont z rozdziału 7, z błędem zaokrąglenia jako błędem startu, daje 50.

Jednym zdaniem: przy $r = 4$ odwzorowanie zużywa jedną cyfrę dwójkową dokładności na krok, więc 53 cyfry liczby podwójnej precyzji starczą na mniej więcej pięćdziesiąt kroków. *Pojedyncza precyzja*, której używają karty graficzne i wiele programów uczenia maszynowego, bo zajmuje połowę pamięci, zachowuje 24 cyfry dwójkowe, czyli około 7 dziesiętnych; ta sama arytmetyka przewiduje 21 kroków, a wytrzyma ona 23.



Rysunek 16.2: Odległość od prawdziwej orbity jednej dziesiątej, w skali logarytmicznej, dla odwzorowania logistycznego przy $r = 4$. Błąd każdego przebiegu podwaja się co krok, wzdłuż kropkowanej linii, aż wypełni cały przedział: po mniej więcej dwudziestu krokach w pojedynczej precyzji, pięćdziesięciu w podwójnej.

Laboratorium 16.2 – Pojedyncza przeciw podwójnej

Zaokrąglaj liczbę do pojedynczej precyzji, pakując ją do czterech bajtów i odczytując z powrotem (plik startowy pokazuje, jak). Napisz odwzorowanie logistyczne przy $r = 4$ z każdym działaniem zaokrąglanym w ten sposób, uruchom je obok wersji podwójnej precyzji i znajdź pierwszy krok, w którym różnią się o więcej niż 0,1.

Plik startowy: `code/labs/ch16/l16_02_single.py`

Test: `code/labs/ch16/test_l16_02_single.py`

Uruchom: `cd code && uv run pytest -k l16_02_single`

Gdzie to spotkasz

Uczenie maszynowe spotyka to codziennie. Sieć neuronową trenuje się, dodając miliony małych liczb na karcie graficznej, której równoległe wątki mogą za każdym razem kończyć w innej kolejności, więc suma zaokrągla się inaczej przy każdym przebiegu. Dwa przebiegi tego samego programu z tym samym ziarnem losowym mogą dać sieci, których wagi różnią się wszędzie – i które zwykle wypadają w testach mniej więcej tak samo. Dokumentacja PyTorch, szeroko używanej biblioteki, mówi wprost, że w pełni powtarzalne wyniki nie są gwarantowane między platformami i wersjami, i oferuje przełącznik wymuszający stałą kolejność działań, kosztem szybkości.

16.4 Dlaczego więcej cyfr cię nie uratuje

Oczywiste lekarstwo to przestać zaokrąglać. Moduł `fractions` Pythona przechowuje każdą liczbę jako dokładny ułamek. Jedna dziesiąta zostaje jedną dziesiątą, a $4x(1-x)$ daje z niej $\frac{9}{25}$, potem $\frac{576}{625}$, bez żadnego błędu.

Zatrzymaj się i pomyśl 16.7

Od $\frac{9}{25}$ do $\frac{576}{625}$ mianownik zmienił się z 25 na $625 = 25 \times 25$. Jeśli jest podnoszony do kwadratu w każdym kroku, co dzieje się z liczbą jego cyfr?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. W każdym kroku mniej więcej się podwaja: podniesienie liczby do kwadratu mniej więcej podwaja liczbę jej cyfr.

code/ch16/more_digits.py

```

35 x = Fraction(1, 10)
36 for n in range(13):
37     if n % 2 == 0:
38         print(f"step {n:2d}: {len(str(x.denominator)):5d} digits")
39     x = 4 * x * (1 - x)

```

Listing 16.4: Dokładne ułamki: mianownik podnoszony do kwadratu w każdym kroku

```

step 0:      2 digits
step 2:      3 digits
step 4:     12 digits
step 6:     45 digits
step 8:    179 digits
step 10:   716 digits
step 12: 2863 digits
16 digits kept: right for 54 steps
32 digits kept: right for 108 steps
64 digits kept: right for 217 steps
128 digits kept: right for 430 steps

```

Po dziesięciu krokach mianownik ma 716 cyfr. Po pięćdziesięciu miałby ich około $7,9 \times 10^{14}$, a samo jego zapisanie zajęłoby jakieś 327 terabajtów. Arytmetyka dokładna liczy każdy krok bez błędu i nie może skończyć. Ostatnie cztery wiersze próbują lekarstwa praktycznego: trzymać stałą, ale większą liczbę cyfr, modułem `decimal` Pythona, i porównać z przebiegiem liczonym z tysiącem cyfr. Szesnaście cyfr wystarcza na 54 kroki, jak w liczbie podwójnej precyzji; dwa razy więcej cyfr wystarcza na dwa razy dłużej. Każda cyfra dziesiętna kupuje mniej więcej 3,3 kroku, bo jest warta mniej więcej 3,3 cyfry dwójkowej.

Pułapka

„Więcej miejsc po przecinku czyni symulację dokładną.” Czyni ją poprawną dłużej, o stałą liczbę kroków na cyfrę, nigdy na zawsze. Żeby śledzić chaotyczną orbitę przez tysiąc kroków, trzeba mieć około trzystu cyfr dziesiętnych i znać start równie dokładnie, czego nie da żaden pomiar. To okrutna arytmetyka z rozdziału 7 od strony komputera: precyzja kupuje horyzont dłuższy o stałą, a nie proporcjonalnie.

16.5 Zła droga, dobry obraz

Każde długie chaotyczne obliczenie w tej książce podążało więc za złą orbitą. Czy narysowało zły obraz? Sposób, by się przekonać, opiera się na jednej obserwacji.

Zatrzymaj się i pomyśl 16.8

Odwzorowanie logistyczne uruchomione do przodu podwaja mały błąd w każdym kroku, przeciętnie. Co robi z małym błędem uruchomienie go wstecz, od każdej wartości do poprzedniej?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Przeciętnie go połowi. Co krok do przodu rozciąga, to krok wstecz ściska.

Zacznij więc od końca obliczonej orbity i idź wstecz. Do każdej wartości prowadzą w jednym kroku dwie liczby, jedna poniżej $\frac{1}{2}$ i jedna powyżej; bierz tę po tej samej stronie co wartość obliczona. Marsz wstecz zmniejsza błędy, więc trzyma się blisko obliczonej orbity aż do startu, a to, co znajduje, liczone z czterystoma cyframi, jest prawdziwą orbitą reguły.

code/ch16/shadow.py

```

20 def shadow(xs: list[float], digits: int = 400) -> list[Decimal]:
21     """A true orbit that stays near the computed orbit xs, found backwards.
22
23     Two numbers lead to y in one step: 1/2 - s/2 and 1/2 + s/2, where s is
24     the square root of 1 - y. Take the one on the same side of 1/2 as the
25     computed value, and step back again.
26     """
27     with localcontext() as ctx:
28         ctx.prec = digits
29         y = Decimal(xs[-1])
30         ys = [y]
31         for x in reversed(xs[:-1]):
32             s = (1 - y).sqrt()
33             y = HALF - s / 2 if x < 0.5 else HALF + s / 2
34             ys.append(y)
35     return ys[::-1]
```

Listing 16.5: Marsz wstecz w poszukiwaniu prawdziwej orbity, za którą podąża obliczona

```

steps computed with 4x(1-x):      1000
shadow's start minus 1/10:        -1.1e-17
farthest its exact orbit strays:  8.6e-15
fraction of a million steps below 0.1:
4x(1-x) 0.205  4x-4xx 0.205  exact 0.205
```

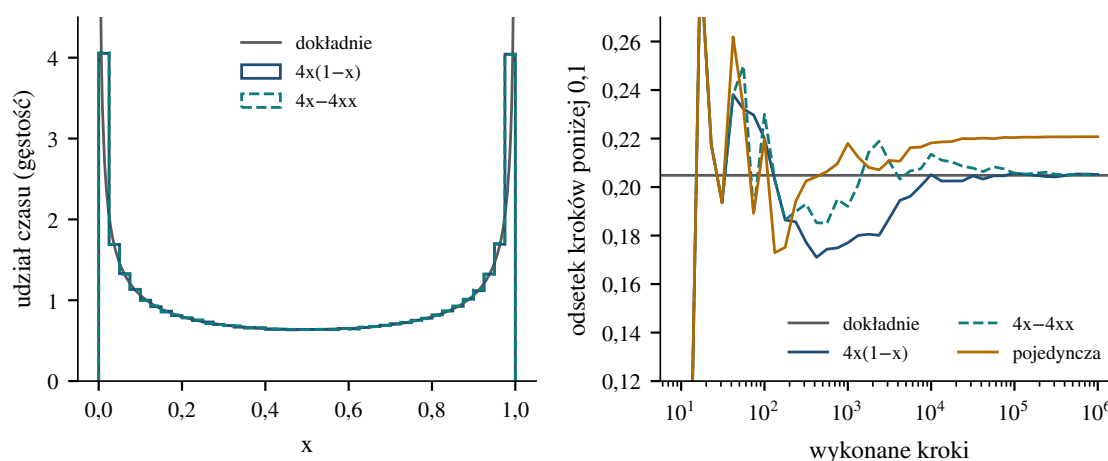
Obliczona orbita 0,1 odeszła od prawdy po 53 krokach. Ale start różniący się od jednej dziesiątej o mniej więcej $1,1 \times 10^{-17}$ ma dokładną orbitę, która trzyma się w odległości

$8,6 \times 10^{-15}$ od każdej z 1000 obliczonych wartości. Komputer nie podążał za orbitą, o którą go poproszono. Podążał, wiernie, za orbitą startu tak bliskiego, że żaden pomiar nie odróżniłby ich od siebie. Ta prawdziwa orbita to jego *cień*.

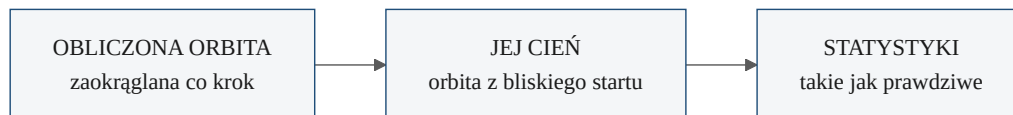
Czego tu nie dowodzimy

Lemat o cieniowaniu (nazywany też lematem o śledzeniu), udowodniony przez Dmitrija Anosowa i Rufusa Bowena, dotyczy układów, które w każdym punkcie rozciągają w niektórych kierunkach i ściskają w innych, w tempie, które nigdy nie słabnie (układów *jednostajnie hiperbolicznych*, takich jak odwzorowanie podwajające). Mówi, że jakkolwiek blisko chcesz, by obliczona orbita miała swój cień, wystarczająco mały błąd na krok to gwarantuje, na zawsze. Odwzorowanie logistyczne przy $r = 4$ nie całkiem jest takie: w pobliżu $x = \frac{1}{2}$ jego krzywa jest płaska i tam marsz wstecz rozciąga. Dla niego cieniowanie potwierdzono starannymi obliczeniami na długich, ale skończonych odcinkach, zwłaszcza Hammel, Yorke i Grebogi w 1987 roku. Ta książka nie dowodzi ani jednego, ani drugiego; dowody są w podręcznikach układów dynamicznych dla zaawansowanych.

Teraz zapłata. Cień jest prawdziwą orbitą, więc spędza czas tak, jak prawdziwe orbity. Mierzą to ostatnie dwa wiersze wydruku: przez milion kroków pierwszy wzór spędza ułamek 0,205 czasu poniżej 0,1, drugi 0,205, a dokładna odpowiedź, ze wzoru, którego ta książka nie wyprowadza, to 0,205. Dwa programy, których drogi rozeszły się po pięćdziesięciu krokach, zgadzają się co do tego, gdzie orbita mieszka. Dlatego ułamkom czasu na skrzydłach z rozdziału 9 i statystykom klimatu z rozdziału 14 można ufać, choć żadnej pojedynczej obliczonej drodze nie można.



Rysunek 16.3: Z lewej: gdzie dwa przebiegi podwójnej precyzji po milion kroków spędzają czas, wobec dokładnej odpowiedzi. Z prawej: udział kroków poniżej 0,1 w miarę wydłużania przebiegów. Przebiegi podwójnej precyzji ustalają się na wartości dokładnej; przebieg pojedynczej precyzji, uwięziony w pętli, na błędnej.



Rysunek 16.4: Obliczona orbita trzyma się blisko prawdziwej orbity z pobliskiego startu, a statystyki tamtej orbity to te, które pokazuje obraz.

16.6 Maszyny, które muszą się powtarzać

Najprostsza różnica ze wszystkich: między 0 a 1 jest 1 065 353 217 liczb pojedynczej precyzji i 4 607 182 418 800 017 409 liczb podwójnej precyzji: ogromnie dużo, ale skończenie wiele. Obliczona orbita musi więc prędzej czy później trafić na wartość, którą już spotkała, a ponieważ reguła za każdym razem daje tę samą następną wartość, od tej chwili krąży po tej samej pętli bez końca.

W pojedynczej precyzji pętle są dość krótkie, by je znaleźć. Od 0,1 odwzorowanie logistyczne przy $r = 4$ błądzi przez 1744 kroki, a potem krąży bez końca po pętli 4344 wartości; 199 różnych startów kończy w zaledwie 5 różnych pętlach. Pętla psuje też statystyki: przebieg pojedynczej precyzji spędza ułamek 0,221 czasu poniżej 0,1, a nie 0,205, co pokazuje rysunek 16.3. Liczby podwójnej precyzji też mają pętle, ale o wiele za długie, by do nich dojść, i to jest praktyczny powód, by chaos liczyć w podwójnej precyzji.

Zatrzymaj się i pomyśl 16.9

Odwzorowanie logistyczne przy $r = 4$ nigdy się nie powtarza, pokrywa cały przedział od 0 do 1 i na pierwszy rzut oka wygląda losowo. Czy byłoby dobrym generatorem liczb losowych?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie, z trzech powodów, które ten rozdział zmierzył. Jego wartości gromadzą się przy 0 i 1: mniej więcej jedną piątą czasu spędza poniżej 0,1, a nie jedną dziesiątą. Każda wartość jest wyznaczona przez poprzednią. A w komputerze wpada w pętlę.

Pułapka

„Chaotyczne odwzorowanie to dobry generator liczb losowych.” Chaos czyni przyszłość nieprzewidywalną dla kogoś, kto nie zna dokładnie startu, a to część tego, czego potrzebuje losowość, nie całość. Dobry generator musi też rozkładać wartości równomiernie, nie pokazywać wzoru między kolejnymi wartościami i powtarzać się dopiero po pętli, której nikt nigdy nie zdoła obejść. Te własności projektuje się za pomocą teorii liczb i sprawdza testami statystycznymi; chaos nie daje ich za darmo.

Generator liczb losowych to deterministyczna reguła na skończonym zbiorze wartości, zbudowana tak, by miała jedną ogromną pętlę. Najprostszy rodzaj, *liniowy generator*

kongruencyjny, to jeden wiersz: pomnóż przez a , dodaj c i zachowaj resztę z dzielenia przez m . Dla $a = 5$, $c = 3$ i $m = 16$, od 0 dostajesz 3, potem $5 \times 3 + 3 = 18$, czyli resztę 2, potem 13, 4, 7, 6, 1, 8, 11, 10, 5, 12, 15, 14, 9 i znowu 0: wszystkie szesnaście możliwych wartości przed pierwszym powtórzeniem. Odwzorowanie podwajające jest generatorem tego rodzaju z $a = 2$ i $c = 0$, a na k cyfrach dwójkowych ginie w k krokach – to upadek z podrozdziału 16.2 w nowym przebraniu. Własny generator Pythona, Mersenne Twister Makoto Matsumoto i Takujiego Nishimury (1998), ma pętlę długości $2^{19937} - 1$. Ponieważ jest deterministyczny, ta sama wartość początkowa, *ziarno*, daje za każdym razem te same „losowe” liczby i tak właśnie losowy eksperyment staje się powtarzalny.

Laboratorium 16.3 – Każda orbita kończy się pętlą

Napisz liniowy generator kongruencyjny i funkcję, która dla dowolnej reguły i startu znajduje, po ilu krokach orbita wchodzi w pętlę i jak długa jest ta pętla. Wypróbuj ją na generatorze powyżej, a potem sam znajdź pętlę odwzorowania logistycznego w pojedynczej precyzji.

Plik startowy: code/labs/ch16/l16_03_loops.py

Test: code/labs/ch16/test_l16_03_loops.py

Uruchom: `cd code && uv run pytest -k l16_03_loops`

Praktyczna lekcja mieści się w jednym zdaniu: nigdy nie ufaj pojedynczemu długiemu przebiegowi chaotycznego obliczenia. Uruchom je jeszcze raz z przedstawionym wzorem, z większą i mniejszą precyzją, z mniejszym krokiem, i wierz temu, co przetrwa. Droga dalej niż pięćdziesiąt kroków nie przetrwa. Statystyki przetrwają.

Podsumowanie

- Liczba **podwójnej precyzji** zachowuje 53 cyfry dwójkowe znaczące. Większość ułamków dziesiętnych, w tym 0,1, zapisuje się niedokładnie, a każda suma i iloczyn są **zaokrąglane**.
- **Odwzorowanie podwajające** odczytuje swój start po jednej cyfrze dwójkowej, więc w komputerze każda orbita ginie w 0; prawdziwa orbita jednej dziesiątej krąży w pętli bez końca.
- Dwa programy dla jednej reguły zaokrąglają inaczej i się rozchodzą. Przy $r = 4$ odwzorowanie logistyczne zużywa **jedną cyfrę dwójkową na krok**: mniej więcej pięćdziesiąt kroków w podwójnej precyzji, dwadzieścia w pojedynczej.
- **Arytmetyka dokładna** podwaja liczbę cyfr w każdym kroku i nie może skończyć; **więcej cyfr** kupuje stałą liczbę kroków na cyfrę.
- Obliczona orbita ma **cień**, prawdziwą orbitę z pobliskiego startu, więc jej **statystyki** są poprawne, choć droga jest błędna.
- Skończona maszyna sprawia, że każda orbita kończy się **pętlą**; generator liczb losowych to reguła zbudowana tak, by ta pętla była ogromna.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

powiedzieć, co zapamiętuje komputer, gdy wpisujesz liczbę, i dlaczego $0,1 + 0,2$ nie daje $0,3$	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
przewidzieć krok, w którym odwzorowanie podwajające ginie w komputerze	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
oszacować, przez ile kroków chaotyczne obliczenie pozostaje poprawne, na podstawie liczby cyfr dwójkowych, które zachowuje	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, którym wynikiem chaotycznej symulacji możesz ufać, i dlaczego	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
opisać generator liczb losowych jako regułę deterministyczną i powiedzieć, dlaczego musi się powtarzać	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dlaczego w komputerze $0,1 + 0,2$ nie równa się $0,3$?
2. Komputer zapamiętuje $0,3$ jako liczbę całkowitą przez 2^{54} . Kiedy jej orbita w odwzorowaniu podwajającym dochodzi do 0 i co robi prawdziwa orbita $\frac{3}{10}$?
3. Pojedyncza precyzja zachowuje 24 cyfry dwójkowe. Przez mniej więcej ile kroków odwzorowania logistycznego przy $r = 4$ jest dobra?
4. Liczony dokładnie od $\frac{1}{10}$ mianownik ma po dziesięciu krokach 716 cyfr. Mniej więcej ile po dwudziestu?
5. Po milionie kroków chaotycznej symulacji czemu możesz ufać: ostatniej wartości czy ułamkowi czasu spędzonego poniżej $0,1$?
6. Dlaczego każda orbita liczona na maszynie musi się w końcu powtórzyć?

Część V

Życie z chaosem

Okiełznać chaos: sterowanie i synchronizacja

Postaw miotłę pionowo na dłoni. Zostawiona sama sobie przewraca się: pion to położenie, którego nie potrafi utrzymać. A jednak możesz ją tak trzymać, jak długo starczy ci cierpliwości, ruchami dłoni tak drobnymi, że ktoś, kto patrzy, mógłby ich nie zauważyć. Nie pokonujesz miotły siłą. Patrzysz, w którą stronę zaczyna się przechylać, i robisz małą poprawkę, raz za razem.

Część II tej książki pokazała, że układ chaotyczny zamienia maleńką różnicę w dużą, i zmierzyła, jak szybko. Ten rozdział odwraca to zdanie. Skoro maleńkie zaburzenie może bardzo zmienić przyszłość układu chaotycznego, to czy maleńkie, dobrze dobrane zaburzenie może ją zmienić tak, jak chcesz? A skoro dwa układy chaotyczne zostawione same sobie zawsze się rozchodzą, to czy da się sprawić, żeby się zgadzały?

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- znaleźć punkt stały odwzorowania logistycznego, którego reguła nie potrafi utrzymać, i policzyć, jak często chaotyczna orbita zbliża się do niego
- obliczyć małą zmianę parametru, która znosi chybiecie, jakie zrobiłby jeden krok
- uruchomić sterownik małych pchnięć na chaotycznym odwzorowaniu i zmierzyć, jak długo czeka i jak duże są jego pchnięcia
- napędzać kopię układu Lorenza jedną liczbą z oryginału i zmierzyć, jak z upływem czasu zmienia się odległość między nimi
- powiedzieć, gdzie wykazano, że sterowanie chaosem i synchronizacja działają, i czym one nie są

17.1 Punkt, którego orbita nie utrzyma

Weź odwzorowanie logistyczne z rozdziału 5, regułę, która zamienia x na $r x(1 - x)$, przy $r = 3,9$, gdzie orbita nigdy się nie ustala i nigdy się nie powtarza. Reguła wciąż ma punkt stały: wartość $1 - 1/r$, którą reguła zostawia dokładnie tam, gdzie jest. Przy $r = 3,9$ to mniej więcej 0,744. Rozdział 5 powiedział też, dlaczego orbita tam nie zostaje. W pobliżu punktu stałego jeden krok mnoży małe chybiecie przez nachylenie reguły w tym punkcie – czynnik rozciągania z rozdziału 3 – a tutaj nachylenie wynosi $-1,9$. Chybiecie o jedną tysięczną staje się prawie dwiema tysięcznymi po drugiej stronie, potem prawie czterema, potem prawie siedmioma. Punkt stały to miotła postawiona na sztorc.

code/ch17/hidden.py

```

13 R = 3.9
14 rule = logistic(R)
15 POINT = 1.0 - 1.0 / R          # where the rule leaves x unchanged
16 STRETCH = logistic_slope(R)(POINT)  # one step multiplies a miss by this

```

Listing 17.1: Punkt stały i to, co jeden krok robi tam z chybeniem

Wiersz zaczynający się od STRETCH prosi bibliotekę książki o nachylenie reguły w punkcie stałym: o to, ile razy jeden krok mnoży tam małe chybenie.

Zatrzymaj się i pomyśl 17.1

Orbita przy $r = 3,9$ jest chaotyczna, a punkt stały odpycha wszystko, co się do niego zbliży. Czy orbita, gdy już odejdzie, wraca kiedykolwiek na odległość mniejszą niż jedna setna od 0,744: nigdy, kilka razy, czy raz za razem?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Raz za razem. W ciągu 10 000 kroków od $x = 0,2$ jest w odległości mniejszej niż jedna setna od punktu stałego w 112 krokach, po raz pierwszy w kroku 25.

Chaotyczna orbita wędruje po całym swoim zakresie i nie pomija żadnej jego części: rozdział 9 pokazał, że to, jak często układ chaotyczny odwiedza każdy obszar, jest jedną z rzeczy, które da się o nim przewidzieć. Funkcja `visits` z listingu liczy kroki, które lądują blisko:

```

fixed point      0.7436
one step later   0.7436
slope there      -1.90
steps within 0.01 of it, out of 10000: 112
the first ten: [25, 175, 176, 177, 285, 546, 547, 548, 549, 703]

```

Zobacz, jak przychodzą te wizyty: seriami sąsiednich kroków. Gdy orbita dotrze blisko, nachylenie $-1,9$ wyrzuca ją tylko stopniowo, prawie podwajając chybenie i zmieniając jego stronę w każdym kroku, więc orbita zostaje kilka kroków, zanim zostanie odrzucona.

Punkt stały to tylko najprostsza z rzeczy, których reguła nie potrafi utrzymać. Są cykle dwóch wartości, trzech i wielu innych długości, każdy z nich niestabilny, a chaotyczna orbita przechodzi kolejno w pobliżu każdego z nich. Atraktor chaotyczny nie jest bezkształtną mgłą. To płatanina niestabilnych cykli, a orbita bez końca przeskakuje z okolicy jednego do okolicy następnego.

17.2 Pchnięcie we właściwej chwili

W 1990 roku Edward Ott, Celso Grebogi i James Yorke opublikowali artykuł zatytułowany „Controlling chaos” („Sterowanie chaosem”), z pomysłem równie przyziemnym jak miotła.

Nie walcz z chaosem. Poczekaj, aż orbita przejdzie blisko niestabilnej orbity, której potrzebujesz, a potem dawaj układowi małe pchnięcia, każde tylko tak duże, jak trzeba, by znieść chybiecie, które zrobiłby następny krok. Pchnięcia trafiały w *parametr*: ustawienie układu, które wolno ci trochę zmieniać. W odwzorowaniu logistycznym tym ustawieniem jest r .

Zatrzymaj się i pomyśl 17.2

Załóżmy, że możesz zmieniać r w każdym kroku, ale nigdy o więcej niż jeden procent: na dowolną wartość między 3,861 a 3,939. Pozostawione przy 3,9 odwzorowanie jest chaotyczne, a jeden procent to prawie żadna zmiana. Czy tak małe pchnięcia mogą utrzymać orbitę w 0,744?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tak. W przebiegu z następnego podrozdziału sterownik, którego pchnięcia nigdy nie przekraczają jednego procenta, zostaje włączony w kroku 100, łapie orbitę w kroku 175 i trzyma ją w punkcie stałym tak długo, jak trwa przebieg. Największe pchnięcie, jakie kiedykolwiek robi, wynosi 0,70 procent wartości r .

Pułapka

„Chaos nie da się sterować.” Chaos zabrania przewidywania na daleką metę: rozdział 7 zmierzył, jak mały błąd rośnie, aż prognoza traci wartość. Sterownik nigdy nie patrzy daleko naprzód. Patrzy o jeden krok naprzód, a na jeden krok reguła chaotyczna jest tak przewidywalna jak filiżanka herbaty: chybiecie mnoży się przez $-1,9$, a nie przez milion. Chaos odbiera ci daleki widok, ale zostawia bliski, a do sterowania potrzeba tylko bliskiego. Chaos wykonuje nawet połowę pracy, bo to on w ogóle doprowadza orbitę w pobliże celu.

Jak duże musi być każde pchnięcie? Rozstrzygają o tym dwa fakty. Pierwszy już znasz: jeden krok mnoży chybiecie przez $-1,9$. Drugi to działanie pokręta. Reguła to r razy $x(1 - x)$, więc podniesienie r o małą wartość c na jeden krok podnosi następną wartość o c razy $x(1 - x)$. W pobliżu punktu stałego $x(1 - x)$ wynosi mniej więcej 0,19; nazwij to *dźwignią*.

Zatrzymaj się i pomyśl 17.3

Orbita jest o 0,001 powyżej punktu stałego. Zostawiona sama sobie, w następnym kroku znajdzie się $1,9 \times 0,001 = 0,0019$ poniżej. Podniesienie r o c podnosi następną wartość mniej więcej o 0,19 razy c . Jakie c znosi chybiecie?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. $c = 0,0019/0,19$, czyli mniej więcej 0,01: podnieś r z 3,9 do 3,91 na ten jeden krok, o mniej więcej ćwierć procenta.

Ogólnie zmiana to chybiecie razy nachylenie, podzielone przez dźwignię, ze zmienionym znakiem: mniej więcej 9,97 razy chybiecie. To cały sterownik.

code/ch17/control.py

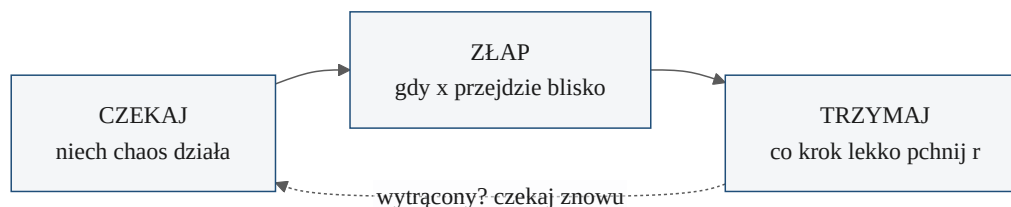
```

14 R0 = 3.9
15 TARGET = 1.0 - 1.0 / R0           # the fixed point, unstable at R0
16 STRETCH = logistic_slope(R0)(TARGET) # one step multiplies a miss by this
17 LEVER = TARGET * (1.0 - TARGET)     # next x moves this much per unit of r
18
19
20 def nudge(x: float, largest: float) -> float:
21     """The change to r, for this step only, that cancels the coming miss.
22
23     Zero if the change needed is bigger than `largest`: then wait.
24     """
25     change = -STRETCH * (x - TARGET) / LEVER
26     return change if abs(change) <= largest else 0.0

```

Listing 17.2: Sterownik: znieś nadchodzące chybiecie albo czekaj

`nudge` to reguła, a jej ostatni wiersz to cierpliwość. Jeśli potrzebna zmiana jest większa niż `largest`, funkcja zwraca zero i odwzorowanie działa dalej nietknięte. Gdy `largest` wynosi jeden procent r , sterownik może działać tylko wtedy, gdy orbita jest w odległości mniejszej niż 0,0039 od punktu stałego, i musi czekać, aż chaos ją tam doprowadzi.



Rysunek 17.1: Trzy nastroje sterownika. Czekaj, dopóki orbita jest poza zasięgiem, łapie ją, gdy chaos przyprowadzi ją blisko, i trzyma ją pchnięciami; wytrącony, czeka znowu.

17.3 Drobne, a nie siła

Teraz uruchom sterownik. Poniższy listing zostawia odwzorowanie w spokoju przez 100 kroków od $x = 0,2$, potem włącza sterownik z pchnięciami najwyżej o jeden procent i od czasu do czasu wypisuje stan.

Zatrzymaj się i pomyśl 17.4

Gdy orbita zostanie złapana, jak duże muszą być pchnięcia, żeby ją tam utrzymać: tak duże jak pierwsze, większe, bo chaos wciąż się odpycha, czy mniejsze?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Mniejsze, a potem prawie żadne. W ciągu 3 kroków od złapania każde pchnięcie jest mniejsze niż miliardowa część r , a wszystkie pchnięcia przebiegu razem dają 0,028.

Pułapka

„Sterować układem to go przemóc.” Sterownik nie dostarcza żadnego ruchu. Punkt stały i wizyty orbity w jego pobliżu należą do odwzorowania; sterownik tylko znosi małe chybiaenia, które widzi z wyprzedzeniem, a złapana orbita nie ma już prawie czego znosić. Wybierasz, które z własnych zachowań układu zachować, a nie narzucasz mu nowego. Dlatego Ott, Grebogi i Yorke mogli trzymać układ chaotyczny na dowolnym z jego wielu niestabilnych cykli i przełączać go między nimi za pomocą tych samych małych pchnięć.

code/ch17/control.py

```

37 largest = 0.01 * R0          # never more than one per cent of r
38 x, biggest = 0.2, 0.0
39 print("step x               change to r")
40 for n in range(301):
41     change = nudge(x, largest) if n >= 100 else 0.0
42     if n % 50 == 0 or abs(change) > 1e-9:
43         print(f"{n:4d} {x:.10f} {change:+.2e}")
44         biggest = max(biggest, abs(change))
45         x = step(x, change)
46 print(f"largest change: {100 * biggest / R0:.2f} per cent of r")

```

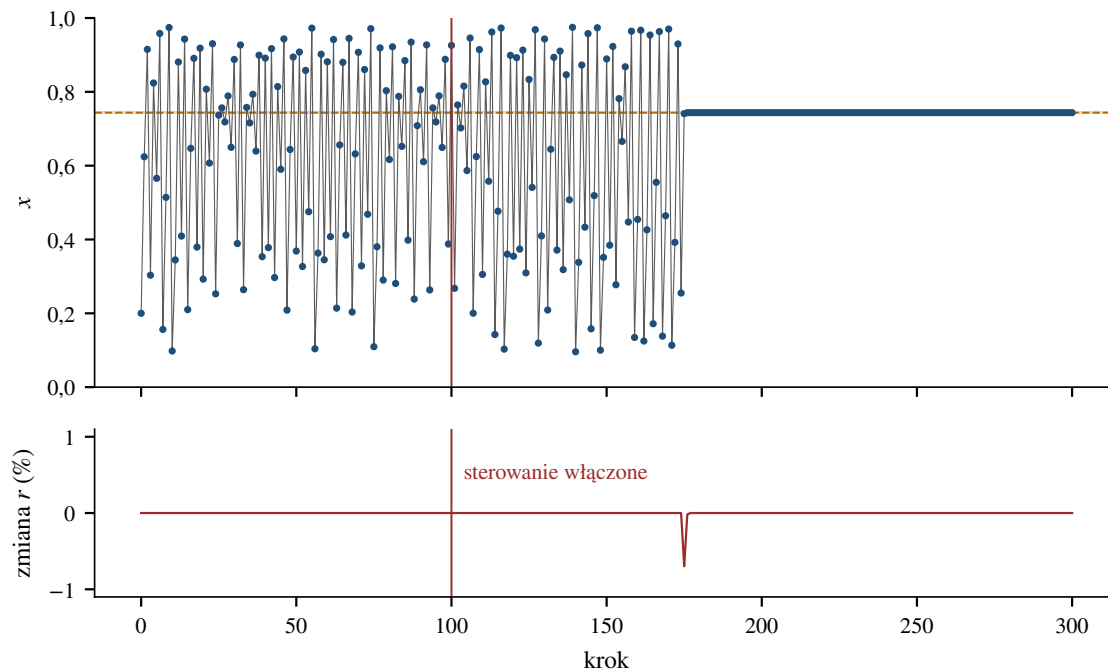
Listing 17.3: Chaos przez sto kroków, potem sterownik

```

step  x               change to r
  0  0.2000000000    +0.00e+00
 50  0.3686282977    +0.00e+00
100  0.9258735893    +0.00e+00
150  0.8891185779    +0.00e+00
175  0.7408600980    -2.72e-02
176  0.7435247144    -6.48e-04
177  0.7435897066    -3.69e-07
200  0.7435897436    +0.00e+00
250  0.7435897436    +0.00e+00
300  0.7435897436    +0.00e+00
largest change: 0.70 per cent of r

```

Python zapisuje $-2,72 \times 10^{-2}$ jako $-2.72e-02$, czyli $-0,0272$. Czytaj wydruk z góry na dół. Przez 100 kroków orbita wędruje. Sterownik zostaje włączony i przez 75 kroków nie robi nic, bo orbita jest poza zasięgiem. W kroku 175 wchodzi w zasięg, dostaje pchnięcie i łąduje znacznie bliżej; każde następne pchnięcie jest dużo mniejsze od poprzedniego, a od kroku 200 wydruk pokazuje dla x wciąż te same dziesięć cyfr.



Rysunek 17.2: Odwzorowanie logistyczne przy $r = 3,9$, zostawione w spokoju na 100 kroków, a potem sterowane. Linia przerywana to punkt stały. Poniżej zmiana wprowadzana do r : zero, poza jednym małym spadkiem w chwili złapania.

Kuszącą alternatywą jest siła: kręcić pokrętkiem, aż chaos ustanie. Rozdział 5 pokazał, że punkt stały jest stabilny tylko wtedy, gdy r jest mniejsze niż 3. Ustaw $r = 2,9$ i już tak zostaw: orbita rzeczywiście się ustali, w punkcie 0,655. To zmiana o 26 procent wartości r , wprowadzona na stałe, a nie daje ci nawet tego stanu, którego chciałeś: punkt stały spokojnego odwzorowania, $1 - 1/2,9$, leży gdzie indziej. Wartość 0,744 jest punktem stałym tylko przy $r = 3,9$, gdzie jest niestabilna, więc żadne stałe ustawienie pokrętkła nie da ci tego stanu. Sterownik utrzymuje go za drobne: wszystkie jego pchnięcia razem dają mniej niż jedną trzydziestą zmiany, jaką siła wprowadza w każdym pojedynczym kroku.

Małe pchnięcia mają swoją cenę, a płaci się ją czasem. Sterownik może działać tylko wtedy, gdy orbita jest w zasięgu, a zasięg rośnie proporcjonalnie do największego pchnięcia, na jakie pozwalasz.

Zatrzymaj się i pomyśl 17.5

Pozwalasz na pchnięcia dziesięć razy mniejsze: jedną dziesiątą procenta wartości r . Co się dzieje z czekaniem przed złapaniem?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Staje się mniej więcej dziesięć razy dłuższe. Uśrednione po tysiącu punktów startowych czekanie trwa 31 kroków przy pchnięciach do dziesięciu procent, 360 przy jednym procencie i 3828 przy jednej dziesiątej procenta.

code/ch17/waiting.py

```

15 def wait(x: float, largest: float) -> int:
16     """Steps from x until the first nudge the controller is allowed."""
17     n = 0
18     while nudge(x, largest) == 0.0:
19         x = step(x, 0.0)
20         n += 1
21     return n

```

Listing 17.4: Ile czasu do pierwszego pchnięcia?

largest nudge	average wait	longest wait
10.0 per cent	31	199
1.0 per cent	360	2202
0.1 per cent	3828	29737

Zasięg maleje dziesięciokrotnie, więc orbita ma dziesięć razy węższy cel do trafienia i średnio potrzebuje dziesięć razy więcej czasu, żeby go znaleźć. Dla żadnego pojedynczego przebiegu nie da się z góry powiedzieć, jak długo potrwa czekanie – to pierwsza wizyta chaotycznej orbity – a najdłuższy z tysiąca przebiegów przy jednym procencie czekał 2202 kroków. Średnia, tak jak statystyki z rozdziału 9, jest stabilna.

Laboratorium 17.1 – Twój własny sterownik

Napisz `nudge` i `hold` dla odwzorowania logistycznego przy dowolnym r , przy którym punkt stały $1 - 1/r$ stał się niestabilny. W punkcie stałym nachylenie wynosi $r(1 - 2x)$, a dźwignia $x(1 - x)$. Testy łapią i trzymają orbitę przy $r = 3,9$ i jeszcze raz przy $r = 3,7$ oraz sprawdzają, że żadne pchnięcie nie jest większe, niż pozwoliłeś.

Plik startowy: code/labs/ch17/l17_01_controller.py

Test: code/labs/ch17/test_l17_01_controller.py

Uruchom: `cd code && uv run pytest -k l17_01_controller`

17.4 Dwa układy chaotyczne, które się zgadzają

W rozdziale 9 trzy równania Lorenza uruchomiono dwa razy, ze startów różniących się o włos, i oba przebiegi rozeszły się na dobre. Dwa układy chaotyczne zestawione same sobie nigdy nie pozostają razem. W 1990 roku Louis Pecora i Thomas Carroll zapytali, co się dzieje, gdy jeden z nich nie jest zostawiony sam sobie. Zbuduj kopię układu Lorenza, która ma własne y i z , podlegające tym samym regułom co w oryginale, ale nie ma własnego x : w każdej chwili dostaje zamiast niego x oryginału.



Rysunek 17.3: Napęd i odpowiedź. Oryginał działa swobodnie; jedna z jego trzech liczb trafia do kopii, która ma tylko dwie pozostałe.

Zatrzymaj się i pomyśl 17.6

Niech y i z kopii startują od zera, w odległości 36,7 od wartości oryginału. Oba układy są chaotyczne. Czy y i z kopii oddalą się od wartości oryginału, tak jak dwa przebiegi z rozdziału 9, czy się zbliżą?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Zbliżą się, i to szybko. Odległość spada poniżej jednej milionowej w chwili 9,8, a w chwili 20,5 obie liczby kopii są równe liczbom oryginału w każdej cyfrze, jaką przechowuje komputer.

code/ch17/sync.py

```

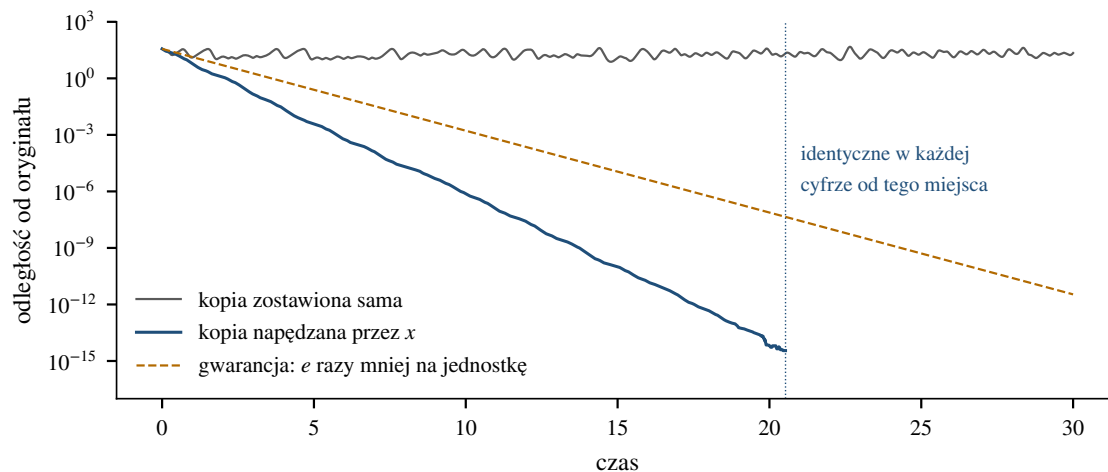
20 def drive_and_copy(s: tuple[float, ...]) -> tuple[float, ...]:
21     """The original (x, y, z) and a copy (y2, z2) handed the original's x."""
22     x, y, z, y2, z2 = s
23     return (SIGMA * (y - x),
24             x * (RHO - z) - y,
25             x * y - BETA * z,
26             x * (RHO - z2) - y2,      # the copy's rules, fed the SAME x
27             x * y2 - BETA * z2)
  
```

Listing 17.5: Oryginał i kopia, która dostaje jego x

time	driven copy	copy left alone
0.0	3.67e+01	3.67e+01
5.0	3.89e-03	1.11e+01
10.0	7.29e-07	2.64e+01
15.0	1.01e-10	1.24e+01
20.0	6.22e-15	2.78e+01
25.0	0.00e+00	2.16e+01
30.0	0.00e+00	2.21e+01

Środkowa kolumna to kopia, która dostaje x . Ostatnia to druga kopia, z tego samego błędnego startu, zostawiona sama sobie. Jedyna różnica między nimi to ta jedna przekazywana liczba, i to ona robi całą różnicę: kopia zostawiona sama sobie nigdy nie zbliża się do oryginału bardziej niż na 5,5, a odległość kopii napędzanej maleje mniej więcej 6 razy w każdej jednostce czasu. Gdy już zgodzą się co do ostatniej cyfry, tak zostaje, bo od tej chwili komputer wykonuje dwa razy te same rachunki.

Dlaczego to działa? Kopia nie należy do pętli, która czyni oryginał chaotycznym. W układzie Lorenza x napędza y i z , a y i z napędzają z powrotem x , i mała różnica krąży po tej pętli i rośnie. Kopia tylko słucha. Gdy kopia dostaje to samo x , jej y i z zapominają, skąd wystartowały, tak jak dwa kubki herbaty w jednym pokoju zapominają o dzielącej je różnicy w rozdziale 1.



Rysunek 17.4: Odległość każdej z kopii od oryginału w skali logarytmicznej. Kopia, która dostaje x , zgrywa się z oryginałem wzdłuż linii prostej, szybciej niż w gwarantowanym tempie; zostawiona sama sobie, pozostaje tak daleko, jak pozwala atraktor.

Pułapka

„Dwa układy chaotyczne nigdy nie mogą się zgadzać.” Dwa układy chaotyczne zostawione same sobie nigdy nie pozostaną razem, bo każdy wzmacnia każdą różnicę, jaka jest między nimi. Sprzężone tak, że jeden słucha drugiego, mogą zgadzać się dokładnie, razem z całym swoim chaosem. Zwróć uwagę, czym ta zgodność nie jest: kopia zgadza się z oryginałem teraz, a nie z jego przyszłością. Synchronizacja to nie przewidywanie.

Czego tu nie dowodzimy

Dlaczego to musi działać, cokolwiek robi x . Oznacz przez e_y i e_z to, jak daleko y i z kopii są od wartości oryginału. Odejmij reguły kopii od reguł oryginału i policz, jak szybko zmienia się $e_y^2 + e_z^2$: każdy wyraz zawierający x się skraca, a to, co zostaje, $-2e_y^2 - 2\beta e_z^2$ przy $\beta = 8/3$, nigdy nie jest dodatnie. Z tej jednej linijki wynika, że odległość maleje co najmniej e razy, czyli mniej więcej 2,718 razy, w każdej jednostce czasu; zmierzony czynnik 6 jest lepszy od gwarancji. Rozumowanie tego rodzaju jest w podręczniku Strogatza *Nonlinear Dynamics and Chaos*. Nie każdy wybór przekazywanej liczby działa, a artykuł Pecory i Carrola podaje test, który to rozstrzyga.

Laboratorium 17.2 – Ile czasu potrzebuje kopia?

Napisz `copy_field`, pięć szybkości zmian oryginału i kopii; `error`, odległość między `y` i `z` kopii a wartościami oryginału; oraz `sync_time`, która prowadzi pięć liczb razem krok po kroku i zwraca pierwszą chwilę, w której błąd spada poniżej zadanej tolerancji. Testy sprawdzają, że trzy kolejne rzędy wielkości zajmują nie więcej czasu, niż pozwala powyższa gwarancja.

Plik startowy: `code/labs/ch17/l17_02_sync.py`

Test: `code/labs/ch17/test_l17_02_sync.py`

Uruchom: `cd code && uv run pytest -k l17_02_sync`

17.5 Co już z tym zrobiono, a czego nie

Wynik Pecory i Carrola od razu podsunął pomysł na dochowanie tajemnicy. Dodaj słabą wiadomość do x oryginału i wyślij sumę. Każdemu, kto podsłuchuje, wydaje się to szumem. Odbiornik zbudowany jako pasująca kopia zgrywa się z chaotyczną częścią sygnału, a odjęcie tego, co odtworzy, zostawia wiadomość. Wkrótce potem zbudowano układy elektroniczne, które to robią. Późniejsze prace pokazały, że wiele takich schematów może złamać podsłuchujący, który przeanalizuje sygnał, a chaos nie zastąpił zwykłego szyfrowania.

Gdzie to spotkasz

W ciągu kilku lat od 1990 roku metoda małych pchnięć utrzymała chaotyczne lasery i układy elektroniczne na orbitach, których same by nie utrzymały. W 1992 roku Alan Garfinkel i współpracownicy zastosowali ją w doświadczeniu na wyizolowanej tkance serca królika, której skurcze stały się nieregularne pod wpływem leku: bodźce elektryczne, zgrane w czasie według tej samej logiki „czekaj, potem pchnij”, przywróciły regularny rytm. Misje kosmiczne korzystają z tej samej obosieczności wrażliwości. W pobliżu miejsc, gdzie przyciąganie Słońca, Ziemi i Księżyca prawie się równoważy, problem trzech ciał z rozdziału 13 jest chaotyczny, a małe odpalenie silników bardzo zmienia tor lotu. Sonda NASA ISEE-3 została w ten sposób wyprowadzona ze swojego stanowiska między Słońcem a Ziemią, dzięki serii bliskich przelotów obok Księżyca i przy niewielkim zużyciu paliwa, a w 1985 roku przeleciała przez warkocz komety.

Ostrożnie

Zachowaj proporcje. Sterowanie chaosem wykazano w laboratorium wiele razy. Nie jest to rutynowe leczenie kliniczne ani standardowa technika przemysłowa; to, czy nieregularność samego serca w ogóle jest chaosem, rozdział 15 nazywa sprawą sporną.

Oto co ten rozdział dodaje do argumentu książki. Część II zmierzyła wrażliwość jako ograniczenie: to przez nią horyzont prognozy jest krótki. Jest ona jednak także dźwignią. Spokojny układ ma jedno zachowanie, a jego zmiana wymaga siły. Układ chaotyczny jest pełen zachowań, których nie potrafi utrzymać, i każde z nich można wybrać i utrzymać za

drobne, jeśli tylko zechcesz patrzeć i czekać. Chaos to nie tylko granica tego, co możesz wiedzieć. To zasób. Rozdział 18 zbiera wszystko, co zmierzyła ta książka, w przewodnik po życiu z jednym i drugim.

Podsumowanie

- Przy $r = 3,9$ odwzorowanie logistyczne ma **punkt stały, którego nie potrafi utrzymać**: jeden krok mnoży chybiecie przez nachylenie, $-1,9$. Chaotyczna orbita wciąż przechodzi blisko niego i blisko wielu **niestabilnych cykli**.
- **Sterownik** czeka na bliskie przejście, a potem zmienia **parametr r** o tyle, ile znosi nadchodzące chybiecie: chybiecie razy nachylenie, podzielone przez **dźwignię**, ze zmienionym znakiem. Przewiduje tylko o jeden krok naprzód, a na to chaos pozwala.
- Gdy orbita zostanie złapana, pchnięcia maleją prawie do zera. Siła, czyli duża i trwała zmiana r , nie daje nawet tego samego stanu. Mniejsze pchnięcia kosztują dłuższe **czekanie**, którego średnia jest stabilna.
- Kopia układu Lorenza, która dostaje x oryginału, zgrywa się z nim: to **synchronizacja**. Zostawiona sama sobie nigdy tego nie robi. Zgodność to nie przewidywanie.
- Sterowanie chaosem i synchronizacja to **wyniki laboratoryjne** – w laserach, układach elektronicznych i tkance serca – oraz narzędzie sterowania lotem sond kosmicznych. Wrażliwość jest **zasobem**, a nie tylko ograniczeniem.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

znaleźć punkt stały odwzorowania logistycznego, którego reguła nie potrafi utrzymać, i policzyć, jak często chaotyczna orbita zbliża się do niego	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
obliczyć małą zmianę parametru, która znosi chybiecie, jakie zrobiłby jeden krok	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
uruchomić sterownik małych pchnięć na chaotycznym odwzorowaniu i zmierzyć, jak długo czeka i jak duże są jego pchnięcia	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
napędzać kopię układu Lorenza jedną liczbą z oryginału i zmierzyć, jak z upływem czasu zmienia się odległość między nimi	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, gdzie wykazano, że sterowanie chaosem i synchronizacja działają, i czym one nie są	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dla odwzorowania logistycznego przy $r = 3,5$: jaki jest punkt stały $1 - 1/r$, jakie jest w nim nachylenie $2 - r$ i czy orbita może w nim pozostać?
2. Przy $r = 3,9$ orbita jest o 0,002 poniżej punktu stałego. Gdzie umieści ją jeden krok bez sterowania i jaka zmiana r znosi chybiecie?
3. Dlaczego sterownik może kierować układem chaotycznym, skoro nikt nie potrafi prognozować go daleko naprzód?
4. Zmniejszasz o połowę największe dozwolone pchnięcie. Co się dzieje z zasięgiem sterownika i mniej więcej ze średnim czekaniem?
5. Dlaczego ustawienie na stałe $r = 2,9$ nie daje ci stanu 0,744?
6. W schemacie Pecory i Carrola co przekazuje się od oryginału do kopii i czy kopia przewiduje wtedy przyszłość oryginału?
7. Wymień dwa miejsca, w których wykazano sterowanie chaosem, i powiedz, czym ono nie jest.

Po co ci chaos: przewodnik terenowy po przewidywaniu

Oto cztery przewidywania, na które możesz trafić w ciągu jednego tygodnia. Tablica mówi, że całkowite zaćmienie Słońca nastąpi 2 sierpnia 2027 roku. Aplikacja pogodowa podaje 70 procent szans na deszcz w sobotę. Wyciąg z funduszu emerytalnego mówi, ile będą warte twoje oszczędności w 2060 roku. A program komputerowy odpowiada na pytanie o przyszłoroczne stopy procentowe bez cienia wątpliwości.

Któremu z nich zaufasz i jak daleko? Teraz możesz na to odpowiedzieć liczbami, które sam policzyłeś. Ten rozdział zamienia książkę w przewodnik terenowy: pięć pytań, które warto zadać każdemu przewidywaniu, każde z odpowiedzią opartą na pomiarze, a potem argument, dlaczego każdy ich potrzebuje.

Czego się nauczysz

Po przerobieniu tego rozdziału będziesz umieć:

- zapytać o każde przewidywanie, czy mała zmiana stanu początkowego rośnie
- policzyć horyzont prognozy i powiedzieć, co dają lepsze dane
- powiedzieć, co da się przewidzieć za horyzontem, i czytać szanse
- odróżnić chaos od szumu, a małą przyczynę od dużego skutku
- zastosować siedem pytań przewodnika do prognozy albo modelu
- wyjaśnić, liczbami, które sam policzysz, dlaczego jedne rzeczy da się przewidzieć na stulecia, a inne tylko na dni

18.1 Pytanie pierwsze: czy mała zmiana rośnie?

Zacznij od najczystszej pary z listy: zaćmienia i deszczu. Obie prognozy powstają tak samo. Komputer bierze zmierzoną teraźniejszość, stosuje dokładne prawa fizyki i idzie krok po kroku naprzód w czasie.

Zatrzymaj się i pomyśl 18.1

Tablica zaćmień jest dobra na tysiące lat, a szczegółowa prognoza pogody na mniej więcej tydzień. Obie powstają z dokładnych praw, stosowanych krok po kroku od zmierzonego stanu początkowego. Na czym polega różnica, w języku tej książki?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Na tym, jak szybko rośnie mały błąd stanu początkowego. W ruchach Ziemi i Księżyca rośnie tak wolno, że przez tysiąclecia pozostaje mały; w powietrzu rośnie, aż staje się tak duży jak sama pogoda.

Sprawdza się to testem, który rozdział 6 zamienił w pomiar: uruchom regułę dwa razy, z dwóch stanów początkowych bliższych sobie, niż potrafiłby je rozróżnić jakikolwiek przyrząd, i patrz na różnicę. Oto trzy reguły, które już znasz z tej książki, każda uruchomiona dwa razy z odległości jednej miliardowej:

code/ch18/grows.py

```

24 print("gap after step:      0      10      20      30      40")
25 for name, rule, start in RULES:
26     a, b = start, start + 1e-9      # two starts a billionth apart
27     gaps = []
28     for step in range(41):
29         if step % 10 == 0:
30             gaps.append(f"{abs(b - a):8.1e}")
31         a, b = rule(a), rule(b)
32     print(f"{name:18s}" + " ".join(gaps))

```

Listing 18.1: Pierwsze pytanie, zadane trzem regułom

gap after step:	0	10	20	30	40
cooling tea	1.0e-09	3.5e-10	1.2e-10	4.2e-11	1.5e-11
logistic, r = 2.8	1.0e-09	9.1e-11	1.0e-11	1.1e-12	1.2e-13
logistic, r = 4	1.0e-09	4.6e-07	5.4e-04	6.1e-01	2.4e-01

Stygnaąca herbata zapomina o zaburzeniu: różnica jest co minutę mnożona przez 0,9. Odwzorowanie logistyczne przy $r = 2,8$ też zapomina, bo osiada w swoim punkcie stałym. Przy $r = 4$ jedna miliardowa rośnie, aż staje się tak duża, jak pozwala samo odwzorowanie. Jeśli różnica maleje, trajektorię da się przewidzieć tak długo, jak długo reguła jest poprawna. Jeśli rośnie, następne pytanie brzmi: jak szybko?

Pytanie pierwsze ma przed sobą jeszcze jedno: czy w ogóle istnieje dokładna reguła? Dla planet i dla trzech równań Lorenza istnieje. W gospodarce czy w epidemii sama reguła zmienia się, gdy ludzie reagują na prognozę, a to jest inna trudność niż chaos.

18.2 Pytanie drugie: jak daleko jest horyzont?

Rosnący błąd jest zwykle w każdym kroku mnożony przez mniej więcej ten sam czynnik, a rozdział 7 zamienił ten czynnik w jedną liczbę: wykładnik Lapunowa, zapisywany λ (grecka litera lambda), czyli logarytm naturalny $\ln$ średniego czynnika. Czynnik 2 to wykładnik $\ln 2$, mniej więcej 0,693. Załóżmy, że stan początkowy jest błędny o e_0 , a prognoza staje się bezużyteczna, gdy błąd przekroczy tolerancję T . Po n krokach błąd to e_0 pomnożone n razy przez ten czynnik, więc sięga T , gdy

$$n \approx \frac{\ln(T/e_0)}{\lambda}.$$

Słowami: horyzont to logarytm tego, ile razy za mały jest twój błąd, podzielony przez wykładnik. Biblioteka książki ma to w jednym wierszu ($e^{**exponent}$ to tam czynnik, odtworzony z wykładnika):

code/src/chaoslab/measure.py

```

65 def horizon(exponent: float, error: float, tolerance: float) -> float:
66     """Steps (or time) until an error of size `error` grows to `tolerance`,
67     if it grows by the factor  $e^{**exponent}$  per step."""
68     return math.log(tolerance / error) / exponent

```

Listing 18.2: Wzór na horyzont w bibliotece książki

Wypróbuj go na odwzorowaniu logistycznym przy $r = 4$, którego wykładnik ten rozdział mierzy jako 0,693 na krok ($\ln 2$, jak pokazuje rozdział 7). Przyjmij, że znasz stan początkowy z dokładnością do sześciu miejsc po przecinku, więc e_0 to jedna milionowa, i uznaj prognozę za bezużyteczną, gdy myli się o więcej niż $T = 0,1$. Wtedy $T/e_0 = 100\,000$, a $n \approx 11,513/0,693 \approx 16,6$ kroku.

Zatrzymaj się i pomyśl 18.2

Kupujesz przyrząd tysiąc razy dokładniejszy i teraz znasz stan początkowy z dokładnością do dziewięciu miejsc po przecinku. Jak długo wytrzyma prognoza: tysiąc razy dłużej, mniej więcej trzy razy dłużej, czy o kilka kroków dłużej?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. O kilka kroków dłużej: mniej więcej o 10,0 więcej, czyli razem 26,6. Podzielenie e_0 przez tysiąc dodaje $\ln 1000/\lambda = 6,908/0,693$ kroku i nic więcej.

To odpowiedź wzoru. Oto odpowiedź samego odwzorowania, zmierzona na piechotę dla 201 różnych stanów początkowych:

code/ch18/thousandfold.py

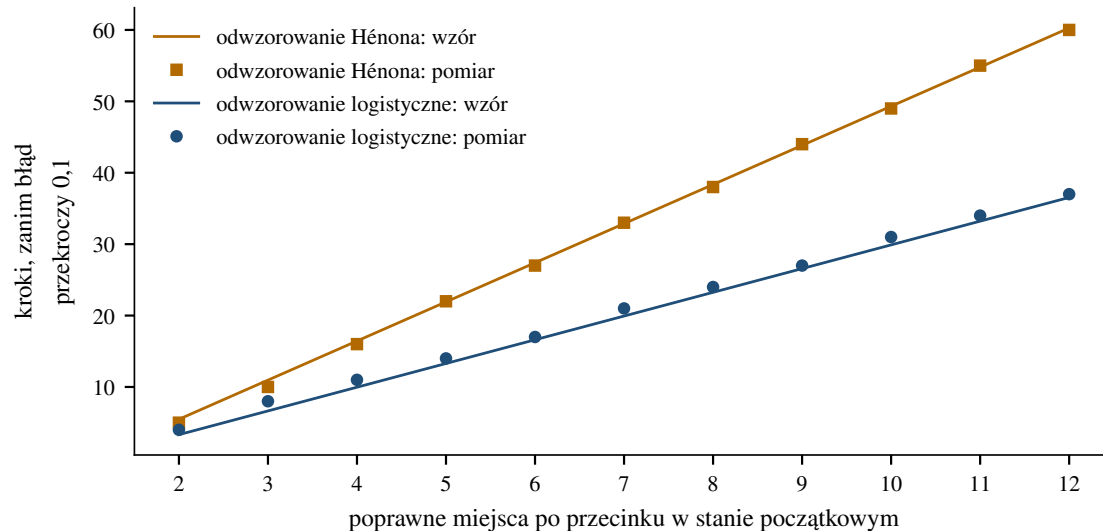
```

20 def typical_horizon(error: float, tolerance: float = 0.1) -> float:
21     """The median, over many starts, of the steps two runs `error` apart
22     stay within `tolerance` of each other."""
23     return median(steps_until_apart(RULE, x0, error, tolerance)
24                   for x0 in STARTS)
25
26
27 def main() -> None:
28     six = typical_horizon(1 / 10**6)      # start known to six digits
29     nine = typical_horizon(1 / 10**9)     # a thousand times better
30     print(f"start right to 6 digits: {six:.0f} steps")
31     print(f"start right to 9 digits: {nine:.0f} steps")
32     print(f"a thousandfold better measurement bought {nine - six:.0f} steps")

```

Listing 18.3: Tysiąc razy lepsze dane, zmierzone

start right to 6 digits: 17 steps
 start right to 9 digits: 27 steps
 a thousandfold better measurement bought 10 steps



Rysunek 18.1: Jak daleko sięga prognoza w zależności od tego, ile miejsc po przecinku stanu początkowego jest poprawnych. Linie to wzór, punkty to mediana wielu zmierzonych par przebiegów. Każda dodatkowa cyfra kupuje tych samych kilka kroków.

Wzór mówił 16,6 i 26,6; odwzorowanie mówi 17 i 27, czyli o 10 więcej. Każda dodatkowa poprawna cyfra kupuje tę samą liczbę kroków, $\ln 10/\lambda = 2,303/0,693$, tutaj mniej więcej 3,3. Horyzont rośnie jak logarytm twojej dokładności, więc lepsze przyrządy i większe komputery mogą tylko dodać stały odcinek. Mały kalkulator zestawia obok siebie trzy układy z tej książki:

code/ch18/horizon.py

```

16 def forecast_length(exponent: float, digits: int,
17                     tolerance: float = 0.1) -> float:
18     """How far ahead a forecast stays within `tolerance`, when the start is
19     right to `digits` decimal places and an error is multiplied by
20     e**exponent every step (or every unit of time)."""
21     error = 1 / 10**digits          # six digits: an error of 0.000001
22     return horizon(exponent, error, tolerance)

```

Listing 18.4: Kalkulator horyzontu

	exponent	horizon with the start right to		
		3 digits	6 digits	9 digits
logistic map	0.693	6.6	16.6	26.6 steps
Henon map	0.420	11.0	27.4	43.8 steps
Lorenz system	0.909	5.1	12.7	20.3 time units

Błędy odwzorowania Hénona rosną wolniej, z wykładnikiem 0,420 na krok, więc każda cyfra kupuje mu 5,5 kroku; układ Lorenza, przy 0,909 na jednostkę czasu, zyskuje 2,5 jednostki czasu na cyfrę. Na wykresie w zależności od liczby poprawnych cyfr każdy horyzont jest linią prostą (rysunek 18.1).

Ta sama arytmetyka wyjaśnia zaćmienie. Obliczenia Układu Słonecznego, które Jacques Laskar przeprowadził w 1989 roku, przytoczone w rozdziale 13, dały planetom wewnętrznym wykładnik około jednej piątej na milion lat: czas Lapunowa, czyli odwrotność wykładnika, wynosi mniej więcej pięć milionów lat. Przyjmij, że znasz położenie Ziemi z dokładnością do metra, i uznaj prognozę za bezużyteczną, gdy błąd dorówna rozmiarom jej orbity, 150 milionom kilometrów. Horyzont to pięć milionów lat razy $\ln 150\,000\,000\,000 \approx 25,7$, czyli jakieś 130 milionów lat, a milimetr dodałby tylko około 35 milionów. Układ Słoneczny jest chaotyczny; w skali tablicy zaćmień jego błędy jeszcze nie zaczęły mieć znaczenia.

Laboratorium 18.1 – Kalkulator horyzontu

Napisz `forecast_length`, która zamienia wykładnik i liczbę poprawnych cyfr w horyzont; `extra_steps`, która mówi, co daje dokładniejszy pomiar; oraz `digits_needed`, która stosuje wzór w drugą stronę. Potem zapytaj, ilu cyfr potrzebowałoby sto kroków odwzorowania logistycznego, i porównaj to z mniej więcej szesnastoma cyframi dziesiętnymi, które mieszczą zwykłe liczby komputera (rozdział 16).

Plik startowy: `code/labs/ch18/l18_01_horizon.py`

Test: `code/labs/ch18/test_l18_01_horizon.py`

Uruchom: `cd code && uv run pytest -k l18_01_horizon`

18.3 Pytanie trzecie: co przetrwa horyzont?

Za horyzontem trajektoria jest stracona. To nie to samo co brak wszelkiej wiedzy, i właśnie tę część teorii chaosu najczęściej się przeocza.

Zatrzymaj się i pomyśl 18.3

Czterdzieści kroków naprzód trajektoria odwzorowania logistycznego jest dawno stracona. Czy o kroku czterdziestym da się jeszcze coś przewidzieć?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tak: szanse. Szansa, że x jest poniżej 0,25 w kroku czterdziestym, wynosi mniej więcej jedną trzecią, i taki sam jest ułamek czasu, który spędza tam każdy długi przebieg. Trajektoria przepadała; jej statystyki nie.

Twórcy prognoz uzyskują takie szanse, uruchamiając model wiele razy, ze stanów początkowych rozłożonych po wszystkim, na co pozwala pomiar, i licząc. To jest zespół, a rozdział 14 buduje go dla pogody. Oto najmniejsza wersja:

code/ch18/beyond.py

```

18 def odds(centre: float, width: float, steps: int) -> list[float]:
19     """After each step, the fraction of MEMBERS runs below 0.25. The runs
20     start spread evenly across `width` around `centre`."""
21     runs = [centre + width * (k / (MEMBERS - 1) - 0.5)
22             for k in range(MEMBERS)]
23     fractions = []
24     for _ in range(steps + 1):
25         fractions.append(sum(1 for x in runs if x < 0.25) / MEMBERS)
26         runs = [RULE(x) for x in runs]
27     return fractions
28
29
30 def long_run(start: float, steps: int = 100_000) -> float:
31     """The fraction of one long run's steps spent below 0.25."""
32     x, below = start, 0
33     for _ in range(steps):
34         below += x < 0.25
35         x = RULE(x)
36     return below / steps

```

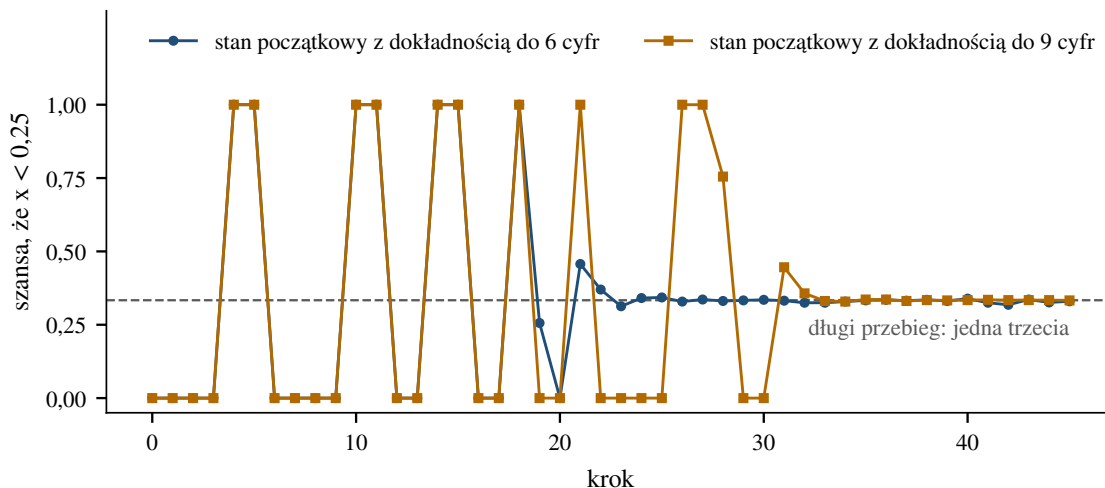
Listing 18.5: Prognoza z zespołu i jeden długi przebieg

chance that $x < 0.25$, start right to		
step	6 digits	9 digits
0	0.00	0.00
4	1.00	1.00
8	0.00	0.00
12	0.00	0.00
16	0.00	0.00
20	0.00	0.00
24	0.34	0.00
28	0.33	0.76
32	0.33	0.36
36	0.34	0.34
40	0.34	0.33

one long run from 0.3: 0.33 of its steps below 0.25
one long run from 0.8: 0.33 of its steps below 0.25

Gdy stan początkowy jest poprawny do sześciu cyfr, wszystkie przebiegi, a jest ich 1000, zgadzają się aż do kroku 18, a prognoza jest jedną pewną odpowiedzią. Potem się rozchodzą, prognoza staje się szansami, a po kilku krokach szanse ustalają się na 0,33: ułamku czasu, który jeden długi przebieg spędza poniżej 0,25, z każdego z dwóch stanów początkowych. Przy dziewięciu cyfrach przebiegi zgadzają się aż do kroku 27, a potem dzieje się to samo (rysunek 18.2).

Prognoza układu chaotycznego ma więc trzy postacie, jedna po drugiej: trajektorię, dopóki przebiegi się zgadzają, szanse, gdy się rozchodzą, i potem statystyki długiego przebiegu. Te ostatnie rozdział 9 nazywa statystykami atraktora, a w przypadku pogody to jest klimat.



Rysunek 18.2: Prognoza zespołowa szansy, że x jest poniżej 0,25. Jest pewna, dopóki przebiegi się zgadzają, podaje szanse, gdy się rozchodzą, i ustala się na ułamku z długiego przebiegu. Trzy cyfry więcej to opóźniają, ale nie powstrzymują.

Pułapka

„Nieprzewidywalne znaczy niepoznawalne.” Za horyzontem nie da się poznać trajektorii, a prawie wszystko inne tak: dokąd układ może dojść, a dokąd nie, jak często odwiedza każdy obszar, jakie są szanse w każdym pytaniu, jakie zechcesz zadać. 70 procent szans na deszcz to nie wykręt synoptyka; to poprawna odpowiedź na pytanie, na które chaos uniemożliwił odpowiedź „tak” albo „nie”. Projekcja klimatyczna prognozuje statystyki, nie wtorki.

Gdzie to spotkasz

„Stożek niepewności”, który amerykańskie Narodowe Centrum Huraganów (National Hurricane Center) rysuje wokół prognozowanego toru burzy, rozszerza się z każdym dniem wyprzedzenia. Jego rozmiar wyznaczają błędy niedawnych prognoz samego centrum, tak aby środek burzy pozostawał w nim mniej więcej dwa razy na trzy: horyzont i zestaw szans na jednym obrazku, publikowany po to, by działały według niego miliony ludzi.

18.4 Pytanie czwarte: chaos, szum czy mała przyczyna?

Nieregularne dane zachęcają do opowieści, a dwie z nich wymagają osobnego pytania.

Pierwsza mówi: „to jest chaos”. Rozdział 15 pokazuje, że szereg chaotyczny i losowy mogą mieć ten sam histogram i że rozróżnia je struktura: nanieś każdą wartość względem następnej, a chaos narysuje krzywą tam, gdzie szum rysuje chmurę, i chaos da się prognozować krok lub dwa naprzód tam, gdzie szumu nie. Ma znaczenie, co masz, i ile jest danych: krótki, zaszumiony zapis może wyglądać prawie jak cokolwiek.

Druga opowieść biegnie wstecz, od dużego skutku do małej przyczyny.

Zatrzymaj się i pomyśl 18.4

Pchnij stygnącą herbatę i odwzorowanie logistyczne przy $r = 4$ o jedną miliardową każde, jak w podrozdziale 18.1. Które pchnięcie coś zmieniło czterdzieści kroków później? Czy zmieniło ułamek czasu, który odwzorowanie spędza poniżej 0,25?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Tylko pchnięcie odwzorowania logistycznego i tylko jego trajektorię: różnica urosła do rozmiarów samego odwzorowania. Ułamek czasu poniżej 0,25 to wciąż mniej więcej jedna trzecia, z każdego stanu początkowego. Pchnięcie herbaty wygasło.

Pułapka

„Małe przyczyny zawsze mają znaczenie.” W układzie stabilnym wygasają, jak w przypadku herbaty. W chaotycznym mała przyczyna zmienia szczegóły za horyzontem i nic w statystykach, a ponieważ każda inna mała przyczyna też zmienia szczegóły, żadnej nie da się potem wskazać jako powodu. Motyl z rozdziału 6 nie spowodował burzy. Obwinianie jednej małej przyczyny o duży skutek po fakcie to opowieść, nie pomiar.

18.5 Pytanie piąte: czy da się tym sterować?

Ostatnie pytanie zamienia ograniczenie w narzędzie.

Zatrzymaj się i pomyśl 18.5

Jeśli drobna zmiana stanu początkowego całkowicie zmienia przyszłość, to czy jest to zła, czy dobra wiadomość dla kogoś, kto chce układem sterować?

Odpowiedz, zanim pójdziesz dalej.

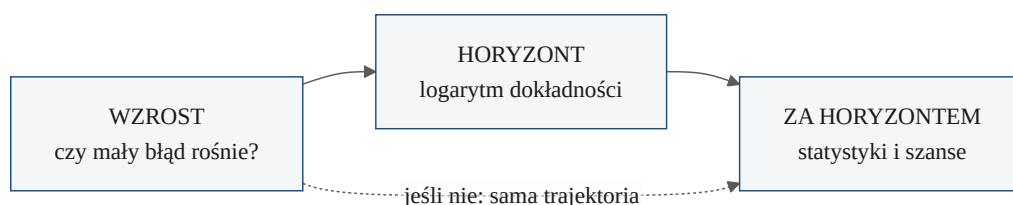
Odpowiedź. Jedno i drugie. Właśnie dlatego nie da się prognozować układu daleko naprzód i właśnie dlatego drobne, dobrze wymierzone pchnięcie może go przesunąć daleko.

Edward Ott, Celso Grebogi i James Yorke uczynili z tego ideę swojej pracy o sterowaniu chaosem z 1990 roku: poczekaj, aż układ przejdzie blisko jednego z niestabilnych cykli ukrytych w jego atraktorze, lekko popchnij parametr, a da się go utrzymać na tym cyklu. Rozdział 17 robi to pchnięciami mniejszymi niż jeden procent i sprawia, że dwa chaotyczne układy idą w równym kroku, jak pokazali w tym samym roku Louis Pecora i Thomas Carroll. Chaos nie jest więc wymówką dla fatalizmu. Układ, którego nie da się prognozować, może być układem, którym da się sterować, a zmiana parametru, na przykład wymuszenia w modelu klimatu, przesuwając statystyki nawet wtedy, gdy żadnej trajektorii nie da się przewidzieć.

18.6 Przewodnik terenowy na jednej stronie

Oto pięć pytań, dwa kolejne, na które książka zapracowała, i miejsce, w którym zmierzyłeś każde z nich:

1. Czy istnieje dokładna reguła i czy mała zmiana stanu początkowego rośnie? (rozdział 6)
2. Jak daleko jest horyzont i co dałyby lepsze dane? (rozdział 7)
3. Co przetrwa horyzont: atraktor, statystyki, szanse? (rozdziały 9 i 14)
4. Czy nieregularność to chaos, czy szum, i czy danych starczy, żeby to rozstrzygnąć? (rozdział 15)
5. Czy da się tym sterować albo przesunąć statystyki parametrem? (rozdział 17)
6. Czy odpowiedź przetrwa mniejszy krok i więcej cyfr? (rozdziały 4 i 16)
7. Czy ktoś po fakcie obwinia małą przyczynę o duży skutek? (rozdział 6)



Rysunek 18.3: Przewodnik terenowy w trzech ruchach. Jeśli mały błąd nie rośnie, przetrwa sama trajektoria; jeśli rośnie, horyzont mówi, jak długo, a za nim pytasz o statystyki i szanse.

A oto znajome prognozy uporządkowane według horyzontu. Tylko jedna liczba jest policzona w tym rozdziale; reszta to rzędy wielkości, niektóre z nich z obiegowej wiedzy, oznaczone według tego, skąd pochodzą.

Prognoza	Horyzont	Skąd ta liczba	Za nim pytaj o
Gdzie są planety	około 130 milionów lat	policzone wyżej	gdzie leżą ich orbity
Zaćmienia	tysiące lat	opublikowane tablice	nic więcej
Pływy astronomiczne	lata	opublikowane tablice	falę sztormową, która jest pogodą
Szczegółowa pogoda	tydzień w praktyce, najwyżej dwa tygodnie	cytowane (Lorenz, 1969)	szanse, potem klimat
Podwójne wahadło	sekundy	rząd wielkości	dokąd pozwala mu jego energia
Kurs akcji z dnia na dzień	nieustalony	nieregularny to nie chaotyczny	rozrzut dawnych zmian

Teraz wyciąg emerytalny i pewny siebie program. Projekcja emerytalna to wynik, jaki dałaby założona stopa zwrotu, więc pytaj o zakres stóp. Pewna siebie odpowiedź o przyszłorocznej gospodarce nie ma horyzontu, dopóki ktoś nie powie, jaka jest jej reguła, a pewny ton to nie przedział błędu.

Laboratorium 18.2 – Przewodnik terenowy na modelu, który sam piszesz

Napisz `error_grows`, `growth_rate` i `long_run_fraction`: trzy pierwsze pytania jako kod, który możesz skierować na dowolną regułę zamieniającą jedną liczbę w następną. Test sprawdza je na stygnącej herbacie i odwzorowaniu logistycznym. Potem napisz na końcu pliku własną regułę (namiot, sinus, populację z odłowem) i zadaj jej wszystkie siedem pytań.

Plik startowy: `code/labs/ch18/l18_02_guide.py`

Test: `code/labs/ch18/test_l18_02_guide.py`

Uruchom: `cd code && uv run pytest -k l18_02_guide`

18.7 Dlaczego każdy tego potrzebuje

Chaos zakończył erę mechanizmu zegara, nie dodając ani jednego losowego składnika: dokładna reguła i prawie dokładny stan początkowy dają przyszłość znaną tylko do horyzontu, który rośnie jak logarytm twojej dokładności.

Mówi, którym prognozą ufać i jak długo. Zaćmienie i burza podlegają prawom tego samego rodzaju; różni je wykładnik. To uczy pokory wobec pewności na długi dystans, bo pojedyncza liczba daleko za horyzontem to zgadywanie przebrane za pomiar, i zaufania do modeli na krótki dystans, bo w obrębie horyzontu są zdumiewająco dobre.

Czyni prawdopodobieństwa szacownymi. Szanse to postać, jaką za horyzontem przybiera poprawna prognoza, a nie porażka prognozowania.

Jest piękny i uniwersalny. Stała Feigenbauma wychodzi z linijki algebry i z prawdziwych cieczy i obwodów (rozdział 8); fraktale nadają linii brzegowej wymiar między jeden a dwa (rozdział 11); zbiór Mandelbrota mieści całą drogę do chaosu (rozdział 12).

I jest zasobem: sterowanie, synchronizacja i wymieszanie, jakie mają mieć liczby losowe komputera (choć rozdział 16 ostrzega, że samo odwzorowanie chaotyczne jest ich kiepskim źródłem).

Teoria chaosu nie jest teorią wszystkiego ani nieładu w ogóle: większość nieregularności świata to szum, niewiedza albo reguła, która ciągle się zmienia.

Zatrzymaj się i pomyśl 18.6

Kolega mówi: „Gospodarka jest chaotyczna, więc każda prognoza jest równie dobra jak inna”. Co jest w tym złe?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Obie połowy. Nikt nie wykazał, że gospodarka podlega stałej chaotycznej regule, a nawet układ chaotyczny nie może robić czegokolwiek: trzyma się swojego atraktora, jego szanse są ustalone, a jego prognoza na krótki dystans może być bardzo dobra.

Pułapka

„**Chaos znaczy: wszystko wolno.**” Chaos to dokładna reguła, której ruch pozostaje w ograniczonym atraktorze, ze stałymi statystykami i horyzontem, który da się policzyć. Nazwanie czegoś chaotycznym to twierdzenie, które da się sprawdzić, a nie licencja na przyklejanie motyli skrzydeł wszystkiemu, co nieprzewidywalne, i nie powód, żeby przestać prognozować.

18.8 Demon Laplace’a po raz ostatni

Rozdział 1 zakończył się zasianiem pytania, a teraz możesz na nie odpowiedzieć.

Zatrzymaj się i pomyśl 18.7

Założmy, że reguła jest dokładna i znasz ją doskonale. Czy przyszłość jest wtedy znana, dowolnie daleko naprzód?

Odpowiedz, zanim pójdziesz dalej.

Odpowiedź. Nie. Jest znana tylko do horyzontu, który rośnie jak logarytm tego, jak dokładnie znasz stan początkowy. Za nim znany jest atraktor, statystyki i szanse.

Demon Laplace’a potrzebuje nieskończenie dokładnej teraźniejszości, a każda dodatkowa cyfra kupuje mu tylko kilka kroków więcej. Mechanizm zegara jest prawdziwy dla zaćmienia i pływów, a kończy się dla pogody, podwójnego wahadła i jednej linijki algebry.

Zostają prognozy, które spotykasz codziennie: pociąg za siedem minut, deszcz w sobotę, projekcja, pewna siebie odpowiedź. Zadaj każdej pięć pytań. Niektórym prognozom zaufasz bardziej niż dotąd, innym znacznie mniej, i po raz pierwszy będziesz umieć powiedzieć dlaczego, liczbą.

Podsumowanie

- **Pytanie pierwsze:** uruchom regułę dwa razy ze stanów początkowych odległych o włos. Jeśli różnica maleje, trajektorię da się prognozować tak daleko, jak długo reguła jest poprawna.
- **Pytanie drugie:** **horyzont** to mniej więcej $\ln(T/e_0)$ podzielone przez wykładnik λ , więc każda poprawna cyfra kupuje tych samych kilka kroków.

- **Pytanie trzecie:** za horyzontem przetrwają **statystyki** i **szanse**, a zespół pokazuje, jak trajektoria zamienia się w szanse.
- **Pytanie czwarte:** nieregularny to nie chaotyczny, dopóki nie pokaże tego struktura, a dużego skutku nie da się potem przypisać jednej małej przyczynie.
- **Pytanie piąte:** wrażliwość czyni chaos **sterowalnym**.
- **Przewodnik terenowy** dodaje dwa pytania i porządkuje znajome prognozy według horyzontu.
- Chaos zakończył erę **mechanizmu zegara** i czyni szanse szacownymi; nie jest licencją na „wszystko wolno”.
- **Demon Laplace’a** znałby przyszłość tylko do horyzontu.

Czy potrafisz?

Oceń się przy każdym punkcie od 1 (nie) do 5 (tak, samodzielnie). Wszystko poniżej 4 nie jest powodem do rozmyślań: wróć do podrozdziału, który tego uczył, i jeszcze raz odpowiedz na jego pytania, zanim pójdziesz dalej.

zapytać o każde przewidywanie, czy mała zmiana stanu początkowego rośnie	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
policzyć horyzont prognozy i powiedzieć, co dają lepsze dane	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
powiedzieć, co da się przewidzieć za horyzontem, i czytać szanse	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
odróżnić chaos od szumu, a małą przyczynę od dużego skutku	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
zastosować siedem pytań przewodnika do prognozy albo modelu	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
wyjaśnić, liczbami, które sam policzysz, dlaczego jedne rzeczy da się przewidzieć na stulecia, a inne tylko na dni	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Punkt kontrolny

Odpowiedz na te pytania na papierze, zanim zajrzysz do dodatku A. Sprawdzają to, czego rozdział uczył, w tej kolejności, w jakiej uczył.

1. Dwa przebiegi reguły zaczynają się w odległości jednej miliardowej; trzydzieści kroków później dzieli je jedna bilionowa. Co mówi pytanie pierwsze?
2. Błędy w pewnym układzie rosną z wykładnikiem 0,5 na dzień. Znasz stan początkowy z dokładnością do 0,001 i możesz tolerować błąd 1. Wiedząc, że $\ln 1000 \approx 6,908$, podaj horyzont.
3. Teraz mierzysz ten stan początkowy dziesięć razy dokładniej. Wiedząc, że $\ln 10 \approx 2,303$, powiedz, o ile dłużej wytrzyma prognoza.
4. Daleko za horyzontem zespół przebiegów odwzorowania logistycznego przy $r = 4$ daje szansę jedną na trzy, że x jest poniżej 0,25. Czy ta prognoza jest bezwartościowa?

5. Szereg codziennych pomiarów wygląda nieregularnie. Wymień dwie rzeczy, które trzeba sprawdzić, zanim nazwie się go chaosem.
6. Dlaczego ta sama własność, która rujnuje dalekie prognozy układu chaotycznego, pozwala nim sterować?
7. Wyjaśnij, dlaczego zaćmienia da się przewidzieć na tysiące lat, a pogodę tylko na mniej więcej tydzień.

Część VI

Dodatki

Odpowiedzi

Odpowiedzi do punktów kontrolnych wszystkich rozdziałów, w kolejności rozdziałów. Każdą zapisano obok jej pytania, a build zebrał je tutaj, więc odpowiedź nie może odpłynąć od pytania, na które odpowiada.

Rozdział 1

1. Stanem jest aktualna godzina i przystanek, na którym jesteś; regułą jest rozkład (razem z założeniem, że autobus się go trzyma); przewidywaniem jest godzina przyjazdu następnego autobusu.
2. Około 19,6 metra w dwie sekundy i około 78,5 w cztery. Dwa razy dłuższy czas, cztery razy dłuższa droga.
3. 74°C , a potem $68,6^{\circ}\text{C}$: różnica zmienia się z 60 na 54, a potem na 48,6.
4. O $1,8$ stopnia: każdy kubek traci jedną dziesiątą własnej różnicy do temperatury pokoju, więc ich różnica też traci jedną dziesiątą.
5. Bo każdy model coś pomija, a użyteczność to kwestia tego, czy jego odpowiedzi są wystarczająco bliskie dla zadawanego pytania, w zakresie, w którym się go używa.
6. Regułę i dokładny stan obecny. Regułę można znać dokładnie; stanu obecnego nigdy, bo każdy pomiar ma błąd.

Rozdział 2

1. 4, 10 i 28.
2. $x \mapsto x - 0,2(x - 20)$, czyli $0,8x + 4$.
3. Około $70/3$, czyli 23 lata. Licząc lata, liczba mieszkańców po raz pierwszy przekracza podwojenie w roku 24; logarytm daje 23,45.
4. $0,2x = 4$, więc $x = 20$, temperatura pokoju; stabilny, bo $|0,8| < 1$.
5. Niestabilny: każdy krok potraja odległość. Orbita z 2 jest o jeden od punktu stałego, potem o 3, 9 i 27: 4, 10, 28.
6. 1 stopień za gorąco: błąd zmalał o połowę i zmienił stronę.

Rozdział 3

1. 1024 na jedenastym i 1023 na pierwszych dziesięciu: każde pole mieści o jedno ziarenko więcej niż wszystkie pola przed nim.
2. Ujemne, dodatnie, ujemne.
3. Zmienia się ono z 0,5 na 0,48 i na 0,42: spada o 0,02 i o 0,08. Dwa razy większe pchnięcie wyrządza cztery razy większą szkodę.
4. Na 0,6 i na 0,2: $1 - 1/2,5$ oraz $1 - 1/1,25$.
5. 0,005 pod punktem stałym, potem 0,0025 nad nim. Ustala się, przestrzeliwując, a każde wahnięcie jest o połowę mniejsze od poprzedniego.
6. Reaguje na zatłoczenie w miarę, jak narasta, a nie z opóźnieniem całego pokolenia: przestrzelenie bierze się z opóźnienia, nie z szybkości wzrostu.

Rozdział 4

1. Około pół kilometra: 90 kilometrów na godzinę to 1,5 kilometra na minutę, a 20 sekund to jedna trzecia minuty.
2. $x = 0,1$ i $v = 1$, więc energia wynosi $0,5 + 0,005 = 0,505$: o jeden procent więcej, czynnik $1 + 0,1 \times 0,1$.
3. Około tysiąca sekund: krok sto razy mniejszy zyskuje sto razy mniej na sekundę.
4. Tempa na początku; pół kroku dalej w pierwszych tempach; pół kroku dalej w drugich; cały krok dalej w trzecich. Przesuwa się ze średnią o wagach 1, 2, 2, 1.
5. Wahanie się tam i z powrotem; przechodzenie przez górę i obracanie się; utratę energii przez tarcie i zatrzymanie się w zwisie.
6. Jej drogi nigdy się nie przecinają, bo każdy stan ma jedną przyszłość, więc ograniczona droga może tylko zatrzymać się w punkcie stałym albo nawinąć na pętlę (Poincaré–Bendixson). Chaos w czasie ciągłym potrzebuje co najmniej trzech.

Rozdział 5

1. $x_1 = 3 \times 0,5 \times 0,5 = 0,75$, a $x_2 = 3 \times 0,75 \times 0,25 = 0,5625$.
2. $1 - 1/2,5 = 0,6$. Nachylenie wynosi tam $2 - 2,5 = -0,5$, między -1 a 1 , więc się utrzymuje.
3. Nachylenie wynosi tam $2 - 3,2 = -1,2$: przesunięcie przy każdym kroku zmienia stronę i rośnie, więc orbita zostaje odepchnięta, w cykl długości dwa.
4. Obiega cykl o okresie cztery; narożniki to cztery wartości, które odwiedza po kolei.
5. Że w badanym odcinku nie znalazł cyklu o długości do sprawdzanej granicy. Nie to, że orbita jest chaotyczna: dłuższy cykl albo orbita, która wciąż się ustala, też dają None.
6. Pogoda albo inny przypadek; błędy w liczeniu; oraz własna deterministyczna reguła wzrostu populacji, jak w odwzorowaniu logistycznym przy $r = 3,9$.
7. Ten sam początek zawsze daje dokładnie tę samą orbitę. Przy $r = 3,9$ nigdy się nie ustala ani nie powtarza, więc nie widać w niej wzoru, choć każda wartość jest wyznaczona przez poprzednią.

Rozdział 6

1. W kroku 30. Aż do kroku 16 różnią się o mniej niż jedną milionową.
2. Po 20 krokach: 2^{20} to trochę ponad milion, a 2^{19} to tylko mniej więcej pół miliona.
3. Bo każde x leży między 0 a 1, więc dwie wartości nigdy nie mogą być od siebie dalej niż o 1. Gdy różnica osiągnie rozmiar całego przedziału, błąka się, średnio wynosząc około 0,4.
4. $\frac{1}{2} + \frac{1}{4}$, więc w systemie dwójkowym 0,11. Połowy to: górna, górna, dolna: 0,75, potem 0,5, potem 0. Odwzorowanie odczytuje cyfry 1, 1, 0.
5. Około 44 kroków. Każda cyfra dziesiętna dodaje trochę ponad trzy kroki, więc cztery kolejne dodają mniej więcej trzynastcie.
6. Ponownie pierwszy przebieg, cyfra po cyfrze, bo reguła jest deterministyczna: ten sam punkt startowy zawsze daje tę samą przyszłość.
7. Nie: każda wartość wynika dokładnie z poprzedniej. Jest deterministyczne, bo ten sam punkt startowy daje tę samą przyszłość; nie jest przewidywalne na długo, bo żaden zmierzony punkt startowy nie ma dość cyfr, żeby reguła mogła z nich korzystać.

Rozdział 7

1. Wcale nie urósł: iloczyn wynosi 1. Zwykła średnia czynników to $\frac{5}{3}$, co sugeruje wzrost, którego nie było.
2. $\ln 20 = \ln 2 + \ln 10 \approx 2,99$. A $0,2 \times 10 = 2$, więc $\ln 0,2 = \ln 2 - \ln 10 \approx -1,61$.
3. $\ln 0,5 \approx -0,69$. Jest ujemny: błędy maleją, w każdym kroku o połowę.
4. Mniej więcej 9: $\ln(0,1/0,001) = \ln 100 = 2 \ln 10 \approx 4,6$, podzielone przez 0,5.
5. Mniej więcej 9 kroków za każdym razem: $\ln 100/0,5 \approx 9,2$. Każda stukrotna poprawa daje tę samą stałą liczbę kroków.
6. Bo horyzont zależy od błędu stanu początkowego tylko przez jego logarytm, a szybszy komputer nie mierzy stanu początkowego ani trochę lepiej. Podwojenie horyzontu wymaga dwa razy więcej poprawnych cyfr stanu początkowego.
7. Okno: wewnątrz chaotycznego zakresu orbita ustaliła się w cyklu, więc błędy znów maleją, a wykładnik spada poniżej zera.

Rozdział 8

1. Gdzie orbita przebywa na dłuższą metę przy tym r , po stanie przejściowym: jedna kropka dla punktu stałego, dwie dla cyklu o okresie dwa, smuga dla chaosu.
2. Około 4,7, z $0,08 \div 0,017$; następny odstęp powinien więc wynosić około $0,017 \div 4,7$, czyli mniej więcej 0,0036.
3. Blisko rozwidlenia cykl, który traci stabilność, mnoży zaburzenie przy każdym obiegu przez prawie -1 , więc ustala się niezwykle powoli, a przebieg o skończonej długości wciąż pokazuje przeskakiwanie między wartościami.
4. To r , przy którym cykl przechodzi przez szczyt garbu, $x = 0,5$, gdzie nachylenie jest zerowe. Jest rozwiązaniem jednego równania (zaczynij od 0,5, zrób tyle kroków, ile wynosi okres, wróć do 0,5), które bisekcja rozwiązuje, połowiąc przedział.
5. Nie. Uniwersalność mówi, że stosunek kolejnych odstępów dąży do tej samej liczby, około 4,6692, dla każdej reguły z jednym gładkim garbem; to, gdzie kończy się każda droga, zależy od reguły.
6. Nie: 3,83 leży w oknie okresu trzy, które otwiera się przy $1 + \sqrt{8}$, około 3,82843. Li i Yorke udowodnili, że ciągła reguła przenosząca przedział w siebie, mająca punkt o okresie trzy, ma cykle każdej długości i punkty startowe, które nigdy nie ustalają się w żadnym cyklu.

Rozdział 9

1. Sprawia, że x rośnie: wir przyspiesza w stronę różnicy temperatur.
2. Nieruchomy garnek, $(0, 0, 0)$, oraz dwa ustalone wiry, $x = y = \pm\sqrt{72}$ (około 8,49) z $z = 27$.
3. Na płaszczyźnie droga nie może przeciąć samej siebie, więc może tylko osiąść, zwijać się spiralą albo krążyć po stałej pętli. Model Lorenza używa trzech, najmniej, ile pozwala drodze przechodzić nad sobą i pod sobą.
4. O około $\ln(1000)/0,9$, czyli mniej więcej 8 jednostek czasu: z mniej więcej dwudziestu do mniej więcej dwudziestu ośmiu, a nie tysiąc razy dłużej.
5. Nie dojdzie: każdy przebieg zostaje ściągnięty na ten sam ograniczony atraktor, na którym z pozostaje mniej więcej między 1 a 48.
6. Połowę. Jednoczesna zmiana znaków x i y nie zmienia reguł, więc oba skrzydła są swoimi lustrzanymi odbiciami.
7. Średnia wysokość z , od 23,48 do 23,57 z pięciu różnych startów; wykładnik Lapunowa, około 0,9, to kolejna.

Rozdział 10

1. W 0,6, potem w 0,8: podwojenie 0,6 daje 1,2, czyli za końcem, więc punkt składa się z powrotem do $2 - 1,2 = 0,8$.
2. 0,08. Pasek ma długość tylko jeden, więc prędzej czy później zgicie złapie jedną rodzynekę, a drugiej nie.
3. $1 - 1,4 \times 0,25 + 0 = 0,65$ i $0,3 \times 0,5 = 0,15$: punkt (0,65, 0,15).
4. $2 \times 0,027 = 0,054$. Nie: plama jest ściskana w jedną stronę i rozciągana w drugą, a jej punkty wciąż się poruszają.
5. Atraktor to zbiór, na którym ruch osiada i którego nigdy nie opuszcza; jego basen to każdy start, który na nim kończy.
6. Gładka krzywa coraz bardziej przypomina prostą. Każda linia atraktora rozpada się na wiązkę linii, raz za razem.
7. Zachowuje tylko punkty, w których droga przepływu przecina wybraną powierzchnię; reguła od jednego przecięcia do następnego jest odwzorowaniem. Reguła o trzech liczbach dotyczy przepływów, a odwzorowanie przeskakuje, więc może składać.

Rozdział 11

1. Oba mogą mieć rację, jeśli mierzyły różnymi linijkami (na mapach w różnej skali): krótsza linijka obchodzi więcej zatok. Każdy powinien był podać, jakiej linijki albo jakiej skali mapy użył.
2. $\left(\frac{4}{3}\right)^3 = \frac{64}{27}$, około 2,37.
3. $\frac{3}{5}$ długości. Trzy kopie, każda pomniejszona pięciokrotnie, dają $D = \ln 3 / \ln 5$, około 0,68.
4. $2^3 = 8$, więc $D = 3$: pełny sześciąt.
5. Około 1,5, bo $2^{1,5} = 2 \times \sqrt{2}$, czyli około 2,83.
6. Za mało punktów zostawia część najmniejszych pudełek nieodwiedzonych, co je zaniża i spłaszcza dopasowaną prostą. Więcej punktów je wypełnia.
7. W jakim zakresie rozmiarów i jak zmierzony. Prawdziwy obiekt może być fraktalem tylko między pewnym największym a najmniejszym rozmiarem.

Rozdział 12

1. $3 + 2i$ oraz 2 : wymnożenie daje $1 - i + i - i \times i = 1 + 1$.
2. $1 + i$ ma kąt 45 stopni, więc jego kwadrat ma kąt 90 i długość 2 (kwadrat długości $1 + i$): to $2i$. Po wymnożeniu $1 + 2i + i \times i = 2i$.
3. W kroku 2: orbita to 0, 2, 6, a 2 nie jest większe od 2, natomiast 6 jest.
4. Tak. Należy do niego każde rzeczywiste c od -2 do $\frac{1}{4}$; orbita 0 pozostaje między -2 a 2.
5. Tak. Jego długość to pierwiastek z $0,36 + 0,81$, trochę więcej niż 1, a podnoszenie do kwadratu stale powiększa długość większą od 1.
6. $c = 1,6 - 2,56 = -0,96$, wewnątrz dużego bąbla, który rozciąga się od $-0,75$ do $-1,25$: cykl długości dwa, tak jak w rozdziale 5 przy tym r .
7. Jego serce i bąble to wartości c , przy których orbita 0 układa się w cykl. Wartości chaotyczne leżą na cienkim brzegu, który na osi rzeczywistej jest igłą między jej oknami.

Rozdział 13

1. Ustalenie energii dwuliczbowego stanu pojedynczego wahadła zostawia jeden

kierunek, wokół zamkniętej pętli. Ustalenie jej dla czterech liczb wahadła podwójnego zostawia trzy, a to wystarcza do chaosu.

2. Ciężarki są metr i dwa metry wyżej, więc $9,81 \times (1 + 2)$, czyli około 29,4 dżula.
3. Błąd kroków jest za duży, by ufać wynikowi. Skracaj krok, aż energia się utrzyma, a potem sprawdź, czy twoja odpowiedź przetrwa kolejne skrócenie kroku o połowę.
4. Że rozejście wywołały przynajmniej częściowo błędy kroków, więc żadnej z tych liczb nie można ufać. Rozejście, które jest fizyką, nie przesuwają się, gdy zmienia się krok.
5. Jego energia, która nigdy się nie zmienia i decyduje, które stany w ogóle da się osiągnąć.
6. Dwa razy przez e : $2,7 \times 2,7$, nieco ponad siedem.
7. Nie. Ich położenie nie da się przewidzieć dalej niż na kilkadziesiąt milionów lat, ale obliczenia sięgające miliardów lat w niemal każdym przebiegu pokazują, że ich orbity zachowują mniej więcej te same rozmiary i kształty.

Rozdział 14

1. To najlepsze oszacowanie obecnego stanu, zbudowane z pomiarów, które nigdy nie są dokładne. Jej błąd rośnie, jak w każdym układzie chaotycznym, aż prognoza zapomni start.
2. 24 procent: 12 z 50.
3. Nie. Po wiarygodnych 80 procentach deszcz pada mniej więcej 80 razy na 100; te prognozy były zbyt pewne siebie.
4. Jest wtedy jeszcze jednym dniem pogody, niezwiązanym z prawdą, a dwa niezwiązane dni różnią się o mniej więcej $\sqrt{2}$ typowego wahania, podczas gdy średnia myli się tylko o jedno wahanie.
5. O dwa razy 1,35, czyli około 2,7 jednostki czasu: każda dziesięciokrotna poprawa kupuje ten sam dodatkowy czas.
6. Pierwsze to pytanie o pogodę, daleko za horyzontem, więc najlepszą odpowiedzią są szanse z klimatu. Drugie to pytanie o klimat, o statystyki przy podanym wymuszeniu, i chaos go nie wyklucza.

Rozdział 15

1. Zachowuje każdą wartość, więc i histogram. Niszczy kolejność, a to w niej widać by było regułę łączącą każdą wartość z następną.
2. Że każdą wartość wyznacza poprzednia: zrobiła go reguła jednej liczby. Szum rozrzuciłby punkty po kwadracie.
3. Pierwszy: krzywa przechodzi tylko przez ograniczoną liczbę kratek, ile by punktów na niej nie wylądowało. Drugi wciąż wypełnia kwadrat, jak szum.
4. Po około ośmiu: $2^8 = 256$, a $0,001 \times 256$ to około 0,26.
5. Oba testy są przy tak małej liczbie zawodne: w pomiarze z tego rozdziału test kratek na dziesięciu zaszumionych wartościach miał rację rzadziej niż rzucona moneta.
6. Jej lata przechodzą od jednakowych, przez naprzemienne i powtarzające się co cztery lata, do niepowtarzających się wcale; a jej chaotyczne dołki są tak głębokie, że prawdziwa choroba by wygasła.
7. Nie: szum też jest nieregularny. Dowodem byłaby prognoza, która wygrywa z surogatami, z błędem, który zaczyna od małego i rośnie równomiernie, w długim, czystym zapisie.

Rozdział 16

1. Ani 0,1, ani 0,2 nie ma dokładnej postaci dwójkowej, więc każda jest zapamiętana jako najbliższa liczba podwójnej precyzji, a suma jest jeszcze raz zaokrąglana.
2. W kroku 54, jeden krok na cyfrę dwójkową. Prawdziwa orbita idzie przez $\frac{3}{5}$, $\frac{1}{5}$, $\frac{2}{5}$, $\frac{4}{5}$, $\frac{3}{5}$ i krąży w tej pętli bez końca.
3. Przez około dwadzieścia (zmierzone: 23): odwzorowanie podwaja błąd zaokrąglania w każdym kroku, zużywając jedną cyfrę dwójkową na krok.
4. Około 732 924: liczba cyfr podwaja się w każdym kroku, a dziesięć podwojeń mnoży ją mniej więcej przez tysiąc.
5. Ułamkowi czasu. Droga podąża za inną orbitą mniej więcej od pięćdziesiątego kroku, ale trzyma się blisko prawdziwej orbity z pobliskiego startu, a ta spędza czas tak, jak prawdziwe orbity.
6. Maszyna mieści tylko skończenie wiele liczb, więc orbita musi wrócić do jakiejś wartości, a reguła powtarza wtedy wszystko, co po niej następuje.

Rozdział 17

1. Mniej więcej 0,714, z nachyleniem $-1,5$. Chybiecie mnoży się przez 1,5 i w każdym kroku zmienia stronę, więc orbita nie może pozostać.
2. Mniej więcej 0,0038 powyżej niego. Obniż r mniej więcej o 0,0038/0,19, czyli o 0,02, na ten jeden krok.
3. Bo sterownik musi przewidzieć tylko jeden krok naprzód, a w ciągu jednego kroku chybiecie rośnie mniej więcej dwukrotnie, a nie wykładniczo bez końca.
4. Zasięg maleje o połowę, więc cel jest o połowę węższy, a średnie czekanie mniej więcej się podwaja.
5. Bo spokojne odwzorowanie ustala się w swoim własnym punkcie stałym, $1 - 1/2,9$, mniej więcej 0,655. Wartość 0,744 jest punktem stałym tylko przy $r = 3,9$, gdzie jest niestabilna.
6. x oryginału. Nie: kopia zgadza się z oryginałem w każdej chwili, a to jest synchronizacja, nie przewidywanie.
7. Dowolne dwa spośród: lasery, układy elektroniczne i wyizolowana tkanka serca w laboratorium. Nie jest to rutynowa technika kliniczna ani przemysłowa.

Rozdział 18

1. Błąd maleje, więc trajektorię da się prognozować tak daleko, jak długo sama reguła jest poprawna.
2. Mniej więcej 6,908/0,5 $\approx$ 13,8 dnia.
3. Mniej więcej o 2,303/0,5 $\approx$ 4,6 dnia dłużej: stały dodatkowy odcinek, nie dziesięć razy dłużej.
4. Nie. To poprawna prognoza: ułamek czasu, który odwzorowanie tam spędza, taki sam z każdego stanu początkowego.
5. Czy każda wartość naniesiona względem następnej rysuje krzywą, a nie chmurę; czy da się go prognozować krok lub dwa naprzód z podobnych chwil z przeszłości; i czy zapis jest dość długi, by to rozstrzygnąć.
6. Układ jest wrażliwy na małe zmiany: drobny błąd rośnie, i tak samo rośnie skutek drobnego, dobrze wymierzonego pchnięcia.
7. Horyzont to $\ln(T/e_0)$ podzielone przez wykładnik. Wykładnik Układu Słonecznego to mniej więcej jedna piąta na milion lat, co daje horyzont około stu milionów lat;

błędy atmosfery rosną tak samo w ciągu dni. Lepsze dane dodają do każdego z nich stały odcinek.

Błędne odczytania chaosu

Każde błędne odczytanie, które książka stara się poprawić, w kolejności, w jakiej wywołują je rozdziały. Każde jest zapisane głosem kogoś, kto w nie wierzy, a po nim następuje to, co jest prawdą, i podrozdział, w którym książka pozwala ci najpierw w nie wpaść. Jeśli trafiłeś tu przed przeczytaniem tego podrozdziału, wpis nabierze sensu później: każda pułapka miała na ciebie zadziałać.

Rozdział 1 *Model to maszyna do przewidywania*

- 11 „**Model to kopia rzeczywistości.**” Model to maszyna do odpowiadania na pytania: liczby na wejściu, liczby na wyjściu, a jedyny uczciwy test to zgodność jego odpowiedzi z tym, co potem zmierzysz. § 1.1
- 12 „**Model, który jest błędny, jest bezużyteczny.**” Każdy model jest błędny, bo coś pomija. Użyteczne pytanie brzmi: czy jest wystarczająco dobry do tego pytania, w tym zakresie. § 1.4

Rozdział 2 *Reguła następnego kroku*

- 21 „**Wzrost wykładniczy to szybki wzrost.**” To wzrost proporcjonalny do tego, co już jest, a mnożnik może być bliski jedności: przy 1 procencie rocznie suma podwaja się po mniej więcej 70 latach. Wzrost wykładniczy gwarantuje nie szybkość, tylko to, że w końcu prześcignie każdy wzrost, który dodaje stałą kwotę. § 2.3
- 22 „**Punkt stały to miejsce, w którym układ w końcu się znajdzie.**” To miejsce, w którym układ pozostaje, jeśli jest w nim dokładnie. Tylko stabilny punkt stały, w którym każdy krok zmniejsza odległość ($|a| < 1$ dla $x \mapsto ax + b$), przyciąga pobliskie stany; niestabilny, jak saldo kredytu, przy którym rata pokrywa odsetki, je odpycha. § 2.5

Rozdział 3 *Sprzężenie zwrotne: dlaczego nic nie rośnie wiecznie*

- 31 „**Skutek dwóch pchnięć to suma ich skutków.**” Tylko dla reguły prostoliniowej. Na krzywej, a taką robi z reguły sprzężenie zwrotne, skutek pchnięcia zależy od tego, gdzie układ już się znajduje: dwa pchnięcia mogą razem zrobić znacznie więcej, znacznie mniej albo coś przeciwnego niż ich suma. § 3.4
- 32 „**Granica wzrostu oznacza, że wzrost łagodnie się zatrzymuje.**” Przy łagodnym tempie wzrostu tak. Przy silnym, albo gdy między przyczyną a skutkiem jest opóźnienie, populacja przestrzeliwuje swój poziom i oscyluje, a po dalszym podkręceniu wahania wcale nie muszą wygasać. § 3.6

Rozdział 4 *Ruch: reguły zmiany i mapa wszystkich stanów*

- 41 „**Mniejszy krok sprawia, że symulacja jest poprawna.**” Sprawia, że jest poprawna dłużej. Błąd metody krokowej powstaje przy każdym kroku i się piętrzy:

Euler z krokami dziesięć razy mniejszymi zyskuje na sprężynie tyle samo energii, tylko dziesięć razy później. § 4.2

42 „Wahadło to prosty układ, więc jego ruch musi być prosty.” Jedno wahadło ma dwie liczby stanu, a gładka reguła na płaszczyźnie może się tylko zatrzymać, krążyć albo uciec. Zawieś na nim drugie, a stan ma cztery liczby i ruch może być chaotyczny. § 4.5

43 „Wahnięcie wahadła trwa tyle samo, niezależnie od wychylenia.” Tylko przy małych wychyleniach. Wahadło o długości metra puszczane z 90 stopni wychyla się o około 18 procent dłużej niż puszczane z 10, a blisko położenia do góry nogami czas wahnięcia rośnie bez ograniczeń. § 4.4

Rozdział 5 *Jedna linijka algebry, każde możliwe zachowanie*

51 „Prosta reguła daje proste zachowanie.” Jedna linijka arytmetyki, $x_{n+1} = r x_n (1 - x_n)$, daje stan ustalony, cykle i orbity, które nigdy się nie powtarzają, gdy kręci się jedynym pokrętkiem r . To, jak skomplikowane wydaje się zachowanie, mało mówi o tym, jak skomplikowana jest reguła. § 5.2

52 „Skoro nie powtórzyło się po tysiącu kroków, to zaraz się powtórzy.” Przy $r = 3,9$ żaden cykl nie przyciąga orbity: w stu tysiącach kroków odwzorowania logistycznego nie wraca ani jedna wartość, a dłuższy przebieg daje tylko więcej wartości, których wcześniej nie było. § 5.5

53 „Nieregularny wynik wymaga przypadkowej albo skomplikowanej przyczyny.” Prosta reguła deterministyczna bez żadnej losowości może dawać wynik równie nieregularny jak przypadek. Sam wynik nie mówi, co go wytworzyło; to była przestroga, którą May w 1976 roku skierował do ekologów. § 5.6

Rozdział 6 *Efekt motyla, zmierzony*

61 „Jeśli uruchomię model chaotyczny jeszcze raz, dostanę coś innego.” Nie z tego samego punktu startowego: reguła chaotyczna daje z tego samego punktu tę samą przyszłość, co do ostatniej cyfry. Przyszłość zmienia zmieniony punkt startowy, choćby najmniej. § 6.5

62 „Motyl wywołuje tornado.” Pytanie Lorenza dotyczyło przewidywania, nie siły: drobne zaburzenie zmienia to, jaka pogoda nastąpi, a skoro każde inne zaburzenie robi to samo, żadne z nich z osobna nie jest przyczyną. § 6.5

63 „Chaos oznacza losowość.” Układ chaotyczny w każdym kroku dokładnie trzyma się swojej reguły. To, co wygląda na losowość, to reguła odczytująca cyfry punktu startowego, których nikt nie zna. § 6.6

Rozdział 7 *Liczba na chaos: wykładnik Lapunowa*

71 „Żeby poznać typowe rozciągnięcie na krok, uśrednij rozciągnięcia.” Wzrost się składa, więc typowy czynnik to ten, który użyty w każdym kroku daje ten sam iloczyn. Uśredniaj logarytmy rozciągnięć, nie same rozciągnięcia: krok $\times 4$ i krok $\times \frac{1}{4}$ zostawiają odstęp tam, gdzie był. § 7.1

72 „Dziesięć razy lepsze dane dają dziesięć razy dłuższe prognozy.” Horyzont to mniej więcej $\ln(T/\delta)/\lambda$, więc dziesięciokrotnie lepszy stan początkowy daje stałe $\ln 10/\lambda$ dodatkowych kroków. Dla odwzorowania logistycznego przy $r = 4$

tysiąckrotna poprawa daje mniej więcej dziesięć kroków, bez względu na wcześniejszy błąd. § 7.4

- 73** „**Większy komputer w końcu uczyni każdy układ przewidywalnym.**” Szybszy komputer nie mierzy stanu początkowego ani trochę lepiej, a horyzont zależy od błędu stanu początkowego tylko przez jego logarytm. Podwojenie horyzontu wymaga dwa razy więcej poprawnych cyfr; tysiąc kroków odwzorowania logistycznego przy $r = 4$ wymagałoby stanu początkowego z dokładnością do mniej więcej trzystu miejsc po przecinku. § 7.4

Rozdział 8 *Droga do chaosu i liczba, której nikt się nie spodziewał*

- 81** „**Przejdźcie od porządku do chaosu to nagle przełączenie.**” Na drodze odwzorowania logistycznego porządek rozpada się w nieskończonej kaskadzie podwojeń okresu, z których każde przychodzi szybciej niż poprzednie; chaos zaczyna się tam, gdzie kaskada się kończy, przy $r \approx 3,5699$. § 8.1
- 82** „**Inne równanie daje inną drogę do chaosu.**” Każda reguła z jednym gładkim garbem idzie tą samą drogą: jej podwojenia leżą przy innych wartościach pokręta, ale odstęp między nimi maleje w tym samym stosunku, stałej Feigenbauma $\delta \approx 4,6692$. § 8.4
- 83** „**Gdy układ raz osiągnie chaos, jest chaotyczny wszędzie dalej.**” Za kaskadą odwzorowanie logistyczne wraca do porządku nieskończenie wiele razy, w oknach o każdym okresie; najszersze, okresu trzy, otwiera się przy $r = 1 + \sqrt{8}$. § 8.5

Rozdział 9 *Trzy równania na pogodę: atraktor Lorenza*

- 91** „**Chaos potrzebuje wielu zmiennych.**” Model Lorenza ma trzy liczby i dwa mnożenia, i jest chaotyczny. Trzy to najmniej, ile potrzebuje przepływ, bo droga na płaszczyźnie nie może przeciąć samej siebie; to też wystarcza. § 9.2
- 92** „**Układ chaotyczny może pójść gdziekolwiek.**” Nieprzewidywalny to nie to samo co nieograniczony. Każdy przebieg Lorenza, z niemal każdego startu, zostaje ściągnięty na ten sam ograniczony zbiór w kształcie motyla, atraktor, i nigdy go nie opuszcza. § 9.4
- 93** „**Jeśli droga jest nieprzewidywalna, nic w niej nie da się przewidzieć.**” Droga jest nieprzewidywalna dalej niż na mniej więcej dwadzieścia jednostek czasu, ale statystyki atraktora – część czasu na każdym skrzydle, średnia wysokość, a nawet wykładnik Lapunowa – wychodzą takie same z każdego startu. § 9.5

Rozdział 10 *Rozciągnij i złóż: dziwne atraktory*

- 101** „**Jeśli układ nigdy nie opuszcza małego obszaru, nie może być bardzo wrażliwy na stan początkowy.**” Obszar ogranicza to, jak *duży* może być błąd, a nie to, jak szybko rośnie. Małe błędy podwaja się raz za razem, aż dorówna rozmiarom obszaru, a od tej chwili dwa przebiegi są tak niezależne jak dwa punkty wrzucone do niego na chybił trafił. § 10.1
- 102** „**Skoro pole wciąż maleje, ruch musi zamierać.**” Odwzorowanie Hénona mnoży każde pole przez 0,3 w każdym kroku, bo mocno ściska w jednym kierunku, a rozciąga w innym. Plama punktów staje się coraz dłuższą i cieńszą wstęgą, której punkty poruszają się bez końca. § 10.4

- 103** „Wystarczy dość mocno powiększyć dowolną krzywą, a stanie się linią prostą.” To prawda dla krzywych gładkich. Każda linia atraktora dziwnego rozpada się w powiększeniu na wiązkę linii, a każda z nich znowu: warstwy w warstwach w każdej skali. § 10.5

Rozdział 11 *Fraktale: mierzenie czegoś, co nie ma długości*

- 111** „Wybrzeże ma długość, tylko trudno ją zmierzyć.” Długość to wartość, na której ustalają się pomiary, gdy linijka się skraca, a pomiary wybrzeża się nie ustalają: pomiary krzywej Kocha rosną $\frac{4}{3}$ razy przy każdym trzykrotnym skróceniu linijki. Długość wybrzeża to fakt o wybrzeżu i o linijce naraz. § 11.2
- 112** „Wymiar musi być liczbą całkowitą.” Liczony z kopii albo z pudełek wymiar jest pomiarem i nie musi być całkowity: dla krzywej Kocha wynosi $\ln 4 / \ln 3$, między jeden a dwa, a dla zbioru Cantora $\ln 2 / \ln 3$, między zero a jeden. Liczby całkowite należą do kształtów gładkich. § 11.3
- 113** „Fraktale to ozdoba: ładne obrazki, żadnego pożytku.” Fraktalne rozgałęzienia w płucach, naczyniach krwionośnych i rzekach wykonują pracę: mieszczą dużą powierzchnię albo długi brzeg w ograniczonej przestrzeni. Prawdziwe obiekty są fraktalami tylko w pewnym zakresie rozmiarów, a podany wymiar powinien mówić w jakim. § 11.6

Rozdział 12 *Iteracja na płaszczyźnie: Julia i Mandelbrot*

- 121** „Liczby zespolone są urojone, więc nie opisują niczego rzeczywistego.” Ta nazwa to historia, nie opis. Liczba zespolona to punkt płaszczyzny z regułą mnożenia, która rozciąga i obraca, więc wszystko, co się rozciąga i obraca – falę, obrót, prąd przemienny – opisują liczby zespolone. § 12.1
- 122** „Nieskończenie wiele szczegółów to nieskończenie wiele informacji.” Każdy szczegół brzegu zbioru Mandelbrota wynika z jednej linijki algebry, $z \rightarrow z^2 + c$, i jednego pytania: czy orbita 0 ucieka? Szczegóły są wytwarzane, a nie przechowywane, a krótki opis może mieć nieskończone konsekwencje. § 12.3
- 123** „Zbiór Mandelbrota to obraz chaosu.” Większość tego, co czarne, to porządek: w sercu orbita 0 ustala się w jednym punkcie, a w każdym bąblu w cyklu, którego długość ten bąbel wyznacza. Nieustalające się orbity odwzorowania logistycznego leżą na cienkim brzegu, który na osi rzeczywistej jest igłą między jej oknami. § 12.5

Rozdział 13 *Wahadła i planety*

- 131** „Wahadło to prosta maszyna, więc jego ruch musi być prosty.” Prawda dla jednego wahadła, którego stan to dwie liczby. Zawieś na nim drugie, a stan to cztery liczby, i puszczane z wysoka wahadło nie układa się w żaden powtarzający się wzór. § 13.1
- 132** „Układ, który zachowuje energię, bez tarcia i bez niczego losowego, nie może być chaotyczny.” Ustalenie energii czterech liczb stanu wciąż zostawia trzy kierunki ruchu, a to wystarcza do chaosu. Dwa wahadła podwójne różniące się o milionową część radiana rozchodzą się w ciągu sekund, choć energia każdego jest stała z dokładnością do około jednej części na 10^8 . § 13.3

- 133 „Układ Słoneczny to doskonały zegar.”** W skali tysięcy lat jest zegarem znakomitym. Laskar stwierdził, że planety wewnętrzne są chaotyczne, z czasem Lapunowa około pięciu milionów lat, więc nie da się przewidzieć, gdzie na orbicie będzie Ziemia za sto milionów lat. § 13.6
- 134 „Chaotyczny Układ Słoneczny to układ niestabilny: planety się rozleca.”** Nieprzewidywalny to nie niestabilny. Obliczenia sięgające miliardów lat w niemal każdym przebiegu pokazują planety na orbitach o mniej więcej tych samych rozmiarach i kształtach; nie da się przewidzieć tylko, gdzie na tych orbitach będą. § 13.6

Rozdział 14 *Pogoda, klimat i prognoza zespołowa*

- 141 „Prognoza to jedna liczba.”** Prognoza to liczba i informacja, na ile jej ufać. Ta sama średnia zespołu zasługuje na zaufanie w dniu, w którym członkowie się zgadzają, i na wzruszenie ramion w dniu, w którym się rozchodzą, a rozrzut zespołu nadąża za wielkością błędu, choć nikt nie widzi prawdy. § 14.2
- 142 „Prognoza 30 procent szansy na deszcz, po której spadł deszcz, była błędna.”** Jeden dzień nie obali prawdopodobieństwa. Prognoza 30 procent jest trafna, jeśli pada w mniej więcej 30 procentach dni, w które ją wydano; to sprawdzenie, wiarygodność, wymaga wielu prognoz, a dobry zespół je przechodzi. § 14.3
- 143 „Skoro nie da się przewidzieć pogody, nie da się też przewidzieć klimatu.”** Pogoda to trajektoria, a klimat to statystyka atraktora. Dwa przebiegi, które nie zgadzają się co do żadnego dnia, zgadzają się co do średniej, wahaniami i tego, jak często pada; zmiana wymuszenia przesuwają te statystyki i właśnie o to pyta projekcja klimatyczna. § 14.5

Rozdział 15 *Życie, rynki i szum: jak odróżnić chaos od przypadku*

- 151 „Ciąg, który wygląda na losowy, musi pochodzić z losowości.”** Odwzorowanie logistyczne przy $r = 4$ wygląda na równie losowe jak przetasowanie jego własnych wartości, a nie ma w nim ani grama przypadku. Losowy wygląd to fakt o wyglądzie; o przyczynie mówi dopiero test porządku. § 15.1
- 152 „Nieregularne dane dowodzą chaosu.”** Nieregularne są i chaos, i szum. Twierdzenie o chaosie wymaga dowodów na regułę: struktury w odwzorowaniu powrotu, prognozy dobrej na początku i pogarszającej się równomiernie oraz zapisu dość długiego i czystego, by jedno i drugie coś znaczyło. § 15.5
- 153 „Chaos na rynkach oznacza, że da się je przewidzieć.”** Chaos da się prognozować tylko na kilka kroków naprzód i wcale poza horyzontem, więc chaotyczne ceny obiecywałyby krótki horyzont z twardą granicą, a nie szklaną kulę; a dowody na chaos o niewielu wymiarach w cenach są słabe i sporne. § 15.6
- 154 „Dobra prognoza o krok naprzód dowodzi chaosu.”** Szereg z pamięcią, jak dobową temperaturę, daje się prognozować na dzień naprzód bez żadnego chaosu. Podpisem chaosu jest prognoza dobra o krok naprzód, pogarszająca się w stałym tempie i wygrywająca z surogatami, które zachowują powolne wzrosty i spadki szeregu. § 15.6

Rozdział 16 *Chaos wewnątrz komputera*

- 161** „**Program deterministyczny zawsze daje ten sam wynik.**” Na tej samej maszynie, z tym samym kompilatorem i bibliotekami, tak. Zmień którąkolwiek z tych rzeczy albo kolejność sumowania, a zmieniają się zaokrąglenia; chaotyczne obliczenie przenosi różnicę na ostatniej cyfrze na pierwszą w mniej więcej pięćdziesiąt kroków. § 16.3
- 162** „**Więcej miejsc po przecinku czyni symulację dokładną.**” Każda dodatkowa cyfra kupuje stałą liczbę poprawnych kroków, nigdy wszystkie: przy $r = 4$ odwzorowanie logistyczne zużywa jedną cyfrę dwójkową na krok, a dokładne ułamki podwajają liczbę cyfr w każdym kroku. § 16.4
- 163** „**Chaotyczne odwzorowanie to dobry generator liczb losowych.**” Chaos daje nieprzewidywalność i nic więcej z tego, czego potrzebuje generator: wartości odwzorowania logistycznego gromadzą się przy 0 i 1, każda wyznacza następną, a w komputerze wpada ono w krótką pętlę. Dobre generatory projektuje się za pomocą teorii liczb tak, by miały jedną ogromną pętlę. § 16.6

Rozdział 17 *Okiełznać chaos: sterowanie i synchronizacja*

- 171** „**Chaos nie da się sterować.**” Chaos zabrania przewidywania na daleką metę, a sterownik musi przewidzieć tylko jeden krok: czeka, aż orbita przejdzie blisko niestabilnej orbity, a potem znosi każde nadchodzące chybiecie pchnięciem parametru. Odwzorowanie logistyczne przy $r = 3,9$ daje się utrzymać w niestabilnym punkcie stałym pchnięciami mniejszymi niż jeden procent. § 17.2
- 172** „**Sterować układem to go przemóc.**” Sterownik nie dostarcza żadnego ruchu: utrzymuje jedno z własnych niestabilnych zachowań układu, znosząc małe chybiecia, a gdy orbita zostanie złapana, pchnięcia maleją prawie do zera. Siła, czyli duża i trwała zmiana parametru, nie daje nawet tego samego stanu. § 17.3
- 173** „**Dwa układy chaotyczne nigdy nie mogą się zgadzać.**” Zostawione same sobie nigdy nie pozostaną razem. Sprzężone tak, że jeden słucha drugiego, mogą zgadzać się dokładnie: kopia układu Lorenza, która dostaje x oryginału, zgrywa się z nim. Ta zgodność dotyczy teraźniejszości, a nie przewidywania przyszłości. § 17.4

Rozdział 18 *Po co ci chaos: przewodnik terenowy po przewidywaniu*

- 181** „**Nieprzewidywalne znaczy niepoznawalne.**” Za horyzontem trajektoria przepada, a prawie wszystko inne jest znane: dokąd układ może dojść, jak często odwiedza każdy obszar i jakie są szanse. Prognoza podana jako szanse to poprawna odpowiedź, nie wykręt. § 18.3
- 182** „**Małe przyczyny zawsze mają znaczenie.**” W układzie stabilnym wygasają. W chaotycznym mała przyczyna zmienia szczegóły za horyzontem i nic w statystykach, a żadnej pojedynczej małej przyczyny nie da się po fakcie obwinić o duży skutek. § 18.4
- 183** „**Chaos znaczy: wszystko wolno.**” Chaos to dokładna reguła zamknięta w ograniczonym atraktorze, ze stałymi statystykami i horyzontem, który da się policzyć. Nazwanie czegoś chaotycznym to twierdzenie do sprawdzenia, a nie powód, żeby przestać prognozować. § 18.7

Matematyka, której potrzebujesz, na czterech stronach

Każde narzędzie, którego używa książka, w jednym miejscu, z rozdziałem, który je wprowadza. Jeśli w środku rozdziału zatrzyma cię jakiś symbol, znajdziesz go tutaj.

C.1 Bardzo duże i bardzo małe liczby

Liczbę taką jak 0,00000000015 trudno odczytać, więc zapisuje się ją jako $1,5 \times 10^{-10}$: cyfry, które mają znaczenie, i o ile miejsc przesunąć przecinek. 10^3 to tysiąc, 10^6 milion, a potęga ujemna przesunęła w drugą stronę, więc 10^{-3} to jedna tysięczna. Kiedy książka mówi, że dwie liczby różnią się „na dziesiątym miejscu po przecinku”, ma na myśli różnicę rzędu 10^{-10} . Książka zaokrągliła to, co drukuje, i mówi, ile cyfr zostawia: wynik jest wart tylko tylu cyfr, ile może potwierdzić stojący za nim pomiar (rozdział 1).

C.2 Reguły, kroki i orbity

Regułę, która zamienia teraźniejszość w następną chwilę, zapisuje się tak:

$$x_{n+1} = f(x_n),$$

co czytamy „następne x to f od tego x ”. Małe n pod linią liczy kroki: x_0 to początek, x_1 stan po jednym kroku i tak dalej. Całą listę $x_0, x_1, x_2, \dots$ nazywamy *orbitą*. *Punkt stały* to stan, który reguła zostawia na miejscu, czyli rozwiązanie równania $x = f(x)$ (rozdział 2).

C.3 Potęgi, wzrost i połowienie

Wielokrotne mnożenie przez ten sam czynnik a daje potęgi: $a \times a \times a = a^3$, a n kroków mnożenia przez a mnoży przez a^n . Jeśli a jest większe od 1, wynik rośnie coraz szybciej; jeśli leży między 0 a 1, maleje do zera. Dziesięć podwojeń mnoży przez $2^{10} = 1024$, niewiele ponad tysiąc, co warto zapamiętać: dziesięć podwojeń to tysiąc, dwadzieścia to milion, a trzydzieści miliard (rozdziały 2, 3 i 6).

C.4 Logarytmy

Logarytm odpowiada na pytanie „ile razy?”. Logarytm naturalny, zapisywany $\ln$, zamienia mnożenie w dodawanie:

$$\ln(a \times b) = \ln a + \ln b.$$

Jeśli więc błąd w każdym kroku mnoży się przez 2, to $\ln$ błędu rośnie w każdym kroku o $\ln 2 \approx 0,693$, czyli o stałą wielkość, a stałą wielkość można uśrednić i porównać. Liczbę, której $\ln$ wynosi 1, nazywamy $e \approx 2,718$, a $\ln 10 \approx 2,303$.

Na pytanie „ile podwojeń mnoży przez N ?” odpowiedź brzmi $\ln N / \ln 2$. Pomnożenie przez tysiąc wymaga około 9,97 podwojeń, a jedna dodatkowa cyfra dziesiętna dokładności jest warta około 3,32 podwojenia. Ten jeden fakt stoi za horyzontem prognozy z rozdziału 7 i za całym wywodem książki o przewidywaniu: za dokładność płaci się cyframi, a każda cyfra kupuje tylko stałą liczbę kroków.

C.5 Nachylenie krzywej

Jak stroma jest krzywa w danym punkcie? Przesuń wejście o małe h w obie strony, zobacz, o ile zmienia się wyjście, i podziel:

$$\text{nachylenie } f \text{ w } x \approx \frac{f(x+h) - f(x-h)}{2h}.$$

Dla $f(x) = x^3$ w punkcie $x = 2$ dokładne nachylenie wynosi 12, a przy $h = 0,01$ przepis daje 12,0001. Nachylenie mówi, jak bardzo błąd wejścia zostaje powiększony na wyjściu: nachylenie 3 go potraja, nachylenie $-\frac{1}{2}$ go połowi i odwraca znak. Wielkość liczby bez jej znaku zapisuje się między kreskami, $|-3| = 3$ (rozdział 2).

C.6 Tempo zmian

Kiedy czas płynie gładko, a nie krokami, reguła mówi, jak szybko każda wielkość zmienia się w tej chwili. Zapisuje się to tak:

$$\frac{dx}{dt} = (\text{jakiś wzór}),$$

co czytamy „tempo zmiany x w czasie”. W tej książce nigdy nie trzeba rozwiązywać takiej reguły algebrą: komputer podąża za nią w wielu małych krokach czasu, a rozdział 4 pokazuje, jak robić to dobrze.

C.7 Sumy i średnie

Długą sumę zapisuje się wielką grecką literą sigma:

$$\sum_{n=1}^N a_n = a_1 + a_2 + \cdots + a_N,$$

a średnia N liczb to ta suma podzielona przez N . Wykładnik Lapunowa z rozdziału 7 jest właśnie taką średnią: średnią z $\ln$ tego, jak bardzo każdy krok rozciąga mały błąd.

C.8 Reszty

„ $x \bmod 1$ ” znaczy „odrzuć część całkowitą i zostaw to, co po przecinku”, więc $2,75 \bmod 1 = 0,75$. Rozdział 6 używa tego, żeby trzymać liczbę między 0 a 1.

C.9 Liczby zespolone

Liczba zespolona to punkt na płaszczyźnie, zapisywany $a + bi$: a kroków w bok i b kroków w górę. Dodawanie dwóch takich liczb dodaje kroki. Mnożenie korzysta ze zwykłej algebry i jednej nowej reguły, $i \times i = -1$, więc

$$(1 + 2i)(3 + 4i) = 3 + 4i + 6i + 8i^2 = -5 + 10i.$$

Odległość $a + bi$ od środka płaszczyzny wynosi $\sqrt{a^2 + b^2}$ (rozdział 12).

C.10 Litery greckie

Książka używa kilku, zawsze do tego samego:

Litera	Nazwa	Co oznacza w tej książce
λ	lambda	wykładnik Lapunowa (rozdział 7)
δ	delta	stała Feigenbauma (rozdział 8)
σ, ρ, β	sigma, rho, beta	trzy ustawienia Lorenza (rozdział 9)
ε	epsilon	mały błąd albo małe pchnięcie
Σ	wielka sigma	suma

Dalsza lektura

Dwie listy: prace, w których idee tej książki pojawiły się po raz pierwszy, i książki, które warto przeczytać potem. Prace służą do sprawdzenia, skąd coś się wzięło; większość wymaga więcej matematyki, niż zakłada ta książka, ale ich pierwsze strony są często jaśniejsze niż wszystko, co o nich później napisano. Książki służą do pójścia dalej, a przy każdej jest uwaga, do czego się nadaje i czego od ciebie wymaga. Kilka z nich ukazało się także po polsku; tytuły przekładów łatwo znaleźć w katalogu biblioteki.

D.1 Gdzie idee pojawiły się po raz pierwszy

Pierre-Simon Laplace, *Essai philosophique sur les probabilités*, 1814. Wizja wszechświata jak mechanizmu zegara w swojej najbardziej pewnej sobie postaci; przywołana w rozdziale 1.

Henri Poincaré, *Science et méthode*, 1908. Pierwsze jasne stwierdzenie, słowami, że małe różnice na początku mogą dać bardzo duże na końcu i że przewidywanie staje się wtedy niemożliwe.

Edward N. Lorenz, „Deterministic nonperiodic flow”, *Journal of the Atmospheric Sciences* 20 (1963), 130–141. Trzy równania z rozdziału 9 i praca, od której zaczęła się współczesna dziedzina.

Tien-Yien Li i James A. Yorke, „Period three implies chaos”, *American Mathematical Monthly* 82 (1975), 985–992. Praca, która dała dziedzinie nazwę (rozdział 8).

Robert M. May, „Simple mathematical models with very complicated dynamics”, *Nature* 261 (1976), 459–467. Odzworowanie logistyczne jako przestroga dla każdego, kto buduje modele (rozdział 5).

Michel Hénon, „A two-dimensional mapping with a strange attractor”, *Communications in Mathematical Physics* 50 (1976), 69–77. Odzworowanie z rozdziału 10.

Mitchell J. Feigenbaum, „Quantitative universality for a class of nonlinear transformations”, *Journal of Statistical Physics* 19 (1978), 25–52. Stała z rozdziału 8.

Benoit B. Mandelbrot, „How long is the coast of Britain? Statistical self-similarity and fractional dimension”, *Science* 156 (1967), 636–638. Wymiar fraktalny, zanim pojawiło się słowo fraktal (rozdział 11).

Floris Takens, „Detecting strange attractors in turbulence”, w: *Dynamical Systems and Turbulence*, Lecture Notes in Mathematics 898 (1981), 366–381. Odtwarzanie kształtu układu z jednej mierzonej wielkości (rozdział 15).

- Jacques Laskar**, „A numerical experiment on the chaotic behaviour of the Solar System”, *Nature* 338 (1989), 237–238. Chaos w orbitach planet (rozdział 13).
- Edward Ott, Celso Grebogi i James A. Yorke**, „Controlling chaos”, *Physical Review Letters* 64 (1990), 1196–1199. Rozdział 17.
- Louis M. Pecora i Thomas L. Carroll**, „Synchronization in chaotic systems”, *Physical Review Letters* 64 (1990), 821–824. Rozdział 17.
- Edward N. Lorenz**, „Predictability: a problem partly solved”, *Seminar on Predictability*, European Centre for Medium-Range Weather Forecasts, 1996. Zabawkowa atmosfera z rozdziału 14.

D.2 Książki na później

- James Gleick**, *Chaos: Making a New Science*, 1987. Historia ludzi, którzy odkryli chaos, opowiedziana jako historia, a nie matematyka. Najlepsza książka do czytania równoległe z tą.
- Edward N. Lorenz**, *The Essence of Chaos*, 1993. Jeden z twórców dziedziny tłumaczy ją czytelnikowi ogólnemu, łagodnie i starannie; naturalny następny krok po częściach II i III.
- Ian Stewart**, *Does God Play Dice? The Mathematics of Chaos*, 1989. Żywa popularna opowieść matematyka, z większą ilością matematyki niż u Gleicka i mniejszą niż w podręczniku.
- Leonard A. Smith**, *Chaos: A Very Short Introduction*, 2007. Krótka i staranna, szczególnie dobra w tym, co chaos oznacza, a czego nie oznacza dla prognozowania.
- Tim Palmer**, *The Primacy of Doubt*, 2022. Prognozy zespołowe i pożytki z niepewności, opowiedziane przez meteorologa, który pomagał je zbudować; książka do przeczytania po rozdziale 14.
- Benoit B. Mandelbrot**, *The Fractal Geometry of Nature*, 1982. Książka, od której zaczęły się fraktale, rozlewna i pełna obrazów.
- Heinz-Otto Peitgen, Hartmut Jürgens i Dietmar Saupe**, *Chaos and Fractals: New Frontiers of Science*, 1992. Bardzo duża, bardzo hojna książka, która buduje wszystko z eksperymentów, jakie możesz zrobić sam; naturalna kontynuacja tej książki dla czytelnika, któremu spodobały się laboratoria.
- Steven H. Strogatz**, *Nonlinear Dynamics and Chaos*, 1994. Podręcznik do przeczytania, gdy będziesz gotów na rachunek różniczkowy: jasny, praktyczny i bardzo lubiany. To tam temat tej książki staje się wykładem uniwersyteckim.
- Kathleen T. Alligood, Tim D. Sauer i James A. Yorke**, *Chaos: An Introduction to Dynamical Systems*, 1996. Podręcznik matematyki napisany przez jednego z twórców dziedziny i jego kolegów, z dowodami.

Manifest

Co zawiera książka, policzone, a nie zadeklarowane. Każdą liczbę poniżej proces budowania oblicza z repozytorium i odmawia dokończenia pracy, jeśli książka się zmieniła, a ta strona nie.

Rejestr

Napisane rozdziały, z 18	18
Listingi, każdy uruchamiany przy budowaniu	80
Laboratoria, każde z szablonem, który nie przechodzi, i rozwiązaniem, które przechodzi	37
Testy sprawdzające laboratoria	132
Pytania „zatrzymaj się i pomyśl”	131
Pytania sprawdzianów, każde z odpowiedzią w dodatku A	119
Błędne odczytania w dodatku B	53
Wykresy	52
Diagramy	17
Wydruki wyników listingów	80
Obliczone liczby w tekście	416
Linie biblioteki chaoslab	447
Fragmenty kodu wydrukowane bez uruchomienia	0

Manifest laboratoriów

1.1	101_01_stone — Kamień w obie strony	4
1.2	101_02_cooling — Kiedy da się to wypić?	6
2.1	102_01_until — Ile kroków do celu?	18
2.2	102_02_fixed — Gdzie się zatrzyma i czy tam zostanie?	21
3.1	103_01_verdict — Odczytaj los punktu stałego z nachylenia . .	33
3.2	103_02_instalments — Późne hamulce i wczesne hamulce . .	37
4.1	104_01_rk4 — Jeden krok RK4	44
4.2	104_02_drift — Zmierz dryf	45
5.1	105_01_stability — Gdzie stan ustalony puszcza	58
5.2	105_02_period — Własny detektor cykli	62
6.1	106_01_apart — Kiedy dwa przebiegi się rozchodzą?	70
6.2	106_02_digits — Znane cyfry, przewidziane kroki	73
7.1	107_01_exponent — Wykładnik z orbity	85
7.2	107_02_horizon — Jak daleko naprzód i jakim kosztem	89
8.1	108_01_superstable — Znajdź punkt orientacyjny sam	99

8.2	108_02_ratio — Stosunek Feigenbauma i miejsce, w którym kończą się podwojenia	102
9.1	109_01_rates — Reguły Lorenza własną ręką	110
9.2	109_02_wings — Dwie liczby, o których start nie decyduje . . .	117
10.1	110_01_knead — Pasek ciasta	123
10.2	110_02_henon — Jeden krok i jego pole	129
11.1	111_01_cantor — Zbiór Cantora z reguły	140
11.2	111_02_boxes — Policz pudełka sam	142
12.1	112_01_escape — Czas ucieczki bez liczb zespolonych	152
12.2	112_02_convert — Od r do c i z powrotem	159
13.1	113_01_energy — Liczba, która nie może drgnąć	164
13.2	113_02_apart — Kiedy dwa puszczenia się rozchodzą?	168
14.1	114_01_chance — Szansa na deszcz i rozliczanie prognoz . . .	180
14.2	114_02_saturation — Kiedy zespół przestaje się rozchodzić? .	182
15.1	115_01_return — Krzywa czy chmura?	192
15.2	115_02_forecast — Zapytaj kilka analogów	196
16.1	116_01_collapse — Kiedy ginie odwzorowanie podwajające? .	205
16.2	116_02_single — Pojedyncza przeciw podwójnej	207
16.3	116_03_loops — Każda orbita kończy się pętlą	213
17.1	117_01_controller — Twój własny sterownik	222
17.2	117_02_sync — Ile czasu potrzebuje kopia?	224
18.1	118_01_horizon — Kalkulator horyzontu	232
18.2	118_02_guide — Przewodnik terenowy na modelu, który sam piszesz	237

Manifest wykresów

ch01-two-models — wzór i reguła krokowa	6
ch01-gap-shrinks — błąd, który zanika	8
ch02-three-fates — trzy reguły liniowe i ich trzy losy	15
ch02-two-accounts — wzrost liniowy wobec wykładniczego .	16
ch02-cobweb — pajęczyna na prostej	23
ch03-s-curve — wzrost wykładniczy a logistyczny	30
ch03-zoom — przybliżanie, aż krzywa wygląda na prostą	32
ch03-overshoot — przestrzelenie i to samo prawo bez opóźnienia	36
ch04-euler-rk4 — Euler uciekający spiralą i RK4	45
ch04-pendulum-map — portret fazowy wahadła	47
ch04-settle — ustalanie się w punkcie i na pętli	48
ch05-four-series — cztery przebiegi w czasie	56
ch05-four-cobwebs — cztery pajęczyny na garbie	60
ch05-window — cykl długości trzy wewnątrz chaosu	62
ch05-two-starts — dwa bliskie początki przy jednym ustawieniu	63
ch06-peel — dwa przebiegi się rozchodzą	68
ch06-gap-log — różnica, która rośnie, a potem sufit	70
ch07-running-average — średnia, która się ustala	85
ch07-horizon-line — horyzont a błąd stanu początkowego . .	87

ch07-exponent-r — wykładnik w zależności od r	90
ch08-diagram — diagram bifurkacyjny	94
ch08-ratios — stosunki dla dwóch reguł zbiegają się	100
ch08-zoom — okno okresu trzy w powiększeniu	103
ch09-butterfly — motyl z dwóch stron	111
ch09-apart — dwa rozchodzące się przebiegi	115
ch09-histograms — histogramy obojętne na start	118
ch10-attractor — atraktor i jego basen	126
ch10-square-folds — kwadrat rozciągnięty i złożony	128
ch10-zooms — atraktor powiększony dwa razy	130
ch11-koch-stages — krzywa Kocha runda po rundzie	136
ch11-three-fates — trzy losy długości	140
ch11-box-counts — zliczenia pudełek w skali logarytmicznej	143
ch12-mandelbrot — zbiór Mandelbrota i trzy powiększenia	153
ch12-julia — zbiór Julii w jednym kawałku i w pyłe	155
ch12-real-axis — oś rzeczywista nad diagramem bifurkacyjnym	157
ch13-two-releases — dwa puszczenia, które się rozchodzą	166
ch13-energy-error — sprawdzian energii	167
ch13-gentle-wild — spokojne i dzikie puszczenia	169
ch14-plume — wachlarz prognozy zespołowej	178
ch14-skill — rozrzut i błąd a wyprzedzenie	181
ch14-climate — statystyki klimatu przy dwóch wymuszeniach	184
ch15-twins — dwa szeregi, jeden histogram	190
ch15-return-maps — krzywa i chmura	190
ch15-forecast — błąd prognozy a liczba kroków naprzód	194
ch15-noisy — krzywa grubiejąca w szumie	195
ch16-collapse — odwzorowanie podwajające ginące w komputerze	205
ch16-parting — dwie precyzje, które się rozchodzą	207
ch16-statistics — statystyki, które przeżywają złą drogę	211
ch17-control — sterowanie włączone w trakcie przebiegu	221
ch17-sync — kopia zgrywająca się z oryginałem	224
ch18-digits — horyzont a liczba poprawnych cyfr	231
ch18-odds — prognoza, która staje się szansami	233

Manifest diagramów

ch01-model-loop.mmd — stan, reguła, przewidywanie, sprawdzenie	9
ch02-feed-back.mmd — pętla iteracji	13
ch03-crowding-loop.mmd — zatłoczenie jako ujemna pętla	28
ch04-step-loop.mmd — reguła zmiany ślędzona krokami	42
ch06-digits.mmd — cyfry na wejściu, kroki na wyjściu	73
ch07-three-steps.mmd — rozciągnięcie, wykładnik, horyzont	86
ch08-universal.mmd — każdy garb, jedna droga, jeden stosunek	101
ch09-feedback.mmd — trzy liczby w pętli	109
ch10-section.mmd — przepływ, przekrój, odwzorowanie	131
ch11-box-recipe.mmd — pokryj, policz, nachylenie	141

ch12-two-questions.mmd — jedna reguła, dwa zbiory	155
ch13-bodies.mmd — dwa ciała, trzy ciała, planety	170
ch14-forecast-cycle.mmd — analiza, zespół, prognoza	179
ch16-shadow.mmd — obliczona orbita i jej cień	211
ch17-wait-catch.mmd — czekaj, złap, trzymaj	219
ch17-drive.mmd — jeden sygnał, od oryginału do kopii	222
ch18-field-guide.mmd — przewodnik terenowy w trzech ruchach	236

Skorowidz

- analiza (meteorologia), 175
- arytmetyka dokładna, 208
- atraktor, 47, 115, 125
 - statystyki, 117, 232
- atraktor dziwny, 121
 - jako fraktal, 142
 - warstwy, 130
- atraktor Hénona
 - wymiar, 142
- bakterie, 26
- basen przyciągania, 125
- bezpieczna łączność, 225
- bisekcja, 96
- błąd
 - stanu początkowego, 8
- chaos, 54
 - definicja, 63
 - dłaczego ma znaczenie, 237
 - odróżnianie od szumu, 188
- cieniowanie, 210
- cyfry dwójkowe, 71
- cykl, 37, 60
- cykl graniczny, 47
- cykl superstabilny, 97
- czas Lapunowa, 86
- czas podwojenia, 16
- czas ucieczki, 149
- demon Laplace’a, 9, 238
- deterministyczność, 63
- determinizm, 76
- diagram bifurkacyjny, 93
- diagram pajęczynowy, 22, 59
- dynamika populacji, 196
- dyssypacja, 127
- efekt motyla, 67, 234
- energia, 164
- epidemia, 196
- Feigenbaum, Mitchell, 97
- fraktal, 130, 134
 - w przyrodzie, 144
- Hénon, Michel, 123
- horyzont prognozy, 229
- horyzont przewidywania, 80, 86
- Hyperion, 171
- iteracja, 12
- klimat, 183
- klimatologia, 181
- kometa Halleya, 9
- konwekcja, 108
- krok czasowy, 41
- krzywa Kocha, 136
- krzywa S, 28
- laboratorium, 3
- Laskar, Jacques, 171
- liczby losowe, 212
- liczby zespolone, 147
 - mnożenie, 147
- liczby zmiennoprzecinkowe, 202
- liniowy generator kongruencyjny, 212
- logarytm, 16
 - naturalny, 82
- Lorenz, Edward, 74, 108
- Mandelbrot, Benoit, 134
- May, Robert, 64
- metoda analogów, 193
- metoda Eulera, 41
- metoda Rungego–Kutty, 43
- model, 2
 - błąd, 7
 - krokowy, 5
 - przewidywanie, 2
 - reguła, 2
 - stan, 2
- model Lorenza-96, 174
- model SIR, 196
- nachylenie, 21, 31

- najbliższy sąsiad, 193
- nieliniowość, 31
- odwzorowanie Hénona, 123
- odwzorowanie logistyczne, 54, 67
 - a zbiór Mandelbrota, 156
 - cztery zachowania, 55
 - reguła, 54
 - w komputerze, 206
- odwzorowanie namiotowe, 121
- odwzorowanie podwajające, 71, 204
- odwzorowanie powrotu, 190
- odwzorowanie sinusowe, 100
- okno okresowe, 62, 103
- okres, 60
- okres półtrwania, 16
- opóźnienie, 34
- orbita, 12
- oscylacja
 - tłumiona, 34
- Ott, Grebogi i Yorke, 217
- paradoks linii brzegowej, 134
- Pecora i Carroll, 222
- podwajanie, 26
- podwojenie okresu, 96
- podwójna precyzja, 202
- Poincaré, Henri, 170
- pole, 127
- portret fazowy, 45
- powtarzalność, 206
- prawo stygnięcia Newtona, 5
- problem trzech ciał, 170
- prognoza, 228
- prognoza pogody, 174
- prognoza probabilistyczna, 179
- prognoza zespołowa, 174, 232
 - rozrzut, 175
- projekcja klimatyczna, 183
- przekrój Poincarégo, 131
- przestrzelenie, 34
- przestrzeń fazowa, 45
- przewidywalność, 76
- przewidywanie
 - jak długo wystarcza, 73
 - przewodnik terenowy, 228, 236
- prędkość, 40
- punkt stały, 18, 31, 57
- niestabilny, 20, 216
- stabilność, 57
- stabilny, 20
- Python, 3
- reguła
 - liniowa, 14
 - następnego kroku, 12
- reguła liniowa, 14
- reguła zmiany, 40
- Richardson, Lewis Fry, 134
- RK4, 43, 110
- rozciąganie i składanie, 121
- rozciągnięcie, 80
- ruch, 40
- rynki, 198
- równanie różniczkowe, 40
- samopodobieństwo, 136
- sprawdzalność prognozy, 181
- sprzężenie zwrotne, 26
 - dodatnie, 27
 - ujemne, 27
- stabilność, 20, 31
- stan, 12
- stan przejściowy, 93
- stała Feigenbauma, 93
- sterowanie chaosem, 216, 235
 - koszt, 219
 - pchnięcie, 217
 - w laboratorium, 225
- superpozycja, 31
- surogat, 188
- symulacja
 - sprawdzian energii, 164
- synchronizacja, 216, 235
 - napęd i odpowiedź, 222
- system dwójkowy, 202
- szum, 188, 234
 - pomiarowy, 195
- trzecie prawo Keplera, 50
- twierdzenie KAM, 168
- twierdzenie Li–Yorke’a, 103
- twierdzenie Poincarégo–Bendixsona, 47
- układ Lorenza, 108
- motyl, 110

- Układ Słoneczny, 162
- uniwersalność, 100
- Verhulst, Pierre-François, 28
- wahadło, 45
- wahadło podwójne, 162
 - stan, 162
- wiarygodność prognozy, 179
- wrażliwość na warunki początkowe, 67,
113
 - test, 228
 - wahadło podwójne, 165
- wszechświat jak zegar, 9
- wykładnik Lapunowa, 80, 229
 - odzworowania logistycznego, 83
 - przepływu, 113
 - w zależności od r , 89
- wymiar
 - pudełkowy, 141
 - z liczby kopii, 138
- wyspy ruchu regularnego, 168
- wyznacznik, 127
- wzrost
 - wykładniczy, 26
- wzrost logistyczny, 28
- wzrost wykładniczy, 16
 - błędu, 69
- zagadnienie dwóch ciał, 50
- zagniatanie ciasta, 121
- zaokrąglanie, 202
- zbiór Cantora, 138
- zbiór Julii, 155
- zbiór Mandelbrota, 147
 - rysowanie, 152
- zliczanie pudełek, 141