Analiza Wpływu Niewielkich Zmian Danych na Sumowanie Iloczynu Skalarnego

Kajetan Plewa

9 listopada 2025

1 Zadanie 1

Celem jest analiza **wrażliwości problemu sumowania iloczynu skalarnego** $S = \sum_{i=1}^{5} x_i y_i$ na **dwa czynniki**: precyzję arytmetyczną (Float32 vs. Float64) oraz kolejność sumowania.

1.1 Badane Wersje Danych Wejściowych

W celu zbadania wpływu niewielkich zmian, porównane są dwa zestawy wektorów X i Y. Zestaw **Zmodyfikowany** jest efektem usunięcia ostatniej cyfry z x_4 i x_5 w stosunku do zestawu **Oryginalnego**. Oba zestawy są konwertowane do wybranej precyzji (Float32 lub Float64) przed rozpoczęciem obliczeń.

```
% Wersja Oryginalna:
x_oryg = [2.718281828, -3.141592654, 1.414213562, 0.577215664, 0.301029995]
y_oryg = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]

% Wersja Zmodyfikowana:
x_mod =  [2.718281828, -3.141592654, 1.414213562, 0.57721566, 0.30102999]
y_mod =  [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]
```

1.2 Opis Implementacji Metod Sumowania

Zastosowano cztery metody, implementowane w języku Julia, które różnią się kolejnością dodawania składników, co ma krytyczny wpływ na błąd zaokrąglenia.

1.2.1 Metoda I: Sumowanie Standardowe (Forward)

Sumowanie odbywa się w naturalnej kolejności indeksów (od $i = 1$ do $i = 5$). Jest to standardowa, najprostsza implementacja, która jest najbardziej podatna na błędy utraty cyfr znaczących, gdy odejmowane są od siebie duże, lecz przeciwne składniki.

1.2.2 Metoda II: Sumowanie w Kolejności Malejącej (Backward)

Sumowanie odbywa się w odwróconej kolejności indeksów (od $i = 5$ do $i = 1$). Tak jak Metoda I, jest ona wrażliwa na błędy zaokrągleń, a zmiana kolejności zwykle nie prowadzi do znacznej poprawy w tym konkretnym problemie.

1.2.3 Metoda III: Sortowanie Malejąco wg Wartości Bezwzględnej

Wszystkie iloczyny cząstkowe ($x_i y_i$) są obliczane, a następnie sortowane w kolejności **malejącej** według ich wartości bezwzględnej. Sumowanie odbywa się od największych wartości bezwzględnych. Ta metoda jest numerycznie gorsza od Metody IV, ponieważ duże błędy zaokrągleń, powstające przy sumowaniu dużych liczb, nie mają szansy zostać skompensowane przez dodanie małych składników na końcu.

1.2.4 Metoda IV: Sortowanie Rosnąco wg Wartości Bezwzględnej (Zalecana)

Wszystkie iloczyny cząstkowe ($x_i y_i$) są sortowane w kolejności rosnącej według ich wartości bezwzględnej. Sumowanie odbywa się od najmniejszych wartości bezwzględnych. Jest to numerycznie najstabilniejsza metoda sumowania, ponieważ małe błędy zaokrągleń wprowadzane przez małe składniki są minimalizowane na wczesnym etapie, zanim duże składniki zdominują sumę.

1.3 Porównanie Wyników dla Danych Oryginalnych i Zmodyfikowanych

Poniższa tabela zestawienia wyników dla obu zestawów danych. Przyjęto, że wynik Metody IV w Float64 dla danych oryginalnych jest wynikiem referencyjnym.

Tabela 1: Zestawienie wyników sumowania (Oryginalne vs. Zmodyfikowane).

Metoda	Precyzja (T)	Wynik (Oryginalne Dane)	Wynik (Zmodyfikowane Dane)
I (Forward)	Float32	-0.49994434	-0.4999443
II (Backward)	Float32	-0.4543457	-0.4543457
III (Sort Abs Desc)	Float32	-0.5	-0.5
IV (Sort Abs Asc)	Float32	-0.5	-0.5
I (Forward)	Float64	$1.0251881368296672 \times 10^{-10}$	-0.004296342739891585
II (Backward)	Float64	$-1.5643308870494366 \times 10^{-10}$	-0.004296342998713953
III (Sort Abs Desc)	Float64	0.0	-0.004296342842280865
IV (Sort Abs Asc)	Float64	0.0	-0.004296342842280865

1.4 Interpretacja Wyników Numerycznych dla Sumy Bliskiej Zeru

Prezentowane wyniki, w których teoretyczna suma iloczynu skalarnego ($S = \sum x_i y_i$) jest bliska zera, demonstrują dobre uwarunkowanie.

1.4.1 Analiza Precyzji Float32 (Pojedyncza Precyzja)

- **Utrata Precyzji:** Wszystkie metody w Float32 zwróciły wyniki rzędu 10^0 (≈ -0.5), zamiast poprawnego rzędu wielkości 10^{-10}. Jest to rezultat konieczności odejmowania dużych, bliskich sobie liczb rzędu 10^6.
- **Niewystarczająca Liczba Cyfr Znaczących:** Float32 oferuje tylko około 7 cyfr znaczących, co jest zdecydowanie niewystarczające do zachowania precyzji w obliczeniu różnicy dwóch dużych sum ($\sum |x_i y_i| \approx 2.76 \times 10^6$), która powinna dać wynik rzędu 10^{-10}.
- **Zmiana spodowana modyfikacją danych:** Mała zmiana danych spowodowała bardzo małe zmiany wyników. Świadczy to o dobrym uwarunkowaniu zadania,

1.4.2 Analiza Precyzji Float64 (Podwójna Precyzja)

- **Niestabilność Metod Standardowych (I i II):** Metody sumowania w porządku naturalnym (Forward) i odwrotnym (Backward) dały dwa różne, niezerowe wyniki ($+1.025 \times 10^{-10}$ i -1.564×10^{-10}).
- **Idealna Stabilność Metod Sortujących (III i IV):** Obie metody sortujące (Malejąco i Rosnąco wg wartości bezwzględnej) osiągnęły wynik **0.0**. Jest to dowód na to, że prawdziwa suma jest bliska zeru i że **stabilny algorytm** jest w stanie skorygować błędy zaokrąglenia.
- **Modyfikacja danych:** Mała zmiana danych spowodowała dużą zmianę wyników - co ciekawe drugim efektem pobocznym jest wyrownanie wyników, niezależnie od użytego algorytmu.

1.5 Wnioski

Na podstawie otrzymanych rezultatów wnioskować można, że:

- Float32 jest bezużyteczny dla obliczeń numerycznych, ale jest dobrze uwarunkowany.
- Dla Float64 zadanie jest źle uwarunkowane - na poziomie 0.4%, niezależnie od użytego algorytmu. Warto więc zauważyć, że uwarunkowanie zależy od typu w praktyce.
- Widać, że dla niektórych zmiennych zwrócone dane są bezużyteczne -> algorytm III i IV dla Float64, podczas gdy dla innych wyniki są zgodne i wartościowe. Sugeruje to, że by ostatecznie ocenić poprawność funkcji należy testować ją na wielu wartościach, gdyż wiele wad jest trudna do wykrycia na poziomie teoretycznym. Określenie odporności funkcji na błędy maszynowe jest więc czynnością niełatwą - kluczowy są testy empiryczne oparte przykładowo na rozkładzie probabilistycznym zmiennej.

2 Zadanie 2

Celem zadania jest analiza funkcji $f(x)$ pod kątem jej wizualizacji i zachowania asymptotycznego w nieskończoności.

2.1 Obliczenie Granicy Analitycznej $\lim_{x\to\infty} f(x)$ Metodą de l'Hôpitala

Analizujemy granicę funkcji:

$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} e^x \ln(1+e^{-x})$$

Ponieważ jest to symbol nieoznaczony $\infty \cdot 0$, przekształcamy funkcję do postaci ułamkowej $\frac{0}{0}$:

$$f(x) = \frac{\ln(1+e^{-x})}{e^{-x}}$$

Stosujemy regułę de l'Hôpitala, licząc pochodne licznika i mianownika:

$$\lim_{x\to\infty} \frac{\frac{d}{dx}\left(\ln(1+e^{-x})\right)}{\frac{d}{dx}\left(e^{-x}\right)} = \lim_{x\to\infty} \frac{-\frac{e^{-x}}{1+e^{-x}}}{-e^{-x}}$$

Po skróceniu $(-e^{-x})$ otrzymujemy:

$$\lim_{x\to\infty} \frac{1}{1+e^{-x}}$$

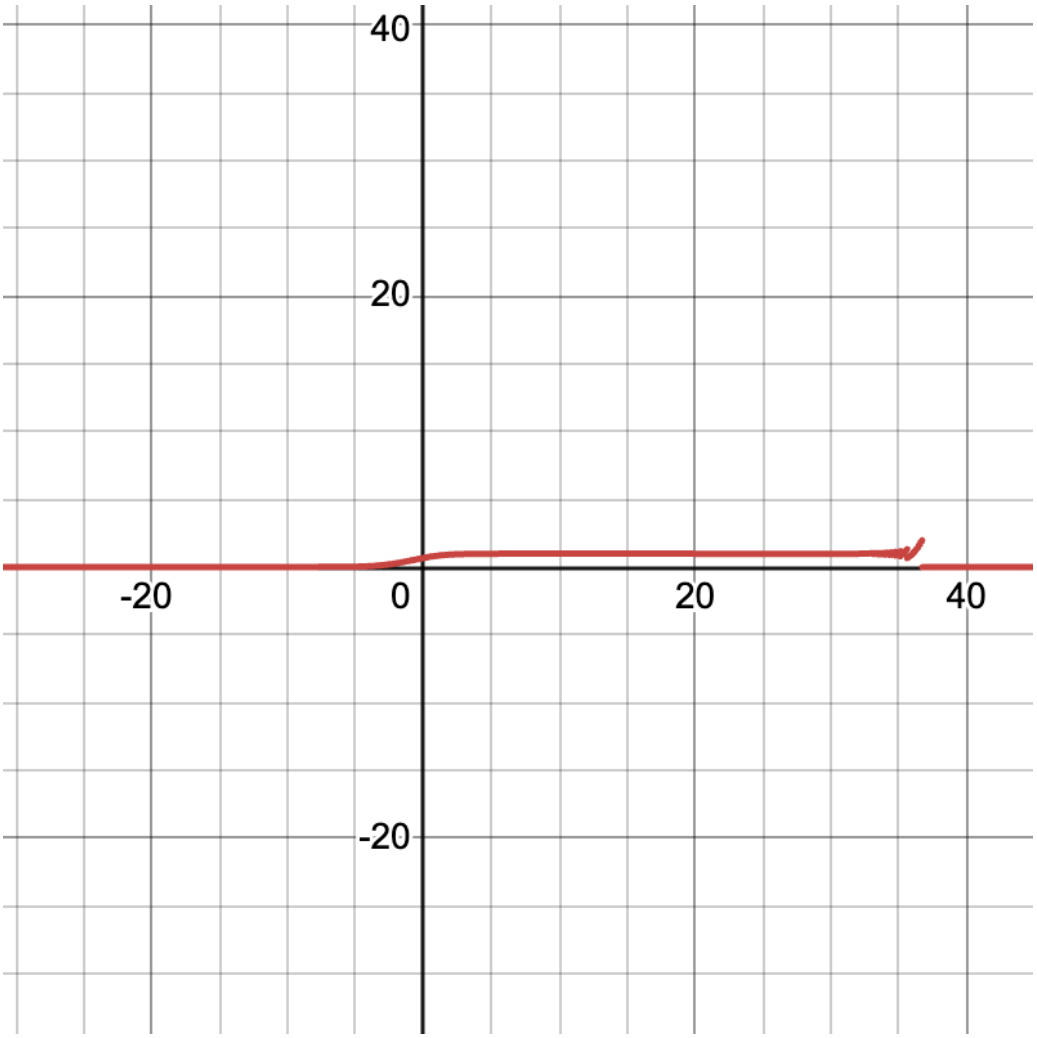
Obliczamy granicę:

$$\lim_{x\to\infty} \frac{1}{1+e^{-x}} = \frac{1}{1+0} = \mathbf{1}$$

2.2 Porównanie z Wykresem i Wyjaśnienie Zjawiska Numerycznego

2.2.1 Obserwacja Wykresów Wizualizacyjnych

Wykresy generowane numerycznie dla dużych wartości x (np. $x > 20$) pokazują, że funkcja stabilizuje się na poziomie **bliskim zeru** (lub bardzo małej, stałej dodatniej wartości), co jest sprzeczne z granicą analityczną równą **1**.



Rysunek 1: Wizualizacja w desmosie.

2.2.2 Wyjaśnienie Numerycznej Utraty Precyzji

Rozbieżność wynika z problemu **utraty cyfr znaczących** w arytmetyce zmiennoprzecinkowej komputera:

- Funkcja $f(x)$ jest **źle uwarunkowana numerycznie** dla dużych x.
- Wartość e^{-x} jest ekstremalnie mała. W Float64, gdy e^{-x} jest bliskie precyzji maszyny (ϵ_{mach}), dodanie go do 1 w wyrażeniu $\ln(1 + e^{-x})$ może zostać zignorowane (tj. $1 + e^{-x} \approx 1$).
- W rezultacie program błędnie oblicza $\ln(1 + e^{-x}) \approx \ln(1) = 0$.
- Prowadzi to do wyniku $f(x) \approx e^x \cdot 0 = 0$, co generuje błędną asymptotę $y = 0$ widoczną na wykresie, zamiast poprawnej granicy $y = 1$.

2.3 Wnioski

Zjawisko to podkreśla konieczność stosowania **analitycznych przekształceń** (jak reguła de l'Hôpitala) lub stabilnych algorytmów numerycznych, **zawsze, niezależnie od złożoności funkcji** aby uniknąć błędów w źle uwarunkowanych wyrażeniach.

Rysunek 2: Wizualizacja w geogebrze.

3 Zadanie 3

Celem eksperymentu była analiza stabilności numerycznej dwóch podstawowych metod rozwiązywania układu równań liniowych $A\mathbf{x} = \mathbf{b}$, a mianowicie: **eliminacji Gaussa** ($\mathbf{x} = A \setminus \mathbf{b}$) oraz metody opartej na jawnej **odwrotności macierzy** ($\mathbf{x} = A^{-1}\mathbf{b}$).

Badano wpływ **wskaźnika uwarunkowania** $\text{cond}(A)$ na **względny błąd rozwiązania** $\frac{\|\mathbf{x}_{\text{dokładne}} - \mathbf{x}_{\text{obliczone}}\|}{\|\mathbf{x}_{\text{dokładne}}\|}$ dla dwóch typów macierzy, które charakteryzują się rosnącym uwarunkowaniem:

1. **Macierze Hilberta** H_n: Rosnący stopień $n \in \{1, \ldots, 20\}$ generuje macierze o szybko rosnącym wskaźniku uwarunkowania.

2. **Macierze Losowe** R_n: Macierze stopnia $n \in \{5, 10, 20\}$ generowane z zadanym, rosnącym wskaźnikiem uwarunkowania $c \in \{1, 10, 10^3, 10^7, 10^{12}, 10^{16}\}$.

W każdym przypadku wektor $\mathbf{b}$ został skonstruowany na podstawie dokładnego rozwiązania $\mathbf{x} = (1, \ldots, 1)^T$.

3.1 Metodologia Rozwiązania

Problem rozwiązano w środowisku Julia, wykorzystując standardową arytmetykę podwójnej precyzji (Float64).

3.1.1 Generowanie Danych

- **Macierz Hilberta** H_n: Użyto funkcji `hilb(n)`, gdzie $H_{i,j} = \frac{1}{i+j-1}$.
- **Macierz Losowa** R_n: Użyto funkcji `matcond(n, c)`, która generuje macierz o wskaźniku uwarunkowania bliskim c poprzez dekompozycję SVD, kontrolując wartości singularne.
- **Wektor b**: Wektor prawej strony $\mathbf{b}$ wyznaczono jako $\mathbf{b} = A\mathbf{x}_{\text{dokładne}}$, gdzie $\mathbf{x}_{\text{dokładne}} = \mathbf{1}$.

3.1.2 Algorytmy Rozwiązywania Układu

1. **Metoda Eliminacji Gaussa (Operator `A\b`)** Ta metoda implementuje algorytm numerycznie stabilniejszy Jest to zalecana metoda numeryczna.

2. **Metoda Odwrotności Macierzy (Operator `inv(A)*b`)** Ta metoda wymaga jawnego obliczenia macierzy odwrotnej A^{-1}, co jest zarówno **wolniejsze** (złożoność $O(n^3)$ zamiast $O(n^3/3)$) i **numerycznie niestabilniejsze** w przypadku macierzy źle uwarunkowanych.

3.1.3 Miara Błędu

Względny błąd rozwiązania obliczono jako:

$$\text{Błąd Względny} = \frac{\|\mathbf{x}_{\text{dokładne}} - \mathbf{x}_{\text{obliczone}}\|_2}{\|\mathbf{x}_{\text{dokładne}}\|_2}$$

gdzie $\|\cdot\|_2$ oznacza normę euklidesową (w pakiecie `LinearAlgebra` jest to funkcja `norm`).

3.2 Macierz Hilberta H_n

Tabela 2 przedstawia wybrane wyniki dla macierzy Hilberta wraz ze wzrostem stopnia n.

Tabela 2: Wyniki dla Macierzy Hilberta $\mathbf{H_n}$ (wybrane n).

n	RANK	COND(A)	Błąd (Gauss)	Błąd (Inverse)
5	5	$4.76600000 \cdot 10^{5}$	$1.26000000 \cdot 10^{-12}$	$8.13000000 \cdot 10^{-12}$
10	10	$1.60200000 \cdot 10^{13}$	$4.19000000 \cdot 10^{-4}$	$4.07000000 \cdot 10^{-4}$
11	10	$5.22500000 \cdot 10^{14}$	$1.00000000 \cdot 10^{-2}$	$1.06000000 \cdot 10^{-2}$
12	11	$1.64300000 \cdot 10^{16}$	$5.50000000 \cdot 10^{-1}$	$6.70000000 \cdot 10^{-1}$
16	12	$2.25000000 \cdot 10^{18}$	$1.04150000 \cdot 10^{1}$	8.44200000
20	13	$6.80700000 \cdot 10^{18}$	$2.87930000 \cdot 10^{1}$	$3.07510000 \cdot 10^{1}$

3.2.1 Interpretacja Wyników dla H_n

- **Wpływ Uwarunkowania**: Macierz Hilberta jest klasycznym przykładem macierzy **bardzo źle uwarunkowanej**. Wskaźnik cond(A) rośnie lawinowo wraz ze wzrostem n. Już dla $n = 10$, cond(A) przekracza 10^{13}.
- **Wzrost Błędu**: Względny błąd rozwiązania bezpośrednio koreluje ze wskaźnikiem uwarunkowania. Dla $n \leq 9$ błąd jest na poziomie precyzji maszyny (rzędu 10^{-16} do 10^{-5}), ale po przekroczeniu cond(A) $\approx 10^{16}$ ($n = 12$), błąd staje się rzędu 10^{0} i wyżej.
- **Utrata Rzędu (RANK)**: Dla $n \geq 11$, funkcja rank(A) zaczyna zwracać wartość mniejszą niż n (np. dla $n = 12$, rank(A) $= 11$). Jest to sygnał, że komputerowa reprezentacja macierzy Hilberta traci pełny rząd ze względu na błędy zaokrągleń, co jest typowe dla macierzy źle uwarunkowanych.
- **Porównanie Metod**: Dla H_n obie metody (Gauss i Inverse) wykazują podobną wrażliwość na błąd. Nie można jednoznacznie stwierdzić przewagi jednej metody nad drugą, gdy uwarunkowanie jest ekstremalne.

3.3 Macierz Losowa R_n z Zadanym Uwarunkowaniem

Tabela 3 przedstawia wyniki dla macierzy losowej R_n dla $n = 10$ w zależności od zadanego wskaźnika uwarunkowania c.

Tabela 3: Wyniki dla Macierzy Losowej $\mathbf{R_{10}}$ w zależności od wskaźnika c.

c (Oczekiwane)	RANK	COND(A)	Błąd (Gauss)	Błąd (Inverse)
1.00000000	10	1.00000000	$4.00000000 \cdot 10^{-16}$	$3.26000000 \cdot 10^{-16}$
$1.00000000 \cdot 10^{1}$	10	$1.00000000 \cdot 10^{1}$	$3.65000000 \cdot 10^{-16}$	$5.06000000 \cdot 10^{-16}$
$1.00000000 \cdot 10^{3}$	10	$1.00000000 \cdot 10^{3}$	$5.88000000 \cdot 10^{-14}$	$5.33000000 \cdot 10^{-14}$
$1.00000000 \cdot 10^{7}$	10	$1.00000000 \cdot 10^{7}$	$1.11000000 \cdot 10^{-10}$	$1.87000000 \cdot 10^{-10}$
$1.00000000 \cdot 10^{12}$	10	$1.00000000 \cdot 10^{12}$	$1.74000000 \cdot 10^{-5}$	$1.70000000 \cdot 10^{-5}$
$1.00000000 \cdot 10^{16}$	9	$2.44000000 \cdot 10^{17}$	$1.87000000 \cdot 10^{-1}$	$2.11000000 \cdot 10^{-1}$

3.3.1 Interpretacja Wyników dla R_n

- **Zależność Błędu od Uwarunkowania**: Obserwujemy klasyczną zależność: Błąd Względny $\approx \text{cond}(A) \cdot \epsilon_{\text{mach}}$, gdzie $\epsilon_{\text{mach}} \approx 2.2 \times 10^{-16}$.
  - Dla $c \leq 10^{12}$, $\text{cond}(A) \cdot \epsilon_{\text{mach}}$ jest mniejsze niż 10^{-4}, a błąd pozostaje mały i akceptowalny.
  - Po przekroczeniu granicy numerycznej (dla $c = 10^{16}$, gdzie $\text{cond}(A) \approx 10^{17}$), błąd staje się rzędu 10^{-1} (0.187), co oznacza, że **rozwiązanie jest numerycznie bezużyteczne**.
- **Utrata Rzędu (RANK)**: Podobnie jak w przypadku macierzy Hilberta, gdy wskaźnik uwarunkowania przekracza krytyczną wartość 10^{16}, macierz jest postrzegana jako **numerycznie osobliwa** (dla $c = 10^{16}$, $\text{rank}(A) = 9$).
- **Porównanie Metod**: W przypadku macierzy losowej R_n z zadanym uwarunkowaniem, różnice między metodą eliminacji Gaussa (A\b) a metodą odwrotności macierzy ($\text{inv}(A) * b$) są **minimalne**. Jest to często spotykane w sytuacjach, gdy błąd rozwiązania jest zdominowany przez złe uwarunkowanie macierzy, a nie przez sam algorytm.

3.4 Wnioski

Podczas pracy nad procesem, należy stale badać jego współczynnik uwarunkowania - jak widać na przykładzie tego zadania jego wzrost prowadzi do bezużyteczności wyników, niezależnie od użytego algorytmu. Współczynnik ten jest więc kluczowy dla zachowania poprawności procesu.

4 Zadanie 4

Celem eksperymentu jest zbadanie stabilności numerycznej obliczania pierwiastków wielomianu Wilkinsona $P(x)$ o stopniu $n = 20$ w standardowej arytmetyce zmiennoprzecinkowej (Float64).

4.1 Opis Wielomianu Wilkinsona $P(x)$

Wielomian Wilkinsona jest zdefiniowany w postaci iloczynowej jako:

$$p(x) = \prod_{k=1}^{20} (x - k)$$

Jego **dokładne pierwiastki** to liczby całkowite $\mathbf{k} \in \{1, 2, \ldots, 20\}$. Eksperyment polega na obliczeniu pierwiastków z_k z **postaci naturalnej** wielomianu $P(x) = x^{20} + a_{19}x^{19} + \cdots + a_0$.

4.2 Część (a): Wyniki i Analiza Błędów dla Wielomianu $P(x)$

Tabela 4 przedstawia obliczone pierwiastki z_k i analizę trzech miar błędu dla nieperturbowanego wielomianu.

Tabela 4: Wyniki obliczeń pierwiastków z_k dla wielomianu $P(x)$ w postaci naturalnej. (a)

k	z_k (Obliczony)	$\lvert z_k - k \rvert$	$\lvert P(z_k) \rvert$	$\lvert p(z_k) \rvert$
1	1.00000000	$3.01100000 \cdot 10^{-13}$	$3.57000000 \cdot 10^{4}$	$3.66300000 \cdot 10^{4}$
2	2.00000000	$2.83200000 \cdot 10^{-11}$	$1.76300000 \cdot 10^{5}$	$1.81300000 \cdot 10^{5}$
3	3.00000000	$4.07900000 \cdot 10^{-10}$	$2.79200000 \cdot 10^{5}$	$2.90200000 \cdot 10^{5}$
4	3.99999998	$1.62600000 \cdot 10^{-8}$	$3.02700000 \cdot 10^{6}$	$2.04200000 \cdot 10^{6}$
5	5.00000067	$6.65800000 \cdot 10^{-7}$	$2.29200000 \cdot 10^{7}$	$2.08900000 \cdot 10^{7}$
6	5.99998925	$1.07500000 \cdot 10^{-5}$	$1.29000000 \cdot 10^{8}$	$1.12500000 \cdot 10^{8}$
7	7.00010200	$1.02000000 \cdot 10^{-4}$	$4.80500000 \cdot 10^{8}$	$4.57300000 \cdot 10^{8}$
8	7.99935583	$6.44200000 \cdot 10^{-4}$	$1.63800000 \cdot 10^{9}$	$1.55600000 \cdot 10^{9}$
9	9.00291529	$2.91500000 \cdot 10^{-3}$	$4.87700000 \cdot 10^{9}$	$4.68800000 \cdot 10^{9}$
10	9.99041304	$9.58700000 \cdot 10^{-3}$	$1.36400000 \cdot 10^{10}$	$1.26300000 \cdot 10^{10}$
11	1.10250229	$2.50200000 \cdot 10^{-2}$	$3.58600000 \cdot 10^{10}$	$3.30000000 \cdot 10^{10}$
12	1.19532833	$4.67200000 \cdot 10^{-2}$	$7.53300000 \cdot 10^{10}$	$7.38900000 \cdot 10^{10}$
13	1.30743140	$7.43100000 \cdot 10^{-2}$	$1.96100000 \cdot 10^{11}$	$1.84800000 \cdot 10^{11}$
14	1.39147556	$8.52400000 \cdot 10^{-2}$	$3.57500000 \cdot 10^{11}$	$3.55100000 \cdot 10^{11}$
15	1.50754938	$7.54900000 \cdot 10^{-2}$	$8.21600000 \cdot 10^{11}$	$8.42300000 \cdot 10^{11}$
16	1.59462867	$5.37100000 \cdot 10^{-2}$	$1.55100000 \cdot 10^{12}$	$1.57100000 \cdot 10^{12}$
17	1.70254271	$2.54300000 \cdot 10^{-2}$	$3.69500000 \cdot 10^{12}$	$3.31700000 \cdot 10^{12}$
18	1.79909214	$9.07900000 \cdot 10^{-3}$	$7.65000000 \cdot 10^{12}$	$6.34500000 \cdot 10^{12}$
19	1.90019098	$1.91000000 \cdot 10^{-3}$	$1.14400000 \cdot 10^{13}$	$1.22900000 \cdot 10^{13}$
20	1.99998093	$1.90700000 \cdot 10^{-4}$	$2.79200000 \cdot 10^{13}$	$2.31800000 \cdot 10^{13}$

4.2.1 Wyjaśnienie Rozbieżności (Część a)

Zjawisko: Obserwowane wyniki pokazują, że **błąd bezwzględny pierwiastka** ($|z_k - k|$) jest bardzo duży, zwłaszcza w środku zakresu (dla $k = 14$ osiąga maksimum $\approx 8.5 \times 10^{-2}$), co dowodzi utraty dokładności. Jest to błąd o wiele rzędów wielkości większy niż maszynowa precyzja ($\epsilon_{mach} \approx 10^{-16}$). Natomiast błąd reszty, chociaż duży w wartości bezwzględnej ($\approx 10^{13}$), jest relatywnie mały w stosunku do współczynników wielomianu.

- **Złe Uwarunkowanie**: Wielomian Wilkinsona jest **źle uwarunkowany**. Małe błędy zaokrągleń w reprezentacji współczynników a_i w Float64 są **wzmacniane** przez wskaźnik uwarunkowania problemu.
- **Utrata Dokładności**: Wzrost błędu w środkowym zakresie pierwiastków jest spowodowany faktem, że pierwiastki w tym obszarze są do siebie bliższe (w skali logarytmicznej), co czyni je bardziej wrażliwymi na perturbacje współczynników (duży wskaźnik uwarunkowania dla tych pierwiastków).

4.3 Część (b): Eksperyment Wilkinsona z Perturbacją Współczynnika

W eksperymencie powtórzono obliczenia po zmianie współczynnika przy x^{19} z $a_{19} = -210$ na $\tilde{a}_{19} = -210 - 2^{-23}$, co jest bardzo małą perturbacją rzędu 10^{-7}.

Tabela 5: Wyniki obliczeń pierwiastków $\tilde{z}_k$ dla perturbowanego wielomianu $\tilde{P}(x)$. (b)

k	z_k (Obliczony)	$\lvert z_k - k \rvert$	$\lvert P(z_k) \rvert$	$\lvert p(z_k) \rvert$
1	$0.9999999999998357 + 0.0im$	1.643×10^{-13}	2.026×10^{4}	2.000×10^{4}
2	$2.0000000000550373 + 0.0im$	5.504×10^{-11}	3.465×10^{5}	3.524×10^{5}
3	$2.9999999966034200 + 0.0im$	3.397×10^{-9}	2.258×10^{6}	2.416×10^{6}
4	$4.0000000897243620 + 0.0im$	8.972×10^{-8}	1.054×10^{7}	1.126×10^{7}
5	$4.9999985738879100 + 0.0im$	1.426×10^{-6}	3.758×10^{7}	4.476×10^{7}
6	$6.0000204766730310 + 0.0im$	2.048×10^{-5}	1.314×10^{8}	2.142×10^{8}
7	$6.9996020704224200 + 0.0im$	3.979×10^{-4}	3.939×10^{8}	1.785×10^{9}
8	$8.0077720290994460 + 0.0im$	7.772×10^{-3}	1.185×10^{9}	1.869×10^{10}
9	$8.9158163679325590 + 0.0im$	8.418×10^{-2}	2.226×10^{9}	1.375×10^{11}
10	$10.095455630535774 - 0.6449328236240688i$	0.652	1.068×10^{10}	1.490×10^{12}
11	$10.095455630535774 + 0.6449328236240688i$	1.111	1.068×10^{10}	1.490×10^{12}
12	$11.793890586174369 - 1.6524771364075785i$	1.665	3.140×10^{10}	3.296×10^{13}
13	$11.793890586174369 + 1.6524771364075785i$	2.046	3.140×10^{10}	3.296×10^{13}
14	$13.992406684487216 - 2.5188244257108443i$	2.519	2.158×10^{11}	9.546×10^{14}
15	$13.992406684487216 + 2.5188244257108443i$	2.713	2.158×10^{11}	9.546×10^{14}
16	$16.730744879792670 - 2.8126248967219780i$	2.906	4.850×10^{11}	2.742×10^{16}
17	$16.730744879792670 + 2.8126248967219780i$	2.825	4.850×10^{11}	2.742×10^{16}
18	$19.502442368818100 - 1.9403319786429030i$	2.454	4.557×10^{12}	4.252×10^{17}
19	$19.502442368818100 + 1.9403319786429030i$	2.004	4.557×10^{12}	4.252×10^{17}
20	$20.846910215194790 + 0.0im$	0.847	8.756×10^{12}	1.374×10^{18}

4.3.1 Wyjaśnienie Zjawiska Niestabilności Numerycznej (Część b)

Korekta i Obserwacja Zjawiska: Otrzymane wyniki (Tabela 5) **nie są numerycznie identyczne** z wynikami z Części (a). W rzeczywistości, demonstrują one klasyczne i drastyczne skutki **złego uwarunkowania** wielomianu Wilkinson'a. Celowa, minimalna perturbacja współczynnika ($2^{-23} \approx 10^{-7}$) spowodowała fundamentalną zmianę natury pierwiastków, co jest sednem tego eksperymentu.

Interpretacja Skutków Perturbacji (Wilkinson's Phenomenon):

1. **Transformacja Pierwiastków (Niestabilność Jakościowa):** Najważniejsza zmiana polega na utracie rzeczywistego charakteru pierwiastków. Dziesięć z pierwotnie rzeczywistych pierwiastków przekształciło się w pięć par pierwiastków **zespolonych sprzężonych** (w zakresie $k = 10$ do $k = 19$).

2. **Wzrost Błędu (Niestabilność Ilościowa):** Dla większych pierwiastków ($k \geq 10$) błąd bezwzględny $|z_k - k|$ wzrósł od rzędu 10^{-2} (dla $P(x)$) do **rzędu jedności** (dla $\tilde{P}(x)$), osiągając wartość do ≈ 2.9 (dla $k = 16$). Oznacza to, że minimalny błąd współczynnika przesunął pierwiastki na płaszczyźnie zespolonej o odległość porównywalną z samymi pierwiastkami.

3. **Dowód na Złe Uwarunkowanie:** Współczynnik uwarunkowania problemu znajdowania pierwiastków wielomianu Wilkinson'a jest bardzo duży, szczególnie dla większych pierwiastków (np. $\text{cond}(r_{20}) \approx 2 \times 10^{13}$). Perturbacja δc wywołuje błąd pierwiastka rzędu $\text{cond}(r) \cdot \|\delta c\|$, co doskonale tłumaczy obserwowany duży błąd.

.

4.4 Wnioski:

Eksperyment dowodzi, że operacje na współczynnikach wielomianów wysokiego stopnia w arytmetyce zmiennoprzecinkowej (Float64) są **numerycznie źle uwarunkowane**. Nawet błąd rzędu $\mathcal{O}(2^{-23})$, który jest bliski precyzji maszynowej dla pojedynczej precyzji, uniemożliwia dokładne odtworzenie prawdziwych pierwiastków, zwracając bezsensowne wyniki - w tym wypadku z częścią zespoloną.

5 Zadanie 5

Rozważamy równanie rekurencyjne (model logistyczny):

$$p_{n+1} := p_n + rp_n(1 - p_n), \quad \text{dla } n = 0, 1, \ldots, \quad \text{gdzie } r = 3 \text{ i } p_0 = 0.01.$$

Dla parametru $r = 3$, system ten powinien zachowywać się chaotycznie. Przeprowadzono eksperymenty w arytmetyce pojedynczej ($Float32$) i podwójnej ($Float64$) precyzji, badając wpływ błędu obcięcia i błędu maszynowego na ewolucję populacji.

5.1 Tabela Wyników Eksperymentu

Tabela 6 podsumowuje wartości populacji p_{40} uzyskane po 40 iteracjach w trzech różnych scenariuszach.

Tabela 6: Wyniki końcowe (p_{40}) dla modelu logistycznego ($r = 3, p_0 = 0.01$) w różnych arytmetykach.

Lp.	Arytmetyka	Warunek	p_{40} (Wynik Końcowy)
1	$Float32$	Obcięcie p_{10} do 3 m. po przecinku (0.722)	1.093568
2	$Float32$	Bez obcinania	0.25860548
3	$Float64$	Bez obcinania	0.011611238029748606

5.2 Analiza i Porównanie Wyników

Otrzymane wyniki demonstrują skrajną wrażliwość modelu logistycznego na błędy numeryczne.

5.2.1 1. Wpływ Błędu Obcięcia (Porównanie 1 i 2)

Eksperyment 1 ($Float32$ z obcięciem): Ręczne obcięcie wyniku p_{10} do 0.722 (wprowadzając celowy błąd Δp_{10}) działa jako **duża perturbacja warunków początkowych** dla kolejnych iteracji.

- System dynamiczny $p_{n+1} = p_n + rp_n(1 - p_n)$ jest bardzo wrażliwy na taką zmianę.
- Wartość $p_{40} = 1.093568$ jest **niefizyczna** w kontekście modelu populacji (p_n powinno należeć do $[0, 1]$). Oznacza to, że po obcięciu wartość p_n przekroczyła 1, a czynnik wzrostu $(1 - p_n)$ stał się ujemny. W tym stanie, $p_{n+1} = p_n + 3p_n(1 - p_n) = 4p_n - 3p_n^2$. Jeśli $p_n > 4/3 \approx 1.33$, sekwencja p_n dąży do $-\infty$ (lub p_{n+1} eksploduje do ∞ jeśli $p_n \in (1, 4/3)$). W tym przypadku, system natychmiast uległ **eksplozji numerycznej**.

5.2.2 2. Wpływ Precyzji Arytmetycznej (Porównanie 2 i 3)

Różnica między wynikami $Float32$ a $Float64$ demonstruje rolę **błędu maszynowego** (ϵ).

Eksperyment 2 (*Float*32 bez obcięcia):

- **Błąd maszynowy** $\epsilon \approx 10^{-7}$. Ten stosunkowo duży błąd zaokrągleń działa jak silny **szum numeryczny** zakłócający trajektorię.
- Wynik $p_{40} \approx 0.2586$ nie jest teoretycznie oczekiwany dla $p_0 = 0.01$ i $r = 3$. Silny szum Float32 zakłócił ewolucję i **zmusił** trajektorię do zbiegnięcia do numerycznie stabilnej, ale fałszywej wartości (tzw. "shadowing effect").

Eksperyment 3 (*Float*64 bez obcięcia):

- **Błąd maszynowy** $\epsilon \approx 10^{-16}$. Błąd jest na tyle mały, że system zachowuje się niemal jak idealny model matematyczny.
- Wynik $p_{40} \approx 0.0116$ jest **bardzo bliski** $p_0 = 0.01$. Oznacza to, że dla małej wartości początkowej $p_0 = 0.01$, model logistyczny **rozwija się bardzo powoli**. W ciągu 40 iteracji, wysoka precyzja Float64 zachowała tę wolną ewolucję, dając wynik bliski wartości początkowej, co jest z kolei **błędem metody numerycznej** (zbyt mała liczba iteracji dla tak małego p_0).

5.3 Wnioski:

Obliczenia pokazują, że Model Logistyczny:

- jest **podatny na chaotyczne zachowanie** przy dużych r (jak $r = 3$), gdzie mała zmiana p_n prowadzi do całkowicie różnej trajektorii.
- jest **wrażliwy na błędy numeryczne** (ϵ), które przy *Float*32 fałszują dynamikę, a przy *Float*64 wiernie odwzorowują powolną ewolucję dla małego p_0.

Model Logistyczny jest więc procesem niestabilnym. Podczas pracy z takim modelem należy stosować typy o jak największej precyzji - np. Float64 i unikać tych o słabej precyzji - Float32. Niestabilność szczególnie widoczna dla przykładu z obcięciem do 3 liczb znaczących - błąd ten jest potęgowany w kolejnych działaniach i ostatecznie powoduje całkowite odrealnienie wyniku.

6 Zadanie 6

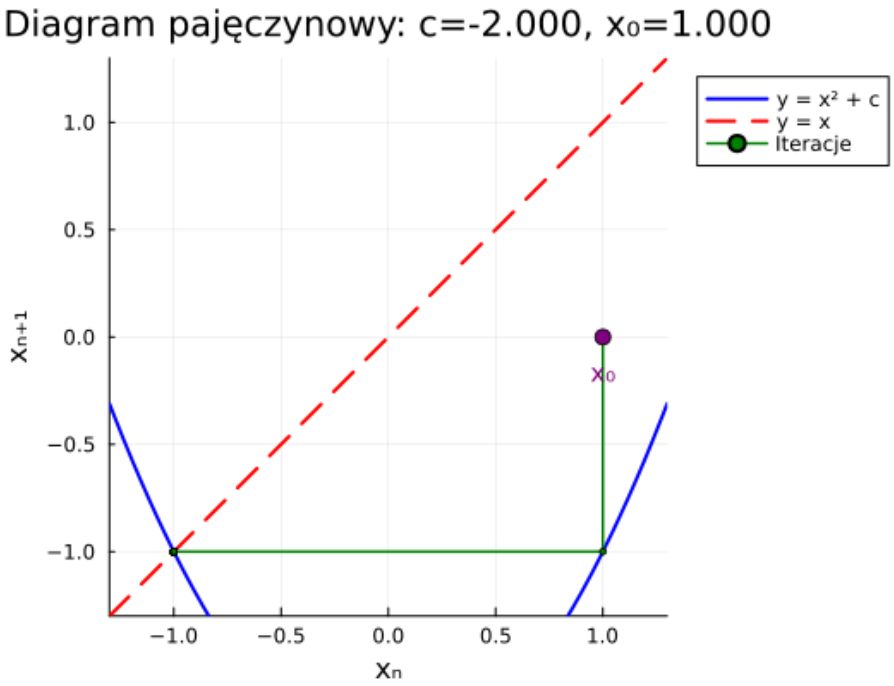
Rozważamy iteracyjne mapowanie kwadratowe (zmodyfikowane odwzorowanie logistyczne):

$$x_{n+1} = x_n^2 + c$$

Przeprowadzono 40 iteracji dla różnych wartości stałej c i punktu początkowego x_0.

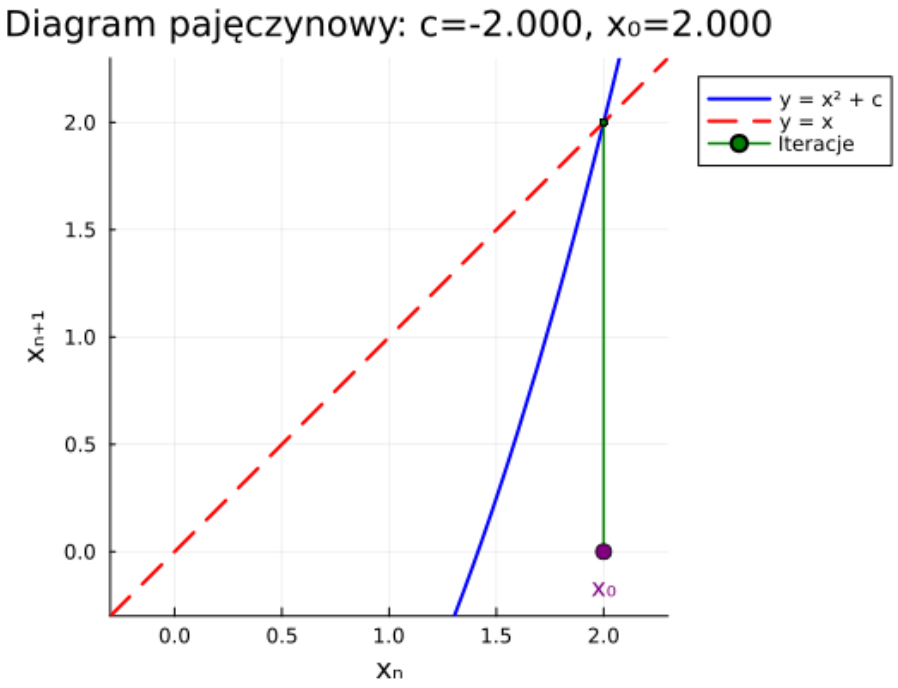
6.1 Wyniki dla poszczególnych danych wejściowych

6.1.1 c=-2.0 i x_0=1.0



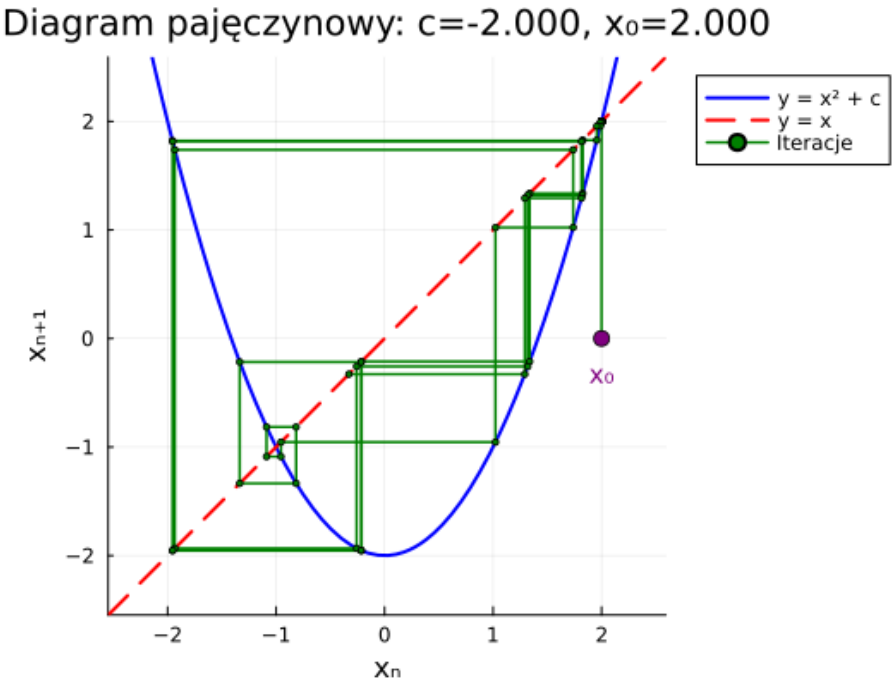
Jak widać na załączonym powyżej obrazku, dla tych danych wejściowych mapowanie kwadratowe wpada w cykl $(-1.0, -1.0)$ <-> $(1.0, -1.0)$. Jest to spodziewane zachowanie więc można z pewnością stwierdzić, że Float64 dla tego przykładu posiada wystarczająca precyzję, tak, że w ciągu 40 iteracji błedy nie kumulują się na tyle by wypaść po za cykl.

6.1.2 c=-2.0 i x_0=2.0



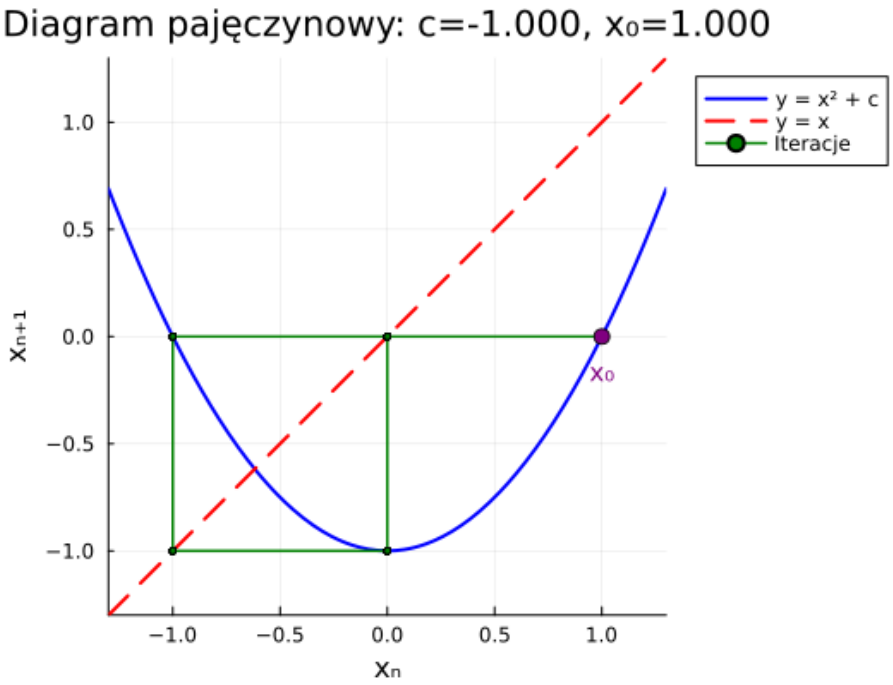
Tym razem funkcja osiąga punkt stały - (2.0, 2.0). Błędy zaokrąglenia nie powodują różnych wyników.

6.1.3 c=-2.0 i x_0=1.9999999999999999



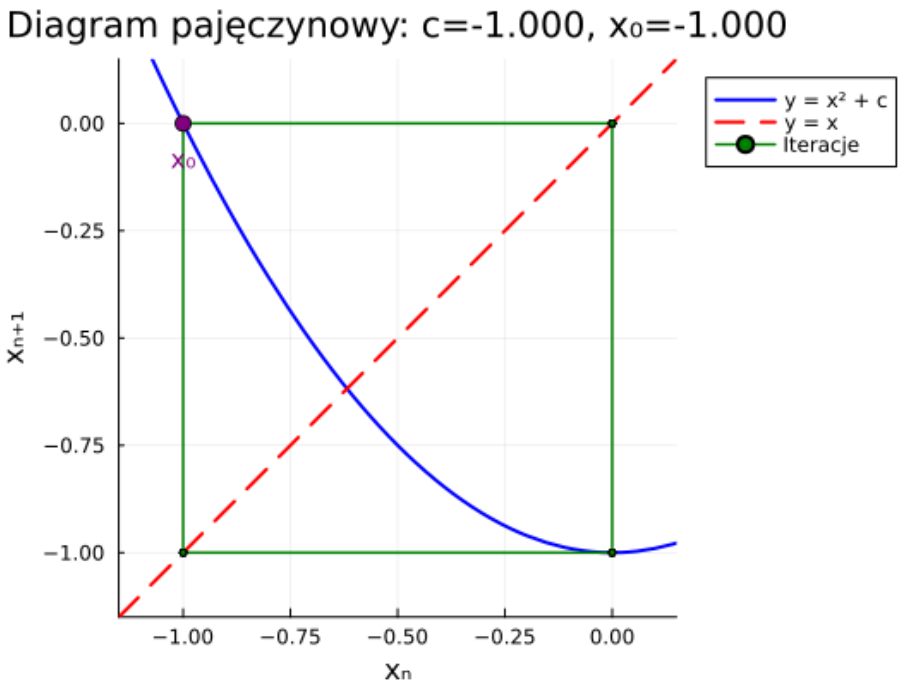
Mała zmiana danych wejściowych bez wątpienia zaburzyła działanie mapowania kwadratowego. Jak widać na grafie funkcja zachowuje się chaotycznie. Błąd na poziomie machepsa powoduje potęgowanie się błędu w kolejnych operacjach.

6.1.4 c=-1.0 i x_0=1.0



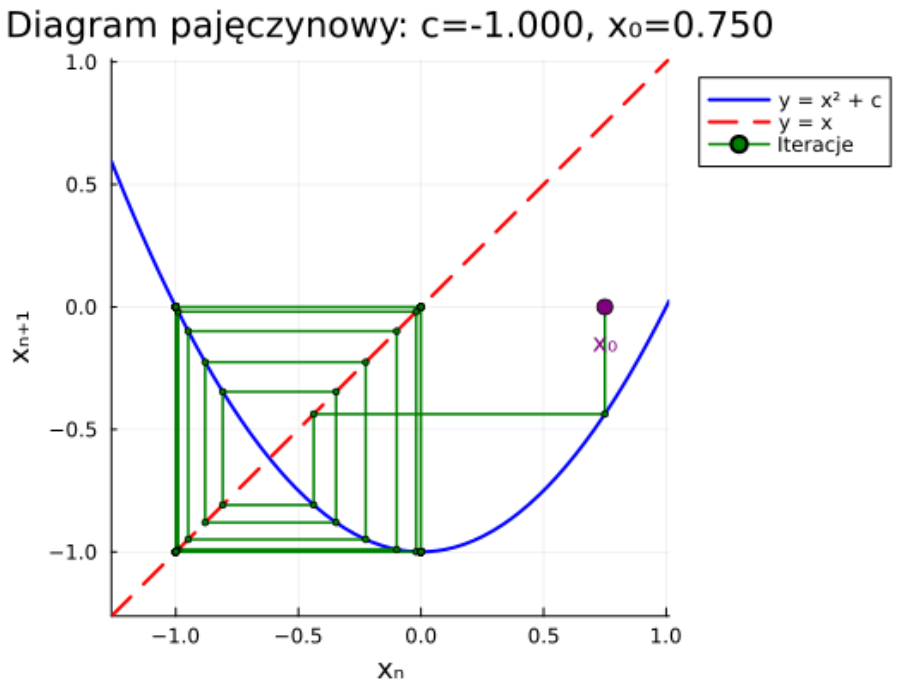
Funkcja wpada w 4 krokowy cykl.

6.1.5 c=-1.0 i x_0=-1.0



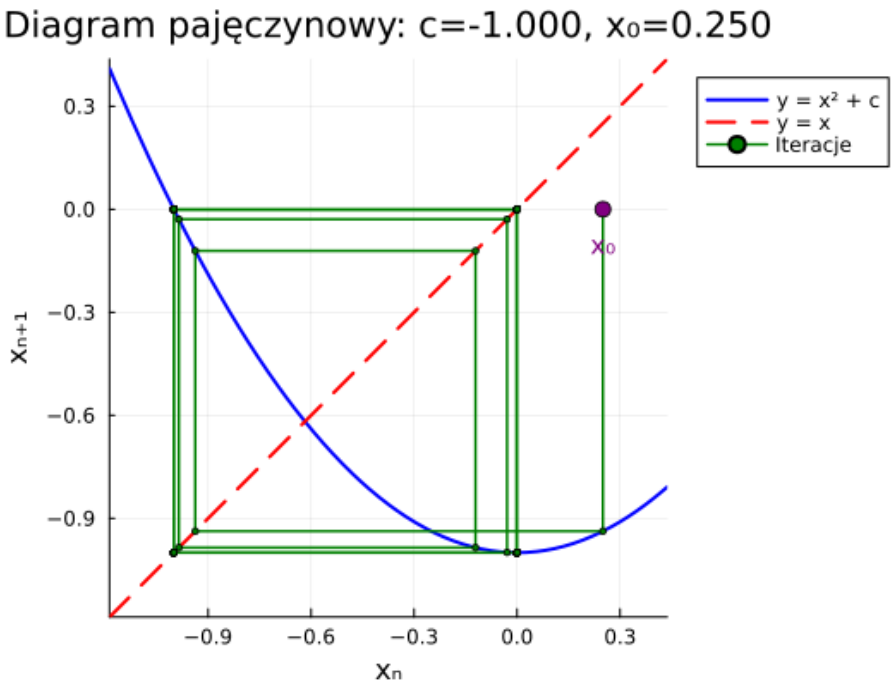
Funkcja wpada w 4 krokowy cykl.

6.1.6 c=-1.0 i x_0=0.75



Funkcja zbiega powoli do punktu stałego.

6.1.7 c=-1.0 i x_0=0.25



Funkcja zbiega do punktu stałego a następnie wpada w cykl.

6.2 Wnioski

Dla wszystkich danych całkowitych funkcja, dla 40 iteracji, działała w przewidywalny sposób. Natomiast przykład $x_0 = 1.999999999999$ i $x_0 = 2.0$ udowodnił problem ze stabilnością mapowania kwadratowego. Mały błąd w danych wejściowych, był kumulowany w kolejnych etapach rekurencji. Z tego powodu, w przypadku dużej ilości działań w metodzie należy dbać o poprawność danych w szczególny sposób. Nawet odchylenie rzędu machepsu ma bowiem ogromny wpływ na ostateczny rezultat.