



Politechnika Wrocławskiego

Wydział Matematyki

Kierunek studiów: Matematyka stosowana

Specjalność: –

Praca dyplomowa – inżynierska

**PROCEDURALNE GENEROWANIE MAP
W GRACH KOMPUTEROWYCH**

Aleksander Rzyhak

słowa kluczowe:
proceduralne generowanie, gry komputerowe, pole losowe, izotropia, szum Perlina, szum Simplex, parkietaż aperiodyczny

krótkie streszczenie:

W pracy wprowadzono dwie nowe metody generowania pól losowych, które służą do opisu terenu w grach komputerowych. Są one rozwinięciem powszechnie używanych algorytmów – szumu Perlina i szumu Simplex. Nowe podejścia, wykorzystując aperiodyczny parkietaż jedną płytka, poprawiają problemy klasycznych szumów dotyczące braku izotropii oraz występowania artefaktów kierunkowych.

Opiekun pracy diplomowej	dr Jakub Ślęzak
	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

- a) kategorii A (akta wieczyste)
- b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

* niepotrzebne skreślić

pieczętka wydziałowa

Wrocław, rok 2025



Faculty of Pure and Applied Mathematics

Field of study: Applied Mathematics

Specialty: –

Engineering Thesis

**PROCEDURAL MAP GENERATION IN
VIDEO GAMES**

Aleksander Rzyhak

keywords:

procedural generation, video games, random field, isotropy, Perlin noise, Simplex noise, aperiodic tessellation

short summary:

The thesis introduces two new methods for generating random fields used to describe terrain in video games. These methods are built upon the widely used algorithms of Perlin noise and Simplex noise. By utilizing a single-tile aperiodic tessellation, the new methods address the issues of Perlin and Simplex noise, specifically the lack of isotropy and the presence of directional artifacts.

Supervisor	dr Jakub Ślęzak
	Title/degree/name and surname	grade	signature

*For the purposes of archival thesis qualified to:**

- a) category A (perpetual files)
- b) category BE 50 (subject to expertise after 50 years)

* delete as appropriate

stamp of the faculty

Wrocław, 2025

Spis treści

1 Wstęp	3
2 Metodologia	4
2.1 Szum Perlina	4
2.2 Szum Simplex	11
3 Analiza klasycznych szumów	15
3.1 Analiza statystyczna	16
3.2 Analiza częstotliwości	21
4 Aperiodyczny parkietaz	22
4.1 Generowanie aperiodycznego parkietazu	24
4.2 Reprezentacja siatki deltoidów	29
5 Nowe szумy	29
6 Analiza nowych szumów	36
6.1 Analiza statystyczna	36
6.2 Analiza częstotliwości	39
7 Podsumowanie	40

1 Wstęp

Generowanie proceduralne to zbiór metod służących do tworzenia danych o zadanych charakterystykach w sposób zautomatyzowany. Algorytmy te służą artystom, twórcom filmów i gier do szybkiego i niewymagającego manualnej pracy generowania rozległych i złożonych struktur, takich jak: ukształtowanie terenu, budynek, postacie, zjawiska pogodowe, a nawet całe światy. Za pomocą uprzednio zdefiniowanych reguł, parametrów i ograniczeń metody te, zwykle z użyciem elementu losowego, zwracają parametry generowanego obiektu.

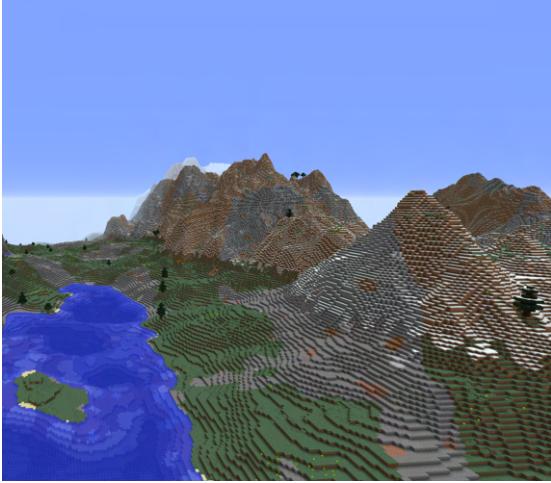
Główną zaletą tego typu metod jest możliwość efektywnego, ze względu na pamięć oraz czas, generowania różnicowanej treści. Pozwalają uniknąć przytrzymywania dużej ilości danych w pamięci komputera i tworzyć je w zależności od stanu programu, w którym są zaimplementowane. Proceduralne generowanie było używane szczególnie w czasach, kiedy komputery nie miały tak dużo pamięci, jak współcześnie.

Pierwsze przykłady wykorzystania proceduralnego generowania poprzedzają komputery cyfrowe. Jeden z nich pochodzi z końca XVIII wieku i dotyczy planszowej gry wojennej stworzonej dla pruskich oficerów w celach treningowych, gdzie pozycje fragmentów planszy mogły być zmieniane [5]. W kulturze popularnej pierwsze użycia metod proceduralnych pojawiły się w grach RPG (ang. Role Playing Games), gdzie za pomocą rzutów kostką gracze mogli tworzyć mapy Lochów, decydować o przebiegu starcia z wrogami oraz statystykach swoich postaci [16].

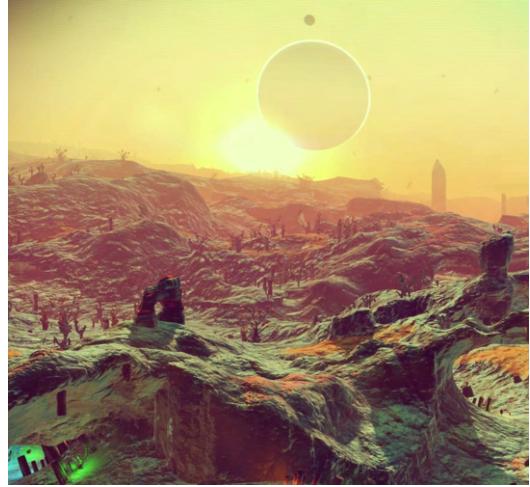
Jednymi z pierwszych cyfrowych przykładów wykorzystania algorytmów proceduralnego generowania są gry komputerowe *Beneath Apple Manor* (1978) i *Rogue* (1980) [2]. Poziomy, które przemierza gracz, przy każdej rozgrywce wyglądają inaczej, ale zachowują strukturę pokojów połączonych w planszę przypominającą labirynt. Dzięki losowaniu planszy gry urozmaicano rozgrywkę oraz zmniejszano rozmiar gry, ponieważ nie trzeba było przytrzymywać plansz w pamięci, a zamiast tego tworzone były w zależności od postępu gracza i etapu gry. Współcześnie algorytmy te mają zastosowania w grafice komputerowej (generowanie tekstur), w animacji oraz w grach komputerowych, gdzie generowane obiekty muszą dodatkowo być interaktywne.

Minusami takiego podejścia mogą być odczuwalna powtarzalność generowanej treści oraz brak punktów charakterystycznych, które byłyby uwzględnione przy manualnym projektowaniu danej struktury.

W tej pracy skupimy się na proceduralnym generowaniu terenu przy pomocy pól losowych. Te same metody mogą być również wykorzystane w innych celach, takich jak generowanie tekstur, rozkładu zasobów w świecie gry, czy też realistycznie wyglądających chmur [12]. Celem proceduralnego generowania terenu jest stworzenie zmiennych opisujących teren (np. górzysty lub pustynny), który wygląda naturalnie, unikalowo i



(a) *Minecraft*, źródło: planetminecraft.com.



(b) *No Man's Sky*, źródło: nmsspot.com.

Rysunek 1: Przykłady wykorzystania proceduralnego generowania terenu w grach komputerowych.

jednocześnie nie jest wymagające obliczeniowo. Dodatkowo algorytmy powinny być zaprojektowane w sposób, dla którego obliczenia w łatwy sposób mogą być wykonywane równolegle, ponieważ najczęściej implementuje się je jako programy działające na karcie graficznej.

Gry takiej jak *Minecraft*, czy *No Man's Sky* są jednymi z najpopularniejszych przykładów użycia szumu do generowania terenu [4]. Zrzuty ekranu prezentujące światy z tych gier widać na Rysunku 1. Gry te potrafią wygenerować bardzo duże światy, nie przechowując ich jednocześnie w pamięci gry i nie wymagając nakładu ludzkiej pracy w ich tworzeniu.

W pracy zbadane i omówione zostaną własności dwóch szumów, będących jednymi z najczęściej stosowanych do generowania terenu. Zaproponowana zostanie modyfikacja omówionych metod, która poprawia ich niedoskonałości, jednocześnie zachowując efektywność klasycznych algorytmów.

2 Metodologia

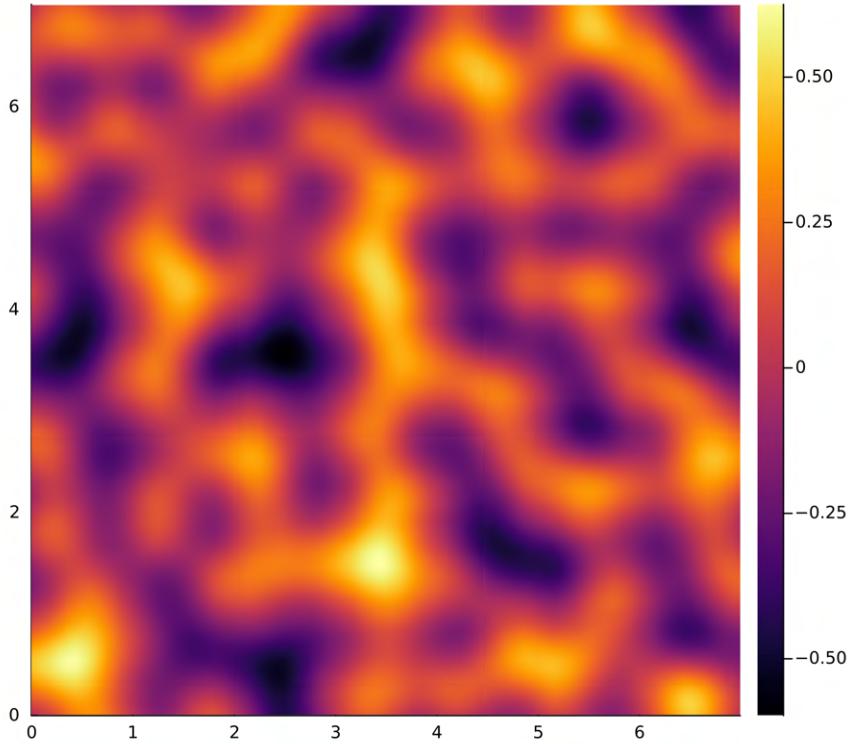
Omawiane metody służą do generowania pól losowych [13] (zwanych również szumami losowymi), będących uogólnieniem procesów stochastycznych.

Definicja 1 Dla przestrzeni probabilistycznej $(\Omega, \mathcal{F}, \mathbb{P})$, pole losowe T_x zdefiniowane na \mathbb{R}^n , jest to funkcja, która do każdej ustalonej wartości $x \in \mathbb{R}^n$ przypisuje zmienną losową z przestrzeni probabilistycznej $(\Omega, \mathcal{F}, \mathbb{P})$.

W pracy omówiony zostanie tylko przypadek dla przestrzeni \mathbb{R}^2 . Wykorzystane szumy pozwalają na zdefiniowanie ich wartości dla dowolnego punktu z \mathbb{R}^2 , natomiast, co oczywiste, w celach wizualizacji i analizy empirycznej procesów użyto ich dyskretnych przybliżeń.

2.1 Szum Perlina

Jednym z dwóch najczęściej używanych szumów jest szum Perlina [11]. Zaproponowany został w 1983 roku przez Kena Perlina. Definiuje się go dla przestrzeni \mathbb{R}^n (u nas \mathbb{R}^2) i przyjmuje wartości z przedziału $[-1, 1]$. Realizacja takiego pola losowego jest ciągła i wizualnie gładka (domyślnie klasy C^2). Zwykle używa się go do generowania wartości w dwóch, trzech lub czterech wymiarach. W pracy skupimy się na przypadku dwuwymiarowym, a wygenerowane wartości będziemy interpretować jako wysokość terenu w danym punkcie. Na Rysunku 2 widać przykładową realizację szumu Perlina. Można zauważyć, że rzeczywiście całość wygląda gładko i ma pewną „górzysto-dolinową” strukturę, która w kontekście generowania terenu jest pożądana.



Rysunek 2: Realizacja szumu Perlina.

Algorytm ten bazuje na siatce kwadratowej i tablicy losowych wektorów zlokalizowanych w wierzchołkach siatki. Wektory te zwykle są długości 1, ale nie jest to wymagane, a ich kąt nachylenia powinien być rozłożony jednostajnie. Często w implementacjach stosuje się uproszczenia i zamiast generować wektory spełniające powyższe warunki, losuje się je z wcześniej przygotowanej listy. Przykładowo dla każdego wierzchołka siatki losujemy wektor z listy $[[1, 0], [1, 1], [0, 1], [-1, 1], [-1, 0], [-1, -1], [0, -1], [1, -1]]$. Jest to szybszy algorytm i korzystało się z niego głównie z uwagi na ograniczenia technologiczne, ale współcześnie nie ma takiego problemu, więc nie będziemy korzystać z tego przybliżenia i wektory będą generowane zgodnie z założeniami. Oznacza to, że wektor losowy \vec{g}_i jest postaci:

$$\vec{g}_i = [\cos(2\pi U_i), \sin(2\pi U_i)], \text{ gdzie } U_i \sim \mathcal{U}(0, 1), (U_i) - \text{ciąg zmiennych IID.}$$

Algorytm 1 przedstawia, w jaki sposób dla tablicy wektorów losowych wygenerować wartość dla punktu o współrzędnych (x, y) . Przyjmujemy, że bok kwadratu w siatce ma długość 1.

Metoda ta jest szybka i prosta w implementacji, lecz z samych kroków algorytmu nie widać znaczenia procedury. Aby lepiej to zrozumieć, poniżej szczegółowo opisano, co dzieje się w kolejnych etapach.

Przed rozpoczęciem wykonywania samego algorytmu, należy przygotować tablicę wektorów losowych g_{nm} , których postać uprzednio zdefiniowaliśmy. Dla tablicy wielkości $N \times M$ odpowiadają one wektorom na siatce kwadratowej wielkości $N - 1 \times M - 1$. Wizualizację takiej siatki wraz z realizacją tablicy wektorów losowych przedstawiono na Rysunku 3. Punkty, dla których możemy wygenerować wartości szumu, muszą znajdować się w zbiorze $[0, N) \times [0, M)$.

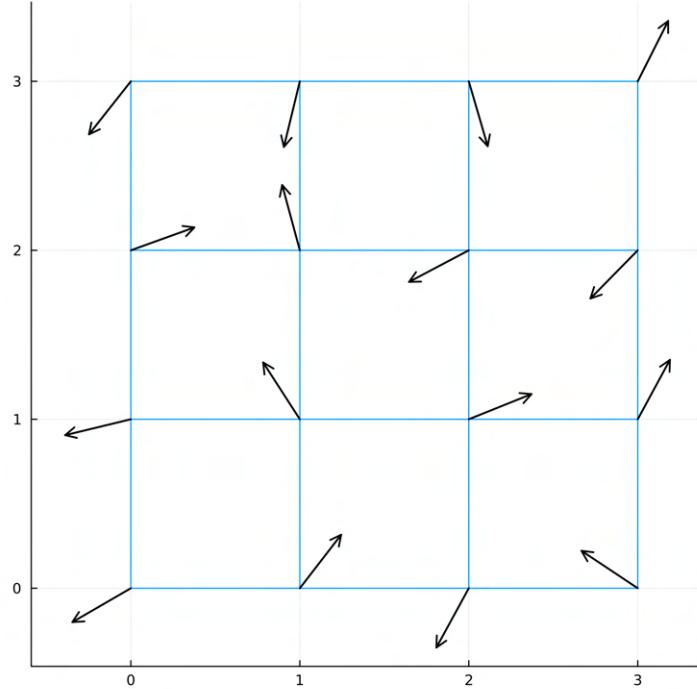
Algorytm 1 Szum Perlina

Dla tablicy wektorów $\vec{g}_{nm} \forall n, m \in \mathbb{Z}$ umieszczonych w wierzchołkach siatki kwadratowej, dla punktu $(x, y) \in \mathbb{R}^2$:

1. oblicz: $i = \lfloor x \rfloor, j = \lfloor y \rfloor,$
2. oblicz: $\Delta x = x - i, \Delta y = y - j,$
3. oblicz: $n_{ij} = \vec{g}_{ij} \cdot [\Delta x, \Delta y], n_{ij+1} = \vec{g}_{ij+1} \cdot [\Delta x, \Delta y - 1], n_{i+1j} = \vec{g}_{i+1j} \cdot [\Delta x - 1, \Delta y], n_{i+1j+1} = \vec{g}_{i+1j+1} \cdot [\Delta x - 1, \Delta y - 1],$
4. oblicz: $m_1 = (n_{i+1j} - n_{ij})\text{smoothstep}_2(\Delta x) - n_{ij}, m_2 = (n_{i+1j+1} - n_{ij+1})\text{smoothstep}_2(\Delta x) - n_{ij+1},$
5. zwróć: $(m_2 - m_1)\text{smoothstep}_2(\Delta y) - m_1,$

gdzie

$$\text{smoothstep}_2(x) = \begin{cases} 0, & \text{gdy } x < 0, \\ 6x^5 - 15x^4 + 10x^3, & \text{gdy } 0 \leq x < 1, \\ 1, & \text{gdy } x \geq 1. \end{cases}$$

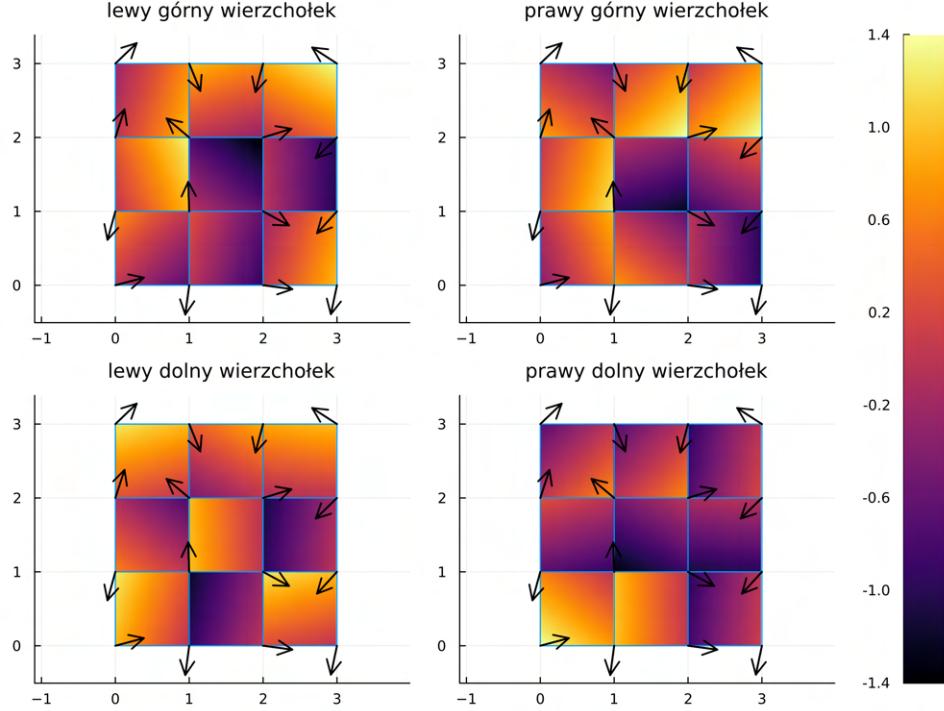


Rysunek 3: Przykładowa siatka z losowymi wektorami. Siatka jest wielkości 3×3 z komórkami o boku długości 1, a wektory są przeskalowane przez 0.4, aby rysunek był bardziej przejrzysty.

W pierwszym kroku algorytmu liczymy część całkowitą każdej współrzędnej. W ten sposób lokalizowana jest komórka siatki, w której znajduje się rozważany punkt (x, y) (i i j to współrzędne lewego dolnego wierzchołka komórki). Kolejnym krokiem jest policzenie wektora przesunięcia $[\Delta x, \Delta y]$ punktu (i, j) i punktu (x, y) .

W kroku trzecim wyliczamy iloczyny skalarne $n_{ij} n_{ij+1} n_{i+1j}$ i n_{i+1j+1} pomiędzy wektorami przesunięcia między punktem (x, y) , a wierzchołkami komórki, w której znajduje się ten punkt z wektorem losowym umieszczonym w tym wierzchołku. Z uwagi na to, że algorytm działa na siatce stworzonej z kwadratów

o boku długości 1, to do wyliczenia wektorów przesunięcia dla wszystkich wierzchołków wystarczy nam wektor $[\Delta x, \Delta y]$. Mamy wtedy dla lewego górnego wierzchołka wektor $[\Delta x, \Delta y - 1]$, dla prawego dolnego wektor $[\Delta x - 1, \Delta y]$ i dla prawego górnego wektor $[\Delta x - 1, \Delta y - 1]$. Na Rysunku 4 przedstawiono wyniki wyliczenia odpowiednich iloczynów skalarnych dla wcześniej wygenerowanej siatki z wektorami losowymi. Jak widać, wartości w poszczególnych komórkach wykazują zależność liniową od x i y oraz zawsze przyjmują wartość 0 dla (x, y) dokładnie w wierzchołku siatki, z którym związana jest wartość.



Rysunek 4: Iloczyny skalarne odpowiednich wektorów przesunięć i wektorów losowych w wierzchołkach siatki. Lewy górny wierzchołek to n_{ij+1} , prawy górny to $n_{i+1,j} + 1$, lewy dolny to n_{ij} i prawy dolny to $n_{i+1,j}$.

Ostatnie dwa kroki polegają na interpolowaniu między wcześniejszymi policzonymi iloczynami skalarnymi. W kroku piątym interpolujemy wartości wzdłuż osi x, czyli interpolujemy między wartościami n_{ij} i $n_{i+1,j}$ otrzymując m_1 oraz między wartościami n_{ij+1} i $n_{i+1,j+1}$ otrzymując m_2 . To, jak duży wpływ na wartość wynikową mają n_{ij} i n_{ij+1} określa Δx , czyli odległość wzdłuż osi x od (x, y) do wierzchołków z lewej strony komórki, natomiast wielkość wpływu $n_{i+1,j}$ i $n_{i+1,j+1}$ określa $1 - \Delta x$. Kolejnym krokiem jest interpolacja otrzymanych wartości, ale tym razem wzdłuż osi y, otrzymując tym sposobem wartość szumu Perlina. Wielkości wpływu m_1 i m_2 ustalone są analogicznie jak w poprzednim kroku, ale wzdłuż osi y. Taka interpolacja sprawia, że losowe wektory z siatki mają wpływ tylko na komórki siatki, z którymi sąsiadują.

Proces interpolacji korzysta z funkcji smoothstep_n , którą definiujemy jako:

$$\text{smoothstep}_n(x) = \begin{cases} 0, & \text{gdy } x < 0, \\ S_n(x), & \text{gdy } 0 \leq x < 1, \\ 1, & \text{gdy } x \geq 1. \end{cases}$$

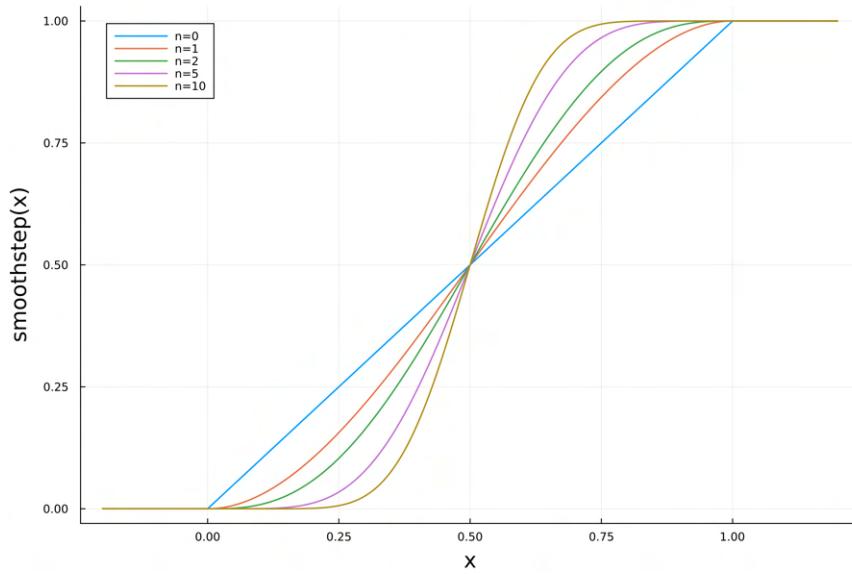
gdzie S_n to wielomian interpolujący między prostą $y = 0$ na przedziale $(-\infty, 0)$, a prostą $y = 1$ na przedziale $(1, \infty)$, w dostatecznie gładki sposób (tak, aby funkcja smoothstep była klasy C^n). Wielomian S_n spełnia takie założenia, jeśli:

$$\begin{cases} S_n(0) = 0 \\ S_n(1) = 1 \\ S'_n(0) = 0 \\ S'_n(1) = 0 \\ \vdots \\ S_n^{(n)}(0) = 0 \\ S_n^{(n)}(1) = 0 \end{cases}$$

Aby taki układ równań był jednoznacznie rozwiązywalny, wielomian S_n musi być rzędu $2n+1$. Rozwiążując dla dowolnego n otrzymujemy wielomian postaci:

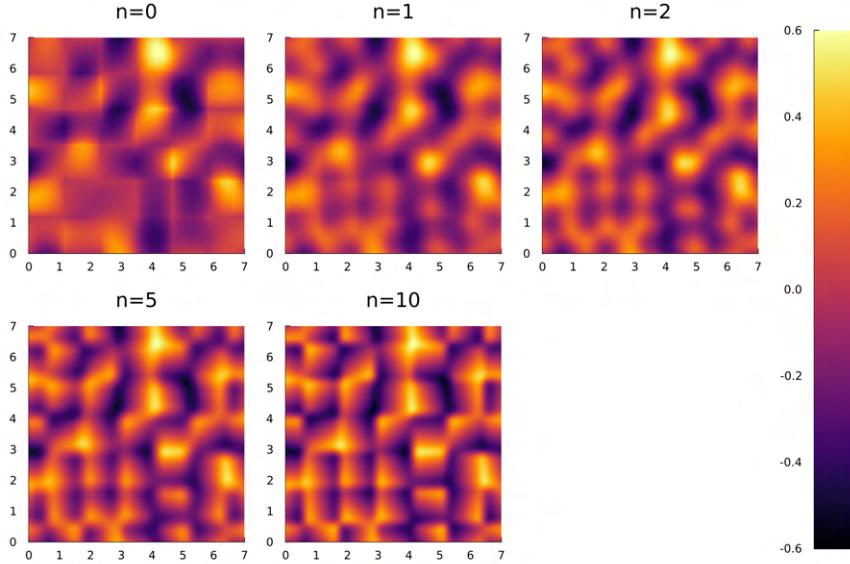
$$S_n = x^{n+1} \sum_{k=0}^n \binom{n+k}{k} \binom{2n+1}{n-k} (-x)^k.$$

Interpolacja takim wielomianem jest szczególnym przypadkiem interpolacji Hermite'a [9]. Na Rysunku 5 przedstawiono wygląd funkcji smoothstep_n dla różnych n . Możemy zauważać, że dla $n = 0$ dostajemy funkcję interpolującą liniowo, co oznacza, że interpolację funkcją smoothstep_n można potraktować jako uogólnienie interpolacji liniowej, które wygładza otrzymaną krzywą.



Rysunek 5: Funkcja smoothstep_n dla $n = 0, 1, 2, 5, 10$.

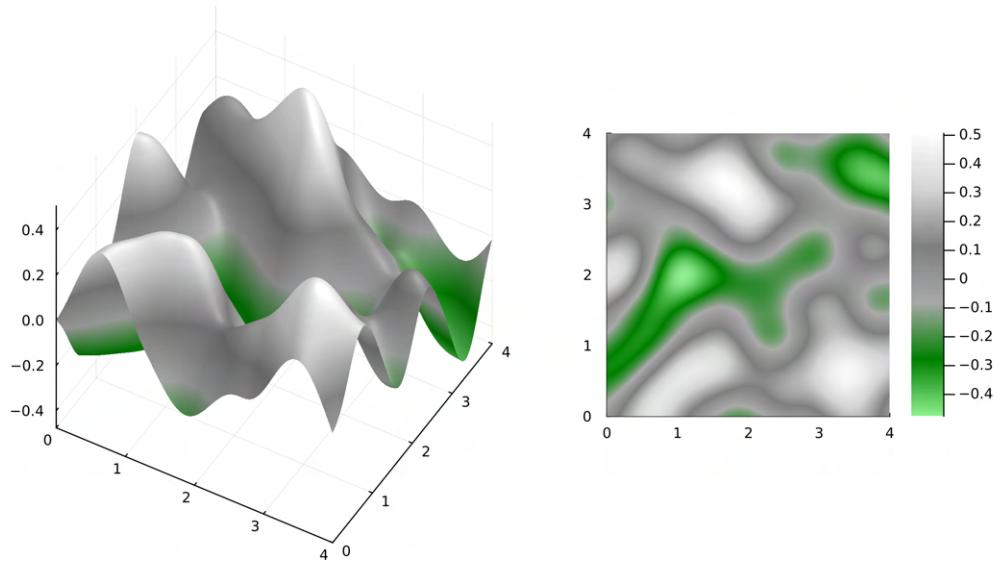
W oryginalnym algorytmie Ken Perlin zaproponował użycie funkcji smoothstep_2 , ale można użyć interpolacji innych rzędów. Na Rysunku 6 przedstawiono szum Perlina dla tej samej tablicy wektorów losowych, ale różnych rzędów funkcji interpolującej. Widać, że interpolacja liniowa nie zwraca wizualnie gładkiego szumu, a dla funkcji smoothstep_n wyższego rzędu bardzo wyraźnie zaczynają być artefakty związane z tym, że algorytm odbywa się na siatce kwadratowej. Dla $n = 1$ i $n = 2$ szумy wyglądają dosyć podobnie i wizualnie gładko, ale dla $n = 2$ wynik wygląda ostrzej. Warto również dodać, że w trakcie działania algorytmu argumenty funkcji smoothstep_n zawsze będą z przedziału $[0, 1]$, więc w implementacji wystarczy wykorzystać wielomian S_n .



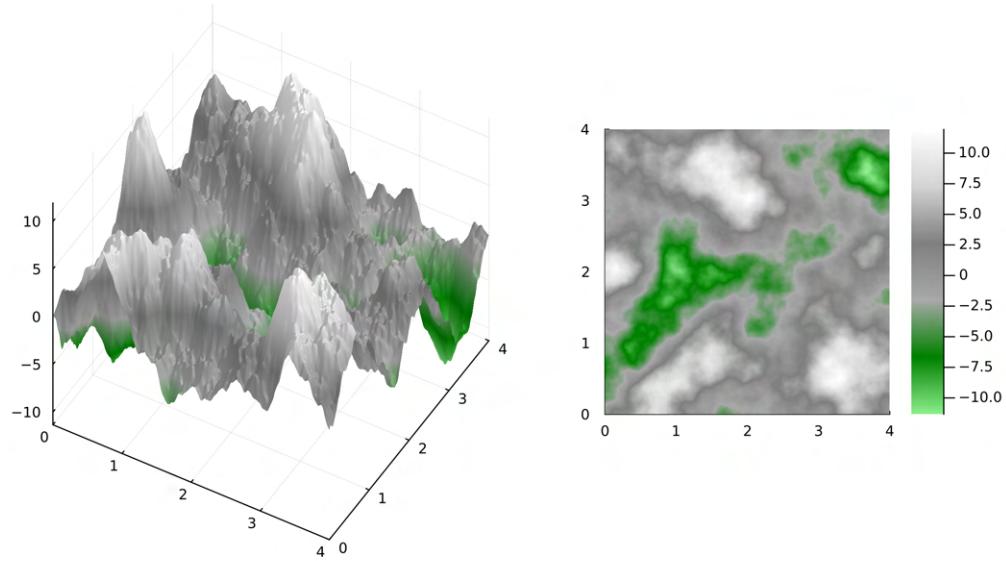
Rysunek 6: Szum Perlina dla funkcji interpolującej rzędu $n = 0, 1, 2, 5, 10$.

W zastosowaniach wygenerowanie jednej realizacji szumu zwykle nie jest wystarczające. Wynika to z tego, że skala i szczegółowość tego szumu zależą od wielkości komórek siatki. Oznacza to, że tak wygenerowana struktura opisująca wysokość terenu będzie nienaturalnie gładka oraz pozbawiona małych wzniesień i nieregularności, które w przyrodzie występują. Widać to na Rysunku 7a, który ma imitować teren górski. Aby urozmaicić taki teren, generuje się więcej realizacji szumu Perlina, ale zmniejsza się wielkość komórek siatki. Następnie dodaje się do siebie odpowiednio przeskalowane wygenerowane realizacje. Skala szumów o mniejszej wielkości komórki zwykle jest mniejsza, aby nie zdominowały szumu, a dodały szczegóły do wyniku. Sumę tych pól losowych nazywa się fraktalnym szumem Perlina, co wynika z wieloskalowości otrzymanego szumu, czyli samopodobieństwa własności statystycznych i geometrycznych na obszarach o różnych wielkościach.

O realizacjach szumu Perlina można myśleć jak o sygnale, a o operacji dodawania kolejnych szumów (nazywanych oktawami), jak o dodawaniu sygnałów o rosnących częstotliwościach i malejących amplitudach. Na Rysunku 7b widać szum wygenerowany taką metodą. Stworzono pięć szumów dla siatek kolejno: 4×4 , 8×8 , 16×16 , 32×32 i 64×64 oraz przemnożono je przez amplitudy kolejno: 20, 5, 3, 2, 0.5. Jest on wyraźnie bardziej szczegółowy i rzeczywiście przypomina teren górski.



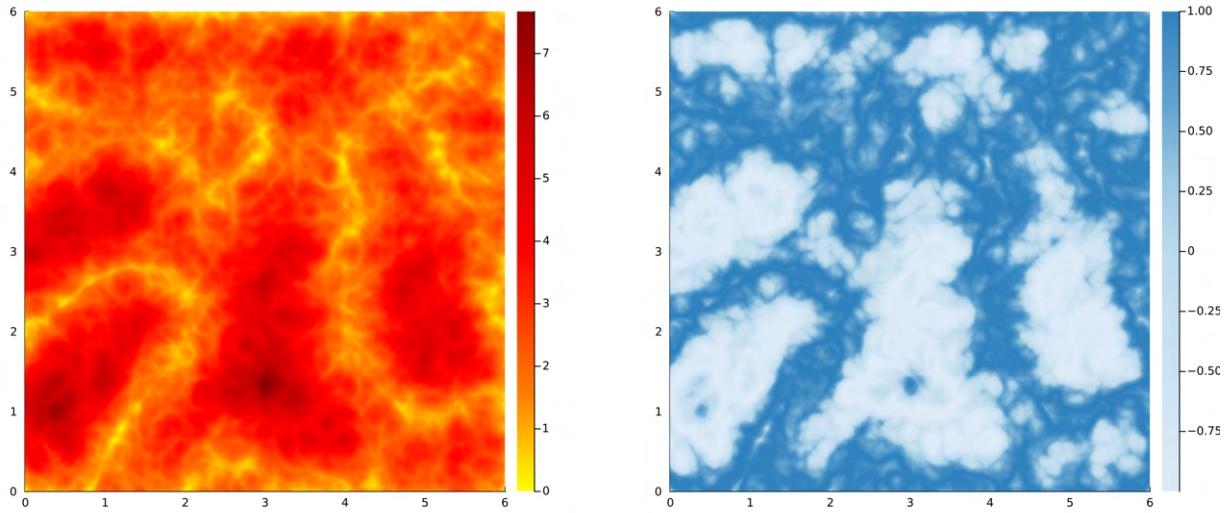
(a) Pojedyncza oktawa szumu Perlina.



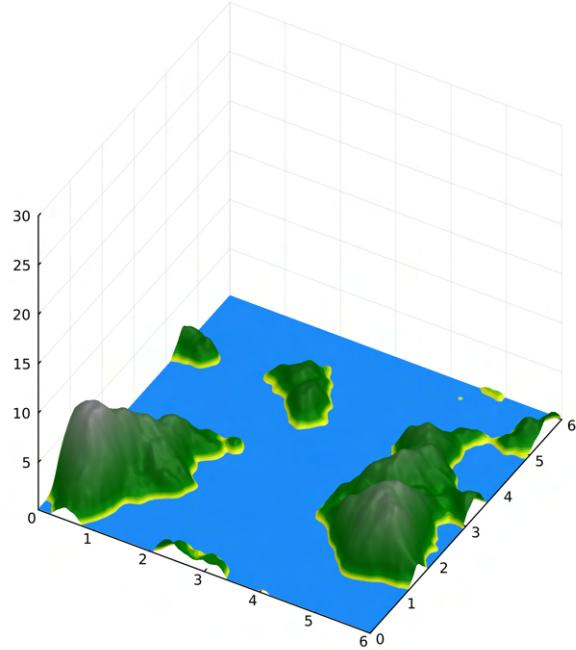
(b) Suma oktaw szumu Perlina.

Rysunek 7: Wykres powierzchniowy oraz mapa cieplna szumu fraktalnego mające przypominać teren górski.

Często rozważane są również różne przekształcenia szumu. Przykładowo biorąc sumę wartości bezwzględnych z odpowiednio przeskaliwanych oktaf szumu Perlina, dostajemy wynik, który może, przy odpowiednim doborze kolorów, przypominać ogień (Rys. 8a), a biorąc sinus takiej sumy, otrzymujemy szum mogący przypominać chmury (Rys. 8b). Możemy również rozważyć obcięcie wartości szumu, które da nam teren z wyspami (Rys. 8c). Wszystkie trzy omówione przykłady stworzono za pomocą dokładnie tych samych realizacji szumu, zmieniając tylko amplitudy poszczególnych oktaf, korzystając z różnych przekształceń oraz innych kolorów do wizualizacji.



(a) Fraktalny szum z wartości bezwzględnych szumu Perlina. (b) Sinus fraktalnego szumu z wartości bezwzględnych szumu Perlina.



(c) Obcięcie fraktalnego szumu Perlina.

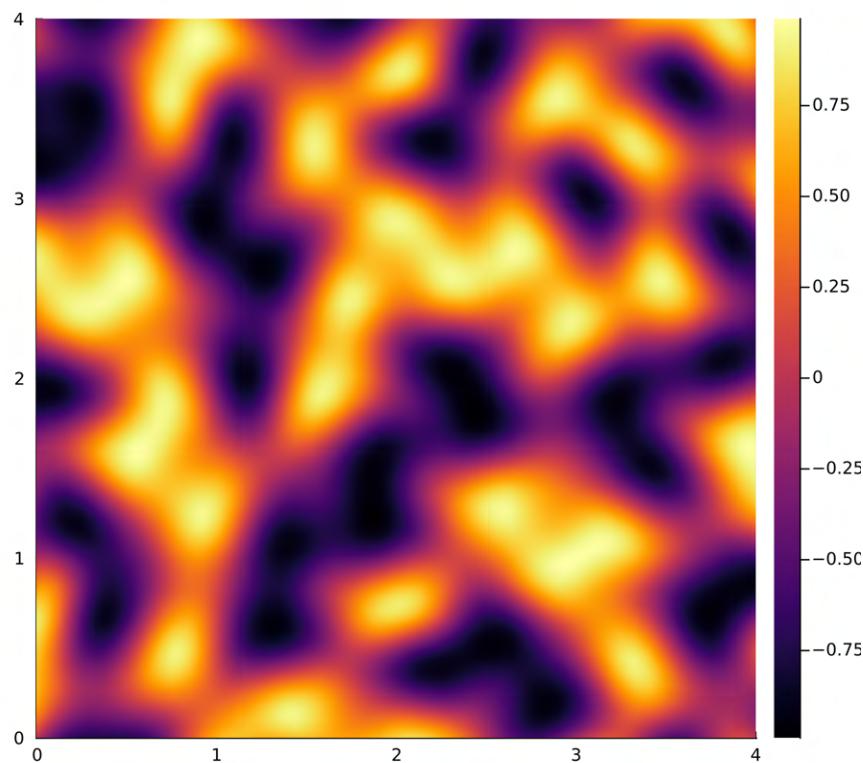
Rysunek 8: Przykładowe przekształcenia szumu i efekty jakie można nimi osiągnąć.

2.2 Szum Simplex

Kolejnym omawianym szumem jest szum Simplex [12], który również został zaprojektowany przez Kena Perliniego. Metoda opublikowana została w 2001 roku i miała być ulepszeniem szumu Perlina. Jest wydajniejsza, a szum ma mniej widocznych artefaktów kierunkowych. Algorytm ten, tak samo jak w przypadku poprzednio

omawianego szumu, generuje pole, które jest wizualnie gładkie. Używa się go najczęściej w dwu-, trzy- lub czterowymiarowych przestrzeniach. Ponownie skupimy się tylko na przypadku dwuwymiarowym. Na Rysunku 9 można zobaczyć przykładową realizację szumu Simplex. Wizualnie wynik jest bardzo podobny do poprzedniej metody.

Metoda ta podobnie jak szum Perlina bazuje na siatce, ale w tym przypadku jest to siatka stworzona z trójkątów równobocznych. Ponownie w wierzchołkach siatki umieszczane są wektory losowe. Algorytm 2 przedstawia, jak dla tablicy wektorów losowych wygenerować wartość szumu w punkcie (x, y) .



Rysunek 9: Realizacja szumu Simplex.

Algorytm 2 Szum Simplex

Dla tablicy wektorów $\vec{g}_{nm} \forall n, m \in \mathbb{Z}$ umieszczonych w wierzchołkach siatki trójkątnej, dla punktu $(x, y) \in \mathbb{R}^2$:

1. oblicz: $s = (x + y) \frac{\sqrt{3}-1}{2}$,
 2. oblicz: $i = \lfloor x + s \rfloor, j = \lfloor y + s \rfloor$,
 3. oblicz: $t = (i + j) \frac{\sqrt{3}-3}{6}$,
 4. oblicz: $x_0 = i + t, y_0 = j + t$,
 5. oblicz: $\Delta x_0 = x - x_0, \Delta y_0 = y - y_0$,
 6. jeśli $\Delta y_0 > \Delta x_0$
 $\Delta i = 0, \Delta j = 1$
jeśli $\Delta y_0 \leq \Delta x_0$
 $\Delta i = 1, \Delta j = 0$,
 7. oblicz: $\Delta x_1 = \Delta x_0 - \Delta i - \frac{\sqrt{3}-3}{6}, \Delta y_1 = \Delta y_0 - \Delta j - \frac{\sqrt{3}-3}{6}, \Delta x_2 = \Delta x_0 - 1 - \frac{\sqrt{3}-3}{3}, \Delta y_2 = \Delta y_0 - 1 - \frac{\sqrt{3}-3}{3}$,
 8. znajdź: $\vec{v}_0 = \vec{g}_{ij}, \vec{v}_1 = \vec{g}_{i+\Delta i j+\Delta j}, \vec{v}_2 = \vec{g}_{i+1 j+1}$
 9. dla $k = 0, 1, 2$ oblicz t_k postaci: $t_k = 0.5 - \Delta x_k^2 - \Delta y_k^2$,
 10. dla $k = 0, 1, 2$ znajdź n_k postaci: $n_k = t_k^4 \vec{v}_k \cdot [\Delta x_k, \Delta y_k] \mathbb{1}_{\{t_k \geq 0\}}$,
 11. zwróć: $100(n_0 + n_1 + n_2)$
-

Algorytm ten jest bardziej złożony niż poprzedni, ale wciąż jest prosty w implementacji oraz jest nawet szybszy do generowania niż szum Perlina. Poniżej przedstawiono wyjaśnienie kolejnych kroków omawianej metody.

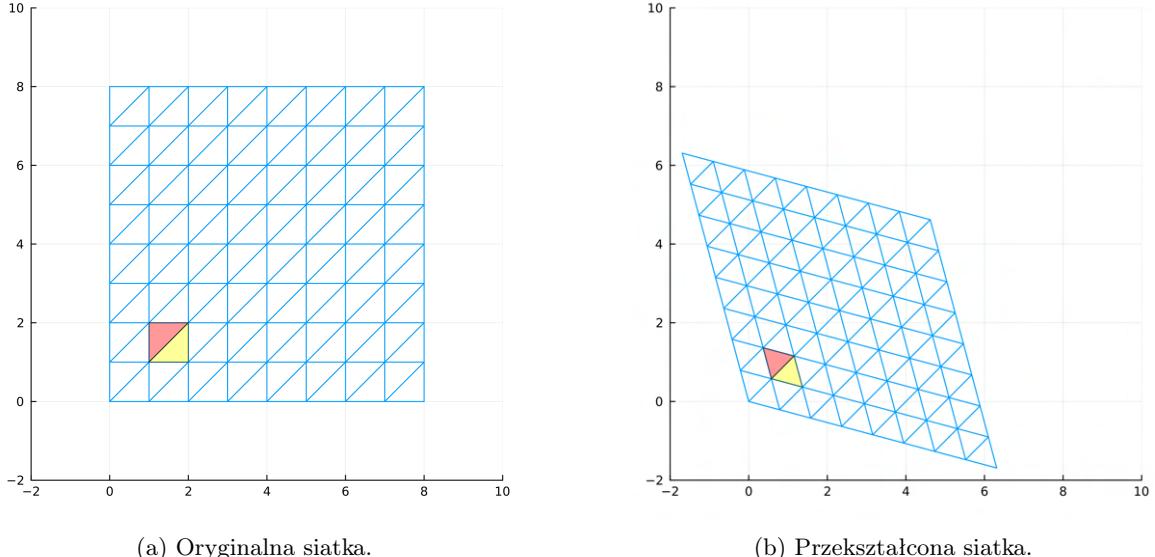
Tablica wektorów losowych \vec{g}_{nm} ponownie jest wymagana jeszcze przed rozpoczęciem algorytmu. Tym razem jednak trudniej jest znaleźć komórkę siatki, w której leży dowolny punkt (x, y) , ponieważ siatka trójkątna jest bardziej złożona niż siatka kwadratowa, która używana była podczas generowania szumu Perlina. Siatka zaprojektowana jest specjalnie, tak aby ten proces uprościć. Konstrukcja jej polega na rozpoczęciu z siatką stworzoną z trójkątów prostokątnych równoramiennych ustawionych tak, aby współczynnik nachylenia przeciwprostokątnych był równy 1. Siatka ta jest zbliżona do siatki kwadratowej, ale z dodatkowym podziałem każdego kwadratu na dwa trójkąty. Następnie traktując boki siatki jako wektory w \mathbb{R}^2 możemy wykonać przekształcenie liniowe przestrzeni wektorowej [1] w taki sposób, aby przekształcona siatka była stworzona z trójkątów równobocznych. Warto również zauważyć, że kwadraty z początkowej siatki przekształcają się w romby. Na Rysunku 10 widać, jak oryginalna siatka przekształcana jest w docelową. Aby otrzymać taką siatkę, należy rozważyć bazę złożoną z wektorów nachylonych -15° ($-\frac{\pi}{12}$) i 105° ($\frac{7\pi}{12}$) do osi x. Oznacza to, że będą one postaci:

$$u'_1 = c \cdot \left[\cos\left(-\frac{\pi}{12}\right), \sin\left(-\frac{\pi}{12}\right) \right] = \frac{c}{2\sqrt{2}} \left[1 + \sqrt{3}, 1 - \sqrt{3} \right],$$

$$u'_2 = c \cdot \left[\cos\left(\frac{7\pi}{12}\right), \sin\left(\frac{7\pi}{12}\right) \right] = \frac{c}{2\sqrt{2}} \left[1 - \sqrt{3}, 1 + \sqrt{3} \right],$$

gdzie $c = \text{const.}$

Korzystając z przekształcenia przestrzeni wektorowej z bazą z wektorami $u_1 = [1, 0], u_2 = [0, 1]$ na przestrzeń z bazą z wektorami u'_1, u'_2 (lub na odwrót), dla $c = 4\sqrt{6}$ jesteśmy w stanie zmapować punkt z przedstawionej siatki regularnej trójkątnej na siatkę kwadratową za pomocą przekształcenia z kroku 1 oraz z siatki kwadratowej na siatkę regularną trójkątną, korzystając z przekształcenia w kroku 3.

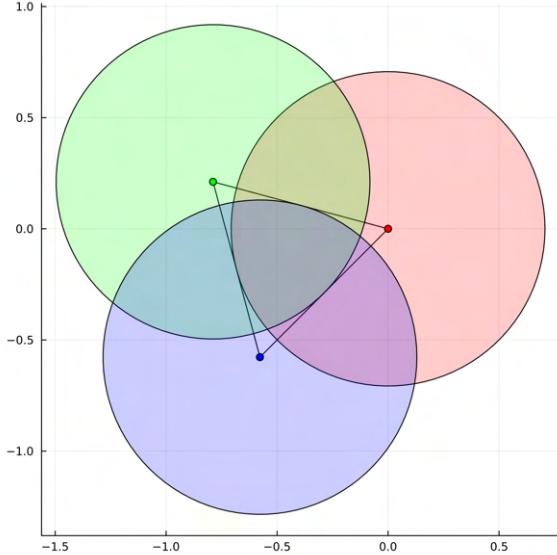


Rysunek 10: Przekształcenie siatki trójkątów prostokątnych na siatkę trójkątów równobocznych.

W kroku pierwszym i drugim algorytmu przekształcamy punkt (x, y) z siatki równobocznej na siatkę, z której łatwo możemy odczytać, w jakiej komórce znajduje się ten punkt. Robimy to, biorąc podлогę, co daje nam współrzędne (i, j) , które mówią o położeniu lewego dolnego wierzchołka kwadratu, w którym jest szukany punkt. W krokach 3 i 4 wracamy ze współrzędnymi (i, j) do przestrzeni z siatką równoboczną za pomocą przekształcenia odwrotnego do tego wykonanego w kroku 1 i 2. Dzięki temu znamy współrzędne (x_0, y_0) lewego dolnego wierzchołka rombu siatki, ale nie wiemy, w którym z dwóch trójkątów zawartych w rombie znajduje się (x, y) . Szukamy zatem w jakiej odległości wzduż osi x i y znajduje się rozważany punkt od znalezionej wierzchołka siatki (krok 5.). Jeśli odległość wzduż osi y jest większa, to znaczy, że (x, y) jest w górnym trójkącie, a w przeciwnym razie jest w dolnym trójkącie (krok 6.).

W kroku 7. znajdujemy wektory przesunięć $[\Delta x_1, \Delta y_1], [\Delta x_2, \Delta y_2]$ między punktem (x, y) , a pozostały mi dwoma wierzchołkami znalezionej trójkąta. Z konstrukcji siatki znamy długości boków oraz nachylenie każdego trójkąta, więc wyliczenie tych wektorów przesunięć nie wymaga przekształcania odpowiednich współrzędnych z siatki prostokątnej do siatki równobocznej, a tylko dodania odpowiednich wartości do wektora $[\Delta x_0, \Delta y_0]$.

Następnie w kroku 9. wyliczane są wartości t_0, t_1, t_2 , które odpowiadają za wpływ wektorów losowych z wierzchołków. Wylicza się je jako różnicę kwadratów wysokości trójkąta siatki równego $\frac{1}{2}$ i długości wektorów przesunięcia od wierzchołków trójkąta. W kroku 10. sprawdzamy, czy dane wartości są mniejsze niż 0. Jeśli któreś są, oznacza to, że wartości wektorów losowych z odpowiadającymi im wierzchołków trójkąta nie będą uwzględniane w ostatecznym wyniku ($n_k = 0$). W przeciwnym przypadku wyliczane są iloczyny skalarnie między wektorami losowymi a wektorami przesunięcia i mnożone są przez odpowiadające im wartości t_k^4 , czyli $n_k = t_k^4 g_{i_k j_k} \cdot [\Delta x_k, \Delta y_k]$, dla $k = 0, 1, 2$ będącego numerem wierzchołka oraz i_k, j_k , będącymi indeksami zależnymi od wierzchołka. Taka konstrukcja wyliczania istotności wartości z danego wierzchołka wynika z tego, że chcemy, aby szum był gładki, zatem wpływ z wierzchołka musi być równy 0 na boku przeciwegłym do niego. Sytuacja ta przedstawiona jest na Rysunku 11, gdzie okręgi o środku w danym wierzchołku odpowiadają za jego „obszar wpływu”. Dodatkowo wpływ ten maleje jak ósma potega odległości od wierzchołka. Ostatecznym wynikiem jest suma odpowiednio wyważonych iloczynów skalarnych pomnożona przez 100, aby wynik znajdował się w przedziale $[-1, 1]$.



Rysunek 11: „Obszar wpływu” wektorów losowych z danych wierzchołków trójkąta.

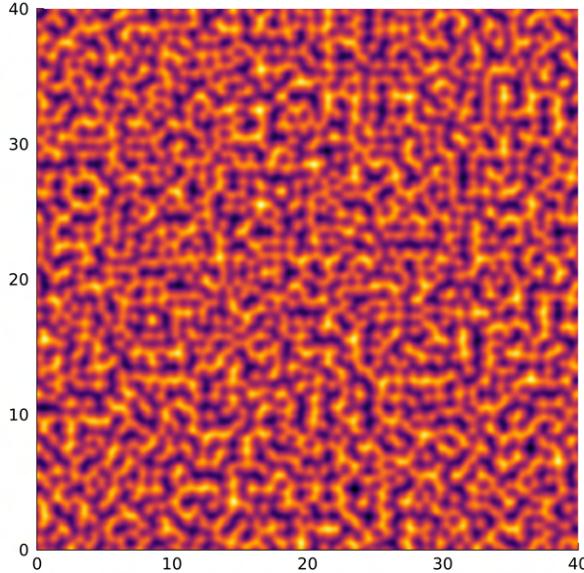
Tak samo jak w przypadku szumu Perlina, zwykle stosuje się sumę szumów o różnych częstotliwościach i amplitudach. Rozważa się również przekształcenia realizacji pola losowego, a wyniki ich są podobne do przykładów z Rysunków 8a, 8b, 8c.

3 Analiza klasycznych szumów

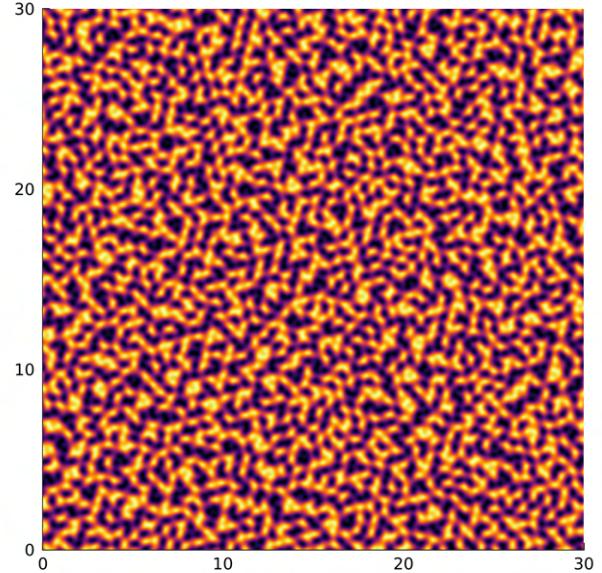
Dużymi problemami szumu Perlina są artefakty kierunkowe związane z siatką kwadratową oraz brak izotropii [12], czyli niezmienniczości względem rotacji. Przez to wynik nie wygląda naturalnie, co jest bardzo istotne, gdy chcemy generować dobrze wyglądające uksztaltowanie terenu. W celu ich poprawienia zaproponowany został szum Simplex, który rzeczywiście działa lepiej pod tymi względami, ale nie pozbywa się całkowicie tych niedoskonałości. Artefakty siatek, na których bazują omawiane pola losowe, są najbardziej widoczne, gdy wygenerujemy wartości szumu dla stosunkowo dużego wycinka przestrzeni. Widać to na Rysunku 12, gdzie w przypadku szumu Perlina wyraźnie widać miejscowe zagęszczenia, które wizualnie układają się w linie nachylone pod kątem 0° , 45° i 90° . Jest to na tyle wyraźny efekt, że można go zobaczyć bez konieczności oglądania tak dużego wycinka szumu, co widać na Rysunku 2. Wynika on między innymi z faktu, że siatka kwadratowa posiada symetrię obrotową rzędu 4 (oznacza to, że przy obrocie o $360^\circ/4 = 90^\circ$ otrzymamy identyczną siatkę) [19]. Dodatkowym problemem okazuje się metoda interpolacji, która mocno bazuje na ksztalcie kwadratu.

W przypadku szumu Simplex nie jest to, aż tak widoczne z uwagi na strukturę siatki, której rząd symetrii obrotowej jest równy 6 (siatka ta opiera się na siatce sześciokątnej), dzięki czemu powstające zagęszczenia będą mogły być skierowane w większą ilość kierunków. Pomaga tutaj również metoda interpolacji, która bazuje na obszarach w kształcie kół, które same w sobie są izotropowe.

W celu ogólnego zbadania omawianych pól losowych oraz sformalizowania problemów związanych z metodami, wykonano analizę generowanych szumów.



(a) Szum Perlina.



(b) Szum Simplex.

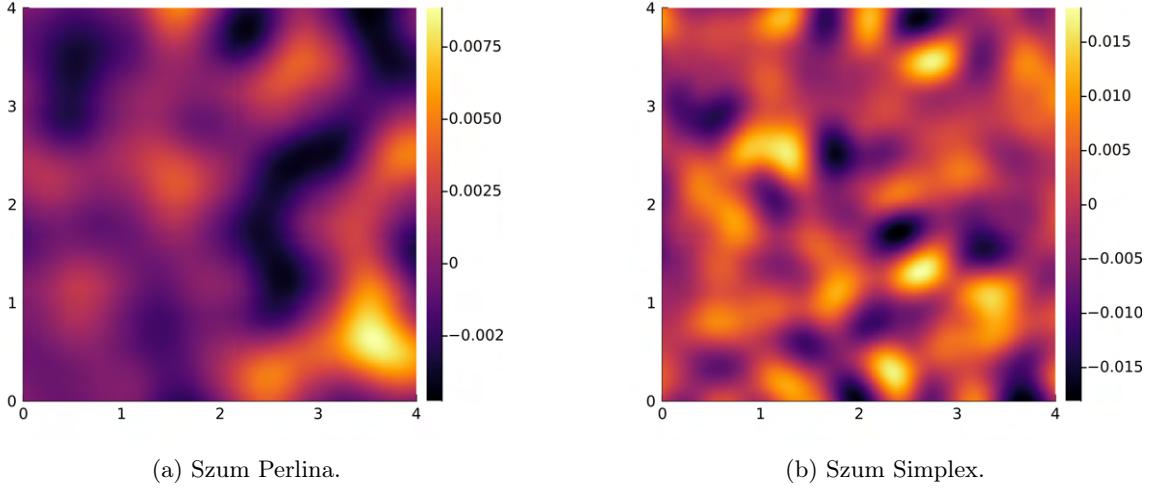
Rysunek 12: Realizacje omawianych szumów na dużych obszarach.

3.1 Analiza statystyczna

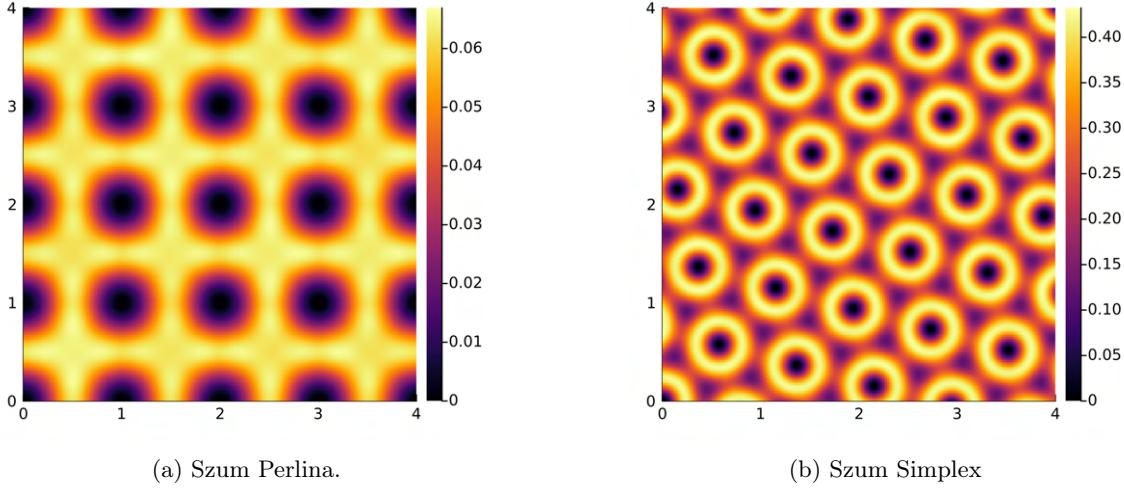
Pierwsza część analizy wykonano symulując 10000 realizacji szumów i wyliczając empirycznie ich podstawowe cechy statystyczne. Szumy generowane były dla argumentów ze zbioru $[0, 4] \times [0, 4]$, biorąc 256×256 równooddalonych punktów. Wykonano wszystko na siatce kwadratowej o długości boku komórki równej 1. W przypadku szumu Simplex na takim samym obszarze siatka jest gęstsza, z uwagi na jej konstrukcję.

Pierwszą cechą, którą zbadano, jest średnia szumu. Wyniki przedstawiono na Rysunku 13. Widać na nim, że średnia obu szumów zbliża się do 0, ale zachowują one pewną strukturę, którą przypomina oryginalne szumy. W obu prawdziwa wartość średniej jest równa 0, z uwagi na symetrię i ograniczoną wynikowego szumu. Za pomocą testu t-Studenta [17] sprawdzono dla każdego punktu hipotezę, czy jego średnia jest równa 0 przeciwko hipotezie, że średnia nie jest równa 0. Wyniki testów wykazały, że znalezione wartości średniej nie są na tyle różne od 0, aby być istotne statystycznie.

Wyliczono również wariancję, której wartości widać na Rysunku 14. Można zauważyc okresowość i regularność związaną z siatką, z pomocą której generowane są pola losowe. W obu przypadkach w wierzchołkach siatki wariancja równa jest 0. Wynika to z tego, że licząc iloczyn skalarny wektora $[0, 0]$ (wektora przesunięcia punktu względem wierzchołka siatki) z dowolnym innym wektorem dostaniemy wartość stale równą 0. Dzięki otrzymanemu wynikowi widać również charakter działania interpolacji. W przypadku szumu Perlina jest zauważalnie „kanciasta”, czego nie można powiedzieć o wyniku dla szumu Simplex. Widać w nim natomiast wcześniej omawiane koliste obszary wpływu danego wierzchołka siatki. Dodatkowo można zauważyc pewne przerwy niebędące w wierzchołkach siatki, dla których wartość wariancji jest mała. Nie jest to problem w omawianym dwuwymiarowym przypadku, natomiast podczas generowania tego szumu dla przestrzeni o większej liczbie wymiarów te przerwy stają się na tyle duże, że wariancja omawianego szumu jest równa, bądź bardzo bliska 0 dla dużej części przestrzeni. Oznacza to, że takie wielowymiarowe pole losowe w znaczącej części będzie stale równe 0, przez co wynik nie będzie wyglądał dobrze.



Rysunek 13: Średnia badanych szumów.



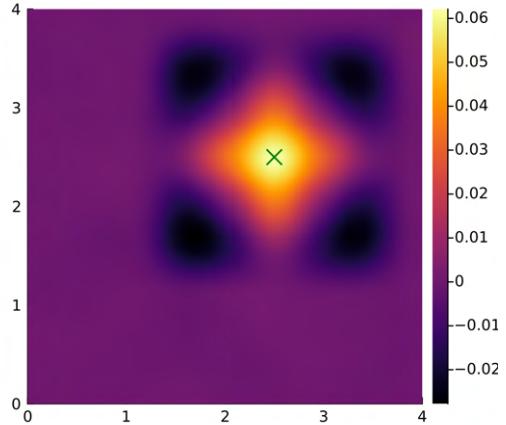
Rysunek 14: Wariancja badanych szumów.

Na Rysunkach 15 i 16 przedstawiono kolejno funkcję autokowariancji γ i autokorelacji ρ omawianych szumów. Dla pola losowego T_p , $p \in \mathbb{R}^2$ zdefiniowane są one następująco:

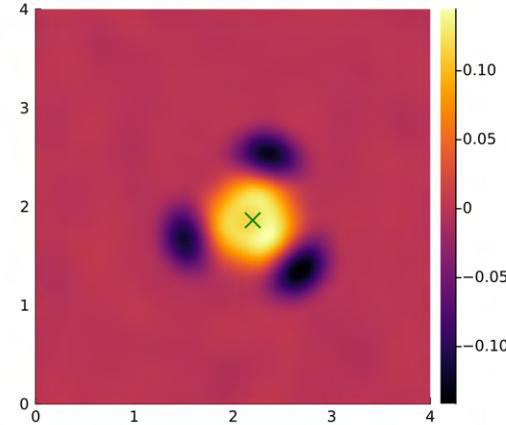
$$\begin{aligned}\gamma(p_0, p_h) &= \text{cov}[T_{p_0}, T_{p_h}] = \mathbb{E}[(T_{p_0} - \mathbb{E}T_{p_0})(T_{p_h} - \mathbb{E}T_{p_h})], \\ \rho(p_0, p_h) &= \frac{\gamma(p_0, p_h)}{\sqrt{\text{Var}[T_{p_0}]\text{Var}[T_{p_h}]}}.\end{aligned}$$

Wynik przedstawia wyestymowaną funkcję autokowariancji i autokorelacji dla ustalonego punktu p_0 (na wykresach zaznaczonym zielonym „x”) i każdego innego punktu p_h rozważanego wycinka przestrzeni. Możemy zauważać pewną okresowość przejawiającą się ujemną zależnością między komórkami siatki. W przypadku szumu Perlina wartości ze środka komórki (Rys. 15a i 16a) mają zauważalnie ujemną zależność dla komórek sąsiadujących ze sobą wierzchołkami. To samo jesteśmy w stanie powiedzieć o szumie Simplex (Rys. 15b i 16b). Gdy rozważymy wartości bliżej wierzchołka komórki (Rys. 15c i 15d), również możemy zauważać ujemną zależność między wartościami z najbliższej komórki siatki. Widać to również z samego algorytmu – odpowiednie iloczyny skalarne są dodatnie z jednej strony, a ujemne z drugiej, przez co zależność jest ujemna. Oczywiście można badać większą liczbę ustalonych punktów (zaznaczonych zielonym „x”), ale badając

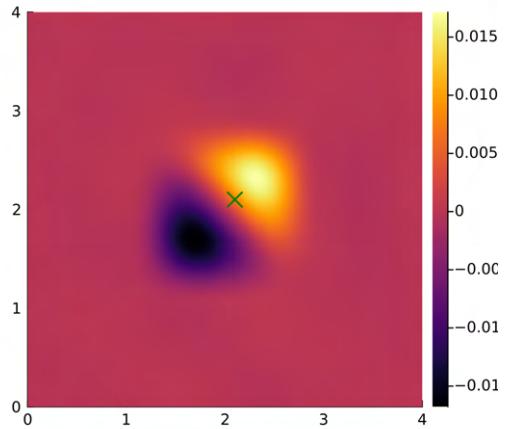
je, wyniki były bardzo podobne do już otrzymanych i nie wniosły żadnych dodatkowych wniosków, więc je pominięto.



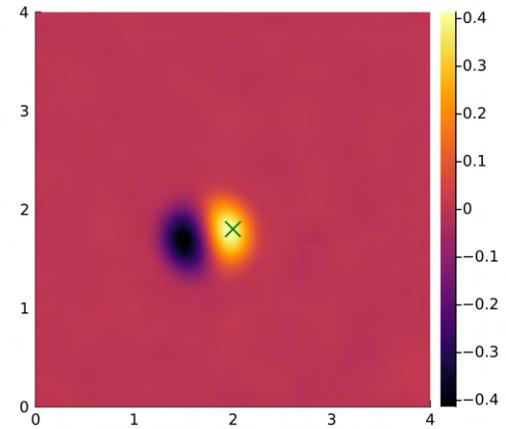
(a) Szum Perlina, środek komórki siatki (2.5, 2.5).



(b) Szum Simplex, środek komórki siatki (1.86, 2.2).

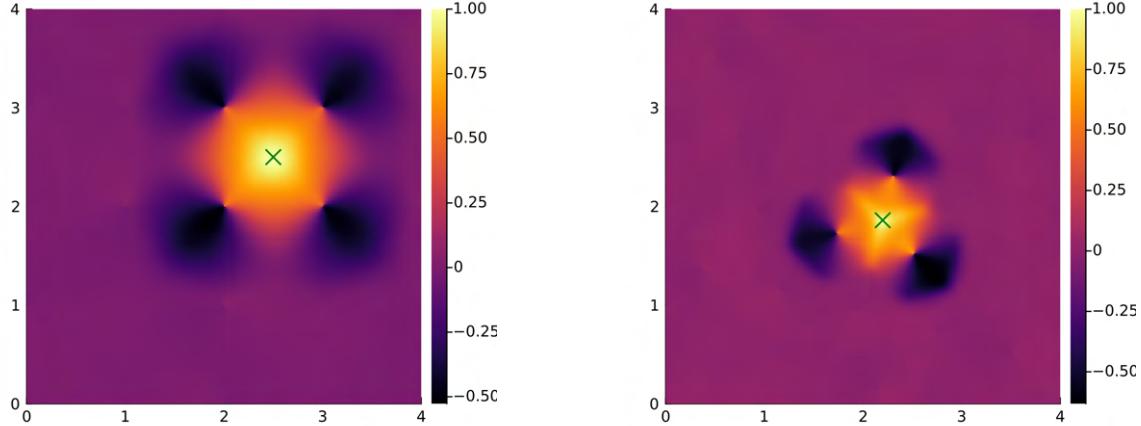


(c) Szum Perlina, przy wierzchołku komórki siatki (2.1, 2.1).



(d) Szum Simplex, przy wierzchołku komórki siatki (1.8, 2).

Rysunek 15: Kowariancja badanych szumów.

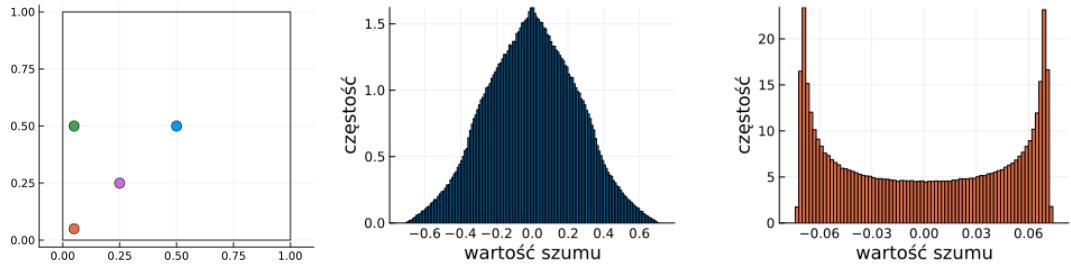


(a) Szum Perlina, środek komórki siatki (2.5, 2.5).

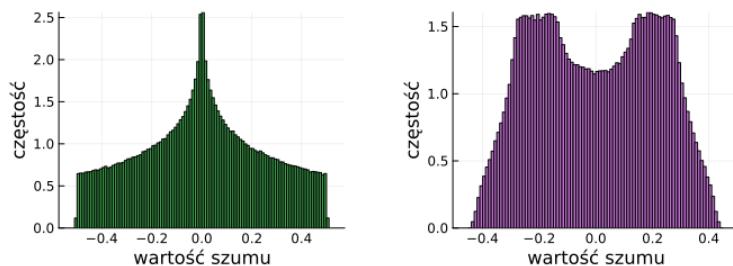
(b) Szum Simplex, środek komórki siatki (1.86, 2.2).

Rysunek 16: Korelacja badanych szumów.

Wyznaczono również histogramy wartości szumów w wybranych punktach (szum Perlina – Rys. 17, szum Simplex – Rys. 18). W tym wypadku nie trzeba rozważać całej siatki, a pojedynczą jej komórkę, z uwagi na to, że rozkład ten jest periodyczny. Nie generowano tym razem realizacji szumu dla wszystkich punktów z obszaru komórki, a wybrano tylko po 4 dla każdej z metod. Z tego powodu zamiast rozważać 10^4 realizacji (jak w przypadku średniej, wariancji, autokowariancji i autokorelacji) wyliczono 10^6 dla każdego punktu, dzięki czemu otrzymane histogramy będą lepiej przybliżały rzeczywistą gęstość punktu pola losowego. W przypadku szumu Perlina zbadano rozkłady dla czterech różnych punktów, których położenie względem całej komórki przedstawiono na Rysunku 17a (kolor punktu odpowiada kolorowi histogramu). Możemy zauważyć, że rozkład pola losowego dla punktu blisko wierzchołka siatki (Rys. 17c) jest dwumodalny, co również widzimy, zbliżając się do środka komórki (Rys. 17). Dla punktów w środku (Rys. 17b) oraz przy boku komórki (Rys. 17d) rozkład nie jest już dwumodalny. W każdym przypadku rozkłady są symetryczne i ograniczone przez różne wartości. Histogramy dla szumu Simplex niezależnie od umiejscowienia badanego punktu są dwumodalne i tak samo jak w poprzednim przypadku są symetryczne i ograniczone różnymi wartościami. Dwumodalność rozkładów można zauważać na realizacjach szumu (Rys. 9 i 12), która przejawia się wyraźnymi i szybkimi przejściami między wartościami ujemnymi a dodatnimi. Oznacza to, że wartości tego szumu w ramach pojedynczej komórki najczęściej przyjmują wartości bliskie wartościom granicznym rozkładu w danym punkcie. Pod względem rozkładów punktowych szum Simplex jest lepszy niż szum Perlina, ponieważ jest on mniej zależny od siatki.

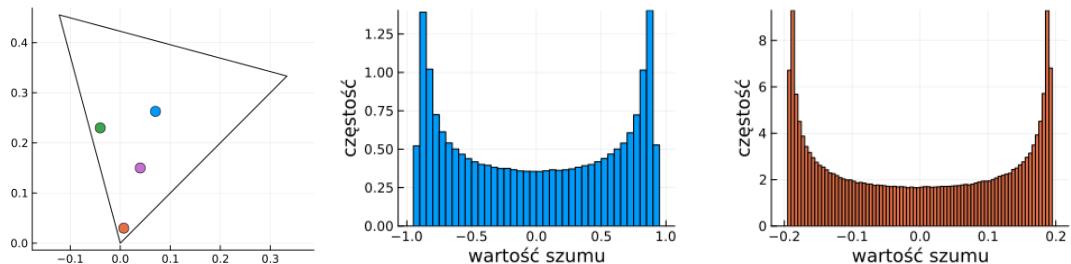


(a) Punkty siatki dla których wyliczono histogram.
 (b) Środek komórki siatki (0.5, 0.5).
 (c) Przy wierzchołku komórki siatki (0.05, 0.05).

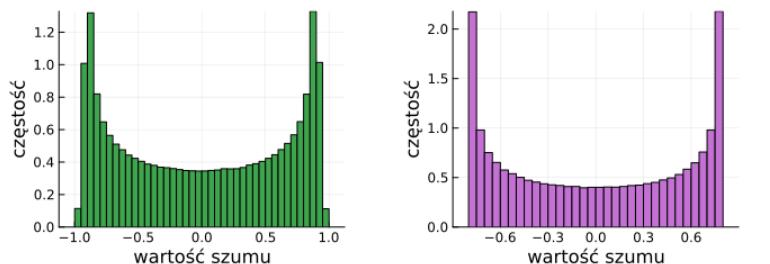


(d) Przy boku komórki siatki (0.05, 0.5).
 (e) Pomiędzy środkiem, a wierzchołkiem siatki (0.25, 0.25).

Rysunek 17: Histogramy punktowe szumu Perlina.



(a) Punkty siatki dla których wyliczono histogram.
 (b) Środek komórki siatki (0.07, 0.26).
 (c) Przy wierzchołku komórki siatki (0.007, 0.03).



(d) Przy boku komórki siatki (-0.04, 0.23).
 (e) Pomiędzy środkiem, a wierzchołkiem siatki (0.04, 0.15).

Rysunek 18: Histogramy punktowe szumu Simplex

3.2 Analiza częstotliwości

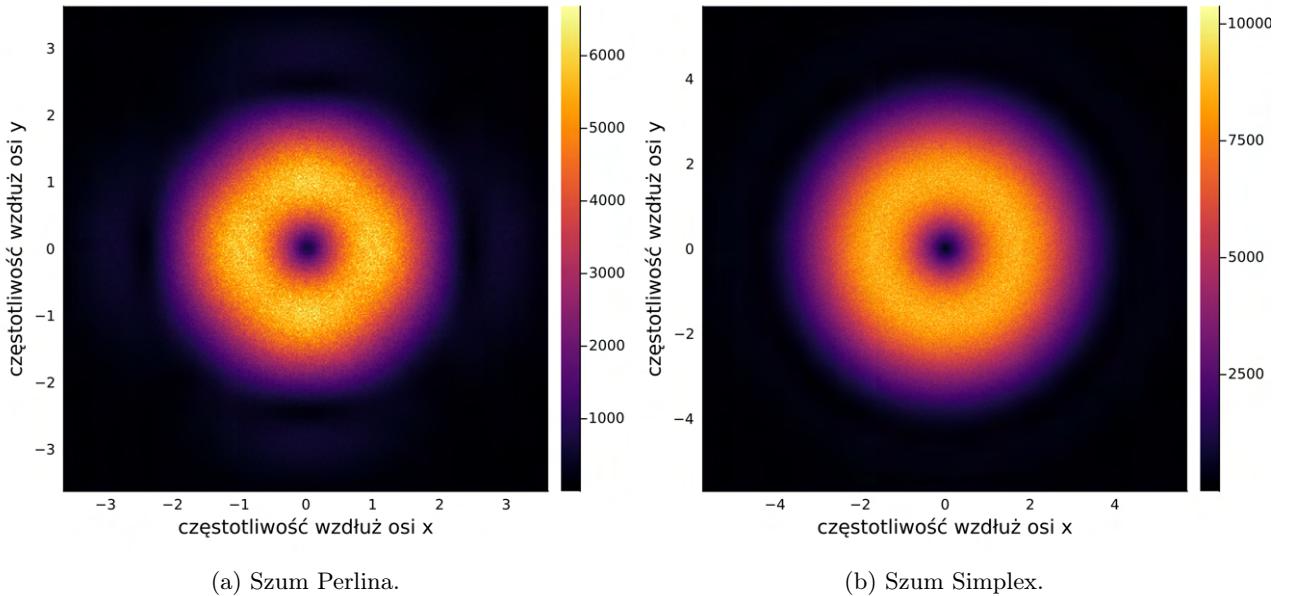
W celu zbadania izotropii chcielibyśmy zbadać częstotliwości występujące w szumach i sprawdzić, czy wzdłuż różnych kierunków są one takie same. Aby tego dokonać, wykorzystano dyskretną transformatę Fouriera. Dla realizacji (x_{nm}) pola losowego (X_{nm}), dla $n = 1, 2, \dots, N$; $m = 1, 2, \dots, M$ dwuwymiarowa dyskretna transformata Fouriera zdefiniowana jest jako [3]:

$$\mathcal{F}[x](k, l) = \sum_{n=1}^N \sum_{m=1}^M x_{nm} \cdot e^{-2\pi i (\frac{k}{N} n + \frac{l}{M} m)}.$$

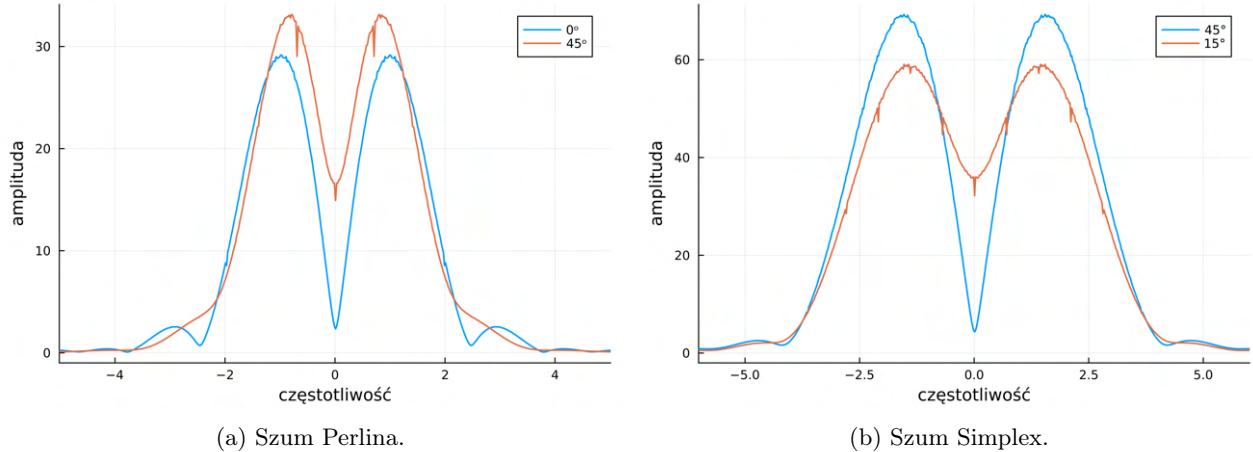
Początkowo wyliczono dwuwymiarową transformatę widoczną na Rysunku 19. Przedstawiony wynik jest średnią modułu transformaty z 100 realizacji pól losowych na obszarze $[0, 100] \times [0, 100]$, biorąc 2000×2000 wartości równooddalonych wzdłuż osi x i y. Dodatkowo nie pokazano całego otrzymanego widma, a jego wycinek z uwagi na duży obszar, gdzie amplituda była bliska 0.

W obu przypadkach kształt jest dosyć okrągły, co świadczyłoby o tym, że pole losowe jest niezmiennicze względem rotacji. Przyglądając się widmu szumu Perlina (Rys. 19a) możemy zauważać, że kształt jest w pewien sposób spłaszczony, przez co wygląda jak zaokrąglony kwadrat obrócony o 45° . Jest to spodziewany wynik, który jest dosyć zauważalny w realizacjach pola losowego. Natomiast w przypadku szumu Simplex nie jesteśmy w stanie zauważać braku izotropii. Spektrum wygląda, jakby miało symetrię kołową. Oczywiście wynika to z większego rzędu symetrii obrotowej siatki regularnej trójkątnej, przez co transformata jest znacznie bardziej zbliżona do takiej, jakiej spodziewalibyśmy się dla szumu izotropowego.

Aby lepiej zbadać występujące częstotliwości, wykonano analizę jednowymiarową, gdzie porównano ze sobą transformaty Fouriera pól losowych w zależności od nachylenia jednowymiarowego wycinka szumu. Tym razem wzięto średnią z $2 \cdot 10^4$ realizacji, gdzie badane odcinki były długości 100. Na Rysunku 20 przedstawiono porównanie najbardziej granicznych przypadków, czyli dla szumu Perlina nachylenia 0° (wzdłuż boków siatki) i 45° (wzdłuż przekątnej siatki), a dla szumu Simplex porównano nachylenia 45° (wzdłuż boków siatki) oraz 15° (wzdłuż prostej przecinającej trójkąty siatki na pół). Jak widać, w obu przypadkach różnice są bardzo wyraźne. Nawet dla metody Simplex, której dwuwymiarowa transformata wygląda na bliską izotropową, różnice są znaczące. Oznacza to, że szumy te rzeczywiście nie posiadają cechy, która wydaje się naturalna w kontekście generowania terenu, czyli niezmienniczości względem rotacji. W dalszej części pracy przedstawimy nowe metody, które lepiej radzą sobie z tym problemem. Na wykresach widać również pojedyncze szybkie spadki amplitudy. Wynikają one z periodyczności samej siatki i pojawiają się dokładnie w wielokrotnościach okresu siatki w danym kierunku.



Rysunek 19: Dwuwymiarowa transformata Fouriera omawianych szumów.



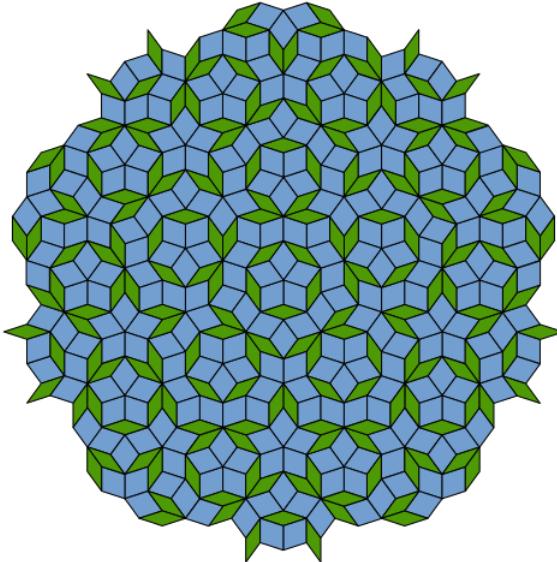
Rysunek 20: Transformata Fouriera jednowymiarowych wycinków omawianych szumów.

4 Aperiodyczny parkietaż

Pomysłem na poprawienie badanych metod jest zmiana siatki, z pomocą której generowane są pola losowe. Konstrukcje siatek można utożsamić z pojęciem parkietażu (również kafelkowania, tessellation), czyli ułożenia zbioru płyt (figur geometrycznych) w sposób, który pokrywa całą przestrzeń (najczęściej płaszczyznę) bez nakładających się na siebie figur [6]. Oznacza to, że działając na pewien zbiór płyt operacjami translacji, rotacji oraz (opcjonalnie) odbicia lustrzanego, jesteśmy w stanie pokryć całą płaszczyznę. Myśląc o siatkach, zwykle myślimy o samych brzegach figur, natomiast w przypadku tessellation zwykle rozważa się całe figury razem z ich wnętrzem. Jest to jedyna różnica, która w żaden sposób nie utrudnia konstruowania siatek za pomocą metod związanych z parkietażem.

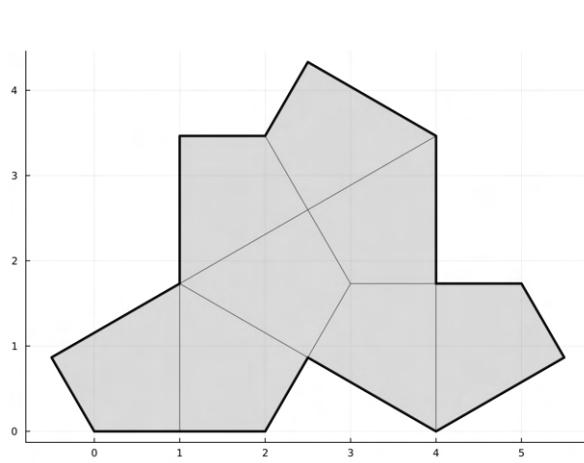
Problemem z dotychczas omawianymi metodami była okresowość i regularność wykorzystanych siatek. Okresowość rozumiemy jako cyklicznie powtarzający się wzorzec w utworzonym parkietażu, albo analogicznie istnienie nietrywialnej translacji całego parkietażu, która w żaden sposób go nie zmieni [18]. Regularność natomiast mówi o tym, że użyte płytki są wielokątami foremnymi. Kafelkowania z powyższymi cechami tworzą struktury, które nie są niezmiennicze względem rotacji i dodatkowo w każdym dowolnie wybranym kierunku, rozumianym jako jednowymiarowy wycinek parkietażu, odległości między kolejnymi bokami płyt są okresowe oraz zauważalnie różne w zależności od kierunku. Powoduje to, że dla szumu generowanego z siatki analogicznej do omawianego parkietażu występują wyraźne zmiany częstotliwości zależne od wybranego kierunku rozchodzenia się szumu.

Oczywistym pomysłem pozbicia się tych problemów jest wykorzystanie parkietażu, który nie posiada omówionych cech. Istnieje wiele nieokresowych tessellacji, z których jedną ze znaczących jest parkietaż Penrose'a [10], wykorzystujący dwie różne płytki w kształcie rombów. Na Rysunku 21 przedstawiono, jak on wygląda. Co ciekawe, jest to pierwszy znany parkietaż, którego symetria obrotowa jest rzędu 5. Mimo to, gdybyśmy wykorzystali go do wygenerowania szumu metodami podobnymi do omawianych, wynik byłby izotropowy. Wynika to z tego, że dla różnych kierunków, wzdłuż których rozchodziłby się szum, nie występowałby okres w kolejnych odległościach od brzegów figur lub występowałby, ale byłby wielokrotnie dłuższy niż w regularnych siatkach, przez co różnice w częstotliwościach wzdłuż różnych kierunków nie byłyby aż tak duże. W praktyce wykorzystanie takiej siatki do generowania szumu nie jest dobrym pomysłem. Problemem jest trudność sprawdzania, w jakiej komórce siatki znajduje się punkt z \mathbb{R}^2 . Byłoby to oczywiście możliwe za pomocą sprawdzenia, czy punkt należy do każdej komórki z jakiegoś ograniczonego obszaru. Nie jest to jednak szybka ani prosta w implementacji metoda, więc chcielibyśmy jej uniknąć.

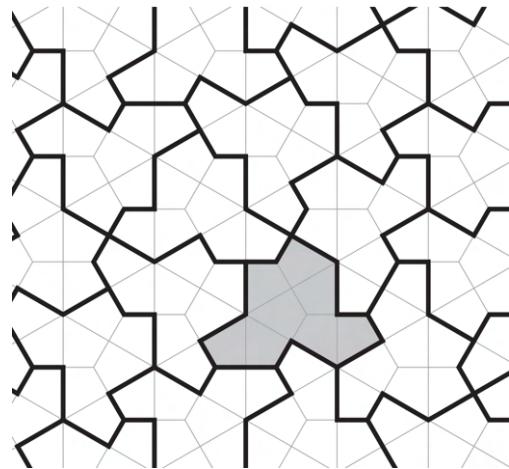


Rysunek 21: Parkietaż Penrose'a stworzony z dwóch różnych rombów; źródło grafiki: [wikipedia.org/wiki/Penrose_tiling](https://en.wikipedia.org/wiki/Penrose_tiling).

Okazuje się jednak, że istnieje tesselação, której struktura nadaje się bardzo dobrze do wykorzystania w generowaniu pola losowego. Jest to aperiodyczny parkietaż jedną płytka, który zaproponowany został w 2023 roku [14]. Na Rysunku 22b przedstawiono wygenerowany za jej pomocą parkietaż oraz samą płytke na Rysunku 22a. Zaproponowana płytka jest odpowiedzią na wieloletnie pytanie, czy istnieje pojedyncza figura, która tworzy tylko nieperiodyczny parkietaż. Jej wskazanie tak naprawdę wiązało się z wskazaniem całej rodziny takich płytaków, które parametryzowane są za pomocą długości swoich boków [14, 15]. W przypadku tej konkretnej płytki ma ona długości boków 1 i $\sqrt{3}$ i nazywać ją będziemy kapeluszem (z ang. *hat*), z uwagi na podobieństwo do niego. Warto również podkreślić, że kapelusz pokrywa całą płaszczyznę, ale wymaga do tego wykorzystania odbicia lustrzanego płytaków. Wśród rozważanej rodziny płytaków wszystkie poza jedną wymagają odbicia lustrzanego. Wyjątkiem jest figura o równych bokach długości 1 , której problemem jest natomiast to, że bez dodania dodatkowych ograniczeń można uzyskać parkietaż, który jest periodyczny. Do naszych zastosowań jednak nie jest istotne, czy do tworzenia kafelkowania używamy odbić lustrzanych. Parkietaż kapeluszem jest dobry do generowania szumu z uwagi na to, że siatka stworzona z kapeluszy składa się z deltoidów (o kątach wewnętrznych 90° , 60° i 120°), które ustawione są na siatce trójkątnej równobocznej, czyli dokładnie takiej, której używamy w przypadku szumu Simplex. Taka konstrukcja pozwala na wykorzystanie metod podobnych do tych ze wspomnianego algorytmu, dzięki którym w szybki i stosunkowo prosty sposób będziemy w stanie zlokalizować dla zadanego punktu z \mathbb{R}^2 , w której komórce siatki się znajduje.



(a) Płytką kapelusz.



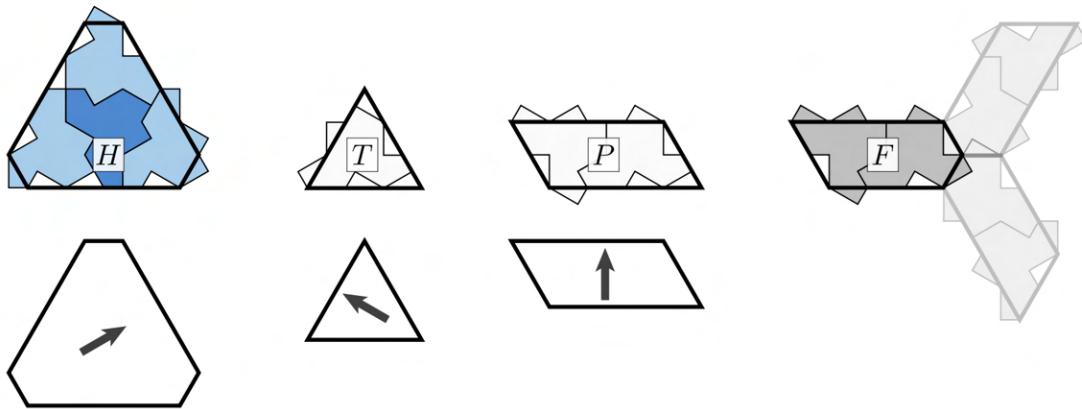
(b) Parkietaż płytka kapelusz; Rys. 1.1 z [14]

Rysunek 22: Płytką tworząca nieperiodyczny parkietaż.

4.1 Generowanie aperiodycznego parkietażu

W przeciwieństwie do tesselaacji kwadratem czy trójkątem, wygenerowanie algorytmicznie parkietażu płytka kapelusz jest bardziej skomplikowane. Wykorzystana do tego zostanie konstrukcja geometryczna, która w pracy autorów płytki wykorzystana została do dowodu pokrywania płaszczyzny przez tę figurę.

Metoda ta polega na wykorzystaniu innego zbioru figur, które nazywać będziemy metapłytkami. Mamy zdefiniowane proste zasady, w jaki sposób pod takie metapłytki podstawić kapelusze. Kształt metapłytek oraz reguły co do ich podstawienia widzimy na Rysunku 23. Mamy cztery różne figury: **H** – nieforemny sześciokąt, **T** – równoboczny trójkąt, **P** – równoległobok oraz **F** – pięciokąt będący ściętym równoległobokiem. Strzałka w przedstawieniu metapłytek jest wymagana, aby jednoznacznie stwierdzić rotacje płytka, które jeśli chodzi o zasady podstawiania pod nie kapeluszy, nie mają symetrii obrotowej (płytki **F** nie wymaga strzałki, ponieważ sama w sobie nie ma symetrii rotacyjnej, da się zatem jednoznacznie stwierdzić jej obrót). Długości ich boków wynikają bezpośrednio z reguł podstawieniowych i są równe: dla **H** – 8, 2; dla **T** – 6; dla **P** – 8, 4; dla **F** – 8, 4, 6, 2, 2.

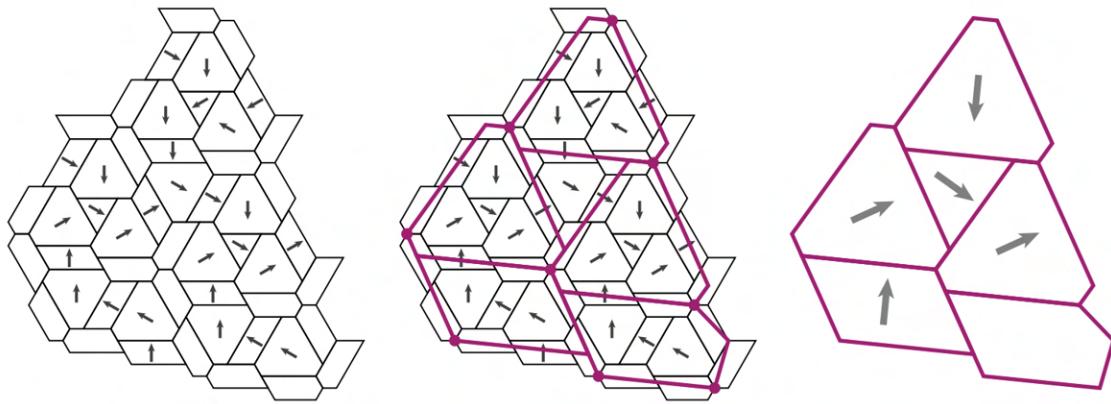


Rysunek 23: Metoda podstawienia kapeluszy pod metapłytki; Rys. 2.5 z [14].

Dla tak zdefiniowanych metapłytek mamy iteracyjną metodę generowania coraz większych powierzchni złożonych z tych figur. Też opiera się ona na podstawieniach, ale niestety nie tak bezpośrednich. W każdej kolejnej iteracji musimy stworzyć nowe płytki, które są tymi samymi figurami, jak podstawowe metapłytki,

ale długości ich boków się zmieniają. Oznacza to, że między kolejnymi iteracjami metapłytki tego samego typu nie są do siebie podobne, za wyjątkiem płytki **T**, która zawsze jest trójkątem równobocznym. Większość kątów w figurach w kolejnych iteracjach pozostaje taka sama, za wyjątkiem figury **F**, której kąt między górną (względem Rysunku 23) podstawą a sąsiednim bokiem po prawej oraz kąt między dolną podstawą a prawym sąsiednim bokiem zmieniają się w kolejnych iteracjach.

Konstrukcję kolejnych metapłytek widać na Rysunku 24. Polega ona na stworzeniu siatki, która można zobaczyć po lewej stronie omawianego rysunku. Kolejnym krokiem jest „opisanie” nowych metapłytek (zaznaczonych na fioletowo) na stworzonej powierzchni, w taki sposób, aby kluczowe wierzchołki (zaznaczone fioletowymi kropkami) były umieszczone w odpowiednich wierzchołkach siatki tak jak na środku rysunku. Możemy dzięki temu wyliczyć wszystkie wymiary oraz względową rotację nowych metapłytek, a następnie powtórzyć czynności, aż otrzymamy figurę odpowiadającą nam wielkości.



Rysunek 24: Konstrukcja kolejnych metapłytek; Rys. 2.6 z [14].

Najtrudniejszą częścią całej metody jest wyliczanie wielkości nowych metapłytek. Korzystając z geometrii i trygonometrii, jesteśmy w stanie policzyć je w sposób, który przedstawiono poniżej. Nie wyprowadzono wszystkich wzorów, ponieważ wykonane rachunki nie wnoszą dużo do wyjaśnienia metody oraz, mimo pozornej złożoności wyników, nie wymagają skomplikowanej matematyki. Do wyprowadzenia poniższych wartości korzystano z twierdzenia sinusów, cosinusów oraz definicji funkcji trygonometrycznych.

Zacznijmy od nazwania wymiarów płytki **F** w i -tej iteracji kolejno: a_i – górną podstawą, b_i – lewy bok i c_i – dolną podstawą. Korzystając z własności, że kąt przy skrajnie prawym wierzchołku płytki **F** zawsze musi być równy $\frac{2\pi}{3}$ (120°) możemy wyznaczyć nachylenie między dolną podstawą a prawym sąsiednim bokiem tej figury, który wynosi:

$$\gamma_i = \frac{\pi}{6} + \text{atan2}\left(\frac{b_i\sqrt{3}}{2}, -a_i + \frac{b_i}{2} + c_i\right).$$

Zauważając dodatkowo, że boki z prawej strony figury **F** muszą być takie same, możemy wyliczyć ich długość:

$$d_i = \frac{\sqrt{3}}{3} \sqrt{\left(-a_i + \frac{b_i}{2} + c_i\right)^2 + \left(\frac{b_i\sqrt{3}}{2}\right)^2}.$$

W ten sposób otrzymujemy wszystkie potrzebne wymiary do zdefiniowania metapłytki **F**, natomiast z konstrukcji siatki na Rysunku 24, możemy wyliczyć długości boków reszty płytek:

- płytka **H**: a_i (dolna podstawa), $c_i - b_i$;
- płytka **T**: $a_i + b_i - c_i$;
- płytka **P**: a_i (dolna podstawa), b_i .

Mając tak sparametryzowane metapłytki, możemy rozpocząć wyliczanie wymiarów kolejnej iteracji metapłytek. Dzięki stworzonej konstrukcji z Rysunku 24 znamy współrzędne kluczowych punktów, więc jesteśmy w stanie policzyć długości między nimi oraz kąty nachylenia odcinków łączących wybrane punkty. Zaczynając od nowej metapłytki **F** z prawej dolnej strony rysunku w bezpośredni sposób wyliczamy długość dolnej podstawy tej płytki, otrzymując c_{i+1} . Możemy również znaleźć kąt ostry α_{i+1} między tą samą podstawą płytki **F** a poziomem.

Teraz wyliczamy odległość z_{i+1} między dwoma dolnymi punktami kluczowymi należącymi do którejkolwiek z metapłytek **H**. Potrzebne nam będzie również nachylenie β_{i+1} między odcinkiem łączącym te dwa punkty a poziomem. Z tymi wymiarami oraz przy wykorzystaniu twierdzenia sinusów możemy znaleźć wartości a_{i+1} i b_{i+1} w następujący sposób:

$$a_{i+1} = \frac{2 \sin\left(\frac{\pi}{3} + \alpha_{i+1} - \beta_{i+1}\right)}{\sqrt{3}},$$

$$b_{i+1} = c_{i+1} - \frac{2 \sin(\beta_{i+1} - \alpha_{i+1})}{\sqrt{3}}.$$

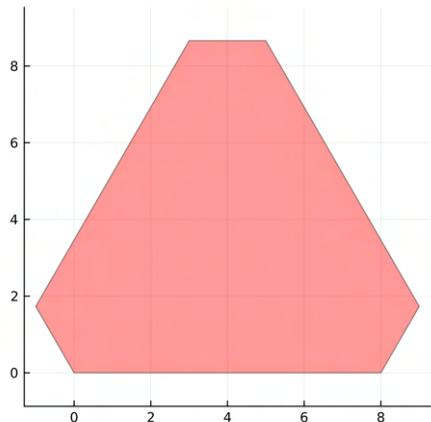
Tym sposobem otrzymujemy wszystkie parametry potrzebne do skonstruowania metapłytek $i+1$ iteracji. W zależności od tego, jak duży obszar stworzony z metapłytek chcemy uzyskać, możemy powtarzać metodę dowolną ilość razy, zaczynając oczywiście od parametrów spełniających konstrukcję początkowych metapłytek, czyli $a_0 = 8$, $b_0 = 4$ i $c_0 = 6$. Dodatkowo w metodzie pojawia się istotny nowy parametr α_{i+1} , który jest względna ujemną rotacją między metapłytkami z iteracji i a z tymi z iteracji $i+1$. Wartość ta również będzie potrzebna, żeby skonstruować powierzchnię stworzoną z takich metapłytek.

Chcąc stworzyć konstrukcję korzystającą z n iteracji, będziemy potrzebowali wartości parametrów a_i , b_i , c_i , α_i dla $i = 0, 1, 2, \dots, n$. Parametr α_0 oznacza nachylenie względem osi x wygenerowanej powierzchni i możemy go dobrać dowolnie w zależności od potrzeby. Zaczynamy algorytm od ustawienia dowolnej z czterech metapłytek n -tej iteracji pod kątem $-(\alpha_0 + \alpha_1 + \dots + \alpha_n)$. Następnie podstawiamy pod nią metapłytki z iteracji $n-1$. Podstawienie metapłytek zachodzi według zasad ze środkowej i prawej części Rysunku 24. Następnie pod wszystkie płytki z iteracji $n-1$ podstawiamy płytki z iteracji $n-2$ i proces ten kontynuujemy, aż dojdziemy do podstawowych metapłytek. Otrzymujemy w ten sposób konstrukcję stworzoną z podstawowych płyt, pod które już w prosty sposób jesteśmy w stanie podstać kapelusze.

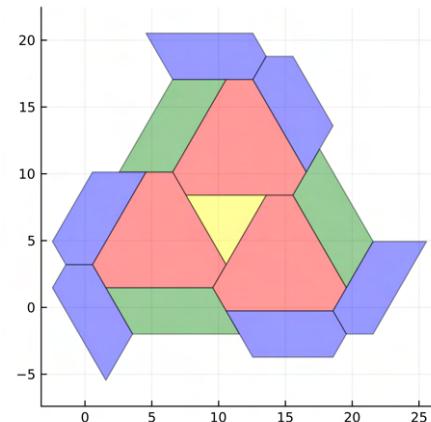
Na Rysunku 25 przedstawiono wyniki algorytmu zaczynającego od metapłytki **H** dla różnych ilości iteracji. Widzimy, że powierzchnia tych konstrukcji, jak i liczba płyt dosyć szybko rośnie. Jest to wzrost wykładniczy wynikający z metody, która bazuje na iteratywnym podstawianiu kilku płyt pod każdą pojedynczą płytke. Dodatkowo na Rysunku 26 widać, jak wygląda wygenerowana powierzchnia dla trzech iteracji, ale zaczynając od innych metapłytek. W praktyce najczęściej będziemy używać tej stworzonej z metapłytki **P** (Rys. 26b) z uwagi na to, że wynik jest co do kształtu całej konstrukcji najbliższy prostokątowi, a na takiej powierzchni najprościej operować. Na obu rysunkach można jeszcze zauważać, że niektóre płytki w wyniku są ciemniejszego koloru. Jest to efekt tego, że wszystkie figury na rysunkach narysowano jako lekko przezroczyste, przez co wynik staje się ciemniejszy w miejscach, gdzie nałożą się na siebie płytki. W niektórych przypadkach zasady, którymi się kierujemy, powodują, że w to samo miejsce kilkukrotnie położymy tę samą płytke. Zwykle nie jest to problem, a jeśli chcemy, to w stosunkowo prosty sposób jesteśmy w stanie pozbyć się powtarzających się płyt za pomocą metody opisanej w podsekcji 4.2.

Na Rysunku 27 przedstawiono ostateczny wynik, jakim jest siatka stworzona z kapeluszy. Jak widać, algorytm rozpoczęto od metapłytki **P**, a wynik dodatkowo obrócono o 15° , przeskalowano przez $\frac{\sqrt{2}}{6}$ i przesunięto, tak aby wierzchołek, w którym spotyka się 6 deltoidów widocznych na Rysunku 22b, był w punkcie $(0, 0)$. Zrobiono to, aby parkietaż pokrywał się z siatką używaną w metodzie Simplex. W rzeczywistym zastosowaniu będziemy używać właśnie tak przekształconej siatki. Kolory płytke odpowiadają kolorom wcześniej utożsamianym z metapłytkami, za które podstawiono kapelusze.

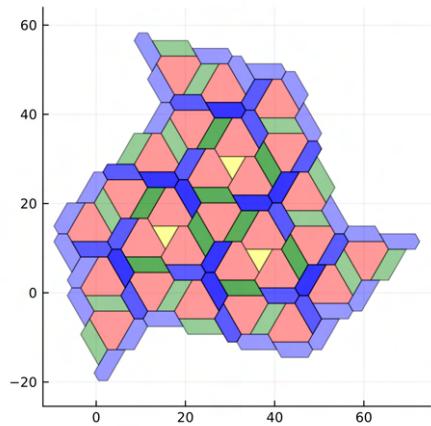
Czas generowania takiej powierzchni rośnie wykładniczo w zależności od ilości iteracji, natomiast jak wcześniej wspomniano, pole wygenerowanej powierzchni również rośnie wykładniczo, więc czas działania algorytmu rośnie potęgowo w stosunku do wielkości generowanej powierzchni.



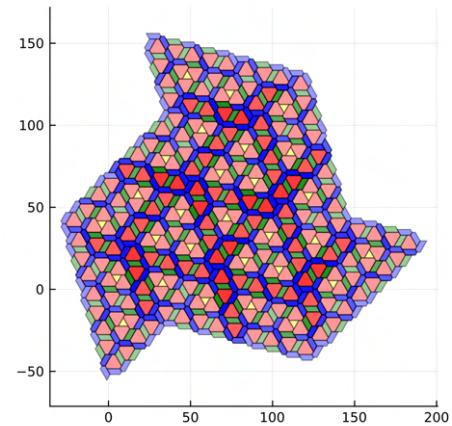
(a) Zero iteracji.



(b) Jedna iteracja.

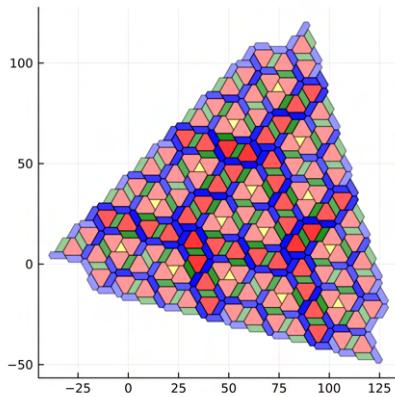


(c) Dwie iteracje.

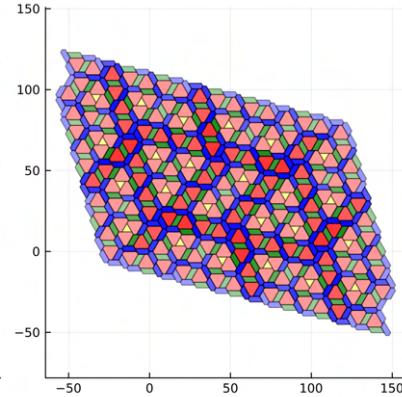


(d) Trzy iteracje.

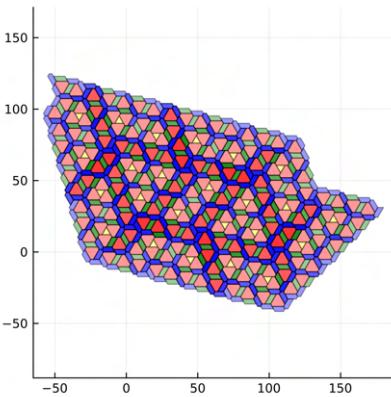
Rysunek 25: Powierzchnie stworzone z metapłytek zaczynając od metapłytki \mathbf{H} dla różnych ilości iteracji.



(a) Metapłytką \mathbf{T} .

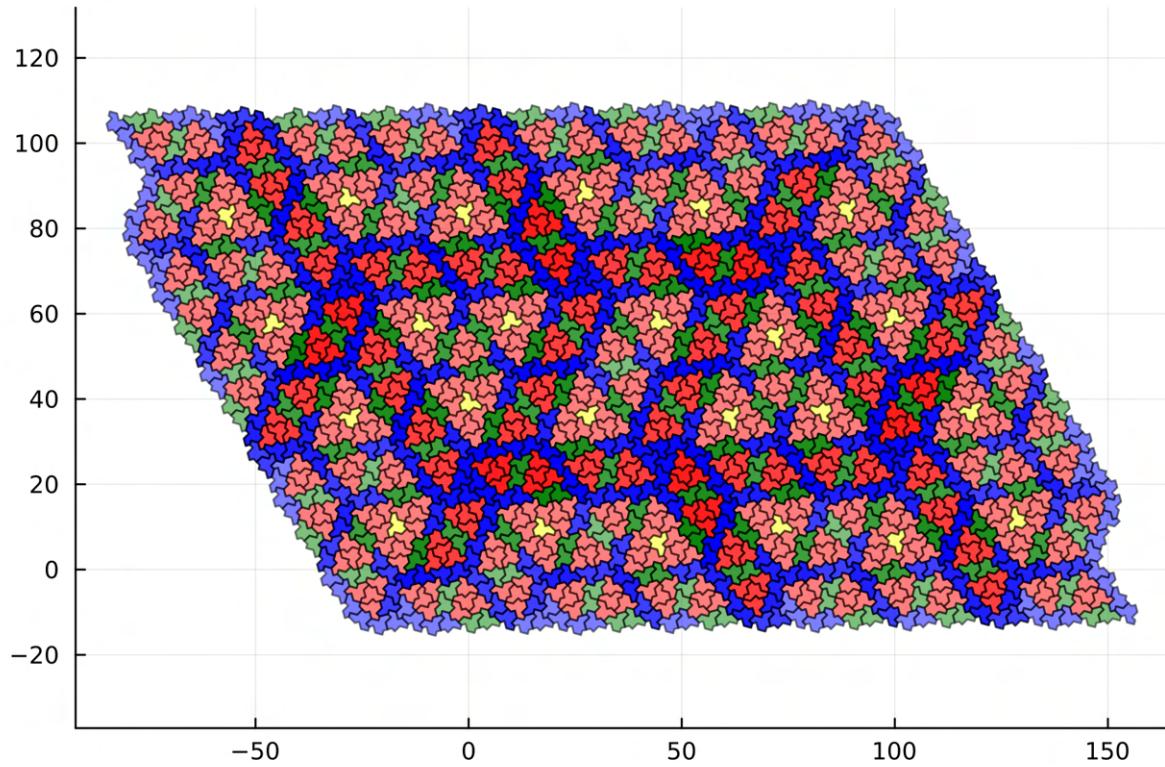


(b) Metapłytką \mathbf{P} .



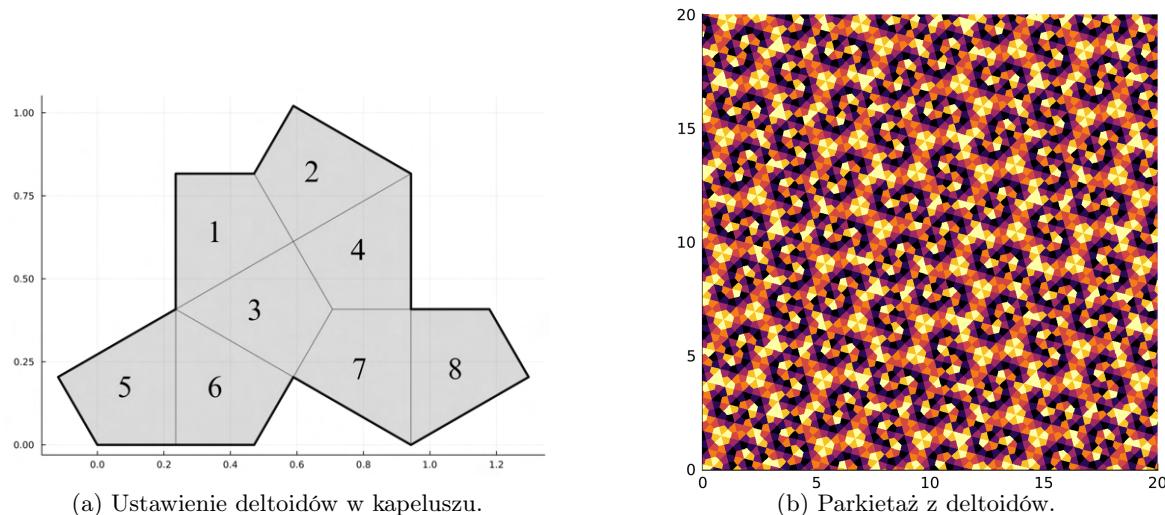
(c) Metapłytką \mathbf{F} .

Rysunek 26: Powierzchnie stworzone z metapłytek w zależności od doboru początkowej płytki.



Rysunek 27: Powierzchnia stworzona z kapeluszy.

Tak stworzona siatka niestety nie wystarczy do wygenerowania szumu, z uwagi na kształt płytka kapelusz. Interpolacja wartości w sposobie analogiczne do metod użytych w szumie Perlina i szumie Simplex nie jest możliwa, ponieważ płytka jest zbyt złożona oraz nie jest wypukła. Bardziej szczegółowe omówienie problemu znajduje się w sekcji 5. Aby sobie z tym poradzić, wykorzystamy fakt, że kapelusze stworzone są z ośmiu deltoidów, które możemy scharakteryzować w unikatowy sposób, przykładowo przypisując im liczby od 1 do 8 w sposób jak na Rysunku 28a. Dodatkowo narysowano mapę cieplną (Rys. 28b), na której pokazano, jak wygląda wycinek takiego parkietażu stworzonego z unikatowych deltoidów.



Rysunek 28: Konstrukcja unikatowych deltoidów.

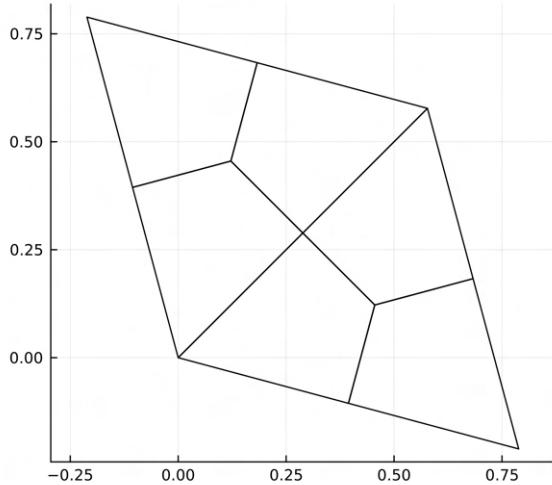
4.2 Reprezentacja siatki deltoidów

Aby móc wykorzystać parkietaż stworzony w poprzedniej podsekcji, będziemy potrzebowali przechowywać informacje dotyczące siatki w jakiejś dyskretnej strukturze danych. Poniżej przedstawiono opis metody, którą wykorzystano.

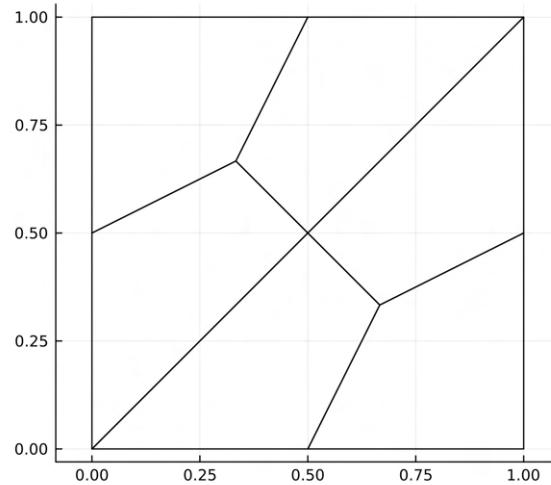
Najwygodniej zrobić to za pomocą deltoidów, tworzących siatkę, którą można podzielić na romby składające się z sześciu takich deltoidów. Na Rysunku 29a przedstawiono, jak wygląda taki romb. Wyliczając rotację deltoidu, jesteśmy w stanie jednoznacznie stwierdzić, w której części rombu znajduje się badana figura, ponieważ możliwe jest tylko 6 różnych kątów obrotu płytek. Potrzebna nam jest również informacja o tym, którym rombem w siatce jest ten, do którego należy dany deltoid. Wykorzystujemy do tego przekształcenie liniowe użyte w metodzie Simplex, dzięki któremu zamieniamy siatkę rombów na siatkę kwadratów o wierzchołkach o wartościach całkowitych. Przekształcenie widać na Rysunku 29b. Możemy teraz, wyliczając podłogę ze średniej z czterech punktów przekształconego deltoidu, jednoznacznie przypisać dwie liczby całkowite do każdego rombu oryginalnej siatki.

Korzystając przykładowo z macierzy z elementami będącymi wektorami długości 6, możemy zapisać wszystkie potrzebne nam informacje, czyli dowolną unikatową wartość płytki kapelusz, z której pochodzi deltoid oraz unikatową wartość deltoidu zdefiniowaną tak, jak na Rysunku 28a. Iterując po każdej płytce z wygenerowanej siatki, zapisujemy odpowiednie informacje w naszej strukturze danych. Korzystając z tej metody, z uwagi na powtarzające się płytki, będziemy nadpisywali informacje, więc musimy uważać, żeby wartości dotyczące unikatowości kapelusza, z którego pochodzą deltoidy, nie zostały nadpisane w sposób, który spowoduje rozbieżności między deltoidami, które powinny pochodzić z tych samych płytaków. Odpowiednio modyfikując tę metodę, możemy również pozbyć się powtórek.

Iterowanie po wszystkich płytach wygenerowanego parkietazu jest dosyć wymagające obliczeniowo, zwłaszcza że ilość płytaków rośnie wykładniczo w zależności od ilości iteracji. Nie jest to jednak duży problem, ponieważ zarówno wygenerowanie parkietazu, jak i zapisanie go jako dyskretna struktura danych musimy przeprowadzić tylko raz, żeby otrzymać obszar, który nas zadowala. Następnie możemy go zapisać i korzystać z niego dowolnie dużą liczbę razy.



(a) Oryginalna siatka.



(b) Przekształcona siatka.

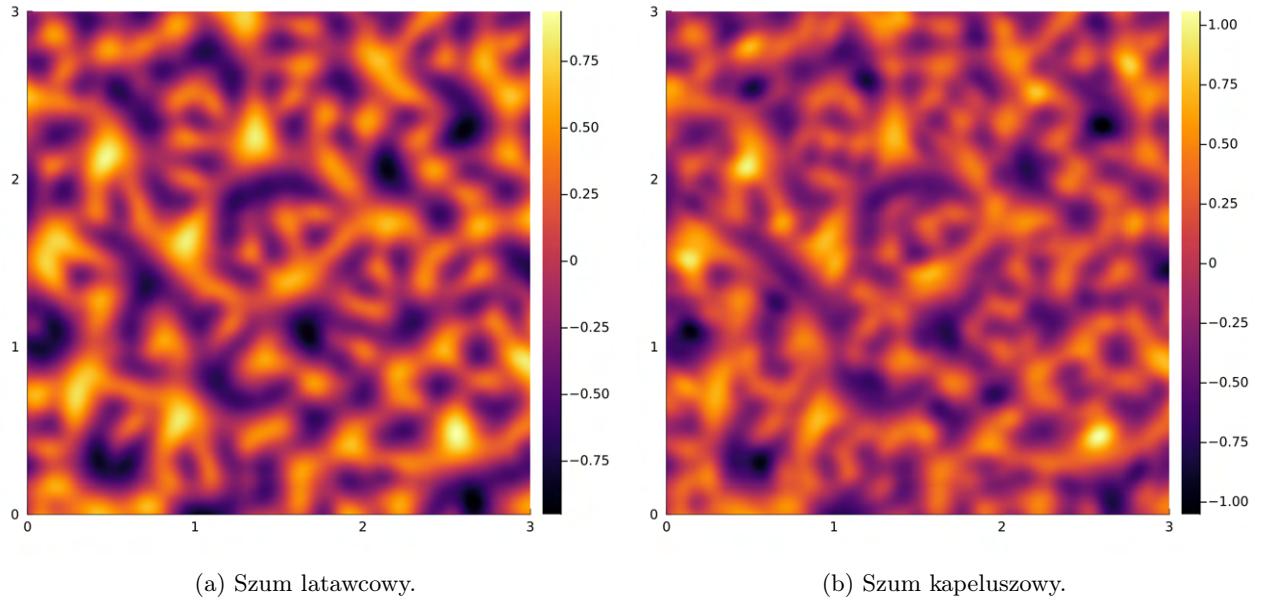
Rysunek 29: Wycinki siatki wykorzystane do zapisania parkietazu jako dyskretna struktura danych.

5 Nowe szumy

Korzystając z metod generowania aperiodycznego parkietazu i przekształcania go na dyskretną siatkę, jesteśmy w stanie opracować dwa nowe szumy. Realizacje ich widać na Rysunku 30. Pierwszy z nich (Rys. 30a), nazywany szumem latawcowym (deltoidy nazywa się latawcami – stąd nazwa) powstał jako efekt uboczny rozważań i nie jest metodą, która miała poprawić problemy klasycznych szumów. Bazuje ona na czystej siatce

deltoidowej, bez dodatkowych informacji pochodzących z płytki kapelusz. Stworzono go, ponieważ do metody generowania właściwego szumu na nieokresowej siatce i tak musielibyśmy zaimplementować taki na siatce deltoidowej. Wygląda wyraźnie inaczej niż poprzednie szumy. Widać na nim zageszczenia tworzące wizualne linie, które również występują w poprzednich metodach, natomiast tutaj są najbardziej widoczne. Wyglądają jednak bardziej naturalnie niż dla szumu Perlina, ponieważ nie tworzą się w tak nienaturalnych (0° , 45° i 90°) kierunkach, a wyglądają mocno losowo.

Druga zaprezentowana realizacja (Rys. 30b) szumu nazywanego szumem kapeluszowym, jest już właściwym polem losowym mającym rozwiązać problemy szumu Perlina i Simplex. Możemy zauważyć, że obie realizacje na Rysunku 30 są do siebie podobne. Wynika to z tego, że szum kapeluszowy również bazuje na siatce deltoidowej, więc wykorzystano taką samą tablicę wektorów losowych. Ta metoda wykorzystuje dodatkowo informacje pochodzące z kapeluszy tworzących aperiodyczny parkietaż. Wynik tego szumu jest trochę mniej ostry, a wyraźne linie, które powstają w przypadku szumu latawcowego, nie są aż tak wyraźne. W dalszej części sekcji wyjaśniono, w jaki dokładnie sposób wyglądają metody, które posłużyły do wygenerowania tych realizacji.



Rysunek 30: Realizacje nowych szumów.

Jak wcześniej wspomniano, obie metody bazują na siatce stworzonej z deltoidów. Musimy więc znać metodę, która będzie w stanie w szybki sposób znaleźć komórkę (tym razem w kształcie deltoidu), w której środku leży punkt, dla którego wyliczamy wartość szumu. Nie jest to na szczęście trudne. Dużym uproszczeniem ponownie jest to, że siatka z deltoidów wpisana jest w siatkę regularną trójkątną używaną w metodzie Simplex. Możemy zatem w ten sam sposób znaleźć komórkę trójkątną, w której znajduje się rozważany punkt. Następnie znajdujemy ten wierzchołek trójkąta, który jest najbliższym punktu, dzięki czemu jesteśmy w stanie jednoznacznie stwierdzić, który deltoid jest tym szukanym. Całą procedurę zwizualizowano na Rysunku 29a, gdzie znajduje się romb podzielony na dwa trójkąty. Linie dodatkowo dzielące te trójkąty na trzy równe części to dokładnie linie, gdzie odległości w metryce euklidesowej od odpowiednich wierzchołków trójkątów są takie same. Dla wybranych dwóch wierzchołków zbiorem punktów, gdzie odległości od nich są równe, będzie prosta prostopadła do boku łączącego te dwa wierzchołki i przecinająca ten bok dokładnie w połowie. Rysując takie proste dla wszystkich par wierzchołków, jesteśmy w stanie skonstruować te same deltoidy, których używamy w siatce.

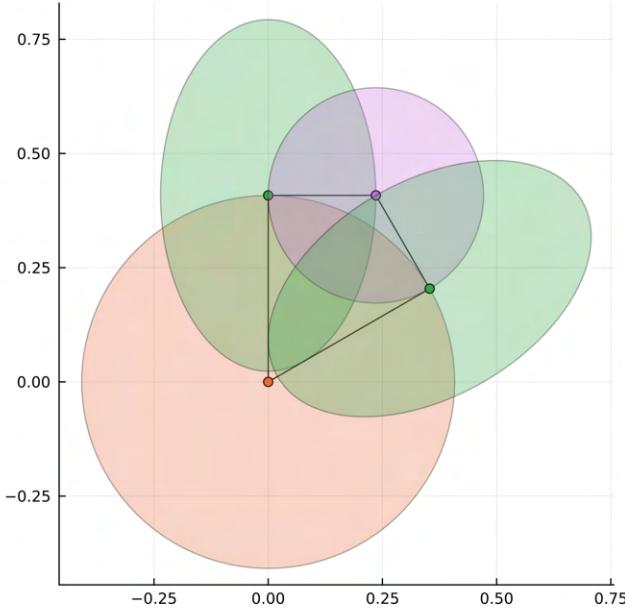
Kolejnym problemem jest metoda interpolacji na deltoidach. Rozwiążanie jest mocno powiązane z tym wykorzystanym w metodzie Simplex. Również korzystać będziemy z „obszarów wpływu”, jednak tym razem nie będą one wszystkie w tym samym kształcie. Wizualizację tych obszarów przedstawiono na Rysunku 31. Widać, że dla dwóch wierzchołków leżących na osi symetrii deltoidu obszary wpływu są w kształcie kół,

jednak nie o takich samych promieniach. Ich długości dobrano tak, aby były styczne do boków figury nie zawierających wierzchołka dotyczącego danego obszaru. Promień dotyczący obszaru wierzchołka przy kącie ostrym (na rysunku kolor pomarańczowy) jest równy $r_1 = \frac{\sqrt{6}}{6}$, natomiast promień drugiego kolistego obszaru (kolor fioletowy) to $r_2 = \frac{\sqrt{2}}{6}$. Jeśli chodzi o dwa pozostałe wierzchołki (kolory zielone) to „obszary wpływu” dla nich zaprojektowano w kształcie elips, których brzeg jest styczny do dłuższego boku przeciwnego oraz tak, aby przecinały wierzchołek przy kącie rozwartym deltoidu. Długości pól osi wielkiej a oraz pól osi małej b znaleziono konstruując odpowiedni układ równań i wynoszą one: $a = \frac{2\sqrt{3}}{9}$, $b = \frac{\sqrt{2}}{6}$ (długość krótszego boku deltoidu). Warto również dodać, że omawiane elipy są pochylone, więc w algorytmie zamiast klasycznego równania elipsy trzeba wykorzystać takie, które pozwala na ich obrót. Jest ono postaci:

$$\frac{((x - x_0) \cos(\theta) + (y - y_0) \sin(\theta))^2}{a^2} + \frac{((x - x_0) \sin(\theta) - (y - y_0) \cos(\theta))^2}{b^2} = 1,$$

gdzie θ to kąt obrotu elipsy.

Ponownie, jak w przypadku wcześniej omówionych metod wyliczamy odpowiednie iloczyny skalarne (dla każdego wierzchołka deltoidu), a następnie korzystając z powyższych rozważań interpolujemy wartości i sumujemy je w analogiczny sposób jak w metodzie Simplex.

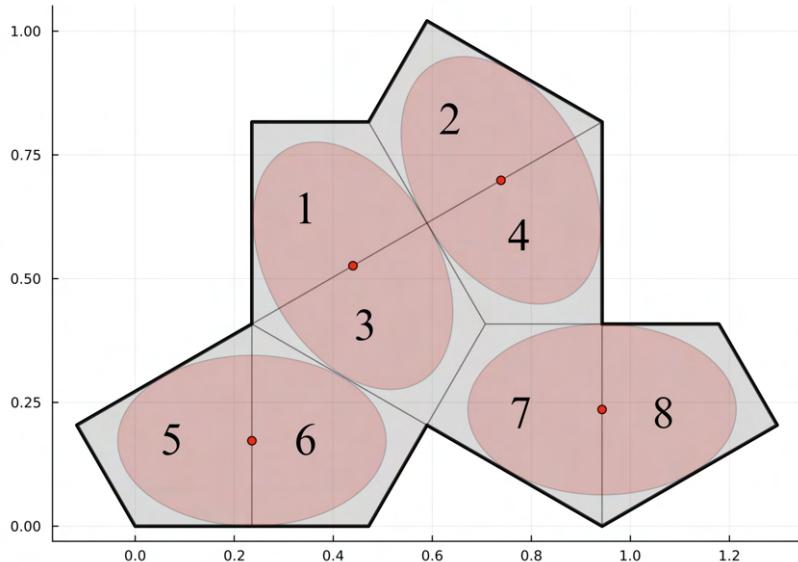


Rysunek 31: „Obszary wpływu” komórki w kształcie deltoidu.

Wszystkie rozważania powyżej dotyczą siatki z deltoidów i w żaden sposób nie uwzględniają siatki z kapeluszy. Metoda służąca do wyodrębnienia w jakiś sposób płytka tworzącej nieokresowy parkietaż opiera się na dodaniu dodatkowych wierzchołków na bokach deltoidów wewnętrz płytka kapelusz. Wierzchołki te ustawione są jak zaznaczono czerwonymi kropkami na Rysunku 32. Te punkty również mają swoje obszary wpływu, które są w kształcie elips, a zaprojektowane są tak, aby nie wychodziły poza brzeg figury. Nie wyliczano wszystkich parametrów elipsy, aby jej brzegi były dokładnie styczne do odpowiednich boków dwóch deltoidów, do których należą. Zamiast tego przyjęto przybliżenie będące trochę mniejszą elipsą, ale dostatecznie dużą, aby otrzymać oczekiwany efekt. Środek obszaru znajduje się na dłuższym boku deltoidu, dokładnie w odległości 1 od wierzchołka przy kącie ostrym. Przyjęta półosi wielka elipsy wynosi $a_h = 1.15 \cdot \frac{\sqrt{2}}{6}$, natomiast półosi mała wynosi $b_h = (\sqrt{3} - 1) \frac{\sqrt{2}}{6}$. Aby dodatkowo wyodrębnić nieokresowość tego parkietazu, dla każdej płytka kapelusz generujemy tylko jeden wektor losowy i „umieszczaemy” go w dodanych wierzchołkach. Zatem dla pewnego punktu na siatce wyliczamy 4 iloczyny skalarne związane z deltoidem oraz dodatkowy iloczyn skalarne związany z nowym wierzchołkiem. Interpolujemy wartości z użyciem „obszarów wpływu”, a następnie, sumując wyniki, otrzymujemy odpowiednią realizację szumu.

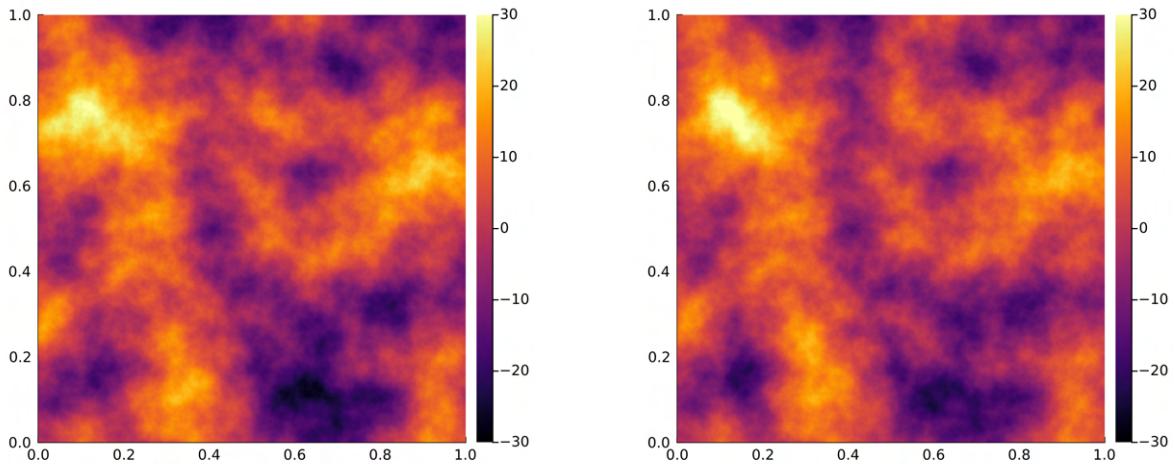
Generowanie szumów jest wolniejsze niż w przypadku wcześniej omawianych metod. Nie jest to na szczeble nieznaczna różnica. Złożoność obliczeniowa nie zmienia się (wciąż jest $O(n)$, gdzie n to ilość generowanych punktów), natomiast wykonujemy więcej operacji. Obie nowe metody bazują na prawie tych samych czynnościach, jak dla szumu Simplex, ale dodają nowe elementy jak „obszary wpływu” będące obróconymi elipsami. Do obracania korzysta się z funkcji trygonometrycznych, których wyliczanie jest stosunkowo wolne, ale można się ich pozbyć, z uwagi na to, że elipy obracamy tylko w kilku ustalonych kierunkach, możemy stabilizować odpowiednie wartości funkcji trygonometrycznych. Dodawane są również dodatkowe bloki warunkowe, bo mamy więcej wierzchołków, co też nieznacznie spowalnia algorytm. Problemem również może się wydawać wygenerowanie siatki aperiodycznej. Rzeczywiście jest to dosyć długi proces, który dodatkowo rośnie wykładniczo wraz z ilością iteracji, natomiast tak naprawdę wystarczy, że raz wygenerujemy siatkę, o wielkości zależnej od potrzeby i możemy jej używać dowolnie wiele razy do różnych zastosowań i do różnych realizacji szumu.

Istnieją również inne problemy nowych metod. Zaproponowany szum kapeluszowy nie nadaje się do implementacji jako procedura wbudowana CPU czy GPU, co spowodowane jest potrzebą przetrzymywania w pamięci tablicy przeglądowej (z ang. *lookup table*), będącej w tym wypadku strukturą danych związaną z generowaną siatką aperiodyczną. Kolejnym problemem jest to, że analityczne policzenie pochodnych szumu nie jest aż takie proste, jak w przypadku szumu Simplex, co jest tego ostatniego dosyć sporą zaletą. Nie jest to niemożliwe, ale wymaga bardziej złożonych rachunków. Można wyliczać numerycznie pochodne, korzystając z różnic skończonych lub innych metod, ale wymaga to znacznie większych zasobów obliczeniowych. Wyliczanie pochodnych przydaje się przykładowo do generowania innych typów szumów przypominających przepływającą ciecz [8]. Problemem jest również brak uogólnienia metody do przestrzeni n-wymiarowych, które zarówno w przypadku szumu Perlina, jak i szumu Simplex jest bardzo proste.



Rysunek 32: Dodatkowe wierzchołki w płytce kapelusz.

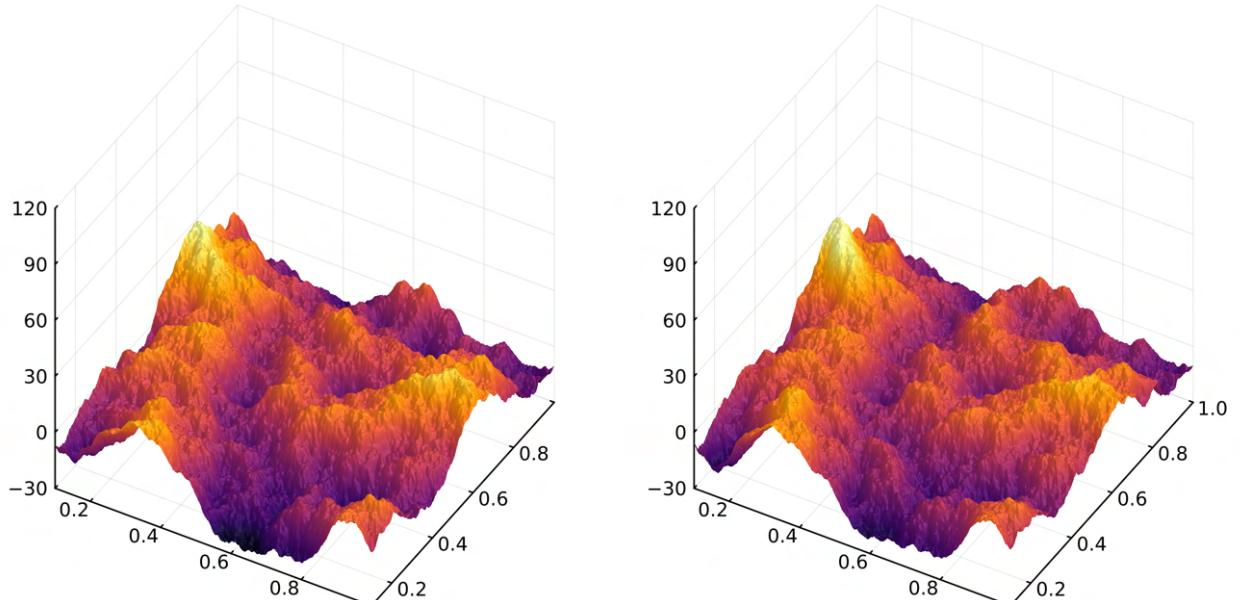
Tak samo jak w przypadku klasycznych metod, w praktyce korzystać będziemy z różnych przekształceń szumów oraz z sumowania realizacji o różnych częstotliwościach, żeby uzyskać szumy fraktalne. Pola losowe po takich przekształceniach wyglądają podobnie jak w przypadku szumów Perlina i Simplex. Na Rysunkach 33 (mapy cieplne) i 34 (wykresy powierzchniowe) przedstawiono nowe szumy jako szumy fraktalne (cztery różne realizacje, każda kolejna o trzykrotnie większej częstotliwości i trzykrotnie mniejszej amplitudzie). Oba wyniki wyglądają dobrze w kontekście generowania terenu. Na wykresach powierzchniowych (Rys. 34) widać, że stworzony krajobraz wygląda naturalnie i jest pozbawiony artefaktów kierunkowych. Wygenerowane mapy różnią się nieznacznie od siebie, ale trudno jednoznacznie opisać te różnice.



(a) Szum latawcowy.

(b) Szum kapeluszowy.

Rysunek 33: Mapy cieplne realizacji nowych fraktalnych szumów.



(a) Szum latawcowy.

(b) Szum kapeluszowy.

Rysunek 34: Wykresy powierzchniowe realizacji nowych fraktalnych szumów.

Algorytmy 3 i 4 przedstawiają dokładne instrukcje potrzebne do wygenerowania realizacji szumów kolejno latawcowego i kapeluszowego. Są one wyraźnie bardziej złożone niż klasyczne metody, ale z uwagi na to, że są to modyfikacje szumu Simplex, nie są dużo trudniejsze w implementacji. Niektóre przedstawione kroki opisano słownie, ponieważ zależą od konkretnej implementacji struktury danych przechowującej informacje o siatce deltoidowej i kapeluszowej. Dodatkowo unikatowe indeksy w metodzie kapeluszowej zdefiniowane są tak, jak pokazano na Rysunku 32, na którym widać płytę niebędącą odbiciem lustrzanym oryginalnej figury. Tak jak wcześniej wspomniano, wszystkie wykorzystane wartości funkcji trygonometrycznych można

stabilicować w celach optymalizacyjnych. Wynik szumu nie jest dokładnie z przedziału $[-1, 1]$, ale graniczne wartości są bliskie -1 i 1 .

Algorytm 3 Szum latawcowy

Dla tablicy wektorów umieszczonych w wierzchołkach siatki deltoidowej, dla punktu $(x, y) \in \mathbb{R}^2$:

1. oblicz: $s = (x + y) \frac{\sqrt{3}-1}{2}$,
 2. oblicz: $i = \lfloor x + s \rfloor$, $j = \lfloor y + s \rfloor$,
 3. oblicz: $t = (i + j) \frac{\sqrt{3}-3}{6}$,
 4. oblicz: $x_0 = i + t$, $y_0 = j + t$,
 5. oblicz: $\Delta x'_0 = x - x_0$, $\Delta y'_0 = y - y_0$,
 6. jeśli $\Delta y'_0 > \Delta x'_0$
 $\Delta i = 0$, $\Delta j = 1$
jeśli $\Delta y'_0 \leq \Delta x'_0$
 $\Delta i = 1$, $\Delta j = 0$,
 7. oblicz: $\Delta x'_1 = \Delta x'_0 - \Delta i - \frac{\sqrt{3}-3}{6}$, $\Delta y'_1 = \Delta y'_0 - \Delta j - \frac{\sqrt{3}-3}{6}$, $\Delta x'_2 = \Delta x'_0 - 1 - \frac{\sqrt{3}-3}{3}$, $\Delta y'_2 = \Delta y'_0 - 1 - \frac{\sqrt{3}-3}{3}$,
 8. oblicz: $m = \min_{i \in \{0,1,2\}} \left(\sqrt{(\Delta x'_i)^2 + (\Delta y'_i)^2} \right) + 1$,
 9. znajdź: $\Delta x_0 = x_m$, $\Delta y_0 = y_m$, gdzie $(x_m)_{m=1,2,3} = (\Delta x'_0, \Delta x'_1, \Delta x'_2)$, $(y_m)_{m=1,2,3} = (\Delta y'_0, \Delta y'_1, \Delta y'_2)$,
 10. znajdź: $\alpha = a_{3\Delta i + m}$, gdzie $(a_n)_{n=1,\dots,6} = \left(\frac{5\pi}{12}, -\frac{\pi}{4}, -\frac{11\pi}{12}, \frac{\pi}{12}, \frac{3\pi}{4}, -\frac{7\pi}{12} \right)$
 11. oblicz: $[\Delta x_1, \Delta y_1] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{6}}{6} [\cos(\alpha + \frac{\pi}{6}), \sin(\alpha + \frac{\pi}{6})]$,
 $[\Delta x_2, \Delta y_2] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{6}}{6} [\cos(\alpha), \sin(\alpha)]$,
 $[\Delta x_3, \Delta y_3] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{6}}{6} [\cos(\alpha - \frac{\pi}{6}), \sin(\alpha - \frac{\pi}{6})]$,
 12. znajdź wektory $\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3$ odpowiadające odpowiednim wierzchołkom komórki siatki,
 13. oblicz: $t_0 = 1 - 6\Delta x_0^2 - 6\Delta y_0^2$,
 $t_1 = 1 - \frac{27}{4} (\Delta x_1 \cos(\alpha + \frac{\pi}{6}) + \Delta y_1 \sin(\alpha + \frac{\pi}{6})) - 18 (\Delta x_1 \sin(\alpha + \frac{\pi}{6}) - \Delta y_1 \cos(\alpha + \frac{\pi}{6}))$,
 $t_2 = 1 - 18\Delta x_2^2 - 18\Delta y_2^2$,
 $t_3 = 1 - \frac{27}{4} (\Delta x_3 \cos(\alpha - \frac{\pi}{6}) + \Delta y_3 \sin(\alpha - \frac{\pi}{6})) - 18 (\Delta x_3 \sin(\alpha - \frac{\pi}{6}) - \Delta y_3 \cos(\alpha - \frac{\pi}{6}))$,
 14. dla $k = 0, 1, 2, 3$ znajdź n_k postaci: $n_k = t_k^4 \vec{v}_k \cdot [\Delta x_k, \Delta y_k] \mathbb{1}_{\{t_k \geq 0\}}$,
 15. zwróć: $8(n_0 + n_1 + n_2 + n_3)$.
-

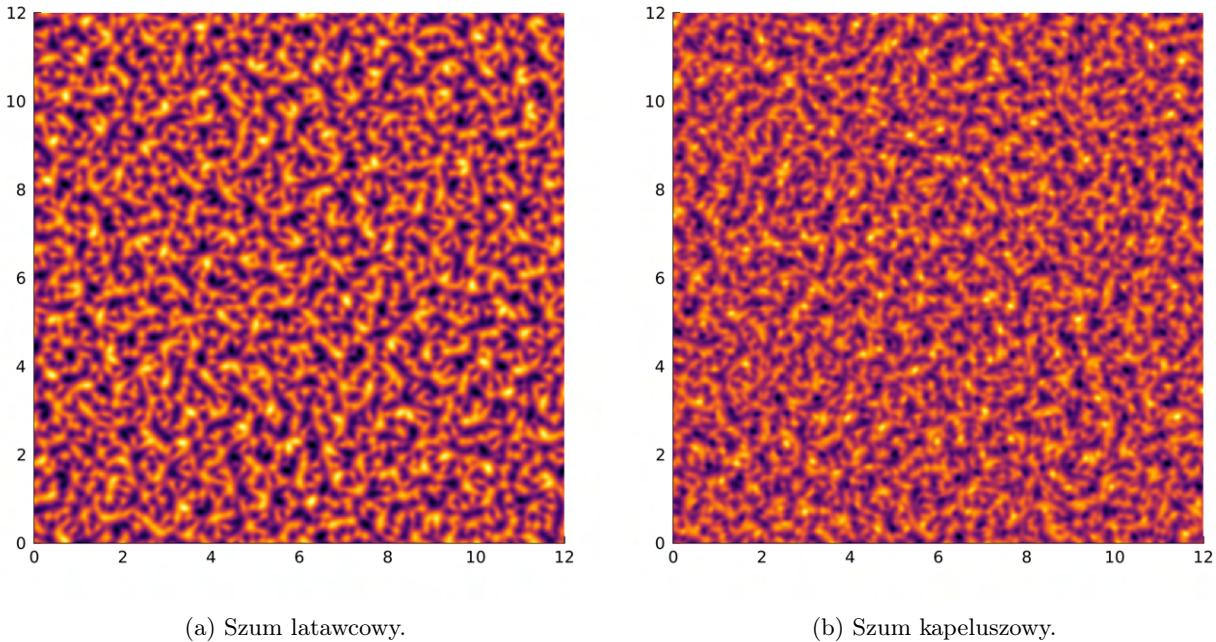
Algorytm 4 Szum kapeluszowy

Dla: tablicy wektorów umieszczonych w wierzchołkach siatki deltoidowej, tablicy unikatowych indeksów deltoidów tworzących kapelusz, tablicy informującej o tym czy dany kapelusz jest odbiciem lustrzanym oraz tablicy wektorów odpowiadających każdej komórce kapeluszowej siatki, dla punktu $(x, y) \in \mathbb{R}^2$:

1. oblicz: $s = (x + y) \frac{\sqrt{3}-1}{2}$,
 2. oblicz: $i = \lfloor x + s \rfloor, j = \lfloor y + s \rfloor$,
 3. oblicz: $t = (i + j) \frac{\sqrt{3}-3}{6}$,
 4. oblicz: $x_0 = i + t, y_0 = j + t$,
 5. oblicz: $\Delta x'_0 = x - x_0, \Delta y'_0 = y - y_0$,
 6. jeśli $\Delta y'_0 > \Delta x'_0$
 $\Delta i = 0, \Delta j = 1$
jeśli $\Delta y'_0 \leq \Delta x'_0$
 $\Delta i = 1, \Delta j = 0$,
 7. oblicz: $\Delta x'_1 = \Delta x'_0 - \Delta i - \frac{\sqrt{3}-3}{6}, \Delta y'_1 = \Delta y'_0 - \Delta j - \frac{\sqrt{3}-3}{6}, \Delta x'_2 = \Delta x'_0 - 1 - \frac{\sqrt{3}-3}{3}, \Delta y'_2 = \Delta y'_0 - 1 - \frac{\sqrt{3}-3}{3}$,
 8. oblicz: $m = \min_{i \in \{0,1,2\}} \left(\sqrt{(\Delta x'_i)^2 + (\Delta y'_i)^2} \right) + 1$,
 9. znajdź: $\Delta x_0 = x_m, \Delta y_0 = y_m$, gdzie $(x_n)_{n=1,2,3} = (\Delta x'_0, \Delta x'_1, \Delta x'_2), (y_n)_{n=1,2,3} = (\Delta y'_0, \Delta y'_1, \Delta y'_2)$,
 10. znajdź: $\alpha = a_{3\Delta i+m}$, gdzie $(a_n)_{n=1,\dots,6} = \left(\frac{5\pi}{12}, -\frac{\pi}{4}, -\frac{11\pi}{12}, \frac{\pi}{12}, \frac{3\pi}{4}, -\frac{7\pi}{12} \right)$
 11. oblicz: $[\Delta x_1, \Delta y_1] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{6}}{6} [\cos(\alpha + \frac{\pi}{6}), \sin(\alpha + \frac{\pi}{6})]$,
 $[\Delta x_2, \Delta y_2] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{6}}{6} [\cos(\alpha), \sin(\alpha)]$,
 $[\Delta x_3, \Delta y_3] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{6}}{6} [\cos(\alpha - \frac{\pi}{6}), \sin(\alpha - \frac{\pi}{6})]$,
 12. znajdź wektory $\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3$ odpowiadające odpowiednim wierzchołkom komórki siatki deltoidowej w której jest punkt (x, y) ,
 13. oblicz: $t_0 = 1 - 6\Delta x_0^2 - 6\Delta y_0^2$,
 $t_1 = 1 - \frac{27}{4} (\Delta x_1 \cos(\alpha + \frac{\pi}{6}) + \Delta y_1 \sin(\alpha + \frac{\pi}{6})) - 18 (\Delta x_1 \sin(\alpha + \frac{\pi}{6}) - \Delta y_1 \cos(\alpha + \frac{\pi}{6}))$,
 $t_2 = 1 - 18\Delta x_2^2 - 18\Delta y_2^2$,
 $t_3 = 1 - \frac{27}{4} (\Delta x_3 \cos(\alpha - \frac{\pi}{6}) + \Delta y_3 \sin(\alpha - \frac{\pi}{6})) - 18 (\Delta x_3 \sin(\alpha - \frac{\pi}{6}) - \Delta y_3 \cos(\alpha - \frac{\pi}{6}))$,
 14. znajdź u - unikatowy indeks komórki deltoidowej, w której jest punkt (x, y) ,
 l - wartość logiczna, czy płytka kapeluszowa w której jest punkt (x, y) , nie jest odbiciem lustrzanym,
 15. jeśli $u \in \{1, 4, 6, 7\}$ i l
 $[\Delta x_4, \Delta y_4] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{2}}{6} [\cos(\alpha - \frac{\pi}{6}), \sin(\alpha - \frac{\pi}{6})]$,
 $t_4 = 1 - \frac{18}{1.3225} (\Delta x_4 \cos(\alpha - \frac{2\pi}{3}) + \Delta y_4 \sin(\alpha - \frac{2\pi}{3})) - \frac{18}{4-2\sqrt{3}} (\Delta x_4 \sin(\alpha - \frac{2\pi}{3}) - \Delta y_4 \cos(\alpha - \frac{2\pi}{3}))$
w innym przypadku
 $[\Delta x_4, \Delta y_4] = [\Delta x_0, \Delta y_0] + \frac{\sqrt{2}}{6} [\cos(\alpha + \frac{\pi}{6}), \sin(\alpha + \frac{\pi}{6})]$,
 $t_4 = 1 - \frac{18}{1.3225} (\Delta x_4 \cos(\alpha + \frac{2\pi}{3}) + \Delta y_4 \sin(\alpha + \frac{2\pi}{3})) - \frac{18}{4-2\sqrt{3}} (\Delta x_4 \sin(\alpha + \frac{2\pi}{3}) - \Delta y_4 \cos(\alpha + \frac{2\pi}{3}))$,
 16. dla $k = 0, 1, 2, 3, 4$ znajdź n_k postaci: $n_k = t_k^4 \vec{v}_k \cdot [\Delta x_k, \Delta y_k] \mathbb{1}_{\{t_k \geq 0\}}$,
 17. zwrócić: $7(n_0 + n_1 + n_2 + n_3)$.
-

6 Analiza nowych szumów

Istotnym pytaniem jest to, czy rzeczywiście udało się zaprojektować szumy, w których nie występują artefakty kierunkowe omawiane na początku sekcji 3. Narysowano większe realizacje nowych szumów (Rys. 35), aby wizualnie ocenić wynik. Jak zauważono wcześniej, w metodzie opartej na czystej siatce deltoidowej występują pewnego rodzaju zagęszczenia tworzące linie. Są one jednak ustalone w losowo wyglądających kierunkach. Właściwy szum, czyli ten opierający się na siatce kapeluszowej, mimo że w dużej części bazuje na tej samej tablicy losowych wektorów, co przedstawiony szum latawcowy, prezentuje się inaczej. Szum wygląda na wyraźnie bardziej „rozmyty” od poprzedniego, co najpewniej jest spowodowane większą ilością wierzchołków siatki, a nienaturalnie wyglądające zagęszczenia są praktycznie niewidoczne, dzięki czemu artefakty związane z siatką również są niedostrzegalne. Wydaje się, jakby udało się stworzyć szum izotropowy i pozabawiony widocznych wad związanych z siatką, nie tracąc jednocześnie dużo na szybkości i możliwości implementacji algorytmu generowania pola losowego w sposób równoległy. W celu dodatkowego zbadania szumów i znalezienia bardziej formalnej odpowiedzi, czy zaproponowany szum jest izotropowy, wykonano analizę analogiczną do tej przeprowadzonej w sekcji 3 dla klasycznych metod.



Rysunek 35: Realizacje nowych szumów na dużych obszarach.

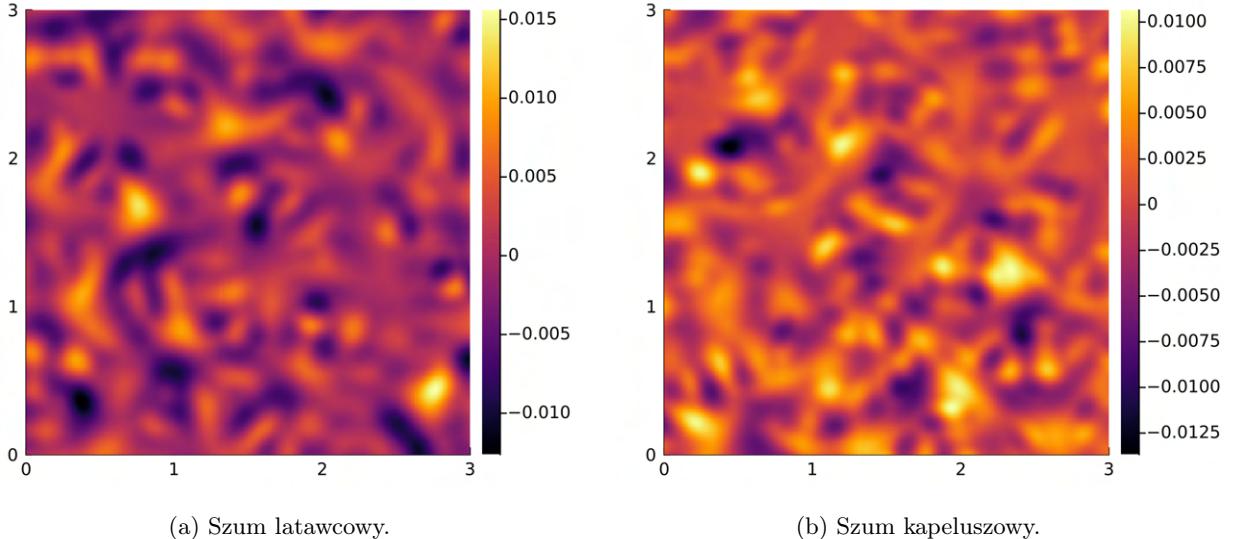
6.1 Analiza statystyczna

Tak samo jak w przypadku klasycznych metod wyliczono podstawowe cechy statystyczne nowych szumów. Z uwagi na duże podobieństwo metod generowania nowych szumów do algorytmu Simplex, wnioski wyciągnięte z tej części powinny być podobne. Wszystkie cechy wyliczono na podstawie 10000 realizacji szumów na powierzchni $[0, 3] \times [0, 3]$ i rozdzielczości 256×256 . Wzięto mniejszą powierzchnię w porównaniu do analizy szumu Perlina i Simplex, ponieważ siatka deltoidowa ma więcej komórek na takiej samej wielkości obszaru, przez co szum jest bardziej skoncentrowany.

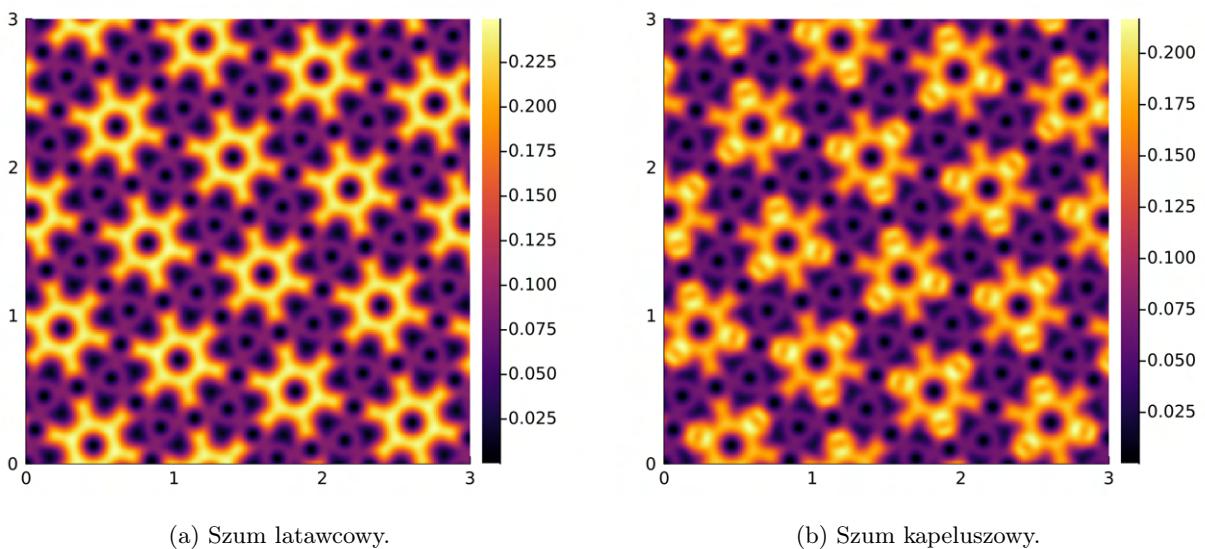
Wyliczono i narysowano (Rys. 36) średnie szumu latawcowego i kapeluszowego. Wyliczone wartości powinny zbiegać do wartości 0, co jak widać się dzieje, natomiast ponownie pozostaje pewna struktura szumów, co również zauważono w zachowaniu średnich klasycznych szumów. Istotność różnic średniej od 0 zbadano, wykonując test t-Studenta, którego wynik nie wykazał, żeby różnice były istotne statystycznie.

Wariancja omawianych pól losowych pokazana jest na Rysunku 37. Widzimy, że wynik składa się z dominujących fragmentów przypominających koła zębate. Wynika to z użytej metody interpolacji. Duże koła odpowiadają „obszarom wpływu” dotyczących wierzchołka przy kącie ostrym deltoidu (kolor pomarańczowy

na Rysunku 31). Obszary te mają największy wpływ na wynik, z uwagi na to, że są największe ze wszystkich i pokrywają znacząco duży obszar deltoidów. Warto również zauważyć, że zawsze znajdują się w wierzchołkach siatki trójkątnej, w którą wpisana jest siatka używana w tych metodach. Jeśli chodzi o wystające „żebry”, dla których wariancja również jest duża, to odpowiadają one „obszaram wpływu” w kształcie elips, a dokładniej fragmentów, które są bliskie wierzchołka przy kącie ostrym deltoidu. Mniejsze okręgi odpowiadają wierzchołkom przy kącie rozwartym płytce. Wariancja szumu kapeluszowego też ma dominujące fragmenty w kształcie kół zębacych. Różnicą w tym wypadku jest zwiększenie na co drugim „żebie” dominujących obszarów. Jest to wynik dodatkowych „obszarów wpływu” dodanych w tej metodzie. Warto zauważyć, że zwiększenie nie są zawsze na tych samych „żebach”.



Rysunek 36: Średnia nowych szumów.

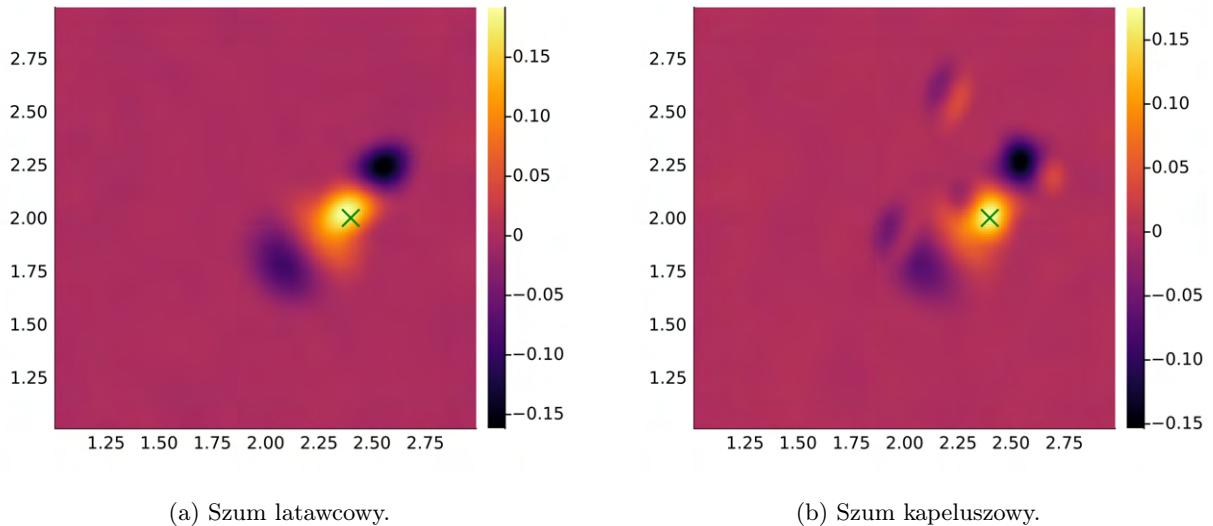


Rysunek 37: Wariancja nowych szumów.

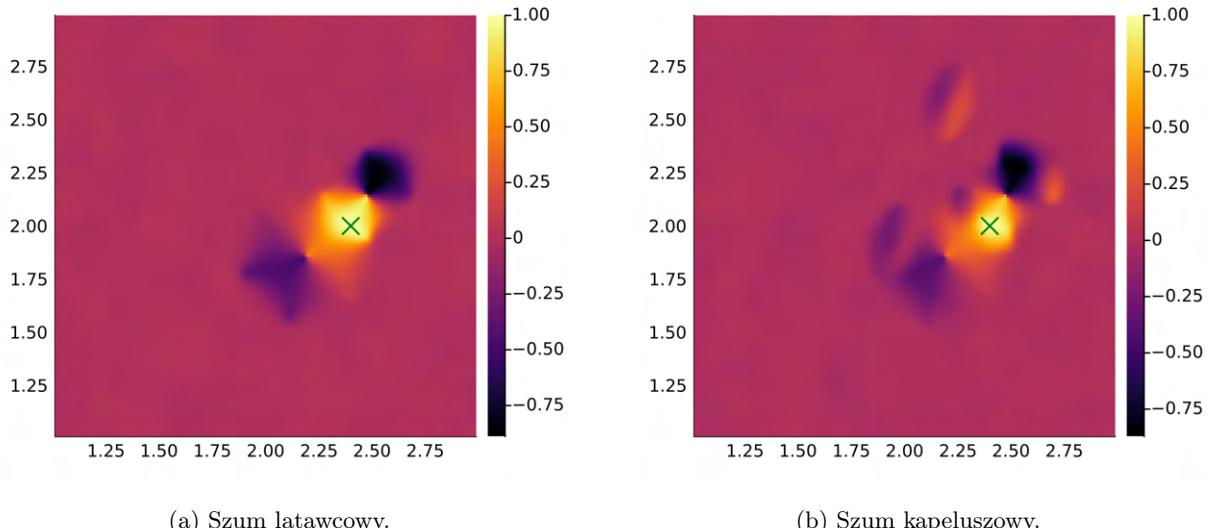
Wyestymowano również funkcje autokowariancji (Rys. 38) oraz autokorelacji (Rys. 39) nowych pól losowych. Na wszystkich wykresach przedstawiono wyestymowane funkcje dla ustalonego punktu zaznaczonego zielonym „x”, będącego w środku komórki i wszystkich innych punktów na obszarze $[1, 3] \times [1, 3]$. Wnioski

są podobne do tych wyciągniętych w sekcji 3.1, w przypadku szumów Perlina i Simplex. Widać ujemną zależność między komórkami sąsiadującymi ze sobą. W przypadku szumu kapeluszowego pojawiają się jeszcze dodatkowe obszary. Wynika to z dodanych wierzchołków w niektórych bokach deltoidów należących do płytka kapelusz. Wartość funkcji autokowariancji i autokorelacji dla tych obszarów jest mniejsza niż dla wartości w komórkach sąsiadujących z komórką, w której jest ustalony punkt, ponieważ zależność ta związana jest z tylko jednym „obszarem wpływu”.

Nie przedstawiono histogramów estymujących gęstość prawdopodobieństwa pól losowych w punktach tak jak w przypadku klasycznych metod. Wynika to z podobieństwa do histogramów szumu Simplex. W każdym sprawdzonym przypadku rozkład był dwumodalny. Podobieństwo do szumu Simplex spowodowane jest podobieństwem metody, jaka używana jest do generowania szumów.



Rysunek 38: Autokowariancja nowych szumów.



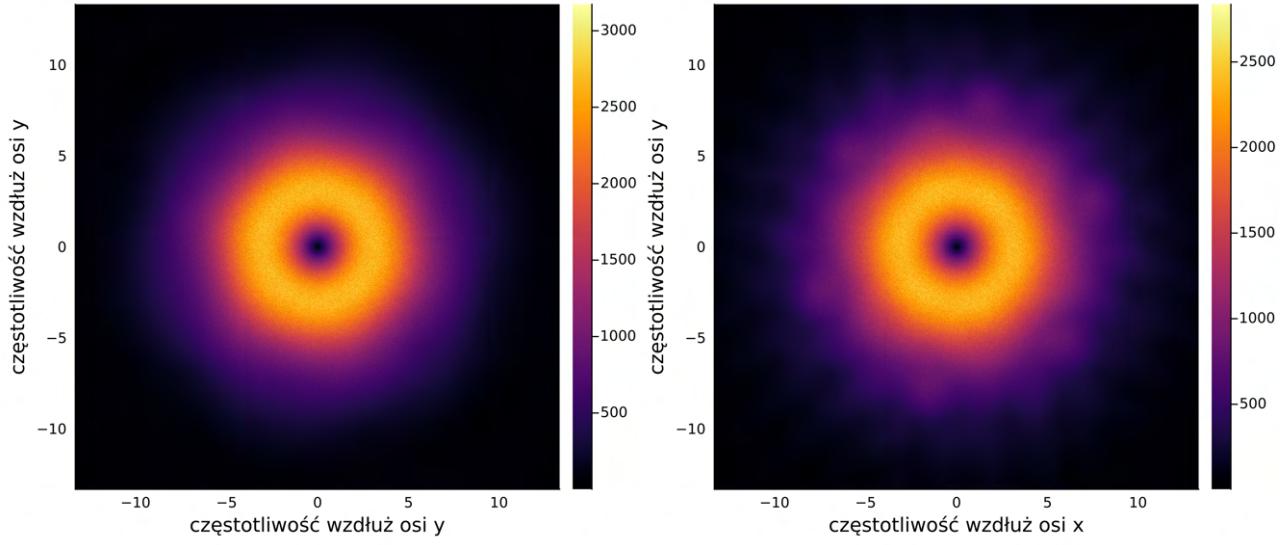
Rysunek 39: Autokorelacja nowych szumów.

6.2 Analiza częstotliwości

Aby stwierdzić, czy zaproponowana metoda rzeczywiście generuje szum izotropowy, wykonano analizę częstotliwości za pomocą transformaty Fouriera. W sekcji 3.2 zdefiniowano postać tej transformaty dla realizacji pola losowego. Tak samo jak w przypadku klasycznych szumów, transformata wyliczona jest jako średnia transformat z wielu realizacji pól losowych wygenerowanych na obszarze kwadratowym $[0, 100] \times [0, 100]$ biorąc 2000×2000 równoodległych wzduż osi x i y wartości. Aby uzyskać amplitudy występujących częstotliwości, wzięto moduł z transformaty.

Przypadek dwuwymiarowy przedstawiony na Rysunku 40 wyliczono na podstawie średniej modułów ze stu transformat realizacji pól losowych. Widmo przedstawione na tych wykresach nie wygląda, jakby było transformatą Fouriera szumów izotropowych. Dla metody opartej na siatce deltoidowej widmo przypomina kształtem zaokrąglony sześciian foremny. Porównując je do widma szumu Perlina, które przypominało zaokrąglony kwadrat, możemy stwierdzić, że jest bliższe izotropii, ale wypada gorzej niż szum Simplex, w którego dwuwymiarowej transformacie Fouriera nie dało się wzrokowo zauważać braku izotropii. W przypadku metody opartej na aperiodycznej siatce, widzimy, że widmo jest bardziej nieregularne niż w dowolnym wcześniej omawianym przypadku. Ciężko stwierdzić, czy tak wyglądająca transformata Fouriera jest bliższa izotropowej niż transformaty dla innych omawianych metod. Kształt wygląda, jakby miał symetrię szóstego rzędu, co jest szczególnie wyraźne w ogonach widma, które jednak z uwagi na swoje stosunkowo małe wartości nie wpływają aż tak mocno na szum. Najprawdopodobniej widmo rzeczywiście jest bliskie takiej symetrii, tak jak widmo szumu latawcowego, ponieważ oba pola losowe bazują na tej samej siatce, ale dla szumu kapeluszowego dodaje się nieregularności za pomocą dodatkowych wierzchołków siatki. Taki kształt może też wynikać z tego, że transformatę liczono na skończonym obszarze.

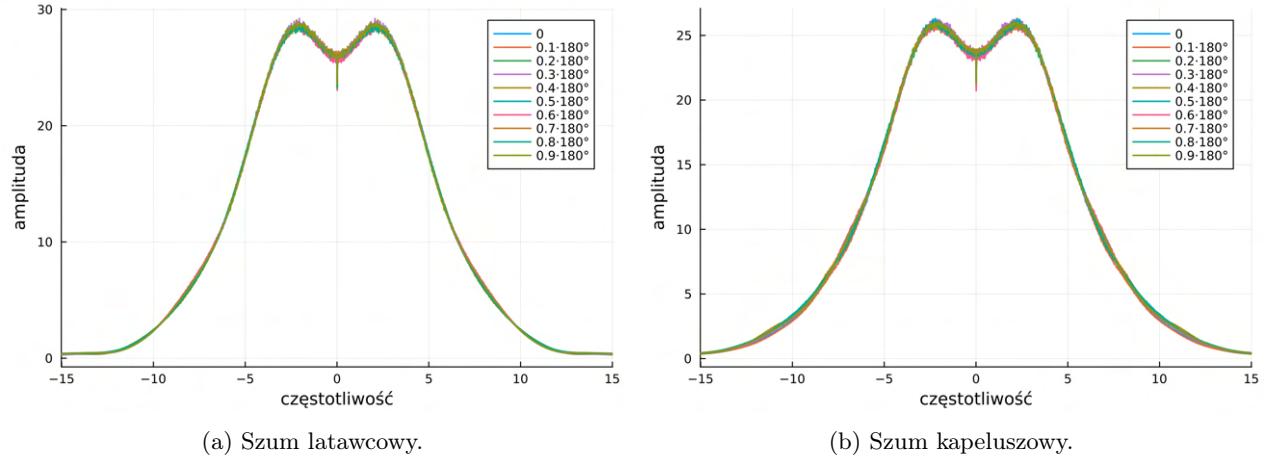
Na Rysunku 41 przedstawiono porównanie jednowymiarowych transformat Fouriera wzduż różnych kierunków. Dla każdego kierunku wzięto średnią z $2 \cdot 10^4$ widm realizacji pól losowych. Na wykresach przedstawiono transformatę dla dziesięciu kierunków, o kątach nachylenia rozłożonych równomiernie od 0° do 180° . Zdecydowano się na takie nachylenie jednowymiarowych wycinków, ponieważ siatka, na której generowane są nowe szумy, jest bardziej złożona niż siatki dla klasycznych metod, przez co nie było wiadomo, w których przypadkach różnice powinny być największe, a testując wycinki wzduż boków siatki przeciwko tym wzduż przekątnych siatki nie było widocznych różnic. Sprawdzono więcej wartości pośrednich nachylenia wycinków szumu, ale nie zauważono różnic, więc narysowano tylko 10 przypadków. Jak widać, widma we wszystkich przypadkach są sobie bardzo bliskie. Niespodziewanym wynikiem jest bliskość pokrywania się transformat dla szumu latawcowego, który bazuje na siatce, która jest okresowa. W przeciwnieństwie jednak do siatki kwadratowej i regularnej trójkątnej nie jest to siatka, w której w każdym wierzchołku spotyka się taka sama grupa figur, przez co okres w odległościach między bokami siatki wzduż różnych kierunków występuje, ale jest znacznie dłuższy niż w przypadku klasycznych siatek, co powoduje mniejsze różnice w rozchodzeniu się szumu w różnych kierunkach. Jednowymiarowe transformaty Fouriera szumu kapeluszowego, jak widać, również nie przejawiają dużych różnic. Można dodatkowo stwierdzić, że różnice są trochę większe niż dla szumu latawcowego, co widać dla mniejszych wartości widma. Może być to spowodowane ograniczonym obszarem, na którym generowano realizację szumu. Warto również zauważać, że w przeciwnieństwie do widm szumu Perlina i Simplex nie występują lub są mniej zauważalne nagłe spadki amplitudy spowodowane okresem samej siatki. Związane jest to z użyciem bardziej skomplikowanych siatek do generowania nowych szumów.



(a) Szum latawcowy.

(b) Szum kapeluszowy.

Rysunek 40: Dwuwymiarowa transformata Fouriera nowych szumów.



(a) Szum latawcowy.

(b) Szum kapeluszowy.

Rysunek 41: Transformata Fouriera jednowymiarowych wycinków nowych szumów.

7 Podsumowanie

Celem i głównym tematem pracy było zaprojektowanie metody generowania dwuwymiarowych pól losowych, która poprawia problem szumów Perlina i Simplex związanych z brakiem izotropii ich realizacji, jednocześnie nie tracąc szybkości i efektywności tych metod.

W ramach pracy przedstawiono i wytlumaczono algorytmy stojące za szumami Perlina i Simplex. Przeanalizowano ich cechy statystyczne (średnią, wariancję, autokowariancję i autokorelację) oraz wyestymowano gęstość w wybranych punktach pól losowych. Wykonano również analizę częstotliwości za pomocą dyskretnej transformaty Fouriera, która wskazała brak izotropii zarówno szumu Perlina, jak i szumu Simplex. Pomyślem na poprawienie braku izotropii było wykorzystanie aperiodycznego parkietużu jedną płytka do stworzenia siatki, na której bazować będzie nowy szum. Szczegółowo przedstawiono algorytm, służący do generowania parkietużu. Dzięki nowej siatce stworzono nowy szum nazwany kapeluszowym oraz jako efekt uboczny generowania siatki powstała jedna metoda nazywana szumem latawcowym. Podobnie jak klasyczne metody

zbadano też te nowe pod względem cech statystycznych. Na końcu wykonano analizę częstotliwości nowych szumów, która potwierdziła, że poprawiono problemy klasycznych metod.

Założony cel pracy udało się osiągnąć. Nowe metody generują pola losowe o dużym stopniu izotropii. Algorytmy jednocześnie dalej pozostają szybkie i efektywne oraz pozwalają na implementację wielowątkową, która jest potrzebna, ponieważ zwykle takie metody implementuje się, używając kart graficznych.

Wykorzystanie aperiodycznego parkietu i zastosowanie go w rzeczywistym problemie jest ciekawe samo w sobie. Omawiana siatka jest nowo opracowanym (2023) wyidealizowanym matematycznym obiektem, który dotychczas nie znalazł wielu zastosowań. W kontekście generowania terenu praca na aperiodycznej siatce ma potencjał lepiej oddawać wiele wzorców występujących w naturze, których przedstawienie jest celem omówionych metod.

W przyszłości warto rozważyć metody generowania izotropowych szumów dla więcej niż dwuwymiarowych przestrzeni. Jest to znacznie trudniejsze, ponieważ aperiodiczne parkietu n-wymiarowych przestrzeni zwykle nie nadają się do efektywnej implementacji potrzebnej w przypadku szumów, których generowanie ma być szybkie. Istnieje jednak parkietu n-wymiarowych przestrzeni bazujący na siatce hipersześciennej [7], który można wykorzystać do efektywnego generowania szumu. Dodatkowo można wykorzystać odpowiednie periodyczne siatki, których okresy w różnych kierunkach są większe niż w przypadku siatki kwadratowej i regularnej trójkątnej.

Do implementacji algorytmów i wykonanej analizy wykorzystano język programowania Julia wraz z bibliotekami Statistics, StatsBase, FFTW oraz Measures i Plots do wizualizacji i wykresów. Kod napisany na potrzeby pracy dostępny jest w repozytorium github.com/knaimadd/IsotropicNoise.

Literatura

- [1] H. Anton and C. Rorres. *Elementary linear algebra: applications version*. Wiley, 2013.
- [2] J. Aycock. *Procedural Content Generation*, page 109–143. Springer Publishing, 2016.
- [3] P. Brémaud. Fourier analysis of stochastic processes. In *Fourier analysis and stochastic processes*, pages 119–179. Springer Publishing, 2014.
- [4] N. Brewer. Computerized dungeons and randomly generated worlds: From rogue to minecraft [scanning our past]. *Proceedings of the IEEE*, 105(5):970–977, 2017.
- [5] J. Brown and M. Scirea. *Procedural Generation for Tabletop Games: User Driven Approaches with Restrictions on Computational Resources*, pages 44–54. Springer Publishing, 2020.
- [6] H. S. M. Coxeter. *Introduction to Geometry (2nd ed.)*. Wiley, 1969.
- [7] C. Goodman-Strauss. An aperiodic pair of tiles inenfor alln 3. *European Journal of Combinatorics*, 20(5):385–395, 1999.
- [8] S. Gustavson and I. McEwan. Tiling simplex noise and flow noise in two and three dimensions. *Journal of Computer Graphics Techniques (JCGT)*, 11(1):17–33, 2022.
- [9] C. Hermite. Sur la formule d'interpolation de lagrange. *Journal für die reine und angewandte Mathematik*, 84:70–79, 1877.
- [10] R. Penrose. The role of aesthetics in pure and applied mathematical research. *Bulletin of the Institute of Mathematics and Its Applications*, 10:266–271, 1974.
- [11] K. Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, 1985.
- [12] K. Perlin. Noise hardware. In *Real-Time Shading SIGGRAPH Course Notes*, 2001.
- [13] J. E. T. Robert J. Adler. *Random Fields and Geometry*. Springer Publishing, 2010.
- [14] D. Smith, J. S. Myers, C. S. Kaplan, and C. Goodman-Strauss. An aperiodic monotile. *Combinatorial Theory*, 4(1), 2024.

- [15] D. Smith, J. S. Myers, C. S. Kaplan, and C. Goodman-Strauss. A chiral aperiodic monotile. *Combinatorial Theory*, 4(2), 2024.
- [16] G. Smith. *An analog history of procedural content generation*. In *Proceedings of the 2015 Conference on the Foundations of Digital Games (FDG)*. Springer Publishing, 2015.
- [17] Student. The probable error of a mean. *Biometrika*, 1908.
- [18] H. Wang. Proving theorems by pattern recognition — ii. *The Bell System Technical Journal*, 40(1):1–41, 1961.
- [19] H. Weil. *Symmetry*. Princeton University Press, 1952.