

Моделирование и анализ случайного блуждания на языке Rust

Кокухина Софья Александровна

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

Целью исследования является реализация модели случайного блуждания на языке программирования Rust. Программа позволяет моделировать одномерное и двумерное случайное блуждание, анализировать статистические характеристики процесса, включая среднеквадратичное смещение, и визуализировать результаты.

Ключевые слова: Rust, случайное блуждание, стохастический процесс, моделирование, визуализация, среднеквадратичное смещение.

Random Walk Simulation in Rust

Kokukhina Sofya Alexandrovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the study is to implement a random walk model in the Rust programming language. The program allows you to simulate one-dimensional and two-dimensional random walks, analyze the statistical characteristics of the process, including the root-mean-square shift, and visualize the results.

Keywords Rust, random walk, stochastic process, simulation, visualization, root-mean-square displacement.

1 Введение

1.1 Актуальность

Случайное блуждание — одна из фундаментальных моделей в физике, математике, биологии и экономике. Оно описывает движение частицы, совершающей случайные шаги, и лежит в основе таких явлений, как броуновское движение, диффузия, финансовые временные ряды и поисковые алгоритмы в графах.

Несмотря на простоту, модель демонстрирует нетривиальные статистические свойства, например, рост среднеквадратичного смещения пропорционально квадратному корню из времени.

1.2 Обзор исследований

В работе К.Е. Самуйлова выполнен обзор моделей случайного блуждания объектов, применяемых при моделировании передвижения

приемопередающих устройств пользователей беспроводной сети пятого поколения (5G) [1]. Рассмотрены модели мобильности, характерные для имитационного моделирования поведения пользователей беспроводной самоорганизующейся сети. [1]. В исследовании К.П. Трубаева анализируются современные системные языки, включая Rust, с акцентом на их применение в высокопроизводительных вычислениях. Отмечается, что Rust обеспечивает безопасность без ущерба для скорости [2]. Д.М. Берлизов демонстрирует использование Rust в многопоточных приложениях, что подчёркивает его потенциал для параллельного моделирования сложных систем [3]. В.С.Широченко указывает на растущее применение Rust в задачах, где критичны безопасность и надёжность, включая научные симуляции [4].

1.3 Цель исследования

Цель исследования заключается не только в разработке модели случайного блуждания на языке Rust, но и в демонстрации возможностей современного системного языка программирования для решения задач научного моделирования. В рамках работы ставятся задачи: изучить основы организации вычислительного эксперимента в Rust, освоить использование внешней библиотеки `rand` для генерации случайных чисел, реализовать алгоритм одномерного и двумерного случайного блуждания, проанализировать статистические характеристики процесса (включая среднеквадратичное смещение), а также выполнить визуализацию результатов с помощью библиотеки `plotters`. Работа направлена на демонстрацию безопасности, производительности и воспроизводимости кода на Rust в контексте научных вычислений.

2 Материалы и методы

Для реализации программы используется язык программирования Rust версии 1.79.0, установленный через менеджер пакетов `rustup`. В комплект поставки входят компилятор `rustc` и система сборки проектов `Cargo`, которая позволяет быстро создавать, управлять зависимостями и собирать проекты. Использование `Cargo` обеспечивает высокую степень воспроизводимости и удобство подключения сторонних библиотек.

В качестве среды разработки применяется `Visual Studio Code`, дополненный расширением `rust-analyzer`, обеспечивающим подсветку синтаксиса, автодополнение кода, проверку типов и подсказки по ошибкам в реальном времени. Такой набор инструментов является оптимальным для начинающих разработчиков, так как сочетает простоту использования и достаточную функциональность для реализации научных приложений.

На первом этапе подготовки создаётся новый консольный проект командой `cargo new random_walk --bin` (Рис.1).

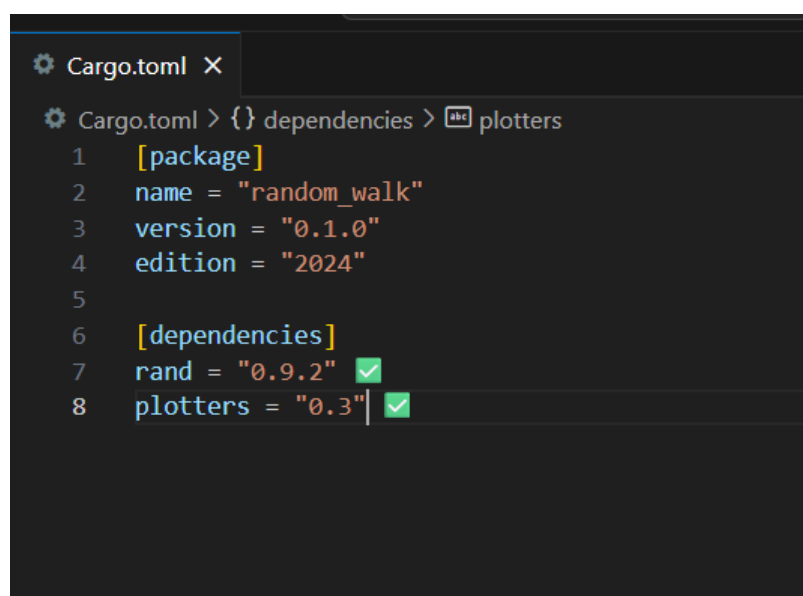
```
C:\Windows\System32>cd C:\Users\Nktt\Documents\Учеба\RustApp

C:\Users\Nktt\Documents\Учеба\RustApp> cargo new random_walk --bin
   Creating binary (application) `random_walk` package
note: see more `Cargo.toml` keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html

C:\Users\Nktt\Documents\Учеба\RustApp>
```

Рисунок 1 – Создание проекта в консоли

Следующим шагом осуществляется подключение внешних библиотек. Для генерации случайных чисел, необходимых для моделирования случайных шагов частицы, используется крейт `rand`. Для визуализации результатов — построения траектории блуждания и графика среднеквадратичного смещения — применяется библиотека `plotters`. Эти зависимости добавляются в файл `Cargo.toml` (Рис.2).



```
Cargo.toml X
Cargo.toml > {} dependencies > plotters
1 [package]
2 name = "random_walk"
3 version = "0.1.0"
4 edition = "2024"
5
6 [dependencies]
7 rand = "0.9.2" ✓
8 plotters = "0.3" ✓
```

Рисунок 2 – Добавление зависимостей в Cargo.toml

3 Результаты

После подготовки проекта и подключения необходимых библиотек можно переходить к реализации модели случайного блуждания. Программа реализует два ключевых аспекта: моделирование траектории блуждания и статистический анализ его свойств. Процесс разработки включает несколько этапов: генерацию случайных шагов, расчёт положения частицы, вычисление среднеквадратичного смещения (RMS) и визуализацию результатов.

3.1 Моделирование двумерного случайного блуждания

Для демонстрации траектории движения реализовано одиночное двумерное случайное блуждание. В основной функции `main` инициализируются начальные координаты частицы. На каждом шаге направление движения выбирается случайным образом в диапазоне от 0 до 2π

с использованием генератора случайных чисел из библиотеки `rand`. Для этого применяется метод `gen_range()` (Рис.3).

```
9  ✓ fn main() -> Result<(), Box<dyn std::error::Error>> {
10      // === Часть 1: Визуализация одной траектории ===
11      let mut rng: ThreadRng = rand::thread_rng();
12      let mut x: f64 = 0.0;
13      let mut y: f64 = 0.0;
14
15      let mut path_x: Vec<f64> = vec![x];
16      let mut path_y: Vec<f64> = vec![y];
17
18      // Один прогул
19  ✓  for _ in 0..N_STEPS {
20      let angle: f64 = rng.gen_range(0.0..std::f64::consts::PI * 2.0);
21      x += angle.cos();
22      y += angle.sin();
23
24      path_x.push(x);
25      path_y.push(y);
26  }
27
```

Рисунок 3 – Реализация двумерного случайного блуждания

3.2 Визуализация траектории

Для построения графика траектории используется библиотека `plotters`. Создаётся объект `BitMapBackend`, который позволяет сохранять изображение в формате PNG. С помощью `ChartBuilder` настраивается область рисования, подписи осей и заголовок графика. Траектория отображается как линейная серия точек, соединённых между собой. График автоматически сохраняется в файл `random_walk_2d_trajectory.png`. Полученная визуализация наглядно демонстрирует хаотичный характер движения частицы, типичный для диффузионных процессов (Рис. 4).

```

28 // Рисуем траекторию
29 let root_drawing_area: DrawingArea<BitMapBackend<'>, ...> =
30     BitMapBackend::new(path: "random_walk_2d_trajectory.png", (800, 600)).into_drawing_area();
31 root_drawing_area.fill(color: &WHITE?);
32 let mut ctx: ChartContext<'_, BitMapBackend<'>, ...> = ChartBuilder::on(&root_drawing_area) Cha...
33     .caption(caption: "2D Random Walk Trajectory", style: ("sans-serif", 30)) &mut ChartBuilder<'_, '...
34     .margin(size: 10) &mut ChartBuilder<'_, '_, BitMapBacken...
35     .x_label_area_size(40) &mut ChartBuilder<'_, '_, BitMapBackend<'>...
36     .y_label_area_size(40) &mut ChartBuilder<'_, '_, BitMapBackend<'>...
37     .build_cartesian_2d(x_spec: -50f64..50f64, y_spec: -50f64..50f64)?;
38
39 ctx.configure_mesh().draw()?;
40
41 ctx.draw_series(LineSeries::new(
42     iter: path_x.iter().zip(path_y.iter()).map(|(&x: f64, &y: f64)| (x, y)),
43     style: &RED,
44 ))? &mut SeriesAnno<'_, BitMapBackend<'>>
45 .label("Trajectory") &mut SeriesAnno<'_, BitMapBackend<'>>
46 .legend(func: |(x: i32, y: i32)| PathElement::new(points: vec![(x, y), (x + 20, y)], style: &RED));
47
48 ctx.configure_series_labels() SeriesLabelStyle<'_, '_, ..., ...>
49     .background_style(&WHITE.mix(0.8)) &mut SeriesLabelStyle<'_, '_, ..., ...>
50     .border_style(&BLACK) &mut SeriesLabelStyle<'_, '_, ..., ...>
51     .draw()?;
52
53 root_drawing_area.present()?;
54 println!("Траектория сохранена в random_walk_2d_trajectory.png");
55

```

Рисунок 4 – Визуализация траектории двумерного блуждания

3.4 Статистический анализ: расчет RMS

Для анализа статистических свойств модели выполняется серия из 1000 независимых блужданий, каждое по 1000 шагов. На каждом шаге вычисляется квадрат расстояния от начала координат. Значения накапливаются в массиве `squared_distances`, после чего усредняются по всем блужданиям. Среднеквадратичное смещение (RMS) определяется как корень из среднего квадрата расстояния. Теоретическое значение RMS для нормальной диффузии задаётся как $\sqrt{t}$ и добавляется для сравнения (Рис.5).

```

57 let mut squared_distances: Vec<f64> = vec![0.0; N_STEPS + 1];
58
59 for _ in 0..N_WALKS {
60     let mut x: f64 = 0.0;
61     let mut y: f64 = 0.0;
62
63     squared_distances[0] += x * x + y * y;
64
65     for step: usize in 1..N_STEPS {
66         let angle: f64 = rng.gen_range(0.0..2.0 * std::f64::consts::PI);
67         x += angle.cos();
68         y += angle.sin();
69
70         squared_distances[step] += x * x + y * y;
71     }
72 }
73
74 // Усредняем
75 for step: usize in 0..N_STEPS {
76     squared_distances[step] /= N_WALKS as f64;
77 }
78
79 // Сохраняем данные RMS
80 let mut file: File = File::create(path: "rms_data.csv").expect(msg: "Не удалось создать файл");
81 writeln!(file, "step,rms,theory_rms").unwrap();
82
83 for step: usize in 0..N_STEPS {
84     let rms: f64 = squared_distances[step].sqrt();
85     let theory_rms: f64 = (step as f64).sqrt(); // Для 2D RMS ~ √t
86     writeln!(file, "{},{},{}", step, rms, theory_rms).unwrap();
87 }
88

```

Рисунок 5 – Вычисление среднеквадратичного смещения

3.4 Визуализация результатов анализа

Результаты анализа визуализируются с помощью второй диаграммы, построенной аналогично первой. На график наносятся:

- синяя линия — смоделированное значение RMS;
- красная пунктирная линия — теоретическая зависимость $\sqrt{t}$.

Сравнение показывает высокую степень соответствия модели теории (Рис.6). Небольшие отклонения обусловлены статистическими флуктуациями, уменьшающимися с ростом числа блужданий.

```

let root_drawing_area: DrawingArea<BitMapBackend<'_, ...>, ...> = BitMapBackend::new(path: "rms_comparison.png", (800, 600)).into_drawing_area()
root_drawing_area.fill(color: &WHITE)?;
let mut ctx: ChartContext<'_, BitMapBackend<'_, ...>, ...> = ChartBuilder::on(&root_drawing_area) Cha...
    .caption(caption: "RMS Displacement vs Time", style: ("sans-serif", 30)) &mut ChartBuilder<'_, '_, BitMapBacken...
    .margin(size: 10) &mut ChartBuilder<'_, '_, BitMapBacken...
    .x_label_area_size(40) &mut ChartBuilder<'_, '_, BitMapBackend<'_, ...>...
    .y_label_area_size(60) &mut ChartBuilder<'_, '_, BitMapBackend<'_, ...>...
    .build_cartesian_2d(x_spec: 0f64..N_STEPS as f64, y_spec: 0f64..(N_STEPS as f64).sqrt() * 1.5)?;

ctx.configure_mesh().draw()?;

// Симулированные данные
ctx.draw_series(
    LineSeries::new(
        iter: (0..N_STEPS).map(|i: usize| (i as f64, squared_distances[i].sqrt())),
        style: &BLUE,
    ),
)? &mut SeriesAnno<'_, BitMapBackend<'_, ...>
.label("Simulated RMS") &mut SeriesAnno<'_, BitMapBackend<'_, ...>
.legend(func: |(x: i32, y: i32)| PathElement::new(points: vec![(x, y), (x + 20, y)], style: &BLUE));

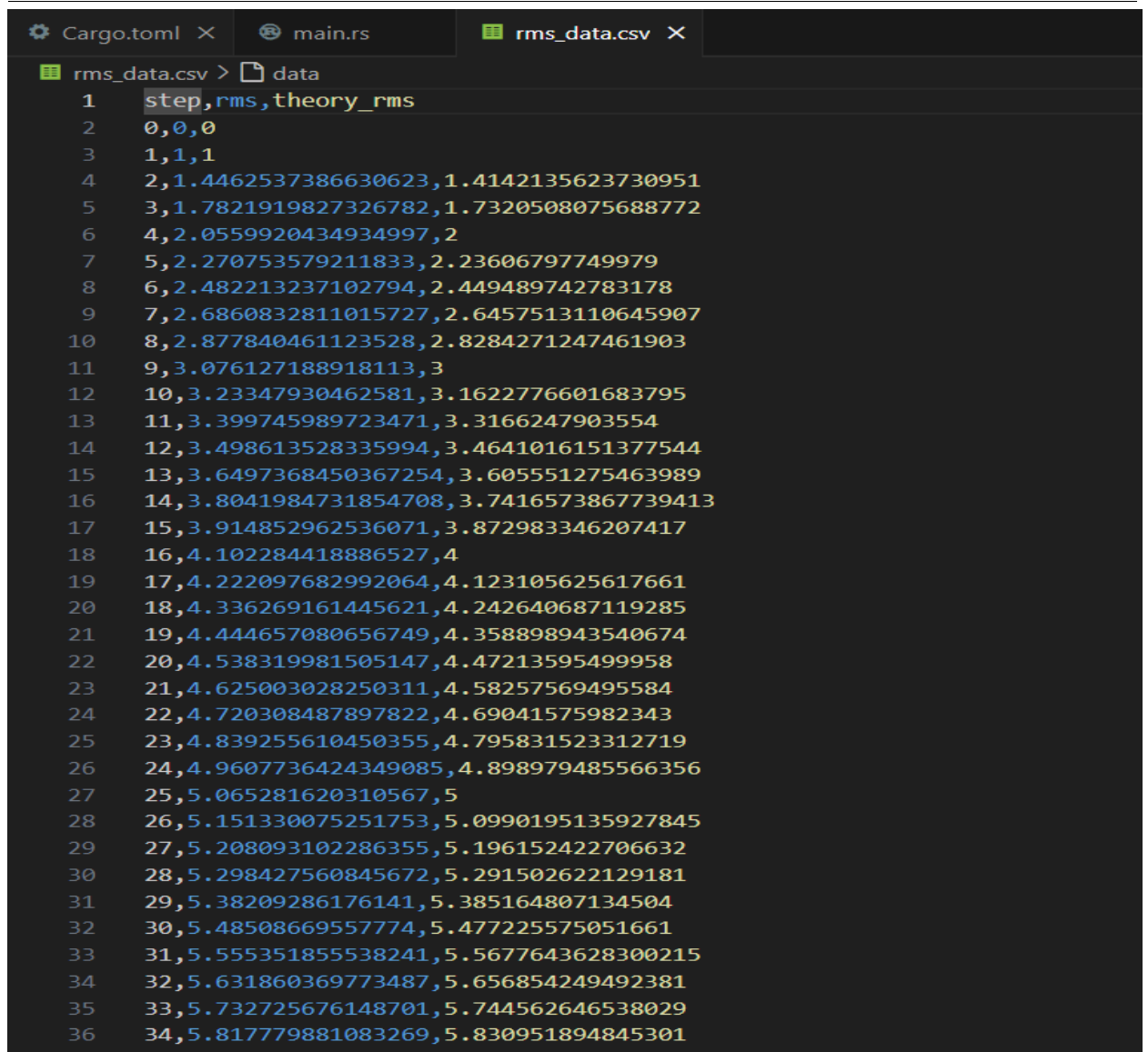
// Теоретическая кривая sqrt
ctx.draw_series(
    LineSeries::new(
        iter: (0..N_STEPS).map(|i: usize| (i as f64, (i as f64).sqrt())),
        style: &RED.dashed(),
    ),
)? &mut SeriesAnno<'_, BitMapBackend<'_, ...>
.label("Theory  $\sqrt{t}$ ") &mut SeriesAnno<'_, BitMapBackend<'_, ...>
.legend(func: |(x: i32, y: i32)| PathElement::new(points: vec![(x, y), (x + 20, y)], RED.stroke_width(2).stroke_style(StrokeStyle::new().l

ctx.configure_series_labels() SeriesLabelStyle<'_, '_, ...>
    .background_style(&WHITE.mix(0.8)) &mut SeriesLabelStyle<'_, '_, ...>
    .border_style(&BLACK) &mut SeriesLabelStyle<'_, '_, ...>
    .draw()?;

```

Рисунок 6 – График RMS: сравнение модели и теории

Все результаты анализа сохраняются в файл rms_data.csv в формате CSV (Рис.7) для возможности дальнейшей обработки в других программах (например, Python, Excel). После завершения кода программа запускается командой cargo run в консоли. В результате выполнения генерируются два графика (Рис.8), (Рис.9) и файл с данными, что позволяет провести полный анализ модели.



```
1 step,rms,theory_rms
2 0,0,0
3 1,1,1
4 2,1.4462537386630623,1.4142135623730951
5 3,1.7821919827326782,1.7320508075688772
6 4,2.0559920434934997,2
7 5,2.270753579211833,2.23606797749979
8 6,2.482213237102794,2.449489742783178
9 7,2.6860832811015727,2.6457513110645907
10 8,2.877840461123528,2.8284271247461903
11 9,3.076127188918113,3
12 10,3.23347930462581,3.1622776601683795
13 11,3.399745989723471,3.3166247903554
14 12,3.498613528335994,3.4641016151377544
15 13,3.6497368450367254,3.605551275463989
16 14,3.8041984731854708,3.7416573867739413
17 15,3.914852962536071,3.872983346207417
18 16,4.102284418886527,4
19 17,4.222097682992064,4.123105625617661
20 18,4.336269161445621,4.242640687119285
21 19,4.444657080656749,4.358898943540674
22 20,4.538319981505147,4.47213595499958
23 21,4.625003028250311,4.58257569495584
24 22,4.720308487897822,4.69041575982343
25 23,4.839255610450355,4.795831523312719
26 24,4.9607736424349085,4.898979485566356
27 25,5.065281620310567,5
28 26,5.151330075251753,5.0990195135927845
29 27,5.208093102286355,5.196152422706632
30 28,5.298427560845672,5.291502622129181
31 29,5.38209286176141,5.385164807134504
32 30,5.48508669557774,5.477225575051661
33 31,5.555351855538241,5.5677643628300215
34 32,5.631860369773487,5.656854249492381
35 33,5.732725676148701,5.744562646538029
36 34,5.817779881083269,5.830951894845301
```

Рисунок 7 – Пример файла rms_data.csv

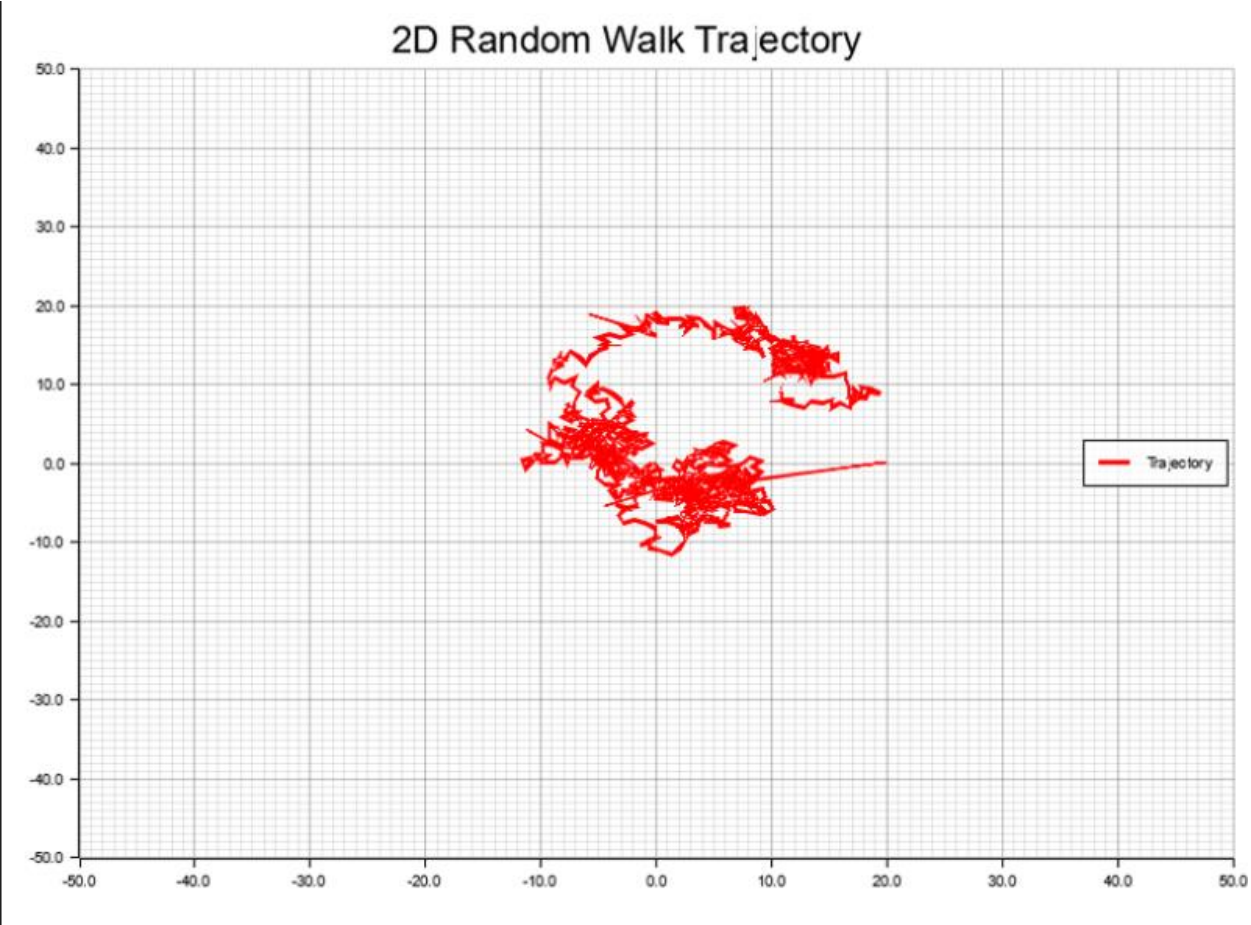


Рисунок 8 – Траектория случайного блуждения на графике

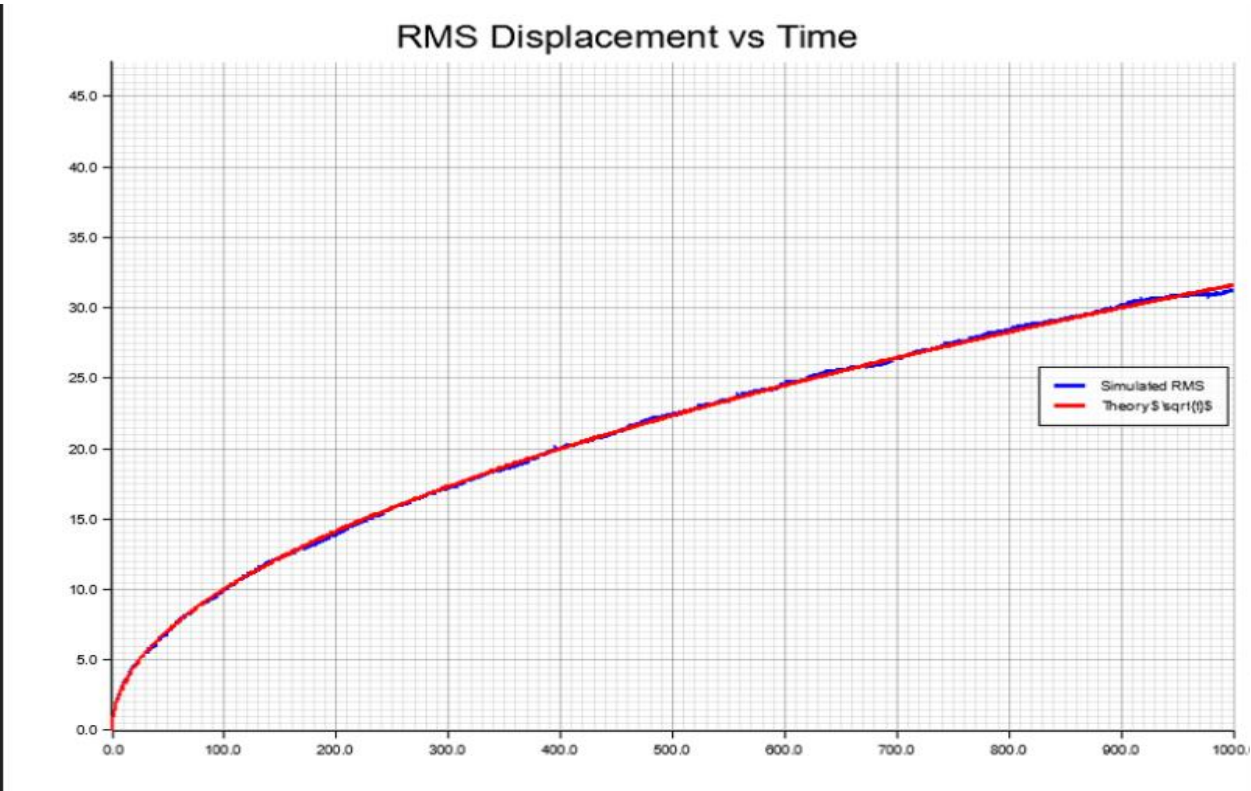


Рисунок 9 – Сравнение теоретической траектории и смоделированной

4

Выводы. Программа, реализованная на языке Rust с использованием библиотек `rand` и `plotters`, позволила смоделировать двумерное случайное блуждание. Генерация случайных направлений движения с помощью метода `random_range` и вычисление координат частицы на каждом шаге обеспечили корректное воспроизведение хаотичного характера движения, характерного для диффузионных процессов.

Программа успешно компилируется и выполняется с помощью команды `cargo run`, генерируя ожидаемые графические и числовые результаты. Исправление ошибок, связанных с устаревшими функциями библиотеки `rand` и корректной передачей стилей в `plotters`, обеспечило стабильную работу кода.

В целом, работа демонстрирует успешное применение программирования на Rust для моделирования и анализа случайного блуждания, подтверждая теоретические ожидания и предоставляя наглядные результаты в виде графиков и данных. Полученные результаты могут быть использованы для дальнейшего изучения диффузионных процессов или адаптации модели под другие задачи.

Библиографический список

1. Самуйлов К. Е., Гайдамака Ю. В., Шоргин С. Я. Применение моделей случайного блуждания при моделировании перемещения устройств в беспроводной сети // Информатика и ее применения. 2018. Т. 12. № 4. С. 2-8.
2. Трубаев К.П. Современные языки программирования на примере языка Rust // Образование. Наука. Производство. Белгород: Белгородский государственный технологический университет им. В.Г. Шухова, 2022. С. 197-200.
3. Берлизов Д.М. Оптимизация выполнения критических секций на языке программирования Rust // Проблемы оптимизации сложных систем. Новосибирск: Институт вычислительной математики и математической геофизики Сибирского отделения РАН, 2016. С. 65-72.
4. Широченко В.С. Использование языка программирования Rust для высокопроизводительных и безопасных вычислений // Инновационные научные исследования в современном мире. Уфа: Общество с ограниченной ответственностью "Научно-издательский центр "Вестник науки", 2023. С. 238-241.