

University of Hull

600093 - Computational Science

Cellular Automata

Nathan Rignall

April 30, 2024

Word count: 1885

Contents

1	Introduction	1
2	Methodology	2
2.1	Cell Movement	2
2.1.1	Simulation Plots	2
2.1.2	Simulation Heatmap	3
2.1.3	Movement Complexity	5
2.1.4	Simulation Steps	6
2.2	Cell Growth	8
2.2.1	Accuracy of the Simulation	9
2.2.2	Model Complexity	10
2.2.3	Changing the Capacity	11
2.2.4	Random Walk	13
2.2.5	Changing the Grid Size	14
3	Conclusion	15
	List of Figures	16
	Bibliography	17
A	Appendix	18
A.1	Integration of Growth Model	18

1 | Introduction

This report shall show the steps taken to simulate the growth of cancer cells in a tissue using the Gompertz Equation to model growth.

Growth for individual cells can be modelled using a differential equation so that the number of cells is a function of time, shown in Equation 1.1. The Gompertz Model of Cell Growth works well in this application because it can accurately model the non exponential nature of cell growth (Tatro 2018). Once an area has reached it's capacity no more tumour cells can form, hence the cells must begin to move through a tissue. This movement can be simulated using simple random walk algorithms (Codling et al. 2008).

$$\frac{dN}{dt} = kN \ln \left(\frac{M}{N} \right) \quad (1.1)$$

This paper specifically shall analyse the computational complexity of such models and evaluate the differences between simulation techniques. The underlying implementation shall be investigated, including random algorithms and accuracy of simulations on computer systems.

Simulations shall use Rust as the programming language, for its performance properties as a low level language (Adam 2023). It also allows for fine control over types, which can be useful for numerical simulations. Fine-grained benchmarks shall be used to evaluate the performance of the simulations using the criterion library (Heisler 2024). Analysis of data generated shall be done using Python with Matplotlib and Numpy.

2 | Methodology

2.1 Cell Movement

The random movement of cells can be modelled using a probability distribution. Probability distributions that can be classified as discrete or continuous, we will analyse uniform and bernoulli type distributions. The uniform distribution is usually regarded as a continuous distribution but can also be used to model discrete variables, all values are equally likely to be selected. The Bernoulli distribution is a discrete distribution that models a single trial with two outcomes, the probability of the two outcomes is p and $1 - p$.

When all directions are equally probable a uniform distribution should be used. If a Bernoulli distribution was selected, it would introduce a bias to the direction of movement. This could be desired in a complex model where factors such as surface tension affect the direction taken by the cell.

The algorithm for random walk shall generate a list of available directions and select one at random. Two different methods of movement shall be compared, a square movement and a diagonal movement.

2.1.1 Simulation Plots

Both directions have bias; square moves to the right, diagonal moves downwards. This is because it is a single run and the number generation is random, hence direction is random. In Figure 2.1 both plots show the same number of steps, but the diagonal plot moves further than the square plot.

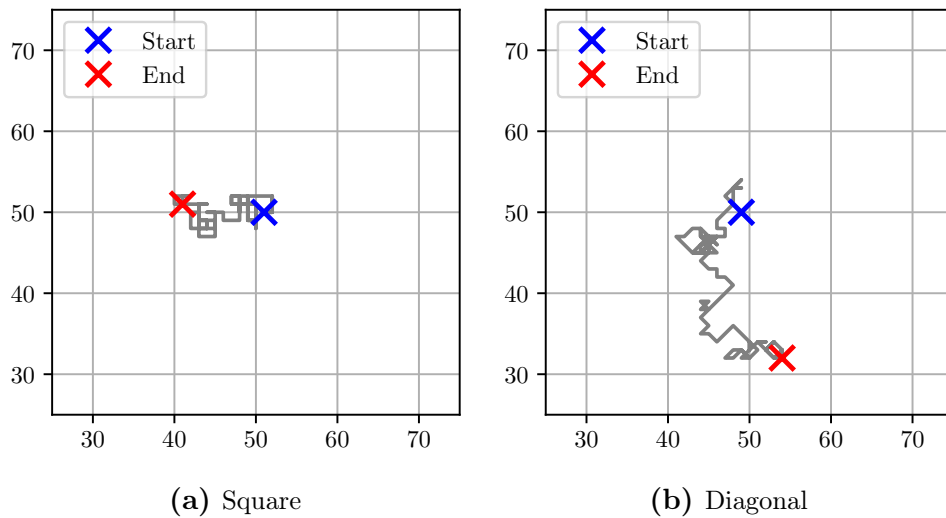


Figure 2.1: Cell movement plots

2.1.2 Simulation Heatmap

The movement of the cells can be visualised using a heatmap. A number of start and end points can be selected using another uniform distribution. Next the random walk can execute until the end point is reached (Figure 2.3), this process can be repeated multiple times to record the visited cells.

The results are shown in Figure 2.2, there are small hotspots but overall the distribution is uniform. With more iterations, the distribution would become more even. This validates the uniform distribution of both the square and diagonal movements.

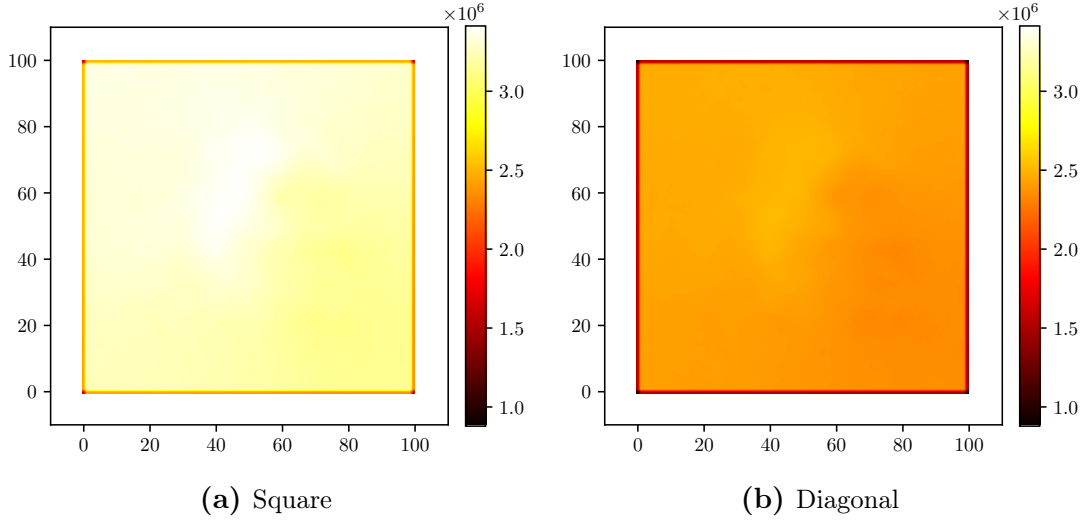


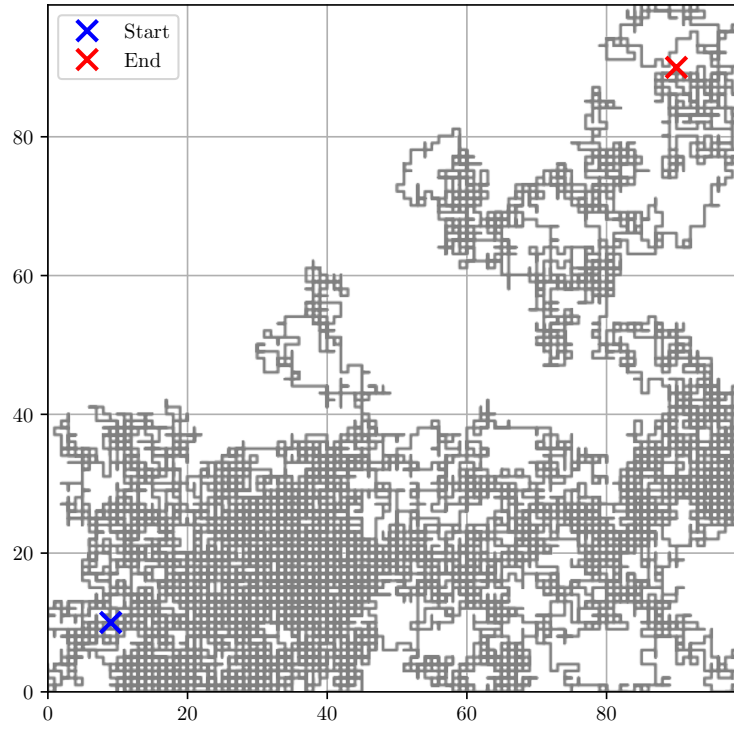
Figure 2.2: Visited cells simulation heatmap

There is a significant difference in the number of visited cells between the square and diagonal movements. This is because the diagonal movement can move further in a single step than the square movement.

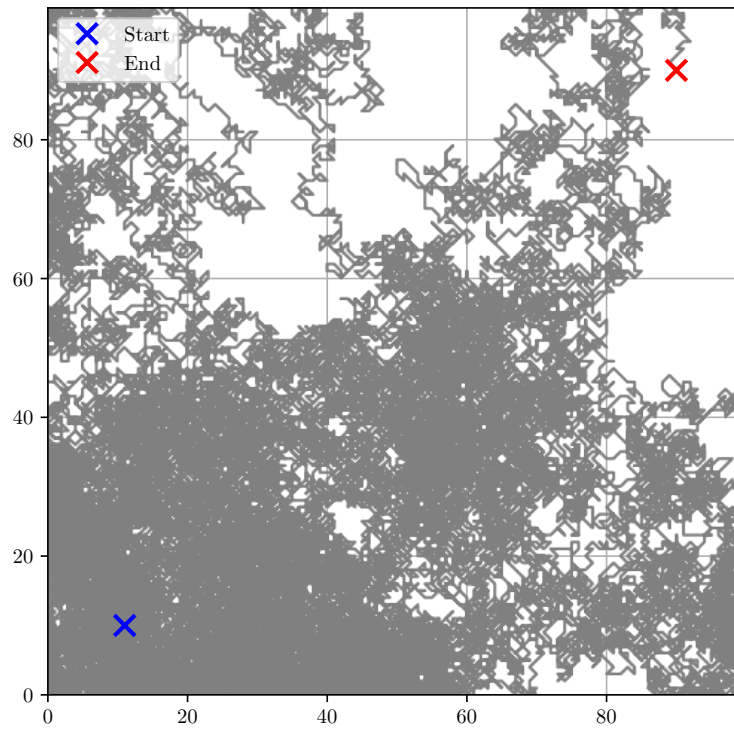
The square can perform one step to the left, right, up or down. The diagonal can perform one step to the left, right, up, down, up-left, up-right, down-left or down-right. Any diagonal movement is equivalent to $\sqrt{1^2 + 1^2} = \sqrt{2}$ square movements.

Hence square movement is on average $(1 \times 4)/4 = 1$ steps. The diagonal movement is on average $(\sqrt{2} \times 4 + 1 \times 4)/8 = 1.2071$ steps. Square is a factor of $1/1.2071 = 0.8284$ compared to diagonal.

In the scene of moving from one point to another, the diagonal movement is more efficient than the square movement. This is different from computational complexity.



(a) Square



(b) Diagonal

Figure 2.3: Cell movement fill plots

2.1.3 Movement Complexity

Using criterion, the computational complexity of the different movements can be measured. The results are shown in Figure 2.4. Square movements take $16.2ns$ compared to diagonal movements which take $42.7ns$, this is a ratio of 0.3794. The diagonal movement is more computationally complex than square movement.

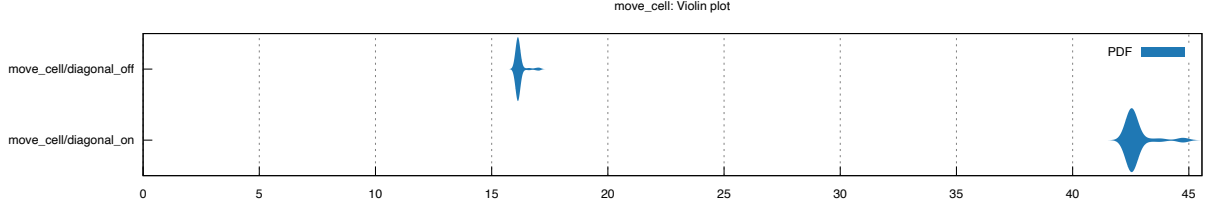


Figure 2.4: Cell movement criterion

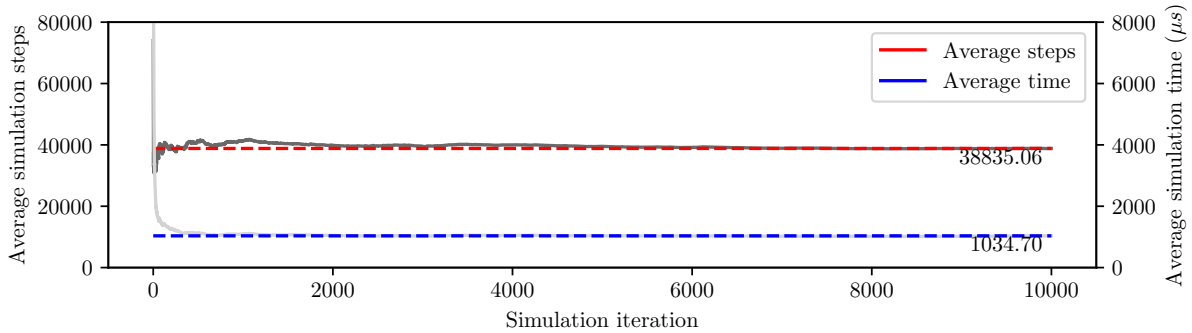
This is because the diagonal movement requires more checks to ensure the cell does not move off the grid. The act of generating a random number is the same for both movements as a single random number is generated to determine the direction of movement.

As single move is of complexity $O(1)$, the complexity of multiple moves is $O(n)$ where n is the number of steps taken.

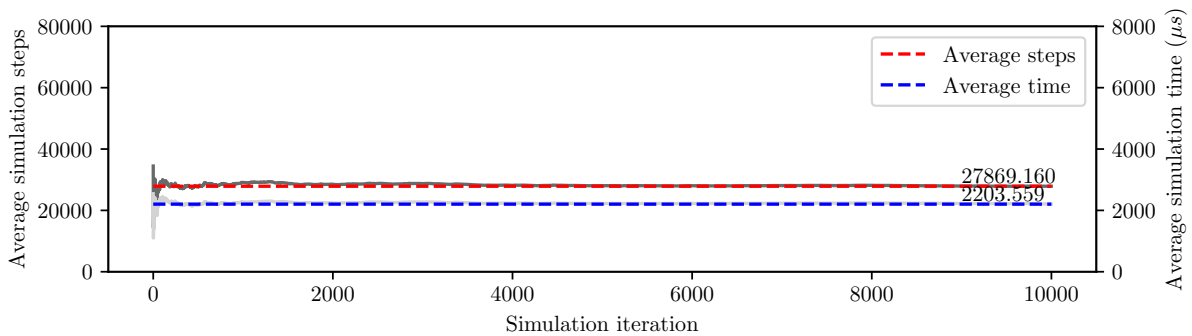
2.1.4 Simulation Steps

By selecting a random start and end point, we can compare the number of steps and time taken by the square and diagonal movements directly. If this movement simulation is performed multiple times, the average number of steps and time taken by the two movements can be compared.

Given enough iterations, the average should stabilise and the difference in the number of steps and time taken by the square and diagonal movements can be calculated.



(a) Square



(b) Diagonal

Figure 2.5: Comparison of square and diagonal simulation steps

Figure 2.5 shows the number of steps taken by the movements for a single simulation with start and end points of (42, 28) and (29, 74).

As suggested in the previous sections, the diagonal movement takes less steps than the square movement to reach the end point. However, the diagonal movement takes longer (time) as the computational complexity is higher.

By taking the same process and applying it to a random selection of start and end points, the average number of steps and time taken by the movements can be compared for different distances travelled (Figure 2.6).

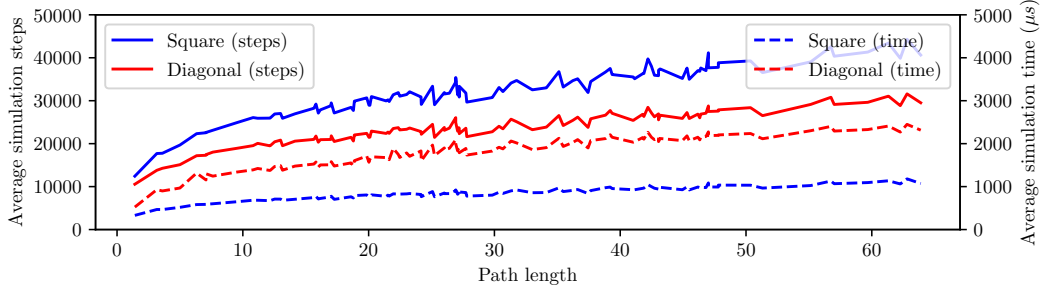


Figure 2.6: Comparison of multiple square and diagonal simulation steps

Next, by plotting the ratio of the average number of steps and time taken by the square and diagonal movements we can establish a factor by which the diagonal movement is more efficient than the square movement (Figure 2.7).

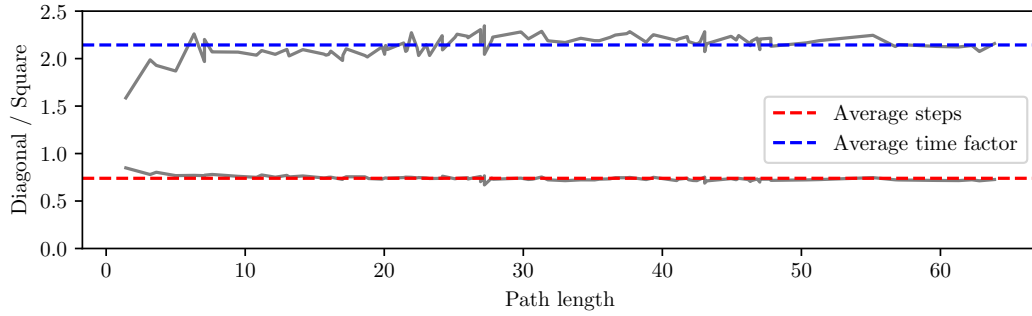


Figure 2.7: Comparison of multiple square and diagonal simulation steps (factor length)

The average factor for simulation steps is 0.7393 which is within the order of magnitude of the expected factor of 0.8284. The discrepancy is likely due to the non-linear nature of the interactions at the edge of the grid.

The average factor for simulation time is 2.1436. Given the square movement is 0.3794 times faster than the diagonal movement, and the diagonal movement takes 0.8284 times as many steps as the square movement, the expected factor for simulation time is $0.3794 \times 0.8284 = 2.184$. This is exceptionally close to the measured factor of 2.1436.

In conclusion, the diagonal movement is more efficient than the square movement in terms of the number of steps taken, but less efficient in terms of computational complexity. The computational complexity is significantly higher for the diagonal movement than the square movement, this overwhelms the number of steps taken. Hence, the square movement is more efficient than the diagonal movement overall.

2.2 Cell Growth

Using the Euler method, we can simulate the differential equation for cell growth shown in Figure 2.8. This uses the rate of growth at a point in time to calculate the number of cells at the next time step.

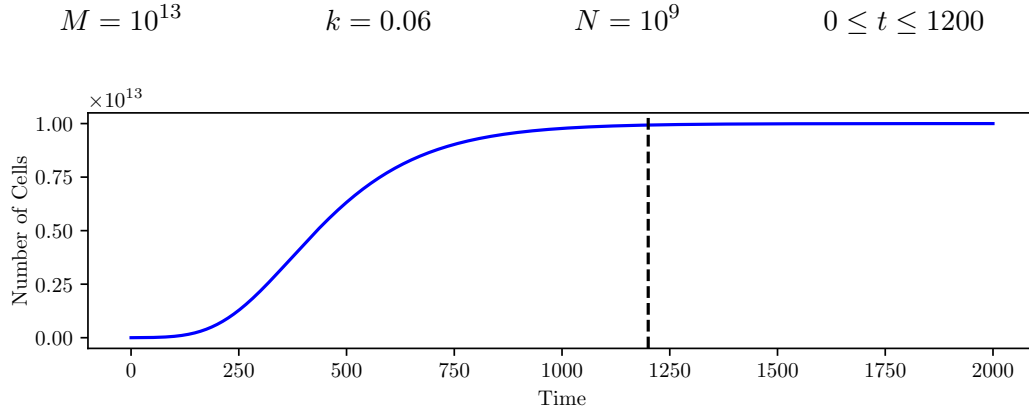


Figure 2.8: Cell growth simulation

The growth appears to reach a steady state at around 1200 time units, at this time the percentage of cells filled is 99.31%. Given the growth is modelled using an exponential function, the growth fills to 100% at $t = \infty$, this applies if the simulation has an infinite accuracy. However on a computer system this value would be rounded up, resulting in a finite time to reach full capacity. Steady state can be defined numerically as:

- Percentage of cells filled is near 100%.
- Rate of change of cell growth is near 0.
- Rate of change of the rate of change of cell growth is near 0.

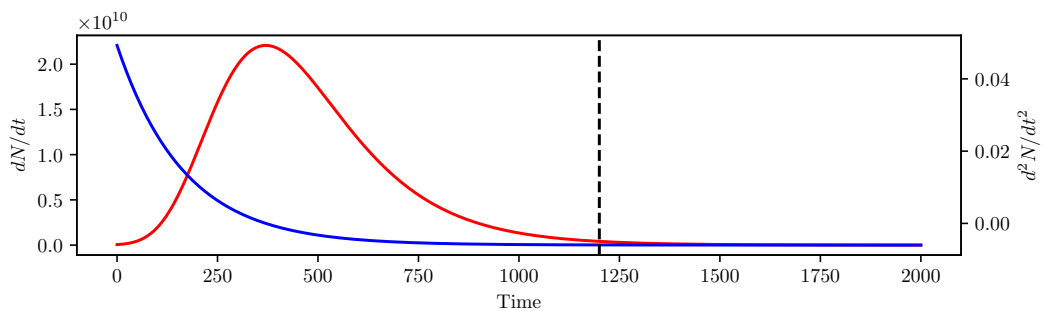
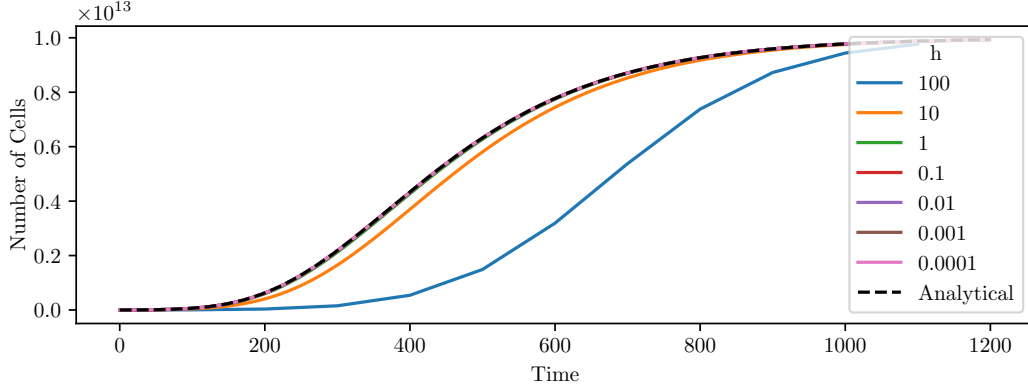


Figure 2.9: Cell growth simulation (dN/dt)

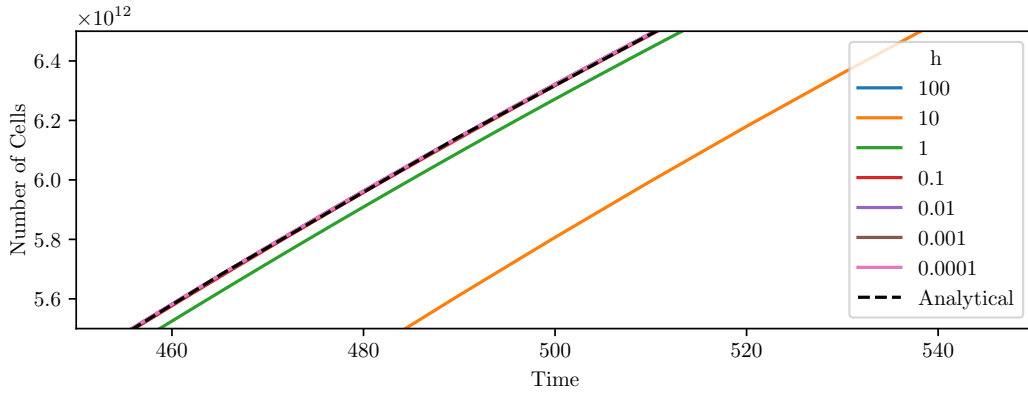
Figure 2.9 shows the rate of growth ($\frac{dN}{dt}$) and the rate of change of the rate of growth ($\frac{d^2N}{dt^2}$) for the simulation, solved analytically. When we reach a value of $t = 1200$, the rate of growth is $4.083e+08$, and the rate of change of the rate of growth is -0.005959 .

2.2.1 Accuracy of the Simulation

When using the Euler method, the time step size (h or Δt) is important for the accuracy of the simulation. Larger values of h will result in larger computation errors relative to the analytical solution. Figure 2.10 shows the growth simulation with different values of h .



(a) Full simulation



(b) Zoomed simulation

Figure 2.10: Cell growth simulation with different dt values

The mean absolute percentage error for different values of h relative to the analytical solution allows us to determine the optimal value of h for the simulation. Smaller values of h result in a more accurate simulation, scaling linearly with the error as shown in Figure 2.11.

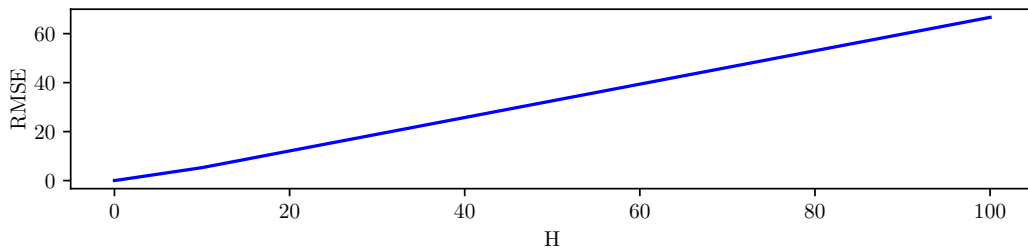


Figure 2.11: Cell growth simulation with different dt values (error)

2.2.2 Model Complexity

The growth model is of $\mathcal{O}(n)$ complexity, where n is the number of steps. However, the value for n is inversely proportional to the value of h ($n = 1/h \times t$), smaller values of h result in a longer computation time.

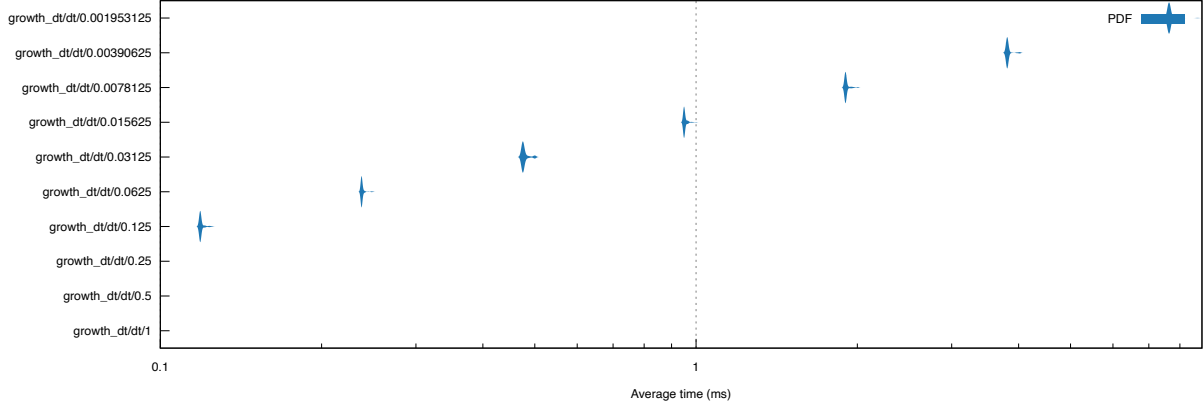


Figure 2.12: Cell growth criterion

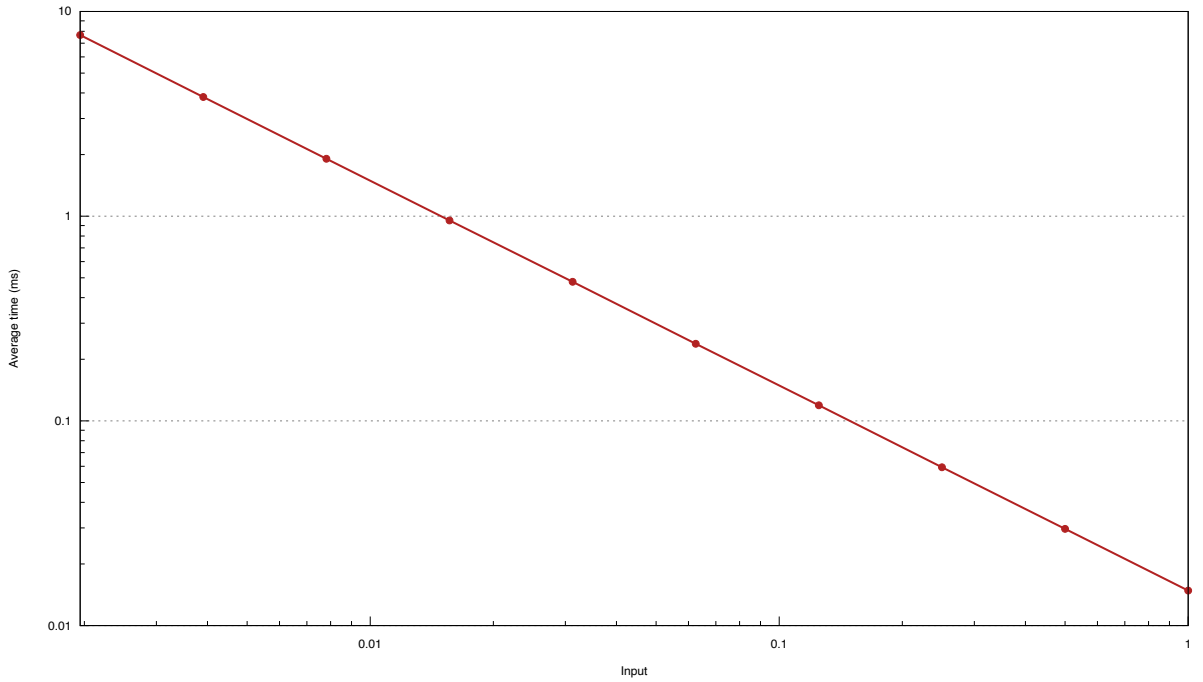


Figure 2.13: Cell growth criterion (line)

By plotting both the value for h and t with logarithmic scales, the reciprocal relationship can be hidden. Criterion benchmarks in Figure 2.12 and Figure 2.13 show that growth is of $\mathcal{O}(n)$ complexity (linear).

2.2.3 Changing the Capacity

Changing the capacity value M for the differential equation will change the rate of growth proportionally, as shown in Figure 2.14.

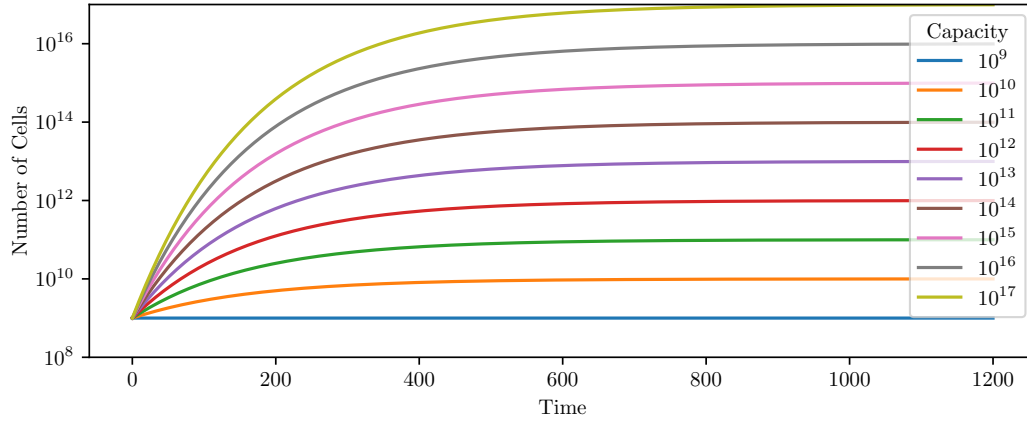


Figure 2.14: Cell growth simulation with different capacity values

However, this is not perfectly linear, as the rate of growth is logarithmic with respect to the capacity value M . This is shown in Figure 2.15 with percentage values.

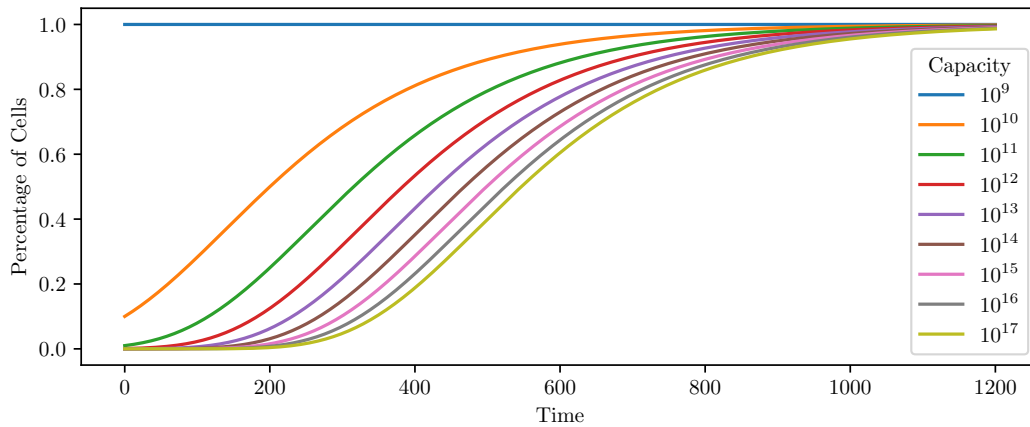


Figure 2.15: Cell growth simulation with different capacity values (percentage)

While this difference is very likely the result of the differential equation itself, it is important to check if accuracy is maintained for different values of M . Figure 2.16 displays that the error increases linearly with the value of M . This is because the larger values of M use more digits and `float64` only has a precision of 15-16 decimal digits, resulting in a larger error.

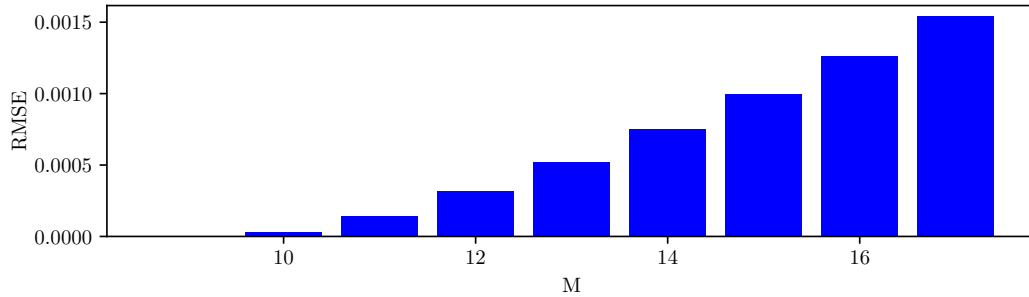


Figure 2.16: Cell growth simulation with different capacity values (error)

We can prove the percentage difference is due to the value of M rather than computational error, by plotting the percentage filled in the situation against the analytical solution (Figure 2.17).

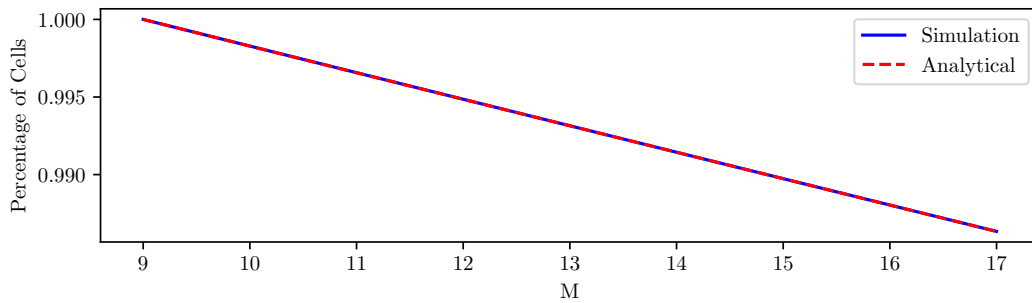


Figure 2.17: Cell growth simulation with different capacity values (percentage error)

2.2.4 Random Walk

Now we can take the random walk and apply it to the cell growth model to simulate the movement of cells within a tissue. Before movement, the cells grow to a maximum capacity or a steady state. Once full, the cell moves in a random direction. Given the random walk is independent of the growth model, we can expect the same results as the independent random walk, displayed in Figure 2.18.

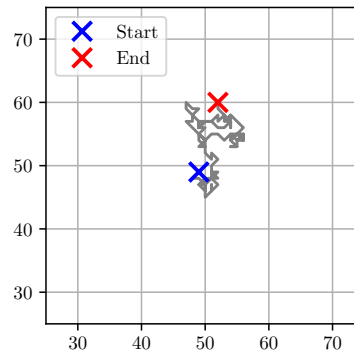


Figure 2.18: Cell growth simulation with diagonal movement

When the random walk reaches a cell it has already visited, it will instantly move to a new cell as the percentage full will be already above the threshold. Hence if the total number cells is plotted, we can expect a linear growth as shown in Figure 2.19 with 100 walk steps. This linear growth can be estimated with a linear function, the growth is always at a steady state, until the tissue/grid is full.

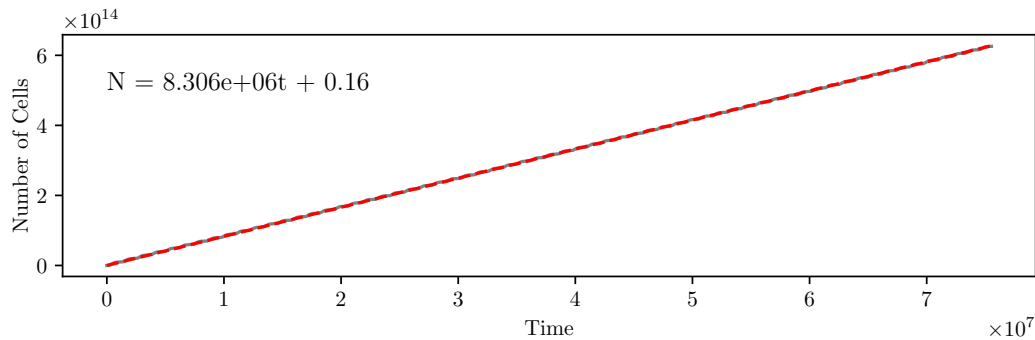


Figure 2.19: Cell growth simulation with diagonal movement (total cells) linear estimation

2.2.5 Changing the Grid Size

All simulations to this point used a grid size of 100x100, with a maximum capacity of 10,000 cells areas. Figure 2.20 shows the simulations steps required to fill the grid for different grid sizes, the graph is plotted against grid area.

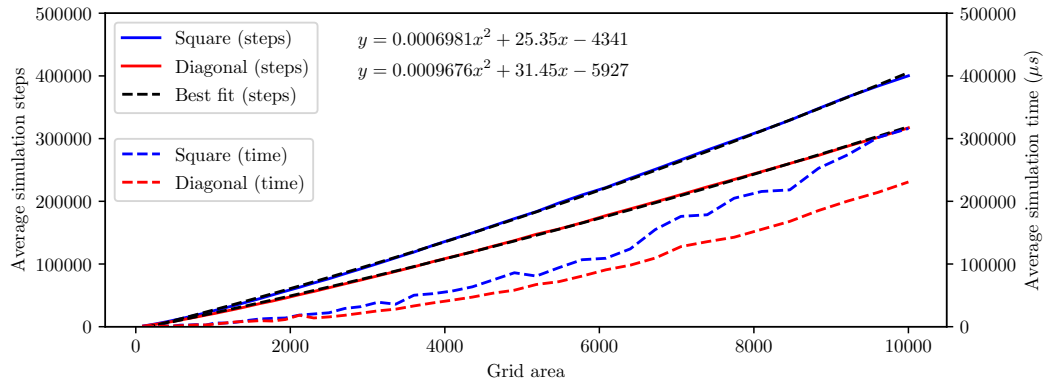


Figure 2.20: Cell growth grid comparison

The relationship between the grid area and the number of steps required to fill the grid is polynomial. The growth model and random walk remain complexity of $\mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$, however the grid size is now a factor in the complexity. This added complexity is related to the concept of filling the grid and is of $\mathcal{O}(n^3)$ if n is the grid width.

3 | Conclusion

This report successfully implemented a simulation of the growth of cancer cells in a tissue using computational techniques. The model's complexity is of $\mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$, where n is the number of steps in the simulation.

Currently the model has no interaction between movement and growth. A Bernoulli distribution for movement based on surface tension and leaky boundary conditions could be implemented to more accurately simulate the growth of cancer cells.

List of Figures

2.1	Cell movement plots	2
2.2	Visited cells simulation heatmap	3
2.3	Cell movement fill plots	4
2.4	Cell movement criterion	5
2.5	Comparison of square and diagonal simulation steps	6
2.6	Comparison of multiple square and diagonal simulation steps	7
2.7	Comparison of multiple square and diagonal simulation steps (factor length) . . .	7
2.8	Cell growth simulation	8
2.9	Cell growth simulation (dN/dt)	8
2.10	Cell growth simulation with different dt values	9
2.11	Cell growth simulation with different dt values (error)	9
2.12	Cell growth criterion	10
2.13	Cell growth criterion (line)	10
2.14	Cell growth simulation with different capacity values	11
2.15	Cell growth simulation with different capacity values (percentage)	11
2.16	Cell growth simulation with different capacity values (error)	12
2.17	Cell growth simulation with different capacity values (percentage error)	12
2.18	Cell growth simulation with diagonal movement	13
2.19	Cell growth simulation with diagonal movement (total cells) linear estimation . .	13
2.20	Cell growth grid comparison	14

Bibliography

- Adam, John (11/23/2023). *Rust – a Concise Overview of the Modern Coding Language*. <<https://kruschecompany.com/rust-language-concise-overview/>> (visited on 04/30/2024).
- Codling, Edward A, Michael J Plank, and Simon Benhamou (04/15/2008). “Random Walk Models in Biology”. In: *Journal of The Royal Society Interface* 5.25, pp. 813–834. DOI: 10.1098/rsif.2008.0014. <<https://royalsocietypublishing.org/doi/10.1098/rsif.2008.0014>> (visited on 04/30/2024).
- Heisler, Brook (04/30/2024). *Bheisler/Criterion.Rs*. <<https://github.com/bheisler/criterion.rs>> (visited on 04/30/2024).
- Tatro, Dyjuan (01/01/2018). “The Mathematics of Cancer: Fitting the Gompertz Equation to Tumor Growth”. In: *Senior Projects Spring 2018*. <https://digitalcommons.bard.edu/senproj_s2018/147>.

A | Appendix

A.1 Integration of Growth Model

Integrate using substitution:

$$\begin{aligned}\frac{dN}{dt} &= kN \ln\left(\frac{M}{N}\right) \\ \frac{dN}{N \ln\left(\frac{M}{N}\right)} &= k dt \\ \int \frac{dN}{N \ln\left(\frac{M}{N}\right)} &= \int k dt\end{aligned}$$

$$\text{with } u = \ln\left(\frac{M}{N}\right) \quad \text{and} \quad \frac{du}{dN} = -\frac{1}{N}$$

$$\begin{aligned}\int \frac{-N du}{Nu} &= \int k dt \\ \int -\frac{1}{u} du &= \int k dt \\ -\ln(|u|) &= kt + c \\ \ln(|u|) &= -kt - c \\ \ln\left(\left|\ln\left(\frac{M}{N}\right)\right|\right) &= -kt - c \\ \ln\left(\frac{M}{N}\right) &= e^{-kt-c} \\ \frac{M}{N} &= e^{e^{-kt-c}} \\ N &= \frac{M}{e^{e^{-kt-c}}}\end{aligned}$$

Calculate c using the initial values provided:

$$M = 10^{13}$$

$$k = 0.06$$

$$N = 10^9$$

$$t = 0$$

$$\begin{aligned} 10^9 &= \frac{10^{13}}{e^{e^{-0.006(0)} - c}} \\ e^{e^{-c}} &= \frac{10^{13}}{10^9} \\ e^{e^{-c}} &= 10^4 \\ -c &= \ln(|\ln(10^4)|) \\ c &= -\ln(|\ln(10^4)|) \end{aligned}$$

Substitute c back into the equation and simplify:

$$\begin{aligned} N &= \frac{M}{e^{e^{-kt} + \ln(|\ln(10^4)|)}} \\ &= \frac{M}{e^{e^{-kt}} e^{\ln(|\ln(10^4)|)}} \\ &= \frac{M}{e^{e^{-kt} \ln(10^4)}} \\ &= \frac{M}{(e^{\ln(10^4)})^{e^{-kt}}} \\ N &= \frac{M}{10^{4e^{-kt}}} \end{aligned}$$