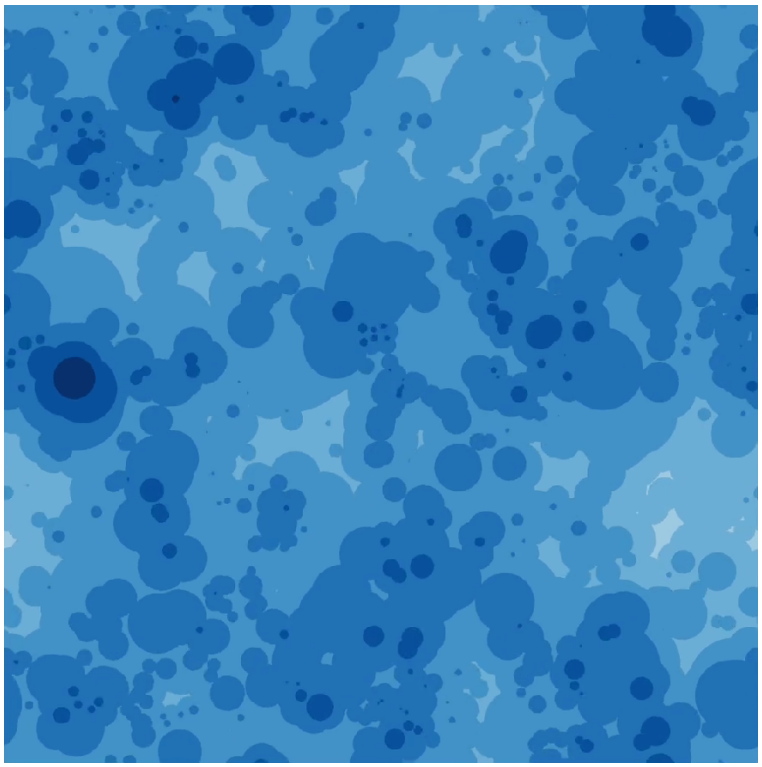
UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE



Master thesis

Johan Fridrik Kjølbro

Cold pools & convective organization

A study of colliding and merging cold pools and the role this plays in convective organization

Advisor: Jan Olaf Mirko Härter

Handed in: August 2, 2021

Contents

Abstract

In this project an alternative code for the circle model is written, a model created by Jan O. Haerter, Steven J. Böing, Olga Henneberg and Silas Boye Nissen to simulate cold pool self-interactions. The code is further expanded upon, increasing the complexity of the model by having it encompass different thermodynamical effects. With these different variations to the original circle model, we explore the impact that these effects have on the lifetime of 'isolated' systems of cold pools as well as the affect that they have on their spatial distribution - like whether or not they promote self-aggregation. We find that the effect that most strongly affects the lifetime is the one associated with having a diurnal cycle. Furthermore the system of cold pools, for all variants (except the diurnal cycle variant) progress into a system that is more clustered than the random control, but it remains unclear whether this is a sign of self-aggregation or not.

1 Introduction

In atmospheric science the physics of clouds, gases and aerosols are studied in an attempt to understand the weather, an ever-changing climate, the oceans and their effect on the atmosphere. A phenomenon encompassing all three objects (clouds, gases and aerosols) are cold pools and they play an important role in understanding convective organization and cloud formation.

Cold pools are as the name suggests 'pools' of cold air and they may form beneath precipitating thunderstorms, when the rainfall partially evaporates and cools the surrounding air. The cooler air being denser, will fall towards the surface and spread across it, taking the shape of an almost circular pool. The expanding cold pool causes wind shears and temperature variations, but it also transports moisture. That moisture is being transported by cold pools, stems from the fact that the evaporated droplets that created the cold pool to begin with, also increased the humidity and this water vapour is pushed outwards and transported along the rim of the expanding cold pool. Cold pools are recognized as playing an important role in convective organization and have been vigorously studied for a long time. They have played a central role in many studies of atmospheric science, even when the subject matter weren't directly focused on these. Take for example the research on squall lines conducted by Rotunno, Klemp and Weisman, back in the '80s - Squall lines are defined as lines of active thunderstorms [1] and they usually form ahead of cold fronts. The research, which was conducted using numerical models, concluded that cold pools and wind shears could positively affect the formation of these aforementioned squall lines, and the theory that was derived from this is now referred to as 'RKW' theory [2]. RKW theory has since been a hot topic of research in atmospheric science and has led to the output of many articles such as [3], [4] or [5]. Cold pools have however also been the main subject of many studies and a lot of research, for example in studies regarding the organization of the sub-cloud layer, their involvement in triggering new convective cells or their role in cloud organization and the effect that cold pools have on the large-scale environment [6], [7] or [8]. Another hot topic has been cold pools and their affect on the tropical boundary layer (TPL) specifically, both over land and ocean. Articles such as [9] have been written on this subject, and in the article they discuss how cold pools dominate the temperature variability observed over the central Indian ocean. Despite the many studies of cold pools and the many articles written on the subject, cold pools and their involvement in for example the triggering of new convective cells, remain poorly understood.
Along with the many different topics of interest related to cold pools, there have been a near equal amount of different approaches to study them. The approaches typically fall into one of two categories, which is either observational by collecting data, like temperature readings, satellite images,

measurements of wind velocities, moisture levels and so forth, or by designing numerical models and simulating a convective system. In this thesis we will be concerned with the latter approach and specifically a model called the 'circle model' designed by Jan O. Haerter, Steven J. Böing, Olga Henneberg and Silas Boye Nissen. The circle model is a model in which cold pools are represented by expanding circles (hence the name) and new convective cells are triggered by the collisions of these. The purpose of this idealized model is to study cold pools and gain an understanding of their self-dynamics, their role in convective organization and how they may lead to self-aggregation. In this project I will write my own version of the circle model in Matlab, create different variants of this model to address complexities that were at first omitted in the original circle model and then analyse and compare these. But before I do any of this, I will begin with a more comprehensive introduction of cold pools and the circle model.

1.1 Cold pools

Cold pools are, as previously defined, a term used to describe a pool of air that is cold relative to the air that surrounds it and is enclosed by isotherms [1]. Isotherms are lines connecting points of equal temperature. A cold pool typically forms beneath precipitating clouds, as a result of the evaporation of rainfall in sub-saturated air. The evaporatively cooled air will create a downdraft, due to the negative buoyancy of the colder air. When the air hits the surface layer it will expand radially across it, taking a shape reminiscent of a disc. As the cold pool expands, surrounding air will be pushed up by the cold pool, which may cause further convection. The interior of the cold pool is itself dry, but the rim of the cold pools have a higher level of humidity [10]. This more humid air at the rim of a cold pool, is the result of the evaporated rainfall being pushed out by the expanding cold pool. The maximal radii for cold pools vary, but lies typically in the range of 10 km to 100 km [6] [7] [8] & [11], and they expand at horizontal velocities exceeding 5 m/s and at the rim of the cold pool vertical velocities of more than 0.5 m/s can be measured [10].
These cold pools may trigger new precipitation events, by lifting air parcels of a high humidity through the negative inhibition layer to its level of free convection. As mentioned there's an uplift or vertical velocity associated with the moist rim of the cold pool. This vertical velocity is a product of mechanical forcing and thermodynamical effects. The expanding cold pool can interact with low-level wind shears to create an uplift mechanically, but the accumulation of moist air at the rim is itself less dense/more buoyant than the dryer air around it and heating would further increase its buoyancy. In Tompkins "Organization of Tropical Convection in Low Vertical Wind Shears: The Role of Water Vapor" (2001) [12], a cloud resolving model is used to conclude that even in the absence of large vertical wind shears the cold pools can trigger precipitation events. That is to say, that the mechanical forcing created by an expanding cold pool may not be the primary factor in triggering new cloud formation and subsequent precipitation events. But aside from the naturally occurring lifting of moist air associated with cold pools, a more extreme lifting may occur when multiple cold pools collide. The moist air at the rim of the colliding cold pools, will through momentum conservation be pushed upwards. As the atmospheric pressure drops, so does the temperature and water vapour will start to condense, creating droplets that sticks to tiny particles of salt or dust in the air and eventually a cloud is created. An illustration of this cold pool life-cycle can be see in fig. 1.

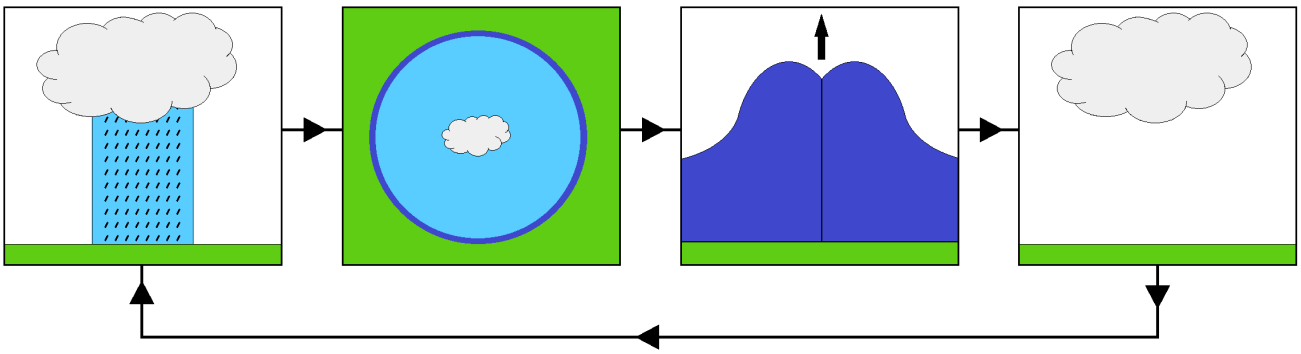


Figure 1: An illustration of the cold pool cycle. Starting with a precipitation event that cools the air, followed by the cool air having dropped towards the ground and expanding radially. The second image is seen from above, with the darker blue air being the humid air at the rim of the cold pool. The third image pictures a cross-sectional view of two cold pool fronts colliding, forcing the air upwards. The water vapour that is being pushed up, condenses and creates a new cloud, that at some later point precipitates restarting the cycle.

This line of thought is in agreement with a subsequent paper by Tompkins [10] and in papers like the one by Jan O. Haerter, et al. [8], where LES (Large Eddy Simulations) are employed to study convective organization. Whether new clouds form as a function of mechanical lifting upon collisions or through thermodynamical effects, like humid air being more buoyant, the event is most likely to happen when two or more cold pools collide - if it is a result of mechanical lifting, the collision is crucial, if it is a function of humidity, two or more fronts meeting would also increase the humidity at this location, increasing the likelyhood of a cloud forming there. To summarize cold pools play a role in organizing the subcloud layer (convective organization, humidity and wind fields), it creates deep convection, it plays a role in the formation of clouds and new precipitation events and it may even play a role in the self-aggregation of cloud formation [13].

1.2 An introduction to the circle model

In order to study these cold pools and their dynamics, an analytical model was created by Jan O. Haerter, Steven J. Böing, Olga Henneberg and Silas Boye Nissen [8] [13] - Supplementary information to the mathematical model used in [8], can be found along with the article at agupubs.onlinelibrary.wiley.com. In short the model simulates cold pool dynamics and their interactions and it does so by initially seeding a multitude of points randomly located in a two dimensional space. These points expands as circles with equal and constant velocity and they represent cold pools expanding across the surface. The boundaries of the space is taken to be periodic, to approximate a very large or even infinite space. When the rim of two or three circles collide at a point in space, a new circle is seeded at that location, with an initial radius of zero. The new circle starts expanding immediately, with the same constant velocity as every other circle in the simulation. This new circle is seeded because of the evaporation of rain from the precipitation event stimulated by the colliding cold pools, as mentioned in the previous section. When two cold pools collide, in the model, their wave front becomes static and inactive - meaning this part of the expanding circle will no longer contribute in another collision event.
In this model cold pools can readily be categorized by their generation. All initial circles would naturally belong to the first generation of cold pools, with every cold pool in generation 2 being seeded as a result of colliding cold pools from generation 1. Hence every circle belonging to generation n with $n \in \mathbb{N}$ would then stem from collisions between circles of generation $n-1$. The reason as to why no circles are created by collisions from circles across different generations and the reason why

the concept of generations in this model is a meaningful concept, is because it is simply not possible within this model to have cross generational collisions. It is not possible because any point in space will be occupied by a circles ”parents” before the circle itself and colliding fronts become static and inactive after collision. To show this, I will begin by considering two colliding cold pools and the newly seeded cold pool that is situated between these, see fig. 2 for an illustration of the setup. I will then attempt to show that the distance from the center of a parent cold pool to an arbitrary point (the red dot) is shorter than the distance from the new cold pool (blue dot) to the same arbitrary point (red dot).

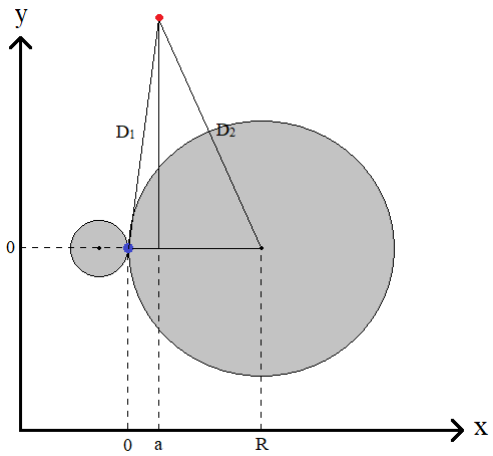


Figure 2: An illustration of the validity of the concept of generations. The two grey cold pools belong to generation N and the blue dot is the cold pool created upon their collision, belonging to generation $N+1$. The red dot is some arbitrary point in space with $x > 0$, whilst D_1 and D_2 are the distances from the cold pool of generation $N+1$ and the distance from the center of the parent cold pool to its right. Furthermore 0 is the x-value of the blue dot, a is the x-value of the red dot and R is the x-value of the parent cold pools center.

I consider arbitrary points in space with an x value greater than zero (the red point) and thus focus on the newly seeded circle and its parent circle to the right - the same arguments would hold for points in space with $x < 0$ by considering the newly seeded cold pool and its parent to the left. This amounts to showing that

$$D_1 \geq D_2 - R, \qquad \text{with } D_1 = \sqrt{a^2 + y^2} \text{ and } D_2 = \sqrt{(R-a)^2 + y^2}. \tag{1}$$

D_1 is the distance from the new cold pool to an arbitrary location with a positive x-value and some y-value, D_2 is the distance from the center of the parent cold pool to the same location and $D_2 - R$ is the distance from the rim of that parent cold pool to that same location (with R being the x-value of the center of the parent cold pool as well as its radius at the time the new cold pool is seeded). It is enough to show that $D_1 \geq D_2 - R$ for the special case of $a = 0$, because as a increase towards R the distance D_2 decreases while D_1 increases and for a exceeding R both D_1 and D_2 increases, but D_1 increases more rapidly. Therefore all that is left to show is thus

$$y \geq \sqrt{R^2 + y^2} - R \qquad \text{for } \forall y \in \mathbb{R}^+, \tag{2}$$

which you obtain by inserting the expressions for D_1 and D_2 in eq. (1) and setting $a = 0$. For $y = 0$ and $\lim_{y\to\infty}$ both expressions are equal. They both increase with y for $\forall y \in \mathbb{R}^+$, their first derivatives

with respect to y are always positive and

$$\left.\frac{\mathrm{d}D_1}{\mathrm{d}y}\right|_{y=0} > \left.\frac{\mathrm{d}D_2 - R}{\mathrm{d}y}\right|_{y=0}, \tag{3}$$

so we can conclude that the distance from the newly seeded cold pool to any point in space is further than the distance from the rim of a parent cold pool to that same location (or equal if infinitely far away). Once we start coding the model in Matlab, in the following sections, it will become apparent that this concept of generations immensely simplifies things.

There are currently two main versions of the model, one in which two colliding cold pools seed a new cold pool and one in which it requires three cold pools to collide at one point. Two cold pool collisions simply seed a new circle in between them, at the halfway point between their rims, whereas another requirement is placed on the three cold pool circle model. In the three cold pool circle model it is required, for a new cold pool to be seeded, that the collision happens such that the point of collision is enclosed on all sites. The idea being, that if the point isn't enclosed by the three colliding cold pools, the air could be pushed out instead of having a forced updraft. This requirement is equivalent to only considering collisions that take place at a location within the triangle that the three colliding cold pool centres form, see fig. 3 for an illustration of the area

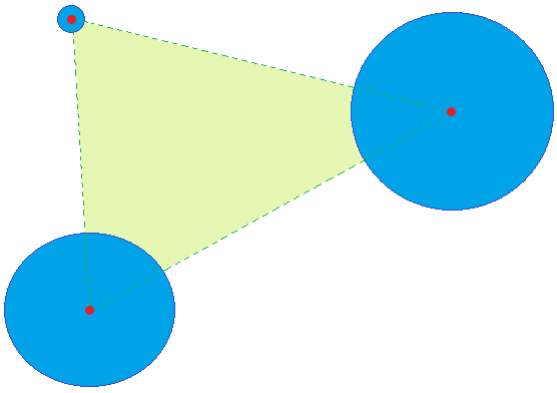


Figure 3: An illustration of the triangle condition. The green area indicates the area of accepted new seeding, the blue circles are the three cold pools and the red dots mark the cold pool centres.

The two and three cold pool circle models are the models that we will continue to work with.

1.3 The project

The models previously mentioned conceptualizes cold pools and their "self-dynamics", and can be used to simulate and extract information about these. Information such as how the number of cold pools change with generations or whether or not these dynamics promote the clustering of clouds (self-aggregation). But as with most models in physics, it is an idealization of what really is. Some of the simplifications made to the model, directly as well as indirectly, includes: cold pools increase infinitely (except for in the two cold pool circle model, which I will discuss in the next few sections), cold pool collisions will always seed new cold pools (if requirements are met), the newly seeded cold pool is situated exactly at the collision point, they instantaneously start growing and they all grow with constant and equal velocities. It would for example be more realistic to consider the process of adding

new cold pools to be stochastic and dependent on the size/age of cold pools involved. Likewise the conditions at the location of collision, like local humidity, could also play a role in the probability of collisions triggering precipitation events. While this ideal model of the self-interactions between cold pools in isolation may yield results that helps us understand the impact of cold pools, it may be missing some interesting information due to the simplifications made. It could be that having non-constant speeds would further promote self-aggregation or maybe restricting the expansion of cold pools would increase the number of cold pools in subsequent generations. In this project I aim to address this, by creating various alternative versions of these cold pool models and explore the differences between these, in an attempt to understand how these different effects affect the results.

2 Methodology

In this section I will discuss the different cold pool models and their underlying code: the two and three cold pool collision models and the various thermodynamical models. I will begin with a more thorough presentation of the two and three cold pool collision models, as they lay the foundation for the other variants of the models. I will start by motivating and discussing some of the underlying assumptions and simplifications of the models, and then proceed with a more detailed account of how the code is written. After that I will discuss the thermodynamical models, proceeding in the same manner as before.

2.1 The circle model

The motivation for and the concept of the circle model and its two main variants, the two and three cold pool collision models, was discussed in the introduction. To quickly summarize, the circle model was created to simulate the dynamics and the interactions of cold pools. These interacting or colliding cold pools, would subsequently result in the formation of new cold pools - by the process discussed previously. The model was meant as a tool, with which one can study these interactions and their implications on convective organization.
The circle model consists of a two dimensional surface, on which cold pools expand. The boundaries of the domain, is taken to be periodic. The expansion of each cold pool happens with equal and constant velocity, resulting in the surface at some point being completely covered by cold pools see fig. 4. The isotropic expansion of the cold pools, means that the expansion is circular, hence the name of the model. The model is initialized by placing cold pools randomly on the 2D surface, with initial radii of zero. Once all the initial cold pools have been placed, they all simultaneously start to expand and new cold pools are seeded at the collision sites. The new cold pools are immediately placed at the collision site and begins expanding with no delay. Once two cold pools have collided, the wave fronts overlapping becomes inactive, see fig. 5. The inactive fronts can no longer participate in new collision events. It is therefore only when either two or three active fronts collide (depending on the variant of the model), that a new cold pool is seeded. The three cold pool variant had the extra condition imposed, that the collision should be enclosed on all sites.
This model, with its conditions of inactive wave fronts and cold pools expanding at constant and equal velocities, naturally lead to the concept of generations. With cold pools belonging to generation N, being 'born' by collisions of cold pools belonging to generation $N-1$. The concept of generations dramatically simplifies calculations, because it allows us to calculate all cold pools of generation $N+1$ directly from generation N, without having to consider the order that the cold pools in generation $N+1$ are being born. Without the concept of generations, we would need to calculate the very first collision

occurring and take this into consideration when calculating the next collision, and then consider both these when considering the next, and so on. This is because every new cold pool, would be added to the one and only ”generation” of cold pools and could potentially interact with any of the already existing cold pools. Therefore being able to calculate all cold pools of an entire generation simultaneously simplifies calculations greatly, but it is not the only assumption made.

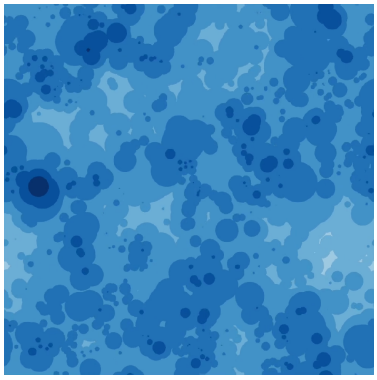


Figure 4: An illustration of the entire square domain being covered in cold pools - produced using Silas Boye Nissens' code for visualization.

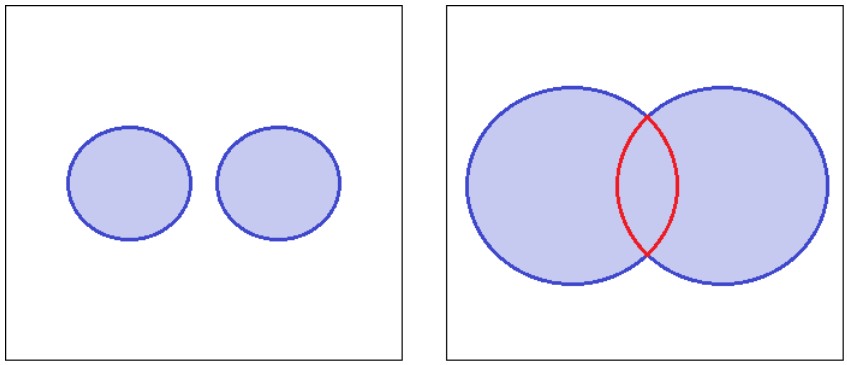


Figure 5: Two cold pools expanding. On top the cold pools have yet to reach one another, below the cold pools have collided and the wave front marked in red is now inactive.

The model as described is obviously an idealization of the nature of cold pools and their interactions, but creating an idealized model is not necessarily a bad thing. Simplifications are often necessary in order to reduce complex physics into something that is at all manageable, and the simplified or idealized cases quite often leads to meaningful results. Physics in general is filled with simplified models, like classical newtonian mechanics, the Heisenberg model (and by extension the Ising model too), mean field theory and the list continues. Nevertheless it is important to realize that the model is a simplification and to keep in mind how it has been simplified. I will here list some of the simplifications made:

1. Dimensionality & topography
2. Periodic boundary conditions
3. Velocities
4. Local environment
5. Circularity
6. Cold pools only from cold pools

1. The space is a completely flat and two dimensional plane. A more realistic picture of the surface, that the cold pool in its expansion traverses, would for example include valleys, hills or trees. Things that could potentially obstruct the expansion of cold pools. While the topography is an idealization, there are of course many locations where this would be a good approximation, especially in flat Denmark.
2. The boundaries are taken to be periodic. This is very commonly done to mimic very large or even infinite systems, without having to compute them in full.
3. The velocities of all cold pools are equal and constant. This would imply that all precipitation events resulting in the formation of cold pools were all of equal scale.

4. There is no reference to the local environment. This for example includes moisture levels, temperatures and winds. These could all potentially impact the interactions of cold pools.
5. The cold pools form perfect circles. This is largely due to some of the previously mentioned simplifications. These are not exactly independent simplifications. A point expanding on a 2D plane with equal and constant velocity in all directions will naturally expand with its rim forming a circle. While the simplifications of equal and constant velocities, can likewise in part be linked to no obstructions in topography and no reference to local environment.
6. Cold pools occur only from the process explained in the introduction. This assumption implies that no cold pools are 'born' from a precipitation event that is not due to the process of moist air being pushed through its negative layer of inhibition by two or three colliding cold pools. This simplification allows us to consider the interactions of cold pools in isolation.

These are some of the underlying assumptions and simplifications of the model. It is good to have them in mind, when considering the limitations of the model. With that said, I will proceed with a more detailed description of how the model runs.

2.1.1 Two cold pool collision model

In this section and the next, I will in detail go through the setup of the code of both the two and three cold pool collision models. I will attempt to explain the thought process behind the different decisions made and explain how these are implemented. The code is written in Matlab and so the language will be thereafter. This code, as previously stated, is based on an existing model made by Jan O. Haerter, Steven J. Böing, Olga Henneberg and Silas Boye Nissen. I will code it independently and therefore it might differ from their code in numerous ways, but the concept of the model remains the same, that is cold pools expand as perfect circles on a square domain with periodic boundary conditions and colliding cold pools create a static wave front that can not partake in new collision events. I will start by detailing the simpler model, namely the two cold pool collision model.

The very first thing that we need to do, is define the parameters of the system, simulation and model. The system parameters are the domain size and number of initial cold pools, or jointly expressed the density of cold pools. The simulation parameters are the number of generations that we are going to compute and the number of nearest neighbours, I will return to the concept of nearest neighbours shortly. The last set of parameters are model dependent parameters and in the two cold pool collision model this includes a maximum and minimum radii, in between which cold pools can collide and create new cold pools. See fig. 6 for the list of parameters.

```
%Initial #CP, domain size, nearest neighbours, generations and max/min radii:
inum = 200; sqlength = 10; center = sqlength/2; near = 15; gen = 24; minR = 0.5; maxR = 2;
```

Figure 6: A screenshot of some defined parameters in the code

The idea was to keep it as simple as possible to begin with, so why the minimum and maximum radii? The reason is that it is necessary for the two cold pool collision model to have these defined. A minimum radii that cold pools need to be before they can successfully create a new cold pool upon collision, is necessary to avoid ever lasting triangular configurations like that shown in fig. 7. This triangular configuration will in the next generation result in a smaller but identical configuration, and in each subsequent generation a smaller configuration of three cold pools will occur. These configurations will happen all over the domain and the number of cold pools will eventually blow up, due to the

existence of these. To avoid it, we add a minimum radii, such that these eventually die out. Incidentally it also makes sense physically, that two cold pool centres that are infinitely close, really is one and the same cold pool or belonging to the same precipitation event. If they're belonging to the same precipitation event or equivalently said one and the same cold pool, we should according to the model avoid these, as we do not have self interaction.

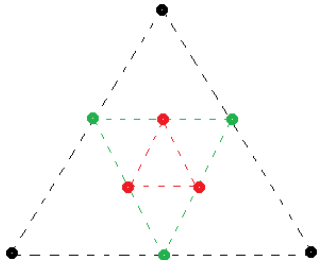


Figure 7: Black dots mark the CP centres of gen. N, green dots gen. $N+1$ and red gen. $N+2$. The stipled lines show the shortest path between cold pools. This configuration continues infinitely.

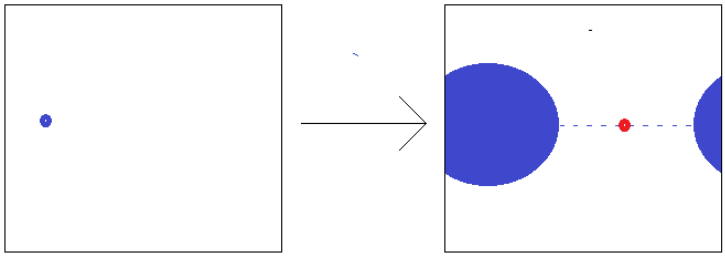


Figure 8: Left square indicates the domain at time t and right square is at some time $t + \Delta t$. The blue circle marks a cold pool and the red dot marks the spot where it will collide with itself.

The maximum radii is similarly defined to avoid self interaction. Imagine only having a single cold pool left in the entire domain. When this lone cold pool expands beyond the boundaries, it may collide with itself due to the periodic boundary conditions and therefore self interact, see fig. 8. To avoid this a maximum radii is selected, one that is smaller than half the length of the domains sides. This too makes sense physically, that cold pools can not expand infinitely. Strictly speaking, a maximum radii is not necessary to avoid self interaction with the way that I treat periodic boundary conditions (which I will relay shortly), but since the original circle model had this included and because it makes sense physically, I also wrote the code with a maximum radii.
These were all the parameters that we needed to define, but before we can begin simulating we still need to occupy the space with cold pools. This is done by first creating a matrix of 5 columns and a number of rows equal to the number of initial cold pools, see fig. 9. Each row in the matrix is the details of a cold pool and the columns in order define the generation it belongs to, its center x-value, its center y-value, the time it was seeded and its number - the unique number assigned to each cold pool will later be used to avoid double or triple counting interactions.

```
%Randomly generating initial setup:
seed = 1; rng(seed); circ = zeros(inum,5); %columns in order: generation, x, y, time seeded and unique number
circ(:,1) = ones(inum,1); circ(:,2) = rand(inum,1)*sqlength; circ(:,3) = rand(inum,1)*sqlength; circ(:,5) = [1:inum]';
```

Figure 9: A screenshot of the cold pool matrix setup in the code

I also define a 'seed' value, to be able to 'randomly' generate the same values, making us able to start with the same randomly generated initial setup multiple times. With the setup defined, we can start the simulation.
Now as previously discussed the model contains a useful concept, namely generations, and with this concept we can focus on each generation of cold pools individually. We will therefore start with a loop, going through the number of generations that we wish to compute and during each execution we will calculate the cold pools of the subsequent generation. Having specified the generation, we will need to go through all pairs of cold pools to consider how these may collide to form new cold pools. The number of such pairs, avoiding double counting, is given by $n(n-1)/2$, with n being the number of cold pools belonging to the generation that we are currently considering. If n is a large number, the

number of unique pairs quickly becomes very large, and it is computationally heavy to consider all these in full. But we don't actually have to consider all these, because not all cold pools are likely to collide, see fig. 10 for an illustration of this.

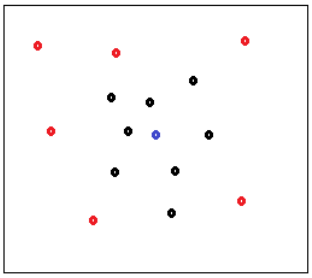


Figure 10: The square indicates the domain, the blue dot marks a CP in the centre of the domain, the black dots mark nearby neighbours of the blue CP and the red dots mark distant neighbours that the blue CP is unlikely to interact with.

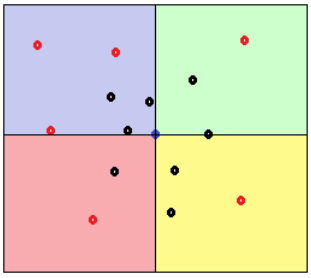


Figure 11: The domain has been split into four quadrants coloured in light green, blue, red and yellow.

The center cold pool colored blue, will most likely not collide with the red colored cold pools further away, because it is surrounded by more closely located cold pools colored black - this is because the wave front becomes inactive after collision. It would be more effective, if we ran through all n cold pools and for each of these only considered pairing them with their nearest neighbours, instead of the every $(n-1)$ other cold pools. There's a lot of different ways one can choose to define a cold pools nearest neighbours, but the way I have chosen to do so is illustrated in fig. 11. For now assume that the cold pool in question is in the center of the domain (I will make sense of this assumption soon), then I define four regions, specifically four quadrants of the square and within each of these I select a number of cold pools given by the parameter 'near' and these will be its nearest neighbours in that region. Now the reason the cold pool in question is located exactly in the center of the domain, is that this is how I actually deal with periodic boundaries! Periodic boundaries could be invoked by surrounding the domain with copies of itself, in a manner illustrated in fig. 12.

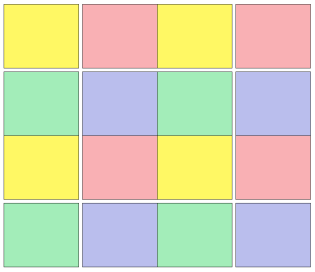


Figure 12: The four centre squares indicate the domain split into quadrants, around it are copies of the quadrants.

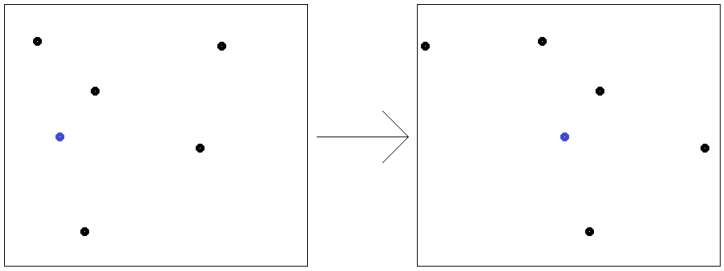


Figure 13: An illustration on how I achieve periodic boundary conditions, by shifting the CPs within the domain using 'mod'.

The four center squares in blue, green, red and yellow combined constitutes the entire domain. Around the domain we could place copies of these quadrants to mimic periodic boundary conditions. This has the advantage of very accurately depicting periodic boundary conditions, but it is computationally heavy to include all these copies. Another way to achieve periodic boundaries without having to surround our domain with copies of itself, is to shift the cold pools in question to the center of the

domain by shifting the entire image periodically and then assume that the boundaries are too far away for an active wave front to reach. This is similar to the assumption we made, when creating the concept of having nearest neighbours. An illustration of this is seen in fig. 13, where the entire domain is shifted periodically such that the blue cold pool is in the center of the square domain. The periodic shifting of cold pools within the domain, is achieved by using the operation modulo on the x-value and y-value columns of the cold pool matrix. For example if we have a cold pool center located at $x = 2$ and $y = 3$ and we want to shift everything, such that it is located at $(5, 5)$, we take the entire column of x-values which we could call 'xval' and redefine it as $\text{xval} = \text{mod}(\text{xval} + 3, \text{sqlength})$ and similarly for all y-values. The assumption that the boundaries are then too far away, is actually no assumption at all. This is because we have defined a max radii smaller than half the side length (to avoid self interaction), and with this the boundaries are too far away for the center cold pool to cross. It may still seem possible for a nearest neighbour cold pool to cross the boundary and interact with the center cold pool from the other side, as illustrated in fig. 14. This however is never possible, because the stippled line B is always longer than A and it is only the first collision between two given cold pools that we consider. Hence the assumption is no assumption at all, and can be seen when comparing results between my code and their code, see fig. 15.

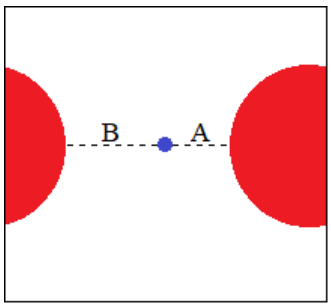


Figure 14: An illustration of a larger red cold pool interacting with a smaller blue cold pool. Only interacting in the middle of the shorter path A.

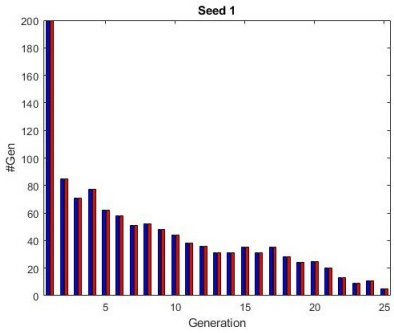


Figure 15: Comparing my two CP circle model to theirs, starting with the same initial setup. Number of cold pools vs. generation. Initial number: 200, generations: 25

The process of selecting a cold pool, shifting the entire domain periodically and calculating its nearest neighbours is something that is looped over all possible initial choices of cold pools within the specified generation, but once this has been done and a specific cold pool has been picked and shifted to the center, then we need to calculate the point where it may collide with its nearest neighbours. This is done by considering the half way point of the shortest distance between the two cold pools, because this is the point where they will collide first, given that they're expanding with equal velocities. But during these calculations, it is more natural to consider the cold pools radii rather than at which time they are seeded, so I will convert the fourth column of the cold pool matrix. Since cold pools are expanding at a constant and equal velocity, converting from time seeded to relative radii is basically changing the sign of the value. Consider having two cold pools, call them A and B, with cold pool A having been seeded at time 0 and B at time 2. Cold pool A is thus older than B and should therefore also be larger at any given time. Changing the sign results in A having the value 0 once again, but B the value -2. These values relate the radii of the cold pools, with A having a radius of 0 when B have a radius of negative two, meaning it hasn't been seeded yet. We could translate the time forward by a constant, say two, such that A has the value 2 and B the value 0, these are the radii at the time B is seeded. To get the radii r_i (with $i \in \{1, 2, 3...\}$) of all cold pools, at the exact moment t_C that some cold pool C is seeded, we would calculate these by $r_i = t_c - t_i$. The fact that we can simply

convert and translate like this, is because of the constant and equal velocities. For simplicities sake, I choose to convert from time to radii, at the moment the center cold pool was seeded. This makes calculations a little simpler. Now consider cold pool A to be the cold pool in the centre and B to be one of its nearest neighbours, then I start by calculating the gradient of the line connecting the centres as grad $= (y_B - y_A)/(x_B - x_A)$ (with the center of A being (x_A, y_A) and similarly defined for B). Then I calculate the points of intersection of this line with the rim of the cold pools, for A this is simply at (x_a, y_a) because its radius is zero, but for B this is at $(x_B, y_B) \pm \frac{r_B}{\sqrt{1+grad^2}}(1, \text{grad})$ with the positive sign if $x_A > x_B$ and negative otherwise. With these points (p_A and p_B) I calculate the collision site as the halfway point between these $p = (p_A + p_B)/2$ and I calculate the distance that the cold pools need to expand before colliding as the distance between p_A and p, for reference see fig. 16.

```
%Calculating new circles
grad = ((n(h,3)-n(1,3))/(n(h,2)-n(1,2)));
%p1 and p2 are the intersection points between the connecting line and the circles.
p1 = [n(1,2) n(1,3)];
if n(1,2) < n(h,2)
p2 = [n(h,2) n(h,3)] - (n(h,4)/sqrt(1 + grad^2))*[1 grad];
else
p2 = [n(h,2) n(h,3)] + (n(h,4)/sqrt(1 + grad^2))*[1 grad];
end
%p is the collision point, p(1,1) is xval, p(1,2) is yval and p(1,3) is distance the circles need to grow to collide.
p = (p1 + p2)/2; p(1,3) = sqrt((p1(1,1) - p(1,1))^2 + (p1(1,2)-p(1,2))^2);
```

Figure 16: A screenshot of collision point calculation in the code of the two cold pool model.

Now that we have calculated the location of a potentially new cold pool, we need to check that it satisfies all conditions. I first check both cold pools have expanded to a size beyond R_{min} but below R_{max}. Then I make sure that no other cold pool, of the same generation, is occupying the space of the collision site. This is to ensure that when the two cold pools in question collide, their wavefronts are still considered active. This is achieved by calculating the distance from the cold pool centre of all nearest neighbours to the collision site (excluding cold pool A and B) and making sure that these are all larger than the radii of these cold pools at the time of collision. Lastly I make sure that I have not already considered the collision between cold pool A and B, when B was taken to be in the center. The conditions can be see in fig. 17.

```
if n(1,4) + p(1,3) > minR && n(h,4) + p(1,3) > minR && n(1,4) + p(1,3) < maxR && n(h,4) + p(1,3) < maxR
A = sqrt((p(1,1)-n(:,2)).^2 + (p(1,2)-n(:,3)).^2);
A(1) = []; A(h-1) = [];
B = n(:,4) + p(1,3);
B(1) = []; B(h-1) = [];
if all(A >= B)
if sum(sum(sort([n(1,5) n(h,5)]) == circmem,2) == 2) == 0
```

Figure 17: A screenshot of a list of conditions in the code of the two cold pool model.

If all conditions are met, then all that there is left to do is to add a row to the matrix of all cold pools (called 'circ' in the code) with the details of this new cold pool. This again includes its generation, its position, its time seeded and so on. It is important here to remember that the position is in the periodically shifted domain and we have converted to radii, so we need to account for this when adding the new cold pool. This procedure then repeats through all nearest neighbours, and this repeats through all cold pools of the specified generation and this repeats through all generations that we decided to calculate at the start of the code. The finished result then being a matrix containing the details of all cold pools, their generations, center locations and time seeded.

2.1.2 Three cold pool collision model

The code detailing the three cold pool collision model is very similarly structured and coded, with some exceptions. The defining parameters are the same as before, excluding a maximum and minimum radii as these are strictly speaking not necessary for this code to function. The setup, generating randomly placed cold pools across the domain is done in the same way as before. We run through the options in much the same way, by looping over the generations, selecting a cold pool, shifting the system periodically, computing its nearest neighbours and then running through all options of groups of three cold pools within the subset of nearest neighbours - periodic boundaries are accounted for in the same way as before. Now how we calculate the locations of new cold pools, differs between the two models. Computing the point at which three cold pools will collide, is basically done by solving this set of equations

$$(x - x_1)^2 + (y - y_1)^2 = (r + r_1)^2 \tag{4}$$
$$(x - x_2)^2 + (y - y_2)^2 = (r + r_2)^2 \tag{5}$$
$$(x - x_3)^2 + (y - y_3)^2 = (r + r_3)^2. \tag{6}$$

With x and y being the point of intersection between all three cold pools and r being the value that the cold pools need to expand before all three circles intersect at one point. The constants x_i, y_i and r_i (with $i \in \{1, 2, 3\}$) are the center coordinates of the three cold pools and their initial radii. It is a set of three equations, with three unknowns, and the way I solve these is by first expanding the squares and moving all constant terms to the RHS as so

$$x^2 + x_i^2 - 2xx_i + y^2 + y_i^2 - 2yy_i = r^2 + r_i^2 + 2rr_i \tag{7}$$
$$x^2 - 2xx_i + y^2 - 2yy_i = r^2 + 2rr_i + C_i, \tag{8}$$

with $i \in \{1, 2, 3\}$ and $C_i \equiv r_i^2 - x_i^2 - y_i^2$. We see that terms containing the unknowns squared appear in all three equations, so I will now subtract $i = 3$ from both $i = 1$ and $i = 2$, to get a set of equations free of terms containing squares of x, y or r

$$-2xx_j - 2yy_j + 2xx_3 + 2yy_3 = 2rr_j + C_j - 2rr_3 - C_3 \tag{9}$$
$$2x(x_3 - x_j) + 2y(y_3 - y_j) = 2r(r_j - r_3) + C_j - C_3 \qquad \text{with } j \in \{1, 2\}. \tag{10}$$

With this set of equations, I can solve for x and y and get the following result

$$x = \frac{(2rr_1 + C_1)(y_2 - y_3) + (2rr_2 + C_2)(y_3 - y_1) + (2rr_3 + C_3)(y_1 - y_2)}{2(x_1(y_3 - y_2) + x_2(y_1 - y_3) + x_3(y_2 - y_1))} \tag{11}$$
$$y = \frac{(2rr_1 + C_1)(x_2 - x_3) + (2rr_2 + C_2)(x_3 - x_1) + (2rr_3 + C_3)(x_1 - x_2)}{2(y_1(x_3 - x_2) + y_2(x_1 - x_3) + y_3(x_2 - x_1))}. \tag{12}$$

These are the solutions to x and y assuming we already know r, which we do not, but we have only really solved two of the three equations (namely eq. (4) and (5)). We can now take these results for x and y, plug them back into eq. (6) and solve for r. It is a little messy, but it is important to realize here that in doing so we get a quadratic equation in r. This is because eq. (11) and (12) is first order in r and x and y appear to second order in eq. (6). This means that in doing so we end up with something like $ar^2 + br + c = 0$, which is solvable. Because it is a quadratic equation, it yields two solutions for r and because y and x depend on r i also get two solutions for these, so we have two sets of solutions that I need to check.
As before, with the two cold pool model, I now move on and check that all conditions are satisfied.

I first check that the set of solutions are real and that $r > 0$, since I can get imaginary and negative values, solving the quadratic equation. Then I check that the point of collision is not already occupied by another cold pool, again to make sure that the wavefronts are considered active upon collision. I chose to check this differently in the three cold pool collision model, because it can be done quicker and the three cold pool model is computationally heavy. It is heavier because there are $(n-2)/3$ times as many ways to select 3 distinct elements without repetition than 2, and for each choice of 3 distinct elements I also have two sets of solutions instead of one. Instead of simply using Pythagoras, I break the process down into three steps. Step 1: check if the point is outside the blue square enclosing the cold pool, see fig. 18, if it is there is no need for step 2 or 3, but if it isn't I proceed with step 2. Step 2: check if the point is inside the red square, again if it is I can skip step 3. Step 3: use Pythagoras. This may seem more cumbersome, but checking if the distance from the center of the circle is larger than the radius of the circle using Pythagoras $\dot{x}^2 + \dot{y}^2 > r^2$ is computationally heavier than checking step 1 $\dot{x} > R$ and $\dot{y} > R$ or step 2 $\dot{x} + \dot{y} \leq R$. This of course relies on the fact that the point is most often outside the blue square, which it in our case is - if not, we could have swapped the order of step 1 and step 2.

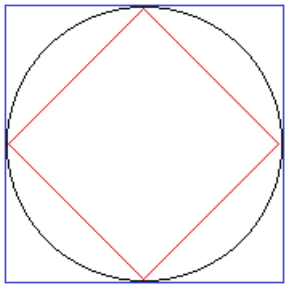


Figure 18: An illustration of how I check whether or not a point is inside a circle.

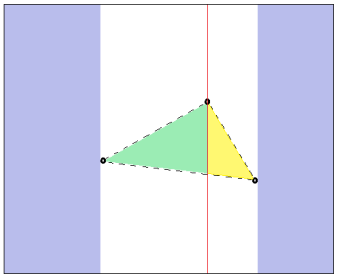


Figure 19: An illustration of how I check whether or not point resides within triangle made by connecting the three CPs.

In the three cold pool model there was also the condition that the collision point should be enclosed on all sides by the colliding cold pools, such that the air was trapped and forced upwards upon collision. This condition was mentioned in the introduction and the condition implies that the collision happens within the triangle formed by connecting the cold pool centres, see fig. 19. To check that this condition is satisfied, I first sort the x_c cold pool centre coordinates, and check that the x_p coordinate of the collision point lies within the range of the smallest x_c and largest - this is equivalent to making sure that it isn't in the light blue area. Then I check if the point is to the left or right of the red line, which I check by seeing if x_p is larger or smaller than the second largest/smallest x_c. If it is larger, I check that it is within the yellow region, by making sure it lies between the stippled lines (if it is smaller I check if it is inside the light green region). There's actually an inbuilt function in Matlab that checks this, called 'inpolygon', but it is computationally heavier to call this function than to check it with the procedure described above. If all conditions are met, I will, as before, add a row to the matrix of cold pools with the details of the new cold pool.
In summary the code runs in much the same way as the previous model, by selecting a generation, picking a cold pool, shifting it to the center of the domain (shifting everything else periodically), converting from time seeded to radii, finding nearest neighbours, going through unique ways to select three distinct elements, calculating collision points, checking conditions, adding new cold pools and repeating the process. This again yields a matrix detailing all cold pools, the generation that they belong to, the location of their centres and so on.

2.2 Thermodynamical effects

As previously discussed, simplifying a model can be both necessary and meaningful when dealing with complex systems, and is very common practise in all fields of physics. While that is true, it is also desirable to model a system as accurately as possible, when possible. Describing nature accurately has the obvious advantage of producing results that are accurately depicting nature, but it often comes at a cost (if at all possible). Increasing the complexity of a model will almost always increase the computational power needed to simulate it. Therefore when considering increasing the complexities of a model, it makes sense to consider which simplifications are worth the effort to eliminate. Some simplifications drastically decrease the workload while barely changing the result, while for others the reverse is true. It would therefore seem prudent to consider these simplifications in more detail, when considering increasing the complexity of the model, so let us discuss some of these simplifications now.

We could start by considering some of the spatial simplifications made, these would include dimensionality, topography and periodic boundary conditions. This pertains to item 1 and 2 in the list of simplifications made earlier. The simplification of a two dimensional space or completely flat surface is one that could well be a good approximation, for example over flat land or large bodies of water. The accuracy of the model, when considering a rougher terrain including hills, dense forest or large structures would then be in question. These objects could potentially obstruct the expansion of cold pools in certain directions, either completely by forming a large wall or partially by forming a steep incline. The anisotropy of the terrain would then prohibit the cold pools from expanding isotropically. We could imagine a model where the expansion of cold pools, were depending on the spatial layout, such that moving up a steep incline would decrease the velocity of the rim moving in that direction. If there were no steep inclines, the simplification of no inclines would seem justifiable. In terrain with tall mountains and deep valleys, this simplification is harder to justify, but changing this would require a completely different model, since the most fundamental assumption of the circle model is circularity. Since the simplification seems justifiable in many cases and eliminating it would require a completely new model, the assumptions made about the topography will therefore be kept.
Now the periodic boundary conditions were meant as a way to mimic a very large system, without having to compute it in full. We could however consider increasing the system directly, as periodic boundaries aren't strictly physical. We could consider increasing the size of the system, by increasing the length of the sides of the square domain. This however wouldn't change the result of any simulation. This is because, while the distances between cold pool centres would appear to increase, the relative distances would remain the same - that is all distances would increase/decrease uniformly. But it is the relative distance between cold pools that determine which will collide and whether or not the path between two/three cold pools will be obstructed by a third/fourth cold pool. So increasing the model requires adding more initial circles, but there is only so many expanding circles that we can compute. Therefore to artificially increase the system, periodic boundaries are invoked. It is a way to more easily make a 'large' system and it's not even that far a stretch. Imagine an infinitely extending system with infinitely many cold pools. If the cold pools are situated randomly, as done in the model, then the system would be more or less homogeneous - seeing as every point in space would be equally likely to be occupied by a cold pool. Then if you select a region in space, the neighbouring regions would be expected to look identical. Having identical neighbouring regions, is exactly like having periodic boundary conditions - surrounding the domain by copies of itself, was in fact discussed earlier as a way to invoke period boundary conditions. So having a system that is very large, is most effectively done by having periodic boundaries and it is therefore a good (and common) approximation. A small disclaimer: changing the size of the domain could impact the two cold pool model or a three cold pool

model with an r_{min} and/or r_{max} if these weren't scaled with the system size, but scaling system size independently has the same effect as scaling r_{max} or r_{min} independently, so the domain size should be considered fixed.

We could also consider item 3 and 5 on the list of simplifications made, the constant and equal velocities of all cold pools and circular expansion. These two simplifications are connected, even though they don't exactly imply the same things. The constant and equal velocities of all cold pools mainly imply that all cold pools are born from precipitation events of equal scale. This simplification could potentially be dealt with, without completely remaking the model. In the two cold pool collision model, having different velocities would simply shift the location of collision towards the more slowly expanding cold pool. Under the assumption that they moved with equal and constant velocity, we calculated the collision point as being in the middle of the shortest line connecting the cold pool centres, measured from the rim of the two circles. Had one circle moved a factor c quicker than the other, this point would be situated $1/(c+1)$ of the way measured from the rim of the more slowly expanding cold pool, such that if $c = 2$ (meaning one cold pool is expanding with twice the velocity of the other) the collision point would be $1/3$ of the way, measured from the more slowly expanding cold pool. This seems like a quick change and for the two cold pool collision model it is, but it is a little more tricky with the three cold pool collision model. In the three cold pool collision model, the term with r_i in (4) would need the addition of a factor that depends on relative velocities. Such a factor would change the equation to solve for r from a quadratic to a quartic (we will see the special case of the factor being 0 later). There are other subtleties to removing this assumption, for example when converting from radii to time seeded in the code we would need to account for all relative velocities, but it should still be solvable. Another 'subtlety' to having cold pools expand at different velocities, is that the concept of generations is no more, this is because a cold pool of generation $N + 1$ could reach a cold pool belonging to generation N before its own parent cold pools, if it expands faster than these. This does not seem like a very subtle change, but if we assume that we can still use the concept of generations, having varying expansion velocities could potentially be done. Now the assumption that cold pools expand as perfect circles, could possibly be chalked up as being the result of the other assumptions made, mainly the assumption about constant velocities. But the question remains, how accurately does this depict the actual shape of a cold pool. Now the LES simulations in Tompkins [12] and [10], seem to suggest somewhat circular cold pools - for reference, take a look at fig. 5 and fig. 6 in [12]. However in the paper by Torri and Zhiming [14], it is found that the cold pool is to good approximation a circle to start with, but due to collisions and interactions between cold pools the circularity quickly decays with time, suggesting that cold pools are only truly circular for a short period of time. The fact that the assumption seems to be decently valid for young cold pools, and the fact that changing it would result in the need for a completely different model, we will keep this assumption.

The simplification of constant velocities is something that could be dealt with and to some extend will be, no spoilers, but the focus of this project is mainly in considering thermodynamical effects, as alluded to by the title of the paper and this section. This is the aspect of the model that we in this project will attempt to increase the complexity of. Take for example the simplification of not considering the local environment, this is a simplification that encompasses many smaller simplifications. One environmental aspect, that could influence whether or not a new cold pool is seeded, is that of the moisture level (or lack thereof) at the collision site. Assuming that the air is very dry, the moisture carried by the cold pools might only serve to bring the moisture level up to something that is less but still dry. As discussed earlier, while the rim of the cold pools are very moist the interior of cold pools are very dry. A system of expanding cold pools, would eventually fill the entire system, making it such that all new cold pool collisions would be guaranteed to happen well inside of previous generations of cold pools. If the cold dry air at the interior of a cold pool haven't reached some sort of equilibrium

when the next generation of cold pools collide inside of it, then it may not result in a new precipitation event. Thus considering the moisture level seems sensible. There are similarly other simplifications and assumptions that we will address, such as infinitely expanding cold pools. The cold pool expands due to the cold air being less buoyant, but the cold air can only be stretched so thinly and cover a finite region, and assuming the second law of thermodynamics apply, the cold pools would also eventually reach temperatures equal to that of its environment and therefore naturally seize to be pushed outwards. Similarly it would be more realistic to have a minimal size that cold pools need to reach, before collisions would result in updrafts strong enough to carry the cold moist air through its level of inhibition. The rim of the cold pool would have to build up momentum. Besides, two cold pools infinitely close to one another, should for all intents and purposes be considered one and the same cold pool. This is also an easier modification and in the two cold pool collision model it was necessary, as discussed earlier. The last simplification that I will here mention is with regards to a model that only allows finite growth through enforcing a maximum cold pool size. The simplification lies in the fact that not all new cold pools are created through mechanical forced updrafts during collisions, but can also be created from the cold moist air reaching temperatures of the surrounding air and naturally rise (moist air being more buoyant than dry air). This means, that while the cold pool may seize to expand it will still have left a ring of moist air and these may be important in regards to creating new precipitation events that leads to new cold pools.
We will now discuss the implementation of some of these complexities, into the already existing models as described in the previous sections.

2.2.1 Size restrictions: R_{min} & R_{max}

While cold pools can expand to cover large regions and grow to sizes of more than a hundred kilometres in diameter [12], they can not expand infinitely. The cold pool will eventually be indistinguishable from its surroundings, which usually happens within a day. At this point and forward it will not contribute to convection and can no longer be credited with playing a role in triggering new precipitation events. While there are natural limits to the sizes that cold pools can reach, there are also limits to how closely two cold pool centres can be and still be considered two separate cold pools. A cold pool will even at its initial state occupy space, and this will usually be with an initial radius in the order of ten to twenty times smaller than the final cold pool radius [15]. Both of these could be reflected in the circle model by restricting the size a cold pool must be, to be considered active, by having a maximum and minimum cold pool size.

This has already been addressed in the two cold pool model, as a means to deal with nonphysical issues arising from the model. The R_{min} kept triangular configurations from causing the number of cold pools to blow up see fig. 7 and R_{max} kept the model from having self interactions see fig. 8. These issues were not inherent in the three cold pool collision model and these extra conditions were therefore omitted initially. Now we're looking to add these conditions to the three cold pool model and we can achieve this by adding a set of conditional statements in much the same way we did for the two cold pool collision model. After calculating the collision point I have a set of conditional statements that I check to see whether or not a cold pool will be seeded, and to that set of conditional statements I add a statement checking whether or not the radii of each cold pool lies within the range $R \in [R_{min}, R_{max}]$ see fig. 20.

```
if nearest(l,4) + p(t,3) <= Rmax && nearest(h,4) + p(t,3) <= Rmax && nearest(k,4) + p(t,3) <= Rmax
if nearest(l,4) + p(t,3) >= Rmin && nearest(h,4) + p(t,3) >= Rmin && nearest(k,4) + p(t,3) >= Rmin
```

Figure 20: A screenshot of the size restricting conditions in the code.

It would have looked neater if I wrote it like all($[R_1, R_2, R_3] \leq R_{max}$) but using all() is heavier than writing three separate conditions, so it is a matter of efficiency. To avoid calculating the collision point between cold pools that are spaced further apart than $2R_{max}$ I add a conditional statement before calculating the point, testing to see that the distance between the cold pools are less than $2R_{max}$. We could also include a conditional statement testing to make sure that the cold pools in question are not too close, this would be a little trickier than simply saying $d > 2R_{min}$ (with d being the distance between two cold pool centres). This is because even though the side lengths of a triangle, created by connecting three cold pool centres, were less than $2R_{min}$, the circumcenter could be a distance larger than R_{min} from all three vertices see fig. 21. To see that in the case of an equilateral triangle $R > R_{min}$ one could use eq. (13) with a, b and c being equal to $2R_{min}$ and A, B and C equal to $\pi/3$. So the condition should be that the distance from a vertex to the circumcenter should be larger than R_{min} or the radius of the circumscribed circle needs to be larger than R_min. Finding the distance R either by calculating the intersection of the perpendicular bisectors and then the distance to one of the vertices or by using that the radius of the circumscribed circle is

$$2R = \frac{a}{\sin(A)} = \frac{b}{\sin(B)} = \frac{c}{\sin(C)}, \tag{13}$$

would defeat the purpose of having the condition to begin with - which is to ease calculations. Besides if the triangle is obtuse, the circumcenter is outside the triangle and it would not pass the triangle condition regardless of whether or not the radius of the circumscribed circle is larger than R_{min}.

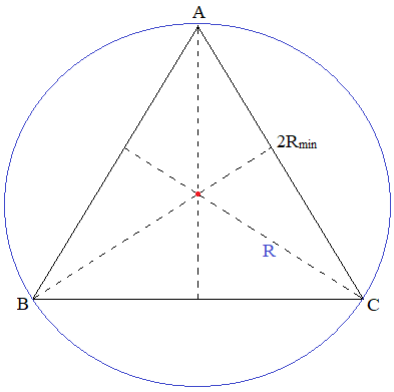


Figure 21: An illustration of the circumcenter of an equilateral triangle with side lengths $2R_min$. R is the radius of the circumscribed circle.

With a maximum and a minimum radius for cold pools, we must also consider how we treat the condition that a cold pool can not be seeded if the location is already occupied by other cold pools of the same generation - this condition was due to wave fronts becoming inactive upon colliding. If we consider a cold pool to not expand beyond R_{max}, the condition ensuring that no other cool pool of the same generation occupies the space, could be changed such that we exclude any cold pools further away than R_{max}. This however affects the assumption, that we can consider cold pools to be divided into generations in the first place. Because if cold pools at any point stop expanding, it is no guarantee that cold pools of generation N occupy a space, before any cold pools of generation $N + 1$ see fig.

22 for an illustration of this. Therefore I will choose to consider cold pools to be expanding infinitely and collide with other cold pools creating static and inactive wave fronts, but not be able to create new cold pools when expanding beyond R_{max}, similarly to what we're already doing in the two cold pool model. This makes it such that I do not need to change the condition that is already in place and that we can still use the concept of generations, the latter we quite heavily rely upon.

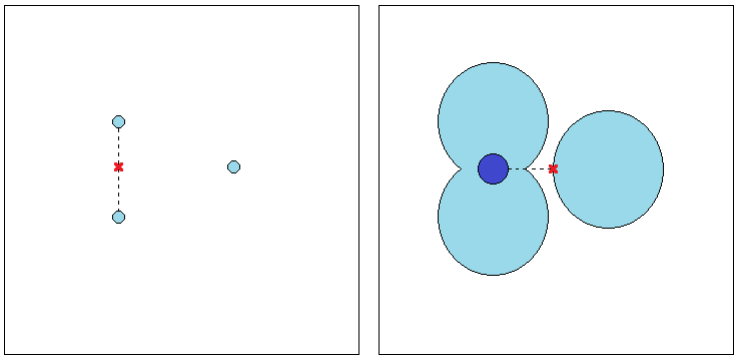


Figure 22: An illustration of how the concept of generations are broken when we restrict cold pool sizes. The dark blue cold pool belonging to gen. $N+1$ can interact with the light blue cold pools of generation N that have all reached R_{max}.

2.2.2 Stochastic process and scaling

In the circle model, when two or three cold pools collide and every condition is satisfied, a new cold pool will be seeded at the collision site 100% of the time. It is at the moment a completely deterministic process. Seeing as the model is based on simplifications and idealizations of nature, it is perhaps reasonable to assume that it should not be completely deterministic. Other factors such as the local environment at a collision site (moisture levels, temperature or wind levels) could potentially play into whether or not a collision between cold pools leads to future precipitation events and subsequent cold pools. But even in the absence of such forces, in a completely isolated system of cold pools, the process may depend on things such as the cold pool sizes. The cold pool size is directly related to its age and thus indirectly related to the velocity of its gust front. The radial velocity of the gust front decreases with the age of the cold pool [16], thus one could expect the collisions to be less volatile and the driving mechanism to be weakened, when larger/older cold pools collide. While the radial velocity decreases, so does the moisture level. The moisture came from evaporated rain during the precipitation event and was carried outwards by the gust front. This moisture is being spread thinly as the cold pool increases in size. The probability to trigger new events, when cold pools collide, could also depend on this. Therefore it seems reasonable that the process shouldn't be completely deterministic, but stochastic instead and that the probability should scale with cold pool sizes.

The way this is implemented in the model is by adding another condition of the type 'if' $L \leq \frac{C}{R_1 R_2 R_3}$ is satisfied then proceed, with L being a randomly generated number and C a constant factor. L lies in the interval $[0, 1]$ and takes any value within that interval with equal probability and the value of the constant C can for example be chosen such that the fraction is 0.5 when the product $R_1 R_2 R_3$ equals $(R_{max}/2)^3$ or it could perhaps be fixed such that the fraction is approximately 0.5 when the average size/age of the cold pools in question R_i (with $i \in \{1, 2, 3\}$) are equal to some other characteristic length of the system. The fraction being 0.4 would mean that there's a 40% of seeding a new cold pool (if all other conditions are met), as $2/5$ of the values in the interval $[0, 1]$ are smaller than or equal to 0.4. With this condition the addition of new cold pools is a stochastic process depending on the

product of cold pool radii. As the cold pools grow, the fraction decreases, yielding the desired result that larger/older cold pools are less likely to seed new cold pools. For small cold pools the fraction may be larger than 1, in which case the condition is always met and a new cold pool is seeded, for large cold pools the fraction will be small but finite, yet in the two cold pool model the probability to seed a cold pool will still be 0% if any radii exceeds R_{max}.

2.2.3 Noise

In the original circle model new cold pools are seeded at the exact location of the collision that generates them and they are seeded instantaneously. The process from an initial collision to a new precipitation event is of course not instantaneous and any wind would shift the location at which new precipitation events occur. But while any wind may shift where a new precipitation event takes place, not all types of winds matter, take for example trade winds: These are winds that blow from east to west near the equator and they are caused by warm moist air at the equator rising due to increased buoyancy, leaving a low pressure, which in turn causes colder air closer to the poles to move towards it. Due to coriolis forces winds in the northern hemisphere are deflected to the 'right' (taken north to be up) and winds in the southern hemisphere are deflected to the left, so when air moves from the north pole to the equator it will be deflected from east to west and the same happens for air moving from the south pole towards the equator. Trade winds could be considered a constant shift to the system, but any constant shift both in space and time would leave the system unchanged in our model. This is because only relative positions and sizes matter. However not all winds are as predictable as trade winds and these more random winds could potentially influence the system greatly.

To implement a random shift in the location of newly seeded cold pools, all we need to do is randomly add/subtract values to the x and y positions of these. Likewise to shift these in time, I randomly add a value to the time that they're seeded. I obviously wont consider subtracting from the time seeded, as a new precipitation event couldn't occur before the collision event that triggers it in the first place. This change leaves the calculation of collision sites unchanged and doesn't add a condition to check for, all we do is add a term like $c(-1+2\text{rand})$ to the x position. With c being a constant that we define at the beginning of the code, determining the limits of the shifting and $(-1+2\text{rand})$ is a random number in the interval $[-1,1]$. At the beginning of the code I define two constants, one that sets the limits for shifts in space and one for time, such that they can easily be set independently.

2.2.4 Memory field

While the air at the rim of a cold pool is moist the interior of the cold pools are dry [12]. Within the circle model, every cold pool collision at any generation, except for the first, will happen inside another cold pool. This dry air could impact the probability that a collision leads to a new event. If the air at some location is very dry, then the air after a collision at that location may just have reached 'normal' values and the air having high moisture levels at impact was a prerequisite for the collision to trigger new precipitation events and subsequent cold pools. It is the impact of this lack of moisture or dryness on the dynamics of cold pools, that I want to look at with this variant.

The way I intend on implementing this, is by adding a conditional statement to our set of conditions, that will compute the probability that the cold pool is seeded, depending on some field and its value at that location. I imagine the field to be a scalar field describing the dryness at any given location at any given time, and I assume the value to depend on the youngest cold pool to cover the area and that cold pools age. Such that if it is relatively old, the moisture level will have had time to

come back to some sort of equilibrium value and it wont impact the probability of the collisions to trigger new events. This field could be considered in full and be updated regularly, but this would be computationally heavy and ineffective. Since the only relevant information is the age of the youngest cold pool that covers the location (due to this being related to the dryness of the air at that location), it would be more effective to find this information specifically when we need it - and this is after we have computed a collision. The youngest cold pool to cover the location of a collision site, will always be a cold pool from the previous generation. So if we're computing collisions in generation N, we should look at cold pools in generation $N-1$ to find the youngest cold pool already covering the area. It could technically be a cold pool from generation N itself, but that would mean that the three/two cold pools in question wouldn't seed a new cold pool anyway, since the area is already covered by another cold pool of that generation (a condition that came as a consequence of wave fronts becoming static once they had collided). Due to this it is natural for this new condition to come after this already existing condition, to avoid needlessly calculating stuff. Now we're looking at generation $N-1$ and we want to find the youngest cold pool that covers a specific location, this is done by computing the distance from all cold pools centres of generation $N-1$ to the location and subtracting this from their radii at the time of collision. With these values, I find the lowest positive value, which represents the youngest cold pool to cover the area and use its index to find that cold pools size/age. With this information I need to seed the new cold pool with some probability, such that a larger/older cold pool covering the area is less likely to prohibit the seeding of a new cold pool and at some point it shouldn't affect the probability at all.
You can see how the condition is written in fig. 23, with $C1$ being the distance from the center of a cold pool in generation $N-1$ to the collision site, $D1$ is its radius at the time of collision and $E1$ is the difference between these, also illustrated in fig. 24.

```
if j > 1
%Calculate distance to point for prevgen
C1 = sqrt( (p(1,1)-prevgen(:,1)).^2 + (p(1,2)-prevgen(:,2)).^2 );
D1 = prevgen(:,3) + p(1,3);
%Pick youngest and use that as scale
E1 = D1 - C1;
index = find(E1 == min(E1(E1>0)));
%Now we make it scale:
scale = c*D1(index);
L1 = rand;
end
if L1 <= scale
```

Figure 23: A screenshot of the memory field condition in the code.

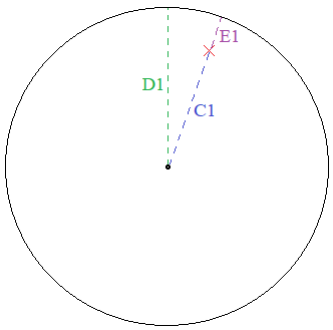


Figure 24: An illustration of the distances $C1$, $D1$ and $E1$.

Prevgen is a matrix containing the information of cold pools from generation $N-1$ (these have naturally also been periodically shifted in space and converted from time seeded to relative radii) and c is a scaling factor defined in the beginning of the code. The probability that a new cold pool is seeded depends on the factor called "scale", which is related to the size/age of the cold pool. I generate a random number $L1$ between $[0,1]$ and accept the condition if $L1 \leq scale$. This can be understood as $scale$ being the probability that a new cold pool is seeded, and it goes from $[0,1]$ (and above 1), as $D1$ increases, linearly with gradient given by c.

2.2.5 Diurnal cycle

In the model when collisions occur, new cold pools are seeded and start to grow immediately, this is another simplification that I intend to look into. The mechanism by which new cold pools were created were explained in detail earlier in the introduction, but note that they were generated by rain events. These rain events, while they may be generated by colliding cold pools born from other rain events, they do not happen immediately upon cold pool collisions. The process takes time and the precipitation event is more likely to happen during certain parts of the diurnal cycle [17], [18] and [19] - a diurnal cycle, is a cycle of recurring events typically every $24h$ as a result of the earth rotating around its axis [1]. The idea is to apply this diurnal cycle structure or pattern to our model, such that instead of having cold pools be seeded as collisions happen, they are instead seeded with some pattern. If we assume cold pools of one generation to approximately be born from events taking place at approximately the same time or during the same day, we could make the pattern such that generation N are rain events that all fall at the same time during a diurnal cycle and $N + 1$ would be events that happen during the next cycle.

The implementation of this is probably the simplest, because no new calculation is needed and no new conditions are required, all we need to do is fix the time that cold pools of some generation is seeded. In fact, since there is no reference to generations prior to generation N when calculating the cold pools of generation $N + 1$, I can completely remove all reference to time and just assume that generation N happen earlier than generation $N + 1$ and later than $N - 1$. All cold pools within a generation is thus seeded at the same time and are at any given time, exactly equally large. In practice this was done making the 'time seeded' equal to zero for all new cold pools.

2.2.6 Persistence of cold pools and moisture rings

We've often talked about or referred to new precipitation events being born from mechanical lifting of moist air through cold pools colliding, but the ring of moist air can trigger new events even after the cold pools have stopped expanding and their vertical wind shears are low [12]. The paper suggest that the triggering of new events could be more thermodynamical in nature, rather than solely dynamical by forcing the moist air upwards with high vertical wind shears. They found that the CAPE values peak at the rim of a cold pool, because of the moist air. CAPE stands for 'convective available potential energy' and it is used in predicting the intensity and the location of convective activity. It is calculated by integrating the buoyancy of a parcel of air up to its level of neutral buoyancy. Despite the air being cold, the moisture makes it such that the CAPE value is positive and the parcel of air will rise. Because of this I imagine a model, where the cold pools expand to some R_{max}, but instead of completely dying out, they leave a ring of moist air that remains active. This moisture ring could then interact with other cold pools to trigger new events. This however creates a 'small' problem: Cold pools can no longer be defined as belonging to a generation. The concept of generations naturally came to be, through the assumption that once wave fronts collide they become static or inactive combined with cold pools expanding infinitely at constant and equal velocities. This assumption made it impossible for a cold pool of generation $N + 1$ to collide with an active wave front of any cold pool belonging to generation N. We discussed the issue of the concept of generations breaking in section 2.2.1, when we talked about restricting cold pool sizes, see fig. 22 for an illustration of how this breaks with the concept of generations. In the size restricting models, we assumed that they continued to expand infinitely and would collide with cold pools of the same generation, but that they were too weak to trigger new events once they had expanded beyond some R_{max}. This would however not be useful for the model that I now have in mind, because if a ring of moisture is left with a radii of R_{max} but the cold pool

continued to grow, leaving the ring of moisture behind, then no other cold pool could reach the ring of moisture without first becoming inactive itself, see fig. 25 for an illustration of this. Therefore I will not assume that they continue to grow beyond R_{max}, but I will rather assume that the concept of generations approximately holds true. There are cases where cross generation collisions could occur, but the majority of collisions would still happen between cold pools of the same generation, and we will ignore the ones that don't. The idea is then to make a variant of the model where cold pools grow to some R_{max}, stops growing, but leaves a moisture ring of radii R_{max} that can interact with other cold pools of the same generation. The interaction will still be a collision, but the driving mechanism for triggering new events will not be a mechanical lifting, but rather a thermodynamical one - where the collision point is a point of large CAPE value.

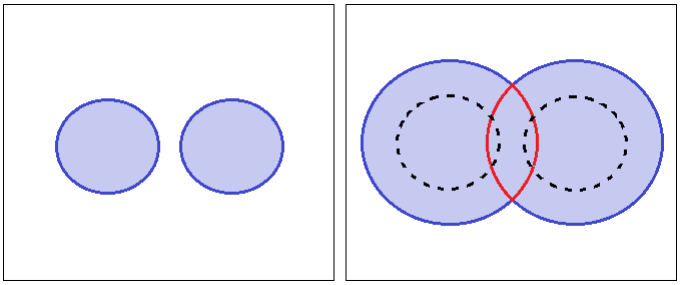


Figure 25: Two cold pools expanding, leaving behind a moisture ring indicated by a stippled black ring. The wave front is 'inactive' once it collides with the moisture ring, indicated by red.

That the concept of generations did not exactly hold true, was not the only obstacle that I encountered while making this model, the implementation was also trickier than one might expect. I will show you how I approached solving this and why this was **very** ineffective (it borderline fails) for the three cold pool circle model and then I will discuss how I then approached solving it.
The biggest change to this model is in how we will calculate the point of collision between cold pools/moisture rings, and the overall approach will be to solve for this in a few steps. The steps differs slightly for the two and three cold pool circle model, so I'll address the easier one first, namely the two cold pool circle model.

Two CP circle model:
Before calculating the collision point between the two cold pools, I make sure that the distance between their centres don't exceed $2R_{max}$, similarly to what we did in the size restricting model - to avoid needless calculations. If they are sufficiently close, I move on to calculating the collision point between them, and this falls into two steps. In the first step we assume that the collision happen before either of the cold pools reach R_{max}, thus calculating the point of collision in the usual way. We then check whether or not this assumption held true, by checking if either of the two cold pools grew to be larger than R_{max} at the time of collision. If they did not exceed R_{max}, we proceed normally, by checking that all other conditions are met and if so we add a cold pool at the point and time of collision. If however one of the two cold pools grew to be larger than R_{max}, we proceed to step two. In step two, we take the larger cold pool and assume that it has reached R_{max}, we then find the point of collision, as the intersection between the line connecting the cold pool centres and the circle of radii R_{max}. The time it took for this to happen, is given by the amount that the smaller cold pool has to grow to reach this point (remember time and cold pool sizes are directly related). This outlines how I calculate the collision point and time between two cold pools and fig. 26 illustrates the steps. I also made a slight change to one of our conditions, namely the condition that no other cold pool of that generation occu-

pies the space already. The condition is changed, such that cold pools don't exceed R_{max}, by making all entries larger than R_{max} equal to R_{max}, by writing something like $M(M > R_{max}) = R_{max}$ and then checking that the condition is satisfied.

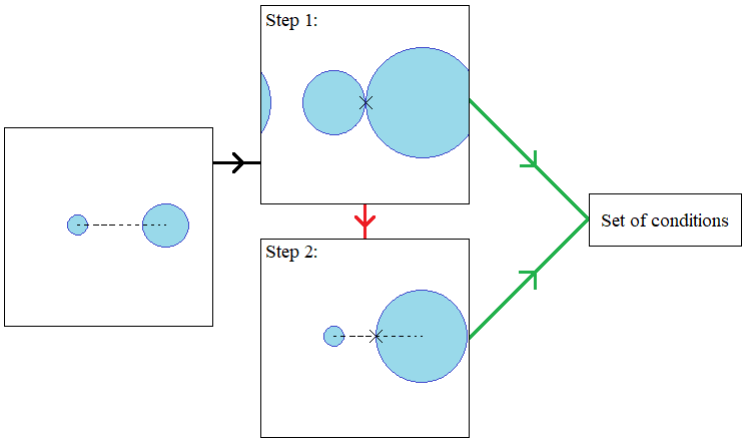


Figure 26: An illustration of the steps taken to calculate the collision point marked by a black cross.

Three CP circle model:
While the two cold pool circle model proceeded painlessly, the three cold pool model is slightly trickier as we will see. Before we begin calculating the collision point, we will again make sure that none of the three cold pools in question are spaced further apart than $2R_{max}$, measured from their centres. If they are close enough to interact, we proceed with calculating the collision point. The process of calculating the collision point will again be split into steps, in this case three steps, which I will outline before diving into any details. The first step is where we again assume all three cold pools to be expanding without restrictions, we calculate the collision point and check whether or not any of the three cold pools have exceeded R_{max}. If not, we proceed to check our set of conditions, but if either of them have grown to be larger than R_{max} we proceed to step two. In step two we grow all three circles, until the largest reaches R_{max} and then we let the remaining two cold pools expand without restrictions. We find the collision point and we again check that neither of these have grown beyond R_{max}. If neither grew to be larger than R_{max} at the time of collision, then we proceed to check that all conditions are met, but if either of them grew to be larger than R_{max} then we go to step three. In step three we also grow the second largest cold pool, such that it has a radii of R_{max} and let the third and smallest cold pool grow without restriction. We check that this last cold pool didn't grow beyond R_{max} and if it didn't we continue to our set of conditions, but if it did we conclude that no collision were to be found between the three cold pools. The order in which this is done (step one, two and three) ensures that the collision point (if one exists) will be the first point of intersection. This may seem as painless as the two cold pool circle model and the first and third step is, but the second step is not. The first step is calculated in the usual way, so this is no issue. The third step is actually easier to calculate than the first step, because once two of the three cold pools have reached R_{max} and overlap there will at max be two intersections between these. The only places where a collision between the three could occur is at one of these two points. The closest of the two will be the first point of intersection between all three cold pools and the time it takes for this to happen is the amount that the smallest cold pool will have had to grow before reaching this (from start to finish), see fig. 27 for an illustration of step three.

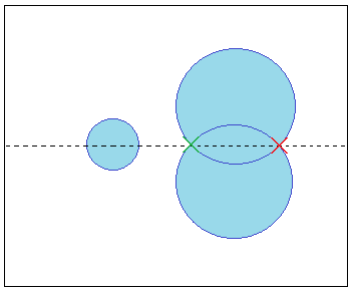


Figure 27: An illustration of step three in the persistent variant of the three cold pool circle model. The two larger cold pools have reached R_{max}, the intersections are marked by crosses and the green cross is where the smaller cold pool will first collide with the two larger cold pools.

Step three was easy, because there were only two options for a collision between all three cold pools, this is not the case in step two so we have to resort to actually calculating the new point of intersection between all three cold pools. The most obvious way to go about calculating this point, is by revisiting eq. (4), (5) and (6), but now with one cold pool having reached R_{max}

$$(x - x_1)^2 + (y - y_1)^2 = (r + r_1)^2 \tag{14}$$
$$(x - x_2)^2 + (y - y_2)^2 = (r + r_2)^2 \tag{15}$$
$$(x - x_3)^2 + (y - y_3)^2 = r_3^2 \qquad \text{with } R_3 = R_{max}. \tag{16}$$

The only difference between now and earlier, is the missing variable r in eq. (16), this however makes all the difference. We could now attempt to solve this in a way similar to what we did earlier, by moving all constants to the RHS of the equations, subtract eq. (16) from eq. (14) and (15) to get a set of equations like

$$2x\,(x_3 - x_j) + 2y(y_3 - y_j) = r^2 + 2rr_j + C_j - C_3 \qquad \text{with } C_j = r_j^2 - x_j^2 - y_j^2 \text{ and } j \in \{1, 2\}. \tag{17}$$

Now this looks almost similar to before, except we're missing a $-2rr_3$ and have an extra r^2 term on the RHS. The r^2 term is what makes this more difficult, because solving eq. (17) for x and y would return solutions that depend on r^2, such that when we plug these back into eq. (16) to solve for r, we're left with a quartic instead of something that is quadratic in r

$$\left(\frac{r^2y_1 - r^2y_2 - 2rr_1y_2 + 2rr_1y_3 + 2rr_2y_1 - 2rr_2y_3 - C_1y_2 + C_1y_3 + C_2y_1 - C_2y_3 - C_3y_1 + C_3y_2}{2(x_1y_2 - x_1y_3 - x_2y_1 + x_2y_3 + x_3y_1 - x_3y_2)} - x_3\right)^2$$
$$+ \left(\frac{r^2x_2 - r^2x_1 + 2rr_1x_2 - 2rr_1x_3 - 2rr_2x_1 + 2rr_2x_3 + C_1x_2 - C_1x_3 - C_2x_1 + C_2x_3 + C_3x_1 - C_3x_2}{/2(x_1y_2 - x_1y_3 - x_2y_1 + x_2y_3 + x_3y_1 - x_3y_2)}\right)^2$$
$$= r_3^2. \tag{18}$$

This quartic can be solved analytically by hand or by using something like Mathematica - neither Maple nor Mathematica can solve eq. (14), (15) and (16) for x, y and r when written in that form, but Mathematica can solve for r when it is written in the form of eq. (18). The solution can not be written on a single page, and when copying and converting from Mathematica syntax to Matlab syntax (so as to make a Matlab function for solving step 2) the solution is more than 13000 lines, see fig. 28 for a screenshot of the last few lines. While this was an accurate and working function, it took way too long for Matlab to compute this. I couldn't run the model for a large number of initial cold pools or for many generations without it crashing or taking ages, so I had to think of another way to solve this.

My second attempt was to try to numerically solve eq. (18) instead of analytically solving it. I made a function to solve for step two, using an inbuilt Matlab function $vpasolve()$, see fig. 29 for a screenshot of it. This was however not that much quicker than using the function I created with the analytical solution, because $vpasolve()$ is part of the Symbolic Toolbox of Matlab, which calls a function named 'Mupadmex' every time it is being used, which again makes it take forever.

```
12997   x2.^2.*y3.^2+4.*x1.^2.*x3.^2.*y3.^2+(-8).
12998   x2.^2.*x3.^2.*y3.^2+4.*x2.^2.*y1.^2.*y3.^
12999   y3.^2+4.*x3.^2.*y1.^2.*y3.^2+(-8).*x1.*x2
13000   x3.*y1.*y2.*y3.^2+8.*x2.*x3.*y1.*y2.*y3.^
13001   y3.^2+4.*x1.^2.*y2.^2.*y3.^2+(-8).*x1.*x3
13002   y2.^2.*y3.^2+8.*x1.*x2.*y1.*y3.^3+(-8).*x
13003   x1.*x3.*y1.*y3.^3+8.*x2.*x3.*y1.*y3.^3+(-
13004   x1.*x2.*y2.*y3.^3+8.*x1.*x3.*y2.*y3.^3+(-
13005   x1.^2.*y3.^4+(-8).*x1.*x2.*y3.^4+4.*x2.^2
13006   1/3)).^(-1/2)).^(1/2)];
```

Figure 28: A screenshot of my first attempt to create a function to solve step 2.

```
1   function cper = ppointtestii(x1,y1,r1,x2,y2,r2,x3,y3,rm,z)
2 -   C1 = r1^2 - x1^2 - y1^2;
3 -   C2 = r2^2 - x2^2 - y2^2;
4 -   C3 = rm^2 - x3^2 - y3^2;
5
6 -   digitsOld = digits(2);
7
8 -   r = vpasolve((-(-z^2*y1 + z^2*y2 + 2*z*r1*y2 - 2*z*r1*y3 -
9 -   x = -(-r.^2*y1 + r.^2*y2 + 2*r*r1*y2 - 2*r*r1*y3 - 2*r*r2*y.
10 -  y = (-r.^2*x1 + r.^2*x2 + 2*r*r1*x2 - 2*r*r1*x3 - 2*r*r2*x1
11
12 -  cper = [x y r];
13
14 -  end
```

Figure 29: A screenshot of my second attempt to create a function to solve step 2.

My third attempt at an exact solution was to consider a rewriting of the problem, such that I perhaps could get it on a form that was more readily solvable and with a neater solution. With this I had two ideas, the first was in considering the line drawn out by the intersection points between two growing cold pools, see fig. 30 for an illustration of said line. The curvature of the line depends on the relative sizes of the growing cold pools, if they're equally large it will be a straight line going perpendicular with the line connecting their centres and crossing it at the midpoint - the curvature of the line may be such that it never crossed the third cold pool, as in fig. 30. I found an expression for the stippled line by starting with eq. (14) and (15) and solving for x and y

$$y = \frac{-2(AB - Ax_2 - y_2) \pm \sqrt{(2(AB - Ax_2 - y_2))^2 - 4(A^2+1)(B^2 - 2Bx_2 - r^2 - 2rr_2 - C_2)}}{2(A^2+1)} \tag{19}$$

$$x = Ay + B \qquad \text{with } A \equiv \frac{y_1 - y_2}{x_2 - x_1} \text{ and } B \equiv \frac{2r(r_1 - r_2) + C_1 - C_2}{2(x_2 - x_1)}. \tag{20}$$

Both x and y were functions of r and I needed a third equation to solve it completely. Now the only third equation that I have is eq. (16), but if I just insert these and solve for r then it would have been no different from our first attempt. Instead I imagined only taking one of the solutions for y (for example the negative solution) and expressing the third cold pool as a half circle. The y solution would be the bottom line curving towards the larger grey cold pool, seen in fig. 30 and the green top of the cold pool would be the half circle. The idea was to check for an intersection between these and hope that since I had narrowed it down to looking at only one of the curved lines and only half the circle of the third cold pool it would return neater results, but the equations to solve resulted in another quartic with an equally ugly solution. The other attempt at getting an exact solution, were to approach it very differently. The idea was to express the shortest distance from the rim of the growing cold pools to the rim of R_{max} cold pool, as a function of the angle θ, see fig. 31. I then needed to equate these distances, $D_1(\theta)$ and $D_2(\theta)$, and find a θ that solves this. When these equate I would have an intersection between all three, because if the shortest distance from some point on the rim of the smaller cold pools to some point on the large R_{max} cold pool is equal, then they're going to collide at that point at the same time, since they're expanding with equal velocities - and the shortest path is of course the straight line path, also illustrated in fig. 31.

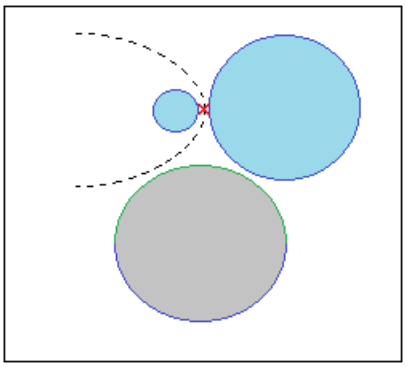


Figure 30: An illustration of the third attempt at an exact solution, expressing the collision line (the dashed line), looking at the lower part (negative y solution) and solving for intersections with the top half (marked green) of the R_{max} cold pool (seen in grey).

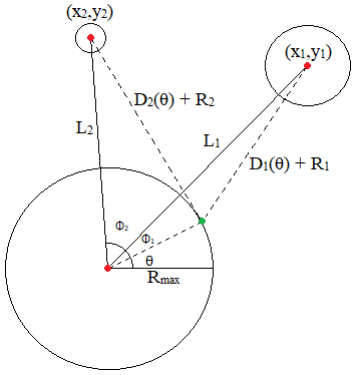


Figure 31: An illustration of the last attempt at getting an exact solution. The red dots mark the cold pool centres and the green dot marks spot on the rim of the R_{max} cold pool given by the angle Θ.

To express $D_1(\theta)$ or $D_2(\theta)$ in terms of known quantities, I used the law of cosines

$$(D_j(\theta) + R_j)^2 = L_j^2 + R_{max}^2 - 2L_j R_{max}\cos(\phi_j), \text{ with } \phi_j = \arctan\left(\frac{y_j}{x_j}\right) - \theta \text{ and } j \in \{1, 2\}. \quad (21)$$

This however can not be written on the form $D_1(\theta) = D_2(\theta)$ and solved for θ unless we're in the special case of $L_1 = L_2$ which is rarely the case. This concluded my attempts at an exact solution, and I accepted having to use an approximate solution. I could have started by just explaining the approximate solution that I was going to employ, but since everything else had been exact and analytical, I thought it would be a good idea to express why this one step in one the models weren't. Now there were several ways to make an approximate solution and the first two that came to mind, were related to my attempts at exact solutions. One approximate solution, could be to solve the quartic using Newtons method, where one starts by guessing a solution x_0 and in n steps approaches a more accurate solution by

$$\boxed{\text{Newtons method}: \quad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}} \quad (22)$$

The idea is that you update your initial guess x_0 by continuously drawing the tangent line to the point $f(x_n)$ and compute where this tangent line intersects with the x-axis and this intersection is typically a better guess for the root of a function, see fig. 32 for an illustration of this. There are however cases where this fails, like if an iteration point is stationary such that the tangent line never intersects with the x-axis. When we're automating this process and doing it many times for many cases, I would not want to stumble into these kinds of issues and so I rejected this method. Another approximate solution would be to assume that $D_1(\theta) + R_1 = D_1(\theta)$, in this case equating $D_1(\theta)$ and $D_2(\theta)$ to solve for θ becomes possible. However the solution that I compute is of course not the exact solution, but worse yet is that I may find a solution where there shouldn't be one in the first place, and so I rejected this too. The approximate method that I ended up using was one that was in line with the original idea of expressing these distances $D_j(\theta)$ (with $j \in \{1, 2\}$). The idea was to compute the intersection between the shortest path between the growing cold pools and the R_{max} cold pool and the rim of the R_{max} cold pool. These would be the points where the smaller cold pools first collide with the R_{max} cold pool respectively. Having these points, I could also calculate the distance from one of the growing cold pools to the first point that the other growing cold pool collides with the R_{max} cold pool, see fig. 33

for an illustration of said lines and points. The green stippled line from CP 2, in fig. 33, is the shortest path from that cold pool to the R_{max} cold pool and the intersection between this line and the rim of the R_{max} cold pool is denoted by a green dot. The green stippled line from CP 1 is also the shortest path from that cold pool to the green dot. The purple stippled lines and purple dot are defined similarly. Now the idea is to compare the shortest distances from the green dot to the rim of CP 1 and CP 2 (this is the distance from the green dot to the red dots denoting the centres of CP 1 and CP 2, with their respective radii subtracted), and similarly compare the shortest distances between the purple dot and the rims of cold pool 1 and 2. If the shortest distance from the green dot to the rim of CP 2 is shorter than the shortest distance from the green dot to the rim of CP 1 **and** the shortest distance from the purple dot to the rim of CP 1 is shorter than the shortest distance from the purple dot to the rim of CP 2, then we know that there will be a point on the rim of the R_{max} cold pool, between the purple and green dot, where CP 1 and CP 2 intersect at the same time. We do not know exactly where this happens, but we know that it will happen, since they grow with equal and constant velocities. I want to add that we do not need to consider any points outside of this region, because any such point would be rejected by the triangle condition anyway.

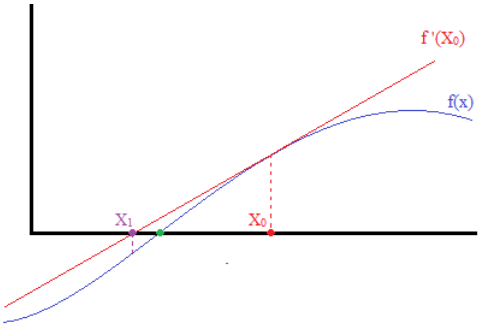


Figure 32: An illustration of Newtons method. The blue line is the function $f(x)$, the red dot denotes our first guess x_0 and the red line is the tangent line at $f(x_0)$. The intersection between the line $f'(x_0)$ and the x-axis marks the next guess x_1, denoted by a purple dot.

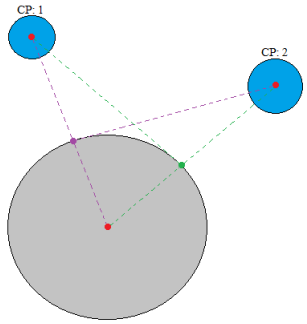


Figure 33: An illustration of the approximate method that I ended up using to solve step two. The larger grey circle is the R_{max}, the smaller blue circles are the still growing cold pools numerated 1 and 2. The stippled green line from CP 2 is the shortest path from this cold pool to the R_{max} CP and the stippled green line from CP 1 to the green dot is the shortest line from CP 1 to this point on the R_{max} CP. The purple stippled lines are similarly defined.

From this point on, we could approach it in a few different ways. Knowing that the point is somewhere between the green and purple dot, we could just assume that the actual point of intersection on the rim will be directly between these. We could also do this in iterations, similar to newtons method, where we compare the distances between these dots and the rims of CP 1 and 2, with the dots getting progressively closer and closer together, such that we approach an exact solution. We could also use the difference between the sizes of CP 1 and CP 2 and their relative distances to the R_{max} cold pool, to determine on which side of the midpoint between the green dot and purple dot the actual intersection lies - if CP 2 is larger than CP 1, the actual point of intersection is going to be closer to the purple dot or if CP 1 is further away from the R_{max} cold pool than CP 2, the intersection will be closer to CP 2. If it lies closer to the purple dot, we could pick the midpoint between the purple dot and the midpoint between the purple and green dots, and assume that this is the point where they all collide. An illustration of these options can be seen in fig. 34. The second option is the heaviest computationally, and since the three cold pool collision is already quite heavy, I chose not to use this option. The first and third option is similarly heavy, but the third option is slightly more accurate and I therefore went with that option.

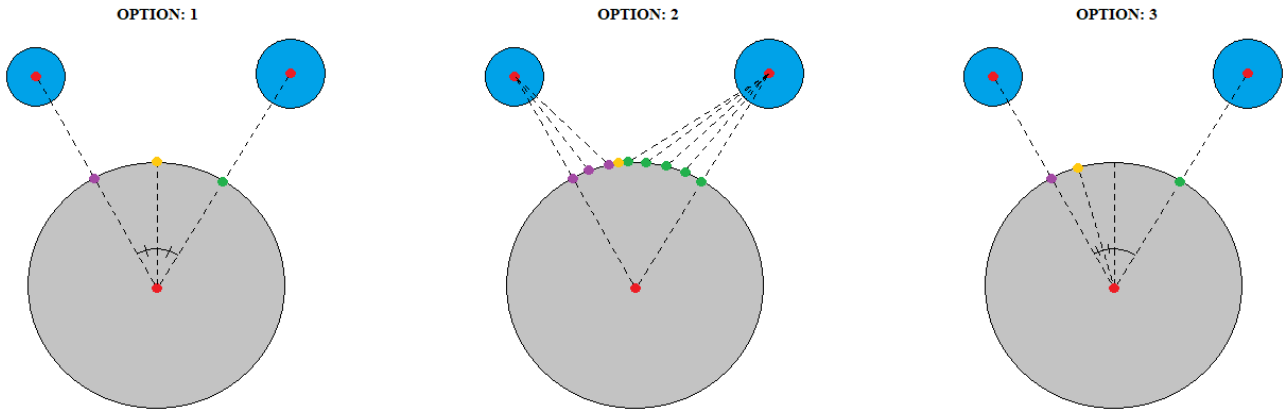


Figure 34: An illustration of the options on how to determine the collision point (marked by a yellow dot).

So this was how I did all three steps in this variant of the three cold pool circle model, step one and three were exact and step two were an approximation. As with the two cold pool circle model, I again changed that one condition where I check whether or not any other cold pool occupies the space already, such that all cold pools grow to be at max R_{max} in radius.

Note: All newer variants were build on the original circle model (two or three cold pool model), so for example not every variant of the three cold pool collision model has a maximum/minimum radius, but every variant of the two cold pool model does.

3 Results

Now it is time to use the various versions of the models, described in the previous section, run simulations and see how these different effects affect the result. The results that I will be looking at can be split into two, with the first result being related to how the number of cold pools change with generations, to see whether it is sustainable in isolation and to compare which of these effects 'kills it off' the quickest or prolongs it the most. The second result that I will be looking at is in terms of how the cold pools distribute themselves on the domain, as we go to higher and higher generations. We want to see whether they space themselves regularly and spreads thin across the domain, if they maintain a random order or if they cluster at certain locations (self-aggregation). But before I get into either of these, I want to look at some of the factors that I've introduced into some of the models, such as R_{min}, R_{max}, the spatial and temporal noise constants or the scaling constants c in the scaling model as well as the memory field model. I will do this to get a sense of how they work and how sensitive the model is to a change in this variable - this will also help me select a reasonable value for these constants, when I in the following sections simulate with the intend to compare the different models.

3.1 Models in isolation: looking at the factors

The first set of constants that I want to look at, is R_{min}, R_{max} and the initial number of cold pools N_I. This set of factors, is a set shared by more than half the models, as every two cold pool collision model has R_{min} and R_{max} and there's one three cold pool collision model with all three as well. Now I already know that if I decrease the number of initial cold pools the simulation dies off more rapidly and if I increase R_{min} and decrease R_{max} enough it also dies off faster, because it narrows the interval of 'active' cold pool sizes and forces dismissals of many collisions. But how do they interplay? It

seems like they have a similar function, so maybe I could just vary one of these (R_{min} and R_{max} or initial number of cold pools) while keeping the other constant and still get the full picture. So what I did was I looked at the standard two cold pool collision model and ran the simulation for various initial number of cold pools (200, 400, 600, 800 and 1000) for the same seed and for 25 generations, whilst varying R_{min} or R_{max} in a way such that $N_I R_{min}$, $N_I R_{min}^2$ or something similar, was held constant. The idea being that if I could achieve a similar picture when varying these factors in a certain way, then they are not independent and going forward if I ever were to want to vary these, I should keep one constant and vary the other (and not vary both). You can see the results of this in fig. 35 below.

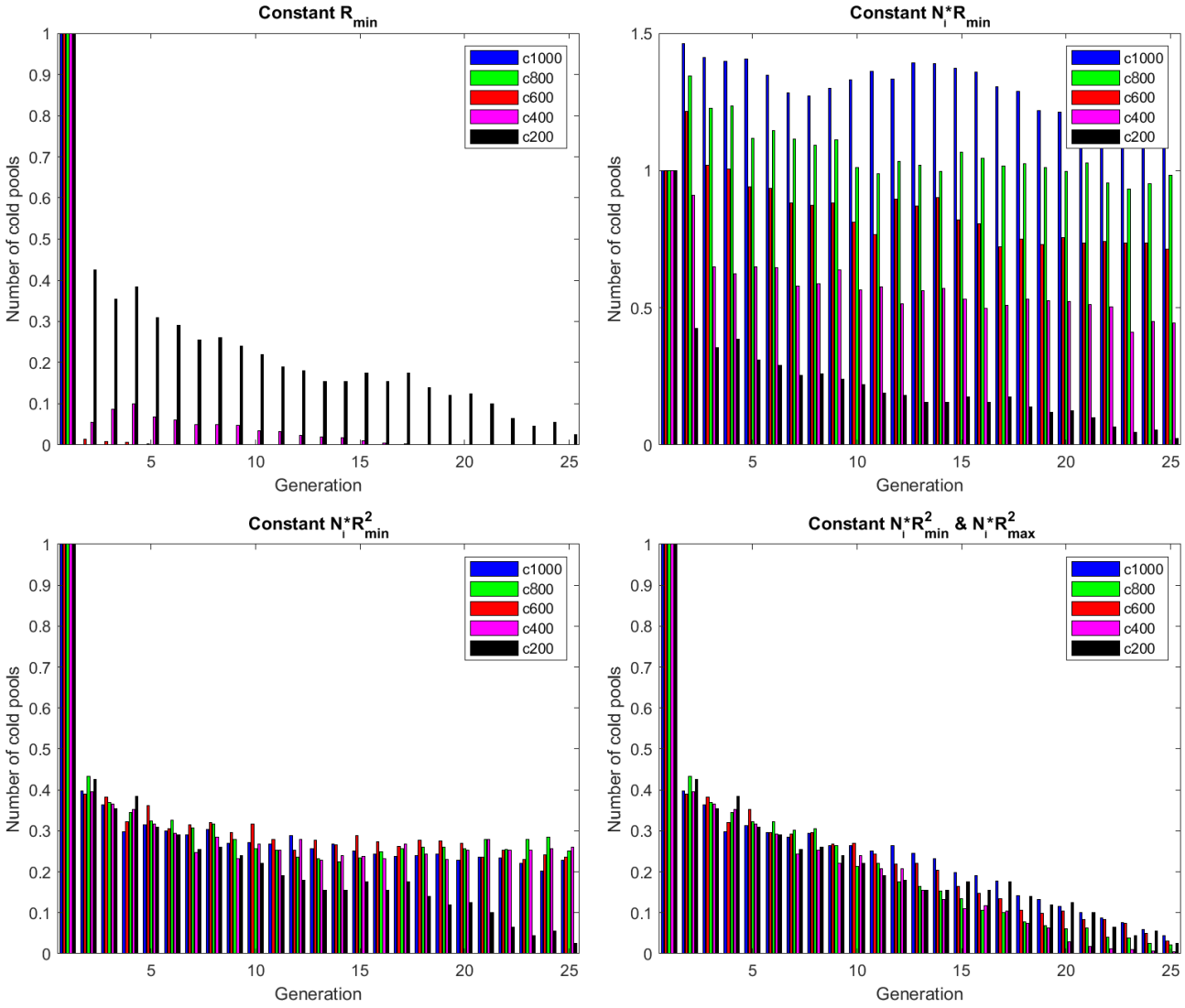


Figure 35: A set of graphs showing the interplay between R_{min}, R_{max} and the initial number of cold pools (called N_I). The first axis shows generation number, going from 0 to 25 and the second axis is showing the number of cold pools normalized by dividing with initial number of cold pools. The blue, green, red, pink and black bars represent the simulations starting with 1000, 800, 600, 400 or 200 initial cold pools respectively. In the first three graphs R_{max} is held constant, whilst in the first graph (top left) R_{min} is held constant, in the second graph (top right) $N_I R_{min}$ is held constant, in the third graph (bottom left) $N_I R_{min}^2$ is held constant and in the last (bottom right) $N_I R_{max}^2$ is held constant as well.

I began by ignoring R_{max} and focusing on R_{min} (that is to say R_{max} was held constant in the first three graphs). It is seen that by keeping both $N_I R_{min}^2$ and $N_I R_{max}^2$ constant the picture looks the same regardless. Therefore it should never be necessary to vary both at the same time and one could

always just vary one of these. That holding these constant creates a similar picture regardless of initial number or R_{min} and R_{max} in isolation, is because this keeps the density relative to the size restrictions (measured in area) constant. By changing the size restrictions (R_{min} and R_{max}) we change the number of cold pools that one can fit into the domain without overlap, whilst changing the initial number of cold pools alters the density within the domain, and when we increase the density we need to lower the area that these occupy, for it to remain unchanged. It could have been guessed from the beginning, but now we've confirmed it too.

The next set of constants I want to explore are the ones related to the noise model, namely the constants that sets the limits for the interval of random shifts in space and random shifts in time. The constants are referred to as RNSM and RNTM in the Matlab script, where RNSM stands for 'random noise space multiplier' and RNTM similarly abbreviated but with the 't' being time. As described in section 2.2.3, the spatial multiplier act on a term such as $(-1 + 2\text{rand})$, which gives a random number in the interval $[-1, 1]$, so that the multiplier directly sets the limits of this interval, whereas the temporal spatial multiplier act on a term such as $rand$, which gives a random number in the interval $[0, 1]$ (we don't allow for cold pools to be seeded before collisions). When using this model there should of course be an upper limit to how random I would allow the system to get, because at some point I would expect the result to no longer be sensible if everything is just randomized. But to get a better understanding of how strongly the system of cold pools reacts to variations in these factors, I will be running the simulations for different values of RNSM and RNTM (whilst keeping the other equal to zero), for a number of seeds and then looking at a graph over the number of cold pools versus the generations. In the fig. 36 below, you see the graphs displaying the results.

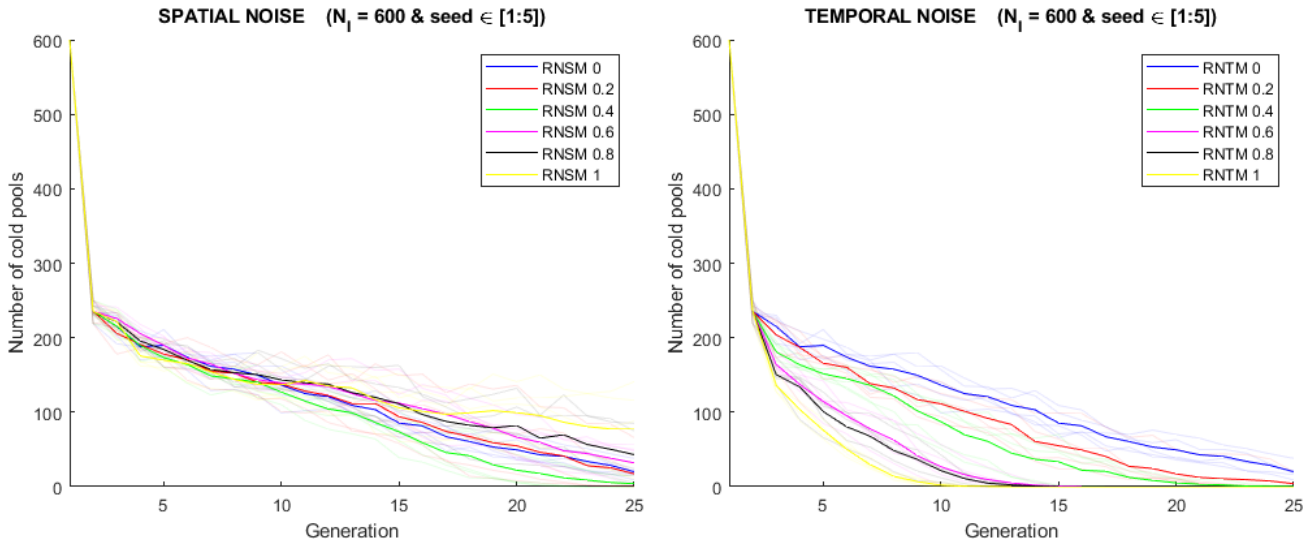


Figure 36: A set of graphs showing how the noise model reacts to variations in the spatial and temporal noise, called RNSM and RNTM respectively. Both has initial cold pool number $N_I = 600$ and every value of 'random noise multiplier' (from 0 to 1) is repeated with 5 different seeds. The more heavily coloured lines are the averages, while the more translucent lines are the results from the various different seeds (keeping the same color coding). All data was collected using the noise variant of the two cold pool collision model. In the first graph (left) you see variations in spatial noise and in the second graph (right) you see variations in temporal noise.

From the graphs in fig. 36 it looks as if larger spatial noise prolongs the lifetime of the system of cold pools, while it the opposite seems to hold true for temporal noise. The result seen in the graph for the temporal noise, can be understood by considering the affect that increasing the time shift can have. By increasing the interval from which we select a time shift randomly, we are likely to be increasing

the difference between when cold pools of one generation is seeded and this directly relates to their relative sizes. When one cold pool gets much larger than the other cold pools of that generation and takes up a larger portion of the domain, it can block the other cold pools from interacting, as illustrated in fig. 37.

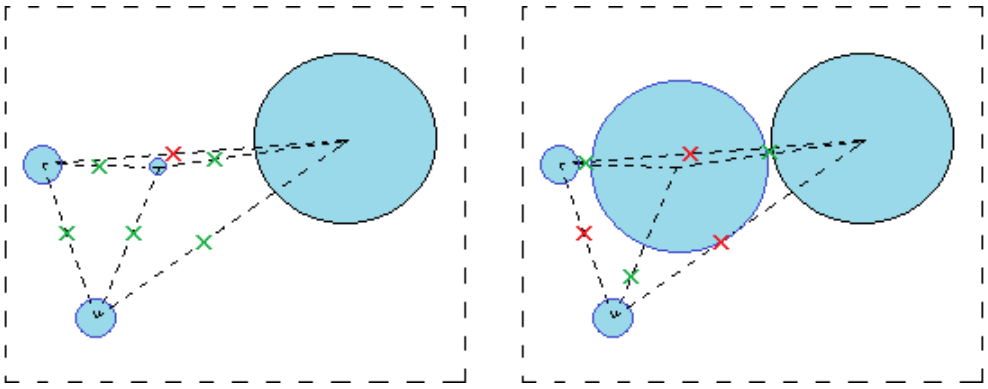


Figure 37: An illustration of the principle of blocking when relative sizes are increased. Both pictures are framed with a stippled line, indicating that these are not the borders of the domain, but just a smaller area within the total domain - so one should not consider periodicity here. On the left you see four cold pools depicted with stippled lines connecting their centres. On the half way points between these lines, measured from the rims of the cold pools, are some red and green crosses. The red crosses denotes a failed collision attempt and green a successful collision - failed because the fronts were inactive when colliding. To the right there's an illustration of how this may look with temporal noise. In this case fewer collisions succeed in seeding new cold pools. Note that the cold pool in the center was not seeded earlier than the picture on the left, but rather some of the other cold pools were delayed.

Another thing that can happen by delaying the seeding of cold pools, is that they can potentially be seeded inside other cold pools of the same generation. In this case, they are completely blocked from interacting with any cold pools of that generation, this is illustrated in fig. 38.

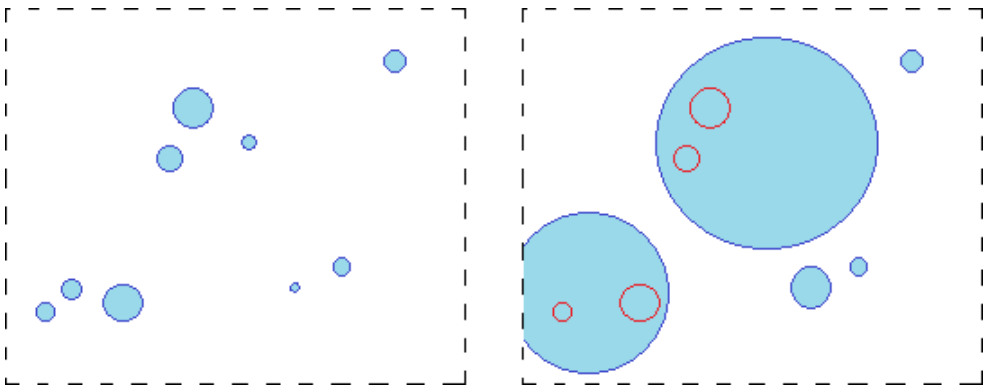


Figure 38: An illustration of cold pools being seeded inside already existing cold pools of the same generation. Both pictures are framed with a stippled line, indicating that these are not the borders of the domain, but just a smaller area within the total domain. On the left is depicted a bunch of cold pools. On the right is an example of how it might look with RNTM > 0. Some of the cold pools have been swallowed by a sibling and its wave front is now inactive, denoted by the red color.

The result seen in the graph of the spatial noise, is a little harder to explain and it's not as drastic a result. The largest value for RNSM gave, on average, the longest lifetime, but the shortest lifetime

was RSNM equal to 0.4 and not 0, also the difference between these are not as large as the difference between the longest and shortest lifetime in the graph for temporal noise. In order from longest to shortest lifetime, we had RNSM equal to $1, 0.8, 0.6, 0, 0.2$ and 0.4, which may suggest that the lifetime is not as strongly correlated to this value. However, the way in which this could make a difference, is by separating two cold pools that are closer than $2R_{min}$, such that they can interact or by moving two cold pools closer together, that was originally separated by more than $2R_{max}$, such that they can also interact. The opposite could also be true though, that we separate cold pools too much or pack them too closely together, and the process by which this happens is random, which would maybe also suggest the order of RNSM from longest to shortest lifetime was also random.

The last set of factors that I want to look at are the ones related to making the process stochastic in the scaling model and the memory field model. I again want to tune these constants, to see how the models react to changes in these factors and to get an idea of what a reasonable value of these may be for the upcoming sections. Starting with the scaling model, the factor that I will be tuning is in the script referred to as 'ScaleC' (named such for being a scaling constant) and as described in section 2.2.2 it shows up as a conditional statement in the form $L \leq \frac{ScaleC}{R_1 R_2}$ with L taking a random value in the interval $[0, 1]$. A low ScaleC would result in the fraction being small and this would make the odds of the condition being satisfied low as well. To get a sense of how sensitive it is to change, I use the scaling version of the two cold pool collision model and vary the factor and then look at the lifetime of the system (number of cold pools versus generations). I again repeat the process with a set of seeds to minimize the effect of variations. The resulting graph can be seen in fig. 39.

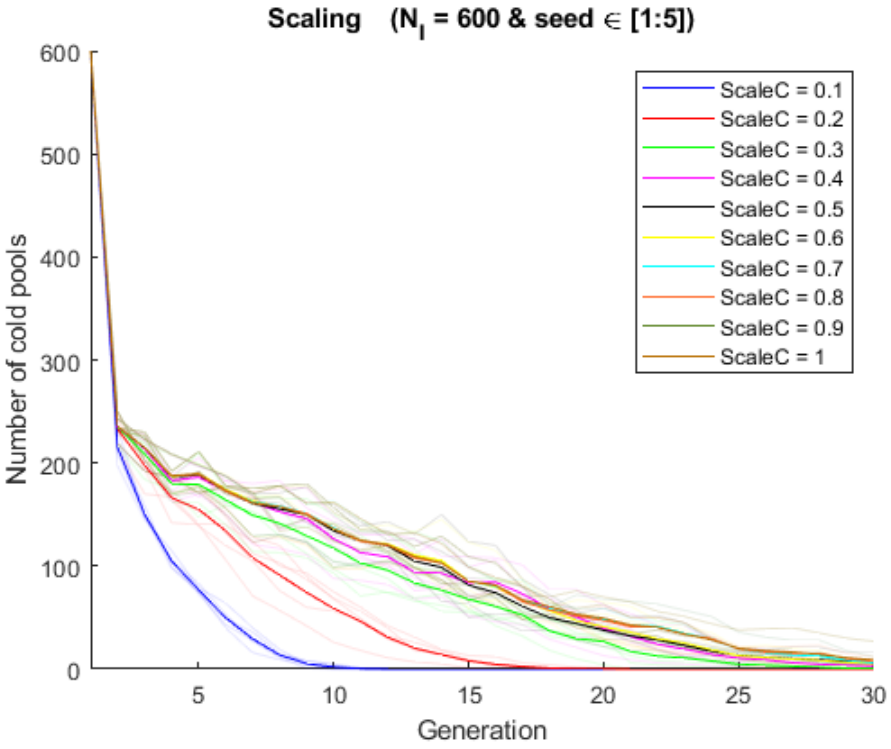


Figure 39: A graph displaying the results of simulating the scaling variant of the two cold pool collision model for various 'ScaleC' factors. Each run was initiated with 600 cold pools and for each value of 'ScaleC' i ran 5 different seeds. The size restricting factors were set to $R_{min} = 0.29$ and $R_{max} = 1.15$. The translucent lines are the simulation results of the different seeds and the strongly coloured lines are the averages (same color coding).

In the figure it can be seen, as predicted, that the lower values of ScaleC result in systems dying out the quickest - the reason for this, was given in the text right above. For a value of 'ScaleC' larger than R_{max}^2, the condition might as well not be there, as the fraction $\frac{ScaleC}{R_1 R_2}$ would be larger than 1 for any set of numbers in $\{(R_1, R_2) \mid (R_1, R_2) \in \mathbb{R}_{\geq 0} \wedge (R_1, R_2) \leq R_{max}\}$, and if either are larger than R_{max} there's a separate condition denying the collision. However, the value of ScaleC for which $\frac{ScaleC}{R_{max}^2} = 1$ is $ScaleC = 1.33$, which we did not reach, but all simulations with $ScaleC > 0.4$ had similar lifetimes, hence most collisions must happen well before R_{max}. One could perhaps solve $\frac{C}{x^2} = 1$ with C being the first ScaleC that is in the clump of simulations in fig. 39, to get a sense of the interval in which collisions generally occur.
The last factor that I want to look at is the factor labelled 'c' in the memory field model. As described in section 2.2.4 it too works as a way to make the process stochastic, by having the probability of a collision being successful depend on the dryness of the collision site. The dryness was related to the youngest cold pool, from the previous generation, that already occupies the space and the probability is related to these by $cD1$ with $D1$ being the size/age of the cold pool as this was related to the dryness. The condition is satisfied if a random number in the interval $[0, 1]$ is smaller than or equal to $cD1$, therefore $cD1$ can be interpreted as the probability that the condition is satisfied. For large enough c the condition will always be satisfied and at which point it becomes redundant, but to get a feeling for when this happens and how sensitive it is to changes in c, I again simulate the model. This is done similarly to before, where I use the memory field version of the two cold pool collision model, with an initial number of cold pools equal to 600, varying the value c and repeating each for a set of seeds, the result can be seen in fig. 40.

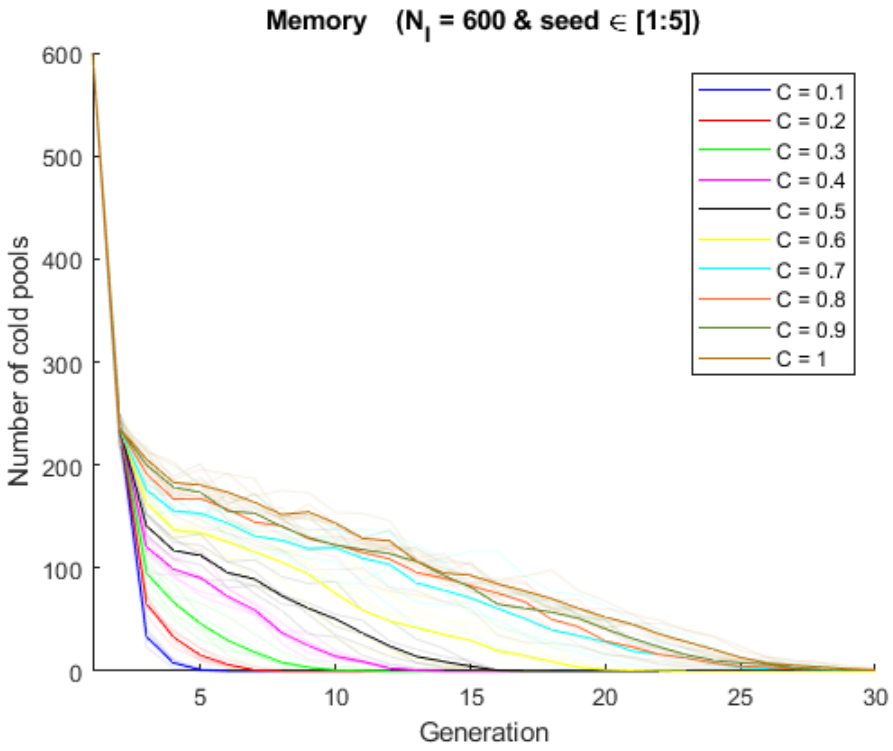


Figure 40: A graph displaying the results of simulating the memory field variant of the two cold pool collision model for various values of c. Each run was initiated with 600 cold pools and for each value of c i ran 5 different seeds. The size restricting factors were set to $R_{min} = 0.29$ and $R_{max} = 1.15$. The translucent lines are the simulation results of the different seeds and the strongly coloured lines are the averages (same color coding).

From the graph it can be seen that there is a strong correlation between the value of c and its lifetime, as expected. From the first to the second generation, there's a large drop in number of cold pools (similar to the previous simulations), this is due to the value of R_{min}. If the initial number of cold pools is large compared to R_{min}, they will be too densely packed to begin with and many cold pools will die off as a result of having to reach a minimum radii before being able to seed new cold pools upon collisions. After the initial drop, they die off one by one, with the lowest value of c dying off the quickest and the highest value of c the slowest. At one point we see diminishing returns by increasing c. As can be seen in fig. 40, the result for $c = 0.7$ up to $c = 1$ is quite similar and this is because at some point $cD1$ will be close to 1 for any value of $D1$. The value that c needs to take, for this to happen, depends on the density of cold pools. This is because $D1$ is related to cold pool sizes and if they're more densely packed it will typically be smaller and if it is more lightly packed it will be larger. The initial number of cold pools sets the density to begin with and thus impacts the value that c must take, but if R_{min} is large then the system will naturally become more lightly packed, making $D1$ larger and impacting the value that c must take in the opposite direction. Therefore the value that c must take, to make $cD1 \geq 1$ for any value of $D1$ would necessarily depend on the initial number of cold pools and R_{min}.

3.2 Comparing the models: cold pool numbers vs. generations

One of the reasons for making these different variations of the circle model, was to gain an understanding of how these effects may affect the system or how strongly they affect the system. It might for example seem obvious that making the process stochastic, such as done in the scaling model, would decrease the number of successful collisions and thereby also decrease the lifetime of the isolated system, but it may not be as obvious how this affects the spatial distribution or just how strongly the lifetime of the system reacts to this. It is also not as obvious what will happen with the diurnal cycle model or the persistent model, but this is what we now aim find out. There are mainly to things to look at, the first being how the number of cold pools develop with time (generations) and secondly how the distribution develops with time. As the title of the section suggests, I will be looking at the first of these two. The approach will be similar to the previous section, where I run the simulations for a number of seeds, and use the matrix which was introduced in section 2.1.1 to read off how the number of cold pools changes with time. The result obtained from running the simulations for the various two cold pool collision models can be seen in fig. 41.

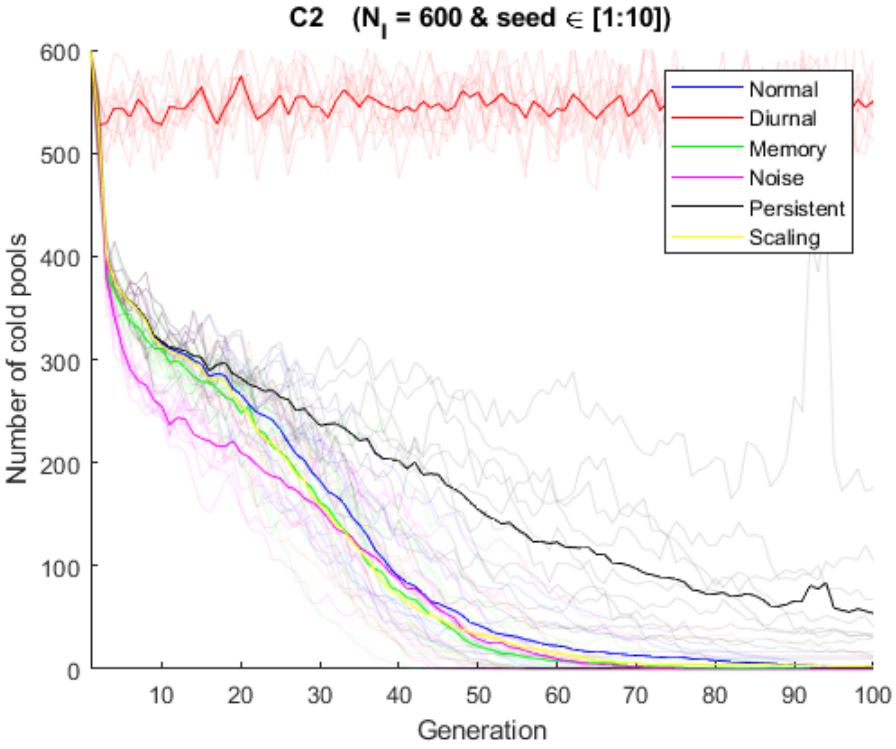


Figure 41: A graph displaying the results of simulating the various two cold pool collision models, all initiated with 600 cold pools, continued for 100 generations and repeated over 10 seeds. The size restricting factors were set to $R_{min} = 0.21$ and $R_{max} = 1.16$, the model specific constants were set to $c = 1.4$ (Memory), ScaleC $= 0.4$ (Scaling) and RNSM $= 0.5$ and RNTM $= 0.2$ (Noise). The translucent lines are results of the different seeds and the strongly coloured lines are the averages over all 10 seeds.

I used a slightly lower R_{min} than in the previous section, to counter the fact that they were packed too densely relative to R_{min} which resulted in the number of cold pools dropping to less than half the initial value, going from generation 1 to generation 2. Because R_{min} was decreased slightly, I increased the factor c in the memory field version of the circle model, to adjust for this - as talked about in the previous section. Despite the curve being less steep, there is still a significant drop in the cold pool number from the first generation to the third and fourth generation. This is again related to R_{min} and a lower value could have been selected to counter act this, but a too low value of R_{min} would have caused the number of cold pools to blow up (as explained in section 2.1.1) and with this value of R_{min} it did not drop below 500 going from the first to the second generation (note the Diurnal model stays above 500). However, this is not detrimental to what we are trying to achieve, namely to compare the different models. As long as the parameters are comparable, which they should be since R_{min}, R_{max} and the set of seeds were the same across all simulations, then the results should be comparable.
Now, from the graph it is seen that the memory and the scaling model both dies off slightly quicker than the normal two cold pool collision model, but only slightly. This is not too surprising, when the effect introduced into these models are ones that limits the chances of collisions being successful. That their lifetimes are only slightly less than that of the original two cold pool circle model, can perhaps be attributed to the value of c and $ScaleC$ selected, since lower values would have caused the systems to die quicker. Even though the regular model as well as the memory, noise and scaling versions of the model, dies off between the 80th and 90th generation, there are some runs that doesn't die off. In some cases, you have cold pools forming lines (similar to squall lines) that span the domain, and when this happens the system stays active much longer - infinitely lasting if all cold pools that remain,

form a perfect line across the domain, see fig. 42 for an illustration of this concept. This state is a direct product of having periodic boundary conditions and it could then be put into question, how one should perceive these. The point of having periodic boundary conditions, is to mimic infinitely large systems, but it could be argued that this is a case of the periodicity being 'abused'. The minimal number required to form this chain is the side length (shortest path) divided by $2R_{max}$ rounded up to the nearest integer, this is the number of maximally large cold pools that can tightly fit in a chain across the shortest path, which in this case is 5 and the largest value of cold pools that can form a chain is the length of the diagonal (longest path) divided by $2R_{min}$ rounded down, which is 33 in this case.

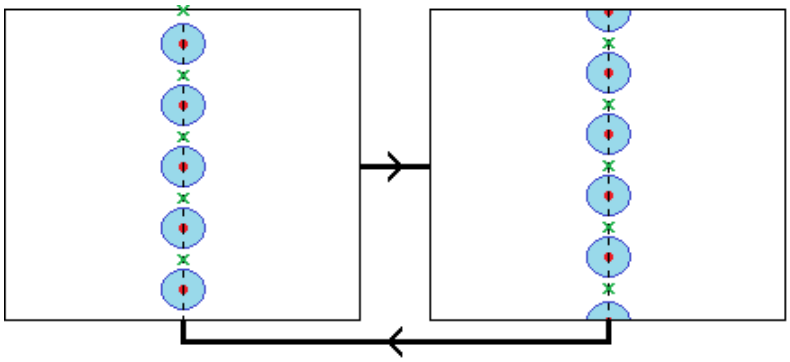


Figure 42: An illustration of the concept of chains forming in the two cold pool circle model. The chain will interchange between the left and the right, going from one generation to another.

Moving on, from the graph in fig. 41 we see that the noise model drops off quickly to begin with, but then slows down and finally dies off around the same time as the memory and scaling variant. The large initial drop of all models are related to the density of cold pools, the memory variant however may swallow nearby cold pools, so we can assume that it reacts more strongly to being closely packed - hence the larger initial drop.
The persistent variant of the circle model is ironically only the second most persistent model. It outlives the standard model (as well as 'scaling', 'noise' and 'memory') and it does so for two reasons. The first reason is that by limiting the amount that cold pools can grow, you decrease cold pools ability to take up a large portion of the domain and block other cold pools, as was illustrated in fig. 37. The second reason is that cold pool fronts only become inactive after collisions, where in the other models they become inactive once they grow larger than R_{max}. So as opposed to the scaling, noise and memory variant, this variant has an effect that affects the lifetime positively and it shows in fig. 41.
The longest living variant by far, is the diurnal cycle model. It doesn't seem to die off, but rather oscillate around the initial drop that occurs due to the density of packaging and R_{min} as previously discussed. Since every cold pool belonging to the same generation is initiated at the same time, they all start growing from zero and it is as if the system restarts itself - in the first generation, every cold pool starts with a radii equal to zero and in the diurnal model every subsequent generation does too. One interpretation of the result could be that unlike the first generation, every subsequent generation will have a lower density and thus a higher R_{min} relative to density, and this could result in no further drops - the density of cold pool packing evens out with the value selected for R_{min}. Another way to interpret this result, could be that all cold pools too closely packed will die off in the first generation, due to R_{min}, in the subsequent generation only the cold pools far enough apart remain and in every subsequent generation they keep this distance, making the system similar to the chain previously described, but instead of a chain it is the entire square domain. In the following section, we will be looking at the distribution of cold pools on the domain, and we will perhaps get some clarity in regards

to this. Notice that the diurnal model is the model with the longest lifetime, for the same reason that the noise model is one of the shortest living models. Starting all cold pools in a generation at the same time makes them all equally large at any given time and so no one or two cold pools take up the entire domain, killing off potential new cold pools.

Moving on, I ran the same simulations with the three cold pool collision models, with the same number of initial cold pools, same parameter values and same number of seeds (and the same seed values of course), such that the setup is the exact same and the results are as comparable as possible. The results can be seen in fig. 43.

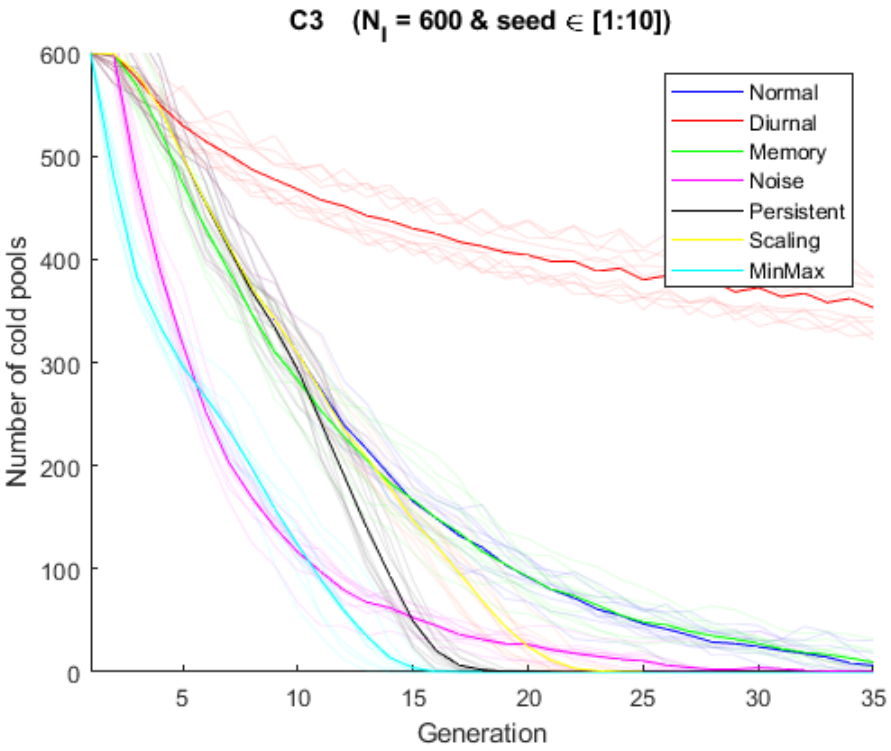


Figure 43: A graph displaying the results of simulating the various three cold pool collision models, all initiated with 600 cold pools, continued for 100 generations and repeated over 10 seeds. The model specific constants were set to $R_{min} = 0.21$ and $R_{max} = 1.16$ (MinMax), $c = 1.4$ (Memory), ScaleC $= 0.4$ (Scaling) and RNSM $= 0.5$ and RNTM $= 0.2$ (Noise). The translucent lines are results of the different seeds and the strongly coloured lines are the averages over all 10 seeds.

From the graph above it can be seen that the variant of the three cold pool collision model with the shortest lifetime is the one with size restrictions (minmax) and it dies off at around generation 15 (with these values for R_{min}, R_{max} and initial number of cold pools). Having restrictions on cold pool activity, will naturally lead to less successful collisions, but it's only in the three cold pool variant that it is displayed just how big a difference this makes (because every two cold pool collision models have size restrictions). In the three cold pool collision variant, there's an extra cold pool that can potentially ruin the successfulness of the collision, by either being too large or too small, so in that sense it may be more sensitive to this restriction - unlike the two cold pool collision model, it doesn't blow up for too small R_{min}.
The variant with the second shortest lifetime is the persistent variant, which had the second longest lifetime in the two cold pool collision model. This is most likely do to the fact that the normal model

(and every other variant apart from minmax) doesn't have any size restrictions, so even though it has the benefit of cold pools not becoming infinitely large and taking up the entire domain, blocking potential collisions as previously discussed, it also has the draw back that cold pools that are too close together or too far apart can not interact, and this seems to outweigh the positive.
The variant with spatial and temporal noise behaves similarly to what was seen in fig. 41 for the two cold pool collision model, where it drops quite significantly to start with and then becomes more smooth before finally going to zero. I would assume it took this form, for the same reason as in the case of the two cold pool version, with the temporal noise or delay of cold pool seedings being the main reason. There are two things that can affect the result negatively, the first being that when cold pools are delayed the difference between cold pool sizes are increased, which can cause the larger cold pools to block the smaller cold pools from expanding and colliding with other cold pools, and the second being that closely packed cold pools can be swallowed by its neighbours and become completely inactive. Both of these affect the result negatively and most strongly when there are more cold pools or when they're more densely packed, which would explain the steeper slope in the beginning of the simulation.
The scaling variant and the memory variant, both act in a way that prohibits some collisions from successfully seeding new cold pools, so we expect to see something similar to what was seen in the two cold pool case. The scaling model acts as expected, for the reasons previously stated, but the memory variant virtually lies on top of the regular or normal cold pool model. To get a result that differs from the normal three cold pool circle model, we would apparently need a lower value for c, which is perhaps suggesting that the cold pools are on average larger when they collide in the three cold pool version of the circle model. This is because the term that defines the probability that a collision successfully seeds a new cold pool (if all other requirements are met) is of the form $cD1$ with $D1$ being the size of the youngest cold pool from the previous generation occupying the space where the cold pools collided. For the probability to be larger, to seed new cold pools in the three cold pool circle model, with the same value for c, would require $D1$ to be larger and for $D1$ to be larger, it would require that collisions are spaced further apart in time or equivalently that cold pools are larger when colliding.
The normal version of the three cold pool circle model is the second most persistent model, with the memory version almost having as long a life expectancy (with this particular value for c). Most of the variants act in a way that prohibits collisions from successfully seeding new cold pools, so it is no surprise that it once again is among the longest living versions. The only version to outlive the regular three cold pool circle model is the diurnal variant, but unlike the two cold pool version, this one slowly but steadily decreases. At the 100th generation, there were on average a little over 300 cold pools remaining, which is still substantially more than any other three cold pool circle model, but quite a lot less than its two cold pool counterpart. It is a general trend that the three cold pool circle models dies off quicker than their two cold pool counterparts, in fact any two cold pool circle model outlives every three cold pool circle model (except for the diurnal variant). We could compare k-combinations of a set of N cold pools, with $k \in \{2, 3\}$, as so

$$\frac{N(N-1)}{2} = \frac{N(N-1)(N-2)}{6} \qquad \text{with solutions } \; N \in \{0, 1, 5\}. \tag{23}$$

For any set of more than five cold pools the RHS of eq. (23) will be larger, but we are operating with sets much larger than five, yet we have that the three cold pool models have a relatively low life expectancy (compared to the two cold pool models). There are several possible explanations for this, one being that there's an extra condition to be satisfied in the three cold pool model, namely the triangle condition. This condition could possible eliminate a lot of potential new cold pools. Another

explanation could be that, even though there are more combinations of three cold pools than two, having to have all three cold pools reach the potential collision site is more demanding than having to have two cold pools reach a collision site. It is more likely that one cold pool will be blocked from reaching, when more cold pools are involved. This is despite the two cold pool variants all having size restrictions too (both R_{min} and R_{max}).
One last thing to note, before we proceed and look at their spatial distribution, is that while every two cold pool circle model had an initial drop in cold pool numbers from the first generation to the second, the three cold pool variant with size restrictions (minmax) did not seem to have this initial drop. Recall that my explanation for this initial drop, was because of R_{min} and the initial density of cold pools. This discrepancy could be explained in two ways, the first being that cold pools seem to be generally larger at collisions for the three cold pool variant, this was concluded earlier in this section when addressing the result of the memory variant of the three cold pool circle model. These larger sizes will most likely lower the importance of R_{min} and the drop will be smaller going from the first to the second generation. The second possible explanation for this is that the three cold pool collision models die off quickly, from the beginning until the end, and so the initial drop may be similar to the drop in cold pool numbers from any generation to the next.

3.3 Clustering & self-aggregation

In this section I want to look at how the cold pools distribute themselves on the domain, to see if there is a tendency for the cold pools to cluster and self-aggregate. I will test for this in two different ways, the first test will be to calculate the distance to every cold pools absolute nearest neighbour, take the average and look at how this changes with generations. The idea is that if this decreases as a function of time (or generations), then cold pools within one generation is more clustered than previous generations. The second test to see how cold pools distribute themselves on the domain, will be to place a grid on the domain, count the number of cold pools within each grid space and look at the distribution. The idea being that if it is more likely to find cold pools in grid spaces with many more cold pools, then you again have some clustering or self-aggregation. I will go into a bit more detail, of how I compute these in the following subsections and then compute and display the results.

NOTE: To keep these tests comparable with the previous results, I use the same data that I collected previously.

3.3.1 The first test: average distance to nearest neighbour

I will begin with a more detailed description of how the test was conducted and how the subsequent plots were produced. So as previously mentioned, I use the data that I collected earlier, to make everything comparable. I start by loading the data from a specific variant into a Matlab script, this could for example be the ten different seeds of the persistent variant of the two cold pool circle model, and then once these are loaded, I go through them one by one. Each set of data describes all the different cold pools that were seeded in that run and I want to go through all these cold pools, generation by generation. I take the first generation, I select a cold pool, I shift it to the center of the domain and every other cold pool will be shifted too (with periodic boundary conditions obtained using modulo). Once it is in the center, I use the nearest neighbour function that I wrote (described in section 2.1.1) to find the nearest neighbour in each quadrant. The nearest of these is selected and the distance between the centres of these are then calculated. The distance is stored, along with the generation that we're currently looking at. Then the process repeats until we have gone through all ten simulation runs,

every 100 generations and every cold pool within each generation. Once all the data is collected, I find the average value of the distance between cold pools and their absolute nearest neighbour for each generation - this average takes data from all ten simulations. The procedure is very similar to how the different models proceeded, with going through the options of cold pools, moving them periodically to the center, using the nearest neighbour function I wrote and then computing (in this case) the distance. Having done this for all the different variants of the two and three cold pool circle models, I want to create some sort of control that I can compare the results to. The control will be the average distance between randomly placed cold pools and their absolute nearest neighbour. I can however not make one default control, as this average distance to nearest neighbour will depend on the number of cold pools in that generation. Therefore I will make separate controls for each two and three cold pool variant. Each control will have a number of cold pools within each generation, from the first to the last, that is equal to the average number of cold pools in that generation of the variant that it is a control to. After using the data available and calculating the average number of cold pools within each generation, I create twenty sets of data containing the information about the randomly placed cold pools, with the number of cold pools in each generation being the before mentioned average. Now I can compute the average distance to the absolute nearest neighbour in the same way as before. Having calculated the average distance between nearest neighbours of the different variants and of their corresponding controls, I get the result that can be seen in fig. 44 below.

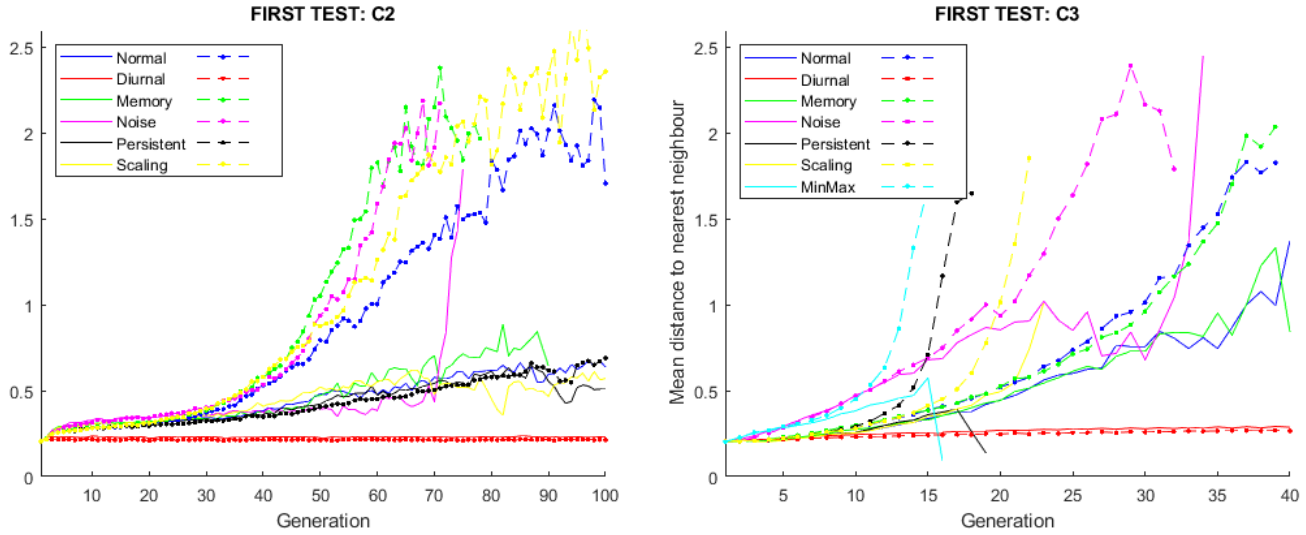


Figure 44: Two graphs displaying the average distance to the absolute nearest neighbour versus generation, for all the different variants of the two and three cold pool circle models. On the left are the two cold pool models and on the right the three cold pool models. The full lines are from the model simulations and the dashed lines are the random controls with the same color coding. For reference the distance 1 is 1/10 of the side lengths of the domain, which is a 10 by 10 square.

I will start by focusing on the results seen in the left graph in fig. 44, namely the results of the two cold pool circle models. For the normal, memory and scaling models the average distance to nearest neighbour increase slightly, from around 0.2 to 0.5, but stays well below their respective controls. That they're well below the random controls, could indicate that they're more likely to cluster or that cold pools that are situated more closely together are more likely to survive. This might seem like a natural conclusion to having a maximal radii, as cold pools with no nearby neighbours will die off, reducing the number of cold pools with a larger distance to a nearest neighbour but leaving the more densely packed areas intact. However, the average did not decrease going from the first to the last generation. One could have assumed that it would have, based on the previous statement. That the average distance increases could be the result of several things, like having a minimal radii, cold pools

of later generations being larger in general (taking up more space) or just that the overall number of cold pools are decreasing. Firstly the minimal radii acts in a way that prohibits cold pools too closely together from interacting and this thins out the number of cold pools in more densely packed areas (remember the initial drop in fig. 36 as a result of R_{min}), this could increase the distance to ones nearest neighbour. Secondly all cold pools start with an initial radii of zero, meaning they are all equally large at any given time in generation one, but later generations will have cold pools that are larger relative to other cold pools in that generation. These larger cold pools will take up more space, which could perhaps push cold pools further away, and this could in turn increase the distance to nearest neighbour for themselves at the very least. Thirdly and lastly the overall number of cold pools decreasing, from generation to generation, could just thin out the domain, making it more likely that any cold pools nearest neighbour would be further and further away. Regardless of the tendency for the distance to increase with time, we see that they are much closer than cold pools that are randomly placed on the domain, so there seems to be some tendency to cluster.
The noise variant of the two cold pool circle model behaves similarly to the before mentioned variants, until around generation 70th where the average distance to nearest neighbour suddenly increases by a lot. There may be two reasons for this, related to time delays and spatial translations. The time delay will act in a way that increases relative cold pool sizes, which as previously discussed could perhaps increase the distance between cold pools, but even more importantly the time delays act in a way that makes cold pools swallow nearby neighbours, as illustrated in fig. 38. Once a nearby neighbour has been swallowed, it will no longer be seeding new cold pools, and this would in turn thin out the area, making the distances to nearest neighbour for subsequent generations larger. The spatial translations will relocate cold pools randomly in an area around the collision site that seeded it, this relocation can put cold pools closer to others or further away, but when the number of cold pools decrease it seems more likely that it would move them further apart. Imagine the domain being covered with cold pools, moving a cold pool in either direction would move it towards some other cold pool, but if the number of cold pools are scarce this relocation could be to an area with no nearby cold pools. The impact that the spatial noise can have is strongest when there are fewer cold pools, which could explain why the average distance to nearest neighbour suddenly increases around generation 70th and not earlier, because at this point there were only very few cold pools left (see fig. 41).
The persistent variant of the two cold pool circle model has an average distance to nearest neighbour that closely follows the random control. This could just be a numbers game, since the persistent model has significantly more cold pools at the 100th generation than the previously mentioned variants. More cold pools, would mean a more densely packed domain, especially for the random control. In fact if we refer to fig. 44 we see that the other variants also follow the random control until around generation 30 to 40, whereas the persistent variant follows the random control until around the 80 to 90th generation, coincidentally if we compare these results to fig. 41 we see that the number of cold pools in the persistent model around generation 80 to 90 is similar to the number of cold pools in the previously discussed variants between generation 30 to 40. This could suggest that the difference between the random control and the data, is mainly found when there are fewer cold pools left. However, I want to note that in the persistent simulations there was one simulation where the number of cold pools briefly soared, can be seen in fig. 41. This would affect the average number of cold pools at around generation 90, which would in turn affect the distance to nearest neighbour for the control in a way that lowers it - therefore this could partially be the reason for this result.
The diurnal model and the random control virtually lies on top of one another, from start till finish and keeps a constant value of around 0.2. This could further be an argument for the fact that the difference in the random control and the data is first seen when the number of cold pools have decreased enough. However, I think it might also indicate that this variant of the circle model is basically just random.

This is because we initiate the setup with randomly located cold pools and the subsequent generation is completely determined by this, but where every other cold pools has a reference to its past through time seeded, this variant has no such memory and this makes it such that every subsequent generation is virtually random too.
The three cold pool collision models display many of the same results, albeit a bit more chaotically. The normal, memory and scaling models all lie well below their controls. The diurnal model again follows the random control closely, from start till finish, however it does not keep a constant value. It changes from around 0.2 to 0.3, from generation 1 to generation 40 and it keeps growing slowly as the overall number of cold pools decrease. The noise model again has a sudden increase in distance to nearest neighbour, which would be explained in the same way as before, this time however the random control has a decrease, which I would assume was just a random fluctuation - had I used more than twenty random simulations, it would have increased more steadily with a decreasing number of cold pools. Lastly the minmax model and the persistent model, were the first to die off, as can be seen in fig. 43. This makes the random controls increase more rapidly than the other controls. The results from the data, of both variants, take a dive at the very end, when there are presumably very few cold pools left. I would assume that this is a result of the few remaining cold pools being situated very closely, but without being able to successfully seed new cold pools.

3.3.2 The second test: grid space and distribution

I will once again start with a more detailed description of how this 'second test' proceeds, so as to make sense of the results that I will be displaying. The test is again a way to try and see if there is a tendency for the cold pools to cluster and self-aggregate, and this particular test will be focusing on how the cold pools distribute themselves on the domain (as opposed to the first test that focused on how they distribute themselves relative to one another). The idea is that I place a square grid onto the domain and count the number of cold pool centres (belonging to the same generation) within each grid space, see fig. 45 for an illustration. Once I have gone through the different generations and counted the number of cold pools within each grid space, I will then compute the probability distribution of the number of cold pool centres in grid spaces. This again requires a control, to give the results a reference. In this case the control will be a binomial distribution, with probability $p = \frac{1}{\#\text{grid-spaces}}$ and trials n equal to the average number of cold pools within that generation. I want to note that this is only an approximate control, since the probability for a cold pool to be placed in a grid space with several other cold pools should probably be lower than the probability for it to be placed in a grid space with no other cold pools. Assuming the probability is independent of the number of cold pools already placed in a grid space, a random distribution should be binomial. The square grid that I place onto the domain, is such that there is 100 grid spaces (it is a 10 by 10 square grid), and the probability associated with the binomial distribution controls is thus 1%.

The code runs by first loading a set of data (collected earlier), for example the 10 seeds of the diurnal variant of the two cold pool circle model, and once the data is loaded I compute the average number of cold pools within each generation (this will be used with the binomial distribution control). I want to look at five generations from start till finish, equally separated, but I don't want to compute the distribution of the 90th or 100th generation of simulations that died off at around the 50th generation. Therefore I want to stop computing distributions at around the last generation with more than one cold pool in the average, thus the increment change in generations will be the number of the last generation divided by five and rounded down to nearest integer.

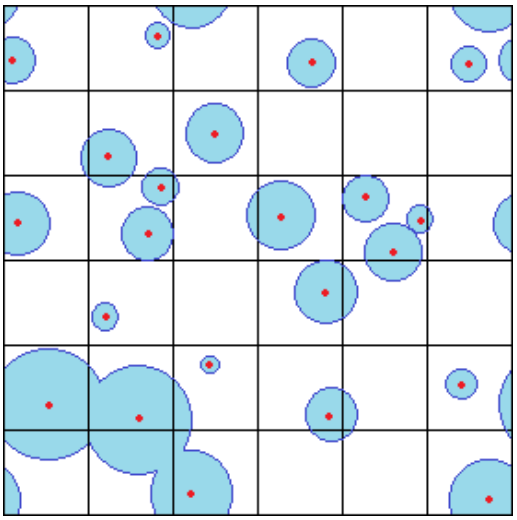


Figure 45: An illustration of the square grid being placed onto the domain.

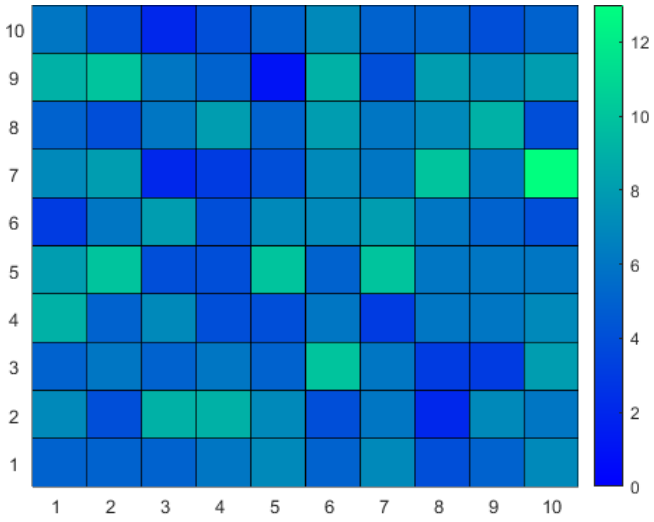


Figure 46: An illustration of how the data for a specific generation within a specific seed is stored.

With the set of generations that I want to look at, I begin by looping over it and during each iteration I loop over the number of seeds (ten) of the data. During each iteration of the loop over the different seeds, I load the data associated with this seed, which contains every generation, x and y positions of the cold pool centres and so on. I remove everything, other than the x and y positions associated with the current generation I'm interested in and then I use the function $floor$ on this data. What this function does, is it simply rounds values down to the nearest integer (including zero). Having done this, I can now more easily determine what grid space they belong to. A grid space in the lowest left corner in fig. 45 has an x-value of zero and a y-value of zero, moving a grid space to the right would increase x by one and moving up would increase y by one. To go through every grid space, counting the number of cold pools in each of these spaces, I use two subsequent loops over the possible values of x and y (zero to nine in steps of one), and at each iteration I count the number of cold pools and store the data. The data is stored in a third order tensor with the two first indices indicating the grid space and the third index indicating the data (or the seed) that we're currently computing and lastly the value that this entry takes is the number of cold pools within that grid space, a visualization of this can be seen in fig. 46. This then repeats for the next seed, until we have gone through every seed. Having gone through every grid space and every seed, we are ready to compute the distribution. I do this by looping from 0 to the largest value of cold pools in one grid space, counting the number of entries in the third order tensor previously mentioned equal to this, so for example counting the number of grid spaces with zero number of cold pools or the number of grid spaces with this maximum number of cold pools. Then I store this information in a vector and divide each entry by the sum of the values of the entries in this vector (to get the information in percentages). Now I have the distribution of a specific generation and I plot this, alongside a binomial distribution with $p = 0.01$ and a number of trials equal to the average number of cold pools within this specific generation of these specific data sets. This is repeated for the five generations that I picked, equally spaced from start to finish, producing five plots. Having done this for each cold pool variant of the two and three cold pool models, I produced the graphs seen in the following pages. The two cold pool circle models can be seen in fig. 47 and the three cold pool circle models can be seen in fig. 48.

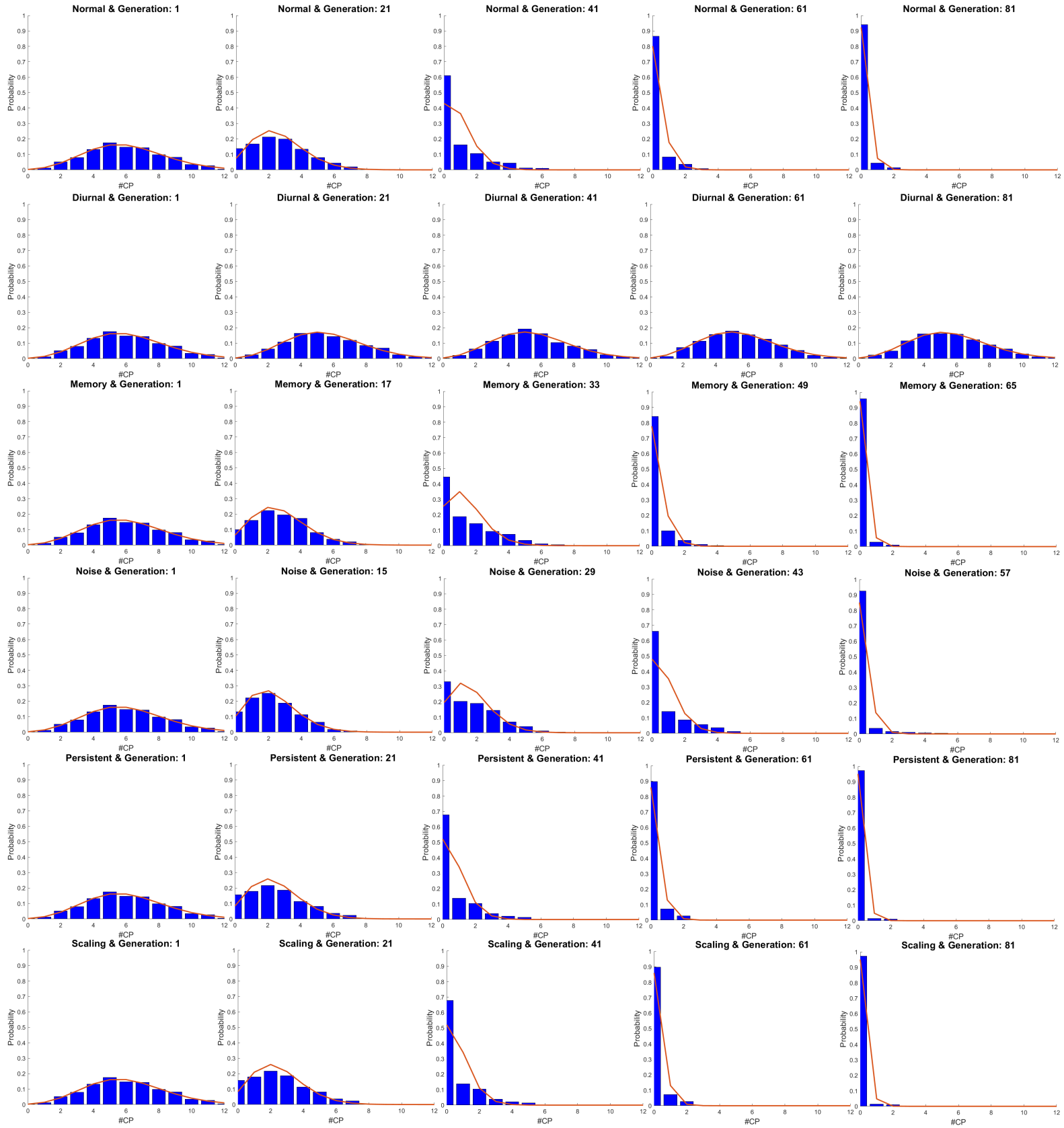


Figure 47: The two cold pool results of the second test. The order from top to bottom is Normal, Diurnal, Memory, Noise, Persistent and Scaling. The first axis is the number of cold pools in a grid space and the second axis is the probability for this to occur.

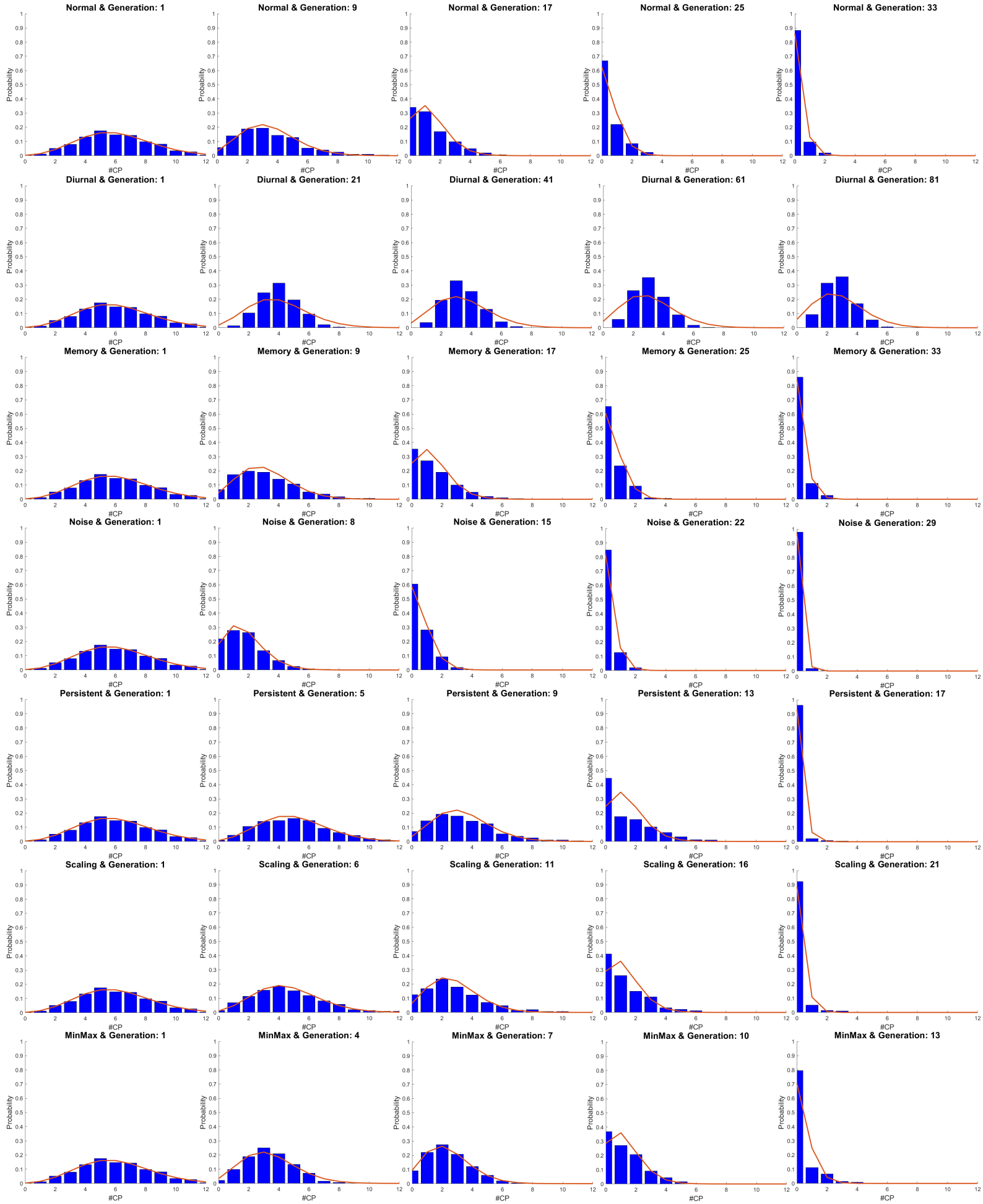


Figure 48: The three cold pool results of the second test. The order from top to bottom is Normal, Diurnal, Memory, Noise, Persistent, Scaling and MinMax.

The results seen in fig. 47 and fig. 48 shows similar results to the ones that we saw in the previous section. It shows that the two cold pool diurnal model is almost random, which is in agreement with the results found in the previous section. However, the three cold pool diurnal model, while also almost following the binomial distribution, shows a tendency to have fewer cases with more cold pools in one grid space for later generations (compared to the binomial distribution). This could be the result of random outliers, but it is hard to tell. For the remaining models, both two and three cold pool circle models, the results show that for later generations there is a larger tendency for clustering to happen, as in almost every third graph the number of grid spaces with zero cold pools is larger than the randomly placed cold pools with the binomial distribution, and thus the probability of finding grid spaces with more cold pools is necessarily higher too, as both distributions have to sum to one. This is also in agreement with the results in the previous section, but it again raises the question of why the cold pools are clustering and if it is just a numbers game.

4 Discussion & Conclusion

To summarize the results showed that the diurnal variant of the circle model (both two and three cold pool models), were the variant with the longest lifetime by far. The noise, scaling and memory models, as well as the three cold pool minmax model (the three cold pool variant with size restrictions), all acted in a way that decreased the overall lifetime of the system. The persistent model was the second most persistent variant of the two cold pool circle models, only outlived by the diurnal cycle model, and it prolonged the lifetime of all size restricting models (all two cold pool circle models and the minmax three cold pool circle model). The two cold pool models had a much longer life expectancy by default, despite $\frac{N(N-1)}{2} < \frac{N(N-1)(N-2)}{6}$ for any $N > 5$, which is a result of fewer restrictions. From the results regarding the distribution of cold pools, we found that the diurnal model was indistinguishable from the random controls, both in the first test where we computed the average distance to ones absolute nearest neighbour and in the second test where we placed a grid on the square domain, counted the number of cold pools in each grid space and calculated the frequency of these values occurring. In both tests, the controls were random, the first test used randomly generated data sets and the second tests used a binomial distribution. Hence the diurnal models did not display any tendency to cluster, since the random controls clustered just as much. All the other variants however displayed a tendency to cluster, which was apparent in the results from both the first and the second tests. Whether clustering or self-aggregation are intrinsic properties of the models or a natural result of the number of cold pools dwindling is harder to conclude. You can make the argument that more densely packed areas of cold pools have a larger life expectancy than areas with a lower density of cold pools, if the last cold pools to disappear are the ones making up these more densely packed areas, than one would expect a similar result to what was found. The average distance to nearest neighbour would be lower than the random controls and grid spaces with a higher number of cold pools would occur more frequently. If this is the case, the system doesn't progress into clusters, but rather the existing clusters that were randomly generated in the initial setup is what remains in the later generations.
Either way we were able to get a sense of the circle model and the impact of these added complexities in the different variants. Even if the question of self-aggregation remains inconclusive, it is clear that the biggest contributor to making the system of cold pools self sustaining is by having some type of diurnal cycle relation with precipitation events.

In the beginning of section 2.1 and in section 2.2 I talked about how the circle model was an idealization of the nature of cold pools and how the model was one in which cold pools acted in isolation,

but I also discussed how simplifications were not necessarily a bad thing and that they are common in all branches of physics, sometimes it is the best that we can do. However, when we are able to get closer to a more realistic picture, it seems sensible to try and do so. In that spirit I added several different complexities to the existing model and this gave us a sense of how these different aspects could affect the system of cold pools. Though the models more closely depict nature, they remain idealizations and the results should be considered as such. The cold pools still expand with equal and constant velocities (except for the persistent model, in which they grew to be R_{max} and stay this size), the cold pools remain perfectly circular throughout their lifetime, the seeding of new cold pools are always a direct result of the dynamics of other cold pools and each of these thermodynamical effects that I added are only added in isolation, where as it would have been more realistic if the model displayed all of these effects simultaneously. When I say that the result should be seen in the light of this being an idealization, it is not only in the sense that the system of cold pools could display different results if more complexities were added, but also the affect of these effects may change if they are added in combinations. In the introduction I referenced RKW theory, and how cold pools and wind shears could act in a way that promoted the occurrence of squall lines, but in RKW theory it is only in combination that they do so. Cold pools in isolation or wind shears in isolation, counteract the formation of squall lines, but together they increase the frequency of the formation of squall lines. While I have no reason to expect that the affect of adding temporal and spatial noise would drastically change by also having precipitation events happen with a diurnal cycle, I can not say that this wouldn't be the case, which is perhaps a good segue into an outlook.

5 Outlook

There are many things that could still be done with respect to this circle model and its many new variants. As alluded to in the previous section, one could make a unified model. One that inhabits every effect simultaneously. A model that has a history of prior generations of cold pools, in terms of a memory field, whilst having the success of cold pool collisions depend on their sizes and so on. I would assume that this could be done, by combining the changes that I made to my version of the regular cold pool model. We could even consider having both two and three cold pool collisions. This could perhaps be achieved, using the existing code for two and three cold pool collisions and go through the two and three cold pool collisions separately for each generation. While these things seems ready to be implemented, adding all these things would also increase the computational power needed to solve it within reasonable times. The persistent model was quite heavy to compute and that was after I went from an analytical solution to an approximate solution, because the analytical solution was a lot heavier to compute repeatedly. While on the subject of the computation becoming heavier, if all effects are added in a unified model or/and considering both two and three cold pool collisions, there are also things that could be done to improve the effectiveness of the code. For example the nearest neighbour function, that I used to reduce the number of combinations to go through to some smaller (but still reasonable number), could potentially be done in a smarter way. I divided the domain into four sections, with the cold pool in question in the very center and only considered a handful of the nearest cold pools in each of these quadrants. I could perhaps have split the domain into more sections, which could make the number of cold pools that I needed to consider in each decrease. Perhaps an altogether different way of considering nearest neighbours would be more effective, like the way Jan and Silas does in their code. Another thing that could be done, to make the computations faster, is by using the parpool function in Matlab, which starts a parallel pool of 'workers' such that you can do different tasks in parallel, but I didn't look much into this.

There are also different variants to be made, such as a model where convection is a limiting factor or one where cold pools expand with different velocities or with decreasing velocities. Having convection be a limiting factor could simply be achieved by putting a ceiling on how many cold pools can be seeded in any given generation, but since all models decreased in numbers, this would only be relevant in a unified model. Having the cold pools expand at different velocities would alter the equations that one would need to solve and would be considerably more challenging than having convection be a limiting factor, nonetheless it should be possible.
Lastly one could also consider approaching the first and second test differently, where instead of comparing generations one could group cold pools in time seeded and look for clusters in time instead of within generations. Generations was a concept that came to be, as a result of wave fronts becoming inactive after colliding with other wave fronts, and while there are some correlation with generations and time seeded, they are not the same. You could have cold pools of the same generation being seeded at very different times and cold pools of different generations be seeded at the same time. Clusters in time might be a more meaningful result than clusters in generations, but I would like to add here that clusters are cold pools that are closely together and the difference in time seeded for cold pools in the same area of the domain is going to be smaller than the difference between time seeded of cold pools in the same generation but in different and distant areas of the domain - so the result may not be that different.

References

[1] Todd S Glickman and Walter Zenk. *Glossary of meteorology*. AMS (American Meteorological Society), 2000.

[2] Richard Rotunno, Joseph B Klemp, and Morris L Weisman. A theory for strong, long-lived squall lines. *Journal of Atmospheric Sciences*, 45(3):463–485, 1988.

[3] George H Bryan, Jason C Knievel, and Matthew D Parker. A multimodel assessment of rkw theory's relevance to squall-line characteristics. *Monthly weather review*, 134(10):2772–2792, 2006.

[4] Leah D Grant, Todd P Lane, and Susan C van den Heever. The role of cold pools in tropical oceanic convective systems. *Journal of the Atmospheric Sciences*, 75(8):2615–2634, 2018.

[5] Leah D Grant, Mitchell W Moncrieff, Todd P Lane, and Susan C van den Heever. Shear-parallel tropical convective systems: Importance of cold pools and wind shear. *Geophysical Research Letters*, 47(12):e2020GL087720, 2020.

[6] Zhe Feng, Samson Hagos, Angela K Rowe, Casey D Burleyson, Matus N Martini, and Simon P de Szoeke. Mechanisms of convective cloud organization by cold pools over tropical warm ocean during the amie/dynamo field campaign. *Journal of Advances in Modeling Earth Systems*, 7(2):357–381, 2015.

[7] Paquita Zuidema, Giuseppe Torri, Caroline Muller, and Arunchandra Chandra. A survey of precipitation-induced atmospheric cold pools over oceans and their interactions with the larger-scale environment. *Surveys in Geophysics*, 38(6):1283–1305, 2017.

[8] Jan O Haerter, Steven J Böing, Olga Henneberg, and Silas Boye Nissen. Circling in on convective organization. *Geophysical Research Letters*, 46(12):7024–7034, 2019.

[9] Simon P de Szoeke, Eric D Skyllingstad, Paquita Zuidema, and Arunchandra S Chandra. Cold pools and their influence on the tropical marine boundary layer. *Journal of the Atmospheric Sciences*, 74(4):1149–1168, 2017.

[10] Adrian M Tompkins. Organization of tropical convection in low vertical wind shears: The role of cold pools. *Journal of the atmospheric sciences*, 58(13):1650–1672, 2001.

[11] Peter G Black. Mesoscale cloud patterns revealed by apollo-soyuz photographs. *Bulletin of the American Meteorological Society*, 59(11):1409–1419, 1978.

[12] Adrian M Tompkins. Organization of tropical convection in low vertical wind shears: The role of water vapor. *Journal of the atmospheric sciences*, 58(6):529–545, 2001.

[13] Jan O Haerter, Böing, and Silas Boye Nissen. The initial symmetry breaking in convective self-aggregation. 2021.

[14] Giuseppe Torri and Zhiming Kuang. On cold pool collisions in tropical boundary layers. *Geophysical Research Letters*, 46(1):399–407, 2019.

[15] David M Romps and Nadir Jeevanjee. On the sizes and lifetimes of cold pools. *Quarterly Journal of the Royal Meteorological Society*, 142(696):1517–1527, 2016.

[16] Olga Henneberg, Bettina Meyer, and Jan O Haerter. Particle-based tracking of cold pool gust fronts. *Journal of Advances in Modeling Earth Systems*, 12(5):e2019MS001910, 2020.

[17] Jee-Hoon Jeong, Alexander Walther, Grigory Nikulin, Deliang Chen, and Colin Jones. Diurnal cycle of precipitation amount and frequency in sweden: observation versus model simulation. *Tellus A: Dynamic Meteorology and Oceanography*, 63(4):664–674, 2011.

[18] Stephen W Nesbitt and Edward J Zipser. The diurnal cycle of rainfall and convective intensity according to three years of trmm measurements. *Journal of Climate*, 16(10):1456–1475, 2003.

[19] Chan Xiao, Weihua Yuan, and Rucong Yu. Diurnal cycle of rainfall in amount, frequency, intensity, duration, and the seasonality over the uk. *International Journal of Climatology*, 38(13):4967–4978, 2018.