Precipitation nowcasting using deterministic and probabilistic spatio-temporal models

Sahil Grover

Supervisor: Dr. Stefan Siegert

April 24, 2020

Abstract

Current weather prediction is centred on the use of complex numerical weather prediction models run by supercomputers. Precipitation nowcasting can benefit from simplified modelling by analysing and reproducing advection processes. This report will focus on the challenges that come with building a simple linear advection model in the programming language R and the application of the model to radar data. The report will also start to address the various ways randomness in radar data can be modelled using a probabilistic model. The project comes across many obstacles from stability issues with finite difference schemes to parameter estimation bias when estimating velocities and variances. Once the model had been constructed, we observed our parameter estimation tools perform quite well in most scenarios, but had certain drawbacks particularly when storms occurred. Whilst our probabilistic model performed quite poorly, our deterministic model shined and showed promise for the capabilities of the model in the future.

Contents

Introduction

The weather forms an important factor in many industries: event planning, construction, sports etc. The prediction of precipitation on small time-scales can be particularly important in time-sensitive events such as the Silverstone F1 Grand Prix where the rain directly impacts the performance of the cars. Short-range precipitation prediction falls under a branch of weather prediction called precipitation nowcasting which will be the focus of this report. The World Meteorological Organization defines nowcasting as "forecasts obtained by extrapolation for a period of 0 to 6 hours" into the future [1]. The focus of this report will center on constructing a model in the programming language R that will be able to make precipitation forecasts 15 to 90 minutes into the future. [2].

Existing weather prediction systems owned by weather services like the Meteorological Office(Met Office) use complex numerical weather prediction models to predict the weather, modelling a large variety of complicated processes to forecast precipitation. Existing literature attempts precipitation nowcasting using a variety of methods from deep learning approaches using recurrent neural networks to estimation of velocities and applying semi-lagrangian schemes for forecasts [3,4]. In this project the main focus will be on estimating advection processes between observed radar images prior, and combining this with a linear advection model to output forecasted radar images. Advection is the transport of some substance or quantity in a liquid [5]. The MAPLE algorithm, developed at the JS Marshall Radar Observatory, applies a similar method using a modified semi-lagrangian scheme but does not apply a probabilistic approach which will be investigated later in this report.
In sections 1 and 2, we'll try to understand our radar data and where it comes from and then use patterns in the data to propose hypotheses that can form the basis of our model. We'll follow this in section 3 with building a function that can simulate advection on our radar images based on parameter inputs. Section 4 will focus on using various statistical methods to estimate the parameters based on radar images prior. Finally, in section 5 we'll evaluate the performance of our model using a variety of indicators.

1 The Data

The majority of weather websites cover time intervals of an 1 hour or more on their main webpage for the daily consumer, however these services do publish data of forecasts and observations in smaller time intervals. The Met Office publish numerous products such as location-specific data, textual data, radar data, etc. I download map overlay observations of the UK in 15 minute intervals from the Met Office datapoint using an API key [6]. This data will form the basis under which I build my model. Before I can analyse this data and draw conclusions, it is necessary to understand where it comes from and the uncertainties that exist. The

Met Office uses weather radars(figure 1.1) to determine the location and intensity of precipitation at a specific moment in time.

Figure 1.1 - Cobbacombe Cross radar, a Doppler weather radar located near Tiverton, England [7].

A weather radar consists of a transmitter and a receiver. The transmitter emits electromagnetic pulses of wavelength 5.6cm which is reflected by the precipitation(shown in figure 1.2). The intensity of the precipitation can be deduced from the intensity of the returning pulse and the location of the precipitation can be deduced from the time taken of the pulse from transmission to return. This data is then adjusted based on a "number of rain gauges" [8].

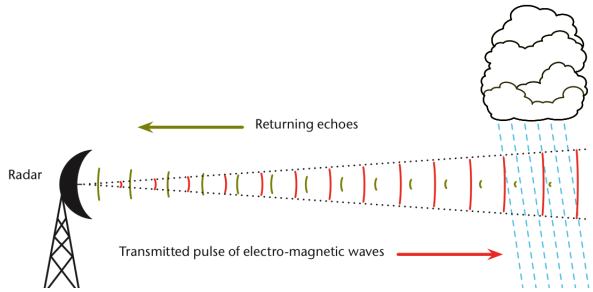


Figure 1.2 - A graphic demonstrating how weather radars retrieve data about precipitation using electromagnetic pulses [9].

The radar generates circular maps of the precipitation intensity up to 255km from radar site. The UK weather radar system currently has 18 radars dotted throughout the UK which are used to produce the map overlays I use throughout this paper. In order to produce these overlays, data from the radars is sent to the Met Office HQ in Exeter in order to produce a composite picture after processing. This processing involves the removal/reduction of:

Cluttering - Produced via reflection from hills and buildings producing permanent echoes.

Anomalous Propagation - Produces false radar echoes due to refraction caused by the pulses passing through air of varying density. When a low temperature inversion exists the range of radar is limited as pulses are refracted into the ground before they can reach the maximum radial distance.

Errors produced via raindrop size variation - Average values are used for precipitation raindrop size and so this will cause radars to underestimate the rain for clouds producing smaller than average raindrops(drizzle) and overestimate for clouds producing larger than average raindrops [9, 10].

More information and other types of meteorological errors can be found in reference [9]. Meteorological errors like these play major roles in the uncertainty of the data we are using as the basis of our model and so it's

important to remember that our prediction will be liable to error due to technical and meteorological causes. In particular we see in figure 1.3 below, a decrease in prediction quality on the outer edges of the coverage by the weather radar stations which is another uncertainty to be aware of.

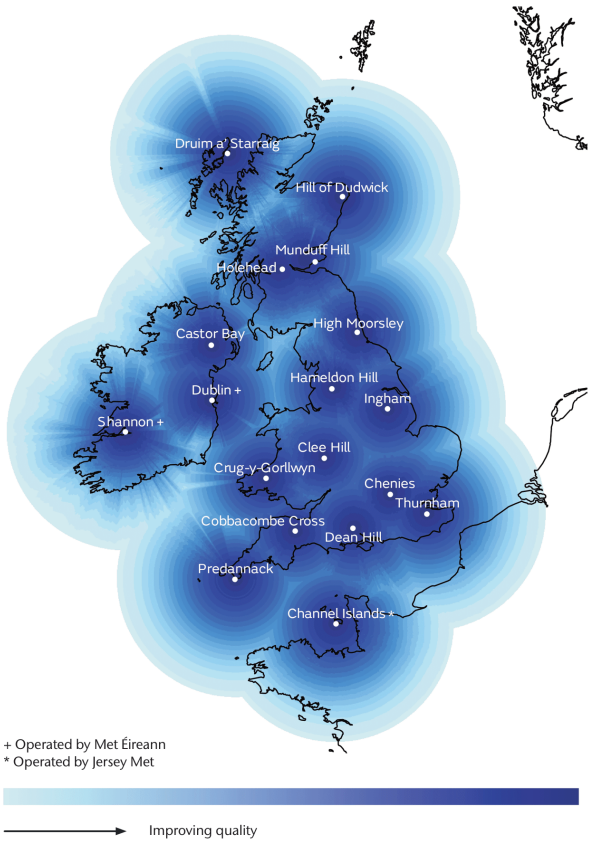


Figure 1.3 - A map displaying the coverage of the 18 weather radar stations situated around the UK and the range and quality of measurements for each station [9].

The map overlay radar images the Met Office datapoint provides are 500x500 in resolution and precipitation values fall inside the same prediction range shown in figure 1.3. Figure 1.4 contains an example of a radar image taken during Storm Ciara at 15:00 on the 9th February.

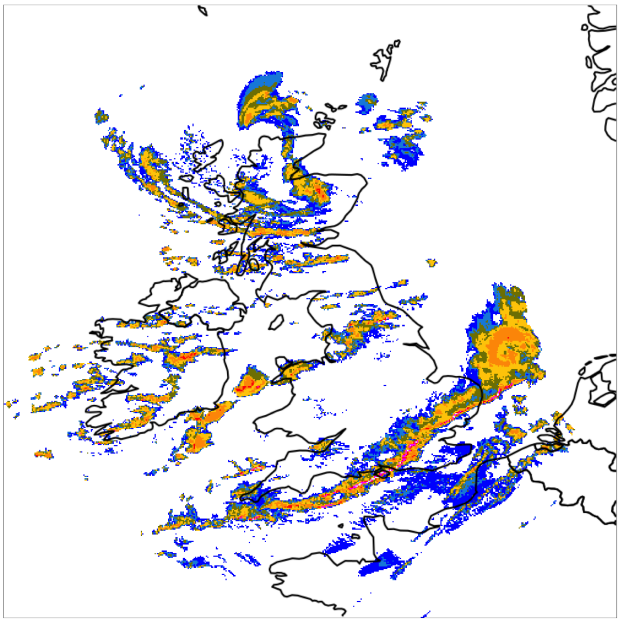


Figure 1.4 - An example radar image with a UK map underlay taken on 09/02/20 at 15:00.

The different colours corresponds to various precipitation intensities on the scale(which can be found in section 3.1) where white is no or very little precipitation and red and magenta indicate a very high value of precipitation. I pull the map overlays from the datapoint using the 'GetRadar.R' file. All files referenced are in the GitHub repository, `https://github.com/sahilkariadata/track_rain`.

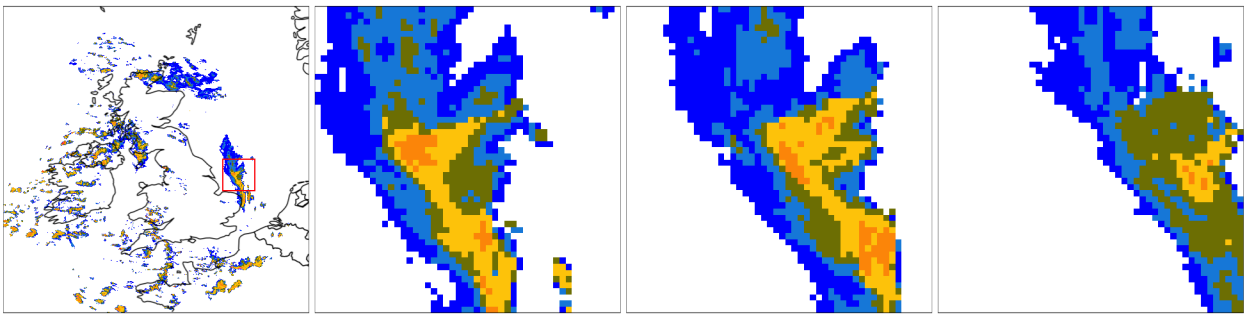


Figure 2.1 - 1 full-sized radar image followed by 3 50x50 cut-out radar images in the same location(red box) captured in intervals of 30 minutes starting from the full-sized image cutout.

2 Observations about the data

After collating the data and analysing consecutive radar images, I observe many interesting patterns. These patterns combined with current knowledge lead me to be able to propose a few hypotheses that I can use to build the model.

1. **The advection of clouds is not the same everywhere within the 500x500 usually.**
2. **Advection in a smaller section within the 500x500 may be the same or similar as groups of clouds travel with similar velocity.**
3. **Diffusion* occurs but has a smaller effect on smaller timescales e.g in 15-60 minute time scenario.** *Note: Any diffusion that does occur can be accounted for as noise.*
4. **Perfect advection does not take place and the different effects that occur can be represented using a probabilistic model e.g. adding a gaussian noise term.**

Observation 2 is of particular importance as it is the contributing assumption towards the validity of my model and the predictions produced. Figure 2.1 is an example where I have cut out a 50x50 to monitor the behaviour of a group of clouds over time. In this report we will refer to groups of non-trivial precipitation values as clouds even though the actual cloud may be larger and not producing rain. From these snapshots, the presence of horizontal advection is apparent which will allow us to later retrieve a velocity in the horizontal direction. I can also see a little diffusion take place, particularly in the reduction of orange level precipitation intensities which supports observation 3. I observe that higher intensity precipitation values tend to occur closer to the center of clouds. The precipitation field maintains its rough shape quite well over time which will yield advantageous when producing predictions on a similar time scale.

3 Building the model

3.1 Image-Matrix Conversion

Modelling the data requires a conversion from visual data into numerical data via changing the individual pixels of our radar images into precipitation intensities. Unfortunately the Met Office does not publish the exact precipitation intensities of the radar images. Therefore I need a scaling system which converts the pixel colours to a specific precipitation intensity. I chose to use the basic scaling system the Met Office provides to daily users that can be found in reference [11] which measures the intensities in mm/hour. I selected the center of the interval for each colour as initial values during conversion from a radar image to a matrix.

Colour	Initial value	Interval
	0	$X_i < 0.01$
	0.255	$0.01 \leq X_i < 0.5$
	0.75	$0.5 \leq X_i < 1$
	1.5	$1 \leq X_i < 2$
	3	$2 \leq X_i < 4$
	6	$4 \leq X_i < 8$
	12	$8 \leq X_i < 16$
	24	$16 \leq X_i < 32$
	48	$32 < X_i$

A function that converts the matrices back into radar images would also be required in order to produce the final prediction. I use the same intervals to convert precipitation intensity to pixel colours in the `matr_to_img()` function. All of my own functions referenced in this paper are defined in 'initialization.R'.

3.2 Advection

In order to build a model that predicts the precipitation field in the future, I need an equation to advect the 2D field. First we'll consider advection in the x-dimension.

3.2.1 Advection along a single dimension

Consider the one-dimensional Linear Advection equation where $\mathbf{X}$ is a 2D vectorised field of precipitation intensities, $\mathbf{X} \in R^d$ where $d = 500^2$ as the radar images are 500x500 in resolution as we'll advect the whole image and cut out the location we are interested in:

$$\frac{\partial \mathbf{X}}{\partial t} + V_x \frac{\partial \mathbf{X}}{\partial x} = 0 \tag{1}$$

Equation 1 describes a wave propagating across the x-axis with velocity V_x [12]. We then introduce a simple finite differencing scheme called the forward upwind scheme in order to apply this to our vector.

$$\frac{X_i^{n+1} - X_i^n}{\Delta t} + V_x \frac{X_i^n - X_{i-1}^n}{\Delta x} = 0 \quad \text{for} \quad V_x \geq 0 \tag{2}$$

$$\frac{X_i^{n+1} - X_i^n}{\Delta t} + V_x \frac{X_{i+1}^n - X_i^n}{\Delta x} = 0 \quad \text{for} \quad V_x < 0 \tag{3}$$

The subscript i refers to vector index and the superscript n refers to the time index. Instead of just iterating this scheme across every grid point, I can change the scheme above

so that it is of a vectorised form which I will refer to as our linear map.

$$\mathbf{X}(t+1) = \mathbf{A}_x\mathbf{X}(t) \tag{4}$$

$\mathbf{A}_x \in R^{d\times d}$ is the advection operator in the form of a big, sparse, square matrix to be defined. $\mathbf{A}_x$ will change based on V_x which dictates the scheme we use. Suppose $V_x \geq 0$ and $\mathbf{A}_x$ uses the scheme in equation 2. Rearranging the scheme gives:

$$X_i^{n+1} = X_i^n - \frac{V_x\Delta t}{\Delta x}(X_i^n - X_{i-1}^n) \tag{5}$$

This can be expressed in matrix form:

$$\mathbf{X}(t+1) = (\mathbf{I} - \mathbf{D}_x\frac{V_x\Delta t}{\Delta x})\mathbf{X}(t) \tag{6}$$

where $\mathbf{I} \in R^{d\times d}$ is the identity matrix and $\mathbf{D}_x \in R^{d\times d}$ is a linear operator for the specific finite difference scheme in equation 5. Looking further into the matrix expansion:

$$C_x\mathbf{D}_x\mathbf{X} = \begin{bmatrix} C_x\mathbf{D} & 0 & 0 & 0 & \dots \\ 0 & C_x\mathbf{D} & 0 & \dots & \\ 0 & 0 & C_x\mathbf{D} & & \\ 0 & \vdots & & \ddots & \\ \vdots & & & & \end{bmatrix}\mathbf{X}$$

and

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots \\ -1 & 1 & 0 & \dots & \\ 0 & -1 & 1 & & \\ 0 & 0 & \ddots & \ddots & \\ \vdots & & & & \end{bmatrix}$$

where $C_x = \frac{V_x\Delta t}{\Delta x}$ is the Courant number, $\mathbf{D} \in R^{\sqrt{d}\times\sqrt{d}}$ is a matrix with dimensions 500x500. $\mathbf{D}_x$ is thereby a matrix with dimensions 250000x250000 which will be stored as a sparse matrix using the 'Matrix' package in R to save space and speed up processing time [13].

3.2.2 Advection along both dimensions

Now considering advection along both axis, I need to find $\mathbf{D}_x$ and $\mathbf{D}_y$ for both schemes. The mathematical equivalent for finding $\mathbf{D}_x$ and $\mathbf{D}_y$ involves using the Kronecker product $\otimes$:
For advection along the x-axis:

$$\mathbf{D}_x = \mathbf{I} \otimes \mathbf{D} \quad for \quad V_x > 0$$

$$\mathbf{D}_x = \mathbf{I} \otimes -\mathbf{D}^T \quad for \quad V_x < 0$$

For advection along the y-axis:

$$\mathbf{D}_y = \mathbf{D} \otimes \mathbf{I} \quad for \quad V_y > 0$$

$$\mathbf{D}_y = -\mathbf{D}^T \otimes \mathbf{I} \quad for \quad V_y < 0$$

The programming language R has a function for computing Kronecker products called `kronecker()` which is compatible with sparse matrices. Once $\mathbf{D}_x$ and $\mathbf{D}_y$ are found, I can use equation 6 and our Courant numbers to find our $\mathbf{A}_x$ and $\mathbf{A}_y$ for advecting along the x-axis and y-axis respectively. Finally multiplying these 2 matrices together gives us our final advection operator $\mathbf{A}$ that will allow us to advect across both axis at once giving the following linear map:

$$\mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t) \tag{7}$$

where $\mathbf{X} \in R^d$ and $\mathbf{A} \in R^{d \times d}$. In order to find out the specific Courant numbers I need the variables: V_x, V_y, Δx, Δy and Δt. The layout of my model means that specific variables are fixed:

$$\Delta x = 3.774 \quad \text{Km}$$

$$\Delta y = 2.886 \quad \text{Km}$$

Δx and Δy are fixed because the resolution of our radar images are fixed. Δx and Δy were found using the longitude and latitude of the corners of the radar image. We expect the Δx to be larger than the Δy due to the curvature of the Earth.

$$\Delta t = 1/4 \quad \text{hours}$$

The time between the radar images is 1/4 of an hour. The velocities, V_x and V_y, can then act as an input to advect our field. Both velocities will be in units km/h.
After building the advection function, specific problems were found when modifying the Courant numbers by modifying the velocities. When choosing a $|C_i| > 1$ for $i = x, y$, via choosing $V_x > 15.096$ or $V_y > 11.544$ the advection process becomes numerically unstable and I retrieve a prediction like in figure 3.1.

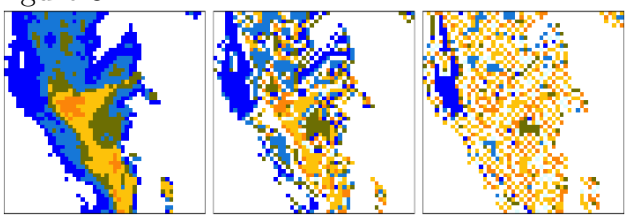


Figure 3.1 - A cutout radar image(left) followed by the same 2 images but advected 15 minutes(center) and 30 minutes(right) into the future respectively. Parameters used: $V_x = 45$, $V_y = 20$.

Finite difference schemes suffer from instability when numerical error created during a time step causes more errors to get magnified as iterations continue. The stability conditions for this finite difference scheme is:

$$|C_i| \leq 1 \quad \text{for} \quad i = x, y \tag{8}$$

which is known as the Courant-Friedrichs-Lewy (CFL) stability criterion and can be found using Von Neumann stability analysis [12, 14, 15]. I also found a different issue occurred if $|C| < 1$ where smearing occurs, otherwise known as numerical diffusion:

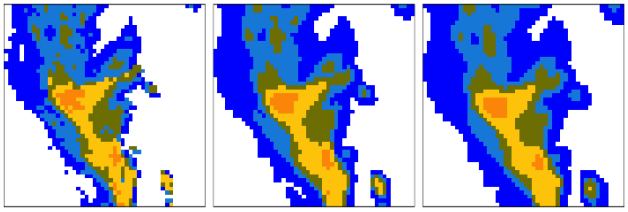


Figure 3.2 - A cutout radar image(left) followed by the same 2 images but advected 30 minutes(center) and 60 minutes(right) into the future respectively. Parameters used: $V_x = 5$, $V_y = 5$.

Although diffusion is a process that occurs naturally, we initially aimed to produce pure advection without numerical artifacts. The diffusion takes place because the resolution of my image is fixed and so the velocity is not high enough to advect a full pixel so it has split the advection between 2 pixels. For example consider a 1D system [2|0|0|0|0] where $V_x = 1$, $\Delta x = 1$, and $\Delta t = 1$ then the resulting system will be [0|2|0|0|0] after 1 time step. However if $V_x = 0.5$, then $C = 0.5$ and the 2 will be split between grid points,

resulting in the system [1|1|0|0|0] after 1 time step thereby causing numerical diffusion. In fact perfect advection with no numerical diffusion only takes place when $|C| = 1$. In order to solve this problem, I chose to modify the function so that C is always one ensuring pure advection. When $C = 1$, and I compute 1 time step, I observe advection by 1 pixel. I would then modify the number of timesteps or number of times the individual advection operators are applied in order to simulate a higher velocity. For example suppose we want to advect a 2D field with velocities, $V_x = 60.384$ and $V_y = 23.088$, to get a prediction for the next 15 minutes, then we'd apply the advection operator in the x-axis, $\mathbf{A}_x$, 4 times and the advection operator in the y-axis, $\mathbf{A}_y$, 2 times. i.e. $\mathbf{A} = \mathbf{A}_x^4 \times \mathbf{A}_y^2$. In order to maintain the current resolution, the velocities inputted would need to be floored to specific discrete values which is a consequence of formulating the model in this way. This simplifies the model to a simple shift of the pixels where the direction and size of the shift in the x and y direction are dictated by the velocities V_x and V_y. Using this method produces the following result:

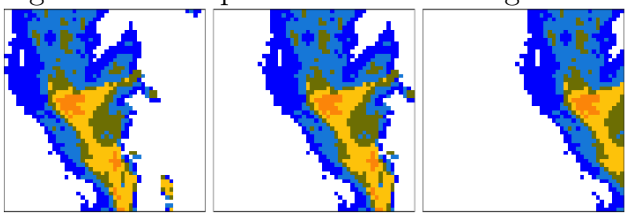


Figure 3.3 - A cutout radar image(left) followed by the same 2 images but advected for 30 minutes(center) and 60 minutes(right) into the future respectively. Parameters used: $V_x = 90.576$, $V_y = 0$.

Note: A shift downwards in this model is a result of advecting with a positive velocity in the y-axis and vice versa for a shift upwards. The `project()` function, found in 'initialization.R', exhibits the deterministic process found in figure 3.3 which showcases a clear shift of pixels between images.

3.3 Gaussian noise

Under observation 4 in section 2, we know perfect advection almost never takes place. This means our deterministic model, $\mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t)$, will fall short of truly representing the real physical process that occurs. This is where the addition of a gaussian noise term will be useful as it will allow us to account for processes I may have not included(e.g diffusion) and account for randomness and uncertainties in the weather radar readings. Our model would then become a vector autoregressive model or VAR(1) model:

$$\mathbf{X}(t+1) = \mathbf{A}\mathbf{X}(t) + \boldsymbol{\varepsilon}(t) \tag{9}$$

where $\boldsymbol{\varepsilon} \in R^d$ is a vector of 250000 random variables and added at each iteration of the equation [16]. I will consider 2 types of noise, independent, identically distributed noise(IID noise) and spatially correlated noise.

3.3.1 IID noise

For IID noise, we consider each grid point as an independent variable that is normally distributed and hence our vector $\boldsymbol{\varepsilon}$ contains

normally distributed random variables with mean 0 and variance σ^2:

$$\varepsilon_i \sim \mathcal{N}(0,\, \sigma^2)$$

where $1 \leq i \leq 250000$. I generate the 250000 normally distributed variables in a column vector using the command `rnorm(250000,mean=0,sd=sigma)`. Using this method causes us to run into some problems since we can't have negative precipitation values and when $X_i = 0$ and $\varepsilon_i < 0$ at a specific grid point, this occurs. To counter this we can take the $\log_2$ of our precipitation field, then add the noise and follow with the exponentiation of the field to base 2. If an initial grid point precipitation value is zero, it will be set to -7 instead of negative infinity when taking $\log_2$. Using this system more or less prevents a negative precipitation value occurring for reasonable variance values.

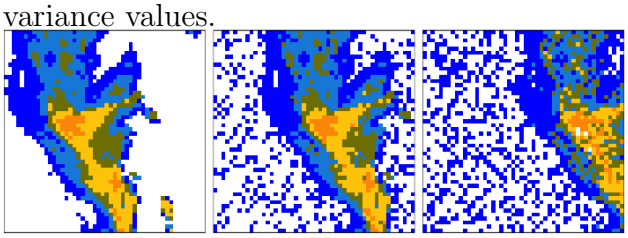


Figure 3.4 - A cutout radar image(left) followed by the same 2 images but advected 30 minutes(center) and 60 minutes(right) into the future respectively. Parameters used: $V_x = 90.576$, $V_y = 0$, $\sigma^2 = 0.02$.

3.3.2 Spatially correlated noise

Realistically the random variables will not be independent of each other as neighbouring grid points will be dependent on each other. One way to account for this dependence is to implement spatially correlated noise by introducing a special case of a Markov random field called a conditional autoregressive model(CAR) [17]. Consider a Markov random field where each grid point is dependent on its neighbours like below.

$$P(\varepsilon_{i,j}|\varepsilon_{-(i,j)}) = P(\varepsilon_{i,j}|\varepsilon_{n(i,j)})$$

where $-(i,j)$ represents all other grid points and $n(i,j) = [(i-1,j),(i+1,j),(i,j-1),(i,j+1)]$ [18].

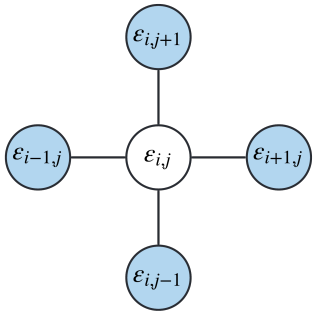


Figure 3.5 - A graphic showing the dependency of each grid point $\varepsilon_{i,j}$ *in our Markov random field, where the circles represent grid points.*

The i and j are indices for the x and y axis of the precipitation field respectively. Each grid point is dependent on its immediate adjacent neighbours with Dirichlet boundary conditions of zero [17]. In order to retrieve our vectorised 2D random field for the spatially correlated noise, $\boldsymbol{\varepsilon} \in R^d$, we consider a large matrix multiplication to carry out the process.

$$\boldsymbol{\varepsilon} = \mathbf{K}^{-1}\mathbf{E}$$

where $\mathbf{E} \in R^d$ is now the vector of IID normal variates and $\mathbf{K} \in R^{d \times d}$ is the CAR operator with $d = 250000$. $\mathbf{K}$ can be found using:

$$\mathbf{K} = \frac{1}{4}\begin{bmatrix} \mathbf{C} & -\mathbf{I} & 0 & 0 & \dots \\ -\mathbf{I} & \mathbf{C} & -\mathbf{I} & 0 & \dots \\ 0 & -\mathbf{I} & \mathbf{C} & -\mathbf{I} & \\ 0 & 0 & \ddots & \ddots & \ddots \\ \vdots & \vdots & & & \end{bmatrix}$$

$\mathbf{I}$ is a 500x500 identity matrix and $\mathbf{C}$ is the 500x500 matrix:

$$\mathbf{C} = \begin{bmatrix} 1 & -1 & 0 & 0 & \dots \\ -1 & 1 & -1 & 0 & \dots \\ 0 & -1 & 1 & -1 & \\ 0 & 0 & \ddots & \ddots & \ddots \\ \vdots & \vdots & & & \end{bmatrix}$$

A simulated realisation of the spatially correlated noise created by the CAR process is shown in figure 3.6:

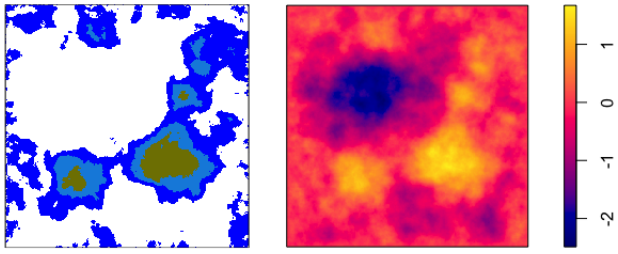


Figure 3.6 - A realisation of the Gaussian noise field created using a $\sigma^2 = 0.0001$(right) and the conversion using the matrix to image intervals(disregarding negative values) in section 3.1(left).

We observe the noise take patterns similar to the radar images we have observed prior. Although since the individual grid points are no longer independent, the implementation will be more difficult than the implementation of the IID noise particularly when estimating the variance. In the 'Model Evaluation' section the IID noise model is the only probabilistic model that will be used for comparison due to time constraints. Both the deterministic model and probabilistic(with noise) model can be applied to produce predictions as will be seen later in the paper.

4 Parameter Estimation

In order to produce predictions with our advection model I need velocity inputs, V_x and V_y, and when considering our probabilistic model, a variance input σ^2 aswell. Earlier we saw our model simplify down to a simple shift of pixels so in essence we want to determine the number of pixels the cloud has shifted. This can be done using least squares estimation.

4.1 Least Squares Estimation

Least squares estimation is a loss function for a specific optimization problem. In this case I will be optimizing to reduce the sum of squared errors(SSE) between 2 precipitation fields. Consider the following 2 precipitation fields: $\mathbf{X}_t = \mathbf{X}(t)$ and $\mathbf{X}_{t+1} = \mathbf{X}(t+1)$ for time notation sake in section 4. Then the optimization problem takes the following form:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} [\mathbf{X}_{t+1} - \mathbf{A}\mathbf{X}_t]^T [\mathbf{X}_{t+1} - \mathbf{A}\mathbf{X}_t] \quad (10)$$

where $\theta = (V_x, V_y)$ and $\mathbf{X}_t$, $\mathbf{X}_{t+1} \in R^c$, are vectorised 2D fields of the cutout location

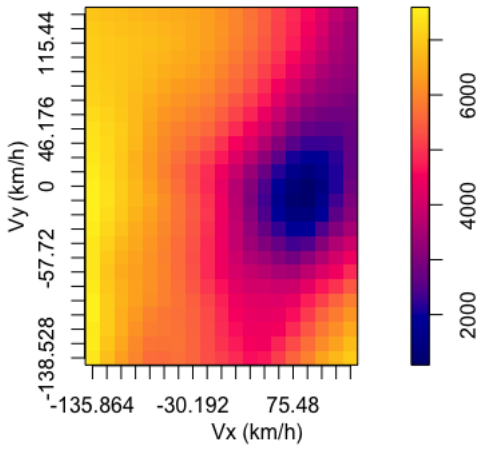
we are interested in therefore c = number of pixels in cutout. I will solve this problem by performing an exhaustive search of all possible discrete values of V_x and V_y up to the maximum velocity of rain clouds which I'll take as 140 km/h. As said earlier I can simplify this problem down to a shift of pixels, so I find the number of pixels shifted for the maximum velocity. I perform the exhaustive search shifting $\mathbf{X}_t$ up to this maximum number of pixels in all directions and find the sum of squared errors(SSE) for each like in equation 10. This is computed by my function `err_map()` which produces a matrix of SSE values based on advecting using the different possible discrete V_x and V_y values where the argmin is based on the location of the minimum in this matrix. This will be evident later when I plot a heatmap of this matrix for analysis.

At a fundamental level, the cross product in equation 10 is comparing the pixels before and after advection which is based on the scale we set in section 3.1. This will give a much higher weight to more intense pixels like yellow and orange than blue. With our current scale a blue pixel is 0.255 and a yellow pixel is 3 meaning a yellow pixel has approximately 12x the weight of a blue pixel. In order to counteract this I can take the $\log_2$ of the field which re-weights the pixels so that the 9 colours have a slightly fairer weighting.

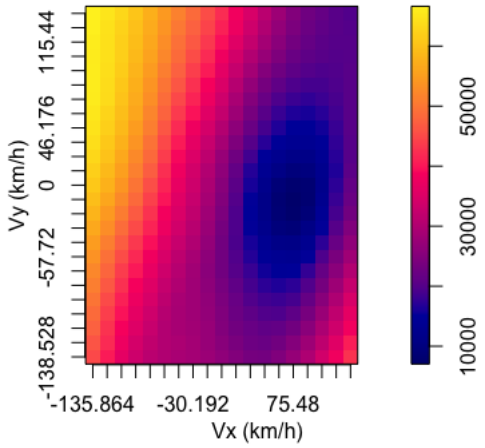
Colour	$X_{i,j}$	$\log_2(X_{i,j})$
	0	-7*
	0.255	-1.971
	0.75	-0.415
	1.5	0.585
	3	1.585
	6	2.585
	12	3.585
	24	4.585
	48	5.585

**White pixels or* 0 *will be reweighted to* -7 *instead of* $-\infty$.

This drastically helps the weighting difference for the magenta and very light blue pixels which are extreme precipitation intensities and would largely affect the shape of our heatmap and have a relatively large effect on the optimum parameters. On the other hand I have to be careful to not give white pixels too large a weighting. Conversely penalising a system that predicts rain when there is none could yield potentially beneficial. Consider the first cutout radar image in figure 2.1 as $\mathbf{X}_0$ and the radar image taken 15 minutes after as $\mathbf{X}_1$. Using my `err_map()` function for the 2 different weighting systems I retrieve the following 2 heatmaps in figure 4.1.



(A)



(B)

Figure 4.1 - 2 heatmaps of the various SSE values between 2 precipitation fields, $\mathbf{A}\mathbf{X}_0$ *and* $\mathbf{X}_1$*, varying* V_x *and* V_y*. A uses the original precipitation scale and B uses the adjusted* $\log_2$ *scale.*

Using the new scale flattens the surface and in particular stretches the trough particularly on the V_y axis. The least squares estimation uses the 2d surface shown and returns $\hat{\theta} = (\hat{V}_x, \hat{V}_y)$, which is the location of the trough. The least squares estimation with the new adjusted scale returns the velocities, $V_x = 75.480$ km/h and $V_y = -11.544$ km/h. A strong V_x is expected as we saw earlier the images from figure 2.1 have a noticeable 'horizontal advection'. Since I can retrieve the velocity between 2 radar images $\mathbf{X}_0$ and $\mathbf{X}_1$, I can use this in combination with my model to make deterministic and probabilistic predictions for $\mathbf{X}_2$, $\mathbf{X}_3$ etc. using the VAR(1) model. In order to validate this system works I can check if velocities between consecutive images are similar as the velocity of a cloud will not drastically change every 15 minute interval. Using the same radar images as in figure 2.1 and more:

Interval	V_x	V_y
$\mathbf{X}_0 \to \mathbf{X}_1$	75.480	-11.544
$\mathbf{X}_1 \to \mathbf{X}_2$	75.480	-11.544
$\mathbf{X}_2 \to \mathbf{X}_3$	75.480	-11.544
$\mathbf{X}_3 \to \mathbf{X}_4$	75.480	-11.544
$\mathbf{X}_4 \to \mathbf{X}_5$	90.576	-11.544
$\mathbf{X}_5 \to \mathbf{X}_6$	60.384	-23.088

As can be seen the least squares estimation method retrieves rather consistent results. The cloud starts to edge off the 50x50 location we are sampling at $\mathbf{X}_5$ explaining the change in parameters found, but also the cloud could just be accelerating/decelerating. For our VAR(1) model, we would need an estimate of the variance. Assuming the residuals are IID, we can take the variance to be the average SSE across each pixel hence:

$$\sigma^2 = \frac{1}{n}[\mathbf{X}_{t+1} - \hat{\mathbf{A}}\mathbf{X}_t]^T[\mathbf{X}_{t+1} - \hat{\mathbf{A}}\mathbf{X}_t]$$

where n is the number of pixels/grid points and $\hat{\mathbf{A}}$ is the advection operator that is based

on $\hat{V}_x$ and $\hat{V}_y$. The variance found is also then on the $\log_2$ scale and the noise will also be added on a similar scale as mentioned previously in section 3.3.1.

Least squares estimation gives a point estimate for my parameters, $\theta = (V_x, V_y, \sigma^2)$, but does not account for parameter uncertainty. Consider a scenario where another SSE value is almost as small as the SSE at the trough, then this set of parameters is almost as likely to be the real parameters but the least squares estimation method will not consider this value. This is where Bayesian inference can be particularly useful in accounting for the uncertainty in parameters.

4.2 Bayesian Inference

Bayesian inference is a method of parameter estimation that treats the parameters θ as random variables, to retrieve a posterior distribution $P(\theta|\text{data})$, using Bayes' rule and a prior distribution. I will then be able to sample from this distribution to retrieve a number of parameter estimates that will account for the uncertainty the least squares estimation method didn't.

In order to make statements about the distribution of $\theta = (V_x, V_y, \sigma^2)$ given the data, I need a joint probability distribution of $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$ conditional on θ, where $\mathbf{X}_t$, $\mathbf{X}_{t+1} \in R^c$ are vectorised 2D fields of the cutout location. I assume that $\mathbf{X}_t$ has the uninformative uniform distribution, $P(\mathbf{X}_t|\theta) \propto 1$ so that:

$$P(\mathbf{X}|\theta) = P(\mathbf{X}_t|\theta)P(\mathbf{X}_{t+1}|\mathbf{X}_t, \theta) \propto P(\mathbf{X}_{t+1}|\mathbf{X}_t, \theta)$$

Using Bayes' rule, the posterior distribution is given by:

$$P(\theta|\mathbf{X}) = \frac{P(\theta)P(\mathbf{X}|\theta)}{P(\mathbf{X})}$$

where $\mathbf{X} = (\mathbf{X_t}, \mathbf{X}_{t+1})$ [19]. I will assume an uninformative uniform prior distribution of the form: $P(\theta) \propto 1$. Then our posterior simplifies as such:

$$P(\theta|\mathbf{X}) \propto P(\mathbf{X}|\theta) \propto P(\mathbf{X}_{t+1}|\mathbf{X}_t, \theta) \quad (11)$$

$P(\mathbf{X}_{t+1}|\mathbf{X}_t, \theta)$ is the likelihood function. If I assume that the residuals ε, from the VAR(1) model are IID, then my likelihood function is given by:

$$\begin{aligned}&\mathcal{L}(\theta; \mathbf{X}_t, \mathbf{X}_{t+1}) \\ &= (2\pi\sigma^2)^{-\frac{n}{2}} e^{-\frac{1}{2\sigma^2}[\mathbf{X}_{t+1} - \mathbf{A}\mathbf{X}_t]^T[\mathbf{X}_{t+1} - \mathbf{A}\mathbf{X}_t]}\end{aligned}$$

where n is the number of pixels. Using standard maximum likelihood estimation procedure, we take the natural log of this equation to simplify the function without changing the location of the maximum from our original function. I also decide to take $e^\tau = \sigma^2$ in order to ensure $\sigma^2 > 0$. Our likelihood function then takes the form:

$$\begin{aligned}l(\theta; \overline{X}_t, \overline{X}_{t+1}) = &-\frac{n}{2}\ln 2\pi e^\tau - \frac{e^{-\tau}}{2}[\overline{X}_{t+1} \\ &- \mathbf{A}\overline{X}_t]^T[\overline{X}_{t+1} - \mathbf{A}\overline{X}_t]\end{aligned} \quad (12)$$

I have already calculated the cross product for different V_x and V_y values and can be retrieved in the form of a matrix using `err_map()`. I can then iterate this likelihood function through a range of values of τ in order to get a 3d grid. Using the same radar images and cross products from figure 3.6 the following cross section plots can be retrieved.

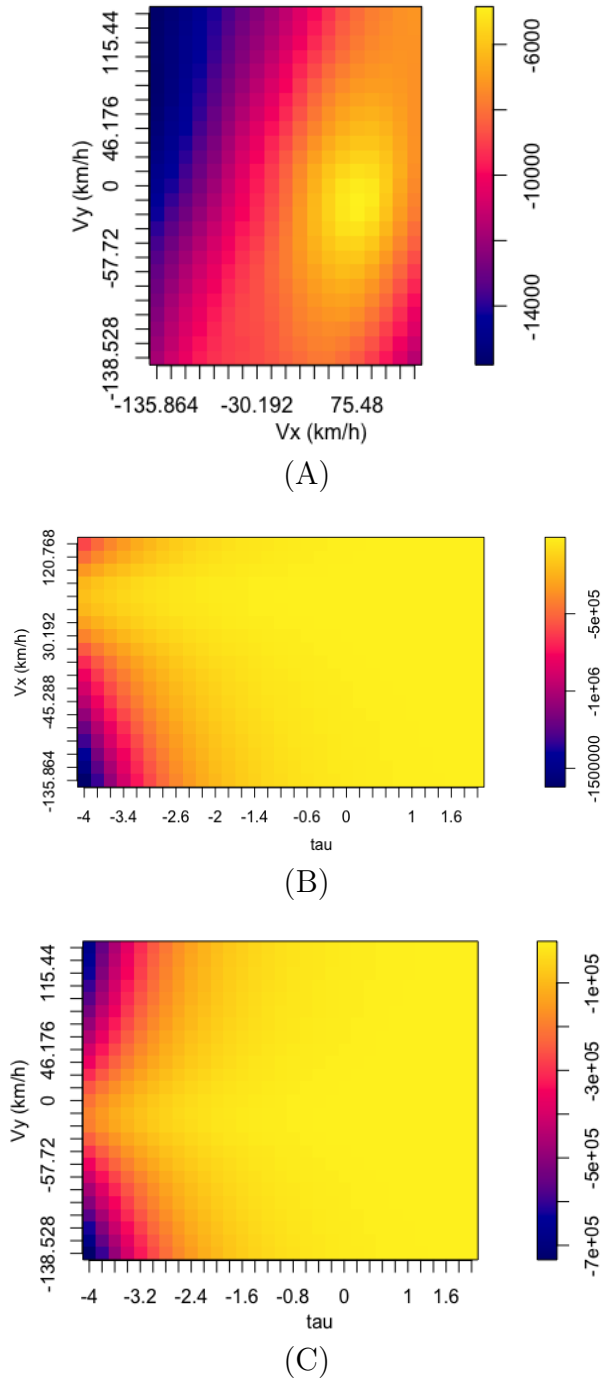


Figure 4.2 - 3 plots of the cross sections of the likelihood grid created by equation 12. The maximum occurs at $\hat{\theta} = (\hat{V}_x, \hat{V}_y, \hat{\tau})$. (A) A cross section plot taken at $\hat{\tau}$ of V_y against V_x. (B) A cross section plot taken at $\hat{V}_y$ of V_x against τ. (C) A cross section plot taken at $\hat{V}_x$ of V_y against τ.

In plot (A) we observe the same surface as in figure 4.1A but inverted because this is a likelihood function not a loss function and the log-likelihood is proportional to the sum of squared errors. We see that plots (B) and (C) don't really say too much about the distribution visually apart from the fact lots of various tau values are similarly likely. For higher tau values we observe higher likelihoods tend to center around V_x,$V_y = 0$. Taking the maximum will retrieve the same parameters as the least squares estimation. To get our posterior distribution, we normalise the likelihood function/grid and sample from the distribution.

$$P(\theta|\mathbf{X}) = \frac{P(\mathbf{X}_{t+1}|\mathbf{X}_t, \theta)}{\int d\theta P(\mathbf{X}_{t+1}|\mathbf{X}_t, \theta)}$$

I approximate this by using the `sample()` function. I can sample m parameter sets θ_i from weights proportional to the likelihood where $1 \leq i \leq m$. Performing bayesian inference on the same 2 precipitation fields used to produce the heatmaps in figure 4.1, $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$, I retrieve consistent values for θ on 10000 samples where $\theta = (75.480, -11.544, 2.718)$.

For a prediction of $\mathbf{X}_{t+2}$, I can generate a

posterior predictive distribution:

$$P(\mathbf{X}_{t+2}|\mathbf{X}) = \int P(\mathbf{X}_{t+2}|\mathbf{X}, \theta)P(\theta|\mathbf{X})d\theta$$

This can be approximated by a method of numerical integration called, Monte Carlo Integration:

$$p(\mathbf{X}_{t+2}|\mathbf{X}) \approx \frac{1}{m}\sum_{i=1}^{m} P(\mathbf{X}_{t+2}|\mathbf{X}_{t+1}, \theta_i)$$

where θ_i are m parameter sets sampled from $P(\theta|\mathbf{X})$ and $P(\mathbf{X}_{t+2}|\mathbf{X}_{t+1}, \theta_i)$ is a multivariate normal distribution with expectation $\mathbf{A}_i\mathbf{X}_{t+1}$ and variance matrix $e^{\tau_i}\mathbf{I}$, where $\mathbf{I}$ is an identity matrix. The posterior predictive distribution is approximated by a gaussian mixture distribution where the expectation of the predictive distribution can be estimated by individual expectations of $\mathbf{X}_{t+2}$ conditional on θ_i, averaged over all i [20]. In other words the posterior predictive mean can be found by advecting $\mathbf{X}_{t+1}$ using the sample parameters θ_i and averaging the resulting vectorised 2D precipitation fields which is carried out by my function `ppm()`.

5 Model Evaluation

With construction of the advection model and parameter estimation functions complete, I am now able to use observed radar data to make predictions. First I will look at the different ways I am going to evaluate these predictions as the strength/performance will range both numerically and visually. I will also look at the particular pitfalls of the model and specific scenarios where the model fails whilst comparing the effectiveness of both the deterministic model and the probabilistic model.

5.1 Techniques

In order to evaluate a model, we will need a quantity to describe the strength of the prediction. One way is to find the sum of squared residuals(SSE) between the predicted and actual precipitation field. This will be done using the `err_score()` function which I'll refer to as the 'error score' throughout this section. A suitable comparison for this number could be using the actual precipitation field and the predicted field before advection to get a score that our error score should be smaller than to show the advection has made an improvement. We'll call this our 'persistence'.
In theory we should see the error score get larger as we predict further into the future as the cloud will change velocity and other processes like diffusion will take place.
A performance indicator for the visual strength of the model could also be useful. I can compare the individual pixel colours of the predicted and actual radar image and I'll denote the number of matched pixels as 'N'.

5.2 Successes and Pitfalls

Like any computational model, this model has its own strengths and weaknesses. The 50x50 location used in figure 2.1(and throughout the 'Building the model' section) follows a solid cloud experiencing clear horizontal advection where the direction of travel

can be deduced qualitatively. However what happens when the cloud is dispersed and the advection direction is not qualitatively clear from a 50x50?

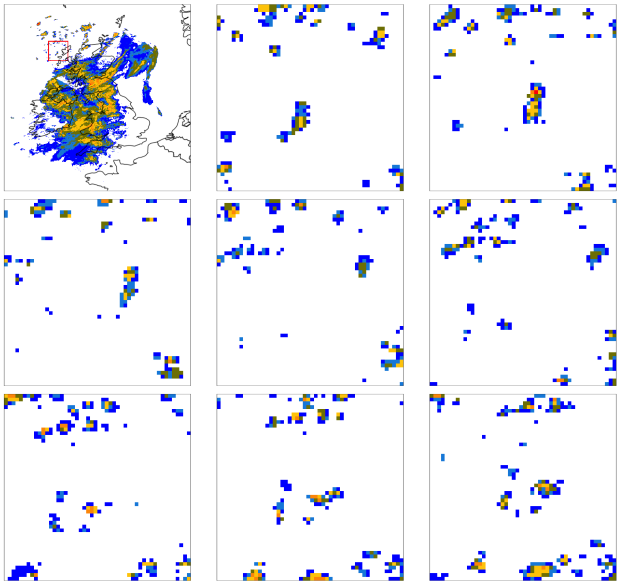


Figure 5.1 - 1 full-sized radar image with a UK underlay at t_0 and 8 50x50 cut-out radar images in the same location(red box) captured in intervals of 15 minutes(from left by row) starting from t_0. Taken on 08/02/2020 between 22:00-23:45.

Unlike figure 2.1, the clouds are quite sparse and dispersed. There is some noticeable horizontal and vertical advection although it is not as clear visually as the example in figure 2.1. We can plot error heat maps of the velocity to observe the certainty and consistency of our model in the intervals: $\mathbf{X}_0 \rightarrow \mathbf{X}_1$, $\mathbf{X}_1 \rightarrow \mathbf{X}_2$, $\mathbf{X}_2 \rightarrow \mathbf{X}_3$, $\mathbf{X}_3 \rightarrow \mathbf{X}_4$ where $\mathbf{X}_0$, $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$ are the first 5 50x50 precipitation fields in figure 5.1.

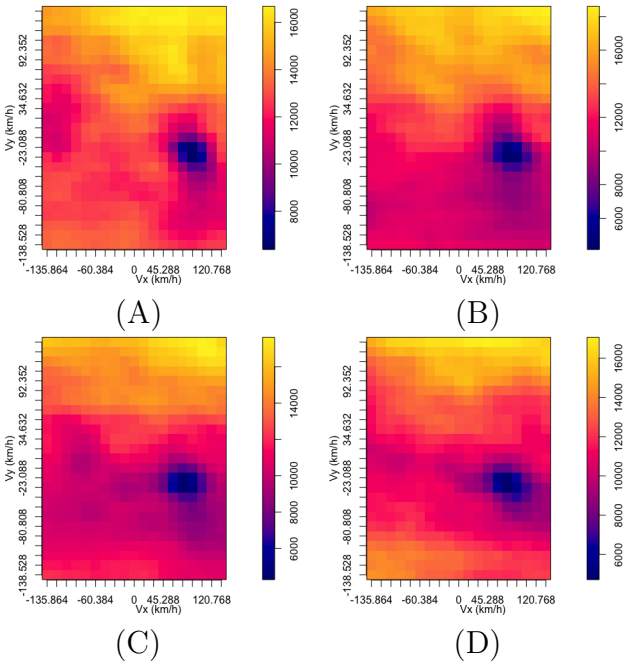


Figure 5.2 - 4 error heatmaps for the precipitation fields in figure 5.1. A covers the interval $\mathbf{X}_0 \rightarrow \mathbf{X}_1$. B covers the interval $\mathbf{X}_1 \rightarrow \mathbf{X}_2$. C covers the interval $\mathbf{X}_2 \rightarrow \mathbf{X}_3$. D covers the interval $\mathbf{X}_3 \rightarrow \mathbf{X}_4$.

The structure of the heat map remains fairly consistent particularly at the trough. The trough is deep which means I can be fairly certain of the velocity found in comparison to other possibilities. My least squares estimation function, `lse()`, finds the optimal set of parameters as (V_x, V_y, σ^2) = (90.576, -34.632, 0.183). Remember a negative velocity in the y-axis represents a shift upwards and so this is expected from the radar images. I use this set of parameters found between the first 2 images in combination with my models to retrieve predictions for the successive images. First looking at the performance of

my deterministic model.

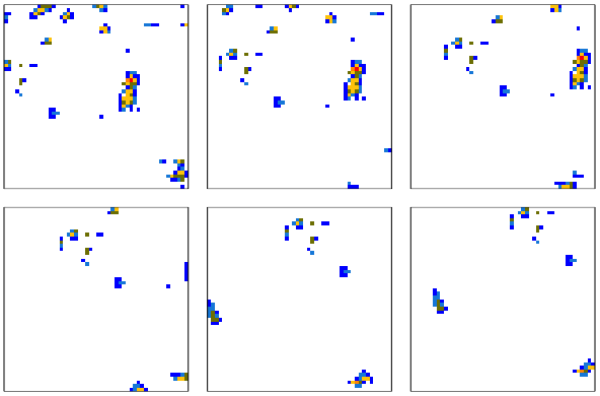


Figure 5.3 - 90 minutes of forecasted images for $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, $\mathbf{X}_5$, $\mathbf{X}_6$, $\mathbf{X}_7$ *using my deterministic model. Parameters used:* ($V_x = 90.576$, $V_y = -34.632$)

I can assess these predictions using the variety of techniques and performance indicators mentioned previously in section 5.1.

Prediction	Error	Persistence	N
$\mathbf{X}_2$	414.988	834.753	2321
$\mathbf{X}_3$	543.768	668.974	2291
$\mathbf{X}_4$	608.143	757.357	2257
$\mathbf{X}_5$	673.311	1120.595	2258
$\mathbf{X}_6$	807.803	1163.686	2260
$\mathbf{X}_7$	726.503	1094.234	2235

We see the error score increase as time goes on, which is expected as the clouds will start to change velocity and other processes like diffusion that have not been modelled will become more of a factor. Our error scores remain below the persistence which proves our deterministic model does present an improvement for all predictions up to 90 minutes into the future for this particular case. The visual strength of the model is high as N is never lower than 2235. Since there are 2500 pixels in this cutout, this means the model keeps above 89% of the pixels the right colour in this case for all forecasts. Similar to the error score we also see the visual score N generally decrease.

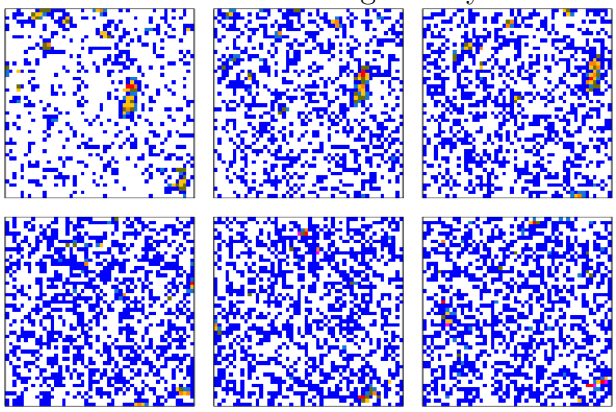


Figure 5.4 - 90 minutes of forecasted images for $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, $\mathbf{X}_5$, $\mathbf{X}_6$, $\mathbf{X}_7$ *using my probabilistic model with IID noise. Parameters used:* ($V_x = 90.576$, $V_y = -34.632$, $\sigma^2 = 0.183$)

Prediction	Error	Persistence	N
$\mathbf{X}_2$	451.802	834.753	1908
$\mathbf{X}_3$	903.877	668.974	1581
$\mathbf{X}_4$	931.724	757.357	1402
$\mathbf{X}_5$	1458.011	1120.595	1354
$\mathbf{X}_6$	4085.701	1163.686	1272
$\mathbf{X}_7$	23721.397	1094.234	1313

We see a major reduction in the visual strength of our model with N starting significantly smaller. We also see the error score increase at a much faster rate as we predict further into the future. The error score is always larger for the probabilistic model than the deterministic model and only

the X_2 prediction is smaller than the persistence. For this case the deterministic model performs much better both numerically and visually.

I also look into another example where the model performs reasonably well even though the direction of advection is not obvious.

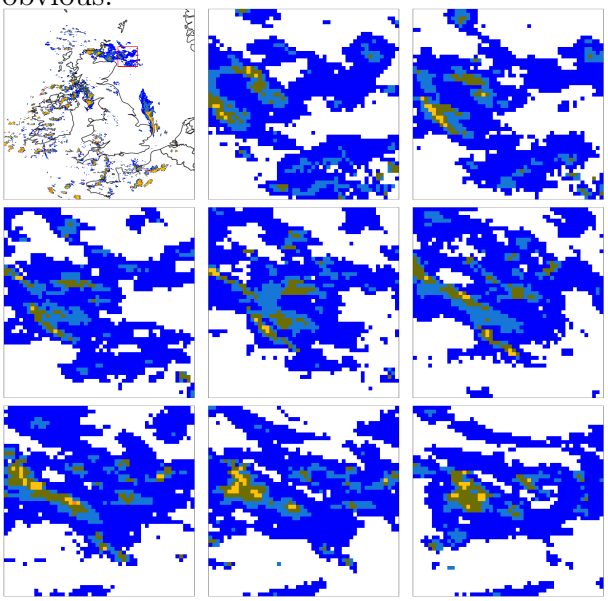


Figure 5.5 - 1 full-sized radar image with a UK underlay at t_0 and 8 50x50 cut-out radar images in the same location(red box) captured in intervals of 15 minutes(from left by row) starting from t_0. Taken on 01/01/2018 00:00-01:45.

The clouds are not as sparse as in figure 5.1 but are not as solid as in figure 2.1 due to some distortion. Examining the likelihood grid and least squares estimation heatmap for $\mathbf{X}_0$ and $\mathbf{X}_1$ we find the optimal set of parameters, $\hat{\theta} = (\hat{V}_x = 60.384, \hat{V}_y = 23.088, \hat{\sigma}^2 = 0.0743)$. We input these parameters into our deterministic model to retrieve the forecasts in figure 5.6.

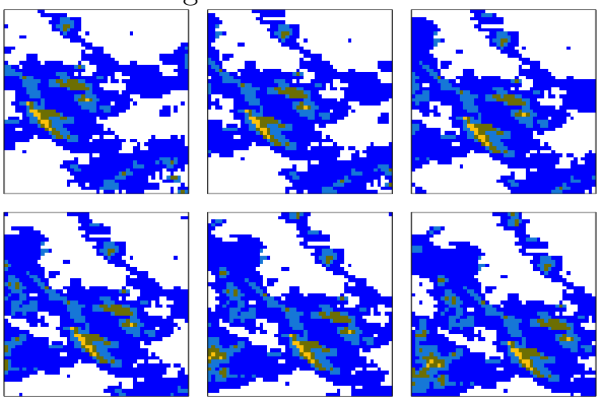


Figure 5.6 - Forecasted images for $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, $\mathbf{X}_5$, $\mathbf{X}_6$, $\mathbf{X}_7$ using my deterministic model. Parameters used: ($V_x = 60.384$, $V_y = 23.088$)

Once again I can use my performance indicators to assess these forecasts.

Prediction	Error	Persistence	N
$\mathbf{X}_2$	204.046	314.203	1655
$\mathbf{X}_3$	319.583	500.653	1295
$\mathbf{X}_4$	435.126	571.482	1140
$\mathbf{X}_5$	624.288	668.291	997
$\mathbf{X}_6$	694.971	630.727	902
$\mathbf{X}_7$	740.458	505.693	910

The interesting difference with this case is we see the error score increase past the persistence for $\mathbf{X}_6$ and $\mathbf{X}_7$, meaning the model has become ineffective when advecting past an hour into the future. We also see this case have a weaker visual strength than in figure 5.3, with N being much lower on the first prediction even though the error score starts much lower. This is because the clouds are much more sparse and there are more white

pixels in figure 5.3 so N can only be taken as a measure of performance on a case by case basis. I can also take a look at how the probabilistic model performs in figure 5.7.

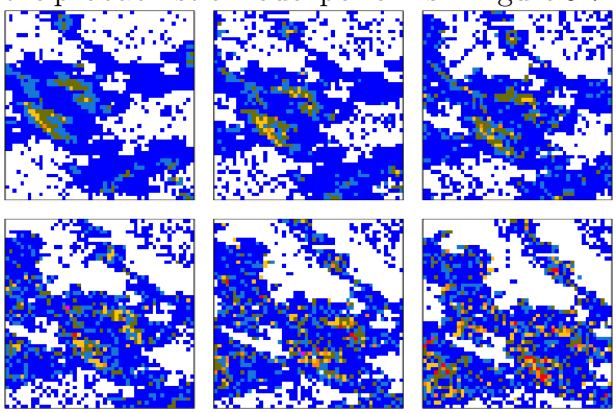


Figure 5.7 - Forecasted images for $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, $\mathbf{X}_5$, $\mathbf{X}_6$, $\mathbf{X}_7$ *using my probabilistic model with IID noise. Parameters used:* ($V_x = 60.384$, $V_y = 23.088$, $\sigma^2 = 0.0743$)

The IID noise reduces the visual strength of the model due to many white pixels turning blue. I can also calculate the performance indicators on these precipitation fields.

Prediction	Error	Persistence	N
$\mathbf{X}_2$	256.121	314.203	1606
$\mathbf{X}_3$	394.273	500.653	1172
$\mathbf{X}_4$	658.017	571.482	1020
$\mathbf{X}_5$	1388.019	668.291	952
$\mathbf{X}_6$	1967.640	630.727	829
$\mathbf{X}_7$	6273.359	505.693	815

We see that only $\mathbf{X}_2$ and $\mathbf{X}_3$ maintain an improvement on the initial image $\mathbf{X}_1$ because those error scores are lower than the persistence. Similar to the case in figure 5.4, we see the visual strength of our model starting smaller and decreasing as we predict further into the future. The error score again seems to increase faster as we advect further into the future aswell.

In general we observe the probabilistic model with IID noise perform much worse than the deterministic model after testing multiple different scenarios which was expected as adjacent pixels will be conditional on each other. I also found a drawback in the model setup when the cutout location is filled with non-white pixels which will occur often in storms like Storm Ciara depending on the size of our cutout.

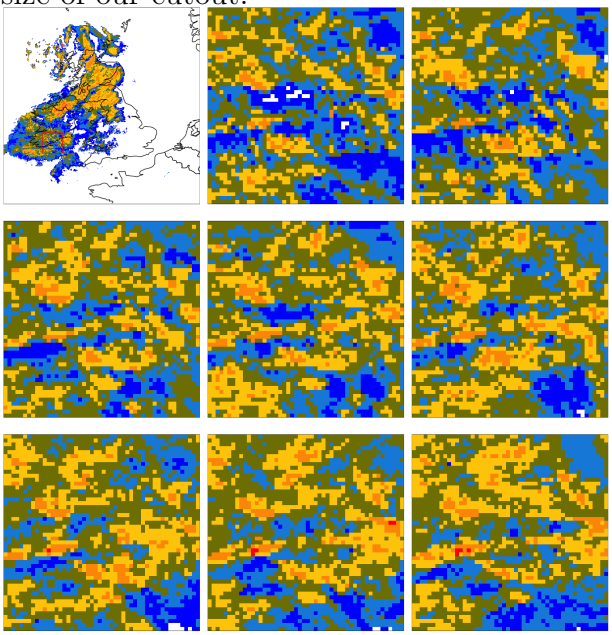


Figure 5.8 - 1 full-sized radar image with a UK underlay at t_0 *and 8 50x50 cut-out radar images in the same location(red box) captured in intervals of 15 minutes(from left by row) starting from* t_0. *Taken on 08/02/2020 between 18:30-20:15.*

No obvious advection can be seen but the images are taken in the middle of a storm and

advection can be seen in consecutive full-sized radar images. I can once again plot the error heatmaps(similar to figure 5.2) to observe various possible velocities where $\mathbf{X}_0$, $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$ are the first 5 50x50 precipitation fields in figure 5.8.

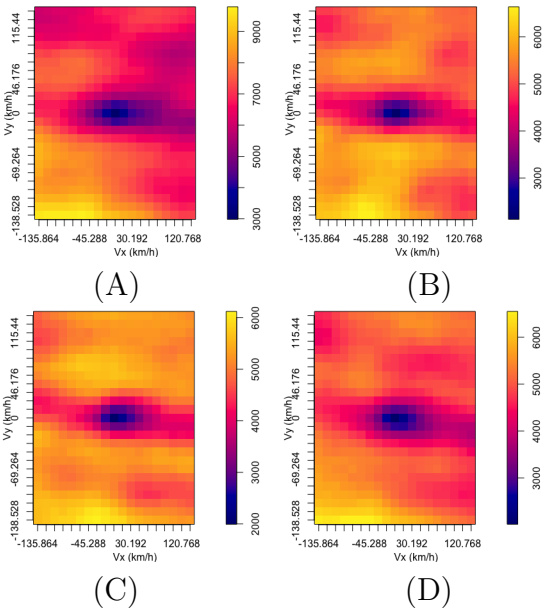


Figure 5.9 - 4 error heatmaps for the precipitation fields in figure 5.8. A covers the interval $\mathbf{X}_0 \rightarrow \mathbf{X}_1$. *B covers the interval* $\mathbf{X}_1 \rightarrow \mathbf{X}_2$. *C covers the interval* $\mathbf{X}_2 \rightarrow \mathbf{X}_3$. *D covers the interval* $\mathbf{X}_3 \rightarrow \mathbf{X}_4$.

We observe the parameter estimation consistently select a zero velocity. The parameter estimation finds the maximum likelihood/optimal velocities at $V_x = 0$ and $V_y = 0$ because the cutout location is filled. If we were to zoom out and use a larger cutout, a non-zero velocity is found although questions would arise about its accuracy as we are selecting a larger group of clouds which is more likely to have multiple groups of varying velocities. This also means our model would need tailoring for each scenario as our standard 50x50 cutout size does not hold up in storms like Storm Ciara.

I've previously taken the location of the trough or maximum likelihood as the set of parameters on which to advect the image and retrieve forecasts. I can also use the posterior predictive mean for our deterministic model which will approximated by retrieving many forecasts and averaging them(section 4.2). This would be futile for our probabilistic model as the noise would just average to zero producing the deterministic case as we increase the number of samples in our Monte Carlo approximation. For the cases in figure 5.1 and figure 5.5, the Bayesian inference function samples only 1 distinct set of parameters over 10000 samples and so the posterior predictive mean would look like our deterministic predictions above. I am going to look at an artificial case where if the model found 2 separate equally likely set of parameters.

I generate 2 artificial radar images where $\mathbf{X}_t$ has 2 separate 6x6 rain clouds of equal size and intensity and $\mathbf{X}_{t+1}$ has 1 copy of the 6x6 rain cloud like in figure 6.0.

Figure 6.0 - 2 artificial radar images, $\mathbf{X}_t$ (left) and $\mathbf{X}_{t+1}$ (right).

Laying out the clouds in this way means that there will be 2 equally likely strong possibilities and the sampler should output each set of parameters half of the time. Looking at the velocity cross section of the likelihood grid (like in figure 4.2a) we observe 2 strong possibilities in figure 6.1.

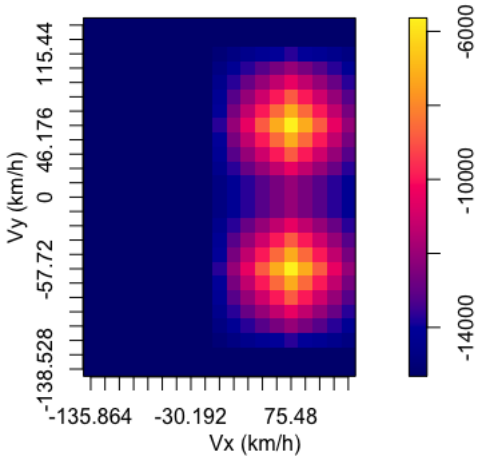


Figure 6.1 - A cross section plot of the likelihood for values of V_y against V_x. Cross section taken at $\tau = \hat{\tau}$.

My `bayesinf()` function then samples the following 10 sets of parameters from the likelihood function when m is set to 10:

Vx	Vy	Variance
75.48	-57.72	1.221403
75.48	-57.72	1.221403
75.48	-57.72	1.221403
75.48	-57.72	1.221403
75.48	57.72	1.221403
75.48	57.72	1.221403
75.48	-57.72	1.221403
75.48	57.72	1.221403
75.48	57.72	1.221403
75.48	57.72	1.221403

Figure 6.0 - 10 samples from the Bayesian inference function on the artificial precipitation fields.

As expected we see 2 unique sets of parameters both sampled 5 times as they are equally likely. Now I use my `ppm()` function to retrieve an approximation for the posterior predictive mean for $\mathbf{X}_{t+2}$ in figure 6.2.

Figure 6.2 - 2 artificial radar images, $\mathbf{X}_{t+1}$ (left) and the posterior predicitve mean for $\mathbf{X}_{t+2}$ (right).

The posterior predictive mean split the cloud into 2 with half the intensity on each which is an unlikely scenario as we estimate the cloud either advected with velocities $V_x = 75.48$, $V_y = 57.72$ or $V_x = 75.48$, $V_y = -57.72$. This is a problem with taking the mean in general and so in cases like this it may be preferable to take the mode precipitation intensity of each pixel as an approximation for the expectation of the posterior predictive distribution. Also since the shape of our heatmaps usually stay semi-consistent for successive time intervals, 1 of the 2 possible parameter

sets may become less likely as we observe more radar images which we could use as an indicator for the more realistic velocity if this artificial case were to happen in reality.

In general we find the deterministic model performs much better numerically and visually than our probabilistic model with IID noise. We found problems occur during storms or whenever the model cutout is filled, as the model will usually select a zero velocity on both axis. Our model generally performs well with sparse clouds, like in figures 5.1 and 5.5, and performs well in almost perfect scenarios like in figure 2.1. In this section we've picked out specific cases and reviewed how our model responds, however an abundance of radar data exists and finding a systematic way to evaluate the model using all this data would be of value in the future.

6 Conclusion and further work

This study has shown how a simple advection equation can be used to produce radar image predictions based on prior images. We have followed basic mathematical modelling procedure in making assumptions based on data in order to formulate our model. This report covered the intricate mathematics involved in converting a partial differential equation to a linear map whilst taking into account various factors like the stability of our finite difference scheme and numerical diffusion. The report also covered the statistical tools that could analyse the data to retrieve specific parameters required in our model. We found the successive heatmaps produced remained fairly consistent which aided the strength of our models predictions.
Whilst our deterministic model performed reasonably well for a prediction 15-90 minutes into the future, we found the probabilistic model performed poorly in almost all cases. The IID noise was based on an incorrect assumption that each pixel is independent and so this led us to observe poor performance from the model. Occasionally we found that the deterministic model was ineffective towards the end of our prediction window like in figure 5.6 where we saw the error score increase past the persistence at 75-90 minutes.

Further work on the model could reduce error scores by accounting for additional processes. For example the addition of a diffusion process would be quite straightforward. Consider the linear advection equation(equation 1 in section 3.2.1), then the advection-diffusion equation would take the form:

$$\frac{\partial \mathbf{X}}{\partial t} + V_x \frac{\partial \mathbf{X}}{\partial x} = K \frac{\partial^2 \mathbf{X}}{\partial x^2}$$

where K is our diffusion constant and all the other variables are the same. We could also complete the implementation of our spatially correlated noise into our model by reformulating our parameter estimation techniques. This could form a better comparison to our deterministic model than the IID probabilistic model.

There is an abundance of data when it comes to radar images and so it could be useful to find a way to take advantage of it. I found the paper in reference [3] particularly interesting which, as mentioned in the introduction, uses a deep learning approach to nowcasting by learning from large datasets. This could form a good comparison for my model and it would be interesting to see how the difference changes as I include more processes in my model. Similarly the MAPLE algorithm, found in reference [4], uses a similar framework to our deterministic model and so it would be interesting to see how the MAPLE algorithm reacts to the problems my model experienced particularly with storm data.

Appendix

All code and data used to reproduce the results with details on required packages can be found at `https://github.com/sahilkariadata/track_rain`.

References

[1] WMO.INT Webteam. Nowcasting. `https://www.wmo.int/pages/prog/amp/pwsp/Nowcasting.htm`.

[2] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[3] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Deep learning for precipitation nowcasting: A benchmark and a new model. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5617–5627. Curran Associates, Inc., 2017.

[4] Urs Germann and Isztar Zawadzki. Scale-dependence of the predictability of precipitation from continental radar images. part i: Description of the methodology. *Monthly Weather Review*, 130(12):2859–2873, 2002.

[5] Svein Linge and Hans Petter Langtangen. *Advection-Dominated Equations*. Springer International Publishing, Cham, 2017.

[6] Met office datapoint. `https://www.metoffice.gov.uk/services/data/datapoint`.

[7] Met Office. Devon's weather radar can now detect the size and shape of rain and snow, Feb 2018. `https://www.devonlive.com/news/devon-news/devons-weather-radar-can-now-1188994`.

[8] Rainfall radar. `https://www.metoffice.gov.uk/weather/learn-about/how-forecasts-are-made/observations/rainfall-radar`.

[9] Fact sheet 15 - weather radar. `https://www.metoffice.gov.uk/research/library-and-archive/publications/factsheets`.

[10] S J Driscoll D L Harrison and M Kitchen. Improving precipitation estimates from weather radar using quality control and correction techniques [unpublished document].

1998. http://cedadocs.ceda.ac.uk/293/1/nimrod_radar_processing.pdf.

[11] Met Office. Uk radar rainfall map, May 2014. https://www.metoffice.gov.uk/public/weather/observation/rainfall-radar#?map=Rainfall.

[12] DM Causon and CG Mingham. *Introductory finite difference methods for PDEs*. Bookboon, 2010.

[13] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2019. R package version 1.2-18.

[14] Remineur. Stability of upwind scheme with forward-euler time integration, Jul 2013. https://showmethemath.wordpress.com/2013/07/07/stability-of-upwind-scheme-with-forward-euler-time-integration/.

[15] Bram van Leer. On numerical dispersion by upwind differencing. *Applied Numerical Mathematics*, 2:379–384, 10 1986.

[16] Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2015.

[17] Havard Rue and Leonhard Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[18] Kovit Danby. Probabilistic models for images markov random fields, Oct 2014. https://www.slideserve.com/kovit/probabilistic-models-for-images-markov-random-fields.

[19] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[20] J. S. Marron and M. P. Wand. Exact mean integrated squared error. *Ann. Statist.*, 20(2):712–736, 06 1992.