Graph Signal Processing

Zac Petersen

School of Electrical Engineering and Telecommunications

Submitted October 22, 2025

Abstract

Graph signal processing is a dense field of research that covers a wide array of problems. Much of its development is built upon the application of traditional signal processing techniques to the graphical domain, but much development is unique entirely to graphical data. Many such techniques have been developed across disciplines; statistics, geography, computer science, discrete mathematics, and signal processing. They apply to a number of problems; compression, prediction, filtering, and computer vision. This report is a preliminary review of a variety of work in these fields and problems over the past century. It works to critique, expand on, link, and reproduce the results of a variety of these works. More broadly, this thesis will aim to produce tangible evidence of the applicability of graph signal processing to a variety of real-world problems, as well as provide novel explanations for their applicability. It will further develop generic techniques for solving problems using graph signal processing problems.

Acknowledgements

I would like to thank my supervisor, Professor V. Solo, for pushing me in the right direction, exposing me to appropriate literature, and offering insights orthogonal to my own.

Abbreviations

FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
GFT	Graph Fourier Transform
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
WSS	Wide Sense Stationary
PCA	Principle Component Analysis
SNR	Signal to Noise Ratio
SGD	Stochastic Gradient Descent
BIC	Bayesian Information Criterion
AR	Autoregression
VAR	Vector Autoregression

Contents

1 Introduction

Graphs were first introduced into the literature through Euler's solution to the Königsberg bridge problem in 1741 [6]. Through the ages, graph theory became relevant to a variety of problems including circuit theory, chemistry, and eventually computer science. Many of these original problems are naturally represented by graphs, but many problems on continuous spaces are simplified by endowing them with a graph structure. One of the earliest such examples is the statistical technique invented by Krige to estimate mineral deposit concentration based on a limited number of samples from boreholes [12]. Despite the age of this field, techniques are not well known, sometimes reinvented across different fields in different forms, and have not uniformly been assessed in practical circumstances.

The present literature has established techniques for compression [16], prediction [12], filtering [26], and computer vision [9] on graphs. Such techniques have been relevant to highly influential areas, including advertising and ranking as in the Google PageRank algorithm [19], risk and stability analysis in econometrics [1], and the application of graph techniques to neural networks as has been used in drug discovery and development [24]. We will explore examples and replicate some of these techniques.

In this report, we replicate techniques and empirically confirm results both with synthetic data sets and a real data set. Two real datasets are used: homogenised daily mean temperature data across Australia as reported by the Bureau of Meteorology (BOM) [18], and daily mean temperature data over north America, extracted as a subset of the global surface temperature data provided by the National Oceanic and Atmospheric Administration of the United States (NOAA) [17]. The BOM dataset is broadly used for developing and exhibiting methods covered, whilst the NOAA dataset is used to validate the broad utility of models created, as well as to investigate how model parameters change over different samples of the same type of data (temperature networks).

2 Background

2.1 An Introduction to Graphs, Graph Signals, and Graph Spectra

A graph is a collection of vertices, $\mathscr{V}$, and edges (pairs of vertices), $\mathscr{E}$[1]. In some applications, edges may be ordered pairs (directed graphs), but for the purposes of this introduction we consider unordered pairs (undirected graphs). Graph structure is often encoded in an adjacency matrix. For a graph with n vertices, the adjacency matrix is the $n \times n$ matrix A such that $A_{ij} = w_{ij}$, the weight of the edge connecting vertices v_i and v_j, or 0 if there is no such edge.

A graph signal is an association of either each vertex $v \in \mathscr{V}$, or each edge $e \in \mathscr{E}$ with some value. This value could be a scalar, vector, or some other value. In spatiotemporal applications, as are addressed in the following sections, each vertex is associated with a time series. For this introduction, we consider scalar data associated with each vertex.

Graph spectra have been covered extensively in the literature [16][21]. To justify graph spectra, we first cover one reason why the typical Fourier transform is useful. In the continuous domain, the Fourier transform is a mapping from functions in the trivial "shifted-deltas" basis into an orthonormal eigenbasis of the derivative operator (complex exponentials). Because derivatives occur extensively in nature, this proves to be a useful decomposition. Additionally, because of the FFT, the DFT can be implemented quickly, in $O(n \log n)$ time.

A key difference between a graph domain and a discrete time domain is that time has a direction. The first order derivative in time, $\frac{d}{dt}$ measures output changes for increasing time. Because there is no specified "increasing" direction, this is not helpful on an undirected graph. We may turn to multivariate calculus to search for isotropic (directionless) scalar to scalar differential forms. The most obvious is the Laplacian ∇^2, which in the time domain (equivalent to $\frac{d^2}{dt^2}$) has sines and cosines as its eigenfunctions. The integral form of the Laplacian is given below.

$$\nabla^2 f(\underset{\sim}{x}) = \lim_{h \to 0} \frac{2n}{S_n h^{n+1}} \int_{\partial B_{n,h}} f(\underset{\sim}{y}) - f(\underset{\sim}{x}) d\underset{\sim}{y}$$

For S_n the surface area of a unit n-ball, and $B_{n,h}$ an n-ball with radius h. Up to scaling this is a comparison of the values around a point and the value at the point, measuring curvature. In the context of graphs, we associate each edge with some weighting, w_{ij}, and assume that some graph signal approximates a signal in a continuous space, where the weighting measures "similarity" or "closeness" of vertices. When the weight is the reciprocal of distance, the negative

[1] Graphs are made up of 0-dimensional components (vertexes) and 1-dimensional components (edges). Generalised graphs with up to n-dimensional components are introduced as n-complexes, and addressed in great depth in Grady, 2010 [10].

Laplacian is approximated by the Laplacian matrix, defined below.

$$L_{ij} = \begin{cases} \sum_{e_{ik} \in \mathscr{E}} \frac{1}{w_{ik}} & \text{if } i = j \\ -\frac{1}{w_{ij}} & \text{if } i \neq j \end{cases}$$

Note that in this context graph data may be represented by a vector $\mathbf{x}$ of the signal value at vertex v_i. We then define the GFT as the transformation that takes a graph signal into an orthonormal basis for the Laplacian. Again notice that when operating on one point (e.g. $v = (1\ 0\ ...\ 0)^\top$, this compares the value at some vertex to the values at adjacent vertices, up to scaling. Since in the time domain $\nabla^2 \cos(\omega t) = -\omega^2 \cos(\omega t)$, and the Laplacian matrix represents the Laplacian operator up to a negative constant, the eigenvalues of the equation $\mathbf{Lx} = \lambda \mathbf{x}$ are said to correspond to a frequency (or spatially, wavenumber) of $\sqrt{\lambda}$. That is, the units of λ are the inverse square of the units of the edge weights [21]. Given the decomposition $L = V \Lambda V^{-1}$, the GFT is equivalent to V^{-1}.

The GFT can be derived in another way, again adjacent to traditional signal processing. Instead of being framed in terms of the derivative, as in the continuous case, the DFT can be described as a mapping into the eigenbasis of the shift operator, z^{-1} [16]. We can represent the discrete time domain in this context as a circular domain, where the shift moves a unit impulse around the loop. The relevant directed graph for the time domain is shown in Fig. 1. The derivation of the GFT in this context simply requires the definition of a shift operator. Typically, choices of a shift operator will involve spreading an impulse at a vertex between neighbouring vertices, as in Fig. 2.

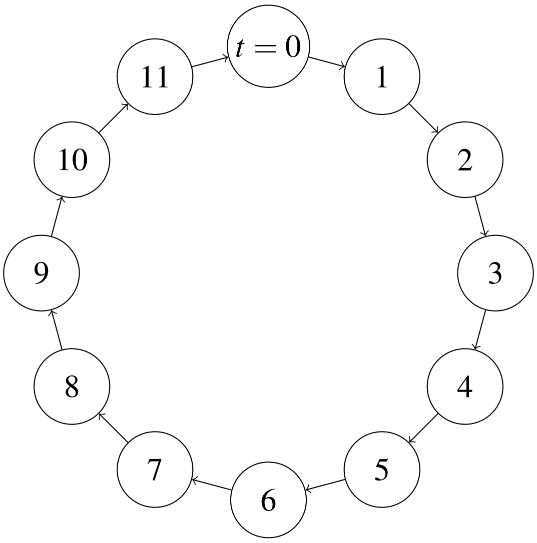

Figure 1: A graph for a time-domain with 12 samples.

Note that these two methods provide entirely different definitions of the GFT. This is common throughout the literature, and there are yet more ways to define the graph Laplacian.

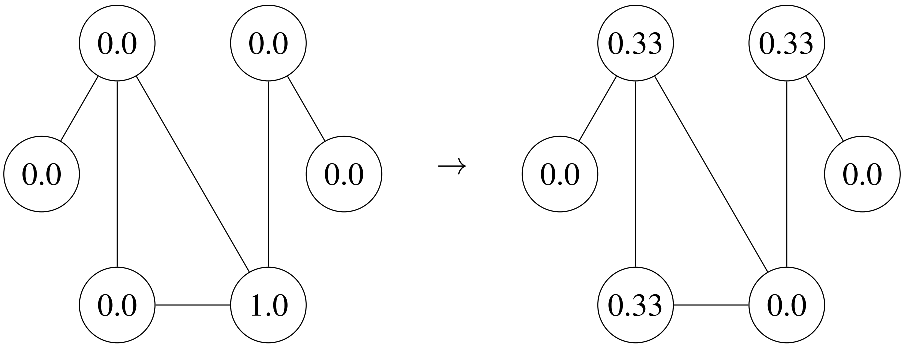

Figure 2: A impulse spreading to nearby nodes under a particular shift operator.

3 Literature Review

3.1 Graph Products and Fast GFTs

Part of the ubiquitous utility of the DFT across data science, algorithms, and engineering, is that it may be sped up significantly from naïve $O(n^2)$ time to $O(n \log n)$ time by the FFT. Underpinning the FFT is the division of the DFT into the DFT into a number of components. In particular, the speed up relies upon the splitting of the domain into subdomains for each prime factor of the size of the DFT [4]. The most common FFT algorithm, the Cooley-Tukey algorithm, offers no speed up for a DFT over a time domain whose length is prime.

We may hope then, that we can speed up the GFT for many graphical domains. Moura, 2014 [16], links products of domain sizes for the DFT to graph products of time-domain graphs as was described in 2.1. Spatiotemporal data is shown to be representable as a graph product, such that not only do values at future time steps depend on the previous time step at the same location, but also at neighbouring locations in the previous time step via the strong graph product, as in Fig. 3. They also show that across three choices of graph product, the overall GFT is the same,

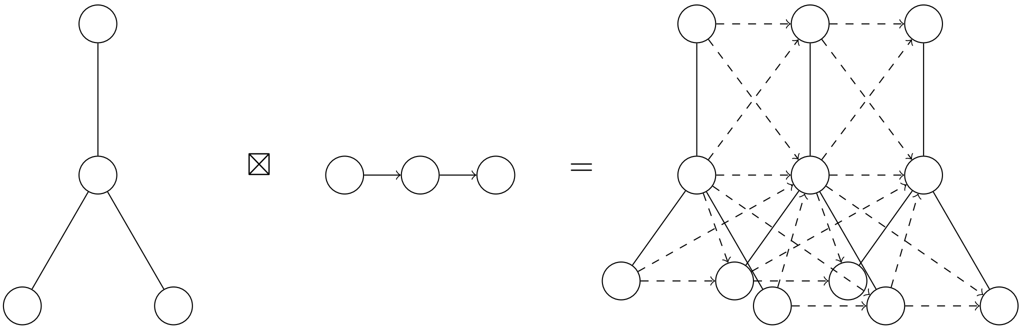

Figure 3: The strong product of two graphs.

only the eigenvalues change. For the two factor graphs with GFTs A and B, the product graph has GFT C as below, where $\otimes$ is the Kronecker product of matrices.

$$C = A \otimes B$$

Note that for moderately sized graphs, this Kronecker product results in an exorbitantly large space complexity. For graph factors with a and b vertices respectively, the GFT matrix C is a^2b^2. So this factorisation helps enormously, reducing the space requirement to $a^2 + b^2$. Finding the GFT of a graph with n vertices naïvely requires $O(n^3)$ operations to perform an eigendecomposition of a shift matrix. Moura's method reduces the time complexity in this case from $O(a^3b^3)$ to $O(a^3 + b^3)$, a massive saving. Moura does not describe how to perform this GFT quickly, without the need to use the massive C matrix, which gives $O(a^2b^2)$ time. This limitation is addressed in 4.2 In fact, the GFT can also be performed faster with this method. The computation time T_C can be reduced to $bT_A + aT_B$. If no more factorisation is available, this gives a complexity of $O(ba^2 + ab^2)$, a speed up of $O(a+b)$. Moura notes this speed up for filtering operations, but not for the GFT itself. The matrix-vector product can be represented as below [13].

$$(A \otimes B)\mathbf{x} = BXA^\top$$

Where the matrix X represents horizontally stacking a chunks of $\mathbf{x}$ each of size b. The left matrix-matrix product is $O(b^2a)$, and the right $O(a^2b)$. This yields an intuitive method for computing the GFT. Since the graph product makes a copy of the graph B for every vertex of A, we first run B's GFT on each copy of B, and then each eigenvector of B's spectra has a component for every vertex of graph A. For each eigenvector, we run A's GFT. In the spatiotemporal case, where we take the graph product of a time domain and a spacial domain, this involves taking the FFT of every vertices' time series, and then taking the GFT for each frequency. For time series of length a, and graph with b vertices, this speeds the process up beyond the once-factorable speed to $O(ab^2 + ba\log a)$. For small graphs and long time series this is a fairly efficient algorithm.

An important question then is how often are graphs factorable? Some key graphs are factorable. Spatiotemporal data, as we saw, lattices, as in image processing, and toroids, as in the higher-dimensional DFT, are all factorable. It has been shown that, under the Cartesian graph product, factorable graphs can be factorized in linear time [11]. A distribution of random graphs can be expressed as a distribution of adjacency matrices. Taking the eigenvectors of these matrices adds an additional degree of randomness. Supposing each entry of each matrix can take on one of F values, given the space complexity requirements, we can propose a crude approximation for the probability of kronecker matrix factorability.

$$P(\exists A, B : C = A \otimes B) \approx \frac{F^{a^2+b^2}}{F^{a^2b^2}}$$

We expect there is a vanishingly small probability of finding a factor in the case of random graphs. Approximate solutions to the equation, that is the nearest-kronecker-product problem have been discussed as a method for dimensionality reduction [13], which could prove a useful tool for approximating the GFT in large graphical contexts.

3.1.1 Application to Compression Problems

Data compression is a fairly universal problem, as data storage is expensive. One method for lossy data compression using the DFT in traditional signal processing involves taking the DFT of a signal, and discarding low-magnitude components. Because eigenvectors describe global structure of the domain, often most of the signal is contained within relatively few eigenvectors. JPEG compression, which is a commonly used image compression technique performs a similar process, using the discrete cosine transform (DCT) rather than the DFT [32].

Moura applies the fast GFT to compress spatiotemporal daily temperature data across the United States for 150 stations across 1 year (365 measurements) [16]. A subset of the coefficients of the spatiotemporal GFT eigenvectors are used, selecting the coefficients with largest magnitude. Their results are shown in Fig. 4.

Fraction of Coefficients Used	1/50	1/20	1/15	1/10	1/7	1/5	1/3
RMSE (%)	4.9	3.5	3.1	2.6	2.1	1.6	0.7

Figure 4: US Data Daily Temperature (2002) Compression Results

Moura does not discuss how much of this compression can be attributed simply to time-domain compression, and how much is due to graph-domain compression. Methods for evaluating this are discussed in the preliminary work in 4.2.

Storing sparse coefficients like this however requires knowledge of the eigenvectors corresponding to each coefficient, as they are not so structures as the eigenvectors of the DFT. This requires storage costs of n^2 floats for n weather stations, which in Moura's case represents $150^2/(150 \cdot 365) \approx 41\%$ of the original storage cost. Alternatively, the decompression process can involve finding the GFT matrix through storing the associated graph metadata. I.e. the edges, their weights (or how these were found from for example, location data). The paper assigns edges via a nearest-neighbour scheme, and weights through equation (29) in their preceding paper [15], modified for clarity below. Note d denotes the distances between measurement points, and σ denotes a scaling factor.

$$w_{ij} = \frac{e^{-d_{ij}^2/\sigma^2}}{\sqrt{\sum_{k:e_{ik}\in\mathscr{E}} e^{-d_{ik}^2/\sigma^2} \sum_{k:e_{jk}\in\mathscr{E}} e^{-d_{jk}^2/\sigma^2}}}$$

The choice of nearest-neighbour scheme and the use of a Gaussian kernel rather than some other is unjustified, except to say they their use is common and has shown some success. We will justify other kernels later when discussing the Matérn kernel in 3.4.1, and will point out the success of the Gaussian in 4.5.3.

3.2 Graph Windowing and Convolution with the GFT

In time-domain convolution first arises in filter application. A linear time-invariant filter can always be expressed as a convolution of a time signal $x[n]$ with an impulse response $h[n]$ as below.

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Of course, in the graph domain, n and k are vertices, and there is no natural subtraction of vertices. One intuitive option is to call $n-k$ the shortest distance between the vertices. The shortest distance between two vertices in a graph may be found via Dijkstra's algorithm [30] in $O((V+E)\log E)$, or all distances may be found via the Floyd-Warshall algorithm [31] in $O(V^3)$ (for V the number of vertices and E the number of edges). Another prominent method, highlighted by Stankovic et. al. [22], amongst others, is to note one of the most important properties of time-domain convolution, frequency domain multiplication. In the frequency domain, convolution becomes an elementwise multiplication. So, we can define convolution as follows.

1. Take the GFT of $x[n]$ and $h[n]$ to get $X[k]$ and $H[k]$.
2. Multiply to get $\hat{X}[k] = X[k]H[k]$.
3. Apply the IGFT to get $\hat{x}[n] = x[n] * h[n]$, the convolution.

Naturally, this requires eigendecomposing a graph operator to get the GFT and IGFT, which as discussed in 3.1 can be slow when a graph has many vertices.

As discussed in 2.1, a graph shift operator is an analog for the time shift operator. That is, just as an impulse response can be represented in terms of its Z-transform, which via the convolution formula allows for filter application, some graph filters $\boldsymbol{H}$ can be represented in terms of the graph shift operator $\boldsymbol{A}$, as below.

$$\boldsymbol{H} = \sum_{n=-\infty}^{\infty} h_n \boldsymbol{A}^n$$

In particular, a linear graph shift operator most generally is any matrix that applies to the graph signal vector. The graph filters that can be represented in terms of the graph shift operator are in some sense ”space invariant”, as a time-domain filter may be ”time invariant”.

In traditional signal processing, more complex data analysis is performed through the use of the short-time Fourier transform, which allows for analysis of frequency data over time. Because the natural domain of the DFT is a toroidal space, that is the beginning is in some sense ”attached” to the end, due to the aliasing process of digital sampling, spectral leakage occurs, so windowing is used to provide a more useful estimate of frequency data. Windowing involves elementwise multiplying a time domain signal by a window function, say $w[n]$. Another helpful thing about windowing is that we need not process a whole dataset. Although this is less of a problem in the time domain, as the FFT is fast, in the graph domain, a graph with say a million vertices would be near impossible to eigendecompose. As such, windowing graph data to a local area would be immensely helpful. Stankovic et. al. describe a more ideal circumstance, where the GFT is computed and the windows are defined from there, which allows for local-space analysis. Nevertheless, the option for less ideal windowing remains.

Stankovic et. al. also describe the implementation of graph bandpass filters, to target a particular range of the eigenspectrum, via the chebyshev set of filters, modified via the expansion above to use the graph operator. It is important to note that although general dense matrix multiplication is $O(n^3)$, for sparse matrices, as in the graph operators we have described so far, multiplication is much faster for relatively low powers of $\boldsymbol{A}$, as in $\boldsymbol{A}^k$, $k << n$. In 4.5.2 we discuss an good choice of a low-pass filter on a particular graph operator, though efficient implementation is still difficult. We also discuss how a graph operator should be chosen, and link the low-pass graph filter to the minimum MSE optimal noise reduction filter, the Wiener filter. The theory for the Wiener filter in the time domain is discussed in 3.5.2.

3.3 Temporal Stochastic Processes; Predictive Filters

When assessing the utility of spatial modelling for the analysis of spatiotemporal data, it must be evaluated against the use of temporal modelling alone. Temporal modelling is ubiquitous, as being able to predict the future is almost universally a lucrative enterprise. Additionally, temporal modelling provides many techniques that can be applied with modification in the spatial domain. For these reasons, we will introduce temporal stochastic processes and predictive filters.

A stochastic process X is a collection of random variables $X_t \forall t \in \mathbb{R}$. We may have a discrete time domain if instead $t \in \mathbb{Z}$. Because time only flows in one direction, X_t can only depend upon random noise, as well as $X_\tau \forall \tau \leq t$.

A stochastic process X is considered stationary if the distribution of X_t is independent of t. Such a process is further called wide-sense stationary (WSS) if $\mathrm{Cov}(X_t, X_{t-\tau})$ is independent of t [37].

In the discrete domain, linear regression (discussed in 3.5) can be used to regress a signal against its past values. This gives an optimal (in the MSE sense) linear deterministic predictive filter.

Examples of more sophisticated filters include the Wiener and Kalman filters, both adaptive (non-deterministic) filters [36].

As a note on the appropriate application of linear regressive models, we recall the first-order model for some signal T and noise ε.

$$\begin{aligned} T_t &= \xi_0 T_{t-1} + \varepsilon_t \\ &= \xi_0^2 T_{t-2} + \xi_0 \varepsilon_{t-1} + \varepsilon_t \\ &\dots \\ &= \xi_0^N T_{t-N} + \sum_{i=0}^{N-1} \xi_0^i \varepsilon_{t-i} \end{aligned}$$

This means that T can be given as a sum of random variables ε_τ drawn from the same random distribution, along with a scaling factor. When autoregression explains the data well, $\xi_0 \approx 1$, so the central limit theorem approximately applies, making T more normally distributed. When the $\xi_0 = 0$, T is drawn from the same distribution as ε_τ. As such, autoregression tends to be more successful on Gaussian data. Higher order filters can address more complex distributions. Another set of important filters are autoregressive moving average (ARMA) filters, but for brevity we will not discuss these.

3.4 Continuous-Domain Spatial Stochastic Processes

Temporal stochastic processes discuss random signals X_t dependent on a time $t \in \mathbb{R}$. Spatial stochastic processes discuss random signals $X_{\underset{\sim}{x}}$ dependent on a position in space $\underset{\sim}{x} \in \mathbb{R}^n$. Spatial stochastic processes occur throughout nature. Geostatistical processes, like the distribution of trees, hills, and other geographic features are examples of a single realisation of a spatial stochastic process. Just like in the time domain, the ability to predict/interpolate spatial features allows for significant cost savings in sampling.

Just as in the discrete domain we constrict t to $t \in \mathbb{Z}$, space is often discretised either via the lattice $\mathbb{Z}^n$, or by a graph. Just as time is sampled for practical reasons, so too is the continuous spatial domain often sampled. Spatial discretisations often come in the form of graphs. To do analysis effectively on graphs, it will serve us well to understand the analytical results from continuous-domain spatial stochastic processes.

3.4.1 The Matérn Kernel

Matérn (1960) discusses stationary spatial stochastic processes [3]. Here, stationarity means the distribution looks identical around any point in space. In particular, Matérn shows that for a natural extension of type-III probability density functions (a popular broad class of probability density functions including Normal distributions and other common distributions) onto spatial processes, the correlation function between points $\underset{\sim}{x}$ and $\underset{\sim}{y}$ is given as below.

$$\rho(\underset{\sim}{x},\underset{\sim}{y}) = \text{const. } ||\underset{\sim}{x}-\underset{\sim}{y}||_2^{\nu} K_{\nu}(\lambda||\underset{\sim}{x}-\underset{\sim}{y}||_2)$$

For parameters ν and λ, and K_{ν} the modified bessel function of the second kind with order ν. This kernel is isotropic, so that its covariance is radially symmetric, because it only depends on the distance between $\underset{\sim}{x}$ and $\underset{\sim}{y}$.

Whittle (1954) finds a case where the Matérn kernel naturally arises [34]. Working from the discrete domain, he arrives at an equation equivalent to the one below.

$$(\nabla^2 - \lambda^2)X_{\underset{\sim}{x}} = \varepsilon_{\underset{\sim}{x}}$$

For $\varepsilon_{\underset{\sim}{x}}$ uncorrelated white noise. We revisit this model in the preliminary work in 4.4.1. In two dimensions ($\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$) he found the correlation function to be a Matërn kernel with $\nu = 1$, as below.

$$\rho(\underset{\sim}{x},\underset{\sim}{y}) = \lambda||\underset{\sim}{x}-\underset{\sim}{y}||_2 K_1(\lambda||\underset{\sim}{x}-\underset{\sim}{y}||_2)$$

In his later work, he found the correlation function in all dimensions, except $n = 1$, which is addressed in the preliminary work in 4.4.3 [35]. For $r = ||\underset{\sim}{x}-\underset{\sim}{y}||_2$, the correlation function ρ is as below.

$$\rho(r) = \frac{2^{n/2-1}}{\Gamma(2-n/2)}(\lambda r)^{2-n/2}K_{2-n/2}(\lambda r)$$

This model breaks down when $n > 4$. Intuitively, the diffusion equation in 5 or more dimensions has in some sense too many dimensions to diffuse through, so that variance at each point goes to infinity, as noise throughout the space increases the signal variance at each point.

3.4.2 Kriging

Kriging is a practical process for spatial interpolation, given a covariance function between points in a space, and a number of samples throughout the space of a realisation of the stationary stochastic process. Kriging was invented by the eponymous Krige, originally proposed for estimation of gold content in the Witwatersrand in South Africa for mine site evaluation [12]. Since then it has been used across many areas of geostatistical estimation, and also across other many other fields, just one of which is hyperparameter estimation for machine learning methods [28].

There are many methods for empirical estimation of a spatial covariance function, but for brevity we will here assume the covariance function is known. The simplest form of Kriging is simple Kriging, and is equivalent to linearly regressing the data at the interpolation point against the sample points, as discussed in 3.5.1. In particular, we consider the spatial process $X_{\underset{\sim}{x}}$, sampled at k points $\underset{\sim}{x}_i$, to be interpolated at a point $\underset{\sim}{y}$. If the sample points have covariance matrix between each other $\mathbf{\Sigma}$, and the covariances between the $\underset{\sim}{x}_i$ and the interpolation point $\underset{\sim}{y}$ are given in the vector $\underset{\sim}{\Gamma} : \Gamma_i = \mathrm{Cov}(X_{\underset{\sim}{x}}, X_{\underset{\sim}{y}})$, then the interpolation $\hat{X}_{\underset{\sim}{y}}$ is given as below.

$$\hat{X}_{\underset{\sim}{y}} = X_{\underset{\sim}{x}}^{\top} \mathbf{\Sigma}^{-1} \underset{\sim}{\Gamma}$$

For the vector of samples $X_{\underset{\sim}{x}}$. Because the spatial process is stationary, if its mean is μ, we can check the bias of the prediction below.

$$\begin{aligned} E(\hat{X}_{\underset{\sim}{y}}) &= E(X_{\underset{\sim}{x}}^{\top}) \mathbf{\Sigma}^{-1} \underset{\sim}{\Gamma} \\ &= \mu \sum_{i=0}^{k} (\mathbf{\Sigma}^{-1} \underset{\sim}{\Gamma})_i \end{aligned}$$

So that the process is biased. Finding the least-squares unbiased estimator leads to the ordinary Kriging estimate. The formula below presents the solution, with μ a Lagrange multiplier that can be discarded.

$$\begin{aligned} \begin{pmatrix} \underset{\sim}{\hat{w}} \\ \mu \end{pmatrix} &= \begin{pmatrix} \mathbf{\Sigma} & \mathbf{1} \\ \mathbf{1}^{\top} & 0 \end{pmatrix}^{-1} \begin{pmatrix} \underset{\sim}{\Gamma} \\ 1 \end{pmatrix} \\ \hat{X}_{\underset{\sim}{y}} &= X_{\underset{\sim}{x}}^{\top} \underset{\sim}{\hat{w}} \end{aligned}$$

This is the estimate implemented in the preliminary work in 4.4.5.

The Kriging error is the expected RMSE of the interpolation at each point. In the Ordinary

Kriging case it can be computed using the formula below.

$$\mathrm{Var}(\hat{X_0} - X_0) = \begin{pmatrix} \underset{\sim}{\hat{w}}^\top & -1 \end{pmatrix} \begin{pmatrix} \mathbf{\Sigma} & \underset{\sim}{\Gamma} \\ \underset{\sim}{\Gamma}^\top & \mathrm{Var}(X_0) \end{pmatrix} \begin{pmatrix} \underset{\sim}{\hat{w}} \\ -1 \end{pmatrix}$$

3.5 Least Squares Estimation

3.5.1 Linear Regression

Linear regression is one of the simplest and most powerful data fitting techniques. It is often credited to Adrien-Marie Legendre and Carl Friedrich Gauss in their work on predicting planetary movements [23]. It is common due to its low complexity in terms of the amount of data being processed, being $O(n)$ for n datapoints per feature, and ease of interpretibility due to the simplicity of linear modelling. We cover standard results seen in most statistics textbooks, like [7]

Linear regression seeks to model some set of l outputs $\underset{\sim}{y} \in \mathbb{R}^l$, each observed n times and stored in the matrix $\boldsymbol{Y} \in \mathbb{R}^{n\times l}$. It is modelled in terms of a set of k features $\underset{\sim}{x} \in \mathbb{R}^k$, each of which have again been observed n times and stored in the matrix $\boldsymbol{X} \in \mathbb{R}^{n\times k}$. The relationship is modelled linearly as below.

$$\boldsymbol{Y} = \boldsymbol{\beta X} + \boldsymbol{\varepsilon}$$

For $\boldsymbol{\beta} \in \mathbb{R}^{k\times l}$ a set of ”regression coefficients”, where $\boldsymbol{\beta}_{ij}$ encodes how much feature x_i linearly impacts output y_j, and $\boldsymbol{\varepsilon} \in \mathbb{R}^{n\times l}$ some error. The error is modelled as normal and 0-mean, because, due to the central limit theorem, normality of error is common if the model is an appropriate one. Then, we seek the maximum-likelihood estimator of $\boldsymbol{\beta}$, such that $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}}(P(\boldsymbol{\varepsilon}|\boldsymbol{\beta}))$. Due to normality it can be shown this is equivalent to the least-squared error estimation criteria, i.e. $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}}(||\boldsymbol{\varepsilon}||_2^2)$. Using basic matrix calculus techniques, the major result can be shown.

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{Y}$$

With the appropriate order of operations, this evaluation has time complexity of order $O(nk^2 + k^3 + nkl + k^2l) = O(k^2(n+k) + kl(n+k)) = O(k(k+l)(n+k))$. Since the model is generally only performant when $n >> k$ and $n >> l$, this model is fairly fast ($O(n)$) for few features and outputs.

One significant way to interpret the solution is covariance estimation. For 0-mean data, $\boldsymbol{X}^\top \boldsymbol{X} \approx \underset{\sim}{\mathbf{\Sigma}_x}$, the covariance matrix of $\underset{\sim}{x}$, and $\boldsymbol{X}^\top \boldsymbol{Y} \approx \mathbf{\Gamma}_{\underset{\sim}{x}\underset{\sim}{y}}$, the cross-covariance matrix of $\underset{\sim}{x}$ and $\underset{\sim}{y}$. So as we see more and more data, that is $n \to \infty$ we can represent the true $\boldsymbol{\beta}$ as below.

$$\boldsymbol{\beta} = \boldsymbol{\Sigma}_{\underset{\sim}{x}}^{-1}\boldsymbol{\Gamma}_{\underset{\sim}{x}\underset{\sim}{y}}$$

3.5.2 The Wiener Filter

The Wiener filter is a noise-reduction filter that aims to remove noise subject to least-squares estimation of a desired signal. That is, we have the problem below in $\mathbb{R}^n$.

$$\underset{\sim}{y}(t) = \underset{\sim}{x}(t) + \underset{\sim}{\varepsilon}(t)$$

For 0-mean normal noise $\underset{\sim}{\varepsilon}(t)$. The assumption of noise normality as discussed in 3.5.1 gives rise to the least-squares minimisation problem via the maximum likelihood estimation of $\underset{\sim}{x}(t)$ given $\underset{\sim}{y}(t)$. The solution to this problem $\underset{\sim}{\hat{x}}(t)$ can be stated simply in terms of known covariance matrices. The below solution is commonly referred to as the minimum mean-square error estimator [33].

$$\underset{\sim}{\hat{x}}(t) = \boldsymbol{\Sigma}_{\underset{\sim}{x}}(\boldsymbol{\Sigma}_{\underset{\sim}{x}} + \boldsymbol{\Sigma}_{\underset{\sim}{\varepsilon}})^{-1}\underset{\sim}{y}(t)$$

For the covariances of $\underset{\sim}{x}(t)$ and $\underset{\sim}{\varepsilon}(t)$, $\boldsymbol{\Sigma}_{\underset{\sim}{x}}$ and $\boldsymbol{\Sigma}_{\underset{\sim}{\varepsilon}}$ respectively. This filter is said to be adaptive, as its structure depends on the data, which is to say its covariance.

3.6 Regularisation as Graph Selection

In many applications we can consider all pairwise relationships between a set of variables. This corresponds, rather simply to the complete graph of pairwise relationships. In reality, we expect many pairwise relationships to be irrelevant or not useful. In this case, we may wish to prune down the complete graph to a much sparser graph to represent only the useful pairwise relationships. In applications where there is an underlying graph present, this can be useful for identifying missing links, and in other applications it can simplify model structure and reduce size. In particular, for a Gaussian random field, strength of pairwise relationships are identified as the elements of the precision matrix, $\boldsymbol{\Sigma}^{-1}$. Friedman, J. et. al. [8] were the first to do this, applying L1 regularisation to the likelihood function of the inverse covariance matrix. We can derive this likelihood as below for k-dimensional vectors.

$$
\begin{aligned}
\mathscr{N}(\underset{\sim}{\mu},\mathbf{\Sigma}) &\sim (2\pi)^{-k/2}\det(\mathbf{\Sigma})^{-1/2}\exp(-\frac{1}{2}(\underset{\sim}{x}-\underset{\sim}{\mu})^\top\mathbf{\Sigma}^{-1}(\underset{\sim}{x}-\underset{\sim}{\mu}))\\
P(\underset{\sim}{x_0},\underset{\sim}{x_1},\ldots,\underset{\sim}{x_n}) &= \prod_{i=0}^{n}(2\pi)^{-k/2}\det(\mathbf{\Sigma})^{-1/2}\exp(-\frac{1}{2}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^\top\mathbf{\Sigma}^{-1}(\underset{\sim}{x_i}-\underset{\sim}{\mu}))\\
\ln(P(\underset{\sim}{x_0},\underset{\sim}{x_1},\ldots,\underset{\sim}{x_n})) &= \sum_{i=0}^{n}-\frac{k}{2}\ln(2\pi)+\frac{1}{2}\ln\det(\mathbf{\Sigma}^{-1})-\frac{1}{2}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^\top\mathbf{\Sigma}^{-1}(\underset{\sim}{x_i}-\underset{\sim}{\mu}))\\
&= const.+\frac{n}{2}(\ln\det(\mathbf{\Sigma}^{-1})-\frac{1}{n}\sum_{i=0}^{n}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^\top\mathbf{\Sigma}^{-1}(\underset{\sim}{x_i}-\underset{\sim}{\mu}))
\end{aligned}
$$

up to a constant and proportionality:

$$
\begin{aligned}
&= \ln\det(\mathbf{\Sigma}^{-1})-\frac{1}{n}\sum_{i=0}^{n}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^\top\mathbf{\Sigma}^{-1}(\underset{\sim}{x_i}-\underset{\sim}{\mu})\\
&= \ln\det(\mathbf{\Sigma}^{-1})-\mathrm{tr}(\mathbf{\Sigma}^{-1}\frac{1}{n}\sum_{i=0}^{n}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^\top(\underset{\sim}{x_i}-\underset{\sim}{\mu}))\\
&= \ln\det(\mathbf{\Sigma}^{-1})-\mathrm{tr}(\mathbf{\Sigma}^{-1}S)
\end{aligned}
$$

for S the sample covariance.

Regularisers on the log-likelihood are computationally easier to deal with that those in pure likelihood space, and they have the added advantage of often being connected to a particular prior on the parameters. For example, consider the L1 regularised precision matrix likelihood equation below.

$$\mathscr{L} = \ln\det(\mathbf{\Sigma}^{-1})-\mathrm{tr}(\mathbf{\Sigma}^{-1}S)+\lambda\sum_{i,j}|\mathbf{\Sigma}^{-1}_{i,j}|$$

It is well known [25] that the L1 regulariser places a Laplacian prior on each parameter independently, that is a priori we consider $P(\mathbf{\Sigma}^{-1}_{i,j}=\theta)\propto e^{-\lambda|\theta|}$. With an appropriate prior, such as L1 or L0 regularisation, this leads to zeroing out of some parameters at the maximum likelihood estimator. With a sufficiently stringent prior, this essentially leads to graph selection, as a limited number of pairwise relationships are considered valid, pruning the complete graph of relationships down to a sparser, more managable one.

A notable extension of this system is for graphs where each node is associated with a vector of scalars, rather than a single scalar, where the covariance becomes a rank-3 tensor, rather than a 2d matrix. Z. Yue et. al. [38] come up with a fast algorithm for approximate solving of the L0 regularised vector inverse covariance tensor approximation.

Both these methods address methods for quantifying the graphical relationship at a single point in time between a set of variables for which many samples are available. The methods do not describe how to handle relationships in a graph with time series at each vertex, where relationships may include delayed effects. Examples of such real world networks would be climatic relationships, where it takes time for the climate in one region to effect another region, or a brain network, where neurons influence each other with a synaptic delay.

3.7 Conditional Mutual Information

A key goal of this thesis is to inform understanding of which pairwise relationships are important between different variables. In particular, throughout the thesis we look at multivariate time-varying data. For some pair of time series $x_i(t)$ and $x_j(t)$, we would like to know what amount of the information at i is explicable using the information at j, presuming we have already accounted for (conditioned upon) the data at j. For this we assume that $x_i(t)$ is just one possible trajectory along the random variable X_i, which follows a distribution of such trajectories. Two such random variables X_i and X_j are independent given the remaining data $X_{\neq i,j}$ if the relationship below is satisfied.

$$P(X_i, X_j | X_{\neq i,j}) = P(X_i | X_{\neq i,j}) P(X_j | X_{\neq i,j})$$

A popular and statistically robust way to measure how far apart these two distributions are; the joint distribution and the distribution assuming independence, is the Kullback-Leibler divergence, D_{KL}. The divergence of two distributions is measured as $\int_{\mathscr{X}} P(x) \log \frac{P(x)}{Q(x)}$ for distributions P and Q over domain $\mathscr{X}$. We also integrate with respect to $dX_{\neq i,j}$, to average out all the possible external information. This is precisely the conditional mutual information between the two time series, given the others. Unfortunately, this is essentially a useless tool as stated, as we have just one sample of each timeseries X_i over a massive domain of size $|\mathbb{R}|^{|\mathbb{R}|}$. To make this tool useful, a few changes are needed.

First, we simply discretise the data, reducing the domain to size $|\mathbb{R}|^n$. Then, we must assume a prior distribution on the X variables. This way, our single sample allows for the acquisition of a posterior distribution on the trajectories, which allows for the integration across all possible trajectories in order to determine the mutual information between variables. This still provides a fairly poor estimate, as one sample gives no knowledge of the variance in the underlying distribution. An additional presumption we can make is rough stationarity of the data. This way, we can treat contiguous windows of datapoints as independent samples. The tradeoff here is between accounting for far-reaching effects across long time spans, and enough samples to appropriately predict the distribution of the data.

In the thesis, a key formula will be the simplification of the CMI into an expression in terms of entropy. The expression for entropy of a distribution over domain $\mathscr{X}$ is as below.

$$H(X) = \int_{\mathscr{X}} P(x) \log(P(x)) dx$$

When the base of the logarithm is 2, the unit of this quantity is the Shannon, or sometimes abusively bits. When the logarithm uses the natural base e, the unit is nats. Unless otherwise specified, we will work in nats in this thesis. CMI can be broken down in terms of conditional entropy as below.

$$CMI(X,Y|Z) = H(X|Z) + H(Y|Z) - H(X,Y|Z)$$

In a sense, this tells us that CMI measures how much excess information is contained within X and Y that is not explained either by the information in Z, or by their relationship (once accounting for information from Z).

4 Spatiotemporal Analysis of BOM Dataset

4.1 Bureau of Meteorology ACORN SAT dataset

The ACORN dataset is a dataset of temperature stations across Australia [18]. 104 select stations start before or on $01/03/1975$, and end after or on $31/12/2023$. This range covers 17838 days. Across the stations, around $\approx 1\%$ of data is backfilled (or forwardfilled if missing entries are at the front of the data). The dataset is homogenised, so that some data has been processed to account for station movements, changes in equipment, and changes in site conditions over the decades.

Prediction of temperature data across time and space is of great use across a number of applications. Both long and short term weather forecasting can be of great help across many industries. Crop and cattle futures may change in value depending on long-term temperature changes. Short-term temperature changes can effect power output from renewable energy sources, road conditions, and heat stroke risk for labourers and the general population.

4.2 Efficient Lossy Compression Implementation

We implement Moura's compression scheme discussed in 3.1.1 with $\sigma = 1000km$ performing well on the Australian temperature dataset, as this is close to the mean distance. We compress daily mean temperature from 1975-2023 across Australia. Using the adjacency matrix as the shift matrix with these weights yields similar, though poorer compression results to Moura's, shown in Fig. 5.

Fraction of Coefficients Used	1/50	1/20	1/15	1/10	1/7	1/5	1/3
RMSE (%)	8.54	6.95	6.47	5.80	5.18	4.54	3.43

Figure 5: Australian Mean Daily Temperature Data (1975-2023) Compression Results

As discussed in 3.1, the spatiotemporal GFT is just a combination of DFTs (via FFT) and smaller GFTs, so a relevant question here is how much of this compression is due to the DFT, and how much the GFT? So we perform a similar compression process, only applying the DFT, and we note the compression results in Fig. 6.

Fraction of Coefficients Used	1/50	1/20	1/15	1/10	1/7	1/5	1/3
RMSE (%)	11.79	10.33	9.71	8.69	7.65	6.55	4.69

Figure 6: Australian Mean Daily Temperature Data (1975-2023) Time-Only Compression Results

Clearly, the GFT is a significant factor in the compression performance. Compression results almost identical (to within ±0.05% RMSE) were achieved using the Laplacian matrix operator with inverse distance weights, rather than a Gaussian weighted adjacency matrix, demonstrating some amount of arbitrarity both in choosing appropriate edge weights and matrix structure employed. This is addressed further when we discuss graph filters in 4.5.1

4.3 Temporal Predictive Filters

Prior to the application of predictive spatial methods, we review the application of purely temporal methods on the Australian daily temperature data set. A key point about temperature data is that is has a seasonal component. For example, Fig. 7 shows the monthly mean temperature throughout 1975-2023 at Meekatharra Airport, WA, showing a clear yearly seasonal component to the temperature values.

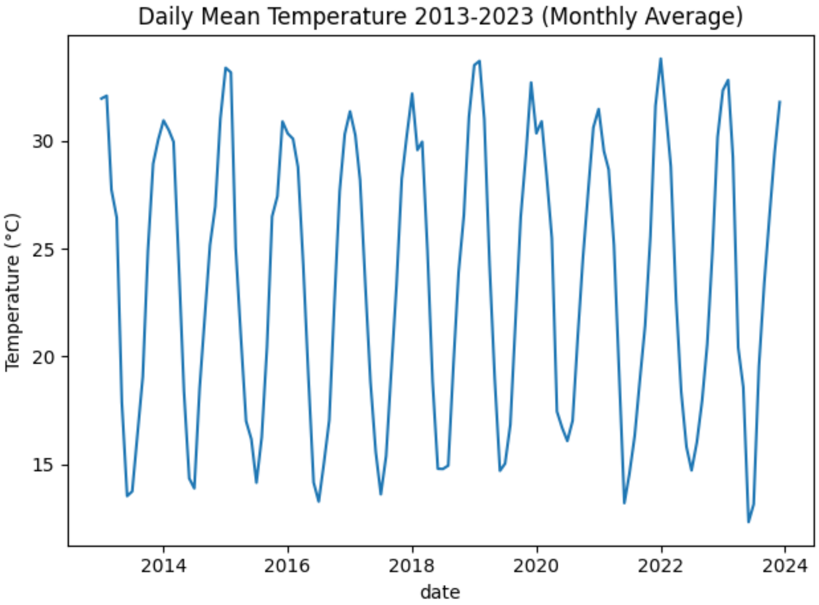

Figure 7: Monthly mean temperature at Meekatharra Airport, WA from 2013-2023.

Many prediction methods, including autoregression, assume wide-sense stationarity (autocorrelation does not change over time). Long-term effects like seasonality introduce non-stationary effects into the data. Before fitting the data further, we thus wish to remove seasonal effects, although may wish to add them back in later. We will discuss two methods for removing seasonality; explicitly modelling the trend and differencing the data to remove non-stationarity. As we will see, seasonality also makes the data not normal, and removing seasonal effects improves normality, which leads to better performance of autoregression.

4.3.1 Explicit Seasonality Modelling

The explicit method involves removing the seasonal component from the data as below. This method removes the DC component, and the first two harmonics of the yearly seasonal component, and possibly more harmonics. Based on observations of the data, the first two harmonics contribute the vast majority of the seasonal component.

$$\hat{T}_t = T_t - \alpha_0 - \beta_0 \sin(2\pi t/365) - \gamma_0 \cos(2\pi t/365) - \beta_1 \sin(4\pi t/365) - \gamma_1 \cos(4\pi t/365)$$

We minimise the RMSE of $\hat{T}_t$ using linear regression, getting $2.8429°C$. Note that in the first-harmonic-only case, the amplitude of the seasonal component is given by $\sqrt{B^2+C^2}$, and the phase relative to cosine (assuming max temps on new years day) by $-\arctan(B/C)$. In Fig. 8 we show the phase (in years) and the amplitude of the seasonal component across the country, fitting only the first harmonic. We can see clear geographical deviations, phase by latitude, and amplitude by how close to the coast a particular station is. These observations are backed up in geography textbooks, explained respectively by solar insolation and the maritime effect [2].

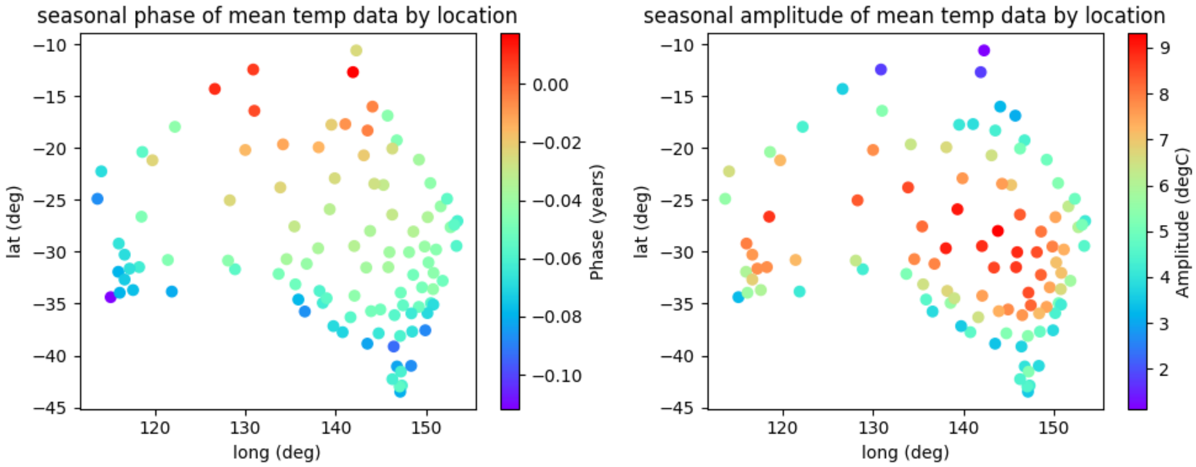

Figure 8: Seasonal component (first harmonic only) of temperature across Australia.

We can also see via histogram how removing the seasonal component acts to normalise the data. Again at Meekatharra Airport we can judge the distribution of temperatures before and after seasonal adjustment. We view the histogram and normal quantile-quantile plot, binning according to Rice's rule in Fig. 9. Note the transition from an approximately bimodal distribution to a relatively normal one, albeit with some skew.

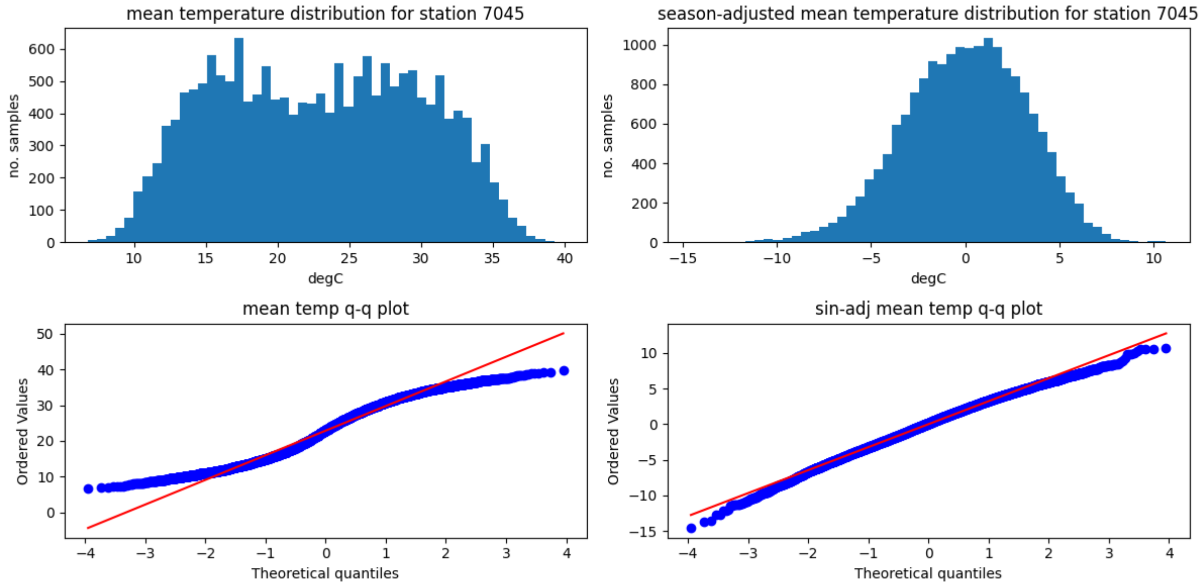

Figure 9: Temperature distribution at Meekatharra Airport before (left) and after (right) seasonal adjustment.

4.3.2 Differencing

If data has a long-term slowly changing mean, as in seasonality, then the difference of that mean is small. I.e. $\mu_t - \mu_{t-1} \approx 0$. This means that if the data T_t is differenced, then the mean, even if it is slowly changing, is sent to 0. In terms of the Z-transform, this is equivalent to filtering the data for a unit-root, as in $\tilde{T} = (1 - z^{-1})T$. Unfortunately, differencing also acts like a derivative, heightening noise in a dataset. Differencing can have a central-limiting effect, improving normality, but it is often weak as it is not the same as summing i.i.d. variables, and it is only between two variables, rather than many. We view the distribution that differencing yields at Meekatharra Airport in Fig. 10. This example is one of the better performing ones for differencing. In general across the dataset differencing typically is poorer than explicit seasonality removal as discussed in 4.3.1. Hence, in the following sections we use explicit seasonality removal.

4.3.3 Geographical Analysis

Fig. 11 shows summary statistics across the country for seasonality adjusted temperature distributions.

Even after adjustment for seasonality, the weather is more extreme in central regions. Skew varies along north/south coastlines (this can be confirmed by looking at the distributions in Tasmania), with south coastlines having positive skew (more extreme hot events, adjusting for seasonality), and north coastlines negative skew (more extreme cold event, adjusting for sea-

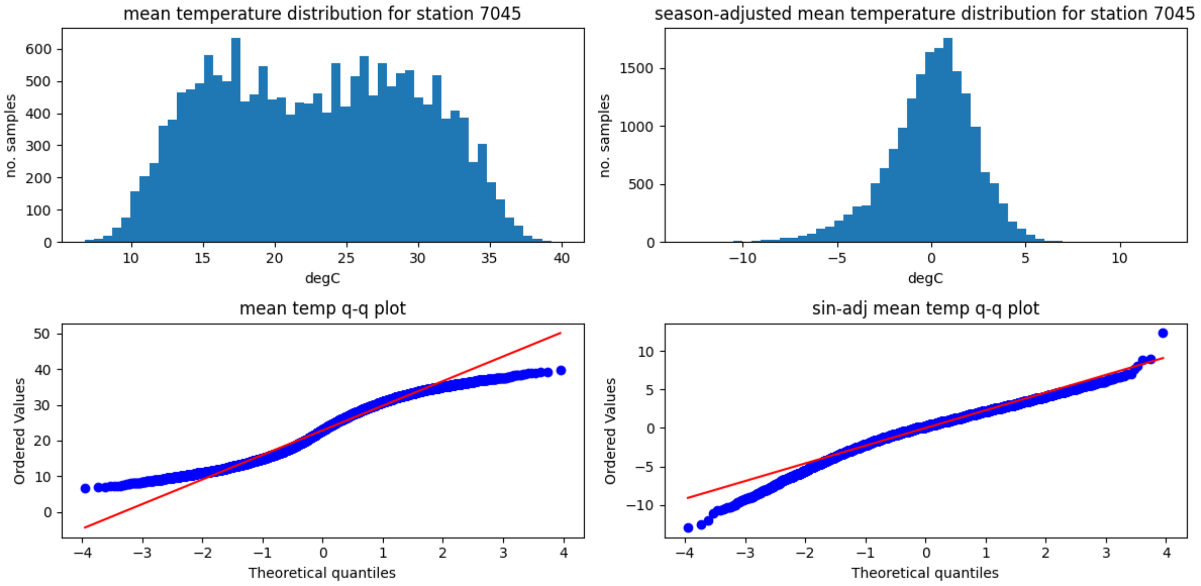

Figure 10: Temperature distribution at Meekatharra Airport before (left) and after (right) differencing.

sonality). Kurtosis (long-tailedness) is mostly small, with a some outliers, particularly along the south coast.

4.3.4 Autoregression

We then assess what order of autoregressive fit is most useful for the remaining error. We start with a high order fit, and reduce the order when terms add little value. Plotting the regression coefficients across the 104 stations for 5 days worth of delay, we get the Fig. 12. Only the first two coefficients are significant, so we use a second order fit with coefficients fitted for each station. For example with the coefficients 0.6 and -0.1 we get the model below.

$$T_t = 0.6T_{t-1} - 0.1T_{t-2} + \varepsilon_t$$

Running the linear autoregression against the data produces a total RMSE of $2.062°C$. This is marginally better than a first order fit with RMSE of $2.093°C$.

A good statistical test for model correctness is the Ljung-Box test. It tests if the unmodelled residuals are uncorrelated Gaussian noise. It checks against the first n delays for statistically significant autocorrelation. As per Fig. 12, we hope that second order autoregression renders removes autocorrelations in the residuals. We run the Ljung-Box test for a range of delays, and note how many of our time series pass the test for each delay in Fig. 13. Even after removing the two major correlations, many timeseries do not pass the test at even modest delays. The situation does not improve for higher order autoregressive filters. The data is hence not sim-

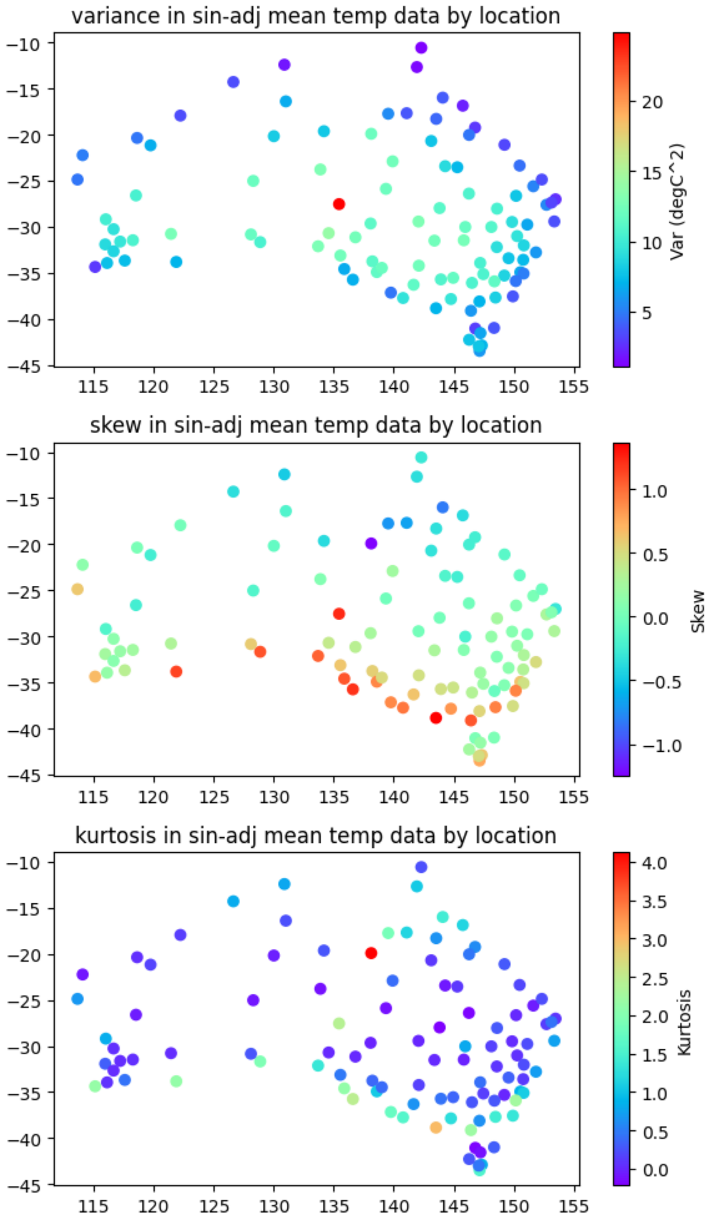

Figure 11: Summary statistics for seasonality adjusted temperature distributions.

ply explained by autoregression. Autoregressive-moving average models directly address the Ljung-Box criteria, but are slow to fit and in initial experiments on the dataset don't significantly improve the results.

4.4 Spatial Autoregressive Processes

Some literature on continuous-domain stochastic processes were discussed in 3.4. From there, a spatial stochastic process X is a collection of random variables in some space S, often $\mathbb{R}^n$, $\underset{\sim}{X}_x \forall \underset{\sim}{x} \in S$. We wish to develop a generic model for autocorrelated spatial processes. For a simple autocorrelated spatial process, we may expect that each signal is highly correlated with

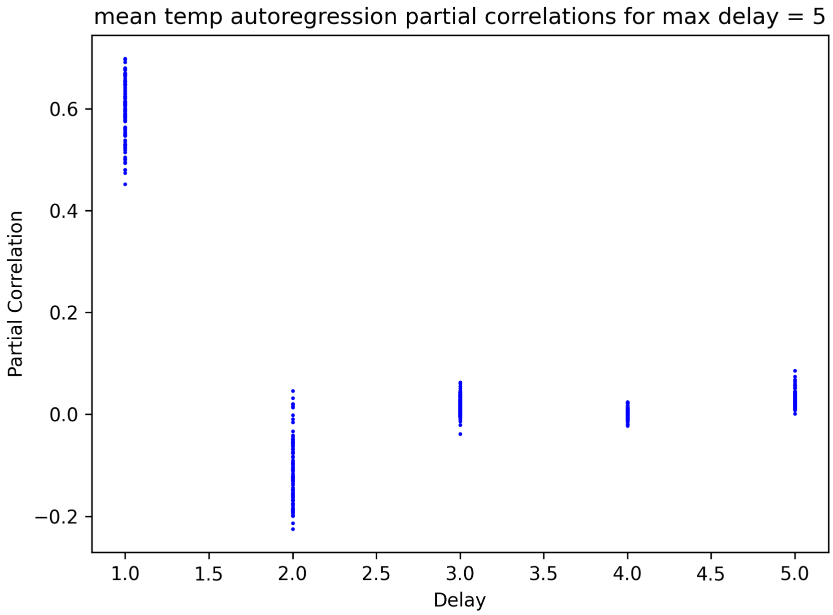

Figure 12: Correlation coefficients for a 5th order autoregression.

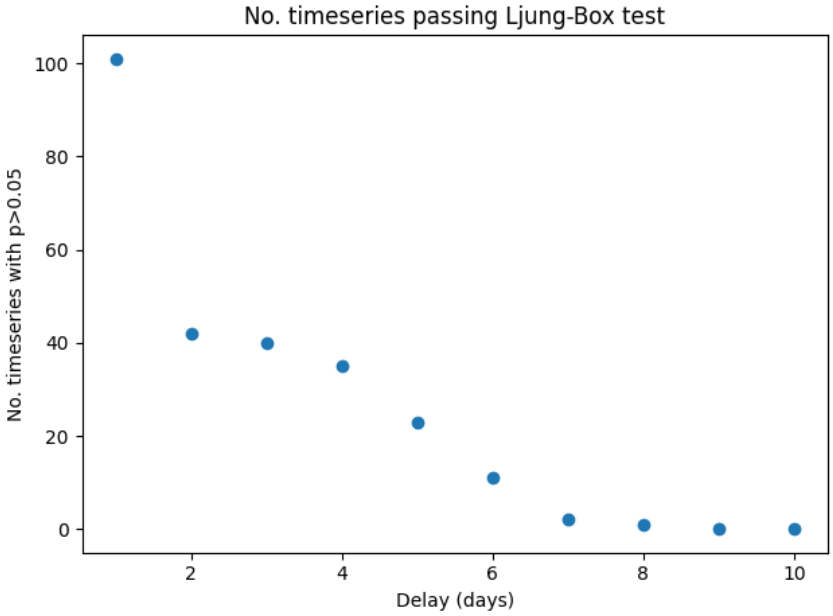

Figure 13: Number of timeseries passing the Ljung-Box test at different delays 1-10.

nearby signals, and less correlated with far away signals. In particular, we can write the signal at a point as a function of the nearby points, and some amount of noise injected at the point, as below.

$$X_{\underset{\sim}{x}} = \frac{1}{S_n h^{n-1}} \int_{\partial B_{n,h}} \alpha(h) X_{\underset{\sim}{x}+d\underset{\sim}{x}} d\underset{\sim}{x} + \gamma(h) n(\underset{\sim}{x})$$

For α and γ functions of h, $B_{n,h}$ the n dimensional ball of radius h with surface area $S_n h^{n-1}$, and $n(\underset{\sim}{x})$ the noise injected at $\underset{\sim}{x}$ with unit variance. Note that this model does not account for noise inside $B_{n,h}$, lumping it all together as noise at $\underset{\sim}{x}$. This equation is very reminiscent of the integral form of the Laplacian discussed in 2.1. For small h, we write $\alpha(h) = 1 - \alpha_1 h - \alpha_2 h^2 + O(h^3)$

$$\begin{aligned}
-\gamma(h)n(\underset{\sim}{x}) &= \frac{1}{S_n h^{n-1}} \int_{\partial B_{n,h}} (1 - \alpha_1 h - \alpha_2 h^2 - ...)X_{\underset{\sim}{x}+\underset{\sim}{dx}} - X_{\underset{\sim}{x}}\underset{\sim}{dx} \\
-\frac{\gamma(h)}{h^2}n(\underset{\sim}{x}) &= \frac{1}{S_n h^{n+1}} \int_{\partial B_{n,h}} X_{\underset{\sim}{x}+\underset{\sim}{dx}} - X_{\underset{\sim}{x}}\underset{\sim}{dx} - (\frac{\alpha_1}{h} + \alpha_2 + O(h))X_{\underset{\sim}{x}}
\end{aligned}$$

In the limit as $h \to 0$, this degenerates to $X_{\underset{\sim}{x}}$ having either no autoregressive component or no noise, or no limit unless $\alpha_1 = 0$ and $\gamma(h) = \gamma h^2 + O(h^3)$. In the non-degenerate case, we can work as follows to find a limit.

$$\begin{aligned}
-\frac{\gamma h^2 + O(h)}{h^2}n(\underset{\sim}{x}) &= \frac{1}{S_n h^{n+1}} \int_{\partial B_{n,h}} X_{\underset{\sim}{x}+\underset{\sim}{dx}} - X_{\underset{\sim}{x}}\underset{\sim}{dx} - (\alpha_2 + O(h))X_{\underset{\sim}{x}} \\
h &\to 0 \\
-\gamma n(\underset{\sim}{x}) &= \frac{1}{2n}\nabla^2 X_{\underset{\sim}{x}} - \alpha_2 X_{\underset{\sim}{x}}
\end{aligned}$$

Setting $\gamma \leftarrow 2n\gamma$, $\lambda^2 \leftarrow 2n\alpha_2$, this reduces to the screened Poisson equation, also referred to as the modified Helmholtz equation.

$$-\gamma n(\underset{\sim}{x}) = (\nabla^2 - \lambda^2)T_{\underset{\sim}{x}}$$

As discussed in 3.4.1, Whittle (1963) proved that the correlation $\rho(T_{\underset{\sim}{x}}, T_{\underset{\sim}{x+r}})$ in the case of Gaussian white noise is given by a Matérn kernel in the distance between points r [35]. The general Matérn kernel is given below.

$$\rho_{\nu,\lambda}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}(\lambda r)^{\nu} K_{\nu}(\lambda r)$$

Where K_ν denotes the modified Bessel function of the second kind of order ν. In this particular problem $\nu = 2 - n/2$. In 2 and 3 dimensions, the correlation function simplifies to those below.

$$\begin{aligned}
2D : \mathrm{Cov}(T_{\underset{\sim}{x}}, T_{\underset{\sim}{x}+\underset{\sim}{r}}) &= \lambda r K_1(\lambda r) \\
3D : \mathrm{Cov}(T_{\underset{\sim}{x}}, T_{\underset{\sim}{x}+\underset{\sim}{r}}) &= e^{-\lambda r}
\end{aligned}$$

The solution in 2 dimensions lines up with what Whittle (1954) found, calling it the "elementary correlation in two dimensions" [34].

This model assumes constant variance, which is clearly not present in the Australian temperature data set. Normalising for variance yields the correlation, and we can plot the correlation (after seasonal adjustment) against the 2D covariance kernel in Fig. 14 (with variance normalised to 1). We also plot the more arbitrary fit $e^{-\lambda|r|}$. Both these models have a single parameter (γ is fixed by the normalisation). Both achieve a similar fit but overall there is a lot of noise in the correlation, despite the high number of samples, suggesting this model is too simple to encode the relationships present in the dataset. The correlation function in 2 dimensions is simply $\lambda r K_1(\lambda r)$.

4.4.1 Modelling

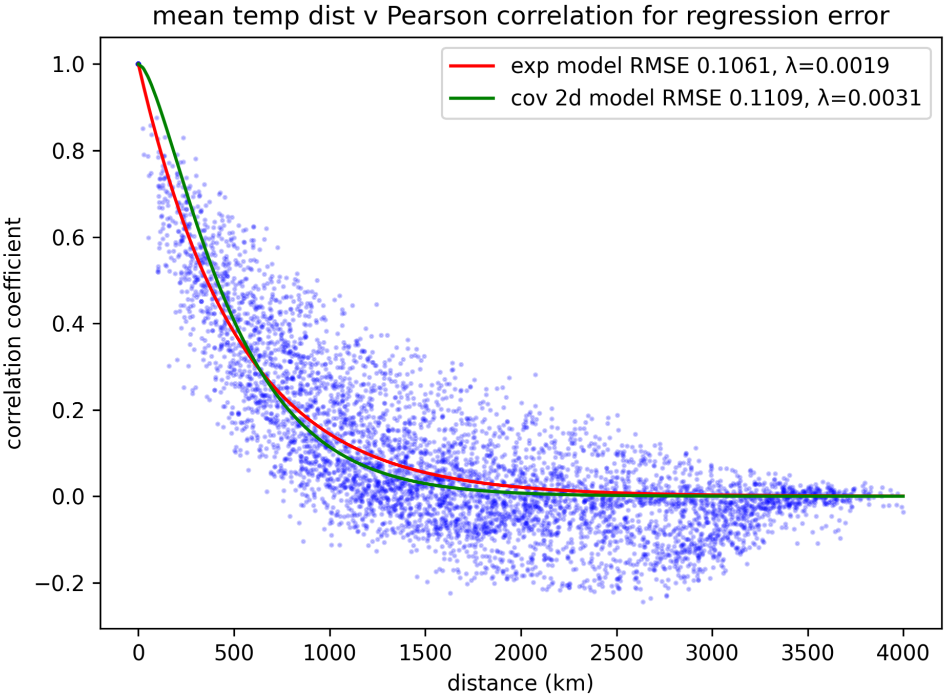

Figure 14: Fitting seasonality adjusted correlations against an exponential model and a Matérn kernel.

4.4.2 Connection to the Matérn Kernel

4.4.3 The 1D Correlation Function

As discussed in 3.4.1, the general solution is invalid when $n = 1$, in which case we can derive the solution directly. The Green's function in 1 dimension is given below.

$$G(r) = \frac{1}{2\lambda} e^{-\lambda r}$$

The covariance is given by the 1 dimensional convolution of the Green's function.

$$\begin{aligned}
\mathrm{Cov}(T_{\underset{\sim}{x}}, T_{\underset{\sim}{x}+\underset{\sim}{r}}) &= \gamma^2 \int_{-\infty}^{\infty} G(\tau)G(r-\tau)d\tau \\
&= \frac{\gamma^2}{4\lambda^2} \int_{-\infty}^{\infty} e^{-\lambda|\tau|-\lambda|r-\tau|} d\tau \\
r &\geq 0 \\
&= \frac{\gamma^2}{4\lambda^2} \left[\int_{-\infty}^{0} e^{\lambda(\tau-r+\tau)} d\tau + \int_{0}^{r} e^{\lambda(-\tau-r+\tau)} d\tau + \int_{r}^{\infty} e^{\lambda(-\tau+r-\tau)} d\tau \right] \\
&= \frac{\gamma^2}{4\lambda^2} \left[\frac{1}{2\lambda} e^{\lambda(2\tau-r)}|_{-\infty}^{0} + re^{-r\lambda} - \frac{1}{2\lambda} e^{-\lambda(2\tau-r)}|_{r}^{\infty} \right] \\
&= \frac{\gamma^2}{4\lambda^3} \left[\frac{1}{2} e^{-\lambda r} + \lambda re^{-\lambda r} + \frac{1}{2} e^{-\lambda r} \right] \\
&= \frac{\gamma^2}{4\lambda^3} (1+\lambda r) e^{-\lambda r}
\end{aligned}$$

The correlation is simply the covariance normalised to 1 when $r = 0$, as below.

$$\rho(r) = (1+\lambda r)e^{-\lambda r}$$

4.4.4 Numerical Confirmation

We confirm the 1D and 2D results numerically. The 1D and 2D domains $\mathbb{R}$ and $\mathbb{R}^2$ can be approximated by the toroids $\mathbb{T}$ and $\mathbb{T}^2$. This is the same approximation the DFT uses. It is significant, because these domains, like the originals, have no boundary, and thus no need to specify boundary conditions. Just as with the DFT however, aliasing can occur when a function in the original domain is of wide enough extent that it overlaps with itself in the approximate domain. We put a lattice graph on the approximate domains, and discretise the differential equation from before.

$$-\gamma n(\underset{\sim}{x}) = (-\mathbf{L} - \lambda^2 \mathbf{I})\mathbf{v}$$

Recalling that $-\mathbf{L}$, the negative Laplacian matrix is an approximation for the continuous Laplacian operator. We generate Gaussian white noise at each point, and invert the matrix operator to solve for $\mathbf{v}$. To increase the confidence of our covariance estimates, we generate noise for a large number of samples. The numerically estimated function along with the theoretically correct function for 1 and 2 dimensions are shown in Fig. 15.

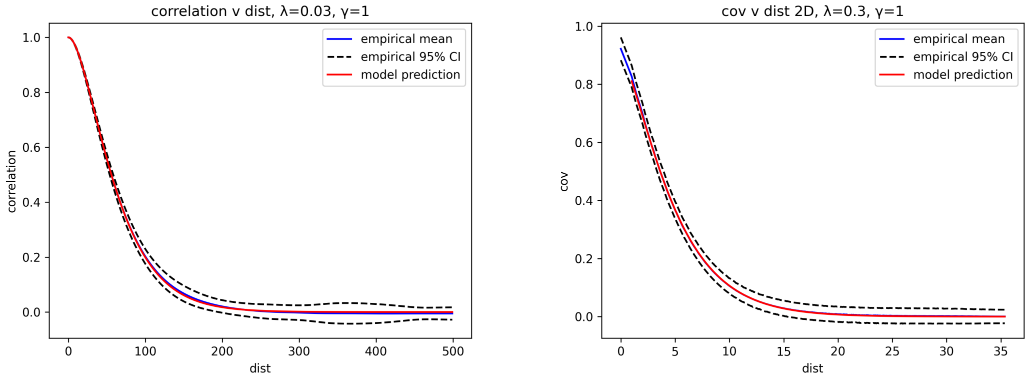

Figure 15: Numerical estimates for 1 (left) and 2 (right) dimensional covariance functions under the screened Poisson equation with unit variance Gaussian noise.

4.4.5 Kriging

Kriging is a collection of methods for interpolating spatial data from a set of observations. The mathematical details of Kriging are covered in 3.4.2. Kriging of Australian temperature data has been performed in Noel (2021) [5]. Using the 3D kernel discussed in 4.4, and swapping covariances for correlations (as variances vary across the domain), we produce the interpolation in Fig. 16.

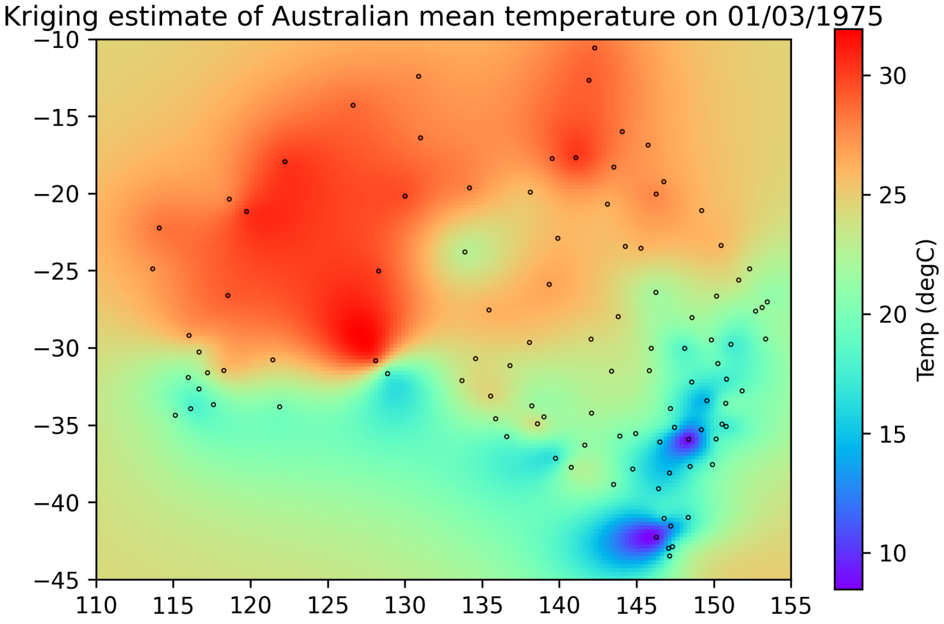

Figure 16: Kriging Australian temperature on 01/03/1975 with the 3D Matërn kernel.

The Kriging error discussed in 3.4.2 operates on variances rather than correlations, so we multiply $\mathbf{\Sigma}_{ij}$ by $\sqrt{\mathrm{Var}(X_i)\mathrm{Var}(X_j)}$, and $\mathrm{Cov}(X_i, X_0)$ by $\sqrt{\mathrm{Var}(X_i)\mathrm{Var}(X_0)}$. For simplicity, we model $\mathrm{Var}(X_0)$ as the global average variance. Also, because a good variance estimate here assumes stationary data, we use the variances of the seasonality adjusted data to improve stationarity.

The Kriging standard error (square root of Kriging error) is shown in Fig. 17.

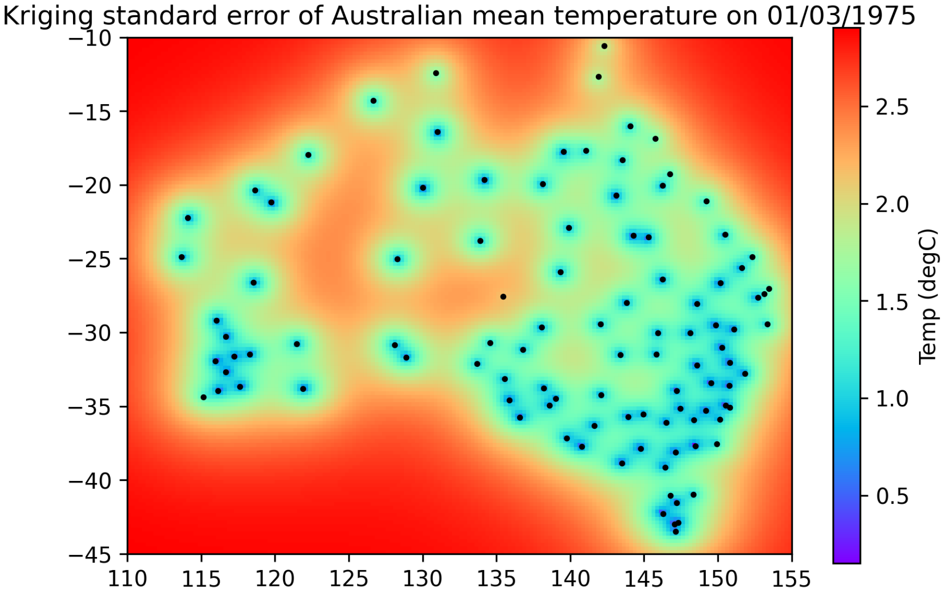

Figure 17: Kriging Standard Error for Australian temperature on 01/03/1975.

This shows that we are more sure of data where we have measured, and less sure further away. It is important to differentiate between the true interpolation error, which is unknown, and the Kriging error, which merely measures the expected RMSE of the interpolation, given how much the data varies.

4.4.6 Limitations of the Matérn Kernel

As observed in 4.4.1, there is significant noise in the fit. As we sample 17838 days, the sample covariances are accurate, so sample noise does not sufficiently explain the variation. The two major assumptions made in the model are anisotropy, that the diffusion coefficient λ is constant throughout the space, and that the independent variance γ^2 injected at each point is constant. The second point here, as measured is not true. Although we work with correlation coefficients, which removes variance, non-uniform variance also effects the correlations throughout the space. We do not discuss analytically addressing these shortcomings in this report, but we do analyse anisotropy in the dataset.

Ideally, for a spatial process in the space S, the correlation function is a function $f : S^2 \to \mathbb{R}$. In our case, it is a function from two latitudes and two longtitudes to a correlation between $(-1, 1)$. Naturally, it is hard to visualise a function of four inputs. Fitting such a smooth function empirically is a problem ripe for machine learning, but this is also not addressed in this report. Instead, we reduce the problem to having just two inputs, distance, and direction. Because the relationships are bidirectional, we assign each covariance relationship a direction ranging 180° from south, to east, to north. We can thus analyse the directionality (anisotropy) of the function as in Fig. 18 In this figure we can see smoothly varying trends in the angle, as

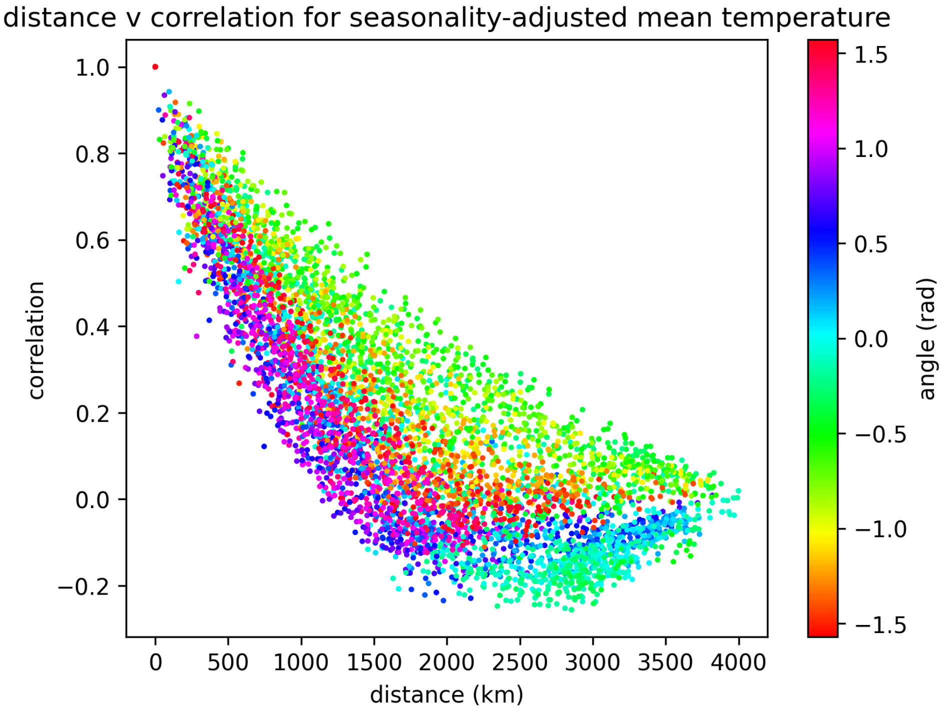

Figure 18: Correlation vs distance and direction. $-\pi/2$ is south, 0 is east, $\pi/2$ is north. Since north and south give the same direction, the colourbar is cyclic.

colours follow bands within the fit, but the function is not one-to-one. For better visualisation, we plot correlation as the colour and distance and direction on the x and y axes respectively in Fig. 19. As we can see, there is still some disagreement between nearby points, but broadly the function exhibits far less noise that the original fit, suggesting part of the data is simply explicable by anisotropy, rather than local effects. Again, without analytical analysis, this is a difficult function to fit empirically without advanced techniques from machine learning.

4.5 Deterministic Graph Filters

4.5.1 Introduction

Just as time domain signal processing can be used to filter noise, so too can graph signal processing. The most natural port from the temporal domain to the graphical domain would be to use the GFT to bring graphical data into the frequency domain, before applying a filter, and then inverting the GFT to get the filtered signal. In order to filter noise efficiently, we need to know what both the noise spectra is, and the signal spectra. We then keep the components of the spectra that have the most signal and least noise (high SNR). This is very similar to the compression

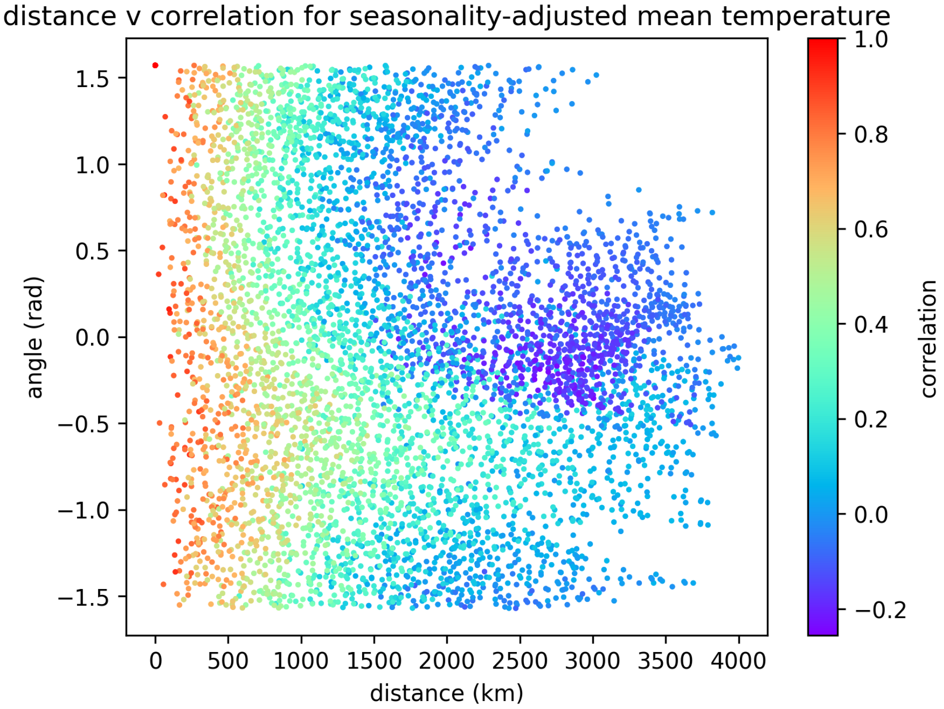

Figure 19: Correlation vs distance and direction. $-\pi/2$ is south, 0 is east, $\pi/2$ is north.

process in 3.1.1, where frequencies containing the most signal are preferred. In our dataset, we have no ground truth to compare the data to, so we will artificially introduce Gaussian sensor noise, then attempt to remove it using a spatial filter, reporting the RMSE between the original and filtered signals for different amounts of injected noise.

$\sigma_{noise}/\sigma_{signal}$ (%)	1.00	5.00	10.0	20.0	50.0	100
RMSE (%)	1.00	4.98	9.81	18.69	37.63	54.34

Figure 20: Spatial noise reduction via Wiener filter.

Clearly, the filter is more effective at noise reduction at high levels of noise, as low levels of noise are hard to distinguish from legitimate high frequency signal.

4.5.2 Well-Chosen Graph Fourier Transforms

Throughout this subsection we consider a graph with n vertices, and a graph operator operator $\boldsymbol{A}$ with eigendecomposition $\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$, and hence graph Fourier transform $\boldsymbol{V}^{-1}$.

From the compression and filtering problems discussed, we note that problems are solved better when more of the data is contained in fewer eigenvectors of the graph Fourier transform. We analyse what happens when we apply the GFT to the data. Assuming the data is 0-mean and

Gaussian, we can write the distribution before and after the GFT application.

$$\begin{aligned} P(\underset{\sim}{X} = \underset{\sim}{x}) &= \text{const. } \exp(-\underset{\sim}{x}^{\top}\boldsymbol{\Sigma}_{\underset{\sim}{x}}^{-1}\underset{\sim}{x}/2) \\ &= \text{const. } \exp(-(\boldsymbol{V}\boldsymbol{V}^{-1}\underset{\sim}{x})^{\top}\boldsymbol{\Sigma}_{\underset{\sim}{x}}^{-1}(\boldsymbol{V}\boldsymbol{V}^{-1}\underset{\sim}{x})/2) \\ &= \text{const. } \exp(-(\boldsymbol{V}^{-1}\underset{\sim}{x})^{\top}(\boldsymbol{V}^{-1}\boldsymbol{\Sigma}_{\underset{\sim}{x}}^{-1}\boldsymbol{V})(\boldsymbol{V}^{-1}\underset{\sim}{x})/2) \\ \boldsymbol{\Sigma}_{\boldsymbol{V}^{-1}\underset{\sim}{x}} &= \boldsymbol{V}^{-1}\boldsymbol{\Sigma}_{\underset{\sim}{x}}\boldsymbol{V} \end{aligned}$$

So then, we wish to choose an orthonormal transformation which concentrates the data given how it transforms covariance. This goal is similar to the goal of dimensionality reduction, where most of the data is contained within just a few dimensions. A common solution to this problem is principle component analysis (PCA), which selects in order the eigenvectors of the covariance matrix in order of descending eigenvalue as the most significant components. In particular in the case of the compression problem, we select the top k eigenvectors of $\boldsymbol{\Sigma}_{\underset{\sim}{x}}$ with k/n the compression ratio. This prompts us to consider the case where the graph operator $\boldsymbol{A} = \boldsymbol{\Sigma}_{\underset{\sim}{x}}$.

Recall from 3.5.2 that the maximum likelihood estimation for a signal of interest $\underset{\sim}{x}$ given additional Gaussian noise $\underset{\sim}{\varepsilon}$ and a measurement $\underset{\sim}{y}$, so that $\underset{\sim}{y} = \underset{\sim}{x} + \underset{\sim}{\varepsilon}$ is as below.

$$\underset{\sim}{\hat{x}} = \boldsymbol{\Sigma}_{\underset{\sim}{x}}(\boldsymbol{\Sigma}_{\underset{\sim}{x}} + \boldsymbol{\Sigma}_{\underset{\sim}{\varepsilon}})^{-1}\underset{\sim}{y}$$

For $\boldsymbol{\Sigma}_{\underset{\sim}{x}}$ and $\boldsymbol{\Sigma}_{\underset{\sim}{\varepsilon}}$ the signal and noise covariance matrices.

With the operator $\boldsymbol{A} = \boldsymbol{\Sigma}_{\underset{\sim}{x}}$ eigendecomposed into $V\Lambda V^{-1}$ we can then write the Wiener filter as below.

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\boldsymbol{-1}} + \boldsymbol{\Sigma}_{\underset{\sim}{\varepsilon}})^{-1} \\ &= \boldsymbol{V}\boldsymbol{\Lambda}(\boldsymbol{V}\boldsymbol{\Lambda} + \underset{\sim}{\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}}\boldsymbol{V})^{-1} \end{aligned}$$

In the case that noise is uncorrelated Gaussian white noise, $\boldsymbol{\Sigma}_{\underset{\sim}{\varepsilon}} = \boldsymbol{I}\sigma^2$ (or any noise whose covariance matrix has the same eigenvectors), the expression can be simplified further.

$$\boldsymbol{W} = \boldsymbol{V}\boldsymbol{\Lambda}(\boldsymbol{\Lambda} + \boldsymbol{I}\sigma^2)^{-1}\boldsymbol{V}^{-1}$$

So that under the graph operator $\boldsymbol{\Sigma}_{\underset{\sim}{x}}$ the optimal linear least squares filter design is equivalent to taking the GFT, applying the simple filter below, and taking the inverse GFT.

$$H(\lambda) = \frac{\lambda}{\lambda + \sigma^2}$$

This optimal filter design requires either an eigendecomposition of the covariance matrix to find the GFT, or an inversion of the matrix $\mathbf{\Sigma}_x + \mathbf{\Sigma}_\varepsilon$. Note in this case that the graph power spectral density of the signal is given by $P(\lambda) = \lambda$. For a real function f with a Lorentz expansion, the operator $f(\mathbf{\Sigma}_{\underset{\sim}{x}})$ admits the transfer function below.

$$H(\lambda) = \frac{f(\lambda)}{f(\lambda) + \sigma^2}$$

An important selection is $f(x) = x^{-1}$, because of the interpretation of the inverse covariance matrix. In particular, an entry $\Sigma^{-1}_{\underset{\sim}{x}ij}$ encodes the dependence of $\underset{\sim}{x_i}$ and $\underset{\sim}{x_j}$ on each other, accounting for dependencies on other variables (conditional dependence). Because variables tend to depend on just a few other variables, while they may be covariant with many, the inversion has a dimensionality reducing effect, so that the operator has sparser entries. This can speed up computation of matrix-vector products, which makes the filter easier to implement.

Some methods for fast computation of the inverse covariance matrix exist, such as the graphical lasso, and constrained ℓ_1 minimisation [29][14]. Both methods allow for high amounts of parallelisation and use gradient descent approaches. They yield sparse structures and can operate on large datasets. They also have the benefits of providing a superior estimate of the inverse covariance matrix when the number of variables is larger than the number of samples, which is often true for large datasets. In our case, we hope the presence of a distance metric and an approximate covariance function on the graph allows us to make reasonable guesses at the structure of the inverse covariance matrix. We will discuss this in the following section.

4.5.3 Empirical Results Across GFTs

First, we will investigate approximations to the inverse covariance matrix, called the precision matrix, after seasonal adjustment, so that the data is approximately normal. Then, we compare how well each operator filters noise.

We expect series data to be dependent on nearby series. We plot how the true inverse precision $\mathbf{\Sigma}^{-1}_{ij}$ vary with both absolute distance, and ”distance order”, which is to say for distance d_{ij}, if it has distance order k, there are $k-1$ distances d_{il} smaller than it. For now we will ignore diagonal entries, as they exhibit unique behaviour. The plot is shown in Fig. 21.

There is no perfect structure in the precision values for either distance or distance order. There

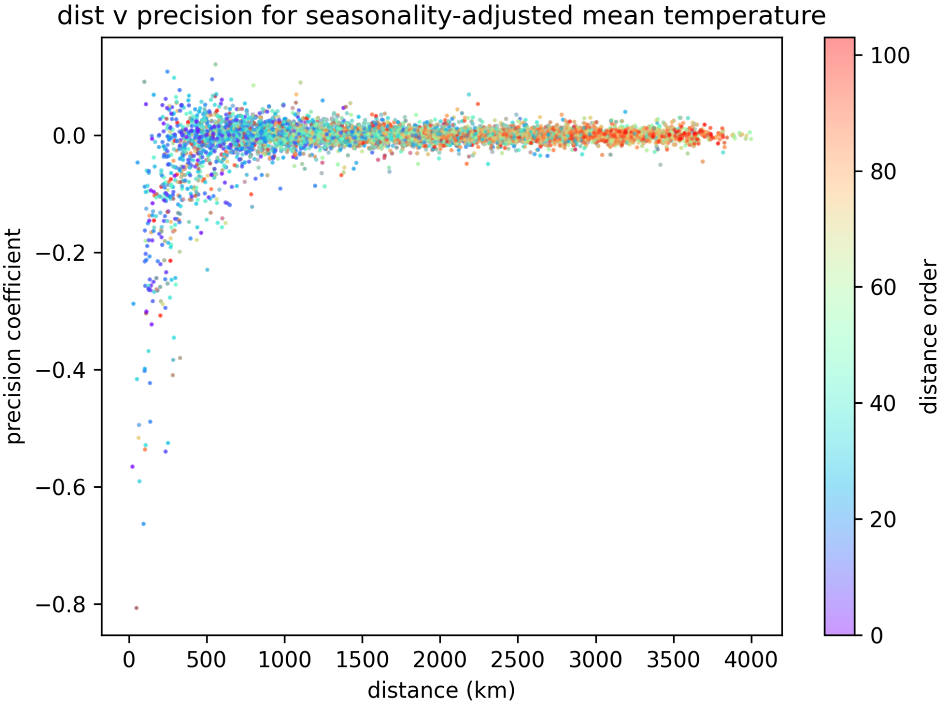

Figure 21: Precision values vs absolute distance and distance order.

is a clear, but noisy trend for nearby points to have more significant negative precision. This is to some degree what we expect, as each precision is a function of all the covariances, and we are attempting to model it as dependent on just one distance. Given that distances are not a perfect corollary to covariances, as discussed in 4.4.1, we also plot precision vs covariance and ”covariance order” in Fig. 22.

As expected, high precision tends to occur with high covariance, but the opposite is not true. Note the stronger relationship here is surprisingly distance vs precision, as it is closer to a one-to-one function. In particular, the significant precisions appear to roughly follow a $-c/d_{ij}$ relationship for a constant $c = 16.25$. These are the off-diagonal weights of the Laplacian operator with inverse distance weights.

We compare matrix structure for the Laplacian with Gaussian weights, as in 4.2 and with inverse distance weights, as appears useful in the precision vs distance graph. We also compare results for the precision matrix found by inverting the sample covariance, and found by inverting the fitted covariance, found by multiplying the correlation model in 4.4.1 by $\sqrt{\text{Var}(X_i)\text{Var}(X_j)}$ using sample variances. RMSE is a poor norm for measuring matrix structure similarity, as it penalises the specific magnitudes significantly, where we mainly care about structure. To ensure that the matrices have similar structure, we stack the entries in a vector and compute the directional agreement, $\frac{\underset{\sim}{u}\cdot\underset{\sim}{v}}{||u||_2||v||_2}$. We compare this directional measure across operators in Fig. 23. From the discussion in 4.5.2, the least-squares filter under the precision matrix is as below.

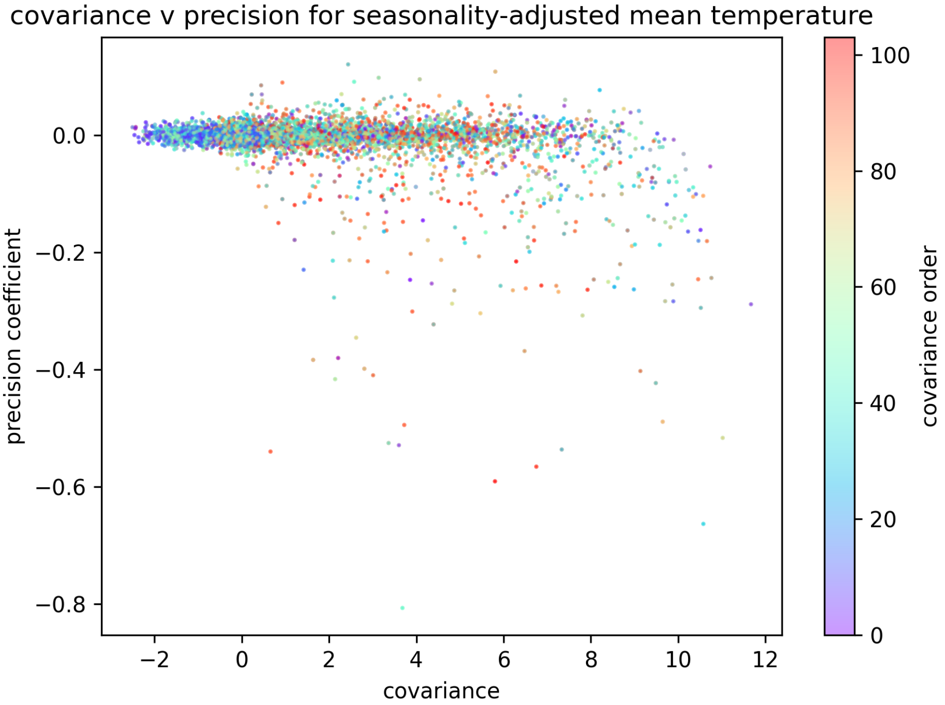

Figure 22: Precision values vs covariance and covariance order.

Operator	Gaussian Laplacian	Inverse Laplacian	Model Precision
Cosine	0.843	0.839	0.126

Figure 23: Structural agreement various operators to the sample precision matrix. Closer to 1 is better. For random vectors, the average measure is 0.

$$H(\lambda) = \frac{1}{1+\lambda\sigma^2}$$

Alternatively, to help the filter cope with innaccuracies to the actual precision matrix, we can use the filter below.

$$H(\lambda) = \frac{\overline{|S(\lambda)|^2}}{\overline{|S(\lambda)|^2} + \sigma^2}$$

Where $\overline{|S(\lambda)|^2}$ is the average signal sample power at each eigenvalue. Of course this method cheats somewhat as it requires knowledge of the true signal. We assess performance of both filters across the operator choices. The filter RMSE or the first measured spectra filter across varying noise levels is given in Fig. 24. The RMSE for the second implicit spectra filter is given in Fig. 25. As expected, the performance is mostly worse, except for the sample precision. In terms of the simply acquired filters (no sampling needed), the Gaussian weighting performs better than the inverse distances. The optimal a priori edge weight selection filter is as of yet unknown, and the superiority of the Gaussian is here unjustified.

$\sigma_{noise}/\sigma_{signal}$ (%)	1.00	5.00	10.0	20.0	50.0	100
Gaussian Laplacian	1.00	4.97	9.84	18.75	37.60	53.51
Inverse Laplacian	1.00	4.98	9.84	18.85	39.08	58.04
Model Precision	1.00	4.97	9.80	18.54	36.72	52.21
Sample Precision	1.00	4.96	9.72	18.09	34.63	48.89

Figure 24: Measured spectra filter RMSE (%) under various operators and noise levels.

$\sigma_{noise}/\sigma_{signal}$ (%)	1.00	5.00	10.0	20.0	50.0	100
Gaussian Laplacian	1.00	5.03	10.15	20.40	42.95	60.28
Inverse Laplacian	1.00	5.36	12.22	28.79	63.32	80.41
Model Precision	1.01	5.74	12.80	25.35	46.99	66.11
Sample Precision	1.00	4.96	9.71	18.09	34.60	48.81

Figure 25: Implicit spectra filter RMSE (%) under various operators and noise levels.

5 Regularisation as Space-Time Graph Selection

5.1 Model Selection with BIC

In trimester one, we investigated models and selected by reduction in RMSE. As any statistician or machine learning engineer would note, a decrease in error does not always mean a good model. It must be weighed against the increase in parameters required to describe the model, to avoid over-fitting. The Bayesian Information Criterion (BIC) is a common and statistically well-justified means of weighing up error (involved via a likelihood function) versus the number of parameters[20]. It is given as follows.

$$BIC = k\ln(n) - 2\ln(\hat{\mathscr{L}})$$

$$k : \text{number of parameters}$$

$$n : \text{number of observations}$$

$$\hat{\mathscr{L}} : \text{maximised model likelihood}$$

Under the presumption of i.i.d. Gaussian residuals, the likelihood function is given below.

$$\mathscr{L} = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(p_i - o_i)^2}{2\sigma^2}}$$

$$p_i : \text{prediction } i$$

$$o_i : \text{observation } i$$

$$\sigma^2 : \text{true error variance}$$

We can then find and maximise the log-likelihood with respect to the true error variance to give the most generous prediction of the likelihood for each model.

$$\begin{aligned}
\ln(\mathscr{L}) &= \sum_i -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma^2) - \frac{(p_i - o_i)^2}{2\sigma^2} \\
&= -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{n}{2\sigma^2}MSE \\
\frac{\partial}{\partial\sigma^2}\ln(\mathscr{L}) &= -\frac{n}{2\sigma^2} + \frac{n}{2(\sigma^2)^2}MSE \\
0 &= -1 + \frac{MSE}{\sigma^2} \\
\sigma^2 &= MSE \\
\ln(\hat{\mathscr{L}}) &= -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(MSE) - \frac{n}{2}
\end{aligned}$$

The proof that this stationary point is a maximum is left to the reader. We can then get the BIC in terms of the MSE and known quantities, rather than the maximised likelihood.

$$BIC = k\ln(n) + n\ln(2\pi) + n\ln(MSE) + n$$

Because we judge models based on smallest BIC w.r.t. MSE and the number of parameters k, terms without these are irrelevant. We thus compare models using the adjusted BIC defined below.

$$\begin{aligned}
BICa &= k\ln(n) + n\ln(MSE) \\
&= k\ln(n) + 2n\ln(RMSE)
\end{aligned}$$

5.2 The $VAR(p)$ model

In trimester one, univariate time domain predictive filters were developed in the form of $AR(p)$ models. These are models of the form below.

$$x_t = \sum_{i=1}^{p} a_i x_{t-i} + \varepsilon_t$$

The multivariate generalisation of this is key in improving on past prediction methods. $VAR(p)$ models are of the form below.

$$\underset{\sim}{x_t} = \sum_{i=1}^{p} \boldsymbol{A}_i \underset{\sim}{x_{t-i}} + \underset{\sim}{\varepsilon_t}$$

Just like $AR(p)$ models, $VAR(p)$ models can be solved with linear regression. We form the matrix $\boldsymbol{A}$ and vector z_t as below to simplify the model.

$$\begin{aligned}
\boldsymbol{A} &= (A_1|A_2|...|A_p) \\
\underset{\sim}{z_t} &= \begin{pmatrix} \underset{\sim}{x_{t-1}} \\ \underset{\sim}{x_{t-2}} \\ \vdots \\ \underset{\sim}{x_{t-p}} \end{pmatrix} \\
\underset{\sim}{x_t} &= \boldsymbol{A}\underset{\sim}{z_t} + \underset{\sim}{\varepsilon_t}
\end{aligned}$$

This can be solved with standard linear regression methods. Note that $VAR(p)$ models for estimating an n dimensional vector at each point in time have pn^2 parameters.
For the seasonality adjusted mean temperature dataset, we improve on RMSE's from trimester one in 3.3 ($2.09°C$ and $2.06°C$ for first and second order AR models respectively) by using $VAR(p)$ models. Models for varying p are compared both by RMSE and BICa in 26.

p	RMSE ($°C$)	BICa ($\times 10^6$)
1	1.72	2.18
2	1.65	2.16
3	1.63	2.29
4	1.63	2.43
5	1.62	2.57

Figure 26: $VAR(p)$ fit RMSE and BICa for seasonality adjusted mean daily temperature prediction.

Based on BICa, the $VAR(2)$ model is best.

5.3 Regularisation as Graph Selection

The full multivariate $VAR(p)$ model considers all relationships between all weather stations. By selecting a graph that imposes a set of relationships on the data, we reduce the number of parameters in the $VAR(p)$ model that need to be specified. Parameter reduction may viably improve BICa score, and provide a more robust predictor. Methods for reducing parameters to avoid over-fitting, which is the purpose of minimising the BICa score are in some cases referred to as regularisation. We will investigate some techniques for regularisation, and how they link to graph selection.

5.3.1 Lasso Regression

A commonly applied regulariser is the Lasso, popularised by Tibshirani[27]. It is popular in part due to its good performance in promoting sparsity, that is in removing parameters by shrinking them to zero. Lasso regression minimises the error plus a penalty term, as shown below.

$$\text{loss} = MSE + \lambda ||\boldsymbol{A}||_F$$

For the $VAR(p)$ model, with $||\boldsymbol{A}||_F = \sum_{i,j} |A_{ij}|$. λ is a free hyperparameter. Lasso regression has no closed-form solution, but it can be iteratively solved using gradient descent, noting that $\frac{d}{dx}|x| = \text{sign}(x)$ is discontinuous at 0. The most common method for dealing with this discontinuity in the gradient is to use proximal gradient descent, which for brevity we will not discuss here. Fig. 27 shows RMSE, BICa, and the total number of parameters for $VAR(2)$ lasso models with varying λ.

λ	RMSE (°C)	BICa ($\times 10^6$)	no. params
0.02	1.66	2.027	10865
0.04	1.67	1.996	7212
0.06	1.67	1.991	5568
0.08	1.68	1.994	4648
0.10	1.69	2.001	4014

Figure 27: $VAR(2)$ lasso fit RMSE, BICa, and number of parameters for seasonality adjusted mean daily temperature prediction.

The best model here in the BICa sense is found with $\lambda = 0.06$, achieving a slightly worse RMSE than the standard $VAR(2)$ model, but with just 5568 parameters, where the original model has 21632. This gives an average of $5568/104 = 53.54$ relationships for each temperature time series.

5.3.2 l_0 Regularisation

Our goal broadly is to minimise the BICa. For a $VAR(p)$ model with $\boldsymbol{A}$ the parameter matrix, the number of parameters is equal to the l_0 norm of the matrix, $||\boldsymbol{A}||_0$. Then the model BICa to be minimised becomes the expression below.

$$\begin{aligned} BICa &= ||\boldsymbol{A}||_0 \ln(n) + n\ln(MSE) \\ &\propto \ln(MSE) + \frac{\ln(n)}{n}||\boldsymbol{A}||_0 \end{aligned}$$

This can be seem as a non-linear variant of the traditional l_0 regularised least-squares problem, shown below.

$$\text{loss} = MSE + \lambda||\boldsymbol{A}||_0$$

Because of the non-convexity of the l_0 norm, both these problems are difficult to solve, and even iterative solvers can be prone to instability and getting trapped in local minima. A common approximation is to use coordinate descent algorithms to optimise MSE for each coordinate in the matrix A, and to consider if keeping the coordinate improves or worsens the loss function. We will investigate an approximate method first, before returning to the coordinate descent algorithm.

5.3.3 Masking

The goal of l_0 optimisation is essentially to choose an optimal set of active matrix entries, and then to minimise the MSE with respect only to those entries. i.e. we apply a mask to the matrix and optimise within the mask. We will derive how to optimise this expression.

We may consider the $VAR(p)$ model across all times as below.

$$\begin{aligned}(\underset{\sim}{x_p}|\underset{\sim}{x_{p+1}}|\ldots|\underset{\sim}{x_n}) &= \boldsymbol{A}(\underset{\sim}{z_p}|\underset{\sim}{z_{p+1}}|\ldots|\underset{\sim}{z_n}) + (\underset{\sim}{\varepsilon_p}|\underset{\sim}{\varepsilon_{p+1}}|\ldots|\underset{\sim}{\varepsilon_n})\\ \boldsymbol{X} &= \boldsymbol{AZ} + \boldsymbol{E}\end{aligned}$$

We can then consider the data by rows.

$$\boldsymbol{X}_i = \boldsymbol{A}_i\boldsymbol{Z} + \boldsymbol{E}_i$$

This can be seen as regressing $\underset{\sim}{x_{ti}}$ against the active members of $\boldsymbol{A}_i$. If we denote the vector of active weights in $\boldsymbol{A}_i$ as $\boldsymbol{A}_i^a$, and $\boldsymbol{Z}$ only including rows corresponding to these active entries $\boldsymbol{Z}^{ia}$, we are left with the standard linear regression equation below.

$$\boldsymbol{X}_i = \boldsymbol{A}_i^a\boldsymbol{Z}^{ia} + \boldsymbol{E}_i$$

This can be solved with standard linear regression techniques, i.e. $\boldsymbol{A}_i^a = (\boldsymbol{Z}^{ia}\boldsymbol{Z}^{ia\top})^{-1}\boldsymbol{Z}^{ia}\boldsymbol{X}_i^\top$. We test and compare a variety of methods for choosing masks. Because our data is geographical, distances encode some information about the data. We can threshhold by distances or find nearest neighbours to exploit this geographic information. Alternatively we can use global information about the data - partial correlation describes conditional dependence between time series, so we can threshhold a mask using this quantity. Thirdly, we can perform another method for promoting sparsity, such as Lasso regression, and then reregress using the mask found with Lasso, which necessarily gives a better MSE than the Lasso solution, hence improving BICa. These methods are quantitatively compared for various parameter choices in Fig. 28.

λ	RMSE ($°C$)	BICa ($\times 10^6$)	no. params
distance (km) $< \lambda$			
1200	1.69	2.065	7632
1400	1.68	2.058	9280
1600	1.67	2.060	10832
λ nearest neighbours			
38	1.83	2.359	7904
40	1.83	2.357	8320
42	1.83	2.358	8736
partial correlation $> \lambda$th percentile			
25	1.76	2.175	5408
30	1.75	2.171	6490
35	1.74	2.174	7572
reregression with λ Lasso			
0.06	1.66	1.964	5568
0.08	1.67	1.961	4648
0.10	1.67	1.962	4014

Figure 28: $VAR(2)$ masked fit RMSE, BICa, and number of parameters across various activity conditions and empirically well-chosen parameters.

Based upon this analysis, reregression using the Lasso selector is the best performing model amongst those tested. In second is the distance-selected model, which uses the underlying geographic data to model weight importance. The distance-selected model uses many more active weights achieving the same MSE as the Lasso selection.

5.3.4 l_0 Coordinate Descent

We will now derive the l_0 coordinate descent algorithm. Our goal is to simply express the component of the MSE which is contributed by a particular parameter, keeping all others fixed, and then to minimise the MSE with respect to that parameter. We can then see if keeping the parameter or discarding it is optimal for the l_0 loss function. We proceed as below.

$$
\begin{aligned}
MSE &= \sum_t ||\underset{\sim}{x_t} - \boldsymbol{A}\underset{\sim}{z_t}||_2^2 \\
&= \sum_t (\underset{\sim}{x_t} - \boldsymbol{A}\underset{\sim}{z_t})^\top (\underset{\sim}{x_t} - \boldsymbol{A}\underset{\sim}{z_t}) \\
&= \sum_t ||\underset{\sim}{x_t}||_2^2 - 2\sum_t \underset{\sim}{x_t}^\top \boldsymbol{A}\underset{\sim}{z_t} + \sum_t \underset{\sim}{z_t}^\top \boldsymbol{A}^\top \boldsymbol{A}\underset{\sim}{z_t} \\
&\text{Discarding components not dependent on } \boldsymbol{A} : \\
&= -2\sum_{t,i,j} x_{ti}A_{ij}z_{tj} + \sum_{t,k,l,m} z_{tk}z_{tl}A_{mk}A_{ml} \\
&\text{Considering only components dependent on } A_{ij} : \\
MSE_{ij} &= -2\sum_t x_{ti}A_{ij}z_{tj} + \sum_t z_{tj}^2 A_{ij}^2 + 2\sum_{t,l\neq j} z_{tj}z_{tl}A_{ij}A_{il} \\
&\text{With } \Gamma_{ij} = \sum_t x_{ti}z_{tj}, \Sigma_{ij} = \sum_t z_{ti}z_{tj} \\
&= -2A_{ij}\Gamma_{ij} + A_{ij}^2\Sigma_{jj} + 2A_{ij}\sum_{l\neq j}\Sigma_{jl}A_{il}
\end{aligned}
$$

We can then optimise this expression with respect to A_{ij} to get A_{ij}^*.

$$
\begin{aligned}
\frac{\partial MSE_{ij}}{\partial A_{ij}} &= -2\Gamma_{ij} + 2A_{ij}^*\Sigma_{jj} + 2\sum_{l\neq j}\Sigma_{jl}A_{il} = 0 \\
A_{ij}^* &= \frac{\Gamma_{ij} - \sum_{l\neq j}\Sigma_{jl}A_{il}}{\Sigma_{jj}}
\end{aligned}
$$

The l_0 loss due to this parameter is then either $MSE_{ij}(0) = 0$ if the parameter is ignored, or $MSE_{ij}(A_{ij}^*) + \lambda$ if the optimal choice is made. We can thus compare each of these two options and take the lesser at each step to reduce the l_0 loss. Because the l_0 regression problem is non-convex, this style of optimisation may result in finding a local rather than a global minima. With that noted, in our case testing with both trivial initialisations, such as ones on the diagonal, and more complex ones, like the lasso reregression model, yield very similar results. The most optimal achieved BICa is shown in Fig. 29.

λ	RMSE ($°C$)	BICa ($\times 10^6$)	no. params
0.005	1.66	1.946	3875

Figure 29: l_0 coordinate descent optimal BICa result.

5.3.5 Group l_0

In applications beyond temperature datasets, each vertex may be associated with a number of scalar values. E.g. temperature and pressure. In this case, we can come up with an algorithm for l_0 coordinate descent in the case of vector-values vertices. Where in scalar regression, the prediction at the next time step is given by multiplication of a matrix of scalars with a vector of scalar, in vector regression we can multiply a matrix of matrices with a vector of vectors. This summation is shown below.

$$\underset{\sim}{x_{ti}} = \sum_j \boldsymbol{A}_{ij} \underset{\sim}{x_{tj}}$$

A simple application of this is to consider the vector of values at each point as the values at the past p timesteps. This can be seen as grouping together some of the parameters in the VAR matrix $\boldsymbol{A}$, hence the name group l_0, where the parameters in a group are optimised together and may be zeroed out together. We will derive from the equivalent expression for MSE_{ij} from the previous section the optimal selection of $\boldsymbol{A}_{ij}$.

$$MSE_{ij} = -2\sum_t \underset{\sim}{x_{ti}}^\top \boldsymbol{A}_{ij} \underset{\sim}{z_{tj}} + \sum_t \underset{\sim}{z_{tj}}^\top \boldsymbol{A}_{ij}^\top \boldsymbol{A}_{ij} \underset{\sim}{z_{tj}} + 2\sum_{t,l\neq j} \underset{\sim}{z_{tj}}^\top \boldsymbol{A}_{ij}^\top \boldsymbol{A}_{il} \underset{\sim}{z_{tl}}$$

In order to seperate the components which are time-dependent from the components which are parameter-dependent we can use the trace identity below.

$$\begin{aligned}\sum_t \underset{\sim}{x_t}^T \boldsymbol{M} \underset{\sim}{y_t} &= \sum_t tr(\boldsymbol{M} \underset{\sim}{y_t} \underset{\sim}{x_t}^\top) \\ &= tr(\boldsymbol{M} \sum_t \underset{\sim}{y_t} \underset{\sim}{x_t}^\top)\end{aligned}$$

Using this identity with $\boldsymbol{\Gamma}_{ij} = \sum_t \underset{\sim}{x_{ti}} \underset{\sim}{z_{tj}}^\top$ and $\boldsymbol{\Sigma}_{ij} = \sum_t \underset{\sim}{z_{ti}} \underset{\sim}{z_{tj}}^\top$, we can simplify MSE_{ij} as below.

$$MSE_{ij} = -2tr(\boldsymbol{A}_{ij}\boldsymbol{\Gamma}_{ij}^\top) + tr(\boldsymbol{A}_{ij}^\top \boldsymbol{A}_{ij} \boldsymbol{\Sigma}_{jj}^\top) + 2\sum_{l\neq j} tr(\boldsymbol{A}_{ij}^\top \boldsymbol{A}_{il} \boldsymbol{\Sigma}_{lj})$$

In the simple case of a grouping of VAR parameters in time, we are not predicting the next vector of values in time, but using an input vector to predict a scalar. This simplifies the above expression to the one below for a vector of parameters $\underset{\sim}{A_{ij}}$ rather than a block of such parameters $\boldsymbol{A}_{ij}$. It also makes the covariance between the output scalar and input vector a vector $\underset{\sim}{\Gamma_{ij}}$ rather than $\mathbf{\Gamma}_{ij}$.

$$MSE_{ij} = -2\underset{\sim}{\Gamma_{ij}}^{\top}\underset{\sim}{A_{ij}} + \underset{\sim}{A_{ij}}^{\top}\mathbf{\Sigma}_{jj}\underset{\sim}{A_{ij}} + 2\underset{\sim}{A_{ij}}^{\top}\sum_{l\neq j}\mathbf{\Sigma}_{lj}\underset{\sim}{A_{il}}$$

Taking derivatives and optimising we get the optimal choice of paramters $\underset{\sim}{A_{ij}}^{*}$ as below.

$$\underset{\sim}{A_{ij}}^{*} = \mathbf{\Sigma}_{jj}^{-1}(\underset{\sim}{\Gamma_{ij}} - \sum_{l\neq j}\mathbf{\Sigma}_{lj}\underset{\sim}{A_{il}})$$

Running l_0 coordinate descent with time-domain grouping like this yields very similar results to regular l_0, in a comparable length of time, suggesting that the prediction fidelity gained from seperating these parameters is minimal. The optimal results of the group l_0 coordinate descent are shown in 30.

λ	RMSE ($°C$)	BICa ($\times 10^6$)	no. params
0.01	1.67	1.960	4090

Figure 30: l_0 coordinate descent optimal BICa result.

5.4 Dynamic Modelling with SGD

Online learning often dramatically reduces the number of effective parameters a model has. This is because parameters are learnt on-the-fly only on data seen so far, not on all data. Any parameters than are not fitted on the whole dataset, but only on data prior to that which is predicted with those parameters, are not counted in the BIC calculation. Additionally, online learning can account for changing system conditions in a simple model. The $VAR(p)$ model can be reframed into an online predictor via stochastic gradient descent (SGD). At each time step, we predict the true varying parameters of the $VAR(p)$ model using gradient descent. This gradient descent update is derived below.

$$\begin{aligned}
\underset{\sim}{x_t} &= \hat{\boldsymbol{A}}_t \underset{\sim}{z_t} + \underset{\sim}{\varepsilon_t} \\
0 &= \frac{\partial ||\underset{\sim}{\varepsilon_t}||_2^2}{\partial \hat{\boldsymbol{A}}_t} \\
&= 2\underset{\sim}{\varepsilon_t} \frac{\partial \underset{\sim}{\varepsilon_t}}{\partial \hat{\boldsymbol{A}}_t} \\
&= 2\underset{\sim}{\varepsilon_t} \frac{\partial}{\partial \hat{\boldsymbol{A}}_t} \underset{\sim}{x_t} - \hat{\boldsymbol{A}}_t \underset{\sim}{z_t} \\
&= -2\underset{\sim}{\varepsilon_t}\underset{\sim}{z_t}^{\top} \\
\hat{\boldsymbol{A}}_{t+1} &= \hat{\boldsymbol{A}}_t + 2\eta \underset{\sim}{\varepsilon_t}\underset{\sim}{z_t}^{\top} \\
\eta &: \text{ learning rate}
\end{aligned}$$

This is called an explicit update, because we update the array $\hat{\boldsymbol{A}}_{t+1}$ with the gradient evaluated at the old parameter $\hat{\boldsymbol{A}}_t$. This method is simpler, but is prone to numerical instability. For learning rates that are too high, the errors can grow without bound. Additionally, the optimal learning rate is very dependent not just on the structure of relationships in the data, but also on the magnitude of the data. The model results for this simple form of SGD are shown in Fig. 31.

η ($\times 10^{-5}$)	RMSE ($°C$)	BICa ($\times 10^{6}$)	no. params
2	1.75	2.069	1
4	1.74	2.059	1
6	1.76	2.100	1

Figure 31: $VAR(2)$ SGD fit RMSE, BICa, and number of parameters across learning rates.

One method to make η less data-magnitude dependent is to normalise the gradient by the magnitude of the data vector, $||\underset{\sim}{z_t}||_2$. The results of this normalisation are shown in Fig. 32.

η ($\times 10^{-3}$)	RMSE ($°C$)	BICa ($\times 10^{6}$)	no. params
1.0	1.74	2.047	1
1.5	1.73	2.042	1
2.0	1.74	2.057	1

Figure 32: $VAR(2)$ SGD normalised fit RMSE, BICa, and number of parameters across learning rates.

Both these options achieve a worse RMSE than other models, but due to a low number of parameters, achieve comparable BICa to the distance-selected masking method, but worse than the lasso-selected masking method, both shown in Fig. 28.

5.4.1 Implicit Updates

So far we have discussed explicit gradient updates. For a loss function at time step t given by $Q_t(\boldsymbol{A})$, such updates are expressed as below.

$$\hat{\boldsymbol{A}}_{t+1} = \hat{\boldsymbol{A}}_t - \eta \nabla Q_t(\hat{\boldsymbol{A}}_t)$$

This contrasts to the implicit update below.

$$\hat{\boldsymbol{A}}_{t+1} = \hat{\boldsymbol{A}}_t - \eta \nabla Q_t(\hat{\boldsymbol{A}}_{t+1})$$

This is much less trivial to solve, as we get an equation of the form below.

$$\hat{\boldsymbol{A}}_{t+1} + \eta \nabla Q_t(\hat{\boldsymbol{A}}_{t+1}) = \hat{\boldsymbol{A}}_t$$

Which is not necessarily as easy to solve. We can proceed as below.

$$\begin{aligned}
\hat{\boldsymbol{A}}_t &= \hat{\boldsymbol{A}}_{t+1} + \eta \nabla Q_t(\hat{\boldsymbol{A}}_{t+1}) \\
\nabla Q_t(\hat{\boldsymbol{A}}_{t+1}) &= \nabla || \underset{\sim}{x_t} - \hat{\boldsymbol{A}}_{t+1} \underset{\sim}{z_t} ||_2^2 \\
&= \frac{\partial}{\partial \hat{\boldsymbol{A}}_{t+1}} (\underset{\sim}{x_t} - \hat{\boldsymbol{A}}_{t+1} \underset{\sim}{z_t})^\top (\underset{\sim}{x_t} - \hat{\boldsymbol{A}}_{t+1} \underset{\sim}{z_t}) \\
&= 2(\underset{\sim}{x_t} - \hat{\boldsymbol{A}}_{t+1} \underset{\sim}{z_t})(-\underset{\sim}{z_t}^\top) \\
&= -2\underset{\sim}{x_t}\underset{\sim}{z_t}^\top + 2\hat{\boldsymbol{A}}_{t+1}\underset{\sim}{z_t}\underset{\sim}{z_t}^\top \\
\hat{\boldsymbol{A}}_t + 2\eta \underset{\sim}{x_t}\underset{\sim}{z_t}^\top &= \hat{\boldsymbol{A}}_{t+1} + 2\eta \hat{\boldsymbol{A}}_{t+1}\underset{\sim}{z_t}\underset{\sim}{z_t}^\top \\
\hat{\boldsymbol{A}}_{t+1} &= (\hat{\boldsymbol{A}}_t + 2\eta \underset{\sim}{x_t}\underset{\sim}{z_t}^\top)(\boldsymbol{I} + 2\eta \underset{\sim}{z_t}\underset{\sim}{z_t}^\top)^{-1}
\end{aligned}$$

Normally the large matrix inversion here would be computationally expensive ($O(n^3)$), but because it is a rank-1 update of the identity matrix, i.e. the identity matrix plus an outer product, we can apply the Sherman-Morrison inversion formula, as below.

$$\hat{\boldsymbol{A}}_{t+1} = (\hat{\boldsymbol{A}}_t + 2\eta \underset{\sim}{x_t}\underset{\sim}{z_t}^\top)(\boldsymbol{I} - \frac{2\eta}{1 + 2\eta || \underset{\sim}{z_t} ||_2^2} \underset{\sim}{z_t}\underset{\sim}{z_t}^\top)$$

For brevity we will omit the full working for the simplification, but the expression simplifies to that below.

$$\hat{\boldsymbol{A}}_{t+1} = \hat{\boldsymbol{A}}_t + \frac{2\eta}{1 + 2\eta || \underset{\sim}{z_t} ||_2^2} \underset{\sim}{\varepsilon_t}\underset{\sim}{z_t}^\top$$

Which appears as a stabilised version of explicit SGD, scaling down updates with very large inputs. The results of applying this are very similar to that of explicit SGD and normalised SGD, so we will not discuss this method further.

6 Application of Conditional Mutual Information

As discussed in 3.7, once we choose a prior distribution in parameter space, like the one induced by a VAR model, or a VAR model with l_0 regularisation, or by another prior, and we fit the model to the data, we can determine the conditional mutual information between each pair of timeseries. Once we fit our model with the maximum likelihood estimate, we can evaluate the conditional mutual information between each pair of timeseries, which gives a clearer picture than either correlation or precision can about how interrelated two variables are. Then, from the modelled distribution, we can analytically determine the conditional information between two timeseries modelled by the distribution.

6.1 Calculating CMI for VAR(p) Models

We recall the general VAR(p) model below.

$$\underset{\sim}{x_t} = \sum_{\tau=1}^{p} \boldsymbol{A}_\tau \underset{\sim}{x_{t-\tau}} + \underset{\sim}{\varepsilon_t}$$

The two key features of this model are the autoregression matrices $\boldsymbol{A}_\tau$ and the driving noise covariance $\boldsymbol{\Sigma}_\varepsilon$. We recall the required formulae for conditional mutual information discussed in 3.7, given below in terms of $\boldsymbol{A}_\tau$ and $\boldsymbol{\Sigma}_\varepsilon$.

$$\begin{aligned}
A(j\omega) &= I - \sum_{\tau} e^{-j\omega\tau} \boldsymbol{A}_\tau \\
A(j\omega) &= \Big(g_1(j\omega) \;\vdots\; g_2(j\omega) \;\vdots\; \ldots \;\vdots\; g_N(j\omega) \Big) \\
\rho_{ij|\neq ij} &= \frac{-g_i^H \boldsymbol{\Sigma}_\varepsilon g_j}{\sqrt{(g_i^H \boldsymbol{\Sigma}_\varepsilon g_i)(g_j^H \boldsymbol{\Sigma}_\varepsilon g_j)}} \\
R(\omega) &= \frac{-F^{-1}(\omega)}{\sqrt{diag(F^{-1}(\omega)) diag(F^{-1}(\omega))^T}} \\
CMI_{ij} &= -\frac{1}{2} \int_{\pi}^{\pi} \ln(1 - |\rho_{ij\neq ij}|^2) \frac{d\omega}{2\pi}
\end{aligned}$$

Where the square root and division are performed elementwise for $\rho_{ij|\neq ij}$.

$$CMI_{ij} = H(X_i|X_{\neq i,j}) + H(X_j|X_{\neq i,j}) - H(X_i,X_j|X_{\neq i,j})$$

To this end, we need to describe the distributions $X_i, X_j|X_{\neq i,j}$ and $X_i|X_{\neq i,j}$ within the VAR model. These quanties follow typical marginalition rules and conditioning for a multivariate normal distribution following the error covariance. We derive the distributions below, for $\mathbf{\Omega}$ the precision matrix of the error, and $\mathbf{\Omega}_{a,b}$ the submatrix of precisions of set of variables a with set b. We denote collections of timeseries as vectors, with the whole collection denoted $\underset{\sim}{X}$.

$$\begin{aligned}
P(\underset{\sim}{X}) &= \prod_t P(\underset{\sim}{\varepsilon^t}) \\
P(\underset{\sim}{\varepsilon^t}) &\sim \mathscr{N}(\underset{\sim}{0}, \mathbf{\Omega}) \\
\underset{\sim}{\mu^*} &= -\mathbf{\Omega}_{ij,ij}^{-1}\mathbf{\Omega}_{ij,\neq ij}\underset{\sim}{\varepsilon_{\neq i,j}} \\
P(\varepsilon_i^t, \varepsilon_j^t|\underset{\sim}{\varepsilon_{\neq i,j}^t}) &\sim \mathscr{N}(\underset{\sim}{\mu^*}, \mathbf{\Omega}_{ij,ij}) \\
P(X_i, X_j|X_{\neq i,j}) &= \prod_t P(\varepsilon_i^t, \varepsilon_j^t|\underset{\sim}{\varepsilon_{\neq i,j}^t}) \\
P(\varepsilon_i^t|\underset{\sim}{\varepsilon_{\neq i,j}^t}) &= \mathscr{N}(\underset{\sim}{\mu^*}, (\mathbf{\Omega}_{ij,ij}^{-1})_{i,i}^{-1}) \\
P(X_i|X_{\neq i,j}) &= \prod_t P(\varepsilon_i^t|\underset{\sim}{\varepsilon_{\neq i,j}^t})
\end{aligned}$$

Note that one benefit of using the precision matrix rather than the covariance matrix is that the expressions for the precisions of the desired distributions are very simple and very low complexity to compute from an estimate of the precision matrix. Now, we derive the expression for the relevant entropies, with $\mathscr{X}$ denoting the domain of time series, i.e. functions from $[0..n] \rightarrow \mathbb{R}$.

$$H(X_i|X_{\neq i,j}) = \int_{\mathscr{X}} P(X_i|X_{\neq i,j}) \log(P(X_i|X_{\neq i,j}))dX_i$$

Because there is a one-to-one linear mapping between trajectories X_i and error signals ε_i, we can make a change of variables from $X_i \rightarrow \varepsilon_i$, over the same domain. The proof of this is irrelevant to the point here, so is omitted. The errors are independent at each time step, and entropy is linear for independent variables, so the entropy is equivalent to n times the entropy of a single distribution of the error, as below.

$$H(X_i|X_{\neq i,j}) = n\int_{\mathbb{R}} P(\varepsilon_i^t|\underset{\sim}{\varepsilon_{\neq i,j}^t}) \log(P(\varepsilon_i^t|\underset{\sim}{\varepsilon_{\neq i,j}^t}))d\varepsilon_i$$

There is a known expression for the entropy of a multivariate Gaussian in d variables. We use this below to find this entropy.

$$\begin{aligned}
H(\mathcal{N}(\underset{\sim}{\mu},\boldsymbol{\Omega})) &= \frac{1}{2}\log((2\pi e)^d|\boldsymbol{\Omega}|^{-1}) \\
H(X_i|X_{\neq i,j}) &= \frac{n}{2}\log((2\pi e)^d(\boldsymbol{\Omega}^{-1}_{ij,ij})_{i,i}) \\
H(X_j|X_{\neq i,j}) &= \frac{n}{2}\log((2\pi e)^d(\boldsymbol{\Omega}^{-1}_{ij,ij})_{j,j})
\end{aligned}$$

The expression for the remaining entropy is similar.

$$H(X_i,H_j|X_{\neq i,j}) = \frac{n}{2}\log((2\pi e)^d|\boldsymbol{\Omega}_{ij,ij}|^{-1})$$

Then, the CMI is given as below.

$$\begin{aligned}
CMI_{ij} &= H(X_i|X_{\neq i,j}) + H(X_j|X_{\neq i,j}) - H(X_i,X_j|X_{\neq i,j}) \\
&= \frac{n}{2}\log((2\pi e)(\boldsymbol{\Omega}^{-1}_{ij,ij})_{i,i}) + \frac{n}{2}\log((2\pi e)(\boldsymbol{\Omega}^{-1}_{ij,ij})_{j,j}) - \frac{n}{2}\log((2\pi e)^2|\boldsymbol{\Omega}_{ij,ij}|^{-1}) \\
&= \frac{n}{2}\log((\boldsymbol{\Omega}^{-1}_{ij,ij})_{i,i}(\boldsymbol{\Omega}^{-1}_{ij,ij})_{j,j}|\boldsymbol{\Omega}_{ij,ij}|)
\end{aligned}$$

There is a well-known formula relating the inverse of a 2x2 matrix in terms of its determinant and other elements of the matrix, through the cofactors of the matrix. This is shown below.

$$\begin{aligned}
\boldsymbol{A} &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \\
\boldsymbol{A}^{-1} &= \frac{1}{|\boldsymbol{A}|}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}
\end{aligned}$$

This allows us to simplify the CMI equation as below.

$$\begin{aligned}
CMI_{ij} &= \frac{n}{2}\log(\boldsymbol{\Omega}_{j,j}\boldsymbol{\Omega}_{i,i}\frac{|\boldsymbol{\Omega}_{ij,ij}|}{|\boldsymbol{\Omega}_{ij,ij}|^2}) \\
&= \frac{n}{2}\log(\frac{\boldsymbol{\Omega}_{j,j}\boldsymbol{\Omega}_{i,i}}{|\boldsymbol{\Omega}_{ij,ij}|}) \\
&= \frac{n}{2}\log(\frac{\boldsymbol{\Omega}_{jj}\boldsymbol{\Omega}_{ii}}{\boldsymbol{\Omega}_{jj}\boldsymbol{\Omega}_{ii} - \boldsymbol{\Omega}^2_{ij}})
\end{aligned}$$

Bibliography

[1] Acemoglu D., et. al. systemic risk and stability in financial networks. *National Bureau of Economic Research*, 2013.

[2] Barry R., Chorley R. *Atmosphere, Weather, and Climate (9th ed.)*. Routledge, 2010.

[3] Bertil Matérn. *Spatial Variation*. Reports of the Swedish Institute of Experimental Forestry, 1960.

[4] Cooley J., Tukey J. An algorithm for the machine calculation of complex fourier series. *American Mathematical Society*, 1965.

[5] Cressie N., Moores M. Spatial statistics. *National Institute for Applied Statistics Research Australia*, 2021.

[6] Euler L. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Petropolitanae*, pages 128–140, 1741.

[7] Freedman, David A. *Statistical Models: Theory and Practice*. Cambridge University Press, 2009.

[8] Friedman, Jerome and Hastie, Trevor and Tibshirani, Robert. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2008.

[9] Grady L., Alvino C. Reformulating and optimizing the mumford-shah functional on a graph — a faster, lower energy solution. *Siemens Corporate Research*, 2008.

[10] Grady L., Polimeni J. Discrete calculus. *Springer*, 2010.

[11] Imrich W., Peterin I. Recognizing cartesian products in linear time. *Elsevier*, 2005.

[12] Krige D. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical Metallurgical & Mining Society of South Africa*, 1951.

[13] Loan C. The ubiquitous kronecker product. *Cornell University*, 1999.

[14] Meinshausen N., Buhlmann P. High-dimensional graphs and variable selection with the lasso. *arXiv:math/0608017v*, 2006.

[15] Moura J., Sandryhaila A. Discrete signal processing on graphs. *IEEE*, 2012.

[16] Moura J., Sandryhaila A. Big data analysis with signal processing on graphs. *IEEE Signal Processing Magazine*, pages 80–90, 2014.

[17] NOAA. Global surface temperature data, 2025. https://www.ncei.noaa.gov/pub/data/ghcn/daily/.

[18] Bureau of Meteorology. Acorn sat data set, 2023. http://www.bom.gov.au/climate/data/acorn-sat/, Accessed 22/03/2025.

[19] Page L. The pagerank citation ranking: bringing order to the web. *Stanford Digital Library Project*, 1998.

[20] Schwarz, Gideon E. *Estimating the dimension of a model.* Annals of Statistics, 1978.

[21] Stanković L., et. al. Vertex-frequency graph signal processing: A comprehensive review. *Elsevier*, 2020.

[22] Stankovic L., et. al. Vertex-frequency graph signal processing: A comprehensive review. *Elsevier*, 2020.

[23] Stigler, Stephen M. *The History of Statistics: The Measurement of Uncertainty before 1900.* Cambridge: Harvard, 1986.

[24] Sun M., et. al. Graph neural networks: A review of methods and applications. *AI Open*, 2020.

[25] T. Park, G. Casella. The bayesian lasso. *Journal of the American Statistical Association*, 2008.

[26] Taubin G., et. al. Optimal surface smoothing as filter design. *IBM T.J. Watson Research Center*, 1996.

[27] Tibshirani, Robert. *Regression Shrinkage and Selection via the lasso.* Journal of the Royal Statistical Society, 1996.

[28] Toal D., et. al. Kriging hyperparameter tuning strategies. *University of Southampton*, 2020.

[29] Tony C., et. al. A constrained l1 minimization approach to sparse precision matrix estimation. *arXiv:1102.2233v1*, 2011.

[30] Various. Dijkstra's algorithm, 2025. https://en.wikipedia.org/wiki/Dijkstra's_algorithm,Accessed20/04/2025.

[31] Various. Floyd-warshall algorithm, 2025. https://en.wikipedia.org/wiki/Floyd-Warshall_algorithm,Accessed20/04/2025.

[32] Various. Jpeg, 2025. https://en.wikipedia.org/wiki/JPEG, Accessed 12/04/2025.

[33] Various. Minimum mean square error, 2025. `https://en.wikipedia.org/wiki/Minimum_mean_square_error`, Accessed 12/04/2025.

[34] Whittle P. On stationary processes in the plane. *Applied Mathematics Laboratory, New Zealand Department of Scientific and Industrial Research*, 1954.

[35] Whittle P. Stochastic-processes in several dimensions. *Bulletin of the International Statistical Institute*, pages 974–994, 1963.

[36] Wiener N. *Extrapolation, interpolation, and smoothing of stationary time series.* MIT Press, 1949.

[37] Wikipedia. Stochastic process, 2025. `https://en.wikipedia.org/wiki/Stochastic_process`, Accessed 22/03/2025.

[38] Z. Yue, P. Sundaram, V. Solo. Fast block-sparse estimation for vector networks. *School of Electrical Engineering and Telecommunications, UNSW, Sydney, AUSTRALIA, Martinos Center for Biomedical Imaging, Harvard Medical School, Charlestown, MA, USA*, 2020.

Appendix A - Python Code

All figures in this report were produced using python code available at: `https://github.com/jimaginary/ELECHONS`.