Topic Title: Graph Signal Processing

Student Name: Zac Petersen **Student ID: 5416118**

A. **Problem statement**

Traditional digital signal processing provides techniques for compression, prediction,
filtration, and interpolation. The domain of application is limited to signals on
periodic, directed, discrete domains. This domain can be seen as a directed graph
about a loop. This thesis asks how we may extend these techniques to arbitrary
graphs, such that the domain of application may be extended.

B. **Objective**

The objective of this thesis is to map traditional signal processing theory onto
arbitrary graphs, in order to get the benefits of traditional signal processing like
compression, prediction, filtration, and interpolation on these arbitrary domains.

C. **My solution**

Comparison of graph fourier transforms for parameter-free compression and filtration.
Continuous domain autoregression equations.
Anisotropic distance-correlation fitting of autoregressive equations.
Application of fit to interpolation.
Arbitrary methods for prediction.
Bandpass conditional mutual information for parameterised edge selection.

D. **Contributions** (at most one per line, most important first)

Method to fit anisotropies and multiple diffusion on irregularly sampled systems.
Interpretation of bandpass conditional mutual information for edge selection.
Efficient implementation of algorithm in reference [7].
Comparison of prediction techniques on multivariate temporal data via BIC.
Critique of inverse covariance modelling.
Critique of overuse of BIC leading to Goodhart effects.

E. **Suggestions for future work**

Modelling non-stationary irregularly sampled spatial systems.
Development of cheap proxies for CMI for edge selection on very large graphs.
Application of graph Fourier transforms and filtration to real control processes.

While I may have benefited from discussion with other people, I certify that this report is entirely my own work, except where appropriately documented acknowledgements are included.

Signature:  Date: 22 / 11 / 2025

Pointers

List relevant page numbers in the column on the left. Be precise and selective: Don't list all pages of your report!

7	Problem Statement
7	Objective

Theory (up to 5 most relevant ideas)

15-17	Continuous-domain spatial stochastic processes.
24	Conditional mutual information.
31-33	Spatial autoregressive processes.
38-41	Extending the Matern kernel to deal with anisotropies and negative correlations.
10-12	Graph products and fast GFTs

Method of solution (up to 5 most relevant points)

38-42	Anisotropic distance-correlation fitting of autoregressive equations.
47-48	BIC model selection.
42-44	Comparison of graph wiener filters.

Contributions (most important first)

38-42	Method to fit correlation on irregularly sampled systems.
62-66	Interpretation of bandpass CMI for edge selection.
26	Efficient implementation of algorithm in reference [7].
48-55	Prediction comparison on multivariate temporal data via BIC.
72-75	Critique of inverse covariance modelling.
58-59	Critique of overuse of BIC leading to Goodhart effects.

My work

32-35,40	System block diagrams/algorithms/equations solved
47-48	Description of assessment criteria used
43,51,53, 60	Description of procedure (e.g. for experiments)

Results

36,42,47,65	Succinct presentation of results
45-46,65,72-75	Analysis
72-75	Significance of results

Conclusion

77	Statement of whether the outcomes met the objectives
76	Suggestions for future research

Literature: (up to 5 most important references)

10-12	[18] Moura J., Sandryhaila A. 2014.
16	[38] Whittle P. 1963.
17	[14] Krige D. 1951.
24-25	[12] I. Gel'fand and A. Yaglom, 1959

Graph Signal Processing

Zac Petersen

School of Electrical Engineering and Telecommunications

Submitted November 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 review of a variety of works 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 [14]. 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 [18], prediction [14], filtering [29], and computer vision [10] on graphs. Such techniques have been relevant to highly influential areas, including advertising and ranking as in the Google PageRank algorithm [21], 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 [27]. We will explore examples and replicate some of these techniques both on and off graphs. We will also seek to develop new techniques in these areas in order to improve on results.

In this report, we replicate techniques and empirically confirm results both with synthetic data sets and a real data set. We also develop a number of new techniques which we apply to these datasets. Two real datasets are used: homogenised daily mean temperature data across Australia as reported by the Bureau of Meteorology (BOM) [20], 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) [19]. 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 [18][24]. 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 [11].

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 [24]. 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} [18]. 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.

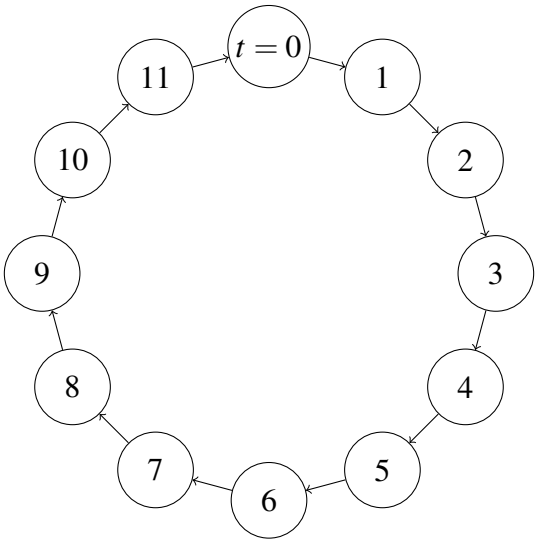

Figure 1: A graph for a time-domain with 12 samples.

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.

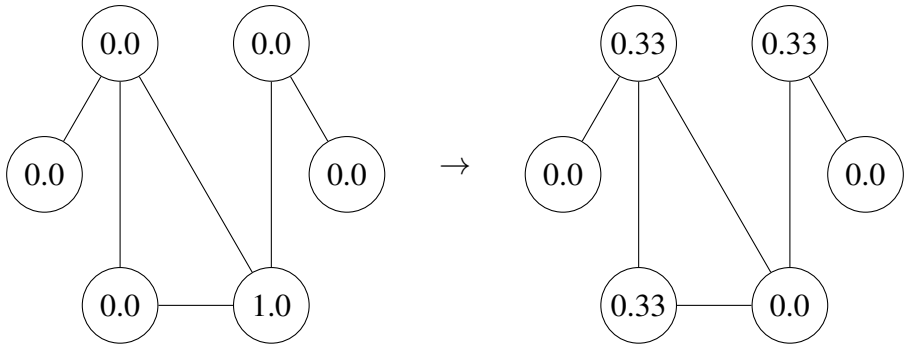

Figure 2: A impulse spreading to nearby nodes under a particular shift operator.

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 [18], 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,

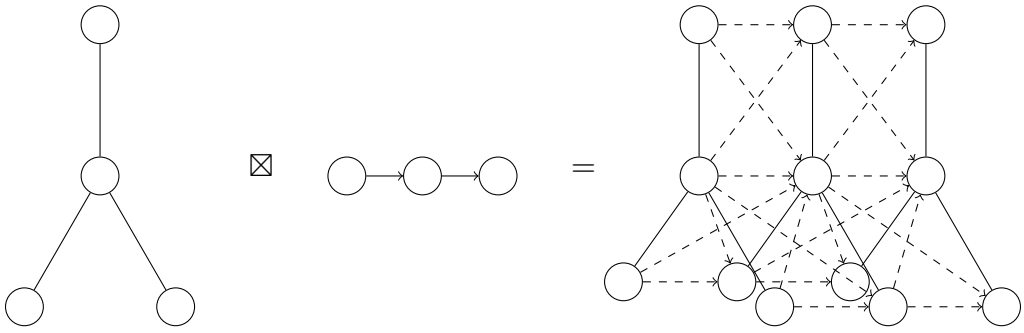

Figure 3: The strong product of two graphs.

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 [15].

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

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 [13]. 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 [15], 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 [35].

Moura applies the fast GFT to compress spatiotemporal daily temperature data across the United States for 150 stations across 1 year (365 measurements) [18]. 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 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 [17], 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 4.6.3.

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 [33] in $O((V+E)\log E)$, or all distances may be found via the Floyd-Warshall algorithm [34] in $O(V^3)$ (for V the number of vertices and E the number of edges). Another prominent method, highlighted by Stankovic et. al. [25], 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 4.6.2 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 [40].

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 [39].

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 [37]. 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 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 4.4.3 [38]. 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 [14]. 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 [31].

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.

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

This is the estimate implemented 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.

$$\text{Var}(\hat{X}_0 - X_0) = \begin{pmatrix} \hat{w}_{\sim}^{\top} & -1 \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma} & \underset{\sim}{\Gamma} \\ \underset{\sim}{\Gamma}^{\top} & \text{Var}(X_0) \end{pmatrix} \begin{pmatrix} \hat{w}_{\sim} \\ -1 \end{pmatrix}$$

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 [26]. 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.

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

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 $O(nk^2 + k^3 + nkl + k^2 l) = O(k^2(n+k) + kl(n+k)) = O(k(k+l)(n+k))$. 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, $\boldsymbol{X}^{\top}\boldsymbol{X} \approx \underset{\sim}{\boldsymbol{\Sigma}}_x$, the covariance matrix of $\underset{\sim}{x}$, and $\boldsymbol{X}^{\top}\boldsymbol{Y} \approx \underset{\sim}{\underset{\sim}{\boldsymbol{\Gamma}}}_{xy}$, 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 [36].

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

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. [9] 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.

$$\mathcal{N}(\underset{\sim}{\mu},\mathbf{\Sigma}) \sim (2\pi)^{-k/2}\det(\mathbf{\Sigma})^{-1/2}\exp(-\frac{1}{2}(\underset{\sim}{x}-\underset{\sim}{\mu})^{\top}\mathbf{\Sigma}^{-1}(\underset{\sim}{x}-\underset{\sim}{\mu}))$$

$$P(\underset{\sim}{x_0},\underset{\sim}{x_1},\ldots,\underset{\sim}{x_n}) = \prod_{i=0}^{n}(2\pi)^{-k/2}\det(\mathbf{\Sigma})^{-1/2}\exp(-\frac{1}{2}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^{\top}\mathbf{\Sigma}^{-1}(\underset{\sim}{x_i}-\underset{\sim}{\mu}))$$

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

up to a constant:

$$\begin{aligned}&= \frac{n}{2}(\ln\det(\mathbf{\Sigma}^{-1})-\frac{1}{n}\sum_{i=0}^{n}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^{\top}\mathbf{\Sigma}^{-1}(\underset{\sim}{x_i}-\underset{\sim}{\mu}))\\ &= \frac{n}{2}(\ln\det(\mathbf{\Sigma}^{-1})-\mathrm{tr}(\mathbf{\Sigma}^{-1}\frac{1}{n}\sum_{i=0}^{n}(\underset{\sim}{x_i}-\underset{\sim}{\mu})^{\top}(\underset{\sim}{x_i}-\underset{\sim}{\mu})))\\ &= \frac{n}{2}(\ln\det(\mathbf{\Sigma}^{-1})-\mathrm{tr}(\mathbf{\Sigma}^{-1}S))\end{aligned}$$

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 [28] 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.

This method quantifies the graphical relationship between scalar random variables for which many samples are available. The methods do not describe how to handle relationships in a graph with time series or vectors at each vertex, where relationships may include delayed or multivariate effects. Examples of such real world time-series 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.

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. [41] come up with a fast algorithm for approximate solving of the L0 regularised vector inverse covariance tensor approximation. As it is relevant to this work, we will derive and implement a simpler version of this algorithm, namely the fast algorithm in the case of a regular matrix with scalar entries.

3.6.1 Fast L0 Spare Inverse Covariance

First, we recall the log-likelihood function for the error precision matrix $\boldsymbol{\Omega}_\varepsilon$ in terms of the sample error covariance $\boldsymbol{S}$ and number of samples n.

$$\mathscr{L}\mathscr{L} = n\log\det\boldsymbol{\Omega} - n\mathrm{tr}(\boldsymbol{S}\boldsymbol{\Omega})$$

Then, we wish to minimise the BIC of the log-likelihood for the number of parameters k, as below.

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

This can be expressed as an l_0 regression. To simplify things for the algorithm, we presume that the diagonal elements of $\boldsymbol{\Omega}$ are all non-zero. We can then express the function $F(\boldsymbol{\Omega})$ that we wish to minimise as below.

$$F(\boldsymbol{\Omega}) \propto -\log\det\boldsymbol{\Omega} + \mathrm{tr}(\boldsymbol{S}\boldsymbol{\Omega}) + \frac{\ln n}{2n}\sum_{i\neq j} I(\boldsymbol{\Omega}_{ij} \neq 0)$$

For simplicity, we call $\frac{\ln n}{2n} = \lambda$, as in a typical l_0 regression. Note that this comes with a number of parameters equal to $\frac{1}{2}\sum i \neq jI(\boldsymbol{\Omega}_{ij} \neq 0)$, because the precision matrix is symettric.

One of the key insights of the coordinate descent optimisation algorithm employed here, is to consider optimising row i and column i together, keeping everything else constant. This contrasts with other algorithms that may optimise either rows individually, columns individually, or entries ij in the matrix individually. Because each set of row and column i represents the relationships with variable i, we can swap this row and column with the last row and column in the matrix to simplify the expressions for the matrix structure. After optimisation, the rows and columns can be moved back to where they belong. We hence partition the matrix $\boldsymbol{\Omega}$ to be

optimised and the sample covariance $\boldsymbol{S}$ as below.

$$\boldsymbol{\Omega} = \begin{pmatrix} \boldsymbol{\Omega}_{-i} & \underset{\sim}{\Omega}_{-i,i} \\ \underset{\sim}{\Omega}_{-i,i}^{\top} & \omega_{i,i} \end{pmatrix}$$

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_{-i} & \underset{\sim}{S}_{-i,i} \\ \underset{\sim}{S}_{-i,i}^{\top} & s_{i,i} \end{pmatrix}$$

We also introduce a similar partitioning on the estimate covariance, $\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1}$.

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{-i} & \underset{\sim}{\Sigma}_{-i,i} \\ \underset{\sim}{\Sigma}_{-i,i}^{\top} & \sigma_{i,i} \end{pmatrix}$$

We can then rewrite the objective function as below.

$$\begin{aligned} F(\boldsymbol{\Omega}) = &-\log\det(\boldsymbol{\Omega}_{-i}) - \log(\omega_{i,i} - \underset{\sim}{\Omega}_{-i,i}^{\top}\boldsymbol{\Omega}_{-i}^{-1}\underset{\sim}{\Omega}_{-i,i}) \\ &+ \operatorname{tr}(\boldsymbol{S}_{-i}\boldsymbol{\Omega}_{-i}) + 2\underset{\sim}{S}_{-i,i}\cdot\underset{\sim}{\Omega}_{-i,i} + \omega_{i,i}s_{i,i} \\ &+ \lambda\sum_{k\neq l} I(\boldsymbol{\Omega}_{-i,kl} \neq 0) + 2\lambda\sum_{k} I(\underset{\sim}{\Omega}_{-i,i,k} \neq 0) \end{aligned}$$

Removing terms independent of variable i :

$$\begin{aligned} F_i(\omega_{i,i}, \underset{\sim}{\Omega}_{-i,i}) = &-\log(\omega_{i,i} - \underset{\sim}{\Omega}_{-i,i}^{\top}\boldsymbol{\Omega}_{-i}^{-1}\underset{\sim}{\Omega}_{-i,i}) \\ &+ 2\underset{\sim}{S}_{-i,i}\cdot\underset{\sim}{\Omega}_{-i,i} + \omega_{i,i}s_{i,i} + 2\lambda\sum_{k} I(\underset{\sim}{\Omega}_{-i,i,k} \neq 0) \end{aligned}$$

We then minimise with respect first to $\omega_{i,i}$.

$$\begin{aligned} \frac{\partial F_i}{\partial \omega_{i,i}} &= -(\omega_{i,i} - \underset{\sim}{\Omega}_{-i,i}^{\top}\boldsymbol{\Omega}_{-i}^{-1}\underset{\sim}{\Omega}_{-i,i})^{-1} + s_{i,i} \\ \omega_{i,i}^{*} &= \underset{\sim}{\Omega}_{-i,i}^{\top}\boldsymbol{\Omega}_{-i}^{-1}\underset{\sim}{\Omega}_{-i,i} + s_{i,i}^{-1} \end{aligned}$$

We then substitute this minimisation into F_i such that we may minimise with respect to $\underset{\sim}{\Omega}_{-i,i}$.

$$\begin{aligned}
F_i(\omega_{i,i}, \underset{\sim}{\Omega}_{-i,i}) &= -\log(s_{i,i}^{-1}) + 2\underset{\sim}{S}_{-i,i} \cdot \underset{\sim}{\Omega}_{-i,i} \\
&\quad + (\underset{\sim}{\Omega}_{-i,i}{}^{\top} \mathbf{\Omega}_{-i}^{-1} \underset{\sim}{\Omega}_{-i,i} + s_{i,i}^{-1}) s_{i,i} + 2\lambda \sum_k I(\underset{\sim}{\Omega}_{-i,i,k} \neq 0) \\
&\quad \text{Removing constant terms:} \\
F_i(\omega_{i,i}, \underset{\sim}{\Omega}_{-i,i}) &= 2\underset{\sim}{S}_{-i,i} \cdot \underset{\sim}{\Omega}_{-i,i} + \underset{\sim}{\Omega}_{-i,i}{}^{\top} \mathbf{\Omega}_{-i}^{-1} \underset{\sim}{\Omega}_{-i,i} s_{i,i} \\
&\quad + 2\lambda \sum_k I(\underset{\sim}{\Omega}_{-i,i,k} \neq 0)
\end{aligned}$$

And then we minimise this expression with respect to the j^{th} entry of $\underset{\sim}{\Omega}_{-i,i}$. To avoid indicies getting confusing, we rename this vector $\underset{\sim}{B}^i$. For each entry of $\underset{\sim}{B}^i$, $\underset{\sim}{B}^i_j$, we can optimise its contribution to F_i, F_{ij}.

$$F_{ij}(B_j^i) = \begin{cases} 2s_{i,j}B_j^i + 2s_{i,i}B_j^i(\Omega_{-i}^{-1})_{j,-j} \cdot \underset{\sim}{B}^i_{-j} + s_{i,i}(B_j^i)^2(\Omega_{-i}^{-1})_{j,j} & B_j^i \neq 0 \\ 2\lambda & B_j^i = 0 \end{cases}$$

$$\begin{aligned}
&\text{if } B_j^i \neq 0: \\
\frac{\partial F_{ij}}{\partial B_j^i} &= 2s_{i,j} + 2(\Omega_{-i}^{-1})_{j,-j} \cdot \underset{\sim}{B}^i_{-j} s_{i,i} + 2s_{i,i}B_j^i(\Omega_{-i}^{-1})_{j,j} \\
(B_j^i)^* &= -s_{i,i}^{-1}(\Omega_{-i}^{-1})_{j,j}^{-1}(s_{i,j} + (\Omega_{-i}^{-1})_{j,-j} \cdot \underset{\sim}{B}^i_{-j} s_{i,i}) \\
F_{ij}^* &= \min(F_{ij}((B_j^i)^*), 2\lambda)
\end{aligned}$$

We can then choose each B_j^i accordingly.

The key speed up realised in the paper involves the difficulty of recalculating $\mathbf{\Omega}_{-i}^{-1}$ for the relevant value of i after some entries have changed from a previous minimisation step. A general formula is leveraged for the relevant part of the inverse of a matrix in terms of a matrix smaller by one row and column.

$$\begin{aligned}
A &= \begin{pmatrix} A_0 & \underset{\sim}{v} \\ \underset{\sim}{v}^{\top} & u \end{pmatrix} \\
A^{-1} &= \begin{pmatrix} A_0^{-1} + A_0^{-1}\underset{\sim}{v}s^{-1}\underset{\sim}{v}^{\top}A_0^{-1} & -A_0^{-1}\underset{\sim}{v}s^{-1} \\ -s^{-1}\underset{\sim}{v}^{\top}A_0^{-1} & s^{-1} \end{pmatrix} \\
&\text{with } s = u - \underset{\sim}{v}^{\top}A_0^{-1}\underset{\sim}{v}
\end{aligned}$$

The partitions of the inverse can be related as below.

$$B = A^{-1} = \begin{pmatrix} B_0 & \underset{\sim}{w} \\ \underset{\sim}{w}^\top & x \end{pmatrix}$$
$$B_0 = A_0^{-1} + \underset{\sim}{w} x^{-1} \underset{\sim}{w}^\top$$
$$A_0^{-1} = B_0 - \underset{\sim}{w} x^{-1} \underset{\sim}{w}^\top$$

This formula tells us exactly how to update A_0^{-1} cheaply if we can compute the entire inverse A^{-1}. Also, if we update B, these formulas tell us how to cheaply update A_0^{-1} and the whole A.

The overall coordinate descent algorithm thus works as follows.

- Input: S, the sample covariance
- Initialise $\mathbf{\Omega}$, e.g. to the inverse of the diagonal entries of S, and compute (trivially) $\mathbf{\Sigma} = \mathbf{\Omega}^{-1}$
- Choose a variable i.
- Compute $\mathbf{\Omega}_{-i}^{-1}$ cheaply using the formula in terms of components of $\mathbf{\Sigma}$.
- Optimise the choice of the row/column B^i of the precision matrix.
- Optimise the choice of $\omega_{i,i}$ in the precision matrix.
- Update $\mathbf{\Sigma}$ cheaply.
- Repeat across all variables.
- Repeat until convergence.

The optimisation of each entry of the precision matrix with this method is $O(n)$, for n the matrix size.

This algorithm is implemented and used in 6.2.

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.

The concept of mutual information was first discussed by Claude Shannon [23], although the modern terminology came later. A phenomenal result for the calculation of CMI for stationary timeseries, even with a single sample, comes from I. Gel'fand and A. Yaglom [12], expressed below.

$$CMI(x_i(t), x_j(t) | x_{\neq i,j}(t)) = -\frac{T}{2} \int_{-\pi}^{\pi} \ln(1 - |\rho_{i,j|\neq ij}(\omega)|^2) \frac{d\omega}{2\pi}$$

With $\rho_{i,j|\neq ij}(\omega)$ the correlation between the ω frequency components of i and j after regression on the other components. This quantity can be expressed as below.

$$\rho_{i,j|\neq ij}(\omega) = \frac{F_{ij|\neq ij}(\omega)}{\sqrt{F_{ii|\neq ij} F_{jj|\neq ij}}}$$

For $F_{ij|\neq ij}$ the cross frequency spectrum between i and j after regression of other signals, and $F_{ii|\neq ij}$ and $F_{jj|\neq ij}$ the power spectrums after regression of other signals. This produces a viable method for finding the information between two stationary signals.

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 [20]. 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 $\pm 0.05\%$ 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. A significant advantage of inverse distance weights over Gaussian weights is that they do not have a scale parameter, or even if distance is scaled, the GFT does not change, as the adjacency matrix only experiences a scalar multiplication. This makes the inverse distance model parameter-free for spatial data. This is addressed further when we discuss graph filters in 4.6.1.

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.

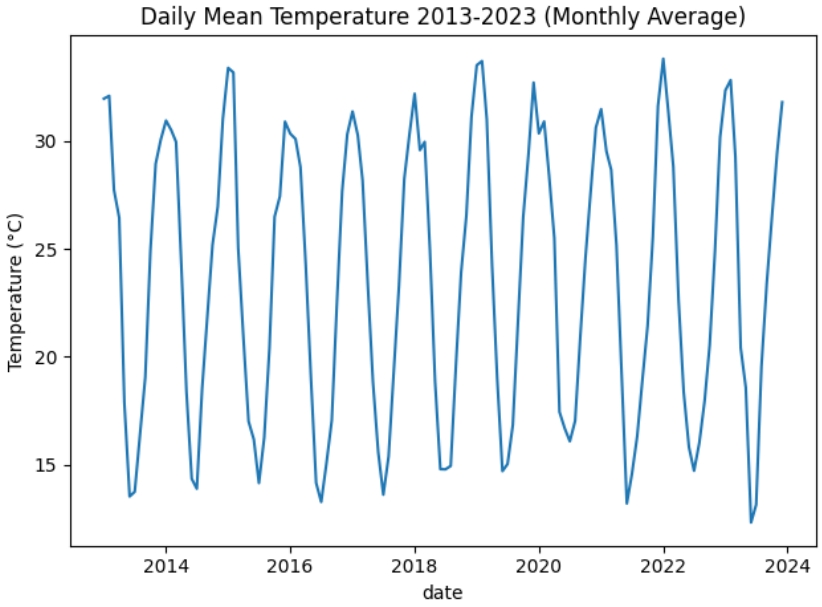

Figure 7: Monthly mean temperature at Meekatharra Airport, WA from 2013-2023.

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

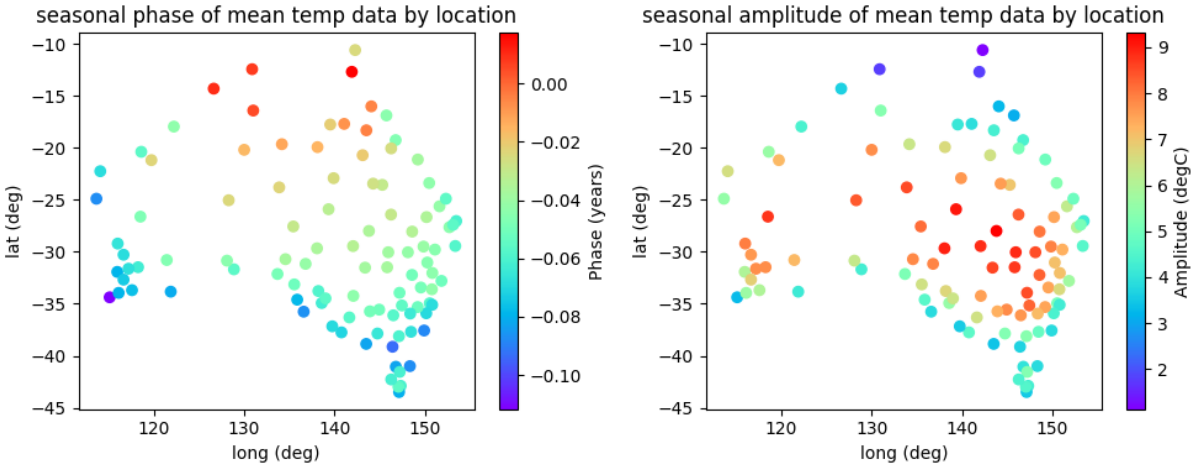

Figure 8: Seasonal component (first harmonic only) of temperature across Australia.

geography textbooks, explained respectively by solar insolation and the maritime effect [2].

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.

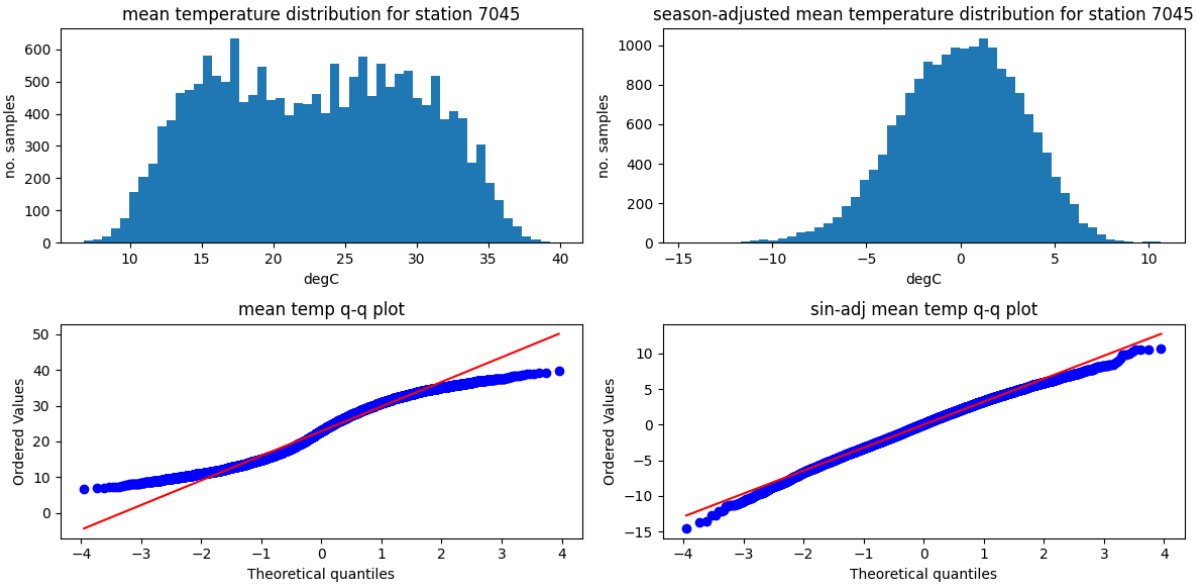

Figure 9: Temperature distribution at Meekatharra Airport before (left) and after (right) seasonal adjustment.

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-

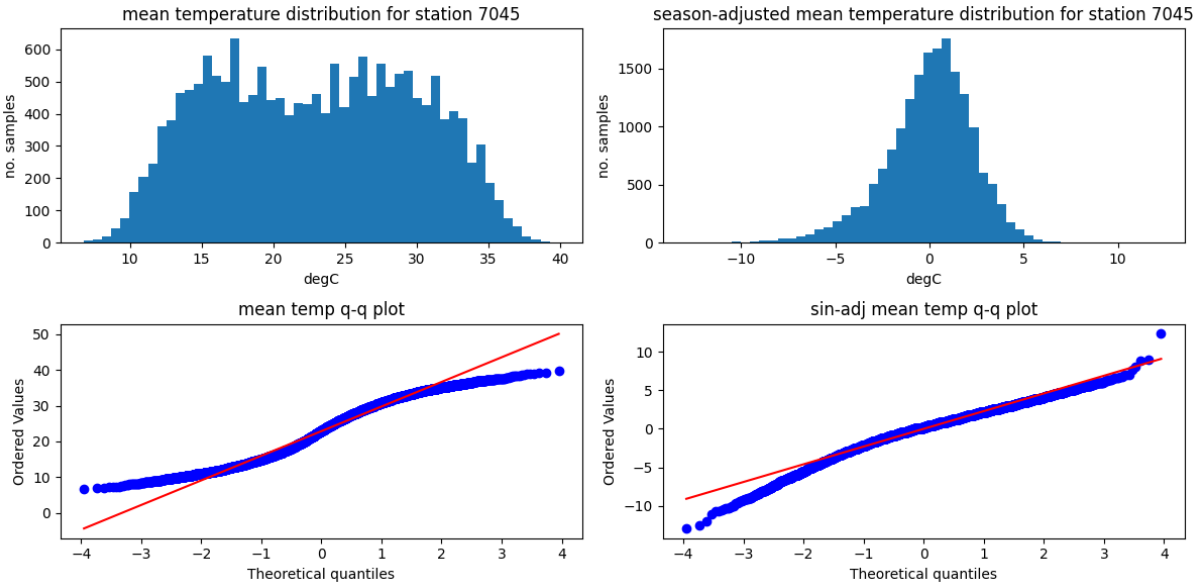

Figure 10: Temperature distribution at Meekatharra Airport before (left) and after (right) differencing.

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-

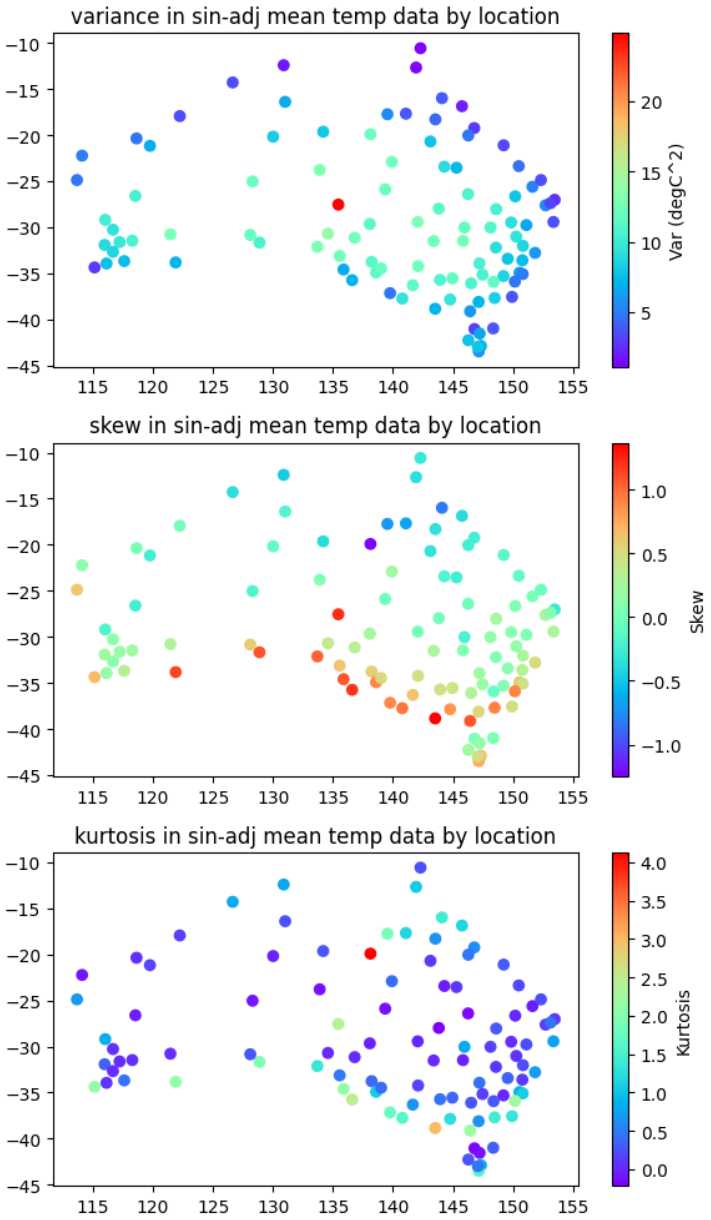

Figure 11: Summary statistics for seasonality adjusted temperature distributions.

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 $\mathbb{R}^n$, $X_{\underset{\sim}{x}} \forall \underset{\sim}{x} \in S$. 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

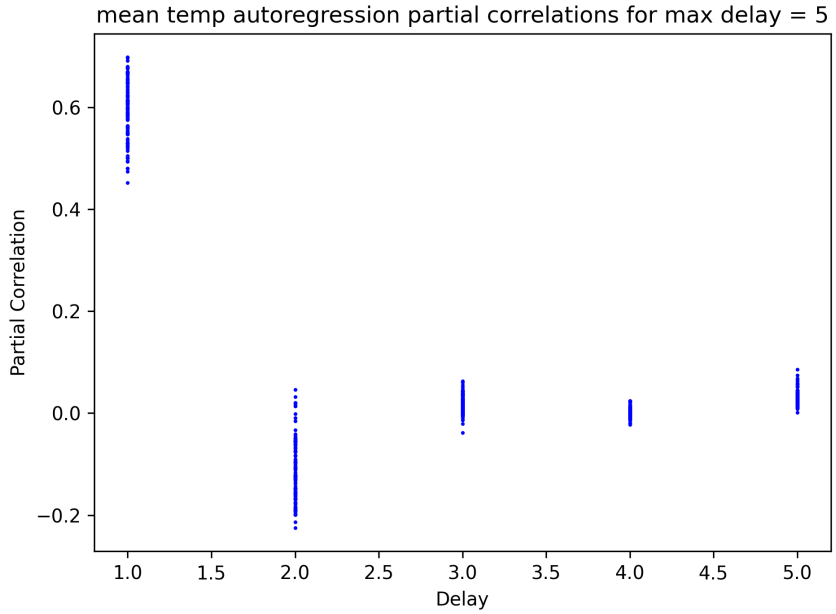

Figure 12: Correlation coefficients for a 5th order autoregression.

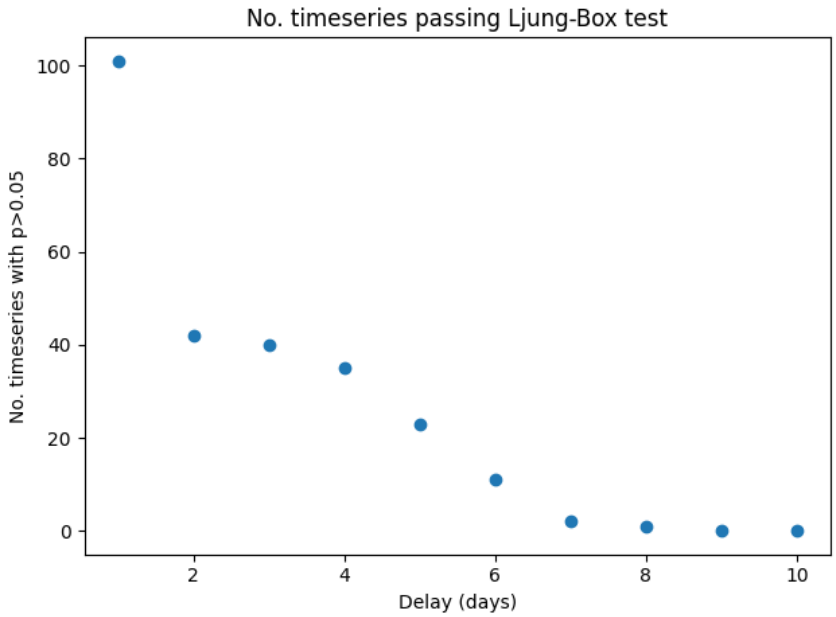

Figure 13: Number of timeseries passing the Ljung-Box test at different delays 1-10.

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.

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

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 $\alpha(h) = 1 - \alpha_1 h - \alpha_2 h^2 + O(h^3)$

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

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.

$$-\frac{\gamma h^2 + O(h^3)}{h^2} n(\underset{\sim}{x}) = \frac{1}{S_n h^{n+1}} \int_{\partial B_{n,h}} X_{\underset{\sim}{x}+\underset{\sim}{dx}} - X_{\underset{\sim}{x}} \underset{\sim}{dx} - (\alpha_2 + O(h)) X_{\underset{\sim}{x}}$$
$$h \to 0$$
$$-\gamma n(\underset{\sim}{x}) = \frac{1}{2n} \nabla^2 X_{\underset{\sim}{x}} - \alpha_2 X_{\underset{\sim}{x}}$$

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 [38]. 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.

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

The solution in 2 dimensions lines up with what Whittle (1954) found, calling it the "elementary correlation in two dimensions" [37].

This model assumes constant variance, which is clearly not present in the Australian tempera-

ture 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

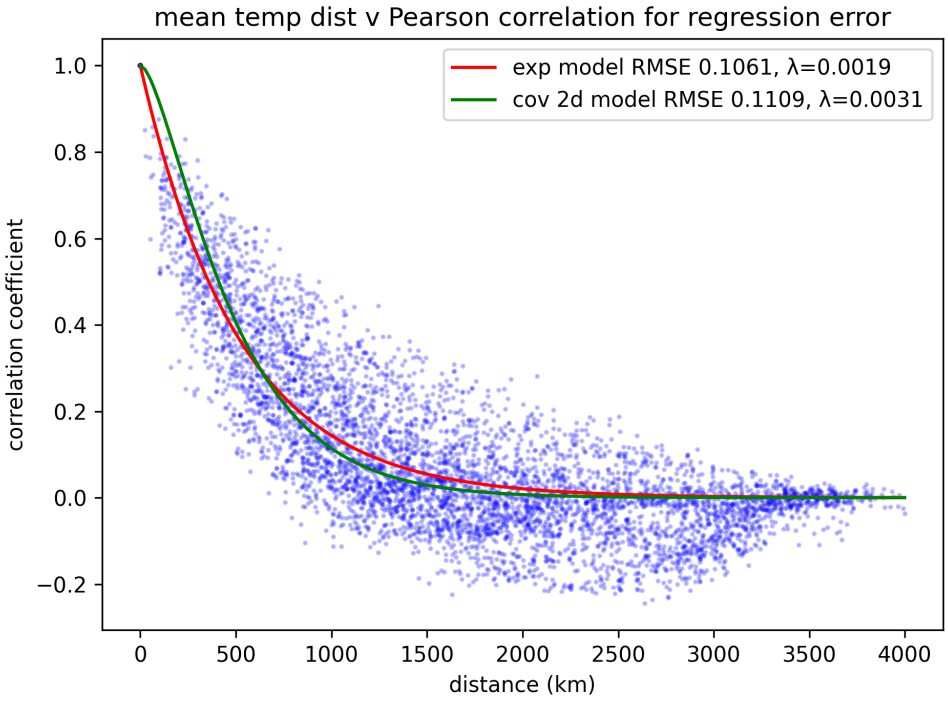

Figure 14: Fitting seasonality adjusted correlations against an exponential model and a Matérn kernel.

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.

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

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.

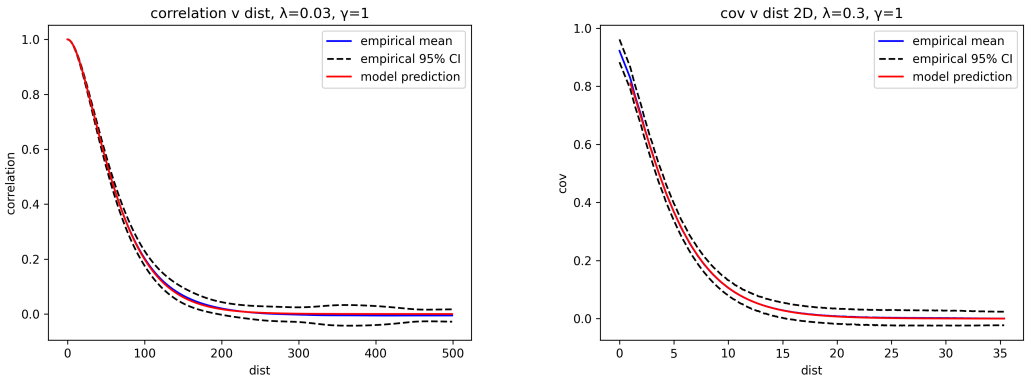

Figure 15: Numerical estimates for 1 (left) and 2 (right) dimensional covariance functions under the screened Poisson equation with unit variance Gaussian noise.

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.

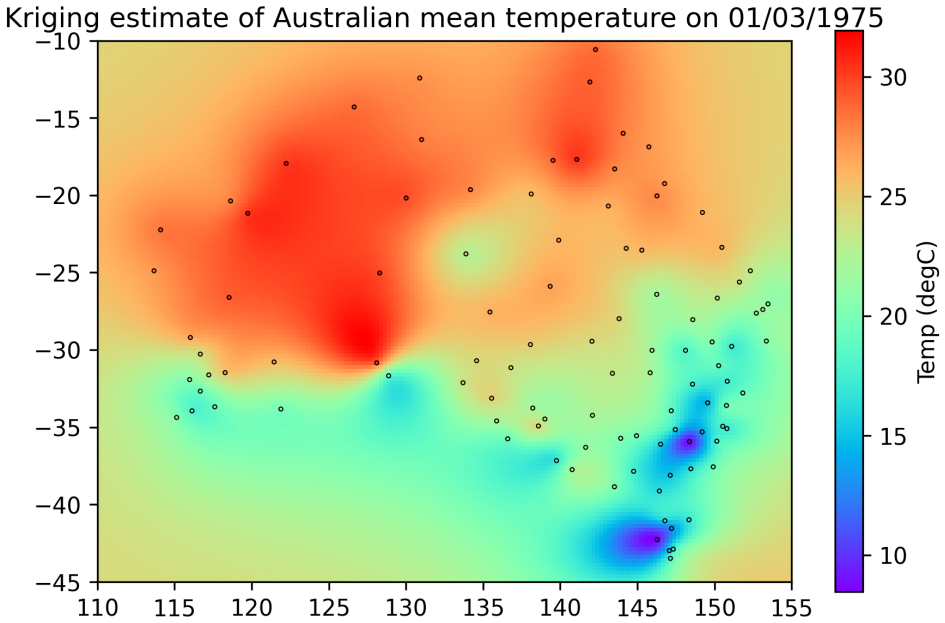

Figure 16: Kriging Australian temperature on 01/03/1975 with the 3D Matërn kernel.

The Kriging error discussed in 3.4.2 operates on variances rather than correlations, so we multiply $\mathbf{\Sigma}_{ij}$ by $\sqrt{\text{Var}(X_i)\text{Var}(X_j)}$, and $\text{Cov}(X_i, X_0)$ by $\sqrt{\text{Var}(X_i)\text{Var}(X_0)}$. For simplicity, we model $\text{Var}(X_0)$ 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.

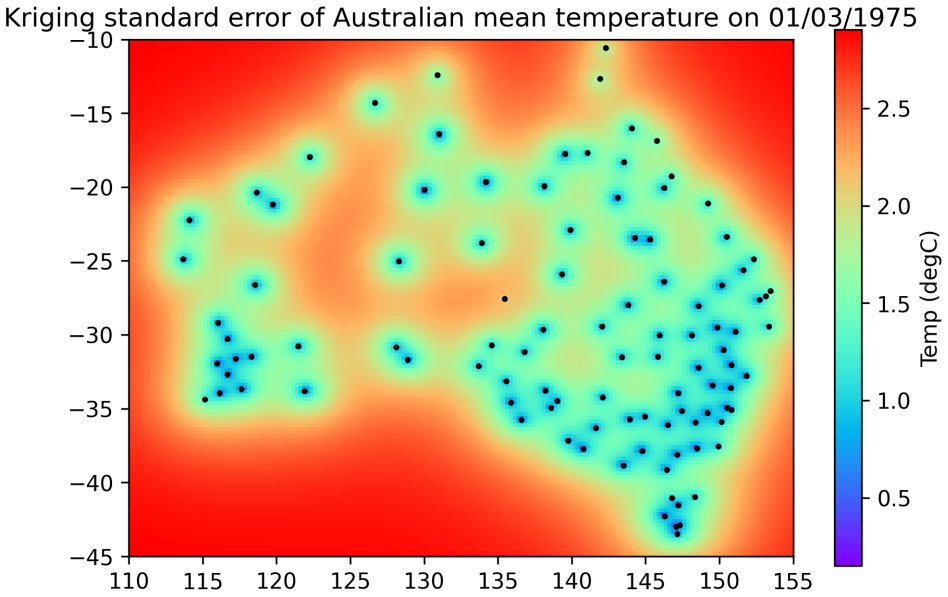

Figure 17: Kriging Standard Error for Australian temperature on 01/03/1975.

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

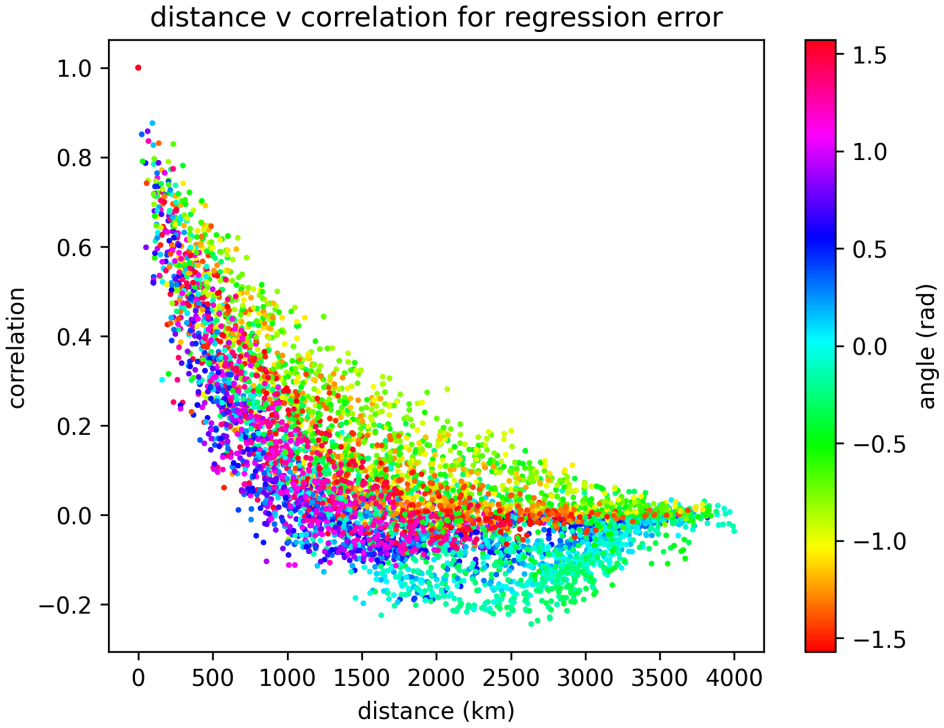

Figure 18: Correlation vs distance and direction. $-\pi/2$ is south, 0 is east, $\pi/2$ is north. Since north and south give the same direction, the colourbar is cyclic.

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 Extending the Matérn Kernel to deal with Anisotropies and Negative Correlations

To make the matern kernel able to deal with anisotropies, we can take replace our diffusion constant with a diffusion matrix, as in the correlation function below for the displacement vector $\underset{\sim}{dx}$.

$$c(\underset{\sim}{dx}) = \exp(-||\boldsymbol{A}\underset{\sim}{dx}||_2)$$

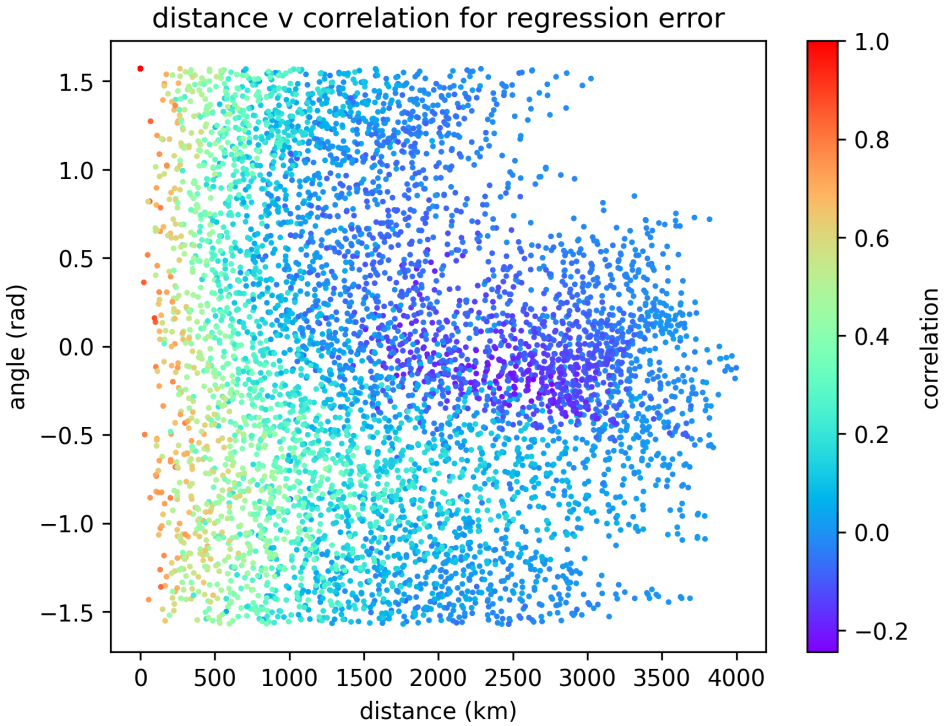

Figure 19: Correlation vs distance and direction. $-\pi/2$ is south, 0 is east, $\pi/2$ is north.

Using pyTorch Autograd to optimise the least squares error with gradient descent, this model reduces the correlation fit RMSE by 10% from 0.1061 to 0.0892. Because $||\boldsymbol{A}\underset{\sim}{dx}||_2 = \sqrt{\underset{\sim}{dx}^\top \boldsymbol{A}^\top \boldsymbol{A}\underset{\sim}{dx}}$, and there are many such $\boldsymbol{A}$ matrices yielding the relevant symettric $\boldsymbol{A}^\top\boldsymbol{A}$ matrix, we just present that matrix below.

$$\boldsymbol{A}^\top\boldsymbol{A} = 10^{-7}\begin{pmatrix} 0.35 & 0.024 \\ 0.024 & 0.35 \end{pmatrix}$$

We can identify how this changes the distance measurement by eigendecomposing the matrix as below. Note we look at $\sqrt{\lambda}$ rather than λ as this measure is in the same units as the diffusion constant from the simpler model, km^{-1}.

$$\sqrt{\lambda} = \begin{pmatrix} 0.0011 & 0.0024 \end{pmatrix}$$
$$\boldsymbol{V} \approx 0.71\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$$

This tells us that the correlation drops off slowest in the north-west/south-east running direction, by the first column of $\boldsymbol{V}$, and fastest in the north-east/south-west running direction, by the second column of $\boldsymbol{V}$.

As of the writing of this thesis, I have not been able to connect this strong implication about the anisotropy fit with the geography literature.

Next, we can deal with negative correlations. A common equation, suspiciously similar to our screened Poisson equation, is the white noise-driven Helmholtz equation with correlation function below.

$$-\gamma n(\underset{\sim}{x}) = (\nabla^2 + \lambda^2) T_{\underset{\sim}{x}}$$
$$c(\underset{\sim}{dx}) = \exp(-j\lambda ||\underset{\sim}{dx}||_2)$$

This equation leads to oscillatory behaviour. If we allow the single system to have multiple diffusion coefficients in such a way so that the correlation function is a sum of solutions, we can get correlations of the form below.

$$c(\underset{\sim}{dx}) = a\cos(\lambda ||\underset{\sim}{dx}||_2 + \phi)$$

In order for this function to have a nicely behaved curvature around $\underset{\sim}{dx} \approx \underset{\sim}{0}$, we need $\phi = 0$, which enforces $a = 1$, because correlation must be 1 when $\underset{\sim}{dx} = \underset{\sim}{0}$.

The physical intuition of the screened poisson equation with a real diffusion constant is fairly simple, it just describes a continuous autoregressive stationary spatial process. The helmholtz equation arises rather when waves become involved, describing oscillatory effects through the space. When temperature effects take time to distribute, they can cause oscillatory effects.

This oscillating correlation function on its own is dubious, because it does not decay in space. This is easily fixed by allowing the original λ in the screened Poisson equation to be complex, $\lambda = a + bj$. We then get the generic stationary correlation model below.

$$c(\underset{\sim}{dx}) = \sum_{k=1}^{N} \exp(-a||\underset{\sim}{dx}||_2)\cos(b||\underset{\sim}{dx}||_2)$$

This model could admit a phase in the cosine function and a constant out the front and still achieve a correlation of 1 at $||\underset{\sim}{dx}||_2 = 0$, but it does not help the fit, so we ignore this case for simplicity. We can combine this model with the anisotropic correction below.

$$c(\underset{\sim}{dx}) = \sum_{k=1}^{N} w_k \exp(-||\boldsymbol{A}_k \underset{\sim}{dx}||_2)\cos(||\boldsymbol{B}_k \underset{\sim}{dx}||_2)$$
$$\sum_k w_k = 1$$

Again using PyTorch Autograd, we can minimise this equation. For $N = 1$, we get a small improvement from 0.0892 RMSE to 0.0824, and with $N = 2$ we get a large improvement down to 0.0746. $N = 3$ does not improve the model much, achieving an RMSE of 0.0739. The

parameters (and eigendecompositions) for the well-performing $N = 2$ model are given below.

$$\begin{aligned}
\boldsymbol{A}_1^\top \boldsymbol{A}_1 &= 10^{-8} \begin{pmatrix} 0.943 & 0.269 \\ 0.269 & 0.291 \end{pmatrix} \\
\sqrt{\lambda_{A1}} &= \begin{pmatrix} 0.00044 & 0.00102 \end{pmatrix} \\
\boldsymbol{V}_{A_1} &= \begin{pmatrix} -0.34 & 0.94 \\ 0.94 & 0.34 \end{pmatrix} \\
\boldsymbol{A}_2^\top \boldsymbol{A}_2 &= 10^{-6} \begin{pmatrix} 0.111 & 0.057 \\ 0.057 & 0.169 \end{pmatrix} \\
\sqrt{\lambda_{A2}} &= \begin{pmatrix} 0.0028 & 0.0045 \end{pmatrix} \\
\boldsymbol{V}_{A_2} &= \begin{pmatrix} -0.85 & 0.52 \\ 0.52 & 0.85 \end{pmatrix} \\
\boldsymbol{B}_1^\top \boldsymbol{B}_1 &= 10^{-8} \begin{pmatrix} 0.597 & 0.711 \\ 0.711 & 0.998 \end{pmatrix} \\
\sqrt{\lambda_{B1}} &= \begin{pmatrix} 0.00024 & 0.00124 \end{pmatrix} \\
\boldsymbol{V}_{B_1} &= \begin{pmatrix} -0.797 & 0.604 \\ 0.604 & 0.797 \end{pmatrix} \\
\boldsymbol{B}_2^\top \boldsymbol{B}_2 &\approx \boldsymbol{0} \\
w_1, w_2 &= 0.53, 0.47
\end{aligned}$$

From this fit, we see that the second fit component is non-oscillatory, and decays faster than the other components. The first component is oscillatory, and decays on a slower lengthspan (lower $\sqrt{\lambda}$s) than the second component.

We plot how this fit maps to a correlation-distance-angle plot in 20. This figure clearly displays some of the desired anisotropic features for a fit of 19, including the different diffusion speeds by angle, and wells of negative correlation.

Interestingly, in three dimensions the fundamental distance-covariance function is naturally decomposed by a laplace transform. This is not true in one or two (spatial) dimensions, where the fundamental decomposition is into other functions, given by convolutions of appropriate Matérn kernels. In one-directional one-dimensional environments, a laplace transform again becomes appropriate.

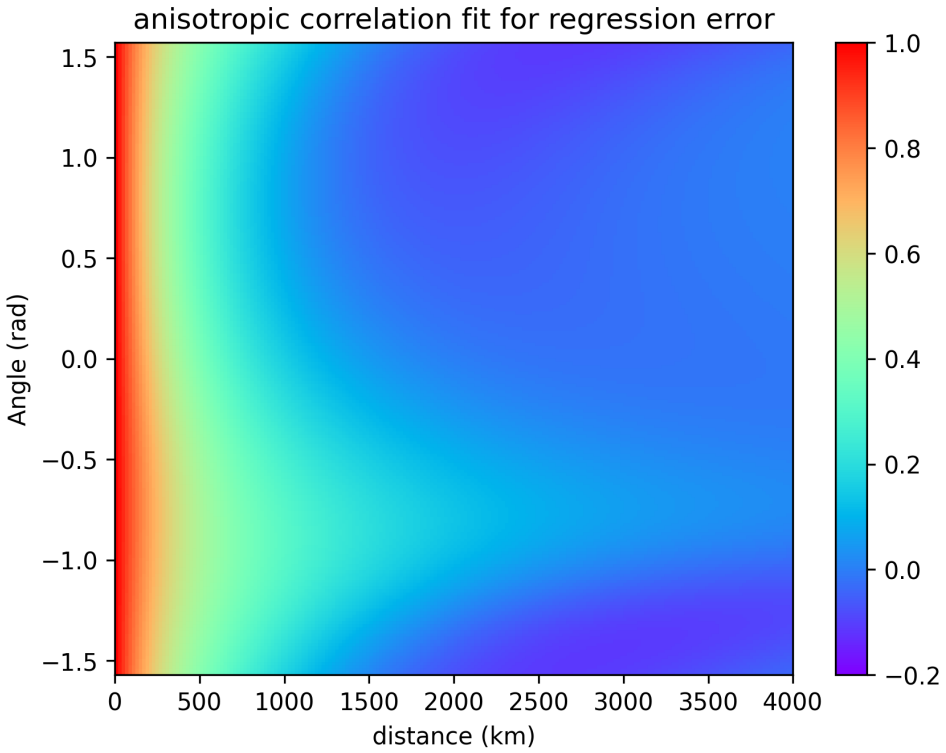

Figure 20: Correlation fit vs distance and direction. $-\pi/2$ is south, 0 is east, $\pi/2$ is north.

4.6 Deterministic Graph Filters

4.6.1 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 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.

4.6.2 Well-Chosen Graph Fourier Transforms

Throughout this subsection we consider a graph with n vertices, and a graph shift 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.

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

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 $\underset{\sim}{\Sigma}_x$ with k/n the compression ratio. This prompts us to consider the case where the graph operator $\boldsymbol{A} = \underset{\sim}{\Sigma}_x$.

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.

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

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

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

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

In the case that noise is uncorrelated Gaussian white noise, $\underset{\sim}{\Sigma}_\varepsilon = \boldsymbol{I}\sigma^2$ (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 $\mathbf{\Sigma}_{\underset{\sim}{x}}$ 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 [32][16]. 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.

4.6.3 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.

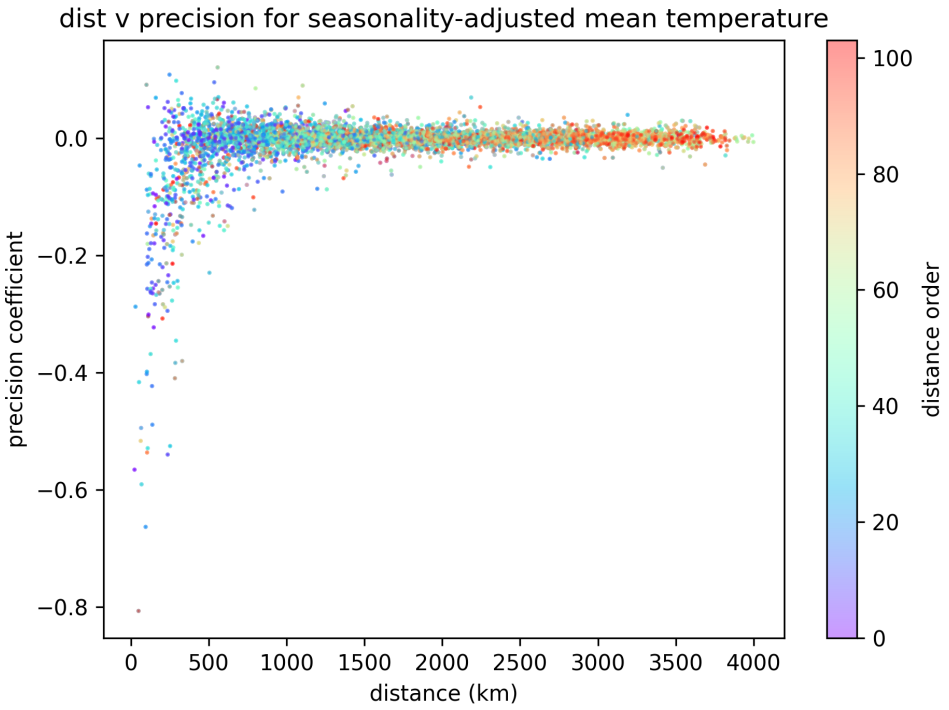

Figure 21: Precision values vs absolute distance and distance order.

There is no perfect structure in the precision values for either distance or distance order. There 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 (non-anisotropic) 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 4.6.2, the least-squares filter under the precision

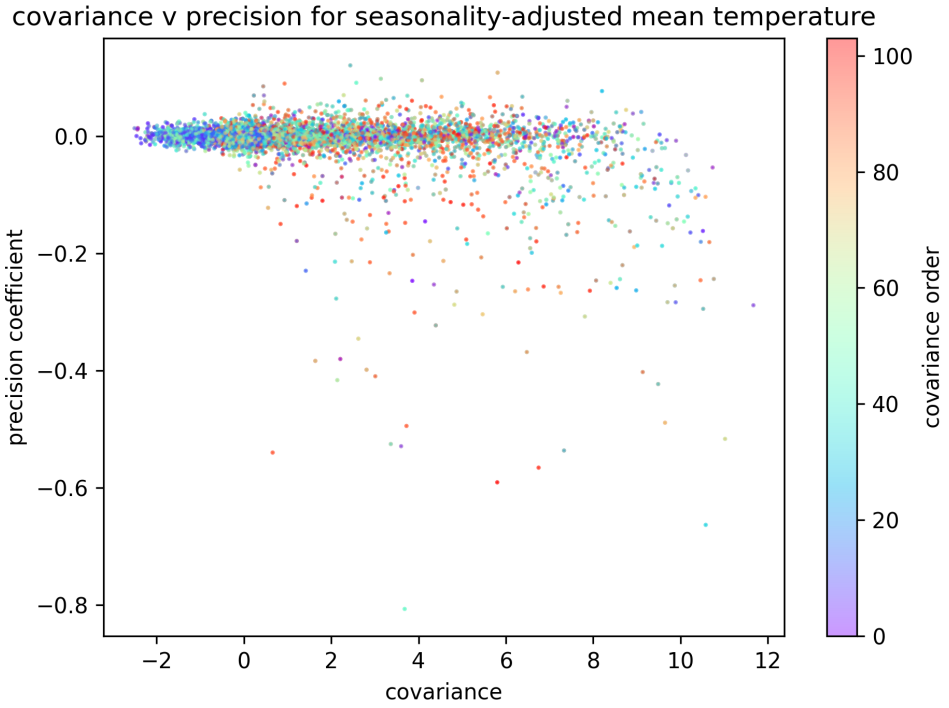

Figure 22: Precision values vs covariance and covariance order.

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.

matrix is as below.

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

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[22]. It is given as follows.

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

k : number of parameters

n : number of observations

$\hat{\mathscr{L}}$: 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 : prediction i

o_i : observation i

σ^2 : 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.

$$\boldsymbol{A} = (A_1|A_2|...|A_p)$$

$$\underset{\sim}{z_t} = \begin{pmatrix} \underset{\sim}{x_{t-1}} \\ \underset{\sim}{x_{t-2}} \\ \vdots \\ \underset{\sim}{x_{t-p}} \end{pmatrix}$$

$$\underset{\sim}{x_t} = \boldsymbol{A}\underset{\sim}{z_t} + \underset{\sim}{\varepsilon_t}$$

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 ($2.09^\circ C$ and $2.06^\circ C$ 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 Fig. 26.

p	RMSE ($^\circ C$)	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[30]. 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 ($^\circ C$)	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.

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

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 $\underset{\sim}{\boldsymbol{Z}}$ 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 ($^\circ C$)	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 ($^\circ C$)	BICa ($\times 10^6$)	no. params
0.005	1.66	1.946	3904

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 Fig. 30.

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

Figure 30: l_0 group 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 ($^\circ C$)	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 ($^\circ C$)	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.

5.5 Issues with BIC and overuse of l0

BIC is essentially a variant on l0 regularisation. It evaluates a model based upon both its likelihood and how many parameters it uses. A difficulty arises when we consider what consitutes a parameter. We additionally should recall Goodhart's law, a cautionary saying within machine learning: "When a measure becomes a target, it ceases to be a good measure".

BIC typically is only used to quantify the impact of the real parameters of a system. We can imagine a pathological example of l_0 regression where we regress an output signal against many signals equal to the output plus a little noise. Typical regression assigns a significant regression coefficient to each input. Ridge (l_2) regression acts to homogenise the regression coefficients, which can be more robust because the different sources of noise may cancel out more than otherwise, i.e. it reduces our confidence in large regression coefficients. l_0 regression may choose just one input to rely upon, as it is seen as a "simpler" model than taking for example the average of all inputs, but exposes the model to the noise of just one input, which may worsen signal-to-noise ratio.

The issue here is that "number of parameters" is a fickle thing. l_0 purports to restrict the number of parameters by selecting between an exponential number of models, not accounting for the parameters in choice of model (i.e. the space of choices of which matrix entries are 0). This contributes significantly to the effective number of parameters in the overall l_0 model, rather than just the number of parameters in the particular lucky choice of which inputs should be chosen for each output.

In the literature for example, when discussing ridge regression, it does not set any entries to 0, yet is often considered to reduce the effective number of parameters used. Tibshirani, Hastie, and Friedman [8] define the effective number of parameters by how sensitive the output predic-

tion of a model is to the output of the data, as below.

$$\begin{aligned} p &= \sum_{i=1}^{N} \frac{\partial \hat{y}_i}{\partial y_i} \\ &= \sum_{i=1}^{N} \frac{\partial}{\partial y_i} \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top y_i \\ &= \mathrm{tr}(\boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top) \\ &= \mathrm{tr}((\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{X}) \\ &= \mathrm{tr}(\boldsymbol{I}_N) \\ &= N \end{aligned}$$

This is exactly the number of effective parameters we expect. $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top$ is called the smoothing matrix, that takes the sample output to the predicted output. For ridge regression, the smoothing matrix is as below.

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top$$

Which shrinks the total "effective dimension" of the fit, by making the output fit to the sample data, $\hat{y}$, less sensitive to the input data y.

The choice of regularisation we use informs our choice of prior. The l_0 prior biases towards two signals having either no predictive power on each other at all, or as much as measured in the sample. In a setting like spatial temperature modelling, this fit may be innappropriate - our prior may be that we expect two temperature signals in the same space to have some predictive power over each other, no matter how far apart they are, simply by nature of being in the same environment.

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 $\boldsymbol{\Sigma}_\varepsilon$. We also provide a computationally efficient computation for all pairwise $\rho_{ij|\neq ij(\omega)}$ values in terms of the partial coherence matrix.

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

Where the square root and division are performed elementwise for $\rho_{ij|\neq ij}$. Note that the per-omega evaluation of $F^{-1}(\omega)$ does not require a recomputation of any matrix inversions, only the computation of $A(j\omega)$ along with 2 matrix multiplications.

6.2 Estimating the Error Precision

As we have seen in the previous section, the CMI calculation relies heavily upon finding a fit of $\boldsymbol{A}_\tau$, as well as $\boldsymbol{\Omega}_\varepsilon$. We have discussed many choices of $\boldsymbol{A}_\tau$ in 5.3. We now turn our attention to choosing $\boldsymbol{\Omega}_\varepsilon$. In 3.6 we discussed a fast algorithm for finding a BIC optimal estimation of $\boldsymbol{\Omega}_\varepsilon$. This estimation is shown in Fig. 33 compared with the inverse of the sample covariance of the error under a basic $VAR(2)$ model fit.

The l_0 estimate of $\boldsymbol{\Omega}$ retains only 60.34% of the matrix entries that are in the sample precision matrix, providing a simpler model of the error structure. The max and min on the colourbars are tighter than the whole range of values in the plot to accentuate how the l_0 fit is more robust to noise than the regular fit.

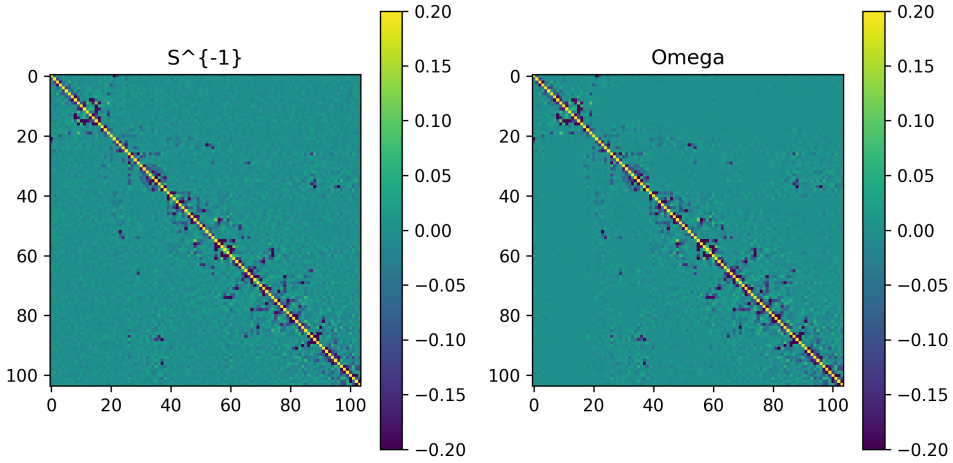

Figure 33: The sample precision (left) against the BIC-optimal l_0 estimate precision (right).

6.3 CMI Evaluation Variants

We now have two selections to make before computing the CMI. First, how to compute the VAR regression matrices. The two major choices we have here are the simple choice of direct VAR regression, and the more computationally expensive choice of the BIC-optimal l_0 regression. Second, how to compute the error precision matrix. Again, we can either take the sample precision, or the BIC-optimal l_0 estimate of the error precision. This gives us four choices of how to calculate the CMI. Another key point is that the integral that gives the CMI does not have a closed form solution, so we must integrate numerically. Fortunately, the computations are fast, so even a very fine numerical integration is fairly fast. All the information of Fourier transform of a length n digital signal is contained within n evenly spaced samples of its frequency spectrum. Hence, in order to retain all the fidelity of an exact computation of the CMI, our numerical integration needs to have n samples. We can observe some important geographic patterns in these estimations of the CMI in Fig. 34. There are a number of interesting observations to note here. The first and clearest is that strong connections between timeseries tend aswell to be short in geographical distance. Secondarily, that the bottom two plots which utilise an l_0 $VAR(2)$ regression rather than a plain $VAR(2)$ regression assign a much more significant CMI to far-reaching coastal connections across the great Australian bite, as well as between the two northern tips of the country. This seems somewhat contrary to a simple expectation - that temperature relationships should be mostly connections between nearby stations, as these are most related in their distributions, and additionally, and that l_0 regressions should reduce spurious connections that arise from noise. In response, we can interpret this phenomenon either as suggesting that l_0 regression, especially l_0 VAR regression, as in the bottom two plots, is innappropriate here for some reason, as it attaches high CMI to spurious connections, or that these long-range connections are an integral part of this model of temperature evolution over time.

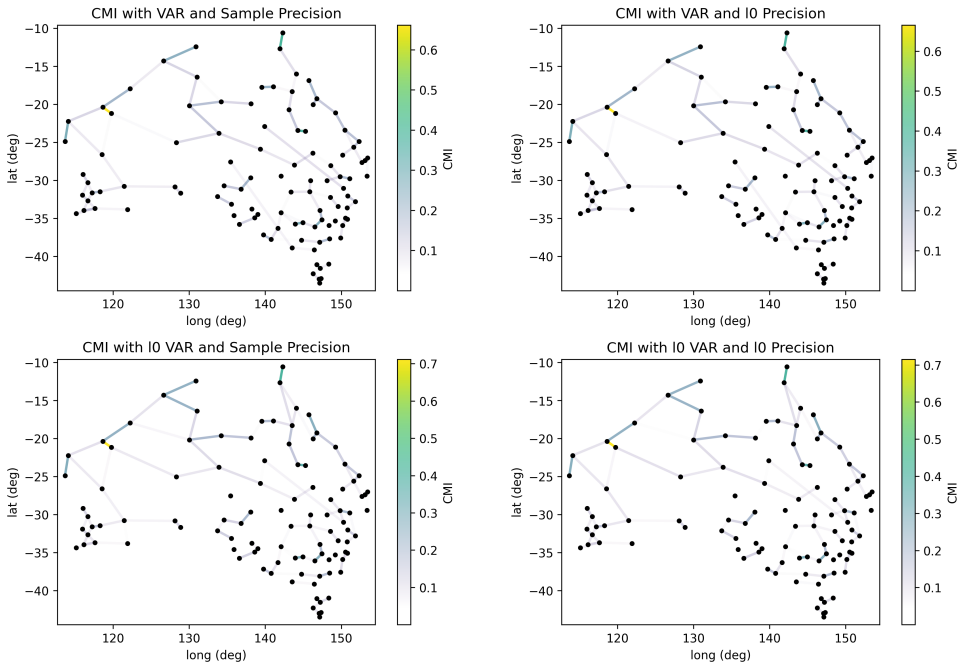

Figure 34: The various estimations of CMI plotted as edge relationships between stations.

A reasonable interpretation of the effect of these connections is that although the stations may not physically be effecting each other on a time scale of a 2-day VAR connection, the super-systems that influence both of them over time may be a significant contributing factor to the information between each side of the country. These long connections do not only run across the coast, but also runs from the left to the right of the country across the inland as well. This certainly seems to suggest some interesting long-range connection between either side of the country. It is important to note that for the most part too it is a west-east running effect, rather than a north-south running effect.

6.4 Frequency-domain Interpretation of CMI

The CMI formula aggregates a per-frequency component across the range of digital frequencies. It is of importance to be able to intuitively understand what these components represent. In particular, we can consider the simple case of the mutual information of two sinusoids of the same frequency, $x_1(t)$ and $x_2(t)$, with multivariate normally distributed amplitude and phase. That is to say, there may be some predictive power in knowing the amplitude or phase of one of

the signals. In this non-conditional case, $\rho_{12}(\omega)$ can be computed as below.

$$
\begin{aligned}
x_1(t) &= A_1\cos(\omega_0 t+\phi_1)\\
x_2(t) &= A_2\cos(\omega_0 t+\phi_2)\\
\rho_{12}(\omega_0) &= \frac{S_{12}(\omega)}{\sqrt{S_{11}(\omega)S_{22}(\omega)}}\\
&\text{for simplicity, considering only } \omega>0\\
S_{11}(\omega) &= \mathbb{E}(\frac{A_1^2}{2}\delta(\omega-\omega_0))\\
&= \frac{\mu_{A1}^2+\sigma_{A1}^2}{2}\delta(\omega-\omega_0)\\
S_{22}(\omega) &= \frac{\mu_{A2}^2+\sigma_{A2}^2}{2}\delta(\omega-\omega_0)\\
S_{12}(\omega) &= \mathbb{E}(X_1(\omega)X_2^*(\omega))\\
&= \mathbb{E}(\frac{A_1A_2}{2}e^{-j(\phi_1-\phi_2)}\delta(\omega-\omega_0))\\
&= \frac{\mu_{A1}\mu_{A2}+\sigma_{A1,A2}}{2}e^{-j(\phi_1-\phi_2)}\delta(\omega-\omega_0)\\
|\rho_{12}(\omega_0)|^2 &= \frac{\mu_{A1}^2\mu_{A2}^2+\sigma_{A1,A2}^2+2\mu_{A1}\mu_{A2}\sigma_{A1,A2}}{(\mu_{A1}^2+\sigma_{A1}^2)(\mu_{A2}^2+\sigma_{A2}^2)}
\end{aligned}
$$

In the case that we allow for negative amplitudes, and give each sinusoid a mean amplitude of 0, we see an interesting result.

$$
\begin{aligned}
|\rho_{12}(\omega_0)|^2 &= (\frac{\sigma_{A1,A2}}{\sigma_{A1}\sigma_{A2}})^2\\
&= \rho_{A1,A2}^2
\end{aligned}
$$

We can interpret $|\rho_{12}(\omega)|^2$ then as measuring, in the case of 0 mean amplitude, the squared correlation between the amplitudes of the signal at ω. This of course cannot be measured for a single signal sample, but can only be measured if many samples of the signal can be taken. Additionally, for two random variables, the wrapping function we see in the CMI formula, as shown below, represents the mutual information between the variables whose correlation we are plugging in.

$$CMI_{ij}(\omega) = -\frac{1}{2}\ln(1-|\rho_{ij}(\omega)|^2)$$

That is, each frequency component tells us how much the information we gain about the amplitude of one of the signals when we learn the amplitude of the other. The conditional component tells us that we account for all other information sources before determining this. A major

limitation of this measure is that while it encodes information between the amplitudes at each frequency component, it does not convey the information shared between phases at each frequency component. A good source of future work would be to develop and investigate the validity of such a measure, but this task is left to future research.

At first this lack of phase information seems like a massive drawback, but there is something to be salvaged here. In particular, an extreme example of where this predictivity is still useful is where two signals have significant CMI at all frequencies, but a sample of one of the related signals exhibits a particularly large contribution at a particular frequency. This informs us that we expect the other signal to also have large contribution at that frequency. That is, rather then telling us how much trends are exactly predicted between the two signals, the periodicity of effects in one signal can inform us of the periodicity of effects in another signal, but not whether the effect will be leading or lagging.

We can then look back to the CMI formula before to extract and interpret its per-frequency contribution, as has been little applied and reflected on in the literature. We recall the CMI formula and extract its per-frequency contribution below.

$$\begin{aligned} CMI_{ij} &= -\frac{1}{2}\int_{-\pi}^{\pi} \ln(1-|\rho_{ij|\neq ij}(\omega)|^2)\frac{d\omega}{2\pi} \\ CMI_{ij}(\omega) &= -\frac{1}{4\pi}\ln(1-|\rho_{ij|\neq ij}(\omega)|^2) \\ &\quad -\frac{1}{4\pi}\ln(1-|\rho_{ij|\neq ij}(-\omega)|^2) \end{aligned}$$

If we can identify an interest in a particular frequency band of the CMI, it is also useful to define the bandpass cmi (BPCMI), defined below.

$$\begin{aligned} BPCMI_{ij}(\omega_0,\omega_1) &= -\frac{1}{2}\int_{\omega_0}^{\omega_1} \ln(1-|\rho_{ij|\neq ij}(\omega)|^2)\frac{d\omega}{2\pi} \\ &\quad -\frac{1}{2}\int_{-\omega_1}^{-\omega_0} \ln(1-|\rho_{ij|\neq ij}(\omega)|^2)\frac{d\omega}{2\pi} \end{aligned}$$

As an intuitive description for example, a low pass filter measure of CMI aggregates the extent to which low frequency, that is to say, longer term trends in two signals are conditionally predictive of each other. Higher frequency components describe the conditional predictive power between short term trends in two signals. This has the potential to be a powerful effect. The bandpass CMIs for the four equally sized bands from 0 to π are shown in Fig. 35.

This figure reveals a few significant features. First, that a lot of the content between the bands is very similar. At least in this example it is the case that if we know that the amplitude of the high-frequency components of two signals covary, we can predict (with some uncertainty), that their low frequency components will covary as well. Some interesting edges vary in strength

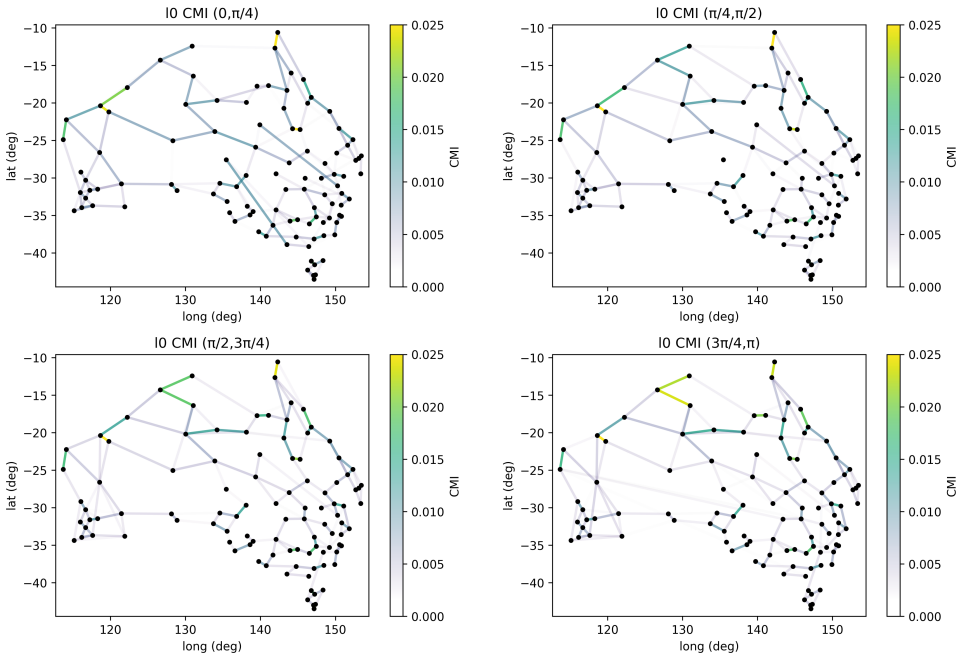

Figure 35: Australian Bandpass CMI Maps

significantly between the frequency bands. It seems at first that the lowpass CMI is very similar to that in the highpass CMI. We are then interested in plotting them against each other, as is done in Fig. 36.

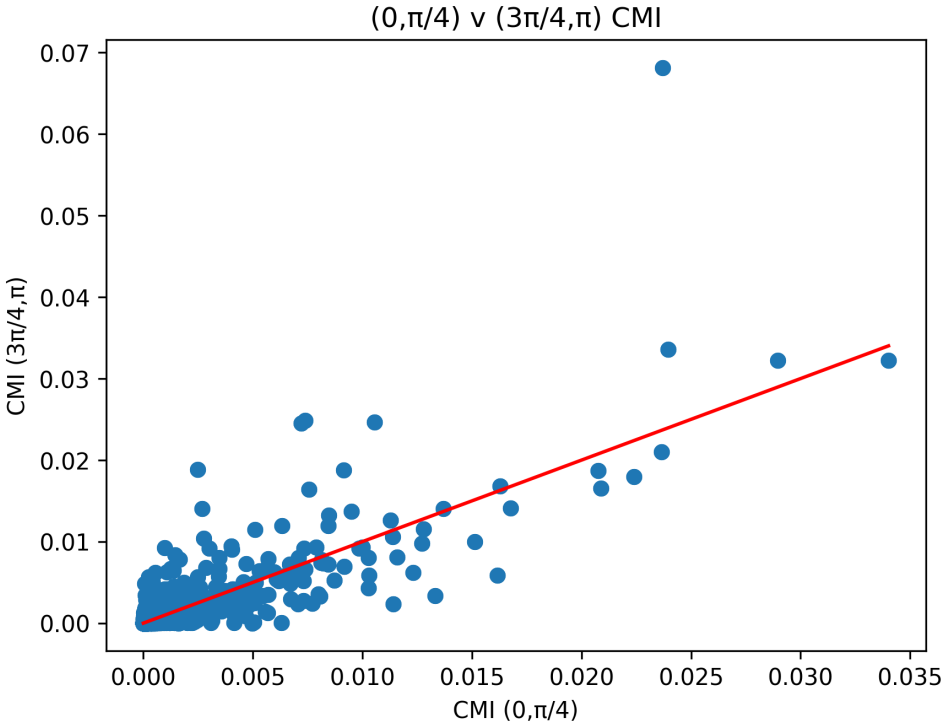

Figure 36: Australian lowpass vs highpass CMI comparison

Based on the figure, the relationship between low and high pass CMI is rougher than the map

initially shows. Although the CMI in different bands has similar structure, that is significant components tend to still be significant, the CMI values themselves vary greatly.

A significant foreseeable advantage of frequency band interpretation of CMI is in the fact that when computing total CMI, we make the total summation across all frequencies, where indeed we may be more concerned about our capacity for prediction only in the frequency band containing the most power in the signal. Part of the desire for this correction is the fact that the $\rho_{ij}(\omega)$ term explicitly adjusts for signal power at the considered frequency, normalising out the power of the signals at each frequency. One can imagine how with two random sinusoids, the CMI of interest would in fact be at a single frequency, and any other frequency contributions, though equally weighted, may constitute mostly noise.

We wish to see if we can make any mathematical interpretation about whether the bandpass distance threshholding relationship in CMI has any clear explanation. For simplicity, lets consider the $F^{-1}(\omega)$ matrix for a $VAR(1)$ autoregression.

$$\begin{aligned} F^{-1}(\omega) &= (I - e^{-j\omega}\boldsymbol{A})^H \boldsymbol{\Sigma}_\varepsilon^{-1} (I - e^{-j\omega}\boldsymbol{A}) \\ &= \boldsymbol{\Sigma}_\varepsilon^{-1} + \boldsymbol{A}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A} - e^{j\omega} \boldsymbol{A}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} - e^{-j\omega} \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A} \\ &= \boldsymbol{\Sigma}_\varepsilon^{-1} + \boldsymbol{A}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A} - ((e^{-j\omega} \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A})^H + (e^{-j\omega} \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A})) \end{aligned}$$

The similarity across bands is demonstrated by the fact that the most significant term (as $\boldsymbol{A}$ tends to shrink inputs in stable autoregression), is not frequency dependent, $\boldsymbol{\Sigma}_\varepsilon^{-1} + \boldsymbol{A}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A}$. The frequency-dependent terms are constructed from $\boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A}$, for which each entry ij is a precision-weighted description of how input j predicts output i, accounting for indirect relationships where j predicts k through the regression/transition matrix $\boldsymbol{A}$, and then through the smoothing precision matrix $\boldsymbol{\Sigma}_\varepsilon^{-1}$ predicts i. One can reason that for higher order $VAR(p)$ models, a similar thing occurs, where each of the cross terms account from splitting the prediction into two steps of length $k + l \leq 2p$.

If we neglect the small $\boldsymbol{A}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{A}$ factor, for entries $\boldsymbol{\Sigma}_{\varepsilon ij}^{-1} < 0$, i.e. when i and j's errors conditionally covary, at low frequencies, e.g. DC $\omega = 0$, i and j's magnitude predicting each other via $\boldsymbol{A}$ increases the magnitude of the entry $F_{ij}^{-1}(\omega)$. Exactly the opposite occurs at high frequencies, e.g. $\omega = \pi$, where i and j's magnitudes predicting each other through $\boldsymbol{A}$ decreases CMI if i and j's errors covary. What this suggests, is that at low frequencies, predictions across different time scales (immediately through the error covariance, as well as over a lag through the regression matrix) compound in CMI, whereas for high frequencies they negate. This is not quite as simple as the relationship we observe, but it is quite similar - low-frequency CMI compounds influence across lags, whereas high-frequency CMI contrasts influence across lags.

7 Verifying and Disputing Results with NOAA Data

Throughout this work, we have observed a variety of results on Australian mean daily temperature data. Some key results that we can seek to verify using the US data set include:

- 8.54% RMSE for a compression ratio of 1:50 using the space-time graph fourier transform.
- Geographical predictors of seasonality phase and amplitude.
- Predictors of temperature variance and skew.
- Fitting the correlation vs distance function with a 3D Matern kernel with $\lambda = 0.0019 km^{-1}$, and with an anisotropic kernel.
- Observations of the anisotropies in the correlation vs distance function.
- BIC success and superiority of the l_0 coordinate descent regression.
- Stronger long-range CMI connections when l_0 regression is used.

7.1 Compression Results

Compression results on NOAA data vs the BOM data are shown below.

Fraction of Coefficients Used	1/50	1/20	1/15	1/10	1/7	1/5	1/3
RMSE BOM (%)	8.54	6.95	6.47	5.80	5.18	4.54	3.43
RMSE NOAA (%)	13.71	11.14	10.29	9.05	7.91	6.78	4.93

Figure 37: NOAA Mean Daily Temperature Data (1975-2023) Compression Results

The NOAA data contains data with much greater variation that the BOM data, and as such it is more difficult to compress efficiently. Despite this it shows the same behaviour of being effectively compressed by use of a graph fourier transform dependent only on the distances in the system.

7.2 Geographical Predictors

In the BOM dataset, we observed in 4.3.1 that seasonal phase varied in great part with latitude, the areas nearer to the equator lagging those further south. In the NOAA dataset, we see a much larger variation in phase, with extreme results close the the equator, as in Fig. **??**. This

is explained in part by the seasonal amplitude variation; in the NOAA dataset, the seasonal amplitude for stations near the equator is minimal, leaving phase freer to wander arbitrarily. Unlike the Australian dataset, the seasonal amplitude varies less by coastality, but by latitude,

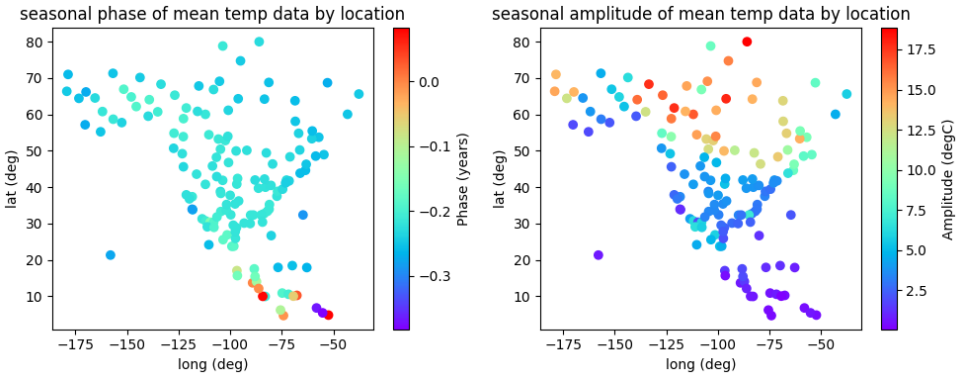

Figure 38: Seasonal phase and amplitude of mean daily temperature in north America.

with Canadian temperatures being much more variable than those in the United States. This is a significant difference, and shows that the relationship of geography to climate is more complicated than simply looking at latitude and coastality.

7.3 Correlation vs Distance Fit and its Anisotropies

The NOAA dataset exhibits some similarity in its correlation vs distance function with that found for the BOM dataset. Fig. 39 shows how the loose 3D Matérn kernel fits the NOAA data with a fairly similar $\lambda = 0.0022km^{-1}$ diffusion constant to that found in the BOM dataset, $\lambda = 0.0019km^{-1}$. It is important to note here that the NOAA data we are looking at covers a much larger geographical area than the BOM dataset, including many more longer-range connections. An interesting idea would be to regress both systems with an up/downweighting of important of some samples based on the distribution of how many relationships fall into a certain distance bucket. For example, if there are many fewer $100km$ connections than $2000km$ connections, though we may care more about accurately fitting the short-range connections, perhaps it would be pertinent to regress with this accounted for. Regardless, the fit here exhibits similar behaviour to the BOM dataset - the 3D kernel fits better than the 2D kernel, and the diffusion constant is very similar.

Similar to the BOM dataset, we exhibit significant noise in the fit, and especially interestingly we observe the same bump in negative correlations for some connections peaking around $2000km$. The envelope of the two functions is very similar in structure, suggesting great similarity between the two systems. As in the BOM dataset, we can add a third coloured dimension through connection angle, as in Fig. 40. To aptly compare the similarities to the BOM dataset, we restrict connections to those below $4000km$.

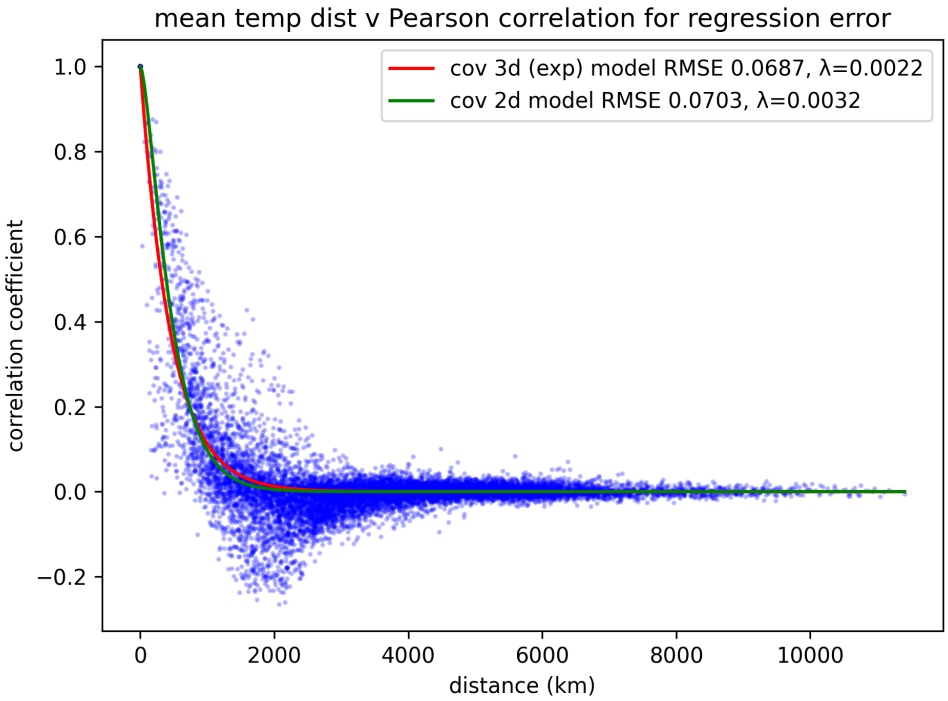

Figure 39: Correlation vs distance for connections in north America.

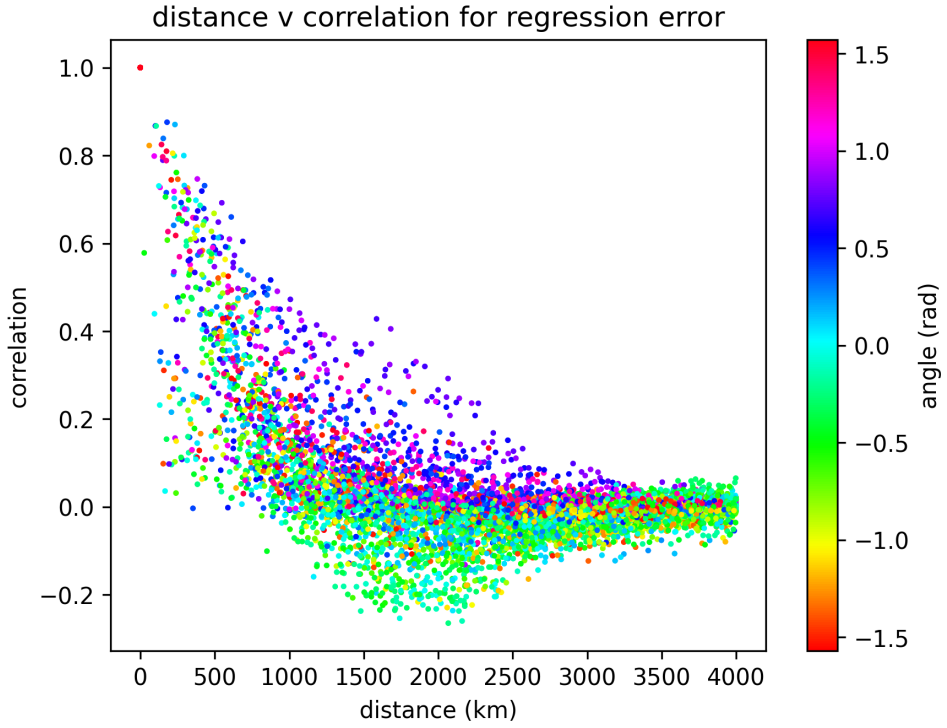

Figure 40: Correlation vs distance and angle for connections in north America. 0 is east, $\pi/2$ is north, $-\pi/2$ is south.

The addition of angle, like in the BOM dataset exhibits intriguing striation in the correlations, with the lower and negative band of correlations running north-west to south-east, the upper band of correlations running north-east to south-west, and the middling correlations running

either north-south or east-west. This pattern is exactly opposed to the relationship seen in the Australian data. A clear hypothesis here is that this swapping is due to the BOM data coming from the southern hemisphere, and the NOAA data from the northern hemisphere. More datasets, or a more advanced model of the correlation function is required here.

The addition of angle here, although it reveals similar patterns to the BOM model, is much noiser, and does not seem as simply explained by distance and angle alone. One theory here is that the NOAA dataset contains a much larger area, and thus contains patterns that change gradually over large areas. This noise is better exhibited in Fig. 41.

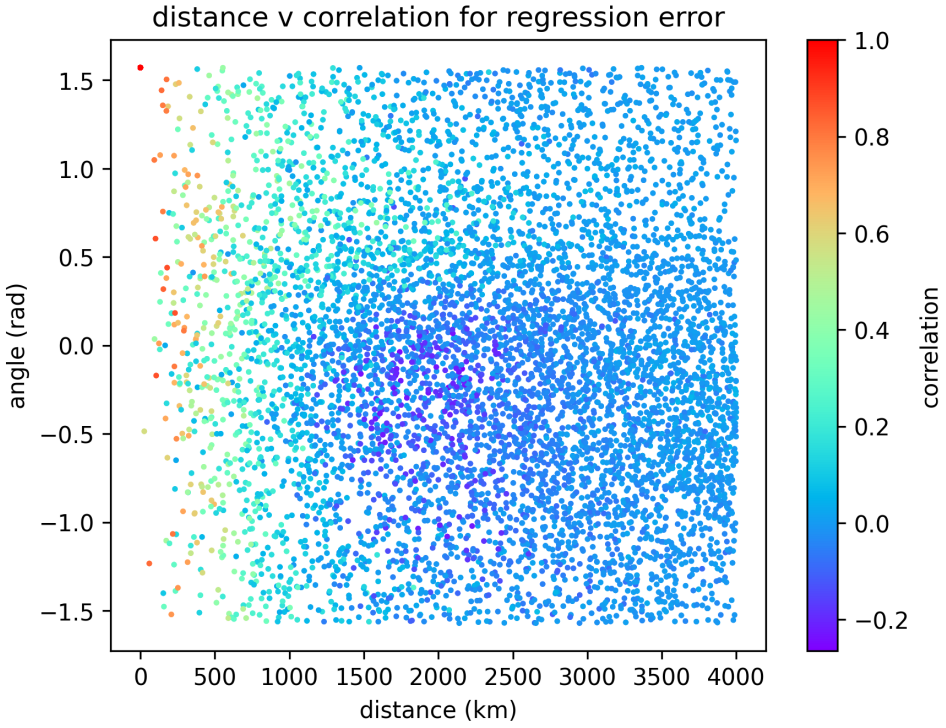

Figure 41: Correlation vs distance and angle for connections in north America. 0 is east, $\pi/2$ is north, $-\pi/2$ is south.

The fit of this correlation function achieved by the algorithm in 4.5 is quite similar in nature to that achieved for the BOM dataset, and performs similarly in terms of RMSE, with an overall RMSE of 0.0598, though this may be dominated by the fact that the dataset has many more negligible long-distance relationship compared with the BOM dataset. As in 41, the anisotropic relationship is reversed such that north-east/south-west running relationships tend to be higher correlated than those running north-west/south-east. This suggests this anisotropy may be caused by something hemispherical, but that it is of the same nature in both the southern hemisphere and northern hemisphere, although more work is needed here. Interestingly, in the distance-angle-correlation plot in 41, we see not just noise but bimodality, i.e. layering, especially for very negative values and values closer to 0 around $2000km$. This may suggest some positional dependence due to non-stationarity of the system.

7.4 L0 Coordinate Descent

We compare the quality of the l_0 coordinate descent fit, optimising parameter λ choice for BIC for the NOAA model. We can note first that the seasonality-adjusted RMSE of the NOAA data is much higher than that of the BOM data, $6.8081°C$ vs $2.8429°C$. We choose the $VAR(2)$ regression as the initial solution for the optimisation. The results are shown in 42. Even after

λ	RMSE ($°C$)	BICa ($\times 10^6$)	no. params
0.005 (BOM)	1.66	1.946	3904
0.0 (VAR)	1.865	4.983	43808
0.0125	1.886	4.497	6222

Figure 42: l_0 coordinate descent optimal BICa result on NOAA data, with raw VAR and BOM fit results for comparison.

VAR regression, the RMSE for the NOAA data has higher variance than the BOM dataset, but it experiences a more significant reduction in prediction error from the seasonal model (1.886/2.8429=0.66 for NOAA vs 1.67/2.0616=0.81) than the BOM dataset. The BICa achieved in the NOAA fit looks worse, but it is not valid to compare BICa across models with a differing number of observations. More appropriately, we still see comparable improvements in RMSE and reduction in parameters when using an l_0 fitted $VAR(2)$ model on the NOAA dataset, as we also saw on the BOM dataset. In order to achieve a BICa optimal fit, the λ value in the l_0 regression differs, in some great part due to differences in the number of observations between the two datasets. This represents a limitation of interpretting the λ choice in l_0 regression.

7.5 CMI Connectivity and Frequency Dependence

The CMI connectivity graph across the full frequency spectrum, both with and without use of l_0 regularisation on the VAR regression matrix and on the error precision matrix is shown in 43. This plot shows vastly more sporadicity that the BOM dataset, and seems much less interpretable. In the BOM dataset, most connections represented short-range connections to nearest neighbours. This dataset exhibits some wildly long connections. Either this implies some unknown deep connection between some far away sections of the data, which may be the case, due to spatially periodic processes, or the much larger noise exhibited in this dataset causes the CMI measurment to be much poorer quality than that in the BOM dataset.

Fig. 44 shows the bandpass CMI for the l_0 regularised CMI prediction on the NOAA dataset. Like the previous figure, we also see a significant amount of sporadicity in the relationships. Although all frequency bands exhibit this behaviour, the low frequency relationships exhibit many more long-range relationships. This is as we see in the BOM data, somewhat expected, as long-range effects in space have more of an opportunity to occur over longer spans of time

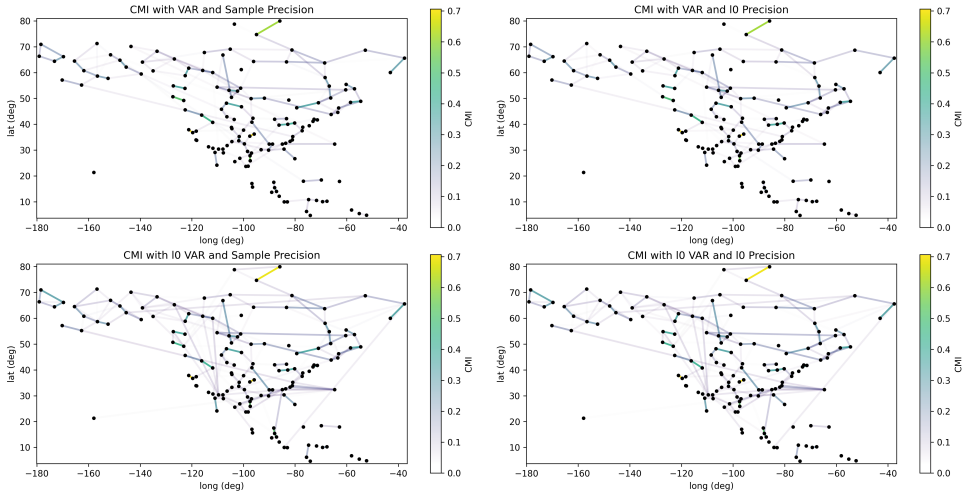

Figure 43: CMI connectivity graph for NOAA dataset with various choices of use of l_0 regularisation.

rather than shorter ones. A possible basic explanation for this is simply delays in the progression of effects between two locations. We see a similar possibility of this effect in the BOM dataset, but it is much weaker and therefore the observation could be spurious. Additionally, the north American dataset covers a much larger geographic area, allowing for more of these long-range effects to be applicable.

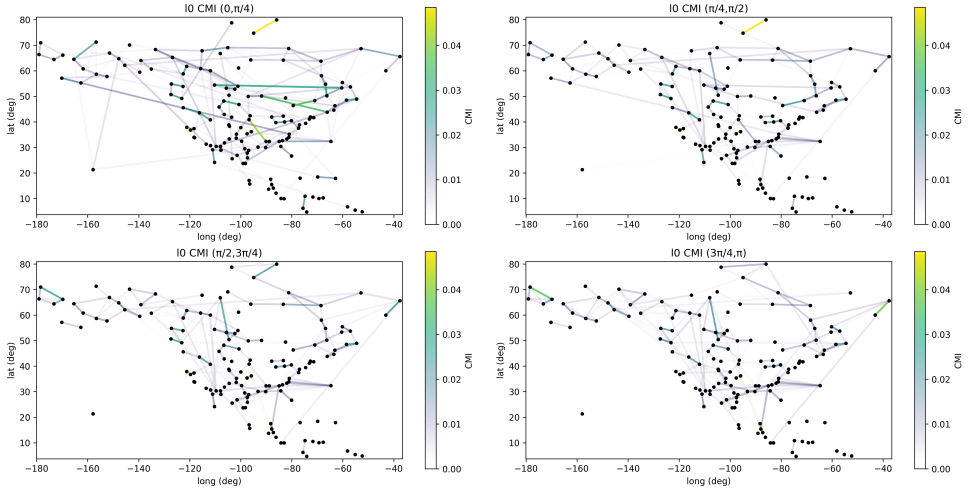

Figure 44: CMI connectivity graph for NOAA dataset in various frequency bands, using l_0 regularisation for both the VAR and precision matrices.

8 Difficulties with Inverse-Covariance System Identification

Throughout this work, we have treated the inverse covariance (precision) matrix as a more viable option for describing the behaviour of a graphical system and the covariance matrix itself.

This is not without good reason. In particular, the covariance matrix entries do not identify a unique relationship between variables, as relationships between confounding variables are included. The precision matrix expressely looks at the relationship between two variables, conditioned on the values of other variables. This is a much better method generally for measuring direct relationships between variables, rather than relationships due to confounding variables. Fig. 45 demonstrates this problem, where rain causes both umbrellas and traffic, so umbrella use may be correlated with traffic, but clearly there is no causal relationship between them.

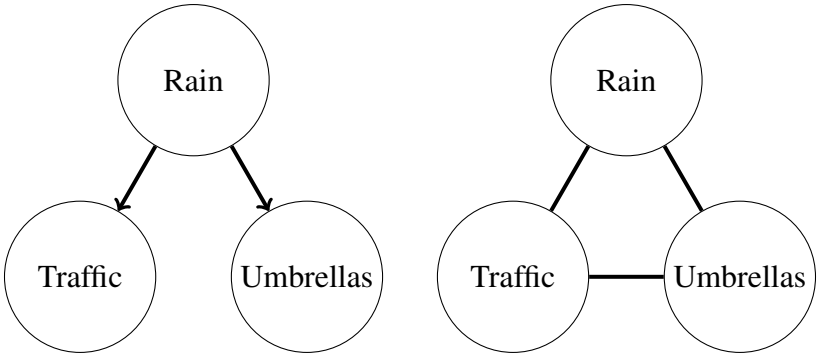

Figure 45: Causal relationship between three variables (left), vs the correlation relationship between the variables (right)

This analogy breaks down however when describing a continuous spatiotemporal process like temperature. We expect all points to have a causal effect only on those points right next to them, as all other influences are via those points. When such a system is sampled, that is we record the time series data at a discrete set of points, the correlation between these points is retained, but the precision information changes drastically. This is to say that the choice of sampling has a significant effect on the precisions between timeseries.

Not all hope is lost here for the case of identifying subsampled graphical systems, or indeed continuous systems, using the precision matrix. As we see in the case of Kriging, as discussed in 3.4.2 and demonstrated in 4.4.5, the sampling structure provides useful information about how to usefully interpolate out of sample data.

In terms of parameter efficiency, as we have seen in 4.3.3, the Matérn kernel provides an okay fit to the covariance matrix, but not a good enough one to allow for accurate representation contained within parameter-reduced l_0 coordinate descent models.

For accurate predictive results, it seems that the precision matrix is the appropriate choice of mathematical tool for feature extraction, especially in domains that are naturally graphical. In continuous domains, or heavily subsampled graphs, finding an accurate correlation function, through identification of its Matérn kernel dimension, and analysis of appropriate anisotropies, is a key step in understanding the underlying process that is driving the system.

In order to determine the manner in which the precision matrix is effected by subsampling, we can consider how the precision matrix changes when one variable is removed, as was seen in the derivation of the graphical l_0 coordinate descent algorithm in 3.6, shown below.

$$\boldsymbol{\Omega} = \begin{pmatrix} \boldsymbol{\Omega}_0 & \underset{\sim}{w} \\ \underset{\sim}{w}^\top & \omega \end{pmatrix}$$

$$\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}_0 & \underset{\sim}{s} \\ \underset{\sim}{s}^\top & \sigma \end{pmatrix}$$

$$\boldsymbol{\Sigma}_0^{-1} = \boldsymbol{\Omega}_0 - \omega^{-1} \underset{\sim}{w}\underset{\sim}{w}^\top$$

First, we must carefully note that off-diagonal entries of the precision matrix are inverted in their relationship between sign and covariance. That is, positive entries reflect negatively correlated variables (after accounting for conditional relationships), and negative entries reflect positively correlated variables.

Intuitively, this formula tells us that when one graph vertex is removed, each pair of variables connected to the removed vertex have their connection strengthed (made more negative) when their connections to the removed variable are in the same direction, and weakened when their connections are in the opposite direction.

8.1 Simulations Demonstrating Utility of Covariance Modelling vs Inverse Covariance Modelling

We can highlight the difficulty of interpretting these relationships as defining underlying graph structure by turning back to the simulations discussed in 4.4.4. We have a discretised 2D lattice domain upon which the screened Poisson equation acts, and when we randomly choose subsets of the lattice points as "sensor points", identifying the inverse covariance matrix gives us a graph that may help in optimal interpolation or prediction, through processes like Kriging, does not actually identify the underlying structure of the system, that is a 2D lattice, rather than say just a 2D array of sensor points embedded in a 3D space where the diffusion process is occuring. Whereas, if we look at the distance-correlation function in the subsampled system, we can observe a strong fit with the appropriate 2D Matérn kernel. That is, the distance-correlation function reveals an underlying topological truth about the nature of the system, where the inverse covariance does not.

Like as in 4.4.4, we run a discretised screened Poisson problem on a 2D lattice with $\lambda = 0.5$

and unit distances between lattice points. To make things clearer in this case, we do not use a flat torus $\mathbb{T}^2$, but just use a 2D plane on its own. When we sample densely, that is at every point, the CMI perfectly informs us of the structure of the graph, as in 46. The correlation-distance plot on the other hand provides us only some hazy information about the underlying topology and process producing the data.

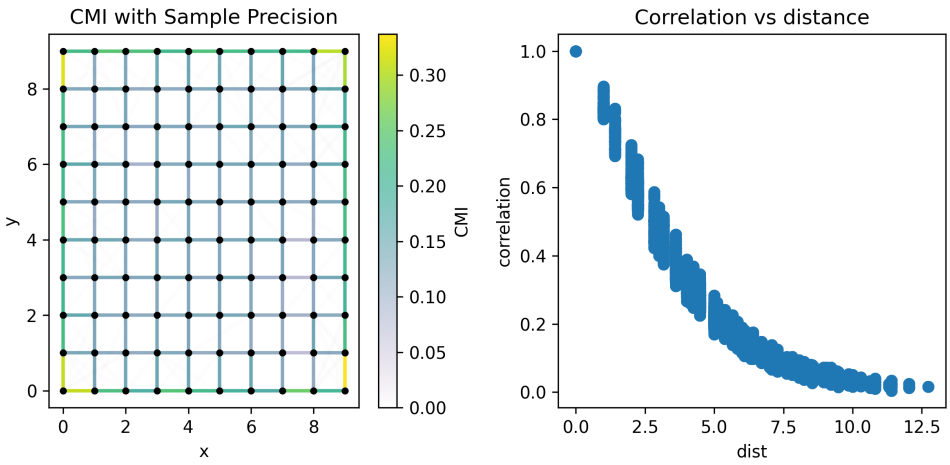

Figure 46: CMI (left) vs correlation-distance plot on a densely sampled 2D lattice.

When we sample more sparsely, say sampling only 30% of points for the same process, the situation is quite different. As in 47, the CMI gives us very little semblance of the underlying causal structure in the system, exactly because it is unaware of many of the causes. On the other hand, the correlation-distance plot encodes incredibly similar information to the more densely sampled version, experiencing very little degradation in quality.

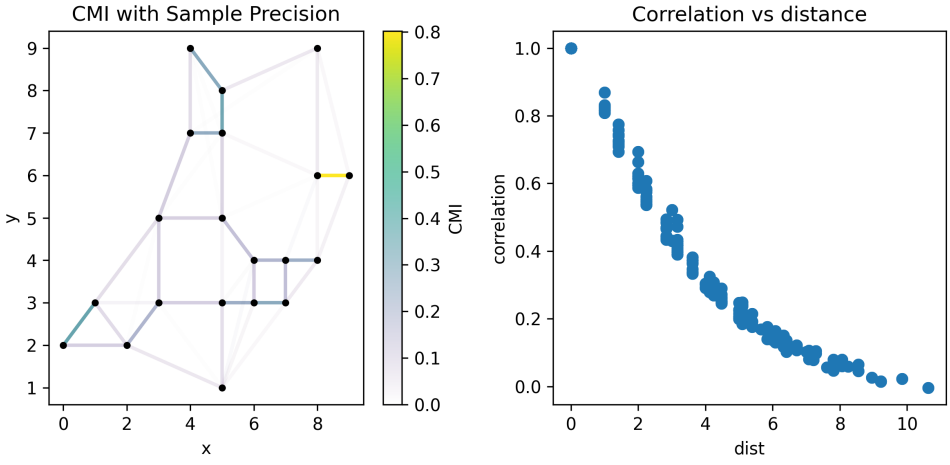

Figure 47: CMI (left) vs correlation-distance plot on a sparsely sampled 2D lattice.

What we can gather from this is that CMI is a superior tool for defining the structure of a graphical process when almost all of the relevant variables are observed. When some or many variables are unobserved, the correlation-distance plot delivers a lot of information about the

underlying system that the CMI cannot provide. In the continuous temporal and spatial case, like for weather data, and many other data types, CMI requires dense sampling to serve significant utility, whereas direct covariance/correlation prediction can still serve to identify features of the system like topology and the diffusion coefficients acting on the system, defining the scale of the spatial process.

9 Further Work

This thesis has identified areas of interest in the graph-signal processing literature. It has implemented some works and verified their results, and has identified its own results. It has also made a thrust at questioning the possible overuse of inverse covariance modelling in many graph signal processing applications. Despite this, we are left with more questions than we are answers. Some major questions are those below.

- Can we quantify the seasonality amplitude and phase, and other geographic predictors of mean, variance, and skew, reasonably from geographic principles, across the entire globe?
- How do we model non-stationary diffusion processes effectively?
- If the natural autoregressive distance-correlation functions in space are Matérn kernels, what are the natural distance-time-correlation functions for space-time autoregressive processes?
- Is there a measure of mutual information similar to CMI that accounts for phase information?
- Given limited samples, exactly how well can we identify the topology and differential equation acting on a system from its correlation function? For systems not driven by noise, but by a chosen input, are there superior methods for system identification?
- Is there a stronger geographic predictor of the partial correlations than either distance or nearest-neighbours?
- How relevant are correlation function fitting and inverse covariance fitting when an application comes with an explicitly defined graph?
- How can we apply graph Fourier transforms and filtration to real control processes?

10 Conclusion

This thesis has shown how traditional digital signal processing can be extended to new domains. Just as digital signal processing provides good solutions to practical and fundamental problems such as data compression, prediction, filtration, and interpolation, graph signal processing expands these problems to be solvable on graphical domains.

Through analysis of the Australian and North American daily temperature datasets, graph-time data can be efficiently compressed, similar to the JPEG compression algorithm. Sparsely sampled (non-radar) temperature information can still yield a reasonable prediction of next-day temperatures, with RMSEs respectively of $1.66°C$ and $1.89°C$. The graph fourier transform allows for effective filtration of noisy spatial data, akin to the Weiner filter from traditional signal processing. Kriging allows us to efficiently interpolate temperature data from a limited number of temperature sensors, conditioned on effective fitting of a correlation function. The development of a fundamental autoregressive spatial model allowed us to develop high-quality correlation function fits, allowing for system identification necessary for interpolation.

Another avenue of investigation for graph system identification was conditional mutual information, allowing for the effective identification of graph edges in a system. The development of the bandpass CMI concept allowed for the seperation of graphical relationships into long-term predictive relationships and short-term predictive relationships, and the similarity between the graph identifications was noted.

The two forms of graph system identification - correlation fitting, and the use of CMI via the inverse correlation fit, were compared and contrasted. It was proposed that CMI is always the best tool for both within-graph predictive applications, and for natural graph domains, whereas correlation fitting serves to identify the differential equation acting on a system more effectively. CMI suffers from degradation of interpretibility with sparse subsampling, where correlation fitting tends to maintain its benefits even for sparsely sampled settings.

Overall, this thesis has presented graph signal processing as a field for future research and interest with a myriad of practical applications, and a wealth of fundamental theory that has many opportunities for future development.

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, J. and Hastie, T. and Tibshirani, R. *The Elements of Statistical Learning*. 2001.

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

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

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

[12] I. Gel'fand and A. Yaglom,. Calculation of the amount of information about a random function contained in another such function. *American Mathematical Society Translations vol. 2, no. 12*, pages 199–247, 1959.

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

[14] 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.

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

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

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

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

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

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

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

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

[23] Shannon, C. E. A mathematical theory of communication. *Bell System Technical Journal*, 1948.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[41] 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.

11 Appendix A - Python Code

All figures in this report were produced using python code available at: `https://github.com/jimaginary/ELECHONS`. To produce the figures from this thesis, both python, and the python modules from ELECHONSrequirements.txt must be installed as below. All commands should be run from the root of the ELECHONS directory.

```
python -m pip install -r requirements.txt
```

Next the BOM/NOAA data should be downloaded. Each take quite some time, so it is recommended to only install the dataset that will be used. They can be downloaded as below.

```
python -m scripts.download_BOM_data
python -m scripts.download_NOAA_data
```

The data then needs to be cleaned and processed, which can be done as below.

```
python -m scripts.process_BOM_data
python -m scripts.process_NOAA_data
```

Then, we are ready to reproduce the figures from the report. Each figure can be reproduced by running the below command, with {n} replaced with the figure number from the report.

```
python -m fig_scripts.fig_{n}
```

The figure or its results (for a table) will be reproduced and outputted into either the terminal in the case of tabular results, or into the plts folder, or a subfolder. Note that some figures, such as demonstrative diagrams, that do not result from running an analysis script, do not have a corresponding generation script. Some figures are generated by the same command, in which case only the earliest such figure has a script, for example figures 5 and 6.