Efficient Filtering and Fitting of Models Derived from Integro-Difference Equations

Evan Tate Paterson Hughes

Table of contents

Chapter 1

Introduction

The Integro-Difference equation model (here abbreviated as IDEM [1]) is dynamics-based spatio-temporal aiming to model diffusion and convection by making the value of a process a weighted average of it's previous time, plus noise.

[NOTE: I intend to create a more thorough background for the introduction here.]

[1] Historically, this has been abbreviated as IDE. However, with that abbreviation almost universally meaning 'Integrated Development Environment', here, we choose to include the 'M' in the abbreviation.

Chapter 2

Integro-difference Based Dynamics

As common and widespread as the problem is, spatio-temporal modelling still presents a great deal of difficulty. Inherently, Spatio-Temporal datasets are almost always high-dimensional, and repeated observations are usually not possible.

Traditionally, the problem has been tackled by the moments (usually the means and covariances) of the process in order to make inference (Wikle, Zammit-Mangion, and Cressie (2019), for example, call this 'descriptive' modelling). While this method can be sufficient for many problems, there are many cases where we are underutilizing some knowledge of the underlying dynamic systems involved. For instance, in temperature models, we know that temperature has movement (convection) and spread (diffusion), and that the state at any given time will depend on its state at previous times [1]. We call models which make use of this 'dynamic' models.

A general way of writing such hierarchical dynamical models might be

$$\begin{aligned} Y_{t+1}(\cdot) &= \mathscr{M}_t(Y_0(\cdot), \dots, Y_t(\cdot)) + \omega_t(\cdot), \quad t = 0, \dots, T-1, \\ Z_t(\cdot) &= \mathscr{O}_t(Y_t(\cdot)) + x(\cdot)^\intercal \boldsymbol{\beta} + \epsilon_t(\cdot), \quad t = 1, \dots, T. \end{aligned}$$

This describes the scalar random fields $Z_t(\cdot), Y_t(\cdot) \in \mathbb{R}$ over the space $\mathscr{D} \subset \mathscr{R}^d$, which are the observed data and unobserved dynamic process, respectively. $\mathscr{M}_t$ here is a non-random 'propagation operator', defining how the process evolves with respect to it's previous state(s), and $\mathscr{O}_t$ is a non-random 'observation operator', defining how observations of a given process state are taken. Both these fields have random (usually time-independent) additive random fields, $\omega_t(\cdot), \epsilon_t(\cdot)$, and we also include non-random measured linear covariate terms $x(\cdot)^\intercal \boldsymbol{\beta}$.

If we discretize the space into n \$spatial locations $\{\boldsymbol{s}_i\}_{i=1,\dots,n}$, assume the operator are linear, assert a Markov condition, and assume the errors are all normal, we get a simple linear dynamic system;

$$\begin{aligned} \boldsymbol{Y}_{t+1} &= \boldsymbol{M}_t \boldsymbol{Y}_t + \boldsymbol{\omega}_t, \quad t = 0, \dots, T-1, \\ \tilde{\boldsymbol{Z}}_t &= \boldsymbol{O}_t \boldsymbol{Y}_t + \boldsymbol{\epsilon}_t, \quad t = 1, \dots, T, \end{aligned} \tag{2.1}$$

where we have written $\boldsymbol{Y}_t = (Y_t(\boldsymbol{s}_1), \dots, Y_t(\boldsymbol{s}_n))$, and similar for $\boldsymbol{Z}_t, \boldsymbol{\epsilon}_t$ and $\boldsymbol{\omega}_t$, and we have written $\tilde{\boldsymbol{Z}}_t = \boldsymbol{Z}_t + X^\intercal \boldsymbol{\beta}$. This is a well-known type of system, the process Y can easily be estimated either directly of with a Kalman filter/smoother and variants, which will be discussed later.

However, this model is restrictive and high-dimensional; $\boldsymbol{M}_t$, the primary quantities which needs estimation, is of dimension $n \times n$, of which there are T matrices to be estimated. Even if we allow the propagation matrix to be invariant in time, we can still only make predictions at the stations $\{\boldsymbol{s}_i\}$.

[1] at least, in a discrete-time scenario. Integro-difference based mechanics can be derived from continuous-time convection-diffusion processes, see Liu, Yeo, and Lu (2022)

This motivates a different approach; in particular, one which allows us to estimate the random field at arbitrary points $Y_t(\boldsymbol{s})$ using some spectral decomposition, which would alleviate these problems.

The Integro-difference equation model attempts to generalise Equation 2.1 into the continuous space by replacing the discrete linear $\boldsymbol{M}_t$ by a continuous integral equivalent;

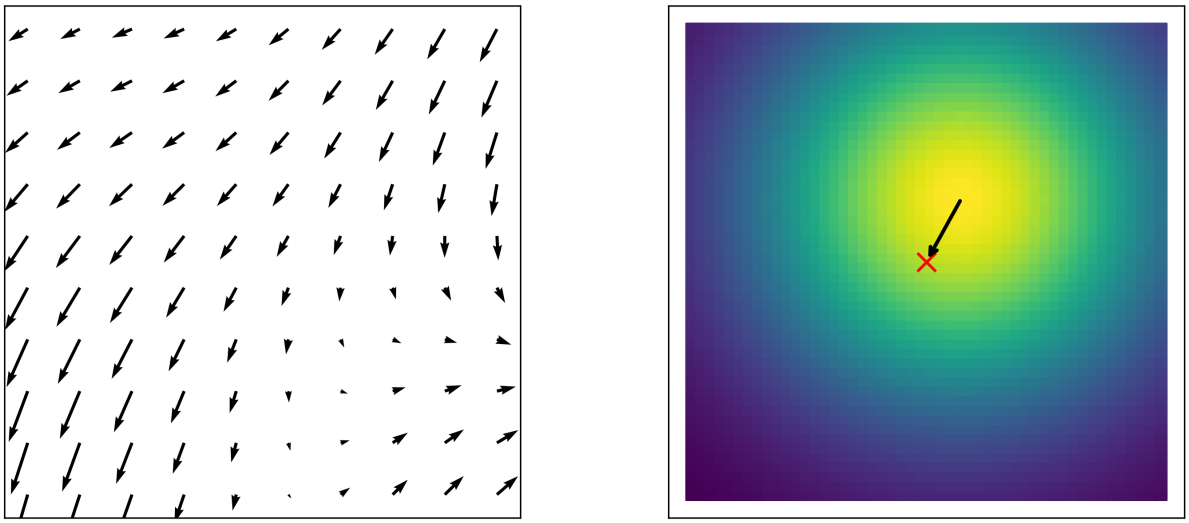
$$\begin{aligned} Y_{t+1}(\boldsymbol{s}) &= \int_{\mathscr{D}_s} \kappa_t(\boldsymbol{s},\boldsymbol{r}) Y_t(\boldsymbol{r}) d\boldsymbol{r} + \omega_t(\boldsymbol{s}), \quad t = 0, \dots, T-1, \\ Z_t(\boldsymbol{s}) &= Y_t(\boldsymbol{s}) + X(\boldsymbol{s})^\intercal \boldsymbol{\beta} + \epsilon_t(\boldsymbol{s}), \quad t = 1, \dots, T. \end{aligned} \tag{2.2}$$

Where $\omega_t(\boldsymbol{s})$ is a small scale gaussian variation with no temporal dynamics (Cressie and Wikle 2015 call this a 'spatially descriptive' component), $\boldsymbol{X}(\boldsymbol{s})$ are spatially varying covariates (for example, in a large-scale climate scenario, this might be latitude, concentration of some chemical/element like nitrogen) $\kappa(\boldsymbol{s},\boldsymbol{r})$ is the driving 'kernel' function, and ϵ_t is a Gaussian white noise 'measurement error' term.

Our operator is now $\mathscr{M}(Y_t(\boldsymbol{s})) = \int_{\mathscr{D}_s} \kappa_t(\boldsymbol{s},\boldsymbol{r}) Y_t(\boldsymbol{r}) d\boldsymbol{r}$, which can model diffusion and convection by choosing the shape of κ (which, from now on, we will assume to be temporally invariant). This kernel defines how each point in space is affected by every other point in space at the previous time. For example, if we choose a Gaussian-like shape,

$$\kappa(\boldsymbol{s},\boldsymbol{r};\boldsymbol{m},a,b) = a \exp\left(-\frac{1}{b}|\boldsymbol{s}-\boldsymbol{r}+\boldsymbol{m}(\boldsymbol{s})|^2\right),$$

then the 'flow' would be in the direction of $-\boldsymbol{m}(\boldsymbol{s})$, and the diffusion would be controlled by b and a. This creates a 'spatially variant kernel', where the direction of flow varies across the space, as in Figure 2.1.



(a) Invariant Kernel Direction

(b) Invariant Kernel Strength

Figure 2.1: A spatially variant kernel across the region $[0,1]\times[0,1]$. The kernel direction is shown on the left, and on the right is the amount that each point affects the point $(0.5, 0.5)$, marked with a red cross. 'Flow' is allowed to vary by a function $\boldsymbol{m}(\boldsymbol{s})$ which is chosen randomly using a basis expansion (see Section 3.3). The other two parameters are set at $a = 150$, $b = 0.2$.

Chapter 3

Spectral Representations

The key to being able to computationally work with IDEMs, as perhaps originally made by Wikle and Cressie (1999), is to work with the spectral decomposition of the process, in order to coerce the model hierarchy into a more familiar linear dynamical system form, like Equation 2.1.

This kind of dimension-reduction allows us to parametrise spatial fields with as few or as many parameters as we want.

3.1 Process decomposition

Choose a complete class of spatial spectral basis functions, $\{\phi_i(\cdot) : \mathscr{D} \to \mathbb{R}\}_{i=1,\dots}$, and decompose the process spatial field at each time;

$$Y_t(\boldsymbol{s}) \approx \sum_{i=1}^{r} \alpha_{i,t}\phi_i(\boldsymbol{s}), \quad t = 0, \dots, T. \tag{3.1}$$

where we truncate the expansion at some $r \in \mathbb{N}$. Notice that we can write this in vector/matrix form, where we consider the vector field $\boldsymbol{\phi}(\cdot) = (\phi_1(\cdot), \dots, \phi_r(\cdot))^\intercal$; considering times $t = 1, 2, \dots, T$, we set

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{s}) &= (\phi_1(\boldsymbol{s}), \phi_2(\boldsymbol{s}), \dots, \phi_r(\boldsymbol{s}))^\intercal, \\ \boldsymbol{\alpha}_t &= (\alpha_{1,t}, \alpha_{2,t}, \dots, \alpha_{r,t})^\intercal. \end{aligned} \tag{3.2}$$

Now, (Equation 3.1) gives us, for any $\boldsymbol{s} \in \mathscr{D}$,

$$Y(\boldsymbol{s}; t) \approx \boldsymbol{\phi}^\intercal(\boldsymbol{s})\alpha(t). \tag{3.3}$$

We can effectively now work exclusively with $\boldsymbol{\alpha}_t = (\alpha_{1,t}, \dots, \alpha_{r,t})^\intercal$. To do so, we need to find the evolution equation of $\boldsymbol{\alpha}_t$, as given below.

Theorem 3.1.1 (Spectral form of the state evolution)**.** *Define the* Gram matrix*;*

$$\Psi \coloneqq \int_{\mathscr{D}_s} \boldsymbol{\phi}(\boldsymbol{s})\boldsymbol{\phi}(\boldsymbol{s})^\intercal d\boldsymbol{s}. \tag{3.4}$$

Then, the basis coefficients evolve by the equation

$$\boldsymbol{\alpha}_{t+1} = M\boldsymbol{\alpha}_t + \boldsymbol{\eta}_t, \tag{3.5}$$

where $M = \Psi^{-1}\int\int \boldsymbol{\phi}(\boldsymbol{s})\kappa(\boldsymbol{s},\boldsymbol{r})\boldsymbol{\phi}(\boldsymbol{r})^\intercal d\boldsymbol{r}d\boldsymbol{s}$ *and* $\boldsymbol{\eta}_t = \Psi^{-1}\int \boldsymbol{\phi}(\boldsymbol{s})\omega_t(s)d\boldsymbol{s}$.

Proof. (Adapting from Dewar, Scerri, and Kadirkamanathan 2008), write out the process equation, (Equation 2.2), using the first equation of (Equation 3.3);

$$Y_{t+1}(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^\intercal \alpha_{t+1} = \int_{\mathscr{D}_s} \kappa(\boldsymbol{s},\boldsymbol{r})\boldsymbol{\phi}(\boldsymbol{r})^\intercal \boldsymbol{\alpha}_t d\boldsymbol{r} + \omega_t(\boldsymbol{s}),$$

We then multiply both sides by $\boldsymbol{\phi}(s)$ and integrate over $\boldsymbol{s}$

$$\begin{aligned}\int_{\mathscr{D}_s} \boldsymbol{\phi}(\boldsymbol{s})\boldsymbol{\phi}(\boldsymbol{s})^\intercal d\boldsymbol{s}\boldsymbol{\alpha}_{t+1} &= \int \boldsymbol{\phi}(\boldsymbol{s})\int \kappa(\boldsymbol{s},\boldsymbol{r})\boldsymbol{\phi}(\boldsymbol{r})^\intercal d\boldsymbol{r}d\boldsymbol{s}\ \boldsymbol{\alpha}_t + \int \boldsymbol{\phi}(\boldsymbol{s})\omega_t(s)d\boldsymbol{s}\\ \Psi\boldsymbol{\alpha}_{t+1} &= \int\int \boldsymbol{\phi}(\boldsymbol{s})\kappa(\boldsymbol{s},\boldsymbol{r})\boldsymbol{\phi}(\boldsymbol{r})^\intercal d\boldsymbol{r}d\boldsymbol{s}\ \boldsymbol{\alpha}_t + \int \boldsymbol{\phi}(\boldsymbol{s})\omega_t(s)d\boldsymbol{s}.\end{aligned}$$

So, finally, pre-multipling by the inverse of the gram matrix, Ψ^{-1} (Equation 3.4), we arrive at the result. □ □

3.2 Spectral form of the Process Noise

We still have to set out what the process noise, $\omega_t(\boldsymbol{s})$, and it's spectral counterpart, $\boldsymbol{\eta}_t$, are. Dewar, Scerri, and Kadirkamanathan (2008) fix the variance of $\omega_t(\boldsymbol{s})$ to be uniform and uncorrelated across space and time, with $\omega_t(\boldsymbol{s}) \sim \mathcal{N}(0,\sigma^2)$ It is then easily shown that $\boldsymbol{\eta}_t$ is also normal, with $\boldsymbol{\eta}_t \sim \mathcal{N}(0,\sigma^2\Psi^{-1})$.

However, in practice, we simulate in the spectral domain; that is, if we want to keep things simple, it would make sense to specify (and fit) the distribution of $\boldsymbol{\eta}_t$, and compute the variance of $\omega_t(\boldsymbol{s})$ if needed.

Lemma 3.2.1. *Let* $\boldsymbol{\eta}_t \sim \mathcal{N}(0,\Sigma_\eta)$, *and* $\mathbb{C}\text{ov}[\boldsymbol{\eta}_t,\boldsymbol{\eta}_{t+\tau}] = 0$, $\forall\tau > 0$. *Then* $\omega_t(\boldsymbol{s})$ *has covariance*

$$\mathbb{C}\text{ov}[\omega_t(\boldsymbol{s}),\omega_{t+\tau}(\boldsymbol{r})] = \begin{cases}\boldsymbol{\phi}(\boldsymbol{s})^\intercal\Sigma_\eta\boldsymbol{\phi}(\boldsymbol{r}) & \text{if } \tau = 0\\ 0 & \text{else}\end{cases}$$

Proof. Consider $\Psi\boldsymbol{\eta}_t$, and consider the case $\tau = 0$. It is clearly normal, with zero expectation and variance (using Equation 3.4),

$$\begin{aligned}\mathbb{V}\text{ar}[\Psi\boldsymbol{\eta}_t] &= \Psi\,\mathbb{V}\text{ar}[\boldsymbol{\eta}_t]\Psi^\intercal = \Psi\Sigma_\eta\Psi^\intercal,\\ &= \int_{\mathscr{D}_s}\boldsymbol{\phi}(\boldsymbol{s})\boldsymbol{\phi}(\boldsymbol{s})^\intercal d\boldsymbol{s}\ \Sigma_\eta\int_{\mathscr{D}_s}\boldsymbol{\phi}(\boldsymbol{r})\boldsymbol{\phi}(\boldsymbol{r})^\intercal d\boldsymbol{r}\\ &= \int\int_{\mathscr{D}_s^2}\boldsymbol{\phi}(\boldsymbol{s})\boldsymbol{\phi}(\boldsymbol{s})^\intercal\ \Sigma_\eta\ \boldsymbol{\phi}(\boldsymbol{r})\boldsymbol{\phi}(\boldsymbol{r})^\intercal d\boldsymbol{r}d\boldsymbol{s}\end{aligned} \tag{3.6}$$

Since it has zero expectation, we also have

$$\begin{aligned}\mathbb{V}\text{ar}[\Psi\boldsymbol{\eta}_t] &= \mathbb{E}[(\Psi\boldsymbol{\eta}_t)(\Psi\boldsymbol{\eta}_t)^\intercal] = \mathbb{E}[\Psi\boldsymbol{\eta}_t\boldsymbol{\eta}_t^\intercal\Psi^\intercal] \\ &= \mathbb{E}\left[\int_{\mathscr{D}_s}\boldsymbol{\phi}(\boldsymbol{s})\omega_t(\boldsymbol{s})d\boldsymbol{s}\int_{\mathscr{D}_s}\boldsymbol{\phi}(\boldsymbol{r})^\intercal\omega_t(\boldsymbol{r})d\boldsymbol{r}\right] \\ &= \int\int_{\mathscr{D}_s^2}\boldsymbol{\phi}(\boldsymbol{s})\,\mathbb{E}[\omega_t(\boldsymbol{s})\omega_t(\boldsymbol{r})]\,\boldsymbol{\phi}(\boldsymbol{r})^\intercal d\boldsymbol{s}d\boldsymbol{r}.\end{aligned} \tag{3.7}$$

We can see that, comparing (Equation 3.6) and (Equation 3.7), we have

$$\mathbb{C}\text{ov}[\omega_t(\boldsymbol{s}),\omega_t(\boldsymbol{r})] = \mathbb{E}[\omega_t(\boldsymbol{s})\omega_t(\boldsymbol{r})] = \boldsymbol{\phi}(\boldsymbol{s})^\intercal\Sigma_\eta\boldsymbol{\phi}(\boldsymbol{r}).$$

Since, once again, $\mathbb{E}[\boldsymbol{\omega}_t(\boldsymbol{s})] = 0$.

For the $\tau \neq 0$ case, it is simple to show that the covariance is 0.

□ □

3.3 Kernel Parameterisations

Next is the part of the system, which defines the dynamics; the kernel function, κ. There are a few ways to handle the kernel. One of the most obvious is to expand it out into a spectral decomposition as well;

$$\kappa \approx \sum_i \beta_i\psi(\boldsymbol{s},\boldsymbol{r}).$$

This can allow for a wide range of interestingly shaped kernel functions, but see how these basis functions must now act on $\mathbb{R}^2\times\mathbb{R}^2$; to get a wide enough space of possible functions, we would likely need many terms in the spectral expansion.

A much simpler approach would be to simply parametrise the kernel function, to $\kappa(\boldsymbol{s},\boldsymbol{r},\boldsymbol{\theta}_\kappa)$. We then establish a simple shape for the kernel (e.g. Gaussian) and rely on very few parameters (for example, scale, shape, offsets). The example kernel used in the `jaxidem` is a Gaussian-shape kernel;

$$\kappa(\boldsymbol{s},\boldsymbol{r};\boldsymbol{m},a,b) = a\exp\left(-\frac{1}{b}|\boldsymbol{s}-\boldsymbol{r}+\boldsymbol{m}|^2\right).$$

Of course, this kernel lacks spatial dependence. We can add spatial variance back by adding dependence on $\boldsymbol{s}$ to the parameters, for example, varying the offset term as $\boldsymbol{m}(\boldsymbol{s})$. Of course, now we are back to having entire functions as parameters, but taking the spectral decomposition of the parameters we actually want to be spatially variant seems like a reasonable middle ground (Cressie and Wikle 2015). The actual parameters of such a spatially-variant kernel are then the spectral coefficients for the expansion of any spatially variant parameters, as well as any constant parameters. This is precisely what is plotting in Figure 2.1, where the spectral coefficients are randomly sampled from a multivariate normal distribution;

$$\boldsymbol{m}(\boldsymbol{s}) = \begin{pmatrix}\sum_{i=1}^{r_m}\phi_{\kappa,i}(\boldsymbol{s})m_i^{(x)} \\ \sum_{i=1}^{r_m}\phi_{\kappa,i}(\boldsymbol{s})m_i^{(y)}\end{pmatrix},$$

where $m_i^{(x)}$ and $m_i^{(y)}$ are coefficients for the x and y coordinates respectively, and $\phi_{\kappa,i}(\boldsymbol{s})$ are basis functions (e.g. bisquare [1]) functions in Figure 2.1).

[1]The bisquare functions, here, $\phi_i(\boldsymbol{s}) = [1-\frac{\|\boldsymbol{s}-\boldsymbol{c}_i\|}{w_i}]^2\cdot I(\|\boldsymbol{s}-\boldsymbol{c}_i\|<w_i)$ for i 'centroids' or 'knots', $\boldsymbol{c}_i\in\mathscr{D}$, each with 'radius' w_i

3.4 IDEM as a linear dynamical system

To summarise, we have taken a truncated spectral decomposition to write the Integro-difference equation model as a more traditional linear dynamical system form (Equation 3.5). All that is left is to include our observations in our system.

Lets assume that at each time t there are n_t observations at locations $\boldsymbol{s}_{1,t}, \dots, \boldsymbol{s}_{n_t,t}$. We write the vector of the process at these points as $\boldsymbol{Y}(t) = (Y(s_{1,t}; t), \dots, Y(s_{n_t,t}; t))^\intercal$, and, in it's expanded form $\boldsymbol{Y}_t = \Phi_t \boldsymbol{\alpha}_t$, where $\Phi \in \mathbb{R}^{r \times n_t}$ is

$$\{\Phi_t\}_{i,j} = \phi_i(s_{j,t}).$$

For the covariates, we write the matrix $X_t = (\boldsymbol{X}(\boldsymbol{s}_{1,t}), \dots, \boldsymbol{X}(\boldsymbol{s}_{1=n_t,t})^\intercal$. We then have

$$\begin{aligned}
\boldsymbol{Z}_t &= \Phi \boldsymbol{\alpha}_t + X_t \boldsymbol{\beta} + \boldsymbol{\epsilon}_t, \quad t = 1, \dots, T, \\
\boldsymbol{\alpha}_{t+1} &= M \boldsymbol{\alpha}_t + \boldsymbol{\eta}_t, \quad t = 0, 1, \dots, T-1, \\
M &= \int_{\mathscr{D}_s} \boldsymbol{\phi}(\boldsymbol{s}) \boldsymbol{\phi}(\boldsymbol{s})^\intercal d\boldsymbol{s} \int_{\mathscr{D}_s^2} \boldsymbol{\phi}(\boldsymbol{s}) \kappa(\boldsymbol{s}, \boldsymbol{r}; \boldsymbol{\theta}_\kappa) \boldsymbol{\phi}(\boldsymbol{r})^\intercal d\boldsymbol{r} d\boldsymbol{s},
\end{aligned}$$

Writing $\tilde{\boldsymbol{Z}}_t = \boldsymbol{Z}_t - X_t \boldsymbol{\beta}$,

$$\begin{aligned}
\tilde{\boldsymbol{Z}}_t &= \Phi_t \boldsymbol{\alpha}_t + \boldsymbol{\epsilon}_t, \qquad t = 1, 2, \dots, T, \\
\boldsymbol{\alpha}_{t+1} &= M \boldsymbol{\alpha}_t + \boldsymbol{\eta}_t, \qquad t = 0, 1, \dots, T-1.
\end{aligned} \tag{3.8}$$

We should also initialise $\boldsymbol{\alpha}_0 \sim \mathcal{N}^r(\boldsymbol{m}_0, \Sigma_0)$, and fix simple distributions to the noise terms,

$$\begin{aligned}
\boldsymbol{\epsilon}_t &\overset{\text{iid}}{\sim} \mathcal{N}_{n_{\text{obs}}}(0, \Sigma_\epsilon), \\
\boldsymbol{\eta}_t &\overset{\text{iid}}{\sim} \mathcal{N}_R(0, \Sigma_\eta),
\end{aligned}$$

which are independent in time.

As in, for example, (Wikle and Cressie 1999), Equation 3.8 is now in a traditional enough form that the Kalman filter can be applied to filter and compute many necessary quantities for inference, including the marginal likelihood. We can use these quantities in either an EM algorithm or a Bayesian approach, or directly maximise the marginal data likelihood

We now move on to an example simulation of this kind of model using its spectral decomposition and `jaxidem`.

3.5 Example Simulation

We can now use the above to simulate easily from such models; once we have chosen the appropriate decompositions, we simply compute M and propagate $\boldsymbol{\alpha}_t$ as we would when simulating any other linear dynamic system. We then use the spectral coefficients to generate $Y_t(\boldsymbol{s})$ and $Z_t(\boldsymbol{s})$ in the obvious way.

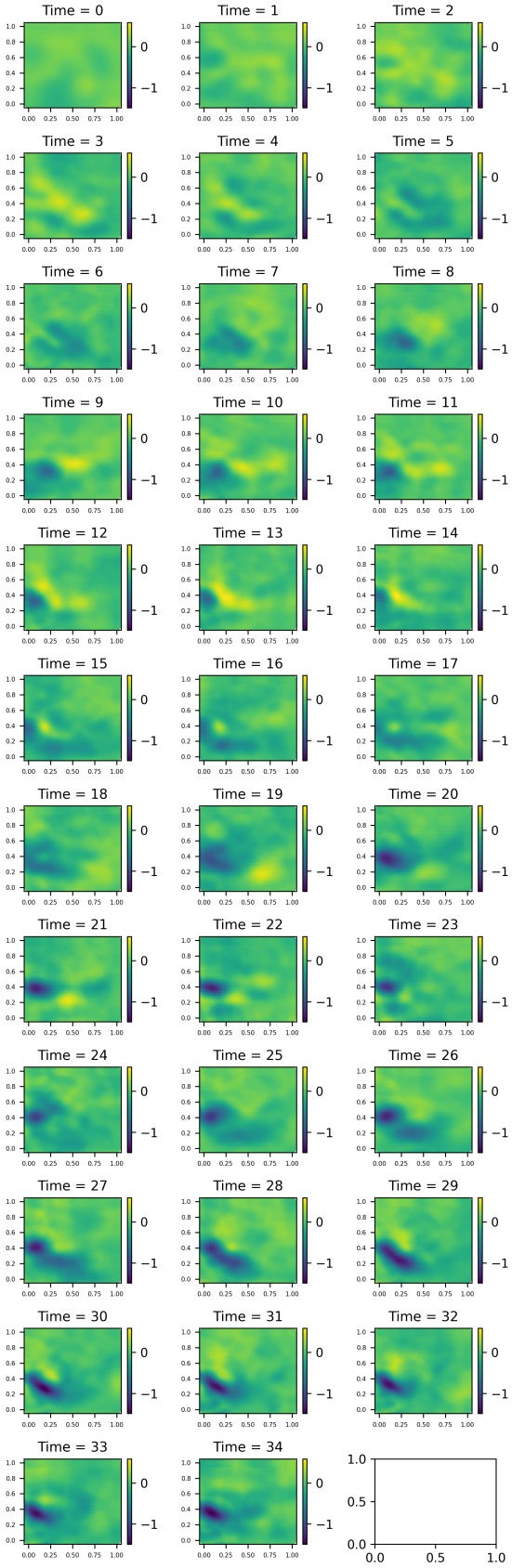
`jaxidem` implements this in the function `sim_idem`, or through the more user-friendly method `idem.IDEM.simulate`. An object of the `IDEM` class contains all the necessary information about basis decompositions, and the simulate methods calls `simIDEM` without compromising its jit-ability (although just-in-time computation obviously isn't as important for simulation, the jit-ed function could save compile time if someone want to simulate from many models).

The `gen_example_idem` method creates a simple IDEM object without many required parameters;

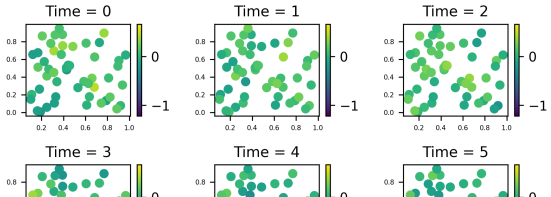
```
key = jax.random.PRNGKey(42)
keys = jax.random.split(key, 3)

model = idem.gen_example_idem(keys[0], k_spat_inv=False)
# Simulation
T = 35
nobs = 50

coords = jax.random.uniform(
                keys[1],
                shape=(nobs, 2),
                minval=0,
                maxval=1,
            )

times = jnp.repeat(jnp.arange(1, T + 1), coords.shape[0])
rep_coords = jnp.tile(coords, (T, 1))
x = rep_coords[:,0]
y = rep_coords[:,1]

process_data, obs_data = model.simulate(keys[2], x, y, times)
```

The resulting objects are of class `st_data`, containing a couple of niceties for handling spatio-temporal data, while still storing all data as JAX arrays. For example, the `show_plot`, `save_plot` and `save_gif` methods provide easy plotting;

```
process_data.save_plot('figure/process_data_example.png')
obs_data.save_plot('figure/obs_data_example.png')
```



(a) Process Simulation



Chapter 4

The Kalman filter, and its many flavours

The Kalman filter gives us linear estimates for the distribution of $\boldsymbol{\alpha}_r \mid \{\boldsymbol{Z}_t = \boldsymbol{z}_t\}_{t=0,\dots,r}$ in any dynamical system like Equation 2.1. Now that we have written the IDEM in this form, this filter can now help compute estimates for the moments of the state $\boldsymbol{\alpha}_t$. The Kalman filter also computes the marginal data likelihood, $\pi(\{\boldsymbol{z}_t\}_{t=1,\dots,T} \mid \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the model parameters. This allows us to perform maximum-likelihood estimation (as well as any other likelihood-based method of optimization). We will not prove the Kalman filter here, (for that, see, for example, Shumway, Stoffer, and Stoffer 2000).

Since it's initial formulation in the 50s by a variety of authors (Kálmán included) there have been many variations of the Kalman filter proposed, even as recently as this decade with the temporally paralellised Kalman filter, more technically a variant of the information form of the Kalman filter, by Särkkä and García-Fernández (2020).

4.1 The Kalman Filter

Firstly, we should establish some notation. Write

$$\begin{aligned} m_{i|j} &= \mathbb{E}[\boldsymbol{\alpha}_i \mid \{\boldsymbol{Z}_t = \boldsymbol{z}_t\}_{t=0,\dots,j}], \\ P_{i|j} &= \mathbb{V}\mathrm{ar}[\boldsymbol{\alpha}_i \mid \{\boldsymbol{Z}_t = \boldsymbol{z}_t\}_{t=0,\dots,j}], \\ P_{i,j|k} &= \mathbb{C}\mathrm{ov}[\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_k \mid \{\boldsymbol{Z}_t = \boldsymbol{z}_t\}_{t=0,\dots,k}]. \end{aligned}$$

For the initial terms, we choose Bayesian-like prior moments $m_{0|0} = m_0$ and $P_{0|0} = \Sigma_0$. For convenience and generality, we write Σ_η and Σ_ϵ for the variance matrices of the process and observations. Note that, if the number of observations change at each time point (for example, due to missing data), then Σ_ϵ should be time varying (even in its shape); we could either always keep it as uncorrelated so that $\Sigma_\epsilon = \mathrm{diag}(\sigma_\epsilon^2)$, or perhaps put some kind of distance-dependant covariance function to it.

To move the filter forward, that is, given $m_{t|t}$ and $P_{t|t}$, to get $m_{t+1|t+1}$ and $P_{t+1|t+1}$, we first *predict*

$$\begin{aligned} \boldsymbol{m}_{t+1|t} &= M\boldsymbol{m}_{t|t}, \\ P_{t+1|t} &= MP_{t|t}M^\intercal + \Sigma_\eta, \end{aligned} \tag{4.1}$$

then we add our new information, *update*, with z_t;

$$\begin{aligned} \boldsymbol{m}_{t+1|t+1} &= \boldsymbol{m}_{t+1|t} + K_{t+1}\boldsymbol{e}_{t+1} \\ P_{t+1|t+1} &= [I - K_{t+1}\Phi_{t+1}]P_{t+1|t} \end{aligned} \tag{4.2}$$

where K_{t+1} is the *Kalman gain*;

$$K_{t+1} = P_{t+1|t}\Phi_{t+1}^{\intercal}[\Phi_{t+1}P_{t+1|t}\Phi_{t+1}^{\intercal} + \Sigma_{\epsilon}]^{-1}, \quad t = 0, \ldots, T-1,$$

and $\boldsymbol{e}_{t+1}$ are the *prediction errors*

$$\boldsymbol{e}_{t+1} = \tilde{\boldsymbol{z}}_{t+1} - \Phi_{t+1}\boldsymbol{m}_{t+1|t}, \quad t = 1, \ldots, T.$$

Starting with m_0 and P_0, we can then iteratively move across the data to eventually compute $m_{T|T}$ and $P_{T|T}$.

Assuming Gaussian all random variables here are Gaussian, this is the optimal mean-square estimators for these quantities, but even outside of the Gaussian case, these are optimal for the class of *linear* operators.

We can compute the marginal data likelihood alongside the Kalman filter using the prediction errors $\boldsymbol{e}_t$. These, under the assumptions we have made about $\boldsymbol{\eta}_t$ and $\boldsymbol{\epsilon}_t$ being normal, are also normal with zero mean and variance

$$\mathbb{V}\text{ar}[\boldsymbol{e}_t] = \Sigma_t = \Phi_t P_{t|t-1}\Phi_t^{\intercal} + \Sigma_{\epsilon}. \tag{4.3}$$

Therefore, the log-likelihood at each time is

$$\mathscr{L}(Z \mid \boldsymbol{\theta}) = -\frac{1}{2}\sum \log\det(\Sigma_t(\boldsymbol{\theta})) - \frac{1}{2}\sum \boldsymbol{e}_t(\boldsymbol{\theta})^{\intercal}\Sigma_t(\boldsymbol{\theta})^{-1}\boldsymbol{e}_t(\boldsymbol{\theta}) - \frac{n_t}{2}\log(2 * \pi).$$

Summing these across time, we get the log likelihood for all the data.

A simplified example of the Kalman filter function, written to be JAX compatible, used in the package is this;

```
@jax.jit
def kalman_filter(m_, P_0, M, PHI_obs, Sigma_eta, Sigma_eps, ztildes):
    nbasis = m_0.shape[0]
    nobs = ztildes.shape[0]

    @jax.jit
    def step(carry, z_t):
        m_tt, P_tt, _, _, ll, _ = carry

        # predict
        m_pred = M @ m_tt
        P_pred = M @ P_tt @ M.T + Sigma_eta

        # Update
        # Prediction Errors
        eps_t = z_t - PHI_obs @ m_pred

        Sigma_t = PHI_obs @ P_pred @ PHI_obs.T + Sigma_eps
        # Kalman Gain
        K_t = (jnp.linalg.solve(Sigma_t, PHI_obs)@ P_pred.T).T

        m_up = m_pred + K_t @ eps_t
        P_up = (jnp.eye(nbasis) - K_t @ PHI_obs) @ P_pred

        # likelihood of epsilon, using cholesky decomposition
        ll_new = ll - 0.5 * n * jnp.log(2*jnp.pi) - \
            0.5 * jnp.log(jnp.linalg.det(Sigma_t)) -\
            0.5 * e.T @ jnp.linalg.solve(Sigma_t, e)

        return (m_up, P_up, m_pred, P_pred, ll_new, K_t), (m_up, P_up, m_pred, P_pred, ll_new, K_t,)

    carry, seq = jl.scan(
        step,
        (m_0, P_0, m_0, P_0, 0, jnp.zeros((nbasis, nobs))),
        ztildes.T,
    )

    return (carry[4], seq[0], seq[1], seq[2][1:], seq[3][1:], seq[5][1:])
```

For the documentation of the method provided by the package, see `filter_smoother_functions.kalman_filter`.

4.2 The Information Filter

In some computational scenarios, it is beneficial to work with vectors of consistent dimension. In Python JAX, the efficient `scan` method works only with such arrays; JAX has no support for jagged arrays, and traditional for loops will likely lead to long compile times when jit-compiled. Although there are some tools in JAX to get around this problem (namely the `jax.tree` functions which allow mapping over PyTrees), scan is still a large problem; since the Kalman filter is, at it's core, a scan-type operation (scanning over the data), this causes a large problem when the observation dimension is changing, as is frequent with many spatio-temporal data.

But it is possible to re-write the Kalman filter in a way which is compatible with this kind of data. The 'information filter' (sometimes called inverse Kalman filter or other names) involves transforming the data into its 'information form', which will always have consistent dimension, allowing us to avoid jagged scans.

The information filter is simply the Kalman filter re-written to use the Gaussian distribution's canonical parameters [1], those being the information vector and the information matrix. If a Gaussian distribution has mean $\boldsymbol{\mu}$ and variance matrix Σ, then the corresponding *information vector* and *information matrix* is $\nu = \Sigma^{-1}\mu$ and $Q = \Sigma^{-1}$, correspondingly.

Theorem 4.2.1. *The Kalman filter can be rewritten in information form as follows (for example, Khan 2005). Write*

$$\begin{aligned} Q_{i|j} &= P_{i|j}^{-1} \\ \boldsymbol{\nu}_{i|j} &= Q_{i|j}\boldsymbol{m}_{i|j} \end{aligned}$$

and transform the observations into their 'information form', for $t = 1, \ldots, T$

$$\begin{aligned} I_t &= \Phi_t^\intercal \Sigma_\epsilon^{-1} \Phi_t, \\ i_t &= \Phi_t^\intercal \Sigma_\epsilon^{-1} \boldsymbol{z}_t. \end{aligned} \tag{4.4}$$

The prediction step now becomes

$$\begin{aligned} \boldsymbol{\nu}_{t+1|t} &= (I - J_t)M^{-1}\boldsymbol{\nu}_{t|t} \\ Q_{t+1|t} &= (I - J_t)S_t \end{aligned} \tag{4.5}$$

where $S_t = M^{-\intercal} Q_{t|t} M^{-1}$ and $J_t = S_t[S_t + \Sigma_\eta^{-1}]^{-1}$.

Updating is now as simple as adding the information-form observations;

$$\begin{aligned} \boldsymbol{\nu}_{t+1|t+1} &= \boldsymbol{\nu}_{t+1|t} + i_{t+1} \\ Q_{t+1|t+1} &= Q_{t+1|t} + I_{t+1}. \end{aligned} \tag{4.6}$$

Proof in Appendix (Section A.2.)

We can see that the information form of the observations (Equation 4.4) will always have the same dimension [2]. For our purposes, this means that `jax.lax.scan` will work after we 'informationify' the data, which can be done using `jax.tree.map`. This is implemented in the functions `information_filter` and `information_filter_indep` (for uncorrelated errors).

There are other often cited advantages to filtering in this form. It can be quicker that the traditional form in certain cases, especially when the observation dimension is bigger than the state dimension (since you solve a smaller system of equations with $[S_t + \Sigma_\eta]^{-1}$ in the process dimension instead of $[\Phi_t P_{t+1|t} \Phi_t^\intercal + \Sigma_\epsilon]^{-1}$ in the observation dimension) (Assimakis, Adam, and Douladiris 2012).

The other often mentioned advantage is the ability to use a flat prior for α_0; that is, we can set Q_0 as the zero matrix, without worrying about an infinite variance matrix. While this is indeed true, it is actually possible to do the same with the Kalman filter by doing the first step analytically, see Section A.3.

As with the Kalman filter, it is also possible to get the data likelihood in-line as well. Again, we would like to stick with things in the state dimension, so working directly with the prediction errors $\boldsymbol{e}_t$ should be avoided. Luckily, by multiplying the errors by $\Phi_t^\intercal \Sigma_\epsilon^{-1}$, we can define the 'information errors' $\boldsymbol{\iota}_t$;

$$\begin{aligned} \boldsymbol{\iota}_t = \Phi_t^\intercal \Sigma_\epsilon^{-1} \boldsymbol{e}_t &= \Phi_t^\intercal \Sigma_\epsilon^{-1} \tilde{\boldsymbol{z}}_t - \Phi_t^\intercal \Sigma_\epsilon^{-1} \Phi_t m_{t|t-1} \\ &= i_t - I_t Q_{t|t-1}^{-1} \boldsymbol{\nu}_{t|t-1}. \end{aligned}$$

[1] that is, the parameters of the Gaussian distribution in it's exponential family form

[2] that being the process dimension, previously labelled r, the number of basis functions used in the expansion of the process

The variance of this quantity is also easy to find;

$$\begin{aligned}\mathbb{V}\text{ar}[\boldsymbol{\imath}_t] &= \Phi_t^\intercal \Sigma_\epsilon^{-1}\, \mathbb{V}\text{ar}[\boldsymbol{e}_t]\Sigma_\epsilon^{-1}\Phi_t \\ &= \Phi_t^\intercal \Sigma_\epsilon^{-1}[\Phi_t P_{t|t-1}\Phi_t^\intercal + \Sigma_\epsilon]\Sigma_\epsilon^{-1}\Phi_t \\ &= \Phi_t^\intercal \Sigma_\epsilon^{-1}\Phi_t Q_{t|t-1}^{-1}\Phi_t^\intercal \Sigma_\epsilon^{-1}\Phi_t \Phi_t^\intercal \Sigma_\epsilon^{-1}\Phi_t \\ &= I_t Q_{t|t-1}^{-1} I_t^\intercal + I_t =: \Sigma_{\imath,t}.\end{aligned}$$

Noting that $\boldsymbol{\imath}$ clearly still has mean zero, this allows us once again to compute the log likelihood, this time through $\boldsymbol{\imath}$

$$\mathscr{L}(z_t \mid \boldsymbol{\theta}) = -\frac{1}{2}\sum \log\det(\Sigma_{\imath,t}(\boldsymbol{\theta})) - \frac{1}{2}\sum \boldsymbol{\imath}_t(\boldsymbol{\theta})^\intercal \Sigma_{\imath,t}(\boldsymbol{\theta})^{-1}\boldsymbol{\imath}_t(\boldsymbol{\theta}) - \frac{r}{2}\log(2 * \pi).$$

4.3 The Square-Root filters

In certain high-dimensional cases, the Kalman filter (and, indeed, the information filter) can encounter numerical stability issues. For example, in the predict step of the standard Kalman filter, note the update step for the variance matrix

$$P_{t+1|t+1} = [I - K_{t+1}\Phi_{t+1}]P_{t+1|t}.$$

Somewhat masked within this equation is two (often very small) variance matrices subtracted from eachother. While analytically, the result is still guaranteed to be positive (semi-)definite, when done in floating point arithmetic (especially in single-precision or lower), the result can often be numerically indefinite. When the variances are very low (as they often become in these Kalman filters), the eigenvalues come out very close to zero and can tick over to becoming negative erroneously. This can lead to definiteness issues with all the other variance matrices, most crucially Σ_t Equation 4.3. When this happens, computation of the likelihood likely fails (certainly when such a computation involves a Cholesky decomposition). Even if such is rare to happen with 64-bit precision, modern GPU hardware tends to be much more efficient with Single (32-bit) precision, so it may still be desirable to increase stability if it permits using a lower precision. The Square-root filter and the SVD filter are such algorithms.

4.3.1 The Square-root Kalman filter

The square-root Kalman filter has it's origins soon after the standard Kalman filter gained popularity (Kaminski, Bryson, and Schmidt 1971). Of course, computational and memory constraints necessitated stable and memory-efficient approaches, while today the standard Kalman filter (and, more recently, it's parallel counterpart, to be covered in section [TBD]) usually suffice.

As its name suggests, this variant involves carriyng through the square roots of variances [3] instead of the variances themselves. This leads to, at least in some sense, an increased precision, and we can always guarentee that, at least analytically, the square of these square roots (the variances) are positive (semi-)definite.

While the square root filter has been known for a long time (even used during NASA's Apollo program), more recently, (Tracy 2022) wrote it neatly in terms of the QR decomposition, and this is what we base the presentation on here.

The key observation used for this filter is that if we have the sum of two equations where a square root is known for both, it can be written

[3] A matrix A is said to be a 'square root' of a positive-definate matrix X if $A^\intercal A = X$. Note that these square roots are not unique, but can be 'rotated' by an arbitrary unitary matrix. The 'canonical' square root is the Cholesky factor, the unique upper (or occasionally lower) triangular square root. This can be found for arbitraty square roots by taking the QR decomposition (or RQ decomposition), which effectively computes the upper-triangular square root, R, and the unitary transformation $Q^\intercal$ necessary to get there.

$$
\begin{aligned}
X + Y &= A^\intercal A + B^\intercal B \\
&= \begin{bmatrix} A^\intercal & B^\intercal \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}
\end{aligned}
$$

Taking the QR decomposition of the vertical block yields QR, and since $(QR)^\intercal\,(QR) = R^\intercal Q^\intercal QR = R^\intercal R$, so R is a square root of $X + Y$. This motivates the following 'QR operator'

$$
\mathrm{qr}_R(A, B),
$$

as the matrix R in the QR decomposition of the block matrix

$$
\begin{bmatrix} A \\ B \end{bmatrix}.
$$

Beginning with the Cholesky decomposition of the initial variances, $P_0 = U_0^\intercal U_0$, $\Sigma_\eta = U_\eta^\intercal U_\eta$ and $\Sigma_\epsilon = U_\epsilon^\intercal U_\epsilon$ the predict step for the variance becomes

$$
U_{t+1|t} = \sqrt{P_{t+1|t}} = \mathrm{qr}_R(U_{t|t} M^\intercal, U_\eta),
$$

with the step for the means being the same as before (Equation 4.1). The prediction errors, prediction variance and Kalman gain are now

$$
\begin{aligned}
\boldsymbol{e}_{t+1} &= \tilde{\boldsymbol{z}}_t - \Phi_{t+1} \boldsymbol{m}_{t+1|t}, \\
\Sigma_{t+1} &= \Phi_{t+1} P_{t+1|t} \Phi_{t+1}^\intercal + \Sigma_\epsilon, \\
\sqrt{\Sigma_{t+1}} &= U_{e,t+1} = \mathrm{qr}_R(\Phi_{t+1} U_{t+1|t}, U_\epsilon), \\
K_{t+1} &= P_{t+1|t} \Phi_{t+1}^\intercal \Sigma_{t+1}^{-1} = U_{t+1|t}^\intercal U_{t+1|t} \Phi_{t+1}^\intercal (U_{e,t+1}^\intercal U_{e,t+1})^{-1} \\
&= (U_{e,t+1}^{-1} U_{e,t+1}^{-\intercal} \Phi_{t+1} U_{t+1|t}^\intercal U_{t+1|t})^\intercal
\end{aligned}
$$

where the last equation for the Kalman gain can easily be solved with a computationally efficient triangular solve.

Finally, the update step for the mean is simply

$$
\boldsymbol{m}_{t+1|t+1} = \boldsymbol{m}_{t|t+1} + K_{t+1} \boldsymbol{e}_{t+1}.
$$

However, for the update we use the so-called Joseph stabilised form (sometimes used in the derivation of the Kalman filter)

$$
\begin{aligned}
P_{t+1|t+1} &= \mathbb{C}\mathrm{ov}[\boldsymbol{\alpha}_t - \boldsymbol{m}_{t+1|t+1}] \\
&= \mathbb{C}\mathrm{ov}[\boldsymbol{\alpha}_t - \boldsymbol{m}_{t|t+1} - K_{t+1}(\tilde{\boldsymbol{z}}_{t+1} - \Phi_{t+1} \boldsymbol{m}_{t+1|t})] \\
&= \mathbb{C}\mathrm{ov}[\boldsymbol{\alpha}_t - \boldsymbol{m}_{t|t+1} - K_{t+1}(\Phi_{t+1} \boldsymbol{m}_{t+1} + \boldsymbol{\epsilon}_{t+1} - \Phi_{t+1} \boldsymbol{m}_{t+1|t})] \\
&= \mathbb{C}\mathrm{ov}[(I - K_{t+1}\Phi_{t+1})(\boldsymbol{\alpha}_t - \boldsymbol{m}_{t+1|t}) - \boldsymbol{\epsilon}_{t+1}] \\
&= (I - K_{t+1}\Phi_{t+1}) \mathbb{C}\mathrm{ov}[\boldsymbol{\alpha}_t - \boldsymbol{m}_{t+1|t}](I - K_{t+1}\Phi_{t+1})^\intercal + \mathbb{C}\mathrm{ov}[\boldsymbol{\epsilon}_{t+1}] \\
&= (I - K_{t+1}\Phi_{t+1}) P_{t+1|t} (I - K_{t+1}\Phi_{t+1})^\intercal + \Sigma_\epsilon
\end{aligned}
$$

which is often simplified further to Equation 4.2, but as discussed that involves negation of two square root matrices; this form is more complicated and involves more matrix computation, but guarentees that the result will be positive (semi-)definite. Furthermore, this is also in a form that allows us to easily find the square root with the QR trick;

$$U_{t+1|t+1} = \sqrt{P_{t+1|t+1}} = \mathrm{qr}_R(U_{t+1|t}(I - K_{t+1}\Phi_{t+1})^\intercal, U_\epsilon).$$

Of course, from here, we can similarily easily compute the data likelihood using $U_{e,t+1}$, using standard techniques; the multivariate normal likelihood is usually computed using the cholesky decomposition of the variance matrix anyway. The result is an algorithm which is of a higher order than the standard Kalman filter, but the stability is often worth the comprimise. Once jit-compiled, the function `sqrt_filter_indep` on a moderately sized IDEM (on a discrete GPU) on 64-bit precision [4] takes approximately 23.5ms, compared to `kalman_filter_indep` taking approximately 7.8ms, achieving similar log-likelihoods (whith some difference due to precision). However, running the code in 32-bit causes the Kalman filter likelihood computation to fail, the square-root filter succeeds at a time of 7.0ms.

4.3.2 Square-root Information filter

Very similarily, we can write the information filter using the square roots of the information matrices. We will label roots of 'information-type' matrices with R, and 'variance-type' matrices (their inverse) with U.

We now carry the data's information matrix's (Equation 4.4) square root as well, $R_t^{(I)} = \sqrt{(\Phi_t^\intercal \Sigma_\epsilon^{-1} \Phi_t)}$, with the same observation vector.

So, once again, beginning with the lower-triangular cholesky decomposition $Q_0 = R^\intercal R$, and the upper-triangular $\Sigma_\eta = U_\eta^\intercal U_\eta$ and $\Sigma_\epsilon = U_\epsilon^\intercal U_\epsilon$.

So, to predict step for the information matrix (Equation 4.5) becomes

$$\begin{aligned} Q_{t+1|t} &= (M Q_{t|t}^{-1} M^\intercal + \Sigma_\eta)^{-1} \\ &= (M R_{t|t}^{-1} R_{t|t}^{-\intercal} M^\intercal + U_\eta^\intercal U_\eta)^{-1} \\ &= \left[(M R_{t|t}^{-1}, U_\eta^\intercal) \begin{pmatrix} R_{t|t}^{-\intercal} M^\intercal \\ U_\eta \end{pmatrix} \right]^{-1} \\ R_{t+1|t}^{-1} &= \mathrm{qr}_R(R_{t|t}^{-\intercal} M^\intercal, U_\eta) \end{aligned} \tag{4.7}$$

This must now be explicitly inverted, which isn't a big problem since it is upper-triangular.

The update on the information vector is now

$$\begin{aligned} \nu_{t+1|t} &= Q_{t+1|t} M Q_{t|t}^{-1} \nu_{t|t} \\ &= R_{t+1|t}^\intercal R_{t+1|t} M R_{t|t}^{-1} R_{t|t}^{-\intercal} \nu_{t|t}, \end{aligned} \tag{4.8}$$

which can be done, as in the square-root Kalman filter's Kalman gain computation, using forward-solves.

Now the update step is

$$\begin{aligned} \nu_{t+1|t+1} &= \nu_{t+1|t} + i_{t+1} \\ Q_{t+1|t+1} &= Q_{t+1|t} + I_{t+1} \\ &= R_{t+1|t}^\intercal R_{t+1|t} + R_{t+1}^{(I)\intercal} R_{t+1}^{(I)} \\ R_{t+1|t+1} &= \mathrm{qr}_R(R_{t+1|t}, R_{t+1}^{(I)}). \end{aligned} \tag{4.9}$$

[4] which must be explicitely enabled in JAX

4.4 Smoothing

Beyond the filtering, another task is *smoothing*. That is, filters estimate $\boldsymbol{m}_{T|T}$ and $P_{T|T}$, but there is use for estimating $\boldsymbol{m}_{t|T}$ and $P_{t|T}$ for all $t = 0, \ldots, T$.

We simply work backwards from $\boldsymbol{m}_{T|T}$ and $P_{T|T}$ values using what is known as the *Rauch-Tung-Striebel (RTS) smoother*;

$$\begin{aligned} \boldsymbol{m}_{t-1|T} &= \boldsymbol{m}_{t-1|t-1} + J_{t-1}(\boldsymbol{m}_{t|T} - \boldsymbol{m}_{t|t-1}), \\ P_{t-1|T} &= P_{t-1|t-1} + J_{t-1}(P_{t|T} - P_{t|t-1})J_{t-1}^{\intercal}, \end{aligned} \tag{4.10}$$

where,

$$J_{t-1} = P_{t-1|t-1}M^{\intercal}[P_{t|t-1}]^{-1}.$$

We can clearly see, then, that it is crucial to keep the values in Equation 4.1.

We can then also compute the lag-one cross-covariance matrices $P_{t,t-1|T}$ using the *Lag-One Covariance Smoother*. This will b useful, for example, in the expectation-maximisation algorithm later. From

$$P_{T,T-1|T} = (I - K_T\Phi_T)M P_{T-1|T-1},$$

we can compute the lag-one covariances

$$P_{t,t-1|T} = P_{t|t}J_{t-1}^{\intercal} + J_t[P_{t+1,t|T} - M P_{t-1|t-1}]J_{t-1}^{\intercal} \tag{4.11}$$

These values can be used to implement the expectation-maximisation (EM) algorithm which will be introduced later.

Chapter 5

EM Algorithm (NEEDS A LOT OF WORK, PROBABLY IGNORE FOR NOW)

Instead of the marginal data likelihood, we may instead want to work with the 'full' likelihood, including the unobserved process, $l(\boldsymbol{z}(1), \dots, \boldsymbol{z}(T), \boldsymbol{Y}(1), \dots, \boldsymbol{Y}(T) \mid \boldsymbol{\theta})$, or, equivalently, $l(\boldsymbol{z}(1), \dots, \boldsymbol{z}(t), \boldsymbol{\alpha}(1), \dots, \boldsymbol{\alpha}(T) \mid \boldsymbol{\theta})$. This is difficult to maximise directly, but can be done with the EM algorithm, consisting of two steps, which can be shown to always increase the full likelihood.

Firstly, the E step is to find the function

$$\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}') = \mathbb{E}_{\boldsymbol{Z}(t) \sim p(Z \mid \boldsymbol{\alpha}(t), \boldsymbol{\theta})}[\log p_{\boldsymbol{\theta}}(Z^{(T)}, A^{(T)}) \mid Z^{(T)}], \tag{5.1}$$

where $Z^{(T)} = \{\boldsymbol{z}_t\}_{t=0,\dots,T}$, $A^{(T)} = \{\boldsymbol{\alpha}_t\}_{t=0,\dots,T}$ and $A^{(T-1)} = \{\boldsymbol{\alpha}_t\}_{t=0,\dots,T-1}$. This approximates $\log p_\theta(Z^{(T)}, A^{(T)})$.

Proposition 5.0.1. *We have [NOTE: This may well be wrong in places…]*

$$\begin{aligned}
-2\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}') &= \mathbb{E}_{Z^{(T)} \sim p(Z \mid A^{(T)}, \boldsymbol{\theta}')}[\log p_{\boldsymbol{\theta}}(Z^{(T)}, A^{(T)} \mid Z^{(T)} = z^{(T)})] \\
&\overset{c}{=} \sigma_\epsilon^2 [\sum_{t=0}^{T} \boldsymbol{z}_t^\intercal z_t - 2\Phi_t (\sum_{t=1}^{T} \boldsymbol{z}_t^\intercal \boldsymbol{m}_{t|T}) - 2(\sum_{t=0}^{T} \boldsymbol{z}_t^T) X_t \boldsymbol{\beta} \\
&\qquad + \Phi_t^\intercal (\sum_{t=0}^{T} \operatorname{tr}\{P_{t|T} - \boldsymbol{m}_{t|T} \boldsymbol{m}_{t|T}^\intercal\}) \Phi_t + 2X_t \boldsymbol{\beta} \Phi_t (\sum_{t=0}^{T} \boldsymbol{m}_{t|T}) + (\sum_{t=1}^{T} X_t^\intercal \boldsymbol{\beta}^\intercal \boldsymbol{\beta} X_t)] \\
&\quad + \operatorname{tr}\{\Sigma_\eta^{-1} [(\sum_{t=1}^{T} P_{t|T} - m_{t|T}) - 2M (\sum_{t=1}^{T} P_{t,t-1|T} - \boldsymbol{m}_{t-1,T} \boldsymbol{m}_{t|T}^\intercal) \\
&\qquad + M (\sum_{t=1}^{T} P_{t-1|T} - \boldsymbol{m}_{t-1|T} \boldsymbol{m}_{t-1|T}^\intercal) M^\intercal]\} \\
&\quad + \operatorname{tr}\{\Sigma_0^{-1} [P_{0|T} - m_{0|T} m_{0|T}^\intercal - 2\boldsymbol{m}_{0|T} \boldsymbol{m}_0 + \boldsymbol{m}_0 \boldsymbol{m}_0^\intercal]\} \\
&\quad + \log(\det(\sigma_\epsilon^{2T} \Sigma_\eta^{T+1} \Sigma_0))
\end{aligned} \tag{5.2}$$

Proof. See appendix. □

In the EM algorithm, we maximise the full likelihood by changing $\boldsymbol{\theta}$ in order to increase (Equation 5.2), which can be shown to guarantee that the Likelihood $L(\boldsymbol{\theta})$ also increases. The idea is then that repeatedly alternating between adjusting $\boldsymbol{\theta}$ to increase Equation 5.2, and then doing the filters and smoothers to obtain new values for $\boldsymbol{m}_{t|T}$, $\boldsymbol{P}_{t|T}$, and $\boldsymbol{P}_{t,t-1|T}$.

Chapter 6

Algorithm for Maximum Complete-data Likelihood estimation

Overall, our algorithm for Maximum Likelihood estimation is:

1. Set $i = 0$ and take an initial guess for the parameters we are considering, $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_i$
2. Starting from $\boldsymbol{m}_{0|0} = \boldsymbol{m}_0$, $P_{0|0} = \Sigma_0$, run the **Kalman Filter** to get $\boldsymbol{m}_{t|t}$, $P_{t|t}$, and K_t for all t Equation 4.2,
3. Starting from $\boldsymbol{m}_{T|T}$, $P_{T|T}$, run the **Kalman Smoother** to get $\boldsymbol{m}_{t|T}$, $P_{t|T}$, and J_t for all t (Equation 4.10),
4. Starting from $P_{T,T-1|T} = (I - K_n A_n) M P_{T-1|T-1}$, run the **Lag-One Smoother** to get $\boldsymbol{m}_{t,t-1|T}$ and $P_{t,t-1|T}$ for all t Equation 4.11,
5. Use the above values to construct $\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}')$ in Equation 5.2,
6. Maximise the function $\mathcal{Q}(\boldsymbol{\theta}; \boldsymbol{\theta}')$ to get a new guess $\boldsymbol{\theta}_{i+1}$, then return to step 2,
7. Stop once a certain criteria is met.

Appendix A

Appendix

A.1 Woodbury's identity

The following two sections will make heavy use of the Woodbury identity.

Lemma A.1.1 (Woodbury's Identity)**.** *We have, for conformable matrices* A, U, C, V*,*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}. \tag{A.1}$$

Additionally, we have the variant

$$(A + UCV)^{-1}UC = A^{-1}U(C^{-1} + VA^{-1}U)^{-1}. \tag{A.2}$$

Proof. We only prove (Equation A.2), since various proofs of (Equation A.1) are well known (see, for example, the Wikipedia page).

Simply multipliying (Equation A.1) by CU, (similar to Khan 2005, although there is an error in their proof)

$$\begin{aligned}(A + UCV)^{-1}UC &= A^{-1}UC - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}UC \\ &= A^{-1}UC - A^{-1}U(C^{-1} + VA^{-1}U)[(C^{-1} + VA^{-1}U)C - I] \\ &= A^{-1}U(C^{-1} + VA^{-1}U)\end{aligned}$$

as needed. □ □

A.2 Proof of Theorem 4.2.1

Proof. Firstly, for the prediction step, using $S_t = M^{-\intercal}Q_{t|t}M^{-1}$ and $J_t = S_t(\Sigma_\eta^{-1} + S_t)^{-1}$ and the identities Equation A.1 and Equation A.2,

$$
\begin{aligned}
Q_{t+1|t} = P_{t+1|t}^{-1} &= (MQ_{t|t}^{-1}M^\intercal + \Sigma_\eta)^{-1} \\
&= S_t - J_t S_t = (I - J_t)S_t,
\end{aligned}
$$

where we used $A = MQ_{t|t}^{-1}M^\intercal$, $C = \Sigma_\eta$ and $U = C = I$ in Equation A.1. Thurthermore,

$$
\begin{aligned}
\boldsymbol{\nu}_{t+1|t} &= Q_{t+1|t}\boldsymbol{m}_{t+1|t} \\
&= Q_{t+1|t}MQ_{t|t}^{-1}\boldsymbol{\nu}_{t|t} = Q_{t+1|t}(MQ_{t|t}^{-1})\boldsymbol{\nu}_{t|t} \\
&= (I - J_t)M^{-\intercal}Q_{t|t}M^{-1}(MQ_{t|t}^{-1})\boldsymbol{\nu}_{t|t} \\
&= (I - J_t)M^{-\intercal}\boldsymbol{\nu}_{t|t}.
\end{aligned}
$$

For the update step,

$$
\begin{aligned}
Q_{t+1|t+1} &= P_{t+1|t+1}^{-1} \\
&= (Q_{t+1}^{-1} - Q_{t+1|t}^{-1}\Phi_{t+1}^\intercal[\Phi_{t+1}\Sigma_\epsilon\Phi_{t+1}^\intercal + \Sigma_\epsilon]^{-1}\Phi_{t+1}Q_{t+1|t}^{-1})^{-1} \\
&= ((Q_{t+1|t} + \Phi_{t+1}^\intercal\Sigma_\epsilon^{-1}\Phi_{t+1})^{-1})^{-1} = Q_{t+1|t} + \Phi_{t+1}^\intercal\Sigma_\epsilon^{-1}\Phi_{t+1} \\
&= Q_{t+1|t} + I_{t+1}.
\end{aligned}
$$

Then, writing $\boldsymbol{m}_{t+1|t+1}$ in terms of $Q_{t+1|t}$ and $\boldsymbol{\nu}_{t+1|t}$

$$
\begin{aligned}
\boldsymbol{m}_{t+1|t+1} &= Q_{t+1|t}^{-1}\boldsymbol{\nu}_{t+1|t} - Q_{t+1|t}^{-1}\Phi_{t+1}^\intercal[\Phi_{t+1}Q_{t+1|t}^{-1}\Phi_{t+1}^\intercal + \Sigma_\epsilon]^{-1}[\tilde{\boldsymbol{z}}_{t+1} - \Phi_{t+1}Q_{t+1|t}^{-1}\boldsymbol{\nu}_{t+1t}] \\
&= (Q_{t+1|t}^{-1} - Q_{t+1|t}^{-1}\Phi_{t+1}^\intercal[\Phi_{t+1}Q_{t+1|t}^{-1}\Phi_{t+1}^\intercal + \Sigma_\epsilon]^{-1}\Phi_{t+1}Q_{t+1|t}^{-1})\boldsymbol{\nu}_{t+1|t} \\
&\quad + Q_{t+1|t}^{-1}\Phi_{t+1}^\intercal[\Phi_{t+1}Q_{t+1|t}^{-1}\Phi_{t+1}^\intercal + \Sigma_\epsilon]^{-1}\tilde{\boldsymbol{z}}_{t+1} \\
&= [Q_{t+1|t} + I_{t+1}]^{-1}\boldsymbol{\nu}_{t+1|t} \\
&\quad + [Q_{t+1|t} + I_{t+1}]^{-1}\Phi_{t+1}\Sigma_\epsilon^{-1}\tilde{\boldsymbol{z}}_{t+1},
\end{aligned}
$$

and now noting that $\boldsymbol{\nu}_{t+1|t+1} = (Q_{t+1|t} + I_{t+1})\boldsymbol{m}_{t+1|t+1}$, we complete the proof. □ □

A.3 Truly Vague Prior with the Kalman Filter

It has been stated before that one of the large advantages of the information filter is the ability to use a completely vague prior $Q_0 = 0$. While this is true, it is actually possible to do this in the Kalman filter by 'skipping' the first step (contrary to some sources, such as the Wikipedia page as of January 2025).

Theorem A.3.1. *In the Kalman Filter (Section 4.1), if we allow $P_0^{-1} = 0$, effectively setting infinite variance, and assuming the propagator matrix M is invertible, we have*

$$
\begin{aligned}
\boldsymbol{m}_{1|1} &= (\Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1}\Phi_1\Sigma_\epsilon^{-1}\tilde{\boldsymbol{z}}_1, \\
P_{1|1} &= (\Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1}.
\end{aligned}
\tag{A.3}
$$

Therefore, starting with these values then continuing the filter as normal, we can perform the Kalman filter with 'infinite' prior variance.

[NOTE: The requirement that M be invertible should be droppable, see the proof below]

Proof. Unsurprisingly, the proof is effectively equivalent to proving the information filter and setting $Q_0 = P_0^{-1} = 0$.

For the first predict step (Equation 4.1),

$$\begin{aligned} \boldsymbol{m}_{1|0} &= M\boldsymbol{m}_0, \\ P_{1|0} &= MP_0M^\intercal + \Sigma_\eta. \end{aligned}$$

By (Equation A.1),

$$\begin{aligned} P_{1|0}^{-1} &= \Sigma_\eta^{-1} - \Sigma_\eta^{-1}M(P_0^{-1} + M^\intercal\Sigma_\eta^{-1}M)^{-1}M^\intercal\Sigma_\eta^{-1} \\ &= \Sigma_\eta^{-1} - \Sigma_\eta^{-1}M(M^\intercal\Sigma_\eta^{-1}M)^{-1}M^\intercal\Sigma_\eta^{-1} \\ &= \Sigma_\eta^{-1} - \Sigma_\eta^{-1} = 0. \end{aligned}$$

So, moving to the update step (Equation 4.2),

$$\boldsymbol{m}_{1|1} = M\boldsymbol{m}_0 + P_{1|0}\Phi_1[\Phi_1P_{1|0}\Phi_1^\intercal + \Sigma_\epsilon]^{-1}(\tilde{\boldsymbol{z}}_1 - \Phi M\boldsymbol{m}_0).$$

Applying (Equation A.2) with $A = P_{1|0}^{-1}, U = \Phi_1, V = \Phi_1^\intercal, C = \Sigma_\epsilon^{-1}$,

$$\begin{aligned} \boldsymbol{m}_{1|1} &= M\boldsymbol{m}_0 + (P_{1|0}^{-1} + \Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1}\Phi_1^\intercal\Sigma_\epsilon^{-1}(\tilde{\boldsymbol{z}}_1 - \Phi_1M\boldsymbol{m}_0) \\ &= M\boldsymbol{m}_0 + (\Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1}\Phi_1^\intercal\Sigma_\epsilon^{-1}\tilde{\boldsymbol{z}}_1 - (\Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1}\Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1M\boldsymbol{m}_0 \\ &= (\Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1}\Phi_1^\intercal\Sigma_\epsilon^{-1}\tilde{\boldsymbol{z}}_1. \end{aligned}$$

For the variance, we apply the (Equation A.1) with $A = P_{1|0}^{-1}, U = \Phi_1^\intercal, V = \Phi_1, C = \Sigma_\epsilon^{-1}$,

$$\begin{aligned} P_{1|1} &= (I - P_{1|0}\Phi_1^\intercal[\Sigma_\epsilon + \Phi_1^\intercal P_{1|0}\Phi_1]^{-1}\Phi_1)P_{1|0} \\ &= (P_{1|0}^{-1} + \Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1} \\ &= (\Phi_1^\intercal\Sigma_\epsilon^{-1}\Phi_1)^{-1}, \end{aligned}$$

as needed. □ □

It is worth noting that (Equation A.3) seems to make a lot of sense; namely, we expect the estimate for $\boldsymbol{m}_0$ to look like a correlated least squares-type estimator like this.

Assimakis, Nicholas, Maria Adam, and Anargyros Douladiris. 2012. "Information Filter and Kalman Filter Comparison: Selection of the Faster Filter." In *Information Engineering*, 2:1–5. 1.

Cressie, Noel, and Christopher K Wikle. 2015. *Statistics for Spatio-Temporal Data*. John Wiley & Sons.

Dewar, Michael, Kenneth Scerri, and Visakan Kadirkamanathan. 2008. "Data-Driven Spatio-Temporal Modeling Using the Integro-Difference Equation." *IEEE Transactions on Signal Processing* 57 (1): 83–91.

Kaminski, Paul, Arthur Bryson, and Stanley Schmidt. 1971. "Discrete Square Root Filtering: A Survey of Current Techniques." *IEEE Transactions on Automatic Control* 16 (6): 727–36.

Khan, Mohammad Emtiyaz. 2005. "Matrix Inversion Lemma and Information Filter." *Honeywell Techonology Solutions Lab, Bangalore, India*.

Liu, Xiao, Kyongmin Yeo, and Siyuan Lu. 2022. "Statistical Modeling for Spatio-Temporal Data from Stochastic Convection-Diffusion Processes." *Journal of the American Statistical Association* 117 (539): 1482–99.

Särkkä, Simo, and Ángel F García-Fernández. 2020. "Temporal Parallelization of Bayesian Smoothers." *IEEE Transactions on Automatic Control* 66 (1): 299–306.

Shumway, Robert H, David S Stoffer, and David S Stoffer. 2000. *Time Series Analysis and Its Applications*. Vol. 3. Springer.

Tracy, Kevin. 2022. "A Square-Root Kalman Filter Using Only QR Decompositions." *arXiv Preprint arXiv:2208.06452*.

Wikle, Christopher K, and Noel Cressie. 1999. "A Dimension-Reduced Approach to Space-Time Kalman Filtering." *Biometrika* 86 (4): 815–29.

Wikle, Christopher K, Andrew Zammit-Mangion, and Noel Cressie. 2019. *Spatio-Temporal Statistics with r*. CRC Press.