A. Background on Message Passing for Bayesian Inference

In this section, we provide a formal description of message-passing algorithms for Bayesian inference. Then, in Appendix B, we give full details of the particular message-passing algorithm we use in this work.

Given a factorisation of the joint probability model $g(\boldsymbol{f}) = g(f_1, \ldots, f_n)$ of the form

$$g(\boldsymbol{f}) \propto \prod_{i=1}^{n} \phi_i(f_i) \prod_{j=1}^{n} \phi_{ij}(f_i, f_j), \tag{A.1}$$

for nodewise and pairwise factors ϕ_i and $\phi_{i,j}$, this induces a sparse graphical structure on the variables $\{f_i\}_{i=1}^n$, where two nodes f_i and f_j are connected if and only if the pairwise factor $\phi_{i,j}$ is non-constant. We can further augment this graph by including the non-constant factors $\{\phi_i\}_{i\in V}$ and $\{\phi_{ij}\}_{(i,j)\in E}$ as additional nodes, and joining the pairs of nodes (ϕ_i, f_i), (ϕ_{ij}, f_i), and (ϕ_{ij}, f_j) by edges. This augmented structure is referred to as the *factor graph representation* of the probabilistic model (2), which is a bipartite graph with one side of nodes being the set of variables f_i and the other side being the set of factors ϕ_i, ϕ_{ij}.

A.1. Loopy Belief Propagation

Given a factor graph representation of the model, one can approximate the marginal probabilities $\{g(f_i)\}_{i=1}^n$ efficiently by a message-passing algorithm, referred to as *loopy belief propagation*. This begins by defining a set of so-called *messages* $m^t_{f_i\to\phi_\alpha}(f_i)$, $m^t_{\phi_\alpha\to f_i}(f_i)$ between each of the variable and factor nodes. The messages are initialised to $m^0_{f_i\to\phi_\alpha}(f_i) = m^0_{\phi_\alpha\to f_i}(f_i) = 1$ and updated by iterating the following steps:

$$m^{t+1}_{f_i\to\phi_\alpha}(f_i) = \phi_i(f_i) \prod_{\substack{\phi_\beta\sim f_i\\ \phi_\beta\neq\phi_\alpha}} m^t_{\phi_\beta\to f_i}(f_i), \quad i = 1, \ldots, n, \tag{A.2}$$

$$m^{t+1}_{\phi_\alpha\to f_i}(f_i) = \int_{\mathbb{R}} \phi_{ij}(f_i, f_j) m^t_{f_j\to\phi_\alpha}(f_j)\,\mathrm{d}f_j, \quad i = 1, \ldots, n. \tag{A.3}$$

The updates (A.2)–(A.3) can be done either in parallel, serially, or following more specific scheduling rules (Elidan et al. 2006; Gonzalez et al. 2009; Van der Merwe et al. 2019) – generally, there are no restrictions on the order in which we pass the messages between nodes. Upon convergence, say at iteration $t = T$, we can then estimate the marginal distribution at each variable $g(f_i)$, by taking

$$g(f_i) \approx \frac{1}{Z} \prod_{\phi_\alpha\sim f_i} m^T_{\phi_\alpha\to f_i}(f_i), \quad \text{where} \quad Z = \prod_{\phi_\alpha\sim f_i} \int_{\mathbb{R}} m^T_{\phi_\alpha\to f_i}(f_i)\mathrm{d}f_i, \tag{A.4}$$

for $i = 1, \ldots, n$. It is well-known that, when the graph is tree-structured, convergence is guaranteed and the message-passing algorithm yields the exact marginals. However, if the graph contains loops, then convergence is not guaranteed and moreover the obtained marginals may not be exact. Various techniques have been proposed to improve this. In particular, one can make *fractional updates* to the messages (Wiegerinck and Heskes 2002), which we express in the form

$$m_{f_i\to\phi_\alpha}(f_i)^{\frac{1}{c}} = \phi_i(x_i) \prod_{\substack{\phi_\beta\sim f_i\\ \phi_\beta\neq\phi_\alpha}} m_{\phi_\beta\to f_i}(f_i) m_{\phi_\alpha\to f_i}(f_i)^{1-\frac{1}{c}}, \tag{A.5}$$

$$m_{\phi_\alpha\to f_i}(f_i)^{\frac{1}{c}} = \int_{\mathbb{R}} \phi_{ij}(f_i, f_j)^{\frac{1}{c}} m_{f_j\to\phi_\alpha}(f_j)^{\frac{1}{c}}\,\mathrm{d}f_j, \tag{A.6}$$

for constants $c_\alpha \in \mathbb{N}$. In this work, we use an extension of this by Ruozzi and Tatikonda (2013), where c_α is also allowed to take any real values, including negative ones. A benefit of this re-weighting scheme is that one can guarantee convergence even in the loopy setting by taking c_α large enough, or by setting them to be negative (Ruozzi and Tatikonda 2013). While the message updates in the re-weighted scheme are modified according to (A.5)–(A.6), the marginal computation is still performed in the same way as standard message passing (A.4).

Next, we discuss the details on how to compute the messages (A.2)–(A.3) and (A.5)–(A.6) more concretely in the special but important case of joint Gaussian probability models.

A.2. Gaussian Setting

For Gaussian probability models, which we assume to have zero mean for the time being, we can write down the joint probability explicitly as

$$p(\boldsymbol{f}) \propto \exp\left(-\frac{1}{2}\boldsymbol{f}^\top \boldsymbol{P}\boldsymbol{f}\right) = \exp\left(-\sum_{i\in V}\frac{1}{2}[\boldsymbol{P}]_{ii}f_i^2 - \sum_{(i,j)\in E}[\boldsymbol{P}]_{ij}f_if_j\right), \tag{A.7}$$

for some precision matrix $\boldsymbol{P}$ and (V,E) is the graph induced by the sparsity pattern of $\boldsymbol{P}$, described earlier. This naturally gives us factors of the form

$$\phi_i(f_i) := \exp\left(-\frac{1}{2}[\boldsymbol{P}]_{ii}f_i^2\right), \quad \text{and} \quad \phi_{ij}(f_i,f_j) := \exp\left(-[\boldsymbol{P}]_{ij}f_if_j\right), \tag{A.8}$$

which we notice are also Gaussian up to a normalisation constant. Since the factors are all proportional to Gaussian densities, and the family of Gaussians are closed under products and integrals, the messages and states computed by (A.2)–(A.4) are also expected to be Gaussians. For simplicity, let us parameterise the variable-to-factor messages (A.2) as

$$m_{f_i\to\phi_{ij}}(f_i) = \exp\left(-\frac{1}{2}\alpha_{ij}f_i^2 - \beta_{ij}f_i\right), \tag{A.9}$$

and the factor-to-variable messages (A.3) as

$$m_{\phi_{ij}\to f_i}(f_i) = \exp\left(-\frac{1}{2}a_{ij}f_i^2 - b_{ij}f_i\right). \tag{A.10}$$

We then compute the update rules for the two types of messages as follows.

Variable-to-factor message. From (A.2), we get

$$m_{f_i\to\phi_{ij}}(f_i) = \phi_i(f_i)\prod_{\substack{k\sim i\\k\neq j}} m_{\phi_{ik}\to f_i}(f_i) \propto \exp\left(-\frac{1}{2}\left(P_{ii} + \sum_{\substack{k\sim i\\k\neq j}} a_{ik}\right)f_i^2 - \sum_{\substack{k\sim i\\k\neq j}} b_{ik}f_i\right), \tag{A.11}$$

giving us the following updates on the parameters α_{ij} and β_{ij}

$$\alpha_{ij} \leftarrow P_{ii} + \sum_{\substack{k\sim i\\k\neq j}} a_{ik}, \qquad \beta_{ij} \leftarrow \sum_{\substack{k\sim i\\k\neq j}} b_{ik}. \tag{A.12}$$

Factor-to-variable message. Now using (A.3), we have

$$\begin{aligned}
m_{\phi_{ij}\to f_i}(f_i) &= \int \phi_{ij}(f_i, f_j)\mu_{f_j\to\phi_{ij}}(f_j)\mathrm{d}f_j && \text{(A.13)}\\
&\propto \int \exp\left(-\frac{1}{2}\alpha_{ij}f_j^2 - \beta_{ij}f_j - P_{ij}f_if_j\right)\mathrm{d}f_j && \text{(A.14)}\\
&\propto \int \exp\left(-\frac{1}{2}\alpha_{ij}\left(f_j - \frac{\beta_{ij}}{\alpha_{ij}}\right)^2 - P_{ij}f_if_j\right)\mathrm{d}f_j && \text{(A.15)}\\
&= \int \exp\left(-\frac{1}{2}\alpha_{ij}\left(f_j - \frac{\beta_{ij}}{\alpha_{ij}}\right)^2 - P_{ij}f_i\left(f_j - \frac{\beta_{ij}}{\alpha_{ij}}\right) - \frac{\beta_{ij}P_{ij}}{\alpha_{ij}}f_i\right)\mathrm{d}f_j && \text{(A.16)}\\
&\propto e^{-\frac{\beta_{ij}P_{ij}}{\alpha_{ij}}f_i}\int \exp\left(-\frac{1}{2}\begin{pmatrix} f_i \\ f_j - \frac{\beta_{ij}}{\alpha_{ij}}\end{pmatrix}^\top \begin{pmatrix} 0 & P_{ij} \\ P_{ij} & \alpha_{ij}\end{pmatrix}\begin{pmatrix} f_i \\ f_j - \frac{\beta_{ij}}{\alpha_{ij}}\end{pmatrix}\right)\mathrm{d}f_j && \text{(A.17)}\\
&= e^{-\frac{\beta_{ij}P_{ij}}{\alpha_{ij}}f_i}\int \exp\left(-\frac{1}{2}\begin{pmatrix} f_i \\ f_j - \frac{\beta_{ij}}{\alpha_{ij}}\end{pmatrix}^\top \begin{pmatrix} -\alpha_{ij}P_{ij}^{-2} & P_{ij}^{-1} \\ P_{ij}^{-1} & 0\end{pmatrix}^{-1}\begin{pmatrix} f_i \\ f_j - \frac{\beta_{ij}}{\alpha_{ij}}\end{pmatrix}\right)\mathrm{d}f_j && \text{(A.18)}\\
&\propto \exp\left(-\frac{\beta_{ij}P_{ij}}{\alpha_{ij}}f_i + \frac{1}{2}\frac{P_{ij}^2}{\alpha_{ij}}f_i^2\right). && \text{(A.19)}
\end{aligned}$$

where we used the Gaussian marginalisation rule to yield the last line. Thus, we have the following update rule on the parameters a_{ij} and b_{ij} determining the factor-to-variable messages

$$a_{ij} \leftarrow -\frac{P_{ij}^2}{\alpha_{ij}}, \qquad b_{ij} \leftarrow \frac{\beta_{ij}P_{ij}}{\alpha_{ij}}. \tag{A.20}$$

In the case of Gaussian models with non-zero means, we have the following slight modification of the model

$$p(\boldsymbol{f}) \propto \exp\left(-\sum_{i\in V}\left(\frac{1}{2}P_{ii}f_i^2 - h_if_i\right) - \sum_{(i,j)\in E}P_{ij}f_if_j\right), \tag{A.21}$$

for some vector $\boldsymbol{h} \in \mathbb{R}^N$, giving us factors of the form

$$\phi_i(f_i) := \exp\left(-\frac{1}{2}P_{ii}f_i^2 + h_if_i\right), \quad \text{and} \quad \phi_{ij}(f_i, f_j) := \exp\left(-P_{ij}f_if_j\right). \tag{A.22}$$

This only changes the variable-to-factor message update rule (A.12) to

$$\alpha_{ij} \leftarrow P_{ii} + \sum_{\substack{k\sim i\\ k\neq j}} a_{ik}, \qquad \beta_{ij} \leftarrow -h_i + \sum_{\substack{k\sim i\\ k\neq j}} b_{ik}. \tag{A.23}$$

The factor-to-variable message update rules (A.20) remain unchanged.

A.3. Re-weighted message updates

Using the re-weighted scheme (A.5)–(A.6) of Wiegerinck and Heskes 2002, if we now parameterise the *fractional messages* by

$$m_{f_i \to \phi_{ij}}(f_i)^{\frac{1}{c}} = \exp\left(-\frac{1}{2}\alpha_{ij} f_i^2 - \beta_{ij} f_i\right), \quad m_{\phi_{ij} \to f_i}(f_i)^{\frac{1}{c}} = \exp\left(-\frac{1}{2}a_{ij} f_i^2 - b_{ij} f_i\right), \tag{A.24}$$

then the update rules on its coefficients change as

$$\alpha_{ij} \leftarrow P_{ii} + c \sum_{\substack{k \sim i \\ k \neq j}} a_{ik} + (c-1) a_{ij}, \tag{A.25}$$

$$\beta_{ij} \leftarrow -h_i + c \sum_{\substack{k \sim i \\ k \neq j}} b_{ik} + (c-1) b_{ij}, \tag{A.26}$$

$$a_{ij} \leftarrow -\frac{(P_{ij}/c)^2}{\alpha_{ij}}, \tag{A.27}$$

$$b_{ij} \leftarrow \frac{\beta_{ij}(P_{ij}/c)}{\alpha_{ij}}. \tag{A.28}$$

This can be checked by direct computation. Intuitively, for $c \in \mathbb{N}$, the update rule (A.5)–(A.6) can be understood as updating a fraction of the message each time, where each update is allowed to use information from the remaining fraction of its own message. From a variational inference perspective, this can also be understood in terms of minimising the α-divergence between the variational and true distributions with $\alpha = 2/c - 1$ (Wiegerinck and Heskes 2002; Minka 2004).

The marginal distribution at f_i is then $p(f_i) = \mathcal{N}(f_i | \mu_i, \sigma_i^2)$, where

$$\mu_i = s_i / \sigma_i^{-2}, \tag{A.29}$$

$$s_i = h_i + c \sum_{k \sim i} b_{ki}, \quad \text{and} \tag{A.30}$$

$$\sigma_i^{-2} = P_{ii} + c \sum_{k \sim i} a_{ki}. \tag{A.31}$$

B. Complete Description of Our Message-Passing Algorithm

Algorithm 2 gives full details of the message-passing algorithm we use in this work. This is an extension of the algorithm introduced by Ruozzi and Tatikonda (2013) (in turn a generalisation of Wiegerinck and Heskes (2002)), with the addition of multigrid, damping, and early stopping

Algorithm 2 Re-weighted message passing with multigrid, damping, and early stopping

```
procedure MULTIGRID MP(levels)
    for level in levels do
        if first level then
            $m_{ij} = (0, 10^{-8})$ for $i, j \in$ level
        else
            $m_{ij}$ = UPSCALE MESSAGES($m_{ij}$, previous level, level)
        end if
        $m_{ij}$ = ITERATE MP UNTIL CONVERGENCE(level, $m_{ij}$)
    end for
    return $p(f_i)$ = COMPUTE MARGINAL($c, \{m_{ki} : f_k \sim f_i\}$) for all $i \in$ last level
end procedure

procedure ITERATE MP UNTIL CONVERGENCE(level, $m_{ij}^{t=0}$)
    $\{f_i\}_{i=1}^{p}, \{\phi_i\}_{i=1}^{p}, \{\phi_{ij}\}_{i,j=1}^{p}$ = create factor graph at level    ▷ $p$ = number of grid points at level
    for $t \in \{1, \dots T\}$ do
        for $f_i$ in the graph do
            for $f_j \in \{f_j \sim f_i\}$ do
                $m'$ = COMPUTE OUTGOING MESSAGE($c, \phi_i, \{(\phi_{ki}, m_{ki}^{t-1}) : \{f_k \sim f_i\} \backslash f_j\}$)
                $m_{ij}^{t+1} = (1 - \eta) m_{ij}^{t} + \eta m'$
            end for
        end for
        if EARLY STOP($\{m_{ij}^1, m_{ij}^2, m_{ij}^{t-1}, m_{ij}^{t}\}_{\text{all } i,j}$) then break
    end for
    return $\{m_{ij}^T\}_{\text{all } i,j}$
end procedure
```

The functions that the algorithm depends on are defined as follows:

- COMPUTE OUTGOING MESSAGE($c, \phi_i, \{(\phi_{ki}, m_{ki}^{t-1}) : \{f_k \sim f_i\} \backslash f_j\}$) = (a'_{ij}, b'_{ij}) where a'_{ij} and b'_{ij} are defined by Equations (A.27) and (A.28) respectively in Appendix A.
- COMPUTE MARGINAL($c, \{m_{ki} : f_k \sim f_i\}$) is given by Equation (A.29) – (A.31).
- We stop iterating if the following inequality holds:

$$\text{EARLY STOP}(\{m_{ij}^1, m_{ij}^2, m_{ij}^{t-1}, m_{ij}^{t}\}_{\text{all } i,j}) = \text{mean}(\text{abs}(m_{ij}^t - m_{ij}^{t-1})) < \tau\, \text{mean}(\text{abs}(m_{ij}^1 - m_{ij}^2)),$$

  where the mean is taken over all i, j, and $\tau = 0.001$ based on Figure 6.
- UPSCALE MESSAGES: The edge values on the finer grid are initialised to the edge values of the preceding coarser grid. This is equivalent to initialising the marginals of the finer grid to those of the coarser grid, but avoids explicitly calculating the marginals.

C. Details on the Finite Difference Discretisation

In this section, we provide details on the discretisation of the SPDE (3), from which we derive our graphical model to which message passing is then applied. We only consider finite difference discretisation. However, it would also be possible to use finite elements.

C.1. Discretisation of the Differential Operator

First, consider the operator $\mathcal{L} := (\kappa^2 - \Delta)^{\alpha/2}$ defining the LHS of (3). This is a differential operator provided $\alpha/2 \in \mathbb{N}$. For simplicity, consider the system in one spatial dimension and $\alpha/2 = 1$. Taking a 1D grid with equal spacing $\Delta x = h$, a finite difference discretisation of the LHS of (3) reads

$$(\kappa^2 - \Delta) f(x_i) \approx \kappa^2 f_i - \frac{f_{i+1} + f_{i-1} - 2 f_i}{h^2}, \tag{A.32}$$

where $\{x_i\}_{i=0}^N$ are the grid points and we used the shorthand notation $f_i := f(x_i)$. This gives us a finite dimensional approximation $(\kappa^2 - \Delta) f \approx \boldsymbol{L}\boldsymbol{f}$, where $\boldsymbol{f} = (f_0, \dots, f_N)^\top$ and the matrix $\boldsymbol{L}$ has entries

$$[\boldsymbol{L}]_{ij} = \begin{cases} \kappa^2 + 2/h^2, & \text{if} \quad i = j, \\ -1/h^2, & \text{if} \quad |i - j| = 1, \\ 0, & \text{otherwise} \end{cases} \tag{A.33}$$

for $i, j = 1, \dots, N-1$, which is sparse and banded, with bandwidth equal to one. On the boundaries of the domain, $i, j \in \{0, N\}$, we impose the Dirichlet condition, which sets $f_0 = f_N = 0$, and therefore we can exclude them from the vector $\boldsymbol{f}$, resulting in a $(N-2)$-dimensional linear system. We may also consider the periodic boundary condition, which sets $f_{-1} = f_N$ and $f_{N+1} = f_0$, when such terms (called ghost points) appear in equation (A.32).

On a 2D $N_x \times N_y$ grid, one can apply a similar discretisation to get the approximation $\mathcal{L}f := (\kappa^2 - \Delta) f \approx \boldsymbol{L}\boldsymbol{f}$, where now, $\boldsymbol{f}$ is a vector in $\mathbb{R}^{N_x N_y}$ and $\boldsymbol{L} \in \mathbb{R}^{N_x N_y \times N_x N_y}$. We can also obtain higher-order powers of the operator $\mathcal{L}$ by taking $\mathcal{L}^n f \approx \boldsymbol{L}^n \boldsymbol{f}$. In our experiments, we use the `findiff` package (Baer 2018) to perform the discretisation of the operator $\mathcal{L}$.

C.2. White Noise Discretisation

For the Gaussian white-noise term $\mathcal{W}$ in (3) (assuming that we are in a compact domain $C \subset \mathbb{R}^2$), we claim that this can be approximated by a stochastic process $\mathcal{W}^N : C \to \mathbb{R}$ of the form

$$\mathcal{W}^N(x) = \sum_{i=1}^N \frac{z_i}{\sqrt{\Delta x \Delta y}} \mathbf{1}_{C_i}(x), \tag{A.34}$$

where

$$z^N = (z_1, \dots, z_N) \sim \mathcal{N}(0, I_N) \tag{A.35}$$

and $\{C_i\}_{i=1}^N$ denotes a finite-difference discretised rectangular cell in C whose volume $\Delta x \Delta y$ vanishes as $N \to \infty$. First, we formally define the spatial white-noise process as follows.

Definition C.1 (Lototsky and Rozovsky 2017)**.** Given a probability triple $(\Omega, \mathcal{F}, \mathbb{P})$, a random element $\mathcal{W} : \Omega \to L^2(C, \mathbb{R})^*$ (the space of continuous functionals on $L^2(C, \mathbb{R})$) is called a (zero-mean) Gaussian white-noise process in $L^2(C, \mathbb{R})$ if it satisfies the following properties:

1. For every $f \in L^2(C, \mathbb{R})$, we have $\mathcal{W}f = 0$.
2. For every $f, g \in L^2(C, \mathbb{R})$, we have $\mathbb{E}[\mathcal{W}f, \mathcal{W}g] = \langle f, g \rangle_{L^2}$.

To justify that the process $\mathcal{W}^N$ approximates $\mathcal{W}$, for every $f \in L^2(C,\mathbb{R})$, we have

$$\left\langle \mathcal{W}^N, f \right\rangle_{L^2} = \sum_{i=1}^{N} \frac{z_i}{\sqrt{\Delta x \Delta y}} \int\int_{C_i} f(x)\mathrm{d}x\mathrm{d}y. \tag{A.36}$$

Now for small $|C_i| = \Delta x \Delta y$, we have the approximation

$$\int\int_{C_i} f(x)\mathrm{d}x\mathrm{d}y \approx f(x_i)\Delta x \Delta y, \tag{A.37}$$

for some arbitrary $x_i \in C_i$ (e.g. the central point of C_i). Thus, we have the approximation

$$\left\langle \mathcal{W}^N, f \right\rangle_{L^2} \approx \sum_{i=1}^{N} \frac{z_i f(x_i)}{\sqrt{\Delta x \Delta y}} \Delta x \Delta y = \sum_{i=1}^{N} z_i f(x_i)\sqrt{\Delta x \Delta y} \tag{A.38}$$

and we see that the random variable $\left\langle \mathcal{W}^N, h \right\rangle_{L^2}$ is Gaussian with moments

$$\mathbb{E}\left[\left\langle \mathcal{W}^N, f \right\rangle_{L^2}\right] = \sum_{i=1}^{N} \underbrace{\mathbb{E}[z_i]}_{=0} f(x_i)\sqrt{\Delta x \Delta y} = 0 \tag{A.39}$$

$$\mathbb{E}\left[\left\langle \mathcal{W}^N, f \right\rangle_{L^2} \left\langle \mathcal{W}^N, g \right\rangle_{L^2}\right] = \sum_{i=1}^{N}\sum_{j=1}^{N} \underbrace{\mathbb{E}[z_i z_j]}_{=\delta_{ij}} f(x_i)g(x_j)\Delta x \Delta y = \sum_{i=1}^{N} f(x_i)g(x_i)\Delta x \Delta y. \tag{A.40}$$

Taking $N \to \infty$, these converge as

$$\mathbb{E}\left[\left\langle \mathcal{W}^N, f \right\rangle_{L^2}\right] \to 0 \tag{A.41}$$

$$\mathbb{E}\left[\left\langle \mathcal{W}^N, f \right\rangle_{L^2} \left\langle \mathcal{W}^N, g \right\rangle_{L^2}\right] = \sum_{i=1}^{N} f(x_i)g(x_i)\Delta x \Delta y \to \int_C f(x)g(x)\mathrm{d}x\mathrm{d}y = \langle f, g \rangle_{L^2}. \tag{A.42}$$

Note that the latter convergence follows from the definition of Riemann integration. Thus, the moments of $\left\langle \mathcal{W}^N, f \right\rangle_{L^2}$ converge to the moments of $\mathcal{W}f$ as $N \to \infty$ and since $f, g \in L^2(C,\mathbb{R})$ were chosen arbitrarily, we have the convergence in law

$$\mathcal{W}^N \to \mathcal{W}. \tag{A.43}$$

Putting this together, we find a discretised representation of (3) in the form

$$\boldsymbol{L}\boldsymbol{f} = \frac{1}{\sqrt{\Delta x \Delta y}} \boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(0, I). \tag{A.44}$$

In practice, we wish to have control over the marginal variances of the process $\boldsymbol{f}$ that can be tuned on the data. To achieve this, we modify the expression slightly as

$$\boldsymbol{L}\boldsymbol{f} = \sqrt{\frac{\sigma^2 q}{\Delta x \Delta y}} \boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(0, I). \tag{A.45}$$

where q is a constant that takes the form

$$q := \frac{(4\pi)^{d/2}\kappa^{2\nu}\Gamma(\nu + d/2)}{\Gamma(\nu)}, \tag{A.46}$$

This allows $\mathbb{E}[f_i^2] = \sigma^2$ for all $i = 1, \ldots, N$ and we can subsequently treat σ as a tunable parameter. We note that expression (A.46) can be obtained from the limit $q^{-1} = \lim_{\boldsymbol{x}' \to \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{x}')/\sigma^2$, where $k(\boldsymbol{x}, \boldsymbol{x}')$ is the Matérn kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{\ell}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{\ell}\right). \tag{A.47}$$

Here, K_ν is the modified Bessel function of the second kind, Γ is the Gamma function, and the hyper-parameters ν, σ^2 and ℓ govern the smoothness, variance and lengthscale of the corresponding process respectively.

C.3. *Extension to Spatiotemporal Systems*

Beyond the spatial setting considered here, we can also construct GMRF representations of linear spatiotemporal SPDEs, such as the 1D stochastic heat equation

$$\frac{\partial u}{\partial t} = \nu \Delta u + \sigma \mathcal{W}, \tag{A.48}$$

for some coefficients $\nu, \sigma > 0$. In general, we can describe a first-order-in-time discretisation of a temporally evolving system in the form

$$\boldsymbol{A}\boldsymbol{u}^{n+1} = \boldsymbol{B}\boldsymbol{u}^n + \frac{\sigma}{\sqrt{\Delta t \Delta x}} z^{n+1}, \quad z_{n+1} \sim \mathcal{N}(0, I), \quad n = 0, \ldots, N-1, \tag{A.49}$$

for some time step n, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are some matrices determined by the numerical method used for time-discretisation, such as the Crank-Nicolson scheme. Let us also assume a random initial condition distributed according to a Gaussian

$$\boldsymbol{u}^0 \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{P}^{-1}), \tag{A.50}$$

for some mean $\boldsymbol{m}$ and precision $\boldsymbol{P}$. Assuming that we can write $\boldsymbol{P} = (\Delta t \Delta x/\sigma^2)\, \boldsymbol{L}^\top \boldsymbol{L}$ for some invertible matrix $\boldsymbol{L}$, the initial condition can be re-expressed as

$$\boldsymbol{L}\boldsymbol{u}_0 = \boldsymbol{L}\boldsymbol{m} + \frac{\sigma}{\sqrt{\Delta t \Delta x}} z_0, \quad z_0 \sim \mathcal{N}(0, \boldsymbol{I}). \tag{A.51}$$

Then, combining equations (A.49) and (A.51), we get a large matrix-vector system of the form

$$\begin{pmatrix} \boldsymbol{L} & 0 & 0 & \cdots & 0 & 0 \\ -\boldsymbol{B} & \boldsymbol{A} & 0 & \cdots & 0 & 0 \\ 0 & -\boldsymbol{B} & \boldsymbol{A} & \cdots & 0 & 0 \\ & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\boldsymbol{B} & \boldsymbol{A} \end{pmatrix} \begin{bmatrix} \boldsymbol{u}^0 \\ \boldsymbol{u}^1 \\ \boldsymbol{u}^2 \\ \vdots \\ \boldsymbol{u}^N \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}\boldsymbol{m} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \frac{\sigma}{\sqrt{\Delta t \Delta x}} \begin{bmatrix} z^0 \\ z^1 \\ z^2 \\ \vdots \\ z^N \end{bmatrix}. \tag{A.52}$$

Now, denoting the matrix on the LHS by $\boldsymbol{M}$, we have the solution

$$\begin{bmatrix} \boldsymbol{u}^0 \\ \boldsymbol{u}^1 \\ \boldsymbol{u}^2 \\ \vdots \\ \boldsymbol{u}^N \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{Lm} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \frac{\sigma^2}{\Delta t \Delta x} (\boldsymbol{M}^\top \boldsymbol{M})^{-1} \right), \tag{A.53}$$

which is a GMRF if the precision $\boldsymbol{M}^\top \boldsymbol{M}$ is sparse. In natural parameterisation, this has a neater expression, with precision matrix

$$\boldsymbol{Q} = \frac{\Delta t \Delta x}{\sigma^2} \boldsymbol{M}^\top \boldsymbol{M} \tag{A.54}$$

and shift vector

$$\boldsymbol{b} := \boldsymbol{P}\boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{Lm} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \frac{\Delta t \Delta x}{\sigma^2} \boldsymbol{M}^\top \begin{bmatrix} \boldsymbol{Lm} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \frac{\Delta t \Delta x}{\sigma^2} \begin{bmatrix} \boldsymbol{L}^\top \boldsymbol{Lm} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{Pm} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{A.55}$$

Example C.2. Using the Crank-Nicolson scheme, one can discretise the system (A.48) as

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \frac{\Delta t}{2} \left(\boldsymbol{L}\boldsymbol{u}^{n+1} + \boldsymbol{L}\boldsymbol{u}^n \right) + \sqrt{\frac{\sigma^2 \Delta t}{\Delta x}} \boldsymbol{z}^{n+1} \tag{A.56}$$

$$\Rightarrow \mathbf{A}\boldsymbol{u}^{n+1} = \boldsymbol{B}\boldsymbol{u}^n + \frac{\sigma}{\sqrt{\Delta t \Delta x}} \boldsymbol{z}^{n+1}, \tag{A.57}$$

where $\boldsymbol{L}$ is the discretised Laplacian operator and

$$\boldsymbol{A} := \frac{1}{\Delta t}\boldsymbol{I} - \frac{1}{2}\boldsymbol{L}, \qquad \boldsymbol{B} := \frac{1}{\Delta t}\boldsymbol{I} + \frac{1}{2}\boldsymbol{L}. \tag{A.58}$$

D. Additional Results

D.1. Choice of Hyperparameters

We perform a grid search to set the c and learning rate hyperparameters, which are crucial to the convergence of the algorithm. If c has too small a magnitude, or if the learning rate is too high, the algorithm will diverge. We perform the grid search on simulated data of various grid sizes. Figure 4 highlights the speed of convergence of several values of c, and Table 2 gives the full results. Based on these results we choose $c = 10$ and $\eta = 0.6$ for the rest of our experiments.

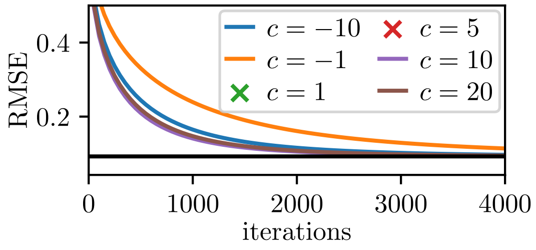

Figure 4. *Convergence of message passing for different values of c for a grid size of* 256×256*. For each c we plot the result for the best* η*.* × *indicates that the algorithm diverged. The black horizontal line shows INLA.*

Table 2. *Grid search over c and* η *for various grid sizes and an observation density of 5%. We report the ratio* $\frac{\textit{RMSE of message passing}}{\textit{RMSE of INLA}}$ *after* 4000 *iterations, with lower values being better. "-" indicates that the method diverged for that combination of hyperparameters.*

		c						
grid size	η	-10	-2	-1	1	5	10	20
128×128	0.6	3.83	3.88	3.99	-	-	**3.82**	**3.82**
	0.7	-	3.85	3.92	-	-	-	-
	0.8	-	-	-	-	-	-	-
256×256	0.6	2.04	2.19	2.40	-	-	**1.98**	2.00
	0.7	-	2.11	2.28	-	-	-	-
	0.8	-	-	-	-	-	-	-
512×512	0.6	1.30	1.68	2.18	-	-	**1.13**	1.17
	0.7	-	1.49	1.90	-	-	-	-
	0.8	-	-	-	-	-	-	-

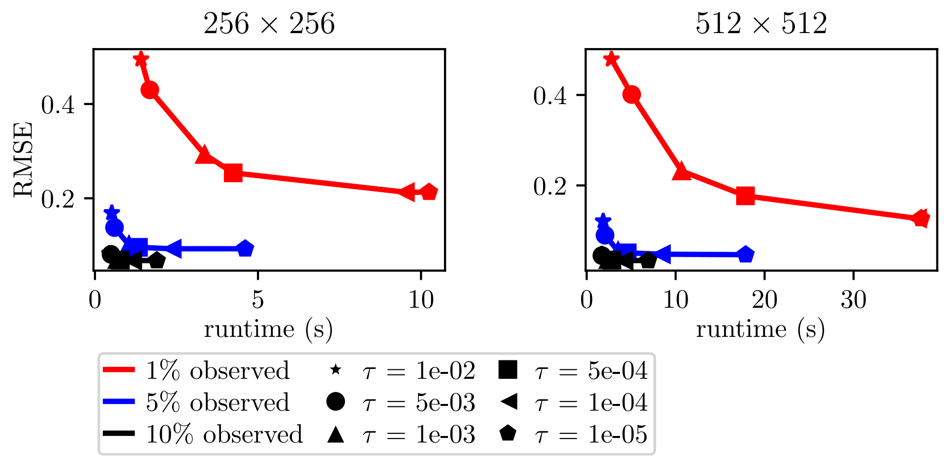

Figure 5. *Effect of the early stopping hyperparameter, τ, on the error and runtime of multigrid message passing. We consider two grid sizes,* 256×256 *and* 512×512*, and* 1%, 5%, *and* 10% *of the grid being observed. Based on these results we select* $\tau = 1 \times 10^{-3}$.

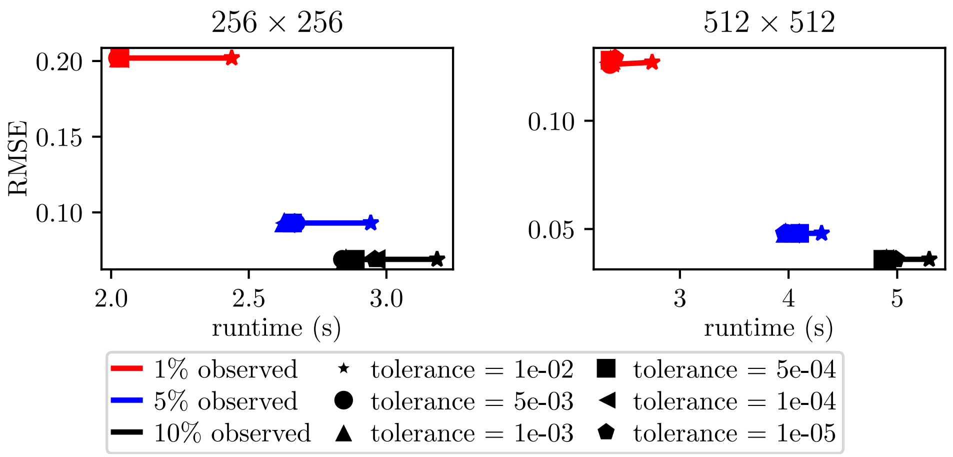

Figure 6. *Effect of the early stopping hyperparameter on the error and runtime of 3D-Var. In particular, we search over the* tol *hyperparameter of the JAXopt L-BFGS optimiser. We consider two grid sizes,* 256×256 *and* 512×512*, and* 1%, 5%, *and* 10% *of the grid being observed. Based on these results we select* tol $= 1 \times 10^{-3}$.

D.2. Temperature Data

Figure 7 showcases the L_1 error for our message passing approach, our 3D-Var implementation and for the prior mean field. The mean of ERA5 surface temperature from 06:00UTC 1st January for the years 2000 to 2019 forms the prior mean field. This prior mean field is mapped to the high-resolution grid and the high-resolution data is valid for 06:00 UTC 1st January 2020. For the DA approaches we see an improvement when compared to the error of the prior mean, with the message-passing approach performing better. The message passing implementation benefits from the multigrid approach with observations from the satellite tracks being propagated away from the locations. It should be noted that a hyperparameter tuning of the 3D-Var implementation, which is not carried out here, could improve its performance.

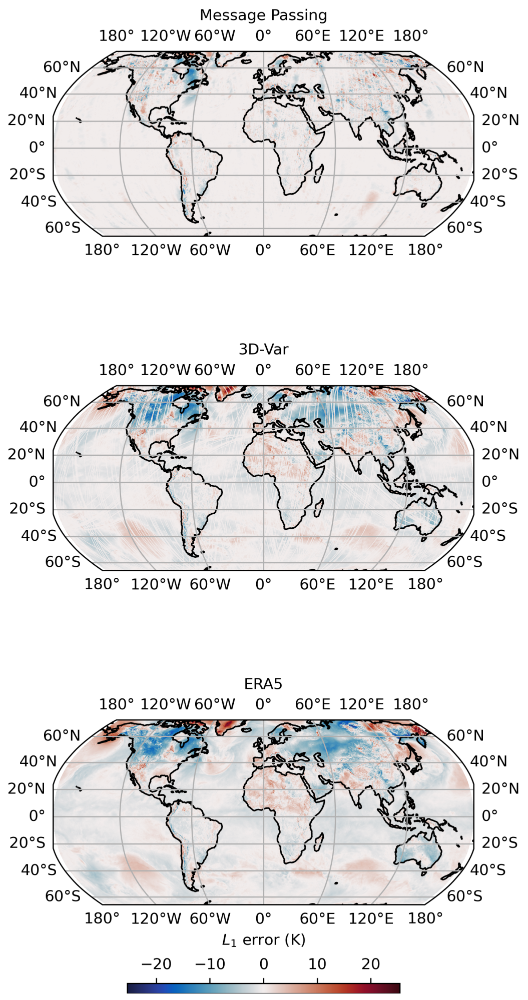

Figure 7. *L_1 errors for the message passing, 3D-Var and the prior (ERA5) against the high resolution Met Office Unified Model temperature data.*

E. Experiment Details

All experiments are performed on a 24-core AMD Threadripper 3960X CPU and an Nvidia RTX 3090 GPU. We use JAX version 0.4.23, jaxlib 0.4.23+cuda12.cudnn89.

Message passing hyperparameters. Unless otherwise mentioned we use $c = 10$ and $\sigma = 0.6$, as selected in the grid search in Appendix D.1. We set $T = 10,000$, though this many iterations is rarely used because of the early stopping.

3D-Var hyperparameters. We use the default hyperparameters of L-BFGS as implemented in JAXopt version 0.8.3. In particular max iterations = 500, stopping tolerance = 10^{-3}, and zoom line search.

R-INLA. We use version 23.9.9.

Synthetic data. We generate the data by simulating the SPDE given in Equation (3), using a finite difference solver. We set $\alpha = 2$, $\kappa = \sqrt{2\nu}/l$, with $\nu = 1$, $l = 0.15$, and $\sigma = 1.1$. We then use this SPDE as the prior for inference.

Global temperature data. Message passing uses three multigrid levels of grid sizes 625×375, 1250×750 and 2500 × 1500. Table 3 lists the satellites used to generate the observation locations. To specify the prior we set $\alpha = 2$, $\kappa = \sqrt{2\nu}/l$, with $\nu = 1$, $l = 0.2$, and $\sigma = 1.9$.

Table 3. *List of satellites used to generate the observation locations in the global temperature data example.*

NOAA 15	DMSP 5D-3 F16 (USA 172)	NOAA 18	METEOSAT-9 (MSG-2)
EWS-G1 (GOES 13)	DMSP 5D-3 F17 (USA 191)	FENGYUN 3A	FENGYUN 2E
NOAA 19	GOES 14	DMSP 5D-3 F18 (USA 210)	EWS-G2 (GOES 15)
COMS 1	FENGYUN 3B	SUOMI NPP	FENGYUN 2F
METEOSAT-10 (MSG-3)	METOP-B	FENGYUN 3C	METEOR-M 2
HIMAWARI-8	FENGYUN 2G	METEOSAT-11 (MSG-4)	ELEKTRO-L 2
HIMAWARI-9	GOES 16	FENGYUN 4A	CYGFM05
CYGFM04	CYGFM02	CYGFM01	CYGFM08
CYGFM07	CYGFM03	FENGYUN 3D	NOAA 20
GOES 17	FENGYUN 2H	METOP-C	GEO-KOMPSAT-2A
METEOR-M2 2	ARKTIKA-M 1	FENGYUN 3E	GOES 18
NOAA 21 (JPSS-2)	METEOSAT-12 (MTG-I1)	TIANMU-1 03	TIANMU-1 04
TIANMU-1 05	TIANMU-1 06	METEOR-M2 3	TIANMU-1 07
TIANMU-1 08	TIANMU-1 09	TIANMU-1 10	FENGYUN 3F
TIANMU-1 11	TIANMU-1 20	TIANMU-1 21	TIANMU-1 22
TIANMU-1 15	TIANMU-1 16	TIANMU-1 17	TIANMU-1 18

REFERENCES

Elidan, G., McGraw, I., and Koller, D. (2006). “Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing". In: *Conference on Uncertainty in Artificial Intelligence*, pp. 165–173.

Baer, M. (2018). *findiff Software Package*. URL: https://github.com/maroba/findiff.

Gonzalez, J., Low, Y., and Guestrin, C. (2009). “Residual Splash for Optimally Parallelizing Belief Propagation". In: *Artificial Intelligence and Statistics*, pp. 177–184.

Lototsky, S. V. and Rozovsky, B. L. (2017). *Stochastic Partial Differential Equations*.

Minka, T. (2004). *Power EP*. Tech. rep. Microsoft Research, Cambridge.

Van der Merwe, M., Joseph, V., and Gopalakrishnan, G. (2019). “Message Scheduling for Performant, Many-core Belief Propagation". In: *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1-–7.