MAGI Algorithm Description

Kamil Hlubek

June 16, 2025

Notation

- $\mathcal{I} = \{t_1, \dots, t_n\}$: Discretization time grid.
- D: Number of dimensions (state variables).
- n: Number of discretization time points.
- k: Number of ODE parameters.
- N_{obs}: Number of distinct observation time points (may be less than n).
- $Y \in \mathbb{R}^{n \times D}$: Observation matrix on grid $\mathcal{I}$, with NaNs for unobserved points/times. $Y_{j,d}$ is observation for dimension d at time t_j. Let $Y_d = Y_{:,d}$ be the vector for dimension d.
- $X \in \mathbb{R}^{n \times D}$: Latent state matrix on grid $\mathcal{I}$. $X_{j,d} = x_d(t_j)$. Let $X_d = X_{:,d}$ be the vector for dimension d.
- $\theta \in \Theta \subset \mathbb{R}^k$: Vector of ODE parameters. Θ defines the parameter bounds $[\theta_{lower}, \theta_{upper}]$.
- $\sigma \in \mathbb{R}^D_{>0}$: Vector of observation noise standard deviations for each dimension. $\sigma^2 = (\sigma_1^2, \dots, \sigma_D^2)$.
- $\phi_d = (\phi_{d,1}, \phi_{d,2})$: GP hyperparameters (variance $\phi_{d,1}$, lengthscale $\phi_{d,2}$) for dimension d. $\phi \in \mathbb{R}^{2 \times D}_{>0}$.
- $\mathcal{K}_{\phi_d}(t, t')$: GP covariance kernel function for dimension d.
- $f(x, \theta, t) : \mathbb{R}^D \times \mathbb{R}^k \times \mathbb{R} \to \mathbb{R}^D$: ODE function, $dx/dt = f(x, \theta, t)$. f_d is the d-th component. Let $F_d \in \mathbb{R}^n$ be the vector where $[F_d]_j = f_d(X_{j,:}, \theta, t_j)$.
- $J_x(x, \theta, t) = \nabla_x f(x, \theta, t) \in \mathbb{R}^{D \times D}$: Jacobian of f w.r.t. state x.
- $J_\theta(x, \theta, t) = \nabla_\theta f(x, \theta, t) \in \mathbb{R}^{D \times k}$: Jacobian of f w.r.t. parameters θ.
- $C_{\phi_d} \in \mathbb{R}^{n \times n}$: GP prior covariance matrix for dim d, $[C_{\phi_d}]_{jj'} = \mathcal{K}_{\phi_d}(t_j, t_{j'})$.
- $C'_{\phi_d} \in \mathbb{R}^{n \times n}$: First time derivative matrix, $[C'_{\phi_d}]_{jj'} = \frac{\partial}{\partial t}\mathcal{K}_{\phi_d}(t, t')|_{t=t_j, t'=t_{j'}}$.
- $C''_{\phi_d} \in \mathbb{R}^{n \times n}$: Second time derivative matrix, $[C''_{\phi_d}]_{jj'} = \frac{\partial^2}{\partial t \partial t'}\mathcal{K}_{\phi_d}(t, t')|_{t=t_j, t'=t_{j'}}$.
- ϵ: Small jitter value for numerical stability ($\epsilon \approx 10^{-6}$).
- $C_{inv,d} = (C_{\phi_d} + \epsilon I)^{-1}$: Inverse of jittered prior covariance matrix.
- $m_{\phi_d} = C'_{\phi_d} C_{inv,d}$: Matrix relating X_d to the conditional mean of its derivative $\dot{X}_d$. $\mathbb{E}[\dot{X}_d | X_d] = m_{\phi_d} X_d$ (assuming zero prior mean).

- $K_{\phi_d} = C''_{\phi_d} - m_{\phi_d}(C'_{\phi_d})^\mathsf{T}$: Conditional covariance matrix $\mathbb{V}[\dot{X}_d|X_d]$.
- $K_{inv,d} = (K_{\phi_d} + \epsilon I)^{-1}$: Inverse of jittered conditional covariance matrix (precision matrix).
- M^{Band}: Banded matrix approximation of matrix M with bandwidth b.
- $\beta = (\beta_{deriv}, \beta_{level}, \beta_{obs})$: Prior temperature vector.
- Ψ: Full parameter vector for sampling, e.g., $(\text{vec}(X), \theta, \log\sigma)$ or $(\text{vec}(X), \theta)$.
- $||\mathbf{v}||^2_\mathbf{A} = \mathbf{v}^\mathsf{T}\mathbf{A}\mathbf{v}$: Squared Mahalanobis norm.

1 Initialization

The goal of initialization is to determine suitable starting values for the latent states ($X^{(0)}$), ODE parameters ($\theta^{(0)}$), observation noise standard deviations ($\sigma^{(0)}$), and GP hyperparameters ($\phi^{(0)}$). These serve as the initial state $\Psi^{(0)}$ for the MCMC sampler.

1.1 Handling User-Provided Initial State

If a complete initial state vector Ψ_{init} is provided by the user, it is unpacked directly into $X^{(0)}$, $\theta^{(0)}$, and possibly the log-transformed $\log\sigma^{(0)}$. The parameter vector $\theta^{(0)}$ is clamped to lie within the specified bounds $\Theta = [\theta_{lower}, \theta_{upper}]$. The flag `sigma_is_fixed` is determined based on whether Ψ_{init} includes the $\log\sigma$ component. If σ is sampled (i.e., `sigma_is_fixed` is false), the initial value is set as $\sigma^{(0)} = \exp(\log\sigma^{(0)})$. If σ is fixed, $\sigma^{(0)}$ is set to the value σ_{fixed} provided in the configuration (this is mandatory if Ψ_{init} implies fixed sigma). Similarly, if Ψ_{init} implies fixed sigma, the GP hyperparameters ϕ_{fixed} must be provided in the configuration, and $\phi^{(0)}$ is set to ϕ_{fixed}. If Ψ_{init} is provided, the subsequent initialization steps for ϕ, σ, X, θ are skipped, and the algorithm proceeds directly to pre-computing the GP covariance structures using $\phi^{(0)}$.

1.2 Initialize GP Hyperparameters (ϕ) and Noise (σ)

This step is performed if Ψ_{init} was not provided, AND either the GP hyperparameters ϕ_{fixed} were not provided or the noise standard deviations σ_{fixed} were not provided (meaning σ needs to be sampled).

The determination of whether σ is fixed relies on the configuration: `sigma_is_fixed` is true if and only if both σ_{fixed} and ϕ_{fixed} are provided.

The initialization proceeds dimension by dimension ($d = 1, \ldots, D$):

1. Identify the valid (non-NaN) observations $Y_{valid,d}$ in the d-th column of Y and their corresponding times $t_{valid,d}$.
2. Compute initial guesses for the log-transformed hyperparameters: $(\log\phi^{(guess)}_{d,1}, \log\phi^{(guess)}_{d,2})$ and $\log\sigma^{(guess)}_d$. These guesses are derived from basic statistics of $Y_{valid,d}$ (e.g., variance, median absolute deviation) and the time range of observations. If ϕ_{fixed} was provided (but σ_{fixed} was not), its values are used for the ϕ guess: $(\log\phi_{fixed,1,d}, \log\phi_{fixed,2,d})$.
3. Define the Negative Log Marginal Likelihood (NLML) objective function for dimension d. Assuming a GP prior $x_d \sim \mathcal{GP}(0, \mathcal{K}_{\phi_d})$ and observation model $y_d = x_d + \epsilon_d$ with $\epsilon_d \sim \mathcal{N}(0, \sigma_d^2 I)$, the marginal likelihood integrates out x_d. The NLML, as a function of log-parameters $\lambda = (\log\phi_{d,1}, \log\phi_{d,2}, \log\sigma_d)$, is:

$$L_{NLML}(\lambda_1, \lambda_2, \lambda_3) = \frac{1}{2}\log|\mathbf{K}_{\phi^*_d} + (\sigma^*_d)^2\mathbf{I}| + \frac{1}{2}\mathbf{Y}^\mathsf{T}_{valid,d}(\mathbf{K}_{\phi^*_d} + (\sigma^*_d)^2\mathbf{I})^{-1}\mathbf{Y}_{valid,d} + \text{const}$$

where $\phi^*_{d,1} = \exp(\lambda_1)$, $\phi^*_{d,2} = \exp(\lambda_2)$, $\sigma^*_d = \exp(\lambda_3)$, and $\mathbf{K}_{\phi^*_d} = [\mathcal{K}_{(\phi^*_{d,1},\phi^*_{d,2})}(t, t')]_{t,t' \in t_{valid,d}}$.

4. Minimize $L_{NLML}(\lambda)$ using a numerical optimization algorithm (e.g., Nelder-Mead), starting from the initial guess $\lambda^{(guess)}$, to obtain the optimized log-parameters $\lambda^* = (\log \phi^*_{d,1}, \log \phi^*_{d,2}, \log \sigma^*_d)$.

5. Store the optimized parameters:

   - If ϕ_{fixed} was not provided, store $\phi_{est,1,d} = \exp(\lambda^*_1)$ and $\phi_{est,2,d} = \exp(\lambda^*_2)$.
   - If `sigma_is_fixed` is false, store $\sigma_{est,d} = \exp(\lambda^*_3)$.

After iterating through all dimensions, set the final initial GP hyperparameters $\phi^{(0)}$ to ϕ_{est} (if estimated) or ϕ_{fixed} (if provided). Set the initial noise $\sigma^{(0)}$ to σ_{est} (if estimated) or σ_{fixed} (if provided and fixed).

1.3 Initialize Latent States (X)

This step is performed if Ψ_{init} and X_{init} were not provided. Initialize the latent state matrix $X^{(0)} \in \mathbb{R}^{n \times D}$. For each dimension $d = 1, \ldots, D$, identify the non-NaN observations $Y_{valid,d}$ at times $t_{valid,d}$. Linearly interpolate these values onto the full discretization grid $\mathcal{I}$ to obtain the initial trajectory $X^{(0)}_{:,d}$. Handle cases with fewer than two observations appropriately (e.g., constant extrapolation).

If X_{init} was provided, set $X^{(0)} = X_{init}$ after checking dimensions.

1.4 Initialize ODE Parameters (θ)

This step is performed if Ψ_{init} and θ_{init} were not provided. Initialize the ODE parameter vector $\theta^{(0)} \in \mathbb{R}^k$. Typically, this is done by taking the midpoint of the provided bounds $\Theta = [\theta_{lower}, \theta_{upper}]$, or offsetting slightly from a bound if one side is infinite.

If θ_{init} was provided, set $\theta^{(0)} = \theta_{init}$.

Finally, clamp $\theta^{(0)}$ to ensure it lies within the bounds: $\theta^{(0)} = \max(\theta_{lower}, \min(\theta^{(0)}, \theta_{upper}))$.

1.5 Pre-compute GP Covariance Structures

Using the initialized or provided GP hyperparameters $\phi^{(0)}$, pre-compute the necessary covariance matrices and their banded approximations for each dimension $d = 1, \ldots, D$. These matrices are essential for evaluating the GP prior terms ($\log p(X_d)$) and the GP conditional derivative terms ($\log p(\dot{X}_d = F_d | X_d)$) in the log-likelihood. Store these in a structure, `gp_cov_all_dims`.

For each dimension d:

1. Select the kernel function $\mathcal{K}_{\phi_d}$ based on the kernel type specified in the configuration and the parameters $\phi_d = \phi^{(0)}_{:,d}$.

2. Compute the dense $n \times n$ prior covariance matrix $C_{\phi_d} = [\mathcal{K}_{\phi_d}(t_j, t_{j'})]^n_{j,j'=1}$.

3. Compute the dense first time derivative matrix C'_{ϕ_d}.

4. Compute the dense second time derivative matrix C''_{ϕ_d}.

5. Compute the inverse of the jittered prior covariance: $C_{inv,d} = (C_{\phi_d} + \epsilon I)^{-1}$. This is needed for the $\mathcal{L}_{level}$ term and its gradient.

6. Compute the matrix $m_{\phi_d} = C'_{\phi_d} C_{inv,d}$. This relates the level X_d to the conditional mean of the derivative $\mathbb{E}[\dot{X}_d | X_d] = m_{\phi_d} X_d$ (assuming zero prior mean).

7. Compute the conditional covariance matrix $K_{\phi_d} = \mathbb{V}[\dot{X}_d | X_d] = C''_{\phi_d} - m_{\phi_d} (C'_{\phi_d})^\mathsf{T}$.

8. Compute the inverse of the jittered conditional covariance (precision matrix): $K_{inv,d} = (K_{\phi_d} + \epsilon I)^{-1}$. This is needed for the $\mathcal{L}_{deriv}$ term and its gradient.

9. Create banded matrix approximations using the specified bandwidth b: $C_{inv,d}^{Band}$, $m_{\phi_d}^{Band}$, $K_{inv,d}^{Band}$. These are used for efficient computation.

10. Store $C_{inv,d}^{Band}$, $m_{\phi_d}^{Band}$, $K_{inv,d}^{Band}$ in the structure `gp_cov_all_dims[d]`. (The dense matrices K_{ϕ_d}, C_{ϕ_d} etc. may also be stored for reference).

1.6 Construct Initial Sampler State $\Psi^{(0)}$

This step constructs the final initial state vector for the MCMC sampler, unless it was provided directly as Ψ_{init}.

- If `sigma_is_fixed` is true:

$$\Psi^{(0)} = \begin{pmatrix} \text{vec}(X^{(0)}) \\ \theta^{(0)} \end{pmatrix}$$

- If `sigma_is_fixed` is false: Compute the log-transformed noise $\log \sigma^{(0)} = \log(\max(\sigma^{(0)}, \text{small\_val}))$ (element-wise, ensuring positivity).

$$\Psi^{(0)} = \begin{pmatrix} \text{vec}(X^{(0)}) \\ \theta^{(0)} \\ \log \sigma^{(0)} \end{pmatrix}$$

This $\Psi^{(0)}$ is the starting point for the MCMC sampler described in Section 3.

2 Log-Likelihood & Gradient Calculation: $\mathcal{L}(\Psi)$ and $\nabla\mathcal{L}(\Psi)$

This section describes the calculation of the log-posterior density $\mathcal{L}(\Psi)$ and its gradient $\nabla\mathcal{L}(\Psi)$ with respect to the full parameter vector $\Psi = (\text{vec}(X), \theta, \dots)$. The log-posterior combines the log-likelihood of the observations, the log-prior density from the GP assumptions on levels X and derivatives $\dot{X}$, and the log-prior density of the ODE parameters θ (often assumed uniform within bounds, contributing only via the bounds). The gradient is required for HMC sampling.

2.1 Parameter Unpacking and Transformation

Given an input parameter vector Ψ:

1. Unpack Ψ into the latent state matrix $X \in \mathbb{R}^{n \times D}$ and the ODE parameter vector $\theta \in \mathbb{R}^k$.

2. Check if σ is fixed using the `sigma_is_fixed` flag determined during initialization.

3. If `sigma_is_fixed` is true, use the fixed value $\sigma = \sigma^{(0)}$ determined during initialization. The log-Jacobian contribution is $\mathcal{L}_{jac} = 0$.

4. If `sigma_is_fixed` is false, unpack the remaining elements of Ψ as $\log \sigma \in \mathbb{R}^D$. Transform to the original scale: $\sigma = \exp(\log \sigma)$ (element-wise). Ensure resulting $\sigma_d > 0$. The log-Jacobian determinant for this transformation is $\mathcal{L}_{jac} = \sum_{d=1}^{D} (\log \sigma)_d$. This term is added to the total log-likelihood, effectively corresponding to a $p(\sigma_d) \propto 1/\sigma_d$ prior on σ_d.

2.2 Log-Likelihood Term Calculation

Initialize the total log-likelihood $\mathcal{L} = \mathcal{L}_{jac}$. Pre-compute the ODE function values $F \in \mathbb{R}^{n \times D}$ where $F_{j,d} = f_d(X_{j,:}, \theta, t_j)$ for all $j = 1..n, d = 1..D$.

Iterate through each dimension $d = 1, \dots, D$:

1. Retrieve the pre-computed banded matrices $C_{inv,d}^{Band}$, m_d^{Band}, $K_{inv,d}^{Band}$ from `gp_cov_all_dims[d]`.

2. Define vectors for the current dimension: $X_d = X_{:,d}$, $F_d = F_{:,d}$, $Y_d = Y_{:,d}$.
3. Identify the set of time indices $\mathcal{V}_d = \{j | Y_{j,d} \text{ is not NaN}\}$ where observations are available for dimension d.
4. Calculate the level fit error: $\texttt{fitLevelError}_d = X_d - Y_d$. Set entries corresponding to NaN observations in Y_d to zero.
5. Calculate the expected derivative based on the GP mean: $\texttt{mphi\_x}_d = m_d^{Band} X_d$.
6. Calculate the derivative fit error: $\texttt{fitDerivError}_d = F_d - \texttt{mphi\_x}_d$.
7. Calculate intermediate terms needed for the likelihood components:
   - $\texttt{Kinv\_fitDerivError}_d = K_{inv,d}^{Band} \times \texttt{fitDerivError}_d$
   - $\texttt{Cinv\_x}_d = C_{inv,d}^{Band} \times X_d$
8. Calculate the three components of the log-likelihood for dimension d:
   - **Level Log-Likelihood ($\log p(X_d)$):** The GP prior assumes $X_d \sim \mathcal{N}(0, C_{\phi_d})$. The log-density is:

     $$\log p(X_d) = -\frac{1}{2} X_d^\top C_{\phi_d}^{-1} X_d - \frac{1}{2} \log |C_{\phi_d}| - \frac{n}{2} \log(2\pi)$$

     Using the pre-computed inverse of the jittered matrix $C_{inv,d}$ and the term $\texttt{Cinv\_x}_d = C_{inv,d} X_d$, and ignoring constant terms and log-determinants (if ϕ is fixed), this component is calculated as:

     $$\mathcal{L}_{level,d} = -\frac{1}{2} X_d^\top C_{inv,d} X_d = -\frac{1}{2} X_d^\top (\texttt{Cinv\_x}_d)$$

     This penalizes deviations from the zero mean prior, weighted by the prior precision $C_{inv,d}$, corresponding to $-\frac{1}{2}\mathcal{L}_{GP,d}$ in Eq. (5) of the reference PDF [71-73].
   - **Observation Log-Likelihood ($\log p(Y_d | X_d, \sigma_d^2)$):** The model assumes $Y_{j,d} | X_{j,d} \sim \mathcal{N}(X_{j,d}, \sigma_d^2)$ independently for observed points $j \in \mathcal{V}_d$. The joint log-density is:

     $$\begin{aligned} \log p(Y_d(\mathcal{V}_d) | X_d(\mathcal{V}_d), \sigma_d^2) &= \sum_{j \in \mathcal{V}_d} \log p(Y_{j,d} | X_{j,d}, \sigma_d^2) \\ &= \sum_{j \in \mathcal{V}_d} \left[-\frac{1}{2} \log(2\pi\sigma_d^2) - \frac{(X_{j,d} - Y_{j,d})^2}{2\sigma_d^2} \right] \\ &= -\frac{|\mathcal{V}_d|}{2} \log(2\pi\sigma_d^2) - \frac{1}{2\sigma_d^2} \sum_{j \in \mathcal{V}_d} (X_{j,d} - Y_{j,d})^2 \end{aligned}$$

     Using the pre-calculated $\texttt{fitLevelError}_{d,j} = X_{j,d} - Y_{j,d}$, this is computed as:

     $$\mathcal{L}_{obs,d} = -\frac{1}{2\sigma_d^2} \sum_{j \in \mathcal{V}_d} (\texttt{fitLevelError}_{d,j})^2 - \frac{|\mathcal{V}_d|}{2} \log(2\pi\sigma_d^2)$$

     This represents the goodness-of-fit, corresponding to $-\frac{1}{2}\mathcal{L}_{Obs,d}$ in Eq. (5) [73].
   - **Derivative Log-Likelihood ($\log p(\dot{X}_d = F_d | X_d)$):** From GP properties, the conditional distribution is $\dot{X}_d | X_d \sim \mathcal{N}(m_{\phi_d} X_d, K_{\phi_d})$. The manifold constraint implies $\dot{X}_d = F_d$. Evaluating the log-density of this conditional normal distribution at F_d:

     $$\log p(F_d | X_d) = -\frac{1}{2} (F_d - m_{\phi_d} X_d)^\top K_{\phi_d}^{-1} (F_d - m_{\phi_d} X_d) - \frac{1}{2} \log |K_{\phi_d}| - \frac{n}{2} \log(2\pi)$$

Using the jittered inverse $K_{inv,d} = (K_{\phi_d} + \epsilon I)^{-1}$ and the derivative error $E_d = \texttt{fitDerivError}_d = F_d - \texttt{mphi\_x}_d$, and ignoring constants and log-determinants, this component is calculated as:

$$\mathcal{L}_{deriv,d} = -\frac{1}{2} E_d^{\mathsf{T}} K_{inv,d} E_d = -\frac{1}{2} (\texttt{fitDerivError}_d)^{\mathsf{T}} (\texttt{Kinv\_fitDerivError}_d)$$

This term enforces the ODE manifold constraint by penalizing the squared Mahalanobis distance between the ODE dynamics F_d and the GP's expected derivative $m_{\phi_d} X_d$, weighted by the conditional precision $K_{inv,d}$. It corresponds to $-\frac{1}{2}\mathcal{L}_{Constraint,d}$ in Eq. (5) [74-77].

9. Add the weighted components to the total log-likelihood:

$$\mathcal{L} \leftarrow \mathcal{L} + \frac{\mathcal{L}_{obs,d}}{\beta_{obs}} + \frac{\mathcal{L}_{deriv,d}}{\beta_{deriv}} + \frac{\mathcal{L}_{level,d}}{\beta_{level}}$$

2.3 Gradient Calculation

The gradient $\nabla \mathcal{L}(\Psi)$ required for HMC is computed by summing the gradients of the weighted log-likelihood components derived above. These gradients are obtained analytically using standard matrix and vector calculus.

Analytical Gradient Components The gradient of the total log-likelihood is the sum of the gradients of its constituent parts (weighted by inverse temperatures $1/\beta$), plus the gradient of the log-Jacobian term $\mathcal{L}_{jac}$ (if applicable). We consider the gradient of each main term:

- **Gradient of $\mathcal{L}_{level,d}$**: Since $\mathcal{L}_{level,d} = -\frac{1}{2} X_d^{\mathsf{T}} C_{inv,d} X_d + \text{const}$, its gradient w.r.t. X_d is $\nabla_{X_d} \mathcal{L}_{level,d} = -C_{inv,d} X_d$. It has no dependence on θ or σ.

- **Gradient of $\mathcal{L}_{obs,d}$**: Since $\mathcal{L}_{obs,d} = -\frac{1}{2\sigma_d^2} \sum_{j \in \mathcal{V}_d} (X_{j,d} - Y_{j,d})^2 + \ldots$, its gradient w.r.t. $X_{j,d}$ is $\nabla_{X_{j,d}} \mathcal{L}_{obs,d} = -\frac{1}{\sigma_d^2}(X_{j,d} - Y_{j,d})$ for observed points ($j \in \mathcal{V}_d$) and zero otherwise. The gradient w.r.t. σ_d is $\nabla_{\sigma_d} \mathcal{L}_{obs,d} = \frac{1}{\sigma_d^3} \sum_{j \in \mathcal{V}_d} (X_{j,d} - Y_{j,d})^2 - \frac{|\mathcal{V}_d|}{\sigma_d}$. It does not depend on θ.

- **Gradient of $\mathcal{L}_{deriv,d}$**: This term, $\mathcal{L}_{deriv,d} = -\frac{1}{2}(F_d - m_{\phi_d} X_d)^{\mathsf{T}} K_{inv,d} (F_d - m_{\phi_d} X_d) + \text{const}$, depends on X (directly via $m_{\phi_d} X_d$ and indirectly via $F_d = f(X, \theta, t)$) and θ (via F_d). Let $E_d = F_d - m_{\phi_d} X_d$ be the error vector. Using the chain rule for quadratic forms ($\nabla_z(-\frac{1}{2} v^{\mathsf{T}} A v) = -(\nabla_z v)^{\mathsf{T}} A v$ for symmetric A), we get:

  - $\nabla_\theta \mathcal{L}_{deriv,d} = -(\nabla_\theta E_d)^{\mathsf{T}} K_{inv,d} E_d$. Since only F_d depends on θ, $\nabla_\theta E_d = \nabla_\theta F_d$, where the (j,p)-th element of $\nabla_\theta F_d$ is $\frac{\partial f_d(X_{j,:}, \theta, t_j)}{\partial \theta_p}$. Thus, $\nabla_\theta \mathcal{L}_{deriv,d} = -(\nabla_\theta F_d)^{\mathsf{T}} K_{inv,d} (F_d - m_{\phi_d} X_d)$.
  - $\nabla_X \mathcal{L}_{deriv,d} = -(\nabla_X E_d)^{\mathsf{T}} K_{inv,d} E_d$. Here, $\nabla_X E_d = \nabla_X F_d - \nabla_X (m_{\phi_d} X_d)$. $\nabla_X F_d$ involves the ODE state Jacobian J_x, specifically $\frac{\partial F_{j,d}}{\partial X_{j',d'}} = \delta_{jj'} J_x(j)[d,d']$. The gradient $\nabla_X (m_{\phi_d} X_d)$ involves $m_{\phi_d}^{\mathsf{T}}$. Combining these leads to contributions to $\nabla_X \mathcal{L}_{j',d'}$ involving both $J_x(j')[d,d']$ and $(m_{\phi_d})^{\mathsf{T}}$.

  This term does not depend directly on σ.

- **Gradient of $\mathcal{L}_{jac}$**: If σ is sampled, $\mathcal{L}_{jac} = \sum \log \sigma_d$. Then $\nabla_{\log \sigma} \mathcal{L}_{jac} = \mathbf{1}$, and gradients w.r.t X, θ are zero.

The subsequent steps detail how these analytical gradient components are computed and accumulated efficiently using the pre-calculated matrices and Jacobians.

Initialize gradient components: $\nabla_X \mathcal{L} = \mathbf{0} \in \mathbb{R}^{n \times D}$, $\nabla_\theta \mathcal{L} = \mathbf{0} \in \mathbb{R}^k$, $\nabla_\sigma \mathcal{L} = \mathbf{0} \in \mathbb{R}^D$.

Accumulating Gradients:

1. **Loop through dimensions ($d = 1..D$):** Calculate contributions independent of ODE coupling.
   - $\nabla_X \mathcal{L}_{:,d} \leftarrow \nabla_X \mathcal{L}_{:,d} + \frac{1}{\beta_{level}}(\nabla_{X_d} \mathcal{L}_{level,d}) = \nabla_X \mathcal{L}_{:,d} - \frac{1}{\beta_{level}} C_{inv,d} X_d$
   - For $j \in \mathcal{V}_d$: $\nabla_X \mathcal{L}_{j,d} \leftarrow \nabla_X \mathcal{L}_{j,d} + \frac{1}{\beta_{obs}}(\nabla_{X_{j,d}} \mathcal{L}_{obs,d}) = \nabla_X \mathcal{L}_{j,d} - \frac{1}{\beta_{obs}\sigma_d^2}(\texttt{fitLevelError}_{d,j})$
   - $\nabla_X \mathcal{L}_{:,d} \leftarrow \nabla_X \mathcal{L}_{:,d} + \frac{1}{\beta_{deriv}}(\nabla_{X_d} \mathcal{L}_{deriv,d})_{\text{mphi}} = \nabla_X \mathcal{L}_{:,d} + \frac{1}{\beta_{deriv}}(m_d^{Band})^\mathsf{T}(\texttt{Kinv\_fitDerivError}_d)$ (From $-m_{\phi_d} X_d$ part of $\mathcal{L}_{deriv,d}$)
   - If not `sigma_is_fixed`: $\nabla_\sigma \mathcal{L}_d \leftarrow \nabla_\sigma \mathcal{L}_d + \frac{1}{\beta_{obs}}(\nabla_{\sigma_d} \mathcal{L}_{obs,d}) = \nabla_\sigma \mathcal{L}_d + \frac{1}{\beta_{obs}}\left(\frac{\text{SSE}_d}{\sigma_d^3} - \frac{|\mathcal{V}_d|}{\sigma_d}\right)$

2. **Loop through time ($j = 1..n$):** Calculate contributions involving ODE coupling from the F_d term in $\mathcal{L}_{deriv}$.
   - Compute Jacobians $J_x(j) = \nabla_x f(X_{j,:}, \theta, t_j)$ and $J_\theta(j) = \nabla_\theta f(X_{j,:}, \theta, t_j)$.
   - For each dimension $d = 1..D$ (of the derivative error term):
     - Let $KFE_{d,j} = [\texttt{Kinv\_fitDerivError}_d]_j$.
     - Accumulate gradient w.r.t. $X_{j,d'}$ ($d' = 1..D$):

       $$\nabla_X \mathcal{L}_{j,d'} \leftarrow \nabla_X \mathcal{L}_{j,d'} - \frac{1}{\beta_{deriv}} J_x(j)[d, d'] \cdot KFE_{d,j}$$

       (This corresponds to the $(\nabla_X F_d)^\mathsf{T}(-K_{inv,d} E_d)$ part, summed over d).
     - Accumulate gradient w.r.t. θ:

       $$\nabla_\theta \mathcal{L} \leftarrow \nabla_\theta \mathcal{L} - \frac{1}{\beta_{deriv}} J_\theta(j)[d, :]^\mathsf{T} \cdot KFE_{d,j}$$

       (This corresponds to the $(\nabla_\theta F_d)^\mathsf{T}(-K_{inv,d} E_d)$ part, summed over d).

Assemble Final Gradient Vector $\nabla \mathcal{L}(\Psi)$:

1. Combine the accumulated $\nabla_X \mathcal{L}$ (flattened) and $\nabla_\theta \mathcal{L}$ into the appropriate parts of the final gradient vector $\nabla \mathcal{L}$.

2. If `sigma_is_fixed` is false, calculate the gradient with respect to the sampled variable $\log \sigma$:

   $$\nabla_{\log \sigma} \mathcal{L} = (\nabla_\sigma \mathcal{L} \odot \sigma) + \mathbf{1}$$

   (The first term uses the chain rule, the second is the gradient of $\mathcal{L}_{jac}$). Assign this $\nabla_{\log \sigma} \mathcal{L}$ to the final elements of $\nabla \mathcal{L}$.

The function returns the pair $(\mathcal{L}, \nabla \mathcal{L})$.

3 Sampling

The final stage employs a Markov Chain Monte Carlo (MCMC) algorithm, typically the No-U-Turn Sampler (NUTS), a variant of Hamiltonian Monte Carlo (HMC), to draw samples from the posterior distribution $p(\Psi|Y) \propto \exp(\mathcal{L}(\Psi))$.

3.1 Sampler Setup

1. Initialize the NUTS sampler, providing the log-likelihood function $\mathcal{L}(\Psi)$ and its gradient $\nabla\mathcal{L}(\Psi)$ calculated as described in Section 2.
2. Define the HMC metric M, which represents the mass matrix (often diagonal Euclidean, meaning M is diagonal). This influences the kinetic energy term in the Hamiltonian.
3. Define the numerical integrator (typically Leapfrog) used to simulate Hamiltonian dynamics, along with an initial step size ϵ_0.
4. Configure an adaptor (e.g., StanHMCAdaptor) responsible for tuning the step size ϵ and potentially the metric M during the initial "burn-in" or "warmup" phase of the MCMC run. The goal is often to achieve a target acceptance ratio (e.g., 0.8).

3.2 MCMC Iteration Loop

Set the current state of the chain $\Psi_{current} = \Psi^{(0)}$ (from Section 1). Initialize storage $\mathcal{S}$ for accepted samples. Iterate i from 1 to N_{iter}:

1. Sample momentum vector p from its conditional distribution, typically $p \sim \mathcal{N}(0, M)$.
2. Propose a new state (Ψ^*, p^*) by simulating the Hamiltonian dynamics $H(\Psi, p) = -\mathcal{L}(\Psi) + \frac{1}{2}p^\top M^{-1}p$ starting from $(\Psi_{current}, p)$. The NUTS algorithm dynamically determines the simulation length (number of Leapfrog steps) to avoid random walks and efficiently explore the parameter space.
3. Calculate the Metropolis-Hastings acceptance probability:
$$\alpha = \min\left(1, \exp\left(H(\Psi_{current}, p) - H(\Psi^*, p^*)\right)\right)$$
$$\alpha = \min\left(1, \exp\left(\mathcal{L}(\Psi^*) - \mathcal{L}(\Psi_{current}) + \frac{1}{2}p^\top M^{-1}p - \frac{1}{2}(p^*)^\top M^{-1}p^*\right)\right)$$
4. Sample a random number $u \sim \text{Uniform}(0, 1)$.
5. If $u < \alpha$, accept the proposal: $\Psi_{current} = \Psi^*$. Otherwise, reject: $\Psi_{current}$ remains unchanged.
6. If the current iteration i is after the burn-in phase $(i > N_{burn})$, store the current state $\Psi_{current}$ in the sample collection $\mathcal{S}$.
7. If the current iteration i is within the burn-in phase $(i \leq N_{burn})$, use the outcome of the step (acceptance statistic, proposed state) to update the step size ϵ and potentially the metric M via the configured adaptor.

3.3 Post-processing

After the MCMC loop completes, process the collected samples $\Psi^{(i)} \in \mathcal{S}$ (for $i = N_{burn} + 1, \ldots, N_{iter}$):

1. For each sample $\Psi^{(i)}$, unpack it into its components: $X^{(i)} \in \mathbb{R}^{n \times D}$, $\theta^{(i)} \in \mathbb{R}^k$.
2. Determine $\sigma^{(i)}$:
   - If `sigma_is_fixed` was true during the run, set $\sigma^{(i)} = \sigma^{(0)}$ (the fixed value).
   - If `sigma_is_fixed` was false, extract the $\log\sigma^{(i)}$ component from $\Psi^{(i)}$ and transform it back: $\sigma^{(i)} = \exp(\log\sigma^{(i)})$.

The final output of the algorithm is the set of post-burn-in samples $\{(X^{(i)}, \theta^{(i)}, \sigma^{(i)})\}_{i=N_{burn}+1}^{N_{iter}}$. These samples represent draws from the posterior distribution and can be used for inference (e.g., calculating means, credible intervals).