Model Analysis: maf_gp.py

Model Analysis: `maf_gp.py`

This document provides a detailed comparison of the three model definitions in `maf_gp.py` (`model`, `model_n`, and `model_n_hv`) and an in-depth analysis of the likelihood function, specifically for the multi-output case.

1. Model Comparison

The file contains three distinct model formulations. The key differences lie in their **parameterization strategy** (Centered vs. Non-Centered), **data handling** (Single vs. Multi-output), and **prior distributions**.

Feature	`model`	`model_n`	`model_n_hv`
Type	Baseline	Reparameterized	Multi-Output Reparameterized
Parameterization	**Centered**Parameters sampled directly from priors.	**Non-Centered**Parameters sampled from $N(0, 1)$ and scaled.	**Non-Centered**Parameters sampled from $N(0, 1)$ and scaled.
Outputs	Single (Implicitly Horizontal)	Single (Switchable H/V)	**Dual (Horizontal & Vertical)**
Mean Function	$P\sin(\alpha)$	$P\sin(\alpha)$ OR $P\cos(\alpha)$	$P\sin(\alpha)$ AND $P\cos(\alpha)$
Priors	Set A	Set A	**Set B (Updated)**

Detailed Prior Comparison

A critical difference is that `model_n_hv` uses a different set of physical priors compared to `model` and `model_n`.

Parameter	`model` / `model_n` (Set A)	`model_n_hv` (Set B)	Change Analysis
E_1	$161,000 \pm 2,000$	$165,000 \pm 6,050$	**Mean ↑, Uncertainty ↑ ×3**
E_2	$11,380 \pm 100$	$11,500 \pm 250$	Mean ↑, Uncertainty ↑
ν_{12}	0.32 ± 0.01	0.36 ± 0.005	**Mean ↑, Uncertainty** ↓ ×0.5
ν_{23}	0.43 ± 0.01	0.41 ± 0.01	Mean ↓
G_{12}	$5,170 \pm 70$	$5,000 \pm 80$	Mean ↓

> [!NOTE] `model_n_hv` assumes significantly higher uncertainty for E_1 but tighter constraints on ν_{12}.

Hyperparameter Priors

In addition to the physical parameters, the model also estimates hyperparameters governing the Gaussian Process emulator and measurement noise. These are consistent across all models (though `model_n` and `model_n_hv` use reparameterization).

Hyperparameter	Description	Prior Distribution
$\mu_{emulator}$	GP Mean Offset	$\mathcal{N}(0, 0.01)$
$\sigma_{emulator}$	GP Amplitude	Exponential(20.0)
$\sigma_{measure}$	Measurement Noise	Exponential(100.0)
λ_P	Length Scale (Load)	LogNormal$(1.5, 0.5)$
λ_α	Length Scale (Angle)	LogNormal$(0.34, 0.5)$
λ_{E1}	Length Scale (E_1)	LogNormal$(11.0, 0.5)$
λ_{E2}	Length Scale (E_2)	LogNormal$(8.3, 0.5)$
$\lambda_{\nu 12}$	Length Scale (ν_{12})	LogNormal$(-0.80, 0.5)$
$\lambda_{\nu 23}$	Length Scale (ν_{23})	LogNormal$(-0.80, 0.5)$
λ_{G12}	Length Scale (G_{12})	LogNormal$(7.7, 0.5)$

Note: The length scales λ determine how quickly the model discrepancy changes with respect to changes in inputs or parameters. A small length scale implies a "wiggly" function, while a large one implies a smooth function.

Bias Modeling

The code also supports adding bias terms to account for systematic discrepancies in specific parameters. This is modeled hierarchically:

1. **Global Scale Parameter**: A global scale σ_b is sampled from an Exponential prior.
2. **Experiment-Specific Bias**: For each experiment i, a specific bias b_i is sampled from $\mathcal{N}(0, \sigma_b)$.
3. **Application**:
   - E_1 **Bias**: Added to the parameter samples for each experiment before emulation.
   - α **Bias**: Added to the angle input for each experiment.

Bias Type	Prior for Scale σ_b	Implied Mean of Scale	Physical Interpretation
E_1 **Bias** (b_{E1})	Exponential(0.001)	$1,000$ MPa	Accounts for variability in stiffness between experiments.
α **Bias** (b_α)	Exponential(1/rad(10°))	0.1745 rad $\approx 10^\circ$	Accounts for misalignment of the specimen fibers.

Note: The rate parameter for α bias is $1/(10 \cdot \frac{\pi}{180}) \approx 5.73$.

2. Likelihood Analysis

All models use a **Gaussian Process (GP) Likelihood**. The observed data $\mathbf{y}$ is modeled as coming from a Multivariate Normal distribution:

$$\mathbf{y} \sim \mathcal{N}(\mu(\mathbf{x}, \theta), \mathbf{K}(\mathbf{x}, \theta) + {}_\epsilon)$$

Where:

- $\mu(\mathbf{x}, \theta)$: The physics-based mean function (Finite Element surrogate).
- $\mathbf{K}(\mathbf{x}, \theta)$: The GP covariance matrix representing emulator uncertainty (bias).
- ${}_\epsilon$: Measurement noise.

The `model_n_hv` Likelihood (Dual Output)

The `model_n_hv` function is unique because it performs **Multi-Output Inference**. It evaluates the likelihood of observing both Horizontal (H) and Vertical (V) extensions simultaneously for a given set of material parameters.

A. Conditional Independence

The model assumes that given the true material parameters θ and the GP emulator state, the residuals for Horizontal and Vertical data are independent. The total log-likelihood is the sum of the individual log-likelihoods:

$$\log P(\mathbf{y}_h, \mathbf{y}_v|\theta) = \log P(\mathbf{y}_h|\theta) + \log P(\mathbf{y}_v|\theta)$$

In the code:

```
# Sample Horizontal
numpyro.sample("data_h", dist.MultivariateNormal(loc=mean_vector_h, covariance_matrix=cov_ma

# Sample Vertical
numpyro.sample("data_v", dist.MultivariateNormal(loc=mean_vector_v, covariance_matrix=cov_ma
```

B. Physical Coupling (The "Why")

Although the likelihood statements are separate, the inference is **strongly coupled** via the shared parameters:

1. **Shared Physics (θ)**: The same $E_1, E_2, ...$ must explain both the shear deformation (H) and the normal deformation (V).
   - $\mu_h = \text{Emulator}(\theta) \cdot P\sin(\alpha)$
   - $\mu_v = \text{Emulator}(\theta) \cdot P\cos(\alpha)$
   - *Effect*: This constrains the parameter space significantly. A parameter set that fits H well but V poorly will have a low total likelihood.
2. **Shared Emulator Structure (K)**:
   - The covariance matrix $\mathbf{K}$ is identical for both outputs.
   - *Effect*: The model assumes the "smoothness" and "scale" of the model discrepancy is similar for both directions.
3. **Shared Noise Model**:
   - A single `sigma_measure` is used for both.
   - *Effect*: Assumes similar sensor noise characteristics for both measurement types.

3. Code Implementation Details

Centered vs. Non-Centered Parameterization

Centered (`model`):

```
# Harder for MCMC to sample if geometry is complex
x = numpyro.sample("x", dist.Normal(mu, sigma))
```

Non-Centered (`model_n`, `model_n_hv`):

```
# Easier for MCMC (Standard Normal)
x_n = numpyro.sample("x_n", dist.Normal(0, 1))
x = mu + sigma * x_n
numpyro.deterministic("x", x)
```

Why? This decouples the dependency between the mean/scale and the sample value, avoiding “funnel” pathologies in the posterior geometry, leading to more efficient sampling.

Directional Logic

- **`model`**: Hardcoded to $\sin(\alpha)$ (Horizontal).
- **`model_n`**: Checks `if direction == 'h'` to choose between sin and cos.
- **`model_n_hv`**: Computes **both** and samples **both**.

```
# model_n_hv logic
mean_vector_h = mean_emulator * input_xy[:,0] * jnp.sin(input_xy[:,1])
mean_vector_v = mean_emulator * input_xy[:,0] * jnp.cos(input_xy[:,1])
```