Estimating cosmological parameters from galaxy power spectrum using `BayesFlow`

— Project Report —

Simulation Based Inference

Maria Levina, 230579

Rahul Vishwkarma, 236862

January 11, 2026

TU Dortmund University

Contents

1 Introduction

In modern engineering and scientific research, computational models are increasingly used to simulate complex physical systems whose behavior cannot be easily observed or measured experimentally. Such simulations, often referred to as computer experiments, represent deterministic mappings between input parameters and output responses generated by numerical codes. However, high-fidelity computer models can be computationally expensive, requiring significant time and resources for each simulation run.

As a case study, the project investigates the mechanical response of a Sun Sail structure, consisting of membrane, truss, edge cables, and support cables modeled via finite element analysis method using Kratos Multi-physics. Softare The Sun Sail experiences external loads such as snow pressure while its material properties (e.g., Young's modulus, Poisson's ratio, pre-stress values) are subject to stochastic variability. The primary output quantity is the maximum Cauchy stress within the membrane~~—~~, used as a failure criterion when exceeding rupture stress thresholds.

Given the large number of uncertain parameters and the limited computational budget ~~($n_{\max} \ll 200$ runs)~~, constructing an efficient surrogate model becomes essential. Kriging, also known as Gaussian Process regression, provides a probabilistic interpolation technique capable of predicting code outputs at ~~untested~~ unseen parameter combinations while quantifying the associated prediction uncertainty. By training this surrogate model on strategically selected design points obtained from computer experiments, one can perform sensitivity analysis and uncertainty propagation with minimal additional simulation cost.

This report presents the theoretical background of computer experiments and Kriging techniques applied for uncertainty quantification. It outlines the mathematical formulation of Gaussian Process models, describes experimental design strategies for selecting optimal simulation points, and demonstrates how Kriging approximations can replicate finite element results with quantified confidence intervals for the Sun Sail case study.

2 Problem definition

2.1 Dataset Overview

The dataset used in this study was generated through a planned computational experiment ~~rather than physical measurements or surveys~~. Each data point corresponds to one simulation performed using the *Kratos Multiphysics* software, which employs the finite element method (FEM) to compute mechanical responses of a sun sail structure under specified conditions. Because Kratos simulations are computationally expensive, it was not feasible to perform an exhaustive sampling of all possible input combinations. Instead, an optimized experimental design approach, specifically an **Optimized Latin Hypercube Sampling (OLH)** method based on genetic algorithms, was implemented to select representative input samples within the ~~normalized~~ six-dimensional unit Cube ~~space~~. Each selected input configuration was then mapped to its physical range using inverse cumulative distribution functions (CDFs) corresponding to known probability distributions of ~~material and loading parameters~~. input Variable

The study uses a finite set of $n_{\max} = 200$ simulations as its sample size. This represents a planned experiment with stratified sampling across all six independent input variables. The design ensures uniform coverage across the multidimensional parameter space while maintaining statistical independence among variables. Each simulation produces one output value, the maximum Cauchy stress in the membrane, resulting in a dataset consisting of 200 paired observations (6-dimensional inputs and one scalar output).

2.2 Variable description

The structural membrane system is defined by several material and loading parameters associated with different components—namely the membrane, truss, edge cables, and support cables. Although only a subset of these parameters is treated as uncertain within the surrogate modeling and uncertainty quantification analyses, all quantities listed in Table 1 are required as input to the *Kratos Multiphysics* model, where they enter the finite element computations.

Six principal membrane-related parameters together with the pre-stress values in the edge and support cables are modeled as random variables that govern the probabilistic characterization of the structural response. The rupture stress, denoted by $\sigma_{\text{mem},y}$ (MPa), represents the material strength limit used for failure assessment. While it does not directly participate in the finite element model, this parameter is utilized during post-processing to compare against predicted maximum Cauchy stresses and classify each simulation outcome as either structurally safe or failed.

Random Variables Considered in the Model:

- **Membrane Young's modulus** E_{mem} [GPa] — quantifies the elastic stiffness of the membrane and governs its overall rigidity under tensile loading.
- **Membrane Poisson's ratio** ν_{mem} [–] — defines the ratio of transverse to longitudinal strain; controls coupling between orthogonal in-plane deformation modes.

- **Membrane thickness** t_{mem} [mm] — geometric property affecting both stress magnitude and spatial distribution; treated as deterministic in this analysis.
- **Membrane pre-stress** σ_{mem} [MPa] — isotropic initial tension applied prior to external loading; stabilizes membrane geometry and influences its deformation response.
- **Surface loading** f_{mem} [kPa] — external distributed pressure acting on the membrane surface; represents the primary excitation driving structural response.
- **Rupture stress** $\sigma_{\text{mem},y}$ [MPa] — material strength threshold used exclusively during post-processing for failure classification based on comparison with predicted stresses.
- **Edge cable pre-stress** σ_{edg} [MPa] — initial tension within edge cables that determines boundary stiffness and facilitates load transfer along the perimeter.
- **Support cable pre-stress** σ_{sup} [MPa] — initial tension within support cables contributing to global equilibrium and maintaining structural stability of the sail configuration.

In addition to the membrane parameters, the global structural model implemented in *Kratos Multiphysics* requires material and geometric properties for the truss, edge cables, and support cables. These quantities are essential for defining the overall structural stiffness and load-transfer characteristics but are treated as deterministic within the uncertainty analysis. The truss is characterized by its Young's modulus E_{tru} (GPa) and cross-sectional area A_{tru} (cm^2), which together determine its axial rigidity and resistance to deformation. Similarly, both the edge and support cables are defined by their respective Young's moduli (E_{edg}, E_{sup} in GPa) and diameters (d_{edg}, d_{sup} in mm); these parameters govern tensile stiffness and ensure that adequate membrane tension is maintained under loading.

It should be noted that not all variables listed in Table 1 are uncertain in the surrogate-modeling or uncertainty-quantification sense. A subset, primarily the membrane properties and cable pre-stresses, is modeled as random to capture variability in material behavior and loading conditions. The remaining quantities such as elastic moduli, cross-sectional areas, and diameters are treated as deterministic constants because they represent fixed design attributes of the structure. Nevertheless, all parameters are essential inputs for each finite element simulation carried out in *Kratos Multiphysics*. The primary model output analyzed in this study is the maximum Cauchy stress within the membrane, denoted by $\sigma_{\text{mem,max}}$ (MPa). This quantity is computed for every simulation case and serves as the key response variable for developing surrogate models and performing uncertainty propagation.

Table 1: Model input parameters for the structural membrane system.

Part	Quantity	Symbol	Unit	Distribution	Mean	Std. dev.
Membrane	Young's modulus	E_{mem}	[GPa]	Lognormal	0.60	0.09
Membrane	Poisson's ratio	ν_{mem}	[–]	Uniform	0.40	0.0115
Membrane	Thickness	t_{mem}	[mm]	Deterministic	1.00	–
Membrane	Pre-stress	σ_{mem}	[MPa]	Lognormal	4.00	0.80
Membrane	Surface loading	f_{mem}	[kPa]	Gumbel	0.40	0.12
Membrane	Rupture stress[1]	$\sigma_{\text{mem},y}$	[MPa]	Lognormal	11.00	1.65
Truss	Young's modulus	E_{tru}	[GPa]	Deterministic	205.00	–
Truss	Cross-sectional area	A_{tru}	[cm^2]	Deterministic	25.00	–
Edge cable	Young's modulus	E_{edg}	[GPa]	Deterministic	205.00	–
Edge cable	Diameter	d_{edg}	[mm]	Deterministic	12.00	–
Edge cable	Pre-stress	σ_{edg}	[MPa]	Lognormal	353.678	70.735
Support cable	Young's modulus	E_{sup}	[GPa]	Deterministic	205.00	–
Support cable	Diameter	d_{sup}	[mm]	Deterministic	12.00	–
Support cable	Pre-stress	σ_{sup}	[MPa]	Lognormal	400.834	80.166

[1] Used only for failure assessment; not included as a model parameter.

2.3 Data Quality

All data used in this study were generated through deterministic numerical simulations rather than physical experiments. Consequently, the dataset contains no missing values or measurement errors. Each simulation run yields identical results for identical input parameters, reflecting the deterministic nature of the finite element solver employed in *Kratos Multiphysics*. Minor numerical discrepancies may occur due to mesh discretization or solver convergence tolerances; however, these variations are negligible relative to the overall magnitudes of the computed responses.

The resulting dataset is therefore considered effectively noise-free and well suited for surrogate modeling applications without requiring additional preprocessing or data imputation. Overall, it constitutes high-quality synthetic experimental data that reliably represent system behavior under parameter uncertainty while remaining computationally feasible within the imposed limit of $n_{\max} = 200$ simulations.

3 Statistical Methods

3.1 Optimized Latin Hypercube Sampling

In this section, the input designs used to train and evaluate the surrogate models are constructed using an optimized Latin hypercube sampling (LHS) strategy. The goal is to obtain a space-filling design in the six-dimensional input space while keeping the total number of model evaluations limited to $n_{\max} = 200$. This restriction is necessary because each model evaluation requires running the computationally expensive finite element model Kratos [Mataix Ferrándiz et al., 2023].

Latin Hypercube (LH) Design

A Latin Hypercube (LH) can be represented by a $N \times p$ matrix L with N rows and p columns, where each column is a permutation of the integers $1, \ldots, N$. Each row L corresponds to a discrete sample point, and we can write

$$L = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1p} \\ \vdots & \ddots & \vdots \\ u_{N1} & \cdots & u_{Np} \end{bmatrix}, \tag{1}$$

where u_i denotes the i^{th} sample point. The aim is to obtain N points in the parameter space $\mathbb{R}^p$, considering the probability distributions of the parameters, and the main difficulty lies in constructing an optimized Latin hypercube.

Genetic algorithm for LH optimization

To optimize Latin hypercubes, we consider the force-based criterion

$$G(L) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{1}{\|u_i - u_j\|^2}, \tag{2}$$

which can be interpreted as the total repulsive force between sample points when they are viewed as electrically charged particles. This objective serves as the fitness function in the genetic algorithm. The genetic algorithm is a general-purpose optimization framework that iteratively improves a population of candidate solutions. The steps are implemented as follows,

Initial population: An initial population of N_{pop} random Latin hypercubes is generated, where N_{pop} is chosen to be even.

Selection: After evaluating all individuals, the best $N_{\text{pop}}/2$ Latin hypercubes, i.e. those with the lowest values of $G(L)$, are retained as survivors and the remaining half is discarded.

Crossover: From the $N_{\text{pop}}/2$ survivors, a new set of N_{pop} children is produced. The current best Latin hypercube is placed as child number 1 and child number $N_{\text{pop}}/2 + 1$. For each survivor with index $k \in \{2, \dots, N_{\text{pop}}/2\}$, two children are generated:

- The first child is obtained by taking the best Latin hypercube and replacing one randomly chosen column with the corresponding column from the kth survivor; this child is assigned index k.
- The second child is obtained by taking the kth survivor and replacing one randomly chosen column with the corresponding column from the best Latin hypercube; this child is assigned index $N_{\text{pop}}/2 + k$.

Mutation: Mutation is applied to all children except the best Latin hypercube (the first child). For each column of a Latin hypercube, a random number uniformly distributed in $[0, 1]$ is drawn; if this number is less than a prescribed threshold p_{mut}, two randomly chosen elements within that column are swapped. This preserves the Latin hypercube structure while introducing diversity into the population.

The selection, crossover, and mutation steps are repeated until the stopping criterion based on the accumulated improvement of $G(L)$ over a fixed number of generations is met, or until a maximum number of generations is reached.

Mapping to the Physical Input Space

The optimized LHS design L yields N points in the unit hypercube $[0, 1]^6$. However, the physical input variables are modeled as random variables with prescribed, marginal distributions. To transform the optimized design into physically meaningful input samples, the inverse cumulative distribution function (CDF) is applied. Specifically, for each input variable $k = 1, \dots, 6$ and each design point $m = 1, \dots, N$, the physical realization $x_{k,m}$ is obtained as below,

$$x_{k,m} = F_k^{-1}(u_{k,m}) \,, \tag{3}$$

where $u_{k,m} \in (0, 1)$ denotes the corresponding normalized LHS coordinate and F_k is the CDF of the k-th input variable. This transformation ensures that each column of the resulting design matrix $X \in \mathbb{R}^{N\times 6}$ follows the prescribed marginal distribution while preserving the optimized space-filling properties of the LHS design.

3.2 Surrogate Models

3.2.1 Support Vector Regression

Support Vector Regression (SVR) seeks a function that balances prediction accuracy and model flatness by minimizing a regularized objective function [Awad & Khanna, 2015]. SVR employs an ε-insensitive loss, which ignores errors with absolute value below ε and penalizes only larger deviations. The parameter ε defines the width of an ε-tube around the regression function: decreasing ε narrows the tube, increases the number of support vectors, and reduces sparsity, whereas increasing ε widens the tube and typically yields fewer support vectors. Because small perturbations inside the tube are not penalized, the ε-insensitive region enhances robustness to noise. Alternative convex loss functions, such as linear, quadratic, and Huber losses, may also be used; the Huber loss is smoother and combines robustness for small errors with stronger penalties for large deviations. The choice of loss depends on prior knowledge of the noise affecting the data, the desired sparsity of the solution, and computational considerations.

In the linear ε-insensitive SVR formulation, the optimization problem is

$$\min_{w,b,\xi_i,\xi_i^*} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right), \tag{4}$$

subject to

$$\begin{aligned} y_i - w^\top x_i - b &\le \varepsilon + \xi_i, & i &= 1,\dots,N,\\ w^\top x_i + b - y_i &\le \varepsilon + \xi_i^*, & i &= 1,\dots,N,\\ \xi_i, \xi_i^* &\ge 0, & i &= 1,\dots,N. \end{aligned} \tag{5}$$

Here, w and b define the regression parameter , ξ_i and ξ_i^* are slack variables measuring violations of the ε-tube, and $C > 0$ is a regularization parameter that controls the trade-off between minimizing the norm of w (model flatness) and penalizing prediction errors.

Kernel SVR and nonlinear functions

The linear SVR formulation in (8)–(9) assumes that the regression function $f(x)$ is linear in the input space. For nonlinear relationships, SVR maps the inputs into a (typically higher-dimensional) feature space via a transformation $\varphi(x)$ and seeks a function of the form

$$f(x) = w^\top \varphi(x) + b. \tag{6}$$

Directly working with $\varphi(x)$ is usually impractical, so SVR employs the kernel trick: inner products $\varphi(x_i)^\top \varphi(x_j)$ are replaced by a positive definite kernel function $k(x_i, x_j)$ satisfying Mercer's condition. This leads to the dual formulation, in which the solution depends on the training data only through kernel evaluations.

In the kernel SVR setting, the regression function can be written as

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\, k(x^i, x) + b, \tag{7}$$

where $k(\cdot,\cdot)$ is the chosen kernel, and α_i, α_i^* are dual variables obtained from the SVR dual optimization problem. Only data points with nonzero $\alpha_i - \alpha_i^*$ (the support vectors) contribute to the prediction. Typical choices for $k(\cdot,\cdot)$ include linear, polynomial, and radial basis function (RBF) kernels, whose hyperparameters, together with C and ε, are selected by cross-validation.

3.2.2 Support Vector Regression

Support Vector Regression (SVR) seeks a function that balances prediction accuracy and model flatness by minimizing a regularized objective function [Awad & Khanna, 2015]. SVR employs an ε-insensitive loss, which ignores errors with absolute value below ε and penalizes only larger deviations. The parameter ε defines the width of an ε-tube around the regression function: decreasing ε narrows the tube, increases the number of support vectors, and reduces sparsity, whereas increasing ε widens the tube and typically yields fewer support vectors. Because small perturbations inside the tube are not penalized, the ε-insensitive region enhances robustness to noise. Alternative convex loss functions, such as linear, quadratic, and Huber losses, may also be used; the Huber loss is smoother and combines robustness for small errors with stronger penalties for large deviations. The choice of loss depends on prior knowledge of the noise affecting the data, the desired sparsity of the solution, and computational considerations.

In the linear ε-insensitive SVR formulation, the optimization problem is

$$\min_{\beta,\xi_i,\xi_i^*} \quad \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right), \tag{8}$$

subject to

$$\begin{aligned} y_i - \beta^\top x_i &\le \varepsilon + \xi_i, && i = 1, \dots, n, \\ \beta^\top x_i - y_i &\le \varepsilon + \xi_i^*, && i = 1, \dots, n, \\ \xi_i, \xi_i^* &\ge 0, && i = 1, \dots, n. \end{aligned} \tag{9}$$

Here, $x_i \in \mathbb{R}^{p+1}$ denotes the augmented input vector, whose first component is fixed to 1 so that it captures the intercept term, while the remaining p components correspond to the original covariates. In this parametrization, the coefficient vector $\beta \in \mathbb{R}^{p+1}$ includes the intercept β_0 as its first entry and p additional coefficients β_1,β_2,...,β_p for the covariates.

Kernel SVR and nonlinear functions

The linear SVR formulation in (8)–(9) assumes that the regression function $f(x)$ is linear in the input space, where $x \in \mathbb{R}^{p+1}$. For nonlinear relationships, SVR maps the inputs into a (typically higher-dimensional) feature space via a transformation $\varphi(x)$ and seeks a function of the form

$$f(x) = \beta^\top \varphi(x), \tag{10}$$

where the first component of $\varphi(x)$ is typically set to 1 so that the corresponding entry of β plays the role of the intercept. Also note that now β has the same dimension as $\varphi(x)$. Directly working with $\varphi(x)$ is usually impractical, so SVR employs the kernel trick: inner products $\varphi(x_i)^\top \varphi(x_{i'})$ are replaced by a positive definite kernel function $k(x_i, x_{i'})$ satisfying Mercer's condition. This leads to the dual formulation, in which the solution depends on the training data only through kernel evaluations.

In the kernel SVR setting, the regression function can be written as

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\, k(x_i, x), \tag{11}$$

where $k(\cdot, \cdot)$ is the chosen kernel, and α_i, α_i^* are dual variables obtained from the SVR dual optimization problem. Only data points with nonzero $\alpha_i - \alpha_i^*$ (the support vectors) contribute to the prediction. Typical choices for $k(\cdot, \cdot)$ include linear, polynomial, and radial basis function (RBF) kernels, whose hyperparameters, together with C and ε, are selected by cross-validation.

3.2.3 Polynomial Regression

The polynomial regression model generalizes linear regression by augmenting it with polynomial functions of the predictors while remaining linear in the coefficients. For $x \in \mathbb{R}$, a polynomial model of degree d is written as

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_d x^d + \varepsilon,$$

where y is the response variable, x is the independent variable, β_0 is the intercept, $\beta_1, \ldots, \beta_d$ are the coefficients of the polynomial terms, ε is a random error term (typically assumed to have mean zero and constant variance σ^2), and d denotes the degree of the polynomial.

In the multivariate setting with p predictors $x_1, x_2, \ldots, x_p$, polynomial regression includes higher powers of each variable as well as interaction terms (products of different predictors) up to the chosen degree d. A general polynomial model of degree d can be expressed schematically as

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j=i}^{p} \beta_{ij} x_i x_j + \sum_{i=1}^{p} \sum_{j=i}^{p} \sum_{k=j}^{p} \beta_{ijk} x_i x_j x_k + \cdots + \beta_{12\cdots p} x_1 x_2 \cdots x_p + \varepsilon,$$

where β_0 is the intercept, β_i are coefficients for the linear terms, β_{ij} correspond to quadratic terms and two-way interactions, β_{ijk} to cubic terms and three-way interactions, and so forth up to degree d. The error term ε again represents random noise. This construction allows the model to capture complex nonlinear relationships among the predictors and the response while preserving the linear-in-parameters structure needed for standard least-squares estimation.

3.2.4 Polynomial Regression TU

Polynomial regression extends the classical linear regression model by incorporating polynomial terms of the predictor variables to capture nonlinear relationships between inputs and responses. For a single independent variable x, a polynomial regression model of degree d can be expressed as:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_d x^d + \varepsilon,$$

where:

- y is the dependent (response) variable to be predicted,
- x is the independent (predictor) variable,
- β_0 is the intercept, representing the value of y when $x = 0$,
- β_1, β_2, …, β_d are regression coefficients corresponding to each polynomial term,
- ε is the random error term, assumed to follow a normal distribution with mean zero and constant variance (σ^2),
- d denotes the degree of the polynomial, i.e., the highest power of x included in the model.

When multiple predictor variables are present, polynomial regression generalizes by including not only powers of individual variables but also all possible interaction products among them up to a chosen degree. This enables modeling of complex dependencies between predictors and response variables.

For p independent variables $x_1, x_2, \ldots, x_p$, a general polynomial regression model of degree d can be written as:

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j=i}^{p} \beta_{ij} x_i x_j + \sum_{i=1}^{p} \sum_{j=i}^{p} \sum_{k=j}^{p} \beta_{ijk} x_i x_j x_k + \cdots + \beta_{12\cdots p} x_1 x_2 \cdots x_p + \varepsilon,$$

where:

- p is the total number of predictor variables,
- coefficients such as β_i, β_{ij}, β_{ijk}, etc., correspond respectively to linear, quadratic (two-way interaction), and cubic (three-way interaction) terms,
- higher-order terms allow increased flexibility for approximating nonlinear trends in data.

Although true physical systems may not strictly follow polynomial relationships, appropriately chosen low-degree polynomials often provide accurate approximations while maintaining interpretability and computational efficiency.

3.2.5 Gaussian processes Regression

Gaussian process (GP) regression admits different but equivalent interpretations, most notably the weight-space and function-space views. In this report, the *function-space* perspective is adopted.

3.2.6 Function-space view of Gaussian processes

A function-space perspective on Gaussian process regression can be obtained by formulating the model directly in function space. In this view, a Gaussian process (GP) defines a prior distribution over functions. A Gaussian process is defined as a collection of random variables such that any finite subset of them has a joint multivariate Gaussian distribution. It is fully characterized by a mean function and a covariance (kernel) function.

For $x \in \mathbb{R}^{p+1}$, a real-valued stochastic process $f(x)$, the mean function $m(x)$ and covariance function $k(x, x')$ are given by

$$m(x) = \mathbb{E}[f(x)], \qquad k(x, x') = \mathbb{E}\big[(f(x) - m(x))(f(x') - m(x'))\big],$$

and the process is denoted compactly as

$$f(x) \sim \mathcal{GP}\big(m(x), k(x, x')\big).$$

For notational convenience, the mean function $m(x)$ is often set to zero in applications, although non-zero mean functions can also be used when appropriate.

Prediction with noise-free observations

In practice, the goal of Gaussian process regression is not merely to draw sample functions from the prior, but to update this prior using information contained in the training data. In the simplest case of noise-free observations, we assume $y_i = f_i$ that the pairs $\{(x_i, f_i)\}_{i=1}^n$ are known exactly. Collecting the corresponding function values into a vector f, and denoting by f_* the function values at a set of test inputs X_*, the joint prior distribution of f and f_* is multivariate Gaussian,

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right),$$

where $K(X, X)$ is the $n \times n$ covariance matrix over the training inputs, $K(X, X_*)$ is the $n \times n_*$ matrix of covariances between training and test inputs, and $K(X_*, X_*)$ is the $n_* \times n_*$ covariance matrix over the test inputs.

To obtain the posterior distribution over function values at the test points, we condition this joint Gaussian prior on the observed training outputs f. The resulting conditional distribution is

$$f_* \mid X_*, X, f \sim \mathcal{N}\big(\mu_*, \ \Sigma_*\big),$$

with posterior mean and covariance given by

$$\mu_* = K(X_*, X)\, K(X, X)^{-1} f,$$

$$\Sigma_* = K(X_*, X_*) - K(X_*, X)\, K(X, X)^{-1} K(X, X_*).$$

Sampling function values at the test inputs then amounts to evaluating μ_* and Σ_* and drawing from this multivariate normal distribution. This formulation extends directly to multidimensional inputs by evaluating the covariance function on vector-valued arguments; the algebra is unchanged, although visualizing the resulting sample functions becomes more challenging in higher dimensions.

Prediction with Noise-free Observations TU In many practical applications, we are not primarily interested in sampling random functions from the Gaussian process prior but rather in updating our knowledge about the underlying function based on available training data. To illustrate this concept, consider the special case of **noise-free observations**, where each training input x_i is associated with a known output value f_i, i.e.,

$$\{(x_i, f_i) \mid i = 1, \dots, n\}.$$

Under the Gaussian process prior assumption, the joint distribution of training outputs f and test outputs f^* can be expressed as:

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right),$$

where: - $K(X,X)$ is the covariance matrix among all n training inputs, - $K(X,X^*)$ is an $n \times n^*$ matrix containing covariances between training and test inputs, - $K(X^*,X)$ and $K(X^*,X^*)$ are defined analogously for test points.

To obtain predictions consistent with observed data, we condition this joint prior on the known training values. This conditioning operation yields the posterior distribution over possible functions:

$$f^*|X^*, X, f \sim N\left(K(X^*,X)\,K(X,X)^{-1}\,f,\; K(X^*,X^*) - K(X^*,X)\,K(X,X)^{-1}\,K(X,X^*)\right).$$

The posterior mean represents our best estimate of function values at new test locations (X^*), while the posterior covariance quantifies prediction uncertainty. Function samples can be drawn directly from this multivariate normal distribution using its mean and covariance matrices.

This formulation provides a mathematically efficient way to incorporate deterministic observations into Gaussian process modeling without explicitly rejecting inconsistent prior samples. Although visualization becomes more challenging for multidimensional input spaces, extending these computations simply requires evaluating covariance functions according to their multidimensional form—typically through kernel expressions such as those defined by equation (2.16).

Prediction with Noisy Observations

For many practical modelling problems, the true function values are not directly observed; instead, only noisy measurements of the form $y = f(x) + \varepsilon$ are available, where ε denotes additive noise. Assuming that this noise is independent, identically distributed Gaussian with variance σ_n^2, the prior over the noisy observations has covariance

$$\operatorname{cov}(y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq},$$

or, in matrix form,

$$\operatorname{cov}(y) = K(X, X) + \sigma_n^2 I,$$

where δ_{pq} is the Kronecker delta, equal to one if $p = q$ and zero otherwise, and I is the identity matrix.

Under this noise model, the joint prior distribution of the noisy training outputs and the latent function values at test inputs can be written as

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right).$$

Conditioning on the observed data yields the Gaussian process regression predictive distribution

$$f_* \mid X, y, X_* \sim \mathcal{N}\big(\bar{f}_*,\ \operatorname{cov}(f_*)\big),$$

with mean

$$\bar{f}_* \mathbb{E}[f_* \mid X, y, X_*] = K(X_*, X)\big[K(X, X) + \sigma_n^2 I\big]^{-1} y,$$

and covariance

$$\operatorname{cov}(f_*) = K(X_*, X_*) - K(X_*, X)\big[K(X, X) + \sigma_n^2 I\big]^{-1} K(X, X_*).$$

3.3 RBF

In this chapter, the squared exponential (SE) covariance function is used as the main illustrative example, with alternative covariance functions introduced later. The covariance function specifies the covariance between any pair of function values through their inputs,

$$\operatorname{cov}\big(f(x_p), f(x_q)\big) \;=\; k(x_p, x_q) \;=\; \exp\!\big(-\tfrac{1}{2}\,\|x_p - x_q\|^2\big)\,.$$

Here, the covariance between outputs is expressed explicitly as a function of the corresponding input locations. For this particular kernel, points whose inputs are very close in the input space have covariance close to one, and the covariance decays smoothly as the distance between inputs increases.

It can be shown that the SE covariance function is equivalent to a Bayesian linear regression model with an infinite number of basis functions. More generally, any positive definite covariance function $k(\cdot, \cdot)$ admits (possibly infinite) expansions in terms of basis functions, as formalized by Mercer's theorem. In the specific case of the SE kernel, one can obtain this covariance function by considering a linear combination of infinitely many Gaussian-shaped basis functions.

3.4 Specification of the Tools Used

All computational and statistical analyses in this project were carried out using open-source software tools within the Python programming environment. The following packages and frameworks were employed:

- **Kratos Multiphysics:** A high-fidelity finite element analysis (FEA) framework used to compute structural responses, specifically the maximum Cauchy stress values of the sun sail membrane under varying material and loading conditions. Kratos provided deterministic simulation outputs that served as ground truth data for surrogate model training.
- **Python Programming Language:** Version 3.12.10 was used as the primary environment for data processing, surrogate modeling, and visualization. All computations were performed on a standard workstation with multi-core CPU support.
- **Scikit-learn Library (`sklearn`):** Utilized for implementing machine learning algorithms including Support Vector Regression (`SVR`), Polynomial Regression (`PolynomialFeature`
and `LinearRegression`), and Gaussian Process Regression (`GaussianProcessRegressor`). These modules provide reliable optimization routines and built-in cross-validation capabilities.
- **NumPy and SciPy:** Used for numerical operations such as matrix manipulation, array handling, linear algebraic computations, kernel evaluations, and statistical analysis.
- **Matplotlib and Seaborn:** Employed for plotting regression results, residual distributions, correlation matrices, and uncertainty visualizations to aid interpretation of model performance.
- **Statistical Tables and Metrics:** Standard performance metrics mean squared error (MSE) were computed using Scikit-learn's evaluation functions. No external statistical tables were required due to deterministic data generation from Kratos simulations.

All tools were selected for their reproducibility, compatibility with open-source scientific workflows, and proven reliability in engineering computation contexts.

3.5 Model Assumptions and Verification

The surrogate modeling framework developed in this project is based on several fundamental assumptions regarding both data characteristics and model behavior. These assumptions are essential to ensure valid inference from trained models; each assumption was either theoretically justified or empirically verified through diagnostic analysis.

1. **Deterministic Data Assumption:** The FEM simulations performed by Kratos Multiphysics are deterministic—identical inputs yield identical outputs without random noise. This allows surrogate models (SVR, Polynomial Regression, GPR) to be trained under noise-free conditions where interpolation rather than smoothing is appropriate.

2. **Independence of Input Variables:** The six input parameters (material properties and load conditions) were sampled independently using optimized Latin Hypercube design. Independence was ensured by construction since each variable's sampling distribution was defined separately before transformation via inverse cumulative distribution functions (CDFs).

3. **Homoscedasticity (Constant Variance):** Polynomial regression assumes constant variance of residuals across predicted values. This assumption was evaluated by plotting residuals versus fitted predictions; absence of systematic patterns confirmed approximate homoscedasticity.

4. **Normality of Residuals:** For least-squares estimators in polynomial regression to remain unbiased with valid confidence intervals, residuals should follow a normal distribution centered at zero. Normality was checked visually through Q–Q plots between standardized residuals and theoretical quantiles; minor deviations near tails were acceptable given deterministic simulation data.

5. **Stationarity in Gaussian Process Regression:** GPR assumes that covariance between points depends only on their relative distance rather than absolute position—implying stationarity. This property was examined by inspecting kernel behavior $k(x_i, x_j)$ over sample distances; consistent decay patterns validated stationary covariance structure.

6. **Kernel Smoothness Assumption (GPR):** The chosen Radial Basis Function kernel presumes smooth underlying relationships between inputs and outputs. Given continuous mechanical behavior simulated via FEM equations, this assumption is physically justified.

7. **ε-Insensitive Loss Function Validity (SVR):** SVR relies on an ε-margin around training points where small errors are ignored to achieve generalization balance. Since Kratos outputs exhibit no stochastic variation but nonlinear trends across parameter space, applying a small ε ensures stable interpolation without unnecessary penalization.

   Verification involved comparing predicted versus actual stresses across test samples; consistent alignment along the diagonal line indicated valid margin settings.

   Cross-validation techniques further supported these assumptions by evaluating model stability under different train-test splits; high R^2 values (¿0.99) confirmed robustness without overfitting.

Overall, all underlying assumptions hold true within this computational context because: (i) simulations are deterministic, (ii) sampling ensures independence among variables and (iii) diagnostic plots confirm appropriate statistical behavior. Therefore, the surrogate models built upon these foundations provide reliable approximations for expensive finite element evaluations while maintaining mathematical validity across all tested configurations.

References

Awad, M., & Khanna, R. [2015]. *Efficient learning machines: Theories, concepts, and applications for engineers and system designers* 1st ed. Apress Berkeley, CA. https://doi.org/10.1007/978-1-4302-5990-9

Mataix Ferrándiz, V., Bucher, P., Zorrilla, R., Rossi, R., Cornejo, A., Celigueta, M. A., Roig, C., Masó, M., Warnakulasuriya, S., Casas, G., Nuñez, M., Dadvand, P., Latorre, S., de Pouplana, I., Irazábal González, J., Arrufat, F., Chandra, B., Geiser, A., Sautter, K. B., . . . Garáte, J. [2023]. *Kratosmultiphysics/kratos: Release 9.3* Version 9.3. Zenodo. https://doi.org/10.5281/zenodo.7681287