**Eberhard Karls Universität Tübingen**

Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

**Bachelor Thesis Cognitive Science**

# Exploring Action Policies in Iterative Probabilistic PDE Solvers

Leon Fuß

25.08.2025

**Reviewer**

Prof. Dr. Philipp Hennig
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

**Supervisor**

Tim Weiland
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen

**Fuß, Leon** - 6047384
*Exploring Action Policies in Iterative Probabilistic PDE Solvers*
Bachelor Thesis Cognitive Science
Eberhard Karls Universität Tübingen
Period: 24.04.2025 - 25.08.2025

# Abstract

Partial differential equations (PDEs) govern critical physical phenomena across various fields of science and engineering. Their numerical solution traditionally provides only point estimates, without principled quantification of uncertainty. Recent advances in probabilistic numerical methods—specifically physics-informed Gaussian processes and the IterGP framework—enable uncertainty-aware PDE solving. However, these approaches typically rely on generic action policies that ignore the rich mathematical structure of PDEs.

This thesis develops structure-informed action policies that systematically exploit PDE characteristics for more efficient uncertainty reduction in iterative probabilistic solvers. We identify two key structural properties of PDEs: spectral decomposition into eigenmodes and temporal causality in time-dependent problems. Leveraging these insights, we propose three novel action policies. (1) **Continuous Adaptive Policy** uses FFT-based spectral analysis to target energetic frequency components. (2) **Time Slice Policy** exploits temporal causality by prioritising time slices of problem domain. (3) **Temporal First Policy** combines systematic temporal frequency targeting with spatial refinement in a two-phase approach.

A comprehensive evaluation of heat and wave equations across diverse parameter configurations reveals that exploiting temporal structure achieves significantly improved uncertainty quantification in fewer iterations compared to conjugate gradient baselines. The **Temporal First Policy** proves most effective while maintaining acceptable mean convergence. These results validate the core premise: principled exploitation of PDE mathematical structure brings substantial computational efficiency gains in probabilistic numerical methods.

Our work advances probabilistic numerics, demonstrating how domain-specific knowledge can be systematically incorporated into iterative algorithms. This opens pathways for structure-aware computational tools that respect the mathematical foundations underlying physical phenomena.

# Acknowledgements

I would like to express my deepest gratitude to my supervisor, Tim Weiland, without whom this work would not have been possible. His patient guidance and unwavering support proved invaluable, particularly during those moments when the topic seemed too fast to grasp. Tim always had time for me, approaching every question with an open ear and mind. I could not have asked for a better supervisor.

I am also grateful to Prof. Philipp Hennig for his insightful constributions during our meetings,which greatly enhanced my understanding of the topic. He guided me to consider important aspects I hadn't initially recognised, which greatly strengthened the work.

Finally, I extend my appreciation to my family and girlfriend, whose motivation and encouragement kept me focused, especially during those periods when I had lost sight of my thesis.

Their collective support made this thesis possible.

# Contents

## 1. Introduction

Partial differential equations (PDEs) form the mathematical foundation for understanding physical phenomena across various scientific and engineering disciplines [1]. These range from climate modelling to thermal management in electronics. Analytical solutions exist only for a narrow class of problems with simple geometries and boundary conditions. Real-world applications require numerical methods. These transform continuous PDE problems into finite-dimensional algebraic systems, trading mathematical exactness for computational feasibility. Traditional numerical PDE solvers face a critical limitation: they provide only point estimates without principled uncertainty quantification. In safety-critical applications—such as bridge design and drug delivery systems —engineers need not just predictions, but reliable confidence bounds that quantify computational trustworthiness.

**Probabilistic PDE Solving and Computational Challenges**

Probabilistic numerical methods address this gap by treating numerical computation as statistical inference. Physics-informed Gaussian processes model unknown solutions as random functions and condition prior beliefs on PDE constraints, boundary conditions, and measurements. This framework naturally yields posterior uncertainty estimates and leverages the mathematical fact that GPs remain Gaussian under conditioning on linear functionals [2].

However, implementation reveals a computational bottleneck: conditioning requires operations with the Gram matrix $\boldsymbol{G} = \mathcal{L}k\mathcal{L}^T$, where $\mathcal{L}$ denotes PDE constraints and $k$ represents the prior covariance. For practical discretisations, direct inversion incurs prohibitive $\mathcal{O}(N^3)$ complexity [3].

The IterGP framework addresses this challenge using iterative uncertainty quantification. Instead of computing exact representer weights $\boldsymbol{v}^* = \boldsymbol{G}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$, IterGP treats these as latent variables and iteratively refines beliefs through tractable matrix-vector operations. Each iteration computes a scalar observation $\alpha_i = \boldsymbol{s}_i^T \boldsymbol{r}_{r-1}$ in a direction specified by action an vector $\boldsymbol{s}_i$, enabling exact Bayesian updates via rank-one precision modifications [4].

**The Action Policy Challenge**

This shift transforms the computational challenge from matrix inversion to strategic action selection. The sequence $\{s_i\}$ determines which solution components receive computational attention, governing convergence and uncertainty reduction patterns. However, existing action policies such as conjugate gradients, Cholesky decomposition, and random selection operate generically without exploiting PDE structure. They treat $\boldsymbol{G}$ as an arbitrary symmetric positive definite matrix, ignoring its origin from physical phenomena with well-understood spectral properties.

**Research Contributions**

This thesis examines whether structure-informed action policies achieve faster uncertainty reduction than generic approaches. PDEs possess rich structural properties: spectral decomposition into eigenmodes and temporal causality in time-dependent problems. Three novel policies are developed to exploit these characteristics:

1. **Continuous Adaptive Policy**: Fast Fourier Transform (FFT)-based spectral analysis targeting energetic frequency components
2. **Time Slice Policy**: Temporal causality awareness through spatial location prioritisation
3. **Temporal First Policy**: Two-phase approach combining systematic temporal frequency targeting with spatial refinement

A comprehensive evaluation of heat and wave equations demonstrates that temporal structure exploitation achieves significant uncertainty quantification in fewer iterations compared to the conjugate gradient baselines, validating principled PDE structure exploitation for efficient probabilistic numerical methods.

# 2. Background

## 2.1. Partial Differential Equations

### 2.1.1. From Physical Intuition to Discrete Formulation

This derivation follows Sanderson's approach to partial differential equations [5], which intuitively connects discrete physical reasoning to continuous mathematics.

Consider the fundamental problem of heat conduction in a one-dimensional rod. Rather than beginning with abstract differential operators, we start with a discrete approximation that captures the essential physics. Imagine the rod divided into discrete points $x_1, x_2, ..., x_n$ with corresponding temperatures $T_1(t), T_2(t), ..., T_{m(t)}$.

The key physical idea is simple: heat moves from warmer to cooler regions. At any interior point $i$, the temperature change rate depends on how $T_i$ compares to neighbours $T_{i-1}$ and $T_{i+1}$.

If the neighbours' average temperature exceeds the current temperature:

$$\frac{T_{i-1} + T_{i+1}}{2} > T_i,$$

then $T_i$ should increase. The rate of change is proportional to this difference:

$$\frac{\mathrm{d}T_i}{\mathrm{d}t} = \alpha \left( \frac{T_{i-1} + T_{i+1}}{2} - T_i \right),$$

where $\alpha > 0$ is a material constant.

**Reformulation in Terms of Second Differences**

To connect this discrete formulation with continuous mathematics, we reformulate the expression above using second differences. Rearranging the right-hand side:

$$\frac{\mathrm{d}T_i}{\mathrm{d}t} = \frac{\alpha}{2}((T_{i+1} - T_i) - (T_i - T_{i-1})).$$

Define the discrete second difference operator:

$$\Delta^2 T_i := (T_{i+1} - T_i) - (T_i - T_{i-1}) = T_{i+1} - 2T_i + T_{i-1}.$$

Then our evolution equation becomes:

$$\frac{\mathrm{d}T_i}{\mathrm{d}t} = \frac{\alpha}{2} \Delta^2 T_i. \tag{1}$$

The second difference $\Delta^2 T_i$ shows how much point $i$ differs from its neighbours' average, acting as a discrete analogue of curvature.

### 2.1.2. Transition to the Continuous Case

As the spacing between discrete points vanishes, the discrete second difference approaches the second partial derivative. In the continuous limit, our temperature is a function $u(x, t)$ where:

$$\lim_{\Delta x \to 0} \frac{\Delta^2 T_i}{(\Delta x)^2} = \frac{\partial^2 u}{\partial x^2}$$

The discrete evolution (Eq. 1) correspondingly becomes:

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2},$$

where $\kappa$ is the thermal diffusivity. This is the canonical *heat equation* or *diffusion equation.*

**Mathematical Interpretation**

The heat equation expresses the principle that the temporal rate of change at any point is proportional to the spatial gradient of temperature there. The second derivative $\frac{\partial^2 u}{\partial x^2}$ quantifies how much the temperature at position $x$ deviates from the mean of its infinitesimal neighbours.

When $\frac{\partial^2 u}{\partial x^2} > 0$, the profile is curved up and temperature rises. When $\frac{\partial^2 u}{\partial x^2} < 0$, it curves down and the temperature falls.

**Extension to Higher Dimensions**

For heat conduction in two or three spatial dimensions, the same physical reasoning remains, but each point has additional neighbours. The heat equation extends to:

$$\frac{\partial u}{\partial t} = \kappa \nabla^2 u, \tag{2}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplacian operator. The Laplacian is a multidimensional extension of the second derivative, quantifying how a point's value differs from the mean of all neighbouring values [6, sec. 2.3 ].

### 2.1.3. General Linear PDE Framework

The heat equation exemplifies the abstract structure underlying all linear partial differential equations. By examining our previously derived heat equation more carefully, we can reveal how it fits into a much broader mathematical framework.

We can rearrange Eq. 2 to isolate all terms involving the unknown function $u$ on one side:

$$\frac{\partial u}{\partial t} - \kappa \nabla^2 u = 0.$$

This reformulation suggests defining an abstract differential operator, denoted as $\mathcal{D}$, that captures the left-hand side of the equation. Here, $\mathcal{D}$ represents the action of combining all the differential terms (such as derivatives with respect to time and space) acting on the unknown function $u$:

$$\mathcal{D}[u] = \frac{\partial u}{\partial t} - \kappa \nabla^2 u.$$

Our heat equation then takes the general form:

$$\mathcal{D}[u] = f$$

where $\mathcal{D} : U \to V$ is a linear differential operator between appropriate function spaces, and in our case $f = \underline{0}$ represents a homogeneous equation.

The crucial property that makes $\mathcal{D}$ a linear operator is that it satisfies the linearity condition. For any functions $u, v$ and scalars $c_1, c_2$:

$$\mathcal{D}[c_1 u + c_2 v] = c_1 \mathcal{D}[u] + c_2 \mathcal{D}[v].$$

This linearity emerges directly from the linearity of differentiation itself. Since both $\frac{\partial}{\partial t}$ and $\nabla^2$ are linear operations, any linear combination of them inherits this fundamental property [6, ch. 1].

**Wave equation**

The power of this abstract framework becomes evident when we consider fundamentally different physical phenomena. Consider wave propagation: when you pluck a guitar string or drop a pebble in a pond, disturbances propagate outward at a characteristic speed without the medium itself flowing. Unlike heat diffusion, which smooths out temperature differences, waves preserve and transport energy whilst maintaining their oscillatory character. The key insight is that wave acceleration at any point should be proportional to the local curvature, but with a crucial difference: whereas heat flows to eliminate curvature, waves use curvature to drive continued oscillation. If the spatial curvature $\nabla^2 u$ is positive (the wave height is below the average of infinitesimal surrounding points), the wave should accelerate upward. This physical reasoning leads to:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u,$$

where $c$ represents the wave speed and the second time derivative captures the oscillatory acceleration. Rearranging to match our general structure:

$$\frac{\partial^2 u}{\partial t^2} - c^2 \nabla^2 u = 0,$$

we obtain $\mathcal{D}[u] = f$ with:

$$\mathcal{D}[u] = \frac{\partial^2 u}{\partial t^2} - c^2 \nabla^2 u$$

and $f = \underline{0}$ [6, sec. 2.4].

**The Problem of Underspecification**

Having established the general form $\mathscr{D}[u] = f$, a natural question arises: How do we actually solve such equations? A moment's reflection reveals a fundamental issue—the differential equation alone provides insufficient information to determine a unique solution. Consider our heat equation $\mathscr{D}[u] = \frac{\partial u}{\partial t} - \kappa \nabla^2 u = 0$. If $u(\boldsymbol{x}, t)$ is a solution, then so is $u(\boldsymbol{x}, t) + C$ for any constant $C$. More generally, we can add any function $v(\boldsymbol{x}, t)$ that satisfies $\mathscr{D}[v] = 0$ to obtain another solution. The differential equation constrains how the solution behaves locally, but says nothing about the overall magnitude or specific values the solution must attain. This mathematical ambiguity poses a severe practical problem. In real-world applications, we seek the temperature distribution in a specific rod, the wave pattern from a particular disturbance, or the pressure field around a given obstacle. The physics demands a unique, determined solution.

**Constraining the Solution: Initial and Boundary Conditions**

The resolution lies in providing additional constraints that pin down the solution uniquely. These constraints come in two essential forms: initial conditions that specify the state at some initial time, and boundary conditions that specify the behaviour at the spatial boundaries of our domain. For time-dependent problems like the heat and wave equations, initial conditions specify the state of the system at $t = 0$:

$$u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}).$$

For second-order temporal equations like the wave equation, we additionally require the initial velocity [6, sec. 2.4]:

$$\frac{\partial u}{\partial t}(\boldsymbol{x}, 0) = v_0(\boldsymbol{x}).$$

Boundary conditions, meanwhile, constrain the solution at the spatial boundaries of our domain $\partial\Omega$. Two fundamental types prove most important for our purposes: Dirichlet Boundary Conditions specify the value of the solution on the boundary:

$$u(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \partial\Omega.$$

These represent situations where we directly control or measure the quantity of interest at the boundary—perhaps by maintaining a constant temperature or clamping a vibrating membrane. Neumann Boundary Conditions specify the normal derivative at the boundary:

$$\frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}) = h(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \partial\Omega$$

where $\boldsymbol{n}$ denotes the outward unit normal. These conditions constrain the flux or rate of change across the boundary, representing insulated surfaces in heat problems or free boundaries in wave problems. The combination of a differential equation $\mathcal{D}[u] = f$ with appropriate initial and boundary conditions forms a well-posed problem: a problem that possesses a unique solution that depends continuously on the given data [6, ch. 2].

#### 2.1.4. Numerical Discretisation: The Inevitable Compromise

Whilst our mathematical framework provides a rigorous foundation for understanding PDEs, the harsh reality is that closed-form analytical solutions exist only for a remarkably narrow class of problems—typically those with simple geometries, constant coefficients, and elementary boundary conditions. For the vast majority of practical applications, where complex domains, variable material properties, or nonlinear source terms are encountered, we are compelled to seek approximate solutions through numerical discretisation.

The fundamental principle underlying all discretisation methods is the transformation of the infinite-dimensional PDE problem into a finite-dimensional algebraic system. Rather than seeking the exact function $u(\boldsymbol{x}, t)$ defined over a continuous domain, we instead approximate this function using a finite collection of degrees of freedom — whether these represent function values at discrete points, expansion coefficients in a chosen basis, or integrated quantities over subdomains.

##### Discretisation Methodologies

Several well-established approaches exist for achieving this discretisation, each with distinct mathematical foundations and computational characteristics. Here we present two very common approaches:

**Finite Difference Methods** represent perhaps the most intuitive approach, directly approximating partial derivatives using discrete difference formulae. At each grid point $x_i$, spatial derivatives are replaced by differences quotients:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}.$$

This local approximation, when applied systematically across the computational domain, yields a sparse linear system relating the unknown function values $\{u_i\}$ [7].

**Finite Volume Methods**, which shall feature prominently in our subsequent analysis, discretise the domain into control volumes and enforce conservation laws in integral form. For each volume $V_i$, the governing equation $\mathcal{D}[u] = f$ is integrated:

$$\int_{V_i} \mathcal{D}[u]\,\mathrm{d}\boldsymbol{x} = \int_{V_i} f\,\mathrm{d}\boldsymbol{x}.$$

This approach proves particularly natural for conservation-law PDEs and maintains inherent conservation properties [7].

**The Linear System Challenge**

Regardless of the specific discretisation approach employed, all these methods share a common mathematical structure: they transform the original PDE into a large linear system of the form:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b},$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the system matrix encoding the discrete differential operator and boundary conditions, $\boldsymbol{u} \in \mathbb{R}^n$ represents the vector of unknown degrees of freedom, and $\boldsymbol{b} \in \mathbb{R}^n$ contains the discretised source terms and boundary data.

The dimension $n$ of this system scales directly with the resolution of the discretisation. For a three-dimensional problem discretised on a uniform grid with $N = 100$ points per dimension, this yields systems with one million unknowns. High-fidelity simulations routinely require resolutions exceeding $N = 1000$ producing systems with $n > 10^9$ degrees of freedom.

The computational challenge posed by such systems is substantial. Direct solution methods, whilst providing exact answers, exhibit non-linear complexity and become prohibitively expensive for large $n$. Moreover, the storage requirements for factorised forms often exceed available memory, particularly for three-dimensional problems.

## 2.2. Gaussian Processes

Having established the mathematical framework for linear PDEs and their discretisation challenges, we now introduce Gaussian processes as a principled probabilistic approach to function approximation that will prove particularly well-suited to the linear structure we have uncovered.

### 2.2.1. Intuitive Foundation

The fundamental insight underlying Gaussian processes lies in shifting perspective from parametric models — where we specify a functional form with unknown parameters — to a direct probabilistic model over the space of functions themselves. Rather than constraining our approximation to lie within a predetermined parametric family, Gaussian processes allow us to express prior beliefs about the smoothness, periodicity, and other structural properties of the unknown function $u(\boldsymbol{x})$ directly.

Consider the problem of approximating an unknown function $u : \mathcal{X} \to \mathbb{R}$ from a finite collection of observations. In the parametric approach, we might assume $u(\boldsymbol{x}) \approx \sum_{i=1}^{n} \theta_i \varphi_{i(\boldsymbol{x})}$ for some basis functions $\varphi_i$ and learn the coefficients $\theta_i$. The Gaussian process perspective instead treats the function $u$ itself as a random object, specifying a probability distribution directly over the infinite-dimensional space of functions.

### 2.2.2. Definition

A Gaussian process (GP) defines a probability distribution over functions $u : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ represents the input domain. Formally, we write:

$$u \sim \mathrm{GP}(\mu, k),$$

where $\mu : \mathcal{X} \to \mathbb{R}$ is the *mean function* and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the *covariance function* (or kernel). The defining property is that any finite collection of function evaluations follows a multivariate Gaussian distribution:

$$\begin{pmatrix} u(x_1) \\ \vdots \\ u(x_n) \end{pmatrix} \sim \mathcal{N}\left( \begin{pmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{pmatrix}, \begin{pmatrix} k(x_1, x_2) & \dots & k(x_1, k_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{pmatrix} \right).$$

This probabilistic framework suits PDE problems well because it naturally reflects the linear structure in $\mathcal{D}[u] = f$. Since GPs are closed under conditioning and differentiation is linear, applying $\mathcal{D}$ to a Gaussian process yields another Gaussian process. [3, sec. 2.2]

## 2.3. Physics Informed Gaussian Processes

Having established the mathematical framework for linear PDEs and the probabilistic foundation of Gaussian processes, we now turn to their powerful synthesis. Physics-informed Gaussian processes represent an approach to incorporating known physical laws directly into probabilistic function approximation, enabling us to harness both data and mechanistic knowledge simultaneously.

### 2.3.1. Mathematical Foundation: Closure Under Linear Functionals

The fundamental insight enabling physics-informed Gaussian processes lies in a crucial property: Gaussian processes remain Gaussian under conditioning on linear functionals of their sample paths. This mathematical foundation allows us to incorporate differential equation constraints as probabilistic observations.

Let $f \sim \mathrm{GP}(m, k)$ be an Gaussian process with sample space $\mathcal{B}$ and let $\mathcal{L} : \mathcal{B} \to \mathbb{R}^n$ be an bounded linear operator. For observation vector $\boldsymbol{y} \in \mathbb{R}^n$ and Gaussian noise $\varepsilon \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ independent of $f$, the conditional process is:

$$f \mid \mathcal{L} + \varepsilon = y \sim \mathrm{GP}\left(m_{\text{post}}, k_{\text{post}}\right)$$

with mean and convariance:

$$m_{\text{post}}(\boldsymbol{x}) = m(\boldsymbol{x}) + \mathcal{L}[k(\boldsymbol{x}, \cdot)]^\top \left(\mathcal{L}k\mathcal{L}^\top + \Sigma\right)^\dagger (\boldsymbol{y} - \mathcal{L}[m] - \boldsymbol{\mu}) \tag{3.1}$$

$$k_{\text{post}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1, \boldsymbol{x}_2) - \mathcal{L}[k(\boldsymbol{x}_1, \cdot)]^\top \left(\mathcal{L}k\mathcal{L}^\top + \Sigma\right)^\dagger \mathcal{L}[k(\cdot, \boldsymbol{x}_2)]. \tag{3.2}$$

This result, seen in [2], provides the theoretical foundation for incorporating physical constraints as linear observations of the unknown function.

Note that since differentiation is a bounded linear operation, our discretized differential operators $\mathcal{D}$ from Section 2.1.3 are precisely the types of bounded linear functionals to which this theorem applies. This connection enables us to treat PDE constraints $\mathcal{D}[u] = f$ as observations of the function $u$ through the linear operator $\mathcal{D}$.

### 2.3.2. From Finite Volume Methods to Information Operators

To bridge the gap between continuous mathematical formulation and computational implementation, we must construct finite-dimensional approximations of our infinite-dimensional PDE constraints. Here, we establish the connection between classical finite volume discretisation and the information operators that enable probabilistic PDE solving.

Recall from Section 2.1.4 that finite volume methods discretise the domain into control volumes $V_i$ and enfore conservation laws in the integral form:

$$\int_{V_i} \mathcal{D}[u]\,\mathrm{d}x = \int_{V_i} f\,\mathrm{d}x.$$

This approach can be transformed naturally to discrete observations:
Let $\{V_i\}_{i=0}^{n} \subset \Omega$ denote our control volume partition of the domain. We define the **observation operator**:

$$\mathcal{L} : \mathcal{B} \to \mathbb{R}^n, \quad \mathcal{L}[u] = \begin{pmatrix} \int_{V_1} \mathcal{D}[u]\,\mathrm{d}x \\ \vdots \\ \int_{V_n} \mathcal{D}[u]\,\mathrm{d}x \end{pmatrix}$$

which we can employ for conditioning using Eq. 3.
Note that whilst $\mathcal{D}$ and $\mathcal{L}$ may appear closely related, they serve fundamentally different mathematical roles and possess distinct domain and codomain structures. The **differential operator** $\mathcal{D} : \mathcal{B} \to \mathcal{B}$ operates entirely within the function space, transforming one function into another function. In stark contrast, the **information operator** $\mathcal{L} : \mathcal{B} \to \mathbb{R}^n$ discretises our observations, extracting finite-dimensional information from the infinite-dimensional function space. This operator bridges the gap between the continuous mathematical formulation and computational implementation by selecting specific aspects of the function that we can numerically observe and constrain. The information operator thus reduces the infinite degrees of freedom inherent in function space to the $n$ finite degrees of freedom that our computational framework can manipulate.

### 2.3.3. Unified Conditioning Framework

Real-world PDE problems involve multiple heterogeneous constraints that must be balanced simultaneously: the differential equation itself, boundary conditions, initial conditions, and potentially noisy measurements. The power of the physics-informed GP framework lies in its natural ability to combine these diverse constraint types into a unified probabilistic formulation.

#### Multiple Constraint Types

Consider $M$ distinct constraint types, each encoded by a linear operator $\mathcal{L}_j : \mathcal{B} \to \mathbb{R}^{n_j}$ for $j = 1, ..., n$. For example:

- PDE constraints: $\mathcal{L}_1[u] = \boldsymbol{f}_{\mathrm{PDE}}$

- Dirichlet boundary conditions: $\mathcal{L}_2[u] = \boldsymbol{g}_{\text{BC}}$
- Initial conditions $\mathcal{L}_3[u] = \boldsymbol{u}_0$

We can construct an aggregated operator:

$$\mathcal{L} : \mathcal{B} \to \mathbb{R}^n, \quad \mathcal{L}[u] = \begin{pmatrix} \mathcal{L}_1[u] \\ \mathcal{L}_2[u] \\ \vdots \\ \mathcal{L}_{M[u]} \end{pmatrix}$$

where $N = \sum_{j=1}^{M} n_j$ is the total number of constraints. The combined constraint becomes:

$$\mathcal{L}_{[u]} = f = \begin{pmatrix} \boldsymbol{f}_{\text{PDE}} \\ \boldsymbol{g}_{\text{BC}} \\ \vdots \\ \boldsymbol{u}_0 \end{pmatrix}$$

And the information matrix $\mathcal{L}k\mathcal{L}^T$ (so-called Gram Matrix) exhibits block structure:

$$\mathcal{L}k\mathcal{L}^T = \begin{pmatrix} \mathcal{L}^{(1)}k\left(\mathcal{L}^{(1)}\right)^T & \mathcal{L}^{(1)}k\left(\mathcal{L}^{(2)}\right)^T & \dots & \mathcal{L}^{(1)}k\left(\mathcal{L}^{(M)}\right)^T \\ \mathcal{L}^{(2)}k\left(\mathcal{L}^{(1)}\right)^T & \mathcal{L}^{(2)}k\left(\mathcal{L}^{(2)}\right)^T & \dots & \mathcal{L}^{(2)}k\left(\mathcal{L}^{(M)}\right)^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{L}^{(M)}k\left(\mathcal{L}^{(1)}\right)^T & \mathcal{L}^{(M)}k\left(\mathcal{L}^{(2)}\right)^T & \dots & \mathcal{L}^{(M)}k\left(\mathcal{L}^{(M)}\right)^T \end{pmatrix} \tag{4}$$

The diagonal blocks $\mathcal{L}^{(j)}k\left(\mathcal{L}^{(j)}\right)^T$ capture correlations within each constraint type, whilst the off-diagonal blocks $\mathcal{L}^{(i)}k\left(\mathcal{L}^{(j)}\right)^T$ encode correlations between different constraint types. This block structure proves computationally advantageous, enabling efficient matrix assembly and potentially block-diagonal preconditioning strategies. The posterior GP under combined constraints follows the same conditioning form:

$$u \mid \mathcal{L}[u] = \boldsymbol{f} \sim \text{GP}\left(m_{\text{post}}, k_{\text{post}}\right)$$

with closed form mean and variance in accordance with Eq. 3. This unified framework naturally balances the different constraint types through their relative information content, as encoded in the combined information matrix [2].

## 2.4. IterGP

### 2.4.1. The Computational Uncertainty Challenge

As established in Section 2.1.4, discretising PDEs inevitably leads to large linear systems $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$. The physics-informed GP framework from Section 2.3 transforms this challenge through probabilistic conditioning on linear observations $\mathcal{L}[u] = \boldsymbol{f}$, where $\mathcal{L}$ represents our information operator (e.g., finite volume constraints from Section 2.3.2). However, this conditioning process generates a fundamental computational bottleneck centred on what we have previously termed the information matrix $\mathcal{L}k\mathcal{L}^T \in \mathbb{R}^{n\times n}$ from Eq. 4.

**The Gram Matrix System**

The core computational challenge in GP inference lies in computing the posterior mean and covariance according to Eq. 3.1 and Eq. 3.2. This fundamentally requires operations with the Gram matrix $\boldsymbol{G} = \mathcal{L}k\mathcal{L}^T$, specifically computing $\boldsymbol{G}^{-1}\boldsymbol{y}$ to obtain the representer weights. Direct approaches to this computation exhibit $O(N^3)$ complexity, rendering exact inference computationally prohibitive for the large systems arising from high-resolution PDE discretisations.

**Approximation as Necessity**

This computational reality compels practitioners to resort to approximate methods: iterative solvers such as conjugate gradients or inducing point approximations. These approaches make GP inference computationally feasible, but introduce a critical oversight that undermines the probabilistic foundations we have carefully established.

Most approximation strategies assume the approximate posterior is the exact posterior from Eq. 3.1 and Eq. 3.2, which neglects the uncertainty introduced by computational limits and creates inconsistency. While we account for data uncertainty, we overlook the uncertainty from finite computation.

**The Computational Uncertainty Gap**

This oversight is particularly problematic for the PDE applications that motivate our work. The mathematical posterior $\mathrm{GP}(\mu_{\text{post}}, k_{\text{post}})$ is theoretically well-defined, but, due to computational constraints, we cannot compute it exactly. As a result, when we use approximate inference, we obtain a surrogate posterior that inevitably differs from the ideal, yet we typically ignore the resulting approximation error.

The resulting uncertainty estimates become unreliable precisely when accurate quantification proves most crucial — e.g., in safety-critical engineering applications where PDE simulations inform consequential decisions. Our carefully constructed probabilistic framework loses its principled foundation when computational limitations force us to abandon exact inference without proper accounting for the induced uncertainty.

### 2.4.2. Iterative Uncertainty Quantification: The IterGP Solution

Wenger et al. [4] provide an elegant resolution to the computational uncertainty challenge through their IterGP framework. Rather than attempting to compute the exact representer weights $\boldsymbol{v}^* = \boldsymbol{G}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$ at prohibitive $\mathcal{O}(N^3)$ cost, IterGP treats these weights as latent variables and iteratively refines our belief about them through tractable matrix-vector operations.

#### The Key Insight

The key insight underlying IterGP lies in recognising that we can maintain a probabilistic belief about the unknown representer weights $\boldsymbol{v}^*$ and update this belief through computations performed on the data. Instead of solving $\boldsymbol{G}\boldsymbol{v}^* = \boldsymbol{y} - \boldsymbol{m}$ exactly, we begin with a prior belief $p(\boldsymbol{v}^*) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{G}^{-1})$ and iteratively condition this belief through one-dimensional projections of the residual.

At iteration $i$, IterGP computes a single scalar observation:

$$\alpha_i = \boldsymbol{s}_i^T \boldsymbol{r}_{i-1}$$

where $\boldsymbol{s}_i \in \mathbb{R}^N$ represents the **action vector** that determines which aspect of the solution receives computational attention and $\boldsymbol{r}_{i-1} = (\boldsymbol{y} - \boldsymbol{\mu}) - \boldsymbol{G}\boldsymbol{v}_{i-1} \in \mathbb{R}^N$ is the current residual. This scalar observation can equivalently be written as:

$$\alpha_i = \boldsymbol{s}_i^T \boldsymbol{G}(\boldsymbol{v}^* - \boldsymbol{v}_{i-1})$$

revealing that we observe how much error exists in the direction specified by $\boldsymbol{s}_i$. Critically, computing $\alpha_i$ requires only a single matrix-vector multiplication with $\boldsymbol{G}$, making it computationally tractable.

#### Mathematical Framework: Rank-One Precision Updates

Each scalar observation $\alpha_i$ enables an exact Bayesian update of our belief about $\boldsymbol{v}^*$ through rank-one modifications to the precision matrix. The posterior belief after $i$ iterations becomes $p(\boldsymbol{v}^*) = \mathcal{N}(\boldsymbol{v}^*; \boldsymbol{v}_i, \boldsymbol{\Sigma}_i)$ where:

$$\boldsymbol{v}_i = \boldsymbol{v}_{i-1} + \frac{\alpha_i}{\eta_i}\boldsymbol{d}_i, \qquad \boldsymbol{v}_0 = \underline{0}$$

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{i-1} - \frac{1}{\eta_i}\boldsymbol{d}_i\boldsymbol{d}_i^T, \qquad \boldsymbol{\Sigma}_0 = G^{-1}$$

The search direction $\boldsymbol{d}_i = (I - C_{i-1}\boldsymbol{G})\boldsymbol{s}_i$ and normalisation constant $\eta_i = \boldsymbol{s}_i^T \boldsymbol{G}\boldsymbol{d}_i$ ensure that each update maintains the geometric structure necessary for convergence guarantees.

Curcially, we can equivalently track the precision matrix $\boldsymbol{C}_i = \boldsymbol{\Sigma}_i^{-1}$ through rank-one-updates:

$$\boldsymbol{C}_i = \boldsymbol{C}_{i-1} + \frac{1}{\eta_i}\boldsymbol{d}_i\boldsymbol{d}_i^T, \quad \boldsymbol{C}_0 = \underline{0}$$

This formulation reveals that $\boldsymbol{C}_i$ represents a rank $i$ approximation to $G^{-1}$, systematically improving as we incorporate more computational effort. Each iteration adds exactly one rank to our precision approximation by incorporating the scalar information $\alpha_i$.

**Algorithm Structure**

The complete IterGP algorithm (Algorithm 1 from [4]) operates by iteratively selecting action vectors $\boldsymbol{s}_i$, computing the resulting scalar observations $\alpha_i$, and updating both the representer weight estimate $\boldsymbol{v}_i$ and precision approximation $\boldsymbol{C}_i$. The combined posterior uncertainty naturally decomposes as:

$$\text{Combined Uncertainty} = \text{Mathematical Uncertainty} + \text{Computational Uncertainty}$$

where computational uncertainty quantifies how our limited iterations affect the quality of our representer weight estimates.

**Convergence Guarantees**

IterGP provides rigorous theoretical guarantees that distinguish it from heuristic approximation methods. The framework ensures that:

1. **Mean Convergence**: The posterior mean converges to the mathematical posterior mean in RKHS norm [4, Theorem 1]
2. **Exact Recovery**: After at most $N$ iterations with linearly indendent actions, we recover the exact mathematical posterior
3. **Uncertainty Bounds**: The combined uncertainty proves a tight worst-case bound on the error between the approximation posterior mean and the true latent function [4, Theorem 2]

This final property proves particularly valuable in situations where reliable uncertainty estimates are needed. Unlike traditional approximation methods that may underestimate uncertainty, IterGP's computational uncertainty component ensures we never become overconfident about solutions obtained with limited computational resources. The IterGP framework thus provides a principled foundation for maintaining probabilistic integrity whilst making GP-based PDE solving computationally tractable. Rather than abandoning uncertainty quantification in pursuit of computational efficiency, it explicitly accounts for the trade-off between computational budget and posterior confidence through its composition of uncertainty.

### 2.4.3. Action Policies: Directing Computational Resources

The power of the IterGP framework extends beyond its capacity for uncertainty quantification to encompass the strategic deployment of computational effort through action sequences $\{\boldsymbol{s}_i\}$. These policies fundamentally determine which components of the solution space receive computational attention at each iteration, thereby governing both convergence characteristics and uncertainty reduction patterns.

#### Policy Definition and Role

An action policy defines the sequence of action vectors $\{\boldsymbol{s}_i\}_{i=1}^{N}$, which direct computational resources to specific components of the solution space. Each action $\boldsymbol{s}_i \in \mathbb{R}^N$ determines which linear combination of residual components we observe. This is done through the scalar projection $\alpha_i = \boldsymbol{s}_i^T \boldsymbol{r}_{i-1}$. The choice of policy directly influences both the convergence rate of the representer weights and the spatial distribution of computational uncertainty.

#### Conjugate Gradient Policy

The conjugate gradients (CG) policy was one of the most prominent approaches explored within the IterGP framework. It sets $\boldsymbol{s}_i = \boldsymbol{r}_{i-1}$, where $\boldsymbol{r}_{i-1}$ is the current residual. This choice ensures that IterGP with CG actions recovers the classical conjugate gradient algorithm exactly in its posterior mean [4, Theorem S5].

The CG policy has strong theoretical foundations rooted in Krylov subspace methods. It minimizes the $\boldsymbol{G}$-norm of the error $\|\boldsymbol{x}^* - \boldsymbol{x}_i\|_{\boldsymbol{G}}$ at each iteration, where $\boldsymbol{G}$ represents the system matrix. This makes it optimal among all Krylov subspace methods for symmetric positive definite systems. The stability and convergence of CG depend on the condition number of $\boldsymbol{G}$ [8, sec. 7.3, 9, 9.2].

**Cholesky Policy**

The Cholesky policy systematically selects standard unit vectors $\boldsymbol{s}_i = e_{\pi(i)}$ where $\pi$ represents the predetermined pivoting strategy ($\boldsymbol{s}_i = e_i$ without pivoting). This approach corresponds to computing a partial Cholesky decomposition of the Gram matrix $\boldsymbol{G}$, with IterGP-Chol recovering the classical Cholesky factorisation exactly [4, Theorem S3].

Unlike CG's global approach, the Cholesky policy provides local uncertainty reduction. It targets specific datapoints sequentially, reducing computational uncertainty near selected points whilst leaving uncertainty large in unvisited regions. This localised behaviour proves advantageous when prior knowledge suggests certain regions of the domain require higher precision than others.

**Random Policy**

Random action policies select $\boldsymbol{s}_i$ by sampling from a specified distribution. This is typically uniform over the unit sphere or discrete uniform over standard basis vectors. While random policies lack the deterministic convergence guarantees of CG or Cholesky, they offer unique advantages for large-scale problems.

Random methods allow parallel implementations, as different processors can independently sample actions and contribute to the solution. Furthermore, randomized approaches often show robust performance across diverse problem classes. This helps avoid the worst-case scenarios that can trouble deterministic methods on adversarially structured problems [9].

### 2.4.4. The PDE Structure Gap

Having established the landscape of existing action policies, we now arrive at the central limitation that motivates our research contribution. Whilst CG, Cholesky and random policies each possess distinct advantages for their respective use cases, they all share a fundamental shortcoming: they operate generically without exploiting the structure inherent in PDE problems.

**Generic Limitations of Standard Policies**

Standard policies treat the Gram matrix $\boldsymbol{G} = \mathcal{L}k\mathcal{L}^T$ as an arbitrary symmetric positive definite matrix, applying action selection strategies that depend only on algebraic properties such as residual magnitudes, predetermined orderings, or stochastic sampling. This generic approach entirely ignores the fact that $\boldsymbol{G}$ arises from a specific mathematical context: the discretisation of partial differential equations with physical and spectral properties.

The CG policy, despite its optimality within Krylov subspaces, selects actions based solely on the current residual $\boldsymbol{r}_{i-1}$, without considering whether this direction aligns with the natural modes of the underlying PDE. Similarly, Cholesky policies follow predetermined pivot orderings that may bear no relationship to the problem's eigenmode structure. Random policies, by construction, cannot systematically exploit any underlying mathematical patterns.

**The Structure of PDEs**

Those patterns manifest in several fundamental ways that directly relate to the action selection problem:

- **Frequency-Domain Decomposition**: Linear PDEs naturally admit frequency-domain analysis through their eigenfunction expansions $u(x, t) = \sum_k a_{k(t)} \phi_k(x)$. The spatial eigenmodes $\phi_k$ corresponds to different frequency components, which can possibly be targeted using FFT-based techniques. [10, ch. 1]
- **Causal Strucuture**: Time-dependent PDEs exhibit inherent directionality where information propagates from earlier to later times, but not vice versa. For parabolic equations like the heat equation, this creates a natural temporal ordering where future states depend on past states but not the reverse. This causal structure suggests that action policies should respect temporal dependencies rather than treating all time points equivalently. [6, sec. 7.1]

The structure gap of generic policies motivates our main contribution: action policies that try to explicitly exploit the structure of PDEs in order to reduce uncertainty more efficiently. The following chapter develops this approach, demonstrating how directional and FTT-based action selection can lead to faster uncertainty convergence.

# 3. Structure-Informed Action Policies

## 3.1. Design Principles

The central challenge motivating our research lies in developing action policies that systematically reduce uncertainty more efficiently than existing generic approaches whilst maintaining computational tractability and numerical reliability. Our design philosophy centres on four fundamental principles that guide the construction of structure-informed action policies.

### 3.1.1. Primary Goal: Accelerated Uncertainty Reduction

Our overarching objective is to reduce posterior uncertainty in IterGP more rapidly than standard policies such as conjugate gradients. The computational uncertainty component of IterGP's posterior variance directly reflects how much our limited iterations affect the quality of our solution estimates. Faster uncertainty reduction translates to more reliable confidence bounds with fewer computational resources. In order to achieve this goal, we are willing to sacrifice overall best mean convergence as long as the mean is reasonable close to the optimal one.

Our approach explores whether exploiting PDE-specific structure - such as frequency-domain characteristics or temporal causality - can lead to action sequences that reduce uncertainty more efficiently. The specific mechanisms matter less than the end result: achieving reliable posterior distributions with fewer matrix-vector multiplications

### 3.1.2. Computational Tractability

The second principle requires that any sophisticated action selection strategy must remain computationally reasonable. There is no benefit in developing theoretically superior policies if their computational overhead negates the advantages of faster uncertainty reduction.

#### Complexity Constraints

Action computation should preferably exhibit sublinear complexity per iteration, where $i$ represents the current iteration number. This ensures that action selection overhead remains negligible compared to the dominant $\mathcal{O}(N)$ cost of matrix-vector multiplication $\boldsymbol{G}\boldsymbol{s}_{\boldsymbol{i}}$.

#### Memory Efficiency

Action policies should avoid storing large auxiliary matrices that scale with system size $N$. For problems with thousands of degrees of freedom, maintaining additional $N \times N$ matrices would be prohibitive.

#### Practical Implementations

Methods that require more complex handling should leverage standard computational primitives that are already highly optimised.

The key insight is that even modest improvements in uncertainty reduction per iteration can provide substantial benefits when the per-iteration overhead remains small. A policy that reduces uncertainty 20% faster while adding only 10% computational overhead represents a practical advantage.

#### 3.1.3. Numerical Stability

The third principle ensures that our action policies maintain numerical reliability across diverse problem conditions. Sophisticated action selection strategies can sometimes lead to ill-conditioned subproblems or near-singular search directions that compromise the overall algorithm's stability.

When structure-based policies encounter difficulties - such as unclear spectral structure or near-singular matrices - they should revert to the established fallback methods like conjugate gradients for the affected iteration, rather than introducing numerical problems.

#### 3.1.4. Implementation Philosophy

These three principles collectively establish a pragmatic framework for developing improved action policies. Our approach prioritises demonstrable improvements in uncertainty reduction over theoretical elegance, recognising that practical advances in computational science require methods that work reliably across diverse problem conditions.

**Empirical validation**

Rather than relying solely on theoretical arguments, we evaluate our policies through systematic comparison with established baselines on representative PDE problems. The ultimate test is whether our methods reduce computational uncertainty more efficiently in practice.

**Clear fallback paths:**

Every structure-informed policy includes well-defined conditions under which it reverts to conjugate gradients, ensuring that our methods never perform worse than established approaches in terms of reliability.

The subsequent sections detail how these principles manifest in concrete algorithmic implementations for heat and wave equations, demonstrating that principled exploitation of PDE structure can indeed lead to more efficient uncertainty quantification.

## 3.2. Failed Approaches

Prior to presenting the successful action policies, this section examines a promising approach that, although ultimately unsuccessful, provided deeper insight into the mathematical foundations of IterGP.

### 3.2.1. Calibrated Conjugate Gradients

Our initial approach attempted to reduce IterGP's computational uncertainty conservatism by leveraging the well-established spectral convergence properties of conjugate gradients. The motivation stemmed from Axelsson & Kaporin's analysis of CG convergence, which demonstrates that CG achieves superlinear convergence "when a sufficient number of extreme eigenvalues components have been essentially damped out" [11]

**Theoretical Motivation**

The standard IterGP-CG approach treats all unresolved components of the precision matrix approximation $\boldsymbol{C}_i \approx \hat{\boldsymbol{G}}^{-1}$ equally when computing computational uncertainty. However, CG theory suggests that this uniform treatment may be overly conservative. Axelsson & Kaporin [11] establish that CG convergence typically proceeds through three phases:

1. **Sublinear phase**: Initial rapid convergence depends on the initial error vector
2. **Linear phase**: Convergece dominated by the spectral condition number
3. **Superlinear phase** Achieved through systematic damping of extreme eigenvalue components

Our hypothesis was that if we could track which eigenvalue components had been "essentially damped out" during the CG process, we would proportionally reduce the computational uncertainty without compromising IterGP's worst-case guarantees.

**Proposed Approach**

**Eigenvalue Resolution Trakcing**: We attempted to define a "resolved eigenvalue fraction":

$$\rho_i = \frac{\text{number of sufficiently damped eigencomponents}}{\text{total number of eigencomponents}}.$$

The idea was to calibrate computational uncertainty as:

$$k_{\text{comp},i}^{\text{calibrated}}(\cdot,\cdot) = (1-\rho_i)\cdot k_{\text{comp},i}(\cdot,\cdot).$$

**Fundamental Challenges Encountered**

Implementation efforts revealed several significant challenges:

**Lack of Rigorous Theory**: Despite extensive literature review, we could not establish rigorous bounds connecting eigenvalue component damping to total uncertainty reduction. We were struggling to find papers that describe:

- Quantitatively, thresholds for when components are "sufficiently damped"
- Rigorous relationships between damped eigencomponents and total error reduction.

Although we investigated multiple strategies for deriving conservative uncertainty bounds from eigenvalue convergence information, none proved practically feasible.

**Matrix-Free Constraint Violation:** Attempts to independently manipulate the computational uncertainty component conflict with IterGP's underlying algebraic framework. The core issue lies in IterGP's combined uncertainty function:

$$k_i(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_i(\boldsymbol{x}_1, \boldsymbol{x}_2) - k_i(\boldsymbol{x}_1, X) C_i k_i(X, \boldsymbol{x}_2)$$

It is an indivisible unit where matrix inversion cancels out algebraically.

IterGP's computational efficiency stems from working with the combined uncertainty directly, which avoids explicit computation of matrix inverses. The framework never separately computes $k_{\text{math}}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $k_{\text{comp},i}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ as independent quantities. Instead, these components exist only as a conceptual decomposition of the unified posterior covariance, where the mathematical structures ensure that expensive matrix inversions are eliminated through algebraic cancellation.

Our proposed calibration approach required decomposing the combined uncertainty into its mathematical and computational components, then applying a reduction factor $(1 - \rho_i)$ specifically to the computational term before recombining them. However, this decomposition necessitates explicit computation of terms that IterGP deliberately avoids through its algebraic structure. Any of our attempts to isolate and modify the computational uncertainty component reintroduced the very matrix inversions that IterGP's design eliminates.

## 3.3. Successful Action Policies

### 3.3.1. Continouous Adaptive Policy

Traditional action policies in IterGP, such as conjugate gradients, operate purely in the spatiotemporal domain without exploiting the frequency structure inherent in PDE residuals. The conjugate gradient policy selects actions based solely on the residual: $\boldsymbol{s}_k = \boldsymbol{r}_{k-1}$, treating all frequency components equally, regardless of their relative importance to the underlying PDE solution.

However, partial differential equations naturally decompose into frequency components through their eigenmode structure as described in Section 2.4.4. This spectral structure suggests that it might be beneficial to select actions that prioritize components where the residual exhibits the strongest characteristics, rather than applying uniform treatment across all frequencies.

The continuous adaptive policy addresses this gap by constructing actions through frequency-domain analysis of the residual. Rather than selecting actions based purely on spatial magnitude, this approach:

1. **Transforms the residual to the frequency domain** to reveal its spectral energy distribution
2. **Applies energy-dependent weighting** to emphasise frequencies where uncertainty is concentrated
3. **Constructs actions that naturally target dominant spectral components** without requiring prior knowledge of the PDE's eigenmode structure

We design this frequency-aware approach to achieve three goals:

- **Adaptive resolution**: Automatically adjusts frequency emphasis based on the residual's actual spectral content
- **Energy preservation**: Maintains spectral relationships whilst amplifying significant frequency components

**Mathematical Formulation**

The continuous adaptive policy constructs action vectors through energy-weighted frequency domain filtering:

$$\boldsymbol{s}_k = \mathcal{F}^{-1}[\boldsymbol{w}_k \odot \mathcal{F}[r_{k-1}]]$$

where $\boldsymbol{r}_{k-1}$ is the current residual, $\mathcal{F}$ denotes the discrete Fourier transformation and $\odot$ represents element-wise multiplication.

For a residual of length $N$, the discrete Fourier transform produces $N$ frequency components indexed by $j \in \{1, 2, ..., N\}$. The frequency weights $\boldsymbol{w}_{\boldsymbol{k}}$ are derived from the residual's spectral energy distribution:

$$E_j = \left|\mathcal{F}[r_{k-i}]_j\right|^2.$$

We normalise and apply quadratic energy amplification:

$$\tilde{E}_j = \frac{E_j}{\sum_{k=1}^{N} E_k}$$

$$w_j = \tilde{E} \cdot \alpha \tilde{E}^2$$

where $\alpha = 2$ is the emphasis parameter. This formulation ensures that:

- High-energy frequencies receive quadratic amplification
- Low-energy frequencies are preserved
- All spectral components still contribute to the final action vector

**2D Logarithmic Energy Amplification**

For spatiotemporal problems with grid dimensions $(n_t, n_x)$, the 2D implementation instead uses a more conservative logarithmic amplification strategy instead of the quadratic approach in 1D. This design choice addresses the increased complexity of 2D frequency interactions and prevents over-amplification that could destabilise the algorithm.

The 2D energy computation follows the same pattern:

$$E_{i,j} = \left|\mathcal{F}_{\text{2D}[r_{k-1}]_{i,j}}\right|^2, \text{total energy: } E_{\text{total}} = \sum_{m=1}^{n_t} \sum_{n_1}^{n_x} E_{m,n}.$$

However, the weighting function uses logarithmic scaling to prevent over-amplification:

$$w_{i,j} = \frac{E_{i,j}}{E_{\text{total}}} \cdot \left(1 + \ln\left(1 + \frac{E_{i,j}}{E_{\text{total}}}\right)\right).$$

The logarithmic approach in 2D serves two important purposes:

1. **Stability in Higher Dimensions**: 2D problems have $n_t \times n_x$ frequency components compared to $n$ in 1D. We observed numerical instabilities when using quadratic amplification across the larger 2D frequency space.
2. **Smooth Energy Weighting**: The logarithmic function provides smooth, continuous weighting that naturally compresses the dynamic range of energy ranges whilst still emphasising high-energy frequencies

**Computational Complexity**

The continuous adaptive policy incurs significantly higher computational costs per iteration compared to standard action policies, due to its frequency-domain operations. The spectral approach incurs substantial computational cost through:

1. **Forward FFT**: $\mathcal{O}(n \log n)$ to transfrom residual to frequency domain
2. **Frequency domain processing**: $\mathcal{O}(n)$ for energy computation and weight application
3. **Inverse FFT**: $\mathcal{O}(n \log n)$ to return to spatial domain

**Total per-iteration complexity**: $\mathcal{O}(n \log n)$ for 1D problems, $\mathcal{O}(n_t n_x \log(n_x n_t))$ for 2D spatiotemporal problem.

This represents an overhead factor of $n \log n$ compared to CG for 1D problems, so this policy can only be justified when the number of iterations required for acceptance is significantly reduced.

### 3.3.2. Time Slice Policy

Whilst the continuous adaptive policy exploits frequency-domain structure, it operates without consideration of the fundamental causal structure inherent in time-dependent PDEs. For parabolic equations such as the heat equation, information propagates naturally from earlier to later times, creating an inherent temporal directionality that suggests computational resources should be allocated accordingly.

The time slice policy addresses this gap by directly targeting the temporal structure of spatiotemporal problems. Rather than treating all spatial locations equally, this approach identifies and prioritises spatial regions where temporal evolution exhibits the highest residual magnitude, thereby concentrating computational effort where uncertainty reduction yields the most benefit.

**Mathematical Formulation**

For spatiotemporal problems discretised on an $n_x \times n_x$ grid, the time slice policy operatos by reshaping the residual vector $\boldsymbol{r}_{k-1} \in \mathbb{R}^{n_t \cdot n_x}$ into its natural matrix form $\boldsymbol{R}_{k-1} \in \mathbb{R}^{n_t \times n_x}$, where rows correspond to temporal indices, and columns represent spatial locations.

The policy identifies the spatial location $j^*$ with the maximum residual:

$$j^* = \arg \max_{j \in \{1,\ldots,n_x)} \|\boldsymbol{R}_{k-1}[:, j]\|_2.$$

The action vector then targets the complete temporal evolution at this spatial location:

$$\boldsymbol{s}_k = \text{vec}(\boldsymbol{S}_k)$$

where the action matrix $\boldsymbol{S}_k \in \mathbb{R}^{n_t \times n_x}$ is defined as:

$$\boldsymbol{S}_k[:, i] = \begin{cases} \boldsymbol{R}_{k-1}[:, i] \text{ if } i = j^* \\ \mathbf{0} \text{ else} \end{cases}.$$

This formulation ensures that computational effort concentrates on the spatial location experiencing the most significant temporal dynamics, whilst still using inexpensive residual operations.

**Spatial Mixing Enhancements**

The basic time slice policy exhibits excellent temporal targeting but may suffer from spatial locality limitations. To address this, we developed an enhanced variant that incorporates spatial neighbourhood information through weighted contributions from adjacent spatial locations.

The spatial mixing variant modifies the action construction to include weighted contributions from spatial neighbours:

$$\boldsymbol{S}_{k[:,i]} = \begin{cases} \boldsymbol{R}_{k-1}[:, i] \text{ if } i = j^* \\ w \cdot \boldsymbol{R}_{k-1}[:, i] \text{ if } i \in \{j^* - 1, j^* + 1\} \subset \{1, ..., n_x\} \\ \mathbf{0} \text{ else} \end{cases}$$

where $w \in [0.1, 0.5]$ represents the spatial mixing weight. These additions help prevent pathological clustering of actions around a single spatial location and exploits spatial correlations more efficently.

**Computational Complexity**

The time slice policy exhibits significant computational advantages over the previously seen frequency-domain approaches. The core operations consist of:

1. **Residual Reshaping**: $\mathcal{O}(1)$ memory indexing operation
2. **Norm Computation**: $\mathcal{O}(n_t \cdot n_x)$ for computing spatial norms
3. **Action Construction** $\mathcal{O}(n_t)$ for copying temporal slice data

**Total per-iteration complexity**: $\mathcal{O}(n_t \cdot n_x)$

This represents a substantial improvement over the continuous adaptive policy's $\mathcal{O}(n_t n_x \log(n_x n_t))$ complexity. Moreover, the time slice policy's operations - particularly the spatial norm computations - admit efficient vectorised implementations that further reduce computational overhead in practice.

### 3.3.3. Temporal First Policy

The temporal first policy represents a fundamental departure from generic action selection strategies by explicitly recognising and exploiting the **causal directionality** inherent in time-dependent PDEs. Unlike heat conduction in space, which can propagate in all directions, temporal evolution in parabolic equations flows unidirectionally from past to future. This asymmetry suggests that computational resources should be allocated with temporal precedence, leading to the policy's distinctive two-phase approach. However, instead of choosing time slices as seen in the Time Slice Policy, we choose the actions based on the frequency domain.

#### Core Philosophy and Design Rationale

The central insight underlying the temporal first policy is that spatiotemporal PDE residuals possess a hierarchical structure where temporal frequencies often exhibit clearer, more systematic patterns than their spatial counterparts. In heat diffusion problems, for instance, the temporal evolution follows a well-understood exponential decay modes, whilst spatial patterns may be complicated by irregular boundary conditions, heterogeneous material properties, or complex geometries.

Rather than treating temporal and spatial components with equal priority, the temporal first policy implements a **structured attention mechanism** that systematically targets temporal frequency bands whilst maintaining controlled spatial awareness.

#### Algorithmic Framework

The temporal first policy operates through a twofold computational strategy with the following key parameters:

- temporal_phase_fraction $\in (0, 1)$: fraction of total iterations dedicated to temporal targeting (default: 0.4)
- num_temporal_bands: number of discrete temporal frequency bands for systematic progression (default: 6)
- spatial_coverage $\in (0, 1]$: fraction of spatial frequeny spectrum preseved during temporal phase (default: 0.5)
- spatial_refinement_policy: strategy for Phase 2 (e.g., CG, Continuous Adaptive,...)

**Phase 1: Systematic Temporal Frequency Targeting**

During the first $\lfloor \text{temporal\_phase\_fraction} \times \text{max\_iterations} \rfloor$ iterations, the policy implments **progressive temporal band targetting** using 2D FFt decomposition. For spatiotemporal problems discretised on an $n_t \times n_x$ grid, the residual vector $\boldsymbol{r}_{k-1} \in \mathbb{R}^{n_t \cdot n_x}$ is reshaped into matrix form $\boldsymbol{R}_{k-1} \in \mathbb{R}^{n_t \times n_x}$ as seen before.

These temporal frequency bands are created using logarithmic spacing to provide enhanced resolution at low frequencies.

$$\{\ell_1, ..., \ell_{\text{num\_temporal\_bands}}\} = \text{range}(0, \ln(\omega_{\max} + 1), \text{length} = \text{num\_temporal\_bands})$$

$$\text{band\_edges} = \{0, \lfloor e^{\ell_1} \rfloor, ..., \lfloor e^{\ell_{\text{num\_temporal\_bands}}} \rfloor = \omega_{\max}\}$$

where $\omega_{\max}$ is the highest frequency of the FFT.

**2D Frequency Domain Filtering with Spatial Awareness**

The core computational procedure employs rectangular filtering in the 2D frequency domain, enabling temporal focus whilst preserving essential spatial coupling. The process proceeds as follows:

1. **Forward 2D FFT Transform**:

$$\hat{\boldsymbol{R}}_{k-1}[\omega_t, \omega_x] = \mathcal{F}_{2D}[\boldsymbol{R}_{k-1}][i, j], \quad i = 1, ..., n_t, \quad j = 1, ..., n_x$$

2. **Frequency Index Mapping**: For FFT output indices $(i, j)$, the corresponding temporal and spatial frequencies are:

$$\omega_t = \begin{cases} i - 1 \text{ if } i \leq \frac{n_t}{2} \\ i - n_t - 1 \text{ if } i > \frac{n_t}{2} \end{cases}, \quad \omega_x = \begin{cases} i - 1 \text{ if } i \leq \frac{n_x}{2} \\ i - n_t - 1 \text{ if } i > \frac{n_x}{2} \end{cases}$$

This mapping accounts for the standard FFT convention, where negative frequencies appear in the second half of the frequency array.

3. **Temporal Band selection**: For the current temporal band $b$ the frequency range is:

$$\Omega_{t,b} = [\text{band\_edges } [b], \text{band\_edges } [b + 1])$$

4. **Spatial Coverage Cutoff**: The spatial frequency cutoff implements controlled spatial awareness

$$\omega_{x,\max} = \left\lfloor \text{spatial\_coverage} \times \frac{n_x}{2} \right\rfloor$$

This parameter controls the trade-off between temporal focus and spatial detail preservation:

- spatial_coverage = 0.3: Preserve only broad spatial patterns
- spatial_coverage = 0.5: Balances spatial awareness
- spatial_coverage = 1.0: Include all spatial details

5. **Rectangular Frequency Domain Filter**: The filtered FFT coefficients are constructed through rectangular masking:

$$\hat{\boldsymbol{R}}_{k,\text{filtered}}[\omega_t, \omega_x] = \begin{cases} \hat{\boldsymbol{R}}_{k-1}[\omega_t, \omega_x] \text{ if } |\omega_t| \in \Omega_{t,b} \wedge |\omega_x| \leq \omega_{x,\max} \\ 0 \text{ otherwise} \end{cases}$$

5. **Inverse 2D FFT Transform**:

$$R_k[i, j] = \mathcal{F}^{-1}\left[\hat{R}_{k,\text{ filtered}}[\omega_x, \omega_t]\right]$$

$$r_k = \text{vec}(R_k)$$

**Phase 2: Spatial Refinement**
After completing the temporal targeting phase, the policy transitions to spatial refinement using the specified spatial_refinement_policy. This modular design enables combination with any existing action strategy.

**Computational Complexity**
The total complexity of this policy depends on the phase and the chosen spatial refinement policy. The temporal phase exhibits the same complexity of $\mathcal{O}(n_t n_x \log(n_x n_t))$ as seen in Section 3.3.1 due to the transformation in the frequency domain.
The amortised total complexity depends on the temporal phase fraction. With the default 40% temporal phase, approximately 60% of iterations operate at the spatial policy's complexity, providing computational efficiency when paired with lower-cost spatial methods like conjugate gradients.

# 4. Experimental Design

This section details the experimental framework designed to evaluate the performance of structure-informed action policies against established baselines. Our experimental design addresses our central research question: can our action policies that try to exploit PDE-specific structure achieve faster uncertainty reduction compared to generic approaches in the IterGP framework?

## 4.1. Experimental Framework and Implementation

### 4.1.1. Software Implementation

Our experiments utilise the *GaussPDE.jl* package developed by Tim Weiland, which provides a comprehensive framework for probabilistic PDE solving using physics-informed Gaussian processes. We extended the framework by implementing the IterGP methodology, which enables iterative uncertainty-aware PDE solving with customisable action policies.

The implementation maintains full compatibility with the existing *GaussPDE.jl* architecture whilst introducing the capability for systematic comparison of action policies for iterative solving. All experiments are conducted using Julia 1.11 on a MacBook Pro (M1) and the ML-Cloud (CPU-Galvani).

### 4.1.2. Action Policy Portfolio

Our experimental evaluation encompasses six distinct action policies, to represent different approaches to exploiting PDE structure:

**Baseline Policy**

- **Conjugate Gradients (CG)**: The established baseline policy, providing optimal convergence within Krylov subspaces and serving as our primary comparison policy.

**Structure-Informed Policies**

- **Continuous Adaptive**: Frequency-domain policy using energy-weighted spectral filtering with 2D FFT decomposition and logarithmic energy amplification.
- **Time Slice**: Including both variants (with and without neighbours) that exploit temporal causality by targeting spatial locations with maximum temporal residual evolution.
- **Temporal-First + Continous Adaptive**: A compositional two-phase approach combining systematic temporal frequency targeting (40% of iterations) followed by continuous adaptive spatial refinement.
- **Temporal-First + CG**: Similar two-phase structure but employing only 4 bands and frequency targeting of only 20% of iterations, followed by spatial refinement through CG.

## 4.2. Test Problem Configuration

### 4.2.1. Heat Equation Test Suite

The parabolic heat equation serves as our first test bed, governed by:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u$$

$$\alpha = \frac{\kappa}{c\rho}$$

where $\kappa$ is the thermal conductivity, $c$ is the specific heat capacity and $\rho$ the density. We evaluate six parameter configurations designed to probe different aspects of diffusion dynamics:

| **Configuration** | $\kappa$ | $\rho$ | **c** | **IC Coefficients** | **Physical Interpretation** |
|---|---|---|---|---|---|
| Slow Diffusion | 0.05 | 1.0 | 1.0 | [1.0, 0.5, 0.25, 0.125] | High-contrast, localised evolution |
| Medium Diffusion | 0.1 | 1.0 | 1.0 | [1.0, 0.5, 0.25, 0.125] | Reference diffusion rate |
| Fast Diffusion | 0.2 | 1.0 | 1.0 | [1.0, 0.5, 0.25, 0.125] | Rapid equilibration |
| High Amplitude | 0.1 | 1.0 | 1.0 | [2.0, 1.0, 0.5, 0.25] | Enhanced signal strength |
| Low Amplitude | 0.1 | 1.0 | 1.0 | [0.5, 0.2, 0.1, 0.05] | Weak signal detection |
| High Density | 0.1 | 2.0 | 1.0 | [1.0, 0.5, 0.25, 0.125] | Altered material properties |

Table 1: Heat Equation Configurations

Each configuration employs truncated sine series initial conditions $u(x,0) = \sum_{i=1}^{4} a_i \sin(i\varphi x)$ with Dirichlet boundary conditions u(0,t) = u(1,t) = 0 on the spatiotemporal domain $[0,2] \times [0,1]$.

### 4.2.2. Wave Equation Test Suite

The hyperbolic wave equation provides a complementary test environment:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u$$

Five parameter variants explore different wave propagation characteristics:

| **Configuration** | **c** | **IC Displacement** | **IC Velocity** | **Wave Behaviour** |
|---|---|---|---|---|
| Reference | 2.0 | [1.0, 0.5, 0.25, 0.125] | [0.0, 0.0, 0.0, 0.0] | Standard propagation |
| Enhanced Modes | 2.0 | [1.0, 0.6, 0.35, 0.2] | [0.0, 0.0, 0.0, 0.0] | Enriched frequency content |
| Faster Wave | 2.4 | [1.0, 0.5, 0.25, 0.125] | [0.0, 0.0, 0.0, 0.0] | Increased propagation speed |
| With Velocity | 2.0 | [1.0, 0.5, 0.25, 0.125] | [0.2, 0.1, 0.05, 0.025] | Non-zero initial velocity |
| Redistributed | 2.0 | [0.9, 0.55, 0.3, 0.15] | [0.0, 0.0, 0.0, 0.0] | Alternative mode distribution |

Table 2: Wave Equation Configurations

Each configuration employs truncated sine series for both displacement and velocity initial conditions on the spatiotemporal domain $[0,2] \times [0,1]$. Displacement and velocity are both provided as initial conditions using a truncated sine series. The boundaries are set up using homogeneous Dirichlet boundary conditions imposed on both spatial boundaries: $u(0,t) = u(1,t) = 0$.

## 4.3. Observation Strategy and Discretisation

Following the original purpose of the GaussPDE.jl framework, we employ finite volume method (FVM) discretisation for PDE constraint generation as described by Tim Weiland et al. [12] This approach integrates the differential operator over control volumes:

$$\int_{V_i} D[u]\,\mathrm{d}x = \int_{V_i} f\,\mathrm{d}x$$

**Spatial Resolution**: $\Delta x = 0.05$ yielding approximately 20 spatial control volumes. **Temporal Resolution**: $\Delta t = 0.05$ producing approximately 40 temporal intervals. **Total PDE Constraints**: Approximately 800 volumetric observations per test case.

### 4.3.1. Multi-Type Observation Framework

Each test problem incorporates heterogeneous observation types to better reflect realistic PDE solving scenarios:

- **Initial Conditions**: 20 point observations of $u(\boldsymbol{x},0)$ (heat) or $u(\boldsymbol{x},0)$ and $\frac{\partial u}{\partial t}(\boldsymbol{x},0)$ (wave)
- **Boundary Conditions**: 40 Dirichlet boundary observations enforcing $u(\partial\Omega,t) = 0$
- **PDE Constraints**: 800 FVM observations ensuring differential equation satisfaction

### 4.3.2. Iterative Solver Configuration

Our implementation of the IterGP framework processes observations in sequential layers even though they are stored in the same matrix:

1. **Layer 1**: Intitial conditions using Cholesky policy
2. **Layer 2**: Boundary conditions using Cholesky policy
3. **Layer 3**: PDE constraints using the test policy (CG, Continuous Adaptive, etc.)

Cholesky was chosen for the boundary and initial conditions to provide a numerically stable base to work for the action policies in the third layer.

## 4.4. Evaluation Methodology

### 4.4.1. Baseline Computation

For each parameter configuration, we establish a baseline using direct cholesky conditioning:

$$p(u|\mathcal{L}[u] = y) = \mathrm{GP}(\mu_{\mathrm{post}}, k_{\mathrm{post}})$$

This baseline represents the mathematically exact posterior (up to numerical precision) and serves as the reference for all comparative metrics.

### 4.4.2. Convergence Metrics

Our evaluation framework focuses on three core quantitative metrics that capture the essential aspects of policy performance:

1. **Mean Squared Error**: $\|\mu_k - \mu_{\mathrm{baseline}}\|^2$ - posterior mean convergece to exact solution
2. **Variance Squared Error**: $\|\mathrm{diag}(\Sigma_k) - \mathrm{diag}(\sigma_{\mathrm{baseline}})\|^2$ - marginal uncertainty convergence
3. **KL Divergence**:

$$D_{\mathrm{KL}}(\mathcal{N}_b \parallel \mathcal{N}_i) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_i^{-i}\Sigma_b) - k + (\mu_i - \mu_b)^T \Sigma_i^{-1}(\mu_i - m_b) + \mathrm{logdet}\left(\frac{\Sigma_i}{\Sigma_b}\right)\right]$$

   where $b$ stands for baseline, $i$ represents the current iteration and $k$ is the dimension of the GP.

To enable meaningful comparison across different parameter configurations and policies, we employ a systematic normalisation approach:

1. **First-Iteration Scaling**: Mean and variance squared error are normalised by their respective values at the first iteration of each policy:

$$\mathrm{Metric}_{\mathrm{norm},k} = \frac{\mathrm{Metric}_k}{\mathrm{Metric}_1}$$

   This removes parameter-dependent scaling effects whilst preserving relative convergence behaviour, as the first iteration is the same regardless of the chosen policy.
2. **Cross-Parameter Averaging**: For each policy, normalised metrics are averaged across all parameter configurations:

$$\text{Policy Performance}_k = \frac{1}{N_{\mathrm{params}}} \sum_{p=1}^{N_{\mathrm{params}}} \mathrm{Metric}^p_{\mathrm{norm},k}$$

This aggregation strategy ensures robust policy comparison whilst testing across diverse PDE parameter regimes.

#### 4.4.3. Evaluation Grid

All metrics are computed on a standardised evaluation grid:

- **Temporal Points**: 40 uniformly spaced points in $[0, 2]$
- **Spatial Points**: 50 uniformly spaced points in $[0, 1]$
- **Total Evaluation Points**: 2000 spatiotemporal locations

## 5. Results

This section is a evaluation of our structure-informed action policies against the conjugate gradients baseline across both heat and wave equation test cases. Our experimental results demonstrate that exploiting PDE-specific structure, particularly temporal causality, yields substantial improvements in uncertainty reduction efficiency whilst maintaining acceptable mean convergence.

### 5.1. Heat Equation Performance

The heat equation test suite provides our first evaluation framework, with results averaged across six parameter configurations to ensure robust policy comparison. Figure 1 presents the convergence behaviour across our three key metrics: variance convergence (a), mean convergence (b), and covariance structure similarity via KL-Divergence (c).

#### 5.1.1. Variance Convergence: The Primary Success

The most relevant result emerges in variance convergence Figure 1(a), where temporal first policies show substantial advantages over the conjugate gradients baseline. Specifically, the "Temporal First CG Fast" policy achieves variance reduction after approximately 120 iterations, whereas the standard CG requires nearly 200 iterations to match this performance.

Notably, other structure-informed policies (Continuous Adaptive, Time Slice, and Time Slice + Neighbours) did not outperform CG in either variance reduction or covariance recovery, as they remained within the same performance range as the baseline. Furthermore, KL-Divergence (Figure 1(c)) of all action policies closely follow the CG convergence trajectory, thus demonstrating comparable uncertainty quantification.

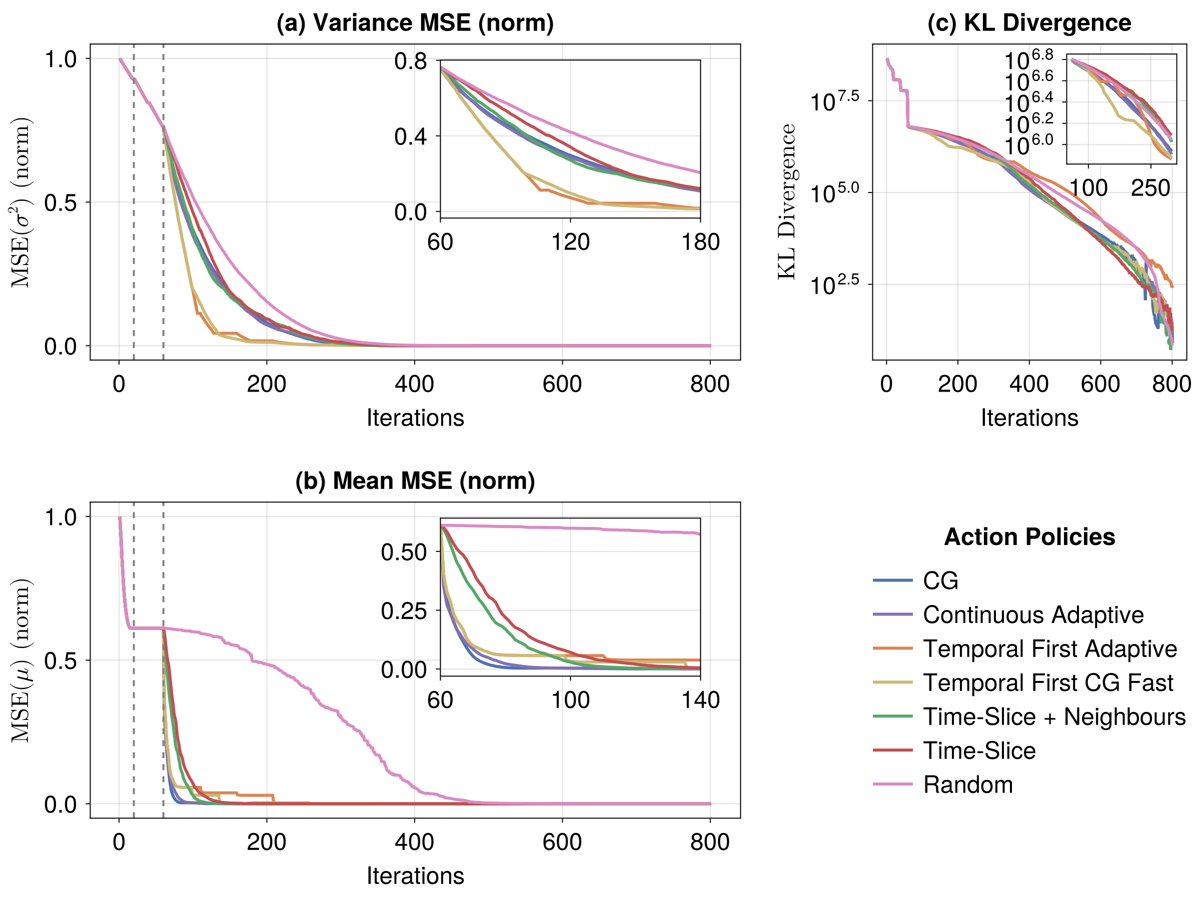

Figure 1: **Heat Equation Policy Comparison**. Convergence comparison of structure-informed action policies for the heat equation test suite. All metrics are normalised by their first iteration values and averaged across six parameter configurations (Table 1) (a) MSE Variance (norm): Temporal first policies show a substantial reduction in the first iterations, even outperforming CG, while other policies approximate the convergence rate of CG. (b) MSE Mean (norm): All policies exhibit a rapid mean convergence within 100 iterations, with temporal first policies maintaining competitive mean accuracy whilst prioritising uncertainty reduction. (c) KL-Divergence: No clear improvements over the baseline can be observed.

This highlights that not all structure-exploiting approaches fulfil their design goals or yield practical improvements. In contrast, only systematic targeting of temporal frequency structure achieves meaningful gains over the baseline, applying across both marginal uncertainty reduction and global covariance structure recovery.

### 5.1.2. Mean Convergence: Acceptable Trade-offs

Figure 1(b) reveals that our policies achieve competitive mean convergence while also providing decent uncertainty reduction. All policies converge rapidly in the first 100 iterations. The mean squared error drops from about 0.8 to below 0.05.

The temporal first policies have slightly higher residual mean error compared to pure CG during iterations 80-200. The slower “Temporal First Adaptive” variant maintains a heightened MSE error up to iteration 200. This limitation can be addressed through parameter tuning, as shown by the improved performance of the “Temporal First CG Fast” variant.

The remaining structure-informed policies (Continuous Adaptive and Time Slice variants) perform marginally worse than CG but remain within acceptable tolerance bounds. These modest mean convergence compromises are justifiable given the substantial variance reduction advantages of the temporal first approaches.

This trade-off aligns with our design philosophy. We explicitly prioritise uncertainty quantification over marginal improvements in point estimates. Reliable confidence bounds are often more valuable than perfect mean convergence in practical PDE applications, where decision-making under uncertainty is paramount.

## 5.2. Wave Equation Performance

The wave equation test suite provides validation of our structure-informed policies across hyperbolic PDEs, with results averaged across five parameter configurations to assess robustness across different wave propagation characteristics. The wave equation results reveal both the strengths and limitations of temporal structure exploitation when applied to oscillatory phenomena.

### 5.2.1. Variance Convergence: Sustained Advantages with Reduced Margins

The temporal first policies maintain their advantages in variance convergence for the wave equation, though with reduced margins compared to the heat equation case. The “Temporal First CG Fast” policy achieves variance reduction after approximately 185 iterations that the standard CG policy requires 220+ iterations to match.

The covariance structure recovery, as measured by KL-Divergence Figure 2(c), reveals no improvement, mirroring the results observed for the heat equation. The temporal first adaptive action policy appears to even worsen convergence between iterations 220-400. Note that the plot excludes the final iterations of some policies due to high numerical instabilities encountered during KL-Divergence computations at later stages.

This more modest improvement reflects the fundamental differences between parabolic and hyperbolic PDEs. Wave equations exhibit oscillatory behaviour that creates a more

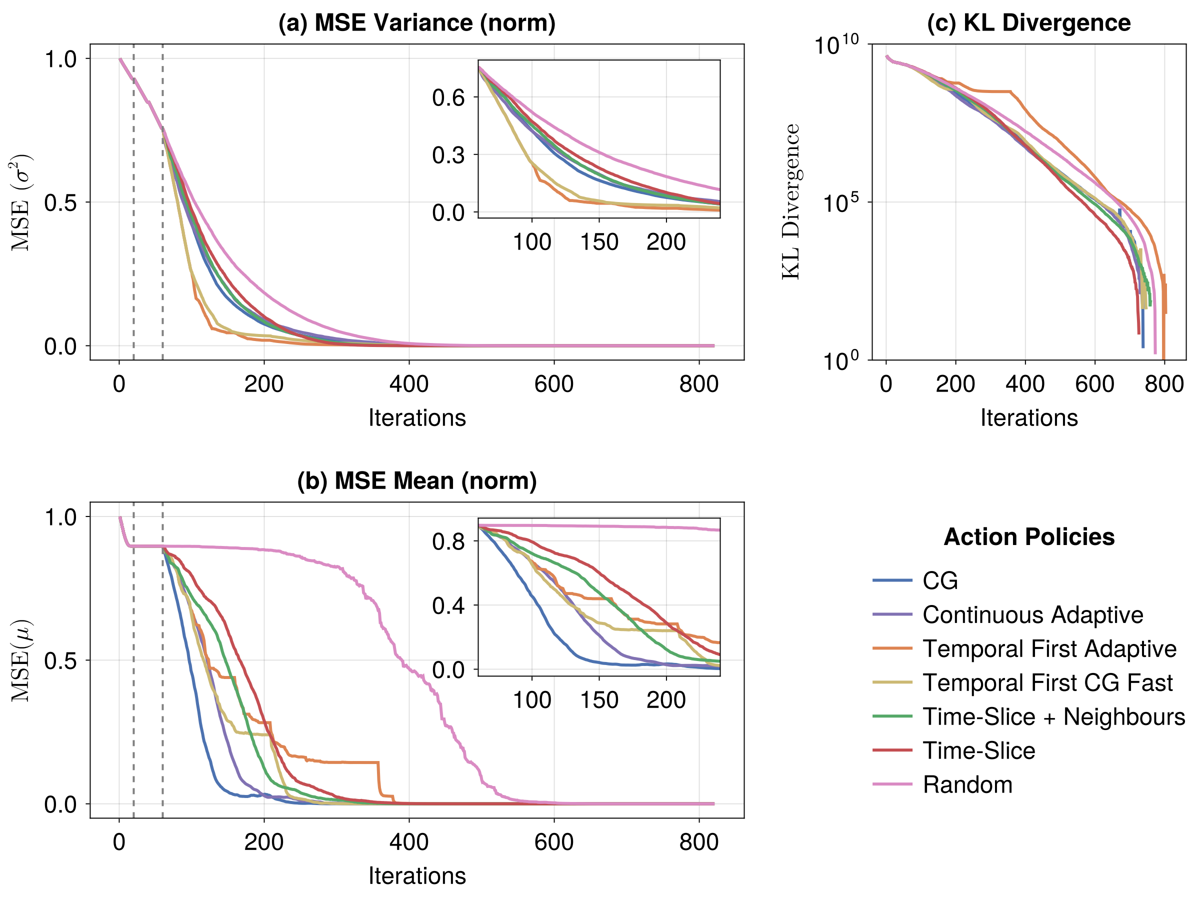

Figure 2: **Wave Equation Policy Comparison**. Convergence comparison of structure-informed action policies for the wave equation test suite. All metrics are normalised by their first iteration values and averaged across six parameter configurations (Table 2) (a) MSE Variance (norm): Temporal first policies show a substantial reduction in the first iterations, even outperforming CG, while other policies approximate the convergence rate of CG. (b) MSE Mean (norm): Our Policies converge more slowly in relation to CG. The Temporal First Adaptive policy only reaches an MSE of under 0.1 after 300+ iterations. (c) KL-Divergence: No clear improvement over the baseline can be observed.

complex spectral structure compared to the exponential decay modes characteristic of heat diffusion, reducing the effectiveness of systematic temporal frequency targeting. The other structure-informed policies (Continuous Adaptive, Time Slice variants) again failed to outperform CG, remaining within comparable performance ranges. This consistent pattern across both PDE types reinforces that only temporal first approaches provide reliable uncertainty reduction benefits.

#### 5.2.2. Mean Convergence: Performance Degradation: Performance Degradation and Trade-offs

Wave equation results reveal more pronounced trade-offs in mean convergence compared to heat equation performance. Standard CG achieves mean convergence around iteration 170, whilst our temporal first policies require 220+ iterations.

The “Temporal First CG Fast” policy demonstrates the value of early switching to spatial refinement, as its earlier transition to CG (at 20% of iterations rather than 40%) enables faster mean convergence recovery. This suggests that the optimal temporal phase fraction, unfortunately, depends on the underlying PDE characteristics.

The slower converging “Temporal First Adaptive” policy exhibits more pronounced mean convergence degradation, maintaining elevated MSE error well beyond iteration 200. This performance difference highlights the importance of spatial refinement strategy selection for hyperbolic PDEs.

# 6. Conclusion

## 6.1. Key Insights

### 6.1.1. Temporal Structure Exploitation: Success and Limitations

The temporal first policy demonstrates the most significant success among our structure-informed approaches, achieving substantial variance reduction improvements over conjugate gradients baselines across both heat and wave equation test cases. This policy’s two-phase architecture—systematic temporal frequency targeting followed by spatial refinement—consistently outperforms generic methods by exploiting the inherent causal directionality of time-dependent PDEs.

However, the results reveal parameter sensitivity that affects both effectiveness and computational efficiency. While the standard 40% temporal phase performs well for heat equations, wave equation results expose mean convergence stagnation when temporal exploration extends too long. The “fast” variant with a reduced 20% temporal phase demonstrates superior mean recovery while maintaining uncertainty benefits, indicating that optimal temporal phase fractions may depend on underlying PDE characteristics. This parameter sensitivity represents both the policy’s strength—its adaptability to different mathematical structures—and its primary limitation requiring problem-specific tuning.

In stark contrast, alternative structure-informed approaches consistently fail to deliver practical advantages. The continuous adaptive, time slice, and spatial mixing policies remain within comparable performance ranges to conjugate gradients without achieving meaningful improvements in uncertainty quantification. This selective success demonstrates that not all structure-exploiting approaches bring computational advantages. Frequency-domain filtering without temporal awareness (continuous adaptive) and spatial targeting by residual magnitude (time slice variants) do not match the strong convergence properties of established methods.

### 6.1.2. Computational Cost Considerations

The temporal first policy is one of our most computationally expensive approaches because of its 2D FFT operations in the temporal targeting phase. However, the results suggest this higher cost is justifiable for certain use cases. The large variance convergence improvements—achieving in 120-180 iterations what CG requires in 200-220 iterations—show possible computational savings even with higher per-iteration costs.

There exists a computational sweet spot where running the temporal first policy for limited iterations, followed by a transition to lower-cost methods, may provide optimal efficiency. The plots demonstrate that even brief temporal structure exploitation yields disproportionate uncertainty reduction benefits during early iterations.

## 6.2. Limitations and Future Directions

### 6.2.1. Scope Limitation

Our analysis focuses on linear PDEs with structured finite volume discretisations. This restricts generalisability to three primary cases: nonlinear PDE systems, where spectral structure depends on the problem; irregular geometries, where FFT-based methods become inefficient; and adaptive mesh refinement, where temporal structure may be inconsistent.

### 6.2.2. Future Directions: Parameter Tuning

The core challenge addressed by our research is determining optimal temporal phase parameters as well as choosing a fitting spatial refinement policy without advance knowledge of convergence. The effectiveness of the temporal-first policy hinges on accurately weighing temporal exploration against spatial refinement while keeping the computational costs in mind.

**Adaptive Phase Duration Control**

Future research should focus on developing online algorithms that dynamically adjust temporal phase duration using real-time convergence indicators. Potential avenues include spectral energy analysis, which tracks the stabilization of residual temporal patterns, signaling diminished returns from continued temporal targeting. Another approach is marginal variance tracking, which switches to CG once convergence stagnates or crosses a predefined threshold. Additionally, residual structure evaluation could potentialy measure stabilization in residuals, indicating reduced benefits from continued temporal targeting.

Preliminary investigations into spectral energy analysis and residual structure evaluation were conducted during this thesis, but failed to yield reliable switching criteria.

## 6.3. Summary of Contributions

This thesis examines the development and evaluation of structure-informed action policies—rules dictating which information to acquire during computation—for iterative probabilistic partial differential equation (PDE) solving within the IterGP framework. This research addresses a fundamental limitation in existing probabilistic numerical methods: the failure of generic action policies to exploit the rich mathematical structure inherent in PDEs.

Building on this foundation, a systematic two-phase approach was developed to explicitly target temporal frequency structure through FFT-based filtering, followed by spatial refinement using established methods. This represents the first successful exploitation of PDE temporal causality within the IterGP framework.

To assess the effectiveness of these methods, a comprehensive evaluation was conducted using heat and wave equations with diverse parameter configurations. The results demonstrate that temporal-first policies achieve reductions in iterations for equivalent uncertainty quantification compared to conjugate gradients baselines.

## 6.4. Final Remarks

This thesis demonstrates that principled exploitation of PDE mathematical structure can yield substantial improvements in probabilistic numerical methods. Whilst not all approaches proved successful, we can conclude that the frequency domain represents a promising avenue for optimising iterative GP-based PDE solvers, particularly through systematic temporal structure targeting.

This work contributes to the broader goal of developing computational tools that respect and leverage the mathematical foundations underlying physical phenomena. The path towards more efficient and reliable probabilistic PDE solving lies not in abandoning established methods, but in thoughtfully augmenting them with structure-aware algorithmic innovations.

# Bibliography

[1] D. Borthwick, *Introduction to Partial Differential Equations*, 1st ed. in Universitext. Springer Cham, 2016, p. xvi + 283. doi: 10.1007/978-3-319-48936-0.

[2] M. Pförtner, I. Steinwart, P. Hennig, and J. Wenger, "Physics-Informed Gaussian Process Regression Generalizes Linear PDE Solvers." [Online]. Available: https://arxiv.org/abs/2212.12474

[3] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* Cambridge, MA: MIT Press, 2006. [Online]. Available: http://www.gaussianprocess.org/gpml

[4] J. Wenger, G. Pleiss, M. Pförtner, P. Hennig, and J. P. Cunningham, "Posterior and Computational Uncertainty in Gaussian Processes," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., Curran Associates, Inc., 2022, pp. 10876–10890. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/4683beb6bab325650db13afd05d1a14a-Paper-Conference.pdf

[5] G. Sanderson, "But what is a partial differential equation? | Differential Equations Chapter 2." [Online]. Available: https://www.youtube.com/watch?v=ly4S0oi3Yz8

[6] L. C. Evans, *Partial Differential Equations*, 2nd ed., vol. 19. in Graduate Studies in Mathematics, vol. 19. Providence, RI, USA: American Mathematical Society, 2010, p. 749.

[7] K. W. Morton and D. F. Mayers, *Numerical Solution of Partial Differential Equations: An Introduction*, 2nd ed. Cambridge University Press, 2005.

[8] J. R. Shewchuk and others, "An introduction to the conjugate gradient method without the agonizing pain," 1994.

[9] T. Strohmer and R. Vershynin, "A Randomized Kaczmarz Algorithm with Exponential Convergence," *Journal of Fourier Analysis and Applications*, vol. 15, no. 2, pp. 262–278, doi: 10.1007/s00041-008-9030-4.

[10] J. P. Boyd, *Chebyshev and Fourier spectral methods.* Courier Corporation, 2001. [Online]. Available: https://depts.washington.edu/ph506/Boyd.pdf

[11] O. Axelsson and I. Kaporin, “On the sublinear and superlinear rate of convergence of conjugate gradient methods,” *Numerical Algorithms*, vol. 25, no. 1, pp. 1–22, doi: 10.1023/A:1016694031362.

[12] T. Weiland, M. Pförtner, and P. Hennig, “Scaling up Probabilistic PDE Simulators with Structured Volumetric Information.” [Online]. Available: https://arxiv.org/abs/2406.05020

# LLM Usage

This thesis used Claude Sonnet 4.0 for grammar correction, spelling correction, and proper language. Only the language, not the underlying content, was modified.

The *GaussPDE.jl* project additions where written with assistance from Claude Sonnet 4.0 as well as GitHub Copilot (Claude Sonnet 3.7). LLMs where used to enhance the understanding of the overall codebase. No major parts were generated on their own.

## Erklärung

Laut Beschlüssen der Prüfungsausschüsse Bioinformatik, Informatik, Informatik Lehramt, Kognitionswissenschaft, Machine Learning, Medieninformatik und Medizininformatik der Universität Tübingen vom 05.02.2025. Gültig für Abschlussarbeiten (B.Sc./M.Sc./B.Ed./M.Ed.) in den zugehörigen Fächern. Bei Studienarbeiten und Hausarbeiten bitte nach Maßgabe des/ der jeweiligen Prüfers/Prüferin.

### 1. Allgemeine Erklärungen

Hiermit erkläre ich:

- ☒ Ich habe die vorgelegte Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt.
- ☒ Ich habe alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet.
- ☒ Die Arbeit war weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens.
- ☐ Falls ich ein elektronisches Exemplar und eines oder mehrere gedruckte und gebundene Exemplare eingereicht habe (z.B., weil der/die Prüfer/in(nen) dies wünschen): Das elektronisch eingereichte Exemplar stimmt exakt mit dem bzw. den von mir eingereichten gedruckten und gebundenen Exemplar(en) überein.

### 2. Erklärung bezüglich Veröffentlichungen

Eine Veröffentlichung ist häufig ein Qualitätsmerkmal (z.B. bei Veröffentlichung in Fachzeitschrift, Konferenz, Preprint, etc.). Sie muss aber korrekt angegeben werden. Bitte kreuzen Sie die für Ihre Arbeit zutreffende Variante an:

- ☒ Die Arbeit wurde bisher weder vollständig noch in Teilen veröffentlicht.
- ☐ Die Arbeit wurde in Teilen oder vollständig schon veröffentlicht. Hierfür findet sich im Anhang eine vollständige Tabelle mit bibliographischen Angaben.

### 3. Nutzung von Methoden der künstlichen Intelligenz (KI, z.B. chatGPT, DeepL, etc.)

Die Nutzung von KI kann sinnvoll sein. Sie muss aber korrekt angegeben werden und kann die Schwerpunkte bei der Bewertung der Arbeit beeinflussen. Bitte kreuzen Sie alle für Ihre Arbeit zutreffenden Varianten an und beachten Sie, dass die Varianten 3.4 - 3.6 eine vorherige Absprache mit dem/der Betreuer/in voraussetzen:

- ☐ **3.1. Keine Nutzung:** Ich habe zur Erstellung meiner Arbeit keine KI benutzt.
- ☒ **3.2. Korrektur Rechtschreibung & Grammatik:** Ich habe KI für Korrekturen der Rechtschreibung und Grammatik genutzt, ohne dass es dabei zu inhaltlich relevanter Textgeneration oder Übersetzungen kam. Das heißt, ich habe von mir verfasste Texte in derselben Sprache korrigieren lassen. Es handelt sich um rein sprachliche Korrekturen, sodass die von mir ursprünglich intendierte Bedeutung nicht wesentlich verändert oder erweitert wurde. Im Zweifelsfall habe ich mich mit meinem/r Betreuer/in besprochen. Alle genutzten Programme mit Versionsnummer sind im Anhang meiner Arbeit in einer Tabelle aufgelistet.
- ☒ **3.3. Unterstützung bei der Softwareentwicklung:** Ich habe KI als Unterstützung beim Schreiben von Code in der Softwareentwicklung genutzt. Es handelt sich hierbei lediglich um Unterstützung und nicht um die automatische Generierung von größeren Programm-

Teilen. Im Zweifelsfall habe ich mich mit meinem/r Betreuer/in besprochen. Alle genutzten Programme mit Versionsnummer sind im Anhang meiner Arbeit in einer Tabelle aufgelistet.

☐ **3.4. Übersetzung:** Ich habe *nach vorheriger Absprache und mit Erlaubnis meines/r Betreuer/in* KI zur Übersetzung von mir in einer anderen Sprache geschriebenen Texte genutzt. Jede derartige Übersetzung ist im laufenden Text gekennzeichnet und der Anhang meiner Arbeit enthält eine Tabelle mit einem vollständigen Nachweis aller übersetzten Textstellen und der verwendeten Programme mit Versionsnummer.

☐ **3.5. Code-Generierung:** Ich habe *nach vorheriger Absprache und mit Erlaubnis meines/r Betreuer/in* KI zur Erzeugung von Code in der Softwareentwicklung genutzt. Der Anhang meiner Arbeit enthält eine Tabelle mit einem vollständigen Nachweis aller derartigen Nutzungen, der verwendeten Programme mit Versionsnummer und der verwendeten Prompts.

☐ **3.6. Text-Generierung:** Ich habe *nach vorheriger Absprache und mit Erlaubnis meines/r Betreuer/in* KI zur Erzeugung von Text in meiner Arbeit genutzt. Jede derartige Verwendung von KI ist im laufenden Text gekennzeichnet und der Anhang meiner Arbeit enthält eine Tabelle mit einem vollständigen Nachweis aller derartigen Nutzungen, der verwendeten Programme mit Versionsnummer und der verwendeten Prompts.

**Falls ich in irgendeiner Form KI genutzt haben (siehe oben), dann erkläre ich:**

Mir ist bewusst, dass ich die Verantwortung trage, falls es durch die Verwendung von KI zu fehlerhaften Inhalten, zu Verstößen gegen das Datenschutzrecht, Urheberrecht oder zu wissenschaftlichem Fehlverhalten (z.B. Plagiaten) kommt.

## 4. Abschluss und Unterschrift(en)

Mir ist bekannt, dass ein Verstoß gegen diese Erklärung prüfungsrechtliche Konsequenzen haben und insbesondere dazu führen kann, dass die Prüfungsleistung mit „nicht ausreichend„ bzw. die Studienleistung mit „nicht bestanden“ bewertet wird und bei mehrfachem oder schwerwiegendem Täuschungsversuch eine Exmatrikulation erfolgen bzw. ein Verfahren zur Entziehung eines eventuell verliehenen akademischen Titels eingeleitet werden kann.

| | | |
|---|---|---|
| | | |
| Vorname, Nachname Student/in | Ort, Datum | Unterschrift |

Die Punkte 3.4 - 3.6 erfordern eine Zustimmung des/r Betreuer/in. Sollten Sie einen dieser Punkte angekreuzt haben, dann sollte der/die Betreuer/in bitte hier unterschreiben:

**Ich habe der oben genannten Nutzung von KI zur Erstellung der Arbeit zugestimmt.**

| | | |
|---|---|---|
| | | |
| Vorname, Nachname Betreuer/in | Ort, Datum | Unterschrift |