EXPLAINABLE SEQUENTIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose formulating stochastic model predictive control into a coalition game to use Shapley values for feature attribution. Such analysis is crucial for transparency and achieving optimal outcomes in high-stake applications such as portfolio optimization and autonomous driving. We categorize Shapley values estimation methods into three families: those based on weighted linear regression, sampling permutations, and multilinear extension. We survey, benchmark, and provide valuable insight into these methods, previously not attempted in this context. Our experiments show that halved Owen sampling from multilinear extension and KernelShap-Paired from weighted linear regression, both utilizing antithetic sampling, perform best.

1 INTRODUCTION

The stochastic model predictive control framework encompasses many sequential decision-making problems (Allgower et al., 2004). It has extensive applications in finance, including portfolio optimization and dynamic option hedging (Primbs, 2018). It also plays a role in operations research, such as supply chain management (Subramanian et al., 2012). Additionally, it applies to robotics and drones, such as quad-copters and manipulators (Limon Daniel., 2010). It also has applications in critical areas of autonomous driving (Batkovic et al., 2022) and technical system automation (Howes et al., 2014). These diverse applications underscore a fundamental requirement: a comprehensive understanding of how each decision and component within the system influences overall performance. This understanding is more than just a technical necessity. It is also a strategic asset in achieving optimal outcomes (Minh et al., 2022).

Portfolio optimization in finance involves complex financial markets and portfolios. They require careful analysis of how each stock affects metrics like the Sharpe ratio, annualized return, and maximum drawdown. Investors and stakeholders may have different levels of financial understanding. Explaining how each stock contributes to the portfolio's overall performance helps them understand the reasoning behind the portfolio's structure. This increases transparency and builds trust between the portfolio manager and the stakeholders (Kuiper et al., 2022).

Imagine a stock with a low return that is important for reducing risk. It shows why diversification is crucial (Koumou, 2020). Explaining the importance of such a stock within a portfolio is challenging. Portfolio optimization frameworks are complex. They include many factors beyond individual stocks. For example, Boyd et al. (2017) describe a framework that involves leverage, turnover, factor limits, and weight limits. They also consider errors in return and risk forecasts. There are different methods to measure risk, such as standard deviation and variance (Markowitz, 1952). Other methods include distributionally robust approaches (Nguyen et al., 2021). These components interact with each other. This makes it hard to explain the effects of a single stock.

We provide an example to clarify the problem. Picture a simple case with the stocks A and B. Stock A has a higher expected return. The system might buy only Stock A because it hits the maximum GMV (Gross Merchandise Value) limit. A trivial explanation method assigns the whole return to Stock A. This approach undermines the potential importance of Stock B. If Stock A is not available, the system buys Stock B. This results in a non-zero return. The correct approach is to consider the potential return of both stocks. The trivial explanation overlooks the interaction between the stocks and the GMV constraint. In practical applications, a system utilizes many more stocks for longer periods. Moreover, more sophisticated performance metrics exist, such as the Sharpe ratio and maximum drawdown.

Despite its importance, there is a significant lack of research on explaining how stochastic model predictive control systems make decisions. We see this from the limited data on metrics like the Google Trends Popularity Index. In contrast, researchers extensively study the field of explainable AI. Figure 1 shows this trend. Motivated by this research gap, we address the problem of additive component attribution in stochastic model predictive control. We define it as follows:

$$g(x) = \phi_0 + \sum_{i=1}^{M} x_i \phi_i,$$

where M is the number of components. They correspond to stocks, constraints, and loss terms in portfolio optimization. $x \in \{0,1\}^M$ is an indicator vector. It shows the presence or absence of each component in the system. Assume f evaluates the system's performance. It is the Sharpe ratio, annualized return, or maximum drawdown in the previous example. g is a simplified version of f with binary inputs. ϕ_0 denotes the baseline performance of the system. It means the performance without any of the M components. $\phi_1 \cdots \phi_M$ indicate the role of each component in the simplified performance metric g.

Among methods for additive attribution, only Shapley values (Shapley, 1953) satisfy desirable properties, including local accuracy, missingness, and consistency (Lundberg & Lee, 2017). We discuss these properties in Appendix A.2.

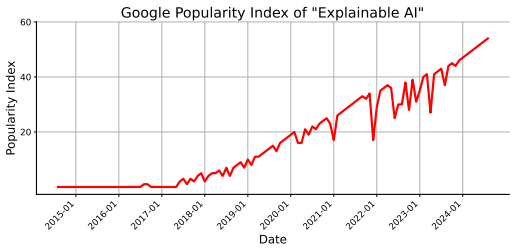


Figure 1: Google Trends Popularity Index of the term "Explainable AI" over the last decade (2014–2024). The maximum value is 100. There is insufficient data to measure the popularity index for terms such as "Explainable Sequential Optimization" or "Explainable Stochastic Model Predictive Control."

Calculating exact Shapley values requires 2^M evaluations of the performance metric. M is the number of components. This calculation considers each subset of components. Various estimation methods address this complexity (Castro et al., 2009; 2017; Lundberg & Lee, 2017; Lomeli et al., 2019; Covert & Lee, 2020; Simon & Vincent, 2020; Burgess & Chapman, 2021; Okhrati & Lipani, 2021; Mitchell et al., 2022). Researchers have not used many of these methods in stochastic model predictive control. This encourages us to benchmark them thoroughly within this context. Our experiments demonstrate that halved Owen sampling (Okhrati & Lipani, 2021) and KernelShap-Paired (Covert & Lee, 2020) achieve the highest performance in this context.

Contributions. The contributions of this paper are as follows: I) To the best of our knowledge, we perform the first systematic study of explaining stochastic model predictive control systems. II) We survey existing methods for estimating Shapley values. We provide valuable insights based on this survey. Then, we benchmark them in the context of stochastic model predictive control.

The structure of the rest of the paper is as follows: we provide the related work in Section 2. We discuss stochastic model predictive control and its explainability in Section 3 and Section 4, respectively. Section 5 provides the details of the experiments. We conclude the paper in Section 6.

2 Related Work

Local Model-Agnostic Explanation. LIME (Ribeiro et al., 2016) uses user-defined binary features around a sample. These binary features are simpler and depend on the original features. LIME samples random points in the neighborhood. It assigns weights based on their distance from the original point. It then fits a weighted linear regression model on these points to provide an explanation. If the

binary features show whether the original features are present or absent, LIME's feature attributions match Shapley values. This happens when the sample weights follow a specific equation (Lundberg & Lee, 2017).

The Shapley value of a missing feature is zero. The contrastive explanation method (Dhurandhar et al., 2018) emphasizes that absent features play an essential role in explanations. It defines two sets of features: pertinent positive and pertinent negative. Pertinent positive features are the minimal features that must be present to achieve the current output. Pertinent negative features must be absent for the current output to hold.

Ribeiro et al. (2018) define anchors as rules based on key features of a sample. If these rules apply, the output for that sample is likely to remain consistent. This holds even if other feature values change. Anchors resemble pertinent positive features in the contrastive explanation method. Anchors make explanations more stable across different samples. However, they potentially become overly complex. This complexity makes the explanations harder to understand and less insightful.

Explainability in Stochastic Model Predictive Control. Agogino et al. (2019) explore various methods to create explanations for a neural network controller. These methods include Bayesian rule lists (Letham et al., 2015), function analysis, single time step integrated gradients (Sundararajan et al., 2017), grammar-based decision trees (Lee et al., 2018) sensitivity analysis combined with temporal modeling with LSTMs (Ribeiro et al., 2016), and explanation templates.

Quade et al. (2020) build upon earlier research on symbolic regression techniques (Vladislavleva et al., 2008; Schmidt & Lipson, 2009; Quade et al., 2016) to determine the optimal control strategies for dynamic systems based on various optimization criteria or cost functions. Their findings highlight a significant benefit of using interpretable symbolic regression models compared to more opaque neural network approaches.

Dang et al. (2021) utilize existing knowledge of the common bang-bang characteristics in low-input penalized model predictive control to approximate the control law through sparse state space sampling. This approach involves employing support vector machines to identify the switching surface within the sampled model predictive control solution.

Jorissen et al. (2021) use a white-box modeling approach to achieve explainability. They explicitly represent the physical and thermodynamic principles governing the heating, ventilation, and air conditioning system. This approach makes the system's behavior easier to understand. The model directly links input variables like temperature setpoints and weather conditions to the system's outputs and control actions.

Stevens & De Smedt (2022) explore explainability in process outcome prediction by emphasizing two key aspects: interpretability and faithfulness. They develop metrics to assess these aspects across the main dimensions of process data. This approach compares inherently generated explanations and those produced using post-hoc methods. Additionally, the authors provide guidelines to aid in model selection. They highlight the impact of different preprocessing steps, model complexities, and explainability techniques on the overall explainability of the model.

Salehi & Doncieux (2024) incorporate prior knowledge of environment kinematics and dynamics to improve explainability. They achieve this by simplifying the problem and reducing the need for exploration. This approach expresses the agent's decisions in terms of physically meaningful entities.

Explainability With Shapley Values. Previous research applies Shapley values to a wide range of problems. For instance, Moretti et al. (2008) combine Shapley values and statistics to analyze gene expression data. Maleki (2015) estimate Shapley values using stratified sampling and apply them to smart grid management. Tarashev et al. (2016) use Monte Carlo simulation to estimate Shapley values for risk attribution. Landinez-Lamadrid et al. (2017) propose an optimized algorithm for Shapley values estimation and apply it to supply chain management. Marcilio-Jr & Eler (2021) employ KernelShap in dimensionality reduction. Finally, Bertossi et al. (2023) utilize exact Shapley values in database management.

3 STOCHASTIC MODEL PREDICTIVE CONTROL

A stochastic model predictive control system consists of system dynamics and the control problem. We denote system dynamics with f. Its inputs are the current state x_t and control input u_t^*. It predicts the system's next state x_{t+1}. Equation 1 provides its formulation.

$$x_{t+1} = f\left(u_t^*(\theta), x_t; \xi_t\right) \quad t = 0, \ldots, T-1, \tag{1}$$

where θ is a hyperparameter with a feasible set Θ. $\xi_t \sim \Xi_t$ specifies a stochastic variable. Let $\hat{x}_{\tau|t}$ indicate the estimate of x_τ at time t. Denote $\hat{f}$ as the approximate dynamics. One obtains the control input u_t^* by solving Problem 2. One solves this problem for each time step t from 0 to $T-1$.

$$\begin{aligned} u_{t|t}^*(\theta), \ldots, u_{t+H-1|t}^*(\theta) \in \underset{u_{t|t}, \ldots, u_{t+H-1|t}}{\arg\min} \quad & \sum_{\tau=t}^{t+H-1} \mathop{\mathbb{E}}_{\xi_\tau \sim \Xi_\tau} \left[q\left(\theta, u_{\tau|t}, \hat{x}_{\tau|t}; \xi_\tau\right)\right] \\ \text{subject to} \quad & \hat{x}_{\tau+1|t} = \hat{f}\left(u_{\tau|t}, \hat{x}_{\tau|t}; \xi_\tau\right), \\ & u_{\tau|t} \in U\left(\theta; \xi_\tau\right), \end{aligned} \tag{2}$$

where $u_t^*(\theta) = u_{t|t}^*(\theta)$. U is the feasible set for control input. q denotes the control loss. Note that one does not use $u_{t+1|t}^*(\theta), \ldots, u_{t+H-1|t}^*(\theta)$. One optimizes over them only for future planning. One obtains $x = [x_0, \ldots, x_T]$, $u^*(\theta) = [u_0^*(\theta), \ldots, u_{T-1}^*(\theta)]$ after solving Problem 2 for t from 0 to $T-1$. Let $\xi = [\xi_0, \ldots, \xi_T]$. A performance metric $p\left(u^*(\theta), x; \xi\right)$ evaluates the system. Adding optional loss terms or constraints to the system potentially improves performance. For instance, one often adds a risk term because the problem is stochastic. Examples include standard deviation, variance, and distributionally robust measures.

We study the setting where the hyperparameters θ indicate the presence or absence of optional loss terms and constraints in Problem 2. This problem has various applications in areas such as robotics (AlAttar et al., 2022; Song & Scaramuzza, 2022), operations research (Schwenzer et al., 2021), and finance (Boyd et al., 2017). Specifically, $\theta = [\theta^{(c)}, \theta^{(l)}] \in \{0,1\}^{|\theta|}$. $\theta^{(c)}$ and $\theta^{(l)}$ denote the presence or absence of optional constraints and loss terms, respectively.

Let $U^{(o)} = [U_0^{(o)} \cdots U_{|\theta^{(c)}|-1}^{(o)}]$ denote the feasible set corresponding to optional constraints. $|\theta^{(c)}|$ is the number of optional constraints. The feasible set of included optional constraints is $\cap_{\{i:\theta_i^{(c)}=1\}} U_i^{(o)}$. $\{i : \theta_i^{(c)} = 1\}$ indicates the set of indices i of $\theta^{(c)}$ for which $\theta_i^{(c)} = 1$. Let $U^{(m)}$ denote the feasible set associated with mandatory constraints. The feasible set corresponding to all used constraints is $U(\theta) = U^{(m)} \cap \left(\cap_{\{i:\theta_i^{(c)}=1\}} U_i^{(o)}\right)$.

We use $q^{(m)}$ to denote the summation of mandatory loss terms. $\tilde{q}^{(o)} = [q_0^{(o)} \ldots q_{|\theta^{(l)}|-1}^{(o)}]$ indicates the vector of optional loss terms. $|\theta^{(l)}|$ is the number of optional loss terms. The final control loss is $q = q^{(m)} + \theta^{(l)\top} \tilde{q}^{(o)}$. Problem 3 shows the updated control problem.

$$\begin{aligned} u_{t|t}^*(\theta), \ldots, u_{t+H-1|t}^*(\theta) \in \\ \underset{u_{t|t}, \ldots, u_{t+H-1|t}}{\arg\min} \quad & \sum_{\tau=t}^{t+H-1} \mathop{\mathbb{E}}_{\xi_\tau \sim \Xi_\tau} \left[q^{(m)}\left(u_{\tau|t}, \hat{x}_{\tau|t}; \xi_\tau\right) + \theta^{(l)\top} \tilde{q}^{(o)}\left(u_{\tau|t}, \hat{x}_{\tau|t}; \xi_\tau\right)\right] \\ \text{subject to} \quad & \hat{x}_{\tau+1|t} = \hat{f}\left(u_{\tau|t}, \hat{x}_{\tau|t}; \xi_\tau\right), \\ & u_{\tau|t} \in U^{(m)}\left(\xi_\tau\right) \cap \left(\cap_{\{i:\theta_i^{(c)}=1\}} U_i^{(o)}\left(\xi_\tau\right)\right) \end{aligned} \tag{3}$$

4 EXPLAINABLE STOCHASTIC MODEL PREDICTIVE CONTROL

We aim to explain how each component in Problem 3 affects a performance metric p. $\theta = [\theta^{(c)}, \theta^{(l)}]$ represents the optional components. $\theta^{(c)}, \theta^{(l)}$ corresponds to the optional constraints and loss term respectively. These components interact with each other. Hence, explaining their effect is highly non-trivial. We propose to utilize Shapley values for this purpose. They consider all subsets of components to account for their interaction. We discuss Shapley values in detail in Appendix A. Shapley values are the only additive attribution method that satisfies local accuracy, missingness, and consistency (Lundberg & Lee, 2017). We discuss these properties in Appendix A.2. We formulate our problem into a coalition game and provide the mapping in Table 1.

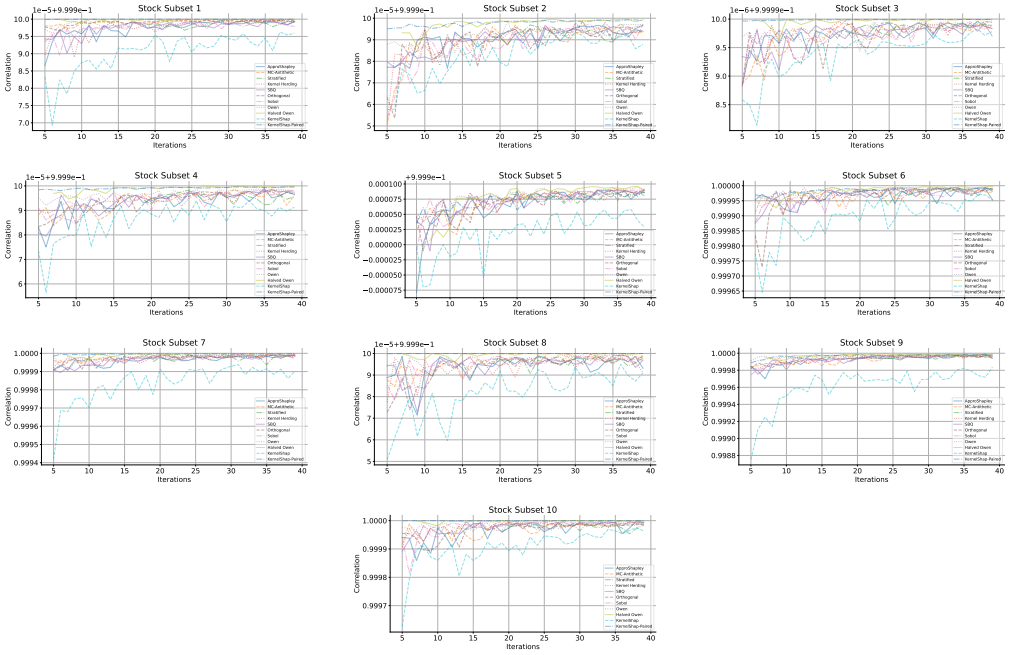


Figure 2: Comparison of the estimators based on their correlation with exact Shapley values. The higher the correlation, the better. We take the mean over five runs. Annualized return is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. Some estimators require a minimum number of samples. We do not report their performance for the number of samples below their minimum.

There are several formulations for the exact computation of Shapley values. We provide them in Appendix A.1. They require $2^{|\theta|}$ evaluations, where $|\theta|$ is the number of components. Several methods estimate Shapley values. We discuss these methods in detail in Appendix B. We categorize them based on the formulation they estimate as follows:

Table 1: Mapping between a coalition game and an SMPC (Stochastic Model Predictive Control) system.

Coalition Game	SMPC
Player i	θ_i
Payoff function v	Performance metric p
Presence of player i in the coalition	$\theta_i = 1$
Absence of player i in the coalition	$\theta_i = 0$
The i-th Shapley value ϕ_i	Effect of θ_i on p

- Weighted linear regression: KernelShap (Lundberg & Lee, 2017), KernelShap-Paired (Covert & Lee, 2020), and SGD-Shapley (Simon & Vincent, 2020).
- Sampling permutations: ApproShapley (Castro et al., 2009), stratified sampling with Neyman allocation (Castro et al., 2017), antithetic sampling (Lomeli et al., 2019), improved Monte Carlo simulation (Simon & Vincent, 2020), stratified Bernstein sampling (Burgess & Chapman, 2021), kernel herding (Mitchell et al., 2022), sequential Bayesian quadrature (Mitchell et al., 2022), orthogonal spherical codes (Mitchell et al., 2022), and Sobol sequences on the sphere (Mitchell et al., 2022).
- Multilinear extension: Owen sampling (Okhrati & Lipani, 2021) and halved Owen sampling (Okhrati & Lipani, 2021).

Researchers have not applied many of these methods to stochastic model predictive control. So their performance in this context is unknown. This motivates us to thoroughly benchmark them in the next section.

5 Experiments

Stochastic Model Predictive Control System. We perform our experiments in portfolio optimization (Boyd et al., 2017). There are multiple performance metrics with different levels of complexity

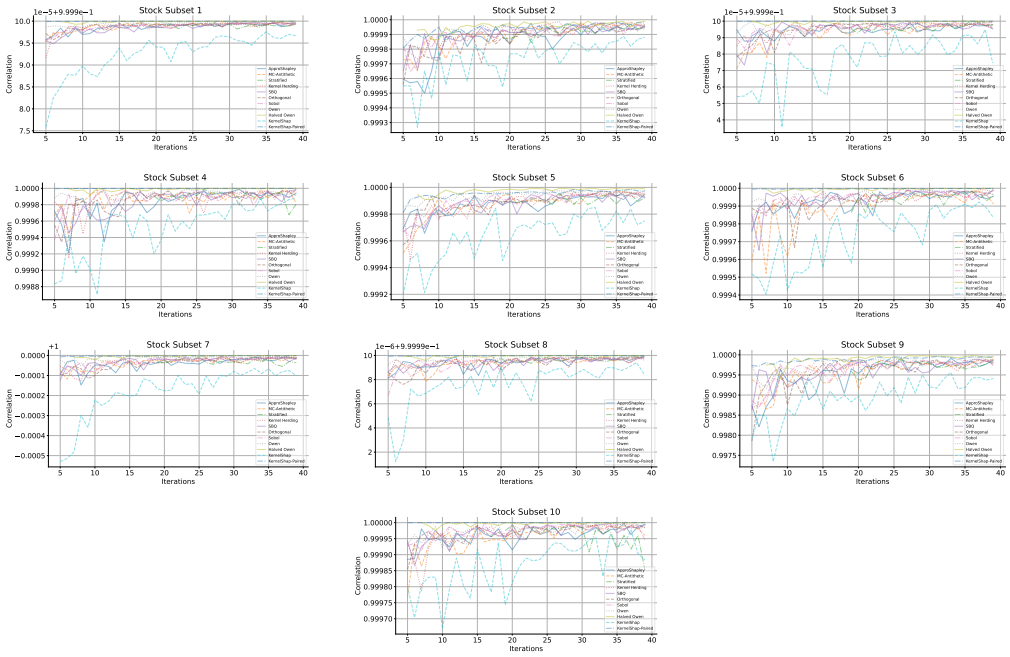


Figure 3: Comparison of the estimators based on correlation with exact Shapley values. The higher the correlation, the better. We take the mean over five runs. Sharpe ratio is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. Some estimators require a minimum number of samples. We do not report their performance for the number of samples below their minimum.

in this context. Examples include the Sharpe ratio and annualized return. We use the `cvxportfolio` package (Busseti et al., 2017) to generate a real-world stochastic model predictive control system. Various components in our system complicate the interactions between the stocks. They include full covariance risk measure, min-cash weight and long-only constraints, and leverage, turnover, weights, and factors limits. The system has 30 steps.

Optional Components. We aim to explain the effect of each stock on each performance metric. In our framework, we consider loss terms and constraints as components. We use no-trade constraints to formulate stocks into components in our framework. We retrieve data of 100 stocks from `yfinance` . In this context, computing exact Shapley values for 100 players is impractical. It requires 2^{100} evaluations. Moreover, each evaluation requires solving Problem 3 for $T = 30$ periods. To circumvent this issue, we design ten coalition games. In each game, we consider only ten stocks as optional components. The optional components are the players in the coalition game. We maintain the other 90 stocks in the system to complicate interactions. These stocks are mandatory components of the system. These real-world settings highlight why we need a comprehensive benchmark of the estimation methods in this context.

Metrics. We define two metrics to evaluate the estimation methods: Spearman correlation coefficient and mean squared error. We compute these values between the exact and estimated Shapley values. We repeat the experiments five times and report the mean and standard deviation.

Results. Figures 2 to 5 show the performance of the Shapley value estimators across 40 iterations. Tables 2 to 5 provide the mean and standard deviation after 20 and 40 iterations. Other estimators heavily outperform SGD-Shapley (Simon & Vincent, 2020). Hence, we do not include it in the figures. On average, the halved Owen (Okhrati & Lipani, 2021) has the highest performance across all experiments. KernelShap-Paired (Covert & Lee, 2020) follows it.

5.1 Discussion

WLR-Based Sampling. KernelShap (Lundberg & Lee, 2017) and SGD-Shapley (Simon & Vincent, 2020) use weighted linear regression (WLR) to estimate Shapley values. They differ in solving the weighted linear regression problem. We discuss this problem in Appendix B.1. One of the

Table 2: Comparison of the estimators based on mean squared error against exact Shapley values after 20 iterations. The lower the error, the better. We take the mean and standard deviation over five runs. Annualized return is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. We highlight the lowest value for each subset.

	Stock Subset 1	Stock Subset 2	Stock Subset 3	Stock Subset 4	Stock Subset 5	Stock Subset 6	Stock Subset 7	Stock Subset 8	Stock Subset 9	Stock Subset 10
Monte Carlo	1.03e-08±7.21e-09	2.91e-11±1.73e-11	4.69e-11±1.91e-11	2.00e-08±9.39e-09	8.48e-08±4.93e-08	2.65e-09±2.26e-09	1.45e-09±1.21e-09	1.00e-10±1.12e-10	4.35e-08±1.52e-08	9.61e-06±7.16e-06
Monte Carlo-Anti	2.81e-09±1.28e-09	9.41e-12±5.72e-12	2.25e-11±1.27e-11	1.19e-08±3.63e-09	4.84e-08±2.90e-08	4.34e-09±4.59e-09	1.23e-09±7.66e-10	7.14e-11±3.84e-11	3.85e-08±2.60e-08	1.92e-06±1.69e-06
Stratified	4.88e-09±2.70e-09	2.86e-11±3.21e-11	2.60e-11±1.58e-11	2.83e-08±1.32e-08	3.86e-08±2.31e-08	1.30e-09±1.21e-09	1.85e-09±1.22e-09	6.12e-11±2.54e-11	5.46e-08±2.36e-08	7.36e-06±4.75e-06
Kernel Herding	7.31e-09±3.85e-09	2.27e-11±2.11e-11	9.42e-12±6.72e-12	2.08e-08±6.61e-09	5.87e-08±2.61e-08	2.70e-09±2.82e-09	9.20e-10±6.54e-10	8.09e-11±6.67e-11	2.97e-08±1.99e-08	5.01e-06±3.72e-06
SBQ	1.10e-08±7.62e-09	2.16e-11±1.66e-11	4.30e-11±2.06e-11	1.79e-08±1.62e-08	5.95e-08±2.71e-08	2.27e-09±1.20e-09	7.14e-10±2.74e-10	7.76e-11±6.15e-11	5.97e-08±2.45e-08	3.13e-06±2.12e-06
Orthogonal	2.37e-09±9.05e-10	1.56e-11±1.50e-11	1.83e-11±1.06e-11	2.71e-08±1.86e-08	5.44e-08±3.03e-08	3.50e-09±2.50e-09	6.67e-10±2.39e-10	5.49e-11±5.76e-11	3.75e-08±1.12e-08	3.90e-06±1.68e-06
Sobol	4.72e-09±4.17e-09	2.88e-11±1.24e-11	3.41e-11±1.90e-11	1.52e-08±9.84e-09	7.14e-08±3.46e-08	1.92e-09±1.56e-09	4.71e-10±3.51e-10	3.05e-11±1.91e-11	8.30e-08±7.44e-08	4.41e-06±3.38e-06
Owen	3.70e-09±3.65e-09	1.59e-11±6.64e-12	4.33e-11±3.06e-11	1.28e-08±6.83e-09	3.09e-08±8.50e-09	2.97e-09±1.76e-09	8.99e-10±5.83e-10	3.22e-11±2.06e-11	2.33e-08±1.14e-08	3.81e-06±1.87e-06
Owen-Anti	3.18e-11±7.91e-12	3.84e-12±3.22e-12	8.19e-13±3.46e-13	1.70e-09±6.75e-10	2.17e-08±9.31e-09	6.65e-10±3.77e-10	7.28e-11±3.28e-11	3.05e-13±1.52e-13	1.16e-08±1.27e-08	1.10e-08±6.54e-09
WLR	1.81e-08±5.58e-09	2.62e-11±1.34e-11	5.90e-11±3.10e-11	3.14e-08±1.78e-08	1.75e-07±1.12e-07	4.56e-09±2.30e-09	6.90e-09±2.52e-09	1.60e-10±9.80e-11	2.38e-07±1.22e-07	1.07e-05±5.91e-06
WLR-Anti	3.01e-11±1.26e-11	3.76e-12±9.94e-13	1.60e-12±1.10e-12	2.62e-09±6.66e-10	7.17e-08±5.05e-08	8.50e-10±2.21e-10	1.65e-10±1.35e-10	1.76e-13±9.71e-14	2.04e-08±9.77e-09	2.18e-08±7.11e-09
SGD-Shapley	4.90e-04±1.02e-04	3.97e-07±1.49e-07	1.80e-05±5.93e-06	6.00e-04±1.89e-04	4.73e-04±1.71e-04	1.10e-05±3.42e-06	5.46e-06±1.08e-06	1.34e-06±4.58e-07	1.64e-04±4.16e-05	3.34e-02±7.44e-03

Table 3: Comparison of the estimators based on mean squared error against exact Shapley values after 40 iterations. The lower the error, the better. We take the mean and standard deviation over five runs. Annualized return is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. We highlight the lowest value for each subset.

	Stock Subset 1	Stock Subset 2	Stock Subset 3	Stock Subset 4	Stock Subset 5	Stock Subset 6	Stock Subset 7	Stock Subset 8	Stock Subset 9	Stock Subset 10
Monte Carlo	2.52e-09±1.28e-09	7.44e-12±1.60e-12	2.82e-11±1.00e-11	1.03e-08±8.91e-09	2.93e-08±1.90e-08	3.66e-09±1.81e-09	4.76e-10±2.98e-10	4.55e-11±3.59e-11	4.81e-08±4.31e-08	1.43e-06±7.48e-07
Monte Carlo-Anti	1.54e-09±4.95e-10	2.15e-11±1.49e-11	2.08e-11±2.46e-11	1.28e-08±7.70e-09	2.70e-08±1.47e-08	1.22e-09±7.14e-10	2.19e-10±1.99e-10	2.65e-11±2.60e-11	2.45e-08±1.49e-08	3.52e-06±1.36e-06
Stratified	4.17e-09±2.34e-09	1.40e-11±1.13e-11	1.91e-11±1.09e-11	1.54e-08±1.42e-08	4.65e-08±1.75e-08	1.84e-09±1.87e-09	9.21e-10±3.99e-10	2.63e-11±1.58e-11	4.31e-08±1.67e-08	3.67e-06±3.22e-06
Kernel Herding	4.94e-09±5.18e-09	1.05e-11±8.90e-12	9.48e-12±7.47e-12	1.05e-08±5.35e-09	3.31e-08±1.42e-08	1.01e-09±1.12e-09	4.87e-10±2.26e-10	7.44e-11±6.56e-11	1.93e-08±6.11e-09	2.83e-06±9.60e-07
SBQ	2.11e-09±1.18e-09	1.23e-11±1.41e-11	2.91e-11±2.40e-11	8.83e-09±6.48e-09	2.13e-08±8.82e-09	8.12e-10±8.50e-10	5.58e-10±2.00e-10	3.18e-11±1.99e-11	3.05e-08±1.31e-08	1.60e-06±1.36e-06
Orthogonal	1.15e-09±4.69e-10	1.27e-11±1.41e-11	1.82e-11±1.18e-11	1.33e-08±5.99e-09	3.06e-08±1.75e-08	5.60e-10±2.20e-10	2.73e-10±1.00e-10	1.75e-11±1.47e-11	2.13e-08±1.22e-08	3.99e-06±3.46e-06
Sobol	4.68e-09±3.88e-09	6.57e-12±5.53e-12	1.09e-11±5.98e-12	6.34e-09±1.11e-09	2.47e-08±1.44e-08	1.08e-09±5.18e-10	4.11e-10±2.26e-10	6.87e-11±6.37e-11	2.18e-08±8.76e-09	1.73e-06±1.86e-06
Owen	2.35e-09±2.27e-09	6.47e-12±5.62e-12	1.10e-11±7.12e-12	4.98e-09±2.57e-09	2.58e-08±1.13e-08	1.60e-09±1.06e-09	3.76e-10±2.36e-10	1.89e-11±1.54e-11	4.98e-09±1.91e-09	1.83e-06±1.17e-06
Owen-Anti	1.27e-11±5.56e-12	1.88e-12±4.00e-13	1.22e-12±5.53e-13	8.74e-10±1.82e-10	2.17e-08±1.77e-08	2.38e-10±1.18e-10	2.19e-11±1.48e-11	9.86e-14±6.97e-14	1.02e-08±3.10e-09	6.18e-09±3.55e-09
WLR	9.40e-09±5.15e-09	2.88e-11±1.12e-11	4.59e-11±4.29e-11	2.96e-08±1.19e-08	1.47e-07±3.15e-08	3.10e-09±8.26e-10	3.83e-09±1.63e-09	6.75e-11±3.21e-11	1.12e-07±2.36e-08	7.06e-06±4.31e-06
WLR-Anti	1.11e-11±5.58e-12	2.10e-12±8.67e-13	5.03e-13±1.71e-13	1.25e-09±3.25e-10	3.74e-08±2.07e-08	6.45e-10±2.20e-10	5.43e-11±1.44e-11	1.37e-13±3.98e-14	1.58e-08±6.92e-09	1.40e-08±7.77e-09
SGD-Shapley	3.25e-04±7.69e-05	2.98e-07±5.42e-08	1.60e-05±4.55e-06	4.23e-04±9.76e-05	1.99e-04±3.46e-05	8.20e-06±1.75e-06	3.41e-06±8.70e-07	1.14e-06±3.43e-07	9.70e-05±1.92e-05	2.94e-02±5.62e-03

challenges in this problem is dealing with infinite weights. KernelShap addresses this by replacing infinite weights with a large constant value, specifically 10^6. After this substitution, KernelShap solves the problem using the closed-form solution. On the other hand, SGD-Shapley uses projected stochastic gradient. It performs a single iteration of this algorithm after each evaluation. Hence, it requires more samples for convergence. Using stochastic gradient descent instead of the closed-form solution is standard in high-dimensional problems. In such problems, inverting matrices is a computational bottleneck. However, this situation does not apply when estimating Shapley values. Here, the primary concern is sample efficiency.

Owen Sampling vs. Stratified Sampling. Owen sampling (Okhrati & Lipani, 2021) estimates Shapley values using the multilinear extension. Stratified sampling (Castro et al., 2017; Burgess & Chapman, 2021) estimates Shapley values using permutations. We discuss Owen sampling in Appendix B.3 and stratified sampling in Appendix B.2.3. These methods use different formulations of Shapley values but are very similar in concept. Stratified sampling partitions the permutation space into subspaces called strata. The goal is to make the samples within each stratum as similar as possible. At the same time, one wants the samples in different strata to be as different as possible. When estimating Shapley values, one defines the strata based on the coalition size in the samples. Owen sampling estimates the Shapley value for the i-th player using $\phi_i = \int_0^1 e_i(q)\, dq$. Here, $e_i(q)$ is the expected marginal contribution of player i to a coalition. This expectation assumes that each player has a probability q of being included in the coalition. In practice, one approximates this integral using a Riemann sum. One evaluates the Riemann sum at points $q = 0, \frac{1}{Q}, \cdots, 1$. The parameter Q controls how accurate the Riemann sum is. Each value of q in Owen sampling corresponds to the expected number of players in a coalition. This matches a stratum in stratified sampling.

Antithetic Sampling. Antithetic sampling is a variance reduction technique in Monte Carlo simulation (Rubinstein & Kroese, 2016; Lomeli et al., 2019). It chooses negatively correlated samples instead of independent and identically distributed samples. The negative correlation between sample pairs balances out extremes in the sampling distribution. This ensures more uniform coverage of the space and reduces the overall variance of the estimate. We conduct an ablation study on all three families of Shapley value estimation methods: weighted linear regression, sampling permutations, and multilinear extension. Our experiments reveal that antithetic sampling significantly enhances the performance of the baseline methods in the weighted linear regression and multilinear extension approaches. Figure 6 provides the results.

Table 4: Comparison of the estimators based on mean squared error against exact Shapley values after 20 iterations. The lower the error, the better. We take the mean and standard deviation over five runs. Sharpe ratio is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. We highlight the lowest value for each subset.

	Stock Subset 1	Stock Subset 2	Stock Subset 3	Stock Subset 4	Stock Subset 5	Stock Subset 6	Stock Subset 7	Stock Subset 8	Stock Subset 9	Stock Subset 10
Monte Carlo	7.26e-07±4.37e-07	4.86e-09±2.97e-09	1.59e-07±1.84e-07	1.45e-04±1.52e-04	8.01e-05±3.63e-05	3.30e-06±2.41e-06	2.42e-07±1.17e-07	7.91e-10±6.59e-10	2.76e-05±1.09e-05	1.54e-03±1.56e-03
Monte Carlo-Anti	4.46e-07±2.20e-07	4.11e-09±1.50e-09	1.97e-07±1.03e-07	8.00e-05±7.92e-05	4.64e-05±2.54e-05	1.90e-06±4.88e-07	1.28e-07±5.40e-08	5.98e-10±3.20e-10	1.13e-05±7.32e-06	2.19e-03±1.74e-03
Stratified	7.77e-07±5.44e-07	4.84e-09±3.63e-09	1.69e-07±5.18e-08	5.92e-05±5.48e-05	5.46e-05±2.69e-05	3.79e-06±2.46e-06	1.81e-07±1.21e-07	9.26e-10±8.09e-10	1.80e-05±1.20e-05	1.75e-03±7.66e-04
Kernel Herding	1.07e-06±4.06e-07	3.17e-09±1.85e-09	1.68e-07±1.33e-07	4.10e-05±2.36e-05	5.50e-05±4.07e-05	1.21e-06±8.67e-07	7.14e-08±3.50e-08	7.29e-10±3.45e-10	1.98e-05±1.40e-05	8.31e-04±9.44e-04
SBQ	6.82e-07±5.38e-07	4.19e-09±4.43e-09	1.47e-07±1.10e-07	9.33e-05±6.11e-05	6.80e-05±2.10e-05	1.38e-06±8.21e-07	7.80e-08±6.19e-08	1.67e-09±9.97e-10	1.51e-05±7.87e-06	1.03e-03±1.05e-03
Orthogonal	4.62e-07±2.19e-07	4.99e-09±4.56e-09	2.60e-07±2.56e-07	6.62e-05±5.78e-05	7.53e-05±3.90e-05	1.82e-06±9.21e-07	1.47e-07±1.04e-07	9.93e-10±1.21e-09	7.40e-06±1.67e-06	1.52e-03±9.25e-04
Sobol	2.34e-07±1.40e-07	2.81e-09±2.66e-09	1.30e-07±1.44e-07	5.72e-05±4.57e-05	5.69e-05±2.56e-05	1.90e-06±1.26e-06	8.25e-08±2.63e-08	8.17e-10±5.88e-10	1.26e-05±8.81e-06	1.37e-03±1.42e-03
Owen	2.15e-07±1.65e-07	3.09e-09±2.56e-09	8.12e-08±8.39e-08	5.48e-05±4.37e-05	1.95e-05±4.76e-06	1.65e-06±8.23e-07	5.02e-08±3.75e-08	4.09e-10±2.65e-10	9.11e-06±6.58e-06	1.80e-03±8.92e-04
Owen-Anti	6.12e-09±4.59e-09	1.18e-09±5.64e-10	3.75e-11±3.15e-11	4.73e-07±1.68e-07	1.17e-05±5.91e-06	4.77e-08±1.60e-08	3.80e-09±2.26e-09	2.22e-11±1.14e-11	3.92e-06±2.20e-06	5.39e-07±3.87e-07
WLR	2.02e-06±9.89e-07	1.62e-08±8.91e-09	3.84e-07±2.72e-07	5.14e-04±2.76e-04	2.08e-04±5.89e-05	8.97e-06±7.62e-06	7.73e-07±3.62e-07	3.43e-09±2.50e-09	5.59e-05±3.15e-05	6.19e-03±3.33e-03
WLR-Anti	5.78e-09±1.76e-09	1.90e-09±5.25e-10	1.13e-10±4.38e-11	4.37e-07±5.44e-08	2.36e-05±5.37e-06	4.77e-08±8.60e-09	9.13e-09±3.56e-09	1.87e-11±4.59e-12	8.36e-06±4.60e-06	1.04e-06±1.08e-07
SGD-Shapley	5.94e-02±2.43e-02	7.28e-06±1.83e-06	4.73e-03±7.66e-04	9.90e-02±2.21e-02	9.17e-02±1.75e-02	4.81e-03±7.66e-04	8.51e-04±2.09e-04	1.71e-04±2.78e-05	1.09e-02±3.46e-03	5.58e+00±1.14e+00

Table 5: Comparison of the estimators based on mean squared error against exact Shapley values after 40 iterations. The lower the error, the better. We take the mean and standard deviation over five runs. Sharpe ratio is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. We highlight the lowest value for each subset.

	Stock Subset 1	Stock Subset 2	Stock Subset 3	Stock Subset 4	Stock Subset 5	Stock Subset 6	Stock Subset 7	Stock Subset 8	Stock Subset 9	Stock Subset 10
Monte Carlo	3.33e-07±2.78e-07	2.15e-09±1.61e-09	8.91e-08±3.49e-08	4.33e-05±4.26e-05	2.26e-05±3.68e-06	1.05e-06±6.05e-07	6.91e-08±3.63e-08	1.79e-10±2.29e-10	8.15e-06±7.30e-06	5.74e-04±7.31e-04
Monte Carlo-Anti	3.55e-07±1.67e-07	2.08e-09±1.38e-09	1.22e-07±5.19e-08	9.81e-05±7.74e-05	3.27e-05±2.17e-05	2.24e-06±2.02e-06	5.66e-08±2.99e-08	1.68e-10±1.09e-10	7.66e-06±2.17e-06	1.55e-03±7.74e-04
Stratified	9.03e-07±9.65e-07	2.17e-09±1.96e-09	8.27e-08±3.18e-08	1.04e-04±7.09e-05	3.94e-05±2.35e-05	1.52e-06±9.83e-07	1.42e-07±9.72e-08	1.69e-10±8.85e-11	1.04e-05±7.60e-06	4.31e-03±2.42e-03
Kernel Herding	2.46e-07±9.24e-08	2.36e-09±1.17e-09	9.35e-08±8.61e-08	4.64e-05±1.64e-05	2.62e-05±1.96e-05	8.33e-07±4.55e-07	9.21e-08±6.43e-08	4.12e-10±1.63e-10	1.07e-05±7.78e-06	7.72e-04±5.32e-04
SBQ	3.40e-07±1.34e-07	1.64e-09±6.85e-10	1.12e-07±8.86e-08	3.03e-05±2.24e-05	5.83e-05±4.37e-05	6.57e-07±4.39e-07	4.51e-08±2.73e-08	5.91e-10±3.48e-10	1.00e-05±5.98e-06	2.54e-04±2.07e-04
Orthogonal	1.65e-07±1.06e-07	1.92e-09±1.26e-09	1.20e-07±1.18e-07	5.94e-05±4.59e-05	2.98e-05±1.64e-05	8.88e-07±4.27e-07	6.48e-08±3.95e-08	2.26e-10±1.85e-10	1.34e-05±1.42e-05	6.25e-04±4.74e-04
Sobol	2.33e-07±2.04e-07	1.83e-09±1.83e-09	6.38e-08±5.92e-08	8.62e-06±3.68e-06	2.19e-05±9.67e-06	7.60e-07±5.27e-07	8.61e-08±6.19e-08	4.53e-10±3.09e-10	6.84e-06±2.72e-06	1.46e-03±9.56e-04
Owen	2.18e-07±8.17e-08	1.64e-09±5.82e-10	2.93e-08±1.24e-08	1.59e-05±1.14e-05	1.07e-05±6.25e-06	2.84e-07±8.92e-08	3.72e-08±8.35e-09	1.70e-10±1.62e-10	6.91e-06±4.34e-06	5.06e-04±1.82e-04
Owen-Anti	2.86e-09±1.50e-09	1.00e-09±1.05e-09	3.05e-11±1.14e-11	2.78e-07±1.90e-07	6.61e-06±2.05e-06	2.00e-08±1.17e-08	1.15e-09±4.11e-10	1.18e-11±4.84e-12	2.39e-06±1.31e-06	7.03e-07±1.92e-07
WLR	1.49e-06±1.35e-06	4.28e-09±2.89e-09	4.49e-07±3.46e-07	1.15e-04±6.42e-05	1.49e-04±1.29e-04	3.68e-06±1.28e-06	4.09e-07±1.81e-07	1.44e-09±7.29e-10	3.04e-05±5.55e-06	3.25e-03±1.05e-03
WLR-Anti	4.16e-09±1.04e-09	8.69e-10±2.92e-10	4.28e-11±1.41e-11	3.23e-07±6.64e-08	1.37e-05±7.52e-06	2.74e-08±9.08e-09	4.70e-09±7.93e-10	1.13e-11±3.16e-12	3.38e-06±2.10e-06	5.21e-07±2.60e-07
SGD-Shapley	4.49e-02±2.23e-02	6.83e-06±7.47e-07	3.12e-03±9.13e-04	4.59e-02±1.94e-02	8.21e-02±1.26e-02	2.64e-03±7.33e-04	4.21e-04±1.82e-04	1.03e-04±2.95e-05	6.56e-03±2.06e-03	1.87e+00±1.06e+00

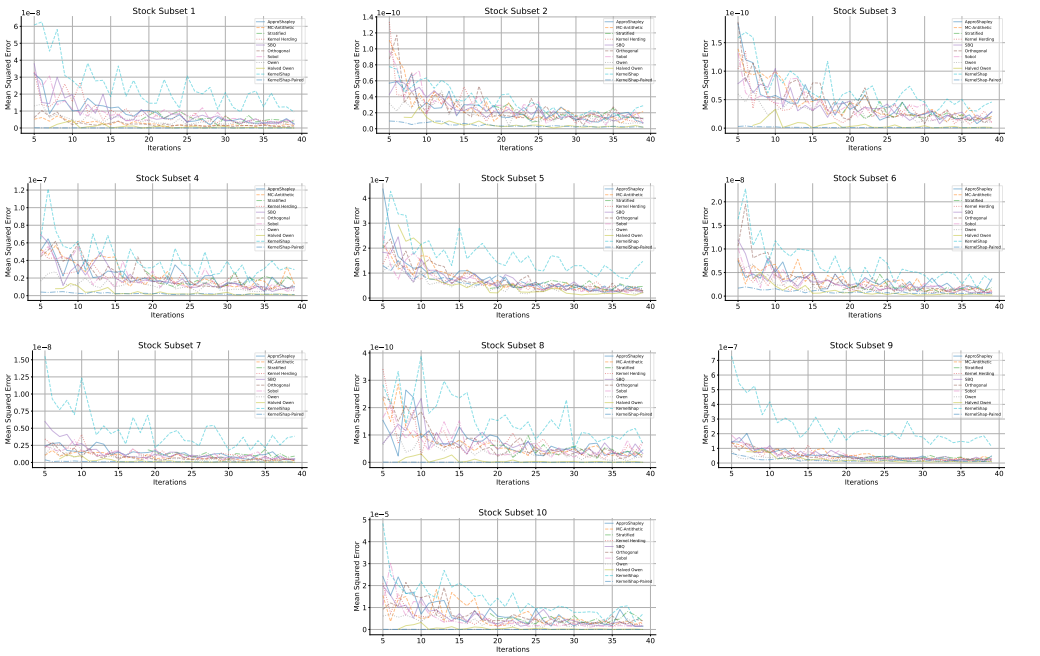


Figure 4: Comparison of the estimators based on mean squared error against exact Shapley values. The lower the error, the better. We take the mean over five runs. Annualized return is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. Some estimators require a minimum number of samples. We do not report their performance for the number of samples below their minimum.

6 Conclusion

We conduct the first systematic study to explain stochastic model predictive control systems. These systems have extensive applications in high-stakes scenarios. However, there is limited work exploring their explainability. We demonstrate that the large number of components and their interactions present significant challenges in explaining these systems. We formulate these systems as a coali-

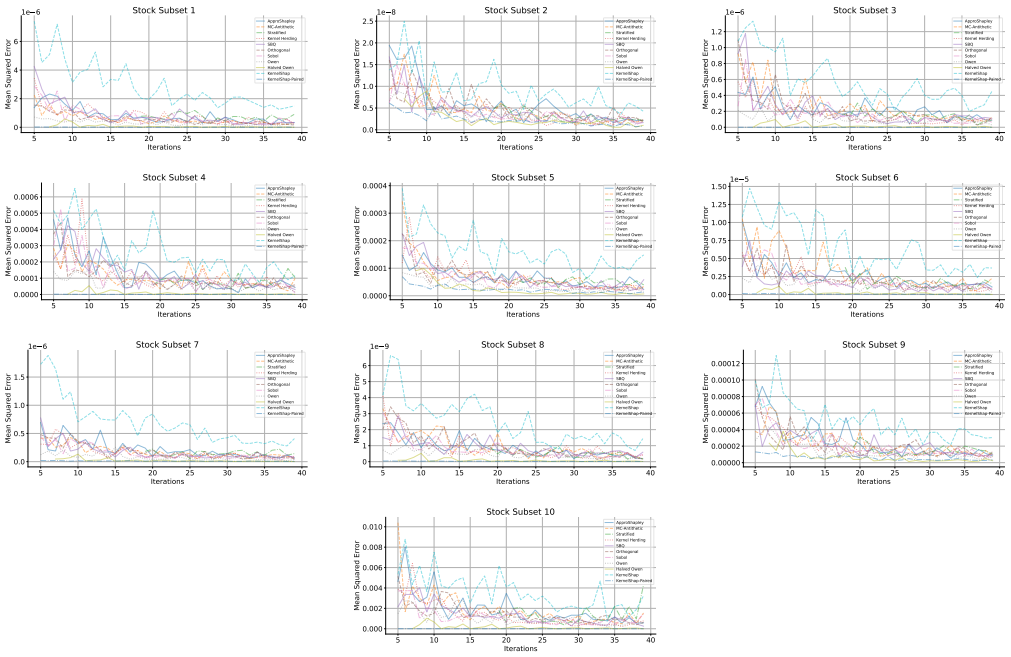
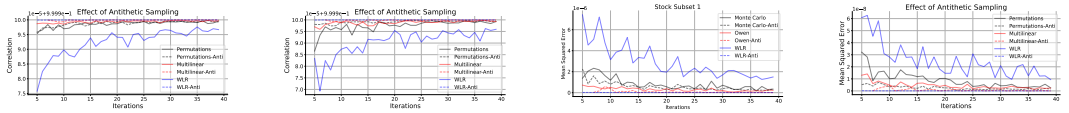


Figure 5: Comparison of the estimators based on mean squared error against exact Shapley values. The lower the error, the better. We take the mean over five runs. Sharpe ratio is the system's performance metric. Calculating the exact Shapley values for 100 players is impractical. We partition the stocks into ten subsets of ten stocks each. In each subset, the stocks are optional. The stocks in other subsets are mandatory. Some estimators require a minimum number of samples. We do not report their performance for the number of samples below their minimum.



(a) Correlation with exact Shapley values. Sharpe ratio is the system's performance metric.

(b) Correlation with exact Shapley values. Annualized return is the system's performance metric.

(c) Mean squared error against Shapley values. Sharpe ratio is the system's performance metric.

(d) Mean squared error against Shapley values. Annualized return is the system's performance metric.

Figure 6: The effect of antithetic sampling on baseline Shapley value estimation methods from different families. We use ApproShapley as the baseline for permutation-based sampling, Owen for multilinear extension, and KernelShap for weighted linear regression (WLR). We average results on the first stock subset over five runs. For correlation, higher values are better. For mean squared error, lower values are better. Antithetic sampling significantly improves estimation accuracy in weighted linear regression and multilinear extension methods.

tion game and use Shapley values to quantify the impact of each component on performance metrics. Computing exact Shapley values is computationally intensive. This is because they account for all possible subsets of components and their interactions. We benchmark Shapley values estimation methods in the context of stochastic model predictive control and provide valuable insights. Our experiments show that halved Owen sampling and KernelShap-Paired offer the best performance.

Limitations and Future Work. Estimating Shapley values remains prohibitive in ultra-high-dimensional problems. Existing estimation methods do not fully address this challenge. Gradient-based explanation methods potentially offer a solution within end-to-end differentiable stochastic model predictive control frameworks. Differentiable optimization (Amos & Kolter, 2017; Agrawal et al., 2019) allows for gradient calculation through optimization problems. Several gradient-based explanation techniques exist in explainable artificial intelligence (Simonyan et al., 2014; Sundararajan et al., 2017; Smilkov et al., 2017). We plan to adapt these methods to stochastic model predictive control.

REFERENCES

Adrian Agogino, Ritchie Lee, and Dimitra Giannakopoulou. Challenges of explaining control. In *2nd ICAPS Workshop on Explainable Planning (XAIP'19)*, 2019.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Ahmad AlAttar, Digby Chappell, and Petar Kormushev. Kinematic-model-free predictive control for robotic manipulator target reaching with obstacle avoidance. *Frontiers in Robotics and AI*, 9, February 2022. doi: 10.3389/frobt.2022.809114. URL https://doi.org/10.3389/frobt.2022.809114.

Frank Allgower, Rolf Findeisen, Zoltan K Nagy, et al. Nonlinear model predictive control: From theory to application. *Journal-Chinese Institute Of Chemical Engineers*, 35(3):299–316, 2004.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pp. 136–145. PMLR, 2017.

Ivo Batkovic, Mario Zanon, and Paolo Falcone. Model predictive control for safe autonomous driving applications. In *AI-enabled Technologies for Autonomous and Connected Vehicles*, pp. 255–282. Springer, 2022.

Leopoldo Bertossi, Benny Kimelfeld, Ester Livshits, and Mikaël Monet. The shapley value in database management. *ACM Sigmod Record*, 52(2):6–17, 2023.

Stephen Boyd, Enzo Busseti, Steve Diamond, Ronald N Kahn, Kwangmoo Koh, Peter Nystrup, Jan Speth, et al. Multi-period trading via convex optimization. *Foundations and Trends® in Optimization*, 3(1):1–76, 2017.

James P Boyle and Richard L Dykstra. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in Order Restricted Statistical Inference: Proceedings of the Symposium on Order Restricted Statistical Inference held in Iowa City, Iowa, September 11–13, 1985*, pp. 28–47. Springer, 1986.

Mark Alexander Burgess and Archie C Chapman. Approximating the shapley value using stratified empirical bernstein sampling. In *IJCAI*, pp. 73–81, 2021.

Enzo Busseti, Steven Diamond, and Stephen Boyd. Cvxportfolio. https://github.com/cvxgrp/cvxportfolio, January 2017. Portfolio Optimization and Back–Testing.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & operations research*, 36(5):1726–1730, 2009.

Javier Castro, Daniel Gómez, Elisenda Molina, and Juan Tejada. Improving polynomial estimation of the shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research*, 82:180–188, 2017.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation via linear regression. *arXiv preprint arXiv:2012.01536*, 2020.

Tony Dang, Frederik Debrouwere, and Erik Hostens. Approximating nonlinear model predictive controllers using support vector machines. In *2021 Australian & New Zealand Control Conference (ANZCC)*, pp. 155–160. IEEE, 2021.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems*, 31, 2018.

Persi Diaconis. Group representations in probability and statistics. *Lecture notes-monograph series*, 11:i–192, 1988.

E Knuth Donald et al. The art of computer programming. *Sorting and searching*, 3(426-458):4, 1999.

Steve Howes, Janarde Le Pore, Ivan Mohler, and Nenad Bolf. Implementing advanced process control for refineries and chemical plants. *Goriva i maziva: časopis za tribologiju, tehniku podmazivanja i primjenu tekućih i plinovitih goriva i inžinjerstvo izgaranja*, 53(2):97–119, 2014.

Yunlong Jiao and Jean-Philippe Vert. The kendall and mallows kernels for permutations. In *International Conference on Machine Learning*, pp. 1935–1944. PMLR, 2015.

Filip Jorissen, Damien Picard, Kristoff Six, and Lieve Helsen. Detailed white-box non-linear model predictive control for scalable building hvac control. In *Modelica Conferences*, pp. 315–323, 2021.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

William R Knight. A computer method for calculating kendall's tau with ungrouped data. *Journal of the American Statistical Association*, 61(314):436–439, 1966.

Gilles Boevi Koumou. Diversification and portfolio theory: a review. *Financial Markets and Portfolio Management*, 34(3):267–312, 2020.

Ouren Kuiper, Martin van den Berg, Joost van der Burgt, and Stefan Leijnen. Exploring explainable ai in the financial sector: Perspectives of banks and supervisory authorities. In *Artificial Intelligence and Machine Learning: 33rd Benelux Conference on Artificial Intelligence, BNAIC/Benelearn 2021, Esch-sur-Alzette, Luxembourg, November 10–12, 2021, Revised Selected Papers 33*, pp. 105–119. Springer, 2022.

Daniela C Landinez-Lamadrid, Diana G Ramirez-Ríos, Dionicio Neira Rodado, Kevin Armando Parra Negrete, and Johana Patricia Combita Niño. Shapley value: its algorithms and application to supply chains. *Inge Cuc*, 13(1):53–60, 2017.

Ritchie Lee, Mykel J Kochenderfer, Ole J Mengshoel, and Joshua Silbermann. Interpretable categorization of heterogeneous time series data. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 216–224. SIAM, 2018.

Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. 2015.

Camacho Eduardo F. Limon Daniel., Alamo Teodoro. Mpc for tracking piecewise constant references for constrained linear systems. *Automatica*, 46(1):189–198, 2010.

Maria Lomeli, Mark Rowland, Arthur Gretton, and Zoubin Ghahramani. Antithetic and monte carlo kernel estimators for partial rankings. *Statistics and Computing*, 29:1127–1147, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Sasan Maleki. *Addressing the computational issues of the Shapley value with applications in the smart grid*. PhD thesis, University of Southampton, 2015.

Horia Mania, Aaditya Ramdas, Martin J Wainwright, Michael I Jordan, and Benjamin Recht. On kernel methods for covariates that are rankings. 2018.

Wilson E Marcilio-Jr and Danilo M Eler. Explaining dimensionality reduction results using shapley values. *Expert Systems with Applications*, 178:115020, 2021.

Harry Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952. doi: j.1540-6261.1952.tb01525.x.

Dang Minh, H Xiang Wang, Y Fen Li, and Tan N Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, pp. 1–66, 2022.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for shapley value estimation. *The Journal of Machine Learning Research*, 23(1):2082–2127, 2022.

Stefano Moretti, Danitsja van Leeuwen, Hans Gmuender, Stefano Bonassi, Joost Van Delft, Jos Kleinjans, Fioravante Patrone, and Domenico Franco Merlo. Combining shapley value and statistics to the analysis of gene expression data in children exposed to air pollution. *BMC bioinformatics*, 9:1–21, 2008.

Thomas Muir. On a simple term of a determinant. In *Proc. Royal Society Edinburg*, volume 21, pp. 441–477, 1898.

Viet Anh Nguyen, Soroosh Shafiee, Damir Filipović, and Daniel Kuhn. Mean-covariance robust risk measurement. *arXiv preprint arXiv:2112.09959*, 2021.

Ramin Okhrati and Aldo Lipani. A multilinear sampling algorithm to estimate shapley values. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 7992–7999. IEEE, 2021.

Guillermo Owen. Multilinear extensions of games. *Management Science*, 18(5-part-2):64–79, 1972.

Sergey M. Plis, Terran Lane, and Vince D. Calhoun. Permutations as angular data: Efficient inference in factorial spaces. In *2010 IEEE International Conference on Data Mining*, pp. 403–410, 2010. doi: 10.1109/ICDM.2010.122.

James A. Primbs. Applications of mpc to finance. In Basil Kouvaritakis and Mark Cannon (eds.), *Handbook of Model Predictive Control*, pp. 667–686. Springer, 2018. doi: 10.1007/978-3-319-77489-3_27.

Markus Quade, Markus Abel, Kamran Shafi, Robert K Niven, and Bernd R Noack. Prediction of dynamical systems by symbolic regression. *Physical Review E*, 94(1):012214, 2016.

Markus Quade, Thomas Isele, and Markus Abel. Machine learning control—explainable and analyzable methods. *Physica D: Nonlinear Phenomena*, 412:132582, 2020.

Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian monte carlo. *Advances in neural information processing systems*, pp. 505–512, 2003.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

Achkan Salehi and Stephane Doncieux. Data-efficient, explainable and safe box manipulation: Illustrating the advantages of physical priors in model-predictive control. In *6th Annual Learning for Dynamics & Control Conference*, pp. 13–24. PMLR, 2024.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: an engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117:1327–1349, August 2021. doi: 10.1007/s00170-021-07682-3. URL https://doi.org/10.1007/s00170-021-07682-3.

Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The annals of statistics*, pp. 2263–2291, 2013.

Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

Grah Simon and Thouvenot Vincent. A projected stochastic gradient algorithm for estimating shapley value applied in attribute importance. In *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4*, pp. 97–115. Springer, 2020.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017. URL http://arxiv.org/abs/1706.03825.

Yunlong Song and Davide Scaramuzza. Policy search for model predictive control with application to agile drone flight. *IEEE Transactions on Robotics*, 38(4):2114–2130, February 2022. doi: 10.1109/TRO.2022.3141602. URL https://doi.org/10.1109/TRO.2022.3141602.

Alexander Stevens and Johannes De Smedt. Explainable predictive process monitoring: Evaluation metrics and guidelines for process outcome prediction. *arXiv preprint arXiv:2203.16073*, 2022.

Arun Subramanian, James B. Rawlings, and Christos T. Maravelias. Model predictive control of hybrid systems: Stability and robustness. *AIChE Journal*, 58(8):2385–2398, 2012. doi: 10.1002/aic.12788.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Nikola Tarashev, Kostas Tsatsaronis, and Claudio Borio. Risk attribution using the shapley value: Methodology and policy applications. *Review of Finance*, 20(3):1189–1213, 2016.

Ekaterina J Vladislavleva, Guido F Smits, and Dick Den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Transactions on Evolutionary Computation*, 13(2):333–349, 2008.

A SHAPLEY VALUES

The Shapley value is a key concept in cooperative game theory. It measures each participant's contribution to the total payoff of a coalition of players. It is based on the principle of fair distribution and gives each player a share of the total payoff that matches their contribution across all possible groups.

A.1 FORMULATIONS

Let N be the set of all players in the coalition game with size $|N|$. We denote the payoff function with v. It inputs a subset of players and outputs a real number. Mathematically, $v : \mathcal{P}(N) \to \mathbb{R}$, where $\mathcal{P}(N)$ is the power set of N. Formulations to compute Shapley values include weighting based on coalition size, mean over permutations, and the multilinear extension. In the next sections, we discuss these formulations.

A.1.1 WEIGHTING BASED ON COALITION SIZE

The marginal contribution of a player i to a coalition S excluding player i is represented by the difference $v(S\cup\{i\})-v(S)$. To compute the Shapley value for player i, denoted as ϕ_i, the following equation is used:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left(v\left(S \cup \{i\}\right) - v\left(S\right) \right). \tag{4}$$

The coefficient within this formulation can be further illustrated as:

$$\frac{|S|!(|N| - |S| - 1)!}{|N|!} = \frac{1}{|N| \binom{|N|-1}{|S|}}, \tag{5}$$

where $\binom{|N|-1}{|S|}$ is the binomial coefficient denoting the number of distinct coalitions of size $|S|$ that can be formed from the set N excluding player i. $|N|$ represents the total count of players involved in the game.

A.1.2 MEAN OVER PERMUTATIONS

An alternative methodology for computing Shapley values is the mean over permutations approach, which emphasizes the sequence in which players join a coalition. This method evaluates all potential permutations of player participation, calculating a player's average marginal contribution across these varying orders.

A permutation π in the context of the player set N is defined as an ordered arrangement of distinct indices $\{\pi_1, \ldots, \pi_{|N|}\}$, where each π_i is a unique identifier corresponding to a player within the set $\{1, \ldots, |N|\}$.

Under this framework, the Shapley value for player i, denoted as ϕ_i, is determined by the following equation:

$$\phi_i = \frac{1}{|N|!} \sum_{\pi \in \Pi} \left[v([\pi]_{i-1} \cup \{i\}) - v([\pi]_{i-1}) \right], \tag{6}$$

Here, Π encompasses all permutations involving the $|N|$ players. Within this equation, $v([\pi]_{i-1})$ represents the coalition value composed of players who are positioned before player i in the permutation π. Conversely, $v([\pi]_{i-1} \cup \{i\})$ indicates the value of the coalition after the inclusion of player i. The difference between these two values calculates the marginal contribution of player i to coalition $[\pi]_{i-1}$.

The factor $\frac{1}{|N|!}$ averages over all $|N|!$ possible player permutations.

A.1.3 MULTILINEAR EXTENSION

The work of Owen (1972) extend the Shapley values framework by incorporating continuous player contributions into cooperative games. Traditionally, contributions in such games were assumed to

be binary (present or not present), but this extension allows each player i (where $i = 1, \ldots, |N|$) to contribute on a continuous scale. These contributions are represented by a continuous variable q_i, constrained within the interval $0 \leq q_i \leq 1$. To accommodate this continuous approach, the payoff function is generalized as follows:

$$\tilde{v}(q_1, \ldots, q_{|N|}) = \sum_{S \subseteq N} \left(\prod_{j \in S} q_j \prod_{j \notin S} (1 - q_j) \right) v(S). \tag{7}$$

This equation encapsulates the notion that the value of a coalition depends not only on the presence or absence of players but also on the extent of their participation, as indicated by q_i. The probabilistic interpretation of q_i as the likelihood of player i participating in the coalition leads to the following formula for calculating Shapley values:

$$\phi_i = \int_0^1 e_i(q)\, dq, \tag{8}$$

where

$$e_i(q) = \mathop{\mathbb{E}}_{S \sim \text{Uniform}(\mathcal{P}(N \setminus \{i\}))} \left[v\left(S \cup \{i\}\right) - v\left(S\right) \right]. \tag{9}$$

Uniform $(\mathcal{P}\left(N \setminus \{i\}\right))$ refers to a uniform distribution over the power set of $N \setminus \{i\}$.

A.1.4 Based on Synergy

Intuitively, the synergy of a coalition, denoted as $w(S)$, can be conceptualized as the additional value generated by the coalition as a whole, above and beyond the aggregate value of its individual subsets. The formal definition is as follows:

$$w(S) = v(S) - \sum_{T \subset S} w(T), \tag{10}$$

where $v(S)$ denotes the payoff of the coalition S, and $w(S)$ represents its synergy. The equation establishes that the synergy of any given coalition S is derived by deducting the sum of the synergies of its subsets from its total value.

Importantly, since the synergy of an empty set is equated to its payoff, i.e., $w(\emptyset) = v(\emptyset) = \phi_0$, Equation 10 can be transformed into a unique non-recursive form by applying the principle of inclusion-exclusion. The transformed equation is as follows:

$$w(S) = \sum_{T \subseteq S} (-1)^{|S|-|T|} v(T), \tag{11}$$

which is proved with mathematical induction. This equation underscores that a coalition's synergy is the alternating sum of the payoffs of its subsets.

Furthermore, the synergy concept provides an alternative formulation for Shapley values as follows:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{w(S \cup \{i\})}{|S| + 1}. \tag{12}$$

Equation 12 posits that the Shapley value of player i, denoted as ϕ_i, is calculated by aggregating the appropriately normalized synergy contributions from all coalitions of which player i is a part. The synergy of each such coalition is divided equally among all its members.

A.2 Properties

The Shapley value possesses several intrinsic properties that underpin its fairness and consistency in allocating credit among players in a coalition. These properties are outlined as follows:

- Efficiency: The collective Shapley values of all features sum up to the total value generated by the coalition. It is mathematically represented as:

$$\sum_{i \in N} \phi_i = v(N) - v(\emptyset), \tag{13}$$

where ϕ_i is the Shapley value of feature i, $v(N)$ denotes the total payoff of the coalition, and $v(\emptyset)$ denotes the payoff of the empty set.

- Symmetry: If two features contribute identically to every coalition, they are assigned the same Shapley value. This is formally expressed as:

$$\phi_i = \phi_j \quad \text{if} \quad v(S \cup \{i\}) - v(S) = v(S \cup \{j\}) - v(S) \quad \text{for all} \quad S \subseteq N \setminus \{i, j\}. \tag{14}$$

- Dummy: A feature with no marginal contribution to any coalition receives a Shapley value of zero:

$$\phi_i = 0 \quad \text{if} \quad v(S \cup \{i\}) - v(S) = 0 \quad \text{for all} \quad S \subseteq N \setminus \{i\}. \tag{15}$$

- Linearity: The Shapley value is linear in the payoff function. If a payoff function is a linear combination of other functions, the Shapley value is the same linear combination of the values for these functions:

$$\phi_i(\alpha v + \beta w) = \alpha \phi_i(v) + \beta \phi_i(w). \tag{16}$$

Here, $\phi_i(\cdot)$ denotes the Shapley value for feature i based on the given value function.

B Shapley Values Estimation Methods

We divide Shapley values estimation methods into three categories based on the Shapley values formulation they approximate: methods based on weighted linear regression, sampling permutations, and multilinear extension.

B.1 Weighted Linear Regression

Algorithms adopting the weighted linear regression formulation for Shapley value calculation include KernelShap, KernelShap-Paired, and SGD-Shapley. The following sections cover these algorithms in detail.

B.1.1 KernelShap

Lundberg & Lee (2017) pioneer the use of weighted least squares for calculating Shapley values, a method they term KernelShap. They consider a linear model represented as follows:

$$g(x) = \phi_0 + \sum_{i=1}^{|N|} \phi_i x_i, \tag{17}$$

where $x \in \{0,1\}^{|N|}$, and each x_i denotes the absence (0) or presence (1) of player i in a coalition for $i = 1, \ldots, |N|$. Utilizing weighted linear regression as specified in Equation 18, the parameters $\phi_1^*, \ldots, \phi_{|N|}^*$ are optimized to approximate the Shapley values for players $1, \ldots, |N|$. This optimization is performed on a training set of input-output pairs $(x, v'(x))$, where v' represents the payoff function with an indicator vector as its input.

$$\phi_0^*, \ldots, \phi_{|N|}^* \in \underset{\phi_0, \ldots, \phi_{|N|}}{\arg\min}\ L(v', g, w) \tag{18}$$

The weighting function $w(\cdot)$ and loss function $L(\cdot)$ in the model are defined by:

$$w(x) = \frac{|N| - 1}{\binom{|N|}{\mathbf{1}^\top x} \left(\mathbf{1}^\top x\right) \left(|N| - \mathbf{1}^\top x\right)}, \tag{19}$$

$$L(v', g, w) = \frac{1}{|X|} \sum_{x \in X} \left[w(x) \left[v'(x) - g(x) \right]^2 \right], \tag{20}$$

where $X \subseteq \{0,1\}^{|N|}$ represents the set of coalition inputs for estimating Shapley values. Equation 18 has a unique solution in practice, particularly when $|X|$ significantly exceeds $|N|$. The weight function $w(x)$ depends solely on $\mathbf{1}^\top x$, the count of non-zero elements in x. It is established that $\phi_1^*, \ldots, \phi_{|N|}^*$ converge to the exact Shapley values when $X = \{0,1\}^{|N|}$, i.e., when all possible coalitions are utilized in the computation (Lundberg & Lee, 2017). ϕ_0^* is not associated with any specific player but remains a parameter of the model. The cases where $w(\mathbf{1})$ and $w(\mathbf{0})$ are considered infinite (due to the denominator becoming zero) impose constraints that ensure $\phi_0^* = g(\mathbf{0}) = v'(\mathbf{0})$ and $\sum_{i=0}^{|N|} \phi_i^* = g(\mathbf{1}) = f(\mathbf{1})$. In practical applications, these infinite weights are often managed by assigning a large numerical value, such as 10^6, to them.

B.1.1.1 Closed-Form Solution The weighted linear regression problem, as specified in Equation 18, admits a closed-form solution via the weighted least squares method. One must first recast the problem in matrix form to derive this solution. Let us define $\tilde{X}$ to be a $2^{|N|} \times (|N|+1)$ binary matrix, where each row corresponds to an element of the set X, augmented by a leading column of ones to accommodate the intercept term ϕ_0, that is, $\tilde{X}[:,1] = \mathbf{1}$. Moreover, let $\phi = [\phi_0, \ldots, \phi_{|N|}]^\top$ represent the vector of parameters, and let $\tilde{v}'$ be a vector containing the payoffs for each coalition, ordered identically to $\tilde{X}$. Hence, the optimization problem 18 translates to the following matrix equation:

$$\phi^* \in \arg\min_{\phi} \left[\left(\tilde{v}' - \tilde{X}\phi \right)^\top W \left(\tilde{v}' - \tilde{X}\phi \right) \right], \tag{21}$$

where W is a diagonal matrix containing the coalition weights corresponding to the order of $\tilde{X}$. The solution to this optimization problem, the weighted least squares estimate of ϕ^*, is given by:

$$\phi^* = \left(\tilde{X}^\top W \tilde{X} \right)^{-1} \tilde{X}^\top W \tilde{v}'. \tag{22}$$

This closed-form solution requires the evaluation of the payoff function v' for each possible coalition, an operation with exponential complexity in the number of players. Nonetheless, in practice, using a representative subset of all coalitions to construct $\tilde{X}$, W, and $\tilde{v}'$ is feasible.

B.1.2 KernelSHAP-Paired

Covert & Lee (2020) propose to improve KernelShap by utilizing antithetic sampling from Monte Carlo simulation (Rubinstein & Kroese, 2016). The key to this method is using negatively correlated input pairs, specifically x and $\mathbf{1} - x$, instead of independent and identically distributed (i.i.d.) points. This approach reduces the variance of the estimator. In Shapley value estimation, where each sample represents a coalition of players, using pairs like $(x, \mathbf{1} - x)$ ensures that each player's inclusion and exclusion are considered within the same sample set. This results in a more balanced representation of coalitions, hence, a reduction in variance. The modification to the weighted linear regression problem is expressed as follows:

$$\phi_0^*, \ldots, \phi_{|N|}^* \in \arg\min_{\phi_0, \ldots, \phi_{|N|}} \left\{ \frac{1}{2|X|} \sum_{x \in X} \left[w(x) \left(v'(x) - g(x) \right)^2 + w(\mathbf{1} - x) \left(v'(\mathbf{1} - x) - g(\mathbf{1} - x) \right)^2 \right] \right\}. \tag{23}$$

B.1.3 SGD-Shapley

Simon & Vincent (2020) suggest estimation of Shapley values via projected stochastic gradient descent, addressing the infinite weights for the empty set and grand coalition, $w(\mathbf{0})$ and $w(\mathbf{1})$ as described in Equation 19. They circumvent the challenge posed by these weights by introducing the constraint $\sum_{i=1}^{|N|} \phi_i = v'(\mathbf{1}) - v'(\mathbf{0})$ and eliminating ϕ_0 from the set of optimization variables. Consequently, the weighted linear regression problem originally presented in Equation 18 is reconstituted as:

$$\phi_1^*, \ldots, \phi_{|N|}^* \in \underset{\phi_1,\ldots,\phi_{|N|}}{\arg\min} \frac{1}{|X|} \sum_{x \in X} w(x) \left[v'(x) - \left(v'(\mathbf{0}) + \sum_{i=1}^{|N|} \phi_i x_i \right) \right]^2, \quad \text{subject to} \sum_{i=1}^{|N|} \phi_i = v'(\mathbf{1}) - v'(\mathbf{0}), \tag{24}$$

Here, w represents the weight function for coalitions, v' is the payoff function, X is the set of binary vectors indicating the presence or absence of players in coalitions, and ϕ_i^* stands for the estimated Shapley value for player i, for all players indexed from 1 to $|N|$.

To further refine their algorithm and enhance its theoretical grounding, the authors incorporate an additional constraint $\|\phi\| \leq D$, where $\phi = [\phi_1, \ldots, \phi_{|N|}]^\top$ and D is a positive constant, which, in practice, may be chosen sufficiently large. This gives rise to the final optimization formulation:

$$\phi_1^*, \ldots, \phi_{|N|}^* \in \underset{\phi_1,\ldots,\phi_{|N|}}{\arg\min} \frac{1}{|X|} \sum_{x \in X} w(x) \left[v'(x) - \left(v'(\mathbf{0}) + \sum_{i=1}^{|N|} \phi_i x_i \right) \right]^2, \quad \text{subject to } \phi \in K_1 \cap K_2, \tag{25}$$

where K_1 and K_2 denote the convex feasible sets derived from the constraints $\sum_{i=1}^{|N|} \phi_i = v'(\mathbf{1}) - v'(\mathbf{0})$ and $\|\phi\| \leq D$, respectively. To solve the optimization problem described in Equation 25, Simon & Vincent (2020) utilize projected stochastic gradient descent. During each iteration, after updating the optimization variable $\phi^{(t)}$ according to the gradient derived from a single sample x, the variable is projected onto the intersection of the sets K_1 and K_2. The direct projection onto $K_1 \cap K_2$ is nontrivial as it necessitates solving an additional optimization problem without a simple closed-form solution. Alternatively, separate projections onto K_1 and K_2 yield closed-form solutions:

$$\text{Proj}_{K_1}(\phi^{(t)}) = \phi^{(t)} - \frac{1}{|N|} \left(\sum_{i=1}^{|N|} \phi_i^{(t)} - (v'(\mathbf{1}) - v'(\mathbf{0})) \right), \tag{26}$$

$$\text{Proj}_{K_2}(\phi^{(t)}) = \min \left(1, \frac{D}{\|\phi^{(t)}\|} \right) \phi^{(t)}, \tag{27}$$

where $\text{Proj}_{K_1}(\phi^{(t)})$ and $\text{Proj}_{K_2}(\phi^{(t)})$ denote the respective projections of $\phi^{(t)}$ onto K_1 and K_2. Subsequently, Dykstra's projection algorithm (Boyle & Dykstra, 1986), as outlined in Algorithm 1, is employed to project $\phi^{(t)}$ onto $K_1 \cap K_2$.

Algorithm 1 Dykstra's Algorithm

Input: $\phi_t \in \mathbb{R}^{|N|}$, $\text{Proj}_{K_1}(\cdot)$, and $\text{Proj}_{K_2}(\cdot)$.
Result: Projection of ϕ_t onto $K_1 \cap K_2$.
$p = 0,\ q = 0$.
while not converged **do**
    $\phi_t^{(\text{prev})} = \phi_t$.
    $\phi_t = \text{Proj}_{K_1}(\phi_t^{(\text{prev})} + p)$.
    $p = (\phi_t^{(\text{prev})} + p) - \phi_t$.
    $\phi_t^{(\text{prev})} = \phi_t$.
    $\phi_t = \text{Proj}_{K_2}(\phi_t^{(\text{prev})} + q)$.
    $q = (\phi_t^{(\text{prev})} + q) - \phi_t$.
end while
Return ϕ_t.

B.2 SAMPLING PERMUTATIONS

The second category of methods for approximating Shapley values involves sampling permutations. This approach is grounded in a reformulation of Equation 6 as an expectation:

$$\phi_i = \mathop{\mathbb{E}}_{\pi \sim \text{Uniform}(\Pi)} \left[v([\pi]_{i-1} \cup \{i\}) - v([\pi]_{i-1}) \right], \tag{28}$$

where Π represents the set of all permutations of size $|N|$, with $|N|$ denoting the number of players in the game. In this context, v is the payoff function taking players as input, and $[\pi]_{i-1}$ comprises the players preceding player i in a given permutation π. Subsequent sections discuss various methodologies to approximate the expectation described in Equation 28. These methodologies encompass ApproShapley and its enhanced versions, such as antithetic sampling, stratified sampling, kernel-based methods, and sampling in real space.

B.2.1 APPROSHAPLEY

ApproShapley (Castro et al., 2009) uses Monte Carlo simulation to estimate Shapley values. This approach involves randomly sampling permutations from the set of all possible permutations and then calculating the average of a player's incremental contributions across these permutations. The Monte Carlo estimate of the Shapley value for player i can be formalized as follows:

$$\phi_i = \frac{1}{|\tilde{\Pi}|} \sum_{\pi \in \tilde{\Pi}} \left[v([\pi]_{i-1} \cup \{i\}) - v([\pi]_{i-1}) \right], \tag{29}$$

Here, $\tilde{\Pi} \subseteq \Pi$ represents the subset of sampled permutations. The set Π consists of all permutations of the player set of size $|N|$. In this method, $\tilde{\Pi}$ is selected randomly from Π, and the expectation is estimated by averaging over the marginal contributions from these sampled permutations.

B.2.1.1 Improved Monte Carlo Simon & Vincent (2020) propose the improvement provided in Algorithm 2 which achieves a twofold increase in computational efficiency.

Algorithm 2 Improved Monte Carlo Algorithm for Estimating Shapley Values

Input: Value function v, a set of permutations $\tilde{\Pi}$.
Result: Estimation of $\phi_i = [\phi_1, \ldots, \phi_{|N|}]^\top$.
$\phi = \mathbf{0}$, $m^{(0)} = v(\emptyset)$, $m^{(\text{prev})} = 0$, $m^{(\text{new})} = m^{(0)}$.
for $\pi \in \tilde{\Pi}$ **do**
    for i in π **do**
        $m^{(\text{prev})} = m^{(\text{new})}$.
        $m^{(\text{new})} = v\left([\pi]_{i-1} \cup \{i\}\right)$.
        $\phi_i = \phi_i + \left(m^{(\text{new})} - m^{(\text{prev})}\right)$.
    end for
    $m^{(\text{prev})} = 0$, $m^{(\text{new})} = m^{(0)}$.
end for
$\phi = \frac{\phi}{|\tilde{\Pi}|}$.
Return ϕ.

In this algorithm, $[\pi]_{i-1}$ represents the set of players preceding player i in permutation π. The term $m^{(0)}$, initialized as $v(\emptyset)$, is introduced to avoid redundant calculations of $v(\emptyset)$ in each outer iteration. The calculation of $m^{(\text{new})} - m^{(\text{prev})}$ captures the marginal contribution of player i to the coalition $[\pi]_{i-1}$.

B.2.2 ANTITHETIC SAMPLING

Rubinstein & Kroese (2016) propose to use antithetic sampling in Monte Carlo simulations to reduce variance. Unlike standard Monte Carlo methods that rely on independent and identically distributed (i.i.d.) sampling, antithetic sampling chooses negatively correlated samples. This negative correlation between sample pairs balances out the extremes in the sampling distribution, leading to a more

uniform coverage of the space and, consequently, reducing the overall variance of the estimate. Variance is a measure of dispersion in a set of data points. In standard Monte Carlo simulations, random, independent sampling can lead to clusters of data points in some regions of the space while leaving other areas sparsely sampled. This non-uniform coverage can inflate the variance of the estimated value. Antithetic sampling, by contrast, mitigates this issue through its negatively correlated sampling strategy. When a data point falls in one region of space, its antithetic counterpart will likely fall in a different area, ensuring a more even space exploration.

Lomeli et al. (2019) propose to use Monte Carlo simulation with antithetic sampling to estimate Shapley values. By considering pairs of coalitions as $(\pi, \text{reverse}(\pi))$, antithetic sampling ensures that the permutations are not just randomly sampled but are chosen in a manner that each sample is counterbalanced by its reverse. Antithetic sampling is expressed in Equation 30.

$$\phi_i = \frac{1}{2|\tilde{\Pi}|} \sum_{\pi \in \tilde{\Pi}} \left[(v([\pi]_{i-1} \cup \{i\}) - v([\pi]_{i-1})) + (v([\text{reverse}(\pi)]_{i-1} \cup \{i\}) - v([\text{reverse}(\pi)]_{i-1})) \right]. \tag{30}$$

B.2.3 Stratified Sampling

Stratified sampling is another variance reduction technique in Monte Carlo simulation. This method involves partitioning the function's domain into distinct subgroups, known as strata, and then executing Monte Carlo simulations within each subgroup independently. The purpose of this stratification is to minimize the variance within each group and maximize it between different groups, thereby ensuring a more representative sampling across the entire domain. The expectation estimation using stratified sampling in a Monte Carlo simulation is mathematically expressed as follows:

$$\mathbb{E}[f] \approx \sum_{j=1}^{n} w_j \left(\frac{1}{n_j} \sum_{i=1}^{n_j} f(X_{ji}) \right), \tag{31}$$

where n represents the total number of strata, w_j denotes the weight of the j-th stratum (usually proportional to the stratum's size relative to the entire domain), n_j is the number of samples within the j-th stratum, and $X_{j,i}$ are the samples from this stratum. The function f symbolizes the simulated process, with $\mathbb{E}[f]$ as the expected value.

B.2.3.1 Connection With Antithetic Sampling Stratified sampling reduces variance by dividing the sample space into distinct, homogeneous strata and ensuring that each is adequately sampled. Antithetic sampling, on the other hand, creates pairs of negatively correlated samples. Antithetic sampling can be conceptualized as a particular case of stratified sampling where the domain is bifurcated into two complementary strata. At each step, we perform stratified sampling with two strata; each sample comes from a stratum. Using negatively correlated variables within these pairs is akin to ensuring diversity among strata in traditional stratified sampling.

B.2.3.2 Methods To use stratified sampling to estimate Shapley values, a stratum is defined by the size of the coalition to which the marginal contribution is measured. This is equivalent to the location of player i in a permutation π. Hence, $n = |N|$, where n denotes the total number of strata, and $|N|$ is the number of players in the game. A critical component of stratified sampling is allocating sample sizes to each stratum. Stratified sampling methods to estimate Shapley values differ by dividing the sampling budget among different strata. We survey the existing methods in the next sections.

B.2.3.2.1 Stratified Sampling With Neyman Allocation Neyman allocation is an approach used in stratified sampling to minimize the variance of the estimated mean. The main idea behind Neyman allocation is to allocate more samples to strata with greater variability and larger size, as these strata influence the overall estimate variance more. Consider a population divided into n strata. Let n_j be the size of the j-th stratum, σ_j^2 be the variance within the j-th stratum, and B be the total sample size. The Neyman allocation for the j-th stratum, n_j, is as follows:

$$n_j = \lfloor B \cdot \frac{\sigma_j^2}{\sum_{i=1}^{n} \sigma_i^2} \rfloor \tag{32}$$

This allocation is derived to minimize the variance of the stratified mean under the constraint of a fixed total sample size. Castro et al. (2017) propose using stratified sampling with Neyman allocation to sample permutations to estimate Shapley values. The process is described in Algorithm 3.

Algorithm 3 Stratified Sampling With Neyman Allocation for Estimating Shapley Values

Input: Sample budget B.
Result: Shapley value estimation for the i-th player, i.e. ϕ_i.
for $k = 1$ to n **do**
  $m_k = \lfloor B/2n \rfloor$.
  $\phi_i^{(k)} = 0$.
  $s_i^{(k)} = 0$.
  for $j = 1$ to m_k **do**
    $t = v([\pi^{(i,k)}]_{i-1} \cup \{i\}) - v([\pi^{(i,k)}]_{i-1})$. # $\pi^{(i,k)} \sim$ Uniform($\Pi^{(i,k)}$).
    $\phi_i^{(k)} = \phi_i^{(k)} + t$.
    $s_i^{(k)} = s_i^{(k)} + t^2$.
  end for
  $\mathbb{V}^{(i,k)} = \frac{1}{m_k - 1} \left(s_i^{(k)} - \frac{\left(\phi_i^{(k)}\right)^2}{m_k} \right)$.
end for
for $k = 1$ to n **do**
  $m'_k = \lfloor B \frac{\mathbb{V}_i^{(k)}}{\sum_{k=1}^{n} \mathbb{V}_i^{(k)}} \rfloor - m_k$.
end for
for $k = 1$ to n **do**
  for $j = 1$ to m'_k **do**
    $\phi_i^{(k)} = \phi_i^{(k)} + v([\pi^{(i,k)}]_{i-1} \cup \{i\}) - v([\pi^{(i,k)}]_{i-1})$. # $\pi^{(i,k)} \sim$ Uniform($\Pi^{(i,k)}$).
  end for
  $\phi_i^{(k)} = \frac{\phi_i^{(k)}}{m_k + m'_k}$.
end for
$\phi_i = \frac{\sum_{k=1}^{n} \phi_i^{(k)}}{n}$.
Return ϕ_i.

B.2.3.2.2 STRATIFIED EMPIRICAL BERNSTEIN SAMPLING Burgess & Chapman (2021) propose a stratified empirical Bernstein sampling technique to minimize the sampling error for both with-replacement and without-replacement scenarios. For random variable $a \leq X \leq b$, its width, denoted by D, is defined as $b - a$. Given sequentially drawn random samples $X_{i,1}, \ldots, X_{i,m_i}$, first they define some statistics for each stratum. The average for the first m_i samples of stratum i is given by

$$\bar{X}_{i,m_i} = \frac{1}{m_i} \sum_{j=1}^{m_i} X_{i,j}, \tag{33}$$

and the biased and the unbiased sample variance for stratum i are computed as follows:

$$\hat{\sigma}_i^2 = \frac{1}{m_i} \sum_{j=1}^{m_i} \left(X_{i,j} - \bar{X}_{i,m_i} \right)^2, \; \hat{\hat{\sigma}}_i^2 = \frac{1}{m_i - 1} \sum_{j=1}^{m_i} \left(X_{i,j} - \bar{X}_{i,m_i} \right)^2, \tag{34}$$

where $\hat{\sigma}_i^2$ denotes the biased sample variance, and $\hat{\hat{\sigma}}_i^2$ is the unbiased sample variance.

Given probability p and weights τ for strata, the bounds are as follows:

$$\mathbb{P}\left(\sqrt{\sum_{i=1}^{n}\tau_i(\bar{X}_{i,m_i}-\mu_i)^2} \geq \sqrt{\alpha+\left(\sqrt{\beta}+\sqrt{\gamma}\right)^2}\right) \leq p, \tag{35}$$

where

$$\alpha = \sum_{i=1}^{n}\frac{4}{17}\Omega_{m_i}^{n_i}D_i^2\tau_i^2,\ \beta = \log\left(\frac{3}{p}\right)\left(\max_i \tau_i^2\Psi_{m_i}^{n_i}D_i^2\right), \tag{36}$$

$$\gamma = 2\sum_{i=1}^{n}\tau_i^2\Psi_{m_i}^{n_i}\left(m_i-1\right)\frac{\hat{\sigma}_i^2}{m_i} + \log\left(\frac{6n}{p}\right)\sum_{i=1}^{n}\tau_i^2D_i^2\Psi_{m_i}^{n_i}\Psi_{m_i}^{n_i} + \log\left(\frac{3}{p}\right)\left(\max_i \tau_i^2\Psi_{m_i}^{n_i}D_i^2\right), \tag{37}$$

where for sampling with replacement, Ω_m^n and Ψ_m^n are as follows:

$$\Omega_m^{n^{(w/)}} = \Psi^{n^{(w/)}} = \frac{1}{m}. \tag{38}$$

On the other hand, for sampling without replacement, Ω_m^n and Ψ_m^n are as follows:

$$\Omega_m^{n^{(w/o)}} = \sum_{k=m}^{n-1}\frac{1}{k^2},\ \Psi^{n^{(w/o)}} = \sum_{k=m}^{n-1}\frac{n}{k^2(k+1)}. \tag{39}$$

The authors suggest calculating two bounds, one applicable when samples are drawn with replacement and another for when samples are drawn without replacement. Before each sampling event, they estimate both bounds and select the tighter of the two to guide the sampling process. The final procedure is provided at Algorithm 4. Note that in this context, a random variable corresponds to the marginal contribution of a player to a coalition given a permutation, i.e., $v([\pi]_{i-1}\cup\{i\})-v([\pi]_{i-1})$, where v is the payoff function taking a coalition of players as input, and $[\pi]_{i-1}$ comprises the players preceding player i in a given permutation π. An upper bound for the width of marginal contribution should be available as D. a_j, b_j, c_j, and d_j for $j = 0, 1$ in lines 12-19 are used to calculate α, β, and γ in the Equation 35.

B.2.4 Kernel Monte Carlo

In Monte Carlo simulations, utilizing a kernel to guide the sampling process represents a significant refinement over simple random sampling. This kernel-based approach, called kernel Monte Carlo, leverages the information from previously sampled points to make informed decisions about subsequent sampling locations.

The process begins with selecting initial sample points randomly or based on some heuristic. As the simulation progresses, instead of selecting future points independently, a kernel function is employed to determine the next points based on the properties of previously sampled points. This kernel function establishes a probability distribution over the sample space, where the likelihood of choosing a particular point depends on its relationship to the existing samples. The kernel function's characteristics determine the nature of this relationship. A commonly used kernel might assign higher probabilities to points near previously sampled points, encouraging local exploration of the sample space. Alternatively, the kernel might prioritize under-explored regions, guiding the simulation to sample more diversely and cover a broader domain area.

If we denote x_i as the i-th sampled point and $K(x_i, \cdot)$ as the kernel function centered at x_i, the probability of sampling a new point x_{m+1} can be expressed as

$$P(x_{m+1}) \propto \sum_{i=1}^{m} K(x_i, x_{m+1}), \tag{40}$$

where m is the number of points already sampled. The kernel function K aggregates the influences of all previous samples to determine the sampling probability of new points. The kernel approach to Monte Carlo simulations is particularly advantageous when dealing with complex or multi-modal

Algorithm 4 Stratified Empirical Bernstein Sampling for Estimating Shapley Values

Input: Probability p, strata number n, initial sample numbers m_i, initial stratum sample variances $\hat{\sigma}_i^2$, weights τ_i, widths D_i, sample budget B.
Result: Shapley value estimation for the i-th player, i.e. ϕ_i.
while $\sum_i^n m_i < B$ **do**
  $k^* = -1$.
  $l^* = \infty$.
  $j^* = -1$.
  for $k = 1$ to n **do**
    $m_k = m_k + 1$.
    $a = [0, 0]$, $b = [0, 0]$, $c = [0, 0]$, $d = [0, 0]$.
    for $i = 0$ to n **do**
      $\Omega_{\min} = \min(\Omega_{m_i}^{n_i{}^{(w/)}}, \Omega_{m_i}^{n_i{}^{(w/o)}})$.
      $\Psi_{\min} = \min(\Psi_{m_i}^{n_i{}^{(w/)}}, \Psi_{m_i}^{n_i{}^{(w/o)}})$.
      $a_0 = a_0 + \log(6n/p) D_i^2 \Psi_{m_i}^{n_i{}^{(w/)}} \Omega_{\min} \tau^2$.
      $a_1 = a_1 + \log(6n/p) D_i^2 \Psi_{m_i}^{n_i{}^{(w/o)}} \Omega_{\min} \tau^2$.
      $b_0 = \max(b_0, \log(3/p) D_i^2 \Psi_{m_i}^{n_i{}^{(w/)}} \Psi_{\min} \tau^2)$.
      $b_1 = \max(b_1, \log(3/p) D_i^2 \Psi_{m_i}^{n_i{}^{(w/o)}} \Psi_{\min} \tau^2)$.
      $c_0 = c_0 + 2\Psi_{m_i}^{n_i{}^{(w/)}} ((m_i - 1)\hat{\hat{\sigma}}_i^2 / m_i) \tau^2$.
      $c_1 = c_1 + 2\Psi_{m_i}^{n_i{}^{(w/o)}} ((m_i - 1)\hat{\hat{\sigma}}_i^2 / m_i) \tau^2$.
      $d_0 = d_0 + \frac{4}{17} D_i^2 \Omega_{m_i}^{n_i{}^{(w/)}} \tau^2$.
      $d_1 = d_1 + \frac{4}{17} D_i^2 \Omega_{m_i}^{n_i{}^{(w/o)}} \tau^2$.
    end for
    $j = \arg\min_{\hat{j}} (d_{\hat{j}} + (\sqrt{c_{\hat{j}} + a_{\hat{j}} + b_{\hat{j}}} + \sqrt{b_{\hat{j}}})^2)$.
    $w^* = (d_j + (\sqrt{c_j + a_j + b_j} + \sqrt{b_j})^2)$.
    if $w^* < l^*$ **then**
      $k^* = k$.
      $l^* = w^*$.
      $j^* = j$.
    end if
    $m_k = m_k - 1$.
  end for
  if $j^* == 0$ **then**
    Sample for stratum k^* with replacement.
  else if $j^* == 1$ **then**
    Sample for stratum k^* without replacement.
  end if
  $m_{k^*} = m_{k^*} + 1$.
  Update $\bar{X}_{k^*, m_{k^*}}$, $\hat{\sigma}_{k^*}^2$, and $\hat{\hat{\sigma}}_{k^*}^2$.
end while
$\phi_i = \frac{\sum_{i=1}^n \tau_i \bar{X}_{i,m_i}}{\sum_{i=1}^n \tau_i}$.
Return ϕ_i.

distributions in discrete spaces. The adaptive nature of the sampling process, guided by the kernel function, allows for more efficient space exploration. It balances the need to thoroughly investigate areas around known high-value points (exploitation) with discovering potentially valuable regions that have not been sampled yet (exploration).

In kernel Monte Carlo simulations, the kernel function K is a fundamental component that determines how each new sampling point is chosen based on the existing samples. This kernel function is not just a simple measure of distance or proximity between points in the sample space; rather, it often involves embedding the points into a higher-dimensional space where their similarities can be assessed more effectively. The concept of embedding in the context of kernel functions refers to mapping the original data points into a higher-dimensional feature space. In this transformed space, relationships between points that might not be apparent in the original space become more discernible. The kernel function K then evaluates the similarity between these embedded points.

Suppose we have a function Φ that maps a data point x to a new space, i.e., $\Phi(x)$. The kernel function $K(x_i, x_j)$ then measures the similarity between points x_i and x_j based on their images in this new space, often calculated as an inner product as follows:

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle. \tag{41}$$

This inner product in the feature space allows the kernel to capture complex relationships and patterns in the data that are not evident in the original space. For instance, in a discrete setting, the embedding might allow the kernel to recognize and emphasize certain patterns or sequences in the data points, which are crucial for the estimation task. In the next sections, we will first review existing kernel functions on permutations and then discuss more sophisticated kernel methods.

B.2.4.1 Kernel Functions Mitchell et al. (2022) propose adopting kernel Monte Carlo to estimate Shapley values. Choices of kernels applicable to permutation space include Kendall, Mallows, and Spearman kernels, which we discuss in the next sections.

B.2.4.1.1 KENDALL KERNEL To comprehend the foundation of the Kendall kernel, we must first discuss the nature of the Kendall rank correlation coefficient, denoted by τ. Initially introduced by Kendall (1938) and further developed in the context of kernel methods by Jiao & Vert (2015), τ is a non-parametric statistic that measures the ordinal association between two variables.

Consider a dataset comprising pairs of observations $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, with x and y representing two distinct variables. Kendall τ is computed by evaluating the concordance and discordance between all pairs of observations as follows:

$$\tau = \frac{\sum_{i<j} \operatorname{sgn}(x_i - x_j) \cdot \operatorname{sgn}(y_i - y_j)}{\binom{n}{2}},$$

where n denotes the total count of observations, and $\operatorname{sgn}(\cdot)$ signifies the sign function, which is defined for any pair (x_i, x_j) as

$$\operatorname{sgn}(x_i - x_j) = \begin{cases} 1 & \text{if } x_i > x_j, \\ 0 & \text{if } x_i = x_j, \\ -1 & \text{if } x_i < x_j. \end{cases}$$

The value of τ offers insights into the nature of the relationship between the variables: a positive τ suggests a similar ranking by both variables, a negative τ implies an inverse relationship and a τ near zero indicates little to no association.

Turning our focus to permutations, the Kendall kernel operates on two permutations, π and π', of length $|N|$. It commences by ascertaining the number of concordant and discordant pairs, mathematically expressed as

$$n_{\text{con}}(\pi, \pi') = \sum_{i<j} \left[\mathbb{1}_{\pi(i)<\pi(j)} \mathbb{1}_{\pi'(i)<\pi'(j)} + \mathbb{1}_{\pi(i)>\pi(j)} \mathbb{1}_{\pi'(i)>\pi'(j)} \right], \tag{42}$$

$$n_{\text{dis}}(\pi, \pi') = \sum_{i<j} \left[\mathbb{1}_{\pi(i)<\pi(j)} \mathbb{1}_{\pi'(i)>\pi'(j)} + \mathbb{1}_{\pi(i)>\pi(j)} \mathbb{1}_{\pi'(i)<\pi'(j)} \right]. \tag{43}$$

A pair (i, j), with $i, j \in \{1, \ldots, |N|\}$, is labeled concordant in the permutations π and π' if the order of i and j is consistent across both permutations. If not, the pair is deemed discordant. The sum of concordant and discordant pairs equals the total number of pairings $\binom{|N|}{2}$, which is the count of all possible unordered pairs of the set $\{1, \ldots, |N|\}$. The Kendall kernel is then defined as the normalized difference between the numbers of concordant and discordant pairs:

$$K_\tau(\pi, \pi') = \frac{n_{\text{con}}(\pi, \pi') - n_{\text{dis}}(\pi, \pi')}{\binom{|N|}{2}}. \tag{44}$$

This kernel measures the degree of similarity between permutations. A direct implementation of the Kendall kernel requires $O(|N|^2)$ operations, which can be computationally intensive for coalition games. However, Knight (1966) propose an efficient $O(|N| \log(|N|))$ algorithm, which leverages the concept of the merge sort.

The embedding function associated with the Kendall kernel K_τ is as follows:

$$\Phi_\tau(\pi) = \left(\frac{1}{\sqrt{\binom{|N|}{2}}} \left(\mathbb{1}_{\pi(i)>\pi(j)} - \mathbb{1}_{\pi(i)<\pi(j)} \right) \right)_{1 \le i < j \le |N|}, \tag{45}$$

which captures the pairwise ordinal relations within the permutation π. The Kendall kernel between two permutations π and π' can be expressed as the inner product of their respective embeddings:

$$K_\tau(\pi, \pi') = \Phi_\tau(\pi)^\top \Phi_\tau(\pi'). \tag{46}$$

B.2.4.1.2 MALLOWS KERNEL Like the Kendall kernel, the Mallows kernel, as introduced in the work by Jiao & Vert (2015), is constructed on the foundation of discordant pairs in two permutations. The Mallows kernel for a pair of permutations π and π' is defined as

$$K_M^\lambda(\pi, \pi') = e^{-\lambda n_{\text{dis}}(\pi, \pi') / \binom{|N|}{2}}, \tag{47}$$

where λ is a non-negative parameter regulating the impact of discordance on the kernel's value, and $n_{\text{dis}}(\pi, \pi')$ denotes the number of discordant pairs between the permutations π and π'. Mitchell et al. (2022) propose to add $\binom{|N|}{2}$to scale the kernel's output relative to the size of the permutation set.

The Mallows kernel does not have an explicit feature space representation as readily as the Kendall kernel. As noted by Mania et al. (2018), an explicit feature map for the Mallows kernel would involve a complex and high-dimensional representation. The computation of the Mallows kernel is usually performed directly using the formula stated above without explicitly referencing a feature space.

B.2.4.1.3 SPEARMAN KERNEL Mitchell et al. (2022) propose the Spearman Kernel based on the unnormalized Spearman rank distance. The Spearman rank distance is a measure utilized to quantify the discrepancy between two sets of rankings. It is useful when the rankings do not necessarily follow a normal distribution, allowing for non-parametric statistical analysis. The unnormalized form of the Spearman rank distance between two permutations, π and π', each of length $|N|$, is defined as the sum of the squared differences between the ranks of each element in the two permutations as follows:

$$D_{\text{Spearman}}(\pi, \pi') = \sum_{i=1}^{|N|} (\pi(i) - \pi'(i))^2. \tag{48}$$

The term $(\pi(i) - \pi'(i))^2$ represents the squared rank difference for the i-th element between the permutations π and π'. The Spearman rank distance aggregates these squared differences across all elements to provide a cumulative measure of the rank divergence between the two permutations. The distance is larger when there is a greater disparity in the rankings and reaches a minimum value of zero when the rankings are identical. The Spearman rank distance $D_{\text{Spearman}}(\pi, \pi')$ is a semi-metric of negative type (Diaconis, 1988). Mitchell et al. (2022) leverage this property to transform the Spearman rank distance into a kernel, denoted as K_{Spearman}, using the connection between semi-metrics of negative type and kernels as detailed by Sejdinovic et al. (2013). Assuming K_{Spearman} is unknown, the Spearman rank distance can be expressed in the context of a kernel as follows:

$$D_{\text{Spearman}}(\pi, \pi') = K_{\text{Spearman}}(\pi, \pi) + K_{\text{Spearman}}(\pi', \pi') - 2K_{\text{Spearman}}(\pi, \pi'), \tag{49}$$

where the Spearman rank distance is formulated by:

$$D_{\text{Spearman}}(\pi, \pi') = \sum_{i=1}^{|N|} (\pi(i) - \pi'(i))^2 = \pi^\top \pi + \pi'^\top \pi' - 2\pi^\top \pi'. \tag{50}$$

Here, the kernel $K_{\text{Spearman}}(\pi, \pi')$ naturally emerges as the inner product $\pi^\top \pi'$, satisfying the relationship in Equation 49. The feature map Φ_{Spearman} associated with the Spearman kernel corresponds to the permutation itself, hence $\Phi_{\text{Spearman}}(\pi) = \pi$.

B.2.4.2 Expected Values for Kernels In this section, we detail the methodology proposed by Mitchell et al. (2022) for computing the expected value of each kernel for a fixed point π and the uniform distribution over permutations. We denote the uniform distribution over permutations with $\mathbb{P}_\Pi = \text{Uniform}(\Pi)$, where Π indicates the set of all permutations of size $|N|$.

B.2.4.2.1 KENDALL KERNEL The computation of the expected value for the Kendall kernel is direct. The following equation represents it:

$$\forall \pi \in \Pi, \quad \mathbb{E}_{\pi' \sim \Pi}[K_\tau(\pi, \pi')] = 0.$$

B.2.4.2.2 SPEARMAN KERNEL The expected value for the Spearman kernel is straightforward to compute. The formula is given as

$$\forall \pi \in \Pi, \quad \mathbb{E}_{\pi' \sim \Pi}[K_{\text{Spearman}}(\pi, \pi')] = \frac{|N|(|N|+1)^2}{4}.$$

B.2.4.2.3 MALLOWS KERNEL Computing the expected value for the Mallows kernel is more challenging. We begin by discussing inversions in a permutation, probability-generating, and moment-generating functions.

B.2.4.2.3.1 Preliminaries

- Inversion: Given a permutation π, an inversion is a pair of elements (π_i, π_j) satisfying $\pi_i > \pi_j$ and $i < j$. Take, for example, the permutation $[1, 3, 2, 4]$, which contains a single inversion: the pair $(3, 2)$. In this case, three is larger and appears before 2. The total number of inversions in π, denoted as n_{inv}, equates to the count of discordant pairs compared to the identity permutation $[1 \dots |N|]$, expressed as $n_{\text{inv}} = n_{\text{dis}}(\pi, [1 \dots |N|])$.
- The probability-generating and the moment-generating functions: The probability-generating function for a discrete random variable X is given by

$$G_X(s) = \sum_{k=0}^{\infty} P(X = k) \cdot s^k,$$

where $G_X(s)$ encapsulates the probabilities of X's outcomes into a function. The moment-generating function for discrete distributions is defined as

$$M_X(t) = E[e^{tX}] = \sum_k e^{tk} P(X = k).$$

The moment-generating function encodes all the moments of X's distribution. The n-th derivative of $M_X(t)$ evaluated at $t = 0$ yields the n-th moment of X.

B.2.4.2.3.2 Calculating the Expectation

Back to calculating the expected value for Mallows kernel, Muir (1898) compute the probability-generating function for the number of inversions n_{inv} in a permutation. This function is given by

$$G_{n_{\text{inv}}}(s) = \prod_{j=1}^{|N|} \frac{1 - s^j}{j(1 - s)}, \tag{51}$$

where $G_{n_{\text{inv}}}(s)$ is the probability-generating function of n_{inv}, and $|N|$ denotes the length of the permutation.

Based on this, the moment generating function, $M_{n_{\text{inv}}}(t)$, is derived as follows:

$$M_{n_{\text{inv}}}(t) = G_{n_{\text{inv}}}(e^t) = \prod_{j=1}^{|N|} \frac{1 - e^{tj}}{j(1 - e^t)} = \mathbb{E}[e^{tn_{\text{inv}}}], \tag{52}$$

indicating that $M_{n_{\text{inv}}}(t)$, the moment generating function, expresses the expected exponential growth of tn_{inv}. As it was mentioned earlier, the total number of inversions in π, denoted as n_{inv}, equates to the count of discordant pairs compared to the identity permutation $[1 \dots |N|]$. Hence we have

$$M_{n_{\text{inv}}}\left(-\frac{\lambda}{\binom{|N|}{2}}\right) = \mathbb{E}\left[e^{-\frac{\lambda n_{\text{inv}}}{\binom{|N|}{2}}}\right] = \mathbb{E}_{\pi' \sim \text{Uniform}(\Pi)}\left[K_M^{\lambda}([1 \dots |N|], \pi')\right]. \tag{53}$$

Mitchell et al. (2022) utilize the right-invariance property of the number of discordant pairs, n_{dis}, expressed as

$$n_{\text{dis}}(\pi, \pi') = n_{\text{dis}}\left(\tau(\pi), \tau(\pi')\right) \quad \text{for all} \quad \tau \in \Pi, \tag{54}$$

to reach the following conclusion:

$$\forall \tau \in \Pi, \quad \mathbb{E}_{\pi' \sim \text{Uniform}(\Pi)}[K_M^{\lambda}([1 \dots |N|], \pi')] = \mathbb{E}_{\pi' \sim \text{Uniform}(\Pi)}[K_M^{\lambda}(\tau[1 \dots |N|], \tau\pi')] \tag{55}$$

$$= \mathbb{E}_{\pi' \sim \text{Uniform}(\Pi)}[K_M^{\lambda}(\tau[1 \dots |N|], \pi')], \tag{56}$$

which ultimately leads to the expectation for Mallows kernel as follows:

$$\forall \pi \in \Pi, \quad \mathbb{E}_{\pi' \sim \text{Uniform}(\Pi)}[K_M^{\lambda}([1 \dots |N|], \pi')] = \mathbb{E}_{\pi' \sim \text{Uniform}(\Pi)}[K_M^{\lambda}(\pi, \pi')] \tag{57}$$

$$= \prod_{j=1}^{|N|} \frac{1 - e^{-\lambda j / \binom{|N|}{2}}}{j(1 - e^{-\lambda / \binom{|N|}{2}})}. \tag{58}$$

In the next sections, we discuss how the aforementioned kernels are used to estimate Shapley values. Methodologies include kernel herding and sequential Bayesian quadrature.

B.2.4.3 Kernel Herding The kernel herding algorithm selects samples by maximizing the following criterion for the next sample $\pi^{(n+1)}$:

$$\pi^{(n+1)} = \underset{\pi}{\operatorname{argmax}} \left[\mathbb{E}_{\pi' \sim \text{Uniform}(\Pi)}[K(\pi, \pi')] - \frac{1}{n+1} \sum_{i=1}^{n} K(\pi, \pi^{(i)}) \right],$$

where $\mathbb{E}_{x' \sim \text{Uniform}(\Pi)}[K(\pi, \pi')]$ is the expected kernel evaluation over the distribution Uniform(Π), and $\frac{1}{n+1} \sum_{i=1}^{n} K(\pi, \pi^{(i)})$ is the average kernel evaluation over the samples so far. The selected sample $\pi^{(n+1)}$ is the one that, when added to the current set of samples, yields the largest discrepancy from the true mean embedding in the reproducing kernel Hilbert space, hence improving the representation of Uniform(Π). Kernel herding improves the rate of error reduction to $O(1/n)$, surpassing the $O(1/\sqrt{n})$ rate achieved by standard random sampling methods. The term $\mathbb{E}_{x' \sim \text{Uniform}(\Pi)}[K(\pi, \pi')]$ is calculated for Kendall, Spearman, and Mallows kernels in the previous sections. Mitchell et al. (2022) propose to approximate the argmax in the above equation by sampling a limited number of permutations and taking the one maximizing the objective function. In this way, kernel herding would have the computational complexity of $O(n^2)$ for producing n permutations. The marginal contribution of the sampled permutations estimates Shapley values, i.e.:

$$\phi_i \approx \frac{1}{n} \sum_{j \in [1,\ldots,n]} \left[v([\pi^{(j)}]_{i-1} \cup \{i\}) - v([\pi^{(j)}]_{i-1}) \right], \tag{59}$$

where v is the payoff function taking players as input, and $[\pi^{(j)}]_{i-1}$ comprises the players preceding player i in a given permutation $\pi^{(j)}$, ϕ_i is the Shapley value for the i-th player, and $\pi^{(1)} \ldots \pi^{(n)}$ are generated with kernel herding.

B.2.4.4 Sequential Bayesian Quadrature In this section, we first describe sequential Bayesian quadrature (Rasmussen & Ghahramani, 2003). Then, explain how it can estimate Shapley values. Consider the task of computing the integral $Z = \int f(x)p(x)dx$, where f represents a function, and p signifies the input density. This procedure begins by assigning a Gaussian process (GP) as a prior over f. Utilizing a Gaussian process in this context is called Bayesian quadrature, enabling us to derive a posterior distribution over Z.

The mean of the posterior over Z is expressed as a linear combination of the function values at the sampled points $x^{(1)}, \ldots, x^{(n)}$, illustrated in the following equation:

$$\mathbb{E}_{\text{GP}}[Z | f(x^{(1)}), \ldots, f(x^{(n)})] = \sum_{i=1}^{n} z_i^T \Sigma^{-1} f(x^{(i)}), \tag{60}$$

In this equation, $z_i = \int K(x, x^{(i)})p(x)dx$ denotes the expected kernel function value at the point $x^{(i)}$. Minimizing the posterior variance of Z is a key step in sequential Bayesian quadrature to guide sampling. This variance is given as follows:

$$\mathbb{V}_{\text{GP}}[Z | f(x^{(1)}), \ldots, f(x^{(n)})] = \int \int K(x, x')p(x)p(x')\, dx\, dx' - z^T \Sigma^{-1} z, \tag{61}$$

where Σ^{-1} is the inverse of the kernel covariance matrix. The selection of sample points is performed in a manner that sequentially minimizes this variance. Once the samples $x^{(1)}, \ldots, x^{(n)}$ are selected, the integral Z is estimated as follows:

$$Z \approx \sum_{i=1}^{n} w_i f(x^{(i)}), \tag{62}$$

where the weights w_i are derived by solving the linear system $\Sigma w = z$.

Now we describe how sequential Bayesian quadrature could be utilized to estimate Shapley values (Mitchell et al., 2022). Consider the integral $z_i = \int K(x, x^{(i)})p(x)dx$, previously computed for

kernels such as Kendall, Spearman, and Mallows. This integral is a constant dependent only on the problem's dimensionality. Focusing on the sequential Bayesian quadrature variance (Equation 61), it is observed that the nested integrals' value does not depend on z and can thus be ignored when minimizing for z. The challenge then is minimizing the term $-z^T\Sigma^{-1}z$. Mitchell et al. (2022) propose approximating this by sampling a limited number of points and selecting the one minimizing the expression. An important computational consideration is the inversion of the kernel covariance matrix Σ. Utilizing the Cholesky decomposition reduces the computational cost. The Cholesky decomposition is formulated as $\Sigma = LL^T$, where L is a lower triangular matrix. In contrast to the typical $O(n^3)$ complexity of direct matrix inversion, Cholesky factorization reduces this to solving two systems of linear equations. By first solving $LY = I$ for Y, with I being the identity matrix, and then $L^TX = Y$ for X, we efficiently obtain X as Σ^{-1}. Furthermore, when a new row and column are appended to Σ, updating its Cholesky factorization is not an extensive operation. Therefore, sequentially generating n samples to approximate Shapley values via sequential Bayesian quadrature results in an overall complexity of $O(n^3)$. We refer to Mitchell et al. (2022) for error analysis in reproducing kernel Hilbert spaces.

B.2.5 Sampling in Real Space

Various advanced Monte Carlo sampling techniques, typically employed in real space, are adapted for Shapley value estimation by defining a mapping from real space to permutation space. This adaptation encompasses methods such as Sobol sequences on the sphere and orthogonal spherical codes.

B.2.5.1 Mapping From Real Space $\mathbb{R}^{|N|-1}$ to Permutation Space We start with providing background and then explain the process of mapping a point from real space $\mathbb{R}^{|N|-1}$ to the hypersphere $\mathbb{S}^{|N|-2}$ and then from the hypersphere $\mathbb{S}^{|N|-2}$ to the permutation space.

B.2.5.1.1 Preliminaries This section provides background on permutohedrons, the Cayley graph, and their connection to hypersphere $\mathbb{S}^{|N|-2}$.

B.2.5.1.1.1 Permutohedrons

A permutohedron is a geometric representation that serves to understand the complexities of permutations, particularly when considering a specific permutation π. This polytope is associated with the permutations of a set and is uniquely characterized for a set of $|N|$ elements as an $(|N|-1)$-dimensional figure. It encapsulates all possible permutations of these elements in its structure. Each vertex of the permutohedron represents a distinct permutation; hence, for any permutation π of $|N|$ elements, there is a corresponding vertex on this $(|N|-1)$-dimensional polytope. The vertices and edges of the symbolize adjacency between permutations, where two permutations are considered adjacent if they can be obtained from one another by a single swap of adjacent elements. Therefore, the neighboring vertices of π on the permutohedron represent permutations that are reachable from π through such a swap. Though the permutohedrons exists within an $|N|$-dimensional space, it itself is $(|N|-1)$-dimensional. The vertex placement for π within this geometry provides an understanding of how π is positioned in relation to other permutations, illustrating the number of swaps needed to transition between them. The permutohedron corresponding to the permutation space of size $|N|$ lies on the following hyperplane:

$$\sum_{i=1}^{|N|} \pi^{-1}(i) = \frac{|N|(|N|+1)}{2}, \tag{63}$$

which is simply the summation of numbers from 1 to $|N|$. Also, the normal vector is as follows:

$$\vec{n} = \left[\frac{1}{\sqrt{|N|}} \quad \frac{1}{\sqrt{|N|}} \quad \cdots \quad \frac{1}{\sqrt{|N|}}\right]. \tag{64}$$

B.2.5.1.1.2 Caley Graph

A Cayley graph is a graphical representation used in group theory to visualize the structure of a group and its operation. Given a group G and a set of generators S for G, the Cayley graph is constructed by representing each element of G as a vertex and connecting two vertices with an edge if one can be obtained from the other by applying a generator from the set S. The connection between permutohedrons and Cayley graphs becomes evident when considering permutation groups. A permutohedron for a set of size $|N|$ can be viewed as a geometric representation of a permutation group, where each vertex represents a permutation and edges indicate a single transposition between permutations. Similarly, using transpositions as generators, a Cayley graph of a permutation group exhibits a structure where vertices represent permutations and edges correspond to applying a transposition. This graph has interesting properties. For instance, the antipode of a vertex, which denotes the vertex farthest from it, is the vertex corresponding to the inverse of its permutation. Regarding the kernels previously discussed, the Kendall distance of two permutations is equal to the graph distance of their corresponding vertices. Moreover, the Spearman distance between two permutations equals the Euclidean distance between their corresponding vertices.

B.2.5.1.1.3 Hypersphere $\mathbb{S}^{|N|-2}$

Note that each vertex of the permutohedron, which corresponds to each vertex of the Caley graph, lies on the following hypersphere $\mathbb{S}^{|N|-2}$:

$$\sum_{i=1}^{|N|} \pi^{-1}(i)^2 = \frac{|N|(|N|+1)(2|N|+1)}{6}, \tag{65}$$

which is the summation of squared numbers from 1 to $|N|$. Given the connection between the hypersphere $\mathbb{S}^{|N|-2}$ and the Caley graph and the intuitive properties of the Caley graph, which was discussed before, the hypersphere $\mathbb{S}^{|N|-2}$ is a continuous relaxation of the permutation space. Plis et al. (2010) discover this connection, and Mitchell et al. (2022) utilize it to estimate Shapley value. In the next section, we describe how a point sampled from real space $\mathbb{R}^{|N|-1}$ is mapped to the hypersphere $\mathbb{S}^{|N|-2}$, and then the permutation space.

B.2.5.1.2 MAPPING FROM REAL SPACE $\mathbb{R}^{|N|-1}$ TO HYPERSPHERE $\mathbb{S}^{|N|-2}$ Mapping samples from real space $\mathbb{R}^{|N|-1}$ to the hypersphere $\mathbb{S}^{|N|-2}$ is straightforward. Donald et al. (1999) propose normalizing each sample to have unit length and demonstrate that independent Gaussian random variables in real space $\mathbb{R}^{|N|-1}$ are uniformly distributed on the hypersphere $\mathbb{S}^{|N|-2}$ using this mapping.

B.2.5.1.3 MAPPING FROM HYPERSPHERE $\mathbb{S}^{|N|-2}$ TO PERMUTATION SPACE To map a vector x from the hypersphere $\mathbb{S}^{|N|-2}$, first it needs to be projected into the hyperplane of the permutohedron, i.e. $\sum_{i=1}^{|N|} \pi^{-1}(i) = \frac{|N|(|N|+1)}{2}$. For this purpose, matrix $U \in \mathbb{R}^{|N|-1 \times |N|}$ is defined as follows:

$$U = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 1 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & -(|N|-1) \end{bmatrix}, \tag{66}$$

It is easy to show $\hat{U}\vec{n} = 0$, as the summation of each row of U is zero, and $\vec{n}$ is a scalar multiple of the unit vector. In this way, $x \in \mathbb{S}^{|N|-2}$ is mapped into the hyperplane of the permutohedron using the following equation:

$$\tilde{x} = \hat{U}^\top x, \tag{67}$$

where $\hat{U}$ denotes U with row vectors normalized. It is proved that $\tilde{x}$ lies on the hyperplane of the permutohedron by simply showing

$$\tilde{x}^\top \vec{n} = x^\top \hat{U} \vec{n} = 0. \tag{68}$$

Once the point on the hyperplane of the permutohedron, i.e., $\tilde{x}$, is available, the closest vertex of the Caley graph, which denotes the closest permutation to it, is found using the following equation:

$$\pi = \text{argsort}(\tilde{x}^\top). \tag{69}$$

B.2.5.2 Sampling Permutations From Real Space $\mathbb{R}^{|N|-1}$ In this section, we provide the final algorithm to generate permutations by sampling from the real space $\mathbb{R}^{|N|-1}$ so that the resulting distribution of the generated permutations is uniform.

Algorithm 5 Sampling Permutation From Real Space $\mathbb{R}^{|N|-1}$

1: **Result:** A permutation of length $|N|$.
2: $x = \text{Normal}(0, 1)$.
3: $x = \frac{x}{\|x\|}$.
4: $\hat{x} = U^\top x$.
5: $\pi = \text{argsort}(\hat{x})$.
6: **Return** π.

Note that the starting points on the real space $\mathbb{R}^{|N|-1}$ have a standard normal distribution, but following the steps described in the algorithm, the resulting points on the hypersphere $\mathbb{S}^{|N|-2}$ and the permutation space have a uniform distribution.

B.2.5.3 Orthogonal Spherical Codes Unlike the independent random samples used in standard Monte Carlo methods, orthogonal Monte Carlo generates samples that are orthogonal to each other. This orthogonality ensures a more uniform distribution of samples across the space, minimizing overlap and redundancy. Mitchell et al. (2022) propose to use orthogonal Monte Carlo to sample permutations to estimate Shapley values. In the first step, $|N|-1$ orthonormal vectors are generated in the real space $\mathbb{R}^{|N|-1}$. This is done with the Gram-Schmidt process. The Gram-Schmidt process starts with $|N|-1$ linearly independent vectors as input. It leaves the first vector as it is. Then, it involves iteratively adjusting each subsequent vector in the original set. Given the original vector y_i, the process modifies it to be orthogonal to all previously processed vectors $x_1, x_2, \ldots, x_{i-1}$. This is accomplished by projecting y_i onto each of the previously-obtained orthogonal vectors x_j and subtracting this projection from y_i, which mathematically is

$$x_i := y_i - \sum_{j=1}^{i-1} \text{proj}_{x_j}(y_i),$$

where $\text{proj}_{x_j}(y_i)$ is the projection of y_i onto x_j, and is calculated as

$$\text{proj}_{x_j}(y_i) = \frac{\langle y_i, x_j \rangle}{\langle x_j, x_j \rangle} x_j.$$

The pseudo-code of the Gram-Schmidt process is provided in Algorithm 6.

Once a set of $|N|-1$ orthonormal vectors in the real space $\mathbb{R}^{|N|-1}$ is generated, they are converted to permutations by computing $\text{argsort}(\hat{U^\top} x)$ for $x \in X$, where X is the output of Gram-Schmidt process. This results $|N|-1$ permutations. Antithetic sampling is also proposed to accelerate convergence. The final procedure to generate $2(|N|-1)$ permutations by sampling orthogonal vectors is outlined in Algorithm 7. v is the payoff function taking players as input, $[\pi]_{i-1}$ comprises the players preceding player i in a given permutation π, and line 9 is motivated by antithetic sampling. Mitchell et al. (2022) prove Algorithm 7 is an unbiased estimator of Shapley values.

Algorithm 6 Gram-Schmidt Process

Input: A set of $|N|-1$ linearly independent vectors $Y = \{y_1, y_2, ..., y_{|N|-1}\}$.
Result: Set of orthonormal vectors $X = \{x_1, x_2, ..., x_{|N|-1}\}$.
$X = \emptyset$.
for $i = 1$ to $|N|-1$ **do**
    $x_i = y_i$.
    for $j = 1$ to $i-1$ **do**
        $x_i = x_i - \frac{\langle x_j, y_i \rangle}{\langle x_j, x_j \rangle} x_j$.
    end for
    $x_i = \frac{x_i}{\|x\|}$.
    Add x_i to the set X.
end for
Return X.

Algorithm 7 Orthogonal Spherical Codes for Shapley Value Estimation

Input: A set of $|N|-1$ linearly independent vectors $Y = \{y_1, y_2, ..., y_{|N|-1}\}$, value function v.
Result: Estimation of Shapley value ϕ_i for the i-th player.
$\phi_i = 0$.
$X = \text{GramSchmidtProcess}(Y)$.
for x in X **do**
    $\hat{x} = U^\top x$.
    $\pi = \text{argsort}(\hat{x})$.
    $\phi_i = \phi_i + v([\pi]_{i-1} \cup \{i\}) - v([\pi]_{i-1})$.
    $\pi = \text{argsort}(-\hat{x})$.
    $\phi_i = \phi_i + v([\pi]_{i-1} \cup \{i\}) - v([\pi]_{i-1})$.
end for
$\phi_i = \phi_i / (2(|N|-1))$.
return ϕ_i.

Note that this mapping is similar to the concept of spherical codes, which aims to find the optimal arrangement of points on the sphere so that the minimum distance between any two points is maximized.

B.2.5.4 Sampling Using Sobol Sequence We start by providing some background to facilitate explaining this method, then explain the sampling process.

B.2.5.4.1 PRELIMINARIES In this section, we review the Sobol sequence and generalized polar coordinate system, which are essential to understanding how Shapley values could be estimated by sampling using the Sobol sequence.

B.2.5.4.1.1 Sobol Sequence

A Sobol sequence is a quasi-random, low-discrepancy sequence used in numerical analysis to estimate the properties of high-dimensional integrals. Constructed using methods of digital sequences, it is defined over the field of two elements and generates points in a unit hypercube $[0,1)^{|N|-1}$. The sequence employs a set of direction numbers for each dimension derived from primitive polynomials over a finite field.

The properties of Sobol sequences are notable for their uniform distribution, as the sequence fills the space more uniformly than uncorrelated random points. This means that the proportion of points within any small region of the unit hypercube should be approximately equal to the region's volume. Moreover, the sequences exhibit low discrepancy, meaning they have a lower deviation from the uniform distribution compared to random sequences. They also maintain good projection properties onto lower-dimensional subspaces, retaining their low-discrepancy nature even when some dimensions are ignored.

B.2.5.4.1.2 Generalized Polar Coordinate System

The Generalized Polar Coordinate System is an extension of the traditional two-dimensional polar coordinate system, adapted for higher-dimensional spaces, specifically for $|N|-1$ dimensions. This system represents a point using one radial coordinate and $|N|-2$ angular coordinates. The radial coordinate, denoted as r, indicates the point's distance from a central reference point or the origin and is a non-negative number. It is defined similarly across all dimensions, given by $r = \sqrt{x_1^2 + x_2^2 + \cdots + x_{|N|-1}^2}$, where $x_1, x_2, \ldots, x_{|N|-1}$ are the Cartesian coordinates of the point.

The angular coordinates, denoted as $\theta_1, \theta_2, \ldots, \theta_{|N|-2}$, are generalizations of angles to higher dimensions. These angular coordinates are akin to angles in a plane but are extended to encompass $|N|-1$ dimensions. The first $|N|-3$ angles, $\theta_1, \ldots, \theta_{|N|-3}$, typically range from 0 to π. In contrast, the final angle, $\theta_{|N|-2}$, varies from 0 to 2π, similar to the azimuthal angle in traditional polar coordinates.

The angular coordinates on the sphere are independent and have probability density functions:

$$f(\theta_{|N|-3}) = \frac{1}{2\pi},$$

and

$$f(\theta_j) = \frac{1}{\mathcal{B}\left(\frac{|N|-j-2}{2}, \frac{1}{2}\right)} \sin^{(|N|-j-3)}(\theta_j),$$

for $1 \leq j < |N|-3$, where $\mathcal{B}$ is the beta function. The $\mathcal{B}$ function scales the $\sin^{(|N|-j-3)}(\theta_j)$ term such that when you integrate $f(\theta_j)$ over the interval $[0, \pi]$, the result is one to reflect the total probability theorem. The cumulative distribution function for the angular coordinates is as follows:

$$F_j(\theta_j) = \int_0^{\theta_j} f_j(u)\, du. \tag{70}$$

B.2.5.4.2 SAMPLING PROCESS Mitchell et al. (2022) propose an approach to estimate Shapley values using a Sobol sequence. Initially, they generate a set of samples Y within the unit range $[0,1]^{|N|-2}$ using the Sobol sequence. These samples serve as the cumulative distribution function value for the angular coordinates on the hypersphere $\mathbb{S}^{|N|-2}$. Using root-finding techniques, the angles are then obtained by solving a specific equation, indicated as Equation 70. They set the radius to one to ensure all samples are on the hypersphere's surface. In the next step, the location of the samples in the Cartesian coordinates system is calculated. These samples on the hypersphere are subsequently projected onto the hyperplane of the permutohedron. The permutohedron hyperplane is defined by the equation $\sum_{i=1}^{|N|} \pi^{-1}(i) = \frac{|N|(|N|+1)}{2}$. They apply the projection using the matrix outlined in Equation 66. Following the projection, the points on the permutohedron hyperplane are mapped to the nearest vertex of the permutohedron, corresponding to a vertex of the Cayley graph. Finally, with the obtained set of n sampled permutations, they estimate Shapley values for the i-th player using the following equation:

$$\phi_i \approx \frac{1}{n} \sum_{j \in [1,\ldots,n]} \left[v([\pi^{(j)}]_{i-1} \cup \{i\}) - v([\pi^{(j)}]_{i-1}) \right], \tag{71}$$

where v denotes the payoff function that takes players as input, and $[\pi^{(j)}]_{i-1}$ includes the players preceding player i in a given permutation $\pi^{(j)}$, with ϕ_i representing the Shapley value for the i-th player. The procedure is detailed in Algorithm 8.

Algorithm 8 Sobol Sequences on the Sphere for Shapley Value Estimation

Input: Number of samples n, value function v.
Result: Estimation of Shapley value ϕ_i for the i-th player.
$\phi_i = 0$.
$Y =$ SobolSequence $(n, |N| - 2)$.
for y in Y **do**
  $\theta = \mathbf{0}$.
  for $j = 1$ to $|N| - 2$ **do**
    $\theta_j = F_j^{-1}(y_j)$.
  end for
  $x =$ PolarToCartesian$(r = 1, \theta = \theta)$.
  $\hat{x} = U^\top x$.
  $\pi =$ argsort$(\hat{x})$.
  $\phi_i = \phi_i + v([\pi]_{i-1} \cup \{i\}) - v([\pi]_{i-1})$.
end for
$\phi_i = \phi_i / n$.
return ϕ_i.

B.3 MULTILINEAR EXTENSION

This family of methods estimates Shapley values by approximating the multilinear extension formulation of Shapley values, which is as follows:

$$\phi_i = \int_0^1 e_i(q)\, dq, \tag{72}$$

where $e_i(q)$ measures the expected increase in value when player i joins a random subset of other players. This is calculated as

$$e_i(q) = \mathop{\mathbb{E}}_{S \sim \text{Uniform}(\mathcal{P}(N \setminus \{i\}))} \left[v\left(S \cup \{i\}\right) - v\left(S\right) \right], \tag{73}$$

and Uniform $(\mathcal{P}\left(N \setminus \{i\}\right))$ refers to a uniform distribution over the power set of $N \setminus \{i\}$, meaning all possible groups of players excluding player i are equally likely to be chosen.

Okhrati & Lipani (2021) propose to estimate Shapley values by approximating the integral in Equation 72. To estimate the integral $\phi_i = \int_0^1 e_i(q)\, dq$ using a Riemann sum, the interval from 0 to 1

is divided into Q equal subintervals of width $\Delta q = \frac{1}{Q}$. Sample points $q_1, q_2, \ldots, q_Q$ are chosen in each subinterval. The function $e_i(q)$ is estimated at these points using a Monte Carlo simulation with M samples. The values $e_i(q_1), e_i(q_2), \ldots, e_i(q_Q)$ are obtained through this simulation. Each rectangle's area, calculated with height $e_i(q_k)$ estimated by Monte Carlo and width Δq, is computed. The Riemann sum $\sum_{k=1}^{Q} e_i(q_k) \cdot \Delta q$ is the sum of these areas and approximates the integral ϕ_i. The final procedure proposed by Okhrati & Lipani (2021) is provided in Algorithm 9. In this algorithm, v' represents the payoff function, which inputs an indicator vector. The variable Q determines the precision of the Riemann sum. The term M specifies the accuracy of the Monte Carlo estimation for approximating the expectation value in Equation 73. Line 7 of the algorithm ensures that player i is not included in the coalition represented by the binary vector x. Subsequently, in line 8, the marginal contribution of player i in the coalition denoted by x is computed, where e_i refers to the i-th unit vector:

Algorithm 9 Owen Sampling for Shapley Value Estimation

1: **Input:** Value function v', Q, M.
2: **Result:** Estimation of Shapley value ϕ_i for the i-th player.
3: $\phi_i = 0$.
4: **for** q in $[0, 1/Q, 2/Q, \ldots, 1]$ **do**
5: **for** $m = 1$ to M **do**
6: $x = \text{Bern(q)}$.
7: $x[i] = 0$.
8: $\phi_i = \phi_i + v'(x + e_i) - v'(x)$.
9: **end for**
10: **end for**
11: $\phi_i = \phi_i/(M(Q+1))$.
12: **return** ϕ_i.

Okhrati & Lipani (2021) propose to utilize antithetic sampling to reduce the variance of the estimator provided in Algorithm 9. To achieve this, each sample needs to be paired with the sample with the least correlated. Regarding coalitions represented by indicator vectors, this corresponds to $\mathbf{1} - x$, where x denotes the sampled coalition. The updated procedure is provided at Algorithm 10. Note that as the number of samples in the inner loop is doubled, q in the outer loop goes only to 0.5 instead of 1. In this way, the total number of samples remains the same at $(M(Q+1)$. Owen (1972) propose the multilinear extension to games. Hence, Okhrati & Lipani (2021) names algorithms 9 and 10 Owen sampling and Halved Owen sampling.

Algorithm 10 Halved Owen Sampling for Shapley Value Estimation

1: **Input:** Value function v', Q, M.
2: **Result:** Estimation of Shapley value ϕ_i for the i-th player.
3: $\phi_i = 0$.
4: **for** q in $[0, 1/Q, 2/Q, \ldots, 0.5]$ **do**
5: **for** $m = 1$ to M **do**
6: $x = \text{Bern(q)}$.
7: $x[i] = 0$.
8: $\phi_i = \phi_i + v'(x + e_i) - v'(x)$.
9: $x = \mathbf{1} - x$.
10: $x[i] = 0$.
11: $\phi_i = \phi_i + v'(x + e_i) - v'(x)$.
12: **end for**
13: **end for**
14: $\phi_i = \phi_i/(M(Q+1))$.
15: **return** ϕ_i.

B.3.1 Connection to Stratified Sampling

The multilinear extension for estimating Shapley values is related to stratified sampling. In stratified sampling, the population is divided into different groups or strata, and samples are taken from each

stratum. Each value of q in the integral from 0 to 1 in the Shapley value estimation process can be considered as defining a stratum. This is because q sets the expected size of the coalition, S, influencing the probability of including any particular player in a random subset. Each value of q corresponds to a different stratum of coalition sizes. In the context of the Shapley value, $e_i(q)$ represents the expected marginal contribution of player i when joining a randomly selected subset of players, S, from all subsets not containing i. This is akin to taking a sample from the stratum defined by q. The uniform distribution over the power set $\mathcal{P}(N \setminus \{i\})$ ensures that each potential coalition is equally likely to be chosen, analogous to ensuring equal representation of each stratum in the sample. The integration of $e_i(q)$ from 0 to 1 aggregates the contributions across all these strata. The expected contributions of player i across all possible coalition sizes, weighted by their probability, are summed up by integrating. This parallels how results from different strata are combined in stratified sampling to produce an overall estimate.