Data Driven Decoupling Of Multivariate Functions

Nathan Bouquet
Department of Advanced Computing Sciences
Faculty of Science and Engineering
Maastricht University
Maastricht, The Netherlands

***Abstract*—We consider the problem of decoupling multivariate functions using only input-output data, without requiring access to symbolic expressions. The approach is based on Canonical Polyadic Decomposition (CPD) applied to a third-order Jacobian tensor, which captures partial derivative information across each input point. We implement and compare several Jacobian estimation strategies, including finite difference methods, local regression techniques (linear, ridge, polynomial), and a global polynomial fit with analytical Jacobian evaluation. Finite difference methods validate the decoupling pipeline under exact function evaluations, with the complex-step method achieving machine-precision accuracy. In the data-driven setting, local polynomial regression is the most accurate Jacobian estimation method, achieving CPD errors as low as 10^{-6} for polynomial functions, and around 10^{-3} to 10^{-2} for more complex non-linear functions. The global polynomial fit performs well for polynomials but fails to generalize for non-linear functions. These results confirm that accurate Jacobian estimation and decoupling are feasible from input-output data alone, providing a foundation for interpretable non-linear modeling when symbolic functions are unavailable.**

I. Introduction

Multivariate polynomial functions appear in many areas of science and engineering, including system identification, signal processing, and control systems [1] [2]. These functions often involve complex interactions between multiple input variables, making it difficult to interpret or analyze them. One approach to simplifying such models is *tensor-based decoupling* [3] [4], which transforms a multivariate function into a sum of univariate functions composed with linear transformations of the inputs. This effectively reduces a multi-input multi-output (MIMO) problem into a set of simpler single-input single-output (SISO) problems, enhancing interpretability.

Mathematically, the goal is to express a vector-valued multivariate polynomial $\mathbf{f}(\mathbf{u})$ in a decoupled form:

$$\mathbf{f}(\mathbf{u}) = \mathbf{W}\,\mathbf{g}(\mathbf{V}^\top \mathbf{u}),$$

where $\mathbf{W}$ and $\mathbf{V}$ are transformation matrices, and $\mathbf{g}$ is a vector of univariate functions acting on scalar linear combinations of the input vector $\mathbf{u}$.

The method proposed by Dreesen et al. [3] achieves this decoupling by first computing the Jacobian matrix of $\mathbf{f}$ with respect to the input variables, arranging multiple Jacobians into a third-order tensor, and then applying CPD to extract the transformation matrices and the univariate structure.

However, this method assumes access to symbolic expressions of $\mathbf{f}$, which limits its applicability in real-world scenarios where only empirical data is available. During my research work, we extend the symbolic decoupling method to a data-driven setting, where Jacobians are estimated numerically from input-output data.

Central Research Question:
How can we accurately estimate the Jacobian tensor from data, and how does this affect the precision of the decoupling?
Sub-questions:

- Given access to function evaluations, how accurately can the Jacobian be estimated using finite difference methods?
- How accurately can Jacobians be estimated from only input-output data using local regression techniques?
- How do the number of sample points and the number of nearest neighbors k affect the quality of the Jacobian estimation and the resulting CP decomposition?
- How robust and precise are these estimation strategies when applied to non-linear functions beyond multivariate polynomials?

Through a series of experiments on synthetic multivariate functions, ranging from polynomials to more complex non-linear functions, we evaluate and compare several Jacobian estimation strategies. These include both exact (finite difference) and data-driven methods, such as local linear, ridge, and polynomial regression, as well as global polynomial fitting combined with complex-step differentiation. We assess how sample size, neighborhood size k, and regularization affect the accuracy of the estimated Jacobians and the precision of the resulting CP decomposition. Our results aim at bridging the gap between symbolic and data-driven decoupling, by demonstrating how such methods can be applied when only input-output data is available and by providing insight into the robustness and limitations of different estimation methods as function complexity increases.

II. Methods

This section describes the methods used to extend the symbolic decoupling framework into a data-driven setting. The process involves generating synthetic multivariate polynomial functions, estimating their Jacobian tensors from data using

This thesis was prepared in partial fulfilment of the requirements for the Degree of Bachelor of Science in Data Science and Artificial Intelligence, Maastricht University. Supervisor(s): Philippe Dreesen, Martijn Boussé.

numerical differentiation and local regression techniques, and performing CPD to extract a decoupled representation.

A. Symbolic Pipeline

As a baseline, we first replicated the symbolic decoupling method proposed by Dreesen et al. [3], where the Jacobian matrices of a multivariate polynomial $\mathbf{f}(\mathbf{u}) \in \mathbb{R}^m$ are computed analytically using MATLAB's symbolic toolbox. These Jacobians are stacked into a third-order tensor and decomposed using CPD, implemented via the Tensorlab library [5].

To verify the correctness of the decomposition, we computed the CPD approximation error

$$\varepsilon = \frac{\|\mathcal{J} - \hat{\mathcal{J}}\|_F}{\|\mathcal{J}\|_F},$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\hat{\mathcal{J}}$ is the reconstructed tensor from the decomposition. If ε is close to machine precision, which is in practical settings values below 10^{-10} [6], we considered the symbolic CPD to be accurate and reliable. This symbolic pipeline serves as a reference point for assessing the quality of the data-driven decoupling methods.

B. Finite Difference Methods

To estimate Jacobians from data points, we implemented several finite difference methods that approximate partial derivatives of $\mathbf{f}$ at sampled input locations. We let h denote a small perturbation step size and $\mathbf{e}_k$ the k-th unit vector. For each output component f_j, the partial derivative with respect to input variable u_k is approximated as follows:

- **Forward Differentiation:** this method is simple and one-sided, but suffers from first-order truncation error. It approximates the partial derivative of the function as

$$\frac{\partial f_j}{\partial u_k} \approx \frac{f_j(\mathbf{u} + h\mathbf{e}_k) - f_j(\mathbf{u})}{h},$$

  where $f_j(\mathbf{u} + h\mathbf{e}_k)$ is the j-th component of the function evaluated at a point $\mathbf{u}$ perturbed in the k-th direction by a small step size h, and $\mathbf{e}_k$ is the standard basis vector. This gives a first-order estimate of the slope of f_j with respect to u_k.
- **Backward Differentiation:** also first-order accurate, this method is useful when forward evaluations are unstable or infeasible. It approximates the partial derivative as

$$\frac{\partial f_j}{\partial u_k} \approx \frac{f_j(\mathbf{u}) - f_j(\mathbf{u} - h\mathbf{e}_k)}{h},$$

  where the function is evaluated at a point slightly behind $\mathbf{u}$ along the k-th coordinate direction.
- **Central Differentiation:** this symmetric method achieves second-order accuracy and generally offers improved precision. It uses function evaluations on both sides of $\mathbf{u}$ and approximates the derivative as

$$\frac{\partial f_j}{\partial u_k} \approx \frac{f_j(\mathbf{u} + h\mathbf{e}_k) - f_j(\mathbf{u} - h\mathbf{e}_k)}{2h}.$$

- **Complex Step Differentiation:** this technique avoids subtraction and achieves machine-precision accuracy for real-analytic functions. It approximates the derivative as

$$\frac{\partial f_j}{\partial u_k} \approx \frac{\mathrm{Im}(f_j(\mathbf{u} + \mathrm{i}h\mathbf{e}_k))}{h},$$

  where i is the imaginary unit, and $\mathbf{e}_k$ is the k-th unit vector in the input space.

These methods are used to estimate the Jacobian matrix $\mathbf{J}(\mathbf{u}) \in \mathbb{R}^{m \times n}$ at each evaluation point.

C. Data-Driven Jacobian Estimation

In addition to finite difference methods, we estimated Jacobians from only input-output data using local regression techniques. This enabled Jacobian approximation in scenarios where we did not have direct access to the function, but only to previously collected evaluations. This local estimation strategy is particularly useful in settings where data are irregularly spaced, as it does not rely on structured perturbations.

a) Local Linear Regression: For each evaluation point $\mathbf{u}^{(i)}$, we found the k closest input points using Euclidean distance, and computed the local input and output differences as:

$$\Delta\mathbf{X} = [\mathbf{u}^{(j)} - \mathbf{u}^{(i)}] \in \mathbb{R}^{m \times k}, \quad \Delta\mathbf{Y} = [\mathbf{f}(\mathbf{u}^{(j)}) - \mathbf{f}(\mathbf{u}^{(i)})] \in \mathbb{R}^{n \times k}.$$

The Jacobian $\mathbf{J}^{(i)} \in \mathbb{R}^{n \times m}$ is then estimated by solving the local system $\Delta\mathbf{Y} \approx \mathbf{J}^{(i)}\Delta\mathbf{X}$ using one of the following methods:

- **Unregularized (linear) regression:**

$$\mathbf{J}^{(i)} \approx \Delta\mathbf{Y}\,\Delta\mathbf{X}^{\dagger},$$

  where $\Delta\mathbf{X}^{\dagger}$ is the Moore–Penrose pseudoinverse of $\Delta\mathbf{X}$. This is implemented as a matrix division $\mathbf{J}^{(i)} = \Delta\mathbf{Y}/\Delta\mathbf{X}$ in MATLAB.
- **Ridge regression (regularized):**

$$\mathbf{J}^{(i)} \approx \Delta\mathbf{Y}\,\Delta\mathbf{X}^{\top}(\Delta\mathbf{X}\Delta\mathbf{X}^{\top} + \lambda\mathbf{I})^{-1},$$

  where $\lambda > 0$ is a regularization parameter.
  Regularization is implemented to improve robustness when the local input matrix $\Delta\mathbf{X}$ is ill-conditioned or nearly rank-deficient. This usually occurs when input points are closely clustered, leading to near-linear dependence among the rows of $\Delta\mathbf{X}$, which amplifies noise during matrix inversion. It can also arise when the underlying function is highly non-linear: in such cases, the local linear approximation becomes a poor fit, and small perturbations in the input produce large, uneven variations in the output. This can lead to large variances in the estimated derivatives, especially when a small neighborhood is used. Ridge regression mitigates these effects by penalizing large coefficients, which helps stabilize the solution in regions where the local neighborhood is poorly conditioned. While ridge regression may not always outperform linear regression especially for simple polynomial functions, it often reduces estimation error in more complex or curved regions of the input space [7].

b) Local Polynomial Approximation: To go beyond linear approximations, we also implement local polynomial regression. Around each center point $\mathbf{u}^{(i)}$, we collected its k-nearest neighbors and construct a regression model that fits a polynomial of degree d to the input-output differences. A polynomial basis of monomials up to degree d is generated, and the coefficients are estimated via least squares. The Jacobian is extracted from the coefficients of the first-order monomial terms.

Let $\Phi \in \mathbb{R}^{(k+1)\times K}$ be the design matrix formed by evaluating the multivariate monomials of degree up to d for each neighbor. Solving the system:

$$\Delta \mathbf{y}_l^{(i)} \approx \Phi^{(i)} \mathbf{c}_l^{(i)}, \quad \text{for each output } l = 1, \ldots, n$$

yields coefficient vectors $\mathbf{c}_l$ from which the Jacobian entries are extracted by identifying the linear monomial terms (e.g., x, y, z) in the basis.

This method allows for a better approximation of non-linear behavior and has shown more precise accuracy, particularly when the function exhibits mild curvature around the neighborhood of interest.

c) Global Polynomial Regression: In addition to local regression methods, we implemented a two-step procedure that first fits a global multivariate polynomial approximation of the function, and then estimates its Jacobian analytically from the fitted model. The polynomial model is constructed by generating a multivariate Vandermonde matrix of monomial terms up to a specified degree d, and solving a least-squares problem to determine the coefficients. This yields a closed-form polynomial approximation $\hat{\mathbf{f}}(\mathbf{x})$, which can be evaluated at any input.

Once the model is fitted, we analytically compute the Jacobian matrix at each input point by differentiating the polynomial with respect to each input variable. Since the polynomial structure is fully known from the coefficients, this allows efficient and precise Jacobian estimation using only input-output data.

D. Tensor Construction and CP Decomposition

After estimating Jacobians at N sample points, we stacked them into a third-order tensor $\mathcal{J} \in \mathbb{R}^{m\times n\times N}$, where each frontal slice $\mathcal{J}(:,:,i)$ corresponds to the Jacobian matrix at the i-th input point.

We applied CPD to approximate the tensor as a sum of rank-one components as

$$\mathcal{J} \approx \sum_{j=1}^{r} \mathbf{w}_j \circ \mathbf{v}_j \circ \mathbf{h}_j,$$

where $\mathbf{w}_j \in \mathbb{R}^m$, $\mathbf{v}_j \in \mathbb{R}^n$, and $\mathbf{h}_j \in \mathbb{R}^N$. The resulting factor matrices $\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_r]$ and $\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_r]$ are used to recover an approximate decoupled representation of the original function.

To assess the reconstruction quality of the decomposition, we use the CPD error introduced in Section II-A.

E. Visualization of Decoupling Accuracy

To visualize the quality of the decoupled approximation, we reconstructed the decoupled function $\hat{\mathbf{f}}(\mathbf{x}) = \mathbf{W}\,\mathbf{g}(\mathbf{V}^\top \mathbf{x})$ from the CPD factor matrices $\mathbf{W}$, $\mathbf{V}$ and a global polynomial fit of the univariate functions g_i. These functions are recovered using symbolic regression from projected inputs via a Vandermonde system. The reconstructed function was then evaluated on the input data points and compared to the true function values.

We generated a scatter plot of the reconstructed output $\hat{\mathbf{f}}(\mathbf{x})$ against the ground truth $\mathbf{f}(\mathbf{x})$ to assess accuracy. A tight alignment along the diagonal indicates a precise approximation.

III. Implementation

This section describes the implementation of the various Jacobian estimation methods and tensor decomposition steps used in this study. All methods were implemented in MATLAB.

A. Finite Difference Jacobian Estimation

To estimate the Jacobian at each sample point using finite differences, we perturbed each input dimension individually and approximated the corresponding partial derivatives. For each sample point:

- Evaluate the function at the unperturbed point;
- Perturb the input along each dimension according to the finite difference method (forward, backward, or central), and evaluate the function at the perturbed inputs;
- Compute the partial derivatives and populate the Jacobian matrix for that point.

The Jacobians for all points are stacked into a third-order tensor $\mathcal{J} \in \mathbb{R}^{n\times m\times N}$.

B. Complex-Step Differentiation

To estimate the Jacobian using the complex-step differentiation method, we perturbed each input dimension by adding a small imaginary component and evaluated the imaginary part of the function output. This method avoids subtractive cancellation and achieves high numerical precision. For each input variable x_j, where $j = 1, \ldots, m$, the function is evaluated at $\mathbf{x} + \mathrm{i}h\mathbf{e}_j$, and the derivative is computed accordingly.

C. Local Linear Regression Jacobian Estimation

To estimate Jacobians from only input-output data, we applied local linear regression using k-nearest neighbors:

- For each sample point, we identified the k nearest neighbors in the input space;
- Centered the neighborhood around the current sample point i by computing: $\Delta X = X(:, \text{neighbors}) - X(:, i)$, $\Delta Y = Y(:, \text{neighbors}) - Y(:, i)$;
- Solve the linear system: $\mathbf{J}_i \leftarrow \Delta\mathbf{Y}/\Delta\mathbf{X}$ via least squares.

The estimated Jacobians are stored in the tensor $\mathcal{J} \in \mathbb{R}^{n\times m\times N}$.

D. Local Polynomial Approximation

Local polynomial regression generalizes the local linear regression method by fitting a higher-degree polynomial to each neighborhood. The process is as follows:

- Generate monomial features up to a total degree d;
- For each point, collect $k+1$ neighbors and construct a local Vandermonde matrix;
- Center the neighborhood around the current sample point i by computing: $\Delta X = X(:, \text{neighbors}) - X(:, i)$, $\Delta Y = Y(:, \text{neighbors}) - Y(:, i)$.
- Solve a least-squares problem to fit polynomial coefficients for each output dimension
- Extract the linear coefficients corresponding to first-order monomials to construct the Jacobian.

E. Tensor Construction and CP Decomposition

The estimated Jacobian matrices $\mathbf{J}^{(i)} \in \mathbb{R}^{m \times n}$ for each sample point are stored as frontal slices in a third-order tensor $\mathcal{J} \in \mathbb{R}^{m \times n \times N}$. This tensor is passed to the `cpd` function from the Tensorlab library, specifying the decomposition rank r. The accuracy of the decomposition is assessed via the CPD approximation error shown in section (3.D). In the symbolic case, the error reaches machine precision, providing a baseline to evaluate the impact of numerical estimation and noise.

IV. Experiments

To evaluate the effectiveness of data-driven Jacobian estimation and its impact on the decoupling process, we conduct a series of experiments on synthetic multivariate functions. These functions are designed to have a known decoupled structure, allowing us to compare the estimated Jacobian tensor and its CPD decomposition directly against symbolic ground truth.

We investigate the behavior of three Jacobian estimation strategies: finite differences, local regression, and global polynomial fitting combined with the complex-step derivative method. The input points $\mathbf{u}^{(i)}$ are sampled randomly using MATLAB's `rand` function, which draws uniformly from the interval $[0,1]^n$ [8]. These samples are then shifted to match the desired input domain. To reduce the influence of sampling variability, each experiment is repeated over 20 independent trials. This number of repetitions provides a good balance between statistical reliability and computational cost, and is consistent with common practice in simulation-based studies [9]. Results are reported as the mean with error bars representing ±1 standard deviation.

To prevent numerical instability in finite difference approximations, step sizes smaller than 10^{-5} are excluded. This is consistent with standard practice in numerical differentiation, where very small perturbations can amplify round-off errors due to finite machine precision [10], [11]. All experiments are performed on a suite of synthetic multivariate functions of increasing complexity. These include:

- $f_1(\mathbf{u}) = \begin{bmatrix}(x+y)^2 \\ (x-y)^3\end{bmatrix}$, sampled from $[0,1]^2$. Decoupled as:
$$\mathbf{W} = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix}1 & 1 \\ 1 & -1\end{bmatrix}, \quad g(\mathbf{z}) = \begin{bmatrix}z_1^2 \\ z_2^3\end{bmatrix}$$
- $f_2(\mathbf{u}) = \begin{bmatrix}\sin(x+y)+z^2 \\ xyz + \cos(z)\end{bmatrix}$, sampled from $[-0.5, 0.5]^3$. Decoupled as:
$$\mathbf{W} = \mathbf{I}_2, \quad \mathbf{V} = \begin{bmatrix}1 & 0 \\ 1 & 0 \\ 0 & 1\end{bmatrix}, \quad g(\mathbf{z}) = \begin{bmatrix}\sin(z_1)+z_2^2 \\ z_1 z_2 + \cos(z_2)\end{bmatrix}$$
- $f_3(\mathbf{u}) = \begin{bmatrix}\exp(xy)+\sin(z) \\ \log(1+x^2+y^2) \\ (x+y+z)^3\end{bmatrix}$, sampled from $[-0.5, 0.5]^3$. Decoupled as:
$$\mathbf{W} = \mathbf{I}_3, \quad \mathbf{V} = \begin{bmatrix}1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1\end{bmatrix}, \quad g(\mathbf{z}) = \begin{bmatrix}\exp(z_1)+\sin(z_2) \\ \log(1+z_1^2) \\ z_3^3\end{bmatrix}$$

These functions were chosen to test the decoupling method across a variety of non-linear structures, including polynomial, trigonometric, exponential, and logarithmic components.

As an additional experiment, we assessed the accuracy of the decoupled functions recovered from CPD by comparing the reconstructed outputs $\hat{\mathbf{f}}(\mathbf{x})$ to the true function values $\mathbf{f}(\mathbf{x})$, as described in Section II-E. The scatter plot provides an intuitive measure of how well the decoupling preserves the original function behavior.

A. Finite Difference Jacobian Estimation

These experiments explore how the perturbation step size and number of input samples affect the accuracy of the Jacobian tensor and the quality of the CPD-based decoupling. Jacobians are estimated using forward, backward, central, and complex-step differentiation methods, and compared against symbolic ground truth.

- **CPD error vs. step size:** we assessed how the overall decoupling accuracy, as measured by CPD approximation error, varies with the finite difference step size h. It is expected that larger step sizes lead to increased error due to truncation error [10].
- **CPD error vs. number of sample points:** we fixed the step size to 10^{-5} and varied the number of input samples N, analyzing the impact on the CPD reconstruction error. Increasing N is expected to improve the approximation quality by providing a denser and more representative sampling of the input space [7].

B. Data-Driven Jacobian Estimation via Local Regression

These experiments explore a more realistic and challenging scenario in which Jacobians must be estimated from only input-output data. We evaluated the effectiveness of local linear regression, ridge regression, and local polynomial regression for this task, and assessed their impact on the quality

of the resulting decoupling across functions of increasing complexity.

- **CPD error vs. number of points N and nearest neighbors k:** we analyzed how the number of sample points N and the neighborhood size k affect the accuracy of the decoupling. Larger N generally improves performance by increasing the density of local neighborhoods. However, the choice of k involves a trade-off: using a too small neighborhood may yield unstable estimates, while a too large neighborhood may oversmooth the function [7]. For polynomial functions such as f_1, we expect local polynomial regression to outperform linear and ridge regression. For more complex functions like f_2 and f_3, which include logarithmic, exponential, and trigonometric components, it is unclear whether polynomial regression can generalize effectively.
- **CPD error vs. number of points N, nearest neighbors k, and regularization parameter λ:** we studied the influence of regularization on Jacobian estimation using ridge regression. For various combinations of N and k, we varied the regularization strength λ and record the resulting CPD error. A properly chosen λ is expected to improve numerical stability by addressing local ill-conditioning, especially in non-linear regions of the input space.
- **Linear vs. Ridge regression (with fixed λ):** after selecting an appropriate regularization parameter λ, we compared the performance of linear and ridge regression across functions of increasing complexity. The goal is to assess which method performs best for each function type. Ridge regression is expected to outperform unregularized regression in cases where the function is highly non-linear or where the local neighborhoods are ill-conditioned, such as for f_2 and f_3. For simpler functions like f_1, the benefits of regularization may be less significant.

C. Global Polynomial Regression

Finally, we evaluated the global polynomial regression method with analytical Jacobian evaluation. This approach fits a multivariate polynomial model to the entire input-output dataset and then estimates the Jacobian at each input point by differentiating the polynomial analytically with respect to each input variable. We assessed the decoupling performance using the CPD error versus number of sample points.

We anticipate excellent results for the purely polynomial function f_1, since the polynomial fit will capture its structure exactly. For more complex functions f_2 and f_3, which contain sinusoidal, exponential, and logarithmic components, it is unclear whether a polynomial approximation will generalize well; these experiments are used to explore this question.

V. Results and Discussions

This section presents the results of the experiments, focusing on how step size, sample size, and regression parameters influence the accuracy of Jacobian estimation and the quality of the resulting decoupling. The main evaluation metric is the CPD approximation error, complemented by additional analyses of Jacobian estimation accuracy.

A. Finite Difference Experiments

Initial experiments used finite difference methods to validate the symbolic pipeline, analyzing how the step size h and number of sample points N affect the CPD error. Among these methods, the complex-step method gave the most accurate results, achieving CPD errors near machine precision ($\approx 10^{-16}$). This confirms its known superiority for derivative estimation in settings where function evaluations are available [12]. The central difference method also performed very well, although its accuracy degraded for very small step sizes due to round-off effects [10]. These results provided a baseline for evaluating data-driven Jacobian estimation strategies.

a) CPD Error vs. Step Size.: for both forward and backward differentiation methods, the CPD approximation error decreases as the step size h becomes smaller (see Figure 1). This behavior is consistent with expectations from truncation error theory [10], where smaller step sizes lead to more accurate derivative estimates. In contrast, the central difference method shows an increase in error when h decreases below 10^{-3} (see Figure A2). Although this may seem counterintuitive, it is explained by numerical round-off effects, which dominate at very small step sizes due to subtractive cancellation [10]. Despite this, the central method still outperforms forward and backward differences overall, achieving a CPD error of approximately 10^{-15} at a step size of 10^{-2}. The complex-step method performs best, maintaining CPD error at machine precision ($\approx 10^{-16}$) across all step sizes (see Figure A3). This confirms its superior numerical stability and accuracy.

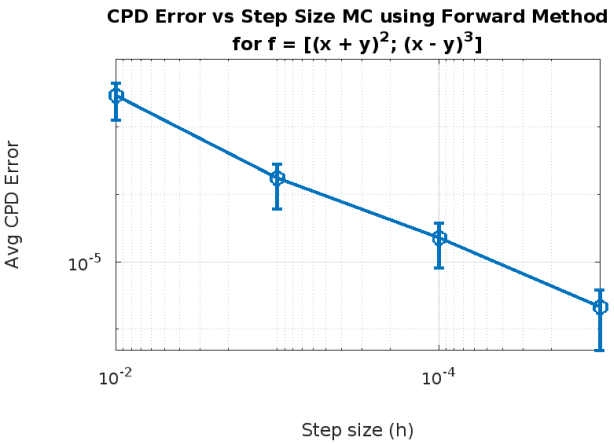

Fig. 1. CPD error against step size using the forward difference method. As expected, the error decreases as the step size h becomes smaller, consistent with truncation error theory, which predicts improved accuracy for smaller h [10].

b) CPD Error vs. Number of Sample Points.: as expected, increasing the number of sampled points leads to more

accurate decompositions. For the forward method, backward and central differentiation methods, the CPD error decreases steadily with an increase in sample size (see Figure A5), with the central method consistently achieving the highest accuracy amongst the three. The complex-step method yields the lowest overall CPD error, remaining close to machine precision ($\approx 10^{-16}$) across all sample sizes (see Figure A6). Its performance appears relatively flat, with only minor variation—aside from a small irregularity around $N = 500$. This suggests that while the complex-step method is highly stable, it is less sensitive to changes in sample density compared to the other finite difference methods.

behavior. Notably, smaller neighborhood sizes such as $k = 5$ and $k = 10$ produced relatively poor results for these complex functions, with CPD errors around 10^{-1} to 10^{-2} for $k = 5$ (see Figure A8), and around 10^{-2} for $k = 10$ (see Figure A9). This suggests that more neighbors are needed to capture sufficient local structure when the function is highly non-linear. For f_1, however, the CPD error remains low even at $k = 5$, staying around 10^{-3} (see Figure A7). The differences in CPD error across methods remained relatively small, especially for larger N. These results suggest that all methods face limitations in capturing complex behavior, though local polynomial regression appears slightly more robust.

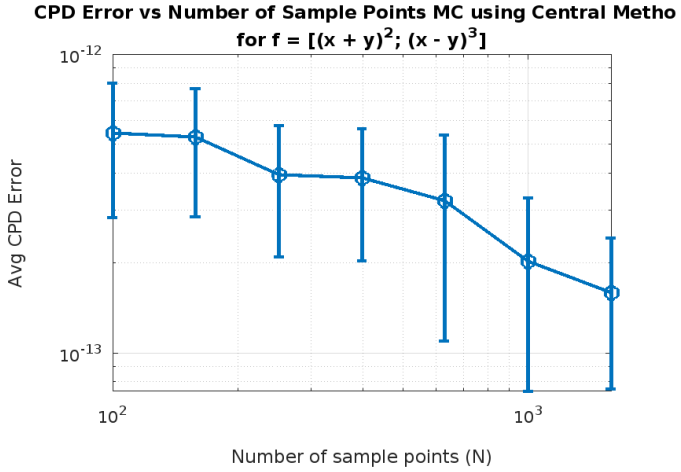

Fig. 2. CPD error vs. number of sample points N using the central difference method. Although the decrease is not perfectly linear, a clear downward trend is observed as N increases. The smallest CPD error is approximately 10^{-12} at $N = 3000$. The relatively large error bars indicate variability across different random samplings.

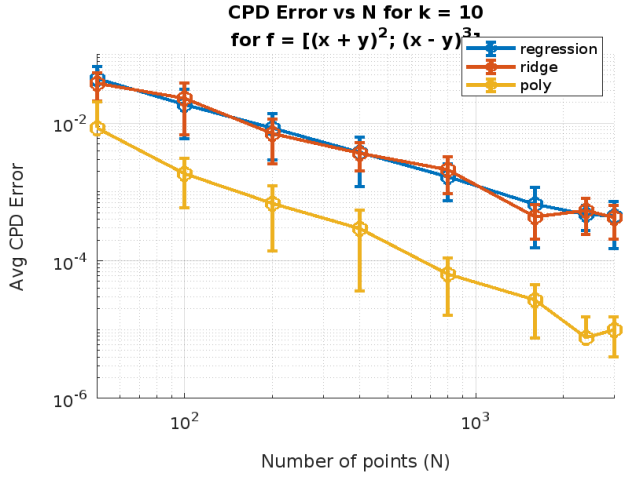

Fig. 3. CPD error vs. number of sample points N using local regression methods with $k = 10$ nearest neighbors for function f_1. A clear decreasing trend is observed for all methods as N increases, with polynomial regression yielding the lowest CPD error across all sample sizes.

B. Local Regression Experiments

In the data-driven setting, we implemented and evaluated local linear, ridge, and polynomial regression methods under varying sample sizes N and neighborhood sizes k. Across all methods, increasing N consistently led to lower CPD errors, validating the benefit of denser sampling [7]. Choosing an appropriate number of neighbors also proved important: having too small a value of k produced unstable estimates, whereas using too large a value led to over-smoothing.

a) CPD Error vs. Number of Points and Nearest Neighbors.: Using local linear regression, ridge regression, and polynomial regression to estimate Jacobians from passively sampled input-output data yields stable and interpretable results. For the polynomial function f_1, polynomial regression produced the best results, achieving a minimum CPD error of 6.82×10^{-6} with $N = 3000$ and $k = 10$ (see Figure 3). For more complex functions such as f_2 and f_3, polynomial regression still outperformed the other methods, although the CPD error increased to the range of 10^{-3}–10^{-2} (see Figure 4), reflecting the added difficulty of capturing non-polynomial

b) Effect of Regularization.: adding regularization to local linear regression yields results that are very similar to the unregularized case. Smaller values of the regularization parameter λ, such as 10^{-5}, consistently lead to lower CPD error across different sample sizes. These results confirm that combining a moderate number of neighbors with low regularization and a sufficiently large sample size offers the best trade-off between robustness and accuracy. The best result observed for f_1 in this setting was a CPD error of 4.85×10^{-4}, achieved with $N = 3000$, $k = 10$, and $\lambda = 10^{-5}$ (see Figure 5), which is virtually identical to the minimum error obtained without regularization. For larger regularization parameters, such as $\lambda = 10^{-2}$, performance degrades noticeably. CPD errors increase to around 10^{-2}, and become more sensitive to the choice of k, with error curves for different neighbor values spreading widely. For instance, only $k = 50$ yields an error near 10^{-3}, while $k = 10, 20, 30$ result in higher CPD errors in the range of 10^{-2} (see Figure A10).

c) Linear vs. Ridge Regression: after selecting an appropriate regularization parameter $\lambda = 10^{-5}$, we compared the performance of linear and ridge regression across functions of increasing complexity. In all cases, the CPD error consis-

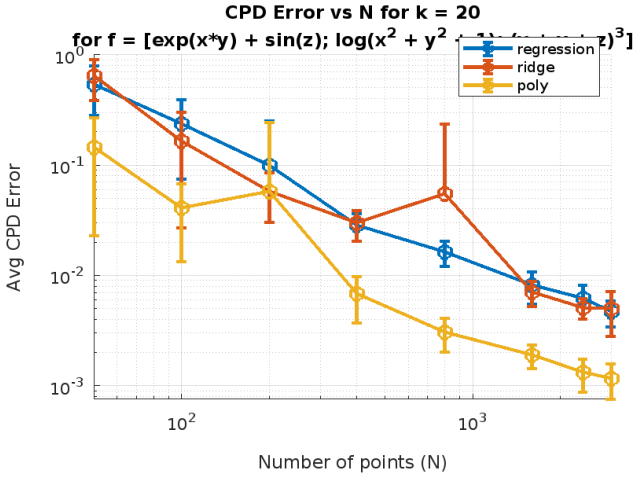

Fig. 4. CPD error vs. number of sample points N using local regression methods with $k = 20$ nearest neighbors for function f_3. All methods show a clear decrease in error as N increases, with polynomial regression achieving the lowest CPD error across all sample sizes. The minimum error reaches approximately 10^{-3} at $N = 3000$, indicating that polynomial regression performs best even for complex, non-linear functions.

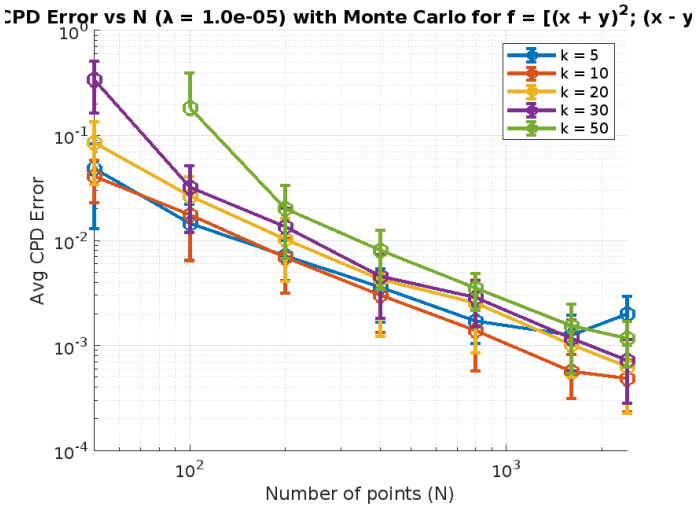

Fig. 5. Effect of regularization parameter $\lambda = 10^{-5}$ on CPD error across different sample sizes N and numbers of neighbors k. The error curves for different k values show similar behavior, with minor variation for $k = 30$ and $k = 50$. For all cases, the CPD error decreases with increasing N and stabilizes around $10^{-3.5}$.

tently decreases as the number of sample points N increases, reaffirming that denser sampling improves the accuracy of Jacobian estimation and the resulting decoupling.

For the polynomial function f_1, the lowest CPD error was 4.01×10^{-4}, achieved with $N = 3000$ and $k = 20$ using local linear regression (see Figure 6). Both linear and ridge regression perform similarly for $k > 5$, with nearly overlapping error trends across N. For the more complex functions f_2 and f_3, ridge regression achieves lower CPD errors of 3.05×10^{-2} and 3.69×10^{-3}, compared to 3.36×10^{-2} and 3.58×10^{-3} respectively for linear regression (see Figures A11, A12). While ridge regression performs slightly better on the more complex functions, the differences in CPD error remain small, and both methods exhibit nearly identical trends across all settings.

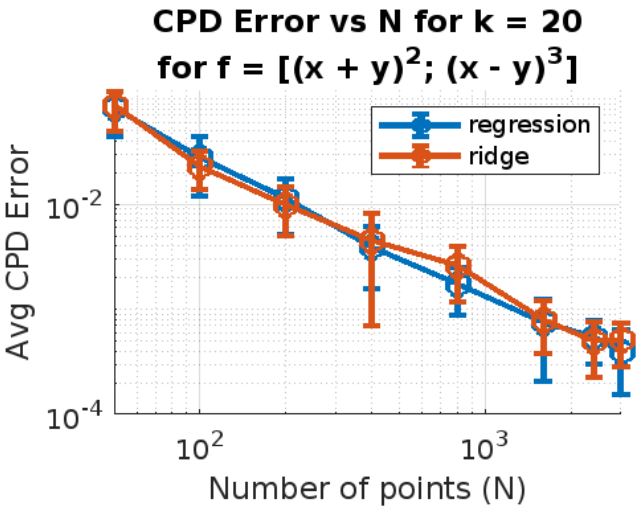

Fig. 6. CPD error against the number of sample points N for function f_1, comparing local linear and ridge regression (with $\lambda = 10^{-5}$) using $k = 20$ neighbors. Both methods follow a similar decreasing trend, with local linear regression achieving the lowest CPD error of 4.01×10^{-4}.

C. Global Polynomial Regression

The global polynomial regression method performs nearly identically to complex-step differentiation for the polynomial function f_1. In the CPD error versus number of sample points experiment, the error remains stable across all values of N, achieving values around 10^{-16} (see Figure 7). This confirms that the polynomial fit accurately captures the structure of the underlying polynomial function.

However, for the more complex functions f_2 and f_3, which include sinusoidal, exponential, and logarithmic components, the method fails to generalize. In the CPD error versus number of sample points experiment, the error is significantly higher, around 10^{-3} with irregular trends for f_2 (see Figure A13), and a slight improvement for f_3, which decreases to approximately 10^{-3} at $N = 1000$ before stabilizing. This reflects the limited ability of polynomial fits to approximate more complex nonlinear behaviors.

These results indicate that although global polynomial regression is highly effective for purely polynomial functions, it struggles to represent non-polynomial functions. Consequently, the analytically derived Jacobians from such fits are inaccurate in these cases, highlighting the limitations of this method for general nonlinear function estimation.

D. Decoupled Function Reconstruction Accuracy

Beyond CPD error, it is essential to evaluate how well the decoupled representation recovers the original function outputs. To assess this, we compared the reconstructed output values $\hat{\mathbf{f}}(\mathbf{x})$ against the true outputs $\mathbf{f}(\mathbf{x})$ across the

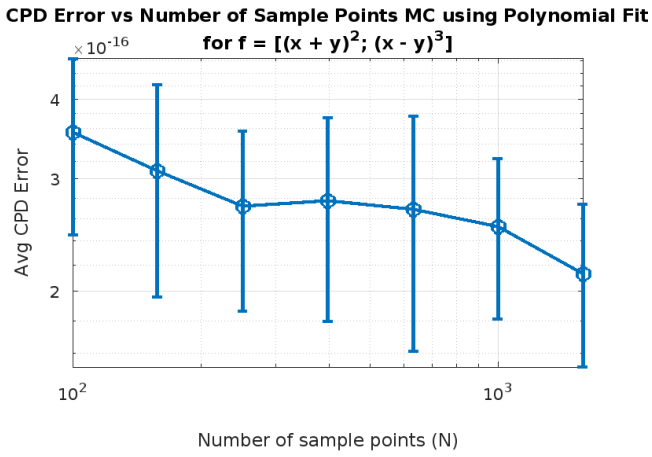

Fig. 7. CPD error vs. number of sample points for the Global Polynomial Regression applied to f_1. The error remains nearly constant around 10^{-16}, confirming that the polynomial fit perfectly captures the structure of the underlying polynomial function.

input space. Figure 8 presents a scatter plot of true vs. reconstructed outputs for function f_3, using the polynomial regression method. The tight clustering along the diagonal indicates accurate function reconstruction. This confirms that the combination of local polynomial Jacobian estimation along with the CPD decoupling can preserve the structure of even complex nonlinear functions.

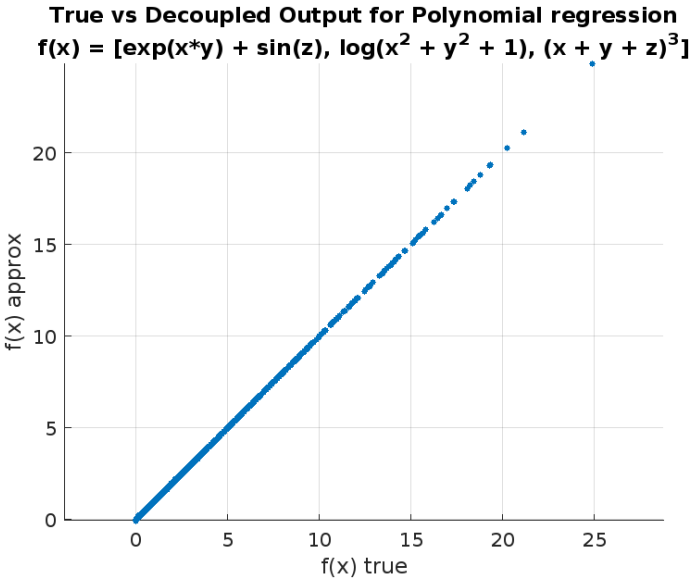

Fig. 8. Scatter plot comparing reconstructed vs. true function outputs for f_3, using polynomial regression with 1000 sample points. The strong diagonal alignment indicates that the decoupled representation successfully reconstructs the nonlinear structure of the original function.

VI. Conclusion

This thesis set out to extend the decoupling framework for multivariate functions into a data-driven setting, where Jacobians are estimated from input-output data and used to compute CPD for function decoupling. We implemented and evaluated multiple estimation methods, including finite difference methods, local regression techniques, and a global polynomial fit. A key contribution of this work is the systematic evaluation of how sample size, neighborhood size, and regularization influence Jacobian estimation and decoupling performance. This type of analysis provides valuable insight into how these methods behave under realistic constraints where symbolic access to the function is unavailable.

Finite difference methods served to validate the symbolic decoupling pipeline. The complex-step method demonstrated machine-precision CPD errors, confirming its accuracy and stability. These results provided a baseline for comparison with data-driven approaches. In the data-driven setting, local polynomial regression produced the most accurate results. For polynomial functions, it achieved CPD errors as low as 6.82×10^{-6}, while for more complex non-linear functions, the CPD error ranged from 10^{-3} to 10^{-2}. Increasing the number of sample points N consistently improved Jacobian accuracy, and selecting an appropriate number of neighbors k proved important to avoid underfitting or over-smoothing. The global polynomial fit performed excellently on polynomial functions but failed to generalize to more non-linear structures, reinforcing the strength of local methods for such cases.

These findings demonstrate that accurate Jacobian estimation and decoupling are feasible without symbolic access to the function. They also highlight the trade-offs between different methods depending on function complexity, sampling density and local neighborhood structure, providing a foundation for interpretable non-linear modeling in data-driven settings.

This study focused on synthetic data under ideal conditions. One key limitation is that the methods, show reduced performance on non-linear functions. Moreover, only uniform random sampling was explored. Future work could evaluate how alternative sampling densities or input distributions affect performance, and further investigate model robustness on real-world datasets.

In summary, this thesis bridges the gap between symbolic and data-driven decoupling of multivariate functions by presenting a set of reliable, interpretable, and accurate methods for estimating Jacobians and recovering decoupled representations from input-output data alone.

References

[1] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall, 1999.

[2] R. Pintelon and J. Schoukens, *System Identification: A Frequency Domain Approach*, 2nd ed. John Wiley & Sons, 2012.

[3] P. Dreesen, M. Ishteva, and J. Schoukens, "Decoupling multivariate polynomials using first-order information and tensor decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 2, pp. 864–879, 2015.

[4] P. Comon, "Tensors: A brief introduction," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 44–53, 2014.

[5] N. Vervliet, O. Debals, L. Sorber, M. V. Barel, and L. D. Lathauwer, "Tensorlab 3.0—numerical optimization strategies for large-scale structured tensor problems," *ESAT-SCD Technical Report*, 2016. [Online]. Available: https://www.tensorlab.net

[6] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.
[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.
[8] The MathWorks, Inc., *rand - Uniformly distributed random numbers*, MathWorks, 2024, https://www.mathworks.com/help/matlab/ref/rand.html.
[9] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
[10] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, 7th ed. McGraw-Hill Education, 2015.
[11] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced grids," *Mathematics of Computation*, vol. 51, no. 184, pp. 699–706, 1988.
[12] W. Squire and G. Trapp, "Using complex variables to estimate derivatives of real functions," *SIAM Review*, vol. 40, no. 1, pp. 110–112, 1998.

APPENDIX

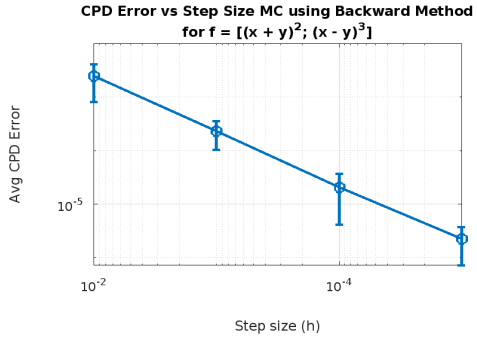

Fig. A1. CPD error vs. step size for the backward difference method. As expected, the error decreases with smaller h, confirming the trade-off between numerical accuracy and step size resolution.

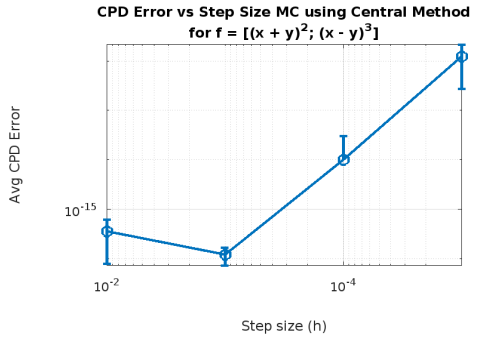

Fig. A2. CPD error vs. step size for the central difference method. The error remains low ($\approx 10^{-15}$) for $h \geq 10^{-3}$, but increases at smaller h due to round-off errors from subtractive cancellation [10].

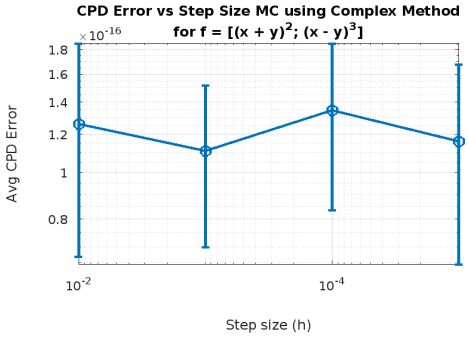

Fig. A3. CPD error vs. step size for the complex-step method. The error remains constant near 10^{-16} across all tested step sizes, demonstrating exceptional stability and precision. This confirms the method's advantage in avoiding subtractive cancellation, and highlights it as the most accurate among all tested finite differentiation methods.

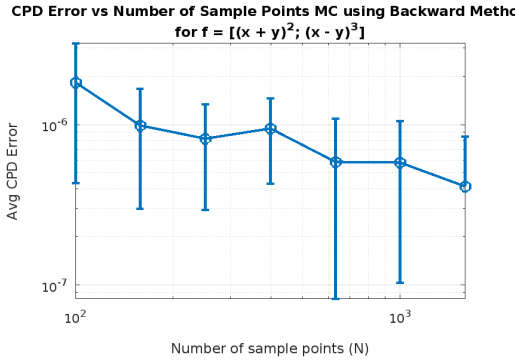

Fig. A4. CPD error vs. number of sample points N for the backward difference method. While the decrease is not perfectly linear, a clear trend of decreasing error with increasing N is visible. The relatively large error bars reflect high variability across different random samplings, highlighting sensitivity to sample placement.

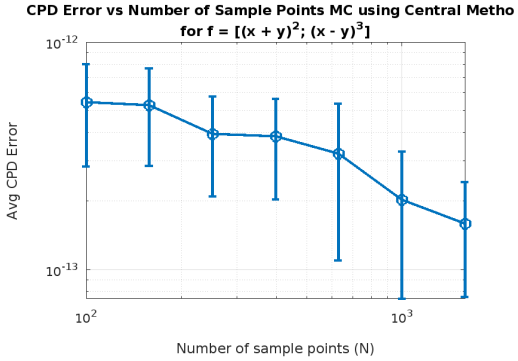

Fig. A5. CPD error vs. number of sample points N for the central difference method. A clear decreasing trend is observed as N increases, with the error reaching values of 10^{-12}. This reflects the method's high accuracy and strong performance under dense sampling.

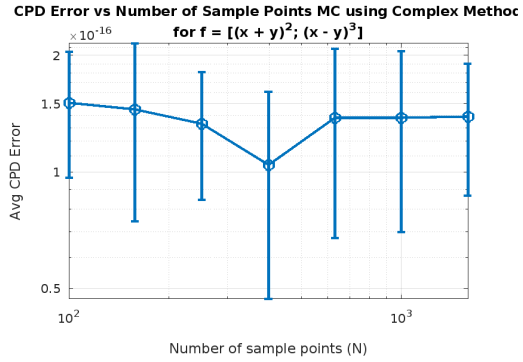

Fig. A6. CPD error vs. number of sample points N for the complex-step method. The error remains constant near 10^{-16} across all values of N, demonstrating exceptional numerical stability and machine-precision accuracy regardless of sampling density.

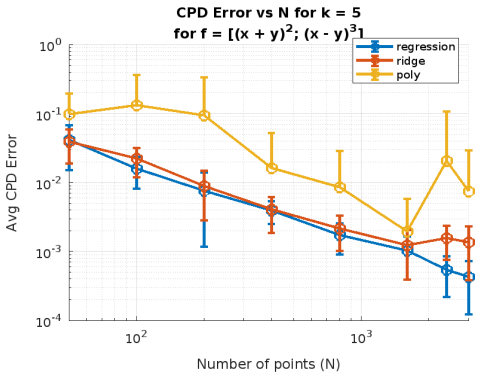

Fig. A7. CPD error vs. number of sample points N using local regression methods with $k = 5$ nearest neighbors for function f_1. A clear decreasing trend is observed for both local linear and ridge regression as N increases, with linear regression achieving the lowest CPD error across all sample sizes. Polynomial regression initially decreases but begins to increase again after $N \approx 1500$.

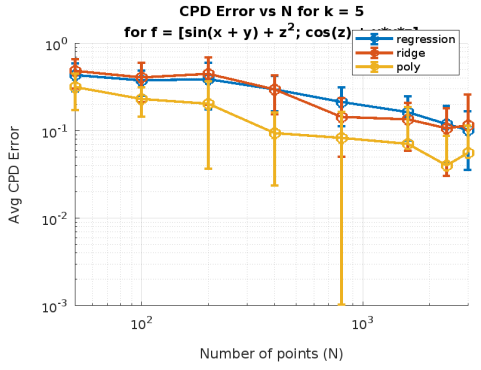

Fig. A8. CPD error vs. number of sample points N using local regression methods with $k = 5$ nearest neighbors for function f_1. A slight decreasing trend is observed for all methods as N increases, with polynomial regression yielding the lowest CPD error across all sample sizes around 10^{-1}.

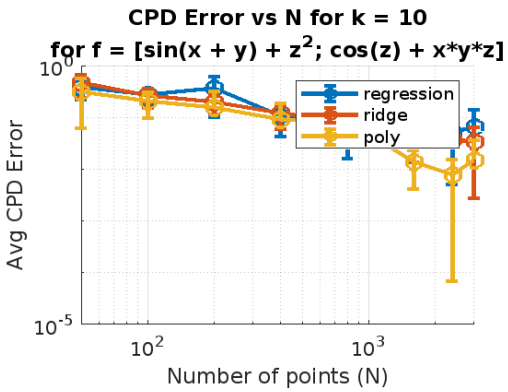

Fig. A9. CPD error vs. number of sample points N using local regression methods with $k = 10$ nearest neighbors for function f_1. A slight decreasing trend is observed for all methods as N increases, with polynomial regression yielding the lowest CPD error across all sample sizes around 10^{-2}.

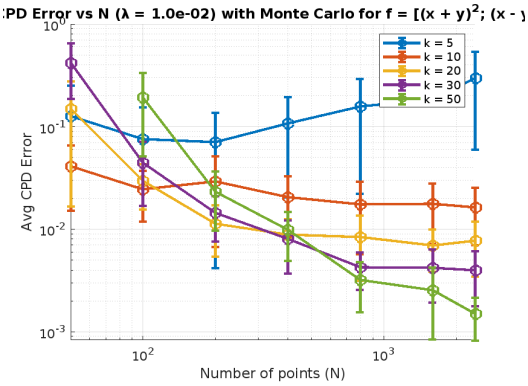

Fig. A10. Effect of regularization parameter $\lambda = 10^{-2}$ on CPD error across varying numbers of points N and neighbors k. For $k = 5$, the CPD error remains high and relatively flat around 10^{-1}, indicating poor performance. For $k = 10, 20, 30$, and 50, a clear downward trend is observed as N increases, with final CPD errors approaching 10^{-3}.

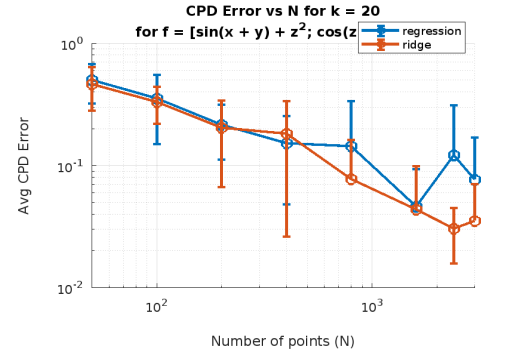

Fig. A11. CPD error vs. number of sample points N for function f_2, comparing local linear and ridge regression (with $\lambda = 10^{-5}$) using $k = 20$ neighbors. Both methods show similar behavior up to approximately $N = 1000$. Beyond that point, the error for linear regression fluctuates, while ridge regression continues to decrease smoothly, converging to around $10^{-1.5}$ at $N = 3000$. Ridge regression performs best for f_2.

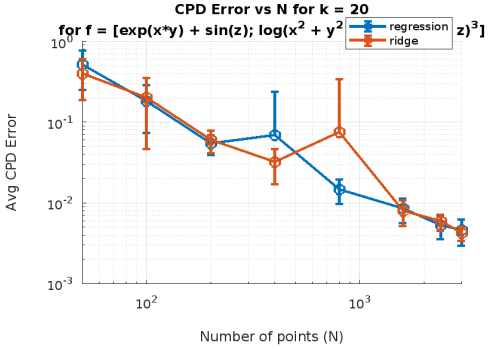

Fig. A12. CPD error vs. number of sample points N for function f_3, comparing local linear and ridge regression (with $\lambda = 10^{-5}$) using $k = 20$ neighbors. Both methods show similar behavior, with the error decreasing to approximately $10^{-2.5}$ at $N = 3000$. Small fluctuations are observed around $N = 750$ for linear regression and $N = 1000$ for ridge regression.

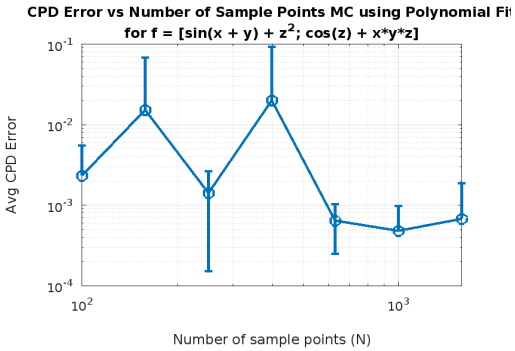

Fig. A13. CPD error vs. number of sample points N for the Global Polynomial Regression applied to f_2. The error fluctuates across sample sizes and stabilizes around 10^{-3} near $N = 750$, indicating limited generalization to non-polynomial functions.