

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Learning-Augmented and Structure-Preserving Methods for Conservation Law Solvers

Permalink

<https://escholarship.org/uc/item/9r16v0bg>

ISBN

9798273395190

Author

DeGrendele, Christopher Joseph

Publication Date

2025-12-10

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

**LEARNING-AUGMENTED AND STRUCTURE-PRESERVING
METHODS FOR CONSERVATION LAW SOLVERS**

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

by

Christopher DeGrendele

December 2025

The Dissertation of Christopher
DeGrendele is approved:

Professor Dongwook Lee, Chair

Professor Ashesh Chattopadhyay

Professor Nicholas Brummell

Peter Biehl
Vice Provost and Dean of Graduate Studies

Copyright © by
Christopher DeGrendele
2025

Table of Contents

List of Figures	vi
List of Tables	xv
Abstract	xviii
Dedication	xx
Acknowledgments	xxi
1 Introduction	1
1.1 Conservation Laws	1
1.1.1 Hyperbolic, parabolic, and mixed conservation laws	2
1.1.2 Compressible Euler equations	3
1.1.3 Hyperbolic conservation laws and discretization	4
1.1.4 High-order fluxes and the order barrier near shocks	8
1.1.5 Time integration	10
1.2 Machine Learning and Statistical Surrogates	13
1.2.1 Basic concepts in supervised learning	14
1.2.2 From pointwise models to operator learning	16
1.2.3 Fourier Neural Operator (FNO)	17
1.2.4 UNO and related architectures	17
1.2.5 Gaussian processes as statistical surrogates	18
2 Gaussian Process Based Numerical Methods	20
2.1 Introduction	20
2.1.1 Previous Studies of GP Regression in Shock-capturing Schemes .	23
2.1.2 Extending GP’s Data Conversion Property to Interpolations and Linear Operators	25
2.2 Gaussian Process for Modeling Functions	26
2.2.1 GP Prior and Posterior Predictions	26
2.2.2 Extending GP’s Posterior Prediction to Linear Operations	29
2.3 Choosing a Kernel Function and GP Stencils	31
2.3.1 Squared Exponential (SE) Kernel	31

2.3.2	1D GP Stencils	34
2.3.3	2D GP Stencils	35
2.4	List of Applications	37
2.5	Kernels for Smooth Function Approximations and Numerical Results . .	39
2.5.1	Constrained Kernels	39
2.6	Conclusion	57
3	Convergence of Kernel Based Methods	59
3.1	Introduction	59
3.2	Convergence Analysis	63
3.2.1	Taylor Expansion in GP Predictions to Analyze GP's Effective Rate of Convergence	63
3.2.2	Effective Order of Convergence (EOC)	66
3.3	Choosing a Kernel Function and GP Stencils	67
3.3.1	Squared Exponential (SE) Kernel	68
3.4	Results: Operator-constrained data prediction	68
3.4.1	Numerical setup	69
3.5	Results: Choosing Optimal Hyperparameters	75
3.5.1	Choice of Stencil	76
3.5.2	Choice of Kernel Length Scale ℓ	83
3.6	Results: Numerical PDEs	86
3.6.1	1D Diffusion	88
3.7	Conclusion	91
4	Shock Capturing in Gaussian Process Methods	94
4.1	DAS Kernel for Non-Oscillatory Approximation to Discontinuous Functions	94
4.1.1	Conclusion	102
4.2	GP-MOOD Framework	103
4.2.1	Methodology	103
4.2.2	Numerical Results	105
4.2.3	Conclusion	106
5	Fully Conservative Pressure Equilibrium Preserving Methods	114
5.1	Introduction	114
5.2	Derivation of the Pressure Equilibrium Preserving Condition	117
5.3	Derivation of High-Order APEC	123
5.3.1	Summary of Proposed Scheme - High-Order APEC	128
5.3.2	Error Analysis of the High-Order APEC Scheme	128
5.4	Derivation of a Fully Conservative PEP Scheme	135
5.4.1	An Example PEP Scheme	136
5.4.2	Equal/near-equal states	137
5.4.3	Proposed PEP Scheme	137
5.4.4	Extension to Multiple Dimensions and N Species	142
5.4.5	Summary of Proposed PEP Scheme	143
5.4.6	Limiting behavior and consistency of the scheme	143

5.5	Taylor Series Expansion	146
5.5.1	Taylor Expansion Method	146
5.6	Results	148
5.6.1	1D smooth interfaces advection problem - calorically perfect gas	148
5.6.2	1D smooth interfaces advection problem - Stiffened Gas	152
5.6.3	1D smooth interfaces advection problem — van der Waals	155
5.7	Taylor Results	162
5.8	Sensitivities to Reciprocal Condition Number	169
5.8.1	Local behavior of α and β	169
5.8.2	Global pressure-error behavior across EOS	170
5.9	Extension of the Proposed PEP Scheme to Single Species	172
5.10	Conclusion	173
6	Coupling of Machine Learning Based Diffusion	177
6.1	Introduction	177
6.2	Methodology	181
6.2.1	Mathematical formulation and coupling	182
6.2.2	Framework I: E-IVP (Explicit-IVP)	182
6.2.3	Framework II: I-DS (Implicit-Dataset)	183
6.2.4	Training Methodology	185
6.2.5	Stability analysis	187
6.3	Burgers Diffusion: Framework I (E-IVP) Model Evaluation	188
6.3.1	Architecture Comparison	189
6.4	Burgers Diffusion: Framework II (I-DS) Evaluation	192
6.4.1	Kappa Generalization and Loss Behavior	193
6.4.2	Spectral Behavior Across κ	195
6.4.3	Trajectory from $t = 0$	196
6.4.4	Numerical Properties of the Learned Diffusion Operator for Burgers	197
6.5	Euler IVP with Implicit Heat Conduction (Framework II)	205
6.5.1	Loss-Function Behavior	206
6.5.2	Spectral Error Structure	207
6.6	Conclusion	207
7	Multiphase LaunchPad Simulations	213
7.0.1	Motivation	213
7.0.2	Introduction	215
7.0.3	Artemis Launchpad Simulations	218
7.0.4	Simulation Methodology	219
7.0.5	Integrated Loads	224
7.0.6	Frequency Analysis of Sensors	226
7.0.7	Conclusion	232
	Bibliography	235

List of Figures

1.1	Top: finite difference grid representing pointwise values at cell location i . Bottom: finite volume grid representing cell-averaged values at cell location i	7
2.1	1D GP stencil. The red “*” marks indicate the i -th cell interface locations we choose for x_* in 1D. The accuracy of the native GP point interpolations at x_* is $(2r_{gp} + 1)$, showing (a) the 3rd-order stencil, (b) the 5th-order stencil, and (c) the 7th-order stencil, where r_{gp} denotes the respective integer-valued GP stencil radius from the cell center x_i . A higher-order GP stencil can be easily configured with $r_{gp} > 3$	35
2.2	2D GP stencils of $r_{gp} = 2$. The central cell $\mathbf{x}_m = (x_{i_m}, y_{j_m})$ is the cell where GP predictions are computed at the surrounding interface locations, $(x_{i_m \pm 1/2}, y_{j_m})$ or $(x_{i_m}, y_{j_m \pm 1/2})$, or corner locations, $(x_{i_m \pm 1/2}, y_{j_m \pm 1/2})$	36
2.3	GP’s prediction of $\nabla^2 \sin(x) \sin(y)$ on $[0, 2\pi] \times [0, 2\pi]$ using a 128×128 grid resolution with $r_{gp} = 2$ and $\ell = \pi/10$. The GP result shows a great visual agreement with the exact Laplacian calculation. The convergence study for this case is shown in Table 2.7.	48
3.1	Effective order of convergence for the 2D point-to-point (Pt→Pt, left) and volume-average to point (VA→Pt, right) operators. Both use the blocky-diamond stencil and predict in the upper-left corner, $\mathbf{x}_* = (x_i + \frac{1}{2}\Delta x, y_i + \frac{1}{2}\Delta y)$. The colors denote GP stencil radius r_{gp} (red: $r_{gp} = 1$, blue: $r_{gp} = 2$, green: $r_{gp} = 3$). Dashed lines show the observed numerical convergence rates, and solid lines show the theoretical proxy p_{eff} from the Taylor expansion analysis.	74
3.2	Effective order of convergence for the 2D point-to-derivative (Pt→ ∂_y , left) and volume-average to second derivative (VA→ ∂_{xx} , right) operators. Both use the blocky-diamond stencil and predict in the upper-left corner, $\mathbf{x}_* = (x_i + \frac{1}{2}\Delta x, y_i + \frac{1}{2}\Delta y)$. Colors indicate stencil radius (red: $r_{gp} = 1$, blue: $r_{gp} = 2$, green: $r_{gp} = 3$), with dashed lines for numerical results and solid lines for the theoretical p_{eff}	75

3.3	Effective order p_{eff} (solid) and measured convergence rates (dashed) for $\text{Pt} \rightarrow \partial_y$ (left) and $\text{VA} \rightarrow \partial_x$ (right) at a corner location with $r_{\text{gp}} = 3$. Colors denote stencil geometry: cross (red), diamond (blue), blocky-diamond (green), and square (purple). The diamond, blocky-diamond, and square stencils exhibit very similar accuracy and closely follow the theoretical prediction, while the cross stencil converges more slowly and is slightly underpredicted by p_{eff}	81
3.4	Effective order p_{eff} (solid) and measured convergence rates (dashed) for $\text{Pt} \rightarrow \partial_x$ evaluated at an x -interface for $r_{\text{gp}} = 3$. All four stencil geometries share the same predicted effective order, and the numerical slopes agree closely, with the square stencil attaining the smallest absolute error and the cross stencil lagging in magnitude while retaining the same order.	82
3.5	Leave-one-out change in p_{eff} for $\text{VA} \rightarrow \text{Pt}$ on the square stencil. Left: $r_{\text{gp}} = 2$. Each cell shows the decrease in effective order when that stencil point is removed, revealing which locations contribute most strongly to the reconstruction accuracy. Right: $r_{\text{gp}} = 3$. The dominant contributions remain clustered in a diamond-like region around the target corner, while far-corner points have very small marginal impact on p_{eff}	83
3.6	Leave-one-out change in p_{eff} for $\text{VA} \rightarrow \partial_{xy}$ on the square stencil. Left: $r_{\text{gp}} = 2$, where the most influential points already hint at a blocky-diamond structure aligned with the predicted corner location. Right: $r_{\text{gp}} = 3$, where the blocky-diamond region becomes even more pronounced, with the point nearest the predicted corner changing p_{eff} by $\mathcal{O}(10^{-1})$ while the far corners contribute at the level of $\mathcal{O}(10^{-2})$, motivating the blocky-diamond stencil as a reduced geometry that preserves almost all of the attainable accuracy.	84
3.7	Effective order proxy p_{eff} for the $\text{VA} \rightarrow \partial_{xx}$ operator with $r_{\text{gp}} = 3$ as a function of the maximum Taylor order N_T retained in the series. The curve rapidly approaches the measured convergence rate (approximately fifth order) as N_T increases. By $N_T = 15$ the predicted order is already very close to the observed value, and by $N_T = 20$ it is essentially converged; adding further terms up to $N_T = 30$ yields only diminishing improvements.	85
3.8	Effect of the SE kernel length scale on the $\text{VA} \rightarrow \text{Pt}$ operator using the blocky-diamond stencil with $r_{\text{gp}} = 2$, evaluated at the corner interface. The solid curve (left axis) shows the predicted effective order $p_{\text{eff}}(\ell/h)$; the dashed curve (right axis) shows the measured L^1 error at $N = 256$. The optimal ratio $\ell/h \approx 8$ simultaneously maximizes p_{eff} and minimizes the numerical error, demonstrating that the OOA framework provides an accurate and inexpensive predictor of optimal hyperparameters.	87

3.9	Relative L_2 error for the one-dimensional diffusion equation $u_t = \mu u_{xx}$ with periodic boundary conditions and initial condition $u(x, 0) = \sin(2\pi x)$. The spatial operator is constructed from a GP with an SE kernel and centered stencils of radius $r_{gp} \in \{1, 2, 3\}$. Solid lines show the measured errors; dashed lines indicate the predicted convergence slopes p_{theory} obtained from the GP-Taylor analysis of the diffusion operator $\mathcal{D}_{xx}^{\text{GP}}$. The close agreement demonstrates that the OOA framework correctly predicts the spatial order of the discrete operator and the resulting PDE right-hand side.	90
4.1	Heat map comparison of kernels. (Top left) SE kernel with $\ell = 1.0$. (Top right) SE kernel with $\ell = 0.35$. (Bottom left) NN kernel with $\sigma = \sigma_0 = 1$. (Bottom right) DAS kernel.	97
4.2	Interpolation across a single discontinuity using our GP point-to-point interpolation comparing different kernels. $N_x = 20$ for the left panel and $N_x = 40$ for the right. (Left) The hyperparameters of the kernels are varied on each kernel holding $r_{gp} = 2$. The SE kernel is varied over ℓ values of 0.1, 1.0, and 10. The NN kernel is varied over σ values of 1, 10, and 100. We keep $\sigma_0 = 1$ as it has little effect on the solution. DAS does not have any hyperparameters and is held at constant. (Right) The stencil order is varied from $r_{gp} = 1, 2, 3$ for each kernel choice. The SE kernel utilizes $\ell = 0.1$ and the NN kernel uses $\sigma = 100$ and $\sigma_0 = 1$	108
4.3	Comparison of point-to-point interpolation on three different profiles using the SE, NN, and DAS kernels. The NN kernel uses $\sigma = \sigma_0 = 1$, and the SE kernel uses $\ell = 12\Delta x$. (Top) The function, f_1 , is a piecewise function defined on $[0, 1.5]$ by $f_1(x) = 0$ for $0 \leq x < 0.3$ and $1.0 \leq x \leq 1.5$; $f_1(x) = e^{-2x} \sin(10\pi x) + 1/2$ for $0.3 \leq x < 0.6$; $f_1(x) = 1$ for $0.6 \leq x < 1.0$. (Bottom) The function, f_2 , is a piecewise function defined on $[-4, 4]$ by $f_2(x) = \sin(\pi x)$ for $ x \geq 1.0$; $f_2(x) = 3$ for $-1.0 < x \leq -0.5$ and $0 < x \leq 0.5$; $f_2(x) = 1$ for $-0.5 < x \leq 0.0$; $f_2(x) = 2$ for $0.5 < x \leq 1.0$. All GP calculations were performed using $N_x = 100$ grid resolutions on each domain with $r_{gp} = 2$. The reference exact solution is shown using $N_x = 1000$	109
4.4	Condition number of $\mathbf{K}$ plotted as a function of grid spacing Δx . (Left) Comparison of the three kernels for various stencil sizes. The hyperparameters are held at $\ell = 1.8$ for SE and $\sigma = \sigma_0 = 1$ for NN, while r_{gp} is varied for all cases. (Right) Comparison of the three kernels for hyperparameter values. We set $r_{gp} = 2$ for all kernels and hyperparameters are varied according to the legend.	110
4.5	Plot of pressure with contours on the left and the order of the MOOD reconstruction on the right. The top figures are for original 5th order MOOD method, with SE kernel everywhere, where the bottom figures use the DAS kernel for the 2nd order method. Both methods present great symmetry and we see an increase in the higher order method when using the DAS kernel which will lead to greater accuracy overall.	111

4.6	Plot of $\log 10(\rho)$ of the Mach 800 Astrophysical Jet problem varying in time. In the bottom two panes we can see some asymmetries start to develop that did not occur in the original GP-MOOD 5th order simulation. Here we use the SE kernel for the 5th order method, DAS kernel for the 3rd (2nd) order method and FOG as a last resort. Top Left: $t = 0.00875$ sec, Top Right: $t = 0.016875$ sec, Bottom Left: $t = 0.0253125$ sec, Bottom Right: $t = 0.0340625$ sec.	112
4.7	Plot of order of reconstruction of the Mach 800 Astrophysical Jet problem varying in time. In the bottom two panes we can see some asymmetries start to develop that did not occur in the original GP-MOOD 5th order simulation. Here we use the SE kernel for the 5th order method, DAS kernel for the 3rd (2nd) order method and FOG as a last resort. Top Left: $t = 0.00875$ sec, Top Right: $t = 0.016875$ sec, Bottom Left: $t = 0.0253125$ sec, Bottom Right: $t = 0.0340625$ sec	113
5.1	Construction of the high-order interface flux at $i + 1/2$ as a weighted sum of multiple cell-pair contributions. Dashed gray lines connect cell centers to their arrow origins. For second-order accuracy (blue), a single pair $\langle i, i + 1 \rangle$ forms the sub-interface state. Fourth-order accuracy adds two additional pairs: $\langle i - 1, i + 1 \rangle$ (orange) and $\langle i, i + 2 \rangle$ (green). Sixth-order accuracy further extends the stencil with three more pairs: $\langle i - 2, i + 1 \rangle$ (red), $\langle i - 1, i + 2 \rangle$ (purple), and $\langle i, i + 3 \rangle$ (brown). Each arrow pair represents a sub-interface contribution formed by averaging the pointwise values at two cell centers, and the final interface flux is the weighted sum of all these sub-interface contributions according to Equation 5.35. . . .	126
5.2	Second-order numerical solutions at $t = 20$ for the ideal-gas test case defined in Equation 5.116. The four panels show: (top-left) the proposed PEP scheme; (top-right) the APEC scheme of [142]; (bottom-left) the PEP method of [55]; and (bottom-right) the second-order KEEP scheme (FC-NPE). In each panel, blue, orange, green, and red denote ρY_1 , ρY_2 , u , and p , respectively.	151
5.3	L_2 pressure error (left) and total-energy error (right) for the ideal-gas test case. Curves correspond to the APEC $\mathcal{O}(2)$ scheme (blue), the FC-NPE scheme (orange), the Fujiwara PEP scheme (green), the double-flux APE scheme (red), and the proposed PEP scheme (purple).	152
5.4	L_2 pressure error (left) and total-energy error (right) for the higher-order APEC and NPE schemes applied to the ideal-gas test. APEC curves are plotted as solid lines and NPE curves as dashed lines, with colors denoting the formal order of accuracy: $\mathcal{O}(2)$ (blue), $\mathcal{O}(4)$ (orange), $\mathcal{O}(6)$ (green), and $\mathcal{O}(8)$ (red). The proposed PEP scheme is shown in purple for reference.	153

5.5	Second-order numerical solutions at $t = 7$ for the stiffened-gas test case defined in Equation 5.116. The four panels show: (top-left) the proposed PEP scheme; (top-right) the APEC scheme of [142]; (bottom-left) the PEP method of [55]; and (bottom-right) the second-order KEEP scheme (FC-NPE). In each panel, blue, orange, green, and red denote ρY_1 , ρY_2 , u , and p , respectively.	156
5.6	L_2 pressure error for the stiffened-gas test case. Curves correspond to the APEC $\mathcal{O}(2)$ scheme (blue), the FC-NPE scheme (orange), the Fujiwara PEP scheme (green), and the proposed PEP scheme (red).	157
5.7	L_2 pressure error for the higher-order APEC and NPE schemes applied to the stiffened-gas test. APEC curves are plotted as solid lines and NPE curves as dashed lines, with colors denoting the formal order of accuracy: $\mathcal{O}(2)$ (blue), $\mathcal{O}(4)$ (orange), $\mathcal{O}(6)$ (green), and $\mathcal{O}(8)$ (red). The proposed PEP scheme is shown in purple for reference.	158
5.8	Relative L_2 pressure error for the ideal-gas test case using the Taylor-regularized family Taylor- (n) with $n = 1, \dots, 5$. Curves show the NPE scheme (gray), the proposed PEP scheme (black), and Taylor-regularized variants from first through fifth order: $n = 1$ (blue), $n = 2$ (orange), $n = 3$ (green), $n = 4$ (red), and $n = 5$ (purple). All runs employ the same second-order base discretization, with n indicating the highest power of $\Delta(\rho Y_i)$ retained in the Taylor surrogate.	163
5.9	Relative L_2 pressure error for the stiffened-gas mixture test case using the Taylor-regularized family Taylor- (n) with $n = 1, \dots, 5$. Curves show the NPE scheme (gray), the proposed PEP scheme (black), and Taylor-regularized variants from first through fifth order: $n = 1$ (blue), $n = 2$ (orange), $n = 3$ (green), $n = 4$ (red), and $n = 5$ (purple). All runs employ the same second-order base discretization, with n indicating the highest power of $\Delta(\rho Y_i)$ retained in the Taylor surrogate.	164
5.10	Relative L_2 pressure error for the van der Waals mixture test case using the Taylor-regularized family Taylor- (n) with $n = 1, \dots, 5$. Curves show the NPE scheme (gray), the proposed PEP scheme (black), and Taylor-regularized variants from first through fifth order: $n = 1$ (blue), $n = 2$ (orange), $n = 3$ (green), $n = 4$ (red), and $n = 5$ (purple). All runs employ the same second-order base discretization, with n indicating the highest power of $\Delta(\rho Y_i)$ retained in the Taylor surrogate.	165
5.11	Normalized profiles at $t = 0.2$ for the van der Waals mixture test. The four panels show: (top-left) the initial condition; (top-right) the proposed PEP scheme; (bottom-left) the APEC $\mathcal{O}(2)$ scheme; and (bottom-right) the second-order NPE (KEEP) baseline. In each panel, ρY_1 , ρY_2 , u , and p are scaled as indicated in the legend to fit on common axes and are colored blue, orange, green, and red respectively.	166

5.12	Pressure (left) and temperature (right) profiles at $t = 0.2$ for the van der Waals test case, comparing the initial condition, the proposed PEP scheme, APEC $\mathcal{O}(2)$, and the second-order NPE (KEEP) baseline. Colors: blue – initial condition, orange – NPE, green – APEC $\mathcal{O}(2)$, red – proposed PEP.	167
5.13	Relative L_2 pressure error $\varepsilon_p(t)$ for the van der Waals test, comparing the proposed PEP scheme, APEC $\mathcal{O}(2)$, and the second-order NPE (KEEP) baseline. Colors: black – proposed PEP, blue – APEC $\mathcal{O}(2)$, orange – FC-NPE baseline.	167
5.14	Relative L_2 pressure error for higher-order APEC and NPE (KEEP) schemes, together with the proposed PEP method, applied to the van der Waals mixture test. APEC curves are shown as solid lines and NPE curves as dashed lines. Colors indicate formal order and scheme: blue – $\mathcal{O}(2)$, orange – $\mathcal{O}(4)$, green – $\mathcal{O}(6)$, red – $\mathcal{O}(8)$, purple – proposed PEP (formally second order).	168
5.15	Local sensitivity of the PEP correction coefficients α (left) and β (right) to the reciprocal condition number in an ideal-gas two-species mixture. Each curve shows α or β as a function of the species jump $\Delta(\rho Y_i)$. The analytic closed-form solution (no regularization) displays strong oscillations as $\Delta(\rho Y_i) \rightarrow 0$, while the APEC baseline (grey) corresponds to $\alpha = \beta = 0$. The pseudoinverse results with fixed <code>rcond</code> values interpolate between these limits, and the automatic r_g -based choice yields a smooth decay of α and β to zero near the determinant-limiting regime.	171
5.16	Sensitivity of long-time pressure error to the reciprocal condition number used in the Moore–Penrose pseudoinverse for the three EOS configurations (ideal-gas, stiffened-gas, van der Waals). Each panel shows the ℓ_1 pressure error versus flow-through time. The second-order APEC scheme and the FC-NPE baseline are plotted as references, along with the proposed fully conservative PEP scheme for several fixed <code>rcond</code> values and the automatic choice r_g from Eq. (5.88).	176
6.1	Architecture comparison for the Burgers E–IVP setting. All models have $\approx 0.2\text{M}$ parameters and are trained with the same baseline MSE loss.	189
6.2	UNO loss–function comparison for Burgers E–IVP.	191
6.3	Spectral error vs time for UNO under the same loss configurations as in Figure 6.2.	192
6.4	Representative spectral–error heatmaps for UNO under various loss configurations.	193
6.5	Spectral error vs time for UNO trained with different noise levels.	194
6.6	Spectral–error heatmaps for different noise levels in UNO (MSE–only).	195
6.7	UNO L_2 error vs. time for Burgers I–DS with $\kappa = 0.06$ across several loss configurations.	196
6.8	UNO L_2 error vs. time for Burgers I–DS with $\kappa = 0.17$ across several loss configurations.	197

6.9	UNO L_2 error vs. time for Burgers I–DS with $\kappa = 0.22$ across several loss configurations.	198
6.10	UNO L_2 error vs. time for Burgers I–DS with $\kappa = 0.28$ across several loss configurations.	199
6.11	Spectral–error heatmap for Burgers I–DS with $\kappa = 0.06$ across UNO loss configurations.	200
6.12	Spectral–error heatmap for Burgers I–DS with $\kappa = 0.28$ across UNO loss configurations.	201
6.13	Rollouts from $t = 0$ for several κ values (top) and for $\kappa = 0.045$ over longer time horizons (bottom). UNO_all_on maintains much better long–time fidelity than UNO_mse.	202
6.14	Comparison of spectral error when predicting the diffusive divergence directly versus predicting the flux F and taking a discrete divergence. Predicting F leads to amplification of high–frequency error when the divergence operator is applied.	203
6.15	Drift in total conserved quantity over time when predicting $\nabla \cdot F$ directly versus predicting F and differencing it. Predicting F leads to a noticeable conservation drift, while predicting the divergence is substantially more conservative.	204
6.16	Top: L_2 error vs. time for the two crossed configurations UNO_MUSCL→WENO (unstable) and UNO_WENO→MUSCL (stable). Bottom: corresponding maximum eigenvalue of the one–step Jacobian. Configuration A experiences an eigenvalue blowup exceeding 9 and fails at $t \approx 0.57$, while configuration B maintains a roughly constant eigenvalue near 4 and remains stable over the full 2.5s rollout.	205
6.17	Configuration A (UNO_MUSCL→WENO). Physical profiles compared to ground truth at $t = 0.52, 0.54, 0.56$. The solution rapidly destabilizes, consistent with the eigenvalue blowup.	206
6.18	Configuration B (UNO_WENO→MUSCL). Profiles at $t = 0.6, 0.75, 1.0$ (Framework I continuation). The solution remains stable with moderate error but no runaway growth.	207
6.19	Spectral error heatmap for configuration A. By $t \approx 0.54$, errors exceed $\mathcal{O}(1)$ across the spectrum, matching the onset of instability.	208
6.20	Spectral error heatmap for configuration B. Some low–frequency error develops, but high–frequency behavior remains controlled and the solution stays bounded.	209
6.21	Maximum eigenvalue of the one–step implicit prediction operator as a function of time for a sweep of diffusivities $\kappa \in [10^{-3}, 3]$. Smaller κ values produce eigenvalues closer to 1, while larger κ yields stronger damping (eigenvalues further below 1). All curves remain below 1, indicating linear stability across the entire κ range.	210
6.22	UNO loss-function comparison for Euler I–DS at $\kappa = 0.5$, continued from $t = 0.5$ to $t = 2.0$. The all–on loss yields the lowest error, followed by substep 10 and the spectral loss.	211

6.23	Spectral-error heatmaps for Euler I-DS at $\kappa = 0.5$. Most loss configurations exhibit similar patterns, with substep 5 showing a noticeably poorer high-frequency behavior.	212
7.1	Lofted Artemis I launch simulation using the LAVA Cartesian computational fluid dynamics solver. The image shows a slice through the exhaust plumes where the black line indicates the 90% plume exhaust/air interface. Launchpad surfaces are colored by pressure, where red is high and blue is low. The water mass fraction is shown using nested isocontours in teal.	219
7.2	Lofted Artemis I launch simulation using the LAVA Cartesian computational fluid dynamics solver. Plume gas mass fraction is shown where white is large and dark orange is low. Launchpad surfaces are colored by pressure, where red is high and blue is low. The water mass fraction is shown using nested isocontours in teal.	220
7.3	Lofted Artemis I launch simulation using the LAVA Cartesian computational fluid dynamics solver. The image shows a slice through the exhaust plumes where the black line indicates the 90% plume exhaust/air interface. Launchpad surfaces are colored by pressure, where red is high and blue is low. The water mass fraction is shown using nested isocontours in teal	220
7.4	Slice of the mesh of the Artemis II Max Drift 9 second case where boxes are plotted, each box is 8 cells wide in each spatial dimension. Left: +x view looking in the west direction. Right: +y view looking in the south direction. The mobile launcher is oriented such that the tower is on the north size of the pad. In this case, you can see the $8dx_{\text{fine}}$ mesh as the outer cone and surrounding the tower structure. Slightly inside of this is the $4dx_{\text{fine}}$ mesh that extends about 2/3rds of the way down the tower. Inside this, there is the $2dx_{\text{fine}}$ mesh cone at the exit of the nozzles. The dx_{fine} mesh is not visible, but is on the outlet of the nozzles.	222
7.5	Pressure signal from a point probe located on top of the tower for Artemis II Max Drift 9 seconds. At the start of the simulation ($t = 9$ sec), we can see the transient behavior as the pressure oscillates. Eventually, the signal becomes statistically stationary and we do analysis on the final 2 seconds of the signal indicated by the purple box, throwing away the first 2 seconds. The red points are the actual sensor locations and the blue ones are the nearest point on the triangulation of the geometry.	224
7.6	Here the maximum pressure experienced over the last two seconds of simulation is plotted for the Artemis I 07 second simulations. This is to compare the pressure loading for wet vs dry to see how the water systems play a role. These are plotted using the same color bar. In the wet simulation, we see significantly higher pressures on the deck surrounding the exhaust hole, in the exhaust hole, as well as some hot spots. Left: Wet Artemis I 07 sec, Right: Dry Artemis I 07 sec.	226

7.7	Here the maximum pressure experienced over the last two seconds of simulation is plotted for the Artemis I 07 second simulations. This is to compare the pressure loading for wet vs dry to see how the water systems play a role. This is the same visualization and data from 7.6 just zoomed in to see the elevator shaft that was damaged during launch. Here we clearly see some hot spots in the wet simulation that are not there in the dry. Left: Wet Artemis I 07 sec, Right: Dry Artemis I 07 sec.	227
7.8	Here the standard deviation of pressure experienced over the last two seconds of simulation is plotted for the Artemis I 07 second simulations. We see a higher standard deviation indicating higher pressure fluctuations. Left: Wet Artemis I 07 sec, Right: Dry Artemis I 07 sec.	228
7.9	Plotted is a bar graph where we compare the average total pressure force over the last 2 seconds of simulation on each component. In blue is the wet Artemis I 07 sec simulation and in red the dry Artemis I 07 sec wet simulation. For most of the components, water provided a damping effect decreasing the total pressure. However, for some outliers, the pressure is increased.	229
7.10	Plotted is the time series of the normal component of pressure on the east facing elevator shaft for the last 2 seconds of simulation time. This is the component that experienced the highest increase in pressure force going from the dry simulation to the wet simulation. In blue, the wet Artemis I 07 sec simulation is plotted and the dry Artemis I 07 sec simulation is plotted in red. Here we can see a significantly higher pressure sustained.	230
7.11	Pressure plotted as a function of time for the Artemis I sensor data in black. The blue points represent the average pressure over the last two seconds of simulation time for the Artemis I dry simulation and the orange points represents the Artemis I wet simulation. Left: A sensor on the far side of the tower. Right: A sensor on the plume side of the tower. In light blue, we label Sensor Drift when the sensor reads nonphysical pressure values.	231
7.12	Overall Sound Pressure Level (OASPL) plotted as a function of time for the Artemis I sensor data in black. The blue points represent the OASPL value computed using the last two seconds of simulation time for the Artemis I dry simulation and the orange points represents the Artemis I wet simulation. Left: A sensor on the far side of the tower. Right: A sensor on the plume side of the tower.	232
7.13	Sound Pressure Level (SPL) plotted as a function of frequency for the Artemis I sensor data in orange. The blue curves represent the SPL values computed using the last two seconds of simulation time for the Artemis I dry simulation and the green curve represents the Artemis I wet simulation. Left: A sensor on the far side of the tower. Right: A sensor on the plume side of the tower. Each vertical pane represents a different simulation, in order from top to bottom this is the Artemis I 3 second, 5 second, 7 second, 7.76 second, and 9 second simulations. . . .	234

List of Tables

2.1	A list of GP-recipe applications, including the baseline point-evaluation operator denoted by PE, e.g., $\mathcal{L}(g(\mathbf{x})) = g(\mathbf{x})$ for a function g . The input is a N -vector $\mathbf{g}$ whose h th component $[\mathbf{g}]_h = g(\mathbf{x}_h)$ is shown in the first column. The output in the second column is a scalar function value evaluated at $\mathbf{x} = \mathbf{x}_*$. Following Eq. (2.5), $\mathcal{L}_1$ is applied to $K(\mathbf{x}, \mathbf{x}')$ and $\mathcal{L}_2$ is applied to $K(\mathbf{x}_*, \mathbf{x}')$, acting on either $\mathbf{x}$ or $\mathbf{x}'$, or both. For simplicity, we assume that all calculations are performed using Cartesian geometry. In the first and second columns, we also show three abbreviations, “pt”, “va”, and “der”, representing pointwise, volume-averaged, and derivative data types in each input and output pair. The notation I_h is reserved to denote a 1D cell $[x_{h-1/2}, x_{h+1/2}]$ centered at x_h , and similarly for J_k centered at y_k , as well as for I_* centered at x_* and for J_* centered at y_* .	38
2.2	Grid convergence results for the 1D point-to-point interpolation in Section 2.5.1.1. GP calculations are performed using the SE kernel with $r_{gp} = 1, 2, 3$ to approximate the function values $f(x_{i+1/2})$ from the cell center input values $f(x_i)$ for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$. We choose $\ell = 0.05$ for the length scale hyperparameter.	42
2.3	Grid convergence results for the 2D point-to-point interpolation in Section 2.5.1.2. GP calculations are performed using the SE kernel with $r_{gp} = 1, 2, 3$ to approximate the function values $f(x_{i+1/2}, y_{j+1/2})$ from the cell center input values $f(x_i, y_j)$ for $f(x, y) = e^{-2x} \sin(4\pi y) + x e^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$. As discussed, we set the blocky diamond stencil as our default 2D stencil configuration. We choose $\ell = 0.05$ for the length hyperparameter.	43
2.4	Grid convergence results for the 1D point-to-1st derivative conversion in Section 2.5.1.3. GP calculations are performed at cell interface locations $x_{i+1/2}$ using the kernel in Eq. (2.19) with $r_{gp} = 1, 2, 3$ and $\ell = 0.05$ to approximate the function $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$.	45
2.5	Grid convergence results for the 1D point-to-2nd derivative conversion in Section 2.5.1.3 using the kernel in Eq. (2.20). All other GP (hyper)parameters are the same as in Table 2.4.	46

2.6	Grid convergence results for the 1D point-to-3rd derivative conversion in Section 2.5.1.3 using the kernel in Eq. (2.21). All other GP (hyper)parameters are the same as in Table 2.4.	47
2.7	Grid convergence results for the 2D point-to-Laplacian conversion in Section 2.5.1.4. GP considers to approximate the Laplacian of the function $f(x, y) = \sin(x) \sin(y)$ on $[0, 2\pi] \times [0, 2\pi]$ at $(x_{i+1/2}, y_{j+1/2})$ by taking the function values $f(x_i, y_j)$ at cell centers as input. The prediction utilizes the kernel in Eq. (2.26) with $\ell = \pi/10$	49
2.8	Grid convergence results for the 1D point-to-volume average conversion in Section 2.5.1.5. Input values are $f(x_i)$ for the function $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$ and GP predicts $\frac{1}{\Delta x} \int_{I_i} f(x) dx$ at x_i using the kernel in Eq. (2.28) with $\ell = 0.05$	50
2.9	Grid convergence results for the 2D point-to-volume average conversion in Section 2.5.1.6. Input values are $f(x_i, y_j)$ for the function $f(x, y) = e^{-2x} \sin(4\pi y) + x e^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$ and GP predicts $\frac{1}{\Delta x \Delta y} \int_{I_i} \int_{I_j} f(x, y) dx dy$ at (x_i, y_j) using the kernel in Eq. (2.30) with $\ell = 0.05$	52
2.10	Grid convergence results for the 1D volume-average-to-point conversion in Section 2.5.1.7. Input values are the integral average values $\frac{1}{\Delta x} \int_{I_i} f(x) dx$ at cell centers x_i and output values are pointwise function values $f(x_{i+1/2})$ at cell interface locations for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$. GP utilizes the kernel in Eq. (2.31) with $\ell = 0.05$	54
2.11	Grid convergence results for the 2D volume average-to-point conversion in Section 2.5.1.8. Input values are the integral average values $\frac{1}{\Delta x \Delta y} \int_{I_i} \int_{I_j} f(x, y) dx dy$ at cell centers (x_i, y_j) and output values are pointwise function values $f(x_{i+1/2}, y_{i+1/2})$ at cell corner locations for $f(x, y) = e^{-2x} \sin(4\pi y) + x e^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$. GP utilizes the kernel in Eq. (2.32) with $\ell = 0.05$	56
2.12	Grid convergence results for the 2D volume-average-to-point conversion in Section 2.5.1.9. Input values are the integral average values $\frac{1}{\Delta x \Delta y} \int_{I_i} \int_{I_j} f(x, y) dx dy$ at cell centers (x_i, y_j) and output values are the cross derivative $\partial_{xy} f(x_{i+1/2}, y_{i+1/2})$ evaluated at cell corner locations for $f(x, y) = e^{-2x} \sin(4\pi y) + x e^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$. GP utilizes the kernel in Eq. (2.33) with $\ell = 0.05$	58
3.1	Anticipated target orders (third column) and GP's effective order of convergence (EOC in Eq. (3.7)) for six data approximations at $\mathbf{x}_*$ (fourth – sixth columns) on four different GP stencils with $r_{gp} = 2$. EOCs are calculated at (i) the central cell in the fourth column, (ii) the x interfaces of the central cell in the fifth column, and (iii) the cell corners of the central cell in the sixth column.	93
4.1	Grid convergence results for the 1D baseline point-to-point interpolation using the NN kernel in Eq. (4.1). The NN prediction takes cell center function values $f(x_i)$ as input for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$ and computes $f(x_{i+1/2})$ using $\sigma = \sigma_0 = 1$	98

4.2	Grid convergence results for the 1D baseline point-to-point interpolation using the DAS kernel in Eq. (4.2). The DAS prediction takes cell center function values $f(x_i)$ as input for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$ and computes $f(x_{i+1/2})$	99
4.3	Comparison of SE and DAS kernels for intermediate order in MOOD cascade.	107
5.1	Coefficients (a_{q,i_s}) for central difference approximation of first derivative as shown in [80] and derived in [53]. q is the order of accuracy of the method and i_s is the stencil size.	125
6.1	Loss-function configurations used in the comparison study.	187
7.1	Summary of the suite of simulations by trajectory time with cell count in millions, vehicle elevation, wall clock time and cpu hours. All the dry simulations were run on 150 AMD Rome nodes each with 128 cores. The wet simulation was run on 180 Rome nodes. The total cost of this suite of simulations was ≈ 34 million core hours.	223

Abstract

Learning-Augmented and Structure-Preserving Methods for Conservation Law

Solvers

by

Christopher DeGrendele

In this work, we develop numerical methods for conservation laws that explore statistical, structure-preserving, and machine-learning-based approaches, each built on top of traditional numerical solvers. First, we develop a general Gaussian-process-based “recipe” for constructing high-order linear operators such as interpolation, reconstruction, and derivative approximations. Building on this recipe, we derive a kernel-agnostic convergence theory for GP-based operators that interprets them as generalized finite-difference schemes, defines an effective order-of-accuracy proxy that captures non-ideal truncation-error structure, and uses this metric to select stencil geometries and kernel hyperparameters analytically. We then introduce a new second-order kernel, Discontinuous Arcsin (DAS), that is stationary and prevents oscillations. DAS is integrated into a shock-capturing framework called the Multidimensional Optimal Order Detection (MOOD) method and shows an increase in efficiency by admitting less first order cascades. Next, we address the long-standing problem of spurious pressure oscillations in compressible multi-component and real-fluid simulations by introducing a fully conservative pressure-equilibrium-preserving scheme and a high-order fully conservative approximate variant that apply to arbitrary equations of state. Unlike existing approaches, these methods avoid non-conservative updates or EOS-specific constructions,

and on smooth interface advection tests with ideal-gas, stiffened-gas, and van der Waals fluids they reduce spurious pressure oscillations by orders of magnitude relative to current schemes. We then propose a hybrid numerical–machine learning framework for mixed hyperbolic–parabolic systems in which only the diffusive contribution is learned while the hyperbolic fluxes are advanced with standard shock-capturing methods, enabling timesteps at a hyperbolic CFL. Within this framework, we compare several neural architectures and loss designs on viscous Burgers tests and on the one-dimensional Euler equations with heat conduction, showing that U-shaped neural operators combined with multi-step and TVD-style regularization improve long-time stability and spectral behavior, and we analyze the resulting coupled schemes via eigenvalue-based stability diagnostics. Finally, we apply high-order, shock-capturing finite-difference methods within NASA’s Launch Ascent and Vehicle Aerodynamics (LAVA) framework to quantify acoustic and pressure loads on the Artemis Mobile Launcher, including multiphase simulations of water-suppression systems and comparisons to flight data that inform hardware design for future missions. Collectively, this work offers a set of targeted advances in kernel-based numerical operators, conservative schemes and learning-augmented solvers each aimed at improving accuracy, stability, or efficiency in complex multiphysics flow simulation.

Dedicated to GAC,

For teaching me to wonder.

Acknowledgments

What an honor it has been to pursue my PhD with so many wonderful people around me. I can't help but laugh at the irony of this being a single-author document, considering how many incredible people have made it possible. I can't thank you all enough.

First of course is Dr. Dongwook Lee, my advisor. Thank you for the freedom to pursue my own path as a scientist. Admittedly, it was a windy one - but I could always count on you. I'm extremely grateful for your guidance, flexibility, and mentorship (and your patience).

Thank you to Dr. Nic Brummell and Dr. Ashesh Chattopadhyay for agreeing to serve on my committee and your feedback. Ashesh, thank you for your guidance on the machine learning aspects of this work. It was a great pleasure to work with you.

I was so extremely lucky to spend the majority of my PhD at NASA where I worked on the Launch Ascent Vehicle Aerodynamics (LAVA) Team. Thank you Dr. Michael Barad and Dr. Francois Cadieux, for being fantastic supervisors, mentors, and colleagues. Giving me the freedom and flexibility to work on exciting projects while pursuing my doctorate was a great gift. A special thank you to Dr. Nguyen Ly who agree to take over all of my work (before he was forced to due to the government shut down) so I had time to finish. It goes without saying the PEP method would not be possible without your guidance, but I truly enjoyed working on it together. Thank you for being a great friend and collaborator. Thank you to Scott Neuhoff for teaching me how to run launchpad simulations. Mike, Francois, Nguyen, Scott, Emre, James and I made up the

“LAVA Launchpad Team”. Most of this work was never published or showcased out of NASA, but it remains some of the coolest and most exciting projects I’ve ever worked on. It was truly the highlight of my PhD. Thanks to Dr. Bram Rodgers for the welcome distractions, guidance, answering my linear algebra questions, and friendship. To Jared Duensing, thank you for leading the LAVA team and supporting my PhD pursuit. The group would not be what it is without your hard work. I feel lucky to call you my boss. My appreciation extends to all current and former LAVA team members. Many days I question how I accidentally ended up as a member of such a brilliant team.

There are a couple people I have to thank that started me on this journey. Dr. Alan Calder, Dr. Matthew Reuter, and Dr. Donald Willcox. Thank you for giving me the ability to grow and teaching me in the early days what it meant to be a scientist. It is because of you three that I continued on this path.

I was also fortunate enough to spend three summers at Berkeley Lab in the CCSE group. Thank you to Dr. Donald Willcox, Dr. Ann Almgren, Dr. Andrew Myers, and Dr. Andy Nonaka for welcoming me into the group. Those summers were formative for me, both technically and personally. I’m extremely grateful for your mentorship and the ability to work with such an amazing team.

Thank you to my friends. Eric Slota (for following me from New York, and for the legendary tapioca 220D), Andrew Tuma (for being the only person who actually tried to read this and of course supporting me along the way), Harrison Hoffman (for being my role model), Joe Soto (for the laughs and friendship), Kiran Eiden (for being such a wonderful friend, and for going through the PhD trenches alongside me), Angelo San Pablo (for following me to Santa Cruz, and for always being someone I can count on),

Eric Schieferstein (for the friendship and all the trips together), and Philippe Sabbagh (for being the coolest person I know, and for teaching me how to fly). Thank you to HAAC LLC for the strategic guidance and ongoing support. (WWYB)

Lastly, I want to thank my family. Dr. Mom, Dad, and Caroline, your unwavering support has meant everything to me. Thank you for cheering me on and always being there for me. Ellen, if there were one person I could not have done this without, it would be you. Thank you for your love, patience, and belief in me on the days I had none to spare. You carried more of this journey than anyone will ever see, and I'm endlessly grateful to get to share my life with you. Camp and Nova, thank you for all the cuddles and for getting me outside after long days of writing. I love you all.

As each big chapter in my life ends, I always realize that it's not the accomplishments I remember most, but the people who were there along the way.

Thank you for being there with me.

Support for this work was provided by NASA's Pathways program, NASA's Transformational Tools and Technologies (TTT) project, NASA's Exploration Ground Systems (EGS) project under the Exploration Systems Development Mission Directorate (ESDMD), the National Science Foundation under grants AST-1908834 and AST-2307684, the University of California Dissertation Year Fellowship and Baskin School of Engineering (BSOE) Dean's Fellowship. Computer time has been provided by NASA's Advanced Supercomputing (NAS) facility at NASA Ames Research Center and the Lux supercomputer at UC Santa Cruz, funded by NSF MRI grant AST-1828315.

Chapter 1

Introduction

1.1 Conservation Laws

Accurately modeling the physical world is a fascinating challenge requiring applied mathematics, physics, and computer science. This is normally done via partial differential equations (PDEs) where deriving equations is often far simpler than solving them. The famous “Navier-Stokes Millennium Problem,” [51] with its million-dollar reward, highlights the complexities involved in finding solutions to fluid dynamics equations. Numerical simulation serves as a practical alternative to elusive analytical solutions, with significant implications across multiple disciplines. These methods discretize the equations and the domain, making the problem tractable for computational algorithms. These methods are widely used in fields ranging from aerospace engineering, climate modeling, and biomedical engineering, where precise predictions of physical phenomena are crucial. Fluid dynamics, modeled by the Euler and Navier-Stokes equations, is based on the principles of conservation laws. These laws ensure that quantities

like mass, momentum, and energy are conserved as fluids move and interact. By applying numerical methods to these conservation laws, we can simulate and predict the complex behavior of fluids in various scenarios.

1.1.1 Hyperbolic, parabolic, and mixed conservation laws

We are interested in systems of conservation laws that may include both hyperbolic (advective) and parabolic (diffusive) operators. In conservative form, we write

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \nabla \cdot \mathbf{G}(\mathbf{U}, \nabla \mathbf{U}) + \mathbf{S}, \quad (1.1)$$

where $\mathbf{U}$ denotes the conserved variables, $\mathbf{F}$ is the hyperbolic (advective) flux, $\mathbf{G}$ is the parabolic (diffusive) flux, and $\mathbf{S}$ represents source terms. This formulation ensures that any change in $\mathbf{U}$ over time inside a control volume is balanced by the net flux across the boundary together with internal sources and sinks.

When $\mathbf{G} = 0$, we obtain a purely hyperbolic conservation law,

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \mathbf{S}, \quad (1.2)$$

whose wave structure is governed by the flux Jacobian

$$\mathbf{A}(\mathbf{U}) \equiv \frac{\partial \mathbf{F}}{\partial \mathbf{U}}. \quad (1.3)$$

Classical examples include the compressible Euler equations and ideal magnetohydrodynamics.

When $\mathbf{F} = 0$, the system reduces to a purely parabolic equation,

$$\frac{\partial \mathbf{U}}{\partial t} = \nabla \cdot \mathbf{G}(\mathbf{U}, \nabla \mathbf{U}) + \mathbf{S}, \quad (1.4)$$

with the canonical example given by the one-dimensional heat equation $u_t = \kappa u_{xx}$. Here the spatial operator is smoothing and information spreads in space at the continuous level, and purely parabolic problems are often treated with implicit or semi-implicit time integration to avoid the severe timestep restriction associated with diffusion.

Many physically relevant models combine both advective and diffusive processes and therefore fall into the mixed hyperbolic–parabolic class (1.1). A prototypical example is the compressible Navier–Stokes system, which augments the hyperbolic Euler fluxes with viscous and thermal diffusion. In such systems, the hyperbolic flux $\mathbf{F}$ determines the characteristic wave propagation and is naturally handled by conservative, upwind, or Riemann-solver-based schemes, while the parabolic flux $\mathbf{G}$ governs diffusion and introduces a stiff parabolic timescale. It is therefore natural to regard the hyperbolic and parabolic contributions as distinct operators that are discretized using complementary numerical strategies while retaining the conservative formulation and characteristic wave structure of the advective part [144].

1.1.2 Compressible Euler equations

As a concrete example of a purely hyperbolic system in the above framework, consider the compressible Euler equations,

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ E \end{pmatrix}, \quad \mathbf{F}_j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p \delta_{ij} \\ u_j(E + p) \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad (1.5)$$

where ρ is the mass density, representing the mass of the fluid per unit volume; u_i is the fluid velocity in the i th direction; E is the total energy; p is the pressure; and δ_{ij} is the

Kronecker delta, which is 1 when $i = j$ and 0 otherwise, used to handle the interaction of velocities in different directions in tensor notation. The inviscid Euler system fits into Eq. (1.2) with $\mathbf{G} \equiv 0$, while the compressible Navier–Stokes equations augment these fluxes with viscous and thermal diffusion terms in $\mathbf{G}$ and thus fall under the mixed hyperbolic–parabolic form (1.1).

1.1.3 Hyperbolic conservation laws and discretization

We are therefore interested in the solution of systems of hyperbolic conservation laws of the form

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \mathbf{S}, \quad (1.6)$$

where $\mathbf{U}$ is a vector of n conserved variables, $\mathbf{F}$ is the flux vector, and $\mathbf{S}$ is a vector of source terms.

We can rewrite Eq. (1.6) via the chain rule as

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A} \cdot \nabla \mathbf{U} = \mathbf{S} \quad \text{where} \quad \mathbf{A} \equiv \frac{\partial \mathbf{F}}{\partial \mathbf{U}}, \quad (1.7)$$

where $\mathbf{A}$ is known as the flux Jacobian. In general, the system (1.6) is hyperbolic if the Jacobian of the flux can be written as

$$\mathbf{A} = \frac{\partial \mathbf{F}(\mathbf{U})}{\partial \mathbf{U}} = \mathbf{R} \mathbf{\Lambda} \mathbf{R}^{-1}, \quad (1.8)$$

where $\mathbf{\Lambda}$ is a diagonal matrix with real eigenvalues and $\mathbf{R} = [\mathbf{r}_1 | \mathbf{r}_2 | \dots | \mathbf{r}_m]$ is a matrix that contains the right eigenvectors $\mathbf{r}_k$. Substituting Eq. (1.8) into Eq. (1.7) yields

$$\mathbf{R}^{-1} \frac{\partial \mathbf{U}}{\partial t} + \mathbf{\Lambda} \mathbf{R}^{-1} \nabla \mathbf{U} = \mathbf{S}, \quad (1.9)$$

and defining $\mathbf{W} \equiv \mathbf{R}^{-1}\mathbf{U}$ as the characteristic variables gives

$$\frac{\partial \mathbf{W}}{\partial t} + \mathbf{A} \cdot \nabla \mathbf{W} = \mathbf{S}. \quad (1.10)$$

Since $\mathbf{A}$ is an $n \times n$ diagonal matrix, we now have a set of n linearly independent hyperbolic conservation laws to solve. While the hyperbolic conservation law can be solved in either form (Eq. 1.6 or Eq. 1.10), we typically opt for the conservation form as it is more computationally efficient and easier to implement. The characteristic form is more challenging to solve because it requires transforming the conservative variables into characteristic variables using the eigenvector matrices. This process can be computationally intensive, compared to just using the conservative variables, and potentially numerically unstable due to the possibility of $\mathbf{A}$ being rank-deficient. The characteristic form is nevertheless useful conceptually and in localized constructions, such as Riemann solvers and limiters, for explicitly handling wave propagation [144].

The primary objective in the numerical solution of hyperbolic conservation laws is to accurately compute the evolution of the conserved variables $\mathbf{U}$ over time. This is generally posed as an initial value problem (IVP) where one specifies the initial conditions of $\mathbf{U}$ as well as the simulation domain and boundary conditions. For one

spatial dimension this can be represented as

Given: Initial conditions $\mathbf{U}(x, t = 0) = \mathbf{U}_0(x)$,

Simulation domain $\Omega = (x_{\text{lo}}, x_{\text{hi}}) \subset \mathbb{R}$,

Boundary conditions

$$\mathbf{U}(x_{\text{lo}}, t) = \mathbf{U}_{\text{lo}}(t), \quad t > 0,$$

$$\mathbf{U}(x_{\text{hi}}, t) = \mathbf{U}_{\text{hi}}(t), \quad t > 0,$$

Find: Solution $\mathbf{U}(x, t)$ for $x \in \Omega$, $t > 0$.

To numerically solve Eq. 1.6, we approximate the continuous physical domain with a discrete computational grid and calculate the change in the conserved quantities over time at each cell. First we discretize the domain Ω into N cells with grid spacing

$$\Delta x = \frac{x_{\text{hi}} - x_{\text{lo}}}{N},$$

and define grid points $x_i = x_{\text{lo}} + i\Delta x$ for $i = 0, 1, \dots, N$. There are two well-established ways to solve the above IVP, finite difference methods (FDM) and finite volume methods (FVM), visualized in Figure 1.1. In FDM the conservative variables are represented as pointwise values at the cell locations and in FVM the conserved quantities are integral averages over the cell volume. The FVM acts on the integral of conserved quantities over cells, which can be seen by taking the integral of Eq. 1.6 over a volumetric region Ω ,

$$\int_{\Omega} \left(\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) \right) d\Omega = \int_{\Omega} \mathbf{S} d\Omega, \quad (1.11)$$

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{U} d\Omega + \int_{\partial\Omega} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n} dA = \int_{\Omega} \mathbf{S} d\Omega, \quad (1.12)$$

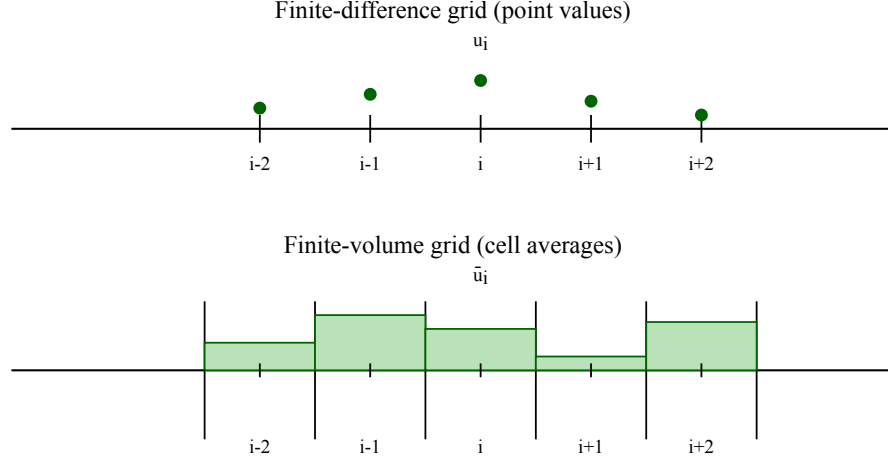


Figure 1.1: Top: finite difference grid representing pointwise values at cell location i . Bottom: finite volume grid representing cell-averaged values at cell location i .

where $\mathbf{n}$ is the outward unit normal vector. In Eq. 1.12 one can see that in order for a conserved quantity $\mathbf{U}$ to either increase or decrease in time, there must be some flux of that value that travels normal to the boundary of the cell.

In both FDM and FVM, the semi-discrete form of the hyperbolic conservation law can be written as

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \sum_{d=1}^D \frac{\hat{\mathbf{F}}_{i+\frac{1}{2},d}^n - \hat{\mathbf{F}}_{i-\frac{1}{2},d}^n}{\Delta x_d} = \mathbf{S}_i^n, \quad (1.13)$$

where $U_i^n = \mathbf{U}(x_i, t_n)$ represents the cell value at timestep n and spatial location i , and $\hat{\mathbf{F}}_{i\pm\frac{1}{2},d}^n$ denote the numerical fluxes at the cell interfaces in the d -th spatial direction.

The numerical fluxes are approximations of the analytical fluxes $\mathbf{F}(\mathbf{U})$ and are essential for ensuring stability and accuracy in the numerical solution. A key challenge is to define the numerical flux function $\hat{\mathbf{F}}$ that accurately captures the physical behavior of the flux across cell interfaces while ensuring that the numerical method remains stable

and convergent.

1.1.4 High-order fluxes and the order barrier near shocks

We now focus on the construction of fluxes for hyperbolic conservation laws. A natural starting point is to approximate the flux gradients using high-order central finite differences. In Eq. (1.6) we are interested in approximating $\nabla \cdot \mathbf{F}$. In one spatial dimension, a central finite difference approximation can be constructed at arbitrarily high accuracy by including more grid points in the stencil. For example, the second-, fourth-, and sixth-order central finite-difference formulas for a scalar flux F are

$$\begin{aligned} \left(\frac{dF}{dx}\right)_i^{(2)} &= \frac{\mathbf{F}_{i+1} - \mathbf{F}_{i-1}}{2\Delta x}, \\ \left(\frac{dF}{dx}\right)_i^{(4)} &= \frac{-\mathbf{F}_{i+2} + 8\mathbf{F}_{i+1} - 8\mathbf{F}_{i-1} + \mathbf{F}_{i-2}}{12\Delta x}, \\ \left(\frac{dF}{dx}\right)_i^{(6)} &= \frac{\mathbf{F}_{i+3} - 9\mathbf{F}_{i+2} + 45\mathbf{F}_{i+1} - 45\mathbf{F}_{i-1} + 9\mathbf{F}_{i-2} - \mathbf{F}_{i-3}}{60\Delta x}, \end{aligned} \tag{1.14}$$

where $\left(\frac{dF}{dx}\right)_i^{(n)}$ denotes the n th-order central approximation to the first derivative in x . For smooth solutions, such schemes are attractive: they are easy to implement, high-order accurate, and do not introduce excessive artificial damping of resolved modes.

However, for nonlinear hyperbolic conservation laws, the solution can develop shocks and contact discontinuities even from smooth initial data. In the presence of such discontinuities, purely central high-order schemes behave poorly: they do not provide sufficient dissipation to control spurious oscillations and typically exhibit Gibbs-type ringing around shocks. A successful discretization must therefore introduce some form of upwinding or numerical dissipation that is strong enough to stabilize discontinuities, while remaining as weak as possible in smooth regions to preserve high-order accuracy.

One simple way to incorporate dissipation into a central discretization is to augment the high-order central derivative with an artificial viscosity (AV) term,

$$\frac{\partial \hat{\mathbf{F}}}{\partial x} \approx \left(\frac{dF}{dx} \right)_i^{(n)} + \text{AV}_i, \quad (1.15)$$

where AV_i is a dissipative operator constructed, for example, from higher-order derivatives or from jump indicators based on neighboring cell values. Choosing this artificial viscosity is delicate: too much dissipation yields a method that is stable but overly diffusive and unable to resolve fine-scale structures; too little dissipation leads to non-linear instabilities and eventual blow-up. A large portion of the finite-difference shock-capturing literature is devoted to designing AV operators, sensors, and switches that activate dissipation primarily near shocks while remaining essentially inactive in smooth regions.

More generally, numerical fluxes for hyperbolic conservation laws can be broadly categorized into two classes [144]: upwind or Godunov-type fluxes, which use characteristic or Riemann-solver information to bias the stencil in the direction of wave propagation and thereby introduce controlled, physically motivated dissipation; and centered or non-upwind fluxes, which approximate flux gradients symmetrically and must be supplemented with artificial viscosity, filtering, or other regularization mechanisms to remain stable in the presence of shocks. Modern high-resolution methods often blend these ideas: they behave like high-order, nearly central schemes in smooth regions, but become effectively first-order and strongly dissipative near discontinuities through limiters, nonlinear switches, or adaptively added dissipation.

This behavior is not merely a design choice but is enforced by a fundamen-

tal accuracy barrier. For scalar conservation laws, Godunov’s theorem states that any monotone conservative scheme cannot be more than first-order accurate. A closely related result, often formulated in the TVD framework, is that any total-variation-diminishing (TVD) scheme is at most first-order accurate at local extrema and across discontinuities. In particular, any scheme that is both non-oscillatory (in the sense of being monotone or TVD) and conservative must reduce to first order in the neighborhood of shocks.

This explains the design philosophy of essentially all modern high-resolution methods: achieve high-order accuracy in smooth regions using wide, nearly central stencils, but allow the method to drop to first-order accuracy in the vicinity of shocks through nonlinear limiting, upwind biasing, or the activation of artificial viscosity. A significant amount of effort therefore goes into constructing numerical fluxes, limiters, and switches that make this transition as sharp as possible.

1.1.5 Time integration

Up until now, our discussion has focused on the spatial structure of conservation laws and on achieving accurate interface fluxes. However, accurate time integration is equally crucial for the stability and overall accuracy of numerical methods in CFD. Once a spatial discretization has been chosen, the semi-discrete system takes the form

$$\frac{d\mathbf{U}}{dt} = L(\mathbf{U}), \tag{1.16}$$

where L denotes the discrete spatial operator (including both hyperbolic and parabolic contributions). The choice of time-stepping scheme must respect the stability con-

straints imposed by both the hyperbolic and parabolic parts.

For purely hyperbolic systems, explicit time integrators are constrained by the usual hyperbolic Courant–Friedrichs–Lewy (CFL) condition,

$$\Delta t \leq C_{\text{hyp}} \frac{\Delta x}{\max_i |\lambda_i|}, \quad (1.17)$$

where $\{\lambda_i\}$ are the characteristic speeds associated with the flux Jacobian and C_{hyp} is a scheme-dependent constant, commonly denoted the CFL number, which represents the fraction of a grid spacing that the fastest-moving wave is allowed to travel in a single timestep.

For purely parabolic systems such as the heat equation, explicit time discretizations are subject to a much more restrictive parabolic timestep condition,

$$\Delta t \leq C_{\text{par}} \frac{\Delta x^2}{\nu_{\text{max}}}, \quad (1.18)$$

where ν_{max} is an effective diffusion coefficient and C_{par} is a scheme-dependent constant that plays an analogous role to the CFL number for diffusion. In this case, the restriction reflects the fact that explicit methods can only resolve diffusive propagation over a fraction of a grid cell per timestep.

For mixed hyperbolic–parabolic systems, the overall timestep must satisfy both constraints,

$$\Delta t \leq \min \left(C_{\text{hyp}} \frac{\Delta x}{\max_i |\lambda_i|}, C_{\text{par}} \frac{\Delta x^2}{\nu_{\text{max}}} \right), \quad (1.19)$$

which motivates the use of implicit or semi-implicit treatments of the parabolic terms in many practical schemes.

Runge–Kutta methods provide a widely used family of time integrators for such semi-discrete systems. The first Runge–Kutta method (RK1) was introduced in [132]

and later extended to higher-order schemes, including the classical fourth-order Runge–Kutta method (RK4) [79]. In the context of high-resolution schemes for conservation laws, particular attention is given to strong stability-preserving (SSP) Runge–Kutta methods, which are designed to inherit nonlinear stability properties (such as monotonicity or total variation diminishing (TVD) behavior) from the forward Euler method under a given CFL restriction.

The Shu–Osher formulation represents a general framework for constructing SSP Runge–Kutta methods. These methods are designed to ensure that if the spatial discretization satisfies a strong stability condition, then the fully discrete method (spatial plus temporal discretization) will also satisfy this stability condition, possibly under a modified CFL bound. This attribute is vital, especially when dealing with shock waves or discontinuities, as it prevents the numerical solution from exhibiting non-physical oscillations or instabilities.

In Shu–Osher form, an s -stage SSP Runge–Kutta scheme can be written as

$$u^{(i)} = \sum_{k=0}^{i-1} \left(\alpha_{ik} u^{(k)} + \Delta t \beta_{ik} L(u^{(k)}) \right), \quad i = 1, \dots, s, \quad (1.20)$$

where $u^{(0)} = u^n$ is the solution at time level t^n , the $u^{(i)}$ are intermediate stage values, L represents the spatial operator, Δt is the timestep, and the coefficients α_{ik} and β_{ik} define the particular Shu–Osher scheme. The final update to obtain u^{n+1} from u^n typically involves a convex combination of these intermediate stages, ensuring the SSP property whenever the coefficients satisfy appropriate nonnegativity conditions and the timestep respects the underlying CFL constraint.

By appropriately choosing the coefficients α_{ik} and β_{ik} , one can construct meth-

ods of different orders while ensuring the desired stability characteristics. A widely used example is the three-stage third-order SSP Runge–Kutta method (SSPRK(3,3)) [58], which can be written as

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n), \\ u^{(2)} &= \frac{3}{4} u^n + \frac{1}{4} u^{(1)} + \frac{1}{4} \Delta t L(u^{(1)}), \\ u^{n+1} &= \frac{1}{3} u^n + \frac{2}{3} u^{(2)} + \frac{2}{3} \Delta t L(u^{(2)}). \end{aligned} \tag{1.21}$$

This scheme achieves third-order accuracy in time while preserving the strong stability properties of the underlying forward Euler step under a suitably restricted timestep. In the following sections, we will combine such SSP time integrators with the high-order spatial flux constructions for conservation law based systems.

1.2 Machine Learning and Statistical Surrogates

The numerical methods discussed above rely on an explicit, hand-crafted discretization of the governing conservation laws. In each case, the evolution operator is derived from first principles. We specify the flux function $\mathbf{F}$, choose a spatial discretization, and then advance in time with a temporal integrator. Machine learning (ML) offers a complementary perspective. Instead of explicitly deriving all components of the numerical scheme, we approximate certain maps or operators directly from data. In the context of conservation laws, these data may come from high-fidelity simulations, reduced-order models, or experiments. The goal is not to replace these methods in this work, but to blend data-driven approximations with numerical methods.

In this dissertation we consider two closely related families of data-driven sur-

rogates. The first uses neural networks as flexible, parametric function approximators that learn deterministic input–output maps. The second uses statistical learning methods, in particular Gaussian processes (GPs), that represent a distribution over functions and therefore return not only a prediction but also a principled uncertainty estimate. Both viewpoints can be used to approximate solution maps, time-stepping maps, or individual components of a numerical method but differ in how prior assumptions are encoded. Neural networks typically encode inductive bias through architecture and training. GPs encode it directly through the choice of kernel and prior.

1.2.1 Basic concepts in supervised learning

At its core, supervised machine learning considers a family of parametric functions

$$f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y},$$

where $\mathcal{X}$ is an input space, $\mathcal{Y}$ is an output space, and θ denotes a collection of trainable parameters, for example the weights and biases of a neural network. Given a dataset of input–output pairs

$$\mathcal{D} = \{(x^{(m)}, y^{(m)})\}_{m=1}^M, \tag{1.22}$$

the learning problem is posed as an optimization,

$$\theta^{\star} = \arg \min_{\theta} \mathcal{L}(\theta) = \arg \min_{\theta} \frac{1}{M} \sum_{m=1}^M \ell(f_{\theta}(x^{(m)}), y^{(m)}), \tag{1.23}$$

where ℓ is a pointwise loss function, typically a mean-squared error for regression tasks. The optimization is usually carried out with some gradient-based optimization method.

Neural networks are a flexible class of function approximators used as f_θ . In their simplest form, fully connected feed-forward networks map an input vector through a sequence of affine transformations and elementwise nonlinearities,

$$\mathbf{z}^{(0)} = x, \quad \mathbf{z}^{(\ell+1)} = \sigma(W^{(\ell)}\mathbf{z}^{(\ell)} + \mathbf{b}^{(\ell)}), \quad (1.24)$$

for layers $\ell = 0, \dots, L - 1$, with trainable matrices $W^{(\ell)}$ and biases $\mathbf{b}^{(\ell)}$, and with σ a nonlinear activation function. Convolutional neural networks extend this idea to grid-structured data by exploiting locality and translation invariance.

From the perspective of numerical PDEs, such networks can be seen as highly parameterized surrogates that learn mappings between states, parameters, and outputs of interest. For example, a NN may approximate the solution map $\mathbf{U}_0 \mapsto \mathbf{U}(\cdot, t)$ for fixed t , a time-stepping map $\mathbf{U}^n \mapsto \mathbf{U}^{n+1}$, or a subcomponent of a numerical method such as a flux, closure term, or diffusive operator. In some settings, the learned model can also act as a hyperparameter or switch within a numerical method [84].

Purely data-driven models that ignore the underlying physics can generalize poorly outside the training distribution and may violate fundamental constraints such as conservation or stability. This motivates physics-informed approaches, in which prior knowledge of the governing equations, conservation laws, or symmetries is built into the learning procedure [147]. In simple settings, such as physics-informed neural networks [120](PINNs), this is often done by augmenting the loss function with penalty terms that measure violations of the PDE residual, boundary and initial conditions, or other constraints. More generally, one can design the loss to enforce any principle given the mathematical form, for example conservation, stability, or invariance. Equally impor-

tant, the network architecture itself encodes a priori assumptions about the solution, such as smoothness, locality, or symmetries, so the choice of model class is already a form of physics-informed bias.

1.2.2 From pointwise models to operator learning

Standard neural networks approximate finite-dimensional maps, for example from a vector of grid values to another vector of grid values. In many PDE applications, however, the underlying object of interest is not just a finite-dimensional map, but a solution operator. Given an input function, such as an initial condition, forcing, or coefficient field, the PDE defines an operator that maps this function to another function,

$$\mathcal{G} : u_0(x) \mapsto u(x, t_{\text{final}}), \quad (1.25)$$

where u_0 is the initial condition function. Operator learning aims to parameterize $\mathcal{G}$ directly in a way that is robust to changes in resolution and domain discretization.

Neural operators are a class of architectures designed for this purpose. Given function-valued inputs and outputs, they approximate the underlying operator via a sequence of liftings, integral transformations, and pointwise nonlinearities. By composing layers and projecting back to the physical space, one obtains an approximation of the PDE solution operator. The goal is not just to interpolate between fixed-resolution images, but to learn a map between function spaces that can be evaluated on different meshes.

1.2.3 Fourier Neural Operator (FNO)

The Fourier neural operator (FNO) [90] is a neural network architecture that works in Fourier space rather than directly in physical space. Instead of learning a kernel $k_\theta(x, y)$ as a function of position, FNO learns how different Fourier modes interact. In each layer, the input field $v(x)$ is first transformed to Fourier space to obtain coefficients $\hat{v}(k)$, a learned linear transformation is applied to a chosen set of modes, and the result is transformed back to physical space. In simple terms, FNO repeatedly performs a Fourier transform, a learnable modification of the modes, and an inverse transform. After the inverse transform, the output is combined with a pointwise linear term to form the next representation.

By keeping only a finite number of Fourier modes, FNO concentrates its capacity on the most energetic and smooth parts of the solution, while the Fourier basis still provides global coupling across the domain. This makes FNO particularly appealing for PDEs, where solutions often have large-scale smooth structure with additional fine-scale features.

1.2.4 UNO and related architectures

The U-Net neural operator (UNO) [119] adapts the U-Net architecture [131], originally developed for image processing, to the setting of operator learning. Whereas FNO emphasizes global interactions through Fourier modes, UNO remains in physical space and relies on convolutional layers arranged in an encoder–decoder structure with skip connections. The basic idea is to start from the input field on the original grid, gradually compress it to coarser representations where large-scale patterns are easier to

capture, and then reconstruct it back to the original resolution while reintroducing fine-scale information through the skip connections. In practice, UNO processes the data at multiple scales simultaneously, using convolutional filters to capture local patterns and the encoder–decoder pathway to capture global structure.

Several other neural operator architectures follow similar principles. Deep operator networks, graph-based neural operators, and hybrid spectral–physical models all aim to learn mappings between function spaces that are flexible enough to handle nonlinear and multiscale behavior, while also being able to work across different resolutions and, when possible, respect known structure such as symmetries, invariants, or conservation laws.

1.2.5 Gaussian processes as statistical surrogates

Neural networks are typically used as deterministic surrogates. For fixed parameters θ , the prediction $f_\theta(x)$ is a single best estimate obtained by minimizing a training loss. Gaussian processes provide a complementary, explicitly probabilistic viewpoint. Rather than selecting a particular parametric form and fitting its weights, a GP defines a probability distribution over admissible functions that can fit the given data, so that predictions naturally come with an associated uncertainty.

A GP makes predictions of agnostic function value estimations by defining a probabilistic distribution over all admissible functions that can fit the given data. Constructing a collection of random variables is construed as defining a GP, where a finite number of which follow a joint multivariate Gaussian distribution. We say the

function f is distributed as a GP and write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')) , \quad (1.26)$$

where $\mathbf{x}$ and $\mathbf{x}'$ are two points (not necessarily distinct) in $\mathbb{R}^D$. That any agnostic function $f(\mathbf{x})$ is in GP according to Eq. (2.1) implies that the observation of the function evaluation of f at a new point $\mathbf{x}_* \in \mathbb{R}^D$ (where the true value $f(\mathbf{x}_*)$ has not been observed) follows a joint multivariate Gaussian distribution, allowing us to make a probabilistically meaningful estimation on $f(\mathbf{x}_*)$ with a predicted uncertainty.

In practice, specifying a GP amounts to choosing a mean function and a covariance kernel that encode prior beliefs about the smoothness, scale, or structure of the underlying function. Once these are fixed, GP regression proceeds by conditioning the joint Gaussian prior on observed data, yielding a closed-form expression for both the posterior mean (the best estimate of the function at new points) and the posterior covariance (the associated uncertainty). This provides a flexible, nonparametric framework in which predictions adapt to the data while still retaining an analytically tractable measure of confidence. In this work we will look at utilizing GPs to replace polynomial methods in high-order reconstruction of fluxes in conservation laws.

Chapter 2

Gaussian Process Based Numerical Methods

2.1 Introduction

Many physical phenomena in science and engineering applications pose numerical representations to function approximations that involve interpolations, derivatives, and integrals. Popular numerical methods for interpolation include polynomial interpolations, spline and Hermite interpolations, and rational function interpolations. For numerical derivatives, finite differences are perhaps the most popular method of all, while other alternatives are spectral methods, spline differentiation, and least square differentiation, to name a few. For integrals, there exist many quadrature rules to numerically evaluate the values of integrated functions. All of the methods mentioned can be broadly classified as parametric as they formulate function approximations by assuming a fixed functional form (e.g., polynomial vs. trigonometric functions) and con-

figure relevant function parameters (e.g., polynomial coefficients, polynomial degrees). The primary aim of this chapter is to present various types of function approximations based on the nonparametric Gaussian process (GP) modeling approach, where no assumption is made for a specific functional form but rather the function approximation is entirely determined by observed data.

Since its first introduction in [152], GP has been employed as a data-driven tool to study data-dominant problems in various applications. The scientific interests in GP regression brought broader attention in the subsequent work [123] by the same authors, where supervised machine learning using GP regression was presented. Prior supervised machine learning models were mostly parametric in nature, which lacked the ability to generalize to complex data sets and often faced data overfitting issues [122]. Neural networks are a subset of this class of approaches and thus inherited many of these problems. By contrast, GP is a nonparametric, kernel-based Bayesian approach to data regression making data-driven predictions. By being nonparametric, GP regression operates directly on defining a probability distribution over all admissible functions that fit the data instead of building around the probability distribution of parameters of any specific functional form.

The GP data regression and predictions are formulated by specifying a prior (on a target function space, e.g., smooth vs. discontinuous, periodic, etc.) and calculating the associated updated posterior distribution inferred by the conditional distribution of an unobserved new data point given the observed data points. This gives the ability to compute the predictive posterior distribution on a set of new data points. The underlying principle in GP regression is to establish the relationship between input and

output data points by correlating them and letting them determine the inferred data complexity in the sense of Bayesian inference [123].

Practitioners usually adopt one of two viewpoints to understand GPs: the Bayesian and function-space. Under the Bayesian viewpoint, estimations of function values are linear [64] in the constructed function space, referred to as the weight-space view that centers around probability distributions over linear weights of the sampled input data. The function-space view directly defines a distribution over functions, which ultimately arrives at the same results as with the weight-space approach [123, 135]. The equivalence of the function-space and weight-space approaches is justified by realizing that defining a distribution of functions is identical to defining each set of distributions of weights, where a collection of sets of weights comprises a collection of functions. The function-space view is what we adopt by default in this paper.

The interest in GP has grown over decades rapidly as a powerful tool for modeling correlated datasets in a wide range of application areas, including machine learning [123], data science [92, 42], deep neural networks [100, 56], time-series modeling [130, 52, 40, 25], robotics and control systems [46, 78], computer vision [125], medical applications [158], financial modeling and price forecasting [61, 156], to name a few. Tutorial articles [135, 153] have also been reported to provide the community with practical introductions of GP regression as a powerful tool for understanding agnostic function approximations.

2.1.1 Previous Studies of GP Regression in Shock-capturing Schemes

Apart from the above applications, GP regression has been recently extended to a class of high-order methods in finite difference (FD) and finite volume (FV) formalisms. In [83], GP’s updated posterior mean function was first introduced as a high-order method for a one-dimensional system of Euler’s equations. It demonstrated, as a proof-of-concept, that the new GP method can deliver a variable odd-order convergence rate from the 3rd to the 11th accuracy on a smooth Gaussian profile advection. This preliminary study also considered the GP method on the 1D Shu-Osher shock-tube problem [137] where the shock strength is mild. The GP shock-capturing strategy made a leap forward in the subsequent study reported in [127]. This work established a foundational ground where GP’s updated posterior mean function as a high-order reconstruction method was integrated into WENO’s shock-capturing framework [69]. The GP solution was tested against a suite of 1D shock-tube tests in hydrodynamics and magnetohydrodynamics. An extension to multiple spatial dimensions, dubbed GP-WENO, was first studied in the finite difference method (FDM) framework in [128], featuring the variable odd-order accuracy on smooth flows and stable shock-capturing capability on discontinuous flows.

The mathematical ideas of the GP-WENO method have been formulated into a more general kernel-based FV reconstruction method in the finite volume method (FVM) framework in [102, 101]. Unlike the use of symmetric positive definite kernels in the previous GP studies [83, 127, 128] that explored the GP covariance kernel functions, the work in [102, 101] utilized non-symmetric kernel functions (that are no longer GP

covariance kernels) and achieved the numerical stability near shocks using the adaptive-order WENO approach [12, 127]. This kernel-based FVM with WENO (or KFVM-WENO) is shown to be at least sixth-order accurate on the smooth nonlinear isentropic vortex problem [136, 139, 128] and robust on a selectively chosen set of benchmark problems, including a three-dimensional compressible isothermal turbulence problem and the Taylor-Green vortex problem. Interested readers are encouraged to consult more details on the setups and references in [101].

A new effort to restructure the shock-capturing paradigm has also been considered in GP. The majority of the modern shock-capturing algorithms take the so-called a priori approach, implying that the shock-detection mechanism is to be always engaged for non-oscillatory fluid data approximations. All of the aforementioned schemes in the previous paragraph belong to this class. On the other hand, an alternative method called a posteriori approach maintains non-oscillatory solutions near shocks by post-processing its solutions, where the solutions are cascaded to non-oscillatory, positivity-preserving lower-order solutions if unphysical oscillations are detected in shock regions. This order-cascading strategy is referred to as the MOOD (multidimensional optimal order detection) method and was originally proposed and studied in [36, 48, 49, 47, 22] for polynomial-based methods. In [82, 20], the high-order GP method was successfully integrated into the MOOD architecture (replacing the polynomial counterparts). It showed that the new GP-MOOD scheme gains significant computational efficiency by removing the need for expensive nonlinear shock-detecting switches (e.g., slope limiters, smoothness indicators) while delivering high-order solutions ranging the third, fifth, and seventh-order accuracy in two-dimensional finite volume simulations. Moreover, the GP

scheme has been extended to a third-order accurate prolongation algorithm in the finite volume adaptive mesh refinement simulations [124].

2.1.2 Extending GP’s Data Conversion Property to Interpolations and Linear Operators

One aspect that allows GP regression great flexibility, particularly in high-order FV reconstruction in multiple spatial dimensions, is its ability to convert data to any specified type. As already investigated (at least partly) in the GP shock-capturing methods mentioned above, GP can convert different data types from one grid location to another in a simpler manner than in the existing approaches [103, 27]. For example, a standard FV polynomial reconstruction scheme would require taking volume-averaged values at cell centers as input to reconstruct pointwise Riemann states as output at cell interface locations in 1D. In multiple dimensions, a quadrature-based integration scheme would need to be used to convert these pointwise estimates to a face integral flux value. A popular way to obtain high-order accurate Riemann states in conventional polynomial-based multidimensional reconstruction schemes is to add transverse corrections for each spatial direction dimension-by-dimension. An example in 3D would include three separate reconstructions, cascading from cell averages in 3D to face averages, face averages to line averages, and finally, line averages to pointwise values and vice versa [103, 27]. Although this process is not particularly difficult, there are many steps that can be computationally compounded, and one may need to apply boundary conditions on intermediate quantities [101], which results in added data communication in parallel computing.

In GP, however, such data-type conversions can be achieved by applying the needed integral operations directly to the underlying pointwise GP covariance kernels, thanks to the fact that GP is invariant under linear operations [123]. This aspect is the centerpiece of the current study, where we apply GP approximation to a collection of linear operations, such as derivatives and integrations, which are frequently used in modern science and engineering applications.

2.2 Gaussian Process for Modeling Functions

In this section, we give a concise outline of the GP regression theory from the function-space perspective to prepare basic building blocks that are useful for the rest of the paper.

2.2.1 GP Prior and Posterior Predictions

A GP makes predictions of agnostic function value estimations by defining a probabilistic distribution over all admissible functions that can fit the given data. Constructing a collection of random variables is construed as defining a GP, where a finite number of which follow a joint multivariate Gaussian distribution. We say the function f is distributed as a GP and write:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')) , \quad (2.1)$$

where $\mathbf{x}$ and $\mathbf{x}'$ are two points (not necessarily distinct) in $\mathbb{R}^D$. That any agnostic function $f(\mathbf{x})$ is in GP according to Eq. (2.1) implies that the observation of the function

evaluation of f at a new point $\mathbf{x}_* \in \mathbb{R}^D$ (where the true value $f(\mathbf{x}_*)$ has not been observed) follows a joint multivariate Gaussian distribution, allowing us to make a probabilistically meaningful estimation on $f(\mathbf{x}_*)$ with a predicted uncertainty.

In a more specific term, assume that we have observed the function values of f at a finite set of points $\{\mathbf{x} : \mathbf{x} \in \mathbb{R}^D\}$ probabilistically (but no information is available for the function f itself) in terms of the prior GP distribution, where the GP prior is determined by the following two functions:

- a mean function $m(\mathbf{x}) = \bar{f}(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, which is the expected function value of f at the input $\mathbf{x} \in \mathbb{R}^D$, and
- a symmetric positive definite covariance GP kernel function

$K(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \bar{f}(\mathbf{x}))(f(\mathbf{x}') - \bar{f}(\mathbf{x}'))]$, which correlates the dependence between the function values at two input points (not necessarily distinct) $\mathbf{x}$ and $\mathbf{x}'$.

Let us denote the input points by $\mathbf{x}_i \in \mathbb{R}^D, i = 1, \dots, n$, and the $D \times n$ matrix by $\mathbf{X} = [\mathbf{x}_1 | \dots | \mathbf{x}_n]$ whose columns are $\mathbf{x}_i$. We also denote the corresponding (observed or sampled) input data vector by $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]^T$ and the resulting $n \times n$ covariance matrix by $[\mathbf{K}]_{ij} \equiv K(\mathbf{x}_i, \mathbf{x}_j)$. Given the observations $\mathbf{f}$ on $\mathbf{X}$, suppose we want to make a probabilistic prediction on $f_* = f(\mathbf{x}_*)$ at a new point $\mathbf{x}_*$. Then, GP regression furnishes a probabilistic prediction for the new $\mathbf{x}_*$ by drawing f_* from the posterior distribution $p(f_* | \mathbf{f}, \mathbf{X}, \mathbf{x}_*)$. Using Bayes' Theorem, the conditional property applied to the joint Gaussian prior distribution on $\mathbf{f}, \mathbf{X}$, and $\mathbf{x}_*$ yields the GP posterior distribution $p(f_* | \mathbf{f}, \mathbf{X}, \mathbf{x}_*)$ [123, 127] described by

- a newly *updated posterior mean function* $\tilde{f}_* \equiv \bar{f}(\mathbf{x}_*) + \mathbf{k}_*^T \mathbf{K}^{-1} \cdot (\mathbf{f} - \bar{\mathbf{f}})$, and
- a newly *updated posterior covariance* $U^2 \equiv k_{**} - \mathbf{k}_*^T \mathbf{K}^{-1} \cdot \mathbf{k}_*$,

where $[\mathbf{k}_*]_i = K(\mathbf{x}_*, \mathbf{x}_i)$. Following our previous studies, we regard the updated posterior mean function to be GP's high-order approximation to any agnostic function evaluations at any given point $\mathbf{x}_*$. Often, the prior mean function is set to zero, $\bar{f}(\mathbf{x}) = \bar{\mathbf{f}} = 0$, to lessen the associated computational footprints, simplifying the expression of the posterior mean function further to

$$\tilde{f}_* = \mathbf{z}_*^T \cdot \mathbf{f}, \quad \text{where } \mathbf{z}_*^T = \mathbf{k}_*^T \mathbf{K}^{-1}. \quad (2.2)$$

The grid- and kernel-dependent vector $\mathbf{z}_*^T = \mathbf{k}_*^T \mathbf{K}^{-1}$ is called the prediction vector [127, 128], independent of the data $\mathbf{f}$, and hence can be calculated, stored, and reused as long as the grid (or the kernel) remains unchanged. If we further restrict ourselves to a ‘stationary’ kernel on a uniform grid (see more discussions on stationary kernels in Section 3.3.1), only one of these vectors, $\mathbf{z}_*^T$, needs to be stored for the entire grid. For any given data, $\mathbf{f} \in \mathbb{R}^D$ (e.g., $2r_{gp} + 1$ density values $\mathbf{f} = [\rho_{i-r_{gp}}, \dots, \rho_i, \dots, \rho_{i+r_{gp}}]^T$ on a one-dimensional stencil of size $2r_{gp} + 1$ with $D = 2r_{gp} + 1$), the cost of calculating Eq. (2.2) is simply $\mathcal{O}(2D + 1)$ floating point operations during the rest of a simulation. Here, the integer-valued r_{gp} is called a GP stencil radius, which is a key factor that determines the order of accuracy of GP solutions. The posterior covariance conveys an uncertainty in the prediction given by the posterior mean. As such, it can serve as an indicator of how confident the GP prediction is in GP modeling applications. However, it is of limited interest to the purpose of the current paper and its role will be overlooked.

2.2.2 Extending GP's Posterior Prediction to Linear Operations

To further extend GP regression to broader applications, we first note that symmetric positive definite GP kernels are closed under additions, multiplications, vertical rescaling, convolutions, and linear operations [123, 20]. The underlying GP prediction models, such as Eq. (2.2), can be transformed into another by such operations while preserving the probabilistic GP predictions embedded in the underlying posterior distribution. For the purpose of this paper, we mainly focus on producing new kernels under various linear operators.

Before proceeding further, let us first review the invariance property of GP under linear operators, denoted by $\mathcal{L}$, over the function space GP has generated. It can be shown that

$$\mathcal{L}(f) \sim \mathcal{GP}\left(\mathcal{L}\left(m(\mathbf{x})\right), \mathcal{L}\left(K(\mathbf{x}, \mathbf{x}')\right)\right) \text{ if } f \sim \mathcal{GP}\left(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')\right), \quad (2.3)$$

where

$$\mathcal{L}\left(K(\mathbf{x}, \mathbf{x}')\right) = \mathcal{L}_{\mathbf{x}}\left(\mathcal{L}_{\mathbf{x}'}\left(K(\cdot, \mathbf{x}')\right)\right) = \mathcal{L}_{\mathbf{x}'}\left(\mathcal{L}_{\mathbf{x}}\left(K(\mathbf{x}, \cdot)\right)\right), \quad (2.4)$$

depending on whether $\mathcal{L}$ acts on either $\mathbf{x}$ or $\mathbf{x}'$, or both. The proof is straightforward and is in [44].

Using this linearly invariant property, we can generalize the basic GP prediction at $\mathbf{x}_*$ in Eq. (2.2) to various other applications. A general prediction form can be put into the following form,

$$p_* = \mathcal{L}_2\left(\mathbf{k}_*^T\right) \mathcal{L}_1\left(\mathbf{K}^{-1}\right) \cdot \mathbf{f}, \quad (2.5)$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are individual linear operators. Simplifying, we can write

$$p_* = \mathbf{t}_*^T \mathbf{C}^{-1} \cdot \mathbf{f}, \quad (2.6)$$

where $\mathbf{C} = \mathcal{L}_1(K(\mathbf{x}, \mathbf{x}'))$, $\mathbf{t}_* = \mathcal{L}_2(K(\mathbf{x}_*, \mathbf{x}'))$, $\mathcal{L}_1$ and $\mathcal{L}_2$ are linear operators applied to their respective kernel matrices, and p_* is our prediction quantity at $\mathbf{x} = \mathbf{x}_*$. A trivial case is when both $\mathcal{L}_1$ and $\mathcal{L}_2$ are the point-evaluation functional, e.g., $\mathcal{L}_1(g(\mathbf{x})) = \mathcal{L}_2(g(\mathbf{x})) = g(\mathbf{x})$ for any function g , in which case we recover the basic form in Eq. (2.2).

A non-trivial example is readily found in our previous studies [127, 20], where GP's posterior mean function has been extended to a high-order FV reconstruction method. In this case, the input $\mathbf{f} = [\bar{q}_1, \dots, \bar{q}_N]^T$ is a one-dimensional vector of length N flattened to contain cell volume-averaged quantities on each cell center at $\mathbf{x}_h$ (e.g., $\mathbf{x}_h = (x_{i_h}, y_{j_h})$ in 2D Cartesian),

$$\bar{q}_h = \frac{1}{\Delta\mathcal{V}(\mathbf{x}_h)} \int_{\mathcal{V}(\mathbf{x}_h)} q(\mathbf{x}) d\mathcal{V}(\mathbf{x}), \quad (2.7)$$

sampled from a local stencil in $D = 1, 2, 3$ spatial dimensions. The output p_* is a point-wise high-order reconstructed Riemann state value. Here, $\Delta\mathcal{V}(\mathbf{x}_h)$ is the cell volume of the h th cell, $\mathcal{V}(\mathbf{x}_h)$, e.g., $\mathcal{V}(\mathbf{x}_h) = I_h \times J_h = [x_{i_h-1/2}, x_{i_h+1/2}] \times [y_{j_h-1/2}, y_{j_h+1/2}]$ in a 2D Cartesian geometry. To account for the input-output relationship, the operator $\mathcal{L}_1$ for $K(\mathbf{x}, \mathbf{x}')$ is chosen as a cell volume integral operator over $\mathbf{x}$ and $\mathbf{x}'$,

$$\mathcal{L}_1(\cdot) = \frac{1}{\Delta\mathcal{V}(\mathbf{x}_h)\Delta\mathcal{V}(\mathbf{x}'_k)} \int_{\mathcal{V}(\mathbf{x}_h)} \int_{\mathcal{V}(\mathbf{x}'_k)} (\cdot) d\mathcal{V}(\mathbf{x}) d\mathcal{V}(\mathbf{x}'), \quad (2.8)$$

while $\mathcal{L}_2$ for $\mathbf{k}_* = K(\mathbf{x}_*, \mathbf{x}')$ with a fixed $\mathbf{x} = \mathbf{x}_*$ is a volume integral operator over $\mathbf{x}'$ only,

$$\mathcal{L}_2(\cdot) = \frac{1}{\Delta\mathcal{V}(\mathbf{x}'_k)} \int_{\mathcal{V}(\mathbf{x}'_k)} (\cdot) d\mathcal{V}(\mathbf{x}'). \quad (2.9)$$

Applying $\mathcal{L}_1$ and $\mathcal{L}_2$ to the corresponding kernels produces a GP reconstruction in FVM with the following integrated kernels, $\mathbf{C}$ and $\mathbf{t}_*$, each of which corresponds to their respective basic kernel, $\mathbf{K}$ and $\mathbf{k}_*$. In componentwise form, they are

$$[\mathbf{C}]_{hk} = \frac{1}{\Delta\mathcal{V}(\mathbf{x}_h)\Delta\mathcal{V}(\mathbf{x}'_k)} \int_{\mathcal{V}(\mathbf{x}_h)} \int_{\mathcal{V}(\mathbf{x}'_k)} K(\mathbf{x}, \mathbf{x}') d\mathcal{V}(\mathbf{x}) d\mathcal{V}(\mathbf{x}'), \quad (2.10)$$

$$[\mathbf{t}_*]_k = \frac{1}{\Delta\mathcal{V}(\mathbf{x}'_k)} \int_{\mathcal{V}(\mathbf{x}'_k)} K(\mathbf{x}_*, \mathbf{x}') d\mathcal{V}(\mathbf{x}'). \quad (2.11)$$

It is noteworthy to mention the input-output relationship here. The first operator $\mathcal{L}_1$ establishes the covariance relationship of the input data at $\mathbf{x}$ and $\mathbf{y}$, respecting a given data type (e.g., $\mathbf{C}$ for volume-averaged and $\mathbf{K}$ for pointwise values). The second operator $\mathcal{L}_2$ then takes on the computed covariance and channels the correlations to a scalar output in a form we want to obtain (e.g., pointwise Riemann states). This is the main mechanism for how GP can flexibly achieve different types of linear operations on various data types, combine them, and compute an anticipated quantity at $\mathbf{x}_*$, where the location $\mathbf{x}_*$ itself is arbitrary, to begin with.

2.3 Choosing a Kernel Function and GP Stencils

In this section, we briefly discuss our options for choosing a GP kernel function and GP local stencil configurations.

2.3.1 Squared Exponential (SE) Kernel

The kernel calculation of the GP modeling begins with choosing an appropriate GP kernel function, K . For this section, we select the squared exponential (SE)

covariance kernel,

$$K(\mathbf{x}, \mathbf{x}') = K_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \Sigma^2 \exp \left[-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2} \right] = \Sigma^2 \exp \left[-\frac{r^2}{2\ell^2} \right], \quad (2.12)$$

which is one of the most widely used GP kernel functions [123, 17]. We follow our previous approach to set $\Sigma^2 = 1$ for simplicity, which does not affect the posterior mean function. The second hyperparameter ℓ determines the correlation length scale the GP model prefers. It controls how heavily weighted nearby points are compared to those that are further away in space. There is some flexibility in choosing ℓ while an optimal choice is available by minimizing the negative log-likelihood function with respect to ℓ [123]. In general, a larger ℓ is preferable, but such a choice comes at the cost of making the $\mathbf{K}$ a stiffer matrix and harder to invert.

Our choice of the SE kernel is supported by three prime attractive properties it affords. First, the Gaussian function delivers a C^∞ prior function space, which guarantees that its data fitting accuracy by those sampled admissible C^∞ functions is convergent linearly with its stencil size. The built-in regularity also provides the SE function with analytic forms of derivatives and integrations, which makes SE much more accessible to various applications. Second, if we assume a constancy of the two hyperparameters, Σ^2 and ℓ , the kernel becomes a function of the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$ only, hence making it a stationary kernel. It is also data-independent as described in Section 2.2.1. These two points together greatly simplify the GP prediction in Eq. (2.2) to a simple dot product of $\mathcal{O}(2D + 1)$ since $\mathbf{z}_*^T$ can be pre-computed in an initial step. Third, the SE kernel facilitates dimensional factorization, i.e.,

$$\exp \left[-\frac{r^2}{2\ell^2} \right] = \prod_d \exp \left[-\frac{r_d^2}{2\ell^2} \right], \quad (2.13)$$

where the radial distance between $\mathbf{x}$ and $\mathbf{x}'$ is $r^2 = \sum_d r_d^2$ with r_d in each d dimension. This factorization property enables a seamless extension from 1D to a higher spatial dimension, making our GP prediction capability readily feasible in multidimensional cases.

Unfortunately, these desirable properties of SE could be severely compromised when $\mathbf{x} \rightarrow \mathbf{x}'$ or $\ell \rightarrow \infty$, which results in a poorly conditioned kernel matrix $\mathbf{K}$. In practice, the latter case is less of an issue as we choose ℓ a bounded value. Our typical choice for ℓ is either a constant that represents a characteristic length scale of a given profile or a scale distance proportional to the grid scale set by $\ell = M\Delta$ with $\Delta = \min_d \Delta_d$ and $6 \leq M \leq 12$. The former case with $\mathbf{x} \rightarrow \mathbf{x}'$ causes the singularity issue more frequently as this scenario occurs as one resolves the grid scales, $\Delta \rightarrow 0$. We have documented several different approaches that address the issue, most recently in [20] as well as in [127, 128, 124]. The simplest (and easiest) approach is adopted in the current study, which utilizes quadruple precision in compiling only a couple of routines that are relevant to calculate $\mathbf{z}_*^T$ once and for all. Unlike the usual doubt, the added computational cost of handling quadruple-precision is a bare minimum and does not impact the overall performance of GP at all.

It is worth noting the different types of GP kernels, particularly those that are non-smooth. Although SE performs well on smooth datasets, its posterior prediction is prone to underperform with oscillations when datasets contain discontinuities that are common in shock-dominant fluid dynamics simulations. There are better-suited non-smooth kernels such as the Matérn class, Wendland, rational quadratic functions, a neural network covariance kernel function, etc. [123, 140, 50] for discontinuous datasets.

On smooth data, though, these kernels exhibit limited usability because their solution accuracy, modeled by the latent functions with finite-order differentiability, is reduced and lower than that of SE, independent of the input data size $\mathbf{f}$. This is in contrast to the fact that the accuracy of SE increases linearly (or decreases) with increasing (or decreasing) size of the input data, which can be readily controlled by the GP stencil configuration. Later in Section 4.1, we introduce a new discontinuous kernel that is superior in fitting discontinuous data without oscillation while it converges at second-order on smooth data. We discuss GP stencils in the next section.

2.3.2 1D GP Stencils

We take a vector of size $(2r_{gp} + 1)$ for our 1D GP stencils, containing contiguous cells from $x_{i-r_{gp}}$ to $x_{i+r_{gp}}$ centered at x_i . With this stencil configuration, a native point-to-point GP interpolation that takes an input $\mathbf{f} = [q_{i-r_{gp}}, \dots, q_{i+r_{gp}}]^T$ for a function q with $q_j = q(x_j)$ to produce a function interpolation $q(x_*) = q_*$ with $x_* \in [x_{i-r_{gp}}, x_{i+r_{gp}}]$ becomes a $(2r_{gp} + 1)$ -th accurate interpolation scheme following the stencil size linearly. Our default choice of x_* in 1D is $x_{i \pm 1/2}$, the i -th cell interface locations, where various numerical methods (e.g., finite difference, finite volume, discontinuous Galerkin, etc.) calculate high-order numerical quantities such as Riemann states and numerical fluxes.

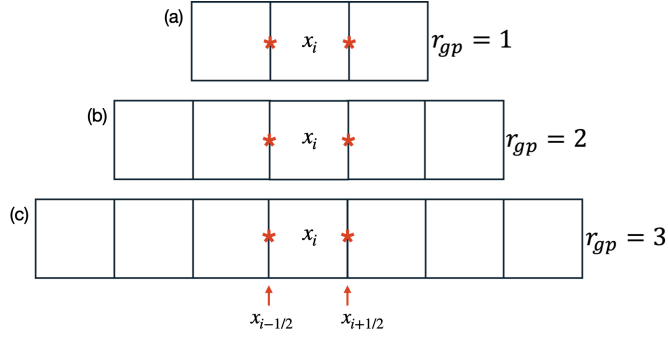


Figure 2.1: 1D GP stencil. The red “*” marks indicate the i -th cell interface locations we choose for x_* in 1D. The accuracy of the native GP point interpolations at x_* is $(2r_{gp} + 1)$, showing (a) the 3rd-order stencil, (b) the 5th-order stencil, and (c) the 7th-order stencil, where r_{gp} denotes the respective integer-valued GP stencil radius from the cell center x_i . A higher-order GP stencil can be easily configured with $r_{gp} > 3$.

2.3.3 2D GP Stencils

GP stencils can naturally extend to a higher spatial dimension. This is yet another feature that makes GP regression genuinely multidimensional and advantageous in handling multidimensional stencil data over the existing polynomial counterparts. For instance, in Fig. 2.2, we display four different GP stencil configurations in 2D with $r_{gp} = 2$ centered at $\mathbf{x}_m = (x_{im}, y_{jm})$, including (a) the cross stencil, (b) the diamond stencil, (c) the blocky-diamond stencil, and (d) the square stencil.

To decide on an optimal 2D stencil, we examine the numerical convergence rates for various GP prediction operations on each configuration and choose the most compact stencil configuration that meets the expected linear convergence rate, e.g., $(2r_{gp} + 1)$ for the baseline point-to-point interpolation. On the cross stencil in (a),

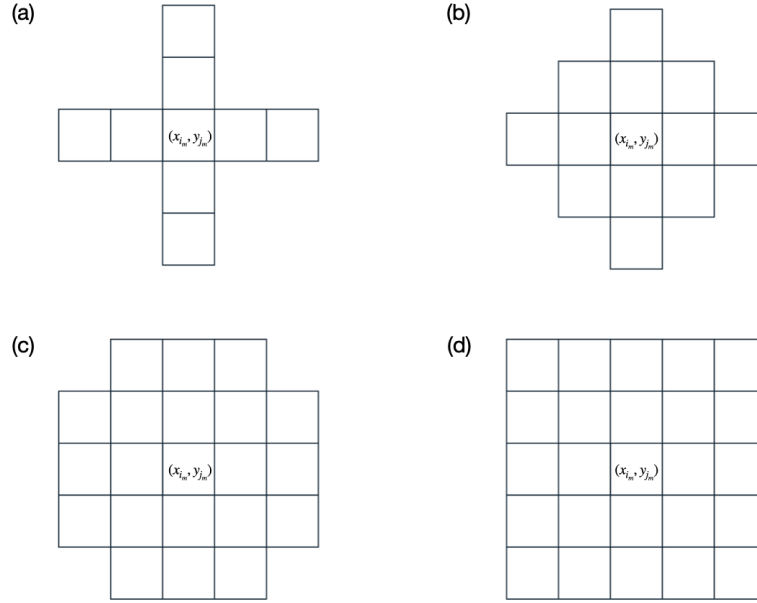


Figure 2.2: 2D GP stencils of $r_{gp} = 2$. The central cell $\mathbf{x}_m = (x_{i_m}, y_{j_m})$ is the cell where GP predictions are computed at the surrounding interface locations, $(x_{i_m \pm 1/2}, y_{j_m})$ or $(x_{i_m}, y_{j_m \pm 1/2})$, or corner locations, $(x_{i_m \pm 1/2}, y_{j_m \pm 1/2})$.

convergence falls short of delivering the anticipated convergence rate on calculations involving any off-axis information simply because it includes the axis-aligned cells only. The diamond stencil in (b) adds four extra corner cells to the cross configuration in (a), and as a result, it demonstrates a more improved accuracy than (a) when multidimensional calculations are considered in particular. It was indeed chosen as the default configuration for all 2D FV simulations in [20]. Yet, we found it delivers a reduced convergence rate in some applications considered in the current study, and we opted it out as a default choice. The two configurations in (c) and (d) successfully meet the anticipated convergence rates on various applications. While their performances on convergence rates are equivalent, the “blocky diamond” configuration (c) has a smaller

stencil size than the full square stencil in (d). This makes us choose the blocky diamond shape in (c) for our default GP stencil configuration. We also remark that 3D stencil configurations can be decided in a similar way, although we mainly focus on 2D calculations in the current study.

2.4 List of Applications

In the rest of the chapter, we present numerical results to validate the findings in the previous section further. We begin by providing a list of selected applications based on their popularity in modeling numerical differential equations. Most frequent operations include derivatives and integrals applied to pointwise and volume-averaged data types. We limit our interests to 1D and 2D applications only, although extensions to 3D applications are straightforward.

Table 2.1 summarizes a set of linear operations that are particularly of interest in discretizing partial differential equations. The input $\mathbf{g}$ is a 1D vector of length N that contains all participating grid data values from a local GP stencil (see Figs. 2.1 and 2.2), which are concatenated (or flattened) into a 1D vector form if the data values are collected from a multi-dimensional stencil. As discussed in the previous section, our default GP stencil in 2D is the blocky diamond shape in Fig. 2.2(c).

Common types of input data values appearing in numerical modeling are shown in the first column, showing only a single component. Target outputs are displayed in the second column, which are scalar values that represent linearly transformed function evaluations at the point of interest, $\mathbf{x} = \mathbf{x}_*$. For exposition purposes, we consistently

Table 2.1: A list of GP-recipe applications, including the baseline point-evaluation operator denoted by PE, e.g., $\mathcal{L}(g(\mathbf{x})) = g(\mathbf{x})$ for a function g . The input is a N -vector $\mathbf{g}$ whose h th component $[\mathbf{g}]_h = g(\mathbf{x}_h)$ is shown in the first column. The output in the second column is a scalar function value evaluated at $\mathbf{x} = \mathbf{x}_*$. Following Eq. (2.5), $\mathcal{L}_1$ is applied to $K(\mathbf{x}, \mathbf{x}')$ and $\mathcal{L}_2$ is applied to $K(\mathbf{x}_*, \mathbf{x}')$, acting on either $\mathbf{x}$ or $\mathbf{x}'$, or both. For simplicity, we assume that all calculations are performed using Cartesian geometry. In the first and second columns, we also show three abbreviations, “pt”, “va”, and “der”, representing pointwise, volume-averaged, and derivative data types in each input and output pair. The notation I_h is reserved to denote a 1D cell $[x_{h-1/2}, x_{h+1/2}]$ centered at x_h , and similarly for J_k centered at y_k , as well as for I_* centered at x_* and for J_* centered at y_* .

1D Input	1D Output	$x, x' \in \mathbb{R}$		Section
		$\mathcal{L}_1(K(x, x'))$	$\mathcal{L}_2(K(x_*, x'))$	
$f(x_h)$	$f(x_*)$	PE	PE	2.5.1.1
$f(x_h)$	$f^{(m)}(x_*)$	PE	$\frac{d^m}{dx^m}(\cdot)$	2.5.1.3
$f(x_h)$	$\frac{1}{\Delta x} \int_{I_*} f(x) dx$	PE	$\frac{1}{\Delta x} \int(\cdot) dx$	2.5.1.5
$\frac{1}{\Delta x} \int_{I_h} f(x) dx$	$f(x_*)$	$\frac{1}{\Delta x \Delta x'} \int \int(\cdot) dx dx'$	$\frac{1}{\Delta x'} \int(\cdot) dx'$	2.5.1.7
2D Input	2D Output	$\mathbf{x} = (x, y), \mathbf{x}' = (x', y') \in \mathbb{R}^2$		Section
		$\mathcal{L}_1(K(\mathbf{x}, \mathbf{x}'))$	$\mathcal{L}_2(K(\mathbf{x}_*, \mathbf{x}'))$	
$f(x_h, y_k)$	$f(x_*, y_*)$	PE	PE	2.5.1.2
$f(x_h, y_k)$	$\partial_y^n \partial_x^m f(x_*, y_*)$	PE	$\partial_y^n \partial_x^m(\cdot)$	2.5.1.4
$f(x_h, y_k)$	$\frac{1}{\Delta x \Delta y} \int_{I_*} \int_{J_*} f(x, y) dx dy$	PE	$\frac{1}{\Delta \mathbf{x}} \int(\cdot) d\mathbf{x}$	2.5.1.6
$\frac{1}{\Delta x \Delta y} \int_{I_h} \int_{J_k} f(x, y) dx dy$	$f(x_*, y_*)$	$\frac{1}{\Delta \mathbf{x} \Delta \mathbf{x}'} \int \int(\cdot) d\mathbf{x} d\mathbf{x}'$	$\frac{1}{\Delta \mathbf{x}'} \int(\cdot) d\mathbf{x}'$	2.5.1.8
$\frac{1}{\Delta x \Delta y} \int_{I_h} \int_{J_k} f(x, y) dx dy$	$\partial_y^n \partial_x^m f(x_*, y_*)$	$\frac{1}{\Delta \mathbf{x} \Delta \mathbf{x}'} \int \int(\cdot) d\mathbf{x} d\mathbf{x}'$	$\partial_y^n \partial_x^m (\frac{1}{\Delta \mathbf{x}'} \int(\cdot) d\mathbf{x}')$	2.5.1.9

choose the variable $\mathbf{x}$ (over $\mathbf{x}'$) to assign the evaluation point, $\mathbf{x} = \mathbf{x}_*$, although the roles of $\mathbf{x}$ and $\mathbf{x}'$ are interchangeable in general. The $\mathcal{L}_1$ operation transforms the kernel

function K linearly to construct the covariance kernel matrix $\mathbf{K}$, embedding the two-point correlations between the input data values at $\mathbf{x} = \mathbf{x}_h$ and $\mathbf{x}' = \mathbf{x}_k$ into $[\mathbf{K}]_{hk}$ in the data type prescribed in the first column. Such correlations are then channeled into the output via $\mathcal{L}_2$ by translating the correlated input data information into the target output at $\mathbf{x}_*$ in various data types. The operations of the $\mathcal{L}_2$ columns in the 1D and 2D boxes are to be finalized by plugging $\mathbf{x} = \mathbf{x}_*$ to obtain the outputs in the second column. Interested readers are encouraged to consult the visual representation of the kernels in Fig. 2 of [135].

2.5 Kernels for Smooth Function Approximations and Numerical Results

We present GP performance results on 1D and 2D grids for smoothly varying differentiable function approximations. For all smooth function test cases in this section, we use the SE kernel on the blocky diamond GP stencils of three GP radii, $r_{gp} = 1, 2$, and 3. Later in Section 4.1, we introduce a new GP kernel called the DAS (Discontinuous ArcSin) kernel to approximate discontinuous functions, which will be compared with the SE kernel and an existing neural network kernel to demonstrate its exceptional oscillation-control property at discontinuities.

2.5.1 Constrained Kernels

To conduct the tasks in Table 2.1, we first present the constrained kernels associated with different linear operators. These kernels are derived by applying linear

operators $\mathcal{L}_1$ to $K(\mathbf{x}, \mathbf{x}')$ and $\mathcal{L}_2$ to $K(\mathbf{x}_*, \mathbf{x}')$ as described in the caption of Table 2.1. Nothing needs to be done to the SE kernel for the baseline point-evaluation operator PE, which is in Eq. (3.8), and we do not repeat it here. In all cases, the input vector $[\mathbf{g}]_h = g(\mathbf{x}_h) = \mathcal{L}_1(f(\mathbf{x}_h))$ takes a linear operator $\mathcal{L}_1$ of the pointwise function values, $[\mathbf{f}]_h = f(\mathbf{x}_h)$, sampled from the cell centers of the blocky square. For example, in the volume average to point conversion, $\mathbf{g}$ would represent the volume-averaged data which is supplied as an input to the calculation. The general form of the GP prediction with linear constraints is introduced in Eq. (2.5), and is shown here again:

$$p_* = \mathbf{z}_*^T \cdot \mathbf{g} = \mathbf{t}_*^T \mathbf{C}^{-1} \cdot \mathbf{g} = \left(\mathcal{L}_2(K(\mathbf{x}_*, \mathbf{x}')) \right)^T \left(\mathcal{L}_1(K(\mathbf{x}, \mathbf{x}')) \right)^{-1} \cdot \mathbf{g}. \quad (2.14)$$

2.5.1.1 Kernels for pointwise function values to a pointwise interpolation in 1D

The basic SE kernel K_{SE} in Eq. (3.8) is all we need to calculate the baseline pointwise-to-pointwise function interpolation that takes an input data $\mathbf{f}$ to predict $f(\mathbf{x}_*)$, where both $\mathcal{L}_1$ and $\mathcal{L}_2$ are the point-evaluation operator PE. In this case, the GP prediction at $\mathbf{x}_*$ is simply the updated posterior mean in Eq. (2.2) using the SE kernel matrix, $\mathbf{K} = \mathbf{K}_{\text{SE}}$, i.e.,

$$f(\mathbf{x}_*) = \mathbf{z}_*^T \cdot \mathbf{f}, \quad \text{where } \mathbf{z}_*^T = \mathbf{k}_*^T \mathbf{K}^{-1}. \quad (2.15)$$

As an example, we present a grid convergence study to approximate the function $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$ discretized at $x = x_i$ on five grid resolutions, $N_x = 16, 32, 64, 128, 256$ in Table 2.2. GP takes the cell-centered function values $f(x_i)$ as input on three GP stencils of size $r_{gp} = 1, 2, 3$ to make function value predictions at

$x = x_{i+1/2}$. In all experimental grid convergence tests hereafter, we display the standard experimental order of convergence (EOC) rates defined by

$$\text{EOC} = \frac{\ln(E_c/E_r)}{\ln(2)}, \quad (2.16)$$

where E_c and E_r are the errors in the corresponding norm on the coarse and the next refined resolutions (e.g., E_c on 32 and E_r on 64), respectively, calculated against the exact solution on each grid. We see that EOCs are third, fifth, and seventh, respectively on $r_{gp} = 1, 2, 3$, successfully following the anticipated order, $(2r_{gp} + 1)$, on each grid resolution.

2.5.1.2 Kernels for pointwise function values to a pointwise interpolation in 2D

Similarly, the 2D pointwise interpolation uses the same SE kernel except that the kernel function K_{SE} now takes the two-dimensional form with $\mathbf{x} = (x, y), \mathbf{x}' = (x', y') \in \mathbb{R}^2$,

$$K_{\text{SE}}(\mathbf{x}, \mathbf{x}') = e^{-\frac{(x-x')^2}{2\ell^2}} e^{-\frac{(y-y')^2}{2\ell^2}}. \quad (2.17)$$

Once a set of function values on a 2D GP stencil is flattened into a 1D vector, $\mathbf{f}$, in an order consistent with the correlation order in the kernel matrix $\mathbf{K}_{\text{SE}}$, the 2D calculation proceeds with the same expression as in Eq. (2.15).

As in the 1D case in the previous section, we also present a 2D grid convergence study in Table 2.3. We discretize $f(x, y) = e^{-2x} \sin(4\pi y) + xe^{-y} \cos(2\pi x)$ at cell centers (x_i, y_j) over the square domain $[0, 1] \times [0, 1]$. Three GP calculations on the blocky stencil with $r_{gp} = 1, 2, 3$ task to predict the function values at the cell corner locations

Table 2.2: Grid convergence results for the 1D point-to-point interpolation in Section 2.5.1.1. GP calculations are performed using the SE kernel with $r_{gp} = 1, 2, 3$ to approximate the function values $f(x_{i+1/2})$ from the cell center input values $f(x_i)$ for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$. We choose $\ell = 0.05$ for the length scale hyperparameter.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	1.58e-02	1.93e-02	4.13e-02	7.95e-03	9.55e-03	1.81e-02	4.20e-03	5.11e-03	1.01e-02
32	5.61e-03	6.79e-03	1.68e-02	1.05e-03	1.39e-03	3.76e-03	4.39e-04	5.11e-04	1.15e-03
64	7.99e-04	9.65e-04	2.63e-03	4.83e-05	6.16e-05	1.80e-04	4.47e-06	6.30e-06	1.80e-05
128	1.01e-04	1.23e-04	3.44e-04	1.60e-06	2.05e-06	6.00e-06	3.85e-08	5.35e-08	1.57e-07
256	1.27e-05	1.54e-05	4.34e-05	5.05e-08	6.46e-08	1.89e-07	3.07e-10	4.25e-10	1.24e-09
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	—	—	—	—	—	—	—	—	—
32	1.490	1.506	1.298	2.924	2.777	2.266	3.259	3.323	3.133
64	2.812	2.815	2.672	4.438	4.500	4.385	6.616	6.341	6.000
128	2.983	2.974	2.936	4.919	4.912	4.907	6.862	6.879	6.840
256	2.997	2.992	2.988	4.983	4.985	4.990	6.968	6.977	6.985

$(x_{i+1/2}, y_{j+1/2})$ on five grid resolutions, $N_x = 16, 32, 64, 128, 256$. Again, the 2D EOC rates are third, fifth, and seventh, respectively on $r_{gp} = 1, 2, 3$, satisfactorily demonstrating the expected rates of convergence in all cases. In particular, the experimental result with $r_{gp} = 2$ is consistent with the anticipated and effective target order study for the blocky diamond stencil in Table 3.1. As mentioned, we set the blocky diamond GP stencil as our default 2D stencil in all 2D calculations hereafter.

Table 2.3: Grid convergence results for the 2D point-to-point interpolation in Section 2.5.1.2. GP calculations are performed using the SE kernel with $r_{gp} = 1, 2, 3$ to approximate the function values $f(x_{i+1/2}, y_{j+1/2})$ from the cell center input values $f(x_i, y_j)$ for $f(x, y) = e^{-2x} \sin(4\pi y) + xe^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$. As discussed, we set the blocky diamond stencil as our default 2D stencil configuration. We choose $\ell = 0.05$ for the length hyperparameter.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	4.13e-02	4.91e-02	1.13e-01	1.35e-02	1.58e-02	3.89e-02	4.55e-03	5.40e-03	1.39e-02
32	8.30e-03	9.98e-03	2.53e-02	2.26e-03	2.70e-03	6.73e-03	8.98e-04	1.06e-03	2.39e-03
64	1.02e-03	1.24e-03	3.37e-03	8.29e-05	1.00e-04	2.81e-04	1.02e-05	1.22e-05	3.14e-05
128	1.26e-04	1.53e-04	4.24e-04	2.56e-06	3.10e-06	8.40e-06	8.15e-08	9.78e-08	2.57e-07
256	1.56e-05	1.90e-05	5.29e-05	7.95e-08	9.64e-08	2.62e-07	6.31e-10	7.59e-10	2.01e-09
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	—	—	—	—	—	—	—	—	—
32	2.32	2.30	2.15	2.58	2.55	2.53	2.34	2.35	2.54
64	3.02	3.01	2.91	4.77	4.75	4.58	6.45	6.44	6.25
128	3.02	3.02	2.99	5.02	5.01	5.06	6.97	6.97	6.94
256	3.01	3.01	3.00	5.01	5.01	5.00	7.01	7.01	7.00

2.5.1.3 Kernels for pointwise function values to a derivative in 1D

The GP prediction from pointwise input values to a derivative takes $\mathcal{L}_1 = \text{PE}$ and $\mathcal{L}_2$ is to be a relevant derivative operator at $x = x_*$. In 1D, we get

$$\mathcal{L}_2(K_{\text{SE}}(x_*, x')) = \left(\frac{d^m}{dx^m} \mathbf{K}_{\text{SE}}(x, x') \right) \bigg|_{x=x_*}. \quad (2.18)$$

For brevity, we only show kernels for the first, second, and third derivatives here. For the first derivative ($m = 1$), we get

$$\mathcal{L}_2(K_{\text{SE}}(x_*, x')) = -\frac{(-2x_* + 2x') e^{-\frac{(x_* - x')^2}{2\ell^2}}}{2\ell^2}. \quad (2.19)$$

The second derivative kernel ($m = 2$) takes of the form

$$\mathcal{L}_2(K_{\text{SE}}(x_*, x')) = \frac{\left(-1 + \frac{(x_* - x')^2}{\ell^2}\right) e^{-\frac{(x_* - x')^2}{2\ell^2}}}{\ell^2}, \quad (2.20)$$

and finally, for the third derivative ($m = 3$), the kernel is given as

$$\mathcal{L}_2(K_{\text{SE}}(x_*, x')) = \frac{\left(-3 + \frac{(x_* - x')^2}{\ell^2}\right) (x - x') e^{-\frac{(x_* - x')^2}{2\ell^2}}}{\ell^4}. \quad (2.21)$$

Numerical convergence results are shown in Tables 2.4 to 2.6, respectively for the first, second, and third derivatives of the function $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ defined on $[0, 1]$. As before, given the pointwise function values $f(x_i)$ at cell centers, we compute $f'(x_{i+1/2})$, $f''(x_{i+1/2})$, and $f'''(x_{i+1/2})$ at the cell interface locations using the corresponding kernel functions in Eqs. (2.19) to (2.21). As the grid refines, these test results satisfy the anticipated and effective convergence rates in Table 3.1 very well as the order of the derivative varies.

2.5.1.4 Kernels for pointwise function values to a derivative in 2D

In a similar fashion, kernels for derivatives in 2D can be obtained by considering $\mathcal{L}_1 = \text{PE}$ and $\mathcal{L}_2$ is a respective derivative operator at $\mathbf{x} = (x, y)$, followed by the derivative evaluation at $\mathbf{x} = \mathbf{x}_* = (x_*, y_*)$. A general form is written as

$$\mathcal{L}_2(K_{\text{SE}}(\mathbf{x}_*, \mathbf{x}')) = \left(\frac{\partial^{m+n}}{\partial y^n \partial x^m} K_{\text{SE}}(\mathbf{x}, \mathbf{x}') \right) \bigg|_{\mathbf{x}=\mathbf{x}_*}. \quad (2.22)$$

Table 2.4: Grid convergence results for the 1D point-to-1st derivative conversion in Section 2.5.1.3. GP calculations are performed at cell interface locations $x_{i+1/2}$ using the kernel in Eq. (2.19) with $r_{gp} = 1, 2, 3$ and $\ell = 0.05$ to approximate the function $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	3.35e-01	4.05e-01	8.93e-01	8.90e-02	1.05e-01	2.14e-01	1.13e-02	1.44e-02	3.01e-02
32	1.06e-01	1.29e-01	2.81e-01	1.52e-02	1.84e-02	3.87e-02	5.40e-03	6.69e-03	1.57e-02
64	3.04e-02	3.70e-02	1.02e-01	1.12e-03	1.40e-03	3.74e-03	7.29e-05	9.24e-05	2.32e-04
128	8.34e-03	1.01e-02	2.84e-02	7.76e-05	9.84e-05	2.89e-04	1.31e-06	1.79e-06	5.23e-06
256	2.13e-03	2.59e-03	7.31e-03	5.05e-06	6.43e-06	1.89e-05	2.15e-08	2.98e-08	8.79e-08
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	–	–	–	–	–	–	–	–	–
32	1.662	1.654	1.668	2.550	2.516	2.463	1.069	1.101	0.937
64	1.804	1.798	1.466	3.760	3.715	3.373	6.211	6.179	6.080
128	1.864	1.871	1.841	3.853	3.834	3.695	5.802	5.689	5.471
256	1.970	1.963	1.958	3.943	3.935	3.932	5.925	5.910	5.896

We only show a couple of them here for exposition purposes. The kernel for the first-order partial derivative in x is shown to be

$$\frac{\partial}{\partial x} K_{SE}(\mathbf{x}_*, \mathbf{x}') = -\frac{(-2x_* + 2x') e^{-\frac{(x_* - x')^2}{2\ell^2}} e^{-\frac{(y_* - y')^2}{2\ell^2}}}{2\ell^2}. \quad (2.23)$$

Similarly, the kernel for ∂_y becomes

$$\frac{\partial}{\partial y} K_{SE}(\mathbf{x}_*, \mathbf{x}') = -\frac{(-2y_* + 2y') e^{-\frac{(x_* - x')^2}{2\ell^2}} e^{-\frac{(y_* - y')^2}{2\ell^2}}}{2\ell^2}. \quad (2.24)$$

The kernel for the cross derivative ∂_{xy} is

$$\frac{\partial}{\partial x} \frac{\partial}{\partial y} K_{SE}(\mathbf{x}_*, \mathbf{x}') = \frac{(-2x_* + 2x')(-2y_* + 2y') e^{-\frac{(x_* - x')^2}{2\ell^2}} e^{-\frac{(y_* - y')^2}{2\ell^2}}}{4\ell^4}, \quad (2.25)$$

Table 2.5: Grid convergence results for the 1D point-to-2nd derivative conversion in Section 2.5.1.3 using the kernel in Eq. (2.20). All other GP (hyper)parameters are the same as in Table 2.4.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	2.98e+01	3.68e+01	7.89e+01	1.60e+01	1.91e+01	3.65e+01	8.79e+00	1.07e+01	2.07e+01
32	4.51e+01	5.47e+01	1.36e+02	9.32e+00	1.24e+01	3.36e+01	4.01e+00	4.66e+00	1.04e+01
64	2.59e+01	3.12e+01	8.50e+01	1.73e+00	2.20e+00	6.44e+00	1.66e-01	2.34e-01	6.67e-01
128	1.31e+01	1.59e+01	4.46e+01	2.30e-01	2.95e-01	8.64e-01	5.74e-03	7.98e-03	2.34e-02
256	6.59e+00	8.05e+00	2.26e+01	2.92e-02	3.74e-02	1.09e-01	1.84e-04	2.55e-04	7.43e-04
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	–	–	–	–	–	–	–	–	–
32	-0.595	-0.574	-0.790	0.781	0.622	0.120	1.133	1.197	0.990
64	0.801	0.810	0.682	2.430	2.494	2.385	4.594	4.320	3.967
128	0.982	0.971	0.930	2.911	2.902	2.898	4.855	4.871	4.831
256	0.990	0.986	0.981	2.976	2.977	2.982	4.961	4.969	4.977

and lastly the kernel for ∂_{xx} is

$$\frac{\partial^2}{\partial x^2} K_{SE}(\mathbf{x}_*, \mathbf{x}') = \frac{\left(-1 + \frac{(x_* - x')^2}{\ell^2}\right) e^{-\frac{(x_* - x')^2}{2\ell^2}} e^{-\frac{(y_* - y')^2}{2\ell^2}}}{\ell^2}. \quad (2.26)$$

Other higher-order derivatives can be readily obtained in a similar fashion.

As a proof of concept, we show numerical results for computing a Laplacian operator of a function $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ to demonstrate the flexibility of the GP method. In this example, we choose a simple function $f(x, y) = \sin(x) \sin(y)$ on a square domain $[0, 2\pi] \times [0, 2\pi]$ although any other smooth functions would work well too. The function, the GP prediction of the Laplacian at cell corners $(x_{i+1/2}, y_{j+1/2})$, and the exact Lapla-

Table 2.6: Grid convergence results for the 1D point-to-3rd derivative conversion in Section 2.5.1.3 using the kernel in Eq. (2.21). All other GP (hyper)parameters are the same as in Table 2.4.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	2.07e+03	2.49e+03	5.50e+03	5.69e+02	6.73e+02	1.38e+03	7.08e+01	8.76e+01	1.83e+02
32	2.52e+03	3.06e+03	6.72e+03	4.04e+02	4.89e+02	1.03e+03	1.48e+02	1.84e+02	4.31e+02
64	2.91e+03	3.55e+03	9.76e+03	1.20e+02	1.50e+02	3.99e+02	8.06e+00	1.02e+01	2.57e+01
128	3.23e+03	3.92e+03	1.10e+04	3.35e+01	4.24e+01	1.24e+02	5.83e-01	8.00e-01	2.34e+00
256	3.32e+03	4.05e+03	1.14e+04	8.76e+00	1.12e+01	3.28e+01	3.86e-02	5.35e-02	1.58e-01
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	–	–	–	–	–	–	–	–	–
32	-0.287	-0.300	-0.289	0.496	0.459	0.417	-1.067	-1.069	-1.238
64	-0.209	-0.214	-0.538	1.753	1.708	1.369	4.202	4.169	4.070
128	-0.151	-0.143	-0.174	1.839	1.820	1.680	3.788	3.675	3.457
256	-0.039	-0.046	-0.051	1.934	1.925	1.923	3.916	3.901	3.887

cian $\nabla^2 f(x_{i+1/2}, y_{j+1/2})$ are plotted in Fig. 2.3, where the GP solution and the exact solutions match well visually. To quantify GP's results, we conducted a grid convergence study for this test in Table 2.7. In all test cases, GP solutions successfully converge on each grid resolution following the anticipated and effective convergence rates described in Table 3.1.

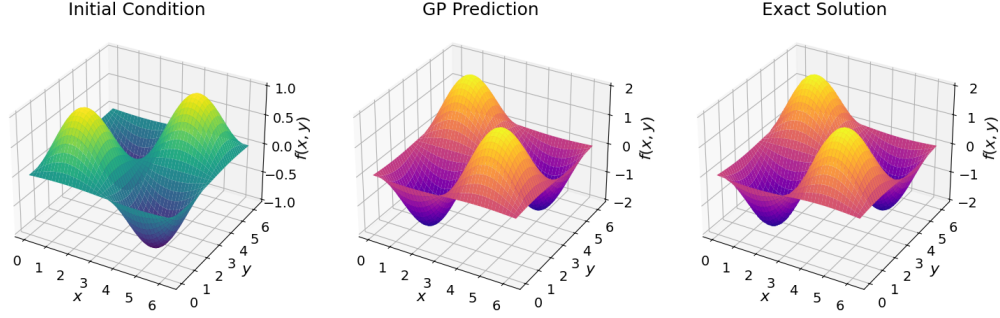


Figure 2.3: GP’s prediction of $\nabla^2 \sin(x) \sin(y)$ on $[0, 2\pi] \times [0, 2\pi]$ using a 128×128 grid resolution with $r_{gp} = 2$ and $\ell = \pi/10$. The GP result shows a great visual agreement with the exact Laplacian calculation. The convergence study for this case is shown in Table 2.7.

2.5.1.5 Kernels for pointwise function values to a volume average value in 1D

As the input values are pointwise, we choose $\mathcal{L}_1$ =PE to correlate pointwise values at x and x' , while we set $\mathcal{L}_2$ to be a volume-averaging integral operator over x , followed by the integral evaluation at $x = x_*$. That is,

$$\mathcal{L}_2(K_{SE}(x_*, x')) = \left(\frac{1}{\Delta x} \int_{I_*} K_{SE}(x, x') dx \right) \Big|_{x=x_*}. \quad (2.27)$$

Here, the operation of $\mathcal{L}_2$ can be understood as a bridging step that converts the pointwise values to the target volume average value at $x = x_*$ by considering a Bayesian inference between the pointwise input data at each x' and the sought volume-averaging evaluation at $x = x_*$. Unlike other differential or point-evaluation operations where the point x_* could be chosen from the interior or boundaries of a cell I_* , we set x_* to be a

Table 2.7: Grid convergence results for the 2D point-to-Laplacian conversion in Section 2.5.1.4. GP considers to approximate the Laplacian of the function $f(x, y) = \sin(x) \sin(y)$ on $[0, 2\pi] \times [0, 2\pi]$ at $(x_{i+1/2}, y_{j+1/2})$ by taking the function values $f(x_i, y_j)$ at cell centers as input. The prediction utilizes the kernel in Eq. (2.26) with $\ell = \pi/10$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	1.15e+02	2.13e+01	5.90e+00	5.58e+01	1.04e+01	3.07e+00	1.72e+01	3.04e+00	6.89e-01
32	8.34e+01	1.47e+01	3.65e+00	3.08e+01	5.43e+00	1.34e+00	1.33e+01	2.35e+00	5.75e-01
64	3.86e+01	6.82e+00	1.64e+00	3.89e+00	6.85e-01	1.64e-01	5.37e-01	9.48e-02	2.25e-02
128	1.83e+01	3.23e+00	7.70e-01	4.54e-01	8.02e-02	1.92e-02	1.64e-02	2.89e-03	6.79e-04
256	9.10e+00	1.61e+00	3.73e-01	5.63e-02	9.95e-03	2.31e-03	5.05e-04	8.92e-05	2.05e-05
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	–	–	–	–	–	–	–	–	–
32	0.47	0.53	0.69	0.86	0.93	1.19	0.37	0.37	0.26
64	1.11	1.11	1.15	2.99	2.99	3.04	4.63	4.63	4.67
128	1.08	1.08	1.09	3.10	3.09	3.09	5.03	5.03	5.05
256	1.00	1.01	1.05	3.01	3.01	3.05	5.02	5.02	5.05

cell center location to make the volume-averaging meaningful in our calculations. The same is true for the 2D volume averages in the next subsection. With this choice of x_* and $I_* = [x_* - \Delta x/2, x_* + \Delta x/2]$, Eq. (2.27) is written in terms of the error function by,

$$\begin{aligned}
\mathcal{L}_2(K_{SE}(x_*, x')) &= \sqrt{\frac{\pi}{2}} \frac{\ell}{\Delta x} \left(\operatorname{erf} \left(\frac{(x_* + \Delta x/2 - x')}{\sqrt{2}\ell} \right) - \operatorname{erf} \left(\frac{(x_* - \Delta x/2 - x')}{\sqrt{2}\ell} \right) \right) \\
&= \sqrt{\frac{\pi}{2}} \frac{\ell}{\Delta x} \left(\operatorname{erf} \left(\frac{(\Delta_{x^*x'} + 1/2)}{\sqrt{2}\ell/\Delta x} \right) - \operatorname{erf} \left(\frac{(\Delta_{x^*x'} - 1/2)}{\sqrt{2}\ell/\Delta x} \right) \right), \quad (2.28)
\end{aligned}$$

where $\Delta_{x^*x'} = (x_* - x')/\Delta x$.

Since the output is the integral average, $\frac{1}{\Delta x} \int_{I_*} f(x) dx$, the grid Δx from the

integration cancels with the averaging Δx . Hence, for each r_{gp} , the anticipated convergence rate is the same as the $(2r_{gp} + 1)$ rate in the baseline pointwise to pointwise conversion in Section 2.5.1.1 and Table 3.1. Experimental GP results are collected in Table 2.8, demonstrating the anticipated third, fifth, and seventh-order convergence rates on $r_{gp} = 1, 2, 3$, respectively.

Table 2.8: Grid convergence results for the 1D point-to-volume average conversion in Section 2.5.1.5. Input values are $f(x_i)$ for the function $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$ and GP predicts $\frac{1}{\Delta x} \int_{I_i} f(x) dx$ at x_i using the kernel in Eq. (2.28) with $\ell = 0.05$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	1.04e-02	1.27e-02	2.72e-02	5.17e-03	6.23e-03	1.18e-02	2.71e-03	3.30e-03	6.55e-03
32	3.71e-03	4.49e-03	1.11e-02	6.77e-04	9.00e-04	2.43e-03	2.82e-04	3.28e-04	7.42e-04
64	5.30e-04	6.41e-04	1.75e-03	3.14e-05	4.00e-05	1.17e-04	2.88e-06	4.06e-06	1.16e-05
128	6.73e-05	8.18e-05	2.29e-04	1.04e-06	1.33e-06	3.91e-06	2.48e-08	3.46e-08	1.02e-07
256	8.43e-06	1.03e-05	2.89e-05	3.29e-08	4.21e-08	1.23e-07	1.99e-10	2.75e-10	8.02e-10
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	–	–	–	–	–	–	–	–	–
32	1.488	1.502	1.296	2.934	2.790	2.277	3.266	3.330	3.143
64	2.804	2.807	2.660	4.429	4.491	4.375	6.610	6.335	5.997
128	2.979	2.970	2.933	4.916	4.909	4.904	6.859	6.876	6.837
256	2.996	2.991	2.987	4.982	4.984	4.989	6.967	6.976	6.984

2.5.1.6 Kernels for pointwise function values to a volume average value in 2D

Similar to the 1D case, GP converts pointwise function input values to predict a volume average value at $\mathbf{x} = \mathbf{x}_* = (x_*, y_*)$ in 2D using $\mathcal{L}_1$ =PE and

$$\begin{aligned}\mathcal{L}_2(K_{\text{SE}}(\mathbf{x}_*, \mathbf{x}')) &= \left(\frac{1}{\Delta \mathbf{x}} \int_{I_* \times J_*} K_{\text{SE}}(\mathbf{x}, \mathbf{x}') d\mathbf{x} \right) \Big|_{\mathbf{x}=\mathbf{x}_*} \\ &= \left(\frac{1}{\Delta x \Delta y} \int_{I_*} \int_{J_*} K_{\text{SE}}(\mathbf{x}, \mathbf{x}') dx dy \right) \Big|_{\mathbf{x}=\mathbf{x}_*}.\end{aligned}\tag{2.29}$$

Again, the $\mathcal{L}_2$ operation channels the pointwise input values to the 2D volume average value at $\mathbf{x}_*$ by constructing a correlation between the two different data type values, i.e., the pointwise inputs sampled from cell center points $\mathbf{x}'$ and the sought volume average value anchored at the point $\mathbf{x}_*$ centered in a 2D rectangular cell $I_* \times J_* = [x_* - \Delta x/2, x_* + \Delta x/2] \times [y_* - \Delta y/2, y_* + \Delta y/2]$. Due to the dimensional factorization property in the SE kernel in Eq. (2.13), the 2D version of the volume-averaging kernel in Eq. (3.13) can be easily written as

$$\mathcal{L}_2(K_{\text{SE}}(\mathbf{x}_*, \mathbf{x}')) = \prod_{d=x,y} \sqrt{\frac{\pi}{2}} \frac{\ell}{\Delta d} \left(\text{erf} \left(\frac{(\Delta_{\mathbf{x}^* \mathbf{x}'}^d + 1/2)}{\sqrt{2}\ell/\Delta d} \right) - \text{erf} \left(\frac{(\Delta_{\mathbf{x}^* \mathbf{x}'}^d - 1/2)}{\sqrt{2}\ell/\Delta d} \right) \right),\tag{2.30}$$

where $\Delta_{\mathbf{x}^* \mathbf{x}'}^d = \mathbf{e}_d \cdot (\mathbf{x}_* - \mathbf{x}')/\Delta d$ with $\mathbf{e}_d$ being the unit vector in each d -direction. Note that Eqs. (2.28) and (2.30) recover the $\mathbf{t}_*$ kernel form in [20]. As in the 1D case, GP converges to the anticipated third, fifth, and seventh-order solutions in 2D as well, as shown in Table 2.9.

Table 2.9: Grid convergence results for the 2D point-to-volume average conversion in Section 2.5.1.6. Input values are $f(x_i, y_j)$ for the function $f(x, y) = e^{-2x} \sin(4\pi y) + xe^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$ and GP predicts $\frac{1}{\Delta x \Delta y} \int_{I_i} \int_{I_j} f(x, y) dx dy$ at (x_i, y_j) using the kernel in Eq. (2.30) with $\ell = 0.05$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	2.71e-02	3.23e-02	7.39e-02	8.54e-03	1.00e-02	2.46e-02	2.96e-03	3.52e-03	9.23e-03
32	5.52e-03	6.63e-03	1.68e-02	1.46e-03	1.75e-03	4.37e-03	5.77e-04	6.81e-04	1.53e-03
64	6.83e-04	8.25e-04	2.24e-03	5.40e-05	6.51e-05	1.82e-04	6.61e-06	7.90e-06	2.03e-05
128	8.39e-05	1.02e-04	2.82e-04	1.67e-06	2.02e-06	5.48e-06	5.27e-08	6.32e-08	1.66e-07
256	1.04e-05	1.27e-05	3.53e-05	5.18e-08	6.28e-08	1.71e-07	4.09e-10	4.91e-10	1.30e-09
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	–	–	–	–	–	–	–	–	–
32	2.30	2.28	2.14	2.54	2.52	2.49	2.36	2.37	2.59
64	3.01	3.01	2.90	4.76	4.75	4.59	6.45	6.43	6.24
128	3.02	3.01	2.99	5.01	5.01	5.06	6.97	6.97	6.93
256	3.01	3.01	3.00	5.01	5.01	5.00	7.01	7.01	7.00

2.5.1.7 Kernels for volume average values to a pointwise value in 1D

The conversion from volume averages at cell centers to a pointwise value at x_* is one of the crucial building blocks in finite volume methods. The most common example is found in high-order reconstruction schemes that aim to achieve high-order accurate values called the Riemann states at cell interfaces, $x_* = x_{i\pm 1/2}$. Using GP, such a conversion can be formulated by the two kernels in Eqs. (2.10) and (2.11) for an input vector $\mathbf{g}$, whose h th component is a 1D volume-averaged quantity $\frac{1}{\Delta x} \int_{I_h} f(x) dx$ of a function f . These volume average values between x and x' are correlated by

$\mathcal{L}_1(K_{\text{SE}}(x, x'))$ that takes the volume-averaging integration of $K_{\text{SE}}(x, x')$ at both x and x' ,

$$\begin{aligned} \mathcal{L}_1(K_{\text{SE}}(x, x')) = & \sqrt{\pi} \left(\frac{\ell}{\Delta x} \right)^2 \left\{ \left(\frac{\Delta_{xx'} + 1}{\sqrt{2}\ell/\Delta x} \operatorname{erf} \left[\frac{\Delta_{xx'} + 1}{\sqrt{2}\ell/\Delta x} \right] + \frac{\Delta_{xx'} - 1}{\sqrt{2}\ell/\Delta x} \operatorname{erf} \left[\frac{\Delta_{xx'} - 1}{\sqrt{2}\ell/\Delta x} \right] \right) \right. \\ & + \frac{1}{\sqrt{\pi}} \left(\exp \left[-\frac{(\Delta_{xx'} + 1)^2}{2(\ell/\Delta x)^2} \right] + \exp \left[-\frac{(\Delta_{xx'} - 1)^2}{2(\ell/\Delta x)^2} \right] \right) \\ & \left. - 2 \left(\frac{\Delta_{xx'}}{\sqrt{2}\ell/\Delta x} \operatorname{erf} \left[\frac{\Delta_{xx'}}{\sqrt{2}\ell/\Delta x} \right] + \frac{1}{\sqrt{\pi}} \exp \left[-\frac{\Delta_{xx'}^2}{2(\ell/\Delta x)^2} \right] \right) \right\}. \quad (2.31) \end{aligned}$$

These correlated volume averages are then channeled through the second operation by $\mathcal{L}_2(K_{\text{SE}}(x_*, x'))$ where the volume-averaging operation is kept at x' while the pointwise evaluation is to take place at $x = x_*$. This is be done by finalizing the conversion process using Eq. (2.28).

Rigorously speaking, the $\mathcal{L}_2$ kernels are different in converting pointwise inputs to a volume average in Section 2.5.1.5 and volume average inputs to a pointwise output in this section. This clarification is noted in Table 2.1 making clear distinctions between $\frac{1}{\Delta \mathbf{x}} \int(\cdot) d\mathbf{x}$ and $\frac{1}{\Delta \mathbf{x}'} \int(\cdot) d\mathbf{x}'$ to indicate over which variable ($\mathbf{x}$ or $\mathbf{x}'$) the integration is computed. It is worth mentioning that there is no actual difference in the final expression of the kernel, though, since $\mathbf{x}$ and $\mathbf{x}'$ are interchangeable, and the SE kernel is symmetric positive definite. Lastly, the pointwise prediction at x_* is obtained by combining the kernels in Eqs. (2.28) and (2.31) according to Eq. (2.14).

Similar to the case in Section 2.5.1.5, we expect the volume average-to-pointwise conversion converges at $(2r_{gp} + 1)$. Our experimental results confirm this convergence rate for each $r_{gp} = 1, 2, 3$ in Table 2.10.

Table 2.10: Grid convergence results for the 1D volume-average-to-point conversion in Section 2.5.1.7. Input values are the integral average values $\frac{1}{\Delta x} \int_{I_i} f(x) dx$ at cell centers x_i and output values are pointwise function values $f(x_{i+1/2})$ at cell interface locations for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$. GP utilizes the kernel in Eq. (2.31) with $\ell = 0.05$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	1.74e-02	2.25e-02	4.81e-02	1.27e-02	1.55e-02	3.03e-02	7.28e-03	8.90e-03	1.80e-02
32	6.92e-03	8.43e-03	2.11e-02	1.27e-03	1.72e-03	4.68e-03	5.72e-04	6.51e-04	1.46e-03
64	1.05e-03	1.27e-03	3.46e-03	6.61e-05	8.44e-05	2.47e-04	6.23e-06	8.75e-06	2.48e-05
128	1.34e-04	1.63e-04	4.58e-04	2.25e-06	2.89e-06	8.46e-06	5.55e-08	7.73e-08	2.27e-07
256	1.69e-05	2.06e-05	5.78e-05	7.17e-08	9.17e-08	2.68e-07	4.48e-10	6.20e-10	1.81e-09
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	—	—	—	—	—	—	—	—	—
32	1.331	1.420	1.189	3.321	3.173	2.697	3.669	3.773	3.624
64	2.721	2.731	2.607	4.264	4.348	4.242	6.523	6.218	5.878
128	2.967	2.958	2.920	4.876	4.871	4.868	6.810	6.822	6.777
256	2.993	2.988	2.984	4.973	4.975	4.981	6.952	6.962	6.969

2.5.1.8 Kernels for volume average values to a pointwise value in 2D

The 1D calculation in the previous section is easily extended to 2D by combining $\mathcal{L}_2(K_{\text{SE}}(\mathbf{x}_*, \mathbf{x}'))$ in Eq. (2.30) and $\mathcal{L}_1(K_{\text{SE}}(\mathbf{x}, \mathbf{x}'))$ defined by

$$\begin{aligned} \mathcal{L}_1(K_{\text{SE}}(\mathbf{x}, \mathbf{x}')) = & \prod_{d=x,y} \sqrt{\pi} \left(\frac{\ell}{\Delta d} \right)^2 \left\{ \left(\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d + 1}{\sqrt{2}\ell/\Delta d} \operatorname{erf} \left[\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d + 1}{\sqrt{2}\ell/\Delta d} \right] + \frac{\Delta_{\mathbf{x}\mathbf{x}'}^d - 1}{\sqrt{2}\ell/\Delta d} \operatorname{erf} \left[\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d - 1}{\sqrt{2}\ell/\Delta d} \right] \right) \right. \\ & + \frac{1}{\sqrt{\pi}} \left(\exp \left[-\frac{(\Delta_{\mathbf{x}\mathbf{x}'}^d + 1)^2}{2(\ell/\Delta d)^2} \right] + \exp \left[-\frac{(\Delta_{\mathbf{x}\mathbf{x}'}^d - 1)^2}{2(\ell/\Delta d)^2} \right] \right) \\ & \left. - 2 \left(\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d}{\sqrt{2}\ell/\Delta d} \operatorname{erf} \left[\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d}{\sqrt{2}\ell/\Delta d} \right] + \frac{1}{\sqrt{\pi}} \exp \left[-\frac{(\Delta_{\mathbf{x}\mathbf{x}'}^d)^2}{2(\ell/\Delta d)^2} \right] \right) \right\}. \quad (2.32) \end{aligned}$$

The resulting pointwise GP prediction has been used for designing two-dimensional high-order accurate finite volume Riemann states at $\mathbf{x}_* = (x_{i\pm 1/2}, y_j)$ or $(x_i, y_{j\pm 1/2})$ in [20], directly converting the volume average quantities $\mathbf{g}$ to a point value p_* via the one-step dot product between the (pre-computed and saved) prediction vector $\mathbf{z}_*^T$ and the input vector $\mathbf{g}$ in Eq. (2.14). This approach is an effective kernel-based alternative to the existing polynomial-based high-order multi-dimensional reconstruction schemes that incorporate high-order correction terms with Taylor series expansions [103, 27].

In Table 2.11, we once again confirm that the experimental convergence follows the $(2r_{gp} + 1)$ rate for each GP radius, $r_{gp} = 1, 2, 3$, similar to the results in Section 2.5.1.6.

2.5.1.9 Kernels for volume-average values to a derivative in 2D

To make a GP prediction of a derivative value at $\mathbf{x}_*$ using 2D volume average values as input, one extra step is required in the calculation of Section 2.5.1.8. While

Table 2.11: Grid convergence results for the 2D volume average-to-point conversion in Section 2.5.1.8. Input values are the integral average values $\frac{1}{\Delta x \Delta y} \int_{I_i} \int_{I_j} f(x, y) dx dy$ at cell centers (x_i, y_j) and output values are pointwise function values $f(x_{i+1/2}, y_{i+1/2})$ at cell corner locations for $f(x, y) = e^{-2x} \sin(4\pi y) + x e^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$. GP utilizes the kernel in Eq. (2.32) with $\ell = 0.05$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	4.72e-02	5.60e-02	1.32e-01	1.72e-02	2.02e-02	5.05e-02	7.97e-03	9.46e-03	2.33e-02
32	1.03e-02	1.24e-02	3.17e-02	2.81e-03	3.36e-03	8.36e-03	1.10e-03	1.29e-03	2.88e-03
64	1.34e-03	1.62e-03	4.42e-03	1.14e-04	1.37e-04	3.87e-04	1.42e-05	1.70e-05	4.37e-05
128	1.67e-04	2.03e-04	5.63e-04	3.61e-06	4.37e-06	1.18e-05	1.18e-07	1.41e-07	3.71e-07
256	2.08e-05	2.54e-05	7.05e-05	1.13e-07	1.37e-07	3.72e-07	9.19e-10	1.11e-09	2.92e-09
Grid Res.	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	—	—	—	—	—	—	—	—	—
32	2.20	2.18	2.06	2.62	2.59	2.59	2.86	2.87	3.02
64	2.94	2.93	2.84	4.63	4.61	4.43	6.27	6.25	6.04
128	3.00	3.00	2.97	4.98	4.97	5.03	6.92	6.91	6.88
256	3.01	3.00	3.00	5.00	5.00	4.99	7.00	7.00	6.99

we keep the $\mathcal{L}_1(K_{SE}(\mathbf{x}, \mathbf{x}'))$ kernel the same, we need to add the sought differential operation at $\mathbf{x}_*$ to the $\mathcal{L}_2$ kernel operation in Eq. (2.30) by taking respective derivatives of it. For example, to pursue ∂_{xy} from the volume average input values, we define the

kernel to be

$$\begin{aligned}
\mathcal{L}_2(K_{\text{SE}}(\mathbf{x}_*, \mathbf{x}')) = & \\
& \prod_{d=x,y} \frac{\partial}{\partial d} \sqrt{\frac{\pi}{2}} \frac{\ell}{\Delta d} \left\{ \text{erf} \left(\frac{(\Delta_{dd'} + 1/2)}{\sqrt{2}\ell/\Delta d} \right) - \text{erf} \left(\frac{(\Delta_{dd'} - 1/2)}{\sqrt{2}\ell/\Delta d} \right) \right\} \Big|_{d=d_*} \\
= & \prod_{d=x,y} \frac{1}{\Delta d} \left\{ \exp \left(-\frac{(1 + 2\Delta_{d_*d'})^2}{8(\ell/\Delta d)^2} \right) - \exp \left(-\frac{(1 - 2\Delta_{d_*d'})^2}{8(\ell/\Delta d)^2} \right) \right\}. \quad (2.33)
\end{aligned}$$

In Table 2.12, we demonstrate the experimental convergence rates for the cross derivative approximations using the kernel in Eq. (2.33). The anticipated convergence rate in this case is to be $(2r_{gp} - 1)$ with the cross derivative as output. The results in Table 2.12 show that the EOC rates are a little higher than $(2r_{gp} - 1)$ for each $r_{gp} = 1, 2, 3$, which is also observed in the effective order of convergence rates in Table 3.1 for the blocky diamond stencil.

2.6 Conclusion

We have presented a kernel-based GP approach to approximating function values under various linear operations. These operations involve function interpolations, numerical derivatives and integrations, and input-output data type conversions, all appearing very commonly in a broad spectrum of science and engineering applications. In particular, these operations are crucial building blocks in numerical modeling across popular discrete fluid dynamics methods, including finite difference, finite volume, finite element, Galerkin, and many others. This study showed that the GP approach can deliver a variable order of convergence depending on the GP stencil size determined by the GP radius, r_{gp} . Instead of conventional polynomial-based methods, these GP

Table 2.12: Grid convergence results for the 2D volume-average-to-point conversion in Section 2.5.1.9. Input values are the integral average values $\frac{1}{\Delta x \Delta y} \int_{I_i} \int_{I_j} f(x, y) dx dy$ at cell centers (x_i, y_j) and output values are the cross derivative $\partial_{xy} f(x_{i+1/2}, y_{i+1/2})$ evaluated at cell corner locations for $f(x, y) = e^{-2x} \sin(4\pi y) + x e^{-y} \cos(2\pi x)$ on $[0, 1] \times [0, 1]$. GP utilizes the kernel in Eq. (2.33) with $\ell = 0.05$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	8.73e+00	1.06e+01	2.68e+01	5.85e+00	6.93e+00	1.65e+01	1.52e+00	2.00e+00	6.40e+00
32	1.50e+00	2.01e+00	7.96e+00	5.13e-01	6.33e-01	2.16e+00	1.40e-01	1.82e-01	6.25e-01
64	1.48e-01	2.69e-01	1.28e+00	1.65e-02	3.07e-02	1.67e-01	8.88e-04	1.64e-03	8.62e-03
128	5.01e-02	7.00e-02	3.53e-01	7.65e-04	1.12e-03	6.78e-03	1.75e-05	2.57e-05	1.44e-04
256	1.65e-02	2.14e-02	7.73e-02	7.10e-05	9.26e-05	3.62e-04	3.96e-07	5.16e-07	1.96e-06
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	—	—	—	—	—	—	—	—	—
32	2.54	2.39	1.75	3.51	3.45	2.94	3.44	3.45	3.36
64	3.34	2.90	2.64	4.95	4.37	3.70	7.30	6.80	6.18
128	1.56	1.94	1.85	4.43	4.78	4.62	5.67	6.00	5.90
256	1.60	1.71	2.19	3.43	3.59	4.22	5.47	5.64	6.20

methods can be readily adopted in the existing differential, integral, interpolation, and reconstruction data fitting problems.

Chapter 3

Convergence of Kernel Based Methods

3.1 Introduction

Partial differential equations (PDEs) are widely used to describe physical phenomena in science and engineering. The numerical solution to these systems generally utilize functional approximations that involve interpolations, derivatives or integrals. Finite difference methods are a great example of this concept, numerically discretizing derivative approximations to solve an underlying PDE. When choosing a centered or biased stencil for a finite difference method, it is well known how to solve for the convergence rate and order of accuracy. Extending to higher dimensions, this analysis is more involved but still possible. Generally, one designs a finite difference method such that the Taylor series expansion exactly cancels such that you get an integer convergence.

While finite difference methods remain popular due to their conceptual simplicity and ease of implementation, numerous alternative discretization strategies offer compelling advantages for different problem classes. Spectral methods, for instance,

achieve exponential convergence rates for smooth solutions by representing derivatives through global polynomial or trigonometric basis functions, far surpassing the algebraic convergence of finite differences. Radial basis function methods eliminate the need for structured grids entirely, computing derivatives through differentiation of globally or locally supported basis functions centered at scattered points—particularly valuable for complex geometries or moving boundary problems. Compact schemes [86] provide an elegant middle ground, achieving higher-order accuracy than standard finite differences while maintaining locality through implicit formulations. More exotic approaches like differential quadrature methods generalize the finite difference philosophy by using all available function values (not just nearest neighbors) to approximate derivatives, while discontinuous Galerkin methods combine the geometric flexibility of finite elements with the conservation properties of finite volumes. Another class is gaussian process based methods.

The application of Gaussian processes (GPs) to numerical solution of differential equations has emerged as a promising research direction over the past two decades, offering a probabilistic framework that naturally quantifies uncertainty and adapts to irregular geometries. The foundational work by [59] first demonstrated how GPs could be employed to solve linear PDEs by exploiting the closure of Gaussian processes under linear operations. This principle, thoroughly developed in [123], enables the transformation of differential operators into covariance functions, thereby converting PDE problems into kernel learning tasks. Subsequent developments have significantly expanded the theoretical understanding and practical applicability of GP-based PDE solvers. [108] introduced the concept of “numerical homogenization” through GPs, showing that clas-

sical finite element methods could be interpreted as specific instances of GP regression with particular kernel choices. This unifying perspective revealed deep connections between probabilistic numerics and traditional discretization schemes. The work by [38] further formalized this connection within the framework of Bayesian numerical methods, demonstrating that GP-based solvers produce posterior distributions that encode both the solution estimate and its associated numerical uncertainty.

Convergence analysis of GP methods has received considerable attention, though the theoretical landscape remains less mature than for classical methods. [155] established convergence rates for GP interpolation in Sobolev spaces, providing a rigorous foundation for understanding approximation properties of kernel methods. [33] demonstrated that for elliptic PDEs, GP methods with appropriately chosen kernels achieve optimal convergence rates comparable to finite element methods. The work of [120] on physics-informed neural networks, while not strictly GP-based, shares similar philosophical underpinnings and has influenced the development of physics-informed GP methods. More recently, [16] provided a comprehensive convergence theory for kernel-based collocation methods, establishing connections between kernel smoothness, fill distance, and convergence rates. They showed that under suitable regularity conditions, GP methods can achieve spectral convergence for smooth solutions, matching the performance of global spectral methods while maintaining greater flexibility in handling complex geometries.

Despite these advances, several theoretical challenges remain. The selection of optimal kernel functions and hyperparameters for specific PDEs or operators lacks a complete theoretical framework, often relying on empirical tuning or maximum likeli-

hood estimation. [116] addressed this by developing adaptive kernel selection strategies based on local solution properties, though computational costs remain a concern. The primary aim of this chapter is two fold:

1. Provide a mathematical framework to analyze the convergence of gaussian process based numerical methods.
2. Show the flexibility of kernel based methods on numerical test problems while comparing our theoretical framework.

One aspect that allows GP regression great flexibility is its ability to convert data to any specified type defined as an arbitrary operator. As already investigated (at least partly) in the GP shock-capturing methods mentioned above, GP can convert different data types from one grid location to another in a simpler manner than in the existing approaches [103, 27]. For example, a standard FV polynomial reconstruction scheme would require taking volume-averaged values at cell centers as input to reconstruct pointwise Riemann states as output at cell interface locations in 1D. In multiple dimensions, a quadrature-based integration scheme would need to be used to convert these pointwise estimates to a face integral flux value. A popular way to obtain high-order accurate Riemann states in conventional polynomial-based multidimensional reconstruction schemes is to add transverse corrections for each spatial direction dimension-by-dimension. An example in 3D would include three separate reconstructions, cascading from cell averages in 3D to face averages, face averages to line averages, and finally, line averages to pointwise values and vice versa [103, 27]. Although this process is not particularly difficult, there are many steps that can be computationally

compounded, and one may need to apply boundary conditions on intermediate quantities [101], which results in added data communication in parallel computing.

In GP, however, such data-type conversions can be achieved by applying the needed integral operations directly to the underlying pointwise GP covariance kernels, thanks to the fact that GP is invariant under linear operations [123]. This aspect is the centerpiece of the current study, where we apply GP approximation to a collection of linear operations which are frequently used in modern science and engineering applications. It’s important to note, that while GP numerical methods have been extremely successful, all results were proven using numerical convergence and empirical tuning of hyperparameters. There are currently no tools to analyze theoretical convergence given a stencil shape, hyperparameter, and some linear operator to convert to or from.

3.2 Convergence Analysis

We will now introduce our analysis to estimate the convergence rate of kernel-based numerical methods. We emphasize that this analysis is performed without specifying a particular kernel, stencil configuration, or hyperparameters. This framework is completely general for any kernel-based numerical method and, importantly, for any linear operator transformation as established in Section 2.2.

3.2.1 Taylor Expansion in GP Predictions to Analyze GP’s Effective Rate of Convergence

To systematically analyze GP’s effective order of accuracy, we leverage a Taylor series expansion approach in GP-produced approximations. This method is inspired

by classical error analysis in finite difference methods [87] and extends the framework proposed in [104] to non-standard numerical methods, such as GP-based schemes, where perfect error term cancellation is not guaranteed.

3.2.1.1 General Framework for Linear Operator Predictions

Recall from Eq. (2.6) that GP predictions for arbitrary linear operators take the form

$$p_* = \mathbf{t}_*^T \mathbf{C}^{-1} \cdot \mathbf{f}, \quad (3.1)$$

where $\mathbf{C} = \mathcal{L}_1(K(\mathbf{x}, \mathbf{x}'))$ encodes the data covariance structure (respecting the input data type) and $\mathbf{t}_* = \mathcal{L}_2(K(\mathbf{x}_*, \mathbf{x}'))$ channels these correlations to produce the desired output type. The key observation is that, regardless of the operators $\mathcal{L}_1$ and $\mathcal{L}_2$, this prediction can always be expressed as a weighted sum:

$$p_* = \sum_i w_i f_i, \quad (3.2)$$

where $\mathbf{w} = \mathbf{t}_*^T \mathbf{C}^{-1}$ and f_i represents the input data at location $\mathbf{x}_i$. This weighted sum formulation holds whether we are predicting point values, derivatives, volume averages, or any other linear transformation of the underlying function.

3.2.1.2 Taylor Expansion Analysis

The universality of the weighted sum representation in Eq. (3.2) enables a unified Taylor expansion analysis applicable to all linear operators. For 1D GP predictions on a stencil of size $(2r_{gp} + 1)$, we Taylor expand each input $f(x_i)$ about the prediction

point $x = x_*$:

$$p_* = \sum_{i=1}^{2r_{gp}+1} w_i \left(\sum_{\alpha=0}^k \frac{f^{(\alpha)}(x_*)}{\alpha!} (x_i - x_*)^\alpha \right) + \mathcal{O}(h^{k+1}), \quad (3.3)$$

where h is the grid spacing. Rearranging by collecting terms of equal order yields

$$p_* = \sum_{\alpha=0}^k \frac{f^{(\alpha)}(x_*)}{\alpha!} \left(\sum_{i=1}^{2r_{gp}+1} w_i (x_i - x_*)^\alpha \right) + \mathcal{O}(h^{k+1}). \quad (3.4)$$

This formulation explicitly reveals how the kernel weights w_i —which encode both the kernel choice and the linear operators $\mathcal{L}_1$ and $\mathcal{L}_2$ —determine the coefficients of each Taylor expansion term. The coefficient of the α -th order term is $c_\alpha = \sum_i w_i (x_i - x_*)^\alpha / \alpha!$, and these coefficients directly govern the accuracy of the prediction.

3.2.1.3 Extension to Multiple Dimensions

This strategy extends naturally to GP predictions in 2D and 3D using the multidimensional Taylor expansion for $f : \mathbb{R}^d \rightarrow \mathbb{R}$, $d = 2$ or 3 :

$$f(\mathbf{x}_i) = \sum_{|\boldsymbol{\alpha}|=0}^k \frac{\partial^{|\boldsymbol{\alpha}|} f(\mathbf{x}_*)}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} \frac{\mathbf{h}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} + \mathcal{O}(\|\mathbf{h}\|^{k+1}), \quad (3.5)$$

where $\mathbf{h} = \mathbf{x}_i - \mathbf{x}_*$ and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}_0^d$ is a multi-index with $|\boldsymbol{\alpha}| = \sum_{j=1}^d \alpha_j$.

For predictions involving multidimensional linear operators (e.g., Laplacian ∇^2 , gradient ∇ , or mixed derivatives), the same weighted sum structure in Eq. (3.2) applies where the weights now encode the multidimensional operator through $\mathbf{t}_* = \mathcal{L}_2(K(\mathbf{x}_*, \mathbf{x}'))$. Substituting the Taylor expansion yields

$$p_* = \sum_{|\boldsymbol{\alpha}|=0}^k \frac{\partial^{|\boldsymbol{\alpha}|} f(\mathbf{x}_*)}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} \frac{1}{\boldsymbol{\alpha}!} \left(\sum_i w_i \mathbf{h}_i^{\boldsymbol{\alpha}} \right) + \mathcal{O}(\|\mathbf{h}\|^{k+1}), \quad (3.6)$$

where $\mathbf{h}_i = \mathbf{x}_i - \mathbf{x}_*$. The coefficients $c_{\alpha} = \sum_i w_i \mathbf{h}_i^{\alpha} / \alpha!$ again determine the accuracy, with the analysis framework remaining unchanged regardless of operator complexity or dimension.

3.2.2 Effective Order of Convergence (EOC)

In traditional numerical methods, the formal order of accuracy is determined by the leading-order nonzero term in the Taylor series truncation error. However, in GP-based methods, and kernel-based methods in general, lower-order terms may persist with small but nonzero coefficients due to the lack of perfect error cancellation that occurs in polynomial-based schemes. These residual terms affect the actual observed convergence rate in practice but do not dominate the convergence behavior.

To quantify the overall contribution of different terms across all orders, we introduce the effective order of convergence (EOC) defined as a weighted average:

$$\text{EOC} = \frac{\sum_{\ell=1}^k \ell |c_{\ell}|}{\sum_{\ell=1}^k |c_{\ell}|}, \quad (3.7)$$

where k is the number of terms retained in the Taylor expansion and c_{ℓ} represents the coefficient of the ℓ -th order term. In one dimension, $c_{\ell} = \sum_i w_i (x_i - x_*)^{\ell} / \ell!$. In multiple dimensions with multi-index α , we set $\ell = |\alpha|$ and sum over all multi-indices with $|\alpha| = \ell$.

The EOC metric provides a holistic view of GP accuracy by accounting for all error contributions rather than only the dominant term. It effectively computes the “center of mass” of the error distribution across different orders, weighted by the magnitude of each coefficient. This metric is particularly valuable for GP methods

because:

- It captures the influence of non-negligible lower-order terms that persist due to incomplete cancellation
- It provides a single scalar measure that correlates well with observed convergence rates in numerical experiments
- It applies uniformly across all linear operators, providing consistent accuracy assessment whether predicting function values, derivatives, volume averages, or more complex transformations

We set $k = 20$ for the current study and find it provides sufficient resolution for error analysis across all operators and stencil configurations examined. The independence of this analysis from specific kernel choices or hyperparameters makes EOC a robust tool for understanding GP convergence behavior in diverse applications.

3.3 Choosing a Kernel Function and GP Stencils

Up to this point, we have described GP calculations for an arbitrary operator in an abstract definition without defining a kernel or a stencil. In practice, one must make some choice of kernel, its hyperparameters, and a stencil choice. As mentioned previously, this choice needs to be done using empirical tuning and convergence tests over many combinations of the above. We briefly discuss our options for choosing a GP kernel function and GP local stencil configurations for the following numerical test sections.

3.3.1 Squared Exponential (SE) Kernel

For the remainder of this work, we select the squared exponential (SE) covariance kernel,

$$K(\mathbf{x}, \mathbf{x}') = K_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \Sigma^2 \exp \left[-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2} \right] = \Sigma^2 \exp \left[-\frac{r^2}{2\ell^2} \right], \quad (3.8)$$

which is one of the most widely used GP kernel functions [123, 17] and has been investigated extensively in our previous high-order GP studies. We follow our previous approach to set $\Sigma^2 = 1$ for simplicity, which does not affect the posterior mean function. The second hyperparameter ℓ determines the correlation length scale the GP model prefers. It controls how heavily weighted nearby points are compared to those that are further away in space. There is some flexibility in choosing ℓ while an optimal choice is available by minimizing the negative log-likelihood function with respect to ℓ [123]. In general, a larger ℓ is preferable, but such a choice comes at the cost of making the $\mathbf{K}$ a stiffer matrix and harder to invert.

3.4 Results: Operator-constrained data prediction

We adopt the linear-operator formulation in Section 2.2 and the compact expression $p_* = \mathbf{t}_*^\top \mathbf{C}^{-1} \mathbf{f}$ from Eq. (2.6), where $\mathbf{C} = \mathcal{L}_1(K)$ encodes the input data type and $\mathbf{t}_* = \mathcal{L}_2(k_*)$ channels the desired output. We note that the weights $\mathbf{w}^\top \equiv \mathbf{z}_*^\top = \mathbf{t}_*^\top \mathbf{C}^{-1}$ are equivalent across the entire grid for a stationary kernel and a given r_{gp} and ℓ . Therefore these can be precomputed. Following [126], we compute $\mathbf{w}$ in quadruple precision and cast to double in the evaluation stage. Below we give a unified numerical setup and then present four 2D operators; additional examples appear in [44].

3.4.1 Numerical setup

We measure the GP reconstruction errors on a uniform Cartesian grid over $[0, 1] \times [0, 1]$ with $N_x = N_y \equiv N$ and $h = 1/N$. We vary $N \in \{32, 64, 128, 256\}$ to assess convergence. Local stencils use the blocky–diamond shaped stencil with $r_{\text{gp}} \in \{1, 2, 3\}$. The data is generated using the following input function

$$f(x, y) = e^{-2x} \sin(4\pi y) + x \cos(2\pi x) e^{-y}. \quad (3.9)$$

For each (N, r_{gp}) , the GP is queried at the interface-center (cell corner)

$$(x_\star, y_\star) = (x_i + \frac{dx}{2}, y_i + \frac{dy}{2}). \quad (3.10)$$

The squared-exponential (SE) kernel is used with length scale $\ell = 0.05$ held constant across all resolutions. Errors are reported in a global L^1 norm over the evaluation set, and the observed order is the negative least-squares slope of $\log_{10}(\text{error})$ vs. $\log_{10}(N)$. The theoretical proxy p_{eff} is obtained from the Taylor expansion analysis developed earlier in Equation (3.7) and evaluated at the same $(x_\star, y_\star)$, stencil, and ℓ . In each plot, the line of slope p_{eff} is drawn through the coarsest numerical point for visual comparison.

We consider four linear operators in 2D, grouped into two “CFD-type” reconstruction operators and two derivative operators:

- Point-to-point interpolation (Pt→Pt),
- Volume-average to point reconstruction (VA→Pt),
- Point to first derivative in y (Pt→ ∂_y),

- Volume-average to second derivative in x (VA $\rightarrow \partial_{xx}$).

Each operator corresponds to a specific pair of linear maps $(\mathcal{L}_1, \mathcal{L}_2)$ in the framework of Section 2.2. We present the associated transformed kernels for each.

For the point-to-point operator (Pt $\rightarrow$ Pt), no transformation is applied; both $\mathcal{L}_1$ and $\mathcal{L}_2$ are point-evaluation functionals, so the transformed kernel is simply the original squared-exponential kernel.

$$K^{(\text{Pt to Pt})}(\mathbf{x}, \mathbf{x}') \equiv K_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{(x-x')^2}{2\ell^2}\right] \exp\left[-\frac{(y-y')^2}{2\ell^2}\right], \quad (3.11)$$

$$k_{\star}^{(\text{Pt to Pt})}(\mathbf{x}_{\star}, \mathbf{x}') \equiv K_{\text{SE}}(\mathbf{x}_{\star}, \mathbf{x}').$$

For the volume-average to point operator (VA $\rightarrow$ Pt), $\mathcal{L}_1$ applies a cell-volume integral over both arguments of the kernel while $\mathcal{L}_2$ performs a pointwise evaluation, producing a doubly integrated SE kernel.

$$\begin{aligned} K^{(\text{VA to Pt})}(\mathbf{x}, \mathbf{x}') &\equiv \frac{1}{\Delta x \Delta y} \int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} \int_{y-\frac{\Delta y}{2}}^{y+\frac{\Delta y}{2}} \frac{1}{\Delta x' \Delta y'} \int_{x'-\frac{\Delta x'}{2}}^{x'+\frac{\Delta x'}{2}} \int_{y'-\frac{\Delta y'}{2}}^{y'+\frac{\Delta y'}{2}} K_{\text{SE}}(\mathbf{x}, \mathbf{x}') \\ &= \prod_{d=x,y} \sqrt{\pi} \left(\frac{\ell}{\Delta d} \right)^2 \\ &\quad \times \left\{ \left(\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d + 1}{\sqrt{2}\ell/\Delta d} \operatorname{erf} \left[\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d + 1}{\sqrt{2}\ell/\Delta d} \right] + \frac{\Delta_{\mathbf{x}\mathbf{x}'}^d - 1}{\sqrt{2}\ell/\Delta d} \operatorname{erf} \left[\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d - 1}{\sqrt{2}\ell/\Delta d} \right] \right) \right. \\ &\quad \left. + \frac{1}{\sqrt{\pi}} \left(\exp \left[-\frac{(\Delta_{\mathbf{x}\mathbf{x}'}^d + 1)^2}{2(\ell/\Delta d)^2} \right] + \exp \left[-\frac{(\Delta_{\mathbf{x}\mathbf{x}'}^d - 1)^2}{2(\ell/\Delta d)^2} \right] \right) \right. \\ &\quad \left. - 2 \left(\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d}{\sqrt{2}\ell/\Delta d} \operatorname{erf} \left[\frac{\Delta_{\mathbf{x}\mathbf{x}'}^d}{\sqrt{2}\ell/\Delta d} \right] + \frac{1}{\sqrt{\pi}} \exp \left[-\frac{(\Delta_{\mathbf{x}\mathbf{x}'}^d)^2}{2(\ell/\Delta d)^2} \right] \right) \right\}. \end{aligned} \quad (3.12)$$

$$\begin{aligned} k_{\star}^{(\text{VA to Pt})}(\mathbf{x}_{\star}, \mathbf{x}') &\equiv \frac{1}{\Delta x \Delta y} \int_{x_{\star}-\frac{\Delta x}{2}}^{x_{\star}+\frac{\Delta x}{2}} \int_{y_{\star}-\frac{\Delta y}{2}}^{y_{\star}+\frac{\Delta y}{2}} K_{\text{SE}}((x, y), \mathbf{x}') dy dx \Big|_{\mathbf{x}=\mathbf{x}_{\star}} \\ &= \prod_{d=x,y} \sqrt{\frac{\pi}{2}} \frac{\ell}{\Delta d} \left(\operatorname{erf} \left(\frac{\Delta_{\mathbf{x}_{\star}\mathbf{x}'}^d + 1/2}{\sqrt{2}\ell/\Delta d} \right) - \operatorname{erf} \left(\frac{\Delta_{\mathbf{x}_{\star}\mathbf{x}'}^d - 1/2}{\sqrt{2}\ell/\Delta d} \right) \right). \end{aligned} \quad (3.13)$$

For the point-to-first-derivative operator ($\text{Pt} \rightarrow \partial_y$), $\mathcal{L}_1$ is point evaluation in both kernel arguments, and $\mathcal{L}_2$ applies the first derivative $\partial/\partial y$ to the prediction. In other words, we keep the same covariance kernel as in the $\text{Pt} \rightarrow \text{Pt}$ case and simply apply an additional differential operator to the prediction kernel:

$$K^{(\text{Pt to } \partial y)}(\mathbf{x}, \mathbf{x}') \equiv K^{(\text{Pt to Pt})}(\mathbf{x}, \mathbf{x}'), \quad (3.14)$$

$$k_{\star}^{\text{Pt to } \partial y}(\mathbf{x}_{\star}, \mathbf{x}') \equiv \frac{\partial}{\partial y_{\star}} \left(k_{\star}^{(\text{Pt to Pt})}(\mathbf{x}_{\star}, \mathbf{x}') \right). \quad (3.15)$$

For the volume-average to second-derivative operator ($\text{VA} \rightarrow \partial_{xx}$), $\mathcal{L}_1$ performs cell-volume integration in both kernel arguments, and $\mathcal{L}_2$ applies the second derivative $\partial^2/\partial x^2$ to the prediction. Thus, the covariance over the input data is the same as in the $\text{VA} \rightarrow \text{Pt}$ case, and only the prediction kernel is modified by an additional second derivative:

$$K^{(\text{VA to } \partial x \partial x)}(\mathbf{x}, \mathbf{x}') \equiv K^{(\text{VA to Pt})}(\mathbf{x}, \mathbf{x}'), \quad (3.16)$$

$$k_{\star}^{\text{VA to } \partial x \partial x}(\mathbf{x}_{\star}, \mathbf{x}') \equiv \frac{\partial^2}{\partial x_{\star}^2} \left(k_{\star}^{(\text{VA to Pt})}(\mathbf{x}_{\star}, \mathbf{x}') \right). \quad (3.17)$$

We emphasize the kernel modifications are done symbolically and as a pre-computation step, therefore no additional or high cost is occurred by utilizing these transformed kernels.

Figures 3.1–3.2 compare the measured effective order of convergence against the theoretical prediction p_{eff} for the four operator classes considered. In all figures, the GP stencil radius is indicated by color (red: $r_{\text{gp}} = 1$, blue: $r_{\text{gp}} = 2$, green: $r_{\text{gp}} = 3$), with dashed lines showing the numerically observed slopes and solid lines showing the theoretical p_{eff} obtained from the Taylor expansion analysis. The test cases and parameter choices are selected so that conditioning and floating-point effects are negligible.

For the two CFD-like operators (Pt→Pt and VA→Pt), shown in Figure 3.1, we observe excellent agreement across all stencil radii. The theoretical prediction slightly overestimates the effective order for $r_{\text{gp}} = 1$ and slightly underestimates it for $r_{\text{gp}} = 3$, but the deviations are small and the overall trends match closely.

For the derivative operators (Pt→ ∂_y and VA→ ∂_{xx}), shown in Figure 3.2, the agreement between the measured slopes and the theoretical p_{eff} is noticeably poorer at the smallest stencil radius. For $r_{\text{gp}} = 1$ the theory substantially overpredicts the observed effective order indicating that the discrete stencil does not realize all of the low-order cancellations assumed in the Taylor-based model. As the stencil is widened to $r_{\text{gp}} = 2$ the discrepancy is reduced, and by $r_{\text{gp}} = 3$ the numerical convergence rates for both derivative operators align very closely with the theoretical p_{eff} , mirroring the level of agreement seen for the reconstruction operators in Figure 3.1.

This pattern can be understood directly from a Taylor expansion. Any of our GP-based operators evaluated at a target point $\mathbf{x}_\star$ can be written as a weighted sum of nearby function values,

$$p_\star = \sum_j w_j f(\mathbf{x}_j),$$

where the prediction weights w_j are determined by the kernel and depend on the stencil radius r_{gp} (while the kernel hyperparameters, including the SE length scale ℓ , are kept fixed). For simplicity, consider a one-dimensional grid with spacing h and offsets $\Delta x_j = x_j - x_\star$. A Taylor expansion of f about $x_\star$ gives

$$f(x_j) = f(x_\star) + \Delta x_j f'(x_\star) + \frac{1}{2} \Delta x_j^2 f''(x_\star) + \frac{1}{6} \Delta x_j^3 f^{(3)}(x_\star) + \dots \quad (3.18)$$

Substituting this into the prediction and collecting like terms yields

$$\begin{aligned}
p_\star - f(x_\star) = & \left(\sum_j w_j \Delta x_j \right) f'(x_\star) + \frac{1}{2} \left(\sum_j w_j \Delta x_j^2 \right) f''(x_\star) \\
& + \frac{1}{6} \left(\sum_j w_j \Delta x_j^3 \right) f^{(3)}(x_\star) + \dots .
\end{aligned} \tag{3.19}$$

Writing $\Delta x_j = h \xi_j$ with $|\xi_j| = \mathcal{O}(1)$, we can express the truncation error schematically as

$$p_\star - f(x_\star) = \sum_{p \geq 1} c_p(r_{\text{gp}}) h^p, \tag{3.20}$$

where each coefficient $c_p(r_{\text{gp}})$ is a linear combination of the weights w_j and the dimensionless offsets ξ_j .

Unlike in classical finite-difference schemes, we do not enforce any moment conditions on the weights w_j (such as $\sum_j w_j \Delta x_j = 0$) that would guarantee exact cancellation of low-order terms. For a given kernel and stencil, the coefficients $c_p(r_{\text{gp}})$ are whatever the GP regression produces: some low-order coefficients may be small, some may be comparable to higher-order ones, and there is no a priori reason for strict integer orders to emerge.

The numerically observed convergence rate is governed by the dominant exponent in (3.20) over the range of grid spacings h that we test, which typically behaves like an effective $\mathcal{O}(h^{p_{\text{dom}}})$ law with p_{dom} close to the smallest index p for which $c_p(r_{\text{gp}})$ is non-negligible. In contrast, our theoretical proxy p_{eff} from (3.7) is constructed as a weighted average of the indices p , based on the analytic coefficients $c_p^{(\text{theory})}$ obtained from the SE kernel for the same stencil. If a low-order coefficient such as $c_1(r_{\text{gp}})$ is small but not zero, p_{eff} can be noticeably larger than p_{dom} , because the higher-order terms receive substantial weight in the average.

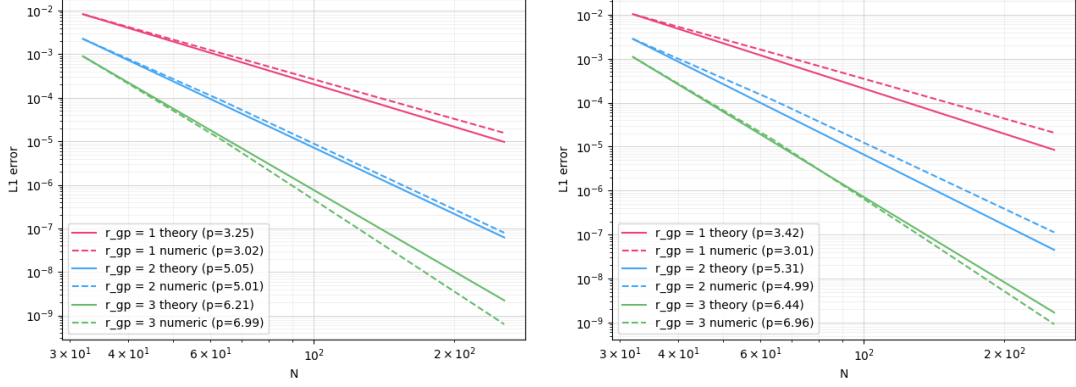


Figure 3.1: Effective order of convergence for the 2D point-to-point (Pt→Pt, left) and volume-average to point (VA→Pt, right) operators. Both use the blocky-diamond stencil and predict in the upper-left corner, $\mathbf{x}_* = (x_i + \frac{1}{2}\Delta x, y_i + \frac{1}{2}\Delta y)$. The colors denote GP stencil radius r_{gp} (red: $r_{\text{gp}} = 1$, blue: $r_{\text{gp}} = 2$, green: $r_{\text{gp}} = 3$). Dashed lines show the observed numerical convergence rates, and solid lines show the theoretical proxy p_{eff} from the Taylor expansion analysis.

This is exactly what we observe for the derivative operators at $r_{\text{gp}} = 1$ in Figure 3.2: the Taylor coefficients predicted by the kernel yield a non-negligible first-order contribution, so the measured convergence behaves much like an $\mathcal{O}(h)$ method, even though the spectrum of coefficients implies an effective index $p_{\text{eff}} \gtrsim 2$. As the stencil radius is increased (e.g. to $r_{\text{gp}} = 3$), the same analysis shows that the low-order coefficients are more strongly suppressed relative to the higher-order ones, and the dominant exponent p_{dom} moves closer to the averaged value p_{eff} . In this regime the slopes from the numerical convergence studies align closely with the theoretical prediction.

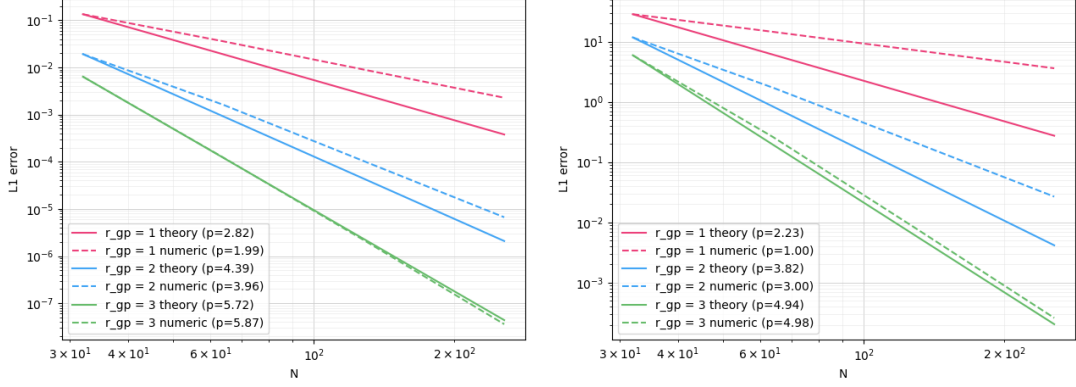


Figure 3.2: Effective order of convergence for the 2D point-to-derivative ($\text{Pt} \rightarrow \partial_y$, left) and volume-average to second derivative ($\text{VA} \rightarrow \partial_{xx}$, right) operators. Both use the blocky-diamond stencil and predict in the upper-left corner, $\mathbf{x}_* = (x_i + \frac{1}{2}\Delta x, y_i + \frac{1}{2}\Delta y)$. Colors indicate stencil radius r_{gp} (red: $r_{\text{gp}} = 1$, blue: $r_{\text{gp}} = 2$, green: $r_{\text{gp}} = 3$), with dashed lines for numerical results and solid lines for the theoretical p_{eff} .

3.5 Results: Choosing Optimal Hyperparameters

Kernel-based numerical methods offer a high degree of flexibility, but this flexibility comes at a cost: there are many hyperparameters whose values strongly influence accuracy. In contrast to traditional finite-volume or finite-difference schemes, where most choices are fixed for you, kernel methods introduce additional degrees of freedom such as the stencil geometry, stencil radius, and kernel length scale. Poor choices can easily degrade accuracy, and in multidimensional settings the space of possible configurations grows rapidly.

In current practice, these hyperparameters are often selected by running a large number of convergence studies on representative meshes, testing various stencils and kernel parameters to determine which combination performs best. This approach

becomes prohibitively expensive in higher spatial dimensions or when multiple operators must be tuned. One of the advantages of our order-of-accuracy (OOA) framework is that it allows these decisions to be made analytically by examining the local Taylor coefficients and the resulting effective order p_{eff} , we can identify favorable hyperparameter choices without performing any numerical convergence sweeps.

In this section, we illustrate this idea through two representative hyperparameter decisions that arise in kernel-based discretizations. First we’ll look at the stencil geometry and second the choice of the SE kernel length scale ℓ . In both cases we show that the OOA analysis predicts the optimal configuration with a few simple algebraic calculations, providing clear guidance that would otherwise require expensive mesh-refinement studies.

3.5.1 Choice of Stencil

Using the effective order of convergence in Eq. (3.7), we now examine the numerical convergence order of different 2D stencils to determine which configuration provides the most effective accuracy. This selection process is unnecessary for 1D GP stencils as their configuration is uniquely determined by the GP radius r_{gp} , and 1D GP predictions follow linear convergence rates as expected.

To demonstrate, we computed EOCs on four different GP stencil configurations, (a) – (d) in Fig. 2.2, for six different GP operations (see Table 3.1) where input vector $\mathbf{f}$ contains cell-centered pointwise values $[\mathbf{f}]_i = f_i$ sampled from each GP stencil. We display the respective results in Table 3.1. Also shown as reference in the third column of Table 3.1 are the so-called ‘anticipated target orders of accuracy’ we would

normally expect in, e.g., 1D conventional finite difference approximations, assuming that the baseline interpolation $\mathbf{f} \rightarrow f(\mathbf{x}_*)$ follows the $(2r_{gp} + 1)$ th convergence rate to begin with. In this sense, we expect that the anticipated order of accuracy should decrease by one from the baseline $(2r_{gp} + 1)$ rate as each differential approximation is added to the approximating operation. In all test cases, we set the GP radius by $r_{gp} = 2$, and the baseline interpolation is expected to be fifth-order accurate. Approximations to partial derivatives are considered at three different locations of $\mathbf{x}_*$, including the central cell location (x_{i_m}, y_{j_m}) , the x interface location $(x_{i_m \pm 1/2}, y_{j_m})$, and the cell corner location $(x_{i_m \pm 1/2}, y_{j_m \pm 1/2})$. We omit the y interface calculations since they are symmetric calculations of the x interface results, so we don't repeat them.

We aim to choose an optimal stencil configuration that can deliver an optimal balance between a higher EOC and a lower amount of stencil points. First, we see that the EOCs on the cross stencil achieve the fifth-order accuracy for the baseline point-to-point interpolation case at the x interface. However, the stencil delivers only the third-order convergence at the corner location. This is expected since the stencil lacks points along the diagonal direction, which carries important information in diagonally-oriented calculations. A similar deficiency is also found in the cross derivative calculation $\partial_{xy}f(\mathbf{x}_*)$ that only achieves an EOC ~ 1.3 , lower than the other grid-aligned counterparts, ∂_{xx} or ∂_{yy} . Nevertheless, EOCs fall short of the anticipated orders on the remaining first and second derivative test cases at the cell corners, which makes us conclude that the most compact stencil is suboptimal and inappropriate.

Second, the results on the diamond stencil prove to be better than those on the cross case in derivative tests, although they fail to show the anticipated fourth- and

third-order rates in some cases, especially on the cross derivative case. Comparing this $\text{EOC} \sim 2.02$ with the cross stencil $\text{EOC} \sim 1.3$, adding the extra four corner cells to the cross configuration generally helps improve the diagonal calculation, except that it is insufficient enough to meet the third-order anticipated accuracy.

In comparison, the EOCs on the blocky diamond and square stencils are shown to produce orders that match the anticipated orders. In fact, we see that the EOCs are higher than the anticipated accuracy in most cases, convincing the validity of the stencil configurations. From this analysis, we select the blocky diamond as our default configuration since it delivers qualitatively equivalent results to the ones on the square stencil while being a bit more compact.

There are a couple of notes we want to discuss. First, on all stencils, the baseline interpolation $\mathbf{f} \rightarrow f(\mathbf{x}_*)$ at the central cell (x_{i_m}, y_{j_m}) is exact due to the property of GP to exactly recover data given to its prior distribution. Second, it is worth pointing out one outlier case $\mathbf{f} \rightarrow \partial_{xy}f(\mathbf{x}_*)$ on the cross stencil, which returns a convergence of exactly 0 due to the perfect cancellations in the leading term. This can be seen in Eq. (2.25) that $(-2x + 2x')$ will always be 0 in the vertical strip of the cross stencil and $(-2y + 2y')$ will always be zero in the horizontal strip, leading to effectively zero contributions from all points on the stencil.

Lastly, we make observations on the convergence of the derivative terms, which is shown to behave differently from the anticipated order. As discussed, we normally expect a drop in order of accuracy for each derivative taken, so we would naively look for at least fourth-order convergence for $\mathbf{f} \rightarrow \partial_x f(\mathbf{x}_*)$ and $\mathbf{f} \rightarrow \partial_y f(\mathbf{x}_*)$. For example, the x interface calculations of the x derivative are equivalent between the diamond

and blocky diamond stencils with EOCs ~ 3.6448 , close to the anticipated fourth-order convergence. Although a similar rate is observed in the y derivative EOC on the diamond stencil (~ 3.6555), it is interesting to see a higher EOC ~ 4.76 is achieved on the blocky diamond for the same problem. Similar higher orders are also found on the blocky diamond on the other second derivative tests involving the y derivative as well. The y direction improvements on the blocky stencil can be seen by looking at the number of cells along the y direction, above and below the x interface location $x_{i\pm 1/2}$. The diamond stencil has three points, while the blocky diamond has five. This added directional contribution to the blocky configuration results in an EOC that is indeed higher than the anticipated fourth order. This reasoning explains the identical EOCs at the corner location $(x_{i_m\pm 1/2}, y_{j_m\pm 1/2})$ on the blocky stencil, e.g., EOCs ~ 5.0371 between $\mathbf{f} \rightarrow \partial_x f(\mathbf{x}_*)$ and $\mathbf{f} \rightarrow \partial_y f(\mathbf{x}_*)$ and the EOCs ~ 4.0972 between $\mathbf{f} \rightarrow \partial_{xx} f(\mathbf{x}_*)$ and $\mathbf{f} \rightarrow \partial_{yy} f(\mathbf{x}_*)$ as the participating cell counts are consistently symmetric in x and y directions. The results of the blocky diamond and the square stencils are quite similar, and they both perform well in meeting the anticipated orders. It may seem a bit strange why some of the EOCs on the square stencil are lower than those on the blocky stencil. We suspect that this counter-intuitive behavior could be related to an overfitting issue, where addition of data points beyond what is necessary may overfit the data and degrade GP predictions.

We now connect these effective-order observations with the OOA-based predictions of p_{eff} . We begin with the same setup as in Section 3.4.1, using $N \in \{16, 32, 64, 128\}$ and fixing the stencil radius at $r_{\text{gp}} = 3$. For each operator, we compare four 2D stencil geometries (cross, diamond, blocky-diamond, and square) at the cell corner. In all plots,

dashed lines denote the measured convergence rates and solid lines denote the theoretical effective order p_{eff} obtained from the Taylor expansion analysis; the four stencil shapes are distinguished by color (cross in red, diamond in blue, blocky-diamond in green, square in purple).

Figure 3.3 show the results for the operators $\text{Pt} \rightarrow \partial_y$ and $\text{VA} \rightarrow \partial_x$. In both cases, the diamond, blocky-diamond, and square stencils achieve nearly identical accuracy, and the agreement between p_{eff} and the measured slopes is very good across all resolutions. The cross stencil behaves differently: it converges at a lower rate and with visibly larger errors, while the theoretical p_{eff} curve lies slightly below the measured slope. This reflects the fact that, at a corner location, the cross stencil samples only along coordinate axes and misses diagonal directions that contribute nontrivially to the mixed Taylor terms; the analytic effective order is therefore more conservative, whereas the particular smooth test function used here benefits from partial cancellation of those neglected contributions, resulting in a somewhat steeper observed slope.

To separate corner geometry effects from stencil sparsity, we also consider the operator $\text{Pt} \rightarrow \partial_x$ evaluated at an x -interface rather than at a corner (Figure 3.4). The OOA analysis predicts the same effective order, $p_{\text{eff}} \approx 3.05$, for all four stencil geometries, reflecting the fact that they share the same local symmetry with respect to the derivative direction. The measured slopes, however, are not identical: the cross stencil converges at roughly 2.6, while the diamond, blocky-diamond, and square stencils all achieve observed orders in the range 4.2–4.3. Thus, in this case the richer stencils slightly outperform the local effective order, whereas the cross stencil underperforms it. This is consistent with p_{eff} acting as a local bias proxy: it correctly identifies that adding

diagonal points does not change the leading Taylor structure, but in this particular smooth test the additional points in the fuller stencils induce favorable cancellations in higher-order terms, steepening the observed numerical slope beyond the nominal p_{eff} .

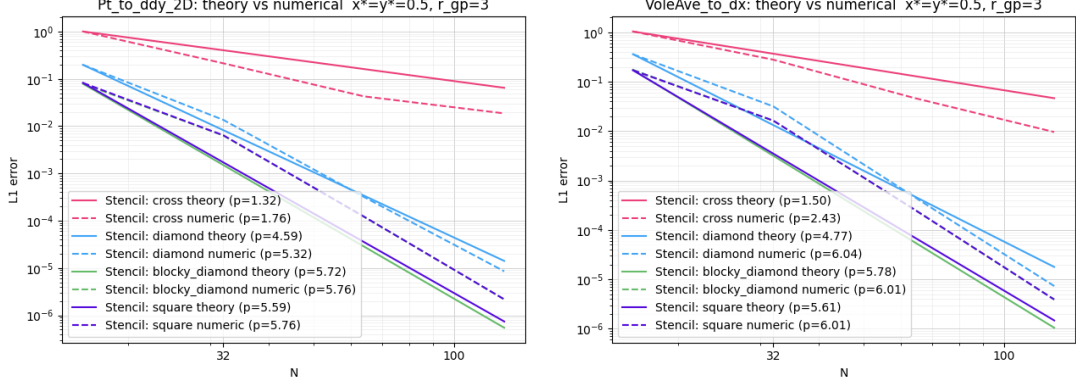


Figure 3.3: Effective order p_{eff} (solid) and measured convergence rates (dashed) for $\text{Pt} \rightarrow \partial_y$ (left) and $\text{VA} \rightarrow \partial_x$ (right) at a corner location with $r_{\text{gp}} = 3$. Colors denote stencil geometry: cross (red), diamond (blue), blocky-diamond (green), and square (purple). The diamond, blocky-diamond, and square stencils exhibit very similar accuracy and closely follow the theoretical prediction, while the cross stencil converges more slowly and is slightly underpredicted by p_{eff} .

To probe stencil geometry more finely, we perform a leave-one-out (LOO) analysis for two operators, $\text{VA} \rightarrow \text{Pt}$ and $\text{VA} \rightarrow \partial_{xy}$, at $r_{\text{gp}} \in \{2, 3\}$. Starting from the full square stencil, we compute p_{eff} with all points included and then recompute it with a single point removed; the difference in effective order is assigned to that point. The resulting heatmaps, shown in Figures 3.5–3.6, reveal the marginal importance of each stencil point for the two operators.

For $\text{VA} \rightarrow \partial_{xy}$, the most influential points form a pronounced blocky-diamond

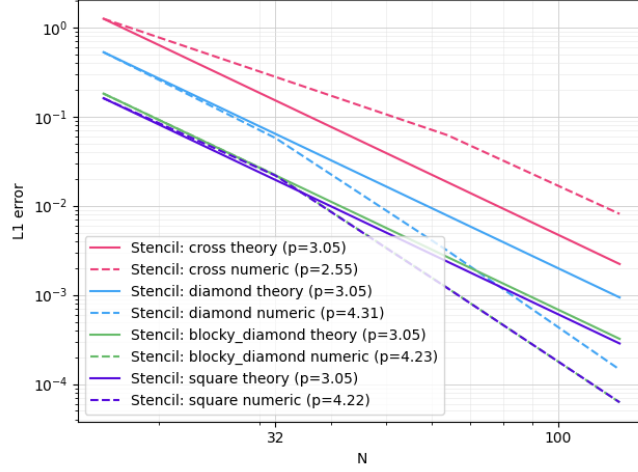


Figure 3.4: Effective order p_{eff} (solid) and measured convergence rates (dashed) for $\text{Pt} \rightarrow \partial_x$ evaluated at an x -interface for $r_{\text{gp}} = 3$. All four stencil geometries share the same predicted effective order, and the numerical slopes agree closely, with the square stencil attaining the smallest absolute error and the cross stencil lagging in magnitude while retaining the same order.

pattern aligned with the target corner. At $r_{\text{gp}} = 3$, the point closest to the predicted corner changes p_{eff} by roughly 6×10^{-1} , while the far diagonal corners contribute on the order of 10^{-2} . This suggests that a reduced blocky-diamond stencil, which retains the high-impact points and omits the far corners, should achieve nearly the same effective order as the full square stencil at lower cost. In contrast, for $\text{VA} \rightarrow \text{Pt}$ the high-impact region is more nearly diamond-shaped, and the optimal reduced stencil may differ; nonetheless, in both cases the LOO maps identify a small set of dominant points and show that many outer-corner points have very low marginal contribution to p_{eff} .

To quantify how many Taylor terms are needed in practice for the order-of-accuracy analysis, we perform a term-sweep for the $\text{VA} \rightarrow \partial_{xx}$ operator with $r_{\text{gp}} = 3$

and evaluate p_{eff} as a function of the maximum Taylor order N_T retained in the series. Figure 3.7 shows that the predicted effective order quickly approaches the measured value (approximately five) as N_T increases. Using only a small number of terms substantially underestimates the observed order, but by $N_T = 15$ the discrepancy is already very small, and by $N_T = 20$ the theoretical p_{eff} has essentially converged. Including additional terms up to $N_T = 30$ yields only marginal improvements.

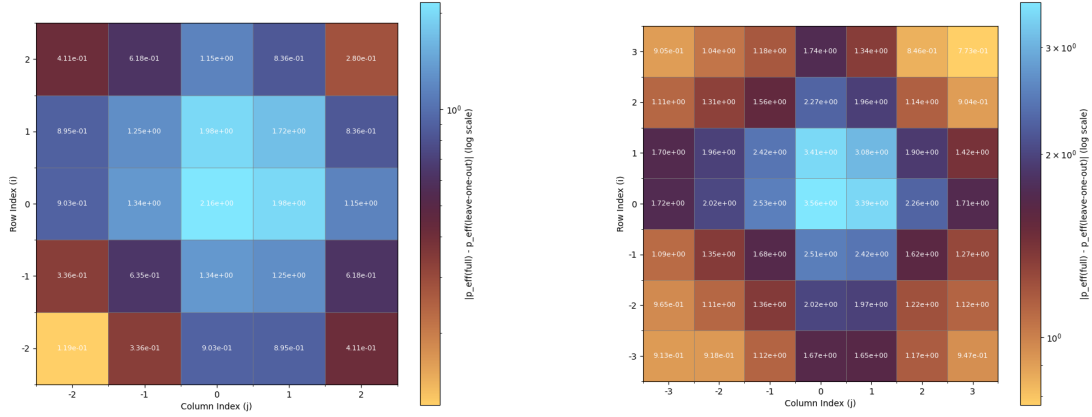


Figure 3.5: Leave-one-out change in p_{eff} for VA→Pt on the square stencil. Left: $r_{\text{gp}} = 2$. Each cell shows the decrease in effective order when that stencil point is removed, revealing which locations contribute most strongly to the reconstruction accuracy. Right: $r_{\text{gp}} = 3$. The dominant contributions remain clustered in a diamond-like region around the target corner, while far-corner points have very small marginal impact on p_{eff} .

3.5.2 Choice of Kernel Length Scale ℓ

We now examine the effect of the SE kernel length scale ℓ . Unlike classical polynomial schemes, where accuracy is determined almost entirely by the stencil, the behavior of GP-based operators depends very sensitively on the ratio ℓ/h . As we will see

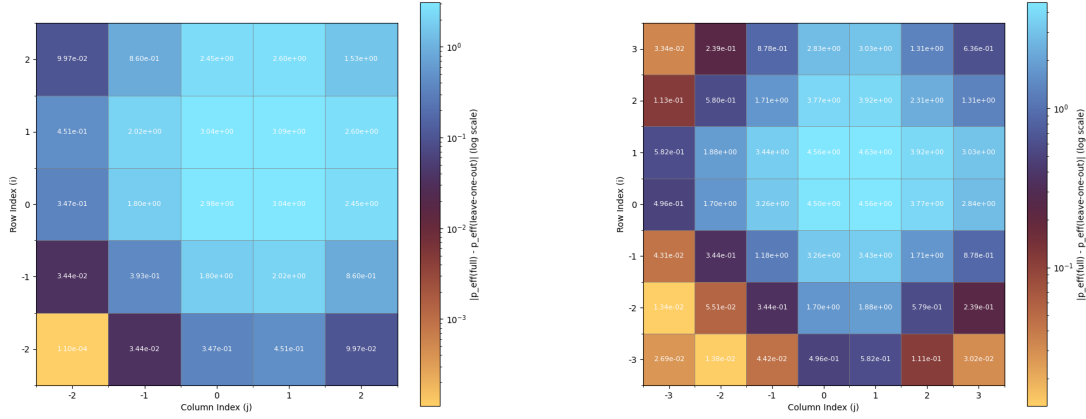


Figure 3.6: Leave-one-out change in p_{eff} for $VA \rightarrow \partial_{xy}$ on the square stencil. Left: $r_{gp} = 2$, where the most influential points already hint at a blocky–diamond structure aligned with the predicted corner location. Right: $r_{gp} = 3$, where the blocky–diamond region becomes even more pronounced, with the point nearest the predicted corner changing p_{eff} by $\mathcal{O}(10^{-1})$ while the far corners contribute at the level of $\mathcal{O}(10^{-2})$, motivating the blocky–diamond stencil as a reduced geometry that preserves almost all of the attainable accuracy.

below, different choices of ℓ/h can produce dramatically different convergence rates and error levels. In typical practice this forces one to run a hyperparameter grid search (or full convergence study) over ℓ to find a usable configuration, which becomes increasingly expensive in multiple dimensions. Our OOA framework provides an alternative: it predicts favorable ranges of ℓ/h directly from the local Taylor coefficients, without any parameter sweeps.

Figure 3.8 illustrates this sensitivity for the $VA \rightarrow Pt$ operator evaluated at the upper-right cell corner using the blocky–diamond stencil with $r_{gp} = 2$. The dual-axis plot shows the theoretical effective order $p_{eff}(\ell/h)$ (left axis) and the measured L^1 error

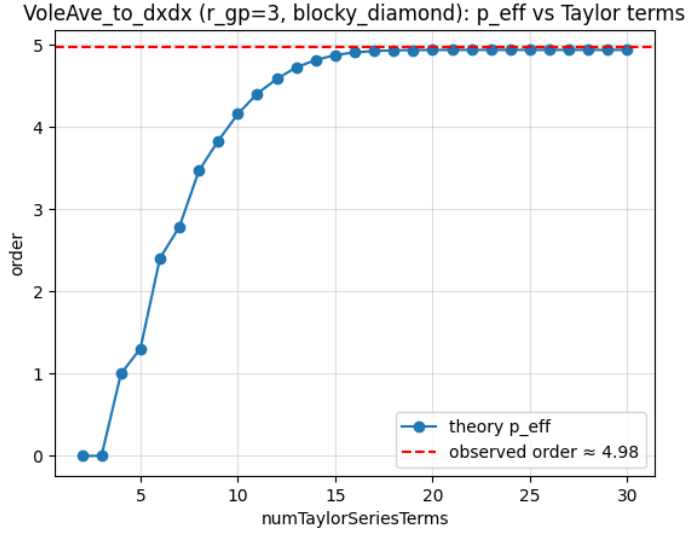


Figure 3.7: Effective order proxy p_{eff} for the $\text{VA} \rightarrow \partial_{xx}$ operator with $r_{\text{gp}} = 3$ as a function of the maximum Taylor order N_T retained in the series. The curve rapidly approaches the measured convergence rate (approximately fifth order) as N_T increases. By $N_T = 15$ the predicted order is already very close to the observed value, and by $N_T = 20$ it is essentially converged; adding further terms up to $N_T = 30$ yields only diminishing improvements.

at the finest resolution $N = 256$ (right axis). The agreement between the two curves is notable: the value of ℓ/h that maximizes the predicted effective order is exactly the value that minimizes the numerical error. In this example, the optimal choice occurs at $\ell/h = 8$ where the theoretical p_{eff} reaches its maximum and the L^1 error reaches its global minimum.

This result highlights two essential points. First, the OOA framework reliably predicts the optimal hyperparameter regime without requiring full convergence studies. Second, the sharp variation of both p_{eff} and the error with respect to ℓ/h underscores the sensitivity of GP-based reconstructions: suboptimal choices of ℓ can reduce accuracy by more than an order of magnitude. The ability of the OOA analysis to identify the optimal value cheaply makes it a practical tool for kernel based solver design.

3.6 Results: Numerical PDEs

The preceding sections focused on local GP-based differential operators in isolation. Given a stencil, kernel, and length scale ℓ , the OOA framework assigns an effective order of accuracy p_{eff} to the resulting discrete operator that approximates a particular derivative or reconstruction. In actual use, however, these operators appear as building blocks inside time-dependent PDE solvers, where they are applied to form the spatial part of the right-hand side.

To keep the discussion focused on accuracy rather than shock-capturing or limiting, we restrict attention to smooth, linear parabolic problems such as diffusion and heat equations. In all such cases, the PDE can be viewed as a time derivative

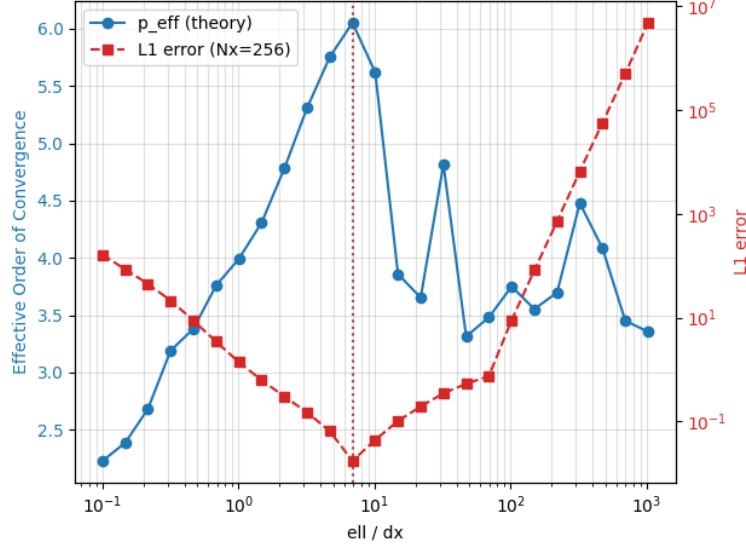


Figure 3.8: Effect of the SE kernel length scale on the VA→Pt operator using the blocky–diamond stencil with $r_{\text{gp}} = 2$, evaluated at the corner interface. The solid curve (left axis) shows the predicted effective order $p_{\text{eff}}(\ell/h)$; the dashed curve (right axis) shows the measured L^1 error at $N = 256$. The optimal ratio $\ell/h \approx 8$ simultaneously maximizes p_{eff} and minimizes the numerical error, demonstrating that the OOA framework provides an accurate and inexpensive predictor of optimal hyperparameters.

equal to a spatial operator applied to u . On a grid, that spatial operator is replaced by a discrete GP-based operator constructed from kernel derivatives and nearby function values. This discrete operator is no different than other operators we have looked at in earlier sections. Therefore, the predicted effective order p_{eff} provides a direct prediction for the spatial order of the semi-discrete PDE right-hand side.

In the remainder of this section we illustrate this connection on a representative parabolic test cases. We consider a one-dimensional diffusion equation, where the GP operator directly approximates the diffusive flux divergence.

3.6.1 1D Diffusion

We begin with the one-dimensional diffusion equation

$$u_t = \mu u_{xx}, \quad x \in [0, 1], \quad (3.21)$$

with periodic boundary conditions, viscosity $\mu = 0.2$, and initial condition

$$u(x, 0) = \sin(2\pi x). \quad (3.22)$$

The exact solution is

$$u(x, t) = \exp(-\mu(2\pi)^2 t) \sin(2\pi x), \quad (3.23)$$

which we use to compute errors at the final time.

The spatial part of the PDE is the diffusive operator

$$\mathcal{L}u = \mu u_{xx} = \frac{\partial}{\partial x}(-\mu u_x) = F_x, \quad F = -\mu u_x, \quad (3.24)$$

so the equation can be written in flux form as $u_t = F_x$. The GP-based discretization approximates this operator directly. We use a stationary squared-exponential kernel and centered one-dimensional stencils of radius $r_{\text{gp}} \in \{1, 2, 3\}$ on grids with $N_x \in \{16, 32, 64, 128\}$. For each grid point x we construct a diffusion kernel by applying the second-derivative operator to the prediction kernel $k_\star$,

$$k_\star^{\text{diff}}(x_\star, x) = \mu \frac{\partial^2}{\partial x^2} k_\star(x_\star, x),$$

and use this to form the GP posterior mean of $u_{xx}(x)$. This defines a discrete GP operator $\mathcal{D}_{xx}^{\text{GP}}$ and the semi-discrete right-hand side

$$\left. \frac{du}{dt} \right|_x = \mu u_{xx}(x) \approx (\mathcal{D}_{xx}^{\text{GP}} u)(x). \quad (3.25)$$

This is algebraically equivalent to first forming the flux $F = -\mu u_x$ with a one-derivative kernel and then differentiating once more, but using the diffusion kernel $k_\star^{\text{diff}}$ is cheaper and simpler for this experiment. The kernel length scale ℓ is held fixed across all grids, with $\ell = 0.05$ as before.

Time integration uses first-order forward Euler. To ensure that temporal error does not limit the observed spatial convergence, we follow [85] and scale the time step with the grid spacing h according to

$$\Delta t \propto h^{q+1}, \quad q = 2r_{\text{gp}}, \quad (3.26)$$

so that the $\mathcal{O}(\Delta t)$ time-discretization error is higher order than the spatial error predicted by the OOA analysis of the GP diffusion operator $\mathcal{D}_{xx}^{\text{GP}}$. In practice we take

$$\Delta t = \Delta t_{\text{base}} \left(\frac{h}{h_{\text{base}}} \right)^{q+1},$$

with a very small base step Δt_{base} and march for a fixed number of steps to a short final time t . At the final time, we evaluate the exact solution on the prediction points and report the relative L_2 error.

Figure 3.9 shows these errors versus N_x on a log-log scale for $r_{\text{gp}} = 1, 2, 3$, together with reference lines whose slopes are the theoretical orders p_{theory} obtained from the GP-Taylor analysis of the diffusion operator $\mathcal{D}_{xx}^{\text{GP}}$. The measured convergence rates closely follow the predicted slopes for all three stencil radii, confirming that the OOA framework accurately characterizes the spatial order of the discrete operator $\mathcal{L}_h$ and, consequently, the semi-discrete PDE right-hand side for this GP-based diffusion solver.

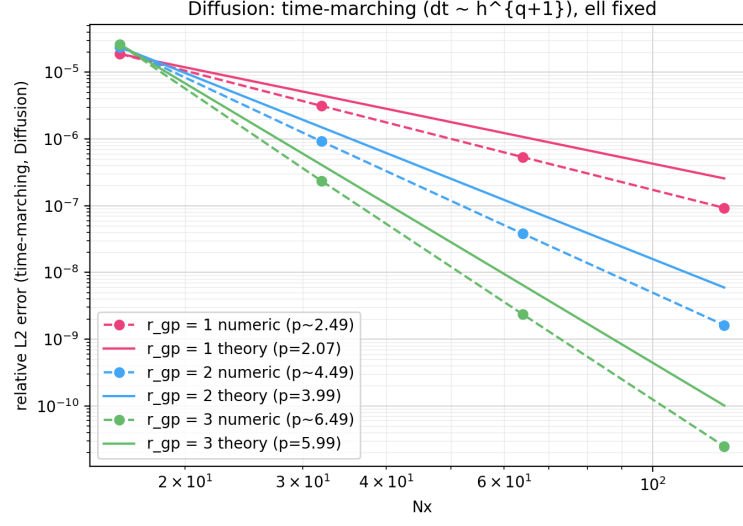


Figure 3.9: Relative L_2 error for the one-dimensional diffusion equation $u_t = \mu u_{xx}$ with periodic boundary conditions and initial condition $u(x, 0) = \sin(2\pi x)$. The spatial operator is constructed from a GP with an SE kernel and centered stencils of radius $r_{\text{gp}} \in \{1, 2, 3\}$. Solid lines show the measured errors; dashed lines indicate the predicted convergence slopes p_{theory} obtained from the GP–Taylor analysis of the diffusion operator $\mathcal{D}_{xx}^{\text{GP}}$. The close agreement demonstrates that the OOA framework correctly predicts the spatial order of the discrete operator and the resulting PDE right-hand side.

3.7 Conclusion

This work attempts to build a systematic theory of accuracy and robustness for kernel-based numerical methods in complex multiphysics applications. We developed a unified convergence analysis for kernel-based numerical methods constructed from Gaussian process regression. By expressing general linear-operator predictions in the compact form $p_* = \mathbf{t}_*^\top \mathbf{C}^{-1} \mathbf{f}$, we showed that all such schemes can be interpreted as weighted sums over local data, independent of the specific kernel or stencil details. A multidimensional Taylor expansion of this weighted sum leads to closed-form expressions for the truncation-error coefficients, from which we defined an effective order-of-accuracy proxy p_{eff} that aggregates contributions from all retained Taylor terms. This metric captures the impact of imperfect cancellation of lower-order terms that is characteristic of kernel-based schemes and correlates closely with observed numerical convergence rates.

Using this operator-oriented framework, we systematically compared several GP stencil geometries and operator types in two spatial dimensions. For a fixed kernel and hyperparameters, the predicted effective order p_{eff} accurately anticipated the measured convergence slopes for interpolation, reconstruction, and differential operators. The analysis also highlighted the role of stencil geometry: reduced blocky–diamond stencils preserve nearly all of the attainable accuracy of larger square stencils while offering a smaller footprint. We further examined the sensitivity of kernel-based operators to the squared-exponential length scale, demonstrating that the optimal ratio ℓ/h can be identified directly from $p_{\text{eff}}(\ell/h)$ without resorting to exhaustive convergence studies.

In our tests, the value of ℓ/h that maximizes p_{eff} also minimizes the numerical error.

Finally, we embedded the GP-based diffusion operator into a time-dependent solver for smooth parabolic problems and verified that the end-to-end convergence rate of the PDE solution follows the ordering predicted by the local operator analysis. This confirms that the effective-order framework extends naturally from isolated operators to full GP-based PDE solvers.

The tools developed here provide a practical pathway for designing and tuning kernel-based discretizations: given a stencil, kernel, and candidate hyperparameters, one can predict the attainable accuracy and identify promising configurations before running any large-scale simulations. Importantly, the framework was developed without assumption of the operator or kernel so this can be applied to a much wider class of problems than shown here.

Table 3.1: Anticipated target orders (third column) and GP's effective order of convergence (EOC in Eq. (3.7)) for six data approximations at $\mathbf{x}_*$ (fourth – sixth columns) on four different GP stencils with $r_{gp} = 2$. EOCs are calculated at (i) the central cell in the fourth column, (ii) the x interfaces of the central cell in the fifth column, and (iii) the cell corners of the central cell in the sixth column.

GP stencil	Operation	Anticipated Order	EOC at (x_{i_m}, y_{j_m})	EOC at $(x_{i_m \pm 1/2}, y_{j_m})$	EOC at $(x_{i_m \pm \frac{1}{2}}, y_{j_m \pm \frac{1}{2}})$
Cross (Fig. 2.2(a))	$\mathbf{f} \rightarrow f(\mathbf{x}_*)$	5	Exact	5.2174	3.0330
	$\mathbf{f} \rightarrow \partial_x f(\mathbf{x}_*)$	4	3.8093	3.6448	1.3237
	$\mathbf{f} \rightarrow \partial_y f(\mathbf{x}_*)$	4	3.8093	1.6109	1.3237
	$\mathbf{f} \rightarrow \partial_{xx} f(\mathbf{x}_*)$	3	3.5950	3.2457	2.1973
	$\mathbf{f} \rightarrow \partial_{xy} f(\mathbf{x}_*)$	3	0.0000	1.9361	1.3040
	$\mathbf{f} \rightarrow \partial_{yy} f(\mathbf{x}_*)$	3	3.5950	1.3944	2.1973
Diamond (Fig. 2.2(b))	$\mathbf{f} \rightarrow f(\mathbf{x}_*)$	5	Exact	5.2174	5.1315
	$\mathbf{f} \rightarrow \partial_x f(\mathbf{x}_*)$	4	3.8093	3.6448	3.4121
	$\mathbf{f} \rightarrow \partial_y f(\mathbf{x}_*)$	4	3.8093	3.6555	3.4121
	$\mathbf{f} \rightarrow \partial_{xx} f(\mathbf{x}_*)$	3	3.5950	3.2457	3.2984
	$\mathbf{f} \rightarrow \partial_{xy} f(\mathbf{x}_*)$	3	2.2689	2.1356	2.0204
	$\mathbf{f} \rightarrow \partial_{yy} f(\mathbf{x}_*)$	3	3.5950	3.4222	3.2984
Blocky Diamond (Fig. 2.2(c))	$\mathbf{f} \rightarrow f(\mathbf{x}_*)$	5	Exact	5.2171	5.7126
	$\mathbf{f} \rightarrow \partial_x f(\mathbf{x}_*)$	4	3.8093	3.6448	5.0371
	$\mathbf{f} \rightarrow \partial_y f(\mathbf{x}_*)$	4	3.8093	4.7600	5.0371
	$\mathbf{f} \rightarrow \partial_{xx} f(\mathbf{x}_*)$	3	3.5950	3.2457	4.0972
	$\mathbf{f} \rightarrow \partial_{xy} f(\mathbf{x}_*)$	3	4.0719	3.8882	3.7237
	$\mathbf{f} \rightarrow \partial_{yy} f(\mathbf{x}_*)$	3	3.5950	4.8469	4.0972
Square (Fig. 2.2(d))	$\mathbf{f} \rightarrow f(\mathbf{x}_*)$	5	Exact	4.9229	4.9844
	$\mathbf{f} \rightarrow \partial_x f(\mathbf{x}_*)$	4	3.8121	3.7314	4.9225
	$\mathbf{f} \rightarrow \partial_y f(\mathbf{x}_*)$	4	3.8090	4.3776	4.9481
	$\mathbf{f} \rightarrow \partial_{xx} f(\mathbf{x}_*)$	3	3.6626	3.2551	3.5499
	$\mathbf{f} \rightarrow \partial_{xy} f(\mathbf{x}_*)$	3	3.9009	3.8126	3.7697
	$\mathbf{f} \rightarrow \partial_{yy} f(\mathbf{x}_*)$	3	3.6375	4.6015	3.5523

Chapter 4

Shock Capturing in Gaussian Process

Methods

4.1 DAS Kernel for Non-Oscillatory Approximation to Discontinuous Functions

Kernel selection is a crucial step in obtaining accurate and reliable outputs in GP models. As we have discussed so far for continuous function approximations, we can capture the underlying patterns and structures of target functions within the data by carefully choosing appropriate kernel functions. This, in turn, ensures that our model is capable of making accurate predictions and generalizing them well to unseen data points. The kernel in GP models not only serves as a mathematical function but also establishes a GP prior that expresses our assumptions about the underlying relationship in the data. By specifying a particular kernel, we are making a strong assumption about the smoothness, periodicity, and correlation structure of the function

that is being modeled. Thus, the choice of a kernel directly impacts the resulting GP model’s performance, and different types of data require different kernel choices.

In this section, we further extend our kernel investigation beyond smooth function approximations and explore appropriate kernel selection for discontinuous functions. Discrete approximation to discontinuous function data is a major research topic in numerical modeling, where the main emphasis is dedicated to producing non-oscillatory solutions across sharp gradients. To demonstrate, let us consider a test problem of reconstructing pointwise data at a shock interface. We will analyze performances of three different kernels, including K_{SE} in Eq. (3.8), the so-called neural network kernel K_{NN} [123] given by

$$K_{NN}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1} \left(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}'}}{\sqrt{(1 + 2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1 + 2\tilde{\mathbf{x}'}^\top \Sigma \tilde{\mathbf{x}'})}} \right), \quad (4.1)$$

and lastly our new “Discontinuous ArcSin kernel” (or the DAS kernel for short) K_{DAS} that takes of the form of

$$K_{DAS}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1} \left(\frac{\exp \left(-\sqrt{(\mathbf{x} - \mathbf{x}')^\top (\mathbf{x} - \mathbf{x}')} \right)}{\prod_{i=1}^d [1 + 2(1 + (x_i - x'_i)^2)]} \right). \quad (4.2)$$

In Eq. (4.1), $\tilde{\mathbf{x}} = (1, \mathbf{x})^T = (1, x_1, \dots, x_d)^\top$ is an augmented vector of $\mathbf{x}$ with d being the number of spatial dimensions, which must also be the size of $\mathbf{x}$ and $\mathbf{x}'$. We also set $\Sigma = \text{diag}(\sigma_0^2, \sigma^2)$ following [123]. The augmentation with unity prevents the kernel from failing to be symmetric semi-positive definite—a requirement for GP kernel functions—when $\mathbf{x} = \mathbf{x}' = \mathbf{0}$ especially in the diagonal entries, keeping its eigenvalues to be non-negative. The functions sampled from this NN kernel can be viewed as superpositions of error functions, $\text{erf}(u_0 + ux)$, where σ_0^2 controls the variance of u_0 while σ^2 controls

u , in which case they respectively influence how much these sample functions are offset from the origin and their scaling on the x -axis [123].

Below, we discuss how these three kernels are compared on data predictions across sharp discontinuities, particularly emphasizing the improved non-oscillatory shock-capturing abilities of the new DAS kernel. First, we remark that the SE and DAS kernels are a stationary kernel, which means it only depends on the ‘distance’ between observed input data points at $\mathbf{x}$ and $\mathbf{x}'$ rather than the actual values of them. That is, SE and DAS are invariant to translations in the input space. This is computationally efficient because K^{-1} only needs to be computed once, saved, and reused over computational steps as long as the computational grid configuration remains static. This is not the case for the NN kernel. It is non-stationary; hence its inverse matrix must to be computed at each input location. The non-stationary property of K_{NN} makes it computationally challenging even for small problems, not to mention big problems with increasing grid resolutions.

The above observations motivate us to design K_{DAS} to be a stationary kernel. In Fig. 4.1, we show the kernel heat maps. The way to read this map is by considering each point $\mathbf{x} = (x, y)$ as an input to the kernel. The color scale represents the value of the kernel: the higher the value, the larger the weight or importance the GP will place on that point. Noticeably, we see that K_{DAS} takes the maximum value along a very thin diagonal band. The consequence is K_{DAS} ’s strong prediction between the data points separated by, namely, the bandwidth, δ_b , where it is seen that $\delta_b \sim \mathcal{O}(\Delta m)$ with $\Delta m = \max_{d=x,y} \{\Delta d\}$. On the other hand, if any data points are separated by a distance larger than δ_b , i.e., 10 or more cells away from each other, the correlation of

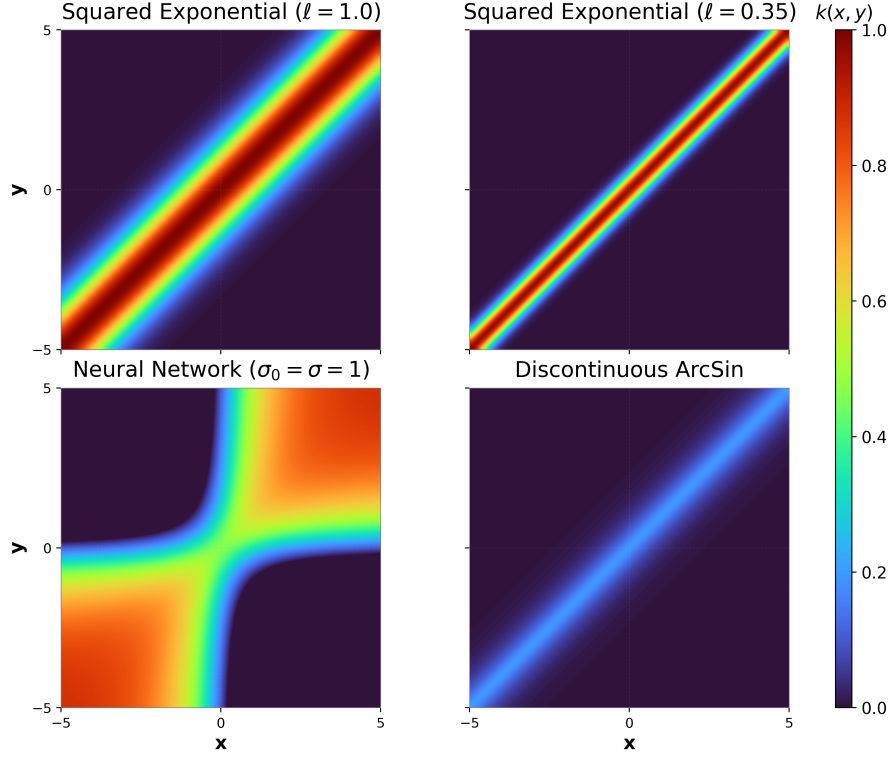


Figure 4.1: Heat map comparison of kernels. (Top left) SE kernel with $\ell = 1.0$. (Top right) SE kernel with $\ell = 0.35$. (Bottom left) NN kernel with $\sigma = \sigma_0 = 1$. (Bottom right) DAS kernel.

them rapidly drops with the increasing inter-distance and K_{DAS} 's predictions become insignificant. By design, henceforth, K_{DAS} is apt to work very well for narrow fluid structures such as shocks or discontinuities. Also importantly, for the same data values at different grid locations, K_{DAS} 's stationary property returns the same kernel value as long as the distance between the data is the same. This is an important consistency concern for numerical methods.

Contrary to the well-confined thin-band structure in the stationary K_{DAS} kernel, the non-stationary effects in the NN kernel are evident in the heat map with an

increased area of high kernel values. With K_{NN} , GP correlations depend on the actual values of $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$, thereby the heat map appears to be widely spread out across the large area of the entire domain. It is true that the NN kernel can be tuned to produce a more confined, narrow band-like structure by increasing σ from the current value of 1 in Fig. 4.1. Nevertheless, one still faces the non-stationary effects, including the need for inverting the kernel at each input location of $\mathbf{x}$ and $\mathbf{x}'$. We present K_{NN} 's heat map using $\sigma_0 = \sigma = 1$ to visualize the non-stationary effects in a clear way.

Table 4.1: Grid convergence results for the 1D baseline point-to-point interpolation using the NN kernel in Eq. (4.1). The NN prediction takes cell center function values $f(x_i)$ as input for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$ and computes $f(x_{i+1/2})$ using $\sigma = \sigma_0 = 1$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	2.05e-02	2.41e-02	4.56e-02	5.06e-03	5.90e-03	1.12e-02	1.16e-03	1.33e-03	2.39e-03
32	2.60e-03	2.97e-03	5.20e-03	1.59e-04	1.80e-04	3.25e-04	8.83e-06	9.72e-06	1.65e-05
64	3.27e-04	3.76e-04	7.40e-04	4.83e-06	5.55e-06	1.05e-05	6.34e-08	7.17e-08	1.26e-07
128	4.04e-05	4.65e-05	9.83e-05	1.49e-07	1.71e-07	3.32e-07	4.77e-10	5.37e-10	9.41e-10
256	5.01e-06	5.77e-06	1.25e-05	4.59e-09	5.26e-09	1.06e-08	4.35e-12	5.22e-12	1.51e-11
Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	—	—	—	—	—	—	—	—	—
32	2.975	3.019	3.133	4.995	5.034	5.108	7.042	7.091	7.178
64	2.993	2.981	2.812	5.038	5.020	4.955	7.122	7.083	7.040
128	3.017	3.014	2.913	5.014	5.023	4.978	7.055	7.059	7.061
256	3.013	3.011	2.975	5.025	5.021	4.972	6.777	6.685	5.958

First, we examine the convergence of the three kernels on smooth data. For the

Table 4.2: Grid convergence results for the 1D baseline point-to-point interpolation using the DAS kernel in Eq. (4.2). The DAS prediction takes cell center function values $f(x_i)$ as input for $f(x) = e^{-x} \sin(4\pi x) \cos(2\pi x)$ on $[0, 1]$ and computes $f(x_{i+1/2})$.

Grid Res.	$r_{gp} = 1$			$r_{gp} = 2$			$r_{gp} = 3$		
	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error	L_1 error	L_2 error	L_∞ error
16	3.56e-02	4.24e-02	8.69e-02	3.53e-02	4.20e-02	8.60e-02	3.52e-02	4.20e-02	8.58e-02
32	9.19e-03	1.08e-02	2.29e-02	9.15e-03	1.07e-02	2.28e-02	9.11e-03	1.07e-02	2.27e-02
64	2.26e-03	2.65e-03	5.58e-03	2.26e-03	2.65e-03	5.58e-03	2.26e-03	2.64e-03	5.57e-03
128	5.61e-04	6.54e-04	1.38e-03	5.61e-04	6.54e-04	1.38e-03	5.60e-04	6.53e-04	1.38e-03
256	1.39e-04	1.62e-04	3.43e-04	1.39e-04	1.62e-04	3.43e-04	1.39e-04	1.62e-04	3.43e-04
	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC	L_1 EOC	L_2 EOC	L_∞ EOC
16	–	–	–	–	–	–	–	–	–
32	1.953	1.977	1.924	1.947	1.970	1.914	1.952	1.977	1.916
64	2.022	2.024	2.037	2.017	2.019	2.033	2.013	2.015	2.030
128	2.013	2.018	2.015	2.012	2.017	2.014	2.010	2.015	2.012
256	2.009	2.010	2.011	2.009	2.010	2.010	2.008	2.009	2.010

baseline pointwise interpolation, both K_{SE} and K_{NN} feature a rate of convergence order $(2r_{gp} + 1)$ on smooth data as shown in Tables 2.2 and 2.3 and Table 4.1, respectively. The K_{DAS} lags behind significantly in this regard and demonstrates a fixed second order convergence rate regardless of stencil sizes as shown in Table 4.2. Despite its inferior convergence rate on smooth flows, the true value of the DAS kernel is to be emphasized and highlighted in relation to discontinuous flow problems. In Figs. 4.2 and 4.3, we present how these three kernels perform in approximating data predictions across shocks and discontinuities for interpolation.

In the first four panels of Fig. 4.2 (from left to right and top to bottom), we

see that K_{NN} performs better than K_{SE} showing a decreased amount of oscillations at the discontinuity. Nonetheless, the shock oscillations persist in both kernels for all three choices of GP radius, $r_{gp} = 1, 2, 3$, and the level of oscillations is not seen to be controlled by varying the size of the GP stencil, i.e., irrelevant to modeling accuracy of the two kernels. As discussed, K_{NN} has two tunable parameters, σ and σ_0 . We find that increasing σ helps improve its performance for shock capturing but does not completely remove shock oscillations.

In contrast, our new K_{DAS} kernel in the bottom two panels shows very promising results with no grid-scale oscillations at the jump interface. The DAS kernel’s second-order insensitivity to r_{gp} is also observed on the bottom right panel, where all three calculations with different radii behave nearly the same way.

In Fig. 4.3, the shock-capturing performance of the three kernels is further compared in two additional compound wave configurations. Again, the DAS kernel outperforms the NN and SE kernels at the discontinuities, displaying no observable oscillations, while all three kernels behave equally well on the smooth parts. The results in Figs. 4.2 and 4.3 suggest that K_{DAS} can be used in a shock-capturing code selectively at shocks and discontinuities within a similar framework of GP-MOOD [20] as part of hybridizing non-oscillatory stable second-order GP solutions with DAS near shocks/discontinuities while high-order GP solutions with SE elsewhere. This topic will be investigated in our future work.

In Fig. 4.4, we compare the condition number of the covariance kernel matrices using the three kernel functions, K_{SE} , K_{NN} , and K_{DAS} . On the left, we examine the condition number of each kernel matrix as a function of grid spacing Δx across different

stencil sizes. On the right, we compare how the condition number varies with Δx for different values of the hyperparameters, e.g., ℓ for the SE kernel and σ for the NN kernel. Our analysis reveals that the choices of ℓ for K_{SE} and σ for K_{NN} dramatically impact the condition number, with the potential for extremely high values. It is always desirable to keep condition numbers as low as possible in numerical calculations to prevent the corresponding kernel matrix from being singular. Compared to such sensitive behaviors in K_{SE} and K_{NN} , the K_{DAS} kernel demonstrates consistently low condition numbers less than 10^5 over the wide range of grid resolutions up to 10^4 . The low condition number with K_{DAS} assures that its relevant calculations are highly reliable on a wide spectrum of discrete grid scales without experiencing numerical issues due to singularity.

It is evident in Figs. 4.2 and 4.3 that the discontinuous data-fitting with DAS is oscillation-free, tremendously outperforming the SE and NN kernels. There are two main reasons for the DAS kernel’s excellent performance on interpolation of discontinuous data. The first is the small region of dependence seen in Fig. 4.1. Regardless of r_{gp} , the DAS kernel maintains this narrow band, which leads to a lower order solution, in this case, second order. The second reason is because of the low condition number property. We can explain this by noting that the discontinuous data-fitting problem is a dot product operation of the two vectors $\mathbf{z}_*^T$ and $\mathbf{f}$ in Eq. (2.2) from the mathematical perspective. The higher the condition number of $\mathbf{K}$, the more the $\mathbf{z}_*^T$ becomes ill-conditioned, which negatively impacts the accuracy and stability of the data-fitting problem. Equivalently, $\mathbf{z}_*^T$ becomes well-conditioned for the kernel matrices with low condition numbers. This relationship between the condition number and the discontinuous data-fitting performance is observed in the results of Figs. 4.2 and 4.4 too. In

the right panel of Fig. 4.4, we see that, among the various kernels under comparison, the condition numbers of the NN kernel with $\sigma = 100$ and the DAS kernel are the closest to each other over a wide range of grid resolutions. Therefore, we can expect that the data-fitting performance of NN would be similar to that of DAS. Indeed, in the middle left panel in Fig. 4.2, the NN kernel with $\sigma = 100$ produces the least oscillatory interpolation across the discontinuity, demonstrating that its solution is qualitatively comparable to the DAS solution in the bottom left panel.

4.1.1 Conclusion

In this section, we have demonstrated several novel applications of Gaussian Processes (GPs) within the context of computational fluid dynamics (CFD) and numerical methods. By applying linear operators to the kernel matrices, we utilize a generalized prediction formula for GPs. This allows for the direct computation of various quantities, such as derivatives and volume averages, enhancing the versatility of GPs in numerical methods. We presented convergence tables for various test cases, demonstrating the high-order accuracy and robustness of our approach. We proposed a new kernel, the Discontinuous Arc Sin (DAS) kernel, designed to handle discontinuities and shocks more effectively than traditional kernels like the Squared Exponential (SE) and Neural Network (NN) kernels. The DAS kernel is also a stationary kernel, unlike the NN kernel, which makes it much more computationally efficient.

4.2 GP-MOOD Framework

4.2.1 Methodology

We now shift our focus to integrating kernel switching into the GP-MOOD framework. The Multidimensional Optimal Order Detection (MOOD) method, introduced in [37], is an a posteriori shock capturing algorithm. The MOOD method aims to use the highest order of reconstruction possible without admitting oscillations or non-physical states. It achieves this by employing a cascade of reconstruction orders, with the lowest order being a first-order Godunov method (FOG). The key idea of the MOOD method is to iteratively adjust the reconstruction order based on the acceptability of the resulting states. Initially, the highest possible order of reconstruction is attempted to maximize accuracy. If the resulting solution contains unacceptable states the reconstruction for that particular cell is redone with a lower-order scheme. This process continues until a suitable, stable solution is found. Thus, the MOOD method dynamically adapts the reconstruction order to maintain stability and physical accuracy. We'll use the MOOD method to simulate the compressible Euler equations defined by

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ E \end{pmatrix}, \quad \mathbf{F}_j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p \delta_{ij} \\ u_j(E + p) \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (4.3)$$

where ρ is the mass density, representing the mass of the fluid per unit volume; u_i is the fluid velocity in the i th direction; E is the total internal energy; p is the pressure; and δ_{ij} is the Kronecker delta, which is 1 when $i = j$ and 0 otherwise, used to handle the interaction of velocities in different directions.

We build upon the framework established in [21], which extends the MOOD method to incorporate Gaussian process (GP) reconstruction. This approach leverages the flexibility and accuracy of Gaussian processes to enhance the reconstruction step in the MOOD framework. Our contribution is the addition of kernel switching capability within the GP-MOOD framework. By allowing the use of multiple kernels, we can further optimize the reconstruction process. Different kernels can capture different characteristics of the data, and switching between them enables the method to adapt more effectively to varying flow features. This enhancement provides greater flexibility and potentially improves the accuracy and robustness of the shock capturing algorithm.

There are three acceptability criterion in the GP-MOOD algorithm, the Physical Admissibility Detection (PAD) where $\rho_{i,j} > 0$ and $p_{i,j} > 0$, Computer science Admissibility Detection (CAD) where any updated state $q_{i,j}$ must not be a NaN or Inf and the Discrete maximum principle (DMP). The DMP is essentially a numerical oscillation detector, where we compare the updated state of $q_{i,j}$ to the previous timestep's neighbors.

In this modified GP-MOOD framework we will use three cascade orders, a 5th order reconstruction using the SE kernel, an intermediate 3rd order reconstruction using the DAS kernel (3rd order refers to the stencil size, however we have seen in Table 4.2 that it is second order) and finally a first order method, FOG. We'll compare this to a 5/3/1 algorithm of only the SE kernel. We will go through two test problems.

4.2.2 Numerical Results

First the 2D Riemann problem configuration 3. The physical setup for the 2D Riemann problem involves a square computational domain divided into four quadrants, each with different initial conditions for pressure, density, and velocity. The initial discontinuities at the interfaces between these quadrants lead to the development of complex wave interactions. To validate the simulation, we look for the accurate capture and propagation of shock waves, rarefaction waves, and contact discontinuities. Key validation metrics include the correct prediction of wave speeds and positions and the preservation of symmetry. The solution and order of reconstruction is shown in Figure 4.5. In this figure, there is a significant increase in the middle order of reconstruction when using the DAS kernel. This is most likely because of its shock capturing abilities preventing the MOOD method from triggering to go down in order again, where the 3rd order SE method will still oscillate. This will ultimately lead to an increase in accuracy since we are using a higher order reconstruction on more cells which should give a more accurate solution. When comparing the top left and bottom left panes in Figure 4.5 the shear flow region created by the collision of the the fluid quadrants moving in different directions shows an increase in vorticity structure with the DAS kernel showing the product of the increased use in the higher order method compared to the traditional MOOD loop.

In the Mach 800 astrophysical jet setup, a high-velocity jet is injected into a stationary ambient medium within a rectangular computational domain. The jet has a Mach number of 800, indicating it travels at 800 times the speed of sound in the

ambient medium. The validation of this simulation involves observing the formation and evolution of shock structures, turbulence, and mixing layers. We look for the accurate representation of shock fronts and the preservation of symmetry. In Figure 4.6 the log of density is plotted. After some time, some asymmetries start to develop which were not present in the original MOOD method for the 5th order solution. The cause of this is unknown and will require some investigation into the multidimensional effects of DAS.

In table 4.3 we show the significant increase in the intermediate order cells when using the DAS kernel rather than the SE kernel. For the Riemann problem, there is a 552% increase in the intermediate order and for the mach 800 jet there is a 102% increase. Since the DAS kernel allows the MOOD method to significantly increase the intermediate order of reconstruction, the solution will be higher order and thus more accurate. It is also worth pointing out that the 2D Riemann problem also saw a slight increase in the lowest order method, but the Mach 800 saw a significant decrease in the lowest order method as well.

4.2.3 Conclusion

In this section we present the first CFD results using GP based reconstruction with kernel switching. By using the DAS kernel as the intermediate cascade order in the MOOD algorithm, we show higher order solutions at a cheaper cost compared to the pure SE counterpart in a 5/3/1 MOOD cascade. These benefits result from the MOOD method admitting more solutions on the intermediate order when using the DAS kernel compared to the SE kernel.

Table 4.3: Comparison of SE and DAS kernels for intermediate order in MOOD cascade.

Order of Stencil	Riemann Problem (C3)				Mach 800 Jet			
	SE Count	SE %	DAS Count	DAS %	SE Count	SE %	DAS Count	DAS %
5	65083	99.309	64683	98.698	358126	99.479	358236	99.51
3	70	0.106	392	0.598	400	0.111	810	0.225
1	383	0.584	461	0.703	1474	0.409	954	0.265

For the Riemann problem shown in Figure 4.5 and the Mach 800 Jet problem shown in Figure 4.7 the cell count of the final order of reconstruction determined by the MOOD method is shown as well as the percentages of each cell count for the final timestep. The values are compared between using the SE kernel for the intermediate order and the DAS kernel for the intermediate order.

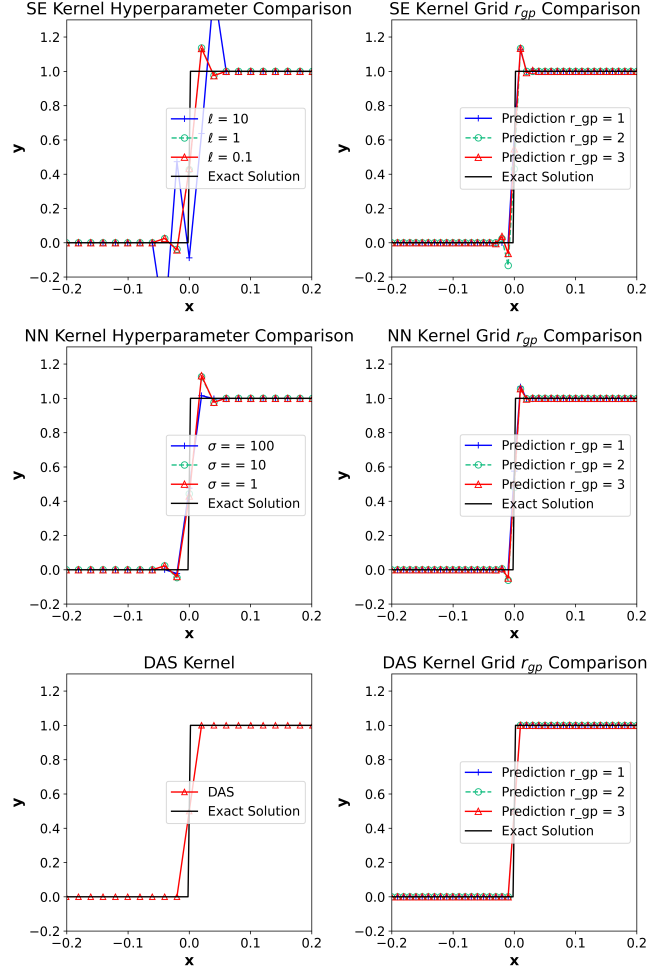


Figure 4.2: Interpolation across a single discontinuity using our GP point-to-point interpolation comparing different kernels. $N_x = 20$ for the left panel and $N_x = 40$ for the right. (Left) The hyperparameters of the kernels are varied on each kernel holding $r_{gp} = 2$. The SE kernel is varied over ℓ values of 0.1, 1.0, and 10. The NN kernel is varied over σ values of 1, 10, and 100. We keep $\sigma_0 = 1$ as it has little effect on the solution. DAS does not have any hyperparameters and is held at constant. (Right) The stencil order is varied from $r_{gp} = 1, 2, 3$ for each kernel choice. The SE kernel utilizes $\ell = 0.1$ and the NN kernel uses $\sigma = 100$ and $\sigma_0 = 1$.

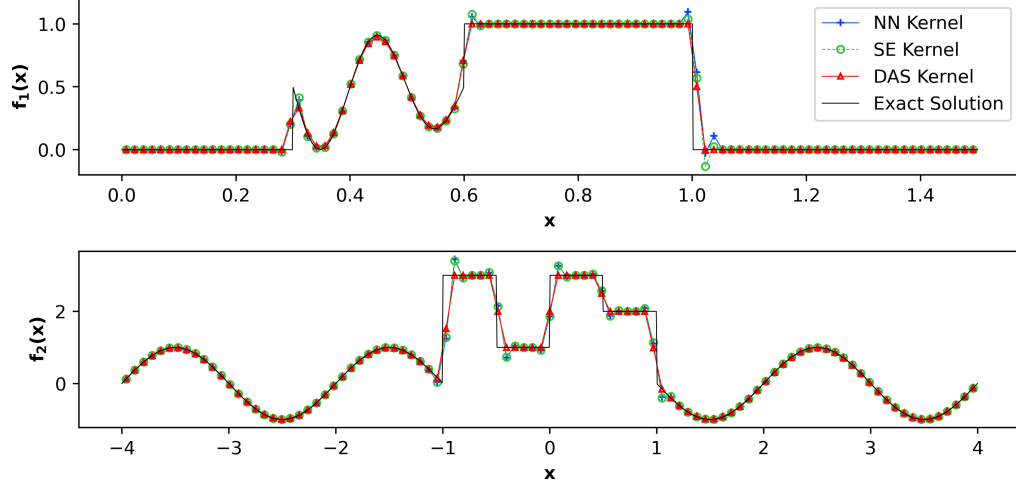


Figure 4.3: Comparison of point-to-point interpolation on three different profiles using the SE, NN, and DAS kernels. The NN kernel uses $\sigma = \sigma_0 = 1$, and the SE kernel uses $\ell = 12\Delta x$. **(Top)** The function, f_1 , is a piecewise function defined on $[0, 1.5]$ by $f_1(x) = 0$ for $0 \leq x < 0.3$ and $1.0 \leq x \leq 1.5$; $f_1(x) = e^{-2x} \sin(10\pi x) + 1/2$ for $0.3 \leq x < 0.6$; $f_1(x) = 1$ for $0.6 \leq x < 1.0$. **(Bottom)** The function, f_2 , is a piecewise function defined on $[-4, 4]$ by $f_2(x) = \sin(\pi x)$ for $|x| \geq 1.0$; $f_2(x) = 3$ for $-1.0 < x \leq -0.5$ and $0 < x \leq 0.5$; $f_2(x) = 1$ for $-0.5 < x \leq 0.0$; $f_2(x) = 2$ for $0.5 < x \leq 1.0$. All GP calculations were performed using $N_x = 100$ grid resolutions on each domain with $r_{gp} = 2$. The reference exact solution is shown using $N_x = 1000$.

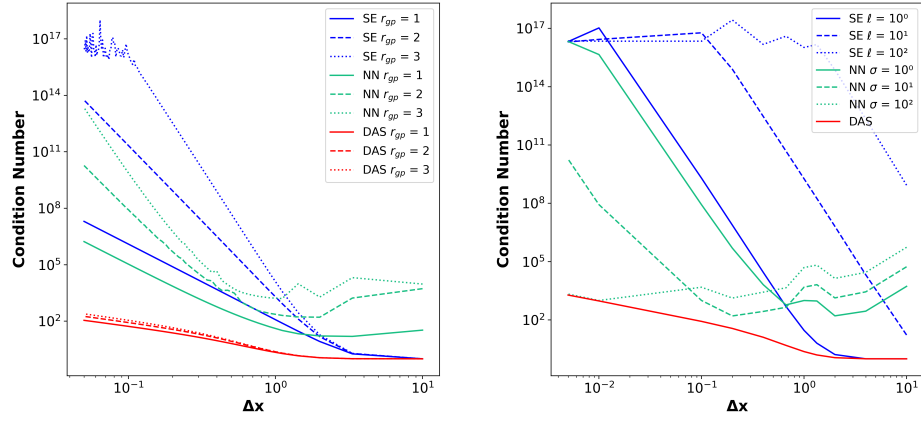


Figure 4.4: Condition number of $\mathbf{K}$ plotted as a function of grid spacing Δx . (Left) Comparison of the three kernels for various stencil sizes. The hyperparameters are held at $\ell = 1.8$ for SE and $\sigma = \sigma_0 = 1$ for NN, while r_{gp} is varied for all cases. (Right) Comparison of the three kernels for hyperparameter values. We set $r_{gp} = 2$ for all kernels and hyperparameters are varied according to the legend.

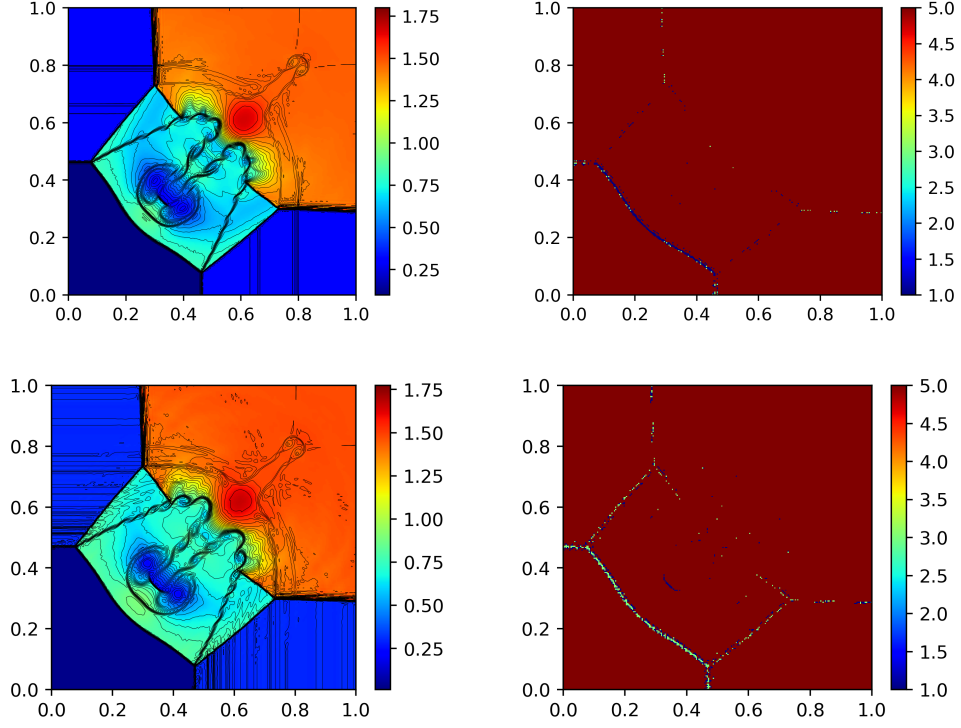


Figure 4.5: Plot of pressure with contours on the left and the order of the MOOD reconstruction on the right. The top figures are for original 5th order MOOD method, with SE kernel everywhere, where the bottom figures use the DAS kernel for the 2nd order method. Both methods present great symmetry and we see an increase in the higher order method when using the DAS kernel which will lead to greater accuracy overall.

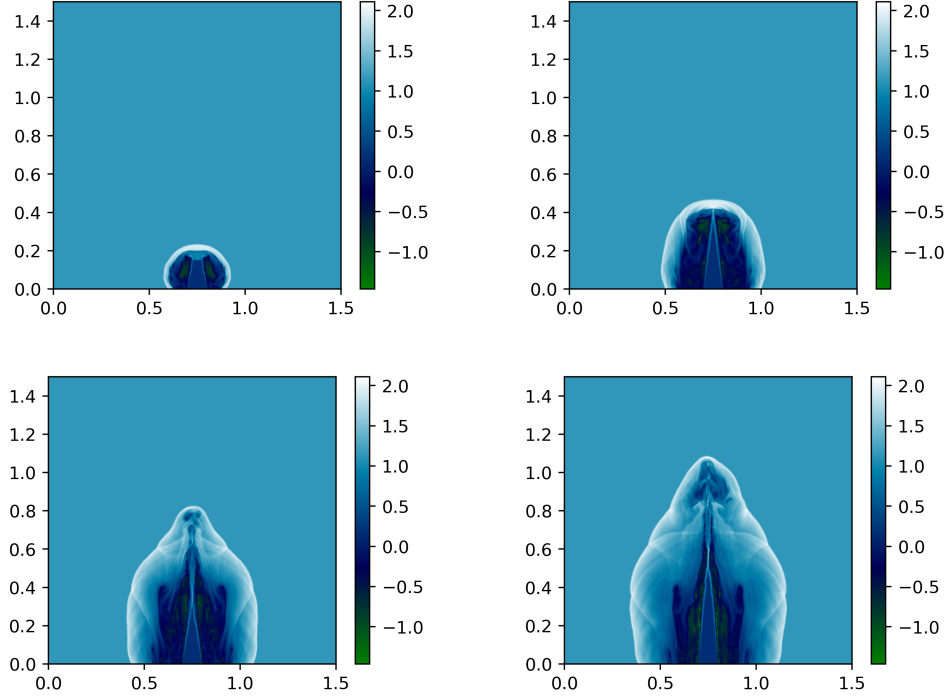


Figure 4.6: Plot of $\log_{10}(\rho)$ of the Mach 800 Astrophysical Jet problem varying in time.

In the bottom two panes we can see some asymmetries start to develop that did not occur in the original GP-MOOD 5th order simulation. Here we use the SE kernel for the 5th order method, DAS kernel for the 3rd (2nd) order method and FOG as a last resort. Top Left: $t = 0.00875$ sec, Top Right: $t = 0.016875$ sec, Bottom Left: $t = 0.0253125$ sec, Bottom Right: $t = 0.0340625$ sec.

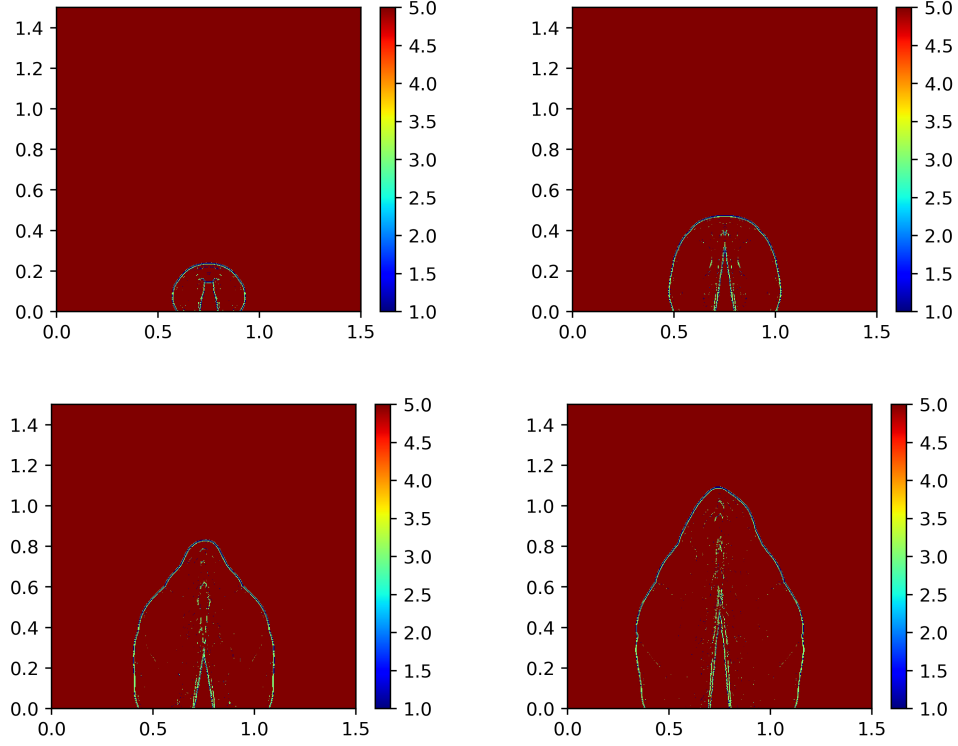


Figure 4.7: Plot of order of reconstruction of the Mach 800 Astrophysical Jet problem varying in time. In the bottom two panes we can see some asymmetries start to develop that did not occur in the original GP-MOOD 5th order simulation. Here we use the SE kernel for the 5th order method, DAS kernel for the 3rd (2nd) order method and FOG as a last resort. Top Left: $t = 0.00875$ sec, Top Right: $t = 0.016875$ sec, Bottom Left: $t = 0.0253125$ sec, Bottom Right: $t = 0.0340625$ sec

Chapter 5

Fully Conservative Pressure Equilibrium Preserving Methods

5.1 Introduction

Spurious pressure oscillations have been observed and studied in multi-component and real-fluid simulations for decades [3]. The root cause of this error is well known: when the ratio of specific heats is not constant across species, or when, in real fluids, the effective γ varies with local thermodynamic conditions, small inconsistencies enter the discrete thermodynamic state at the cell interface. In practice the resulting pressure error is of the order of the discretization error per step, but it can and will grow indefinitely [55]. For simulations with long time horizons or with uniform-pressure initial data, this growth will eventually dominate the solution, making it a necessary consideration to address.

A number of approaches have been developed to preserve the pressure-equilibrium

property (PEP). A simple option is to adopt a non-conservative update in which the numerical flux update is applied directly to the pressure field [71]. This enforces pressure equilibrium exactly at the discrete level, but it does so at the cost of strict conservation of total energy. More recently, [35] proposed a discontinuous Galerkin scheme that likewise evolves a pressure-evolution equation in place of the total-energy equation, and then adds Abgrall-type correction terms so that the final method is velocity- and pressure-equilibrium-preserving and (semi-discretely) energy-conservative for mixtures of thermally perfect gases. A related and widely used idea, often referred to as the double-flux method [99], updates the conservative variables and, in addition, evolves a pressure flux that is used as a correction to the total energy. In practice this aligns the energy update more closely with the pressure evolution and generally reduces energy drift, though the scheme remains formally non-conservative. There are also quasi-conservative formulations [2, 94] in which the governing equations are modified to evolve an auxiliary term related to the effective ratio of specific heats; for ideal gases this can restore PEP under the altered dynamics. Some systems, such as the five-equation model for multiphase flows [5], satisfy PEP by construction and require no additional treatment. Complementary mechanisms include consistent artificial dissipation designed to damp spurious pressure modes without corrupting equilibrium states [143], and hybrid schemes that switch between quasi-conservative and double-flux updates to balance robustness and conservation locally [24].

More recently, [55] presented the first fully conservative scheme that numerically satisfies PEP for the ideal-gas and stiffened-gas equations of state (with zero reference energy q). Their analysis identifies an exact algebraic relation coupling the

discrete species and energy updates so that a uniform-pressure state is an invariant of the semi-discrete system during contact advection. Building on the same principle, [142] derived an approximate pressure-equilibrium conservative (APEC) scheme for general equations of state by enforcing an approximate form of that algebraic condition. As formulated, APEC substantially reduces pressure oscillations for complex EOSs, but the original presentation is limited to a second-order scheme in space.

In this work we take two steps. First, we extend APEC to high-order by formulating interface-consistent energy corrections that are compatible with high-order reconstruction and flux-differencing, removing the second-order limitation while retaining the intended pressure stabilization. Second, we derive, for an arbitrary EOS, the first fully conservative scheme that exactly spatially satisfies the PEP condition at the semi-discrete level. The resulting discretization preserves conservation of mass, momentum, and energy while significantly minimizing accrued pressure error over time.

This paper is organized as follows. Section 5.2 derives the discrete pressure-equilibrium condition, Sections 5.3 and 5.4 develop high-order APEC and a fully conservative PEP scheme for general EOSs, and Section 5.6 presents numerical results for ideal-gas, stiffened-gas, and van der Waals mixtures.

5.2 Derivation of the Pressure Equilibrium Preserving Condition

For the purpose of the derivation, we consider the 1D Euler equation for two species written as

$$\begin{aligned}
\frac{\partial(\rho Y_1)}{\partial t} + \frac{\partial(\rho Y_1 u)}{\partial x} &= 0 \\
\frac{\partial(\rho Y_2)}{\partial t} + \frac{\partial(\rho Y_2 u)}{\partial x} &= 0 \\
\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u u + p)}{\partial x} &= 0 \\
\frac{\partial(\rho E)}{\partial t} + \frac{\partial[(\rho E + p)u]}{\partial x} &= 0,
\end{aligned} \tag{5.1}$$

where $E = e + u^2/2$

We will base our derivation of the discrete spatial pressure-equilibrium criterion on the approach of Fujiwara [55], with some alterations to explicitly demonstrate the temporal convergence aspect of the pressure-equilibrium scheme. Consider a solution of the multispecies 1D Euler equation using the method of lines, where a spatial discretization is employed to convert the PDE system into an ODE system of the discretized flow variables in time

$$\left. \frac{d(\rho Y_1)}{dt} \right|_m + F_{\rho Y_1}|_m = 0 \tag{5.2}$$

$$\left. \frac{d(\rho Y_2)}{dt} \right|_m + F_{\rho Y_2}|_m = 0 \tag{5.3}$$

$$\left. \frac{d(\rho u)}{dt} \right|_m + F_{\rho u}|_m = 0 \tag{5.4}$$

$$\left. \frac{\partial(\rho E)}{\partial t} \right|_m + F_{\rho E}|_m = 0 \tag{5.5}$$

where $F_{\rho Y_1}|_m, F_{\rho Y_2}|_m, F_{\rho u}|_m, F_{\rho E}|_m$ are the discretized convective flux operators and m is the spatial grid point index. In this study, we consider the split form of the Kinetic

Energy Entropy Preserving (KEEP) scheme [81] for these convective flux operators

$$F_{\rho Y_1}|_m = \frac{C_1|_{m+\frac{1}{2}} - C_1|_{m-\frac{1}{2}}}{\Delta x} \quad (5.6)$$

$$F_{\rho Y_2}|_m = \frac{C_2|_{m+\frac{1}{2}} - C_2|_{m-\frac{1}{2}}}{\Delta x} \quad (5.7)$$

$$F_{\rho u}|_m = \frac{M|_{m+\frac{1}{2}} - M|_{m-\frac{1}{2}}}{\Delta x} + \frac{\Pi|_{m+\frac{1}{2}} - \Pi|_{m-\frac{1}{2}}}{\Delta x} \quad (5.8)$$

$$F_{\rho E}|_m = \frac{K|_{m+\frac{1}{2}} - K|_{m-\frac{1}{2}}}{\Delta x} + \frac{I|_{m+\frac{1}{2}} - I|_{m-\frac{1}{2}}}{\Delta x} + \frac{S|_{m+\frac{1}{2}} - S|_{m-\frac{1}{2}}}{\Delta x}, \quad (5.9)$$

where

$$C_1|_{m+\frac{1}{2}} = \rho Y_1|_{m+\frac{1}{2}} \frac{u|_m + u|_{m+1}}{2} \quad (5.10)$$

$$C_2|_{m+\frac{1}{2}} = \rho Y_2|_{m+\frac{1}{2}} \frac{u|_m + u|_{m+1}}{2} \quad (5.11)$$

$$M|_{m+\frac{1}{2}} = \rho|_{m+\frac{1}{2}} \frac{u|_m + u|_{m+1}}{2} \frac{u|_m + u|_{m+1}}{2} \quad (5.12)$$

$$\Pi|_{m+\frac{1}{2}} = \frac{p|_m + p|_{m+1}}{2} \quad (5.13)$$

$$K|_{m+\frac{1}{2}} = \rho|_{m+\frac{1}{2}} \frac{u|_m + u|_{m+1}}{2} \frac{u|_m u|_{m+1}}{2} \quad (5.14)$$

$$I|_{m+\frac{1}{2}} = \rho e|_{m+\frac{1}{2}} \frac{u|_m + u|_{m+1}}{2} \quad (5.15)$$

$$S|_{m+\frac{1}{2}} = \frac{u|_m p|_{m+1} + u|_{m+1} p|_m}{2} \quad (5.16)$$

We refer to this baseline numerical method as our Non-Pressure-Equilibrium method “NPE”. In the method of line, the conserved variables $\rho Y_1, \rho Y_2, \rho u, \rho E$ are directly updated using time-stepping of the semi-discrete ODEs. On the other hand, the derived variable p is indirectly updated from the conserved variables via

$$p = \text{EOS}(\rho e, \rho Y_1, \rho Y_2) = \text{EOS}\left(\rho E - \frac{1}{2} \frac{(\rho u)^2}{\rho}, \rho Y_1, \rho Y_2\right) \quad (5.17)$$

The pressure-equilibrium preserving property is analyzed for an initial flow state with constant velocity and pressure: $u(x, 0) = U_0$ and $p(x, 0) = p_0$. First, we

analyze the velocity-equilibrium aspect of the scheme. Analyzing velocity equilibrium first ensures that no artificial momentum errors propagate into the pressure field, allowing a clean assessment of the pressure-equilibrium condition. Assuming a single-step temporal scheme for the update of flow variables from time t_n to time t_{n+1} , then

$$u_m^{n+1} - u_m^n = \left(\frac{\partial u}{\partial \rho u} \right)_{\rho|_m} \Big|_m^n (\rho u_m^{n+1} - \rho u_m^n) + \left(\frac{\partial u}{\partial \rho} \right)_{\rho u|_m} \Big|_m^n (\rho_m^{n+1} - \rho_m^n) + \text{H.O.} \quad (5.18)$$

$$= -\frac{1}{\rho|_m^n} F_{\rho u}|_m^n \Delta t + \frac{U_0}{\rho|_m^n} (F_{\rho Y_1}|_m^n + F_{\rho Y_2}|_m^n) \Delta t + O(\Delta t^2). \quad (5.19)$$

Here, we only collect the leading order terms in the time stepping, leading to a first-order global convergence of the velocity equilibrium property. The same treatment will be made for the pressure equilibrium property later, with further remarks. Substituting the KEEP scheme into the (implicit) velocity update rule, under the assumption of constant velocity and pressure at time step t_n , gives

$$u_m^{n+1} - u_m^n = \frac{U_0^2}{\rho|_m^n} \left[-\left(\rho|_{m+\frac{1}{2}}^n - \rho|_{m-\frac{1}{2}}^n \right) + \left(\rho Y_1|_{m+\frac{1}{2}}^n - \rho Y_1|_{m-\frac{1}{2}}^n \right) + \left(\rho Y_2|_{m+\frac{1}{2}}^n - \rho Y_2|_{m-\frac{1}{2}}^n \right) \right] \frac{\Delta t}{\Delta x} + O(\Delta t^2). \quad (5.20)$$

From here, we see that with the choice of mid-point density

$$\rho|_{m+\frac{1}{2}} \equiv \rho Y_1|_{m+\frac{1}{2}} + \rho Y_2|_{m+\frac{1}{2}}, \quad (5.21)$$

the velocity equilibrium is preserved with first order global convergence in time for all grid spacing.

In this manner, we refer to the present scheme as being spatially exact in preserving velocity equilibrium. Further remarks on this point will be made shortly when the same treatment is considered for the pressure.

For pressure, since the update will consider the internal energy through the EOS, we first look at the implicit update rule for $\rho e = \rho E - \frac{1}{2}(\rho u)^2/\rho$, once again assuming a single-step temporal scheme,

$$\begin{aligned} \rho e|_m^{n+1} - \rho e|_m^n &= \left(\frac{\partial \rho e}{\partial \rho E} \right)_{\rho u, \rho} \bigg|_m^n (\rho E|_m^{n+1} - \rho E|_m^n) + \left(\frac{\partial \rho e}{\partial (\rho u)} \right)_{\rho E, \rho} \bigg|_m^n (\rho u|_m^{n+1} - \rho u|_m^n) \\ &\quad + \left(\frac{\partial \rho e}{\partial \rho} \right)_{\rho E, \rho u} \bigg|_m^n (\rho|_m^{n+1} - \rho|_m^n) + \text{H.O.} \end{aligned} \quad (5.22)$$

$$= - F_{\rho E}|_m^n \Delta t + u|_m^n F_{\rho u}|_m^n \Delta t - \frac{1}{2} (u|_m^n)^2 F_{\rho}|_m^n \Delta t + O(\Delta t^2) \quad (5.23)$$

From Equation 5.23, we now substitute the split-form KEEP fluxes (5.10)–(5.16). We then regroup the resulting face contributions into an internal-energy advection part and two “cancellation” parts, which vanish identically under a uniform-velocity and uniform-pressure state at t_n .

$$\begin{aligned} \rho e|_m^{n+1} - \rho e|_m^n &= - \left[F_{\rho E}|_m^n - u|_m^n F_{\rho u}|_m^n + \frac{1}{2} (u|_m^n)^2 F_{\rho}|_m^n \right] \Delta t + O(\Delta t^2) \\ &= - \frac{\Delta t}{\Delta x} \left[\left(I|_{m+\frac{1}{2}}^n - I|_{m-\frac{1}{2}}^n \right) \right. \\ &\quad + \left(S|_{m+\frac{1}{2}}^n - S|_{m-\frac{1}{2}}^n \right) - u|_m^n \left(\Pi|_{m+\frac{1}{2}}^n - \Pi|_{m-\frac{1}{2}}^n \right) \\ &\quad + \left(K|_{m+\frac{1}{2}}^n - K|_{m-\frac{1}{2}}^n \right) - u|_m^n \left(M|_{m+\frac{1}{2}}^n - M|_{m-\frac{1}{2}}^n \right) \\ &\quad \left. + \frac{1}{2} (u|_m^n)^2 \left((C_1 + C_2)|_{m+\frac{1}{2}}^n - (C_1 + C_2)|_{m-\frac{1}{2}}^n \right) \right] + O(\Delta t^2). \end{aligned} \quad (5.24)$$

Assuming a constant velocity, U_0 and pressure p_0 at t_n in Equation 5.24, we

get

$$\rho e|_m^{n+1} - \rho e|_m^n = -U_0 \left(\rho e|_{m+\frac{1}{2}}^n - \rho e|_{m-\frac{1}{2}}^n \right) \frac{\Delta t}{\Delta x} + O(\Delta t^2). \quad (5.25)$$

Now, the implicit update rule for pressure can be derived for a single-step temporal scheme,

$$\begin{aligned} p|_m^{n+1} - p|_m^n &= \left(\frac{\partial p}{\partial \rho e} \right)_{\rho Y_1, \rho Y_2} \Big|_m^n \left(\rho e|_m^{n+1} - \rho e|_m^n \right) \\ &\quad + \left(\frac{\partial p}{\partial \rho Y_1} \right)_{\rho e, \rho Y_2} \Big|_m^n \left(\rho Y_1|_m^{n+1} - \rho Y_1|_m^n \right) \\ &\quad + \left(\frac{\partial p}{\partial \rho Y_2} \right)_{\rho e, \rho Y_1} \Big|_m^n \left(\rho Y_2|_m^{n+1} - \rho Y_2|_m^n \right) + \text{H.O.} \end{aligned} \quad (5.26)$$

$$\begin{aligned} &= -U_0 \left[\left(\frac{\partial p}{\partial \rho e} \right)_{\rho Y_1, \rho Y_2} \Big|_m^n \left(\rho e|_{m+\frac{1}{2}}^n - \rho e|_{m-\frac{1}{2}}^n \right) \right. \\ &\quad + \left(\frac{\partial p}{\partial \rho Y_1} \right)_{\rho e, \rho Y_2} \Big|_m^n \left(\rho Y_1|_{m+\frac{1}{2}}^n - \rho Y_1|_{m-\frac{1}{2}}^n \right) \\ &\quad \left. + \left(\frac{\partial p}{\partial \rho Y_2} \right)_{\rho e, \rho Y_1} \Big|_m^n \left(\rho Y_2|_{m+\frac{1}{2}}^n - \rho Y_2|_{m-\frac{1}{2}}^n \right) \right] \frac{\Delta t}{\Delta x} + O(\Delta t^2). \end{aligned} \quad (5.27)$$

Thus, we have derived the condition for spatially exact pressure equilibrium preserving. This criterion is expressed in terms of the mid-point values for internal energy and the partial densities and is identical in form to the criterion derived in [55], with the added context of temporal convergence,

$$\begin{aligned} &\left(\frac{\partial p}{\partial \rho e} \right)_{\rho Y_1, \rho Y_2} \Big|_m^n \left(\rho e|_{m+\frac{1}{2}}^n - \rho e|_{m-\frac{1}{2}}^n \right) + \left(\frac{\partial p}{\partial \rho Y_1} \right)_{\rho e, \rho Y_2} \Big|_m^n \left(\rho Y_1|_{m+\frac{1}{2}}^n - \rho Y_1|_{m-\frac{1}{2}}^n \right) \\ &+ \left(\frac{\partial p}{\partial \rho Y_2} \right)_{\rho e, \rho Y_1} \Big|_m^n \left(\rho Y_2|_{m+\frac{1}{2}}^n - \rho Y_2|_{m-\frac{1}{2}}^n \right) = 0 \end{aligned} \quad (5.28)$$

To rewrite (5.28) as a condition on the mid-point jump in ρe , we solve for $\left(\rho e|_{m+\frac{1}{2}}^n - \rho e|_{m-\frac{1}{2}}^n \right)$ and express the resulting coefficients using the EOS along with a

constant pressure. This yields

$$\begin{aligned}
\left(\rho e|_{m+\frac{1}{2}}^n - \rho e|_{m-\frac{1}{2}}^n\right) &= - \sum_{i=1}^N \left. \frac{\left(\frac{\partial p}{\partial \rho Y_i}\right)_{\rho e, \rho Y_{j \neq i}}}{\left(\frac{\partial p}{\partial \rho e}\right)_{\rho Y_1, \dots, \rho Y_N}} \right|_m^n \left(\rho Y_i|_{m+\frac{1}{2}}^n - \rho Y_i|_{m-\frac{1}{2}}^n\right) \\
&= \sum_{i=1}^N \left. \left(\frac{\partial \rho e}{\partial \rho Y_i}\right)_{\rho Y_{j \neq i}, p} \right|_m^n \left(\rho Y_i|_{m+\frac{1}{2}}^n - \rho Y_i|_{m-\frac{1}{2}}^n\right).
\end{aligned} \tag{5.29}$$

Finally, we write the PEP criterion as

$$\Rightarrow \rho e|_{m+\frac{1}{2}} - \rho e|_{m-\frac{1}{2}} = \sum_{i=1}^N \left. \left(\frac{\partial \rho e}{\partial \rho Y_i}\right)_{\rho Y_{j \neq i}, p} \right|_m \left(\rho Y_i|_{m+\frac{1}{2}} - \rho Y_i|_{m-\frac{1}{2}}\right). \tag{5.30}$$

We emphasize that the pressure-equilibrium criterion derived above is only spatially exact, in the sense that pressure error converges with the same (linear) rate with time step size for all mesh spacings. This is a common property of pressure-equilibrium preserving conservative methods [55] since we only explicitly update the conservative variables $(\rho Y_i, \rho u, \rho E)$. Nonetheless, relative to the NPE discretization, the present approach removes the need to rely on mesh refinement to recover pressure equilibrium, since the spatial PEP condition is enforced independently of Δx . Any remaining pressure error in the fully discrete scheme is then attributable to the time discretization and is proportional to the time-step size, i.e., $O(\Delta t)$ for the single-step update used in this derivation. Higher-order temporal variants can be constructed by applying standard multi-stage or multi-step integrators to the same Euler-type updates presented here, which would yield $O(\Delta t^q)$ temporal error for a q th-order method. Extensions of the above analysis to higher-order truncations in Δt is the subject of future works.

To apply Equation 5.30 we first split into a left and right condition shown as

$$\rho e|_m - \rho e|_{m-\frac{1}{2}} = \sum_{i=1}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)_{\rho Y_{j \neq i, p}} \Big|_m \left(\rho Y_i|_m - \rho Y_i|_{m-\frac{1}{2}} \right) \quad (5.31)$$

$$\rho e|_{m+\frac{1}{2}} - \rho e|_m = \sum_{i=1}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)_{\rho Y_{j \neq i, p}} \Big|_m \left(\rho Y_i|_{m+\frac{1}{2}} - \rho Y_i|_m \right) \quad (5.32)$$

We then shift Equation 5.31 by one cell to the right to get

$$\rho e|_{m+1} - \rho e|_{m+\frac{1}{2}} = \sum_{i=1}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)_{\rho Y_{j \neq i, p}} \Big|_{m+1} \left(\rho Y_i|_{m+1} - \rho Y_i|_{m+\frac{1}{2}} \right). \quad (5.33)$$

where in practice for the development of our scheme, Equations 5.32 and 5.33 represent the left and right interface conditions one must satisfy to preserve the pressure-equilibrium property.

5.3 Derivation of High-Order APEC

The Approximately Pressure-Equilibrium-preserving Conservative scheme (APEC) [142] is derived by considering an approximation to the PEP criterion (Equation 5.30) rather than satisfying Equations 5.32 and 5.33 directly. This is done by subtracting Equation 5.32 from Equation 5.33 and solving for the half point value $\rho e|_{m+\frac{1}{2}}$ which gives us:

$$\begin{aligned} \rho e|_{m+\frac{1}{2}}^{\text{APEC}} = & \frac{\rho e|_m + \rho e|_{m+1}}{2} + \frac{1}{2} \sum_{i=1}^N \left[\left(\frac{\partial \rho e}{\partial \rho Y_i} \right)_{\rho Y_{j \neq i, p}} \Big|_m \left(\rho Y_i|_{m+\frac{1}{2}} - \rho Y_i|_m \right) \right. \\ & \left. - \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)_{\rho Y_{j \neq i, p}} \Big|_{m+1} \left(\rho Y_i|_{m+1} - \rho Y_i|_{m+\frac{1}{2}} \right) \right], \end{aligned} \quad (5.34)$$

where N is the number of species. This condition is a necessary but not sufficient condition to satisfy the PEP property and thus it is an approximate method. Terashima [142] couples this newly derived halfpoint value with the KEEP scheme and shows a considerable decrease in pressure error and an increase in stability for an arbitrary EOS. Importantly, this correction term can also be applied to any flux, but the scheme is left at second order. We will now present the derivation of a high-order variant of this scheme.

We begin by presenting a high-order extension of the KEEP fluxes from [80], expressed in split two-point flux form. This approach follows the general methodology first introduced in [114] for extending any split convective operator to higher orders. For a general scalar variable ϕ in a convective term $\rho\phi u$ the numerical flux to an arbitrary order of accuracy is defined as

$$\begin{aligned} \tilde{\mathcal{F}}_\phi \Big|_{(m \pm \frac{1}{2})} \equiv \\ 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{\rho|_{(m \mp \ell)} + \rho|_{(m \mp \ell \pm i_s)}}{2} \frac{\phi|_{(m \mp \ell)} + \phi|_{(m \mp \ell \pm i_s)}}{2} \frac{u|_{(m \mp \ell)} + u|_{(m \mp \ell \pm i_s)}}{2}, \end{aligned} \quad (5.35)$$

where q is the order of the scheme, i_s is the size of the sub-stencil, and a_{q,i_s} are the central difference approximation coefficients as shown in Table 5.1,

We make the observation that the high-order flux in Equation 5.35 constructs the interface state as a weighted sum of multiple sub-interface contributions, where each sub-interface is formed from cell pairs $(m \mp \ell, m \mp \ell \pm i_s)$. This is shown clearly for orders 2, 4, and 6 in Figure 5.1. For the internal energy flux, each sub-interface state can be

Order	Coefficients			
q	$a_{q,1}$	$a_{q,2}$	$a_{q,3}$	$a_{q,4}$
2	$\frac{1}{2}$			
4	$\frac{2}{3}$	$-\frac{1}{12}$		
6	$\frac{3}{4}$	$-\frac{3}{20}$	$\frac{1}{60}$	
8	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$

Table 5.1: Coefficients (a_{q,i_s}) for central difference approximation of first derivative as shown in [80] and derived in [53]. q is the order of accuracy of the method and i_s is the stencil size.

expressed as:

$$\rho e|_{(m \mp \ell, m \mp \ell \pm i_s)}^{\text{sub-interface}} = \frac{\rho e|_{(m \mp \ell)} + \rho e|_{(m \mp \ell \pm i_s)}}{2} \quad (5.36)$$

which, when coupled with the corresponding velocity and density produces a pressure error analogous to the second-order case.

We therefore propose to apply an APEC correction to each sub-interface. By applying the correction derived in Equation 5.34 to each cell pair in the stencil, we ensure that every sub-interface contribution maintains approximate thermodynamic con-

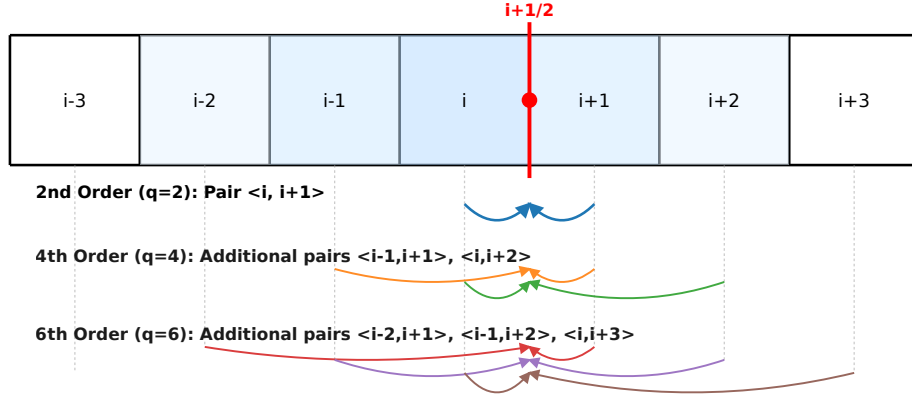


Figure 5.1: Construction of the high-order interface flux at $i + 1/2$ as a weighted sum of multiple cell-pair contributions. Dashed gray lines connect cell centers to their arrow origins. For second-order accuracy (blue), a single pair $\langle i, i + 1 \rangle$ forms the sub-interface state. Fourth-order accuracy adds two additional pairs: $\langle i - 1, i + 1 \rangle$ (orange) and $\langle i, i + 2 \rangle$ (green). Sixth-order accuracy further extends the stencil with three more pairs: $\langle i - 2, i + 1 \rangle$ (red), $\langle i - 1, i + 2 \rangle$ (purple), and $\langle i, i + 3 \rangle$ (brown). Each arrow pair represents a sub-interface contribution formed by averaging the pointwise values at two cell centers, and the final interface flux is the weighted sum of all these sub-interface contributions according to Equation 5.35.

sistency between ρe and ρY_i . This leads to a high-order APEC scheme where the correction is distributed across all sub-interfaces weighted by their respective central difference coefficients a_{q,i_s} .

The high-order APEC corrected interface value for internal energy, $\rho e|_{m+\frac{1}{2}}$ is defined as

$$\rho e|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} \equiv 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{\rho e|_{(m\mp\ell)} + \rho e|_{(m\mp\ell\pm i_s)}}{2} - \sum_{i=1}^N \left[\frac{\left(\frac{\partial \rho e}{\partial \rho Y_i} \right) \rho Y_{j \neq i, p} \Big|_{(m\mp\ell)}}{2} - \left(\frac{\partial \rho e}{\partial \rho Y_i} \right) \rho Y_{j \neq i, p} \Big|_{(m\mp\ell\pm i_s)} \frac{\rho Y_i|_{(m\mp\ell)} - \rho Y_i|_{(m\mp\ell\pm i_s)}}{2} \right], \quad (5.37)$$

where N is the number of species, ρY_i is the mass density of species k . When defining the scheme, we also have the freedom to pick a flux or half-point values for ρY_1 and ρY_2 because both constraints were used in the derivation of the approximate condition (Equation 5.34). APEC chooses to make these the average of the neighboring points

$$\rho Y_i|_{m+\frac{1}{2}}^{\text{APEC}} = \frac{\rho Y_i|_m + \rho Y_i|_{m+1}}{2} \quad (5.38)$$

the equivalent in the high-order extension of APEC is to follow Equation 5.35 which we will write explicitly for clarity

$$\rho Y_i|_{m+\frac{1}{2}} \equiv 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{\rho Y_i|_{(m\mp\ell)} + \rho Y_i|_{(m\mp\ell\pm i_s)}}{2}, \quad (5.39)$$

where i is the mass fraction number.

5.3.1 Summary of Proposed Scheme - High-Order APEC

Utilizing $\rho e|_{m+\frac{1}{2}}$ as written in Equation 5.37 we also define the interface of density as

$$\rho|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \left[\frac{\rho Y_1|_{(m\mp\ell)} + \rho Y_1|_{(m\mp\ell\pm i_s)}}{2} + \frac{\rho Y_2|_{(m\mp\ell)} + \rho Y_2|_{(m\mp\ell\pm i_s)}}{2} \right]. \quad (5.40)$$

We can now write the complete high-order APEC scheme as

$$C_1|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{\rho Y_1|_{(m\mp\ell)} + \rho Y_1|_{(m\mp\ell\pm i_s)}}{2} \frac{u|_{(m\mp\ell)} + u|_{(m\mp\ell\pm i_s)}}{2} \quad (5.41)$$

$$C_2|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{\rho Y_2|_{(m\mp\ell)} + \rho Y_2|_{(m\mp\ell\pm i_s)}}{2} \frac{u|_{(m\mp\ell)} + u|_{(m\mp\ell\pm i_s)}}{2} \quad (5.42)$$

$$M|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \left(\rho|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} \right) \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \left[\frac{u|_{(m\mp\ell)} + u|_{(m\mp\ell\pm i_s)}}{2} \right]^2 \quad (5.43)$$

$$\Pi|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{p|_{(m\mp\ell)} + p|_{(m\mp\ell\pm i_s)}}{2} \quad (5.44)$$

$$K|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \left(\rho|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} \right) \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{u|_{(m\mp\ell)} + u|_{(m\mp\ell\pm i_s)}}{2} \frac{u|_{(m\mp\ell)} + u|_{(m\mp\ell\pm i_s)}}{2} \quad (5.45)$$

$$I|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \left(\rho e|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} \right) \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{u|_{(m\mp\ell)} + u|_{(m\mp\ell\pm i_s)}}{2} \quad (5.46)$$

$$S|_{m+\frac{1}{2}}^{\text{APEC } \mathcal{O}(q)} = 2 \sum_{i_s=1}^{q/2} a_{q,i_s} \sum_{\ell=0}^{i_s-1} \frac{u|_{(m\mp\ell)} p|_{(m\mp\ell\pm i_s)} + u|_{(m\mp\ell\pm i_s)} p|_{(m\mp\ell)}}{2} \quad (5.47)$$

5.3.2 Error Analysis of the High-Order APEC Scheme

Since APEC is an approximate PEP scheme, it is worthwhile to analyze the error of pressure acquired by utilizing this scheme. We define a pressure error “flux” as

$$f_{PE}|_m = \frac{\rho e|_{m+\frac{1}{2}} - \rho e|_{m-\frac{1}{2}}}{\Delta x} - \sum_{i=1}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)_{\rho Y_j \neq i, p} \bigg|_m \left(\frac{\rho Y_i|_{m+\frac{1}{2}} - \rho Y_i|_{m-\frac{1}{2}}}{\Delta x} \right), \quad (5.48)$$

which is Equation 5.30 rewritten to account for the fact that the scheme will not exactly satisfy this relationship and some pressure error will be present. For a PEP scheme $f_{PE}|_m$ should be exactly 0. We can then choose the order of our method, in this work we'll focus on orders 2, 4, 6, and 8, and get the interface values needed to define the high-order APEC scheme, $\rho e|_{m+\frac{1}{2}}$, $\rho Y_1|_{m+\frac{1}{2}}$, and $\rho Y_2|_{m+\frac{1}{2}}$ defined by Equations 5.37 and 5.39 respectively. We perform a Taylor series analysis by expanding the variables ρY_i , ρe , and the partial derivatives $\left(\frac{\partial \rho e}{\partial \rho Y_i} \right)_{\rho Y_j \neq i, p}$ about the cell center m in terms of the grid spacing Δx .

Each quantity q at stencil point $m + \ell$ is represented as

$$q_{m+\ell} = \sum_{n=0}^{N_T} q_m^{(n)} (\ell \Delta x)^n / n!, \quad (5.49)$$

where $q_m^{(n)}$ denotes the n^{th} derivative of q evaluated at cell center m and N_T is chosen sufficiently large to capture the leading order terms. These Taylor expansions are then substituted into Equations 5.37 and 5.39 to obtain symbolic expressions for the interface values $\rho e|_{m+\frac{1}{2}}$, $\rho Y_i|_{m+\frac{1}{2}}$ (and their corresponding minus-side values). Finally, these interface values are used to evaluate $f_{PE}|_m$ from Equation 5.48, which is then expanded and we analyze the leading error term.

We note that the 0^{th} order term in all of the following Taylor expansions is

$$R_{PE}|_m \equiv \frac{\partial \rho e}{\partial x} \bigg|_m - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \bigg|_m \frac{\partial \rho Y_i}{\partial x} \bigg|_m \right) \quad (5.50)$$

which we define as $R_{PE}|_m$ for the residual of pressure error at cell m . This is different from Equation 5.48 because this is an exact quantity at cell m which does not rely on any discretizations nor approximations. At $t = 0$ it is exactly satisfied since we initialize all experiments in this paper in pressure equilibrium. However, as shown in Equation 5.23 even for a PEP scheme, there will be some temporal error in pressure leading to the 0th order propagation of f_{PE} throughout the simulation through this $R_{PE}|_m$ term. Below, since everything is Taylor expanded around cell m we drop the spacial subscripting notation for the remainder of this section.

$$f_{PE}^{\text{KEEP } \mathcal{O}(2)} = R_{PE} + \frac{\Delta x^2}{6} \left[\frac{\partial^3 \rho e}{\partial x^3} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^3 \rho Y_i}{\partial x^3} \right) \right] + \text{H.O.T.} \quad (5.51)$$

$$\begin{aligned} f_{PE}^{\text{APEC } \mathcal{O}(2)} = & R_{PE} + \frac{\Delta x^2}{6} \left[\frac{\partial^3 \rho e}{\partial x^3} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^3 \rho Y_i}{\partial x^3} \right) \right] \\ & - \frac{\Delta x^2}{4} \sum_{i=0}^N \left[\frac{\partial \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial \rho Y_i}{\partial x} \right] \\ & + \text{H.O.T.} \end{aligned} \quad (5.52)$$

$$f_{PE}^{\text{KEEP } \mathcal{O}(4)} = R_{PE} - \frac{\Delta x^4}{30} \left[\frac{\partial^5 \rho e}{\partial x^5} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^5 \rho Y_i}{\partial x^5} \right) \right] + \text{H.O.T.} \quad (5.53)$$

$$f_{PE}^{\text{APEC } \mathcal{O}(4)} = R_{PE} - \frac{\Delta x^4}{30} \left[\frac{\partial^5 \rho e}{\partial x^5} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^5 \rho Y_i}{\partial x^5} \right) \right] \quad (5.54)$$

$$+ \frac{\Delta x^4}{12} \sum_{i=0}^N \left[\frac{\partial \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x} \frac{\partial^4 \rho Y_i}{\partial x^4} \right. \\ \left. + 2 \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^3 \rho Y_i}{\partial x^3} + 2 \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial \rho Y_i}{\partial x} \right] \quad (5.55)$$

+ H.O.T.

$$f_{PE}^{\text{KEEP } \mathcal{O}(6)} = R_{PE} + \frac{\Delta x^6}{140} \left[\frac{\partial^7 \rho e}{\partial x^7} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^7 \rho Y_i}{\partial x^7} \right) \right] + \text{H.O.T.} \quad (5.56)$$

$$f_{PE}^{\text{APEC } \mathcal{O}(6)} = R_{PE} + \frac{\Delta x^6}{140} \left[\frac{\partial^7 \rho e}{\partial x^7} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^7 \rho Y_i}{\partial x^7} \right) \right] \quad (5.57)$$

$$- \frac{\Delta x^6}{40} \sum_{i=0}^N \left[\frac{\partial \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x} \frac{\partial^6 \rho Y_i}{\partial x^6} + 3 \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^5 \rho Y_i}{\partial x^5} + 5 \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^4 \rho Y_i}{\partial x^4} \right. \\ \left. + 5 \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^3 \rho Y_i}{\partial x^3} + 3 \frac{\partial^5 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^5} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{\partial^6 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^6} \frac{\partial \rho Y_i}{\partial x} \right]$$

+ H.O.T.

$$f_{PE}^{\text{KEEP } \mathcal{O}(8)} = R_{PE} - \frac{\Delta x^8}{630} \left[\frac{\partial^9 \rho e}{\partial x^9} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^9 \rho Y_i}{\partial x^9} \right) \right] + \text{H.O.T.} \quad (5.58)$$

$$f_{PE}^{\text{APEC } \mathcal{O}(8)} = R_{PE} - \frac{\Delta x^8}{630} \left[\frac{\partial^9 \rho e}{\partial x^9} - \sum_{i=0}^N \left(\frac{\partial \rho e}{\partial \rho Y_i} \frac{\partial^9 \rho Y_i}{\partial x^9} \right) \right] \quad (5.59)$$

$$\begin{aligned} & + \frac{\Delta x^8}{630} \sum_{i=0}^N \left[\frac{9}{2} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^8 \rho Y_i}{\partial x^8} + 18 \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^7 \rho Y_i}{\partial x^7} + 42 \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^6 \rho Y_i}{\partial x^6} \right. \\ & + 63 \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^5 \rho Y_i}{\partial x^5} + 63 \frac{\partial^5 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^5} \frac{\partial^4 \rho Y_i}{\partial x^4} + 42 \frac{\partial^6 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^6} \frac{\partial^3 \rho Y_i}{\partial x^3} \\ & \left. + 18 \frac{\partial^7 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^7} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{9}{2} \frac{\partial^8 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^8} \frac{\partial^1 \rho Y_i}{\partial x^1} \right] \\ & + \text{H.O.T.} \end{aligned}$$

In order to simplify these statements, we make an important observation. R_{PE} will propagate any pressure error that is accumulated by either temporal error or an inexact handling of PEP. Therefore, once any pressure error is introduced it will propagate linearly in time through the system. This is not interesting in this analysis because we know, due to APEC being an approximate method, that the error will accumulate. We are interested in the additional terms which show how the spatial discretizations will introduce pressure error and how that compares between the KEEP and APEC schemes. Therefore, we analyze $t = 0$ where $R_{PE}|_m = 0$. This gives us the ability to simplify further by noting that one can take the analytical derivative of $\frac{\partial^n \rho e}{\partial x^n}$ in Equation 5.50 via the chain rule which can be expanded as

$$\frac{\partial^n \rho e}{\partial x^n} = \sum_{i=1}^N \left(\sum_{k=0}^q \binom{q}{k} \frac{d^k}{dx^k} \left(\frac{\partial \rho e}{\partial (\rho Y_i)} \right) \frac{d^{n-k}}{dx^{n-k}} \left(\frac{\partial (\rho Y_i)}{\partial x} \right) \right), \quad (5.60)$$

where q is the order of the scheme, N is the number of species, and n is the

order of derivative. We can use Equation 5.60 to convert the terms from the APEC Taylor expansion and the KEEP Taylor expansion to compare them directly. For orders 2, 4, 6, and 8 we simplify to the following:

$$f_{PE}^{\text{KEEP } \mathcal{O}(2)} = \sum_{i=1}^N \left[\frac{1}{3} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{1}{6} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.61)$$

$$f_{PE}^{\text{APEC } \mathcal{O}(2)} = \sum_{i=1}^N \left[\frac{1}{12} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^2 \rho Y_i}{\partial x^2} - \frac{1}{12} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.62)$$

$$f_{PE}^{\text{KEEP } \mathcal{O}(4)} = \sum_{i=1}^N \left[-\frac{2}{15} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^4 \rho Y_i}{\partial x^4} - \frac{1}{5} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^3 \rho Y_i}{\partial x^3} \right. \\ \left. - \frac{2}{15} \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^2 \rho Y_i}{\partial x^2} - \frac{1}{30} \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.63)$$

$$f_{PE}^{\text{APEC } \mathcal{O}(4)} = \sum_{i=1}^N \left[-\frac{1}{20} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^4 \rho Y_i}{\partial x^4} - \frac{1}{30} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^3 \rho Y_i}{\partial x^3} \right. \\ \left. + \frac{1}{30} \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{1}{20} \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.64)$$

$$+ \frac{1}{30} \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{1}{20} \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.65)$$

$$f_{PE}^{\text{KEEP } \mathcal{O}(6)} = \sum_{i=1}^N \left[\frac{3}{70} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^6 \rho Y_i}{\partial x^6} + \frac{3}{28} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^5 \rho Y_i}{\partial x^5} + \frac{1}{7} \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^4 \rho Y_i}{\partial x^4} \right. \\ \left. + \frac{3}{28} \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^3 \rho Y_i}{\partial x^3} + \frac{3}{70} \frac{\partial^5 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^5} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{1}{140} \frac{\partial^6 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^6} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.66)$$

$$f_{PE}^{\text{APEC } \mathcal{O}(6)} = \sum_{i=1}^N \left[\frac{1}{56} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^6 \rho Y_i}{\partial x^6} + \frac{9}{280} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^5 \rho Y_i}{\partial x^5} + \frac{1}{56} \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^4 \rho Y_i}{\partial x^4} \right. \\ \left. - \frac{1}{56} \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^3 \rho Y_i}{\partial x^3} - \frac{9}{280} \frac{\partial^5 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^5} \frac{\partial^2 \rho Y_i}{\partial x^2} - \frac{1}{56} \frac{\partial^6 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^6} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.67)$$

$$f_{PE}^{\text{KEEP } \mathcal{O}(8)} = \sum_{i=1}^N \left[-\frac{4}{315} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^8 \rho Y_i}{\partial x^8} - \frac{2}{45} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^7 \rho Y_i}{\partial x^7} - \frac{4}{45} \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^6 \rho Y_i}{\partial x^6} \right. \\ \left. - \frac{1}{9} \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^5 \rho Y_i}{\partial x^5} - \frac{4}{45} \frac{\partial^5 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^5} \frac{\partial^4 \rho Y_i}{\partial x^4} - \frac{2}{45} \frac{\partial^6 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^6} \frac{\partial^3 \rho Y_i}{\partial x^3} \right. \\ \left. - \frac{4}{315} \frac{\partial^7 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^7} \frac{\partial^2 \rho Y_i}{\partial x^2} - \frac{1}{630} \frac{\partial^8 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^8} \frac{\partial^1 \rho Y_i}{\partial x^1} \right], \quad (5.68)$$

$$f_{PE}^{\text{APEC } \mathcal{O}(8)} = \sum_{i=1}^N \left[-\frac{1}{180} \frac{\partial^1 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^1} \frac{\partial^8 \rho Y_i}{\partial x^8} - \frac{1}{63} \frac{\partial^2 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^2} \frac{\partial^7 \rho Y_i}{\partial x^7} - \frac{1}{45} \frac{\partial^3 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^3} \frac{\partial^6 \rho Y_i}{\partial x^6} \right. \\ \left. - \frac{1}{90} \frac{\partial^4 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^4} \frac{\partial^5 \rho Y_i}{\partial x^5} + \frac{1}{90} \frac{\partial^5 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^5} \frac{\partial^4 \rho Y_i}{\partial x^4} + \frac{1}{45} \frac{\partial^6 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^6} \frac{\partial^3 \rho Y_i}{\partial x^3} \right. \\ \left. + \frac{1}{63} \frac{\partial^7 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^7} \frac{\partial^2 \rho Y_i}{\partial x^2} + \frac{1}{180} \frac{\partial^8 \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^8} \frac{\partial^1 \rho Y_i}{\partial x^1} \right]. \quad (5.69)$$

The Taylor expansions show that, at a specified order q , the high-order APEC construction yields uniformly smaller leading coefficients in the pressure-error flux f_{PE} than the corresponding KEEP discretization, together with a sign structure that promotes cancellation among mixed derivatives. In particular, comparing the second-, fourth-, sixth-, and eighth-order KEEP forms in Equation 5.61, Equation 5.63, Equation 5.66, and Equation 5.68 with their APEC counterparts in Equation 5.62, Equation 5.64, Equation 5.67, and Equation 5.69 shows strictly smaller magnitude prefactors

multiplying the mixed products $\frac{\partial^n \left(\frac{\partial \rho e}{\partial \rho Y_i} \right)}{\partial x^n} \frac{\partial^m \rho Y_i}{\partial x^m}$ and, for APEC, alternating signs that induce partial cancellation across the stencil. There are three isolated exceptions to the strictly smaller prefactors—two composition-derivative terms in the $\mathcal{O}(8)$ scheme and one first-derivative composition term in the $\mathcal{O}(6)$ scheme—but these constitute a small subset of the many contributing coefficients and do not alter the overall trend that APEC systematically reduces the pressure-error contribution relative to KEEP.

Because APEC corrects each sub-interface in the split flux, the error reduction is spread across the stencil, tightening the bound on $|f_{PE}|$ without lowering the formal order of accuracy. Consequently, for a given mesh and q , the high-order APEC scheme preserves the formal order of the underlying central operator while delivering smaller truncation constants in f_{PE} , implying reduced spurious pressure oscillations and improved near-equilibrium preservation under arbitrary EOS, while temporal propagation through R_{PE} proceeds as in the baseline formulation.

5.4 Derivation of a Fully Conservative PEP Scheme

We now begin the derivation of a fully conservative, spatially exact PEP scheme. Such a scheme must satisfy Equations 5.32 and 5.33 exactly. Here we treat these as separate constraints, rather than combining them into one as done in APEC, which yields two equations for three unknown interface quantities: $\rho Y_1|_{m+\frac{1}{2}}$, $\rho Y_2|_{m+\frac{1}{2}}$, and $\rho e|_{m+\frac{1}{2}}$. This underdetermined system admits multiple possible closures, each corresponding to a distinct conservative PEP variant.

5.4.1 An Example PEP Scheme

The simplest possible variant of this is to let $\rho e|_{m+\frac{1}{2}}$ be equal to the average values of the cells adjacent via

$$\rho e|_{m+\frac{1}{2}} = \frac{\rho e|_m + \rho e|_{m+1}}{2}. \quad (5.70)$$

With this choice of $\rho e|_{m+\frac{1}{2}}$, we are left with a 2×2 system to solve. We define ϵ_i as $\left(\frac{\partial \rho e}{\partial \rho Y_i}\right)_{\rho Y_{j \neq i}, p}$ and note that $\rho e|_{m+\frac{1}{2}}$ is defined in some arbitrary manner and it can be considered a known quantity. We can then write the system as

$$\begin{pmatrix} \epsilon_1|_{m+1} & \epsilon_2|_{m+1} \\ \epsilon_1|_m & \epsilon_2|_m \end{pmatrix} \begin{pmatrix} \rho Y_1|_{m+\frac{1}{2}} \\ \rho Y_2|_{m+\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \rho e|_{m+\frac{1}{2}} - \rho e|_{m+1} + \epsilon_1|_{m+1} \rho Y_1|_{m+1} + \epsilon_2|_{m+1} \rho Y_2|_{m+1} \\ \rho e|_{m+\frac{1}{2}} - \rho e|_m + \epsilon_1|_m \rho Y_1|_m + \epsilon_2|_m \rho Y_2|_m \end{pmatrix}. \quad (5.71)$$

As $\epsilon_i|_{m+1} \rightarrow \epsilon_i|_m$, the two rows of the coefficient matrix in Equation 5.71 become nearly linearly dependent and its determinant tends to zero. In this regime the system is increasingly ill-conditioned. An equivalent conclusion is obtained if one treats a mass fraction as a free variable and instead solves for $\rho Y_1|_{m+\frac{1}{2}}$ and $\rho e|_{m+\frac{1}{2}}$. We therefore expect the linear system Equation 5.71 to be numerically unreliable when the left and right states are nearly equal.

5.4.2 Equal/near-equal states

As we have seen, as $|\epsilon_i|_{m+1} - \epsilon_i|_m| \rightarrow 0$ the PEP scheme becomes unstable, but an important realization is this is exactly where the scheme is not needed. Assume uniform pressure and $\epsilon_i|_m = \epsilon_i|_{m+1} = \epsilon_i^*$ we then rewrite Equations 5.32 and 5.33 as

$$\rho e|_{m+\frac{1}{2}} - \rho e|_m = \sum_{i=1}^N \epsilon_i^* \left(\rho Y_i|_{m+\frac{1}{2}} - \rho Y_i|_m \right), \quad (5.72)$$

$$\rho e|_{m+1} - \rho e|_{m+\frac{1}{2}} = \sum_{i=1}^N \epsilon_i^* \left(\rho Y_i|_{m+1} - \rho Y_i|_{m+\frac{1}{2}} \right). \quad (5.73)$$

Adding the two gives the cell-centered relation

$$\rho e|_{m+1} - \rho e|_m = \sum_{i=1}^N \epsilon_i^* \left(\rho Y_i|_{m+1} - \rho Y_i|_m \right), \quad (5.74)$$

so the two interface constraints are consistent with the single condition at cell centers.

When $\epsilon_i|_m = \epsilon_i|_{m+1}$, this cell-centered relation is exact, and the centered averages

$$\rho e|_{m+\frac{1}{2}} = \frac{1}{2} (\rho e|_m + \rho e|_{m+1}), \quad \rho Y_i|_{m+\frac{1}{2}} = \frac{1}{2} (\rho Y_i|_m + \rho Y_i|_{m+1}) \quad (5.75)$$

satisfy each interface equation exactly, since both sides equal one half of the corresponding cell-centered difference.

5.4.3 Proposed PEP Scheme

In the design of the proposed PEP scheme, we keep in mind that the PEP linear system (5.71) becomes ill-conditioned in regions where $|\epsilon_i|_{m+1} \approx |\epsilon_i|_m$ —precisely where strong composition jumps are absent and a PEP correction is not essential. We therefore make the following practical decision: rather than solving directly for an unknown interface value $\phi|_{m+\frac{1}{2}}$, we choose to assume the form

$$\phi|_{m+\frac{1}{2}} = \frac{\phi|_m + \phi|_{m+1}}{2} + C \quad (5.76)$$

where C is now the unknown constant we are solving. This has two benefits. First, since there are numerical instabilities related to solving the system 5.71, this provides a straightforward way to recover stability through the correction term C . Rather than trying to solve for $\phi|_{m+\frac{1}{2}}$ which is numerically unstable we let C be unstable and can easily let this be 0 in regions where we detect this is needed. The second benefit to this is we can modify any numerical scheme to be pressure equilibrium preserving. When these coefficients are zero, we will exactly recover the numerical scheme of our choosing. With the goal of pressure equilibrium, we let this underlying method be APEC [142]. We will now present the proposed half point values of the scheme

$$\rho Y_1|_{m+\frac{1}{2}}^{\text{PEP}} = \frac{\rho Y_1|_m + \rho Y_1|_{m+1}}{2} + \alpha, \quad (5.77)$$

$$\rho Y_2|_{m+\frac{1}{2}}^{\text{PEP}} = \frac{\rho Y_2|_m + \rho Y_2|_{m+1}}{2} - \alpha, \quad (5.78)$$

$$\begin{aligned} \rho e|_{m+\frac{1}{2}}^{\text{PEP}} &= \frac{\rho e|_m + \rho e|_{m+1}}{2} + \beta \\ &\quad - \sum_{i=1}^N \left[\frac{\epsilon_i|_{m+1}}{2} \left(\rho Y_i|_{m+1} - \rho Y_i|_{m+\frac{1}{2}} \right) - \frac{\epsilon_i|_m}{2} \left(\rho Y_i|_{m+\frac{1}{2}} - \rho Y_i|_m \right) \right]. \end{aligned} \quad (5.79)$$

For consistency between mass fraction we choose to modify $\rho e|_{m+\frac{1}{2}}$ as well. This scheme also maintains the flux of ρ exactly as the KEEP scheme, unlike [55]. This decomposition of the flux has a very useful feature in analyzing the determinate limit of the formulation because it is now isolated only in α and β , which we only need to prove that their limit is zero for consistency. Furthermore, this decomposition allows for the most simple treatment of the near-limiting behavior, where floating point error

can pollute the calculation of the fraction where both numerator and denominator are near zero. The key to this simple treatment is that the choice of the leading term is arbitrary, as long as it represents a consistent flux.

We now have two unknowns, α and β , paired with our two constraints, Equations 5.32 and 5.33. We can solve these analytically and write the algebraic closed form as:

$$\alpha = \frac{\sum_{i=1}^N (\epsilon_i|_m + \epsilon_i|_{m+1}) (\rho Y_i|_m - \rho Y_i|_{m+1}) + 2(\rho e|_{m+1} - \rho e|_m)}{2[(\epsilon_1|_m - \epsilon_1|_{m+1}) - (\epsilon_2|_m - \epsilon_2|_{m+1})]}, \quad (5.80)$$

$$\beta = \frac{[(\epsilon_1|_m + \epsilon_1|_{m+1}) - (\epsilon_2|_m + \epsilon_2|_{m+1})] \left[\sum_{i=1}^N (\epsilon_i|_m + \epsilon_i|_{m+1}) (\rho Y_i|_m - \rho Y_i|_{m+1}) - 2(\rho e|_m - \rho e|_{m+1}) \right]}{4[(\epsilon_1|_m - \epsilon_1|_{m+1}) - (\epsilon_2|_m - \epsilon_2|_{m+1})]}. \quad (5.81)$$

As expected, α and β inherit the same numerical instability problem because their denominator will approach 0 as $\epsilon_i|_m \approx \epsilon_i|_{m+1}$. From here there are two equivalent approaches, one could design some switching where based on some tolerance based on $|\epsilon_i|_{m+1} - \epsilon_i|_m|$ one can either choose to evaluate α, β algebraically if it is numerically safe or let $\alpha, \beta = 0$ and recover the second order APEC scheme. The second approach and the one we will take in this work is we retain the system of equations but rewrite such that we solve for α, β directly

$$\begin{bmatrix} \epsilon_{1|m+1} - \epsilon_{2|m+1} & -1 \\ -\epsilon_{1|m} + \epsilon_{2|m} & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \\
\begin{bmatrix} \sum_i^N \left[\epsilon_{i|m+1} \left(-\frac{1}{2} \rho Y_{i|m} + \frac{1}{2} \rho Y_{i|m+1} \right) - \left(-\frac{\epsilon_{i|m}}{2} + \frac{\epsilon_{i|m+1}}{2} \right) \frac{-\rho Y_{i|m} + \rho Y_{i|m+1}}{2} \right] + \frac{1}{2} (\rho e_{|m} - \rho e_{|m+1}) \\ \sum_i^N \left[\epsilon_{i|m} \left(-\frac{1}{2} \rho Y_{i|m} + \frac{1}{2} \rho Y_{i|m+1} \right) + \left(-\frac{\epsilon_{i|m}}{2} + \frac{\epsilon_{i|m+1}}{2} \right) \frac{-\rho Y_{i|m} + \rho Y_{i|m+1}}{2} \right] + \frac{1}{2} (\rho e_{|m} - \rho e_{|m+1}) \end{bmatrix} \quad (5.82)$$

Solving this system requires using a pseudo-inverse algorithm for numerical stability. We leverage the Moore-Penrose [1] algorithm which has a reciprocal condition number parameter. This is a tolerance parameter used to determine which singular values should be treated as effectively zero. For the purposes of the PEP scheme, this parameter controls how smoothly we transition between the fully conservative PEP update and the approximate APEC flux in space. Larger values produce more gradual spatial changes in the flux formulation, which can improve robustness by avoiding abrupt local switches but also enlarge the region where the lower-accuracy APEC approximation is active; smaller values sharpen the transition, preserving the high-order PEP formulation over a wider portion of the domain but making the scheme more sensitive to roundoff and localized instability. Over-utilizing the APEC correction can therefore introduce an unnecessary loss of accuracy, while under-utilizing it risks allowing small pressure imbalances to persist and grow into long-time pressure oscillations. In practice, we bias this parameter toward the conservative side, since the scheme still performs well with moderate APEC utilization and long-term stability of the pressure field is a key metric in this work. There is some flexibility to this choice, but through some analysis we can derive a safe choice which we will utilize.

In order to choose a single global tolerance r_g we look at the conditioning of the local 2×2 system used to compute (α, β) . At each cell we form

$$\theta|_m \equiv \epsilon_1|_m - \epsilon_2|_m, \quad (5.83)$$

since θ is the coefficient that controls when the two interface constraints become nearly redundant in Equation 5.82. Because the magnitude of $|\theta|$ can vary widely across EOS and thermodynamic states, we normalize the θ -column only for the conditioning estimate by introducing

$$s|_{m+\frac{1}{2}} = \sqrt{\max(|\theta|_m|, |\theta|_{m+1}|, 1)}, \quad \tilde{A}|_{m+\frac{1}{2}} = \begin{bmatrix} \theta|_{m+1} / s|_{m+\frac{1}{2}} & -1 \\ -\theta|_m / s|_{m+\frac{1}{2}} & 1 \end{bmatrix}. \quad (5.84)$$

We then define the symmetric matrix

$$G|_{m+\frac{1}{2}} \equiv \tilde{A}^\top|_{m+\frac{1}{2}} \tilde{A}|_{m+\frac{1}{2}}, \quad (5.85)$$

whose eigenvalues $0 \leq \lambda_{\min} \leq \lambda_{\max}$ provide a direct measure of conditioning.

If $\tilde{A}|_{m+\frac{1}{2}}$ is singular then $\lambda_{\min} = 0$, and if it is nearly singular then $\lambda_{\min} \ll \lambda_{\max}$. In particular, the condition number of $G|_{m+\frac{1}{2}}$ can be written as

$$\kappa(G|_{m+\frac{1}{2}}) = \frac{\lambda_{\max}(G|_{m+\frac{1}{2}})}{\lambda_{\min}(G|_{m+\frac{1}{2}})}, \quad (5.86)$$

and we define the local reciprocal condition number

$$r|_{m+\frac{1}{2}} = \frac{\lambda_{\min}(G|_{m+\frac{1}{2}})}{\lambda_{\max}(G|_{m+\frac{1}{2}})}, \quad (5.87)$$

which decreases toward 0 as the local solve becomes more ill-conditioned. In order to choose a single global tolerance, we simply take the maximum over all faces at the initial

condition

$$r_g = \max_{m \in Nx} r|_{m+\frac{1}{2}}, \quad (5.88)$$

and then enforce a small lower bound $r_g \geq 10^{-13}$ to avoid effectively vanishing tolerances in nearly singular regions. This value is then used in the Moore–Penrose pseudoinverse when solving the original (unscaled) system $A(\alpha, \beta)^\top = b$ at every interface. In our implementation, r_g is computed once from the initial condition and held fixed for the remainder of the simulation. Taking the maximum over faces, together with the lower bound, provides a simple, robust choice: it admits slightly more pressure error than strictly necessary in some regions, but avoids introducing additional local switching and performs well for all test cases considered here. For more complex problems, one could instead recompute r periodically (e.g. each timestep) or allow a space-dependent $r|_{m+\frac{1}{2}}$ to better balance accuracy and robustness. For additional information on the sensitivity to this parameter and values used in this study, we point the reader to 5.8.

5.4.4 Extension to Multiple Dimensions and N Species

The construction of the proposed PEP scheme as well as the high-order APEC scheme generalizes directly to d spatial dimensions and N species. Because the correction acts only on the mass fraction and total–energy fluxes, the momentum flux is advanced with the standard multidimensional discretization and no additional terms are required. Extending the scheme to N species requires modifying only two species to enforce the criterion in Equations 5.32 and 5.33 while leaving all remaining components unchanged. The choice of which two to correct is flexible, though we recommend selecting the two species with the largest mass fractions.

5.4.5 Summary of Proposed PEP Scheme

We summarize the proposed PEP scheme as a modification to the KEEP scheme where $\rho Y_1|_{m+\frac{1}{2}}^{\text{PEP}}$, $\rho Y_2|_{m+\frac{1}{2}}^{\text{PEP}}$ and $\rho e|_{m+\frac{1}{2}}^{\text{PEP}}$ are defined in Equations 5.77, 5.78, and 5.79 respectively. We let $\rho|_{m+\frac{1}{2}}^{\text{PEP}} = \rho Y_1|_{m+\frac{1}{2}}^{\text{PEP}} + \rho Y_2|_{m+\frac{1}{2}}^{\text{PEP}}$.

$$C_1|_{m+\frac{1}{2}} = \rho Y_1|_{m+\frac{1}{2}}^{\text{PEP}} \frac{u|_m + u|_{m+1}}{2} \quad (5.89)$$

$$C_2|_{m+\frac{1}{2}} = \rho Y_2|_{m+\frac{1}{2}}^{\text{PEP}} \frac{u|_m + u|_{m+1}}{2} \quad (5.90)$$

$$M|_{m+\frac{1}{2}} = \rho|_{m+\frac{1}{2}}^{\text{PEP}} \frac{u|_m + u|_{m+1}}{2} \frac{u|_m + u|_{m+1}}{2} \quad (5.91)$$

$$\Pi|_{m+\frac{1}{2}} = \frac{p|_m + p|_{m+1}}{2} \quad (5.92)$$

$$K|_{m+\frac{1}{2}} = \rho|_{m+\frac{1}{2}} \frac{u|_m + u|_{m+1}}{2} \frac{u|_m u|_{m+1}}{2} \quad (5.93)$$

$$I|_{m+\frac{1}{2}} = \rho e|_{m+\frac{1}{2}}^{\text{PEP}} \frac{u|_m + u|_{m+1}}{2} \quad (5.94)$$

$$S|_{m+\frac{1}{2}} = \frac{u|_m p|_{m+1} + u|_{m+1} p|_m}{2} \quad (5.95)$$

5.4.6 Limiting behavior and consistency of the scheme

We now consider the limiting behavior of α and β as shown in Equation 5.80 and 5.81. The goal is to show that these quantities approach 0 as the left and right state are equal to each other. This will then recover the original APEC scheme in the near equal limit and the scheme will be consistent. We analyze the numerator and denominator of each coefficient separately.

Numerator of α Starting with the numerator of α we define

$$\text{num}_\alpha = \sum_i^N [(\epsilon_i|_m + \epsilon_i|_{m+1})(\rho Y_i|_m - \rho Y_i|_{m+1})] + 2(\rho e|_{m+1} - \rho e|_m). \quad (5.96)$$

We also define $\delta_i \equiv \rho Y_i|_{m+1} - \rho Y_i|_m$ and we Taylor expand ϵ at cell $m+1$ around cell m as

$$\epsilon_i|_{m+1} = \epsilon_i|_m + \sum_j^N \xi_{i,j}|_m \delta_j + O(\delta^2). \quad (5.97)$$

Similarly we expand $\rho e|_{m+1}$ as

$$\rho e|_{m+1} = \rho e|_m + \sum_j \epsilon_j|_m \delta_j + O(\delta^2). \quad (5.98)$$

Plugging Equation 5.97 and 5.98 into Equation 5.96 we obtain

$$\text{num}_\alpha = \sum_i -2 \epsilon_i|_m \delta_i - \sum_j \xi_j|_m \delta_j^2 + \sum_i 2 \epsilon_i|_m \delta_i. \quad (5.99)$$

Which after simplification implies $\text{num}_\alpha \propto O(\delta^2)$.

Numerator of β We analyze β the same way where we define

$$\begin{aligned} \text{num}_\beta = & \left[(\epsilon_1|_m + \epsilon_1|_{m+1}) - (\epsilon_2|_m + \epsilon_2|_{m+1}) \right] \\ & \left(\sum_{i=1}^N (\epsilon_i|_m + \epsilon_i|_{m+1})(\rho Y_i|_m - \rho Y_i|_{m+1}) - 2(\rho e|_m - \rho e|_{m+1}) \right). \end{aligned} \quad (5.100)$$

We again use the Equations 5.97 and 5.98 and substitute these into 5.100 and collecting like terms, all contributions at order $\mathcal{O}(\delta)$ cancel:

$$\text{num}_\beta = -2(\epsilon_1|_m - \epsilon_2|_m) \sum_{i,j} \xi_{i,j}|_m \delta_i \delta_j + \mathcal{O}(\delta^3). \quad (5.101)$$

This implies $\text{num}_\beta \propto \mathcal{O}(\delta^2)$.

Denominator of β For the denominator of β we similarly define

$$\text{denom}_\beta = 4 \left[(\epsilon_1|_m - \epsilon_1|_{m+1}) - (\epsilon_2|_m - \epsilon_2|_{m+1}) \right]. \quad (5.102)$$

Using the same Taylor expansion (5.97) for $\epsilon_i|_{m+1}$ and expressing the result in terms of δ_j , we obtain

$$\begin{aligned} \text{denom}_\beta &= 4 \left[- \sum_j \xi_{1,j}|_m \delta_j + \sum_j \xi_{2,j}|_m \delta_j \right] + \mathcal{O}(\delta^2) \\ &= -4 \sum_j (\xi_{1,j}|_m - \xi_{2,j}|_m) \delta_j + \mathcal{O}(\delta^2). \end{aligned} \quad (5.103)$$

This then implies $\text{denom}_\beta \propto \mathcal{O}(\delta)$. Combining (5.101) and (5.103), we find

$$\beta = \frac{\text{num}_\beta}{\text{denom}_\beta} \propto \frac{\mathcal{O}(\delta^2)}{\mathcal{O}(\delta)} = \mathcal{O}(\delta), \quad (5.104)$$

and in the limit of vanishing δ

$$\lim_{\delta \rightarrow 0} \beta = 0. \quad (5.105)$$

An analogous argument applies to α . From the analysis above we already know that $\text{num}_\alpha \propto \mathcal{O}(\delta^2)$, and the denominator of α differs from denom_β only by a constant factor of 2, so that $\text{denom}_\alpha \propto \mathcal{O}(\delta)$ as well. Hence

$$\alpha = \frac{\text{num}_\alpha}{\text{denom}_\alpha} \propto \frac{\mathcal{O}(\delta^2)}{\mathcal{O}(\delta)} = \mathcal{O}(\delta), \quad (5.106)$$

and in the limit of vanishing δ we likewise obtain

$$\lim_{\delta \rightarrow 0} \alpha = 0. \quad (5.107)$$

Therefore, in the limit $\delta \rightarrow 0$ both correction coefficients vanish, $\alpha, \beta \rightarrow 0$, and the proposed scheme reduces exactly to APEC.

5.5 Taylor Series Expansion

Due to the numerical instabilities of α and β in the PEP scheme as defined in Equation 5.80 and 5.81, we consider introducing an approximation to these parameters via a Taylor series expansion which will remain numerically stable even when $\rho Y_1 = \rho Y_2$.

5.5.1 Taylor Expansion Method

We adopt the half-point forms introduced previously in the proposed PEP scheme with Equations 5.77, 5.78, and 5.79 and use the same analytic expressions for α and β whose closed forms are given in Equations 5.80 and 5.81. As a reminder, both variables can suffer numerical instabilities in the limit $\epsilon_i|_m \rightarrow \epsilon_i|_{m+1}$. Although the method is consistent, at finite precision small oscillations may appear. Here we regularize by Taylor expanding the state at cell $m + 1$ about cell m .

We expand the coefficient fields in composition space to order n :

$$\epsilon_i|_{m+1} = \sum_{k=0}^n \frac{1}{k!} \sum_{j_1, \dots, j_k} \partial_{j_1} \cdots \partial_{j_k} \epsilon_i|_m \prod_{\ell=1}^k \Delta(\rho Y_{j_\ell}) + \mathcal{O}(\|\Delta \rho \mathbf{Y}\|^{n+1}), \quad (5.108)$$

$$\rho e|_{m+1} = \sum_{k=0}^n \frac{1}{k!} \sum_{j_1, \dots, j_k} \partial_{j_1} \cdots \partial_{j_k} (\rho e)|_m \prod_{\ell=1}^k \Delta(\rho Y_{j_\ell}) + \mathcal{O}(\|\Delta \rho \mathbf{Y}\|^{n+1}), \quad (5.109)$$

where $\partial_j \equiv \partial/\partial(\rho Y_j)$ and $\Delta(\rho Y_i) \equiv \rho Y_i|_{m+1} - \rho Y_i|_m$. We denote these truncated evaluations by

$$\tilde{\epsilon}_i^{(n)} \equiv \epsilon_i|_{m+1} \Big|_{\text{Taylor-}n}, \quad \tilde{\rho e}^{(n)} \equiv \rho e|_{m+1} \Big|_{\text{Taylor-}n}.$$

The closed forms in Equations 5.80 and 5.81 can be regarded as functions of the state at cells m and $m + 1$ which we write as

$$\alpha|_{m+\frac{1}{2}} = \alpha(\rho Y_1|_m, \rho Y_2|_m, \rho e|_m, \epsilon_1|_m, \epsilon_2|_m, \rho Y_1|_{m+1}, \rho Y_2|_{m+1}, \rho e|_{m+1}, \epsilon_1|_{m+1}, \epsilon_2|_{m+1}), \quad (5.110)$$

$$\beta|_{m+\frac{1}{2}} = \beta(\rho Y_1|_m, \rho Y_2|_m, \rho e|_m, \epsilon_1|_m, \epsilon_2|_m, \rho Y_1|_{m+1}, \rho Y_2|_{m+1}, \rho e|_{m+1}, \epsilon_1|_{m+1}, \epsilon_2|_{m+1}). \quad (5.111)$$

We now define the n^{th} order Taylor-regularized approximations by replacing only the $m+1$ inputs with their Taylor surrogates about cell m and enforcing a proportional-composition relation,

$$\widetilde{\rho Y_1|_{m+1}} = \rho Y_1|_m + C \left(\rho Y_2|_{m+1} - \rho Y_2|_m \right),$$

Substituting these surrogates into 5.80 and 5.81 shows that, under the proportional-composition relation, the factors generated by the Taylor-expanded $\tilde{\epsilon}_i^{(n)}$, $\tilde{\rho e}^{(n)}$ and the proportional update of ρY_1 enter both the numerators and denominators with matching powers; hence the common factors cancel identically, removing the singularity as $\epsilon_i|_{m+1} \rightarrow \epsilon_i|_m$.

Finally, we let $\tilde{\epsilon}_i^{(n)}$ and $\tilde{\rho e}^{(n)}$ denote the n -th order Taylor expansions of ϵ_i and ρe about cell m . Plugging this into Equations 5.110 and 5.111 we define the n -th order Taylor expansion of α and β as

$$\alpha|_{m+\frac{1}{2}} \approx \widetilde{\alpha^{(n)}}|_{m+\frac{1}{2}} \equiv \alpha \left(\rho Y_1|_m, \rho Y_2|_m, \widetilde{\rho Y_1|_{m+1}}, \rho Y_2|_{m+1}, \rho e|_m, \tilde{\rho e}^{(n)}, \epsilon_1|_m, \epsilon_2|_m, \tilde{\epsilon}_1^{(n)}, \tilde{\epsilon}_2^{(n)} \right), \quad (5.112)$$

$$\beta|_{m+\frac{1}{2}} \approx \widetilde{\beta^{(n)}}|_{m+\frac{1}{2}} \equiv \beta \left(\rho Y_1|_m, \rho Y_2|_m, \widetilde{\rho Y_1|_{m+1}}, \rho Y_2|_{m+1}, \rho e|_m, \tilde{\rho e}^{(n)}, \epsilon_1|_m, \epsilon_2|_m, \tilde{\epsilon}_1^{(n)}, \tilde{\epsilon}_2^{(n)} \right). \quad (5.113)$$

Since we use the same half point definitions as in Equations 5.89-5.95 total mass, momentum, and energy remain exactly conserved. Taking $\alpha=\beta=0$ this scheme will recover the underlying APEC fluxes preserving consistency with the proposed PEP

scheme. Most importantly, the numerical instability in the closed forms 5.80–5.81 is eliminated by the Taylor regularization with the proportionality constraint.

We have now introduced a Taylor-regularized family that we will denote Taylor- (n) . This label refers solely to the order of the Taylor series retained in the surrogate inputs, i.e. the highest power of $\Delta(\rho Y_i)$ included with $n = 1, \dots, 5$ and does not indicate the formal convergence order of the overall scheme. In all results below, we employ the same second-order base discretization as in the proposed PEP scheme, while varying $n \in \{1, \dots, 5\}$ to study the effect of including successively higher-order $\Delta(\rho Y_i)$ terms in the Taylor surrogates.

5.6 Results

We now present multiple varying interface problems with a uniform pressure initialization and various equation of states to show the pressure equilibrium properties of the proposed schemes.

5.6.1 1D smooth interfaces advection problem - calorically perfect gas

We begin with the simulation of a calorically perfect gas. Such a gas is governed by the equation of state

$$\rho e = \frac{P}{\bar{\gamma} - 1}, \quad (5.114)$$

where $\bar{\gamma}$ is the specific heat ratio of the mixture which can be written as

$$\frac{1}{\bar{\gamma} - 1} = \bar{M} \sum_{i=0}^N \frac{1}{\gamma_i - 1} \frac{Y_i}{M_i} \quad (5.115)$$

where N is the number of species, γ_i and M_i are the ratio of specific heats and the molecular weight of species i respectively.

We run a smooth two-fluid interface test where fluid 1 is a diatomic (air-like) species with $\gamma = 1.4$ and $M = 28$, g/mol, and fluid 2 is a monatomic (helium-like) species with $\gamma = 1.66$ and $M = 4$, g/mol.

The mass fractions start from a sinusoid, while velocity and pressure are set to uniform:

$$\begin{pmatrix} (\rho Y_1)_0 \\ (\rho Y_2)_0 \\ u_0 \\ p_0 \end{pmatrix} = \begin{pmatrix} \frac{w_1}{2} (1 - \sin(kx)) + 0.1 \\ \frac{w_2}{2} (1 + \sin(kx)) + 0.1 \\ 1.0 \\ 0.9 \end{pmatrix}, \quad (5.116)$$

with $(w_1, w_2) = (0.6, 0.2)$ and $k = 4\pi$. We use a 4th order Runge-Kutta time integration scheme with a CFL 0.6 and a periodic domain $x \in [0, 1]$ with 501 grid points. The setup starts in exact pressure/velocity equilibrium, so any drift we see later is purely numerical.

Figure 5.2 shows four second-order solutions at $t = 20$ of the ideal-gas test case described in Equation 5.116. The top-left is the proposed PEP scheme in Equations 5.89-5.95, top-right is APEC as proposed in [142], bottom-left is the Fujiwara [55] PEP scheme, and the bottom-right is the $\mathcal{O}(2)$ KEEP scheme which we will refer to as FC-NPE, fully conservative non pressure equilibrium. The first three look good with no sign of spurious pressure oscillations while the FC-NPE scheme is being dominated by

pressure error at this time.

To track this quantitatively, we use a relative L_2 pressure error over the N interior cells:

$$\varepsilon_p(t) = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\frac{p_i(t)}{p_0} - 1 \right)^2} \quad (5.117)$$

In Figure 5.3, the pressure error of FC-NPE (orange) grows quickly. APEC $\mathcal{O}(2)$ stays around 10^{-6} . The Fujiwara scheme is spatially exact and holds near $\approx 10^{-12}$. Our PEP scheme also remains near 10^{-12} with a small increase at later times due to an accumulation of errors introduced by the pseudoinverse switch that toggles between APEC and PE behavior.

We also monitor total energy using the same relative L_2 formulation:

$$\varepsilon_E(t) = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\frac{E_i(t)}{E_0} - 1 \right)^2} \quad (5.118)$$

Figure 5.3 shows energy is conserved for all methods except the double-flux approach, which deviates from the rest.

We next compare the higher-order formulations in Figure 5.4, which shows relative L_2 pressure error (left) and total-energy error (right) for APEC and KEEP (NPE) at second, fourth, sixth, and eighth order. Higher accuracy comes with wider stencils, and for APEC each increase in formal order yields a clear reduction in pressure error. Because APEC is approximate, it does not enforce pressure equilibrium exactly; instead, the imbalance decreases systematically with order, becoming very small by about $\mathcal{O}(6)$, close to the fully conservative baseline. In contrast, the corresponding NPE variants exhibit growing pressure error at higher orders, highlighting that pressure-equilibrium preservation remains important even for wide-stencil, high-order schemes.

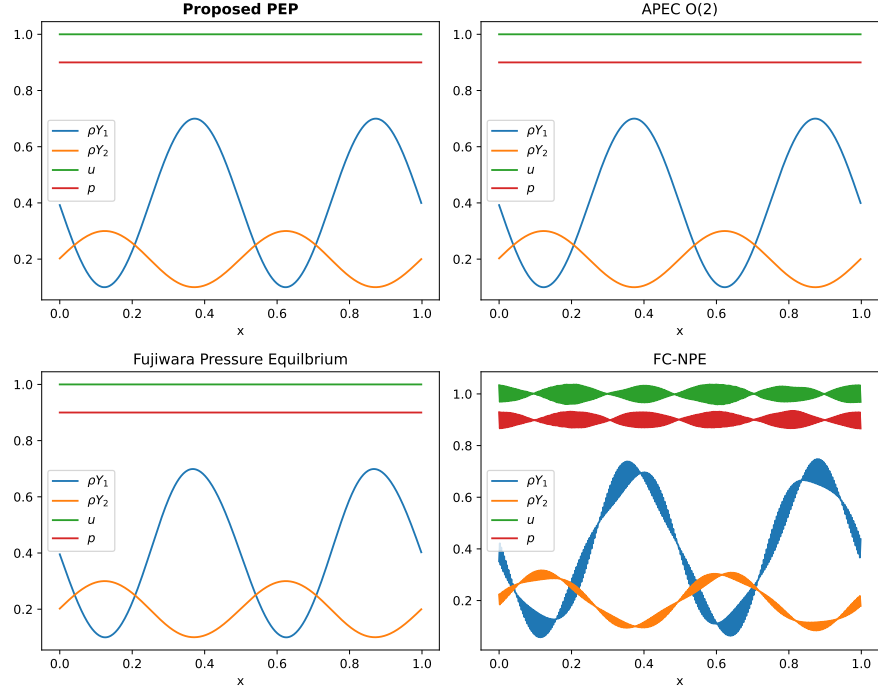


Figure 5.2: Second-order numerical solutions at $t = 20$ for the ideal-gas test case defined in Equation 5.116. The four panels show: (top-left) the proposed PEP scheme; (top-right) the APEC scheme of [142]; (bottom-left) the PEP method of [55]; and (bottom-right) the second-order KEEP scheme (FC-NPE). In each panel, blue, orange, green, and red denote ρY_1 , ρY_2 , u , and p , respectively.

On the energy side, all methods are fully conservative, so the relative errors reported in the right panel fluctuate around zero at the level of numerical precision. Notably, the APEC $\mathcal{O}(2)$ scheme is already very stable, and the higher-order APEC variants inherit this robustness while further reducing the pressure error, with the $\mathcal{O}(6)$ and $\mathcal{O}(8)$ forms essentially at numerical precision and remaining accurate for longer times than the PEP scheme, albeit at the cost of a larger stencil and higher formal order. We will no longer show energy-conservation plots for subsequent test cases, as they provide no additional

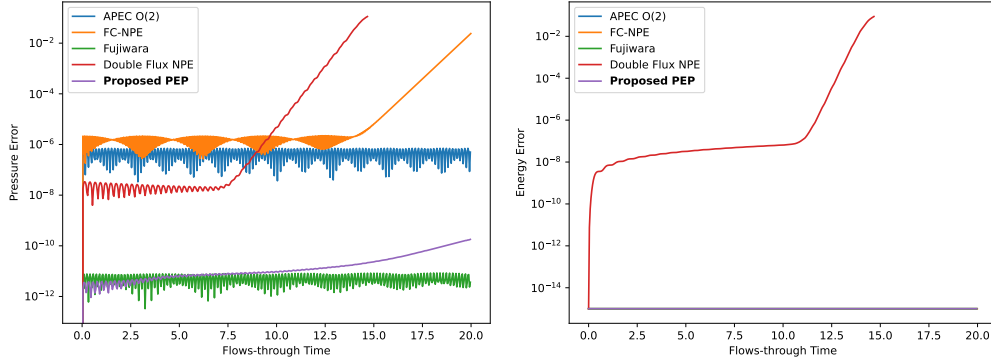


Figure 5.3: L_2 pressure error (left) and total-energy error (right) for the ideal-gas test case. Curves correspond to the APEC $\mathcal{O}(2)$ scheme (blue), the FC-NPE scheme (orange), the Fujiwara PEP scheme (green), the double-flux APE scheme (red), and the proposed PEP scheme (purple).

insight once full conservation is verified.

5.6.2 1D smooth interfaces advection problem - Stiffened Gas

We now consider a multi-species stiffened-gas, where the equation of state can be written as

$$\rho e = \frac{p}{\bar{\gamma} - 1} + \bar{A} + \rho \bar{Q}, \quad (5.119)$$

where $\bar{\gamma}$ is defined in Equation 5.115, $\bar{A}$ is the mixture stiffening-pressure energy offset (pseudo-pressure contribution), defined as

$$\bar{A} = \bar{M} \sum_{i=0}^N \frac{\gamma_i p_{\infty,i}}{\gamma_i - 1} \frac{Y_i}{M_i}, \quad (5.120)$$

where $p_{\infty,i}$ the stiffening pressure of species i . $\bar{Q}$ is the mixture reference specific internal energy (heat of formation) contribution, defined as

$$\bar{Q} = \bar{M} \sum_{i=0}^N \frac{q_i}{M_i} Y_i. \quad (5.121)$$

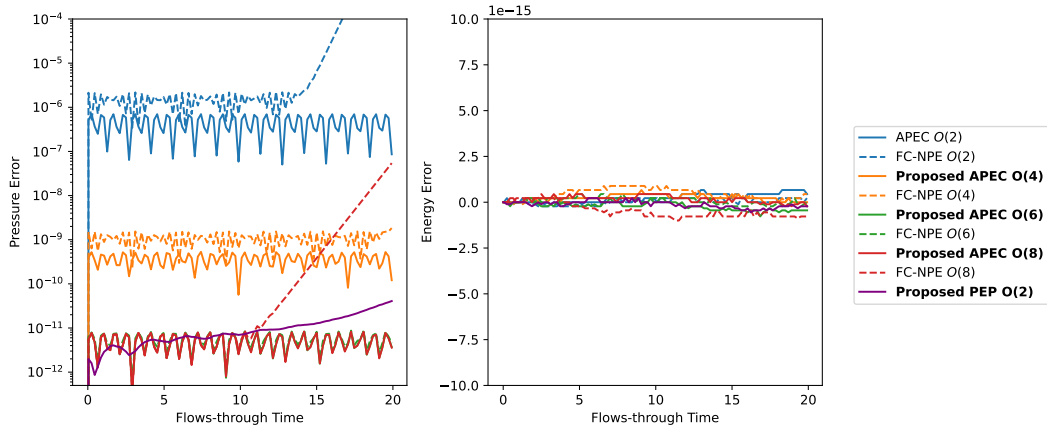


Figure 5.4: L_2 pressure error (left) and total-energy error (right) for the higher-order APEC and NPE schemes applied to the ideal-gas test. APEC curves are plotted as solid lines and NPE curves as dashed lines, with colors denoting the formal order of accuracy: $\mathcal{O}(2)$ (blue), $\mathcal{O}(4)$ (orange), $\mathcal{O}(6)$ (green), and $\mathcal{O}(8)$ (red). The proposed PEP scheme is shown in purple for reference.

where q_i an energy offset of species i .

We define the two fluids as: fluid 1 (water-like) with $\gamma_1 = 3$, $p_{\infty,1} = 0.1$, $q_1 = -0.1$, $M_1 = 18$ g/mol; and fluid 2 (air-like) with $\gamma_2 = 1.4$, $p_{\infty,2} = 0$, $q_2 = 0$, $M_2 = 29$ g/mol. We consider the same initial-value problem as in Eq. 5.116, taking $(w_1, w_2) = (0.6, 0.2)$ and $k = 4\pi$. Time integration uses classical RK4 with CFL 0.6 on a periodic domain $x \in [0, 1]$ with 501 grid points.

Note that, to the authors' knowledge, there is no fully conservative PEP scheme that accommodates $q_i \neq 0$ other than the one proposed here. The formulation of [55] covers both ideal-gas and stiffened-gas EOS but is not compatible when the q parameter is nonzero. We therefore deliberately choose one fluid with $q \neq 0$ to expose the consequences of this incompatibility and to assess the behavior of the proposed method, APEC, and a non-pressure-equilibrium (NPE/KEEP) baseline under identical conditions.

Figure 5.5 shows the same four second-order methods as before. In the bottom-right panel, the NPE solution exhibits clear pressure oscillations. The other three methods look clean at this scale, with no visible pressure artifacts.

Figure 5.6 zooms in on the pressure equilibrium error. The Fujiwara scheme is no longer at numerical precision because it is incompatible with $q \neq 0$. Our proposed method attains numerical precision. As in the ideal-gas case, APEC reduces error relative to NPE, while NPE quickly diverges. Our method shows a slight increase in pressure error due to the Moore–Penrose pseudoinverse which can introduce some pressure error as well as small oscillations, consistent with the behavior observed in the ideal-gas case.

Figure 5.7 compares the higher-order APEC and NPE (KEEP) formulations for the stiffened-gas system. As with the ideal-gas case, the NPE schemes begin with modest pressure error at low order, but this error eventually grows and dominates the solution, leading to rapid degradation at later times. In contrast, all APEC variants remain stable, and each increase in formal order yields a corresponding reduction in pressure imbalance. The $\mathcal{O}(6)$ and $\mathcal{O}(8)$ APEC schemes achieve errors approaching numerical precision and maintain this accuracy over long integration times, outlasting the PEP method, whose oscillations arise from the pseudoinverse-based switching mechanism.

5.6.3 1D smooth interfaces advection problem — van der Waals

For the last test problem we adopt a van der Waals mixture EOS with classical mixing rules. The EOS is

$$\left(p + \frac{\bar{a}}{v^2}\right)(v - \bar{b}) = RT, \quad (5.122)$$

where v is the specific volume and R is the universal gas constant. The mole fraction of species i is

$$X_i = \frac{Y_i/M_i}{\sum_{j=1}^N Y_j/M_j}, \quad (5.123)$$

with $N = 2$ in this study. The mixture parameters are

$$\bar{a} = X_1^2 a_{11} + 2X_1 X_2 a_{12} + X_2^2 a_{22}, \quad (5.124)$$

$$\bar{b} = X_1 b_1 + X_2 b_2. \quad (5.125)$$

The pure-component constants are taken from critical data,

$$a_{ii} = \frac{27(RT_{c,i})^2}{64P_{c,i}}, \quad (5.126)$$

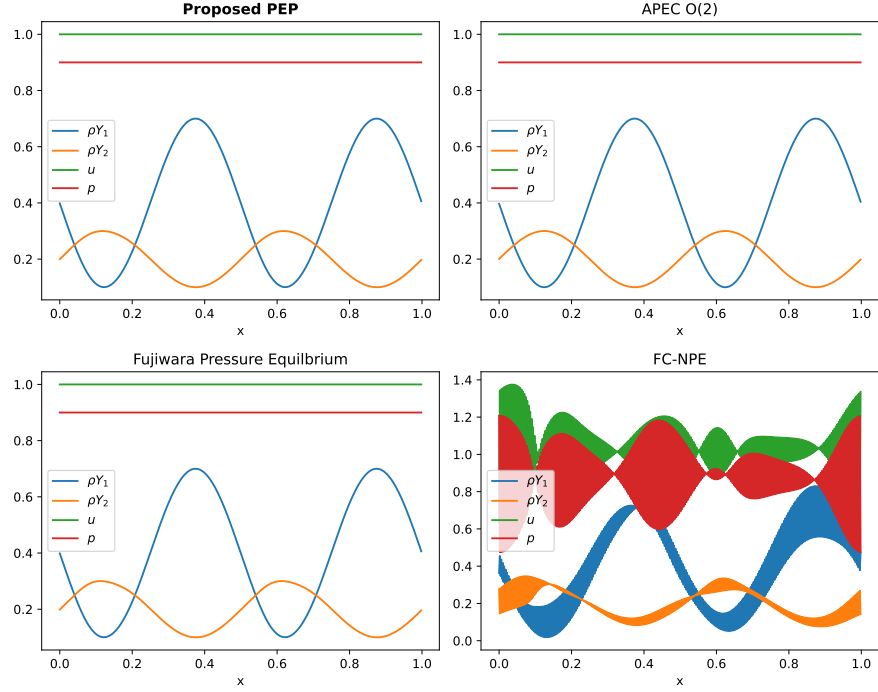


Figure 5.5: Second-order numerical solutions at $t = 7$ for the stiffened-gas test case defined in Equation 5.116. The four panels show: (top-left) the proposed PEP scheme; (top-right) the APEC scheme of [142]; (bottom-left) the PEP method of [55]; and (bottom-right) the second-order KEEP scheme (FC-NPE). In each panel, blue, orange, green, and red denote ρY_1 , ρY_2 , u , and p , respectively.

$$b_i = \frac{RT_{c,i}}{8 P_{c,i}}, \quad (5.127)$$

and the cross term follows

$$a_{12} = a_{21} = \sqrt{a_{11}a_{22}}. \quad (5.128)$$

We now rewrite the EOS in terms of the compressibility factor $Z \equiv pv/(RT)$.

Introducing the dimensionless parameters

$$a = \frac{\bar{a} P}{R^2 T^2}, \quad (5.129)$$

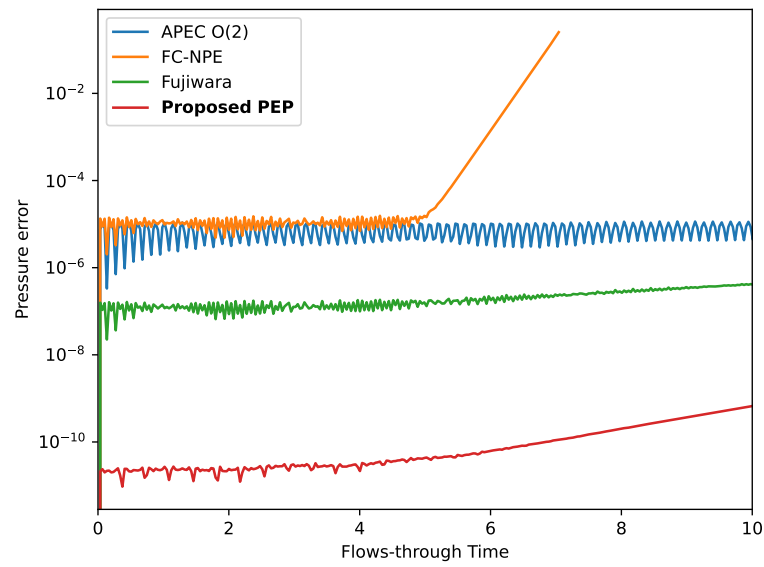


Figure 5.6: L_2 pressure error for the stiffened-gas test case. Curves correspond to the APEC $\mathcal{O}(2)$ scheme (blue), the FC-NPE scheme (orange), the Fujiwara PEP scheme (green), and the proposed PEP scheme (red).

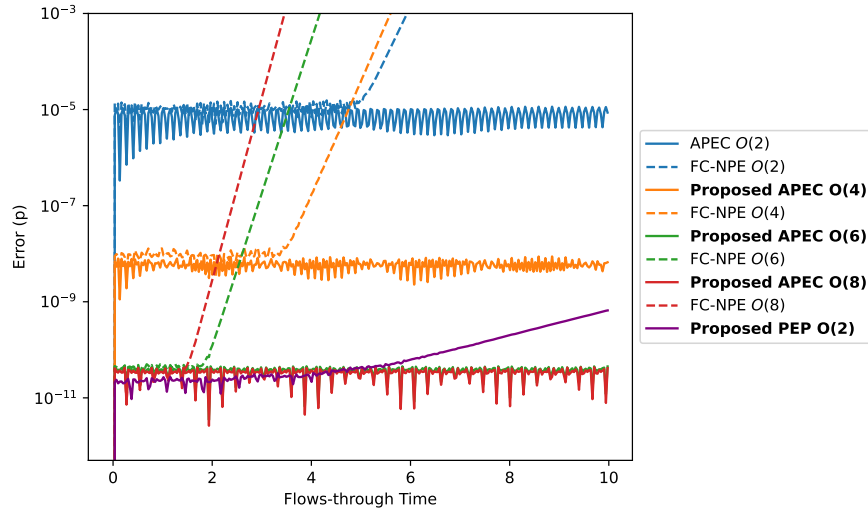


Figure 5.7: L_2 pressure error for the higher-order APEC and NPE schemes applied to the stiffened-gas test. APEC curves are plotted as solid lines and NPE curves as dashed lines, with colors denoting the formal order of accuracy: $\mathcal{O}(2)$ (blue), $\mathcal{O}(4)$ (orange), $\mathcal{O}(6)$ (green), and $\mathcal{O}(8)$ (red). The proposed PEP scheme is shown in purple for reference.

$$b = \frac{\bar{b} P}{RT}, \quad (5.130)$$

the relation becomes a cubic in Z ,

$$Z^3 - (1 + b) Z^2 + a Z - a b = 0. \quad (5.131)$$

At each grid point we solve this cubic for Z and select the physically admissible real root with $Z > b$ (ensuring $v - b > 0$). The specific volume then follows from

$$v = \frac{Z R T}{P}. \quad (5.132)$$

We now consider a two-fluid mixture representative of nitrogen and oxygen, governed by a van der Waals equation of state. The species parameters are

$$\gamma = [1.4, 1.4], \quad M = [28, 32] \text{ g/mol}, \quad T_c = [126.2, 154.5] \text{ K}, \quad P_c = [3.39, 5.04] \text{ MPa}. \quad (5.133)$$

On a periodic domain $x \in [0, 1]$ with $N = 100$ cells, we initialize smooth composition and temperature variations at uniform pressure and velocity. The primitive initial conditions are

$$\begin{pmatrix} Y_{1,0} \\ Y_{2,0} \\ u_0 \\ T_0 \\ p_0 \end{pmatrix} = \begin{pmatrix} 0.4 \sin(2\pi x) + 0.5 \\ 1 - Y_{1,0} \\ 100 \\ 600 - 300 \sin(2\pi x) \\ 60 \times 10^5 \end{pmatrix}, \quad (5.134)$$

where p_0 is in Pa and T_0 in K. At each grid point we compute the mole fractions X_i , assemble the mixture van der Waals parameters $\bar{a}$ and $\bar{b}$, form a and b , solve the

cubic for Z with the $Z > b$ selection, recover v , and then construct the conservative variables using the van der Waals relations.

In this configuration the flow is again initialized in exact pressure and velocity equilibrium, but with strong coupled variations in composition and temperature. The van der Waals EOS makes the thermodynamic response particularly sensitive to these variations, so this test provides a test case on how each scheme balances pressure equilibrium against errors in composition and temperature.

Figure 5.11 shows normalized profiles for the second order schemes at $t = 0.2$ (corresponding to 20 flow-through times). The top-left panel shows the initial condition, while the top-right panel displays the proposed PEP solution, the bottom-left panel the APEC $\mathcal{O}(2)$ scheme, and the bottom-right panel the second-order NPE (KEEP) baseline. All methods exhibit some distortion of the advected composition profiles after such long integration times, with noticeable amplitude loss and slight phase shifts relative to the initial data. A distinctive feature of the proposed PEP solution is the appearance of new peaks in the ρY_2 profile near $x \approx 0.8$. This arises because the method enforces pressure equilibrium by correcting both ρY_1 , ρY_2 , and the energy simultaneously, thereby spreading the thermodynamic correction across multiple fields rather than concentrating it in a single variable. In contrast, APEC and NPE both show smoother ρY_2 behavior near this location but at the price of a larger pressure and temperature imbalance as shown in Figure 5.12. This shows the pressure (left) and temperature (right) profiles at $t = 0.2$. In the pressure panel, the proposed PEP solution is visually indistinguishable from the initial condition which indicates the scheme is working as designed to maintain pressure equilibrium. Both APEC $\mathcal{O}(2)$ and the NPE

baseline drift away from the initial uniform pressure: NPE develops a more oscillatory profile but interestingly APEC deviates further away, but with a smoother profile. In the temperature panel, all schemes incur some error due however, the proposed PEP scheme again exhibits the smallest departure from the initial temperature profile (at the cost of the deviations in composition). APEC uses the temperature to correct pressure error and therefore has the largest deviation in temperature.

The time evolution of the pressure–equilibrium error is quantified in Figure 5.13, which plots the relative L_2 pressure error $\varepsilon_p(t)$ defined in Equation 5.117 for the proposed PEP scheme, APEC $\mathcal{O}(2)$, and the NPE baseline. The proposed PEP method maintains ε_p between 10^{-11} and 10^{-10} throughout the integration, essentially at the level of numerical precision for this problem. In contrast, both APEC and NPE are around $\mathcal{O}(10^{-5})$ with NPE actually having a lower error for later times. This is confirmed by Figure 5.12 where we see APEC deviate further in pressure but smoother. Before any large blowup in pressure occurs, all three curves accumulate dispersion error which causes the simulation to fail when ρY_1 dips below 0 at $x \approx 0.9$. This trend can be seen coming in Figure 5.11 as all 3 profiles have a small dip in ρY_1 approaching zero. The proposed PEP scheme shares this downfall with APEC and NPE as central schemes incur some dispersion error that leads to a not bound preserving method. Handling this long–time dispersive behavior with an appropriate regularization, such as a tailored viscosity or filter, is an important direction for future work, but we emphasize the reason the scheme fails is not due to pressure error anymore as the Proposed PEP maintains a pressure error of $\approx 10^{-10}$ for the full 20 periods.

Finally, Figure 5.14 shows the higher–order APEC and NPE (KEEP) variants,

together with the proposed PEP scheme, for the van der Waals test. As in the ideal-gas and stiffened-gas cases, APEC curves are plotted as solid lines and NPE curves as dashed lines, with colors denoting the formal order of accuracy ($\mathcal{O}(2)$, $\mathcal{O}(4)$, $\mathcal{O}(6)$, $\mathcal{O}(8)$) as before. We again focus on the relative L_2 pressure error $\varepsilon_p(t)$. For this more challenging EOS with large temperature variations, the wider APEC stencils yield a notable benefit: the thermodynamic corrections induced by the approximate pressure-equilibrium constraint are spread over a broader number of points. As a result, the APEC $\mathcal{O}(6)$ and $\mathcal{O}(8)$ schemes remain extremely stable over the time window shown ($t \leq 20$, i.e. roughly 2000 flow-through times), with pressure errors near machine precision and smooth profiles in both composition and temperature. The NPE $\mathcal{O}(4)$ scheme, by contrast, develops significantly larger pressure error and is visibly “blown up” on the same scale.

5.7 Taylor Results

Figures 5.8–5.10 show that the Taylor-regularized family can be used with the raw closed-form α and β expressions without any switching or pseudoinverse regularization across all three equations of state (ideal-gas, stiffened-gas, and van der Waals mixture). In every case, the Taylor expansion removes the numerical instability that otherwise appears in the closed forms. Even at low order, the Taylor- (n) schemes achieve stability and accuracy comparable to the APEC $\mathcal{O}(2)$ method, and they remain stable far longer than the NPE scheme. As the Taylor order increases, the pressure error decreases and the solutions begin to approach the accuracy of the proposed PEP

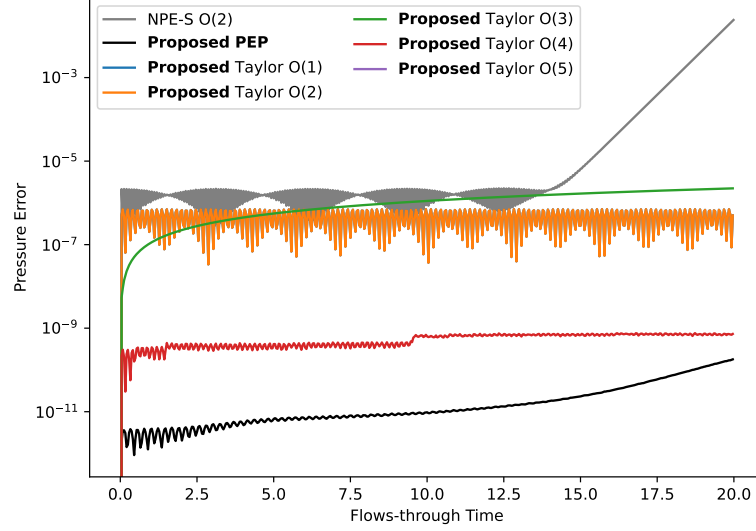


Figure 5.8: Relative L_2 pressure error for the ideal-gas test case using the Taylor-regularized family Taylor- (n) with $n = 1, \dots, 5$. Curves show the NPE scheme (gray), the proposed PEP scheme (black), and Taylor-regularized variants from first through fifth order: $n = 1$ (blue), $n = 2$ (orange), $n = 3$ (green), $n = 4$ (red), and $n = 5$ (purple). All runs employ the same second-order base discretization, with n indicating the highest power of $\Delta(\rho Y_i)$ retained in the Taylor surrogate.

scheme, but they still do not fully match its accuracy. Notably, the higher-order Taylor variants remain extremely stable and do not exhibit the late-time growth seen in the PEP curve, which arises from the accumulation of roundoff-level noise in the switching mechanism. However, this increased robustness comes at significant computational cost of evaluating all mixed derivatives that appear in the higher-order Taylor expansions. This makes this scheme increasingly more expensive with higher Taylor expansions and is considerably more expensive than the lightweight coefficient evaluation used in PEP.

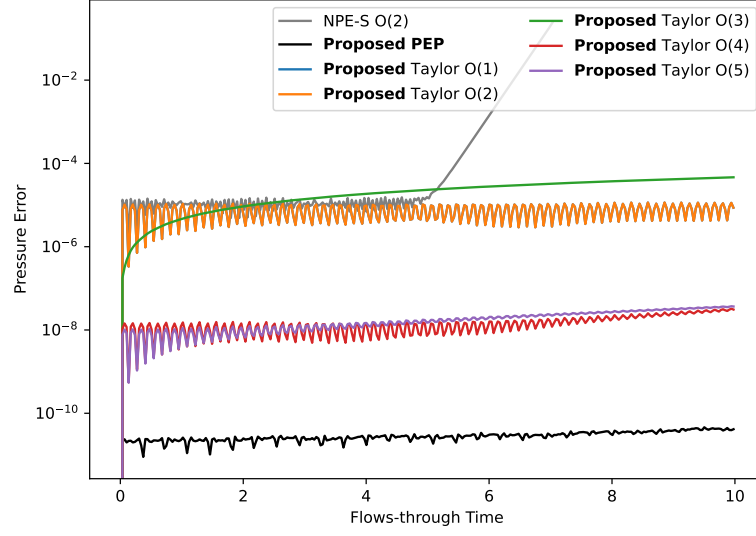


Figure 5.9: Relative L_2 pressure error for the stiffened-gas mixture test case using the Taylor-regularized family Taylor- (n) with $n = 1, \dots, 5$. Curves show the NPE scheme (gray), the proposed PEP scheme (black), and Taylor-regularized variants from first through fifth order: $n = 1$ (blue), $n = 2$ (orange), $n = 3$ (green), $n = 4$ (red), and $n = 5$ (purple). All runs employ the same second-order base discretization, with n indicating the highest power of $\Delta(\rho Y_i)$ retained in the Taylor surrogate.

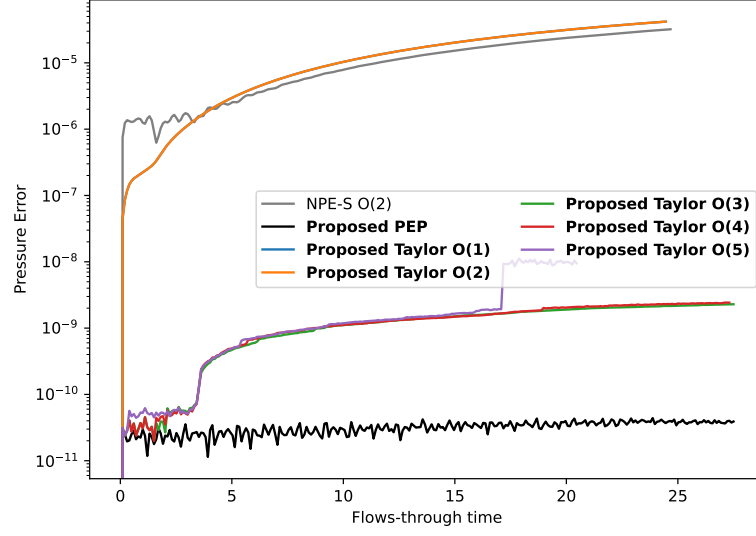


Figure 5.10: Relative L_2 pressure error for the van der Waals mixture test case using the Taylor-regularized family Taylor- (n) with $n = 1, \dots, 5$. Curves show the NPE scheme (gray), the proposed PEP scheme (black), and Taylor-regularized variants from first through fifth order: $n = 1$ (blue), $n = 2$ (orange), $n = 3$ (green), $n = 4$ (red), and $n = 5$ (purple). All runs employ the same second-order base discretization, with n indicating the highest power of $\Delta(\rho Y_i)$ retained in the Taylor surrogate.

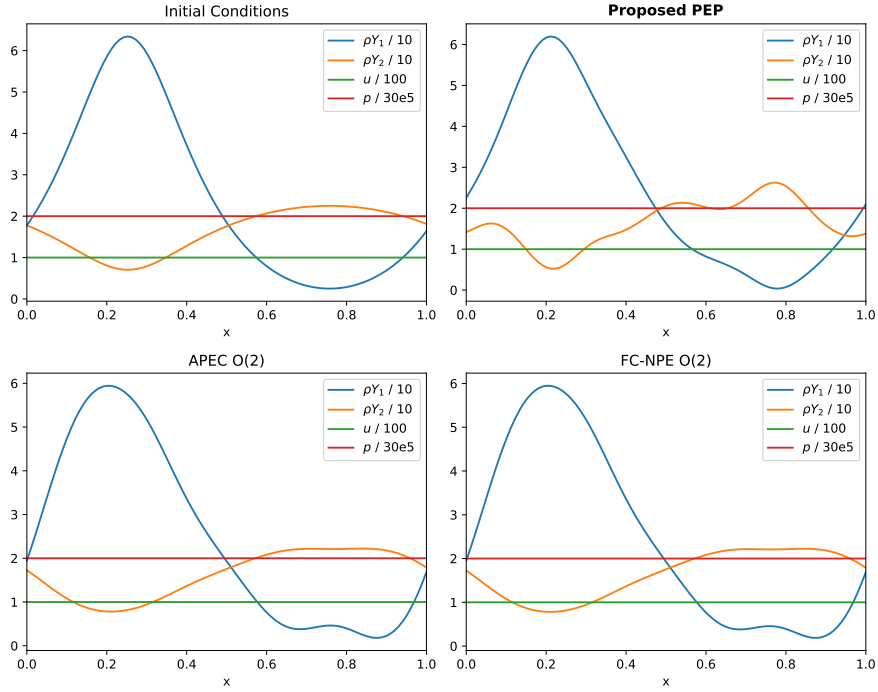


Figure 5.11: Normalized profiles at $t = 0.2$ for the van der Waals mixture test. The four panels show: (top-left) the initial condition; (top-right) the proposed PEP scheme; (bottom-left) the APEC $\mathcal{O}(2)$ scheme; and (bottom-right) the second-order NPE (KEEP) baseline. In each panel, ρY_1 , ρY_2 , u , and p are scaled as indicated in the legend to fit on common axes and are colored blue, orange, green, and red respectively.

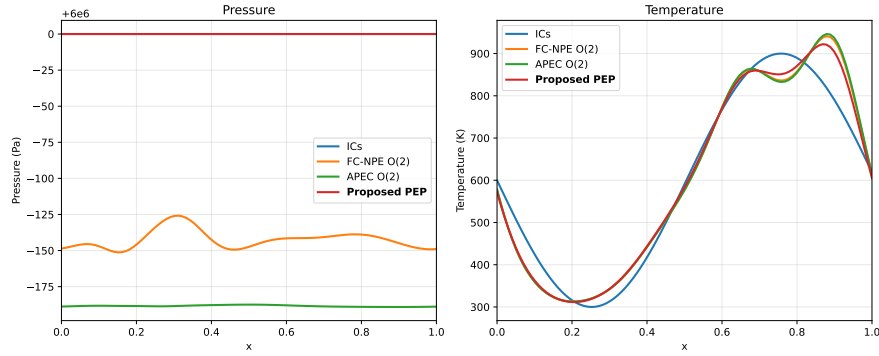


Figure 5.12: Pressure (left) and temperature (right) profiles at $t = 0.2$ for the van der Waals test case, comparing the initial condition, the proposed PEP scheme, APEC $\mathcal{O}(2)$, and the second-order NPE (KEEP) baseline. Colors: blue – initial condition, orange – NPE, green – APEC $\mathcal{O}(2)$, red – proposed PEP.

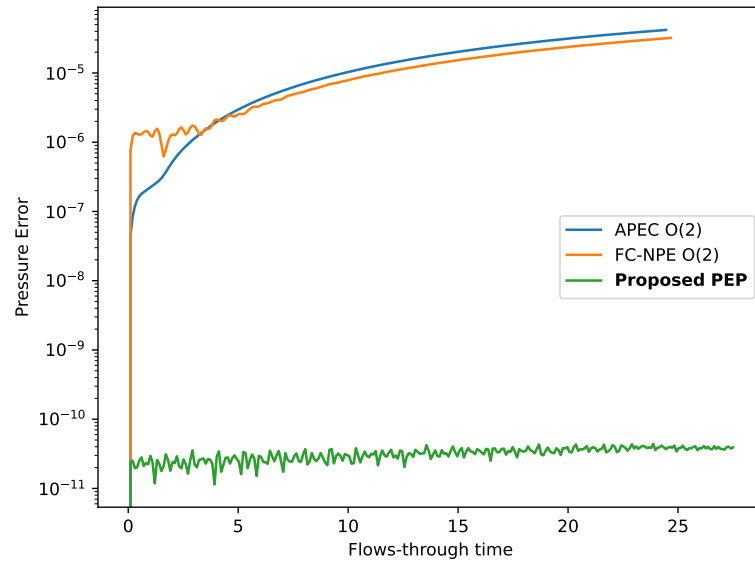


Figure 5.13: Relative L_2 pressure error $\varepsilon_p(t)$ for the van der Waals test, comparing the proposed PEP scheme, APEC $\mathcal{O}(2)$, and the second-order NPE (KEEP) baseline. Colors: black – proposed PEP, blue – APEC $\mathcal{O}(2)$, orange – FC-NPE baseline.

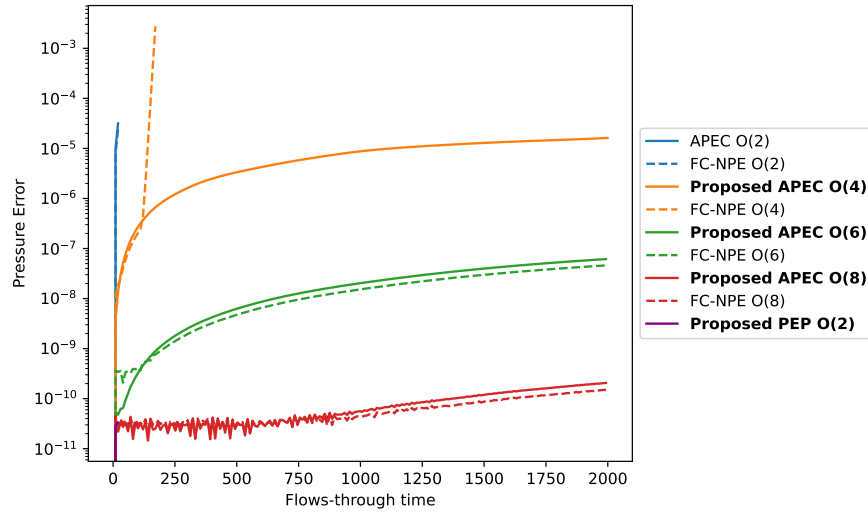


Figure 5.14: Relative L_2 pressure error for higher-order APEC and NPE (KEEP) schemes, together with the proposed PEP method, applied to the van der Waals mixture test. APEC curves are shown as solid lines and NPE curves as dashed lines. Colors indicate formal order and scheme: blue – $\mathcal{O}(2)$, orange – $\mathcal{O}(4)$, green – $\mathcal{O}(6)$, red – $\mathcal{O}(8)$, purple – proposed PEP (formally second order).

5.8 Sensitivities to Reciprocal Condition Number

In Section 5.4.3 we introduced a global reciprocal condition number r_g defined by Eq. (5.88), obtained by scanning the local face-wise condition estimates $r|_{m+\frac{1}{2}}$ at the initial condition. For the three EOS configurations considered in this work, the resulting values are

$$\text{Ideal gas: } r_g \approx 1.16 \times 10^{-5},$$

$$\text{Stiffened gas: } r_g \approx 3.20 \times 10^{-6},$$

$$\text{van der Waals: } r_g \approx 2.46 \times 10^{-9}.$$

These values reflect the different conditioning properties of the PEP system across equations of state for the fluid systems and ICs presented. It's important to note that these values were for the specific ICs and fluid systems and won't be optimal generally for an arbitrary fluid system or IC with the same EOS.

5.8.1 Local behavior of α and β

To build intuition for the role of the reciprocal condition number, we first examine its effect on the local PEP correction coefficients α and β . We consider the two-species ideal-gas mixture used in the main text and set up a simple interface experiment. Starting from uniform pressure and velocity, we prescribe a species jump

$$\rho Y_{1,m+1} = \rho Y_{1,m} + \Delta(\rho Y_i), \quad \rho Y_{2,m+1} = \rho Y_{2,m} + \Delta(\rho Y_i), \quad (5.135)$$

and, for each value of $\Delta(\rho Y_i)$, compare the closed-form analytic solution (5.80)–(5.81), the Moore–Penrose pseudoinverse with several fixed reciprocal condition number values,

which we denote, `rcond`, and the automatic choice based on the r_g construction in Eq. (5.88). In all cases, the second-order APEC scheme corresponds to $\alpha = \beta = 0$.

Figure 5.15 shows the resulting α (left panel) and β (right panel) as functions of $\Delta(\rho Y_i)$. The analytic closed-form solution exhibits the strongest oscillations: as $\Delta(\rho Y_i) \rightarrow 0$, the underlying linear system becomes nearly singular, and roundoff errors are amplified into large, oscillatory variations in α and β . Very small tolerances (e.g. `rcond` = 10^{-14}) closely track this analytic curve and therefore inherit its oscillations. On the other hand, very large tolerances (e.g. `rcond` = 10^{-4}) clamp the correction rapidly toward zero causing the coefficients α and β are forced to vanish even for moderately sized composition jumps, so the scheme behaves like APEC over a wide range of $\Delta(\rho Y_i)$.

Between these extremes there is clearly an optimal range of `rcond` values. As $\Delta(\rho Y_i)$ decreases toward zero, the ideal behavior is for the PEP corrections to decay smoothly to $\alpha = \beta = 0$, so that the scheme transitions gradually from the fully conservative PEP correction to the APEC fallback without numerical oscillations. In Figure 5.15, intermediate tolerances such as `rcond` = 10^{-6} achieve this behavior. Larger tolerances (e.g. `rcond` = 10^{-4}) cut off this decay too sharply and effectively switch to APEC too early, while smaller tolerances oscillate in the near-singular regime. For the ideal-gas test, the best-performing fixed choice `rcond` = 10^{-6} which is the closest to the global, conditioning-based estimate, $r_g \approx 1.16 \times 10^{-5}$.

5.8.2 Global pressure-error behavior across EOS

The local picture in Section 5.8.1 manifests at the global level as a balance between accuracy and robustness in the long-time pressure error. To quantify this effect,

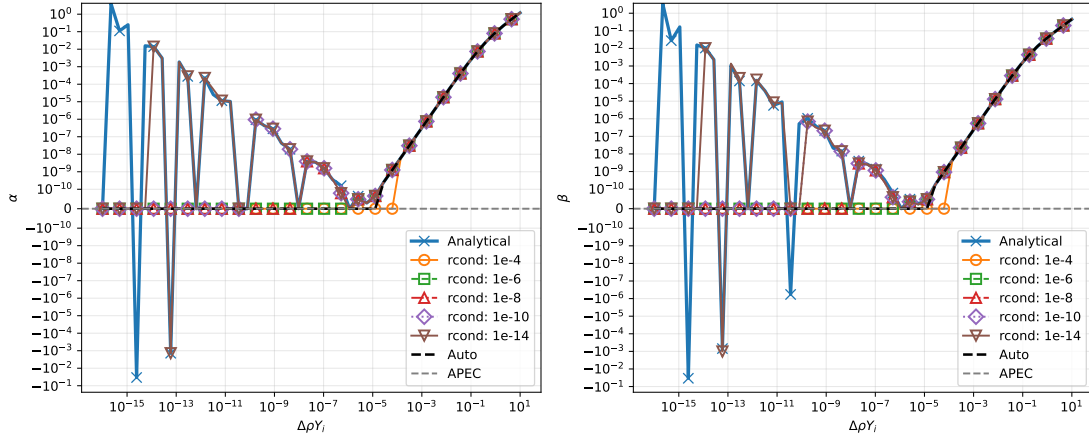


Figure 5.15: Local sensitivity of the PEP correction coefficients α (left) and β (right) to the reciprocal condition number in an ideal-gas two-species mixture. Each curve shows α or β as a function of the species jump $\Delta(\rho Y_i)$. The analytic closed-form solution (no regularization) displays strong oscillations as $\Delta(\rho Y_i) \rightarrow 0$, while the APEC baseline (grey) corresponds to $\alpha = \beta = 0$. The pseudoinverse results with fixed `rcond` values interpolate between these limits, and the automatic r_g -based choice yields a smooth decay of α and β to zero near the determinant-limiting regime.

we repeat the ideal-gas, stiffened-gas, and van-der-Waals numerical experiments using the fully conservative PEP scheme, but vary the reciprocal condition number used in the Moore–Penrose pseudoinverse. In addition to the automatic r_g choice from Eq. (5.88), we consider several fixed tolerances (here `rcond` = 10^{-14} , 10^{-10} , 10^{-8} , 10^{-6} , and 10^{-4}) to sample both the under-regularized and over-regularized regimes. The resulting ℓ_2 pressure errors as functions of flows-through time are shown in Figure 5.16. Each panel includes the second-order APEC baseline and the fully conservative NPE (FC-NPE) reference, together with the proposed PEP scheme for the various `rcond` values and the

automatic r_g choice as presented in the main body.

The trends mirror the local behavior of α and β . When `rcond` is too small, the pseudoinverse admits nearly all singular values, including those associated with ill-conditioned faces. The resulting oscillatory corrections in α and β produce high-frequency noise in the fluxes and a rapid growth of the global pressure error. When `rcond` is too large, the solver discards many small singular values, effectively switching to the APEC flux over a large portion of the domain; the solution remains robust, but the pressure error grows more quickly and approaches the APEC baseline.

In between, there is an intermediate range in which the PEP correction is active where it is most needed (near strong composition jumps) and smoothly deactivated in the ill-conditioned regime, yielding both low pressure error and good long-time stability. Across all three EOS, the automatically selected r_g lies in this intermediate regime and closely tracks the best fixed-`rcond` choices, consistent with the local behavior in Figure 5.15. This suggests that the conditioning-based construction of r_g provides a simple, problem-aware default that does not require manual tuning, while still delivering a smooth and physically sensible transition between the fully conservative PEP update and its APEC fallback.

5.9 Extension of the Proposed PEP Scheme to Single Species

Although the proposed PEP formulation is developed for multi-component flows with general equations of state, we present a simplified variant for single-species flow. In this case, there is only one conserved species density, which we denote by $\rho Y =$

ρ . The pressure-equilibrium criterion shown in Equation 5.30 reduces to corrections on ρ and ρe only. We can therefore define coefficient correction terms α' and β' as

$$\rho|_{m+\frac{1}{2}}^{\text{PEP}} = \frac{\rho|_m + \rho|_{m+1}}{2} + \alpha', \quad (5.136)$$

$$\rho e|_{m+\frac{1}{2}}^{\text{PEP}} = \frac{\rho e|_m + \rho e|_{m+1}}{2} + \beta' - \left[\frac{\epsilon_{m+1}}{2}(\rho_{m+1} - \rho_{m+\frac{1}{2}}) - \frac{\epsilon_m}{2}(\rho_{m+\frac{1}{2}} - \rho_m) \right] \quad (5.137)$$

where we leave the APEC correction in Equation 5.137 as before so as α', β' approach zero the scheme exactly recovers the single-species APEC scheme. The extension to species > 2 is discussed in Section 5.4.4.

5.10 Conclusion

This work, presented in [45], introduced two complementary strategies for suppressing spurious pressure oscillations in multi-component flows with general equations of state: a high-order extension of the approximate pressure-equilibrium-preserving (APEC) correction, and a fully conservative scheme that enforces the discrete PEP condition exactly. Although APEC remains approximate and cannot preserve pressure equilibrium exactly, its high-order extension substantially improves accuracy, and in practice its pressure errors often approach those of the fully conservative framework while retaining excellent robustness. The fully conservative PEP scheme, by contrast, satisfies the PEP criterion exactly and preserves all conserved quantities, but its long-time behavior is more sensitive to numerical noise introduced by the pseudo-inverse, leading to a characteristic pressure-error growth distinct from that of previous methods. We also present the Taylor based methods which for an increased computational

cost, can increase the long term stability as well as decrease pressure error to a significant degree while not introducing any numerical instability.

These findings highlight several promising directions for future work. Incorporating a targeted pressure-equilibrium artificial viscosity [143] may help regularize the conservative update and suppress long-time noise amplification introduced by the proposed PEP scheme but could also help with the dispersion error. Likewise, hybrid schemes that blend high-order APEC with a regularized PEP correction could improve stability without sacrificing accuracy. One could pursue a genuine high-order extension of the fully conservative PEP method, in which the exact PEP criterion is enforced across all sub-stencil intersections, mirroring the construction of the high-order APEC scheme. However, because the present conservative scheme already enforces the exact criterion, its limitations now arise primarily from numerical noise and dispersion rather than from a lack of formal order, suggesting that any high-order conservative PEP method must directly address these sources of instability.

Finally, the broader impact of this framework lies in its generality. Both the high-order APEC correction and the proposed conservative PEP update act as modular correction steps and can be incorporated into any numerical flux function. While this work employed KEEP as a specific starting point, the methodology is not restricted to it. Extensions to solvers relying on traditional Riemann fluxes are straightforward and represent a natural next step.

In summary, high-order APEC provides a practical and effective tool for mitigating spurious pressure oscillations for arbitrary EOS and multi-species systems, while the fully conservative PEP formulation establishes a flexible and extensible foundation

for future development of accurate and fully conservative multi-component flow solvers.

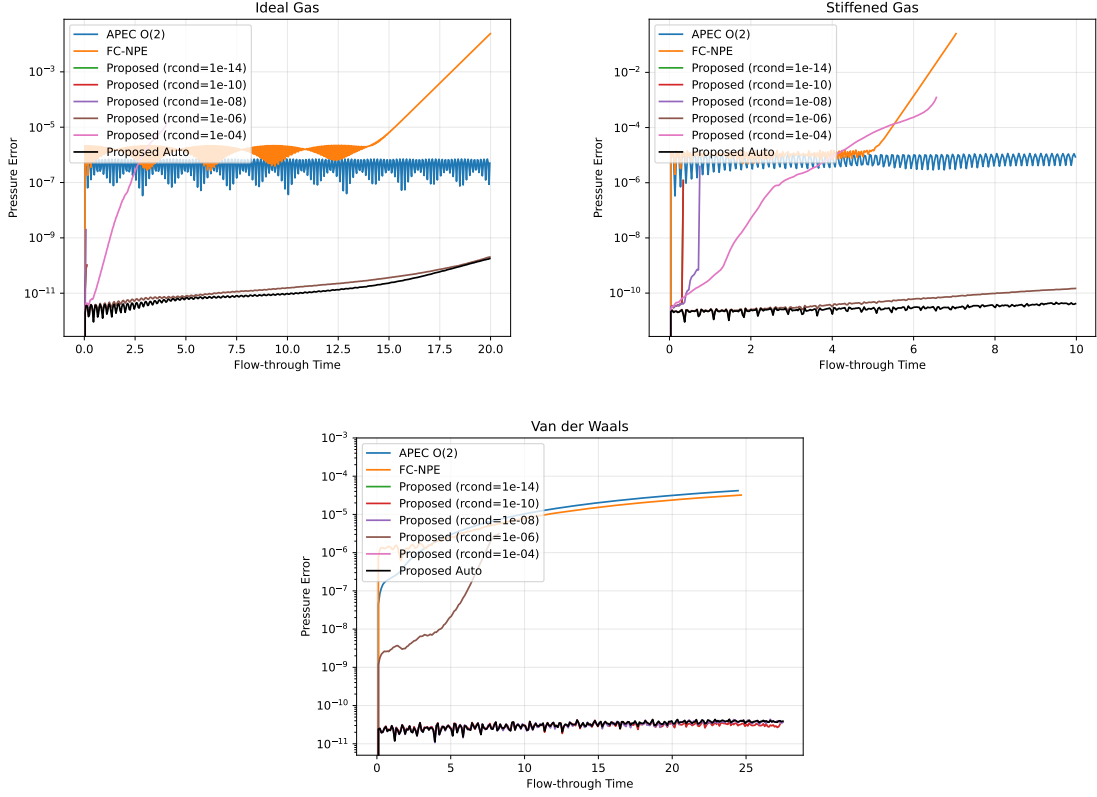


Figure 5.16: Sensitivity of long-time pressure error to the reciprocal condition number used in the Moore–Penrose pseudoinverse for the three EOS configurations (ideal-gas, stiffened-gas, van der Waals). Each panel shows the ℓ_1 pressure error versus flow-through time. The second-order APEC scheme and the FC-NPE baseline are plotted as references, along with the proposed fully conservative PEP scheme for several fixed `rcond` values and the automatic choice r_g from Eq. (5.88).

Chapter 6

Coupling of Machine Learning Based Diffusion

6.1 Introduction

Partial differential equations containing both hyperbolic and parabolic terms are fundamental to modeling physical phenomena across science and engineering, from fluid dynamics and heat transfer to electromagnetic wave propagation and quantum mechanics. The numerical solution of these mixed hyperbolic-parabolic systems remains computationally challenging due to the severe timestep restrictions imposed by stability requirements. While hyperbolic terms typically require timesteps of order $\Delta t \sim \mathcal{O}(\Delta x)$ according to the Courant-Friedrichs-Lewy (CFL) condition [41], parabolic diffusion terms impose a significantly more restrictive constraint of $\Delta t \sim \mathcal{O}(\Delta x^2)$ [88, 145]. This quadratic dependence on spatial resolution becomes the dominant factor limiting computational efficiency, particularly as spatial resolution increases to capture fine-scale

physics in computational fluid dynamics applications [7, 18].

Traditional approaches to circumvent the parabolic timestep restriction have focused primarily on implicit and implicit-explicit (IMEX) time integration schemes. Implicit methods eliminate the timestep restriction but introduce substantial computational overhead through the need to solve large linear systems at each timestep [10, 60]. IMEX schemes, which treat diffusive terms implicitly and advective terms explicitly, have shown promise in balancing stability and computational cost [11, 73, 109]. These methods have been successfully applied to various applications ranging from atmospheric modeling to reaction-diffusion systems [57, 19]. However, IMEX schemes still require solving implicit systems for the parabolic terms and can suffer from order reduction and stability issues when the implicit and explicit components interact strongly [54, 150]. Recent developments in IMEX Runge-Kutta methods have improved stability regions and accuracy [39, 67], yet the fundamental computational burden of implicit solves remains.

The emergence of machine learning techniques for scientific computing has opened new avenues for accelerating PDE solvers. Physics-informed neural networks (PINNs) directly incorporate PDE constraints into the loss function, enabling mesh-free solution approximation [121, 72]. However, PINNs typically require retraining for different initial or boundary conditions and can struggle with complex multi-scale dynamics due to spectral bias—the tendency of neural networks to preferentially learn low-frequency components [118, 151]. Neural operators, which learn mappings between function spaces rather than point-wise solutions, offer a more flexible framework [97, 77]. The Fourier Neural Operator (FNO) has demonstrated remarkable success by

parameterizing integral kernels directly in Fourier space, achieving speedups of several orders of magnitude compared to traditional solvers [90]. Extensions such as U-Net-based neural operators [119] and geometry-aware variants [89] have further expanded applicability to complex domains.

Recent work has increasingly focused on hybrid approaches that combine the strengths of traditional numerical methods with machine learning acceleration. Neural-network closures for subgrid physics in large-eddy simulations can improve both accuracy and efficiency [75], and ML-accelerated weather models show that substantial speedups are possible while maintaining physical consistency [110, 29]. At the same time, studies of spectral bias highlight limitations in capturing high-frequency dynamics [30, 105], which is directly relevant when designing hybrids that leverage learned components without sacrificing numerical stability.

The PDE-Refiner framework addresses error accumulation in autoregressive neural PDE solvers through iterative refinement [91], while message-passing neural networks have shown promise in preserving local conservation properties [26]. Work on neural stencils [13] and learned finite volume methods [93] demonstrates the potential for ML-enhanced discretization schemes. Several studies have explored embedding discretized PDEs directly into network architectures, drawing connections between ResNets and numerical schemes [32, 133, 31]. Graph neural networks inspired by diffusion processes have shown improved stability for long-time integration [28]. However, most existing hybrid approaches either replace entire solution components or learn corrections to existing schemes, rather than strategically targeting specific computational bottlenecks.

A critical challenge in applying neural networks to PDE solving is the spectral bias phenomenon, where networks preferentially learn low-frequency components before high-frequency ones [118, 157]. This bias is particularly problematic for turbulent flows and multi-scale systems where high-frequency components are essential for physical accuracy. Recent work has proposed various strategies to overcome spectral bias, including Fourier feature encodings [141, 151], multi-scale architectures [95], and adaptive activation functions [68]. Understanding and mitigating spectral bias is crucial for developing stable hybrid numerical-ML schemes that can accurately capture the full spectrum of physical scales.

In this work, we propose a novel hybrid approach that specifically targets the parabolic timestep restriction by replacing only the parabolic flux divergence calculation with neural network predictions, while retaining traditional numerical methods for hyperbolic terms. This selective replacement strategy is motivated by several key observations. First, parabolic terms typically represent smooth, diffusive processes that are well-suited to neural network approximation, unlike hyperbolic terms which can develop sharp discontinuities and shocks. Second, by eliminating the $\mathcal{O}(\Delta x^2)$ timestep restriction from parabolic terms, simulations can proceed at the less restrictive hyperbolic CFL limit, potentially achieving order-of-magnitude speedups. Third, maintaining traditional shock-capturing schemes for hyperbolic terms preserves important conservation properties and ensures robust handling of discontinuities.

We investigate multiple aspects of this approach: comparing different neural network architectures for their effectiveness in predicting parabolic terms; exploring loss function designs including spectral losses that preserve frequency content and multi-

step consistency losses that improve temporal stability; investigating training strategies including unrolled training and noise injection following insights from previous work [134, 148, 76]; and examining different strategies for coupling neural predictions with numerical schemes.

We provide comprehensive stability analysis of the coupled ML-numerical schemes through eigenvalue analysis and long-term integration tests. Our experiments span three canonical PDE systems of increasing complexity: the Burgers equation with diffusion, the 1D Euler equations with heat conduction, and the 1D viscous Euler equations with both heat conduction and viscosity. Through extensive numerical experiments, we demonstrate that our approach can achieve stable, accurate simulations with significantly larger timesteps than traditional methods, while maintaining physical fidelity and conservation properties. The results show that carefully designed hybrid schemes can overcome fundamental numerical limitations while preserving the robustness and interpretability that make traditional methods reliable for critical applications.

6.2 Methodology

We propose a hybrid numerical-machine learning framework for mixed hyperbolic-parabolic PDEs that preserves traditional discretizations for hyperbolic fluxes while replacing the parabolic contribution with a neural-network (NN) prediction of the diffusive divergence. This section first states the mathematical setting and coupling, then introduces two operating frameworks: **E-IVP** (Explicit-IVP) for parabolic-timestep continuation, and **I-DS** (Implicit-Dataset) for learned implicit increments evaluated

across diverse initial conditions and diffusivities κ .

6.2.1 Mathematical formulation and coupling

We consider the following form of a hyperbolic–parabolic conservation law

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \nabla \cdot (\mathbf{D}(\mathbf{U}, \nabla \mathbf{U})), \quad (6.1)$$

where $\mathbf{U}$ is the state of conserved variables, $\mathbf{F}$ the inviscid (hyperbolic) flux, and $\mathbf{D}$ the viscous (parabolic) flux, possibly nonlinear in $\mathbf{U}$ and its gradients. In a cell-centered finite-volume scheme the semi-discrete update is

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x} \left(\tilde{F}_{i+\frac{1}{2}}^{\text{hyp}} - \tilde{F}_{i-\frac{1}{2}}^{\text{hyp}} \right) + D_i^{\text{NN}}, \quad (6.2)$$

with numerical hyperbolic fluxes $\tilde{F}^{\text{hyp}}$ and an NN prediction D_i^{NN} for the parabolic divergence at cell centers.

Explicit diffusion imposes the parabolic stability bound

$$\Delta t \propto \frac{\Delta x^2}{\nu}, \quad (6.3)$$

while implicit treatments relax this restriction at the cost of solving a global linear system each step (tridiagonal in 1D constant-coefficient settings; sparse iterative in multi-D).

6.2.2 Framework I: E–IVP (Explicit–IVP)

We consider two training frameworks, in the first, **E–IVP**, we advance at the parabolic timestep and train the NN to emulate the explicit diffusive divergence which we define as

$$\mathcal{D}(\mathbf{U}) \equiv \nabla \cdot (\mathbf{D}(\mathbf{U}, \nabla \mathbf{U})).$$

We now define the model

$$\widehat{\mathcal{D}}_\theta(\mathbf{U}) \approx \mathcal{D}(\mathbf{U})$$

where $\widehat{\mathcal{D}}_\theta$ is the neural surrogate with parameters θ , having the same input–output layout as $\mathcal{D}$

We integrate from t_0 to t_{final} and continue beyond t_{final} to assess long-horizon stability and accuracy. This IVP continuation isolates the parabolic contribution and measures how well D^{NN} reproduces classical diffusion dynamics over sustained evolution. It is also a much simpler problem which will probe stability issues.

6.2.3 Framework II: I–DS (Implicit–Dataset)

In the **I–DS** setting, we adopt an IMEX view and learn the implicit one-step diffusive effect directly, then evaluate it across diverse initial conditions and diffusivities κ .

Let $G(\mathbf{U}^n)$ denote the explicit right-hand-side built from the hyperbolic discretization, written as

$$G(\mathbf{U}^n) := -\frac{1}{\Delta x} \left(\tilde{F}_{i+\frac{1}{2}}^{\text{hyp}}(\mathbf{U}^n) - \tilde{F}_{i-\frac{1}{2}}^{\text{hyp}}(\mathbf{U}^n) \right).$$

Let L_{diff} denote the discrete diffusion operator. A single first-order IMEX step can be written as the linear solve

$$(I - \Delta t \kappa L_{\text{diff}}) \mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t G(\mathbf{U}^n). \quad (6.4)$$

Define the implicit update (increment) by

$$\mathcal{D}^{\text{imp}}(\mathbf{U}^n) := \mathbf{U}^{n+1} - \mathbf{U}^n = (I - \Delta t \kappa L_{\text{diff}})^{-1}(\mathbf{U}^n + \Delta t G(\mathbf{U}^n)) - \mathbf{U}^n. \quad (6.5)$$

In **I-DS**, we introduce its neural surrogate, with the same input–output layout,

$$\widehat{\mathcal{D}}_{\theta}^{\text{imp}}(\mathbf{U}^n) \approx \mathcal{D}^{\text{imp}}(\mathbf{U}^n).$$

This approximates the net one-step effect of the implicit diffusion solve, without forming or inverting the operator in (6.4) at runtime. Using the same explicit hyperbolic discretization as in (6.2), the NN-coupled IMEX step is written

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t G(\mathbf{U}^n) + \widehat{\mathcal{D}}_{\theta}^{\text{imp}}(\mathbf{U}^n). \quad (6.6)$$

We evaluate this learned implicit update across varied ICs and κ to probe robustness to state and parameter variation.

6.2.3.1 Boundary Condition Handling

We use ghost cells to enforce boundary conditions consistently for both the numerical solver and the neural network. Before each timestep, ghost layers are populated according to the specified boundary types. The padded field, including ghosts, is then provided to the network so that its receptive field near the physical boundary is well-posed. The network returns predictions on the padded domain; only the interior-cell outputs (excluding ghosts) are admitted into (6.2). This procedure ensures that the NN sees boundary-consistent inputs and no spurious gradients arise from truncated stencils. This approach ensures that the neural network sees physically consistent boundary context while maintaining the same stencil width as the numerical scheme.

6.2.4 Training Methodology

We use three architectures: a fully connected DNN, a Fourier Neural Operator (FNO) with learned spectral filters, and a U-Net (UNO) encoder–decoder. All three models have roughly 0.2 million parameters. For Framework II training, we embed κ and Δt into the input to let the models adapt to different diffusion levels. All models are trained to 5000 epochs with the AdamW optimizer with a learning rate 3×10^{-4} , $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay 10^{-3} .

6.2.4.1 Loss Functions

(1) Mean Squared Error (MSE) in physical space. Directly matches the predicted parabolic term to the target on the full domain:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{(N + 2G)} \sum_{i=1}^{N+2G} \|D_{\text{ML},i} - D_{\text{ref},i}\|_2^2, \quad (6.7)$$

(2) Spectral loss (FFT magnitude matching). To preserve frequency content and avoid ghost–cell artifacts, we compute FFT magnitudes using only the interior cells, excluding ghost cells. Let $\mathcal{F}$ denote the 1D FFT along the spatial axis and keep K modes. The loss is

$$\mathcal{L}_{\text{spectral}} = \frac{1}{K} \sum_{k=0}^{K-1} (|(\mathcal{F}(D_{\text{ML}}))_k| - |(\mathcal{F}(D_{\text{ref}}))_k|)^2. \quad (6.8)$$

(3) Multi-step consistency (state-evolution). We unroll the hybrid solver for M substeps and compare the ML-evolved state to a reference numerical evolution (adaptive Δt from CFL, Δx , and wave/diffusive speeds). With discount $\gamma \in (0, 1]$,

$$\mathcal{L}_{\text{substep}} = \sum_{m=1}^M \gamma^m \frac{1}{N} \sum_{i=1}^N \|U_{\text{ML},i}^{(m)} - U_{\text{ref},i}^{(m)}\|_2^2, \quad (6.9)$$

(4) Spectral multi-step consistency. Same unroll as above, but compares the magnitudes of the interior FFTs of the evolved states. To avoid introducing spectral content from ghost cells, we compute all FFTs on the interior cells only.

$$\mathcal{L}_{\text{spec-substep}} = \sum_{m=1}^M \gamma^m \frac{1}{K} \sum_{k=0}^{K-1} \left(\left| (\mathcal{F}(U_{\text{ML}}^{(m)}))_k \right| - \left| (\mathcal{F}(U_{\text{ref}}^{(m)}))_k \right| \right)^2, \quad (6.10)$$

where the FFTs are taken on the interior slice only.

(5) TVD substep penalty (total-variation nonincrease). To discourage spurious amplification, we penalize any increase in total variation on the interior across substeps. For a state channel $u \in \mathbb{R}^N$, define

$$\text{TV}(u) = \sum_{i=1}^{N-1} |u_{i+1} - u_i|.$$

the penalty is

$$\mathcal{L}_{\text{tvd}} = \sum_{m=1}^M \gamma^m \frac{1}{C_{\text{state}}} \sum_{c=1}^{C_{\text{state}}} \max\left(\text{TV}(U_{\text{ML}}^{(m)}[c]) - \text{TV}(U_{\text{ML}}^{(m-1)}[c]), 0\right). \quad (6.11)$$

We combine the terms with nonnegative weights selected by hyperparameter search:

$$\begin{aligned} \mathcal{L}_{\text{total}} = & \mathcal{L}_{\text{MSE}} + \lambda_{\text{spectral}} \mathcal{L}_{\text{spectral}} + \lambda_{\text{substep}} \mathcal{L}_{\text{substep}} \\ & + \lambda_{\text{spec-substep}} \mathcal{L}_{\text{spec-substep}} + \lambda_{\text{tvd}} \mathcal{L}_{\text{tvd}}. \end{aligned} \quad (6.12)$$

We consider the following combinations of terms in Table 6.1 where many of the following plots are labeled $a - h$ corresponding to a model trained with $a - h$ loss function in the table.

Table 6.1: Loss-function configurations used in the comparison study.

Task	Name	$\lambda_{\text{spectral}}$	λ_{substep}	$\lambda_{\text{spec-substep}}$	λ_{tvd}	N_{sub}
a	MSE only	0	0	0	0	0
b	MSE + spectral	0.2	0	0	0	0
c	MSE + substep (1)	0	1	0	0	1
d	MSE + substep (5)	0	1	0	0	5
e	MSE + substep (10)	0	1	0	0	10
f	MSE + substep (10) + spec-substep	0	1	0.2	0	10
g	MSE + substep (10) + TVD-substep	0	1	0	1	10
h	All terms on	0.2	1	0.2	0.1	10

6.2.5 Stability analysis

We assess stability of the coupled numerical–ML scheme by viewing a single time step as a nonlinear map

$$\mathbf{u}^{n+1} = \mathbf{F}(\mathbf{u}^n; \boldsymbol{\theta}, \kappa, \Delta t), \quad (6.13)$$

where $\mathbf{u}^n \in \mathbb{R}^N$ denotes the grid state at time step n , $\boldsymbol{\theta}$ are the learned weights, and $(\kappa, \Delta t)$ are fixed physical and temporal parameters. Following standard linear stability analysis, we consider the Jacobian of the one-step map with respect to the state,

$$\mathbf{J}(\mathbf{u}^*) = \left. \frac{\partial \mathbf{F}}{\partial \mathbf{u}} \right|_{\mathbf{u}=\mathbf{u}^*} \in \mathbb{R}^{N \times N}, \quad (6.14)$$

evaluated at a reference state $\mathbf{u}^*$ (typically a numerical steady state). The eigenvalues $\{\lambda_i\}_{i=1}^N$ of $\mathbf{J}(\mathbf{u}^*)$ describe how small perturbations are amplified or damped by one time step of the method. We summarize this through the spectral radius

$$\rho(\mathbf{J}(\mathbf{u}^*)) = \max_i |\lambda_i|. \quad (6.15)$$

If $\rho(\mathbf{J}) \leq 1$, perturbations do not grow from one step to the next and the scheme is linearly stable at $\mathbf{u}^*$; if $\rho(\mathbf{J}) > 1$, at least one mode grows and the scheme is linearly unstable.

In our implementation, we compute $\mathbf{J}(\mathbf{u}^*)$ by automatic differentiation of the full discrete map $\mathbf{F}(\mathbf{u}; \boldsymbol{\theta}, \kappa, \Delta t)$ with respect to the state $\mathbf{u}$, treating $(\boldsymbol{\theta}, \kappa, \Delta t)$ as fixed parameters. This provides a square $N \times N$ Jacobian directly in the same functional representation used during training. By monitoring the largest-magnitude eigenvalue $\max_i |\lambda_i|$ as a function of training time and of $(\kappa, \Delta t)$, we obtain a diagnostic that both anticipates and explains failures of the learned integrator.

6.3 Burgers Diffusion: Framework I (E-IVP) Model Evaluation

In this section we study the performance of the proposed hybrid scheme in the explicit parabolic–timestep setting (Framework I). Viscous Burgers’ equation provides a clean test case for isolating the learned diffusive divergence from the hyperbolic flux. We generate data up to a terminal time of $t = 0.5$ and then continue the hybrid solver beyond the data horizon to evaluate long–time behavior. The focus here is exclusively on the machine–learning components: architectural choices, loss–function variants, and robustness to noise. All numerical discretizations (MUSCL–Rusanov for advection and centered diffusion) remain fixed throughout this section.

6.3.1 Architecture Comparison

We begin by comparing three architectures trained with the same baseline MSE loss: a DNN, a Fourier Neural Operator (FNO), and a U-shaped neural operator (UNO). Each model contains approximately 0.2M parameters. Figure 6.1 shows the continuation error of each model.

DNN exhibits a sharp growth in L_2 error almost immediately after the data horizon. FNO achieves a lower error at early times but gradually drifts and tends toward the same runaway behavior as DNN at later times. UNO maintains substantially better long-term stability, outperforming the other architectures by roughly half an order of magnitude in the continuation interval. Based on this comparison, we select UNO for all subsequent experiments in this section.

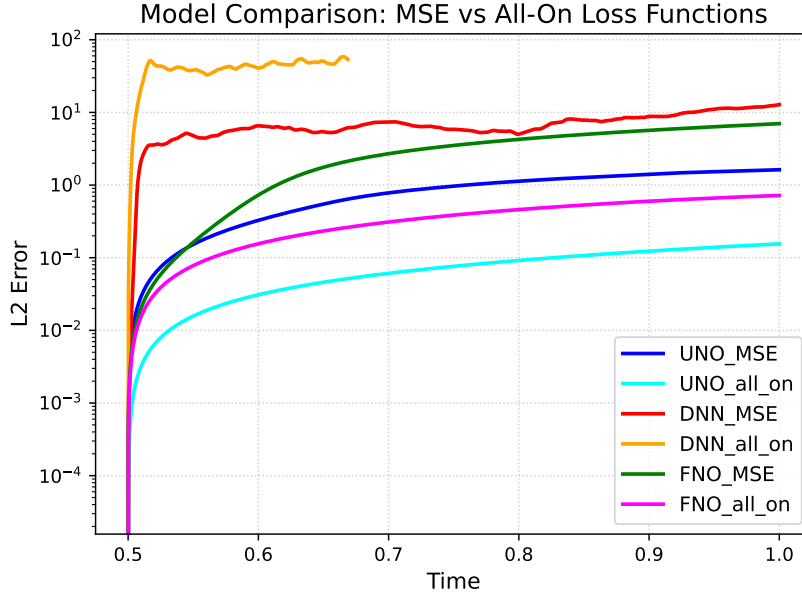


Figure 6.1: Architecture comparison for the Burgers E-IVP setting. All models have ≈ 0.2 M parameters and are trained with the same baseline MSE loss.

Loss Function Variants for UNO

We now study the effect of the loss-function components defined in Equation 6.12. Figure 6.2 compares the L_2 continuation error for the MSE-only model, models trained with spectral or substep consistency terms, and the full “all-on” configuration. The corresponding spectral error evolution is shown in Figure 6.3, and representative spectral-error heatmaps are collected in Figure 6.4.

The heatmaps reveal that the UNO MSE-only and MSE+spectral models behave similarly: neither captures sufficient temporal structure to predict beyond the next step, and their high-frequency grow rapidly in time. Introducing substep consistency (1, 5, and 10 substeps) progressively increases the time over which high-frequency modes remain stable; the associated error structures propagate farther upward in time in the heatmap. TVD produces similar behavior with slightly reduced low-frequency error. The substep term implicitly provides some spectral regularization.

In the L_2 error curves, the TVD model achieves the lowest loss, nearly two orders of magnitude below the MSE baseline. The all-on, substep-10, and substep-5 models form a cluster with similar error levels, followed by substep-1. MSE is the worst performer, and while the spectral loss improves early-time accuracy it eventually tracks back up toward the MSE curve at later times. The spectral-substep model very closely follows the MSE curve as well; despite containing a substep component, it does not provide additional benefit here.

Finally, the spectral-error-vs-time plot mirrors the L_2 behavior, indicating that the dominant source of error in these models originates in high-frequency content;

if the spectrum were not responsible for most of the drift, the L_2 and spectral curves would not correlate as strongly.

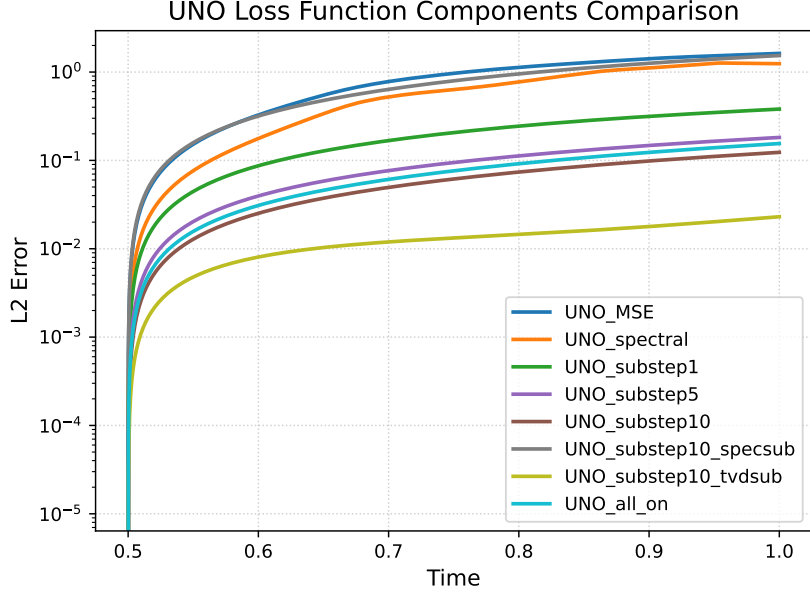


Figure 6.2: UNO loss-function comparison for Burgers E-IVP.

Noise Robustness

To assess robustness, we train UNO (with MSE only) under several levels of additive training noise. Figure 6.5 shows the resulting spectral-error curves, and Figure 6.6 displays the corresponding heatmaps.

The spectral-error curves exhibit a non-monotone dependency on noise magnitude. Noise levels 10^{-2} , 10^{-5} , and 10^{-1} produce lower spectral error than the no-noise model, and these three curves nearly overlap. Noise levels 10^{-4} and 10^{-3} give larger errors. This suggests the presence of an optimal noise band: moderate levels of noise appear large enough to regularize the operator and prevent overfitting to brittle high-frequency

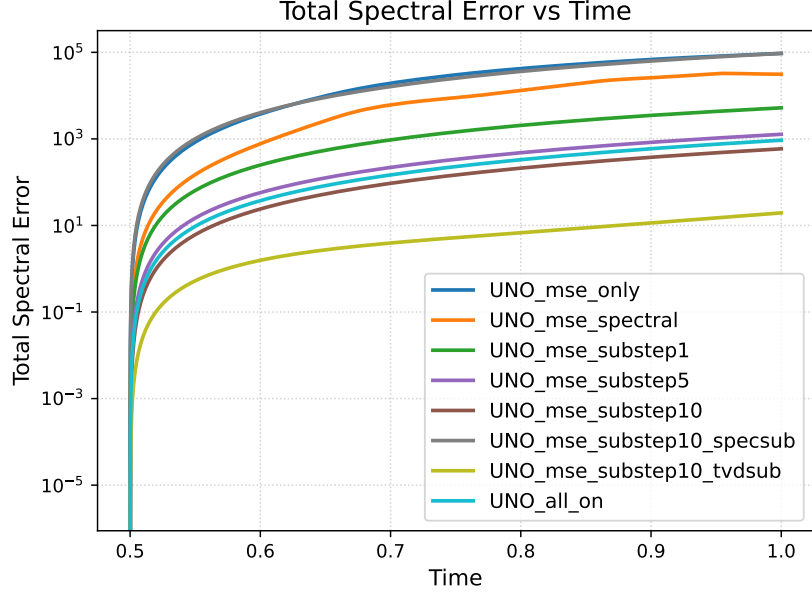


Figure 6.3: Spectral error vs time for UNO under the same loss configurations as in Figure 6.2.

correlations, but not so large as to distort the underlying diffusive dynamics. The heatmaps themselves do not reveal a clear trend in spectral structure between noise variants, indicating the noise does not help the model recover from high frequency noise error as initially predicted.

6.4 Burgers Diffusion: Framework II (I-DS) Evaluation

In this section we evaluate the I-DS (Implicit-Dataset) training framework on viscous Burgers' equation. Unlike the explicit E-IVP setting where the network predicts the explicit diffusive divergence, here the model learns the implicit diffusive increment

$$D_{\text{imp}}(U^n) = U^{n+1} - U^n, \quad (6.16)$$

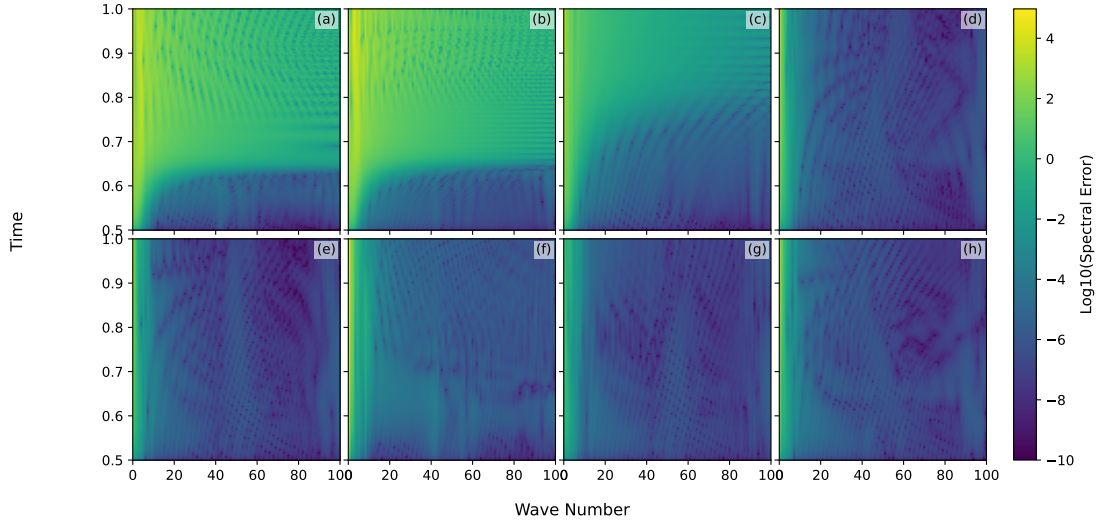


Figure 6.4: Representative spectral-error heatmaps for UNO under various loss configurations.

corresponding to the one-step IMEX update in Equation 6.5. This implicit formulation enables testing across diffusivity κ (interpolation and extrapolation) as well as evaluating long-horizon behavior starting from the raw initial condition at $t = 0$.

6.4.1 Kappa Generalization and Loss Behavior

We first examine error vs. time across several diffusivities, $\kappa \in \{0.06, 0.17, 0.22, 0.28\}$ and loss configurations. For each κ , Figures 6.7–6.10 show the L_2 error curves over time for the UNO variants (MSE, spectral, substep, TVD, and all-on).

Across all κ , the spectral-only model is consistently the worst performer. For $\kappa = 0.06$ and $\kappa = 0.17$, the MSE-only model is the second-worst, with substep-based losses performing better. At higher κ , however, the ordering reverses: for $\kappa = 0.22$, the substep-1 variant performs the best among the substep family, and for $\kappa = 0.28$,

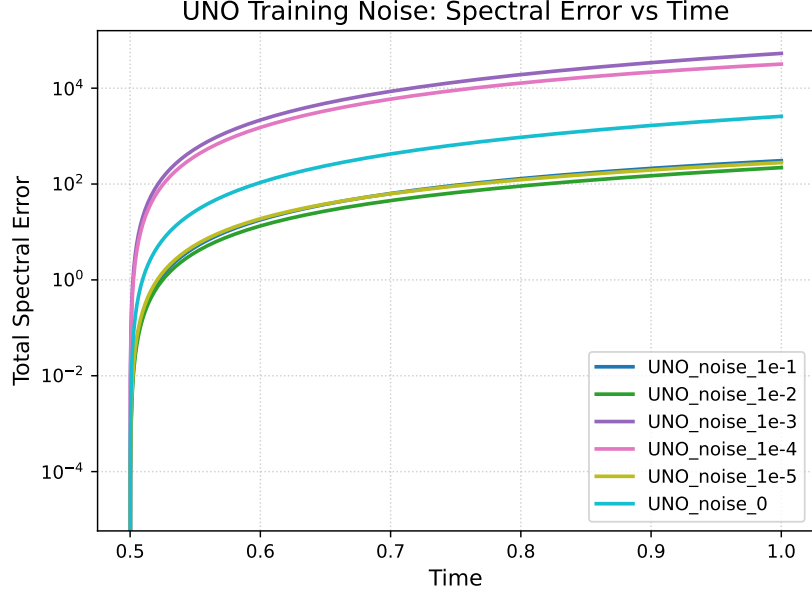


Figure 6.5: Spectral error vs time for UNO trained with different noise levels.

the MSE-only model becomes one of the top performers. The rankings vary non-monotonically with κ : configurations that perform best at small κ can perform poorly at larger κ , and vice versa. The only consistent pattern is that spectral-only is always the worst.

At $\kappa = 0.06$, the all-on configuration has the lowest error, followed by substep-1 and substep-5, while spectral-only is worst. At $\kappa = 0.17$, substep-1 is dramatically better than all other variants, with all-on and substep-10 forming a mid-range group. At $\kappa = 0.22$, substep-10-spectral-substep achieves the lowest error, with all-on among the poorest performers. At $\kappa = 0.28$, the top group includes the MSE-only, substep-10, and substep-10-spectral-substep models, while all-on, substep-1, and substep-5 degrade substantially.

Overall, no single loss configuration is uniformly best across κ . Performance

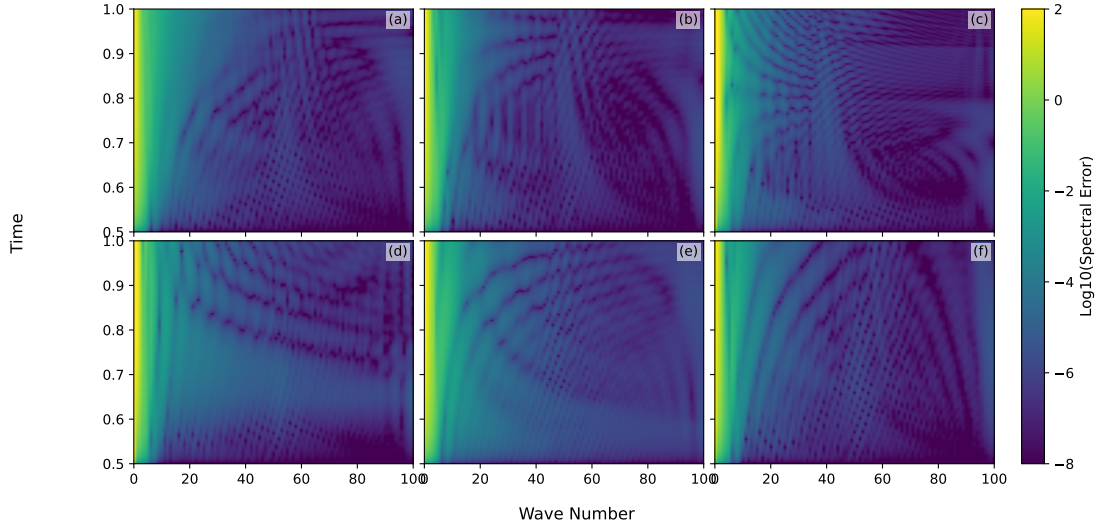


Figure 6.6: Spectral-error heatmaps for different noise levels in UNO (MSE-only).

depends strongly on diffusion strength, with high- κ regimes favoring different loss combinations than low- κ regimes.

6.4.2 Spectral Behavior Across κ

To better understand how the implicit operator behaves across diffusion strengths, we compare spectral-error heatmaps for two representative cases, $\kappa = 0.06$ and $\kappa = 0.28$, shown in Figures 6.11 and 6.12.

At $\kappa = 0.28$, the spectral structure is less clean than in the explicit E-IVP case, but there is a clear improvement in high-frequency behavior when increasing substeps: the MSE and substep-1 models exhibit noticeably worse high-frequency error compared to substep-5 and substep-10. The TVD variant performs particularly well, with strong suppression of high-frequency error at later times.

At $\kappa = 0.06$, improvements are smaller in magnitude and trends are less pro-

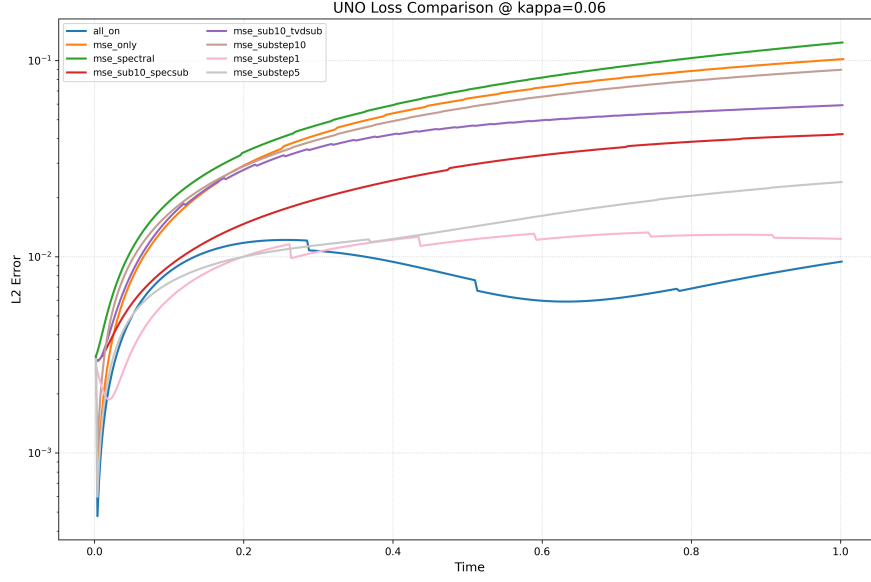


Figure 6.7: UNO L_2 error vs. time for Burgers I-DS with $\kappa = 0.06$ across several loss configurations.

nounced. There is some reduction in late-time high-frequency error when comparing substep-1 to substep-10, but differences across variants are relatively modest.

6.4.3 Trajectory from $t = 0$

We now examine whether the learned implicit operator can reconstruct solution trajectories when started directly from the initial condition. Figure 6.13 compares UNO_mse and UNO_all_on.

The left panel shows rollouts to $t = 0.25$ for $\kappa \in \{0.025, 0.045, 0.085, 0.205\}$. Across all four cases, UNO_all_on exhibits visibly better agreement with the reference solution compared to UNO_mse, with reduced phase and amplitude error and improved shape preservation.

The right panel examines longer rollouts for $\kappa = 0.045$ at $t = 0.25, 0.5, 0.75$.

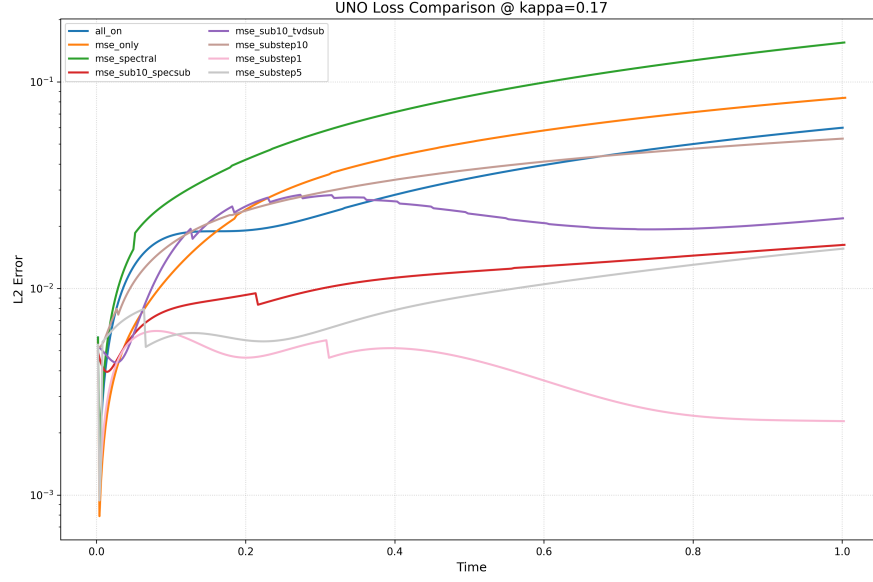


Figure 6.8: UNO L_2 error vs. time for Burgers I-DS with $\kappa = 0.17$ across several loss configurations.

UNO_all_on shows almost no divergence from the reference even at $t = 0.75$, whereas UNO_mse steadily degrades as time advances. These results highlight the benefit of the richer loss structure: the MSE-only model fails under long-time continuation, while the all-on configuration maintains stable and accurate trajectories over extended horizons.

6.4.4 Numerical Properties of the Learned Diffusion Operator for Burgers

In this section we examine several numerical aspects of the learned diffusion operator for viscous Burgers: the choice of predicting the divergence of the diffusive flux versus the flux itself, the sensitivity of the hybrid scheme to switching the underlying hyperbolic discretization, and the behavior of the one-step eigenvalues as the diffusivity κ varies.

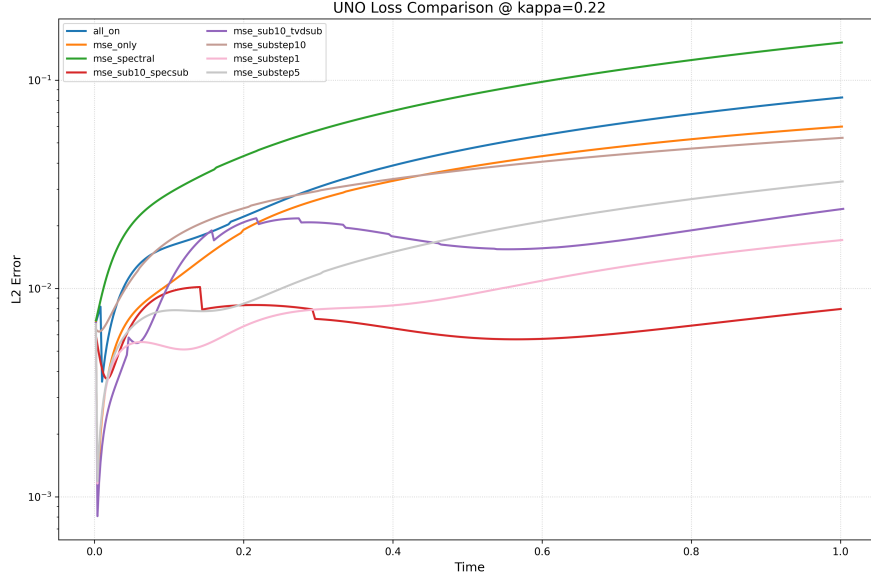


Figure 6.9: UNO L_2 error vs. time for Burgers I-DS with $\kappa = 0.22$ across several loss configurations.

6.4.4.1 Predicting $\nabla \cdot F$ versus Predicting F

In Framework I there are two natural choices for the learned parabolic term. One can either (i) predict the raw diffusive flux F and then apply the finite-volume divergence operator, or (ii) predict the divergence $\nabla \cdot F$ directly. The second approach has two practical advantages.

First, predicting the divergence directly ensures that the additional term is conservative in the finite-volume sense. The network output is interpreted as a cell-centered approximation to the diffusive divergence, and when this is used inside a conservative update, the discrete sum of the added term over the domain can be enforced to vanish. In contrast, predicting fluxes and then differencing them can easily introduce small inconsistencies at cell faces that lead to a slow drift in the total conserved quantity.

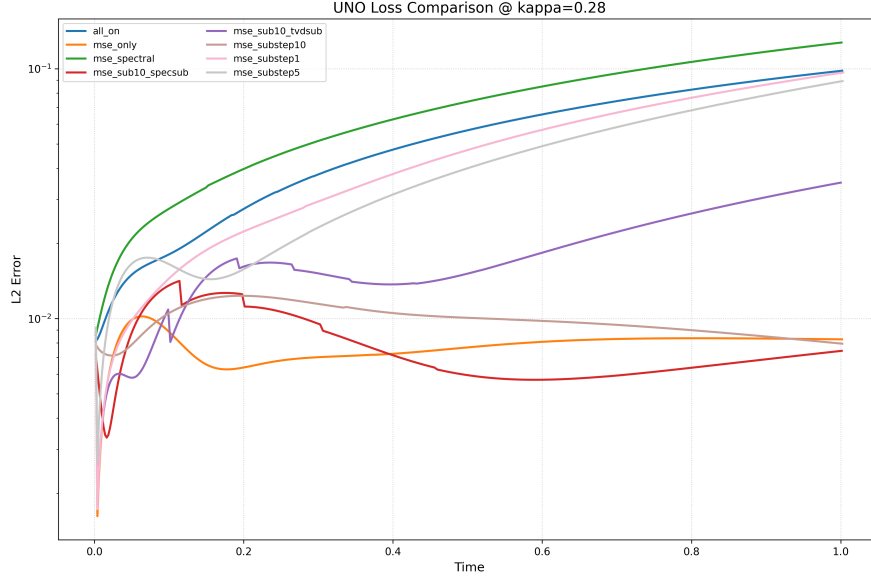


Figure 6.10: UNO L_2 error vs. time for Burgers I-DS with $\kappa = 0.28$ across several loss configurations.

Second, a simple spectral argument shows that predicting F and then taking a discrete divergence amplifies high frequencies. In Fourier space, a diffusion operator has a symbol proportional to $-k^2$, whereas a flux has symbol proportional to $-k$; applying a discrete derivative to the learned flux reintroduces an additional factor of k and therefore magnifies any high-frequency error in the network output. This effect is illustrated in Figure 6.14, which compares the spectral error obtained when predicting the divergence directly versus predicting the flux and differencing it.

Figure 6.15 shows the corresponding drift in total conserved quantity over time. When predicting fluxes, small interface inconsistencies accumulate and lead to a noticeable conservation drift, whereas predicting the divergence directly yields a much flatter curve. Together these results motivate our choice to treat $\nabla \cdot F$ as the primary

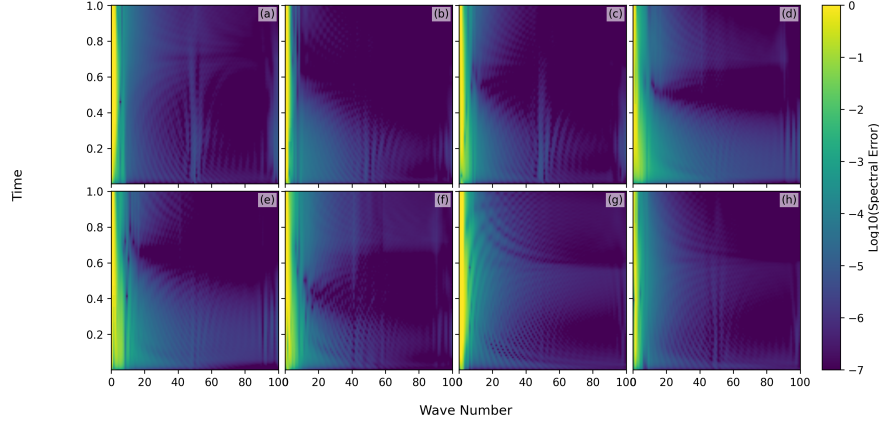


Figure 6.11: Spectral-error heatmap for Burgers I-DS with $\kappa = 0.06$ across UNO loss configurations.

learning target in the Burgers experiments.

6.4.4.2 Sensitivity to Switching the Hyperbolic Discretization

The learned diffusion operator is trained in the context of a particular hyperbolic discretization, and there is no guarantee that it will remain well behaved if the underlying hyperbolic scheme is changed. To study this effect, we examine two “crossed” configurations: (A) a model trained with MUSCL for the hyperbolic term but used with a WENO flux at test time (UNO_MUSCL→WENO), and (B) a model trained with WENO but tested with MUSCL (UNO_WENO→MUSCL).

Figure 6.16 shows the behavior of both configurations. The top panel plots the L_2 error over time, while the bottom panel plots the maximum eigenvalue of the one-step Jacobian on a synchronized time axis.

In the UNO_MUSCL→WENO case, the eigenvalue curve begins near 1 but rises sharply as the solution evolves, exceeding 9 shortly before the scheme blows up

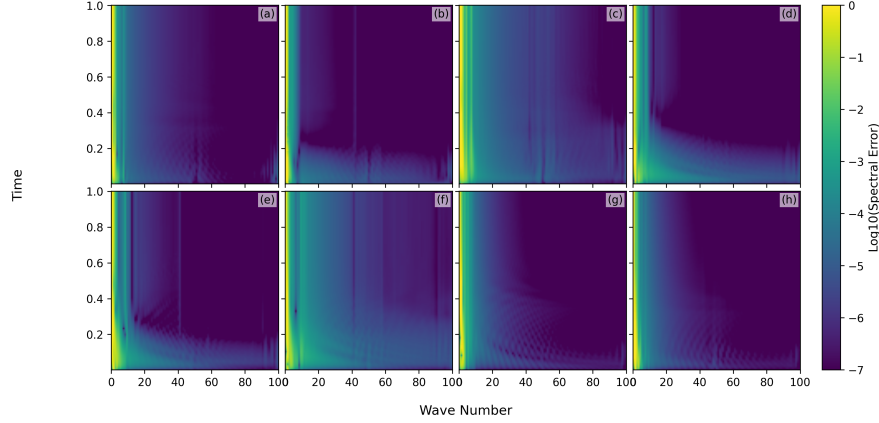


Figure 6.12: Spectral-error heatmap for Burgers I-DS with $\kappa = 0.28$ across UNO loss configurations.

around $t \approx 0.57$. The corresponding L_2 error curve exhibits the same catastrophic growth. In contrast, the UNO_WENO→MUSCL case (configuration B) shows a maximum eigenvalue of approximately 1 at the start but this value remains steady throughout the rollout, and the solution remains stable over the entire 2.5 s continuation interval. We emphasize that stable does not mean accurate in this case.

These behaviors are visible directly in physical-space profiles. Figure 6.17 shows configuration A at $t = 0.52, 0.54, 0.56$. The solution departs rapidly from the reference and exhibits clear signs of instability, consistent with the eigenvalue blowup. In contrast, Figure 6.18 shows configuration B at $t = 0.6, 0.75, 1.0$ (Framework I continuation starting from the training horizon $t = 0.5$). Here the solution remains stable, with moderate error but no qualitative breakdown.

Spectral error heatmaps reinforce this picture. Figure 6.19 shows errors exceeding $\mathcal{O}(1)$ across nearly the entire spectrum by $t \approx 0.54$, precisely where the solution destabilizes. Figure 6.20 exhibits some low-frequency error growth but maintains good

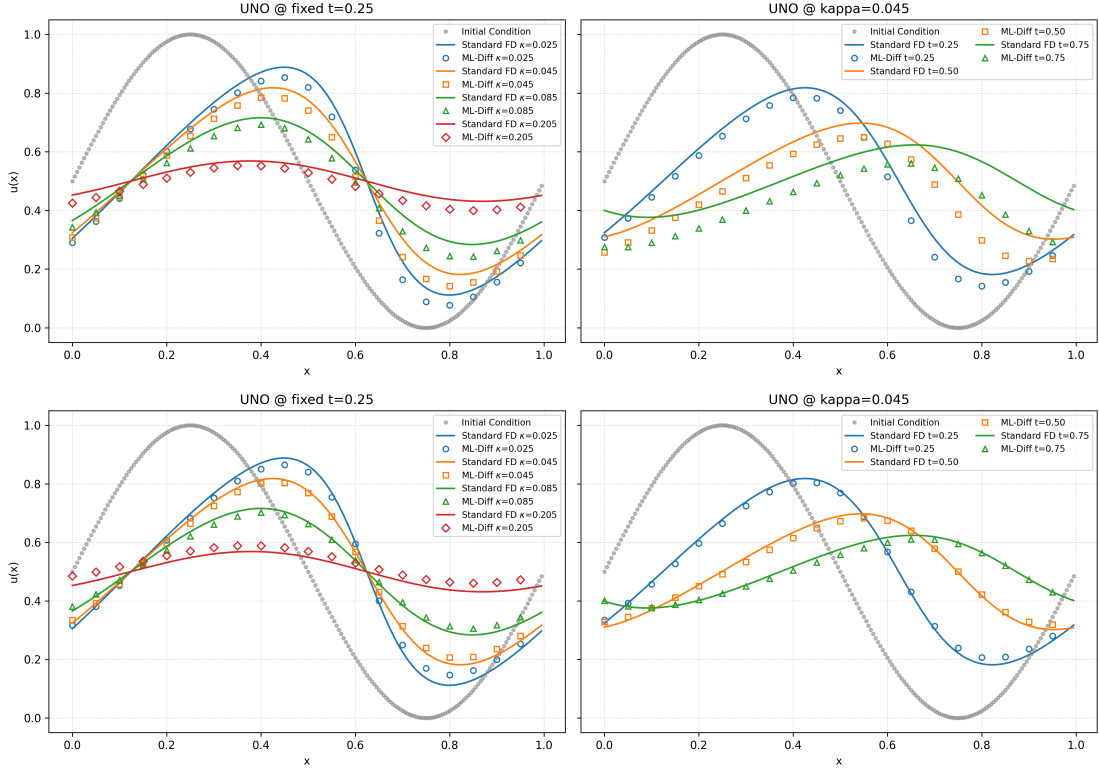


Figure 6.13: Rollouts from $t = 0$ for several κ values (top) and for $\kappa = 0.045$ over longer time horizons (bottom). UNO_all.on maintains much better long-time fidelity than UNO_mse.

high-frequency behavior and remains bounded in time.

Overall these results make clear that the learned diffusion operator is not interchangeable across hyperbolic discretizations: switching from the more dissipative MUSCL to the less dissipative WENO flux destabilizes the coupled system (A), whereas switching in the opposite direction (B) introduces additional numerical dissipation and remains stable. Stability therefore depends critically on the specific pairing between the learned parabolic operator and the underlying hyperbolic flux.

These crossed experiments highlight that the learned operator has implicitly

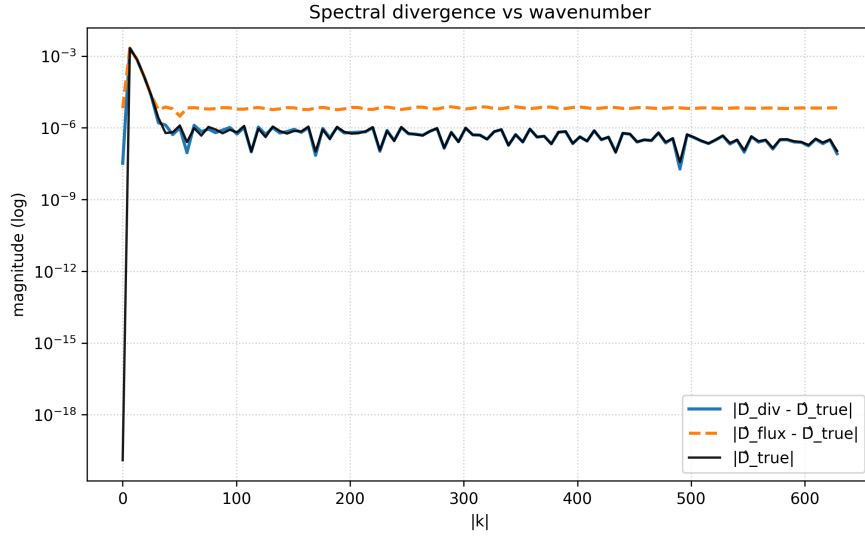


Figure 6.14: Comparison of spectral error when predicting the diffusive divergence directly versus predicting the flux F and taking a discrete divergence. Predicting F leads to amplification of high-frequency error when the divergence operator is applied.

adapted to the diffusion level of the hyperbolic flux used during training. When a model trained together with the more dissipative MUSCL flux is instead paired with a higher-order, less diffusive WENO flux at test time, the missing background diffusion causes the rollout to blow up. In the reverse direction, pairing a model trained with WENO with the more diffusive MUSCL flux simply adds extra damping: the solution remains stable, but the profiles become noticeably more diffusive and less accurate, as expected.

6.4.4.3 Eigenvalue Dynamics Across the Diffusivity κ

Finally, we examine how the eigenvalues of the learned implicit predictor behave as the diffusivity κ is varied. Figure 6.21 shows a κ -sweep from $\kappa = 10^{-3}$ to $\kappa = 3$,

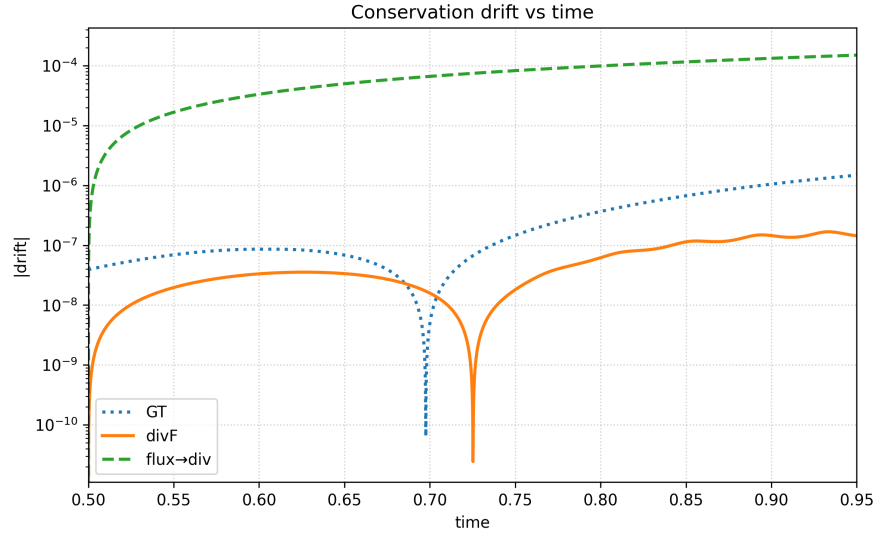


Figure 6.15: Drift in total conserved quantity over time when predicting $\nabla \cdot F$ directly versus predicting F and differencing it. Predicting F leads to a noticeable conservation drift, while predicting the divergence is substantially more conservative.

plotting as a function of time the maximum eigenvalue of the one-step prediction operator for each κ in the I-DS framework.

In general, smaller values of κ produce eigenvalues closer to 1, indicating a more marginally stable update. As κ increases, the maximum eigenvalue moves farther below 1, reflecting stronger diffusive damping. However when we reach $\kappa = 3$ which is significantly outside the training data we see larger eigenvalues again. Across the entire sweep, however, all curves remain strictly below 1, so the learned implicit predictor is linearly stable for every diffusivity tested. This is a positive indication of the robustness of the implicit hybrid solver, even though stability does not automatically imply accuracy. The observed errors in the Burgers I-DS experiments still vary substantially with κ .

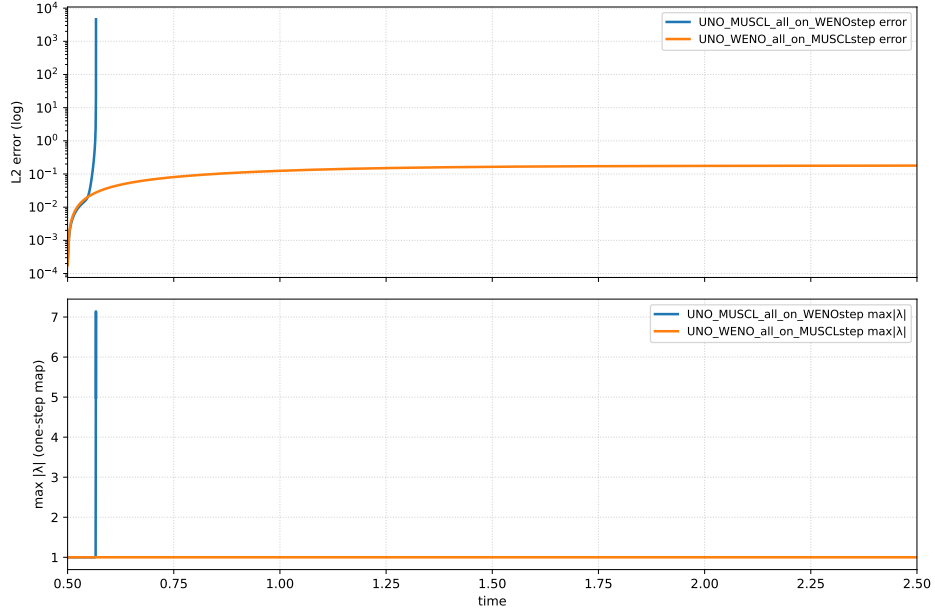


Figure 6.16: Top: L_2 error vs. time for the two crossed configurations UNO_MUSCL $\rightarrow$ WENO (unstable) and UNO_WENO $\rightarrow$ MUSCL (stable). Bottom: corresponding maximum eigenvalue of the one-step Jacobian. Configuration A experiences an eigenvalue blowup exceeding 9 and fails at $t \approx 0.57$, while configuration B maintains a roughly constant eigenvalue near 4 and remains stable over the full 2.5s rollout.

6.5 Euler IVP with Implicit Heat Conduction (Framework II)

We now evaluate the implicit-dataset (I-DS) framework on the one-dimensional Euler equations with Fourier heat conduction. In this setting the network predicts the implicit diffusive increment applied to the energy equation, while the hyperbolic fluxes are handled by a standard finite-volume solver. Training data consist of short IMEX rollouts from a variety of smooth and discontinuous initial conditions (e.g. shock, sinu-

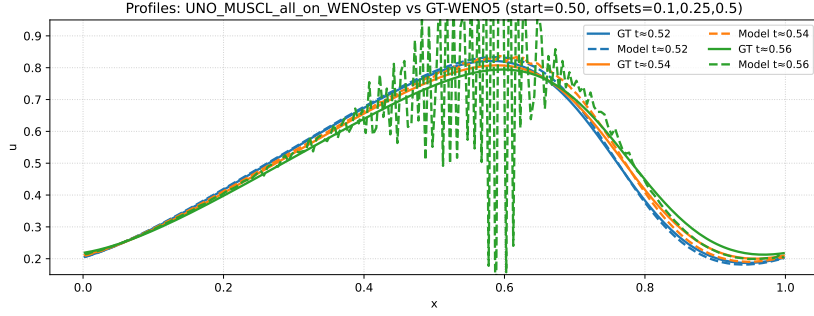


Figure 6.17: Configuration A (UNO_MUSCL→WENO). Physical profiles compared to ground truth at $t = 0.52, 0.54, 0.56$. The solution rapidly destabilizes, consistent with the eigenvalue blowup.

soidal temperature, Gaussian temperature), and include a range of thermal conductivities $\kappa \in [0.5, 4.0]$. As in the Burgers experiments, each training sample provides the current conservative state, the local timestep, and the dt-weighted divergence of the heat flux. All continuation tests begin from the final training time.

6.5.1 Loss-Function Behavior

Figure 6.22 shows a representative loss-function comparison at $\kappa = 0.5$ over a training horizon of $t \in [0.5, 2.0]$. The all-on configuration performs best, maintaining the lowest L_2 error throughout the rollout. It is followed by substep 10, then the spectral loss, and finally the MSE+spectral+specsub combination. The one-substep model behaves similarly to pure MSE, while five substeps performs noticeably worse, and ten substeps again improves. Overall the ranking is less structured than in the Burgers IVP case, but the all-on loss remains consistently the most robust.

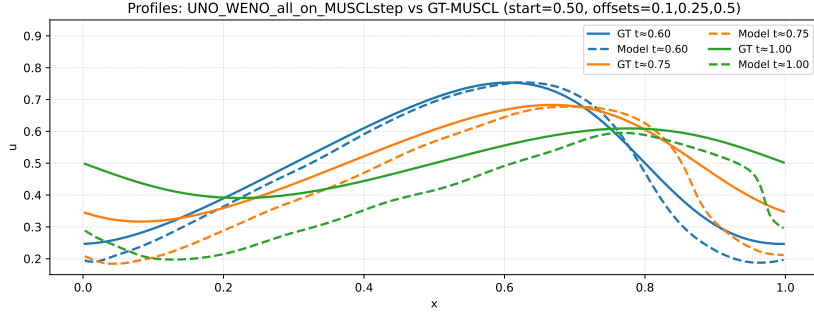


Figure 6.18: Configuration B (UNO_WENO→MUSCL). Profiles at $t = 0.6, 0.75, 1.0$ (Framework I continuation). The solution remains stable with moderate error but no runaway growth.

6.5.2 Spectral Error Structure

To examine how these loss variants differ spectrally, Figure 6.23 shows the spectral-error heatmaps at $\kappa = 0.5$. Unlike the Burgers IVP case, clear trends are not apparent: most variants yield similar spectral patterns, with the exception of substep 5, which exhibits substantially larger high-frequency error. This is likely due to the first-order update structure used in these experiments, which introduces additional numerical diffusion and tends to blur any fine-grained spectral differences. A higher-order implicit update would help clarify these trends and is left for future work.

6.6 Conclusion

We have investigated a hybrid numerical–machine learning strategy for mixed hyperbolic–parabolic PDEs in which traditional finite-volume discretizations handle the hyperbolic fluxes, while a neural network predicts only the parabolic contribution. By focusing the learned component on the diffusive term, either as a cell-centered diver-

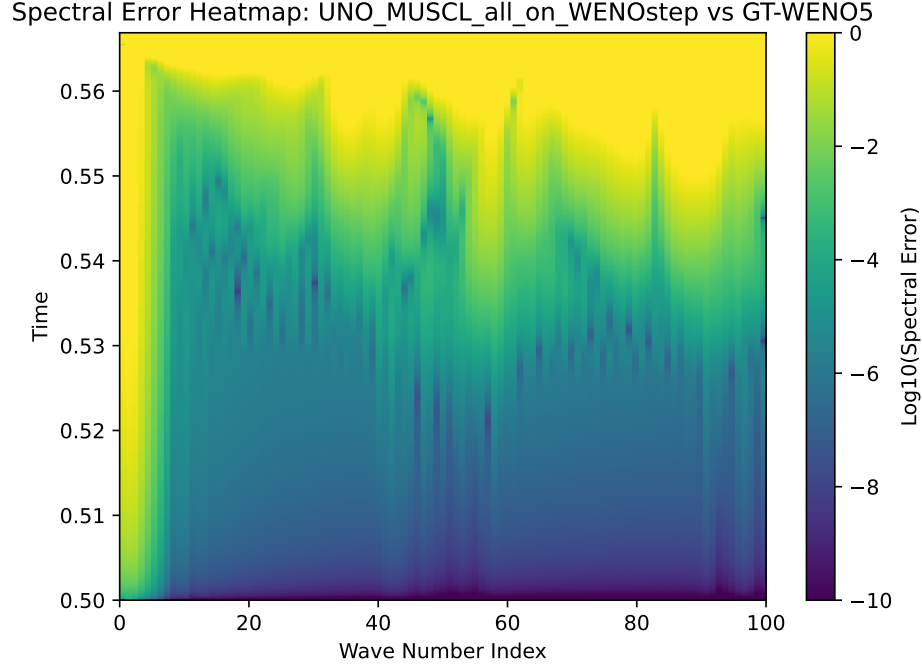


Figure 6.19: Spectral error heatmap for configuration A. By $t \approx 0.54$, errors exceed $\mathcal{O}(1)$ across the spectrum, matching the onset of instability.

gence (E-IVP) or as an implicit one-step increment (I-DS), the framework targets the parabolic timestep restriction directly without discarding the structure, conservation properties, and shock-capturing capabilities of the underlying solver.

Across viscous Burgers' tests we found that U-shaped neural operators provided improved long-time behavior relative to fully connected networks and Fourier Neural Operators, especially when combined with multi-step and TVD-inspired loss terms that suppress high-frequency error growth and reduce continuation error beyond the training horizon. In the implicit-dataset setting the learned diffusive increment generalizes across a broad range of diffusivities κ and remains linearly stable in eigenvalue tests, although performance is problem dependent and no single loss configuration

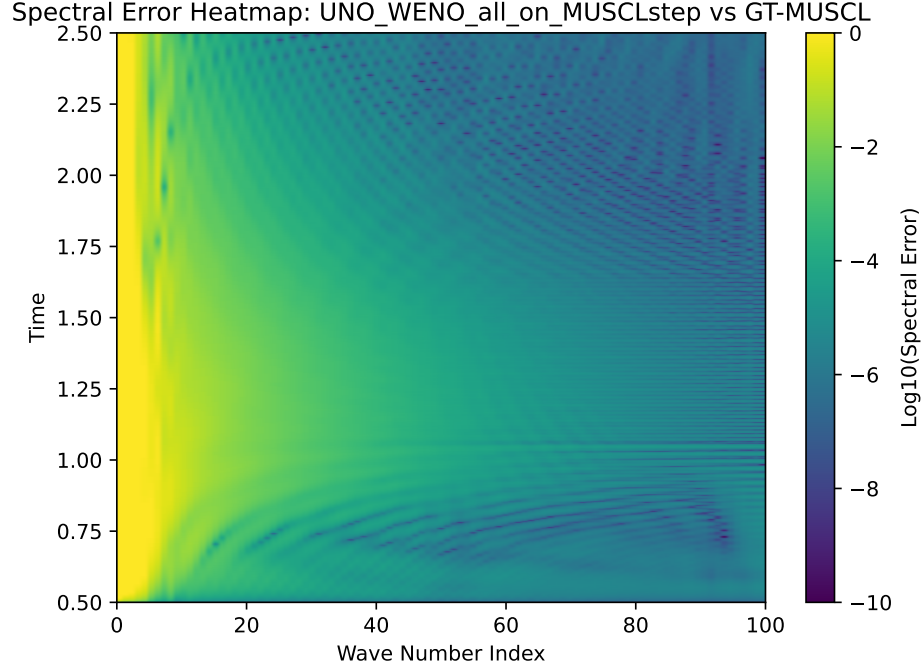


Figure 6.20: Spectral error heatmap for configuration B. Some low-frequency error develops, but high-frequency behavior remains controlled and the solution stays bounded.

dominates across all diffusion strengths. For the Euler heat-conduction experiments the all-on configuration is consistently among the most accurate, but differences between loss variants are smaller, in part due to the first-order time integration used here.

Our numerical and stability studies highlight an important coupling effect. The learned parabolic operator implicitly adapts to the diffusion level supplied by the hyperbolic flux. When a model trained with a more dissipative MUSCL flux is paired with a higher-order, less diffusive WENO flux, eigenvalues of the one-step map grow rapidly and the scheme becomes unstable. Reversing this pairing adds extra damping and yields a stable but more diffusive solution. This sensitivity emphasizes that learned operators cannot be treated as interchangeable black boxes and must instead be designed

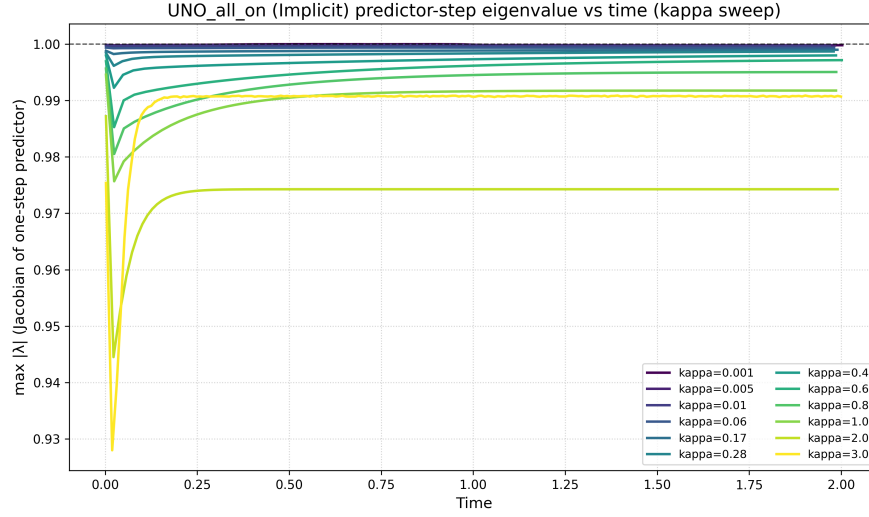


Figure 6.21: Maximum eigenvalue of the one-step implicit prediction operator as a function of time for a sweep of diffusivities $\kappa \in [10^{-3}, 3]$. Smaller κ values produce eigenvalues closer to 1, while larger κ yields stronger damping (eigenvalues further below 1). All curves remain below 1, indicating linear stability across the entire κ range.

and trained with the discretizations and coupled schemes in mind.

The same framework has important implications for large-scale multidimensional simulations. In traditional IMEX and fully implicit schemes the diffusive update is obtained from a global linear or nonlinear solve that requires iterative methods with repeated sparse matrix vector products and global communication, especially at scale with MPI. In contrast, the learned parabolic operator is evaluated through neural network forward passes that are dominated by local dense tensor operations and are embarrassingly parallel across grid points, batches, and multiple right-hand sides. On modern hardware this computation maps naturally to GPUs and distributed data parallelism and can proceed with minimal inter-node communication. For high-dimensional problems

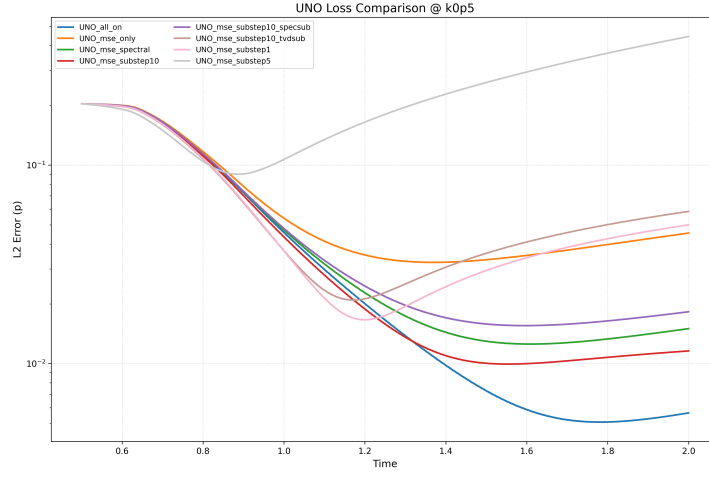


Figure 6.22: UNO loss-function comparison for Euler I-DS at $\kappa = 0.5$, continued from $t = 0.5$ to $t = 2.0$. The all-on loss yields the lowest error, followed by substep 10 and the spectral loss.

and long time integrations where implicit solves are a major cost, replacing the parabolic solve with a learned operator therefore offers a clear path to significant speedups while preserving the backbone of a conservative finite-volume hyperbolic solver.

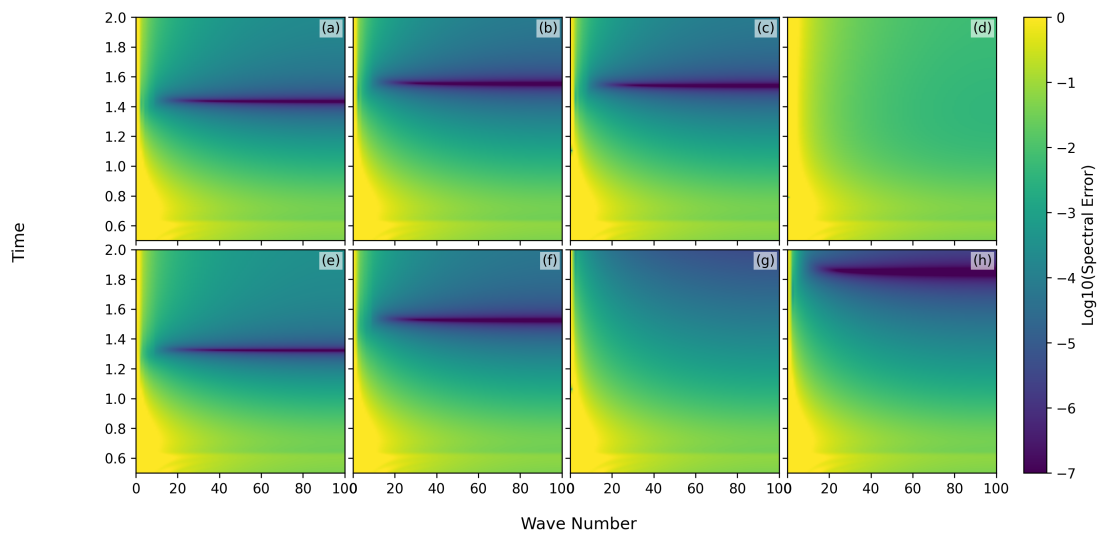


Figure 6.23: Spectral-error heatmaps for Euler I-DS at $\kappa = 0.5$. Most loss configurations exhibit similar patterns, with substep 5 showing a noticeably poorer high-frequency behavior.

Chapter 7

Multiphase LaunchPad Simulations

7.0.1 Motivation

NASA's Artemis program [138] marks a significant step in space exploration, aiming to return humans to the Moon and eventually prepare for manned missions to Mars. This mission uses the Space Launch System (SLS) [107] which is the most powerful rocket designed by NASA. Artemis I launched on November 16, 2022 at Kennedy Space Center's (KSC) Launch Complex 39B (LC-39B) on top of the Mobile Launcher (ML-1). The launch of a rocket produces powerful pressure waves at startup known as ignition overpressure (IOP) waves as well as induce extreme pressure environments on the surrounding structures. These can be dangerous to the vehicle, pad, or the payload. Starting with the Shuttle Mission STS-1, water-based sound suppression has been used as an attempt to help control some of these powerful acoustic waves as well as dampen pressure loads from the plume on the pad and structure. During the Artemis I launch the ML-1 sustained significant damage, most notably the elevator doors and umbilicals [70]. With the Artemis II launch approaching, there is interest to both understand the

environment which caused these damages as well as prepare for a worst case trajectory, in which the vehicle takes off close to the tower (leading to the highest pressure forces on the tower). This information will ultimately go towards structural improvements for the Mobile Launcher 2 (ML-2) which Artemis II will use.

The first NASA mission to use a water-based sound suppression system was the launch of Space Shuttle Columbia (STS-1) on April 12, 1981 which introduced the Space Shuttle heritage Sound Suppression Water System (SSWS). SSWS went on to support 135 successful launches over the course of 30 years [63]. Similar water systems have now been a part of many launch environments for many years. The European Space Agency (ESA) showed 3-5dB reduction with their water-based acoustic suppression systems on the VEGA launchpad [113]. There are two primary types of water systems used for acoustic wave suppression [98]. The first is a below the deck where water is injected into the exhaust plume which has been shown to reduce far-field noise by dispersing rocket exhaust [6]. In the context of the ML-1 water system, this is known as the IOP water system. The second is an above-deck system where water is injected over the deck. This can be utilized for sound suppression as well as protection of the launchpad via cooling. The ML-1 has 5 rainbirds which represent the above-deck system which offer a 2-3dB reduction [65]. The ML-1 has a third water source on the crest of the flame deflector below the exhaust hole. Other noise reduction techniques include using a Flame Deflector and trench to channel the exhaust away from the pad [146] as well as optimization of rocket nozzle design [149].

Predicting the acoustic environment prior to launch has always been a challenge. Empirical methods exist [115] but are rough estimates and do not capture tran-

sients nor multi-physics effects. Numerical simulation can give us a much better estimate of the environment without over-simplifications as well as account for all the effects and geometry we need.

Using the Launch Ascent and Vehicle Aerodynamics (LAVA) framework [74], [14] NASA has simulated launch environments which have helped characterize launch environments prior and post launch [106], [8], [15].

7.0.2 Introduction

To analyze launch environments effectively requires high-fidelity multiphase CFD simulations with physics models to capture effects of evaporation and condensation of the launchpad water systems interacting with the hot plume gas. Therefore, in this section we are interested in simulating compressible multiphase flow. Most literature focuses on the numerical modeling of two-phase flows which is what we will use to introduce this topic. There are several approaches to model two phase flows but the most popular are Two-fluid models and One-fluid models.

Two-fluid models [66] take the approach of modeling each phase with its own separate set of conservation equations. There then needs to be an interphase exchange of all the conserved quantities. Two-fluid models are impractical at the scale of launchpad simulations and other problems of engineering relevance [117]. We therefore constrain ourselves further to the more practical one-fluid model in which there is a full range of approaches within.

The five equation model [4] assumes mechanical and thermo equilibrium between the two phases so there is only one additional equation for the conservation of

mass of each phase. This assumption implies that the fluid is well mixed on the scale of the grid resolution. We also advect the volume fraction, recognizing that volume fractions must sum to 1, we only need to advect one in the case of two phase flow. The five equation model gets its name because in 1D two-phase flows, this would lead to 5 equations. However in practice this expands to $2N + D$ where D is the problem's spatial dimensions and N is the number of species we are considering. It's important to note that the volume fraction isn't a conserved quantity and will require special consideration. The Allaire model can be written as for N species

$$\begin{aligned}
\partial_t (\alpha_k \rho_k) + \nabla \cdot (\alpha_k \rho_k \mathbf{u}) &= 0 \\
\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p &= 0, \\
\partial_t E + \nabla \cdot [(E + p) \mathbf{u}] &= 0, \\
\partial_t \alpha_k + \mathbf{u} \cdot \nabla \alpha_k &= 0, \quad k = 0, \dots, N-1,
\end{aligned} \tag{7.1}$$

where $\mathbf{u}$ is the velocity vector and p is the mixture pressure. α_k is the volume fraction of phase k and ρ_k is the density of the phase k . This system is not closed, in one spatial dimension there are 5 equations but 6 unknowns, $\alpha_1, \rho_1, \rho_2, u, e_1, e_2$ where e_k is the specific internal energy of phase k . We will need to close this system with an equation of state for each species as well as assuming each species is in pressure equilibrium.

Here we utilize the stiffened gas EOS for the liquid phase [62]

$$\frac{p_k}{\gamma_k - 1} + \frac{\gamma_k p_k^\infty}{\gamma_k - 1} = \rho_k e_k \tag{7.2}$$

where γ_k and p_k^∞ are parameters used to represent the fluid. γ_k is the ratio of

specific heats and p_k^∞ represents the compressibility of the gas, when $p_k^\infty = 0$ we recover the ideal gas law and use the ideal gas EOS for all other gases in the system.

We couple the hyperbolic step to two physics modules, a thermal relaxation step and a phase transition step. The thermal relaxation step relaxes the fluid system from the hyperbolic step such that each species temperature is equal. The Allaire hyperbolic step coupled with the thermal relaxation step was shown in [154] to be equivalent to solving the 4 equation homogeneous relaxation (HRM) model. The main benefit to using this approach rather than solving the HRM equations directly is because the HRM equations have a non monotonic speed of sound, known as the Wood speed of sound. This can cause difficulties in the numerical solution of the hyperbolic step because it can introduce sonic points even in subsonic flow [112]. Constructing a robust Riemann solver to handle this is a difficult task and the non-monotonicity could affect the propagation of acoustic waves [111]. The five-equation model by Allaire does not exhibit the non-monotonic speed of sound characteristic and we can take the hyperbolic step without this worry and then correct for the error incurred with a thermal relaxation step.

We use a phase transition model presented in [34] to account for the cavitating, evaporating and condensing of water between the liquid and vapor state. When at saturation, a state of thermal-chemical equilibrium is computed where the liquid and vapor phases have equivalent Gibbs free energies. This model has the capability to couple an arbitrary number of additional ideal gases but only resolves the transition effects between liquid water and vapor water. The Gibbs free energy is given by

$$g_k(p_k, T_k) = (\gamma_k c_{v,k} - q'_k) T_k - c_{v,k} T_k \ln \left[\frac{T_k^{\gamma_k}}{(p_k + p_k^\infty)^{\gamma_k - 1}} \right] + q_k. \quad (7.3)$$

where T_k represents the temperature of species k , p_k represents the pressure of species k , γ_k is the specific heat ratio (ratio of specific heats) for species k , $c_{v,k}$ is the specific heat at constant volume for species k , q'_k is a term related to the heat transfer for species k , q_k is a term related to the internal energy for species k , and p_k^∞ is a reference pressure for species k .

7.0.3 Artemis Launchpad Simulations

To understand the complex interactions and potential damages to the Mobile Launcher (ML-1 and ML-2) during Artemis missions, we present a suite of 14 lofted vehicle simulations. 13 of these simulations were run without the water systems denoted as “dry”. They are divided into two separate trajectories, one following the Artemis I launch and the other is a max drift trajectory for Artemis II where the vehicle flies closest to the tower. In these simulations, the vehicle is held at a static position corresponding to a time after lift off. The final simulation is the wet simulation which in addition to the vehicle exhaust simulates the water systems on the launchpad.

Each simulation aims to replicate the launch environment’s pressure conditions at various trajectory points. These simulations are designed to capture the peak loads and critical phases of the launch but are not time-accurate, meaning they do not precisely model the startup sequence or the continuous movement of the vehicle. Instead, the simulations use a lofted approach, where key moments are selected to represent the

most critical phases of the launch trajectory.

Unsteady phenomena, such as the ignition overpressure (IOP) waves and the complex interactions of the rocket plume with the launchpad structure, are inherently challenging to model due to their transient nature. By focusing on specific trajectory points, we aim to capture the maximum pressure loads that the launchpad and vehicle may experience. From pressure data, we can also derive acoustic signals to study the effect of water on loudness.

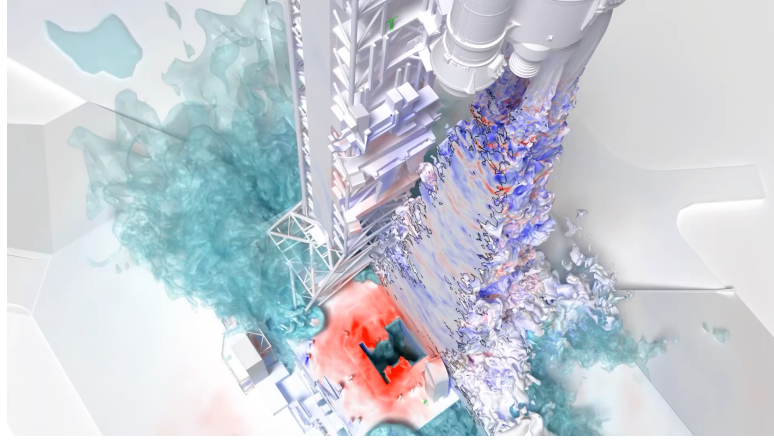


Figure 7.1: Lofted Artemis I launch simulation using the LAVA Cartesian computational fluid dynamics solver. The image shows a slice through the exhaust plumes where the black line indicates the 90% plume exhaust/air interface. Launchpad surfaces are colored by pressure, where red is high and blue is low. The water mass fraction is shown using nested isocontours in teal.

7.0.4 Simulation Methodology

Each dry simulation was performed using the LAVA Cartesian code using a 5th order accurate weighted essentially non-oscillatory (WENO5) convective flux, 3rd

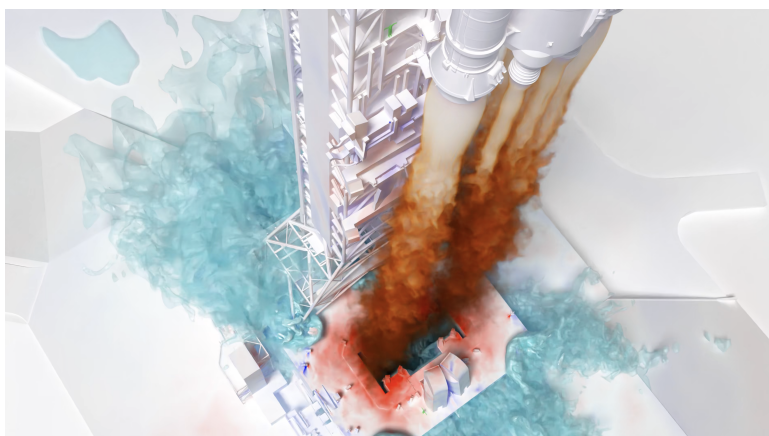


Figure 7.2: Lofted Artemis I launch simulation using the LAVA Cartesian computational fluid dynamics solver. Plume gas mass fraction is shown where white is large and dark orange is low. Launchpad surfaces are colored by pressure, where red is high and blue is low. The water mass fraction is shown using nested isocontours in teal.

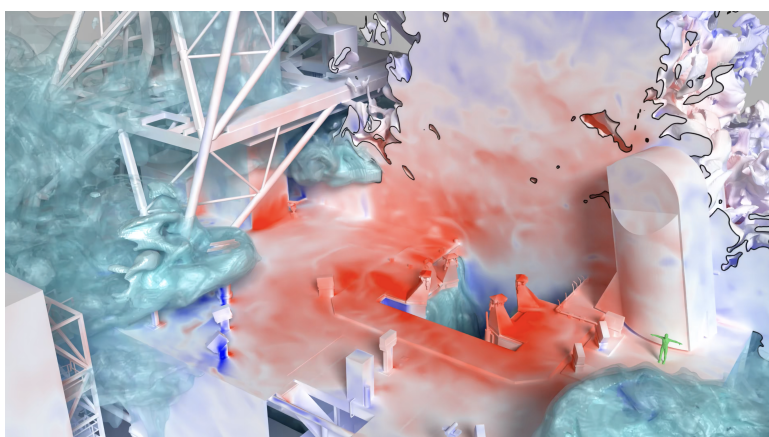


Figure 7.3: Lofted Artemis I launch simulation using the LAVA Cartesian computational fluid dynamics solver. The image shows a slice through the exhaust plumes where the black line indicates the 90% plume exhaust/air interface. Launchpad surfaces are colored by pressure, where red is high and blue is low. The water mass fraction is shown using nested isocontours in teal.

order Strong Stability Preserving Runge- Kutta (SSPRK33) time integration and an immersed-boundary method (IBM) detailed in [23] for the geometries. We simulate 3 species, air, plume gas from the Space Shuttle Main Engines (SSME), also known as the R-25 engines, and the plume gas from the Solid Rocket Boosters (SRBs). We will parametrize using a finest mesh spacing, dx_{fine} , which represents roughly 40 grid cells across the nozzle. The computational mesh is setup with $4dx_{\text{fine}}$ tagging on all geometry surfaces on the mobile launcher. We provide cylindrical refinement regions below each rocket nozzles gradually coarsening to allow for small scale turbulent structures to develop. The nozzles are tagged at dx_{fine} , followed by $2dx_{\text{fine}}$, $4dx_{\text{fine}}$, and $8dx_{\text{fine}}$ in concentric cones. For cases where the vehicle is significantly further away from the pad (9, 11, and 15 seconds), the $4dx_{\text{fine}}$ plume cones do not touch the zero deck. For the 7 second cases and earlier, the $4dx_{\text{fine}}$ mesh touches the zero deck and is used for all water inlet conditions and areas where the water's interaction with the launchpad surfaces is important.

The wet simulation was performed using the LAVA Cartesian code using a 5th order accurate weighted compact nonlinear scheme (WCNS) convective flux, first order Forward Euler (RK1) time integration and an immersed-boundary method (IBM) for the geometries. We simulate 5 species, liquid water, water vapor, air, and plume gas from the SSMEs and SRBs. The hyperbolic solver is coupled with a thermal relaxation model followed by a phase transition model which simulates the evaporation and condensation of liquid water and water vapor.

All simulations are run with a ramping on the nozzle boundary conditions of 0.01 seconds. This is to reduce the severity of the shock wave and ensure numerical

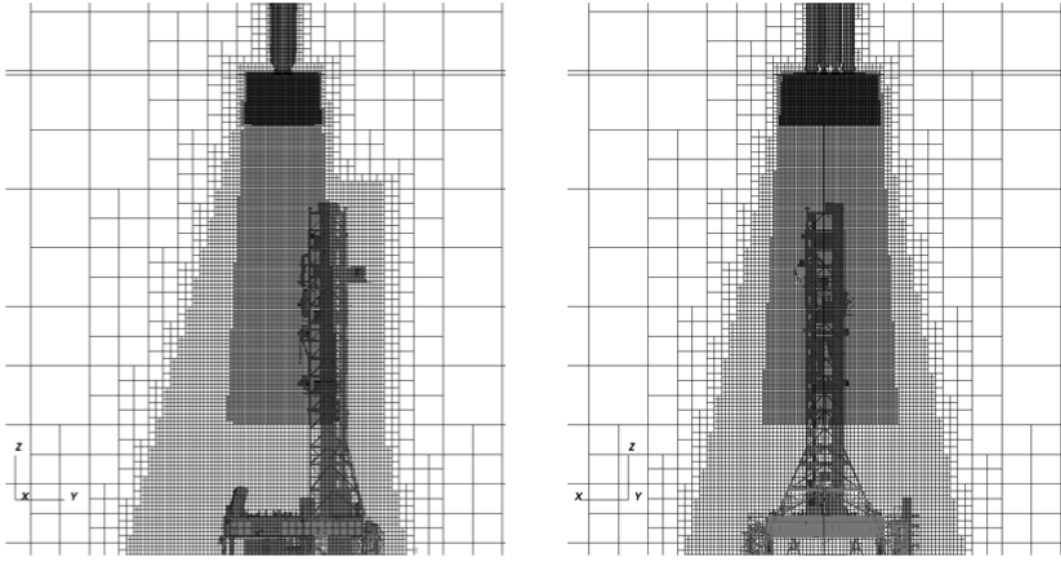


Figure 7.4: Slice of the mesh of the Artemis II Max Drift 9 second case where boxes are plotted, each box is 8 cells wide in each spatial dimension. Left: +x view looking in the west direction. Right: +y view looking in the south direction. The mobile launcher is oriented such that the tower is on the north side of the pad. In this case, you can see the $8dx_{\text{fine}}$ mesh as the outer cone and surrounding the tower structure. Slightly inside of this is the $4dx_{\text{fine}}$ mesh that extends about 2/3rds of the way down the tower. Inside this, there is the $2dx_{\text{fine}}$ mesh cone at the exit of the nozzles. The dx_{fine} mesh is not visible, but is on the outlet of the nozzles.

stability. We run these simulations for 4 seconds of physical time allowing the initial 2 seconds of transient behavior to flush out and capture the statistically stationary behavior of the plume impact on the structure. For the earlier in time simulations, where the vehicle is lofted closer to the pad, this transient takes less time to flush out and for the higher trajectory points, this transient behavior can take even longer than 2 seconds of simulation time. In Figure 7.5 we can clearly see the last 2 seconds have

Traj time (s)	Flight	Cells (M)	Vehicle Elevation (m)	Wall Clock	CPU Hours
3	Artemis I	567.2	20	4 Days	1843200
	Artemis II MD	541.2	18.5	4 Days	1843200
5	Artemis I	601.3	54	4 Days	1843200
	Artemis II MD	560.7	48	4 Days	1843200
7	Artemis I	593.1	108	4 Days	1843200
	Artemis II MD	568.3	94	4 Days	1843200
	Artemis I Wet	628.0	108	18 Days	9953280
7.76	Artemis I	605.3	130	4 Days	1843200
9	Artemis I	614.0	182	4 Days	1843200
	Artemis II MD	603.3	157	4 Days	1843200
11	Artemis I	683.7	278	4 Days	1843200
	Artemis II MD	647.4	239	4 Days	1843200
15	Artemis I	907.4	532	4 Days	1843200
	Artemis II MD	839.9	458	4 Days	1843200

Table 7.1: Summary of the suite of simulations by trajectory time with cell count in millions, vehicle elevation, wall clock time and cpu hours. All the dry simulations were run on 150 AMD Rome nodes each with 128 cores. The wet simulation was run on 180 Rome nodes. The total cost of this suite of simulations was ≈ 34 million core hours.

produced a very consistent mean with nominal fluctuations around it. We use various sensor locations as well as integrated loads on components of the launchpad to judge if we have reached a statistically stationary point. From that point, we require at least 2 seconds of simulation time to produce good statistics and have enough data for frequency analysis.

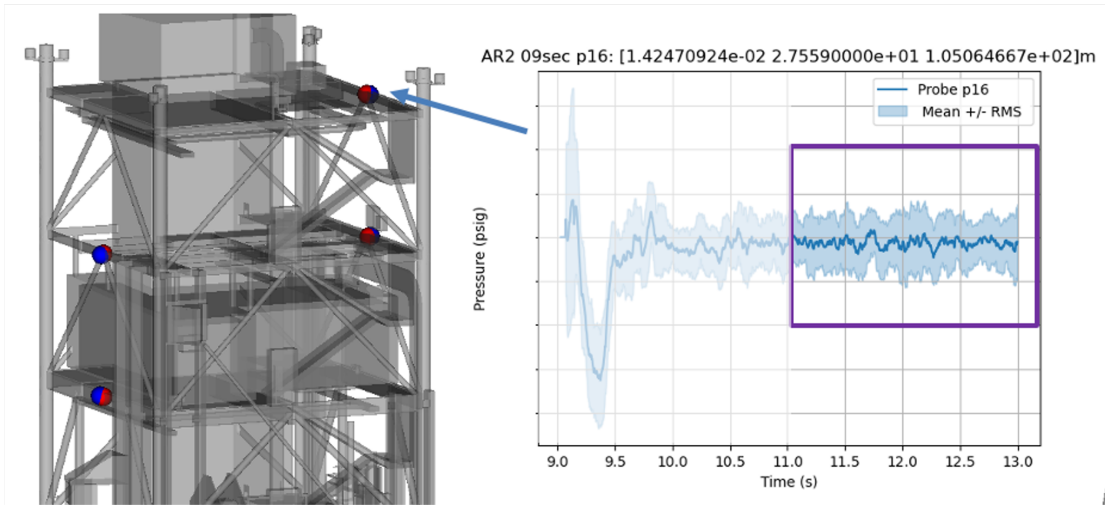


Figure 7.5: Pressure signal from a point probe located on top of the tower for Artemis II Max Drift 9 seconds. At the start of the simulation ($t = 9$ sec), we can see the transient behavior as the pressure oscillates. Eventually, the signal becomes statistically stationary and we do analysis on the final 2 seconds of the signal indicated by the purple box, throwing away the first 2 seconds. The red points are the actual sensor locations and the blue ones are the nearest point on the triangulation of the geometry.

7.0.5 Integrated Loads

One of the main areas of interest by KSC is the forces on specific areas of the launchpad. We analysed this in two separate ways, the first was on the triangulation of the mobile launcher. Using solution files, we are able to extract the full statistical information, mean, max, min, and standard deviation, on each triangle. This results in high resolution plots where engineers can see exactly where the hot spots are. This analysis is shown in Figures 7.6, 7.7, 7.8. We can see that there are areas where water actually increases the pressure forces and fluctuations compared to the dry case. This

phenomenon occurs because the plume has significantly more momentum than the water, causing it to sweep the water away in its direction of travel. Although the water counteracts some of the plume's force, it is quickly redirected in the direction of the plume. When this water mixture with greater momentum collides with structures, it results in increased pressure, than if just the plume were impacting it.

The second main way we analyse pressure forces is by integrating the force over a specific component. This is done by "coloring" the triangulation, where we can pick out specific parts of interest, where a collection of triangles colored a certain way represents a component. For example, the elevator door shafts are componentized separately, so we can analyze the entire face as a whole. Other notable components are the rainbirds, camera, tower structure, etc. In Figure 7.9 we show the average total pressure force over the last 2 seconds on each of our components. Here we can see, water provided a damping force, by decreasing the total pressure that most components experience. Although, there are some outliers, most notably the elevator shafts, which all have an increase in pressure from the wet simulation. We believe this is because of the north center rainbird's water all gets redirected backwards impacting into the elevator structure. This this could be one of the mechanisms that caused this elevator shaft to break in the first place. It's important to note, that while water may not be doing a perfect job of decreasing pressure forces everywhere, it will decrease the temperature of the deck and launch pad structures significantly due to the high specific heat of water.

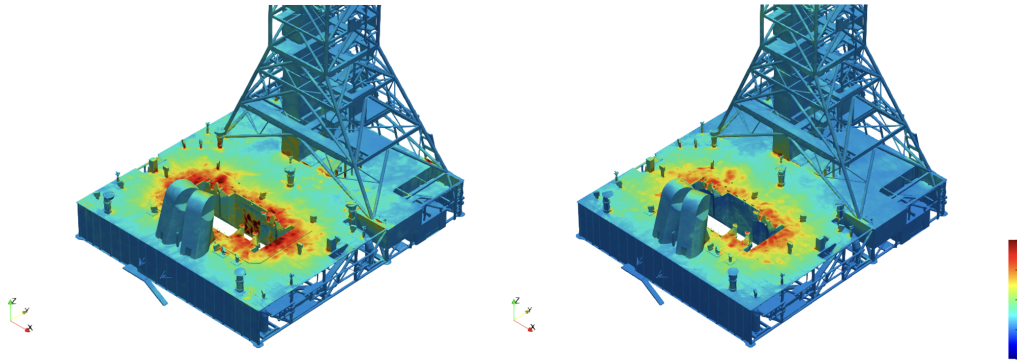


Figure 7.6: Here the maximum pressure experienced over the last two seconds of simulation is plotted for the Artemis I 07 second simulations. This is to compare the pressure loading for wet vs dry to see how the water systems play a role. These are plotted using the same color bar. In the wet simulation, we see significantly higher pressures on the deck surrounding the exhaust hole, in the exhaust hole, as well as some hot spots. Left: Wet Artemis I 07 sec, Right: Dry Artemis I 07 sec.

7.0.6 Frequency Analysis of Sensors

The other main purpose of the water is sound suppression. As discussed in the introduction of this section, water has been shown and used to successfully reduce the loudness of many rocket launches. In this section, we will compare the loudness in decibels (dB) of sensor data from the Artemis I launch, the wet simulation, and the dry simulation. We do this by computing the Power Spectral Density (PSD) of the last 2 seconds of the raw pressure signal. We do this using the Welch method with 1 second segments with 50% overlap and a Hann window to mitigate spectral leakage. Spectral leakage is a phenomena that happens when taking the PSD over non periodic

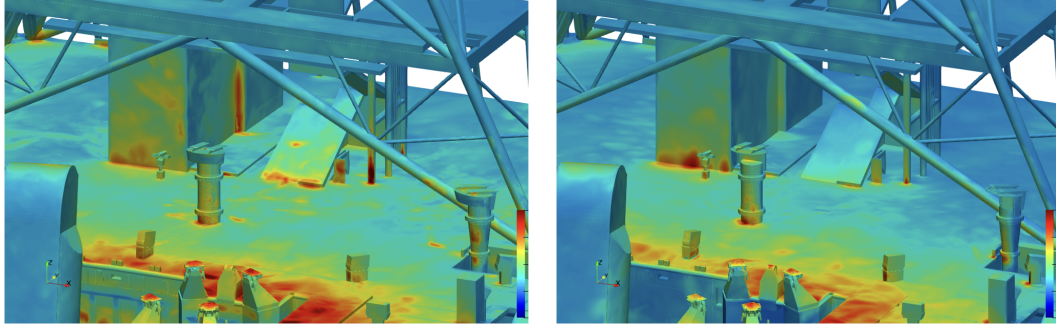


Figure 7.7: Here the maximum pressure experienced over the last two seconds of simulation is plotted for the Artemis I 07 second simulations. This is to compare the pressure loading for wet vs dry to see how the water systems play a role. This is the same visualization and data from 7.6 just zoomed in to see the elevator shaft that was damaged during launch. Here we clearly see some hot spots in the wet simulation that are not there in the dry. Left: Wet Artemis I 07 sec, Right: Dry Artemis I 07 sec.

data. When cutting into segments, a signal can become non-periodic and discontinuous leading to extra high frequency data that wasn't there without the cut. The 50% overlap allows us to retain more of the low frequency content. With this processing, we will get a frequency vs Sound Pressure Level (SPL) plot where the SPL came from the PSD. This calculation can be expressed as

$$\text{PSD}(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) \cdot w(n) \cdot e^{-j2\pi f n/N} \right|^2 \quad (7.4)$$

where $\text{PSD}(f)$ is the Power Spectral Density at frequency f , N is the number of points in each segment, $x(n)$ is the raw pressure signal at point n , $w(n)$ is the Hann window function at point n , j is the imaginary unit, and f is the frequency.

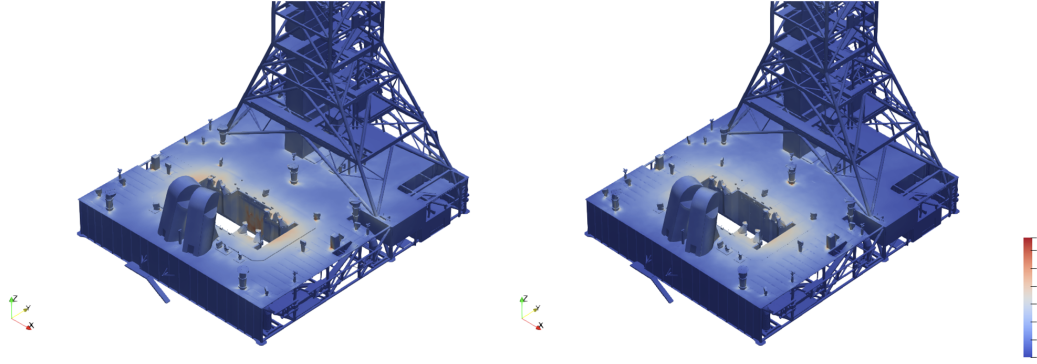


Figure 7.8: Here the standard deviation of pressure experienced over the last two seconds of simulation is plotted for the Artemis I 07 second simulations. We see a higher standard deviation indicating higher pressure fluctuations. Left: Wet Artemis I 07 sec, Right: Dry Artemis I 07 sec.

From this, we can compute an Overall Sound Pressure Level (OASPL) (dB) that represents the total noise from the system with

$$\text{OASPL} = 20 * \log_{10} \left(\sqrt{\sum_i^N \frac{\text{PSD}(f_i)}{p_{ref}}} \right) \quad (7.5)$$

where N represents the number of segments p_{ref} is the reference pressure usually set to $20\mu Pa$.

For the sensor data, we use the same process as above but also apply a 1 Hz high-pass filter to reduce sensor drift. Due to the placement of these sensors on a launchpad, where they are exposed to the intense conditions of a rocket launch, they experience high temperatures and exhibit thermal “drift” in their response, resulting in pressure readings. However, we still attempt to extract useful information from these sensors in the higher frequency ranges. This analysis was done on many sensors, but for

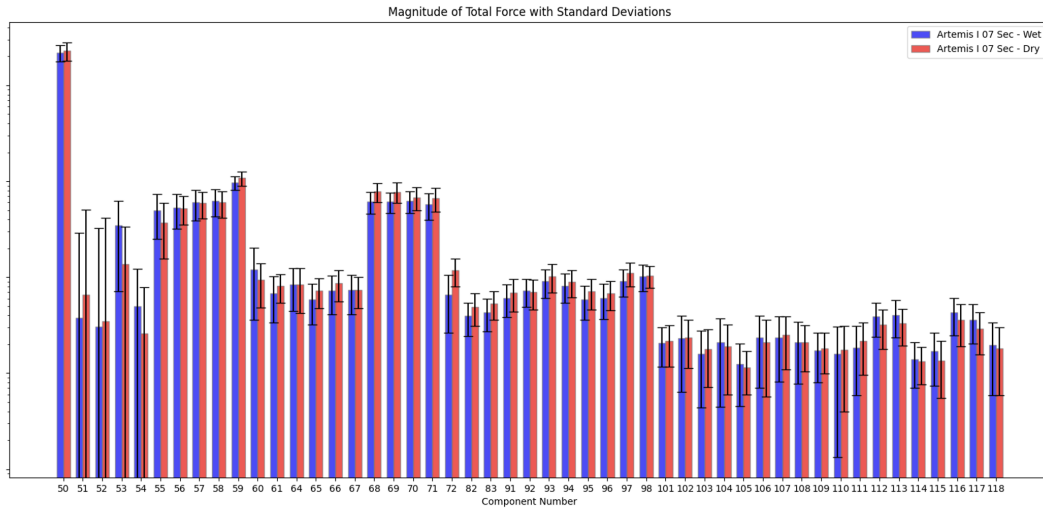


Figure 7.9: Plotted is a bar graph where we compare the average total pressure force over the last 2 seconds of simulation on each component. In blue is the wet Artemis I 07 sec simulation and in red the dry Artemis I 07 sec wet simulation. For most of the components, water provided a damping effect decreasing the total pressure. However, for some outliers, the pressure is increased.

brevity we'll present two, a sensor on the plume side of the tower and one on the far side. In general we see better agreement between the sensors and CFD on the far side, away from the plume, which is expected because these are more protected from the harsh environment. Sensor drift can be seen clearly in Figure 7.11 where the pressure signal is plotted vs time with sensor drift regions are indicated. In that region, the sensor starts to read nonphysically large values. However, taking a look at the frequency domain in Figure 7.13 we can see although there is sensor drift we still get reasonable results. In Figure 7.12 the OASPL vs time is plotted.

In Figure 7.13, there is a significant drop-off in high-frequency contributions

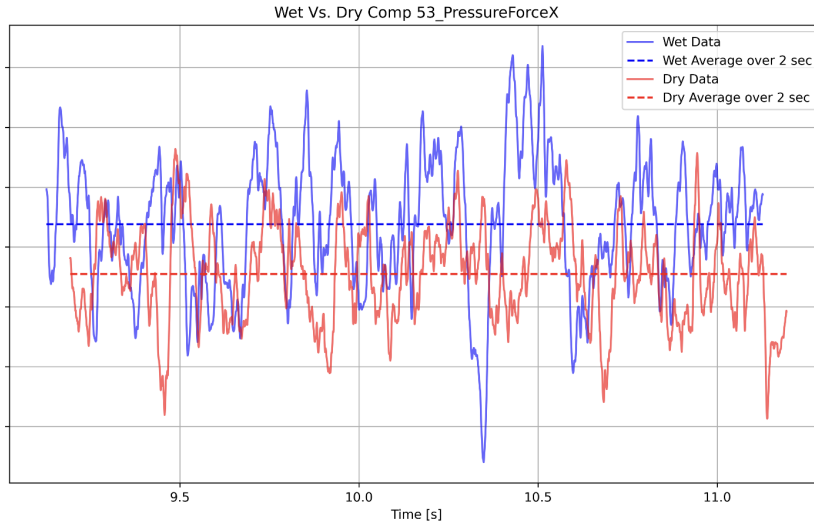


Figure 7.10: Plotted is the time series of the normal component of pressure on the east facing elevator shaft for the last 2 seconds of simulation time. This is the component that experienced the highest increase in pressure force going from the dry simulation to the wet simulation. In blue, the wet Artemis I 07 sec simulation is plotted and the dry Artemis I 07 sec simulation is plotted in red. Here we can see a significantly higher pressure sustained.

to SPL from the introduction of water, shown by the sharp decline in the green curve, which is a general trend across all the sensors. This drop-off can be attributed to several factors related to the resolution and the diffuse interface model used in the simulation.

Firstly, the resolution of the mesh impacts the highest frequency that can be accurately represented. In the regions of importance where water is ejected, the mesh has a resolution of $4dx_{\text{fine}}$, whereas in the tower, the resolution is $8dx_{\text{fine}}$. Given a grid resolution of $8dx_{\text{fine}}$, with 8 points per wave and a sound speed of 340 m/s, we have a

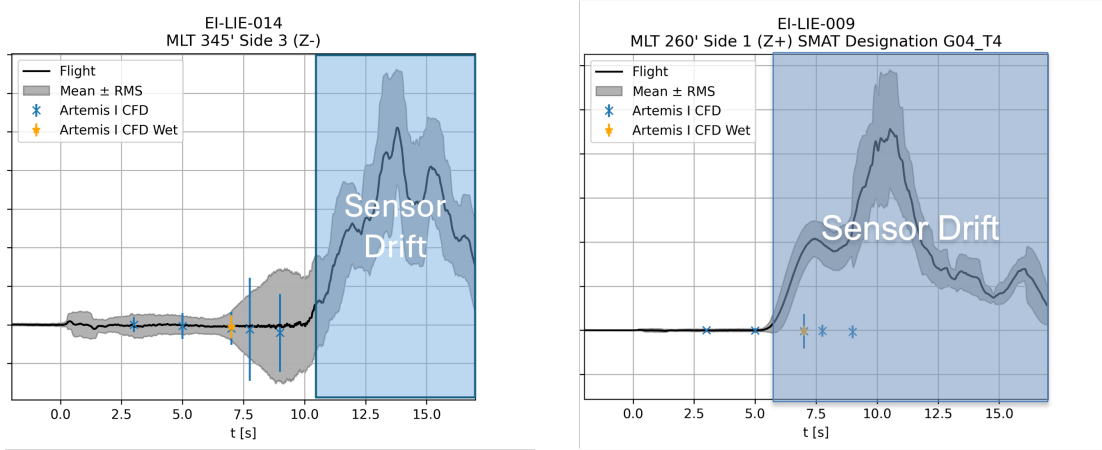


Figure 7.11: Pressure plotted as a function of time for the Artemis I sensor data in black. The blue points represent the average pressure over the last two seconds of simulation time for the Artemis I dry simulation and the orange points represents the Artemis I wet simulation. Left: A sensor on the far side of the tower. Right: A sensor on the plume side of the tower. In light blue, we label Sensor Drift when the sensor reads nonphysical pressure values.

maximum frequency of

$$f_{max} = \frac{v}{8\Delta x} = \frac{340 \text{ m/s}}{8 \times 4dx_{\text{fine}} \text{ m}} \approx 300 \text{ Hz.} \quad (7.6)$$

However, the SPL starts to drop off even before this maximum frequency in some cases. This early drop-off can be further explained by considering the effects of the diffuse interface model used in the simulation. As discussed earlier, we do not model any interface capturing nor do any kind of subgrid resolution, so each cell is a mixture of water and gas. This model inherently tends to smooth out sharp interfaces and gradients, leading to a reduction in high-frequency content.

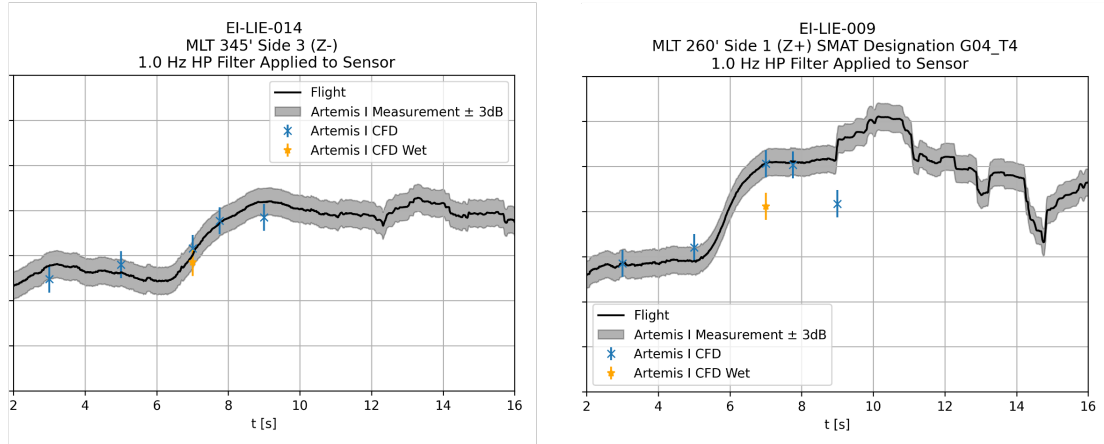


Figure 7.12: Overall Sound Pressure Level (OASPL) plotted as a function of time for the Artemis I sensor data in black. The blue points represent the OASPL value computed using the last two seconds of simulation time for the Artemis I dry simulation and the orange points represents the Artemis I wet simulation. Left: A sensor on the far side of the tower. Right: A sensor on the plume side of the tower.

7.0.7 Conclusion

In conclusion, we present a suite of simulations for pressure loads on the ML-1 for the Artemis I and Artemis II Max Drift trajectory. We validated our approach using launch sensor data, delivered data on the loads of components and studying the effect of water on both pressure loads and sound suppression. We present, to the authors knowledge, the largest (in terms of cell count) multiphase rocket launch environment simulation to date. Very few high fidelity multiphase simulations have been done prior due to the extreme computational expense coupled with complex flow physics. A couple of notable others are [9], [129], and [96]. This simulation and its data is continually being analyzed to help deliver actionable data to KSC engineers planning for Artemis II. In

particular, we showed a possible source of increased pressure due to the rainbird water, provided accurate acoustic prediction information, and pressure statistics over many quantities of interest. [9] presented a similar simulation with the rocket in take off position, where most of the plume will be directed down the trench rather than onto the launchpad itself. During the setup and execution of this case, we were also able to find many avenues of improvement for the numerical schemes used. This work is shown with more detail in [43].

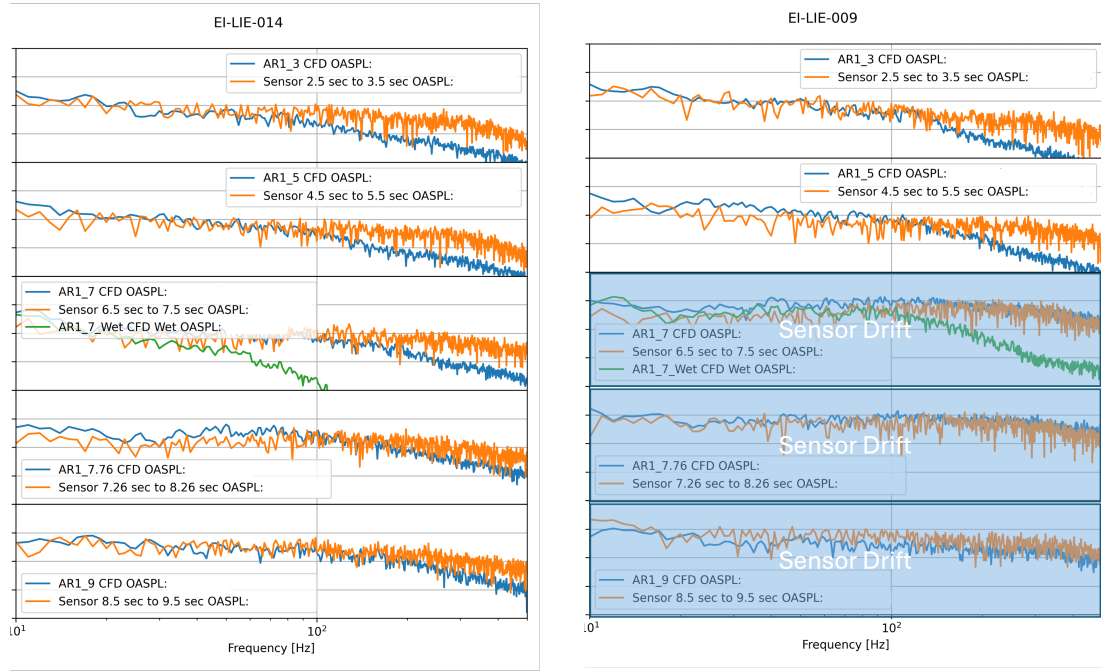


Figure 7.13: Sound Pressure Level (SPL) plotted as a function of frequency for the Artemis I sensor data in orange. The blue curves represent the SPL values computed using the last two seconds of simulation time for the Artemis I dry simulation and the green curve represents the Artemis I wet simulation. Left: A sensor on the far side of the tower. Right: A sensor on the plume side of the tower. Each vertical pane represents a different simulation, in order from top to bottom this is the Artemis I 3 second, 5 second, 7 second, 7.76 second, and 9 second simulations.

Bibliography

- [1] Moore, E.H. (1920) On the Reciprocal of the General Algebraic Matrix (Abstract).
Bulletin of the AMS, 26, 394-395. - References - Scientific Research Publishing.
- [2] Rémi Abgrall. How to Prevent Pressure Oscillations in Multicomponent Flow Calculations: A Quasi Conservative Approach. *Journal of Computational Physics*, 125(1):150–160, April 1996.
- [3] Rémi Abgrall and Smadar Karni. Computations of Compressible Multifluids. *Journal of Computational Physics*, 169(2):594–623, May 2001.
- [4] Grégoire Allaire, Sébastien Clerc, and Samuel Kokh. A Five-Equation Model for the Simulation of Interfaces between Compressible Fluids. *Journal of Computational Physics*, 181(2):577–616, September 2002.
- [5] Grégoire Allaire, Sébastien Clerc, and Samuel Kokh. A Five-Equation Model for the Simulation of Interfaces between Compressible Fluids. *Journal of Computational Physics*, 181(2):577–616, September 2002.
- [6] Daniel Allgood, Grady Saunders, and Lester Langford. *Reduction of Altitude Diffuser Jet Noise Using Water Injection*.

- [7] John David Anderson. *Computational fluid dynamics: the basics with applications*. McGraw-Hill, 1995.
- [8] Jordan B. Angel, Scott Neuhoﬀ, Man Long Wong, Michael F. Barad, and Cetin C. Kiris. *Predicting SLS Launch Environment using a Novel Multiphase Formulation*. 2023.
- [9] Jordan B. Angel, Scott Neuhoﬀ, Man Long Wong, Michael F. Barad, and Cetin C. Kiris. Predicting sls launch environment using a novel multiphase formulation. In *AIAA SciTech Forum and Exposition*, National Harbor, MD, USA, January 23–27 2023. American Institute of Aeronautics and Astronautics, Ames Research Center. NASA Document ID: 20220018985.
- [10] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM, 1998.
- [11] Uri M Ascher, Steven J Ruuth, and Brian TR Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.
- [12] Dinshaw S. Balsara, Sudip Garain, and Chi Wang Shu. An efficient class of WENO schemes with adaptive order. *Journal of Computational Physics*, 326:780–804, 2016.
- [13] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

- [14] Michael Barad, Emre Sozer, Jeffrey Housman, Jared Duensing, Alexander Aghili, Josh Anibal, Chase Ashby, Austin Bell, Francois Cadieux, Timothy Chau, Chris DeGrendele, Ercan Dumlupinar, Luis Fernandes, James Jensen, James Koch, Brandon Lowe, Nguyen Ly, Daniel Maldonado, Olivia Martin, Scott Neuhoﬀ, David Craig Penner, Kiran Ravikumar, Bram Rodgers, Victor Sousa, and Gerrit-Daniel Stich. Launch ascent and vehicle aerodynamics (lava): A NASA mission-critical CFD solution—gearing up for public release. In *Advanced Modeling and Simulation (AMS) Seminars*, June 2025. Presentation.
- [15] Michael Barad, Emre Sozer, Jeffrey Housman, James Jensen, Francois Cadieux, Timothy Sandstrom, and Cetin Kiris. High-resolution launch environment simulations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC 2018)*, Moﬀett Field, CA, United States; Beltsville, MD, United States, 2018. NASA Ames Research Center; Science and Technology Corp.; InuTeq, LLC, IEEE Computer Society; Association for Computing Machinery. Public Use Permitted.
- [16] Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owahdi. Kernel methods for solving diﬀerential equations: Analysis and applications. *Journal of Computational Physics*, 493:112442, 2023.
- [17] C. Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007.
- [18] Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

- [19] Sebastiano Boscarino, Lorenzo Pareschi, and Giovanni Russo. Implicit-explicit Runge-Kutta schemes for hyperbolic systems and kinetic equations in the diffusion limit. *SIAM Journal on Scientific Computing*, 35(1):A22–A51, 2016.
- [20] Rémi Bourgeois and Dongwook Lee. Gp-mood: A positivity-preserving high-order finite volume method for hyperbolic conservation laws. *Journal of Computational Physics*, 471:111603, 2022.
- [21] Rémi Bourgeois and Dongwook Lee. Gp-mood: A positive-preserving high-order finite volume method for hyperbolic conservation laws, 2021.
- [22] Alexandre Bourriaud, Raphaël Loubère, and Rodolphe Turpault. A priori neural networks versus a posteriori mood loop: A high accurate 1d fv scheme testing bed. *Journal of Scientific Computing*, 84(2):1–36, 2020.
- [23] Jonathan Boustani, Francois Cadieux, Gaetan K. Kenway, Michael F. Barad, Cetin C. Kiris, and Christoph Brehm. Fluid-structure interaction simulations of the aspire sr01 supersonic parachute flight test. *Aerospace Science and Technology*, 126:107596, 2022.
- [24] Bradley Boyd and Dorrin Jarrahbashi. A diffuse-interface method for reducing spurious pressure oscillations in multicomponent transcritical flow simulations. *Computers & Fluids*, 222:104924, May 2021.
- [25] Sofiane Brahim-Belhouari and Amine Bermak. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4):705–712, 2004.

- [26] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.
- [27] Pawel Buchmüller and Christiane Helzel. Improved accuracy of high-order WENO finite volume methods on Cartesian grids. *Journal of Scientific Computing*, 61(2):343–368, 2014.
- [28] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.
- [29] Ashesh Chattopadhyay, Ebrahim Nabizadeh, and Pedram Hassanzadeh. Analog forecasting of extreme-causing weather patterns using deep learning. *Journal of Advances in Modeling Earth Systems*, 12(2):e2019MS001958, 2020.
- [30] Ashesh Chattopadhyay, Jaideep Pathak, Ebrahim Nabizadeh, Wahid Bhimji, and Pedram Hassanzadeh. Long-term stability and generalization of observationally-constrained stochastic data-driven models for geophysical turbulence. *Environmental Data Science*, 2:e1, 2023.
- [31] Jiawei Chen, Jonathan Viquerat, and Elie Hachem. Multi-resolution partial differential equations preserved learning framework for spatiotemporal dynamics. *Communications Physics*, 7(1):1–12, 2024.
- [32] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

- [33] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Solving and learning nonlinear PDEs with Gaussian processes. *Journal of Computational Physics*, 447:110668, 2021.
- [34] Alexandre Chiapolino, Pierre Boivin, and Richard Saurel. A simple and fast phase transition relaxation solver for compressible multicomponent two-phase flows. *Computers Fluids*, 150:31–45, 06 2017.
- [35] Eric J. Ching, Ryan F. Johnson, and Andrew D. Kercher. Conservative, pressure-equilibrium-preserving discontinuous galerkin method for compressible, multicomponent flows, 2025.
- [36] Stéphane Clain, Steven Diot, and Raphaël Loubère. A high-order finite volume method for systems of conservation laws—multi-dimensional optimal order detection (MOOD). *Journal of computational Physics*, 230(10):4028–4050, 2011.
- [37] Stéphane Clain, S. Diot, and Raphaël Loubère. A high-order finite volume method for systems of conservation laws—multidimensional optimal order detection (mood). *Journal of Computational Physics*, 230:4028–4050, 06 2013.
- [38] Jon Cockayne, Chris J Oates, Tim J Sullivan, and Mark Girolami. Bayesian probabilistic numerical methods. *SIAM Review*, 61(4):756–789, 2019.
- [39] Sidafa Conde, Sigal Gottlieb, Zachary J Grant, and John N Shadid. Implicit-explicit WENO scheme for the equilibrium dispersive model of chromatography. *Applied Numerical Mathematics*, 123:22–42, 2017.
- [40] Giorgio Corani, Alessio Benavoli, and Marco Zaffalon. Time series forecasting

- with gaussian processes needs priors. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 103–117. Springer, 2021.
- [41] Richard Courant, Kurt Friedrichs, and Hans Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1):32–74, 1928.
- [42] Sourish Das, Sasanka Roy, and Rajiv Sambasivan. Fast gaussian process regression for big data. *Big data research*, 14:12–26, 2018.
- [43] Chris DeGrendele, Scott Neuhoﬀ, Michael Barad, James Jensen, Francois Cadieux, Emre Sozer, and Jared Duensing. Analyzing the Artemis I launch environment with multiphase CFD. In *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC24)*, November 2024. NASA @ SC24 presentation.
- [44] Christopher DeGrendele and Dongwook Lee. Gp-recipe: Gaussian process approximation to linear operations in numerical methods. *arXiv preprint arXiv:2506.03471*, 2025.
- [45] Christopher DeGrendele, Nguyen Ly, Francois Cadieux, Michael Barad, Dongwook Lee, and Jared Duensing. On the construction of high-order and exact pressure equilibrium schemes for arbitrary equations of state, 2025.
- [46] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

- [47] Steven Diot. *La méthode MOOD Multi-dimensional Optimal Order Detection: la première approche a posteriori aux méthodes volumes finis d'ordre très élevé*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2012.
- [48] Steven Diot, Stéphane Clain, and Raphaël Loubère. Improved detection criteria for the multi-dimensional optimal order detection (MOOD) on unstructured meshes with very high-order polynomials. *Computers & Fluids*, 64:43–63, 2012.
- [49] Steven Diot, Raphaël Loubère, and Stéphane Clain. The multidimensional optimal order detection method in the three-dimensional case: very high-order finite volume method for hyperbolic systems. *International Journal for Numerical Methods in Fluids*, 73(4):362–392, 2013.
- [50] Gregory E Fasshauer and Michael J McCourt. *Kernel-based approximation methods using Matlab*, volume 19. World Scientific Publishing Company, 2015.
- [51] Charles Fefferman. Existence and smoothness of the navier-stokes equation. <https://www.claymath.org/millennium-problems/navier-stokes-equation>, 2006.
- [52] Daniel Foreman-Mackey, Eric Agol, Sivaram Ambikasaran, and Ruth Angus. Fast and scalable gaussian process modeling with applications to astronomical time series. *The Astronomical Journal*, 154(6):220, 2017.
- [53] Bengt Fornberg. Generation of Finite Difference Formulas on Arbitrarily Spaced Grids. *Mathematics of Computation*, 51(184):699–706, 1988. Publisher: American Mathematical Society.

- [54] Jason Frank, Willem Hundsdorfer, and Jan G Verwer. Order results for implicit Runge-Kutta methods applied to differential algebraic systems. *SIAM Journal on Numerical Analysis*, 34(6):2318–2337, 1997.
- [55] Yuji Fujiwara, Yoshiharu Tamaki, and Soshi Kawai. Fully conservative and pressure-equilibrium preserving scheme for compressible multi-component flows. *Journal of Computational Physics*, 478:111973, April 2023.
- [56] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. *arXiv preprint arXiv:1808.05587*, 2018.
- [57] Francis X Giraldo, James F Kelly, and Emil M Constantinescu. Implicit-explicit formulations of a three-dimensional nonhydrostatic unified model of the atmosphere (NUMA). *SIAM Journal on Scientific Computing*, 35(5):B1162–B1194, 2013.
- [58] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing runge-kutta schemes. *Mathematics of computation*, 67(221):73–85, 1998.
- [59] Thore Graepel. Solving noisy linear operator equations by Gaussian processes: Application to ordinary and partial differential equations. In *Proceedings of the 20th International Conference on Machine Learning*, pages 234–241, 2003.
- [60] Ernst Hairer and Gerhard Wanner. *Solving ordinary differential equations II: Stiff and differential-algebraic problems*. Springer, 1996.
- [61] Jianan Han, Xiao-Ping Zhang, and Fang Wang. Gaussian process regression

- stochastic volatility model for financial time series. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1015–1028, 2016.
- [62] F. H. Harlow and A. A. Amsden. Fluid dynamics. a lasl monograph. Technical Report LA-4700, Los Alamos Scientific Lab., N. Mex., 1971. NTIS Issue Number 197201, Contract Number W-7405-ENG-36.
- [63] Andrew J. Herron, Michael J. Hays, Andrew M. Smith, and Matthew J. Casiano. Sls ignition overpressure-sound suppression system performance evaluated against historical configurations. In *AIAA SciTech Forum and Exposition*, Orlando, FL, United States, January 2024. American Institute of Aeronautics and Astronautics. NASA Technical Report ID: 20230016919.
- [64] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. 2008.
- [65] Janice Houston, Douglas Counter, and Clothilde Giacomoni. Sls scale model acoustic test liftoff results and comparisons. In *Aerospace Testing Seminar*, Huntsville, AL, United States, 2015. NASA Marshall Space Flight Center, Aerospace Corp., Institute of Environmental Sciences and Technology. Public. Work of the US Gov. Public Use Permitted.
- [66] Mamoru Ishii. Two-fluid model for two-phase flow. *Multiphase Science and Technology*, 5(1-4):1–63, 1990.
- [67] Giuseppe Izzo and Zdzislaw Jackiewicz. Highly stable implicit-explicit Runge-Kutta methods. *Applied Numerical Mathematics*, 113:71–92, 2017.

- [68] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.
- [69] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of computational physics*, 126(1):202–228, 1996.
- [70] Jim Cawley. Teams assess mobile launcher and pad after successful artemis i launch. <https://www.nasa.gov/humans-in-space/teams-assess-mobile-launcher-and-pad-after-successful-artemis-i-launch/>, November 2022. Accessed: 2023-04-25.
- [71] Smadar Karni. Hybrid Multifluid Algorithms. *SIAM Journal on Scientific Computing*, 17(5):1019–1039, September 1996. Publisher: Society for Industrial and Applied Mathematics.
- [72] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.
- [73] Christopher A Kennedy and Mark H Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44(1-2):139–181, 2003.
- [74] Cetin C. Kiris, Jeffrey A. Housman, Michael F. Barad, Christoph Brehm, Emre Sozer, and Shayan Moini-Yekta. Computational framework for launch, ascent,

- and vehicle aerodynamics (lava). *Aerospace Science and Technology*, 55:189–219, 2016.
- [75] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning-accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.
- [76] Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation, 2024.
- [77] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.
- [78] Malte Kuss and Carl Rasmussen. Gaussian processes in reinforcement learning. *Advances in neural information processing systems*, 16, 2003.
- [79] W. Kutta. *Beitrag zur näherungsweise Integration totaler Differentialgleichungen*. Teubner, 1901.
- [80] Yuichi Kuya and Soshi Kawai. High-order accurate kinetic-energy and entropy preserving (KEEP) schemes on curvilinear grids. *Journal of Computational Physics*, 442:110482, October 2021.
- [81] Yuichi Kuya, Kosuke Totani, and Soshi Kawai. Kinetic energy and entropy preserving schemes for compressible flows by split convective forms. *Journal of Computational Physics*, 375:823–853, December 2018.

- [82] Dongwook Lee and Rémi Bourgeois. Gp-mood: a positivity-preserving high-order finite volume method for hyperbolic conservation laws. *Proceedings of the International Astronomical Union*, 16(S362):373–379, 2020.
- [83] Dongwook Lee, Adam Reyes, Carlo Graziani, and Petros Tzeferacos. New high-order methods using gaussian processes for computational fluid dynamics simulations. In *Journal of Physics: Conference Series*, volume 837, page 012018. IOP Publishing, 2017.
- [84] Dongwook Lee, Henry Waterhouse, and Christopher DeGrendele. A Gaussian process method with a neural network for compressible shock capturing. *Journal of Physics: Conference Series*, 2997(1):012018, April 2025. Publisher: IOP Publishing.
- [85] Youngjun Lee, Dongwook Lee, and Adam Reyes. A recursive system-free single-step temporal discretization method for finite difference methods. *Journal of Computational Physics: X*, 12:100098, 2021.
- [86] Sanjiva K Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992.
- [87] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.
- [88] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.
- [89] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar.

- Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24:1–26, 2023.
- [90] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.
- [91] Phillip Lippe, Bastiaan S Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. *Advances in Neural Information Processing Systems*, 37, 2024.
- [92] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.
- [93] Xiaofei Liu, Wei Chen, and Hong Xiao. Learning to solve PDEs with finite volume-informed neural networks in a data-free approach. *Computer Methods in Applied Mechanics and Engineering*, 432:117210, 2025.
- [94] Xu-Dong Liu, R. P. Fedkiw, and S. Osher. A quasi-conservative approach to the multiphase Euler equations without spurious pressure oscillations. *SIAM J. Sci. Stat. Comput*, 1998.
- [95] Ziqi Liu, Wei Cai, and Zhi-Qin John Xu. Multi-scale deep neural network (MscaledDNN) for solving Poisson-Boltzmann equation in complex domains. *Communications in Computational Physics*, 28(5):1970–2001, 2020.

- [96] Chenyu Lu, Zhitan Zhou, Yue Shi, Yiyang Bao, and Guigao Le. Numerical simulations of water spray on launch pad during rocket launching. *Journal of Spacecraft and Rockets*, 58(2):566–574, 2021.
- [97] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.
- [98] Caroline Parsons Lubert. Sixty years of launch vehicle acoustics. *Proceedings of Meetings on Acoustics*, 31(1):040004, 02 2018.
- [99] Peter C. Ma, Yu Lv, and Matthias Ihme. An entropy-stable hybrid scheme for simulations of transcritical real-fluid flows. *Journal of Computational Physics*, 340:330–357, July 2017.
- [100] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.
- [101] Ian CT May and Dongwook Lee. Kfvm-weno: A high-order accurate kernel-based finite volume method for compressible hydrodynamics. *The Astrophysical Journal*, 967(1):6, 2024.
- [102] ICT May and D Lee. High-order genuinely multidimensional finite volume methods via kernel-based weno. In *Journal of Physics: Conference Series*, volume 2742, page 012025. IOP Publishing, 2024.

- [103] Peter McCorquodale and Phillip Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Communications in Applied Mathematics and Computational Science*, 6(1):1–25, 2011.
- [104] Abhishek Mishra, David Salac, and Matthew G. Knepley. On the order of accuracy for finite difference approximations of partial differential equations using stencil composition, 2023.
- [105] Rambod Mojjani, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Interpretable equation discovery for multi-scale flow physics using sparse regression and Fourier transforms. *arXiv preprint arXiv:2402.10157*, 2024.
- [106] Scott Neuhoﬀ, Chris Degrendele, Michael Barad, James Jensen, Francois Cadieux, Emre Sozer, and Jared Duensing. Launch pad rocket plume simulations supporting nasa’s artemis missions. In *SC23: International Conference for High Performance Computing, Networking, Storage, and Analysis*, Mountain View, California, United States, 2023. Ames Research Center, NASA Peer Committee. Public. Work of the US Gov. Public Use Permitted.
- [107] Jeb S. Orr, John H. Wall, Tannen S. VanZwieten, and Charles E. Hall. Space launch system ascent flight control design. In *Proceedings of the 2014 American Astronautical Society Guidance, Navigation, and Control Conference*, pages Session AAS 14–038, Breckenridge, CO, January 31 to February 5 2014. American Astronautical Society, NASA Marshall Space Flight Center.

- [108] Houman Owhadi. Bayesian numerical homogenization. *Multiscale Modeling & Simulation*, 13(3):812–828, 2015.
- [109] Lorenzo Pareschi and Giovanni Russo. Implicit-explicit Runge-Kutta methods and applications to hyperbolic systems with relaxation. *Journal of Scientific Computing*, 25(1):129–155, 2005.
- [110] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.
- [111] Marica Pelanti and Keh-Ming Shyue. A mixture-energy-consistent six-equation two-phase numerical model for fluids with interfaces, cavitation and evaporation waves. *Journal of Computational Physics*, 259:331–357, 2014.
- [112] Fabien Petitpas, Erwin Franquet, Richard Saurel, and Olivier Le Metayer. A relaxation-projection method for compressible flows. part ii: Artificial heat exchanges for multiphase shocks. *Journal of Computational Physics*, 225(2):2214–2248, 2007.
- [113] Rubén Picó, Iván Herrero-Durá, Víctor Sánchez-Morcillo, Luis J. Salmerón Contreras, and Lluís M. García-Raffi. Acoustic behavior of the vega launch pad environment. 06 2016.
- [114] Sergio Pirozzoli. Generalized conservative approximations of split convective

- derivative operators. *Journal of Computational Physics*, 229(19):7180–7190, September 2010.
- [115] Kenneth Plotkin, Louis Sutherland, and Bruce Vu. *Lift-Off Acoustics Predictions for the Ares I Launch Pad*. 2009.
- [116] Christian Prins, Sander Wahls, and Michel Verhaegen. Adaptive kernel methods for solving PDEs: Theory and implementation. *Computer Methods in Applied Mechanics and Engineering*, 399:115361, 2022.
- [117] Andrea Prosperetti and G. Tryggvason. *Computational Methods for Multiphase Flow*, pages –. Cambridge University Press, United Kingdom, 2009.
- [118] Nasim Rahaman, Baratin Aristide, Arpit Devansh, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *Proceedings of the 36th International Conference on Machine Learning*, pages 5301–5310, 2019.
- [119] Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023.
- [120] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [121] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse prob-

- lems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [122] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.
- [123] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation And Machine Learning. MIT Press, 2005.
- [124] Steve Reeves, Dongwook Lee, Adam Reyes, Carlo Graziani, and Petros Tzeferacos. An application of gaussian process modeling for high-order accurate adaptive mesh refinement prolongation. *arXiv preprint arXiv:2003.08508; Accepted for publication in CAMCoS*, 2021.
- [125] Steven I Reeves, Dongwook Lee, Anurag Singh, and Kunal Verma. A gaussian process upsampling model for improvements in optical character recognition. In *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II 15*, pages 263–274. Springer, 2020.
- [126] Adam Reyes, Dongwook Lee, Carlo Graziani, and Petros Tzeferacos. A new class of high-order methods for fluid dynamics simulations using gaussian process modeling: I. one-dimensional case. *Journal of Scientific Computing*, 76, 07 2018.
- [127] Adam Reyes, Dongwook Lee, Carlo Graziani, and Petros Tzeferacos. A new class of high-order methods for fluid dynamics simulations using gaussian process

- modeling: One-dimensional case. *Journal of Scientific Computing*, 76(1):443–480, 2018.
- [128] Adam Reyes, Dongwook Lee, Carlo Graziani, and Petros Tzeferacos. A variable high-order shock-capturing finite difference method with GP-WENO. *Journal of Computational Physics*, 381:189–217, 2019.
- [129] Travis A. Rivord and Brandon Williams. *Validation of Multiphase CFD Predictions of the SLS Launch Environment against Artemis I Flight Data*. 2024.
- [130] Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.
- [131] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
- [132] C Runge. Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, June 1895.
- [133] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364, 2020.
- [134] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph

- networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.
- [135] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of mathematical psychology*, 85:1–16, 2018.
- [136] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations*, pages 325–432. Springer, 1998.
- [137] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. *Journal of Computational Physics*, 83(1):32–78, 1989.
- [138] Marshall Smith, Douglas Craig, Nicole Herrmann, Erin Mahoney, Jonathan Krezel, Nate McIntyre, and Kandyce Goodliff. The artemis program: An overview of nasa’s activities to return humans to the moon. In *2020 IEEE Aerospace Conference*, pages 1–10, 2020.
- [139] Seth C Spiegel, HT Huynh, and James R DeBonis. A survey of the isentropic Euler vortex problem using high-order methods. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2444, 2015.
- [140] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.
- [141] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin

- Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.
- [142] H. Terashima, N. Ly, and M. Ihme. Approximately pressure-equilibrium-preserving scheme for fully conservative simulations of compressible multi-species and real-fluid interfacial flows. *Journal of Computational Physics*, 524:113701, 2025.
- [143] Hiroshi Terashima, Soshi Kawai, and Mitsuo Koshi. Consistent numerical diffusion terms for simulating compressible multicomponent flows. *Computers & Fluids*, 88:484–495, December 2013.
- [144] Eleuterio Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. 01 2009.
- [145] Lloyd N Trefethen. *Finite difference and spectral methods for ordinary and partial differential equations*. Cornell University, 1996.
- [146] Seiji Tsutsumi, Susumu Kato, Kota Fukuda, Ryoji Takaki, and Kyoichi Ui. *Effect of Deflector Shape on Acoustic Field of Launch Vehicle at Lift-Off*.
- [147] Andrew Lawson Tuma. Deep learning assisted hankel dynamic mode decomposition. Master’s thesis, San Diego State University, 2022.
- [148] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative

- PDE-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.
- [149] K. Viswanathan, P. R. Spalart, M. J. Czech, A. Garbaruk, and M. Shur. Tailored nozzles for jet plume control and noise reduction. *AIAA Journal*, 50(10):2115–2134, 2012.
- [150] Haijin Wang, Chi-Wang Shu, and Qiang Zhang. Stability and error estimates of local discontinuous Galerkin methods with implicit-explicit time-marching for advection-diffusion problems. *SIAM Journal on Numerical Analysis*, 46(6):2898–2926, 2008.
- [151] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.
- [152] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.
- [153] Christopher KI Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In *Learning in graphical models*, pages 599–621. Springer, 1998.
- [154] Man Long Wong, Jordan B. Angel, and Cetin C. Kiris. A positivity-preserving eulerian two-phase approach with thermal relaxation for compressible flows with a liquid and gases, 2022.

- [155] George Wynne, François-Xavier Briol, and Mark Girolami. Convergence guarantees for Gaussian process approximations under several observation models. *Electronic Journal of Statistics*, 15(2):5112–5159, 2021.
- [156] Xiaojie Xu and Yun Zhang. Price forecasts of ten steel products using gaussian process regressions. *Engineering Applications of Artificial Intelligence*, 126:106870, 2023.
- [157] Zhi-Qin John Xu, Lulu Zhang, and Wei Cai. On understanding and overcoming spectral biases of deep neural network learning methods for solving PDEs. *Journal of Computational Physics*, 474:111506, 2023.
- [158] Gabriel Ziegler, Gerard R Ridgway, Robert Dahnke, Christian Gaser, Alzheimer’s Disease Neuroimaging Initiative, et al. Individualized gaussian process-based prediction and detection of local and global gray matter abnormalities in elderly subjects. *NeuroImage*, 97:333–348, 2014.