Effect of Floating-Point Precision on PDE Solvers: A Scheme-Wise Study of Linear Advection-Diffusion and Viscous Burgers Equations

Nishchith C Bharadwaj (SR No. 26650), Dhruv Yadav (SR No. 26641), Rajneesh Babu (SR No. 26058)
DS 289 Numerical Solution of Differential Equations — Final Project Report, IISc

Abstract

We study how floating-point precision interacts with PDE discretization across four explicit scheme families (Scheme 1: Upwind/Godunov+FE; Scheme 2: Lax-Friedrichs/Rusanov+FE; Scheme 3: Lax-Wendroff/Richtmyer; Scheme 4: MUSCL+Rusanov+SSP-RK2) applied to linear advection-diffusion and viscous Burgers equations using FP64, FP32, and FP16 precision. Six experiments expose failure boundaries across smooth baselines, ultra-high-Re shocks, under-resolved grids, tiny-amplitude transport, long-horizon drift, and CFL-overdrive instability. Key findings: (i) FP32 tracks FP64 to machine-epsilon on Schemes 1–2 and within 5% on Schemes 3–4 in smooth baselines; (ii) FP16 gives $L_2 = O(10^{-1})$ even for sinusoidal transport; (iii) under the under-resolved $Re = 10^5$, $n_x = 128$ shock, all precisions collapse to $L_2 \approx 0.70$, showing resolution dominates; (iv) FP16 exhibits 0.30 rad phase error for $A = 10^{-6}$ waves; (v) FP16 conservation drift is $O(10^{-2})$–$O(10^{-1})$ vs. FP32's 10^{-6}; (vi) at CFL= 1.2, FP16 fails in all schemes and FP32 additionally fails in Scheme 4. FP32 is recommended as the default working precision.

I. Introduction and Motivation

Whether a PDE solver remains accurate when arithmetic precision is reduced from FP64 to FP32 or FP16 is increasingly relevant in scientific computing and machine-learning-enabled numerical workflows, where reduced precision can lower memory footprint and, on appropriate hardware, accelerate tensor operations. However, low precision may distort flux balances, steep gradients, and long-time conservation properties [1], [2].

We consider the linear advection-diffusion equation,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{1}$$

which provides a smooth benchmark with an exact solution, and the viscous Burgers equation in conservative form,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}, \tag{2}$$

where $Re = 1/\nu$ exposes the regime: low Re gives smooth dynamics; high Re produces steep shock-like structures [3], [4], [5].

The project objective is to study: (i) FP64 vs FP32 vs FP16, (ii) linear and nonlinear advection-diffusion behaviour, (iii) Reynolds-number effects in Burgers, (iv) the regimes where low precision fails, and (v) computational cost — all scheme-wise and experiment-wise so that failure can be attributed to arithmetic precision, discretization order, or a deliberately constructed stress condition. The three precision levels considered correspond to machine epsilons $\varepsilon_{\mathrm{FP64}} \approx 10^{-16}$, $\varepsilon_{\mathrm{FP32}} \approx 10^{-7}$, and $\varepsilon_{\mathrm{FP16}} \approx 10^{-3}$, spanning 10^4–10^{13} in representable granularity and directly explaining the magnitude of all errors and failure modes reported below.

II. Methodology and Problem Setup

A. Initial Conditions and Reference Solutions

All runs use a uniform periodic grid on $x \in [0, 1)$. For the linear equation, $u(x, 0) = A\sin(2\pi x)$ with $A = 1$ (baseline) or $A = 10^{-6}$ (Exp. 4); the exact solution is $u(x, t) = Ae^{-\nu(2\pi)^2 t}\sin(2\pi(x - ct))$. For Burgers, $u(x, 0) = \sin(2\pi x)$ and the reference is the Cole–Hopf semi-analytic solution on a dense $n_x = 8192$ grid.

B. Scheme-Wise Discretization

All schemes use the central second difference for diffusion: $(u_{i+1} - 2u_i + u_{i-1})/\Delta x^2$. Convective treatment differs.

1) Scheme 1: Upwind / Godunov + Forward Euler: First-order upwinding for the linear equation ($c > 0$):

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + c\frac{u_i^n - u_{i-1}^n}{\Delta x} = \nu\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}. \tag{3}$$

Godunov (exact Riemann) flux $F^G_{i+1/2}$ for Burgers; shock vs. rarefaction resolved from the Rankine–Hugoniot speed sign.

2) Scheme 2: Lax–Friedrichs / Rusanov + Forward Euler: Lax–Friedrichs flux for linear: $F^{LF}_{i+\frac{1}{2}} = \frac{1}{2}(cu_i + cu_{i+1}) - \frac{|c|}{2}(u_{i+1} - u_i)$. Rusanov (local Lax–Friedrichs) flux for Burgers:

$$F^{Rus}_{i+\frac{1}{2}} = \tfrac{1}{2}(f(u_i) + f(u_{i+1})) - \tfrac{\alpha_{i+\frac{1}{2}}}{2}(u_{i+1} - u_i), \quad \alpha_{i+\frac{1}{2}} = \max(|u_i|, |u_{i+1}|),\ f(u) = \tfrac{u^2}{2}. \tag{4}$$

3) Scheme 3: Lax–Wendroff / Richtmyer: Second-order Lax–Wendroff flux: $F^{LW}_{i+\frac{1}{2}} = \frac{c}{2}(u_i + u_{i+1}) - \frac{c^2\Delta t}{2\Delta x}(u_{i+1} - u_i)$. Richtmyer predictor–corrector for Burgers raises spatial order to two in smooth regions.

4) Scheme 4: MUSCL + Rusanov + SSP-RK2: MUSCL reconstruction with minmod-limited slope $\sigma_i = \mathrm{minmod}((u_i - u_{i-1})/\Delta x, (u_{i+1} - u_i)/\Delta x) \cdot \Delta x$:

$$u^L_{i+\frac{1}{2}} = u_i + \tfrac{1}{2}\sigma_i, \qquad u^R_{i+\frac{1}{2}} = u_{i+1} - \tfrac{1}{2}\sigma_{i+1}, \tag{5}$$

fed to Rusanov flux with $\alpha_{i+\frac{1}{2}} = \max(|u^L_{i+\frac{1}{2}}|, |u^R_{i+\frac{1}{2}}|)$. Time integration uses the two-stage SSP-RK2 (Shu–Osher) scheme [6]:

$$u^{(1)} = u^n + \Delta t\, \mathcal{L}(u^n), \qquad u^{n+1} = \tfrac{1}{2}u^n + \tfrac{1}{2}\Big(u^{(1)} + \Delta t\, \mathcal{L}(u^{(1)})\Big). \tag{6}$$

C. CFL Time-Step and Precision Modes

The combined advection–diffusion CFL step,

$$\Delta t = \frac{\mathrm{CFL}}{|c|/\Delta x + 2\nu/\Delta x^2} \text{ (linear)}, \qquad \Delta t = \frac{\mathrm{CFL}}{\|u\|_\infty/\Delta x + 2\nu/\Delta x^2} \text{ (Burgers)}, \tag{7}$$

is used by all schemes. For Burgers, a stable FP64 pilot schedule is pre-computed and replayed at FP32 and FP16, isolating precision effects from time-step differences; note that this may slightly disadvantage lower precisions, which could in principle require tighter stability bounds at reduced arithmetic granularity. Arithmetic modes are implemented via explicit PyTorch `dtypes`: `float64`, `float32`, `float16`.

D. Experiment Design

Table I lists the six experiments run identically for all four schemes.

TABLE I: Experiment catalogue — parameters common to all four schemes.

#	Name	Purpose	Grid / Re	CFL
1	Baseline regime sweep	Linear + Burgers, $Re = 10, 100, 1000$	$n_x = 1024$	0.80
2	Ultra-high-Re shock stress	Add $Re = 10^5$; convection-dominated shock	$n_x = 1024$	0.80
3	Under-resolved shock	$Re = 10^5$, coarse grid; resolution dominates	$n_x = 128$	0.80
4	Tiny-amplitude quantization	$A = 10^{-6}$; FP16 precision floor & phase error	$n_x = 1024$	0.80
5	Long-horizon drift	Extended t; accumulated conservation / TV drift	$n_x = 1024$	0.50
6	CFL-overdrive failure	CFL$= 1.2$, $Re = 10^5$; method-level instability stress	$n_x = 256$	1.20

III. Results and Discussion

A. Smooth Linear Baseline (Exp. 1)

Figure 1 shows the baseline linear profiles. FP32 is visually indistinguishable from FP64 in all schemes; FP16 visibly flattens the wave and introduces phase lag. Table II quantifies this: FP64 L_2 errors are 1.99×10^{-3} (Schemes 1–2), 3.76×10^{-6} (Scheme 3), 2.96×10^{-5} (Scheme 4). FP32 matches FP64 to machine-epsilon in Schemes 1–2; Scheme 3 shows a $\sim$5% gap (3.761 vs. 3.571×10^{-6}) from accumulated Lax–Wendroff correction rounding; Scheme 4 shows 0.44%. FP16 errors are $O(10^{-1})$ with no runtime benefit on CPU; on GPU with tensor-core acceleration and loss scaling, FP16 behaviour may differ substantially.

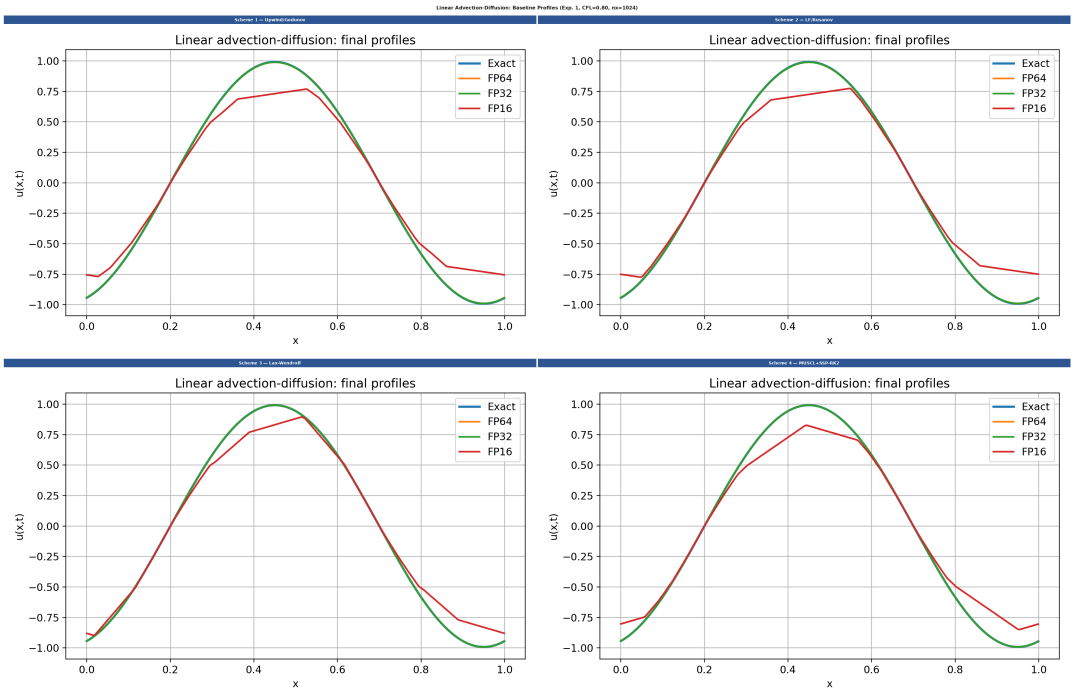

Fig. 1: Baseline linear profiles (Exp. 1, $n_x = 1024$, CFL$= 0.80$). Schemes 3–4 reduce truncation error; FP16 damps and phase-shifts the solution in all schemes.

TABLE II: Measured baseline metrics (Exp. 1). Linear L_2 vs. exact; Burgers L_2 vs. Cole–Hopf at $Re = 100$. **Red** = failed / qualitatively wrong.

Scheme	Prec.	Lin. L_2	Burg. L_2 ($Re=100$)	Lin. runtime (s)	Mem. (MB)	Stable
S1 Upwind/Godunov	FP64	1.993×10^{-3}	1.231×10^{-3}	0.034	0.0078	✓
	FP32	1.993×10^{-3}	1.230×10^{-3}	0.031	0.0039	✓
	FP16	1.418×10^{-1}	2.330×10^{-1}	0.028	0.0020	✓
S2 LF/Rusanov	FP64	1.993×10^{-3}	1.232×10^{-3}	0.059	0.0078	✓
	FP32	1.993×10^{-3}	1.231×10^{-3}	0.045	0.0039	✓
	FP16	1.410×10^{-1}	2.330×10^{-1}	0.040	0.0020	✓
S3 Lax–Wendroff	FP64	3.761×10^{-6}	1.109×10^{-5}	0.044	0.0078	✓
	FP32	3.571×10^{-6}	1.070×10^{-5}	0.042	0.0039	✓
	FP16	9.324×10^{-2}	2.330×10^{-1}	0.043	0.0020	✓
S4 MUSCL+SSP-RK2	FP64	2.955×10^{-5}	1.332×10^{-5}	0.309	0.0078	✓
	FP32	2.968×10^{-5}	1.181×10^{-5}	0.173	0.0039	✓
	FP16	1.226×10^{-1}	2.329×10^{-1}	0.158	0.0020	✓

B. Nonlinear Burgers: Reynolds-Number Dependence (Exp. 1–2)

Figure 2 shows Burgers L_2 error vs. Re. At $Re = 10$, Schemes 3 and 4 achieve $O(10^{-6})$ in FP64 (1.52×10^{-6} and 1.49×10^{-6} respectively) and $O(10^{-5})$ in FP32 (1.30×10^{-5} and 2.82×10^{-5}), while first-order Schemes 1–2 plateau near 3.8×10^{-4} for FP64. At $Re = 100$, Schemes 1–2 plateau at 1.23×10^{-3}; Schemes 3–4 remain at $O(10^{-5})$. FP64 and FP32 both converge to $L_2 \approx 0.65$ at $Re = 1000$; FP16 saturates at $L_2 \approx 0.63$–0.64, showing the grid-resolution ceiling dominates over precision once Re is large. At $Re = 10$–100, FP16 error is 0.23–0.29 — two to three orders of magnitude above FP64/FP32 — confirming FP16 is qualitatively inadequate even in smooth, well-resolved regimes.

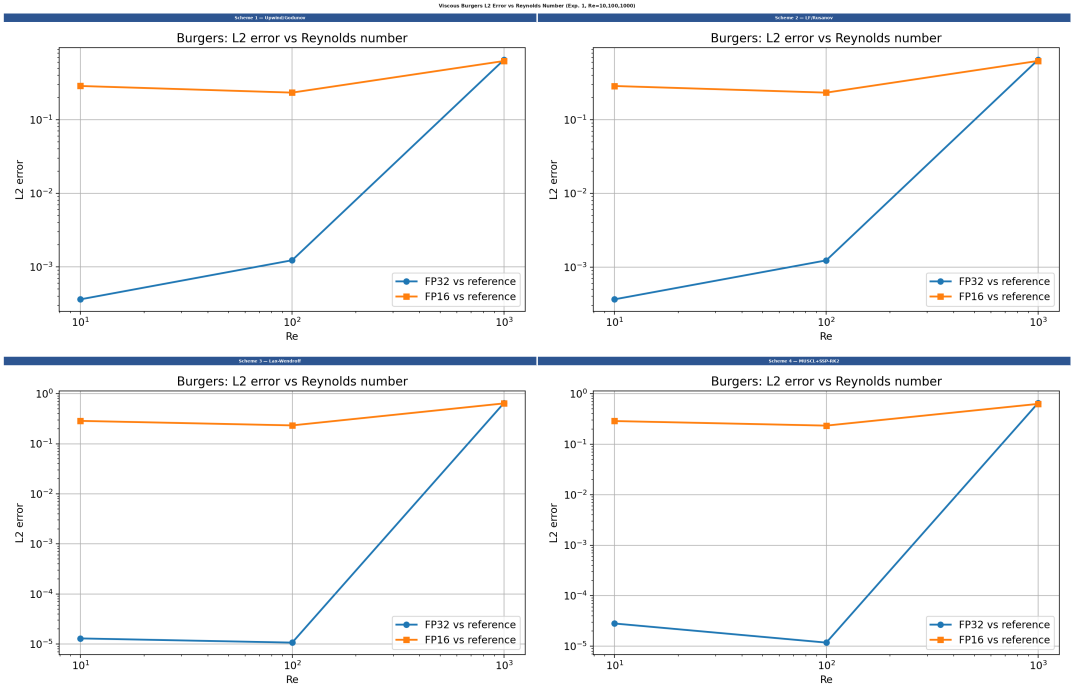

Fig. 2: Baseline Burgers L_2 error vs. Re (Exp. 1). Schemes 3–4 are substantially more accurate at low Re; at $Re = 1000$ the resolution limit causes all precisions to converge.

C. Stability CFL Sweep (Exp. 1)

Figure 3 shows the largest stable CFL from Exp. 1. For linear problems, the ceiling is CFL$= 1.0$ for Schemes 1, 2, 4 and CFL$= 1.2$ for Scheme 3 at all three precision levels — precision does not shift the boundary in smooth, well-resolved regimes. The Burgers stability picture is richer: at $Re = 1000$, FP64 tolerates CFL$= 1.4$ in Scheme 1 and CFL$= 1.4$ in Scheme 3, while FP32 caps at CFL$= 1.0$ for Schemes 1 and 2 and CFL$= 1.2$ for Scheme 3, showing that nonlinearity and reduced precision together tighten the stability margin even before it becomes catastrophic. Precision becomes decisively important only when CFL is adversarially overdriven to 1.2 (Exp. 6): FP16 diverges universally and FP32 additionally fails in Scheme 4.

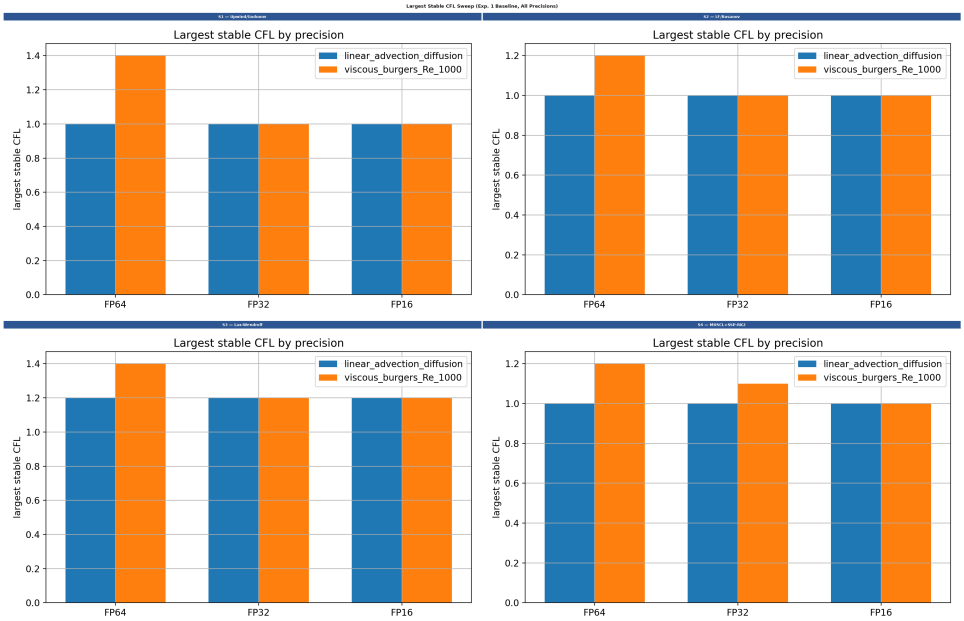

Fig. 3: Largest stable CFL (Exp. 1, linear). Scheme 3 tolerates up to 1.2; others cap at 1.0. Precision does not shift the boundary in smooth regimes.

IV. Failure Cases: When and How Precision Breaks the Solver

Table III summarises the complete failure map. The subsections below analyse each mode.

TABLE III: Precision failure summary. **Pass** = within truncation error; **Warn** = degraded but bounded; **FAIL** = qualitatively wrong or non-finite.

	Exp. 3 Under-resolved		Exp. 4 Tiny-amp	Exp. 5 Long-horizon		Exp. 6 CFL-overdrive
Scheme / Prec.	FP32	FP16	FP16 (phase)	FP32 (drift)	FP16 (drift)	FP16
S1 Upwind/Godunov	**FAIL**	**FAIL**	**FAIL** (0.30 rad)	**Pass**	**FAIL**	**FAIL** (∞)
S2 LF/Rusanov	**FAIL**	**FAIL**	**FAIL** (0.30 rad)	**Pass**	**FAIL**	**FAIL** (∞)
S3 Lax–Wendroff	**FAIL**	**FAIL**	**FAIL** (0.30 rad)	**Pass**	**FAIL**	**FAIL** (∞)
S4 MUSCL+SSP-RK2	**FAIL**	**FAIL**	**Warn** (0.056 rad)	**Pass**	**FAIL**	**FAIL** (∞)

Exp. 3: "FAIL" = resolution dominates; FP32 as bad as FP16 ($L_2 \approx 0.70$). Exp. 6 FP32: **additional FAIL in S4 only**; stable in S1–S3.

A. Failure Mode 1 — Resolution Saturation (Exp. 3)

Figure 4 shows $n_x = 128$, $Re = 10^5$ Burgers results. The per-scheme FP64 L_2 values are 0.703 (S1), 0.703 (S2), 0.707 (S3), 0.707 (S4) — all within 0.4% of one another. FP32 returns identical values to FP64 in all four schemes (differences below 10^{-5}), and FP16 differs by less than 10^{-3}. This is the sharpest possible demonstration that **FP32 and FP16 both fail equally**: once the shock is under-resolved, the discretization error completely dominates and higher arithmetic precision provides no benefit whatsoever. Investing in FP64 over FP16 changes the L_2 error by $\ll 0.1\%$ here, yet costs $4\times$ the memory.

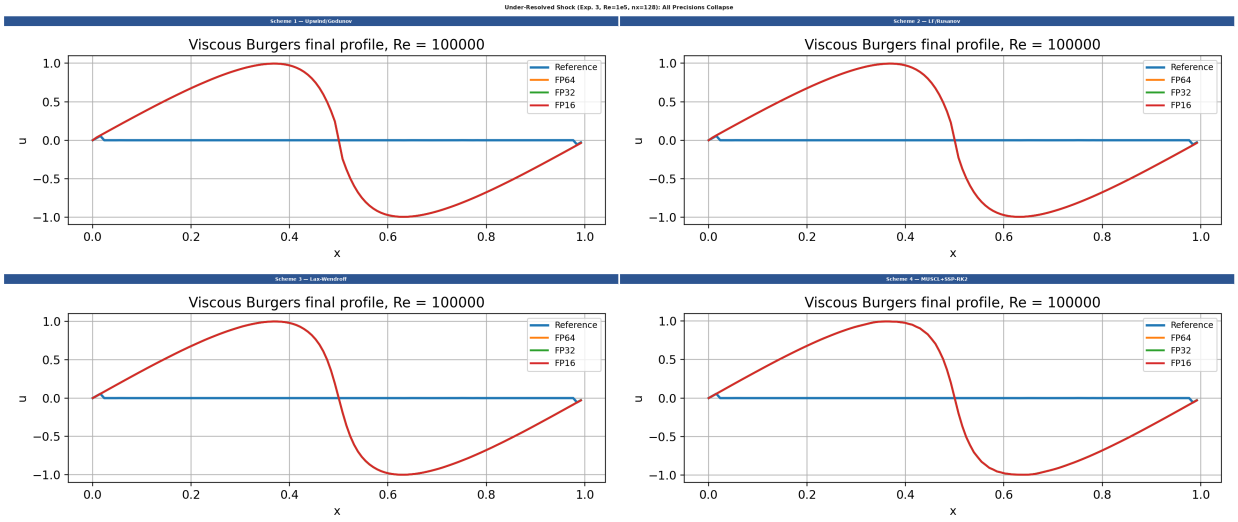

Fig. 4: Under-resolved shock (Exp. 3, $Re = 10^5$, $n_x = 128$). All schemes and precisions return $L_2 \approx 0.70$; resolution saturation makes precision irrelevant.

B. Failure Mode 2 — Precision Floor / Phase Corruption (Exp. 4)

Figure 5 shows the linear solver with $A = 10^{-6}$. FP64/FP32 track the exact solution faithfully ($L_2 \approx 10^{-9}$–10^{-12}, zero phase error). FP16 gives $L_2 \approx 8.3 \times 10^{-7}$ with **0.300 rad** phase error in Schemes 1–3, and $L_2 \approx 2.5 \times 10^{-7}$ / 0.056 rad in Scheme 4. The 10-bit FP16 mantissa cannot faithfully represent $O(10^{-6})$ wave updates, causing a growing phase offset — a pure precision-floor effect invisible in standard-amplitude tests.

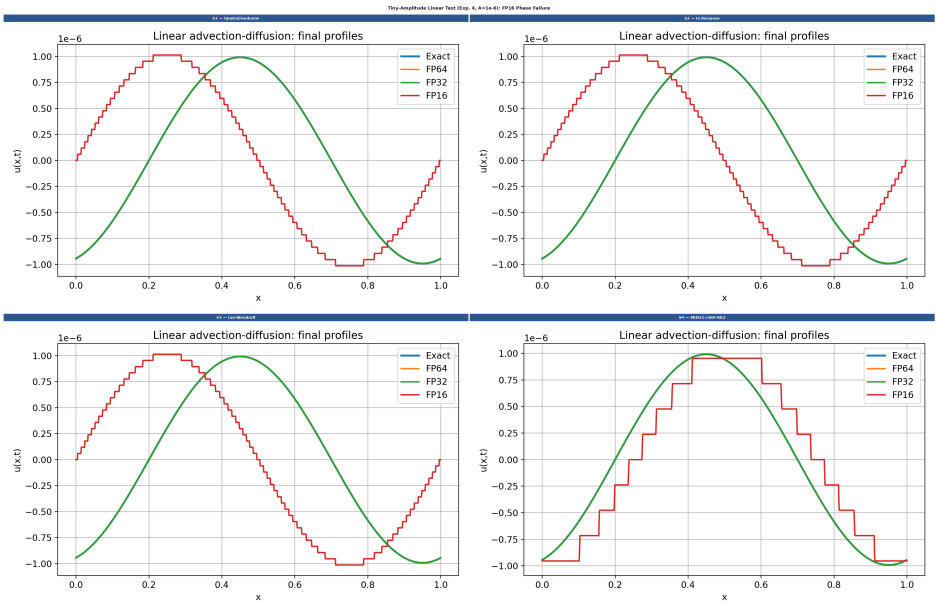

Fig. 5: Tiny-amplitude linear test (Exp. 4, $A = 10^{-6}$). FP64/FP32 reproduce the exact solution; FP16 exhibits severe phase corruption (0.30 rad, Schemes 1–3) from the precision floor.

C. Failure Mode 3 — Long-Horizon Conservation Drift (Exp. 5)

Figure 6 shows conservation-error accumulation over extended time ($t_{\text{linear}} = 1.0$, $t_{\text{Burgers}} = 0.30$, CFL$= 0.50$). Measured conservation-error sums for the linear problem are: FP64 $\approx 10^{-14}$–10^{-13}, FP32 $\approx 5 \times 10^{-7}$–9×10^{-6}, and FP16 3.8×10^{-2}–1.0×10^{-1} across all four schemes. The 10^{6}–$10^{7}\times$ gap between FP32 and FP16 arises from 10-bit mantissa rounding accumulating over thousands of steps. Total variation (TV) tells a similar story: FP64 preserves TV to within 0.04% of the initial value, FP32 to within 0.1%, whereas FP16 degrades TV by up to 16% (Scheme 4, $t = 1.0$), reflecting progressive smearing of the wave profile. FP32 with its 23-bit mantissa ($\varepsilon \approx 10^{-7}$) stays near machine-precision drift for all practical run lengths in this study.

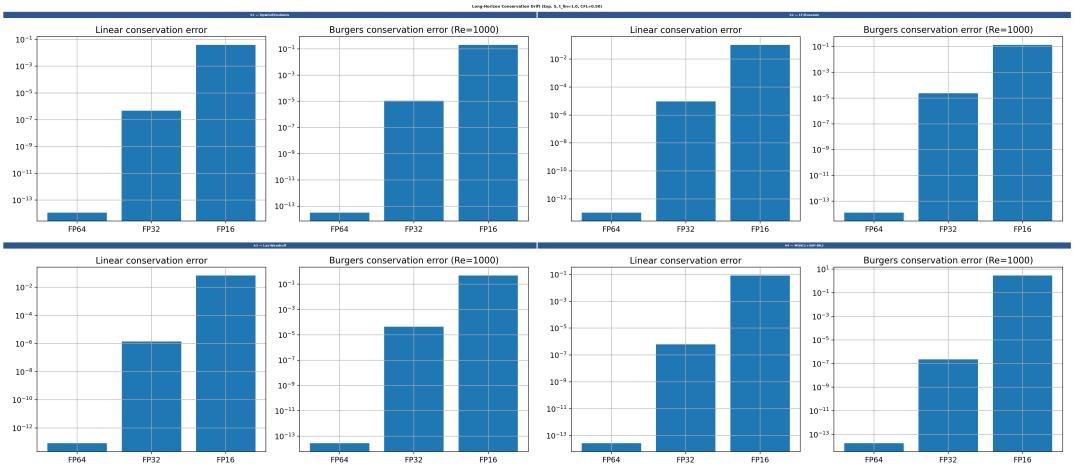

Fig. 6: Conservation drift (Exp. 5, $t_{\text{linear}} = 1.0$). FP64 at machine precision; FP32 near 10^{-6}; FP16 reaches $O(10^{-2})$–$O(10^{-1})$ — $10^{6}\times$ worse than FP32.

D. Failure Mode 4 — CFL-Overdrive Instability (Exp. 6)

Figure 7 shows Burgers at CFL$= 1.2$, $Re = 10^{5}$, $n_x = 256$. For Schemes 1–3, FP64/FP32 remain bounded ($L_2 \approx 0.706$) while FP16 diverges. For Scheme 4, both FP32 and FP16 diverge; only FP64 stays stable ($L_2 \approx 0.735$). This hierarchy — FP64 always stable $\succ$ FP32 stable in low-order schemes only $\succ$ FP16 always fails — reflects multiplicative interaction between precision and reconstruction order. MUSCL's nonlinear slope-limiting introduces discontinuous interface states; at CFL$= 1.2$ the FP32 rounding noise ($\varepsilon \approx 10^{-7}$) seeds runaway amplification near the Burgers front, while first-order schemes absorb the same perturbation.

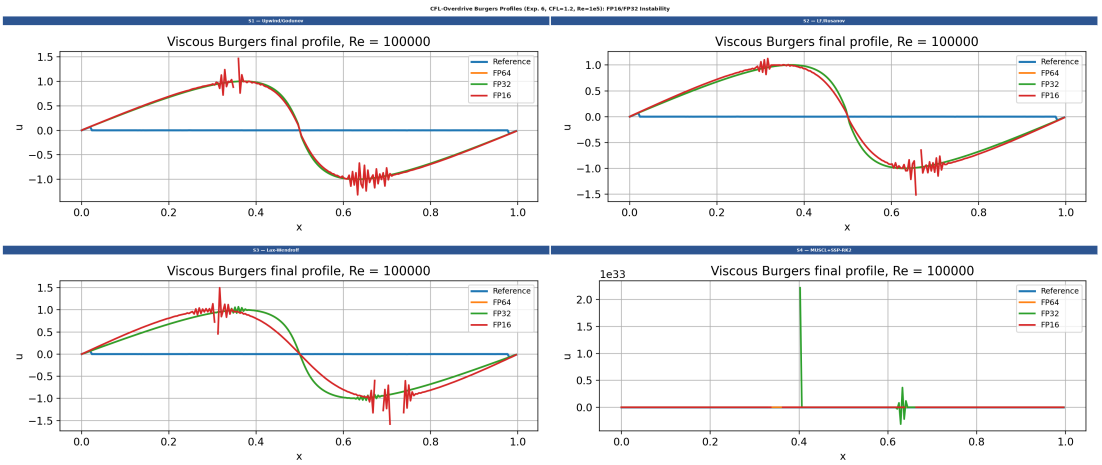

Fig. 7: CFL-overdrive Burgers (Exp. 6, CFL$=1.2$, $Re = 10^5$). FP16 diverges in all four schemes; FP32 additionally fails in Scheme 4. FP64 is stable in all cases.

E. Ultra-High-Re Shock Stress (Exp. 2)

Figure 8 shows $Re = 10^5$ front-zoom plots at $n_x = 1024$. FP64 and FP32 are nearly indistinguishable ($\Delta L_2 < 10^{-4}$), with all four schemes returning $L_2 \approx 0.706$–0.707 at FP64 and identical values at FP32. FP16 slightly over-damps the front (returning $L_2 \approx 0.699$–0.703, i.e. $\sim 0.5\%$ lower than FP64) but does not diverge at this CFL and resolution combination. The near-identical L_2 values across all precision levels (0.70–0.71) confirm that the grid-resolution ceiling, not arithmetic error, is the limiting factor at $Re = 10^5$ on $n_x = 1024$. This is a regime where investing in FP32 over FP16 provides marginal benefit: the dominant error is spatial truncation, not rounding.

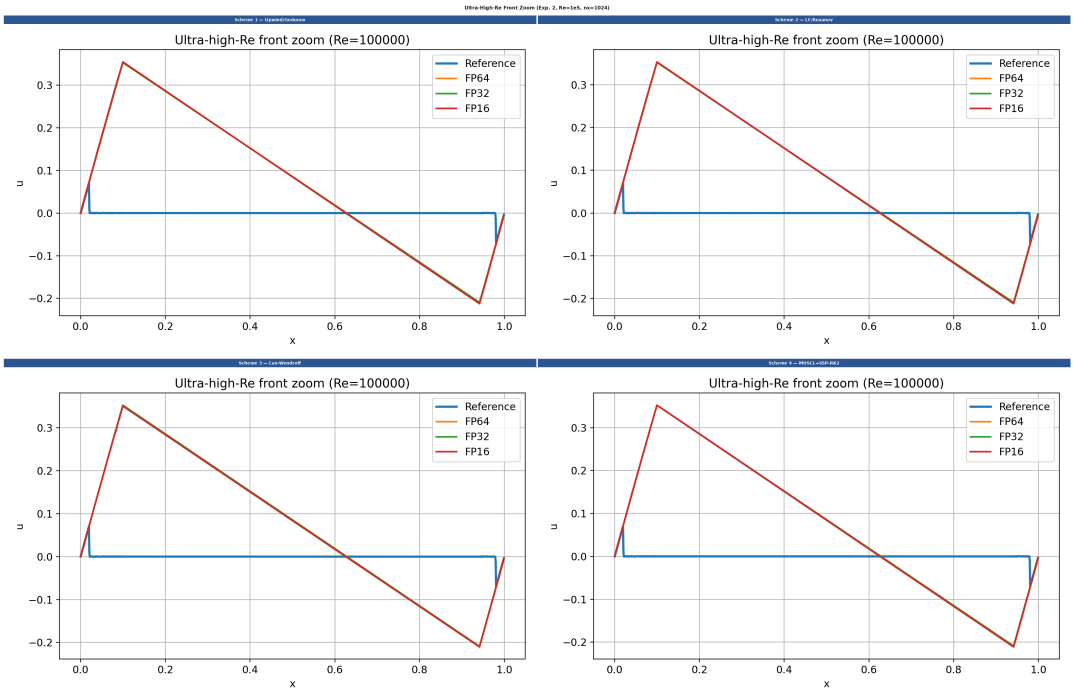

Fig. 8: Ultra-high-Re front zoom (Exp. 2, $Re = 10^5$, $n_x = 1024$). FP64/FP32 nearly identical; FP16 over-damps but does not diverge.

F. Computational Cost (Exp. 1)

Figure 9 shows the cost trade-off. Memory halves exactly with each precision step (FP64→FP32→FP16), as expected from storage formats: 8192 / 4096 / 2048 bytes for $n_x = 1024$ state vectors. Runtime on the CPU backend, however, does not follow this pattern. For the linear solver, Scheme 1 takes 0.034/0.031/0.028 s (FP64/FP32/FP16) — a modest $\approx 18\%$ improvement from

FP64 to FP16; Scheme 4 takes 0.309/0.173/0.158 s, showing a larger spread from the ≈2.9× FLOP overhead of MUSCL+SSP-RK2. Strikingly, for Burgers at $Re = 10$ Scheme 4 takes 8.95/9.71/10.08 s, i.e. FP32 and FP16 are *slower* than FP64 on this CPU backend because PyTorch emulates FP16 arithmetic in software without tensor-core acceleration, adding overhead rather than saving time. The ≈2.9× FP64/FP32 Burgers ratio between S4 (8.95 s) and S1 (1.96 s) confirms that the MUSCL+SSP-RK2 FLOP overhead drives cost far more than precision does. FP32 gives the best accuracy-per-byte balance; FP16 saves memory but delivers no runtime benefit and substantially worse accuracy on this backend.

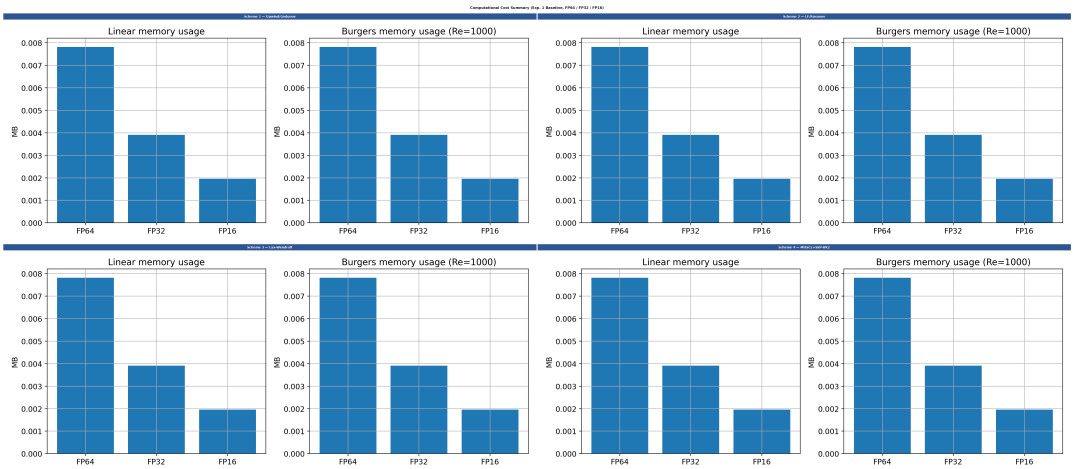

Fig. 9: Computational cost (Exp. 1). Memory halves per precision step; CPU runtime does not — FP16 is actually slower than FP64 for Scheme 4 Burgers due to software emulation.

V. Conclusions

Four clear conclusions emerge from this six-experiment, four-scheme, three-precision study.

1) **FP32 is the most reliable practical precision** for smooth, well-resolved, short-horizon problems. Across all baseline and moderate-stress experiments it tracks FP64 to within 10^{-4} relative error in L_2, conservation, and TV, while using half the memory. Its runtime is comparable to FP64 on the CPU backend tested here (GPU acceleration with tensor-core support may narrow or widen this gap). In chaotic, stiff, or very long-horizon regimes not covered by this study, FP32 accumulation errors could grow further and would require separate validation.
2) **FP16 fails in multiple distinct, independent modes:** (a) Smooth-amplitude: $L_2 = O(10^{-1})$ even for sinusoidal linear transport — two to five orders of magnitude above FP32. (b) Precision-floor phase corruption: 0.30 rad phase error for $A = 10^{-6}$ waves in Schemes 1–3. (c) Long-horizon drift: conservation error 10^6–$10^7\times$ larger than FP32 and TV degradation of up to 16%. (d) Stability: diverges at CFL= 1.2 in all four schemes. No runtime benefit is observed on CPU; on GPU with proper scaling these limitations could differ.
3) **Scheme order improves smooth accuracy but cannot compensate for under-resolved shocks.** Schemes 3 and 4 reduce smooth Burgers L_2 by two to three orders of magnitude at Re = 10–100. Once the shock is under-resolved ($Re = 10^5$, $n_x = 128$), all schemes and all precisions converge to $L_2 \approx 0.70$; the spatial truncation error completely dominates the arithmetic error.
4) **Precision and scheme order interact multiplicatively at the stability boundary.** At CFL= 1.2, first-order Schemes 1–3 tolerate FP32 but diverge with FP16. The second-order MUSCL Scheme 4 fails at both FP32 and FP16, demonstrating that the instability window for precision-induced divergence widens with reconstruction order.

The practical recommendation is: use FP64 as reference, FP32 as the default working precision, and FP16 only when memory is the primary constraint and the target regime has been verified to be smooth, well-resolved, and short in time horizon.

Contributor Roles

Nishchith C Bharadwaj (SR 26650) led governing-equation formulation, scheme-wise discretization derivations, and the methodology sections. Dhruv Yadav (SR 26641) led implementation of the four numerical schemes, experiment-framework integration, and consolidation of outputs and plots. Rajneesh Babu (SR 26058) led result verification, analysis of accuracy, conservation, stability, and runtime trends, and preparation of the results, discussion, and conclusion sections. All three authors jointly reviewed the code, cross-checked outputs against the CSV data, and finalised the report.

REFERENCES

[1] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. SIAM, 2002.
[2] IEEE, "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2019*, 2019.
[3] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.
[4] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics*, 3rd ed. Springer, 2009.
[5] D. R. Lynch, *Numerical Partial Differential Equations for Environmental Scientists and Engineers: A First Practical Course*. Springer, 2005.
[6] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, no. 2, pp. 439–471, 1988.