Reply letter to reviewers

Dear Editor and Reviewers,

We thank the Editor for coordinating the review process. We are also very grateful to the Reviewers for taking the time to review our manuscript. We found their comments helpful for improving the depth of our manuscript, and we have revised it in order to address the comments. Below, the comments are included for reference in italics, while our replies are in boxes, and screenshot(s) of the relevant revision(s) are given below our replies.

Yours sincerely,
Alovya Chowdhury, Georges Kesserwani

Reviewer 1

In the manuscript titled "LISFLOOD-FP 8.2: GPU-accelerated multiwavelet discontinuous Galerkin solver with dynamic resolution adaptivity for rapid, multiscale flood simulation", the authors develop the new LISFLOOD-FP 8.2 version integrates GPU parallelised dynamic and the DG2 solver (GPU-MWDG2) to simulate tsunami-induced flooding. It shows progressively larger speedups over the GPU-DG2 simulations from $L \geq 10$. There are still certain changes and clarifications that the authors should address prior to publication. There are still certain changes and clarifications that the authors should address prior to publication. I believe that the manuscript can be accepted for publication by the GMD after minor revision. Below, I have some general comments for the authors.

General comments:

The paper describes an innovative GPU-MWDG2 solver, so section 2.1 is the core of the calculation method in this paper. To help readers understand the detailed clarification on certain algorithmic. It is recommended that the authors provide some key portions of the code or pseudo-code as appendix. This is merely a suggestion and does not affect the validity of the paper's arguments.

Key portions of the code, in particular pseudocode and descriptions of the optimised CUDA kernels of the encoding process and the parallel tree traversal (PTT), have now been included in Appendix A.

2023), as exemplified in Figure A1a for a simplistic case with $L = 2$. Hence, $s_{[0]}^{(n+1)}$, $s_{[1]}^{(n+1)}$, $s_{[2]}^{(n+1)}$ and $s_{[3]}^{(n+1)}$ all reside in adjacent memory locations when used to produce $s_c^{(n)}$ and $d_{c,\theta}^{(n)}$ (see Figure A1b) and can be accessed using vectorised `float4` instructions (see lines 7 to 22 of Algorithm 1) to ensure coalesced memory access. The produced $s_c^{(n)}$ can then be trivially stored in the appropriate index of the array (see lines 25 to 27 of Algorithm 1), again in a coalesced manner due to the self-similar nature of the Z-order curve between refinement levels n and $n + 1$.

```
__global__
void encode_flow_kernel_mw
(
    // Inputs
)
{
    // Compute Z order indices based on thread index
    HierarchyIndex idx = blockIdx.x * blockDim.x + threadIdx.x;
    HierarchyIndex curr_lvl_idx = get_lvl_idx(level);
    HierarchyIndex next_lvl_idx = get_lvl_idx(level + 1);
    HierarchyIndex parent_idx = curr_lvl_idx + idx;
    HierarchyIndex child_idx = next_lvl_idx + 4 * (parent_idx - curr_lvl_idx);

    // Load 4 child scale coefficients in a coalesced manner using vector loads (float4)
    load_children_vector
    (
        children,
        d_scale_coeffs.eta0,
        d_scale_coeffs.eta1x,
        d_scale_coeffs.eta1y,
        child_idx
    );

    // Use encoding equation to compute and store parent scale coefficient
    d_scale_coeffs.eta0[parent_idx]  = encode_scale_0 (children);
    d_scale_coeffs.eta1x[parent_idx] = encode_scale_1x(children);
    d_scale_coeffs.eta1y[parent_idx] = encode_scale_1y(children);
}
```

Algorithm 1: Optimised CUDA kernel for performing the encoding operation in parallel. Coalesced memory access is ensured because the Z-order indices computed per thread point to adjacent memory locations in the array that is used to $s_c^{(n)}$ and $d_{c,\theta}^{(n)}$. As seen in line 15, vectorised float4 loads can be used to load the children when computing the parent. Storing the parent is trivially coalesced due to the self-similar nature of Z-order indexing across refinement levels n and $n + 1$.

With the GPU-MWDG2 solver, decoding must be performed using a parallel tree traversal algorithm (PTT) to minimise warp divergence (Chowdhury et al., 2023). To do so, the PTT starts by launching as many threads t_n as the number of cells on the finest resolution grid; for example, 16 threads t_0 to t_{15} for traversing the tree in Figure A2a. Each thread independently traverses the tree starting from the cell on the coarsest resolution grid until it reaches its leaf cell c, i.e. until it reaches either a detail that is no longer significant (line 14 of Algorithm 2) or the finest refinement level (line 28 of Algorithm 2). Once a thread reaches a leaf cell, it records the Z-order index of that leaf cell (Figures A2b and A2c).

```
__global__
void traverse_tree_of_sig_details
(
    // Inputs
)
{
    while (keep_on_traversing)
    {
        MortonCode curr_code = ( fine_code >> ( 2 * (solver_params.L - level) ) );
        HierarchyIndex curr_lvl_idx = get_lvl_idx(level);
        HierarchyIndex h_idx = curr_lvl_idx + curr_code;
        bool is_sig = d_sig_details[h_idx];

        if (!is_sig)
        {
            // Record Z order index
            keep_on_traversing = false;
        }
        else
        {
            if (!penultimate_level)
            {
                keep_on_traversing = true;
            }
            else
            {
                // Record Z order index
                keep_on_traversing = false;
            }
        }
    }
}
```

Algorithm 2: Optimised CUDA kernel for performing parallel tree traversal (PTT) that minimises warp divergence. Threads keep traversing the tree of significant details until they reach either an insignificant detail or the finest refinement level. Warp divergence is reduced because threads that end up reaching nearby leaf cells have similar traversal paths.

The paper could benefit from a deeper analysis of the trade-offs between accuracy and efficiency when choosing different error thresholds (ε). The choice of ε = 10-4 vs. 10-3 shows efficiency improvements, more discussion is needed on when to choose one over the other in practical scenarios.

More discussion on when to choose one over the other in practical scenarios has been included in the conclusion. This discussion is in addition to other substantial revisions about validating these choices of ε throughout Sect. 3.

The accuracy and efficiency of dynamic GPU-MWDG2 adaptivity was assessed for four laboratory- and field-scale tsunami-induced flood benchmarks featuring different impact event's complexity and duration (i.e. incorporating either single- or multi-peaked tsunamis and $L = 10$ or $L = 12$). GPU-MWDG2 simulation assessments were performed for $\varepsilon = 10^{-3}$ and 10^{-4}, which were also validated based on accuracy qualification with respect to benchmark-specific measured data. The accuracy assessment consistently confirms that an ε between 10^{-4} and 10^{-3} is valid: at $\varepsilon = 10^{-3}$ GPU-MWDG2 simulations

yield water level predictions as accurate as the GPU-DG2 predictions but using $\varepsilon = 10^{-4}$ can slightly improve velocity-related predictions.

Speedup assessments, based on instantaneous and cumulative metrics, suggested considerable gain when the DEM size and resolution involved $L \geq 10$ and for simulation durations that mostly spanned reduced-complexity events (i.e. single-peak tsunamis): As shown in Table 7, for single-peaked tsunamis, when the DEM required $L = 10$, GPU-MWDG2 was more than 2-fold faster than the GPU-DG2 simulations (i.e. "Monai Valley"), whereas with the DEM requiring $L = 12$, speedups of 3.3-to-4.5-fold could be achieved (i.e. "Seaside, Oregon"); in contrast, for multi-peaked tsunamis, GPU-MWDG2 speedups reduced to 1.8-fold for a DEM requiring $L = 12$ (i.e. "Tauranga Harbour") and to 1.2-fold for a smaller DEM needing $L = 10$ (i.e. "Hilo Harbour").

Table 7: Summary of GPU-MWDG2 runtimes and potential average speedups with respect to GPU-DG2.

				GPU-MWDG2				GPU-DG2
	Tsunami (impact) event		***L* (*square uniform grid*)**	**Runtime**		**Speedup**		**Runtime**
				ε		ε		
Test case	t_{end}	Single-wave tsunami	----	10^{-3}	10^{-4}	10^{-3}	10^{-4}	----
Monai Valley	22.5 (9000 s*)	Yes	10 ($2^{10} \times 2^{10}$)	16 s	20 s	2.5	2.0	40 s
Seaside Oregon	40 s (33 min*)	Yes	12 ($2^{12} \times 2^{12}$)	3.5 min	5.2 min	4.5	3.3	13 min
Tauranga Harbour	40 hr	No (three peaks)	12 ($2^{12} \times 2^{12}$)	7.5 hr	8.1 hr	1.8	1.4	11.3 hr
Hilo Harbour	6 hr	No (eleven peaks)	10 ($2^{10} \times 2^{10}$)	5.3 min	5.8 min	1.3	1.2	6.9 min

*By accounting for the physical scaling factor of the replica.

In summary, the GPU-MWDG2 solver in LISFLOOD-FP 8.2 consistently accelerates GPU-DG2 simulations of rapid multiscale flooding flows, generally yielding the greatest speedups for simulations needing $L \geq 10$—due to its scalability on the GPU which results in larger speedup with increasing L—and driven by single-peaked impact events. Choosing ε closer to 10^{-3} would maximise speedup while choosing an ε closer 10^{-4} may be useful to particularly improve the velocity-related predictions. The LISFLOOD-FP 8.2 code is open source (DOI: 10.5281/zenodo.4073010) as well as the

Figure 7 and Figure 10 are comparisons between GPU-MWDG2 and GPU-DG2, there is no description of the dashed line in the figure caption. The comparison between the two in terms of computational performance is not intuitive enough.

Figure captions have been added that explain the dashed line and include more intuitive explanations of the metrics for assessing the computational performance, e.g. in Sect. 3.1 Monai Valley.

Figure 7: Monai Valley. Measuring the computational performance of the GPU-MWDG2 and GPU-DG2 simulations using the metrics of Table 1 to assess the speedup afforded by GPU-MWDG2 adaptivity: N_{cells},the number of cells in the GPU-MWDG2 non-uniform grid,where the dashed line represents the initial value at the beginning of the simulation; R_{DG2},the computational effort of performing the DG2 solver updates at given timestep relative to the GPU-DG2 simulation; R_{MRA}, the relative computational effort of performing the MRA process at a given timestep; S_{inst}, the instantaneous speedup achieved by the GPU-MWDG2 simulation over the GPU-DG2 simulation at a given timestep; Δt, the simulation timestep; $N_{\Delta t}$, the number of timesteps taken to reach a given simulation time; C_{DG2}, the cumulative computational effort of performing the DG2 solver updates up to a given simulation time; C_{MRA}, the cumulative computational effort of performing the MRA process; C_{tot}, the total cumulative computational effort; S_{acc}, the accumulated speedup of GPU-MWDG2 over GPU-DG2 up to a given simulation time.

The paper touches on the scalability of the solver but could provide a more forward-looking discussion of future applications. For example, how would this solver perform in larger simulations involving urban flooding, river flooding that require coupling with other

environmental models? A brief discussion of scalability in more complex, coupled systems could enhance the paper's impact.

A brief discussion of scalability has been included.

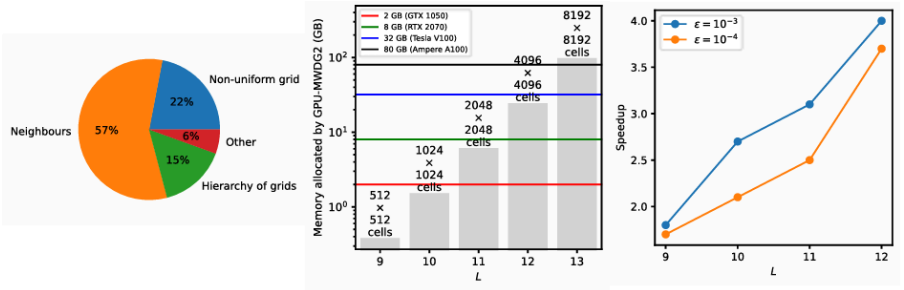

Figure 4: GPU memory consumed by dynamic GPU-MWDG2 adaptivity. Left panel shows the percentage breakdown of the GPU allocated against the maximum refinement level L; the numbers on top of the bars show the number of cells for a given value of L. The horizontal lines indicate the memory limits of four GPU cards. Right panel shows the scalability of the GPU-MWDG2 solver (i.e. an increasing speedup against an increasing workload) by running GPU-MWDG2 and GPU-DG2 simulations against increasing values of L for the "Monai valley" test case (Sect. 3.1) and plotting the speedups of the GPU-MWDG2 simulations compared to the GPU-DG2 simulations.

The right panel of Figure 4 shows speedups of GPU-MWDG2 simulations over GPU-DG2 simulations with respect to L for the "Monai valley" example (Sect. 3.1) to assess the scalability of the GPU-MWDG2 solver in relation to the simulated problem size (defined by L) for the selected A100 GPU card. It can be seen that the larger the simulated problem size, the higher and steeper the speedup. This indicates that the GPU-MWDG2 solver exhibits better scalability potential with increasing size of the simulated problem, such as coupled urban and river flooding over a very large area, requiring $L \geq 13$.

Specific comments:

It is very rare for DOIs and data links to appear in an abstract.

We have removed the DOIs and data links from the abstract.

Abstract. The second-order discontinuous Galerkin (DG2) solver of the two-dimensional (2D) shallow water equations in the raster-based LISFLOOD-FP 8.0 hydrodynamic modelling framework is mostly suited for predicting small-scale transients that emerge in rapid, multiscale floods caused by impact events like tsunamis. However, this DG2 solver can only be used for simulations on a uniform grid where it may yield inefficient runtimes even when using its graphics processing unit (GPU) parallelised version (GPU-DG2). To boost efficiency, the new LISFLOOD-FP 8.2 version integrates GPU parallelised dynamic (in time) grid resolution adaptivity of multiwavelets (MW) with the DG2 solver (GPU-MWDG2). The GPU-MWDG2 solver performs dyadic grid refinement, starting from a single grid cell, with a maximum refinement level, L, based on the resolution of the Digital Elevation Model (DEM). Furthermore, the dynamic GPU-MWDG2 adaptivity is driven by one error threshold, ε, against normalised details of all prognostic variables. Its accuracy and efficiency, as well as the practical validity of recommended ε choices between 10^{-4} and 10^{-3}, are assessed for four laboratory/field-scale benchmarks of tsunami-induced flooding with different impact event complexities (i.e. single- vs. multi-peaked) and L values. Rigorous accuracy and efficiency metrics consistently show that GPU-MWDG2 simulations with $\varepsilon = 10^{-3}$ preserve the predictions of the GPU-DG2 simulation on the uniform DEM grid, whereas $\varepsilon = 10^{-4}$ may slightly improve velocity-related predictions. Efficiency-wise, GPU-MWDG2 yields considerable speedups from $L \geq 10$—due to its scalability on the GPU with increasing L—which can be around 2.0-to-4.5-fold. Generally, the bigger the $L \geq 10$, the lower the event complexity over the simulated duration, and the closer the ε to 10^{-3}, the larger the GPU-MWDG2 speedups over GPU-DG2. The LISFLOOD-FP 8.2 code is open source, under the GPL v3.0 licence, as well as the simulated benchmarks' set-up files and datasets, with a video tutorial and further documentation on https://www.seamlesswave.com/Adaptive (last accessed: 6 July 2025).

Line #219-#220, ensure consistency in the formatting of references. For example, "Kesserwani et al. (2023)" is used multiple times but isn't always consistent in placement or citation style.

Consistency has now been ensured throughout the manuscript using Mendeley Cite.

Figure Legends Overlap: In some figures, such as Figure 4(b), the legends and lines overlap, making it difficult to interpret the data. Please revise the figures to ensure clarity.

Overlapping of legends and lines has been addressed, e.g. in Figure 4.

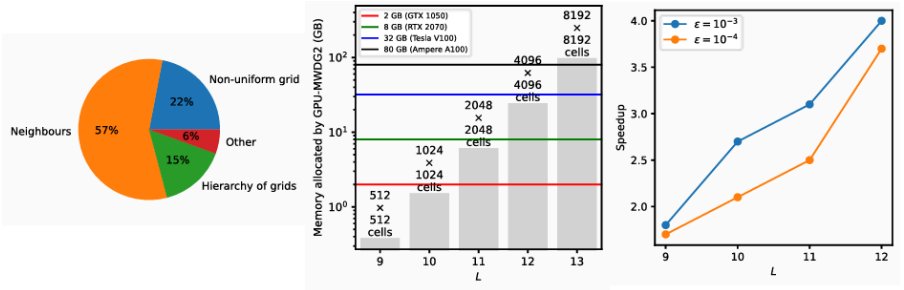

Figure 4: GPU memory consumed by dynamic GPU-MWDG2 adaptivity. Left panel shows the percentage breakdown of the GPU

Reviewer 2

Review of

LISFLOOD-FP 8.2: GPU-accelerated multiwavelet discontinuous Galerkin solver with dynamic resolution adaptivity for rapid, multiscale flood simulation

This paper validates the accuracy and efficiency of a new version of the LISFLOOD-FP inundation model that provides dynamical multiscale grid adaptivity by incorporating the previously developed GPU-MWDG2 solver. The previous version of LISFLOOD-FP only allowed static adaptivity, based on the initial conditions.

The results suggest that the new, adaptive, LISFLOOD-FP is both accurate and efficient (up to 4.5 times faster). The paper represents a significant advance in efficient flood modelling using the standard 2D shallow water equation model. I also believe the paper will be interesting for more general readers, who want to better understand dynamically adaptive methods.

Most of my questions and suggestions are aimed at making the paper more understandable and higher impact by clarifying basic properties of the method, however I also have some concerns about the tolerance and validation of the method.

Questions:

1. The abstract should be revised to be clearer for non-expert users of the LISFLOOD-FP models.

(a) It would be helpful to specify that the model is based on the two-dimensional shallow water equations (not the multilayer three-dimensional shallow water equations used in ocean and atmosphere modelling).

We have specified in the abstract that the model is based on the two-dimensional (2D) shallow water equations.

10 **Abstract.** The second-order discontinuous Galerkin (DG2) solver of the two-dimensional (2D) shallow water equations in the raster-based LISFLOOD-FP 8.0 hydrodynamic modelling framework is mostly suited for predicting small-scale

(b) More details on the error threshold are needed in the abstract and Section 1. Is epsilon a relative error threshold that controls all prognostic variables, or just some variables? What does it measure or control? Or perhaps it is a relative error threshold for the tendencies? Please clarify what epsilon measures/controls and how it set. See also question 3 below: the results don't seem to show that epsilon controls the error of the prognostic variables.

We have included more details on the error threshold.

Abstract.

Abstract. The second-order discontinuous Galerkin (DG2) solver of the two-dimensional (2D) shallow water equations in the raster-based LISFLOOD-FP 8.0 hydrodynamic modelling framework is mostly suited for predicting small-scale transients that emerge in rapid, multiscale floods caused by impact events like tsunamis. However, this DG2 solver can only be used for simulations on a uniform grid where it may yield inefficient runtimes even when using its graphics processing unit (GPU) parallelised version (GPU-DG2). To boost efficiency, the new LISFLOOD-FP 8.2 version integrates GPU parallelised dynamic (in time) grid resolution adaptivity of multiwavelets (MW) with the DG2 solver (GPU-MWDG2). The GPU-MWDG2 solver performs dyadic grid refinement, starting from a single grid cell, with a maximum refinement level, L, based on the resolution of the Digital Elevation Model (DEM). Furthermore, the dynamic GPU-MWDG2 adaptivity is driven by one error threshold, ε, against normalised details of all prognostic variables. Its accuracy and efficiency, as well as the practical validity of recommended ε choices between 10^{-4} and 10^{-3}, are assessed for four laboratory/field-scale benchmarks of tsunami-induced flooding with different impact event complexities (i.e. single- vs. multi-peaked) and L values. Rigorous accuracy and efficiency metrics consistently show that GPU-MWDG2 simulations with $\varepsilon = 10^{-3}$ preserve the predictions of the GPU-DG2 simulation on the uniform DEM grid, whereas $\varepsilon = 10^{-4}$ may slightly improve velocity-related predictions. Efficiency-wise, GPU-MWDG2 yields considerable speedups from $L \geq 10$—due to its scalability on the GPU with increasing L—which can be around 2.0-to-4.5-fold. Generally, the bigger the $L \geq 10$, the lower the event complexity over the simulated duration, and the closer the ε to 10^{-3}, the larger the GPU-MWDG2 speedups over GPU-DG2. The LISFLOOD-FP 8.2 code is open source, under the GPL v3.0 licence, as well as the simulated benchmarks' set-up files and datasets, with a video tutorial and further documentation on https://www.seamlesswave.com/Adaptive (last accessed: 6 July 2025).

Introduction.

raster-formatted DEM file. Meanwhile, the coarsest resolution grid consists of $2^0 \times 2^0 = 1$, i.e. a single cell. On the hierarchy of grids, the scaled DG2 modelled data for each prognostic variable are compressed into higher-resolution MW coefficients, or *details*, which are added to the coarsest resolution data. The details of all the prognostic variables are normalised and packed in a dataset of normalised details, which are then compared to an error threshold $0 < \varepsilon < 1$ for retaining the significant details. The retained significant details are added to the coarsest resolution DG2 modelled data, leading to a multiscale representation of DG2 modelled data on a non-uniform grid. As the scaling, analysis, and reconstruction of DG2 modelled data are inherent to the MRA procedure, the mimetic properties of the reference GPU-DG2 solver on the finest resolution grid ($2^L \times 2^L$) can readily be preserved for $\varepsilon \leq 10^{-3}$, based on studies considering a range for ε between 10^{-6} to 10^{-1} (Kesserwani et al., 2019; Kesserwani & Sharifian, 2020). Another benefit of the MRA procedure is the sole reliance on ε to sensibly control the amount of local grid resolution coarsening for all prognostic variables. Furthermore, it was also shown that, for the same ε, dynamic MWDG2 adaptivity avoids unnecessary refinement compared to its first-order counterpart, and using $\varepsilon \geq 10^{-4}$ leads to simulations that are faster than first-order finite volume solvers on the finest resolution grid (Kesserwani & Sharifian, 2020).

(c) How is the refinement level L defined? Is it a dyadic (power of 2) refinement of a coarsest grid? What is the coarsest grid? How does the number of computational elements depend on L? Does the non-adaptive simulation use the same L (but with no adaptivity)?

Regarding the first two questions:

LISFLOOD-FP 8.2. To elaborate, the MWDG2 solver automates local grid resolution coarsening on an adaptive grid using the multiresolution analysis (MRA) of MW applied to scaled DG2 modelled data (for all the prognostic variables) that exist in a hierarchy of grids. This hierarchy consists of dyadically coarser grids relative to the finest grid in the hierarchy (Kesserwani et al., 2019; Kesserwani & Sharifian, 2020; Sharifian et al., 2019), whereby the finest resolution grid is associated with a maximum refinement level, L, and consists of ($2^L \times 2^L$) cells—with L selected to match the resolution of a raster-formatted DEM file. Meanwhile, the coarsest resolution grid consists of $2^0 \times 2^0 = 1$, i.e. a single cell. On the hierarchy

Regarding the last question, i.e. whether the non-adaptive simulation uses *L*: no, it runs on the DEM grid instead of the square uniform grid, which is the same resolution as the square uniform grid but different dimensions, i.e. *N* x *M* like the DEM grid:

300 **3 Evaluation of GPU-MWDG2 adaptivity**

The efficiency of the GPU-MWDG2 solver is evaluated relative to the GPU-DG2 solver—which is always run on the DEM grid—using the metrics proposed in Table 1. The potential speedup afforded by GPU-MWDG2 adaptivity is hypothesised to

(d) It would be helpful to make more general conclusions about the efficiency of the adaptive method, rather than reporting results for a specific example. What determines the refinement level where the adaptive method is faster? How should the tolerance epsilon be set to achieve an acceptable balance of efficiency and accuracy for a given application?

More general conclusions have been given.

Abstract.

Efficiency-wise, GPU-MWDG2 yields considerable speedups from $L \geq 10$—due to its scalability on the GPU with increasing L—which can be around 2.0-to-4.5-fold. Generally, the bigger the $L \geq 10$, the lower the event complexity over the

25 simulated duration, and the closer the ε to 10^{-3}, the larger the GPU-MWDG2 speedups over GPU-DG2. The LISFLOOD-FP

Conclusion.

In summary, the GPU-MWDG2 solver in LISFLOOD-FP 8.2 consistently accelerates GPU-DG2 simulations of rapid multiscale flooding flows, generally yielding the greatest speedups for simulations needing $L \geq 10$—due to its scalability on the GPU which results in larger speedup with increasing L—and driven by single-peaked impact events.

690 Choosing ε closer to 10^{-3} would maximise speedup while choosing an ε closer 10^{-4} may be useful to particularly improve the velocity-related predictions. The LISFLOOD-FP 8.2 code is open source (DOI: 10.5281/zenodo.4073010) as well as the

2. Introduction

(a) Par 3: Low order finite volume discretizations are typically used because the goal is to preserve various mimetic properties of the discretization (e.g. conservation of mass to machine precision, ensuring that there is nos spurious generation of vorticity from a uniform vorticity field). This is considered essential for climate modelling, for example. Does LISFLOOD-FP discretely conserve mass and other mimetic properties? Does the adaptive version retain these conservation properties? If not, why are mimetic properties like mass conservation not important?

We have addressed these questions now in the introduction.

35 It includes a diffusive wave solver (Hunter et al., 2005), a local inertial solver (Bates et al., 2010), a first-order finite volume solver, and a second-order discontinuous Galerkin (DG2) solver (Shaw et al., 2021), all already supported with robustness treatments (i.e. for topographic and friction discretisation with wetting and drying), ensuring numerical mass conservation to machine precision. The DG2 solver is the most complex numerically, requiring three times more modelled data (degrees of freedom) per prognostic variable and at least twelve times more computations per cell compared to any of the other solvers

40 in LISFLOOD-FP. This complexity arises from its locally conservative formulation that pays off with more inherent mimetic properties (Ayog et al., 2021; Kesserwani et al., 2018), i.e. numerical momentum conservation and reduced (spurious) error dissipation, leading to more accurate velocity fields and long-duration flood simulations (Ayog et al., 2021; Kesserwani &

of computational cells as much as allowed by the complexity of the DEM and flow solution. The LISFLOOD-FP 8.2 version is unique in providing a single, mathematically sound hydrodynamic modelling framework that implements dynamic grid resolution adaptivity entirely on the GPU for achieving raster-grid DG2 simulations that can preserve a similar level of predictive accuracy and robustness as alternative GPU-DG2 simulations on a uniform grid. The framework is formulated in Kesserwani & Sharifian (2020), having been informed by Caviedes-Voullième & Kesserwani (2015), Gerhard et al. (2015), Kesserwani et al. (2015) and Caviedes-Voullième et al. (2020), so as to preserve all the mimetic properties inherent in the reference DG2 hydrodynamic solver (Kesserwani et al., 2018; Kesserwani & Sharifian, 2023). Here, it has been

(b) Par 5: How should the number of retained computational modes scale with the error threshold? This can usually be calculated for adaptive wavelet methods, based on the order or accuracy of the wavelets and dimensionality of the problem (2 in this case).

The number of computational modes retained depends on the complexity of the DEM and the flow solution.

DG2 solver every simulation timestep to achieve as much local grid resolution coarsening as possible. Namely, grid coarsening is applied to the grid cells covering either regions of smooth flow or DEM features, thereby reducing the number of computational cells as much as allowed by the complexity of the DEM and flow solution. The LISFLOOD-FP 8.2 version

(c) Par 5: What is special about L=9? Can you make more general claims about efficiency? Is the time step fixed for all scales?

Some more general reasoning about efficiency has been included, as well as answering the timestep question.

CPU and was based on the reference DG2 hydrodynamic solver in Kesserwani et al. (2018). Chowdhury et al. (2023) devised a computationally efficient GPU implementation of wavelet adaptivity integrated with a first-order finite volume (FV1) solver counterpart. They reported that the speedup of their GPU implementation over the uniform FV1 solver run on the finest $2^L \times 2^L$ resolution grid scaled up with increasing L, becoming considerable for $L \geq 9$. In fact, for $L \geq 9$, with the same global timestep Δt used, there would be enough memory and compute workload to bound the GPU, which the adaptive solver can reduce unlike the uniform solver. Kesserwani & Sharifian (2023) extended the GPU implementation of

3. Section 3

The RMS errors presented in Tables 3 to 6 do not confirm that the tolerance actually controls the error of either the height or velocity variables. A valid dynamically adaptive wavelet method should be characterized by the fact the tolerance controls the error: error should ideally be proportional to tolerance, or, at least the error should decrease significantly when the tolerance is decreased by a factor of 10. In many cases the results show the error decreases by 10-30% or so, and in at least cases the error actually grows, which is concerning since it suggests that epsilon does not control the error of the simulation (at least for these choices of epsilon).

The error, in particular the perturbation error (PE), is only expected to decrease significantly, e.g. around a factor of 10, when considering simulations with smooth numerical solutions.

According to Gerhard et al. (2015) and Gerhard & Müller (2016), a selected ε is valid if two conditions are met:

(i) the "full error" (FE) (the error between the GPU-MWDG2 predictions and the analytical solution) is in the same order of magnitude as the "discretisation error" (DE), i.e. the error between the GPU-DG2 predictions and the reference data; and

(ii) the "perturbation error" (PE) (the error between the GPU-MWDG2 and GPU-DG2 predictions) decreases by one order of magnitude when ε is also reduced by one order of magnitude for smooth numerical solutions.

This is seen in the Monai Valley and Tauranga Harbour test cases.

Table 3: Monai valley. DE, FE and PE quantified using Eqs. (1) to (3) considering measured and predicted time series of the water surface elevation at points 1, 2 and 3 in Figure 7. The errors are computed in order to check for fulfilment of conditions (i) and (ii) and to confirm the validity of selecting $\varepsilon = 10^{-3}$ and 10^{-4} in this test case.

	DE	**FE**		**PE**	
Observation point	-	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
Point 1	0.012429	0.012544	0.012414	0.000814	0.000218
Point 2	0.004829	0.005277	0.004818	0.001606	0.000767
Point 3	0.007427	0.007690	0.007323	0.001209	0.000474

Table 5: Tauranga Harbour. DE, FE and PE quantified using Eqs. (1) to (3) considering measured and predicted time series of the water surface elevation $h + z$ at observation points A Beacon, Tug Harbour, Sulphur Point and Moturiki (top left panel of Figure 12), and the speed ($\sqrt{u^2 + v^2}$) at observation point ADCP. The errors are considered in order to check for fulfilment of conditions (i) and (ii) and confirm the validity of selecting $\varepsilon = 10^{-3}$ and 10^{-4} in this test case.

		DE	**FE**		**PE**	
Observation point	**Quantity**		$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
A Beacon	$h + z$	0.150932	0.153826	0.151580	0.013583	0.002991
Tug Harbour	$h + z$	0.155685	0.157995	0.157293	0.032124	0.008822
Sulphur Point	$h + z$	0.154380	0.162152	0.154730	0.040685	0.009663
Moturiki	$h + z$	0.225445	0.225461	0.225444	0.028375	0.008321
ACDP	$Speed$	0.150932	0.153826	0.151580	0.013583	0.002991

These results merit much more discussion and explanation if they are intended to validate the error control of the adaptive method. It would be very helpful to consider a wider set of tolerances epsilon for large L (e.g. 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6 or another suitable set of 6 or more epsilon values) to properly understand how and if epsilon controls the error of the prognostic variables.

We have now included much more discussion and explanation in the manuscript.

Introduction.

representation of DG2 modelled data on a non-uniform grid. As the scaling, analysis, and reconstruction of DG2 modelled data are inherent to the MRA procedure, the mimetic properties of the reference GPU-DG2 solver on the finest resolution grid ($2^L \times 2^L$) can readily be preserved for $\varepsilon \leq 10^{-3}$, based on studies considering a range for ε between 10^{-6} to 10^{-1} (Kesserwani et al., 2019; Kesserwani & Sharifian, 2020). Another benefit of the MRA procedure is the sole reliance on ε to sensibly control the amount of local grid resolution coarsening for all prognostic variables. Furthermore, it was also shown that, for the same ε, dynamic MWDG2 adaptivity avoids unnecessary refinement compared to its first-order counterpart, and using $\varepsilon \geq 10^{-4}$ leads to simulations that are faster than first-order finite volume solvers on the finest resolution grid (Kesserwani & Sharifian, 2020).

Sect. 3

3 Evaluation of GPU-MWDG2 adaptivity

The efficiency of the GPU-MWDG2 solver is evaluated relative to the GPU-DG2 solver—which is always run on the DEM grid—using the metrics proposed in Table 1. The potential speedup afforded by GPU-MWDG2 adaptivity is hypothesised to depend on the duration and complexity of the impact event (Sect. 2.3) as well as the available DEM grid size, which dictates the choice for the L (Sect. 2.1). Therefore, the evaluation is performed by simulating four realistic test cases of tsunami-induced flooding with various impact event complexity, ranging from a single-wave to wave-train tsunamis, and DEM sizes, requiring L = 10 or 12 for the *square uniform grid*. The properties of the selected test cases are summarised in Table 2. The simulations were run on the Stanage high performance computing cluster of the University of Sheffield using an A100 GPU card with 80 GB of memory to accommodate the memory cost of using $L = 12$ (see Sect. 2.2).

The GPU-MWDG2 simulations are run with $\varepsilon = 10^{-4}$ and 10^{-3}, which is the range of ε where the predictive accuracy of the GPU-DG2 simulations can be preserved while achieving considerable speedups (Kesserwani & Sharifian, 2020, 2023). Here, the practical usability of these ε values is also validated based on the heuristic rule that ε should be proportional to Δx^L multiplied by a scaling factor C_ϵ (Caviedes-Voullième et al., 2020); where, Δx^L is the (finest) resolution of the DEM grid and C_ε represents the ratio between the smallest and largest length scale that can be captured during a simulation. For example, using $\varepsilon = 10^{-3}$ and 10^{-4} in the Monai Valley test case with the available grid resolution $\Delta x^L = 0.007$ yields $C_{10^{-3}} = 0.14$ and $C_{10^{-4}} = 0.014$, respectively, which are the highest length-scale ratios that can be captured in these simulations. Similarly, $C_{10^{-3}}$ and $C_{10^{-4}}$ were estimated for the other test cases based on the best available DEM resolutions, and they listed in Table 2.

According to Gerhard et al. (2015) and Gerhard & Müller (2016), a selected ε is valid if two conditions are met:

(i) the "full error" (FE) (the error between the GPU-MWDG2 predictions and the analytical solution) is in the same order of magnitude as the "discretisation error" (DE), i.e. the error between the GPU-DG2 predictions and the reference data; and

(ii) the "perturbation error" (PE) (the error between the GPU-MWDG2 and GPU-DG2 predictions) decreases by one order of magnitude when ε is also reduced by one order of magnitude for smooth numerical solutions.

In the present work, conditions (i) and (ii) can be approximately evaluated at observation points where measured data are available—to confirm the validity of $\varepsilon = 10^{-3}$ and 10^{-4}—by quantifying FE, DE and PE, which were calculated using the root mean squared error (RMSE):

$$FE = \frac{1}{N_{obs}} \sqrt{\Sigma_i^{N_{obs}} (P_i^{obs} - P_i^{MWDG2})^2} \quad (1)$$

$$DE = \frac{1}{N_{obs}} \sqrt{\Sigma_i^{N_{obs}} (P_i^{obs} - P_i^{DG2})^2} \quad (2)$$

$$PE = \frac{1}{N_{obs}} \sqrt{\Sigma_i^{N_{obs}} (P_i^{MWDG2} - P_i^{DG2})^2} \quad (3)$$

where N_{obs} is the number of observation points and P_i^{obs}, P_i^{DG2} and P_i^{MWDG2} are the i^{th} observation point, GPU-DG2 prediction and GPU-MWDG2 prediction respectively. The predictions may be any of the prognostic variables, to include water surface elevation $h + z$ or the u or v components of the velocity, or velocity-related variables.

Sect. 3.1

In Figure 8, the GPU-MWDG2 predictions at $\varepsilon = 10^{-3}$ and 10^{-4} and the GPU-DG2 prediction of the water surface elevation $h + z$ at points 1, 2 and 3 (i.e. the coloured points in the top left panel of Figure 7) are shown. All three predictions are very similar to each other visually, although at point 2, the wave arrival time is seen to be slightly underpredicted using $\varepsilon = 10^{-3}$. In Table 3, the FE, DE and PE at the same points are shown: the FE and DE are of the same order of magnitude, while the PE decreases by around an order of magnitude since the simulation does not involve strong discontinuities. This shows fulfilment of conditions (i) and (ii), in turn confirming the validity of the selected ε values in this test case.

Table 3: Monai valley. DE, FE and PE quantified using Eqs. (1) to (3) considering measured and predicted time series of the water surface elevation at points 1, 2 and 3 in Figure 7. The errors are computed in order to check for fulfilment of conditions (i) and (ii) and to confirm the validity of selecting $\varepsilon = 10^{-3}$ and 10^{-4} in this test case.

	DE	**FE**		**PE**	
Observation point	-	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
Point 1	0.012429	0.012544	0.012414	0.000814	0.000218
Point 2	0.004829	0.005277	0.004818	0.001606	0.000767
Point 3	0.007427	0.007690	0.007323	0.001209	0.000474

Sect. 3.2

In Figure 11, the time series predicted by GPU-MWDG2 using $\varepsilon = 10^{-3}$ and 10^{-4} and GPU-DG2 of the prognostic variables $h + z$, the u component of the velocity, and the derived momentum variable $M_x = 0.5hu^2$ at points A1, B6 and D4 are shown. Point A1 is one of the left-most crosses in Figure 9, point B6 is one of the central crosses, and point D4 is one of the right-most crosses. At all three points, the predictions are visually similar to each other except at point D4, where, for $\varepsilon = 10^{-3}$, the impact wave height is slightly overpredicted, while the velocity and momentum is overpredicted between 33 s and 43 s. Compared to the measured data, the predictions all follow the trailing part of the measurement time series quite well, but at the peaks, they are underpredicted at point A1, overpredicted at point B6, and out of phase at point D4. These differences, which manifest most noticeably at the peaks, may be due to the incoming bore of the impact wave, which had invalidated measurements and required repeating experiments, possibly introducing uncertainty into the measured data (H. Park et al., 2013).

Table 4 shows the associated DE, FE and PE at points A1, B6 and D4. The errors confirm fulfilment of conditions (i) and (ii) and thus show that selected ε values are valid: the DE and FE are in the same order of magnitude, while decreasing ε from 10^{-3} to 10^{-4} leads to a reduction in the PE—although the reduction is less than an order of magnitude compared to the last test case (Sect. 3.1) due to the strong discontinuities introduced into the numerical solution by the buildings in the urban area.

Table 4: Seaside Oregon. DE, FE and PE quantified using Eqs. (1) to (3) considering measured and predicted time series of the prognostic variables $h + z$, u and M_x at points A1, B6 and D4 (shown in Figure 9). The errors are considered in order to check for fulfilment of conditions (i) and (ii) and confirm the validity of selecting $\varepsilon = 10^{-3}$ and 10^{-4} in this test case.

		DE	FE		PE	
Observation point	**Quantity**	-	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
A1	$h + z$	0.168783	0.168835	0.168764	0.003100	0.001857
	u	0.926009	0.928524	0.925210	0.057247	0.026415
	M_x	0.316526	0.316487	0.316498	0.002713	0.000809
B6	$h + z$	0.065732	0.065925	0.065724	0.004869	0.003087
	u	1.597801	1.598185	1.597701	0.060968	0.042121
	M_x	0.148552	0.148858	0.148491	0.007611	0.003425
D4	$h + z$	0.013469	0.014846	0.012927	0.007300	0.004517
	u	0.145877	0.169860	0.160767	0.194791	0.121198
	M_x	0.005891	0.007861	0.006407	0.007534	0.003531

Sect. 3.3

In Figure 14, the top four panels show the time series of the water surface elevation predicted by GPU-MWDG2 using $\varepsilon = 10^{-3}$ and 10^{-4} and by GPU-DG2 at observation points A Beacon, Tug Harbour, Sulphur Point and Moturiki (top left panel of Figure 12). All of the predictions are visually in agreement with each other as well as the measured data. Meanwhile, the bottom panel of Figure 14 shows the speed ($\sqrt{u^2 + v^2}$) predicted at observation point ADCP, where the predictions are still visually in agreement with each other, but less so with the measured speed, as was also observed for the velocity-related predictions in the last test case (Sect. 3.2). Table 5 quantifies the extent of agreement by showing the associated DE, FE and PE at the observation points. The DE and FE are in the same order of magnitude, while the PE is reduced by close to an order of magnitude when ε is decreased from 10^{-3} to 10^{-4}, fulfilling conditions (i) and (ii) and thereby validating the used ε choices.

Table 5: Tauranga Harbour. DE, FE and PE quantified using Eqs. (1) to (3) considering measured and predicted time series of the water surface elevation $h + z$ at observation points A Beacon, Tug Harbour, Sulphur Point and Moturiki (top left panel of Figure 12), and the speed ($\sqrt{u^2 + v^2}$) at observation point ADCP. The errors are considered in order to check for fulfilment of conditions (i) and (ii) and confirm the validity of selecting $\varepsilon = 10^{-3}$ and 10^{-4} in this test case.

		DE	FE		PE	
Observation point	**Quantity**		$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
A Beacon	$h + z$	0.150932	0.153826	0.151580	0.013583	0.002991
Tug Harbour	$h + z$	0.155685	0.157995	0.157293	0.032124	0.008822
Sulphur Point	$h + z$	0.154380	0.162152	0.154730	0.040685	0.009663
Moturiki	$h + z$	0.225445	0.225461	0.225444	0.028375	0.008321
ACDP	$Speed$	0.150932	0.153826	0.151580	0.013583	0.002991

Sect. 3.4

Figure 17 shows the time series of the water surface elevation at observation points labelled "Control point" and "Tide gauge", and the time series of the u and v velocity components at the points labelled "ADCP HA1125" and "ADCP HA1126" (top left panel in Figure 15) predicted by GPU-DG2 using $\varepsilon = 10^{-3}$ and 10^{-4} and GPU-MWDG2. All the predictions are visually very similar to each other except for the u prediction, where some differences are seen for $\varepsilon = 10^{-3}$ and 10^{-4}. Compared to the measured data, both agreement and differences are seen, but it may not be possible to make a confident judgment due to the sparsity in the measured data—see the jagged shape of the black line.

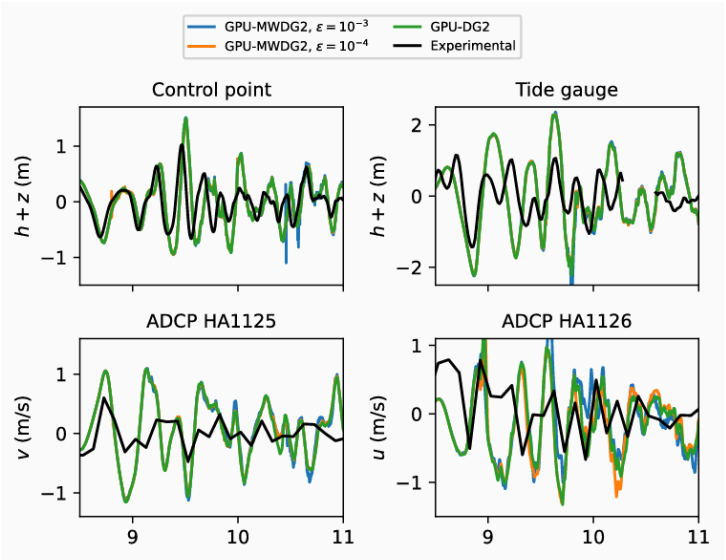

Figure 17: Hilo Harbour. Time series produced by GPU-DG2 and GPU-MWDG2, for the water surface elevation ($h + z$) at "control point" and "Tide gauge" (labelled in Figure 17), and for the velocity components at "ADCP HA1125" and ADCP HA 1126 (labelled in Figure 17), compared to the experimental results.

Table 6 shows the associated DE, FE and PE at the observation points. An outlier is noticeable in this test case for the FE at the Tide gauge and ADCP HA1125 observation points, which is actually smaller than the DE: this might be due missing data between 10 and 11 h (see the gap in the black line in the Tide gauge panel in Figure 17). Nonetheless, generally, the DE and FE in the same order of magnitude, while the PE reduces when ε is decreased from 10^{-3} to 10^{-4}, showing fulfilment of conditions (i) and (ii) and validating the choice of ε.

Table 6: Hilo Harbour. DE, FE and PE quantified using Eqs. (1) to (3) and considering measured vs predicted time series of the prognostic variable $h + z$ at observation points Tide gauge and Control point, and of the prognostic variables v and u at observation points ADCP 1125 and 1126 respectively. The errors are considered in order to check for fulfilment of conditions (i) and (ii) and confirm the validity of selecting $\varepsilon = 10^{-3}$ and 10^{-4} in this test case.

		DE	**FE**		**PE**	
Observation point	**Quantity**	-	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
Control point	$h + z$	0.380857	0.382853	0.381485	0.009817	0.007797
Tide gauge	$h + z$	0.936661	0.941419	0.931434	0.073245	0.051771
ADCP HA1125	v	0.514478	0.477909	0.50105	0.16383	0.221694
ADCP HA1126	u	0.609671	0.587934	0.598625	0.281472	0.263449

Reviewer 3

The manuscript presents an adaptive Discontinuous Galerkin solver for the shallow water equations implemented to be executed on GPUs. The main goal is to show its performance

on tsunami inundation cases, which is well achieved. Overall I have a positive opinion of the manuscript. Nevertheless, I believe that the manuscript could benefit from some additional discussion concerning the observed results and the attribution of the behaviours to different aspects of the numerical solver and the adaptivity.

1- Introduction, line 96-105. The authors refer to "rapid" and "slow-to-rapid" flows. What this means is not clear. Also in line 578.

We have clarified this with examples.

DG2 simulations were shown to accurately reproduce slow to gradual fluvial/pluvial flooding flows at unusually coarse DEM resolutions (Ayog et al., 2021; Kesserwani, 2013; Kesserwani & Wang, 2014), but they primarily excel at capturing the small-scale rapid flow transients that occur over a wide range of spatial and temporal scales (Kesserwani & Sharifian, 2023; Sharifian et al., 2018; Sun et al., 2023). Such transients are typical of flooding flows driven by impact event(s) like tsunami(s) and including zones of flow recirculation past (un)submerged island(s). Hence, DG2 simulations are likely suited for obtaining accurate modelling of rapid, multiscale flooding, such as for tsunami-induced inundations. Within

2- Line 122: "which its dynamic adaptivity". There seems to be something off in the syntax in this sentence.

The syntax has been fixed.

The technical description of the GPU-MWDG2 solver has been reported in previous papers (Kesserwani & Sharifian, 2020, 2023), whose dynamic adaptivity has here been further computationally optimised to improve memory

3- Line 170: why is it necessary to trigger refinement around the wall? In the figure of the Monai grid the refinement seems to be outside of the flow domain. Please explain why this is necessary.

We have now explained why this is necessary.

elevation). For example, for the Monai valley test case, the `wall_height` keyword is specified to 0.5 m to generate a wall that is high enough to prevent any water from leaving the flow domain. The `refine_wall` and `ref_thickness` keywords, followed by an integer for the latter, typically between 16 and 64, should also be typed in the parameter file to prevent GPU-MWDG2 from excessively coarsening the non-uniform grid around the walls within the flow domain (labelled with the curly braces in Figure 3). This prevents very coarse cells in the empty areas, covered by user-added artificial topography, from being next to very fine cells in the actual areas, covered by actual DEM topography, that could otherwise cause unphysical predictions. For the Monai valley test case, the `refine_wall` keyword is specified to trigger refinement around the wall,

4- line 187: 15% of memory is allocated for arrays storing the hierarchy of uniform grids. This is unclear. What needs to be stored for the hierarchy of grids? This seems to be a simple recursion. Do you need to store the geometry for some reason? What is stored in these arrays requiring so much memory?

So much memory is required because the DG2 shape coefficients are deep-copied.

for the specified choice of L. The left panel in Figure 4 shows the percentage breakdown of the memory consumed by the objects involved in GPU-MWDG2 simulations: the GPU-MWDG2 non-uniform grid, the explicitly deep-copied neighbours of each cell in the grid, and the hierarchy of grids involved in the dynamic GPU-MWDG2 adaptivity process (overviewed in Appendix A). It can be seen that 15% of the memory is allocated for arrays that store the shape coefficients of the cells in the hierarchy of uniform grids, while another 6% is allocated for other miscellaneous purposes. Remarkably however, nearly 80% of the overall GPU memory costs come from elsewhere: 22% from arrays storing the shape coefficients of the cells comprising the non-uniform grid, and 57% from arrays storing explicit deep-copies of the neighbouring shape coefficients of each cell (i.e. north, east, south, west). This high memory consumption using deep-copies ensures coalesced memory access when computing the DG2 solver updates (Appendix A). Furthermore, the GPU-MWDG2 solver is coded to allocate GPU memory for the worst-case scenario where there is no grid coarsening at all, thereby negating the need for memory reallocation after any coarsening to maximise the efficiency of dynamic GPU-MWDG2 adaptivity, since memory allocation is a relatively slow operation.

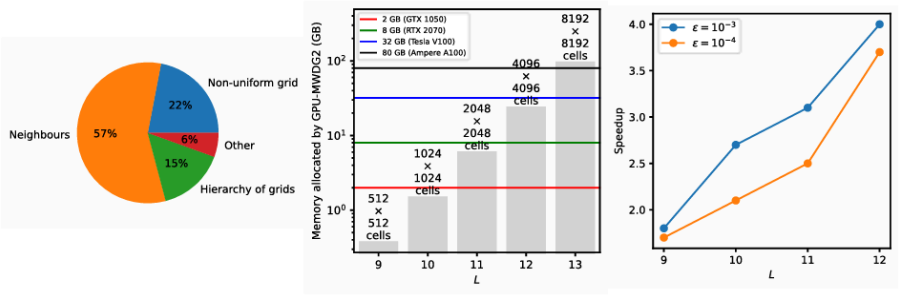

Figure 4: GPU memory consumed by dynamic GPU-MWDG2 adaptivity. Left panel shows the percentage breakdown of the GPU memory consumed by the different objects involved in the GPU-MWDG2 solver. Middle panel shows the amount of GPU memory

5- line 189: "arrays for non-uniform grid". Presumably these are the DG coefficients of states and parameters, are they not? The wording suggests you store "grid" information, no the actual model states and parameters.

Yes, we have clarified this in the immediately above comment.

6- line 190: why is it necessary to store neighbours "explicitly". What does "explicitly" mean? Does it imply data duplication in your arrays?

Yes, it implies data duplication - this is necessary to ensure coalesced memory access.

each cell (i.e. north, east, south, west). This high memory consumption using deep-copies ensures coalesced memory access when computing the DG2 solver updates (Appendix A). Furthermore, the GPU-MWDG2 solver is coded to allocate GPU memory for the worst-case scenario where there is no grid coarsening at all, thereby negating the need for memory

7- Across the different test cases, there is the possibility that the problem size is too small to really exploit the high level of parallelism the A100 GPUs can offer, assuming that parallelism is mostly achieved through parallel processing of elements. If I have understood properly, the highest resolution grids have the following approximate number of cells:

- 380k Monai. Surely very small for the A100.

- 2.4M Seaside Oregon. Seems ok.

- 9M Tauranga Harbour. Seems ok.

- 485k Hilo Harbour. Surely very small for the A100.

The adaptive grids, in turn, as desired, significantly reduce the number of elements to the following approx. maximums:

- 60k Monai

- 230k Oregon

- 3.6M Tauranga

- 195k Hilo

With the exception of Tauranga, it seems these grids will not fully exploit the GPU parallelism.

I don´t think this is a fundamental issue. Nevertheless, the performance analysis seems not to take it into consideration at all. To at least some degree the low arithmetic load of the GPU is likely to explain the large gap between the ratio of cells and the speed-up. The statements in lines 479-481 further seem to support that, for example, the Monai case is simply too small of a problem, since larger cases actually yield a better behaviour.

We now touch on this in the introduction and in Sect. 2. This is a natural comment to make, but in this performance analysis, we are more interested in the *relative* performance of the adaptive solver vs the uniform solver rather than the *absolute* GPU performance of the adaptive solver. Otherwise, and understandably so for a GPU-related problem, discussion on issues like arithmetic intensity, memory-/compute-boundedness, etc, would have been more relevant.

Introduction.

> CPU and was based on the reference DG2 hydrodynamic solver in Kesserwani et al. (2018). Chowdhury et al. (2023) devised a computationally efficient GPU implementation of wavelet adaptivity integrated with a first-order finite volume (FV1) solver counterpart. They reported that the speedup of their GPU implementation over the uniform FV1 solver run on the finest $2^L \times 2^L$ resolution grid scaled up with increasing L, becoming considerable for $L \geq 9$. In fact, for $L \geq 9$, with the same global timestep Δt used, there would be enough memory and compute workload to bound the GPU, which the adaptive solver can reduce unlike the uniform solver. Kesserwani & Sharifian (2023) extended the GPU implementation of

Sect. 2.

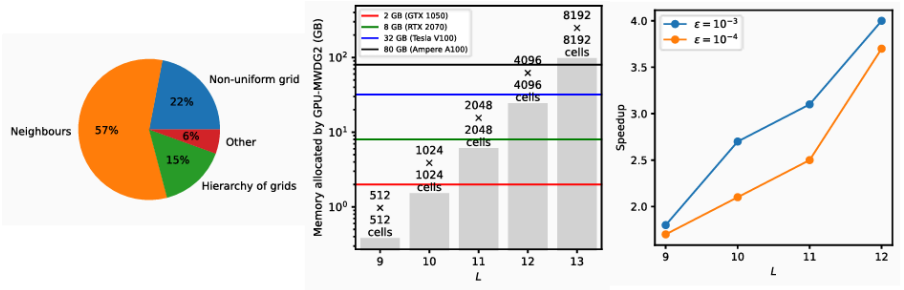

Figure 4: GPU memory consumed by dynamic GPU-MWDG2 adaptivity. Left panel shows the percentage breakdown of the GPU allocated against the maximum refinement level L; the numbers on top of the bars show the number of cells for a given value of L. The horizontal lines indicate the memory limits of four GPU cards. Right panel shows the scalability of the GPU-MWDG2 solver (i.e. an increasing speedup against an increasing workload) by running GPU-MWDG2 and GPU-DG2 simulations against increasing values of L for the "Monai valley" test case (Sect. 3.1) and plotting the speedups of the GPU-MWDG2 simulations compared to the GPU-DG2 simulations.

8- L344, Concerning the reduced dt which is explained by wetting and drying. This is pretty obscure I get why there's a more frequent and aggressive grid coarsening with epsilon=1E-3, but why does this impact "wetting and drying"? How does this impact on wetting and drying result in a reduced time step? Why do you discard CFL-dominated time step sizes, say at shocks as the reason? How are wet/dry fronts treated? Are they not refined to the highest resolution? If they are, why does the time-step size restriction associated to the wetting/drying fronts yield smaller time steps?

We understand this was obscure and this has been clarified now as CFL-dominated timesteps.

flow begins. This slight drop in Δt is likely caused by the existence of planar DG2 solutions per cell with either partially wet portions or thin water layers under friction effects that locally and temporarily triggers smaller-scale timesteps. Either of these are more likely to be present in coarser cells, thereby reducing Δt with $\varepsilon = 10^{-3}$. Due to the smaller Δt at $\varepsilon = 10^{-3}$, the

*8b- later in L411 you mention "wet*dry fronts at coarse cells". This seems counter intuitive... why would you keep coarse cells at a wet/dry front? Isn't this precisely the time of feature you would like to capture? Moreover, it seems to behave poorly, since it requires an aggressive time step reduction. How is it advantageous to keep wet/dry fronts at coarser levels?*

Please see the immediately above comment where we discuss this - we do not keep coarse cells at the wet/dry front.

9- L347-348, regarding the computational effort of MRA. It seems rather important to highlight that Figure 7 shows that R_MRA is significantly larger than R_DG2 and C_MRA is significantly larger than C_DG2. Please comment on this. Earlier in the text you highlight that one would ideally expect a close correlation between the reduction in number of elements (relative to the fine grid) and the speed up. This is obviously not the case, as a reduction to ~16% of elements would imply a speed up in the order of 6x. Therefore the overhead of MRA here is overwhelming, and costs you 3x of speed up. Similar points can be made for the Oregon case, with 10% of the cells, but speed ups below 5, at best.

We now mention that the computational overhead of the MRA may be very large and may even dominate over the DG2 solver updates.

This test case and the previous show that the GPU-MWDG2 solver can achieve at least a 2-fold speedup over the GPU-DG2 solver for tsunami simulations involving a single-wave impact event. Notably though, if the impact event had been present for a larger fraction of the simulation time, it would have introduced a correspondingly higher amount of complexity over time, and a higher cumulative computational effort would have been expected. This type of event will be explored next in Sects. 3.3 and 3.4.

We also mention that this same overhead may detract from achieving the ideal speedup.

2.3 Proposed metrics for analysing GPU-MWDG's runtime efficiency

Assessing the speedup that could be afforded by GPU-MWDG2 adaptivity over a GPU-DG2 simulation is essential. As noted in other works that explored wavelet adaptivity, the computational effort and speedup of a GPU-MWDG2 simulation should ideally correlate with the number of cells in the GPU-MWDG2 non-uniform grid. This correlation is expected since the number of cells dictates the number of DG2 solver updates to be performed. However, this rarely occurs in practice as the ideal speedup is diminished by the additional computational effort spent by GPU-MWDG2 in generating the non-uniform grid every timestep via the MRA process (Chowdhury et al., 2023; Kesserwani et al., 2019; Kesserwani &

10- Line 420-422. This text might be a bit unintentionally deceptive. What would happen if you were to measure cumulative times at t=24 when the flow is really complex? From your own arguments one can say that during the "simple" flow stages (as the incoming wave propagates through a smooth and open bathymetry) you get a speed -up. You later lose a lot of that speed up because of the wet/dry limited time step, thus, the adaptivity at eps=1E-3 is self-damaging.

If you measure Ctot from t=24, is the speed up still larger than 1?

The rationale for this question is this: the fraction of time in this problem with complex flows (the ones that eps=1E-3 doesn't like) is relatively small compared to the rest. What if it were the opposite, i.e., 95% of the time you had complex flows which eps=1E-3 doesn't like?

Yes, we now explicitly spell this out now in the relevant discussion.

This test case and the previous show that the GPU-MWDG2 solver can achieve at least a 2-fold speedup over the GPU-DG2 solver for tsunami simulations involving a single-wave impact event. Notably though, if the impact event had been present for a larger fraction of the simulation time, it would have introduced a correspondingly higher amount of complexity over time, and a higher cumulative computational effort would have been expected. This type of event will be explored next in Sects. 3.3 and 3.4.

11- L 426 "all showing a good agreement with the measured time series". There's some room for debate on this statement. In A1 the peak of h+z is underestimated, it is overestimated in B6, and in D4 it is inaccurate and out of phase. What is true is that the adaptive solution is very similar to the non-adaptive solution. I understand this is the key point for the authors. It is nevertheless questionable is if the non-adaptive solution is an accurate-enough reference. This poses questions on WHY the non-adaptive solution does not capture these clear features. Specifically, could it be related to resolution? Would a higher resolution solve the discrepancies? If yes, why not try?

We now acknowledge the differences, although upscaling the DEM resolution would require L = 13 which is beyond the memory limits.

In Figure 11, the time series predicted by GPU-MWDG2 using $\varepsilon = 10^{-3}$ and 10^{-4} and GPU-DG2 of the prognostic variables $h + z$, the u component of the velocity, and the derived momentum variable $M_x = 0.5hu^2$ at points A1, B6 and D4 are shown. Point A1 is one of the left-most crosses in Figure 9, point B6 is one of the central crosses, and point D4 is one of the right-most crosses. At all three points, the predictions are visually similar to each other except at point D4, where, for $\varepsilon = 10^{-3}$, the impact wave height is slightly overpredicted, while the velocity and momentum is overpredicted between 33 s and 43 s. Compared to the measured data, the predictions all follow the trailing part of the measurement time series quite well, but at the peaks, they are underpredicted at point A1, overpredicted at point B6, and out of phase at point D4. These differences, which manifest most noticeably at the peaks, may be due to the incoming bore of the impact wave, which had invalidated measurements and required repeating experiments, possibly introducing uncertainty into the measured data (H. Park et al., 2013).

12- Figure 13. What are the dashed lines in the Ncells plot?

This has been clarified in the caption:

Figure 13: Tauranga Harbour. Measuring the computational performance of the GPU-MWDG2 and GPU-DG2 simulations using the metrics of Table 1 to assess the speedup afforded by GPU-MWDG2 adaptivity: N_{cells},the number of cells in the GPU-MWDG2 non-uniform grid,where the dashed line represents the initial value at the beginning of the simulation; R_{DG2},the computational effort of performing the DG2 solver updates at given timestep relative to the GPU-DG2 simulation; R_{MRA}, the relative computational effort of performing the MRA process at a given timestep; S_{inst}, the instantaneous speedup achieved by the GPU-MWDG2 simulation over the GPU-DG2 simulation at a given timestep; Δt, the simulation timestep; $N_{\Delta t}$, the number of timesteps taken to reach a given simulation time; C_{DG2}, the cumulative computational effort of performing the DG2 solver updates up to a given simulation time; C_{MRA}, the cumulative computational effort of performing the MRA process; C_{tot}, the total cumulative computational effort; S_{acc}, the accumulated speedup of GPU-MWDG2 over GPU-DG2 up to a given simulation time.

13- Lines 539-541. You comment that the dt history is very similar, unilke previous cases. What is different in this case? Why is the dt not affected by the different adaptive thresholds like in other cases?

This has been clarified now.

drops in R_{DG2} and R_{MRA}. Unlike all of the previous test cases (Sects. 3.1 - 3.3), the time history of Δt is very similar between the GPU-DG2 simulation and the GPU-MWDG2 simulations, regardless of the ε value as they both yield high numbers of cells compared to the uniform GPU-DG2 grid, leading to highly refined non-uniform grids that are unlikely to have coarse cells compared to the previous test cases. Thus, the time history of $N_{\Delta t}$ is virtually identical across all simulations. Given that

14- In Figure 16, despite the small problem size, RDG2 is larger than RMRA, which suggests that the complex dynamics dominate. This is relevant, I believe.

This occurs due to complex flows leading to a highly refined grid, therefore leading to R_{DG2} dominating R_{MRA}.

cells compared to the uniform GPU-DG2 grid, leading to highly refined non-uniform grids that are unlikely to have coarse cells compared to the previous test cases. Thus, the time history of $N_{\Delta t}$ is virtually identical across all simulations. Given that the time histories of $N_{\Delta t}$, R_{DG2} and R_{MRA} are similar for both ε values, the time histories of C_{DG2} and C_{MRA} are also very similar, but importantly, due to the complex flows leading to highly refined non-uniform grids, the former dominates the latter in terms of the total computational effort. Overall, the time history of C_{tot} is very close between the GPU-DG2

15 - L592: "the DEM area correspons to a choice for L >= 9". This is not exactly what the results show.

Yes, we have changed our conclusion to reflect this.

In summary, the GPU-MWDG2 solver in LISFLOOD-FP 8.2 consistently accelerates GPU-DG2 simulations of rapid multiscale flooding flows, generally yielding the greatest speedups for simulations needing $L \geq 10$—due to its scalability on the GPU which results in larger speedup with increasing L—and driven by single-peaked impact events. Choosing ε closer to 10^{-3} would maximise speedup while choosing an ε closer 10^{-4} may be useful to particularly improve the velocity-related predictions. The LISFLOOD-FP 8.2 code and the simulated benchmarks' set-up files and datasets are

16- L953-598. What arguably matters is how long the dynamics are complex, not the fact that there are multiple peaks. You could have a very short simulation around a single peak and get a high density of complex dynamics during a short time. This is effectively what happens in the Tauranga case, as shown in the Sacc plot in figure 13.

Yes, we now clarify this.

Speedup assessments, based on instantaneous and cumulative metrics, suggested considerable gain when the DEM size and resolution involved $L \geq 10$ and for simulation durations that mostly spanned reduced-complexity events (i.e. single-peak tsunamis): As shown in Table 7, for single-peaked tsunamis, when the DEM required $L = 10$, GPU-MWDG2 was more than 2-fold faster than the GPU-DG2 simulations (i.e. "Monai Valley"), whereas with the DEM requiring $L = 12$, speedups of 3.3-to-4.5-fold could be achieved (i.e. "Seaside, Oregon"); in contrast, for multi-peaked tsunamis, GPU-MWDG2 speedups reduced to 1.8-fold for a DEM requiring $L = 12$ (i.e. "Tauranga Harbour") and to 1.2-fold for a smaller DEM needing $L = 10$ (i.e. "Hilo Harbour").

17- In order to disentangle the two (apparently) key conclusions related to point 15 and 16: wouldn't it make sense to run the same test with larger L? In particular smaller cases, such as the Monai case? I understand it is all limited by memory, but memory limitations seem far for the Monai case.

We have now included scalability results in Sect. 2 to showcase this point as well as modified the conclusion to reflect this concept.

Sect. 2.

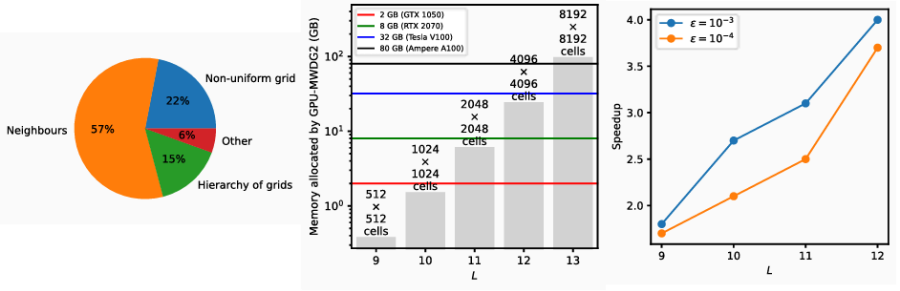

Figure 4: GPU memory consumed by dynamic GPU-MWDG2 adaptivity. Left panel shows the percentage breakdown of the GPU

allocated against the maximum refinement level L; the numbers on top of the bars show the number of cells for a given value of L. The horizontal lines indicate the memory limits of four GPU cards. Right panel shows the scalability of the GPU-MWDG2 solver (i.e. an increasing speedup against an increasing workload) by running GPU-MWDG2 and GPU-DG2 simulations against increasing values of L for the "Monai valley" test case (Sect. 3.1) and plotting the speedups of the GPU-MWDG2 simulations compared to the GPU-DG2 simulations.

Conclusion.

Speedup assessments, based on instantaneous and cumulative metrics, suggested considerable gain when the DEM size and resolution involved $L \geq 10$ and for simulation durations that mostly spanned reduced-complexity events (i.e. single-peak tsunamis): As shown in Table 7, for single-peaked tsunamis, when the DEM required $L = 10$, GPU-MWDG2 was more than 2-fold faster than the GPU-DG2 simulations (i.e. "Monai Valley"), whereas with the DEM requiring $L = 12$, speedups of 3.3-to-4.5-fold could be achieved (i.e. "Seaside, Oregon"); in contrast, for multi-peaked tsunamis, GPU-MWDG2 speedups reduced to 1.8-fold for a DEM requiring $L = 12$ (i.e. "Tauranga Harbour") and to 1.2-fold for a smaller DEM needing $L = 10$ (i.e. "Hilo Harbour").

Table 7: Summary of GPU-MWDG2 runtimes and potential average speedups with respect to GPU-DG2.

				GPU-MWDG2				GPU-DG2
	Tsunami (impact) event		L (*square uniform grid*)	Runtime ε		Speedup ε		Runtime
Test case	t_{end}	Single-wave tsunami	----	10^{-3}	10^{-4}	10^{-3}	10^{-4}	----
Monai Valley	22.5 (9000 s*)	Yes	10 ($2^{10} \times 2^{10}$)	16 s	20 s	2.5	2.0	40 s
Seaside Oregon	40 s (33 min*)	Yes	12 ($2^{12} \times 2^{12}$)	3.5 min	5.2 min	4.5	3.3	13 min
Tauranga Harbour	40 hr	No (three peaks)	12 ($2^{12} \times 2^{12}$)	7.5 hr	8.1 hr	1.8	1.4	11.3 hr
Hilo Harbour	6 hr	No (eleven peaks)	10 ($2^{10} \times 2^{10}$)	5.3 min	5.8 min	1.3	1.2	6.9 min

*By accounting for the physical scaling factor of the replica.

In summary, the GPU-MWDG2 solver in LISFLOOD-FP 8.2 consistently accelerates GPU-DG2 simulations of rapid multiscale flooding flows, generally yielding the greatest speedups for simulations needing $L \geq 10$—due to its scalability on the GPU which results in larger speedup with increasing L—and driven by single-peaked impact events. Choosing ε closer to 10^{-3} would maximise speedup while choosing an ε closer 10^{-4} may be useful to particularly improve the velocity-related predictions. The LISFLOOD-FP 8.2 code and the simulated benchmarks' set-up files and datasets are

18- Table 7: what is "max cells"?

It is the size of the "square uniform grid" at L.

Table 7: Summary of GPU-MWDG2 runtimes and potential average speedups with respect to GPU-DG2.

				GPU-MWDG2				GPU-DG2
	Tsunami (impact) event		L (*square uniform grid*)	Runtime ε		Speedup ε		Runtime
Test case	t_{end}	Single-wave tsunami	----	10^{-3}	10^{-4}	10^{-3}	10^{-4}	----
Monai Valley	22.5 (9000 s*)	Yes	10 ($2^{10} \times 2^{10}$)	16 s	20 s	2.5	2.0	40 s
Seaside Oregon	40 s (33 min*)	Yes	12 ($2^{12} \times 2^{12}$)	3.5 min	5.2 min	4.5	3.3	13 min
Tauranga Harbour	40 hr	No (three peaks)	12 ($2^{12} \times 2^{12}$)	7.5 hr	8.1 hr	1.8	1.4	11.3 hr
Hilo Harbour	6 hr	No (eleven peaks)	10 ($2^{10} \times 2^{10}$)	5.3 min	5.8 min	1.3	1.2	6.9 min

*By accounting for the physical scaling factor of the replica.

19- I would suggest to get rid of lines 605-608, as they appear in the code and data availability statement.

Yes, we have done this.