ADAPTIVE LOCAL TIME STEPPING WITH RKDG2 WATER WAVE MODEL ON NON-UNIFORM GRIDS

Georges Kesserwani[1] & Qiuhua Liang[2]

[1]Department of Civil & Structural Engineering, University of Sheffield, UK, Sheffield S1 3JD
[2]School of Civil Engineering & Geosciences, Newcastle University.UK, Newcastle upon Tyne NE1 7RU
E-mail: g.kesserwani@shef.ac.uk; qiuhua.Liang@ncl.ac.uk

Abstract

We investigate explicit local RKDG2 (second-order Runge-Kutta Discontinuous Galerkin) solutions with the inhomogeneous SWEs (Shallow Water Equations) on non-uniform meshes with local time steps. The incorporated LTS (Local Time Step) algorithm was recently designed and tested for homogenous hyperbolic PDE(s) and was featured by its locality and second-order accuracy Two LTS-RKDG2 schemes that adapt three and four levels of LTSs are configured based on two adaptive meshes that, respectively, adapt two and three levels of refinement. Hydraulic tests are used to verify the LTS-RKDG2 schemes by comparing their performance with their conventional Global Time Step RKDG2 alternatives (GTS-RKDG2). Our results show that the LTS-RKDG2 models produce similar predictions as the GTS-RKDG2 models but with less computational effort reduced by a factor of 1.3 to 2.3 times depending on the test case.

Introduction

Explicit finite volume (FV) Godunov-type numerical methods that solve hyperbolic conservation laws of the unsteady SWEs (Toro, 2001; Guinot, 2003) are widely relevant to water flow simulations and have been receiving numerous developments (Delis & Kampanis, 2009). To summarize, a '*robust*' Godunov-type SWEs numerical solver should be able to maintain its stability and consistency when a flow discontinuity develops, steep topographic gradients are present, a wet/dry front occurs, and high roughness values are combined with very small water depths. However all these advances, it is still imperative to enhance the runtime of these explicit FV models, which may be done by using a non-uniform adapted mesh and increasing the time step.

From this perspective, it is expected that the efficiency of an explicit numerical scheme may suffer as the size of their time steps is restricted by the CFL stability condition. This criterion provides the maximum allowable Global Time Step (GTS) permitted, which actually reduces as a result of a local increase in the velocity magnitude or a local decrease in the cell size, or both; see Eq. (5). Particularly when using non-uniform grids, few smallest cells may impose a restrictive time step on the whole mesh and therefore improvements in accuracy, gained by local mesh refinement, are compensated by longer runtimes. In such a circumstance, a local time step method (LTS) whereby the solution within different cells is advanced by different time steps seems complementary to increase the computational efficiency of an explicit numerical model that uses non-uniform meshes.

Very few published papers dealt with the design and implementation of LTS algorithms with Godunov-type water wave models. Crossley & Wright (2005) first transplanted the concept of LTS into the field of 1D hydrodynamic modelling showing a merit not only in reducing runtimes but also in augmenting the quality of the numerical solution. Sanders (2008) later investigated this topic with a robust 2D Godunov-type SWEs solver based on applications involving frictional flows over irregular topographies with wetting and drying. Moreover, the implicit friction term discretization (IFTD), which is a commonly used practice to stabilize water flow simulations with wetting and drying, was reported conflicting when it is activated with a LTS algorithm. In both investigations, LTS algorithms were integrated with first-order FV water models and the use of five, or more, levels of LTS was discouraged.

This work extends a LTS algorithm into RKDG2 Godunov-type SWEs solutions (LTS-RKDG2) on non-uniform meshes. The considered LTS algorithm was recently established, by Krivodonova (2010), and found to be well-fit with the RKDG2 scheme (Kesserwani & Liang, 2012). It preserves second-order accuracy, conserves locality, applies straightforwardly to the coefficients of the local finite element solution and adapts the time step on each cell according to the cell size (i.e., it takes a LTS of Δt on cells of size Δx, $\Delta t/2$ on cells of size $\Delta x/2$, *etc*). In Krivodonova (2010) the LTS algorithm was verified theoretically in combination with an RKDG2 scheme (LTS-RKDG2). However, simulation results were only presented for homogenous conservation laws, excluding the SWEs.

Further, no information was provided on the relative gain in terms of efficiency. Herein, the Krivodonova LTS algorithm is reformulated and improved in the framework of the 1D RKDG2 scheme solving the SWEs with complex source terms and involving wetting and drying (Kesserwani & Liang, 2012). Using steady and transient tests, the implementation of two LTS-RKDG2 shallow water solvers, that respectively employ three and four LTSs, is verified and the potential improvement in efficiency relative to the associated conventional GTS-RKDG2 is quantified.

Shallow Water Equations (SWEs)

The mathematical model of the SWEs can be written in a conservative matrix form

$$\partial_t \mathbf{U} + \partial_t \mathbf{F}(\mathbf{U}) = \mathbf{S}(\mathbf{U}) \quad (1)$$

In which, x is the longitudinal coordinate and t is the time. $\mathbf{U} = [\eta, q_x]^{\mathrm{T}}$; $\mathbf{F} = [q_x, q_x^2 h^{-1} + 0.5(\eta^2 - 2\eta z)]^{\mathrm{T}}$ and $\mathbf{S}$ is transposed vector containing the source terms. The source term vector $\mathbf{S}$ can be further partitioned into $\mathbf{S} = \mathbf{S_b} + \mathbf{S_f}$ where $\mathbf{S_b} = [0, -g\eta\partial_x z]^{\mathrm{T}}$ and $\mathbf{S_f} = [0, -C_f u|u|]^{\mathrm{T}}$ are, respectively, the topography and friction source terms (g is the acceleration due to gravity, z is the topography, $h = \eta - z$ is the water depth, $u = q_x/h$ the mean velocity and $C_f = g n_M^2 h^{-1/3}$; n_M being the Manning coefficient).

Non-uniform structured mesh

Firstly, a problem domain [x_{min}; x_{max}] is discretized using a coarse uniform mesh consisting of N cells of size Δx. This is called "*background mesh*", on which a cell is termed "*background cell*" and is assigned the minimum level of subdivision, *i.e.*, equal to '0'. Secondly, background cells in those user-selected local zone(s) where mesh refinement is desired are further refined by specifying higher levels of subdivisions up to a user-specified maximum subdivision level 'lev_{max}' ($lev_{max} \in \mathbb{N}$). The refinement is performed in a fractal manner, *i.e.* the cell size reduces by a factor of two whenever the refinement level increases '1'. Finally, the mesh is regularized so that it does not contain adjacent cells with sizes differing by more than a factor of two.

Overall, mesh will consist of cells with levels varying between '0' and 'lev_{max}'. Cells with level '0' are the largest whereas cells with level 'lev_{max}' are the smallest. An arbitrary cell I_{ic} can be classified through an assigned level denoted by '$lev(ic)$' (where $0 \le lev(ic) \le lev_{max}$) and thereby has a size length of $\Delta x_{ic} = \Delta x/2^{lev(ic)}$ (with $\Delta x/2^{lev_{max}} \le \Delta x_{ic} \le \Delta x$). Cell I_{ic} is centered at x_{ic} with the boundary points $x_{ic\pm1/2} = x_{ic} + \Delta x_{ic}/2$, *i.e.* $I_{ic} = [x_{ic-1/2}; x_{ic+1/2}]$.

Review of the GTS-RKDG2 scheme

Over a cell 'I_{ic} ', the RKDG2 method solves for a *local* approximate (piecewise linear) solution to (1), denoted by $\mathbf{U_h} = [\eta_h, (q_x)_h]^{\mathrm{T}}$. The solution is locally spanned by the average and slope coefficients (see within Kesserwani & Liang (2012) for explicit details), i.e. $\mathbf{U_h}(x,t)|_{I_{ic}} = \{\mathbf{U}_{ic}^0(t), \mathbf{U}_{ic}^1(t)\}$

$$\mathbf{U_h}(x,t)|_{I_{ic}} = \mathbf{U}_{ic}^0(t) + \mathbf{U}_{ic}^1(t)\left(\frac{x - x_{ic}}{\Delta x_{ic}/2}\right), \quad (\forall x \in I_{ic}) \quad (2)$$

Given the initial conditions $\mathbf{U}(x,0)$, the local coefficients, $\mathbf{U}_{ic}^0(0)$ and $\mathbf{U}_{ic}^1(0)$, can be initialized. The topography function is approximated similarly, and therefore becomes $\mathrm{z}_h(x)|_{I_{ic}}$, which is spanned by local topography-associated coefficient, z_{ic}^0 and z_{ic}^1. The bed gradient thus writes $\partial_x[\mathrm{z}_h(x)|_{I_{ic}}] = 2\,\mathrm{z}_{ic}^1/\Delta x_{ic}$.

Two stages Runge-Kutta Time-stepping

The time updating from 't' to '$t + \Delta t$' is performed by employing a two-stage RK time stepping method (Shu & Osher, 1988). Denoting by $\left(\mathbf{U}_{ic}^{0,1}\right)^n$ and $\left(\mathbf{U}_{ic}^{0,1}\right)^{n+1}$ to be the solution coefficients at time 't', and '$t + \Delta t$', respectively, the two-stage RK time integration process can be written as

$$\left(\mathbf{U}_{ic}^{0,1}\right)^{n+1/2} = \left(\mathbf{U}_{ic}^{0,1}\right)^n + \Delta t\left(\mathbf{L}_{ic}^{0,1}\right)^n \quad (3)$$

$$\left(\mathbf{U}_{ic}^{0,1}\right)^{n+1} = \frac{1}{2}\left[\left(\mathbf{U}_{ic}^{0,1}\right)^n + \left(\mathbf{U}_{ic}^{0,1}\right)^{n+1/2} + \Delta t\left(\mathbf{L}_{ic}^{0,1}\right)^{n+1/2}\right] \quad (4)$$

At the first RK stage (3), referred to as RK1, the local solution $\left(\mathbf{U}_{ic}^{0,1}\right)^n$ is advanced to an intermediate state $\left(\mathbf{U}_{ic}^{0,1}\right)^{n+1/2}$ relative to '$t^* = t + \Delta t/2$'. Then at the second RK stage (4), denoted as RK2, the solution is marched from the intermediate state at 't^*' to the next time level '$t + \Delta t$'. The DG2 space operators $\left(\mathbf{L}_{ic}^{0,1}\right)^n$ and $\left(\mathbf{L}_{ic}^{0,1}\right)^{n+1/2}$ are evaluated from the solution coefficients at t and t^*, respectively, as we describe in the following. The global time step (GTS) Δt is restricted by the CFL condition with a Courant number of 0.3 (Cockburn & Shu, 2001).

$$\Delta t = min\left[\frac{\Delta x_{ic}}{\left|\left(u_{ic}^{0,1}\right)^n\right| + \sqrt{g\left(h_{ic}^{0,1}\right)^n}}\right] \quad (5)$$

It is obvious from (5) that $\Delta t \to 0$ when $lev_{max} \to \infty$).

Local DG2 space operators

The update of the approximate solution $\mathbf{U_h}(x,t)|_{I_{ic}} = \{\mathbf{U}_{ic}^0(t), \mathbf{U}_{ic}^1(t)\}$ performs via a detached set of ODEs:

$$\begin{aligned} \partial_t \mathbf{U}_{ic}^0(t) &= \mathbf{L}_{ic}^0\left(\mathbf{U}_{ic}^{0,1}, \mathbf{U}_{in}^{0,1}\right) \\ \partial_t \mathbf{U}_{ic}^0(t) &= \mathbf{L}_{ic}^1\left(\mathbf{U}_{ic}^{0,1}, \mathbf{U}_{in}^{0,1}\right) \end{aligned} \quad (6)$$

in which, $\mathbf{L}_{ic}^0$ and $\mathbf{L}_{ic}^1$ are nonlinear vectors of space-functions. The approximating coefficients with subscripts 'in' refers to data relative to the neighbor cells I_{in} surrounding cell I_{ic}. These operators can be manipulated to

$$\mathbf{L}_{ic}^0 = -\frac{\tilde{\mathbf{F}}_{ic+1/2} - \tilde{\mathbf{F}}_{ic-1/2}}{\Delta x_{ic}} + \mathbf{S}(\mathbf{U}_{ic}^0, \mathrm{z}_{ic}^0, \mathrm{z}_{ic}^1) \quad (7)$$

$$\mathbf{L}_{ic}^1 = -\frac{3}{\Delta x_{ic}}\left\{\tilde{\mathbf{F}}_{ic+1/2} + \tilde{\mathbf{F}}_{ic-1/2} - \mathbf{F}\left(\mathbf{U}_{ic}^0 + \frac{\tilde{\mathbf{U}}_{ic}^1}{\sqrt{3}}, \mathrm{z}_{ic}^0, \mathrm{z}_{ic}^1\right) - \mathbf{F}\left(\mathbf{U}_{ic}^0 - \frac{\tilde{\mathbf{U}}_{ic}^1}{\sqrt{3}}, \mathrm{z}_{ic}^0, \mathrm{z}_{ic}^1\right) - \Delta x_{ic}\frac{\sqrt{3}}{6}\left[\mathbf{S}\left(\mathbf{U}_{ic}^0 + \frac{\tilde{\mathbf{U}}_{ic}^1}{\sqrt{3}}, \mathrm{z}_{ic}^0, \mathrm{z}_{ic}^1\right) - \mathbf{S}\left(\mathbf{U}_{ic}^0 - \frac{\tilde{\mathbf{U}}_{ic}^1}{\sqrt{3}}, \mathrm{z}_{ic}^0, \mathrm{z}_{ic}^1\right)\right]\right\} \quad (8)$$

When evaluating (7)-(8), a number of essential *spatial* ingredients should be implemented to maintain the stability of the numerical method and grant its practicability to shallow flow modelling. Firstly, *local* slope coefficients that likely cause numerical instability should be detected

and locally limited by the *minmod* function. The "*hat*" symbol above $\mathbf{U}_{ic}^{1}$ refers to the controlled slope coefficient (i.e., $\widehat{\mathbf{U}}_{ic}^{1}$). Secondly, the flux $\tilde{\mathbf{F}}_{ic+1/2}$ across an interface $x_{ic+1/2}$ (shared by adjacent cells I_{ic} and I_{in} with $in = ic+1$) is obtained via the HLL Riemann problem solution of the two states Thirdly, before the final evaluation of (7) and (8), it is important to further implement a conservative "*wetting and drying condition*" to ensure the positivity of the water height with time evolution. Lastly, the friction source term $\mathbf{S_f}$ should be separately discretized (i.e. not within (7) and (8)) by implicit discretization technique to avoid numerical instabilities that may arise when modelling water flow over dry zone with high roughness Details on how these spatial ingredients are implemented can be found in Kesserwani & Liang (2012). To ease presentation in what follows, the approximating coefficients with subscripts '*in*' will refer to the data relative to the *eastern* neighbor (i.e., cell I_{in}) of cell I_{ic} and so $\{x_{ic+1/2}\} = I_{ic} \cap I_{in}$ represents the edge separating cells I_{ic} and I_{in}.

New LTS-RKDG2 flow model

The second-order LTS approach of Krivodonova (2010) is integrated with the (GTS-)RKDG2 flow model (Kesserwani & Liang 2012) to from the so-called LTS-RKDG2 water wave model. Further to this, special treatments are implemented to retain conservation and the applicability of the LTS-RKDG2 model to shallow flow simulations over frictional topographies with wetting and drying.

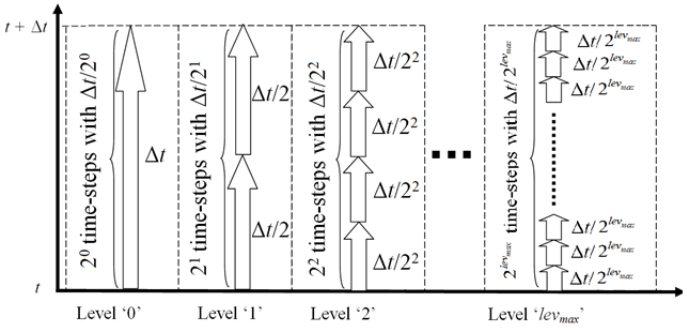

Fig. 1: LTS-RKDG2 calculation of the solution coefficients from 't' to '$t + \Delta t$' on a mesh with multiple levels of refinement '0', ..., 'lev_{max}', where a '***thick arrow***' = one iteration of LTS-RKDG2 calculation.

Basic concept

To simplify presentation, it is assumed that the maximum wave speed does not significantly influence the local *CFL* number and therefore the LTS relative to each cell 'I_{ic}' is solely dependent on its level of refinement *lev*(*ic*) (or cell size $\Delta x_{ic} = \frac{\Delta x}{2^{lev(ic)}}$) so that $\Delta t_{ic} = \frac{\Delta t}{2^{lev(ic)}}$. Δt is the GTS relative to the coarse "*background mesh*" and can be simply defined as

$$\Delta t = min\left[\frac{2^{lev(ic)}\Delta x_{ic}}{\left|\left(u_{ic}^{0,1}\right)^n\right|+\sqrt{g\left(h_{ic}^{0,1}\right)^n}}\right] \quad (9)$$

As illustrates in Fig. 1, the RKDG2 calculation is locally performed with the LTS Δt, $\Delta t/2$, $\Delta t/2^2$, ..., $\Delta t/2^{lev_{max}}$ for cells with level '0', '1', '2', ..., 'lev_{max}' to recursively advance their solution coefficients 1 LTS, 2 LTSs, 4 LTSs,…, $2^{lev_{max}}$ LTSs. Herein, this iterative calculation process is referred to as "LTS-RKDG2 calculation". In the first iteration, the LTS-RKDG2 calculation achieves at cells with level '0' to allow the corresponding solution coefficients to reach time level '$t + \Delta t$'. In the second iteration, the LTS-RKDG2 calculation is undertaken at cells with level '1', and so on, until the finest cells with level 'lev_{max}' are fully updated after $2^{lev_{max}}$ iterations of LTS-RKDG2 calculation. During a simulation, computational cells are grouped according to their level of refinements and the associated mesh may be classified as "*inner cells*" and "*interface cells*". Cells neighbored by cells 'I_{in}' with similar level of refinement are called *Inner cells* 'I_{ic}'; otherwise, they are termed as *Interface cells*.

At an *inner cell* 'I_{ic}', an LTS-RKDG2 calculation is straightforward. Since 'I_{ic}' and 'I_{in}' have the same level of refinement, they thus have the same LTS, *i.e.* $\Delta t_{in} = \Delta t_{ic}$. One step of LTS-RKDG2 (*i.e.*, (3) and (4) with Δt_{ic}) calculation performs in a regular manner as for the GTS-RKDG2 scheme and no special treatment is needed.

An *interface cell* 'I_{ic}' has at least one of its adjacent neighbors of different size *e.g.*, $\Delta x_{in} \neq \Delta x_{ic}$, and thus $\Delta t_{in} \neq \Delta t_{ic}$. This indicates that the LTS-RKDG2 calculation at 'I_{ic}' is different from the one at 'I_{in}'. For that reason, it is essential for LTS-RKDG2 algorithm to impose synchronized '*ghost*' solution coefficients across the cells with different time step(s), or stage(s), to enable Riemann flux calculation within the DG2 operators and allow the RKDG2 calculation to proceed in a classical way as described in the following Subsection.

LTS-RKDG2 calculation at interface cells

Since the mesh is regularized and the LTS-RKDG2 calculation simply applies recurrently, as shown Fig. 1, it suffices to explain one elementary LTS-RKDG2 calculation at *interface cells* relative to a mesh configuration that involves two levels of refinement denoted by '*lev*0' and '*lev*0+1' (*lev*0 is a fixed integer between 0 and $lev_{max} - 1$). Without loss of generality, we assume cell 'I_{ic}' has a '*lev*0' and is defined as a "*large interface cell*" (LIC). Similarly, cell 'I_{in}' has a level of '*lev*0+1' and is referred to as a "*small interface cell*" (SIC). Therefore the solution coefficients at 'I_{ic}' are indicated by a subscript 'L' and those at 'I_{in}' by 'S'.

Firstly, LTS-RKDG2 calculation handles the coefficients at the LIC 'I_{ic}' (*i.e.*, the '*actual*' coefficients) facilitated by artificially reconstructed synchronized coefficients (*i.e.*, '*ghost*' coefficients) at the SIC 'I_{in}'. Then, it is necessary to apply the LTS-RKDG2 calculation to deal with the case where the '*actual*' coefficients are at the SIC 'I_{in}' and the '*ghost*' coefficients are at the LIC 'I_{ic}'.

Coefficients update at the LIC 'I_{ic}'

First, the LTS-RKDG2 calculation starts at the LIC cell 'I_{ic}' where the LTS is $\Delta t_L = \Delta t/2^{lev0}$ and the initial coefficients at time 't' are $(\mathbf{U}_{ic}^{0,1})_L^n$. At its SIC neighbour '$I_{in}$', the initial coefficients $(\mathbf{U}_{in}^{0,1})_S^n$ are also available. Therefore, the DG2 space operators $(\mathbf{L}_{ic}^{0,1})_L^n$ can be evaluated directly and then plugged into (3) to complete the RK1 stage and produce the '*actual*' coefficients $(\mathbf{U}_{ic}^{0,1})_L^{n+1/2}$ relative to the intermediate time state 't^*'. Similarly, time-matching '*ghost*' coefficients $(\mathbf{U}_{in}^{0,1})_{S,ghost}^{n+1/2}$ relative to t^* are constructed at I_{in} by advancing its coefficients using (3) with LTS Δt_L, i.e.

$$(\mathbf{U}_{in}^{0,1})_{S,ghost}^{n+1/2} = (\mathbf{U}_{in}^{0,1})_S^n + \Delta t_L (\mathbf{L}_{in}^{0,1})_S^n \quad (10)$$

After (10), the coefficients are synchronized at 't^*'. The space operators $(\mathbf{L}_{ic}^{0,1})_L^{n+1/2}$ can then be evaluated and used to achieve the RK2 stage in (4) to finally produce the coefficients $(\mathbf{U}_{ic}^{0,1})_L^{n+1}$ associated to '$t + \Delta t_L$'.

Coefficients update at the SIC 'I_{in}'

After obtaining the '*actual*' coefficients at '$t + \Delta t_L$' over cell 'I_{ic}', the SIC 'I_{in}' is now reconsidered to obtain its '*actual*' coefficients at '$t + \Delta t_L$'. However, since the LTS over a SIC is halved (*i.e.*, $\Delta t_S = \Delta t_L/2$), LTS-RKDG2 calculation over 'I_{in}' has to be carried out in two consecutive iterations, starting from the available initial coefficients, $(\mathbf{U}_{in}^{0,1})_S^n$ and $(\mathbf{U}_{ic}^{0,1})_L^n$, at time '$t$' at both SIC '$I_{in}$' and its neighbor LIC '$I_{ic}$'. It should be noted that the '*ghost*' coefficients at SIC 'I_{in}' produced (previously—refer to the previous subsection) using (10) are here inappropriate for the current step and will therefore be ignored. In contrast, at the LIC 'I_{ic}', certain preceding information created by the previous LTS-RKDG2 update for the '*actual*' coefficients over 'I_{ic}' may be saved and reused. In particular, the previously calculated DG2 space operators $(\mathbf{L}_{ic}^{0,1})_L^n$ and $(\mathbf{L}_{ic}^{0,1})_L^{n+1/2}$ at time 't' and time 't^*' are used to define the following quadratic function

$$\emptyset_{ic}^{0,1}(\tau) = (\mathbf{U}_{ic}^{0,1})_L^n + (\mathbf{L}_{ic}^{0,1})_L^n (\tau - t) + \frac{(\mathbf{L}_{ic}^{0,1})_L^{n+1/2} - (\mathbf{L}_{ic}^{0,1})_L^n}{2\Delta t_L}(\tau - t)^2 \quad (11)$$

which will serve to reconstruct '*ghost*' coefficients over the neighbor LIC 'I_{ic}' at an inner fractional-time-step 'τ' and its associated time-stage 'τ^*', namely:

$$[\mathbf{U}_{ic}^{0,1}(\tau)]_{L,ghost}^n = \emptyset_{ic}^{0,1}(\tau); \ \tau \in [t; t + \Delta t_L[\quad (12)$$

$$[\mathbf{U}_{ic}^{0,1}(\tau^*)]_{L,ghost}^{n+1/2} = [\mathbf{U}_{ic}^{0,1}(\tau)]_{L,ghost}^n + \frac{d}{d\tau}\emptyset_{ic}^{0,1}(\tau); \tau^* \in [\tau; t + \Delta t_L[\quad (13)$$

Eqs. (11)-(13) give a second-order accurate data interpolation as confirmed by the theoretical work in [13].

At the first iteration, the coefficients over 'I_{in}' are advanced one LTS from time 't' to time '$t_2 = t + \Delta t_S$'. Initially at time 't', using $(\mathbf{U}_{in}^{0,1})_S^n$ and $(\mathbf{U}_{ic}^{0,1})_L^n$, the space operators $(\mathbf{L}_{in}^{0,1})_S^n$ can be evaluated and then inserted into (3) to achieve the calculation at the RK1 stage and produce $(\mathbf{U}_{in}^{0,1})_S^n$, which are the '*actual*' coefficients at the intermediate time stage $t_1^* = t + \Delta t_S/2$. Meanwhile, over 'I_{ic}', the '*ghost*' coefficients relative to t_1^*, i.e. $(\mathbf{U}_{ic}^{0,1})_{L,ghost}^{n+1/2}$, should be constructed at $\tau = t_1^*$ by means of (11)-(13).

After this, the DG2 space operators $(\mathbf{L}_{in}^{0,1})_S^{n+1/2}$ at time t_1^* can be evaluated and put in (4) to finalize the RK2 stage and produce $(\mathbf{U}_{in}^{0,1})_S^{n+1}$, which are the 'actual' coefficients over I_{in} at time t_2. Meanwhile, the time-matching '*ghost*' coefficients over 'I_{ic}' at t_2, i.e. $(\mathbf{U}_{ic}^{0,1})_{L,ghost}^{n+1}$, should be constructed using (11) and (12) evaluated at $\tau = t_2$.

At the second iteration, the coefficients are reinitialized at 't_2' and another RKDG2 calculation step with the LTS Δt_S is performed to further elevate the coefficients over I_{in} to time level '$t + \Delta t_L$'. That is, both '*actual*' and '*ghost*' coefficients at I_{ic} and I_{in}, respectively, are reinitialized at 't_2' (*i.e.* $(\mathbf{U}_{in}^{0,1})_S^n \leftarrow (\mathbf{U}_{in}^{0,1})_S^{n+1}$ and $(\mathbf{U}_{ic}^{0,1})_L^n \leftarrow (\mathbf{U}_{ic}^{0,1})_{L,ghost}^{n+1}$). The previously employed '*actual*' and '*ghost*' coefficients and their associated DG2 space operators at the inner time stages can be reused (*i.e.*, herein overwritten). Initially at time level 't_2', $(\mathbf{U}_{ic}^{0,1})_L^n$ and $(\mathbf{U}_{in}^{0,1})_S^n$ are synchronized and so the operators $(\mathbf{L}_{in}^{0,1})_S^n$ can be evaluated and then put into (3) to achieve the RK1 stage and produce '*actual*' coefficients $(\mathbf{U}_{in}^{0,1})_S^{n+1/2}$, which now represent the coefficients at the time stage $t_2^* = t_2 + \Delta t_S/2$. Meanwhile, over '$I_{ic}$', time-matching (*i.e.*, relative to t_2^*) '*ghost*' coefficients are reconstructed using (11)-(13) evaluated at $\tau = t_2^*$ to produce $(\mathbf{U}_{ic}^{0,1})_{L,ghost}^{n+1/2}$. Now, $(\mathbf{U}_{ic}^{0,1})_{L,ghost}^{n+1/2}$ and $(\mathbf{U}_{in}^{0,1})_S^{n+1/2}$ are synchronized and can be used to calculate the DG2 operators $(\mathbf{L}_{in}^{0,1})_S^{n+1/2}$ and then inserted into the RK2 stage (4) to finally produce $(\mathbf{U}_{in}^{0,1})_S^{n+1}$, which now represent the coefficients over 'I_{in}' at time '$t + \Delta t_L$'.

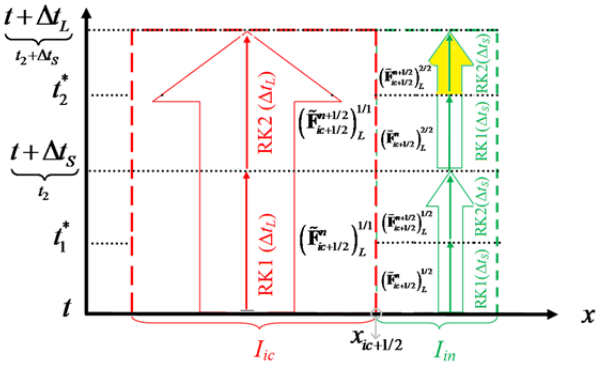

Fig. 2: History of the actual stages of LTS-RKDG2 calculations at a LIC I_{ic} adjacent to a SIC I_{in} and the associated Riemann fluxes. Particular case (when $\Delta t_L = \Delta t$).

Friction term issue and conservation enforcement

Due to the LTS ***dependence*** within the IFTD, its expected side effect on *well-balanced property* (Murillo *et al.*, 2009), may increasingly magnify at those *inner cells*. Further, this aspect complicates the integration of the IFTD with the LTS-RKDG2 calculation at *interface cells* because extra phases of '*ghost*' friction advancement, and removal, need

to be entailed in line with the '*ghost*' coefficients advancement. Therefore, herein, the usability of the IFTD is restricted to those cells where the water height may potentially become infinitesimal. At the other (flood) cells, the friction source term is discretized ***explicitly*** in the DG2 space operators, in a straightforward manner, as it is now free from any LTS dependence.

After the LTS-RKDG2 calculations at the LIC 'I_{ic}' and the SIC 'I_{in}', the sum of Riemann flux quantities cumulated between time 't' and time '$t + \Delta t_L$' at the edge '$x_{ic+1/2}$' may not be equal. For instance, following the notations in Fig. 2, it may happen that the sum of Riemann flux evaluations at '$x_{ic+1/2}$' accumulated from the LTS-RKDG2 calculation at the LIC 'I_{ic}' (*i.e.*, Fig. 2, sum of fluxes with superscript '1/1') is different than the sum of Riemann flux evaluations at '$x_{ic+1/2}$' accumulated during the LTS-RKDG2 calculations at the SIC 'I_{in}' (*i.e.*, Fig. 2, sum of fluxes with superscript '1/2' and '2/2').

To overcome this effect, flux conservation (in time) is artificially enforced at the SIC 'I_{in}' during the *final iteration* of LTS-RKDG2 calculation and, more particularly, at its *final time stage*–namely when the coefficients are awaiting one last step prior to reaching '$t + \Delta t$'. For example, we assume that the mesh only involves two cell's sizes: large cells with the level '0' and small cells with the level '1' (i.e., *lev0* = 0). In this case, $\Delta t_L = \Delta t$ and $\Delta t_S = \Delta t/2$ and flux conservation (in time) is imposed at the SIC 'I_{in}' during the **RK2** ***stage*** and at the ***second round*** (*i.e.*, Fig. 2—right highlighted portion of the thick arrow). This can be done by exceptionally choosing the flux $\left(\tilde{\mathbf{F}}_{ic+1/2}^{n+1/2}\right)_S^{2/2}$ as:

$$\left(\tilde{\mathbf{F}}_{ic+1/2}^{n+1/2}\right)_S^{2/2} = \left[\left(\tilde{\mathbf{F}}_{ic+1/2}^{n} + \tilde{\mathbf{F}}_{ic+1/2}^{n+1/2}\right)_L^{1/1} - \left(\tilde{\mathbf{F}}_{ic+1/2}^{n} + \tilde{\mathbf{F}}_{ic+1/2}^{n+1/2}\right)_S^{1/2} - \left(\tilde{\mathbf{F}}_{ic+1/2}^{n}\right)_S^{2/2}\right] \quad (14)$$

instead of estimating it from the Riemann problem solution.

LTS-RKDG2 model's verification

LTS-RKDG2 and GTS-RKDG2 schemes simulations are run on two non-uniform meshes, referred to as '*mesh 2*' and '*mesh 3*', which have been configured to allow up to '2' and '3' levels of refinement, respectively, while retaining the same total number of computational cell for both meshes. Over these meshes, LTS-RKDG2 local solutions coordinates LTSs of {Δt, $\Delta t/2$, $\Delta t/4$} and {Δt, $\Delta t/2$, $\Delta t/4$, $\Delta t/8$}, respectively. The level of refinement relative to each cell will be indicated by a gray '*diamond*' marker within the sub-figures that illustrate the free-surface elevations. Two tests are considered to investigate the performance of LTS-RKDG2 scheme with respect to the GTS-RKDG2 scheme, while quantifying the relative runtime saving.

Steady flow over a hump with shock

The academic test case involving moving steady transcritical flow over topography, with a shock, is investigated. This test is to simultaneously demonstrate the capability of a numerical method to: converge towards a steady state, accurately balance the flux gradient with the topography gradient, and inherently capture flow transitions and discontinuities. The channel is 1000m length with a hump-shape topography located between x = 125m and x = 875m. An upstream subcritical inflow is imposed through a unit discharge of 20m^2/s and the outflow depth is fixed to 7m. A simulation starts from an initial water height of 9.7m and is desired to stop after a relatively long time evolution (i.e. t = 2000s). Each simulation is performed on a '*mesh 2*' and a '*mesh 3*' type that both consisted of 100 cells.

At first, the channel's bed is assumed frictionless and the GTS-RKDG2 and LTS-RKDG2 schemes are run on the two non-uniform meshes. Fig. 3 present the corresponding profiles acquired by the RKDG2 solvers, respectively. It can be seen that the numerical water depths produced by both solvers match well the analytical solutions and no visual difference is detected among the schemes. On the other hand, both numerical models reached the expected conservative state for the steady discharge solution; this indicates that the current LTS algorithm do not affect the well-balanced property. On '*mesh 2*' and '*mesh 3*', respectively, the runtime saving is about 1.9 and 2.3 times with respect to the GTS-RKDG2 scheme. This shows that the usefulness of the transient LTS-RKDG2 shallow water solver in accelerating the convergence of steady-state problems.

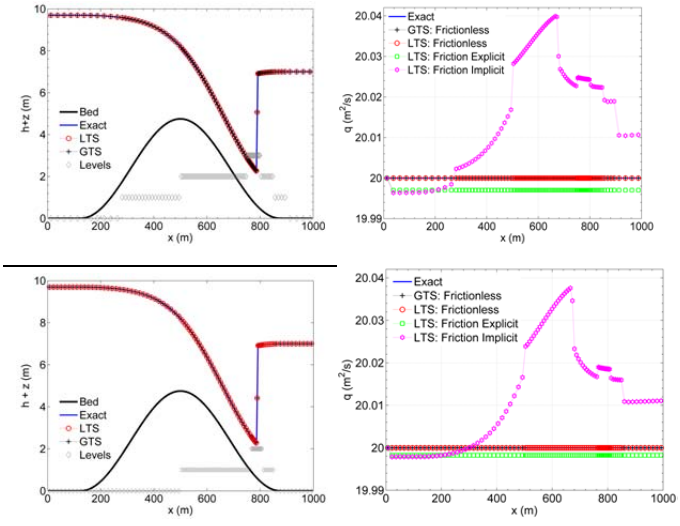

Fig. 3: Steady flow (t = 2000s). Lower: '*mesh* 2'and Upper: '*mesh* 3'.

Secondly, we use this test case to further point up the inconvenience of the IFTD when implemented in conjunction with the LTS-RKDG2 scheme. Therefore, the LTS-RKDG2 method is reconsidered with n_M = 0.033 s/m$^{1/3}$; the simulations are remade on the same meshes but with a focus on comparing the IFTD discretization (i.e., time-dependent) *vs.* the explicit friction term discretization (i.e., independent of the time-step). The solution to the

momentum equation, in terms of steady discharge numerical result, is appended within the discharge plots in Fig. 3. As expected, the use of the IFTD with the LTS-RKDG2 is found to magnify the side effect of the IFTD. This justifies our motivation in using the aforementioned hybrid implicit-explicit friction discretization.

Dam-break wave interacting with a triangular obstacle

The length of the domain is 38m; the initial condition is a still water state (i.e. 0.75 m) held by a dam and the downstream floodplain is dry. For this problem, measured time histories of the water depth are available at point G11 and G13 that are respectively located 11 m, and 13 m downstream of the dam. n_M = 0.0125 and the upstream boundary is a solid wall. A total of 100 cells is used to form meshes of type '*mesh 2*' and '*mesh 3*' and the simulation time is t = 35s. Snapshots of the longitudinal profiles of the free-surface elevations at t = 10s are available in Fig. 4.

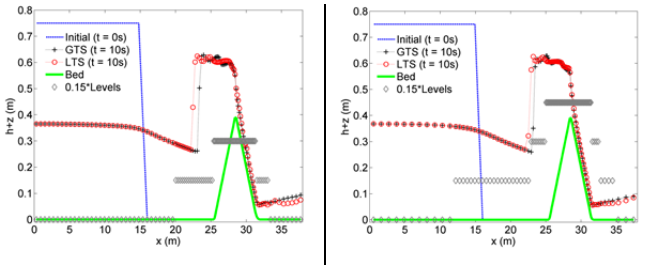

Fig. 4: Transient flow (t = 10s). Left: '*mesh* 2' and Right: '*mesh* 3'.

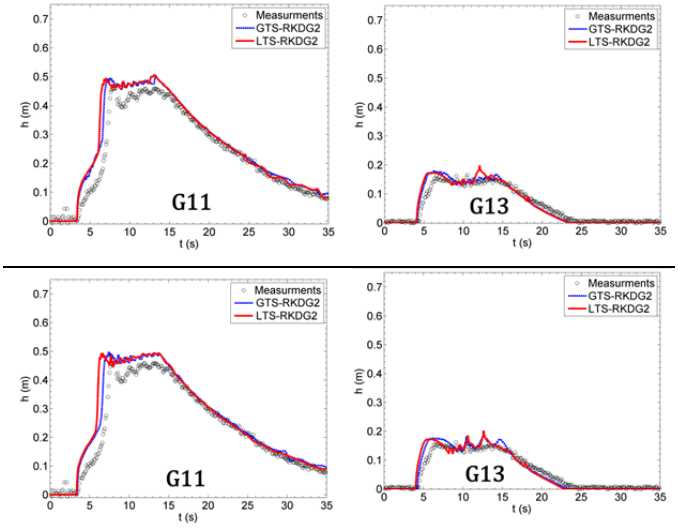

Fig. 5: Transient flow. Depth histories; Lower: '*mesh* 2'; Upper: '*mesh* 3'.

Fig. 5 contains the predicted time histories that are seen to favorably agree with the measured data. Both RKDG2 schemes survived this challenging benchmark for the two considered meshes. In contrast to the previous test, difference between the LTS-RKDG2 and GTS-RKDG2 predictions is detected; this may be attributed to dependence of the IFTD on the LTS and the involvement of relatively high local velocities in this test. However, these differences are little and have inconsequential effects on the usability of the LTS-RKDG2 model. For this case, the use of the non-uniform mesh LTS-RKDG2 scheme with three- and four- levels of LTSs enhanced, respectively, the time efficiency by 1.32 times and 1.36 times over the GTS-RKDG2 scheme.

Conclusions

A second-order LTS algorithm has been integrated with a robust RKDG2 water model on structured non-uniform meshes (LTS-RKDG2). Stabilizing features that enable the practical utility of shallow water numerical models were genuinely retained. Further considerations were given to maintain the flux conservation across cells of different sizes, and to also diminish the adverse effects of the IFTD (i.e. due to the involvement of the LTS within). Two LTS-RKDG2 models, which adapts LTSs of {Δt, $\Delta t/2$, $\Delta t/4$} and {Δt, $\Delta t/2$, $\Delta t/4$, $\Delta t/8$}, were set, tested and compared with the associated GTS-RKDG2 with a particular focus on the relative runtime saving. Our numerical experiments show that the use of LTS algorithm in an RKDG2 SWEs numerical solver is able to generically produce similar prediction as the GTS-RKDG2 counterparts. For the considered tests, the use of an LTS-RKDG2 scheme boosted the computational efficiency, referring to the GTS-RKDG2, by 1.32-to-1.99 times when adapting three LTSs; and by 1.36-to-2.3 times with a 'four' levels LTS-RKDG2 model. Research is currently underway to extend the LTS-RKDG2 model to 2D on dynamically adaptive meshes.

References

Cockburn, B. & Shu, C.-W. (2001). Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. Journal of Scientific Computing, **16**(3): 173-261.

Crossley, A.J. & Wright, N.G.. (2005). Time accurate local time stepping for the unsteady shallow water equations. International Journal for Numerical Methods in Fluids, **48**:775-799.

Delis, A.I. and Kampanis, N.A., *Numerical flood simulation by depth averaged free surface flow models*, in *Environmental Systems, in Encyclopedia of Life Support Systems (EOLSS)*, A. Sydow, Editor. 2009.

Guinot, V. (2003). *Godunov-type schemes: an introduction for engineers*., Elsevier: Amsterdam.

Krivodonova, L. (2010). An efficient local time-stepping scheme for solution of nonlinear conservation laws. Journal of Computational Physics, **229**:8537-8551.

Kesserwani, G. & Liang, Q. (2012). Locally limited and fully conserved RKDG2 shallow water solutions with wetting and drying. Journal of Scientific Computing, **50**:120-144.

Murillo, J., García-Navarro, P., and Burguete, J. (2009). Time step restrictions for well-balanced shallow water solutions in non-zero velocity steady states. International Journal for Numerical Methods in Fluids, **60**:1351-1377.

Sanders, B.F. (2008). Integration of a shallow water model with a local time step. Journal of Hydraulic Research, **46**:466-475.

Toro, E.F. (2001). *Shock-capturing methods for free-surface shallow flows*, John Wiley & Sons.

Shu, C.-W. & Osher, S. (1988). Efficient implementation of essentially non-oscillatory shock-capturing schemes. Journal of Computational Physics, **77:**439-471.