Memo

To
Anton Heinsbroek, Sam van der Zwan

Date	**Reference**	**Number of pages**
July-August, 2019		32

From	**Direct line**	**E-mail**
Mart Borsboom	+31 (0)88 335 8435	mart.borsboom@deltares nl

Subject
Numerical design of a fast, robust and accurate 1D shallow-water solver for pipe flows with large time scales – proposal (WORK IN PROGRESS)

Copy to
Martijn Verhoek

The department H2I has expressed an interest in the use of SUPSUB, a fully nonlinearly implicit 1D shallow-water solver for mixed super- and subcritical flow in channels of arbitrary shape. The first version of this program has been developed in 1994 by Helmus van de Langemheen (Van De Langemheen (1994)), following the work described in Borsboom and Van Der Marel (1992). In 1996 I started a full reevaluation of the applied numerical method and a full redesign of the software implementation. From 1997 to 2000 several improved versions were developed. A short summary of the features and performance of the March 1999 version can be found in Borsboom (1999a).

The numerical algorithm of SUPSUB is described in detail in Borsboom (2001). Using the insights obtained while writing that chapter, in 2001 the development of the next version was started, at the same time moving the code from F77 to F90. Due to a lack of interest that work has never been finished (see also Footnote 21).

At the request of Ruud Lemmens (now retired) from the department IFT (now H2I) I have used SUPSUB in 1999–2001 for the simulation of flows in one of the great man-made rivers in Libya (https://en.wikipedia.org/wiki/Great_Man-Made_River): a pipe of $4\mathrm{m}$ in diameter and nearly 400 kilometers long, cf. Borsboom (1999c). An adjustment of the iterative solution algorithm appeared to be necessary to make SUPSUB suitable for the very efficient computation of the mixed (super- and subcritical) free-surface and pressurized flow in closed conduits. Based on the results, we estimate that SUPSUB is capable of computing the slowly varying flow through such a man-made river about 60 times faster than a SOBEK-type solver, cf. Footnote 7. We expect that this can be improved by a factor 2 to 5 with the improvements of the SUPSUB algorithm presented in this memo.

Nearly all the work on SUPSUB (including most of this memo) has been done in my own time.

In this memo (once it is finished):

- The physical model (equations, boundary conditions, geometry, drying and flooding procedure, . . .) that is solved by SUPSUB.
- Presentation and discussion as well as a summarizing description of the numerical methods (space discretization, time discretization, solution algorithm) that are used and implemented in SUPSUB.

- Improvements/extensions of the above that (at short or long term) may be or are required, or are (highly) desired, to make that SUPSUB's physical model and its numerical implementation (by itself and/or in combination with other software components) are suitable for the *robust, accurate and very efficient simulation of pipe flows, especially those with large time scales*.

As for the last item: *input from H2I required*.

A fairly recent paper on the 1D modeling of mixed free-surface/pressurized flow in pipes (using a Preissmann slot on top of closed conduits to artificially create a free surface at all flow conditions) is Kerger *et al.* (2011). It may be useful to consult this and similar/related publications.

Summary and table of contents

Section 1 – Introduction

Presentation of model equations and of the two-step numerical modeling approach that is applied in SUPSUB.

In **Section 1.1** a short discussion about **The beneficial effects of smoothing** and in **Section 1.1.1** a discussion about the **Advantage of smoothing for the simulation of pipe flows**.

In **Section 1.2 – Other model components** an overview of the model components yet to be specified, such as source terms, boundary conditions, junctions, and structures.

Section 2 – Space discretization

A short and still incomplete section on the discretization of the model equations implemented in SUPSUB: a central vertex-centered finite volume method. No staggering, no upwind (not even at the boundaries).

In **Section 2.1 – Smoothing in space** a short and still incomplete sketch of the methods applied in SUPSUB to ensure sufficiently smooth variables in space. Not only the (initial) solution, but also the (initial) geometry and the (initial) grid should be and must remain sufficiently smooth in order to ensure sufficiently small discretization errors, which is the basis of the applied two-step numerical modeling approach. The SUPSUB space smoothing techniques need some evaluation/rethinking but work pretty well in practice and are considered sufficient to begin with.

Section 3 – Time discretization

Not yet finished, but fairly complete. An analysis of the θ time integration method is presented that shows, as expected, that it is advantageous to use different θ's for different parts of the model equations. For the non-dissipative terms take $\theta = 1/2$ to obtain maximum accuracy; for the dissipative terms take $\theta \approx 0.6$–0.8 to obtain accuracy in combination with enhanced stability. We have added the results of an analysis of a few DIRK2 methods (2-stage Diagonally Implicit Runge-Kutta) to illustrate that a more elaborate time integration method, because it has more parameters for optimization of the method, may have better stability properties and/or may produce the required level of accuracy at lower computational costs.

In **Section 3.1 – Smoothing in time** a few preliminary remarks of what requires attention. To be completed.

Section 4 – Drying and flooding and fully filled pipe

A smooth and continuous procedure is required:

- for the central discretizations applied in SUPSUB, cf. Section 1,
- for the fast and robust modified Newton solver presented in Section 5.

Setting grid cells (explicitly) dry and wet is out of the question.

Variations on the artificial porosity method (effectively a generalization of the Preissmann-slot concept) are discussed. To be continued.

Section 5 – Iterative solution algorithm

Outline of the derivation/construction of a modified Newton solver that solves a nonlinearly implicit time discretization (or nonlinear steady-state discretization) of a flow/transport problem very fast (which implies stability), *at the condition* that the Jacobian and its derivative are sufficiently smooth. This is realized by means of a linearization-error-dependent pseudo-time-step technique, the details of which remain to be developed. Several improvements to the very successful method developed for and applied in SOBEK-RE and SUPSUB are presented.

The pseudo-time-step method presented here has never been applied or published by others, as far as I know. The pseudo-time-step scalings that I find in the literature (didn't do a full survey) are all ad hoc and require calibrations and/or special measures in order to be stable, let alone be fastly converging. A major drawback of all these methods is the global scaling of the pseudo time step: when it needs to be small somewhere, it is made small everywhere, slowing down convergence globally, also when that would only be necessary locally. Needless to say that such a strategy will never be optimal.

In **Section 5.1 – Initializations** some remarks about the importance of sufficiently smooth initializations of the iteration processes, but also about the importance of keeping solutions smooth during the iterations.

In **Section 5.1.1 – Initialization of steady-state solution processes** the suggestion to reduce the computational time of steady-state computations by doing them on a series of grids, starting with a very coarse grid and using the result from a coarse grid to initialize the iteration process on the next finer grid, until the desired grid resolution is reached. This well-known strategy is obviously a simple form of multi-grid acceleration.

In **Section 5.1.2 – Initialization of unsteady solution processes** we present the idea of reducing the computational time of unsteady computations by improvement of the initialization of the iteration processes per time step. Better initial guesses are obtained from higher-order extrapolations in time (linear, quadratic, ...) using the solution of several previous time levels, instead of just taking the solution from the previous time level (the simple constant extrapolation applied in SUPSUB). Of course, the higher the order of the extrapolation, the more solutions of previous time levels need to be available (and stored). Alternatively, we could use an explicit time integration scheme like leap-frog to come up with an estimation of the solution at the next time level. Leap-frog only requires the solution of the last two previous time levels. An analysis shows that leap-frog may be expected to work quite well.

Section 6 – References

1 Introduction

The pipe flow model consists of the 1D shallow-water equations:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \,, \tag{1}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial Q^2/A}{\partial x} + gA\frac{\partial \zeta}{\partial x} = \frac{\partial}{\partial x}\left(\nu A \frac{\partial Q/A}{\partial x}\right) - g\frac{PQ|Q|}{A^2C^2} \,. \tag{2}$$

Space coordinate x $[\mathrm{m}]$ is defined along the axis of the pipe. The unknowns in this system of equations are wetted cross section $A = A(x,t)$ $[\mathrm{m}^2]$ and discharge $Q = Q(x,t)$ $[\mathrm{m}^3/\mathrm{s}]$. Cross-section-averaged flow velocity u $[\mathrm{m}/\mathrm{s}]$ is equal to Q/A.

Given the geometry of the pipe, i.e., the width $W = W(x,z)$ $[\mathrm{m}]$ that may vary along the axis and generally varies in vertical direction z $[\mathrm{m}]$, the relation between A and water level $\zeta = \zeta(x,t)$ $[\mathrm{m}]$ is:

$$A(x,\zeta) = \int_{z_b(x)}^{\zeta(x,t)} W(x,z)\,\mathrm{d}z \,, \tag{3}$$

with $z_b = z_b(x)$ $[\mathrm{m}]$ the vertical coordinate of the bottom, below which we have $W = 0$. Wetted perimeter P $[\mathrm{m}]$ is defined as:

$$P(x,\zeta) = \int_{z_b(x)}^{\zeta(x,t)} \sqrt{4 + \left(\partial W(x,z)/\partial z\right)^2}\,\mathrm{d}z \,. \tag{4}$$

The remaining variables in (2) are viscosity coefficient ν $[\mathrm{m}^2/\mathrm{s}]$ and Chézy coefficient C $[\mathrm{m}^{1/2}/\mathrm{s}]$. Other empirical bottom friction laws, e.g., Manning, can be used as well.

Although the viscosity term can be used for physical modeling purposes, it has primarily been introduced in (2) as part of the two-step numerical modeling technique that will be applied, cf. Figure 1. That technique consists of a *smoothing* step to regularize all non-smooth features in the original continuous problem (model equations, geometry, initial conditions, boundary conditions, any other data) that may be *difficult* to solve numerically with high accuracy and generally requires the use of special techniques like upwind and flux limiters. The *easy problem*, on the other hand, is (assumed to be) sufficiently regularized to allow the construction of the *discretized problem* by means of the straightforward application of accurate central discretizations. Since the amount of smoothing required depends on the (solution-dependent) numerical errors made in the discretization, there is a dynamic *feedback* from the second step to the first step. To ensure that *feedback* and *smoothing* are guaranteed to give small errors in the discretization step, a discretization has been developed that allows to reformulate the *residual* error in the continuous problem *equivalent* with the *discretized problem* (and obtained from a Taylor expansion of the latter) in terms of local errors in the discretized solution. For more information see Borsboom (1998, 2001, 2002).

The idea behind the application of the two-step method is that the effect on the solution of the smoothing errors in the first step should be larger than the discretization errors in the second step. When the smoothing errors are guaranteed to be the dominant numerical modeling errors, inspection of (the effect of) the smoothing errors suffices to get insight in the accuracy of a numerical simulation. Physics-based smoothing errors are generally much easier to interpret than the complex mathematical expressions of discretization errors.

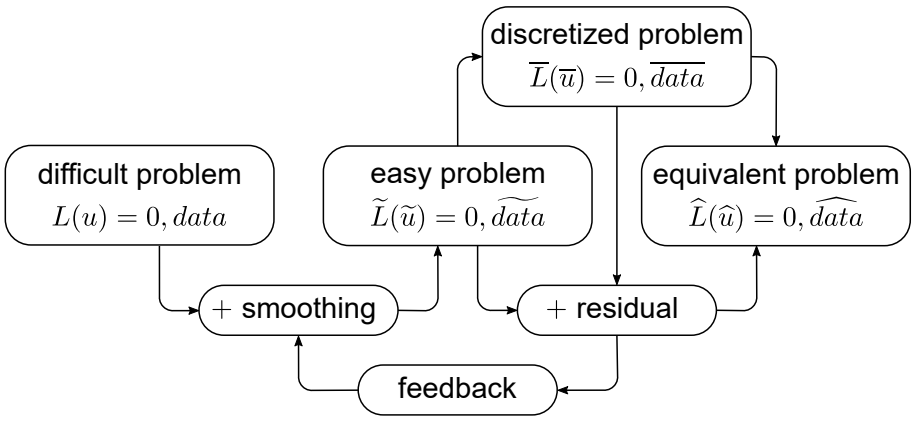

Figure 1: The two-step numerical modeling approach.

The premise behind regularizing the solution of the model equations (1) and (2) by means of only adding/increasing the viscosity (i.e., no smoothing/dissipation added to the continuity equation) is that a sufficiently large viscosity guarantees a sufficiently smooth solution[1] *provided* that all data (geometry, initial condition, boundary conditions, ...) is (or has been made) sufficiently smooth. Since both the (whatever required) smoothing of the data and the (whatever required) viscosity enhancement can be inspected afterwards *and* form the largest numerical modeling errors, the two-step method allows for easy error inspection. We reiterate that the smoothing errors are part of the numerical modeling process and hence are to be considered as *numerical* errors. Since the amount of smoothing required depends on the grid resolution and/or size of the time step, these errors can be reduced by local refinement of the grid and/or reduction of the time step. This is actually the basis of the adaptive grid method that we developed some 20 years ago, cf. Borsboom (1998, 2001).

1.1 The beneficial effects of smoothing

Smoothing can actually improve the accuracy of numerical simulations. In Borsboom (2002) it is shown that for a free-surface flow over an obstacle with a steep bottom slope, more accurate numerical results are obtained on coarse grids when that steep slope is first smoothed locally. On finer and finer grids, when the steep slope is resolved with higher and higher resolution and geometry smoothing has less and less effect, the difference between both approaches vanishes.

The reason for the beneficial effect of smoothing on the numerical accuracy can be explained by considering the energy equation. When the solution is differentiable (and *only* then!), continuity

[1] This idea is nearly 70 years old and has been presented for the first time in Von Neumann and Richtmyer (1950): "*The equations of hydrodynamics are modified by the inclusion of additional terms which greatly simplify the procedures needed for stepwise numerical solution of the equations in problems involving shocks* (MB: or any other steep-gradient phenomenon). *The quantitative influence of these terms can be made as small as one wishes by choice of a sufficiently fine mesh.*"

The same filtering idea to get rid of un(der)resolved solution details at the smallest grid scales is applied in LES (Large Eddy Simulation) and ILES (Implicit LES), see for example Rodi *et al.* (2013); Piomelli (2014).

equation (1) and momentum equation (2) can be combined[2] to energy equation:

$$\frac{\partial\left(Au^2/2+g\int_{z_b}^{\zeta}Wz\,\mathrm{d}z\right)}{\partial t}+\frac{\partial\left(Q(u^2/2+g\zeta)\right)}{\partial x}=\frac{\partial}{\partial x}\left(\nu Q\frac{\partial u}{\partial x}\right)-\nu A\left(\frac{\partial u}{\partial x}\right)^2-g\frac{Pu^2|u|}{C^2}\ . \quad (5)$$

with $u=Q/A$.

Using a conservative finite-volume discretization, the conservation of mass and momentum of (1) and (2) can be maintained at the discrete level. However, due to the discretization errors in the numerical approximation of (1) and (2), the conservation of energy expressed in (5) is generally lost at the discrete level. This may lead to large errors in areas of steep gradients, cf. Stelling and Duinmeijer (2003). Discretizations exist that conserve mass and momentum as well as energy, although the latter in a locally averaged sense only, see for example Van Reeuwijk (2011). Because these discretizations must be central (by definition, since any form of upwind leads to energy dissipation, i.e., no energy conservation anymore), they are not suited for the computation of steep gradients (such schemes are typically used in highly detailed direct numerical simulations of turbulent flows).

In Stelling and Duinmeijer (2003) a different approach is proposed: "*In flow expansions, a numerical approximation is applied that is consistent with the momentum principle. In flow contractions, a numerical approximation is applied that is consistent with the Bernoulli equation.*". The equation applied in flow contractions is the momentum equation in non-conservative depth-averaged form, obtained by subtracting u times (1) from depth-integrated momentum equation (2) and dividing the result by A. Omitting the viscosity term and rewriting the convection term in a conservative form, the result is:

$$\frac{\partial u}{\partial t}+\frac{\partial\left(u^2/2+g\zeta\right)}{\partial x}=-g\frac{Pu|u|}{AC^2}\ . \quad (6)$$

This equation is called in Stelling and Duinmeijer (2003) the "*energy-head conserving*" form of the 1D momentum equation. Like (5), it conserves energy head $u^2/2+g\zeta$ in steady-state applications (in the absence of dissipation), however, (6) is *not* an energy equation and does *not* conserve energy in unsteady applications[3]. Furthermore, it is not clear when to use (2) and when to use (6), i.e., what to do outside areas of strong expansion or contraction, and what to do in the case of a change of flow direction when an expansion becomes a contraction and vice versa. NB, in areas of smooth flow variation, whether an expansion or a contraction, there should be no problem applying a discretization of (2), the equation that is physically to be prefered. These aspects have not been investigated and are barely addressed in Stelling and Duinmeijer (2003).

In the two-step method that we propose the problem of having to design a discretization suitable for steep-gradient phenomena is circumvented by reducing the steepness of gradients (by means of smoothing/regularization) up to the point where discretization errors are guaranteed to be sufficiently small (well, at least that's the idea). This requires a discretization that allows some form of quantification of the effect of discretization errors on the solution, cf. Borsboom (2001), and a regularization that can make the effect of discretization errors on the solution sufficiently small, cf. Borsboom (2001). Coupling the discretization error back to the regularization (the feedback in

[2] Take the sum of (1) times $g\zeta-u^2/2$ and (2) times u.

[3] Also, an equation like (6) can only be formulated in 1D, i.e., there is no multi-dimensional equivalent. NB, there is of course a multi-dimensional equivalent of (5). In Stelling and Duinmeijer (2003) it is claimed that a 2D version of (6) (not expressing the conservation of energy head) "*will not generate energy locally and the local dissipation of energy will be small*". It is not explained why that would be the case.

Figure 1) will then automatically ensure that also the effect of discretization errors in (5), including energy conservation errors, are sufficiently small, even though a discretization of this equation has not been considered.

The results presented in Borsboom (2002) show how this works. Without geometry smoothing a large error in energy conservation[4] is made in the flow contraction area when the grid is too coarse to resolve the steep bottom slope with sufficient accuracy. With geometry smoothing that error is about 5 times smaller and mainly the result of the small energy loss due to the artificial viscosity that is applied locally to ensure a sufficiently smooth solution[5]. A larger amount of geometry smoothing would have reduced this error (a discretization error) further, at the expense of a larger smoothing error in the geometry used in the computation. In practice we try to find a reasonable compromise between smoothing errors in the first step and discretization errors in the second step of the two-step numerical modeling. So far the same set of parameters for artificial viscosity etc. seems to provide each time a similar satisfactory compromise. This is in agreement with the underlying theory.

It is well known that the numerical approximation of functions with strong variations (up to discontinuities, shocks, or even singularities) is generally not very accurate. It is difficult to maintain a reasonable level of numerical accuracy in such situations, and special techniques (upwind, flux limiters, switching between a momentum and an energy principle) are required to cope with them. The non-smoothness inherent to such techniques hampers the development of efficient solvers for fully nonlinearly implicit time integration and steady-state problems. In contrast, the enforcement of smoothness of the two-step modeling approach enables the use of the exact same discretization of the exact same equations everywhere, at the same time ensuring sufficient differentiability of all variables. Both are essential for the development of the key ingredient of a fast and robust fully nonlinearly implicit time-integration scheme/fast and robust steady-state solver: a modified Newton iteration method with optimized pseudo time stepping, cf. Section 5.

1.1.1 Advantage of smoothing for the simulation of pipe flows

In SUPSUB (and many other programs for the simulation of pipe flows) the shallow-water equations (1), (2) are also used for the simulation of pressurized flow in closed conduits. This is made possible by adding artificially a small slot (the Preissmann slot) on top of conduits. This allows to have a free surface at any level above the bottom. By using a very small slot width, the spurious effect of the slot on the flow simulation (increase in water volume, modified wave speed) can be made negligibly small.

Aureli *et al.* (2015) mention that "*in order to model the physics correctly, the slot width should be set in such a way that the speed of gravity waves equals the pressure wave speed*". They continue by stating that "*This constraint generally leads to the estimate of very narrow slots, which induce significant numerical oscillations at flow transitions . . . . In order to overcome this problem, in the practice the slot width is usually set at values between 1% and 10% of the pipe diameter . . .*".

[4] Note that the energy conservation is actually an artifact of the applied model. The inviscid shallow-water equations without bottom friction that are considered ensure energy conservation as long as the solution is differentiable, as in a (smooth) contraction. In reality there will always be some energy loss in a contraction, especially in strong contractions because of the viscous dissipation that occurs in strongly varying flows. The hydrostatic assumption underlying the shallow-water model is not valid anymore either in strong contractions (and expansions), by the way.

[5] The energy loss is modeled by the negative-definite second term in the right-hand side of (5). NB, the first term in the right-hand side of (5) that is in conservative form models the conservative redistribution of energy due to viscosity.

Because of the smoothing of the slot, cf. Borsboom (1999c)[6], it turned out to be possible to use in SUPSUB a slot width as small as $0.0002\mathrm{m}$ on top of a man-made-river pipe of $4\mathrm{m}$ wide, i.e., a slot that is only 0.005% of the pipe diameter. Since the results obtained with this slot were virtually identical to those for a slot of $0.0004\mathrm{m}$, we have never used an even smaller slot (if I remember correctly).

In Aureli *et al.* (2015) (with experimental data that may be very useful for validation) a first-order accurate explicit numerical method is used. The work of other authors that is discussed in this paper involves similar numerical techniques. It seems that a method similar to our second-order accurate fully nonlinearly implicit method has never been developed. That accurate and very efficient method[7] could only be developed and is only successful because of the enforced and hence guaranteed smoothness of *all* variables (solution, geometry, initial and boundary conditions, ...).

1.2 Other model components

Besides the flow equations (1) and (2) to be applied inside pipes we (may) have to deal with, now or in the future (i.e., discretize and solve):

- *Source terms*: given functions, functions of the solution, turned on/off in time (e.g., by some controller), varying in space, some combination of all this, ...? To be included in continuity equation (1) and/or momentum equation (2)? How are source terms to be specified?
  NB, source terms must be (made) smooth as well, both in space and in time, cf. Section 1.1.
- *Boundary conditions*: which types?
- *Junctions*: p.m. for the moment?
- *Structures*: come with locally very steep gradients. It is therefore in general not a good idea to model them as part of the flow problem inside pipes (by specifying some sudden change in geometry, a discontinuous source term, whatever), where smoothness is required and is to be enforced. Unless locally a very fine grid is used, because then the effect of smoothing sudden changes in geometry and/or discontinuous source terms will be local enough. However, we will then have a conflict between the simulation of a flow that is locally so small-scale that it is

[6] We use a Preissmann funnel rather than a Preissmann slot, i.e., slots that are connected to pipes by a short tapered section to enhance smoothness. For maximum smoothness, a tapered section should consist of an exponentially varying width that smoothly connects slot to pipe. This is one of the improvements that we propose, see Section 4 below. In Borsboom (1999c) the short tapered section between pipe and Preissmann slot is composed of several segments with linearly varying width mimicking an exponential variation, an improvement over the single section with linearly varying width that we developed for SOBEK-RE. In León *et al.* (2009) a similar Preissmann slot smoothing is proposed.

[7] In the SUPSUB unsteady man-made-river flow simulations that we have performed some 20 years ago we have used a time step of $1800\mathrm{s}$. This seemed to be a reasonable compromise between accuracy and efficiency. We did once a computation with a twice smaller time step and got virtually the same result. Conclusion: for this application a time step of $1800\mathrm{s}$ is small enough. We have never used a larger time step, if I remember correctly.

Recently we learned from Anton Heinsbroek that the method that IFT/H2I has been using for such simulations (a semi-implicit SOBEK-type method?) requires a time step of about $0.25\mathrm{s}$ in order to be stable. Taking into account the additional work per time step of SUPSUB (about 30 modified Newton iterations per time step required, while each iteration is probably about 3 to 4 times more work than a single SOBEK time step), we estimate that for this man-made river application SUPSUB is about $(1800/30/4)/0.25 = 60$ times more efficient than the method used by IFT/H2I. Because of the very small Preissmann slot, the very high grid resolution, and the second-order accurate discretization in space and in time, the results of the SUPSUB simulations are likely to be more accurate than those of SOBEK. In the period that these computations were made (some 20 years ago), Ruud Lemmens once told me that the SUPSUB results were indeed better than those of SOBEK. To my knowledge, a formal comparison between results has never been made.

The efficiency estimation is for the SUPSUB version of 20 years ago. In this memo several suggestions are made that, besides accuracy improvements, are expected to give an additional factor 2 to 5 gain in efficiency.

partially non-hydrostatic *and* multi-D, and its description by the 1D hydrostatic shallow-water flow equations.
Conclusion: structures are to be modeled by means of separate (empirical) structure formulas describing the local momentum transfer/energy loss through structures in combination with mass conservation. Both equations are steady state, which is acceptable as long as the length of structures is very small compared to the length of relevant unsteady phenomena (the length scales of irrelevant/negligibly small unsteady phenomena do not matter), or when the total length of structures in a pipe is much smaller than the pipe itself. Structure equations to be used in a coupling between pipe segments. Same/similar couplings to be used at boundaries (coupling of a pipe to the outer world) and at junctions (coupling of a pipe to (an)other pipe(s)).

- ...

Other modeling aspects to be decided upon:

- *Specification of geometry*: which forms need to be supported, how to discretize/approximate?
- ...

2 Space discretization

The SUPSUB method, cf. Borsboom (2001).

Unknowns are Q and A => cont.eq.(1) linear in unknowns => linearization of cont.eq. is exact. Linearization of time derivatives of (1) and (2) also exact.

Model equations (1) and (2) are transformed to computational space with coordinate ξ:

$$\frac{\partial x_\xi A}{\partial t} + \frac{\partial Q}{\partial \xi} = 0 \,, \tag{7}$$

$$\frac{\partial x_\xi Q}{\partial t} + \frac{\partial Q^2/A}{\partial \xi} + gA\frac{\partial \zeta}{\partial \xi} = \frac{\partial}{\partial \xi}\left(\frac{\nu}{x_\xi}\frac{\partial Q}{\partial \xi} - \frac{\nu}{x_\xi}\frac{Q}{A}\frac{\partial A}{\partial \xi}\right) - x_\xi g\frac{PQ|Q|}{A^2C^2} \,, \tag{8}$$

with $x_\xi = \partial x/\partial \xi$. Discretization of (7) and (8) on a uniform grid in computational space ξ $[-]$ with grid size $\Delta\xi = 1$, i.e., the coordinate transformation $x = x(\xi)$ defines the grid spacing in physical space.

All functions in computational space (A, Q, ζ, ν, ..., but also x) are approximated piecewise linear on the uniform grid in ξ => leading approximation errors are the 2nd-order interpolation errors. For the compatible finite volume scheme that is applied[8], it can be shown that the leading effect of discretization errors on the solution is proportional to the exact same 2nd-order interpolation errors, Borsboom (2001). To ensure that these errors, made in the second step of the two-step method (Figure 1), are sufficiently small, all functions of ξ that appear in (7) and (8) must be sufficiently smooth, i.e., smoothness also applies to the grid in physical space defined by $x(\xi)$.

Because of unknown A, we do not use A as a function of ζ, but ζ as a function of A, i.e., we need the inverse of geometric relation (3). No problem, cf. SUPSUB.

[8]This finite volume scheme consists of the *exact* integration of (7) and (8) over the finite volumes that run from cell center to cell center, replacing *all* functions of ξ by their piecewise linear approximation. NB, the viscosity term has in (8) been written in the form suitable for the unambiguous application of the finite volume scheme, i.e., no second derivatives of combinations of variables.

When, for whatever reason, a jump/discontinuity in geometry/grid/solution is required and/or needs to be modeled (i.e., local smoothing not acceptable), an interface between two pipe segments is required, since the numerical method applied inside pipes does not support jumps/discontinuities. Across an interface/junction between pipes jumps/discontinuities are possible. See also Section 1.2 under '*Structures*', which are to be modeled by means of interfaces between the two sides of a structure formula and the ends of the pipe segments connected by that structure.

...

2.1 Smoothing in space

Smoothness in ξ required => apply geometry smoothing (and error-dependent artificial viscosity a la SUPSUB).

Also smoothness in z required (?), to ensure well-defined linearizations, which are essential for Newton. => Ensure smooth cross-sectional profiles, etc? At second view: probably not, since in (7) and (8) we only deal with functions in ξ (and t of course). => Ensure that whatever geometry variation, linearization coefficients, ... that we have are (made) sufficiently smooth functions in ξ per time step/steady state *and* per iteration (!). => We need to figure out (at some point) how to do that.

Some preliminary thoughts: with A unknown (and supposedly smooth, by construction of the method), this effectively means (correct?) that all that is required is a method to ensure $\zeta(\xi, t)$ to be a smooth function of $A(\xi, t)$. Is this the only geometry smoothing required? Eh, no, also smoothness of linearization coefficient $1/W(\xi, t)$ as a function of $A(\xi, t)$ required. These smoothings should be such that linearization $(1/W)_{smooth} = \partial\zeta_{smooth}/\partial A$ holds (correct!).

Short-term solution: the simple geometry-smoothing method currently applied in SUPSUB (to be reviewed!): not terrific, but it works. Development of something better is p.m.

3 Time integration scheme

intro ...

Theta method, mid-point rule (or exact at discrete level?) because of higher accuracy and better stability.

Use two different θ's: θ equal to (or perhaps slightly larger than?) $1/2$ for imaginary part (advection/convection and wave propagation, the non-dissipative terms) for best possible accuracy, and θ about 0.6–0.8 for real part (viscosity and bottom friction, the dissipative terms) for good compromise between accuracy and range of applicability.

Let's have a closer look at the numerical time integration by considering the linear model equation (nonlinear effects p.m.):

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = \nu\frac{\partial^2 c}{\partial x^2} \ . \tag{9}$$

Inserting in this equation some Fourier mode $c = C(t)\exp(ikx)$, we obtain for amplitude C of this

solution mode the ordinary differential equation:

$$\frac{\partial C}{\partial t} + iukC = -\nu k^2 C \ , \tag{10}$$

whose evolution from previous time level t^{n-1} to the next time level t^n over the time interval $\Delta t = t^n - t^{n-1}$ is:

$$C^n = C^{n-1} \exp(-iuk\Delta t - \nu k^2 \Delta t) \ , \tag{11}$$

i.e., the change over a time step of size Δt is (mix of a phase change of size $uk\Delta t$ and an amplitude change of size $\nu k^2 \Delta t$):

$$g_{\text{exact}} = \ln\left(\frac{C^n}{C^{n-1}}\right) = -iuk\Delta t - \nu k^2 \Delta t \ . \tag{12}$$

For more general problems involving systems of equations (in particular the shallow-water equations) we would have phase changes due to the odd derivatives in the model equations (advection/convection, wave propagation, dispersion terms) and amplitude changes due to the even derivatives in the model equations including zeroth derivatives (diffusion/viscosity, bottom friction, sources and sinks). These can all be included in an analysis, but here we restrict ourselves for illustration purposes to the simple model equation (9).

In view of the type of problems to be solved (pipe flows with large time scales), we are looking for time discretizations that are accurate for small values of $uk\Delta t$ and $\nu k^2 \Delta t$, and sufficiently stable for large values. Small $uk\Delta t$ and $\nu k^2 \Delta t$ are essentially due to small wave numbers k (*not* due to small Δt!), corresponding to large length scales with large time scales. (Very) large $uk\Delta t$ and $\nu k^2 \Delta t$ are essentially due to large wave numbers (in combination with large time steps) and correspond to small time scales. Because the rapid unsteady physics associated with these small time scales is (assumed to be) virtually non-existent (and hence irrelevant/negligible) in the type of applications that we are considering, accuracy in time is here not required *at the condition* that the behavior of these modes is sufficiently stable.

Notice that, since $uk\Delta t$ is linear in k but $\nu k^2 \Delta t$ is quadratic in k, the values of $\nu k^2 \Delta t$ encountered in practice tend to be (much) larger than those of $uk\Delta t$, especially on computational grids with fine (local) resolutions that allow (very) high k values. For this reason the diffusion/viscosity stability restriction of explicit time integration schemes (of the form $\nu k^2 \Delta t < O(1)$) tends to be more severe than the advection/convection restriction (of the form $uk\Delta t < O(1)$)[9]. To avoid the use of very small time steps, one could decide to (locally) lower the diffusion/viscosity coefficient. It is clear that this technique, applied in D-FLOW FM, modifies the physical model and so should be used with great care.

[9] This is associated with the concept of stiffness: the time scale $1/(\nu k^2)$ of viscosity/diffusion terms may become very small, while at the same time these very small time scales may be physically totally irrelevant, like in the large time-scale simulations that we are concerned with in this memo. The same may apply to terms modeling fast chemical reactions or fast physical interactions, by the way. Explicit schemes are then less well suited; their stability restriction forces the use of unnecessarily very small time steps leading to unnecessarily large computational times.

Terms causing stiffness are best integrated (partially/linearly) implicit in time, e.g., in an Implicit-Explicit (IMEX) method where explicit time integrations of the non-stiff terms and implicit time integrations of the stiff terms are combined, see for example Frank *et al.* (1997). When also the advection/convection terms are 'stiff', in the sense that the smallest $1/(uk)$ that occur are irrelevant in the large-scale unsteady processes to be studied (but relevant in the stability condition of explicit schemes!), fully implicit time-integration schemes are the methods of choice. When designed properly, they are the most efficient way to handle such problems.

To illustrate a few things we briefly consider the class of two-stage Diagonally Implicit Runge-Kutta methods (DIRK2), see for example ROS2 (Rosenbrock-2, a nonlinear version of DIRK2) in Verwer *et al.* (1997). Since the amount of work involved in the two stages per time step is about twice that of the (single-stage) θ–method, we start with a DIRK2 configured in such a way that it corresponds to two θ–methods per time step, each one over half a time step. Applied to (10), this scheme reads:

$$\begin{aligned} &\left(1+\theta_a\, iuk\Delta t/2+\theta_d\,\nu k^2\Delta t/2\right) C^{n-1/2}\\ =&\left(1+(\theta_a-1)\, iuk\Delta t/2+(\theta_d-1)\,\nu k^2\Delta t/2\right) C^{n-1}\,,\\ &\left(1+\theta_a\, iuk\Delta t/2+\theta_d\,\nu k^2\Delta t/2\right) C^{n}\\ =&\left(1+(\theta_a-1)\, iuk\Delta t/2+(\theta_d-1)\,\nu k^2\Delta t/2\right) C^{n-1/2}\,, \end{aligned}$$

with θ_a and θ_d the value of θ of respectively the advection part and the diffusion part of the equation. From this we obtain the amplification factor:

$$g_\theta=\ln\left(\frac{\left(1+(\theta_a-1)\, iuk\Delta t/2+(\theta_d-1)\,\nu k^2\Delta t/2\right)^2}{\left(1+\theta_a\, iuk\Delta t/2+\theta_d\,\nu k^2\Delta t/2\right)^2}\right)\,, \tag{13}$$

To get an impression of how well solution (11) of (10) is approximated by the θ–scheme, i.e., how well (12) is approximated by (13), we plot isolines of $\Im(g_\theta)$ and $\Re(g_\theta)$ as a function of $uk\Delta t$ and $\nu k^2\Delta t$, cf. Figure 2.

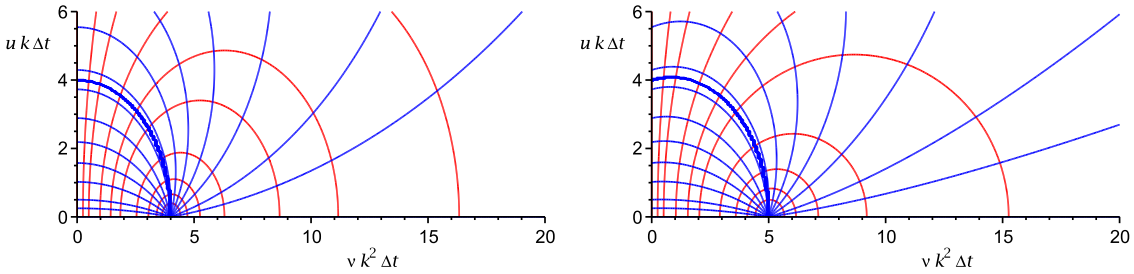

Figure 2: Isolines of $\Im(g_\theta)$ (in blue) and $\Re(g_\theta)$ (in red) as a function of $uk\Delta t$ and $\nu k^2\Delta t$ for $\theta_a=\theta_d=1/2$ (left) and for $\theta_a=1/2$ and $\theta_d=0.6$ (right). Isoline levels of $\Im(g_\theta)$ are $[-3,-5/2,-2,-3/2,-1,-1/2,-1/4,0,1/4,1/2,1,3/2,2,5/2,3]$. Isoline levels of $\Re(g_\theta)$ are $[-1/2,-1/4,0,1/4,1/2,1,3/2,2,3,4,5,6]$. Figures taken from Borsboom (2005–2009, 2019).

For a perfect scheme these isolines would be parallel to either the $\nu k^2\Delta t$ axis or the $uk\Delta t$ axis and at the correct level. In reality this is only obtained for $uk\Delta t,\nu k^2\Delta t\ll O(1)$, because of the limited accuracy of any numerical approximation. When only the behavior in time of relatively long modes (k small) is relevant, this allows the use of large time steps *provided* that the behavior of the scheme for $uk\Delta t,\nu k^2\Delta t\gg 1$ is sufficiently stable. It is easy to see that the Crank-Nicolson scheme ($\theta_a=\theta_d=1/2$) is then a bad choice. From (13) it follows that $\Re(g_\theta)=0$ for $uk\Delta t,\nu k^2\Delta t\to\infty$. In other words, for $uk\Delta t$, $\nu k^2\Delta t$ (very) large the Crank-Nicolson scheme becomes marginally stable.

The obvious and usual remedy to this problem is to use a θ larger than $1/2$. However, for the terms modeling advection/convection, wave propagation, ... (the odd-order non-dissipative space derivatives in the model equations), a value of $\theta>1/2$ introduces an amount of numerical dissipation of $O(\Delta t)$ that quickly leads to the spurious attenuation of propagating information that may be substantial, especially over longer distances. In contrast, the numerical error that a $\theta>1/2$ introduces in

dissipation terms (the even-order space derivatives modeling diffusion, viscosity, bottom friction, ...) is to some extend acceptable and even beneficial[10]. Although a value of $\theta > 1/2$ reduces the accuracy for $\nu k^2 \Delta t \ll 1$ (because it changes the order of accuracy of the time integration scheme from $O(\Delta t^2)$ to $O(\Delta t)$), it improves the accuracy for larger values, cf. the intersections of the red isolines with the $(\nu k^2 \Delta t)$–axis up to about $\nu k^2 \Delta t = 3$ in the right Figure 2 versus those in the left Figure 2. Especially important is the improved behavior of $\Re(g_\theta)$ for very large values of $\nu k^2 \Delta t$ when $\theta > 1/2$. From (13) we obtain $\Re(g_\theta) = \ln\left((\theta_d - 1)^2/(\theta_d)^2\right)$ for $k \to \infty$. In other words, the closer to 1 we take θ_d, the more stable the time integration becomes for large k. For $\theta_d = 0.6$ (the value used in right Figure 2) we have $\Re(g_\theta) = -\ln(4/9) = 0.811$ as $k \to \infty$. Recall that for $\theta_d = 1/2$ (Crank-Nicolson scheme) we have $\Re(g_\theta) = -\ln(1) = 0$ and hence marginal stability as $k \to \infty$.

A better behavior for $k \to \infty$ can be obtained by fully exploiting the degrees of freedom of DIRK2. (Specification of DIRK2 and its amplification factor g_{DIRK2} left out for the moment.) We first show in left Figure 3 the performance of DIRK2 with its coefficients chosen such that the scheme is both *third-order accurate* (the highest order of accuracy possible with this scheme) and *A-stable*, i.e., stable ($\Re(g_{\mathrm{DIRK2}}) \geq 0$) for any value of $uk\Delta t$ in combination with $\nu k^2 \Delta t \geq 0$. Remarkably, this scheme is globally *less* accurate than two times Crank-Nicolson, cf. left Figure 3 compared to left Figure 2. Third-order A-stable DIRK2 is only more accurate than second-order A-stable double Crank-Nicolson for $uk\Delta t, \nu k^2 \Delta t < 0.53$. Because the error of double Crank-Nicolson is rather small ($< 0.58\%$) in this small range of $(uk\Delta t, \nu k^2 \Delta t)$–values, there is hardly any need to apply a more accurate scheme. The fairly small error of double Crank-Nicolson over a range much larger than that of highest-order A-stable DIRK2 on the other hand is very attractive, because it allows the use of larger time steps. For this reason we show in right Figure 3 the performance of a DIRK2-type scheme where the additional degrees of freedom in the design of the scheme have *only* been used to improve the stability properties. It has the nearly 6 times higher value $\Re(g_{\mathrm{DIRK2}}) = -\ln(0.01) = 4.605$ for $k \to \infty$ and a similar accuracy along the $(\nu k^2 \Delta t)$–axis than the two times θ–method with $\theta_a = 1/2$ and $\theta_d = 0.6$ (cf. right Figure 2.

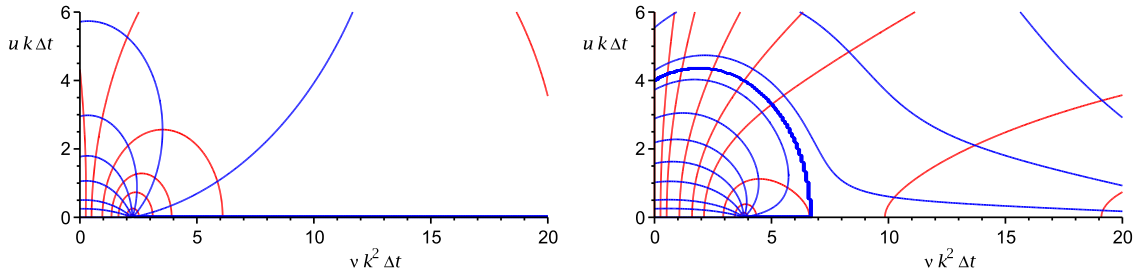

Figure 3: Isolines of $\Im(g_{\mathrm{DIRK2}})$ (in blue) and $\Re(g_{\mathrm{DIRK2}})$ (in red) as a function of $uk\Delta t$ and $\nu k^2 \Delta t$ for stable *third-order accurate DIRK2 (left), and for DIRK2 configured as two times θ–method with $\theta_a = 1/2$ for advection and improved for diffusion (right). For isoline levels see Figure 2. Figures taken from Borsboom (2005–2009, 2019).*

There are *two* third-order accurate DIRK2 configurations. The other one, whose performance is shown in left Figure 4, is *not* A-stable, however, compared to the Crank-Nicolson method of left

[10]Notice that, in view of the physical uncertainties and calibrations often involved in dissipation terms (application of bottom friction models, of turbulence models), but also because of their smoothing quality leading to smoother solutions and hence smaller numerical errors, dissipative numerical errors are generally less of a problem in dissipation terms than in propagation terms anyway, especially in convection-/wave-dominated applications

Figure 2 (and certainly compared to the A-stable third-order accurate DIRK2 of left Figure 3), it is accurate over a much larger range of $(uk\Delta t)$–values when $\nu k^2 \Delta t$ is small. A very small amount of dissipation, which may always be present in practice (background viscosity level, high-order upwind dissipation), is sufficient to guarantee/get unconditional stability. As an alternative one may slightly adapt the DIRK2 scheme's parameters to ensure stability given a certain time step, preferably only when and where necessary (i.e., in regions with virtually no dissipation), which would require some dynamically/solution-dependent/automatic adaptive procedure.

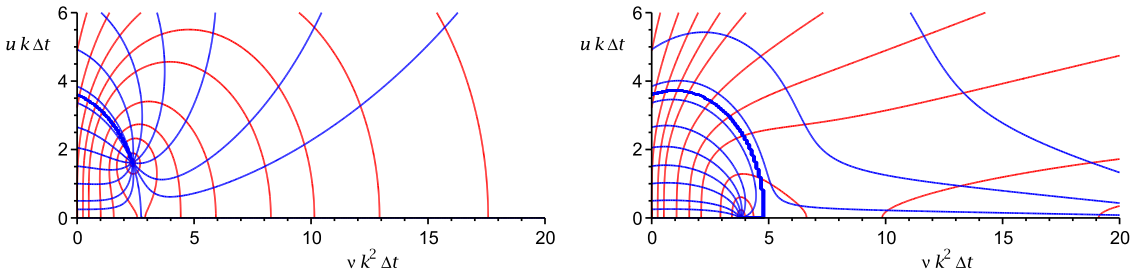

Figure 4: Isolines of $\Im(g_{\mathrm{DIRK2}})$ (in blue) and $\Re(g_{\mathrm{DIRK2}})$ (in red) as a function of $uk\Delta t$ and $\nu k^2 \Delta t$ for unstable *third-order accurate DIRK2 (left), and for* unstable *third-order accurate DIRK2 for advection combined with same improved diffusion as applied in right Figure 3 (right). For isoline levels see Figure 2. Figures taken from Borsboom (2005–2009, 2019).*

Right Figure 4 shows the performance when this third-order accurate DIRK2 is applied to the non-dissipative term in (10) and the same DIRK2 with improved stability as in right Figure 3 is applied to the dissipative term. As a result, the intersections of the blue lines with the $(uk\Delta t)$–axis are the same as in left Figure 4, and the intersections of the red lines with the $(\nu k^2 \Delta t)$–axis are the same as in right Figure 3).

Concluding remarks:

- It has been shown (and is also well known, cf. IMEX methods) that it is advantageous to use different time integration methods (in particular: different θ's) for the non-dissipative terms of the equations (advection, convection, wave propagation, ..., i.e., the terms leading to imaginary eigenvalues in the ODE systems) and for the dissipative terms of the equations (diffusion, viscosity, bottom friction, ..., i.e., the terms leading to real eigenvalues in the ODE systems). Notice that this is *not* well possible in combination with upwind discretizations because of their dissipative nature, which can be quite large (everywhere in the case of low-order upwind; locally in the case of higher-order upwind with flux limiters at the places where flux limiting is maximal).
  In the case of upwind discretizations, there is no distinction possible between the time integration method applied to certain non-dissipative parts of the space-discretized equations and the method applied to certain dissipative parts, *unless* the upwind discretizations are split into a skew-symmetric part (corresponding to a central discretization of first and higher-order odd derivatives) and a symmetric part (corresponding to the dissipation introduces by the upwind schemes). Although certainly not convenient (increase in algorithmic complexity $\Rightarrow$ more complex implementation, higher computational load), this may be possible inside domains. At boundaries it may be virtually impossible.
  The SUPSUB algorithm only uses central discretizations, so fairly easy to use a different time integration for the non-dissipative and the dissipative parts, *provided* that that has been taken

into account in the set-up of the software implementation (not in the one of 20 years ago).

- Other DIRK methods, i.e., those built up of 3 or more stages, may be applicable/interesting as well. For more info see for example the recent publications Kennedy and Carpenter (2016); Boom and Zingg (2018). These are just two out of many publications on time integration methods for (very) stiff systems of equations. (Simulations with very large time steps fall into this category.) On the other hand, results as presented above obtained from analyses of these schemes indicate that (variations on) DIRK2 already seem to be quite well suited to stiff problems.
- DIRK-type methods (or any other Runge-Kutta method for that matter) can be extended by not only using the solution at the current time level but also those from previous time levels. We then enter the realm of Adams-Bashforth-Moulton(-type) (ABM) methods[11]. This can be a cheap way (because it is explicit) to enhance accuracy, i.e., a way to obtain the same or similar performance (in terms of accuracy, stability, robustness, ...) with (much?) larger time steps at hardly any additional computational costs. Note that stability might deteriorate when using explicit information from further back in time. Using the solution from previous time levels is likely to require special measures when time step variation is in order. On the other hand, by using a fully/sufficiently nonlinearly implicit time integration scheme, there is no need anymore to constantly vary the time step in order to meet some explicit stability condition.
- P.m.: a dynamically adapted θ ($\Rightarrow$ a θ that varies in space and time) for better accuracy in combination with robustness, especially when using larger Δt: it is better to shift θ locally from $1/2$ towards 1 and only when necessary, than to use such an accuracy-reducing shift all the time everywhere. NB, with a properly designed/working dynamically adapting θ, the use of two different θ's is expected not to be necessary/useful anymore.

3.1 Smoothing in time

Smoothness in time required => smooth/regularize solution imposed at boundaries, smooth/regularize switching on/off valves, ..., and/or use temporarily a very small Δt in combination with (preferably locally) a theta shifted from $1/2$ toward 1. That is preferably taken care of automatically by means of dynamic adaptation of θ: a time-error-dependent (and hence space-varying) θ that combines accuracy (where possible) with dissipation/robustness (where necessary).

4 Drying and flooding and fully filled pipe

(sort of) Preissmann slot at bottom and top ...

Maranzoni *et al.* (2015): useful reference?

Candy (2017): "To allow efficient time integration over a range of element sizes, an implicit treatment necessitates a continuum approach to interface tracking. A thin film is applied, which as Medeiros and Hagen (2013) note, generally satisfies mass and momentum conservation without significant

[11] Quite some material on the analysis of partially and fully implicit time integration schemes (as well as fully explicit predictor-corrector/ABM schemes and explicit multi-stage RK schemes) is available, but otherwise it is fairly easy to perform such analyses by reusing/adapting existing Maple sheets (I have quite a few of them). In fact, the results presented in the Figures 2, 3 and 4 have been obtained from a Maple sheet with an analysis of implicit ABM methods, setting the coefficients of the solutions of previous time steps equal to zero.

special treatment, and produces a realistic and smooth wetting front."

For the fully nonlinearly implicit solver to work, drying and flooding must be continuous and sufficiently smooth. This is also required by the applied two-step numerical modeling technique.

The intention is to use a continuous drying-and-flooding procedure based on the artificial porosity concept, effectively a generalization of the Preissmann-slot concept. In many Boussinesq-type wave models (including TRITON, cf. Borsboom (2005)), drying and flooding is modeled using artificial porosity.

The artificial porosity formulation of TRITON is not smooth. It is continuous and differentiable everywhere, except at the transition between the porous layer (below a certain small threshold depth $\delta_{\rm dry}$ above the bottom) and normal water (above $\delta_{\rm dry}$ plus bottom level), where it is only continuous. Notice that a Preissmann slot is *not* continuous at the transition between slot and main water column, i.e., the addition of a Preissmann slot to a 1D channel causes a discontinuity in the width of the modified/slot-enhanced channel when the bottom of that channel is flat.

In Borsboom (2013–2016) the favorable properties of an exponentially varying artificial porosity formulation as applied in many Boussinesq-type wave models (including TRITON) is confirmed. It is also shown that the quadratic variation proposed in Van 't Hof and Vollebregt (2005) is indeed a less good idea, although it is certainly better than a Preissmann slot. In Borsboom (2013–2016) a procedure is developed to obtain a smooth transition between an exponential porous layer (or any other layer, for that matter) and the main water column.

A potential 'drawback' of an exponentially varying artificial porosity is that its effect vanishes exponentially fast as the water level gets deeper and deeper below the bottom. Although this effect as such is positive, it leads to extremely small geometry values at large distances below the bottom, which may hamper the reliability of double-precision computations. To avoid this we have recently considered the addition of a tiny Preissmann slot with polynomially varying width (constant, linear, quadratic, cubic, ...) to a Preissmann 'slot' with exponentially varying width, i.e., to an exponential artificial porosity, cf. Borsboom (2019b). This gives a nice compromise between an essentially exponential artificial porosity below the bottom down to moderate depths, and widths that do not become extremely small at larger depths. Of course, the distance below the bottom at which this form can be used is limited. While a strictly exponential formulation can be used down to any distance below the bottom (the width becomes exponentially small but remains positive), the depth at which the mixed form can be used is limited. On the other hand, it can be configured such that it can be used down to whatever distance below the bottom is required (just like the Preissmann slot in fact).

The procedure to be added at the top to allow the computation of pressurized flow in a fully filled pipe is similar: apply a combination of exponential porosity and a tiny Preissmann slot of constant width. This is nothing else but an improved and fully smooth version of the procedure that we applied in the man-made river computations of 20 years ago, cf. Section 1.1.1. Because SUPSUB only supports the specification of geometries with piecewise linear width, the exponentially shaped transition between the main channel/pipe and a tiny Preissmann slot on top of it has been approximation piecewise linearly.

NB, the drying and flooding procedure applied in SUPSUB is simpler. It consists of the addition of a smooth and continuous term to the momentum equation that becomes significant at depths of $O(\delta_{\rm dry})$ and balances the hydrostatic pressure gradient at water depths equal to $\delta_{\rm dry}$, turning off the force that would otherwise drive a layer of water down a slope. It ensures that a layer of water

of about $\delta_{\rm dry}$ (typically $O(0.01{\rm m})$) thick remains whenever and wherever the bottom falls dry. The effective thickness of the layer of water left by the exponential plus tiny polynomial Preissmann-slot method, on the other hand, becomes smaller and smaller as drying progresses and the water level descends further and further below the bottom. It therefore introduces a much smaller perturbation. Moreover, unlike that term that balances the hydrostatic pressure gradient, the Preissmann-slot method does *not* affect the conservation of energy of the shallow-water equations[12]. In view of its advantages, and since a decent Preissmann-slot method is required at, and hence needs to be developed and implemented for, the top of pipes anyway, there is no reason for not applying the Preissmann-slot method for drying and flooding. Of course, we may start with applying the methods used in SUPSUB, which after all work pretty well: addition of a hydrostatic-pressure-gradient balancing term in the momentum equation for the modeling of drying and flooding; and a piecewise linear approximation of an exponential section that connects the top of a pipe to a tiny Preissmann slot for the modeling of pressurized flow.

5 Iterative solution algorithm

After discretization in space (and in time if unsteady computation), a nonlinear system of equations results per steady state or per time step that is roughly of the form[13]:

$$\Delta t_{\rm inv}\mathbf{M}\big(\mathbf{u}^n-\mathbf{u}^{n-1}\big)+\mathbf{f}\big(\theta\mathbf{u}^n+(1-\theta)\mathbf{u}^{n-1}\big)=0\ , \tag{14}$$

with $\Delta t_{\rm inv}=1/\Delta t$, with $\mathbf{M}$ the mass matrix[14] that is only grid-dependent (and hence constant in time because we only consider fixed grids), and with $\mathbf{u}$ the vector of discrete unknowns (the values of A and Q at the grid points). Nonlinear vector function $\mathbf{f}$ contains the discretization in space, which consists of the discretized flow equations inside the domain as well as of (combinations of) discretized flow equations at the boundaries to model the outgoing flow information. Vector function $\mathbf{f}$ also contains the discretized boundary conditions that specify the incoming flow information as well as the, most of time unwanted, reflection at the boundaries.

System of equations (14) applies to both unsteady and steady-state computations. For the latter, take $\Delta t_{\rm inv}=0$ and $\theta=1$. This is how it has been implemented in SUPSUB, and also in SOBEK-RE: the same code takes care of both unsteady and steady-state computations.

[12] Looking for arguments to decide which continuous drying-and-flooding method would be better, a Preissmann-type slot/artificial porosity or a pressure-gradient balance term, I once looked at energy conservation. I must have somewhere the proof of why the former is energy conserving and the latter is not. On the other hand, the proof is very easy, if I remember correctly, so no problem to repeat it.

[13] The form given here is sufficient to illustrate the SUPSUB iterative solution algorithm and to develop some ideas, but the actual form is and will be more elaborate. First of all, we will include independent implicit weighting (i.e., the use of different values of θ) for the non-dissipative terms (propagation, convection) and for the dissipative terms (viscosity, bottom friction), because this provides better possibilities for the optimization of the balance between accuracy and robustness, cf. Section 3. Second, for a more compatible discretization in time (which may be advantageous, cf. Section 3), the way that $\mathbf{u}^{n-1}$ and $\mathbf{u}^n$ are weighted in time for the evaluation of $\mathbf{f}$ will probably be a bit different than indicated here (not decided upon yet), but the essence has been captured. Third, θ is assumed to be fixed and constant; the possibility of having a dynamically adapted time- and space-varying θ has not been indicated.

For convenience, we have left out the artificial viscosity coefficient (an essential part of the method). NB, because the dynamically adapted artifical viscosity is too complex to be linearized efficiently, it is solved by means of underrelaxed explicit updates, which turns out to work remarkably well in practice. We propose to apply the same technique for the (yet to be developed) dynamically adapted time- and space-varying θ that is expected to have a comparable complexity.

[14] Mass matrix $\mathbf{M}$ is, just like most mass matrices, diagonally dominant and symmetric, while for each row the sum of elements equals 1, except for the rows associated with boundary conditions.

Nonlinear system of equations (14) is solved iteratively by means of a customized modified Newton method. The basis of this method is Newton, which because of its quadratic convergence is very efficient when it converges. Unfortunately, because of its lack of robustness, convergence of Newton is anything but guaranteed. Also, quadratic convergence is only obtained upon full and exact linearization, which in practice may be virtually impossible to realize, may be extremely complex to implement, and/or may be excessively expensive to compute.

We introduce modifications to Newton that remedy these drawbacks, at the same time preserving as much as possible the efficiency of Newton. They have been put together, implemented, tested and refined some 20 years ago during the development of SOBEK-RE and later during the development of SUPSUB. As fas as we know, unsteady 1D flow simulations with large time scales[15] carried out by fully nonlinearly implicit SOBEK-RE or SUPSUB consistently take about a factor 50–100 less computational time than semi-implicit SOBEK or D-FLOW FM (see also Footnote 7 on page 8). On the other hand, the last two computational flow models are more efficient for simulations with (very) small time scales (dam-break problems, (shock-, pressure-)wave propagation). SOBEK and D-FLOW FM also tend to be more robust.

The iteration steps of the modified Newton method that we apply are of the form:

$$\left(\Delta t_{\mathrm{inv}}\mathbf{M} + \mathbf{T}_{\mathrm{pseu}}^{n,m-1}\mathbf{M}_{\mathrm{pseu}} + \mathbf{J}^{n,m-1}\right)\Delta\mathbf{u}^{n,m} = \mathbf{res}^{n,m-1}\ , \tag{15}$$

with:

$$\mathbf{res}^{n,m-1} = -\Big(\Delta t_{\mathrm{inv}}\mathbf{M}\big(\mathbf{u}^{n,m-1} - \mathbf{u}^{n-1}\big) + \mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big)\Big)\ . \tag{16}$$

In the right-hand side of (15) we have the residual (16), defined as minus the left-hand side of (14) with the sought solution at next time level n replaced by its approximation $\mathbf{u}^{n,m-1}$ at previous iteration level $m-1$. It is clear[16] that when solution correction $\Delta\mathbf{u}^{n,m} = \mathbf{u}^{n,m} - \mathbf{u}^{n,m-1}$ has become zero, i.e., when the modified Newton method has converged, the residual is zero as well, i.e., the iteratively determined solution $\mathbf{u}^{n,m}$ must at that point be equal to solution $\mathbf{u}^{n}$ of the nonlinear system of equations (14).

The advantage of formulating an iteration process in this so-called delta form is that it separates the problem that we want to solve (the 'physics' in the right-hand side) from how we solve it (the 'numerics' in the left-hand side). Thanks to the delta form we can apply whatever we want in the left-hand side of (15) to improve the properties of the iteration method and/or simplify its implementation: modified Newton, inexact Newton, any type of splitting, ... All this has no effect on the final solution as long as the iteration process is regular enough to prevent it from converging to an incorrect and physically meaningless solution.

A second advantage of the delta form is the insensitivity to round-off errors when implemented properly. To begin with, round-off errors associated with solving the linear system of equations per iteration step in the left-hand side of (15) affect the solution update and *not* the solution itself. Since $\Delta\mathbf{u}^{n,m}$ converges to zero, this round-off error effect vanishes as well, as opposed to what would happen when $\mathbf{u}^{n,m}$ would be computed. Furthermore, the round-off errors that do affect the solution, i.e., those associated with the explicit computation of the residual in the right-hand side of

[15] Many, if not most, flow simulations of practical interest to Deltares and its clients fall into this category.

[16] This is usually not so clear when the next iterative solutions $\mathbf{u}^{n,m}$ and not the iterative solution *updates* $\Delta\mathbf{u}^{n,m}$ are computed in the iteration steps, especially in the software implementation of a numerical algorithm.

(15), can be kept small by a suitable software implementation of the flux balances of the underlying finite-volume discretization. The trick is to first compute the individual terms of the model equations (time derivatives, mass balance, convection, pressure gradient, ...) from their components (solution at previous and next time level, fluxes of mass, convection, pressure, ...). Since the terms as a whole are (much) smaller than the largest components from which they are composed, the effect of round-off errors is kept to a minimum per term. The round-off errors in the second step, where the terms are put together to the discretized model equations, are therefore kept as small as possible as well.

The properties of the modified Newton iterative solution process (convergence speed, robustness) depend entirely on the matrix $\Delta t_{\mathrm{inv}}\mathbf{M} + \mathbf{T}_{\mathrm{pseu}}^{n,m-1}\mathbf{M}_{\mathrm{pseu}} + \mathbf{J}^{n,m-1}$ in the left-hand side of (15). It is this matrix that determines how the next solution correction $\Delta\mathbf{u}^{n,m}$ is obtained from the residual $\mathbf{res}^{n,m-1}$ at the previous iteration.

Jacobian $\mathbf{J}^{n,m-1} \approx \partial\mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big)/\partial\mathbf{u}^{n,m-1}$ is the approximate linearization of $\mathbf{f}$ as a function of $\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}$ with respect to $\mathbf{u}^{n,m-1}$. All terms in the discretization of (7) and (8) are fully and exactly linearized, except for the artificial viscosity coefficient ν and the dynamically adapted θ (once it has been developed) that because of their complexity are kept explicit and are (ν) or will be (θ) underrelaxed for stability. See also Footnote 13.

The purpose of the linearization is to estimate the optimal direction of solution correction $\Delta\mathbf{u}^{n,m}$, given the solution at previous iteration level $m-1$. Right-hand side $\mathbf{res}^{n,m-1}$ determines its size. Because of the highly nonlinear character of the equations, the direction estimate is only accurate for $\Delta\mathbf{u}^{n,m}$ sufficiently small. This becomes clear when we consider the Taylor-series expansion of $\mathbf{f}\big(\theta\mathbf{u}^{n} + (1-\theta)\mathbf{u}^{n-1}\big)$ around previous iterative solution $\mathbf{u}^{n,m-1}$:

$$\begin{aligned}\mathbf{f}\big(\theta\mathbf{u}^{n} + (1-\theta)\mathbf{u}^{n-1}\big) = {}& \mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big) \\ & + \frac{\partial\mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big)}{\partial\mathbf{u}^{n,m-1}}(\mathbf{u}^{n} - \mathbf{u}^{n,m-1}) \\ & + \frac{1}{2}\frac{\partial^2\mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big)}{\partial\mathbf{u}^{n,m-1^2}}(\mathbf{u}^{n} - \mathbf{u}^{n,m-1})^2 \\ & + O\big((\mathbf{u}^{n} - \mathbf{u}^{n,m-1})^3\big)\ .\end{aligned} \tag{17}$$

The equations (15) and (16) are obtained by replacing $\mathbf{f}\big(\theta\mathbf{u}^{n} + (1-\theta)\mathbf{u}^{n-1}\big)$ in (14) by the first two lines in (17). The remaining lines that are nonlinear in the solution correction, are omitted. Because this is an approximation, $\mathbf{u}^{n}$ in solution correction $\mathbf{u}^{n} - \mathbf{u}^{n,m-1}$ is replaced by its approximation at the next iteration level $\mathbf{u}^{n,m}$.

For sufficiently small solution correction $\mathbf{u}^{n} - \mathbf{u}^{n,m-1}$ (approximated per iteration by $\Delta\mathbf{u}^{n,m} = \mathbf{u}^{n,m} - \mathbf{u}^{n,m-1}$) and assuming that all derivatives of $\mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big)$ with respect to $\mathbf{u}^{n,m-1}$ are bounded (which implies smoothness of $\mathbf{f}$ as a function of $\mathbf{u}^{n,m-1}$!), we have that the omitted third line in (17), which is quadratic in the solution correction, is the leading linearization error. For the linearization applied in (15) to be reliable, this error must be sufficiently small, i.e., the following condition must hold[17]:

$$\left|\frac{1}{2}\frac{\partial^2\mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big)}{\partial\mathbf{u}^{n,m-1^2}}\Delta\mathbf{u}^{n,m}\right| < O\left(\left|\frac{\partial\mathbf{f}\big(\theta\mathbf{u}^{n,m-1} + (1-\theta)\mathbf{u}^{n-1}\big)}{\partial\mathbf{u}^{n,m-1}}\right|\right)\ . \tag{18}$$

[17] Note that this condition implies that solution correction $\Delta\mathbf{u}^{n,m} = \mathbf{u}^{n,m} - \mathbf{u}^{n,m-1}$ must be sufficiently small.

This is why the term $\mathbf{T}_{\text{pseu}}^{n,m-1}\mathbf{M}_{\text{pseu}}^{n,m-1}$ has been introduced in the left-hand side of (15). $\mathbf{T}_{\text{pseu}}^{n,m-1} = [1/\Delta t_{\text{pseu},i}^{n,m-1}]$ is a diagonal matrix whose elements consist of the inverse of the local pseudo time steps $\Delta t_{\text{pseu},i}^{n,m-1}$ at the grid points i and at iteration level $n, m-1$. The $\Delta t_{\text{pseu},i}^{n,m-1}$ are to be scaled such that the solution of (15), i.e., solution correction $\Delta\mathbf{u}^{n,m} = [\Delta A_i^{n,m}, \Delta Q_i^{n,m}]$, is as large as possible (for fastest convergence) within the linearization constraint (18) (for robustness, i.e., for guaranteed convergence).

Since (18) is a bit of a vague statement, the definition of what is a proper scaling of the $\Delta t_{\text{pseu},i}^{n,m-1}$ is vague as well. For example, when the $\Delta t_{\text{pseu},i}^{n,m-1}$ are taken rather small, the linearization may be very reliable and hence the convergence may be very robust, however, the $\Delta\mathbf{u}^{n,m}$ will be rather small and so will be the convergence speed. Fortunately, our SUPSUB experience indicates that robustness and convergence speed are not very sensitive to the scaling. Scalings exist that ensure very good robustness while maintaining high convergence speeds for many different applications.

To enhance robustness, pseudo mass matrix $\mathbf{M}_{\text{pseu}}^{n,m-1}$ has been introduced to define a local redistribution of the pseudo time step over the grid points of the computational molecule[18]. The pseudo mass matrix is used in SUPSUB to give the pseudo time step a stabilizing local implicit upwind bias, an idea that we have previously applied successfully in SOBEK-RE, cf. Borsboom (1999b). Mass matrix $\mathbf{M}_{\text{pseu}}^{n,m-1}$ could also be used to realize a stabilizing implicit low-pass filter for the $\Delta A_i^{n,m}$ and/or $\Delta Q_i^{n,m}$ (may actually work better than upwind, to be investigated). Note that the implicit upwind bias or the implicit low-pass filter scales with $1/\Delta t_{\text{pseu},i}^{n,m-1}$ and is therefore automatically turned off upon convergence, when the $\Delta A_i^{n,m}$ and $\Delta Q_i^{n,m}$ vanish and pseudo time stepping is no longer necessary to ensure (18).

A problem with ensuring (18) is that $\mathbf{u}^{n,m} - \mathbf{u}^{n,m-1} = \Delta\mathbf{u}^{n,m}$ is only available *after* having solved (15). For this reason, $\Delta t_{\text{pseu},i}^{n,m-1}$ is scaled with previous solution correction $\Delta\mathbf{u}^{n,m-1}$. To avoid a deterioration of convergence speed or even unstable behavior when $\Delta\mathbf{u}^{n,m-1}$ is a bad estimate of $\Delta\mathbf{u}^{n,m}$, underrelaxation of $\Delta t_{\text{pseu},i}^{n,m-1}$ is applied. Since $\Delta\mathbf{u}^{n,m-1}$ does not exist at the very first iteration $m = 1$ (per time step or per steady state), iteration processes are conservatively initialized with a $\Delta t_{\text{pseu},i}^{n,m-1}$ based on a local pseudo CFL number of about 2. This safe choice ensures robustness, but is at the expense of efficiency, especially in unsteady flow computations when solution changes per time step are small, and hence the $\Delta\mathbf{u}^{n,m-1}$ are (very) small from the very first iteration. That will be especially the case once the initialization per time step is improved, cf. Section 5.1.2. Because of the initialization with a small CFL number and the subsequent underrelaxation, an unnecessary number of iterations is spent in getting the $\Delta t_{\text{pseu},i}^{n,m-1}$ to the large value (and in reaching the high convergence speed) that would have been possible from the beginning of the iteration loop. Here is clearly room for improvement. A summary of the SUPSUB pseudo-time-step algorithm can be found in Section 4.2, *Iterative solution algorithm*, of Borsboom (2001).

The linearization involves $\partial\zeta/\partial A = 1/W$, which poses no problem. Evaluation of the linearization *error* involves $\partial^2\zeta/\partial A^2 = \partial(1/W)/\partial A$. Since W is specified as a (continuous) piecewise linear function of z in SUPSUB, $\partial(1/W)/\partial A$ is only piecewise continuous and *not* smooth, as the SUPSUB method requires. It may be for this reason that it does not seem to have been used[19] in the scaling

[18] The pseudo mass matrix only redistributes the pseudo time step, i.e., it does not add any scaling. It is a dimensionless matrix whose sum of elements per line equals 1. Since the pseudo mass matrix only uses the grid points already used in the discretization, it does not increase the size of the matrix to be inverted.

[19] The unknowns used in SOBEK-RE are Q and ζ, with A a function of ζ. The linearization applied in SOBEK-RE therefore involves $\partial A/\partial\zeta = W$. In the second-order linearization error the term $\partial^2 A/\partial\zeta^2 = \partial W/\partial\zeta$ appears. Since cross-sectional profiles in SOBEK-RE are specified by means of piecewise linearly varying widths W (we have simply

of the pseudo time steps $\Delta t_{\mathrm{pseu},i}^{n,m-1}$. That is, I see in the pseudo-time-step subroutine of SUPSUB that the programming lines that contain (an approximation of) $\partial(1/W)/\partial A$ have been commented out (don't remember the reason for that). Notice that at the transition between free-surface pipe flow and pressurized pipe flow, the variation in W is *huge*. Likewise in the case of drying and flooding. Pressurized flow and drying/flooding in pipes are reasonably well supported by the solution algorithm of SUPSUB (because we have obtained results for a number of difficult applications in very small computational times), but it is clear that in view of the approximations that have been applied in the scaling of $\Delta t_{\mathrm{pseu},i}^{n,m-1}$, there must be significant room for improvement.

One improvement involves a better assessment of the linearization error. Turning directly to (15), we see that a negligible linearization error in fact requires (neglect the effect of the time-step and pseudo-time-step distribution by the mass matrices $\mathbf{M}$ and $\mathbf{M}_{\mathrm{pseu}}^{n,m-1}$, and replace the second derivative of $\mathbf{f}\big(\theta\mathbf{u}^{n,m-1}+(1-\theta)\mathbf{u}^{n-1}\big)$ with respect to $\mathbf{u}^{n,m-1}$ by its approximation $\partial\mathbf{J}^{n,m-1}/\partial\mathbf{u}^{n,m-1}$):

$$\left|\frac{1}{2}\frac{\partial\mathbf{J}^{n,m-1}}{\partial\mathbf{u}^{n,m-1}}\Delta\mathbf{u}^{n,m}\right| < O\big(\left|\Delta t_{\mathrm{inv}}+\mathbf{T}_{\mathrm{pseu}}^{n,m-1}+\mathbf{J}^{n,m-1}\right|\big)\ , \tag{19}$$

i.e., the size of Δt_{inv} and $\mathbf{T}_{\mathrm{pseu}}^{n,m-1}$ contribute in ensuring a small linearization error. Because this effect is ignored, the current scaling of $\mathbf{T}_{\mathrm{pseu}}^{n,m-1}$ applied in SUPSUB is not optimal.

It would be an improvement to set the pseudo time step locally to the largest possible $\Delta t_{\mathrm{pseu},i}^{n,m-1}$ (to get the highest possible convergence speed) satisfying (19) (to ensure a stable Newton convergence process). However, this procedure still has the disadvantage that it needs $\Delta\mathbf{u}^{n,m}$, which is not available[20], i.e., $\Delta t_{\mathrm{pseu},i}^{n,m-1}$ still needs to be scaled with previous solution correction $\Delta\mathbf{u}^{n,m-1}$ and needs to be underrelaxed to avoid stability problems due to the use of this explicit estimate of $\Delta\mathbf{u}^{n,m}$.

At some point we have contemplated the use of a 'predictor' iteration step[21], in which (15) would be solved using some large $\mathbf{T}_{\mathrm{pseu}}^{n,m-1}$ based on a local Courant condition of $O(1)$ (or using the pseudo-time-step matrix $\mathbf{T}_{\mathrm{pseu}}^{n,m-2}$ of the previous iteration). That would give a $\Delta\mathbf{u}^{n,m}$ prediction, or better a prediction of the local ratio's $\Delta A_i^{n,m}/\Delta t_{\mathrm{pseu},i}^{n,m-1}$ and $\Delta Q_i^{n,m}/\Delta t_{\mathrm{pseu},i}^{n,m-1}$, that are subsequently used in (19) to come up with very reliable and accurate estimates of the $\Delta t_{\mathrm{pseu},i}^{n,m-1}$. This would slightly increase the amount of work per iteration[22], but that will be be more than compensated for by the

copied this way of specifying profiles to SUPSUB), this term is piecewise constant and hence not smooth. However, the reason why it has not been used in the scaling of the pseudo time step applied in SOBEK-RE is that this term was not available in the existing program, cf. Borsboom (1999b). The addition of pseudo time stepping to SOBEK-RE was a modification *afterwards* to repair the strong lack of robustness, and hence the possibilities were limited by the already existing code. Once an algorithm has been implemented in a code, it may be very difficult (if not impossible) to implement extensions, improvements or variations of that algorithm if in the set-up of the code the (future) realization of such modifications has not been taken into account.

[20] Chicken-and-egg problem: we need $\Delta\mathbf{u}^{n,m}$ to determine $\mathbf{T}_{\mathrm{pseu}}^{n,m-1}$, but we need $\mathbf{T}_{\mathrm{pseu}}^{n,m-1}$ to be able to compute $\Delta\mathbf{u}^{n,m}$ in the next iteration step.

[21] I vaguely remember having started the implementation of such a technique somewhere in 2001, 2002, when I started working on an updated version of SUPSUB, at the same time moving the code from F77 to F90. Due to a lack of interest that work (done in my own time, like nearly everything of the SUPSUB development) has been discontinued. In 2014 I did some work to prepare SUPSUB for the computation and error assessment of results for some analytical 1D test cases (dambreak, Carrier-Greenspan), to have something to compare the D-FLOW FM validation results for those test cases with. Discontinued for the same reason.

[22] The system of equations per iteration step (15) needs to be computed only once, but the system needs to be solved twice: once with a rough estimation of the $\Delta t_{\mathrm{pseu},i}^{n,m-1}$, once with a very accurate estimation of the optimal $\Delta t_{\mathrm{pseu},i}^{n,m-1}$ based on (19). NB, with a rough estimation of the $\Delta t_{\mathrm{pseu},i}^{n,m-1}$ based on an $O(1)$ local Courant condition, the diagonal of the matrix

reduction of the number of iterations that the increased convergence speed of this improvement is expected to lead to.

Another improvement may come from considering the linearization of (14) *including* the leading linearization error (approximation of third line in (17)) directly, i.e., by replacing (15) by:

$$\left(\Delta t_{\text{inv}}\mathbf{M} + \mathbf{T}_{\text{pseu}}^{n,m-1}\mathbf{M}_{\text{pseu}} + \mathbf{J}^{n,m-1} + \frac{1}{2}\frac{\partial \mathbf{J}^{n,m-1}}{\partial \mathbf{u}^{n,m-1}}\Delta\mathbf{u}^{n,m}\right)\Delta\mathbf{u}^{n,m} = \mathbf{res}^{n,m-1} \ . \tag{20}$$

The optimal $\Delta t_{\text{pseu},i}^{n,m-1}$ are the largest possible values for which the linearization error, i.e., the fourth term of the matrix in the left-hand side, is locally sufficiently small compared to the linear terms, i.e., the first three terms of the matrix. When this is true, (20) can reliably be approximated by (15), the iterations of our modified Newton method.

Equation (20) shows that the smaller the $\Delta t_{\text{pseu},i}^{n,m-1}$ (the larger $\mathbf{T}_{\text{pseu}}^{n,m-1}$), the smaller $\Delta\mathbf{u}^{n,m}$ and hence the smaller the linearization error, but it also shows that the linearization error becomes smaller as the residual $\mathbf{res}^{n,m-1}$ (and hence the solution update $\Delta\mathbf{u}^{n,m}$) vanishes. Given local residual $\text{res}_i^{n,m-1}$ and ignoring $\mathbf{M}$ and $\mathbf{M}_{\text{pseu}}$, it may be possible to determine a very useful estimation of the local $\Delta t_{\text{pseu},i}^{n,m-1}$ from (20) by solving it approximately stating that the fourth term in the matrix (the leading linearization error) should be some (small) fraction of the first three terms. This looks like the best possible pseudo-time-step scaling that is practically feasible (if we can get it to work). A quick check (haven't figured out the details yet) shows that this leads to an expression that is in line with "*the theoretically optimal pseudo-timestep*" on page 303 of Deuflhard (2011) (a rather theoretical book, by the way).

There are many publications on or related to pseudo time stepping, also known as *pseudo-transient continuation*, for example Kelley and Keyes (1998); Gropp *et al.* (2000); Knoll and Keyes (2004); Chisholm and Zingg (2009); Hicken and Zingg (2009); Savant *et al.* (2011); Ceze and Fidkowski (2015); Brown and Zingg (2016); Mavriplis (2018). In all these papers the scaling of the pseudo time step is ad hoc and global, i.e., there is no theory underlying the choice of the pseudo time step and there is no local optimization. The latter means that if a small pseudo time step is required locally, it is applied everywhere[23]. In consequence, when the pseudo-time-step algorithm decides that the iterative solution evolution in pseudo time needs to be slow in some region, it will be slow everywhere in the domain, unnecessarily slowing down the entire iteration process.

In addition to pseudo-transient continuation, Newton methods are often stabilized by underrelaxing the solution correction before applying it to the previous iterative solution[24], i.e., by using the solution update $\mathbf{u}^{n,m} = \mathbf{u}^{n,m-1} + \alpha\Delta\mathbf{u}^{n,m}$, with $0 < \alpha < 1$. Underrelaxation is applied globally

in the left-hand side of (15) is rather large and hence the system of equations easy to solve. It may even be possible to estimate the $\Delta A_i^{n,m}/\Delta t_{\text{pseu},i}^{n,m-1}$ and the $\Delta Q_i^{n,m}/\Delta t_{\text{pseu},i}^{n,m-1}$ sufficiently well using a cheap explicit method.

[23] Often, a space-varying pseudo time step is applied that is based on a global Courant/CFL number. In that case, the same (small) Courant/CFL number is applied everywhere, regardless of whether that would only be necessary locally.

[24] The fact that a convergence process may become unstable when the solution corrections $\Delta\mathbf{u}^{n,m}$ are not underrelaxed but applied as such indicates that the $\Delta\mathbf{u}^{n,m}$ are actually too large and/or in the wrong direction, i.e., insufficiently reliable. This can only happen when the $\Delta t_{\text{pseu},i}^{n,m-1}$ are (locally) too large, i.e., when the linearization errors (the omitted second-order terms) in the modified Newton method are *not* negligible, as a consequence of which the $\Delta\mathbf{u}^{n,m}$ are *not* optimal, i.e., may indeed correct the solution in the wrong direction. The need for underrelaxation indicates that the nonlinear iterative solution process is *not* optimal.

Note that when the $\Delta t_{\text{pseu},i}^{n,m-1}$ are locally too small, although linearization errors can then be neglected and the $\Delta\mathbf{u}^{n,m}$ correct the solutions $\mathbf{u}^{n,m-1}$ in the right direction, the solution corrections are smaller than necessary and the convergence process is not optimal either. For maximum convergence speed it is essential that the $\Delta t_{\text{pseu},i}^{n,m-1}$ are locally scaled just right.

as well, i.e., when in some small area a small α would be required to ensure stability, it is applied everywhere, unnecessarily slowing down the iteration process in the (possibly large) areas where underrelaxation is not required. It can be proven that under certain conditions for α sufficiently small the residual $\mathbf{res}^{n,m-1}$ is guaranteed to decrease, i.e., that the iteration process is guaranteed to converge (Mavriplis (2018)). Any form of rapid convergence is of course out of the question.

We do not know of any work where a locally optimized pseudo time step like the one used in SOBEK-RE and SUPSUB has been developed and applied, i.e., one that is based on ensuring just small enough linearization errors. To our knowledge it is the only approach with a theoretical foundation and an analytical derivation/approximation to maximize nonlinear convergence speed (which, by definition, includes/guarantees stability and robustness of the modified Newton process). The construction of the pseudo time step of SOBEK-RE and SUPSUB includes some simplifications, making that result suboptimal. The approach to follow is to consider (20), which includes all relevant terms: all linear terms and the leading nonlinear term that is neglected in (15). The best possible pseudo time step is therefore the one that makes $\Delta t_{\text{pseu},i}^{n,m-1}$ just small enough to allow the neglect of the nonlinear term, i.e., to allow the approximation of (20) by (15). We expect this $\Delta t_{\text{pseu},i}^{n,m-1}$ to be a major inprovement over the pseudo time step of SOBEK-RE and SUPSUB. This is confirmed by the fact that it is in line (according to a quick analysis; detailed analysis pending) with pseudo-time-step expression (6.41) in Deuflhard (2011) that is proven to be optimal.

Final remark: nonlinear convergence is much faster in unsteady simulations than in steady-state simulations. This is because of the enhancement of the diagonal with $\Delta t_{\text{inv}} = 1/\Delta t$, in combination with the fact that θ in unsteady applications is not 1 but smaller[25]. The effect of linearization errors is therefore relatively smaller. Note that that advantage is only fully exploited when the effect of $1/\Delta t$ is taken into account in the determination of suitable values for $\Delta t_{\text{pseu},i}^{n,m-1}$. Unsteady nonlinear convergence is also faster because in each iteration loop the initial solution is close to the final solution. See also Section 5.1.2 where we propose methods to improve the initialization of unsteady iteration loops for further reduction of the number of nonlinear iterations.

Another final remark: the construction of the optimal $\Delta t_{\text{pseu},i}^{n,m-1}$ is no exact science. That starts already with its definition: it is the pseudo time step that gives convergence to some tolerance level in the smallest number of iterations[26] (which implies stability) for a very wide range of unsteady and steady-state problems. Because the number of iterations does not only depend on $\Delta t_{\text{pseu},i}^{n,m-1}$ but also on the choice of $\mathbf{M}_{\text{pseu}}$, the underrelaxation of the boundary conditions and the numerical viscosity, the initial solution of the modified Newton iteration loops, and everything not yet included in the SUPSUB model (the missing physics that need to be added for pipe-flow simulations), a truly optimal expression for $\Delta t_{\text{pseu},i}^{n,m-1}$ does not exist. This is rather fortunate, since due to the very high nonlinear complexity of the optimization problem a truly optimal expression would be impossible to derive anyway. The best we can do (in terms of getting the best compromise between robustness and convergence speed), is to analyse the Newton iteration process and modify it such as to accommodate its shortcoming (lack of robustness) while getting the highest possible convergence speed along the entire iteration trajectory, thereby minimizing the number of iterations and hence the amount of work. We do not aim for proven stability, as this results in much lower convergence speeds *and* more work per iteration, e.g., to determine iteratively the (maximum possible) underrelaxation coefficient α per

[25] It makes the linear systems of equations (15) easier to solve as well, although that is mainly an advantage when these systems are solved iteratively, as in multi-D applications.

[26] As the rate of convergence depends on the size of the pseudo time step, it varies during an iteration process. It is therefore not meaningful to opt for the highest rate of convergence.

iteration, cf. Mavriplis (2018). Instead, we aim for a technique that with confidence can be argued to be a very good compromise between robustness/stability and convergence speed, with a scaling coefficient that allows to tune the balance between these two. Our experiences with SOBEK-RE and SUPSUB have shown that, if developed, implemented and calibrated properly, this works very, very well.

5.1 Initializations

The entire numerical method (discretization and solution algorithm) is based on the assumption of smoothness and hence requires smoothness. For the correct working of the solution algorithm it is therefore important to also maintain a sufficient level of smoothness during the iteration processes. If, for some reason, smoothness would get lost and intermediate solutions would not be smooth enough anymore, intermediate discretization errors may become large enough for the solution to become meaningless. As a result, the equations may temporarily become very difficult to solve. If that happens, a very small pseudo time step is likely to be necessary to prevent the iteration process from stalling, at the expense of a slow convergence process and large computational times. To avoid this, it is advised to apply techniques that ensure intermediate solutions to always be sufficiently smooth.

An essential component is the initialization: the first step in keeping intermediate discretization errors small is to ensure small discretization errors at the initialization of iteration processes. In other words, initial solutions should satisfy reasonably well the nonlinear systems of equations that are to be solved.

The components that keep solutions smooth during iteration processes are

- convergence-error dependent pseudo time stepping (currently applied in SUPSUB),
- upwind pseudo time stepping (currently applied in SUPSUB) or implicitly smoothed pseudo time stepping (new idea);
- convergence-error dependent underrelaxation of imposed boundary conditions to prevent large solution changes and non-smoothness near boundaries (currently applied in SUPSUB).

5.1.1 Initialization of steady-state solution processes

It is generally not possible to construct by simple means approximations/estimations to steady-steady solutions that satisfy the boundary conditions and at the same time satisfy reasonably well the discretized steady-state equations, while demanding an at most negligibly small effort to compute. The problems are just too complex for that. Because of the very large (non-smooth!) initial convergence errors that this may lead to, initializing a steady-state iteration process by such a solution is considered to be not a good idea.

In SUPSUB a different approach is followed: steady-state solution processes are always initialized by still-water flows, i.e., zero flow velocity and a uniform water level that is high enough for the model to be fully flooded everywhere. In SUPSUB (single-branch) computations we would normally take that water level equal to or slightly higher than the level expected at the inflow boundary[27]. This

[27] That would probably be the highest water level expected at the inflow boundaries when there is more than one inflow boundary. On the other hand, when the coupling of branches in a 1D network model is not applied directly but solved iteratively by means of some domain-decomposition (DD) technique (combined in some way with the iterative solver for

initial solution is of course (infinitely) smooth and is guaranteed to satisfy the discretized equations, by construction.

A still-water flow initial condition does not satisfy the boundary conditions. Because the boundary conditions are underrelaxed and are not applied directly, this does not pose a problem[28]. As a result of applying the boundary conditions 'slowly' in the beginning of an iteration process, but also because of the use of an initially small pseudo time step, initial convergence is slow, cf. Figure 5 for an example. In this so-called *searching phase* with small pseudo time step and strong underrelaxation of the boundary conditions, the solution algorithm is searching where the solution is. From Figure 5 we conclude that on a domain divided in 2000 finite volumes it apparently may take about 600 iterations for the information imposed at the boundaries to travel and propagate through the domain and for the iterative solution to get close to the final solution. When that point is reached, convergence errors become significantly smaller, automatically turning off more and more both the underrelaxation applied at the boundaries and the pseudo time stepping applied inside the domain, leading to a strong increase in convergence speed. When that happens, the iteration process enters the *converging phase*: very fast convergence to the final solution in 20–30 iterations.

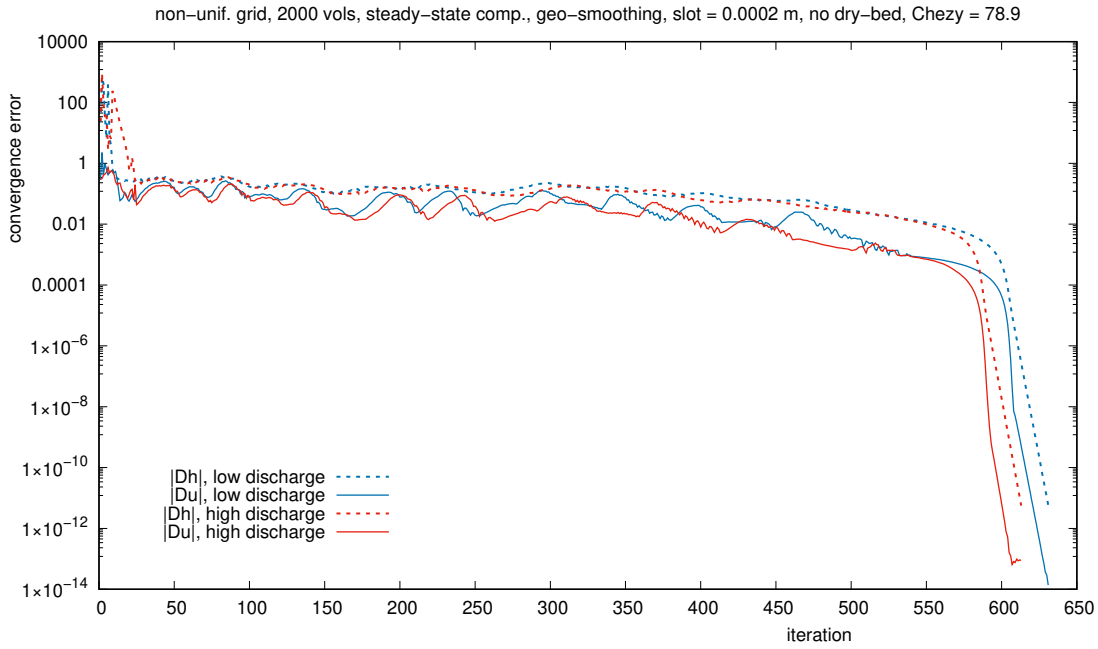

Figure 5: SUPSUB *convergence history of two steady-state man-made river computations on a grid of 2000 finite volumes. Update of water level Δh and flow velocity Δu as a function of the iteration count. Computations dated Feb 2000.*

the equations inside the branches), then branches can have different initial water levels. The difference in water level at junctions is handled through the underrelaxation of the coupling conditions, just like the difference between initial water level and imposed water level is handled at water-level boundaries. (NB, coupling conditions at junctions can be viewed as internal boundary conditions connecting branches. In a DD approach they are treated in largely the same way as ordinary (external) boundary conditions that connect the model with the outer world.)

In Van Der Baan (1998) the DD extension of a previous version of SUPSUB to a network of 1D branches is described. I don't remember how Stefan initialized his steady-state computations. Very likely some form of still-water flow (with water level per branch?) has been used.

[28] Applying boundary conditions directly to an initial still-water flow solution may create very large non-smooth transient solution behavior (shocks) at boundaries that may cause serious convergence problems.

The longer the domain (in terms of number of finite volumes), the longer it takes for the information from the boundaries to be felt throughout the domain and 'overwrite' the initial still-water solution. Indeed, in Borsboom (1999c) where the grids that were considered all consisted of 1000 finite volumes, it is reported that a steady-state computation of nearly the same problem (Preissmann slot on top of pipe of $0.0004\mathrm{m}$ instead of $0.0002\mathrm{m}$) on an equally non-uniform (but twice as coarse) grid took only 431 iterations. This result is slightly older than those reported in Figure 5; the latter may have been obtained with a (slightly) improved version of SUPSUB. This would explain why the difference in number of iterations is less than the expected factor 2.

Given a certain application (geometry, flow conditions, ...), I expect the number of iterations of a steady-state computation to be roughly proportional to the number of grid cells. An easy way to strongly accelerate steady-state computations is therefore to first compute the solution on a fairly coarse grid, and use the interpolation of that solution to a finer grid as initial condition for the steady-state computation on that finer grid. Because the initial solution is then already close to the final solution on that finer grid *and* satisfies the boundary conditions, the (usually long) searching phase will be very small. Repeat the process until the desired grid resolution is reached. This is obviously a simple form of multi-grid acceleration.

Steady-state convergence will be much faster in case of the computation of a series of steady-state solutions for a number of successive boundary conditions. The computation of the first steady state, starting from a still-flow solution, will still take some time. In the computation of each next steady state, however, the initial solution can be taken equal to the previously determined steady state, which satisfies the discretized flow equations (and so must be sufficiently smooth, by construction) and differs at the boundaries only slightly from the next imposed conditions.

5.1.2 Initialization of unsteady solution processes

Solving the systems of unsteady equations per time step generally requires much less iterations than solving systems of steady-state equations. This is because the difference between solutions at consecutive time levels is (or at least should be, for reasons of numerical accuracy) relatively small, but also because these systems of equations are easier to solve due to the discretized time derivative that both increases the diagonal of the matrices and decreases linearization errors (because a term, the time discretization, is added that is linear in the unknowns). The smaller the time step (in terms of dimensionless Courant numbers, i.e., the smaller $u\Delta t/\Delta x$ and $\sqrt{gA/W}\Delta t/\Delta x$), the faster the convergence per time step.

In unsteady SUPSUB flow simulations, the initial solution of the iterative solution procedure per time step is taken equal to the solution of the previous time level just computed. This is a reasonable but not a very good choice. Taking the solution at the next time level initially equal to the one at the previous time level 'freezes' so to speak momentarily the dynamic behavior of the flow. Although the unsteady flow behavior is fully restored once the discretized unsteady flow equations per time step are solved, a relatively large amount of work is required to do so. In other words, although the final, fully converged solution at each next time level is by no means affected by the initial solution of the iterative solver per time step, the number of iterations required to get to those final solutions is. That number tends to be relatively large when the initial conditions are not very close to the final solutions[29].

[29] Such initial conditions lead to relatively large errors in the nonlinear systems of equations to be solved, i.e., the

There is a simple way to improve the initial conditions in unsteady computations: don't take them equal to the solution at previous time levels, but take them equal to a combination of the solutions at several previous time levels. In other words, don't use constant extrapolations that are only zeroth-order accurate, but use linear, quadratic, cubic, ... extrapolations that are first-, second-, third-, ...-order accurate. In unsteady computations, the extrapolation of the evolution in time already computed should give fairly good estimates of the solutions to be determined at next time levels.

To get an idea of what we are talking about, consider the model equation:

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0 \ .$$

Inserting in this equation some Fourier mode $c = C(t)\exp(ikx)$, we obtain for amplitude C of this solution mode the ordinary differential equation:

$$\frac{\partial C}{\partial t} + iukC = 0 \ , \tag{21}$$

whose evolution from previous time level t^{n-1} to the next time level t^n over the time interval $\Delta t = t^n - t^{n-1}$ is:

$$C^n = C^{n-1}\exp(-iuk\Delta t) \ ,$$

i.e., the exact phase change[30] over a time step of size Δt is:

$$g_{\text{exact}} = \ln\left(\frac{C^n}{C^{n-1}}\right) = -iuk\Delta t \ . \tag{22}$$

In line with the θ-weighted time integration scheme of SUPSUB with θ at most only slightly larger than $1/2$, cf. Section 3. we use the Crank-Nicolson scheme for the numerical approximation of (21):

$$(1 + 1/2\, iuk\Delta t)C^n = (1 - 1/2\, iuk\Delta t)C^{n-1} \ .$$

From this we obtain:

$$g_{\text{CN}} = \ln\left(\frac{1 - 1/2\, iuk\Delta t}{1 + 1/2\, iuk\Delta t}\right) \ . \tag{23}$$

By combining (22) and (23) we get the relative error of the Crank-Nicolson time-integration scheme:

$$err_{\text{CN}} = \left|\frac{g_{\text{CN}}}{g_{\text{exact}}} - 1\right| \ . \tag{24}$$

iteration processes per time step start with solutions that inside the domain do *not* satisfy the discretized equations. See the introduction of Section 5.1.1 of why this is considered important. On the other hand, it is likely that initializing the iteration proces for the solution at the next time level with the solution at the previous time level (or, better, a combination of the solutions at several previous time levels, see below) will in general lead to initial convergence errors of modest size that, more importantly, are reasonably smooth.

[30] In general we would have both phase changes and amplitude changes. The former due to (wave) propagation and convection, the latter due to dissipation (viscosity, bottom friction), sources and sinks. All possible to include in the analysis, but here we restrict ourselves for illustration purposes to a simple example.

This error behavior is shown by the black line in Figure 6. We see that the error is (much) smaller than about 5% for $uk\Delta t$ (much) smaller than 1. Notice that for $uk\Delta t \lessapprox 1$ the error behavior scales with the square of $uk\Delta t$, reflecting the second-order accuracy of the Crank-Nicolson method.

For sufficient accuracy in time, time step Δt should be such that $uk\Delta t$ is sufficiently small (at least $\lessapprox 1$), but this is *only* required for those k associated with unsteady flow behavior (k_{unsteady}). The size of wavenumbers associated with steady flow behavior (k_{steady}) does not play a role in the time integration and hence is irrelevant in choosing an appropriate time step. The k_{steady} do play a role of course in choosing the spatial grid resolution. In order for the spatial grid to be able to accommodate all relevant solution modes, its size should be sufficiently small compared to $\min(1/k_{\text{unsteady}}, 1/k_{\text{steady}})$[31]. In applications where the k_{steady} are typically much larger than the k_{unsteady} we have that $k_{\text{steady}}u\Delta t$ may be $\gg 1$ while at the same time $k_{\text{unsteady}}u\Delta t \lessapprox 1$. The former is not a problem; only the latter is relevant for time accuracy. These are the applications where the use of a fully nonlinearly implicit time-integration scheme, if designed properly, will be much more efficient than an explicit, semi-implicit, or even a linearized implicit scheme, because the Courant-type stability condition of the form $u\Delta t/\Delta x < O(1)$ that would then apply would pose limitations to the size of the time step that are much larger than the accuracy requirement $k_{\text{unsteady}}u\Delta t \lessapprox 1$, cf. Borsboom (2019a).

From the above it follows that for the *initialization of unsteady solution processes*, estimations of the solution at the next time level only need to be accurate for solution modes satisfying $uk\Delta t \lessapprox 1$. Solution modes in the range $uk\Delta t \gtrapprox 1$ can be assumed to be (close to) steady. If not, then time step Δt is actually too large.

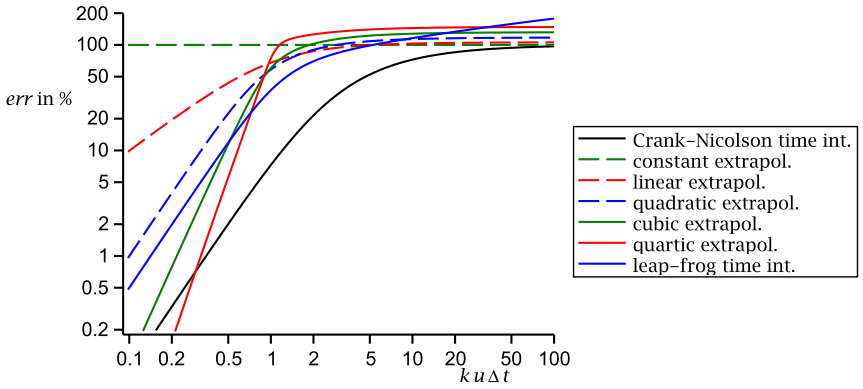

Figure 6: Error (per time step) in implicit Crank-Nicolson time integration, and in several explicit estimations of implicit Crank-Nicolson, as a function of Courant-like number $uk\Delta t$. Figure taken from Borsboom (2012–2013, 2019).

In SUPSUB the solutions at the next time levels n are initialized by means of the solutions at the

[31] This is a strong simplification of what is encountered and needs to be dealt with in practice, but adequate to illustrate the principles. First of all, the separation between steady modes and unsteady modes may not be that clear, and will generally vary in time and in space, just like the wavenumber ranges of steady and unsteady modes. Secondly, the separation between steady modes and unsteady modes is, like the required grid resolution, generally different in different directions (not relevant here where we only consider 1D models). Thirdly, different flow dynamics (waves, currents, drying and flooding, boundary-condition induced, effect of geometry, ...) come with their own set of k_{unsteady} and k_{steady}. There is also the effect of nonlinear flow behavior on k_{unsteady} and k_{steady}. Things like the steepening of waves (ultimately leading to bores and hydraulic jumps) and strong flow variations (e.g., those caused by strong geometry variations) tend to create flow features of very small spatial scales.

previous time levels $n-1$. Per Fourier-mode component:

$$C^{n,0} = C^{n-1} \ . \tag{25}$$

Similar to (22) and (23) we therefore have:

$$g_{\text{constant}} = \ln\left(\frac{C^{n,0}}{C^{n-1}}\right) = 0 \ , \tag{26}$$

and hence:

$$err_{\text{constant}} = \left|\frac{g_{\text{constant}}}{g_{\text{CN}}} - 1\right| = 1 \ . \tag{27}$$

Note that the purpose of (25) (and similar expressions) is to provide an estimate of C^n as determined by the Crank-Nicolson scheme, *not* to provide an alternative time integration scheme. For this reason we compare in (27) g_{constant} with g_{CN}, *not* with g_{exact}[32].

It is clear that (25) is not a very accurate initialization: a 100% error for any value of $uk\Delta t$, cf. the dashed green line in Figure 6. It is easy to see how that can be improved: use 2, 3, 4, 5, ... previous time levels to construct a linear, quadratic, cubic, quartic, ... extrapolation. The result is a first-, second-, third-, fourth-, ...-order accurate estimation of C^n of increasing accuracy for $uk\Delta t < 1$, cf. the dashed blue and red line and solid green and red line in Figure 6.

As an alternative to extrapolation we may consider the use of an explicit time-integration scheme to come up with an estimate $C^{n,0}$ of the solution at the next time level C^n, for example the second-order accurate leap-frog scheme:

$$C^{n,0} = C^{n-2} - 2\,iuk\Delta t\,C^{n-1} \ .$$

The advantage of this iterative-solver initialization is that it is more accurate than second-order extrapolation (compare the solid blue line and the dashed blue line in Figure 6) and requires only two instead of three previous time levels. A possible disadvantage is that it requires the (explicit) evaluation of the space discretization at t^{n-1}. That is not a disadvantage when that evaluation would be required anyway, e.g., in a variable/adaptive theta-method, cf. Section 3.

All explicit initialization methods fail for $uk\Delta t \gtrapprox 1$. As explained above, that should not be a problem, since the values of k in that range should mainly be associated with steady-state solution modes. If not, the time step is too large for the numerical time integration to be accurate.

The improved explicit initialization methods are in particular accurate for $uk\Delta t \ll 1$, i.e., for unsteady solution modes with long length scales (small k) that are of a more global character. Those are the modes that, given some pseudo time step, are the slowest to converge, because their evolution in time (also in pseudo time) is relatively slow (add some reference?). An accurate initialization will therefore strongly reduce the number of iterations required to solve these long modes. Shorter modes tend to convergence rapidly *and* (should) contain less and less unsteady information, i.e., an inaccurate initialization is here less of a problem. In conclusion, we expect a strong efficiency gain in unsteady computations from an easy-to-apply improvement of the constant-extrapolation initialization of SUPSUB (e.g., quadratic extrapolation, leap-frog method).

[32] NB, an expression like (24), where numerical behavior is compared with exact behavior, is suitable for the error in time-integration schemes. It is not suitable for the error in estimations of the solution at the next time level of a time-integration scheme. In that case, comparison with the behavior of that time-integration scheme is required, as in (27).

The usefulness of the explicit next-time-level-solution estimations shown in Figure 6 has been investigated and confirmed in a 1D domain-decomposition study. In that study the implicit coupling between two subdomains is updated explicitly and solved iteratively in order to allow parallelization. The improvement of the explicit initialization of the implicit coupling between the two subdomains that was found, cf. Borsboom (2009), is in full agreement with the results of Figure 6.

6 References

Aureli, F., S. Dazzi, A. Maranzoni and P. Mignosa, 2015. "Validation of single- and two-equation models for transient mixed flows: a laboratory test case." *J. Hydraul. Res.* 53 (4): 440–451. DOI: 10.1080/00221686.2015.1038324.

Boom, P. D. and D. W. Zingg, 2018. "Optimization of high-order diagonally-implicit Runge–Kutta methods." *J. Comput. Phys.* 371: 168–191. DOI: 10.1016/j.jcp.2018.05.020.

Borsboom, M., 1998. "Development of an error-minimizing adaptive grid method." *Appl. Numer. Math.* 26 (1–2): 13–21. DOI: 10.1016/S0168-9274(97)00077-9.

Borsboom, M., 1999a. "Nieuwe versie SUPSUB." Memo, 19 maart 1999.

Borsboom, M., 1999b. "SOBEK, betere schaling van Cflipse." Memo, February 24, 1999.

Borsboom, M., 1999c. "SUPSUB stromingsberekeningen voor een 'great man-made river'." Memo, August 2, 1999.

Borsboom, M., 2001. "Development of a 1-D error-minimizing moving adaptive grid method." In A. Vande Wouwer, P. Saucez and W. E. Schiesser, eds., *Adaptive Method of Lines*, chap. 5, pages 139–180. CRC Press.

Borsboom, M., 2002. "A finite volume method designed for error analysis." In R. Herbin and D. Kröner, eds., *Finite Volumes for Complex Applications III*, pages 705–712. Hermes Penton Science.

Borsboom, M., 2005. "Wave run-up in TRITON." Memo, September 2005.

Borsboom, M., 2005–2009, 2019. "Stability/accuracy analysis of theta method and its better alternative DIRK2/Rosenbrock2." MAPLE worksheet, July 2005–January 2009, August 2019.

Borsboom, M., 2009. "Domain decomposition and (versus?) (un)structured grids - About modeling flexibility and multi-scale modeling (and explicit DD coupling) in hydrodynamic applications." Presentation for section Hydro Software, DSC, September 7, 2009.

Borsboom, M., 2012–2013, 2019. "Analysis of extrapolation in time for better initial guess at DD interfaces." MAPLE worksheet, September 2012–January 2013, July–August 2019.

Borsboom, M., 2013–2016. "Analysis of several artificial porosity formulations to model drying and flooding (run-up/run-down)." MAPLE worksheet, July–August 2013, November 2015, April 2016.

Borsboom, M., 2019a. "About fully implicit time integration and nonlinear iterative solvers (per steady state or per time step)." Presentation for D-Flow FM development team, May 7, 2019.

Borsboom, M., 2019b. "Quick check of drying-and-flooding procedure based on exponential artificial porosity with and without a very tiny Preissmann slot or Preissmann funnel to avoid extremely small channel widths." MAPLE worksheet, May 2019.

Borsboom, M. and M. J. Van Der Marel, 1992. *Development of an adaptive grid procedure that compensates for discretisation errors, with an application to the 1D Burgers' equation*. Tech. Rep. Q1346, Delft Hydraulics.

Brown, D. A. and D. W. Zingg, 2016. "Performance of a Newton–Krylov–Schur algorithm for solving steady turbulent flows." *AIAA J.* 54 (9): 2645–2658. DOI: 10.2514/1.J054513.

Candy, A. S., 2017. "An implicit wetting and drying approach for non-hydrostatic baroclinic flows in high aspect ratio domains." *Adv. Water Resour.* 102: 188–205. DOI: 10.1016/j.advwatres.2017.02.004.

Ceze, M. and K. J. Fidkowski, 2015. "Constrained pseudo-transient continuation." *Int. J. Numer. Methods Eng.* 102 (11): 1683–1703. DOI: 10.1002/nme.4858.

Chisholm, T. T. and D. W. Zingg, 2009. "A Jacobian-free Newton–Krylov algorithm for compressible turbulent fluid flows." *J. Comput. Phys.* 228 (9): 3490–3507. DOI: 10.1016/j.jcp.2009.02.004.

Deuflhard, P., 2011. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*. Computational Mathematics. Springer.

Frank, J., W. Hundsdorfer and J. G. Verwer, 1997. "On the stability of implicit-explicit linear multistep methods." *Appl. Numer. Math.* 25 (2–3): 193–205. DOI: 10.1016/S0168-9274(97)00059-7.

Gropp, W., D. Keyes, L. C. McInnes and M. D. Tidriri, 2000. "Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD." *The International Journal of High Performance Computing Applications* 14 (2): 102–136. DOI: 10.1177/109434200001400202.

Hicken, J. E. and D. W. Zingg, 2009. "Globalization strategies for inexact-Newton solvers." In *Proc. 19th AIAA CFD Conf.*, pages AIAA 2009-4139. AIAA. DOI: 10.2514/6.2009-4139.

Hof, B. Van 't and E. A. H. Vollebregt, 2005. "Modeling of wetting and drying of shallow water using artificial porosity." *Int. J. Numer. Methods Fluids* 48 (11): 1199–1217. DOI: 10.1002/fld.959.

Kelley, C. T. and D. E. Keyes, 1998. "Convergence analysis of pseudo-transient continuation." *SIAM J. Numer. Anal.* 35 (2): 508–523. DOI: 10.1137/S0036142996304796.

Kennedy, C. A. and M. H. Carpenter, 2016. *Diagonally implicit Runge-Kutta methods for ordinary differential equations. A review*. Tech. Rep. NASA/TM–2016–219173, NASA.

Kerger, F., S. Erpicum, B. J. Dewals, P. Archambeau and M. Pirotton, 2011. "1D unified mathematical model for environmental flow applied to steady aerated mixed flows." *Adv. Eng. Softw.* 42 (9): 660–670. DOI: 10.1016/j.advengsoft.2011.04.012.

Knoll, D. A. and D. E. Keyes, 2004. "Jacobian-free Newton-Krylov methods: a survey of approaches and applications." *J. Comput. Phys.* 193 (2): 357–397. DOI: 10.1016/j.jcp.2003.08.010.

León, A. S., M. S. Ghidaoui, A. R. Schmidt and M. H. Garcia, 2009. "Application of Godunov-type schemes to transient mixed flows." *J. Hydraul. Res.* 47 (2): 147–156. DOI: 10.3826/jhr.2009.3157.

Maranzoni, A., S. Dazzi, F. Aureli and P. Mignosa, 2015. "Extension and application of the Preissmann slot model to 2D transient mixed flows." *Adv. Water Resour.* 82: 70–82. DOI: 10.1016/j.advwatres.2015.04.010.

Mavriplis, D. J., 2018. "A residual smoothing strategy for accelerating Newton method continuation." URL https://arxiv.org/abs/1805.03756v1.

Piomelli, U., 2014. "Large eddy simulations in 2030 and beyond." *Phil. Trans. R. Soc. A* 372 (2022). DOI: 10.1098/rsta.2013.0320.

Rodi, W., G. Constantinescu and T. Stoesser, 2013. *Large-Eddy Simulation in Hydraulics*. CRC Press.

Savant, G., C. Berger, T. O. McAlpin and J. N. Tate, 2011. "Efficient implicit finite-element hydrodynamic model for dam and levee breach." *J. Hydraul. Eng.* 137 (9): 1005–1018. DOI: 10.1061/(ASCE)HY.1943-7900.0000372.

Stelling, G. S. and S. P. A. Duinmeijer, 2003. "A staggered conservative scheme for every Froude number in rapidly varied shallow water flows." *Int. J. Numer. Methods Fluids* 43 (12): 1329–1354. DOI: 10.1002/fld.537.

Van De Langemheen, W., 1994. *Simulation of 1-D open channel flow using a moving adaptive grid algorithm*. Tech. Rep. X125, Delft Hydraulics.

Van Der Baan, S., 1998. *A domain-decomposition approach for the numerical computation of sub- and supercritical flow in networks*. Master's thesis, Utrecht University.

Van Reeuwijk, M., 2011. "A mimetic mass, momentum and energy conserving discretization for the shallow water equations." *Comput. Fluids* 46 (1): 411–416. DOI: 10.1016/j.compfluid.2011.01.006.

Verwer, J. G., E. J. Spee, J. G. Blom and W. H. Hundsdorfer, 1997. *A second order Rosenbrock method applied to photochemical dispersion problems*. Tech. Rep. MAS-R9717, CWI.

Von Neumann, J. and R. D. Richtmyer, 1950. "A method for the numerical calculation of hydrodynamic shocks." *J. Appl. Phys.* 21 (3): 232–237. DOI: 10.1063/1.1699639.