VORtech Computing

Industrial and Scientific Computing

P.O. Box 260
2600 AG Delft
The Netherlands

tel. +31(0)15-285 0125
fax. +31(0)15-285 0126
vortech@vortech.nl

Numerical Methods for the Stationary Shallow Water Equations

Master Thesis

Technical Report TR07-03

Date

July 4, 2007

Author(s)

F.L.M van Wageningen-Kessels

Thesis committee	TUDelft	VORtech
	Prof.dr.ir. C. Vuik	Ir. J. Dijkzeul
	Dr.ir. M. van Gijzen	Dr.ir. B. van 't Hof
	Dr. H. Schuttelaars	

By Order of

TU Delft
Delft University of Technology

Summary

QuickFlow is a software program that can be used to analyze the effects of an intervention in a river. These interventions are for example a hole in the bottom resulting from grit winning or a decrease of the bottom at the fairway to increase the capacity of the river. QuickFlow computes an approximation of the new stationary solution of the shallow water equations.

We have improved the methods used by QuickFlow and introduced some new methods. We focussed on two issues: finding the correct solution and finding this solution in little computing time.

The space discretization was copied from an other program: WAQUA. First we have made a tool which checks whether this discretization has been copied correctly from WAQUA. With the use of this tool we have repaired some errors, mainly around the boundaries. Second the possibilities to apply continuation were studied. Third, we have added two methods to analyze the stability of a solution. The above improvements made it possible to approximate the exact solution close enough.

The main improvements concerning the computing time are related to three parts of the computation: 1) the Euler backward method for time integration, 2) the quasi Newton method for linearization and 3) a preconditioned iterative method to solve the linear system.

We have reduced the number of pseudo time steps for the Euler backward method, by using a larger pseudo time step size. The line search process in the quasi Newton method was changed and we have studied whether Broyden's method can be used to approximate the Jacobian. Furthermore, we have studied several iterative methods to solve the linear system. We have replaced the direct method by an LU preconditioned BiCGSTAB method. QuickFlow uses LU decomposition with static pivotting. Furthermore, we have studied when the LU decompostion and the Jacobian should be recomputed and when the old matrices can be used.

We have tested the above improvements for several test problems. At the start of this project it was not possible to solve the most difficult problems. At this moment all test problems can be solved within at most 70 seconds. This computing time was achieved for a test problem with 7718 grid points and a decrease of the bottom depth of 4 meters at the fairway. One of the easiest realistic test problems has 1471 grid points and can now be solved in 4 seconds. These computing times have been reduced with a factor 3 to 40 since the beginning of the project.

Frequently used symbols

Table 1: Symbols

Symbol	Description
$\mathbf{F}(\mathbf{x})$	stationary shallow water equations: $\mathbf{F}(\mathbf{x}) = \mathbf{0}$
$\mathbf{F}'$	Jacobian of $\mathbf{F}$
$\mathbf{x}$	solution vector consisting of U and V velocities and water levels
n	number of pseudo time step
k	number of Newton iteration
$\mathbf{s}$	search direction quasi Newton iteration: $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\, \mathbf{s}^k$
α	line search coefficient
M	mass matrix
Δt	pseudo time step
A	matrix of linearized function: $\mathrm{A} = \frac{1}{\Delta t}\mathrm{M} - \mathbf{F}'(\mathbf{x}^k)$
$\mathbf{b}$	right hand side of linearized function: $\mathbf{b} = \mathbf{F}(\mathbf{x}^k) - \mathbf{F}'(\mathbf{x}^k)\mathbf{x}^k$

Contents

Chapter 1

Introduction

1.1 Background

Water management is very important in a densely populated country with many rivers like the Netherlands. Maintainance and improvement of dykes, riverbeds etc. needs constant attention. In the design of these dykes and riverbeds software is used to predict the flow velocities and water levels.

This Master thesis project is conducted at VORtech, which is an engineering and software company with a lot of mathematical expertise. Rijkswaterstaat is a Dutch governemental institute that is engaged in water management projects. VORtech Computing developes software that is used to do the necessary computations.

WAQUA is a software package that has been developed by Rijkswaterstaat and other companies to predict flows of rivers, seas and oceans. WAQUA uses the two dimensional shallow water equations to compute the flow velocity and water level and its development over time very accurately. One of the major drawbacks of WAQUA is the relatively large computing time to find a stationary solution.

For this purpose QuickFlow was developed. QuickFlow is a program that is intended to solve the stationary shallow water equations for application to rivers in the Netherlands. Its purpose is to quickly find the new stationary solution after an intervention like rising a dyke or a change in the level of the bottom. The results need to be accurate, but not as accurate as with WAQUA. Whenever the user thinks a certain intervention can be applied, this should be checked with WAQUA.

QuickFlow is tightly linked to WAQUA. QuickFlow is based on WAQUA, and needs some input from it, but it has the advantage that it finds a solution much quicker than WAQUA. Towards the end of the design process of an intervention the proposed changes still need evaluation by WAQUA. In Appendix A, the relation between WAQUA and QuickFlow and the general setup of QuickFlow are shown schematically.

At the start of this project QuickFlow could find solutions that look pretty much like the WAQUA-solution for relatively easy problems, such as a small part of a river with a simple geometry. When the geometry of a river part is more complex and/or the interventions 'large', however, was not possible at all to find the stationary solution.

We have first studied the literature to understand WAQUA and QuickFlow and to study some methods that might improve the performance of QuickFlow. The results from this literature research are summarized in the next sections. For a more detailed overview we refer to [5]. In the following chapters we have described the actual improvements we have made to QuickFlow.

1.2 2D shallow water equations

QuickFlow uses the same equations, boundary conditions and discretization as WAQUA does. These are described in more detail in [6, Chapter 2, 4 and 6]. QuickFlow approximates the solution of the two dimensional shallow water equations on a curvilinear grid. On a Cartesian (x, y) grid the equations are as follows:

$$\begin{aligned}
\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} - f\,V + g\frac{\partial \zeta}{\partial x} - \frac{\tau_{\mathrm{bottom},x}}{\rho_0\,H} - \nu_{\mathrm{H}}\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right) &= 0,\\
\frac{\partial V}{\partial t} + U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} + f\,U + g\frac{\partial \zeta}{\partial y} - \frac{\tau_{\mathrm{bottom},y}}{\rho_0\,H} - \nu_{\mathrm{H}}\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right) &= 0,\\
\frac{\partial \zeta}{\partial t} + \frac{\partial (HU)}{\partial x} + \frac{\partial (HV)}{\partial y} &= 0,
\end{aligned}$$

with:

U	velocity in x-direction (m/s)
V	velocity in y-direction (m/s)
ζ	water level (m)
H	total water depth (m)
t	time (s)
f	Coriolis parameter (s^{-1})
g	acceleration due to gravity (m/s^2)
ρ_0	reference density (kg/m^3)
τ_{bottom}	shear stress at bottom $(kg/m/s^2)$
ν_{H}	horizontal eddy viscosity (m^2/s).

The first two equations are the U- and V-momentum equations respectively, the last equation is called the continuity equation.

Boundary conditions

We apply free slip conditions on the closed boundaries. At the inflow boundary we prescribe the discharge. At the outflow boundary we prescribe the water level.

Discretization

The spatial discretization of the two dimensional shallow water equations in QuickFlow is copied from WAQUA. For the momentum equation finite difference schemes (both central and upwind) are used. For the continuity equation a finite volume method is used. We write the resulting discretized two dimensional shallow water equations as:

$$\mathrm{M}\,\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}).$$

1.2.1 Numerical methods in QuickFlow

The equations resulting from the discretization are solved using several numerical methods. In Figure 1.1 a general overview of these methods is given. In Figures 1.2 – 1.6 more detailed schematic views of the numerical methods in QuickFlow are shown. In Equation 1.2.1 we give an overview of the resulting equations. Here and in the rest of this report we use the symbols in Table 1 on page 5.

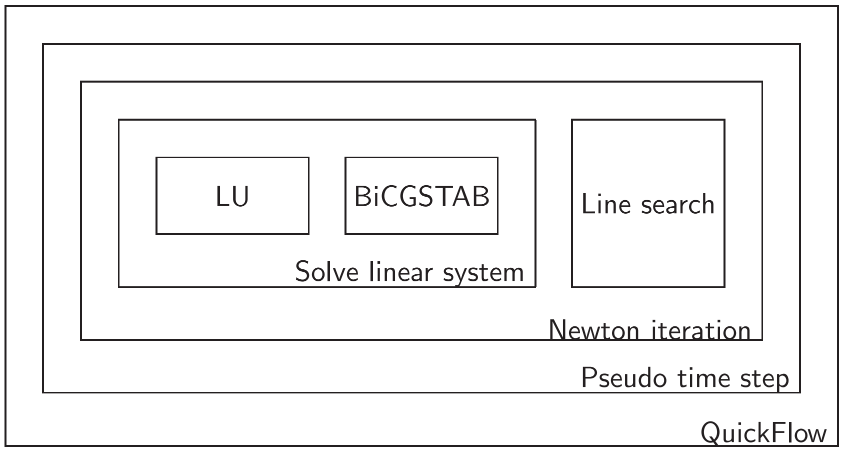


Figure 1.1: Overview of the numerical methods in QuickFlow.

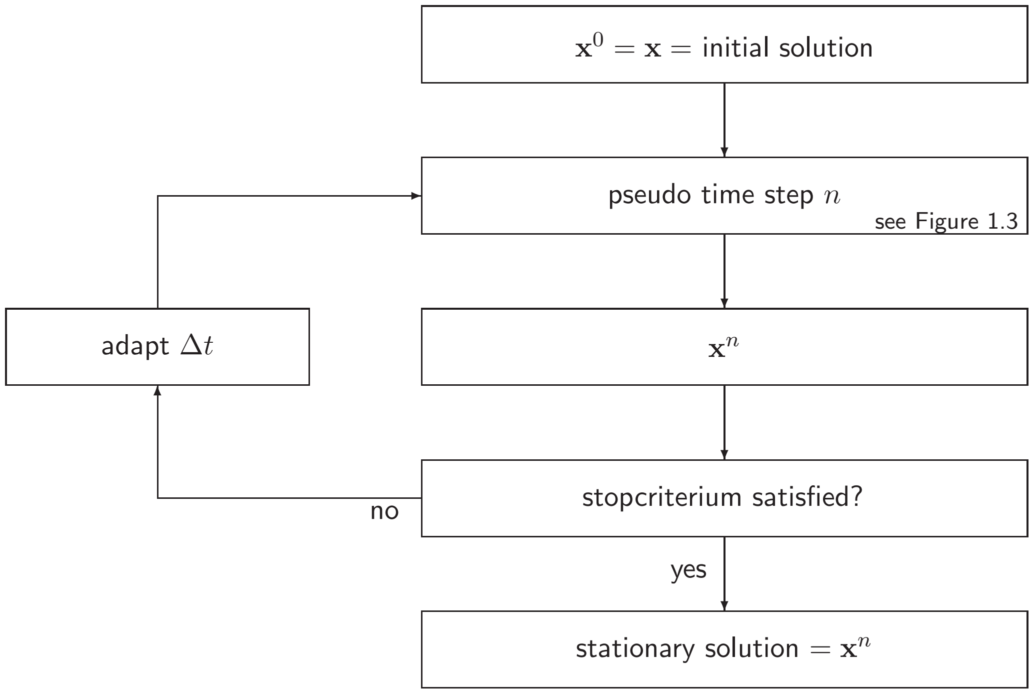


Figure 1.2: Overview of the numerical methods in QuickFlow.

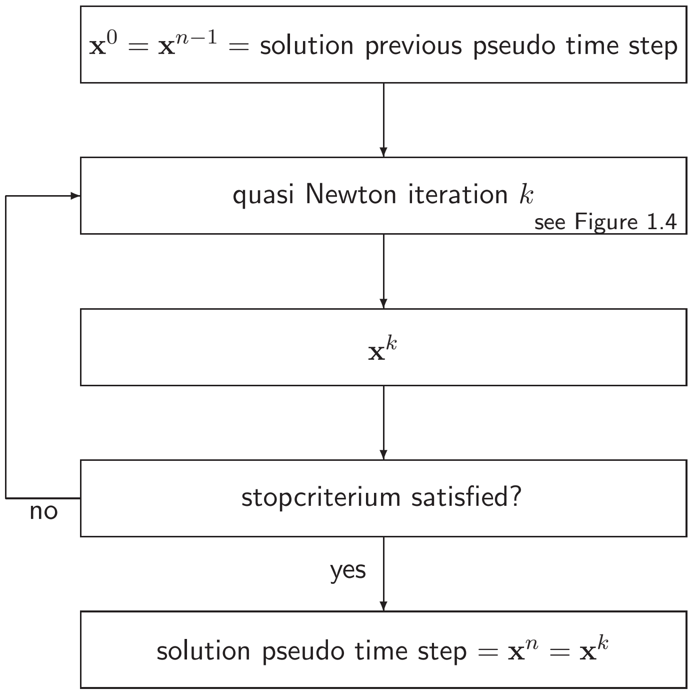

$\mathbf{x}^0 = \mathbf{x}^{n-1} =$ solution previous pseudo time step
quasi Newton iteration k
see Figure 1.4
$\mathbf{x}^k$
stopcriterium satisfied?
no
yes
solution pseudo time step $= \mathbf{x}^n = \mathbf{x}^k$

Figure 1.3: Overview of the processes in one pseudo time step.

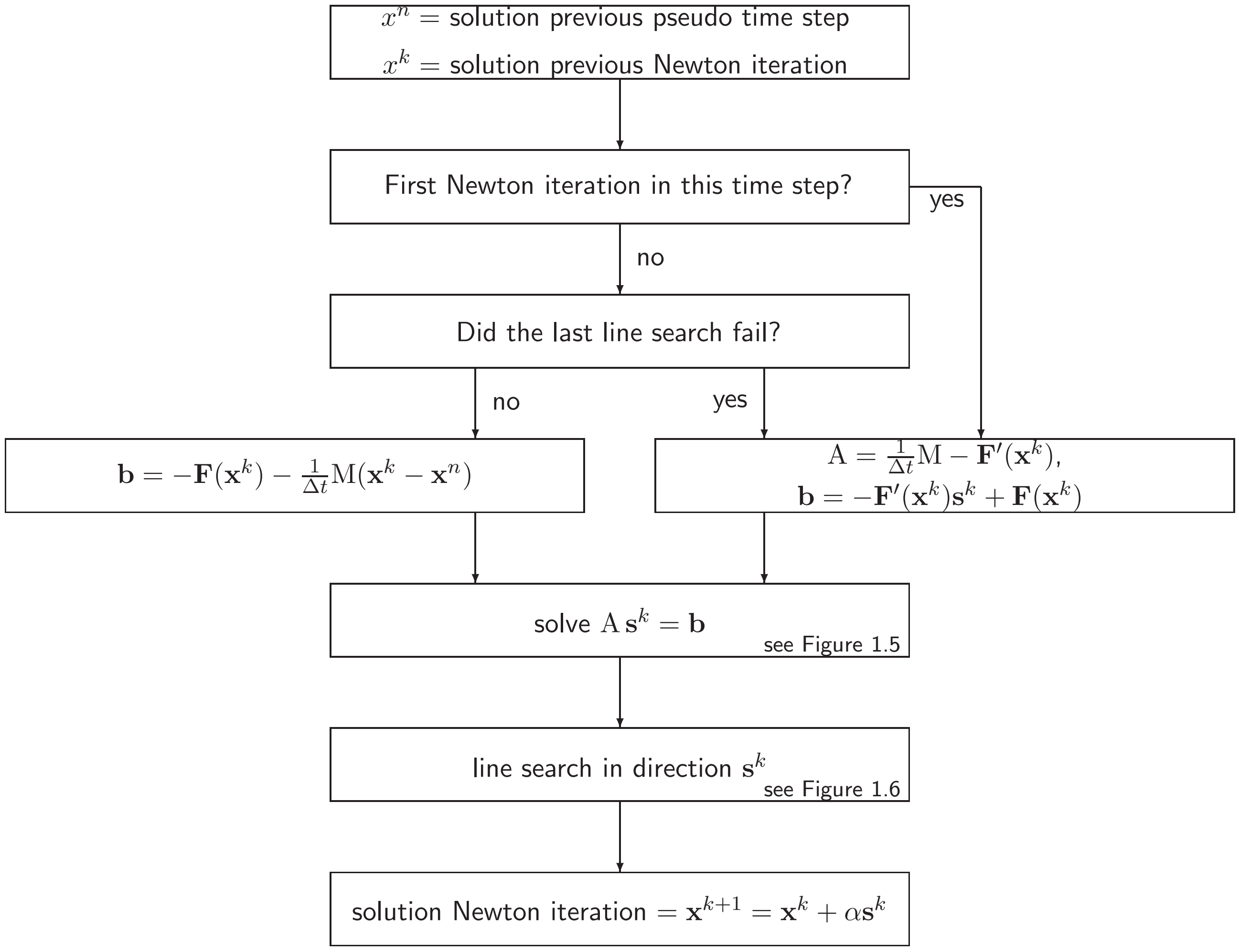

x^n = solution previous pseudo time step
x^k = solution previous Newton iteration
First Newton iteration in this time step?
yes
no
Did the last line search fail?
no
yes
$\mathbf{b} = -\mathbf{F}(\mathbf{x}^k) - \frac{1}{\Delta t}\mathrm{M}(\mathbf{x}^k - \mathbf{x}^n)$
$\mathrm{A} = \frac{1}{\Delta t}\mathrm{M} - \mathbf{F}'(\mathbf{x}^k)$,
$\mathbf{b} = -\mathbf{F}'(\mathbf{x}^k)\mathbf{s}^k + \mathbf{F}(\mathbf{x}^k)$
solve $\mathrm{A}\,\mathbf{s}^k = \mathbf{b}$
see Figure 1.5
line search in direction $\mathbf{s}^k$
see Figure 1.6
solution Newton iteration = $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\mathbf{s}^k$

Figure 1.4: Overview of the processes in one quasi Newton iteration.

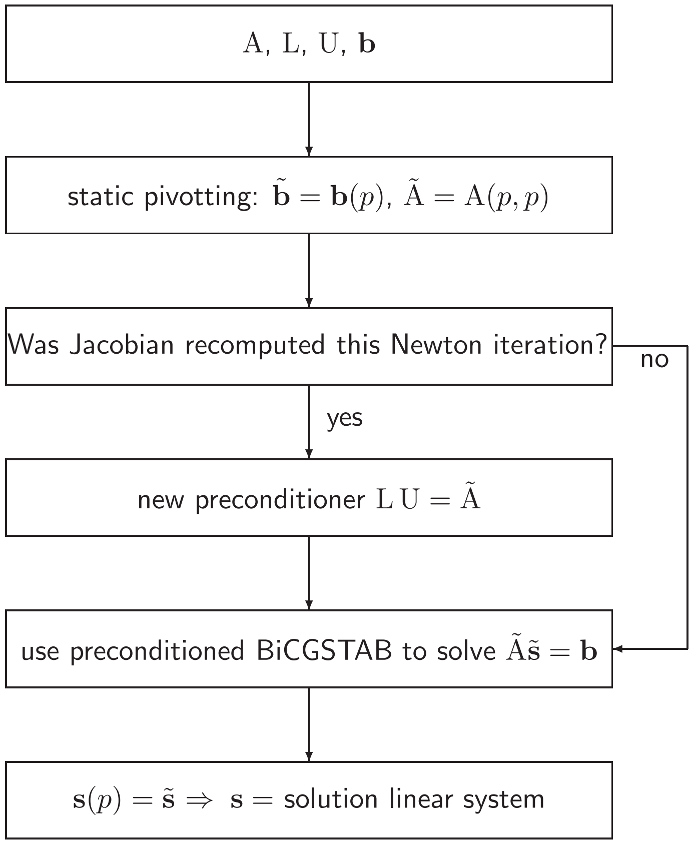

A, L, U, $\mathbf{b}$
static pivotting: $\tilde{\mathbf{b}} = \mathbf{b}(p)$, $\tilde{\mathrm{A}} = \mathrm{A}(p,p)$
Was Jacobian recomputed this Newton iteration?
no
yes
new preconditioner $\mathrm{L\,U} = \tilde{\mathrm{A}}$
use preconditioned BiCGSTAB to solve $\tilde{\mathrm{A}}\tilde{\mathbf{s}} = \mathbf{b}$
$\mathbf{s}(p) = \tilde{\mathbf{s}} \Rightarrow \mathbf{s} =$ solution linear system

Figure 1.5: Overview of the processes in solving the linear system.

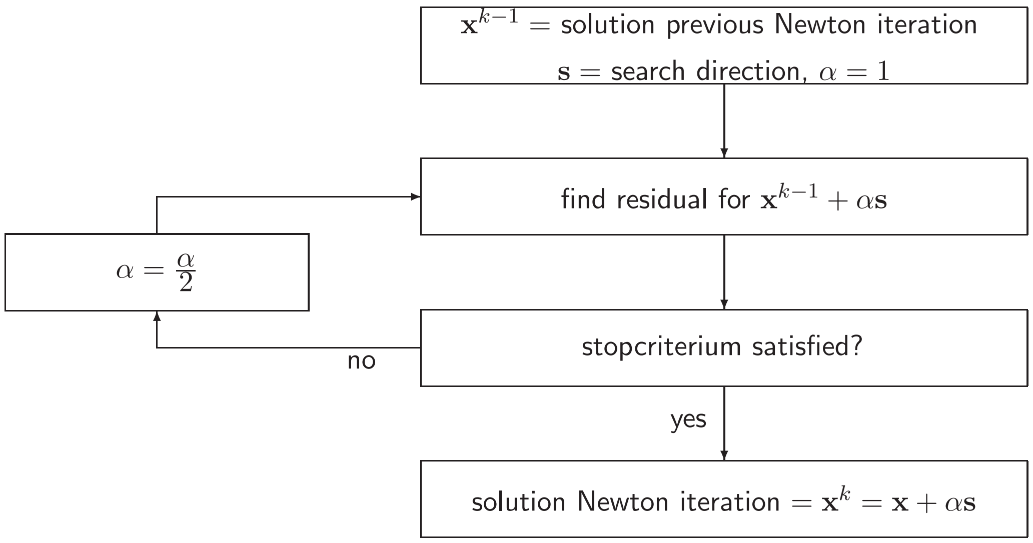

$\mathbf{x}^{k-1} =$ solution previous Newton iteration
$\mathbf{s} =$ search direction, $\alpha = 1$
find residual for $\mathbf{x}^{k-1} + \alpha\mathbf{s}$
$\alpha = \frac{\alpha}{2}$
stopcriterium satisfied?
no
yes
solution Newton iteration $= \mathbf{x}^k = \mathbf{x} + \alpha\mathbf{s}$

Figure 1.6: Overview of the processes in the line search.

$$
\begin{aligned}
\frac{d\mathbf{x}}{dt} &= \mathbf{F}(\mathbf{x}), \\
&\downarrow \quad \text{Euler backward} \\
\frac{1}{\Delta t}\mathrm{M}(\mathbf{x}^{n+1}-\mathbf{x}^{n}) &= \mathbf{F}(\mathbf{x}^{n+1}), \\
&\downarrow \quad \mathbf{s}^{n} = \mathbf{x}^{n+1}-\mathbf{x}^{n} \\
\frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{n} &= \mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{n}), \\
&\downarrow \quad \text{Newton's method} \\
\mathbf{s}^{k+1} &= \mathbf{s}^{k} - \left(\left(\frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{k}-\mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{k})\right)'\right)^{-1}\cdot \\
&\quad \left(\frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{k}-\mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{k})\right), \\
&\downarrow \quad \text{rewrite} \\
\left(\frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{k}-\mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{k})\right)'\mathbf{s}^{k+1} &= \left(\frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{k}-\mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{k})\right)'\mathbf{s}^{k}- \\
&\quad \frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{k}+\mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{k}), \\
&\downarrow \quad \left(\frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{k}-\mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{k})\right)' = \frac{1}{\Delta t}\mathrm{M}-\mathbf{F}'(\mathbf{x}^{n}+\mathbf{s}^{k}) \\
\left(\frac{1}{\Delta t}\mathrm{M}-\mathbf{F}'(\mathbf{x}^{n}+\mathbf{s}^{k})\right)\mathbf{s}^{k+1} &= \left(\frac{1}{\Delta t}\mathrm{M}-\mathbf{F}'(\mathbf{x}^{n}+\mathbf{s}^{k})\right)\mathbf{s}^{k}- \\
&\quad \frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^{k}+\mathbf{F}(\mathbf{x}^{n}+\mathbf{s}^{k}), \\
&\downarrow \quad \mathbf{x}^{k} = \mathbf{x}^{n}+\mathbf{s}^{k} \text{ and rewrite} \\
\left(\frac{1}{\Delta t}\mathrm{M}-\mathbf{F}'(\mathbf{x}^{k})\right)\mathbf{s}^{k+1} &= -\mathbf{F}'(\mathbf{x}^{k})\,\mathbf{s}^{k}+\mathbf{F}(\mathbf{x}^{k}), \\
&\downarrow \quad \mathrm{A} = \frac{1}{\Delta t}\mathrm{M}-\mathbf{F}'(\mathbf{x}^{k}) \text{ and } \mathbf{b} = -\mathbf{F}'(\mathbf{x}^{k})\,\mathbf{s}^{k}+\mathbf{F}(\mathbf{x}^{k}) \\
\mathrm{A}\,\mathbf{s}^{k+1} &= \mathbf{b}.
\end{aligned}
\tag{1.2.1}
$$

1.3 Research goals

The main goals of this project are:

1. improve robustness: QuickFlow should find an accurate stationary solution;
2. reduce computing time.

In order to achieve the first goal we have compared the results from WAQUA with the ones from QuickFlow for some relatively easy test problems, mostly without interventions. This is described in more detail in Chapter 2.

With larger interventions we still found inaccurate solutions. This is solved by continuation, as described in Chapter 3. During this part we discovered that the stationary solution can be unstable, for QuickFlow as well as when WAQUA is used. This is described in Chapter 4.

In the literature research we proposed several methods to reduce the computing time:

- apply (critical) damping boundary conditions to open boundaries;
- use an alternative for Euler backward time integration, with more explicit terms;
- use adaptive time stepping with local variations and/or with larger pseudo time steps for the continuity equation;
- use a quasi Newton method or other alternative for linearization;

The last method is implemented in QuickFlow, together with larger pseudo time steps and an iterative method (instead of a direct method) for solving the linear system. The reduction of computing times is discussed in more detail in Chapter 5.

1.3.1 Test problems

During this project we have used three test problems, with and without interventions. We have chosen for the non realistic but very easy Chezy gutter, the realistic and relatively easy Lek test problem and the relatively difficult Randwijk test problem. These test problems and the applied interventions are described in Appendix B.

Chapter 2

Correctness of the solution

Before reducing the computing times of QuickFlow, we will improve the robustness and reliability. Even for 'difficult' problems QuickFlow should find the stationary solution without errors.

In this chapter we only look at test problems without interventions. We suspect some programming errors, especially near the open boundaries. We find the errors by analyzing the residuals. Therefore we both analyze the residuals and the Jacobian, as they are computed by QuickFlow. We have developed a tool for this analysis, which is described in Appendix C.

To make it easier to find and resolve any errors we first combine two functions in the software. The combination of these functions is described in Section 2.1.

Secondly we search for errors and resolve them. This is described in Section 2.2.

2.1 Combining matrices in linearization

The implementation of the ADI-method (see [6, Chapter 5 and 7], [3]) in QuickFlow used to be more direct than it is now. The linearization resulted in two different matrices: $\mathrm{A}_{\mathrm{left}}$ and $\mathrm{A}_{\mathrm{right}}$. We have replaced them by one matrix $\mathbf{F}'(\mathbf{x})$, which is the Jacobian of $\mathbf{F}$ at $\mathbf{x}$.

It used to be programmed like this:

$$\frac{1}{\Delta t}\mathrm{M}(\mathbf{x}^{n+1} - \mathbf{x}^{n}) + \mathrm{A}_{\mathrm{left}}\, \mathbf{x}^{n+1} = \mathrm{A}_{\mathrm{right}}\, \mathbf{x}^{n} + \mathbf{b}.$$

We are looking for the stationary solution with $\mathbf{s}^n = \mathbf{x}^{n+1} - \mathbf{x}^n = \mathbf{0}$. Therefore, we can also use:

$$\frac{1}{\Delta t}\mathrm{M}(\mathbf{x}^{n+1} - \mathbf{x}^{n}) + \mathbf{F}'(\mathbf{x}^n)\, \mathbf{s}^n = \mathbf{b},$$
$$\text{with } \mathbf{F}'(\mathbf{x}^n) = \mathrm{A}_{\mathrm{left}} - \mathrm{A}_{\mathrm{right}}.$$

Note that this does not change the results substantially, it only makes the matlab code more clear and easier to overview.

2.2 Errors near outflow boundary

QuickFlow can compute either only the residuals $\mathbf{F}(\mathbf{x})$ and mass vector (for example to check for convergence) or it can compute a new Jacobian $\mathbf{F}'(\mathbf{x})$, right hand side $\mathbf{b}$ and mass matrix. to use in the quasi Newton iteration. If everything is correct $\mathbf{F}(\mathbf{x}) \approx \mathbf{F}'(\mathbf{x})\,\mathbf{x} - \mathbf{b}$ holds.

We first check for any errors in the construction of the residuals for the C0 test problem (Chezy without intervention). Near the open boundaries errors can be observed in Figure 2.1a. We focus on the errors near the outflow boundaries (water level boundaries) because here the residuals are largest. Large residuals indicate large errors, as well as residuals that differ much from the residuals in the rest of the domain.

The tool leads us to the term in which an error is made. We first find a large error in the viscosity term $-\nu_{\mathrm{H}}\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right)$. This term was not taken into acount for the outflow boundary.

In the Lek test problem we still see large residuals at the outflow boundary. We now find errors in the pressure term $g\frac{\partial \zeta}{\partial x}$ and in the advection term $U\frac{\partial U}{\partial x}$. Here the distances between grid points are computed in a wrong way. Furthermore, the bottom friction is computed in the same way as for any other grid points:

$$\frac{\tau_{\mathrm{bottom},x}}{\rho_0 H} \approx \frac{gU\sqrt{U^2+V^2}}{HC^2_{\mathrm{2D},x}}.$$

In WAQUA, however, the V-velocity is neglected:

$$\frac{\tau_{\mathrm{bottom},x}}{\rho_0 H} \approx \frac{gU|U|}{HC^2_{\mathrm{2D},x}}. \tag{2.2.1}$$

We correct the errors for the pressure term and the advection term and implement the bottom friction as in equation 2.2.1. We now find residuals of approximately the same size at the boundaries as for the rest of the domain, see Figure 2.1b for the Lek test problem and 2.1c for the Randwijk test problem.

Now the residuals are computed (almost) without errors, we check the constuction of the Jacobian. Here we find and resolve similar programming errors as we found in the construction of the residual.

2.3 Conclusion

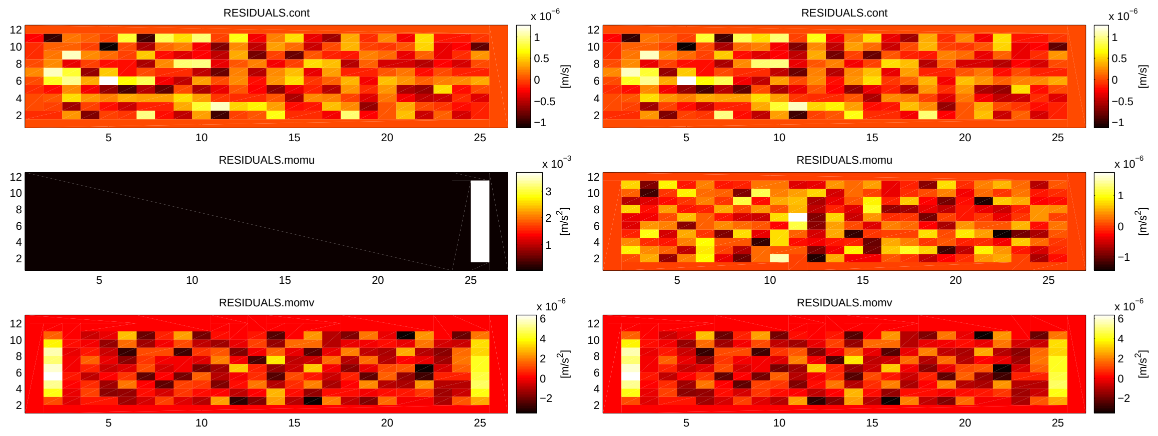
We have removed some redundant parts in the program. Now the matlab code is easier to overview and other improvements in this part of the program can be done with less effort.

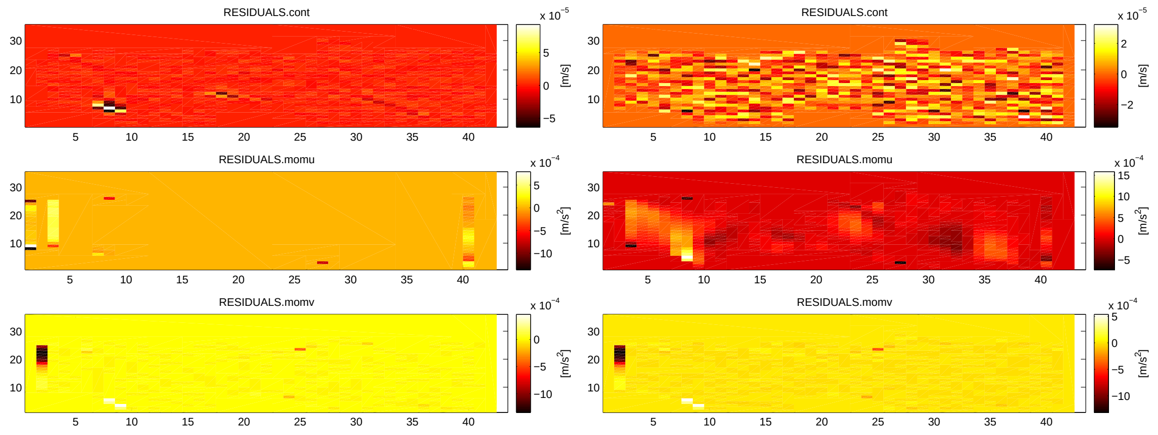
We have made a tool which helps to find errors. With this tool we have found and resolved errors near the outflow boundaries (water level boundaries). The solutions we find after these improvements are more reliable, for all three test problems without interventions.

2.3.1 Recomendations

We have only checked for errors that occur in our test problems. The only boundary conditions that are correctly implemented at this moment are for water level boundaries and discharge boundaries in V-velocity direction. This should be extended for open boundaries in U-velocity directions.

WAQUA has the possibility to use many other boundary conditions, for example velocity, QH- and Riemann invariant boundaries. This is described in [6, Chapter 6]. QuickFlow would


(a) Chezy test problem (flow from left to right).


(b) Lek test problem (flow from right to left).

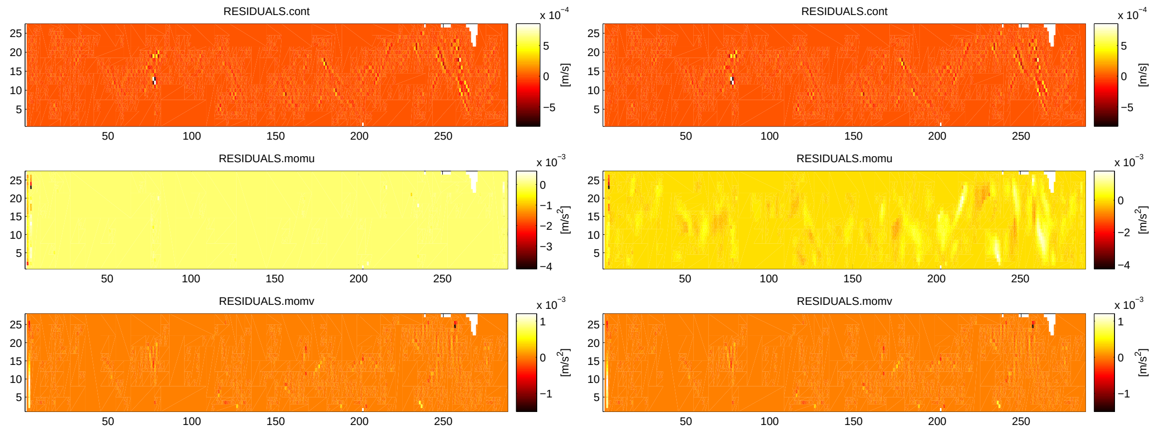

(c) Randwijk test problem (flow from right to left).

Figure 2.1: Residuals of test problems, before improvements (left) and after improvements (right). Note the differences in scales. For each subfigure: top: continuity equation, middle: U momentum equation, bottom: V momentum equation.

be applicable to more situations if these boundary conditions were implemented as well in QuickFlow.

WAQUA has the possibility to recompute the bottom friction coefficient every few iterations. QuickFlow uses the coefficient from the initial solution for the whole computation. The QuickFlow results will become more accurate if the coefficient would be recomputed. However, it is a rather complicated computation, especially when weirs and barriers are involved. Furthermore, the equations will not be quadratic anymore, which makes it more difficult to check the resulting linearized system.

Chapter 3

Continuation

We have noticed that QuickFlow usually can find a stationary solution. However, if the intervention is ’too large’, it can take a long time to find this solution. Furthermore, large differences in water levels occur between neighbouring grid cells, which indicates that the solution might be unreliable. These problems are discussed further in Section 3.1. In Section 3.2 we introduce a solution to these problems: ’continuation’. in Section 3.3 the test results are discussed.

3.1 Analysis of the former results

We use the Chezy gutter as a test problem. We apply the depth intervention CA1, as described in Appendix B. The stationary solution that was found by QuickFlow is shown in Figure 3.1. Note the large differences in velocities and especially in water levels at some neighbouring gridcells, especially around the location of the intervention. Furthermore, ’screens’ were placed at places we do not expect the velocity to be (nearly) zero, for example around $(0.56, 0.2)$. Whenever a velocity at a certain grid point is (nearly) zero, we set the velocity to zero for this point. The point has become ’dry’ and we say that a screen has been placed.

We plot the size of the pseudo time step and the residual in every iteration, see Figure 3.1. We see that the pseudo time step and the residual do not change much until approximately the 140-th iteration. Just before this decrease of the residual screens were placed, because of which the problem became easier to solve. However, this resulted in a ’wrong’ stationary solution.

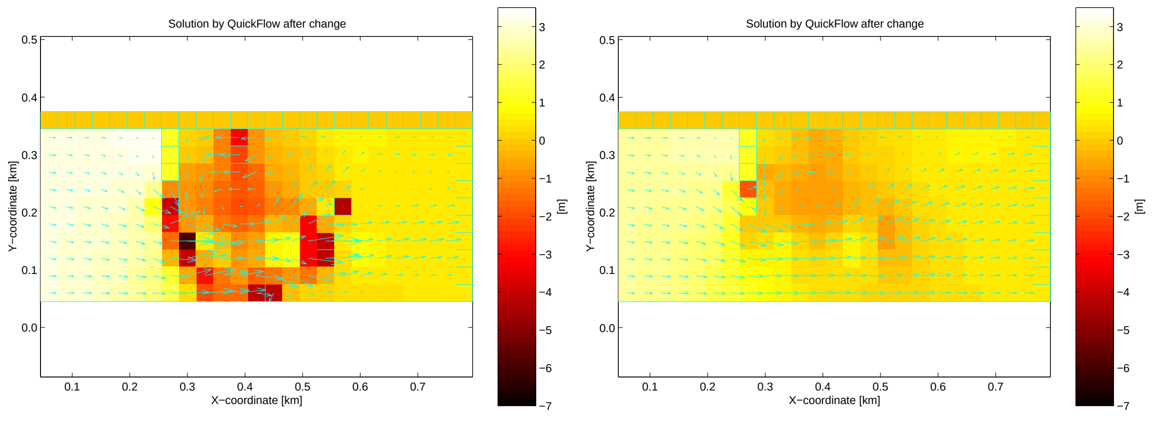
We have also tried ’smaller’ interventions: CA1 with holes and dams of only half the depth and height respectively, or only one hole. For these problems QuickFlow finds a reliable looking stationary solution. We conclude that QuickFlow has difficulties in finding the stationary solution for ’large’ interventions such as in CA1.

3.2 Implementation

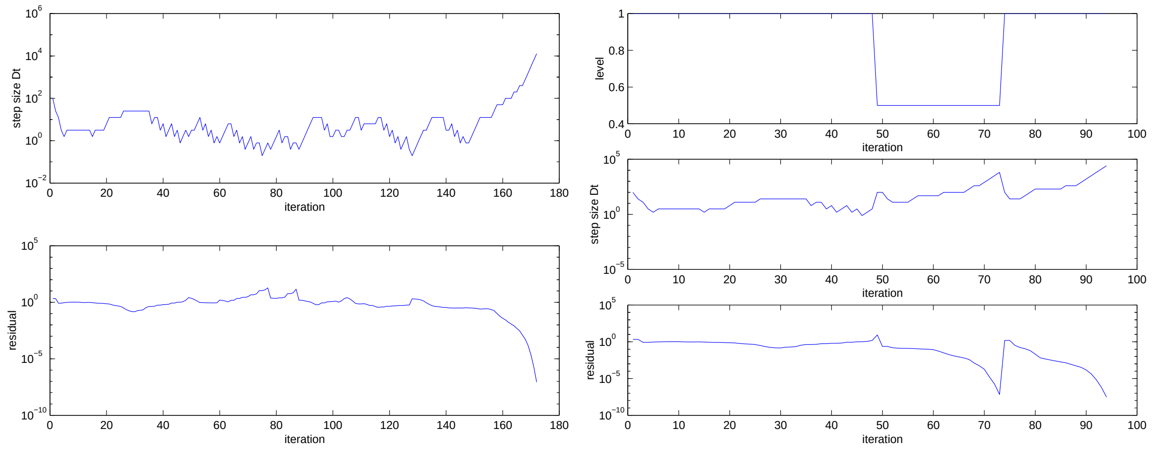
In general continuation means that if a problem seems to be to difficult to solve, a ’less difficult’ problem is solved before. Then the solution of the last problem is used as initial solution for the previous problem.

In our application this means that we first solve a problem with smaller interventions and use the solution of this problem as initial solution for the problem we actually want to solve.

For the implementation of continuation in QuickFlow we need to determine ’how much’ we want to increase or decrease the intervention, this will be discussed in Section 3.2.1. In



(a) Stationary solution



(b) Continuation level, analysis of the size of the pseudo time step and residuals.

Figure 3.1: Results for Chezy test problem with dams and holes (CA1). Without continuation (left) and with continuation (right).

Section 3.2.2 we will discuss under which conditions the intervention is increased or decreased.

3.2.1 Level of adaptation

We introduce a 'level of adaptation' which denotes how 'large' the intervention is. This level is set to 1 for the total intervention (corresponds to the problem we are trying to solve) and it is set to 0 for no intervention at all (initial condition). We can now change the depth following the level of adaptation.

Example 3.2.1 *Suppose we want to solve a problem with a dam of 4 meter. level=1 corresponds with this dam of 4 meter. Imagine that QuickFlow is not able to find the stationary solution.*
We set level=0.5, which corresponds to a dam of 2 meter. Now imagine that QuickFlow can find the stationary solution. We use this solution as the new initial solution.
We set level=1 again (dam of 4 meter) and try to solve the problem with the new initial solution. If this does not give us the stationary solution, we will set level=0.75 (dam of 3 meter), etc.

3.2.2 Conditions for level

We need conditions for which we increase and decrease the level of adaptation. We use the following conditions:

- if many pseudo time steps have been taken at this level and many Newton iterations were necessary in the last pseudo time step, then decrease the level.
- if residual = $\|\mathbf{F}(\mathbf{x})\|_2 < 10^{-7}$ then increase the level;

The second condition is analogous to the condition for stopping the iteration process and using the solution as the stationary solution, before we implemented continuation.

The first condition prevents that QuickFlow keeps focussed on a local optimum while the results do not become any better.

In Figure 3.2 an overview of the implementation of continuation is shown.

3.3 Test results

In Figure 3.1 the solution for the Chezy 1 test problem is shown. We now see gradually changing water levels and flow velocities. We furthermore see that the level of adaptation decreases once. Before this decrease we see that the time step decreases while the residual (slightly) increases. After the decrease of the level of adaptation the reverse can be observed.

Our second test problem is Chezy with an other intervention: CA2, as described in Appendix B. We have solved this problem both without and with continuation. The stationary solutions are shown in Figure 3.3. Note that the solution with continuation looks more reliable than the one without.

3.3.1 Stability

When we try to check our results with WAQUA we notice that WAQUA can not find a stationary solution. Therefore, we check the stability of the QuickFlow results. We use the methods described in Chapter 4. We find that the stationary solution for the CA1 test problem is unstable. We add the functionality that QuickFlow will give a warning if the stationary solution is unstable.

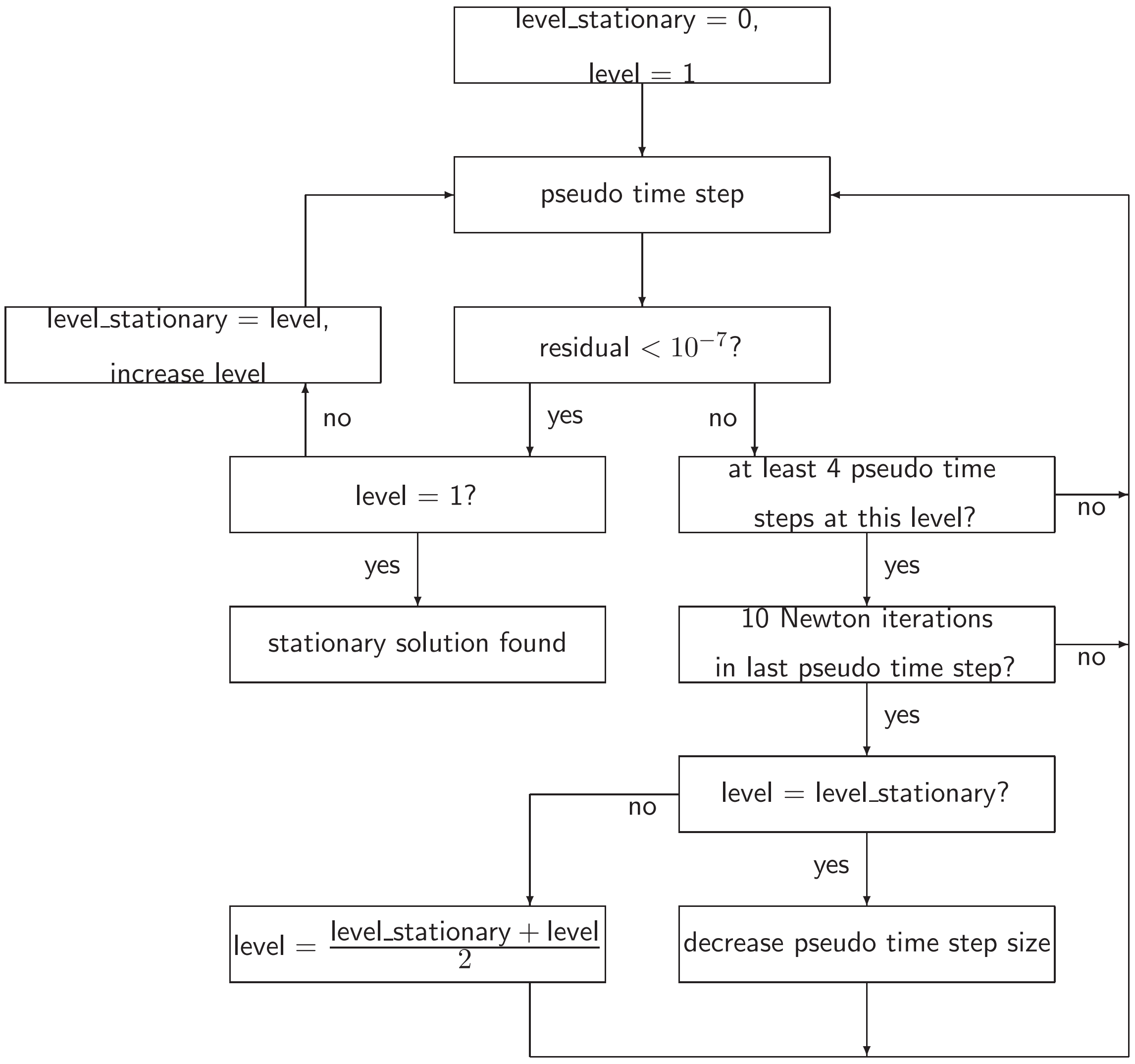

level_stationary = 0,
level = 1
pseudo time step
residual < 10^{-7}?
level_stationary = level,
increase level
no
yes
no
level = 1?
at least 4 pseudo time steps at this level?
no
yes
yes
stationary solution found
10 Newton iterations in last pseudo time step?
no
yes
level = level_stationary?
no
yes
level = $\frac{\text{level\_stationary} + \text{level}}{2}$
decrease pseudo time step size

Figure 3.2: Overview of continuation.

3.3.2 Remark

The results described above were achieved before the changes to reduce the computing time as described in Chapter 5. When we try to reproduce the results with the new configuration, we find other results for the Chezy test problem (see Figure 3.4). We do need continuation to find a stationary solution for the CA1 test problem. However, if we change the configuration a little (recompute the Jacobian every Newton iteration), we do not need continuation to find a stationary solution. We have checked the stability for both solutions, the one with continuation has one pair of eigenvalues with negative real part, the other has three. If we use the method with Crank Nicolson we find that the first one is stable, the second one is not.

With the new configuration discussed in Chapter 5 we do not need continuation to find a stable solution for the other, realistic test problems: Lek and Randwijk with and without interventions.

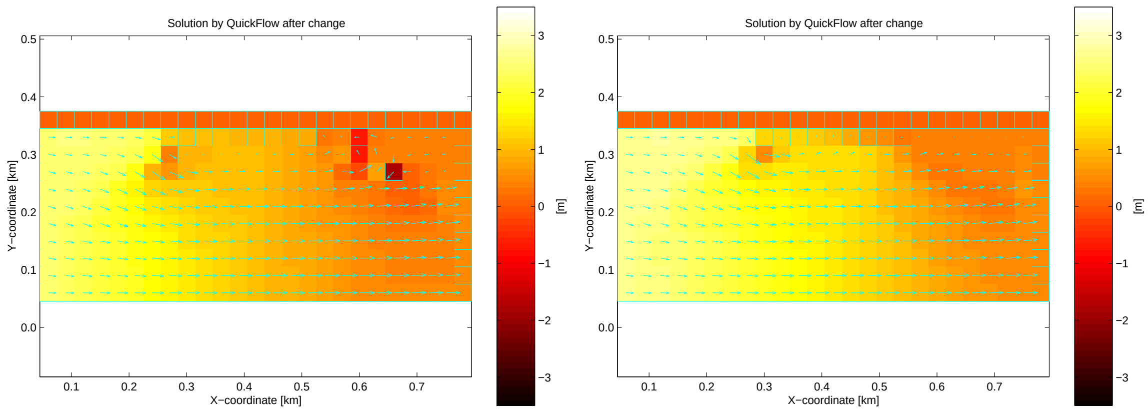
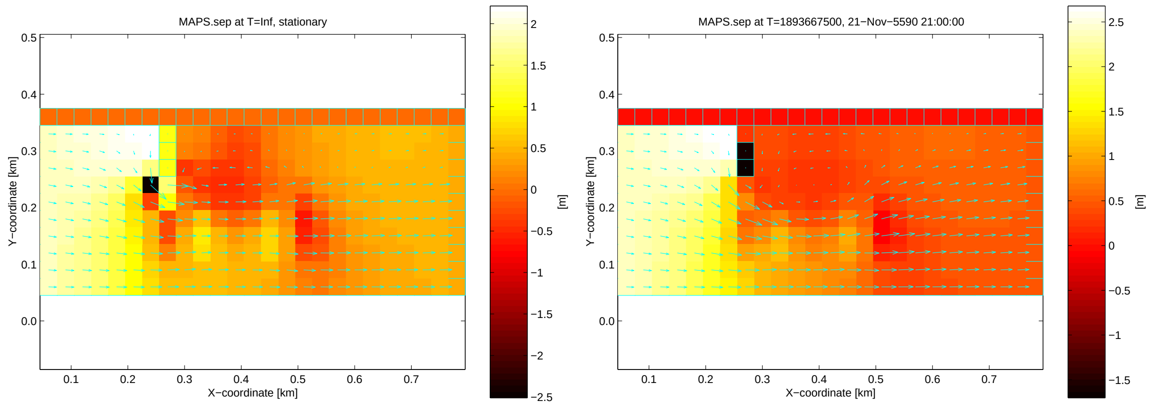


Figure 3.3: Stationary solutions for Chezy test problem with rise of the bottom (CA2). Without continuation (left) and with continuation (right).

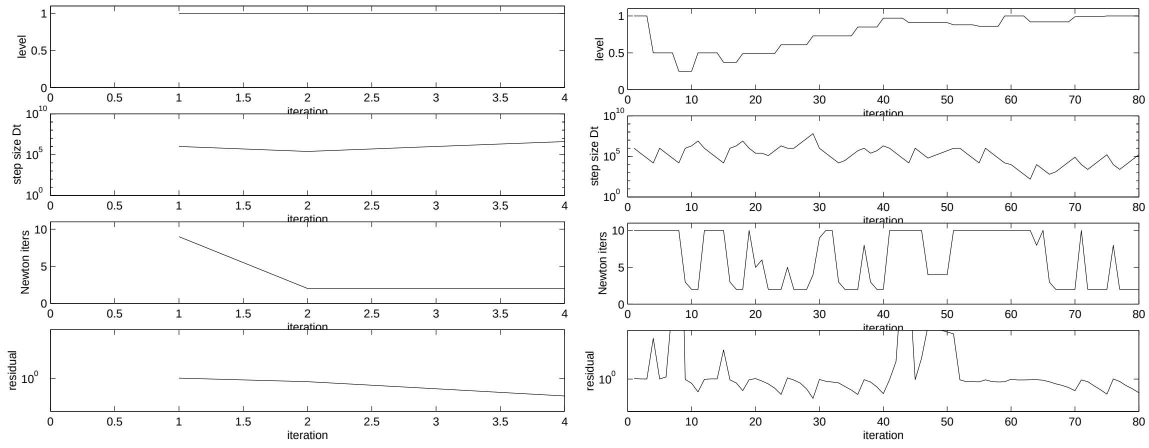
3.4 Conclusion

Now that we have implemented continuation in QuickFlow, we can find a more reliable stationary solution of problems with large interventions. We can even solve unstable problems. QuickFlow will give a warning in this case.

However, after the changes we have made to reduce the computing times, we find that it is rarely necessary to use continuation. The method can still be useful when there is an error in any other part of the computation, for example in the boundary conditions. Continuation can help to find a solution, which can be analysed and errors can be detected.



(a) Stationary solution



(b) Continuation level, analysis of the size of the pseudo time step and residuals.

Figure 3.4: Results for Chezy test problem with dams and holes (CA1). Without continuation and with recomputing the Jacobian (left) and with continuation (right).

Chapter 4

Stability analysis

In some test problems, it is not possible to find a stationary solution with WAQUA. Therefore we can not verify the results from QuickFlow with the results from WAQUA.

This occurs if we try to find the new solution of a problem with a 'large' intervention. In these cases the stationary solution is too far from the initial condition. This gives rise to instabilities.

We have studied and implemented two methods to investigate the stability of a certain solution. In Section 4.1 we discuss the relation between stability and eigenvalues. In Section 4.2 we describe how we implemented this theory in QuickFlow. In Section 4.3 we discuss a method that uses Crank-Nicolson to analyze the stability. The relation and difference between stability and stationarity is discussed in Section 4.4.

4.1 Stability theory and eigenvalue analysis

Suppose we have the linear problem

$$\begin{aligned} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} &= \mathrm{A}\mathbf{x} - \mathbf{b}, \qquad (4.1.1)\\ &\text{with A a matrix of size } n \times n,\\ &\text{with } \mathbf{b} \text{ a vector of size } n,\\ &\lambda_i,\ i = 1,..,n \text{ the eigenvalues of A},\\ &\mathbf{r}_i,\ i = 1,..,n \text{ the corresponding eigenvectors of A},\\ &\text{and } \bar{\mathbf{x}} = \mathrm{A}^{-1}\mathbf{b} \text{ is the stationary solution.} \end{aligned}$$

Definition 4.1 *The stationary solution $\bar{\mathbf{x}}$ is* stable *if for every $\varepsilon > 0$ a $\delta > 0$ exists such that if $\|\mathbf{x}(0) - \bar{\mathbf{x}}\| < \delta$ then $\|\mathbf{x}(t) - \bar{\mathbf{x}}\| < \varepsilon$ for all $t \geq 0$.*

In this definition an arbitrary norm $\|\cdot\|$ is used. [4, Chapter 4]

Theorem 4.1.1 *If the eigenvectors $\mathbf{r}_i$ form a basis for $\mathbb{R}^n$, then the stationary solution $\bar{\mathbf{x}}$ is stable if $Re(\lambda_i) < 0$ for all $i = 1, 2, .., n$.*

Proof The general solution of the linear system (4.1.1) is:

$$\begin{aligned} \mathbf{x}(t) &= \left(\mathbf{x}(0) - \mathrm{A}^{-1}\mathbf{b}\right) e^{\mathrm{A}t} + \mathrm{A}^{-1}\mathbf{b},\\ &\quad \text{with } \mathbf{x}(0) \text{ the initial solution at } t = 0. \end{aligned}$$

Choose any $\varepsilon > 0$, take $\delta = \varepsilon$. Take the perturbed initial solution $\mathbf{x}(0) = \bar{\mathbf{x}} + \delta\mathbf{x}$, with the perturbation $\delta\mathbf{x}$ such that

$$\begin{aligned}\|\mathbf{x}(0) - \bar{\mathbf{x}}\| &= \|\bar{\mathbf{x}} + \delta\mathbf{x} - \bar{\mathbf{x}}\| \\ &= \|\delta\mathbf{x}\| < \delta.\end{aligned}$$

If we now have that $\mathrm{Re}(\lambda_i) < 0$ for all $i = 1, 2, .., n$, then $e^{\mathrm{A}t} < 1$ for $t > 0$. This can easily be verified if we write $\mathrm{A} = \mathrm{SJS}^{-1}$ with J the Jordan form of A:

$$\begin{aligned}e^{\mathrm{A}t} &= e^{\mathrm{SJS}^{-1}t} = \mathrm{S}e^{\mathrm{J}t}\mathrm{S}^{-1} \\ &< \mathrm{S\,I\,S}^{-1} = \mathrm{I} \text{ for all } t > 0.\end{aligned}$$

J_{ij} are the subblocks of J. The inequality above follows from

$$e^{\mathrm{J}_{ij}t} = e^{\lambda_i t}\begin{pmatrix} 1 & t & \frac{t^2}{2!} & \cdots & \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & \frac{t^2}{2!} \\ & \emptyset & & \ddots & t \\ & & & & 1 \end{pmatrix} \to 0 \text{ for } t \to \infty.$$

We now see that the solution will stay 'close' to the stationary solution:

$$\begin{aligned}\|\mathbf{x}(t) - \bar{\mathbf{x}}\| &= \left\|\left(\mathbf{x}(0) - \mathrm{A}^{-1}\mathbf{b}\right)e^{\mathrm{A}t} + \mathrm{A}^{-1}\mathbf{b} - \bar{\mathbf{x}}\right\| \\ &< \|\mathbf{x}(0) - \bar{\mathbf{x}}\| = \|\delta\mathbf{x}\| \\ &< \delta = \varepsilon.\end{aligned}$$

4.2 Eigenvalue analysis in QuickFlow

We have made a tool that checks whether the eigenvalues of the linearized problem

$$\begin{aligned}\mathrm{M}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} - \mathbf{F}'(\mathbf{x}^n)\,\mathbf{x}^{n+1} &= \mathbf{F}(\mathbf{x}^n) - \mathbf{F}'(\mathbf{x}^n)\,\mathbf{x}^n, \\ &\quad \text{with } \mathbf{F}' \text{ the Jacobian of } \mathbf{F},\end{aligned} \tag{4.2.1}$$

are indeed in the left half plane.

The linearized problem (4.2.1) can be rewritten as

$$\begin{aligned}\mathrm{M}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} &= \mathrm{C}\mathbf{x}^{n+1} - \mathbf{d} \\ &\quad \text{with } \mathrm{C} = \mathbf{F}'(\mathbf{x}^n), \text{ and } \mathbf{d} = \mathbf{F}(\mathbf{x}^n) - \mathbf{F}'(\mathbf{x}^n)\,\mathbf{x}^n.\end{aligned}$$

Since M is an identity matrix with some extra zeros on the main diagonal (corresponding to boundary points) we can reorder $\mathbf{x} = \begin{pmatrix}\mathbf{x}_u \\ \mathbf{x}_l\end{pmatrix}$ such that $\mathrm{M}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \begin{pmatrix}\mathrm{I} & 0 \\ 0 & 0\end{pmatrix}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$.

After reordering $\mathrm{C} = \begin{pmatrix}\mathrm{C}_{uu} & \mathrm{C}_{ul} \\ \mathrm{C}_{lu} & \mathrm{C}_{ll}\end{pmatrix}$ and $\mathbf{d} = \begin{pmatrix}\mathbf{d}_u \\ \mathbf{d}_l\end{pmatrix}$ similarly we can write:

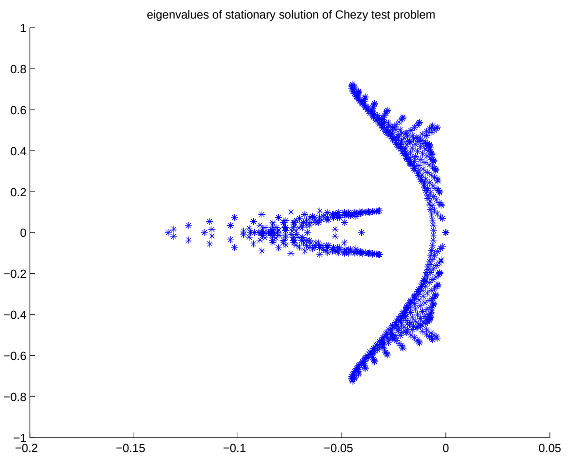


Figure 4.1: Eigenvalues Chézy test problem.

$$\left.\begin{array}{rcccccc} \frac{\mathrm{d}\mathbf{x}_u}{\mathrm{d}t} & = & \mathrm{C}_{uu}\mathbf{x}_u & + & \mathrm{C}_{ul}\mathbf{x}_l & - & \mathbf{d}_u \\ 0 & = & \mathrm{C}_{lu}\mathbf{x}_u & + & \mathrm{C}_{ll}\mathbf{x}_l & - & \mathbf{d}_l \end{array}\right\} \Rightarrow \tag{4.2.2}$$

$$\begin{aligned} \frac{\mathrm{d}\mathbf{x}_u}{\mathrm{d}t} &= \mathrm{C}_{uu}\mathbf{x}_u + \mathrm{C}_{ul}\left(\mathrm{C}_{ll}^{-1}(\mathbf{d}_l - \mathrm{C}_{lu}\mathbf{x}_u)\right) - \mathbf{d}_u \\ &= \underbrace{\left(\mathrm{C}_{uu} + \mathrm{C}_{ul}\mathrm{C}_{ll}^{-1}\mathrm{C}_{lu}\right)}_{=\mathrm{A}}\mathbf{x}_u + \underbrace{\mathrm{C}_{ul}\mathrm{C}_{ll}^{-1}\mathbf{d}_l - \mathbf{d}_u}_{=\mathbf{b}} \qquad (4.2.3) \\ &= \mathrm{A}\mathbf{x}_u - \mathbf{b}_u. \end{aligned}$$

In the previous section, we have seen that the linear system $\frac{\mathrm{d}\mathbf{x}_u}{\mathrm{d}t} = \tilde{\mathrm{C}}\mathbf{x}_u - \tilde{\mathbf{b}}_u$ is stable if the eigenvalues of $\tilde{\mathrm{C}}$ have negative real part. Note that the stability is not influenced by the fact that we do not directly take into account any disturbance on $\mathbf{x}_l$. If we add a perturbation $\delta\mathbf{x}_l$ and substitute in (4.2.2) $\mathbf{x}_l$ by $\tilde{\mathbf{x}}_l = \mathbf{x}_l + \delta\mathbf{x}_l$, we will still find equation (4.2.3).

We use the matlab function `eigs` to find the eigenvalues and hereby check the stability of the stationary solution of the linearized problem. For initial solution of the Chezy test problem without intervention all eigenvalues are plotted in Figure 4.1. Note that a lot of eigenvalues are close to the imaginary axis. This makes it difficult to find the eigenvalues with largest real part with an iterative method, which is used by matlab. In the next section, we will discuss a method that can handle this problem.

4.3 Stability and Crank-Nicolson

Above we discussed that it can be difficult to determine whether all eigenvalues have negative real part. An other method to check the stability is to use an 'unstable numerical scheme', in contrast to the unconditionally stable Euler backward method used in QuickFlow. Crank Nicolson is a numerical method that is unstable whenever the solution has eigenvalues with positive real part, when the solution is stable (all eigenvalues have negative real part), than the Crank Nicolson method is unconditionally stable. Using Crank Nicolson we find the new solution $\mathbf{x}^{n+1}$ by solving

$$\frac{1}{\Delta t}\mathrm{M}(\mathbf{x}^{n+1} - \mathbf{x}^n) + \frac{1}{2}\left(\mathbf{F}(\mathbf{x}^{n+1}) + \mathbf{F}(\mathbf{x}^n)\right) = \frac{1}{2}(\mathbf{b}^{n+1} + \mathbf{b}^n).$$

Let $\mathbf{x}^n$ be an unstable (but stationary) solution. Then for a small perturbation $\delta\mathbf{x}$, the solution $\mathbf{x}^n + \delta\mathbf{x}$ is not stationary. Note that this can only be observed after a sufficient number of sufficiently small time steps. In that case the difference between the original solution and the new solution will grow very large.

We have introduced a function in QuickFlow that uses Crank-Nicolson to analyse the stability. The output of this function consists of plots of the solution after every 10 time steps and of the difference between the original and the new solution. It is up to the user to determine whether the new solution diverges 'too much' from the original solution.

4.4 Note on stability and stationarity

Above we have discussed when a linear system is said to be stable, remember the definition of stability (Definition 4.1).

Consider the system

$$\begin{aligned}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} &= \mathbf{F}(\mathbf{x}_0, t), \\ &\quad \text{with } \mathbf{x}_0 = \mathbf{x}(0) \text{ the initial solution.}\end{aligned}$$

The solution $\bar{\mathbf{x}}$ is called *stationary* if $\mathbf{F}(\bar{\mathbf{x}}, t) = 0$. This solution is also called the *critical* or *equilibrium* point. Note that if a solution is stationary, this does not mean it is stable. In words the concepts can be explained as follows:

If a solution is stationary, the solution will not change at all if time increases and no perturbation occurs. If a solution $\mathbf{x}$ is stable, the solution will stay 'in the neighbourhood' of $\mathbf{x}$ if a small perturbation occurs.

In our application, we sometimes find a stationary solution that is not stable. This can be compared to a pencil balancing on a fingertip. One might be able to balance it such that the pencil stays upright (stationary situation). But a small disturbance (eg a small movement of the hand) can cause the pencil to fall (instability).

Similarly we can find stationary flow velocities and water levels, that are unstable: a small disturbance (eg a ship passing by, a small change in the boundary conditions, or even a wind blow) can cause the system to 'turn' and the flow velocities and water levels can change dramatically. Thus, such stationary solutions are mathematical artefacts and will not occur in physical reality.

4.5 Conclusion

It is important whether a stationary solution is stable: not every stationary solution is stable and (thus) physically realistic. We have implemented two methods to detect instable solutions.

The first method is based on eigenvalue analysis. This method is especially well suited for small problems (Chezy or Lek test problem). In other cases it might take up to 30 minutes to find a result. The second method is based on Crank-Nicolson time integration. This method can be applied whenever unstabilities are suspected to occur. Especially, when it is 'very' unstable this will give a quick indication that a solution in unstable.

4.5.1 Recommendations

It is difficult to formulate 'hard' criteria on the stability of a solution. Therefore, we propose the following approach whenever instabilities are suspected to occur:

- If the problem is small (Lek or smaller) use eigenvalue analysis. If all eigenvalues have negative real parts, the solution is stable.
- If the problem is large or if one or more eigenvalues with (small) positive real part were found, use Crank-Nicolson. Choose an appriopriate time step and number of time steps (for now 2000 steps of 1 second seem to suffice). If the difference between the original solution and the new solution stays small, the solution is stable.

WAQUA is sometimes used to find a stationary solution. It is not well suited for this application, but with the use of restarts, it usually can find a stationary solution. Sometimes, however, the solution might look stationary while in fact it is periodical and unstable. The user does not always notice this instability. QuickFlow can give an indication that a certain solution is unstable. It would be an improvement of WAQUA if it gives such an indication as well. The methods described in this chapter can be used for this purpose.

Chapter 5

Reduction of computing times

One of the goals of this project is to reduce the computing time needed to find the stationary solution. The computation generally consists of two parts: the setting up of the equations and the solving of the equations. The first part was mostly copied from WAQUA. In this chapter several methods are described to solve the equations more efficiently.

In Section 5.1 we focus on the outer loop: time integration with Euler backward. In Section 5.2 we discuss some ways to improve the linarization with quasi Newton methods. Finally, in Section 5.3 we describe how we reduced the computing time to solve the linear system.

Every configuration described below was tested for the LV1 and RA1 test problems. The characteristics of the test problems are summarized in Table 5.1. They are discussed in more detail in Appendix B. The LV1 test problem is a rather difficult version of the Lek test problem, with an intervention of the fairway. The RA1 test problem has a hole in the bottom at a realistic location. Whenever the the results did not give enough information we also tried the test problems without depth intervention and/or with an increase of the depth at the fairway of the Randwijk problem. Usually we chose the most difficult one: RV4.

Table 5.1: Test problems.

Test problem	Intervention
Lek	
L0	none
LV1	decrease of 1 meter of fairway
Randwijk	
R0	none
RA1	one deep hole, next to fairway
RV1	decrease of 1 meter of fairway
RV2	decrease of 2 meter of fairway
RV4	decrease of 4 meter of fairway

In this chapter many possible configurations are described, together with their test results. For every test little changes were made from the departure configuration with:

- large initial pseudo time step (10^6);
- fast changes in initial pseudo time step;
- determination Jacobian only in the first Newton iteration of the pseudo time step, and when the line search failed;

- maximum of 4 line search iterations;
- LU decomposition with static pivotting, using 'symmetric minimum degree' permutation;
- reversed permutation after solving the linear system;
- determination L and U only when a new Jacobian was determined; and
- using BiCGSTAB as iterative method with accuracy of 10^{-3}.

These configurations are discussed in more detail below.

5.1 Time integration: Euler Backward

As described in the Introduction (Chapter 1) we try to find the stationary solution of the discretized shallow water equations

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}),$$

see also Figures 1.2 and 1.3. We use the Euler backward method for time integration. This results in

$$\frac{1}{\Delta t}\mathrm{M}\left(\mathbf{x}^{n+1} - \mathbf{x}^{n}\right) = \mathbf{F}(\mathbf{x}^{n+1}), \tag{5.1.1}$$

with M the mass matrix (roughly spoken an identity matrix with some extra zeros on the main diagonal). For $n \to \infty$ we find the stationary solution with $\mathbf{x}^{n+1} = \mathbf{x}^{n}$.

We have to choose a suitable pseudo time step size Δt. If it is too small, it may take many pseudo time steps before the stationary solution is reached. On the other hand, if it is taken too large, it might be difficult to solve the linearized system of equations. Furthermore, if the present solution is far away from the stationary solution, the pseudo time step should be taken smaller than when the stationary solution is almost reached.

We have tested some values for the initial pseudo time steps and conditions to change the size of the pseudo time step. If the residual of the pseudo time step increases then the size of the pseudo time step is decreased. Furthermore, the number of Newton iterations is taken into account: if many iterations were needed, the pseudo time step becomes smaller, few Newton iterations results in a larger pseudo time step. Schematically the conditions for the pseudo time step are shown in Figure 5.1.

We have tested the following values for k:

	k_1	k_2	k_3	k_4
Slow		2	5	7
Fast	2	4	6	8

Note that for the second set, the pseudo time step will grow faster. The test results are shown in Table 5.2. Furthermore, some characteristics are shown in Figure 5.2.

The larger the pseudo time step, the less pseudo time steps are needed to find the stationary solution which in fact is the solution for $t \to \infty$. On the other hand, a small pseudo time step results in a matrix A which is more diagonally dominant and it is easier to solve the lineair system. Furthermore, a very large pseudo time step is not beneficial and it might become less robust.

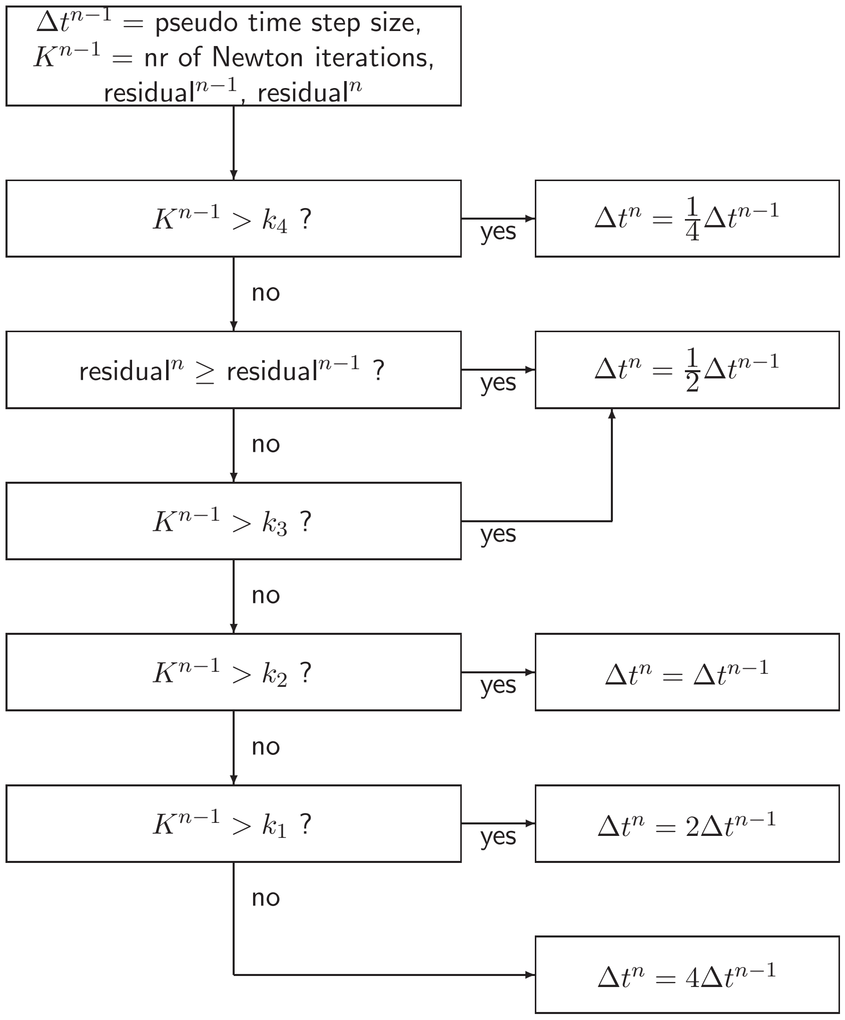

Δt^{n-1} = pseudo time step size,
K^{n-1} = nr of Newton iterations,
residual^{n-1}, residual^{n}
$K^{n-1} > k_4$?
yes
$\Delta t^n = \frac{1}{4}\Delta t^{n-1}$
no
residual^{n} $\geq$ residual^{n-1} ?
yes
$\Delta t^n = \frac{1}{2}\Delta t^{n-1}$
no
$K^{n-1} > k_3$?
yes
no
$K^{n-1} > k_2$?
yes
$\Delta t^n = \Delta t^{n-1}$
no
$K^{n-1} > k_1$?
yes
$\Delta t^n = 2\Delta t^{n-1}$
no
$\Delta t^n = 4\Delta t^{n-1}$

Figure 5.1: Flow chart for choosing the pseudo time step size.

Table 5.2: Test results for the size of the pseudo time step.

$\Delta t_{\text{initial}}$	time step change	CPU time	pseudo time steps	Newton iterations
		test problem LV1		
10^2	Slow	21.20	15	37
10^2	Fast[2]	40.26	17	42
10^4	Slow	12.56	6	13
10^4	Fast	13.56	5	19
10^6	Slow	12.22	4	17
10^6	Fast	**11.70**	4	17
10^8	Slow	**11.82**	4	17
10^8	Fast	**11.83**	4	17
		test problem RA1		
10^2	Slow[1]	.		
10^2	Fast[2]	447.95	31	106
10^4	Slow	229.24	20	64
10^4	Fast	138.23	9	37
10^6	Slow	**35.99**	4	10
10^6	Fast	**36.07**	4	10
10^8	Slow	**35.76**	4	10
10^8	Fast	**35.71**	4	10

[1] No solution was found within 50 pseudo time steps (485.61 CPU seconds).
[2] Continuation (see Chapter 3) was applied.

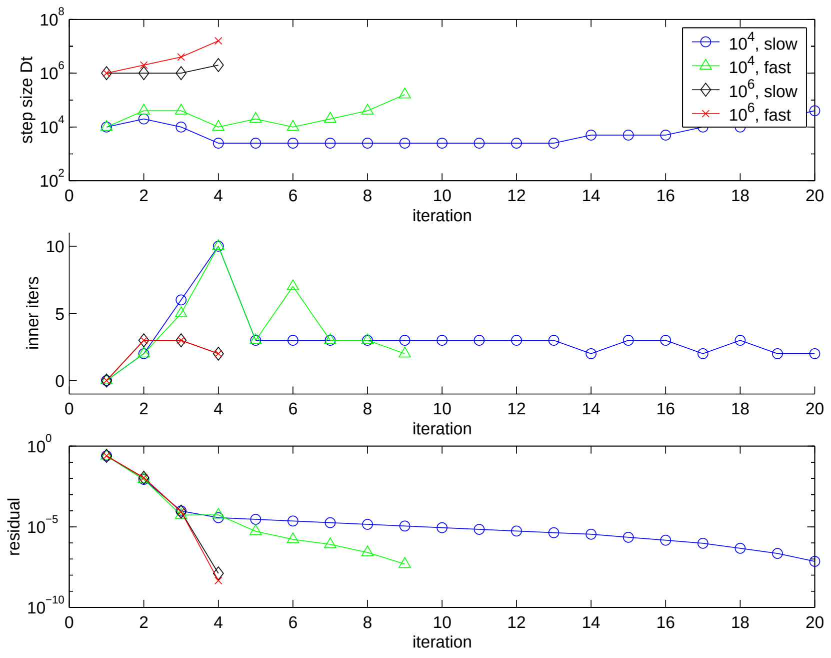


Figure 5.2: Time step size, number of Newton iterations and residuals at every iteration for test problem RA1.

5.2 Linearization: quasi Newton's method

Remember that for the n-th pseudo time step we want to solve equation (5.1.1). This can also be written as:

$$\frac{1}{\Delta t}\mathrm{M}\,\mathbf{s}^n - \mathbf{F}(\mathbf{x}^n + \mathbf{s}^n) = 0, \qquad (5.2.1)$$
$$\text{with } \mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{s}^n.$$

We will apply a quasi Newton's method to find $\mathbf{s}^n$, see also Figure 1.4 and 1.6. If we use ordinary Newton's method we will solve the linearized system:

$$\left(\frac{1}{\Delta t}\mathrm{M} - \mathbf{F}'(\mathbf{x}^k)\right)\mathbf{s}^{k+1} = -\mathbf{F}'(\mathbf{x}^k)\,\mathbf{s}^k + \mathbf{F}(\mathbf{x}^k), \qquad (5.2.2)$$

with $\mathbf{s}^0 = \mathbf{0}$ and $\mathbf{s}^{k+1} \rightarrow \mathbf{s}^n$ for increasing k.

We will use the stopcriterium $\left\|\mathbf{F}(\mathbf{x}^k\right\|_2 < 0.01 \left\|\mathbf{F}(\mathbf{x}^0\right\|_2$: the residual for the new pseudo time should be at least 100 times less than for the old pseudo time.

Note that we will not solve the linear system for the new solution $\mathbf{x}^{n+1}$ but for the change of the solution $\mathbf{s}^n = \mathbf{x}^{n+1} - \mathbf{x}^n$. This results in a linear system which is easier and more accurately to solve with an iterative method.

5.2.1 Choice of Jacobian and Broyden's method

Normally in Newton's method the Jacobian is recomputed every Newton iteration. Especially for large problems this can be very costly: about 40% of the total computing time. Therefore, it can be advantageous to keep the Jacobian during the pseudo time step and possibly update it using Broyden's method. A full Jacobian update is necessary whenever the line search failed (see Section 5.2.2).

Broyden's method

In Broyden's method the Jacobian, is computed in the ordinary way at the beginning of a pseudo time step and whenever the line search failed. In the other Newton iterations the Jacobian is updated:

$$\mathbf{F}'(\mathbf{s}^k) \;\approx\; \mathbf{F}'(\mathbf{s}^{k-1}) + \mathbf{F}(\mathbf{s}^k)\frac{(\mathbf{s}^k - \mathbf{s}^{k-1})^T}{\|\mathbf{s}^k - \mathbf{s}^{k-1}\|_2^2}$$

It can still be very costly to compute the update $\mathbf{F}(\mathbf{s}^k)\,(\mathbf{s}^k - \mathbf{s}^{k-1})^T / \left\|\mathbf{s}^k - \mathbf{s}^{k-1}\right\|_2^2$, which is a full matrix. Therefore we use a modified version of Broyden's method, which does not compute the update itself, but only the vectors $\mathbf{F}(\mathbf{s}^k)$ and $(\mathbf{s}^k - \mathbf{s}^{k-1})^T / \left\|\mathbf{s}^k - \mathbf{s}^{k-1}\right\|_2^2$, for $k = 2, 3, ...$ When solving the linearized system (5.2.2) we only need

$$\mathbf{F}'(\mathbf{s}^k)\,\mathbf{s}^k \;\approx\; \mathbf{F}'(\mathbf{s}^{k-1})\,\mathbf{s}^k + \mathbf{F}(\mathbf{s}^k)\frac{(\mathbf{s}^k - \mathbf{s}^{k-1})^T}{\|\mathbf{s}^k - \mathbf{s}^{k-1}\|_2^2}\,\mathbf{s}^k,$$

which is much easier to compute than the update itself.

Table 5.3: Test results for Jacobian update and LU decomposition.

update within pseudo time step				
Jacobian	L and U	CPU time	pseudo time steps	Newton iterations
		test problem LV1		
yes	yes	9.74	4	11
yes	no	**6.20**	3	8
Broyden	no	13.20	7	16
no	no	11.70	4	17
		test problem RA1		
yes	yes	49.00	3	8
yes	no	**36.57**	3	8
Broyden	no	57.91	6	12
no	no	**36.07**	4	10
		test problem RV4		
yes	yes	190.92	6	28
yes	no [1]	388.91	10	30
Broyden	no	113.40	11	22
no	no	**68.34**	4	16

[1] Continuation (see Chapter 3) was applied.

We have tested the following three possibilities:

- recompute Jacobian every Newton iteration;
- modified Broyden's method as described above;
- recompute Jacobian only in the first Newton iteration of the pseudo time step, and when the line search failed.

The test results are related to the test results for the LU preconditioning of the iterative method, as described in Section 5.3.1. The results are shown in Table 5.3.

If we use Broyden, only few Newton iterations per pseudo time step are needed. And BiCGSTAB needs only little time: 1-2 % of the total computing time, instead of 2-8 % with complete Jacobian update every Newton iteration or with only little Jacobian updates. On the other hand, many pseudo time steps are needed and LU decomposition can be more expensive.

If we choose only to compute a new Jacobian at the first Newton iteration or when the line search failed, it takes less computing time to update the Jacobian (completely or with Broyden), which compensates for the fact that sometimes more pseudo timesteps and Newton iterations are needed.

5.2.2 Line search

Newton's method (equation (5.2.2)) usually only has local convergence. This can be a disadvantage especially when the initial solution is far from the stationary solution. This drawback can be overcome by 'line search'. Instead of taking a full Newton step $\mathbf{s}^k$ a smaller step $\alpha\mathbf{s}^k$ with $\alpha \in (0, 1]$ is taken.

The line search coefficient α can be chosen in many ways. We have tried a relatively easy criterium: we start with $\alpha = 1$, we take $\alpha = \alpha/2$ until a smaller residual than for $\alpha = 0$ is found. We do this at most $i_{\max}$ times. Note that if we set $i_{\max} = 1$ in fact no line search is applied. If the results for every $\alpha \in (0, 1]$ are worse than with $\alpha = 0$ (solution previous Newton iteration), then this line search failed. The test results are summarized in Table 5.4.

Table 5.4: Test results line search. Note: if $i_{\max} = 1$ no line search is applied.

$i_{\max}$	CPU time	pseudo time steps	Newton iterations	avg nr line search iterations per Newton iteration
	test problem L0			
7	4.24	3	7	1.00
	test problem LV1			
7	11.64	4	17	2.29
4	11.70	4	17	2.29
1	**5.95**	4	10	1.00
	test problem R0			
7	24.44	3	6	1.00
	test problem RA1			
7	37.15	4	10	1.00
	test problem RV1			
7	37.27	3	10	1.70
4	43.31	3	13	1.62
1	**30.20**	3	8	1.00
	test problem RV2			
7	88.96	5	20	2.90
4	71.03	4	20	1.95
1	**31.00**	3	8	1.00
	test problem RV4			
7	90.03	5	20	3.05
4	68.34	4	16	2.62
1	**38.45**	4	10	1.00

Note that whenever only one line search iteration is used when $i_{\max}$ is set at a high level (for example $i_{\max} = 7$), it is of no use to set this maximum at a lower level. The number of line search iterations increases when the depth intervention is increased. The time needed for the line search becomes 3 to 5 times larger, when the number of line searches increases. Note that the results for the total computing time are in contradiction with the theory dicussed above. We will do some recommendations on this in Section 5.4.1.

5.2.3 Elimination of boundary conditions

The linearization (5.2.2) of the shallow water equations results in the linear system

$$\begin{aligned} \mathrm{A}\,\mathbf{s} &= \mathbf{b}, \qquad (5.2.3)\\ \text{with } \mathrm{A} &= \frac{1}{\Delta t}\mathrm{M} - \mathbf{F}'(\mathbf{x}^k),\\ \mathbf{b} &= -\mathbf{F}'(\mathbf{x}^k)\,\mathbf{s}^k + \mathbf{F}(\mathbf{x}^k). \end{aligned}$$

Some of the rows in (5.2.3) are related to open boundaries, at which the velocities or water levels are known. Therefore we can eliminate these rows from the linear system. This results in a smaller matrix A, and possibly in a lower condition number for this matrix.

Table 5.5: Test results elimination of boundary conditions.

elimination of boundary conditions	CPU time	pseudo time steps	Newton iterations
	test problem L0		
no	**4.16**	3	7
yes	4.27	3	7
	test problem LV1		
no	**11.70**	4	17
yes	11.95	4	17
	test problem R0		
no	25.18	3	6
yes	**24.62**	3	6
	test problem RA1		
no	36.07	4	10
yes	**35.72**	4	10
	test problem RV4		
no	**68.34**	4	16
yes[1]	1228.30	15	45

[1] Continuation (see Chapter 3) was applied, without continuation there is no (or possibly very slow) convergence.

We have done some tests with and without elimination of these boundary conditions. The results are shown in Table 5.5. Furthermore the matrix structure is shown in Figure 5.3.

The eliminiation of boundary conditions results in 2 to 6 % less nonzeros in A. The condition number becomes 100 times smaller for the Lek test problem without intervention, but only 2 % for the LV1 test problem. It was not possible to determine the condition number for the Randwijk test problem.

The computing time needed for eliminiation of boundary conditions itself is not compensated by a reduction in computing time for solving the linear system.

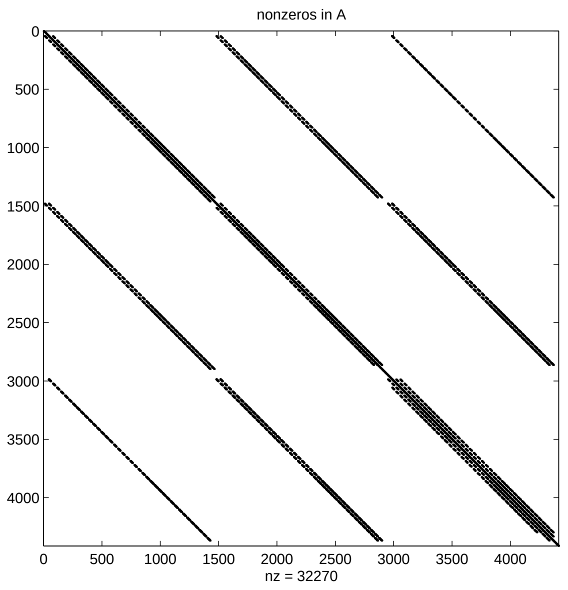
5.3 Solving the linear system: LU preconditioned BiCGSTAB

Remember that we are solving the linear system (5.2.3). An overview of how we solve this system is shown in Figure 1.5.

5.3.1 Preconditioning

An iterative solution method for the linear system (5.2.3) can be more efficient if a preconditioner is applied. We have tested both complete and incomplete LU decomposition as a preconditioner. For some test problems, it was not possible to find a solution using incomplete LU decomposition.

In Table 5.3 the test results are shown. The LU decomposition can be very costly. Therefore, it is advantageous to use the same L and U during the pseudo time step. Furthermore, if the Jacobian was not updated in the present Newton iteration, L and U will remain the same, so recomputation would be redundant.



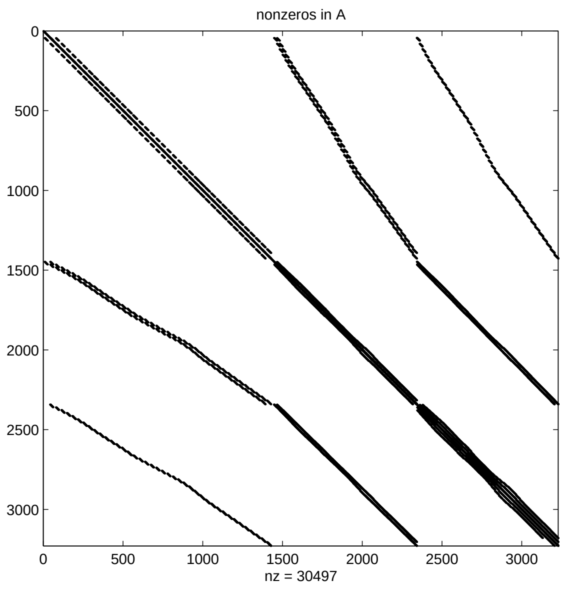


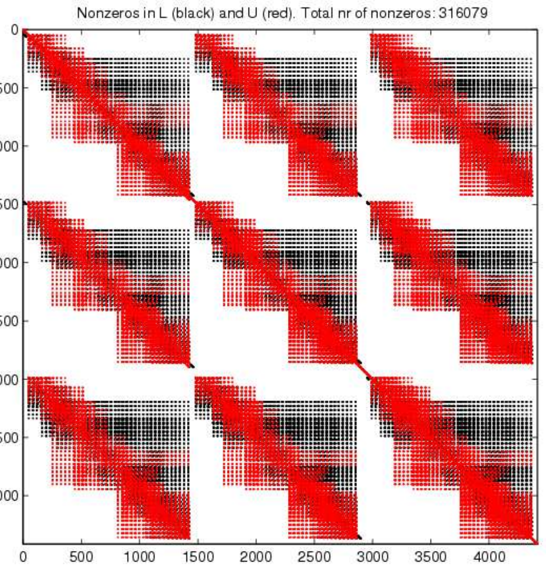
Figure 5.3: Example of nonzeros in matrix A for Lek test problem without intervention without elimination of boundary conditions (left) and with elimination of boundary conditions (right).

Static pivotting

Usually matlab uses a lot of interchanging of rows in the LU decomposition, due to partial pivotting. This operations can take much CPU time. Therefore we suppress pivoting. Instead we do one permutation before applying LU decomposition. We have tried several standard matlab permutations. This results in a permutation vector $\mathbf{p}$ and a permuted version of A: $\tilde{\mathrm{A}} = \mathrm{A}(\mathbf{p}, \mathbf{p})$, L: $\tilde{\mathrm{L}} = \mathrm{L}(\mathbf{p}, \mathbf{p})$ and U: $\tilde{\mathrm{U}} = \mathrm{U}(\mathbf{p}, \mathbf{p})$. We can solve the linear system either by using L, U and $\mathbf{b}$ (option 1) or by using $\tilde{\mathrm{L}}$, $\tilde{\mathrm{U}}$ and $\tilde{\mathbf{b}} = \mathbf{b}(\mathbf{p})$ (option 2). The first option immediately results in $\mathbf{s}$. The second option results in $\tilde{\mathbf{s}}$, which needs reversed permutation to find $\mathbf{s}$. These 2 options are compared in Table 5.6. The structure of the resulting matrices L and U is shown in Figure 5.4.

Table 5.6: Test results for reversed permutation.

option reversed permutation	CPU time	pseudo time steps	Newton iterations
	test problem	**L0**	
1	4.97	3	7
2	**4.16**	3	7
	test problem	**LV1**	
1	12.48	4	17
2	**7.95**	5	13
	test problem	**R0**	
1	28.16	3	6
2	**25.18**	3	6
	test problem	**RA1**	
1	41.83	4	10
2	**36.07**	4	10



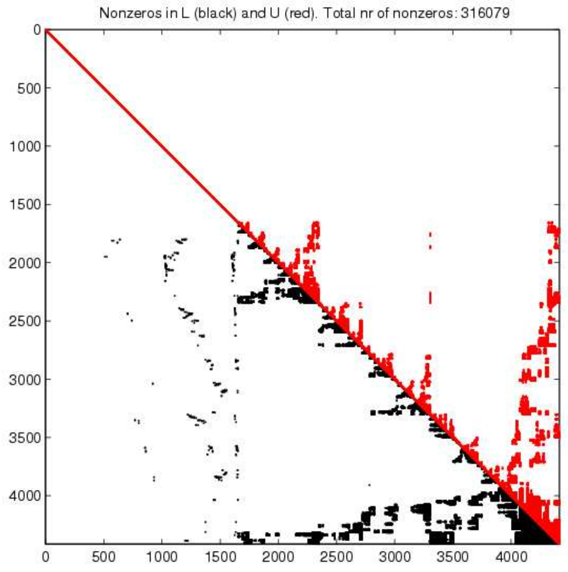


Figure 5.4: Nonzeros in matrices L (black) and U (red) for the Lek test problem without intervention. Option 1 for reversed permutation (before BiCGSTAB) (left) and with option 2 for reversed permutation (after BiCGSTAB) (right).

The linear system with the matrices L and U resulting from option 2 for reversed permutation is much easier to solve (about 3 times faster). This is because L and U now are truly lower and upper triangular matrices respectively.

5.3.2 Iterative Methods

Originally the linear system (5.2.3) was solved directly, using the standard matlab command `s=A\b`. We have tried several iterative methods: BiConjugate Gradient Stabilized (BiCGSTAB) Generalized Minimal Residual (GMRES) and Quasi-Minimal Residual (QMR). We use the implementation of these methods as described in [1]. All these methods are Krylov subspace methods for general matrices. They are especially applicable when the system is unsymmetric and/or indefinite.

One can choose how accurate the linear system should be solved: $\|\mathbf{As}-\mathbf{b}\|_2 < \varepsilon$, for some accuracy ε. Furthermore we have to choose when to restart GMRES. Some test have shown that a restart after 5 GMRES iterations gives good results, this is used in the rest of the tests. The test results for direct and iterative methods to solve the linear system are shown in Table 5.7.

Note that if with accuracy $\varepsilon = 10^{-6}$ only half a BiCGSTAB iteration or one GMRES or QMR iteration is needed, it will make no difference if less accuracy ($\varepsilon = 10^{-3}$) is demanded. The main differences can be seen in the RV4 test problem: more iterations in BiCGSTAB/GMRES/QMR are needed, which results in longer computing times for solving the linear system, especially when a high accuracy ($\varepsilon = 10^{-6}$) is demanded. There are no differences in number of pseudo time steps or Newton iterations. This is because the linear system that is being solved, remains the same, independent of the iterative method or the accuracy.

Table 5.7: Test results for iterative methods and their accuracy.

iterative method (or direct)	accuracy ε	CPU time	avg nr iterations iterative method
	test problem LV1		
direct		**11.65**	
BiCGSTAB	10^{-6}	**11.63**	0.50
GMRES	10^{-6}	11.82	1.00
QMR	10^{-6}	12.13	1.00
	test problem RA1		
direct		**35.42**	
BiCGSTAB	10^{-6}	**35.51**	0.50
GMRES	10^{-6}	36.85	1.00
QMR	10^{-6}	39.11	1.00
	test problem RV4		
direct		89.64	
BiCGSTAB	10^{-3}	**68.34**	2.38
BiCGSTAB	10^{-6} [1]		
GMRES	10^{-3}	80.77	3.19
GMRES	10^{-6}	92.26	5.38
QMR	10^{-3}	92.11	6.94
QMR	10^{-6}	95.04	7.81

[1] No solution found.

5.4 Conclusion

At the start of this project it was not possible to solve 'difficult' test problems. After some adjustments described in Chapters 2 and 3 to overcome this problem we achieved the results shown in Table 5.8. For the relatively easy test problems the computing time was reduced with a factor around 1.2 by these adjustments.

Especially for large problems (Randwijk, with or without intervention) the improvements are substantial: a reduction of the total computing time by up to 40 times. We have also decreased the number of pseudo time step 3 to 19 times.

Most of the reduction of computing time is due to the choice of the size of the pseudo time step. The implementation of the iterative method result in a slight reduction of the computing time. The improvements of the quasi Newton method sometimes have a positive, sometimes a negative effect. This might change when there is more clarity on the prefered maximum of line search iterations (see Section 5.4.1).

We will keep the configuration as described at the beginning of this chapter.

5.4.1 Recommendations

Local variations in pseudo time step

We have also tried to apply local variations in the pseudo time step. We wanted to use a smaller pseudo time step in 'difficult' regions and a larger pseudo time step in 'easy' regions. Difficulties are supposed to occur in shallow regions. These regions can be characterized by small CFL-numbers $\nu = \frac{u\,\Delta t}{\Delta x}$. We have tried several ways to adapt the local pseudo time step

Table 5.8: Comparison of test results for the 'old' and the present configuration.

CPU time		..× reduction of CPU time due to...				pseudo time steps	
before improve-ments	present	pseudo time step size	quasi Newton	iterative method	total	before improve-ments	present
test problem C0							
3.11	0.99	2.5	1.1	1.1	3.1	6	2
test problem L0							
20.89	4.16	2.9	1.1	1.5	5.0	10	3
test problem LV1							
37.25	11.70	3.0	0.7	1.5	3.2	15	4
test problem R0							
475.52	25.18	8.7	1.1	2.0	18.9	21	3
test problem RA1							
1434.55	36.07	19.7	0.9	2.3	39.8	53	4
test problem RV4 [1]							
2091.99	68.34	5.8	.	.	30.6	77	4

[1] Continuation was used in some cases, for some parts no 'fair' comparison possible.

to the CFL-number. We did not succeed in getting any improvements of the computing time. We still think that this should be possible if the idea is implemented in the right way.

Line search and screens

The results for the number of line search iterations as described in Section 5.2.2 are in contradiction with the theory discussed in this Section: without line search less Newton iterations are needed. Furthermore, the solutions differ up to 0.15 m for the water level and 0.3 m/s for the flow velocities when different line search criteria are used. These large differences are only seen for the RV4 test problem, for other test problems the differences can be neglected (smaller than the accuracy with which we solve the problem). See for example Figure 5.5.

We also see a difference in the screens (grid cell boundaries at which the U or V velocities are set to zero). This might be the reason for the differences in the waterlevels and velocities. We note that screens that are placed once, can never be taken away again.

We have furthermore found that if the time step is taken small ($\Delta t_{\text{start}} = 10^4$) the differences in stationary solutions become negectable (10^{-6} m). We suggest that this problem is studied further and that the possibility to remove screens will be added.

Stop criterium with 'forcing terms' for Newton iteration

It might be advantageous to use 'forcing terms', as described in [2] in the stopping criterium for the Newton iteration. In the methods described in this article, close to the stationary solution the stopping criterium will be more strict. In [2] several methods are described to choose a suitable stopping criterium.

These choices for the stopping criterium result in attractive convergence properties (usually quadratic instead of linear convergence) whenever the initial solution is close enough to the stationary solution [2]. We suggest that this idea is further developed and tested for implementation in QuickFlow.

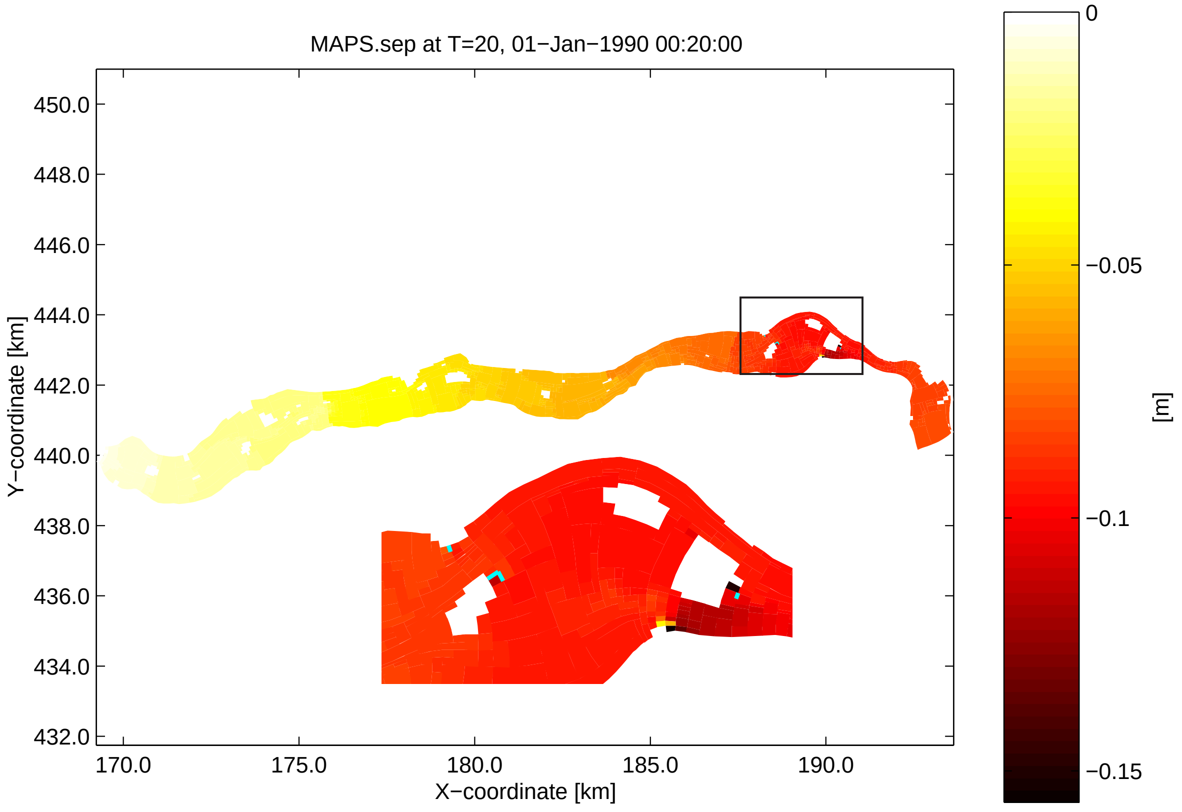


Figure 5.5: Differences between stationary solutions without line search and with a maximum of 7 line search iterations. The little light blue lines (most clear in magnification) denote screens that are present in only one of the solutions.

Stop criterium for BiCGSTAB

For BiCGSTAB we use a stop criterium based on the absolute error: $\|\mathrm{A}\mathbf{s} - \mathbf{b}\|_2 < \varepsilon$, for some ε. It might be better to use a stop criterium that is based on the relative error: $\frac{\|\mathrm{A}\mathbf{s}-\mathbf{b}\|_2}{\|\mathrm{A}\mathbf{s}_0-\mathbf{b}\|_2} = \frac{\|\mathrm{A}\mathbf{s}-\mathbf{b}\|_2}{\|\mathbf{b}\|_2} < \varepsilon$. This is because the linear system can change substantially whenever the size of the pseudo time step changes.

Chapter 6

Conclusions

We have made several improvements to QuickFlow. At the start of this project it was only possible to find the stationary solution for easy problems: Chezy and Lek test problems without or with small interventions.

6.1 Correctness of the solution

We started with repairing some errors in the boundary conditions. For this purpose we made a tool that prints the values of the various terms of the residuals, both for WAQUA and QuickFlow. With this tool errors can be found easily.

Next, we improved QuickFlow for problems with large interventions. We applied continuation to obtain more reliable solutions. After repairing sufficient errors as described above, continuation is not needed for realistic test problems. We have implemented two methods to check whether a certain solution is stable. One is based on eigenvalue analysis, the other on a numerical time integration method (Crank-Nicolson) which itself is unstable if the system has eigenvalues with positive real part. These methods can both be used, depending on the size of the problem at hand. It is important to know whether the stationary solution is stable: especially in WAQUA one might think that a solution is stable, while it is not. The methods have proved to be powerful, but sometimes they use much computing time.

The above improvements ensure that for all test problems the stationary solution can be found and is correct, which was the first goal of this project. Furthermore, they reduced the computing time for the test problems that we could solve at the start of this project by about 20 %.

6.2 Reduce computing times

The second goal of this project was to reduce the computing times. In order to achieve this goal, we first made a tool that prints the computing times for the most important parts of the computation.

The computation generally consists of two parts: the setting up of the equations and the solving of the equations. The first part was mostly copied from WAQUA. In this project we focussed on the second part: how to solve the equations efficiently.

We have reduced the total computing time by introducing and improving several numerical methods in QuickFlow. We have chosen a much larger pseudo time step: therefore fewer pseudo time steps are needed. We have changed the line search algorithm in the quasi Newton iteration. More research is needed to determine the correct approach. This is discussed further in Section 6.3.1.

We replaced the direct method for solving the linear sytem with an iterative one. We found that BiCGSTAB with LU preconditioner gives the best results. Furthermore, we found that it saves computing time if the Jacobian and the L and U are recomputed only a few times: at the beginning of a pseudo time step and whenever the previous line search failed.

All of the above adaptations together resulted in a substantial reduction of computing times: a reduction factor of 3 to 40. The computing time for the Lek test problem with a small intervention (LV1) (a relatively small and easy problem) was reduced from 37 to 12 seconds. The computing time for the Randwijk test problem with a larger intervention (RV1) (a relatively large and difficult problem) was reduced from 1435 to 36 seconds. The stationary solution for each of the test problems we have used can now be found within 70 seconds.

6.3 Recommendations and further research

Before QuickFlow can be used for real applications, some things have to be further improved. Furthermore, we have some recommendations to improve and extend QuickFlow and WAQUA, which are not strictly necessary.

6.3.1 Improvements necessary to QuickFlow

Boundary conditions

At this moment we have only implemented velocity and discharge boundary conditions for open boundaries in V velocity direction. These boundary conditions, and possibly others, should be implemented for the U direction as well. The tool for analyzing residuals can be used to check the implementation.

Screens: drying and flooding

In QuickFlow screens are used to set velocities at zero whenever a certain part of the river becomes dry. However, a screen can never be taken away: flooding is not possible. This is related with the observed difficulties of the line search algorithm. We expect that when screens can be removed, the line search algorithm will become more reliable.

User interface

At the start of this project there was a graphical user interface for QuickFlow. It still exists, but it is unknown whether it still works. At this moment, all communication between the user and QuickFlow is through the matlab command line. The graphical user interface should be checked and adapted whenever necessary.

6.3.2 Other improvements and extensions

Bottom friction

QuickFlow uses the bottom friction coefficient as it was computed by WAQUA. Especially with depth interventions this coefficient might not be realistic. The stationary solution will become more reliable if the bottom friction coefficient can be adapted during the computation according to the new solution. Note that this might seem easy to implement, but it can give a lot of difficulties, especially around weirs and barriers, because the resulting system will not be quadratic anymore but ’ordinary’ nonlinear.

Local variations in pseudo time step size

We have tried to implement local variations in the size of the pseudo time step: for ’difficult’ regions the size should be smaller than for ’easy’ regions. We think that it should be possible to reduce the computing times in this way, but we did not yet achieve this reduction.

Stop criteria

The convergence properties can be improved by applying a forcing term for the stop criterium that is used in the Newton iteration. Furthermore, for BiCGSTAB a stop criterium based on the relative error might result in a reduction of computing times.

Other interventions

We have only implemented and tested interventions related to the bottom depth. An other intervention that occurs often is a change in the bottom structure or roughness. For example trees or bushes are placed or removed. These kind of interventions can be implemented in QuickFlow.

6.3.3 Improvements in WAQUA

Incorporation of QuickFlow in WAQUA

QuickFlow can find a new stationary solution much faster than WAQUA, though the solution from WAQUA might be more accurate. QuickFlow might be incorporated into WAQUA, to be used early in the design process. Later the solution should of course always be checked with WAQUA. This can be done in two ways:

- Make an option in WAQUA which calls QuickFlow’s matlab scripts and let QuickFlow do all the computations.
- Incorporate all methods from QuickFlow into WAQUA: this means rewriting most of the program in fortran code.

The first option is probably the easiest to implement. The second option makes all features of WAQUA (for example recomputing the bottom friction coefficient) available for QuickFlow and is preferable.

Stability analysis in WAQUA

A stationary solution from WAQUA might be instable, especially with large interventions. The methods we introduced for QuickFlow to check the stability of a solution would be a useful extension of WAQUA.

Bibliography

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, second edition, 1994.

[2] S.T. Eisenstat and H.F. Walker. Choosing the forcing terms in an inexact newton method. *SIAM Journal of Scientific Computing*, 17(1):16–32, January 1996.

[3] J.J. Leendertse. *Aspects of a computational model for long-period water-wave propagation*. PhD thesis, University of Technology Delft, 1967.

[4] G.J. Olsder and J.W. van der Woude. *Mathematical System Theory*. VSSD, Delft, third edition, 2005.

[5] F van Wageningen-Kessels. Convergence of quickflow, a steady state solver for the shallow water equations. literature research, December 2006. Report of the literature research for this project.

[6] WL|Delft Hydraulics, Delft. *Technical documentation WAQUA*, April 2005. Version 1.2.

Appendix A

Relation WAQUA and QuickFlow

In Figure A.1 the processes in WAQUA and its relation with QuickFlow is shown. WAQUA makes an SDS-file with information for example the grid, boundary conditions and the solution at several time instances.

QuickFlow uses the data from the SDS-file. In Figure A.2 the processes in QuickFlow are shown. QuickFlow takes the solution of the last time step from the SDS-file. It is assumed that this is the stationary solution.

QuickFlow was designed to compute new stationary solutions after interventions to the bottom. The user provides QuickFlow with information on the adjustments by way of a GUI (Graphical User Interface). QuickFlow computes the new stationary solution after these adjustments. The results are postprocessed and made visible for the user in a plot.

In practical situations the end user would try different adjustments and inspect the results. Then he chooses which adjustment(s) give(s) the desired velocities and water levels. He can check this results with WAQUA, which is supposed to give a more accurate result. Moreover, WAQUA is able to compute time dependent solutions, where QuickFlow only gives the stationary solution.

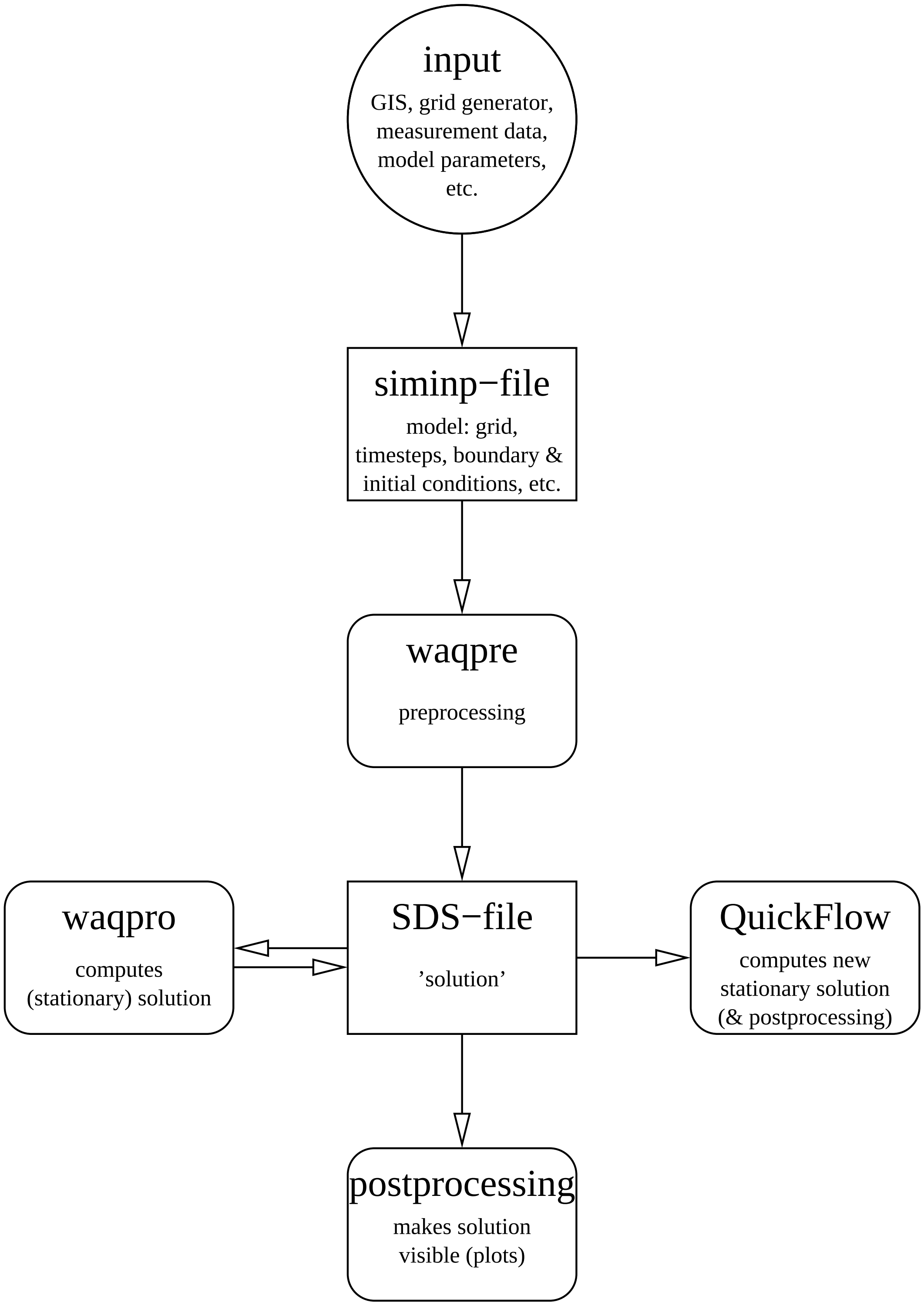

input
GIS, grid generator, measurement data, model parameters, etc.
siminp−file
model: grid, timesteps, boundary & initial conditions, etc.
waqpre
preprocessing
waqpro
computes (stationary) solution
SDS−file
'solution'
QuickFlow
computes new stationary solution (& postprocessing)
postprocessing
makes solution visible (plots)

Figure A.1: Scheme of the relation between WAQUA and QuickFlow.

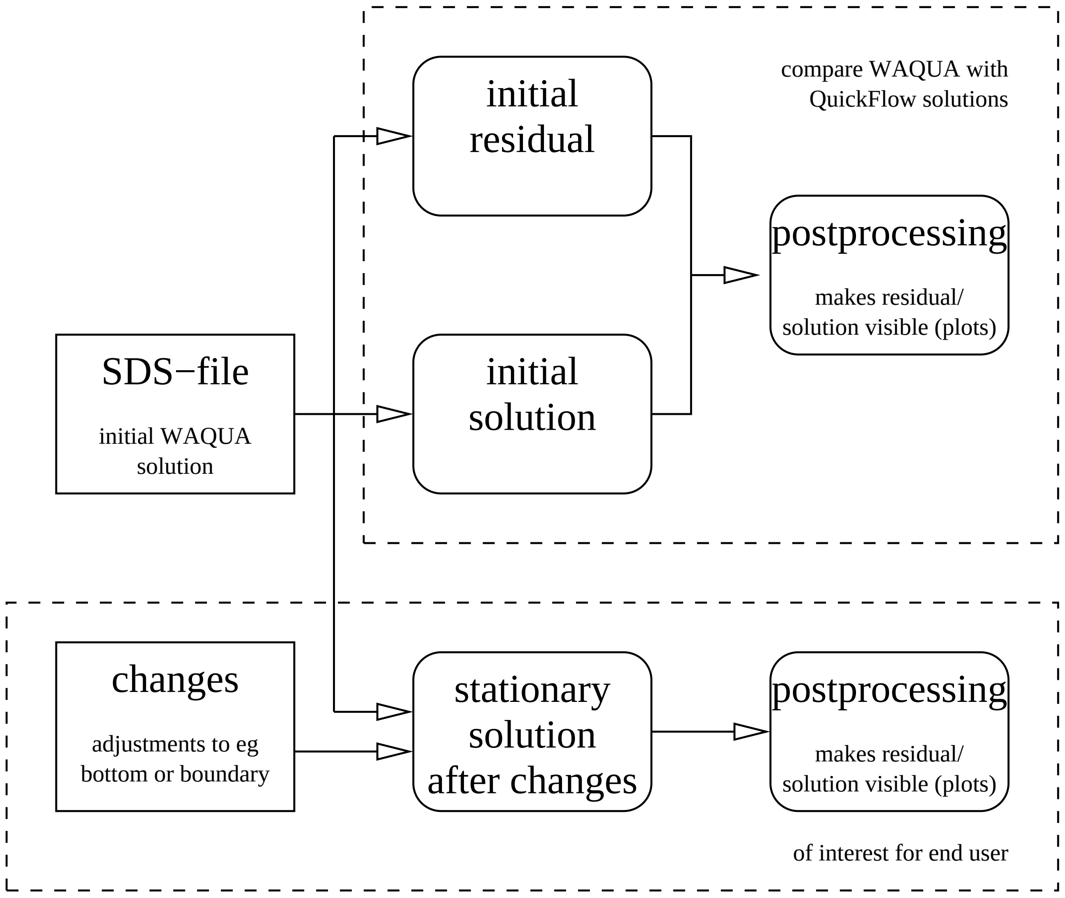

initial residual
compare WAQUA with QuickFlow solutions
postprocessing
makes residual/ solution visible (plots)
SDS−file
initial WAQUA solution
initial solution
changes
adjustments to eg bottom or boundary
stationary solution after changes
postprocessing
makes residual/ solution visible (plots)
of interest for end user

Figure A.2: Scheme of the processes in QuickFlow.

Appendix B

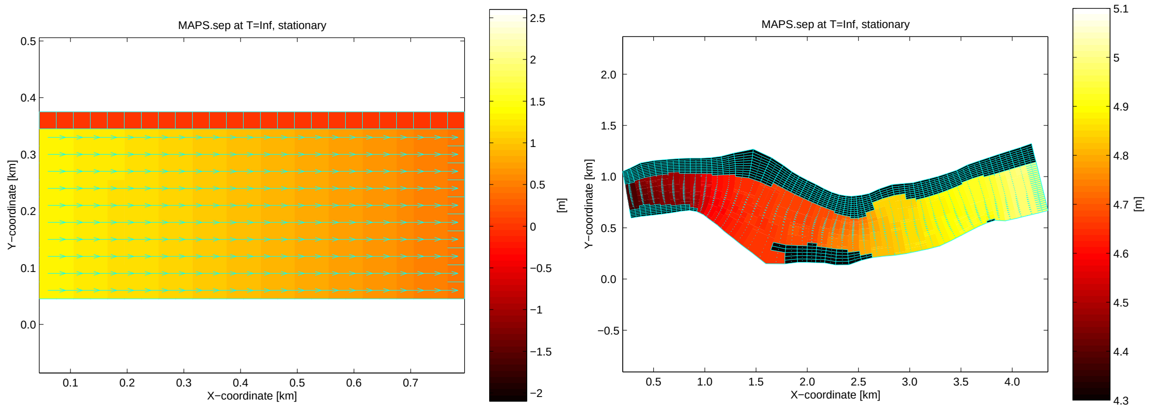
Test problems

We use the test models as described in [5, Chapter 5]. However, we make some changes in the Chézy problem to make it better suited for comparison with the other models. We change the velocity boundary for a water level boundary. Now all test problems have a prescribed discharge at the inflow boundary and a prescribed water level at the outflow boundary.

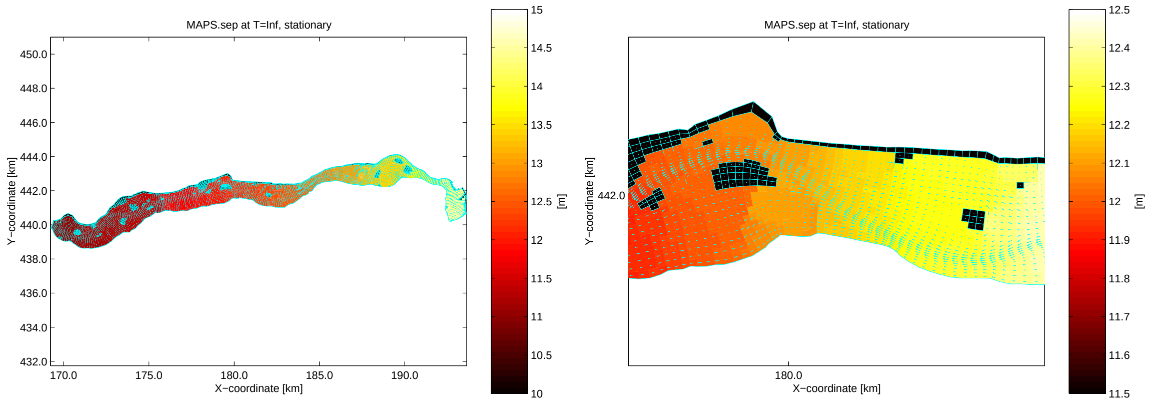
In Figure B.1 the stationary solution of the test models without any intervention is shown. Some characteristics of the test problems are given below.

	Chézy	Lek	Randwijk
length × width (km)	0.75×0.33	4.5×0.5	24×0.5
length × width (columns × rows)	24×10	40×33	285×35
number of grid cells	313	1471	7718
total depth (H), meters	4	4.5	12

We have also made some test problems with interventions to the bottom levels. We have used both rectangular decrease and increase of depths ('dams' or 'holes') and increase of depths at the fairway, see Figures B.2 - B.4.

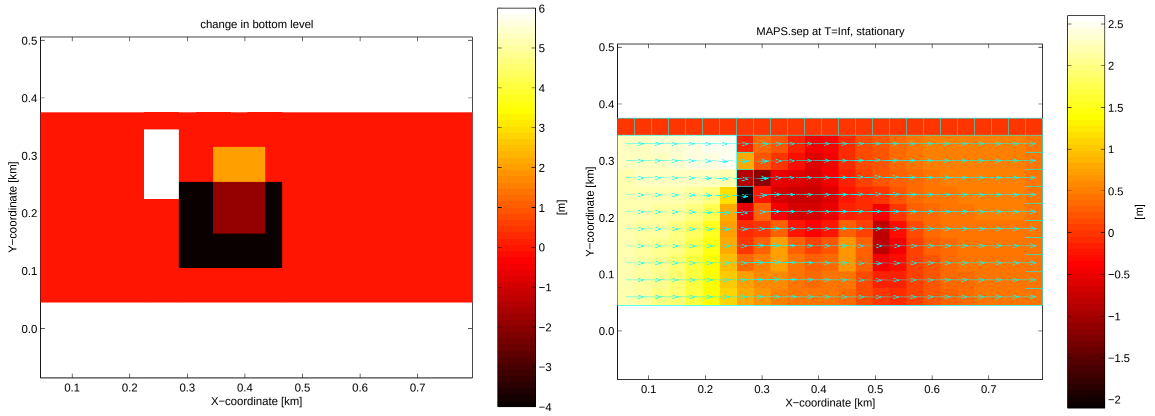


(a) Chézy (C0) (left) and Lek (L0) (right).

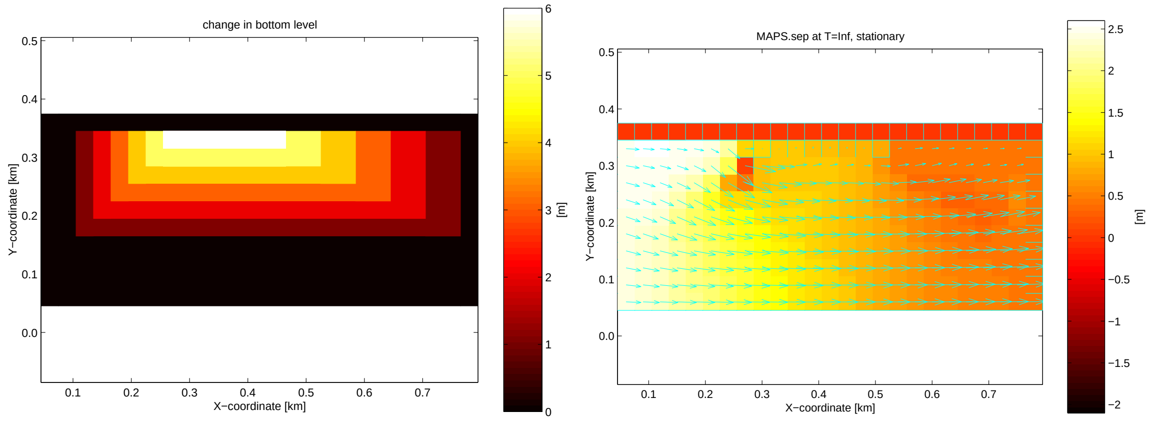


(b) Randwijk (R0) (left) and a detail of this test problem (right).

Figure B.1: Stationary solutions of the test problems without depth adaptations. Water levels are indicated in white- yellow-orange-red, flow velocities are indicated as blue arrows.

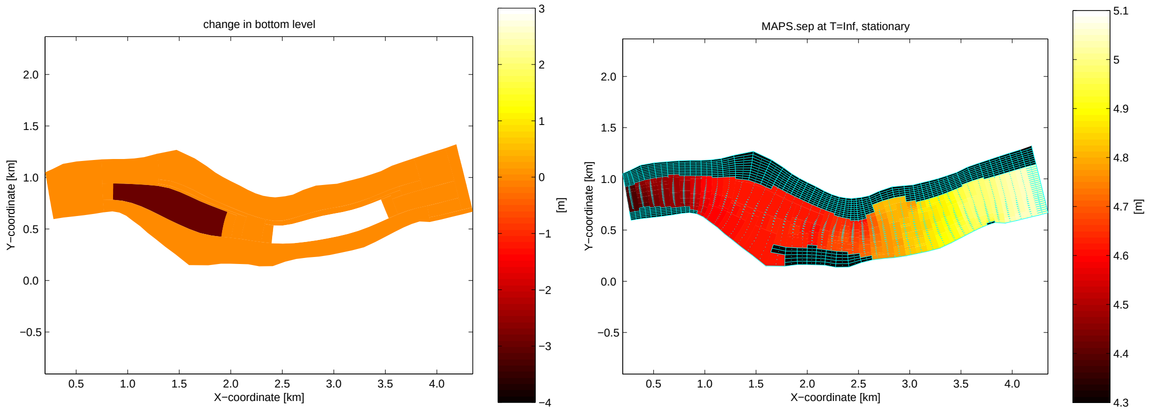


(a) Chézy with some 'dams' and 'holes' (CA1).

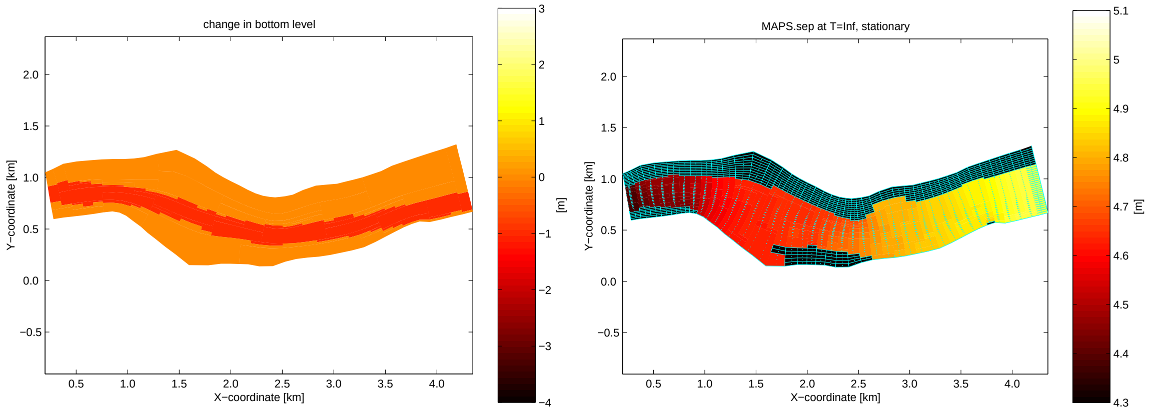


(b) Chézy with an increase of the bottom (CA2).

Figure B.2: Depth adaptation in Chezy test problem (left) and stationary solutions of this test problem with depth interventions (right).

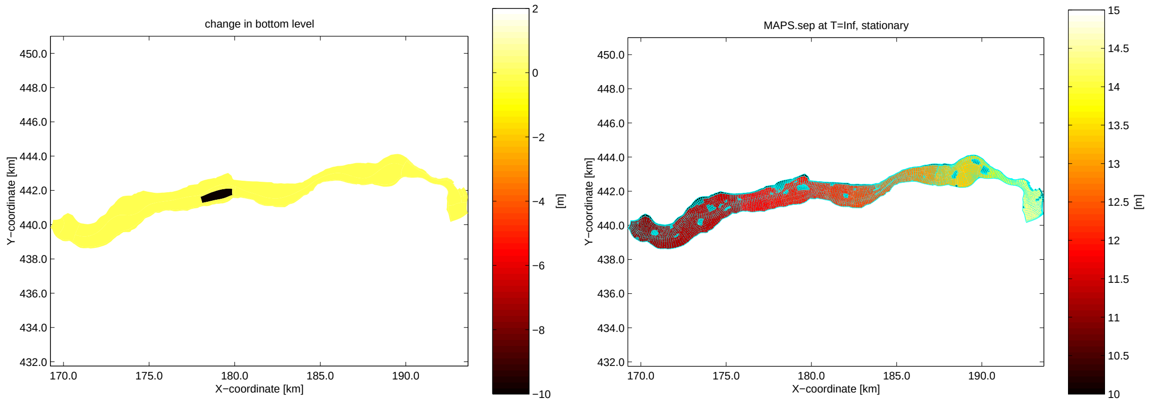


(a) Lek with some 'dams' and 'holes' (LA1).

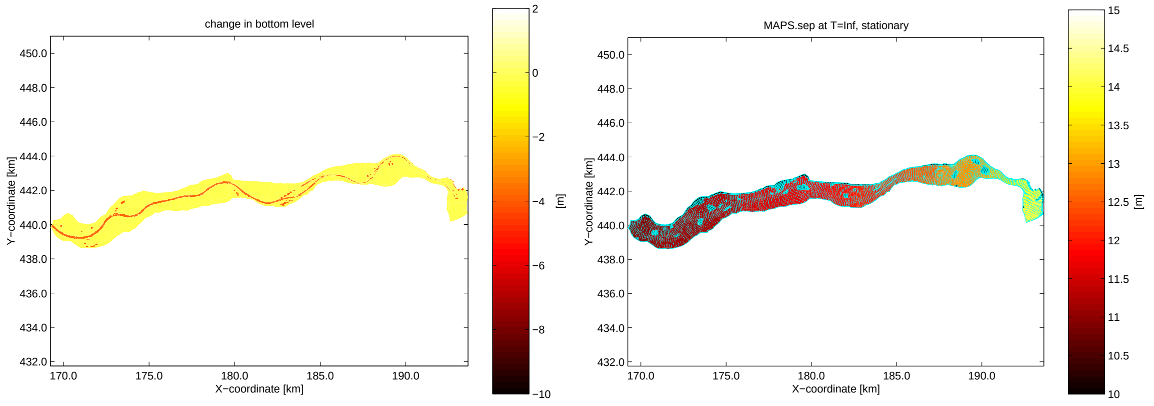


(b) Lek with an decrease of 1 meter of the bottom at the fairway (LV1).

Figure B.3: Depth intervention in Lek test problem (left) and stationary solutions of this test problem with depth interventions (right).



(a) Randwijk with some ’dams’ and ’holes’ (RA1).



(b) Randwijk with an decrease of 4 meter of the bottom at the fairway (RV4). Similarly we have tested a decrease of 2 and 1 meter of the bottom at the fairway: RV2, respectively RV1.

Figure B.4: Depth intervention in Randwijk test problem (left) and stationary solutions of this test problem with depth interventions (right).

Appendix C

Tool for analysing residuals

QuickFlow contains many tools to analyse its results. One of them generates output on the residuals and on the terms of which the residual consists. Furthermore the tool can compare the output with the results from WAQUA.

C.1 Description of the tool

In the function `shallow_water` the residual is constructed as the sum of several terms. QuickFlow used to have a tool to analyse the residuals and the terms of which it consists on a certain grid point. However, the former tool contained some errors and was not easily readable. Therefore, we have rewritten it. It is called by `analyse_nmdbg`.

In Figure C.1 an example of the output is given. The output consists of the following parts:

`Analysis output` shows the coordinates of the grid cell under consideration.

`Continuity` shows the residuals of the continuity equation.

`V-momentum` shows the residuals of the V-momentum equation, including the terms in this equation:

`pressure`	pressure term $g\frac{\partial\zeta}{\partial y}$;
`bottomfric`	bottom friction term $\frac{\tau_{\mathrm{bottom},y}}{\rho_0 H}$;
`viscosity`	viscosity term $\nu_{\mathrm{H}}\left(\frac{\partial^2 V}{\partial x^2}+\frac{\partial^2 V}{\partial y^2}\right)$;
`dvdy`	advection term $V\frac{\partial V}{\partial y}$, this term is split below for both half time steps;
`dvdx`	cross advection term $U\frac{\partial V}{\partial x}$, this term is split below for both half time steps.

As an extra check the sum of these terms is compared with the residual: if `sum - residual` is not very small either the computation of the residual or the computation of the terms incorrect.

`U-impuls` idem, for U-momentum equation.

`Schemes first half time step` shows the schemes that are used for the spatial discretization of the advection terms:

`Central`	Central differences;
`Central2`	with 2 surrounding points ($m-1$ and $m+1$);
`Central4`	with 4 surrounding points ($m-2$, $m-1$, $m+1$ and $m+2$);
`BDF`	backward differences (upwind);
`BDF1`	first order upwind;
`BDF2`	second order upwind;
`BDF1+` of `BDF2+`	upwind is 'backward': $\mathbf{U} > 0$;
`BDF1-` of `BDF2-`	upwind is 'forward': $\mathbf{U} < 0$;
`Zero-adv`	there is no advection.

Example `BDF2+`: $\left.\frac{\partial u}{\partial x}\right|_m = \frac{u_m - u_{m-1}}{\Delta x} + \frac{1}{2}\left(\frac{u_m - u_{m-1}}{\Delta x} - \frac{u_{m-1} - u_{m-2}}{\Delta x}\right).$

`Schemes second half time step` idem for the second half time step.

Later we have added the posibility to compare the results for two situations:

- shallow water full: when the function `shallow_water` constructs the Jacobian $\mathbf{F}'(\mathbf{x})$, the mass matrix M and the right hand side vector $\mathbf{b}$;
- shallow water quick: when the function `shallow_water` constructs the mass vector $\mathbf{m}$ and the residual $\mathbf{F}(\mathbf{x})$.

Furthermore we have added a function that can compare the residuals from 'shallow water quick' with the residuals computed from 'shallow water full':

$$\begin{aligned} \text{residual} &= \mathbf{F}(\mathbf{x}), \\ &= \mathbf{F}'(\mathbf{x})\,\mathbf{x} - \mathbf{b}. \end{aligned}$$

This comparison can be done visually: the residuals $\mathbf{F}(\mathbf{x})$, and $\mathbf{F}'(\mathbf{x})\,\mathbf{x} - \mathbf{b}$ are plotted for the whole domain, as well as the difference $\mathbf{F}(\mathbf{x}) - \mathbf{F}'(\mathbf{x})\,\mathbf{x} - \mathbf{b}$. Whenever these differences are large, errors are suspected in either 'shallow water full' or 'shallow water quick'.

C.2 Recommendations

Some extensions to this tool can be made:

- add possibilities to analyse points on all boundaries (some boundaries are not implemented yet); and
- extend possibilities to analyse continuity equation (only necessary when errors are suspected here).

```
======= Analysis output =======

output QuickFlow over [m,n]=[15   6]
output WAQUA over [m,n]=[15   6]

======= Continuity =======

                 QuickFlow      WAQUA

                5.6973e-07 nog niet beschikbaar

======= V-momentum =======

                      QuickFlow    WAQUA, half1    WAQUA, half2

       residual:     -7.1152e-07     -5.8593e-05      5.6052e-05

       pressure:    0.0000398585    0.0000398585    0.0000398975
     bottomfric:   -0.0000355261   -0.0000355261   -0.0000354798
      viscosity:   -0.0000045318   -0.0000045341   -0.0000044649
           dvdy:   -0.0000001108   -0.0000001108   -0.0000001090
           dvdx:   -0.0000004013   -0.0000582783    0.0000562096

      total sum:     -7.1152e-07     -5.8591e-05      5.6053e-05
 sum - residual:         1.1e-16      2.2473e-09      1.7101e-09

                                dvdx              dvdy
    QuickFlow half 1    0.0000001813   -0.0000001108
    QuickFlow half 2   -0.0000001787   -0.0000001108

======= U-momentum =======

                      QuickFlow    WAQUA, half1    WAQUA, half2

       residual:     -4.2776e-07     -9.0515e-07      5.1801e-07

       pressure:   -0.0119355125   -0.0119355130   -0.0119354740
     bottomfric:    0.0112741647    0.0112741660    0.0112741660
      viscosity:    0.0000049390    0.0000049400    0.0000049395
           dudx:    0.0006567403    0.0006567404    0.0006567276
           dudy:   -0.0000007593   -0.0000007351   -0.0000007831

      total sum:     -4.2776e-07     -4.0165e-07     -4.2402e-07
 sum - residual:      3.1768e-13       5.035e-07     -9.4203e-07

                                dudx              dudy
    QuickFlow half 1    0.0006567403    0.0002748914
    QuickFlow half 2    0.0006567403    0.0002580188

======= Schemes first half time step =======

                      QuickFlow           WAQUA

           dvdx:            BDF2+           BDF2+
           dvdy:         Central2        Central2
           dudx:         Central2        Central2
           dudy:         Central4        Central4

======= Schemes second half time step =======

                      QuickFlow           WAQUA

           dvdx:         Central4        Central4
           dvdy:         Central2        Central2
           dudx:         Central2        Central2
           dudy:            BDF2-           BDF2-
```

Figure C.1: Example output of tool for analysis of residual.

Appendix D

Profiler: Analysis of computing times

We want to decrease the computing times of QuickFlow. Therefore we need to know which part(s) of the computation cost(s) much time. We have made a 'profiler' which writes down the computing times of most parts and subparts of the computation. In Figure D.1 an example of the output of the profiler is shown.

First some information on the pseudo time steps (outer iterations) and the inner iterations (Newton steps) is shown.

Secondly the computing times for every part can be compared. We see for example that it took 27.09 seconds to do this computation, the largest part of which (12.27 seconds, 45.3 %) was taken by solving the linear system.

Thirdly some details for several parts are shown. Shallow water full refers to the setting up of the mass matrix, the Jacobian and the right hand side. Shallow water quick refers to the setting up of the mass vector and the residual.

Finally some details on the iterative method (BiCGSTAB in this case) are printed. For example, it took on average 1.5 iterations per BiCGSTAB call to find a solution that was accurate enough.

```
============  PROFILER  ============

                        number   time per step/iteration
   pseudo time steps       3                       8.73
  longest CPU, PTstep                              8.87
 shortest CPU, PTstep                              8.48
     inner iterations      6                       4.36

                        number     total CPU-time  CPU-time per time    % of total time     % of time step
       initialization      1               0.89                                     3.3
        make Jacobian      3               5.51            1.8367                  20.3                21.0
  solve linear system      6              12.27            2.0450                  45.3                46.8
          line search      6               5.33            0.8883                  19.7                20.3
               others                      3.09                                    11.4                11.8
             subTOTAL                     27.09
        visualization                      0.00                                     0.0                 0.0
                TOTAL                     27.09

              DETAILS
   shallow water full      3               5.51            1.8367                  20.3                21.0
  shallow water quick      6               3.06            0.5100                  11.3                11.7

avg nr LineSrch iter    1.00
nr failed LineSrches       0

     LU decomposition      3               8.69            2.8967                  32.1                33.2
             BiCGSTAB      6               2.78            0.4633                  10.3                10.6

  BiCGSTAB initialize      6               0.05            0.0083                   0.2                 0.2
  BiCGSTAB first half      6               2.63            0.4383                   9.7                10.0

avg nr BiCGSTAB iter    1.50
min nr BiCGSTAB iter    1.00
max nr BiCGSTAB iter    2.00
============  ========  ============
```

Figure D.1: Example output of profiler for analysis of computing times.