Eidgenössische Technische Hochschule Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

A parallel space-time finite difference solver for periodic solutions of the shallow-water equation

P. Arbenz* , A. Hiltebrand and D. Obrist†

Research Report No. 2011-72
December 2011

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

*Chair of Computational Science, ETH Zurich, Switzerland
†Institute of Fluid Dynamics, ETH Zurich, Switzerland

A parallel space-time finite difference solver for periodic solutions of the shallow-water equation

Peter Arbenz[1⋆], Andreas Hiltebrand[2], and Dominik Obrist[3]

[1] ETH Zürich, Chair of Computational Science, Zürich, Switzerland
[2] ETH Zürich, Seminar for Applied Mathematics, Zürich, Switzerland
[3] ETH Zürich, Institute of Fluid Dynamics, Zürich, Switzerland

Abstract. We investigate parallel algorithms for the solution of the shallow-water equation in a space-time framework. For periodic solutions, the discretized problem can be written as a large cyclic non-linear system of equations. This system of equations is solved with a Newton iteration which uses two levels of preconditioned GMRES solvers. The parallel performance of this algorithm is illustrated on a number of numerical experiments.

1 Introduction

In this paper we consider the shallow water equation as a model for the behavior of a fluid in a rectangular basin $\Omega = (0, L_x) \times (0, L_y)$ which is excited periodically. The excitation is caused by periodic swayings of the ground of the basin with a frequency ω, imposing a periodic behavior of the fluid with the period $T = 2\pi/\omega$ [4].

In the classical approach to solve such problems, the transient behavior of the fluid is simulated starting from an arbitrary initial state. This simulation is continued until some periodic steady-state evolves.

We model the fluid in space-time $\Omega \times [0, T)$. We will impose periodic boundary conditions in time. The discretization of the shallow water equations by finite differences in space *and* time leads to a very large nonlinear system of equations that requires parallel solution. The parallelization is done by domain decomposition where the subdomains partition space *and* time in a natural way. This is a large advantage over the classical approach where only space can be partitioned. At the same time, the number of degrees of freedom is larger by $\mathcal{O}(T/\Delta t)$.

Approaches that admit parallelization in time exist but are quite recent and not very popular yet. In the 'parareal' approach [6], the time interval $[0, T]$ is divided in subintervals. On each of these the given system of ODEs is solved. The global solution is obtained by enforcing continuity at the interfaces. Stoll and Wathen [10] discuss an 'all-at-once' approach to solve a PDE-constrained optimization problem. In this problem a state has to be controlled in such a way that it is driven into a desired final state at time T. The discretization is by finite elements in space and finite differences (backward Euler) in time. A symmetric saddle-point problem has to be solved in this approach.

⋆ Corresponding author. mailto:arbenz@inf.ethz.ch

2 Governing equations

The shallow water equations with fringe forcing and Laplacian damping are given by [4, 12]

$$\partial_t a(\mathbf{x},t) + (\boldsymbol{u}\cdot\boldsymbol{\nabla})a = -a\,\boldsymbol{\nabla}\cdot\boldsymbol{u}, \qquad \mathbf{x}\in\Omega, \quad t>0, \tag{1a}$$

$$\partial_t \boldsymbol{u}(\boldsymbol{x},t) + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} = -g\,\boldsymbol{\nabla}h + \varepsilon\Delta\boldsymbol{u} - \Lambda\,\boldsymbol{u}, \qquad \mathbf{x}\in\Omega, \quad t>0, \tag{1b}$$

$$\boldsymbol{u}\cdot\boldsymbol{n} = 0, \qquad \mathbf{x}\in\partial\Omega, \ t>0. \tag{1c}$$

Equations (1a) and (1b) model conservation of mass and momentum, respectively. Here, $\boldsymbol{u}$ is the velocity vector, h is the fluid surface level, z is the ground level, and a is the depth of the fluid (Fig. 1),

$$h(\boldsymbol{x},t) = z(\boldsymbol{x},t) + a(\boldsymbol{x},t). \tag{2}$$

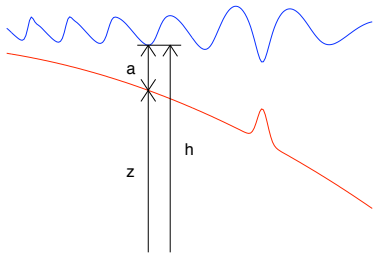


Fig. 1. Illustration of the definition of the ground level z, the water depth a and the surface level h (blue: the fluid surface h; red: the ground surface z).

The ground level z consists of a static part z_0 and a forcing term f oscillating with period T,

$$z(\boldsymbol{x},t) = z_0(\boldsymbol{x}) + f(\boldsymbol{x},t), \qquad f(\boldsymbol{x},t) = f(\boldsymbol{x},t+T).$$

The gravitational constant g in (1b) determines the phase velocity of the shallow water waves as $\sqrt{ga}$. The fringe forcing $-\Lambda\,\boldsymbol{u}$ is used to damp the waves along the boundary $\partial\Omega$ of the computational domain such that it acts like a non-reflecting outflow boundary condition. $\Lambda(\boldsymbol{x})$ is zero away from the boundary such that the forcing term has no effect in most of the interior of Ω. As $\boldsymbol{x}$ approaches $\partial\Omega$, the fringe function Λ rises smoothly to a large value such that the forcing term dominates the other terms in (1b) and we effectively solve the equation $\partial_t\boldsymbol{u} \approx -\Lambda\boldsymbol{u}$ which forces $\boldsymbol{u}$ rapidly toward zero.

The periodicity of the forcing term $f(\boldsymbol{x},t)$ transfers to the solution

$$a(\boldsymbol{x},t) = a(\boldsymbol{x},t+T), \quad \boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{u}(\boldsymbol{x},t+T), \qquad \boldsymbol{x}\in\Omega, \ t>0. \tag{3}$$

These equations give rise to periodic boundary conditions in space and time for our model (1) that we solve in $\Omega \times [0, T)$. Using the periodic boundary conditions (3) together with (1c) we get from the conservation of mass (1a) that

$$0 = \int_\Omega \partial_t a \,\mathrm{d}\boldsymbol{x} + \int_\Omega \boldsymbol{\nabla}\cdot(a\boldsymbol{u})\mathrm{d}\boldsymbol{x} = \mathrm{d}_t \int_\Omega a \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} a\boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}S = \mathrm{d}_t \int_\Omega a \,\mathrm{d}\boldsymbol{x}, \quad (4)$$

which means that the amount of fluid is conserved over time. This amount is determined by the initial data.

3 Newton iteration

Equations (1a)–(1b) can be writen in matrix form as

$$\mathcal{A}(a, \boldsymbol{u}) \begin{bmatrix} a \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} 0 \\ -g\boldsymbol{\nabla} z \end{bmatrix} \quad (5)$$

where

$$\mathcal{A}(a, \boldsymbol{u}) = \begin{bmatrix} \partial t + \boldsymbol{u}\cdot\boldsymbol{\nabla} + (\boldsymbol{\nabla}\cdot\boldsymbol{u})\cdot & 0 \\ g\boldsymbol{\nabla} & \partial t + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\cdot - \varepsilon\Delta + \Lambda\cdot \end{bmatrix}. \quad (6)$$

We use Newton's method to solve this nonlinear equation. To that end we linearize (5) at $(a_\ell, \boldsymbol{u}_\ell)$. Substituting $a = a_\ell + \delta a$ and $\boldsymbol{u} = \boldsymbol{u}_\ell + \delta\boldsymbol{u}$ in (5) and omitting higher order terms we obtain

$$\begin{aligned} &\partial_t(\delta a) + \boldsymbol{\nabla}\delta a\cdot\boldsymbol{u}_\ell + \delta a\,\boldsymbol{\nabla}\cdot\boldsymbol{u}_\ell + \boldsymbol{\nabla}a_\ell\cdot\delta\boldsymbol{u} + a_\ell\,\boldsymbol{\nabla}\cdot\delta\boldsymbol{u} = -\partial_t a_\ell - \boldsymbol{\nabla}(a_\ell\boldsymbol{u}_\ell), \\ &\partial_t(\delta\boldsymbol{u}) + (\delta\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}_\ell + (\boldsymbol{u}_\ell\cdot\boldsymbol{\nabla})\delta\boldsymbol{u} + g\boldsymbol{\nabla}\delta a + \Lambda\delta\boldsymbol{u} - \varepsilon\Delta\delta\boldsymbol{u} \\ &\qquad\qquad = -\partial_t\boldsymbol{u}_\ell - (\boldsymbol{u}_\ell\cdot\boldsymbol{\nabla})\boldsymbol{u}_\ell - g\boldsymbol{\nabla}h - \Lambda\boldsymbol{u}_\ell + \varepsilon\Delta\boldsymbol{u}_\ell, \end{aligned}$$

which can formally be written as

$$\mathcal{H}(a_\ell, \boldsymbol{u}_\ell) \begin{bmatrix} \delta a \\ \delta\boldsymbol{u} \end{bmatrix} = \begin{bmatrix} r_h \\ \boldsymbol{r}_u \end{bmatrix} \quad (7)$$

with

$$\mathcal{H}(a_\ell, \boldsymbol{u}_\ell) = \begin{bmatrix} \partial_t + \boldsymbol{u}_\ell\cdot\boldsymbol{\nabla} + \boldsymbol{\nabla}\cdot\boldsymbol{u}_\ell I & (\boldsymbol{\nabla}a_\ell)\cdot + a_\ell\,\boldsymbol{\nabla}\cdot \\ g\boldsymbol{\nabla} & \partial_t + \dfrac{\partial\boldsymbol{u}}{\partial\boldsymbol{x}} + (\boldsymbol{u}_\ell\cdot\boldsymbol{\nabla}) + \Lambda - \varepsilon\Delta \end{bmatrix}.$$

Here, $\partial\boldsymbol{u}/\partial\boldsymbol{x}$ denotes the Jacobian.

A formal algorithm for solving the nonlinear system of equations (5) now reads

Algorithm 3.1 Newton iteration for periodic solutions of the shallow water equation

1: Choose initial approximations $(a^{(0)}, \boldsymbol{u}^{(0)})$.
2: **for** $k = 0, \ldots,$ `maxIt`-1 **do**
3: Determine residuals according to (5)

$$\begin{bmatrix} r_h^{(\ell)} \\ \boldsymbol{r}_u^{(\ell)} \end{bmatrix} = \begin{bmatrix} 0 \\ -g\boldsymbol{\nabla} z \end{bmatrix} - \mathcal{A}(a^{(\ell)}, \boldsymbol{u}^{(\ell)}) \begin{bmatrix} a^{(\ell)} \\ \boldsymbol{u}^{(\ell)} \end{bmatrix}.$$

4: **If** $\|r_h^{(\ell)}\|^2 + \|\boldsymbol{r}_u^{(\ell)}\|^2 < \eta^2(\|a^{(\ell)}\|^2 + \|\boldsymbol{u}^{(\ell)}\|^2)$ **then** exit.
5: Solve

$$\mathcal{H}(a^{(\ell)}, \boldsymbol{u}^{(\ell)}) \begin{bmatrix} \delta a \\ \delta \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} r_h^{(\ell)} \\ \boldsymbol{r}_u^{(\ell)} \end{bmatrix}$$

for the Newton corrections δa, $\delta \boldsymbol{u}$.
6: Update the approximations $a^{(k+1)} := a^{(\ell)} + \delta a, \quad \boldsymbol{u}^{(k+1)} := \boldsymbol{u}^{(\ell)} + \delta \boldsymbol{u}$.
7: **end for**

4 Discretization

We approximate the shallow-water equation (1) by finite differences in space and time [3,5]. To that end we define grid points

$$(x_i, y_j, t_k) = (i\Delta x, j\Delta y, k\Delta t), \quad \Delta x = L_x/N_x, \ \Delta y = L_y/N_y, \ \Delta t = T/N_t. \quad (8)$$

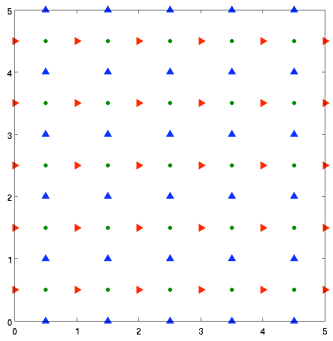


Fig. 2. Grid points for u (▶), v (▲), a (●). Here, $N_x = N_y = 5$.

In this and the following sections we write $\boldsymbol{x} = (x, y)$ and $\boldsymbol{u} = (u, v)$. We approximate the functions a, u, and v at points with (partly) non-integer indices as indicated in Fig. 2,

$$\begin{aligned} a^{(k)}_{i+\frac{1}{2},j+\frac{1}{2}} &\approx a(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}, t_k), \\ u^{(k)}_{i,j+\frac{1}{2}} &\approx u(x_i, y_{j+\frac{1}{2}}, t_k), \\ v^{(k)}_{i+\frac{1}{2},j} &\approx v(x_{i+\frac{1}{2}}, y_j, t_k). \end{aligned}$$

The corresponding grids are called h-grid, u-grid, and v-grid, with N_xN_y, $(N_x-1)N_y$, and $N_x(N_y-1)$ *interior* grid points, respectively. Notice that values of u and v at grid points on the boundary vanish according to (1c).

The finite difference equations corresponding to (1) are defined in one of these grids. Function values of h, u, and v that are required on another grid than where they are defined are obtained through *linear interpolation*. In (1) derivatives in

x- and y-direction are replaced by central differences. The derivatives in time are discretized by a fourth-order 'slightly backward-facing' finite difference stencil,

$$\frac{\partial}{\partial t} f(t) \approx \frac{1}{12\Delta t} \left[3f(t+\Delta t) + 10f(t) - 18f(t-\Delta t) + 6f(t-2\Delta t) - f(t-3\Delta t)\right].$$

The systems (5) and (7) can now be transcribed in systems in the $\sim N_t N_x N_y$ unknowns $a^{(k)}_{i+\frac{1}{2},j+\frac{1}{2}}$, $u^{(k)}_{i,j+\frac{1}{2}}$, and $v^{(k)}_{i+\frac{1}{2},j}$, see [3] for details. Most of the time in Algorithm 3.1 is spent in step 5 in the solution of the linear system (7). The straightforward second-order central difference $\frac{1}{2\Delta t}(f(t+\Delta t) - f(t-\Delta t))$ could also be used to approximate $\partial f/\partial t$. It however has drawbacks. Most of all, it splits the problem in two independent subproblems if N_t is even.

5 Numerical solution

The system matrix $\mathcal{H}$ in (7) has a block structure typical for three-dimensional finite difference discretizations (Fig. 3). The principal 3×3 block structure of $\mathcal{H}$ stems from the three components a, u, and v. The components corresponding to the spatial derivatives give rise to tridiagonal blocks. The components corresponding to the temporal derivative entail cyclic blocks with five bands. These bands can be seen quite well in Fig. 3 because the index in t-direction varies more slowly than the indices of the spatial directions.

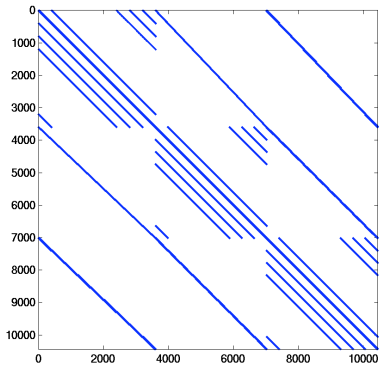


Fig. 3. MATLAB spy of $\mathcal{H}$ for $N_x = N_y = 20$ and $N_t = 9$. The most prominent 3×3 block structure stems from the three components a, u, and v. The diagonal blocks are cyclic.

As $\mathcal{H}$ is non-symmetric we solve system (7) by the GMRES algorithm [9]. The preconditioner is based on the specific structure of $\mathcal{H}$. In a first step, we approximate $\mathcal{H}$ by replacing all diagonal and off-diagonal blocks with the closest

cyclic or Toeplitz blocks [1], i.e., the elements in each (off-)diagonal of a block are replaced by the average of the respective (off-)diagonal. The cyclic $N_t \times N_t$ blocks of the modified $\mathcal{H}$ can now be diagonalized by applying FFTs from left and right. There are $N_x(N_y-1)+(N_x-1)N_y+N_xN_y$ independent FFTs to be applied from either side. Notice that the eigenvalues of the cyclic blocks come in complex conjugate pairs such that complex arithmetics is required in the following steps.

After these diagonalizations, the matrix splits in $N_t' = (N_t-1)/2$ separate but complex spatial problems, corresponding to the individual Fourier modes $k = 0, \ldots, N_t'$. In general, the systems are still too large to be factored. Therefore, in the second step, we use preconditioned GMRES solvers to solve the N_t systems of equations in parallel. We form the preconditioner with the same recipe as before: quantities along the x-axis with equal y-coordinate are averaged to make the tridiagonal blocks Toeplitz blocks. After the diagonalization by FFTs, we arrive at $N_t \cdot N_x$ independent $N_y \times N_y$ problems which can be solved by Gaussian elimination.

6 Parallelization

Our code is parallelized making use of Trilinos [2, 11], a collection of numerical software packages. We use the GMRES solver in the package AztecOO. As AztecOO can only solve real valued systems, we additionally use the package Komplex that generates an equivalent real valued problem out of a complex problem by splitting real and imaginary parts. This is not optimal, because the current implementation of this package explicitly generates the real valued system, which uses again time and memory space. One could avoid this overhead by using Belos, which is also able to solve complex systems (by means of generic programming). Additionally we use the FFTW library for the Fourier transforms and DGBTRF and DGBTRS from LAPACK to solve the banded systems.

In general, the matrices and vectors are distributed such that each process gets approximately the same number of rows. For $\mathcal{A}$ and $\mathcal{H}$, the spatial domain is divided into m_x intervals along the x-axis and m_y intervals along the y-axis. The time domain is divided into m_t intervals. Each process gets then equations corresponding to all variables in one of these $m_xm_ym_t$ blocks. Because of this partitioning we set the number of processes equal to $m_xm_ym_t$.

To construct the small systems we need to average and then to perform a Fourier transform in the time direction. To do this efficiently we need to have the entries for a certain spatial point at all time steps local in one processor's memory. So, the matrix is redistributed in "stripes", where each processor gets all rows corresponding to all time steps for some quantities. To construct the smallest systems we need to have the variables in x-direction grouped together on one processor. During the solution process, we need to have the smallest systems (each connect all y-values) local on one process, such that the (serial) factorization and forward and backward substitution can be performed. While the matrices need only to be redistributed in the beginning of each Newton

iteration step, the solution vectors and right hand sides have to be redistributed prior to every outer GMRES iteration step. In the inner GMRES iterations the vectors need only to be redistributed within the group of processes that solves a certain system. Apart from the fact that each redistribution consumes time, it also uses memory.

Solving the N_t systems in parallel is not straightforward as their condition numbers vary considerably. Some of the systems are diagonally dominant while others are close to singular, cf. Fig. 7. It turned out that the iteration count of one solve is a good prediction of the iteration count of the next solve of the same system. This makes it possible to design a simple, adaptive, and effective load balancing strategy and assign fewer or more processors to a solver. Notice that the matrices split in N_y subblocks which makes the parallelization of these solvers straightforward.

7 Experiments

For the numerical experiments, we use a configuration which yields a periodic steady-state solution with a primary and a secondary wave system (Fig. 4). The primary wave system is generated by a localized oscillation at the ground in the shape of a Gaussian. These waves propagate toward the boundary of the computational domain where they are damped out by the fringe forcing such that no waves are reflected from the boundaries. The propagation of these waves is non-uniform because the waves steepen and their speed is reduced in the more shallow regions of the basin. A small submerged hill in the shallow region of the basin leads to reflections of the primary waves. This results in a secondary wave system which is centered at this hill. Figure 5 shows four snapshots of the periodic steady-state solution taken at $t = 0$, $T/4$, $T/2$, and $3T/4$, respectively.

The discretization parameters of the numerical experiments are listed in Table 1. Note that case 3 uses fewer grid points in the t-direction due to memory limitations.

Table 1. Discretization parameters for the numerical experiments

case	N_x	N_y	N_t
1	100	100	99
2	200	200	99
3	400	400	39

All numerical experiments from Table 1 converge relatively fast (Fig. 6(a)). The residual norm is reduced approximately by an order of magnitude per Newton iteration such that sufficiently converged solutions can be obtained typically within less than ten Newton steps. In the wave system with a hill we observe convergence problems at least in one case. To encounter them, we will in the future increase the robustness of the Newton iteration by backtracking (line search, damping) [8].

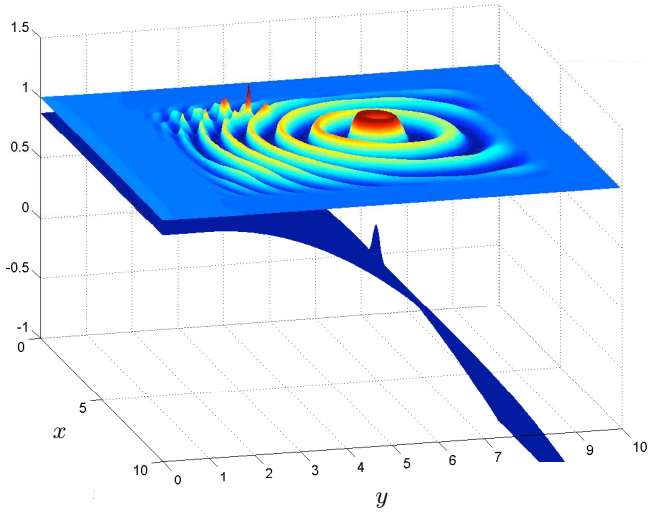
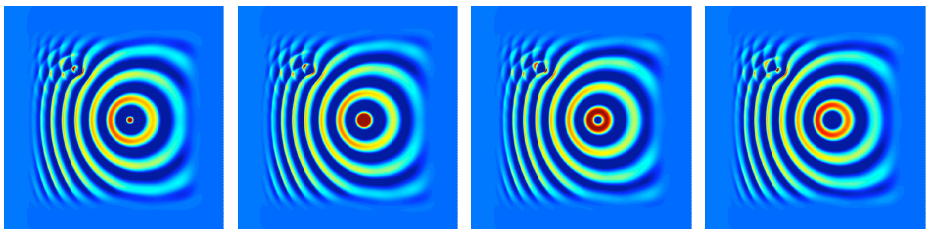


Fig. 4. Configuration for the numerical experiments with an oscillating perturbation in the center and a small submerged hill in the shallow region of the basin (top left).



(a) $t = 0$ (b) $t = T/4$ (c) $t = T/2$ (d) $t = 3T/4$

Fig. 5. Snapshots of the converged periodic solution at different phases.

The numbers of outer GMRES iterations per Newton step is shown in Fig. 7a. It increases successively with each Newton step. The number of inner GMRES iterations (Fig. 7b) shows a strong dependence on the Fourier mode number k. The required work for the small mode numbers is significantly larger than for high mode numbers. This effect can be explained by the diagonal dominance of the inner systems of equations. The mode number k corresponds physically to a frequency and enters the inner systems as a factor on the diagonal entries. Therefore, the inner systems for high mode numbers (high frequencies, $k \approx N_t'$) are diagonally dominant and are easier to solve than the inner systems for small k.

The strong dependence between the mode number k and the number of inner GMRES iterations could lead to a strong load imbalance for the parallel tasks. Therefore, a dynamic scheduling of the inner GMRES solutions was used. It tries to balance the workload across the parallel threads. To this end, the numbers of

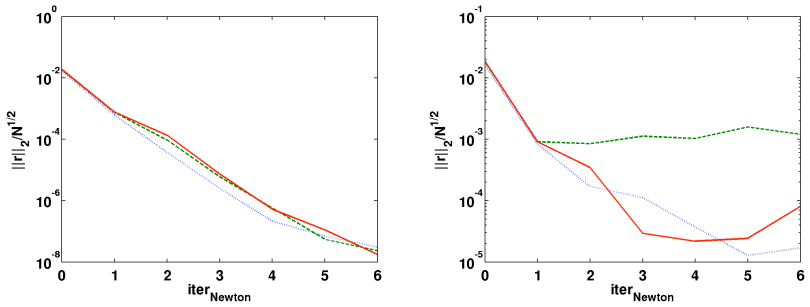


Fig. 6. Convergence of the residual norm over the Newton iterations: (a) without hill (no secondary wave system), (b) with a hill (dotted blue: case 1, dashed green: case 2, solid red: case 3).

inner iterations from the previous Newton step are used to estimate an optimal distribution of the parallel tasks to the different threads.

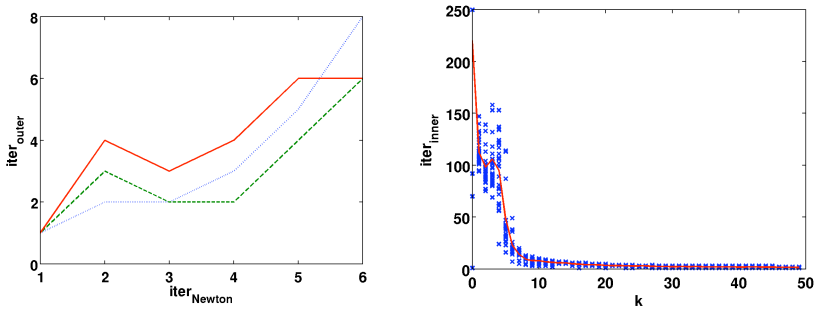
The parallel performance of the solver was measured on the large Brutus cluster at ETH Zurich[4] with AMD Opteron 8380 Quad-Core CPUs and an Infiniband QDR network. Typical turn-around times on 100 cores for case 3 were approximately 430 s. In these simulations, the spatial grid was decomposed into 5 subdomains in each direction and the temporal direction was split into 4 intervals.

Figure 8 illustrates the speed-up and parallel efficiency of the solver by increasing the number of cores from 1 to 100 for the case 1. In view of the relatively small size of case 1, the relevant modules of the solver (solution, construction of the preconditioner, and update of the Newton matrix) scale reasonably well. The immediate drop in the efficiency from 1 to multiple processes is due to the reshuffling of the data to localize the FFT's which is not necessary for $N_{\text{process}} = 1$.

8 Conclusions

We have presented a concept for the parallelization of a periodic shallow-water problem in space *and* in time. The presented results illustrate that the algorithm converges quickly towards the steady-state solution of the problem. Numerical experiments of different sizes show that the algorithm scales reasonably well. Nevertheless, a more efficient preconditioner would improve the performance of the present algorithm. It could be promising to further exploit the periodicity of the sought solution by replacing, for instance, the finite-difference stencil for the time derivative by a Fourier spectral method. This approach has been investigated by the authors for a one-dimensional Burgers equation [7].

[4] http://www.clusterwiki.ethz.ch/legacy/Brutus



(a) (b)

Fig. 7. Number of inner/outer GMRES iterations for case 1 without hill: (a) outer GMRES iterations per Newton iteration (dotted blue: case 1, dashed green: case 2, solid red: case 3), (b) inner GMRES iterations as a function of the mode number k (blue $\times$: single instances of the inner GMRES; solid red: average over all instances of the inner GMRES).

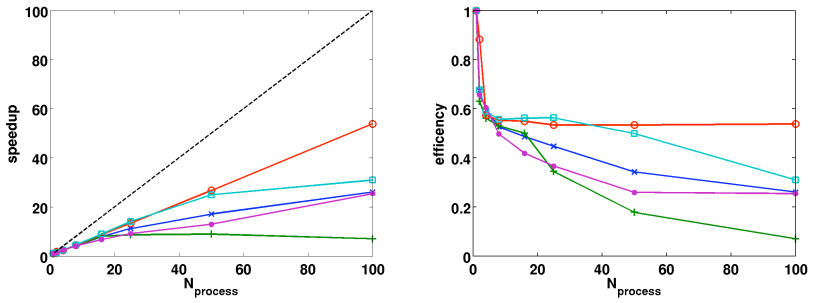
The proposed algorithm should be seen in the context of the current development of supercomputers. The availability of more and more processing units will require modern solvers for fluid dynamics problems to distribute the work to ever more parallel threads. For a large class of fluid dynamics problems with (quasi-) periodic steady-state solutions, this could be achieved by parallelizing the time domain, in addition to a state-of-the-art domain decomposition in space. The classical solution procedure consists of a (often explicit Runge-Kutta) time stepping scheme that is executed for about five periods until the (quasi-) periodic solution is reached. (Here, a period refers to the time for the slowest wave to traverse the domain.) If the spatial grid is refined for accuracy reasons, the CFL stability condition requires that the time step size is reduced proportionally despite a smooth temporal behavior of the solution. Finally, parallelism is restricted to the space dimension(s).

The space-time approach requires additional memory space to store the matrix and preconditioner in the Newton step. But even if a space-time approach leads to an increased overall workload, the turn-around time of the latter can be reduced due to the increased parallelism.

References

[1] T. F. Chan. An optimal circulant preconditioner for Toeplitz systems. *SIAM J. Sci. Stat. Comput.*, 9:766–771, 1988.

[2] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. An overview of the Trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.



(a) (b)

Fig. 8. (a) Parallel speedup and (b) efficiency for the solution of case 1 (green +: initialization; red ○: update; light blue □: preconditioner construction; violet •: solution; dark blue ×: overall; dashed black: ideal speedup).

[3] A. Hiltebrand. Parallel solution of time-periodic problems. Master thesis, ETH Zurich, Institute of Fluid Dynamics, March 2011.

[4] J. Kevorkian. *Partial Differential Equations: Analytical Solution Techniques*. Springer, New York, NY, 2nd edition, 2000.

[5] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, Philadelphia, PA, 2007.

[6] J.-L. Lions, Y. Maday, and G. Turinici. A "parareal" in time discretization of PDE's. *C. R. Math. Acad. Sci. Paris*, 332(7):661–668, 2001.

[7] D. Obrist, R. Henniger, and P. Arbenz. Parallelization of the time integration for time-periodic flow problems. *PAMM*, 10(1):567–568, 2010.

[8] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker. Globalization techniques for Newton–Krylov methods and applications to the fully coupled solution of the Navier–Stokes equations. *SIAM Rev.*, 48(4):700–721, 2006.

[9] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2nd edition, 2003.

[10] M. Stoll and A. Wathen. All-at-once solution of time-dependent PDE-constrained optimization problems. Technical Report 10/47, Oxford Centre for Collaborative Applied Mathematics, Oxford, England, 2010.

[11] The Trilinos Project Home Page. http://trilinos.sandia.gov/.

[12] C. B. Vreugdenhil. *Numerical Methods for Shallow-Water Flow*. Kluwer, Dordrecht, 1994.

Research Reports

No.	Authors/Title
11-72	*P. Arbenz, A. Hiltebrand and D. Obrist* A parallel space-time finite difference solver for periodic solutions of the shallow-water equation
11-71	*M.H. Gutknecht* Spectral deflation in Krylov solvers: A theory of coordinate space based methods
11-70	*S. Mishra, Ch. Schwab and J. Šukys* Multi-level Monte Carlo finite volume methods for shallow water equations with uncertain topography in multi-dimensions
11-69	*Ch. Schwab and E. Süli* Adaptive Galerkin approximation algorithms for partial differential equations in infinite dimensions
11-68	*A. Barth and A. Lang* Multilevel Monte Carlo method with applications to stochastic partial differential equations
11-67	*C. Effenberger and D. Kressner* Chebyshev interpolation for nonlinear eigenvalue problems
11-66	*R. Guberovic, Ch. Schwab and R. Stevenson* Space-time variational saddle point formulations of Stokes and Navier-Stokes equations
11-65	*J. Li, H. Liu and H. Sun* Enhanced approximate cloaking by SH and FSH lining
11-64	*M. Hansen and Ch. Schwab* Analytic regularity and best N-term approximation of high dimensional parametric initial value problems
11-63	*R. Hiptmair, G. Phillips and G. Sinha* Multiple point evaluation on combined tensor product supports
11-62	*J. Li, M. Li and S. Mao* Convergence analysis of an adaptive finite element method for distributed flux reconstruction
11-61	*J. Li, M. Li and S. Mao* A priori error estimates of a finite element method for distributed flux reconstruction
11-60	*H. Heumann and R. Hiptmair* Refined convergence theory for semi-Lagrangian schemes for pure advection