A massively scalable distributed multigrid framework for nonlinear marine hydrodynamics

Glimberg, Stefan Lemvig; Engsig-Karup, Allan Peter; Olson, Luke N.

Published in:
International Journal of High Performance Computing Applications

Link to article, DOI:
[10.1177/1094342019826662](https://doi.org/10.1177/1094342019826662)

Publication date:
2019

Document Version
Peer reviewed version

[Link back to DTU Orbit](#)

Citation (APA):
Glimberg, S. L., Engsig-Karup, A. P., & Olson, L. N. (2019). A massively scalable distributed multigrid framework for nonlinear marine hydrodynamics. *International Journal of High Performance Computing Applications*, 33(5), 855-868 . <https://doi.org/10.1177/1094342019826662>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/318636278>

A Massively Scalable Distributed Multigrid Framework for Nonlinear Marine Hydrodynamics

Article in *International Journal of High Performance Computing Applications* · November 2018

DOI: 10.1177/1094342019826662

CITATION

1

READS

329

3 authors:

[Stefan Lemvig Glimberg](#)

TracInnovations

15 PUBLICATIONS 100 CITATIONS

[SEE PROFILE](#)

[Allan Peter Engsig-Karup](#)

Technical University of Denmark

88 PUBLICATIONS 734 CITATIONS

[SEE PROFILE](#)

[Luke Olson](#)

University of Illinois, Urbana-Champaign

68 PUBLICATIONS 965 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:

DTU Compute GPULAB LIBRARY for massively parallel computations on modern many-core hardware [View project](#)

OceanWave3D (Open Source Marine Hydrodynamics) [View project](#)

A Massively Scalable Distributed Multigrid Framework for Nonlinear Marine Hydrodynamics

Journal Title

XX(X):2–21

© The Author(s) 0000

Reprints and permission:

sagepub.co.uk/journalsPermissions.nav

DOI: 10.1177/ToBeAssigned

www.sagepub.com/

S. L. Glimberg¹, A. P. Engsig-Karup^{1,2}, Luke N. Olson³

Abstract

The focus of this paper is on the parallel scalability of a distributed multigrid framework, known as the DTU Compute GPUlab Library, for execution on GPU-accelerated supercomputers. We demonstrate near-ideal weak scalability for a high-order fully nonlinear potential flow (FNPF) time domain model on the Oak Ridge Titan supercomputer, which is equipped with a large number of many-core CPU-GPU nodes. The high-order finite difference scheme for the solver is implemented to expose data locality and scalability, and the linear Laplace solver is based on an iterative multilevel preconditioned defect correction method designed for high-throughput processing and massive parallelism. In this work, the FNPF discretization is based on a multi-block discretization which allows for large-scale simulations. In this setup, each grid block is based on a logically structured mesh with support for curvilinear representation of horizontal block boundaries in order to allow for an accurate representation of geometric features such as surface-piercing bottom-mounted structures — e.g. mono-pile foundations as demonstrated. Unprecedented performance and scalability results are presented for a system of equations that is historically known as being too expensive to solve in practical applications. A novel feature of the potential flow model is demonstrated, being that a modest number of multigrid restrictions is sufficient for fast convergence, improving overall parallel scalability as the coarse grid problem diminishes. In the numerical benchmarks presented, we demonstrate using 8,192 modern Nvidia GPUs enabling large scale and high-resolution nonlinear marine hydrodynamics applications.

Keywords

High-performance computing, multi-GPU, domain decomposition, Laplace problem, multi-block solver, geometric multigrid, heterogeneous computing, free surface water waves.

1 Introduction

The objective of this work is to detail and benchmark a newly developed distributed multi-grid framework, which we refer to as the DTU Compute GPUlab Library²⁰. The framework offers scalable execution on supercomputers and compute cluster with heterogeneous architectures equipped with many-core coprocessors such as GPUs. The GPUlab library builds upon Thrust³ and the Message Passing Interface²⁴ (MPI) and is designed for extensibility and portability, allowing for fast prototyping of advanced numerical solvers. In addition, by using a layer of abstraction, the library delivers an accessible interface, allowing developers to realize high-performance through auto-tuning. The generic design principles used for implementing the library components are described in¹⁹.

To target marine hydrodynamic simulations, we have developed a GPUlab module based on fully nonlinear potential flow theory for free surface flows. This module is referred to as OceanWave3D-GPU and is developed to achieve $\mathcal{O}(n)$ -scaling with n degrees-of-freedom in the discretization. To achieve scalability, we utilize a massively parallel multi-level preconditioned defect correction scheme proposed in¹². In this paper, we detail several extensions including the use of a multi-block procedure to enable multi-GPU computing, and the development of boundary-fitted curvilinear domains to support real hydrodynamics application. Together, the OceanWave3D-GPU solver is capable of targeting large marine areas due to the high parallel performance and low memory footprint in the simulation.

The OceanWave3D-GPU simulator is an important step in expanding the concept of Numerical Wave Tanks (NWTs)³⁸ toward more complex marine hydrodynamic applications using fully nonlinear potential flow models. Fast hydrodynamics codes have shown potential in real-time naval hydrodynamics simulations and visualisation^{12,20,34}, efficient uncertainty quantification⁵, tsunami propagation²³, and the development, design, and analysis of marine structures such as naval vessels, wave-energy conversion devices⁴⁵ and coastal infrastructures. The multitude of applications motivates the use of both nonlinear wave-wave and wave-structure interactions.

1.1 Applications for Large-Scale Water Wave Simulations

Accurate description of the nonlinear evolution and transformation of ocean waves over possibly changing sea floors is important in the context of ocean modeling and engineering. Recent advances in scalable numerical algorithms allow us to consider fully nonlinear and dispersive wave equations based on full potential theory in the context of large-scale simulations, for which CFD based models are too computationally expensive²⁵. The immediate

¹Department of Applied Mathematics and Computer Science. Technical University of Denmark

²Center for Energy Resources Engineering (CERE). Technical University of Denmark

³Department of Computer Science, University of Illinois at Urbana-Champaign

Corresponding author:

Stefan L. Glimberg

Email: stefanglimberg@gmail.com

advantage is a more complete description in the model and a wider applicability for hydrodynamics calculations, though still subject to the assumptions of non-breaking and irrotational waves. However, model extensions to account for breaking waves are possible²⁷. Simulations based on the fully nonlinear and dispersive equations are attractive for engineering applications where water waves transform nonlinearly over varying bathymetry and when accurate kinematics are required — e.g., the estimation of structural loads. With a large-scale model, wave-wave interactions can be modeled even in the presence of multiple structures covering large areas, such as offshore wind farms.

A comprehensive introduction to hydrodynamics is given in⁴³, while details on numerical modeling are found in³³ or in the shorter survey by Dias and Bridges⁸. A variety of numerical techniques for solving such hydrodynamic model problems have been presented throughout the literature — e.g., see^{7,10,30,46}. Finally, a concise derivation of the fully nonlinear water wave model based on potential flow theory is revisited in¹³.

1.2 Coprocessors

High-performance computing has been transformed over the last decade with the emergence of programmable many-core coprocessors, such as graphics processing units (GPUs). This is driving the need to develop software and algorithms that exploit the parallelism offered by their compute units.

Models based on full potential theory have previously used a variety of discretization methods. Due to the memory layout and access

efficiency on a GPU, the finite difference method has been used to efficiently solve the governing potential flow equations. Memory bandwidth is the performance bottleneck for most scientific computing applications and efficient memory access is therefore key for overall efficiency; this is also supported by recent trends in hardware development^{2,26,36}.

In recent work, we have addressed aspects of robust, accurate and fast three-dimensional simulation of water waves over uneven sea floors^{11,12,18} based on full potential theory. The objective of this work is to enable new hydrodynamic applications by extending the scalability on distributed systems for use with a large number of time steps and billions of spatial degrees of freedom. To this end, we present and benchmark an extension of the parallel model¹² that supports large GPU-accelerated systems by using a hybrid MPI-CUDA implementation.

This work is related to the methods originally proposed by Li and Fleming^{30,31}, which is based on multigrid using low-order discretization. This was extended to a high-order finite difference technique in 2D⁶ and to 3D¹¹, and the approach enables efficient, scalable, and low-storage solution of the Laplace problem using an efficient multigrid strategy proposed and analyzed in^{12,14}.

A novel feature of our approach is that communication requirements are minimized due to a low number of multigrid restrictions, leading to low overhead on massively parallel machines and high scalability. Our benchmark examples verify that the combination of a strong vertical smoother and a favorable coarsening

strategy based on aspect ratios provides good convergence at even few multigrid levels. Reducing the number of multigrid levels leads to a lower communication overhead.

1.3 On GPU accelerated Multigrid solvers for PDE applications

The use of coprocessors, such as GPUs, for elliptic problems was first reported in¹⁶. In the setting of finite elements, weak scalability is explored in²² and later on clusters in²¹. Additional research on both geometric and algebraic multigrid methods have been demonstrated on a number of GPU accelerated applications in various engineering disciplines^{4,15,37,42}. The increasing ratio between compute performance and memory bandwidth favors higher-order discretizations that increase the flop counts to memory transfers and make it possible to tune trade-offs between accuracy and numerical efficiency.

1.4 Paper Contributions

We present a novel and scalable algorithmic implementation for the fully nonlinear potential flow model enabling large-scale marine hydrodynamics simulation in realistic engineering applications. The implementation extends the proof-of-concept for a single-GPU system demonstrated in¹². This is facilitated by a multigrid framework described in²⁰ that is based on a MPI-CUDA implementation. The framework handles curvilinear and logically structured meshes in order to introduce geometric flexibility — e.g. the simulation of mono-pile foundations in marine areas. Through a careful semi-coarsening strategy, we demonstrate that

the rapid convergence of the iterative solver requires only a few multigrid levels due to the finite speed of the hyperbolic surface condition, avoiding an expensive coarse grid problem that typically limits scalability. We provide a benchmark and demonstrate weak scalability on several compute clusters, including the Oak Ridge Titan supercomputer, utilizing up to 8,192 GPUs and enabling high-resolution and large-scale marine hydrodynamic simulation.

2 The Mathematical Model

In recent works, we have focused on the robustness of the numerical scheme^{11,14,30} along with efficiency on emerging many-core architectures^{12,18}. A mathematical model for potential free surface flow in three spatial dimensions can be expressed in terms of the unsteady kinematic and dynamic boundary equations in Zakharov form

$$\partial_t \eta = -\nabla \eta \cdot \nabla \tilde{\phi} + \tilde{w}(1 + \nabla \eta \cdot \nabla \eta), \quad (1a)$$

$$\begin{aligned} \partial_t \tilde{\phi} = & -g\eta - \frac{1}{2} \nabla \tilde{\phi} \cdot \nabla \tilde{\phi} \\ & - \frac{\tilde{w}^2}{2} (1 + \nabla \eta \cdot \nabla \eta), \end{aligned} \quad (1b)$$

where $\nabla \equiv [\partial_x \partial_y]^T$ is the horizontal gradient operator, g is the gravitational acceleration, and $\eta(\mathbf{x}, t)$ is the free surface elevation measured from the still-water level at $z = 0$. In addition, $\tilde{\phi}(\mathbf{x}, t)$ is the scalar velocity potential, and $\tilde{w}(\mathbf{x}, t)$ is the vertical velocity; both are evaluated at the free surface $z = \eta$. Assuming that the flow is inviscid, irrotational, and non-breaking, the continuity equation can be expressed in terms of a scalar velocity potential

function in the form

$$[\nabla^2 + \partial_{zz}]\phi = 0, \quad (2)$$

which requires a closure for the free surface equations. By imposing the free surface boundary conditions (1b) together with a kinematic bottom boundary condition

$$\partial_z \phi + \nabla h \cdot \nabla \phi = 0, \quad z = -h, \quad (3)$$

and assuming impermeable vertical walls at domain boundaries, the resulting Laplace problem is well-posed. From the solution, the velocity field in the fluid volume is $(\mathbf{u}, w) \equiv (\nabla \phi, \partial_z \phi)$.

2.1 Vertical coordinate transformation

To improve efficiency, a fixed time-invariant computational domain is introduced using a classical σ -transformation:

$$\sigma(\mathbf{x}, z, t) = \frac{z + h(\mathbf{x}, t)}{h(\mathbf{x}) + \eta(\mathbf{x}, t)}, \quad (4)$$

which maps $z \mapsto \sigma$ and where $-h \leq z \leq \eta$ and $0 \leq \sigma \leq 1$. The resulting transformed Laplace problem with $\phi(\mathbf{x}, z) \equiv \Phi(\mathbf{x}, \sigma)$ is written as, $\sigma = 1$:

$$\Phi = \tilde{\phi}, \quad (5a)$$

$0 \leq \sigma < 1$:

$$\begin{aligned} \nabla^2 \Phi + (\nabla^2 \sigma)(\partial_\sigma \Phi) + 2 \nabla \sigma \cdot \nabla (\partial_\sigma \Phi) + \\ (\nabla \sigma \cdot \nabla \sigma + (\partial_z \sigma)^2) \partial_{\sigma\sigma} \Phi = 0 \end{aligned} \quad (5b)$$

$\sigma = 0$:

$$(\partial_z \sigma + \nabla h \cdot \nabla \sigma)(\partial_\sigma \Phi) + \nabla h \cdot \nabla \Phi = 0. \quad (5c)$$

These equations, together with the kinematic and dynamic free surface boundary conditions, describe the fully nonlinear potential flow model in Cartesian coordinates, applicable to applications where the physical domain can be represented within a numerical wave tank with regular boundaries.

2.2 Horizontal Mapping to Generalized Coordinates

Irregular horizontal domains can be introduced by rewriting the Cartesian coordinates and by expressing the physical derivatives in terms of the computational domain. This is accomplished by introducing the affine coordinate mapping $\chi(x, y) = \{(x, y) \mapsto (\xi, \gamma)\}$. Then, applying the chain rule, first-order derivatives with respect to the original physical coordinates (x, y) of a function u are described within the regular computational domain (ξ, γ) with the following relations (sub-scripts are used to denote derivatives):

$$u_x = \xi_x u_\xi + \gamma_x u_\gamma, \quad (6a)$$

$$u_y = \xi_y u_\xi + \gamma_y u_\gamma, \quad (6b)$$

under which the following relations hold

$$\xi_x = \frac{1}{J} y_\gamma, \quad \xi_y = -\frac{1}{J} x_\gamma, \quad (7a)$$

$$\gamma_x = -\frac{1}{J} y_\xi, \quad \gamma_y = \frac{1}{J} x_\xi, \quad (7b)$$

where J is the Jacobian of the mapping,

$$J = \det(\mathbf{J}) = x_\xi y_\gamma - x_\gamma y_\xi. \quad (8)$$

Given the above equations, first order derivatives of u can be described exclusively within the computational reference domain as

$$u_x = \frac{y_\gamma u_\xi - y_\xi u_\gamma}{J}, \quad (9a)$$

$$u_y = \frac{x_\xi u_\gamma - x_\gamma u_\xi}{J}. \quad (9b)$$

In a similar way, second-order and mixed derivatives can be derived. Under the assumption of time-invariant domains the Jacobian and its derivatives are constant and may be precomputed to improve computational efficiency.

Having assumed impermeable wall boundaries, the net-flux through these boundaries is zero and is stated in terms of a homogeneous Neumann boundary condition

$$\mathbf{n} \cdot \nabla u = 0, \quad (x, y) \in \partial\Omega, \quad (10)$$

where $\mathbf{n} = (n_x, n_y)^T$ is the outward pointing horizontal normal vector defined at the vertical boundary $\partial\Omega$ surrounding the domain Ω . The normal vector in the physical and generalized coordinates are connected through the following relation

$$\begin{pmatrix} n_x \\ n_y \end{pmatrix} = \begin{pmatrix} \frac{\xi_x}{\sqrt{\xi_x^2 + \gamma_x^2}} & \frac{\gamma_x}{\sqrt{\xi_x^2 + \gamma_x^2}} \\ \frac{\xi_y}{\sqrt{\xi_y^2 + \gamma_y^2}} & \frac{\gamma_y}{\sqrt{\xi_y^2 + \gamma_y^2}} \end{pmatrix} \begin{pmatrix} n_\xi \\ n_\gamma \end{pmatrix}. \quad (11)$$

Note that the normals in the computational domain are trivial, as they are always perpendicular to the rectangular domain boundaries.

Supporting curvilinear coordinates enables complex modeling of off-shore structures, harbors, shorelines, etc., which have significant engineering value over traditional regular wave tank domains^{17,32,39,40,47}. Flexible representation of the discrete domain can also be utilized to spatially adapt the grid to the wavelengths, resulting in higher resolution where shorter waves are present and thereby minimizing over- or under-resolved waves.

3 The Numerical Model

All spatial derivatives in (1b), (1a), and (5) are discretized using standard centered finite differences of flexible-order accuracy, meaning that the size of the stencil operators can be pre-adjusted to allow different orders of accuracy using the method of undetermined coefficients. The stencil implementation is matrix-free to avoid the use of sparse matrix formats, which would require additional index look ups from memory. The stencil operations also allow efficient access to GPU memory as data can be cached³⁶. The temporal surface conditions in (1b) and (1a) are solved with a fourth-order explicit Runge-Kutta method. The time domain $[0, T_{\text{end}}]$ is partitioned uniformly into discrete steps $t_i = i\Delta t, i = 1, 2, \dots$, such that Δt satisfies the CFL stability condition. At each stage of the Runge-Kutta method the vertical velocity at the surface, $\tilde{w}$, is approximated as the vertical derivative of the velocity potential Φ , where Φ is described by the linear system of equations due to the discretization of (5):

$$\mathcal{A}\Phi = \mathbf{b}, \quad \mathcal{A} \in \mathbb{R}^{N \times N}, \mathbf{b} \in \mathbb{R}^{N \times 1}, \quad (12)$$

where N is the total degrees of freedom. The sparse, non-symmetric system (12) is a key computational bottleneck in the simulation and is the focus of our attention. To extend the original work by Li and Fleming³⁰ a multigrid preconditioned defect correction method can be employed, which has been expanded for improved accuracy, efficiency, and robustness^{6,11,12}. The iterative defect correction method is attractive because it has a short and highly parallel outer loop, requiring only one collective all-to-one communication step when computing the residual norm for evaluating the convergence criteria. The preconditioned defect correction method generates a sequence of iterates j , starting from an initial guess $\Phi^{[0]}$,

$$\Phi^{[j]} = \Phi^{[j-1]} + \mathcal{M}^{-1} \mathbf{r}^{[j-1]}, \quad j = 1, 2, \dots, \quad (13)$$

where $\mathbf{r}^{[j-1]} = \mathbf{b} - \mathcal{A}\Phi^{[j-1]}$ denotes the residual at iteration $j - 1$ and $\mathcal{M}$ is a left preconditioner based on nested grids to form a geometric multigrid preconditioner⁴⁴. The iterations continue until the convergence criteria is satisfied,

$$\|\mathbf{r}^{[j]}\| \leq rtol\|\mathbf{b}\| + atol, \quad (14)$$

where $rtol$ and $atol$ are the relative and absolute tolerances, respectively. Previous work has demonstrated that $\mathcal{M}^{-1}$ is effective when based on linearized second-order (low-order) finite differences to allow efficient computation of the preconditioned system, while still obtaining fast algorithmic convergence to the original system of high-order accuracy. In addition, such low-order approximation significantly reduces the computational work of computing the residuals

and relaxations during the preconditioning phase.

4 Spatial Domain and Data Distribution

It is often advantageous to decompose a boundary value problem into smaller subdomains. For one, the computational work can be distributed and solved in parallel to achieve better overall performance. In addition, memory distribution lowers the memory requirements per compute node and allows for larger global problems. However, communication between compute nodes requires frequent message passing, leading to degraded performance, especially for smaller problem sizes.

For the free surface water wave model a data decomposition technique with small domain overlaps is proposed. The computational domain is first decomposed into rectangular non-overlapping subdomains, then so-called *ghost* layers are added to account for grid points that are distributed across adjacent subdomains¹. The size of the ghost layers depends on the size of the finite difference stencil. An example of the decomposition of a simple 17×5 domain into two subdomains is illustrated in Figure 1. Ghost points are updated block-wise, indicated by the arrows, when information from adjacent domains are queried. Thus, subdomains concurrently solve the global boundary value problem by communicating with neighboring subdomains. This approach differs from domain decomposition methods where overlapping local boundary value problems are solved individually⁴¹. Our proposed approach requires all ghost layers to be updated prior to any global

operation, but most importantly it maintains the attractive convergence rate of the multigrid preconditioned defect correction method.

Figure 1. A decomposition of a grid of global size 17×5 into two subdomains with two layers of ghost points. $\bullet$ and $\blacksquare$ represent internal grid points and ghost points respectively. $\circ$ indicate internal points that may be shared between grids to ensure an uneven dimension required by the geometric multigrid solver.

The vertical resolution for any free surface model is significantly lower than the horizontal resolution. Ten vertical grid points, for example, is often sufficient for hydrodynamics, while the number of horizontal grid points is dictated by the size of the physical domain and the wavelengths. The global domain is therefore decomposed using a two-dimensional horizontal topology. For practical applications, the necessary spatial resolution is typically less than 10 nodes in the vertical direction leading to sufficient accuracy in the dispersion relation for wave propagation. Thus, for applications, a large number of nodes can be used to represent the horizontal dimensions covering large marine areas, e.g. to simulate long wave propagation in coastal and harbor areas with varying bathymetry. Such large domains will have a favorable ratio between internal grid points and ghost points compared to smaller domains, causing less communication overhead relative to the computation volume.

4.1 Multi-GPU Data Distribution

A multigrid preconditioner $\mathcal{M}$, can be formed as a series of (sparse) matrix-vector operations. Any such matrix-vector operation $\mathcal{S}$, acting on a subdomain Ω_i may be formulated as

$$\mathcal{S}_i \begin{pmatrix} \mathbf{x}_i \\ \mathbf{g}_i \end{pmatrix} = \mathbf{b}_i, \quad \mathcal{S}_i \in \mathbb{R}^{n \times n}, \mathbf{b}_i \in \mathbb{R}^{n \times 1}, \quad (15)$$

where n is number of degrees of freedom in the subdomain, requiring $n \leq N$. The vectors $\mathbf{x}_i$ and $\mathbf{g}_i$ contain the internal points and the ghost points of subdomain i , respectively. Computing the matrix vector product using $\mathcal{S}_i$ is a local operation, which can be performed by the process (GPU) associated with subdomain Ω_i . To facilitate this, all ghost layers are updated prior to applying $\mathcal{S}_i$ by transferring boundary layer data between GPUs using MPI. The communication overhead due to the exchange of ghost values is related to size of the ghost layers $\mathbf{g}_i$ and is investigated in the following sections. When updating the internal boundaries between neighboring GPUs, memory is first copied from the GPU to the CPU before transferred via MPI. To reduce transfer overhead, this is performed first for the east/west boundaries, to allow asynchronous MPI communication, while the north/south boundaries are copied to the CPU. Figure 2 depicts some typical scenarios. The single workstation offers low memory transfer overhead as no network communication is required. Yet, the number of GPUs on a single workstation is limited due to the number of PCIe sockets (often one or two). In contrast, on a cluster or supercomputer a

distributed memory model is provided, allowing a flexible number of nodes for simultaneous computations, at the expense of lower network bandwidth.

Figure 2. Two common compute systems: A single workstation consisting of a single CPU and two GPUs; and a compute cluster consisting of multiple nodes connected in a high-speed network with a CPU and a GPU on each compute node.

5 Multi-GPU Performance Study

The numerical method has been verified and compared to both analytic and experimental data in previous works^{12,13}. One advantage of the multi-GPU decomposition method presented here is that it preserves the attractive algorithmic efficiency of a single-block multigrid method, which can be confirmed by verifying that the residual norms for both the single- and multi-GPU implementations agree within

machine precision. In the following, Ω_i , represents the i^{th} subdomain, with $i = 1, \dots, P$, where P is the number of subdomains. The number of subdomains is not restricted to match the number of GPUs, however no performance gains are expected compared a one GPU per subdomain. In the following sections the number of subdomains and GPUs are therefore always equal.

The following discussion focuses on the numerical efficiency and convergence of the multi-GPU implementation for solving the Laplace problem (12) for varying problem sizes and increasing number of GPUs. A detailed performance analysis of an optimized single-GPU version can be found in¹². A desktop computer and three compute clusters have been used to study the multi-GPU performance. Machine details are summarized in Table 1.

5.1 Data Distribution Performance

The computational bottleneck is solving the Laplace problem. Introducing multiple subdomains and thereby multiple GPUs, influences the performance of each individual subroutine of the multigrid preconditioned defect correction algorithm. As a measure of performance the average timings based on 100 defect correction iterations are recorded for an increasing number of unknowns. One multigrid V-cycle with Red-Black Z-line relaxation is used for preconditioning together with 6^{th} -order accurate finite differences for the spatial discretization. The number of coarsening levels in a V-cycle is denoted by K . The timings in Table 2 uses $K = K_{\max}$ for each run, meaning that each V-cycle continues until a $5 \times 5 \times 3$ grid

Machine	Desktop	Oscar	Stampede	Titan
Nodes (used)	1	16	32	8,192
CPU	Intel Xeon E5620	Intel Xeon 5540	Intel Xeon E5	AMD Opteron 6274
CPU frequency	2.40 GHz	2.53 GHz	2.4 GHz	2.2 GHz
CPU memory	12 GB	24 GB	32 GB	32 GB
GPU	4 x GeForce GTX 590	2 x Tesla M2050	Tesla K20m	Tesla K20X
CUDA cores	512	448	2,496	2,688
GPU Memory	1.5 GB	3 GB	5 GB	6 GB

Table 1. Hardware configurations for the desktop computer and three large compute clusters used in the tests. CUDA cores and memory are per GPU.

size per subdomain. A semi-coarsening strategy that best preserves the grid isotropy is chosen, implying that the grid is restricted only in the horizontal dimensions until the grid spacing is of similar size as the vertical grid spacing. Hereafter all dimensions are restricted. At each multigrid level two pre- and post-relaxations are used along with four relaxations at the coarsest level. The relative and absolute timings for each subroutine and for an increasing number of subdomains, P , are reported in Table 2. *Residual (high)* in the first column refers to the 6th-order residual evaluation in the defect correction loop, while *residual (low)* refers to the second-order linear residual evaluation in the preconditioning phase.

Note from the numbers in Table 2 that increasing the number of subdomains, and thereby the number of GPUs improves the overall computational time only for problems larger than $513 \times 513 \times 9$. We believe this is acceptable, as it is often difficult for multiple GPUs to efficiently solve problem sizes that fits within the memory of a single GPU. Since each GPU is massively parallel, using multiple GPUs for a small problem introduces costly overhead, which the current implementation has not been able to overcome. Strong scaling (in terms of number of GPUs) for problems of moderate

Figure 3. Performance breakdown of each subroutine as the number of GPUs increases. Timings are for one preconditioned defect correction iteration for a problem size of $1025 \times 1025 \times 9$, on *Oscar* cf. Table 2.

sizes should therefore not be expected to be good in general.

Based on the relative timings in Figure 3 we see that relaxation becomes increasingly dominant as the number of subdomains (thus GPUs) increases. Communication between GPUs occurs at every relaxation step, which happens multiple times at every multigrid level. For the coarse grid levels the surface to volume ratio is larger and hence communication overhead becomes a dominating performance bottleneck. It is therefore particularly desirable to decrease the number of multigrid levels

Subroutine	P	$129 \times 129 \times 9$		$257 \times 257 \times 9$		$513 \times 513 \times 9$		$1025 \times 1025 \times 9$	
		Percent	Time	Percent	Time	Percent	Time	Percent	Time
Residual (high)	1	15.2	0.0010	25.9	0.0036	34.2	0.0133	39.1	0.0529
Residual (low)	1	20.9	0.0014	23.5	0.0033	23.6	0.0092	23.0	0.0311
Relaxation	1	37.3	0.0025	32.5	0.0045	28.3	0.0110	27.4	0.0371
Restriction	1	11.9	0.0008	10.3	0.0014	7.0	0.0027	4.9	0.0067
Prolongation	1	12.1	0.0008	5.7	0.0008	5.0	0.0019	3.9	0.0053
Other	1	2.6	0.0002	2.1	0.0003	2.0	0.0007	1.7	0.0024
Total	1	100.0	0.0068	100.0	0.0139	100.0	0.0389	100.0	0.1355
Residual (high)	2	6.3	0.0009	10.0	0.0021	17.0	0.0071	27.5	0.0271
Residual (low)	2	18.7	0.0026	19.5	0.0042	22.8	0.0096	22.0	0.0216
Relaxation	2	50.7	0.0070	45.9	0.0099	38.1	0.0160	34.3	0.0337
Restriction	2	15.6	0.0022	14.2	0.0031	14.5	0.0061	9.4	0.0092
Prolongation	2	7.7	0.0011	9.3	0.0020	6.6	0.0028	5.5	0.0054
Other	2	1.1	0.0002	1.0	0.0002	1.1	0.0005	1.4	0.0013
Total	2	100.0	0.0139	100.0	0.0216	100.0	0.0420	100.0	0.0983
Residual (high)	4	3.7	0.0007	5.2	0.0014	9.9	0.0041	17.8	0.0139
Residual (low)	4	12.6	0.0023	14.1	0.0037	18.0	0.0074	18.9	0.0149
Relaxation	4	64.1	0.0119	59.1	0.0155	51.8	0.0212	42.0	0.0330
Restriction	4	10.8	0.0020	11.5	0.0030	13.2	0.0054	10.5	0.0083
Prolongation	4	7.8	0.0015	9.2	0.0024	6.3	0.0026	9.7	0.0076
Other	4	0.9	0.0002	0.8	0.0002	0.8	0.0003	1.0	0.0008
Total	4	100.0	0.0186	100.0	0.0262	100.0	0.0409	100.0	0.0784

Table 2. Relative and absolute timings for one defect correction iteration, divided into each subroutine, for an increasing number of unknowns and subdomains. ‘Other’ subroutines include an axpy and a 2-norm evaluation. Time in seconds on *Oscar*.

in order to also reduce the number of relaxation and consequently lower communication requirements. However, multigrid achieves its unique algorithmic scalability and grid independent convergence properties from the fact that it reduces error frequencies by relaxing on *all* grid levels. This is in general important for elliptic problems where all grid points are coupled.

In Section 5.3 a test example illustrates the effect of grid levels and horizontal to vertical aspect ratio on algorithmic and numerical efficiency.

5.2 Numerical Performance of Multigrid Restrictions

As a consequence of the previous observations, an examination of how the number of multigrid restrictions affects the numerical performance for both the single and multi-block solver is

performed. The absolute time *per* outer defect correction iteration is used here as a measure for performance. Thus, timings are independent of any physical properties of the free surface problem since algebraic convergence is not considered. Timings are measured and reported in Table 3 for a variation of levels K and number of subdomains P . The remainder of the solver properties are the same as in the previous example.

Two different speedup measures are reported in Table 3: δ_K is the speedup for K levels in comparison to $K_{\max}$, with a fixed number of subdomains P . Likewise, δ_P is the speedup for P subdomains in comparison to one domain, with a fixed number of restrictions K . We see that $\delta_{K=1} \geq \delta_{K=3} \geq \delta_{K=5} \geq \delta_{\max} \geq 1$, as expected. δ_P is a measure of strong scaling with respect to the number of GPUs (subdomains).

P	K	$257 \times 257 \times 9$			$513 \times 513 \times 9$			$1025 \times 1025 \times 9$		
		Time	δ_K	δ_P	Time	δ_K	δ_P	Time	δ_K	δ_P
1	1	0.0097	1.4	-	0.0322	1.2	-	0.1218	1.1	-
1	3	0.0119	1.2	-	0.0363	1.1	-	0.1315	1.0	-
1	5	0.0132	1.0	-	0.0376	1.0	-	0.1336	1.0	-
1	$K_{\max}$	0.0139	-	-	0.0389	-	-	0.1355	-	-
2	1	0.0108	2.0	0.9	0.0237	1.8	1.4	0.0735	1.3	1.7
2	3	0.0164	1.3	0.7	0.0335	1.3	1.1	0.0854	1.2	1.5
2	5	0.0216	1.0	0.6	0.0400	1.1	0.9	0.0929	1.1	1.4
2	$K_{\max}$	0.0216	-	0.6	0.0420	-	0.9	0.0983	-	1.4
4	1	0.0114	2.3	0.9	0.0185	2.2	1.7	0.0451	1.7	2.7
4	3	0.0196	1.3	0.6	0.0297	1.4	1.2	0.0590	1.3	2.2
4	5	0.0262	1.0	0.5	0.0377	1.1	1.0	0.0709	1.1	1.9
4	$K_{\max}$	0.0262	-	0.5	0.0409	-	1.0	0.0784	-	1.7
8	1	0.0084	2.5	1.1	0.0145	2.1	2.2	0.0288	1.9	4.2
8	3	0.0177	1.2	0.7	0.0230	1.3	1.6	0.0416	1.3	3.2
8	5	0.0211	1.0	0.6	0.0308	1.0	1.2	0.0509	1.1	2.6
8	$K_{\max}$	0.0210	-	0.7	0.0308	-	1.3	0.0543	-	2.5

Table 3. Absolute timings per defect correction iteration using a varying number of multigrid restrictions K , in the preconditioning phase. Notice that timings are for one iteration only and therefore does not include the total solve time. Speedups δ_K and δ_P are relative to $K_{\max}$ and to one subdomain/GPU, respectively, for the same problem size. Time in seconds on *Oscar*.

As before mentioned, multiple GPUs are feasible only for problems of reasonable sizes. Based on the reported numbers we conclude that there is good potential for improving the overall numerical performance, given that few restrictions (low K) is sufficient for rapid convergence. In the following a test is set up to demonstrate how this can be achieved.

5.3 Algorithmic Performance of Multigrid Restrictions

To unify the findings from the previous examples we now introduce and solve a specific nonlinear free surface problem. The purpose of this numerical experiment is to verify if imposing fewer restrictions and thus fewer relaxations to minimize communication, based on the findings in Table 2, can in fact lead to performance improvements as reported in Table 3. The final link that needs to be demonstrated,

is whether the algebraic convergence can be maintained using fewer multigrid restrictions.

A two-dimensional Gaussian distribution is used as initial condition to the free surface elevation within a numerical wave tank with no flux boundaries,

$$\eta(x, y, t) = \kappa e^{-\frac{x^2+y^2}{2\rho^2}}, \quad t = 0, \quad (16)$$

where $\rho = 0.15$ and $\kappa = 0.05$. The wavelength is approximately $\lambda = 1\text{m}$ and the wavenumber $k = 2\pi$. The distance from the seabed to still water level is $h = 1$ and therefore $kh = 2\pi$, which is intermediate depths. The physical size of the domain is extended to fit the initial aspect ratios that we wish to examine. The following number of grid points per wavelength is used $\mathcal{N}_w = 4, 8, 16, 32$, which results in aspect ratios of $\Delta\sigma/\Delta x = 4, 2, 1, 0.5$ at the finest grid level. Figure 4 illustrates how the nonlinear wave travels within the first seconds

with different wave resolutions. The initial wave rapidly propagates throughout the domain and creates multiple waves of various amplitudes.

$$\mathcal{N}_w = 32$$

$$\mathcal{N}_w = 16$$

$$\mathcal{N}_w = 8$$

(a) $T = 0s$ (b) $T = 2s$ (c) $T = 4s$

Figure 4. Nonlinear waves traveling in a closed basin with no flux boundaries, illustrated at three distinct time steps. Total horizontal grid dimensions are 129×129 .

At each time stage the Laplace problem (12) is solved to a relative and absolute tolerance of 10^{-4} and 10^{-5} , respectively, then the number of outer defect corrections are counted. The simulation is continued until the average number of iterations for each defect correction does not change within the first three digits. The results are collected in Table 4 where the aspect ratios and number of restrictions are also reported. The results are encouraging as they indeed confirm that the number of *useful* restrictions are related to the discretization of the physical domain. Load balancing between the GPU threads and multiple GPUs will therefore not be a critical issue, as the discretization of most practical applications allow favorable aspect ratios requiring only few restrictions.

Since ghost layers are always updated prior to any operation, convergence is independent of the number of subdomains. Thus the number of average iterations reported in Table 4 apply, regardless of the number of subdomains (GPUs) used for the same problem.

In addition to the above example, the spectral radius of the stationary iteration matrix for the two-level multigrid with Zebra-Line Gauss-Seidel smoothing is evaluated and plotted in Figure 5. A small spectral radii enables fast convergence, which is seen to be governed by the dimensionless variables for aspect ratio (vertical to horizontal) and wavelength (L) relative to ratio between sea bed slope (hx) and depth (h)^{11,14}. As the semi-coarsening strategy seeks to keep $\Delta\sigma/\Delta x$ close to 1 where the spectral radius is low, the figure also confirms that fast convergence can be achieved for wave propagation applications as intended.

Figure 5. Spectral radius distribution of the stationary iteration matrix of two-level multigrid with Zebra-Line Gauss-Seidel smoothing.

K	$\Delta\sigma/\Delta x$	$257 \times 257 \times 9$		$513 \times 513 \times 9$		$1025 \times 1025 \times 9$	
		Avg. Iter.	Avg. Solve	Avg. Iter.	Avg. Solve	Avg. Iter.	Avg. Solve
1	4	19.30	0.187	19.30	0.622	19.30	2.351
3	4	7.26	0.086	7.24	0.263	7.24	0.952
5	4	5.75	0.076	5.75	0.216	5.75	0.768
$K_{\max}$	4	5.75	0.080	5.74	0.223	5.74	0.778
1	2	10.95	0.106	10.95	0.353	10.95	1.334
3	2	6.54	0.078	6.54	0.237	6.54	0.860
5	2	6.44	0.090	6.44	0.242	6.44	0.860
$K_{\max}$	2	6.44	0.090	6.44	0.251	6.44	0.873
1	1	6.34	0.062	6.34	0.204	6.34	0.772
3	1	4.12	0.049	4.12	0.150	4.12	0.542
5	1	4.08	0.054	4.08	0.153	4.08	0.545
$K_{\max}$	1	4.08	0.057	4.08	0.159	4.08	0.553
1	0.5	4.73	0.046	4.73	0.152	4.73	0.576
3	0.5	4.12	0.049	4.12	0.150	4.12	0.542
5	0.5	4.12	0.054	4.12	0.155	4.12	0.550
$K_{\max}$	0.5	4.12	0.057	4.12	0.160	4.12	0.558

Table 4. Average number of defect correction iterations for the test setup using different numbers of multigrid restrictions and initial aspect ratios. The *Avg. Solve* columns indicate the total solve times on *Oscar* using one GPU, cf. the timings in Table 3.

6 Large Scale Performance Analysis

Given the findings in previous sections, the solver performance is now evaluated on three larger GPU-accelerated clusters. Time per defect correction iteration is again used as the measure of performance since it is independent on the physical properties of the wave model. The waves are resolved with grid points such that three levels of the multigrid preconditioner is sufficient, $K = 3$, following the results from the previous section.

First performance test is performed in the *Oscar* cluster at Brown University. *Oscar* is equipped with Intel Xeon E5630 processors with 2.53GHz and 32GB host memory per compute node. Each node has two Nvidia Tesla M2050 GPUs. Absolute performance timings are summarized in Figure 6a as a function of increasing problem size. The single block timings (Ω_1) evolve as expected, overhead

of launching kernels are evident only for the smallest problem sizes, while for larger problems the time per iteration scales linearly. For the multi-GPU timings a notable effect is observed when communication is dominating and when it is not. After approximately one million degrees of freedom, communication overhead becomes less significant and the use of multiple compute units starts to become beneficial. For the larger problem sizes, close to optimal linear performance is observed.

The second performance scale test is performed on the Stampede cluster at the University of Texas. Stampede is a Dell Linux cluster based on compute nodes equipped with two Intel Xeon E5 (Sandy Bridge) processors, 32GB of host memory, and one Nvidia Tesla K20m GPU. All nodes are connected with Mellanox FDR InfiniBand controllers. The Stampede cluster allows up to 32 GPU nodes to be occupied at once. Absolute performance timings are illustrated in Figure 6b. Notice that

a setup of 16 GPUs allows the solution of problems with more than one billion degrees of freedom in practical times. With an approximate time per iteration of $t_{it} = 0.6$ s at $N = 10^9$, it would be possible to compute a full time step in 3 to 6 seconds, assuming convergence in 5 to 10 iterations. With a time step size of $\Delta t = 0.05$ s, a one minute simulation can be computed in 1 to 2 hours. This is well within a time frame considered practical for engineering purposes. With 32 GPUs available, compute time would reduce to almost half.

A final extremely large-scale performance evaluation is performed on the *Titan* cluster at the Oak Ridge National Laboratory. *Titan* nodes are equipped with AMD Opteron 6274 CPUs and Nvidia Tesla K20X GPUs. They have 32GB host memory and are connected with Gemini interconnect. A scaling test up to 8,192 GPUs is performed and reported in Figure 6c. Timings are well aligned with the previous results, though the curves are less smooth. The implementation of the scalability test alternately doubles the number of grid points in the x - and y -direction. When increasing the problem in the x -direction, only coalesced memory at the ghost layers are introduced, whereas the y -direction adds un-coalesced ghost layers. *Titan* seems to be more sensible to such alternately coalesced/un-coalesced increases.

Weak and strong scaling results for all three compute clusters are depicted in Figure 7a and 7b, respectively. Figure 7a shows the efficiency for each of the three compute clusters as the ratio between the number of GPUs and problem size is constant. There is a performance penalty when introducing multiple GPUs, indicated by

the drop from one to multiple GPUs. Hereafter weak scaling remains almost constant and there is no significant penalty when using additional GPUs. Weak scaling is stable up to the 8,192 GPUs on *Titan*. The strong scaling figure shows the efficiency in terms of inverse cost for a given time consumption at a constant problem size of $N \approx 3.8 \cdot 10^7$. Ideally, increasing the number of GPUs would lead to a corresponding reduction in compute time, leading to constant curves in Figure 7b. However, this is not the case for the given problem size, where parallel overhead becomes critical.

7 Vertical cylinder wave run-up

Predicting water scattering or wave loads on a vertical cylinder is relevant for off shore pile installations such as wind turbines. In this demonstration the wave run-up around a single vertical cylinder in open water is considered using a boundary fitted domain in curvilinear coordinates. Analytic formulations³⁵ and experimental data are available for linear and nonlinear incident waves over even sea bed. For nonlinear water waves, experimental results are presented by Kriebel^{28,29}, where nonlinear diffraction effects are studied and determined to be of significant importance compared to linear theory. To demonstrate the application of a boundary fitted domain, a nonlinear experiment from²⁹ is reproduced. Long and regular incident waves are generated in the inlet zone with a wavelength of $L = 2.73$ m and wave height $H = 0.053$ m. A wave damping outlet zone is used to avoid wave reflections²⁰. The still water depth h is constant and rather shallow, resulting in dimensionless $kh = 1.036$ and $kH = 0.122$.

(a) *Oscar* (b) *Stampede* (c) *Titan*

Figure 6. Absolute performance timings for one defect correction iteration on *Oscar*, *Stampede* and *Titan*. *Oscar* and *Stampede* have been configured for single-precision floating-point arithmetic, whereas *Titan* is using double-precision floating-point.

(a) Weak scaling, with $N \approx 3.8 \cdot 10^7$ per GPU.

(b) Strong scaling, with constant $N \approx 3.8 \cdot 10^7$.

Figure 7. Weak and strong scaling on all three compute clusters. The figures show efficiency in terms of inverse cost versus number of GPUs and time per defect correction, respectively.

Diffraction occurs around the centered vertical cylinder of radius $a = 0.1625$, thus $ka = 0.374$. The nonlinear case uses a 6th-order finite difference approximation and a resolution of $1025 \times 129 \times 9$. The vertical cylinder setup is illustrated in Figure 8.

Measuring the maximum fully developed wave run-up around the cylinder provides the 360° wave height profile in Figure 9. 0° is the opposite side of the pile relative to the

Figure 8. Bottom mounted vertical cylinder wave run-up. Computational grid resolution of $1025 \times 129 \times 9$.

incoming waves. The numerical results are well in agreement with the experiments and they are consistent with previously presented numerical results using similar numerical approaches, e.g.⁹. Though the vertical cylinder case is primarily an illustrative example, the introduction of an efficient solver for large scale simulation, supporting generalised curvilinear coordinates, significantly increases the range of real engineering applications.

8 Conclusion and Future Work

For the Laplace problem of a fully nonlinear free surface model, we have presented a performance analysis of for distributed parallel computations on cluster platforms. The spatial decomposition technique uses stencil-sized ghost layers, preserving the algorithmic efficiency and numerical properties of the established existing solvers for the problem, cf. recent studies^{11,12}. The numerical model is implemented in the DTU GPUlab library developed at DTU Compute and uses Nvidia's

(a) Wave run-up in Cartesian coordinates.

(b) Wave run-up in polar coordinates.

Figure 9. Nonlinear wave run-up presented in both Cartesian and polar coordinates. $kH = 0.122$, $kh = 1.036$, $ka = 0.374$. Experimental data from Kriebel²⁹.

CUDA C programming model and is executed on a recent generation of programmable Nvidia GPUs. In performance tests, we demonstrate good and consistent weak scalability of up to 8,192 GPUs and on three different compute clusters. We have detailed how the model is capable of solving systems of equations with more than 100 billion degrees of freedom for the first time with the current scalability tests restricted by the amount of hardware available

at the time of writing. Also, we highlight that multi-GPU implementations can be a means for further acceleration of run-times in comparison with single GPU computations.

As a first-step, we have demonstrated 2D curvilinear transformation in the horizontal plane to allow treatment of fixed bottom-mounted structures extending vertically throughout the depth of the fluid. This opens a large class of coastal geometries to a fully nonlinear analysis. An example of wave run-up on a mono-pile structure in open water has been demonstrated with good agreement to experimental results. This approach provides the basis for addressing challenges related to problems of engineering interests: (i) real-time computations, (ii) tsunami propagation over large-scales, (iii) marine offshore hydrodynamics, (iv) renewable energy applications, etc. We restricted our focus to deal with multi-GPU scalability and leave more realistic applications as a part of future work.

9 Acknowledgment

The authors would like to thank prof. Jan Hesthaven, senior researcher Andy Terrel, and CUDA fellow Alan Gray for their support in enabling the massive scalability tests. This work was funded by the Danish Council for Independent Research - Technology and Production Sciences (FTP) in Denmark. Nvidia Corporation supported the project with kind Hardware donations through the NVIDIA Academic Partnership Program.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is

supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Part of this research was conducted using computational resources and services at the Center for Computation and Visualization, Brown University.

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper.

References

1. Acklam E, Langtangen HP. Parallelization of explicit finite difference schemes via domain decomposition. Report for Oslo Scientific Computing Archive. Report no. 1998-2, February 1998.
2. Asanovic K, Bodik R, Catanzaro BC, et al. The landscape of parallel computing research: A view from berkeley. Tech. Rep. UCB/EECS-2006-183.
3. Bell N, Hoberock J. Thrust: A productivity-oriented library for cuda. GPU Computing Gems, Jade Edition, Edited by Wen-mei W. Hwu 2, 2011; 359–371.
4. Bell N, Dalton S, Olson L. Exposing fine-grained parallelism in algebraic multigrid methods. SIAM J Sci Comput 2012; 34: 123-152.
5. Bigoni D, Engsig-Karup AP, Eskilsson C. Efficient uncertainty quantification of a fully nonlinear and dispersive water wave model with random inputs. J Eng Math 2016: 1–27.
6. Bingham HB, Zhang H. On the accuracy of finite-difference solutions for nonlinear water waves. J Eng Math 2007; 58: 211–228.
7. Cai X, Pedersen GK, Langtangen HP. A parallel multi-subdomain strategy for solving Boussinesq water wave equations. J Adv Water Resour 2005; 28: 215–233.
8. Dias F, Bridges TJ. The numerical computation of freely propagating time-dependent irrotational water waves. Fluid Dyn Res 2006; 38,12: 803–830.
9. Ducrozet G, Bingham HB, Engsig-Karup AP, Ferrant P. High-order finite difference solution for 3D nonlinear wave-structure interaction. J Hydrodyn Ser B 2010; 22(5): 225–230.
10. Engsig-Karup AP. Unstructured Nodal DG-FEM Solution of High-Order Boussinesq-type Equations. 2006, Technical University of Denmark.
11. Engsig-Karup AP, Bingham HB, Lindberg O. An efficient flexible-order model for 3d nonlinear water waves. J Comput Phys 2008; 228: 2100–2118.
12. Engsig-Karup AP, Madsen MG, Glimberg SL. A massively parallel GPU-accelerated model for analysis of fully nonlinear free surface waves. Int J Numer Methods Fluids 2011; 70 (1): 20–36.
13. Engsig-Karup AP, Glimberg SL, Nielsen AS, Lindberg O. Fast hydrodynamics on heterogeneous many-core hardware. In: Designing Scientific Applications on GPUs. Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series. 2013.
14. Engsig-Karup, AP. Analysis of efficient preconditioned defect correction methods for nonlinear water waves. International Journal for Numerical Methods in Fluids 2014; 74 (10): 749–773.
15. Esler K, Mukundakrishnan K, Natoli V, et al. Realizing the Potential of GPUs for Reservoir Simulation. In: ECMOR XIV - 14th European

- Conference on the Mathematics of Oil Recovery. Sicily, Italy, 8-11 September 2014.
16. Fan Z, Qiu F, Kaufman A, et al. GPU cluster for high performance computing. In SC 04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing. November 2004.
 17. Fang K, Zou Z, Liu Z, et al. Boussinesq modelling of nearshore waves under body fitted coordinate. *J Hydrodyn. Ser B* 2012; 24(2): 235–243.
 18. Glimberg SL, Engsig-Karup AP, Madsen MG. A fast GPU-accelerated mixed-precision strategy for fully nonlinear water wave computations. In: *Numerical Mathematics and Advanced Applications, Proceedings of ENUMATH 2011, 9th European Conference on Numerical Mathematics and Advanced Applications*, Leicester, September 2011.
 19. Glimberg SL, Engsig-Karup AP, Nielsen AS, et al. Development of software components for heterogeneous many-core architectures. In: *Designing Scientific Applications on GPUs*. Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series, 2013.
 20. Glimberg SL. *Designing Scientific Software for Heterogenous Computing - With application to large-scale water wave simulations*. PhD thesis, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark, 2013.
 21. Goddeke D. *Fast and accurate finite-element multigrid solvers for PDE simulations on GPU clusters*. Logos Verlag Berlin GmbH, 2011.
 22. Goddeke D, Strzodka R, Mohd-Yusof J, et al. Exploring weak scalability for FEM calculations on a GPU-enhanced cluster. *Parallel Comput* 2007; 33(10): 685–699.
 23. Grilli ST, Vogelmann S, Watts P. Development of a 3D numerical wave tank for modeling tsunami generation by underwater landslides. *Eng. Anal. Bound. Elem* 2002; 26(4): 301–313.
 24. Gropp W, Lusk E, Skjellum A. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. 2nd edition, 1999, MIT Press, Cambridge, MA.
 25. Hino T, Carrica P, Broglia R, et al. Specialist Committee on CFD in Marine Hydrodynamics. *Proceedings of the 27th International Towing Tank Conference (ITTC)*, 2014, pp. 522–567.
 26. Holewinski J, Pouchet LN, Sadayappan P. High-performance Code Generation for Stencil Computations on GPU Architectures. *Proceedings of the 26th ACM International Conference on Supercomputing, ICS '12*, 2012, pp. 311–320.
 27. Kazolea M, Ricchiuto M. Wave breaking for Boussinesq-type models using a turbulence kinetic energy model. *Research Report RR-8781*, Inria Bordeaux Sud-Ouest, 2016.
 28. Kriebel DL. Nonlinear wave interaction with a vertical circular cylinder. Part I: Diffraction theory. *Ocean Eng* 1990; 17: 345–377
 29. Kriebel DL. Nonlinear wave interaction with a vertical circular cylinder. Part II: Wave run-up. *Ocean Eng* 1992; 19: 75–99
 30. Li B, Fleming CA. A three dimensional multigrid model for fully nonlinear water waves. *Coastal Eng* 1997; 30: 235–258.
 31. Li B, Fleming A. Three dimensional model of Navier-Stokes equations for water waves. *J Waterway, Port, Coastal, and Ocean Eng* 2001; 127: 16–25.
 32. Li YS, Zhan JM. Boussinesq-Type Model with Boundary-Fitted Coordinate System. *J Waterway, Port, Coastal, and Ocean Eng* 2001;

- 127 (3): 152–160.
33. Lin P. Numerical Modeling of Water Waves. Taylor & Francis, 2008.
 34. Lindberg O, Glimberg SL, Bingham HB, et al. Towards real time simulation of ship-ship interaction - part ii: double body flow linearization and GPU implementation. In: Proceedings of The 28th International Workshop on Water Waves and Floating Bodies (IWWF) 2012.
 35. MacCamy RC, Fuchs RA. Wave Forces on Piles: A Diffraction Theory. Technical memorandum. U.S. Beach Erosion Board, 1954.
 36. Micikevicius P. 3D finite difference computation on GPUs using CUDA. In: GPGPU-2: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, 2009. ACM, New York, NY, USA, pp. 79–84.
 37. Naumov M, Arsaev M, Castonguay P, et al. AmgX: A Library for GPU Accelerated Algebraic Multigrid and Preconditioned Iterative Methods. SIAM J. Sci. Comput 2015; 37(5): 602–626.
 38. Nimmala SB, Solomon C, Grilli ST. An Efficient Three-Dimensional FNPF Numerical Wave Tank for Large-Scale Wave Basin Experiment Simulation. J. Offshore Mech. Arct. Eng, 2013; 135(2).
 39. Shi F, Sun W. A variable boundary model of storm surge flooding in generalized curvilinear grids. Int. J. Numer. Methods Fluids 1995; 21(8): 641–651.
 40. Shi F, Dalrymple RA, Kirby JT, et al. A fully nonlinear Boussinesq model in generalized curvilinear coordinates. Coastal Eng 2001; 42: 337–358.
 41. Smith BF, Bjørstad PE, Gropp WD. Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Press, New York, 1996.
 42. Strzodka R, Cohen J, Posey S. GPU-Accelerated Algebraic Multigrid for Applied CFD. 25th International Conference on Parallel Computational Fluid Dynamics, 2013; 63: 381–387.
 43. Svendsen IA, Jonsson IG. Hydrodynamics of coastal regions. 1976, Vol. 3. Den private ingeniørfond, Technical University of Denmark.
 44. Trottenberg U, Oosterlee CW, Schuller A. Multigrid, 1st Edition, 2000, Academic Press.
 45. Verbrugge T, Troch P, Kortenhaus A, et al. Development of a numerical modelling tool for combined near field and far field wave transformations using a coupling of potential flow solvers. In: Proceedings of the 2nd International Conference on Renewable Energies Offshore (RENEW), 2016.
 46. Yeung RW. Numerical methods in free-surface flows. In: Annual review of fluid mechanics, 1982; 14: 395–442.
 47. Zhang H, Zhu L, You Y. A numerical model for wave propagation in curvilinear coordinates. Coastal Eng 2005; 52: 513–533.