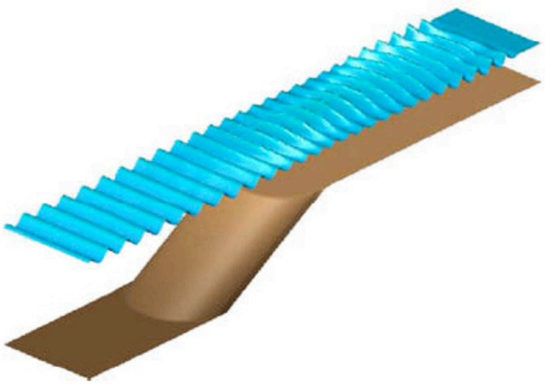
# Very fast simulation of nonlinear water waves in very large numerical wave tanks on affordable graphics cards

By Allan P. Engsig-Karup,
Morten Gorm Madsen and Stefan L. Glimberg
DTU Informatics

Workshop on "GPU computing today and tomorrow"

Kgs. Lyngby, DTU, Aug 18, 2011.

# Requirements

**General requirements**

v Ability to estimate or predict wave kinematics and wave loads on structures
v Estimation of influence of sea floor on wave transformations
v Accurate representation of wave propagation over long distances and times
v Numerical basis for a tool should be generally applicable and reliable
(v) Possibility to simulate waves in realistic structural settings

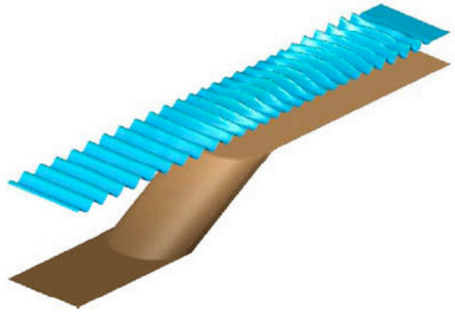
**Current research directions**

v Development of an efficient and scalable parallel algorithm for the numerical tool
v Utilize many-core hardware to maximize performance for fast analysis and large problem sizes
- New robust numerical engineering tools for wave-structure interaction
 (floating wave-energy devices, windmill foundations, etc.)

# Unified model for unsteady potential flow

**Kinematic and dynamic free surface conditions**

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



**Laplace problem**

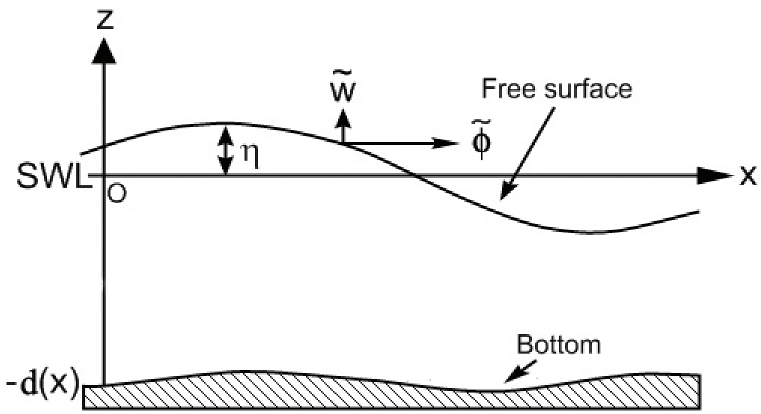
$$
\begin{aligned}
&\phi = \tilde{\phi}, && z = \eta, \\
&\nabla^2 \phi + \partial_{zz}\phi = 0, && -h \le z < \eta, \\
&\partial_z \phi + \nabla h \cdot \nabla \phi = 0, && z = -h.
\end{aligned}
$$

**Vertical free surface velocity**

$$\tilde{w} = \partial_z \phi, \quad z = \eta$$

**Lateral boundary conditions (net-flux conditions)**

$$\mathbf{n} \cdot \nabla \phi + n_z \partial_z \phi = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega$$




*"Modelling basis is far too heavy"*, Cai Et al. (2006)

# OceanWave3D
# - a wave model for coastal engineering

# Case study

- Can we use the new GPU technology and programming models to leverage performance over existing CPU applications?
- **Proof-of-concept** case study
  - Develop new massively parallel algorithm for an engineering application that can utilize GPU architectures
  - Enable fast engineering analysis of fully nonlinear waves at large scales by implementation on affordable commodity hardware
  - Find out how fast can we do robust (real-time?) computer-based analysis and experiments with OceanWave3D model

# OceanWave3D

Courtesy of The Hydraulics and Maritime Research Centre (HMRC) at University College Cork

**Numerical method**

An accurate and robust arbitrary-order finite difference method.

**Computational bottleneck problem (Laplace)**

Efficient and scalable iterative solution of large sparse linear system every time step.

=> Entire algorithm is explicit.

=> Algorithmic efficiency for both linear and nonlinear simulations established.

Can we do better?

# Hardware characteristics

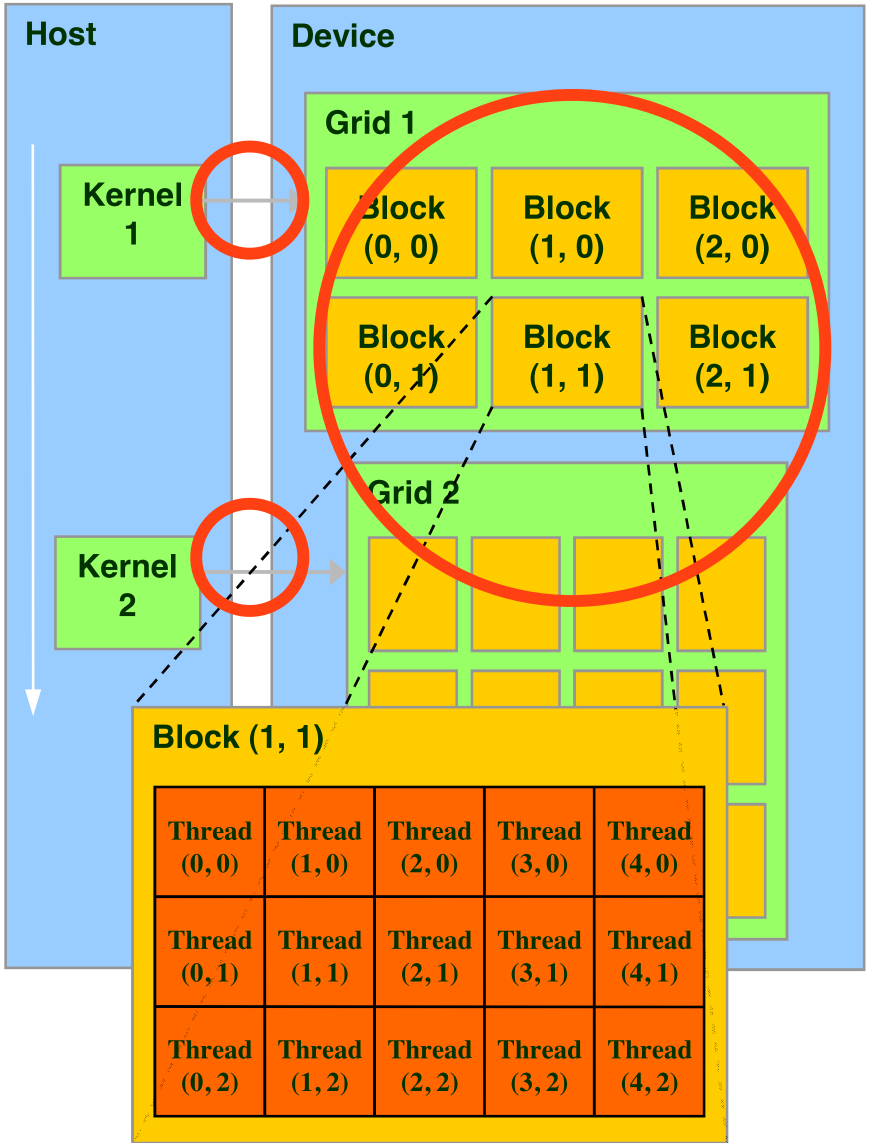
**Host (CPU)**
Should be capable of acting as a task manager for any hardware accelerator device (GPU) connected.

**Device (GPU)**
A device GPU acts as a co-processor to the host CPU and is often used for compute-intensive tasks.

Potential performance bottleneck: **data-transfer**

- **PCIe x16 Gen 2 link bandwidth ~5 GB/s**

- **GPU on-chip bandwidth <192 GB/s**



# Iterative methods

## Summary of development and analysis of past work on the efficient solution of the model equations

| Contributions | 2D | 3D | Iterative method | Accuracy | Storage |
|---|---|---|---|---|---|
| Li & Fleming (1997) | ✓ | ✓ | Multigrid (MG) | 2nd | Low |
| Bingham & Zhang (2006) | ✓ | | GMRES+LU | Flexible | High |
| Engsig-Karup, Bingham & Lindberg (2008) | ✓ | ✓ | GMRES+MG | Flexible | High |
| Engsig-Karup (2010) | ✓ | ✓ | Defect Correction + LU/MG | Flexible | Low |

- Multigrid method is O(n), robust, fast convergence for 2nd order, is memory-limited
- The Standard GMRES method has increasing workload per iteration, not memory-limited
- DC method, is memory-limited and requires less global synchronization
- DC and GMRES methods, robust and require efficient preconditioning to be fast

# Defect Correction method

$$\mathbf{\Phi}^{[k+1]} = \mathbf{\Phi}^{[k]} + \delta^{[k]}, \quad \delta^{[k]} = \mathcal{M}^{-1}(\mathbf{b} - \mathcal{A}\mathbf{\Phi}^{[k]}), \quad k = 0, 1, 2, \ldots$$

Good initial guess
(last solve)

Sparse matrix-vector
product (matrix-free and minimal storage)

**Algorithm 1**: Defect Correction Method for solution of $\mathcal{A}x = b$

```
Choose x^[0]                                   /* initial guess */
k = 0
repeat
    r^[k] = b - A x^[k]                        /* high-order defect */
    Solve M δ^[k] = -r^[k]                     /* preconditioning problem */
    x^[k+1] = x^[k] - δ^[k]                    /* defect correction */
    k = k + 1
until no convergence and k < maximum iterations ;
```

Preconditioning problem
- solve cheaply and efficiently

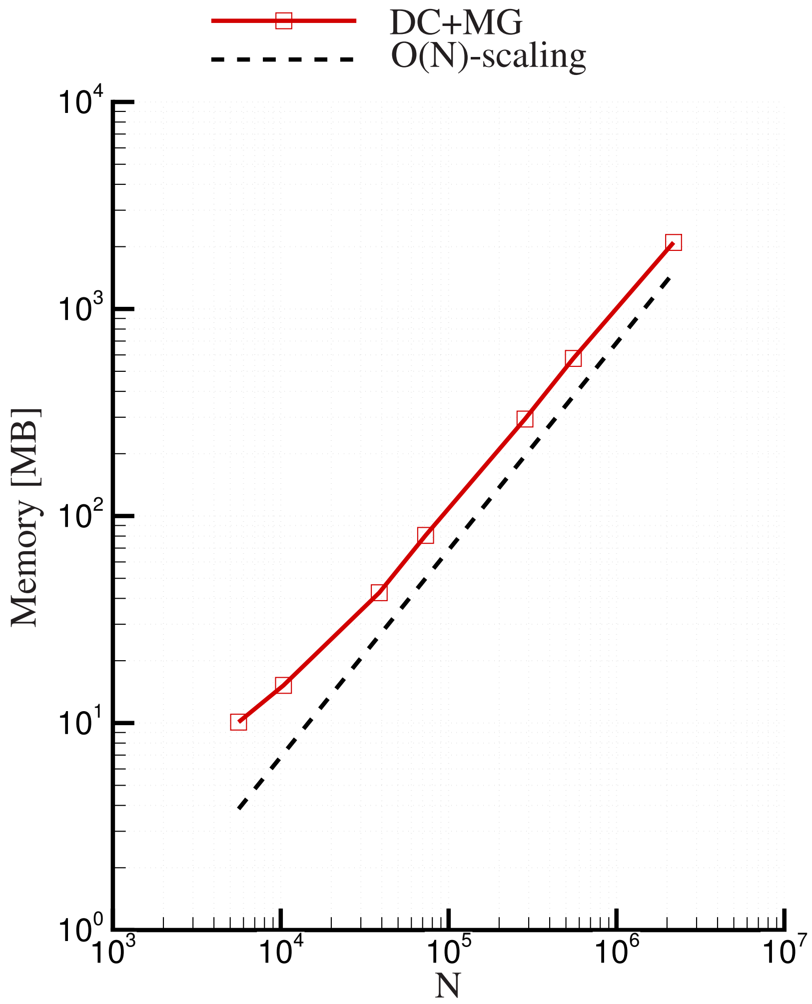
Error estimator (gather step)

Stop criteria (efficiency)

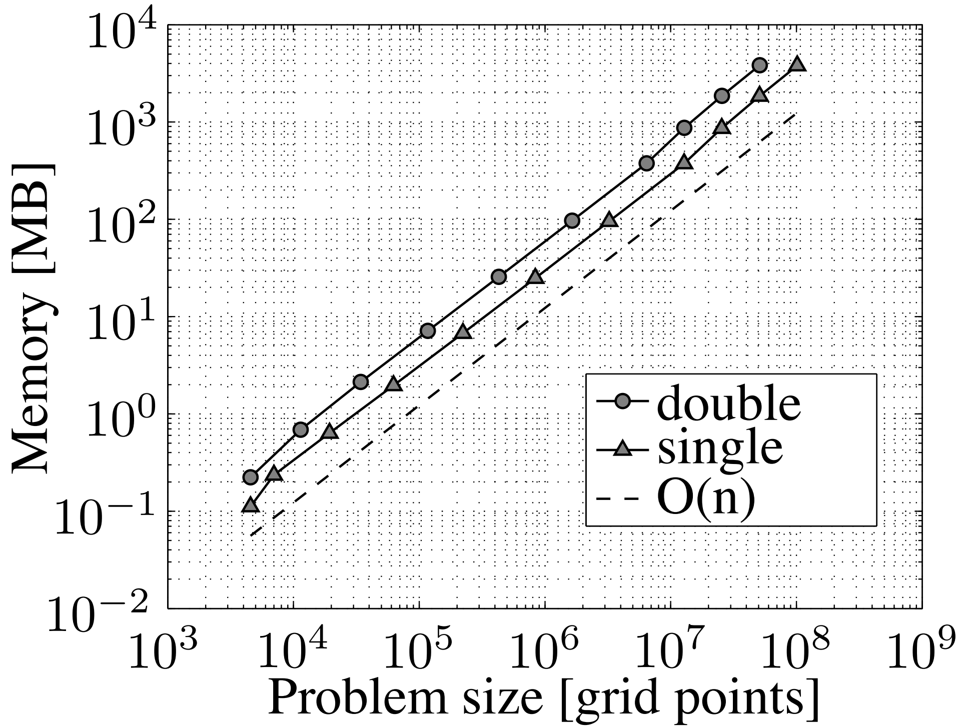
- Algorithm suitable for mixed precision calculations
- Two-step recurrence basis for minimal memory footprint

# OceanWave3D Scalability

## CPU



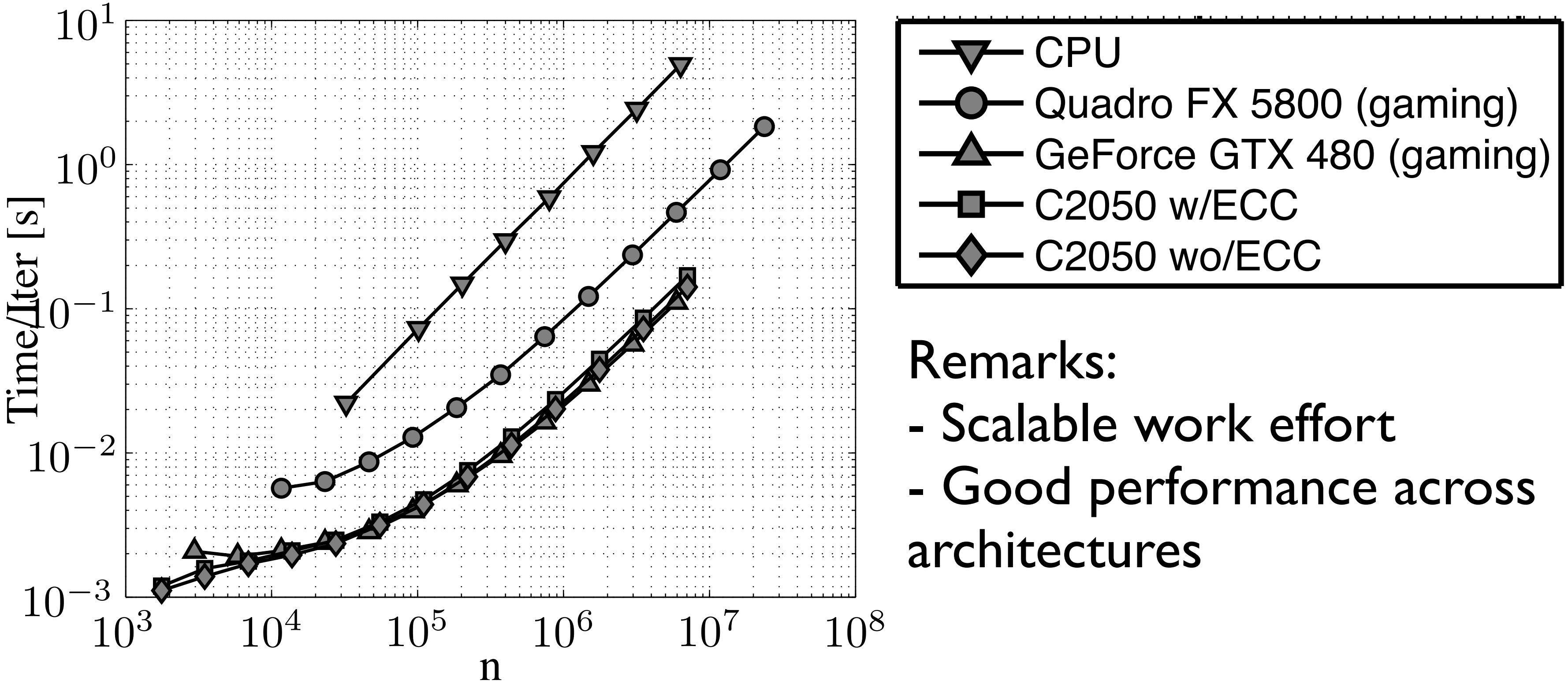

## GPU




**Implementation choices matters**

- Mapping to GPU resulted in a reduction of more than x10 in memory footprint
- from 5.000.000 up to 50.000.000 degrees of freedom in Laplace problem (double precision)
- Less than 4GB RAM required

# Throughput performance



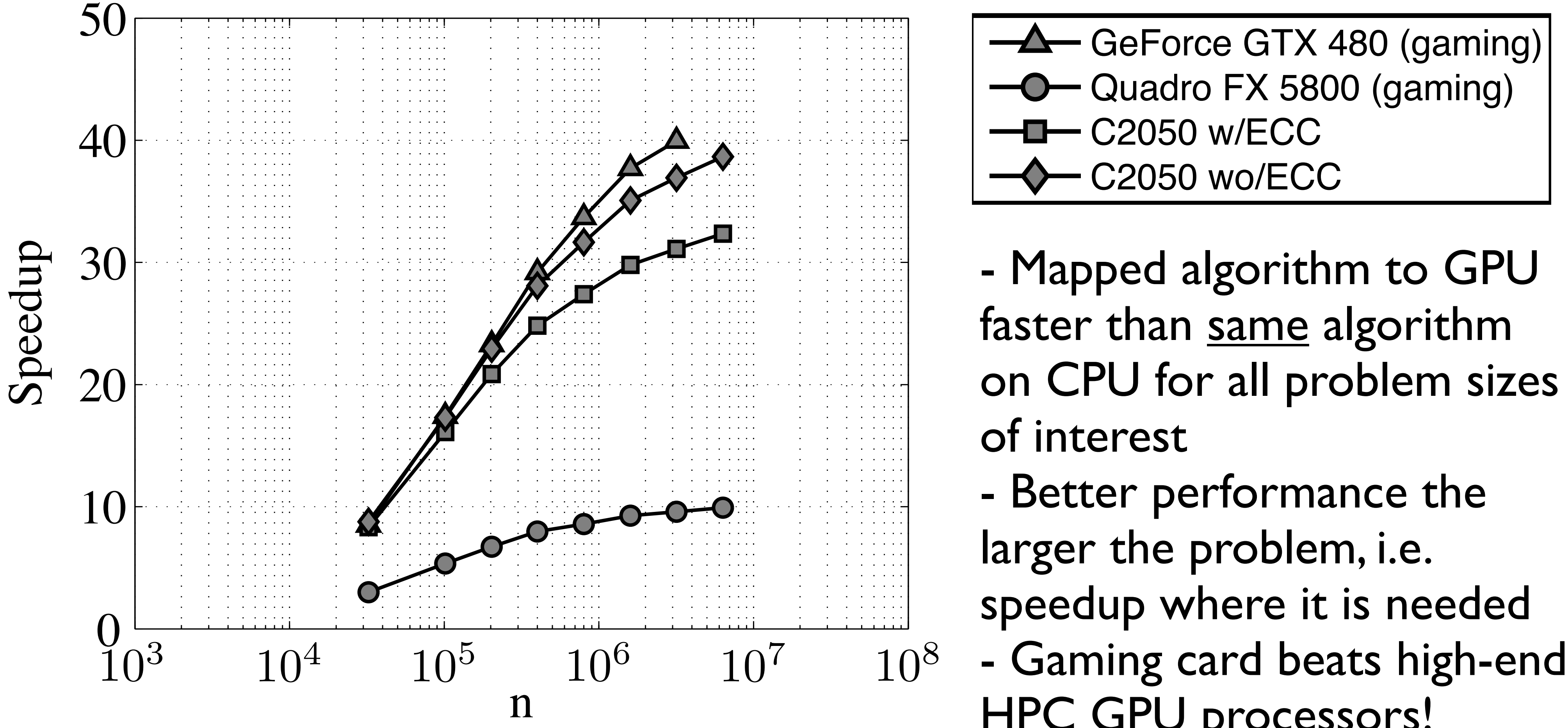

(a) Absolute timings.

Remarks:
- Scalable work effort
- Good performance across architectures

- Throughput performance curves a means for evaluating and confirming performance and capturing current state-of-the-art in practice.
- Without these difficult to make fair comparison between different models

Engsig-Karup, Allan; Madsen, Morten; Glimberg, Stefan. A massively parallel GPU-accelerated model for analysis of fully nonlinear free surface waves. Journal: International Journal for Numerical Methods in Fluids, 2011.

# Relative speedup




(b) Speedup relative to CPU (single thread) code in double precision arithmetic.

- Mapped algorithm to GPU faster than <u>same</u> algorithm on CPU for all problem sizes of interest
- Better performance the larger the problem, i.e. speedup where it is needed
- Gaming card beats high-end HPC GPU processors!

Engsig-Karup, Allan; Madsen, Morten; Glimberg, Stefan. A massively parallel GPU-accelerated model for analysis of fully nonlinear free surface waves. Journal: International Journal for Numerical Methods in Fluids, 2011.

# Precision requirements?

Most applications today use double precision maths to minimize accumulation of round-off errors.

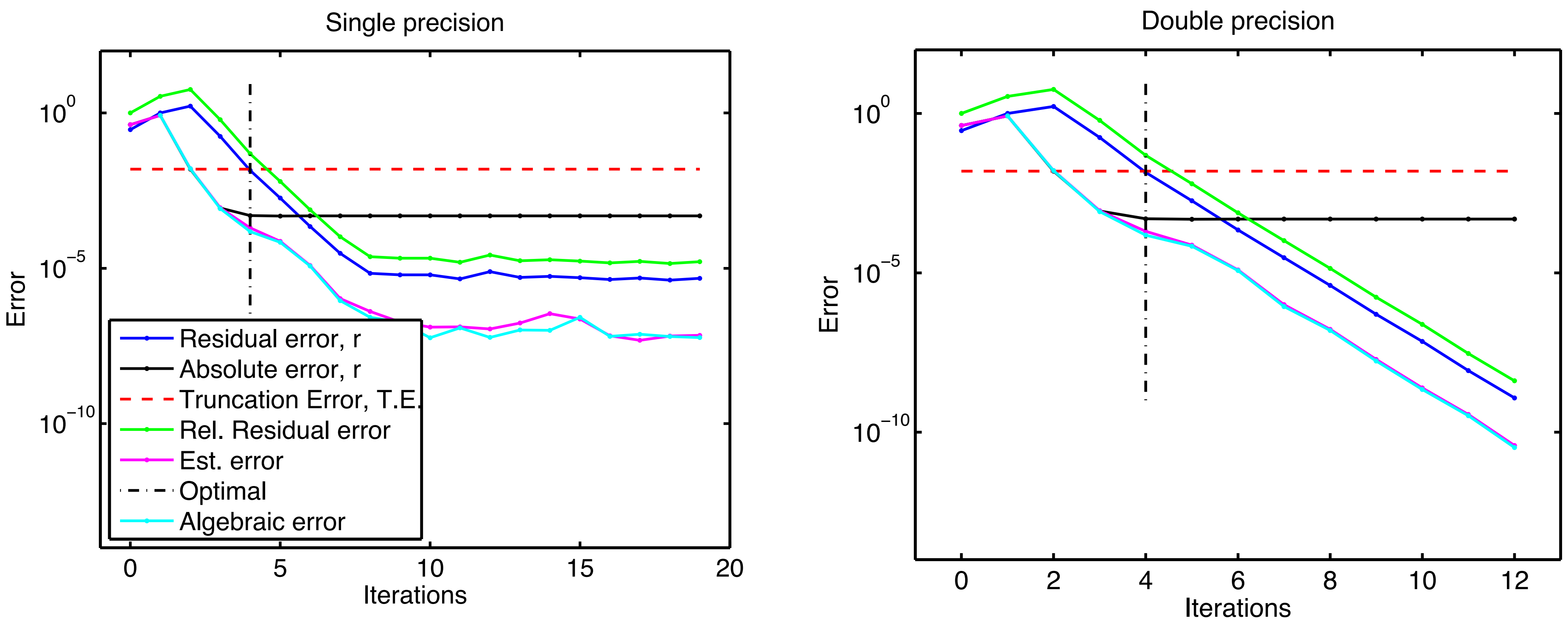
- How much precision do we really need in computations?
- Can we trade precision for speed?
- Is it feasible for practical computations?

If single precision math can be used in parts of code, it is possible to

- Half the size of data-transfers
- For some architectures single precision operations can be processed at more than twice the speed, e.g. GPUs.

Free lunch: x2?

# Single precision vs. double precision

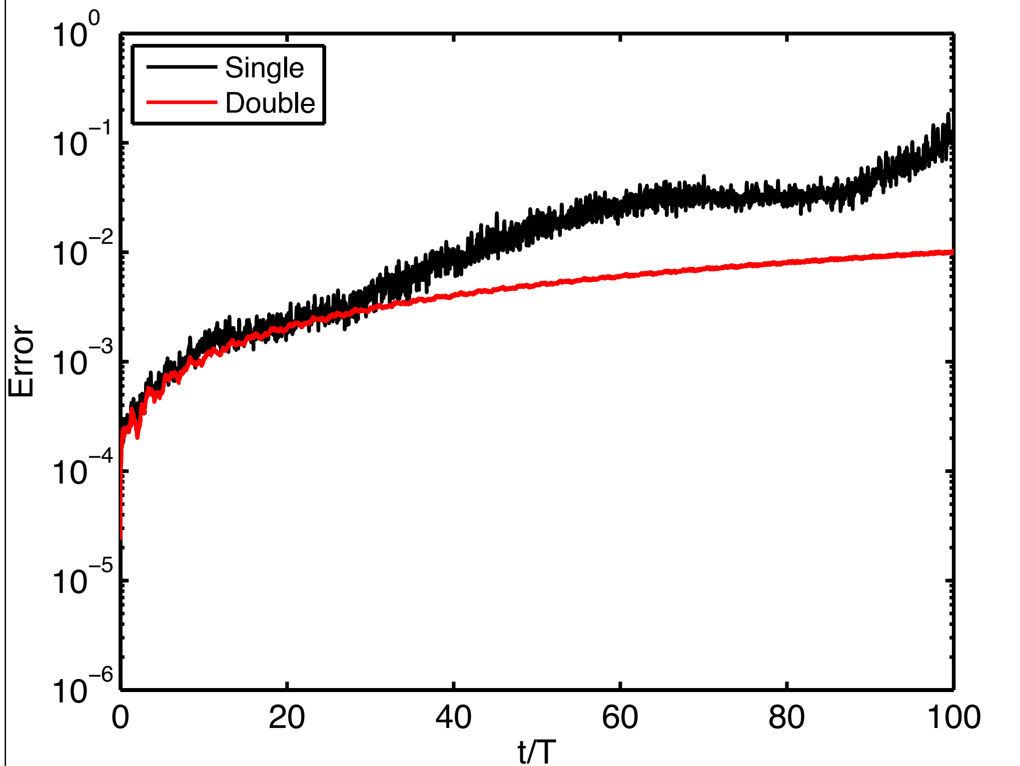



Deep water, SF waves,
kh=6.14, H/L=90% , Nx=15, Nz=9, 6th order stencils

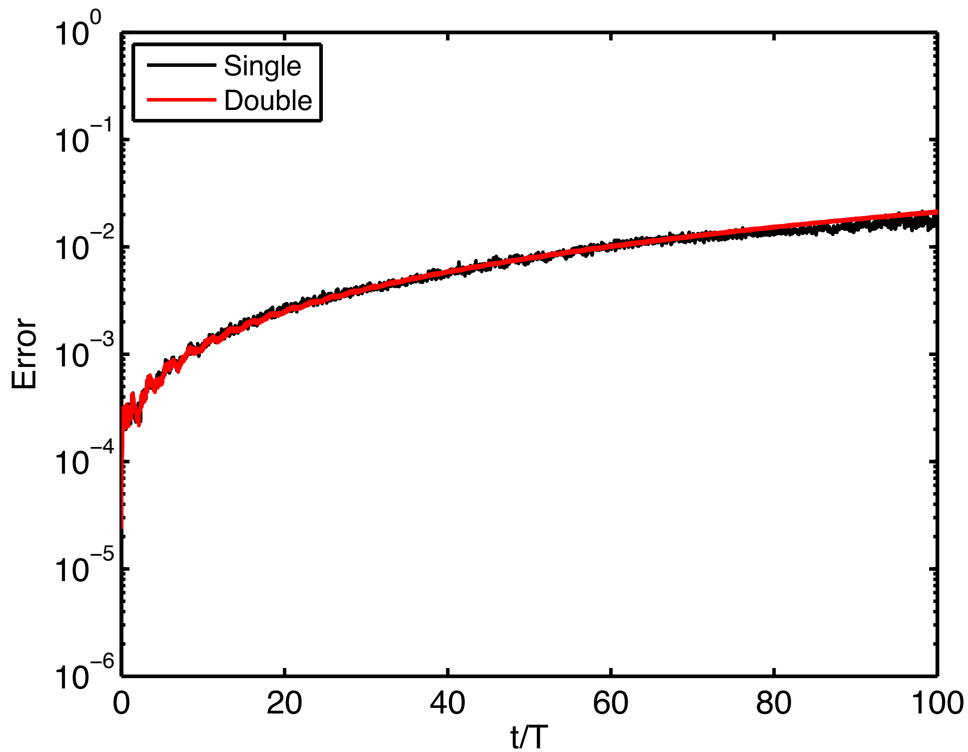
- Overall accuracy can be maintained in solution of Laplace problem

# Single precision vs. double precision

Without filtering




With filtering




**Parameters:** Intermediate water, SF waves, Direct solution
kh=1, H/L=90% , Nx=15, Nz=9, 6th order stencils
SG(6,10) filter strategy every 10 time step

- Errors tends to accumulate faster in single precision without stabilization
- Control at the expense of a mild inexpensive filtering strategy

# OceanWave3D model

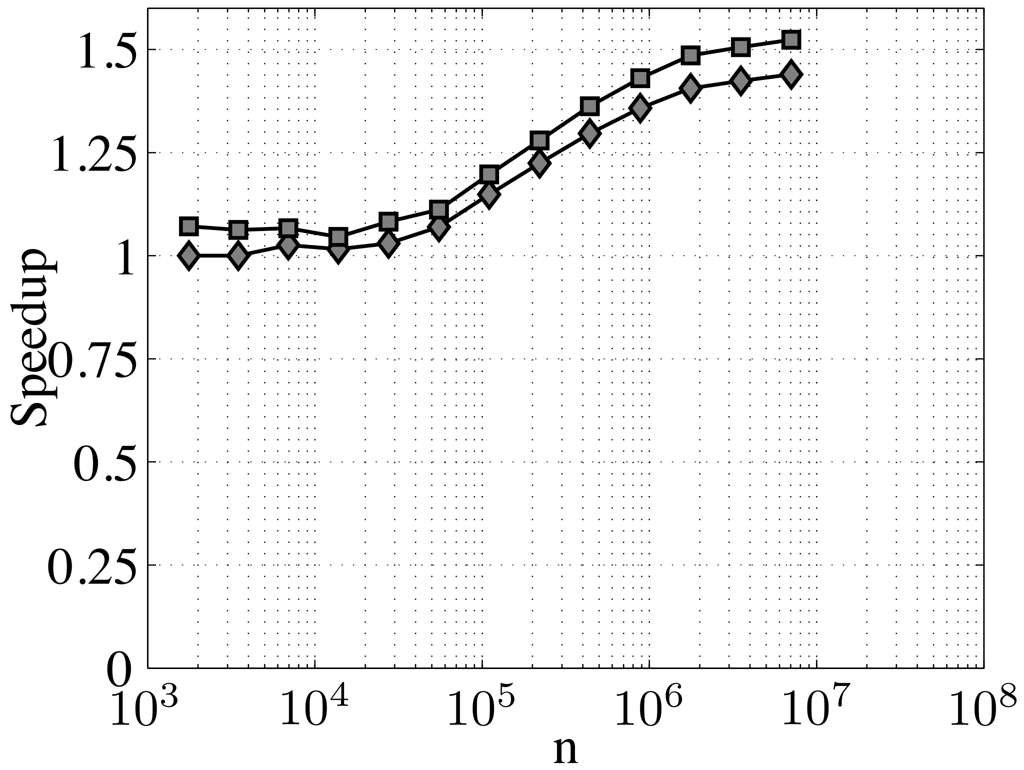



Figure 3. Speedup in scalability tests for C2050 with ECC for a single versus double precision arithmetic comparison. Single precision with ECC (–♦–) and without ECC (–■–). Iterative solver DC+MG-ZLGS-V(1,1) and sixth order spatial discretization have been employed.

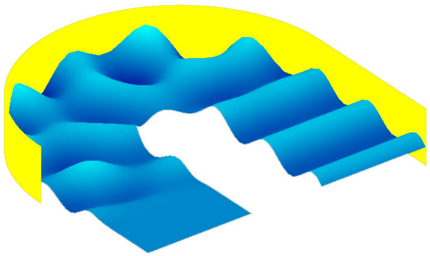
**Preliminary tests in SP**

Total GPU vs. CPU Speedup
==============================
C2050 : **x57** (measured)
GTX 480 : **x60** (estimated)
For sufficiently large problems

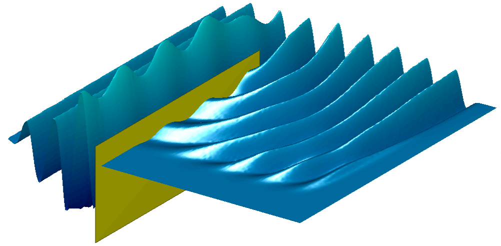
OceanWave3D code for GPUs **not** considered fully optimized.

Brute-force auto-tuning of most expensive kernels level have been used.

**No free lunch yet!**
Expected a factor x2 when using single over double precision...

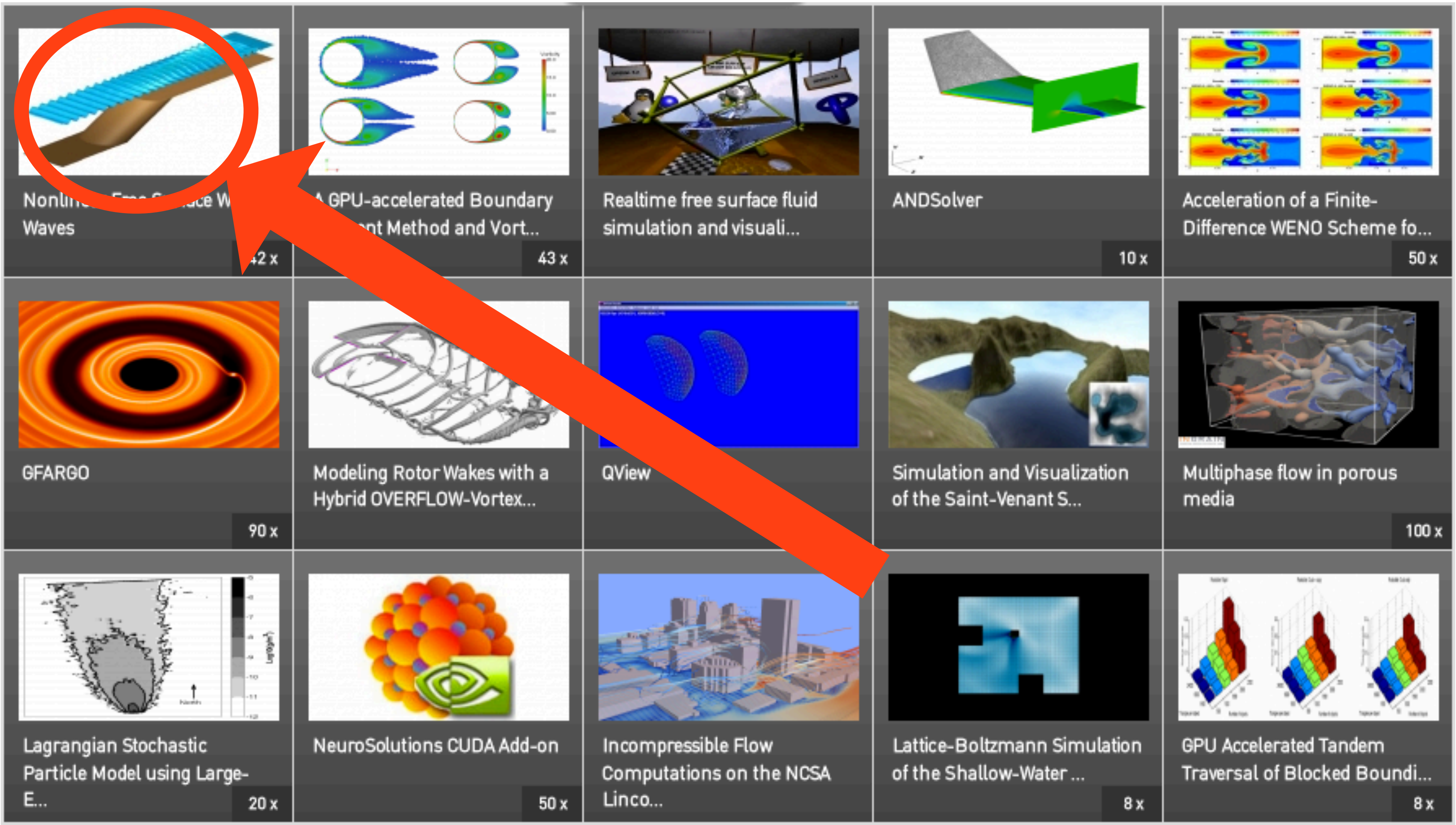


# Outlook



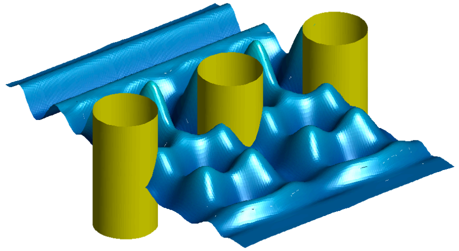
- Further optimization and auto-tuning of existing implementation could improve efficiency further
- Investigate means for leveraging productivity in code development (development cost is often overlooked and not negligible)
- Replace CUDA with OpenCL for better portability across hardware platforms, e.g. execution utilizing both CPUs and GPUs.
- Enable use of multi-GPU systems to efficiently solve even larger problems.
- Real-time computations and analysis requires fast algorithms, improvements in hardware and implementation/optimization effort.

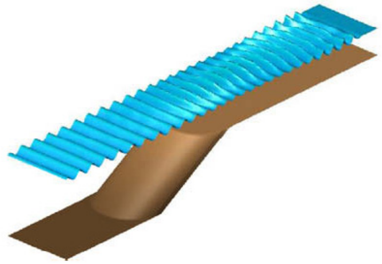
# General-purpose computing

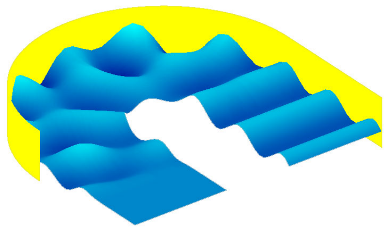


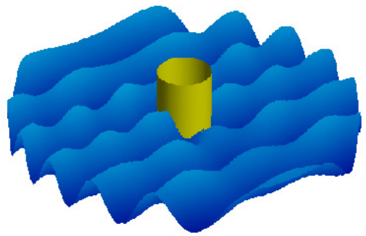

Many different applications from science and engineering show-cased in Nvidia's CUDA zone. All applications written in the CUDA framework after 2007!

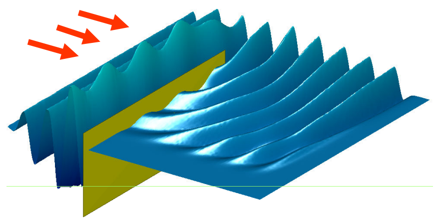
# Questions











**Contact info**

Allan P. Engsig-Karup
E-mail: apek@imm.dtu.dk
Associate Professor in Scientific Computing
Department of Mathematics and Modeling
Technical University of Denmark