Towards Exascale BEM simulations: Hybrid Parallelisation Strategies for Boundary Element Methods

Nicola Giuliani

Advisor: Luca Heltai
Co-Advisor: Andrea Mola

25th February 2016

Outline

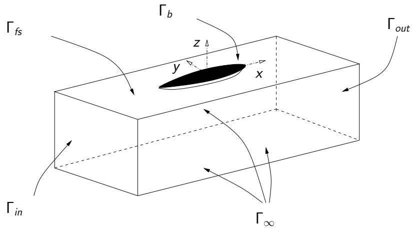
Laplace problem for the kinetic potential

Using the standard Green function for the Laplace problem we get :

$$\alpha(\mathbf{P})\phi(\mathbf{P}) = \int_{\partial\Omega} \left[\frac{\partial\phi}{\partial n} G - \frac{\partial G}{\partial n}\phi \right] ds_y$$

if **P** belongs to the boundary we can compute $\phi(\mathbf{P})$ treating only the boundary of the domain. We take the following assumptions:

- Simply Connected Domain Ω: water surrounding the body
- Unknown: perturbation potential ϕ from $\Phi = U_\infty x + \phi$



Boundary Conditions:

- $\dfrac{\partial\phi}{\partial n} = -\mathbf{V}_\infty \cdot \mathbf{n}$ on Γ_b
- $\dfrac{\partial\phi}{\partial n} = 0$ $\Gamma_{out}\Gamma_\infty$
- $\phi = 0$ on Γ_{in}

Laplace BEM: Starting Situation

We want to speed up the bulk of the Laplace BEM solver developed at the mathLab group. We have the following bottlenecks.

- Full matrices: memory allocations $O(N^2)$
- Very specialised-tuned code structure (`deal.II` + Trilinos)
- Particular requirements (gradient recovery) for industrial applications.

We want to increase dramatically the dimension of our problems. We need to approximate the system in order to be able to solve interesting problems ($N \sim 10^4$). The Fast Multipole Method can be a good choice $O(NlogN) - O(N)$.

- Existing FMM code not flexible enough, we have a working serial code

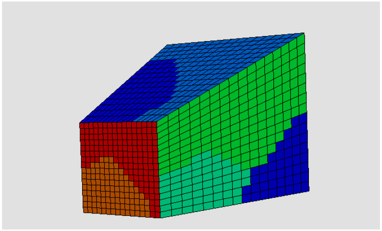
Parallelisation Strategy

BEM has the following characteristics

- Small geometry $\sim 10^4$ dofs, FULL matrices $\sim 10^8$ elements $\rightarrow$ common geometry distributed vectors and matrices.
- Embarrassingly parallel assembling loop $\rightarrow$ ideal to use MPI.

We have the following solutions

- METIS (graph partitioning tool) to subdivide the dofs between MPI processors.



- Trilinos and `deal.II` to manage the subdivisions and MPI

MPI using `deal.II`

Distributed vectors in `deal.II` provide an interface to ease the handle of MPI. We use the wrappers for the Trilinos library.

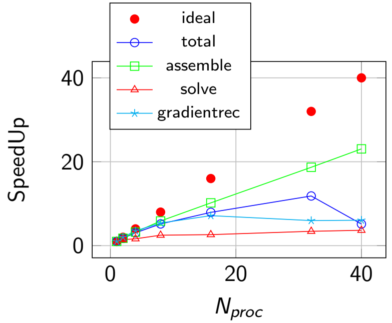
- Interface using global indices: easier to pass from serial to parallel.
- Usage of IndexSet to know the actual dofs *owned* by current processor (eventually *relevant* ghost elements) as well as the MPI communicator $\rightarrow$ set up of Trilinos maps (transparent to the user).
- Internal flag reports if we add or set any values. Need of consistency between processors.
- Communication managed through a simple call to *compress(Operation)*: distribution of ghost elements and resetting of the flag. All processors need to call it (MPI Barrier).

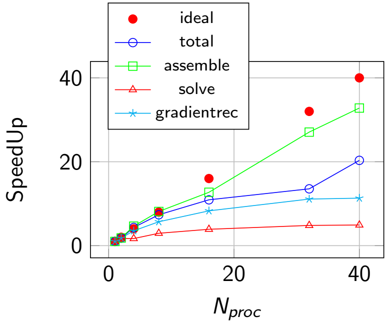
Implementation Keysteps

- Assembling: Split of the degrees of freedom between the MPI processors. NO ghost element required. Every processors assembles ONLY its line of the full matrix.
- Gradient Recovery: classical Finite Element method (!BEM). Need of proper ghosted IndexSet and mapping between scalar and vectorial unknowns.
- Solver: GMRES, Krylov Method with ILU preconditioner of the system matrix

BEM Strong Scaling

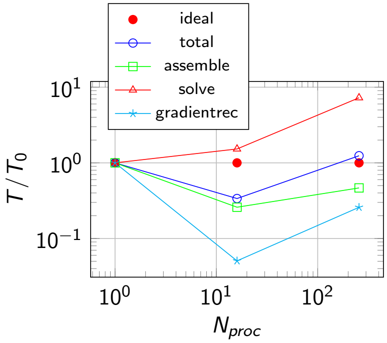
On the left 6k dofs, on the right 25k dofs.

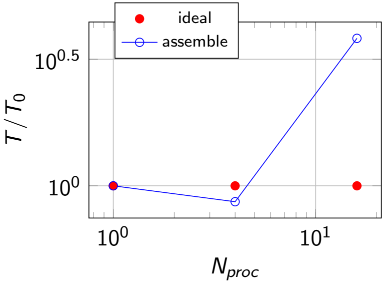




Optimale behaviour of the assembling, Solver issue (preconditioner)

BEM Weak Scaling





Superoptimal behaviour $\rightarrow$ Cell loop $O(N)$ for the considered complexity.

Conclusions BEM

- `deal.II` and METIS: general-automatic-balanced subdivision of the degrees of freedom without dividing the grid.
- `deal.II` and Trilinos: handling of arbitrary dof subdivision.
- Optimal Strong Scaling for the matrix assemblage.
- Actual bottleneck in the solver.
- Memory bottleneck for more than 66k dofs, moved to 64 bits indices.
- Superoptimal Weak Scaling.

Fast Multipole Method: Ingredients

We need some ingredients to be allowed to speak about Fast Multiple Methods.

- A specified acceptable accuracy of computation ϵ: without this point no FMM can exist.
- A hierarchical space subdivision (octree): crucial for Multipole and Local expansions - conversions
- A far field approximation of the fundamental solutions $(G(x,y), \dfrac{\partial G}{\partial n}(x,y)) \rightarrow$ Multipole and Local expansions.

$$K(x,y) = \sum_{k,j=1}^{\infty} \phi_k(x)\psi_j(y) \simeq \sum_{k,j=1}^{M} \phi_k(x)\psi_j(y)$$

Fast Multipole Method: Setting

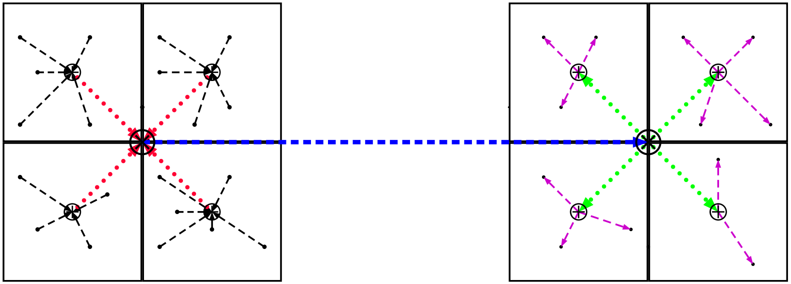
FMM is a method of particle dynamics. Discretise the BIE we get the same situation.

- We have M source (the quadrature points).
- We need to retrieve our unknowns in N nodes (the collocation points).
- We need to consider direct BEM interaction for near couples of points.
- We can use Multipole and Local expansions only if the accuracy estimates are satisfied (same size split boxes).

The starting code is extremely accurate, thus extremely complicate. It handles all the exceptions.

Fast Multipole Method: BEM

We need to value pairwise interaction. FMM splits them using: Ascending phase (M2M), Transfer phase (M2L), Descending phase (L2L).



Only in the golden case scenario (well split, same size block) we can apply all these steps.

Serial Code Profiling

We need a first code profiling to understand which are the bottlenecks of the serial code.

Method	Time	Repetitions
Tree Generation	1.798	1
Ascending Phase Time	0.53285714	42
Descending Phase time	22.91380952	42
Total Time	940	1.

The descending phase appears to be the first real bottleneck.

First Strategy: MPI

Our first solution is the following

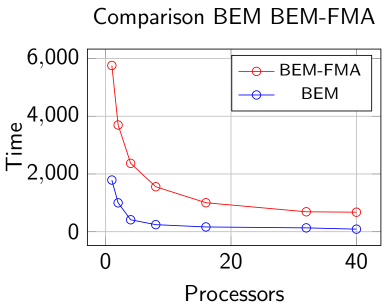
- We keep the METIS grid partitioning system, and the `deal.II` Trilinos handling of the vector
- We repeat the ascending phase on all the processors. We execute a AlltoAllv on every processor and then we perform the M2M.
- We split the direct BEM interactions as before.
- We split the descending phase using a pure MPI parallelisation. Every processors performs only the M2L, L2L it needs.

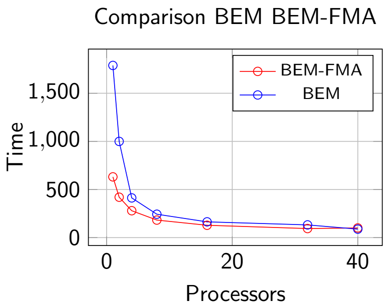
We are introducing a computational overhead

- Some M2L operations are repeated on different processors. We can't expect a perfect scalability even for the descending phase.

BEM or BEM-FMA?

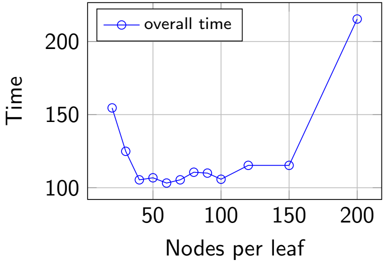
We need to assert that the BEM-FMA algorithm is a proper alternative in term of time for the BEM. This strongly depends on how many nodes per leaf we want to consider.
On the left the comparison with 1, on the right with 60.

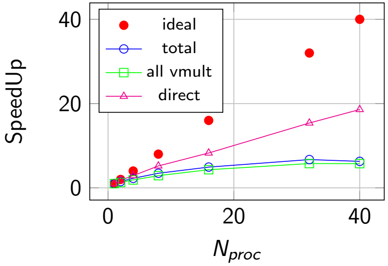




BEM-FMA Strong Scaling 40 procs 25 kdofs

On the left we look for the optimal nodes per block quantity. On the right we analyse BEM-FMA strong scalability.





- We recover the very good MPI scaling for the direct interactions.
- Low overall efficiency: Amdahl law
- We need a hybrid MPI-Multithread strategy

Second Strategy: Mixing MPI TBB

We choose Intel Threading Building Block as multithreading paradigm inside `deal.II` . TBB can be seen as an evolution of the standard OpenMP paradigm. Developed for the object oriented/template based programming style of C++.

- We apply TBB to every part of BEM-FMA.
- We split embarrassingly parallel loop between independent Tasks: TaskGroup
- If we need to handle race conditions: ???

TBB WorkStream

Problems:

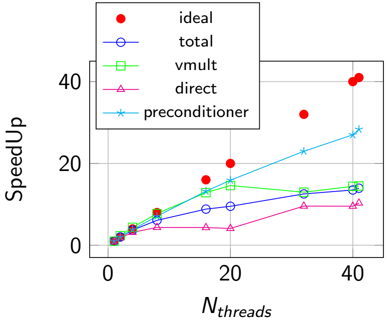
- Explicit synchronisation is rather expensive
- Floating points arithmetic: $a + b + c \neq c + a + b$

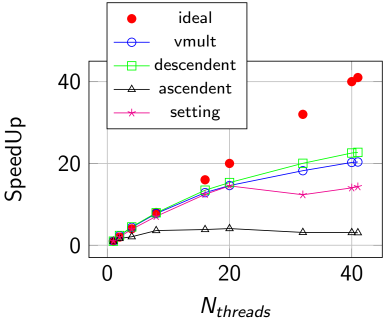
Usual pattern in FiniteElement codes: bunch of parallel operations over a stream of objects and a final reduction. We can split the two processes. Possible solution:

- The embarrassingly parallel, usually local, computations. To be run in any order.
- The final reduction to global memory.
    - The reduction operation should only run on a single thread to avoid explicit synchronisation.
    - The reduction operation should always run in exactly the same order.

That's WorkStream!

BEM-FMA Strong Scaling 98kdofs





We get an improved scalability but we still have the MPI parallelisation only for direct interactions and descending phase.

Conclusions BEM-FMA

- `deal.II` with METIS and Trilinos for the splitting and handling of the unknowns.
- Necessity of a hybrid parallelisation.
- MPI-TBB efficiency of 50% for a single vmult, almost 25% overall.
- MPI can't be used in ascending phase, impossible manual handling of the communications.
- Tested successfully up to 400k dofs. (less memory problems)

Perspectives

Achieved:

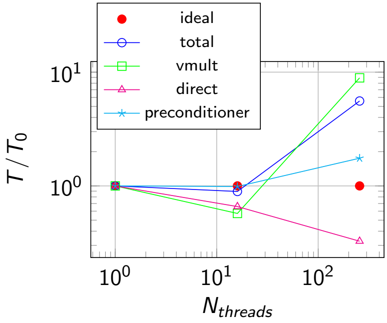
- Full Parallel (MPI) BEM code.
- Full Parallel (MPI+TBB) BEM-FMA algorithm.
- Code to treat any situation up to the 32 bits limit.
- A posteriori adaptive refinement strategies.

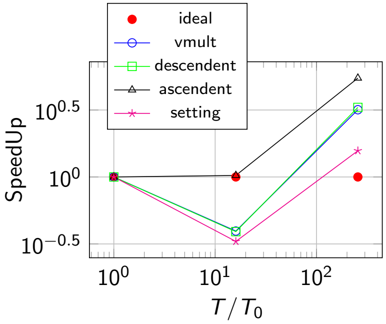
To do list:

- Proper documentation of the code.
- Release of the code.

BEM-FMA Weak Scaling

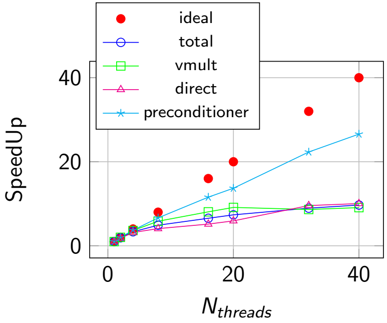
We consider the same computational cost as for the BEM ($O(N^2)$).
On the left the overall scalability, on the right the single vmult.

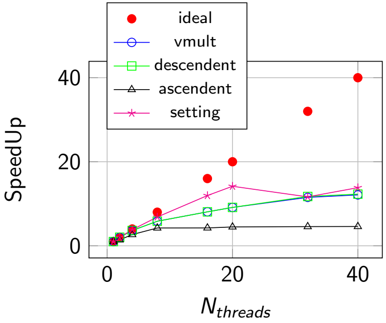




BEM-FMA Strong Scaling 25kdofs

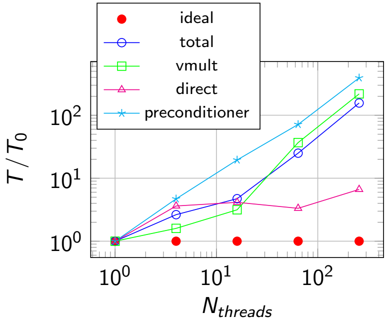
On the left the overall analysis, on the left the breakdown for a single matrix vector multiplication.

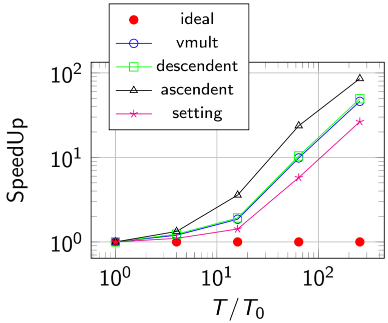




BEM-FMA Weak Scaling 2/2

We consider computational cost ($O(N)$). On the left the overall scalability, on the right the single vmult.





Fast Multipole Method: Introduction

BEMs require the evaluation of all pair-wise interactions in large ensemble of points (support points + quadrature points).

$$u(x) = \sum_{i=1}^{N} w_i K(x, y_i)$$

FMM consists in an approximation of the kernels K through its harmonic expansions.

$$K(x, y) = \sum_{k=1}^{\infty} \phi_k(x)\psi_k(y) \simeq \sum_{i=1}^{M} \phi_k(x)\psi_k(y)$$

For this reason we need an acceptable accuracy.

Potential Flow

With the following hypotheses:

- Incompressible fluid
- Negligible thickness of the boundary layer
- Irrotational flow $\Rightarrow \mathbf{V} = \nabla\Phi$

The conservation equations of mass and linear momentum become:

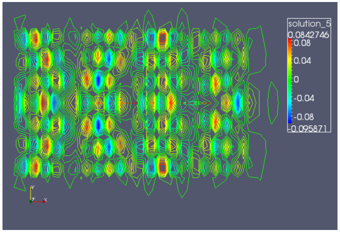
$$\begin{cases} \Delta\Phi = 0 & \text{Laplace Equation} \\ \dfrac{\partial\Phi}{\partial t} + \dfrac{1}{2}\nabla\Phi\cdot\nabla\Phi + gz + \dfrac{P}{\rho} = C(t) & \text{Bernoulli Equation} \end{cases}$$

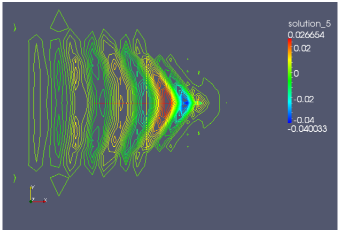
SUPG

$$\chi_j(x) \Rightarrow \chi_j^{SUPG}(x) = \chi_j(x) + \alpha h \nabla \chi_j(x) \cdot \frac{v(x)}{||v||}.$$

- Testes many α, in the results $\alpha = 0.5$

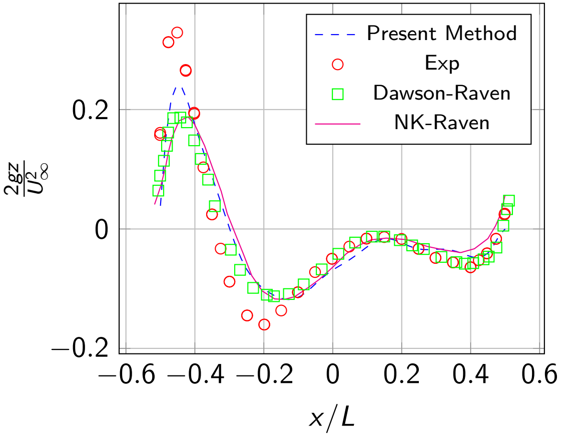
$$\int_\Gamma \chi_i^{SUPG} \chi_j \frac{\partial \phi}{\partial x_j}\, d\gamma = \int_\Gamma \chi_i^{SUPG} \frac{\partial}{\partial x}(\phi)\, d\gamma$$





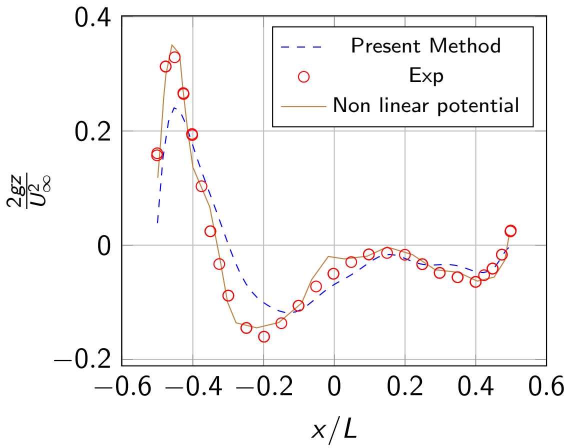
- Symmetry elimination. back

Wigley hull: BEM linearizzati Fr0316



- Good recovery of linearised methods: (NK, Dawson) back

Wigley hull: BEM with double nodes Fr0316



- Good behavior with double nodes [Grilli]
- Peak recovery with non linear method back

Linearised Free Surface Boundary condition

We apply a perturbation theory to the free surface conditions $\Rightarrow$ single condition [Newmann].

Under the following hypotheses:

- *little* waves: $A \ll \lambda$
- stationary solution
- imposition of the boundary condition on the undeformed free surface $\tilde{\Gamma}_{fs}$

We get:

- Linearised free surface condition
  $$U_\infty^2 \frac{\partial^2 \phi}{\partial x^2} + g \frac{\partial \phi}{\partial n} = 0 \qquad \text{on } \tilde{\Gamma}_{fs}$$
- height of the free surface: $\eta = -\frac{U_\infty}{g} \frac{\partial \phi}{\partial x}$ back

Free Surface boundary condition

The free surface boundary condition needs to properly describe the interactions between water and air.
With the hypotheses:

$$\sigma(x,y,z,t) = z - \eta(x,y,t) = 0 \text{ on } \Gamma_{fs} \qquad P(x,y,z,t) = P_a \text{ on } \Gamma_{fs}$$

we get:

- Kinematic condition: $\frac{D(z-\eta(x,y,t))}{Dt} = 0$, Γ_{fs} surface flow
- Dynamic condition $\Rightarrow \frac{\partial \Phi}{\partial t} + \frac{1}{2}\nabla\Phi \cdot \nabla\Phi + g\eta = \frac{1}{2}U_\infty^2$

Using standard perturbation theory we recover the linearised free surface boundary condition:

$$U_\infty^2 \frac{\partial^2 \phi}{\partial x^2} + g\frac{\partial \phi}{\partial n} = 0 \qquad \text{on } \Gamma_{fs}$$

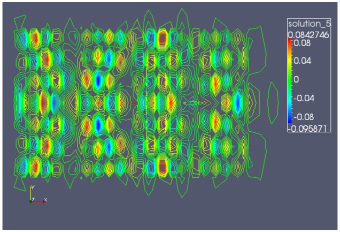
Treatment of the Linearised Free Surface BC

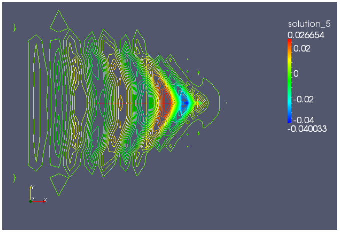
Solutions available in literature:

- Use of a modified Green function G [Neumann Kelvin]
- Use of upwind finite difference schemes in $U_\infty^2 \frac{\partial^2 \phi}{\partial x^2} + g \frac{\partial \phi}{\partial n} = 0$ [C.W.Dawson, 1977]

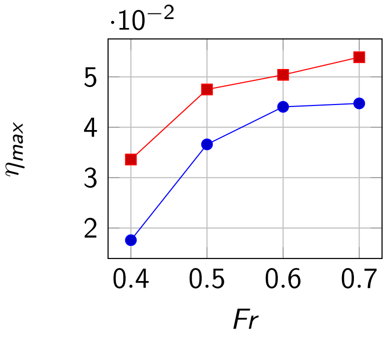
Original solution of the present works [Giuliani, Mola, Heltai, Formaggia, 2015]:

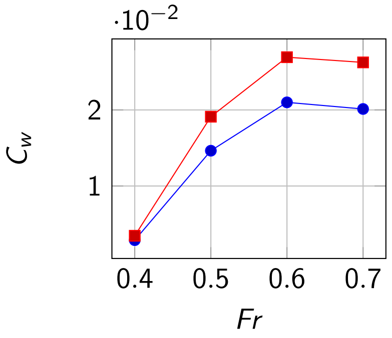
- Derivation in weak form + SUPG $\Rightarrow$ non structured and non conformal grids with low numerical dissipation.





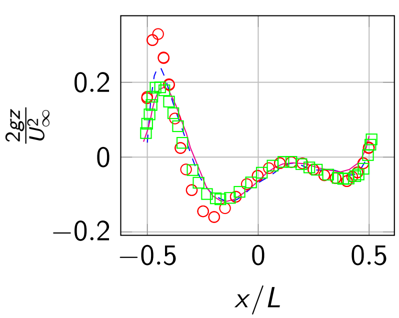
Test case: Submerged prolate spheroid

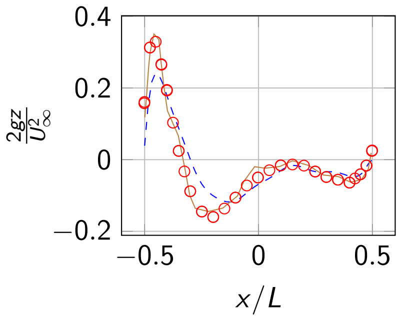




- Comparison present method with theoretical results [Scullen, Havelock]
- Underestimation of the wave height
- Underestimation of the wave drag

Test case: Wigley hull at Fr0316





- Same accuracy as linearised (NK, Dawson)
- Good behavior continuos BEM with double nodes [Grilli]
- Non linear method to recover the peaks