Master in High Performance Computing

Extension of the π-BEM library for use in a seakeeping pipeline

Supervisor(s):
Prof. Gianluigi Rozza,
Dott. Andrea Mola

Candidate:
Lorenzo Fabris

7th edition
2020–2021

Abstract

The problem of seakeeping consists, among other aspects, of studying how a ship reacts to the environmental conditions during navigation, to establish operational limits and verify the seaworthiness of the vessel. A classic approach is to model the ship as a filter, with a set of Response Amplitude Operators (RAO) transforming the wave elevation time series into the motions in the six degrees of freedom. Thanks to the convolutional properties of the Fast Fourier Transform, a wave elevation time series can then be converted to the prediction of ship motions in real time. For different ship speeds, wave directions and wave frequencies, the RAOs can be determined from the added masses, damping and hydrostatic coefficients, and wave-induced forces and moments. These entities can be obtained from the solutions of suitable complex-valued potential flow problems, in an offline phase. This work, developed as part of the Winning a Sea State project in collaboration with Cetena, illustrates the extension of the potential flow solver π-BEM to support these formulations, in order to provide an efficient, scalable, open source basis for a pipeline of this type. Results for the added mass and damping coefficients of a semi-submerged sphere in the zero-speed case compare favorably with the theory.

Contents

Introduction **1**

1 Formulation of the potential flow problems **3**
1.1 Introduction . . . 3
1.2 Equations of motion . . . 3
1.3 Potential flow . . . 6
1.4 Boundary conditions . . . 7
1.5 Post-processing - coefficients retrieval . . . 8
1.6 Practical remarks . . . 10

2 The π-BEM library **12**
2.1 Introduction . . . 12
2.2 High level features . . . 12
2.3 Boundary Element Method . . . 13
2.4 Implementation . . . 14
2.5 Fast Multipole Approximation . . . 16

3 Implementation of the missing features and performance enhancement **18**
3.1 Introduction . . . 18
3.2 Boundary conditions . . . 18
3.2.1 Robin boundary conditions . . . 19
3.2.2 Free surface boundary condition . . . 20
3.2.3 Complex-valued potential . . . 21
3.3 Code optimization . . . 22
3.4 Optimization benchmarks . . . 23
3.4.1 Algebraic solve . . . 25
3.4.2 Fast Multipole Approximation . . . 26
3.5 Complex-valued problem benchmarks . . . 31
3.5.1 Algebraic solve . . . 32
3.5.2 Fast Multipole Approximation . . . 32

4 Case study **37**
4.1 Introduction 37
4.2 Initial problem setup 37
4.3 Experiments and reworks 40

5 Conclusions **48**

Bibliography **53**

Introduction

A crucial problem during ship operations at sea is the prediction of the displacements induced by the waves, due to the frequently tight safety margins to observe for example in heavy loads movement or an helicopter take off. An efficient decision-making support system should be able to inspect the current states of the sea and the ship, predict their evolution for an appropriate time frame and highlight the windows in which critical operations are either permitted or forbidden according to safety conditions, in real time.
This is a multi-faceted problem, which requires the study of both the waves behavior and the ship's response.
Many tools have been established for the characterization of the sea waves, such as the Airy wave theory [19] for modeling regular sinusoidal waves in a linear potential flow framework, and statistical tool such as the Bretschneider and JONSWAP wave energy spectra [4, 20]. The most recent trend in deterministic wave prediction is the analysis of radar measurements, coupled with a wave propagation model in order to obtain the time history of the wave elevation in the reference system of the moving ship. An example is presented in [10].
The prediction of the ship motions, given the waves behavior, usually builds upon a rigid body model with six degrees of freedom. Deterministic forecasting has been approached with either direct integration of the equations of motion in the time domain [21], or by modeling the interaction of ship and waves as a signal processing problem with the ship acting as a filter, and the computations carried out in the wave frequency domain.
Other approaches entail the full order model simulations of Reynolds Averaged Navier Stokes Equations, but although they provide the highest fidelity of solutions, for example in simulating waves breaking, their computational effort forbids their application in real time systems. Model order reduction and Artificial Neural Networks are modern tools that show promise in bridging this gap, but their application to seakeeping remain limited [3, 18, 32].
Regarding direct integration of the equations of motion, this methodology is able to accommodate for non linear effects, such as those involved in precise roll prediction, but the required operations are expensive especially if the wave-induced forces and

moments are to be computed on the exact wet portion of the hull.
The frequency domain approach can instead exploit the convolutional properties of the Fast Fourier Transform [6], resulting in a much cheaper algorithm, at the price of being restricted to linear interactions.
Both of these approaches build anyway on the availability of the equations of motions coefficients: the added mass matrix, the damping coefficients matrix, the hydrostatic coefficients matrix, and ability of deriving wave-induced forces and moments from the time series of wave elevation. These entities are used directly in time domain integration, while the signal processing approach derives a set of linear transfer functions that convert the wave elevation to the motions in each degree of freedom: the Response Amplitude Operators.
One of the first investigations in how these entities can be computed, is given by Cummins in [8], where a set of complex-valued potential flow problems are derived under the assumptions of regular waves and potential wave theory. Further works such as [25, 26] expanded on the formulations and provided comparisons with experimental results.
In time, the preferred solver for these type of problems shifted from strip theory to the Finite Element Methods, to the Boundary Element Methods. Commercial codes used during ships design, such as PRECAL by the Cooperative Ship Research (RCS) [7], are frequently based on this latter approach, but scarce established open source applications are available.
The π-BEM library [12] developed at SISSA MathLab provides an efficient, scalable, free open source software (FOSS) implementation of a Boundary Element Method solver and Fast Multipole Approximation. In the Winning a Sea State project, in collaboration with Cetena [5], π-BEM was enhanced in order to support the potential flow problems used in linear seakeeping theory and a code for the offline phase of a full ship motions prediction pipeline was then initiated. Although the effort on the project moved on to the online phase of the pipeline, the added mass and damping coefficients computed with the new code, for a zero-speed semi-submerged sphere, compared favorably to the theory by Havelock [15].
The first chapter of this thesis describes the formulation of the potential flow problems.
The second chapter presents the π-BEM library.
The third chapter illustrates the implementation of the extensions to the library, which are required to handle the formulations from the first chapter, as well as the performance enhancements added.
The fourth chapter shows the results achieved for a submerged sphere and how they compare favorably to the theory.

Chapter 1

Formulation of the potential flow problems

1.1 Introduction

This chapter will present the mathematical modeling for a ship advancing in regular waves. A common coordinate system will be presented and the equations of motions will be characterized in terms of encounter frequency, under the assumptions of linearity and small regular waves.
The behavior of the fluid will be analyzed through a set of velocity potentials and a suitable formulation of boundary value problems will be discussed.
The resulting potentials will be postprocessed in order to retrieve the coefficients to be used in the equations of motion.
Finally, a discussion about the practical issues of this model will be discussed.

1.2 Equations of motion

Let us consider a ship advancing with mean forward speed U; on still water, the trajectory would be a straight line, but in the presence of waves the ship undergoes deviations and rotations.
To describe these motions, an orthogonal coordinate system is chosen so that the origin is placed on the undisturbed free surface and is fixed according to the mean position of the ship; the z axis extends vertically through the center of gravity of the ship and the x axis is oriented towards the bow.
With the ship modeled as a rigid body, its six degrees of freedom are surge, sway and heave along the x, y and z axis respectively; roll, pitch and yaw denote the rotations over the same axis. Let thus X denote the vector of the six displacements

X_i in each degree of freedom at a given time instant.
The equations of motion for the rigid body, in their general form, present nonlinear

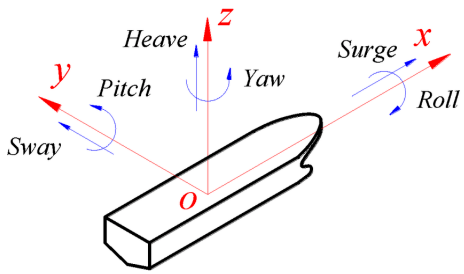

Figure 1.1: Ship motions

terms in the expressions of the angular displacements; by restriction to linear movements, these terms can be ignored in order to simply write

$$M\ddot{X} = F = F_R + F_H + F_W + F_F, \tag{1.1}$$

where M is the generalized mass matrix having the ship's mass on the top half of the diagonal and the inertial moments as the bottom right quadrant; it is to be noted that the inertial moments must be referred to the coordinates system's origin. The external forces and moments vector F is the composition of four different effects:

- F_R the restoring forces, containing the gravity force and the buoyancy response of the fluid to the volume displaced by the wetted hull;
- F_H the hydrodynamic forces, containing the inertial effects of the fluid's relative speed and the dissipation of energy due to the deformation of the free surface (wave making);
- F_W the wave exciting forces, due to the impacting waves;
- F_F the frictional damping effects;

The fluid is assumed inviscid, so that F_F is ignored.
Regarding F_R, linearization allows to express them as $F_R = -CX$ with C a matrix representing the hydrostatic stiffness. Only the coefficients coupling with a substantial variation of the buoyancy forces and moments are non zero, and they are computed from the integration of the geometric positions of the hull at the equilibrium position.
For F_H, it can be written $F_H = -A\ddot{X} - B\dot{X}$ where A and B are respectively the added mass and damping coefficients matrices. Both matrices are symmetric and even more equality relations between the coefficients can be applied in case the hull

present symmetries across the Oxz and/or the Oyz planes [22].
The study of F_W requires a model for the wave system: let us assume a regular wave of amplitude a_w, whose direction forms an angle β with axis x, has frequency ω_0, wave number k_0 and zero phase. For such a wave, elevation in a coordinate system with the x^0 axis aligned with the wave direction gives

$$\eta^0(x^0, t) = a_w \cos(k_0 x^0 - \omega_0 t). \tag{1.2}$$

Transforming in the coordinate system defined at the beginning, elevation at the origin becomes

$$\eta(t) = a_w \cos(k_0 U \cos(\beta) t - \omega_0 t) = a_w \cos(-\omega_e t) \tag{1.3}$$

$$\omega_e = \omega_0 - k_0 U \cos(\beta) \tag{1.4}$$

and ω_e is called the encounter frequency. From this we shall then assume that the vector of wave induced forces and moments contains elements of sinusoidal form

$$F_{Wi}(t) = F_{W0i} \cos(-\omega_e t + \gamma_i). \tag{1.5}$$

If F_{W0} and the γ_i are found for waves of unit amplitude $a_w = 1$, then they constitute a transfer function from the wave elevation to the induced forces and moments.
As the fundamental step for the characterization of the equations of motion for a ship in regular waves, let us assume that the A, B, F_{W0} coefficients and γ phases are constants for a given combination of encounter frequency, wave number, wave direction and ship speed. Then, the motions X will result in having sinusoidal form, with frequency ω_e.
For a more compact treatment of the phases, let us rewrite $X = \mathrm{Re}[\overline{X} e^{-j\omega_e t}]$ and $F_W = \mathrm{Re}[\overline{F} e^{-j\omega_e t}]$, with $\overline{X}, \overline{F} \in \mathbb{C}$ being the vectors of complex amplitudes and $j = \sqrt{-1}$.
The coupled equations of motions become

$$(M + A)\ddot{X} + B\dot{X} + CX = F_W \tag{1.6}$$

$$(-\omega_e^2 (M + A) - j\omega_e B + C)\,\mathrm{Re}[\overline{X} e^{-j\omega_e t}] = \mathrm{Re}[\overline{F} e^{-j\omega_e t}] \tag{1.7}$$

$$(-\omega_e^2 (M + A) - j\omega_e B + C)\overline{X} = \overline{F} \tag{1.8}$$

where we have gotten rid of the real part operator and the complex exponential.
With this model, it then become possible to retrieve the Response Amplitude Operators for the given ship and waves conditions

$$\overline{X} = (-\omega_e^2 (M + A) - j\omega_e B + C)^{-1} \overline{F} \tag{1.9}$$

This model is valid for small regular waves, meaning that the amplitude a is small compared to the wavelength $\lambda = 2\pi/k_0$, and small motions amplitude when compared with the hull dimensions.

1.3 Potential flow

The previous paragraph assumed the fluid to be inviscid; adding the assumptions of it being also irrotational and incompressible, it is well known that the Navier Stokes equations allow for the definition of a velocity potential Φ governed by the Laplace equation $\Delta\Phi = 0$.
Let us decompose the potential Φ, enforcing the sinusoidal behavior described before and relating each component to a role the fluid plays in the phenomenon:

$$\Phi = U(\phi_S - x) + \mathrm{Re}[(\phi_I + \phi_D + \sum_{i=1}^{6} \overline{X}_i \phi_i) e^{-j\omega_e t}] \tag{1.10}$$

The potential is composed of a steady part, due to the reference system moving with mean speed U, and an unsteady, time dependent part. The components are:

- ϕ_S the steady, trailing-wave making potential;
- ϕ_I the incident waves potential;
- ϕ_D the potential for the diffraction of the incident waves;
- ϕ_i the radiation potential of the fluid following the hull motion in the i-th mode;

with all ϕ appearing in the unsteady part being complex amplitudes to the exponential term.
A suitable characterization of ϕ_I can be found in Airy wave theory [19], which provides expressions for regular sinusoidal waves under the same assumptions used above.
Given a regular wave moving in water of depth d, the complex amplitude of the potential for a point with coordinates (x, y, z) is

$$\phi_I = -jg\frac{a_w}{\omega_0}\frac{\cosh(k_0(z+d))}{\cosh(k_0 d)} e^{jk_0(x\cos(\beta)+y\sin(\beta))}, \tag{1.11}$$

where g is the gravitational acceleration. A dispersion relation links the frequency and wave number:

$$\omega_0^2 = g k_0 \tanh(k_0 d). \tag{1.12}$$

Together with the steady terms, the potentials ϕ_I and ϕ_D are responsible for the values of F_W; similarly, the ϕ_i determine F_H and characterize the A and B coefficients.
The unknowns are solved for the ship at hydrostatic equilibrium.
For all problems, the fluid domain Γ is enclosed by:

- Γ_{hull} the mesh for the wetted surface of the hull;
- Γ_{free} the flat free surface at $z = 0$, with a hole accommodating Γ_{hull};
- Γ_{wall} the vertical walls of the tank, usually placed at a large distance from the hull;
- Γ_{bottom} the flat bottom at depth $z = -d$.

The normals of each boundary are oriented as entering the fluid.
It can be shown that all the ϕ appearing in Φ are subject to the Laplace equation in the fluid domain, which then constitutes the governing equation of each problem.

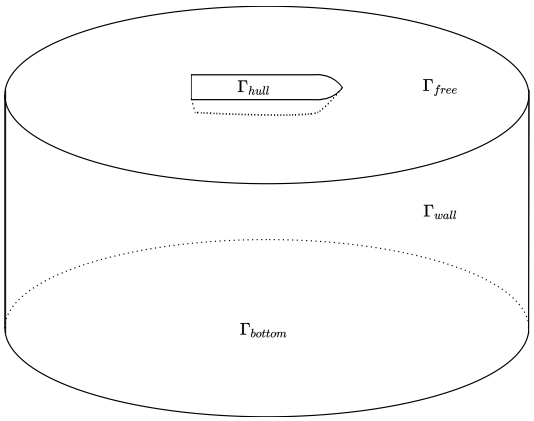

Figure 1.2: The mesh for the potential problems

1.4 Boundary conditions

On Γ_{free}, the conditions come by equating the fluid's pressure to the atmospheric pressure and linearizing:

$$U^2 \frac{\partial^2 \phi_S}{\partial x^2} + g \frac{\partial \phi_S}{\partial z} = 0, \tag{1.13}$$

$$\left(j\omega_e + U \frac{\partial}{\partial x} \right)^2 \phi_D + g \frac{\partial \phi_D}{\partial z} = 0, \tag{1.14}$$

$$\left(j\omega_e + U \frac{\partial}{\partial x} \right)^2 \phi_i + g \frac{\partial \phi_i}{\partial z} = 0 \text{ for } i = 1, .., 6. \tag{1.15}$$

On Γ_{hull}, the conditions come from the application of non penetration criteria:

$$\frac{\partial \phi_S}{\partial n} = U n_1, \tag{1.16}$$

$$\frac{\partial \phi_D}{\partial n} = -\frac{\partial \phi_I}{\partial n}, \tag{1.17}$$

$$\frac{\partial \phi_i}{\partial n} = -j\omega_e n_i + U m_i \text{ for } i = 1, .., 6, \tag{1.18}$$

where n is the generalized normal vector, which for a point $(x, y, z) \in \Gamma_{hull}$ is defined in terms of the normal $\vec{n}$ and the vector from the origin to the point $\vec{r} = (x, y, z)$:

$$(n_1, n_2, n_3) = \vec{n}, \tag{1.19}$$

$$(n_4, n_5, n_6) = \vec{r} \times \vec{n}\,. \tag{1.20}$$

The m terms represent the gradients of steady velocities in the normal directions:

$$(m_1, m_2, m_3) = -(\vec{n} \cdot \nabla)\nabla(\phi_S - x), \tag{1.21}$$

$$(m_4, m_5, m_6) = -(\vec{n} \cdot \nabla)[\vec{r} \times \nabla(\phi_S - x)]. \tag{1.22}$$

On Γ_{wall}, the steady potential must vanish and the unsteady components are subjected to a radiation condition at infinity. A classic candidate is the Sommerfeld radiation condition.

$$\phi_S = 0 \wedge \frac{\partial \phi_S}{\partial n} = 0, \tag{1.23}$$

$$\lim_{R \to +\infty} R\left(\frac{\partial \phi_D}{\partial R} - jk_0\phi_D\right) = 0, \tag{1.24}$$

$$\lim_{R \to +\infty} R\left(\frac{\partial \phi_i}{\partial R} - jk_0\phi_i\right) = 0 \text{ for } i = 1, .., 6, \tag{1.25}$$

where $R = |\vec{r}\,|$.
On Γ_{bottom}, all problems adopt a straightforward non-penetrating condition $\frac{\partial \phi}{\partial n} = 0$.

1.5 Post-processing - coefficients retrieval

Pressure on the hull is can be found through Bernoulli's equation:

$$p = -\rho\left(\frac{\partial \Phi}{\partial t} + \frac{1}{2}|\nabla\Phi|^2 + gz\right), \tag{1.26}$$

$$p = -\rho \,\mathrm{Re}\left\{ \left[-j\omega_e\phi_T + U\nabla(\phi_S - x)\cdot\nabla\phi_T\right] e^{-j\omega_e t} + \frac{1}{2}\left(\nabla\phi_T e^{-j\omega_e t}\right)^2 + \frac{1}{2}U^2\left[\nabla(\phi_S - x)\right]^2 + gz\right\}. \tag{1.27}$$

Integration of p on Γ_{hull} gives the total force on the ship due to the fluid; decomposition in each mode of motion is obtained by considering the contribution of each potential in the i-th component of the generalized normal.
To retrieve the coefficients used in the equations of motion defined before, only the terms presenting the complex exponential and influencing the z coordinate are relevant. The former will give the pulsating forces and moments, the latter will participate in the buoyancy effect.
Focusing on the pulsating forces and moments, by neglecting higher order terms:

$$p_T = -\rho \,\mathrm{Re}[(j\omega_e + U\frac{\partial}{\partial x})\phi_T e^{-j\omega_e t}]. \tag{1.28}$$

Let us link the forces and moments F_W and F_H to the relevant components in the potential:

$$F_{Wi} = -\rho \int_{\Gamma_{hull}} n_i \,\mathrm{Re}[(j\omega_e + U\frac{\partial}{\partial x})(\phi_I + \phi_D)e^{-j\omega_e t}]ds, \tag{1.29}$$

$$F_{Hi} = -\rho \int_{\Gamma_{hull}} n_i \,\mathrm{Re}[(j\omega_e + U\frac{\partial}{\partial x})(\sum_{k=1}^{6} \overline{X}_k\phi_k)e^{-j\omega_e t}]ds, \tag{1.30}$$

for $i = 1, .., 6$.
It is then possible to write

$$\overline{F}_i = -\rho \int_{\Gamma_{hull}} n_i(j\omega_e + U\frac{\partial}{\partial x})(\phi_I + \phi_D)ds, \tag{1.31}$$

$$\sum_{k=1}^{6}[-\omega_e^2 A_{ik} - j\omega_e B_{ik}]\overline{X}_k = -\rho \int_{\Gamma_{hull}} n_i(j\omega_e + U\frac{\partial}{\partial x})(\sum_{k=1}^{6} \overline{X}_k\phi_k)ds, \tag{1.32}$$

$$\omega_e^2 A_{ik} + j\omega_e B_{ik} = \rho \int_{\Gamma_{hull}} n_i(j\omega_e + U\frac{\partial}{\partial x})\phi_k ds, \tag{1.33}$$

thus

$$A_{ik} = \frac{\rho}{\omega_e^2}\int_{\Gamma_{hull}} n_i \,\mathrm{Re}[(j\omega_e + U\frac{\partial}{\partial x})\phi_k]ds, \tag{1.34}$$

$$B_{ik} = \frac{\rho}{\omega_e}\int_{\Gamma_{hull}} n_i \,\mathrm{Im}[(j\omega_e + U\frac{\partial}{\partial x})\phi_k]ds, \tag{1.35}$$

for $i, k = 1, .., 6$.
The steady forces and moments only alter the equilibrium position of the ship and do not participate in the ship's oscillatory motions.

1.6 Practical remarks

Solving the boundary value problems described in this chapter is a difficult task [17].
The complications arise from the interaction of steady and unsteady potential components: the second order derivative appearing in the free surface boundary conditions increases both the computational effort and numerical instability, while the m terms for the hull conditions extend negative effects on ϕ_S the other potentials. When $U = 0$, these difficulties disappear: the free surface boundary condition simplifies to a Robin boundary condition and the m terms vanish.
Conversely, ϕ_S has been studied in the absence of waves in [13], where the second order derivative was obtained through the solution of a subproblem in weak form, stabilized using the Streamline-Upwind-Petrov–Galerkin method.
A variety of approximations have been applied to retrieve one of the simpler formulations: the free surface condition can neglect U if small with respect to the wave frequency, while under slender body assumption or uniform base flow it results $m_5 = -n_3$ and $m_6 = n_2$ as $\nabla\phi_S$ is concentrated along x.
Regarding the mesh, [17] observes that when using panel methods, their dimension influence the quality of the results and recommends, for Γ_{free}, that a wavelength $\lambda = 2\pi/k_0$ is to be covered by at least 5 panels. This constraint leads to very fine meshes when λ is small and conversely when ω_0 is large.
The Sommerfeld boundary conditions are translated to

$$\frac{\partial\phi}{\partial n} - jk_0\phi = 0, \tag{1.36}$$

ignoring the z component of the radiation radius since the panels of Γ_{wall} are usually vertical.
Comparisons of the predicted roll motions with experimental data are usually unsatisfactory. This has been explained with the absence of friction and amplitude-dependent terms in the linear theory. Time domain codes address the issue with a correction of the damping coefficients.
Last, on the use of the RAOs, it must be stressed that their common representation as being simply functions of ω_e is misleading: by effect of the dispersion relation 1.4, the transformation from absolute to encounter wave frequency is usually rewritten as

$$\omega_e = \omega_0 - \omega_0^2 \frac{U}{g} \cos(\beta), \tag{1.37}$$

in deep water.
In case of following waves, that is for $\beta \in (-\pi/2, \pi/2)$, a wave of encounter frequency ω_e observed by the point of view of the moving ship might result from the superposition of multiple regular waves with different absolute frequencies:

actually three, since the encounter frequency is necessarily positive. Negative values found from the relation correspond to waves that are slower than the ship and are thus overtaken.
The respective RAO values might be very different, as the wave number k_0 is instead uniquely related to ω_0 and determines the distribution of the wave induced forces and moments on the hull. Indeed, the potential ϕ_I uses k_0 in its expression. It is then of practical use to store the RAO samples with abscissas in the absolute frequency domain. At runtime, prediction of motions in case of following waves is currently an open problem, with efforts directed towards exploitation of theoretical energy spectra in order to reconstruct the encountered wave components, such as in [23, 24].

Chapter 2

The π-BEM library

2.1 Introduction

The previous chapter introduced the mathematical tools exploited to describe the motions of a ship moving in regular waves, according to a linear model. A set of potential flow problems were given in order to compute the added mass and damping coefficients matrices, and the wave-induced forces and moments complex amplitudes.
This chapter presents π-BEM, a C++ library which provides a solver for the potential flow problems using the Boundary Element Method.
After a description of the high-level features of the library, the actual implementation of the Boundary Element Method will be discussed and the Fast Multipole Approximation (FMA) will be presented.

2.2 High level features

π-BEM: Parallel BEM Solver [12] is developed at SISSA MathLab and provides a high-quality, high-performance implementation of the Boundary Element Method for the solution of boundary value problems governed by the Laplace equation.
It is built on the deal.II library framework [1, 2], which collects a large number of utilities for the numerical solution of Partial Differential Equations using Finite Elements Methods.
The library handles mixed Dirichlet Neumann boundary value problems for real scalar potential, with the possibility of using Elements of arbitrary order.
The starting domain mesh can be refined according to the CAD description of the surfaces and curves, and π-BEM provides an iterative automatic refinement process based on error estimation of the solution found.

The linear algebra package used is Trilinos [16, 29], which implements both threaded and distributed MPI parallelism.
Additionally, an implementation of the FMA of the matrix-vector product is available for faster, albeit less precise, solution of larger problems.
The basic solution is given in terms of the potential and its normal derivative, but π-BEM also provides the retrieval of the full gradient through L_2 projection, in order to handle discontinuities in the normals.
Every aspect of the library can be configured through parameter files.

2.3 Boundary Element Method

The Boundary Element Methods are a family of techniques for the solution of the Laplace equation in a closed domain, exploiting the properties of the Green identity and the Rankine source.
Considering a boundary value problem governed by $\Delta\phi = 0$, in a closed domain Ω with boundary $\Gamma = \partial\Omega$.
We consider $G(\mathbf{r}) = \frac{1}{4\pi|\mathbf{r}|}$ which is the fundamental solution of the Laplace operator $\Delta G(\mathbf{r}) = \delta(\mathbf{r})$ in $\mathbb{R}^3$ and is called free-space Green's function or Rankine source.
Green's third identity states that the value of ϕ in the domain Ω can be retrieved knowing ϕ and $\frac{\partial\phi}{\partial n}$ on Γ

$$\phi(\mathbf{x}) = \int_\Gamma \left[G(\mathbf{x}-\mathbf{y})\frac{\partial\phi}{\partial n}(\mathbf{x}) - \phi(\mathbf{x})\frac{\partial G}{\partial n}(\mathbf{x}-\mathbf{y}) \right] ds_\mathbf{y} \tag{2.1}$$

Restriction to the $\mathbf{x} \in \Gamma$ requires to handle the singularities of G, which is still integrable, and $\frac{\partial G}{\partial n}$, which requires a different approach. Using the Cauchy Principal Value for the non-integrable singularity:

$$\alpha(\mathbf{x})\phi(\mathbf{x}) = \int_\Gamma G(\mathbf{x}-\mathbf{y})\frac{\partial\phi}{\partial n}(\mathbf{x})ds_\mathbf{y} - \int_\Gamma{}^{PV} \phi(\mathbf{x})\frac{\partial G}{\partial n}(\mathbf{x}-\mathbf{y})ds_\mathbf{y} \tag{2.2}$$

where the function $\alpha(\mathbf{x})$ is the remaining part of the Cauchy Principal Value integral and represents the fraction of solid angle with which the domain Ω is seen from the boundary point $\mathbf{x}_i$.
Thus, from the original problem it has been possible to distill a relation which only uses the values of ϕ and $\frac{\partial\phi}{\partial n}$ on Γ, maintaining the ability to evaluate ϕ in each internal point of Ω.
Indeed the main advantage of the Boundary Element Methods is the use of shell meshes.

2.4 Implementation

Both ϕ and $\frac{\partial \phi}{\partial n}$ are discretized using the same Finite Elements, collocated on the refined mesh.
The nodal values of the two functions can then be represented by a pair of vectors $\hat{\phi}_i = \phi(\mathbf{x}_i)$ and $\hat{\gamma}_i = \frac{\partial \phi}{\partial n}(\mathbf{x}_i)$, with $\mathbf{x}_i$ being the coordinates of the i-th node.
The vectors must then respect the relation $(\alpha + N)\hat{\phi} = D\hat{\gamma}$ where N and D are square matrices obtained by an appropriate quadrature scheme:

$$N_{ij} = \sum_{k}^{K} \sum_{q}^{Q_k} \frac{\partial G}{\partial n}(x_i - x_q)\psi_q^j J^k, \tag{2.3}$$

$$D_{ij} = \sum_{k}^{K} \sum_{q}^{Q_k} G(x_i - x_q)\psi_q^j J^k, \tag{2.4}$$

and α is the square matrix with the $\alpha(\mathbf{x}_i) = \sum_{j=1}^{n} N_{ij}$ values on the diagonal. For every cell k, the quadrature basis function is evaluated for the reference cell $\hat{K}$ on $\mathbf{x}_j$ and weighted by J^k, which is the determinant of the transformation from $\hat{K}$ to k.
The resulting matrices are dense, non-symmetric, and the ij-th element represents the contribution of the j-th nodal value to one of the integrals defined above, that is, to the i-th node.
The kernel functions can become singular in the case that two distinct nodes share the same coordinates, which is possible with complicated geometries. This "double nodes" case is handled with an ad hoc quadrature.
The $\hat{\phi}$ and $\hat{\gamma}$ vectors can be solved when suitable boundary conditions constraint part of their values. π-BEM supports mixed Dirichlet Neumann problems where the boundary is partitioned $\Gamma = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$ and for each partition: $\phi = f_D$ in Γ_D and $\frac{\partial \phi}{\partial n} = f_N$ in Γ_N.
Considering the partitioning of the elements, additional scaling vectors are introduced:

$$\overline{\phi} = \begin{cases} \hat{\phi}_i = f_D(\mathbf{x}_i) & \text{if } \mathbf{x}_i \in \Gamma_D \\ 0 & \text{if } \mathbf{x}_i \in \Gamma_N \end{cases}$$

$$\overline{\gamma} = \begin{cases} 0 & \text{if } \mathbf{x}_i \in \Gamma_D \\ \hat{\gamma}_i = f_N(\mathbf{x}_i) & \text{if } \mathbf{x}_i \in \Gamma_N \end{cases}$$

$$\tilde{\phi} = \begin{cases} 0 & \text{if } \mathbf{x}_i \in \Gamma_D \\ \hat{\phi}_i & \text{if } \mathbf{x}_i \in \Gamma_N \end{cases}$$

$$\tilde{\gamma} = \begin{cases} \hat{\gamma}_i & \text{if } \mathbf{x}_i \in \Gamma_D \\ 0 & \text{if } \mathbf{x}_i \in \Gamma_N \end{cases}$$

Multiple boundary conditions of the same type are possible, with different boundary values to be applied. This is handled by simply imposing the known values on the elements of $\overline{\phi}$, $\overline{\gamma}$ based on the actual boundary, so that the vectors are initialized from multiple f_D, f_N.
Then $\hat{\phi} = \overline{\phi} + \tilde{\phi}$ and $\hat{\gamma} = \overline{\gamma} + \tilde{\gamma}$ and it becomes possible to collect known and unknown terms from the original matricial relation:

$$(\alpha + N)\tilde{\phi} - D\tilde{\gamma} = D\overline{\gamma} - (\alpha + N)\overline{\phi} \tag{2.5}$$

In order to find the unknown values using a Krylov solver, π-BEM adopts a single vector $\tilde{t} = \tilde{\phi} + \tilde{\gamma}$ and constructs a right hand side vector (RHS) $b = D\overline{\gamma} - (\alpha + N)\overline{\phi}$. The linear operator A needed to solve $A\tilde{t} = b$ is implemented by partitioning $\tilde{t}$ back into $\tilde{\phi}$ and $\tilde{\gamma}$ and performing the full matrix-vector products. The solver of choice for π-BEM is the Restarted Preconditioned Direct Generalized Minimal Residual Method (GMRES).
The system matrix A can be viewed as built from the columns of N and D, with the j-th column taken from $(\alpha + N)$ if the j-th node is part of Γ_N or by $-D$ otherwise. The default preconditioner uses the Incomplete LU factorization (ILU) of a banded matrix built according to this structure.
In the presence of double nodes, the system must also apply linear constraints to the unknowns. A delicate case is that of two nodes subject to different Dirichlet boundary conditions: if the normals of the two nodes have a non negligible difference, the two normal derivatives must satisfy $|\vec{n}_i|\tilde{\gamma}_i = |\vec{n}_j|\tilde{\gamma}_j$. Other cases simply condense away the row.
All the vectors and matrices used in this workflow are instances of the `TrilinosWrappers::MPI` classes provided by deal.II, which use the Trilinos package Epetra [9] to distribute the data across MPI processes.
The partitioning operations are carried out through element-wise multiplication with masking vectors:

$$\nu_D = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \Gamma_D \\ 0 & \text{if } \mathbf{x}_i \in \Gamma_N \end{cases}$$

$$\nu_N = \begin{cases} 0 & \text{if } \mathbf{x}_i \in \Gamma_D \\ 1 & \text{if } \mathbf{x}_i \in \Gamma_N \end{cases}$$

For the sake of notation, the vectors intended for element wise scaling will be written as square diagonal matrices whose diagonal is the vector itself, so that ν_D is really a shorthand for $\nu_D I$ where I is the identity matrix.
It is then possible to write

$$(\nu_D + \nu_N)_i = 1,$$
$$\nu_D\hat{\phi} = \overline{\phi},$$

$$\nu_N \hat{\phi} = \tilde{\phi},$$
$$\nu_D \hat{\gamma} = \tilde{\gamma},$$
$$\nu_N \hat{\gamma} = \overline{\gamma}$$

and the same principle is applied to α.
The computational complexity of the matrix assemblage is $O(n^2)$ where n is the number of nodes, with a conspicuous constant factor due to the number of quadrature points and the transformations of the weighting factors from the reference cell. The GMRES solver has no guarantee on the number of iterations, but the linear operator implementation performs two matrix-vector product with obvious $O(n^2)$ complexity, which dominates the remaining linear complexity operations due to masking and multiplication with α.

2.5 Fast Multipole Approximation

π-BEM implements an approximation algorithm for matrix-vector products, based on the Fast Multipole Method [14].
The original algorithm has been developed in the context of molecular dynamics, for the approximated evaluation of potentials due to a distribution of charges. The method exploits the fact that the potential from a group of charges which is distant from the evaluation point might be approximated with a suitable truncated series expansion, of guaranteed error estimate.
By building an octree to quickly discern near and distant cells, interactions with the distant charges can be approximated through the harmonic expansions of a low number of cells. The resulting procedure uses pair-wise evaluation of the potential only for close range interactions; depending on the spatial distribution of the charges and the expansions computation, the final computational cost is much lower than the full pair-wise procedure.
Translating this model to the Boundary Element Method's matrices definitions 2.3 and 2.4 it can be seen that:

- the quadrature points act as charges, whose strength is given by the nodal values through the shape functions evaluation
- the Rankine source or its normal derivative represent the potential function, which can be approximated with a complex-valued harmonic series, truncated
- the nodes are the evaluation points

Since the spatial distribution of the nodes is fixed once the mesh has been refined, the octree itself and metadata such as the type of interaction between cells or the list of directly interacting node-quadrature point pairs are computed only once and stored.
Direct interactions are obtained through a sparse-matrix-vector product, assembled with the same procedure as before, but where the elements that link non-interacting nodes are left empty.
Evaluation of the remaining contributions can be divided in two phases: an ascending phase in which every cell summarizes the effects of its children, and a descending phase in which the leaf cells evaluate the distant interactions in its nodes.
During the ascending phase, the leaf cells compute the harmonic expansion of their quadrature points, a task which is sped up by precomputing a set of translator functions. The harmonic expansion coefficients resulting from the combination with the nodal values are then collected by the cell's parent, which can then build its own harmonic expansion by a merge operation. The *far field* expansion of a cell is to be used outside of it.
In the descending phase, each cell must instead merge the *far field* expansions from distant cells of equal or greater size, into a *local* expansion. Bigger cells are already accounted for by the parent cell *local* expansion, while the distant cells of same size are traversed and their *far field* expansions merged one at a time. The leaf cells can evaluate their *local* expansions at their nodes, to finally retrieve the elements of the matrix-vector product output.
The harmonic series truncation exponent p is a critical value for the algorithm, as it guides both the error bound and the work required for merging and translating the expansions. The error decreases exponentially and the series manipulations require $O(p^4)$ work due to having p^2 coefficients in the three dimensional case.
It can be shown that the total time due to expansion manipulation achieves $O(\frac{n}{s}p^4)$ complexity where s represents the average number of evaluation points in a cell. Thus, the full evaluation of the matrix-vector product approximation through the Fast Multipole Method approaches linear time complexity, albeit with large constant.
π-BEM adopts $s = p = 20$ as discussed in [11]; the procedure is also scalable though threads and MPI processes.

Chapter 3

Implementation of the missing features and performance enhancement

3.1 Introduction

The goal is to extend π-BEM in order to support the complex-valued potential flow formulation detailed in the first chapter. This chapter will illustrate the design and implementation of the required features.
Following the analysis of the library from the previous chapter, the gaps are:

- support for Robin boundary conditions;
- support for the linearized free surface boundary condition;
- support for complex-valued potential and complex-valued coefficients in the boundary conditions.

Moreover, many sections of the code were updated to use more modern idioms and more performant functionalities of deal.II and the C++ Standard Template Library (STL).

3.2 Boundary conditions

Building on the existing Dirichlet and Neumann boundary conditions implementation, the new boundary condition types were added by extending the partitioning of the boundary and the nodal values vectors.

In these new boundary conditions, the values of the potential and its normal derivative are constrained in a linear combination: it is then possible to express one as a function of the other and this enabled the extension of the existing linear operator.

3.2.1 Robin boundary conditions

Let us consider a new partitioning of the domain $\Gamma = \Gamma_D \cup \Gamma_N \cup \Gamma_R$, with $\Gamma_D \cap \Gamma_N = \Gamma_D \cap \Gamma_R = \Gamma_N \cap \Gamma_R = \emptyset$.
Recalling the discretization adopted by π-BEM, for a node in Γ_R with coordinates $\mathbf{x}_i$ we have

$$c_0\hat{\phi}_i + c_1\hat{\gamma}_i = c_2. \tag{3.1}$$

First, a new masking vector ν_R was introduced:

$$\nu_R = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \Gamma_R \\ 0 & \text{otherwise} \end{cases}$$

and the previous masking vectors ν_D and ν_N were adjusted accordingly.
Let us update the notation for the existing vectors of nodal values, still using the overline for vector whose values are completely known and the tilde for vectors containing unknowns:

$$\hat{\phi} = \overline{\phi} + \tilde{\phi} + \tilde{\phi}_R \tag{3.2}$$

$$\hat{\gamma} = \tilde{\gamma} + \overline{\gamma} + \tilde{\gamma}_R \tag{3.3}$$

$$\nu_R\hat{\phi} = \tilde{\phi}_R \tag{3.4}$$

$$\nu_R\hat{\gamma} = \tilde{\gamma}_R \tag{3.5}$$

The elements of $\tilde{\gamma}_R$ can be retrieved by a linear transformation of $\tilde{\phi}_R$:

$$\tilde{\gamma}_R = r - R\tilde{\phi}_R \tag{3.6}$$

where r and R are defined as

$$r = \begin{cases} c_2/c_1 & \text{if } \mathbf{x}_i \in \Gamma_R \\ 0 & \text{otherwise} \end{cases}$$

$$R = \begin{cases} c_0/c_1 & \text{if } \mathbf{x}_i \in \Gamma_R \\ 0 & \text{otherwise} \end{cases}$$

then the linear system's governing relation become

$$(\alpha + N)(\overline{\phi} + \tilde{\phi} + \tilde{\phi}_R) = N(\tilde{\gamma} + \overline{\gamma} + \tilde{\gamma}_R) \tag{3.7}$$

$$(\alpha + N)(\overline{\phi} + \tilde{\phi} + \tilde{\phi}_R) = N(\tilde{\gamma} + \overline{\gamma} + r - R\tilde{\phi}_R) \tag{3.8}$$

and by collecting unknown and known terms:

$$(\alpha + N)(\tilde{\phi} + \tilde{\phi}_R) - D(\tilde{\gamma} - R\tilde{\phi}_R) = D(\overline{\gamma} + r) - (\alpha + N)\overline{\phi} \tag{3.9}$$

The only changes to the linear operator given to the GMRES solver consist of four linear vector operations. The RHS and the preconditioner were updated accordingly, as well as the final reconstruction of $\hat{\phi}$ and $\hat{\gamma}$.
The constraints for system condensation were updated by simply having the Robin nodes be considered as being Neumann nodes, but enforcing that in case they share the same coordinates with a proper Neumann node, the latter values from the boundary condition is applied.

3.2.2 Free surface boundary condition

This kind of mixed Robin boundary conditions is much more involved, but follows the same procedure: the partitioning was updated with a new element Γ_{FS}, with the corresponding definition of vectors v_{FS}, $\tilde{\phi}_{FS}$ and $\tilde{\gamma}_{FS}$.
Although the proper expression would contain the value of $\frac{\partial \phi}{\partial z}$, in practice it is always applied to flat surfaces for which it coincides with $\pm\frac{\partial \phi}{\partial n}$.
Thus for a node in Γ_{FS} with coordinates $\mathbf{x}_i$ we have

$$c_0\hat{\phi}_{FSi} + c_1\frac{\partial \hat{\phi}_{FS}}{\partial x}(\mathbf{x}_i) + c_2\frac{\partial^2 \hat{\phi}_{FS}}{\partial x^2}(\mathbf{x}_i) + c_3\hat{\gamma}_{FSi} = c_4 \tag{3.10}$$

For a robust treatment of the partial derivatives along x, we followed the procedure in [13] which solves a subproblem in weak form, in order to express $\tilde{\gamma}_{FS}$ as a linear transformation of $\tilde{\phi}_{FS}$:

$$\tilde{\gamma}_{FS} = r_{FS} - (R_{FS0} + R_{FS1}M^{-1}B + R_{FS2}M^{-1}BM^{-1}B)\tilde{\phi}_{FS} \tag{3.11}$$

where the M and B are square matrices whose elements are defined as

$$M_{ij} = \sum_k^K \sum_q^{Q_k} \check{\psi}_q^i \psi_q^j J^k \tag{3.12}$$

$$B_{ij} = \sum_k^K \sum_q^{Q_k} \check{\psi}_q^i \frac{\partial \psi_q^j}{\partial x} J^k \tag{3.13}$$

Since the problem at hand is transport dominated, the basis function $\check{\psi}$ used in the quadrature is modified according to the Streamline Upwind Petrov Galerkin technique, that is:

$$\check{\psi} = \psi + \delta h \nabla_S \psi \cdot (-1, 0, 0), \tag{3.14}$$

where $\nabla_S \psi$ is the surface gradient on the current node, δh represents the streamwise cell extension and $(-1, 0, 0)$ is the fluid velocity direction. The resulting matrices are non-symmetric, but sparse.
The other vectors involved in the relation are defined as

$$r_{FS} = \begin{cases} c_4/c_3 & \text{if } \mathbf{x}_i \in \Gamma_{FS} \\ 0 & \text{otherwise} \end{cases}$$

$$R_{FS0} = \begin{cases} c_0/c_3 & \text{if } \mathbf{x}_i \in \Gamma_{FS} \\ 0 & \text{otherwise} \end{cases}$$

$$R_{FS1} = \begin{cases} c_1/c_3 & \text{if } \mathbf{x}_i \in \Gamma_{FS} \\ 0 & \text{otherwise} \end{cases}$$

$$R_{FS2} = \begin{cases} c_2/c_3 & \text{if } \mathbf{x}_i \in \Gamma_{FS} \\ 0 & \text{otherwise} \end{cases}$$

To integrate this new relation in the linear operator, it is necessary to invert M. In order to find $x = M^{-1}By$, our implementation solves the linear system $Mx = By$ using GMRES, as explicit inversion is not available to the distributed Trilinos matrices used.
For each invocation of the linear operator, two such systems are solved, with the output from the first being recycled.
Modification of the RHS is trivial, but the preconditioner is much more involved and requires the explicit extraction of the $M^{-1}B$ columns.
Condensation constraints were handled in the same fashion as for the Robin conditions.
At the time of writing, the implementation is not yet mature and is not suitable for production use. Initial results on the generation of Kelvin wakes are nonetheless encouraging.

3.2.3 Complex-valued potential

The Epetra package does not support complex numbers and this constraint is reflected on the `TrilinosWrappers` classes that π-BEM uses. To bypass this limitations, while maintaining the ability to efficiently handle real-valued problems,

the library was modified in order to handle multiple problems defined on the same mesh.
Solving a complex problem then reduces to processing a pair of problems together, where the first defines the real parts of the boundary conditions and the second problem defines the imaginary parts.
Exploiting the decomposition of $Ax = b$ with $x, b \in C$ we can write:

$$\mathrm{Re}[A]\,\mathrm{Re}[x] - \mathrm{Im}[A]\,\mathrm{Im}[x] + j(\mathrm{Im}[A]\,\mathrm{Re}[x] + \mathrm{Re}[A]\,\mathrm{Im}[x]) = \mathrm{Re}[b] + j\,\mathrm{Im}[b] \quad (3.15)$$

This required to split the implementation of the solve procedure, but since the N, D, M and B matrices are purely real, the only contribution to the imaginary part of the system linear operator comes from the coefficients of the boundary conditions, which are implemented as scaling operations. The actual increase in the operator computational cost is then only slightly over a factor of 2, as it is governed by the matrix-vector products of quadratic complexity. The existing implementations can be reused as-is in the new methods, which only handle the combination of inputs and outputs. The object fed to GMRES is just a `BlockVector`, which allows to efficiently reference the original `Vector` objects without unnecessary copies.
Due to the treatment of the new boundary conditions, where the subjected nodes are used as if being of Neumann type, the constraints for system condensation are applied separately to the real and imaginary parts of the unknowns with no need to account for their pairing.
On the other hand, the original preconditioner based on the ILU loses some of its strength, as for non-trivial problems the band size required to include the elements outside the top left and bottom right quadrants would defeat its purpose.
Finally, the adoption of an arbitrary number of problems for the same mesh, and thus the same N and D matrices, gives a great advantage in the solutions of problems such as those presented in the first chapter, as the assemblage of the matrices or the octree needs not be repeated anymore.

3.3 Code optimization

Much of the existing code was simplified in order to gain readability and exploit library code from either the STL, or the deal.II and Trilinos functions. Notable examples were the conversion of initializations originally performed with explicit for loops, to use iterator-based methods: this was common during copy of STL containers like `vector` and `map`, but had dramatic effects for the `SparsityPattern` and `SparseMatrix` objects where the adoption of library functions resulted in conspicuous speedups.
Being distributed, the nodes indices were iterated on with for loops which tested

membership to a local `IndexSet` object. These loops were reimplemented by proper range-based iteration on the `IndexSet` itself, which could exploit the inner workings of the data structure.
Vector operations were reorganized in order to reduce temporary variables and, in some cases, merge together linear traversals such as with the method `sadd`, which scales and merges two vectors at once.
Regarding the FMA, the very large number of metadata structures which stored interaction lists, neighbor cells, *far field* and *local* expansions was frequently accessed with copy construction. The read-only accesses were converted to use constant references.
In the procedure implementing the merge of harmonic series, extremely good results were obtained by caching the evaluations of the complex roots of unity or the basis of the expansion, and refactoring the loops in order to reduce the number of complex products.
Finally, the N and D matrix assemblage was given a threaded implementation using deal.II `WorkStream` pattern [30], based on Intel's Thread Building Blocks [27]. Unfortunately, the vectorized methods which were so efficient in single thread execution, could not be used without locking accesses by row. This meant that implementation of single-threaded and multi-threaded assemblage had to be split, and whether to use one or the other was decided at runtime.

3.4 Optimization benchmarks

π-BEM provides a good number of Teuchos [28] timers to build a basic profiling of its executions.
In order to validate the performance improvement from the code update, a simple benchmark problem was defined by applying Dirichlet and Neumann boundary conditions to a sphere of unit radius. The coarse, initial mesh was a cube whose faces were divided among the boundary conditions, three each. The refinement of the faces was guided by supplying a CAD sphere and the edges followed CAD curves on the sphere. The final refined mesh consisted of 15746 nodes and 15360 cells.
The boundary conditions applied on a point (x, y, z) on the sphere are:

$$\phi(x, y, z) = x + y + z \text{ if } (x, y, z) \in \Gamma_D, \tag{3.16}$$

$$\nabla\phi(x, y, z) = (1, 1, 1) \text{ if } (x, y, z) \in \Gamma_N, \tag{3.17}$$

so that on the surface of the sphere, with the normal being $\vec{n} = (x, y, z)$, ϕ and $\frac{\partial\phi}{\partial n}$ coincide.

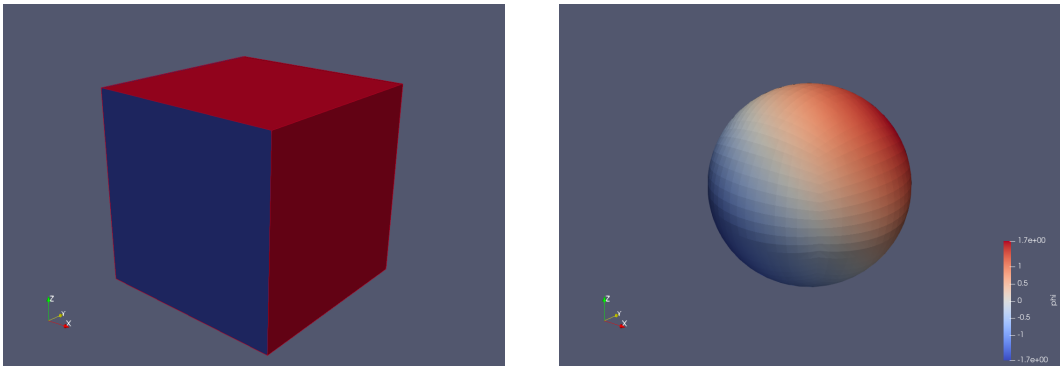

Figure 3.1: On the left, the initial coarse mesh with Γ_D in blue and Γ_N in red; on the right, the resulting ϕ on the refined mesh

Only a strong scaling on a single node was studied, as the code updates were mostly directed towards single core optimization and multithreading. The results on weak scalability from [11] would still hold qualitatively, albeit with changes following the comparison shown here. Scaling was then obtained in three scenarios:

- increasing MPI processes;
- increasing threads on a single MPI process;
- increasing threads on two MPI process, bound to different sockets.

The code was compiled against deal.II version 9.2, built from source linking against the modules available on the SISSA cluster Ulysses. The compiler was GCC 8.3.0, the MPI provider is OpenMPI 3.1.4, the thread library is Intel Thread Building Blocks in the version packaged by deal.II. Executions were carried out on a node of the `gpu1` queue, scaling from one to twenty cores, without Hyper-Threading.
The measurements were split in two cases: solution through the matrix-based "algebraic" solver and through the FMA.
It must be noted that loading and refining the mesh are inherently serial operations which are replicated in all processes. This not only degrades the scalability due to having a fixed overhead, but also implies a barrier to synchronize different MPI processes. Indeed, MPI scaling shows a steady increase of the time spent processing the mesh from 59s in the serial case to 72s at 20 processes.
All results shown are taken from the mean measurement of three runs. Scalability values are taken with the smallest of the average serial times as reference value, in order to compare against the fastest implementation.

3.4.1 Algebraic solve

For the algebraic solve process, the main activities are the assemblage of N, D matrices and the solve process through GMRES.
Of these, the assemble phase takes a prominent role and accounts for the greater portion of the total execution time. The results show the effects of the code optimization on serial execution, lowering the average time spent from 471s to 388s. Scalability through multithreading was absent in the original version and the program even shows a degradation of performance when only scaling on threads. Considering the pure MPI scaling, the revised code outperforms the original and comes close to optimal scalability. The multithreaded implementation, as discussed in the previous section, does not manage to achieve the same performance due to the impossibility of fully exploiting the vectorized initialization of matrix rows.

The actual GMRES execution only accounts for 30s of the serial time and its

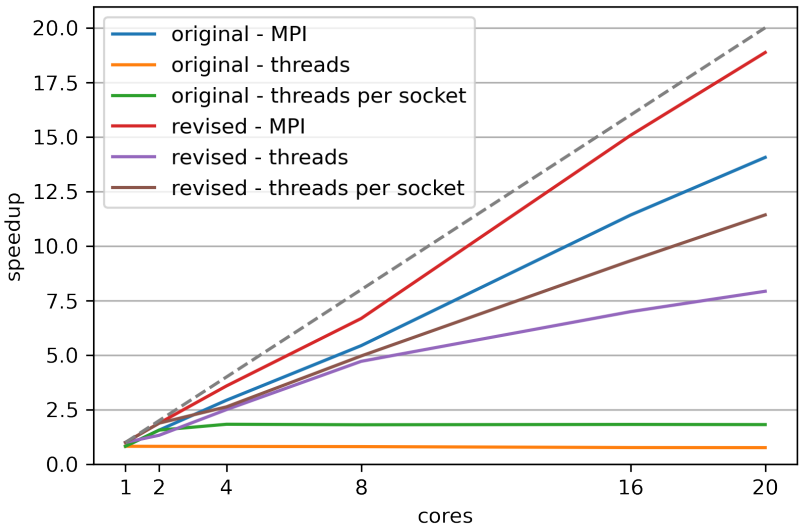

Figure 3.2: Strong scaling of the algebraic solver assemble phase

scaling behavior, aside from being underwhelming, is of limited use other than considering the importance of the preconditioner. This is due to it being heavily dependent on the node partitioning, since the banded matrix used in the ILU varies wildly unless the band size is very big.
The original implementation, scaling threads by socket, actually outperforms the revised code up to 8 cores used, but by the point when the experiment uses the full computing node, the difference between execution with at least two MPI processes is negligible.

Considering the total execution time, thus including the overhead due to mesh

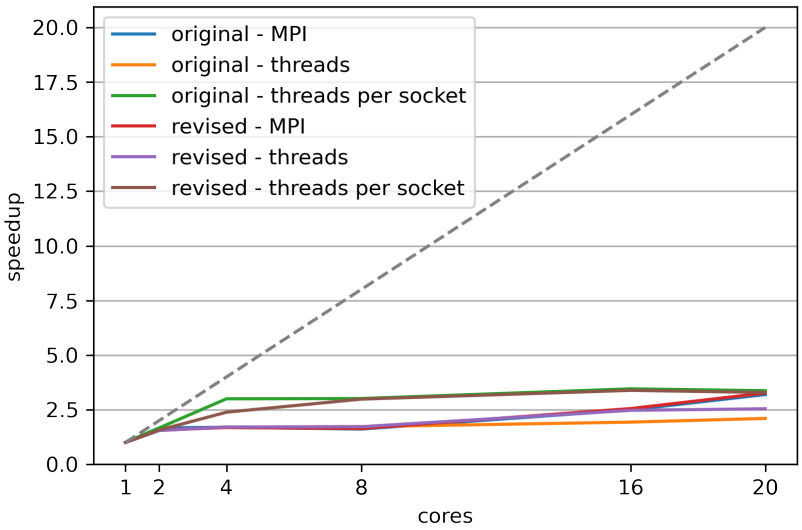

Figure 3.3: Strong scaling of the algebraic solver GMRES phase

processing, the revised code achieves the best results. However, even though the MPI executions obtained better scaling in the assemble phase, the increase in mesh processing time due to synchronization determines that the hybrid setup outperforms them on the full node.

3.4.2 Fast Multipole Approximation

The FMA solve process is composed of more tasks. Broadly speaking, we can isolate a setup phase in which the octree is constructed, the partial N, D matrices for direct interactions are assembled, the coefficients for the *far field* expansions are stored and the preconditioner is generated.
This is followed by the actual solve through GMRES, in which a number of FMA are executed with their direct interactions, ascending and descending phases.
All tasks benefit from both multithreading and distribution of the nodes across MPI processes, albeit with different results: since the octree structure is replicated and there is no inter-process communication of the expansion objects, scaling on the number of MPI processes can only be exploited with regard to the distribution of the partial N, D matrices and in reducing the *local* expansions evaluation at the nodes.

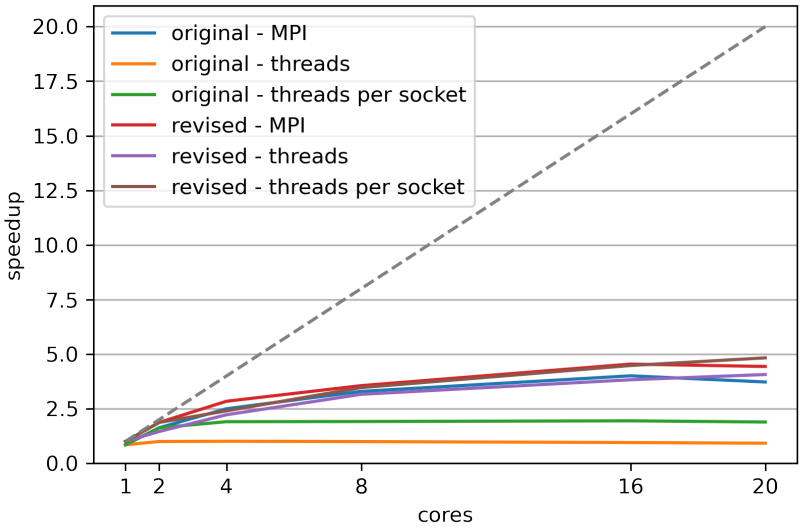

Figure 3.4: Strong scaling of the algebraic solver total execution

Let us now review in detail the behavior of the different phases.
Construction of the octree constitutes a negligible part of the total execution time, as it roughly takes 2s for both the original and revised codes, with limited scalability. The assemblage of the partial matrices takes around 38s of the serial execution, for both codes. Scalability is better when increasing the number of MPI processes, as was expected due to the analogy with the algebraic solver assemble phase. The revised code performs slightly better than the original version.

Initialization of the *far field* expansion coefficients has limited impact on the total execution time, with the original serial execution taking on average 10s, and even more so for the revised code, where it was lowered to 4s. Interestingly, MPI scaling results in a constant degradation of performance. The pure multithread executions for 2 and 4 cores perform even better than the ideal case, possibly due to increased memory bandwidth.

Initialization of the preconditioner is the more expensive of the setup operations, taking 52s in the serial executions. The scalability is not influenced by the execution setup, nor by the code revisions.

Moving to the actual ascending and descending phases, we consider the total time spent performing the matrix-vector products, which depends on the quality of the preconditioner. As was the case for the algebraic solve process, this varies when scaling on MPI processes. The results for the ascending phase follow necessarily the behavior found for the setup of the *far field* expansions coefficients discussed before. This process however has little impact on the final performance, as the

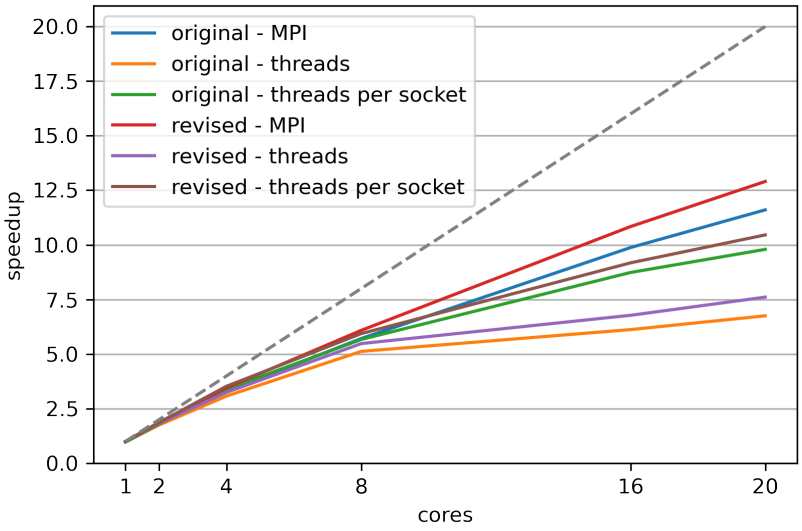

Figure 3.5: Strong scaling of the FMA solver assemble

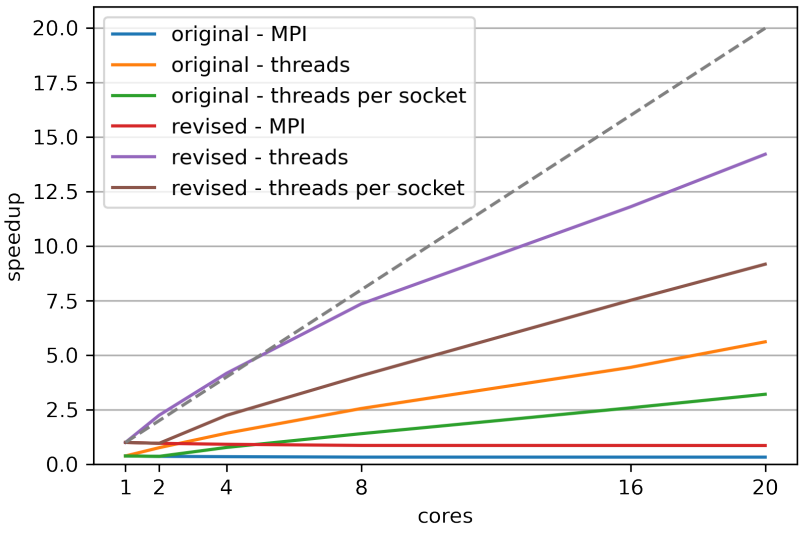

Figure 3.6: Strong scaling of the FMA solver expansion coefficients computation

serial execution only accounts for a fraction of the serial execution: 10s for the original code, 4 in the revised version.

What really defines the FMA performance is the time spent in the descending

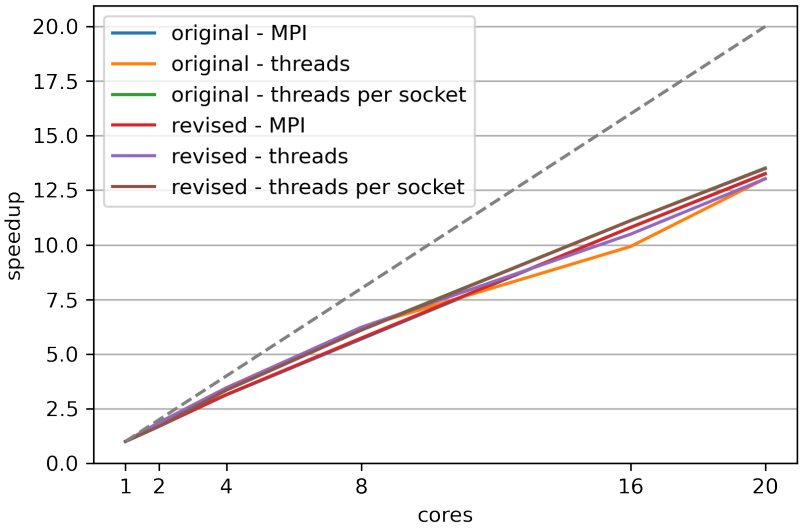

Figure 3.7: Strong scaling of the FMA solver assemble preconditioner

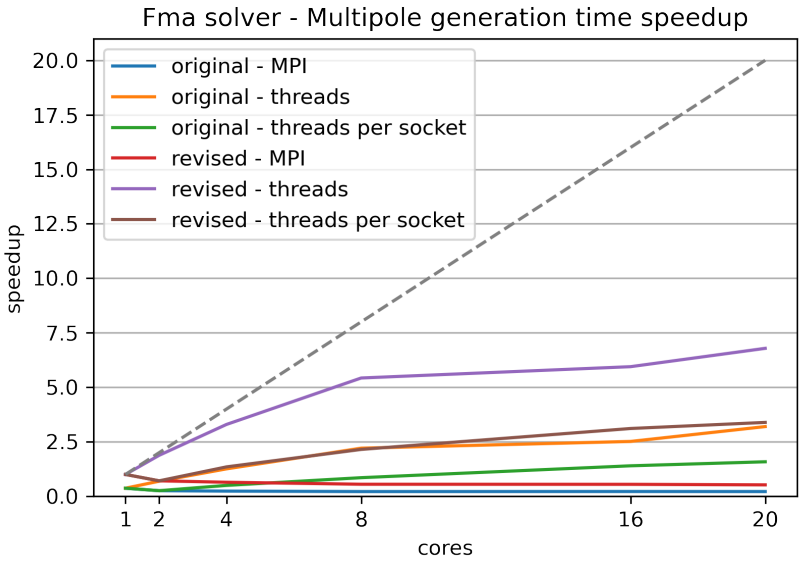

Figure 3.8: Strong scaling of the FMA solver ascending phase

phase, and more specifically the *local* expansions merge: in the serial executions, this phase takes 842s and 221s for the original and revised code, respectively. The large performance gain is due to the changes in the translation primitive of the

merge routine, which achieved a great reduction of the number of complex number products.
It must be observed that this optimization of the serial code does not translate to a better scaling per se, and indeed the plot is unfair to the original code: the scaling curves that would be obtained by using the original code serial execution as reference would closely follow the ones presented here for the revised code, if not even overcoming them.

Considering the scaling of the total execution time, scalability is worse than the

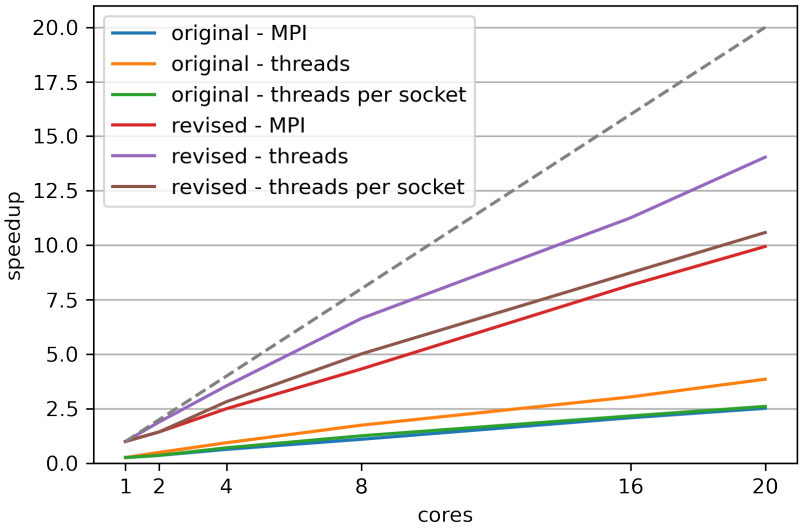

Figure 3.9: Strong scaling of the FMA solver *local* expansions merge and evaluation

algebraic solver case due to mesh processing representing a larger portion of the total time. As was expected following the results on the descending phase, the best setup for the FMA solver procedure is obtained from the pure multithread execution.

Overall, the benchmark problem shows that the revised code consistently improves on the original in all configurations. For the problem at hand, for the revised code the FMA solver outperforms the algebraic one both in serial execution and using the maximum threads on the node. However, the algebraic solver shows a better scaling profile and can exploit more efficiently the MPI communications. On the full node, full MPI configuration, the algebraic solver completes in 130s while the FMA solver takes 190s; while using only threads, the timings are 142s and 99s respectively. In the hybrid scaling configuration, the two solvers performed similarly with 119s and 118s respectively.

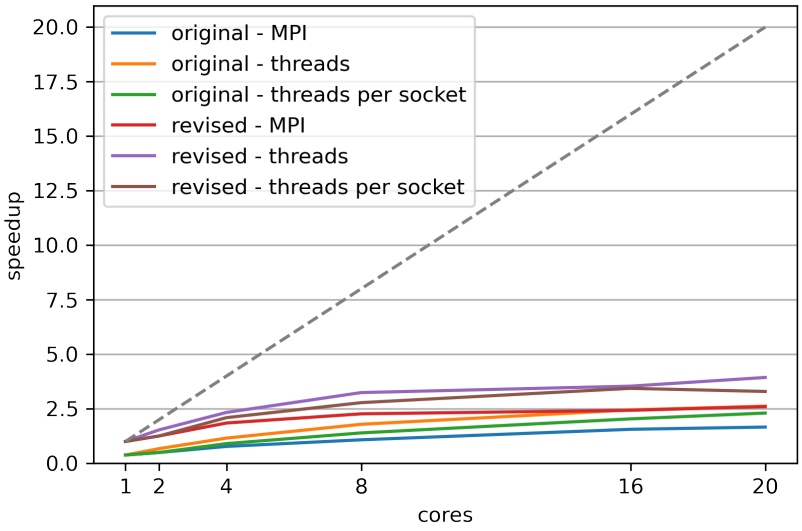

Figure 3.10: Strong scaling of the FMA solver total execution

3.5 Complex-valued problem benchmarks

The behavior of the revised code was investigated on a complex-valued problem involving a Robin boundary condition. The same mesh as the previous experiment was used, but the Neumann boundary condition was switched to a Robin one with complex-valued coefficients.
The boundary conditions applied on a point (x, y, z) on the sphere are:

$$\phi(x, y, z) = (1 - j)(x + y + z + 2) \text{ if } (x, y, z) \in \Gamma_D, \tag{3.18}$$

$$\phi(x, y, z)\frac{1 + j}{2(x + y + z + 2)} + \frac{\partial \phi}{\partial n}(x, y, z) = (1 - j)(x + y + z) + 1 \text{ if } (x, y, z) \in \Gamma_R, \tag{3.19}$$

so that the solution for imposes

$$\phi(x, y, z) = (1 - j)(x + y + z + 2) \ (x, y, z) \in \Gamma, \tag{3.20}$$

$$\frac{\partial \phi}{\partial n} = (1 - j)(x + y + z) \ (x, y, z) \in \Gamma. \tag{3.21}$$

Executions were carried out on the same setup as the previous benchmarks.

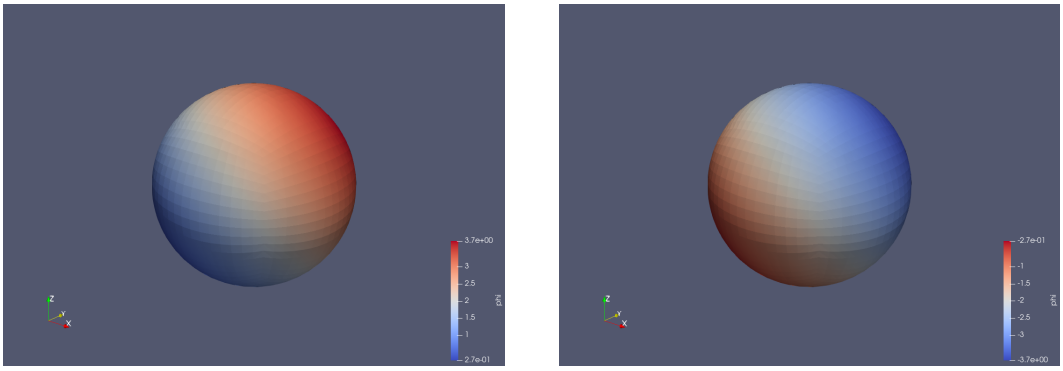

Figure 3.11: On the left, the solution $\mathbf{Re}[\phi]$ on the refined mesh; on the left, $\mathbf{Im}[\phi]$

3.5.1 Algebraic solve

For the algebraic solve process, the assemble phase had no change in implementation and performed the same as the previous benchmark. The linear operator changes show directly in the GMRES solve phase, where the expected slowdown by a factor of around 2 is manifest; the extra linear vector operations are shadowed by the quadratic matrix-vector product cost. There is a slight increase in the number of iterations of the solver, from 50 to 55, and variability due to the preconditioner sensitivity to the node partitioning. Overall, the scalability of the previous benchmark is retained well.

Considering the total execution time, thus including the overhead from the additional data structures, the slowdown behavior trends again to a flat value, slightly above 1. While some variability is still present at the extremes of the cluster node utilization, since the larger part of the execution is spent in the assemble phase, the total execution time is very close to the purely real case, with the same scalability.

3.5.2 Fast Multipole Approximation

The main difference we expect between the FMA and the algebraic solve is that the most time consuming procedure lies in the assemble phase for the former, and in the *local* expansion evaluations for the latter. The changes to the unknown vectors, as induced by the Robin boundary condition implementation chosen, are not expected to reflect on the algebraic matrix-vector products, since the matrices are full. Instead, the harmonic series expansion manipulations are sensitive to the value patterns of the operand vectors, as the expansions of a cell which contains no charges have no effect on other cells, and are thus skipped. In the benchmark problem chosen, half the nodes are subjected to a Dirichlet boundary condition,

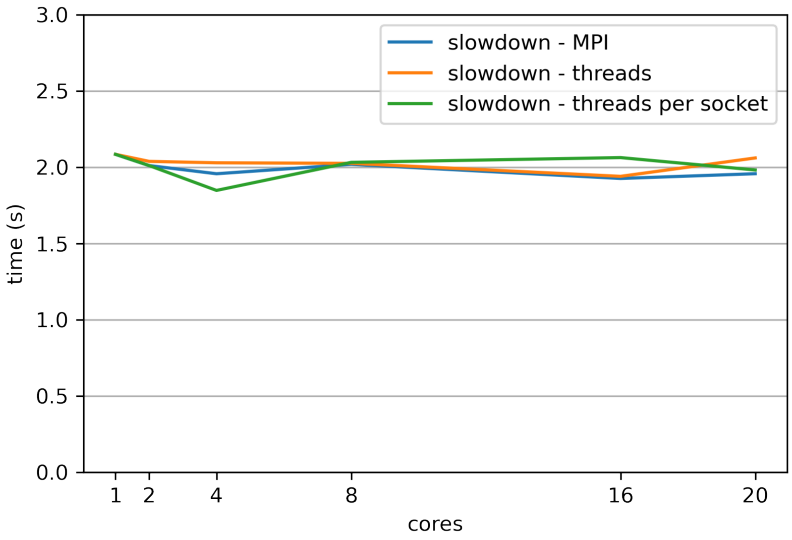

Figure 3.12: Strong scaling of the algebraic solver GMRES phase slowdown, for a complex-valued mixed Dirichlet Robin problem

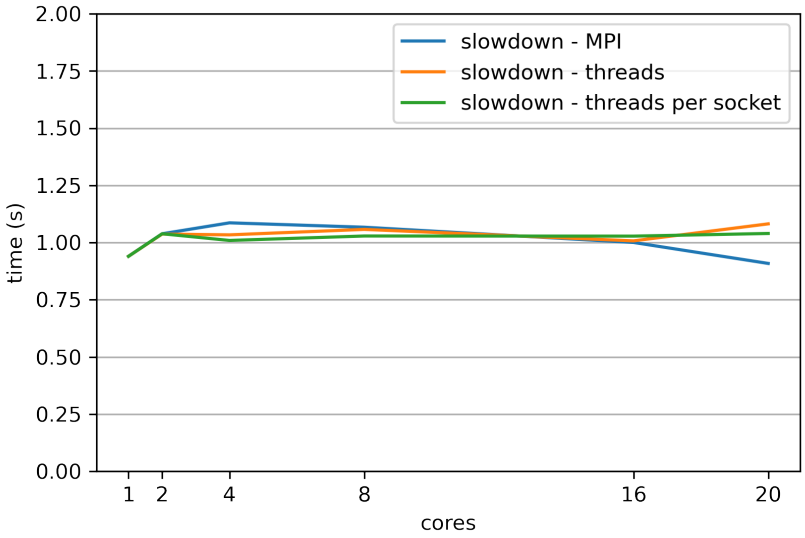

Figure 3.13: Strong scaling of the algebraic solver total execution slowdown, for a complex-valued mixed Dirichlet Robin problem

the other half to the Robin boundary condition: then, the unknown $\hat{\phi}$ does not contain any element fixed to zero and the term $D(\tilde{\gamma} - R\tilde{\phi}_R)$ effectively requires a matrix-vector product where the vector is half full.
In light of this, it is expected that the cost of approximating the N matrix-vector product is double that of the purely real, mixed Dirichlet Neumann case. Moreover, since the operation is repeated for the real and imaginary parts of the unknown vectors, the *far field* and *local* expansions evaluation cost will triplicates in the worst case. Indeed this appears to hold: on average, the current benchmark registers a cumulative time of 648s for the descending phase, in line with triplicating the corresponding time of 221s from the purely real case. The ascending phase instead registers a larger slowdown in the distributed executions, which is to be imputed to the interprocess communications of the nodal values vectors; the effecto on the total execution, however is very limited, as the time spent in this phase was still under 5s at worst.
The increase in GMRES iterations is slim, from 32 to 34, and appears to be absorbed by the actual expansions configuration.
As for the other procedures, no appreciable changes in execution time are observed, since the overhead from the additional vector operations in the GMRES solve, due to the complex-valued implementation, are dominated by the ascending and descending phases anyway.
Scalability results are in line with the previous benchmark. The slowdown is maximum for the serial execution, at slightly above 2, and shows a decreasing trend. This is due to the mesh processing time not scaling, and having a larger effect as the machine utilization increases.

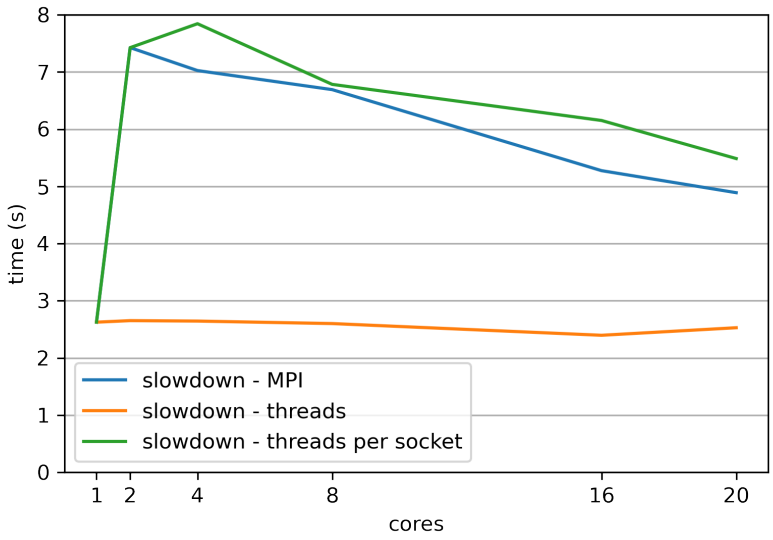

Figure 3.14: Strong scaling of the FMA solver ascending phase slowdown, for a complex-valued mixed Dirichlet Robin problem

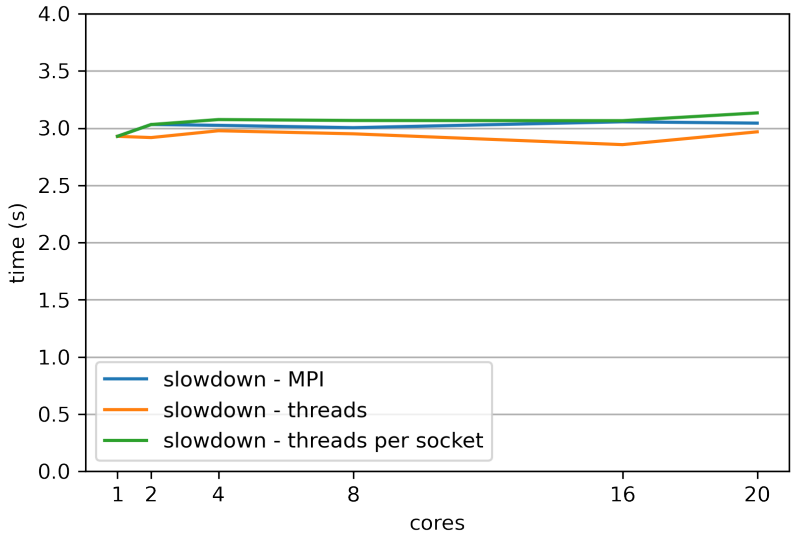

Figure 3.15: Strong scaling of the FMA solver *local* expansions merge and evaluation slowdown, for a complex-valued mixed Dirichlet Robin problem

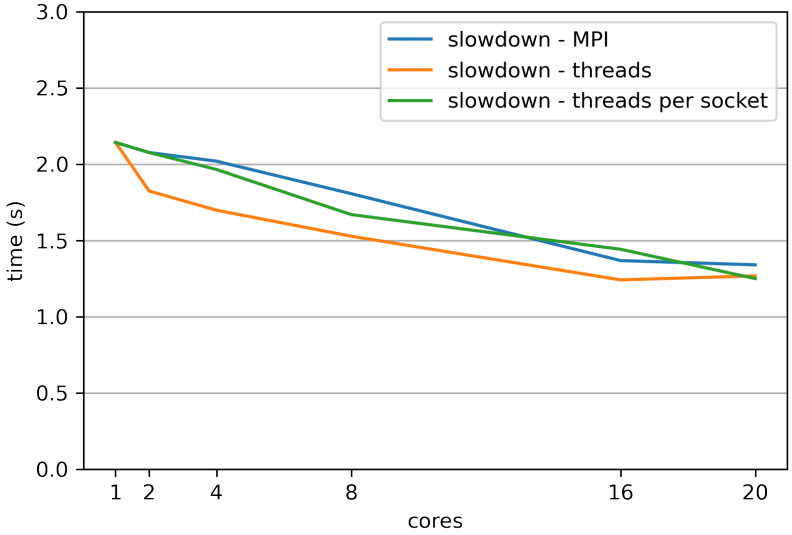

Figure 3.16: Strong scaling of the FMA solver total execution slowdown, for a complex-valued mixed Dirichlet Robin problem

Chapter 4

Case study

4.1 Introduction

With the general problem formulations from the first chapter and thanks to the extensions to the π-BEM library, it is now possible to implement a procedure for the retrieval of the coefficients to use in the equations of motion.
This chapter will present the application of the methodology to a simple semi-submerged sphere in the zero-speed case, and the resulting heave added mass and damping coefficients will be compared with the theory from Havelock.
After a discussion of the preliminary results and the issues encountered, an alternative treatment is proposed in order to overcome the limitations found.

4.2 Initial problem setup

The problem studies the behavior of a sphere of unit radius, semi-submerged so that its baricenter is placed on the still surface of the fluid, with zero speed.
In order to retrieve the coefficients of the equations of motion for a single wave frequency, a single parameter file is generated, specifying the full set of 7 complex-valued potential problems.
The parameter file is fed to a modified version of the π-BEM Driver class, which after the solution of the individual potentials, performs integration on the hull and outputs the added mass and damping coefficients matrices.
After repeating the process for multiple wave frequencies, the outputs are collected and analysed in order to express the coefficients as functions of the frequency, using nondimensionalized quantities.
The fluid is contained in a cylinder of large dimension with respect to the sphere's radius, according to the figure 1.2. Our initial mesh will have height 40 and radius

20.
The coarse mesh is refined according to a set of CAD surfaces and curves, taking care that at least 5 cells are cover a length λ extending from the body. The basic, uniform refinment of π-BEM was extended in order to specify different refinement levels on each boundary, in order to be able to increase the Γ_{free} panel count without the same increase Γ_{bottom} or Γ_{wall}.

All potentials are subject to the Laplace equation in the fluid domain, with the

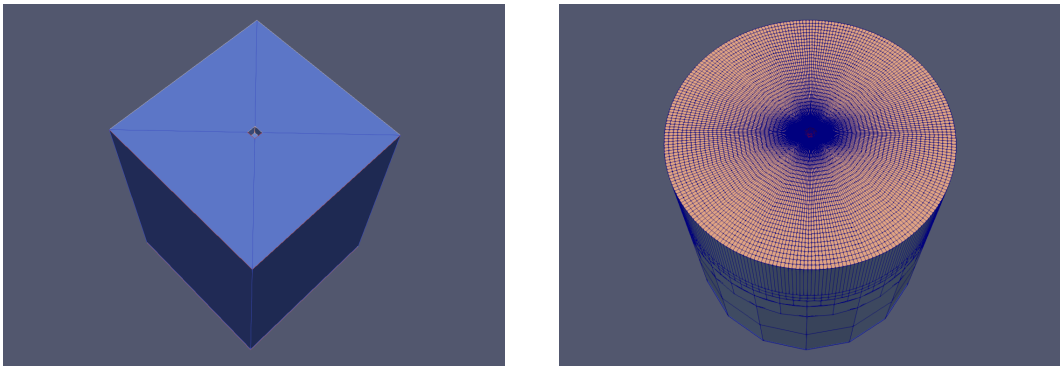

Figure 4.1: On the left, the initial coarse mesh; on the right, a possible result of non-uniform refinement

boundary conditions being determined by the environment: we have $U = 0$, $\beta = 0$ since the sphere is symmetric, the $\omega_e = \omega_0$ changes for each set of problems and sets in turn k_0 due to the dispersion relation 1.12.
The steady wave-making potential can be ignored and the boundary conditions assume the simpler form: On Γ_{free}, due to the normal pointing towards the fluid:

$$\omega_e^2 \phi_D + g \frac{\partial \phi_D}{\partial n} = 0, \tag{4.1}$$

$$\omega_e^2 \phi_i + g \frac{\partial \phi_i}{\partial n} = 0 \text{ for } i = 1, .., 6. \tag{4.2}$$

On Γ_{hull}:

$$\frac{\partial \phi_D}{\partial n} = -\frac{\partial \phi_I}{\partial n}, \tag{4.3}$$

$$\frac{\partial \phi_i}{\partial n} = -j\omega_e n_i \text{ for } i = 1, .., 6, \tag{4.4}$$

where ϕ_I is defined as in 1.11 with unit amplitude $a_w = 1$, and n is the generalized normal vector defined as in 1.19, 1.20.
On Γ_{wall}, the Sommerfeld radiation conditions are:

$$\frac{\partial \phi_D}{\partial n} - jk_0 \phi_D = 0, \tag{4.5}$$

$$\frac{\partial \phi_i}{\partial n} - jk_0\phi_i = 0 \text{ for } i = 1, .., 6. \tag{4.6}$$

On Γ_{bottom}, the non-penetrating conditions give:

$$\frac{\partial \phi_D}{\partial n} = 0, \tag{4.7}$$

$$\frac{\partial \phi_i}{\partial n} = 0 \text{ for } i = 1, .., 6. \tag{4.8}$$

Once all the ϕ_i have been found, integration on Γ_{hull} will give the wave-induced forces and moments amplitudes and added mass and damping coefficients, which after simplification result in:

$$\overline{F}_i = -\rho \int_{\Gamma_{hull}} n_i j\omega_e(\phi_I + \phi_D)ds, \tag{4.9}$$

$$A_{ik} = \frac{\rho}{\omega_e} \int_{\Gamma_{hull}} n_i \,\mathrm{Im}[\phi_k]ds, \tag{4.10}$$

$$B_{ik} = -\rho \int_{\Gamma_{hull}} n_i \,\mathrm{Re}[\phi_k]ds \tag{4.11}$$

for $i, k = 1, .., 6$.

Only the results on A and B will be analysed here; nondimensionalized measures are taken by considering the volume of the displaced fluid $\nabla = \frac{4}{3}\pi r^3$ and for the damping, dividing also by the frequency ω_e. Moreover, the nondimensionalized coefficients are plotted against the nondimensionalized frequency $\omega = \omega_e^2 2r/g$.

$$A'_{ik} = \frac{A_{ik}}{\rho\nabla}, \tag{4.12}$$

$$B'_{ik} = \frac{B_{ik}}{\rho\nabla\omega_e}, \tag{4.13}$$

for $i, k = 1, .., 6$. The problem sets are generated so that the sampling on ω is evenly spaced; we also note that, from a practical standpoint,

The coefficients in position $3, 3$, which couple vertical wave-induced forces and vertical motion, are common benchmarks for the correctness of these codes. Analytically approximated results were given in [15] through truncated series.

The nondimensionalized added mass coefficient has a limiting value of 0.828 at $\omega_e = 0$ and increasing the frequency, grows to 0.88, decreases to a minimum of 0.38 and slowly rises to an asymptotic value, which Havelock speculates to be 0.5 . The damping coefficient starts instead at 0, reaches a maximum of 0.35 and finally descends to 0.

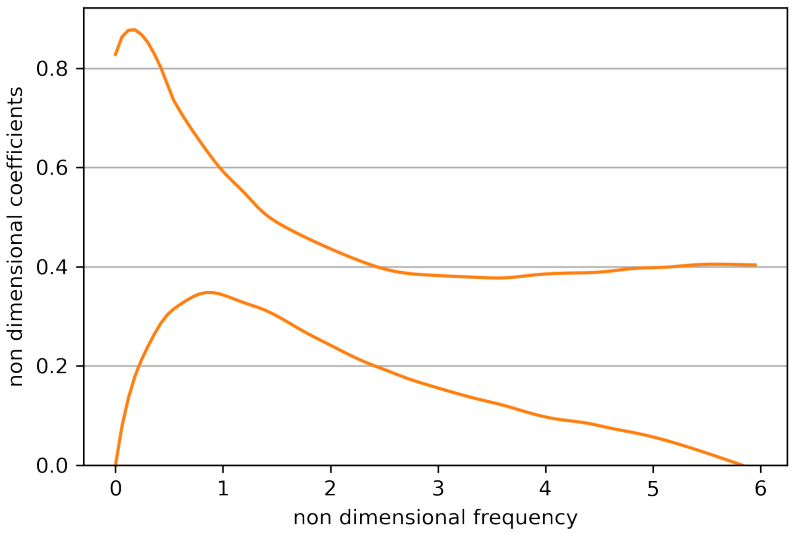

Figure 4.2: The nondimensionalized added mass and damping coefficients for a semi-submerged sphere as determines by Havelock

4.3 Experiments and reworks

The results from our solver, which used the algebraic solve process, present heavy oscillations, but nonetheless follow the trend of Havelock's curves. The tolerance used for GMRES was 10^{-10}.
The experiment was stopped before reaching $\omega = 2$, as the refinements required to have a sufficiently small panels on Γ_{free} brought the nodes count to 10442 and the problem set solve time to 2112s on the development machine. Increasing ω would have almost quadrupled the dofs and increased by a factor of 16 the matrix-vector product cost. It could have warranted for a switch to the FMA, but the real issue would have been with the GMRES not converging in a timely manner due to the degradation of the condition number for the linear operator.
With regard to the individual problems difficulty, as is somehow expected due to the symmetries of the sphere, the surge, sway and heave potentials take more iterations to converge, about double those required by the problems for the roll, pitch and yaw potentials. Solution of the diffraction potential takes only slightly more than the first three problems.

Moreover, we speculate that the oscillations are due to a combination of the Sommerfeld boundary condition on Γ_{wall} and a poor organization of the Γ_{free} tessellation.
Reducing the radius of the tank allows to increase the limit on ω, albeit with

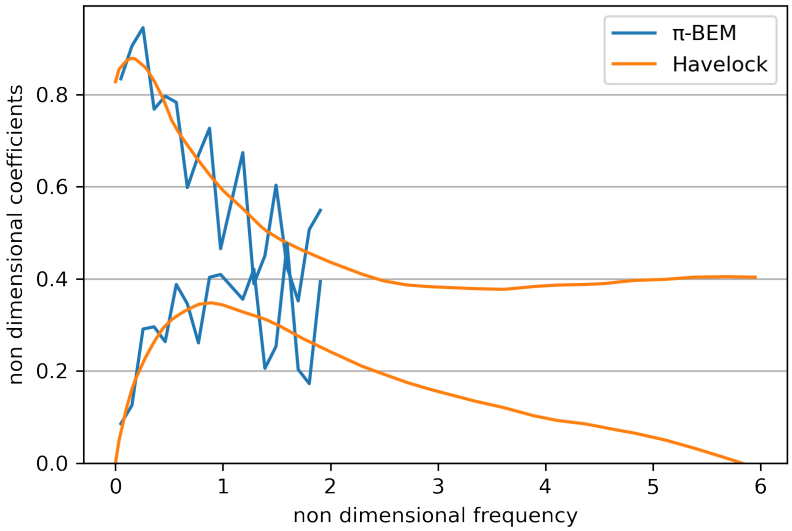

Figure 4.3: The nondimensionalized added mass and damping coefficients for a semi-submerged sphere computed on a cylinder of radius 20

limited success: the mesh now requires a lower number of refinements in order to accommodate the radiation λ in the tessellation. Since in this investigation the steady wave-making potential ϕ_S is ignored, the mesh is not required to have Γ_{wall} at a large distance from the hull.
Indeed, halving the tank radius moves the computational limit to around $\omega = 3.25$. The behavior of the solution near $\omega = 0$ is worse than with the larger mesh, suggesting that the very long radiation wavelength induced by a small frequency is not captured well enough by the nodes of Γ_{free}. We also observe that the oscillation appear to have a larger period compared to those from the larger tank.

Following the consideration on the tessellation of Γ_{free}, we experimented with a different refinement strategy: instead of halving both dimensions of the cells, we only cut along the radii from the hull center. This procedure allows to maintain a sufficiently high number of panels covering the radiated wavelength, but does not waste resources along the radiated isolines.

Indeed, this new strategy proves extremely effective: the largest mesh only contains 3546 nodes and maintains the same solution quality found previously, with a problem set solve time of only 295.6s.

The underwhelming behavior on the smaller tank for low frequency persists, so we abandon it. We rather investigate a naive smoothing strategy: by interpolating the results from π-BEM with the CubicSpline class from SciPy [31], we find the average oscillation period and apply a moving average using the window.

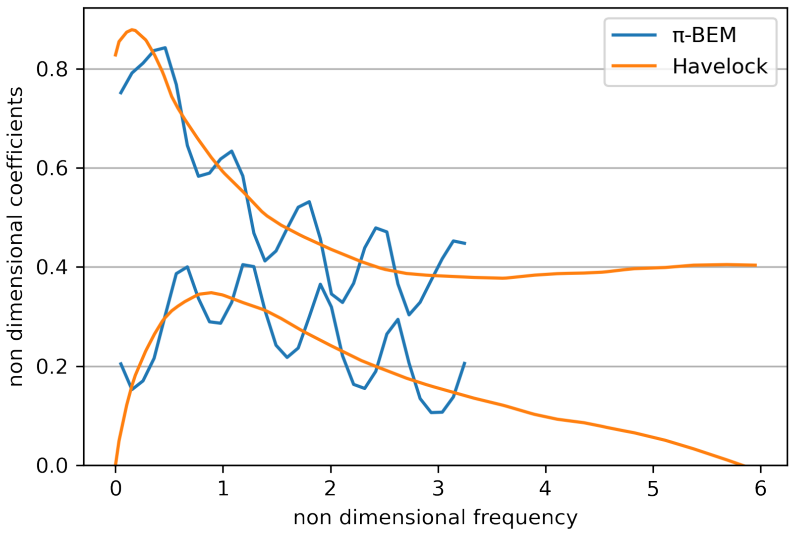

Figure 4.4: The nondimensionalized added mass and damping coefficients for a semi-submerged sphere computed on a cylinder of radius 10

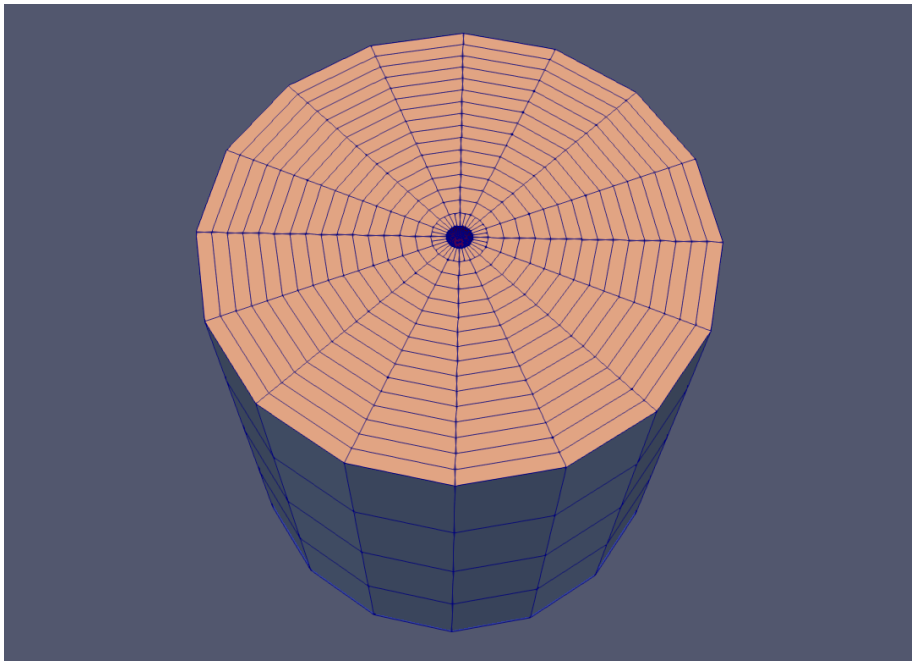

Figure 4.5: A mesh with the free surface refined along the radii

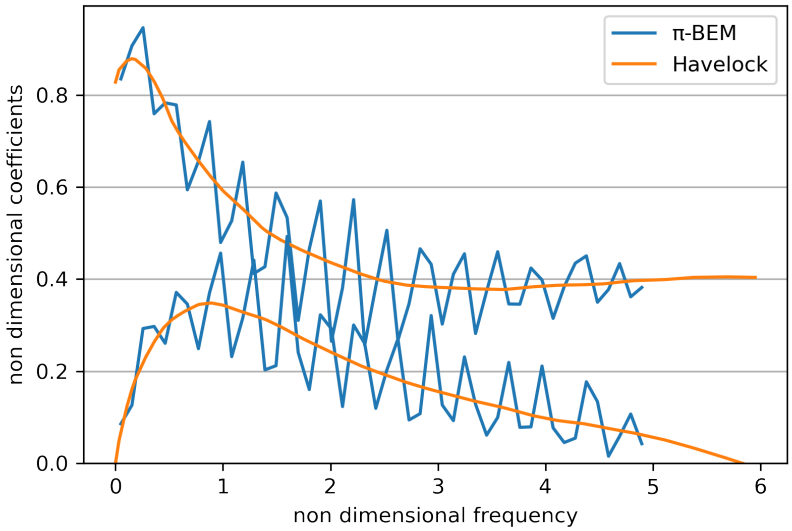

Figure 4.6: The nondimensionalized added mass and damping coefficients computed on a cylinder of radius 20, radial refinement

Indeed the smoothed curves are close to Havelock's results, but irregularities are still present on a large part of the sampled interval.

We now investigate the effect of increasing the order of elements: this determines a setback on the benefits of the radial refinement, as even though one refinement step can now be omitted, the placement of nodes dictates that the non-radial edges are effectively refined.

The results show that the higher order elements consistently decrease the oscillations amplitude, and the smoothed curves are very close to the theoretical solutions. However, the irregularities appear to increase in amplitude for frequencies above 2 and on some intervals, the chosen sampling seems to not be able to properly capture the solution behavior. The larger spikes are reminiscent of the irregular frequencies cited by [17], which other codes treat with so-called "lid-panels", additional cells placed as a continuation of Γ_{free} to cover the hole pierced by the hull.

The process of refining the mesh to accommodate the radiated wavelength gives good results, after post processing, but still requires a large number of nodes, or higher order elements, to be able to properly extract convincing results.

An alternative approach would be to first estimate an optimal relationship between the mesh geometry and the wavelength, then construct a sequence of ad-hoc meshes respecting said relation, one for each ω, without changing the number of nodes.

We opted for fixing the ratio between the wavelength and the distance between the hull and Γ_{wall} to 3, and forced 4 rounds of radial refinement on Γ_{free} in order

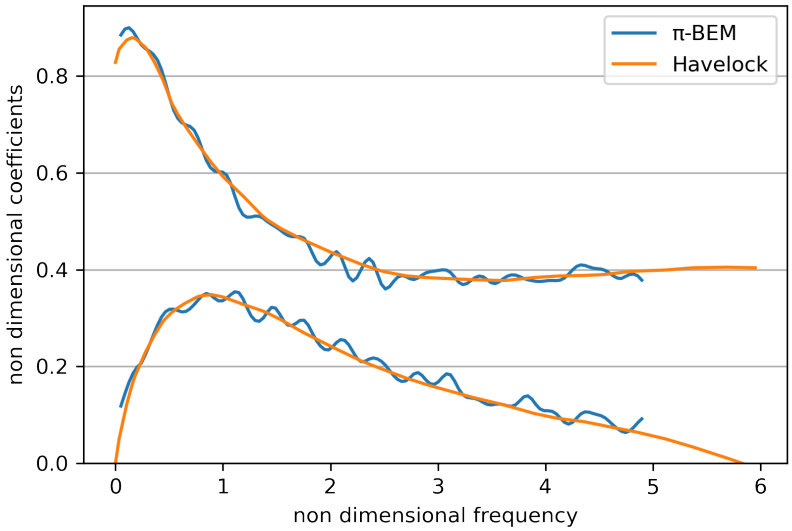

Figure 4.7: The nondimensionalized added mass and damping coefficients computed on a cylinder of radius 20, radial refinement, moving average smoothing

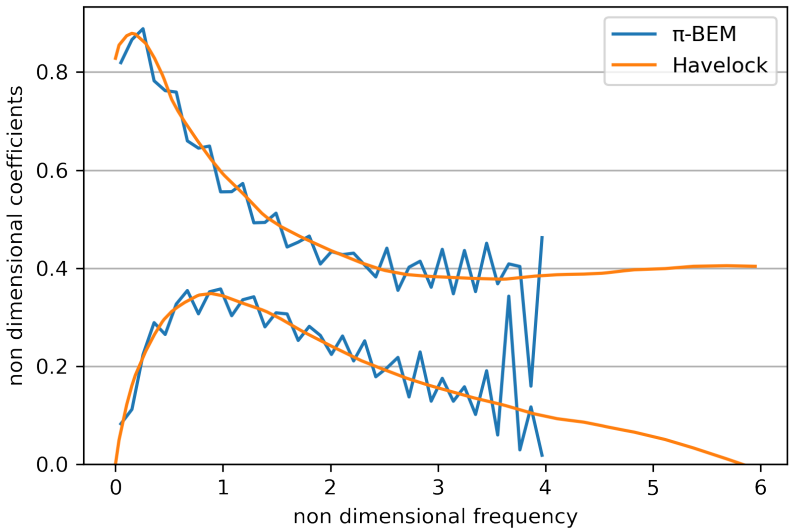

Figure 4.8: The nondimensionalized added mass and damping coefficients computed on a cylinder of radius 20, Q2 elements, radial refinement

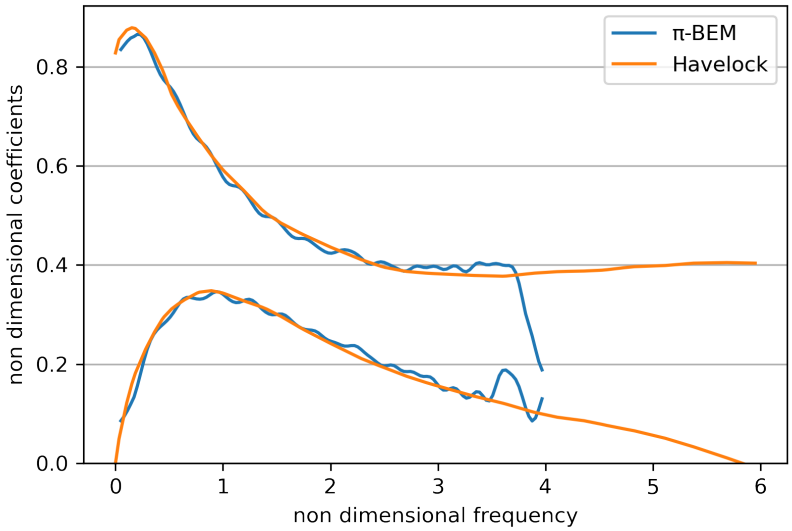

Figure 4.9: The nondimensionalized added mass and damping coefficients for a semi-submerged sphere computed on a cylinder of radius 20, Q2 elements, radial refinement, moving average smoothing

to respect the constraint described before. Although this is somewhat of a magic number, it comes from the empirical observation of the results seen so far.
The solver program was again extended to specify in the parameter file the mesh elements to be scaled by a given factor, and the sequence of problems were generated accordingly.
The results follow qualitatively the desired behavior, although with some deviation

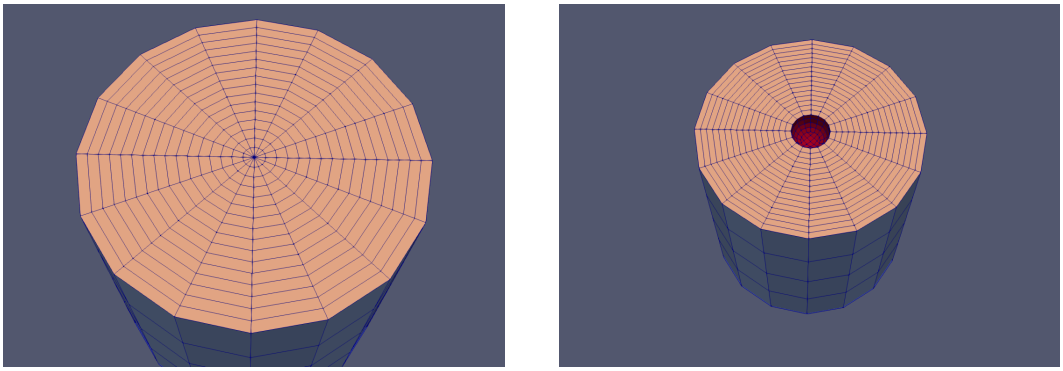

Figure 4.10: On the left, the adaptive mesh for $\omega_e = 0.86$; on the right, the adaptive mesh for $\omega_e = 2.05$, both with Γ_{free} refined along the radii

trends. At the lower frequencies, the added mass coefficient shows a discontinuity and underestimates in value; the tendency is towards overestimating between $\omega = 0.5$ and 3, and the asymptotic behavior seems to tend to 0.35, which is again under the expected result. The damping coefficient follows the target closely until $\omega = 1$, after which decreases much slower than the theory.

Increasing the element order, we observe that the discontinuity for the added

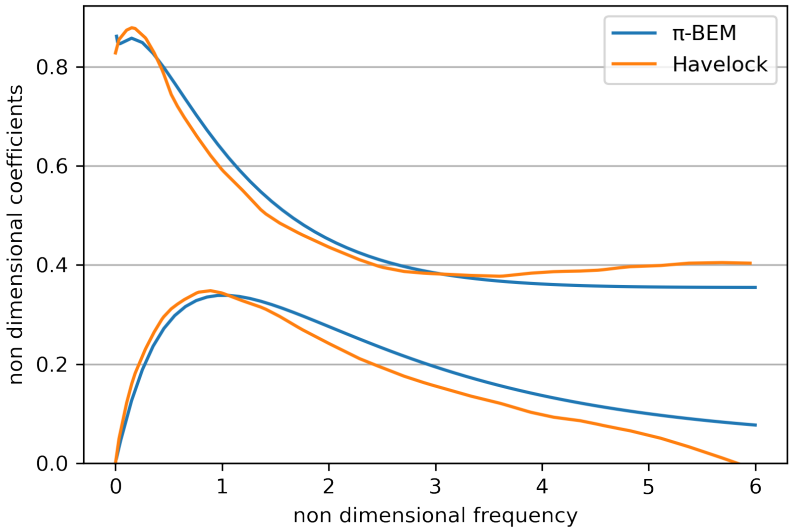

Figure 4.11: The nondimensionalized added mass and damping coefficients computed on the adaptive mesh, linear elements, radial refinement

mass coefficient is more severe, and the overestimation continues up to around $\omega = 4.5$. From that point on, albeit with some localized instability, the agreement with the theory is very good. For the damping coefficient, the pattern seen before is repeated but difference with the theory is much lower; the same instability found for the added mass coefficient show up, and the decrease to zero is still slower than the theory, but faster than the previous experiment.

Finally, we exploit the reduced solution cost to increase the element order again. The irregularity at lower frequency is exacerbated, but at higher values of ω the results are smooth. However, the deviation from the theory is identical to the previous experiment, suggesting this is the limit of the adaptive mesh for the radius-wavelength ratio adopted.

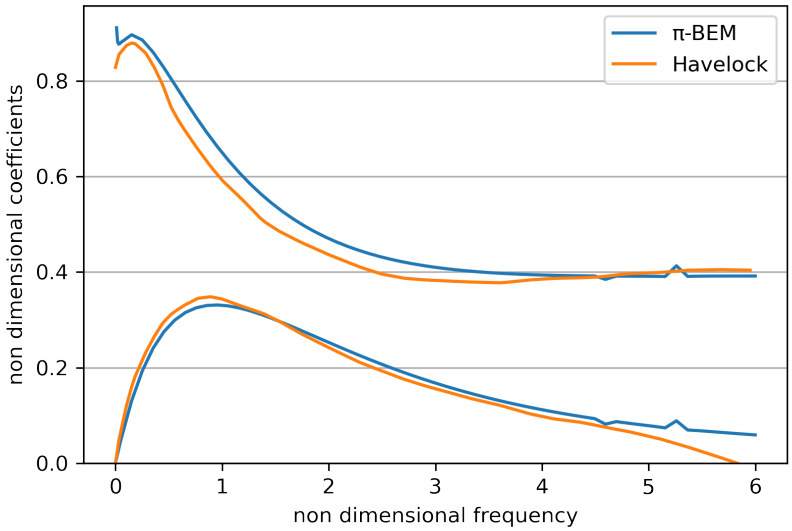

Figure 4.12: The nondimensionalized added mass and damping coefficients computed on the adaptive mesh, Q2 elements, radial refinement

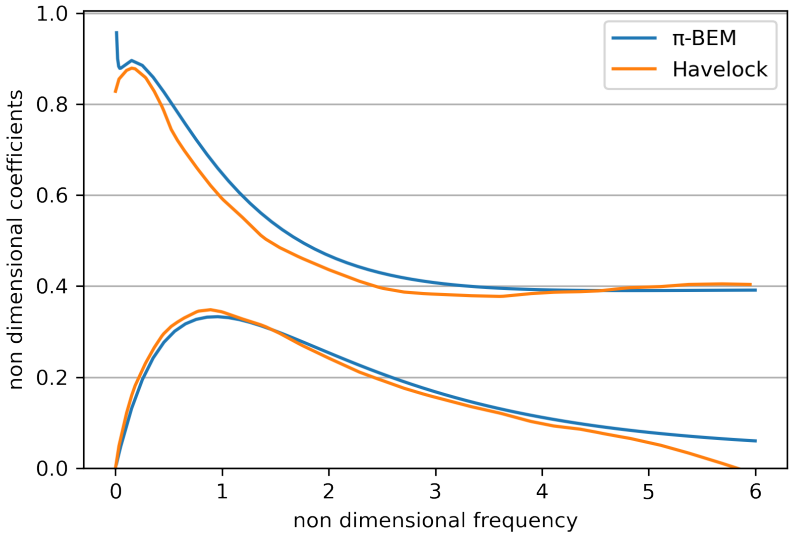

Figure 4.13: The nondimensionalized added mass and damping coefficients computed on the adaptive mesh, Q3 elements, radial refinement

Chapter 5

Conclusions

This work provided the initial steps for the implementation of the offline phase of a linear seakeeping pipeline based on the π-BEM solver, and was carried out in the context of the Winning a Sea State project in collaboration with Cetena.
From the analysis of the theoretical background of the model and the inner working of the library, seamless implementations were devised for Robin and linearized free surface boundary conditions, efficient solution of multiple problems sharing the same geometry, and support for complex-valued formulations.
The enhanced performances of the revised library were compared against the original in a strong scaling benchmark. The good scaling behavior was retained when measured on a problem featuring complex-valued potential and Robin boundary conditions, where the comparison of actual execution times showed agreement with the theoretical estimates. For the algebraic solve process, the linear operator takes roughly double the time of a purely real case, while the Fast Multipole Approximation depends on the nodes partitioning between the different boundary conditions.
Experimenting on the retrieval of the added mass and damping coefficients for a semi-submerged sphere in the zero-speed case showed the versatility of π-BEM and the deal.II framework, allowing to easily implement alternative strategies in the mesh refinement process with the aim to decrease computational costs while retaining solutions quality.
The issue of parasitic oscillations in the solutions were linked to the relationship between the free surface tessellation, the basin walls distance from the ship hull, and the radiated wavelength, as an effect of the Sommerfeld radiation condition applied on the walls.
Refinement of the free surface cells along the radii allowed to retain the solution quality of highly refined meshes with a lower node count. This allowed to increase the order of the elements, with the effect of reducing the amplitude of the oscillations.
A naive smoothing procedure, based on the estimation of the oscillation period

with a purely data-driven procedure, and its removal through a moving average filter, retrieved coefficients very close to the theory.
An alternative strategy, exploiting the hypothesis of the existence of an optimal relationship between wave frequency and mesh geometry, showed promise in the retrieval of qualitatively good results, by substituting the refinement of the mesh with a suitable scaling operation on the mesh.
Although the development effort shifted to the online phase of the motions prediction, the work completed so far constitutes a promising basis for an efficient, scalable, free open source framework.

Bibliography

[1] Daniel Arndt, Wolfgang Bangerth, Bruno Blais, Marc Fehling, Rene Gassmöller, Timo Heister, Luca Heltai, Uwe Köcher, Martin Kronbichler, Matthias Maier, Peter Munch, Jean-Paul Pelteret, Sebastian Proell, Konrad Simon, Bruno Turcksin, David Wells, and Jiaqi Zhang. 2021, accepted for publication. The deal.II Library, Version 9.3. *Journal of Numerical Mathematics* (2021, accepted for publication). https://dealii.org/deal93-preprint.pdf

[2] Daniel Arndt, Wolfgang Bangerth, Denis Davydov, Timo Heister, Luca Heltai, Martin Kronbichler, Matthias Maier, Jean-Paul Pelteret, Bruno Turcksin, and David Wells. 2021. The deal.II finite element library: Design, features, and insights. *Computers & Mathematics with Applications* 81 (2021), 407–422. https://doi.org/10.1016/j.camwa.2020.02.022

[3] Rodrigo Azcueta. 2002. RANSE Simulations for Sailing Yachts Including Dynamic Sinkage & Trim and Unsteady Motions in Waves. https://doi.org/10.3940/rina.ya.2002.17

[4] C. L. Bretschneider. 1952. The generation and decay of wind waves in deep water. *Transactions, American Geophysical Union* 33, 3 (1952), 381. https://doi.org/10.1029/tr033i003p00381

[5] Cetena. 2021. Cetena Website. https://www.cetena.it/en/

[6] James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. 19, 90 (1965), 297–301. https://doi.org/10.1090/s0025-5718-1965-0178586-1

[7] Cooperative Research Ships. 2021. Cooperative Research Ships Website. https://www.crships.org/

[8] W.E. Cummins and David Taylor Model Basin. 1962. *The Impulse Response Function and Ship Motions*. Navy Department, David Taylor Model Basin. https://books.google.se/books?id=GLLANwAACAAJ

[9] The Epetra Project Team. 2021. *The Epetra Project Website.* https://trilinos.github.io/epetra.html

[10] Fabio Fucile. 2015. *Deterministic sea wave and ship motion forecasting: from remote wave sensing to precition error assessment.* Ph.D. Dissertation. Università degli studi di Trieste.

[11] Nicola Giuliani. 2015. Towards exascale BEM simulations: hybrid parallelisation strategies for boundary element methods. (2015). http://urania.sissa.it/xmlui/handle/1963/35164

[12] Nicola Giuliani, Andrea Mola, and Luca Heltai. 2018. π - BEM : A flexible parallel implementation for adaptive, geometry aware, and high order boundary element methods. *Advances in Engineering Software* 121 (jul 2018), 39–58. https://doi.org/10.1016/j.advengsoft.2018.03.008

[13] Nicola Giuliani, Andrea Mola, Luca Heltai, and Luca Formaggia. 2015. FEM SUPG stabilisation of mixed isoparametric BEMs: Application to linearised free surface flows. *Engineering Analysis with Boundary Elements* 59 (Oct. 2015), 8–22. https://doi.org/10.1016/j.enganabound.2015.04.006

[14] Leslie Greengard and Vladimir Rokhlin. 1997. A fast algorithm for particle simulations. *J. Comput. Phys.* 135, 2 (Aug. 1997), 280–292. https://doi.org/10.1006/jcph.1997.5706

[15] Thomas Havelock. 1965. *The collected papers of Sir Thomas Havelock on hydrodynamics.* Office of Naval Research, Department of the Navy.

[16] 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. 2005. An overview of the Trilinos project. *ACM Trans. Math. Software* 31 (2005), 397–423.

[17] O.G. Hizir. 2015. *Three Dimensional Time Domain Simulation of Ship Motions and Loads in Large Amplitude Head Waves.* University of Strathclyde. https://books.google.it/books?id=4uafnQAACAAJ

[18] Manoj Kumar, V. Ananthasubramanian, and S. P. Singh. 2008. Prediction of Heave, Pitch and Roll Ship Motions in Waves Using RANSE Method. In *Volume 5: Materials Technology CFD and VIV.* ASMEDC. https://doi.org/10.1115/omae2008-57479

[19] H. Lamb. 1932. *Hydrodynamics.* Cambridge University Press.

[20] E.V. Lewis. 1989. *Principles of Naval Architecture , Second Revision.* SNAME.

[21] Andrea Mola, Michael Madigan, Leigh Mccue-Weil, Shane Ross, and Mark Stremler. 2010. Multi-physics and Multilevel Fidelity Modeling and Analysis of Olympic Rowing Boat Dynamics. (6 2010).

[22] Radoslav Nabergoj. 2010. *Fondamenti di Tenuta della Nave al Mare.* Università degli studi di Trieste.

[23] Ulrik D. Nielsen. 2017. Transformation of a wave energy spectrum from encounter to absolute domain when observing from an advancing ship. *Applied Ocean Research* 69 (Dec. 2017), 160–172. https://doi.org/10.1016/j.apor.2017.10.011

[24] Ulrik D. Nielsen. 2018. Deriving the absolute wave spectrum from an encountered distribution of wave energy spectral densities. *Ocean Engineering* 165 (10 2018), 194–208. https://doi.org/10.1016/j.oceaneng.2018.07.046

[25] Apostolos Papanikolaou. 1985. On integral-equation-methods for the evaluation of motions and loads of arbitrary bodies in waves. *Ingenieur-Archiv* 55, 1 (1985), 17–29.

[26] Apostolos Papanikolaou and Thomas E. Schellin. 1992. A three-dimensional panel method for motions and loads of ships with forward speed. *SCHIFFSTECHNIK* (1992). http://resolver.tudelft.nl/uuid:128b5c5e-63c5-48c3-b4e8-f0ccadb68497

[27] Chuck Pheatt. 2008. Intel® Threading Building Blocks. *J. Comput. Sci. Coll.* 23, 4 (apr 2008), 298.

[28] The Teuchos Project Team. 2021. *The Teuchos Project Website.* https://trilinos.github.io/epetra.html

[29] The Trilinos Project Team. 2020. *The Trilinos Project Website.* https://trilinos.github.io

[30] Bruno Turcksin, Martin Kronbichler, and Wolfgang Bangerth. 2016. WorkStream – A Design Pattern for Multicore-Enabled Finite Element Computations. *ACM Trans. Math. Softw.* 43, 1, Article 2 (aug 2016), 29 pages. https://doi.org/10.1145/2851488

[31] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[32] Wenjun Zhang and Zhengjiang Liu. 2014. Real-Time Ship Motion Prediction Based on Time Delay Wavelet Neural Network. *Journal of Applied Mathematics* 2014 (08 2014), 1–7. https://doi.org/10.1155/2014/176297