https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

Full parallelization of the finite-element Lagrangian sea ice model neXtSIM for kilometer-scale simulations

Fabien Salmon[1], Pierre Rampal[2,4], Stéphanie Leroux[3], Timothy Williams[4,5], Einar Ólason[4,5], and Nicolas Barral[1]

[1]Inria, Univ. Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, F-33400 Talence, France
[2]Institut des Geosciences de l'Environnement, CNRS, UMR 5001, Saint Martin d'Heres, France
[3]Datlas, Grenoble, France
[4]Nansen Environmental and Remote Sensing Center, Bergen, Norway
[5]Bjerknes Centre for Climate Research, Bergen, Norway

Correspondence: Fabien Salmon (fabien.salmon@inria.fr) and Nicolas Barral (nicolas.barral@inria.fr)

Abstract. Accurate modeling of sea ice dynamics is a major challenge, but also a key requirement for forecasting its future evolution and assessing its impact on climate change. The sea-ice model neXtSIM is specifically designed for this purpose, relying on a Lagrangian framework to accurately capture highly localized features such as leads and ridges, which likely play an important role in controlling the energy exchanges at the interfaces of the atmosphere-ice-ocean coupled system. Despite a parallelisation effort a few years ago, several components of the code, which are intrinsically required by a Lagrangian framework, such as the remeshing procedure, remained sequential, which created significant bottlenecks that limited the model's overall capabilities. To tackle this problem, we further developed the parallel anisotropic mesh adaptation tool ParMMG2D, which demonstrates excellent scalability up to 512 processors with 20 million elements. This tool has been integrated within neXtSIM to enable parallel remeshing, currently limited to homogeneous isotropic meshes. In addition, parallelization of subsequent interpolation, output writing, and drifting buoy tracking has also been achieved. Together, these improvements now enable kilometer-scale simulations within a reasonable timeframe: a one-year simulation with 2 km elements takes only a few days, while a full season with 1 km elements can be simulated in about a week on 128 processors. Therefore, neXtSIM now stands out as cutting-edge software for simulating highly resolved sea ice dynamics, combining the accuracy of a Lagrangian framework with high computational efficiency.

1 Introduction

The Arctic sea ice cover has undergone profound changes over the past few decades (Comiso, 2012; Kwok, 2018). These transformations have been accompanied by a shift in the dynamical regime, characterized by an increase in extreme fracturing events and an acceleration of sea ice drift. The highly non-linear response of sea ice to external forcing presents a significant challenge for current modeling approaches, particularly in capturing these changes and predicting the future evolution of Arctic sea ice. However, addressing this challenge is becoming increasingly critical, both due to the essential role of sea ice in the climate system and to the increasing intensity of industrial activities in the Arctic.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

Unlike most sea ice models, neXtSIM (Rampal et al., 2016; Ólason et al., 2025a) is based on a Lagrangian framework, which is particularly well-suited for preserving highly localized features, such as cracks, leads, or ridges. This software code has been applied to simulate, e.g., the coupled spatio-temporal scaling invariance of sea ice deformation (Rampal et al., 2019), the dramatic dynamical response of the sea ice cover in the Canadian Arctic to an intense atmospheric storm (Rheinlænder et al., 2022), wave–sea ice interactions (Williams et al., 2017; Boutin et al., 2021), and Antarctic sea ice (Santana et al., 2025). In a fully Lagrangian approach, the computational mesh is advected with the sea ice velocity, which inevitably results in significant mesh deformations and distortions over time. Once the mesh elements become too distorted, they must be corrected using a remeshing procedure. This remeshing is currently performed using an adapted version of the BAMG mesh generator originally developed by Hecht (Hecht, 1998). Although neXtSIM has been parallelized (Samaké et al., 2017; Ólason et al., 2025b), BAMG remains inherently sequential, making the remeshing step a computational bottleneck within the model, especially for very fine meshes. For example, adapting a 1 km mesh—including remeshing, domain decomposition, and field interpolation—can take up to ten minutes of computation. Furthermore, at typical sea-ice velocities of the order 10^{-1} m·s^{-1}, a remeshing step may be needed every few tens of minutes of simulated time, *i.e.*, after fewer than ten timesteps. Such a high frequency makes the overall computational cost prohibitive.

To eliminate this bottleneck, BAMG must be replaced by a parallel remeshing tool. We propose using the MMG library (MMG, 2025), an open-source remeshing software. This modern code is widely used in the numerical simulation community through a wide range of applications, such as CO_2 injection (Legentil et al., 2023), calving for glaciers and icebergs (Wheel et al., 2024), crack propagation (Cornejo et al., 2020), biophysics (Rauff et al., 2025), or combustion (Moureau et al., 2021). However, only a sequential version for 2D meshes is currently available, so a parallel implementation must first be developed. Moreover, MMG is originally designed to adapt an entire mesh to physical features, whereas in our case we would like to modify as few elements as possible to reduce numerial error as much as possible.

Parallelizing the remeshing procedure also necessitates parallelizing associated tasks, such as interpolating fields from the old mesh to the updated one or domain decomposition. An additional benefit of parallelization is reduced memory usage, as maintaining a global mesh is no longer required. To fully exploit this advantage, other parts of the code that previously depended on the global mesh must be adapted to operate only on local meshes. In particular, input/output operations and buoy-tracking should be performed in parallel using the local meshes.

This article presents the strategy developed to fully parallelize the neXtSIM code, including its remeshing procedure, by upgrading the MMG library (MMG, 2025) and using it instead of the BAMG library. A brief description of the model is given in Section 2. Section 3 details the parallelization strategy adopted for MMG, together with the associated results and performance. Section 4 addresses the integration of parallel remeshing within the model and then outlines the parallelization of the remaining sequential components of the code, together with the resulting gains in computational cost. Section 5 discusses the performance of the fully parallelized version of neXtSIM, its validation, and an illustration of the new levels of resolution achieved in simulations.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

2 neXtSIM description

This section provides a brief overview of the main characteristics of neXtSIM, focusing on aspects of the model relevant to the improved parallelization discussed here. neXtSIM simulates sea ice by solving equations that describe the ice's melt and growth (thermodynamics) and motion (dynamics), forced by atmospheric and oceanic inputs. The thermodynamic equations are all formulated on a single mesh element, and so their parallelization is trivial. However, solving the dynamics requires the gradient of various quantities between neighboring elements, as well as the advection of prognostic quantities with ice motion. neXtSIM can be used to simulate the complex mechanical behavior of sea ice with a brittle rheology, the latter producing a highly heterogeneous ice field, leading to the use of a Lagrangian advection method (Rampal et al., 2016). For further details on the model, see Ólason et al. (2025a), where the current version of neXtSIM is described.

2.1 Dynamics modeling

The main dynamics module of neXtSIM is based on the Brittle Bingham-Maxwell rheology (BBM, Ólason et al., 2022), although more traditional modules, based on the viscous-plastic rheology (Hibler, 1979), are also available. The BBM rheology has three prognostic variables: velocity ($\mathbf{u}$), internal stress ($\boldsymbol{\sigma}$), and damage (d). These are calculated using two main equations, the momentum equation and the constitutive equation.

The momentum equation used in neXtSIM is (Bouillon and Rampal, 2015; Rampal et al., 2016; Ólason et al., 2022, 2025a)

$$m\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{\nabla}\cdot(\boldsymbol{\sigma}h) + A(\boldsymbol{\tau}_a + \boldsymbol{\tau}_w) + \boldsymbol{\tau}_b + mf\boldsymbol{k}\times\boldsymbol{u} - mg\boldsymbol{\nabla}\eta, \tag{1}$$

where m is the ice mass per unit area, $\boldsymbol{u}$ is the ice velocity, $\boldsymbol{\sigma}$ is the internal stress tensor, h is the ice slab thickness (not volume per unit area), A the fraction of the element covered by ice, ρ the ice density, $\boldsymbol{\tau}_a$ and $\boldsymbol{\tau}_w$ are the atmosphere and ocean stress terms, respectively, $\boldsymbol{\tau}_b = -C_b\boldsymbol{u}$ is the basal stress term introduced by Lemieux et al. (2015), $mf\boldsymbol{k}\times\boldsymbol{u}$ is the Coriolis term, with vertical unit vector $\boldsymbol{k}$, and $mg\boldsymbol{\nabla}\eta$ is the ocean-tilt term. The forcing terms $\boldsymbol{\tau}_a$, $\boldsymbol{\tau}_w$, and η are all calculated from the atmospheric or oceanic states, read from files or received from a different model through a coupler (e.g. Boutin et al., 2023).

The momentum equation is coupled to the constitutive equation through the internal stress term $\boldsymbol{\nabla}\cdot(\boldsymbol{\sigma}h)$. The constitutive equation for BBM is

$$\frac{\partial \boldsymbol{\sigma}}{\partial t} = E\mathbf{K}:\dot{\boldsymbol{\varepsilon}} - \frac{\boldsymbol{\sigma}}{\lambda}\left(1 + \widetilde{P} + \frac{\lambda\dot{d}}{1-d}\right), \tag{2}$$

where E is the ice elasticity, $\dot{\boldsymbol{\varepsilon}}$ is the strain-rate tensor, $E\mathbf{K}:\dot{\boldsymbol{\varepsilon}}$ is the stiffness operator, λ is the viscous relaxation time, $\widetilde{P}$ is a plastic threshold, and $\dot{d}$ is the time derivative of d.

The elasticity and viscous relaxation time evolve with damage as

$$\begin{aligned} E &= E_0(1-d)e^{-C(1-A)} \\ \lambda &= \lambda_0[(1-d)e^{-C(1-A)}]^{\alpha-1}, \end{aligned} \tag{3}$$

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

where E_0 and λ_0 are the elasticity and viscous relaxation time of undamaged ice, and $\alpha > 1$ and $C > 0$ are constants. The strain-rate tensor has the components

$$\begin{aligned} \varepsilon_{11} &= \frac{\partial u}{\partial x} \\ \varepsilon_{22} &= \frac{\partial v}{\partial y} \\ \varepsilon_{12} &= \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right). \end{aligned} \tag{4}$$

The stiffness operator is

$$\begin{pmatrix} (\mathbf{K} : \dot{\varepsilon})_{11} \\ (\mathbf{K} : \dot{\varepsilon})_{22} \\ (\mathbf{K} : \dot{\varepsilon})_{12} \end{pmatrix} = \frac{1}{1-\nu^2} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & 1-\nu \end{pmatrix} \begin{pmatrix} \dot{\varepsilon}_{11} \\ \dot{\varepsilon}_{22} \\ \dot{\varepsilon}_{12} \end{pmatrix} \tag{5}$$

where $\nu \approx 1/3$ is Poisson's ratio. Finally, the plastic threshold is

$$\widetilde{P} = \begin{cases} \frac{P_{\max}}{\sigma_N} & \text{for } \sigma_N < -P_{\max}, \\ -1 & \text{for } -P_{\max} < \sigma_N < 0, , \\ 0 & \text{for } \sigma_N > 0. \end{cases} \tag{6}$$

where

$$P_{\max} = P\left(\frac{h}{h_0}\right)^f e^{-C(1-A)}, \tag{7}$$

where $f \in [1,2]$ is a constant and σ_N is the normal stress.

The change in damage $\dot{d}$ is calculated by comparing the stress state in each element with the Mohr-Coulomb envelope in the $\{\sigma_N, \tau\}$ space of normal and shear stresses

$$\begin{aligned} \sigma_N &= \frac{1}{2}(\sigma_{11} + \sigma_{22}) \\ \tau &= \frac{1}{2}\sqrt{(\sigma_{11} - \sigma_{22})^2 + 4\tau_{12}^2}, \end{aligned} \tag{8}$$

with σ_{ij} the components of $\boldsymbol{\sigma}$. If these values fall within the Mohr-Coulomb envelope,

$$\tau = \mu\sigma_N + c, \tag{9}$$

where μ is the internal friction coefficient and c is the cohesion, then $\dot{d} = 0$. However, if the stress value in the element falls outside the envelope, d is increased to keep them within the envelope. In practice, this amounts to first calculating an intermediate stress state $\boldsymbol{\sigma}'$ using equation (2) with $\dot{d} = 0$ and then updating damage as

$$d^{n+1} = d^n + (1 - d_{\text{crit}})(1 - d^n)\frac{\Delta t}{t_d}, \tag{10}$$

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

and the stresses as

$$\boldsymbol{\sigma}^{n+1} = \boldsymbol{\sigma}' - (1 - d_{\text{crit}})\boldsymbol{\sigma}' \frac{\Delta t}{t_d}, \tag{11}$$

where n notes the previous time step and $n+1$ the current one. At the start of the simulation, $d = 0$ everywhere. Through the damaging process described above, d can approach (but never reach) one. neXtSIM also includes thermodynamic healing processes that reduce d. In equations (10) and (11), Δt is the time-step and

$$t_d = \Delta x \sqrt{\frac{2(1+\nu)\rho}{E}} \tag{12}$$

is a relaxation time, with Δx the spatial resolution. This is intended to ensure that the damage does not propagate faster than an elastic shear wave.

The salient point in the current context is that d increases locally, and this process can be very fast, relative to other dynamic processes. This simulates fracturing in the ice, where the fractures can be only a few elements wide but hundreds of kilometers long. Such deformation zones are a distinctive feature of the Arctic ice cover, observable from space (e.g. Kwok, 2018) and affect the evolution of the ice cover (e.g. Boutin et al., 2023). The Lagrangian advection scheme, developed for neXtSIM, was conceived to better preserve these features.

2.2 Numerical and computing methods

The dynamical equations (1) and (4) are discretized using the finite element method, the implementation described by Samaké et al. (2017) and Ólason et al. (2025a). Both the discretization and the resolution of the resulting system are carried out locally within each partition. neXtSIM is implemented using a distributed-memory paradigm based on MPI (Samaké et al., 2017), with inter-process communication managed through Boost.MPI, an abstraction layer of standard MPI. Although most of the model components are parallelized, the entire remeshing step remains entirely sequential.

2.2.1 Remeshing

As outlined in the introduction, neXtSIM relies on a Lagrangian framework with mesh motion. In its current version (Ólason et al., 2025a), remeshing is activated when the smallest angle of a triangle falls below $10°$. This step is carried out by the sequential BAMG software. Since BAMG is not parallelized, the local meshes must first be merged into a single global mesh on the root process before remeshing. BAMG is specifically designed to preserve as many nodes from the original mesh as possible, restricting modifications to localized regions where remeshing is necessary. This minimizes numerical diffusion during interpolation from the old mesh to the new one. Linear P1 interpolation is applied to nodal fields, while element-centered fields are transferred using the conservative remapping scheme described in (Ólason et al., 2025a). As with remeshing, these interpolations are performed sequentially on the entire mesh. Finally, the mesh is partitioned using METIS (Karypis and Kumar, 1998) for subsequent parallel computations.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

The sequential nature of the remeshing process severely limits the code's parallel efficiency, particularly for fine meshes where frequent mesh adaptation is required. It therefore becomes a bottleneck for simulations on very fine meshes composed of millions of elements.

2.2.2 Inputs-Outputs

The dynamics of sea ice are driven by oceanic and atmospheric conditions. These forcing data are provided in netCDF files, which contain spatiotemporal information over the entire domain for a specified period, based on a given spatial grid and time discretization. In neXtSIM, each process reads these files whenever a new forcing timestep is reached or after a remeshing step, since the physical oceanic and atmospheric fields must then be interpolated onto the updated mesh.

The neXtSIM outputs are stored in either binary or NetCDF files at user-specified times. The writing is performed in the root process for the entire mesh, with one file generated per timestep. The physical fields must therefore be gathered on the root process. A dedicated post-processing tool can read and analyze the information contained in these files.

2.2.3 Drifting buoys tracking

neXtSIM can track the trajectories of selected initial points (referred to as drifters), which can be compared, for example, with the observed paths of real buoys. The trajectory of each drifter is computed in the root process, independently of its location within the partitions. To achieve this, the local displacements of all nodes are first gathered in the root process, where a global interpolation is performed at the drifter positions. The resulting displacements are then added to their previous positions and the updated locations are stored in a file. During this procedure, ice concentration is also checked: drifter positions are updated only if the concentration is non-zero. This requires collecting the local concentration values on the root process and performing a global interpolation at the drifter positions.

3 Development of parallel remeshing

As discussed previously, the Lagrangian approach requires frequent remeshing to eliminate poor-quality elements. In neXtSIM, this task relies on the sequential BAMG tool, which constitutes a significant bottleneck in the numerical workflow. This section introduces the strategy adopted to parallelize the remeshing process within a newly parallelized dedicated code. We first describe the sequential MMG software, then outline the parallelization strategy, and finally present the performance of the resulting tool.

3.1 MMG

MMG (2025) is dedicated to the adaptation of anisotropic meshes. The inputs are an initial mesh and a size map encoded in a Riemannian metric space (commonly termed "the metric"). The output anisotropic mesh is the mesh such that all edges have a size such as prescribed by the metric. The latter comprises a vector space and a field of metric tensors $M = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ over the domain Ω. At each vertex of the mesh, the metric tensors contain information about the size of the elements and their

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

orientation (George et al., 1991; Loseille and Alauzet, 2011a). For instance, to obtain isotropic elements with a characteristic size of h over the entire domain, the metric tensor should be chosen as $\mathcal{M}_{ij} = \frac{1}{h^2} I$ where I denotes the identity matrix.

MMG modifies the initial mesh following four operations (Dapogny et al., 2014):

- Splitting too long edges
- Collapsing too small edges
- Swapping bad quality elements
- Relocating vertices through smoothing to improve element quality

These operations are performed iteratively to converge toward a mesh that closely matches the prescribed metric.

3.2 Parallelization strategy

Parallel remeshing methods are generally divided into two categories. The first focuses on developing parallel mesh generation algorithms (Casagrande et al., 2005; Chrisochoides and Nave, 2003; Ibanez, 2016). A key issue for these methods is to ensure the consistency and conformity of the mesh across partition boundaries. The second integrates sequential remeshing tools within a parallel framework by restricting mesh adaptation to the interior of partitions (Coupez et al., 2000; Cavallo et al., 2005; Benard et al., 2016). This prevents incompatibilities at the partition interfaces and ensures conformity. The mesh is then repartitioned and remeshed iteratively, allowing adaptation at the initial frozen interfaces. A parallel version of MMG was designed previously Cirrottola and Froehly (2019) following a remesh-then-repartition strategy, but was restricted to 3D meshes. In this work, the 2D parallel version ParMMG2D (2025), (Salmon, 2026b; Salmon and Barral, submitted) was built adopting the approach introduced in 3D and enhancing it for the specific sea-ice application.

As done by Cavallo et al. (2005), the repartitioning is based on an advancing-front algorithm. Processor boundaries are dynamically modified; if a vertex is shared by several processors, all elements connected to that vertex are transferred to a single processor. The selected processor is the one with the smallest number of elements among those sharing the vertex. After the element migration, the remeshing process is applied again. Figure 1 gives an example of the advancing-front repartitioning method with three partitions. Here, a single layer of triangles is exchanged, although the number of element layers exchanged can be set to any integer value.

Such an algorithm may produce disconnected partitions, resulting in isolated elements or groups of elements. To prevent this issue, the lists of connected elements are identified after repartitioning. If multiple lists are found, the largest one is retained as the new partition, while the remaining groups are reassigned to a neighboring partition. The target processor is chosen as the one with the smallest number of elements. This process ensures the contiguity of each partition.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

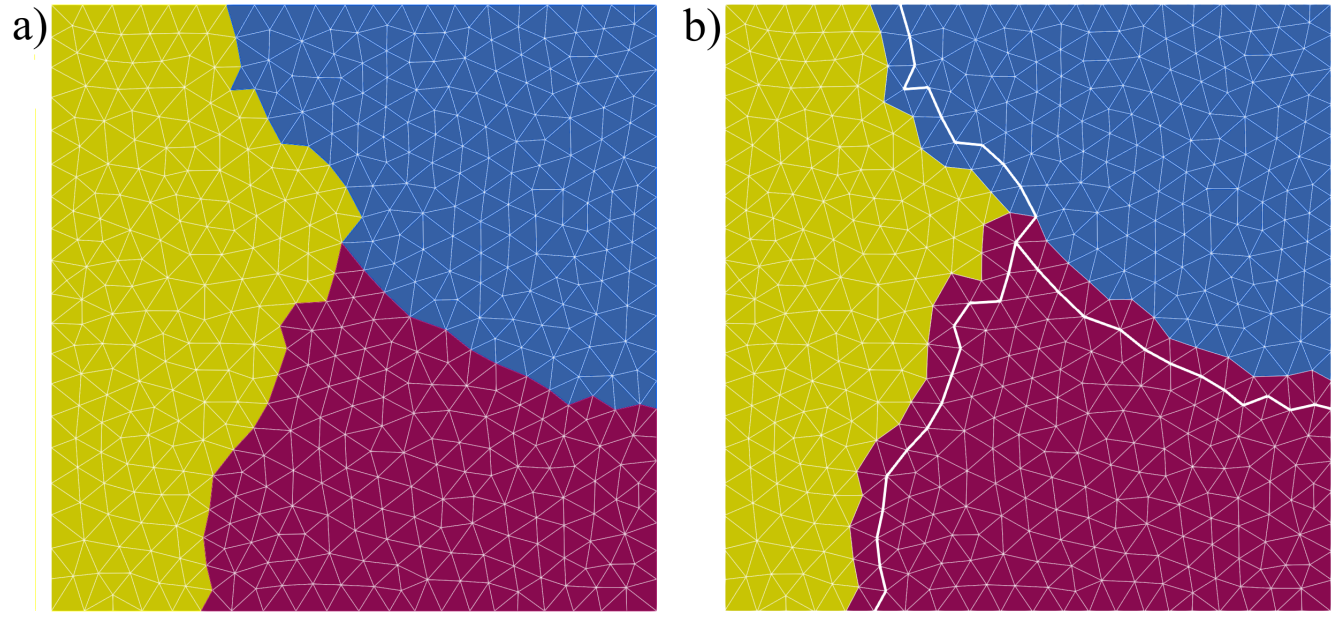

Figure 1. Advancing-front algorithm with three partitions. a) Initial domain decomposition. b) New partitioning following the exchange of one layer of elements. The bold white line indicates the initial processor boundaries.

Following each remeshing step, the metric must be interpolated onto the updated mesh to enable the subsequent remeshing iteration. Since the interface elements remain fixed, the metric is not modified near the partition boundaries. Consequently, the interpolation can be performed using only the previous mesh, without requiring additional elements from neighboring partitions. In ParMMG2D, the interpolation process is done after remeshing and before repartitioning.

At the conclusion of the iterative remeshing process, the domain decomposition may become significantly unbalanced, as the advancing-front algorithm does not inherently ensure load balancing across processors. In such cases, a final repartitioning step is necessary to restore balance. This can be achieved using various well-established partitioning tools, such as Metis (Karypis and Kumar, 1998), Scotch (Pellegrini, 2012), ParMetis (Karypis et al., 1997), or PT-Scotch (Chevalier and Pellegrini, 2008). In ParMMG2D, the user can specify a threshold parameter to control the acceptable level of load imbalance.

3.3 Validation

This section presents several validation cases of increasing complexity to assess the ability of the parallel remesher to produce meshes compliant with the metric. This is first assessed qualitatively : on top of being refined where expected, the meshes must look clean and there should be no imprint of the partition interfaces. Quantitatively, we look at a quality criterion based on the

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

Algorithm 1 ParMMG2D algorithm

```
Preprocessing (check input mesh and metric, analysis)
if the mesh is centralized then
    Partitioning using a domain decomposition tool (Metis by default)
else
    Finding correspondence between nodes from the different partitions at the parallel interfaces
end if
for i = 1 to n_iter do
    Copying mesh, metric
    Sequential MMG remeshing on each partition with frozen boundaries
    Interpolating the metric on the new local mesh
    Repartitioning based on the advancing-front algorithm
end for
if the output mesh must be centralized then
    Merging the decomposed mesh
else if the output mesh must be decomposed but it is unbalanced then
    Repartitioning using a domain decomposition tool (Metis by default)
end if
```

aspect ratio of a triangle K in the metric space (Frey and George, 2007):

$$Q = 4\sqrt{3}\frac{|K|_{\mathcal{M}}}{\sum_{i=1}^{3}\|e_i\|_{\mathcal{M}}^2} \tag{13}$$

where $|K|_{\mathcal{M}}$ is the surface area of element K and $\|e_i\|_{\mathcal{M}}$ is the length of edge i. This quality quantity is equal to 1 for a perfectly shaped element and tends to 0 for degenerate elements.

The first and simplest test case focuses on generating a high-quality uniform mesh starting from a poor initial mesh. Figure 2.a shows the initial degraded mesh, while Fig. 2.b displays the resulting mesh composed of the same number of vertices obtained using a uniform metric with ParMMG2D, employing 4 processors over 4 iterations. To further assess the parallel capabilities of the tool, a finer test case involving approximately one million elements was also performed using 16 processors. This calculation produces a uniform mesh in which 99.99% of the elements (978,779 elements) have a quality between 0.8 and 1, with only 25 elements falling in the range 0.6 to 0.8, which confirms the overall high quality of the resulting mesh, like for the coarse mesh in Figure 2.

In what follows, four classic academic anisotropic test cases are studied. Even if highly anisotropic meshes are not used in neXtSIM, the remesher is expected to handle such meshes appropriately. The input mesh is always a uniform $[-1,1]\times[-1,1]$

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

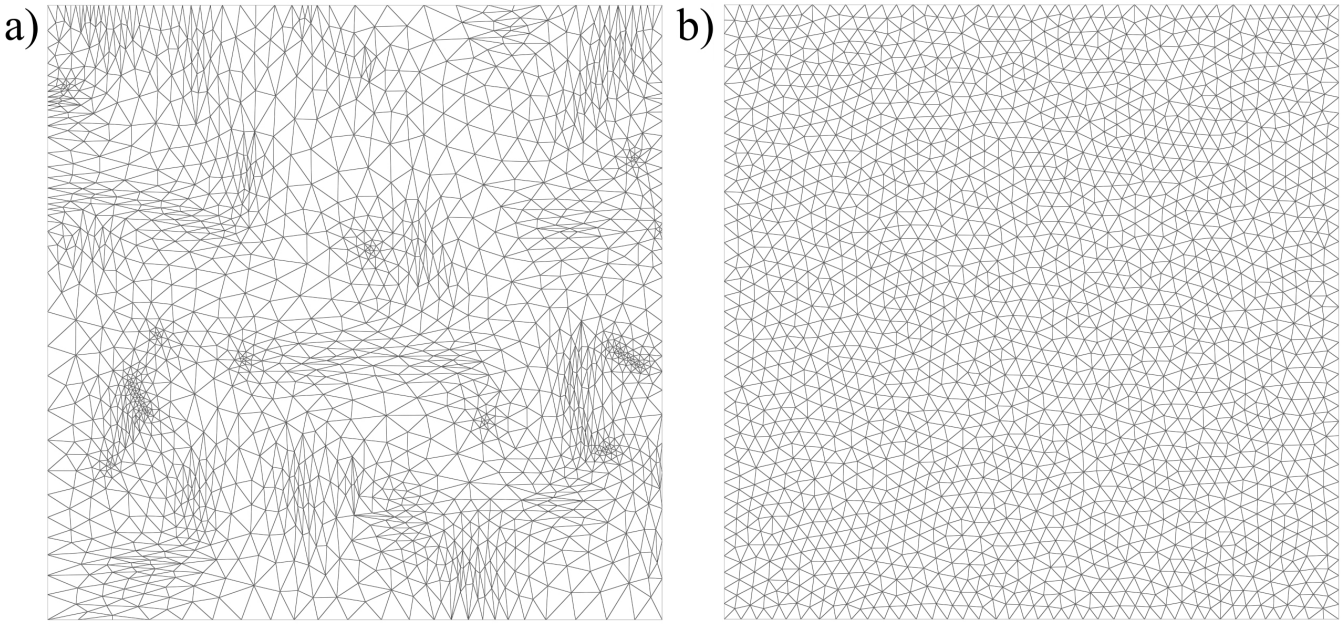

Figure 2. a) Initial degraded mesh. b) Uniform-metric adapted mesh with the same number of vertices, generated using ParMMG2D on 4 processors.

square mesh composed of approximately 500,000 isotropic triangles and 250,000 vertices. For each test case, four parallel iterations were carried out, with three layers of triangles exchanged during the partition advancing-front procedure.

In the first case, let us consider a constant element size of $h_c = 0.02$ in the y direction (so 100 elements along y) and a variable size along x: $h(x) = h_c|1 - e^{-|x-0.5|}| + 0.003$. The metric representing a kind of shock wave is then

$$\mathcal{M}(x) = \begin{pmatrix} \frac{1}{h(x)^2} & 0 \\ 0 & \frac{1}{h_c^2} \end{pmatrix}.$$

Figure 3.a shows the resulting adapted mesh.

In the second case, let us consider a large element size everywhere except in a thin circular area:

$$h(x,y) = \begin{cases} 0.005 & \text{if } 0.99 \leq x^2 + y^2 \leq 1.01 \\ 1 & \text{else if } x^2 + y^2 > 1.01 \\ 0.1 & \text{else} \end{cases}.$$

The resulting metric is isotropic

$$\mathcal{M}(x,y) = \begin{pmatrix} \frac{1}{h(x,y)^2} & 0 \\ 0 & \frac{1}{h(x,y)^2} \end{pmatrix}.$$

Figure 3.b shows the resulting adapted mesh.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

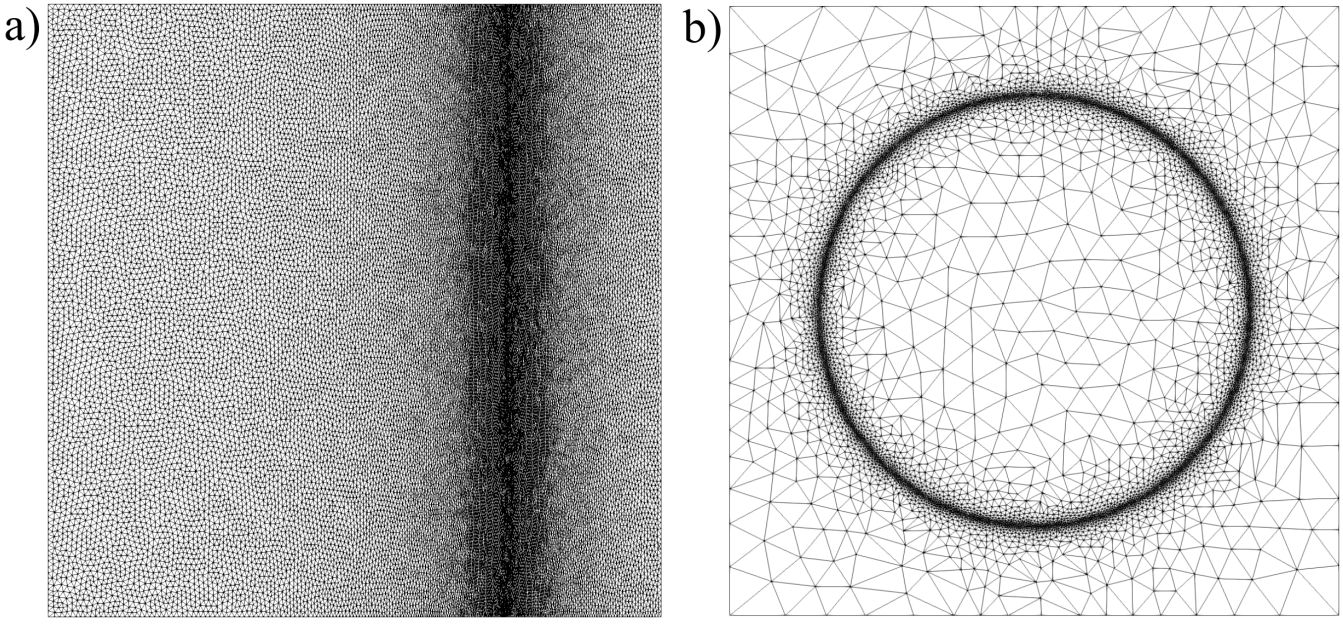

Figure 3. a) Anisotropic mesh capturing a shock wave, generated using ParMMG2D on eight processors. b) Isotropic mesh associated to a thin circle, generated using ParMMG2D on eight processors.

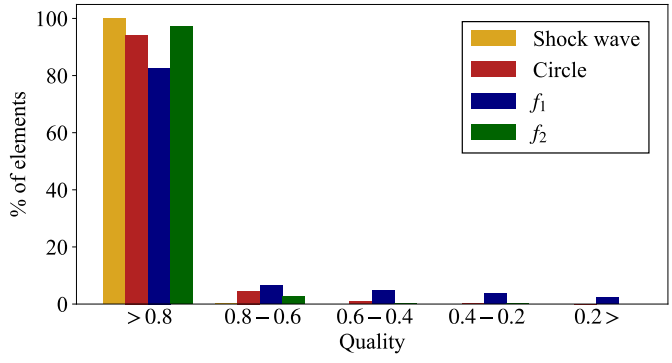

Figure 4. Distribution of element quality for the four academic test cases.

In both test cases, the absence of any noticeable artifacts at processor interfaces highlights the effectiveness of the parallel strategy. The final mesh quality remains high and is comparable to that obtained with the sequential MMG version. For the shock-wave case, 99.9% of the elements have a quality greater than 0.8, while the remaining elements exhibit a quality greater than 0.6 (see Fig. 4). For the circle case, the overall quality of the mesh is also high, with 94% of the elements having a quality greater than 0.8.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

For the following two test cases, we consider analytical functions f_1 and f_2 that are assumed to represent a physical solution of a given problem. The mesh must be adapted to accurately capture this solution. Loseille and Alauzet (2011b) established the mathematical formulation of an optimal metric for a given target number of vertices in the adapted mesh. In both following cases, we apply their formulation, setting the target number of vertices to match that of the initial mesh.

Figure 5 shows the solution function f_1 alongside the resulting mesh obtained from ParMMG2D using the Loseille optimal metric, where

$$f_1(x,y) = \alpha \sin(\beta xy)$$

with

$$\alpha = \begin{cases} 1 & \text{if } \frac{\pi}{50} \leq |xy| \leq \frac{2\pi}{50} \\ 0.01 & \text{otherwise} \end{cases},$$

and

$$\beta = \begin{cases} 1 & \text{if } |xy| < \frac{\pi}{50} \\ 50 & \text{otherwise} \end{cases}.$$

We then consider a function derived by Digonnet et al. (2019). Figure 6 shows the function and the resulting mesh obtained from ParMMG2D with the function

$$f_2(x,y) = a(a(\sqrt{x^2+y^2})) + a(a(\sqrt{(x-1)^2+(y-1)^2}))$$

where

$$a(x) = \tanh\left(\sin\left(\frac{5\pi}{2}x\right)\right).$$

Despite the intrinsic complexity of the functions, the adapted anisotropic meshes correspond to the expectations, without visible traces that reveal the location of processor boundaries. For f_1, some elements exhibit low quality, but more than 80% of the elements have a quality above 0.8. For f_2, the overall quality of the mesh is even higher than in the circular configuration (Figure 4).

3.4 Computing performance

This section addresses the parallel performance of ParMMG2D, evaluated using the previously studied circle and shock wave test cases. The computation times for various numbers of processors and mesh sizes, along with the strong scalability curves for each mesh are shown in Figures 7 and 8, corresponding to the circle and shock wave test cases, respectively. The target number of vertices of the output mesh is equal to the number of vertices of the input mesh. The results for each step of the iterative procedure (metric interpolation on the updated mesh, sequential remeshing, and advancing-front partitioning) are displayed using different colors and symbols in both figures. The processors used in this section and the remaining of the paper are 2×64-core AMD Zen3 Milan EPYC 7763 operating at 2.45 GHz, with 1 TB of memory.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

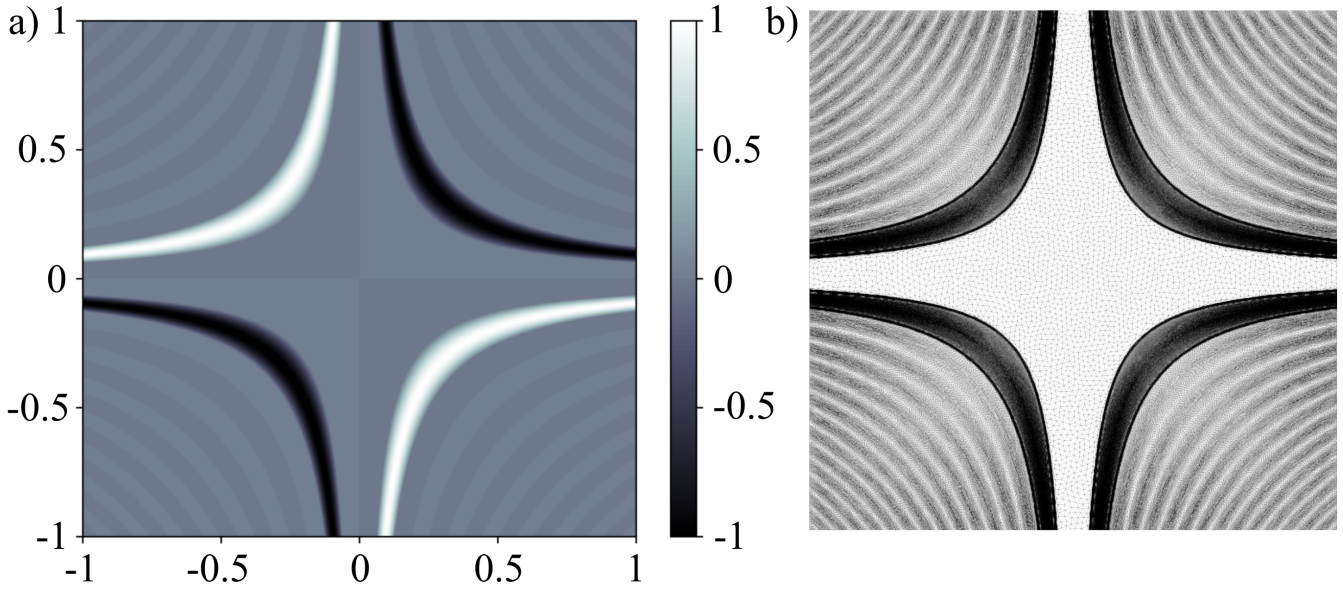

Figure 5. a) Reference function f_1 for mesh adaptation. b) Corresponding adapted mesh using ParMMG2D over 16 processors.

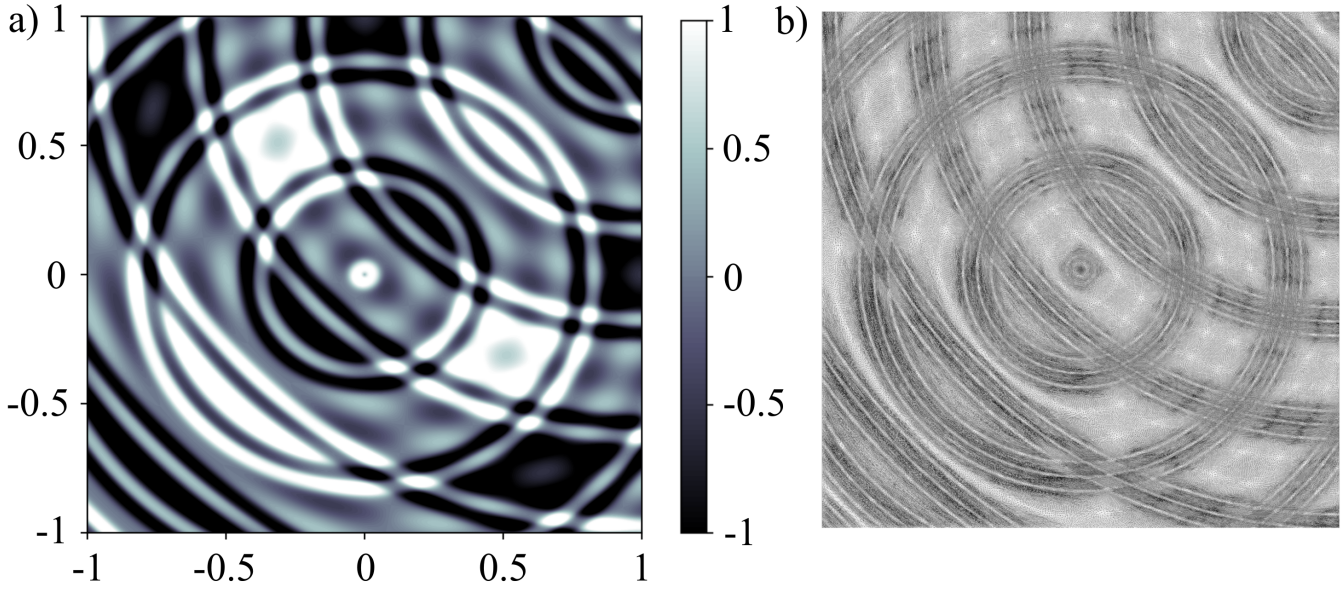

Figure 6. a) Reference function f_2 for mesh adaptation. b) Corresponding adapted mesh using ParMMG2D over 16 processors.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

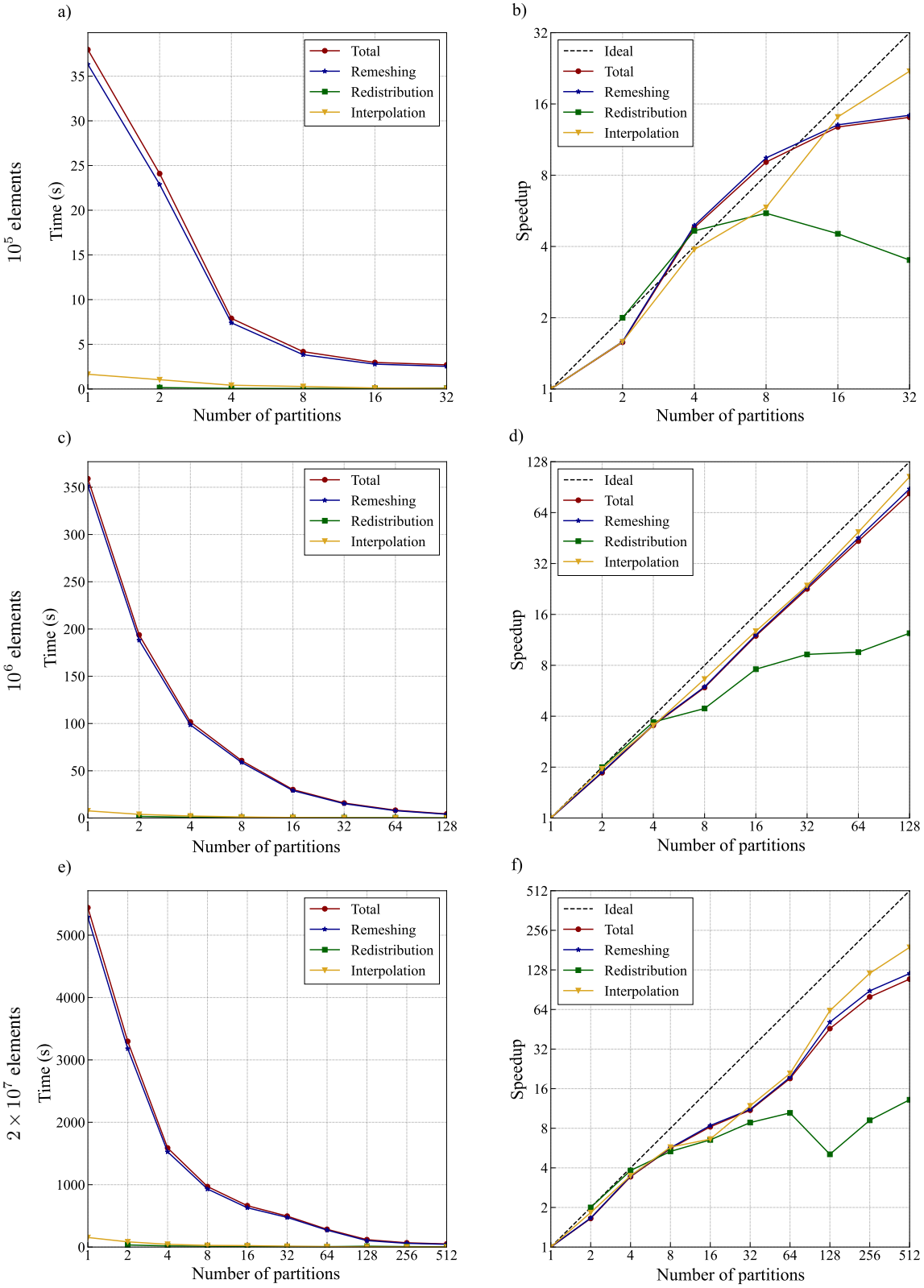

Figure 7. Computation time and strong scalability of each part of the mesh adaptation process for the circle test case with 10^5 elements (a & b), 10^6 elements (c & d), and 2×10^7 elements (e & f).

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

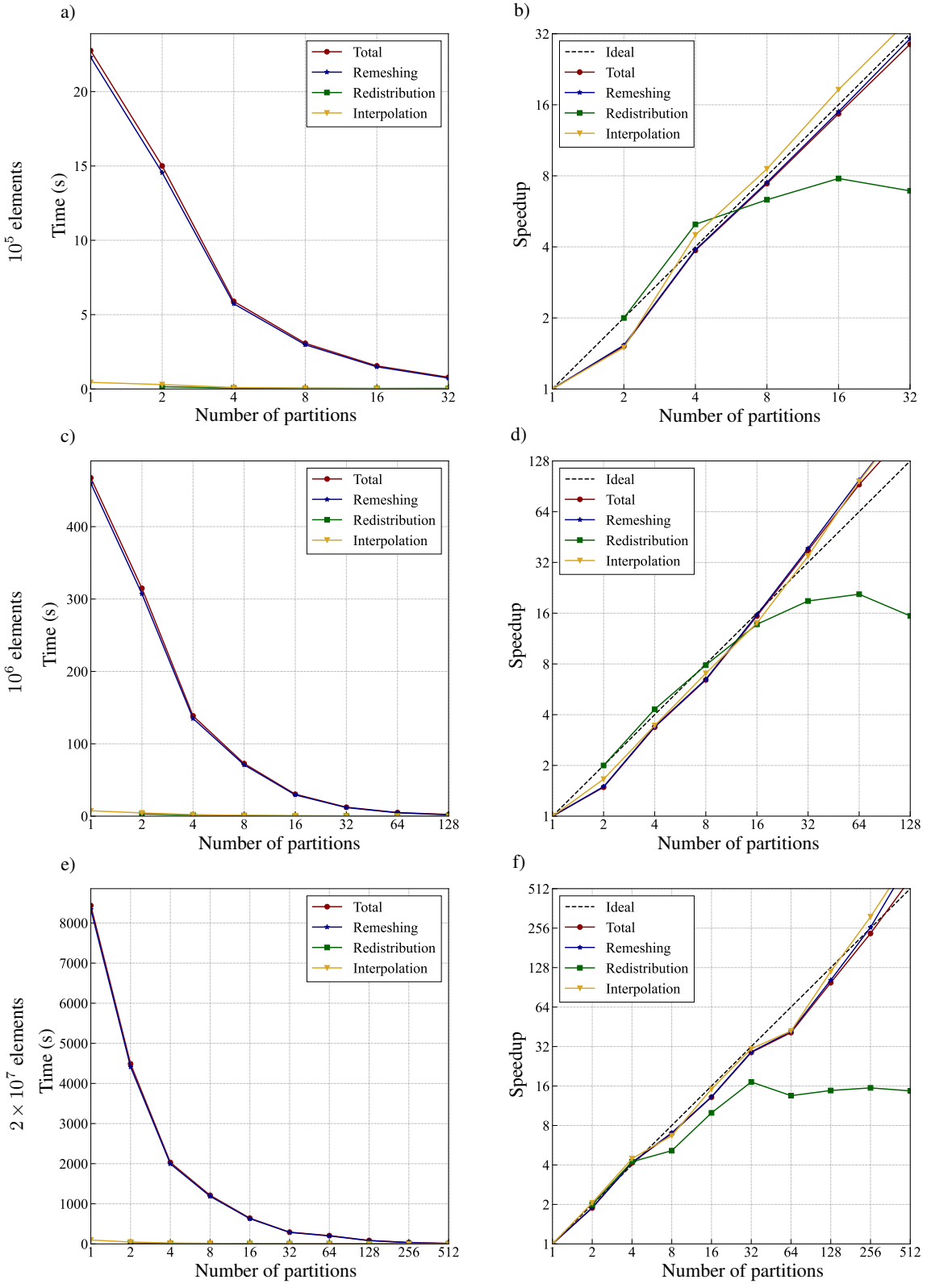

Figure 8. Computation time and strong scalability of each part of the mesh adaptation process for the shock wave test case with 10^5 elements (a & b), 10^6 elements (c & d), and 2×10^7 elements (e & f).

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

Across all configurations, the majority of the computational cost is attributed to the remeshing step, whereas the interpolation and redistribution steps require only a negligible amount of time. For both test cases, mesh adaptation of 10^5 elements is completed within a few seconds with more than four processors. In contrast, adapting a mesh with one million elements requires several hundred seconds when using fewer than four partitions. However, the computation time decreases significantly as the number of partitions increases, becoming negligible with 128 processors. When adapting a mesh with 20 million elements, the computational cost is significant, but it is reduced to a few dozen seconds with 512 processors, corresponding to approximately 40,000 elements per partition. This also means that using several 64-CPU nodes does not degrade performance.

The interpolation of the metric on each updated mesh exhibits excellent scalability up to the highest number of partitions tested. In contrast, the redistribution process does not scale as efficiently. Regardless of the mesh size, a first performance bottleneck typically appears around eight partitions. Beyond this point, the speedup gradually diverges from the ideal scalability curve, although this deviation appears to be configuration-dependent. For example, in the circle test case with 2×10^7 elements, an outlier is observed at 128 partitions, likely caused by the formation of highly non-contiguous partitions following the advancing-front process. As the number of partitions increases, the likelihood of generating non-contiguous partitions also rises, which may explain the reduced efficiency of this approach at large processor counts. Fortunately, the redistribution step remains computationally inexpensive within the overall remeshing process. The most computationally intensive step is the sequential remeshing performed on each partition. Not surprisingly, the scalability curve of this step closely aligns with that of the overall procedure. Although the global algorithm demonstrates high scalability, its performance appears to be configuration-dependent. Scalability is more favorable for the shock wave test case than for the circle.

ParMMG2D is therefore a mesh adaptation tool that delivers strong performance even in challenging configurations, making it well suited to support the simulation of various physical processes when coupled with a parallel code. The next section describes its coupling with the sea ice model neXtSIM.

4 Full parallelization of neXtSIM

Starting from the initial parallelization detailed in Samaké et al. (2017), this section describes the parallelization of the remaining sequential components of neXtSIM, including the use of the parallel remeshing library described in the previous section. It also includes a comparative study with legacy BAMG and concludes with an assessment of the parallel performance of the fully parallelized version of neXtSIM (Salmon, 2026a).

4.1 Parallel remeshing

ParMMG2D aims to replace the sequential remeshing tool Hecht (1998) in neXtSIM. This section presents the choice of different parameters and approaches in the coupling with ParMMG2D.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

Several parameters must be adjusted both in the input and within the ParMMG2D settings when studying sea ice dynamics using neXtSIM. In this work, the metric has been assumed to be isotropic and constant, resulting in a uniform mesh. This choice is motivated by the underlying physical models, some of which are not validated at very small element sizes or on heterogenous meshes. To avoid potential issues, strict control over element size has been enforced throughout the simulations. It is worth mentioning that future studies could leverage the more advanced capabilities offered by ParMMG2D, particularly in terms of non-uniform and anisotropic mesh adaptation.

Each mesh modification requires the interpolation of all physical fields from the old mesh to the newly generated one. This interpolation process inevitably introduces errors and numerical diffusion. However, in sea ice modeling, accurate assessment of damage and precise localization of cracks are essential. Since crack formation is a highly localized phenomenon, significant numerical diffusion is unacceptable. The BAMG algorithm addresses this by modifying only a small number of problematic elements, thereby limiting the impact of interpolation errors. In contrast, ParMMG2D typically remeshes the entire mesh, which must therefore be carefully controlled. To mitigate this issue, it is necessary to define a criterion that selectively targets elements of insufficient quality.

Since the objective is to maintain an isotropic mesh, a minimum angle threshold could be enforced for all triangles. To ensure behavior consistent with that of BAMG, a minimum angle of 0.4 radians (approximately 23^o) is selected. However, near the domain boundaries, elements may become highly distorted due to sea ice motion. To account for this, a stricter angle threshold of 0.74 radians (approximately 42.5^o) is applied to triangles located on the boundary or adjacent to boundary elements. Additionally, triangle sizes must be controlled, as remeshing only a few isolated elements could locally produce a non-uniform mesh. Based on empirical testing, the following size criterion has been adopted: any triangle whose edge length deviates by more than 75% from the targeted uniform size must be modified. To enhance the effectiveness of both criteria, an additional forecasting step is introduced. This step estimates a potential displacement at each node based on the velocity field computed by neXtSIM, using a virtual time step defined as $\frac{1}{4}\frac{\sqrt{\mathcal{A}(K)}}{\overline{u}}$, where $\mathcal{A}(K)$ is the area of triangle K and $\overline{u}$ is the mean velocity of the triangle, averaged over its three nodes. If the predicted configuration of the triangle fails to meet either the angle or size criteria, the element is flagged for modification.

In Sections 3.3 and 3.4, four remeshing iterations were systematically applied across all configurations. However, in the neXtSIM framework, the most efficient compromise is to perform only two remeshing iterations, combined with a repartitioning that exchanges three layers of triangles during the advancing-front process. With this approach, the majority of elements located along the initial parallel interfaces are included in the remeshing, although a small number of elements may remain unprocessed. As there are very few such elements, the likeliness that any of them are of poor quality is low. If such a case does occur, it will naturally be corrected during the remeshing step of the following time step. While this additional remeshing is computationally expensive, experimental tests indicate that it happens sufficiently rarely to be profitable.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

As previously discussed in Section 3.2, the domain decomposition may be significantly unbalanced at the end of the remeshing process, since the advancing-front algorithm does not account for load balancing. Experimental tests indicate that, for triangular meshes containing up to 20 million elements, Metis is the most efficient partitioning tool despite the preliminary step of merging the local meshes on the root process. Unfortunately, we were not able to leverage the parallel capabilities of partitioners such as ParMetis and PT-Scotch to decrease the CPU cost. In neXtSIM, an imbalance threshold of 30% is adopted to trigger a final repartition, as this offers the best trade-off between avoiding expensive mesh decomposition and maintaining efficient numerical resolution of the sea ice equations on a weakly unbalanced mesh. For fine meshes, final decomposition is rarely required, which substantially reduces computation time. Conversely, for coarse meshes, final repartitioning is frequently necessary, though in these cases the partitioning time remains negligible.

4.2 Parallel interpolation

After the remeshing step, the various fields must be interpolated onto the new mesh. To fully exploit the benefits of parallel remeshing, the interpolation should also be parallelized. This avoids merging local meshes into a single global mesh and prevents the additional computational cost of interpolating on such a large mesh.

One approach consists in dividing the domain with a Cartesian grid (Fig. 9), where the cell size is a tunable parameter. In neXtSIM, given the mesh uniformity, a reasonable compromise for the cell size is to target roughly 20 triangles per cell. On each partition, the triangles of the previous mesh are distributed across the grid cells. For example, in Fig. 9, the triangles of partition 12 are split into three groups since the partition spans three different cells. The same Cartesian grid is then applied to the new mesh, and each partition determines which cells intersect its domain. For instance, if a partition contains triangles in the red cell of Fig. 9, it receives from all processors holding old triangles in that cell (in this case, processors 12, 8, 26, 1, 23, and 14) these triangles.

Figure 10 compares the performance of parallel and sequential interpolations for two meshes with element sizes of 2 km and 1 km. The results presented correspond to averages over approximately 100 interpolations. On more than two processors for the 1-km mesh and four processors for the 2-km mesh, the parallel interpolation becomes faster than the sequential one. Since the sequential approach merely gathers all local fields on the root process, whereas the parallel approach requires identifying the triangles to exchange then performing the corresponding communications, the parallel method is less efficient when using a small number of processors. In practice, however, simulations on 2-km or 1-km meshes typically rely on more than two processors, making this limitation negligible. As the number of processors increases, the interpolation time decreases substantially and continues to improve up to 128 partitions for the 2-km mesh and 256 partitions for the 1-km mesh. For example, the computation time is approximately 3 seconds when using 256 processors for a 1-km mesh, while the sequential approach requires approximately 85 seconds. Beyond reducing runtime, the parallel strategy also alleviates memory usage, since the root process no longer needs to store global fields. Moreover, the parallel approach remains effective for coarser meshes: with

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

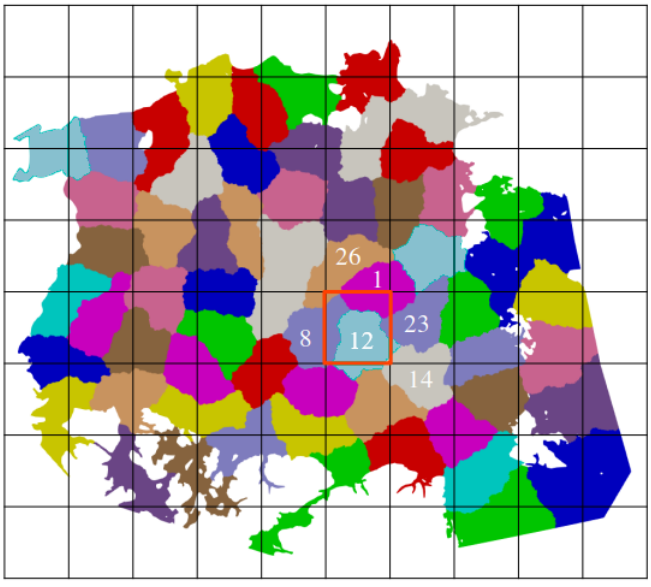

Figure 9. Example of a domain decomposition overlaid with a Cartesian grid, corresponding here to the decomposition before remeshing.

10-km triangles, the interpolation time is on the order of a second or a few tenths of a second, comparable to the sequential version with up to four processors and even smaller when more processors are used.

4.3 Parallel binary outputs

With the full parallelization of the code, the output procedures should also be parallelized to reduce memory consumption. This is achieved through parallel file writing using MPI I/O. The mesh file is generated directly from the local meshes while preserving the same node and element numbering as in the previous sequential outputs. This reordering is performed with a minimal number of MPI exchanges. Similarly, nodal and element fields are stored in the field file from local data only, without gathering them on the root process, and the same reordering strategy as for the mesh is applied.

Figure 11 compares the performance of the parallel and sequential approaches for two meshes with element sizes of 2 km and 1 km. Similarly to the interpolation step discussed previously, the parallel writing time becomes lower than the sequential one for more than 2 processors. While the sequential version only requires gathering all local fields in the root process, the parallel version involves additional operations before writing. As a result, parallel I/O does not yet offset this overhead for a low processor count. Nevertheless, in practice, simulations with 2 km or 1 km meshes typically rely on more than two processors, so this limitation is not critical. When increasing the number of processors, the writing time decreases significantly and continues

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

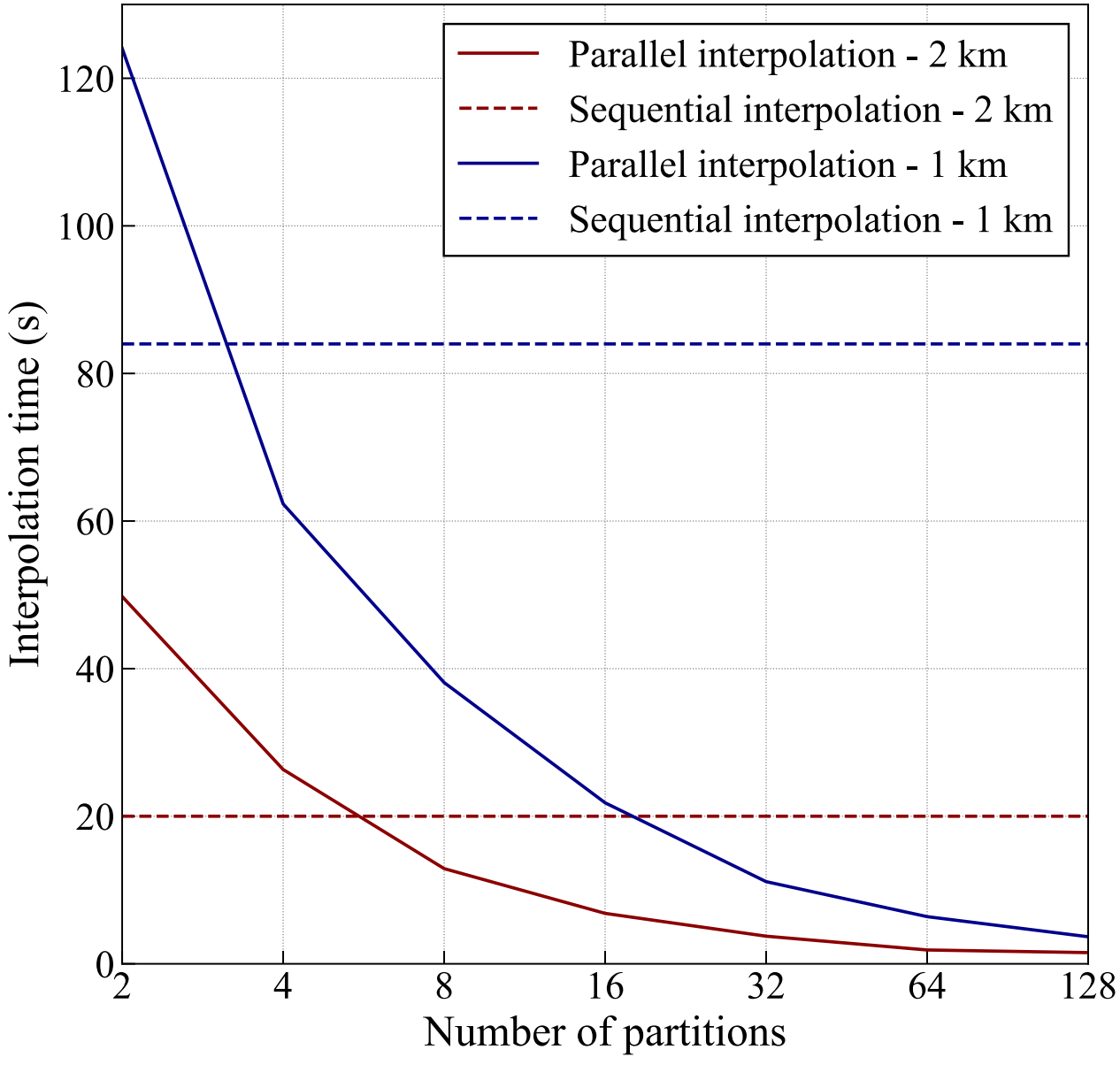

Figure 10. Computation time required for one interpolation, for two meshes with element sizes of 1 km and 2 km, as a function of the number of partitions. For comparison, the sequential timing obtained with the previous version of the code is also reported.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

to improve up to 64 partitions for both meshes. Beyond this point, MPI exchanges become more costly, leading to a slight increase in writing time. In addition to reducing execution time, the parallel strategy also reduces memory requirements. In the previous sequential implementation, gathering all fields and storing the global mesh in the root process could cause memory saturation. This limitation is now removed, for at least up to 15 million elements, corresponding to meshes with 1 km triangles. It should be noted that the parallel approach remains efficient for coarser meshes. For 10 km triangles, the time required to write a single output is only a few tenths of a second for up to 64 processors, which is comparable to the performance of the previous sequential version.

4.4 Parallel NetCDF outputs

The netCDF library supports parallel I/O for both structured and unstructured grids. For unstructured grids, each point must be handled individually, which is inefficient in parallel. Therefore, we adopt a structured approach, dividing the entire grid into rectangular local subgrids. Since local domains are generally arbitrary and non-rectangular, they must first be decomposed into non-overlapping rectangles that fully cover the domain, including areas without data.

The rectangular decomposition is carried out as follows. First, the bounding box of the local domain in the first process is considered as the initial rectangle. The intersection between this rectangle and the bounding box of the second process is then computed. If an intersection exists, it is removed from the second rectangle, which may no longer remain a rectangle. The resulting region is then subdivided into the minimal number of rectangles. The procedure is repeated with the third process, considering intersections with the rectangles obtained from the previous steps, and continues similarly for subsequent processes. This method guarantees that the entire domain is covered by non-overlapping rectangles. Figure 12 illustrates an example of how the decomposed domain is partitioned into rectangles.

In this example, processor 11 handles only a single rectangle. Part of processor 12's domain overlaps with this rectangle, so processor 12 sends the corresponding data to processor 11. At the same time, processor 11 sends the data located at the top of its domain, which lies within rectangle 10, to the appropriate processor. These exchanges are performed by all processes to ensure that each has all the required data locally. Once the data transfer is complete, each process writes the data corresponding to its local rectangles using the netCDF library in parallel.

Figure 13 compares the performance of the parallel and sequential approaches for grids of 3 and 13 million points. With fewer than four processors, the parallel version shows slightly higher writing times, as it involves the additional operations described previously, whereas the sequential version only gathers local fields on the root process. Thus, at low processor counts, parallel I/O does not fully offset this overhead. However, in practice, simulations at this resolution typically use more than four processors, making this limitation negligible. As the number of processors increases, the writing time decreases up to 8 partitions for both grids. Beyond this point, the cost of rectangle decomposition and MPI exchanges grows, causing a slight increase in time. However, this increase is much smaller than in the sequential approach, where MPI exchanges

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

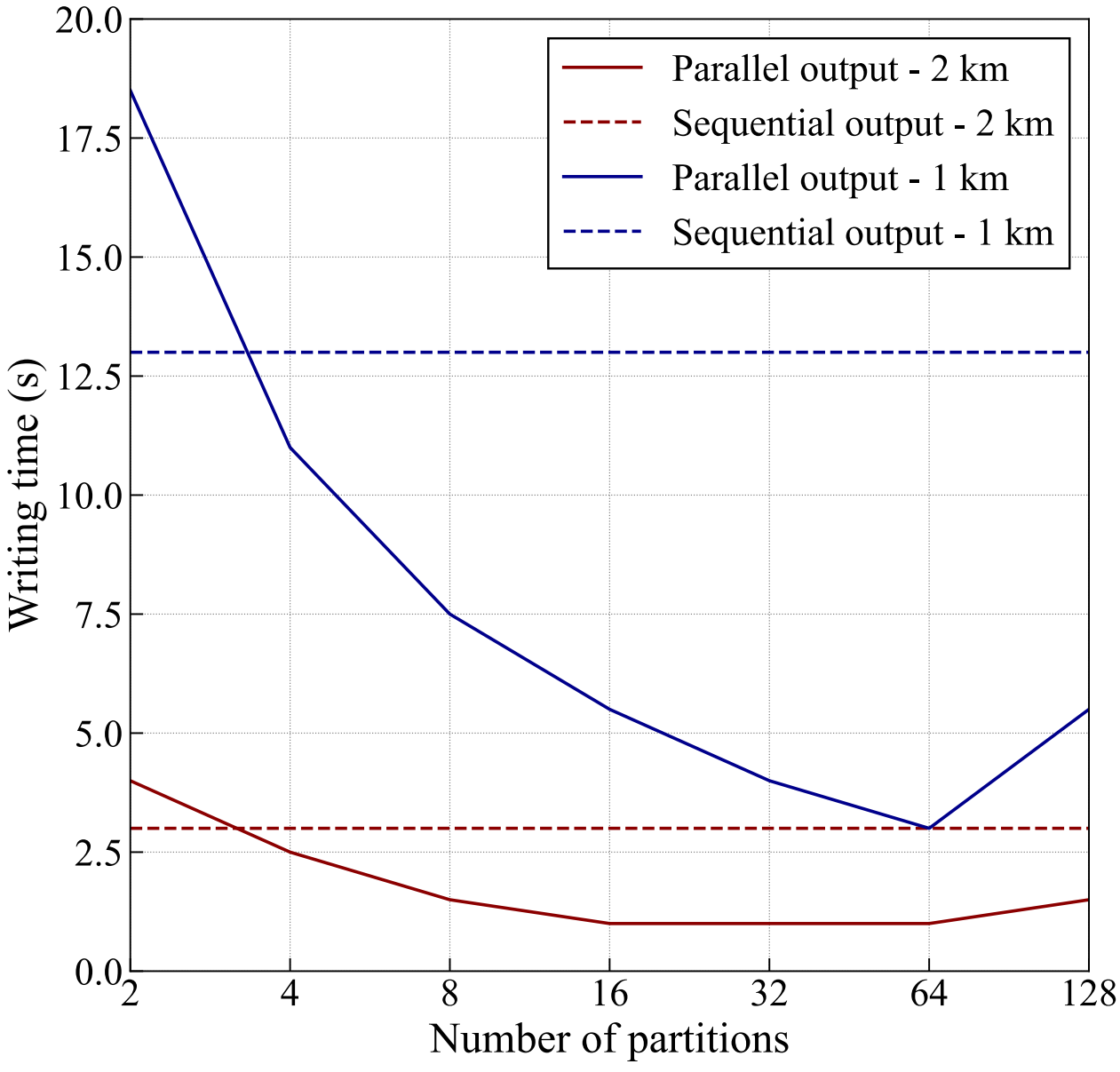

Figure 11. Computation time required for writing a single output, for two meshes with element sizes of 1 km and 2 km, as a function of the number of partitions. For comparison, the sequential timing obtained with the previous version of the code is also reported.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

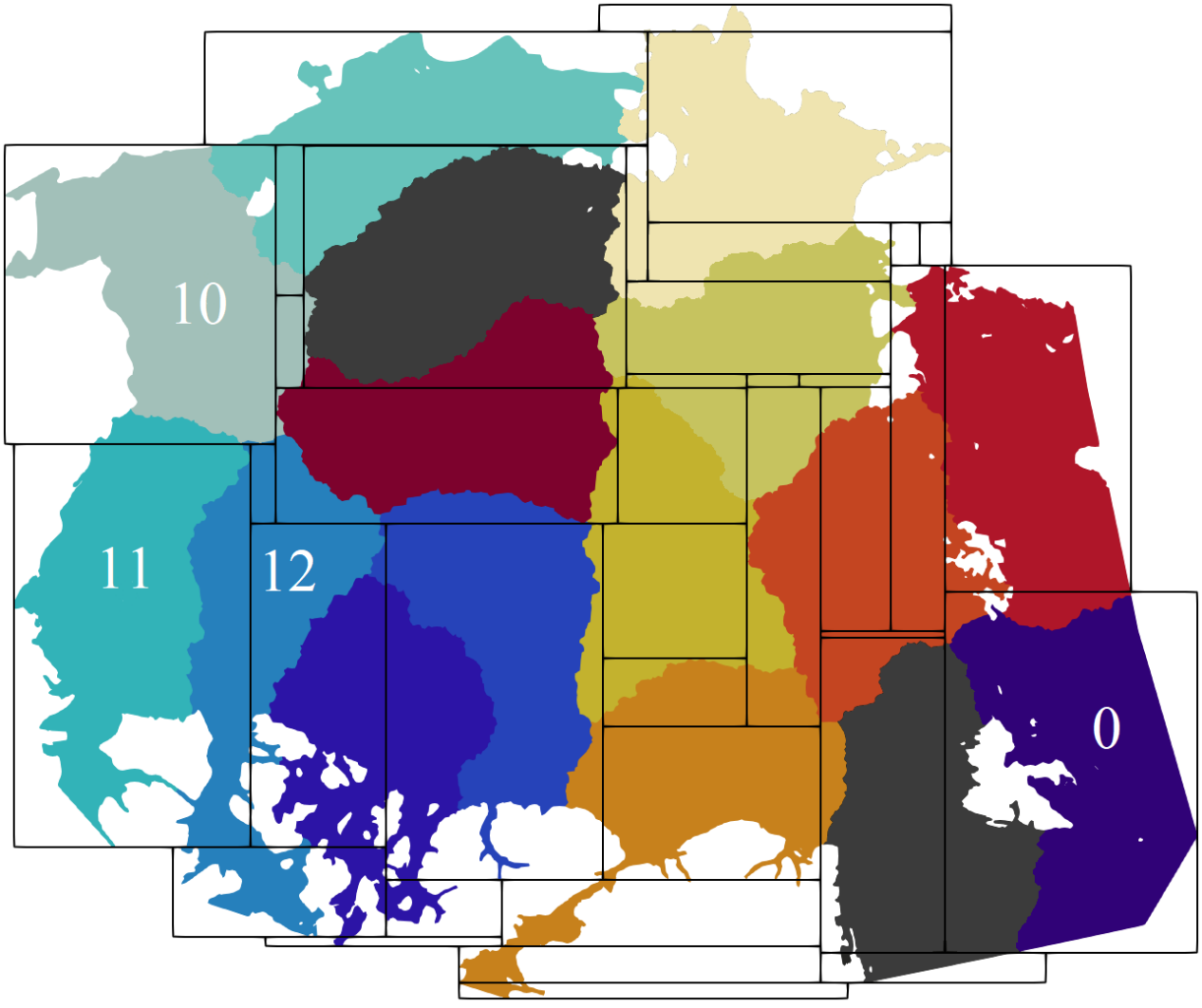

Figure 12. Domain decomposition covered by a rectangular partitioning.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

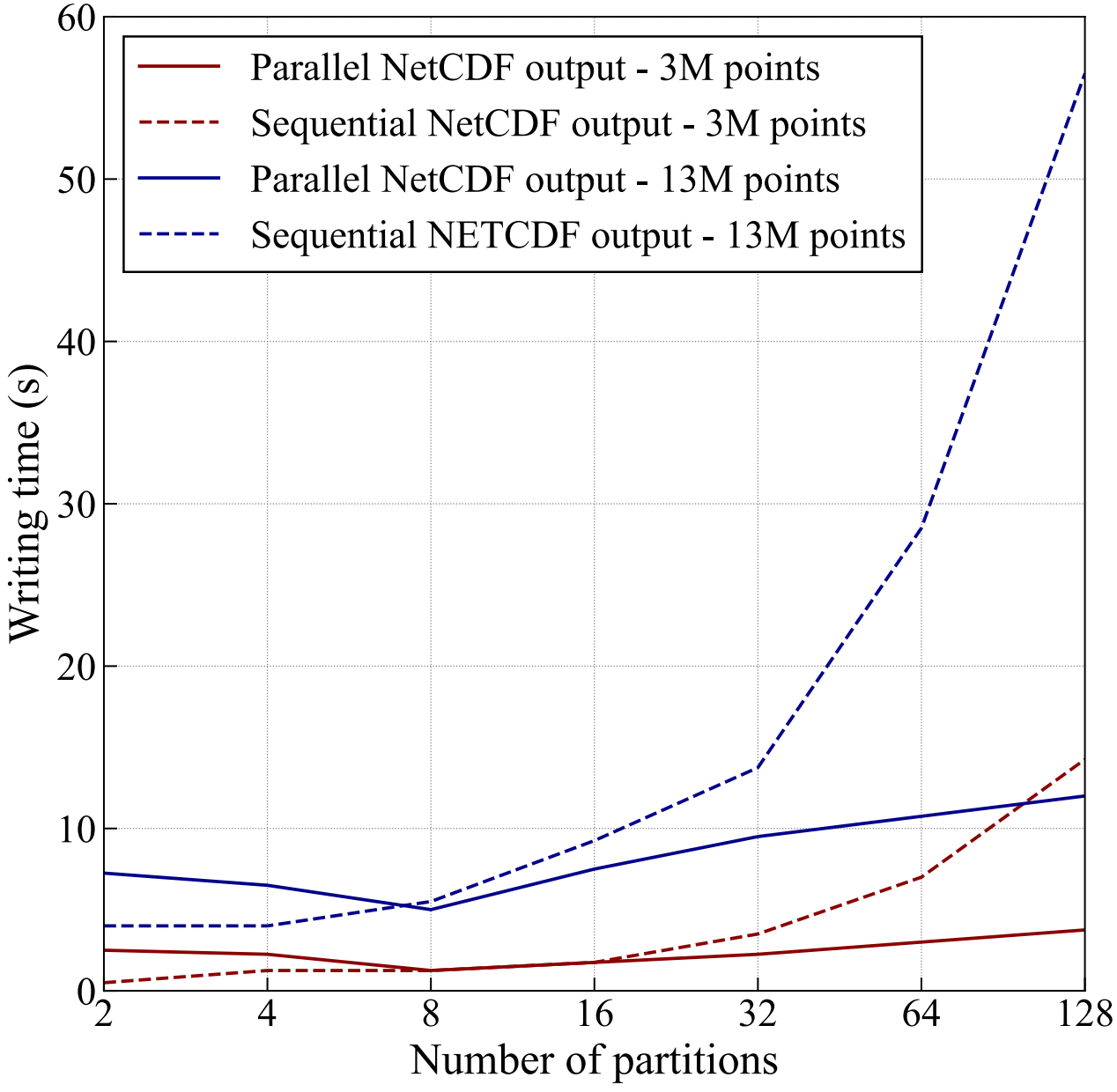

Figure 13. Computation time required for writing a single netCDF output, for two grids composed of 3 and 13 millions points, as a function of the number of partitions. For comparison, the sequential timing obtained with the previous version of the code is also reported.

for gathering data dominate. The combination of rectangular decomposition and smaller MPI exchanges therefore improves writing efficiency. Additionally, the parallel strategy reduces memory requirements, as observed in the previous parallelized operations.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

4.5 Parallel drifting buoys tracking

As the global mesh is no longer accessible, the buoy trajectories must be computed locally on each process. First, the coordinates of all drifting buoys are broadcast to every processor. Then, following the same principle as in the parallel interpolation, the domain is partitioned using a Cartesian grid, and each local triangle is assigned to a cell. For each drifting buoy, we check whether it lies in a cell containing local triangles, and if so, whether it belongs to one of these triangles. When this is the case, its displacement is obtained through barycentric interpolation of the displacements at the three triangle nodes. This strategy avoids the global interpolation required in the sequential approach. Finally, the updated positions of the local drifting buoys are gathered in the root process, which writes them to a file.

Figure 14 compares the performance of the parallel algorithm with that of the previous sequential method for two meshes with element sizes of 2 km and 1 km, using five million drifting buoys. In contrast to the previous cases, the parallel approach outperforms the sequential one for any number of processors. This improvement stems from the simplification of the interpolation process, which is now performed locally at the scale of individual triangles. With 128 processors, the computation time falls below one second for both meshes, making it negligible in the overall simulation cost. As a result, the drifter procedure can now be employed without restriction, regardless of mesh resolution.

5 Validation & Performances of the full model

5.1 Validation

This section compares the results obtained with the previous partially parallel version of neXtSIM and the new fully parallel version on a realistic configuration. It corresponds to the Arctic Sea from February 2006 to the end of the year. Atmospheric forcing is provided by the hourly ERA5 reanalysis (Hersbach et al., 2020), while oceanic forcing is derived from the TOPAZ4 reanalysis (Sakov et al., 2012). The initial conditions for the thickness and concentration of sea ice are taken from the PIOMAS reanalysis (Zhang and Rothrock, 2003).

The extent and volume of sea ice are critical variables as they play a key role in the climate system. In addition, sea ice drift is also a good indicator for assessing the performance of sea ice models, as it strongly influences the sea ice mass balance with its role in ice transport, particularly through the Fram Strait. Fig. 15 shows the evolution of the ice extent, its volume, and the mean drift for two simulations achieved with both versions of neXtSIM.

The seasonal evolution of the extent and volume of sea ice aligns well with the expected patterns in both simulations. Moreover, the differences between the simulations remain small. The mean discrepancies in extent and volume are 5.4×10^3 km^2 and 35.8 km^3, respectively, which represent only 0.1% and 0.3% of their corresponding mean values. These differences are consistent with what would be anticipated given the minor variations introduced by both remeshing procedures. Regarding sea

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

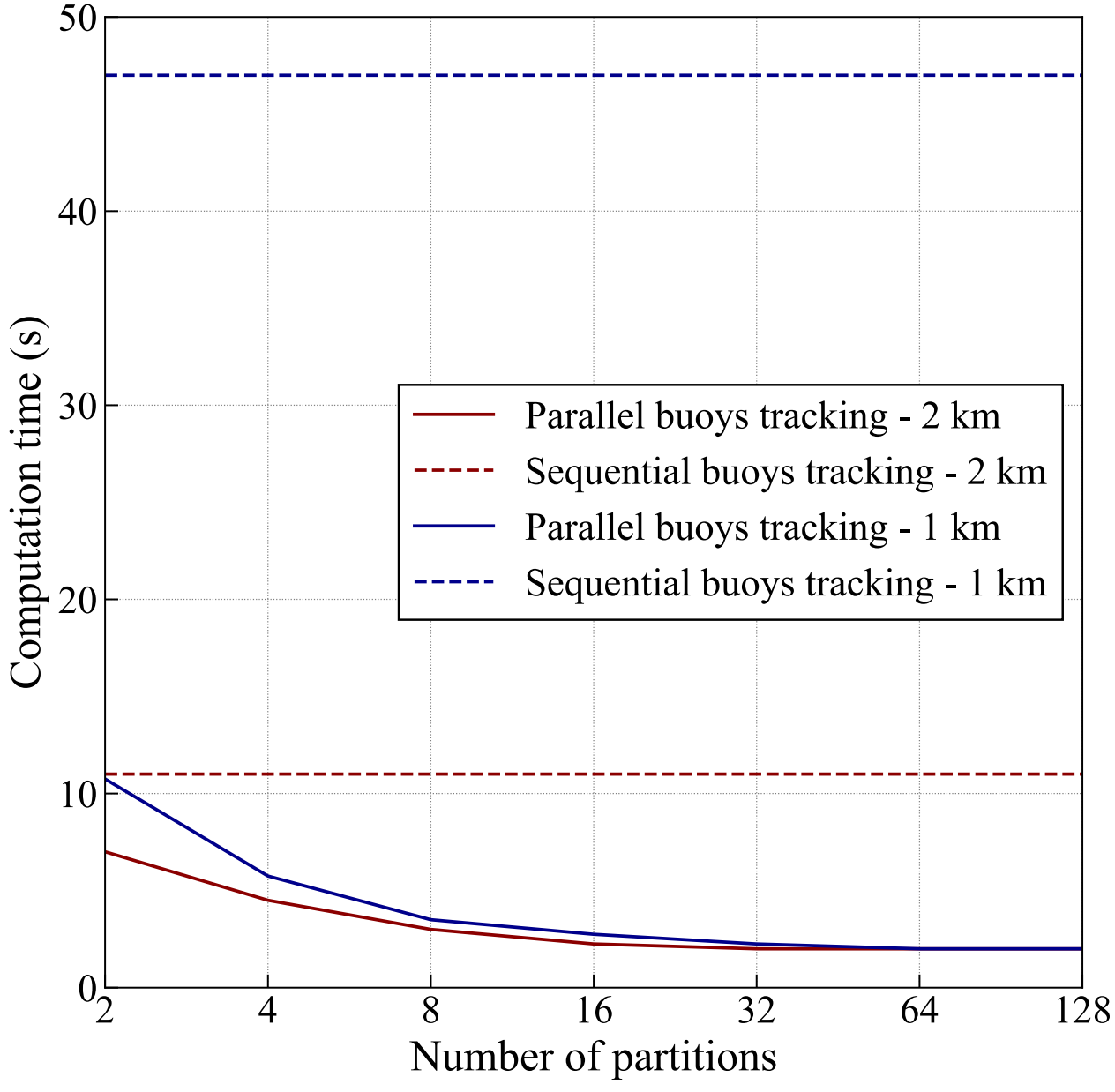

Figure 14. Computation time required for computing and writing a single time five million drifters, for two meshes with element sizes of 1 km and 2 km, as a function of the number of partitions. For comparison, the sequential timing obtained with the previous version of the code is also reported.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

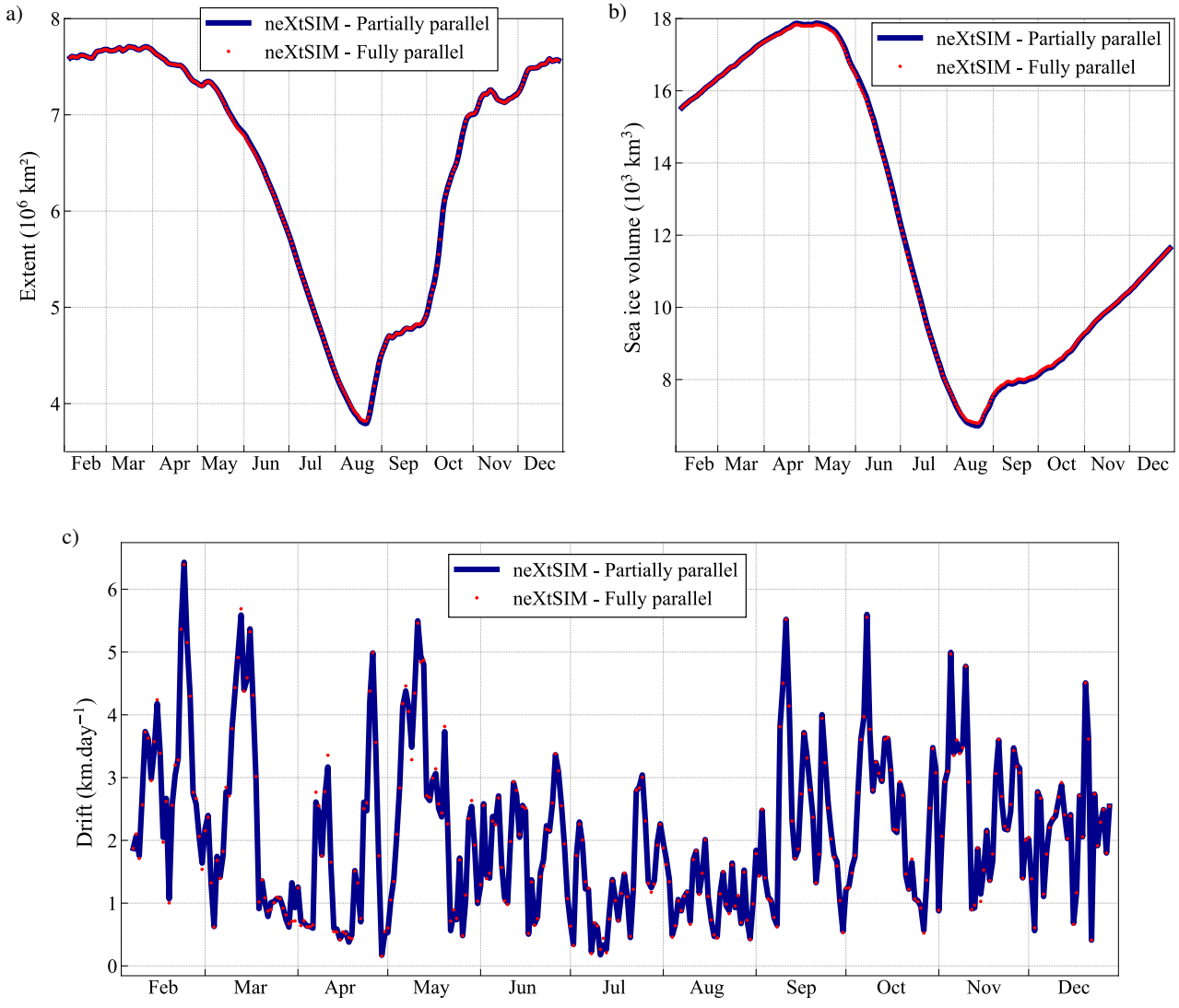

Figure 15. Comparison of numerical solutions obtained throughout 2006 on a mesh composed of 10 km triangles, using the previous, partially parallel version of neXtSIM and the new fully parallel version. a) Sea ice extent, b) Sea ice volume, c) Mean drift.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

ice drift, the inherently more stochastic behavior leads to larger discrepancies between simulations; however, these differences remain within acceptable limits, representing less than 2% of the mean drift values. These results indicate that the integration of the parallel remeshing process within neXtSIM does not introduce errors and shows good reproducibility.

5.2 Computing performance

Figure 16 compares the performance of the previous partially parallel version of neXtSIM with that of the new fully parallel version, for three meshes with element sizes of 1, 2, and 10 km. For both the full model and the complete remeshing procedure, the computation time and the strong scaling are shown for a one-day winter simulation.

With 10-km elements, the fully parallel version of neXtSIM is consistently faster than the initial partially parallel version, meaning that the parallelization benefits also extend to relatively small simulations. With 32 processors, the improvement reaches about 17%. The strong-scaling curves (Fig. 16.b) exhibit similar trends for both versions, but the efficiency drop occurs earlier in the partially parallel version—starting at 8 processors—whereas it only appears from 16 processors in the fully parallel version. Although the remeshing procedure ceases to scale beyond 32 processors, its computing cost has been reduced so drastically that it becomes negligible.

The benefits of full parallelization become even more pronounced for finer meshes. For 1-km and 2-km elements, the entire model exhibits near-perfect strong scaling up to 128 processors (Fig. 16.d & f), which is not the case for the partially parallel version. The model can also run on 256 processors, with a loss of efficiency that remains acceptable. The total runtime is consistently lower with the fully parallel version. For 2-km elements, using 128 processors reduces the computation time by a factor greater than 6. For elements of 1 km, the reduction reaches a factor of 9 with 256 processors. These improvements stem primarily from the parallelization of the remeshing step: in the partially parallel version, remeshing accounts for about 90% of the total runtime at 256 processors, whereas in the fully parallel version it has dropped to 25%.

The developments implemented in neXtSIM now make it possible to perform longer and more accurate sea-ice simulations within a fully Lagrangian framework. With a 10-km resolution, a one-year simulation can be completed in about five hours using 32 processors (without optional model components). At 2-km resolution, one year of simulation requires roughly four days on 128 processors. Finally, at 1-km resolution, simulating an entire sea-ice season takes about one week on 128 processors. This runtime naturally varies depending on the simulated period of the year. Figure 17 shows a few examples of sea ice thickness and damage fields obtained from a simulation performed at 1-km mesh resolution throughout the year 2006. The damage field for September is not shown, as it is not meaningful. The results highlight the level of accuracy now achievable with neXtSIM, particularly for the damage field, as well as for the cracks and ridges visible in the ice thickness field.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

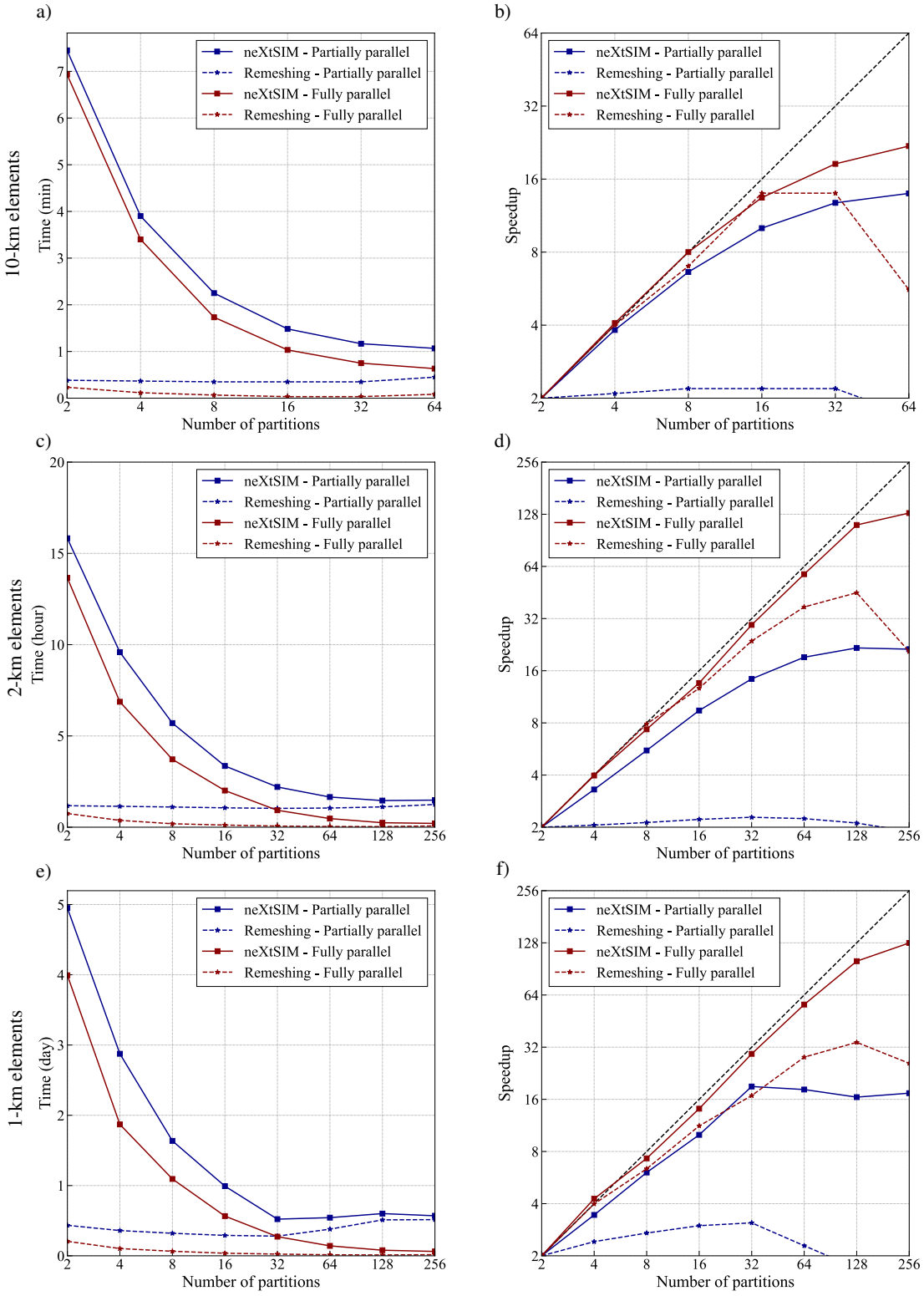

Figure 16. Computation time for one simulation day and corresponding strong scalability of the full neXtSIM model and the complete remeshing workflow, for three meshes composed of 10 km (a & b), 2 km (c & d), and 1 km (e & f) triangular elements.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

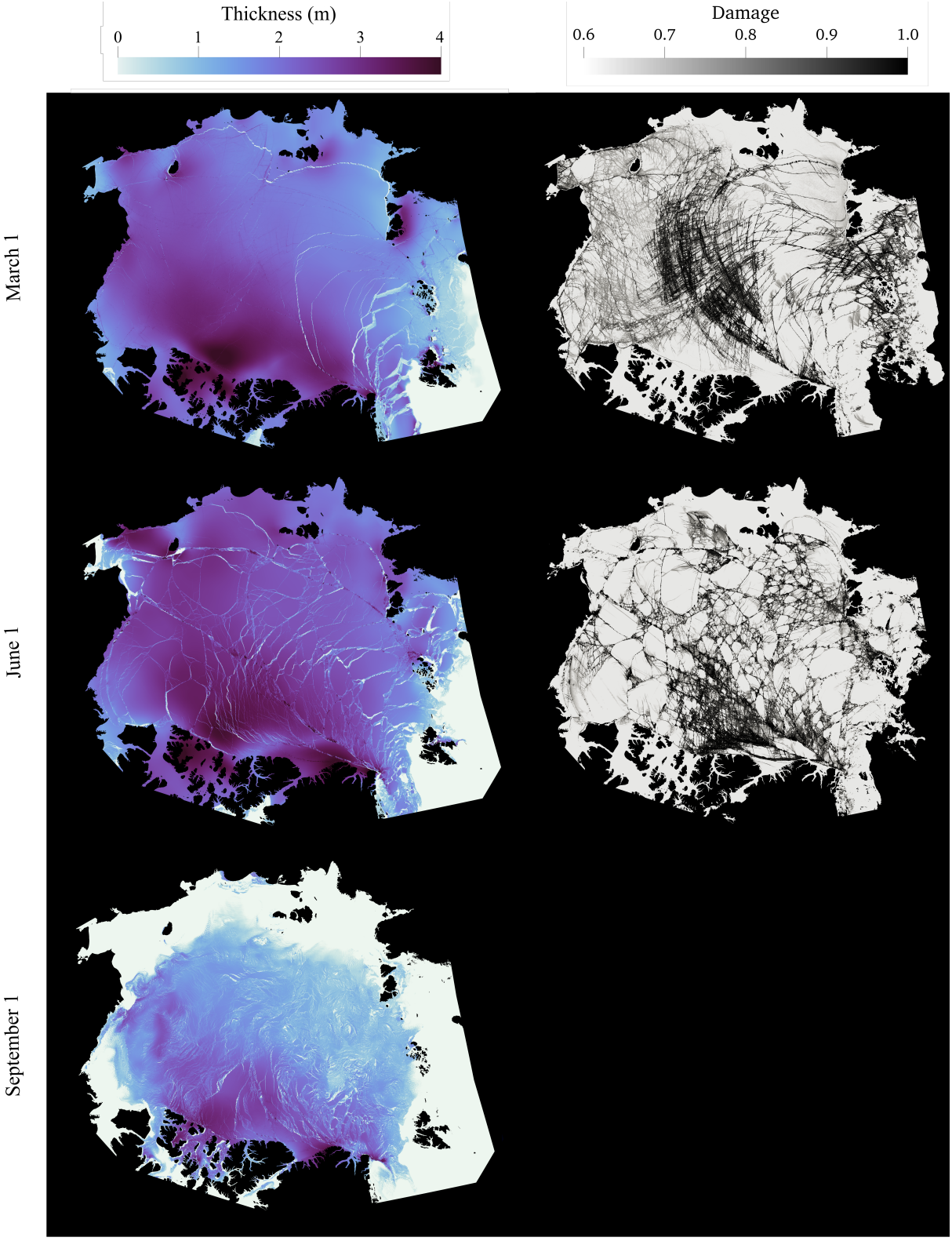

Figure 17. Simulated sea ice thickness and damage on March 1, June 1, and September 1, 2006, using a mesh with 1-km elements. Damage is not shown for the summer month as this variable is not relevant for that period.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

6 Conclusions

Achieving high accuracy in sea-ice dynamics simulations often requires a Lagrangian framework, as used in neXtSIM. However, this approach leads to mesh distortion that must be corrected through frequent remeshing. Because this procedure is computationally expensive, optimizing it is essential to enable highly resolved simulations. This has been the main concern of this paper, where we have described several enhancements within neXtSIM:

- parallelization of the remeshing process using the new tool ParMMG2D
- parallelization of the interpolation from the old mesh to the new mesh
- parallelization of binary and netCDF outputs
- parallelization of drifting buoys tracking

The ParMMG2D mesh-adaptation tool has been developed for sea-ice modeling, which is based on homogeneous and isotropic meshes, but its scope is much broader. It can adapt meshes to any physical process using anisotropic and heterogeneous metrics. Built on a frozen-interface parallel strategy combined with an advancing-front algorithm, the tool has demonstrated strong parallel performance and excellent scalability, even for anisotropic meshes.

The parallelization of the remaining components of neXtSIM also delivers strong performance gains, particularly for fine meshes, where parallel efficiency is the most impactful. For example, for 1-km meshes, the interpolation step is accelerated by a factor of up to 30. These developments significantly reduce the overall runtime—not only for highly resolved meshes but also for standard configurations using 10-km elements. The scalability of neXtSIM has been markedly improved, now remaining efficient up to 256 processors, in contrast with the previously partially parallel version. Despite the computational cost inherent to the Lagrangian framework, it is now feasible to perform 1-year simulations at 2-km resolution within a few days, and to use 1-km elements for seasonal simulations over similar time lengths.

neXtSIM is now approaching the limits of the current computational capabilities. An improvement is the optimization of the input of environmental data, notably by reading only the necessary data, thus reducing memory usage and accelerating the process. In the future, neXtSIM could further benefit from the full capabilities of ParMMG2D by incorporating weakly anisotropic and heterogeneous meshes in sea ice dynamics simulations.

Code and data availability.

- The source code of the previous partially parallel version of neXtSIM is available on Zenodo (Ólason et al. (2025b), https://doi.org/10.5281/zenodo.14724536)
- The source code of the new fully parallel version of neXtSIM is available on Zenodo (Salmon (2026a), https://doi.org/10.5281/zenodo.18415193). The model is also available on https://github.com/nansencenter/nextsim. The code is released under the MIT License

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

- The ParMMG2D software source code is available on Zenodo (Salmon (2026b),https://doi.org/10.5281/zenodo.18154997). It is also available on https://github.com/MmgTools/parMmg2D. The code is released under the GNU Lesser General Public License

Author contributions. PR and NB supervised the research activities. FS carried out the full development of ParMMG2D and implemented the corresponding enhancements in the new version of neXtSIM. FS and TW oversaw the integration of these developments into the public neXtSIM repository. FS wrote most of the original manuscript. All authors reviewed and edited the manuscript, with EÓ and TW who contributed more specifically to Section 2.1. PR and NB secured the funding for the project.

Competing interests. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest

Acknowledgements. This study was financially supported by the Institut des Mathématiques pour la Planète Terre (IMPT).

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

References

Benard, P., Balarac, G., Moureau, V., Dobrzynski, C., Lartigue, G., and d'Angelo, Y.: Mesh adaptation for large-eddy simulations in complex geometries, International journal for numerical methods in fluids, 81, 719–740, https://doi.org/10.1002/fld.4204, 2016.

Bouillon, S. and Rampal, P.: Presentation of the dynamical core of neXtSIM, a new sea ice model, Ocean Modelling, 91, 23–37, https://doi.org/10.1016/j.ocemod.2015.04.005, 2015.

Boutin, G., Williams, T., Rampal, P., Olason, E., and Lique, C.: Wave–sea-ice interactions in a brittle rheological framework, The Cryosphere, 15, 431–457, https://doi.org/10.5194/tc-15-431-2021, 2021.

Boutin, G., Ólason, E., Rampal, P., Regan, H., Lique, C., Talandier, C., Brodeau, L., and Ricker, R.: Arctic sea ice mass balance in a new coupled ice–ocean model using a brittle rheology framework, The Cryosphere, 17, 617–638, https://doi.org/10.5194/tc-17-617-2023, 2023.

Casagrande, A., Leyland, P., Formaggia, L., and Sala, M.: Parallel mesh adaptation, Series on Advances in Mathematics for Applied Sciences, 69, 201, https://doi.org/10.1142/9789812701817_0019, 2005.

Cavallo, P. A., Sinha, N., and Feldman, G. M.: Parallel unstructured mesh adaptation method for moving body applications, AIAA journal, 43, 1937–1945, https://doi.org/10.2514/1.7818, 2005.

Chevalier, C. and Pellegrini, F.: PT-Scotch: A tool for efficient parallel graph ordering, Parallel Computing, 34, 318–331, https://doi.org/10.1016/j.parco.2007.12.001, 2008.

Chrisochoides, N. and Nave, D.: Parallel delaunay mesh generation kernel, International Journal for Numerical Methods in Engineering, 58, 161–176, https://doi.org/10.1002/nme.765, 2003.

Cirrottola, L. and Froehly, A.: Parallel unstructured mesh adaptation using iterative remeshing and repartitioning, Research Report RR-9307, INRIA Bordeaux, équipe CARDAMOM, https://inria.hal.science/hal-02386837, 2019.

Comiso, J. C.: Large Decadal Decline of the Arctic Multiyear Ice Cover, Journal of Climate, 25, 1176–1193, https://doi.org/10.1175/JCLI-D-11-00113.1, 2012.

Cornejo, A., Mataix, V., Zárate, F., and Oñate, E.: Combination of an adaptive remeshing technique with a coupled FEM–DEM approach for analysis of crack propagation problems, Computational Particle Mechanics, 7, 735–752, https://doi.org/10.1007/s40571-019-00306-4, 2020.

Coupez, T., Digonnet, H., and Ducloux, R.: Parallel meshing and remeshing, Applied Mathematical Modelling, 25, 153–175, https://doi.org/10.1016/S0307-904X(00)00045-7, 2000.

Dapogny, C., Dobrzynski, C., and Frey, P.: Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems, Journal of Computational Physics, 262, 358–378, https://doi.org/10.1016/j.jcp.2014.01.005, 2014.

Digonnet, H., Coupez, T., Laure, P., and Silva, L.: Massively parallel anisotropic mesh adaptation, The International Journal of High Performance Computing Applications, 33, 3–24, https://doi.org/10.1177/1094342017693906, 2019.

Frey, P. J. and George, P.-L.: Mesh generation: application to finite elements, Iste, https://doi.org/10.1002/9780470611166, 2007.

George, P. L., Hecht, F., and Vallet, M. G.: Creation of internal points in Voronoi's type method. Control adaptation, Advances in engineering software and workstations, 13, 303–312, https://doi.org/10.1016/0961-3552(91)90034-2, 1991.

Hecht, F.: BAMG: bidimensional anisotropic mesh generator, User Guide. INRIA, Rocquencourt, 17, 1998.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren,

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J.-N.: The ERA5 global reanalysis, Quarterly Journal of the Royal Meteorological Society, 146, 1999–2049, https://doi.org/10.1002/qj.3803, 2020.

Hibler, W. D.: A dynamic thermodynamic sea ice model, Journal of physical oceanography, 9, 817–846, https://doi.org/10.1175/1520-0485(1979)009<0815:ADTSIM>2.0.CO;2, 1979.

Ibanez, D. A.: Conformal mesh adaptation on heterogeneous supercomputers, Ph.D. thesis, Rensselaer Polytechnic Institute, https://doi.org/10.13140/RG.2.2.20170.98243, 2016.

Karypis, G. and Kumar, V.: METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, 1998.

Karypis, G., Schloegel, K., and Kumar, V.: PARMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library, 1997.

Kwok, R.: Arctic sea ice thickness, volume, and multiyear ice coverage: Losses and coupled variability (1958-2018), Environmental research letters, 13, 105 005, https://doi.org/10.1088/1748-9326/aae3ec, 2018.

Legentil, C., Pellerin, J., Raguenel, M., and Caumon, G.: Towards a workflow to evaluate geological layering uncertainty on CO2 injection simulation, Applied Computing and Geosciences, 18, 100 118, https://doi.org/10.1016/j.acags.2023.100118, 2023.

Lemieux, J.-F., Tremblay, L. B., Dupont, F., Plante, M., Smith, G. C., and Dumont, D.: A basal stress parameterization for modeling landfast ice, Journal of Geophysical Research: Oceans, 120, 3157–3173, https://doi.org/10.1002/2014JC010678, 2015.

Loseille, A. and Alauzet, F.: Continuous mesh framework part I: well-posed continuous interpolation error, SIAM Journal on Numerical Analysis, 49, 38–60, https://doi.org/10.1137/090754078, 2011a.

Loseille, A. and Alauzet, F.: Continuous mesh framework part II: validations and applications, SIAM Journal on Numerical Analysis, 49, 61–86, https://doi.org/10.1137/10078654X, 2011b.

MMG: platform website, https://www.mmgtools.org, 2025.

Moureau, V., Lartigue, G., Bénard, P., and Mercier, R.: Parallel and dynamic mesh adaptation of tetrahedral-based meshes for propagating fronts and interfaces: application to premixed combustion, in: 32nd International Conference on Parallel Computational Fluid Dynamics (ParCFD'2021), 2021.

ParMMG2D: platform website, https://github.com/MmgTools/parMmg2D, 2025.

Pellegrini, F.: Scotch and PT-Scotch Graph Partitioning Software: An Overview, in: Combinatorial Scientific Computing, edited by Uwe Naumann, O. S., pp. 373–406, Chapman and Hall/CRC, https://doi.org/10.1201/b11644-15, 2012.

Rampal, P., Bouillon, S., Ólason, E., and Morlighem, M.: neXtSIM: a new Lagrangian sea ice model, The Cryosphere, 10, 1055–1073, https://doi.org/10.5194/tc-10-1055-2016, 2016.

Rampal, P., Dansereau, V., Ólason, E., Bouillon, S., Williams, T., Korosov, A., and Samaké, A.: On the multi-fractal scaling properties of sea ice deformation, The Cryosphere, 13, 2457–2474, https://doi.org/10.5194/tc-13-2457-2019, 2019.

Rauff, A., Herron, M. R., Maas, S. A., and Weiss, J. A.: An algorithmic and software framework to incorporate orientation distribution functions in finite element simulations for biomechanics and biophysics, Acta Biomaterialia, 192, 151–164, https://doi.org/10.1016/j.actbio.2024.11.043, 2025.

Rheinlænder, J. W., Davy, R., Ólason, E., Rampal, P., Spensberger, C., Williams, T. D., Korosov, A., and Spengler, T.: Driving Mechanisms of an Extreme Winter Sea Ice Breakup Event in the Beaufort Sea, Geophysical Research Letters, 49, https://doi.org/10.1029/2022GL099024, 2022.

https://doi.org/10.5194/egusphere-2026-1869
Preprint. Discussion started: 5 June 2026

Sakov, P., Counillon, F., Bertino, L., Lister, K. A., Oke, P. R., and Korablev, A.: TOPAZ4: An ocean sea ice data assimilation system for the North Atlantic and Arctic, Ocean Science, 8, 633–656, https://doi.org/10.5194/os-8-633-2012, 2012.

Salmon, F.: NeXtSIM, fully parallelized version, Zenodo[code], https://doi.org/10.5281/zenodo.18415193, 2026a.

Salmon, F.: MMG 2D parallel version, Zenodo[code], https://doi.org/10.5281/zenodo.18154996, 2026b.

Salmon, F. and Barral, N.: ParMMG2D: functionality overview and user parameters, Tech. rep., INRIA, Bordeaux, submitted.

Samaké, A., Rampal, P., Bouillon, S., and Ólason, E.: Parallel implementation of a Lagrangian-based model on an adaptive mesh in C++: Application to sea-ice, Journal of Computational Physics, 350, 84–96, https://doi.org/10.1016/j.jcp.2017.08.055, 2017.

Santana, R., Boutin, G., Horvat, C., Ólason, E., Williams, T., and Rampal, P.: Modeling Antarctic Sea Ice Variability Using a Brittle Rheology, Journal of Advances in Modeling Earth Systems, 17, https://doi.org/10.1029/2024MS004584Digital Object Identifier (DOI), 2025.

Wheel, I., Benn, D. I., Crawford, A. J., Todd, J., and Zwinger, T.: A new 3D full-Stokes calving algorithm within Elmer/Ice (v9.0), Geoscientific Model Development, 17, 5759–5777, https://doi.org/10.5194/gmd-17-5759-2024, 2024.

Williams, T. D., Rampal, P., and Bouillon, S.: Wave–ice interactions in the neXtSIM sea-ice model, The Cryosphere, 11, 2117–2135, https://doi.org/10.5194/tc-11-2117-2017, 2017.

Zhang, J. and Rothrock, D. A.: Modeling global sea ice with a thickness and enthalpy distribution model in generalized curvilinear coordinates, Monthly Weather Review, 131, 681–697, https://doi.org/10.1175/1520-0493(2003)131<0845:MGSIWA>2.0.CO;2, 2003.

Ólason, E., Boutin, G., Korosov, A., Rampal, P., Williams, T., Kimmritz, M., Dansereau, V., and Samaké, A.: A new brittle rheology and numerical framework for large-scale sea-ice models, Journal of Advances in Modeling Earth Systems, 14, 1055–1073, https://doi.org/10.1029/2021MS002685, 2022.

Ólason, E., Boutin, G., Williams, T., Korosov, A., Regan, H., Rheinlaender, J., Rampal, P., Flocco, D., Samaké, A., Davy, R., Spain, T., and Chua, S.: The next generation sea-ice model neXtSIM, version 2, EGUsphere, pp. 1–33, https://doi.org/10.5194/egusphere-2024-3521, 2025a.

Ólason, E., Boutin, G., Williams, T., Korosov, A., Regan, H., Rheinlænder, J. W., Rampal, P., Flocco, D., Samaké, A., Davy, R., Timothy, S., and Chua, S.: NeXtSIM, partially parallelized version, Zenodo[code], https://doi.org/doi.org/10.5281/zenodo.14724536, 2025b.