A Parallel Hierarchical-Element Method for Contour Dynamics Simulations

R.M. Schoemaker[◇,⋆], P.C.A. de Haas[◇], H.J.H. Clercx[⋆], and R.M.M. Mattheij[◇]

[◇] Dept. of Mathematics and Computing Science and
[⋆] Dept. of Applied Physics
Eindhoven University of Technology, The Netherlands

Abstract

Many two-dimensional incompressible inviscid vortex flows can be simulated with high efficiency by means of the contour dynamics method. Several applications require the use of a hierarchical-element method (HEM), a modified version of the classical contour dynamics scheme by applying a fast multipole method, in order to accelerate the computations substantially. The HEM can be used, for example, to study the large-scale motion of coherent structures in geophysical fluid dynamics where the flow can be modelled by considering the motion in a thin layer of fluid in the presence of a non-uniform background vorticity (due to the latitudinal variation of the Coriolis force). Nevertheless, such simulations demand a substantial computational effort, even when the HEM is used. In this paper it is shown that the acceleration of contour dynamics simulations by means of the HEM can be increased further by parallelising the HEM algorithm. Speed-up, load balance and scalability are parallel performance features which are studied for several test examples. Furthermore, typical simulations are shown, including an application of vortex dynamics near the pole of a rotating sphere. The HEM has been parallelised using OpenMP and tested with up to 16 processors on an Origin 3800 cc-NUMA computer.

1 Introduction

Large-scale vortices are coherent structures that can be found in the oceans, the atmosphere of our planet as well as in the atmosphere of other planets. Examples of terrestrial nature are high and low-pressure areas, eddies in the ocean and the large polar vortex. Other planetary examples are the Great Red Spot on Jupiter, the Great Dark Spot on Neptune and huge rotating dust structures on Mars. The thickness of the layer of fluid these structures evolve in (on Earth: 1 – 10 km), is small compared to the horizontal size of the coherent structure itself (on Earth: 100 – 1000 km). This geometrical confinement, together with the planetary rotation and the density stratification of the fluid layer, implies quasi two-dimensionality of the flow. The motion of such large-scale structures is slow compared to the rotation speed of the planetary body implying that these structures are nearly non-divergent. Their dynamics can in good approximation be described by the two-dimensional incompressible inviscid variant of the Navier-Stokes equations, viz. the 2D Euler equations.

A suitable and elegant numerical technique for simulating two-dimensional (2D) vortex flows is the contour dynamics method [4, 12]. The collection of 2D vortices is approximated by nested patches of uniform vorticity. Only the evolution of the contours needs to be computed for determining the complete dynamics of the vortices and this makes the method efficient because no grid is needed. However, when highly complicated flow patterns emerge during a simulation, the conventional numerical scheme becomes inefficient in its time-complexity. Another class of problems concerns the evolution of vortices in the presence of non-uniform

background vorticity. The non-uniform background vorticity is usually a local approximation of the latitudinal variation of the Coriolis force. For example, when simulating the dynamics of vortices located near the poles of a rotating sphere (the local approximation is denoted as the γ-plane), the initial vorticity distribution is rather intricate due to the necessity to discretise, together with the vorticity patches, the background vorticity field[1]. An acceleration of the method is in this case already appropriate from the start of the simulation. The hierarchical-element method for contour dynamics (HEM) [9] solves for the limited applicability of the conventional scheme.

A hierarchical-element method is a technique that has the computational structure of the fast multipole method [5, 6], and thus makes use of approximations of far-field contributions of vorticity patches. These approximations—while maintaining a high accuracy—make these schemes very efficient. A hierarchical-element method only differs from the fast multipole technique in their approximating elements [2] whereas the hierarchy of these elements is similar and plays a crucial role in both schemes. The HEM enables acceleration of complicated flow simulations. Moreover, the algorithmic structure of the HEM has certain features that makes parallelisation a good means for speeding up contour dynamics simulations to an even greater extent.

The HEM has been parallelised using OpenMP [1], which is a new industry standard for (incremental) parallel programming on shared-memory architectures. The parallel HEM has been tested for speed-up, scalability and load balancing for several test cases and typical vortex configurations including one with a non-uniform background vorticity field. It is shown in this paper that the parallel HEM is an excellent tool for studying flows with non-uniform background vorticity when using a moderate amount of processors, i.e. up to 16 processors.

The remainder of the paper is as follows. The next section explains the contour dynamics method, the fast multipole technique and the hierarchical-element method for contour dynamics in a concise manner. The time-complexity of this method is being analysed in Section 3. Then in Section 4 the parallelisation of the HEM is discussed. Results of numerical experiments, discussion and conclusions are provided in Sections 5 and 6.

2 Contour Dynamics

The spatial discretisation used in a contour dynamics method consists of three parts: The discretisation of the continuous vorticity profile into a piecewise-uniform vorticity distribution, interpolation of the bounding contours of the regions of uniform vorticity, and the redistribution of the nodes on the contours during the simulation. For the present introduction to contour dynamics and the HEM it is sufficient to focus on the first part of the spatial discretisation. Discussion of the two other aspects of the spatial discretisation of presently used contour dynamics algorithm is briefly described. A detailed analysis can be found in Ref. [8] as these aspects are not essential for understanding the parallelisation strategy applied to the HEM algorithm.

2.1 Contour dynamics for piecewise-uniform vorticity distributions

The dynamics of a 2D incompressible, inviscid fluid flow is described by the Euler equation and the equation of mass conservation. The latter implies a divergence-free velocity field or $\nabla \cdot \mathbf{u} = 0$, with $\mathbf{u}(\mathbf{r}, t)$ the velocity vector. The Euler equation, which expresses the balance of

[1] More precisely, the potential vorticity is the conserved quantity and should thus be discretised [10].

linear momentum, is then written as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p \,, \tag{1}$$

with p the pressure and ρ the (constant) density. The vorticity is defined as $\boldsymbol{\omega}(\mathbf{r}, t) = \nabla \times \mathbf{u}$. For a 2D flow, $\mathbf{u} = (u, v)^T$, $\mathbf{r} = (x, y)^T$ and $\boldsymbol{\omega} = \omega\, \mathbf{e}_z$. The non-divergence condition implies the definition of a stream function $\psi(\mathbf{r}, t)$ through $u = \frac{\partial \psi}{\partial y}$ and $v = -\frac{\partial \psi}{\partial x}$. By taking the curl of (1) we obtain an expression for conservation of vorticity of a fluid particle in two dimensions, viz.

$$\frac{D\omega}{Dt} = \frac{\partial \omega}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} = 0 \,, \tag{2}$$

whereas the vorticity ω and the stream function ψ are related through the Poisson equation

$$\nabla^2 \psi = -\omega \,. \tag{3}$$

Using the Green's function of the Laplace operator for an infinite 2D domain, $G(\mathbf{r}; \mathbf{r}') = \frac{1}{2\pi} \ln |\mathbf{r} - \mathbf{r}'|$ (with $|\mathbf{r}| = \sqrt{x^2 + y^2}$), the stream function can be found explicitly as

$$\psi(\mathbf{r}, t) = -\frac{1}{2\pi} \iint_{\mathbb{R}^2} \omega(\mathbf{r}', t)\, \ln |\mathbf{r} - \mathbf{r}'|\, dx' dy' \,. \tag{4}$$

As is depicted in Figure 1, an initially continuous vorticity distribution $\omega(\mathbf{r}, 0)$ is approximated by a piecewise-uniform distribution $\hat{\omega}(\mathbf{r}, 0) = \sum_{m=0}^{M} \omega_m$. It consists in this case of a constant background vorticity ω_0 and a number of nested patches $\mathcal{P}_m$ with vorticity values ω_m (with $m \geq 1$).

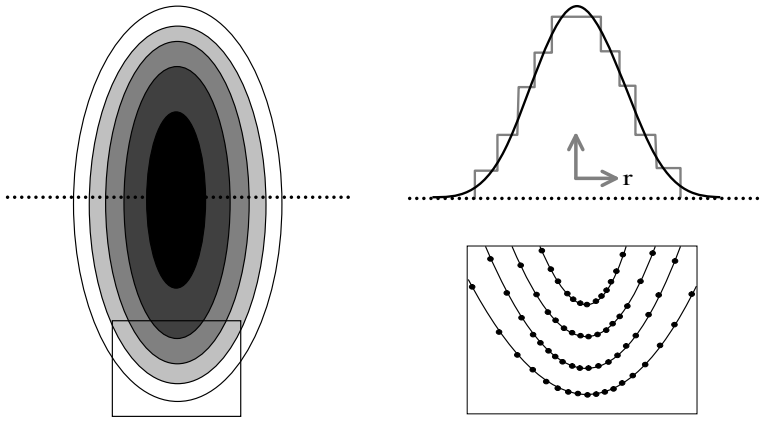

Figure 1 – *Five nested patches with uniform vorticity. Right-top: Vorticity jumps. Right-bottom: Node redistribution.*

We assume for the moment that no background vorticity is present, i.e. $\omega_0 = 0$. Due to the conservation of vorticity the piecewise-uniform distribution of vorticity remains piecewise-uniform in the course of time. The nested patches $\mathcal{P}_m(t)$ deform during the flow evolution although its area is conserved, and the bounding contours $\mathcal{C}_m(t)$ of the patches $\mathcal{P}_m(t)$ will not cut neighbouring boundary contours. Equation (4) can be reformulated as

$$\psi(\mathbf{r}, t) = -\frac{1}{2\pi} \sum_{m=1}^{M} \omega_m \iint_{\mathcal{P}_m(t)} \ln |\mathbf{r} - \mathbf{r}'|\, dx' dy' \,. \tag{5}$$

The velocity field $\mathbf{u}(\mathbf{r},t)$ is obtained by taking the derivatives of $\psi(\mathbf{r},t)$ with respect to x and y and subsequently applying Stokes' theorem for a scalar field. The following expression can be derived [4, 9, 12]:

$$\mathbf{u}(\mathbf{r},t) = -\frac{1}{2\pi}\sum_{m=1}^{M}\omega_m \oint_{\mathcal{C}_m(t)} \ln|\mathbf{r}-\mathbf{r}'|\,d\mathbf{r}' \ , \tag{6}$$

where $d\mathbf{r}'$ denotes an infinitesimal vector tangential to the boundary. From (6) it follows that the evolution of patches of uniform vorticity is fully determined by the evolution of their bounding contours.

For completeness we briefly mention a few other aspects of the present contour dynamics algorithm, such as the contour discretisation, the node (re)distribution and the time integration scheme. The parallelisation strategy of the classical contour dynamics or the HEM algorithm is independent of the technical details of these aspects. The contour integrals are discretised into nodes where particular curvature thresholds are used to determine the precise distribution of the nodes over the contour. The contour element between two nodes is interpolated by linear elements. For a full account of the contour discretisation and interpolation procedure used in the present contour dynamics algorithm we refer to the analysis of the interpolation method (including an error analysis) and the actual implementation procedure provided by Vosbeek [8]. Although we usually start with rather simple shapes of the vorticity patches, the contours $\mathcal{C}_m(t)$ become increasingly complex in course of time. Especially the length of the contours will increase (while keeping the area of the patch constant) and segments of high curvature will inevitably arise. The original node distribution, used to approximate the initial contours, become inadequate when the simulation evolves. In order to meet certain accuracy demands the number of nodes on a contour and the node distribution will be adapted during the simulation. The followed procedure is sketched by Vosbeek [8]. Together with a symplectic time-integration scheme, whose implementation is discussed by Vosbeek and Mattheij [11], a rather high accuracy of the contour dynamics simulation is guaranteed for many 2D vortex flow problems.

The number of nodes plays an essential role because the integration of each contour element between two nodes yields a velocity contribution for every other node in the domain, implying a time-complexity of $\mathcal{O}(N^2)$. During a simulation the number of contour nodes N may grow in time due to the increasing complexity of the piecewise-uniform vorticity distribution. For highly complicated flow patterns, and hence huge numbers of nodes, the conventional contour dynamics approach is therefore of limited use, and an alternative implementation of the contour dynamics method is necessary.

2.2 Fast Multipole and Hierarchical-Element Methods

Interactions between charged particles, between gravitating bodies, or between point vortices are examples of N-body interactions, with N the total number of objects (particles, bodies, vortices etc.). In a numerical attempt to simulate the dynamics of, for example, an N-body system of charged particles, located at the positions $\mathbf{r}_i$ and with (different) charges κ_i (with $i = 1,\ldots,N$), the following expression for the potential (in two dimensions) has to be evaluated at the positions of the N charged particles,

$$\Psi(\mathbf{r}_j) = \frac{1}{2\pi}\sum_{i=1}^{N}\kappa_i \ln|\mathbf{r}_j-\mathbf{r}_i| \ , \quad j=1,\ldots,N \ . \tag{7}$$

With an explicit calculation of all mutual interactions, this simulation requires $\mathcal{O}(N^2)$ operations for computing the potential $\Psi(\mathbf{r}_j)$, at the positions of the N charged particles, due to all the other particles. A similar expression, with a similar computational cost, can be derived for the case of N point vortices.

These computations can be accelerated using a Fast Multipole Algorithm (FMA). In that case the evaluation of the potential $\Psi(\mathbf{r}_j)$ requires $\mathcal{O}(N \log N)$ operations[2], see Greengard and Rohklin [5]. The FMA is based on the concept of combining large numbers of particles into a single computational element—a multipole expansion—if the evaluation point for an effect due to this cluster of particles is far away. Such a multipole expansion for the potential of a cluster of particles centered at the origin in a two dimensional plane has the form

$$\Psi(\mathbf{r}) \doteq a \ln|\mathbf{r}| + \sum_{k=1}^{K} \frac{b_k}{\mathbf{r}^k} \quad \text{with} \quad \left\{ \begin{array}{lcl} a & = & \frac{1}{2\pi} \sum_{i=1}^{N} \kappa_i \\ b_k & = & \frac{1}{2\pi} \sum_{i=1}^{N} \frac{-\kappa_i \mathbf{r}_i^k}{k} \end{array} \right. , \tag{8}$$

where N bodies with strengths κ_i reside in a circle with radius R and $|\mathbf{r}| > R$. The accuracy of this multipole approximation depends mainly on K, which represents the order of the multipole, but also on $\mathbf{r}$ and κ_i.

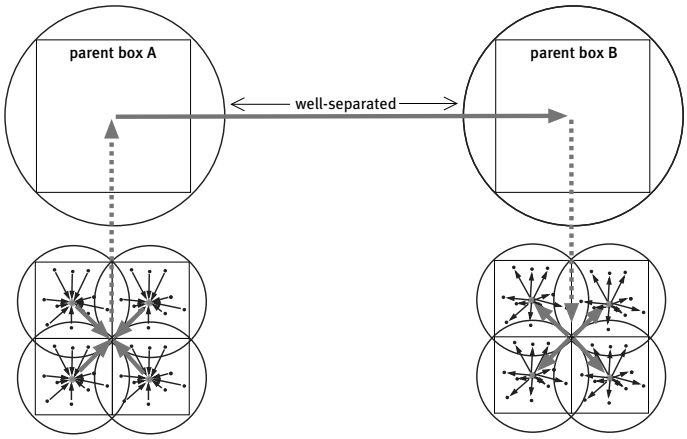

Figure 2 – *A schematic picture of the Fast Multipole Algorithm mechanism.*

The above multipole expansions can be manipulated in the sense that the expansions can be shifted to the centres of larger circles. Furthermore, they can be transformed into local (Taylor) expansions far away (well-separated) from the original multipole centre. These Taylor expansions can on their turn be shifted to the centres of smaller circles. The manipulations of the cluster potential are coordinated in a hierarchical way to obtain accurate approximations throughout the complete domain. That is, the farther away the evaluation point for the potential of a cluster of particles, the larger the cluster is. A hierarchical ordering demands that larger cluster multipole expansions need to be build from smaller cluster expansions. Note that these smaller cluster expansions are also needed for the evaluation of the potential of points which are closer by. The local (Taylor) expansions can be shifted to smaller clusters in order to pass on all approximating contributions from the far field. Figure 2 shows a schematic picture of the basic mechanism of the FMA. For a detailed discussion of the FMA, see Greengard and Rohklin [5].

The multipole expansion of the FMA can be exchanged for any other series approximation, stated that this series converges. If the multipole expansion is exchanged for a Poisson

[2] Under certain circumstances also $\mathcal{O}(N)$, see also Hrycak and Rohklin [6].

integral (Poisson's formula), the same hierarchical mechanism as in the FMA holds (see Anderson [2]). The computational elements are then of a different structure, viz. rings. Poisson's formula can be written in an exact form as

$$\Psi(r,\phi) = \frac{\ln r}{2\pi}\sum_{i=1}^{N}\kappa_i + \frac{1}{2\pi}\int_0^{2\pi}\left(\Psi(a,\theta) - \frac{\ln(a)}{2\pi}\sum_{i=1}^{N}\kappa_i\right)\frac{1-(\frac{a}{r})^2}{1-\frac{2a}{r}\cos(\phi-\theta)+(\frac{a}{r})^2}\,d\theta\,, \quad (9)$$

for $r > a$, and calculates the potential outside a disk with radius a from sources inside the disk. For $r < a$, the potential can be calculated inside a disk due to sources outside the disk according to

$$\Psi(r,\phi) = \frac{1}{2\pi}\int_0^{2\pi}\Psi(a,\theta)\frac{1-(\frac{r}{a})^2}{1-\frac{2r}{a}\cos(\phi-\theta)+(\frac{r}{a})^2}\,d\theta, \quad \text{for } r < a\,. \quad (10)$$

Special integration methods for the integrals in (9) and (10) are discussed in Refs. [2, 9]. The main advantage of this approach is that for the sources not only particles or other point sources can be used inside (or outside) a disk but also given areas of source distributions, like continuous charge distributions or vorticity distributions.

2.3 The HEM for Contour Dynamics

For contour dynamics the source distribution consists of patches of uniform vorticity. For the implementation of an acceleration scheme based on Poisson integrals, Vosbeek *et al.* [9] developed the Hierarchical-Element Method in order to reduce the $\mathcal{O}(N^2)$ operation count typical of the conventional contour dynamics approach. The HEM is an adaptation to the method of Anderson and has a time-complexity of $\mathcal{O}(N)$. We present here an overview of the general structure of the HEM which is relevant for the parallelisation strategy. Further details about the HEM can be found in Ref. [9].

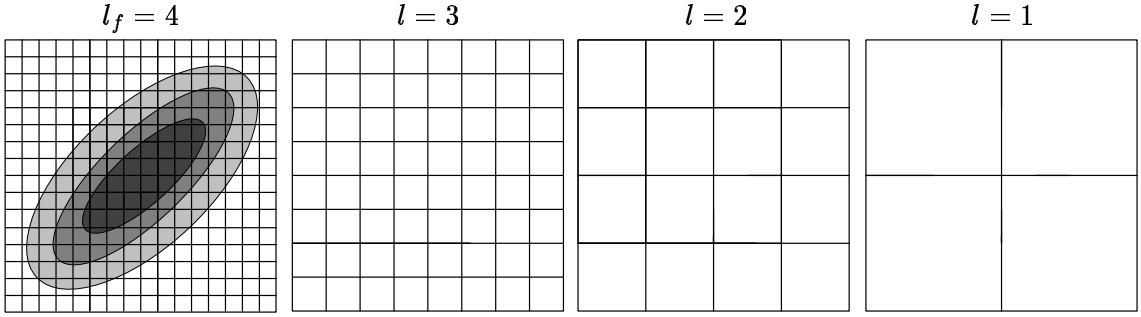

Figure 3 – *The HEM levels and the three nested patches residing at a finest level.*

In the HEM all the patches of uniform vorticity are assumed to reside in a square numerical domain, while the dynamics still involves the infinite plane. This square domain is then subdivided into a set of hierarchical levels with $2^l \times 2^l$ boxes at levels $l = 1, \ldots, l_f$ with l_f a finest level, as is shown in figure 3. Keeping in mind that the numerical approach is based on the use of rings as computational elements (and using Poisson integrals), we introduce at the finest level l_f a ring with K nodes around each box (Figure 4a). K should be of the order of the number of nodes inside one box, ensuring the high efficiency of the method. By means of

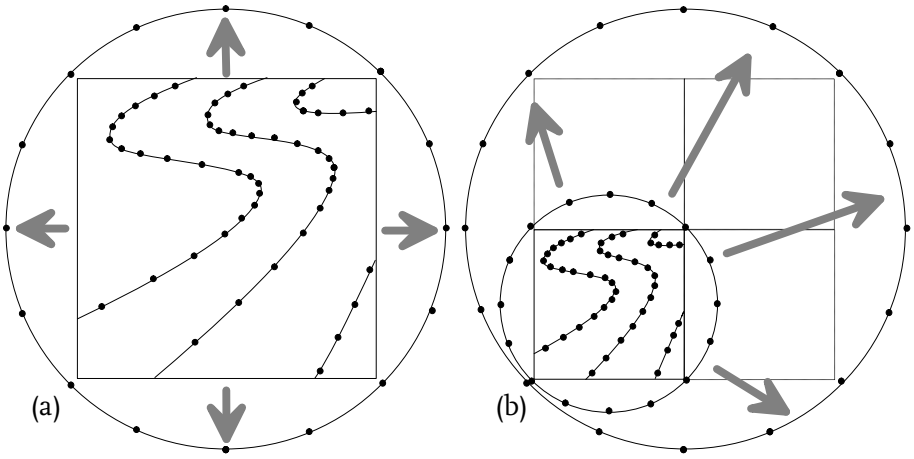

Figure 4 – (a) *Construction of an outer ring around one of the boxes at the finest level by means of direct evaluation.* (b) *Construction of a ring at a consecutive coarser level. Four smaller rings contribute to one coarser ring.*

direct evaluation using (6), the velocity is determined at each node on the ring. These rings are called the outer rings.

Four outer rings at the finest level yield the input for the construction of one outer ring at the subsequent coarser level, the parent level, through the Poisson integral (9). In a similar way outer rings are constructed up to the level $l = 2$ (see Figure 4b). At level $l = 2$, another kind of ring is formed via the constructed outer rings. This ring, a so-called inner ring, is to be used for the construction of inner rings at finer levels again all the way down to level $l = l_f - 1$ by means of (10). This is illustrated in Figure 5 (for $l_f = 4$), at levels $l = 3, \ldots, l_f - 1$ an inner

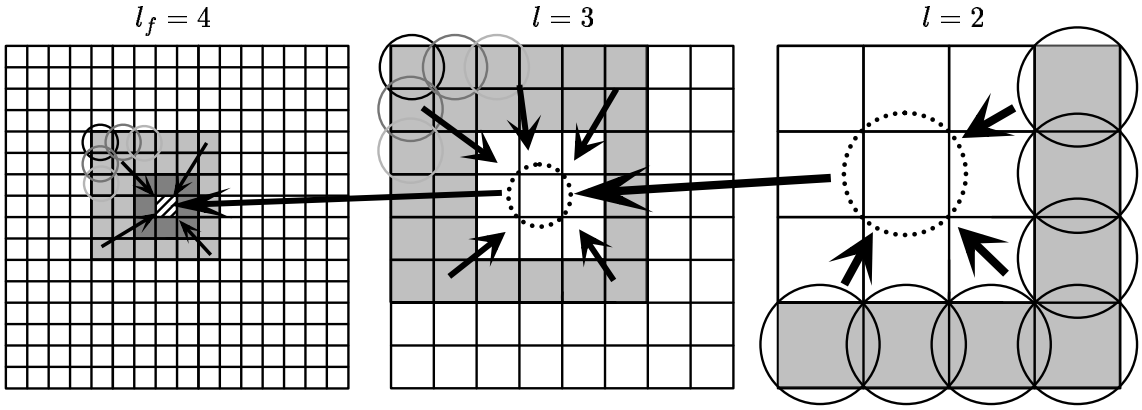

Figure 5 – *The HEM in action for $l_f = 4$. The little box with slanted lines contains evaluation points. Solid rings are outer rings and dotted rings are inner rings. See text for further details.*

ring is constructed from a parent inner ring at level $l - 1$ and from outer rings at the same level l. Level $l = 1$ (four boxes) is irrelevant in these hierarchical approximations. The light grey area at each level in Figure 5 is a so-called well-separated area. It is the area with just the optimal distance for a certain box size (see Ref. [9] for details). The eight dark grey boxes at the finest level are the immediate neighbours of the box containing the evaluation points. Here, direct evaluation of velocities are needed which require $\mathcal{O}(mn)$ operations with m the number of nodes in the eight neighbouring boxes and n the number of nodes in the evaluation box.

Outer rings and inner rings are both needed in the hierarchical tree of levels to calculate all the contributions from the subdivided vorticity at the finest level. The construction of the outer rings is necessary to provide information for all the approximating rings. The inner rings contain all information from the outer regions and pass this information on to the next finer level inner ring.

If a small number of nodes makes up the vorticity field, it is inefficient to use a very fine meshed finest level with many boxes. In this case, the HEM is even more inefficient than the conventional contour dynamics scheme. On the other hand, however, too many nodes in a box is not efficient either because of the expensive direct interaction computations in each small domain of nine boxes at the finest level. The HEM accounts for these two restrictions by determining the hierarchical tree depth in an adaptive manner during a simulation. Optimal intervals for N are being chosen in such a way that K keeps the same order as the number of nodes per box. This leads to an $\mathcal{O}(N)$ method.

3 Algorithm Complexity

The HEM mainly consists of two parts. The most time consuming part of the method is the computation of the velocities, whereas the other much smaller part deals with the redistribution of the nodes. The node redistribution algorithm is identical to the one used in the conventional CD method. See Vosbeek [8] for a detailed discussion on this topic. In order to make a statement about the time-complexity of the HEM, the algorithm for the computation of the velocities at the nodes has been analysed. Identifying those parts within the velocity algorithm that have the largest computational load is of importance when considering parallelisation.

Seven steps can be distinguished in the computation of all the velocities in the HEM tree. Only for a uniform distribution of nodes quantitative results can be obtained, while for non-uniform distributions numerical experiments are needed to assess the time-complexity [2, 9]. For a uniform distribution of nodes, the operation count has been analysed:

1. Finest level direct interactions, with $l_f \geq 1$:
   $C_1 = \left[16 + 24\,(2^{l_f} - 2) + 9\,(2^{l_f} - 2)^2\right] 16^{-l_f} N^2$,
2. Finest level outer ring construction, with $l_f \geq 2$:
   $C_2 = KN$,
3. Finest level contribution by outer rings in the well separated area, with $l_f \geq 2$:
   $C_3 = \left[156 + 528\,(2^{l_f-2} - 1) + 432\,(2^{l_f-2} - 1)^2\right] 4^{-l_f} KN$,
4. Outer ring approximation for levels $l = 2, \ldots, l_f - 1$, with $l_f \geq 3$:
   $C_4 = \frac{4}{3}\left(4^{l_f} - 16\right) K^2$,
5. Inner ring approximation for levels $l = 2, \ldots, l_f - 1$ by outer rings, with $l_f \geq 3$:
   $C_5 = \left[156\,(l_f - 2) + 528\,(2^{l_f-2} - l_f + 1) + 432\,(\frac{1}{3}4^{l_f-2} - 2^{l_f-1} + l_f - \frac{1}{3})\right] K^2$,
6. Inner ring approximation for levels $l = 3, \ldots, l_f - 1$ by inner ring parent, with $l_f \geq 4$:
   $C_6 = \frac{1}{3}\left(4^{l_f} - 64\right) K^2$,

7. Finest level contribution by inner ring parent, with $l_f \geq 3$:

$$C_7 = KN \ .$$

This analysis shows the following asymptotic behaviour for the operation count C_i with increasing l_f: $C_1 \propto 4^{-l_f} N^2$, $C_i \propto KN$ for $i = 2, 3$ and 7, and $C_i \propto 4^{l_f} K^2$ for $i = 4, 5$ and 6. When the total operation count of the seven steps is approximated for large l_f, a quick assessment can be made of the time-complexity $C = \sum_{i=1}^{7} C_i$, i.e.

$$C \propto \alpha 4^{-l_f} N^2 + \beta KN + \gamma 4^{l_f} K^2 \ , \tag{11}$$

with $\alpha = 9$, $\beta = 29$ and $\gamma = \frac{32}{3}$. As can be concluded from the previous paragraph, the three parameters N, K, and l_f determine the time complexity and this is illustrated in Figure 6a (with $K = 17$). These results, obtained for a uniform distribution of nodes, can be compared

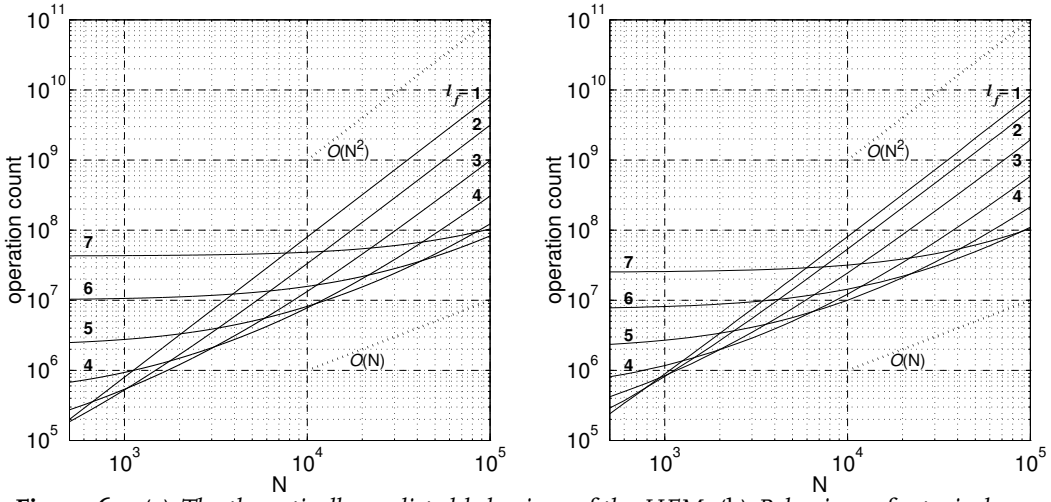

Figure 6 – (a) *The theoretically predicted behaviour of the HEM.* (b) *Behaviour of a typical numerical experiment.*

with a numerical simulation where a non-uniform distribution of nodes is present due to the redistribution of nodes in course of the simulation (see Figure 6b). This comparison supports the claim that the time-complexity of the HEM in numerical simulations, usually with a non-uniform distribution of nodes, is in good agreement with the analytically obtained time-complexity as function of N, K and l_f. The data plotted in Figure 6 do not indicate $\mathcal{O}(N)$ time-complexity for fixed l_f as could be deduced from (11). In that case a large number of nodes always results in an $\mathcal{O}(N^2)$ time-complexity. In order to show the robustness of the $\mathcal{O}(N)$ time-complexity it is necessary to adapt the number of levels l_f during the computation. Usually, this means that l_f increases as the total number of nodes increases. By comparing two subsequent (finest) levels in their intersection point, l_f can be eliminated in (11). This enables the introduction of an estimated operation count C_{ip}, based on the intersection points, which has the following rather simple form:

$$C_{ip} = 53.5KN \ . \tag{12}$$

Formally, this operation count is only valid in the intersection points of the subsequent l_f and, moreover, for sufficiently large l_f. It can be checked, however, that for a relatively small l_f (for $l_f \geq 5$) the estimate (12) is still an acceptable approximation.

As was mentioned at the beginning of this section, it is appropriate to know where the computational load is highest during HEM calculations. It can be deduced from the seven

steps that C_1 and C_3 are the most time-consuming operations during a complete time-step computation. These steps—done at the finest level—represent the operation count of velocity contributions which have to be communicated throughout the domain. This has far-reaching consequences for the parallel behaviour of the HEM as will be highlighted in the following sections.

Before parallelisation of the HEM is discussed, a word on the heuristic approach in handling the different algorithms is appropriate. The velocity computations and the redistribution of the nodes represent between 90% and 99.5% of the computations. The precise amount depends on the flow simulations under consideration. The optimal finest level is determined heuristically each time-step by counting all contributing operations for several l_f in a pre-processing step [9]. This pre-processing step is not part of the parallelisation strategy of the HEM but is of influence in the overall contribution of the velocity and redistribution part. A large contribution of this pre-processing step implies a smaller total contribution of both the velocity and redistribution computations, averaged over a complete simulation.

4 Parallelisation Strategy

The HEM was originally developed for accelerating contour dynamics simulations in a serial computing environment. Parallelisation of the already existing HEM method is achieved by using OpenMP, which is a new industry standard for a set of compiler directives, library routines and environment variables. OpenMP can be used for parallelisation on shared-memory architectures and is especially suited for the incremental parallelisation of existing codes in FORTRAN and C/C++.

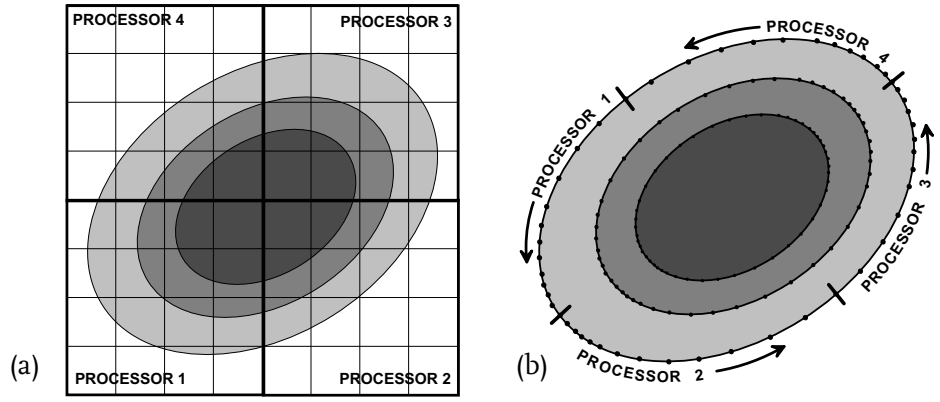

Figure 7 – (a) *Box-wise parallelisation*. (b) *Contour-wise parallelisation*.

It is important to know *a priori* if the numerical scheme can be parallelised in a convenient way. Therefore, the global structure of the HEM should be considered first. Particularly, the parallelisation of the algorithms for the velocity computations and the node redistribution should be taken into account. The numerical structure of the HEM consists of a hierarchy of levels with boxes. At the finest level these boxes contain pieces of contours, whereas coarser boxes contain the approximating outer and inner rings. Velocity computations are carried out at each level in the tree and parallelisation along all hierarchical boxes seems appropriate. Although most of the computations are at the finest level, the ring computations at each consecutive coarser level—requiring a modest amount of CPU-time—should be included for parallelisation as well. Obviously, parallelisation is most effective when implemented per

level. Each level is then decomposed into several subdomains which can have one or several boxes, whereas multiple processors can be assigned to these subdomains (Figure 7a).

The part of the algorithm dealing with the redistribution of the nodes can not be skipped in the parallelisation process because it still contributes significantly, depending on the evolving contours. Parallelisation of this part is however completely different, i.e. redistribution is done in a contour-wise fashion as is depicted in Figure 7b. This paper however will only focus on the more important box-wise parallelisation of the velocity computations, which is carried out independently from the node redistribution. In the HEM the boxes are ordered per level as shown in Figure 8. The coarsest level ($l = 1$) has four boxes, numbered $1, \ldots, 4$ in counter-clockwise manner starting with the left-lowest box. Level $l = 2$ has sixteen boxes, numbered $5, \ldots, 20$, level $l = 3$ has sixty-four boxes, numbered $21, \ldots, 84$, etc. all according to the counter-clockwise route.

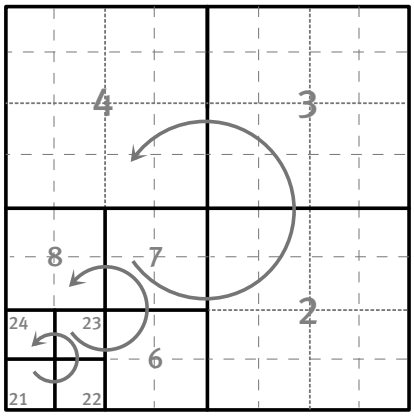

Figure 8 – *Composite schematic of the way the HEM is ordering its boxes at every level (three levels in this example). Note that only parts of the two finer levels have been drawn.*

The processors assigned to subdomains also follow the route according to Figure 8. The assignment of processors can be scheduled in various ways. Two scheduling policies in OpenMP used for the parallel HEM are static scheduling and dynamic scheduling. When each subdomain is treated as a static subdomain, a processor assigned to this subdomain stays put until all processors have finished their own local job. After that, they proceed in unison to their next assigned subdomain. Static scheduling can be very inefficient when the computational load is non-uniform. However, when an algorithm has a predefined uniform load, processors can be assigned to steady portions of the algorithm and minimize the communication overhead and static scheduling would be very efficient. Dynamic scheduling, on the other hand, implies that a processor can proceed with the next available subdomain when it has finished its own local job, even when other processors are still busy with their first computation. Dynamic scheduling suffers more from communication overhead due to the dynamic displacements of processors.

For an unevenly distributed computational load, as is the case in many contour dynamics simulations, processors taking care of less time-consuming jobs have to wait for the slower jobs and will run idle when static scheduling is applied. This so-called load imbalance is illustrated in Figure 9 for a computation with a non-uniform distribution of nodes in the domain. In Figure 9a a static scheduling is applied to a finest level $l_f = 3$ with four processors to assign. Choosing an appropriate subdomain size (or chunk size, a designation used in OpenMP) in the scheduling policy of OpenMP can be convenient for solving a specific load imbalance problem. The chunk size in Figure 9a is four, and the load is clearly out of balance.

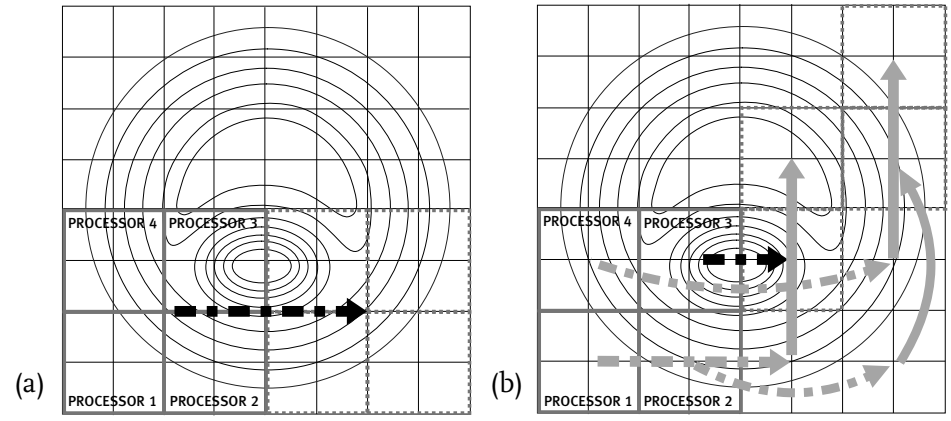

Figure 9 – (a) *Example of static scheduling for the HEM in OpenMP.* (b) *Dynamic scheduling for the same domain.*

Processors 1, 2, and 4 have to wait for number 3 before all can proceed in unison to the next subdomain (black arrow). Figure 9b shows how to solve for the load imbalance when using dynamic scheduling. When processor 1 has finished its computations it will start immediately computing the first available chunk (keeping the ordering in mind). When processor 3—the one with the highest load—has finished (black arrow), processors 1, 2, and 4 have already advanced to the right-top quadrant (gray arrows). The load imbalance for this configuration has been minimised. Dynamical scheduling will apparently be beneficial for parallelisation of the HEM when a smallest possible chunk size is chosen. The chunk size then is as small as one box, contrary to the above example with four boxes for a chunk. The smaller the chunk size, the more balanced the load is after finishing a certain level. Note that the example in Figure 9 is for illustrative use only.[3]

In order to study the overall parallel performance of the HEM, two important features of parallel programming have been analysed, viz. processor scalability and problem scalability. Processor scalability is given through the parameter S, the speed-up of the parallelisation, and is defined as the ratio between the work done by one processor and the total execution time of a parallel programme with P processors: $S = T_1/T_P$. Problem scalability has been studied for different values of l_f and N. The number of nodes N changes continuously during a simulation and because of this, l_f changes accordingly. When l_f is large, more boxes have to be computed and more communication is necessary between processors. In a shared-memory environment this can be a limiting factor for the parallel performance because communication timing is slow compared to the computations. The more computations a single processor can do in its chunk before communicating its results, the better the overall performance. This feature is called the computation-to-communication ratio.

One last important remark has to be made concerning the fraction of the algorithm that has not been parallelised. The part of the algorithm that works serially acts as a weakest link in the overall computation and slows down the method significantly. This can be clearly demonstrated by Amdahl's Law [3], which states that for an ideal parallel machine (neglecting

[3] In case of a domain with only a small number of boxes, it probably would be more efficient to start computing in boxes with the highest density of nodes.

communication overhead) the speed-up is

$$S_{Amdahl} = \frac{P}{Pf + (1 - f)} , \tag{13}$$

where f represents the serial fraction and P the number of processors. In Section 3 it was mentioned that the contribution of velocity and redistribution computations (which have been parallelised) differ for various simulations. For the numerical experiments in the next section this has been taken into account.

5 Numerical Experiments and Discussion

Three different numerical experiments have been performed. The goal of the first experiment is to investigate the scalability of the parallelised HEM, without adding additional nodes during the computation or applying any node redistribution. In the second experiment the role of static and dynamic scheduling on the scalability S is discussed. In this numerical experiment both the number of nodes increases during the simulation (and also l_f increases) and node redistribution is included. In a third numerical experiment the parallel efficiency of simulations with the HEM of the evolution of vortices in a non-uniform background vorticity field has been studied. The effort to obtain a parallelised version of the HEM is particularly aimed at solving this latter kind of problems.

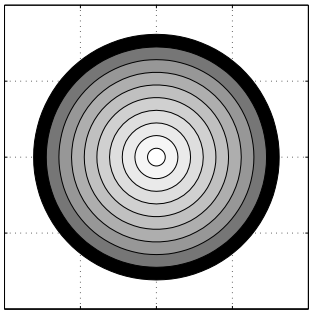

Figure 10 – *The first example shows ten nested concentric vorticity patches in an $l_f = 2$ domain.*

EXAMPLE 1 — A first example is the simulation of concentric patches of uniform vorticity (Figure 10). These patches will stay circular and constant in size. The redistribution of the nodes is therefore unnecessary. During a simulation N is kept constant and the contours rotate in a clockwise direction—due to an overall negative uniform vorticity. The load is distributed symmetrically and can be considered as reasonably uniform, except for the corners of the HEM domain. To gain insight in the processor scalability, l_f is also kept constant during each simulation. This implies that for a certain N the optimal l_f is not changed accordingly. The following values for P, N, and l_f have been chosen:

- The range of processors: $P = 1, 2, 4, 8, 16$
- The problem size: $N = 1000, 2000, 4000, 8000, 16000$
- The choice of finest level: $l_f = 2, 3, 4, 5$

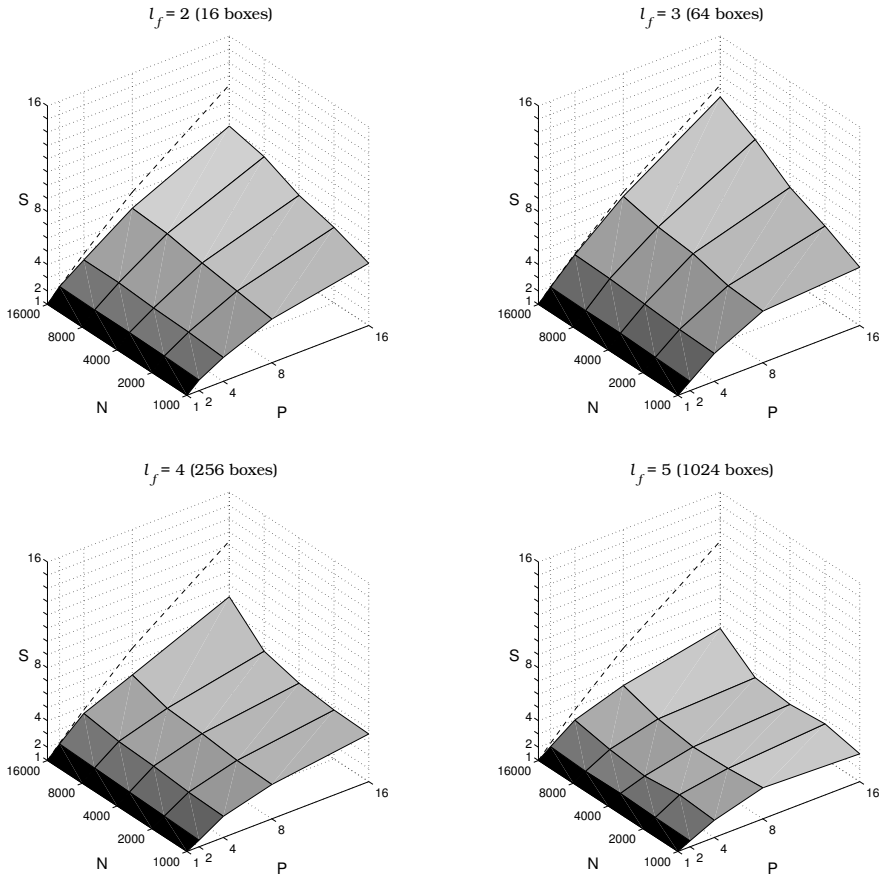

Figure 11 – $S = f(P, N)$ *for* $l_f = 2, \ldots, 5$. *The speed-up for* $N = 16{,}000$ *has a dashed line indicating Amdahl's limit for a serial fraction of* $f = 0.02$.

Each combination of P, N, and l_f represents a separate simulation. The results for the scalability S as function of P and N are given in Figure 11 as 3D graphs for different values of l_f.

According to Figure 11 it is clear that the parallel performance for a small number of processors is high for $l_f = \{2, 3, 4\}$. For $N = 16{,}000$, performance is maximal for $l_f = 3$ in the complete processor range $P = 1, \ldots, 16$. Level $l_f = 5$ seems to be inefficient for all choices of P and N. Furthermore, an increasing computation-to-communication ratio can be observed for increasing N in the figures for $l_f = \{2, 3, 4\}$, while for $l_f = 5$ this increment is only visible for $N \geq 8{,}000$. The amount of effective computations per communication step decreases for smaller boxes. This is supported by the speed-up data for $l_f = \{3, 4, 5\}$ which decrease for increasing l_f. Although the boxes are larger for $l_f = 2$, the speed-up cannot benefit from the computation-to-communication ratio size when P is relatively large due to the non-uniformity of the node distribution. The domain has a mere sixteen boxes for $l_f = 2$ and the processors finish rather quickly in the four corner boxes, while the four central boxes demand most of the CPU time. Some processors will inevitably run idle in the test with $l_f = 2$. This example clearly indicates the parallel efficiency as a function of P, N and l_f. For regular simulations however, a constant l_f is certainly not an option. The HEM would lose its $\mathcal{O}(N)$ behaviour. The adaptive tree-depth adjustments for $N = \{1000, \ldots, 16000\}$ would be $l_f = \{2, 3, 3, 4, 4\}$. For a parallel $\mathcal{O}(N)$ simulation, the overall speed-up, i.e. the final simulation wall-clock time,

is certainly optimal for $4{,}000 < N < 8{,}000$. For the other levels some parallel efficiency has to be sacrificed.

EXAMPLE 2 — This example will highlight the difference between static scheduling and dynamic scheduling. The simulation itself requires $\mathcal{O}(10^3)$ nodes and is a relatively 'cheap' computation compared to the kind of simulations the HEM is designed for. The simulation shows the formation of a tripole from an azimuthally perturbed isolated monopolar vortex (see Figure 12). The tripole rotates counter-clockwise according to the sign of the core vorticity.

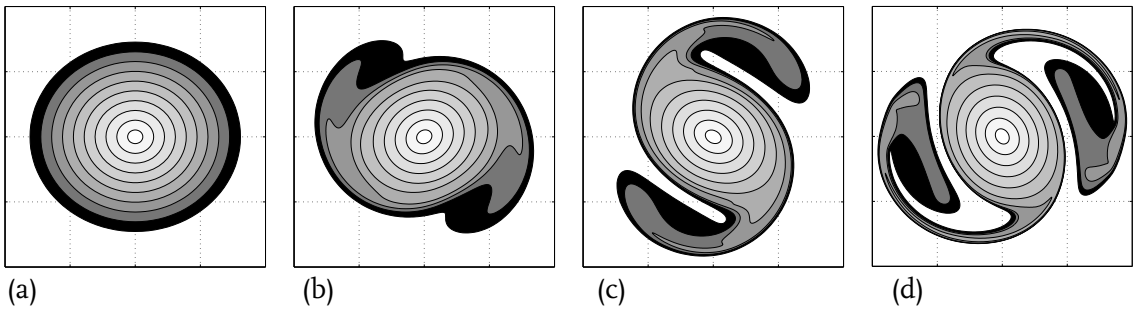

Figure 12 – *From a perturbed isolated monopole to a tripole.*

During the simulation the redistribution of the nodes is essential due to the formation of high-curvature segments in the contours (see, for example, Figures 12c and d). In the present test example the node redistribution algorithm is also parallelised. Additionally, we should keep in mind that the number of nodes N will increase. As a result the tree-depth (i.e. l_f) in the HEM will automatically increase. Figure 13a shows two curves which compare the static and the dynamic scheduling. The static scheduling policy is clearly spoiling the parallel performance. Dynamic scheduling delivers a speed-up curve like the one in Figure 11 for $l_f = 2$ and $l_f = 3$. This is conform the level updates for $500 < N < 5{,}000$. In order to obtain $O(N)$ operation count the finest level here is relatively coarse ($l_f \leq 3$), and some parallel efficiency is lost due to the fact that no optimal benefit of load-balancing can be obtained. As a result a few processors will run idle.

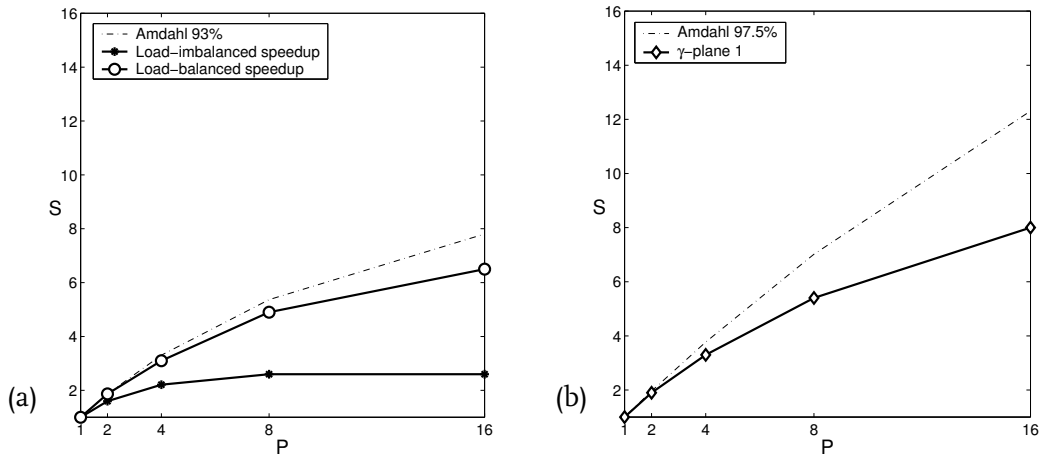

Figure 13 – (a) *Speed-up curves of Example 2. The serial fraction* $f = 0.07$. (b) *Speed-up curve for the* γ*-plane simulation of Example 3,* $f = 0.025$.

EXAMPLE 3 — The last example shows the parallel efficiency of simulations the HEM had been made for originally. It concerns the evolution of an isolated monopole—like the one in Example 1—embedded in a non-uniform background vorticity field. These so-called γ-plane simulations generate complex flow patterns and require at least $\mathcal{O}(10^4)$ nodes. Figure 14 shows four frame-shots of an isolated monopole that is being perturbed by the gradient background rotation. The perfect circular patches of this monopole transform into a tripolar structure while travelling to the centre (dotted cross) of the γ-plane. Note that only a part of the complete HEM domain is shown.

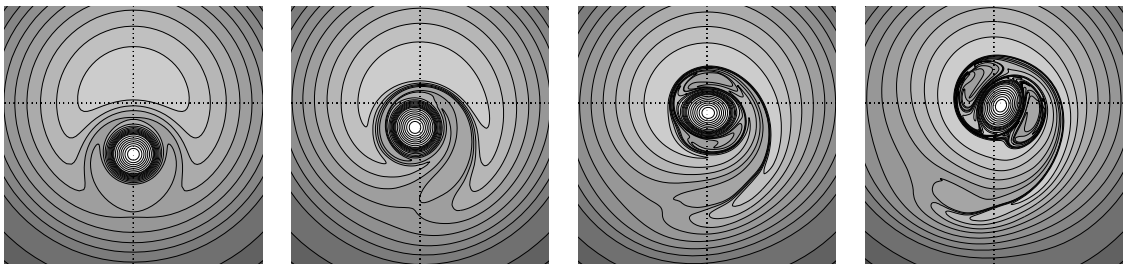

Figure 14 – *Thirty two contours ('gamma 2') show the monopole transforming into a tripole in a non-uniform background vorticity field.*

Two different configurations have been tested. The first simulation—denoted 'gamma 1'—has 21 contours for the background field and the monopole, and a node range of $10{,}000 < N < 30{,}000$. The second simulation ('gamma 2') has 32 contours and a node range of $20{,}000 < N < 45{,}000$. Both simulations start with only a tenth of this number but arrive in the mentioned ranges quite early in the simulation. These tests use considerably more nodes than the numerical simulations of Examples 1 and 2. Redistribution takes care of the node supply, particularly in the monopolar/tripolar region, indicating that the load is highly out of balance in the overall domain. This implies the use of dynamic scheduling with the smallest possible chunk size. Figure 13b shows the speed-up graph for the 'gamma 1' simulation. These speed-up values are higher, compared to Example 2, due to a smaller serial fraction. The efficiency however is getting worse. The reason for this behaviour lies in a larger l_f. The 'gamma 1' test needs more nodes and will adjust to a higher l_f, and thus creating smaller boxes with on the average still the same amount of nodes in a box. However, the computation-to-communication ratio actually decreases. Although a domain with a higher number of boxes can benefit more from the load-balancing effect of dynamic scheduling, more communication is necessary. The net effect is a lower efficiency.

Figure 15 shows three speed-up graphs for the 'gamma 2' simulation. In this test the tree-depth adjusts for a finest level of $l_f = 4$ during the complete computational time-frame. The graph in the middle ($l_f = 4$) shows a speed-up which is worse than both the 'gamma 1' and the tripole test (Figure 13). When l_f is forced to be constant at $l_f = 3$ or $l_f = 5$, the speed-up graphs on the left and right, respectively, are produced. All three are quite similar, albeit that the middle graph is slightly better. However, an overall better result compared to Figure 13 was expected. This is clearly not the case and is most likely due to the unevenly distributed computational load per box at the finest level which starts to worsen the parallel efficiency for this kind of computations. The structure for the 'gamma 2' tripole is very localised and complex, due to the generation of much more fine-scale filamentary structures, and demands much more CPU than the 'gamma 1' tripolar structure. For $l_f = 3$ this has immediate consequences if $P > 4$. Processors that start computing the few boxes with vast numbers of nodes

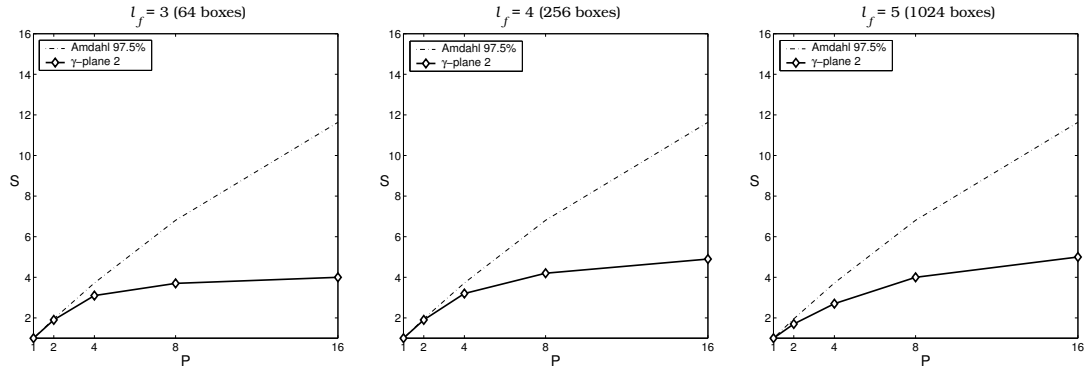

Figure 15 – *Speed-up graphs of the 'gamma 2' simulation for three finest levels. The serial fraction is* $f = 0.025$.

(the monopolar region) are busier than the other processors, which spend their time on the rest of the domain with a relatively small number of nodes. For $l_f = 4$ this is also the case. Figures 16a, 16b, and 16c show the complete HEM domain for the three levels. For $l_f = 5$, i.e. 1,024 boxes, unevenly distributed nodes should not be a severe problem anymore, but here the net effect on the speed-up is negative due to the large communication overhead.

A solution to the above discussed problem could be locally increasing or decreasing l_f such that the average number of nodes per box on the finest level is approximately constant throughout the domain (Figure 16d). The computation-to communication ratio can then be held constant, while the communication overhead is now the only restrictive factor. This adaptive chunk size refinement together with dynamic scheduling should minimise the load though, for all kinds of complicated flow phenomena. This approach is not implemented yet and should be pursued in future investigations.

6 Conclusive remarks

Despite the shared-addressing complication of the HEM—which is inherently involved in its algorithmic structure—the parallel HEM scales (very) good for small numbers of processors ($P \leq 8$) for every test-case discussed in this paper. The already mentioned computation-to-communication ratio shows more efficient speed-up results when the box size is held constant (Example 1). For regular HEM simulations we have however non-uniform node distributions and box size adaptations, implying a higher computation-to-communication ratio for certain boxes. The adaptations are necessary for keeping the favourable $\mathcal{O}(N)$ operation count for the HEM. For $P > 8$, the tripole test (Example 2) delivers a good efficiency. For the γ-plane simulations, however, the efficiency drops when l_f is higher and the local load is increasing (Example 3). The 'gamma 1' test still delivers a nice efficiency for small numbers of processors, but for the 'gamma 2' test which shows the appearance of many small-scale filamentary structures, the load was locally too high.

The crux here is that an increasing l_f demands an increasing amount of communication between (expensive) computations, whereas the dynamic scheduling solves for the non-uniform load better when l_f is actually high. A general solution for this 'trade-off' demands an improved parallelisation strategy: a locally defined tree-depth based on the requirement to keep the average number of nodes per finest level box approximately constant. Communication overhead, though, shall be a limiting factor when massive parallelisation, i.e. $P \sim \mathcal{O}(100 - 1000)$, is to be applied. Nonetheless, the current parallel HEM is an important

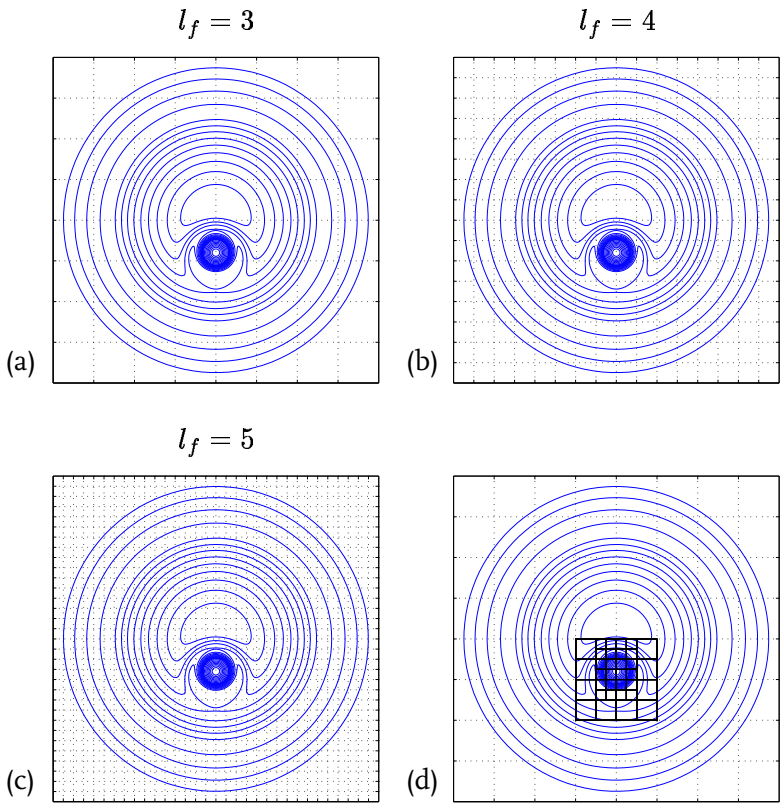

Figure 16 – *The HEM domain with all 32 contours of the 'gamma 2' simulation for the three levels in (a), (b), and (c). In (d) a possible solution for the most efficient load balancing.*

improvement to run γ-plane simulations in a much shorter time for small numbers of processors.

Acknowledgements

We would like to thank Dr. P.W.C. Vosbeek for providing the original HEM code, and for many valuable comments and suggestions.

References

[1] `http://www.openmp.org`.

[2] C.R. Anderson, *An implementation of the fast multipole method without multipoles*, SIAM J. Sci. Statist. Comput. **13** (1992), no. 4, 923–947.

[3] D.E. Culler and J.P. Singh, *Parallel Computer Architecture - A Hardware/Software Approach*, Morgan Kaufmann Publishers, 1999.

[4] D.G. Dritschel, *Contour dynamics and contour surgery: Numerical algorithms for extended, high-resolution modelling of vortex dynamics in two-dimensional, inviscid, incompressible flows*, Comput. Phys. Rep. **10** (1989), no. 3, 77–146.

[5] L. Greengard and V. Rohklin, *A fast algorithm for particle simulations*, J. Comput. Phys **73** (1987), 325–348.

[6] T. Hrycak and V. Rohklin, *An improved fast multipole algorithm for potential fields*, SIAM J. Sci. Comput. **19** (1998), no. 6, 1804–1826.

[7] R. Murty and D. Okunbor, *Efficient parallel algorithms for molecular dynamics simulations*, Parallel Computing **25** (1999), 217–230.

[8] P.W.C. Vosbeek, *Contour Dynamics and Applications to 2D Vortices*, Ph.D. thesis, Eindhoven University of Technology, 1998.

[9] P.W.C. Vosbeek, H.J.H. Clercx, and R.M.M. Mattheij, *Acceleration of contour dynamics simulations with a hierarchical-element method*, J. Comput. Phys. **161** (2000), 287–311.

[10] P.W.C. Vosbeek, H.J.H. Clercx, G.J.F. van Heijst, and R.M.M. Mattheij, *Contour dynamics with non-uniform background vorticity*, Int. J. Comput. Fluid Dyn. **15** (2001), 227–249.

[11] P.W.C. Vosbeek and R.M.M. Mattheij, *Contour dynamics with symplectic time integration*, J. Comput. Phys. **133** (1997), 222–234.

[12] N.J. Zabusky, M.H. Hughes, and V.K. Roberts, *Contour dynamics for the Euler equations in two dimensions*, J. Comput. Phys. **30** (1979), 96–106.