HPC simulations of brownout: a non-interacting particles dynamic model

Journal Title
XX(X):1–19

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Roberto Porcù[1,3], Edie Miglio[1], Nicola Parolini[1], Mattia Penati[1], and Noemi Vergopolan[2]

Abstract

Helicopters can experience brownout when flying close to a dusty surface. The uplifting of dust in the air can remarkably restrict the pilot's visibility area. Consequently, a brownout can disorient the pilot and lead to the helicopter collision against the ground. Given its risks, brownout has become a high-priority problem for civil and military operations (Sabbagh 2006). Proper helicopter design is thus critical, as it has a strong influence over the shape and density of the cloud of dust that forms when brownout occurs. A way forward to improve aircraft design against brownout is the use of particle simulations. For simulations to be accurate and comparable to the real phenomenon, billions of particles are required. However, using a large number of particles, serial simulations can be slow and too computationally expensive to be performed. In this work, we investigate an MPI + multi-GPU approach to simulate brownout. In specific, we use a semi-implicit Euler method to consider the particle dynamics in a Lagrangian way, and we adopt a precomputed aerodynamic field. Here, we do not include particle-particle collisions in the model; this allows for independent trajectories and effective model parallelization. To support our methodology, we provide a speedup analysis of the parallelization concerning the serial and pure-MPI simulations. The results show (i) very high speedups of the MPI + multi-GPU implementation with respect to the serial and pure-MPI ones, (ii) excellent weak and strong scalability properties of the implemented time integration algorithm, and (iii) the possibility to run realistic simulations of brownout with billions of particles at a relatively small computational cost. This work paves the way towards more realistic brownout simulations, and it highlights the potential of HPC for aiding and advancing aircraft design for brownout mitigation.

Keywords

helicopter, brownout, particles dynamics, MPI, CUDA, GPU, Tesla P100

Introduction

In helicopter aviation, the term *brownout* refers to the phenomenon generated by the rotor downwash during landing or take-off on dusty or sandy soils. Brownout restricts the pilot's

[1]MOX-Department of Mathematics, Polytechnic University of Milan, Italy
[2]CEE-Department of Civil and Environmental Engineering, Princeton University, Princeton, NJ
[3]CCSE-Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA

Corresponding author:
Roberto Porcù, CCSE-Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA

Email: roberto.porcu@polimi.it

visibility due to the uplifting of a cloud of dust particles in the air, as exemplified by Figure 1.

(a) Eurocopter EC135 D-HZSG, Brüggemann (2012)

(b) Bell-Boeing V-22 Osprey, Williams (2010)

Figure 1. Helicopters experiencing brownout.

When brownout occurs, the pilot may experience disorientation and loss of control. This is a dangerous situation which may lead the helicopter to crash to the ground, e.g. see Vyrnwy-Jones (1988), Durnford et al. (1995), and Braithwaite et al. (1997). Other consequences of brownout are rotor blades abrasion, damages to the components of the rotor, and loss of power due to air filter occlusion. Single-rotor helicopters are generally severely affected by this problem but, as pointed out in D'Andrea (2010), for dual-rotor helicopters (as tandems and tiltrotors) the situation can be even worse.

Different factors may influence the occurrence and the intensity of brownout: rotor blades rotation speed, rotor configuration, soil moisture, soil particle sizes, land surface texture, weather conditions, and aircraft landing angle. Landing site preparation, specific piloting techniques, and helicopter aerodynamic design are effective remedies to prevent or limit the occurrence of this phenomenon (e.g., see Syal et al. (2011) and Phillips et al. (2010)). However, site preparation preceding the landing is only possible at specific sites, as base camps. Otherwise, specific piloting techniques and aircraft aerodynamic design are then essential in cases of landing at unexpected locations and during severe weather conditions. The design of the rotor blades has a remarkable impact on the shape and intensity of the rotor downwash (e.g., see Ghosh et al. (2010) and Phillips et al. (2010)). Therefore, brownout simulations are critical to (i) provide insights to wisely improve the design of the fuselage and rotor of a helicopter, and to (ii) be used for pilots training purposes when embedded in rotorcraft flight simulators, see for example Wachspress et al. (2008), Duda et al. (2013) and Viertler and Hajek (2015). A further remedy to overcome visibility restriction during brownout is represented by the 3D-LZ Helicopter LADAR Imaging System (e.g. see Schuetz et al. (2010), Szoboszlay et al. (2009) and Savage et al. (2010)). It consists of a virtual image reconstruction solution that employs those waves of the spectrum whose wavelength is of the order of the millimeter. These waves can penetrate typical visual obscurants. Overall, this approach is not resolving problems of rotor abrasion and air filter occlusion, which significantly increase maintenance costs.

During the past few years, industries and academies have been spending a lot of effort to expand our knowledge on brownout and fill up the critical lack of experimental data. For example, Nathan and Green (2008, 2009), and Doehler and Peinecke (2010) provide quantitative and qualitative results of a series of brownout-like experiments to help the validation of numerical simulations. In Wong and

Tanner (2010), Tanner (2011), and Syal and Leishman (2011) the authors present measurements of the brownout cloud and flow data for an EH-60L Black Hawk, obtained through the photogrammetry technique. Wadcock et al. (2008) analyze the characteristics of the rotor wash for a UH-60 Blackhawk and an EH-101 Merlin presenting full-scale tests to be used as validation for CFD simulations. The work by Keller et al. (2006) presents a physics-based aerodynamic analysis and prediction of the rotor wash flow field near the ground. The results provided by these authors can be used to drive debris particles and reproduce brownout in flight simulators.

Earlier numerical simulations of brownout can be found in Wachspress et al. (2008), D'Andrea (2009, 2010, 2011) and Gerlach (2011). In the works by Phillips and Brown (2008, 2009) and Phillips et al. (2010, 2011), the authors demonstrate that the shape of the cloud of dust is mainly affected by the entire helicopter and not only by the rotor configuration. In Tritschler et al. (2014) the authors describe how flight path optimization can mitigate rotorcraft brownout. Phillips and Brown (2009) and Syal and Leishman (2013) include the modeling of dust particles saltation from the ground into rotorcraft brownout problems. The uplift of dust particles lying on the ground is due to the bombardment by previously suspended particles in the flow. The authors show how bombardment can play a critical role in the rapid development of intense dust clouds.

In Hu et al. (2011), Govindarajan (2011) and Govindarajan et al. (2013), the authors discuss different particle-clustering algorithms, such as the fast multipole method, the Gaussian method, the k-means, and the Osiptsov's method, which can reduce the computational cost per time step in brownout simulations. Although these algorithms allow a significant reduction of the time required by brownout simulations for particular conditions, they are problem dependent and have limited applicability. In terms of particle-particle interactions, Syal (2012) and Thomas (2013) found that, during a brownout, the motion of particles does not influence the aerodynamic field if the particle concentration is not high enough.

In this paper, we introduce an algorithm that uses MPI + multi-GPU to simulate particle dynamics in the context of brownout simulations for HPC. In specific, the algorithm uses a semi-implicit Euler scheme to consider particle dynamics in a Lagrangian way. In this study, we account for different amounts, sizes, and densities of the particles. We test the simulations using (i) an analytical and (ii) a precomputed aerodynamic field. For the analytical field, we compute a high-order solution employing the Richardson extrapolation. The precomputed field is an input data provided by previous CFD simulations. In our work, similar to what is proposed by Syal (2012) and Thomas (2013), particle-particle collisions are not included. Also, a benchmark comparison of the simulation performance concerning pure-MPI implementation is presented. The numerical results of this study focus on (i) the speedup obtained through HPC techniques, and on (ii) the scalability with the increasing number of particles. The overarching objective of this study is to develop a robust and efficient parallel computation of brownout that we will use as the basis for subsequent model development. Although preliminary, the initial results show the potential of HPC for accurate and efficient particle tracking simulations applications.

The paper is organized as follows: in Section "Mathematical and numerical model", we describe the equations of motion of the system of particles and the time-integration algorithm we adopt for the simulations of the brownout phenomenon. In Section "HPC implementation", we provide a brief digression on the parallelization of the numerical method we employ. Finally, in Section "Numerical results", we present the outcomes and speedup comparisons for the brownout simulations using the analytical and precomputed aerodynamic fields, as well as a performance comparison against pure-MPI implementation.

Mathematical and numerical model

In this work, the brownout phenomenon is simulated in a Lagrangian way, following the motion of the particles. The adopted time-integration scheme is the semi-implicit Euler method, which is a first-order symplectic integrator capable of preserving the total energy, then reducing the numerical dissipation. For higher-order methods that can be applied to particle mechanics and brownout problems, e.g., Störmer-Verlet or spectral variational integrators, we refer to Porcù (2013) and Miglio et al. (2018).

The dynamics of each particle p are governed by the second Newton's law $\mathbf{F}_p = m_p \mathbf{a}_p$, where $\mathbf{a}_p$ is the acceleration and $\mathbf{F}_p$ is the forcing term, defined as (e.g. see Jasion (2013)):

$$\mathbf{F}_p = \mathbf{F}_p^D + \mathbf{F}_p^G + \mathbf{F}_p^S + \mathbf{F}_p^M + \mathbf{F}_p^B, \qquad (1)$$

where

$$\mathbf{F}_p^D = \frac{1}{2}\rho_{air} \left|\mathbf{v}_{\mathrm{rel},p}\right| \mathbf{v}_{\mathrm{rel},p} \frac{\pi d_p^2}{4} C_{\mathrm{d},p} \qquad (2)$$

is the Stokes drag force (e.g. see Hoerner (1965)) and $\mathbf{F}_p^G = m_p \mathbf{g}$ is the gravity force. The additional terms $\mathbf{F}_p^S$, $\mathbf{F}_p^M$ and $\mathbf{F}_p^B$ are the Saffman, Magnus and Basset forces respectively.

In equation (2) the quantity

$$\mathbf{v}_{\mathrm{rel},p} = \mathbf{v}_p - \mathbf{v}_{air}(\mathbf{x}_p) \qquad (3)$$

represents the relative velocity of the particle with respect to the air at the position $\mathbf{x}_p$; $\mathbf{g}$ is the gravitational acceleration; m_p and d_p denote respectively the mass and the diameter of the particle; ρ_{air} is the density of the air, assumed uniform. The drag coefficient $C_{\mathrm{d},p}$ is inversely proportional to the Reynolds number Re_p of the particle, as follows (e.g. see Johson (2016)):

$$C_{\mathrm{d},p} = \frac{24}{Re_p}, \quad Re_p = \frac{\rho_{air} \left|\mathbf{v}_{\mathrm{rel},p}\right| d_p}{\mu_{air}}, \qquad (4)$$

where μ_{air} denotes the dynamic viscosity of the air, assumed uniform.

The role of the additional terms $\mathbf{F}_p^S$, $\mathbf{F}_p^M$ and $\mathbf{F}_p^B$ is discussed, for example, in Jasion (2013) and Johson (2016). In particular, the Saffman force acts on particles moving in shear flows, and it is negligible when the Reynolds number of the particles is small. Also, the contribution of Magnus force, which is a lift force related to rotating particles in fluids, can be neglected for particles of small sizes. Finally, the Basset force depends on the relative acceleration of the particles with respect to the surrounding fluid. It may be neglected when the fluid/particle density ratio is small.

On the basis of these considerations the forcing term (1) can be simplified as:

$$\mathbf{F}_p = 3\pi d_p \mu_{air} \mathbf{v}_{\mathrm{rel},p} + m_p \mathbf{g}, \qquad (5)$$

where the additional terms $\mathbf{F}_p^S$, $\mathbf{F}_p^M$, and $\mathbf{F}_p^B$ have been neglected according to the consideration previously done. This approach

is consistent with the brownout simulation literature (see Syal (2012) and Thomas (2013)).

According to the semi-implicit Euler scheme, at each time step the new position $\mathbf{x}_p^{k+1}$ and velocity $\mathbf{v}_p^{k+1}$ are computed as follows:

$$\begin{cases} \mathbf{x}_p^{k+1} = \mathbf{x}_p^k + \Delta t \cdot \mathbf{v}_p^{k+1}, \\ \mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \Delta t \cdot \mathbf{a}_p(\mathbf{x}_p^k, \mathbf{v}_p^k), \end{cases} \tag{6}$$

where, according to (5), it is assumed:

$$\mathbf{a}_p(\mathbf{x}_p^k, \mathbf{v}_p^k) = 18 \frac{\mu_{air}}{\rho_p d_p^2} \left(\mathbf{v}_p^k - \mathbf{v}_{air}(\mathbf{x}_p^k) \right) + \mathbf{g}, \tag{7}$$

and $\Delta t = t^{k+1} - t^k$ is the timestep, assumed constant. Since we consider the particles to have a spherical shape, the mass relation $m_p = \frac{1}{6}\pi d_p^3 \rho_p$ is used in (7). The numerical scheme (6) is semi-implicit, then the fulfillment of the stability condition is required.

The mathematical model and the algorithm adopted for this work have specific characteristics which provide the following advantages:

- the Lagrangian tracking of every single particle allows obtaining very accurate solutions;
- the small number of unknowns and variables allows simulating large numbers of particles, improving the reliability of the results;
- the algorithm has excellent scalability properties. This ensures lower computational cost when more hardware resources are available.

Besides, the small timestep size adopted in the semi-implicit Euler scheme can be highly relevant for further model development, such as the inclusion of particle-particle collisions.

HPC implementation

GPUs (Graphic Processor Units) are highly parallel, multi-threaded, many-core processors with huge computational capabilities and high memory bandwidth. We decide to exploit their potential for brownout simulations because they are designed to be extremely efficient for SIMD (single instruction multiple data) problems. In this work, as we do not consider particle-particle collisions, the trajectories of the particles are all independent of each other. For this reason, it is convenient to subdivide the number of particles uniformly among the MPI tasks. Each task then equally distributes the computational load. For instance, the number of particles is evenly distributed among the available GPUs, and then almost identical operations are repeated on each particle at every timestep.

Within this setup, GPU routines – called *kernels* – are invoked by the *host* (a thread on the CPU) and run on the *device* (the GPU). After a kernel has been invoked, the CUDA (Compute Unified Device Architecture) runtime system creates a *grid* of many parallel threads residing on the device. Each thread executes the entire kernel.

During the execution, CUDA threads can access different memory spaces. In general, GPU-parallel applications mostly access the following GPU memory spaces:

- **registers**: small and fast-access local memory which is private to each thread;
- **shared memory**: small and fast-access local memory, shared among all the threads of the same block. It has the same lifetime as the block;
- **global memory**: large slow-access global memory which is visible to all the threads

and is persistent across kernel launches by the same application.

GPUs are designed to have an array of Streaming Multiprocessors (SM), which can be assigned several *blocks* of threads. The thread blocks run independently from each other. Each block can be assigned to any of the available multiprocessors in any order, concurrently or sequentially. Hence, thread blocks can be executed in different ways across any number of cores (automatic scalability).

In this work, we analyze both the strong and weak scalability properties of our parallelization. For the strong scalability analysis, the problem size is kept fixed, and the number of parallel processors is varied to study the change in the solution time. In the weak scalability analysis, the computational load per processor is kept constant as the number of processors varies to determine the difference in the solution time.

For further details on GPU and CUDA programming we refer to Kirk and Wen-mei (2012), Sanders and Kandrot (2010), Ruetsch and Fatica (2013) and to manuals NVIDIA (b), PGI (2019).

Code organization and data structures

The flowchart of the proposed method is illustrated in Figure 2. For realistic simulations of brownout, we employ a precomputed aerodynamic field, which is representative of the rotor downwash. This data contains information about the vectorial velocity of the air around the helicopter at several, equispaced, time instants. It is computed on a structured grid, equispaced on each direction and stored into files, one for each time instant. In our realistic brownout simulations, this precomputed aerodynamic data is used as a

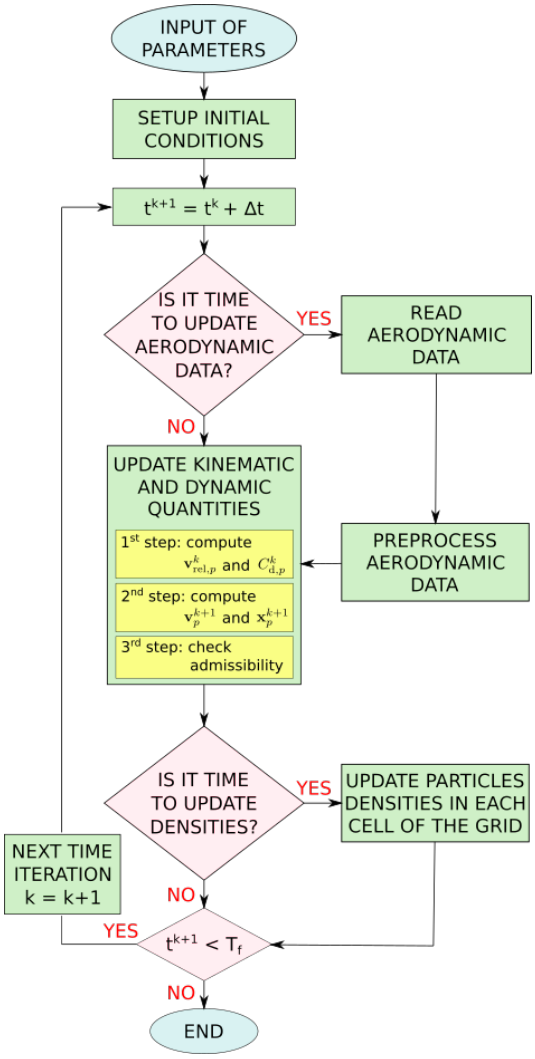


Figure 2. Flowchart of the program to simulate the brownout phenomenon.

given input known at fixed time instants and evenly staggered points in space. These points constitute the vertices of the aerodynamic computational grid.

To optimize efficiency and memory usage, we keep only positions, velocities, and densities of the particles stored in the GPU global memory. Also, we store the "old" and "new" space-averaged aerodynamic field data. At each time iteration, each thread can compute and temporarily store all the other needed kinematic and dynamic variables on its registers. Because of the mutual independence of the trajectories of the particles, no communication is necessary during the kinematic update, so each GPU works only on its variables. The adopted data

layout is a standard struct of arrays. We allocate one struct for each GPU, and each struct contains six monodimensional arrays having the size of the number of particles managed by the corresponding GPU. Three arrays are for the particles coordinates $\mathbf{x}_p = (x_{x,p}, x_{y,p}, x_{z,p})$ and three for the velocities $\mathbf{v}_p = (v_{x,p}, v_{y,p}, v_{z,p})$. Each struct also contains seven tridimensional arrays. One array is used to compute and store the spatial densities of particles in the computational grid, and it has the dimensions of the entire grid. The remaining six arrays contain the three components of the space-averaged air velocity $\mathbf{v}_{air} = (v_{x,air}, v_{y,air}, v_{z,air})$ defined on the whole aerodynamic grid, read from input files. Three of them correspond to the "old" time instant t^k, and the remaining to the "new" time instant t^{k+1}. The data layout for the positions, velocities, and densities of the particles is standard. For what concerns the space-averaged aerodynamics field, we do not divide the workload among the MPI tasks or GPUs, but we require each GPU to compute and store the whole aerodynamic data. The advantage of this approach is that it allows us to avoid host-devices and task-task communication, and it exploits the excellent performance of a GPU when a SIMD computation is required. Finally, we would like to point out that we adopt double-precision floating-point numbers for all the real-valued variables, both on the CPU and GPU.

Initial conditions

The computational domain is a rectangular cuboid with dimensions

$$[-L, L] \times [-L, L] \times [0, L], \tag{8}$$

where L is chosen in order to make it as big as the aerodynamic field domain.

At initial time $t = 0$, particles can be uniformly or randomly distributed inside a horizontal and narrow box of dimensions

$$[-L, L] \times [-L, L] \times [10\,\text{mm}, 25\,\text{mm}].$$

This setting is adopted since our mathematical model does not consider saltation of particles from the ground. Then we initialize particles positions as they are already suspended in the air.

During the simulation, when a particle goes out of the computational domain, its kinematic quantities (*i.e.* positions and velocities) are reset to the corresponding initial conditions.

Aerodynamic data

The aerodynamic field interacts with the particles through the drag force $\mathbf{F}_p^D$, defined in (2). In this study, we use (i) an analytical and (ii) a precomputed and more realistic aerodynamic field.

For the analytical approach we consider a solenoidal field defined in cylindrical coordinates, as follows:

$$\mathbf{v}_{air} = \begin{bmatrix} v_{\rho,air} \\ v_{\theta,air} \\ v_{z,air} \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial \psi(\rho, z)}{\partial z} \\ 0 \\ \dfrac{\partial \psi(\rho, z)}{\partial \rho} \end{bmatrix}, \tag{9}$$

where

$$\psi(\rho, z) = A \sin\left(\frac{\pi \rho}{L}\right) \sin\left(\frac{\pi z}{L}\right)$$

is a potential field with multiplicative factor A and characteristic length L. Since v_θ is always zero and the other two components of the velocity don't depend on the angular coordinate θ, the above aerodynamic field has a cylindrical symmetry along the z direction and it can be naturally assumed as bidimensional.

The precomputed and more realistic aerodynamic field was determined before the brownout simulation. The data relative to this field is stored into files, each corresponding to a fixed time instant multiple of the timestep adopted in the problem. When the time reaches one of these instants, the corresponding aerodynamic data is read. The air velocity is evaluated at the vertices of the aerodynamic grid, which is common to all the files.

To define the pointwise value of the air velocity $\mathbf{v}_{air}$ inside each cell, we preprocess the aerodynamic information immediately after reading the data file. In particular, each GPU parallelly computes the component-wise average of the air velocities at the eight vertices of each cell of the whole aerodynamic domain.

During each time interval $I^n := [t^n, t^{n+1}]$, we keep on the GPU global memory both the old (t^n) and new (t^{n+1}) space-averaged air velocities. Given the current time $t^k \in I^n$, we compute the relative velocity (3) at time t^k through a linear interpolation over the time interval:

$$\mathbf{v}^k_{\mathrm{rel},p} = \mathbf{v}^k_p - \mathbf{v}^k_{air}(\mathbf{x}^k_p), \tag{10}$$

where

$$\mathbf{v}^k_{air}(\mathbf{x}^k_p) = \omega^n \mathbf{v}^n_{air}(\mathbf{x}^k_p) + \omega^{n+1}\mathbf{v}^{n+1}_{air}(\mathbf{x}^k_p),$$

and

$$\omega^n = \frac{t^{n+1} - t^k}{t^{n+1} - t^n}, \quad \omega^{n+1} = \frac{t^k - t^n}{t^{n+1} - t^n}.$$

Time integration

At each timestep, the update of the kinematic and dynamic quantities is organized as follows, as illustrated in Figure 2:

- **1st step** : computation of the "old" (*i.e.* at t^k) relative velocities and drag coefficients as defined in (10) and (4);
- **2nd step** : computation of the "new" (*i.e.* at t^{k+1}) velocities and positions according to (6);
- **3rd step** : check of the admissibility of the updated positions and reset to the initial conditions for those particles whose new coordinates exceed the domain boundaries.

The update of the relative velocities of the particles depends on the information of the averaged air velocity at each particle position. Due to the turbulent nature of the aerodynamic field, particles having consecutive indices in the data structure are likely to have coordinates that are far away from each other in space. Hence, there is a high chance for consecutive threads to access the aerodynamic data array at not consecutive but spread addresses, resulting in GPU global memory uncoalesced accesses problems. This issue can't be solved by loading the aerodynamic data into thread-local memory spaces, like registers or shared memory, because it is needed in its whole, and it exceeds the available resources. Neither the texture memory can be beneficial to solve this problem because texture memory is helpful only when the accesses are clustered. We refer to NVIDIA (b) for further details about these topics.

Nevertheless, to investigate potential input/output implication in our implementation efficiency, we profiled the simulations using both the analytical and the realistic test cases. The operation-wise results are synthesized in Table 1. We observe an averaged timing of 443.01 µs for the analytical field simulation, while 500.88 µs for the realistic field one. The

Table 1. Averaged computational times by operation using 1 GPU to simulate 6.144×10^6 particles.

kernel ID	operation
1	set initial conditions of the particles
2	compute densities of the particles
3	time integration algorithm
4	compute space-averaged air velocities

Simulations with analytical field				
kernel ID	registers nb.	shared mem	avg. time	total %
1	24	0 B	91.402 µs	0%
2	22	0 B	237.90 µs	0.18%
3	72	0 B	443.01 µs	95.94%
Copy HtoD: 13.733MB, 3.32 GB/s			7.54 ms	3.86%

Simulations with realistic field				
kernel ID	registers nb.	shared mem	avg. time	total %
1	24	0 B	91.23 µs	0%
2	22	0 B	141.09 µs	0.12%
3	78	0 B	500.88 µs	96.71%
4	32	0 B	65.89 µs	0.03%
Copy HtoD: 13.733MB, 3.2 GB/s			4.072 ms	2.39%
Copy DtoH: 11.908MB, 11.16 GB/s			153 ms	0.7%

timings show that there is not a significant difference in the performance of the time integration algorithm using the analytical or precomputed fields. We consider this is related to the improvements provided by the most recent GPU hardware architectures.

Densities of particles

The spatial distribution of particles and their densities throughout the computational domain provides us with important information about the visibility conditions around the helicopter. This information can be used, for instance, for image rendering purposes when brownout restricted visibility condition is represented in rotorcraft flight simulators (e.g., see Wachspress et al. (2008)).

We consider the same grid adopted for the aerodynamic field data. On this grid, when the simulation time has reached an integer multiple of a custom time interval (that we set equal to 5×10^{-2} s), we update the densities of the particles. For each cell of this grid, each GPU computes the density over the cell volume of the particles it is managing. At the end of the kernel, we synchronize host and device, and we copy the computed densities on the host memory. Finally, an MPI sum reduction is performed by the master task when each task has gathered all the densities calculated by the GPUs it is managing. Each particle atomically contributes to the total density of its corresponding cell. To perform this computation in parallel on the GPU, we employ the API `AtomicAdd`, which allows achieving mutual exclusion mechanisms that prevent race conditions. Atomic functions work both on global and shared memory. In the latter case, the performance is higher since accesses must be mutually exclusive only for threads of the same block.

Numerical results

In this section, the results obtained with two different aerodynamic fields are analyzed. We run the MPI + multi-GPU parallel simulations on a Dell Linux Cluster made of 80 nodes equipped with 2.4GHz Xeon Broadwell E5-2680 v4 CPUs. Each node has 28 cores, 720 GB RAM, Omnipath interconnection. Each node is also equipped with 4 NVIDIA Tesla 1328 MHz P100, interconnected through Omnipath with a dedicated 16 GB RAM per GPU. We employ the CUDA 10.1 library extensions for Fortran 90 provided by the PGI compiler v19.5. We run the serial and MPI-parallel simulations on an HPE Linux Cluster made of 408 nodes interconnected through Omnipath and equipped with 2.4 GHz Skylake processors. Each node has 40 cores and 192 GB RAM.

In the first case, we consider an analytical solenoidal flow field, as previously described in section "Aerodynamic data". This does not give significant results from the point of view of the brownout phenomenon, but it is

still useful to investigate the performance of the parallelization without overheads due to readings of the files of the aerodynamic field.

In the second case, a realistic flow field obtained from a previous CFD simulation consistent with the brownout problem is considered. In this case, the interest focuses not only on the performance gain achieved through the parallelization but also on the accuracy of the numerical solution.

In both cases we assume that the total number of particles is equally subdivided into three groups of different diameters (d_p):

- "Small" particles with $d_p = 5\,\mu\text{m}$;
- "Medium" particles with $d_p = 10\,\mu\text{m}$;
- "Large"particles with $d_p = 15\,\mu\text{m}$.

We assume the dynamic viscosity of the air μ_{air} uniform and equal to $1.78 \times 10^{-5}\,\text{Pa}\,\text{s}$. Consequently, the timestep Δt must be less than or equal to $2 \times 10^{-4}\,\text{s}$ to guarantee numerical stability of the semi-implicit Euler time-integration scheme adopted (6). For each simulation, we consider a computational domain of the type described by (8) with a characteristic length L equal to 7.5 m, and we set the final time instant equal to 50 s.

To evaluate the efficiency and performance of the proposed algorithm, we perform a speedup analysis in relative and absolute terms. Let us consider a number $n = 2^p$, where p is an integer greater or equal than 1. We compute the "relative" speedup as the ratio of the computational cost between simulations with $n/2$ and n GPUs or MPI parallel tasks. We define the speedup to be "absolute" when it represents the ratio of the computational cost between simulations with 1 and n GPUs or MPI parallel tasks. The latter can be obtained by multiplying all the preceding relative speedups. In the following subsections, we summarize this analysis in several tables, where we report both the relative and absolute speedups. In this way, the reader has an immediate and exhaustive understanding of the performance of the parallelization when different configurations are adopted.

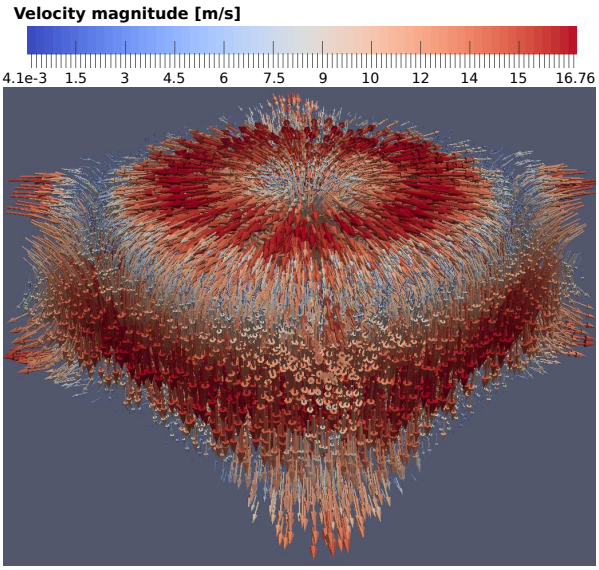
In the subsequent, we discuss the simulations and performance results for both the analytical and realistic aerodynamic fields. The timings presented in the corresponding tables and figures refer to the elapsed time spent for the actual computation, MPI communication, and input of the aerodynamic data (the latter only for simulations with the realistic aerodynamic field). An analysis of the computational cost with profiling tools shows that the time spent in input and MPI operations is marginal.

Analytical aerodynamic field

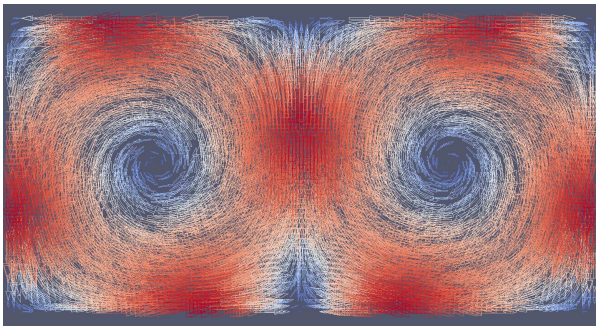
The solenoidal analytical field, previously defined by (9), is illustrated in Figure 3. In the tests using the analytical field we set the multiplicative factor A equal to $40\,\text{m}^2\,\text{s}^{-1}$ and the characteristic length L equal to 7.5 m, which is coincident with the size of the computational domain. In specific, Figure 3 illustrates respectively a tridimensional and bidimensional vectorial plots of the aerodynamic flow velocity field.

Based on the total number of particles, we compare the performance of the simulations using five different configurations N_i, where N_i is equal to $2^{i-1} \times\ 6.144 \times 10^6$ with $i \in \{1, \ldots, 5\}$. Figure 4 illustrates the positions of the particles in the case of the first configuration N_1 at time 12.5 s.

One run of the serial program with configuration N_1 requires about eight days and sixteen hours to be completed. The computational times for the GPU-parallel simulations with one, two, four, eight, and



(a) Three-dimensional top view.



(b) Slice along a vertical plane passing through the origin.

Figure 3. Illustrations of the analytical air velocity.

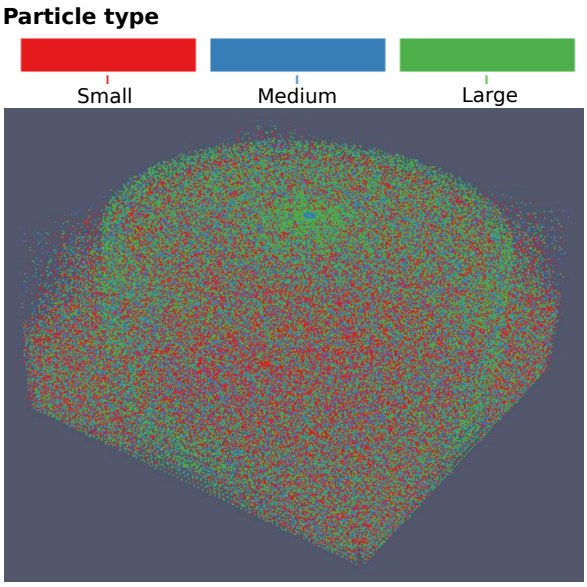


Figure 4. Positions of the particles for a simulation with 6.144×10^6 particles and the analytical aerodynamic field, at time $t = 12.5\,\mathrm{s}$.

sixteen GPUs are reported in Table 2. Comparing the serial and the parallel runs in the case of the first configuration N_1, we observe that the smallest speedup provided by the parallelization (corresponding to the simulation with one GPU) is about 2200X.

For simulations with two, four, eight, and sixteen GPUs, the speedup increases linearly. This demonstrates the excellent weak and strong scalability properties of the proposed implementation. The simulation timings in Table 2 clearly show that for any configuration N_i the speedup obtained doubling the number of GPUs is always close to the theoretical limit of 2X. This result proves the good strong scalability properties of the GPU parallelization, as illustrated in Figure 5. Observing Figure 6, we can also deduce good weak scalability properties. Indeed, the execution time does not change significantly when the number of particles per GPU is kept constant.

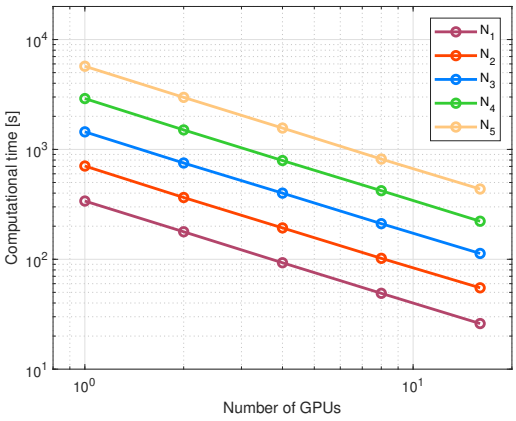


Figure 5. Strong scalability analysis for the simulations with the analytical aerodynamic field.

Realistic aerodynamic field

We run the program adopting a realistic aerodynamic field that was previously computed through CFD simulations. Our goal is to test the parallelization performance on a real brownout-like problem and to investigate the accuracy of the implemented numerical method. Similar to the analytical aerodynamic field case,

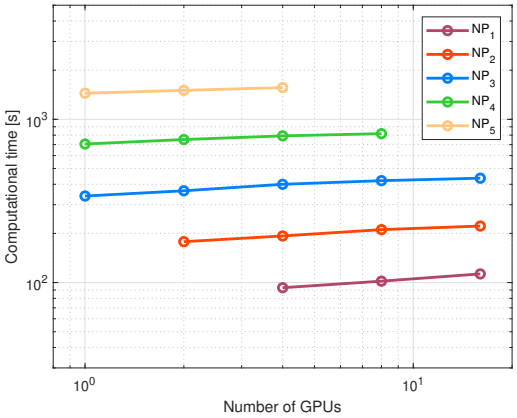


Figure 6. Weak scalability analysis for the simulations with the analytical aerodynamic field. For each line in the plot, the number of particles per GPU is kept constant and is equal to $\mathrm{NP}_i = 2^{i-1} \times 6.144 \times 10^6$ with $i \in \{1, \ldots, 5\}$.

we consider five different configurations with a total number of particles N_i respectively equal to $2^{i-1} \times 6.144 \times 10^6$ with $i \in \{1, \ldots, 5\}$.

We firstly run the code in serial (one MPI-task) and in parallel with pure MPI, successively increasing the number of MPI-tasks. In these simulations, we consider the first configuration N_1 and we set the number of particles equal to 6.144×10^6. The results are reported in Table 3. Figure 7 illustrates the positions of the particles at three different time instants. We observe good properties of strong scalability, as illustrated in Figure 8.

We run the GPU-parallel simulations to assess the performance improvement that can be gained through the proposed implementation. The timings relative to these simulations are reported in Table 4. Observing Figure 9, we can confirm that, even with a realistic aerodynamic field, the excellent strong scalability properties are preserved. The same conclusions hold for the weak scalability properties, as clearly illustrated in Figure 10.

Overall, considering the first configuration N_1 with 6.144×10^6 particles, we observe that the serial implementation takes about nine days and eleven hours to complete. The same configuration runs in parallel on one GPU in about six minutes, which represents a speedup equal to 2160X. A comparison of the timings for this test between the pure MPI and hybrid MPI + multi-GPU configurations is illustrated in Figure 11. We observe that the simulation

Table 2. Computational times and speedup comparison for simulations with the analytical aerodynamic field. Different numbers of particles and GPUs have been adopted.

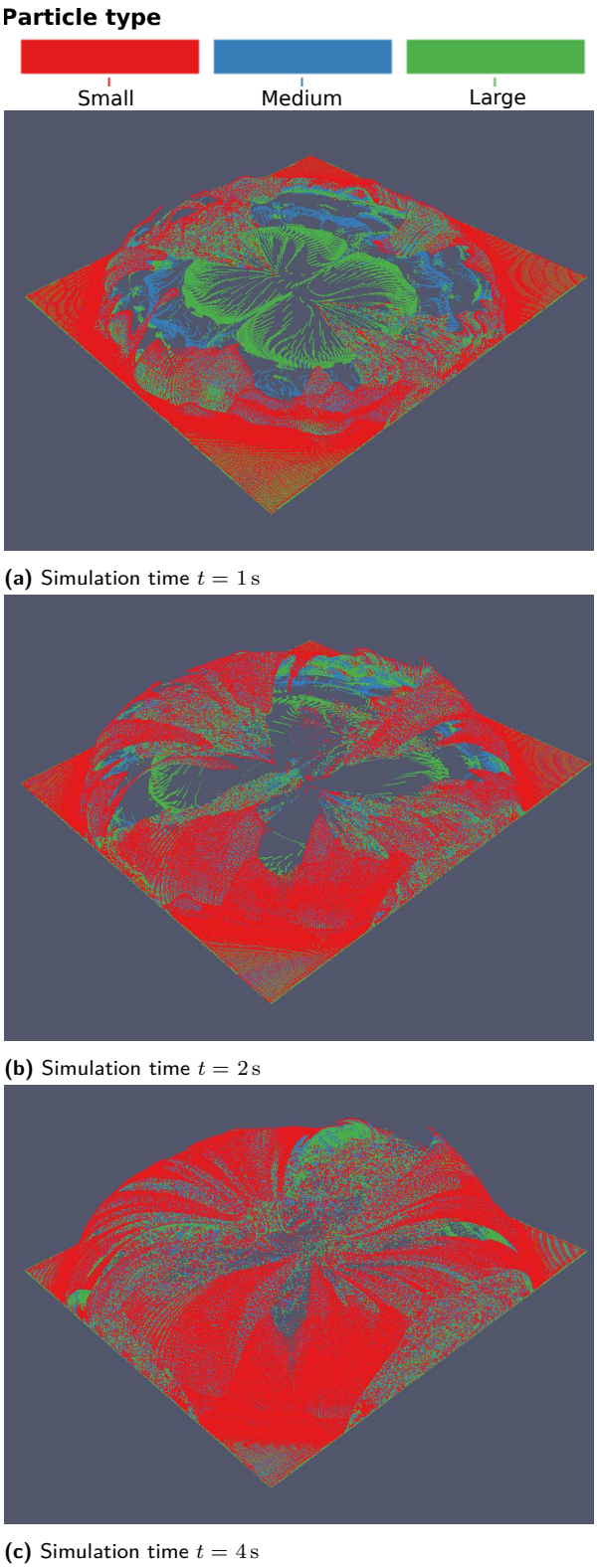
6.144×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:05:39		
2	0:02:58	1.90X	1.90X
4	0:01:33	1.91X	3.65X
8	0:0:49	1.90X	6.92X
16	0:0:26	1.89X	13.04X

12.288×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:11:46		
2	0:06:05	1.93X	1.93X
4	0:03:13	1.90X	3.66X
8	0:01:42	1.89X	6.93X
16	0:00:55	1.86X	12.84X

24.576×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:24:05		
2	0:12:32	1.92X	1.92X
4	0:06:40	1.88X	3.61X
8	0:03:31	1.90X	6.85X
16	0:01:53	1.87X	12.79X

49.152×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:48:23		
2	0:25:05	1.93X	1.93X
4	0:13:12	1.90X	3.67X
8	0:07:01	1.88X	6.90X
16	0:03:42	1.90X	13.08X

98.304×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	1:35:16		
2	0:49:37	1.92X	1.92X
4	0:26:05	1.90X	3.65X
8	0:13:37	1.92X	7.00X
16	0:07:16	1.87X	13.11X



(a) Simulation time $t = 1\,\mathrm{s}$

(b) Simulation time $t = 2\,\mathrm{s}$

(c) Simulation time $t = 4\,\mathrm{s}$

Figure 7. Positions of the particles during the simulation with the real aerodynamic field at three different time instants. The total number of particles is 6.144×10^6.

Table 3. Computational times and speedup comparison for serial and pure-MPI parallel simulations with the realistic aerodynamic field.

6.144×10^6 particles			
MPI tasks	timing (dd:hh:mm:ss)	speedup relative	speedup absolute
1	9:11:04:31		
2	4:22:38:19	1.91X	1.91X
4	2:14:09:23	1.91X	3.65X
8	1:08:40:51	1.90X	6.95X
16	0:17:17:12	1.89X	13.14X
32	0:09:12:00	1.88X	24.68X
64	0:04:53:22	1.88X	46.44X
128	0:02:40:47	1.83X	84.74X
256	0:01:40:00	1.61X	136.25X

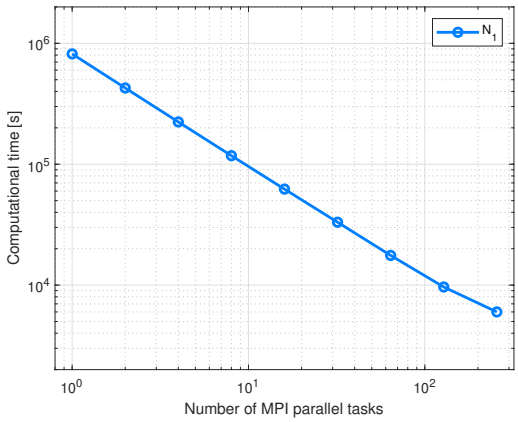


Figure 8. Strong scalability analysis for the pure-MPI parallel simulations with the realistic aerodynamic field and 6.144×10^6 particles.

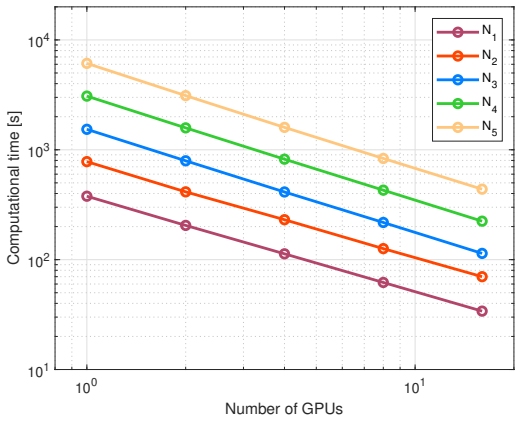


Figure 9. Strong scalability analysis for the simulations with the realistic aerodynamic field.

with 256 MPI-tasks is still almost two orders of magnitude slower than the one with one single GPU.

The simulations can run on the available hardware up to a limit of about 300×10^6 particles per GPU. Simulations with a number of particles close to that limit have been run with one, two, four, eight, and sixteen GPUs,

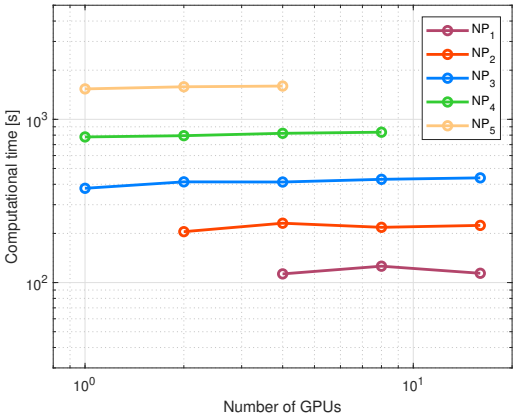


Figure 10. Weak scalability analysis for the simulations with the realistic aerodynamic field. For each line in the plot, the number of particles per GPU is kept constant and is equal to $\mathrm{NP}_i = 2^{i-1} \times 6.144 \times 10^6$ with $i \in \{1, \ldots, 5\}$.

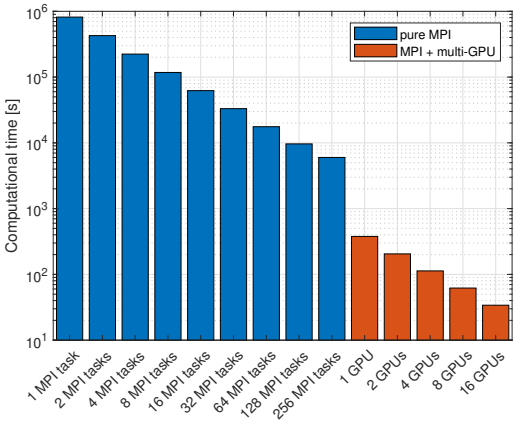


Figure 11. Bar chart for the simulation N_1 with the realistic aerodynamic field. The computational cost, expressed in time required by the pure-MPI and MPI + multi-GPU parallel simulations, is compared.

and the timings relative to their execution are reported in Table 5. Over the limit of about 300×10^6 particles per GPU, the maximum amount of available memory is exceeded, and the execution fails. Hence, we need at least four GPUs to run a simulation with more than one billion particles. In order to investigate and to have results about problems with these characteristics, we run a simulation with 1015.68×10^6 particles adopting two configurations, one purely MPI-parallel with 2000 parallel tasks, and one multi-GPU parallel with 1 CPU process distributing the work to 4 GPUs. The timings requested by these simulations with the described configurations are reported in Table 6.

Given that the particles in these simulations are microscopic (5 µm to 15 µm), one billion particles comprise only a tiny fraction of the domain. Therefore, more particles would be

Table 4. Computational times and speedup comparison for simulations that have been run with different numbers of particles and on 1, 2, 4, 8, and 16 GPUs. The realistic aerodynamic field has been adopted.

6.144×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:06:18		
2	0:03:25	1.84X	1.84X
4	0:01:53	1.81X	3.35X
8	0:01:02	1.82X	6.10X
16	0:00:34	1.82X	11.12X

12.288×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:13:00		
2	0:06:54	1.88X	1.88X
4	0:03:51	1.80X	3.38X
8	0:02:06	1.83X	6.19X
16	0:01:10	1.80X	11.14X

24.576×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:25:36		
2	0:13:14	1.94X	1.94X
4	0:06:53	1.92X	3.72X
8	0:03:38	1.89X	7.05X
16	0:01:54	1.91X	13.47X

49.152×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	0:51:21		
2	0:26:23	1.95X	1.95X
4	0:13:41	1.93X	3.75X
8	0:07:09	1.91X	7.18X
16	0:03:44	1.92X	13.75X

98.304×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	1:41:57		
2	0:51:51	1.97X	1.97X
4	0:26:39	1.95X	3.83X
8	0:13:54	1.92X	7.34X
16	0:07:18	1.90X	13.97X

necessary for a reliable representation of the real phenomenon. For simulations considering multi-billion particles, computational efficiency is a must, and assumptions such as the particle-particle collisions cannot be neglected anymore as it becomes an inherent part of the phenomena.

Table 5. Computational times for the realistic field and comparison of the speedup obtained using 1, 2, 4, 8, and 16 GPUs. The number of simulated particles is close to the limit fixed by memory resources.

288×10^6 particles			
nb. of GPUs	timing (hh:mm:ss)	speedup relative	speedup absolute
1	4:41:00		
2	2:26:31	1.92X	1.92X
4	1:19:03	1.85X	3.56X
8	0:41:49	1.89X	6.72X
16	0:22:18	1.88X	12.60X

Table 6. Computational times for the realistic field and comparison of the speedup obtained using 4 GPUs or 2000 MPI tasks.

1015.68×10^6 particles		
	timing (dd:hh:mm:ss)	speedup
2000 MPI tasks	1:08:28:20	
4 GPUs	0:04:20:14	7.49X

We also carry out a numerical assessment of the convergence order for the implemented method. In this experiment, we repeat the simulation with 6.144×10^6 particles six times, each iteration with a different timestep $\Delta t_i = 2^{2-i} \times 10^{-4}$ s with $i = \{1, \dots, 6\}$. We compute the errors over the positions and the velocities at time $t = 0.45$ s with respect to the values of the corresponding physical quantities provided by the Richardson extrapolation. Figure 12 shows a *loglog* plot of the ℓ^∞-norm of the error over the timestep. Since the semi-implicit Euler numerical scheme used is a first-order integrator, we observe that the performed analysis is in agreement with the expected order.

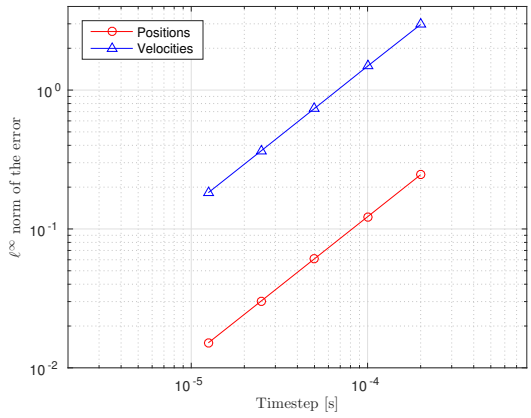


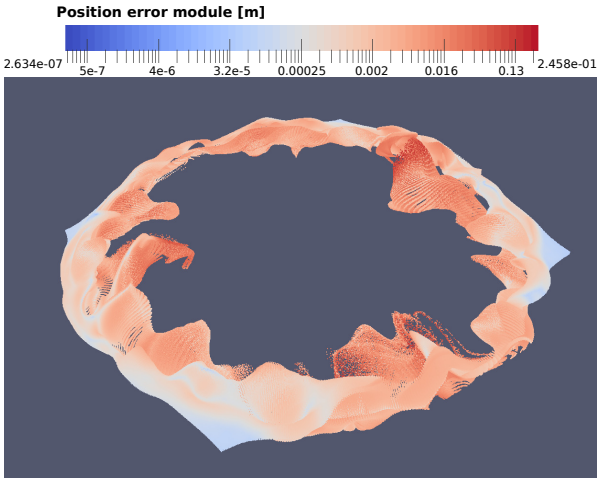
Figure 12. Illustration of ℓ^∞-norm of the error over the timestep size.

Figure 13 describes the spatial distribution of the error. In this figure, the particles are located at their respective computed positions. As expected, the error is higher for those particles which have traveled longer in comparison to those at the advancing front. The particles that, during the computation, leave the domain at least one time are not taken into account in the shown results. Indeed, when a particle goes out of the domain, its physical quantities are reset at its corresponding initial condition, thus resulting in a discontinuity on its trajectory.

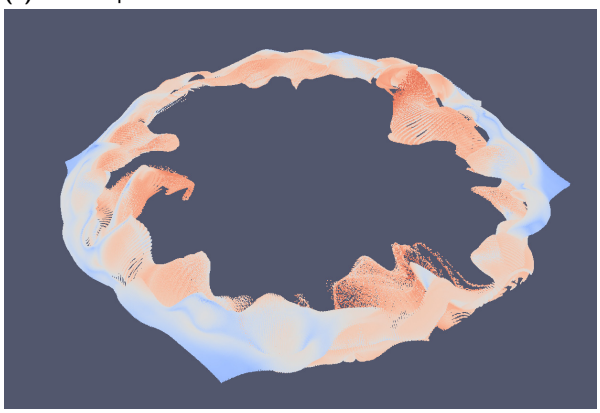
Analyzing each type of particle, it is possible to observe that the error is still decreasing with the timestep, and the convergence orders are always equal to 1, as observed in Figure 14. The larger particles are affected by smaller errors, either for their coordinates and velocities. This can be related to the higher inertia of those particles, which reduces the perturbations. We do not believe the reason to be related to the traveled distance, since the advancing front is composed of the three types of particle, as illustrated in Figure 15.

Conclusions

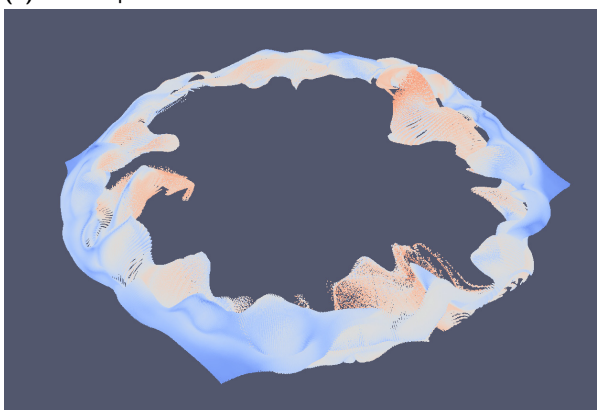
This study proposes an algorithm that uses a semi-implicit Euler scheme to simulate



(a) Timestep $\Delta t = 2 \times 10^{-4}$ s.



(b) Timestep $\Delta t = 5 \times 10^{-5}$ s.



(c) Timestep $\Delta t = 1.25 \times 10^{-5}$ s.

Figure 13. Illustration of the ℓ^∞-norm of the error of the positions in a *log* scale.

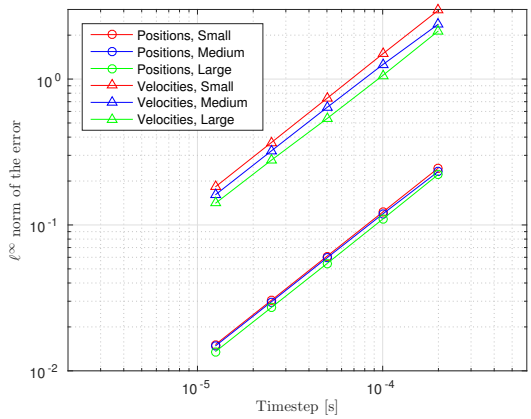


Figure 14. Illustration of the ℓ^∞-norm of the error for each type of particle.

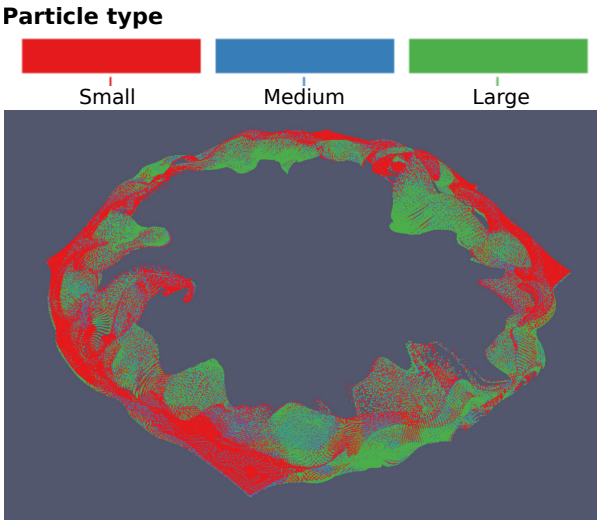


Figure 15. Distribution of the different types of particle at time $t = 0.45$ s.

the dynamics of particles in a Lagrangian way. The proposed methodology is applied to simulate brownout using an MPI + multi-GPU approach. In terms of computational performance, the proposed implementation shows promising initial results, especially when CUDA programming on graphics hardware is used. Overall, our results show big speedup improvements when comparing the brownout simulations using the GPU-parallel and the serial implementation. For example, in the case of a simple test with 6.144×10^6 particles, the serial program runs for nine days and eleven hours to complete fifty seconds of brownout simulation. The same simulation runs in parallel on a single GPU for six minutes, about 2160 times faster than the serial program.

In this paper, we also evaluate brownout simulation's performance considering different numbers of particles and GPUs. From the results, we observe excellent weak and strong scalability properties of the GPU-parallel implementation. Also, we optimize the request

of GPU memory to increase the number of particles that can be managed by any single GPU. Within this setup, the largest number of particles that can be simulated is about 288×10^6 per GPU. Considering this latter result and the excellent weak and strong scalability characteristics, the proposed implementation shows to be capable of simulating billions of particles in a relatively short amount of time. This potential is demonstrated by simulating one billion particles using four GPUs. The simulation is complete after four hours and a half. The same test is 7.5 times slower if run with MPI on 2000 parallel tasks. We conclude that the GPU parallelization allows performing realistic brownout simulations in a reasonable short computational time using a small number of GPUs. The same performance on a multiprocessor environment can only be achieved by employing tens of thousands of threads.

In terms of numerical developments, semi-implicit Euler integrators require a very small timestep to guarantee numerical stability and accuracy. This can be a favorable computational characteristic for these simulations, as it naturally allows us to take into account particle-particle collisions, saltation of particles, and the neglected forces (*i.e.* the Saffman, Magnus, and Basset forces). To implement these additional features, a path forward would be to employ the framework provided by the open-source library AMReX (Zhang (2019)), which offers powerful functionalities for particles handling. Specifically, AMReX allows simulating a collection of particles in a hybrid MPI + OpenMP + Multi-GPU parallel environment. Particles are immersed in a structured mesh which is distributed through the different MPI parallel tasks. Fixing a cut-off distance, which defines a "neighborhood" for each particle, we can use AMReX's routines to fill "neighbor buffers" containing copies of the neighboring ones. In this way, collisions and short-range forces can be efficiently computed looping only a few, closer particles.

However, it has been shown in Porcù (2013) and Miglio et al. (2018) that implicit high-order methods, such as *spectral variational integrators*, can also be highly efficient, allowing a larger timestep size to achieve the same level of accuracy in comparison to lower order methods. In this direction, for future development, it would also be interesting to investigate the implementation of high-order time integrators in brownout simulations.

References

Braithwaite M, Groh S and Alvarez E (1997) Spatial Disorientation in US Army Helicopter Accidents: An Update of the 1987-92 Survey to Include 1993-95. *Army Aeromedical Research Lab, Fort Rucker, AL*.

Brüggemann J (2012) Eurocopter ec135 d-hzsg brownout. https://commons.wikimedia.org/wiki/File:Eurocopter_EC135_D-HZSG_Brownout_I_Br%C3%BCggemann.jpg.

Doehler HU and Peinecke N (2010) Image-based drift and height estimation for helicopter landings in brownout. In: *International Conference Image Analysis and Recognition*. Springer, pp. 366-377.

Duda H, Advani SK and Potter M (2013) Design of the DLR AVES research flight simulator. In: *AIAA Modeling and Simulation Technologies (MST) Conference*, pp. 4737.

D'Andrea A (2009) Numerical analysis of unsteady vortical flows generated by a rotorcraft operating on the ground: The first assessment of helicopter brownout. In: *65th Annual Forum of the American Helicopter Society, Grapevine, TX*. pp. 423-446.

D'Andrea A (2010) Unsteady numerical simulations of helicopters and tiltrotors operating in sandy-desert environment. In: *Proceedings of the American Helicopter Society Specialist's Conference on Aeromechanics*. pp. 667-690.

D'Andrea A (2011) Development and application of a physics-based computational tool to simulate helicopter brownout. In: *37th European Rotorcraft Forum Proceedings, Gallarate (VA), Italy*. pp. 903-916.

Durnford SJ, Crowley JS, Rosado NR, Harper J and DeRoche S (1995) Spatial Disorientation: A Survey of US Army Helicopter Accidents 1987-1992. *Army Aeromedical Research Lab, Fort Rucker, AL*.

Gerlach T (2011) Visualisation of the brownout phenomenon, integration and test on a helicopter flight simulator. *The Aeronautical Journal* 115(1163): 57-63.

Ghosh S, Lohry M and Rajagopalan R (2010) Rotor configurational effect on rotorcraft brownout. In: *28th AIAA Applied Aerodynamics Conference*, pp. 4238.

Govindarajan BM (2011) *Evaluation of particle clustering algorithms in the prediction of brownout dust clouds*. PhD Thesis, University of Maryland. https://drum.lib.umd.edu/handle/1903/11846.

Govindarajan B, Leishman JG and Gumerov NA (2013) Particle-clustering algorithms for the prediction of brownout dust clouds. *AIAA journal* 51(5): 1080-1094. American Institute of Aeronautics and Astronautics.

Hoerner SF (1965) *Fluid-dynamic drag: practical information on aerodynamic drag and hydrodynamic resistance*.

Hu Q, Syal M, Gumerov NA, Duraiswami R and Leishman JG (2011) Toward improved aeromechanics simulations using recent advancements in scientific computing. In: *Proceedings 67th Annual Forum of the American Helicopter Society, Virginia Beach, VA*. pp. 3-5.

Jasion G (2013) *Toward a physics based entrainment model for simulation of helicopter brownout*. PhD Thesis, University of Southampton. https://eprints.soton.ac.uk/355705/.

Johnson RW (2016), *Handbook of fluid dynamics. CRC Press*

Keller JD, Whitehouse GR, Wachspress DA, Teske ME, and Quackenbush TR (2006) A physics-based model of rotorcraft brownout for flight simulation applications. In: *Annual forum proceedings* 62(2): pp. 1098. American Helicopter Society, Inc.

Kirk DB and Wen-mei WH (2012) *Programming massively parallel processors: a hands-on approach*. Newnes.

Lohry MW, Ghosh S and Rajagopalan RG (2011) Graphics hardware acceleration for rotorcraft brownout simulation. In: *20th AIAA Computational Fluid Dynamics Conference*. p. 3224.

Miglio E, Parolini N, Penati M and Porcù R (2018) High-order variational time integrators for particle dynamics. *Communications in Applied and Industrial Mathematics, Sciendo* 9(2): 34-49.

Nathan N and Green R (2008) Measurements of a rotor flow in ground effect and visualization of the brown-out phenomenon. In: *64th Annual Forum of the American Helicopter Society, Montréal, Canada*.

Nathan N and Green R (2009) Flow visualisation of the helicopter brown-out phenomenon. *Aeronautical Journal* 113(1145): 467-478.

NVIDIA (b) *CUDA C Programming Guide*. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.

PGI (2019) *PGI CUDA Fortran Programming Guide and Reference*. https://www.pgroup.com/resources/docs/19.5/pdf/pgi19cudaforug.pdf.

Phillips C, Brown R and Kim HW (2011) Helicopter brownout-can it be modelled? *Aeronautical Journal* 115(1164): 123-133.

Phillips C and Brown RE (2008) The effect of helicopter configuration on the fluid dynamics of brownout. In: *34th European Rotorcraft Forum*. pp. 2398-2426.

Phillips C and Brown RE (2009) Eulerian simulation of the fluid dynamics of helicopter brownout. *Journal of Aircraft* 46(4): 1416-1429.

Phillips C, Kim HW and Brown RE (2010) The flow physics of helicopter brownout. *66th American Helicopter Society Forum: Rising to New Heights in Vertical Lift Technology*

Porcù R (2013) *Metodi numerici e tecniche di programmazione per l'accelerazione di un modello di dinamica di particelle non interagenti*. Master's Thesis, Politecnico di Milano.

Ruetsch G and Fatica M (2013) Cuda fortran for scientists and engineers. *Elsevier*.

Sabbagh L (2006) Flying blind in iraq: Us helicopters navigate real desert storms. *Popular Mechanics* 3.

Sanders J and Kandrot E (2010) *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional.

Savage J, Harrington W, McKinley RA, Burns HN, Braddom S and Szoboszlay Z (2009) 3D-LZ helicopter ladar imaging system. In: *Laser Radar Technology and Applications XV* 7684: pp. 768407. International Society for Optics and Photonics.

Schuetz CA, Stein Jr EL, Samluk J, Mackrides D, Wilson Jp, Martin RD, Dillon TE, and Prather DW (2009) Studies of millimeter-wave phenomenology for helicopter brownout mitigation. In: *Millimetre Wave and Terahertz Sensors and Technology II* 7485: pp. 74850F.

International Society for Optics and Photonics.

Syal M (2012) *Development of a Lagrangian-Lagrangian methodology to predict brownout dust clouds*. PhD Thesis, University of Maryland. http://drum.lib.umd.edu/handle/1903/13095.

Syal M and Leishman JG (2011) Comparisons of Predicted Brownout Dust Clouds with Photogrammetry Measurements. In: *67th Annual Forum of the American Helicopter Society, Virginia Beach, VA*.

Syal M and Leishman JG (2013) Modeling of bombardment ejections in the rotorcraft brownout problem. *AIAA journal* 51(4): 849-866. American Institute of Aeronautics and Astronautics.

Syal M, Rauleder J, Tritschler J and Leishman JG (2011) On the possibilities of brownout mitigation using a slotted-tip rotor blade. In: *29th AIAA Applied Aerodynamics Conference*. pp.3183.

Szoboszlay ZP, Turpin TS and McKinley RA (2009) Symbology for brown-out landings: the first simulation for the 3D-LZ program. In: *65th Annual Forum of the American Helicopter Society, Grapevine, TX*.

Tanner PE (2011) Photogrammetric characterization of a brownout cloud. In: *67th Annual Forum of the American Helicopter Society, Virginia Beach, VA*.

Thomas S (2013) *A GPU-accelerated, hybrid FVM-RANS methodology for modeling rotorcraft brownout*. PhD Thesis, University of Maryland. http://drum.lib.umd.edu/handle/1903/14832.

Tritschler JK, Celi R and Leishman JG (2014) Methodology for rotorcraft brownout mitigation through flight path optimization. *Journal of Guidance, Control, and Dynamics* 37(5): 1524-1538.

Viertler F and Hajek M (2015) Requirements and design challenges in rotorcraft flight simulations for research applications. In: *AIAA Modeling and Simulation Technologies Conference*, pp. 1808.

Vyrnwy-Jones P (1988) Disorientation Accidents and Incidents in US Army Helicopters, 1 January 1980-30 April 1987. *Army Aeromedical Research Lab, Fort Rucker, AL*.

Wachspress DA, Whitehouse GR, Keller JD, McClure K, Gilmore P and Dorsett M (2008) Physics based modeling of helicopter brownout for piloted simulation applications. Technical report, DTIC Document.

Wadcock AJ, Ewing LA, Solis E, Potsdam M and Rajagopalan G (2008) Rotorcraft downwash flow field study to understand the aerodynamics of helicopter brownout. Technical report, DTIC Document.

Wong OD and Tanner PE (2010) Photogrammetric measurements of an EH-60L brownout cloud. In: *66th Annual Forum of the American Helicopter Society, Phoenix, AZ*.

Williams GSS (2010) Osprey takes on brown-out in afghanistan. http://commons.wikimedia.org/wiki/File:Flickr_-_DVIDSHUB_-_Osprey_Takes_on_'Brown-Out'_in_Afghanistan.jpg.

Zhang W, Almgren A, Beckner V, Bell J, Blaschke J, Chan C, Day M, Friesen B, Gott K, Graves D, Katz MP, Myers A, Nguyen T, Nonaka A, Rosso M, Williams S and Zingale M (2019) AMReX: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37). https://escholarship.org/uc/item/00j3z3rd