eCSE05-3 Technical Report

Sam Owens, Paul Bartholomew & Maarten van Reeuwijk

August 15, 2023

Abstract

This report presents the work conducted for the ARCHER2 eCSE05-3 project. The goal was to improve the scalability and object-resolving capability of the uDALES framework (Suter et al., 2022). There are three specific changes involved, which together constitute an upgrade of the codebase from version 1.0 to 2.0. Firstly, uDALES v1.0 has a one-dimensional domain decomposition, which means it is not able to effectively utilise the large number of processors available on ARCHER2. To address this, a two-dimensional domain decomposition was implemented in uDALES v2.0 using the library 2DECOMP&FFT (Li and Laizet, 2010), and the model is shown to scale well for realistic use cases. Secondly, the pressure solver in uDALES v1.0 is not capable of using the Fast Fourier Transform (FFT) algorithm with inflow-outflow boundary conditions, instead relying on slower algorithms. This was made possible in uDALES v2.0, and the implementation has been validated using canonical urban cases. Finally, the immersed boundary method used in uDALES v1.0 only permits obstacles that align with the Cartesian computational grid, thus hindering its ability to model realistic urban geometries. uDALES v2.0 instead describes the geometry using an unstructured triangulated surface, which required a fundamental reformulation of the immersed boundary method and the pre-processing associated with the geometry. The implementation has been validated in both aligned and non-aligned cases.

1 Introduction

The urban atmospheric large-eddy simulation framework uDALES v1.0 (Suter et al., 2022) has several key limitations. It has a one-dimensional domain decomposition, which is implemented using bespoke routines relying on MPI. This means the maximum number of processors that it is possible to use is limited to the number of grid points in one dimension. To illustrate, a typical uDALES v1.0 simulation uses 128-1024 points in each direction, thus only potentially leveraging 1-8 nodes on ARCHER2.

The pressure solver in uDALES is based on the Fast Fourier Transform (FFT) algorithm. For periodic lateral boundary conditions, the FFT is used

to solve the streamwise (x) and spanwise (y) directions, with the vertical (z) direction solved using Gaussian elimination. In uDALES v1.0, for inflow-outflow x boundaries, y is periodic and is solved using the FFT, while x and z are solved using cyclic reduction. This approach is practically around 50% slower than the periodic case.

The immersed boundary method (IBM) is used to resolve the flow past obstacles, with wall functions parametrising subgrid processes close to surfaces, yielding surface fluxes of momentum (shear stress) and sensible heat. The IBM and wall functions are implemented with the assumption that the facets sit exactly on computational cell edges, and so in pre-processing the geometry is constructed under this constraint in a process referred to as voxelisation. This conceptually simplifies the implementation but obviously introduces errors if the original geometry does not actually align with the grid.

uDALES v2.0 has been developed to address these issues. It has a two-dimensional domain decomposition, implemented using the 2DECOMP&FFT library, which was chosen because it has been designed with ease-of-use and scalability in mind. The implementation of this is discussed in section 2. The pressure solver had to be completely rewritten in account of the domain decomposition, and also because the library used for the FFTs (FFTW) is different to uDALES v1.0 (FFTPACK). In addition, it was made capable of using FFT-based methods for inflow-outflow boundary conditions in both x and y, specifically the discrete cosine transform (DCT). This is discussed in section 3. Finally, the geometry is specified as an unstructured triangulated surface, or triangulation, which is given independently of the grid using the STL file format. The triangles are referred to as facets, and have properties that govern their effect on the flow and their surface energy balance. The new IBM and wall function approach is described in section 4.

2 2D domain decomposition

In a 2D domain decomposition, the domain is divided into a number of sub-domains, or 'pencils', with each operated on by one processor (rank). Setting up the basic data structures required for the 2D domain decomposition was straightforward with the 2DECOMP&FFT library. The normal orientation of the pencils is such that x and y are parallelised. Only in the pressure solver is the data transposed, i.e. moved from one pencil orientation to another.

Several modifications to the codebase were made in order to better meet the uDALES use case. uDALES employs halo cells, which means that arrays storing field values on each pencil also stores some values on adjacent pencils. Therefore, each array is slightly larger than solely the number of points in the pencil. This is done because these halo cells are included in the stencil of the finite difference scheme for cells at the edge of each pencil. This naturally requires communication between adjacent ranks every timestep. 2DECOMP&FFT does offer halo cell support, but the functionality is slightly limited. Specifically, there is a single subroutine that takes as an argument an array the size of the

pencil, copies the content into a larger array the size of the pencil plus halos, populates the halo cells by communicating with adjacent ranks, and returns the larger array. Given arrays in uDALES are allocated with halos, it made sense to extract the communication functionality to a standalone subroutine. Whilst this was the most significant change, the array allocation wrapper subroutines were also modified so as to include halo cells. This makes array allocation in uDALES less manual. These changes have been verified in a number of tests, and are currently contained in a fork of the 2DECOMP&FFT library, but will be merged into the main branch in the near future.

Figures 1a and 1b shows the strong scaling for periodic simulations with 1024^3 and 2048^3 points respectively. There is no geometry, hence the IBM not being used, so purely the lower-level routines are being tested. These figures demonstrate that the code scale well up to a large number of processors on ARCHER2.

The case setup shown in section 4 for a non-rotated geometry is used to test the scaling with inflow-outflow boundary conditions, representative of flow conditions typically found in uDALES simulations. For scaling purposes higher resolutions are used, but temperature is not solved for, and the domain width is smaller. The domain size is $190 \times 114 \times 114$ m^3, and the smaller case has a grid size of $640 \times 384 \times 384$, and larger case has a grid size of $1280 \times 768 \times 768$. The speed-up for each case is shown in figures 2a and 2b, respectively. For the smaller case a good parallel efficiency (above 80%) is observed until 32 nodes (4,096 MPI ranks) which corresponds to ≈ 23 k grid points per core. A similar reduction of parallel efficiency below 25 k grid points per core was observed when running the validation case described in section 4, suggesting this is a practical lower limit of work for uDALES. The larger case required 4 nodes minimum and demonstrated good scaling up to 128 ARCHER2 nodes (16,384 CPUs).

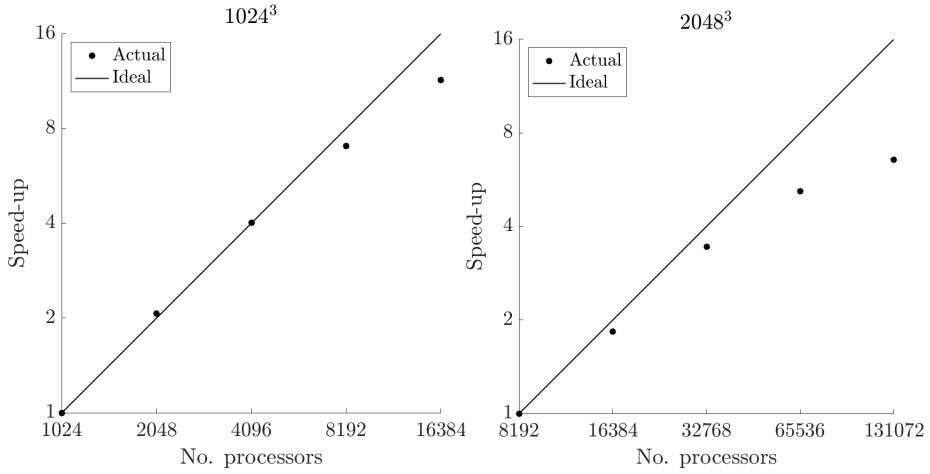


Figure 1: Strong scaling for case without IBM and periodic boundary conditions; (a) 1024^3 points, (b) 2048^3 points.

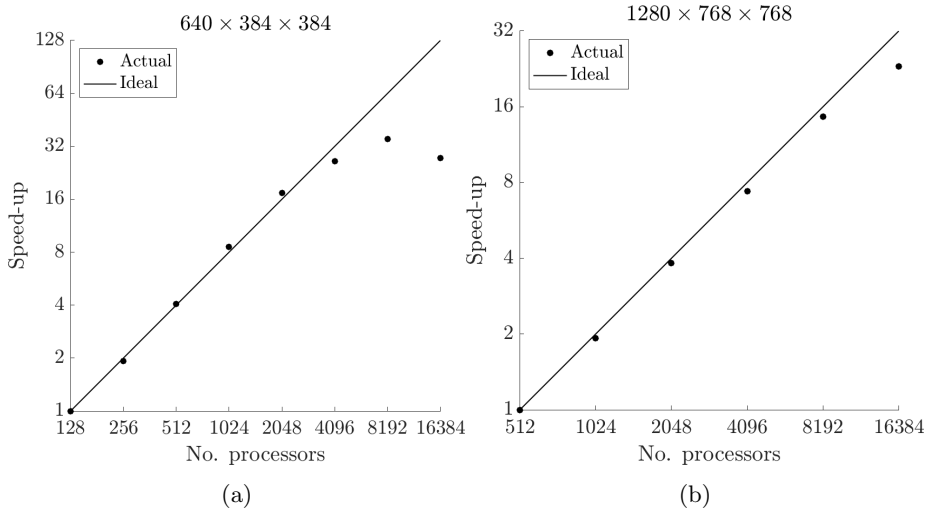


Figure 2: Strong scaling for case with IBM and inflow-outflow boundary conditions; (a) $640 \times 384 \times 384$ points, (b) $1280 \times 768 \times 768$ points.

3 FFT-based pressure solver

For incompressible flows such as those modelled by uDALES, the pressure satisfies a Poisson equation. In uDALES v2.0, the Poisson equation is solved using the FFT in the x and y directions and by default Gaussian elimination (GE) in z. It is possible to use the FFT in z when the grid is equidistant, however there is unlikely to be a performance gain using the FFT rather than GE. To solve each direction requires transposing data between the three pencil orientations, which is achieved using routines from the 2DECOMP&FFT library.

The velocity boundary conditions determine the specific type of transform performed. If velocity is periodic in a given direction, then the pressure is also periodic and the regular discrete Fourier transform (DFT) is used. If the velocity is inflow-outflow, then the pressure has a Neumann boundary condition and a discrete cosine (DCT) transform for a staggered grid is used. The details of these transforms can be found in Schumann and Sweet (1988). uDALES v1.0 used the FFTPACK implementation of the DFT, but the staggered DCT transform is not included in this library. However, it is included in the FFTW library, so this was used in uDALES v2.0.

As the name suggests, the 2DECOMP&FFT library contains extensive auxiliary FFT functionality, but this was not used for uDALES v2.0. This is because the library is primarily designed for performing 3D DFTs, i.e. using the DFT to solve the z direction as well as x and y. Since the z direction is never periodic, this was not useful for uDALES. An option that was considered but not adopted was to build a 2D DFT solver for periodic lateral boundaries using the lower-level routines available in 2DECOMP&FFT. Due to the nature of Fourier transforms, one must be very precise with array sizes and data types, particularly when transforms are performed successively, and it proved difficult to do this with the existing functionality in the library.

When using inflow-outflow boundary conditions and the DCT, the top boundary condition must be treated with care. Specifically, fluid must be allowed to pass through the top of the domain, which is equivalent to setting a non-zero vertical velocity (w) there. The physical reasoning for this is that the incoming flow responds to the geometry, which may result in net motion in the z direction. This non-zero w is determined by the plane-averaged pressure gradient at the top, and without it, the flow is not incompressible at the top, which is unacceptable both numerically and physically. In uDALES v1.0, w was always set to zero, though the flow was incompressible due to the nature of the cyclic reduction based pressure solver. This means that there was no numerical issue, but the non-physical top boundary condition produced noticeable strange behaviour in flows with significant vertical motion. The novel top boundary has been implemented successfully in uDALES v2.0; the proof of which is the fact that the divergence of the velocity field is always zero, and no strange physical behaviour has been noticed thus far.

4 Immersed boundary method

Conceptually, one expects the non-porous, stationary obstacles modelled in uDALES to be no-slip, no-penetration, and to have zero flux across boundaries (apart from those parametrised by wall functions). This last aspect is particularly important for scalars - the IBM needs to be conservative, meaning there should be is no diffusion or advection of heat, moisture, and pollutant concentration between fluid and solid regions. Using the triangulation describing the geometry, the domain can be divided into fluid and solid regions for a given computational grid, and boundary points for both types are identified as those with at least one neighbour of the other type. This is a necessary step in pre-processing, and was implemented first in Matlab then in Fortran.

As far as this aspect to the IBM implemented in uDALES v2.0 is concerned, the boundary across which scalars is conserved is the literal interface between the fluid and solid cells, aligned with the cell edges. This means that when the facets are also aligned with the cell edges, the desired boundary conditions are satisfied exactly. If not, what is meant by no-slip, no-penetration, and zero-flux must be reinterpreted slightly. This approach interprets no-slip and no-penetration to simply mean the velocity at solid points is zero, and the zero flux condition applies locally between each pair of fluid-solid neighbours.

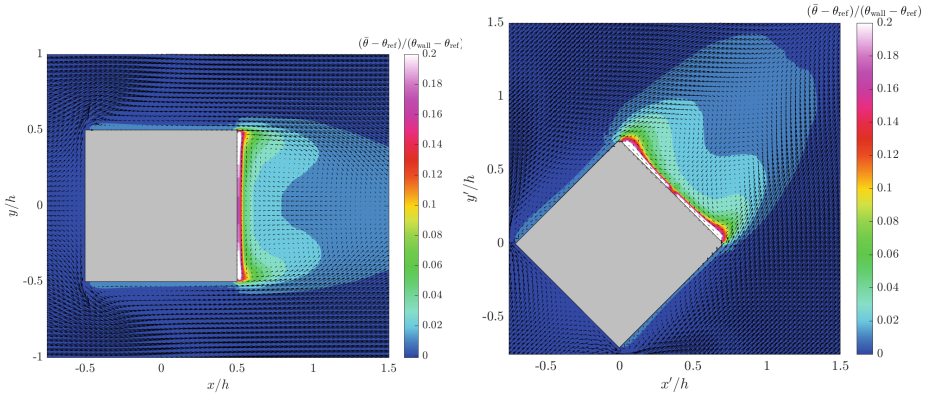
An alternative approach would be to enforce that the value of a variable on the boundary as determined by interpolating the values of nearby points satisfies the boundary condition. For example, one method would be for solid cells that have at least one fluid neighbour, set the value such that when interpolated across the boundary in the normal direction to an 'image point' inside the fluid region, the value or flux on the boundary itself satisfies the desired condition. The obvious limitation with this is the fact that interpolation of any kind is just an approximation. This approach is often used with laminar flows and when using direct numerical simulation (DNS), because it is argued that since the flow is fully-resolved, interpolation is valid. It is not as common with LES because the flow is not fully resolved. Another inherent challenge with this approach is that it is more complicated to ensure global conservation - the net flux between fluid and solid regions due to the advection and turbulent diffusion terms should be zero. In the current approach, global conservation is guaranteed since local conservation is ensured by enforcing zero flux between each fluid-solid pair. In the proposed approach, the zero flux condition is at the boundary itself in an interpolated sense so local conservation no longer holds, but global conservation would still be required. It was decided that this approach was too challenging, and therefore, this aspect of the IBM is conceptually unchanged from uDALES v1.0, though it was implemented from scratch in order to accommodate the 2D domain decomposition and to simplify the code.

The novelty of uDALES v2.0 is in the treatment of the parametrised flux at the boundary, namely the surface shear stress and sensible heat flux, as determined by wall functions. Given a conservative IBM, these fluxes are the only mechanism by which the surface exerts friction drag and exchanges heat with the fluid. Another conservation principle that must be applied here is that

the total heat flux into the fluid is equal to the total heat flux out of the surface in terms of the surface energy balance. This requirement motivates the approach, and whilst this conservation principle is not relevant for momentum flux, the method is also applied for that calculation. The surface is divided cell-wise for each grid into sections, such that each section lies in only one cell, which may be fluid or solid. Each section imparts a flux that is felt by a single fluid boundary point. The form of the wall functions are unchanged from uDALES v1.0, and follow Louis (1979) and the extension by Uno et al. (1995). They generally depend on facet properties (e.g. roughness length), the distance between the fluid boundary point and the wall, and the flow variables at the fluid boundary point. In cases where the facet is not aligned with the grid, the flow variables are reconstructed in a similar manner as described by Ma and Liu (2017). Once the surface shear stress is determined, it must be transformed back to the grid directions as it is a tensor quantity. When using the surface energy balance model, the heat flux is also removed from the facet, and since each section is accounted for exactly once and the sum of their area equals the total surface area, conservation is guaranteed.

The IBM and wall function implementation has been verified in several cases. For illustrative purposes, a validation case is presented that involves heat and inflow-outflow boundary conditions. The set-up is based on an LES study by Boppana et al. (2013), in which the flow is compared against an experiment conducted by Richards et al. (2006). Here the simulations are run at full scale, i.e. 100 times larger than those studies. The geometry is a single cube with side length $h = 19$ m, with its leeward face temperature (θ_{wall}) set to a higher temperature than the ambient air (θ_{ref}). In the original set-up the cube is grid-aligned, and in order to test the ability of the IBM to handle non-aligned geometries, this is compared with a case where the cube has been rotated with respect to the grid, but with the same physical oncoming flow. The original set-up domain size is $L_x \times L_y \times L_z = 10h \times 8h \times 6h$, with grid size $N_x \times N_y \times N_z = 320 \times 256 \times 192$. The rotated set-up domain size is $8h \times 8h \times 6h$, with grid size $256 \times 256 \times 192$. A turbulent inflow boundary condition was generated using a precursor simulation that was set up to match the velocity and turbulent kinetic energy profiles of the comparison studies.

Figure 3 shows the mean scaled potential temperature $(\bar{\theta} - \theta_{\mathrm{ref}})/(\theta_{\mathrm{wall}} - \theta_{\mathrm{ref}})$ and mean velocity vectors in the wake of the cube at $z/h = 0.5$ for each case. There is fairly good agreement, indicating the wall function implementation is reasonably invariant to whether the geometry is aligned with the grid.



(a) Original (b) Rotated

Figure 3: Contours of mean scaled temperature and mean velocity vectors at $z/h = 0.5$.

5 Conclusion

The uDALES framework has been upgraded in order to address its major limitations. The 2D domain decomposition provided by 2DECOMP&FFT means the code is able to use a large number of processors on ARCHER2, and scales as expected. The fully FFT-based pressure solver means that inflow-outflow simulations can be reliably simulated with the same performance as periodic simulations. The novel immersed boundary method is able to model non-aligned geometries more faithfully. The features of uDALES v2.0 and their validation are being written up as a journal paper.

This development has many applications to research into urban flows. The performance upgrade makes it possible to run simulations at much higher resolution, thus enhancing the ability to capture small-scale flow features. It also makes processes with relatively long timescales easier to simulate; for example modelling a complete diurnal cycle of the atmospheric boundary layer. The improved immersed boundary method is particularly advantageous for simulations involving radiation, and when using the model to resolve realistic buildings. Finally, the fact that 2DECOMP&FFT is used for low-level functionality means that any useful future development of this library and other frameworks using it can potentially be harnessed; for example the new IO functionality using ADIOS2 (Godoy et al., 2020) that was part of the ARCHER2 eCSE03-02 project. Further work is currently underway to accelerate pre- and post-processing routines and to enable the use of GPUs via the Excalibur project Turbulence at the Exascale.

References

V. Boppana, Z.-T. Xie, and I. P. Castro. Large-eddy simulation of heat transfer from a single cube mounted on a very rough wall. *Boundary-layer meteorology*, 147(3):347–368, 2013.

W. F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck, et al. Adios 2: The adaptable input output system. a framework for high-performance data management. *SoftwareX*, 12:100561, 2020.

N. Li and S. Laizet. 2DECOMP&FFT-a highly scalable 2D decomposition library and FFT interface. In *Cray user group 2010 conference*, pages 1–13, 2010.

J.-F. Louis. A parametric model of vertical eddy fluxes in the atmosphere. *Boundary-Layer Meteorology*, 17(2):187–202, 1979.

Y. Ma and H. Liu. Large-eddy simulations of atmospheric flows over complex terrain using the immersed-boundary method in the weather research and forecasting model. *Boundary-layer meteorology*, 165:421–445, 2017.

K. Richards, M. Schatzmann, and B. Leitl. Wind tunnel experiments modelling the thermal effects within the vicinity of a single block building with leeward wall heating. *Journal of Wind Engineering and Industrial Aerodynamics*, 94 (8):621–636, 2006.

U. Schumann and R. A. Sweet. Fast fourier transforms for direct solution of poisson's equation with staggered boundary conditions. *Journal of Computational Physics*, 75(1):123–137, 1988.

I. Suter, T. Grylls, B. S. Sützl, S. O. Owens, C. E. Wilson, and M. van Reeuwijk. udales 1.0: a large-eddy simulation model for urban environments. *Geoscientific Model Development*, 15(13):5309–5335, 2022.

I. Uno, X. M. Cai, D. Steyn, and S. Emori. A simple extension of the louis method for rough surface layer modelling. *Boundary-Layer Meteorology*, 76: 395–409, 1995.

Acknowledgement

This work was funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (http://www.archer2.ac.uk)