2.290 Final Project: Modeling Air Pollution in OpenFOAM

Natasha Stamler

May 2022

1 Background

Air pollution kills seven million people around the world every year through stroke, heart disease, lung cancer, and acute respiratory infections [1]. However, air pollution can be difficult to quantify with field measurements due to its spatial and temporal variability [14]. Modeling can provide valuable insight into this variability in pollution concentration. These models can then be used to predict human impacts and to inform mitigation measures, such as changes to the built environment. Computational Fluid Dynamics (CFD) provides the highest spatial and temporal resolution for air pollution modeling. Other models, such as AEROMOD, a Gaussian Plume Model (GPM) developed by the US Environmental Protection Agency (EPA), cannot accurately compute concentrations for complex building geometries or time scales finer than an hour [6]. CFD is thus increasingly being used to understand the dynamics of urban physics at scales from the meteorological to human to address problems related to health, energy, and climate [2, 16, 23]. However, CFD's potential for high computational cost necessitates the development of solvers that implement efficient schemes.

2 Problem Statement

Air pollution modeling is currently largely limited to complex, government-run models, or expensive software, such as Ansys Fluent. Free, open-source CFD software, such as OpenFOAM, only track wind velocity and pressure, rather than particle concentration, with their built-in solvers. However, open-source software provides an opportunity by allowing researchers to modify and add to their existing code. This project develops a solver for aerial pollutant transport building off an existing OpenFOAM CFD solver. It then demonstrates that this solver can be applied to a simple test case. The successful implementation of this solver could enable future evaluation of air pollution impacts at lower cost.

3 Solver

Running OpenFOAM v5 [21] in Windows 10 with blueCFD-Core 2017 [3], this study builds on *buoyantBoussinesqPimpleFoam*, OpenFOAM's transient solver for buoyant, turbulent flow of incompressible fluids. This solver uses the PIMPLE algorithm, outlined in Figure 1, which combines the Pressure Implicit with Splitting of Operators (PISO) [9] and Semi-Implicit Method for Pressure Linked Equations (SIMPLE) [4] algorithms.

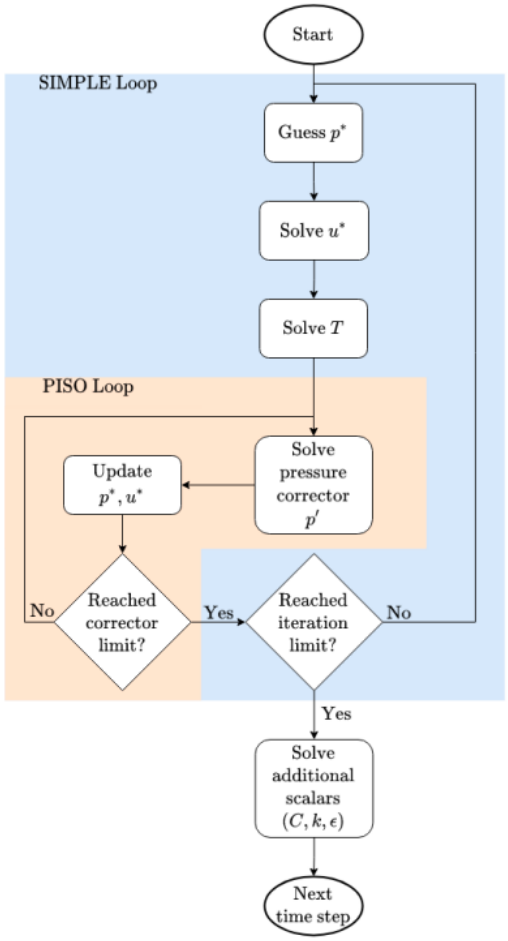


Figure 1: PIMPLE algorithm flowchart [18].

Using a transient model instead of a steady state one enables the handling of conditions that change with time, such as time-varying pollutant sources. Air was assumed to be incompressible due to its low speed (less than 100 m/s).

Environments that are relevant to air pollution are inherently turbulent. An example of this is urban canyons, which are dense building layouts that trap air pollution in cities. They have large length scales from the buildings and the potential for high wind speeds, which can lead to high Reynolds numbers (Re) on the order of $10^7 - 10^9$. This turbulence is represented using Reynolds-averaged Navier-Stokes (RANS) turbulence modeling. RANS is used despite the higher accuracy of Large Eddy Simulation (LES) for simulations such as street canyon airflow [5, 16] due to the computational intensity of LES that makes it infeasible for domains larger than idealized urban canyons [7, 18]. Similarly, most large-scale, urban simulations use RANS turbulence modeling [19].

This solver uses the Boussinesq approximation for the Navier-Stokes equation

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla(p - \rho \boldsymbol{g}\boldsymbol{z} + \nu\nabla^2\boldsymbol{u} - \boldsymbol{g}\alpha(T - T_{ref})) \tag{1}$$

since it introduces errors on the order of 1% if $\Delta T < 15°C$ for air, a realistic temperature gradient for pollutant modeling. The Boussinesq approximation assumes constant fluid density. This approximation linearizes the acceleration term and enables the use of the continuity equation while retaining the effects of density in the momentum equation, such that the fluid satisfies conservation of mass, momentum, and energy.

Pollutant transport was modeled using a passive-scale transport equation. In the x-direction, it was formulated as

$$\frac{\partial \bar{C}}{\partial t} + \bar{u}_i\frac{\partial \bar{C}}{\partial x_i} = \frac{\partial}{\partial x_i}((D_t + \frac{\nu_t}{Sc_t})\frac{\partial \bar{C}}{\partial x_i}), \tag{2}$$

where $\bar{C}$ is the pollutant mass fraction [-], $\bar{u_i}$ is the mean fluid velocity in the x-direction [m/s], D_t is the pollutant mass diffusivity [m^2/s], and Sc_t is the turbulent Schmidt number [-]. This equation was implemented in multiple dimensions in OpenFOAM as

$$\frac{\partial C}{\partial t} + \nabla \cdot (\boldsymbol{\phi} C) - \nabla^2(D_{t,t}C) = 0 \tag{3}$$

```
fvScalarMatrix ConcEqn
(fvm::ddt(Conc)
+ fvm::div(phi, Conc)
- fvm::laplacian(DTT, Conc));
ConcEqn.solve();
```

by simplifying the coefficient on the right-hand side as the effective time-dependent turbulent mass diffusivity of the pollutant [m^2/s],

$$D_{t,t} = D_t + \frac{\nu_t}{Sc_t}, \tag{4}$$

```
volScalarField DTT ("DTT", DT + turbulence->nut()/Sct);
```

and moving everything to the left-hand side. $\boldsymbol{\phi}$ is the volumetric face-flux (flow through the cell faces) [m^3/s] and C is the pollutant concentration [kg/m^3].

This solver uses the K-epsilon turbulence model [10] to simulate mean flow characteristics. It describes turbulence using one equation for k, the turbulent kinetic energy [$\mathrm{m}^2/\mathrm{s}^2$],

$$k = \frac{u_\tau^2}{\sqrt{C_\mu}}, \tag{5}$$

and one for ε, the turbulent kinetic energy dissipation rate [$\mathrm{m}^2/\mathrm{s}^3$],

$$\varepsilon(z) = \frac{u_\tau^3}{\kappa(z + z_0)}, \tag{6}$$

where κ is the dimensionless von Karman constant (0.41), z is the height [m] at which the ground-normal streamwise flow speed profile, u [m/s], is calculated, C_μ is the dimensionless turbulent viscosity constant (0.09), and z_0 is the aerodynamic roughness length [m], which defines the boundary with the roughness sublayer. z_0 varies by landscape and is taken as 0.005 m, the accepted value for unobstructed flow on unvegetated land [22].

$$u_\tau = \frac{\sqrt{\tau_w}}{\rho} \tag{7}$$

is the friction, or shear, velocity [m/s], where

$$\tau_w = \frac{1}{2} C_f \rho U_\infty^2 \tag{8}$$

is the wall shear stress [N/m^2], where C_f is the dimensionless skin friction coefficient, which is a function of Re dependent on the problem geometry, and U_∞ is the freestream ("far away") velocity [m/s^2].

4 Simple Test Case

4.1 Boundary Conditions

A simple test case was built off the hotRoom example case for *buoyantBoussinesqPimpleFoam* to demonstrate the solver. In this case, pollutant concentrations were fixed at the top and bottom of the room at 2 $\mu\mathrm{g}/\mathrm{m}^3$ and 1 $\mu\mathrm{g}/\mathrm{m}^3$, respectively, with a constant air speed of 1 cm/s pointing downwards from ceiling. The complete boundary conditions are described in Table 1.

Table 1: Boundary conditions for simple test case.

	alphat	Conc	epsilon	k	nut	p	p_rgh	u
Floor	alphaJayatilleke WallFunction	fixedValue	epsilonWall Function	kqRWall Function	nutkWall Function	calculated	fixedFlux Pressure	noSlip
Ceiling	alphaJayatilleke WallFunction	fixedValue	epsilonWall Function	kqRWall Function	nutkWall Function	calculated	fixedFlux Pressure	noSlip
Walls	alphaJayatilleke WallFunction	zeroGradient	epsilonWall Function	kqRWall Function	nutkWall Function	calculated	fixedFlux Pressure	fixedValue

α_t is the turbulent thermal diffusivity [m^2/s], represented as alphat in OpenFOAM; Conc is the pollutant concentration [kg/m^3]; ε is represented as epsilon in OpenFOAM; ν_t is the turbulent viscosity [m^2/s], represented as nut in OpenFOAM; p is the static pressure [kg/ms2]; $p_r gh$ is the total hydrostatic pressure [kg/ms^2], represented as p_rgh in OpenFOAM; and T is the temperature [K].

zeroGradient applies a zero-gradient condition from the patch internal field onto the patch faces such that

$$\frac{\partial \phi}{\partial n} = 0. \tag{9}$$

fixedFluxPressure sets the pressure gradient to the provided value such that the flux on the boundary is that specified by the velocity boundary condition. noSlip fixes the velocity as zero at the walls.

4.1.1 Wall Functions

The remaining boundary conditions were described using wall functions. For turbulent flows, the first cell from the wall must be within a very thin viscous sublayer. This necessitates very fine near-wall meshing, increasing the computational time. Wall functions provide sparser near-wall meshes to accurately predict the velocity gradient across the boundary layer without necessitating very fine near-mesh resolution. Wall functions are valid in the fully turbulent zone ($30 > y^+ > 300$), shown in Figure 2. In this zone, the low-law, defined in Equation 10, holds.

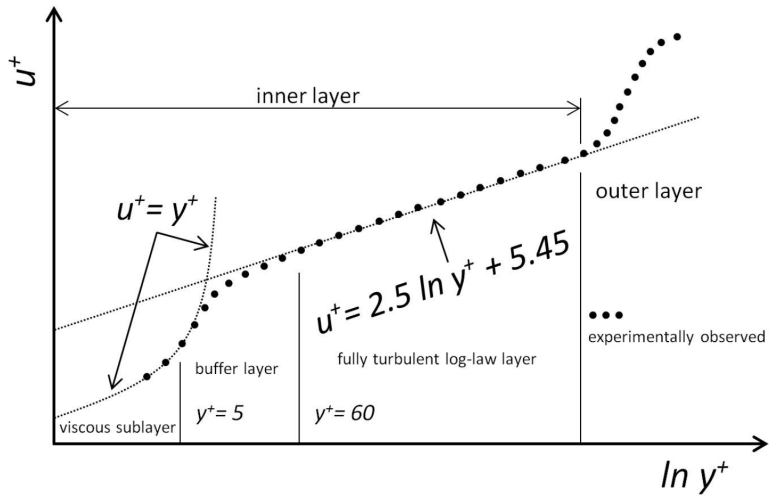


Figure 2: The Law of the Wall, showing the fully turbulent low-law zone [15].

$$u^+ = \frac{1}{\kappa} ln(y^+) + C, \tag{10}$$

where C is a constant, approximately 5.45 for smooth walls, and u^+ is the non-dimensional velocity at a non-dimensional distance y^+ parallel to the wall, such that

$$y^+ = \frac{yu_\tau}{\nu}, \tag{11}$$

where y is the absolute distance from wall [m].

α_t was represented by the alphatJayatillekeWallFunction boundary condition, which describes the wall using the Jayatilleke P-function, defined in Equation 12, which accounts for the resistance to heat transfer across the viscous sublayer [12].

$$P = 9.24(\beta^{\frac{3}{4}} - 1)(1 + 0.28e^{-0.007\beta}), \tag{12}$$

where P is the P-function [-] and $\beta = \frac{Sc_l}{Sc_t}$ is the ratio of the laminar and turbulent Schmidt numbers. It follows that the dimensionless near-wall temperature (T^+) is

$$T^+ = Sc_t(u^+ + P). \tag{13}$$

ε was represented by an epsilonWallFunction boundary condition, which provides a wall constraint on ε for low- and high-Re turbulence models. Applying the stepwise switch (discontinuous) method to blend the ε predictions for the viscous and inertial sublayers, if $y^+ < y^+_{lam}$, then $\varepsilon = \varepsilon_{vis}$, and if if $y^+ \geq y^+_{lam}$, then $\varepsilon = \varepsilon_{log}$. ε_{vis} is ε computed by the viscous sublayer assumptions [$\mathrm{m^2/s^3}$], defined as

$$\varepsilon_{vis} = 2wk\frac{\nu_w}{y^2}, \tag{14}$$

where w is the cell-corner weights [-], k is the turbulent kinetic energy [$\mathrm{m^2/s^2}$], ν_w is the kinematic viscosity of the fluid near the wall [$\mathrm{m^2/s}$], and y is the wall-normal distance [m]. ε_{log} is ε computed by the inertial sublayer assumptions [$\mathrm{m^2/s^3}$], defined as

$$\varepsilon_{log} = wC_\mu\frac{k^{\frac{3}{2}}}{\nu_{t_w}y}, \tag{15}$$

where C_μ is the empirical model constant [-] and ν_{t_w} is the turbulent viscosity near the wall [$\mathrm{m^2/s}$].

k was represented by the kqRWallFunction boundary condition, which provides a simple wrapper around the zero-gradient condition for the cases of high Re (turbulent) flow using wall functions.

ν_t was represented by the nutkWallFunction boundary condition, which provides a wall constraint on ν_t based on k for low- and high-Re turbulence models, expressed as

$$\nu_t = f_{blend}(\nu_{t_{vis}}, \nu_{t_{vis}}) \tag{16}$$

with

$$\nu_{t_{vis}} = 0 \tag{17}$$

$$\nu_{t_{log}} = \nu_w(\frac{y^+\kappa}{ln(Ey^+)} - 1) \tag{18}$$

$$y^+ = C_\mu^{\frac{1}{4}} y \frac{\sqrt{k}}{\nu_w} \tag{19}$$

where f_{blend} is a wall-function blending operator between the viscous and inertial sublayer contributions, $\nu_{t_{vis}}$ is ν_t computed by the viscous sublayer assumptions [m^2/s], $\nu_{t_{log}}$ is ν_t computed by the inertial sublayer assumptions [m^2/s], and E is the wall roughness parameter [-].

4.2 Schemes

The temporal scheme (ddtSchemes) was a Euler implicit time scheme

$$\frac{\partial}{\partial t}(\phi) = \frac{\phi - \phi^0}{\Delta t}. \tag{20}$$

For the spatial schemes, the gradient scheme (gradSchemes) was least-squares, which calculates the cell gradient using least squares.

All divergence schemes (divSchemes) were Gauss upwind, except for div((nuEff*dev2(T(grad(U))))), which was Gauss linear. Gauss upwind, defined in Equation 21, is first order and bounded. It sets the face value according to the upstream value and is equivalent to assuming that the cell values are isotropic (same in all directions) with a value that represents the average value. Its normalized variable diagram is shown in Figure 3.

$$\phi_f = \phi_c, \tag{21}$$

where ϕ_f is the face value and ϕ_c is the upstream value.

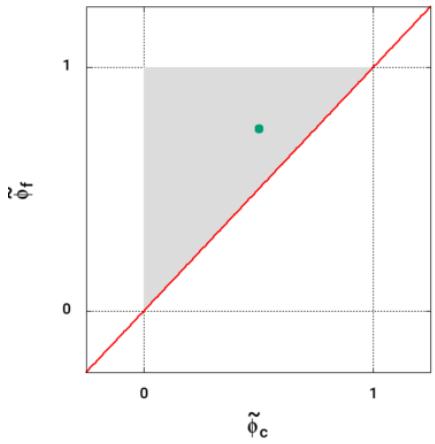


Figure 3: Normalized variable diagram for upwind divergence scheme [20].

Gauss linear, defined in Equation 22, is second order and unbounded. It is often used for isotropic meshes due to low dissipation. Its normalized variable diagram is shown in Figure 4.

$$\phi_f = 0.5(\phi_c + \phi_d), \tag{22}$$

where ϕ_d is the downstream value.

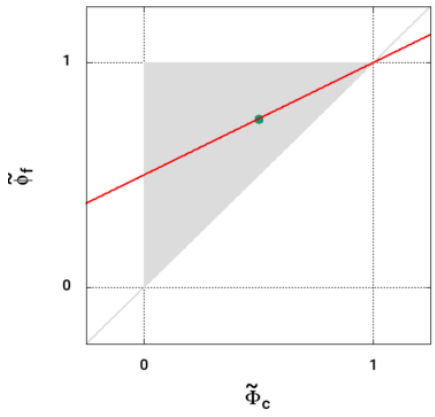


Figure 4: Normalized variable diagram for upwind divergence scheme [11].

The Laplacian scheme (laplacianSchemes) was Gauss linear corrected, which is unbounded, second order, and conservative. The interpolation scheme (interpolationSchemes) was linear (central differencing). The surface-normal gradient

scheme (snGradSchemes) was corrected, an explicit central-difference scheme with non-orthogonal correction.

4.3 Computational Grid

The geometry was meshed in SALOME [17] using a coarse, uniform grid, as shown in Figure 5, to reduce computational intensity and simulation runtime.

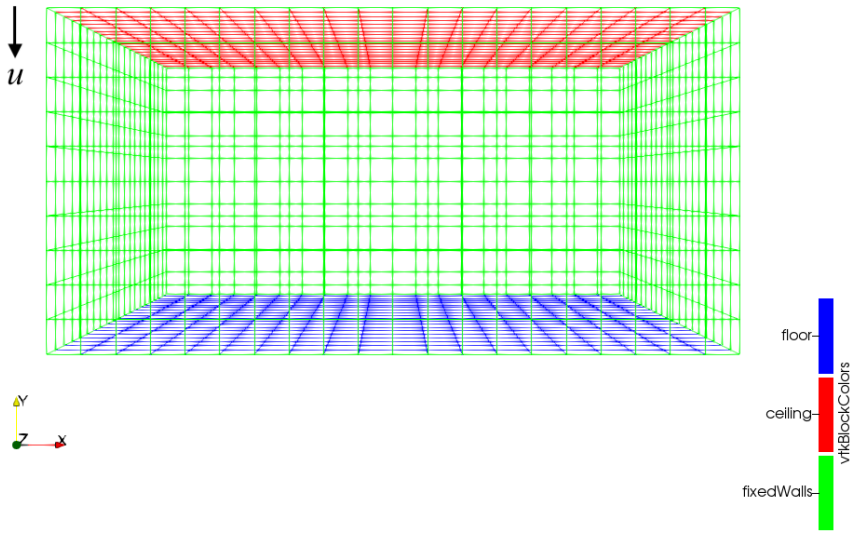
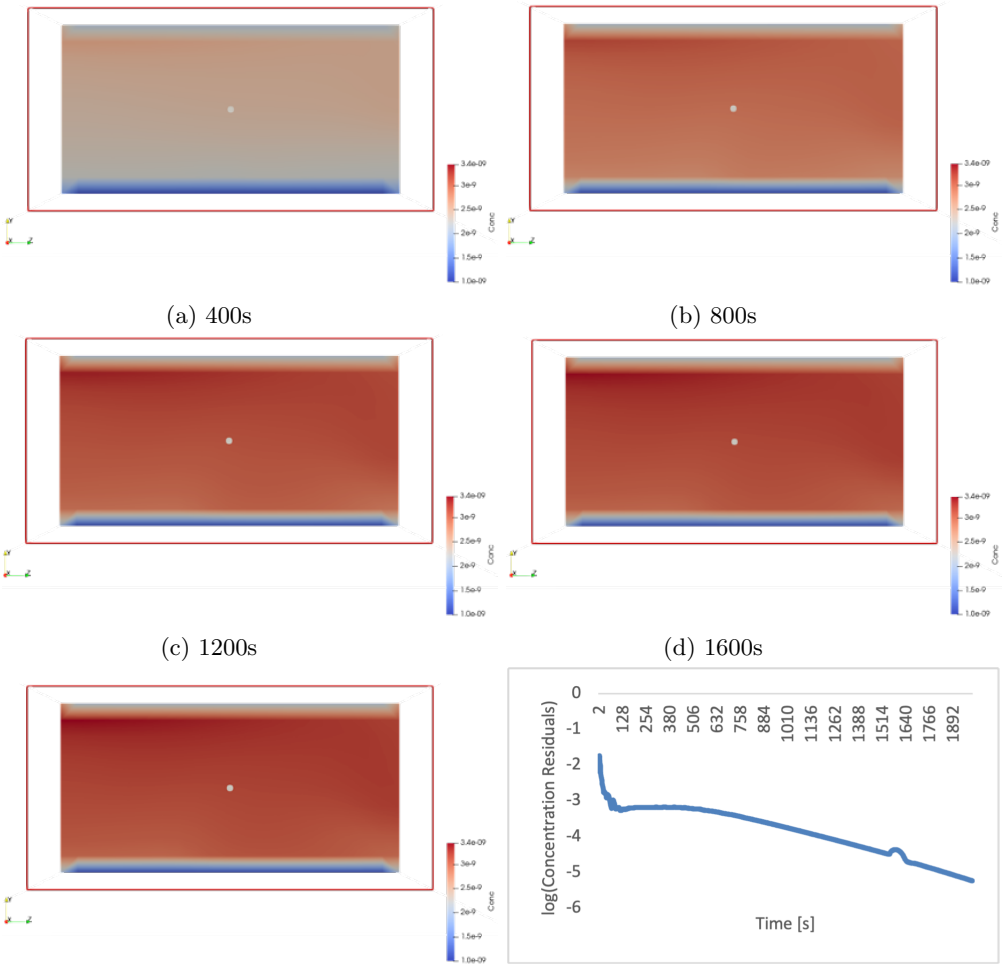


Figure 5: Meshed box in Paraview.

4.4 Results

The results of the simple test case are as shown in Figure 6. Convergence is demonstrated by the residuals approaching zero as time increases. The simulation was stopped once the concentration residuals were less than 10^{-5}. Only the residuals for concentration are shown in Figure 6, but similar results were achieved for the other variables.



(a) 400s (b) 800s

(c) 1200s (d) 1600s

(e) 2000s (f) Semi-log plot of residuals

Figure 6: Test case results after 2000s, including residuals.

5 Discussion and Future Work

Future work should include validating this solver against more established software, such as Ansys Fluent, or experimental results. This solver should then be applied to relevant air pollution scenarios, such as vehicle exhaust, similar to [8], cigarette smoke, or COVID-19 transmission, similar to [13]. While there are many potential applications for this solver, it is important to note that CFD simulations cannot be performed in all meteorological conditions since RANS simulations with passive pollutant transport cannot capture the predicted plume rise caused by thermal instability.

References

[1] *Air pollution*. en. URL: https://www.who.int/westernpacific/health-topics/air-pollution (visited on 12/09/2020).

[2] Bert Blocken. "Computational Fluid Dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations". en. In: *Building and Environment*. Fifty Year Anniversary for Building and Environment 91 (Sept. 2015), pp. 219–245. ISSN: 0360-1323. DOI: 10.1016/j.buildenv.2015.02.015. URL: https://www.sciencedirect.com/science/article/pii/S0360132315000724 (visited on 04/27/2022).

[3] *blueCFD-Core*. 2022. URL: https://bluecfd.github.io/Core/ (visited on 05/09/2022).

[4] L. S. Caretto et al. "Two calculation procedures for steady, three-dimensional flows with recirculation". en. In: *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*. Ed. by Henri Cabannes and Roger Temam. Lecture Notes in Physics. Berlin, Heidelberg: Springer, 1973, pp. 60–68. ISBN: 978-3-540-38392-5. DOI: 10.1007/BFb0112677.

[5] Lup Wai Chew and Leslie K. Norford. "Pedestrian-level wind speed enhancement in urban street canyons with void decks". en. In: *Building and Environment* 146 (Dec. 2018), pp. 64–76. ISSN: 03601323. DOI: 10.1016/j.buildenv.2018.09.039. URL: https://linkinghub.elsevier.com/retrieve/pii/S0360132318305973 (visited on 02/26/2022).

[6] AJ Cimorelli et al. *AERMOD: description of model formulation, US Environmental Protection Agency*. Tech. rep. EPA-454/R-03-004, 2004.

[7] Daniel Elfverson and Christian Lejon. "Use and Scalability of OpenFOAM for Wind Fields and Pollution Dispersion with Building- and Ground-Resolving Topography". en. In: *Atmosphere* 12.9 (Sept. 2021). Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, p. 1124. ISSN: 2073-4433. DOI: 10.3390/atmos12091124. URL: https://www.mdpi.com/2073-4433/12/9/1124 (visited on 02/12/2022).

[8] Yuhan Huang et al. "A review of strategies for mitigating roadside air pollution in urban street canyons". en. In: *Environmental Pollution* 280 (July 2021), p. 116971. ISSN: 0269-7491. DOI: 10.1016/j.envpol.2021.116971. URL: https://www.sciencedirect.com/science/article/pii/S0269749121005534 (visited on 04/27/2022).

[9] R. I Issa. "Solution of the implicitly discretised fluid flow equations by operator-splitting". en. In: *Journal of Computational Physics* 62.1 (Jan. 1986), pp. 40–65. ISSN: 0021-9991. DOI: 10.1016/0021-9991(86)90099-9. URL: https://www.sciencedirect.com/science/article/pii/0021999186900999 (visited on 02/26/2022).

[10] B. E. Launder and D. B. Spalding. "The numerical computation of turbulent flows". en. In: *Computer Methods in Applied Mechanics and Engineering* 3.2 (Mar. 1974), pp. 269–289. ISSN: 0045-7825. DOI: 10.1016/0045-7825(74)90029-2. URL: https://www.sciencedirect.com/science/article/pii/0045782574900292 (visited on 05/05/2022).

[11] *Linear divergence scheme*. 2017. URL: https://www.openfoam.com/documentation/guides/latest/doc/guide-schemes-divergence-linear.html (visited on 04/29/2022).

[12] M.R. Malin. "On the calculation of heat transfer rates in fully turbulent wall flows". en. In: *Applied Mathematical Modelling* 11.4 (Aug. 1987), pp. 281–284. ISSN: 0307904X. DOI: 10.1016/0307-904X(87)90143-0. URL: https://linkinghub.elsevier.com/retrieve/pii/0307904X87901430 (visited on 04/19/2022).

[13] Mariam et al. "CFD Simulation of the Airborne Transmission of COVID-19 Vectors Emitted during Respiratory Mechanisms: Revisiting the Concept of Safe Distance". In: *ACS Omega* 6.26 (July 2021). Publisher: American Chemical Society, pp. 16876–16889. DOI: 10.1021/acsomega.1c01489. URL: https://doi.org/10.1021/acsomega.1c01489 (visited on 05/13/2022).

[14] Helmut Mayer. "Air pollution in cities". en. In: *Atmospheric Environment* 33.24 (Oct. 1999), pp. 4029–4037. ISSN: 1352-2310. DOI: 10.1016/S1352-2310(99)00144-2. URL: https://www.sciencedirect.com/science/article/pii/S1352231099001442 (visited on 04/27/2022).

[15] Dhruv Mehta et al. "A Wall Boundary Condition for the Simulation of a Turbulent Non-Newtonian Domestic Slurry in Pipes". en. In: *Water* 10.2 (Jan. 2018), p. 124. ISSN: 2073-4441. DOI: 10.3390/w10020124. URL: http://www.mdpi.com/2073-4441/10/2/124 (visited on 04/18/2022).

[16] Keigo Nakajima, Ryozo Ooka, and Hideki Kikumoto. "Evaluation of k-Reynolds stress modeling in an idealized urban canyon using LES". en. In: *Journal of Wind Engineering and Industrial Aerodynamics* 175 (Apr. 2018), pp. 213–228. ISSN: 0167-6105. DOI: 10.1016/j.jweia.2018.01.034. URL: https://www.sciencedirect.com/science/article/pii/S0167610517302623 (visited on 02/26/2022).

[17] *Salome Platform*. en-GB. 2022. URL: https://www.salome-platform.org/ (visited on 05/02/2022).

[18] Alec Tauer. "CFD Modeling of Aerial Dispersion of Pollutants in Urban Environments". en. PhD thesis. Milwaukee, Wisconsin: Marquette University, May 2021. URL: https://epublications.marquette.edu/cgi/viewcontent.cgi?article=1660&context=theses_open.

[19] Y. Toparlar et al. "A review on the CFD analysis of urban microclimate". en. In: *Renewable and Sustainable Energy Reviews* 80 (Dec. 2017), pp. 1613–1640. ISSN: 1364-0321. DOI: 10.1016/j.rser.2017.05.248. URL: https://www.sciencedirect.com/science/article/pii/S1364032117308924 (visited on 02/26/2022).

[20] *Upwind divergence scheme*. 2017. URL: https://www.openfoam.com/documentation/guides/latest/doc/guide-schemes-divergence-upwind.html (visited on 04/29/2022).

[21] H. G. Weller et al. "A tensorial approach to computational continuum mechanics using object-oriented techniques". en. In: *Computers in Physics* 12.6 (1998), p. 620. ISSN: 08941866. DOI: 10.1063/1.168744. URL: http://scitation.aip.org/content/aip/journal/cip/12/6/10.1063/1.168744 (visited on 02/26/2022).

[22] World Meteorological Organization. *Guide to Meteorological Instruments and Methods of Observation (WMO-No. 8)*. en. Seventh edition. OCLC: 928941505. Geneva: World Meteorological Organization, 2008. ISBN: 978-92-63-10008-5. URL: https://www.posmet.ufv.br/wp-content/uploads/2016/09/MET-474-WMO-Guide.pdf.

[23] M.H. Zheng et al. "Coupling GIS with CFD modeling to simulate urban pollutant dispersion". In: *2010 International Conference on Mechanic Automation and Control Engineering*. June 2010, pp. 1785–1788. DOI: 10.1109/MACE.2010.5536018.