university of groningen

faculty of science and engineering

Simulation of airflow through a window with an anti-bug grid

Final Project
Course: Modelling and Simulation

November 2024

Authors:
BSc. Martin Opat (s4704126)

BSc. Robin Sachsenweger Ballantyne (s4617096)

Contents

1 Introduction

Figure 1.1: *Bug grid (Flickr.com)*

Bug grids can be placed over open windows to allow air to flow through without letting bugs into someone's house. Presumably, when people open their windows, they want air to flow through their house. However, the installation of the grid decreases the functional cross-sectional area of the window, which might decrease the airflow. If the properties of the grid cause the flow to become turbulent, the reduction in airflow would be more drastic, than due to the reduction in the cross-sectional area alone. Using a fluid simulation, this report aims to capture this phenomenon, and investigate how the grid's thickness and density affect the airflow through a window.

To achieve this goal, a numerical simulation of airflow based on the Navier-Stokes equations will be implemented in Python. The numerical simulation will include discretising both space and time. The time discretisation will be done using the Euler method , and the space discretisation will be done using the finite difference method [1].

In doing so, an investigation into the optimal parameters of the grid that maximise airflow while "keeping the bugs out" will be conducted. Additionally, the applicability and flexibility of fluid simulations in modelling real-world phenomena will be demonstrated. Namely, the numerical stability, and the computational efficiency of the simulation model will be discussed.

1.1 Research question

Main:

- How much does the air flux through a window with a bug grid change with the grid's thickness and density?

Subquestions:

- What are the optimal parameters of the grid that maximise airflow while keeping the bugs out?
- In which section of the window has the airflow decreased the most due to the presence of the bug grid?

2 Model and Methodology

In this report, we implement a numerical simulation of airflow based on Navier-Stokes equations. The choice of the implementation language is Python for its simplicity, and the availability of libraries such as NumPy, Matplotlib, and Py-PDE.

2.1 Model

We simulated a simplistic model of a compressible fluid flow in a rectangular box of size $X \times Y \times Z,\ X > Y, Z$. The window was located at the center of the box, and was aligned to be parallel with its short sides (the ones within the Y-Z plane).

The fluid inside the box was driven by a source term F emulating a pressure-driven channel flow. The values of the source term was drawn at each time step from a normal distribution with a constant positive mean and a small standard deviation, in order to simulate microscopic fluctuations in the flow. The source term was applied to the entire fluid domain, but not to the boundary cells.

The physical fields were used in the simulation to represent the fluid. A vector field of the fluid velocity $\mathbf{u}(x, y, z, t)$, and a scalar field of the density $\rho(x, y, z, t)$. The velocity field was initialized with a constant value of zero everywhere, while the density field was randomly drawn from a normal distribution with a constant mean and a small standard deviation. A periodic boundary conditions:

$$\mathbf{v}(0, y, z, t) = \mathbf{v}(X, y, z, t), \tag{2.1}$$
$$\rho(0, y, z, t) = \rho(X, y, z, t) \tag{2.2}$$

was applied to the short side parallel with the window, while no-slip boundary conditions:

$$\forall x_b, y_b, z_b \in [\text{long sides},\ \text{window}] :\ \mathbf{v}(x_b, y_b, z_b, t) = \mathbf{0} \tag{2.3}$$
$$\forall x_b, y_b, z_b \in [\text{long sides},\ \text{window}] :\ \rho(x_b, y_b, z_b, t) = 0 \tag{2.4}$$
$$\tag{2.5}$$

were applied to the window and the other (long) sides. Additionally, the Neumann boundary condition:

$$\forall x_b, y_b, z_b \in [\text{long sides}] :\ \partial_n \mathbf{v}(x_b, y_b, z_b, t) = 0 \tag{2.6}$$
$$\forall x_b, y_b, z_b \in [\text{long sides}] :\ \partial_n \rho(x_b, y_b, z_b, t) = 0 \tag{2.7}$$

was also applied to the long sides of the box. ∂_n denotes the partial derivative in outward normal direction.

The following list of physical assumptions were made on the fluid flow:

1. Mass conservation
2. Isotropy (i.e., no gravity)
3. Compressible fluid
4. Newtonian fluid

The assumptions listed above yield the Navier-Stokes equation(s) [2] in the following form:

$$\left(\partial_t + \mathbf{u} \cdot \nabla - \frac{\mu}{\rho} \nabla^2 - \left(\frac{\mu}{3\rho} + \frac{\zeta}{\rho}\right) \nabla(\nabla \cdot)\right) \mathbf{u} = -\frac{1}{\rho} \nabla p + \mathbf{f}, \tag{2.8}$$

where ρ is the mass density, $\mathbf{u}$ is velocity, p is the pressure, μ is a dynamic viscosity, ζ is the bulk viscosity, and $\mathbf{f}$ is the source term. Further "massaging" the equation:

$$\partial_t \mathbf{u} + \underbrace{(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{\mu}{\rho}\nabla^2\mathbf{u} - \left(\frac{\mu}{3\rho} + \frac{\zeta}{\rho}\right)\nabla(\nabla\cdot\mathbf{u})}_{f(\mathbf{u})} = -\frac{1}{\rho}\nabla p + \mathbf{f}, \tag{2.9}$$

where $f(\mathbf{u})$ is a function of $\mathbf{u}$ and its spatial (partial) derivatives.

$$\partial_t \mathbf{u} = -f(\mathbf{u}) - \frac{1}{\rho}\nabla p + \mathbf{f} \tag{2.10}$$

Applying the finite difference method [1] to the time derivative in eq. 2.10, we get:

$$\frac{\mathbf{u}_{ijk}^{n+1} - \mathbf{u}_{ijk}^{n}}{\Delta t} = -f(\mathbf{u}^n) - \frac{1}{\rho}\nabla p + \mathbf{f} \tag{2.11}$$

$$\mathbf{u}_{ijk}^{n+1} = \mathbf{u}_{ijk}^{n} - \Delta t\left(f(\mathbf{u}^n) + \frac{1}{\rho}\nabla p + \mathbf{f}\right), \tag{2.12}$$

where Δt is the discrete time-step size, n is the time-step index, and $i, j,$ and k are the indices in the $x, y,$ and z spatial coordinates respectively. We can apply the finite difference method to all the (spatial) derivatives in f in the same way as we did above. Final, to relate the density field to the pressure of the fluid, we use the equation of state for an ideal gas [3]:

$$p = \rho R T. \tag{2.13}$$

For time integration, the forward Euler method was used [1]. The motivation of this choice was the computational simplicity of the method. The forward Euler method is a first-order method, and is known to be numerically unstable [1] for large values of the time-step size. This limitation was counteracted by using an adaptive time-step size, so that the time-step size was reduced when the simulation was close to a critical point.

The above model captures the essential physics of a flow of compressible fluid in a rectangular box. Since, the fluid is compressible, the model can be used to simulate a wide range of fluid flows, e.g., linear, turbulent, and chaotic flows. However, there are limitations to the model, following from the assumptions listed above. No sinks or sources of mass were considered in the model, and the fluid was assumed to Newtonian, meaning that its viscosity depends only on the pressure, and not the shear rate. Note, that the temperature was assumed to remain constant. Similarly, the fluid was assumed to be isotropic, meaning that the flow assumes light particles such that the effect of gravity is negligible. Since the simulation fluid is air, the above assumptions are reasonable.

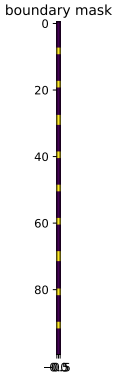
3 Simulation setup

4 Results

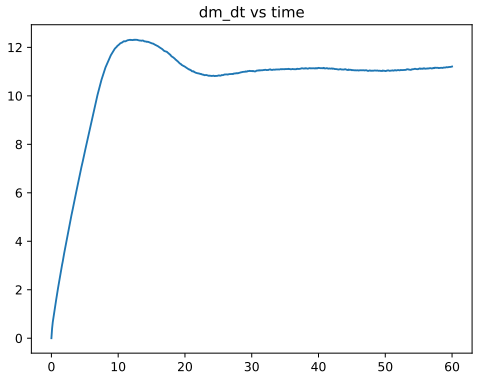
The airflow Q through a cross-sectional area A can be mathematically defined as:

$$Q = \int_A \rho \mathbf{v} \cdot d\mathbf{A} = \frac{dm}{dt} \tag{4.1}$$

The airflow Q was periodically calculated and stored over the duration of the simulation.

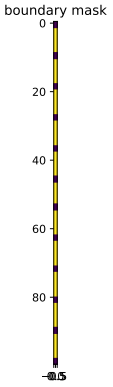


(a) *Example of a boundary mask representing a window with a bug grid.*

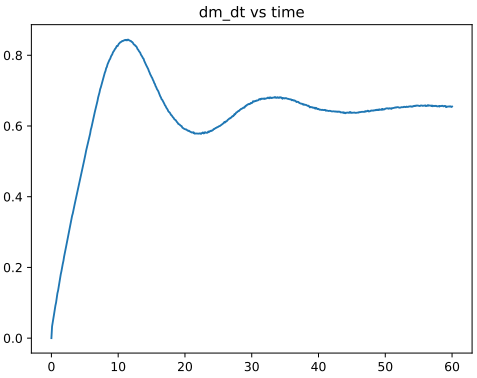


(b) *The plot of airflow Q against time, hole count = 10, hole width = 0.4.*

Figure 4.1: *Comparison of airflow rates through the cross-sectional area of the window for different grid hole sizes.*

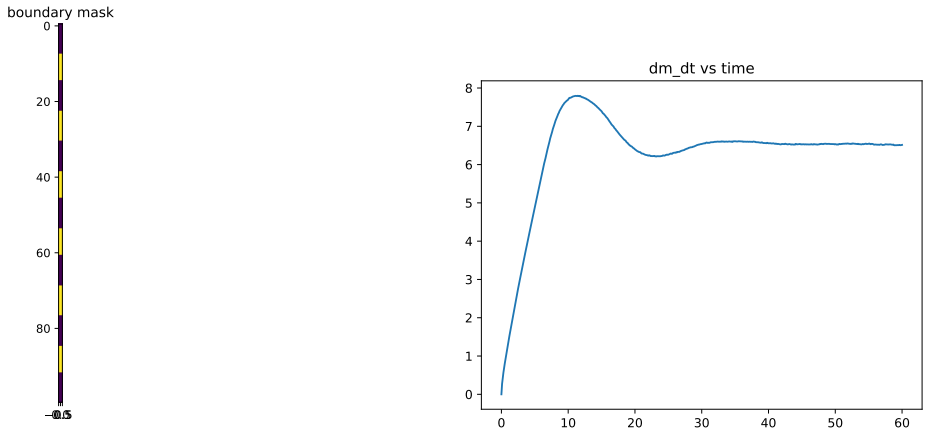


(a) *Example of a boundary mask representing a window with a bug grid.*



(b) *The plot of airflow Q against time, hole count = 12, hole width = 0.1.*

Figure 4.2: *Comparison of airflow rates through the cross-sectional area of the window for different grid hole sizes.*



(a) *Example of a boundary mask representing a window with a bug grid.*

(b) *The plot of airflow Q against time, hole count = 7, hole width = 0.4.*

Figure 4.3: *Comparison of airflow rates through the cross-sectional area of the window for different grid hole sizes.*

From Eq. (4.1), the total mass m of air that flows through the area in time T is given by:

$$m = \int_0^T Q \, dt \tag{4.2}$$

The plot of m against the number of holes in the grid was plotted in Figure 4.4 for four different hole sizes.

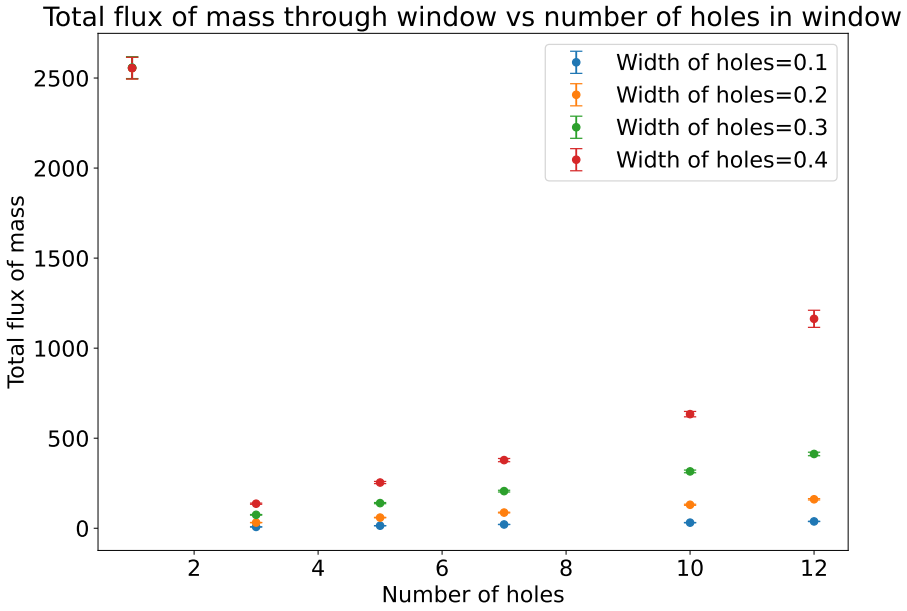


Figure 4.4: *Total mass m through the cross-sectional area of the window vs. the number of holes in the bug grid, plotted for different grid hole sizes.*

In the left-upper corner of Figure 4.4, the data point corresponding to an empty window is present. Note that this value was constantly the same for all widths of the hole, regardless of its colour in the plot. We can also see, that the airflow through the window increases

both with the number of holes and the size of the holes.

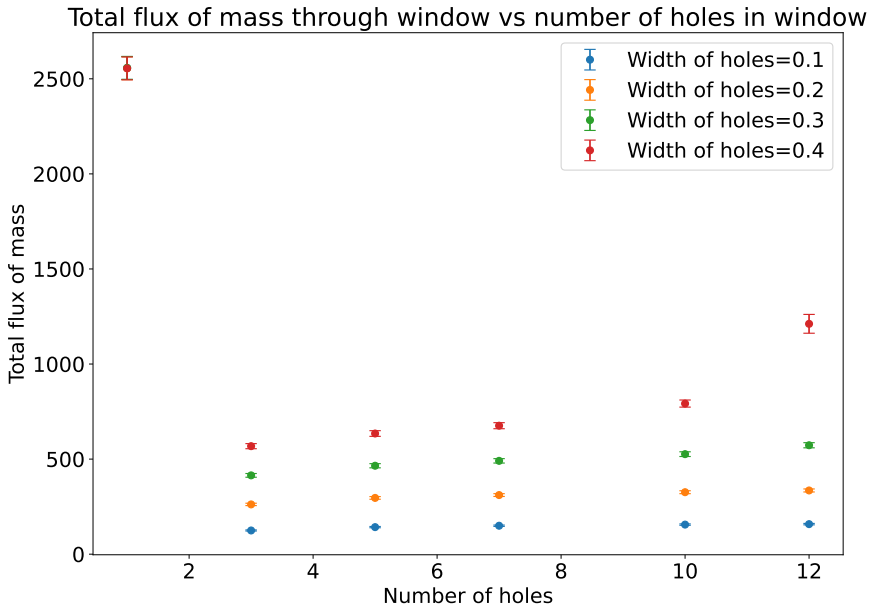


Figure 4.5: *Normalized total mass m through the cross-sectional area of the window vs. the number of holes in the bug grid, plotted for different grid hole sizes.*

Figure 4.5 displays the same data as Figure 4.4, but the airflow is normalized to the actual effective area of the window. We can see that the normalized airflow still increases with the size of the holes, however, the number of holes no longer has a significant impact.

5 Discussion

In this section, we discuss the results described in Section 4, their implications and causes in terms of the limitations of the implemented simulation. We also discuss the choices made in implementing the simulation model, as well as possible improvements.

Using the plots in Figure **??** and Figure **??**, we can qualitatively verify that the fluid simulation is running as expected. More of these plots for other sets of parameters can be found in Appendix A.

To analyze the results quantitatively as well, two physical quantities were derived, implemented, and tracked over the duration of the simulation: the airflow through the window Q and the total mass m that flew through the window in a given time.
The airflow Q is plotted against time in Figures 4.1, 4.2, and 4.3 on the right, with the corresponding bug grid boundary on the left. We can see an initial linear increase in airflow in all three cases. The air in the simulation starts at rest, so this initial linear trend is justified since the flow speed keeps increasing until reaching an equilibrium with the source term. An interesting observation is the appearance of damped oscillations in the transition between the initial (linearly increasing) flow and the final (equilibrium) flow. The interplay between the momentum of the fluid and the pressure gradients can explain these oscillations, and the fluid's viscosity explains the damping.
The total mass m is displayed in Figure 4.4 for all tested parameter values. This plot displays the relation between the airflow through the window and the grid parameters. One can immediately observe a severe reduction in the airflow caused by the presence of the grid. It is, however, difficult to read from this plot whether this reduction is merely proportional to the area of the window that is blocked by the grid. This is why the plot in Figure 4.5 showing the normalized airflow divides the "absolute" airflow measured through the window by the percentage of the window that is open, i.e., not blocked by the grid. This normalization allows for a more direct comparison between the different grid sizes, but most importantly, it allows for a comparison between the window with and without a grid. If the grid were only to block the amount of airflow proportional to the window area it covers, the normalized airflow would be constant for all grid sizes. However, looking at Figure 4.5, we can see that even the normalized airflow is significantly decreased by the presence of the grid. This result implies the presence of the grid introduces additional resistance and turbulence to the airflow.

While the obtained results discussed above nicely show the effect of the grid on the airflow, there are several limitations to the simulation model that need to be addressed. The most significant limitation, and the motivation behind many of the choices made in the implementation, is the computational complexity of the simulation, combined with a lack of computational resources. The most drastic decision made to mitigate this limitation was to switch from a 2D to a 3D simulation. A 2D simulation is significantly less computationally expensive, however, with a 2D simulation one is taking the risk of possibly missing out on important 3D effects. If it were these 3D effects that were the main cause of the reduction of the normalized airflow, then the results obtained from the 2D simulation would not be significant. Luckily, as we discussed above, an unpropotional reduction in airflow was also observed in the 2D simulation, implying that any solely 3D effects that were missed out are negligible w.r.t. the qualitative goals of this study. Of course, it is worth to note that the 3D effects would still be significant in any quantitative measurements.

Another issue experienced during the implementation was the numerical instability of the simulation. The used time integration scheme was a simple forward Euler scheme, which is known to be unstable for the Navier-Stokes equations, as was already discussed in Section 2. One implication of these instabilities was already mentioned in Section 3 - chequerboard oscillations. These oscillations are a result of the discretization of the fields on the grid, and are a common issue in fluid simulations citeharlow1965numerical. A computationally cheap way to mitigate these oscillations is to use artificial viscosity, which was implemented in the simulation. A better, but more computationally expensive, way to mitigate these oscillations would be to use a higher-order time integration scheme, such as the Runge-Kutta 4 scheme [4]. Another observed issue related to the numerical instability was the presence of high frequency noise and large gradients in the fields. Some simpe signal processing techniques were attempted to smooth out the fields, but the results were not satisfactory. Instead, the combination of the artificial viscosity and increased grid resolution that became possible due to the switch to 2D was used to resolve this issue.

One might pose the question, if it is the Navier-Stokes equations that are causing the numerical instability, why not use a different set of equations, such as the Euler equations? The Euler equations are a simplified version of the Navier-Stokes equations, where the viscosity term is neglected **eulerEqs**. This simplification could indeed make the simulation more stable, but it would also make it less accurate. The viscosity term in the Navier-Stokes equations is crucial for the simulation of the airflow through the window as the viscosity is responsible for the reduction in airflow caused by the grid, since it is the viscosity that causes the resistance and turbulence in the flow. Another choice worth discussing is the decision to simulate the flow as a compressible fluid. An argument could be made that at low speed, such as the ones usually experienced in everyday life, air can be for the most part considered incompressible. This would simplify the simulation, as the density would be constant and the simulation would be less computationally expensive. However, this change would not necessarily make the simulation more numerically stable or accurate, and it is even debatable whether this approach would less computationally expensive, as in many cases a Poisson equation for the pressure would need to be solved citeincompressible. For those reasons, the decision was made to implement a compressible fluid simulation, as described in Sections 2 and 3.

Additionally, the correct choice of boundary conditions is crucial for the stability and accuracy of the simulation. Here by correct, such boundary conditions are meant that they are computationally stable while still being physically meaningful.

6 Conclusion

In this report we described the mathematical underpinnings of a numerical model for compressible fluid flow simulation. The physical constraints, initial and boundary conditions were explained. The model was implemented in Python using the finite difference method for space discretization, and forward Euler method for time discretization. The Py-PDE Python library was used to perform the numerical integration, and Just-In-Time compilation into C was performed to optimize the implementation.

7 References

[1] R. H. Landau and C. C. Bordeianu, *Computational Physics: Problem Solving with Python*. John Wiley & Sons, 2015, pp. 86–87, 177–178.

[2] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge university press, 2000, pp. 131–170.

[3] S. J. Blundell and K. M. Blundell, *Concepts in Thermal Physics*. Oxford University Press, Oct. 2009, pp. 6–7, ISBN: 9780199562091. DOI: 10.1093/acprof:oso/9780199562091.001.0001.

[4] W. H. Press, *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007, pp. 907–915, ISBN: 9780521880688.

Appendix A

Implemented visualization techniques