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

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.

Figure 1.1: *Anti-bug grid (Flickr.com)*

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:

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

Then, 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 airflow Q was periodically calculated and stored over the duration of the simulation. The plot of Q against the number of holes in the grid was plotted in Figure 4.1 for four different hole sizes.

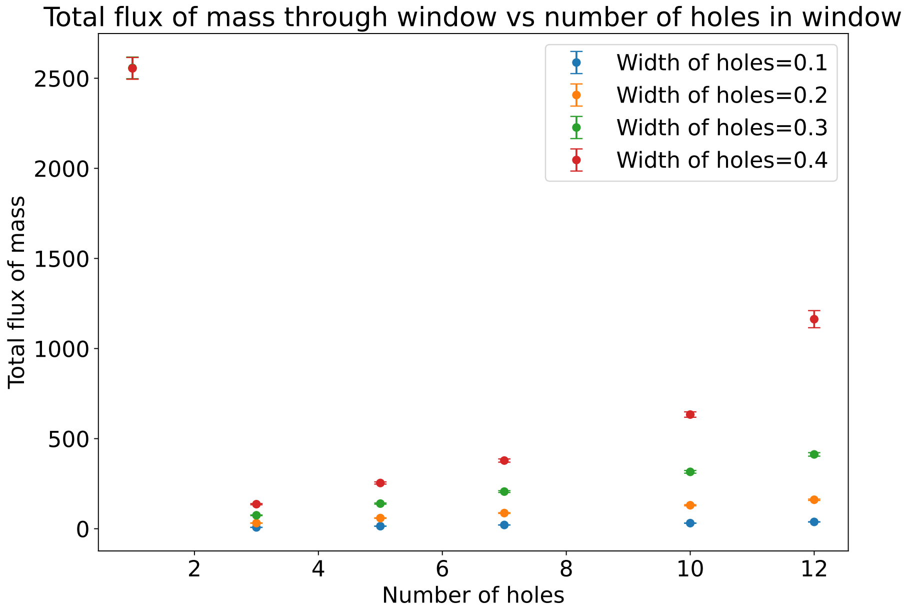

Figure 4.1: *Airflow rate through the cross-sectional area of the window vs. the number of holes in the bug grid, plotted for different grid hole sizes.*

In the left-upper corner of Figure 4.1, 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.

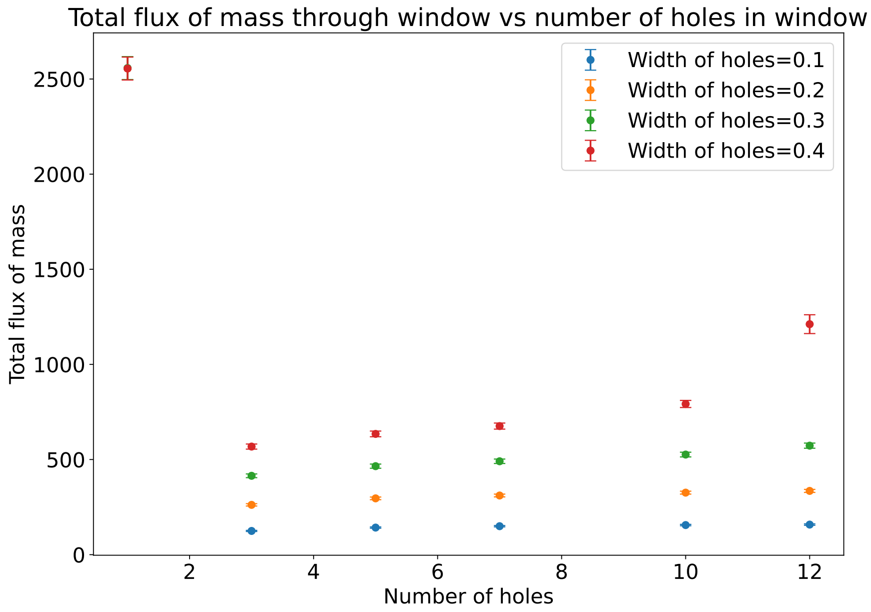

Figure 4.2: *Normalized airflow rate through the cross-sectional area of the window vs. the number of holes in the bug grid, plotted for different grid hole sizes.*

Figure 4.2 displays the same data as Figure 4.1, 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

The plot in Figure 4.1 displays the expected 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-off from this plot whether this reduction merely proportional to the area of the window that is blocked by the grid. This is why the plot in Figure 4.2 showing the normalized airflow divides the "absolute" airflow through measured through the window by the percentage of the window that is actually 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 area of the window it covers, the normalized airflow would be constant for all grid sizes. However, looking at Figure 4.2, 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.

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.

Appendix A

Implemented visualization techniques