INTERNATIONAL BUILDING SIMULATION ASSOCIATION ENGLAND

Proceedings of BSO 2018:
4th Building Simulation and Optimization Conference, Cambridge, UK: 11-12 September 2018

Modelling Computational Fluid Dynamics with Swarm Behaviour

Ljubomir Jankovic
Zero Carbon Lab, School of Creative Arts, University of Hertfordshire
College Lane, Hatfield, AL10 9AB, UK.
L.Jankovic@herts.ac.uk

Abstract

The paper looks at replacing the current top-down approach to modelling, predominantly used in dynamic simulation tools, with a nature inspired bottom-up approach based on principles of swarming. Computational fluid dynamics (CFD) is chosen for this research, as one of the most time-consuming processes under the traditional simulation approach. Generally based on Navier-Stokes simultaneous differential equations, CFD requires considerable user preparation time and considerable CPU execution time. The main reason is that the top-down equations represent the system as a whole and generate a large solution space, requiring a solver to find a solution. However, air and building materials do not have cognitive capabilities to solve systems of equations in order to 'know' how to transfer heat. Instead, heat transfer occurs through proximity interaction between molecules, leading to self-organised behaviour that is much faster than the behaviour modelled by the top-down systems of equations. The paper investigates how the bottom-up approach using the principles of swarming could improve the speed and interactivity of CFD simulation.

Introduction

Swarming is a term that describes collective behaviour of animals, characterised by self-organising patterns of astonishing complexity. Yet there is evidence that this complexity arises from the simplest of local rules applied on an individual level (Reynolds 1987). Swarming is used here as a generic term, which is named differently when applied to group formations of specific living species: flocking refers to swarming formations of birds, schooling or shoaling refers to formations of fish, herding refers to formations of four-legged animals, and swarm behaviour of insects is simply referred to as swarming.

The research aim of this paper is to investigate whether swarming principles can be used as a starting point for bottom-up simulation of computational fluid dynamics (CFD) by developing a proof of concept model.

The specific research objectives are:

1. To develop particle swarming model using the rules of separation, alignment, and cohesion.
2. To eliminate conscious actions associated with swarming rules as the first step towards CFD modelling.
3. To implement principles of physics on each swarm particle, including Newton's Second Law, thus creating the rules for CFD modelling.
4. To qualitatively compare the swarming CFD approach with the traditional approach.
5. To ascertain the steps towards further development requirements and towards experimental validation and integration into simulation tools.

How CFD could benefit from this work? The current static approaches of representing computational fluid dynamics do not enable designers to fully explore complex interactions between the building, people and air movement. The main motivation for this work is an expectation that the nature inspired approach would provide new dynamic and interactive insights into CFD modelling in comparison with the traditional top-down approaches, whilst significantly reducing simulation time.

Previous work

Although there has been a notable development of swarming algorithms and related methods, and although swarm behaviour appears to be fluid-like, there is very little evidence of swarming methods being used for modelling of fluid flow.

Pioneering work in the field of swarm modelling was first published by Craig Reynolds (Reynolds 1987). In his seminal paper entitled 'Flocks, Herds, and Schools: A Distributed Behavioral Model' he introduced simple rules on an individual level, that created realistic looking and dynamically changing swarm formations without a master controller in the model. These rules were further elaborated in a follow up paper (Reynolds 1999).

Although Reynolds's work was focused on applications for film animation, he influenced an entire new area in science and engineering. Adapted swarming principles were applied to modelling of civil engineering structures (Jankovic et al., 2000, Jankovic et al., 2003), leading to emergent models that represented more realistic dynamic behaviour of bridges, in comparison with top-down models based on the finite element method. Particle swarm optimisation was used in specialised aspects of the flow of traffic (Deng, Tong, and Zhang 2010), and also in fluid flow modelling of heat exchangers (Rao and Patel 2010), in representation of river flow (Gupta and Ganti 2011) and in other systems. Although principles of

emergence inspired by Reynolds's work were applied to CFD modelling using cellular automata (Ljubomir Jankovic 2017), there is no evidence that swarm behaviour has been applied to CFD modelling before.

Method

At the start of this research a particle swarming model was developed using the rules published by Craig Reynolds (Reynolds 1999). These included:

1. **Separation**, to handle particle to particle collision detection and avoidance.
2. **Cohesion**, to target average position of the immediate neighbourhood within the swarm and thus form a group together with the other particles.
3. **Alignment**, to adjust own speed and direction in order to match the speed and direction of the neighbouring particles.

Whilst **Separation** from the above list was considered to be 'sub-conscious' action related to collision avoidance, the other two rules, **Cohesion** and **Alignment**, were considered to be 'conscious' actions as they involved a degree of decision making.

In the first phase of the method development, all three of the above behaviours were embedded in each particle. There was no master controller in the model, ensuring that the resultant behaviour would be truly emergent. In order to optimise the speed of execution of the code, all loops were combined in the **Separation** rule, calculating general neighbourhood metrics such as average position and average velocity of the neighbours, and thus leaving the other two rules free from loops.

The development environment for this research was Unity games engine, chosen for its 3D vector physics capabilities (Unity Technologies 2017). The model was placed in a rectangular room of width/depth/height of 20m/20m/6m, with a 3.7m x 10m controllable roof opening, and with transparent front wall (Figure 1).

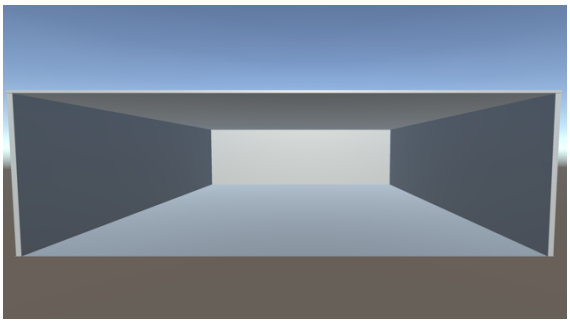

Figure 1: Room in Unity engine with transparent front wall

Swarming model physics

Following the rules by Craig Reynolds (Reynolds 1999), the swarming model physics was developed using pseudo-code introduced in Table 1 through to Table 3.

Table 1: Separation pseudo-code

```
separation += (neignbour[i].position –
myPosition).normalized;
avgSeparation =
separation/neighboursCount;
mySpeed += avgSeparation x sepFactor;
```

Table 2: Cohesion pseudo-code

```
avgPosition += neignbour[i].position;
totalAvgPosition =
avgPosition/neighboursCount;
desiredSpeed = (myPosition –
totalAvgPosition).normalized;
mySpeed += desiredSpeed x
desSpeedFactor;
```

Table 3: Alignment pseudo-code

```
avgSpeed += neignbour[i].speed;
totalAvgSpeed =
(avgSpeed/neighboursCount).normalized;
mySpeed += totalAvgSpeed x
avgSpeedFctor;
```

As the pseudo-code in the above tables represents three-dimensional vector calculus, the operator referred to as '`normalized`' returns the operand vector with a magnitude of one.

The speed of each particle was then processed by the Unity engine (Unity Technologies 2017) to determine three dimensional particle movement.

CFD model physics

After a realistic swarming behaviour was obtained from the model, the 'conscious' actions of **Cohesion** and **Alignment** were removed from the model and replaced by CFD calculations on a particle level. The overall solver, required in top down models, was replaced by particle to particle interaction, as in the initial swarming model.

The particle level calculations included the calculations of thermal conductivity, density, specific heat and temperature for each particle. Heat balance for each cell was calculated as follows:

$$Q = Q_{old} + Q_a + Q_g + Q_r + Q_l + Q_f + Q_b + Q_u + Q_d \quad (1)$$

where

Q – particle heat balance in the current time step (W)
Q_{old} – particle heat balance in the previous time step (W)
Q_a – heat transferred between the particle and room air if the particle is in the room, or between the particle and external air if the particle has left the room (W)
Q_g – heat transferred between the particle and its immediate neighbours (W)
Q_r – heat transferred between the particle and the right wall of the room (W)

Q_l – heat transferred between the particle and the left wall of the room (W)
Q_f – heat transferred between the particle and the front wall of the room (W)
Q_b – heat transferred between the particle and the back wall of the room (W)
Q_u – heat transferred between the particle and the ceiling (W)
Q_d – heat transferred between the particle and the floor (W).

All of the above components of heat balance were modelled as convective heat transfers:

$$Q_x = h\,(T_x - T_{old}) \tag{2}$$

where

Q_x – heat transferred from source x, where x represented *a, g, r, l, f, b, u, d* in the notation from the previous equation
T_{old} – particle temperature in the previous time step.

Particle temperature T was subsequently calculated as

$$T = T_{old} + \frac{Q}{\rho \times V \times c} \tag{3}$$

where

ρ – density (kg/m^3)
V – volume (m^3)
c – specific heat (J/(kg·K)).

Density and specific heat for each particle were calculated in each time step as temperature-dependent, using standard formulae for moist air at 60% relative humidity.

Newton's Second Law was subsequently applied to determine the forces acting on each particle, in order to calculate particle acceleration, speed and movement. The force acting upon each cell particle was calculated as

$$F = F_g - F_b + F_w + F_f + F_p \tag{4}$$

where

F_g – gravitational force, applied in the vertical direction only
F_b – buoyancy force, applied in the vertical direction only
F_w – wind force acting upon each particle
F_p – pressure force $F_p = \Sigma \Delta P_i \times A$
ΔP_i – pressure difference between the particle and each of its i-th immediate neighbours
F_f – friction force, applied in the opposite direction of particle movement
n – number of immediate neighbours for each particle, chosen to be a minimum between total number of particles and 32.

Acceleration and velocity were then calculated for each particle as follows:

$$a = \frac{F}{\rho V} \tag{5}$$

$$v = v_{old} + a \times t \tag{6}$$

where

a – particle acceleration
v – particle velocity in the current time step
v_{old} – particle velocity in the previous time step
t – time step.

Forces, acceleration and velocity were defined as three-dimensional vectors and were handled by the vector calculus within the Unity engine (Unity Technologies 2017), resulting in three-dimensional particle movement.

Comparison with conventional CFD

A conventional CFD model was created in IES Virtual Environment (IES 2017) for comparison purposes. The room dimensions used in this model were the same as in the CFD swarming simulation: width/depth/height of 20m/20m/6m, with a 3.7x10m roof opening.

Figure 2: Room with radiant slab underneath (highlighted in red) in IES VE

Underfloor heating was set in this model using a radiant slab method (IES 2018), in order to have a comparative warm surface as in the swarming CFD model. The method involves a low height room underneath the main room representing radiant slab, so that low height room is heated and the main room is unheated. The floor/ceiling between these two rooms was set to high density concrete with very low surface resistances of 0.001 m^{2}K/W on both sides. The temperature of the heated radiant slab room is controlled by a sensor in the main room. The results of this comparative simulation will be used to evaluate advantages and disadvantages of the swarming approach to CFD simulation.

Simulation experiments

Swarming experiments

The swarming simulation starts with particles lined up along the base of the left and right wall, ready to be deployed in the simulation (Figure 3). These particles are launched 64 at a time on keyboard 'key down' command, so that multiples of 64 particles can be deployed in each simulation (Figure 4).

Soon after the particles are launched into the room, collective behaviour is developed, and a swarm emerges as result of local rules on the particle level (Figure 5). The swarm subsequently explores the space driven by this collective behaviour (Figure 6).

Swarming particles are equipped with trails proportional to their speed, to help with the observation of movement.

When two vertical cylinders are inserted as obstacles into the room, the swarm reacts to these obstacles, breaking before them and re-joining after them (Figure 7).

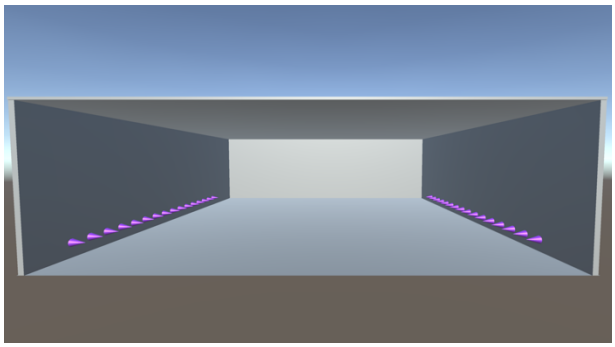

Figure 3: Particles ready for swarming simulation

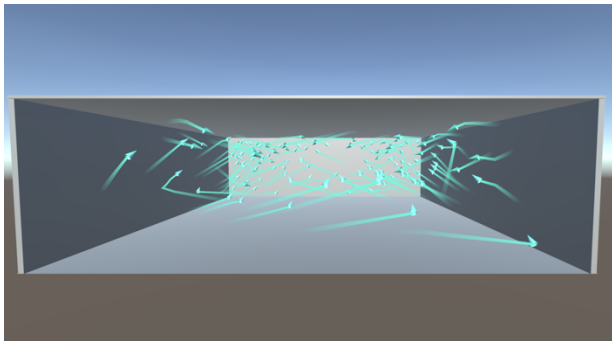

Figure 4: Particles are launched into the room 64 at a time

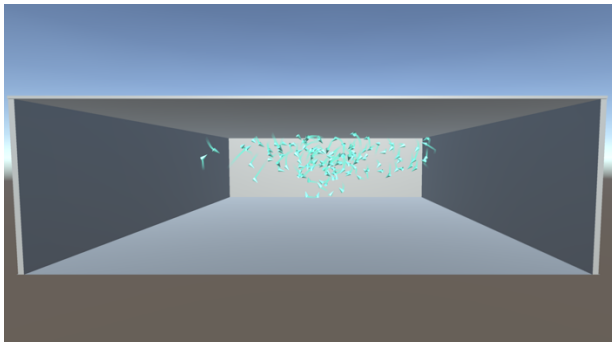

Figure 5: Collective swarming behaviour emerges

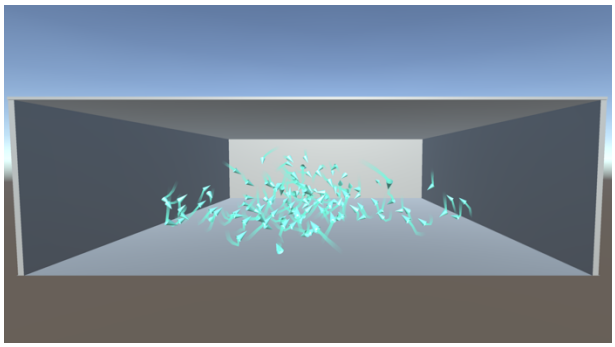

Figure 6: Particles explore the space in a swarm

The images of swarm behaviour simulation shown here were taken during a period of less than a minute, and the simulation continued afterwards without set time limits.

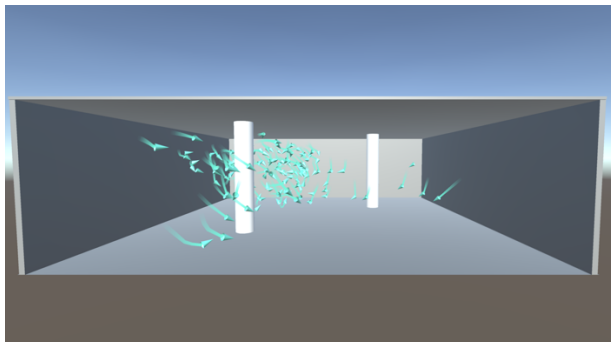

Figure 7: The swarm breaks before an obstacle and re-joins after the obstacle

CFD experiments

In these experiments the particles and the room walls, floor and ceiling are colour coded according to their respective temperatures, and the temperature scale with a range from 0 ºC to 25 ºC is shown to the right of each image. In this simulation the 'sub-conscious' swarming rule of **Separation** is kept and 'conscious' rules of **Cohesion** and **Alignment** are replaced with CFD calculations on a particle level. The room surface and air temperatures are pre-set in the model and kept constant throughout the simulation, and the air particle temperatures are calculated at each time step as specified in the Method section.

At the start of simulation, the particles are lined up along the base of the left and right wall (Figure 8).

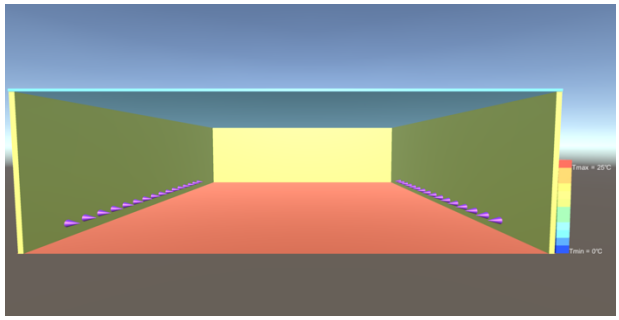

Figure 8: CFD simulation ready to start

Subsequently, a set of 128 particles is launched into the room (Figure 9). They quickly form a flow pattern (Figure 10), where particles receive heat from the warm floor. As particles rise driven by the buoyancy force, their temperature changes through heat transfer with other particles and walls. When they reach the cold ceiling, they lose heat, buoyancy force becomes weaker than gravity, and particles start a downward movement. This flow pattern persists if no other interventions are taken in the model (Figure 11).

An intervention that changes the flow pattern is the opening of the roof hatch. As the roof hatch is opened (Figure 12), the particles rapidly start escaping from the room, propelled by the buoyancy caused by the difference of densities between the warm inside air and the cold outside environment. On leaving the room, the

temperature of each particle drops due to exposure to the cooler outside air.

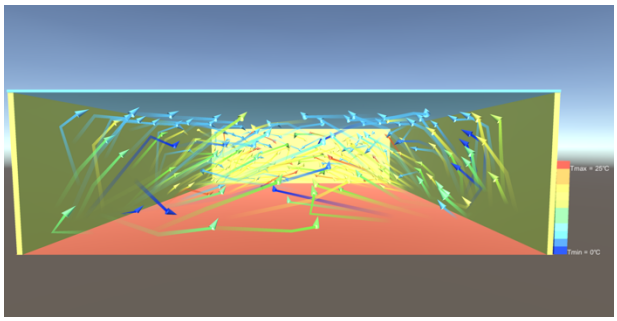

Figure 9: A set of 128 particles is launched into the room

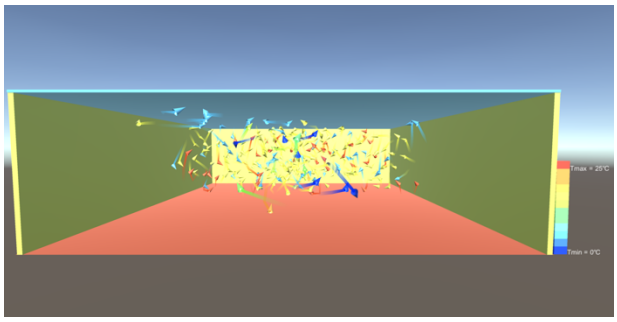

Figure 10: Flow pattern is formed

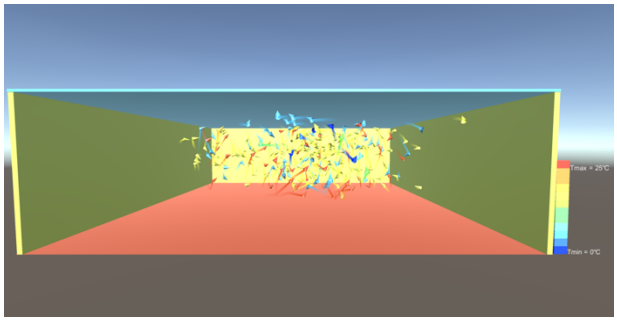

Figure 11: Flow pattern persists

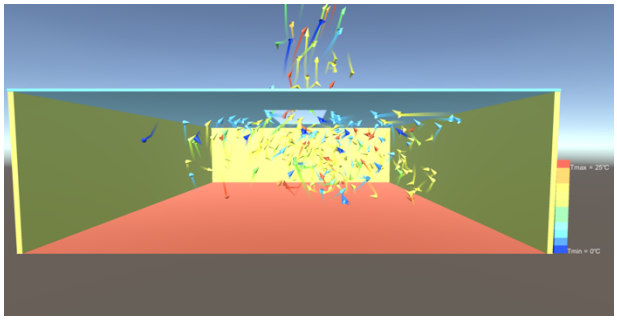

Figure 12: Particles escape through the roof opening

The entire simulation from Figure 8 through to Figure 12 was completed in less than a minute, and the process continued until the simulation was terminated manually.

Comparative CFD simulations

As explained in the Method section, a comparative CFD simulation was developed in IES Virtual environment (IES 2017). The results of hourly simulations were first produced, in order to find equivalent conditions to the swarming CFD simulation (Figure 13). These equivalent conditions were found in the simulation results file on 28th January at 8 am, and were exported into MicroFlo CFD module in IES VE.

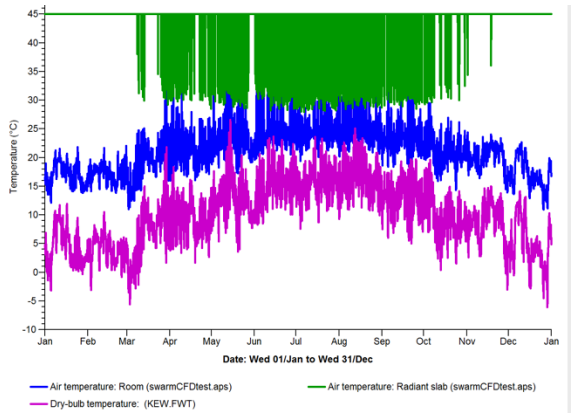

Figure 13: Time series results from IES VE simulation

The results of MicroFlo simulation for the particular day and time were subsequently used to produce representation of temperature distribution shown in Figure 14. Whilst the result indicates rising and falling movement of air represented by darker colours surrounded by brighter colours, the pattern obtained is essentially static and it lacks full representation of air movement. The overall duration of this simulation was 25 minutes, with a result representing a static pattern for a single point in time during the simulation year.

The comparison of results between swarming CFD and conventional CFD is discussed in the next section.

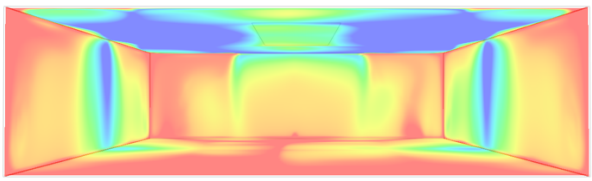

Figure 14: CFD result from IES VE simulation

Discussion and result analysis

As a starting point, a swarming particles model was created using the rules of Separation, Cohesion and Alignment. After the swarming model exhibited realistic behaviour, the Cohesion and Alignment rules, which had been considered to be applied consciously by the swarming agents, were eliminated and replaced by CFD rules underpinned by heat transfer principles and Newtonian physics.

Before the first CFD simulation was carried out, it was expected that the model would exhibit clear and continuous pattern formation, akin to the static patterns

observed in conventional CFD tools. However, much more complex dynamic behaviour occurred. The particles exhibited Brownian movement, colliding with each other and with the boundaries, while gaining heat from warm surfaces and losing heat to cold surfaces. As expected, the particles were rising when they gained sufficient amount of heat and they were falling when they lost heat.

Although this paper does not claim that this proof of concept model fully represents air movement as it occurs in reality, these results show that reality could be far more complex than that represented by conventional CFD tools.

The conventional CFD tool, MicroFlo within IES Virtual Environment that was used for comparison, produced detailed temperature distribution in the room volume for a single point in time within the simulation year. These results were static, and it took 25 minutes of simulation time to produce the results.

In comparison, the swarming CFD simulation was almost instant, producing dynamic results within seconds. However, these results did not include the entire volume of the room, as the particles moved around the room while driven by real time heat gains and losses.

This demonstrates the need for further development of the swarming CFD model, to cover the entire volume of internal spaces. That is not a simple matter of adding more particles, but it would require changes in the model that reduce the impact of colour rendering on the speed of simulation, as well as an introduction of simplified shapes that would render faster than the currently used cones, but would still indicate the direction of particle movement.

Experimental validation of new CFD tools is normally carried out with previously tested software tools that can provide a reference point and a benchmark for the new CFD tool. However, the swarming CFD tool is significantly different from existing tools as it is dynamic, which rules out the static CFD validation tools found in standard simulation software. The validation will therefore need to be carried out in relation to an analogue facility, such as a wind tunnel, or using full scale testing, and/or using water models. This will require access to specialist laboratory facilities and the funding to pay for it.

Assuming that the swarming CFD tool is validated, what would be required to integrate it into an existing building energy simulation tool? This question can be answered in two ways.

First, the integration could be achieved in the form of a game, running as an external process but linked to a designated simulation tool. The advantage of this approach would be that all Unity infrastructure for physics modelling, and all 3D vector calculus, would be supported in their native mode, however this would be a loose integration with the building energy simulation tool and it would require a development of data exchange protocols between the two.

Second, relevant parts of Unity engine infrastructure could be redeveloped in the target simulation tool. Thus, 3D vector calculus and the physics engine would need to be built into the existing simulation tool. This would require more development time, but it would provide compact integration with the building energy simulation tool.

The choice between these two different options will be down to the developer of the simulation tool. Whatever the choice, the integration process will require some development time to implement it.

What opportunities might this approach create for building designers? Current CFD approaches offer detailed but static representation of air distribution in a building for a single point in time in the simulation year. This limits designers in the exploration of dynamically changing conditions in the building, and in running interactive what-if scenarios. For instance, what happens half way through the simulation if we open a window? There are also cases where an existing ventilation system does not fulfil its task and it needs to be improved by additional measures to eliminate overheating. The dynamic interaction of the existing and new ventilation system could be immensely complex and the static tools are not capable of giving sufficient insight into the overall behaviour of such system.

The proposed dynamic approach is capable of reproducing interaction and complex behaviour, as it is based on fundamental principles of particle to particle interaction and laws of physics implemented on a particle level. Therefore, in comparison with the static approach, this dynamic approach will make it easier for building designers to solve complex air flow problems and achieve better thermal comfort for building users, as well as better performing buildings. It will reduce the discrepancy between our expectation of the building behaviour and its actual behaviour, thus reducing the performance gap. It will also give designers more confidence into the performance of their designs, and thus improve building design practice.

The outcome will inevitably bring about a greater insight into the natural processes as they occur in buildings, rather than a ‘frozen’ static representation of these processes offered by the current tools.

Conclusion

The paper presented results of initial research into modelling computational fluid dynamics with swarm behaviour. Clear and nearly static flow pattern formation was expected before the simulation experiments were conducted. However, these experiments showed that flow pattern was dynamic with a high degree of Brownian movement in addition to heat driven flow.

A comparison between the swarming CFD approach and traditional approach showed a dynamic pattern formation that occurred within seconds from the start in the former

approach, while the latter approach took 25 minutes of CPU time to create a static pattern.

Considering that standard simulation tools use Navier-Stokes equations, which describe the system as a whole from the top down, and which have solutions only for simplified cases, a question of trust into the outputs of these tools can legitimately be asked. When we want to fly to the moon, we cannot bring the moon closer to make it easier to get there. Likewise, we cannot legitimately simplify the problem definition just to be able to find a solution if this compromises the accuracy of that solution.

The swarming CFD approach requires improvements in the speed of rendering. For instance, the cones used in the current model take longer to render than some simpler shapes such as cubes. Finding shapes that are less computationally intensive to render but still indicate direction of movement would make this model work faster and capable of using a greater number of particles. These improvements will help to create representation of air movement in the entire room volume. That will need to be followed by experimental validation process in a wind tunnel, water models or full scale tests.

Although this approach requires further development, the proof of concept has been made: swarming principles can be used as a starting point for bottom-up modelling of computational fluid dynamics.

It might be prudent to consider that we may be partially blinded by the conventional methods, which stand as an interface between nature and our understanding of nature. This paper makes a case for investigation of alternative methods for modelling nature that are more akin to the actual processes that occur, rather than to the convoluted mathematical description of these processes that removes certain aspects of representation of the genuine behaviour. Air does not do computation to solve Navier-Stokes equations in order to 'know' which way to flow, so why would we?

References

Deng, Y., H. Tong, and X. Zhang. (2010). Dynamic Shortest Path in Stochastic Traffic Networks Based on Fluid Neural Network and Particle Swarm Optimization. In *2010 Sixth International Conference on Natural Computation*, 5:2325–2329. doi:10.1109/ICNC.2010.5584513.

Gupta, A., and R. Ganti. (2011). Development of ANN River Flow Model Using Particle Swarm Optimization. In *National Conference on Hydraulics and Water Resources-2011*, At Surat, India.

IES. (2017). VE 2017. https://www.iesve.com/VE2017.

IES. (2018). Radiant Slabs in ApacheHVAC. https://www.iesve.com/support/faq/pdf/radiantslabs_faq252.pdf.

Jankovic, L., S. Jankovic, A. H. C. Chan, and G. H. Little. (2000). Structural Simulation Models That Build Themselves. *International Journal of Simulation* 1 (1): 9.

Jankovic, L., S. Jankovic, A. H. C. Chan, and G. H. Little. (2003). Can Bottom-up Modelling in Virtual Reality Replace Conventional Structural Analysis Methods? *Automation in Construction* 12 (2): 133–138. doi:10.1016/S0926-5805(02)00046-8.

Jankovic, L. (2017). Changing the Culture of Building Simulation with Emergent Modelling. In *Building Simulation 2017 - Proceedings of the 15th IBPSA Conference*, 8. San Francisco: IBPSA. doi:10.26868/25222708.2017.062.

Rao, R. V., and V. K. Patel. (2010). Thermodynamic Optimization of Cross Flow Plate-Fin Heat Exchanger Using a Particle Swarm Optimization Algorithm. *International Journal of Thermal Sciences* 49 (9): 1712–1721. doi:10.1016/j.ijthermalsci.2010.04.001.

Reynolds, C. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics* 21 (4): 24–34. http://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/.

Reynolds, C. (1999). Steering Behaviors For Autonomous Characters. http://www.red3d.com/cwr/steer/gdc99/.

Unity Technologies. (2017). Unity Game Development Platform. *Unity*. https://unity3d.com.