EMERGENCE-BASED APPROACH TO COMPUTATIONAL FLUID DYNAMICS

Prof Ljubomir Jankovic
Director, Zero Carbon Lab
Birmingham School of Architecture, Birmingham City University
The Parkside Building, 5 Cardigan Street, Birmingham, UK.
Lubo.Jankovic@bcu.ac.uk

ABSTRACT
Computational Fluid Dynamics (CFD) modelling requires considerable user preparation time and considerable CPU time for reproducing the simplest of behaviours of air movement in and around buildings. This is due to the top-down approach to modelling, in which the system is described with Navier-Stokes simultaneous differential equations, generating large solution space. This paper investigates a radically different approach, in which the system is modelled from the bottom up, thus avoiding the need for Navier-Stokes equations whilst still relying on the same fundamental principles. The paper discusses how this approach can change the culture and the practice of building simulation, with a potential to make sketch drawings become alive and enable hourly time-stepped annual CFD simulations to be conducted on personal computers.

INTRODUCTION
This paper reports on the development of an alternative approach to CFD modelling, which has a potential to considerably reduce the computational intensity in comparison with the current methods, and enable fast and accurate CFD to be carried out as time-stepped simulations over an entire simulation year on personal computers.

Navier-Stokes equations, which were developed in 1820s, are the main method for CFD modelling. They describe the motion of fluid from the top down, using the methods that have not changed much since the 19th century. Although Navier-Stokes equations only represent the Newton’s Second Law and the incompressibility of the fluid (Fefferman, 2006), due to the top-down system based approach no complete solutions exist even today. For this reason the systems being modelled with Navier-Stokes equations need to be simplified in order to find solutions for particular cases. In other words, the problem is adjusted to fit the solution method, instead of adjusting the solution method to fit the problem, with potentially unpredictable consequences.

According to Clay Mathematics Institute (Atiyah and Tate, 2000), Navier-Stokes equations contain one of the seven most important open problems in mathematics that remain unsolved to this date.

Instead of using a system of equations that ultimately cannot be solved, this work uses emergence as a primary modelling method. Emergence as a process occurs in systems with a multiple number of components, in which the components follow simple individual rules, whilst also interacting with other components. Through interaction, and depending on whether the components are connected sparsely, critically or supra-critically, the emergent system will respectively exhibit orderly, complex or chaotic behaviour. Thus, it is through emergence that we get systems in which the whole is more than the sum of its parts, and the behaviour that is not explicitly programmed. Historically, emergence has been at the heart of nature’s designs, producing living organisms on earth for several billion years. Emergence is also the mechanism behind all spontaneous behaviour in the natural world, including the air movement inside and outside of buildings.

Air molecules, or collections of molecules referred to as ‘parcels’, do not ‘know’ how to solve Navier-Stokes equations, but simply follow fundamental rules and self-organise through interaction, resulting in air movement. This research draws on the author’s past work on emergence-based approach to modelling in other fields and seeks to emulate the rules and achieve computational models that are analogous to the underlying natural phenomena of air movement, without an intermediate and inefficient interface of man-made systems of equations.

The paper outlines the method for emergence-based CFD modelling in terms of embedding the underlying physics principles into a Cellular Automata lattice, and investigates the operation of several simple models.

The implications of this work on enabling time-stepped fast CFD simulation over the entire simulation year on personal computers and a potential for change of culture of dynamic modelling and simulation of buildings are discussed.

BACKGROUND
Traditional approach to modelling
In his book ‘A New Kind of Science’, Stephen Wolfram explains that in systems that exhibit complex behaviour, “there is usually no choice but to use some form of approximation” when describing
the system with mathematical equations (Wolfram, 2002). He states “it has almost never been possible to demonstrate that results obtained from such approximations even correctly reproduce what the original mathematical equations would imply.”

This is why there are still open problems in mathematics, such as reported by Atiyah and Tate (2000) in relation to Navier-Stokes equations, which are used as the primary method for modelling the motion of fluids.

Natural phenomena, such as air movement, do not have the means to solve Navier-Stokes equations in order to know which way to go. Instead, these systems simply self-organise, in response to external influences, internal interaction and interaction with boundary conditions. Equations are a human construct, an interface between our knowledge and natural phenomena, which we use to predict behaviour of natural systems. And as simplifications are inevitable in order to enable systems of equations to reproduce complex behaviour of natural systems, it appears that the problem is being modified to suit the solution method, instead of adapting the solution method to suit the problem.

### Top down versus bottom up modelling

The solution of Navier-Stokes equations can be regarded as a top-down modelling approach, in which the entire system is modelled with a system of equations, and a solution of this system, which we already know requires simplifications, is found by applying numerical methods. Top-down models are computed in series, so that the system of mathematical equations is solved completely before the model progresses to the next time step.

Conversely, a bottom-up modelling approach models individual components of a system with simple rules. These rules facilitate interaction between the components, and such interaction gives rise to the system behaviour that is not explicitly programmed, but it emerges from the individual component rules and their interaction. This approach, named ‘emergence’ has been studied intensely (Holland, 2000), and has become one of main approaches in Complexity Science (Kauffman, 2010). This approach has been very successful in reproducing complex behaviour of natural systems, such as animal pigmentation (Ball, 2011), flocking behaviour of birds (Reynolds, 1987), or nest construction behaviour in social insects (Theraulaz and Bonabeau, 1995). Bottom-up models are computed in parallel, so that each component computes its position as result of interactions with the neighbouring components, and driven by internal processes and external influences.

The next section will examine the differences of the computational intensity of these two modelling approaches.

### Top-down versus bottom-up CFD

This section will calculate the total number of floating point operations (FLOPs\(^1\)) for an emergence-based bottom up CFD model and for a Navier-Stokes based top down CFD model.

The calculation is carried out over a two-dimensional lattice (Figure 1), where \(N_x\) and \(N_y\) are the number of cells in X and Y dimensions.

![Figure 1 Square lattice for the experimental working model illustrating interaction of cells with eight neighbours](image)

Each cell interacts with its immediate neighbourhood \(R\), which in two-dimensional space consists of eight cells in total, corresponding to the total number of interactions performed by each cell. This is illustrated in Figure 1 by the black cell surrounded by the eight light-brown cells. The emergence-based simulations reported in this paper use a 200 x 200 square lattice, and therefore the total number of cell-to-cell operations will be

\[
N_x \times N_y \times R = 200 \times 200 \times 8 = 320,000.
\]

The same geometry will be now used to calculate the performance of a Navier-Stokes based CFD model. Groengard and Kropinski (1998) reported a method for solving Navier-Stokes equations using FFT, thus achieving \(O(M \times \log M)\) operations, where \(M\) is a discrete number of points in the domain. The process was conducted iteratively, with up to 128 iterations, until convergence was achieved, in combination with several different numerical methods. Jankovic (2013) reported that FFT requires 11,700 floating point operations per data point. The total number of seconds for this operation was calculated and compared with measured performance of an emergence-based bottom up model, which will be discussed in detail in this paper. The performance comparison is shown in Table 1.

\(^1\) FLOPs = the plural of FLoating point OPeration
Table 1

Time-step based computational intensity comparison between top-down and bottom-up CFD models

<table>
<thead>
<tr>
<th></th>
<th>Interactions (M)</th>
<th>Operations</th>
<th>FLOPs per operation</th>
<th>Iterations</th>
<th>Total FLOPs</th>
<th>Computer speed Giga FLOPS</th>
<th>CPU time (s)</th>
<th>CPU time (minutes)</th>
<th>N-S/E ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navier-Stokes</td>
<td>320,000</td>
<td>5,852,068‡</td>
<td>11700</td>
<td>128</td>
<td>6.85E+10</td>
<td>12</td>
<td>730.338</td>
<td>12.172</td>
<td>-</td>
</tr>
<tr>
<td>Emergence-based</td>
<td>320,000</td>
<td>320,000</td>
<td>2848</td>
<td>1</td>
<td>9.11E+08</td>
<td>12</td>
<td>0.076</td>
<td>0.001</td>
<td>9618</td>
</tr>
</tbody>
</table>

‡ \( M \times \log M \)

The calculation was carried out using performance indicators for a contemporary laptop, with a 2.6 GHz Intel Core i7 processor, performing FFT at 12 Giga FLOPS², as measured by Geekbench ³.

As it can be seen from this table, Navier-Stokes model requires over 12 minutes to complete a single time step, whilst emergence-based model requires 76 milliseconds. This makes the emergence-based model over 9600 times faster, and creates opportunities for a different approach to simulation, as it will be explained later in the paper.

Previous work

There has been a considerable body of work in this field since the 1970s, when initial lattice gas (LG) models were developed, based on interaction of fluid particles on a square lattice (Hardy, Pomeau, and de Pazzis, 1973). That emergence-based model, named HPP after the authors’ initials, had the behaviour restricted in orthogonal directions only, with no diagonal movement.

The HPP model was superseded by FHP model (Frisch, Hasslacher, and Pomeau, 1986), which interacted with cells in six directions, therefore overcoming the restrictions from the previous work. The authors demonstrated that despite discrete Boolean nature of the lattice, it was possible to simulate Navier-Stokes equations, and showed that this approach “can be used to design simple, massively parallel computing machines”.

In the 1980s it was recognised that the lattice gas models were in fact cellular automata (CA) models, which in turn are a representation of physical systems with finite set of values and in a discrete spatial and temporal domain. Salem and Wolfram (1985) worked on fluid modelling using CA and developed an emergence-based model of hydrodynamic flow around a plate. Wolfram subsequently published a basic theory of cellular automaton fluids (Wolfram, 1986).

² FLOPS = Floating point OPerations per Second
³ http://www.primatelabs.com/

Toffoli and Margolus (1987) used cellular automata (CA) to develop simple models of fluid flow. Using simple rules, they achieved emergence-based behaviour of flow tracing, flow past an obstacle, circular waves, and others. They explain that this method is devoid of numerical instabilities and delivering solutions, and their ethos is guided by the notion of “when things are what they seem, we can safely let them do what they must”.

A number of researchers used the Boltzmann equation to replace the need for solving Navier-Stokes equations, thus developing lattice Boltzmann methods for modelling fluid flow (see for instance He et al., 1988).

Developments of structured approaches to using CA for CFD modelling have continued through the 1990s, with significant contributions to modelling physical systems with CA (Chopard, B. and Droz, M., 1998), and through the 2000s on the application of lattice Boltzmann techniques with CA to modelling fluids (Chopard, et al., 2002).

Bagnoli and Rechtman (2008) investigated entropy and chaos in lattice gas CA and achieved a model that exhibits characteristics of hydrodynamic and thermodynamic behaviour.

Toffoli (2009) explained similarities and differences between cellular automata and lattice-gas automata, which have been intertwined in almost all of the previous work. Although these two approaches to modelling fluid dynamics are “two different styles of parallel computation”, “CA provide a quick modelling route to phenomenological aspects of nature”, giving rise to emergence of complex behaviour, “while LG are unmatched as a source of fine-grained models of fundamental aspects of physics”.

Despite the considerable body of work in the application of CA to CFD modelling over the past four decades, it appears that very little has changed in CFD simulations of buildings. The ethos described by Toffoli and Margolus (1987, pp. 171 – 184) as “when things are what they seem, we can safely let them do what they must” has not found its way into changing the culture of building simulation. The
models are still based on systems of Navier-Stokes equations, instead of developments in the direction of emergence-based CFD.

As explained in Table 1, the emergence-based approach is circa 10k times faster, and this paper will explore the opportunities for modelling and simulation of buildings arising from this approach, using a simple bespoke model developed by the author.

METHOD

In essence, there are only two fundamental principles implemented by Navier-Stokes equations: Newton’s Second Law, and incompressibility of fluid. The emergence-based model will therefore do the same, but without a system of equations to solve. The latter will be replaced by the interaction of model components, as shown in Figure 1 Square lattice for the experimental working model illustrating interaction of cells with eight neighbours.

The squares in Figure 1 correspond to cells in a Cellular Automata lattice, a construct widely used for modelling applications in Computer Science and other fields (Wolfram, 2002). Each cell contains a ‘parcel’ of air, a collection molecules that fill a finite area defined by the cell dimensions.

A ‘blueprint’ for the cell is an object, in terminology of object oriented programming. The object contains the implementation of the Second Law, as well as it handles interaction with other object, maintaining the incompressibility of fluid, and replacing the need for a system of equations as result.

Implementation of the Second Law

The Second Law is implemented in each cell as follows:

\[ F = \sum F_i \]  

where

\( F \) = total force on the \((x,y)\) cell, calculated separately in x and y directions

\( F_i \) = individual forces arising from pressure, friction, gravity, buoyancy and wind force

\( n \) = total number of forces acting on each cell.

Acceleration is then calculated separately in x and y directions as:

\[ a = \frac{F}{\rho V} \]  

where

\( \rho \) = density

\( V \) = volume of the cell.

Subsequently, cell velocity is calculated separately in \( x \) and \( y \) directions as:

\[ v^+ = \frac{a}{t} \]  

where

\( t \) = timescale

\( + = \) accumulation over sequential time steps

The change of \((x,y)\) position is subsequently calculated as:

\[ \Delta x^+ = \frac{v_x}{t} \quad \Delta y^+ = \frac{v_y}{t} \]  

Implementation of incompressibility rules

The fluid incompressibility rules are implemented in each cell as shown in Figure 2. As it can be seen from this figure, the incompressibility rules involve the interaction of the cell in position \((x,y)\) with its surrounding neighbourhood consisting of eight cells. The choice as to which rule to apply is dictated by the sign of the cell movement direction, as calculated by Equation (4). The movement is applied only if the accumulated \( \Delta x \) and \( \Delta y \) are greater than the corresponding cell dimension.

These incompressibility rules from Figure 2 are applied to each \( y \) coordinate. The rules that apply to each \( x \) coordinate are similar, but rotated 90° counter clockwise, with reference to the rules in Figure 2.

SOFTWARE IMPLEMENTATION

The emergence-based CFD application described in this paper is implemented in Java. The program consists of two classes: \textit{cell} class and \textit{main} class. The \textit{cell} class implements Second Law and incompressibility rules. The \textit{main} class implements instantiation of the \textit{cell} class into the rectangular grid/lattice; handles calls to the ‘frame’ method in each \textit{cell} instance, running a single time step in it; and handles I/O functions including user interaction.

The system model is created by ‘instantiating’ the \textit{cell} class into required number of copies, which in this particular case is a 200 x 200 lattice, therefore 40,000 instances. During the process of instantiation, each instance is assigned initial properties, which
may differ from instance to instance. In this way the initial and boundary conditions are set in the model.

The following commands are available during the program execution:

- **S** = Step by step
- **A** = Add wall pixels
- **H** = Add heat source pixels
- **N** = Add heat sink pixels
- **D** = Delete pixels
- **B** = Flood fill with wall colour
- **R** = Flood fill with heat source colour
- **K** = Flood fill with heat sink colour
- **G** = Gravity on/off
- **>** = Wind from right on/off
- **<** = Wind from left on/off
- **P** = Print screen to file
- **O** = Output preserved cells
- **I** = Input preserved cells
- **Z** = Run/Pause
- **E** = Exit

These commands enable interactive operation, as it will be explained in the next section.

The examples shown in this paper have been generated on an Apple MacBook Pro, running a 2.6 GHz Intel Core i7 processor under the OS X 10.9.5 operating system.

**EXPERIMENTS AND RESULTS**

A series of test runs were carried out to investigate the behaviour of the model. The results are shown in this section. These results are qualitative only, as the method has not yet been validated through rigorous instrumental testing.

The following parameters characterised the model:

- Heat source temperature 38 °C
- Heat sink temperature 18 °C
- Ambient air temperature 20 °C

All run sequences shown in this section took only a few seconds from the beginning to the end.

The image in Figure 3a) started as a drawing of a horizontal red heat source line and a sloping black obstacle line. After the run is initiated, the model goes through a sequence a) to d), exhibiting emergent behaviour in passing around the obstacle.

The image in Figure 4a) started as a drawing of a vertical heat source line (red) and a vertical heat sink line (blue). After the run is initiated, the flow from the heat source, being warmer than the ambient air, exhibits buoyancy and upward movement, and the flow from the heat sink, being cooler than the ambient air, exhibits a downward movement. This is shown as a sequence from a) to d) in Figure 4.

The image in Figure 5 shows the effect of wind on the upward flow from the heat source. The sequence started in Figure 5a) as a drawing of a horizontal red heat source line. In Figure 5b) wind from the left is initiated, and the flow responds by swaying to the right. This behaviour continues through Figure 5c), and in Figure 5d) wind from the left is switched off. The flow starts recovering into a vertical movement up to Figure 5e). In Figure 5f) wind from the right is switched on and the flow starts swaying to the left. In Figure 5g) wind is switched off and the flow gradually recovers into vertical movement through to Figure 5h).
The sequence of images in Figure 6 shows the effect of vertical wind shields on air flow. The model started as a drawing of a horizontal heat source line (red), and two vertical black wind shield lines. Wind from the left is switched on in Figure 6a) and the vertical air flow responds only in the regions where it is not protected by the shield from the left side, thus swaying towards the right shield in Figure 6b) and through to Figure 6c). In Figure 6d) wind from the left is switched off and wind from the right is switched on. Figure 6e) and f) show the response of air flow swaying to the left, and the effect of vertical shields.

Figure 5 Wind from right and left

Figure 6 The effect of vertical wind shields

Figure 7 An enclosure with floor heating and chimney
The sequence of images in Figure 7 shows air flow in an enclosure with floor heating and a vertical chimney-like opening at the top left. The model started as a line drawing in Figure 7a). After the run is initiated, a flow pattern develops in Figure 7b), and the warm air starts escaping through the chimney in Figure 7c). In Figure 7d) wind from the left is switched on, and the air flow above the chimney starts swaying to the right.

Real time interaction with the model is demonstrated in Figure 8. A freehand inverted funnel shape was drawn with horizontal heat source line at the base, as shown in Figure 8a). After the run started, the flow through the funnel develops in Figure 8b) and c). A sloping obstacle is then inserted into the air flow in Figure 8d) and the effects of that are shown in Figure 8e) and f).

Although the model uses a square lattice, its behaviour is not restricted in diagonal directions, which occurs in sequence of two steps: left+up; left+down; right+up; right+down.

The simplicity of the model description in this paper may appear as lacking reproducibility. However, this simplicity is deceiving, as the equations and cellular automata rules reported here are all there is to it, and are sufficient to embark on its software implementation. For this very reason of the simplicity of the underlying rules of complex phenomena, Gribbin (2009) explains that ‘deep simplicity’ is the essence of complex behaviour in nature.

The work presented shows only temperature distribution, however each ‘parcel’ of air on the square lattice contains information on other significant parameters, including velocity, as shown in Equation (3), and therefore temperature and velocity representations of the model behaviour are interchangeable.

Experimentation with three-dimensional extension of this method is in progress. This will increase the computational intensity of the model. However, adding third dimension in Table 1 to both Navier-Stokes and emergence-based computation will maintain similar performance ratio in favour of the emergence-based approach.

CONCLUSIONS

This work demonstrates how emergence-based CFD can enable simple line drawings to become ‘alive’, and represent air flow driven by heat sources and heat sinks, and influenced by solid obstacles. These elements of the model can be inserted at any point after the simulation was initiated, therefore creating opportunities for interaction with the model behaviour throughout the simulation, thus increasing our understanding of the complex behaviour of air flow in and around buildings.

The method effectively comprises a ‘serous games’ approach to building simulation, in which the user can interact with the sketch of internal or external environment, and can ‘push it and poke it’ and see what happens.

The solution to Navier-Stokes equations is replaced by emergence, facilitated by the interaction of air ‘parcels’ on a square lattice, whilst keeping the two fundamental principles that are modelled by Navier-Stokes equations: application of the Second Law and the incompressibility of fluid.

A nearly 10k times speed increase in comparison with Navier-Stokes based CFD was calculated, suggesting a potential for conducting hourly CFD simulation over the entire simulation year on a personal computer, which has not been possible with the current state of the art software. This would therefore create opportunities for a culture change in building simulation.
The paper reports on early results, and further research and development are planned to improve the scope and functionality of this approach, including an extension into third dimension. This will be followed by experimental validation based on instrumental measurements of air flow in laboratory conditions. Comparisons with top-down Navier-Stokes models will be carried out at the same time.

ACKNOWLEDGEMENT

This work was inspired by related work on emergent modelling applied to structural design in construction industry, funded by the UK Engineering and physical Sciences Research Council (EPSRC), grant ref. GR/M75273/01, in which the author contributed to the development of emergence-based models of civil engineering structures.

NOMENCLATURE

CA = Cellular Automata
CFD = Computational Fluid Dynamics
CPU = Central Processing Unit
FFT = fast Fourier transform
FLOPs = the plural of FLoating point OPeration
FLOPS = FLoating point OPerations per Second
FHP = Frisch, Hasslacher, and Pomeau model
HPP = Hardy, Pomeau, and de Pazzis model
LG = Lattice Gas

REFERENCES


