CFD Plus Marching Learning Application on Natural Ventilation Design

The 3rd Annual Progress Report

Written by Wang Xin

(SID: 52977064, Program:CA/P)

14 Feb of 2021

Contents

1 Introduction of the Research

1.1 Background

In the built environment, adequate natural indoor air ventilation should be provided for the purpose of residents' safety and health when designing a building. A ventilation indicator with minimum 1.5 ACH, as a minimum requirement from the Building (Planning) Regulation of Hong Kong Government, shall be provided for a habitable room of a domestic building based on the performance-based approach [1]. On the other hand, effective natural ventilation application can make contribution to providing good indoor air quality, improving thermal comfort and productivity of occupants as well as saving energy [3].

The Computational Fluid Dynamics (CFD) theory has been developing for decades, which is able to provide sufficient information about indoor flow pattern, temperature, air movement and indoor pollutant distribution etc. The ventilation performance (e.g. ACH) of an indoor environment can be evaluated by using the field information of pressure and velocity etc after the numerical CFD computation as shown in Figure 1. However, the drawback of this method is that a very long computation time is usually unavoidable to get the result because of the large volume of computational grids for discretization of the external and internal building models. Therefore, the optimization of the CFD application on the urban-scale level is researched. Moreover, a solution to this problem from a new perspective is proposed in this study by using CFD plus machine learning algorithm.

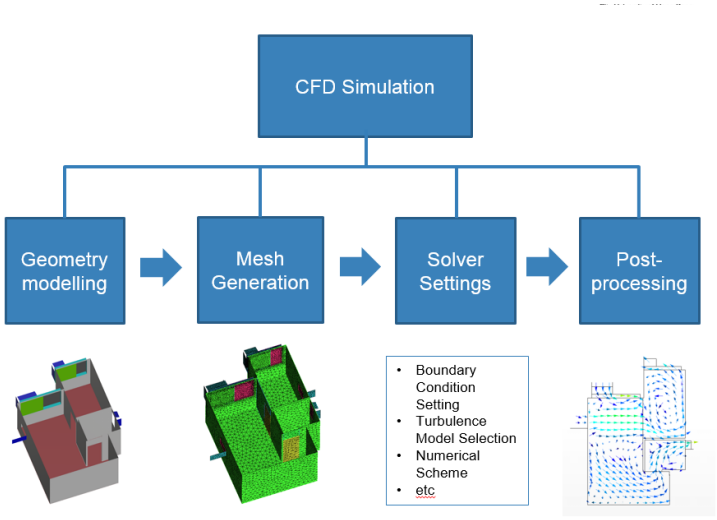


Figure 1 Mechanism of CFD simulation using an internal building model as example

In Chapter 2, a literature review on the CFD theory and machine learning algorithm is described, briefly illustrating the theoretic concept, technique development and major methods used in the research field. In Chapter 3, the CFD theory, methodology and application for evaluation of the urban-scale natural ventilation is elaborated in detail. Thereafter, the new proposed CFD plus machine learning approach is introduced in Chapter 4 and a case study is presented at the end of this chapter. Eventually, in order to implement the CFD simulation introduced in the previous chapters, chapter 5 addresses the practical software design with integration of the CFD working steps. Last but not least, the future study plan is discussed in the last Chapter 6.

1.2 Objective of this study

Firstly, the CFD theory would been investigated deeply, including discretization scheme techniques, high quality mesh generation techniques and advanced turbulence models in order to achieve high-fidelity CFD simulation results. And the error analysis of the decoupled and coupled CFD application on the urban-scale natural ventilation would be studied.

Secondly, the ventilation performance would be predicted by using the regression algorithms based on the dataset from CFD simulations.

Thirdly, an integrated software would be developed to implement the CFD techniques proposed in this study into practice with convenience.

2 Literature review

2.1 CFD theory

Computational Fluid Dynamics (CFD) derived from the classical fluid dynamics and became a specialized discipline along with the advancement of computer technology, as one of three study approaches to fluid flow problems together with theoretical and experiential approaches. There are three main research methods in CFD field, namely direct numerical method, Reynolds averaged method and large eddy method respectively [4].

2.1.1 Direct numerical simulation

Direct numerical simulation is a perfect way to simulate the flow field because it can get absolutely accurate result. However, this method needs very fine grids and very short time step to include the smallest length vortex which is not available in most of cases because of the limit of computational techniques.

2.1.2 Reynolds averaged method

Reynolds averaged method is widely-used in CFD field as it has experienced a long time of development. It has a good reputation for its efficiency and easy implementation [4]. The classical theory convers roughly the period 1895-1970, dating from the introduction of the time-averaged equations of motion by Reynolds [2]. The basic concept is that the wind can be considered as the combination of time-averaged wind and fluctuating wind. The use of averaging procedure leads to the Reynolds averaged equation.

Closure problem

After introducing the averaged value, new unknown item, Reynolds stress ($-\overline{\rho u_i u_j}$ or τ_{ij}) need to be solved. In order to solve the closure problem, the Boussinesq eddy viscosity hypothesis [2] is introduced:

$$\tau_{ij} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3} \rho k \delta_{ij}$$

where $k\ is\ the\ turbulent\ kinetic\ energy,\ \delta_{ij}\ is\ he\ Kronecker\ delta, \mu_T \equiv$ eddy viscosity.

Depending on the number of partial differential equations for solving μ_t, the turbulence models can be categorized into zero-equation, one-equation and two-equation models etc. Among them, k-epsilon model [5] is the most popular in the engineering application. Excepting the k-epsilon standard model, more new and revised models appeared, such as LK model proposed by Launder and Kato [6], MMK model proposed by Murakami et al [7].

2.1.3 Large eddy simulation

With the advance of computational resources in recent years, more complicated techniques such as LES, have attracted more and more attention in wind engineering. Large eddy simulation is an alternative to solve the equation which deals with large eddy and small eddy independently. Direct numerical simulation is used for the large eddies while Subgrid-Scale (SGS) approach is applied for the small eddies. So it can give an accurate prediction of the flow field compared with other turbulence models. But it also requires more powerful computer to generate very fine subgrids and very long time cost. Murakami [7] reviewed the state-of-the-art LES applications in wind engineering and commented that the LES with a dynamic subgrid-scale (SGS) model is a promising tool for accurately predicting the flow field around a bluff body compared with other turbulence models. In fact, a variety of SGS models have been proposed, including the classic Smagorinsky model established by Smagorinsky [8] dynamic Smagorinsky- Lilly model developed by Germano et al. [9] and revised by Lilly [10], wall-adapting local eddy-viscosity model proposed by Nicoud and Ducros [11] and dynamic SGS kinetic energy model presented by Kim and Menon [12].

2.2 Machine learning algorithm

Machine learning describes that model creation from an amount of dataset by using a specified range of algorithms [15]. Based on the characteristics of the learning process, machine learning algorithms can be categorized into supervised learning and unsupervised learning algorithm respectively [14]. In supervised learning techniques, a hypothesis function would be constructed from an m-dimensional input feature vector to an n-dimensional output feature vector and then the free parameters of the hypothesis function would be trained by using a major portion of the dataset. Least square regression algorithm with a polynomial function to fit the dataset is a simple example for supervised learning, in which the feature vector comprised by powers of the input variable to increase the flexibility for a range of functions it can fit.

There are mainly two classes of supervised learning algorithms, namely parametric supervised learning algorithm and non-parametric supervised learning algorithm. The former one characterized as appearance of a specific form of hypothesis function between the input and output vectors while no specific form is assumed for the latter one. The nearest neighbor regression algorithm is an example for non-parametric supervised learning, that can give closer prediction to the true value when enough dataset is given. The other more complex non-parametric algorithms include Gaussian processes [16], kernel regression [17], and locally-weighted regression [18] etc. Parametric supervised learning algorithms are able to make fast prediction once trained successfully while their drawback is lack of flexibility for fitting complex function relationship. Non-parametric algorithms can provide sufficient flexibility but unfavorable scaling properties would be encountered when the dimension size of the feature vector is very large. Neural network model is able to provide the balance between the parametric and non-parametric learning algorithms [19]. The figure below is an example of neural network model, consisting of an input layer, two hidden layers and an output layer. The network begins with an input feature vector in the input layer which would be fed to the first hidden layer and the output from each layer would be fed as the input to the next layer until to the final output layer where the output value are the prediction of interest quantity.

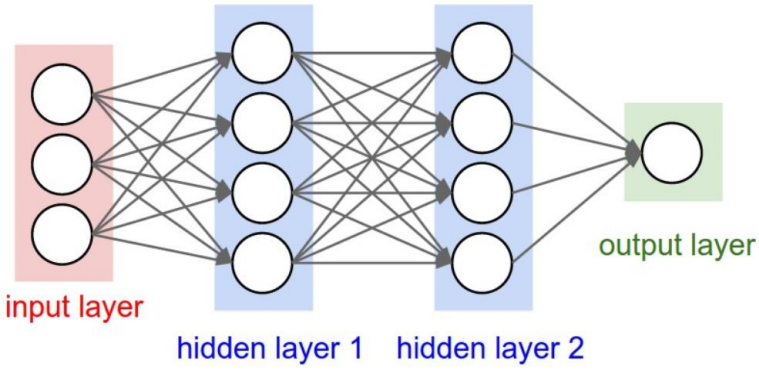


Figure 2 A diagram of neural network model

3 CFD Theory on Urban-Scale Natural Ventilation Application

3.1 CFD simulation methodology

3.1.1 Governing equation

Fluid movements comply with three basic laws which are mass, momentum and energy conservation equations (Abbott and Basco, 1989). They are also called governing equations in CFD theory and the general expression for the conservation equations can be written as shown in the following.

$$\frac{\partial}{\partial t}(\rho\phi) + \frac{\partial}{\partial x_i}(\rho U_i \phi) - \frac{\partial}{\partial x_i}(\Gamma_\phi \frac{\partial \phi}{\partial x_i}) = \mathrm{S}_\phi$$

where t is the time, ρ the air density, ϕ the transport variable such as velocity, U_i (i =1,2 and 3) for three components of momentum, x_i the coordinate, Γ_ϕ the effective diffusion coefficient, S_ϕ the source term.

Please refer to the coefficients in the table below for the continuity, momentum and energy equation respectively.

Equation	ϕ	Γ_ϕ	S_ϕ
Continuity	1	0	0
Momentum	U V W	$\mu_{eff}(\mu+\mu_t)$ $\mu_{eff}(\mu+\mu_t)$ $\mu_{eff}(\mu+\mu_t)$	$-\frac{\partial}{\partial x_i}\left(P+\frac{2}{3}\rho k\right)+\frac{\partial}{\partial x_i}\left[\mu_{eff}\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right)\right]+B_i$
Energy	H	$\frac{\lambda}{C_p}+\frac{\mu_t}{\sigma_H}$	$\frac{\partial \mathrm{P}}{\partial \mathrm{t}}$

Table 1 coefficients for the continuity, momentum and energy equation

3.1.2 Turbulence model

As the air movement within an indoor environment is a turbulent flow problem, a turbulence model is needed to simulate the eddies in the airflow. The $k-\varepsilon$ two equation turbulence model is selected in this study. This model was proposed by Lauder and Spalding (1974). It can be expressed as:

$$k = \frac{1}{2}\left[\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right] \qquad \varepsilon = v_t \overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}}$$

Governing equation for turbulent kinetic energy k:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_i)}{\partial x_i} + \frac{\partial(\rho u_j k)}{\partial x_j} = \frac{\partial}{\partial x_i}\left[\left(\mu + \frac{\mu_t}{Pr_k}\right)\frac{\partial k}{\partial x_i}\right] + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{Pr_k}\right)\frac{\partial k}{\partial x_j}\right] + \mu_t\left[2\left(\frac{\partial u_i}{\partial x_i}\right)^2 + 2\left(\frac{\partial u_j}{\partial x_j}\right)^2 + \left(\frac{\partial u_i}{\partial x_i} + \frac{\partial u_j}{\partial x_j}\right)^2\right] - \rho\varepsilon$$

Governing equation for dissipation of turbulent kinetic energy ε :

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial(\rho\varepsilon u_i)}{\partial x_i} + \frac{\partial(\rho u_j \varepsilon)}{\partial x_j} = \frac{\partial}{\partial x_i}\left[\left(\mu + \frac{\mu_t}{Pr_\epsilon}\right)\frac{\partial k}{\partial x_i}\right] + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{Pr_\epsilon}\right)\frac{\partial k}{\partial x_j}\right] + c_1\frac{\varepsilon}{k}\mu_t\left[2\left(\frac{\partial u_i}{\partial x_i}\right)^2 + 2\left(\frac{\partial u_j}{\partial x_j}\right)^2 + \left(\frac{\partial u_i}{\partial x_i} + \frac{\partial u_j}{\partial x_j}\right)^2\right] - \rho c_2\frac{\varepsilon^2}{k}$$

Where typical constants are

$$Pr_k = 1.0 \quad , \quad Pr_\epsilon = 1.3 \quad , \quad c_1 = 1.44 \quad , \text{ and } \quad c_2 = 1.92$$

And the eddy viscosity is related to ε by $\mu_t = \rho c_\mu \frac{k^2}{\epsilon}$ $where\ c_\mu = 0.09$

3.1.3 Discretization of the governing equation

In order to solve this Partial Differential Equation (PDE), the discretization method is introduced and developed, like Finite Difference Method (FDM), Finite Volume Method (FVM) and Finite Element Method (FEM) etc. In this study, the FVM is adopted, which is the most widely used in CFD simulation. A one dimensional steady incompressible flow is used as an example to illustrate the discretization process.

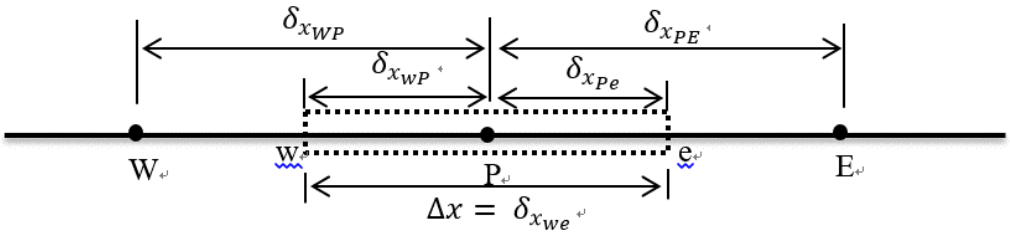


Figure 3 Diagram of discretization in one-dimension

The first step in the finite volume method is to divide the domain into discrete control volumes, which is completed by mesh generation in the whole computational domain. In this simple example, a general nodal point is identified by P and its neighbors in a one-dimensional geometry, the nodes to the west and east, are identified by W and E respectively. The west side face of the control volume is referred to by w and the east side control volume face by e. The distances between the nodes W and P, and between nodes P and E, are identified by $\delta_{x_{WP}}$ and $\delta_{x_{PE}}$ respectively. Similarly the distances between face w and point P and between P and face e are denoted by $\delta_{x_{wP}}$ and $\delta_{x_{Pe}}$ respectively. The control volume width is $\Delta x = \delta_{x_{we}}$.

The second step is the integration of the governing equation over a control volume to yield a discretized equation at its nodal point P.

$$\int_{\Delta V} \frac{d}{dx}(\rho u \phi) dV = \int_{\Delta V} \frac{d}{dx}(\Gamma \frac{d\phi}{dx})\, dV + \int_{\Delta V} S dV$$

$$= (\rho u A \phi)_e - (\rho u A \phi)_w = \left(\Gamma A \frac{d\phi}{dx}\right)_e - \left(\Gamma A \frac{d\phi}{dx}\right)_w + S\Delta V$$

To obtain discretized equations for the convection-diffusion problem we must approximate the terms in equation. It is convenient to define two variables F and D to represent the convective mass flux per unit area and diffusion conductance at cell faces:

$$\mathrm{F} = \rho \mathrm{u} \quad \text{and} \quad \mathrm{D} = \frac{\Gamma}{\delta \mathrm{x}}$$

The cell face values of the variables F and D can written as

$$F_w = (\rho \mathrm{u})_w, \qquad F_e = (\rho \mathrm{u})_e$$

$$D_w = \frac{\Gamma_w}{\delta x_{WP}}, \qquad D_w = \frac{\Gamma_e}{\delta x_{PE}}$$

To calculate gradients at the control volume faces an approximate distribution of properties between nodal points is used.

a. Discretization of the diffusion term

Linear approximations called central differencing is the simplest way of calculating interface values and the gradients and it normally does not require further consideration. In a uniform grid linearly interpolated values for

$$\Gamma_w = \frac{\Gamma_W + \Gamma_P}{2}$$

$$\Gamma_e = \frac{\Gamma_P + \Gamma_E}{2}$$

The diffusive flux terms are evaluated as

$$\left(\Gamma A \frac{d\phi}{dx}\right)_e = D_e(\phi_E - \phi_P)$$

$$\left(\Gamma A \frac{d\phi}{dx}\right)_w = D_w(\phi_P - \phi_W)$$

b. Discretization of the convection term

In order to ensure the stability during the CFD computation, discretization of the convection term should fulfill the requirements of conservativeness, boundedness and transportiveness. Several differencing schemes are discussed below.

Upwind differencing scheme

When the flow is in the positive direction, $u_w > 0, u_e > 0$ ($F_w > 0, F_e > 0$), it sets

$$\phi_w = \phi_W$$

$$\phi_e = \phi_P$$

And the convection term becomes

$$F_e(\phi_P) - F_w(\phi_W)$$

When the flow is in the negative direction, $u_w < 0, u_e < 0$ ($F_w < 0, F_e < 0$), it sets

$$\phi_w = \phi_P$$

$$\phi_e = \phi_E$$

And the convection term becomes

$$F_e(\phi_P) - F_w(\phi_E)$$

Remark: upwind differencing scheme fulfills all the requirements of conservativeness, boundedness and transportiveness, hence, it is stable during iterative computation. But the drawback is also obvious that the accuracy in terms of Taylor's Series Truncation error is just first order and it would produce false diffusion problem in some conditions.

Linear upwind differencing scheme

When the flow is in the positive direction, $u_w > 0, u_e > 0$ ($F_w > 0, F_e > 0$), it sets

$$\phi_w = 1.5\phi_W - 0.5\phi_{WW}$$

$$\phi_e = 1.5\phi_P - 0.5\phi_W$$

And the convection term becomes

$$F_e(1.5\phi_P - 0.5\phi_W) - F_w(1.5\phi_W - 0.5\phi_{WW})$$

When the flow is in the negative direction, $u_w < 0, u_e < 0$ ($F_w < 0, F_e < 0$), it sets

$$\phi_w = 1.5\phi_P - 0.5\phi_E$$

$$\phi_e = 1.5\phi_E - 0.5\phi_{EE}$$

And the convection term becomes

$$F_e(1.5\phi_P - 0.5\phi_E) - F_w(1.5\phi_E - 0.5\phi_{EE})$$

Remark: linear upwind differencing scheme satisfies the characteristics of conservativeness and transportiveness while it is not bounded in some conditions that would result in negative coefficient in the algebraic equations. It is considered to have higher accuracy as second order than the upwind scheme.

TVD differencing scheme

In order to keep the merits of the upwind scheme meanwhile having higher order accuracy, the Total Variation Diminishing scheme is developed by introducing the flux limiter function. Van

Leer limiter function is used in this study to compare with other differencing scheme mentioned above.

Remark: TVD scheme has the characteristics of conservativeness, transportiveness and boundedness in the same time it has higher order accuracy.

c. Discretization of the source term

For the source term, the finite volume method approximates the source term by means of a linear form:

$$S\Delta V = S_u + S_P\phi_P$$

After discretization process for diffusion, convection and source terms, the PDE equations would be converted to a set of algebraic equations, which can be written as in the general form

$$a_P\phi_P = \sum a_{nb}\phi_{nb} + b$$

Adopting the linear upwind scheme as an example, the coefficients a_{nb} are shown below:

$$a_W = (D_w + 1.5\alpha F_w + 0.5\alpha F_e)$$

$$a_{WW} = -0.5\alpha F_w$$

$$a_E = (D_e - 1.5(1-\alpha)F_e - 0.5(1-\alpha)F_w$$

$$a_{EE} = 0.5(1-\alpha)F_e$$

$$b = S_u\Delta V$$

$$a_P = a_W + a_{WW} + a_E + a_{EE} + F_e - F_w - S_P\Delta V$$

Where α equals to 1 if the flow direction is positive and α equals to 0 if the flow direction is negative.

3.1.4 Solver for algebraic equations

The SIMPLE algorithm

The acronym SIMPLE stands for Semi-Implicit Method for Pressure-Linked Equations. The algorithm is essentially a guess and correct procedure for the calculation of pressure. To initiate

the SIMPLE calculation process a pressure field p^* is guessed. Discretized momentum equations are solved using the guessed pressure field.

$$a_w u_w^* = \sum a_{nb} u_{nb}^* + (p_W^* - p_P^*) A_w + b_w$$

$$a_s v_s^* = \sum a_{nb} v_{nb}^* + (p_S^* - p_P^*) A_s + b_s$$

Correction p' is defined as the difference between the correct pressure field p and the guessed pressure field p^*, so that $p = p^* + p'$. Similarly we have $u = u^* + u'$, $v = v^* + v'$.

$$a_w u_w' = \sum a_{nb} u_{nb}' + (p_W' - p_P') A_w$$

$$a_s v_s' = \sum a_{nb} v_{nb}' + (p_S' - p_P') A_s$$

SIMPLE algorithm omit $\sum a_{nb} u_{nb}'$ and $\sum a_{nb} v_{nb}'$ terms and simply the equation

$$u_w' = d_w (p_W' - p_P')$$

$$v_s' = d_s (p_S' - p_P')$$

Where $d_w = \frac{A_w}{a_w}$ and $d_s = \frac{A_s}{a_s}$

The modified equations become:

$$\mathrm{u}_w = u_w^* + d_w (p_W' - p_P')$$

$$v_s = v_s^* + d_s (p_W' - p_P')$$

As mentioned beforehand, the velocity field should satisfy continuity equation. Correction p' can be calculated by

$$a_P p_P' = a_W p_W' + a_E p_E' + a_S p_S' + a_N p_N' + b_P'$$

Under relaxation

The pressure correction equation is susceptible to the divergence. Under relaxation is a common method to control the divergence by：

$$p^{new} = p^* + \alpha_p p'$$

Where α_p is the pressure under-relaxation factor. Taking α_p between 0 to 1 allows us to control the divergence. If the value is large, the speed of divergence is large but the stability reduce, vice-versa.

This method is also can be used for velocity.

$$u^{new} = \alpha_u u' + (1 - \alpha_u) u^*$$

$$v^{new} = \alpha_v v' + (1 - \alpha_v) v^*$$

Usually the under-relaxation factor is set as 0.5 at the first run. Then according the iteration condition, the suitable value can be decided.

The logical diagram for the SIMPLE algorithm is shown in Appendix A.

3.2 Optimization of CFD application on urban-scale level

3.2.1 Problem formulation

The CFD model of urban-scale built environment normally covers several kilometers of land with high building density and complex topography features. It is considered as a difficult problem for discretization of such model in CFD meshing process. Traditionally, the unstructured tetrahedron meshing strategy has been adopting widely in the industry application because of the automatic mesh element generation feature. However, the drawback of this meshing method is also very obvious that huge volumes of mesh elements would be generated with approximately more than double times total number of mesh elements when compared to the structured meshing method under similar grid resolution configuration. Another troublesome issue of the unstructured meshing method is lack of flexibility on mesh size setting control near the atmosphere boundary layer of the model where the computational result is normally more interested by users and thereby the mesh size is required adequately fine for simulation accuracy. In terms of the structured meshing method, although it has the merits on less number of mesh elements and can fit the demand on the smaller mesh size near the atmosphere boundary layer in height direction by easily controlling the aspect ratio of the hexahedron mesh, it lack of flexibility to fit the irregular building shapes in a convenient way.

3.2.2 Hybrid mesh strategy tailored for urban-scale built environment model

To address the difficulties appeared in the unstructured and structured meshing methods, the tailored hybrid mesh strategy for urban-scale built environment model is proposed in this study which has considered the meshing demands from the large scale model with sufficient flexibility on controlling the mesh size setting near atmosphere boundary layer, fitting the complex building shapes and keeping the total number of mesh elements small. The characteristics of the CFD simulation for urban-scale level ventilation is summarized as below:

1. Computational domain is quite large in length, width and height;
2. Wind profile of the atmosphere boundary condition is required and finer grid resolution near the ground level is more accurate;
3. Coarse mesh is allowable for outer zone while fine mesh is required near building surfaces and ground surfaces;

4. Boundary layer is required when studying the wind performance at the pedestrian level.

With consideration of the abovementioned characteristics and producing satisfactory mesh scheme, the hybrid mesh strategy would divide the computational domain into outer zone and inner zone and structured meshing method would be applied to the outer zone while the unstructured meshing method for the inner zone. And at the interface of the outer and inner zones, pyramid elements are used to connect each other. To be more detailed, the outer zone would be cut into 17 sub-blocks keeping the mesh setting at the interfaces among the sub-blocks same for connectivity as shown in Figure 4.

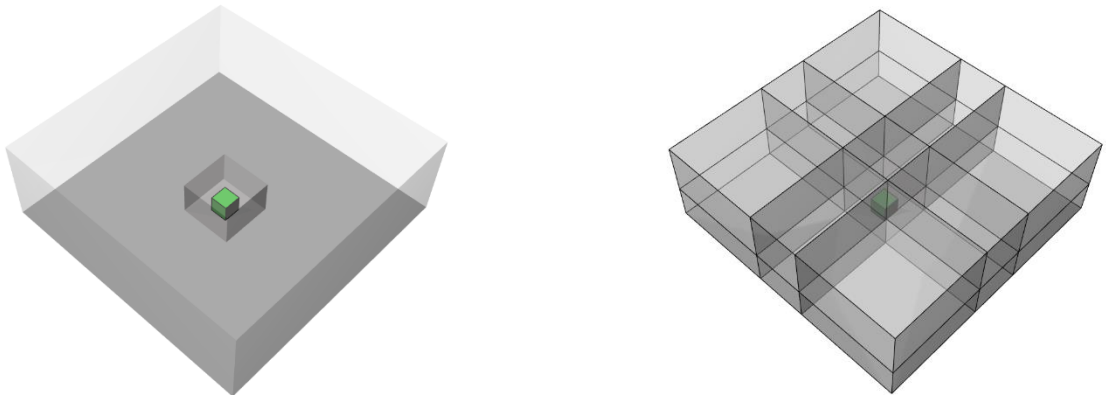


Figure 4 Division of the model into outer zone and inner zone

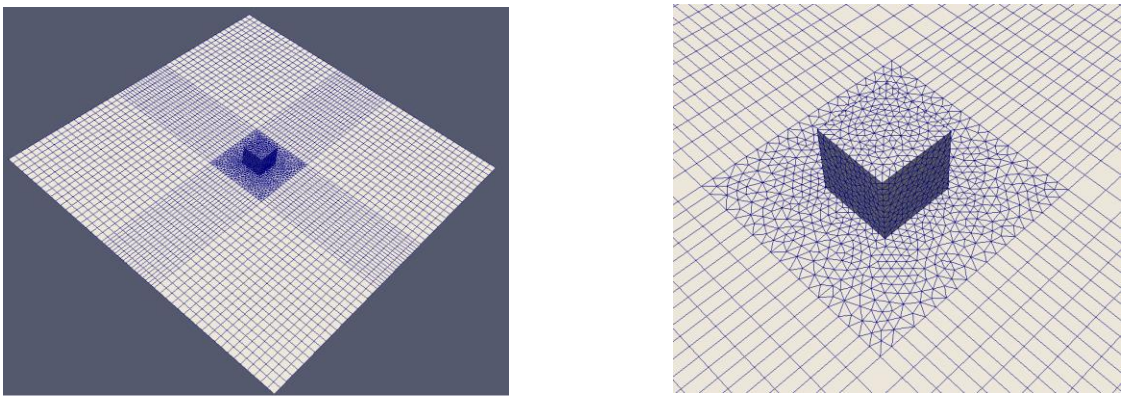


Figure 5 Mesh generation illustration by using the hybrid mesh strategy

Taking a simple model with only one cubic building as an example as illustrated in Figure 5, only the inner zone which contains the building shape used the unstructured mesh while structured mesh were applied to all the sub-blocks in the outer zone. Figure 6 shows connectivity at the interface of the outer and inner zone through the sectional plot.

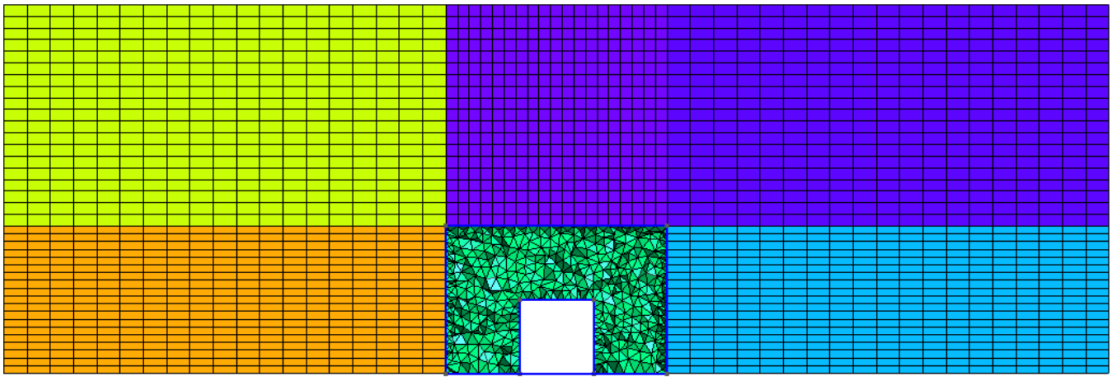


Figure 6 Sectional plot of the hybrid mesh scheme

3.2.3 Hybrid mesh application examples

To explore the applicability of the hybrid mesh strategy under various conditions, several geometries from simplicity to complexity are demonstrated in this section. Simple single building model with different shapes are tested first and the result is satisfactory as shown in Figure 7. In the next step, a cluster of high-rise residential buildings are tested, which represent the urban-scale model and the result in Figure 8 has shown the good mesh quality of this hybrid mesh scheme. Finally, a complex building model presenting the industrial application level is explored as demonstrated in Figure 9.

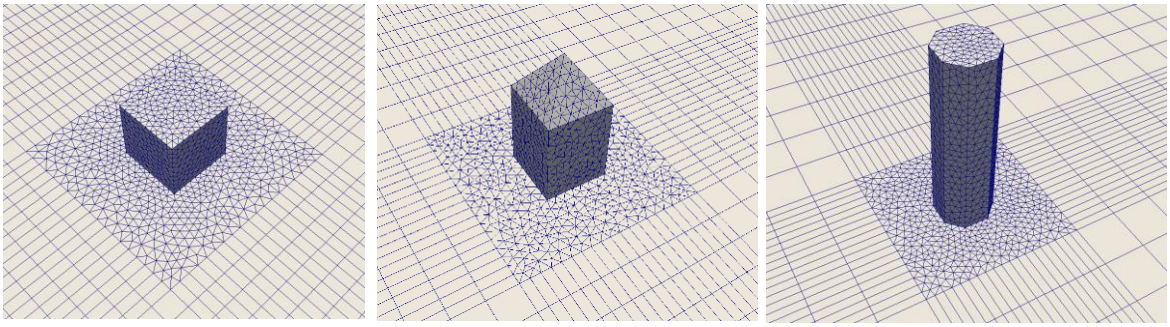


Figure 7 Hybrid mesh example for some primitive geometry

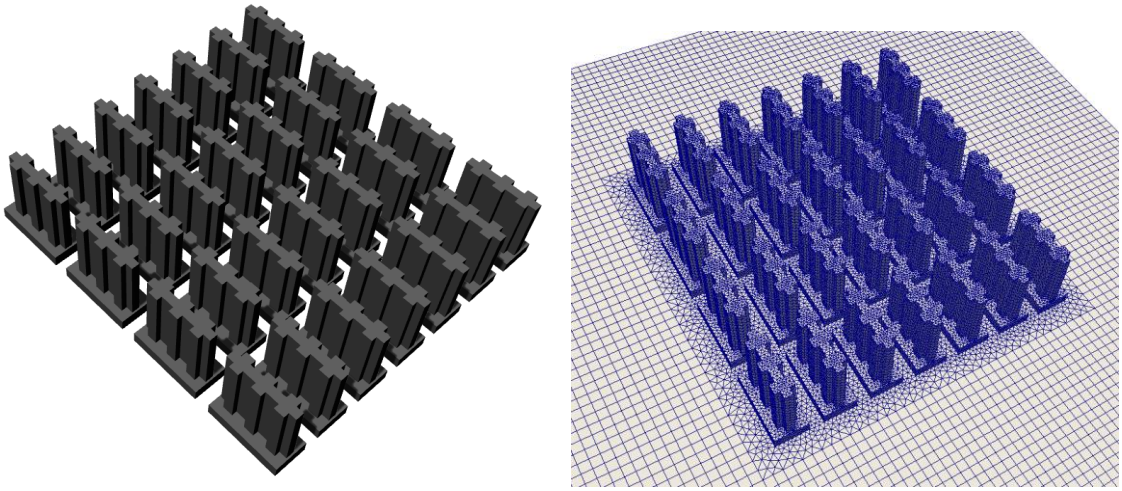


Figure 8 Hybrid mesh example for high-rise building clusters in urban-scale

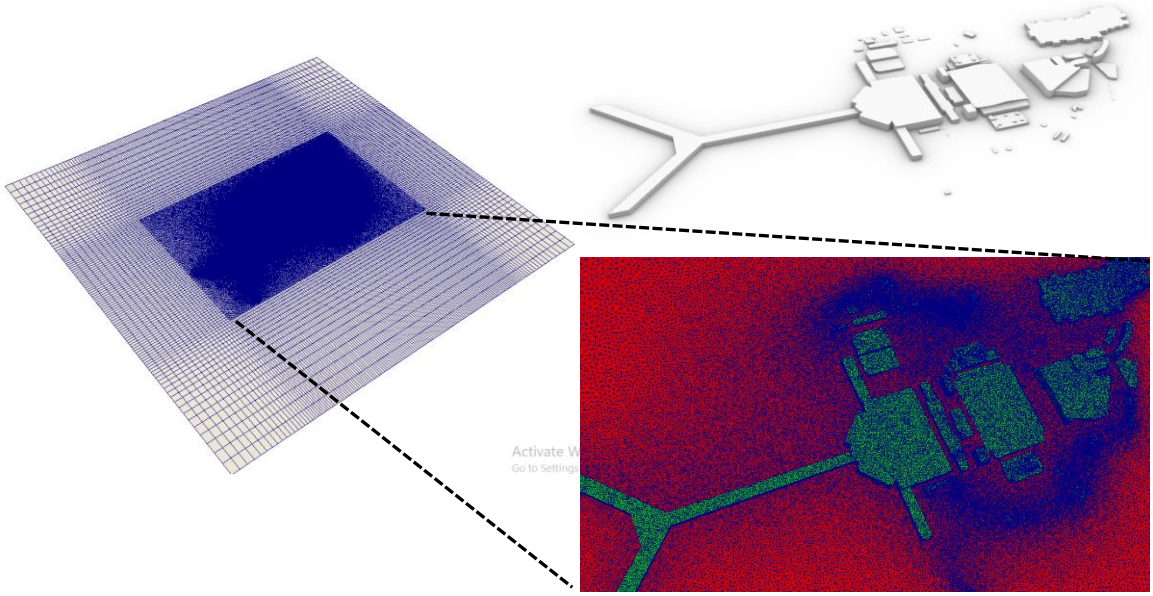
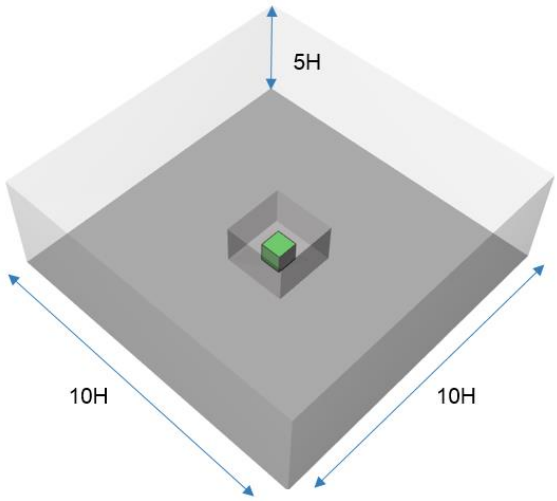


Figure 9 Hybrid mesh example for industry level practical model

3.2.4 Performance of the hybrid mesh scheme

In order to compare the performance of the traditional all tetrahedron unstructured mesh scheme and this hybrid mesh scheme proposed in this study, two case studies are carried out to count the total number of mesh elements and total computational time consumed using each mesh scheme. In order to have the comparable result, the mesh resolution are remained same for both scheme.

In first case, a 6m x 6m x 6m cubic building is used for mesh generation and CFD simulation. The parameters of the physical model and mesh size setting is illustrated in Figure 10. And a direct comparison between the two mesh schemes is shown Figure 11.



- Building geometry: 6m Cubic block
- Computational domain: 10 H * 10H * 5H (H is the height of the building, i.e. 6m)
- Mesh size resolution:

1) 15 * 15 * 15 for the building surface
2) 50 * 50 * 35 for the outer zone boundary surface

Figure 10 Computational domain of the first case study

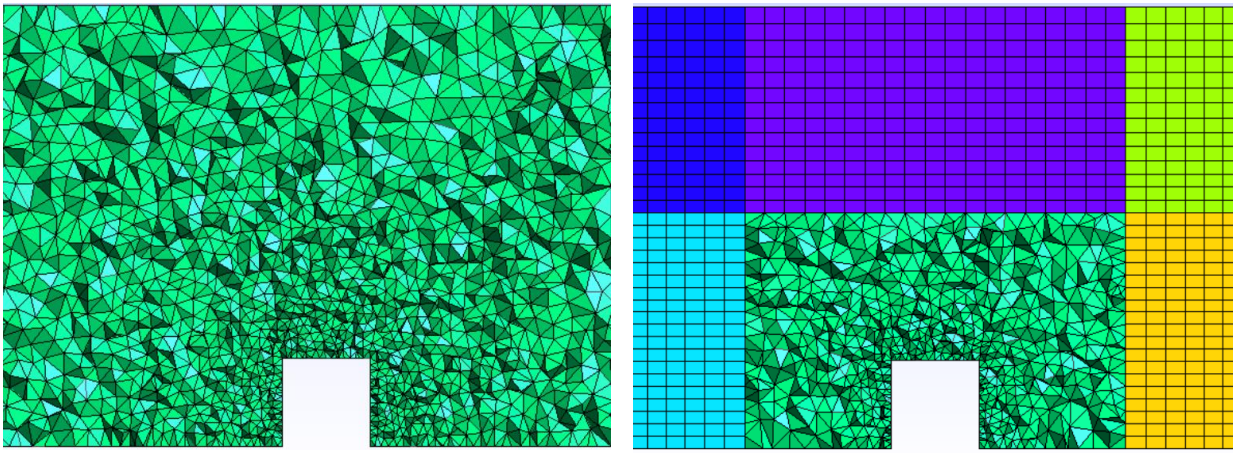


Figure 11 Comparison of the tetrahedron mesh scheme and the hybrid mesh scheme in the first case study

In the second case study, the high-rise building clusters are used for comparison and the demonstration of the mesh configuration of both scheme is shown in Figure 12.

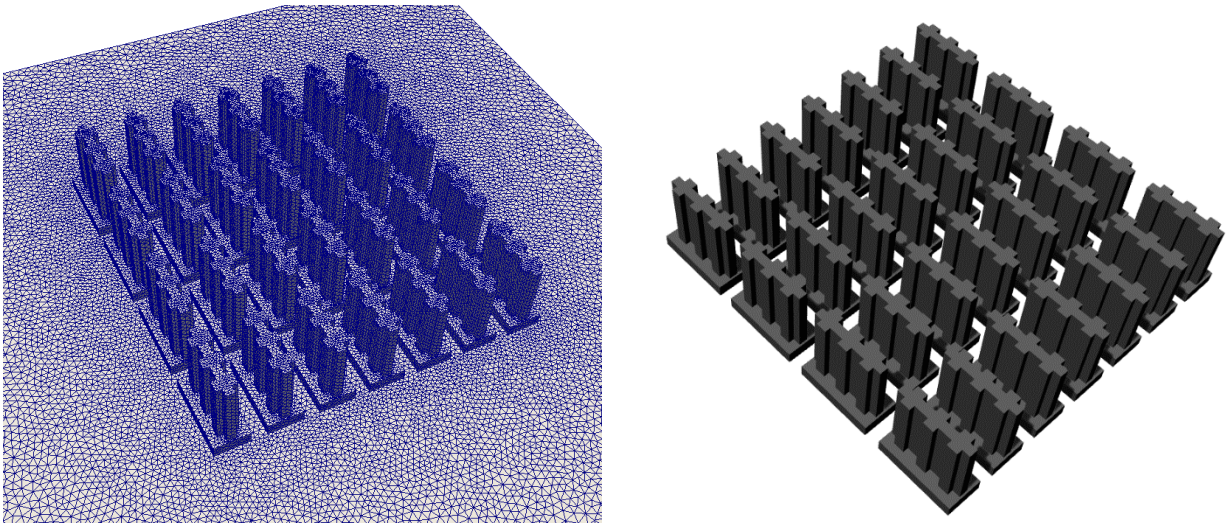


Figure 12 Comparison of the tetrahedron mesh scheme and the hybrid mesh scheme in the second case study

The CFD simulations were carried out using the two mesh schemes for these two case studies through CFD code named OpenFOAM 4.0. To achieve the comparable result, a single CPU core (Intel i5 5200u 2.2GHz) was used for computation until convergence (600 iteration steps for current case) and the same standard k-epsilon turbulence model with SIMPLE algorithm was selected. The performance of the two mesh schemes are summarized in the table below.

	Unstructured Mesh Scheme	Hybrid Mesh Scheme
Number of Tetrahedron	378,624	81,893
Number of Hexahedron	0	66,038
Number of Pyramid	0	1,805
Total Number of Mesh Element	378,624	149,736
Reduction of Mesh Elements	60.5%	
Total Computational Time	1,947 s (equivalent to 32 min)	403 s (equivalent to 6.7 min)
Reduction of Time	79.3%	

Table 2 Performance statistics for the first case study

	Unstructured Mesh Scheme	Hybrid Mesh Scheme
Number of Tetrahedron	2,166,591	445,709
Number of Hexahedron	0	196,403
Number of Pyramid	0	4,485
Total Number of Mesh Element	2,166,591	646,597
Reduction of Mesh Elements	70.2%	
Total Computational Time	13503 s (equivalent to 3.75 hr)	2865 s (equivalent to 0.79 hr)
Reduction of Time	78.8%	

Table 3 Performance statistics for the second case study

3.3 An adaptive octree-based 3D mesh generation algorithm

3.3.1 Problem formulation

With reference to the four characteristics of the urban-scale built environment on mesh configuration as stated in Section 3.2.2, the hybrid mesh strategy is proposed. Instead of using structured hexahedron in the outer zone, an alternative way is to adopt octree-based cartesian grid unit to fill the outer zone as shown in Figure 13. The previous method uses transfinite algorithm on the geometry edges first and then generates 3D hexahedron mesh. Though it is simple and straightforward, the drawback becomes obvious if there are multiple buildings in the computational domain because a great amount of sub-blocks have to be divided. Therefore, the octree-based mesh generation algorithm is provided to deal with this situation.

Comparing to the previous method, the octree-based mesh generation algorithm has the following advantages.

1. Sub-blocks are no longer prerequisite for mesh generation;
2. Each mesh element can be manipulated due to the tree data structure feature;
3. Each mesh element is dividable.

Because of the above features, it can handle the multiple buildings case simply by refining the mesh elements near the buildings region and then removing the mesh elements occupied by the buildings regions.

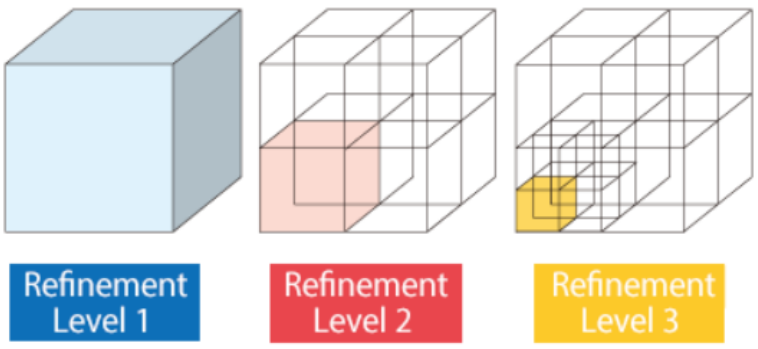


Figure 13 The Octree-based mesh structure

3.3.2 Octree data structure

The octree-based mesh generation algorithm is implemented by a specific tree data structure, called octree data structure as shown in Figure 14. The octree is initiated by a root node first. And one node can be linked to 1 parent node and 8 sub-nodes. The node with sub-nodes is called parent node and the sub-node is called child node. The top node without parent node is called root node while the one without sub-nodes is called leaf node. The root node layer is defined as level 0 and the node level increases by one at each sub-nodes layer. Each leaf node would present one mesh element and the size of mesh element depends on the node level of the leaf node.

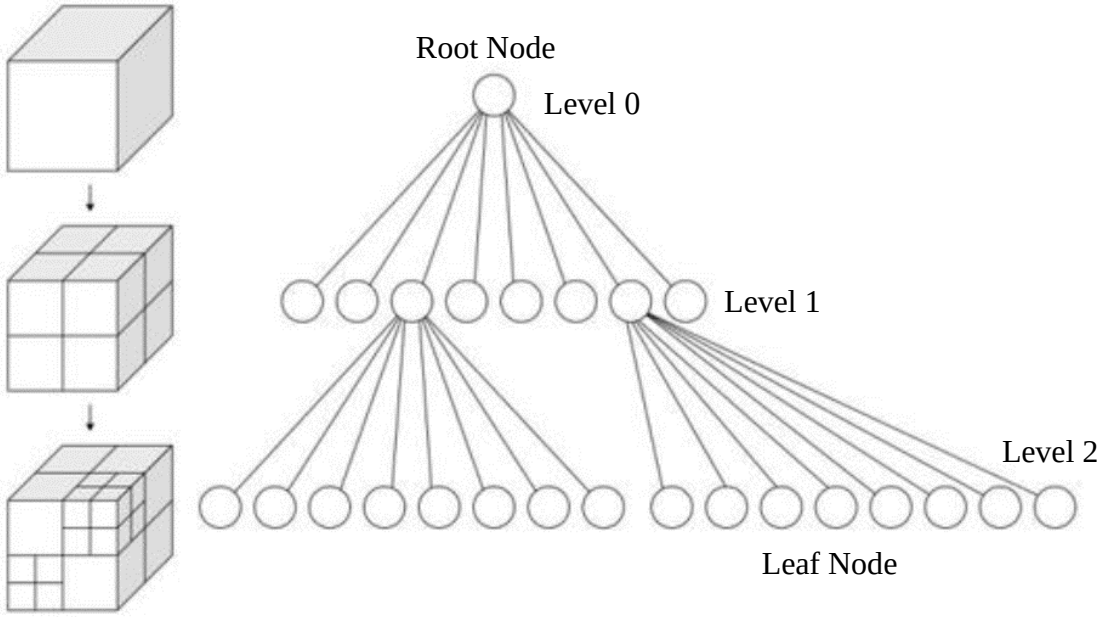


Figure 14 A diagram for the octree data structure

As the number of mesh elements for CFD application on urban-scale built environment is normally ranged from million to hundreds of millions magnitude, it requires that the program execution efficiency for mesh generation is of importance. Considering the full object oriented capability, high performance and extensive application libraries, the C++ programming language is suitable for the mesh generation algorithm and thereby used in the study.

The C++ classes as illustrated in Figure 15 and Figure 16 demonstrate the basic fields and methods of octree data structure. To realize the nodes in the tree structure, the OctNode class is developed

with two fields, *OctNode* parent* and *vector<OctNode*> children* to setup the linkage relationship. The whole tree is implemented by the Octree class with a root node (OctNode* root).

```
Class OctNode
{
string nid;
vector<double> center;
int level;
OctNode* parent;
vector<OctNode*> children;
…
void addChildNode(string _nid, int _level, vector<double> _center);
…
};
```

```
Class Octree
{
OctNode* root;
…
void define_root();
void splitNode (OctNode* node);
…
};
```

Figure 15 The Octree data structure by C++ class

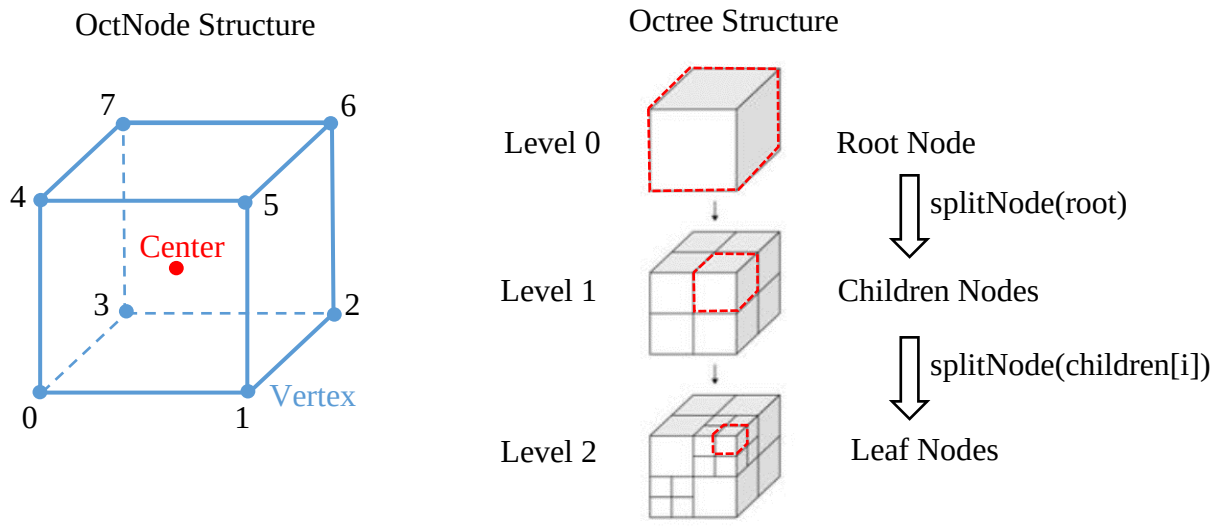


Figure 16 The Octree data structure diagram

3.3.3 Algorithm design and implementation

The algorithm consists of several key components and the framework is show in Figure 17. An octree mesh generation application could be designed with three main steps in general:

1. read an input geometry;
2. generate octree date structure;
3. convert to a specific CFD mesh format and save the mesh.

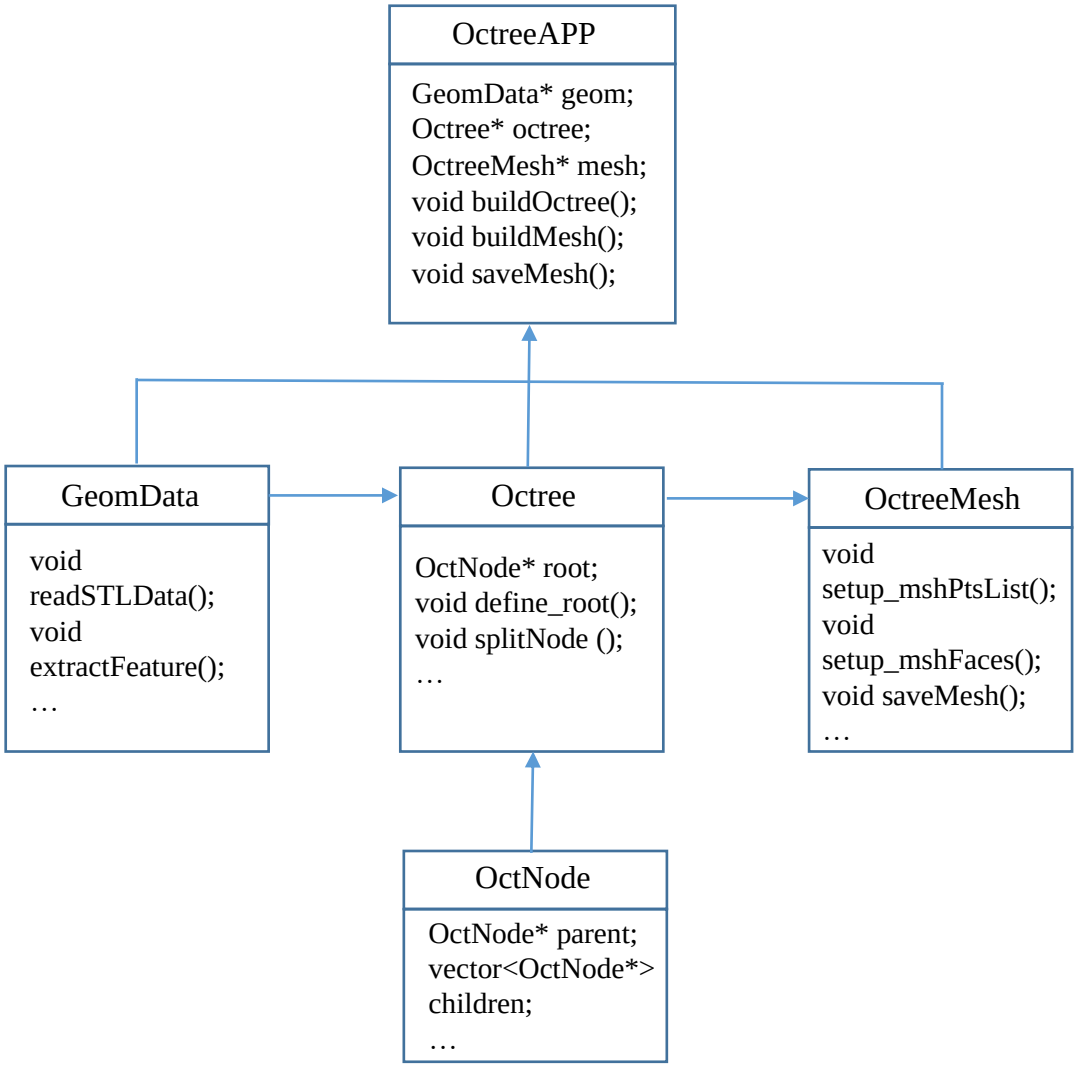


Figure 17 The algorithm design framework

In this study, the C++ class of OctreeAPP is defined to realize the algorithm in the top level underpinning by class GeomData, Octree and OctreeMesh in the second layer. The data flow process is started from GeomData then passed to Octree and eventually OctreeMesh. The main functions of each class are explained in the table below.

C++ Class	Function
OctreeAPP	Initiate the mesh generation application and control the operation process
GeomData	Read, process and store the input geometry data, e.g. the list of points and triangles from the STL file
Octree	Initiate root node and split nodes to generate required octree data structure
OctreeMesh	Convert the octree data structure to a specific CFD mesh format and output the mesh data for a CFD solver.

Keeping Octree with 2:1 Balance

The octree structure with 2:1 size balance is defined as the node level difference between the focused node and its neighboring nodes is less than 2. As shown in Figure 18, there are two kinds of 2:1 balance condition in 2D case. The neighboring nodes with a shared face is split to become 2:1 balanced under the face balance condition while the neighboring nodes with a shared corner would also be divided under the corner balance condition [21]. Applying the similar concept to 3D case, the number of neighboring nodes is 6 for face balance, 18 for edge balance and 26 for corner balance as shown in Figure 19.

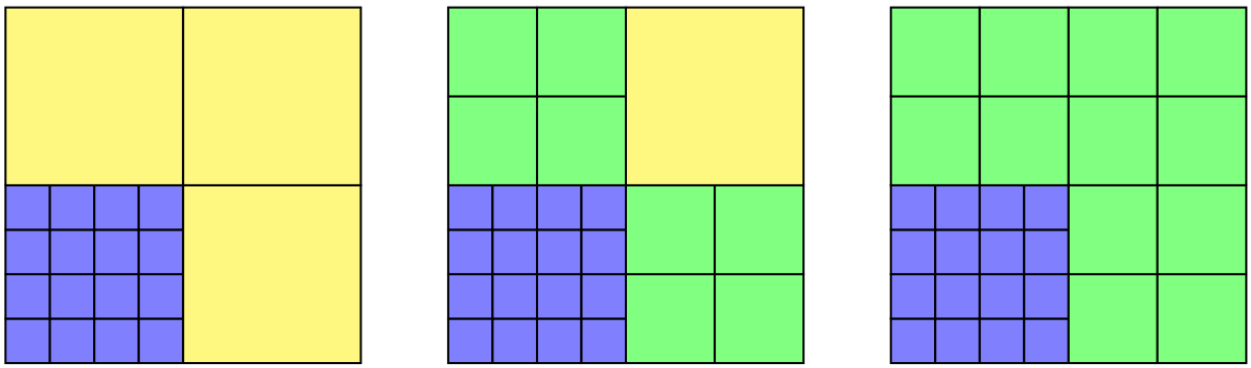
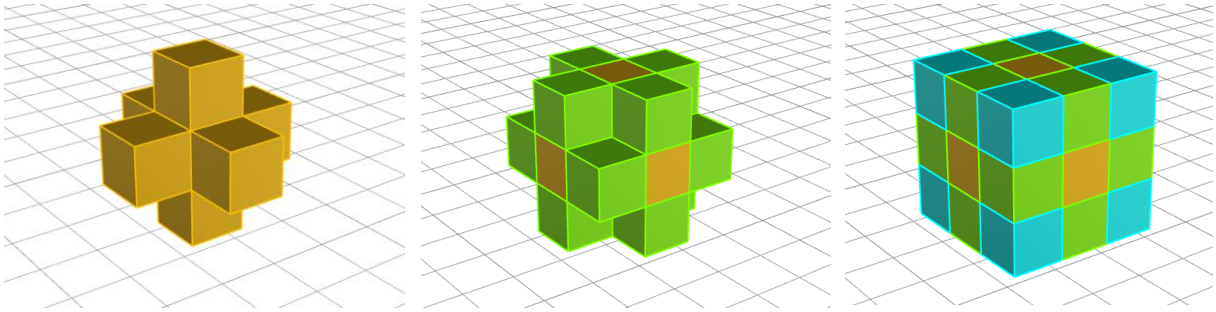


Figure 18 Examples of 2:1 balance in the 2D case



Face balanced Edge balanced Corner balanced

Figure 19 Examples of 2:1 balance in the 3D case

Balance Type	2D Case	3D Case
Face balanced	4 neighboring nodes involved	6 neighboring nodes involved
Edge balanced	N/A	18 neighboring nodes involved
Corner balanced	8 neighboring nodes involved	26 neighboring nodes involved

Ensuring the 2:1 size balanced octree is important to obtain a high-quality mesh for numerical computation. From face balance to corner balance, the smoothness of the mesh becomes better meanwhile the number of mesh elements soars. In this study, the edge balance is considered appropriate in terms of both mesh quality and total mesh amount. Thereby, this is adopted for a 3D mesh generation case.

Mesh density definition

The core of the algorithm is to build up an octree-based 3D mesh automatically by dividing root node (representing the 3D computational space) recursively into 8 octants until the defined mesh density is achieved. To control the mesh density in the octree, there are three types of sizing functions.

- Split node by minimum level size: the node will be split recursively until the defined minimum level size.

- Split node by geometry feature: the node will be split recursively until the number of geometry features in the node is not more than the defined size or the defined maximum level size is achieved.
- Split node by custom defined region: the nodes within the pre-defined region, e.g. a cube, will be split recursively until the defined node level size or the defined maximum level size is achieved.

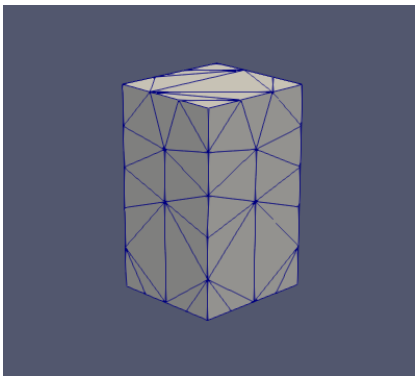
Octree generation procedure

The generation of the octree data structure works in the following steps.

1. Define mesh density, including the minimum node level, maximum node level, maximum number of geometry features in a node and the node level in the custom defined region;
2. Define a root node which covers the whole computational domain;
3. Divide the node until fulfilling the mesh density achieved meanwhile keep the octree balanced;
4. Exclude the exterior nodes which are occupied by the building regions;

3.3.4 Case study

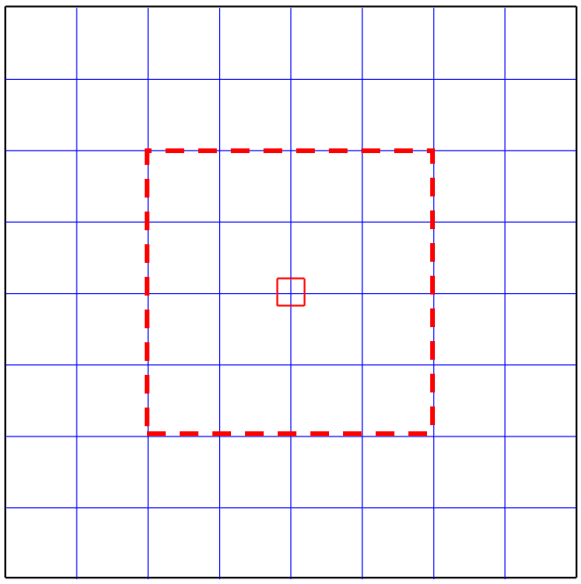
A STL format data is used as the geometry input to give an example to demonstrate how the octree-based mesh generation algorithm works, The STL geometry data represents the geometry surface information by a series of triangles as shown in Figure 20. The "GeomData" Class provides functionalities to read, process and store the geometry data as a list of vertices and a list of triangles.



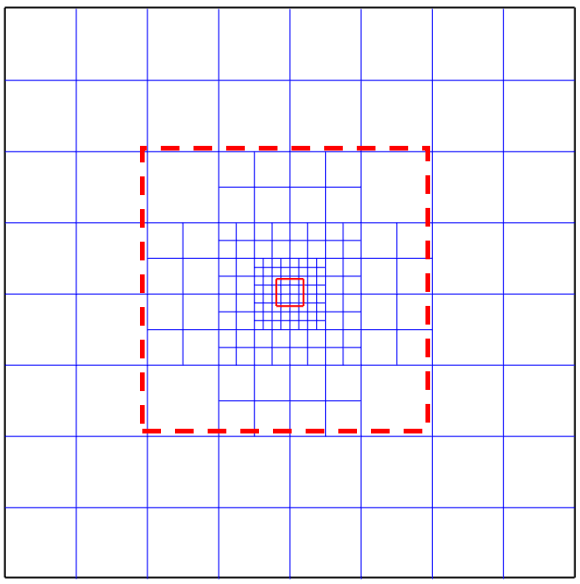
```
solid Bldgs
  facet normal -0 1 0
    outer loop
      vertex 7.1389145851135254 4.4308819770812988 0
      vertex 8.1256046295166016 4.4308819770812988 1.2999999523162842
      vertex 8.1256046295166016 4.4308819770812988 0
    endloop
  endfacet
  facet normal -0 1 0
    outer loop
      vertex 8.1256046295166016 4.4308819770812988 1.2999999523162842
      vertex 7.1389145851135254 4.4308819770812988 0
      vertex 6.1522250175476074 4.4308819770812988 0
    endloop
  endfacet
  facet normal -0 1 -0
    outer loop
      vertex 6.1522250175476074 4.4308819770812988 3.25
      vertex 8.1256046295166016 4.4308819770812988 2.5999999046325684
      vertex 8.1256046295166016 4.4308819770812988 1.2999999523162842
    endloop
  endfacet
  facet normal 0 1 -0
    outer loop
      vertex 6.1522250175476074 4.4308819770812988 3.25
      vertex 8.1256046295166016 4.4308819770812988 3.9000000953674316
      vertex 8.1256046295166016 4.4308819770812988 2.5999999046325684
    endloop
  endfacet
  facet normal 0 1 -0
    outer loop
      vertex 8.1256046295166016 4.4308819770812988 6.5
      vertex 8.1256046295166016 4.4308819770812988 5.1999998092651367
      vertex 6.8100180625915527 4.4308819770812988 6.5
    endloop
  endfacet
```

Figure 20 Geometry information represented by STL format

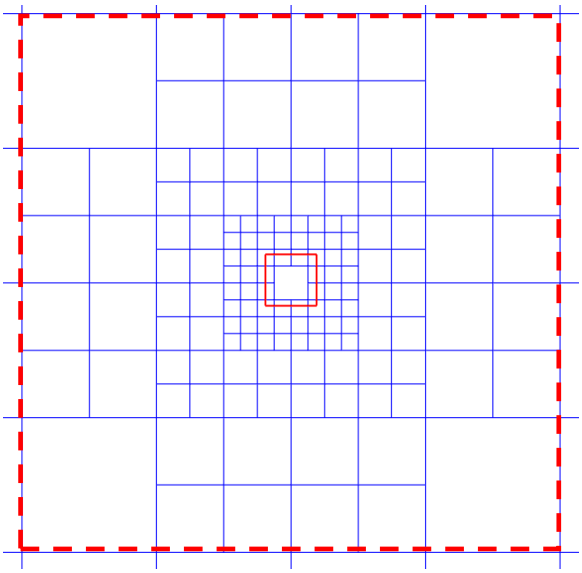
In this case, the minimum node level is defined as 3 while the maximum is 6. And the maximum number of geometry feature points allowed in a node is 1. Next, the generation of octree data structure could be started based on the defined mesh density and the generation process is as shown in Figure 21.



a) Root node defined first to cover the whole domain and then split node until reaching the minimum node level.



b) Split node adaptively to capture the geometry feature meanwhile keeping the octree balanced



c) Exclude nodes within the geometry from the octree.

Figure 21 The octree-based mesh generation process

After generation of the octree, the final step is to convert the octree data into a specific CFD mesh format and save the mesh. In this practice, the OpenFOAM mesh format [22] is used, in which the mesh is constructed by a list of points, faces, owners, neighbors and boundaries as shown in Figure 22.

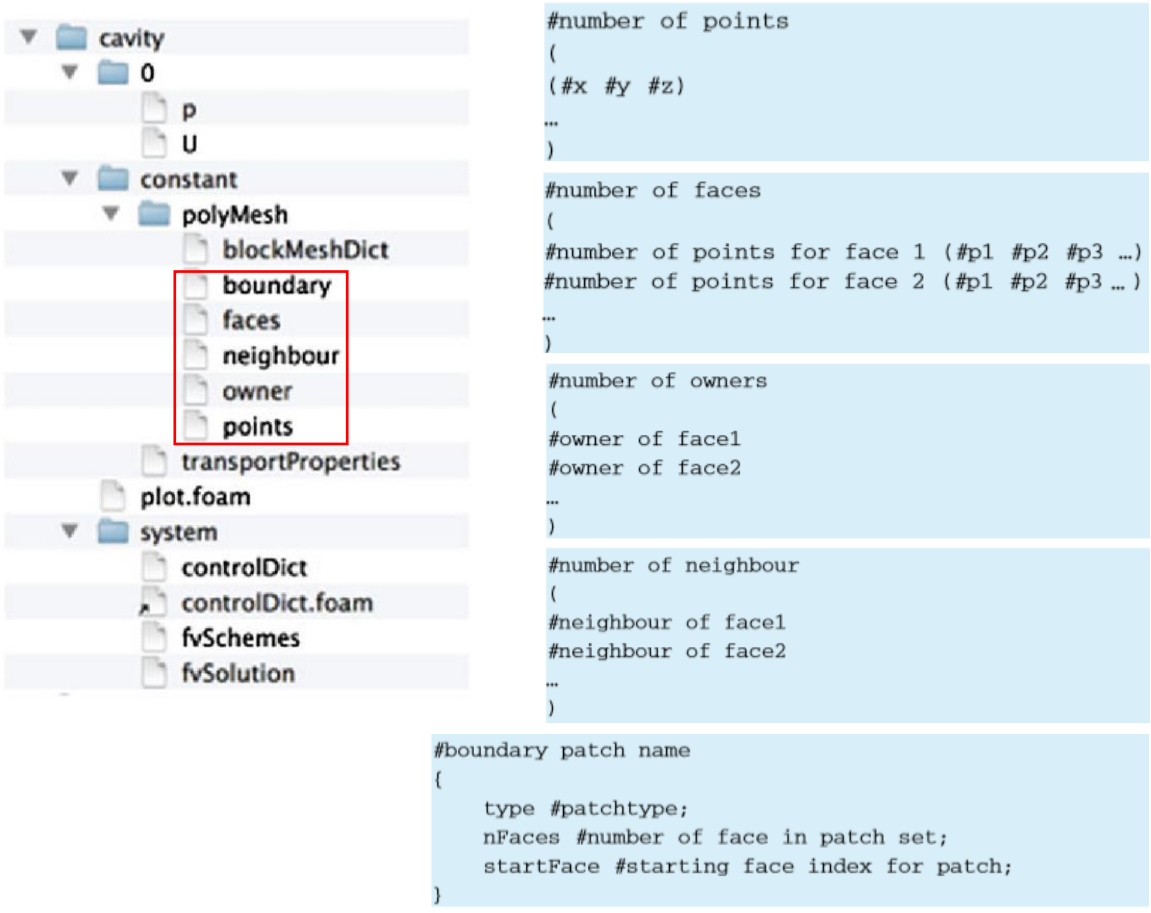
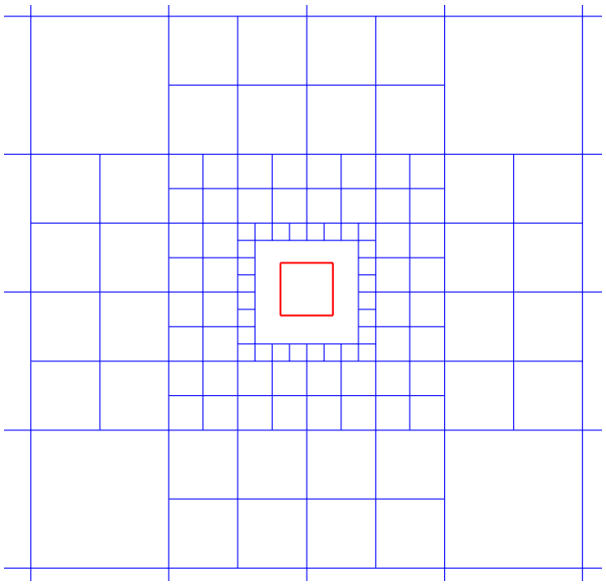


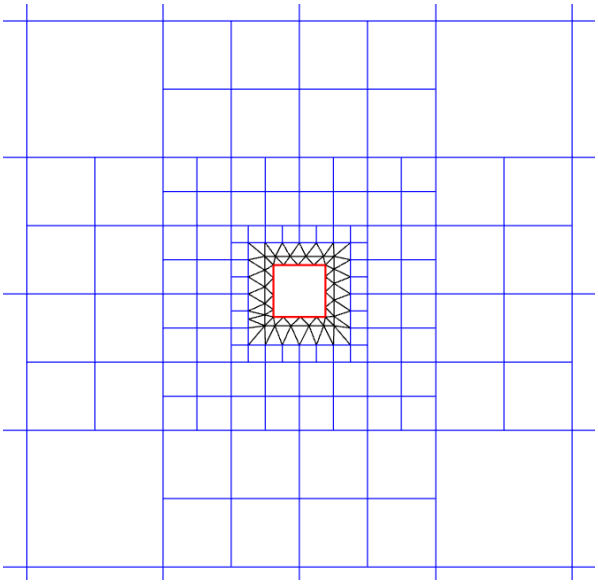
Figure 22 The files for the OpenFOAM mesh format [23]

3.3.5 The way forward

To complete the algorithm, the next step is to remove the leaf nodes near the building region. The node removal function is implemented by defining a buffer zone offset from each building surface and remove the nodes that are within the buffer zone. Finally, unstructured mesh would be used to fit the geometry surface and fill the buffer zone as shown in Figure 23.



d) Delete nodes within the buffer zone offset from the geometry surfaces.



e) Reconstruct with tetrahedral mesh element

Figure 23 The last two steps to be implemented in the mesh generation process

Eventually, the hybrid mesh strategy stated in the last section is improved and generalized in terms of the overall mesh efficiency and flexibility by using the octree-based mesh generation algorithm.

3.4 Error analysis of the decoupled and coupled CFD application

3.4.1 Problem formulation

In the engineering practice, the indoor CFD air flow simulation can be carried out in an independent way by setting the pressure values for each opening area as the boundary conditions, ignoring the external built environment. However, to impose correct boundary conditions for the independent indoor CFD simulation, i.e. without consideration of the external wind simulation, can be a difficult engineering judgment in the design practice, especially when there are complex surrounding obstructions which is common in the urban area [3]. A good approach to deal with this problem is to coupling the external and internal physical model together in the simulation but it would definitely result in a huge amount of grids and prolong the computation time dramatically as the indoor environment requires much smaller mesh size than the outdoor environments. Another solution is to conduct the outdoor only CFD simulation to achieve the pressure field data along the building envelop first and then the boundary condition data required by the indoor CFD simulation would be ready. Hence this approach is done in the decoupled way. But the accuracy of this decoupled method is still questionable.

Therefore, in this section, a case study would be carried out particularly to investigate the accuracy of decoupled simulations for prediction of the indoor air flow using the coupled simulation as the reference case. The working procedure is that external CFD simulation on urban scale would be carried out first and then provide the boundary information required by the indoor CFD simulation, e.g. the pressure distribution on the building envelop. At last, each indoor zone within the building can be simulated independently.

3.4.2 Case study

1) Model setup

In this case study, a 6m x 6m x 6m cubic building with two openings is studied as the full-scare measurement data is presented in the literature which can be used as a comparison with this case study. The size of the openings is same with 0.35m x 0.25m. Opening 1 is located 1m above the ground level while Opening 2 is located 1m below the roof respectively. The wind direction considered in this study is from South which means wind blows perpendicularly to the cube surface through the lower level opening. The computational domain size was set as 5H upstream, 10H

downstream and 5H away from each side of the cube and 5H above the cube. Structured mesh as shown in the figure below is used for simulation. Cells were distributed with a grid expansion ratio of 1.05 from the cube surfaces to the outer surfaces of the domain and the region near cube surfaces and openings was further refined. A more detailed information about the mesh would be elaborated in the next paragraph.

The CFD solution was obtained with OpenFOAM 4.0 in this study, which is a general-purpose finite volume method based CFD code supporting both structured and unstructured grids. The turbulence model adopted is standard k – epsilon with several upwind based discretization schemes for the convection term. SIMPLE algorithm with GAMG solver is adopted for the discretized algebraic equations.

Boundary conditions for this case include inlet, outlet, ground, cube surfaces, upper domain surface and side domain surface. For the inlet boundary, log-law velocity profile is used while zero gradient is used for the outlet boundary. Ground and cube surfaces are set as no-slip wall. Symmetry boundary condition are set for the upper and side domain.

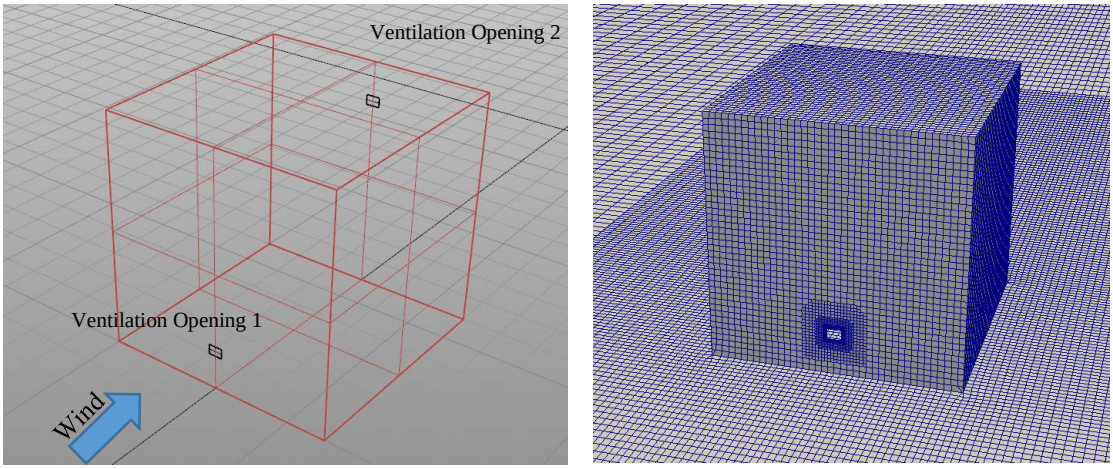


Figure 24 physical model setup for the cube with two openings

2) Grid sensitivity test

Mesh size has a critical influence on the accuracy of CFD solution. Too coarse grids would produce inaccurate results while too fined grids consume a lot of computation resources and require much longer time to a converged results. In order to find the appropriate grid configuration, grid sensitivity test was carried out and the case setup is shown in the table below. To make fully use

of the limited computation resources, finer meshes are distributed near the cube surface and the openings while the mesh size for the region far away from the cube becomes larger gradually with an expansion ratio of 1.05.

Mesh Name	Mesh Length Scale (m)	Resolution on Cube Surface	Resolution at Openings	Total Number of Cells
Grid 30	0.2	30	14 x 10	733,000
Grid 40	0.15	40	18 x 14	1,435,000
Grid 60	0.1	60	28 x 20	3,457,000

Table 4 A summary of the mesh configurations used in this study

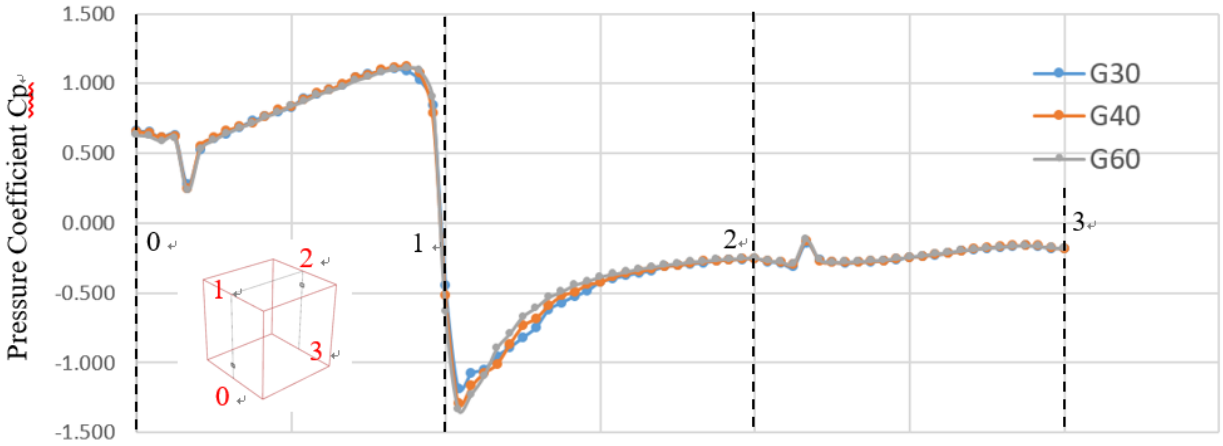


Figure 25 Pressure coefficient distribution along the cube surfaces

The figure above shows the pressure coefficient along the line 0-1-2-3 with a comparison of the three mesh configurations. It is noted that the three mesh schemes have the similar trend and all of them is able to reproduce the characteristics of the pressure coefficient distribution on the building envelop. It is also observed that agreement are achieved along line 0-1 and line 2-3 while larger discrepancy is appeared along line 1-2. The reason for this problem is relevant to the stronger turbulence intensity on the roof top of the cube. Using the Grid 60 as the reference case, Grid 30 has an overall Mean Relative Error (MRE) of 5.7% for all the testpoints of pressure coefficient while the value for Grid 40 is only 3.4%. On the other hand, for the pressure coefficient at the openings, Grid 30 has a larger error with MRE of 18.7% for Opening 1 and 22.5% for Opening 2 whereas the values of Grid 40 are 7.6% and 5.8%. Therefore, by balancing speed of simulation

and accuracy of solution, Grid 40 is adopted for further study. The data used for calculation of the pressure coefficient is provided in Appendix C.

3) Possible Results

As the case study is still in progress, there is no clear result and conclusion are produced yet. The possible results might cover the field information for the pressure, velocity etc from the coupled and decoupled simulation. And the accuracy of method used in this study would be compared with other methods from envelop flow theory.

4 Machine Learning Algorithm based on CFD Simulation Data

4.1 Machine learning methodology

The mechanism of this new proposed CFD plus machine learning approach can be explained in three steps. Firstly, higher-fidelity CFD simulation results should be collected which should cover a wide range of environmental conditions and building geometries. The results will be used as the dataset for training and validating the machine learning algorithm. Secondly, the machine learning algorithm should be trained until finding the optimal coefficients which makes the lost function has the lowest error value. Thirdly, the trained algorithm can be used to predict value in a much shorter time when receiving a new input data without carrying out additional CFD simulation.

Using the methodology of CFD theory, the simulation results are available, which would be used as the dataset for machine learning algorithm. More precisely, the regression algorithm would be adopted in this study to make prediction. In regression algorithm, a hypothesis function is required to be formulated with respect to some parameters, that is able to give an output when feeding an input. And the goal is to find the parameters that would minimize the difference between the output value and the true value using a large amount of dataset. Several regression algorithms would be used in this study. They are least-squares (LS), regularized LS (RLS), L1-regularized LS (LASSO), robust regression (RR) respectively [3]. The working mechanism of these algorithms are similar, thereby, the LS regression algorithm would be taken as an example to demonstrate how it working mechanism. In this algorithm, the polynomial function is adopted as the hypothesis function to predict the true value.

$$f(x,\theta) = \sum_{k=0}^{K} x^k \theta_k = \phi(x)^T \theta$$

where $(x) = [x^0, \dots, x^K]^T$, $\theta = [\theta_0, \dots, \theta_K]^T$.

Our goal is to obtain the best estimation of the hypothesis function given independently identically distributed dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$.

The derivation process for calculating the best estimated parameter $\hat{\theta}$ is shown below.

We want to find the optimal parameter that can minimize the error between the true value and the output value of the function.

$$\hat{\theta} = \min_{\theta} \sum_{i=1}^{N} [y_i - f(x_i)]^2$$

Rewrite it in a matrix form,

$$\hat{\theta} = \min_{\theta} \|y - \Phi^T \theta\|^2 = \min_{\theta} y^T y - 2y^T \Phi^T \theta + (\Phi^T \theta)^T (\Phi^T \theta)$$

where $y = [y_1, \dots, y_n]^T$, $\Phi = [\phi(x_1), \dots, \phi(x_n)]$.

To get the optimal result of the parameter, set the first order derivative to zero,

$$\frac{\partial(\cdot)}{\partial \theta} = 0$$

We can calculate that,

$$\hat{\theta} = (\Phi\Phi^T)^{-1}\Phi y$$

After we have the parameter, we can use this trained machine learning algorithm to predict the true value for new input,

$$f(x_*) = \phi(x_*)^T \hat{\theta}$$

There are some other algorithms with different objective function to give additional features, like robustness to outliers in the data set. And the results of these algorithms are summarized below. For more details on the implement of these algorithms, please refer to Appendix B.

Method	Objective function	Parameter estimate $\hat{\theta}$	Prediction for input x_*
Least-squares (LS)	$\|y - \Phi^T\theta\|^2$	$\hat{\theta}_{LS} = (\Phi\Phi^T)^{-1}\Phi y$	$f(x_*) = \phi(x_*)^T\hat{\theta}$
Regularized LS (RLS)	$\|y - \Phi^T\theta\|^2 + \lambda\|\theta\|^2$	$\hat{\theta}_{RLS} = (\Phi\Phi^T + \lambda I)^{-1}\Phi y$	$f(x_*) = \phi(x_*)^T\hat{\theta}$
L1-regularized LS (LASSO)	$\|y - \Phi^T\theta\|^2 + \lambda\|\theta\|_1$	QP solver	$f(x_*) = \phi(x_*)^T\hat{\theta}$
Robust regression (RR)	$\|y - \Phi^T\theta\|_1$	LP solver	$f(x_*) = \phi(x_*)^T\hat{\theta}$

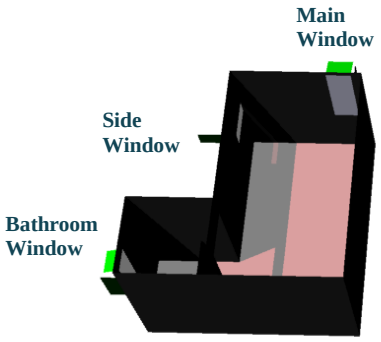
Table 5 A summary of the regression algorithms used in this study

4.2 Case study

4.2.1 Model setup

A simple residential flat consisting of a living room and bathroom with three windows in total, is selected in the case study. There are five input variables in the dateset, and the first two input variable represent the design parameters enabling variation of window size while the last three input variables represent the environmental parameters (static pressure values in this case). The value range for each input variable used in this study is illustrated in the table below.

Input variable	Value range
X1	0.45 ~ 0.65 m2
X2	0.21 ~ 0.35 m2
X3	0 ~ 0.4 Pa
X4	0 ~ 0.4 Pa
X5	0 ~ 0.4 Pa



In order to cover a wide range of conditions, 3 different values are taken for Main Window Area (X1=0.45, 0.55 and 0.65), 3 different values for Side Window Area (X2=0.21, 0.28 and 0.35), and 5 different values for each pressure location (0, 0.1, 0.2, 0.3 and 0.4). After the combinations of these 5 input variables, 540 different combinations are produced by reducing the repeated cases, for example X3=0, X4=0, X5=0 is same as X3=0.1, X4=0.1, X5=0.1. The output variable in this case study is Air Change Rate for the flat (ACH), used as the ventilation performance indicator.

Input/Output variable	Description	Number of possible values
X1	Main Window Area	3
X2	Side Window Area	3
X3	Pressure value for Main Window	5
X4	Pressure value for Side Window	5
X5	Pressure value for Bathroom Window	5
Y	ACH of the flat	1125 – 585(repeated) = 540

Table 6 the number of combinations of the five input variables and its corresponding output

Therefore, 540 nos CFD simulations, using k-epsilon turbulence model, are carried out eventually to calculate the ACH of the flat. 70% of the dataset would be used as training dataset while 30% of them would be used as validation dataset.

4.2.2 Result discussion for case study

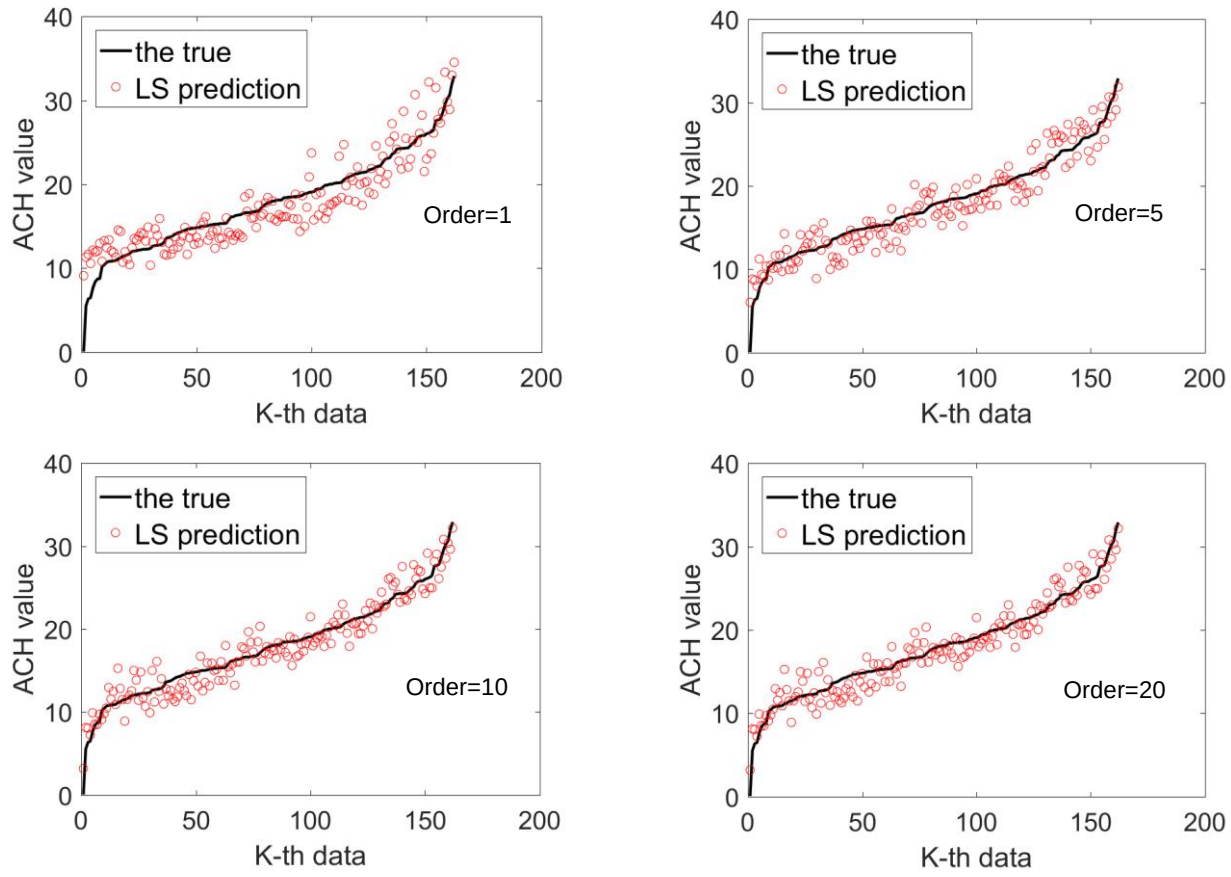


Figure 26 Prediction from the LS algorithm with different order number against the true value in the dataset

Order	Mean Absolute Error
1	1.92
5	1.53
10	1.27
20	1.27

Table 7 A summary of the mean absolute error with the increase of order number in LS algorithm

The above figure 4 shows the results of predicted ACH by Least Squares (LS) Method and true ACH by CFD simulation. Four different orders for the polynomial function are investigated, from one to twenty to improve the accuracy of the algorithm. Based on the observation of the results, no matter what order is selected, all of them is able to predict the overall trend in the dataset correctly with the Mean Absolute Error (MAE) ranging from 1.27 to 1.92. And when comparing the MAE, higher order function can give more accurate prediction and when the order number increases to ten, the accuracy remains stable, which illustrates order of ten is sufficient to represent the complexity of the input and output relationship. Therefore, the order of ten is kept for the other regression algorithms.

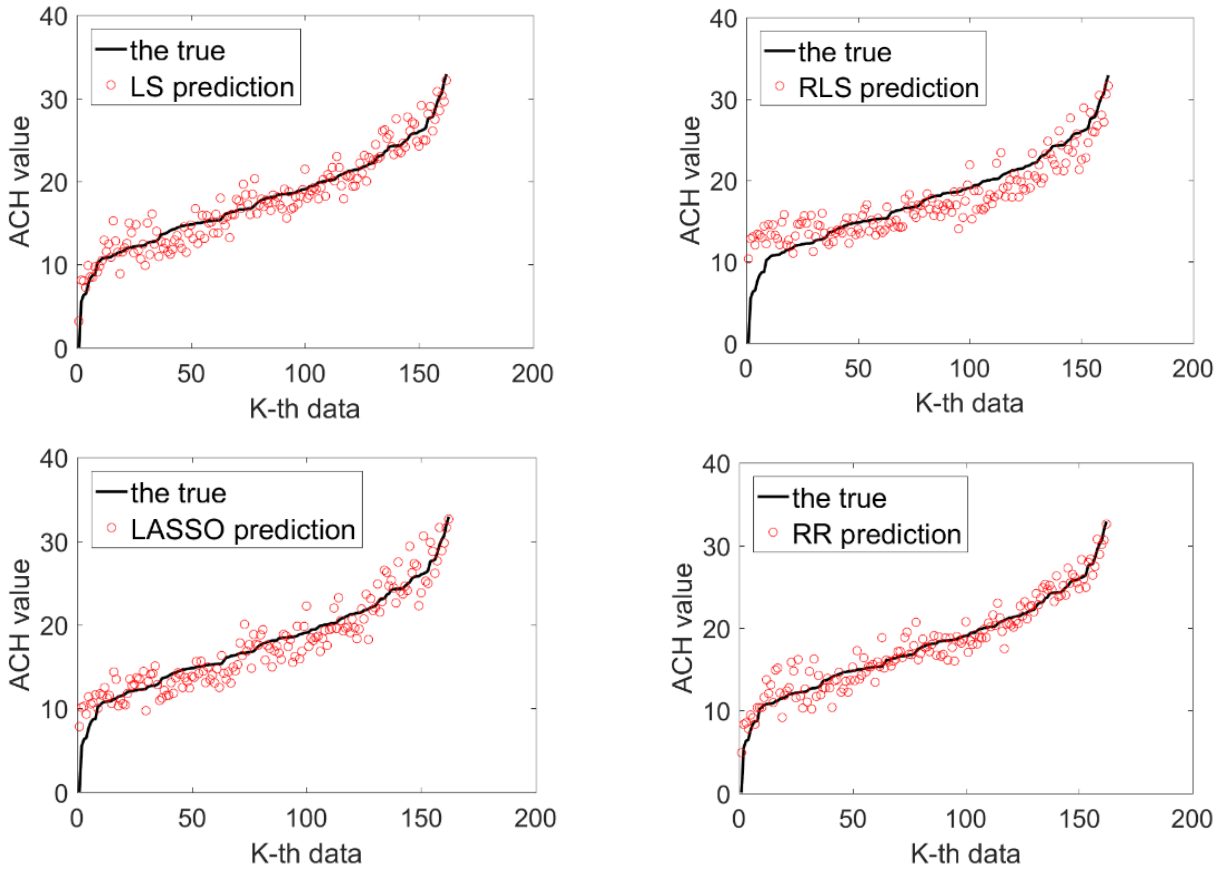


Figure 27 Prediction from the four algorithms against the true value in the dataset

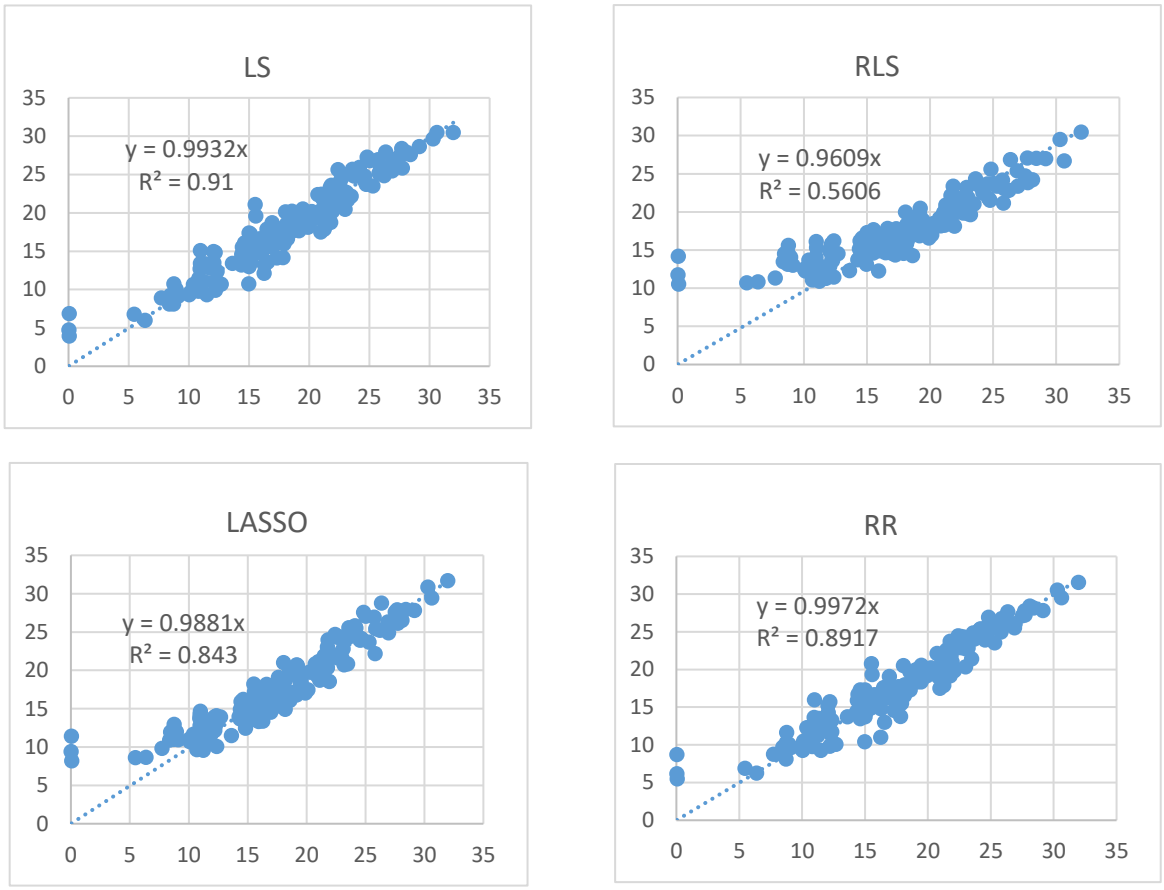


Figure 28 Results of correction relationship between the prediction and the true value in the dataset

Algorithm	MAE	MRE
LS prediction	1.27	8.00%
RLS prediction	1.95	13.12%
LASSO prediction	1.55	9.88%
RR prediction	1.16	7.62%

Table 8 A summary of the mean absolute/relative error from four algorithms

Four different regression algorithms are used to predict the ACH value. It is noted that RR method has the highest accuracy with MAE of 1.16 and MRE of 7.62%. In terms of the correlation relationship, LS method gets the highest with R^2 of 0.91. On the other hand, all regression methods have very large errors for the data in the beginning part of the x-axis. The reason might be related to the polynomial function assumption and more insight investigation would be done in the future study.

4.2.3 Conclusion for case study

The case study has proposed a new approach to estimate the ventilation performance of a flat by using machine learning algorithms. The result of this study support the feasibility of using machine learning tools to estimate the building parameters. The polynomial regression algorithms used in this study is able to make a fast prediction of the ACH with reasonable error. The model developed in this study can also be further developed as a parametric design tool to optimize the ventilation performance by changing the values of the input variables.

5 Integrated Software Design for CFD Application Implementation

5.1 Problem formulation

It has been demonstrated by researchers that CFD is powerful in simulating air flow and thermal processes in built environment for building design optimization. However, CFD simulation is an expensive tool with steep learning curves in terms of not only the CFD theory but also the complex application procedure. Lack of interoperability between CFD simulation tools and architectural CAD tools makes it an expensive practice. Hence, to solve this problem, an integrated software design to implement the CFD application on urban-scale level built environment simulation would be discussed in this section.

5.2 Framework of the software design

In order to have an integrated working environment for CFD application, the software framework would include the four main components, namely geometry component, mesh component, solver component and post-processing component respectively. And the software would play as the platform role to provide connectivity among these components.

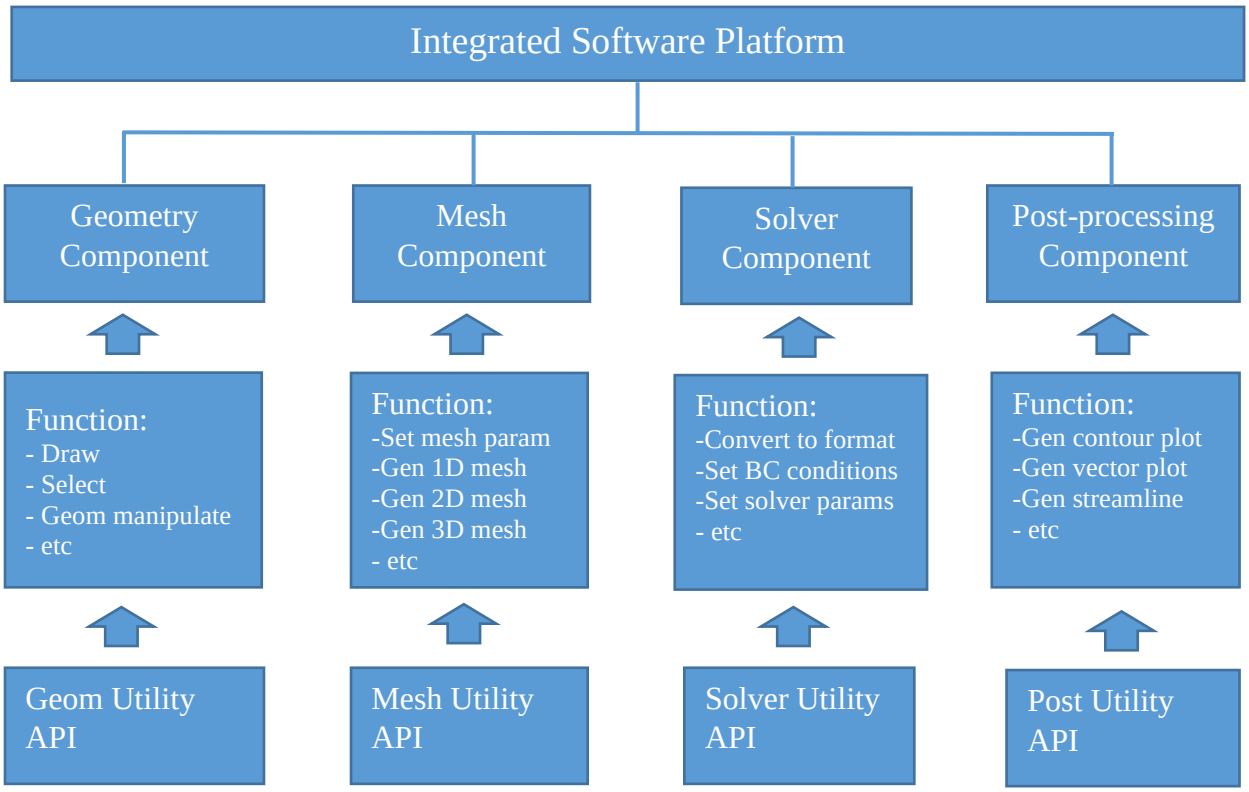


Figure 29 An overview of the software framework

The C++ is selected as the programming language to implement of the software design as there are abundant open source libraries written in C++ which support the development this software. Firstly, Qt is used as the GUI platform to design the software layout, window and all kinds of widgets. The OpenCascade would provide the fundamental geometry creation and modelling algorithm for the geometry component design. Gmsh would provide the meshing functionality for the mesh component design. OpenFOAM would be the CFD code to run the actual simulation for the solver component and finally post-processing of the CFD result data would be dealt with by ParaView.

5.3 Feature of the software

The purpose of development of software is to make the CFD simulation process more automatic and efficient. Hence, the software would be able to have the following features.

Interoperability

In the traditional working process, modelling, meshing, solving and post-processing are carried out by different software tools which increase the complexity and require much human involvement. Hence, the interoperability among the CFD working steps is a main feature in this software design which is realized through an integrated operating environment.

Interactivity

Interactive operating experience for users during the mesh size parameter setting is of importance. It is a good feature to immediately reflect the parameter effect that users set on points, lines, surfaces etc to have an intuitive visual perception. The ultimate goal is to the easy and handy means to have the high-quality mesh generation. Another aspect of the interactivity is to allow users setup boundary condition for CFD solver by selecting corresponding surfaces in the GUI environment instead of editing in the script context.

Adaptability

This software would have a feature exclusively for urban-scale ventilation simulation through the adaptive hybrid mesh strategy which has been introduced in Chapter 3. This feature would optimize the CFD simulation with less amount of mesh elements and less computational time.

5.4 Layout and function design

In this section, the layout and function design of the software would be described. The GUI layout is implemented by using the API of Qt. The Mainwindow Class inherited from QMainWindow Class is used as the application window, which provide the central window widget, menu bar widget etc and is the container of other user customized widgets. In the menu bar, six menus are created to support the functions of the software, namely File Menu, View Menu, Geometry Menu, Mesh Menu, Solver Menu and Post Menu respevtively. A user defined OccView Class which inherits the QGLWidget Class is set as the central window widget.

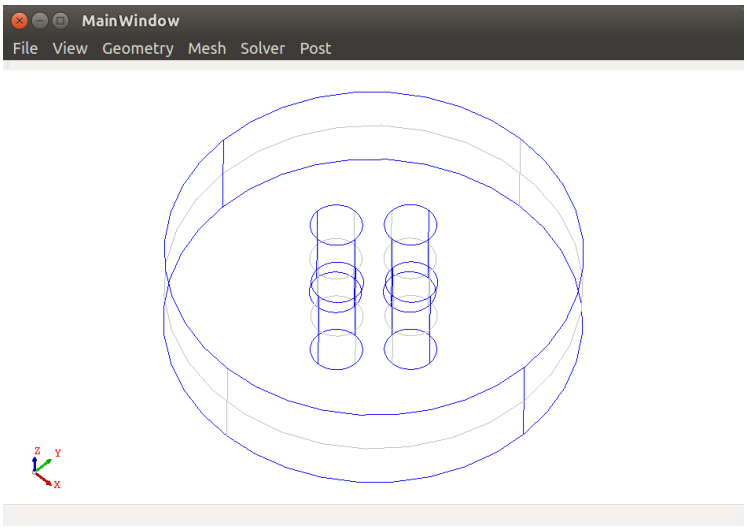
The functions of the software are listed below. A part of the functions have implemented while the majority are still under development.

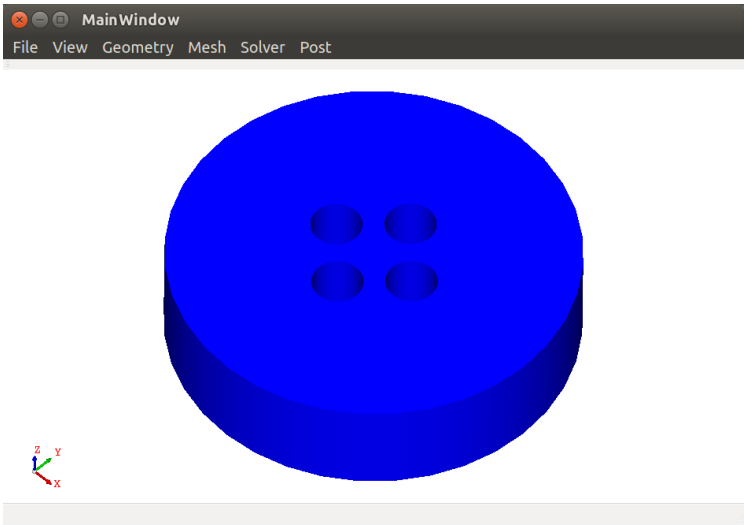
File
- Load geometry from file - Load mesh from file - Save geometry - Save mesh
View
- Pan - Zoom - Rotate - Fit to window - Display Mode - Clear
Geometry
- Draw - Select
Mesh
- Mesh1D - Mesh2D - Mesh3D - Boundary layer mesh
Solver
- Convert mesh format - Merge mesh - Set BC conditions - Set solver parameters - Run solver

Post
- Create Point - Create Line - Create Plane - Plot Contour - Plot Vector - Plot Streamline

Demonstration

As the software is still under development, parts of the functions are demonstrated to have an overview of the software GUI.





6 Future Study Plan

The problem stated in the chapter 3 would be continuing in order to finding a reliable method that connects the external model to the internal model successfully. Case study in chapter 4 has shown the feasibility of using CFD plus machine learning algorithm to make prediction on the indoor ventilation performance. There are several enhancements that could be further studied in the future. Firstly, to make the model more applicable to different design conditions, the input vector selection could be investigate further. Secondly, more complicated machine learning algorithms could be adopted, such as Neutral Network Model which can provide higher flexibility when fitting the unknown functional relationship between the input feature vector and the corresponding output vector, and hen the accuracy of these algorithms can be compared. The integrated software design and development would be continued to implement the functions described in chapter 5. Last but not least, wind tunnel test could be carried out to give an accuracy comparison of the result from CFD simulation, machine learning algorithm and wind tunnel experiment.

Reference

1. Lighting and Ventilation Requirements – Performance-based Approach. Buildings Department, Hong Kong Government.

2. Versteeg HK, Malalasekera W. An introduction to computational fluid dynamics: The finite volume method. Longman Scientific & Technical, New York. 1995.

3. T. Yang et al. A comparison of CFD and Full-scale Measurements for Analysis of Natural Ventilation. International Journal of Ventilation. 2006.

4. Huang S, Li QS and Xu S. Numerical evaluation of wind effects on a tall steel building by CFD. Journal of Constructional Steel Research, 63:612-627. 2007

5. Li QS, Fang JQ, Jeary AP, Paterson DA. Computation of wind loading on buildings by CFD. Hong Kong Institution of Engineers, Transactions, 2(4):253-66. 1998.

6. Launder BE, Kato M. Modeling flow-induced oscillations in turbulent flow around a square cylinder. ASME Fluid Eng. Conf. 1993.

7. Murakami S. Overview of turbulence models applied in CWE-1997. Journal of Wind Engineering and Industrial Aerodynamics, 74-76:1-24. 1998

8. Smagorinsky J. General circulation experiments with the primitive equations. I. the basic experiment. Monthly Weather Reviw, 91:99-164. 1963.

9. Germano M, Piomelli U, Moin P, Cabot WH. A dynamic subgrid scale eddy viscosity model. Physics of Fluids, A3(7): 1760-5. 1991

10. Lilly DK. A proposed modification of the Germano subgrid-scale closure model. Physics of Fluids, 83:159-69. 1992.

11. Nicoud F, Ducros F. Subgrid-scale stress modeling based on the square of e velocity gradient tensor. Flow, Turbulence and Combustion, 62(3):183-200. 1999.

12. Kim WW, Menon S. Application of the localized dynamic subgrid-scale model to turbulent wall-bounded flows. Technical report AIAA-97-0210. Reno (NV): American Institute of Aeronautics and Astronautics, 35th Aerospace Sciences Meeting. 1997.

13. Christopher Bishop. Pattern Recognition and Machine Learning. Springer-Verlag New York, 2006.

14. Ethem Alpaydin. Introduction to Machine Learning. The MIT Press, London. 2010

15. Brendan T. et al. A Machine Learning Strategy to Assist Turbulence Model Development. American Institute of Aeronautics and Astronautics. 2015

16. Rasmussen C.E. and Williams C.K.I.. Gaussian Processes for Machine Learning. MIT Press. 2006

17. Waston G.S.. Smooth Regression Analysis. The Indian Journal of Statistics. 1964.

18. Cleveland W.S. et al. Variance Reduction Methods I. Monte Carlo Simulation: IEOR E4703. 2004

19. Rosenblatt F. The perceptron, a perceiving and recognizing automation Project Para. Cornell Aeronautical Laboratory. 1957.

20. Franke J. et al. COST Action C14: Recommendations on the use of CFD in predicting pedestrian wind environment. Urban Civil Engineering. 2004.

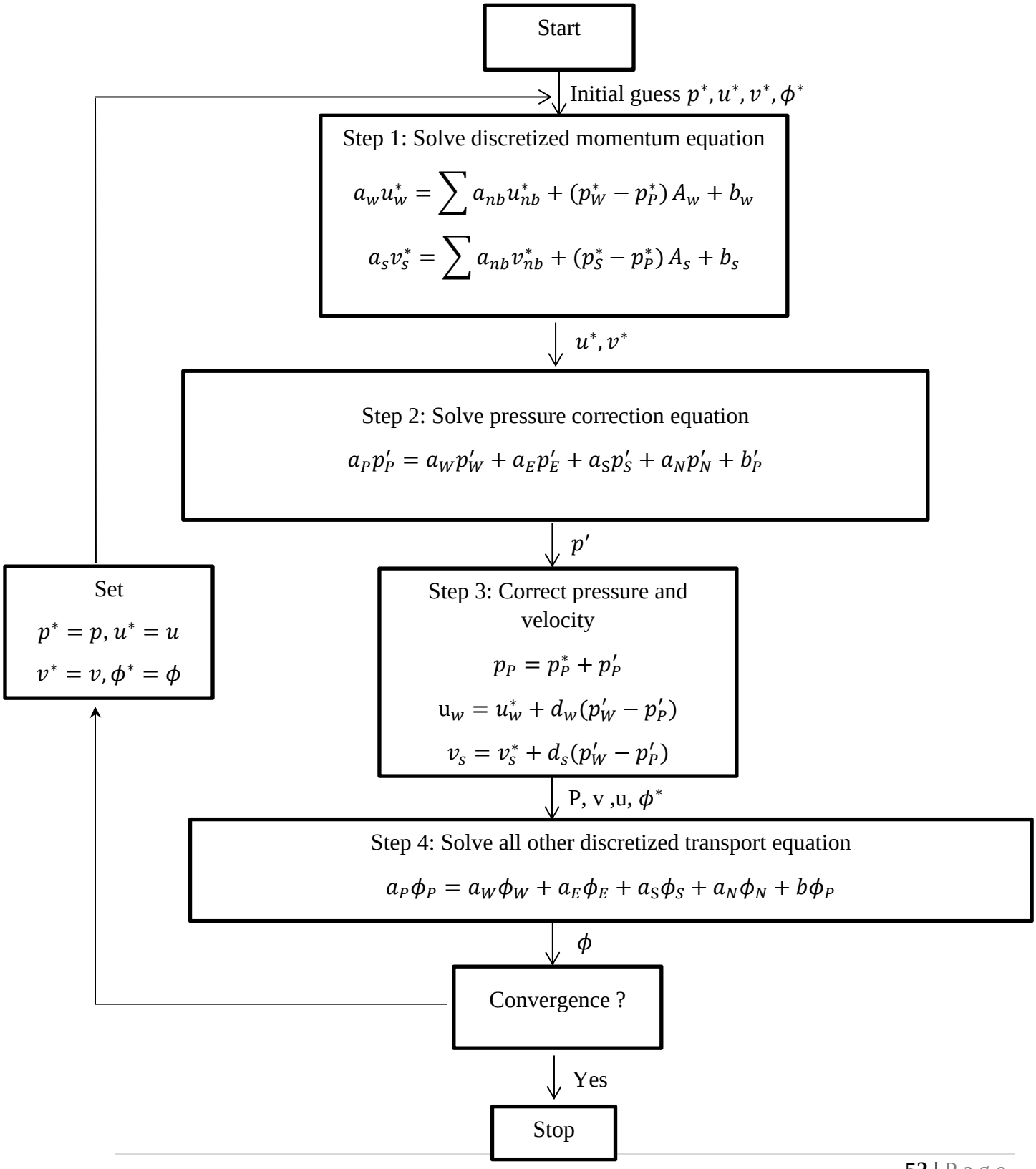
21. Tobin I. et al. Low-Cost Parallel Algorithms for 2:1 Octree Balance. 2012 IEEE 26th International Parallel and Distributed Processing Symposium. 2012.

22. OpenFOAM (2015) Version 2.3.x. http://www.openfoam.org.

23. F. Moukalled et al. The Finite Volume Method in Computational Fluid Dynamics An Advanced Introduction with OpenFOAM® and Matlab®. 2016.

Appendix A

Logical working procedure of SIMPLE algorithm



Appendix B

The Matlab code for the machine algorithms used in this study.

```
function y = phi(X,order)
X=X(:);
myLength=length(X);
i=2;
my_phi=[];
my_phi(1,:)=ones(1,myLength);
while(i <= order+1)
  my_phi(i,:)=X'.^(i-1);
  i=i+1;
end
y=my_phi;
end

function [my_est_theta] = est_theta_LS(x, y,
order)
y=y(:);
my_est_theta=pinv(phi(x,order)*phi(x,order
)')*phi(x,order)*y;
end

function [ my_est_theta ] =
est_theta_RLS(x, y, order,lamda)
y=y(:);
I=eye(size(phi(x,order)*phi(x,order)'));
my_est_theta=pinv(phi(x,order)*phi(x,order
)'+lamda*I)*phi(x,order)*y;
end

function [ my_est_theata ] =
est_theta_LASSO( x, y, order,lamda )
y=y(:);
PHI=phi(x,order);
H=[PHI* PHI', -PHI*PHI'; -PHI*PHI', PHI*
PHI'];
f= lamda - [PHI*y;-PHI*y];
A=-1*eye(size(H));
b=zeros(size(H,1),1);
X=quadprog(H,f,A,b);
theta_plus=X(1:length(X)/2);
theta_minus=X(length(X)/2+1:length(X));
my_est_theata = theta_plus-theta_minus;
end
function [ my_est_theata ] =
est_theta_RR( x, y, order )
y=y(:);
PHI=phi(x,order);
D=size(PHI,1);
n=length(x);
A=[-PHI', -eye(n); PHI', -eye(n)];
f=[zeros(D,1);ones(n,1)];
b=[-y;y];
X=linprog(f,A,b);
my_est_theata = X(1:D);
end

function hy = h(x,theta)
hy=(phi(x,length(theta)-1))'*theta;
end
```

Appendix C

Summary of the static pressure and velocity magnitude for Grid 30, Grid 40 and Grid 60 mesh configuration

	Grid 30		Grid 40		Grid 60	
Probe	static p	mag U	static p	mag U	static p	mag U
Line 0 -1						
0	4.50	0.09	4.46	0.06	4.39	0.08
1	4.44	0.29	4.40	0.30	4.33	0.31
2	4.17	0.80	4.16	0.74	4.10	0.78
3	4.27	0.98	4.22	0.93	4.24	0.92
4	1.86	1.74	1.67	1.73	1.59	1.70
5	3.60	1.27	3.76	1.40	3.67	1.29
6	4.13	1.18	4.16	1.32	4.13	1.20
7	4.35	1.30	4.49	1.10	4.48	1.24
8	4.66	1.12	4.72	1.23	4.77	1.18
9	4.99	1.19	4.87	1.20	4.98	1.13
10	5.21	1.13	5.20	1.11	5.31	1.05
11	5.43	1.07	5.54	1.01	5.53	0.99
12	5.66	0.99	5.70	0.96	5.86	0.88
13	6.10	0.83	6.03	0.84	6.08	0.81
14	6.32	0.74	6.36	0.72	6.41	0.68
15	6.54	0.65	6.52	0.65	6.62	0.59
16	6.74	0.55	6.84	0.50	6.83	0.49
17	7.13	0.33	7.13	0.33	7.13	0.33
18	7.29	0.23	7.27	0.24	7.32	0.21
19	7.43	0.20	7.50	0.15	7.58	0.11
20	7.53	0.31	7.66	0.36	7.71	0.24
21	7.44	0.81	7.67	0.54	7.79	0.58
22	7.00	1.24	7.35	1.06	7.65	0.91
23	5.77	1.86	5.37	2.07	6.33	1.79
Line 1-2						
24	-3.06	3.84	-3.63	3.94	-4.47	4.10
25	-8.16	1.10	-8.93	0.88	-9.43	0.63
26	-7.41	0.30	-8.08	0.53	-8.66	0.91
27	-7.22	0.38	-7.43	0.73	-7.72	0.83
28	-6.57	0.59	-6.99	0.71	-6.34	0.56
29	-6.14	0.57	-6.00	0.56	-5.60	0.38
30	-5.65	0.51	-5.12	0.35	-4.75	0.13
31	-5.16	0.41	-4.74	0.25	-4.31	0.06
32	-4.30	0.19	-4.12	0.08	-3.78	0.23
33	-3.94	0.10	-3.63	0.16	-3.50	0.34
34	-3.63	0.09	-3.43	0.24	-3.16	0.49
35	-3.36	0.18	-3.09	0.39	-2.98	0.58
36	-2.92	0.37	-2.95	0.46	-2.74	0.70

37	-2.75	0.46	-2.71	0.59	-2.61	0.77
38	-2.60	0.55	-2.51	0.71	-2.49	0.84
39	-2.47	0.63	-2.43	0.76	-2.34	0.94
40	-2.36	0.70	-2.28	0.86	-2.26	0.99
41	-2.18	0.84	-2.16	0.95	-2.14	1.08
42	-2.10	0.90	-2.11	0.99	-2.08	1.13
43	-2.03	0.96	-2.02	1.08	-1.99	1.20
44	-1.98	1.02	-1.94	1.15	-1.94	1.25
45	-1.88	1.12	-1.90	1.19	-1.88	1.31
46	-1.83	1.18	-1.85	1.26	-1.84	1.35
47	-1.80	1.22	-1.81	1.34	-1.80	1.42
Line 2-3						
48	-1.76	1.38	-1.78	1.49	-1.78	1.56
49	-1.92	0.09	-1.91	0.25	-1.93	0.12
50	-1.98	0.13	-1.96	0.10	-1.98	0.06
51	-2.16	0.28	-2.09	0.21	-2.06	0.17
52	-1.01	1.26	-0.90	1.30	-0.86	1.29
53	-1.84	0.35	-1.89	0.41	-1.88	0.39
54	-1.94	0.57	-1.99	0.66	-2.00	0.63
55	-2.01	0.77	-1.99	0.83	-1.99	0.81
56	-1.96	0.73	-1.98	0.86	-1.97	0.84
57	-1.92	0.88	-1.92	0.87	-1.93	0.84
58	-1.88	0.88	-1.89	0.86	-1.89	0.83
59	-1.83	0.87	-1.81	0.84	-1.81	0.82
60	-1.71	0.83	-1.72	0.82	-1.76	0.80
61	-1.65	0.81	-1.67	0.80	-1.67	0.76
62	-1.58	0.78	-1.58	0.75	-1.58	0.72
63	-1.52	0.74	-1.53	0.73	-1.51	0.69
64	-1.38	0.65	-1.43	0.67	-1.42	0.63
65	-1.32	0.60	-1.34	0.60	-1.36	0.58
66	-1.26	0.54	-1.25	0.51	-1.28	0.50
67	-1.20	0.47	-1.21	0.46	-1.23	0.43
68	-1.16	0.39	-1.15	0.33	-1.19	0.35
69	-1.14	0.23	-1.14	0.20	-1.16	0.21
70	-1.19	0.21	-1.17	0.17	-1.20	0.11
71	-1.26	0.21	-1.27	0.20	-1.27	0.17
72	-1.29	0.19	-1.29	0.17	-1.29	0.13