

W. Tian, T. A. Sevilla, D. Li, W. Zuo, M. Wetter 2018. "Fast and Self-Learning Indoor Airflow Simulation Based on In Situ Adaptive Tabulation." *Journal of Building Performance Simulation*, 11(1), pp. 99-112.

## **Fast and Self-Learning Indoor Airflow Simulation Based on In Situ Adaptive Tabulation**

Wei Tian<sup>1</sup>, Thomas Alonso Sevilla<sup>1</sup>, Dan Li<sup>1</sup>, Wangda Zuo<sup>1\*</sup>, Michael Wetter<sup>2</sup>

<sup>1</sup>*University of Miami, Coral Gables, FL*

<sup>2</sup>*Lawrence Berkeley National Laboratory, Berkeley, CA*

\**Corresponding Author: w.zuo@miami.edu*

# **Fast and Self-Learning Indoor Airflow Simulation Based on In Situ Adaptive Tabulation**

## **ABSTRACT**

Fast simulation for stratified indoor airflow distributions is desired for various applications, such as design of advanced indoor environments, emergency management, and coupled annual energy simulation for buildings with stratified air distributions. Reduced order models trained by pre-computed computational fluid dynamics results are fast, but their prediction may be inaccurate when applied for conditions outside the training domain. To overcome this limitation, we propose a fast and self-learning model based on an in situ adaptive tabulation (ISAT) algorithm, which is trained by a fast fluid dynamics (FFD) model as an example. The idea is that the ISAT will retrieve the solutions from an existing data set if the estimated prediction error is within a pre-defined tolerance. Otherwise, the ISAT will execute the FFD simulation, which is accelerated by running in parallel on a graphics processing unit, for a full scale simulation. This paper systematically investigates the feasibility of the ISAT for indoor airflow simulations by presenting the ISAT-FFD implementation alongside results related to its overall performance. Using a stratified indoor airflow as an example, we evaluated how the training time of ISAT was impacted by four factors (training methods, error tolerances, number of inputs, and number of outputs). Then we demonstrated that a trained ISAT model can predict the key information for inputs both inside and outside the training domain. The ISAT was able to answer query points both inside and close to training domain using retrieve actions within a time less than 0.001s for each query. Finally, we provided suggestions for using the ISAT for building applications.

Key words: In Situ Adaptive Tabulation, Fast Fluid Dynamics, Reduced Order Model, Self-Learning, Indoor Airflow Simulation

## **Nomenclature:**

***CFD***: Computational Fluid Dynamics

***FFD***: Fast Fluid Dynamics

***OpenCL***: Open Computing Language

***GPU***: Graphics Processing Unit

***ISAT***: In Situ Adaptive Tabulation

***ISAT-FFD***: ISAT algorithm coupled with the FFD

***ROM***: Reduce Order Model

***EOA***: Ellipsoid of Accuracy

**$\mathbf{x}^{(q)}$** : Query point in m dimension space

**$\mathbf{x}^{(0)}$** : Tabulation point in m dimension space in ISAT table

**$T_{wall}$** : Temperature on west, east, south, and north walls

**$T_{floor}$** : Temperature on the floor

**$T_{inlet}$** : Temperature at inlet

**$|V|_{inlet}$** : Velocity magnitude at inlet

**$T_{occ}$** : Temperature on average of the occupant zone, where  $Z < 1.22$  m

**$|V|_{occ}$** : Velocity magnitude on average of the occupant zone, where  $Z < 1.22$  m

**$T_{sensor}$** : Temperature at the sensor location (1.22m, 1.22m, 1.24m)

**$|V|_{sensor}$** : Velocity magnitude at the sensor location (1.22m, 1.22m, 1.24m)

## 1 Introduction

Fast simulation of stratified indoor airflow distributions is desired for building applications, such as design of stratified indoor environments (Wang et al. 2010), emergency management (Zhai et al. 2003), and coupled annual energy simulation for buildings with stratified air distributions (Zhai et al. 2002). Although Computational Fluid Dynamics (CFD) models are used in the design and evaluation of ventilation performance with stratified air distribution, they are usually too slow to meet the requirement of a fast computing speed (Zhai et al. 2002).

To reduce the computing time, researchers proposed a Fast Fluid Dynamics (FFD) model as an intermediate method between the CFD and multi-zone airflow network models (Zuo and Chen 2009; Zuo 2010; Zuo et al. 2010). FFD solves the same Navier-Stokes and balance equations for energy and species that CFD does. By using different mathematical algorithms, FFD can increase the computing speed by 30 times when compared to CFD (Zuo and Chen 2009; Zuo 2010; Zuo et al. 2010). In addition, one can further accelerate the FFD simulation by running it in parallel on graphics processing units (GPUs) (Zuo and Chen 2010a). Due to its high speed, FFD has been used for simulating various airflows inside and around buildings (Zuo and Chen 2009; Zuo 2010; Zuo et al. 2010; Zuo and Chen 2010c, 2010b, 2010a; Jin et al. 2012; Jin et al. 2013). Furthermore, the FFD was coupled with the Modelica Buildings library (Wetter et al. 2014) for integrated simulations of indoor environment and building HVAC systems (Zuo et al. 2016). Although significantly faster than CFD, FFD is still not fast enough for the aforementioned applications.

In order to perform fast simulation of stratified indoor airflows, researchers proposed to use reduced order models (ROMs) (Kolokotsa et al. 2009; Hazyuk et al. 2012; Desta et al. 2004; Hiyama et al. 2010; Kim et al. 2015; Ahuja et al. 2011). A common

approach is to use a regression model with a limited number of inputs in order to construct the data-driven ROMs based on pre-calculated CFD results (Chen and Kooi 1988).

However, they can rarely reflect the dynamics of a full order CFD model. On the other hand, ROMs can be built by using the Principal Orthogonal Decomposition method to extract important features (snapshots) of the flow and then project them to a Linear Time Invariant system (Li et al. 2013). Such ROMs can partially maintain the dynamics of the full order CFD model. Although it can be time consuming to run various CFD simulations to generate training data, the trained ROMs can compute the solution almost instantaneously by either interpolating or extrapolating using an existing data set. However, conventional ROMs can only perform well when the inputs are within or near the training domain. Consequently, if the inputs are too far outside the training domain, the ROMs may resolve them without any guaranteed accuracy (Stockwell and Peterson 2002).

Obviously, it is too expensive to train a ROM for a domain which includes all the possible inputs of the application. Therefore, to overcome this drawback of conventional ROMs, we propose to a fast and self-learning indoor airflow simulation method. The idea is that we will train the ROM within a domain in which the system is most likely to operate. If the trained ROM cannot project the solutions accurately, a full scale CFD simulation will be executed. The newly generated data from the CFD simulation will then be used to enlarge the training domain for the ROM.

To realize the proposed fast and self-learning airflow simulation method, we selected an in situ adaptive tabulation (ISAT) algorithm. ISAT is a general function approximation method. ISAT was originally proposed to speed up combustion simulations (Pope 1997). It stores key simulation data in a data table and linearly interpolates the

solutions from the table if the inputs are within the region where the interpolation accuracy is guaranteed. Otherwise, it executes a full scale simulation to obtain the solution.

Given that FFD is a full scale airflow simulation model of high speed, the ISAT, a fast and self-learning approach, can be evaluated on accelerating indoor airflow simulation using FFD. Although ISAT has accelerated multi-species combustion simulations (Singer and Pope 2004; Singer et al. 2006), it is not clear that ISAT will be suitable for indoor airflow simulations. This study implemented and then evaluated the performance of the proposed ISAT-FFD model. We first introduce mathematical descriptions of FFD and ISAT. We then illustrate the implementation of the ISAT-FFD model on a hybrid computing platform consisting of a central processing unit (CPU) and a GPU. Afterwards, using a stratified indoor airflow, we assess the performance of ISAT at the training stage as well as the evaluation stage. Finally, we present the conclusion and potential applications of this research.

## 2 Fast Fluid Dynamics

FFD solves the Navier-Stokes equations:

$$\frac{\partial \mathbf{U}_i}{\partial t} = -\mathbf{U}_j \frac{\partial \mathbf{U}_i}{\partial x_j} + \nu \frac{\partial^2 \mathbf{U}_i}{\partial x_j^2} - \frac{1}{\rho} \frac{\partial P}{\partial x_i} + \mathbf{F}_i \quad (1)$$

where  $\mathbf{U}_i$  and  $\mathbf{U}_j$  are the velocity component in  $x_i$  and  $x_j$  directions, respectively,  $\nu$  is the kinematic viscosity,  $\rho$  is the fluid density,  $P$  is the pressure,  $t$  is the time, and  $\mathbf{F}_i$  is the source term, such as the buoyancy force. FFD splits the Navier-Stokes equation into the following three equations:

$$\frac{\partial \mathbf{U}_i}{\partial t} = -\mathbf{U}_i \frac{\partial \mathbf{U}_i}{\partial x_j} \quad (2)$$

$$\frac{\partial \mathbf{U}_i}{\partial t} = \nu \frac{\partial^2 \mathbf{U}_i}{\partial x_j^2} + \mathbf{F}_i \quad (3)$$

$$\frac{\partial \mathbf{U}_i}{\partial t} = -\frac{1}{\rho} \frac{\partial P}{\partial x_i} \quad (4)$$

FFD first solves the advection equation (2) using a semi-Lagrangian method (Courant et al. 1952). It then solves the diffusion equation (3) with an implicit scheme. Finally, it solves the pressure equation (4) together with the continuity equation

$$\frac{\partial \mathbf{U}_i}{\partial x_i} = 0 \quad (5)$$

using a projection-correction method (Chorin 1967). FFD also applies a similar algorithm to solve the conservation equations of energy and species. For more details of the FFD model, one may refer to the literature (Zuo and Chen 2009; Zuo 2010; Zuo et al. 2010; Jin et al. 2012; Jin et al. 2013; Zuo et al. 2012; Yang 2013).

### 3 In Situ Adaptive Tabulation

#### 3.1 Mathematical Description of ISAT

For a nonlinear model such as CFD or FFD, we describe the outputs  $\mathbf{y}$  as a function of the inputs  $\mathbf{x}$ :

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \quad (6)$$

$$\mathbf{x} = \{x_1, x_2, \dots, x_m\} \quad (7)$$

$$\mathbf{y} = \{y_1, y_2, \dots, y_n\} \quad (8)$$

where  $\mathbf{x}$  is a set of independent scalar variable  $x_i$ ; and  $\mathbf{y}$  is a set of dependent scalar variable  $y_i$ . As an example, if a *query point*  $\mathbf{x}^{(q)}$  is close to a *tabulation point*  $\mathbf{x}^{(0)}$ , ISAT can estimate  $\mathbf{y}(\mathbf{x}^{(q)})$  using a linear interpolation:

$$\mathbf{y}(\mathbf{x}^{(q)}) \approx \mathbf{y}^l(\mathbf{x}^{(q)}) = \mathbf{y}(\mathbf{x}^{(0)}) + \mathbf{A}(\mathbf{x}^{(0)})\delta\mathbf{x} \quad (9)$$

$$\mathbf{A}_{ij}(\mathbf{x}^{(0)}) = \frac{\partial f_i(\mathbf{x}^{(0)})}{\partial x_j}, \delta \mathbf{x} = (\mathbf{x}^{(q)} - \mathbf{x}^{(0)}) \quad (10)$$

where  $\mathbf{A}(\mathbf{x}^{(0)})$  is called the mapping gradient matrix ( $n \times m$ ) at  $\mathbf{x}^{(0)}$  and it is stored together with  $\mathbf{x}^{(0)}$  in the record (Pope 1997); here,  $i$  and  $j$  are index. Note that the superscript which is located inside the parenthesis, for example, in  $\mathbf{x}^{(q)}$ ,  $q$  denotes the  $q_{th}$  query point. The parenthesis is used to avoid confusion of interpreting  $\mathbf{x}^{(q)}$  as  $\mathbf{x}$  to the power of  $q$ . To calculate  $\mathbf{A}(\mathbf{x}^{(0)})$ , we can use user defined function or the ISAT algorithm using two neighbor points in the table. For instance, the ISAT algorithm finds neighbour record to  $\mathbf{x}^{(0)}$ , saying  $\mathbf{x}^{(1)}$ . Then the mapping gradient matrix can be calculated as below.

$$\mathbf{A}_{ij}(\mathbf{x}^{(0)}) = \frac{f_i(\mathbf{x}^{(0)}) - f_i(\mathbf{x}^{(1)})}{x^{(0)}_j - x^{(1)}_j}. \quad (11)$$

ISAT can automatically detect if a linear interpolation can be performed for  $\mathbf{x}^{(q)}$  based on its relationship to the region of accuracy of  $\mathbf{x}^{(0)}$  within which the interpolation error is not larger than error tolerance. Finally, the interpolation error  $\varepsilon$  is defined as:

$$\varepsilon = \left\| \mathbf{B} \left( \mathbf{y}(\mathbf{x}^{(q)}) - \mathbf{y}^l(\mathbf{x}^{(q)}) \right) \right\|_2 \leq \varepsilon_{total} \quad (12)$$

where  $\varepsilon_{total}$  is the total error tolerance for all outputs, which is a scalar variable set by the user.  $\mathbf{B}$  is an  $n \times n$  scaling diagonal matrix which is predefined by the users before executing ISAT (Pope 1997). Its primary function is to make the interpolation error of each output comparable given that number of outputs is larger than one. For example, suppose an output consisting of velocity magnitude and temperature, and velocity magnitude and temperature varying at 0.01 m/s and 0.1 °C, it is important to multiply the interpolation error of velocity magnitude by a factor of 10 to ensure that the accuracy of velocity magnitude prediction is not neglected.



Now, by assuming a constant approximation of  $\mathbf{y}(\mathbf{x}^{(q)}) = \mathbf{y}(\mathbf{x}^{(0)})$ , and substituting Equation (9) in Equation (12), one obtains:

$$\varepsilon = \|\mathbf{BA}\delta\mathbf{x}\|_2 \leq \varepsilon_{total} \quad (13)$$

$$\frac{\delta\mathbf{x}^T \mathbf{A}^T \mathbf{B}^T \mathbf{BA} \delta\mathbf{x}}{\varepsilon_{total}^2} \leq 1 \quad (14)$$

Equation (14) is the criterion used to determine if the linear interpolation Equation (9) is valid for point  $\mathbf{x}^{(q)}$ . By subsequently applying Cholesky decomposition (Tuma 2002) to the semi-definite symmetrical matrix  $\frac{\mathbf{A}^T \mathbf{B}^T \mathbf{BA}}{\varepsilon_{total}^2}$ , one can obtain the equation below,

$$\delta\mathbf{x}^T \mathbf{Q}^T \mathbf{\Lambda} \mathbf{Q} \delta\mathbf{x} \leq 1 \quad (15)$$

where  $\mathbf{Q}$  is an  $m \times m$  unitary matrix and  $\mathbf{\Lambda}$  is an  $m \times m$  diagonal matrix. Equation (15) defines a hyper-ellipsoid area, which is called the *Ellipsoid of Accuracy* (EOA) in ISAT. The radius in the  $i$  direction,  $l_i$ , is defined as:

$$l_i = \frac{\varepsilon_{total}}{\sigma_i} \quad (16)$$

The  $\sigma_i$  is the  $i_{th}$  diagonal entry of  $n \times m$  diagonal matrix  $\mathbf{\Sigma}$ , which is derived from Singular Value Decomposition of  $\mathbf{BA}$ . The sketch of the EOA in different dimensions is shown in Figure 1. It is worthy to note that due to the incorporation of the constant assumptions and essence of linear approximation of nonlinear function, it is not guaranteed that the interpolation error  $\varepsilon$  is less than the error tolerance  $\varepsilon_{total}$  for all the points in the EOA (Pope 1997).

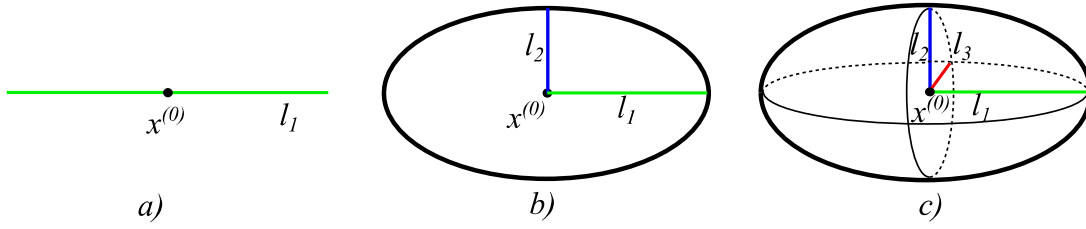


Figure 1 Sketch of *EOA* in different dimensions (a: 1D; b: 2D; c: 3D)

### 3.2 ISAT Workflow

Figure 2 shows the workflow of ISAT. Given a query point  $\mathbf{x}^{(q)}$ , ISAT will perform one of the following three actions: *retrieve*, *grow* or *add*. As the first step (step 1.1), ISAT will look up in the data table and find the nearest data point to  $\mathbf{x}^{(q)}$ , which is assumed hereby to be  $\mathbf{x}^{(0)}$ . Then step 1.2 is used to check if the query point  $\mathbf{x}^{(q)}$  is within the EOA of  $\mathbf{x}^{(0)}$ . If so, the *retrieve* action will be executed using Equation (9) in step 2.1, and its linear interpolation  $\mathbf{y}^l(\mathbf{x}^{(q)})$  will be returned in step 2.2. If not, an evaluation of the nonlinear Equation (6) in step 3.1 will be performed. Furthermore, the inequality (12) will be used to determine if the difference between the solution of Equations (6) and (9) is less than the error tolerance  $\varepsilon_{total}$  in step 3.2. If the inequality (12) is met, the *grow* action will be performed to enlarge the EOA with minimal volume increase to contain the query point  $\mathbf{x}^{(q)}$  by updating the matrix  $\mathbf{Q}$  and  $\mathbf{A}$  in Equation (15) in step 4 using Householder matrix algorithm and rank-one modification algorithm (Pope 2008). The query point  $\mathbf{x}^{(q)}$  is going to be abandoned. Otherwise, the *add* action will be performed to add the query point  $\mathbf{x}^{(q)}$  as one additional record in the table of step 5.

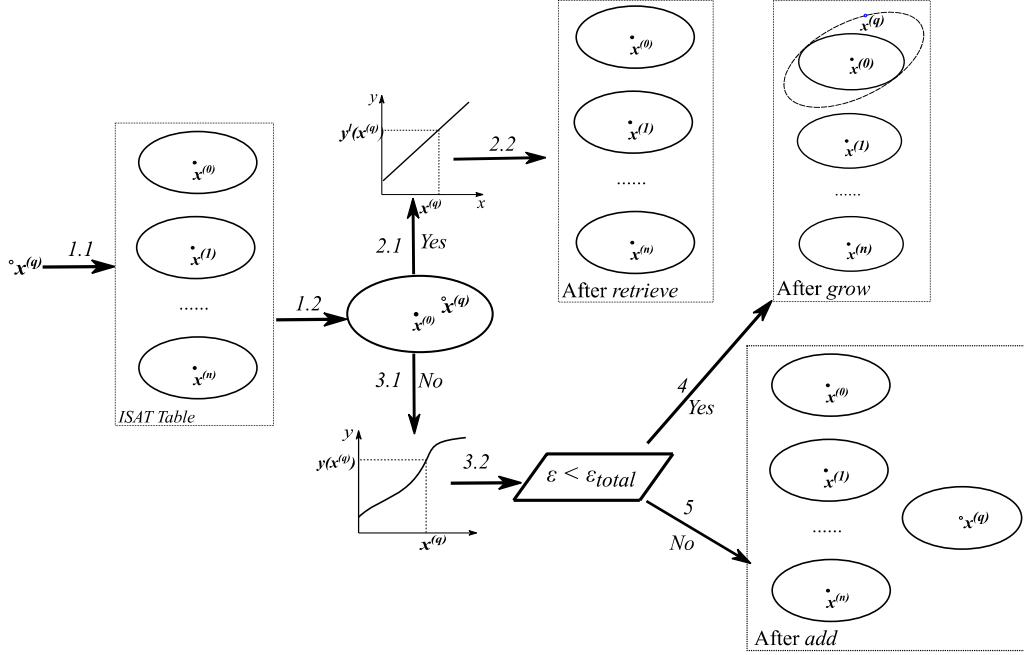


Figure 2 Workflow of ISAT

### 3.3 Training Method for ISAT

As a self-learning method, the ISAT's learning time depends on the training method. Besides a conventional *constant interval* method as sketched in Figure 3(a), we propose an *automatic interval refinement* method to generate the training data as shown in Figure 3(b). The *constant interval* method requires manually defining the interval in each dimension of  $\mathbf{x}$ . After all the query points are evaluated, the table is considered fully trained. The *automatic interval refinement* method dynamically refines the interval by halving the last one if the training is not complete. The completeness of training can be measured by:

$$N_{add+grow} \leq \eta N_{queries} \quad (17)$$

where  $N_{add+grow}$  is the total number of *add* and *grow* under the current interval,  $N_{queries}$  is the number of queries, and  $\eta$  is a constant coefficient. By default,  $\eta = 0$  and  $N_{add+grow} =$

0 indicate that the training is complete if under the current interval there are no *add* and *grow* actions generated.

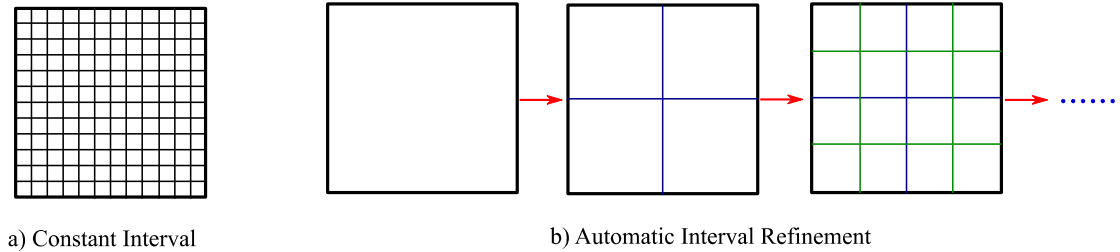


Figure 3 *Constant interval* method and *automatic interval refinement* method to train the ISAT table.

#### 4 ISAT-FFD Integration

The ISAT algorithm and FFD models have been implemented in previous studies (Pope 1997; Zuo and Chen 2009). We will now describe the implementation of the ISAT and FFD coupling, which enables setting up the ISAT algorithm, launching the FFD simulation, and extracting the FFD results for the ISAT algorithm. As shown in

Figure 4, the first stage in the ISAT-FFD scheme is initialization. This includes setting up ISAT parameters, e.g. error tolerance and dimensions of inputs and outputs, and defining the training range for the studied problem. In the training stage, query points within the training domain are generated to populate the ISAT data table. If using the *constant interval* method, the ISAT table is completely trained once all the generated points are evaluated. If using *automatic interval refinement* method, the ISAT table training is completed once inequality (17) in section 3.3 is met. After the ISAT table is trained, the program moves to the evaluation stage, where query points within the evaluation domain are generated as inputs. During the evaluation stage, most queries will be handled by *retrieve* and the remaining few queries will be answered by calling FFD.

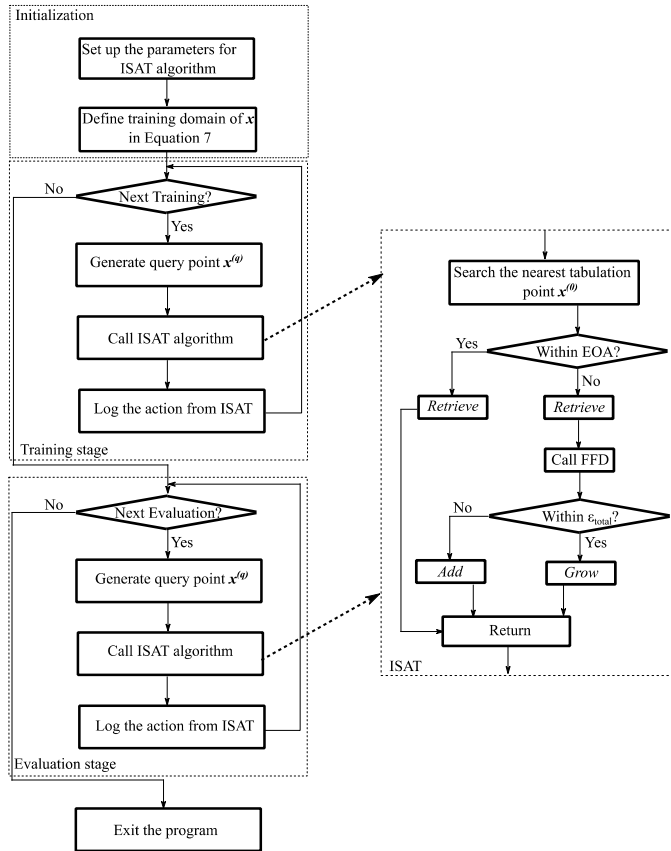


Figure 4 Framework of ISAT-FFD approach

The ISAT-FFD framework was implemented using C code. To speed up the FFD simulation, a FFD program running in parallel on a graphic processing unit (GPU) was developed (Tian et al. 2016). The parallelization was realized using a hybrid code of C and OpenCL language (Khronos 2012). Cornell University provided the original ISAT source codes written in FORTRAN. The source codes are available at:

<https://tcg.mae.cornell.edu/isat.html>. The simulation was performed using a DELL workstation with a Xeon E5-1603 CPU and AMD FirePro W8100 GPU.

## 5 Numerical Experiments

To evaluate the feasibility and performance of the proposed ISAT-FFD model for indoor airflow simulations, we evaluated it using a mixed convection flow in a room with a

box at the centre. Since the ISAT algorithm only allows one error tolerance for all outputs, we used the scaling matrix  $\mathbf{B}$  defined in Equation (12) to convert the errors from different outputs into a single aggregated error. However, users will have to construct the scaling matrix  $\mathbf{B}$  on a case-by-case basis. The following two sections will discuss the case description and the construction of the scaling matrix  $\mathbf{B}$  for this case study.

### ***5.1 Case Description***

As sketched in Figure 5, this case involves a stratified airflow with strong buoyance in a closed space, by adding a heat source to the obstacle and controlling the temperature of the walls (Wang and Chen 2009). The box is located at the center of the room. We chose this case other than a real room because the experimental data obtained from a well-controlled environmental chamber had high quality which was often used for model validation (Chen and Srebric 2002). The inlet velocity magnitude and temperature are 1.36  $m/s$  and 22.2  $^{\circ}C$ , respectively. Note that the inlet velocity is normal to the inlet surface area. The temperatures of the box surface, top, floor, and other walls are 36.7  $^{\circ}C$ , 25.8  $^{\circ}C$ , 26.9  $^{\circ}C$ , and 27.4  $^{\circ}C$ , respectively. Note that the velocity at and inside obstacle is 0. The grid size is  $40 \times 40 \times 40$  and the simulation time is 300s with a time step size of 0.05s. The FFD model and numerical settings have been validated in a previous study (Tian et al. 2016).

After validating the FFD model, we presented the velocity magnitude and temperature contours at the plane sliced at  $Y=1.22$  m, as shown in the Figure 6. Both contours show that the strong buoyance airflow has a stratified distribution in velocity and temperature. In terms of velocity, high jet is formed near the ceiling and circulation was shown between the box and room. Regarding the temperature, the plume clearly occurred above the heated box.

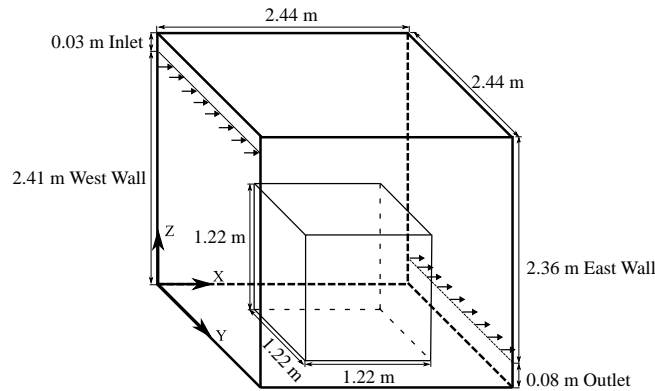


Figure 5 Schematic of the buoyance airflow in an empty room with a box.

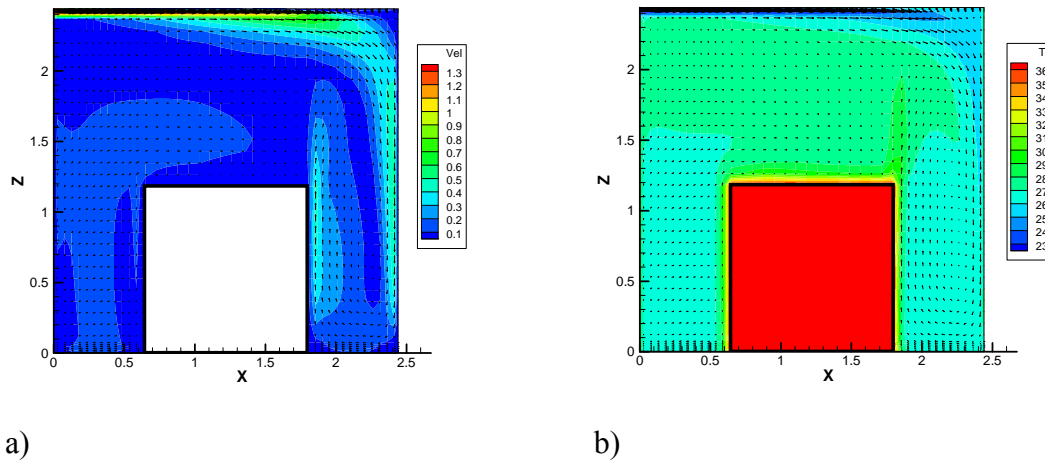
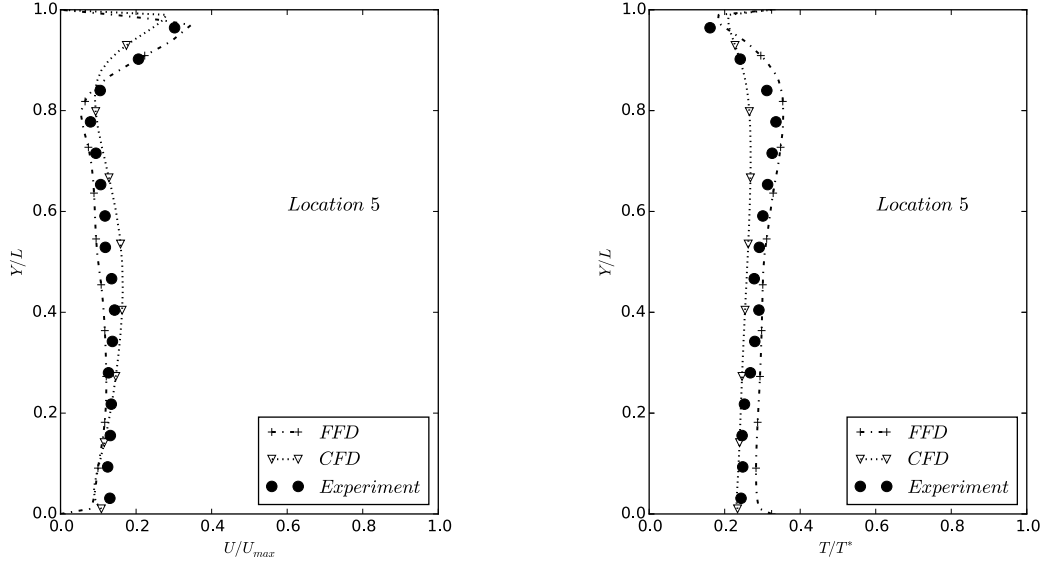


Figure 6 Velocity contour (a) and temperature contour (b) at plane at  $Y=1.22$  m

To further show FFD's capability to simulate such stratified non-isothermal flow, Figure 7 compares the velocity magnitude and temperature profiles at one point from FFD, CFD, and experimental data. Profiles predicted by FFD is in reasonable accordance to the experiment at comparable level as CFD does. A Previous study (Tian et al. 2016) has systematically studied the accuracy of parallel FFD model on various flows, ranging from lid driven cavity flow, to isothermal flow in an empty box, to stratified non-isothermal flow

in a room with a box located at center. It is found that for the above case studies, FFD has comparable simulation accuracy as CFD does.



a)

b)

Figure 7 Comparison of velocity magnitude profile (a) and temperature profile (b) at point (2.0574 m, 1.22 m) at XY plane

## 5.2 Construction of the Scaling Matrix $B$

For the mixed convection flow defined above, we defined two sets of inputs: temperature (20 - 30 °C) and inlet velocity magnitude (1 - 2 m/s) as shown in Table 1. The corresponding outputs can be normalized as:

$$V_{norm} = \frac{|V|}{|V|_{inlet}} \quad (18)$$

$$T_{norm} = \frac{T - T_{min}}{T_{max} - T_{min}} \quad (19)$$



where  $|V|_{inlet}$  is the velocity magnitude at the inlet;  $T_{min}$  and  $T_{max}$  are the lowest temperature and highest temperature in the inputs, respectively. To get the highest ratio, we divided the highest  $T_{norm}$  by the lowest  $V_{norm}$ . The results were summarized in Table 1. As the ratios are in order of 10 for all the tests, we set the diagonal entries of  $\mathbf{B}$  corresponding to velocity magnitude outputs as 10.

Table 1 Inputs and normalized outputs of the sensitivity study

Scenario #	Input Variables				Normalized Output Variables				Highest Ratio
	$T_{floor}$ (°C)	$T_{wall}$ (°C)	$T_{inlet}$ (°C)	$ V _{inlet}$ (m/s)	$T_{occ}$	$ V _{occ}$	$T_{sensor}$	$ V _{sensor}$	
1	25	25	20	1	0.984	0.1034	1.184	0.0799	14.8
2	30	25	20	1	1.136	0.1285	1.308	0.0634	20.6
3	25	30	20	1	0.756	0.1293	0.875	0.0645	13.6
4	25	25	20	2	0.946	0.1971	1.372	0.0174	78.9
5	30	30	25	2	0.796	0.1993	1.048	0.0159	65.9

## 6 Simulation Results

To gain comprehensive understanding of ISAT-FFD for indoor airflow simulation, we studied its performance in both the training and evaluation stages. The results and findings are presented in the following two sections.

### 6.1 Performance of ISAT-FFD in Training Stage

This section accounts the performance of the ISAT-FFD model at the training stage. It focuses on the cost of training (measured by training time) associated with four key factors: training methods, error tolerances, numbers of inputs, and numbers of outputs.

#### 6.1.1 Training Methods

This test compares the training time by using the two training methods proposed in Section 3.3. We used two inputs consisting of  $T_{wall}$  and  $T_{floor}$  and one output of  $T_{occ}$ . In the training domain both temperatures ranged from 25 to 30 °C. The error tolerance for the

output was set to 0.4. When using the *constant interval* method, the increment in each dimension was defined to be 0.1  $K$ .

By using the *constant interval* method, the ISAT-FFD evaluated 2,601 queries, which led to 1,424 *retrieve* actions, 1,130 *grow* actions, and 47 *add* actions, respectively. Even with a powerful GPU, the ISAT-FFD using the *constant interval* method took 24.8 hours to complete the training. On the contrary, the ISAT-FFD using the *automatic range refinement* method assessed 66,049 queries, which is approximately 25 times more than the ISAT-FFD with *constant interval* method did. However, about 99.9 % of the total queries were resolved by *retrieve* actions and only 0.01% of the queries resulted in *add* (13) and *grow* (59) actions that required calling the FFD simulation. It is worth mentioning that the ISAT will perform accuracy tests randomly based on the frequency of *grow* actions by calling the FFD. The accuracy tests may shrink the EOA to ensure the accuracy of the *retrieve* action. Simulations using the *constant interval* method triggered 103 accuracy tests and the number reduced to 31 when using the *automatic range refinement* method. As a result, the ISAT-FFD trained with the *automatic range refinement* method only took roughly 8% of the time when compared to the ISAT-FFD with the *constant interval* method.

To explain the training time difference between the two training methods, we plotted the *add* actions performed by both approaches (Figure 8a). As expected, the *automatic range refinement* method led to a more scattered distribution of *add* actions in the training domain than the *constant interval* method. Because each *add* action resulted in a new record in the ISAT table, fewer *add* actions mean that the ISAT needed fewer records to cover the training domain. Figure 8(b) compares the trajectories of training time

used by the two methods. Since the time required by *retrieve* actions are negligible compared to that for *add* and *grow* actions, the time increase pattern actually reflects the distribution of *add* and *grow* actions during the training. The increasing rate of the trajectory in the simulation using the *constant interval* method was approximately constant, which suggests the uniform distribution of *add* and *grow* actions over the whole evaluation. The training time by the *automatic range refinement* method increased fast for the first 100 queries, and then the rate of increase flattened afterwards. This indicates that the *add* and *grow* actions mainly happened at the beginning of the training, and the ISAT-FFD model then can answer the remaining queries mainly using the *retrieve* actions.

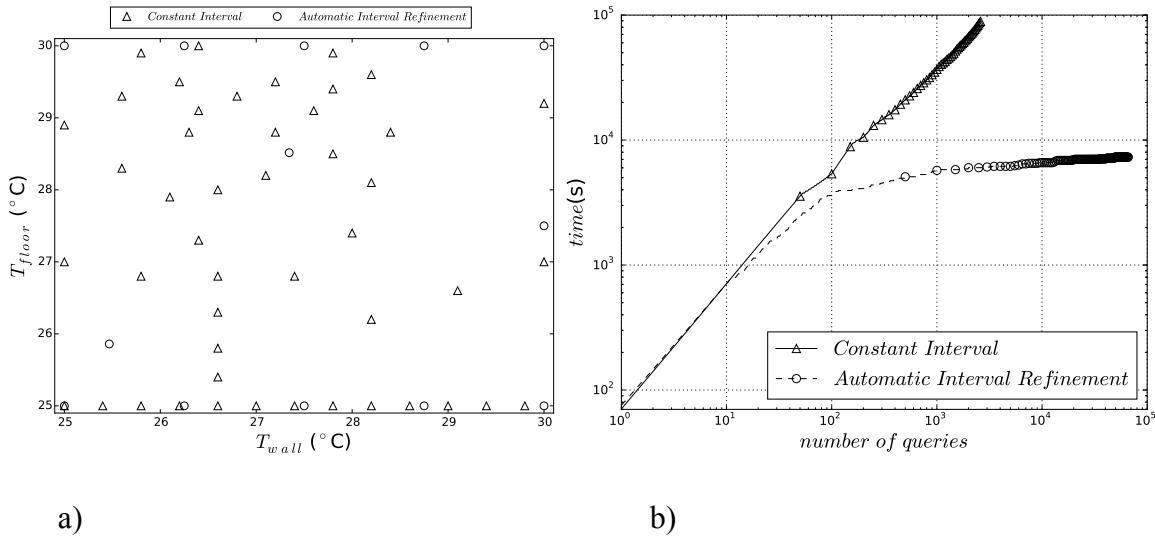


Figure 8 Comparison of the *add* actions (a) and training time (b) of the ISAT-FFD using different training methods

6.1.2 Error Tolerances

Equation (16) explains the impact of the error tolerance on the shape of the EOA. Namely, the larger the error tolerance, the larger the size of the EOA and the less time the

training process takes. To set up a numerical test, two inputs consisting of  $T_{wall}$  and  $T_{floor}$  and one output of  $T_{occ}$  were used. In the training domain both temperatures ranged from 25 to 30 °C. The *automatic range refinement* method was used and the error tolerance varied from 0.2 to 1.0 with increments of 0.2.

The simulation results (Figure 9) show that, with the lowest error tolerance (0.2) the total number of queries was 263,169, out of which there were 262,813 (99.9%) *retrieve*, 304 *grow*, and 52 *add* actions. The total training time was approximately 8.2 hours. By comparison, using a larger error tolerance (1.0) the number of queries was only 4,225, of which 99.3% are *retrieve*. As a result, it only took 0.64 hours to complete the training. When the error tolerance increased from 0.2 to 0.4, the training time dropped significantly from 8.2 hours to 2.1 hours. Subsequently, the declining rate of training time flattened. The relationship between the error tolerance and training time can be represented by a regression curve which fits into the power function with a  $R^2$  larger than 99%. It is worth to note that the coefficients of the power function will likely vary from case to case.

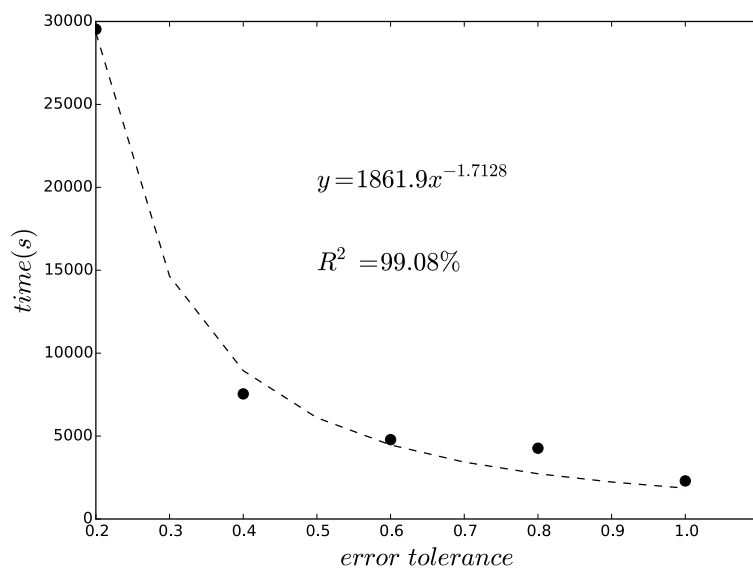


Figure 9 Relationship between error tolerance and training time

### 6.1.3 Number of Inputs

The number of inputs affects the training time as it determines the dimensions of the training domain. Also, the indoor airflow is usually sensitive to the boundary conditions, which in this study are the inputs of the ISAT-FFD. Changing the number of inputs will impact the mapping gradient matrix  $\mathbf{A}$  in Equation (10) and hence change the shape of the EOA. To show the potential application of ISAT-FFD in real control purpose, we studied nine scenarios with the number of inputs varying from 1 to 9. The detailed information of inputs is shown in Table 2. We set  $T_{occ}$  as the output and the error tolerance as 1.0, which allows a prediction error of 1.0 °C. The *automatic range refinement* method was also used in this scenario. To reduce the computing time for scenarios with large number of inputs, we set the  $\eta$  in Equation (17) as 0.005.

Table 2 Inputs and corresponding training domain in different scenarios

Scenario #	Input $\mathbf{x}$	Training Domain
1	$[T_{wall}]$	$T_{wall} \in [25.0, 30.0]$
2	$[T_{wall}, T_{floor}]$	$T_{floor} \in [25.0, 30.0]$ and the rest is the same as Scenario #1
3	$[T_{wall}, T_{floor}, T_{inlet}]$	$T_{inlet} \in [20.0, 25.0]$ and the rest is the same as Scenario #2
4	$[T_{wall}, T_{floor}, T_{inlet},  V _{inlet}]$	$ V _{inlet} \in [1.0, 2.0]$ and the rest is the same as Scenario #3
5	$\begin{bmatrix} T_{wall}, T_{floor}, T_{top}, \\ T_{inlet},  V _{inlet} \end{bmatrix}$	$T_{top} \in [25.0, 30.0]$ and the rest is the same as Scenario #4
6	$\begin{bmatrix} T_{wall}, T_{floor}, T_{top}, T_{east}, \\ T_{inlet},  V _{inlet} \end{bmatrix}$	$T_{east} \in [25.0, 30.0]$ and the rest is the same as Scenario #5
7	$\begin{bmatrix} T_{wall}, T_{floor}, T_{top}, T_{east}, \\ T_{west}, T_{inlet},  V _{inlet} \end{bmatrix}$	$T_{west} \in [25.0, 30.0]$ and the rest is the same as Scenario #6
8	$\begin{bmatrix} T_{north}, T_{south}, T_{floor}, T_{top}, \\ T_{east}, T_{west}, T_{inlet},  V _{inlet} \end{bmatrix}$	$T_{north} \in [25.0, 30.0]$ and the rest is the same as Scenario #7
9	$\begin{bmatrix} T_{block}, T_{north}, T_{south}, T_{floor}, \\ T_{top}, T_{east}, T_{west}, T_{inlet},  V _{inlet} \end{bmatrix}$	$T_{block} \in [30.0, 35.0]$ and the rest is the same as Scenario #8

As Shown in Figure 10, when one input was used (Scenario #1), the ISAT-FFD needed 9 queries to complete the training in 0.1 hours. Among them, only 2 *grow* and 3 *add* actions were needed. If the number of inputs is 4 (Scenario #4), the ISAT-FFD needed 655 queries, including 9 *add* and 95 *grow* actions. It took 2.5 hours to complete the training. Further increasing the number of inputs to 9 (Scenario #9), we found that number of queries went up to around two million consisting of 493 *grow* and 42 *add* actions. The training process took 12 hours. Figure 10 shows that power function can depict the relationship between the number of inputs and training time. It is worth to mention that the coefficients of the power function will likely vary for different cases.

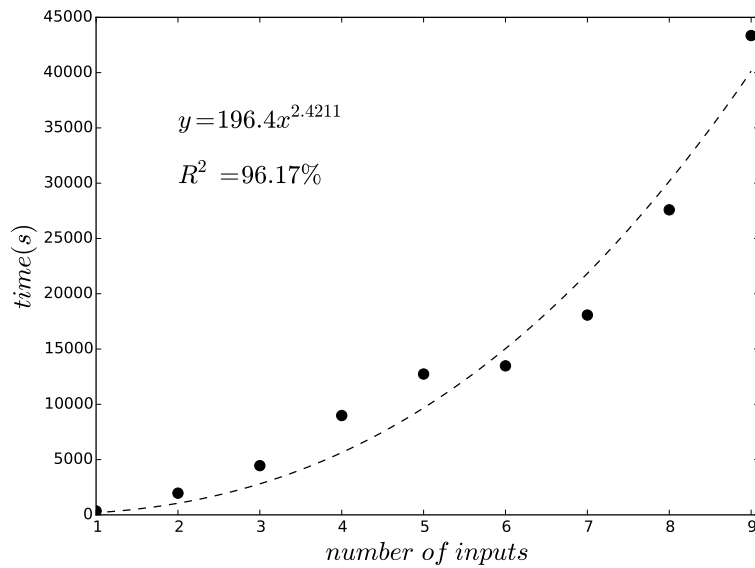


Figure 10 the relationship between number of inputs and ISAT training time

#### 6.1.4 Number of Outputs

The mapping gradient matrix  $\mathbf{A}$  defined in Equation (10) is also affected by the number of outputs. Thus, the training time of the ISAT-FFD is furthermore a function of the number of outputs. Here we defined four scenarios with the number of outputs from one to four (Table 3). When the number of output is larger than two, we used the scaling matrix

$\mathbf{B}$  to make the interpolation error of each output comparable. After applying the scaling matrix  $\mathbf{B}$ , we intended to set the single error tolerance  $\epsilon_{single}$  for each output as 0.4. As introduced before, ISAT allows only one total error tolerance  $\epsilon_{total}$ , which is calculated using the following definition:

$$\epsilon_{total} = \sqrt{\sum_{1}^n \epsilon_{single}^2} = \sqrt{n} \epsilon_{single} \quad (20)$$

where  $\epsilon_{single}$  is the error tolerance for the single output and  $n$  is the number of outputs.

Table 3 Selection of outputs, scaling matrix and total error tolerance for different scenarios

Scenario #	Output $y$	Scaling Matrix $\mathbf{B}$	$\epsilon_{total}$	Note
1	$[T_{occ}]$	No	0.4	$T_{occ} = \frac{1}{N} \sum_{z \leq 1.22} T_i$
2	$[T_{occ},  V _{occ}]$	$\begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}$	0.5657	$ V _{occ} = \frac{1}{N} \sum_{z \leq 1.22} \sqrt{u_i^2 + v_i^2 + w_i^2}$
3	$[T_{occ},  V _{occ}, T_{sensor}]$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	0.6928	Sensor location: (1.22, 1.22, 1.24) m
4	$[T_{occ},  V _{occ}, T_{sensor},  V _{sensor}]$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$	0.8	Sensor location: (1.22, 1.22, 1.24) m

In this case, we also used two inputs consisting of  $T_{wall}$  and  $T_{floor}$  and both temperatures ranged from 25 to 30 °C. We also used *automatic range refinement* method during the training.

Simulation results showed that when the number of outputs was as small as one (Scenario #1), the training can be completed with 66,049 queries within about 2 hours.

While increasing the number of outputs to four (Scenario #4), the required queries raised

roughly 64 times to 4,198,401. The number of *grow* and *add* actions increased about twice the amount to 118 and 25, respectively. Correspondingly, the training time also increased about twice to 6.74 hours. By further conducting the nonlinear regression between the training time and number of outputs, as shown in Figure 11, it was found that the time growth along the number of outputs were fitted into a power function. Again, it is important to keep in mind that the coefficient in the regression model is not universal for all cases.

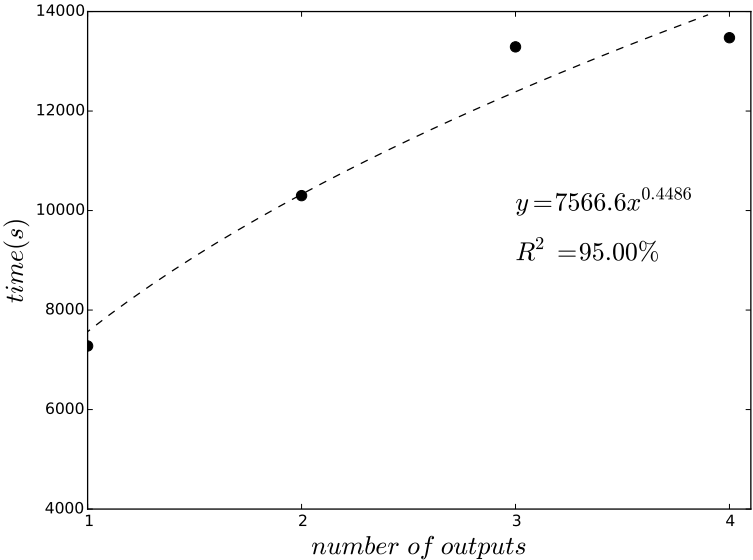


Figure 11 Relationship between the number of outputs and ISAT-FFD training time

**6.2 Performance of ISAT-FFD in Evaluation Stage**

This section evaluates the performance of a trained ISAT-FFD model by testing it with different new inputs. We first evaluated how it performed under different sizes of an evaluation domain. We then tested how the error tolerance in the training affected the aggregated local errors during the evaluation. The ISAT-FFD was first trained using four inputs and four outputs. The inputs and training domain are defined by Scenario 4 in Table 2. The outputs and error tolerance are defined by Scenario 4 in Table 3. Similarly, the *automatic range refinement* method was used to train the ISAT-FFD. Due to the high



sensitivity and turbulence of the flow, we found that it was not possible to complete the training for the entire training domain within an acceptable time. Thus, we set the  $\eta$  in Equation (17) to 0.0005, which will deem the training completed if five or less *add* or *grow* actions are generated per 10,000 queries. Even with this setting, the training process took 23.2 hours to evaluate the total 1,185,921 queries with 122 *add* and 850 *grow* actions.

### 6.2.1 Performance of ISAT under Different Evaluation Size

The variation of the evaluation domain is defined as:

$$T_{other} \in [25.0 - a, \quad 30.0 + a] \quad (21)$$

$$T_{floor} \in [25.0 - a, \quad 30.0 + a] \quad (22)$$

$$T_{inlet} \in [20.0 - a, \quad 25.0 + a] \quad (23)$$

$$|V|_{inlet} \in [1.00 - b, 2.00 + b] \quad (24)$$

where  $a$  and  $b$  are the constants defined in the Table 4. We used a normal distribution to pick the value of query points in each dimension. For instance, Figure 12 shows the distribution of probability for one of the inputs,  $T_{wall}$ . This will cause the ratio of queries residing in the training domain be higher than the ratio of evaluation domains covered by the training domain. For instance, in Scenario 1 of Table 4, only 52% of evaluation domain is covered by the training domain, but 90.7% of the evaluating queries fall into the training domain. We believe that this is closer to a real situation than the data generated by a uniform distribution. A set of 108 queries were generated based on that methodology.

Table 4 Generation of different evaluation domains

Scenario #	$a$	$b$	$S_{training}/S_{evaluation}$	Queries Within the Training Domain (Percentage)
1	0.5	0.05	52%	98 (90.7%)
2	1.0	0.10	30%	87 (80.6%)
3	1.5	0.15	19%	70 (64.8%)
4	2.0	0.20	12%	55(50.9%)
5	2.5	0.25	8%	37(34.3%)

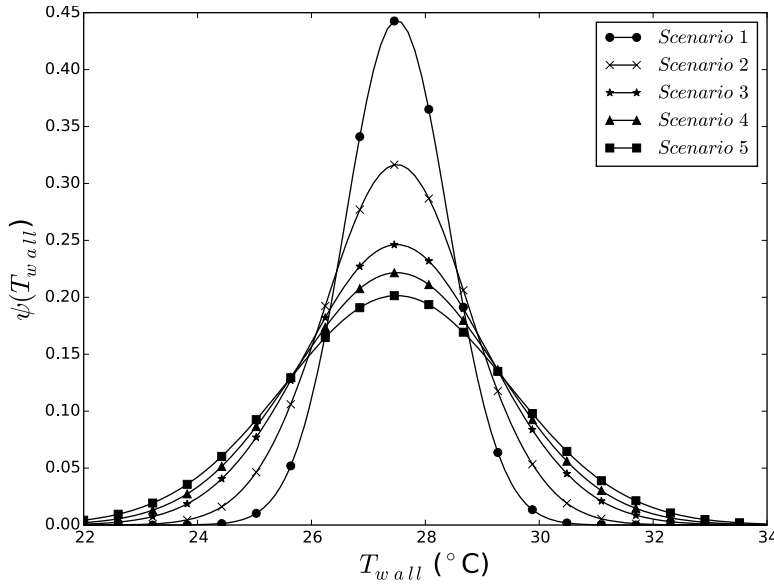


Figure 12 Normal distribution used to randomly generate query points for temperatures of other walls.

To better visualize a five-dimension data points on two-dimension plotting, we used the oval shape to represent four inputs and surface color of oval shape to show the *retrieve* error for the inputs. The coordinates of center point of ellipsoid represent the  $T_{wall}$  and  $T_{floor}$ . The red dash line represents the training domain and the blue solid line represents the evaluation domain. The remaining two inputs, consisting of  $T_{inlet}$  and  $V_{inlet}$ , are presented by half-length and half-height of the ellipsoid after normalization:

$$T^* = \frac{T_{inlet} - (20.0 - a)}{(25.0 + a) - (20.0 - a)} = \frac{T_{inlet} - 20.0 + a}{5 + 2a} \quad (25)$$

$$V^* = \frac{|V|_{inlet}}{2.00 + b} \quad (26)$$

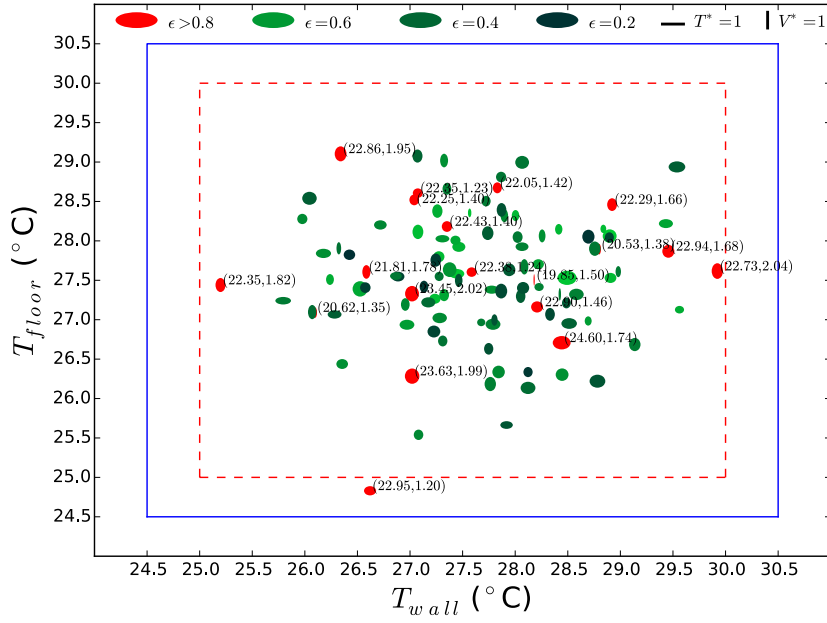
where  $a$  and  $b$  are provided in Table 4 and varies scenario by scenario.

For scenario #1, because the generated query points were largely residing inside the training domain (90%), all the queries were resolved by *retrieve* actions (Figure 13a).

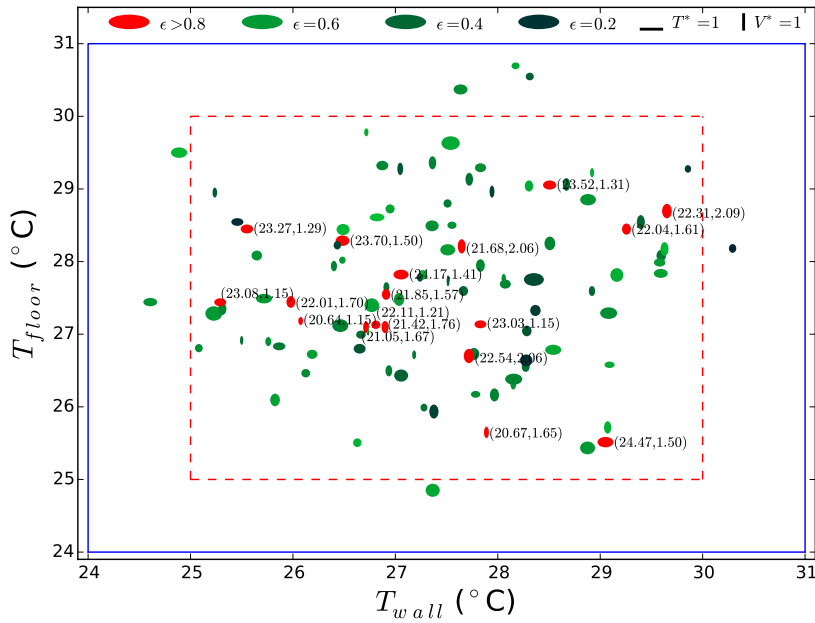
Although there were 10 points located outside the training domain, ISAT still predicted the outputs using extrapolation. As a result, the evaluation took only 0.001s, which is negligible.

To evaluate the accuracy of the ISAT-FFD model, we compared the outputs of the ISAT-FFD with the standalone FFD simulation outputs. Here we consider the FFD simulation outputs as reference. Since the *add* and *grow* actions actually returned the results of a FFD simulation, the only source of the error was from the *retrieve* action which approximated the FFD outputs using a linear interpolation. As coloured in red, 19 points were retrieved with actual error larger than the error tolerance of 0.8. The maximum actual error is 1.54, which is about 1.92 times larger than the error tolerance.

The results of Scenario #2 were similar to Scenario #1, as most of the queries (80%) were located within the training domain. There were 21 evaluation query points out of the domain. Again, ISAT can handle all the queries using *retrieve* action, as shown in Figure 13b.



a)



b)

Figure 13 Distribution of evaluation points for *retrieve* actions (a) Scenario 1 (b) Scenario 2

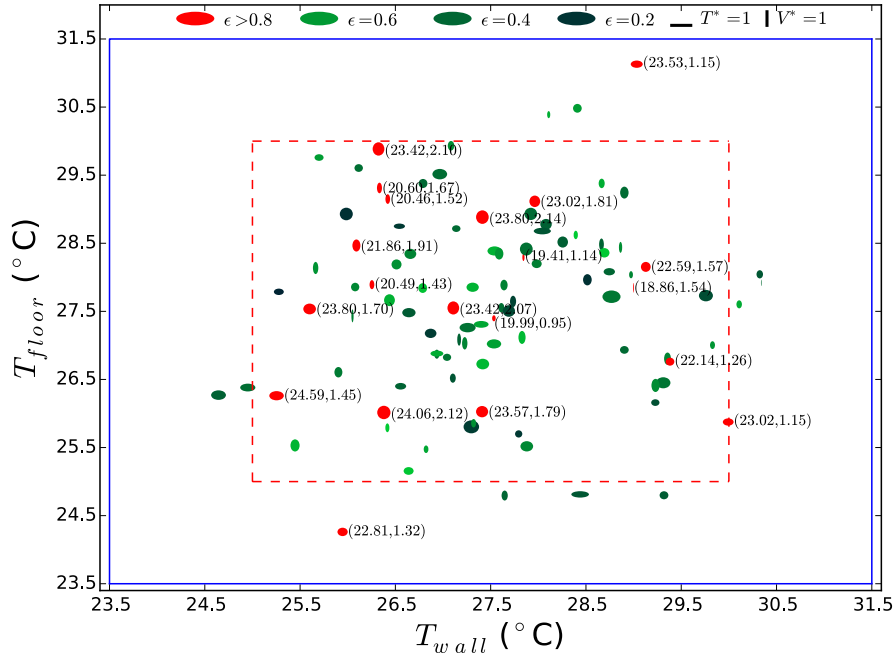
When the training domain coverage percentage dropped to 19% in Scenario #3 and 38% of the query points were outside the training domain, about 92% of query points can be handled by *retrieve* actions as shown in Figure 14a. Among them, 20 points were retrieved with actual error larger than the error tolerance of 0.8. The maximum actual error is 1.62, which is about 2.02 times larger than the error tolerance. Moreover, there are 1 *grow* and 5 *add* actions, which were all located outside the training domain, as shown in Figure 14b.

Similarly, further decreasing the coverage as low as 12% and 8% caused that 49.1% and 65.7% of evaluation points were outside the training domain, therefore less query points are evaluated by *retrieve* actions (Figure 14c and Figure 14e) and more needed either *grow* or *add* action (Figure 14d and Figure 14f).

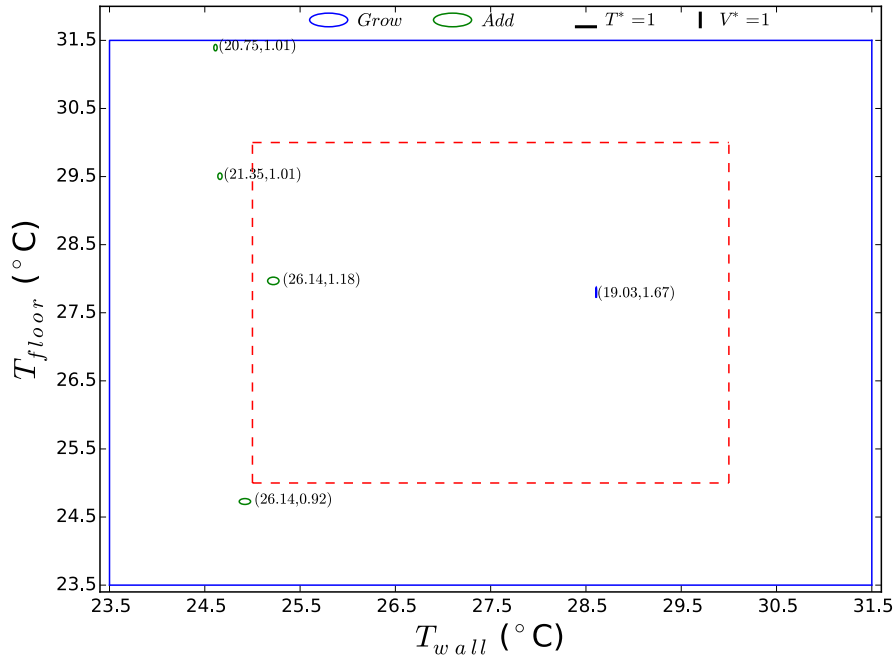
We further define speedups  $n$  to quantify how fast the ISAT-FFD can be:

$$n = \frac{T_{DI}}{T_{ISAT}} \quad (27)$$

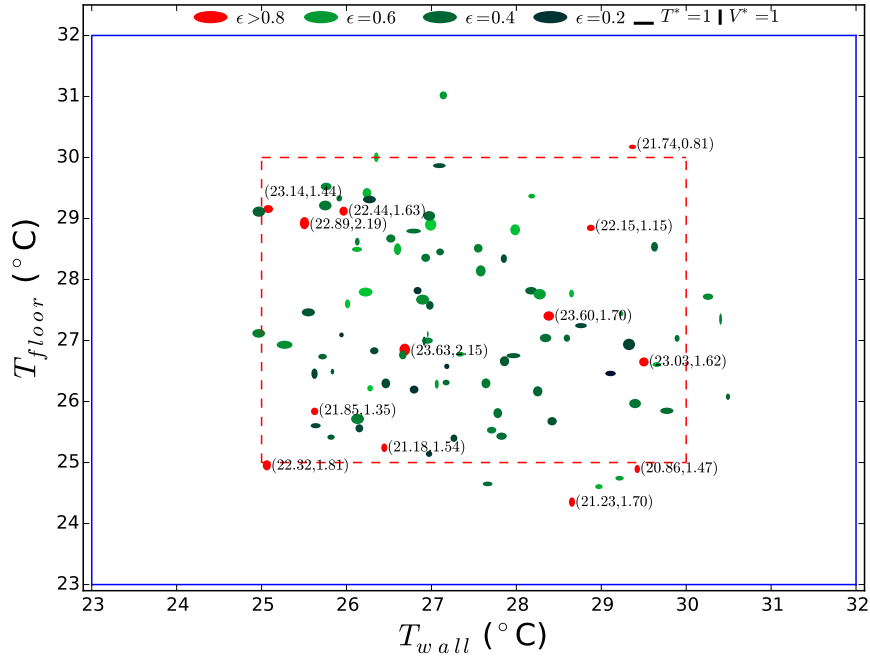
where  $T_{DI}$  is the total estimated time for all queries done by directly launching a FFD simulation;  $T_{ISAT}$  is the time cost for the all queries by the ISAT-FFD. We found that if the training domain covers over 30% of the evaluation domain and queries within training domain exceed 80.6%, the speedup can be as high as 1.5 million times. Even when the coverage percentage was as low as 8%, the speedup can be 5.9 times.



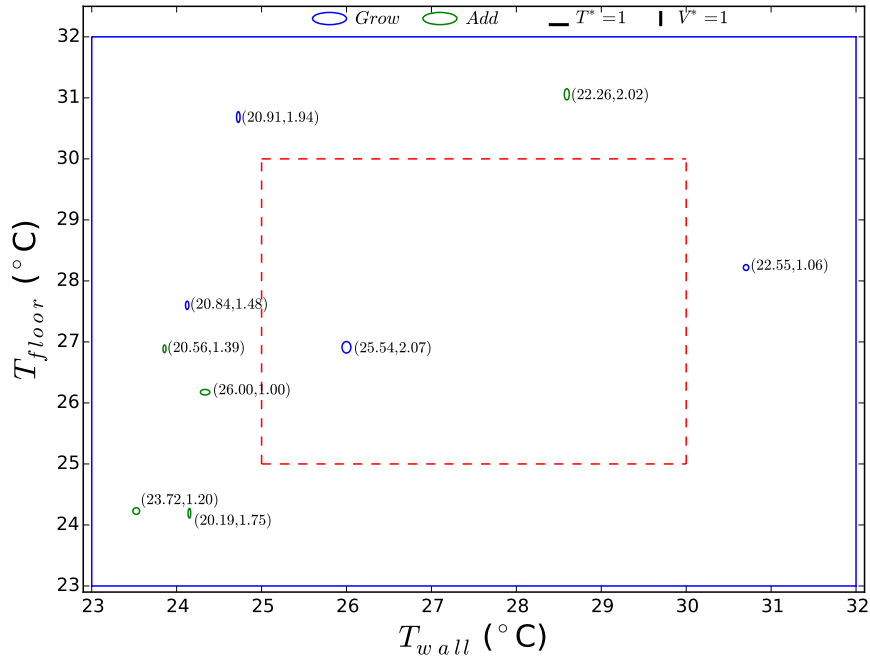
a)



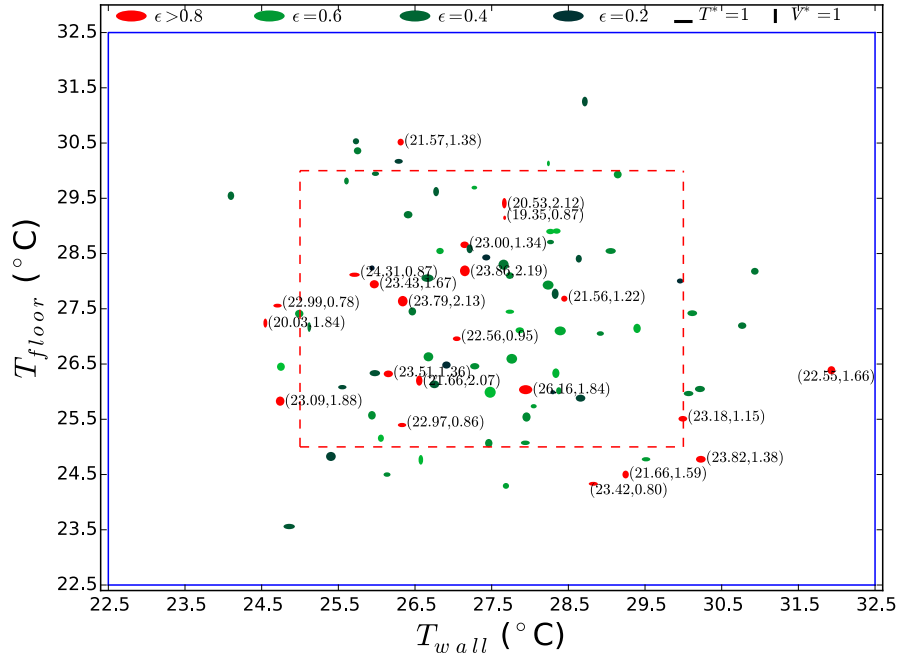
b)



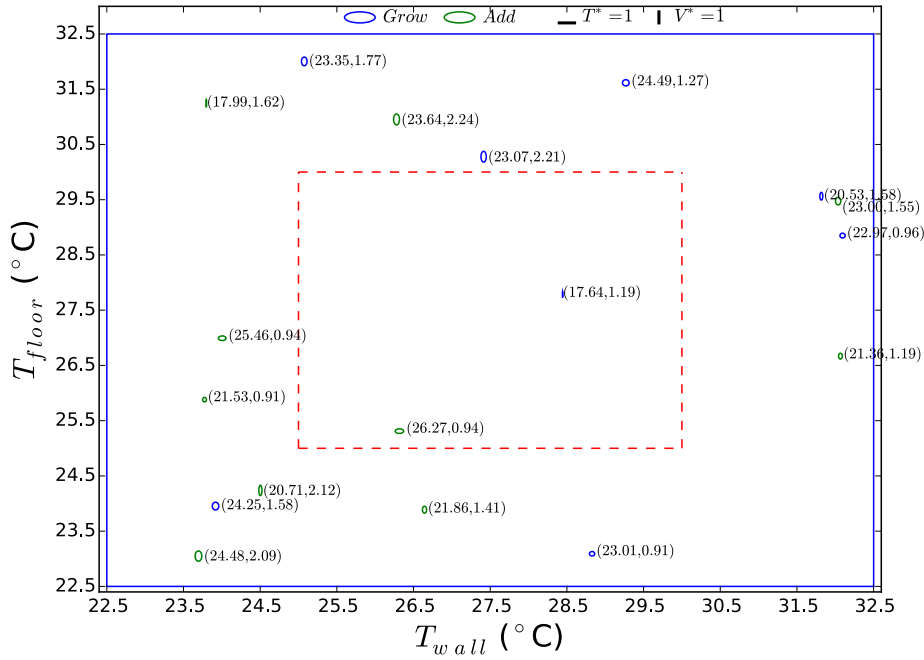
c)



d)



e)



f)



Figure 14 Distribution of evaluation points for scenario 3-5. (a) *Retrieve* points in scenario 3; (b) *Add and grow* points in Scenario 3; (c) *Retrieve* points in Scenario 4; (d) *Add and grow* points in Scenario 4; (e) *Retrieve* points in Scenario 5; (f) *Add and grow* points in Scenario 5;

### 6.2.2 Performance of ISAT-FFD under different error tolerance

The error tolerance  $\epsilon_{total}$  defined in Equation (20) was reported to have an impact on the aggregated errors during the evaluation (Pope 1997). To study their relationship in the context of airflow simulations, we performed the ISAT-FFD simulation with the error tolerance ranging from 0.8 to 2.4 with increments of 0.4. As a result, the single error tolerance for each output increased from 0.4 (0.4 K for temperature and 0.04 m/s for velocity magnitude) to 1.2 (1.2 K for temperature and 0.12 m/s for velocity magnitude). The number of inputs and outputs and training method are identical to those in Section 6.2. The evaluation domain is defined as Scenario 1 in Table 4.

Using an error tolerance of 0.8, the aggregated errors during the evaluation were 63.735 for the 108 queries. The error for one *retrieve* during the evaluation on average was 0.59, which indicated that the overall performance of the ISAT-FFD was accurate. With the highest error tolerance being that of 2.8 in training, the total aggregated errors were 110.473 for the 108 queries. On average each *retrieve* action contributed 1.02 to the accumulated error, which is about 0.37 times of the error tolerance. Figure 15 shows the regression curve between the error tolerance and mean retrieve error in the evaluation. The solid line represents the error tolerance in the training. The regression curve shows that when the error tolerance increased, the mean retrieve error in evaluation also grew to some extent, yet it was always below the error tolerance of the training. It is worth to point out

that the coefficient in the regression equation only pertains to the specific case study presented in this paper.

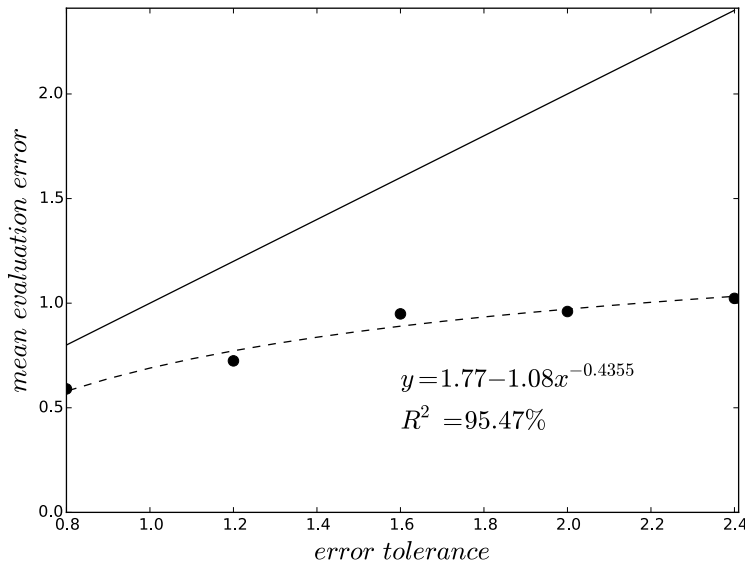


Figure 15 the relationship between accumulated error and error tolerance settings

## 7 Conclusion

In this paper we explored the feasibility of the ISAT in predicting key information for indoor airflows using FFD for airflow simulation as an example. First, the investigation showed that the *automatic interval refinement* method is an efficient approach for training ISAT. Second, the error tolerance, the number of inputs, as well as the number of outputs, can significantly impact the training time. Exploiting a trained ISAT, we found that it performed differently depending on various sizes of the training domain. Lastly, we identified that the error tolerance during training could affect the mean retrieve error during an evaluation.

To conclude, a well-trained ISAT table is capable of providing timely and reasonably accurate predictions of indoor airflows simulations. Before training the ISAT, users need to validate the high order models (e.g. CFD or FFD) using experimental data to

ensure the accuracy of flow field prediction, from which the training data is extracted. Afterwards, the validated high order models can generate the reliable training data to train the ISAT for predicting the key information. In addition, before applying ISAT into real project, a sensitivity study needs to be done to select as few inputs as possible to reduce training time. Also, it is critical to determine an appropriate error tolerance for the ISAT training since trade-offs exist between accuracy, performance, and training time. Finally, it is beneficial to make the trained domain cover the evaluation domain as much as possible in order to avoid the need of *grow* or *add* actions.

## **ACKNOWLEDGMENTS**

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Award No. DE-EE0007688. The authors at the University of Miami also received the support from the University of Miami Provost's Research Award to Wangda Zuo. The authors also thank Professor Stephen B. Pope at the Cornell University for his help in our research.

This work emerged from the Annex 60 project, an international project conducted under the umbrella of the International Energy Agency (IEA) within the Energy in Buildings and Communities (EBC) Programme. Annex 60 will develop and demonstrate new generation computational tools for building and community energy systems based on Modelica, Functional Mockup Interface and BIM standards.

## References

- Ahuja, S., A. Surana, and E. Cliff. 2011. Reduced-Order Models for Control of Stratified Flows in Buildings. *Proceedings of the Proceedings of the 2011 American Control Conference*.
- Chen, Q., and J. v. d. Kooi. 1988. Accuracy—a Program for Combined Problems of Energy Analysis, Indoor Airflow, and Air Quality. *Ashrae Transactions*, 94:196-214.
- Chen, Q., and J. Srebric. 2002. A Procedure for Verification, Validation, and Reporting of Indoor Environment CFD Analyses. *HVAC&R Research*, 8 (2):201-16.
- Chorin, A. J. 1967. A Numerical Method for Solving Incompressible Viscous Flow Problems. *Journal of Computational Physics*, 2 (1):12-26.
- Courant, R., E. Isaacson, and M. Rees. 1952. On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences. *Communications on Pure and Applied Mathematics*, 5 (3):243-55.
- Desta, T. Z., K. Janssens, A. Van Brecht, J. Meyers, M. Baelmans, and D. Berckmans. 2004. CFD for Model-Based Controller Development. *Building and Environment*, 39 (6):621-33.
- Hazyuk, I., C. Ghiaus, and D. Penhouet. 2012. Optimal Temperature Control of Intermittently Heated Buildings Using Model Predictive Control: Part I—Building Modeling. *Building and Environment*, 51:379-87.
- Hiyama, K., S. Kato, and Y. Ishida. 2010. Thermal Simulation: Response Factor Analysis Using Three-Dimensional CFD in the Simulation of Air Conditioning Control. *Proceedings of the Building Simulation*.
- Jin, M., W. Zuo, and Q. Chen. 2012. Improvement of Fast Fluid Dynamics for Simulating Airflow in Buildings. *Numerical Heat Transfer, Part B Fundamentals*, 62 (6):419-38.
- Jin, M., W. Zuo, and Q. Chen. 2013. Simulating Natural Ventilation in and around Buildings by Fast Fluid Dynamics. *Numerical Heat Transfer, Part A: Applications*, 64 (4):273-89.
- Khronos, G. "The Opencl Specification, Version 1.2."  
<https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf>.
- Kim, D., J. Braun, E. Cliff, and J. Borggaard. 2015. Development, Validation and Application of a Coupled Reduced-Order CFD Model for Building Control Applications. *Building and Environment*, 93:97-111.
- Kolokotsa, D., A. Pouliezios, G. Stavrakakis, and C. Lazos. 2009. Predictive Control Techniques for Energy and Indoor Environmental Quality Management in Buildings. *Building and Environment*, 44 (9):1850-63.
- Li, K., H. Su, J. Chu, and C. Xu. 2013. A Fast-POD Model for Simulation and Control of Indoor Thermal Environment of Buildings. *Building and Environment*, 60:150-7.
- Pope, S. B. 1997. Computationally Efficient Implementation of Combustion Chemistry Using in Situ Adaptive Tabulation. *Combust. Theory Modelling*, 1:41-63.
- Pope, S. B. 2008. Algorithms for Ellipsoids. *Cornell University Report No. FDA:08-1*.
- Singer, M., and S. Pope. 2004. Exploiting ISAT to Solve the Reaction-Diffusion Equation. *Combustion theory and modelling*, 8 (2):361-83.
- Singer, M., S. Pope, and H. Najm. 2006. Operator-Splitting with Isat to Model Reacting Flow with Detailed Chemistry. *Combustion theory and modelling*, 10 (2):199-217.

- Stockwell, D. R., and A. T. Peterson. 2002. Effects of Sample Size on Accuracy of Species Distribution Models. *Ecological modelling*, 148 (1):1-13.
- Tian, W., T. A. Sevilla, and W. Zuo. 2016. A Systematic Evaluation of Accelerating Indoor Airflow Simulations Using Cross-Platform Parallel Computing. *Journal of Building Performance Simulation*:1-13. doi: 10.1080/19401493.2016.1212933.
- Tuma, M. 2002. A Note on the LDLT Decomposition of Matrices from Saddle-Point Problems. *SIAM journal on matrix analysis and applications*, 23 (4):903-15.
- Wang, L. L., W. S. Dols, and Q. Chen. 2010. Using CFD Capabilities of Contam 3.0 for Simulating Airflow and Contaminant Transport in and around Buildings. *HVAC&R Research*, 16 (6):749-63.
- Wang, M., and Q. Chen. 2009. Assessment of Various Turbulence Models for Transitional Flows in an Enclosed Environment (Rp-1271). *HVAC&R Research*, 15 (6):1099-119.
- Wetter, M., W. Zuo, T. S. Noudui, and X. Pang. 2014. Modelica Buildings Library. *Journal of Building Performance Simulation*, 7 (4):253-70. doi: 10.1080/19401493.2013.765506.
- Yang, P. 2013. "Real-Time Building Airflow Simulation Aided by GPU and FFD." Concordia University.
- Zhai, Z., Q. Chen, P. Haves, and J. Klems. 2002. On Approaches to Couple Energy Simulation and Computational Fluid Dynamics Programs. *Building and Environment*, 37 (8-9):857-64.
- Zhai, Z., J. Srebric, and Q. Chen. 2003. Application of CFD to Predict and Control Chemical and Biological Agent Dispersion in Buildings. *International Journal of Ventilation*, 2 (3):251-64.
- Zuo, W. 2010. "Advanced Simulations of Air Distributions in Buildings." Purdue University.
- Zuo, W., and Q. Chen. 2009. Real-Time or Faster-Than-Real-Time Simulation of Airflow in Buildings. *Indoor Air*, 19 (1):33-44.
- Zuo, W., and Q. Chen. 2010a. Fast and Informative Flow Simulations in a Building by Using Fast Fluid Dynamics Model on Graphics Processing Unit. *Building and Environment*, 45 (3):747-57.
- Zuo, W., and Q. Chen. 2010b. Fast Simulation of Smoke Transport in Buildings. *Proceedings of the the 41st International HVAC&R congress*, Beograd, Serbian, December 1-3.
- Zuo, W., and Q. Chen. 2010c. Simulations of Air Distribution in Buildings by FFD on GPU. *HVAC&R Research*, 16 (6):783-96.
- Zuo, W., J. Hu, and Q. Chen. 2010. Improvements in FFD Modeling by Using Different Numerical Schemes. *Numerical Heat Transfer Part B-Fundamentals*, 58 (1):1-16. doi: 10.1080/10407790.2010.504694.
- Zuo, W., M. Jin, and Q. Chen. 2012. Reduction of Numerical Viscosity in FFD Model. *Engineering Applications of Computational Fluid Mechanics*, 6 (2):234-47.
- Zuo, W., M. Wetter, W. Tian, D. Li, M. Jin, and Q. Chen. 2016. Coupling Indoor Airflow, HVAC, Control and Building Envelope Heat Transfer in the Modelica Buildings Library. *Journal of Building Performance Simulation*, 9 (4):366-81.