VOL. 12, NO. 10, MAY 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

# UTILIZING OPEN SOURCE SOFTWARE RUNNING IN INEXPENSIVE HIGH PERFORMANCE COMPUTING SYSTEM FOR CFD APPLICATIONS

Mohamed Sukri Mat Ali[1], Low Lee Leong[1], Mohd Nurazam Ramly[1], Sheikh Ahmad Zaki Shaikh Salim[1] and Sallehuddin Muhammad[2]
[1]Wind Engineering Laboratory, Malaysia-Japan International Institute of Technology, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia
[2]Department of Engineering, UTM Razak School of Engineering and Advanced Technology, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia
E-Mail: sukri.kl@utm.my

## ABSTRACT

The high cost of conducting research is a significant issue for the successfulness of any research project. For research activities involving flow simulation, the licensing fee for the numerical software and the cost to acquire powerful machine are the main factors contributing to the high cost. This paper reports our experiences in setting up a cost effective way of doing computational fluid dynamics (CFD). The actions involve two areas, i.e., software and hardware. For the software, open source softwares are utilized, particularly the OpenFOAM® as the CFD package. For the hardware, a parallel computer made from a cluster of inexpensive desktop computer is constructed. This architecture is found able to meet our requirement in investigating various flow problems that include aeroacoustics, vibration and wind engineering for ventilation.

**Keywords:** open source, OpenFOAM, parallel computing, CFD.

## INTRODUCTION

The use of numerical simulation to solve and analyze problems are the common, if not a vital tool, in most research fields. Particularly, in fluid dynamics, numerical simulation or better known as Computational Fluid Dynamics (CFD) is an important tool to get a better insights and detailed data that for complex problems made the measurement from experiment is difficult or requires very expensive equipment.

Historically, CFD allowed only for a simple fluid problem, to the extent that three dimensional problems were almost not possible to be solved. In the aspect of speed in solving a particular fluids problem, the need for more powerful machines outweights the need for better numerical method and algorithms. However, CFD users are usually not interested to get involve in designing a machine that it purposes specifically to be used for CFD applications. They rely mostly on a machine that has been available in the market that actually have been designed to cater a wide range of market.

Search engine Google Scholar can give a general estimation on how much research have been done on 'CFD Modelling' and 'CFD Supercomputing'. As May 25 of 2015, about 128,000 articles can be found about 'CFD Modelling' if compared to only 17,300 articles on 'CFD Supercomputing'. Therefore, any study that can homogenous these two fields, i.e., CFD and computer, will definitely contribute to the CFD users in guiding them to a solution for a faster solution in solving CFD analysis.

Code developers have the total freedom to modify and distribute their code, without relying to other CFD package providers. In universities or research institutes, the computer codes are mostly developed by the postgraduate students and postdoctoral fellows with the 'advice' by the head of projects or professors. However, the code developers usually not stay with the institution for a long time, as the nature of their work is based on the period of the project. Consequently, the computer code is not sustained properly and people who inherit the computer code have the tendency to modify the code architecture differently. Hence, a 'spaghetti code' is formed that make it not practical for a general fluid problems.

With the use of Object-Oriented Programming, 'spaghetti code' can be prevented. This technique allows for the generalisation of the programming language. The generalisation is the 'object' at which the detailed language can be hidden, allowing for a shorter and simple computer architecture. Currently, most CFD packages (commercial or open source) adapt this technique.

For CFD practitioners, they usually not interested in the architecture of the CFD program. What the most important criteria for a CFD package for them is that the CFD program able to solve the fluid problem accurately without the hustle spending a long time to learn how to use the CFD package. This is what have been offered by all commercial CFD packages, and it definitely comes with a high price in the form of 'licence fees'. For those who have no grant, there is no chance for them to use the commercial CFD package.

Open source CFD package is the alternative to commercial CFD package and it is increasingly popular. The introduction of open source software to simulate fluids problems to new users always come with two main challenges or 'stigma': 1- the reliability of the software to give accurate results and 2- the time requires to learn the open source software. This paper hopefully can negate that 'stigma'.

This paper is structured with the following contains. The concept of OpenFOAM® is explained in next section. After that, the steps how to setting up a parallel computing using a cluster of workstations is given. This paper is concluded in last section.

**OpenFOAM®**

OpenFOAM was originally developed by a strong research group at The Imperial College, London. The principal developers are Henry Weller and HrvojeJasak [1]. HrvojeJasak has previously working with CD-adapco and Fluent Inc, where he has been pioneering the use of object orientation programming in commercial CFD softwares.

OpenFOAM is written in C++ and it has more than half a million lines of code. In 2004, Henry Weller has founded OpenCFDLtd and made the OpenFOAM software protected under the GNU general public license. Therefore, OpenFOAM is always be a free software and users have the full freedom to modify and redistribute the software. For a worth to read article about Open-FOAM for computational Fluid Dynamics, reader can refer to [2].
A basic step how to simulate fluid problem using OpenFOAM is presented. This step is designed for the new user of OpenFOAM. With this basic step, reader can explore OpenFOAM by itself. Here, three basic 'F-U-N' steps are introduced, with the assumption that the user has installed OpenFOAM properly beforehand. Reader is referred to this link for the guide of OpenFOAM installation: http://www.openfoam.org/download/ubuntu.php.

**Find**

OpenFOAM has an extensive set of solver that covered almost all fluid dynamics problems. Readers are referred to the official OpenFOAM Foundation webpage for the list of standard solvers available in OpenFOAM; http://www.openfoam.org/features/ standard-solvers.php.
From the list, find one solver that is very relevant to the problem that is going to be solved. For example, if the problem involves two incompressible fluid problems, and the volume of fluid (vof) is the adequate approach to solve the problem, *interFoam*is the best candidate for the solver. Then, find a related OpenFOAM tutorial case on *interFoam* to be the template case for the problem. This tutorial case can be found in $FOAM_TUTORIALS directory, i.e., /home/user/OpenFOAM/OpenFOAM-2.2.x/tutorials. For this example, the *damBreak*tutorial case is the best candidate to be the template for this problem. Copy this template directory and rename it.

**Undo**

After the suitable case template has been found, undo the necessary parameters that have been pre-set for the previous tutorial case problem so that the current problem conditions can be accommodate. The case directory of OpenFOAM consists of three main subdirectories. Figure-1 shows the structure of the OpenFOAM case directory.

For example, if the problem geometry is totally different from the one available in the tutorial case, the user can create a new geometry. There are three ways of doing that, and all these methods can be done in the subdirectory of /constant/polyMesh of the main case directory. For a simple geometry, it is advisable to use the *blockMesh*OpenFOAM's utility as it is the most basic mesh generators inOpenFOAM. For a complex geometry, user can use *snappyHexMesh*OpenFOAM's utility or using the third party mesh generators and export it toOpenFOAM using mesh conversion utilities available in OpenFOAM.

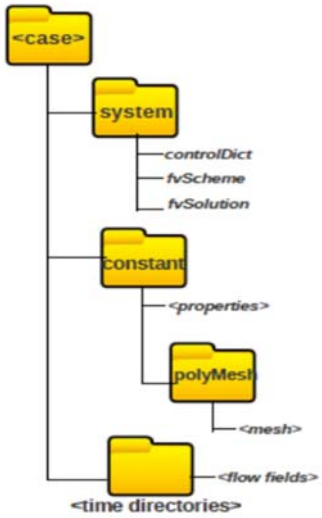



**Figure-1.** OpenFOAM case directory structure.

The same 'undo' procedure is applied if the user like to change other parameters, e.g. turbulent model, discretization schemes, boundary condition and more. Reader are referred to the OpenFOAM user guide for the general operation of the OpenFOAM: http://cfd.direct/openfoam/user-guide/.

**Number of processors**

Running OpenFOAM job can be made faster using parallel processors. Current computers usually have multi core processors. Even though OpenFOAM can be ran in a single processor, OpenFOAM is very compatible with openMPI, an open source Message Passing Interface. Thus, the user can make the full use of the computer performance to expedite the simulations. To run the OpenFOAM job in parallel, the computational domain needs to be decomposed into several smaller domains that are set by the user. This can be done easily using *decomposePar* utility where the user can set the number of processors in /system subdirectory under the main case directory. This utility splits the geometry and fields into several small computational domains, and it assigns each of the domain into different processors. Therefore, the simulation can be made quicker if more processors are used.

**PARALLEL COMPUTING**

The parallel computing can be constructed in-house to tylor the machine specifically to the user need. It also can be made cheaper but with similar, if not higher, performances than a single unit supercomputer. As mentioned by [3], the key factor in building inexpensive parallel computer is the integration between the suitable hardware with the software system. NASA Beowulf project has dedicated to this issue intensively and proposed its cluster structure system. Beowulf cluster system [4] has been used by many as the model for setting up in-house parallel computing.

In setting up a parallel computer from cluster of personal computers or workstations, five main tools have been used. These tools and its associated functions are listed in Table-1.

**Table-1.** Installation tools.

| Tools | Description |
|---|---|
| **Red Hat** | |
| Kickstart | The Red Hat Kickstart installation method is used to automatically perform unattended operating system installation and configuration. |
| SSH passwordless | The networking between the Computer is made possible after SSH passwordless is set. |
| NFS | File sharing is made using NFS. Network File System (NFS) is a distributed file system protocol. |
| TORQUE | The Terascale Open-source Resource and QUEue Manager (TORQUE) is a distributed resource manager providing control over batch jobs and distributed compute nodes. |
| Ganglia | Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. |

**Example**

Here, an example of setting up a parallel computing system using a cluster of workstations is briefly explained. The workstation is HP Z420 with the operating system of Ubuntu 14.0 LTS.

**1. Disable firewall:**
- *ufw disable*

**2. Disable and Remove AppArmor:**
- *sudo /etc/init.d/apparmor stop*
- *sudo update-rc.d -f apparmor remove*
- *sudo apt-get remove apparmor apparmor-utils-y*

**3. Set up an NFS mount: On head node:**
- *sudo apt-get update*
- *sudo apt-get install nfs-kernel-server*
- *mkdir /mnt/ WEE*
- *sudogedit /etc/exports*
- add this line: */mnt/WEE*10.0.0.1/255.255.255.0
- *(rw, sync, no_root_squash, no_subtree_check)*
- reboot

**4. Set up an NFS mount:**
On compute node:
- *sudo apt-get update*
- *sudo apt-get install nfs-common*
- *mkdir /mnt/ WEE*
- *sudogedit /etc/fstb*
- add this line:
- 10.0.0.1 *:/mnt/WEE*
- */mnt/WEE n f s defaults* 0 0
- reboot

**5. Set up SSH keys passwordless:**
- *sudo apt-get install openssh-server openssh-client*
- *ssh-keygen*
- press enter twice
- *cd .ssh*
- *cp id_rsa.pub authorized_keys*
- *chmod 744 authorized_keys*
- *scpauthorized_keys open f oam@*10.0.0.2: */home/open f oam/.ssh*

**6. Add all the hostname to */etc/hosts*:**
- *sudo /etc/init.d/apparmor stop*
- *sudoupdate -rc.d - f apparmor remove*
- *sudo apt-get remove apparmor apparmor-utils-y*

After all these steps have been successfully done, shut-down all machines and then power up the Head Node and follow by the Compute Node.

**Comparison**

A comparison study has been made to assess if the in-house parallel computing has a better performance than the built-in supercomputer. The problem geometry is flow over a square cylinder at high Reynolds Number. The total number of cell is 200,000.

In Wind Engineering for Environment (WEE) laboratory of MJIIT, UTM Kuala Lumpur Campus, there are five (5) parallel computing systems at which three are clustered from five workstations, one is clustered from six workstations and another is clustered from seven workstations. For this performance comparison, even though up to 20 CPUs can be used, only eight(8) CPUs are used for both computing systems (HPC UTM and In-house).

**Table-2.** Performance comparison between high performance computer and in-house parallel computing system.

| Item | HPC UTM | In-house (WEE Lab) |
|---|---|---|
| CPU type | Intel Xeon DP | Intel Xeon Processor E5-1620 |
| RAM(GHz) | 3.0 | 3.6 |
| CPUs | 16 x 4 | 4 |
| Time | 5 days | 3 Day |

From Table-2, the in-house parallel computing outperform the HPC UTM. To date, WEE laboratory has simulated various flow problems. The cases that require high performance machine are the cases that involve more than 5 million of cells, moving mesh problem, multiphase and noise. WEE laboratory has the access to three supercomputers from different providers (HPC-UTM, MyREN-Malaysia and ERSA-Australia).But, thein-house parallel computing system is still the better choice as it simulates faster than these supercomputers when using the same number of processing nodes.

## COMPARISON AND VALIDATION

The reliability of open source software in simulating various flow problems can be assessed using comparisons and/or grid convergence index (GCI) studies. One of the famous benchmark case is flow over a square cylinder. This problem, even though is simple, it is challenging to model accurately the separation points, free shear layers and von Karman vortex that is formed downstream of the square cylinder.

Ali *et al*. [5] assessed the accuracy of OpenFOAM simulation for flow over a square cylinder. Their comparison study with others showed in a good agreement. Table-3 below summarises the results.

**Table-3.** Comparison of [5] with other similar studies [5].

| Author | Method | $CL_{rms}$ | $CD_{mean}$ | St |
|---|---|---|---|---|
| Okajima, 1982, Sohankar *et al.*, 1999 | Experiments | - | 1.40 | 0.148 to 0.155 |
| Doolan, 2009 | DNS | 0.296 | 1.44 | 0.156 |
| Sohankar *et al.*, 1998 | DNS | 0.230 | 1.44 | 0.165 |
| Inoue *et al*., 2006 | DNS | 0.282 | 1.40 | 0.151 |
| Ali *et al.* 2009 | DNS | 0.285 | 1.47 | 0.161 |

Additionally, [5] also assessed the accuracy of the results by using the grid convergence index. Grid convergence index (GCI) is an assessment method for proper representation of the accuracy of the results. It is based on the Richardson extrapolation technique. Using this method, Ali et al. obtained GCI of less than 5%.

## EXAMPLE

### Noise radiated from flow over a blunt trailing edge

Haris *et al*. [6] used CFD techniques to simulate a flat plate for investigation aeroacoustic phenomenon from flow past a blunt trailing edge. The acoustics analysis requires accurate value of the boundary layer that developed at the trailing edge of the plate. Haris et al. [6] compared velocity profiles obtained using DNS of OpenFOAM with the theoretical values. Three theoretical approaches are used, i.e., parabolic, cubic and sine wave; which is can be obtained from the following equations, respectively:

$$\frac{u}{U_\infty} = 2\eta - \eta^2 \tag{1}$$

$$\frac{u}{U_\infty} = 1.5\eta - \frac{\eta^3}{2} \tag{2}$$

$$\frac{u}{U_\infty} = \sin\left(\pi\frac{\eta}{2}\right) \tag{3}$$

where $\eta$ is $y/\delta$ and $\delta$ is the boundary layer thickness and can be estimated theoretically from equation4.

$$\frac{\delta}{x} = \frac{5.5}{\mathrm{Re}_x^{\frac{1}{2}}} \tag{4}$$

Figure-2 indicates the boundary layer profile obtained from the numerical simulation using OpenFOAM agree well with the theoretical values.

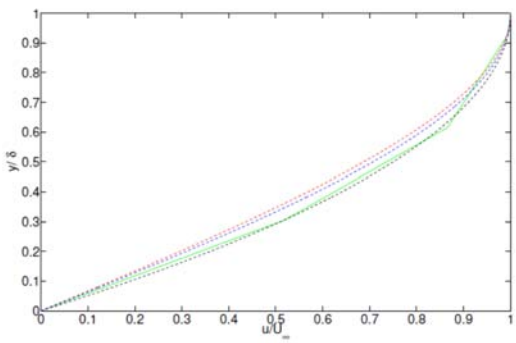



**Figure-2.** Velocity profile at 7D streamwise locations. Green-solid line: DNS, Black-broken line: parabolic, Red-broken line: cubic, Blue-broken line: sine wave [6].

**Flow over two rectangular prisms in tandem**

Prime *et al*. [7] investigate the effect of nose geometry, inter-carriage spacing and yaw angle on the aerodynamic characteristics of two prisms using OpenFOAM. Figure-4 shows the time-averaged streamwise velocity profile at 4.3D location above the prisms surfaces. The simulation is done using k-Ω SST of OpenFOAM. The data agrees well with experimental hotwire data measured in the Anechoid Wind Tunnel (AWT) at The University of Adelaide.

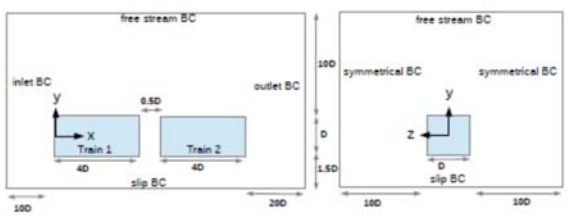



**Figure-3.** Two prisms arranged in tandem [7].

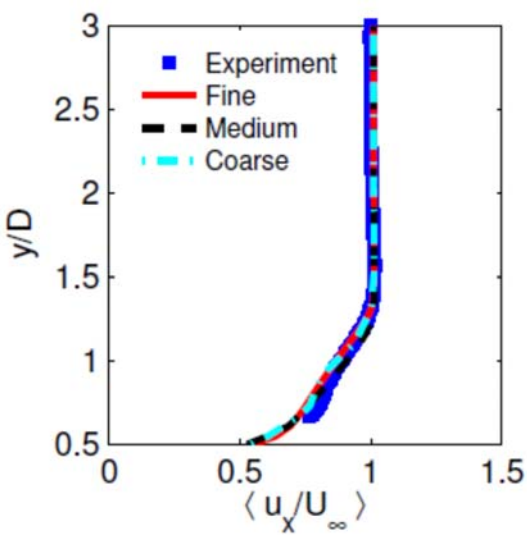



**Figure-4.** Time-averaged streamwise velocity at 4.3D streamwise location above the trains surfaces [7].

**Noise radiated from flow over bluff body**

Ali *et al*. [8] explored noise control for bluff body with four different shapes, i.e.; Triangle, Ellipse, Circular and Square. The grid solution is similar to the finest grid solution used by the extensive grid refinement study of [5].

The sound directivities are compared with the result of [9] who directly calculated the sound radiation using compressible DNS analysis (see Figure-5). In this study, the sound directivities are obtained using this two-dimensional Curle's equation:

$$p'(x,t) = \frac{x_i}{2^{3/2}\pi c_0^{1/2} r^{3/2}} \int_{-\infty}^{\tau} \left[ \frac{\partial F_i}{\partial t'} \right] \frac{dt'}{\sqrt{\tau - t'}} \tag{5}$$

The time gradient of lift fluctuations $\left(\frac{\partial F_i}{\partial t'}\right)$ is obtained from the direct numerical simulation of the near field and $\tau = t - r/c_0$ is the retarded time. The flow simulation is done using DNS of OpenFOAM.

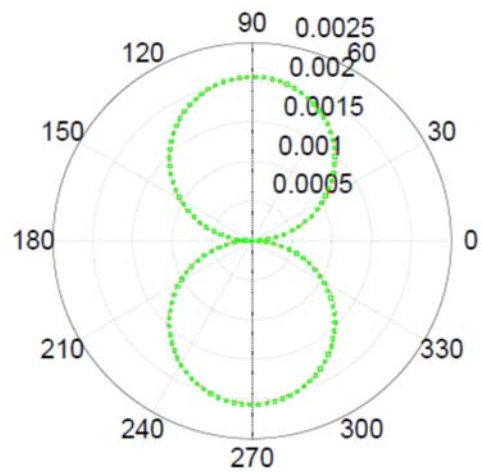
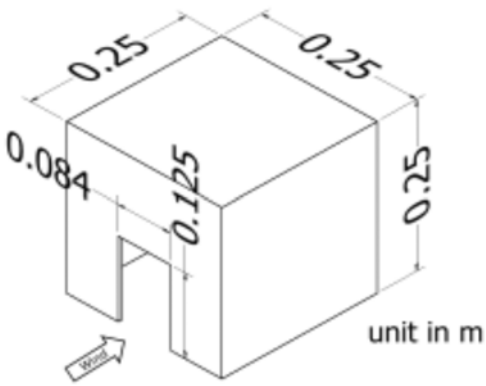



**Figure-5.** Sound directivity comparison between current study and DNS of [9] for a circular cylinder at R=75D [8].

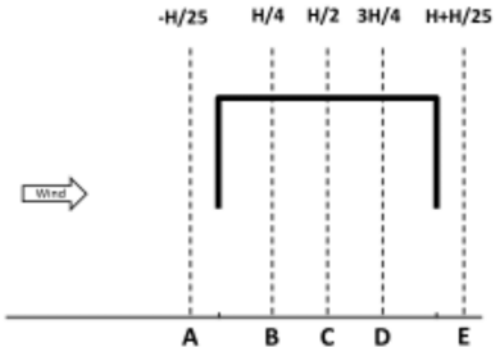
The result is in good agreement with the observations of acoustic analysis by [9]. Thus, it indicated that OpenFOAM is able to reproduce the same results.

**Wind flow over a repeating unit of simplified urban model**

Kasim *et al*. [10] made a validation study for ventilation performance of different layout. Figure-6 shows the problem geometry. This case is made similar as [11] so that a comparison study can be made.




a) schematic view of building




b) sampling locations

**Figure-6.** Wind ventilation over a repeating unit of simplified urban model [10] three different RANS turbulent models has been compared with the experimental measurement of [11]. Figure-7 shows the comparison results. The simulations, that are obtained using OpenFOAM CFD package, are in good agreement with the experimental data.

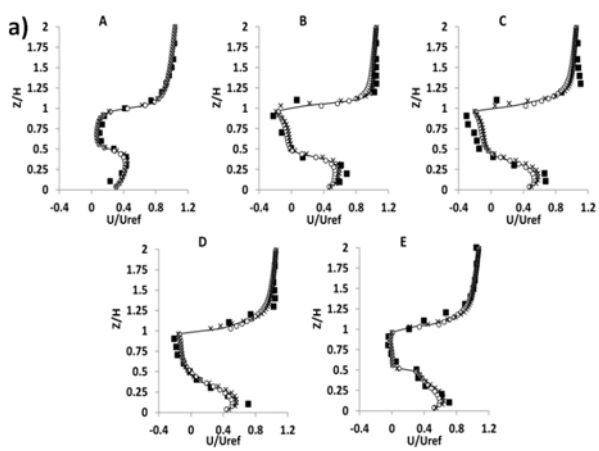



**Figure-7.** Comparison between OpenFOAM results [10] and experiment measurement of [11]. Normalized streamwise mean velocity profile. Square: Wind Tunnel [11], Circular: standard k-epsilon, Cross: SST k-Omega, Solid line: RNG [10].

## CONCLUSIONS

The use of open source software for CFD applications is emphasized by suggesting OpenFOAM CFD package. The basic steps to start with OpenFOAM is presented by introducing three simple steps of *F-U-N*. The cost reduction when using the CFD open source is more when it is running on the inexpensive parallel computing system of cluster workstations. The basic step to setting up this machine is presented. The tools require to setting up this machine is also free.

## ACKNOWLEDGEMENT

This article is part of the project supported by the GUP grant no: Q.K130000.2643.09J79 and FRGS grant no: R.K130000.7843.4F479. The use of HPC-UTM is also acknowledged.

## REFERENCES

[1] Weller H. G., Tabor G., Jasak H., and Fureby C. 1998. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in Physics. 12(6): 620-631.

[2] Chen G., Xiong Q., Morris P. J., Paterson E. G., Sergeev A., and Wang Y. 2014. OpenFOAM for computational fluid dynamics. Notices of the AMS. 61(4): 354-363.

[3] Uthayopas P., Angskun T., and Maneesilp J. 1998. Building a parallel computer from cheap PCs: SMILE cluster experiences. Proceedings of the Second Annual National Symposium on Computational Science and Engineering, Bangkok, Thailand.

[4] Sterling T. L. 2002. Beowulf cluster computing with Linux. MIT press.

[5] Ali M. S. M., Doolan C. J., and Wheatley V. 2009. Grid convergence study for a two-dimensional simulation of flow around a square cylinder at a low reynolds number. Seventh International Conference on CFD in the Minerals and Process Industries, CSIRO (CSIRO Australia, Melbourne, Australia).

[6] Haris M., Aishah S. N., Ali M., Sukri M., Salim S., Zaki S. A., and Mahzan M. I. 2014. Numerical simulation of noise radiated from a blunt trailing edge. Applied Mechanics and Materials. 629: 3-8.

[7] Prime Z., Moreau D. J., Doolan C. J., Ali M. S. M., Salleh S. M., and Haris S. N. A. M. 2014. Flow modelling and noise generation of interacting prisms. 20th AIAA/CEAS Aeroacoustics Conference (16 Jun 2014-20 Jun 2014: Atlanta, GA).

[8] Ali M. S. M., Salim S. A. Z. S., Ismail M. H., Muhamad S. and Mahzan M. I. 2013. Aeolian tones radiated from flow over bluff bodies. Open Mechanical Engineering Journal. 7: 48-57.

[9] Inoue O. and Hatakeyama N. 2002. Sound generation by a two-dimensional circular cylinder in a uniform flow. Journal of Fluid Mechanics. 471: 285-314.

[10] Kasim N. F. M., Zaki S. A., Hagishima A., Ali M. S. M., Shirakashi M., Arai N., and Razak A. A 2014. CFD Study of Cross Ventilation Performance of Different Buildings Layouts.

[11] Jiang Y., Alexander D., Jenkins H., Arthur R., and Chen Q. 2003. Natural ventilation in buildings: measurement in a wind tunnel and numerical simulation with large-eddy simulation. Journal of Wind Engineering and Industrial Aerodynamics. 91(3): 331-353.

[12] Doolan C. J. 2009. Flat-plate interaction with the near wake of a square cylinder. AIAA journal. 47(2): 475-479.

[13] Evola G. and Popov V. 2006. Computational analysis of wind driven natural ventilation in buildings. Energy and Buildings. 38(5): 491-501.

[14] Inoue O., Iwakami W. and Hatakeyama N. 2006. Aeolian tones radiated from flow past two square cylinders in a side-by-side arrangement. Physics of Fluids. 18(4): 046104.

[15] Okajima A. 1982. Strouhal numbers of rectangular cylinders. Journal of Fluid Mechanics. 123: 379-398.

[16] Sohankar A., Norberg C., and Davidson L. 1998. Low-Reynolds-number flow around a square cylinder at incidence: study of blockage, onset of vortex shedding and outlet boundary condition. International journal for numerical methods in fluids. 26(1): 39-56.

[17] Sohankar A., Norberg C., and Davidson L. 1999. Simulation of three-dimensional flow around a square cylinder at moderate Reynolds numbers. Physics of Fluids. 11(2): 288-306.