A Comparison of Interpolation Methods in Fast Fluid Dynamics

Leonard, Eric; Qiao, Hongtao; Nabi, Saleh

TR2021-118 October 19, 2021

Abstract

This paper examines the use of OpenFOAM's cellPoint and cellPointWallModified interpolation routines in fast fluid dynamics (FFD). In addition, an arrangement of OpenFOAM's data structures is proposed which speeds up FFD and an adjustment to FFD which incorporates the Boussinesq approximation is discussed. The findings of this paper show that the use of the cellPointWallModified interpolation routine in FFD produces results closer to reference data in higher Reynolds number flows than the use of the cellPoint interpolation routine, a result which highlights the important role that interpolation plays in the accuracy of FFD.

International High Performance Buildings Conference 2021

A Comparison of Interpolation Methods in Fast Fluid Dynamics

Eric LEONARD, Hongtao QIAO*, Saleh NABI

Mitsubishi Electric Research Laboratories,
Cambridge, MA, USA
rick.leonard8@gmail.com, qiao@merl.com, nabi@merl.com

* Corresponding Author

ABSTRACT

This paper examines the use of OpenFOAM's cellPoint and cellPointWallModified interpolation routines in fast fluid dynamics (FFD). In addition, an arrangement of OpenFOAM's data structures is proposed which speeds up FFD and an adjustment to FFD which incorporates the Boussinesq approximation is discussed. The findings of this paper show that the use of the cellPointWallModified interpolation routine in FFD produces results closer to reference data in higher Reynolds number flows than the use of the cellPoint interpolation routine, a result which highlights the important role that interpolation plays in the accuracy of FFD.

1. INTRODUCTION

The efficient use of air conditioners in buildings is an important, ongoing area of research. In order to develop accurate models of the performance of an air conditioner, it is necessary to incorporate information about the room that the air conditioner cools. The accuracy of a coupled simulation which models an air conditioner as well as the dynamics of the air flow in a room induced by the air conditioner can be enhanced through the use of computational fluid dynamics (CFD). By using a CFD simulation to generate detailed information about the dynamical and thermal patterns of air flow in a room generated by an air conditioner, researchers can, for instance, determine locations for the air conditioner which will lead to more efficient performance. Although there are great benefits to using CFD in a coupled simulation which models the performance of an air conditioner and the room that it cools, CFD simulations can be slow as described in Qiao et al. (2019).

Concerns around the speed of CFD have facilitated an interest in an alternative algorithm to traditional computational fluid dynamics called fast fluid dynamics (FFD) in which the solutions to the Navier-Stokes equations are approximated through the use of a time-splitting method which generates separate equations for the dynamics related to advection, diffusion, and pressure gradients as described in Zuo (2010). The authors of this paper have implemented a version of FFD using the open-source software toolbox OpenFOAM which is in line with the work of Liu et al. (2016). One particularly interesting aspect of FFD is that the advection equation that it generates necessitates the use of an interpolation routine to determine intermediate velocities at points not located at the cell centers of meshes designed to cover the domain of interest. Following this line of inquiry, the current paper compares and contrasts the use of two of OpenFOAM's interpolation routines - cellPoint and cellPointWallModified - in the FFD algorithm. In addition, this paper presents a novel use of OpenFOAM's data structures which speeds up FFD and discusses the incorporation of the Boussinesq approximation into FFD.

2. METHODOLOGY

The following section introduces FFD as well as describes a way in which OpenFOAM's data structures can be arranged to speed-up FFD. Incorporation of the Boussinesq approximation into FFD is also discussed.

2.1 Basic Fast Fluid Dynamics Algorithm

When considering only pressure and velocity in incompressible flow and with an index repeated in a term implying summation of that index over the number of relevant dimensions, the Navier-Stokes equations are given by Equation (1).

$$\frac{\partial U_i}{\partial x_i} = 0 \tag{1a}$$

$$\frac{\partial U_i}{\partial t} + U_j\frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial^2 U_i}{\partial x_j \partial x_j} + \frac{1}{\rho}B_i \tag{1b}$$

The fast fluid dynamics algorithm breaks up the momentum equation, Equation (1b), into three separate equations when in discretized form, one equation each for the effects of advection, diffusion, and pressure gradients. The time-splitting approach of FFD can be quantitatively described by Equation (2).

$$\frac{U_i^* - U_i^n}{\Delta t} = -U_j^n\frac{\partial U_i^*}{\partial x_j} \tag{2a}$$

$$\frac{U_i^{**} - U_i^*}{\Delta t} = \nu\frac{\partial^2 U_i^{**}}{\partial x_j \partial x_j} + \frac{1}{\rho}B_i \tag{2b}$$

$$\frac{U_i^{n+1} - U_i^{**}}{\Delta t} = -\frac{1}{\rho}\frac{\partial p^{n+1}}{\partial x_i} \tag{2c}$$

To ensure that the velocity field is divergence-free at time step $n + 1$, one can take the divergence of Equation (2c) and utilize Equation (1a) to derive a Poisson equation for the pressure as given in Equation (3).

$$\frac{\partial^2 p^{n+1}}{\partial x_i \partial x_i} = \frac{\rho}{\Delta t}\frac{\partial U_i^{**}}{\partial x_i} \tag{3}$$

FFD then solves Equation (2a) to obtain U_i^*, Equation (2b) to obtain U_i^{**}, Equation (3) to obtain p^{n+1}, and Equation (2c) to obtain U_i^{n+1}. While FFD solves Equation (2b) and Equation (3) by standard means, FFD solves Equation (2a) via a semi-Lagrangian scheme: Equation (2a) can be taken to represent the idea that a fluid particle's velocity will remain unchanged as it advects from a departure point $\mathbf{x}_d$ where it has velocity $\mathbf{U}^n$ to an arrival point $\mathbf{x}_a$ where it has velocity $\mathbf{U}^*$. The departure point $\mathbf{x}_d$ can be approximated using known variables as in Equation (4).

$$\mathbf{x}_d = \mathbf{x}_a - \Delta t \cdot \mathbf{U}^n(\mathbf{x}_a) \tag{4}$$

$\mathbf{U}^*$ at the arrival point $\mathbf{x}_a$, $\mathbf{U}^*(\mathbf{x}_a)$, can be related to $\mathbf{U}^n$ at the departure point $\mathbf{x}_d$, $\mathbf{U}^n(\mathbf{x}_d)$, as in Equation (5).

$$\mathbf{U}^*(\mathbf{x}_a) = \mathbf{U}^n(\mathbf{x}_d) \tag{5}$$

Using Equation (4), Equation (5) can be rewritten as in Equation (6).

$$\mathbf{U}^*(\mathbf{x}_a) = \mathbf{U}^n\big(\mathbf{x}_a - \Delta t \cdot \mathbf{U}^n(\mathbf{x}_a)\big) \tag{6}$$

By taking arrival points to be the cell centers of the mesh covering the domain of interest, Equation (6) can be used to determine $\mathbf{U}^*$ in the FFD algorithm. Because the departure point $\mathbf{x}_d$ will, in most instances, not coincide with a cell center as in Figure 1, $\mathbf{U}^n(\mathbf{x}_d)$ has to be determined through interpolation as in Liu et al. (2016).

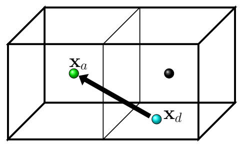


Figure 1: Illustration of a departure point (●) advecting to an arrival point (●) which coincides with a cell center; ● denotes a generic cell center

The authors of this paper implemented FFD using OpenFOAM following the work of Liu et al. (2016). OpenFOAM has a number of different interpolation routines which can be used to determine $\mathbf{U}^n(\mathbf{x}_d)$, two of which are the cellPoint and cellPointWallModified routines. Once the cell to which an interpolation point belongs is determined, the cellPoint method, as discussed in an OpenFOAMWiki (2010), works by covering each face of the cell with triangles, forming tetrahedrons composed of these triangles and the cell center, determining which tetrahdron encloses the interpolation point, and performing linear interpolation using inverse distance weights. The cellPointWallModified method, as documented in the OpenFOAM-7 Source Code (2019), is defined only when the interpolated variable is a vector (velocity, for instance) and works similarly to the cellPoint method except that the values of the interpolated variable at points on the boundaries of the domain are extrapolated from their values at the cell centers and then modified in such a way that they do not point out of the domain. Figure 2 compares and contrasts these two interpolation routines.

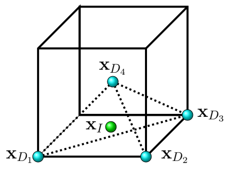


Figure 2: Data points $\mathbf{x}_{D_1}$ - $\mathbf{x}_{D_4}$ (●) used to evaluate a fluid property at an interpolation point $\mathbf{x}_I$ (●); if a data point lies on a boundary face, the cellPoint interpolation routine considers the value of the fluid property at that point to be as prescribed by the relevant boundary condition while the cellPointWallModified interpolation routine bases the value of the fluid property at the boundary data point on the value at the cell center

The cellPointWallModified interpolation routine significantly improves the accuracy of FFD in the following way: suppose, along the lines of Figure 2, $\mathbf{x}_{D_i}$, $i = 1, \ldots, 4$ are the positions of points used to perform interpolation in which $\mathbf{x}_{D_1}$, $\mathbf{x}_{D_2}$, and $\mathbf{x}_{D_3}$ are cell vertices of a face which lie on the boundary of the mesh, $\mathbf{x}_{D_4}$ is a cell center, and $\mathbf{x}_I$ is the interpolation point. Suppose also that f is a component of velocity which is to be interpolated and $d(\mathbf{x}_{D_i}, \mathbf{x}_I)$ denotes the distance between $\mathbf{x}_{D_i}$ and $\mathbf{x}_I$. With $\Delta \equiv \sum_{i=1}^{4} \frac{1}{d(\mathbf{x}_{D_i},\mathbf{x}_I)}$, then, without loss of generality for the following argument, the value of the velocity component f at $\mathbf{x}_I$, f_I, can be taken to be based on the values of f at the four data points, f_{D_1} - f_{D_4}, as in Equation (7).

$$f_I = \frac{1}{\Delta}\left(\frac{f_{D_1}}{d(\mathbf{x}_{D_1}, \mathbf{x}_I)} + \frac{f_{D_2}}{d(\mathbf{x}_{D_2}, \mathbf{x}_I)} + \frac{f_{D_3}}{d(\mathbf{x}_{D_3}, \mathbf{x}_I)} + \frac{f_{D_4}}{d(\mathbf{x}_{D_4}, \mathbf{x}_I)}\right) \tag{7}$$

If, as previously specified, $\mathbf{x}_{D1}$, $\mathbf{x}_{D2}$, and $\mathbf{x}_{D3}$ lie on a boundary face and f_I lies close to this face and away from the cell center then, according to Equation (7), $\frac{1}{d(\mathbf{x}_{D_1},\mathbf{x}_I)}, \frac{1}{d(\mathbf{x}_{D_2},\mathbf{x}_I)}, \frac{1}{d(\mathbf{x}_{D_3},\mathbf{x}_I)} >> \frac{1}{d(\mathbf{x}_{D_4},\mathbf{x}_I)}$ and $f_I \approx \frac{1}{\Delta}\left(\frac{f_{D_1}}{d(\mathbf{x}_{D_1},\mathbf{x}_I)} + \frac{f_{D_2}}{d(\mathbf{x}_{D_2},\mathbf{x}_I)} + \frac{f_{D_3}}{d(\mathbf{x}_{D_3},\mathbf{x}_I)}\right)$. If, in addition, the values of f_{D_1}, f_{D_2}, and f_{D_3} are taken to be those prescibed by the relevant boundary conditions, as in the cellPoint interpolation routine, then the value of f_I will not be accurately approximated in high Reynolds number flows because of a failure to take into account the steep velocity gradients that exist near boundaries as noted by Kundu et al. (2012). The cellPointWallModified interpolation routine rectifies this problem near the boundary by basing f_{D_1}, f_{D_2}, and f_{D_3} on the value of f in the cell center, f_{D_4}, when f_{D_1}, f_{D_2}, and f_{D_3} lie on a boundary face. Because f_{D_4} represents the value of f at an interior point in the flow, the cellPointWallModified routine better resolves the steep velocity gradients present near boundaries in high Reynolds number flows. In order to observe the differences in performance when used in FFD, the cellPoint and cellPointWallModified interpolation routines were tested on three different cases and the outcomes of these tests are presented in the Results section of this paper.

Because the present authors implemented FFD in OpenFOAM, an arrangement of OpenFOAM's data structures which sped-up the authors' implementation of FFD is now described. OpenFOAM enables its users to represent the velocity at the cell centers of a mesh through the use of data structures called *Fields*, as documented in the OpenFOAM Programmer's Guide (2015). In addition, the authors' implementation of FFD solved the matrix equations generated by Equation (2b) through the use of an iterative solver which used as its initial guess for $\mathbf{U}^{**}$ at time step $n + 1$ the value of $\mathbf{U}^{**}$ at time step n, as documented in the OpenFOAM User Guide (2019). By using separate *Fields* for $\mathbf{U}$, $\mathbf{U}^{*}$, and $\mathbf{U}^{**}$, the solution to Equation (2b) was sped-up because at each time step the initial guess for $\mathbf{U}^{**}$ was its value at the previous time step and as a simulation approached steady-state fewer iterations were needed to solve Equation (2b) as the change in $\mathbf{U}^{**}$ diminished.

2.2 Incorporation of the Boussinesq Approximation into FFD

When performing simulations of air flow in rooms related to the performance of air conditioners, it is clearly important to account for thermal effects. Use of the Boussinesq approximation allows for the incorporation of thermal effects into CFD without having to solve the fully compressible Navier-Stokes equations. In particular, the Boussinesq approximation assumes that the fluid's density is constant except when generated by buoyant forces and it is noteworthy that Gray and Giorgini (1976) demonstrated the validity of the Boussinesq approximation for air at room temperature. The Navier-Stokes equations with the Boussinesq approximation are given by Equation (8).

$$\frac{\partial U_i}{\partial x_i} = 0 \tag{8a}$$

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j \partial x_j} - g_i \beta \tau \tag{8b}$$

$$\frac{\partial \tau}{\partial t} + U_j \frac{\partial \tau}{\partial x_j} = \alpha \frac{\partial^2 \tau}{\partial x_j \partial x_j} \tag{8c}$$

The relative temperature, τ, is defined as $\tau \equiv T - T_0$. In order to incorporate the change to the body force term generated by the Boussinesq approximation into FFD, the authors of this paper utilized a technique wherein at each time step after the semi-Lagrangian procedure was employed to determine $\mathbf{U}^*$ an iterative procedure was used which first solved Equation (2b), Equation (3), Equation (2c), and Equation (8c) for the intermediate variables $\widetilde{\mathbf{U}}^{**}, \tilde{p}^{n+1}, \widetilde{\mathbf{U}}^{n+1}, \tilde{\tau}^{n+1}$ and then solved Equation (2b), Equation (3), Equation (2c), and Equation (8c) again with the intermediate variables to determine $\mathbf{U}^{**}, p^{n+1}, \mathbf{U}^{n+1}, \tau^{n+1}$. The first iteration of this procedure can be quantitatively described by Equation (9a) - Equation (9d).

$$\frac{\widetilde{U}_i^{**} - U_i^*}{\Delta t} = \nu \frac{\partial^2 \widetilde{U}_i^{**}}{\partial x_j \partial x_j} - g_i \beta \tau^n \tag{9a}$$

$$\frac{\partial^2 \tilde{p}^{n+1}}{\partial x_i \partial x_i} = \frac{\rho}{\Delta t}\frac{\partial \widetilde{U}_i^{**}}{\partial x_i} \tag{9b}$$

$$\frac{\widetilde{U}_i^{n+1} - \widetilde{U}_i^{**}}{\Delta t} = -\frac{1}{\rho}\frac{\partial \tilde{p}^{n+1}}{\partial x_i} \tag{9c}$$

$$\frac{\partial \tilde{\tau}^{n+1}}{\partial t} + \widetilde{U}_j^{n+1} \frac{\partial \tilde{\tau}^{n+1}}{\partial x_j} = \alpha \frac{\partial^2 \tilde{\tau}^{n+1}}{\partial x_j \partial x_j} \tag{9d}$$

The second iteration of this procedure can be quantitatively described by Equation (10a) - Equation (10d).

$$\frac{U_i^{**} - U_i^*}{\Delta t} = \nu \frac{\partial^2 U_i^{**}}{\partial x_j \partial x_j} - g_i \beta \tilde{\tau}^{n+1} \tag{10a}$$

$$\frac{\partial^2 p^{n+1}}{\partial x_i \partial x_i} = \frac{\rho}{\Delta t}\frac{\partial U_i^{**}}{\partial x_i} \tag{10b}$$

$$\frac{U_i^{n+1} - U_i^{**}}{\Delta t} = -\frac{1}{\rho}\frac{\partial p^{n+1}}{\partial x_i} \tag{10c}$$

$$\frac{\partial \tau^{n+1}}{\partial t} + U_j^{n+1} \frac{\partial \tau^{n+1}}{\partial x_j} = \alpha \frac{\partial^2 \tau^{n+1}}{\partial x_j \partial x_j} \tag{10d}$$

As a way of providing a visual summary, Figure 3 displays a flowchart of FFD with the Boussinesq approximation.

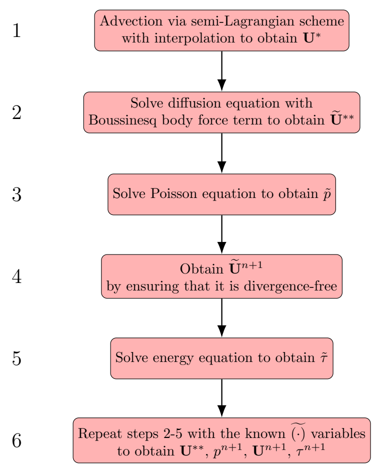

1
Advection via semi-Lagrangian scheme with interpolation to obtain $\mathbf{U}^*$
2
Solve diffusion equation with Boussinesq body force term to obtain $\widetilde{\mathbf{U}}^{**}$
3
Solve Poisson equation to obtain $\tilde{p}$
4
Obtain $\widetilde{\mathbf{U}}^{n+1}$ by ensuring that it is divergence-free
5
Solve energy equation to obtain $\tilde{\tau}$
6
Repeat steps 2-5 with the known $\widetilde{(\cdot)}$ variables to obtain $\mathbf{U}^{**}$, p^{n+1}, $\mathbf{U}^{n+1}$, τ^{n+1}

Figure 3: Flowchart of FFD with the Boussinesq approximation

3. RESULTS

The following section presents and discusses results from the authors' implementation of FFD in OpenFOAM for three different cases: flow in a 2D cavity, flow in a 2D forced convection cavity, and flow in a 3D mixed convection cavity. Table 1 displays the specifications for these cases while Table 2 displays the runtimes of FFD when using cellPoint interpolation as compared to cellPointWallModified interpolation. All subsequent velocity and temperature plots in this section are taken from the final time step of the simulations to which these plots belong. While the results and discussions in this section compare the accuracy of using cellPoint interpolation in FFD with the accuracy of using cellPointWallModified interpolation in FFD, examination of Table 2 demonstrates that the runtimes of these two different versions of FFD are roughly equivalent.

Table 1: Case Specifications

Case	Simulation Time (sec)	Δt (sec)	Re	Mesh
2D Cavity	10	0.01	100	64x64
2D Forced Conv.	400	1	~5000	44x27
3D Mixed Conv.	100	0.05	~2000	44x44x44

Table 2: Case Runtimes (in seconds) when using a 2 GHz processor

Interpolation Routine	2D Cav.	2D Forc. Conv.	3D Mix. Conv.
cellPoint	20.7	13.2	1629.36
cellPointWallModified	20.79	14.17	1692.01

3.1 2D Cavity

The 2D cavity case, pictured in Figure 4a, consists of flow in a two-dimensional square cavity driven by a top plate which moves with a constant velocity while the other boundaries remain stationary. Figure 4b displays the mesh used to run the simulation.

To check the current authors' implementation of FFD, Figure 5 compares velocity profiles generated by FFD using both the cellPoint and cellPointWallModified routines to reference data given in Ghia et al. (1982). Examination of Figure 5 demonstrates that, in this low Reynolds number flow, using both interpolation routines in FFD yield accurate results for the 2D cavity.

Figure 6 plots the number of iterations needed to solve the matrix equations for U_x^{**} and U_y^{**} as the simulation proceeded forward in time. Examination of Figure 6 demonstrates that the number of iterations does indeed decrease with time as the simulation approaches steady-state and the initial guesses to solve for U_x^{**} and U_y^{**} at each time step become closer to their true values.

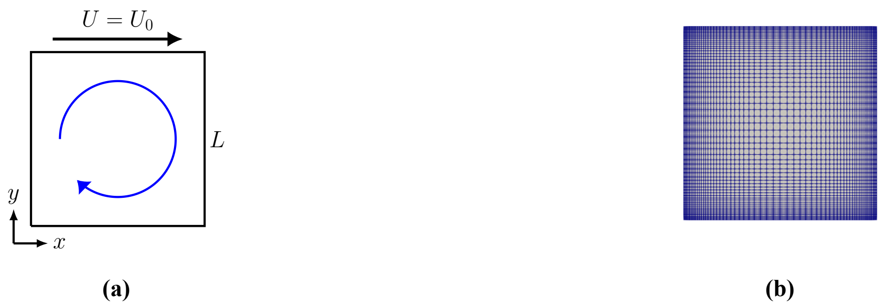


Figure 4: Illustration of the 2D cavity case: (a) Schematic (based on Figure 4-1 of Zuo (2010)) (b) Mesh

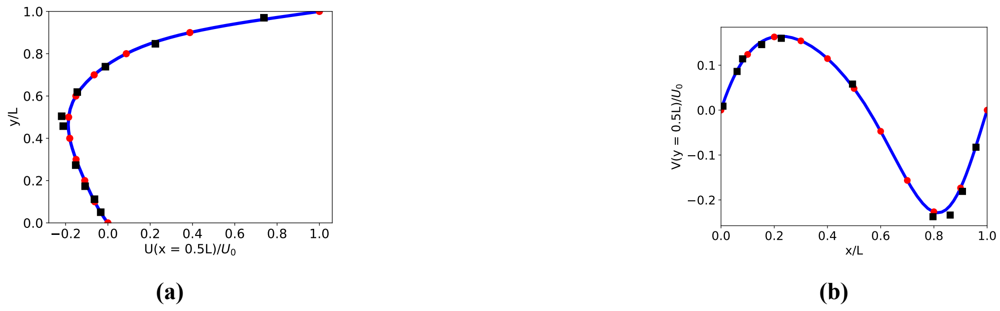


Figure 5: Velocity profiles from the 2D cavity case: —, cellPoint; ●, cellPointWallModified; ■, Reference data from Ghia et al. (1982)

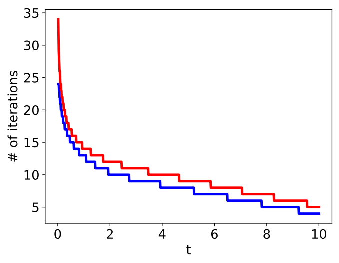
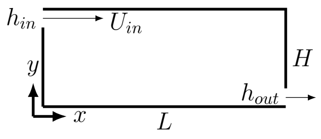


Figure 6: Number of iterations necessary to solve the matrix equations for U_x^{**} and U_y^{**} vs. simulation time (sec): —, U_x^{**}; —, U_y^{**}

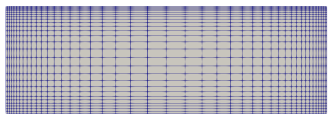
3.2 2D Forced Convection

The 2D forced convection case, pictured in Figure 7a, consists of a 2D, rectangular cavity in which an inlet stream flows in from the upper left corner of the domain and an aperture in the bottom right corner of the domain allows for mass flow out of the cavity. Table 3 provides numerical values for the parameters specified in Figure 7a, while Figure 7b displays the mesh used to run the simulations. The values in Table 3 are the same as in Liu et al. (2016) while the mesh in Figure 7b is based on the mesh in Figure 2b of the same journal article.

Figure 8a displays instantaneous streamlines from the FFD simulation of the forced convection cavity with the cellPoint interpolation routine while Figure 8b displays instantaneous streamlines from the FFD simulation with the cellPoint-WallModified interpolation routine. Figure 8c displays a plot of how the streamwise velocity varies along the height of the forced convection cavity from two-thirds of the way down the length of the cavity while Figure 8d charts how the streamwise velocity varies along the length of the cavity starting in the middle of the inlet slot for the FFD simulations with the cellPoint and cellPointWallModified interpolation routines as compared against reference data from Nielsen et al. (1978). Examination of Figure 8c and Figure 8d demonstrates that the FFD simulation with the cellPoint-WallModified interpolation routine does indeed do a better job of capturing the steep velocity gradient present at the top of the domain in this high Reynolds number flow. In addition, comparison of Figure 8a and Figure 8b demonstrates that the FFD simulation with the cellPointWallModified interpolation routine generates a smoother, more well-formed recirculation bubble in the center of the cavity. Because the cellPointWallModified routine adjusts the treatment of the velocity at the wall, Figure 8 demonstrates that the way in which the conditions at the wall are incorporated into a solver affect not only the behavior of the fluid near the wall, but also the behavior within the interior of the domain as can be seen by the stark change in streamline pattern between Figure 8a and Figure 8b.



(a)

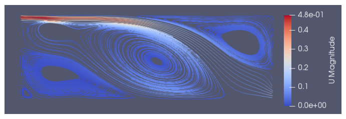


(b)

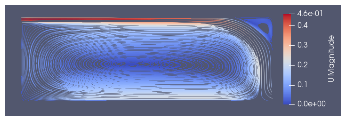
Figure 7: Illustration of the 2D forced convection case: (a) Schematic (based on Figure 4-9 of Zuo (2010)) (b) Mesh

Table 3: Numerical values of the parameters displayed in the schematic of the 2D forced convection case, Figure 7a

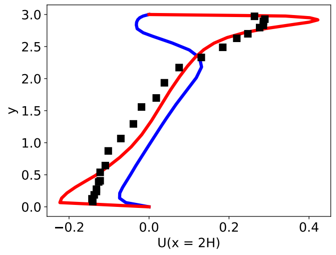
Parameter	L	H	h_{in}	h_{out}	U_{in}
Value	9 m	3 m	0.168 m	0.48 m	0.455 m/s



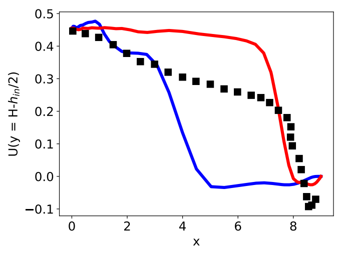
(a)



(b)



(c)



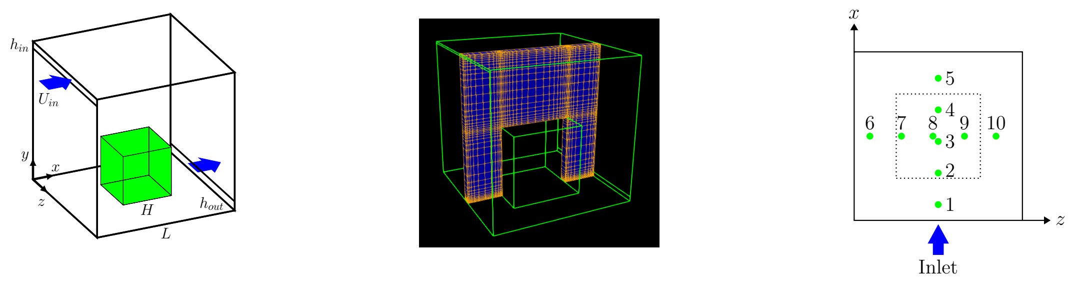
(d)

Figure 8: Results of the FFD simulation of the 2D forced convection case: (a) Streamlines from the simulation with the cellPoint interpolation routine color-coded by the magnitude of velocity (b) Same plot as (a) except cellPoint-WallModified was used (c) Plot of $U(x = 2H)$ vs. y ——, cellPoint; ——, cellPointWallModified; ■, Reference data from Nielsen et al. (1978) (d) Plot of x vs. $U(y = H - h_{in}/2)$ with the same color-coding as in (c)

3.3 3D Mixed Convection

The 3D mixed convection case, pictured in Figure 9a, consists of a cubic cavity in which there exists a solid cubic box at the bottom of the cavity. An inlet stream sends air into the cavity and there exists an outlet slot through which mass can leave the cavity. The walls of the cavity and the box are maintained at different temperatures so that it is necessary to account for thermal effects. Table 4 provides numerical values, equivalent to those in Liu et al. (2016), for the parameters relevant to this case. Because the temperature distribution is non-uniform in the 3D mixed convection cavity, FFD with the Boussinesq approximation as in Section 2.2 was used to simulate this case. Figure 9b displays the mesh used to conduct the simulations. The mesh in Figure 9b is based on Figure 9b of Liu et al. (2016). To analyze the results of the simulation Figure 10 plots the variation of a scaled magnitude of velocity along the y axis at four different locations while Figure 11 plots the variation of a scaled temperature at the same locations as in Figure 10. The points in the x-z plane from which the data from Figure 10 and Figure 11 are taken are visualized in Figure 9c.

Examination of Figure 10 demonstrates that the FFD simulation with the cellPointWallModified routine not only generates an inlet stream with more momentum than its counterpart simulation with the cellPoint interpolation routine, behavior which is in line with the other higher Reynolds number flow tested in this paper (2D forced convection), but also possesses greater momentum in the interior of the domain, results which are closer to the reference data. Examination of Figure 11 demonstrates that the FFD simulation with the cellPointWallModified routine does a better job of capturing the temperature in the interior of the domain, but also is more effective at capturing the trend of the temperature distribution near the inlet (Figure 11a).



(a) **(b)** **(c)**

Figure 9: Illustration of the 3D mixed convection case based on Figure 9 of Liu et al. (2016): (a) Schematic (b) Mesh (c) View from above of the x-z locations from which the data in Figure 10 and Figure 11 are taken

Table 4: Numerical values of the parameters displayed in the schematic of the 3D mixed convection case, Figure 9a

Parameter	L	H	h_{in}	h_{out}	U_{in}
Value	2.44 m	1.22 m	0.03 m	0.08 m	0.455 m/s
Parameter	T_{in}	T_{box}	$T_{\text{cavity ceiling}}$	$T_{\text{cavity walls}}$	$T_{\text{cavity floor}}$
Value	22.2 °C	36.7 °C	25.8 °C	27.4 °C	26.9 °C

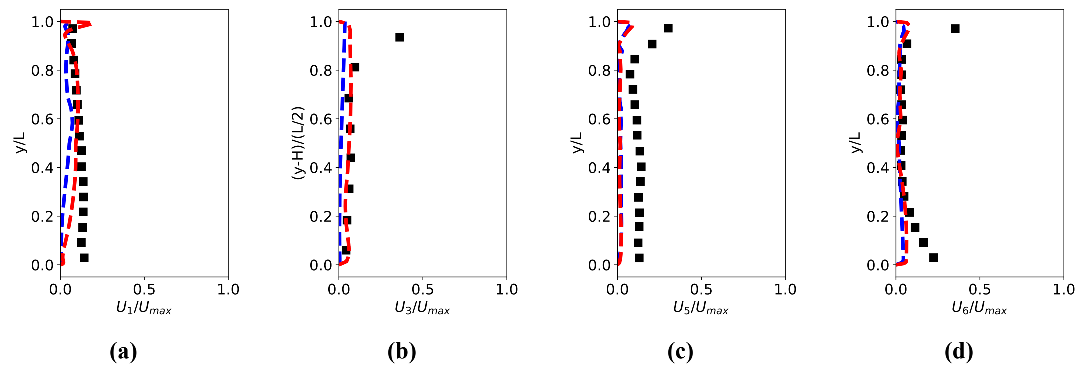


Figure 10: Plots of $|\mathbf{U}|/U_{\max}$ vs. y/L at locations (specified by Figure 9c): (a) 1 (b) 3 (c) 5 (d) 6; $U_{\max}$ = 1.5 m/s; - - -, cellPoint; - - -, cellPointWallModified; ■, Reference data from Wang and Chen (2009)

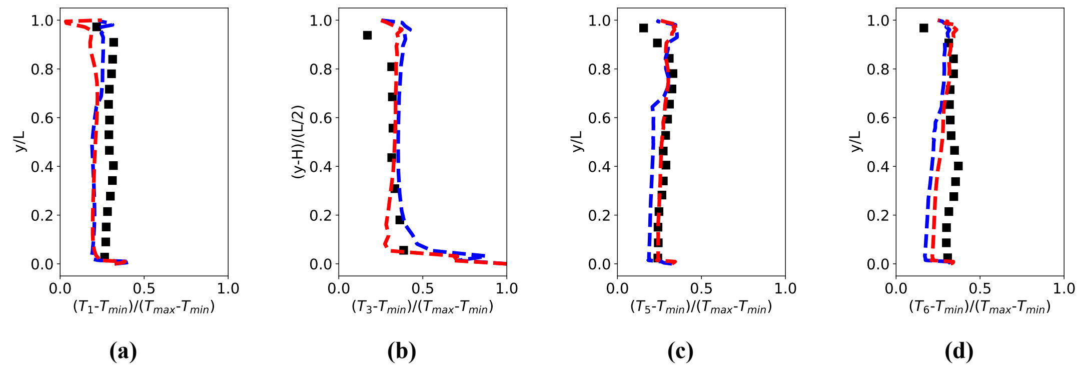


Figure 11: Plots of $(T - T_{\min})/(T_{\max} - T_{\min})$ vs. y/L at locations (specified by Figure 9c): (a) 1 (b) 3 (c) 5 (d) 6; $T_{\min}$ = 22.2 °C, $T_{\max}$ = 36.7 °C; - - -, cellPoint; - - -, cellPointWallModified; ■, Reference data from Wang and Chen (2009)

4. CONCLUSIONS

In this paper, FFD was implemented in OpenFOAM with two different interpolation schemes, cellPoint and cellPointWallModified, and applied to three different test cases. While the timings of FFD with cellPoint interpolation and cellPointWallModified interpolation were roughly equivalent, FFD with cellPointWallModified interpolation produced more accurate simulations in higher Reynolds number flows by adjusting the nature of the interpolation scheme near the boundaries of the domain. Given this information, it appears that the version of FFD with the cellPointWallModified interpolation scheme is superior to the version of FFD with the cellPoint interpolation scheme.

NOMENCLATURE

α thermal diffusivity ($\mathrm{m^2\,s^{-1}}$)
β coefficient of thermal expansion ($\mathrm{K^{-1}}$)
$\mathbf{B}$ body forces (N)
$d(\cdot,\cdot)$ distance (m)
Δ sum of reciprocal distances ($\mathrm{m^{-1}}$)
f generic velocity component ($\mathrm{m\,s^{-1}}$)
f_D velocity component at data point ($\mathrm{m\,s^{-1}}$)
f_I velocity component at interpolation point ($\mathrm{m\,s^{-1}}$)
$\mathbf{g}$ gravitational acceleration ($\mathrm{m\,s^{-2}}$)
ρ density ($\mathrm{kg\,m^{-3}}$)
p pressure (Pa)
τ relative temperature (K)
T temperature (K)
T_0 reference temperature (K)
t time (s)
Δt time step (s)
$\mathbf{U}$ velocity ($\mathrm{m\,s^{-1}}$)
ν kinematic viscosity ($\mathrm{m^2\,s^{-1}}$)
$\mathbf{x}$ spatial coordinates (m)
$\mathbf{x}_a$ arrival point (m)
$\mathbf{x}_d$ departure point (m)
$\mathbf{x}_D$ data point (m)
$\mathbf{x}_I$ interpolation point (m)
Superscript
n current time step
$n+1$ succeeding time step
$*$, $**$ intermediate time variables
~ intermediate iteration variables

REFERENCES

Ghia, U., Ghia, K., and Shin, C. (1982). High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of computational physics*, 48(3):387–411.

Gray, D. and Giorgini, A. (1976). The validity of the boussinesq approximation for liquids and gases. *International Journal of Heat and Mass Transfer*, 19(5):545–551.

Kundu, P., Cohen, I., and Dowling, D. (2012). Chapter 12-turbulence. *Fluid Mechanics (Fifth Edition), Academic Press, Boston*, pages 541–620.

Liu, W., Jin, M., Chen, C., You, R., and Chen, Q. (2016). Implementation of a fast fluid dynamics model in openfoam for simulating indoor airflow. *Numerical Heat Transfer, Part A: Applications*, 69(7):748–762.

Nielsen, P., Restivo, A., and Whitelaw, J. (1978). The velocity characteristics of ventilated rooms. *Journal of Fluids Engineering*, 100:291–298.

OpenFOAM-7 Source Code (2019). cellPointWallModified Header File. https://github.com/OpenFOAM/OpenFOAM-7/blob/master/src/finiteVolume/interpolation/interpolation/interpolationCellPointWallModified/interpolationCellPointWallModified.H.

OpenFOAM Programmer's Guide (2015). Section 2.3.2 Defining a geometricField in OpenFOAM. http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf.

OpenFOAM User Guide (2019). Section 4.6.1.1 Solution tolerances. http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf.

OpenFOAMWiki (2010). OpenFOAM guide/Interpolation (by cell). https://openfoamwiki.net/index.php/OpenFOAM_guide/Interpolation_(by_cell)#interpolationCellPoint.

Qiao, H., Nabi, S., and Laughman, C. (2019). Performance evaluation of hvac systems via coupled simulation between modelica and openfoam.

Wang, M. and Chen, Q. (2009). Assessment of various turbulence models for transitional flows in an enclosed environment (rp-1271). *Hvac&r Research*, 15(6):1099–1119.

Zuo, W. (2010). *Advanced simulations of air distributions in buildings*. Purdue University.