symmetry

MDPI

Article

A Special Multigrid Strategy on Non-Uniform Grids for Solving 3D Convection–Diffusion Problems with Boundary/Interior Layers

Tianlong Ma [1], Lin Zhang [1], Fujun Cao [2] and Yongbin Ge [1,*]

[1] Institute of Applied Mathematics and Mechanics, Ningxia University, Yinchuan 750021, China; matianlong315@126.com (T.M.); zhanglin15@163.com (L.Z.)
[2] School of Science, Inner Mongolia University of Science & Technology, Baotou 014010, China; caofujun@imust.edu.cn
* Correspondence: gyb@nxu.edu.cn

Abstract: Boundary or interior layer problems of high-dimensional convection–diffusion equations have distinct asymmetry. Consequently, computational grid distributions and linear algebraic systems arising from finite difference schemes for them are also asymmetric. Numerical solutions for these kinds of problems are more complicated than those symmetric problems. In this paper, we extended our previous work on the partial semi-coarsening multigrid method combined with the high-order compact (HOC) difference scheme for solving the two-dimensional (2D) convection–diffusion problems on non-uniform grids to the three-dimensional (3D) cases. The main merit of the present method is that the multigrid method on non-uniform grids can be performed with a different number of grids in different coordinate axes, which is more efficient than the multigrid method on non-uniform grids with the same number of grids in different coordinate axes. Numerical experiments are carried out to validate the accuracy and efficiency of the present method. It is shown that, without losing the high precision, the present method is very effective to reduce computing cost by cutting down the number of grids in the direction(s) which does/do not contain boundary or interior layer(s).

Keywords: 3D convection–diffusion equation; HOC scheme; non-uniform grids; multigrid method; partial semi-coarsening

check for updates

Citation: Ma, T.; Zhang, L.; Cao, F.; Ge Y. A Special Multigrid Strategy on Non-Uniform Grids for Solving 3D Convection–Diffusion Problems with Boundary/Interior Layers. *Symmetry* **2021**, *13*, 1123. https://doi.org/10.3390/sym13071123

Academic Editors: Liang-Jian Deng, Xian-Ming Gu and Tai-Xiang Jiang

Received: 29 May 2021
Accepted: 21 June 2021
Published: 24 June 2021

1. Introduction

In this paper, we consider the three-dimensional (3D) steady convection–diffusion equation in the form of

$$-\left(\frac{\partial^2\Phi}{\partial x^2}+\frac{\partial^2\Phi}{\partial y^2}+\frac{\partial^2\Phi}{\partial z^2}\right)+p(x,y,z)\frac{\partial\Phi}{\partial x}+q(x,y,z)\frac{\partial\Phi}{\partial y}+r(x,y,z)\frac{\partial\Phi}{\partial z}=f(x,y,z), \quad (1)$$

where the meaning of each symbol in Equation (1) and in this paper is listed in the following nomenclature.

Equation (1) can be regarded as the model equation of many transport processes of fluid flows and heat transfer [1,2]. It can describe the convection and diffusion of various physical quantities, such as concentrations, heat, momentum, etc. Numerical simulation of fluid dynamics problems is computationally intensive, but it may still be a huge challenge for 3D problems even on the most advanced computers limited to the computing power and storage capacity. Higher precision computational methods combined with faster iterative approaches has seemed to be promising to get satisfying precision solutions with acceptable CPU time during the past three decades [3–7].

High-order compact (HOC) difference methods have been developed and extensively used to solve various convection–diffusion problems [3–5,8–15]. The attractive merits

of HOC schemes include high accuracy, good stability and easy treatment of boundary conditions, etc. However, the majority of the existing HOC schemes are built on uniform grids [3,8,10,12]. For many realistic fluid flow and heat transfer problems which have asymmetry distribution of variation in their domains, such as boundary or interior layer problems [4,5,9,11,13–15], if uniform grids are still used in the whole domain, numerical results would deteriorate unless very fine grids are distributed to obtain acceptable accuracy. However, it is not necessary to do it like this because reasonable non-uniform grid distribution is a good choice, and it would be very effective to handle such problems. To do this, Kalita et al. [11] developed an HOC scheme on rectangular non-uniform grids to solve the 2D convection–diffusion problems. The scheme shows very good scale resolution with various non-uniform grid spacing patterns. Afterwards, this HOC scheme was extended to the 3D case by Ge and Cao [5] and Shanab et al. [15]. At the same time, we notice that the same number of grids in all coordinate axes is used for all the numerical examples they considered [5,11,15]. However, for those problems with local large gradients, the physical quantities change steeply in just one direction (for 2D) or one/two direction(s) (for 3D), while, in the other direction(s), solutions change smoothly, and using the same number of grids in all directions is unnecessary. It would cause a big waste of storage space and computing cost. Thus, one aim of this paper is to design a special grid distribution strategy to fix this defect.

To efficiently solve the linear algebraic system which is resulting from finite difference schemes, the multigrid method is proposed [16]. Up to now, the multigrid method has been employed to solve various elliptic equations discretized by the HOC difference schemes [5–7,9,14,17–24]. However, since most HOC schemes are proposed on the uniform grids, consequently, the multigrid methods are carried out on uniform grids [6,18–21,24]. Ge and Zhang et al. [4,9] employed the fourth-order compact discretization schemes and the multigrid method to solve 2D and 3D convection–diffusion equation with boundary layers on non-uniform grids. Because the grid transformation was used to transform the non-uniform grid to a uniform one, the multigrid method was also implemented on uniform grids in the computational plane. In addition, it is worthy of being pointed out that, in Ref. [22], a multigrid method is developed to solve the system of linear equations arising from the HOC scheme [11] on non-uniform grids for 2D problems, and it was extended to 3D cases [5,7]. It is a pity that the same number of grids in all coordinate axes must be used because the restriction and interpolation operators needed by the multigrid method are constructed just for a full-coarsening grid (see Refs. [5,7] for more details). Actually, it is not necessary to distribute the same number of grids for those boundary or interior layer problems when local big gradient(s) only occur(s) in partial coordinate direction(s).

To overcome the defect, based on the idea of Zhang [19] about the partial semi-coarsening strategy for solving a 2D Poisson equation, Cao et al. [23] developed a special method to solve the 2D boundary or interior layer problems of the convection–diffusion equation. The main merit of the grid distribution strategy is that it permits to set a different number of grids for x- and y-coordinate axes, respectively. Under such circumstances, computing cost is greatly reduced by setting a small number of grids in non-dominant direction(s), but the computed accuracy is not lost. In this paper, we tend to generalize the method in [23] to 3D cases and develop a special multigrid method on non-uniform grids to solve the 3D convection–diffusion boundary or interior layer problems (1). The outline of this paper is arranged as follows. In Section 2, the HOC scheme that is proposed by Ge and Cao [5] will be extended for solving the 3D convection–diffusion Equation (1) on non-uniform grids. In Section 3, we develop a special multigrid strategy including partial semi-coarsening procedure, specified restriction and interpolation operators. After that, numerical examples are employed to illustrate the accuracy and efficiency of the present method in Section 4. Section 5 presents the conclusions of this paper.

2. HOC Scheme on Non-Uniform Grids

We consider a cubic domain $\Omega = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$, and divide the interval $[a_1, a_2]$, $[b_1, b_2]$ and $[c_1, c_2]$ into sub-intervals by the points $a_1 = x_0, x_1, x_2, \ldots, x_{N_x-1}, x_{N_x} = a_2$, $b_1 = y_0, y_1, y_2, \ldots, y_{N_y-1}, y_{N_y} = b_2$ and $c_1 = z_0, z_1, z_2, \ldots, z_{N_z-1}, z_{N_z} = c_2$. In the x-axis, the backward and forward step lengths can be respectively given by $x_b = x_i - x_{i-1}$, $x_f = x_{i+1} - x_i$, $1 \le i \le N_x - 1$. In addition, similarly, on the y- and z-axes, $y_b = y_j - y_{j-1}$, $y_f = y_{j+1} - y_j$, $1 \le j \le N_y - 1$, $z_b = z_k - z_{k-1}$, $z_f = z_{k+1} - z_k$, $1 \le k \le N_z - 1$. If and only if $x_f = x_b$, $y_f = y_b$ and $z_f = z_b$, the grid turns to be uniform. Figure 1 shows a non-uniform cubic grid stencil in the subdomain.

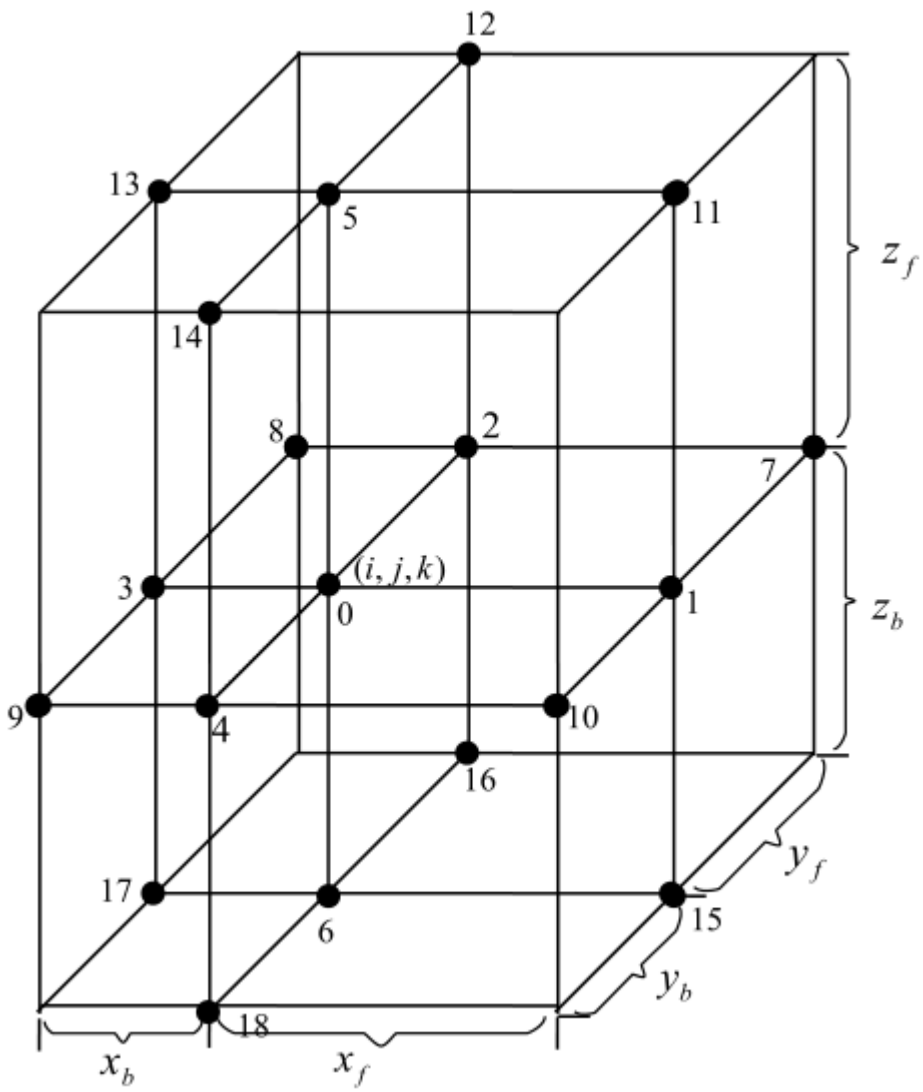


Figure 1. The stencil of 3D non-uniform grids.

The discretization value of $\Phi(x, y, z)$ at a reference point (x, y, z) denoted by Φ_0 and the other 18 neighboring grid points are defined analogously as Figure 1. The HOC difference scheme on non-uniform grids for the 3D convection–diffusion Equation (1) developed by Ge and Cao [5] is given as

$$\begin{aligned}(&-A_{11}\delta_x^2 - A_{22}\delta_y^2 - A_{33}\delta_z^2 + A_1\delta_x + A_2\delta_y + A_3\delta_z + A_{12}\delta_x\delta_y + A_{13}\delta_x\delta_z + A_{23}\delta_y\delta_z \\ &+ A_{112}\delta_x^2\delta_y + A_{113}\delta_x^2\delta_z + A_{223}\delta_y^2\delta_z + A_{122}\delta_x\delta_y^2 + A_{133}\delta_x\delta_z^2 + A_{233}\delta_y\delta_z^2 \\ &+ A_{1122}\delta_x^2\delta_y^2 + A_{1133}\delta_x^2\delta_z^2 + A_{2233}\delta_y^2\delta_z^2)\Phi_0 = F_0, \end{aligned} \tag{2}$$

where the coefficients A_{11}, A_{22}, A_{33}, A_1, A_2, A_3, A_{12}, A_{13}, A_{23}, A_{112}, A_{113}, A_{223}, A_{122}, A_{133}, A_{233}, A_{1122}, A_{1133}, A_{2233} and F_0 are given by following

$$A_{11} = 1 - (H_1 + H_2 p_0 + 2H_2\delta_x)p_0, \quad A_{22} = 1 - (K_1 + K_2 q_0 + 2K_2\delta_y)q_0, \tag{3}$$

$$A_{33} = 1 - (L_1 + L_2 r_0 + 2L_2\delta_z)r_0, \tag{4}$$

$$\begin{aligned} A_1 = {} & [1 + (H_1 + H_2 p_0)\delta_x + (K_1 + K_2 q_0)_1\delta_y + (L_1 + L_2 r_0)\delta_z \\ & + H_2\delta_x^2 + K_2\delta_y^2 + L_2\delta_z^2]p_0, \end{aligned} \tag{5}$$

$$\begin{aligned} A_2 = {} & [1 + (H_1 + H_2 p_0)\delta_x + (K_1 + K_2 q_0)_1\delta_y + (L_1 + L_2 r_0)\delta_z \\ & + H_2\delta_x^2 + K_2\delta_y^2 + L_2\delta_z^2]q_0, \end{aligned} \tag{6}$$

$$\begin{aligned} A_3 = {} & [1 + (H_1 + H_2 p_0)\delta_x + (K_1 + K_2 q_0)_1\delta_y + (L_1 + L_2 r_0)\delta_z \\ & + H_2\delta_x^2 + K_2\delta_y^2 + L_2\delta_z^2]r_0, \end{aligned} \tag{7}$$

$$A_{12} = (H_1 + H_2 p_0 + 2H_2\delta_x)q_0 + (K_1 + K_2 q_0 + 2K_2\delta_y)p_0, \tag{8}$$

$$A_{13} = (H_1 + H_2 p_0 + 2H_2\delta_x)r_0 + (L_1 + L_2 r_0 + 2L_2\delta_z)p_0, \tag{9}$$

$$A_{23} = (K_1 + K_2 q_0 + 2K_2\delta_y)r_0 + (L_1 + L_2 r_0 + 2L_2\delta_z)q_0, \tag{10}$$

$$A_{112} = -K_1 + (H_2 - K_2)q_0, \quad A_{113} = -L_1 + (H_2 - L_2)r_0, \tag{11}$$

$$A_{223} = -L_1 + (K_2 - L_2)r_0, \quad A_{122} = -H_1 + (K_2 - H_2)p_0, \tag{12}$$

$$A_{133} = -H_1 + (L_2 - H_2)p_0, \quad A_{233} = -K_1 + (L_2 - K_2)q_0, \tag{13}$$

$$A_{1122} = -H_2 - K_2, \quad A_{1133} = -H_2 - L_2, \quad A_{2233} = -K_2 - L_2, \tag{14}$$

$$\begin{aligned} F_0 = {} & [1 + (H_1 + H_2 p_0)\delta_x + (K_1 + K_2 q_0)\delta_y + (L_1 + L_2 r_0)\delta_z \\ & + H_2\delta_x^2 + K_2\delta_y^2 + L_2\delta_z^2]f_0, \end{aligned} \tag{15}$$

and

$$H_1 = \frac{1}{3}(x_f - x_b) - \frac{1}{6}x_f x_b p_0, \quad H_2 = \frac{1}{12}(x_f^2 + x_b^2 - x_f x_b) - \frac{1}{24}x_f x_b(x_f - x_b)p_0, \tag{16}$$

$$K_1 = \frac{1}{3}(y_f - y_b) - \frac{1}{6}y_f y_b q_0, \quad K_2 = \frac{1}{12}(y_f^2 + y_b^2 - y_f y_b) - \frac{1}{24}y_f y_b(y_f - y_b)q_0, \tag{17}$$

$$L_1 = \frac{1}{3}(z_f - z_b) - \frac{1}{6}z_f z_b r_0, \quad L_2 = \frac{1}{12}(z_f^2 + z_b^2 - z_f z_b) - \frac{1}{24}z_f z_b(z_f - z_b)r_0. \tag{18}$$

The finite difference operators appearing in Figure 2 are given in Appendix A. It notes that the order of the truncation error of the scheme is fourth on uniform grids if $x_f = x_b$, $y_f = y_b$ and $z_f = z_b$ and at least third on non-uniform grids if $x_f \neq x_b$ or $y_f \neq y_b$ or $z_f \neq z_b$. Shanab et al. [15] also got this scheme with a similar derivation process (more details, see Refs. [5,15]).

$128 \times 32 \times 8$
$\Downarrow$ ①
$64 \times 32 \times 8$
$\Downarrow$ ②
$32 \times 32 \times 8$
$\Downarrow$ ③
$16 \times 16 \times 8$
$\Downarrow$ ④
$8 \times 8 \times 8$
$\Downarrow$ ⑤
$4 \times 4 \times 4$
$\Downarrow$ ⑥
$2 \times 2 \times 2$

Figure 2. Process of grid partial semi-coarsening.

3. Partial Semi-Coarsening Multigrid Method

The multigrid method is a very efficient iterative method for solving the large systems of algebraic equations arising from the discretizations of elliptic boundary value problems [25]. The main logic of the multigrid method is based on the recognition that classical relaxation methods can strongly dampen the oscillatory error components, but converge slowly for smooth error components. However, oscillation or smoothness is comparative, i.e., the smooth error components on fine grids become oscillatory when they are transferred to coarse grids. Therefore, the multigrid method splits the errors into high frequency components and low frequency components and solves them in different subspaces. Firstly, we perform a few relaxation sweeps on the fine grid and then transfer the residual of the fine grid to the next coarse grid and smooth it again on the coarse grid. This process is repeated successively until the coarsest grid. Then, the corrections are interpolated back to finer grids successively and relaxation sweeps are implemented again until the process reaches the finest grid. This procedure is called a multigrid V-cycle. Restriction operator, interpolation operator and relaxation operator are three essential components of the multigrid method. We will discuss them respectively in the following parts.

Mulder [26] firstly proposed a semi-coarsening multigrid method to solve the convection problems. After that, Liu and Liu [27] employed it to simulate the whole process of flow transition in 3D flat plate boundary layers. In 2002, Zhang [19] developed a partial semi-coarsening multigrid method for solving the 2D Poisson equation. Partial semi-coarsening means that, after semi-coarsening, on a certain coarse grid, the number of grids in both directions must be same; then, the standard full coarsening process is carried out in both directions until the coarsest grid is reached. Numerical results indicate that the computational cost of the partial semi-coarsening strategy is correspondingly decreased greatly with fewer grids in the non-dominant axis than that in the dominant axis, but computed accuracy does not lose compared to the full grid coarsening process with the same number of grids in both directions. Ge [20] extended such a method to the 3D case. However, the grid distribution in [19,20] is uniform although unequal mesh-size is employed for different coordinate directions.

In this paper, we will develop a partial semi-coarsening multigrid method with non-uniform grids to solve the 3D convection–diffusion problems. The merit of computational methods with non-uniform grids is that it is more flexible to treat local large gradient problems, such as boundary layers or local singularities, etc. It is necessary to show the 3D grid partial semi-coarsening process here. For example, we assume that the computational domain is a unit cube, and the number of grids on the x-, y- and z-axes are N_x, N_y, and N_z, respectively. Then, if we simplify to write $N_x = 128$, $N_y = 32$, $N_z = 8$ as $128 \times 32 \times 8$, Figure 2 gives the process of partial semi-coarsening for 3D non-uniform grids, in which the steps ①–④ are semi-coarsening and steps ⑤ and ⑥ are full-coarsening. The whole process is called partial semi-coarsening. The key technique for implementing the partial semi-coarsening multigrid method is to construct new restriction and interpolation operators for different grid coarsening processes.

3.1. Restriction Operator

As a matter of convenience but without loss of generality, we suppose that the x- and y-axes are dominant, where the boundary or interior layers exist, but the z-axis is not dominant. Under these circumstances, it is reasonable to put more grid points in x- and y-axes than on the z-axis. Because the numbers of girds in x-, y-, and z-axes may be different, the restriction operators must be individually built. Firstly, we use $r_{i,j,k}$ to represent the residual on the fine grid point (i, j, k), and $\bar{r}_{\bar{i},\bar{j},\bar{k}}$ to represent the corresponding residual on the corresponding coarse grid point $(\bar{i}, \bar{j}, \bar{k})$. It notes that $i = 2\bar{i}$, $j = 2\bar{j}$ and $k = 2\bar{k}$. For the restriction operator of the residual, we design it as follows:

(i) If $N_x > N_y > N_z$, i.e., the number of grids is the biggest in the x-direction, so we only conduct grid coarsening in the x-direction. A one-way weighting average strategy is used. In Figure 3, we define $L_1 = x_f/2$, $L_0 = (x_b + x_f)/2$, $L_3 = x_b/2$ and $L_x = x_b + x_f$—in

which x_f and x_b are forward and backward step lengths of the fine grids. Two big black points denote the grid points on the fine grid level, while a small black point represents the grid point on the coarse grid level. Its residual is evaluated by itself and its two neighboring grid points on the fine grid level. According to the one-way average strategy introduced by Cao et al. [23], the restriction operator for the residual from fine grid to coarse grid is written out as follows:

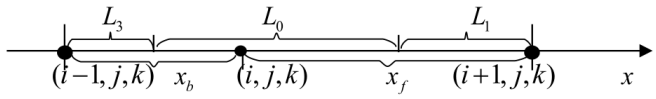


Figure 3. Length in the x-direction for the restriction operator.

$$\bar{r}_{\bar{i},j,k} = \frac{1}{L_x}(L_1 r_{i-1,j,k} + L_0 r_{i,j,k} + L_3 r_{i+1,j,k}). \tag{19}$$

(ii) If $N_x = N_y > N_z$, under this circumstance, there exist two dominant directions, i.e., x- and y-axes. The two-way average strategy introduced by Ge and Cao [22] is employed to evaluate the residual on the coarse grid level. The basic principle of it is the area law described by Liu [28]. Figure 4 gives the areas $S_i (i = 0, 1, \ldots, 8)$ for the non-uniform subdomain. The corresponding full weighting restriction operator can be explicitly given out as:

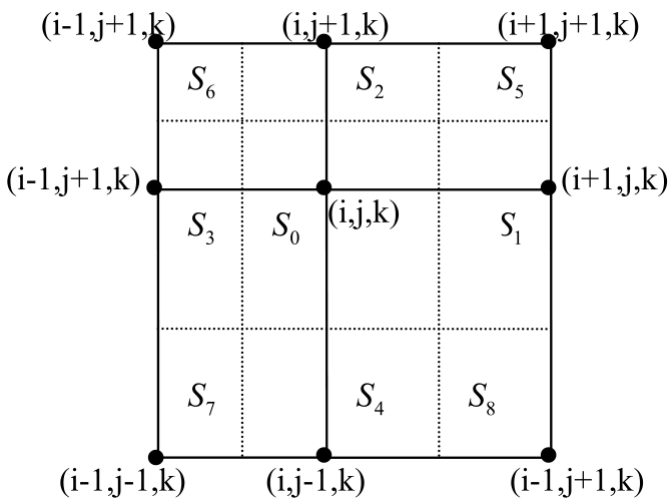


Figure 4. Areas in both the x- and y-directions for the restriction operator.

$$\begin{aligned}\bar{r}_{\bar{i},\bar{j},k} = \frac{1}{S}(&S_0 r_{i,j,k} + S_1 r_{i-1,j,k} + S_2 r_{i,j-1,k} + S_3 r_{i+1,j,k} + S_4 r_{i,j+1,k} \\ &+ S_5 r_{i-1,j-1,k} + S_6 r_{i+1,j-1,k} + S_7 r_{i+1,j+1,k} + S_8 r_{i-1,j+1,k}).\end{aligned} \tag{20}$$

where the S_i $(i = 1, \ldots, 8)$ are defined as follows:

$$\begin{aligned}&S = (x_f + x_b) \times (y_f + y_b), \quad S_0 = \tfrac{1}{4}(x_f + x_b) \times (y_f + y_b), \quad S_1 = \tfrac{1}{4}x_f \times (y_f + y_b), \\ &S_2 = \tfrac{1}{4}y_f \times (x_f + x_b), \quad S_3 = \tfrac{1}{4}x_b \times (y_f + y_b), \quad S_4 = \tfrac{1}{4}y_b \times (x_f + x_b), \\ &S_5 = \tfrac{1}{4}x_f \times y_f, \quad S_6 = \tfrac{1}{4}x_b \times y_f, \quad S_7 = \tfrac{1}{4}x_b \times y_b, \quad S_8 = \tfrac{1}{4}x_f \times y_b.\end{aligned}$$

(iii) If $N_x = N_y = N_z$, for each grid point on the coarse grid level, Ref. [7] proposed a full-weighting restriction operator on a non-uniform grid based on the volume law [28]. Under this circumstance, we can use the full weighting restriction operator directly (see Appendix B).

3.2. Interpolation Operator

(i) If $N_x > N_y > N_z$, under this condition, interpolation is only needed to be conducted on the x-axis. Figure 5 shows the coarse grid points (big black points), the fine grid point (small and big black points), and the step length of the coarse grid stencil L_x. According to the average strategy introduced by Cao et al. [23], the interpolation operator can be explicitly given out as:

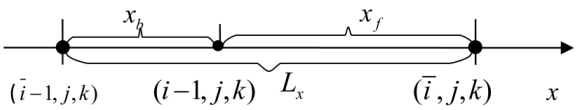


Figure 5. Length in the x-direction for the interpolation operator.

$$r_{i,j,k} = \bar{r}_{\bar{i},j,k}, \quad r_{i-1,j,k} = \frac{1}{L_x}(x_f \bar{r}_{\bar{i}-1,j,k} + x_b \bar{r}_{\bar{i},j,k}). \tag{21}$$

(ii) If $N_x = N_y > N_z$, under such circumstance, interpolation should be performed on both the x- and y-axes. Figure 6 shows the areas $S_i (i = 1, 2, 3, 4)$ used by the non-uniform interpolation operator. In addition, big black points represent the grid points on the coarse grid level and small black points the grid points on the fine grid level. The interpolation operator can be explicitly given out as:

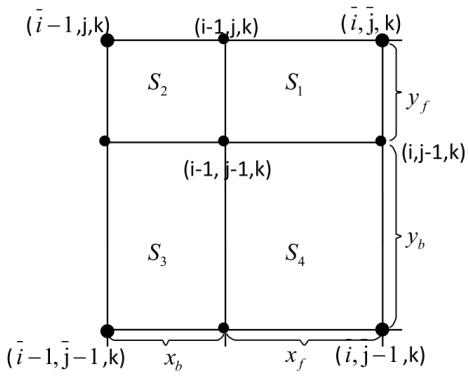


Figure 6. Areas in both the x- and y-directions for the interpolation operator.

$$r_{i,j,k} = \bar{r}_{\bar{i},\bar{j},k}, \quad r_{i-1,j,k} = \frac{1}{x_f + x_b}(x_f \bar{r}_{\bar{i}-1,\bar{j},k} + x_b \bar{r}_{\bar{i},\bar{j},k}), \tag{22}$$

$$r_{i,j-1,k} = \frac{1}{y_f + y_b}(y_f \bar{r}_{\bar{i},\bar{j}-1,k} + y_b \bar{r}_{\bar{i},\bar{j},k}), \tag{23}$$

$$r_{i-1,j-1,k} = \frac{1}{S}(S_1 \bar{r}_{\bar{i}-1,\bar{j}-1,k} + S_2 \bar{r}_{\bar{i},\bar{j}-1,k} + S_3 \bar{r}_{\bar{i},\bar{j},k} + S_4 \bar{r}_{\bar{i}-1,\bar{j},k}), \tag{24}$$

where

$$S = (x_f + x_b) \times (y_f + y_b), \;\; S_1 = x_f \times y_f, \;\; S_2 = x_b \times y_f, \;\; S_3 = x_b \times y_b, \;\; S_4 = x_f \times y_b.$$

(iii) If $N_x = N_y = N_z$, for this case, we directly use the tri-linear interpolation operator on a non-uniform grid based on the volume law proposed in [7] (see Appendix C).

3.3. Relaxation Operator

The role of relaxation operators (such as Jacobi, Gauss–Seidel iteration, etc.) for the multigrid method is not to remove the errors, but to dump the high frequency components of the errors on the current grid while leaving the low frequency components to be removed by the coarser grids. In [19,20], the line Gauss–Seidel relaxation (for 2D) and plane Gauss–Seidel relaxation (for 3D) were used to remove the high frequency error

components in the dominant direction(s) and prove to be very efficient and robust. In [7], the point Gauss–Seidel relaxation, the red-black Gauss–Seidel relaxation, the four-color Gauss–Seidel relaxation, and the alternating direction plane Gauss–Seidel relaxation were used as smoothers for the multigrid method, and their smoothing effect was compared with numerical experiments. In this paper, we use the line Gauss–Seidel relaxations as smoothers if the boundary or interior layers just exist in one direction as done in [19]; the plane Gauss–Seidel relaxations is used if the boundary or interior layers exist in two directions, as done in [20]. Once the boundary or interior layers exist in all three directions, the alternating plane Gauss–Seidel relaxation is applied as a smoother as done in [7].

4. Numerical Experiments

The following four convection–diffusion problems with boundary layers or interior layers are computed to validate the high accuracy and high efficiency of the present method. The definition domain of all problems is set to be $\Omega = [0,1] \times [0,1] \times [0,1]$, and the Dirichlet boundary conditions are considered. All computations are coded by the Fortran 77 language, and the programs are stopped when the residuals in the L_2 norm on the finest grids are cut down by 10^{10}. Maximum absolute error (Error) on the finest grid, the CPU time in seconds, numbers of multigrid $V(\nu_1, \nu_2)$ cycles (Num), and convergence order are reported. The convergence order is computed by:

$$Order = \frac{\log(Error1/Error2)}{\log(N_2/N_1)}, \tag{25}$$

where $Error1$ and $Error2$ are the maximum absolute errors estimated at two different grids with the numbers of grids N_1 and N_2, respectively.

4.1. Problem 1

$$-(\Phi_{xx} + \Phi_{yy} + \Phi_{zz}) + p(x,y,z)\Phi_x + q(x,y,z)\Phi_y + r(x,y,z)\Phi_z = f(x,y,z), \tag{26}$$

where

$$\begin{aligned} p(x,y,z) &= \mathrm{Re}x(x-1)(1-2y)\,(1-2z), \\ q(x,y,z) &= \mathrm{Re}y(y-1)(1-2x)\,(1-2z)\,, \\ r(x,y,z) &= \mathrm{Re}z(z-1)(1-2x)\,\,(1-2y). \end{aligned} \tag{27}$$

The exact solution is

$$\Phi(x,y,z) = e^{-\sigma(x-0.5)^2 - y^2 - z^2}. \tag{28}$$

A local large gradient solution exists at $x = 0.5$ with big σ, so we just refine grids on the x-axis while setting fewer grids on the y- and z-axes by using the grid distribution as follows:

$$x_i = \frac{i}{N_x} + \frac{\lambda_x}{2\pi}\sin(\frac{2\pi i}{N_x}), \qquad y_j = \frac{j}{N_y}, \qquad z_k = \frac{k}{N_z}. \tag{29}$$

where $\lambda_x \in (-1,1)$ is a stretching parameter that controls the distribution of grids on the x-axis. For example, $-1 < \lambda_x < 0$ indicates that more grid points are clustered to the boundary $x = 0$ and $x = 1$, whereas $0 < \lambda_x < 1$ indicates that more grid points are clustered around the middle of the region $x = 0.5$. If $\lambda_x = 0$, the grid turns out to be uniform in the computational region.

We set $Re = 10^2, \sigma = 10^3$ and multigrid $V(2,2)$ cycles. Table 1 shows the computed results of maximum errors, multigrid iteration numbers, CPU time and comparisons with the full-coarsening grids [5]. We find that the computed accuracy does not lose when less grids are used in the two non-dominant directions. For example, when $\lambda_x = 0.6$, the maximum error with $32 \times 16 \times 16$ grids is almost the same as that of $32 \times 32 \times 32$. It is not

strange because the local gradient solution only occurs in the x-direction and not in the y- or z-direction. Thus, we can set fewer grids in them to reduce the computational cost and make the computation more efficient and cost-effective. For instance, on a non-uniform grid, the number of grid $32 \times 16 \times 16$ is just a quarter of that of grid $32 \times 32 \times 32$, and the CPU time with $32 \times 16 \times 16$ grids is also just a quarter of that of $32 \times 32 \times 32$ grids. Under these conditions, we can find that the multigrid of partial semi-coarsening is much more efficient than that of full-coarsening as long as the multigrid iteration numbers and the CPU time are being compared. We note that $64 \times 64 \times 64$, $32 \times 32 \times 32$ and $16 \times 16 \times 16$ grids still used for a partial semi-coarsening grid algorithm is to show that the partial semi-coarsening grid algorithm can reduce to the full-coarsening grid algorithm if $N_x = N_y = N_z$.

Table 2 gives the numerical results at $Re = 10^2, \sigma = 10^4$ with $\lambda_x = 0.75$. We notice that, when σ is bigger, the solution in the interior layer changes more violently. However, the computed results on non-uniform grids are still very accurate. On the other hand, we notice that, when the number of grids in the dominant direction, i.e., the x-axis, is fixed, reducing the number of grid points on the y- and z-axes can still get almost equivalent accurate solutions as with fine grids in all three of the directions. By the comparison of the numerical results with $64 \times 64 \times 64$ full-coarsening grids and $64 \times 16 \times 16$ partial seim-coarsening grids, we can get that, for this case, a coarse grid with only a one-sixteenth grid of the fine grid is enough to achieve nearly the same accuracy as the fine grid in the three directions. As a result, the CPU time of a correspondingly partial semi-coarsening multigrid method is just about one-sixteenth of the cost by the full-coarsening multigrid method.

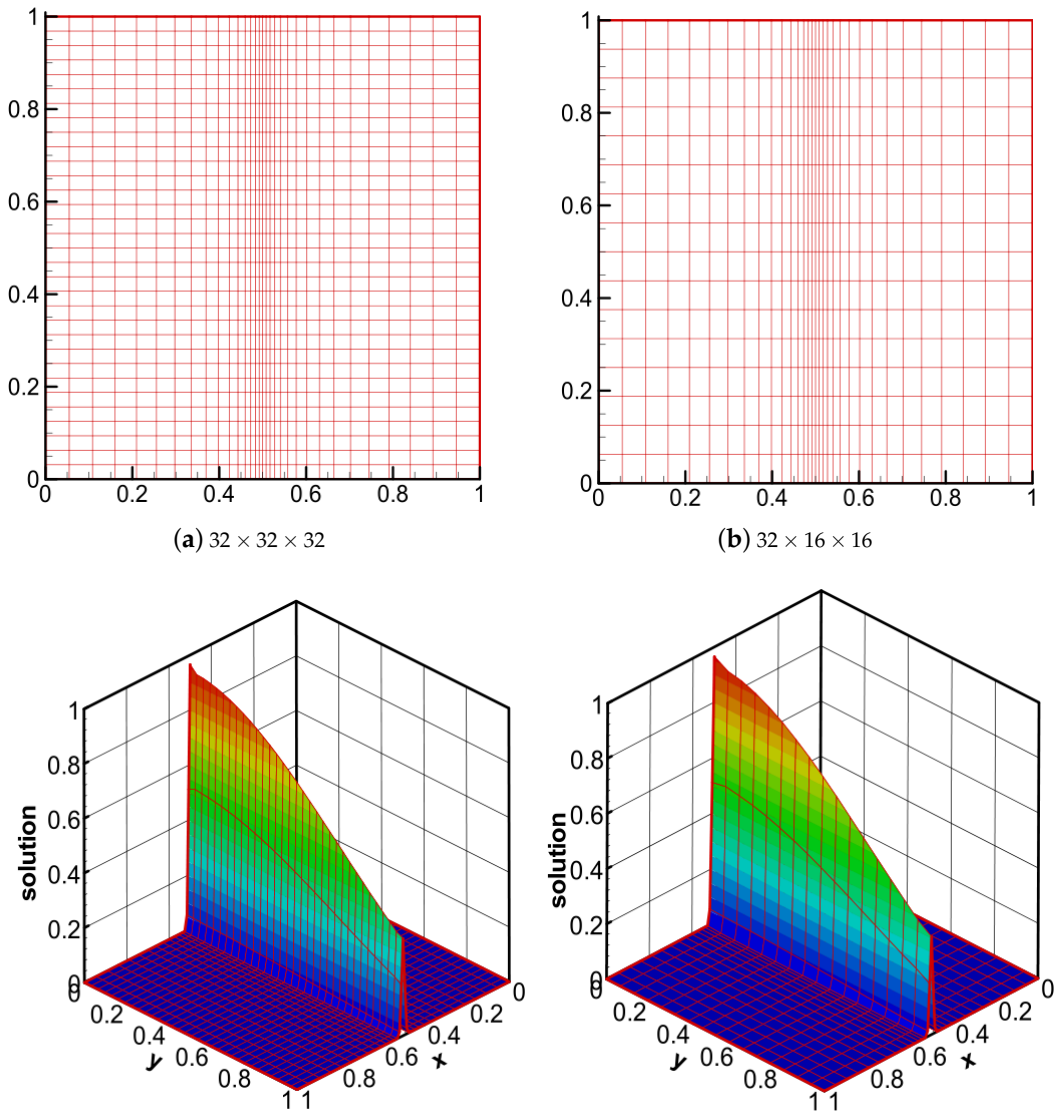
Figure 7a,b show the grid with $32 \times 32 \times 32$ and $32 \times 16 \times 16$, with $\lambda_x = 0.75$ in the xoy plane, respectively. Figure 7c,d show the computed solutions on $32 \times 32 \times 32$ grids and $32 \times 16 \times 16$ grids at the plane $z = 0.375$ for $\sigma = 10^4$. We can see that the solution with $32 \times 16 \times 16$ grids is almost indistinguishable from that with $32 \times 32 \times 32$ grids. It also demonstrates that distributing more grids on non-dominant directions is unnecessary. However, computational efficiency is improved, and computing cost is saved because less grid points are involved.

Table 1. Maximum errors, multigrid iteration numbers, and CPU time for Problem in Section 4.1 ($\sigma = 10^3$, $\lambda_x = 0.6$).

Full-Coarsening Grids [5]				**Partial Semi-Coarsening Grids**			
$N_x \times N_y \times N_z$	**Num**	**CPU**	**Error**	$N_x \times N_y \times N_z$	**Num**	**CPU**	**Error**
$64 \times 64 \times 64$	11	5.023	2.175×10^{-4}	$64 \times 64 \times 64$	11	5.023	2.175×10^{-4}
				$64 \times 32 \times 32$	9	1.482	2.476×10^{-4}
				$64 \times 16 \times 16$	6	0.296	5.240×10^{-4}
				$64 \times 8 \times 8$	6	0.125	1.831×10^{-3}
$32 \times 32 \times 32$	11	0.999	2.732×10^{-3}	$32 \times 32 \times 32$	11	0.999	2.732×10^{-3}
				$32 \times 16 \times 16$	7	0.250	2.784×10^{-3}
				$32 \times 8 \times 8$	5	0.063	4.211×10^{-3}
$16 \times 16 \times 16$	10	0.141	7.539×10^{-2}	$16 \times 16 \times 16$	10	0.141	7.539×10^{-2}
				$16 \times 8 \times 8$	11	0.047	7.606×10^{-2}

Table 2. Maximum errors, multigrid iteration numbers, and CPU time for Problem in Section 4.1 ($\sigma = 10^4$, $\lambda_x = 0.75$).

Full-Coarsening Grids [5]				**Partial Semi-Coarsening Grids**			
$N_x \times N_y \times N_z$	**Num**	**CPU**	**Error**	$N_x \times N_y \times N_z$	**Num**	**CPU**	**Error**
$64 \times 64 \times 64$	12	8.658	1.533×10^{-3}	$64 \times 64 \times 64$	12	8.658	1.533×10^{-3}
				$64 \times 32 \times 32$	9	2.215	1.533×10^{-3}
				$64 \times 16 \times 16$	6	0.499	1.568×10^{-3}
				$64 \times 8 \times 8$	5	0.140	2.434×10^{-3}
$32 \times 32 \times 32$	12	1.076	2.690×10^{-2}	$32 \times 32 \times 32$	12	1.076	2.690×10^{-2}
				$32 \times 16 \times 16$	8	0.234	2.827×10^{-2}
				$32 \times 8 \times 8$	6	0.078	2.950×10^{-2}



(**a**) 32 × 32 × 32 (**b**) 32 × 16 × 16

(**c**) 32 × 32 × 32 (**d**) 32 × 16 × 16

Figure 7. (**a**,**b**) Non-uniform grids with $\lambda_x = 0.75$ in the xoy plane; (**c**,**d**) computed solutions on a non-uniform grid, for $\sigma = 10^4$, $Re = 10^2$, at the plane $z = 0.375$, Problem in Section 4.1.

4.2. Problem 2

$$-\varepsilon(\Phi_{xx} + \Phi_{yy} + \Phi_{zz}) + \frac{1}{1+y}\Phi_y = f(x, y, z). \tag{30}$$

The exact solution is

$$\Phi(x, y, z) = z[e^{y-x} + 2^{-1/\varepsilon}(1+y)^{1+1/\varepsilon}]. \tag{31}$$

For this example, when ε is small, a large gradient solution just occurs in the axis $y = 1$, but not on x- and z-axes, so we use the following grid functions:

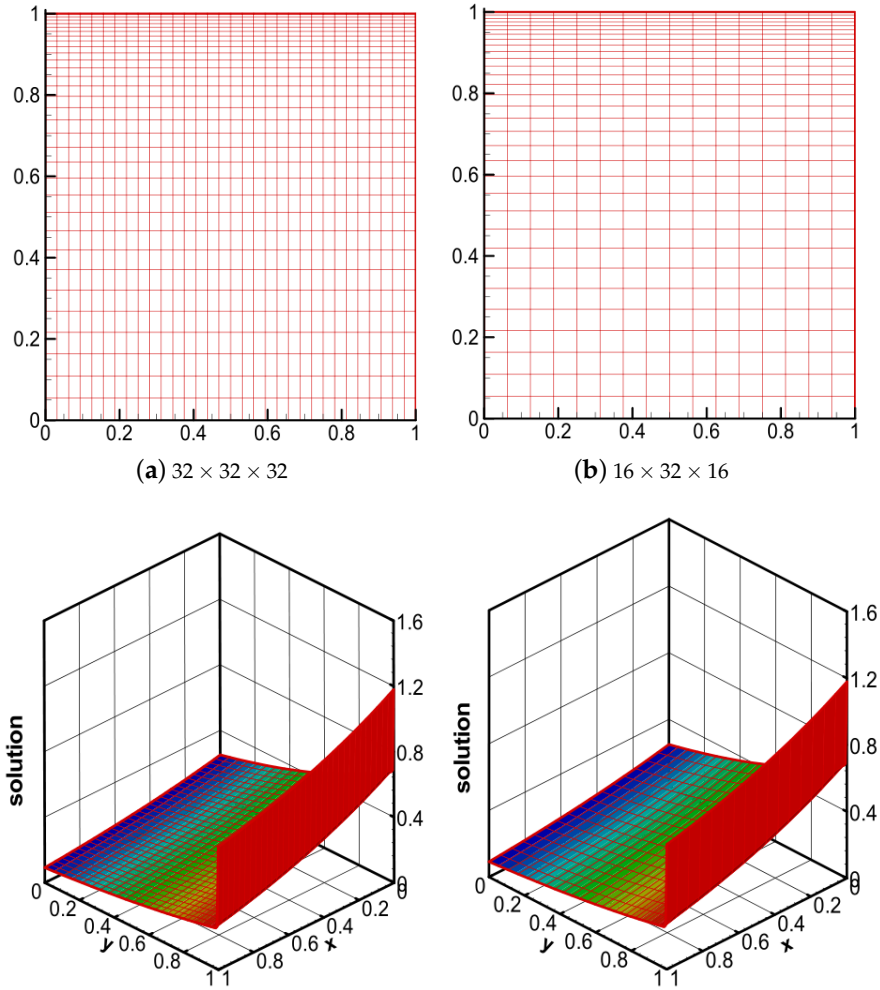
$$x_i = \frac{i}{N_x}, \quad y_j = \frac{j}{N_y} + \frac{\lambda_y}{\pi}\sin(\frac{\pi j}{N_y}), \quad z_k = \frac{k}{N_z}. \tag{32}$$

The closer λ_y is to 1, the more grid points are clustered near $y = 1$.

Tables 3 and 4 list the computed results for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-3}$, with $\lambda_y = 0.55$ and $\lambda_y = 0.9$, respectively. By observing the maximum errors, we find that numerical

solution precision does not decrease when we set fewer grids along the x- and z-axes while the number of grids along the y-axis is fixed. In other words, if the large gradient solution only happens in one coordinate axis, it is not necessary to use the same number of grids for the other two coordinate axes. Under such circumstances, computational cost and storage space would be saved. Additionally, we notice that, for this problem, the multigrid $V(2,2)$ cycles are very efficient and more cost-effective than the multgrid method of full-coarsening [5] with fewer numbers of multgrid iterations and less CPU time. When $\varepsilon = 10^{-3}$, the boundary layer is steeper, but we still can get very accurate solutions on the non-uniform grids with few grid points on the non-dominant axes. The multigrid method of partial semi-coarsening still works well and is superior to the multigrid method of full-coarsening [5].

Figure 8a,b show the grid with $32 \times 32 \times 32$ and $16 \times 32 \times 16$ with $\lambda_y = 0.9$ in the xoy plane, respectively. Figure 8c,d show the computed solution on $32 \times 32 \times 32$ grids and $16 \times 32 \times 16$ grids at the plane $z = 0.5$ for $\varepsilon = 10^{-3}$. We can see that the solutions computed by $16 \times 32 \times 16$ grids and by $32 \times 32 \times 32$ grids have no perceivable difference. It also illustrates that the partial semi-coarsening grid is a very effective strategy to deduce the computational cost and storage space.



(**a**) $32 \times 32 \times 32$ (**b**) $16 \times 32 \times 16$

(**c**) $32 \times 32 \times 32$ (**d**) $16 \times 32 \times 16$

Figure 8. (**a**,**b**) Non-uniform grids with $\lambda_y = 0.9$ in xoy plane; (**c**,**d**) computed solutions on a non-uniform grid, for $\varepsilon = 10^{-3}$, at the plane $z = 0.5$, Problem in Section 4.2.

Table 3. Maximum errors, multigrid iteration numbers, and CPU time for Problem in Section 4.2 ($\varepsilon = 10^{-2}$, $\lambda_y = 0.55$).

Full-Coarsening Grids [5]				Partial Semi-Coarsening Grids			
$N_x \times N_y \times N_z$	Num	CPU	Error	$N_x \times N_y \times N_z$	Num	CPU	Error
$64 \times 64 \times 64$	13	25.370	3.875×10^{-6}	$64 \times 64 \times 64$	13	25.370	3.875×10^{-6}
				$32 \times 64 \times 32$	15	7.066	3.944×10^{-6}
				$16 \times 64 \times 16$	7	0.733	4.357×10^{-6}
				$8 \times 64 \times 8$	5	0.140	3.860×10^{-6}
$32 \times 32 \times 32$	11	1.950	6.095×10^{-5}	$32 \times 32 \times 32$	11	1.950	6.095×10^{-5}
				$16 \times 32 \times 16$	6	0.312	6.564×10^{-5}
				$8 \times 32 \times 8$	5	0.094	5.897×10^{-5}
$16 \times 16 \times 16$	6	0.156	1.196×10^{-3}	$16 \times 16 \times 16$	6	0.156	1.196×10^{-3}
				$8 \times 16 \times 8$	4	0.047	1.271×10^{-3}

Table 4. Maximum errors, multigrid iteration numbers, and CPU time for Problem in Section 4.2 ($\varepsilon = 10^{-3}$, $\lambda_y = 0.9$).

Full-Coarsening Grids [5]				Partial Semi-Coarsening Grids			
$N_x \times N_y \times N_z$	Num	CPU	Error	$N_x \times N_y \times N_z$	Num	CPU	Error
$64 \times 64 \times 64$	10	11.500	3.828×10^{-4}	$64 \times 64 \times 64$	10	11.500	3.828×10^{-4}
				$32 \times 64 \times 32$	6	1.911	4.102×10^{-4}
				$16 \times 64 \times 16$	6	0.480	4.082×10^{-4}
				$8 \times 64 \times 8$	6	0.120	3.885×10^{-4}
$32 \times 32 \times 32$	8	1.181	9.275×10^{-3}	$32 \times 32 \times 32$	8	1.181	9.275×10^{-3}
				$16 \times 32 \times 16$	6	0.234	9.232×10^{-3}
				$8 \times 32 \times 8$	6	0.094	8.737×10^{-3}

4.3. Problem 3

$$-(\Phi_{xx} + \Phi_{yy} + \Phi_{zz}) + p(x,y,z)\Phi_x + q(x,y,z)\Phi_y + r(x,y,z)\Phi_z = f(x,y,z) \ , \tag{33}$$

where

$$\begin{aligned} p(x,y,z) &= \mathrm{Re}x(x-1)(1-2y)\,(1-2z)\,, \\ q(x,y,z) &= \mathrm{Re}y(y-1)(1-2x)\,(1-2z)\,, \\ r(x,y,z) &= \mathrm{Re}z(z-1)(1-2x)\,\,(1-2y)\,. \end{aligned} \tag{34}$$

The exact solution is

$$\Phi(x,y,z) = e^{-\sigma[(x-0.5)^2-(y-0.5)^2]-z^2}. \tag{35}$$

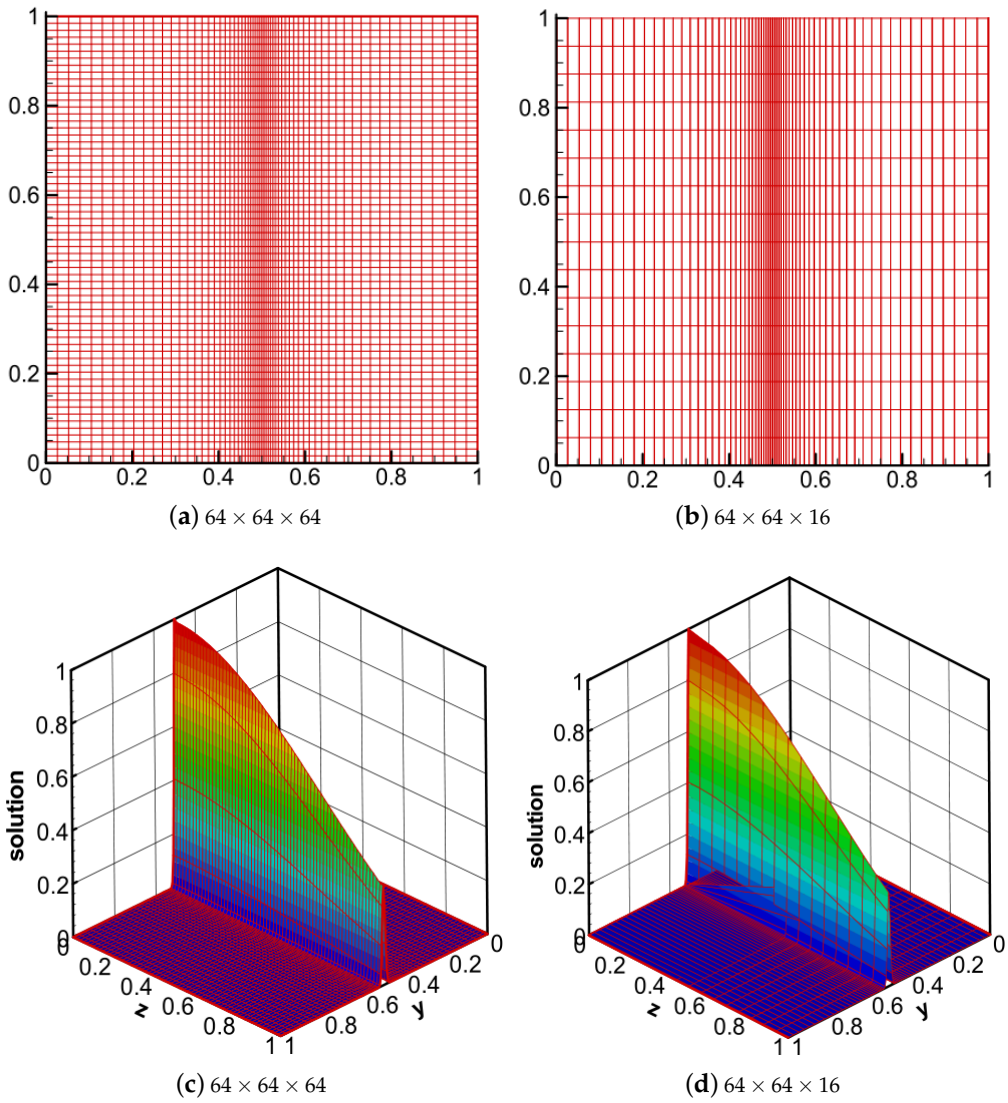
The problem has steep interior layers on the axes $x = 0.5$ and $y = 0.5$ when σ is large. Thus, we distribute many more non-uniform grids on these two axes while fewer uniform grids on the z-axis. The grid functions are given as follows:

$$x_i = \frac{i}{N_x} + \frac{\lambda_x}{2\pi}\sin(\frac{2\pi i}{N_x}), \qquad y_j = \frac{j}{N_y} + \frac{\lambda_y}{\pi}\sin(\frac{\pi j}{N_y}), \qquad z_k = \frac{k}{N_z}. \tag{36}$$

Table 5 shows the numerical results by using the multigrid methods of full-coarsening and partial semi-coarsening for the Problem in Section 4.3 with $\sigma = 10^3$. We choose $\lambda_x = \lambda_y = 0.55$. A multigrid $V(2,2)$ cycle is used. We find that the computational cost (CPU time) dramatically descends because comparatively fewer grid points are used in a non-dominant direction (z-direction). However, the computational accuracy does not drop distinctly. Fewer $V(2,2)$ cycles are needed for the multigrid method of partial semi-coarsening than that of the full-coarsening. It indicates that the multigrid method of partial semi-coarsening is more efficient and cost-effective than that of full-coarsening.

Table 6 gives the computed results when $\sigma = 10^4$ for the Problem in Section 4.3. $\lambda_x = \lambda_y = 0.7$ is used for both the full-coasening multigrid method and the partial semi-coarsening multigrid method. Under such circumstance, the solution in the interior layer changes more violently. The partial semi-coarsening with $64 \times 64 \times 16$ grids gets almost the same accuracy for the numerical results as full-coarsening with $64 \times 64 \times 64$ grids. However, the $V(2,2)$ cycles consumed by the multigrid method of partial semi-coarsening are only half that of the full-coarsening. Meanwhile, the CPU time consumed by the multigrid method of partial semi-coarsening is only about one tenth of that by the full-coarsening.

Figure 9 shows the computed solution (c) with the full-coarsening non-uniform $64 \times 64 \times 64$ grids (a), and the computed solution (d) with the partial semi-coarsening on non-uniform $64 \times 64 \times 16$ grids (b), with $\lambda_x = \lambda_y = 0.7$ for $\sigma = 10^4$, $Re = 10^2$, at the plane $x = 0.5$, for the Problem in Section 4.3. We observe that the difference between them is almost indistinguishable. Thus, we can draw that using fewer grids along the non-dominant direction does not influence the precision of the numerical solution. However, computational cost will be lowered since relatively fewer grids are used.



(**a**) $64 \times 64 \times 64$ (**b**) $64 \times 64 \times 16$

(**c**) $64 \times 64 \times 64$ (**d**) $64 \times 64 \times 16$

Figure 9. (**a**,**b**) Non-uniform grids with $\lambda_x = \lambda_y = 0.7$ in *yoz* plane; (**c**,**d**) Computed solutions on non-uniform grid, for $\sigma = 10^4$, $Re = 10^2$, at the plane $x = 0.5$, Problem in Section 4.3.

Table 5. Maximum errors, multigrid iteration numbers, and CPU time for Problem in Section 4.3 with $\sigma = 10^3$, $\lambda_x = \lambda_y = 0.55$.

Full-Coarsening Grids [5]				Partial Semi-Coarsening Grids			
$N_x \times N_y \times N_z$	Num	CPU	Error	$N_x \times N_y \times N_z$	Num	CPU	Error
$64 \times 64 \times 64$	19	24.320	3.827×10^{-4}	$64 \times 64 \times 64$	19	24.320	3.827×10^{-4}
				$64 \times 64 \times 32$	12	8.615	3.853×10^{-4}
				$64 \times 64 \times 16$	11	3.089	4.175×10^{-4}
				$64 \times 64 \times 8$	11	1.373	4.741×10^{-4}
$32 \times 32 \times 32$	13	1.684	6.652×10^{-3}	$32 \times 32 \times 32$	13	1.684	6.652×10^{-3}
				$33 \times 32 \times 16$	11	0.764	7.116×10^{-3}
				$32 \times 32 \times 8$	11	0.327	7.382×10^{-3}
$16 \times 16 \times 16$	10	0.156	1.515×10^{-1}	$16 \times 16 \times 16$	10	0.156	1.515×10^{-1}
				$16 \times 16 \times 8$	10	0.079	1.558×10^{-1}

Table 6. Maximum errors, multigrid iteration numbers, and CPU time for Problem in Section 4.3 with $\sigma = 10^4$, $\lambda_x = \lambda_y = 0.7$.

Full-Coarsening Grids [5]				Partial Semi-Coarsening Grids			
$N_x \times N_y \times N_z$	Num	CPU	Error	$N_x \times N_y \times N_z$	Num	CPU	Error
$64 \times 64 \times 64$	34	42.870	7.505×10^{-3}	$64 \times 64 \times 64$	34	42.870	7.505×10^{-3}
				$64 \times 64 \times 32$	17	10.890	7.856×10^{-3}
				$64 \times 64 \times 16$	16	4.012	7.968×10^{-3}
				$64 \times 64 \times 8$	16	1.856	8.067×10^{-3}
$32 \times 32 \times 32$	18	2.287	1.916×10^{-1}	$32 \times 32 \times 32$	18	2.287	1.916×10^{-1}
				$32 \times 32 \times 16$	16	0.983	1.923×10^{-1}
				$32 \times 32 \times 8$	16	0.453	1.905×10^{-1}

4.4. Problem 4

$$-\varepsilon(\Phi_{xx} + \Phi_{yy} + \Phi_{zz}) + p(x,y,z)\Phi_x + q(x,y,z)\Phi_y + r(x,y,z)\Phi_z = f(x,y,z), \tag{37}$$

where

$$\begin{aligned} p(x,y,z) &= -x(1-y)(2-z), \\ q(x,y,z) &= -y(1-z)(2-x), \\ r(x,y,z) &= -z(1-x)(2-y). \end{aligned} \tag{38}$$

The exact solution is

$$\Phi(x,y,z) = \frac{e^{x/\varepsilon} + e^{y/\varepsilon} + e^{z/\varepsilon} - 2}{e^{1/\varepsilon} - 1}. \tag{39}$$

We notice that the Problem in Section 4.4 has a steep solution gradient along axes $x = 1$, $y = 1$ and $z = 1$ when ε is small. Under this circumstance, we use grid distribution function as follows in order to set more grid points in the boundary layers on the x-, y-, and z-axes:

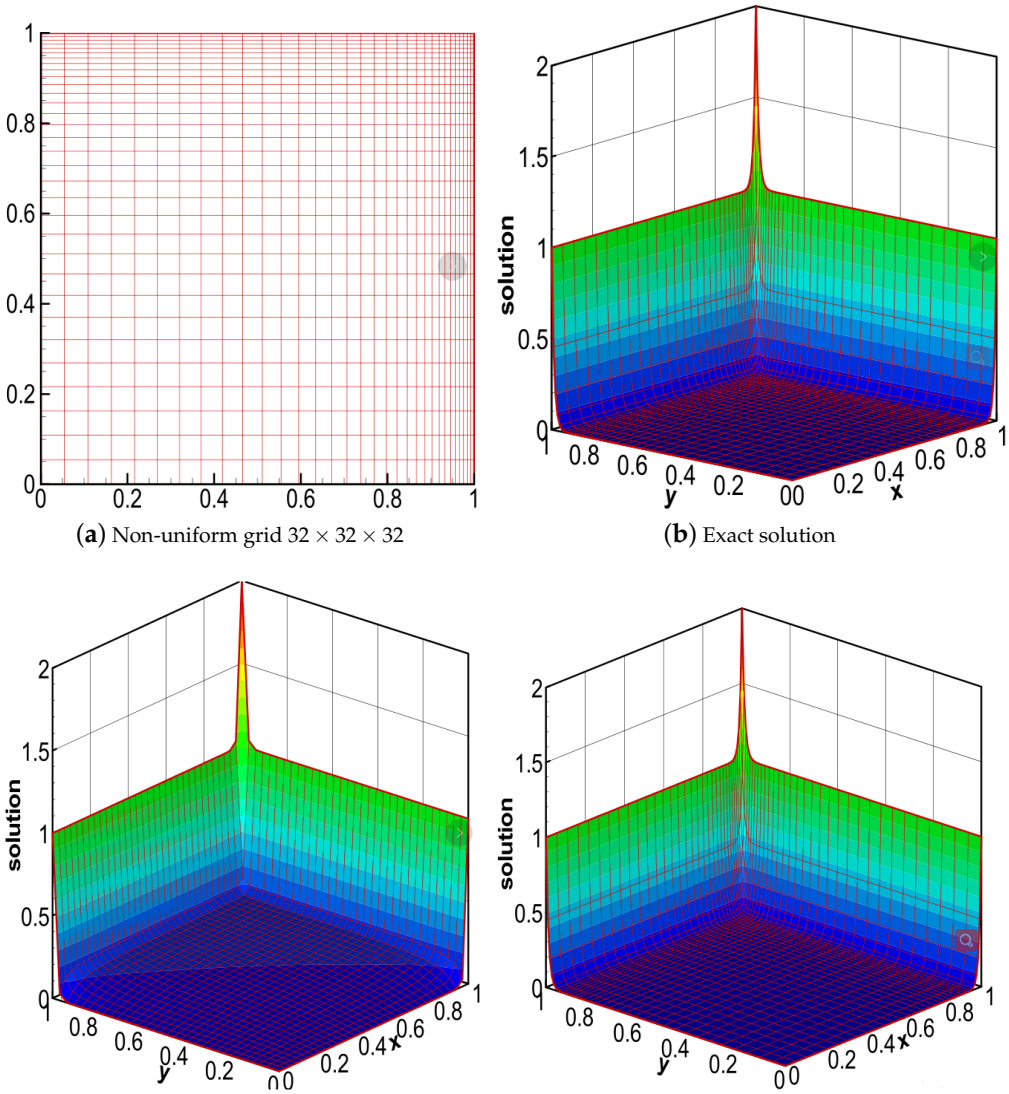
$$x_i = \frac{i}{N_x} + \frac{\lambda_x}{\pi}\sin(\frac{\pi i}{N_x}),\ \ y_j = \frac{j}{N_y} + \frac{\lambda_y}{\pi}\sin(\frac{\pi j}{N_y}),\ \ z_k = \frac{j}{N_z} + \frac{\lambda_z}{\pi}\sin(\frac{\pi j}{N_z}). \tag{40}$$

When $\lambda_x = \lambda_y = \lambda_z = 0.75$, the grid with $32 \times 32 \times 32$ in the xoy plane is shown in Figure 10a.

Table 7 gives the maximum error, convergence order, number of multigrid $V(2,2)$ cycles, and CPU time of the full-coarsening multigrid method with uniform grids and non-uniform grids for the Problem in Section 4.4. We choose $\varepsilon = 0.1, 0.05$ and 0.01.

Correspondingly, grid stretching parameters are set to be $\lambda_x = \lambda_y = \lambda_z = 0.35$, 0.5 and 0.8, respectively. We notice that the maximum errors on non-uniform grids are superior to that on uniform grids. In addition, the multigrid methods are very efficient with both uniform and non-uniform grids. Under this condition, a few more multigrid $V(2,2)$ cycles are needed on non-uniform grids than that on uniform grids. When $\varepsilon = 0.01$, the computed results on the uniform grids are inaccurate, whereas very accurate solutions are obtained on the non-uniform grids. It fully demonstrates that the multigrid method of the partial semi-coarsening in this paper can reduce to that of the full-coarsening if boundary or interior layers exist on all three coordinate axes. In other words, the full-coarsening multigrid method in [5] is the special case of the present partial semi-coarsening multigrid method.

Figure 10 shows the exact solution (b), numerical solution on uniform grid (c), and numerical solution on the non-uniform grid (d) for $\varepsilon = 10^{-2}$, $\lambda_x = \lambda_y = \lambda_z = 0.8$ at the plane $z = 0.8125$. By observing the computed results of Figure 10c,d with the exact solution of Figure 10b, we can get that the numerical solution computed on the non-uniform grid is more accurate than that computed on the uniform grid.



(**a**) Non-uniform grid 32 × 32 × 32 (**b**) Exact solution

(**c**) Uniform grid 32 × 32 × 32 (**d**) Non-uniform grid 32 × 32 × 32

Figure 10. (**a**) Non-uniform grid with $\lambda_x = \lambda_y = \lambda_z = 0.8$ in the xy plane; (**b**) exact solution; (**c**,**d**) computed solutions, for $\varepsilon = 10^{-2}$, at the plane $z = 0.8125$, Problem in Section 4.4.

Table 7. Maximum errors, multigrid iteration numbers, and CPU time for Problem in Section 4.4.

$N_x \times N_y \times N_z$	Uniform Grid				Non-Uniform Grid			
	Num	**CPU**	**Error**	**Order**	**Num**	**CPU**	**Error**	**Order**
$\varepsilon = 0.1$							$\lambda_x = \lambda_y = \lambda_z = 0.35$	
$16 \times 16 \times 16$	6	0.203	4.096×10^{-4}		6	0.203	4.452×10^{-5}	
$32 \times 32 \times 32$	7	2.028	2.615×10^{-5}	3.97	8	2.153	2.748×10^{-6}	4.02
$64 \times 64 \times 64$	7	16.110	1.646×10^{6}	3.99	11	23.910	1.712×10^{-7}	4.00
$\varepsilon = 0.05$							$\lambda_x = \lambda_y = \lambda_z = 0.5$	
$16 \times 16 \times 16$	6	0.234	6.348×10^{-3}		6	0.223	2.921×10^{-4}	
$32 \times 32 \times 32$	7	2.122	4.279×10^{-4}	3.89	10	2.852	1.784×10^{-5}	4.03
$64 \times 64 \times 64$	7	17.510	2.730×10^{-5}	3.97	15	36.230	1.111×10^{-6}	4.01
$\varepsilon = 0.01$							$\lambda_x = \lambda_y = \lambda_z = 0.8$	
$16 \times 16 \times 16$	6	0.171	7.759×10^{-1}		6	0.218	8.423×10^{-3}	
$32 \times 32 \times 32$	6	1.419	1.585×10^{-1}	2.29	12	2.591	4.989×10^{-4}	4.08
$64 \times 64 \times 64$	6	10.220	1.474×10^{-2}	3.43	26	43.180	3.076×10^{-5}	4.02

5. Conclusions

In this study, we have extended the partial semi-coarsening multigrid method introduced in Ref. [23] for solving 2D boundary or interior layer problems of convection–diffusion equations to 3D cases. The highlight of this method is that it is allowed to use different numbers of grids in different coordinate axes, so it is cost-effective and suitable for solving the problems with local large gradients or boundary/interior layers that only exist on one or two axes for 3D problems. It overcomes the defect in our previous work in Ref. [5] with same numbers of grid points being used in all coordinate axes, which leads to a big waste of CPU time and storage. On the other hand, if the problems with local large gradients or boundary layers exist in all three coordinate axes, the present partial semi-coarsening multigrid method reduces to the full-coarsening multigrid method [5]. Thus, the present partial semi-coarsening strategy is more flexible for different types of boundary or interior layer problems. The computed results of numerical experiments validate that the partial semi-coarsening multigrid method, by using fewer grid points in non-dominant direction(s), is very efficient and cost-effective compared to the full-coarsening multigrid method, but the computed accuracy is not lost.

We emphasize that, although this work is the generalization of the work for 2D in Ref. [23], it is not straightforward—at least, their implementations are nontrivial—because only one dominant direction exists for 2D boundary or interior layers problems, but two dominant directions may exist for 3D cases. This inevitably adds complexity for the algorithm designing and program implementation. In addition, the present method can also be extended to solve 2D and 3D boundary layer fluid flow problems governed by the incompressible Navier–Stokes equations. This is beyond the scope of this paper, and we plan to do this research in the near future.

Author Contributions: Conceptualization, Y.G.; Methodology, T.M., F.C., and Y.G.; software, F.C. and T.M.; validation and formal analysis, F.C.; investigation, T.M. and F.C.; resources, F.C. and Y.G.; writing—original draft preparation, T.M., F.C. and L.Z.; writing—review and editing, Y.G. and L.Z.; funding acquisition, F.C. and Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Natural Science Foundation of China (Grants No. 11772165, 11961054, 11902170,11801287), Inner Mongolia Autonomous Region "Youth Science and Technology Talents" support program (Grant No. NJYT20B15), Inner Mongolia Scientific Fund Project (Grants No. 2018BS01002,2018LH01008), Innovation Fund Project of Inner Mongolia University of Science and Technology—Excellent Youth Science Fund Project(Grant No. 2019YQL02), National Youth Top-Notch Talent Support Program of Ningxia, and the First Class Discipline Construction Project in Ningxia Universities: Mathematics.

Data Availability Statement: The data presented in this paper are calculated through numerical experiments and are available.

Acknowledgments: We would like to thank the editors and the referees whose constructive comments and suggestions are helpful to improve the quality of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

2D	two-dimensional
3D	three-dimensional
HOC	high-order compact
Φ	unknown function
p	convective coefficient on the x-axis
q	convective coefficient on the y-axis
r	convective coefficient on the z-axis
f	the forcing function
x	spatial coordinate on the x-axis
y	spatial coordinate on the y-axis
z	spatial coordinate on the z-axis
Ω	computational domain
$\partial\Omega$	domain boundary
CPU	computing time
$[a_1, a_2]$	computational domain on the x-axis, $a_1 < a_2$
$[b_1, b_2]$	computational domain on the y-axis, $b_1 < b_2$
$[c_1, c_2]$	computational domain on the z-axis, $c_1 < c_2$
x_i	discrete point coordinate on the x-axis
y_i	discrete point coordinate on the y-axis
z_i	discrete point coordinate on the z-axis
N_x	number of discrete grids on the x-axis
N_y	number of discrete grids on the y-axis
N_z	number of discrete grids on the z-axis
i	discrete grid index on the x-axis on the fine grid
j	discrete grid index on the y-axis on the fine grid
k	discrete grid index on the z-axis on the fine grid
$\bar{i}$	discrete grid index on the x-axis on the coarse grid
$\bar{j}$	discrete grid index on the y-axis on the coarse grid
$\bar{k}$	discrete grid index on the z-axis on the coarse grid
x_b	backward step length on the x-axis
x_f	forward step length on the x-axis
y_b	backward step lengths on the y-axis
y_f	forward step lengths on the y-axis
z_b	backward step lengths on the z-axis
z_f	forward step lengths on the z-axis
δ_x, δ_x^2	the first and second order central difference operator on the x-axis
δ_y, δ_y^2	the first and second order central difference operator on the y-axis
δ_z, δ_z^2	the first and second order central difference operator on the z-axis
r	residual on the fine grid
$\bar{r}$	residual on the coarse grid
$S_i (i = 1, \cdots, 8)$	area of the i-th subregion
ν_1	pre-smoothing times
ν_2	post-smoothing times
Error	maximum absolute error
Num	numbers of multigrid iterations
Order	convergence order
log	base 10 logarithmic function
Re	Reynolds number
σ	boundary or interior layer parameters
ϵ	diffusion coefficient
λ_x	grids stretching parameters on the x-axis
λ_y	grids stretching parameters on the y-axis
λ_z	grids stretching parameters on the z-axis

Appendix A. Details of the Finite Difference Operators

The expressions of the finite difference operators appearing in Equation (2) are listed as follows [5]:

$$\delta_x\Phi_0 = \frac{1}{2h}\left(\frac{x_b}{x_f}\Phi_1 - \frac{x_f}{x_b}\Phi_3\right) + \left(\frac{1}{x_b} - \frac{1}{x_f}\right)\Phi_0, \tag{A1}$$

$$\delta_y\Phi_0 = \frac{1}{2k}\left(\frac{y_b}{y_f}\Phi_2 - \frac{y_f}{y_b}\Phi_4\right) + \left(\frac{1}{y_b} - \frac{1}{y_f}\right)\Phi_0, \tag{A2}$$

$$\delta_z\Phi_0 = \frac{1}{2l}\left(\frac{z_b}{z_f}\Phi_5 - \frac{z_f}{z_b}\Phi_6\right) + \left(\frac{1}{z_b} - \frac{1}{z_f}\right)\Phi_0, \tag{A3}$$

$$\delta_x^2\Phi_0 = \frac{1}{h}\left[\frac{\Phi_1}{x_f} - \left(\frac{1}{x_f} + \frac{1}{x_b}\right)\Phi_0 + \frac{\Phi_3}{x_b}\right], \tag{A4}$$

$$\delta_y^2\Phi_0 = \frac{1}{k}\left[\frac{\Phi_2}{y_f} - \left(\frac{1}{y_f} + \frac{1}{y_b}\right)\Phi_0 + \frac{\Phi_4}{y_b}\right], \tag{A5}$$

$$\delta_z^2\Phi_0 = \frac{1}{l}\left[\frac{\Phi_5}{z_f} - \left(\frac{1}{z_f} + \frac{1}{z_b}\right)\Phi_0 + \frac{\Phi_6}{z_b}\right], \tag{A6}$$

$$\delta_x\delta_y\Phi_0 = \frac{1}{4hk}(\Phi_7 - \Phi_8 - \Phi_{10} + \Phi_9), \tag{A7}$$

$$\delta_x\delta_z\Phi_0 = \frac{1}{4hl}(\Phi_{11} - \Phi_{13} - \Phi_{15} + \Phi_{17}), \tag{A8}$$

$$\delta_y\delta_z\Phi_0 = \frac{1}{4kl}(\Phi_{12} - \Phi_{14} - \Phi_{16} + \Phi_{18}), \tag{A9}$$

$$\delta_x\delta_y^2\Phi_0 = \frac{1}{2hk}\left[\frac{1}{y_f}(\Phi_7 - \Phi_8) - \left(\frac{1}{y_f} + \frac{1}{y_b}\right)(\Phi_1 - \Phi_3) + \frac{1}{y_b}(\Phi_{10} - \Phi_9)\right], \tag{A10}$$

$$\delta_x^2\delta_y\Phi_0 = \frac{1}{2hk}\left[\frac{1}{x_f}(\Phi_7 - \Phi_{10}) - \left(\frac{1}{x_f} + \frac{1}{x_b}\right)(\Phi_2 - \Phi_4) + \frac{1}{x_b}(\Phi_8 - \Phi_9)\right], \tag{A11}$$

$$\delta_x\delta_z^2\Phi_0 = \frac{1}{2hl}\left[\frac{1}{z_f}(\Phi_{11} - \Phi_{13}) - \left(\frac{1}{z_f} + \frac{1}{z_b}\right)(\Phi_1 - \Phi_3) + \frac{1}{z_b}(\Phi_{15} - \Phi_{17})\right], \tag{A12}$$

$$\delta_x^2\delta_z\Phi_0 = \frac{1}{2hl}\left[\frac{1}{x_f}(\Phi_{11} - \Phi_{15}) - \left(\frac{1}{x_f} + \frac{1}{x_b}\right)(\Phi_5 - \Phi_6) + \frac{1}{x_b}(\Phi_{13} - \Phi_{17})\right], \tag{A13}$$

$$\delta_y\delta_z^2\Phi_0 = \frac{1}{2kl}\left[\frac{1}{z_f}(\Phi_{12} - \Phi_{14}) - \left(\frac{1}{z_f} + \frac{1}{z_b}\right)(\Phi_2 - \Phi_4) + \frac{1}{z_b}(\Phi_{16} - \Phi_{18})\right], \tag{A14}$$

$$\delta_y^2\delta_z\Phi_0 = \frac{1}{2kl}\left[\frac{1}{y_f}(\Phi_{12} - \Phi_{16}) - \left(\frac{1}{y_f} + \frac{1}{y_b}\right)(\Phi_5 - \Phi_6) + \frac{1}{y_b}(\Phi_{14} - \Phi_{18})\right], \tag{A15}$$

$$\delta_x^2\delta_y^2\Phi_0 = \frac{1}{hk}\left[\frac{\Phi_7}{x_f y_f} + \frac{\Phi_8}{x_b y_f} + \frac{\Phi_9}{x_b y_b} + \frac{\Phi_{10}}{x_f y_b} + \left(\frac{1}{x_f y_f} + \frac{1}{x_f y_b} + \frac{1}{x_b y_f} + \frac{1}{x_b y_b}\right)\Phi_0 \right.$$
$$- \left(\frac{1}{x_f y_f} + \frac{1}{x_f y_b}\right)\Phi_1 - \left(\frac{1}{x_f y_f} + \frac{1}{x_b y_f}\right)\Phi_2 - \left(\frac{1}{x_b y_f} + \frac{1}{x_b y_b}\right)\Phi_3$$
$$\left. - \left(\frac{1}{x_f y_b} + \frac{1}{x_b y_b}\right)\Phi_4\right], \tag{A16}$$

$$\delta_x^2\delta_z^2\Phi_0 = \frac{1}{hl}\left[\frac{\Phi_{11}}{x_f z_f} + \frac{\Phi_{13}}{x_b z_f} + \frac{\Phi_{15}}{x_f z_b} + \frac{\Phi_{17}}{x_b z_b} + \left(\frac{1}{x_f z_f} + \frac{1}{x_b z_f} + \frac{1}{x_f z_b} + \frac{1}{x_b z_b}\right)\Phi_0 \right.$$
$$- \left(\frac{1}{x_f z_f} + \frac{1}{x_f z_b}\right)\Phi_1 - \left(\frac{1}{x_b z_f} + \frac{1}{x_b z_b}\right)\Phi_3 - \left(\frac{1}{x_f z_f} + \frac{1}{x_b z_f}\right)\Phi_5$$
$$\left. - \left(\frac{1}{x_f z_b} + \frac{1}{x_b z_b}\right)\Phi_6\right], \tag{A17}$$

$$\delta_x^2\delta_z^2\Phi_0 = \frac{1}{kl}\left[\frac{\Phi_{12}}{y_f z_f} + \frac{\Phi_{14}}{y_b z_f} + \frac{\Phi_{16}}{y_f z_b} + \frac{\Phi_{18}}{y_b z_b} + \left(\frac{1}{y_f z_f} + \frac{1}{y_b z_f} + \frac{1}{y_f z_b} + \frac{1}{y_b z_b}\right)\Phi_0 \right.$$
$$- \left(\frac{1}{y_f z_f} + \frac{1}{y_f z_b}\right)\Phi_2 - \left(\frac{1}{y_b z_f} + \frac{1}{y_b z_b}\right)\Phi_4 - \left(\frac{1}{y_f z_f} + \frac{1}{y_b z_f}\right)\Phi_5$$
$$\left. - \left(\frac{1}{y_f z_b} + \frac{1}{y_b z_b}\right)\Phi_6\right], \tag{A18}$$

in which $h = \frac{1}{2}(x_f + x_b)$, $k = \frac{1}{2}(y_f + y_b)$, $l = \frac{1}{2}(z_f + z_b)$.

Appendix B. Restriction Operator on a Non-Uniform Grid When $N_x = N_y = N_z$

When $N_x = N_y = N_z$, the full weighting restriction operator on non-uniform grids is explicitly written out as [7]

$$\begin{aligned}\bar{r}_{\bar{i},\bar{j},\bar{k}} = \frac{1}{V}(&V_0 r_{i,j,k} + V_1 r_{i-1,j,k} + V_2 r_{i,j-1,k} + V_3 r_{i+1,j,k} + V_4 r_{i,j+1,k} + V_5 r_{i,j,k-1} + V_6 r_{i,j,k+1} \\ &+ V_7 r_{i-1,j-1,k} + V_8 r_{i+1,j-1,k} + V_9 r_{i+1,j+1,k} + V_{10} r_{i-1,j+1,k} + V_{11} r_{i-1,j,k-1} \\ &+ V_{12} r_{i,j-1,k-1} + V_{13} r_{i+1,j,k-1} + V_{14} r_{i,j+1,k-1} + V_{15} r_{i-1,j,k+1} + V_{16} r_{i,j-1,k+1} \\ &+ V_{17} r_{i+1,j,k+1} + V_{18} r_{i,j+1,k+1} + V_{19} r_{i-1,j-1,k-1} + V_{20} r_{i+1,j-1,k-1} + V_{21} r_{i+1,j+1,k-1} \\ &+ V_{22} r_{i-1,j+1,k-1} + V_{23} r_{i-1,j-1,k+1} + V_{24} r_{i+1,j-1,k+1} + V_{25} r_{i+1,j+1,k+1} \\ &+ V_{26} r_{i-1,j+1,k+1}),\end{aligned} \tag{A19}$$

in which

$$\begin{aligned}&V = (x_f + x_b) \times (y_f + y_b) \times (z_f + z_b), \\ &V_0 = \frac{1}{8}(x_f + x_b) \times (y_f + y_b) \times (z_f + z_b), \quad V_1 = \frac{1}{8}x_f \times (y_f + y_b) \times (z_f + z_b), \\ &V_2 = \frac{1}{8}y_f \times (x_f + x_b) \times (z_f + z_b), \quad V_3 = \frac{1}{8}x_b \times (y_f + y_b) \times (z_f + z_b), \\ &V_4 = \frac{1}{8}y_b \times (x_f + x_b) \times (z_f + z_b), \quad V_5 = \frac{1}{8}z_f \times (x_f + x_b) \times (y_f + y_b), \\ &V_6 = \frac{1}{8}z_b \times (x_f + x_b) \times (y_f + y_b), \quad V_7 = \frac{1}{8}x_f \times y_f \times (z_f + z_b),\end{aligned}$$

$$V_8 = \frac{1}{8}x_b \times y_f \times (z_f + z_b),\ V_9 = \frac{1}{8}x_b \times y_b \times (z_f + z_b),\ V_{10} = \frac{1}{8}x_f \times y_b \times (z_f + z_b),$$
$$V_{11} = \frac{1}{8}x_f \times z_f \times (y_f + y_b),\ V_{12} = \frac{1}{8}y_f \times z_f \times (x_f + x_b),\ V_{13} = \frac{1}{8}x_b \times z_f \times (y_f + y_b),$$
$$V_{14} = \frac{1}{8}y_b \times z_f \times (x_f + x_b),\ V_{15} = \frac{1}{8}x_f \times z_b \times (y_f + y_b),\ V_{16} = \frac{1}{8}y_f \times z_b \times (x_f + x_b),$$
$$V_{17} = \frac{1}{8}x_b \times z_b \times (y_f + y_b),\ V_{18} = \frac{1}{8}y_b \times z_b \times (x_f + x_b),\ V_{19} = \frac{1}{8}x_f \times y_f \times z_f,$$
$$V_{20} = \frac{1}{8}x_b \times y_f \times z_f,\ V_{21} = \frac{1}{8}x_b \times y_b \times z_f,\ V_{22} = \frac{1}{8}x_f \times y_b \times z_f,$$
$$V_{23} = \frac{1}{8}x_f \times y_f \times z_b,\ V_{24} = \frac{1}{8}x_b \times y_f \times z_b,\ V_{25} = \frac{1}{8}x_b \times y_b \times z_b,\ V_{26} = \frac{1}{8}x_f \times y_b \times z_b.$$

Appendix C. Interpolation Operator on a Non-Uniform Grid When $N_x = N_y = N_z$

When $N_x = N_y = N_z$, the tri-linear interpolation operator on non-uniform grids is explicitly written out as [7]

$$r_{i,j,k} = \bar{r}_{\bar{i},\bar{j},\bar{k}}, \quad r_{i-1,j,k} = \frac{1}{x_f + x_b}(x_f \bar{r}_{\bar{i}-1,\bar{j},\bar{k}} + x_b \bar{r}_{\bar{i},\bar{j},\bar{k}}), \tag{A20}$$

$$r_{i,j-1,k} = \frac{1}{y_f + y_b}(y_f \bar{r}_{\bar{i},\bar{j}-1,\bar{k}} + y_b \bar{r}_{\bar{i},\bar{j},\bar{k}}), \quad r_{i,j,k-1} = \frac{1}{z_f + z_b}(z_f \bar{r}_{\bar{i},\bar{j},\bar{k}-1} + z_b \bar{r}_{\bar{i},\bar{j},\bar{k}}), \tag{A21}$$

$$r_{i-1,j-1,k} = \frac{1}{S_{xy}}(S_{1xy}\bar{r}_{\bar{i}-1,\bar{j}-1,\bar{k}} + S_{2xy}\bar{r}_{\bar{i},\bar{j}-1,\bar{k}} + S_{3xy}\bar{r}_{\bar{i},\bar{j},k} + S_{4xy}\bar{r}_{\bar{i}-1,\bar{j},\bar{k}}), \tag{A22}$$

$$r_{i,j-1,k-1} = \frac{1}{S_{yz}}(S_{1yz}\bar{r}_{\bar{i},\bar{j}-1,\bar{k}-1} + S_{2yz}\bar{r}_{\bar{i},\bar{j},\bar{k}-1} + S_{3yz}\bar{r}_{\bar{i},\bar{j},\bar{k}} + S_{4yx}\bar{r}_{\bar{i},\bar{j}-1,\bar{k}}), \tag{A23}$$

$$r_{i-1,j,k-1} = \frac{1}{S_{xz}}(S_{1xz}\bar{r}_{\bar{i}-1,\bar{j},\bar{k}-1} + S_{2xz}\bar{r}_{\bar{i},\bar{j},\bar{k}-1} + S_{3xz}\bar{r}_{\bar{i},\bar{j},\bar{k}} + S_{4xy}\bar{r}_{\bar{i}-1,\bar{j},\bar{k}}), \tag{A24}$$

$$r_{i-1,j-1,k-1} = \frac{1}{\tilde{V}}(\tilde{V}_1 r_{\bar{i}-1,\bar{j}-1,\bar{k}-1} + \tilde{V}_2\bar{r}_{\bar{i},\bar{j}-1,\bar{k}-1} + \tilde{V}_3\bar{r}_{\bar{i},\bar{j},\bar{k}-1} + \tilde{V}_4\bar{r}_{\bar{i}-1,\bar{j},\bar{k}-1} + \tilde{V}_5\bar{r}_{\bar{i}-1,\bar{j}-1,\bar{k}}$$
$$+ \tilde{V}_6\bar{r}_{\bar{i},\bar{j}-1,\bar{k}} + \tilde{V}_7\bar{r}_{\bar{i},\bar{j},\bar{k}} + \tilde{V}_8\bar{r}_{\bar{i}-1,\bar{j},\bar{k}}), \tag{A25}$$

in which

$$S_{xy} = (x_f + x_b) \times (y_f + y_b),\ S_{yz} = (y_f + y_b) \times (z_f + z_b),\ S_{xz} = (x_f + x_b) \times (z_f + z_b),$$
$$S_{1xy} = x_f \times y_f, \quad S_{2xy} = x_b \times y_f, \quad S_{3xy} = x_b \times y_b, \quad S_{4xy} = x_f \times y_b,$$
$$S_{1yz} = y_f \times z_f, \quad S_{2yz} = y_b \times z_f, \quad S_{3yz} = y_b \times z_b, \quad S_{4zy} = y_f \times z_b,$$
$$S_{1xz} = x_f \times z_f, \quad S_{2xz} = x_b \times z_f, \quad S_{3xz} = x_b \times z_b, \quad S_{4xz} = x_f \times z_b,$$

and

$$\tilde{V} = (x_f + x_b) \times (y_f + y_b) \times (z_f + z_b),$$
$$\tilde{V}_1 = x_f \times y_f \times z_f, \quad \tilde{V}_2 = x_b \times y_f \times z_f, \quad \tilde{V}_3 = x_b \times y_b \times z_f, \quad \tilde{V}_4 = x_f \times y_b \times z_f,$$
$$\tilde{V}_5 = x_f \times y_f \times z_b, \quad \tilde{V}_6 = x_b \times y_f \times z_b, \quad \tilde{V}_7 = x_b \times y_b \times z_b, \quad \tilde{V}_8 = x_f \times y_b \times z_b.$$

References

1. Batchelor, G.K. *An Introduction to Fluid Dynamics*; Cambridge University Press: Cambridge, UK, 1967.
2. Blazek, J. *Computational Fluid Dynamics: Principles and Applications*; Butterworth-Heinemann: Oxford, UK, 2005.
3. Zhang, J. An explicit fourth-order compact finite difference scheme for three-dimensional convection–diffusion equation. *Commun. Numer. Methods Eng.* **1998**, *14*, 263–280. [CrossRef]
4. Zhang, J.; Ge, L.; Gupta M.M. Fourth order compact difference schemes for 3D convection diffusion eqution with boundary layers on non-uniform grid. *Neural Parallel Sci. Comput.* **2000**, *8*, 373–392.

5. Ge, Y.; Cao, F. A high order compact difference scheme and multigrid method for solving the 3D convection diffusion equation on non-uniform grids. In Proceedings of the 2012 Fourth International Conference on Computational and Information Sciences (ICCIS'12), Chongqing, China, 17–19 August 2012; pp. 714–717. [CrossRef]
6. Dai, R.; Wang, Y.; Zhang, J. Fast and high accuracy multiscale multigrid method with multiple coarse grid updating strategy for the 3D convection–diffusion equation. *Comput. Math. Appl.* **2013**, *66*, 542–559. [CrossRef]
7. Ge, Y.; Cao, F.; Zhang, J. A transformation-free HOC scheme and multigrid method for solving the 3D Poisson equation on non-uniform grids. *J. Comput. Phys.* **2013**, *234*, 199–216. [CrossRef]
8. Gupta, M.M.; Manohar, R.P.; Stephenson, J.W. A single cell high order scheme for the convection–diffusion equation with variable coefficients. *Int. J. Numer. Meth. Fluids* **1984**, *4*, 641–651. [CrossRef]
9. Ge, L.; Zhang, J. High accuracy iterative solution of convection diffusion equation with boundary layers on non-uniform grids. *J. Comput. Phys.* **2001**, *171*, 560–578. [CrossRef]
10. Radhakrishna Pillai, A.C. Fourth-order exponential finite difference methods for boundary value problems of convective diffusion type. *Int. J. Numer. Meth. Fluids* **2001**, *37*, 87–106. [CrossRef]
11. Kalita, J.C.; Dass, A.K.; Dalal, D.C. A transformation-free HOC scheme for steady convection–diffusion on non-uniform grids. *Int. J. Numer. Meth. Fluids* **2004**, *44*, 33–53. [CrossRef]
12. Tian, Z.F.; Dai, S.Q. High-order compact exponential finite difference methods for convection–diffusion type problems. *J. Comput. Phys.* **2007**, *220*, 952–974. [CrossRef]
13. Farrell, P.A.; Hegarty, A.F.; Miller, J.J.H.; O'Rordan, E.; Shishkin, G.I. *Robust Computational Techniques for Boundary Layers*; Champan & Hall/CRC: Boca, Raton, FL, USA, 2000.
14. Zhang, J.; Sun, H.; Zhao, J. High order compact scheme with multigrid local mesh refinement procedure for convection diffusion problems. *Comput. Methods Appl. Mech. Eng.* **2002**, *191*, 4661–4674. [CrossRef]
15. Shanab, R.A.; Seddek, L.F.; Mohamed, S.A. Non-uniform HOC scheme for the 3D convection–diffusion equation. *Appl. Comput. Math.* **2013**, *2*, 64–77. [CrossRef]
16. Brandt, A. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.* **1977**, *31*, 333–390. [CrossRef]
17. Li, M.; Zheng, Z.; Pan, K. Extrapolation multiscale multigrid method for solving 2D Poisson equation with sixth order compact scheme. *J. Appl. Math. Comput.* **2019**, *60*, 589–604. [CrossRef]
18. Gupta, M.M.; Kouatchou, J.; Zhang, J. Comparison of second- and fourth-order discretization for multigrid Poisson solvers. *J. Comput. Phys.* **1997**, *132*, 226–232. [CrossRef]
19. Zhang, J. Multigrid method and fourth-order compact scheme for 2D Poisson equation with unequal mesh-size discretization. *J. Comput. Phys.* **2002**, *179*, 170–179. [CrossRef]
20. Ge, Y. Multigrid method and fourth-order compact difference discretization scheme with unequal meshsizes for 3D Poisson equation. *J. Comput. Phys.* **2010**, *229*, 6381–6391. [CrossRef]
21. Gupta, M.M.; Kouatchou, J.; Zhang, J. A compact multigrid solver for convection–diffusion equations. *J. Comput. Phys.* **1997**, *132*, 123–129. [CrossRef]
22. Ge, Y.; Cao, F. Multigrid method based on the transformation-free HOC scheme on non-uniform grids for 2D convection diffusion problems. *J. Comput. Phys.* **2011**, *230*, 4051–4070. [CrossRef]
23. Cao F.; Ge Y.; Sun H.-W. Partial semi-coarsening multigrid method based on the HOC scheme on non-uniform grids for the convection–diffusion problems. *Int. J. Comput. Math.* **2017**, *94*, 2356–2372. [CrossRef]
24. Medina, A.C.; Schmid, R. Solution of high order compact discretized 3D elliptic partial differential equations by an accelerated multigrid method. *J. Comput. Appl. Math.* **2019**, *350*, 343–352. [CrossRef]
25. Wesseling, P. *An Introduction to Multigrid Methods*; Wiley: Chichester, UK, 1992.
26. Mulder, W.A. A new multigrid approach to convection problems. *J. Comput. Phys.* **1989**, *83*, 303–317. [CrossRef]
27. Liu, C.; Liu, Z. Multigrid mapping and box relaxation for simulation of the whole process of flow transition in 3D boundary layers. *J. Comput. Phys.* **1995**, *119*, 325–341. [CrossRef]
28. Liu, C. Multilevel Adaptive Methods in Computational Fluid Dynamics. Ph.D Thesis, University of Colorado Denver, Denver, CO, USA, 1989.