Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona,Spain, July 9-13, 2018

ICCFD10-2018-203

An Algorithm for Unsteady Incompressible Flows on an Adaptively Refined Quadtree Grid

Kshitiz K. Subedi*, Matthew V. Fischels* and R.G. Rajagopalan*
Corresponding author: kshitiz@iastate.edu

* Department of Aerospace Engineering, Iowa State University, Ames, Iowa, USA.

Abstract: Unsteady flow over bluff bodies are numerically difficult to solve as the fluid flow equations are stiff in both space and time domains. The flow features change rapidly during the simulation due to the changes in system non-linearities. In this paper, a Runge Kutta based lowspeed algorithm is developed for adaptively refined quadtree grids. Quadtree grids provide better grid adaption based on the flow characteristics. The Runge Kutta based time integration schemes are found to be more efficient. Results for candidate cases are compared with traditional time integration schemes such as Crank-Nicholson, and are found to be more efficient and robust.

keywords: Quadtree grid, Runge Kutta, Incompressible flow, adaption, SIMPLER

1 Introduction

Grid generation on complex bodies need special attention to model the body accurately. In addition, the flow over these bodies may involve rapidly changing flow fields, and flow adaption may be the most economical way to capture the flow accurately. Quadtree and Octree grid generation techniques are fast and suitable for complex bodies and flow adaption [1][2]. In this paper, a new algorithm based on Runge Kutta time integration is developed for lowspeed flows which are governed by unsteady incompressible Navier Stokes equations. These equations are stiff as there is no explicit conservation equation for the pressure and pose many challenges to the numerical algorithm. Pressure based methods based on SIMPLE/SIMPLER algorithms [3] are a popular approach that indirectly specifies the pressure field via the continuity equation. The discretized momentum equations are substituted in the discretized continuity equation to obtain the pressure correction and/or pressure equation. These equations are solved iteratively to obtain the correct velocity and pressure field. In unsteady flows, these algorithms necessitate the use of relaxation and several sub-iterations at each time step. Also, the approximate nature of pressure correction equation has caused convergence issues for various problems. The idea of developing more efficient and faster algorithms has led to the formulation of Runge Kutta (RK) based SIMPLER algorithms [4][5]. These RK based algorithms are found to simulate both steady and unsteady problems accurately with the significant reduction in the required computation time on both structured and unstructured grids [4]. In an effort to improve computation time for complex bodies and flows that may require flow adaption, a Runge Kutta based lowspeed flow algorithm is developed in this paper for an adaptively refined quadtree and octree grids.

2 Development of Flow Solver

The unsteady, laminar incompressible Navier stokes equation governs the fluid flow problems used in this research. For a fluid passing through an infinitesimal fixed control volume in space, these equations can be written in integral form as

$$\frac{\partial}{\partial t}\int\limits_{C.V} \rho d\forall + \int\limits_{C.S} (\rho \vec{V} \cdot d\vec{A}) = 0 \tag{1}$$

$$\frac{\partial}{\partial t}\int\limits_{C.V}\rho\vec{V}d\forall\ +\int\limits_{C.S}(\vec{V}\rho\vec{V}\cdot d\vec{A})=\int\limits_{C.V}\rho\vec{f}d\forall\ +\int\limits_{C.S}(\widetilde{\Pi}\cdot d\vec{A})\ +\int\limits_{C.V}\vec{S}d\forall \tag{2}$$

where t is time, ρ is density, $\vec{V}$ represents the flow velocity, $\vec{f}$ represents the body force, $\widetilde{\Pi}$ is the stress tensor, and $\vec{S}$ is the source term including all other terms that are not accounted for in presented terms. Also, $d\vec{A}$ is the elemental area of control surface $C.S$ and $d\forall$ is the volume of the infinitesimal control volume $C.V$. For an incompressible flow, the density is a constant.

The governing equations can be separately discretized in space and time so that different order of accuracy can be obtained in space and time.

2.1 Spatial Discretization of a General Transport Equation

The governing fluid flow equations inherit a conservation principle. A scalar transport equation which has the same form is the representative of these flow equations. This scalar equation is discretized spatially in the context of two dimensional co-ordinates for simplicity, and the extension to 3D follows the same principle. Also, the discretized momentum equations can be inferred from the discretized scalar transport equation.

The integral formulation of a generalized transport equation for a scalar ϕ can be written as

$$\frac{\partial}{\partial t}\int\limits_{C.V}(\rho\phi)d\forall\ +\int\limits_{C.S}(\rho\vec{V}\phi\cdot d\vec{A})=\int\limits_{C.S}(\Gamma\nabla\phi\cdot d\vec{A})\ +\int\limits_{C.V}Sd\forall \tag{3}$$

where Γ is the diffusion coefficient,and S represents the source term.

Discretizing above equation about each face 'f' of a generalized unstructured control volume, we get

$$\frac{\partial(\rho\phi)}{\partial t}\Delta\forall\ +\sum_{f}(\rho\vec{V}\phi\cdot\vec{A})_f=\sum_{f}(\Gamma\nabla\phi\cdot\vec{A})_f\ +S\Delta\forall \tag{4}$$

where,

F_f=$(\rho\vec{V}\cdot d\vec{A})_f$ is the mass flow rate out of face 'f' of the control volume $\Delta\forall$,

G_f=$(\Gamma\nabla\phi\cdot d\vec{A})_f$ is the transport due to diffusion through the face 'f', and

ϕ_f is the value of ϕ at the face 'f' of control volume

2.1.1 Convection Term

The convection term in the discretized transport equation is the product of F_f, and ϕ_f. Second-Order upwind scheme is utilized to find the value of ϕ at the face of a control volume. In this scheme, the value of

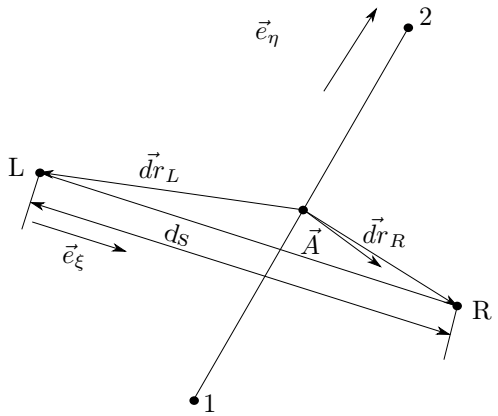


Figure 1: Stencil at a face of a control volume

ϕ_f is determined with the values of ϕ and its gradient at the upwind cell as:

$$\phi_f = \phi_{upwind} + \nabla\phi_{r_{upwind}} \cdot d\vec{r} \tag{5}$$

where ϕ_{upwind} is the value of ϕ at the upwind cell, $\phi_{r_{upwind}}$ is the reconstruction gradient at the upwind cell, and $d\vec{r}$ is the vector from the centroid of the upwind cell to the centroid of the face. The convection term can then be written as:

$$\begin{aligned} F_f\phi_f &= max(F_f,0)(\phi_L + \nabla\phi_{r_L} \cdot d\vec{r_L}) \; + min(F_f,0)(\phi_R + \nabla\phi_{r_R} \cdot d\vec{r_R}) \\ &= max(F_f,0)\phi_L + min(F_f,0)\phi_R + max(F_f,0)\nabla\phi_{r_L} \cdot d\vec{r_L} + min(F_f,0)\nabla\phi_{r_R} \cdot d\vec{r_R} \\ &= max(F_f,0)\phi_L + min(F_f,0)\phi_R + S_{conv} \end{aligned} \tag{6}$$

During the solution, the values ϕ_L and ϕ_R are treated implicitly, while the convective source term S_{conv} is treated explicitly.

2.1.2 Diffusion Term

The transport due to diffusion through the face 'f' of a control volume is defined as:

$$G_f = (\mu\nabla\phi \cdot \vec{A})_f \tag{7}$$

For the consistent spatial discretization, the control volume for the evaluation of this diffusive flux is kept the same. In terms of the co-ordinate system(ξ,η), this can be written as:

$$\nabla\phi \cdot \vec{A} = \frac{\partial\phi}{\partial\phi}(A_x\xi_x + A_y\xi_y) + \frac{\partial\phi}{\partial\eta}(A_x\eta_x + A_y\eta_y) \tag{8}$$

where A_x and A_y are the cartesian components of the area vector $\vec{A}$.

For the face 1-2 shown in figure 1,

$$\nabla\phi \cdot \vec{A} = \frac{\phi_R - \phi_L}{ds}\frac{\vec{A}\cdot\vec{A}}{\vec{A}\cdot\vec{e_\xi}} - \frac{\phi_2 - \phi_1}{A}\frac{\vec{A}\cdot\vec{A}}{\vec{A}\cdot\vec{e_\xi}}(\vec{e_\xi}\cdot\vec{e_\eta}) \tag{9}$$

In this equation, the first term is the primary component of diffusion, and the second term is the secondary cross diffusion component. Avoiding the computation of face tangents and nodal values, this second term can be written as the difference between the total diffusion, and the primary diffusion component.

$$G_f = \mu_f\frac{\phi_R - \phi_L}{ds}\frac{\vec{A}\cdot\vec{A}}{\vec{A}\cdot\vec{e_\xi}} + \mu_f\left(\overline{\nabla\phi}\cdot\vec{A} - \overline{\nabla\phi}\cdot\vec{e_\xi}\frac{\vec{A}\cdot\vec{A}}{\vec{A}\cdot\vec{e_\xi}}\right) \tag{10}$$

where $\overline{\nabla\phi}$ is the average of the gradient at the two adjacent cell centers. This can be written as:

$$G_f = D_f(\phi_R - \phi_L) + S_{diff} \tag{11}$$

where,

$$\begin{aligned} D_f &= \mu_f\frac{1}{ds}\frac{\vec{A}\cdot\vec{A}}{\vec{A}\cdot\vec{e_\xi}} \\ S_{diff} &= \mu_f\left(\overline{\nabla\phi}\cdot\vec{A} - \overline{\nabla\phi}\cdot\vec{e_\xi}\frac{\vec{A}\cdot\vec{A}}{\vec{A}\cdot\vec{e_\xi}}\right) \end{aligned}$$

During the solution, the values ϕ_L and ϕ_R are treated implicitly, while the diffusive source term S_{diff} is

treated explicitly.

2.1.3 Gradient Term

Green-Gauss theorem is used to compute the reconstruction gradient term used in the convective term.

$$\nabla\phi_r = \frac{\alpha}{\forall}\sum_f \bar{\phi}_f \vec{A_f} \tag{12}$$

where, $\bar{\phi}_f = \frac{\phi_L+\phi_R}{2}$ is the average of the values of ϕ at the neighboring cells across the face 'f'. The term 'α' is a limiter used to prevent the reconstruction from introducing local extrema.

On the other hand, the cell derivatives of secondary diffusion terms are computed as:

$$\nabla\phi = \frac{1}{\forall}\sum_f \phi_f \vec{A_f} \tag{13}$$

where, $\phi_f = \frac{\phi_{f_L}+\phi_{f_R}}{2}$

The value of ϕ at a face of a cell 'c' is computed as:

$$\phi_{f_c} = \phi_c + \nabla\phi_{r_c} \tag{14}$$

2.1.4 Limiter

Spatial discretization using second order upwind scheme requires the use of limiter to prevent the generation of spurious solutions in the flow. Second order limiter by Venkatakrishnan [6] is used to prevent the formation of local extrema in the flow field, by limiting the reconstruction gradient such that the value at a face of a control face is bounded by it's left and right cell. The details of the limiter and its use can be found in literature [6].

2.2 Temporal Discretization

There are several temporal discretization schemes of different order of accuracy. These time stepping methods dictate the accuracy, efficiency and the robustness of a flow solver especially to the unsteady flow problems. In this section, various time stepping methods are presented.

2.2.1 α Scheme

The time integration of a general variable ϕ over a time period Δt using α scheme is given as:

$$\int_t^{t+\Delta t} \phi dt = \left[\alpha\phi + (1-\alpha)\phi^o\right]\Delta t \tag{15}$$

where, ϕ is the value of the variable at the new time step $t+\Delta t$, and ϕ^o is the value at previous time step t. The value of parameter 'α' is the weightage attributed to the value of variable at new and old time step. In particular, '$\alpha = 0$' gives explicit scheme, '$\alpha = 0.5$' leads to Crank-Nicholson scheme and '$\alpha = 1$' gives fully implicit scheme.

Substitution of convection, diffusion and gradient terms, and then integrating the resulting spatially discretized equation over time Δt , we get

$$\begin{aligned}\frac{\left[(\rho\phi) - (\rho\phi)^o\right]}{\Delta t}\Delta\forall \; + \alpha\sum_f F_f\phi + (1-\alpha)\sum_f (F_f\phi)^o = \\ \alpha\sum_f G_f + (1-\alpha)\sum_f G_f^o \; + \alpha S\Delta\forall + (1-\alpha)S^o\Delta\forall\end{aligned} \tag{16}$$

This can be written in the form of

$$a_P \phi_P = \sum_f a_{nb} \phi_{nb} + b \tag{17}$$

where, ϕ's are the cell center values of ϕ, the a's are the coefficients of ϕ's, 'nb' represents the neigboring terms , and b is the source term containing external sources (if available), terms from previous time step , and the source terms from convection and diffusion terms.

The momentum equations can then be written as

$$a_{u-P} u_P = \sum a_{u-nb} u_{nb} + b_u - \sum_f p_f A_{xf} \tag{18}$$

$$a_{v-P} v_P = \sum a_{v-nb} v_{nb} + b_v - \sum_f p_f A_{yf} \tag{19}$$

where b_u and b_v are the source terms of momentum equations, p is the pressure, and A_{xf} and A_{yf} are the x and y component of face area.

2.2.2 Runge Kutta Schemes

The spatially discretized scalar transport equation over a control volume of volume $\Delta\forall$ surrounding a point P can be written as:

$$\rho \Delta \forall \frac{\partial \phi}{\partial t} = \sum a_{nb} \phi_{nb} - a_P \phi_P + S_c \Delta \forall \tag{20}$$

Similar formulation leads to the following momentum equations:

$$\frac{\partial u}{\partial t} = \frac{\sum a_{u-nb} u_{nb} - a_{u-P} u_P + b_u - \sum_f p_f A_{xf}}{\rho \Delta \forall} = \frac{R_u}{\rho \Delta \forall} = F_u(t, u) \tag{21}$$

$$\frac{\partial v}{\partial t} = \frac{\sum a_{v-nb} v_{nb} - a_{v-P} v_P + b_v - \sum_f p_f A_{yf}}{\rho \Delta \forall} = \frac{R_v}{\rho \Delta \forall} = F_v(t, v) \tag{22}$$

where the coefficents (a's) here are obtained using spatially discretized terms and are dependent on velocity. These are the first-order ordinary differential equation (ODEs) which can be then integrated in time using Runge Kutta methods to get the final discretized equations. The Runge Kutta methods involve a weighted average of the function F evaluated at different stages. The general form of Runge Kutta based time integration for u-momentum equation can be written as:

$$(u_P)_s = (u_P)^n + \Delta t \sum_{l=1}^{s} \alpha_{s,l} F_u\big(t^n + \gamma_l \Delta t, u_l\big) \qquad \text{for } 1 \le s \le S \tag{23}$$

$$\phi^{n+1} = \phi^n + \Delta t \sum_{s=1}^{S} \beta_s F\big(t^n + \gamma_s \Delta t, \phi_s\big) \tag{24}$$

with the constraint

$$\gamma_s = \sum_{l=1}^{S} \alpha_{s,l} \qquad \text{for } 1 \le s \le S \tag{25}$$

where $\Delta t = t^{n+1} - t^n$ is the time step size, ϕ_s is defined as the value of ϕ at the s^{th} stage, and l is a free index used in the summation. The coefficients α, β, and γ determine the specific RK method and are often defined in a Butcher tableau [7] as depicted in Table 1.

Runge Kutta methods can be categorized into explicit and implicit methods. Of the pool of available

Table 1: Form of the Butcher tableau.

γ_1	α_{11}	α_{12}	$\cdots$	α_{1l}	$\cdots$	α_{1S}
γ_2	α_{21}	α_{22}	$\cdots$	α_{2l}	$\cdots$	α_{2S}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
γ_s	α_{s1}	α_{s2}	$\cdots$	α_{sl}	$\cdots$	α_{sS}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
γ_S	α_{S1}	α_{S2}	$\cdots$	α_{Sl}	$\cdots$	α_{SS}
	β_1	β_2	$\cdots$	β_l	$\cdots$	β_S

Runge Kutta methods, RKSIMPLER algorithm [5] is based on four-stage Low Storage Explicit Runge Kutta method, and three stiffly accurate Diagonally Implicit Runge Kutta (DIRK) methods (with one, two and three stages) are explored in IRKSIMPLER algorithm namely IRKB1, IRKB2 and IRKB3 respectively [4]. Details about various Runge Kutta methods and used coefficients can be found in [4][7].

Using this Runge Kutta based time integration scheme, the stage equations for x momentum can be written as:

$$(u_P)_s = (u_P)^n + \Delta t \sum_{l=1}^{s} \alpha_{s,l} F_u\big(t^n + \gamma_l \Delta t, u_l\big) \qquad \text{for } 1 \le s \le S \tag{26}$$

Utilizing the definiton of F_u, the fully discretized x-momentum equation for each stage becomes

$$a'_{u-P}(u_P)_s = \sum (a_{u-nb})_s (u_{nb})_s + b'_u - \sum p_f A_{xf} \tag{27}$$

where,

$$\begin{aligned} a'_{u-P} &= (a_{u-P})_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t} \\ b'_u &= (b_u)_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t} (u_P)^n + \frac{R_s(u)}{\alpha_{s,s}} \\ R_s(u) &= \rho \Delta \forall \sum_{l=1}^{s-1} \alpha_{s,l} F_u\big(t^n + \gamma_l \Delta t, u_l\big) \end{aligned}$$

Similarly, the discretized form of the y-momentum equation is

$$a'_{v-P}(v_P)_s = \sum (a_{v-nb})_s (v_{nb})_s + b'_v - \sum p_f A_{yf} \tag{28}$$

where,

$$\begin{aligned} a'_{v-P} &= (a_{v-P})_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t} \\ b'_v &= (b_v)_s + \frac{\rho \Delta \forall}{\alpha_{s,s} \Delta t} (v_P)^n + \frac{R_s(v)}{\alpha_{s,s}} \\ R_s(v) &= \rho \Delta \forall \sum_{l=1}^{s-1} \alpha_{s,l} F_v\big(t^n + \gamma_l \Delta t, v_l\big) \end{aligned}$$

In the above equations, the pressure source term results from integrating $\frac{\partial p}{\partial x}$ and $\frac{\partial p}{\partial y}$ over the control volume. The resulting discretization involves a sum over the faces 'f' of the control volume surrounding a point 'P'. Here, p_f is the pressure at the centroid of the face. An interpolation procedure is required to find

the pressure at the face, since pressure is available only at the cell centers. The pressure p_f is calculated by averaging the reconstructed face pressure from the two cells neighboring the face.

$$p'_{f_c} = p_c + \nabla p_{r_c} \cdot \vec{dr} \tag{29}$$

where, p'_{f_c} is the value of p at the face 'f' of a cell 'c', and hence

$$p_f = \frac{(p'_{f_L} + p'_{f_R})}{2} \tag{30}$$

2.3 Pressure Velocity Coupling

In this paper, all the primitive flow variables are stored at the center of the control volume. The difficulty arises while defining the grid that is staggered with respect to the existing quadtree grid [8]. This collocated approach of defining the flow variables necessitates the interpolation from the values defined at the cell centers to calculate the mass flux at the faces. The linear interpolation for pressure gradient at a face results in pressure velocity decoupling which is famously known as 'Checkerboard' problem [3], resulting in non-physical solutions. The checkerboard pressure distribution problem associated with this collocated approach is alleviated using Rhie and Chow interpolation [9]. This practice inherently contains a third order pressure gradient term [9] that serves as the tool to couple pressure and velocity fields by expressing the face velocity using the driving pressure difference across that face.

The fully discretized momentum equations for cell 'P' for each stage are

$$a'_{u-P}(u_P)_s = \sum (a_{u-nb})_s (u_{nb})_s + b'_u - \Big(\frac{\partial p}{\partial x}\Delta\forall\Big)_s \tag{31}$$

$$a'_{v-P}(v_P)_s = \sum (a_{v-nb})_s (v_{nb})_s + b'_v - \Big(\frac{\partial p}{\partial y}\Delta\forall\Big)_s \tag{32}$$

These can be written as

$$(u_P)_s = (\hat{u}_P)_s - [\tilde{d_P^u}]_s \Big(\frac{\partial p}{\partial x}\Big)_s \tag{33}$$

$$(v_P)_s = (\hat{v}_P)_s - [\tilde{d_P^v}]_s \Big(\frac{\partial p}{\partial y}\Big)_s \tag{34}$$

where,

$$(\hat{u}_P)_s = \frac{\sum (a_{u-nb})_s (u_{nb})_s + b'_u}{a'_{u-P}} \qquad [\tilde{d_P^u}]_s = \frac{\Delta\forall}{a'_{u-P}}$$

$$(\hat{v}_P)_s = \frac{\sum (a_{v-nb})_s (v_{nb})_s + b'_v}{a'_{v-P}} \qquad [\tilde{d_P^v}]_s = \frac{\Delta\forall}{a'_{v-P}}$$

In vector form,

$$(\vec{V_P})_s = (\vec{\hat{V}_P})_s - [\tilde{d_P}]_s (\nabla p_P)_s \tag{35}$$

The formulation of pseudo velocity at the face 'f' gives

$$(\vec{\hat{V}_f})_s = (\vec{V_f})_s + [\tilde{d_f}]_s (\nabla p_f)_s \tag{36}$$

Now, the pseudo velocity at the face 'f' can be approximated as the average of the cell center values.

$$(\vec{\hat{V}_f})_s = \frac{1}{2}\Big[(\vec{\hat{V}_L})_s + (\vec{\hat{V}_R})_s\Big] \tag{37}$$

$$= \frac{1}{2}\Big[(\vec{V_L})_s + (\vec{V_R})_s + [\tilde{d_L}]_s (\nabla p_L)_s + [\tilde{d_R}]_s (\nabla p_R)_s\Big] \tag{38}$$

The mass flow rate at the face 'f' is then given by

$$(f_f)_s = \left(\rho \vec{V_f} \cdot \vec{A}\right)_s \tag{39}$$

$$= \frac{1}{2}\rho\Big((\vec{V_L})_s + (\vec{V_R})_s\Big) \cdot \vec{A} + \frac{1}{2}\rho\vec{A} \cdot \Big([\tilde{d_L}]_s(\nabla p_L)_s + [\tilde{d_R}]_s(\nabla p_R)_s\Big) - \rho\vec{A} \cdot [\tilde{d_f}]_s(\nabla p_f)_s \tag{40}$$

Using Mathur's Assumption [10],

$$[\tilde{d_f}]_s = [\tilde{d_L}]_s = [\tilde{d_R}]_s = [\tilde{d}]_s = \frac{\Delta\forall_L + \Delta\forall_R}{\bar{a}_{pL} + \bar{a}_{pR}} \tag{41}$$

where,

$$\bar{a}_{pL} = \frac{1}{2}\Big(a'_{(u-P)L} + a'_{(v-P)L}\Big) \tag{42}$$

$$\bar{a}_{pR} = \frac{1}{2}\Big(a'_{(u-P)R} + a'_{(v-P)R}\Big) \tag{43}$$

This results in the expression of mass flow rate as

$$(f_f)_s = \frac{1}{2}\rho\Big((\vec{V_L})_s + (\vec{V_R})_s\Big) \cdot \vec{A} + \frac{1}{2}\rho[\tilde{d}]_s\vec{A} \cdot \Big((\nabla p_L)_s + (\nabla p_R)_s\Big) - \rho\vec{A} \cdot [\tilde{d}]_s\nabla p_f \tag{44}$$

The second term on RHS is now approximated using volume weighting of the left and right cell center gradient as

$$\frac{1}{2}\rho[\tilde{d}]_s\vec{A} \cdot \Big((\nabla p_L)_s + (\nabla p_R)_s\Big) = \rho[\tilde{d}]_s\vec{A} \cdot \left(\frac{\nabla p_L\Delta\forall_L + \nabla p_R\Delta\forall_R}{\Delta\forall_L + \Delta\forall_R}\right)_s \tag{45}$$

$$= \rho[\tilde{d}]_s\vec{A} \cdot \bar{\nabla}p \tag{46}$$

The third term on the RHS is also approximated as

$$\rho\vec{A} \cdot [\tilde{d}]_s\nabla p_f = \rho[\tilde{d}]_s\left[\frac{p_R - p_L}{ds}\frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \vec{e_\xi}} + \bar{\nabla}p \cdot \vec{A} - \bar{\nabla}p \cdot \vec{e_\xi}\frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \vec{e_\xi}}\right]_s \tag{47}$$

Substituting and rearranging these terms, the mass flow rate can now be written as

$$(f_f)_s = \frac{1}{2}\rho\Big((\vec{V_L})_s + (\vec{V_R})_s\Big) \cdot \vec{A} - (df)_s(p_R - p_L)_s + \rho\left(\frac{\Delta\forall_L + \Delta\forall_R}{\bar{a}_{pL} + \bar{a}_{pR}}\right)_s \bar{\nabla}p \cdot \vec{e_\xi}\frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \vec{e_\xi}} \tag{48}$$

where,

$$df_s = \frac{\rho}{ds}\left(\frac{\Delta\forall_L + \Delta\forall_R}{\bar{a}_{pL} + \bar{a}_{pR}}\right)\frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \vec{e_\xi}} \tag{49}$$

The mass flow rate at a face 'f' can be be rewritten as

$$(f_f)_s = (\hat{f}_f)_s - (df)_s(p_R - p_L)_s \tag{50}$$

where,

$$(\hat{f}_f)_s = \frac{1}{2}\rho\Big((\vec{V_L})_s + (\vec{V_R})_s\Big) \cdot \vec{A} + \rho\left(\frac{\Delta\forall_L + \Delta\forall_R}{\bar{a}_{pL} + \bar{a}_{pR}}\right)_s \bar{\nabla}p \cdot \vec{e_\xi}\frac{\vec{A} \cdot \vec{A}}{\vec{A} \cdot \vec{e_\xi}} \tag{51}$$

The integrated and the discretized continuity equation can also be written as

$$\sum_f f_f = 0 \tag{52}$$

The above equation of mass flow rate when substituted in continuity equation gives

$$(a_{p-P})_s(p_P)_s = \sum_{nb}(a_{p-nb})_s(p_{nb})_s + (b_p)_s \tag{53}$$

where,

$$(b_p)_s = -\sum_f(\hat{f}_f)_s \tag{54}$$

The formulation procedure for pressure equation that are given in this section are similar for the α based scheme. The only changes involved are that the momentum coefficients are given by the equations 18 and 19, and the subscript 's' representing stage can be dropped.

2.4 Pressure Correction Equation

Explicitly based RKSIMPLER algorithm solves the pressure equation once at each time step [5], and uses that solved pressure explicitly for the solution of momentum equations. Implicitly based IRKSIMPLER algorithm formulates the pressure equation at each stage by utilizing the discretized momentum stage equations into continuity equation.The Runge Kutta based schemes don't require pressure correction equation at all [4][5]. On the other hand, in 'α' scheme based SIMPLER algorithm, the pressure correction equation is required to correct an incorrect or guessed pressure field so that correct pressure field can be obtained which will ultimately satisfy the mass and momentum conservation at each time step [3]. The basic underlying principle for the formulation of this equation follows Patankar [3].

The mass flow rate at a face with the guessed velocity field can be written as

$$f_f^* = \frac{1}{2}\rho\Big((\vec{V}_L^*) + (\vec{V}_R^*)\Big)\cdot\vec{A} - (df)(p_R^* - p_L^*) + \rho\left(\frac{\Delta\forall_L + \Delta\forall_R}{\bar{a}_{pL} + \bar{a}_{pR}}\right)\bar{\nabla}p\cdot\vec{e_\xi}\frac{\vec{A}\cdot\vec{A}}{\vec{A}\cdot\vec{e_\xi}} \tag{55}$$

If f_f' is the mass flow rate correction defined as

$$f_f' = -(df)(p_R' - p_L') \tag{56}$$

The discretized continuity equation can be written as

$$\sum_f(f_f^* + f_f') = 0 \tag{57}$$

Combining equations 55 and 57 give the following pressure correction equation

$$(a_{p'-P})p_P' = \sum_{nb} a_{p'-nb}p_{nb}' + b_{p'} \tag{58}$$

where, $b_{p'} = -\sum_f f_f^*$ is the total mass flow rate into the cell.

The cell centered velocities are then corrected as,

$$u_P = u_P^* - \frac{\sum_f p'_f A_{xf}}{a_P^u} \tag{59}$$

$$v_P = v_P^* - \frac{\sum_f p'_f A_{yf}}{a_P^v} \tag{60}$$

3 Results

3.1 Lid Driven Cavity

Although the development of these Runge Kutta based schemes on quadtree grid are for simulating unsteady incompressible flow efficiently, the two dimensional steady lid driven cavity problem is an excellent test case for evaluating the newly developed schemes. This is done by simulating the flow inside the lid driven cavity until the steady state has been reached. This lid-driven cavity, shown in figure 2, with the dimension $L \times L$ has a constant speed of lid (U_{lid}) at the top, and three other boundaries represented by the no-slip viscous walls. The developed Runge Kutta based schemes are compared against the Crank-Nicholson(CN) based SIMPLER algorithm (with the number of subiterations fixed at 30) at a Reynolds number (defined by $Re = \rho U_{lid} L/\mu$) of 100 and 1000. Simulations are run with a constant time step, and the steady state results are compared using the u-velocity profile along the vertical centerline, and v-velocity along the horizontal centerline.

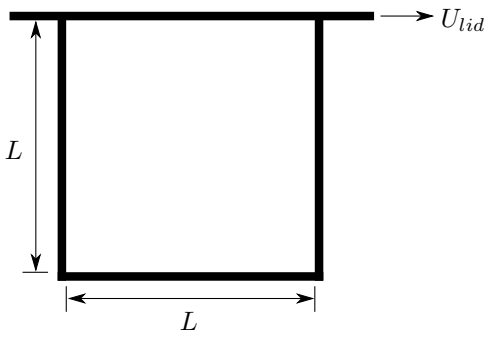


Figure 2: Lid Driven Cavity

In figures (3)-(6), it can be seen that the flow solution results for both explicit and implicit Runge Kutta based schemes are in perfect agreement with Crank Nicholson based SIMPLER algorithm for both cases of Reynolds number, since the same spatial discretization is used for all these algorithms. These results verify that the Runge Kutta based algorithms can accurately simulate the steady flows [4]. The implicit Runge Kutta schemes (IRKSIMPLER) allow for larger time steps when compared to their explicit counterpart (RKSIMPLER). RKSIMPLER algorithm has the time step restriction based on the stability requirement according to CFL criterion for explicit method.

3.2 Flow over a Flat Plate Normal to the Flow

The thin flat plate placed normal to the flow direction is simulated for the laminar flow at high Reynolds number of 17800 [5]. A schematic of the flat plate problem is shown in figure 7. At this Reynolds number, the streamlines near the flat plate periodically breaks off alternately at the edges of the flat plate resulting in an unsteady flow. These streamlines carry vorticity of large magnitude away from the plate, and this periodic vortex shedding from the flat plate results in an oscillating lift and drag force on the flat plate.

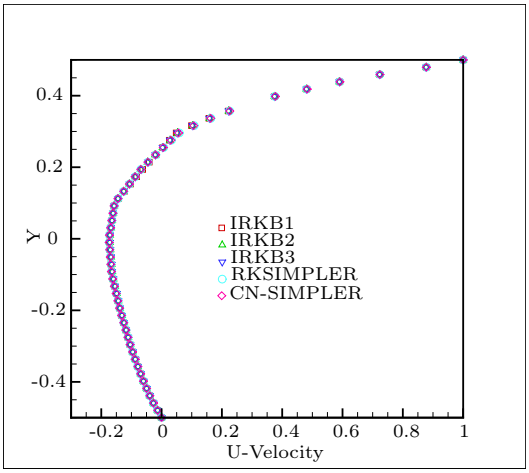


Figure 3: U-velocity profile along a vertical centerline (Re=100)

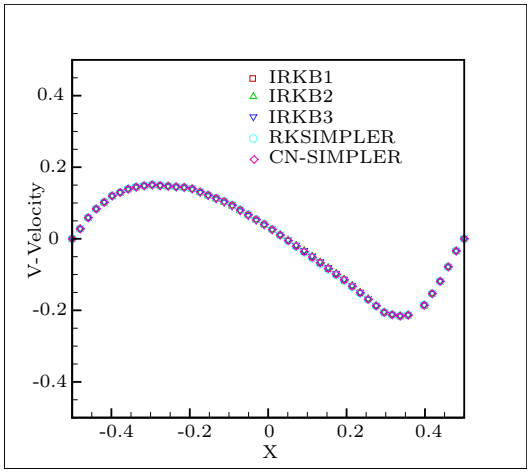


Figure 4: V-velocity profile along a horizontal centerline (Re=100)

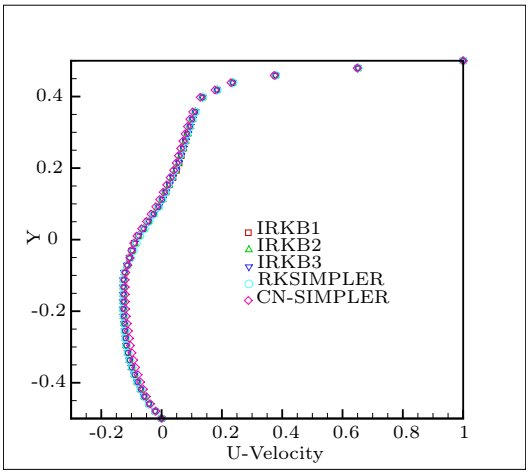


Figure 5: U-velocity profile along a vertical centerline (Re=1000)

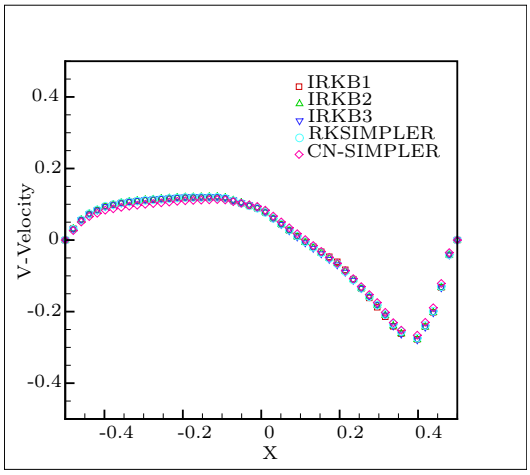


Figure 6: V-velocity profile along a horizontal centerline (Re=1000)

Strouhal number($Sr = fL/U$) is used as a measure of unsteady vortex shedding where f is the frequency of shedding. Simulation is run for the non-dimensional time of 400 with an uniform flow at the inlet boundary while the velocity is adjusted at the outflow boundary using the value of neighboring cell for the overall mass conservation. The top and bottom boundaries are set to freestream conditions.

The time history of drag coefficient for all the algorithms is shown in figure 8. The drag coefficient begin to oscillate after some time, and achieve a constant oscillatory pattern known as unsteady convergence. Different algorithms (and with different time steps) reach the unsteady convergence at different simulation times [4]. Table 2 shows the comparison of various schemes based on maximum allowable time step, mean drag coefficient, Strouhal number, CPU time required, and speed up attained with maximum allowable time step.

3.3 Flow over a Circular Cylinder

The flow over a 2D circular cylinder is used as a case to validate the developed formulations for quadtree grid with body fitted cells near the cylinder. A schematic of the circular cylinder problem is shown in figure 9. Figure 10a and 10b show the quadtree grid and the body fitted type cells near the cylinder respectively. The generated body fitted quadtree grid consists of 6.1120e+03 nodes, 1.2213e+04 faces and 6.1010e+03 cells.

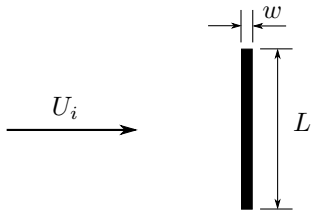


Figure 7: Thin Flat plate normal to the flow

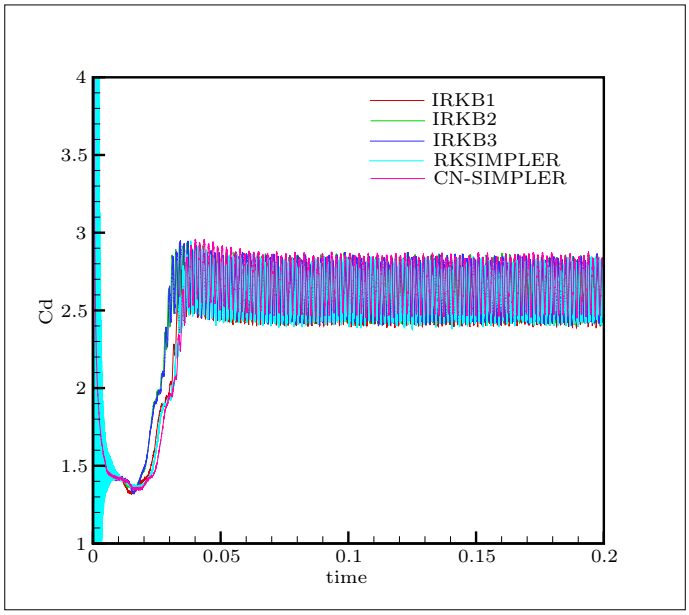


Figure 8: Time history of drag coefficient for flat plate problem

Table 2: Comparison of Flat plate results for various algorithms

Algorithm	Maximum allowable time step	Mean drag coefficient C_d	Strouhal Number Sr	CPU time required (in minutes)	Speed up (relative to CN-SIMPLER)
CN-SIMPLER	2.00×10^{-5}	2.5145	0.146	19.71	1.00
IRKB1	5.00×10^{-6}	2.4661	0.148	14.25	1.38
IRKB2	1.00×10^{-5}	2.5097	0.148	13.41	1.47
IRKB3	5.00×10^{-6}	2.5092	0.148	28.26	0.69
RKSIMPLER	1.05×10^{-6}	2.4676	0.150	20.26	0.97

To validate against the established literature [11][12][13], the flow over a circular cylinder of unit diameter is simulated for a Reynolds number of 300 and for the non-dimensional time of 200 since accurate measurement data are available at this Reynolds number. The boundary conditions are kept similar to the case of flat plate. As in the case of flat plate, simulation is run with impulsive start in time, and with the march in time the vortices begin to shed from the top and bottom of the cylinder, and ultimately reach the unsteady convergence.

The streamlines for the flow over a cylinder is shown in figure 10c. The time history of drag coefficient for all the formulated schemes with the maximum allowable time step is shown in figure 11. The presence of dominant periodicity because of regular vortex shedding from the top and bottom of cylinder is expressed

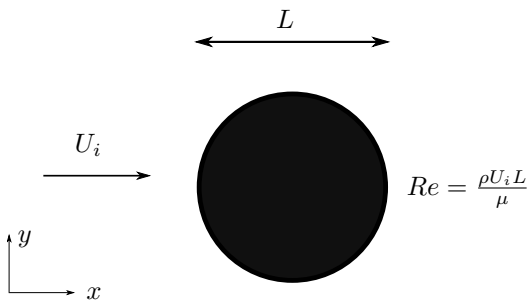
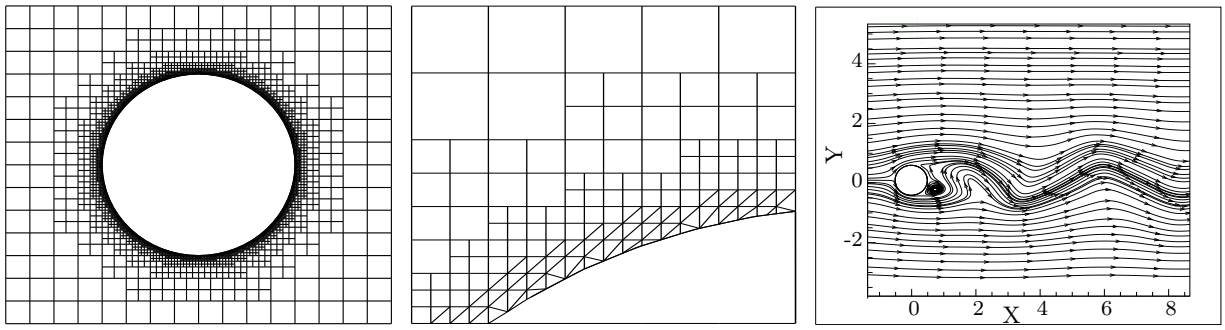


Figure 9: Schematic of a circular cylinder problem

in the form of Strouhal number. Table 3 presents the comparison of various schemes based on maximum allowable time step, mean drag coefficient, Strouhal number, CPU time required, and speed up attained for the cylinder problem.



(a) Quadtree grid (b) Body fitted type cells (c) Streamlines

Figure 10: Cartesian unstructured grid with body fitting (a) and (b), and Streamlines showing flowfield around a circular cylinder

Table 3: Comparison of results of various algorithms for a circular cylinder problem

Algorithm	Maximum allowable time step	Mean drag coefficient C_d	Strouhal Number Sr	CPU time required (in minutes)	Speed up (relative to CN-SIMPLER)
CN-SIMPLER	6.67×10^{-2}	1.198	0.180	9.23	1.00
IRKB1	2.00×10^{-2}	1.197	0.192	6.50	1.42
IRKB2	4.00×10^{-2}	1.208	0.192	6.11	1.51
IRKB3	2.00×10^{-2}	1.211	0.200	18.63	0.49
RKSIMPLER	1.67×10^{-3}	1.224	0.192	20.98	0.44

For a large range of Reynolds number, the experimental results show that the Strouhal number remains close to 0.2. The experimental values from Williamson [13] for Strouhal number and mean drag coefficient are 0.203 and 1.22 respectively. To study the behavior of these formulations, finer grid (1.0803e+04 nodes, 2.1527e+04 faces and 1.0724e+03 cells) is used. As the grid is refined, the maximum allowable time step for all the schemes decreases. The average drag coefficient and Strouhal number on this refined grid for various formulations are given in table 4.

The 2D computations from our formulation over predicts the value of mean drag coefficient slightly. These have been attributed to the development of three dimensional features beyond $Re \sim 200$ and the higher level of Reynolds stresses behind the bluff body [14].

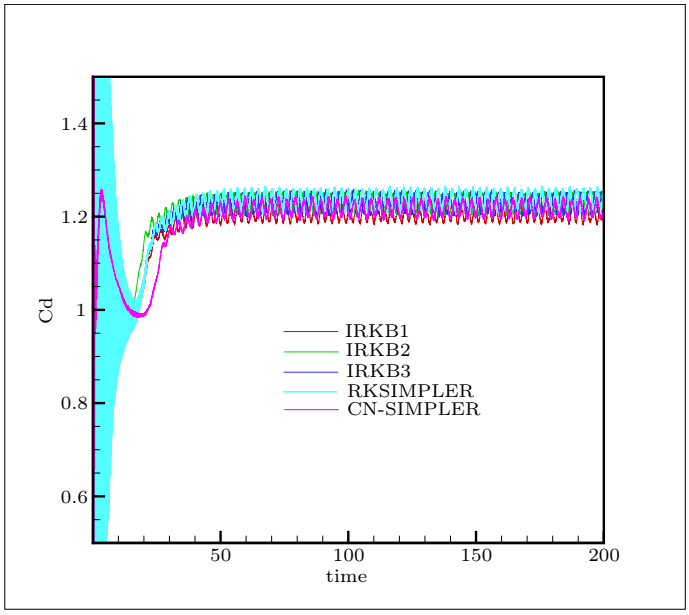


Figure 11: Time history of drag coefficient for circular cylinder problem

Table 4: Mean drag coefficient and Strouhal number for various formulations

Algorithm	Mean drag coefficient C_d	Strouhal Number Sr
CN-SIMPLER	1.281	0.180
IRKB1	1.280	0.200
IRKB2	1.263	0.200
IRKB3	1.282	0.200
RKSIMPLER	1.288	0.200

3.4 Flow over a sphere

The three dimensional unsteady Navier Stokes equations are solved on an octree grid with the body fitted cells around a sphere. Laminar flow simulation is carried out at a Reynolds number of 300 and for the non-dimensional time of 150, with the flow adapted to vorticity. The uniform inflow and mass corrected velocity outflow boundary conditions are used. All other boundary conditions are set to freestream conditions.

Starting from the same initial grid and fixed level of adaption refinement, the implicit IRKSIMPLER algorithms are tested for the accurate capture of the flow with maximum possible time step. The explicit solver RKSIMPLER required considerable small time step for convergence thereby increasing the runtime which was estimated to be significantly high compared to the other solvers. Therefore, RKSIMPLER algorithm is not tested for this case. The final grid information for these adaptive solvers for the setup case are given in table 5.

Table 5: Final Grid Information for the adaptive solvers

Algorithm	Final Grid
IRKB1	209774 nodes, 886311 faces, 341642 cells
IRKB2	257947 nodes, 1021728 faces, 385231 cells
IRKB3	223744 nodes, 925896 faces, 354445 cells

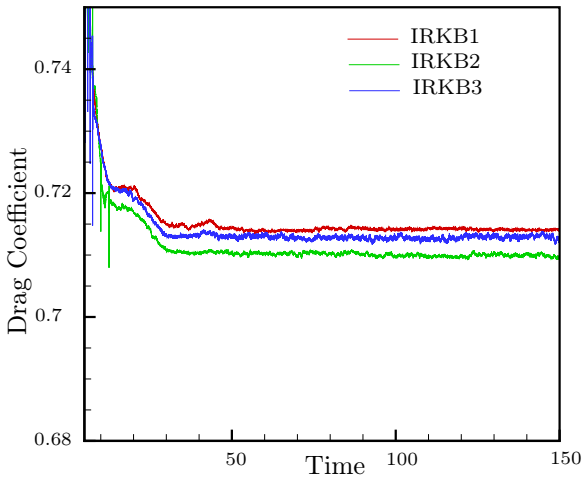


Figure 12: Time history of drag coefficient for a sphere

Figure 12 shows the time history of the drag coefficient for the simulated cases. Although there are some oscillations, these solvers are unable to capture the unsteady vortex shedding, and the drag coefficient looks almost steady. The mean coefficient of drag is about 0.715 for all the solvers which are higher than the previous established results of about 0.655 [15][16][17], and 0.671 [18].

With a view to capture the unsteady vortex shedding, maintaining runtime constraint, these IRKSIMPLER solvers along with CNSIMPLER solver are tested to adapt to finer grid than previous case. The adapted final grid information for these solvers are given in table 6.

Table 6: Final Grid Information for the adaptive solvers for second case

Algorithm	Final Grid
IRKB1	381329 nodes, 1960971 faces, 493339 cells
IRKB2	389764 nodes, 1384467 faces, 500850 cells
IRKB3	412588 nodes, 1443729 faces, 519141 cells
CN-SIMPLER	387151 nodes, 1400568 faces, 5101460 cells

Figure 13 shows the time history of drag coefficient for this case. The mean drag coefficient obtained for all the IRKSIMPLER cases are near 0.70, and 0.671 for CN-SIMPLER case, which is still slightly higher than the published results. For the finer grids, the drag coefficient approaches accurate results, but these solvers still failed to capture the unsteady vortex shedding, and resulted in almost steady solution. Refining the grid further might capture this vortex shedding, and result in accurate mean drag coefficient. However, it seems to increase the runtime to a great extent.

4 Conclusions

A new Runge Kutta (RK) based SIMPLER algorithm on an adaptively refined quadtree grid is developed for incompressible flow. These RK based schemes solve the governing conservation equations by effectively coupling the pressure and velocity, and without the need of pressure correction equation. Also, the collocated approach in these formulations don't require the use of relaxation. These formulations are shown to simulate the steady and unsteady problems both accurately and efficiently, thereby making these formulations robust. The nature of quadtree type of grids provides better grid adaption, and accurate capture of flow characteristics.

Results for the laminar flow 2D and 3D simulations for a wide range of Reynolds numbers are presented. The candidate cases of both internal and external type of flow are taken, and the developed formulations are tested against each other with a view to compare their efficiency and robustness. The IRKSIMPLER solvers

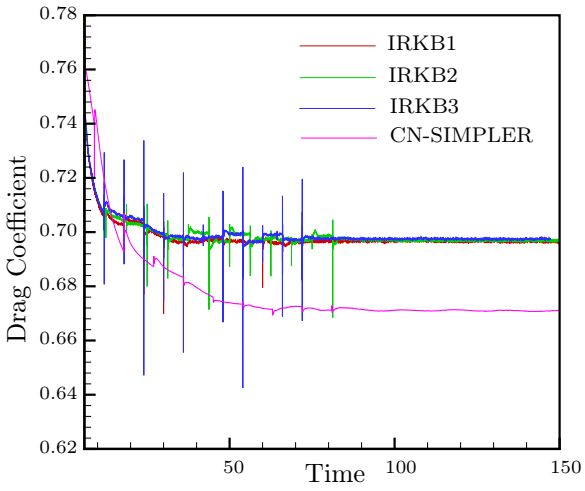


Figure 13: Time history of drag coefficient for a sphere on finer grid

are more stable than the explicit RKSIMPLER solver but the computational time per time step is higher. The explicit nature of RKSIMPLER algorithm requires satisfaction of the CFL criterion, and hence require small time step for a stable solution. Of all the developed schemes, IRKB2 solver seems more efficient and robust in terms of its accuracy and requires less computation time.

Acknowledgments

This work was supported by Georgia Tech's Vertical Lift Research Center of Excellence (VLRCOE). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of VLRCOE. The authors would also like to express gratitude to William Polzin for providing technical support and knowledge in this research.

References

[1] S.S. Ochs. *An adaptively-refined quadtree grid method for incompressible flows*. PhD thesis, Iowa State University, Ames, IA, 1998.

[2] D. De Zeeuw and K.G. Powell. An Adaptively Refined Cartesian Mesh Solver for the Euler Equations. *Journal of Computational Physics*, 104:56–58, 1994.

[3] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corp, New York, 1980.

[4] M.V. Fischels and R.G. Rajagopalan. A Family of Runge-Kutta Based Algorithms for Unsteady Incompressible Flows. *AIAA Journal*, 55(8):2630–2644, 2017.

[5] R.G. Rajagopalan and A.D. Lestari. RK-SIMPLER: An Explicit Time Accurate Algorithm for Incompressible Flows . *AIAA Journal*, 54(2):616–624, 2016.

[6] V. Venkatakrishnan. On the Accuracy of Limiters and Convergence to Steady State Solutions. *AIAA paper*, 1993.

[7] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2003.

[8] M.Peric, R. Kessler, and G. Scheuerer. Comparison of Finite-Volume Numerical Methods with Staggered and Colocated Grids . *Computers and Fluids*, 16:389–403, 1988.

[9] C.M. Rhie and W.L. Chow. Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation. *AIAA Journal*, 21:1525–1532, 1983.

[10] S.R. Mathur and J.Y Murthy. A pressure Based Method for Unstructured Meshes. *Numerical Heat Transfer,Part B: Fundamentals*, 31:195–215, 1997.

[11] B.N. Rajani, A. Kandasamy, and S. Majumdar. Numerical simulation of laminar flow past a circular cylinder. *Applied Mathematical Modeling*, 33:1228–1247, 2009.

[12] A. Kianifar and E.Y. Rad. Numerical Simulation of Unsteady Flow with Vortex Shedding Around Circular Cylinder. *Latest Trends on Theoretical and Applied Mechanics*, 2010.

[13] C.H.K. Williamson. Vortex dynamics in the cylinder wake. *Annual Review of Fluid Mechanics*, 28:477–539, 1996.

[14] R. Mittal and S. Balachandar. On the inclusion of three-dimensional effects in simulation of two-dimensional bluff-body wake flows. *ASME Fluids Engineering Divison Summer Meeting*, 1997.

[15] B. Fornberg. Steady viscous flow past a sphere at high Reynolds numbers. *Journal of Fluid Mechanics*, 190:471–489, 1988.

[16] G.S Constantinescu and K.D. Squires. LES and DES Investigations of Turbulent Flow over a Sphere. *AIAA Journal*, 2000.

[17] T.A. Johnson and V.C. Patel. Flow past a Sphere up to a Reynolds Number of 300. *Journal of Fluid Mechanics*, 378:19–70, 1999.

[18] A.G. Tomboulides and S.A. Orszag. Numerical investigation of transitional and weak turbulent flow past a sphere. *Journal of Fluid Mechanics*, 416:45–73, 2000.