Ninth International Conference on Computational Fluid Dynamics (ICCFD9), Istanbul, Turkey, July 11-15, 2016

ICCFD9-2016-125

Runge-Kutta Based Algorithms for More Efficient Simulation of Unsteady Incompressible Flows

Matthew V. Fischels[1], R. G. Rajagopalan[1]

[1] Department of Aerospace Engineering, Iowa State University, Ames, Iowa, 50011

Abstract: A new algorithm, IRK-SIMPLER, is developed based on implicit Runge-Kutta (RK) methods. IRK-SIMPLER uses diagonally implicit Runge-Kutta (DIRK) stages to integrate the momentum equations in time and solves a pressure equation each DIRK stage to satisfy continuity. An iterative loop is used to solve the momentum and pressure equations without the need for relaxation. IRK-SIMPLER is tested on an unsteady 3D simulation and is found to result in speedups of up to 62 times compared to the traditional SIMPLER algorithm and up to 56 times over the explicit RK-SIMPLER algorithm. To further improve the efficiency of the IRK-SIMPLER algorithm methods such as approximate factorization and Multigrid methods are exploited. When approximate factorization is used to solve the implicit momentum equations in the IRK-SIMPLER algorithm, the runtime required reduces by up to 48% compared to IRK-SIMPLER without the approximate factorization. Multigrid methods, when applied, to the IRK-SIMPLER algorithm, the solution reaches the converged state each time step with up to 50% less runtime. Multigrid methods are most advantageous when higher accuracy (i.e. lower residual values) are desired at each time step such as when an accurate transient solution is sought.

Keywords: Numerical Algorithms, Runge-Kutta Methods, Unsteady Simulation, Implicit Methods, Multigrid.

1 Introduction

As the desire for simulating more complex engineering flows increases, developers often turn to hardware acceleration to reduce the time required to solve. Great advances have been made and are continuing to be made in hardware acceleration techniques, but improvements in the algorithms and numerical methods can also accelerate the simulation independent of the hardware used. This paper examines a new algorithm, IRK-SIMPLER, for simulating unsteady incompressible flows. In addition, applying proven efficient numerical methods such as approximate factorization and Multigrid to the IRK-SIMPLER algorithm is shown to further reduce the runtime required to accurately simulate unsteady incompressible flows.

1.1 Unsteady Incompressible Flow Algorithms

In solving unsteady incompressible fluid flow problems, traditional iterative algorithms can become numerically inefficient as problems become increasingly complex. Pressure is not a variable in the continuity equation and leads to the difficulty of not having an explicit equation for pressure. Three approaches have been often used to remedy this problem: (a) artificial compressibility methods[1], (b) pressure-based methods[2], and (c) vorticity-streamfunction approach[3].

The pressure-based methods derive an equation for pressure from the discretized continuity and momentum equations. Most pressure-based methods commonly used today have their roots in the SIMPLE and SIMPLER algorithms[2]. The SIMPLER algorithm is based on an exact pressure equation and an approximate pressure correction equation that adjusts velocity to satisfy continuity. The SIMPLER algorithm requires relaxation and sub-iterations at each time step to converge.

In an attempt to improve the efficiency of pressure-based methods and increase the convergence rate, many variants of the pressure-based algorithms have been developed including PISO[4], CLEAR[5], IDEAL[6] and RK-SIMPLER[7]. Among these algorithms, CLEAR and IDEAL use relaxation and sub-iterations to converge at each time step, while PISO and RK-SIMPLER are able to update the solution in time without requiring sub-iterations within a time step.

RK methods are commonly used in solving Euler equations[8, 9] and the compressible Navier-Stokes equations[10, 11, 12] because the spatially discretized conservation equations form coupled ordinary differential equations that are simply integrated in time with RK methods. However, for incompressible flow, there does not exist an explicit equation for pressure. The spatially discretized continuity and momentum equations form differential algebraic equations with an index of 2 as defined in Hairer[13]. Solving the incompressible Navier-Stokes equations with RK methods is more intricate than the compressible Navier-Stokes equations. Several incompressible algorithms that use RK methods have been developed [14, 15, 16, 17, 18, 19, 20, 21]. Sanderse[21] developed an algorithm that uses explicit RK methods and a projection method to integrate in time while achieving higher temporal order of accuracy. Sanderse has shown that to achieve the higher temporal order of accuracy of the RK method, continuity must be solved at each RK stage. The primary interest so far in RK based methods is on the accuracy of the algorithm and not on the the efficiency of the algorithms, which is often not presented or discussed.

The RK-SIMPLER[7] algorithm is a pressure-based method developed to solve the incompressible Navier-Stokes equations using explicit RK methods. The RK-SIMPLER algorithm forms an equation for pressure by combining the discretized continuity and momentum equations in exactly the same way as the original SIMPLER algorithm. The solution of this equation is then used as an explicit source term in the momentum equations. This results in the momentum equations becoming ordinary differential equations, which are integrated in time with explicit RK stages. However, the RK-SIMPLER algorithm often requires small time steps due to the explicit nature of the algorithm[7].

To relax the time step restrictions, implicit RK stages can be used in place of the explicit stage equations. An implicit RK algorithm developed in Fischels[22] follows the original RK-SIMPLER algorithm closely with the exception of integrating the momentum equation with implicit RK stages This algorithm is shown to allow larger time steps and reduce the runtime for model problems when compared to the original explicit RK-SIMPLER algorithm. A new algorithm which improves upon this implicit RK algorithm called IRK-SIMPLER is developed in this paper and will center on reformulating the pressure equation for each stage based on the RK stage equations for momentum, thereby satisfying continuity better at each stage.

1.2 Numerical Methods

The numerical methods used to solve the equations in any given algorithm can make a large difference in the runtime required. A majority of the computations required in implicit algorithms is in solving the systems of equations. Traditional methods for solving the linear systems of equations are Gauss-Seidel and other simple point-by-point methods. In [7, 22] a line-by-line Tri-Diagonal Matrix Algorithm (TDMA)[2] is used with symmetric Gauss-Geidel marching of grid lines in all dimensions. This method is found to be more efficient than point-by-point methods, but is suitable only for structured grid systems.

To reduce the computations required to solve the momentum equations approximate factorization[23] can be used. Approximate factorization allows a 3D system of equations to be solved as three sets of 1D systems of equations along grids lines in the x, y, and z directions. Solving several relatively small 1D systems of equations is generally much faster than solving a larger 3D system of equations. Approximate factorization, as its name suggests, does require an approximation which introduces an error of order δt^2. This approximate factorization is integrated into the IRK-SIMPLER algorithm to solve the momentum equations at the beginning of each RK stage. Given enough iterative loops in the IRK-SIMPLER algorithm, any errors introduced by the approximate factorization will be removed because the exact momentum equations are solved within the iterative loop.

Multigrid methods[24, 25] have been developed and found to accelerate the solution of systems of equations. When solving a system of equations on one fine grid with traditional solution methods, the errors in the solution with high frequency (wavelengths proportional to the grid spacing) are removed very quickly but errors that have low frequency (with wavelengths much larger than grid spacing) take many iterations to remove. Multigrid methods dampen out all wavelengths of error at the same rate by iterating on successively

coarser girds. With multigrid methods the system is solved for a few iterations on the finest grid, the error is restricted to coarser grids where an equivalent system of equations is solved for a few iterations, then the errors are prolonged back up to the finer grids where the fine grid variables are corrected and solved for a few more iterations to remove any errors introduced in prolongation. These multigrid methods are efficient at solving systems of equations, especially large systems of equations which would otherwise require many iterations on a single grid.

A multigrid algorithm for incompressible flows has been developed by Zori[26] called FAS-SIMPLER which solves the SIMPLER algorithm on each grid level while restricting and prolonging the velocity and pressure. This multigrid algorithm resulted in large reduction in time. In this paper, a multigrid version of the unsteady IRK-SIMPLER algorithm is introduced following a similar procedure to Zori. The traditional V-cycle multigrid with fixed iterations on each grid level is examined as well as the cycle-C multigrid method of Brandt[24] which automatically determines when to restrict to a coarser grid and when to prolong to a finer grid.

2 IRK-SIMPLER Algorithm

2.1 Theory

The governing equations for fluid flow are the Navier-Stokes equations and for incompressible flows in three-dimensional Cartesian coordinates can be written as

$$\frac{\partial(\rho u)}{\partial t}+\frac{\partial(\rho uu)}{\partial x}+\frac{\partial(\rho vu)}{\partial y}+\frac{\partial(\rho wu)}{\partial z}=-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right)+\frac{\partial}{\partial z}\left(\mu\frac{\partial u}{\partial z}\right)+S_u \tag{1}$$

$$\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho uv)}{\partial x}+\frac{\partial(\rho vv)}{\partial y}+\frac{\partial(\rho wv)}{\partial z}=-\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial z}\left(\mu\frac{\partial v}{\partial z}\right)+S_v \tag{2}$$

$$\frac{\partial(\rho w)}{\partial t}+\frac{\partial(\rho uw)}{\partial x}+\frac{\partial(\rho vw)}{\partial y}+\frac{\partial(\rho ww)}{\partial z}=-\frac{\partial p}{\partial z}+\frac{\partial}{\partial x}\left(\mu\frac{\partial w}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial w}{\partial y}\right)+\frac{\partial}{\partial z}\left(\mu\frac{\partial w}{\partial z}\right)+S_w \tag{3}$$

$$\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}+\frac{\partial(\rho w)}{\partial z}=0 \tag{4}$$

The velocity components in (x, y, z) are (u, v, w), ρ is the fluid density, μ is the fluid dynamic viscosity, p is the pressure, and (S_u, S_v S_w) are momentum sources. The notation here follows closely to that found in Rajagopalan[7]. The momentum equations (Equations 1-3) can be spatially discretized to form the following semi-discrete equations.

$$\frac{du}{dt}=\frac{R_u}{\rho\Delta x\Delta y\Delta z} \tag{5}$$

$$\frac{dv}{dt}=\frac{R_v}{\rho\Delta x\Delta y\Delta z} \tag{6}$$

$$\frac{dw}{dt}=\frac{R_w}{\rho\Delta x\Delta y\Delta z} \tag{7}$$

where R_u, R_v, and R_w contain all spatially discretized terms (convection, diffusion, pressure, and source terms) and can be expressed as

$$R_u=\sum a_{u-nb}u_{nb}-a_{u-P}u_P+b_u-\Delta y\Delta z(p_e-p_w) \tag{8}$$

$$R_v=\sum a_{v-nb}v_{nb}-a_{v-P}v_P+b_v-\Delta x\Delta z(p_n-p_s) \tag{9}$$

$$R_w=\sum a_{w-nb}w_{nb}-a_{w-P}w_P+b_w-\Delta x\Delta y(p_f-p_b) \tag{10}$$

The a values are coefficients and the b values are source terms. The summation is over neighboring (referred to as nb) grid points which for a seven point stencil include the east, west, north, south, front, and back grid points (E, W, N, S, F, and B), and P refers to the primary grid point of a cell. The coefficients of

these equations are determined by the spatial discretization scheme used. This paper follows a finite volume discretization scheme developed in Patankar[2] and used in Rajagopalan[7].

The continuity equation (Equation 4) can be integrated over the three-dimensional Cartesian control volume surrounding the primary grid point (P) to yield the following algebraic equation.

$$(\rho u \Delta y)_e - (\rho u \Delta y)_w + (\rho v \Delta x)_n - (\rho v \Delta x)_s + (\rho w \Delta z)_f - (\rho w \Delta z)_b = 0 \tag{11}$$

The six control volume faces are east, west, north, south, front, and back faces (e, w, n, s, f, and b). The semi-discrete momentum equations (Equations 5-7) and discrete continuity equation (Equation 11) form differential algebraic equations (Hairer[13]) and have the following form.

$$\frac{d\vec{V}}{dt} = f(\vec{V}, p) \tag{12}$$

$$0 = g(\vec{V}) \tag{13}$$

The spatially discretized momentum terms are contained in f, and the discrete continuity terms are contained in g. This form of the equations does not lead to a straightforward solution of pressure, as the momentum equations contain pressure as a source term and the continuity equation does not explicitly contain pressure.

Patankar[2] developed a method by which pressure can be calculated by substituting the discretized momentum equations into the discrete continuity equation. The pressure equation has the following form.

$$(a_{p-P})p_P = \sum (a_{p-nb})p_{nb} + b_p \tag{14}$$

The coefficients and source of the pressure equation, namely (a_{p-P}), (a_{p-nb}), and b_p, are derived from the momentum coefficients and are functions of velocity. If the velocity is known, this pressure equation can be used to calculate a pressure field that satisfies continuity. In the SIMPLER algorithm the pressure and momentum equations are solved iteratively with relaxation along with a pressure correction equation to simultaneously satisfy mass and momentum conservation at each time step. The RK-SIMPLER algorithm[7] solves the pressure equation once each time step and then uses that pressure in the explicit solution of the momentum equations.

The IRK-SIMPLER algorithm is developed as an extension of the RK-SIMPLER algorithm, which has shown to efficiently and accurately simulate unsteady flow problems, but time step restrictions may be severe in high Reynolds number cases[7]. Using implicit RK methods in place of explicit methods, IRK-SIMPLER attempts to improve the time step restrictions and reduce runtime. The IRK-SIMPLER algorithm uses implicit RK methods to integrate momentum equations in time and solves a pressure equation to satisfy continuity, with no approximate or correction equations required. No relaxation is needed for solving any of these equations.

For IRK-SIMPLER a pressure equation is reformulated each stage from the momentum stage equations, as opposed to the Crank-Nicolson based momentum equations used in RK-SIMPLER. In IRK-SIMPLER, diagonally implicit RK (DIRK) methods[27] are used to integrate the momentum equations, and the stage equations for x momentum are

$$(u_P)_s = (u_P)^n + \Delta t \sum_{l=1}^{s} \alpha_{s,l} F_u\big(t^n + \gamma_l \Delta t, u_l\big) \qquad \text{for } 1 \le s \le S \tag{15}$$

where s is the stage index, S is the number of stages in the DIRK method, and $\alpha_{s,l}$ and γ_l are coefficients of the specific DIRK method used. The function F_u is defined as

$$F_u(t, u) = \frac{\partial u}{\partial t} = \frac{R_u}{\rho \Delta x \Delta y \Delta z} \tag{16}$$

Rearranging and using the definition of F_u, the fully discrete x momentum equation for each stage becomes

$$a'_{u-P}(u_P)_s = \sum (a_{u-nb})_s (u_{nb})_s + b'_u + \Delta y \Delta z (p_w - p_e)_s \tag{17}$$

where

$$a'_{u-P} = (a_{u-P})_s + \frac{\rho\Delta x\Delta y\Delta z}{\alpha_{s,s}\Delta t} \qquad b'_u = (b_u)_s + \frac{\rho\Delta x\Delta y\Delta z}{\alpha_{s,s}\Delta t}(u_P)^n + \frac{R_s(u)}{\alpha_{s,s}} \tag{18}$$

$$R_s(u) = \rho\Delta x\Delta y\Delta z\sum_{l=1}^{s-1}\alpha_{s,l}F_u\big(t^n + \gamma_l\Delta t, u_l\big) \tag{19}$$

Similarly, the discretized form of the y momentum equation is

$$a'_{v-P}(v_P)_s = \sum(a_{v-nb})_s(v_{nb})_s + b'_v + \Delta x\Delta z(p_s - p_n)_s \tag{20}$$

where

$$a'_{v-P} = (a_{v-P})_s + \frac{\rho\Delta x\Delta y\Delta z}{\alpha_{s,s}\Delta t} \qquad b'_v = (b_v)_s + \frac{\rho\Delta x\Delta y\Delta z}{\alpha_{s,s}\Delta t}(v_P)^n + \frac{R_s(v)}{\alpha_{s,s}} \tag{21}$$

$$R_s(v) = \rho\Delta x\Delta y\Delta z\sum_{l=1}^{s-1}\alpha_{s,l}F_v\big(t^n + \gamma_l\Delta t, v_l\big) \tag{22}$$

and the discretized form of the z momentum equation is

$$a'_{w-P}(w_P)_s = \sum(a_{w-nb})_s(w_{nb})_s + b'_w + \Delta x\Delta y(p_b - p_f)_s \tag{23}$$

where

$$a'_{w-P} = (a_{w-P})_s + \frac{\rho\Delta x\Delta y\Delta z}{\alpha_{s,s}\Delta t} \qquad b'_w = (b_w)_s + \frac{\rho\Delta x\Delta y\Delta z}{\alpha_{s,s}\Delta t}(w_P)^n + \frac{R_s(w)}{\alpha_{s,s}} \tag{24}$$

$$R_s(w) = \rho\Delta x\Delta y\Delta z\sum_{l=1}^{s-1}\alpha_{s,l}F_w\big(t^n + \gamma_l\Delta t, w_l\big) \tag{25}$$

The x, y, and z momentum equations are rewritten as

$$(u_P)_s = \hat{u}_s + (d_u)_s(p_w - p_e)_s \qquad (v_P)_s = \hat{v}_s + (d_v)_s(p_s - p_n)_s \qquad (w_P)_s = \hat{w}_s + (d_w)_s(p_b - p_f)_s \tag{26}$$

with

$$\hat{u}_s = \frac{\sum(a_{u-nb}u_{nb})_s + b'_u}{a'_{u-P}} \qquad \hat{v}_s = \frac{\sum(a_{v-nb}v_{nb})_s + b'_v}{a'_{v-P}} \qquad \hat{w}_s = \frac{\sum(a_{w-nb}w_{nb})_s + b'_w}{a'_{w-P}} \tag{27}$$

$$(d_u)_s = \frac{\Delta y\Delta z}{a'_{u-P}} \qquad (d_v)_s = \frac{\Delta x\Delta z}{a'_{v-P}} \qquad (d_w)_s = \frac{\Delta x\Delta y}{a'_{w-P}} \tag{28}$$

At each stage, the momentum equations (Equation 26) are substituted into the discrete continuity equation (Equation 11) at each control volume face to yield a discrete equation for pressure.

$$\begin{aligned}(a_{p-P})_s(p_P)_s = {} & (a_{p-E})_s(p_E)_s + (a_{p-W})_s(p_W)_s + (a_{p-N})_s(p_N)_s + (a_{p-S})_s(p_S)_s \\ & + (a_{p-F})_s(p_F)_s + (a_{p-B})_s(p_B)_s + (b_p)_s\end{aligned} \tag{29}$$

with

$$(a_{p-E})_s = \rho\Delta y\Delta z(d_{u-e})_s \qquad (a_{p-W})_s = \rho\Delta y\Delta z(d_{u-w})_s \tag{30}$$

$$(a_{p-N})_s = \rho\Delta x\Delta z(d_{v-n})_s \qquad (a_{p-S})_s = \rho\Delta x\Delta z(d_{v-s})_s \tag{31}$$

$$(a_{p-F})_s = \rho\Delta x\Delta y(d_{w-f})_s \qquad (a_{p-B})_s = \rho\Delta x\Delta y(d_{w-b})_s \tag{32}$$

$$(a_{p-P})_s = (a_{p-E})_s + (a_{p-W})_s + (a_{p-N})_s + (a_{p-S})_s + (a_{p-F})_s + (a_{p-B})_s \tag{33}$$

$$(b_p)_s = (\hat{u}_w - \hat{u}_e)_s \rho \Delta y \Delta z + (\hat{v}_s - \hat{v}_n)_s \rho \Delta x \Delta z + (\hat{w}_b - \hat{w}_f)_s \rho \Delta x \Delta y \tag{34}$$

The pressure equation cannot be solved directly without a known velocity field because the pressure coefficients and source are functions of the unknown velocity components at the current stage s. Therefore, at each stage the pressure and momentum equations must be solved to satisfy all conservation equations.

The procedure for solving pressure and velocity is as follows. First, the implicit form of the momentum equations (Equations 17, 20, and 23) are solved for $(u, v, w)_s$ with the momentum coefficients and pressure lagged by one stage (i.e., assume $p_s = p_{s-1}$ and $a_s = a_{s-1}$ where at stage $s = 1$, $p_{s-1} = p^n$ and $a_{s-1} = a^n$). With updated velocity components, the momentum coefficients are updated to the s^{th} stage. Then, $(d_u, d_v, d_w)_s$ are calculated from Equation 28, and the pressure coefficients are calculated (Equations 30-33). An iterative loop is then preformed, which follows closely to that of the inner loops in the IDEAL algorithm[6]. First, pseudo-velocities $(\hat{u}, \hat{v}, \hat{w})_s$ are calculated from Equation 27. Then, the pressure source $(b_p)_s$ is calculated from Equation 34 and the pressure equation (Equation 29) is solved for p_s. With an updated pressure, the velocity components are updated with the explicit form of the momentum equaions, Equation 26, using the most recent pseudo-velocities.

This process of calculating pseudo-velocity, solving pressure, and updating momentum explicitly is an efficient method to simultaneously satisfy the continuity and momentum equations without the need for relaxation or multiple implicit solutions of the momentum equations. The pressure coefficients are constant throughout the iterative loop while the pressure source is recomputed as the velocity is updated. In practice, only a few iterations of the iterative loop are needed to solve for pressure and velocity (five iterations are used for all simulation in this paper).

The momentum coefficients are updated once after solving the momentum equations implicitly and can also be updated after the iterative loop to improve the coefficient. After recomputing the momentum coefficients the iterative loop is repeated. Recalculating the coefficients requires extra computations, but as will be shown in the results, recalculating coefficients more than once can allow the algorithm to take larger time steps and reduce runtime.

The final update to the $n+1$ time level for DIRK methods forms an explicit equation; however if stiffly accurate methods are used this final update becomes unnecessary. Stiffly accurate DIRK methods[27] require $(u, v, w, p)^{n+1} = (u, v, w, p)_S$. By using stiffly accurate DIRK methods in IRK-SIMPLER, once the final stage values of pressure and velocity are completed, the velocity and pressure at time level $n+1$ are known without requiring further computation.

To start the IRK-SIMPLER algorithm, before the first time step, the domain is initialized with values of pressure and velocity. If physically accurate initial conditions are not known, a uniform field is found to work well. When starting with uniform initial conditions, the conservation equations will not necessarily be satisfied at the beginning of the time step, but after the first time step is completed, all equations will tend towards conservation.

A flowchart for the IRK-SIMPLER algorithm is shown in Figure 1. IRK-SIMPLER is tested with a two stage DIRK method from Alexander[27], with its coefficients as follows.

$$\begin{aligned} \alpha_{1,1} &= \alpha_{2,2} = \alpha = 1 - \sqrt{2}/2 & \alpha_{2,1} &= 1 - \alpha \\ \gamma_1 &= \alpha & \gamma_2 &= 1 \\ \beta_1 &= 1 - \alpha & \beta_2 &= \alpha \end{aligned}$$

The coefficients are also shown in a Butcher tableau[28] in Figure 2.

2.2 Results

The IRK-SIMPLER algorithm is tested on a 3D square cylinder with laminar unsteady vortex shedding to examine the accuracy and efficiency of the new IRK-SIMPLER algorithm. For comparison, the SIMPLER algorithm with Crank-Nicolson time integration[29] and the RK-SIMPLER algorithm[7] are also tested.

A 3D square cylinder will shed vortices for certain Reynolds numbers flows. Figure 3 shows a schematic of the problem. Up to a critical Reynolds number of about 150 the vortex shedding is 2D with no variation along the z direction[30]. Above the critical Reynolds number flow will become 3D with secondary vortices

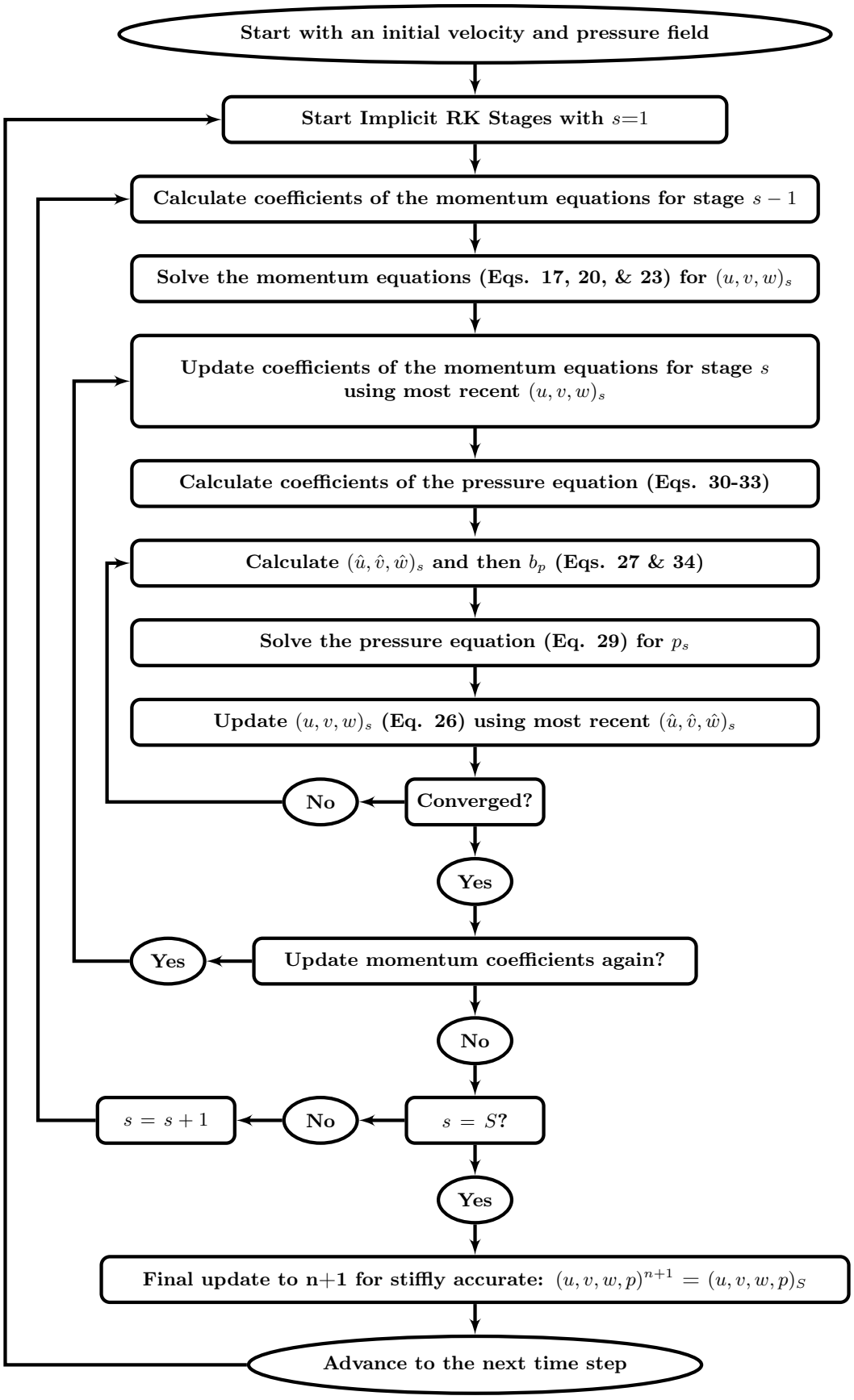

Figure 1: Diagram of IRK-SIMPLER Algorithm.

α	α	
1	$1-\alpha$	α
	$1-\alpha$	α

Figure 2: DIRK2, $\alpha = 1 - \sqrt{2}/2$

present. To test the IRK-SIMPLER algorithm in 3D, the 3D square cylinder with Reynolds number of 175 will be used following the same problem setup as [30]. The grid used is 162x82x26 with 16 grid cells along the cylinder in the x and y direction and 24 grid cells along the cylinder in the z direction with appropriate grid stretching to the boundaries. The body surfaces are no-slip walls, the left boundary is uniform inflow, the right boundary is a velocity outflow corrected for mass conservation, and all other boundaries are inviscid walls. Simulations are started impulsively and are run for 400 seconds. The traditional SIMPLER algorithm with Crank-Nicolson time integration is used as a baseline with the number of sub-iterations within each time step fixed at 20. Both the Power Law[2] and QUICK[31] schemes are used for spatial discretization. The QUICK schemes is implemented with the aid of Lagrange interpolation as developed in [32].

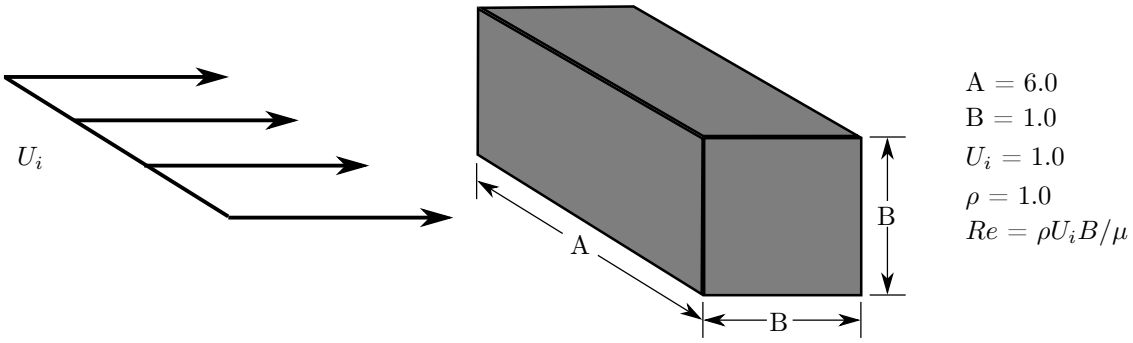

Figure 3: Schematic of the 3D square cylinder problem.

Contour plots of z-vorticity on the x-y mid-plane (z=3) are plotted in Figures 4 and 6 for the Power Law and QUICK schemes respectively, and y-vorticity on the x-z mid-plane (y=5) is plotted in Figures 5 and 7 for the Power Law and QUICK schemes respectively. From the y-vorticity plots it is apparent that the Power Law scheme is not able to capture the secondary vortices for the coarse grid used, while the QUICK scheme does capture the secondary vortices when using the same grid.

The coefficient of drag on the square cylinder versus when using the Power Law and QUICK schemes are plotted in Figures 8 and 9. There is variation between the methods for when the shedding starts and the methods do not line up exactly. This behavior is similar to the vortex shedding for a 2D flat plate investigated in [22]. For the Power Law simulations, the flow reaches an unsteady convergence with 2D shedding of vortices at a constant rate. For the QUICK simulations, the flow seems to reach an unsteady convergence but then both lift and drag peaks drop down and rise back up again. This occurs when the secondary vortices begin to form.

Tables 1 and 2 show results for the Power Law and QUICK schemes. The average coefficient of drag for the Power Law simulations approach 1.661 and the Strouhal number approaches 0.144. These values agree well with other simulation results[30]. For temporal accuracy, a average coefficient of drag of 1.661 ± 0.001, or within 0.1% of the converged value, is used.

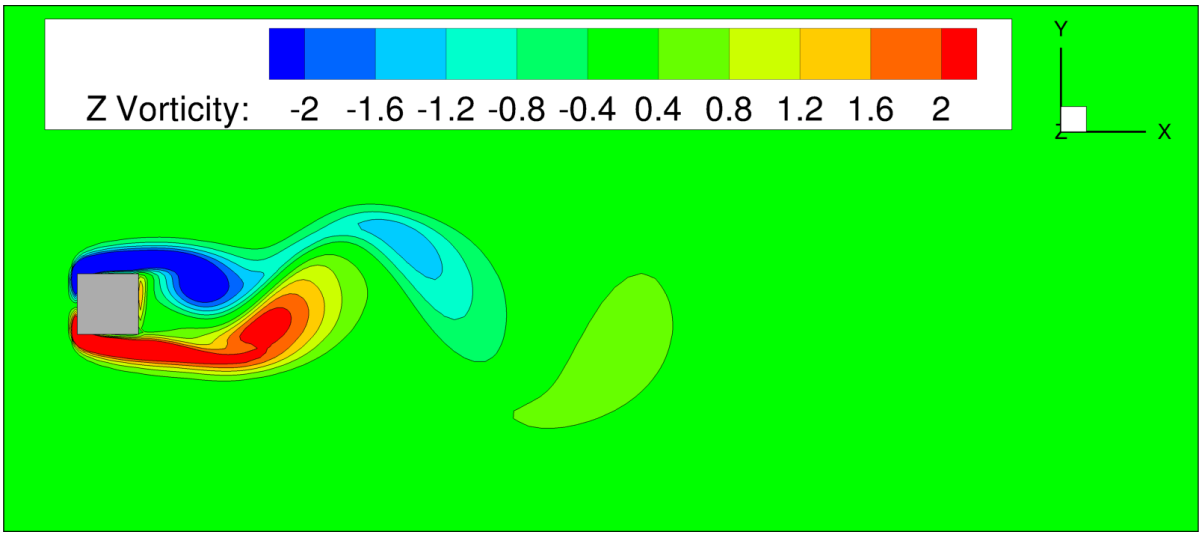

Figure 4: 3D square cylinder: contours of z-vorticity at x-y mid-plane using Power Law scheme.

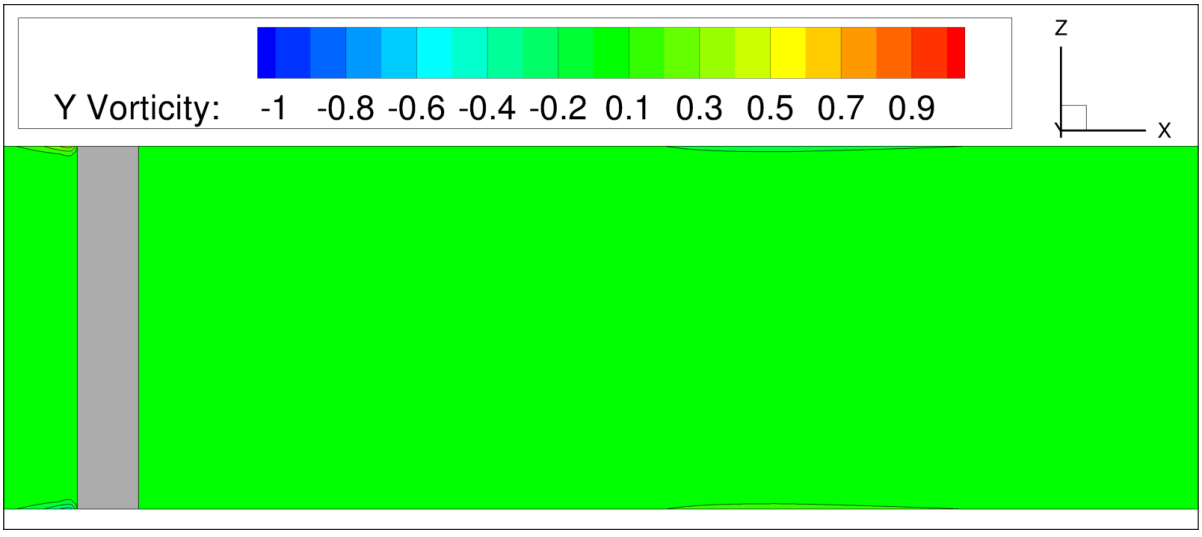

Figure 5: 3D square cylinder: contours of z-vorticity at x-y mid-plane using Power Law scheme.

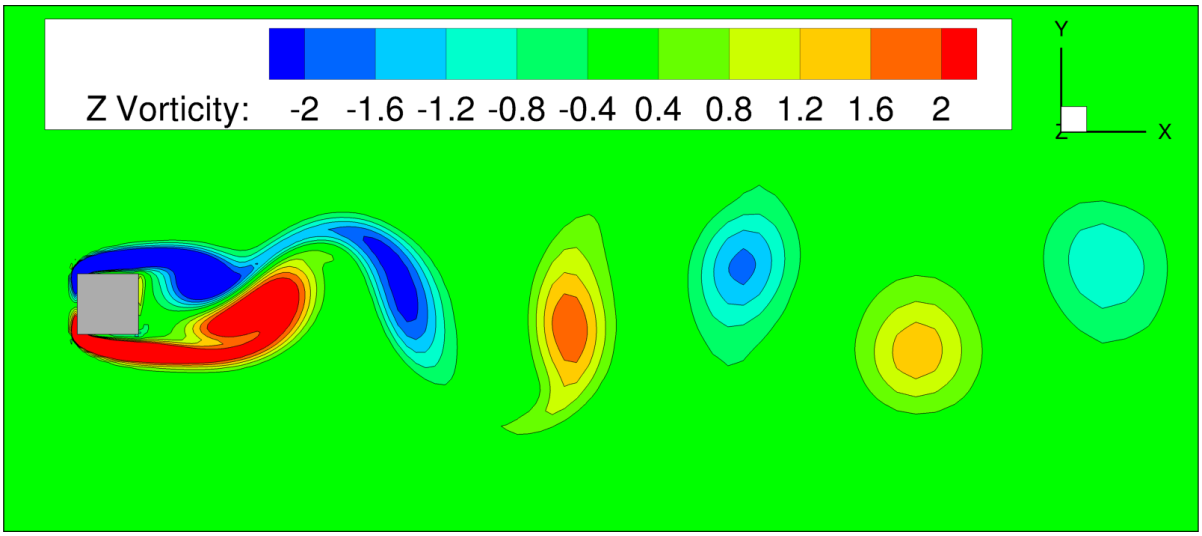

Figure 6: 3D square cylinder: contours of z-vorticity at x-y mid-plane using QUICK scheme.

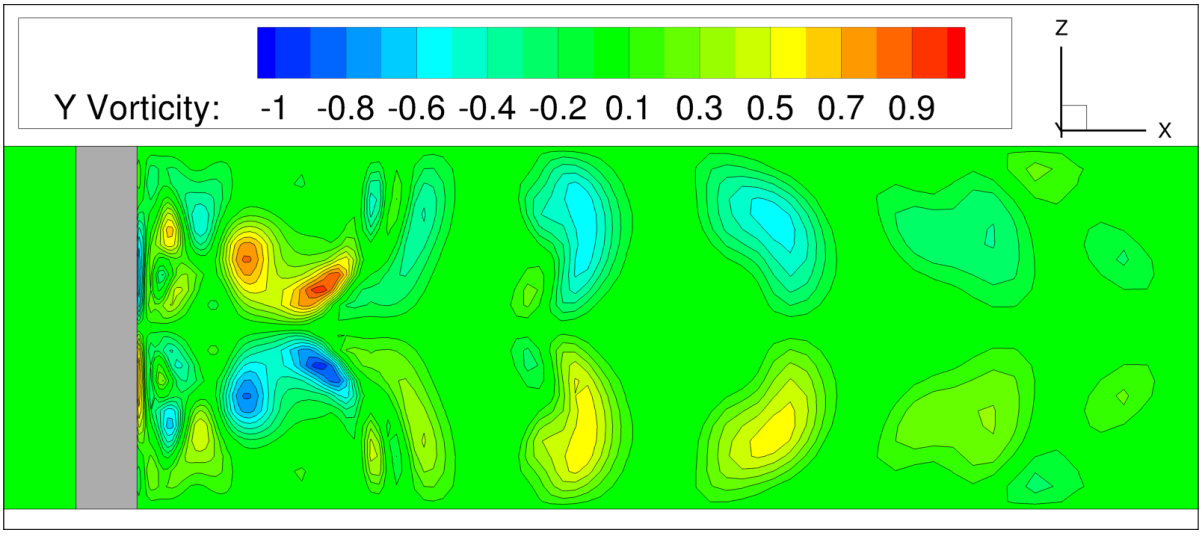

Figure 7: 3D square cylinder: contours of z-vorticity at x-y mid-plane using QUICK scheme.

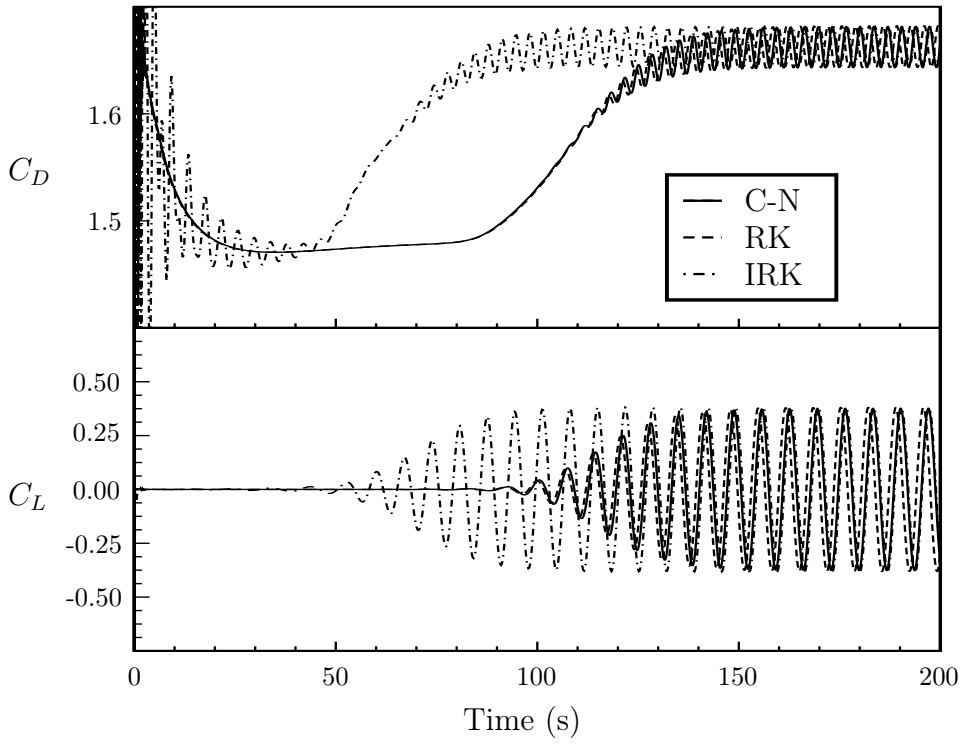

Figure 8: Coefficient of drag and lift on the 3D square cylinder using the Power Law scheme.

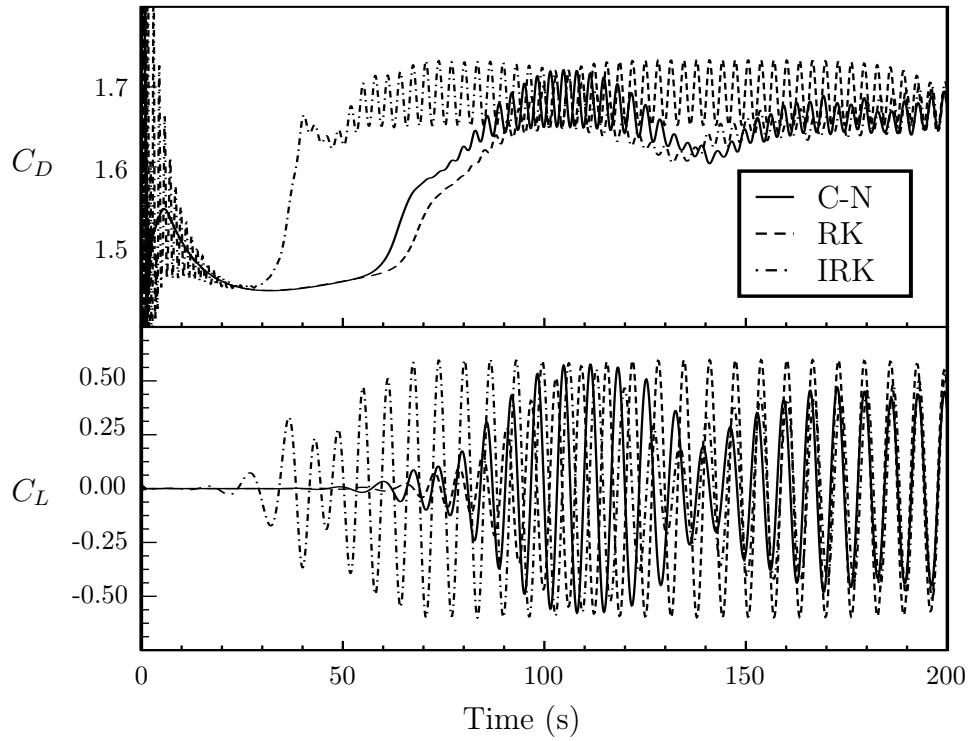

Figure 9: Coefficient of drag and lift on the 3D square cylinder using the QUICK scheme.

	Mom. Coef. Updates	Δt_{max} (sec.)	$\overline{C_D}$ at Δt_{max}	Sr at Δt_{max}	Δt_{Acc} (sec.)	CPU Time at Δt_{Acc} (min.)	Speedup at Δt_{Acc}	
C-N	—	1.00	1.796	0.112	0.10	1304.75	1.0	
RK	—	0.06	1.674	0.144	0.002	1181.05	1.1	Power
IRK	1	0.30	1.663	0.146	0.20	52.68	24.8	Law
IRK	2	0.70	1.660	0.143	0.70	21.11	61.8	

Table 1: 3D square cylinder Power Law results (speedup is relative to C-N).

	Mom. Coef. Updates	Δt_{max} (sec.)	$\overline{C_D}$ at Δt_{max}	Δt_{Acc} (sec.)	$\overline{C_D}$ at Δt_{Acc}	CPU Time at Δt_{Acc} (min.)	Speedup at Δt_{Acc}	
C-N	—	0.30	1.671	0.10	1.667	1277.44	1.0	
RK	—	0.04	1.658	0.002	1.652	1187.55	1.1	QUICK
IRK	1	0.06	1.664	0.06	1.664	191.03	6.7	
IRK	2	0.60	1.668	0.60	1.668	28.69	44.5	

Table 2: 3D square cylinder QUICK results (speedup is relative to C-N).

3 Application of Efficient Numerical Methods

3.1 Approximate Factorization

3.1.1 Theory

Approximate factorization[23] can be used to reduce the computations required to solve systems of equations. Approximate factorization allow for a 3D system of differential equations to be reduced to decoupled 1D systems. The approximate factorization of the momentum equations used in the IRK-SIMPLER algorithm is presented next. Starting with the DIRK stage equation for the x momentum equation, Equation 16, and substituting the F_u function results in:

$$(u_P)_s - (u_P)^n = \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}\Big[a_{u-E}u_E + a_{u-W}u_W - a^x_{u-P}u_P + a_{u-N}u_N + a_{u-S}u_S - a^y_{u-P}u_P + a_{u-F}u_F + a_{u-B}u_B - a^z_{u-P}u_P + b_u - \Delta y\Delta z(p_e - p_w)\Big]_s + R_s(u)\frac{\Delta t}{\rho\Delta x\Delta y\Delta z} \tag{35}$$

where a^x_{u-P}, a^y_{u-P}, and a^z_{u-P} are the central coefficient terms coming from the x, y, and z derivatives respectively and $a^x_{u-P} + a^y_{u-P} + a^z_{u-P} = a_{u-P}$. Using $\Delta u_s = u_s - u^n$ leads to

$$\begin{aligned}(\Delta u_P)_s = {} & \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}\Big[a_{u-E}\Delta u_E + a_{u-W}\Delta u_W - a^x_{u-P}\Delta u_P + a_{u-N}\Delta u_N + a_{u-S}\Delta u_S - a^y_{u-P}\Delta u_P \\ & + a_{u-F}\Delta u_F + a_{u-B}\Delta u_B - a^z_{u-P}\Delta u_P + b_u - \Delta y(p_e - p_w)\Big]_s + R_s(u)\frac{\Delta t}{\rho\Delta x\Delta y\Delta z} \\ & + \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}\Big[(a_{u-E})_s u^n_E + (a_{u-W})_s u^n_W + (a_{u-N})_s u^n_N + (a_{u-S})_s u^n_S \\ & + (a_{u-F})_s u^n_F + (a_{u-B})_s u^n_B - (a_{u-P})_s u^n_P\Big]\end{aligned} \tag{36}$$

Defining

$$(\Delta b_u)_s = \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}\Big[(a_{u-E})_s u_E^n + (a_{u-W})_s u_W^n + (a_{u-N})_s u_N^n + (a_{u-S})_s u_S^n \tag{37}$$
$$(a_{u-F})_s u_F^n + (a_{u-B})_s u_B^n - (a_{u-P})_s u_P^n + (b_u)_s - \Delta y(p_e - p_w)_s + \frac{1}{\alpha_{s,s}} R_s(u)\Big]$$

leads to

$$(\Delta u_P)_s = \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}\Big[a_{u-E}\Delta u_E + a_{u-W}\Delta u_W - a^x_{u-P}\Delta u_P + a_{u-N}\Delta u_N + a_{u-S}\Delta u_S - a^y_{u-P}\Delta u_P \tag{38}$$
$$+ a_{u-F}\Delta u_F + a_{u-B}\Delta u_B - a^z_{u-P}\Delta u_P\Big]_s + (\Delta b_u)_s$$

Next, operators f_x and f_y are defined such that

$$f_x\Big((\Delta u_P)\Big) = a_{u-E}\Delta u_E + a_{u-W}\Delta u_W - a^x_{u-P}\Delta u_P \tag{39}$$
$$f_y\Big((\Delta u_P)\Big) = a_{u-N}\Delta u_N + a_{u-S}\Delta u_S - a^y_{u-P}\Delta u_P \tag{40}$$
$$f_z\Big((\Delta u_P)\Big) = a_{u-F}\Delta u_F + a_{u-B}\Delta u_B - a^z_{u-P}\Delta u_P \tag{41}$$

which results in

$$\Big[1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_x - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_y - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_z\Big](\Delta u_P)_s = (\Delta b_u)_s \tag{42}$$

Up to this point the DIRK stage equations have only been manipulated. The following step is where the approximate factorization occurs.

$$\Big[1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_x - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_y - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_z\Big]$$
$$\approx \Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_x\Big)\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_y\Big)\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_z\Big) \tag{43}$$

which leads to

$$\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_x\Big)\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_y\Big)\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_z\Big)(\Delta u_P)_s = (\Delta b_u)_s \tag{44}$$

The error introduced in this approximation is $(\alpha_{s,s}/\rho\Delta x\Delta y\Delta z)^2\Delta t^2(f_xf_y + f_xf_z + f_yf_z)$
$+(\alpha_{s,s}/\rho\Delta x\Delta y\Delta z)^3\Delta t^3 f_xf_yf_z$ which is order Δt^2 in time. The solution can then be found in four steps by first defining

$$(\Delta u_P^{**})_s = \Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_z\Big)(\Delta u_P)_s \tag{45}$$
$$(\Delta u_P^{*})_s = \Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_y\Big)(\Delta u_P^{**})_s \tag{46}$$

- Step 1: Solve for $(\Delta u_P^*)_s$

$$\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_x\Big)(\Delta u_P^{*})_s = (\Delta b_u)_s \tag{47}$$

- Step 2: Solve for $(\Delta u_P^{**})_s$

$$\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho\Delta x\Delta y\Delta z}f_y\Big)(\Delta u_P^{**})_s = (\Delta u_P^{*})_s \tag{48}$$

- Step 3: Solve for $(\Delta u_P)_s$

$$\Big(1 - \frac{\alpha_{s,s}\Delta t}{\rho \Delta x \Delta y \Delta z} f_z\Big)(\Delta u_P)_s = (\Delta u_P^{**})_s \tag{49}$$

- Step 4: Update $(u_P)_s$

$$(u_P)_s = (u_P)^n + (\Delta u_P)_s \tag{50}$$

For a 3D structured Cartesian system, step 1 consists of solving independent 1D tri-diagonal systems of equations in the x direction for each y=constant and z=constant grid line. Step 2 consists of solving independent 1D tri-diagonal systems of equations in the y direction for each x= *constant* and z= *constant* grid line. Step 3 consists of solving independent 1D tri-diagonal systems of equations in the z direction for each x= *constant* and y= *constant* grid line. Solving these 1D systems of equations generally requires fewer computations than solving the full 3D system of equations.

In the IRK-SIMPLER algorithm, the approximate factorization is used at the beginning of each RK stage when the momentum equations are solved in the implicit form (Equations 17, 20, and 23). Instead of using an iterative 3D solution method, approximate factorization allows the momentum equations to be solved as sets of 1D systems of equations, and are solved independently with the TDMA. In IRK-SIMPLER, the momentum equations are only solved implicitly once at the beginning of each stage. After solving the momentum equations, the iterative loop takes place which simultaneously solves pressure and velocity. Therefore, errors introduced by the approximate factorization will be removed by the iterative loops.

3.1.2 Results

Table 3 shows the results for the 3D square cylinder problem using the QUICK scheme with and without approximate factorization. Using approximate factorization reduces the runtime when updating the momentum coefficients both once and twice with one update requiring 48% less runtime and two updates requiring 31% less runtime.

	Mom. Coef. Updates	Δt_{max} (sec.)	$\overline{C_D}$ at Δt_{max}	CPU Time at Δt_{max} (min.)	Speedup at Δt_{max}
IRK w/o AF	1	0.06	1.660	191.03	6.7
IRK w/ AF	1	0.07	1.651	99.73	12.8
IRK w/o AF	2	0.60	1.667	28.69	44.5
IRK w/ AF	2	0.60	1.673	19.78	64.6

Table 3: 3D square cylinder with QUICK using IRK-SIMPLER with and without approximate factorization (AF) results (speedup is relative to C-N).

3.2 Multigrid

3.2.1 Theory

Multigrid methods allow linear systems of equations to be solved with fewer computations and less runtime than with traditional methods[24, 25]. There are a variety of multigrid methods that have been developed for many different problems. For incompressible flow, a multigrid method as been developed based on the SIMPLER algorithm called FAS-SIMPLER[26]. A similar multigrid method will be developed for unsteady incompressible flow based on the IRK-SIMPLER algorithm to improve the convergence rate at each time step.

In the IRK-SIMPLER algorithm, to reach a converged solution (where all equations have low residual) at each time step several iterations of the iterative loop must be completed. This is where most of the computations occur in the IRK-SIMPLER algorithm and where the algorithm can slow down. To speed up the convergence rate, multigrid methods will be developed.

The non-linear system of equations to be solved has the form

$$A\mathbf{u} = \mathbf{f} \tag{51}$$

When attempting to solve for $\mathbf{u}$, after some iteration an approximate value $\mathbf{v}$ is known. The error can be defined as $\mathbf{e} = \mathbf{u} - \mathbf{v}$, however, this error is not computable without knowing the exact solution for $\mathbf{u}$. A computable measure of how close the approximate solution is to the exact solution is the residual, $\mathbf{r} = \mathbf{f} - A\mathbf{v}$. Equation 51 can be rearranged as

$$A\mathbf{e} = \mathbf{r} \tag{52}$$

As discussed in the introduction, using common iterative solution methods like Gauss-Seidel on a single grid will quickly remove any high frequency errors but low frequency errors require many iterations to be removed. To remedy this problem, restricting the problem to coarser grids will allow the low frequency errors to be removed much quicker. In FAS multigrid methods the variables being solved for ($\mathbf{u}$) are restricted to the coarser grids. This is useful for non-linear problems where the coefficients are dependent on the variables. In non-FAS methods only the error ($\mathbf{e}$) is restricted to the coarser grids. For the non-linear incompressible Navier-Stokes equations, the coefficients are dependent on the variables and so a FAS multigrid method will be used in present research. The notation for the restriction of values from a fine grid k to a coarse grid $k-1$ is $I_k^{k-1}\mathbf{v}^k$. The restriction of the variables in present research is accomplished by tri-linear interpolation while the restriction of the residuals for finite volume methods (which are integrated quantities) requires an addition, not an interpolation, of the fine grid residuals contained in each coarse grid volume.

After some iterations on the fine grid the high frequency errors have been removed while the low frequency errors remain. At this point we will call the solution $\mathbf{v}_{old}^k$ and calculate the residual $\mathbf{r}^k$. On the coarse grid the error will be defined by

$$\mathbf{e}^{k-1} = \mathbf{u}^{k-1} - I_k^{k-1}\mathbf{v}_{old}^k \tag{53}$$

Substituting this into Equation 52 leads to the equation to be solved on the coarse grid $k-1$.

$$A^{k-1}\mathbf{u}^{k-1} = I_k^{k-1}\mathbf{r}^k + A^{k-1}(I_k^{k-1}\mathbf{v}_{old}^k) \tag{54}$$

The value of $\mathbf{u}^{k-1}$ is initialized to $I_k^{k-1}\mathbf{v}_{old}^k$ when starting on the next coarser grid level. After some iterations on the coarse grid the low frequency errors will be removed. These values are then restricted to the next coarser grid in the same manner. Once the coarsest grid has been reached, the errors will be prolonged back up to the finer grids. The notation for prolongation of values from a coarse grid $k-1$ to a fine grid k is $I_{k-1}^k\mathbf{v}^{k-1}$. For present research, all prolongation is accomplished with tri-linear interpolation. The correction of fine grid values is found by

$$\mathbf{v}_{new}^k = \mathbf{v}_{old}^k + I_{k-1}^k(\mathbf{v}^{k-1} - I_k^{k-1}\mathbf{v}_{old}^k) \tag{55}$$

By prolonging errors to the fine grid some high frequency errors may be introduced, so a few more iterations are computed. Then the error is prolonged to the next finer grid until the finest grid level is reached. At this point one multigrid iteration has been completed. This form of multigrid iterations is called V-cycle[25], where all grid levels are visited in order from fine to coarse while restricting and then all grid levels are visited in order from coarse to fine while prolonging. In the presented simulations, the V-cycle is used in the IRK-SIMPLER algorithm with a fixed number of three iterations on each grid level. This fixed number has not been rigorously optimized and future research could include examining the impact of the number of iterations on each grid level.

Another form or multigrid iterations is called cycle-C and has been developed by Brandt[24]. Instead of visiting each grid level in order, cycle-C determines whether to move to a coarser or finer grid by monitoring the residuals. For present simulation the residual value monitored is the sum of the L2 norm of all the momentum equation residuals, $r = L2(\mathbf{r}_{x-mom}) + L2(\mathbf{r}_{y-mom}) + L2(\mathbf{r}_{z-mom})$. Cycle-C can move up or down grids at any point during the iterations as the criteria are met.

When determining whether to restrict to a coarser grid, the convergence rate, or the rate that residuals are dropping, is monitored. If the convergence rate is slow on the current grid, the high frequency errors

have been removed and the values will be restricted to the next coarser grid. Restriction will occur if

$$r^k > \eta r^k_{last} \tag{56}$$

where r^k_{last} is the residual value from the last iteration on the current grid level. The value of η is taken from Brandt at 0.6. Whenever moving to a new grid level the value of r^k_{last} is initialized to a large value, $r^k_{last} \to \infty$.

When determining whether to prolong to a finer gird, the convergence level, or the magnitude of the current residual, is monitored. If the convergence level is below a tolerance for that grid level, that grid level solution is converged and the values will be prolonged to the next finer gird. Prolongation will occur if

$$r^k < r^k_{tol} \tag{57}$$

On the finest grid, $k = N$, the value of r^N_{tol} is set to be the overall tolerance level for the problem. If this criteria is met on the finest grid level then the solution is found. On coarser grid levels, the value of r^k_{tol} is determined by a parameter δ and the next finer grid level residual by

$$r^k_{tol} = \delta r^{k+1}_{last} \tag{58}$$

In other words, if the residual on a coarse grid level is less than the residual on the next finer level by a specified amount, δ, that solution is deemed converged and will be prolonged to the next finer grid level. The value of δ is taken from Brandt as 0.3. This cycle-C method of Brandt allows for the most efficient use of the multiple grid levels by monitoring the residual and determining the optimal time to restrict and prolong. On the other hand, the V-cycle will simply go up and down the grid levels in order whether or not the convergence rate is slow or if the solution is converged. Both the V-cycle and cycle-C will be applied to the IRK-SIMPLER algorithm to determine if the convergence rate can be improved.

In the IRK-SIMPLER algorithm, iterative loops are preformed which simultaneously solve pressure and velocity. To get a converged solution each RK stage many iterations are required. Multigrid methods will be applied to reduce the number of computations required to get a converged solution at each RK stage. The initial solution of the momentum equations in IRK-SIMPLER is only solved once and does not use the multigrid method. The multigrid methods are used in the iterative loops to accelerate the converge rate of pressure and momentum equations. When using multigrid with the IRK-SIMPLER algorithm, the pressure and velocity are restricted and prolonged and the residual of the pressure and momentum equations are restricted. Each time a new grid level is reached the non-linear momentum and pressure coefficients are computed with the latest approximation.

Both V-cycle and cycle-C multigrid methods are applied to IRK-SIMPLER. A diagram of the V-cycle multigrid IRK-SIMPLER algorithm is shown in Figure 10, and a diagram of the cycle-C multigrid IRK-SIMPLER algorithm is shown in Figure 11.

3.2.2 Results

Multigrid methods are more efficient than single grid method when a system of equations is to be solved to a low residual level. In the previous sections, simulations were run with only a few iterations to complete the simulation as fast as possible. Although the results were accurate, the residual values were not guaranteed to reach a certain level each time step. The IRK-SIMPLER algorithm can instead be run with as many iterations as required to reach a given residual tolerance. Figure 12 shows the mass and momentum residuals versus CPU time for the IRK-SIMPLER algorithm and the multigrid IRK-SIMPLER method for one RK stage. The single grid method residuals drop quickly at first, when the high frequency errors are being removed, but they drop much slower when removing the low frequency errors. The multigrid method allows the residual to drop consistantly at a constant rate, but for about the first 25 seconds, the single grid momentum residuals are actually lower than the multigrid method.

The multigrid IRK-SIMPLER algorithm is tested on the 3D square cylinder problem with the QUICK scheme and same grid as previously used. All algorithms and method will run with a time step of $\Delta t = 0.60$ seconds with as many iterations as needed to yield the specified residual tolerance. Results are given in Table 4. Multigrid does not decrease the runtime when the residual tolerance is 1×10^{-6}, but for a residual

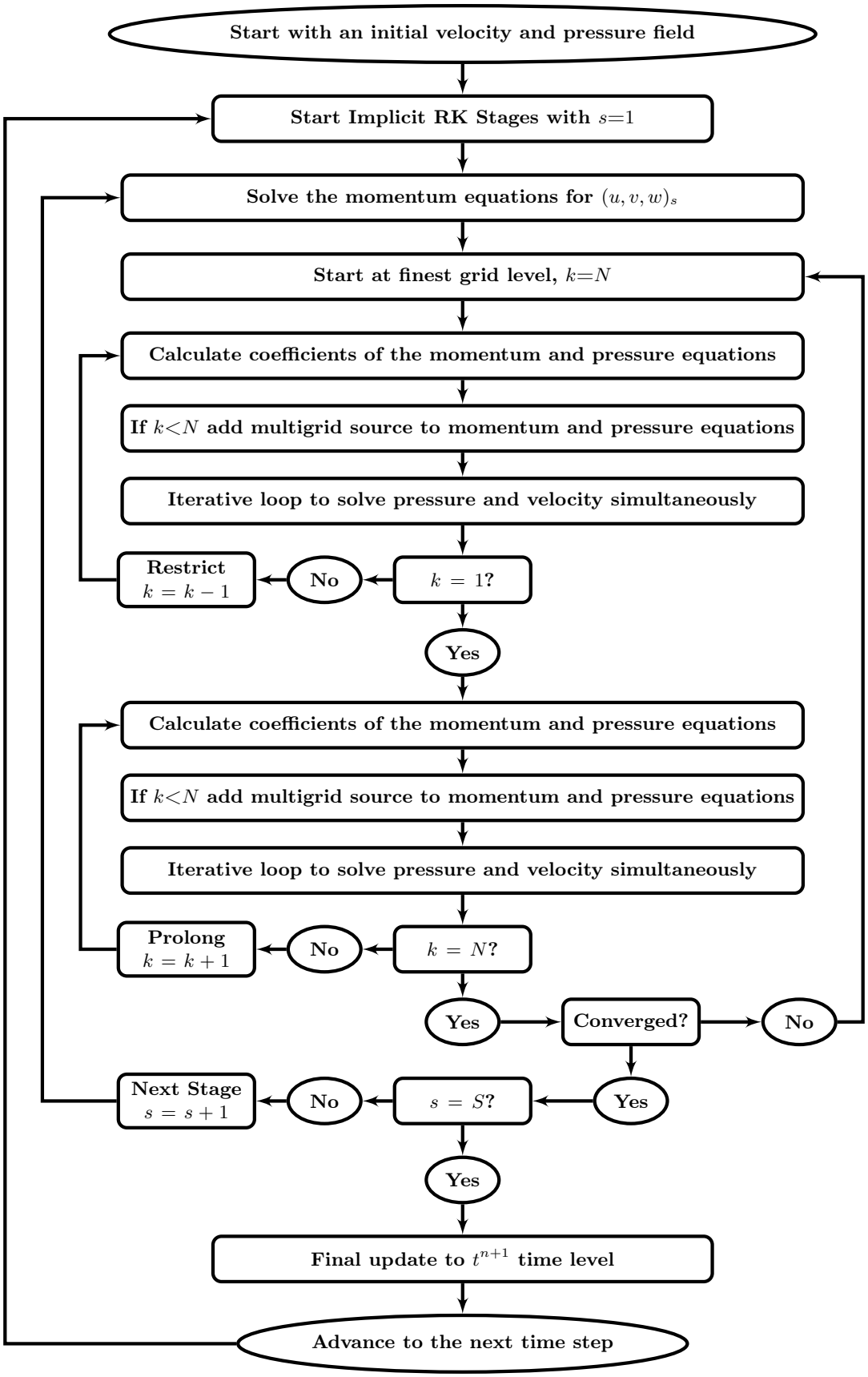

Figure 10: Diagram of V-cycle Multigrid IRK-SIMPLER Algorithm.

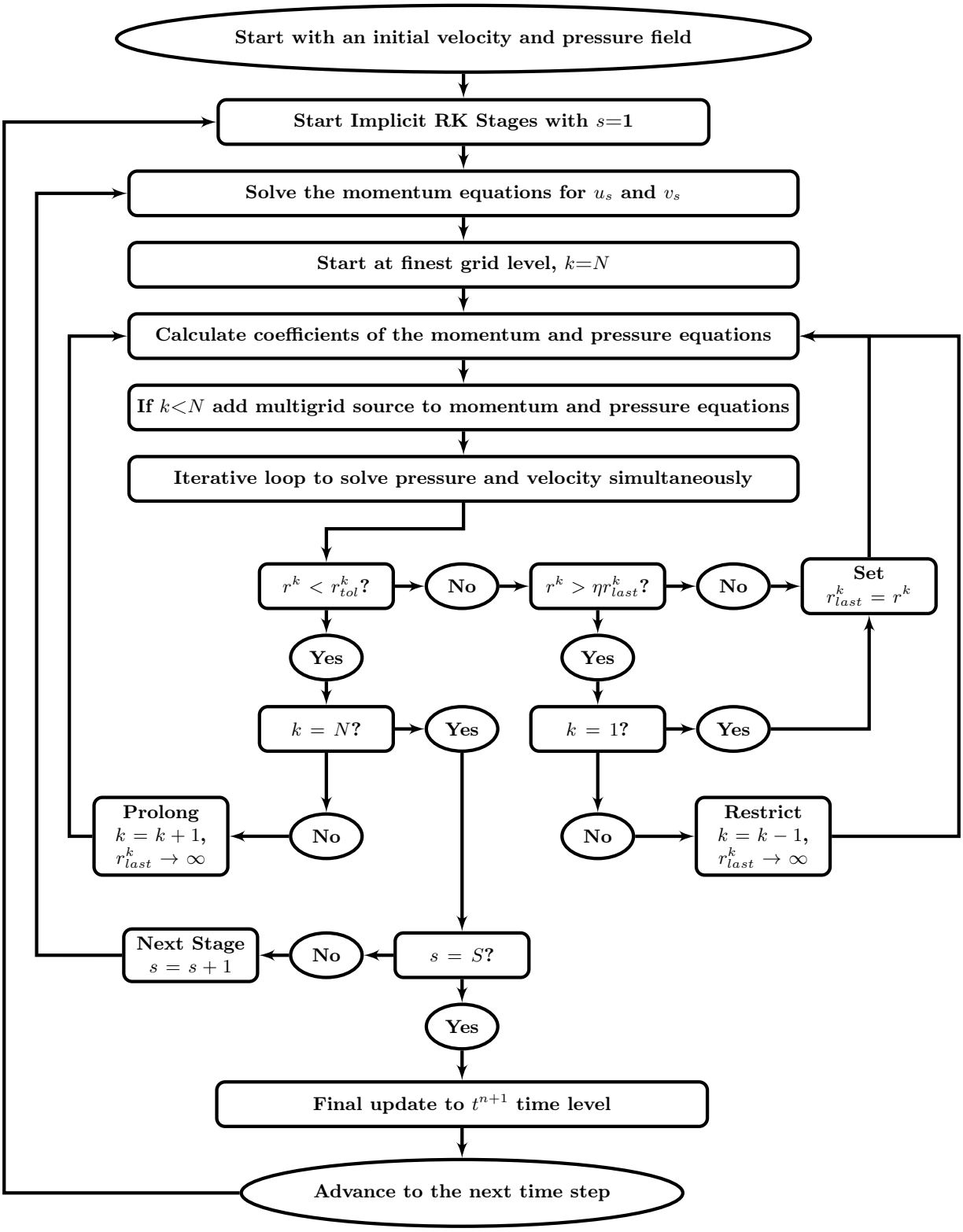

Figure 11: Diagram of cycle-C Multigrid IRK-SIMPLER Algorithm.

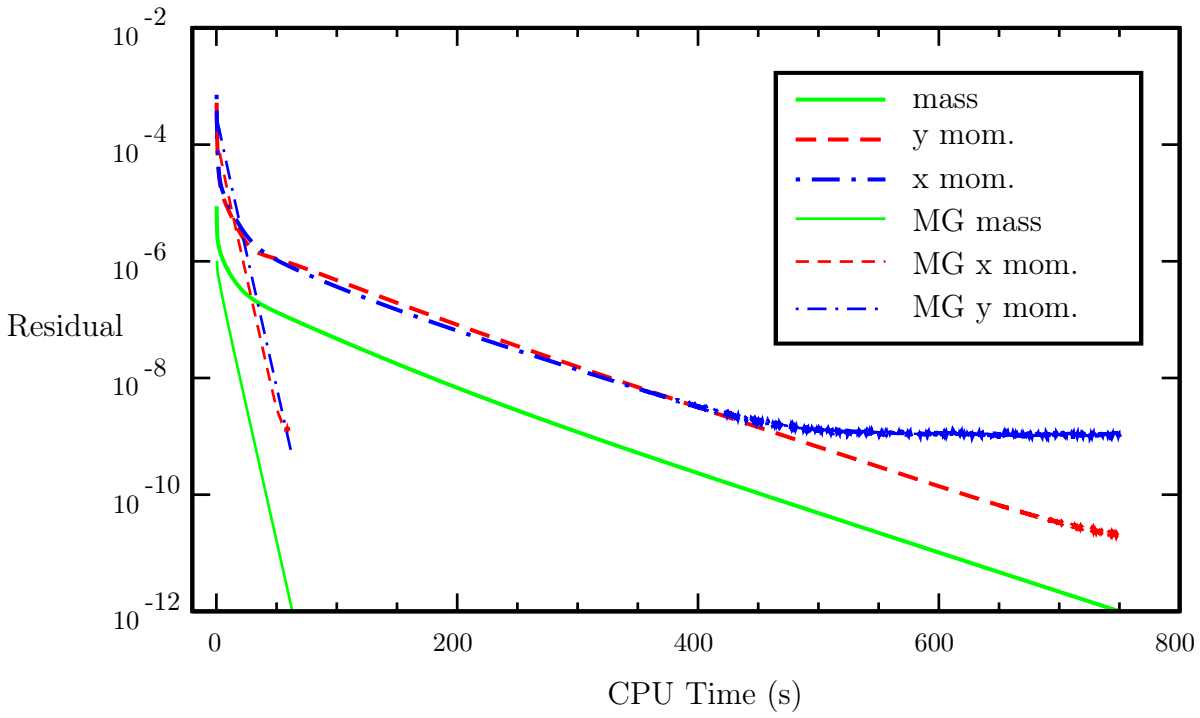

Figure 12: Residual values with and without multigrid for IRK-SIMPLER.

tolerance of 1×10^{-8}, cycle-C reduces the runtime by 50% while V-cycle reduces the runtime by 24%.

	Multigrid Method	Residual Tolerance	$\overline{C_D}$ at Δt	CPU Time at Δt (min.)	Speedup at Δt
IRK	—	1×10^{-6}	1.684	473.35	1.0
IRK	V-cycle	1×10^{-6}	1.640	886.37	0.5
IRK	cycle-C	1×10^{-6}	1.665	594.05	0.8
IRK	—	1×10^{-8}	1.685	1486.07	1.0
IRK	V-cycle	1×10^{-8}	1.666	1131.68	1.3
IRK	cycle-C	1×10^{-8}	1.683	743.78	2.0

Table 4: 3D square cylinder IRK-SIMPLER with multigrid results (4 grid levels).

The 3D square cylinder case presented here has a relatively coarse grid. Multigrid methods are most effective for highly refined grids with large systems of equations. More research is suggested to investigate the IRK-SIMPLER algorithm with multigrid for simulations with more refined grids.

4 Conclusions

The new IRK-SIMPLER algorithm is developed for efficient simulation of unsteady incompressible flows by integrating the momentum equations with DIRK methods and deriving a pressure equations each stage by substituting the momentum equations into the continuity equation. To solve the pressure and momentum equations efficiently, IRK-SIMPLER solves the momentum equations in the implicit form once and then

preforms a iterative loop which solves the pressure equation and updates the momentum equations with a simple explicit expression. This efficient loop allows both the pressure and momentum equations to be satisfied simultaneously without requiring relaxation or multiple implicit solutions of the momentum equations. The IRK-SIMPLER algorithm is tested on a 3D unsteady problem and is found to speedup the simulation compared to the legacy SIMPLER algorithm by up to 62 times.

To further enhance the efficiency of IRK-SIMPLER, two efficient numerical methods are integrated into the algorithm. The first method is approximate factorization which allows for the momentum equations to be solved as sets of independent 1D systems of equations instead of 3D systems of equations. Using approximate factorization allows for a reduction in runtime by up to 46%.

The second efficient method examined is multigrid. Two types of FAS multigrid cycles, V-cycle and cycle-C, are applied to the IRK-SIMPLER algorithm. The cycle-C multigrid method is found to preform 30+% faster than the V-cycle method. The cycle-C multigrid method reduces the runtime to achieve a converged solution at each RK stages every time step, resulting in up to 50% less runtime required. The lower the tolerance specified for the residuals the more efficient the multigrid method becomes. More research is required to investigate the multigrid method for problems with a larger number of grid cells as multigrid methods are known to preform very well on high density grids.

References

[1] Chorin, A. J., "A Numerical Method for Solving Incompressible Viscous Flow Problems," *Journal of Computational Physics*, Vol. 2, 1967, pp. 12-26.

[2] Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing, Washington, 1980.

[3] Tannehill, J. C., Anderson, D. A., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Taylor and Francis, Philadelphia, 1997.

[4] Issa, R. I., "Solution of the implicitly discretised fluid flow equations by operator-splitting," *Journal of Computational Physics*, Vol. 62, No. 1, 1986, pp. 40-65.

[5] Tao, W. Q., Qu, Z. G., and He, Y. L., "A Novel Segregated Algorithm for Incompressible Fluid Flow and Heat Transfer Problems – CLEAR (Coupled and Linked Equations Algorithm Revised) Part I: Mathematical Formulation and Solution Procedure," *Numerical Heat Transfer, Part B Fundamentals*, Vol. 45, No. 1, 2004, pp. 1-17.

[6] Sun, D. L., Qu, Z. G., He, Y. L., and Tao, W. Q., "An Efficient Segregated Algorithm for Incompressible Fluid Flow and Heat Transfer Problems – IDEAL (Inner Doubly Iterative Efficient Algorithm for Linked Equations) Part I: Mathematical Formulation and Solution Procedure," *Numerical Heat Transfer, Part B*, Vol. 53 No. 1, 2008, pp. 1-17.

[7] Rajagopalan, R. G. and Lestari, A.D., "RK-SIMPLER: An Explicit Time Accurate Algorithm for Incompressible Flows," To be published in *AIAA Journal*.

[8] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," *AIAA 14th Fluid and Plasma Dynamic Conference*, AIAA Paper 1981-1259, June 1981.

[9] Jameson, A. and Baker, T. J, "Solution of the Euler equations for complex problems," *AIAA Journal*, Vol. 1929, 1983, pp. 293-302.

[10] Kennedy, C. A., Carpenter, M. H., and Lewis, R. M., "Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations," *Applied Numerical Mathematics*, Vol. 35, 2000, pp. 177-219.

[11] Bijl, H., Carpenter, M. H., Vatsa, V. N., and Kennedy, C. A., "Implicit Time Integration Schemes for the Unsteady Compressible Navier-Stokes Equations: Laminar Flow," *Journal of Computational Physics*, Vol. 179, 2002, pp. 313-329.

[12] Jothiprasad, G., Mavriplis, D. J., and Caughey, D. A., "Higher-order time integration schemes for the unsteady Navier-Stokes equations on unstructured meshes," *Journal of Computational Physics*, Vol. 191, 2003, pp. 542-566.

[13] Hairer, E., Lubich, C., and Roche, M., *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer, Berlin, 1989.

[14] Le, H. and Moin, P., "An Improvement of Fractional Step Methods for the Incompressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 92, 1991, pp. 369-379.

[15] Cabuk, H., Sung, C. H., and Modi, V., "Explicit Runge-Kutta Method for Three-Dimensional Internal Incompressible Flows," *AIAA Journal*, Vol. 30, No. 8, 1992, pp. 2024-2031.

[16] Morinishi, Y., Lund, T. S., Vasilyev, O. V., and Moin, P., "Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow," *Journal of Computational Physics*, Vol. 143, 1998, pp. 90-124.

[17] Pereira, J. M. C., Kobayashi, M. H., and Pereira, J. C. F., "A Fourth-Order Accurate Finite Volume Compact Method for the Incompressible Navier-Stokes Solutions," *Journal of Computational Physics*, Vol. 167, 2001, pp. 217-243.

[18] Kampanis, N. A. and Ekaterinaris, J. A., "A Staggered Grid, High-Order Accurate Method for the Incompressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 215, 2006, pp. 589-613.

[19] Nikitin, N., "Third-Order-Accurate Semi-Implicit Runge-Kutta Scheme for Incompressible Navier-Stokes Equations," *International Journal for Numerical Methods in Fluids*, Vol. 51, 2006, pp. 221-233.

[20] Ijaz, M., "Implicit Runge-Kutta methods to simulate unsteady incompressible flows," Ph.D. Thesis, Texas A&M Univ., College Station, TX, 2007.

[21] Sanderse, B. and Koren, B., "Accuracy analysis of explicit Runge-Kutta methods applied to the incompressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 231, 2012, pp. 3041-3063.

[22] Fischels, M. V. and Rajagopalan, R. G.,"Comparison of Pressure-Based Runge-Kutta Schemes for Unsteady Incompressible Flows," *22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-3203, Dallas, June 22-26, 2015.

[23] Douglas, J. and Gunn, J. E., "A General Formulation of Alternating Direction Methods - Part I. Parabolic and Hyperbolic Problems," *Numer. Math.*, Vol. 6, 1964, pp. 428-453.

[24] Brandt, A., "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Mathematics of Computation*, Vol. 31, No. 138, 1977, pp. 333-390.

[25] Briggs, W. L., *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[26] Zori, L. A. and Rajagopalan, R. G., "The FAS multigrid method for the segrated solution of the SIMPLER algorithm," *Proceedings of the 6th International Symposium on CFD*, 3:1509-1514, Lake Tahoe, Nevada, 1995.

[27] Alexander, R., "Diagonally Implicit Runge-Kutta Methods for Stiff O.D.E.'s," *SIAM Journal on Numerical Analysis*, Vol. 14, No. 6, 1977, pp. 1006-1021.

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

[29] Kelkar, K. M., and Patankar, S. V., "Numerical Prediction of Vortex Shedding Behind a Square Cylinder," *International Journal for Numerical Methods in Fluids*, Vol. 14, No. 3, 1992, pp. 327-341.

[30] Saha, A. K., Biswas, G., and Muralidhar, K., "Three-dimensional study of flow past a square cylinder at low Reynolds numbers," *International Journal of Heat and Fluid Flow*, Vol. 24, 2003, pp. 54-66.

[31] Leonard, B. P., "A stable and accurate convective modeling procedure based on quadratic upstream interpolation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 19, 1979, pp. 59-98.

[32] Rajagopalan, R.G. and Yu, C., "Use of Lagrange Interpolation in Modeling Convective Kinematics," *Numberical Heat Transfer, Part B Fundamentals*, Vol. 36, No. 2, 1999, pp. 233-240.