ARTICLE IN PRESS
JID: CAF [m5G;April 21, 2018;15:15]

Computers and Fluids 000 (2018) 1–8

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Development of a coupled matrix-free LU-SGS solver for turbulent compressible flows

Jiří Fürst

Dept. of Technical Mathematics, Faculty of Mechanical Engineering, Center of Advanced Aerospace Technology, Czech Technical University in Prague, Technicka Street 4, 16607, Prague 6, Czech Republic

A R T I C L E I N F O

Article history:
Received 31 August 2017
Revised 9 March 2018
Accepted 13 April 2018
Available online xxx

Keywords:
Finite volume method
CFD
Coupled solver
Implicit method
Compressible flows
Transonic flows

A B S T R A C T

The paper deals with the development of the lower-upper symmetric Gauss–Seidel (LU-SGS) matrix-free finite volume solver for the simulation of compressible flows within the framework of the OpenFOAM package. The solver evaluates the convective fluxes using approximate Riemann solvers with limited piece-wise linear reconstructions whereas the viscous fluxes are approximated using a central scheme. The time evolution is realised through the backward differentiation formula of first or second order. The system of non-linear equations is then solved with the help of the matrix-free LU-SGS method. The developed solver is used to solve several flow problems and compared to a pressure-based segregated solver. Our numerical experiments indicate that the LU-SGS solver is more efficient for flows with higher Mach numbers and provides better resolution of shock waves. Moreover the LU-SGS solver benefits from the low memory footprint and does not use any problem specific setup.

1. Introduction

Computational Fluid Dynamics (CFD) is a fundamental tool for analysis and optimization of aerodynamic design. In the field of aeronautical engineering, turbomachinery or internal combustion engines, it is often possible to encounter high-speed flows with a significant compressibility effect. In these cases, non-linear phenomena such as shock waves with sudden changes in pressure, temperature, and velocity occur. These changes, based on the basic conservation laws, are tightly coupled.

Great efforts have been made in the development of numerical methods for the solution of compressible flows at various Mach numbers. For the low Mach number or incompressible flows, the so-called pressurebased methods represent an efficient solution procedure. Especially the segregated Semi-Implicit Pressure Linked Equations (SIMPLE) method [1] or its variants are commonly used for this kind of flows. Although being originally developed for incompressible or weakly compressible flows, the later developments [2–4] broadened the range of applicability of the pressure-based methods to high speed flows including transonic cases with shocks. Moreover, the development of coupled pressure-based methods [5,6] shows an important increase in the efficiency over the segregated approach at the cost of larger complexity of the code and larger memory requirements.

An alternative to the segregated pressure-based methods are the so-called coupled density-based solvers which use the concept of shock-capturing Godunov schemes using Riemann solvers, see e.g. [7]. This family of methods has been developed especially for convection dominated high speed flows and provides very robust schemes which are usually a preferred choice for transonic or hypersonic flows with shock waves, or flows with combustion.

OpenFOAM [8] is the widely used open source framework for the numerical solution of partial differential equations using the finite volume method. The C++ based library contains several CFD modules targeted mostly to the pressure-based segregated solvers. So far, there is only one coupled density-based solver using the central-upwind scheme of Kurganov and Tadmor [9] (the so-called *rhoCentralFoam*) in the standard OpenFOAM package. The FOAM-extend package (an extended fork of the OpenFOAM package) contains a specialized library developed originally under the name *densityBasedTurbo* by O. Born et al. [10]. The library provides a general framework for the development of coupled density-based solvers with several numerical fluxes including the Roe, the Rusanov, the HLLC, or the AUSM+up fluxes [7,11]. Moreover, the FOAM-extend package contains also a ready made solver called *dbnsTurbFoam* using this library. Unfortunately, both solvers, the *rhoCentralFoam* as well as the *dbnsTurbFoam*, use explicit methods for integration in time which leads to a strong stability limit on the time step and therefore both solvers become extremely inefficient in the case of flows at high Reynolds numbers with a mesh refinement in the vicinity of the wall.

E-mail address: Jiri.Furst@fs.cvut.cz

https://doi.org/10.1016/j.compfluid.2018.04.020
0045-7930/© 2018 Elsevier Ltd. All rights reserved.

The matrix-free lower-upper symmetric Gauss–Seidel (LU-SGS) scheme [12] is widely used because of its simplicity and very low memory requirements. The LU-SGS solver based on the OpenFOAM framework has been reported by Kim and Kim in [13] for the steady state case without giving enough details on the method. The article [14] describes the implementation of the LU-SGS scheme for inviscid compressible flows both for steady state and unsteady flows using the dual time stepping technique. In this article authors develop the scheme using the Steger-Warming flux splitting which needs the evaluation of the flux Jacobians and small matrix (5 by 5) operations. A similar method has been used also by Heyns et al. [15].

The current work is based on the description of the implementation of LU-SGS scheme given in the paper [14] and replaces the evaluation of the flux Jacobian by the finite difference approximation resulting in a true matrix-free method. Moreover, the method has been extended to the unsteady case assuming both the rotating reference frame or the arbitrary Lagrangian–Eulerian method.

2. The numerical method

The motion of a compressible gas is described by the following system of equations expressing conservation of mass, momentum, and energy

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{U}) = 0,$$
$$\frac{\partial(\rho\mathbf{U})}{\partial t} + \nabla\cdot(\rho\mathbf{U}\otimes\mathbf{U}) + \nabla p = \nabla\cdot\tau,$$
$$\frac{\partial(\rho E)}{\partial t} + \nabla\cdot[(\rho E + p)\mathbf{U}] = \nabla\cdot(\tau\cdot\mathbf{U}) + \nabla\cdot(k\nabla T),$$

where ρ is the density, $\mathbf{U}$ is the velocity vector, p is the pressure, $\boldsymbol{\tau}$ is the (effective) stress tensor, E is the specific total energy, k is the (effective) thermal conductivity, and T is the temperature. The system is closed by the equation of state for ideal gas $p/\rho = rT$ where r is the specific gas constant.

Spatial derivatives at the left hand side of the previous equations represent inviscid terms whereas the right hand side represents viscous terms. The above equations can be expressed in the integral form suitable for the finite volume approximation as follows (assuming a fixed control volume Ω at first)

$$\frac{d}{dt}\int_\Omega \mathbf{W}\,dV + \oint_{\partial\Omega}(\mathbb{F} - \mathbb{F}^v)\cdot d\mathbf{S} = 0, \tag{1}$$

where $\mathbf{W} = [\rho, \rho\mathbf{U}, \rho E]$ is the vector of conservative variables and $\mathbb{F}$ and $\mathbb{F}^v$ represent inviscid and viscous fluxes.

The integral form of the system Eq. (1) is discretized in space using the collocated finite volume method (FVM) by taking

$$\mathbf{W}_i(t) = \frac{1}{|\Omega_i|}\iiint_{\Omega_i}\mathbf{W}(\mathbf{x},t)\,dV,$$

where Ω_i is the mesh cell i. Hence Eq. (1) can be approximated as

$$|\Omega_i|\frac{d\mathbf{W}_i}{dt} = -\mathbf{R}(\mathbf{W})_i = -\sum_{j\in N_i}(\mathbb{F}_{ij} - \mathbb{F}^v_{ij})\cdot\mathbf{S}_{ij}, \tag{2}$$

where N_i is the set of indices of neighbor cells, $\mathbf{S}_{ij} = ||\mathbf{S}_{ij}||\mathbf{n}_{ij}$ is the scaled normal vector to the face shared between cells i and j oriented towards to cell j and $\mathbb{F}_{ij}$ and $\mathbb{F}^v_{ij}$ are the numerical fluxes. The viscous fluxes $\mathbb{F}^v_{ij}$ in 2 are discretized directly with the OpenFOAM built-in schemes (e.g. central scheme) and the discretization of inviscid fluxes $\mathbb{F}_{ij}$ is made using limited piece-wise linear reconstructions with the AUSM+up [11] or HLLC numerical fluxes [7].

2.1. The LU-SGS scheme for steady flows

In the case of steady state problem, the above devised system of non-linear ordinary differential equations can be solved using a local pseudo-time marching method. The time derivative is replaced by the first order backward difference using the local value of the time step

$$|\Omega_i|\frac{\mathbf{W}_i^{n+1} - \mathbf{W}_i^n}{\Delta t_i} = -\mathbf{R}(\mathbf{W}^{n+1})_i \approx -\mathbf{R}(\mathbf{W}^n)_i - \sum_j \frac{\partial\mathbf{R}(\mathbf{W}^n)_i}{\partial\mathbf{W}_j}\left(\mathbf{W}_j^{n+1} - \mathbf{W}_j^n\right),$$

or denoting by $\Delta\mathbf{W}^n = \mathbf{W}^{n+1} - \mathbf{W}^n$

$$\sum_j\left(\frac{|\Omega_i|}{\Delta t_i}\mathbb{I} + \frac{\partial\mathbf{R}(\mathbf{W}^n)_i}{\partial\mathbf{W}_j}\right)\Delta\mathbf{W}_j^n = -\mathbf{R}(\mathbf{W}^n)_i. \tag{3}$$

This type of the implicit method was used successfully by many authors, see e.g. [12] or [16] in combination with standard linear equation solvers such as GMRES [17]. The main disadvantage of the straightforward approach is that one has to assemble the Jacobian matrix $\partial\mathbf{R}/\partial W$, which needs quite a huge amount of the memory. Together with the demands of the linear solver, the memory requirements can grow up to $50 - 100$ times the size of the array of unknowns. Such large memory requirements could be prohibitive and therefore we use the simple matrix-free lower-upper symmetric Gauss–Seidel method (LU-SGS) despite its lower performance in comparison with e.g. GMRES or multigrid methods.

In the first step, the Jacobian matrix $\partial\mathbf{R}/\partial\mathbf{W}$ is replaced by its lower order approximation calculated using thin-layer approximation for viscous terms and the first order approximation of convective terms with Rusanov flux

$$\mathbb{F}_{ij}\cdot\mathbf{S}_{ij} \approx \frac{1}{2}\left(\mathbb{F}(\mathbf{W}_i) + \mathbb{F}(\mathbf{W}_j)\right)\cdot\mathbf{S}_{ij} - \frac{\lambda_{ij}}{2}\left(\mathbf{W}_j - \mathbf{W}_i\right),$$

where λ_{ij} is the spectral radius of Jacobian of $\mathbb{F}\cdot\mathbf{S}$, i.e. $\lambda_{ij} = |\mathbf{u}_{ij}\cdot\mathbf{S}_{ij}| + a_{ij}||\mathbf{S}_{ij}||$ with $\mathbf{u}_{ij}$ being the velocity at the face between the cells i and j and a_{ij} is the sound speed. Thanks to the simplicity of the Rusanov flux the low order residual reduces to

$$\mathbf{R}(\mathbf{W})_i^{lo} = \frac{1}{2}\sum_{j\in N_i}\lambda^*_{ij}\mathbf{W}_i + \frac{1}{2}\sum_{j\in N_i}\left(\mathbb{F}(\mathbf{W}_j)\cdot\mathbf{S}_{ij} - \lambda^*_{ij}\mathbf{W}_j\right)$$

where λ^* includes both the spectral radii of Jacobians of convective terms and the thin-layer approximation of viscous terms, hence

$$\lambda^*_{ij} = \lambda_{ij} + \frac{||\mathbf{S}_{ij}||}{||\mathbf{x}_i - \mathbf{x}_j||}\max\left(\frac{4}{3\rho_{ij}}, \frac{\gamma}{\rho_{ij}}\right)\left(\frac{\mu}{Pr} + \frac{\mu_T}{Pr_T}\right),$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the position vectors of centers of cells i and j, ρ_{ij} is the density at the face between cells i and j, γ is the ratio of specific heats, μ and μ_T are the molecular and turbulent viscosities, Pr is the Prandtl number, and Pr_T is the turbulent Prandtl number, see [12]. Finally the product of Jacobians with $\Delta\mathbf{W}$ is approximated with finite differences and the system of equations is solved with the LU-SGS method.

Let L and U denote the sets of cells belonging to lower and upper part of the matrix:

$$L_i = \{j\in N_i : j < i\},$$
$$U_i = \{j\in N_i : j > i\}.$$

Then the matrix-free LU-SGS scheme can be written using the following two step procedure:

$$D_i\Delta\mathbf{W}_i^{(1)} = -\mathbf{R}_i - \frac{1}{2}\sum_{j\in L_i}\left[\Delta\mathbb{F}_j^{(1)}\cdot\mathbf{S}_{ij} - \lambda^*_{ij}\Delta\mathbf{W}_j^{(1)}\right], \tag{4}$$