Computers and Fluids 147 (2017) 41–62

Contents lists available at ScienceDirect

Computers and Fluids

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

Development and validation of a multi-strand solver for complex aerodynamic flows

Vinod K. Lakshminarayan[a,1,*], Jayanarayanan Sitaraman[b,1], Beatrice Roget[a,1], Andrew M. Wissink[c,2]

[a] Science & Technology Corporation, NASA Ames Research Center, Moffett Field, CA USA
[b] Parallel Geometric Algorithms LLC, Sunnyvale, CA USA
[c] US Army Aviation Development Directorate - ADD (AMRDEC), Moffett Field, CA USA

ARTICLE INFO

Article history:
Received 12 August 2016
Revised 16 December 2016
Accepted 1 February 2017
Available online 2 February 2017

Keywords:
Strand grid
Multi-strand approach
Compressible flow
Finite volume method
Dual-mesh paradigm
Cartesian AMR
Hovering rotor
Tip vortices

ABSTRACT

The strand grid approach is a flow solution method where a prismatic-like grid using "strands" is grown to a short distance from the body surface to capture the viscous boundary layer and the rest of the domain is covered using an adaptive Cartesian grid. The approach offers several advantages in terms of nearly automatic grid generation and adaptation, ability to implement fast and efficient flow solvers that use structured data in both the strand and Cartesian grids, and the development of efficient and highly scalable domain connectivity algorithm. An improvement to this approach is the multi-strand strategy, where multiple strands are allowed from each surface vertex to enhance grid resolution near sharp corners. This paper introduces a fully parallel and highly efficient flow solver called mStrand that is developed from ground-up to operate on multi-strand meshes. The strand solver is integrated to HPCMP CREATE™-AV Helios framework to simulate complex aerodynamic flows. Detailed validation of the solver is shown on problems with varying degrees of complexity and comparison with experimental data. A performance study shows that the strand solver is nearly as efficient as a structured grid solver.

1. Introduction

The use of computational fluid dynamics (CFD) arise in many scientific and engineering applications. One of the most notable difficulties in applying CFD early in the design process is the lack of automation in mesh generation. With the advent of High Performance Computing (HPC), the fraction of time required for mesh generation and problem setup has increased significantly. Automating this process, coupled with efficient parallel algorithms, will enable turnaround times that are appropriate for engineering design.

Traditionally, there are two primary types of flow solvers available - structured grid solvers and unstructured grid solvers. Structured grid solvers are efficient and allow use of higher order algorithms. The efficiency of structured grid solvers enables use of many more grid points to provide accurate solution. However, generating structured grid, even while using multi-block structured and/or overset meshes, can be extremely challenging. On the other hand, unstructured codes generally offer more flexibility and automation in grid generation and problem setup for geometrically-complex bodies such as fuselages and hubs. But, the overall computational cost associated with such solvers to provide comparative level of accuracy as a structured grid solver is significantly higher. Therefore, in order to perform high-fidelity modeling and simulation that has potential for very high degrees of automation, an approach is needed that achieves efficiency comparable to structured grid solvers but with a more automated mesh generation strategy.

One such approach is the strand grid approach [1,2]. In this approach, the volume mesh generation process is fully automated by extending a viscous-quality prismatic mesh directly from a surface tessellation composed of triangles, quadrilaterals, or other shapes. A near-body volume grid that resolves the viscous boundary layer is constructed by inflating the surface grid along curves that can be represented with few parametric quantities. These curves, referred to as "strands", are most commonly just straight lines, each represented by a pointing vector and a length (see Fig. 1). Strands extrude a short distance from the solid boundary and are clipped with a "clipping index", when intersected with the adaptive Cartesian grids, which cover the rest of the domain to the outer boundaries.

* Corresponding author.
E-mail addresses: vinod.k.lakshminarayan.ctr@mail.mil (V.K. Lakshminarayan), jayanarayanan.sitaraman.ctr@mail.mil (J. Sitaraman), beatrice.f.roget.ctr@mail.mil (B. Roget), andrew.m.wissink.civ@mail.mil (A.M. Wissink).
[1] Research Scientist.
[2] Aerospace Engineer.

http://dx.doi.org/10.1016/j.compfluid.2017.02.002
0045-7930/

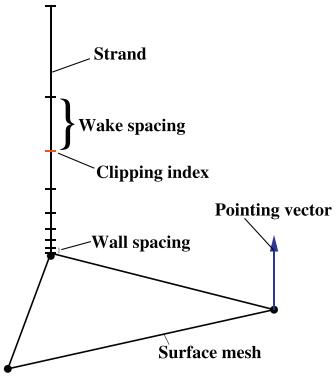

Fig. 1. Strand pointing vector and clipping index.

Strand near-body meshes combined with Cartesian off-body grids not only support automatic viscous mesh generation and adaptation but also provide a number of other advantages. First, both strand and Cartesian meshes possess some grid structure, facilitating efficient implementations of high-order accurate discretizations and solution methods such as high-order finite differencing, line-implicit solvers, and directional multi-grid coarsening. Second, each strand on the strand mesh can be represented with minimal information such as strand direction, length etc., thus reducing the entire volume mesh description to primarily a surface definition. This allows use of highly efficient and highly scalable domain connectivity package [3] to facilitate data exchange between the two mesh types. Third, both the strand and Cartesian grids easily permit use of Adaptive Mesh Refinement (AMR) enabling local refinement around important flow features. Because the strand grid is structured in the normal direction, unstructured refinement is performed only on the surface which avoids many of the cell quality issues that often occur with frequent and persistent adaptation of 3D tetrahedral elements.

Although a strand-based framework offers a potential solution for complete automation, a robust strand solver that can simulate realistic engineering problems has not yet been realized. From the several research articles published [2,4–8] in the past, it is seen that one of the primary challenges in simulating complex geometries with this framework arises when handling convex corners. With a single strand per surface vertex, the regions near convex corners are not well resolved, which results in poor solver convergence and accuracy. The use of multiple strands allows better resolution of these corners, but the grid can have extremely small volumes that can affect the convergence of the flow solver. Furthermore, surface tessellations with concave-convex type edges/corners can pose scenarios where a single-strand direction cannot yield a valid volume grid. Having multiple strand provides an elegant solution to this problem. Generation of multi-strand type meshes has been explored in detail by Haimes et al. [9]. In this work, a multi-strand (mStrand) solver is developed from ground up to be able to handle multiple strands emanating from the surface as well as small volumes resulting from such a mesh. The solver is implemented to be fully parallel from its inception and particular emphasis is made on its robustness and efficiency.

As a part of this work, mStrand is integrated with the Helios framework to simulate complex aerodynamic problems. The Helios software [10–12] is a high-fidelity rotary-wing product of the HPCMP CREATE^{TM}-AV (air vehicles) program [13]. The code was introduced in 2010 to provide a high-fidelity analysis capability to the DoD for the acquisition of new rotary-wing aircraft and is now in routine use at several DoD sites and US Helicopter companies. Helios uses a dual mesh paradigm [14] as the basis of the CFD aerodynamics solution procedure, where a near-body solver is used to capture viscous flow around complex geometry, and block structured Cartesian solver [15] in the off-body to resolve the wake through a combination of high-order algorithms and adaptive mesh refinement (AMR). The PUNDIT [16] domain connectivity software facilitates parallel data exchange between the two mesh types as well as enables relative motion between the mesh systems. Coordination of the different codes is managed through a lightweight and flexible Python-based infrastructure [17,18]. Once fully tested, mStrand is intended to become one of the primary near-body solver in the Helios framework.

The remainder of the paper is organized as follows. Section 2 provides a detailed description of mStrand. Section 3 performs accuracy analysis of the flow solver. This is then followed by a description of the mesh generation strategies used in this work in Section 4. Detailed validation of mStrand in both standalone mode and as part of Helios framework is provided in Section 5. A preliminary performance analysis of mStrand is performed in Section 6, where the strand solver is compared with the performance of established unstructured and structured grid solvers. Concluding remarks are offered in Section 7.

2. Multi-strand (mStrand) flow solver

2.1. Governing equations

The Reynolds-averaged Navier–Stokes (RANS) equations in a general moving coordinate system in three dimensions is solved in mStrand. Turbulence closure is accomplished with the Spalart–Allmaras (SA) model [19]. The RANS-SA equations may be expressed as

$$\frac{\partial Q}{\partial t} + \frac{\partial F_j}{\partial x_j} - \frac{\partial F_j^v}{\partial x_j} = S, \tag{1}$$

where the conserved variables, Q, inviscid fluxes, F_j, viscous fluxes, F_j^v, and source term, S, are defined as

$$Q = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \\ \rho \tilde{\nu} \end{pmatrix}, \quad F_j = \begin{pmatrix} \rho(u_j - u_j^b) \\ \rho u_i(u_j - u_j^b) + p\delta_{ij} \\ \rho e(u_j - u_j^b) + p u_j \\ \rho \tilde{\nu}(u_j - u_j^b) \end{pmatrix},$$

$$F_j^v = \begin{pmatrix} 0 \\ \sigma_{ij} \\ \sigma_{ij} u_i - q_j \\ \frac{\eta}{\sigma} \frac{\partial \tilde{\nu}}{\partial x_j} \end{pmatrix}, \quad S = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \mathcal{P} - \mathcal{D} + C_{b2} \rho \frac{\partial \tilde{\nu}}{\partial x_k} \frac{\partial \tilde{\nu}}{\partial x_k} \end{pmatrix}. \tag{2}$$

Here, ρ is the density, u_i is the Cartesian velocity vector, u_i^b is the mesh velocity vector, e is the total energy per unit mass, $\tilde{\nu}$ is the turbulence working variable, p is the pressure, σ_{ij} is the deviatoric stress tensor, q_j is the heat flux vector, and η/σ is the turbulent diffusion coefficient. The turbulent source term consists of a production term, $\mathcal{P}$, and a destruction term $\mathcal{D}$. The stress tensor is defined as

$$\sigma_{ij} = 2(\mu + \mu_T) s_{ij}, \tag{3}$$

where μ is the dynamic viscosity, μ_T is the turbulent viscosity, and s_{ij} is the rate of strain tensor, defined as

$$s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}. \tag{4}$$