Computers & Fluids 80 (2013) 310–319

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

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

Unicorn: Parallel adaptive finite element simulation of turbulent flow and fluid–structure interaction for deforming domains and complex geometry

Johan Hoffman, Johan Jansson, Rodrigo Vilela de Abreu, Niyazi Cem Degirmenci, Niclas Jansson *, Kaspar Müller, Murtazo Nazarov, Jeannette Hiromi Spühler

Computational Technology Laboratory, School of Computer Science and Communication, KTH, Royal Institute of Technology, SE-100 44 Stockholm, Sweden

ARTICLE INFO

Article history:
Received 15 September 2011
Received in revised form 4 February 2012
Accepted 6 February 2012
Available online 22 February 2012

Keywords:
Unicorn
DOLFIN
FEniCS
Parallel adaptive finite element method
Open source software
Turbulent flow
Fluid–structure interaction
Complex geometry
Deforming domain

ABSTRACT

We present a framework for adaptive finite element computation of turbulent flow and fluid–structure interaction, with focus on general algorithms that allow for complex geometry and deforming domains. We give basic models and finite element discretization methods, adaptive algorithms and strategies for efficient parallel implementation. To illustrate the capabilities of the computational framework, we show a number of application examples from aerodynamics, aero-acoustics, biomedicine and geophysics. The computational tools are free to download open source as *Unicorn*, and as a high performance branch of the finite element problem solving environment *DOLFIN*, both part of the FEniCS project.

1. Introduction

In this paper we present our work on finite element simulation of turbulent flow and fluid–structure interaction for complex geometry and deforming domains, in the form of a computational framework with focus on adaptive algorithms for parallel computer architectures. We present our open source implementation of the algorithms in the form of a high performance branch of the finite element problem solving environment *DOLFIN* [1,2] and the unified continuum mechanics solver *Unicorn* [3]. DOLFIN and Unicorn are parts of the FEniCS project [4], with the goal to automate the scientific software process by relying on general implementations and code generation, for robustness and to enable high speed of software development. To illustrate the capacity of the computational tools, we present snapshots from a number of application projects, together with parallel performance results.

We target a large family of problems of continuum mechanics, including incompressible and compressible flow, and fluid–structure interaction, described by (i) conservation of mass, (ii) balance of momentum, and (iii) conservation of energy, together with constitutive laws for fluids and solids. The basic laws (i)–(iii) take a generic form and can thus be handled by a general discretization strategy, with a common implementation. The constitutive laws, on the other hand, are specific for each problem, and are treated as data. The current implementation of Unicorn consists of a small collection of finite element solver implementations for continuum mechanics models, to the most part based on general code with only a minimum of code dedicated to the particular model.

Simulation of turbulent flow is based on the General Galerkin (G2) [5] framework, where the effect of unresolved scales of turbulence is modeled by numerical dissipation from residual based stabilization, similar to an Implicit Large Eddy Simulation (LES) [6]. Similarly, unresolved shocks and discontinuities in compressible flow are treated by shock capturing stabilization.

Finite element methods have the benefit of a firm mathematical foundation which enables quantitative a posteriori error analysis, which forms the basis for adaptive methods where the computational mesh is modified to satisfy certain error tolerances guided by a posteriori error indicators. Fluid–structure interaction and deforming domains are treated by moving mesh algorithms and Arbitrary Lagrangian–Eulerian (ALE) methods. Parallel efficiency is essential, where adaptive methods for unstructured meshes pose

* Corresponding author.
E-mail addresses: jhoffman@csc.kth.se (J. Hoffman), jjan@csc.kth.se (J. Jansson), rvda@csc.kth.se (R. Vilela de Abreu), ncde@csc.kth.se (N.C. Degirmenci), njansson@csc.kth.se (N. Jansson), kasparm@csc.kth.se (K. Müller), murtazo@csc.kth.se (M. Nazarov), spuhler@csc.kth.se (J.H. Spühler).

0045-7930/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compfluid.2012.02.003

particular challenges, and we here point out key algorithms of our software implementation.

The Unicorn and DOLFIN branches focused on high performance parallel computing were released open source in 2010, initially with only the adaptive incompressible flow solver fully parallelized [7]. Recently, also the fluid–structure interaction solver was parallelized, with the compressible flow solver expected to be fully parallelized in 2012. Thus the Unicorn computational framework offers open source high performance simulation tools for a wide range of applications in computational mechanics.

The outline of the paper is the following: first we recall the basic continuum mechanics models we target, including turbulent flow and fluid–structure interaction, we then present the basic finite element algorithms underlying adaptivity and moving meshes, with particular focus on distributed parallel algorithms. We conclude the paper by a number of applications and a discussion of future work.

2. Basic models

2.1. The Navier–Stokes equations

We consider fluid enclosed in a fixed, open domain Ω in three-dimensional space $\mathbb{R}^3$ with boundary Γ over a time interval $I = [0, \hat{t}]$ with initial time zero and final time $\hat{t}$.

We seek the *density* ρ, *momentum* $\boldsymbol{m} = \rho\boldsymbol{u}$, with $\boldsymbol{u} = (u_1, u_2, u_3)$ the *velocity*, and the *total energy* E as functions of $(\boldsymbol{x}, t) \in Q \equiv \Omega \times I$, where $\boldsymbol{x} = (x_1, x_2, x_3)$ denotes the coordinates in $\mathbb{R}^3$. The equations for $\hat{\boldsymbol{u}} \equiv (\rho, \boldsymbol{m}, E)$ read:

$$\begin{aligned}
&\partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0,\\
&\partial_t \boldsymbol{m} + \nabla \cdot (\boldsymbol{m} \otimes \boldsymbol{u} + \mathbb{I} p) = \boldsymbol{g} + \nabla \cdot (2\mu\varepsilon(\boldsymbol{u}) + \lambda(\nabla \cdot \boldsymbol{u})\mathbb{I}),\\
&\partial_t E + \nabla \cdot (E\boldsymbol{u} + p\boldsymbol{u}) = \nabla \cdot ((2\mu\varepsilon(\boldsymbol{u}) + \lambda(\nabla \cdot \boldsymbol{u})) \cdot \boldsymbol{u} + \kappa \nabla T),\\
&\hat{\boldsymbol{u}}(\cdot, 0) = \hat{\boldsymbol{u}}^0,
\end{aligned} \tag{1}$$

where $p = p(\boldsymbol{x}, t)$ is the *pressure* of the fluid, $\otimes$ denotes the tensor product, $\mathbb{I}$ denotes the identity matrix in $\mathbb{R}^3$, $\partial_t = \partial/\partial t$, $\boldsymbol{g} = (\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3)$ is a given *volume force* (e.g. gravity) acting on the fluid, $\hat{\boldsymbol{u}}^0 = \hat{\boldsymbol{u}}^0(\boldsymbol{x})$ represents initial conditions,

$$\varepsilon(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right),$$

is the strain rate tensor, and $\kappa \geqslant 0$ the thermal conduction parameter. The viscosity parameters are assumed to satisfy conditions $\mu > 0$, $\lambda + 2\mu > 0$. For simplicity we sometimes use the approximation $\lambda = 0$.

Further, the total energy $E = k + e$, where $k = \rho|\boldsymbol{u}|^2/2$ is the *kinetic energy*, with $|\boldsymbol{u}|^2 \equiv \boldsymbol{u}_1^2 + \boldsymbol{u}_2^2 + \boldsymbol{u}_3^2$, and $e = \rho T$ is the *internal energy* with T the *temperature* scaled so that $c_v = 1$, where c_v is the heat capacity under constant volume.

For very high Reynolds numbers we may approximate the Navier–Stokes equations by inviscid flow, where the viscosity coefficients and thermal conductivity are zero, resulting in the Euler equations:

$$\begin{aligned}
&\partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0 \quad \text{in } Q,\\
&\partial_t \boldsymbol{m} + \nabla \cdot (\boldsymbol{m} \otimes \boldsymbol{u} + \mathbb{I} p) = \boldsymbol{g} \quad \text{in } Q,\\
&\partial_t E + \nabla \cdot (E\boldsymbol{u} + p\boldsymbol{u}) = 0 \quad \text{in } Q,\\
&\hat{\boldsymbol{u}}(\cdot, 0) = \hat{\boldsymbol{u}}^0 \qquad \text{in } \Omega.
\end{aligned} \tag{2}$$

The number of unknowns including the pressure is six but there are only five equations in (1), (2); for a *perfect gas*, we close the system with the following *state equation*:

$$p = (\gamma - 1)e = (\gamma - 1)\rho T = (\gamma - 1)(E - \rho|\boldsymbol{u}|^2/2),$$

expressing the pressure p as a function of density ρ and temperature T, where $\gamma = c_p$ is the *adiabatic index* with c_p the heat capacity under constant pressure, and $(\gamma - 1)$ is the *gas constant*.

For a perfect gas, the *speed of sound* c is given by $c^2 = \gamma(\gamma - 1)T$, and the *Mach number* is defined as $M = |\boldsymbol{u}|/c$, with $\boldsymbol{u}$ the velocity of the gas.

2.2. Incompressible flow

For low Mach numbers one may use the approximation of incompressible flow, corresponding to a divergence free condition on the velocity. The density may be variable, or for small density variations be approximated as constant ρ_0, leaving only the momentum equation and the divergence free condition. We then have the following equations: find $\hat{\boldsymbol{u}} \equiv (\boldsymbol{u}, p)$ such that:

$$\begin{aligned}
&\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p = \boldsymbol{g} + 2\nu\nabla \cdot \varepsilon(\boldsymbol{u}) \quad \text{in } Q,\\
&\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } Q,\\
&\hat{\boldsymbol{u}}(\cdot, 0) = \hat{\boldsymbol{u}}^0 \quad \text{in } \Omega.
\end{aligned} \tag{3}$$

with $\nu = \mu/\rho_0$ the kinematic viscosity.

2.3. Turbulent flow

Direct Numerical Simulation (DNS) of turbulent flow is not possible in the general cases that we target, of high Reynolds numbers and complex geometry. In a Large Eddy Simulation (LES) [6] only the largest scales of the flow are resolved, leaving the smallest, unresolved turbulent scales to be taken into account only through a subgrid model. Similarly, shocks and discontinuities in compressible flow can be left unresolved to be modeled through shock capturing regularization, and the effect of turbulent boundary layers can be approximated by wall shear stress models [8].

Our approach to simulation of turbulent flow is based on the General Galerkin (G2) method, where numerical stabilization based on the residual of the equations models the effect of unresolved features in the flow, from turbulence to shocks [5]. The effect of unresolved turbulent boundary layers is modeled by a skin friction model for the wall shear stress.

2.4. Unified continuum fluid–structure interaction

For robustness we choose a monolithic approach to fluid–structure interaction (FSI), which we derive from the basic conservation laws. We here also seek a phase function θ and introduce the unified Cauchy stress σ for all phases. The incompressible unified continuum fluid–structure model [9] reads:

$$\begin{aligned}
&\rho(\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}) + \nabla \cdot \sigma = \boldsymbol{g} \quad \text{in } Q,\\
&\nabla \cdot \boldsymbol{u} = 0 \quad \text{in } Q,\\
&\partial_t \theta + (\boldsymbol{u} \cdot \nabla)\theta = 0 \quad \text{in } Q,\\
&\hat{\boldsymbol{u}}(\cdot, 0) = \hat{\boldsymbol{u}}^0 \quad \text{in } \Omega,
\end{aligned}$$

where the phase function θ defines the solid and fluid domains by:

$$\begin{aligned}
&\Omega_s(t) = \{x : x \in \Omega, \theta(x, t) = 0\},\\
&\Omega_f(t) = \{x : x \in \Omega, \theta(x, t) = 1\}.
\end{aligned}$$

For example, we can define a Newtonian fluid and an incompressible Neo-Hookean solid (here in stress rate form):

$$\begin{aligned}
&\sigma = -\sigma_D + pI,\\
&\sigma_D = \theta\sigma_f + (1 - \theta)\sigma_s,\\
&\sigma_f = 2\mu_f\varepsilon(\boldsymbol{u}),\\
&\partial_t \sigma_s = 2\mu_s\varepsilon(\boldsymbol{u}) + \nabla\boldsymbol{u}\sigma_s + \sigma_s\nabla\boldsymbol{u}^\top,
\end{aligned}$$

where the subscript s denotes solid and f denotes fluid.

The FSI problem is thus treated as a multiphase flow problem, where a phase function θ identifies the solid and fluid, respectively. Typically we let the finite element mesh track the solid