FLOWPRO - MULTIPURPOSE CFD SOFTWARE WRITTEN IN JAVA

A. Pecka[1], O. Bublík[2], J. Vimmr[3]

Abstract: FlowPro is a multipurpose open-source CFD software, which is being developed by the researchers of the Department of Mechanics of the Faculty of Applied Sciences at the University of West Bohemia. FlowPro is designed to solve various non-linear systems of hyperbolic-parabolic partial differential equations in one, two or three dimensions. It is also capable of solving fluid-structure interaction problems. The computational core is based on the discontinuous Galerkin method, a numerical method capable of obtaining high-order accurate solutions. In order to achieve a high performance the distributed computing is also possible. FlowPro is implemented in the Java programming language in order to simplify the use on various platforms. A notable strength of FlowPro is the possibility to implement new mathematical models via a well documented API. Another advantage is the option to automate complex computations via the Matlab/Python interface.

Keywords: FlowPro; discontinuous Galerkin method; CFD

1 Introduction

FlowPro is a software designed to solve a wide range of non-linear hyperbolic-parabolic partial differential equations in one, two or three dimensions. Let $\Omega \subset \mathbb{R}^D$ be a time-dependent domain in general and let D be the dimension. We consider the following system of M equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}_\alpha}{\partial x_\alpha}(\boldsymbol{u}, \partial \boldsymbol{u}) = \boldsymbol{s}(\boldsymbol{u}, \partial \boldsymbol{u}), \tag{1}$$

where $\alpha = 1, \dots, D$ is the summation index, $\boldsymbol{u}$ is the unknown vector, $\boldsymbol{f}_\alpha$ is the total flux and $\boldsymbol{s}$ is the source term. The total flux can be separated into the advection and diffusion as follows

$$\boldsymbol{f}_\alpha(\boldsymbol{u}, \partial \boldsymbol{u}) = \boldsymbol{f}_\alpha^A(\boldsymbol{u}) - \boldsymbol{f}_\alpha^D(\boldsymbol{u}, \partial \boldsymbol{u}). \tag{2}$$

The advection and diffusion terms are dealt with differently from the numerical point of view. The concrete mathematical model is determined by the flux functions. FlowPro contains an API called FlowProAPI, which enables the user of FlowPro to define new or modify existing mathematical models by implementing the advection and diffusion fluxes and the production term as well as boundary and initial conditions according to the well documented interface.

The main purpose of FlowPro is to simulate compressible fluid flow problems, hence there are a number of predefined fluid flow mathematical models, e.g. compressible Euler and Navier-Stokes equations, the shallow water equations and the ideal magnetohydrodynamics equations. FlowPro is also capable of solving problems with moving mesh and the fluid-structure interaction problems. For this reason, FlowProAPI contains an interface for the definition of the structure model. The structure model consisting of particles connected by springs and dampers is already available in FlowPro, but just as before a user can defined new structure models.

The underlying algorithm for the solution of the system (1) is based on an implicit discontinuous Galerkin scheme, which is to be described in the upcoming section. The discontinuous Galerkin method [1, 2] is a substantially stable and robust method that naturally offers an arbitrary high order of spatial accuracy by choosing basis polynomial of appropriate order.

[1] Aleš Pecka; KME, FAV, University of West Bohemia in Pilsen; pecka@kme.zcu.cz
[2] Ondřej Bublík; NTIS, FAV, University of West Bohemia in Pilsen; obublik@ntis.zcu.cz
[3] Jan Vimmr; NTIS, FAV, University of West Bohemia in Pilsen; jvimmr@ntis.zcu.cz

2 Implicit discontinuous Galerkin scheme

Let $\mathcal{T} = \{\Omega_1, \Omega_2, \dots, \Omega_K\}$ be the partition of the domain Ω. Note that the domain $\Omega = \Omega(t)$ and the individual elements $\Omega_k = \Omega_k(t)$ are time dependent in general although we do not always explicitly write that. The discontinuous Galerkin discretisation [1, 2] is based on replacing the infinite-dimensional space with its finite-dimensional subspace, which we choose as

$$S_h = \{w \in L^2(\Omega) : \; w|_{\Omega_k} \in P^q(\Omega_k), \; \forall \Omega_k \in \mathcal{T}\}, \tag{3}$$

where $P^q(\Omega_k)$ is the space of polynomial of degree up to q on Ω_k. Taking the dot product of each side of (1) with a test function $\boldsymbol{\phi} \in [S_h]^M$, integrating it over a mesh element Ω_k and applying the divergence theorem yields

$$\int_{\Omega_k} \frac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{\phi} \, \mathrm{d}\Omega - \int_{\Omega_k} \boldsymbol{f}_\alpha \cdot \frac{\partial \boldsymbol{\phi}}{\partial x_\alpha} \mathrm{d}\Omega + \oint_{\partial \Omega_k} \mathcal{F}(\boldsymbol{u}^\pm, \, \partial \boldsymbol{u}^\pm, \, \vec{n}) \cdot \boldsymbol{\phi}^- \, \mathrm{d}S - \int_{\Omega_k} \boldsymbol{s} \cdot \boldsymbol{\phi} \, \mathrm{d}\Omega = 0. \tag{4}$$

In case of the discontinuous Galerkin method we allow discontinuities at the boundary of each element Ω_k and so the value in not known on $\partial\Omega_k$. Therefore, we approximate the normal total flux $\boldsymbol{f}_\alpha(\boldsymbol{u}) n_\alpha$ by the total numerical flux $\mathcal{F}$ on $\partial\Omega_k$. The advection and diffusion parts are approximated separately as

$$\begin{aligned} \boldsymbol{f}_\alpha^A(\boldsymbol{u}) n_\alpha &\approx \mathcal{F}^A(\boldsymbol{u}^\pm, \vec{n}), \\ \boldsymbol{f}_\alpha^D(\boldsymbol{u}, \partial \boldsymbol{u}) n_\alpha &\approx \mathcal{F}^D(\boldsymbol{u}^\pm, \partial \boldsymbol{u}^\pm, \vec{n}), \end{aligned}$$

where $\mathcal{F} = \mathcal{F}^A - \mathcal{F}^D$. We use the Lax-Friedrich flux as the advective numerical flux and for the diffusion part we use the interior penalty method.

Both the test function $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{x}, \, t)$ and the elements $\Omega_k = \Omega_k(t)$ are time dependent in general. Hence, we need to employ the Reynolds transport theorem to bring the time derivative outside the integral. Reynolds transport theorem along with the assumption that the test function moves with the mesh velocity yields

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega_k} \boldsymbol{u} \cdot \boldsymbol{\phi} \, \mathrm{d}\Omega - \int_{\Omega_k} (\boldsymbol{f}_\alpha - V_\alpha \boldsymbol{u}) \cdot \frac{\partial \boldsymbol{\phi}}{\partial x_\alpha} \mathrm{d}\Omega - \int_{\Omega_k} \boldsymbol{s} \cdot \boldsymbol{\phi} \, \mathrm{d}\Omega \\ + \oint_{\partial \Omega_k} \Big[\mathcal{F}(\boldsymbol{u}^\pm, \, \partial \boldsymbol{u}^\pm, \, \vec{n}) - V_\alpha n_\alpha \{\!\{\boldsymbol{u}\}\!\} \Big] \cdot \boldsymbol{\phi}^- \, \mathrm{d}S = 0, \end{aligned} \tag{5}$$

where $\{\!\{\boldsymbol{u}\}\!\} = \frac{1}{2}(\boldsymbol{u}^- + \boldsymbol{u}^+)$ denotes the average value. It remains to choose basis functions φ_i^k for each element Ω_k, substitute the basis functions for the test function $\boldsymbol{\phi}$ and expand the solution as a linear combination of basis functions

$$\boldsymbol{u}_h(\boldsymbol{x}, t) \Big|_{\Omega_k} = \sum_j \mathbf{u}_j^{(k)}(t) \; \varphi_j^{(k)}(\boldsymbol{x}, t). \tag{6}$$

In FlowPro the Lagrange basis functions are used.

The semi-discrete scheme can be written in the matrix form as the following system of ordinary differential equations

$$\frac{\mathrm{d}(\mathbf{MU})}{\mathrm{d}t} = \mathbf{R}(\mathbf{U}), \tag{7}$$

where $\mathbf{M}$ is the global mass matrix, $\mathbf{U}$ is the global vector of basis coefficients and $\mathbf{R}$ is the global residual vector. The semi-discrete scheme is discretised by the backward difference formula (BDF). Applying BDF of order R results in the non-linear system of equations

$$\frac{1}{\Delta t_n} \sum_{r=0}^{R} a_r (\mathbf{MU})^{n+1-r} - \mathbf{R}(\mathbf{U}^{n+1}) = \mathbf{0}, \tag{8}$$

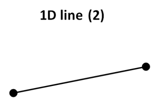
with given coefficients $a_0, \ldots, a_R$. Applying Newton's method to the non-linear system of algebraic equation (8) we acquire the following iterative procedure

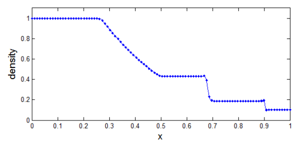
$$\overbrace{\left[\frac{a_0}{\Delta t_n}\mathbf{M}_s^{n+1} - \frac{\partial \mathbf{R}}{\partial \mathbf{U}}(\mathbf{U}_s^{n+1})\right]}^{\mathbf{A}} \overbrace{\Delta\mathbf{U}}^{\boldsymbol{x}} = \overbrace{\mathbf{R}(\mathbf{U}_s^{n+1}) - \frac{1}{\Delta t_n}\left[\sum_{r=1}^{R} a_r(\mathbf{MU})^{n+1-r} + a_0(\mathbf{MU})_s^{n+1}\right]}^{\boldsymbol{b}},$$
$$\mathbf{U}_{s+1}^{n+1} = \mathbf{U}_s^{n+1} + \Delta\mathbf{U}, \tag{9}$$

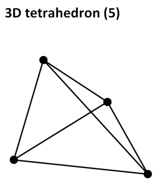
which is initiated by $\mathbf{U}_0^{n+1} = \mathbf{U}^n$. The linear system (9) is solved by the GMRES algorithm with the block diagonal preconditioner.

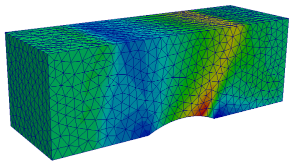
3 Implementation

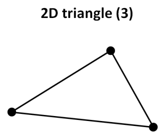
One of the strengths of FlowPro is that the algorithm is written in a hight level programming language, Java to be specific, using object oriented programming including inheritance in many occasions. This makes the solver of FlowPro very flexible. For instance the mathematical model or the structural model for the FSI simulation can be defined by the users with the aid of FlowProAPI. A couple of basic fluid and structure models are available in FlowPro. Furthermore, the solver is capable of solving problems in one, two or three dimensions without code duplicity. This flexibility is heavily dependent on Java's inheritance. Below are examples of available mesh elements in one, two and three dimensions each with a sample computation.

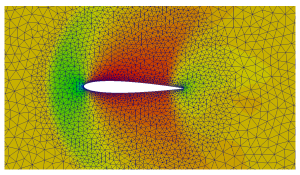


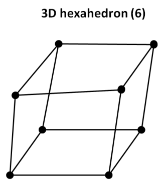


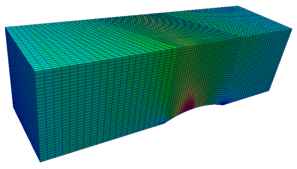


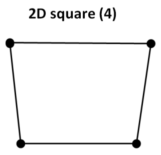


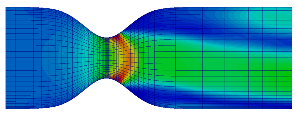


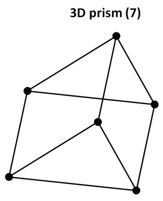


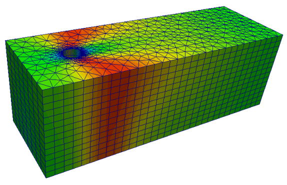












Each element has its associated quadrature points. New mesh elements can also be defined, e.g. mesh elements with curved boundary.

FlowPro is capable of utilising all of the CPU's computational power by employing multiple threads for computing. In particular, the subject for parallelisation is the linear solver. A majority of the computational time during execution of an iterative linear solver is spent on matrix-vector multiplications $\mathbf{Av}$, where $\mathbf{A}$ is a sparse matrix that appears in the linear system and $\mathbf{v}$ is a given column vector. The parallel implementation lies in performing the products of rows of $\mathbf{A}$ with the vector $\mathbf{v}$ in parallel. Other vector operations involved in the algorithm are parallelised in the same manner. This type of implementation is suitable to be performed among CPU cores of each computational node since the CPU cores share memory. On the other hand, inefficient for parallel computing among nodes in the computer network, as the data transfer among nodes would be too frequent.

FlowPro also supports distributed computing, that is parallel computing on the level of a computer network. For this purpose the overlapping Schwarz method [3] is used. For this purpose an analogy to an MPI library was implemented using Java socket. It is designed not only for computer clusters, but also for a heterogeneous computer networks.

4 Example simulations

As it has been already mentioned, FlowPro has a couple of mathematical models available and on the top of that it supports creation of new models. Below are sample simulations for a couple of mathematical models obtained by FlowPro.

Shallow water equations

The system of shallow water equations reads

$$\begin{aligned} \frac{\partial h}{\partial t} + \frac{\partial (h u_j)}{\partial x_j} &= 0, \\ \frac{\partial h u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(h u_i u_j + \frac{g h^2}{2}\delta_{ij} \right) &= 0. \end{aligned} \tag{10}$$

We choose a water drop simulation as an example. The water level at various times is plotted in Figure 1.

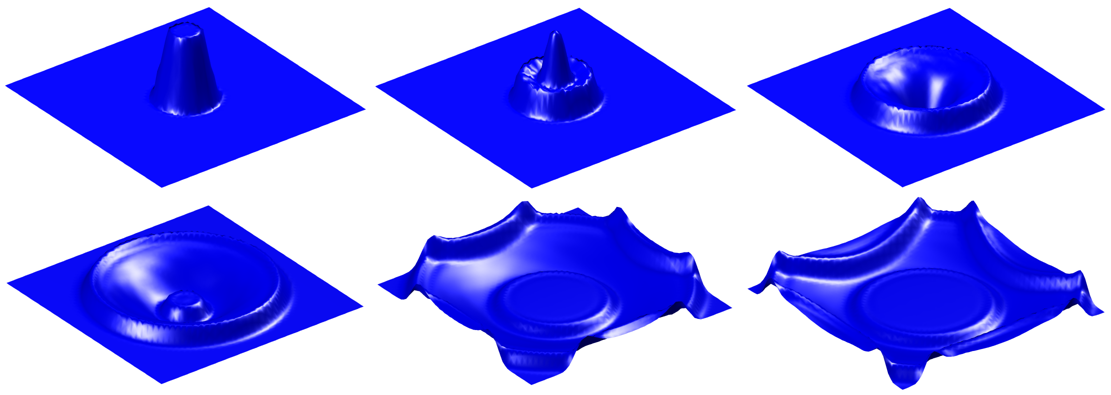


Figure 1: Shallow water equations - water drop simulation.

Euler/Navier-Stokes equations

The system of compressible Euler/Navier-Stokes equations can be written in the dimensionless form as follows

$$\begin{aligned} \frac{\partial \varrho}{\partial t} + \frac{\partial (\varrho u_j)}{\partial x_j} &= 0, \\ \frac{\partial \varrho u_i}{\partial t} + \frac{\partial}{\partial x_j}(\varrho u_i u_j + p\delta_{ij}) &= \frac{1}{\text{Re}}\frac{\partial \tau_{ij}}{\partial x_j}, \\ \frac{\partial E}{\partial t} + \frac{\partial (E+p)u_j}{\partial x_j} &= \frac{1}{\text{Re}}\frac{\partial}{\partial x_j}\left[\tau_{ij} u_i + \frac{\kappa}{\kappa - 1}\frac{1}{Pr}\frac{\partial}{\partial x_j}\left(\frac{p}{\varrho}\right)\right], \end{aligned} \tag{11}$$

which is completed by the following relations

$$p = (\kappa - 1)\left[E - \frac{1}{2}\varrho\sum_{i=1}^{2} u_i^2\right], \tag{12}$$

$$\tau_{ij} = \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}. \tag{13}$$

Here we choose three sample simulations, namely an inviscid flow around MIG21, Kármán vortex street, which is viscous internal flow and finally a viscous flow around NACA0012 aerofoil. These examples are respectively show in Figures 2, 3 and 4.

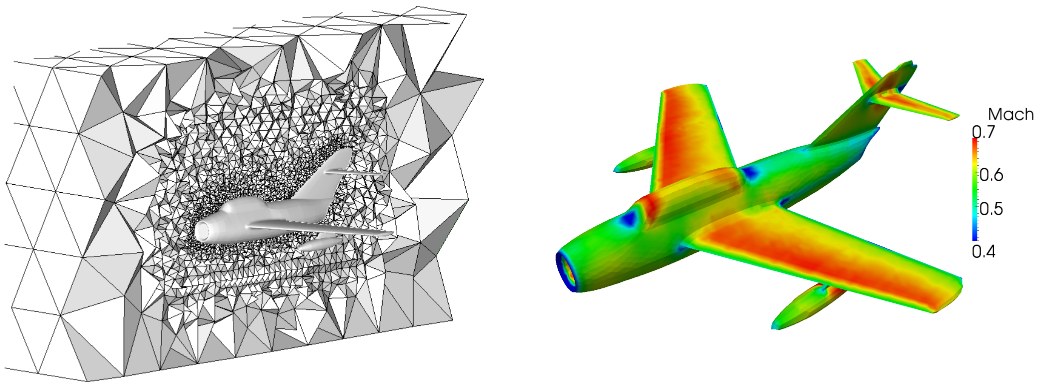


Figure 2: Euler equations - flow around MIG15 with the free-stream Mach number 0.5.

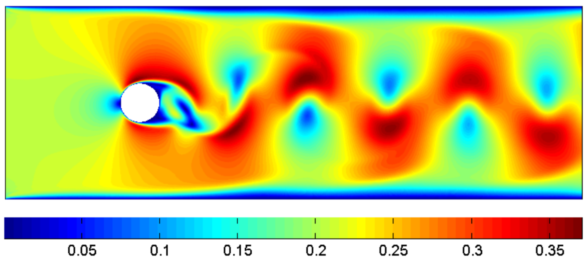


Figure 3: Navier-Stokes equations - Kármán vortex street with the Reynolds number 3000 and free-stream Mach number 0.2.

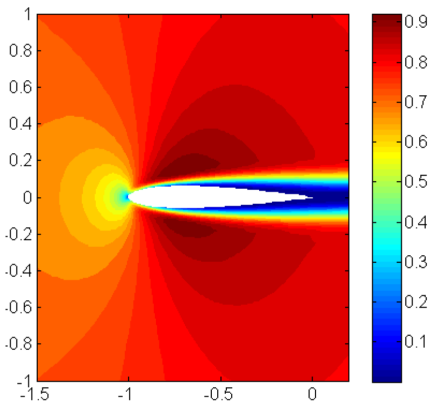


Figure 4: Navier-Stokes equations - flow around NACA0012 aerofoil with the Reynolds number 5000 and free-stream Mach number 0.75.

Ideal magnetohydrodynamics equations

The system of ideal magnetohydrodynamics equations reads

$$\begin{aligned}
\frac{\partial \varrho}{\partial t} + \frac{\partial(\varrho u_j)}{\partial x_j} &= 0,\\
\frac{\partial \varrho u_i}{\partial t} + \frac{\partial}{\partial x_j}(\varrho u_i u_j + p\delta_{ij} + \frac{1}{2}B_k B_k - B_i B_j) &= 0,\\
\frac{\partial B_i}{\partial t} + \frac{\partial}{\partial x_j}(u_i B_j - B_i u_j) &= 0,\\
\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_j}\left[(E + p + \frac{1}{2}B_k B_k)u_j - B_j(u_k B_k)\right] &= 0,\\
\frac{\partial B_j}{\partial x_j} &= 0.
\end{aligned} \tag{14}$$

Here we choose a classical magnetohydrodynamics rotor test problem [4], where the initial condition is a rotating cylinder with density ten times that of the surrounding environment. A snapshot of the solution at time $t = 0.15$ is shown in Figure 5.

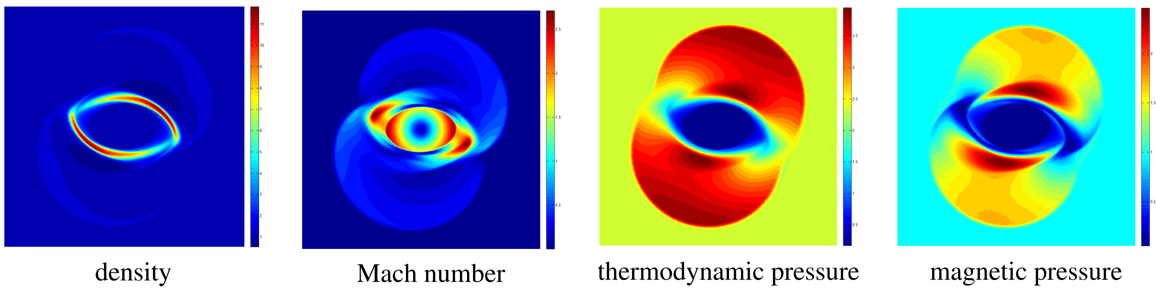


Figure 5: Ideal magnetohydrodynamics equations - rotor test problem at time $t = 0.15$.

Acknowledgement

This work was supported by the project SGS-2016-038.

References

[1] COCKBURN, Bernardo and Chi-Wang SHU. The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. SIAM Journal on Numerical Analysis. 1998, 35(6), 2440-2463.

[2] DOLEJŠÍ, V., M. HOLÍK and J. HOZMAN. Efficient solution strategy for the semi-implicit discontinuous Galerkin discretization of the Navier-Stokes equations. Journal of Computational Physics. 2011, 230(11), 4176-4200.

[3] SCHWARZ, H. A. Über einen Grenzübergang durch alternierendes Verfahren. Vierteljahrsschrift der Naturforschenden Gesellschaft, 1870, 15, 272-286.

[4] TÓTH, Gábor. The $\Delta \cdot B = 0$ Constraint in Shock-Capturing Magnetohydrodynamics Codes. Journal of Computational Physics. 2000, 161(2), 605-652.