INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **41**:519–535 (DOI: 10.1002/fld.450)

Application of the method of lines to unsteady compressible Euler equations

Pavel Šolín[1,*,†] and Karel Segeth[2]

[1] *TICAM, The University of Texas at Austin, ACES 6.330, C0200, Austin, TX 78731, U.S.A.*
[2] *Mathematical Institute of the Academy of Sciences of the Czech Republic, Prague*

SUMMARY

In the sense of method of lines, numerical solution of the unsteady compressible Euler equations in 1D, 2D and 3D is split into three steps: First, space discretization is performed by the first-order finite volume method using several approximate Riemann solvers. Second, smoothness and Lipschitz continuity of RHS of the arising system of ordinary dimensional equations (ODEs) is analysed and its solvability is discussed. Finally, the system of ODEs is integrated in time by means of implicit and explicit higher-order adaptive schemes offered by ODE packages ODEPACK and DDASPK, by a backward Euler scheme based on the linearization of the RHS and by higher-order explicit Runge–Kutta methods. Time integrators are compared from several points of view, their applicability to various types of problems is discussed, and 1D, 2D and 3D numerical examples are presented. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: compressible Euler equations; finite volume schemes; method of lines; semi-discrete problem; solvability and uniqueness; ordinary differential equations

1. INTRODUCTION

Let the domain $\Omega \subset \mathbb{R}^d$ occupied by the fluid be bounded and have a piecewise smooth boundary $\partial\Omega$. The d-dimensional compressible Euler equations consist of the continuity equation, d Euler momentum equations and energy equation. We will consider them in the space–time cylinder $Q_T = \Omega \times (0, T)$ $(T > 0)$ and write them in the form

$$\frac{\partial \boldsymbol{w}}{\partial t} + \sum_{s=1}^{d} \frac{\partial \boldsymbol{f}_s(\boldsymbol{w})}{\partial x_s} = 0 \tag{1}$$

* Correspondence to: P. Solin, TICAM, The University of Texas at Austin, ACES 6.330, 1 University Station, Austin, TX 78712, U.S.A.
† E-mail: solin@ticam.utexas.edu

Contract/grant sponsor: Grant Agency of the Czech Republic; contract/grant number: 201/01/1200.

Received 15 May 2000
Revised 7 October 2002

where

$$\boldsymbol{w} = \begin{pmatrix} \rho \\ \rho v_1 \\ \vdots \\ \rho v_d \\ e \end{pmatrix}, \quad \boldsymbol{f}_s(\boldsymbol{w}) = \begin{pmatrix} \rho v_s \\ \rho v_s v_1 + \delta_{s1} p \\ \vdots \\ \rho v_s v_d + \delta_{sd} p \\ (e+p) v_s \end{pmatrix}, \quad s = 1, \ldots, d \tag{2}$$

and

$$e = \frac{p}{\kappa - 1} + \frac{1}{2} \rho |\boldsymbol{v}|^2 \tag{3}$$

We use the standard notation: t—time, $x_1, \ldots, x_d$—Cartesian co-ordinates in $\mathbb{R}^d$, $\boldsymbol{x} = (x_1, \ldots, x_d)$, ρ—density, $\boldsymbol{v} = (v_1, \ldots, v_d)$—velocity vector with components $v_1, \ldots, v_d$ in the directions $x_1, \ldots, x_d$, p—pressure, e—total energy, δ_{sj}—Kronecker delta, $\kappa > 1$—Poisson adiabatic constant. Due to physical reasons it is also suitable to require $\rho > 0$, $p > 0$. The functions $\boldsymbol{f}_s$, $s = 1, \ldots, d$ are called inviscid (Euler) fluxes and are defined in the set

$$\mathscr{Q} = \left\{ (w_1, \ldots, w_{d+2}) \in \mathbb{R}^{d+2};\ w_1 > 0, w_{d+2} - \frac{w_2^2 + \cdots + w_{d+1}^2}{2 w_1} > 0 \right\} \tag{4}$$

System (1) and (3) is equipped with the initial conditions

$$\boldsymbol{w}(\boldsymbol{x}, 0) = \boldsymbol{w}^0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \tag{5}$$

and a set of boundary conditions which will be specified later.

2. SEMI-DISCRETE PROBLEM

Integrating Equation (1) over a finite volume Ω_i, $1 \leqslant i \leqslant N$, where N denotes the number of finite volumes, and using Green's theorem and the *rotational invariance* of the Euler equations (for a detailed description, see e.g. Reference [1]), we obtain

$$\begin{aligned} \int_{\Omega_i} \frac{\partial \boldsymbol{w}}{\partial t}(\boldsymbol{x}, t)\, \mathrm{d}\boldsymbol{x} = &- \sum_{j \in S_{\text{int}}(i)} \int_{\partial \Omega_{ij}} \mathbb{T}_d^{-1}(v_{ij}) \boldsymbol{f}_1(\mathbb{T}_d(v_{ij}) \boldsymbol{w}(\boldsymbol{x}, t))\, \mathrm{d}S \\ &- \sum_{\alpha \in S_{\text{wall}}(i)} \int_{\partial \Omega_{i\alpha}} \mathbb{T}_d^{-1}(v_{i\alpha}) \boldsymbol{f}_1(\mathbb{T}_d(v_{i\alpha}) \boldsymbol{w}(\boldsymbol{x}, t))\, \mathrm{d}S \\ &- \sum_{i\alpha \in S_{io}(i)} \int_{\partial \Omega_{i\alpha}} \mathbb{T}_d^{-1}(v_{i\alpha}) \boldsymbol{f}_1(\mathbb{T}_d(v_{i\alpha}) \boldsymbol{w}(\boldsymbol{x}, t))\, \mathrm{d}S \end{aligned} \tag{6}$$

for all $t \in (0, T)$ and $1 \leqslant i \leqslant N$. The index set $S_{\text{int}}(i)$ contains indices of finite volumes adjacent to the finite volume Ω_i for all $1 \leqslant i \leqslant N$. The index sets $S_{\text{wall}}(i)$ and $S_{io}(i)$ contain indices of faces $\partial\Omega_{i\alpha}$ of the finite volume Ω_i lying in the parts of the boundary $\partial\Omega$ representing solid wall and inlet/outlet, respectively. Symbols $v_{ij}, v_{i\alpha} \in \mathbb{R}^d$ denote unit normal vectors to $\partial\Omega_{ij}, \partial\Omega_{i\alpha}$, respectively, pointing outward of Ω_i. The rotational matrices $\mathbb{T}_d(v)$, $v \in \mathbb{R}^d$, $|v| = 1$, $d = 1,2,3$, are defined as follows:

In 1D, the normal vector v has the form $v = \cos\alpha$ with either $\alpha = 0$ or π. The rotational matrix $\mathbb{T}_1(v)$ has the form

$$\mathbb{T}_1(v) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

In 2D, expressing v in spherical co-ordinates as $v = [\cos(\alpha), \sin(\alpha)]^{\mathrm{T}}$, $\alpha \in \langle -\pi, \pi \rangle$, we define

$$\mathbb{T}_2(v) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

In 3D, expressing v in spherical co-ordinates as

$$v = [\cos(\alpha)\cos(\beta), \sin(\alpha)\cos(\beta), \sin(\beta)]^{\mathrm{T}}, \quad \alpha \in \langle -\pi, \pi \rangle, \; \beta \in \left\langle -\frac{\pi}{2}, \frac{\pi}{2} \right\rangle \tag{9}$$

the matrix $\mathbb{T}_3(v)$ has the form

$$\mathbb{T}_3(v) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos(\alpha)\cos(\beta) & \sin(\alpha)\cos(\beta) & \sin(\beta) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & -\cos(\alpha)\sin(\beta) & -\sin(\alpha)\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Replacing in (6) the function $\boldsymbol{w}(\boldsymbol{x},t)$ with a function $\boldsymbol{w}_h(\boldsymbol{x},t)$ which is piecewise constant in space on the finite volumes $\Omega_1, \Omega_2, \ldots, \Omega_N$ and approximating the right-hand side integrals by the *numerical fluxes* H_{int}, H_{wall} and H_{io}, we obtain

$$\begin{aligned} \frac{\partial \boldsymbol{w}_h}{\partial t}(\boldsymbol{x},t)|_{\Omega_i} = & -\frac{1}{|\Omega_i|} \sum_{j \in S_{\text{int}}(i)} H_{\text{int}}(\boldsymbol{w}_h(\boldsymbol{x},t)|_{\Omega_i}, \boldsymbol{w}_h(\boldsymbol{x},t)|_{\Omega_j}, v_{ij}(\boldsymbol{x}), |\partial\Omega_{ij}|) \\ & -\frac{1}{|\Omega_i|} \sum_{\alpha \in S_{\text{wall}}(i)} H_{\text{wall}}(\boldsymbol{w}_h(\boldsymbol{x},t)|_{\Omega_i}, v_{i\alpha}(\boldsymbol{x}), |\partial\Omega_{i\alpha}|) \\ & -\frac{1}{|\Omega_i|} \sum_{\alpha \in S_{io}(i)} H_{io}(\boldsymbol{w}_h(\boldsymbol{x},t)|_{\Omega_i}, \boldsymbol{w}_h^B(\boldsymbol{x},t)|_{\partial\Omega_{i\alpha}}, v_{i\alpha}(\boldsymbol{x}), |\partial\Omega_{i\alpha}|) \end{aligned} \tag{11}$$

$t \in (0, T)$, $1 \leqslant i \leqslant N$, where $\nu_{i\alpha}$ denotes the unit normal vector to $\partial\Omega_{i\alpha}$ pointing outward of Ω. Symbols $|\Omega_i|$ and $|\partial\Omega_{i\alpha}|$ denote the d-dimensional measure of $|\Omega_i|$ and $(d-1)$-dimensional measure of $\partial\Omega_{i\alpha}$, respectively. The function $\boldsymbol{w}_h^B$ represents a boundary state corresponding to the inlet/outlet boundary face $\partial\Omega_{i\alpha}$ which will be discussed later.

2.1. Numerical flux

The numerical flux H_{int} through interior faces $\partial\Omega_{ij}$ has the well-known form

$$H_{\text{int}}(\boldsymbol{w}_i, \boldsymbol{w}_j, \nu_{ij}, |\partial\Omega_{ij}|) = |\partial\Omega_{ij}| \mathbb{T}_d^{-1}(\nu_{ij}) \boldsymbol{f}_R(\mathbb{T}_d(\nu_{ij})\boldsymbol{w}_i, \mathbb{T}_d(\nu_{ij})\boldsymbol{w}_j) \tag{12}$$

where $\boldsymbol{f}_R$ is the *approximate Riemann solver*. The numerical flux H_{wall} through the solid-wall faces $\partial\Omega_{i\alpha}$ has the form

$$H_{\text{wall}}(\boldsymbol{w}_i, \nu_{i\alpha}, |\partial\Omega_{i\alpha}|) = |\partial\Omega_{i\alpha}| [0, p(\boldsymbol{w}_i), 0, \dots, 0]^{\mathrm{T}} \tag{13}$$

where

$$p(\boldsymbol{w}_i) = (\kappa - 1)\left(w_{i,d+2} - \frac{w_{i,2}^2 + \cdots + w_{i,d+1}^2}{2w_{i,1}} \right) \tag{14}$$

denotes the pressure corresponding to the state $\boldsymbol{w}_i$. In this approximation, the impermeability condition $\boldsymbol{v} \cdot \nu_{i\alpha} = 0$ for solid walls was used, see e.g. Reference [2] for details. Finally, the numerical flux H_{io} through the inlet/outlet faces $\partial\Omega_{i\alpha}$ has the form

$$H_{io}(\boldsymbol{w}_i, \boldsymbol{w}_{i\alpha}, \nu_{i\alpha}, |\partial\Omega_{i\alpha}|) = |\partial\Omega_{i\alpha}| \mathbb{T}_d^{-1}(\nu_{i\alpha}) \boldsymbol{f}_1(\boldsymbol{w}^B(\mathbb{T}_d(\nu_{i\alpha})\boldsymbol{w}_i, \mathbb{T}_d(\nu_{i\alpha})\boldsymbol{w}_{i\alpha})) \tag{15}$$

where $\boldsymbol{w}_{i\alpha}$ represents the boundary state corresponding to the face $\partial\Omega_{i\alpha}$. The function $\boldsymbol{w}^B$ represents a switch deciding how much information is diffused from the boundary into the domain, depending on the number of outgoing characteristics of the approximate solution $\boldsymbol{w}_h$. Switches of this type are typical for hyperbolic systems. We use a function

$$\boldsymbol{w}^B(\boldsymbol{w}_L, \boldsymbol{w}_R) = \begin{cases} \boldsymbol{w}_L \text{ if } w_{L,2} - \sqrt{\kappa \dfrac{p_L}{\varrho_L}} \geqslant 0 \text{ (supersonic outlet)} \\ [w_{L,1}, \dots, w_{L,d+1}, w_{R,d+2}]^{\mathrm{T}} \\ \text{if } w_{L,2} - \sqrt{\kappa \dfrac{p_L}{\varrho_L}} < 0 \text{ and } w_{L,2} \geqslant 0 \text{ (subsonic outlet)} \\ [w_{R,1}, \dots, w_{L,d+1}, w_{L,d+2}]^{\mathrm{T}} \\ \text{if } w_{L,2} + \sqrt{\kappa \dfrac{p_L}{\varrho_L}} \geqslant 0 \text{ and } w_{L,2} < 0 \text{ (subsonic inlet)} \\ \boldsymbol{w}_R \text{ if } w_{L,2} + \sqrt{\kappa \dfrac{p_L}{\varrho_L}} < 0 \text{ (supersonic inlet)} \end{cases} \tag{16}$$

extrapolating the energy density e at the subsonic inlet and extrapolating the density ϱ and all components of the momentum $\varrho\boldsymbol{v}$ at the subsonic outlet. At the supersonic outlet, all components of the approximate solution are extrapolated. No information is extrapolated at the supersonic inlet.

2.2. System of non-linear ordinary differential equations

Put

$$\boldsymbol{y}^0 = \begin{bmatrix} \boldsymbol{y}_1^0 \\ \boldsymbol{y}_2^0 \\ \vdots \\ \boldsymbol{y}_N^0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_h(\boldsymbol{x},0)|_{\Omega_1} \\ \boldsymbol{w}_h(\boldsymbol{x},0)|_{\Omega_2} \\ \vdots \\ \boldsymbol{w}_h(\boldsymbol{x},0)|_{\Omega_N} \end{bmatrix} \in \mathscr{Q}^N \tag{17}$$

and

$$\boldsymbol{F}_{\text{int}}(\boldsymbol{y}) = \begin{bmatrix} -\frac{1}{|\Omega_1|}\sum_{j\in S_{\text{int}}(1)} H_{\text{int}}(\boldsymbol{y}_1,\boldsymbol{y}_j,\nu_{1j},|\partial\Omega_{1j}|) \\ -\frac{1}{|\Omega_2|}\sum_{j\in S_{\text{int}}(2)} H_{\text{int}}(\boldsymbol{y}_2,\boldsymbol{y}_j,\nu_{2j},|\partial\Omega_{2j}|) \\ \vdots \\ -\frac{1}{|\Omega_N|}\sum_{j\in S_{\text{int}}(N)} H_{\text{int}}(\boldsymbol{y}_N,\boldsymbol{y}_j,\nu_{Nj},|\partial\Omega_{Nj}|) \end{bmatrix} \tag{18}$$

$$\boldsymbol{F}_{\text{wall}}(\boldsymbol{y}) = \begin{bmatrix} -\frac{1}{|\Omega_1|}\sum_{\alpha\in S_{\text{wall}}(1)} H_{\text{wall}}(\boldsymbol{y}_1,\nu_{1\alpha},|\partial\Omega_{1\alpha}|) \\ -\frac{1}{|\Omega_2|}\sum_{\alpha\in S_{\text{wall}}(2)} H_{\text{wall}}(\boldsymbol{y}_2,\nu_{2\alpha},|\partial\Omega_{2\alpha}|) \\ \vdots \\ -\frac{1}{|\Omega_N|}\sum_{\alpha\in S_{\text{wall}}(N)} H_{\text{wall}}(\boldsymbol{y}_N,\nu_{N\alpha},|\partial\Omega_{N\alpha}|) \end{bmatrix} \tag{19}$$

$$\boldsymbol{F}_{io}(\boldsymbol{y},t) = \begin{bmatrix} -\frac{1}{|\Omega_1|}\sum_{\alpha\in S_{io}(1)} H_{io}(\boldsymbol{y}_1,\boldsymbol{w}_{1\alpha},\nu_{1\alpha},|\partial\Omega_{1\alpha}|) \\ -\frac{1}{|\Omega_2|}\sum_{\alpha\in S_{io}(2)} H_{io}(\boldsymbol{y}_2,\boldsymbol{w}_{2\alpha},\nu_{2\alpha},|\partial\Omega_{2\alpha}|) \\ \vdots \\ -\frac{1}{|\Omega_N|}\sum_{\alpha\in S_{io}(N)} H_{io}(\boldsymbol{y}_N,\boldsymbol{w}_{N\alpha},\nu_{N\alpha},|\partial\Omega_{N\alpha}|) \end{bmatrix} \tag{20}$$

where $\boldsymbol{w}_{i\alpha}(t) = \boldsymbol{w}_h^B(\boldsymbol{x},t)|_{\partial\Omega_{i\alpha}}$ represents boundary conditions at the faces $\partial\Omega_{i\alpha}$, $\alpha\in S_{io}$, $1\leqslant i\leqslant N$. Our aim is to find coefficients $\boldsymbol{y}\in C^1(0,T;\mathscr{Q}^N)$ to the approximate solution

$$\boldsymbol{w}_h(\boldsymbol{x},t) = \sum_{i=1}^{N} \boldsymbol{y}_i(t)\chi_i(\boldsymbol{x}) \tag{21}$$

satisfying

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{F}(\boldsymbol{y}(t), t) \quad \text{for all } t \in (0, T) \tag{22}$$

$$\boldsymbol{y}(0) = \boldsymbol{y}^0 \tag{23}$$

The symbol $\chi_i(\boldsymbol{x})$, $1 \leqslant i \leqslant N$ denotes Ω_i-characteristic functions. Note that the right-hand side vector function

$$\boldsymbol{F}(\boldsymbol{y}(t), t) = \boldsymbol{F}_{\text{int}}(\boldsymbol{y}(t)) + \boldsymbol{F}_{\text{wall}}(\boldsymbol{y}(t)) + \boldsymbol{F}_{io}(\boldsymbol{y}(t), t) \tag{24}$$

has the form corresponding to (11).

3. SOLVABILITY OF THE SEMI-DISCRETE PROBLEM

The existence and uniqueness of solution to the semi-discrete problem is a consequence of the basic existence and uniqueness theorem in the theory of ODEs (see e.g. References [3–5]).

3.1. Interior component $\boldsymbol{F}_{\text{int}}(\boldsymbol{y})$

In this contribution, we will analyse the well-known approximate Riemann solvers $\boldsymbol{f}_R(\boldsymbol{q}_L, \boldsymbol{q}_R)$ of Steger–Warming (see References [2, 6]), Vijayasundaram (see References [2, 7]), Van Leer (see Reference [2]) and Osher–Solomon (see References [2, 8, 9]). Modern, modified schemes such as e.g. HLLE (see References [10, 11]), AUSM (see Reference [12]) or recently appeared AUSMD, AUSMV, AUSMDV (see Reference [13]) inherit the essential properties implying the unique solvability of the semi-discrete problem and can be analysed in the same way.

Approximate Riemann solvers of Steger–Warming, Vijayasundaram and Van Leer are continuous in $\mathcal{Q}^2$. Their Jacobi matrices

$$\frac{\mathrm{D}\boldsymbol{f}_R(\boldsymbol{q}_L, \boldsymbol{q}_R)}{\mathrm{D}\boldsymbol{q}_L}, \quad \frac{\mathrm{D}\boldsymbol{f}_R(\boldsymbol{q}_L, \boldsymbol{q}_R)}{\mathrm{D}\boldsymbol{q}_R} \tag{25}$$

are continuous in $\mathcal{Q}_0 \equiv \mathcal{Q}^2 \setminus (\mathcal{Q}_{v_1-c=0} \cup \mathcal{Q}_{v_1=0} \cup \mathcal{Q}_{v_1+c=0})^2$ where

$$\mathcal{Q}_{v_1-c=0} = \left\{ \boldsymbol{q} \in \mathcal{Q};\ q_2 - \sqrt{\frac{\kappa(\kappa-1)\left(q_{d+2} - \dfrac{\sum_{i=2}^{d-1} q_i^2}{2q_1}\right)}{q_1}} = 0 \right\} \tag{26}$$

$$\mathcal{Q}_{v_1=0} = \{\boldsymbol{q} \in \mathcal{Q};\ q_2 = 0\} \tag{27}$$

$$\mathcal{Q}_{v_1+c=0} = \left\{ \boldsymbol{q} \in \mathcal{Q};\ q_2 + \sqrt{\frac{\kappa(\kappa-1)\left(q_{d+2} - \dfrac{\sum_{i=2}^{d-1} q_i^2}{2q_1}\right)}{q_1}} = 0 \right\} \tag{28}$$

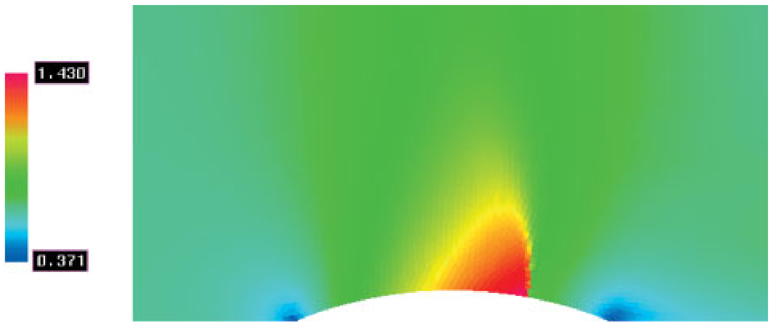

Plate 1. Steady-state Mach number distribution.

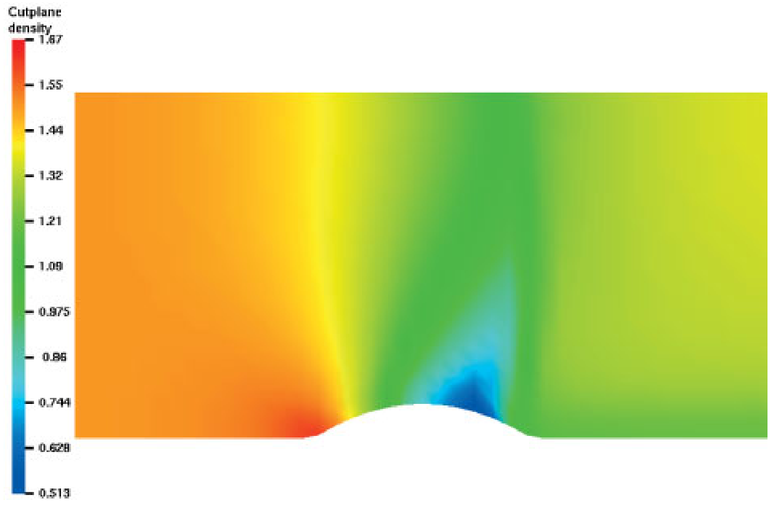

Plate 2. Steady density distribution, cut $y = 0$.

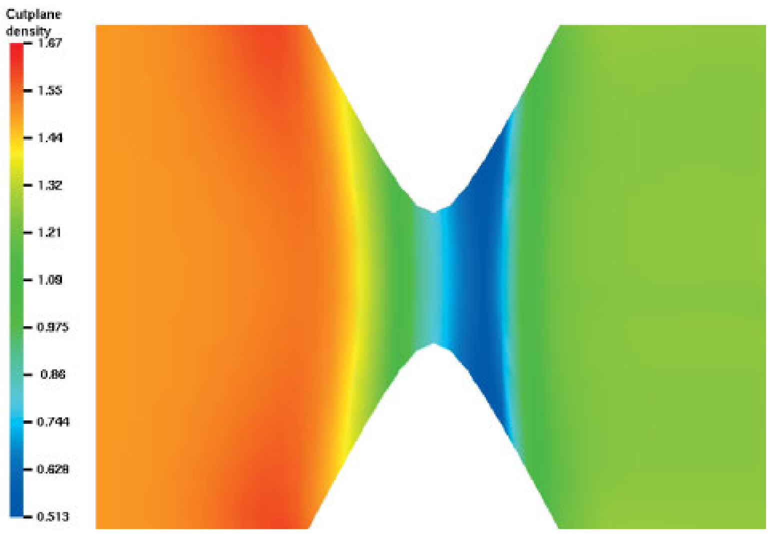

Plate 3. Steady density distribution, cut $z = 0.11$.

and are bounded in each compact subset of $\mathscr{Q}_0$. Obviously, the approximate Riemann solver by Osher–Solomon and its Jacobi matrices (25) are continuous in $\mathscr{Q}^2$. Thus, the vector function $\boldsymbol{F}_{\text{int}}(\boldsymbol{y})$, defined by (18), *is Lipschitz continuous* in $\omega \equiv \mathscr{Q}^N$ for all approximate Riemann solvers mentioned (and generally for many others).

3.2. Solid-wall component $\boldsymbol{F}_{\text{wall}}(\boldsymbol{y})$

As function (14) is smooth in $\mathscr{Q}$, the vector function $\boldsymbol{F}_{\text{wall}}(\boldsymbol{y})$ defined by (19) *is smooth* in $\omega \equiv \mathscr{Q}^N$.

3.3. Inlet/outlet component $\boldsymbol{F}_{io}(\boldsymbol{y}, t)$

On the contrary, switch (16) is obviously *not continuous*, and means a potential discontinuity of the right-hand side $\boldsymbol{F}$. This fact is demonstrated here analytically in the case of relatively simple inlet/outlet boundary conditions, but is valid generally.

3.4. Conclusion

A switch like (16), translating the behaviour of characteristic curves of compressible Euler equations into the inlet/outlet boundary conditions, is a common property of all hyperbolic problems. In the case of our model inlet/outlet boundary conditions, we can analytically confirm the known fact that *treatment of inlet/outlet boundary conditions influences the solvability of the semi-discrete problem in a crucial way*. Specifically in our case, discontinuity may easily be introduced into the right-hand side if no additional care is taken about the situation, when the flow across an inlet/outlet edge changes its type from subsonic to supersonic and/or inlet to outlet (and vice versa).

When we want to respect the characteristic behaviour of the solved equations, we *must* allow for a switch of type (16) in the numerical scheme. Therefore, to make sure that the right-hand side remains continuous also across branches of that switch, we *must monitor the flow through inlet and outlet boundary faces*. Boundary states $\boldsymbol{w}_{i\alpha}$ have to be adjusted according to the outgoing characteristics of the approximate solution in such a way that values returned by switch (16) *do not change discontinuously* when the flow changes its type.

4. NUMERICAL EXAMPLES

In the following few examples we illustrate our long-term experience with various ODE solvers. Their performance generally strongly depends on various aspects of the solved problem (not only stiffness, but also the position of the initial state in the set of admissible states and others). Therefore, it is not easy to define what "usual behaviour" of the methods means, and *it is definitively not possible to make conclusions using only a few presented examples*. Let us emphasize that it is our aim only to point out various aspects related to problem-specific choice of suitable ODE schemes, and that we do not intend to advocate some of them. Some conclusions can indeed be made, regarding, e.g. presence or lack of local error control, memory requirements, etc.

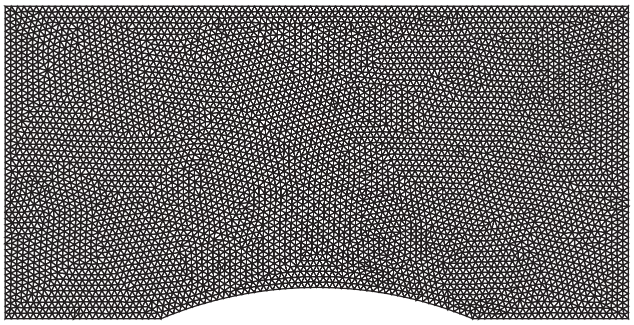
Figure 1. Unstructured triangular grid in the GAMM channel.

4.1. *Transonic flow in the 2D GAMM channel*

The GAMM channel is a standard test example for transonic flow simulations (name rooted in German “Gesselschaft für Angewandte Mathematik und Mechanik). The channel is 2 m long, 1 m high, with a 10% circular bump in the middle of its bottom. The fluid (in our case the air) flows from the left to the right. The upper and lower sides represent solid walls. Its geometry and an unstructured grid shown in Figure 1 consists of $N = 11\,048$ triangles.

Initial data are chosen constant with $\varrho^0 = 1.5\ \mathrm{kg/m^3}$, $\boldsymbol{v}^0 = [205.709227, 0]^{\mathrm{T}}$ m/s and $p^0 = 101\,000$ Pa. We prescribe the density $\varrho = 1.5\ \mathrm{kg/m^3}$ and velocity $\boldsymbol{v} = [205.709227, 0]^{\mathrm{T}}$ m/s at the subsonic inlet and $p^0 = 101\,000$ Pa at the subsonic outlet. For the construction of the numerical flux, we use the approximate Riemann solver by Osher–Solomon (for its 2D form see Reference [9]).

We will integrate in time using the following ODE schemes:

- Standard explicit Runge–Kutta schemes of order one, two and four, equipped with the largest possible time step that still keeps them stable.
- Backward Euler scheme with linearized RHS of the system of ODEs, again with a constant (largest stable) time step. The arising sparse system of linear equations is preconditioned with ILU and solved iteratively by suitable biconjugate gradient methods.
- Implicit stiff and non-stiff higher-order adaptive schemes offered by the ODE packages ODEPACK and DDASPK. The stiff ones are in both cases based on backward difference formula (BDF), and the non-stiff ones on Adams scheme and implicit Runge–Kutta methods, respectively. We paid special attention to apply the solvers under equivalent conditions, unless stated otherwise. Most important among these conditions are:
    - ○ identical full block structure of the sparse Jacobi matrix of the right-hand side (each finite volume corresponds to a block in the matrix of the size $(d+2)\times(d+2)$, where d is spatial dimension, and there are $M+1$ blocks per block-row, where M is the number of neighbours of a finite volume),
    - ○ identical ILU preconditioning of sparse Jacobi matrix of the right-hand side,
    - ○ identical relative and absolute bounds for local error (we have chosen level of 10^{-5}).

  For detail description of these ODE schemes and miscellaneous input parameters we refer to References [14–18]. Detail information on these solvers, including also description of input parameters, can be found also directly in comments within the public domain source codes.

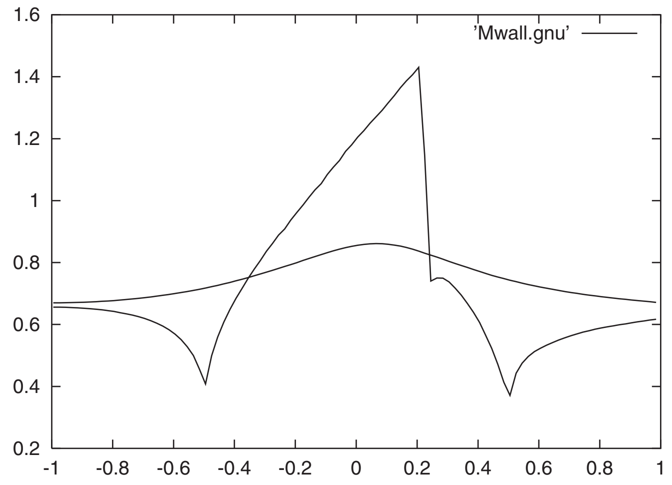

Figure 2. Mach number on the solid walls.

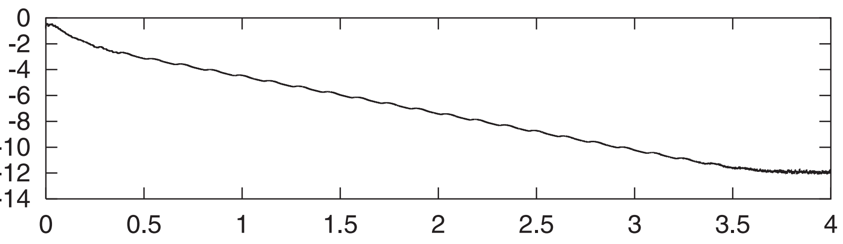

Figure 3. CPU time convergence history for the explicit 1st-order Runge–Kutta (Euler) scheme.

All solvers converge to the same steady-state solution shown in Plate 1 (colour map of the Mach number) and Figure 2 (Mach number on solid walls). This steady solution is in good agreement with results of other authors. In Figure 2, one can even see the *Zierep singularity* (the Mach number is briefly increasing just after the shock) which is a means for the rating of the quality of numerical schemes on this standard test channel.

In Figures 3–9 we compare convergence histories with the CPU time (CPU hours) on the x-axis and the decimal logarithm of the norm

$$\left\| \sum_{i=1}^{N} \dot{\mathbf{y}}_i(t)\chi_i \right\|_{L^\infty(\Omega)} \tag{29}$$

on the y-axis. The approximate solution is considered steady if norm (29) of its time derivative achieves a sufficiently small prescribed tolerance. These histories will be discussed together with the 3D ones in the last section of the paper.

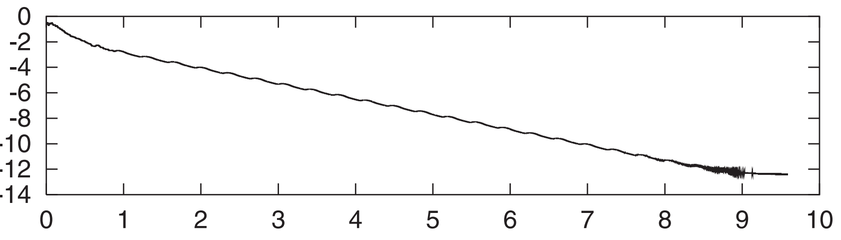

Figure 4. CPU time convergence history for the explicit 2nd-order Runge–Kutta scheme. Notice that although we used the largest possible CFL constant to keep the computation stable, the size of the time step did not achieve two times the size corresponding to explicit Euler method. The reasons are fast initial developments in the solution, when all schemes tend to leave the set of admissible states, computing negative densities and/or pressures. This implies additional restrictions on the size of the time step.

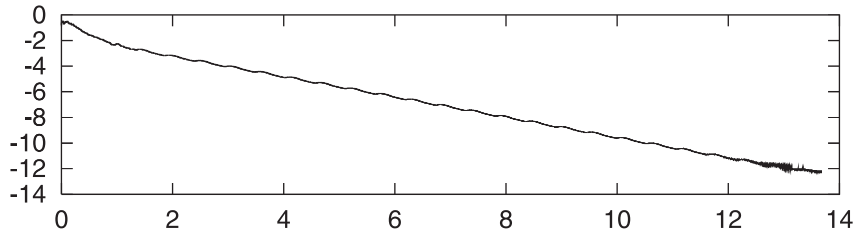

Figure 5. CPU time convergence history for the explicit 4th-order Runge–Kutta scheme. Notice a similar effect as with the 2nd-order scheme. Let us emphasize that this behaviour depends on the specific problem.

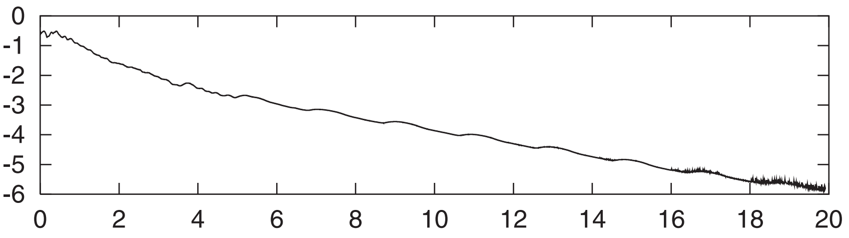

Figure 6. CPU time convergence history for the semi-implicit backward Euler scheme. Time steps are approximately 10 times larger in comparison with explicit Euler scheme; however, assembling and solution of the arising linear system influence the performance significantly.

In Figures 10–16 we show how frequently the RHS of the ODE system is used by the schemes. We again plot convergence histories but now with thousands of calls to the RHS on the x-axis. The meaning of the y-axis stays unchanged, i.e. the decimal logarithm of norm (29).

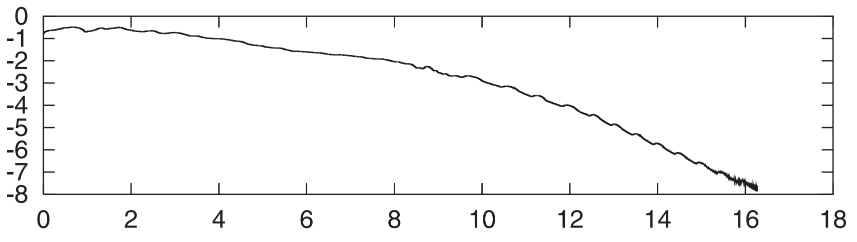

Figure 7. CPU time convergence history for the implicit adaptive BDF offered by DDASPK, with diagonal preconditioning of the Jacobi matrix of the right-hand side (this was the only preconditioning that we succeeded to implement, investing the same effort that we needed to interface with ODEPACK).

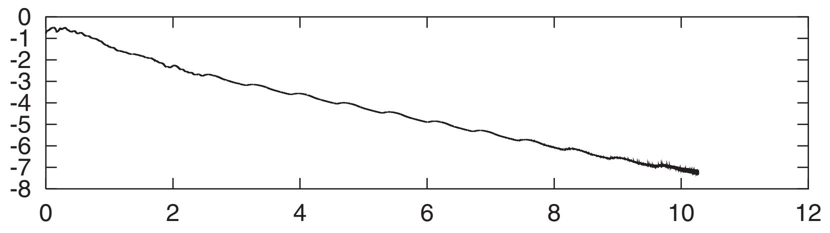

Figure 8. CPU time convergence history for the implicit adaptive Adams scheme offered by ODEPACK.

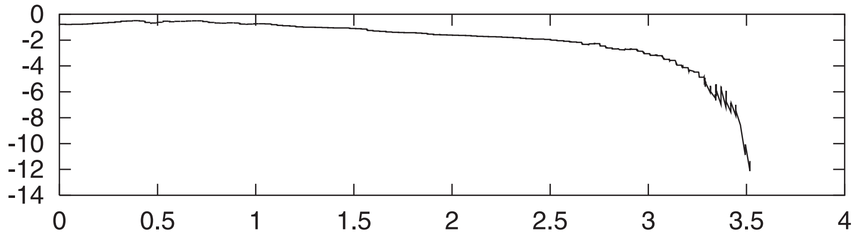

Figure 9. CPU time convergence history for the implicit adaptive BDF offered by ODEPACK.

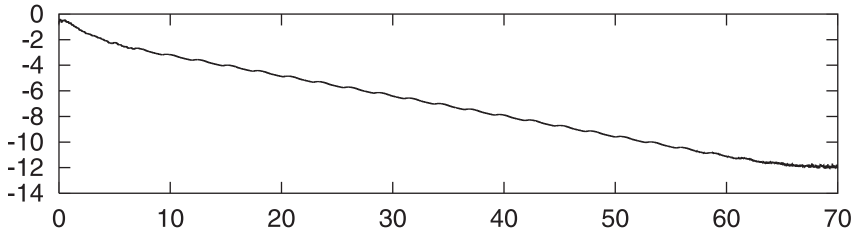

Figure 10. RHS calls convergence history for the explicit 1st-order Runge–Kutta (Euler) scheme.

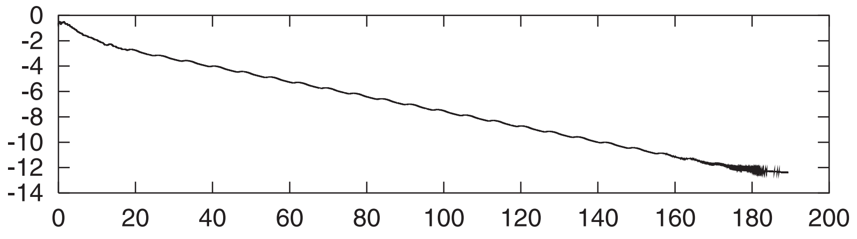

Figure 11. RHS calls convergence history for the explicit 2nd-order Runge–Kutta scheme.

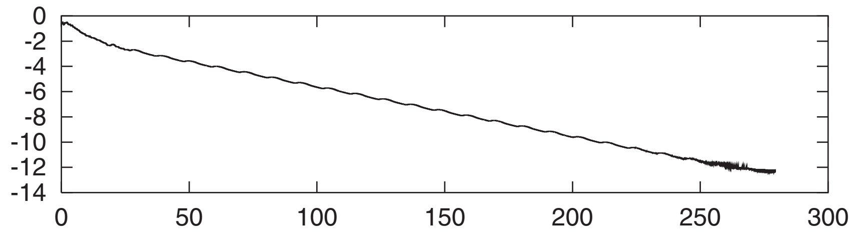

Figure 12. RHS calls convergence history for the explicit 4th-order Runge–Kutta scheme.

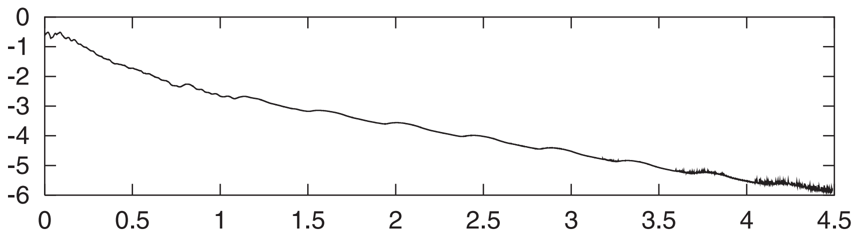

Figure 13. RHS calls convergence history for the semi-implicit backward Euler scheme.

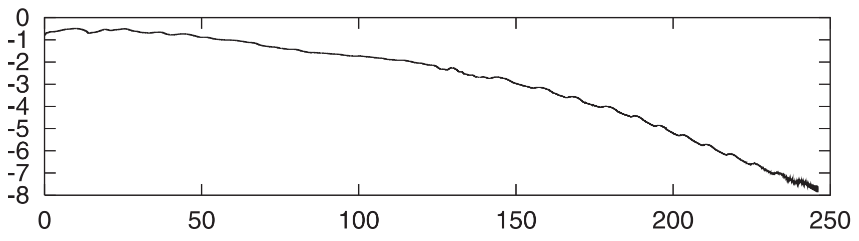

Figure 14. RHS calls convergence history for the implicit adaptive BDF offered by DDASPK.

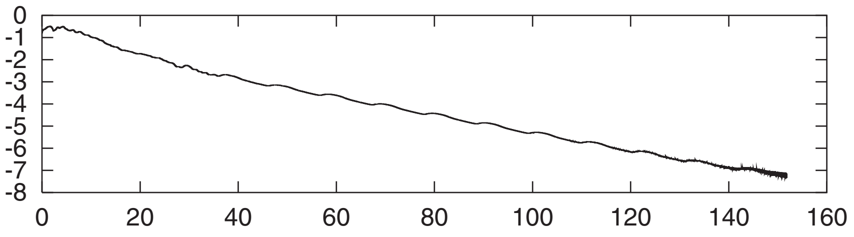

Figure 15. RHS calls convergence history for the implicit adaptive Adams scheme offered by ODEPACK.

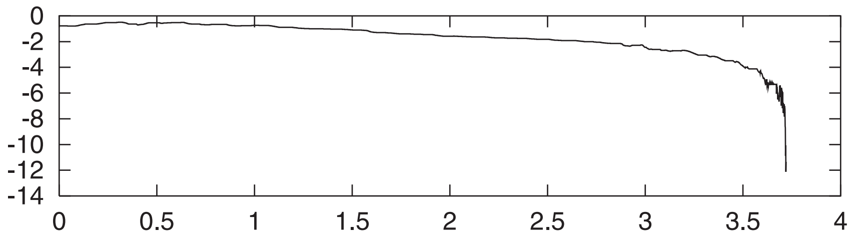

Figure 16. RHS calls convergence history for the implicit adaptive BDF offered by ODEPACK.

4.2. Transonic flow in a 3D channel with a saddle

In this subsection, we will deal with the numerical solution of transonic flow in a 3D channel containing a parabolic saddle (extension of the GAMM channel to 3D). The channel is 2 m long, 1.5 m wide and 1 m high. The midpoint of its bottom is placed at the origin of the system of co-ordinates and its edges are parallel to the corresponding axes. The saddle itself is described by the function

$$s(x_1,x_2) = -x^2 + \frac{0.8}{3}\,y^2 + 0.1 \tag{30}$$

The bottom of the channel is shown in Figure 17. Quadrilateral boundary faces corresponding to $x_L = -1$ and $x_R = 1$ represent inlet and outlet, remaining boundary faces, containing trace of the parabolic bump, represent solid walls.

Initial data is chosen constant with values $\varrho^0 = 1.5\,\mathrm{kg/m^3}$, $\boldsymbol{v}^0 = [190.0, 0, 0]^{\mathrm{T}}\,\mathrm{m/s}$ and $p^0 = 101\,000\,\mathrm{Pa}$. The air flows through the channel from the left to the right with the boundary conditions $\varrho^0 = 1.5\,\mathrm{kg/m^3}$, $\boldsymbol{v}^0 = [190.0, 0, 0]^{\mathrm{T}}\,\mathrm{m/s}$ at the inlet and $p^0 = 101\,000\,\mathrm{Pa}$ at the outlet.

The domain is covered with a structured tetrahedral grid consisting of $N = 144\,000$ finite volumes. For the construction of the numerical flux, the approximate Riemann solver by Osher–Solomon [2] is used.

Integration in time is performed using the previously described ODE schemes.

The steady-state solution to the problem is presented in Plates 2 and 3. Convergence histories showing the speed of convergence to the steady-state (again with the CPU hours on the x-axis and the decimal logarithm of norm (29) on the y-axis) are presented in Figures 18–22.

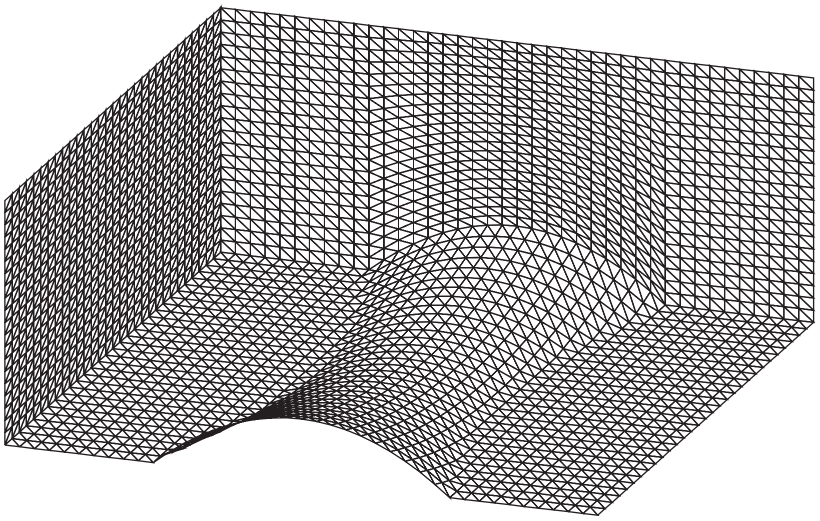
Figure 17. 3D curved channel.

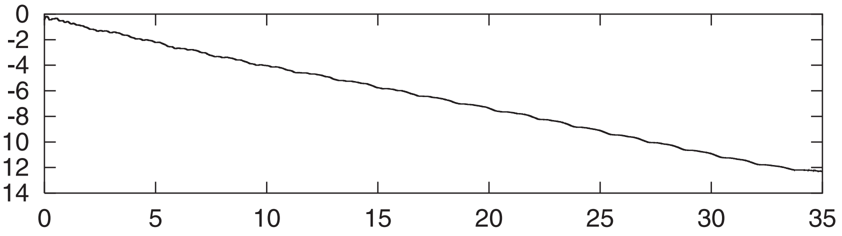

Figure 18. CPU time convergence history for the explicit 1st-order Runge–Kutta (Euler) scheme.

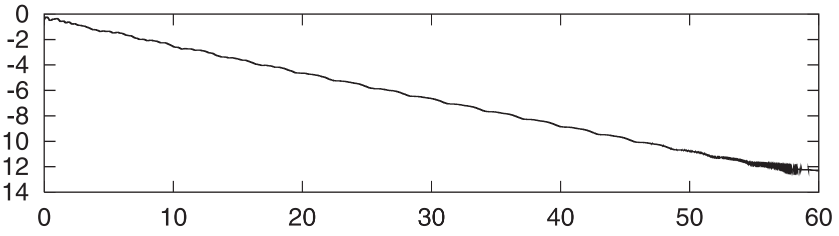

Figure 19. CPU time convergence history for the explicit 2nd-order Runge–Kutta scheme.

4.3. Conclusions and numerical experience

The Method of lines separates the time integration from the space discretization, which certainly brings more modularity into the numerical simulation of the flow evolution problem. We analysed existence and uniqueness of solution to the semi-discrete problem (a system

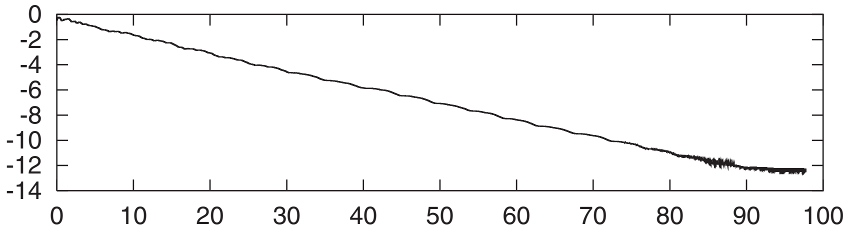

Figure 20. CPU time convergence history for the explicit 4th-order Runge–Kutta scheme.

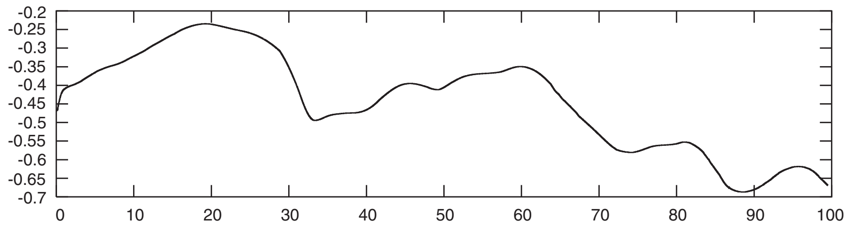

Figure 21. CPU time convergence history for the implicit adaptive BDF offered by DDASPK, with diagonal preconditioning of the Jacobi matrix of the right-hand side.

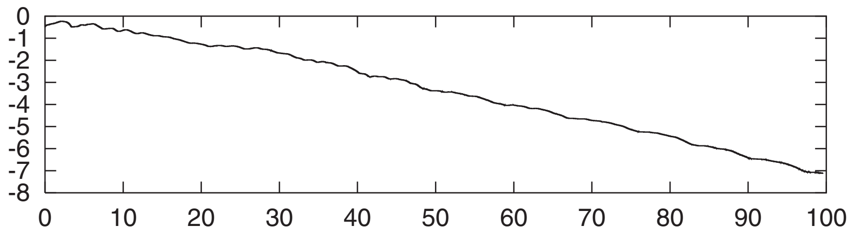

Figure 22. CPU time convergence history for the implicit adaptive Adams scheme offered by ODEPACK.

of ordinary differential equations), and formulated necessary conditions for the treatment of inlet/outlet boundary conditions, based on the behaviour of characteristic curves, preventing the right-hand side from becoming discontinuous. We also have shown that it makes sense to apply various ODE solvers to the numerical integration. An “optimal” choice however strongly depends on the character of the solved problem together with the type of results that we are interested in, and it is not possible to make universal statements on applicability of various schemes.

What we can say is that application of ODE packages ODEPACK and DDASPK can be a fast way to improve the stability and accuracy of the time integration mainly for smaller problems. For problems with a large number of degrees of freedom, application of implicit adaptive schemes can be limited either by their high memory requirements (as, e.g. BDF in ODEPACK, which needs to store Jacobi matrix of the right-hand side) or by large amount of CPU time caused by an excessive number of evaluations of the right-hand side (DDASPK—

see Figures 7 and 21). Let us however mention that this probably would improve when DDASPK offered easier access to more sophisticated preconditioners. For the sake of comparison, we decided to spend the same amount of time with interfacing with both these packages. ODEPACK could offer ILU preconditioning easily.

On the other hand, we have not succeeded in applying the implicit adaptive backward difference formula from ODEPACK to the presented 3D problem due to its huge storage requirements. In our opinion, with computers of today's standard (here we are not speaking about supercomputers with terabytes of allocatable memory), these implicit schemes seem not yet really relevant for large three-dimensional problems.

On the other hand, ODE packages always help to overcome stability problems (which are not negligible for explicit Runge–Kutta schemes). What is even more important, the ODE packages always perform certain precision control which helps to eliminate problems with negative pressures and densities within the computation. These troubles are usual for problems with badly shaped meshes or steep solution gradients (appearing, e.g. in simulation of nozzles and elsewhere). In such cases, due to lack of precision of the approximate time integration, the approximate solution to the semi-discrete problem easily leaves the exact one and runs out of the set of admissible states.

Concerning the precision, also for large problems there is the possibility to take advantage of a precise implicit higher-order adaptive scheme of the Adams type, which is offered by ODEPACK. This scheme is quite efficient also for large problems (see Figures 8, 22). Even if one does not have troubles with negative pressures and densities, controlled precision is particularly useful for non-stationary problems when one is interested in precise solution at all time levels. For steady-state computations of large problems, particularly with good quality meshes and reasonable gradients, explicit Runge–Kutta schemes are expected to be efficient. Obviously, some precision problems can be solved by increasing the order of the scheme or by decreasing the time step.

For steady-state computations, accurate ODE solvers sometimes can unnecessarily lose CPU time by their effort to achieve the prescribed tolerance for the approximate solution at all time levels. Practical computations show that an attempt to decrease the accuracy usually does not lead to dramatical improvements in computational times.

The time integration based on the backward Euler method (see Figures 6, 13) with linearized right-hand side is more stable and more precise than the explicit Runge–Kutta schemes. Moreover, it requires an essentially lower number of calls to the right-hand side than the other schemes (except for the implicit adaptive BDF from ODEPACK applicable only to smaller problems). Its efficiency depends on the quality of the iterative solution of the arising system of linear equations. We have used basic biconjugate gradient methods with the standard incomplete LU preconditioning. Due to the results published in Reference [19], we expect that the application of suitable multigrid techniques to the system of linear equations would improve the efficiency dramatically. In our opinion, in combination with suitable multigrid techniques, this scheme is very promising also for very large problems.

Last agreeable aspect of the mentioned ODE packages is that they are able to perform implicit adaptive time integration with the user only providing a subroutine for *explicit evaluation of the right-hand side*. In fact, here one invests the same amount of work as when using simpler, self-implemented explicit schemes. With the ODE packages, the work reduces to the implementation of a subroutine which processes the vector of coefficients of the approximate solution to the vector of derivatives. The user need not even call this subroutine—

the complete time integration is carried out by the ODE solver itself. The subroutines can be written as well in Fortran as in other programming languages (we have experience with C and C++). In the latter case, Fortran packages are to be linked to the resulting executable.

ACKNOWLEDGEMENTS

This work was partly supported by Grant 201/98/0528 of the Grant Agency of the Czech Republic.

REFERENCES

1. Šolín P. On the method of lines (application in fluid dynamics and a-posteriori error estimation). *Doctoral dissertation*, Faculty of Mathematics and Physics, Charles University, Prague, 1999.
2. Felcman J, Šolín P. Construction of the Osher–Solomon scheme for 3D Euler equations. *East-West Journal of Numerical Mathematics* 1998; **6**:43–64.
3. Coddington A, Levinson N. *Ordinary Differential Equations*. McGraw-Hill: New York, 1955.
4. Kurzweil J. *Ordinary Differential Equations*. SNTL: Prague, 1978.
5. Lambert JD. *Computational Methods in Ordinary Differential Equations*. John Wiley & Sons: London–New York, 1973.
6. Steger JL, Warming RF. Flux vector splitting of the inviscid gas dynamics equations with applications to finite difference methods. *Journal of Computational Physics* 1981; **40**:263–293.
7. Vijayasundaram G. Transonic flow simulation using upstream centered scheme of Godunov type in finite elements. *Journal of Computational Physics* 1986; **63**:416–433.
8. Osher S, Solomon F. Upwind difference schemes for hyperbolic systems of conservation laws. *Mathematics of Computation* 1982; **38**:339–374.
9. Spekreijse SP. Multigrid solution of the steady Euler equations. *Ph.D. Thesis*, Center for Mathematics and Computer Science, Amsterdam, 1988.
10. Einfeldt B. On Godunov-type methods for gas-dynamics. *SIAM Journal on Numerical Analysis* 1988; **25**:357–393.
11. Einfeldt B, Munz CC, Roe PL, Sjorgreen B. On Godunov-type methods near low densities. *Journal of Computational Physics* 1991; **92**:273–295.
12. Liou M-S, Steffen CJ. A new flux-splitting scheme. *Journal of Computational Physics* 1993; **107**:23–29.
13. Wada Y, Liou M-S. An accurate and robust flux-splitting scheme for shock and contact discontinuities. *SIAM Journal on Scientific Computing* 1997; **18**:633–657.
14. Hindmarsh AC. ODEPACK, a systematized collection of ODE solvers. *IMACS Transactions on Scientific Computation* 1983; **1**:55–64.
15. Brown PN, Byrne GD, Hindmarsh AC. VODE, a variable-coefficient ODE solver. *SIAM Journal on Scientific and Statistical Computing* 1989; **10**:1038–1051.
16. Brown PN, Hindmarsh AC. Reduced storage matrix methods in stiff ODE systems. *Journal of Applied Mathematics & Computers* 1989; **31**:40–91.
17. Petzold LR. A description of DDASSL: a differential/algebraic system solver. *IMACS Transactions on Scientific Computation* 1983; **1**:65–68.
18. Petzold LR. DAE's and the stability of moving mesh systems of partial differential equations. *Zeitrschrift fuer Angewandte Mathematik und Mechanik* 1996; **76**.
19. Haag R. Robuste Mehrgitterverfahren für die Euler- und Navier–Stokes Gleichungen. *Bericht N92/2*, University Stuttgart, 1997.