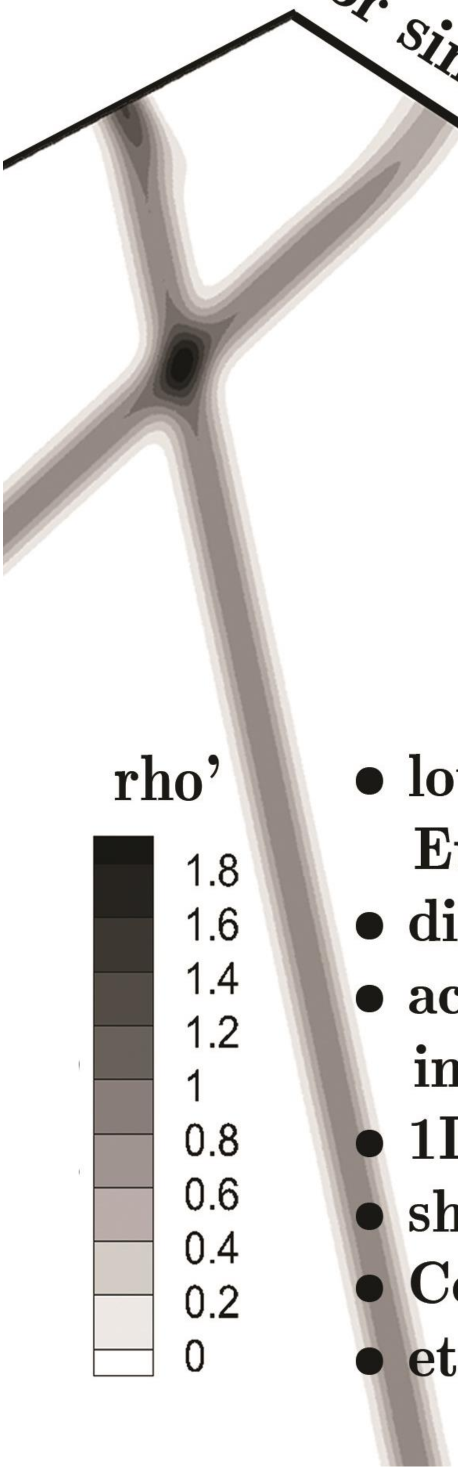
ColESo

Collection of **e**xact **so**lutions
for verification of numerical algorithms
for simulation of compressible flows



- lots of solutions of the linearized Euler equations in free space
- diffraction by corner and cylinder
- acoustical boundary layers in a square or cylindrical channel
- 1D Riemann problem
- shock wave profile
- Couette flow with different mu(T)
- etc.

This is the user manual for the ColESo library – Collection of Exact Solutions for the verification of numerical algorithms for simulation of compressible flows. ColESo is an open-source project available from https://github.com/bahvalo/ColESo. ColESo is written in C++ and contains an interface for programs in C and FORTRAN.

The core of ColESo is a set of solutions of the linearized Euler equations and the linearized Navier – Stokes equations on a steady uniform background field. The library contains simple solutions like a planar wave in free space as well as rather complex ones like an acoustic wave in a planar or cylindrical channel considering the viscosity and heat conductivity. Besides, ColESo contains several nonlinear solutions.

This document consists of three sections.

I. Governing equations.

II. “Filing cabinet” of the solutions implemented in ColESo. Each card contains the class name corresponding to the solution, the problem setup and such properties of the solution as smoothness, behavior at infinity[1] and computational complexity. Illustrations are provided to most of the solutions.

III. Usage of ColESo and common details of the implementation.

Mathematical proof of the solutions and details of their implementation are not included into this document.

[1] Behavior as $\mathbf{r} \to \infty$ for a fixed time

I. Governing equations

The Navier – Stokes equations for an ideal gas has the form

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathcal{F}_{\tilde{n}}(\mathbf{Q}) = \nabla \cdot \mathcal{F}_{v}(\mathbf{Q}, \nabla \mathbf{Q}) + \mathbf{S}(t, \mathbf{r}, \mathbf{Q}), \tag{1}$$

$$\mathbf{Q} = \begin{pmatrix} \rho \\ \rho \mathbf{u} \\ E \end{pmatrix}, \quad \mathcal{F}_{\tilde{n}}(\mathbf{Q}) = \begin{pmatrix} \rho \mathbf{u} \\ \rho \mathbf{u} \otimes \mathbf{u} + pI \\ (E+p)\mathbf{u} \end{pmatrix}, \quad \mathcal{F}_{v}(\mathbf{Q}, \nabla \mathbf{Q}) = \begin{pmatrix} 0 \\ \tau \\ \tau \cdot \mathbf{u} - \sigma \end{pmatrix}. \tag{2}$$

Here $E = \rho \mathbf{u}^2 / 2 + \rho \varepsilon$ is the full energy, ε is the specific internal energy given by $\varepsilon = p / ((\gamma - 1)\rho)$. Unless specifically stated, $S(t, \mathbf{r}, \mathbf{Q})$ is zero. Stress tensor is defined as $\tau_{ij} = \mu \left(\nabla_i u_j + \nabla_j u_i - (2/3)\delta_{ij} \nabla_k u_k \right)$, where μ is the dynamic viscosity coefficient. Heat flux vector is given by $\sigma = -\gamma \mu \nabla \varepsilon / \mathrm{Pr}$, where Pr is the Prandtl number. Dependency of μ on density or temperature is specified for a specific test.

The Euler equations result from the Navier – Stokes system by dropping the viscosity and heat conductivity. In the case of discontinuous solutions, the integral form of the equations is considered, namely,

$$\frac{d}{dt} \int_V \mathbf{Q} dV + \int_{\partial V} \mathcal{F}_c(\mathbf{Q}) \cdot \mathbf{n} dS = \int_V \mathbf{S}(t, \mathbf{r}, \mathbf{Q}) dV, \tag{3}$$

with the additional condition of entropy non-decreasing. Here V is an arbitrary domain.

Several solutions below are exact solutions for the incompressible Navier – Stokes (INSE) or incompressible Euler equations (IEE). After replacing the ideal gas equation-of-state by $\rho \equiv 1$ and assuming μ = *const*, system (1)–(2) reduces to

$$\nabla \cdot \mathbf{u} = 0,$$
$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \mu \Delta \mathbf{u}.$$

Most of the solutions below are exact solutions of the linearized equations. The concept of linearized equations implies that a “background” field $\bar{\mathbf{Q}}(t, \mathbf{r})$ satisfying (1)–(2) is prescribed. To get the linearized Navier – Stokes equations (LNSE), one may substitute $\mathbf{Q} = \bar{\mathbf{Q}} + \epsilon \mathbf{Q}'$ into (1)–(2) (the source term may explicitly depend on ϵ) on and neglect the terms of order ϵ^2. For an arbitrary background field, these equations are cumbersome. We consider the case that $\bar{\mathbf{Q}}$ is a steady uniform flow, i. e. does not depend on t and $\mathbf{r}$. By multiplying the variables by constants we can assume without loss that $\bar{\rho}=1$, $\bar{p}=1/\gamma$. Then Navier – Stokes equations linearized on steady uniform fields can be represented in the form

$$\begin{aligned}
&\frac{d\rho'}{dt}+\nabla_j u'_j = S'_\rho(t,\mathbf{r}),\\
&\frac{du'_i}{dt}+\nabla_i p' = \mu\left(\Delta u'_i+\frac{1}{3}\nabla_i(\nabla_j u'_j)\right)+S'_{u_i}(t,\mathbf{r}),\\
&\frac{d(p'-\rho')}{dt}=\frac{\mu}{\mathrm{Pr}}\Delta(\gamma p'-\rho')+\left(S'_p(t,\mathbf{r})-S'_\rho(t,\mathbf{r})\right),
\end{aligned} \tag{4}$$

where $d/dt=\dfrac{\partial}{\partial t}+(\bar{\mathbf{u}}\cdot\nabla)$ is the substantial derivative.

The Euler equations can be also linearized on a discontinuous background field. In this case the linearization is applied to (3) and takes into account displacements of discontinuities of order ϵ.

If the background field is continuous, then the linearized Euler equations (LEE) can be obtained from LNSE by dropping the viscosity and heat conductivity terms. Particularly, if $\bar{\rho}=1$, $\bar{\mathbf{u}}=const$, $\bar{p}=1/\gamma$, the LEE take the form

$$\frac{dp'}{dt}+\mathrm{div}\mathbf{u}'=S'_p(t,\mathbf{r}),\quad \frac{d\mathbf{u}'}{dt}+\nabla p'=S'_\mathbf{u}(t,\mathbf{r}),\quad \frac{d}{dt}(p'-\rho')=S'_p(t,\mathbf{r})-S'_\rho(t,\mathbf{r}). \tag{5}$$

The equation for the entropy pulsation ($p'-\rho'$) disjoins and becomes a transport equation with the velocity of background field.

Now consider a particular case of (5). Let the entropy perturbation be zero ($S'_p(t,\mathbf{r})\equiv S'_\rho(t,\mathbf{r}), p'\equiv\rho'$), the source term in the momentum equation be zero ($S'_\mathbf{u}(t,\mathbf{r})\equiv 0$), and the velocity field be potential ($\mathbf{u}'=\nabla W$). In this case, W is called the velocity potential. Then for some (generally another) velocity potential W system (5) yields

$$\frac{d^2W}{dt^2}-\Delta W=-S'_p(t,\mathbf{r}),\quad \mathbf{u}'=\nabla W,\quad p'=-\frac{dW}{dt}.$$

In the reference frame moving with the background velocity, this reduces to the wave equation.

II. “Filing cabinet” of the solutions

Simple solutions that preserve the form of initial data

s_Polynom: one-dimensional polynomial profile

Class	s_Polynom
Math. model	Linearized Euler equations / transport equation
Solution	Entropy wave, i. e. $u' = p' = 0$, $\rho'(t,x) = \rho'(0, x - \bar{u}t)$. $\rho'(0,x)$ is a Chebyshev polynomial of a given order
Smoothness	Infinitely smooth
Behavior at infinity	Solution grows as $x \to \infty$
Purpose	Check that the truncation error is zero on polynomials

s_4peak: entropy perturbation “4 peaks”

Class	s_4peak
Math. model	Linearized Euler equations / transport equation
Solution	Entropy wave. $\rho'(0,x)$ is prescribed by a composition of elementary functions and shown in Fig. 1
Smoothness	Discontinuous
Behavior at infinity	Has a finite support
Purpose	A popular test for accuracy analysis of methods for discontinuous solutions

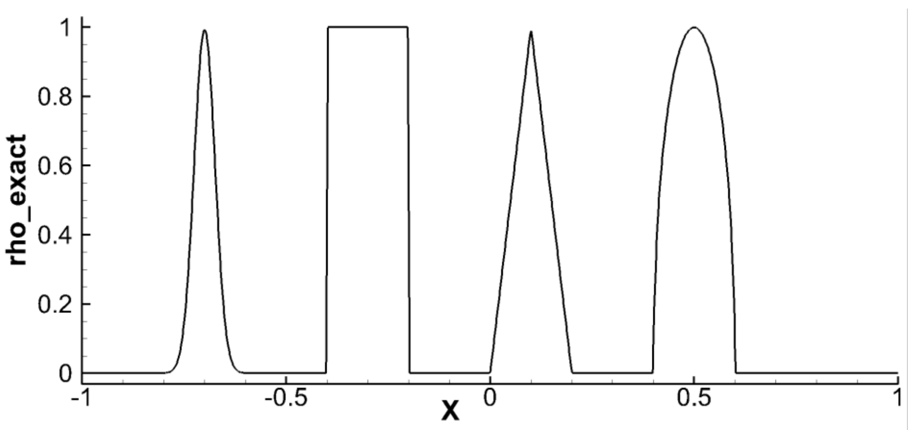


Fig. 1. Initial conditions for the “4 peaks” problem

s_PlanarSinus, s_PlanarGauss: planar acoustic wave

Class	s_PlanarSinus, s_PlanarGauss
Math. model	Wave equation (background flow is admissible)
Solution	Planar acoustic wave $\begin{pmatrix} \rho' \\ \mathbf{u}' \\ p' \end{pmatrix} = f(\mathbf{e}\cdot(\mathbf{r}-\bar{\mathbf{u}}t)-\bar{c}t)\begin{pmatrix} 1/\bar{c}^2 \\ \mathbf{e}/(\bar{\rho}\bar{c}) \\ 1 \end{pmatrix}$, where **e** is a vector with unit length. Sine and Gaussian wave profiles are admissible, see Fig. 2. In the first case, $f(x) = A\sin(2\pi\nu x)$ or $f(x) = A\sin(2\pi\nu x)\Theta(x)$. In the second case, $f(x) = A\exp(-(\ln 2)x^2/b^2)$ or f is a lattice of such pulses. Fig. 3 shows a solution when the normal to the wave front (**e**) is not collinear to any of the coordinate axes
Purpose	Simplest solution for the verification. As most of the smooth solution, it is usable for the verification of high-accuracy methods for compressible flows. By the way, s_PlanarSinus also admits non-smooth solutions.

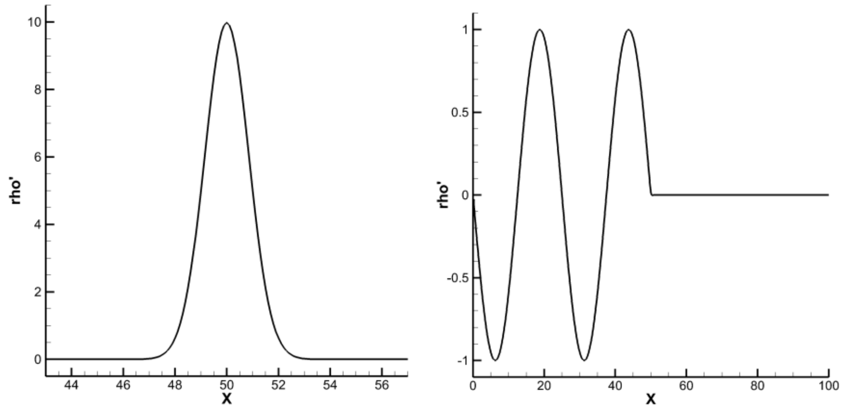


Fig. 2. Gaussian and sine profiles of the initial perturbation

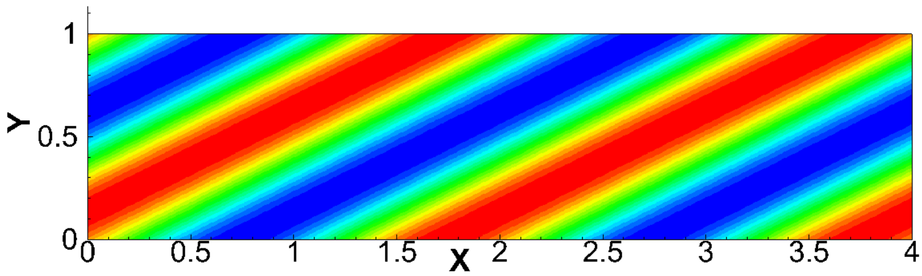


Fig. 3. Planar wave, when the normal to the wave front is not collinear to any of coordinate axes

s_EntropyVortex: entropy and vortex perturbations

Class	s_EntropyVortex
Math. model	Linearized Euler equations
Domain	$t > 0, \mathbf{r} \in \mathbb{R}^2$
Problem setup	Solution contains only vortex and entropy components. Initial conditions are $\rho' = A_2 \exp(-(\ln 2)(x^2 + y^2)/b^2)$, $p' = 0$, и $\mathbf{u}' = \mathrm{rot}(0,0,Q)^T$, where $Q(x,y) = A_1 \exp(-(\ln 2)(x^2 + y^2)/b^2)$, see Fig. 4. Perturbations are transported with the background speed without deformations
Smoothness	Infinitely smooth
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Purpose	Simple problem for the verification

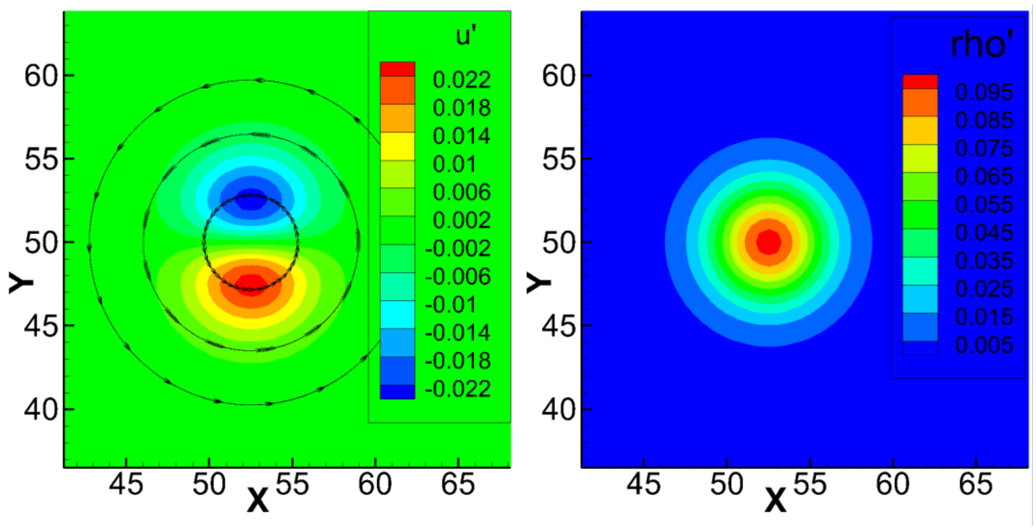


Fig. 4. Vortex (at the left) and entropy (at the right) perturbations travelling with the velocity of the background flow

Linear one-dimensional problems

s_Source1D: one-dimensional problem with a source term

Class	s_Source1D
Math. model	Wave equation
Domain	$t > 0, x \in \mathbb{R}$
Problem setup	Zero initial data, source term $S'_p(t,x) = f(x)g(t)$. Space form can be Gaussian or $f(x) = \cos^2(\pi x/(2b))\,\Theta(b-x)$; time dependency of the source term is $g(t) = \sin^p(2\pi f t)\,\Theta(t_{\max} - t)\Theta(t - t_{\min})$, where $p = 1$ or $p = 4$
Smoothness	Different variants are possible
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	Requires computation of an integral

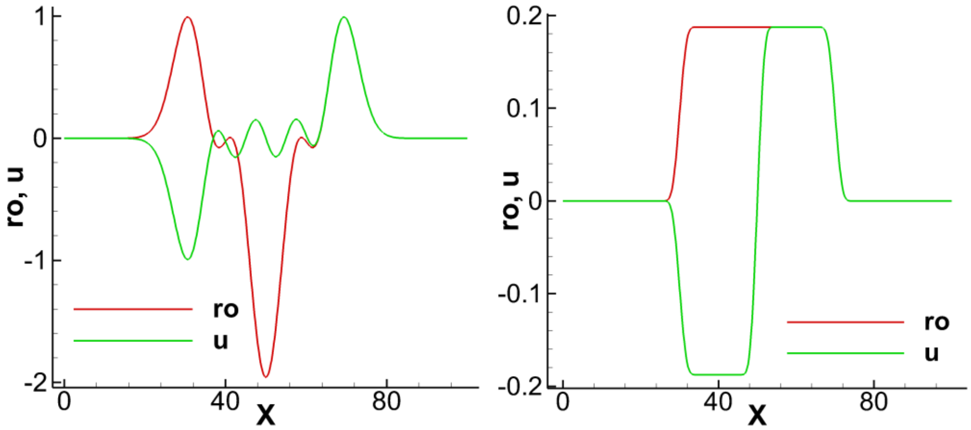


Fig. 5. Possible solutions of the 1D problem with the source term

s_PlanarAcousticShock: acoustic wave moving through a shock

Class	s_PlanarAcousticShock
Math. model	Linearized Euler equations with a discontinuous background field
Domain	$t > 0,\ x \in \mathbb{R}$
Problem setup	Background field is a steady shock at the point X_0. Linearized Euler equations hold in $x > X_0$ and $x < X_0$; at $x = X_0$ the linearized Hugoniot conditions are imposed taking into account the pulsation of the shock velocity. Initial data prescribe the right acoustic wave with Gaussian profile located in the supersonic domain
Smoothness	Discontinuous at the discontinuity point of the background field
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions
Purpose	Accuracy analysis of the numerical methods for solving gas dynamics problems with discontinuous solutions (this problem is assumed to be solved using the Euler equations)
Note	In solving this problem using the Euler equations, entropy wave is reproduced erroneously by most of the numerical methods unless the mesh is small enough to trace the movement of the shock

The solution of this problem for density pulsation is shown in Fig. 6. Different colors show solutions at different time moments. Small green peak at the right corresponds to the entropy wave.

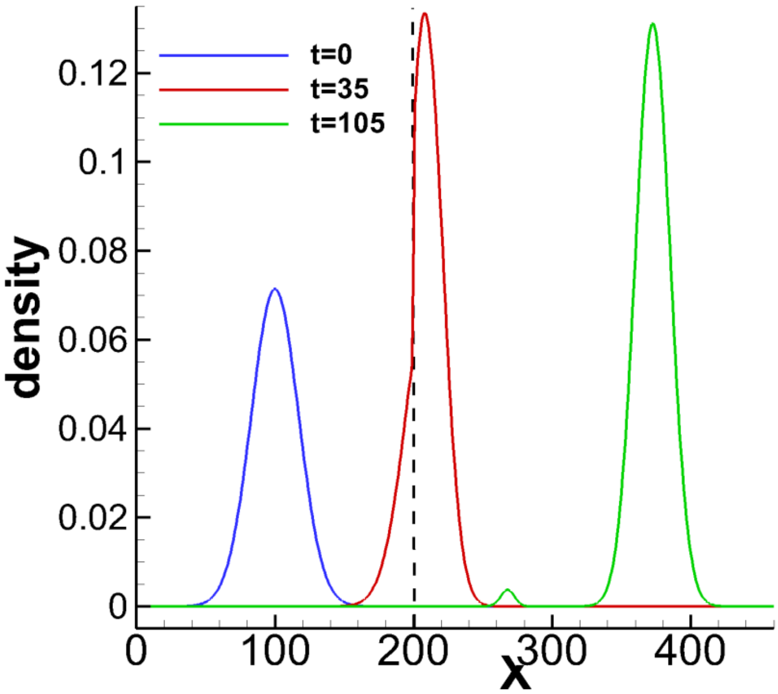


Fig. 6. Solution for the acoustic wave moving through a shock

Linear two-dimensional axisymmetric problems

s_Gaussian2D: problem with initial pulse with Gaussian profile

Class	s_Gaussian2D
Math. model	Wave equation (background flow is admissible)
Domain	$t > 0, \mathbf{r} \in \mathbb{R}^2$
Problem setup	Initial data are specified as $\rho'(0,r) = p'(0,r) = \exp\left(-\ln 2\,(r/b)^2\right), \quad \mathbf{u}'(0,r) = 0$. Lattice of pulses of this form is admissible, which corresponds to periodical boundary conditions
Smoothness	Infinitely smooth
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	The solution is calculated differently depending on the relation between time, distance from the center and width of the pulse
Purpose	Popular test for high-order methods

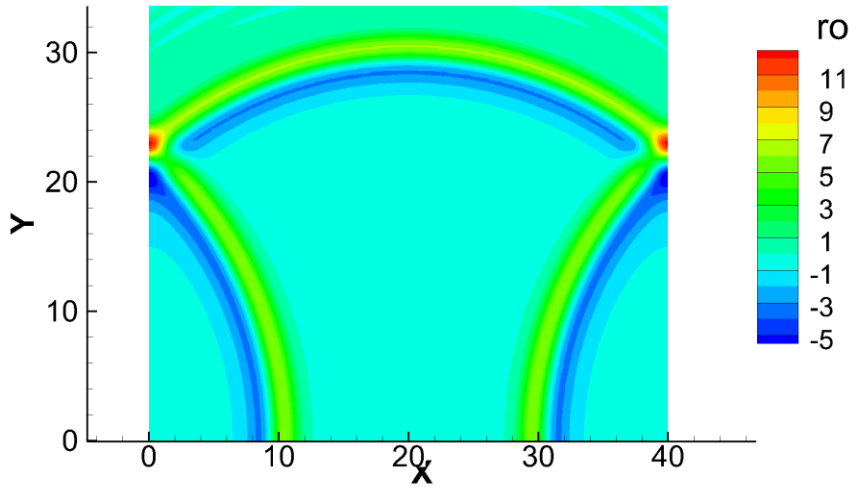


Fig. 7. Solution for 2D problem with the initial data given by 1D lattice of pulses. Symmetry condition is imposed at $y = 0$

s_IVP2D: problem with a local initial pulse with arbitrary profile

Class	s_IVP2D
Math. model	Wave equation (background flow is admissible)
Domain	$t > 0, \mathbf{r} \in \mathbb{R}^2$
Problem setup	Initial data are specified as an axisymmetric pulse $\rho'(0,r) = p'(0,r) = f(r), \quad \mathbf{u}'(0,r) = 0$. $f(r)$ can be either Gaussian (in this case the solution coincides with s_Gaussian2D), or $f(r) = \cos^2(\pi r/(2b))\,\Theta(b-r)$. Lattice of pulses is admissible, which corresponds to periodical boundary conditions
Smoothness	Finitely or infinitely smooth
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	The solution is calculated as a convolution of f with the fundamental solution of the wave equation (2D integral calculation needed)

s_Source2D: acoustical source of a finite size

Class	s_Source2D
Math. model	Wave equation
Domain	$t \in \mathbb{R}, \mathbf{r} \in \mathbb{R}^2$
Problem setup	Source term is given as $S'_p(t,\mathbf{r}) = f(|\mathbf{r}|)\sin(\omega t)$ acting for $t \in (-\infty,\infty)$, and the radiation condition is imposed as $\mathbf{r} \to \infty$. f is as Gaussian. In practice, initial data are prescribed using the exact solution at $t = 0$
Smoothness	Infinitely smooth
Behavior at infinity	**Slowly** vanishes as $\mathbf{r} \to \infty$
Computational complexity	Integral calculation is needed
Purpose	Solution is given for the sake of completeness. It is inconvenient for the verification of numerical methods due to the reflection from the boundaries of the computational domain

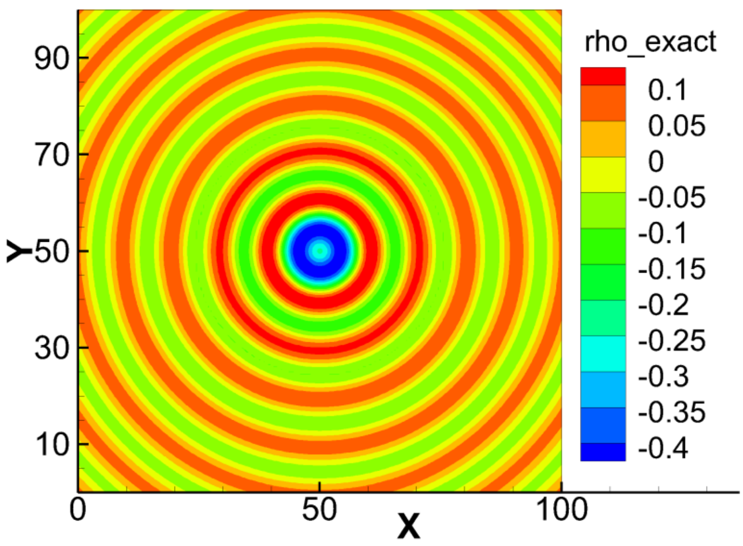


Fig. 8. Wave from a 2D source

Linear three-dimensional problems in free space

s_Gaussian3D: problem with initial pulse with Gaussian profile

Class	s_Gaussian3D
Math. model	Wave equation (background flow is admissible)
Domain	$t > 0, \mathbf{r} \in \mathbb{R}^3$
Problem setup	Initial data are specified as $\rho'(0,r) = p'(0,r) = \exp\left(-\ln 2\,(r/b)^2\right), \quad \mathbf{u}'(0,r) = 0$. Lattice of pulses of this form is admissible, which corresponds to periodical boundary conditions
Smoothness	Infinitely smooth
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions (in contrast with the similar 2D problem)
Purpose	May be used for the verification of both high-accuracy methods and far-field noise prediction models

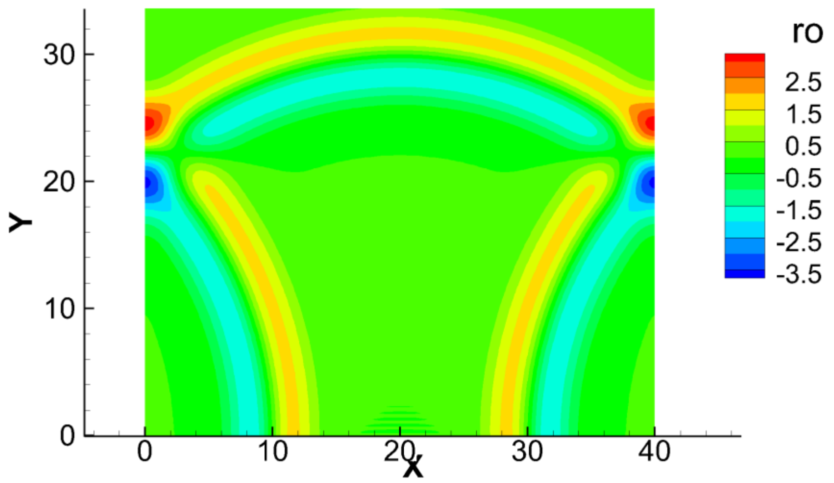


Fig. 9. Solution for 3D problem with the initial data given by 1D lattice of pulses

s_Source3D: acoustical source of a finite size

Class	s_Source3D
Math. model	Wave equation
Domain	$t > 0, \mathbf{r} \in \mathbb{R}^3$
Problem setup	Zero initial data, source term $S'_p(t,r) = f(r)g(t)$. Space form can be Gaussian or $f(r) = \cos^2(\pi r/(2b))\,\Theta(b-r)$; time dependency of the source term is $g(t) = \sin^p(2\pi ft)\,\Theta(t_{\max} - t)\Theta(t - t_{\min})$, where p = 1 or p = 4
Smoothness	Different variants are possible
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	Integral calculation is needed
Purpose	May be used for the verification of both high-accuracy methods and far-field noise prediction models

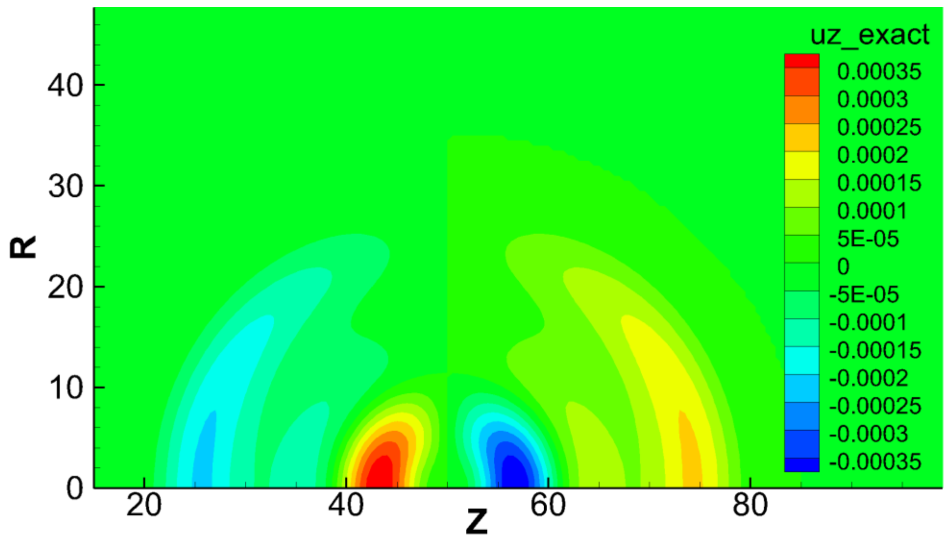


Fig. 10. Solution for the 3D problem with the finite-sized source term for one of the velocity components

s_PointSource: moving point source

Class	s_PointSource
Math. model	Wave equation (background flow is admissible)
Domain	$t > t_{\min}, \mathbf{r} \in \mathbb{R}^3$
Problem setup	Source term $S'_p(t,\mathbf{r}) = \delta(\mathbf{r})g(t)$. The function g($t$) is given by $g(t) = \sin^p(2\pi f t)\,\Theta(t_{\max} - t)\Theta(t - t_{\min})$, where p = 1 or p = 4. At $t = t_{\min}$ the solution is zero. If $t_{\min}$ is less than the moment when the time integration starts, than the initial conditions should be prescribed using the exact solution
Smoothness	At $\mathbf{r} = 0$, $t_{min} < t < t_{max}$ the solution has a singularity; apart from $\mathbf{r} = 0$ the solution is bounded. If we put $t_{max} \leqslant 0$, then during the time integration the solution will be bounded and its smoothness will depend on the smoothness of $g(t)$
Behavior at infinity	Solution has a finite support
Computational complexity	Composition of elementary functions
Purpose	May be used for the verification of both high-accuracy methods and far-field noise prediction models

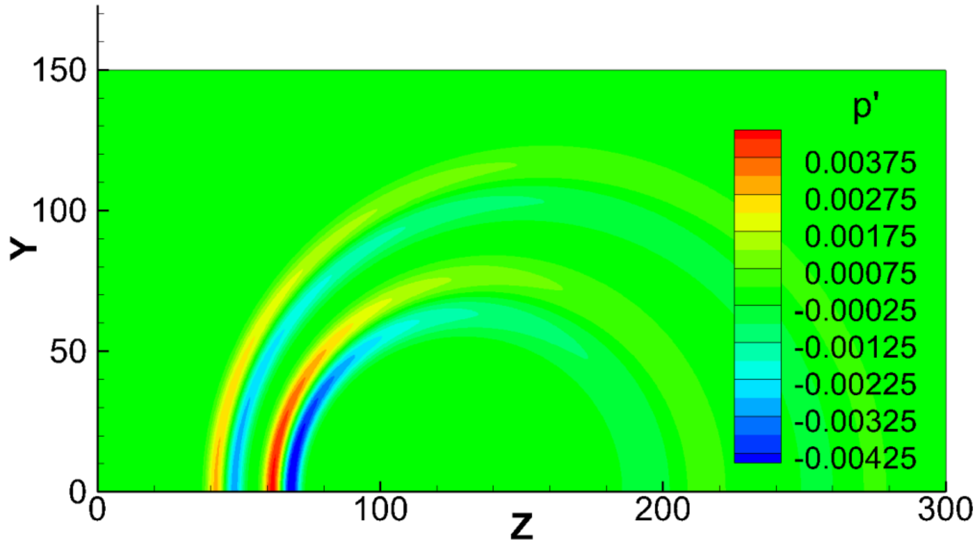


Fig. 11. Solution for the 3D problem with the moving point source

s_RotatingDipole: rotating dipole

Class	s_RotatingDipole
Math. model	Wave equation
Domain	$t \in \mathbb{R}, \mathbf{r} \in \mathbb{R}^3$
Problem setup	Source term is given by $S'_p(t,\mathbf{r}) = -4\pi\nabla\delta(\mathbf{r})\cdot\left(\Omega_{\omega t}\mathbf{e}\right)$, where $\mathbf{e}$ is a vector and $\Omega_{\omega t}$ is a rotation operator on the angle $\lvert\boldsymbol{\omega} t\rvert$ around the axis $\boldsymbol{\omega}/\lvert\boldsymbol{\omega}\rvert$. Radiation condition at $\mathbf{r} \to \infty$
Smoothness	Singularity at $\mathbf{r} = 0$
Behavior at infinity	**Slowly** vanishes as $\mathbf{r} \to \infty$
Solution	The solution for the wave potential has the form [5] $W(t,\mathbf{r}) = -\mathrm{Re}\left[\exp(i\omega t)(1-i\omega\lvert\mathbf{r}\rvert)\dfrac{\mathbf{r}\cdot(\mathbf{e}-i\mathbf{e}')}{\lvert\mathbf{r}\rvert^3}\right]$, where $\mathbf{e}' = [\boldsymbol{\omega}\times\mathbf{e}]/\lvert\boldsymbol{\omega}\rvert$
Purpose	For far-field noise prediction models, particularly, for tonal rotor noise

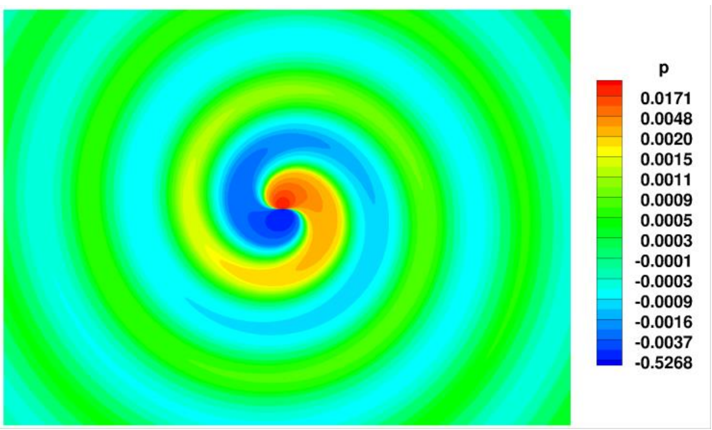


Fig. 12. Solution for the rotating dipole source in the plain crossing the support of the source and orthogonal to its axis

s_Coaxial: acoustical wave between coaxial cylinders

Class	s_Coaxial
Math. model	Wave equation
Domain	$t > 0$, space domain is given in the cylindrical coordinates by the condition $r_{\min} < r < r_{\max}$
Problem setup	Neumann conditions on the cylinder walls. The case $r_{\min} = 0$ is admissible, then the boundedness is imposed at $r = 0$. Single-mode solution is considered, i. e. the density pressure and cylindrical components of the velocity have the form $f(r)\exp(ik_z z + i\nu\phi + i\omega t)$. Axial wave number k_z and azimuthal wave number ν are set by the user, and frequency ω is defined by the dispersion relation
Smoothness	Infinitely smooth
Behavior at infinity	Domain is finite in r, and there are periodical conditions by z
Computational complexity	Composition of elementary and cylindrical functions
Note	If $r = 0$, then the solution is a particular case of s_ViscAcCylinder with zero viscosity and heat conductivity coefficients

Solutions for three problem setups are shown at Fig. 13. In the first problem, zero radial mode was used; in the last two problems, second radial mode was used. On Fig. 13 (c) shown is the sum of two solutions obtained by opposite values of k_z.

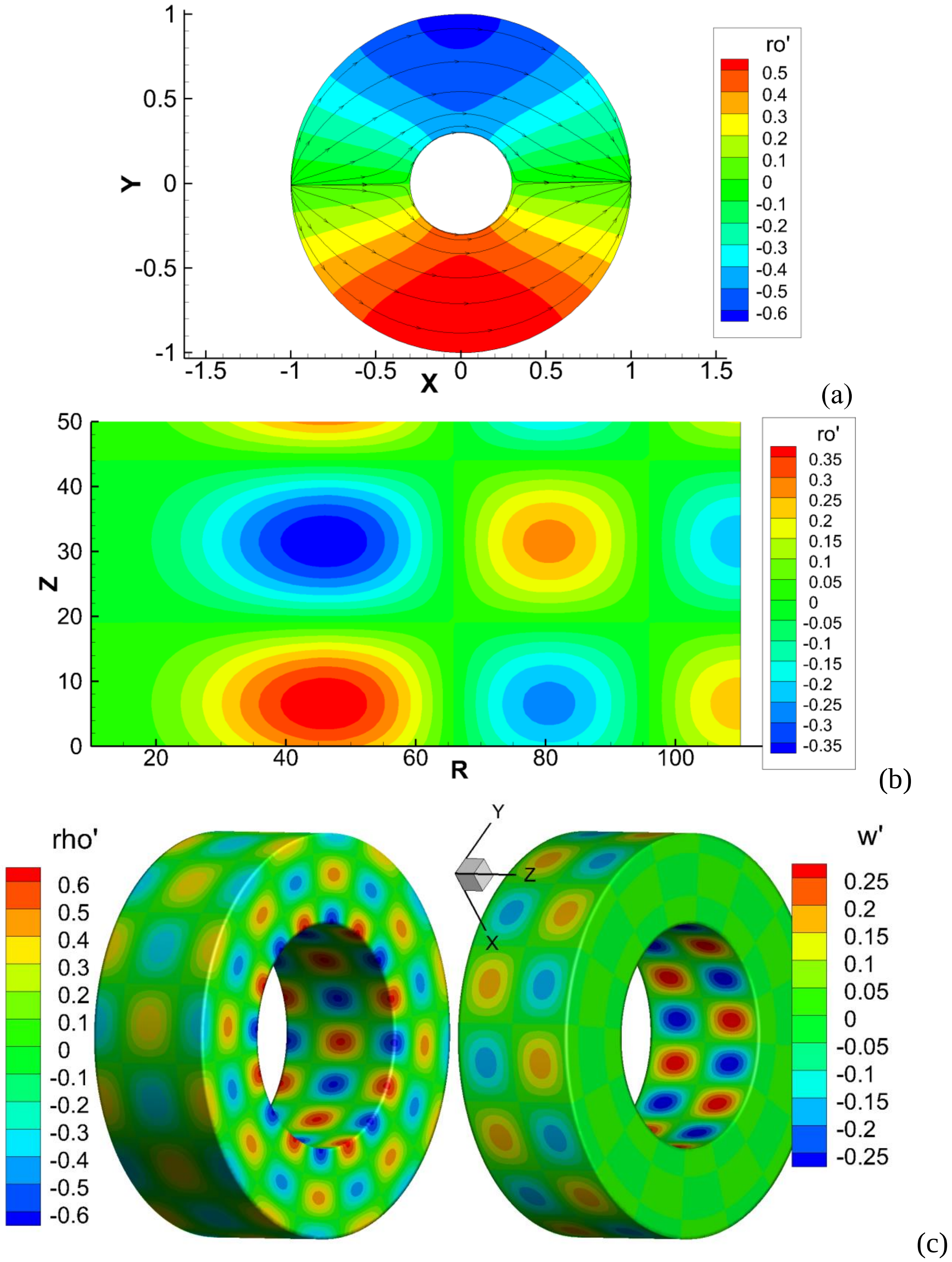


Fig. 13. Wave between coaxial cylinders.
(a): $k_z = 0$, $\nu = 1$, $r_{\min} = 0.3$, $r_{\max} = 1$; (b): $k_z = \pi/25$, $\nu = 4$, $r_{\min} = 10$, $r_{\max} = 110$;
(c): sum of two solutions with $k_z = \pm 3\pi$, $\nu = 8$, $r_{\min} = 5/9$, $r_{\max} = 1$.

Diffraction problems

s_Corner: diffraction on a corner 2π/n

Class	s_Corner
Math. model	Wave equation
Domain	$t > 0$, 2D corner with the angle $2\pi/n$, where n is a natural number. If n = 1, then the domain is a plane without a half-line
Problem setup	On the sector boundary, slip conditions are prescribed. Initial conditions are $\rho'(0,\mathbf{r}) = p'(0,\mathbf{r}) = \exp\left(-\ln 2\,(\lvert\mathbf{r}-\mathbf{r}_0\rvert/b)^2\right), \quad \mathbf{u}'(0,\mathbf{r}) = 0$. The essential support of the initial data (i. e. where the solution is greater than the machine zero) must not intersect with the corner, i. e. on the corner boundaries there holds $\rho'(0,\mathbf{r}) \approx 0$.
Smoothness	Infinitely smooth on each compact set that does not contain a vertex of the corner. At this vertex, the solution is continuous if $n > 1$ (the smoothness depends on n), and for n = 1 there is a singularity for velocities
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	The solution is calculated as the convolution of the Green function (of the wave equation with these boundary conditions) with the initial data. 2D integral calculation is required

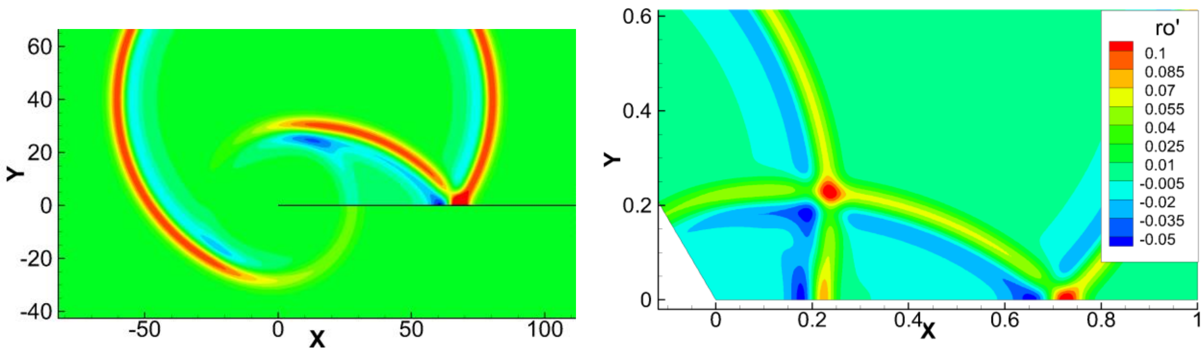


Fig. 14. Diffraction of the wave from a Gaussian pulse. Left: on the half-line (n = 1), right: on a corner (n = 3)

s_CornerPlanar: diffraction of a planar wave a corner 2π/n

Class	s_CornerPlanar
Math. model	Wave equation
Domain	$t > 0$, 2D corner with the angle $2\pi/n$, where n is a natural number. If $n = 1$, then the domain is a plane without a half-line
Problem setup	On the sector boundary, slip conditions are prescribed. Initial conditions prescribe a planar wave with Gaussian profile and possibly its reflections from the corner boundaries. Enumeration of these reflections may be difficult, so it is convenient to specify the initial conditions using the exact solution at $t = 0$
Smoothness	Infinitely smooth on each compact set that does not contain a vertex of the corner. At this vertex, the solution is continuous if $n > 1$ (the smoothness depends on n), and for $n = 1$ there is a singularity for velocities
Behavior at infinity	The solution **does not** vanish as $\mathbf{r} \to \infty$
Computational complexity	Integral calculation is required (1D integral, in contrast to s_Corner)

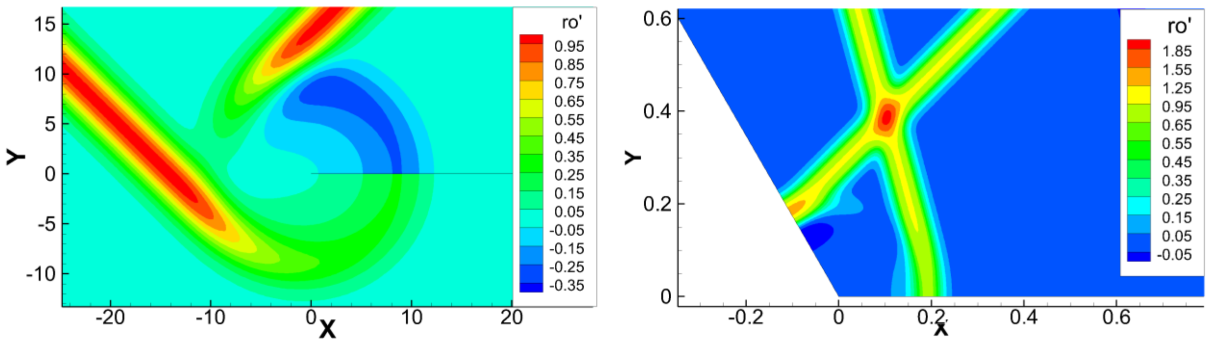


Fig. 15. Diffraction of the planar wave.
Left: on the half-line ($n = 1$), right: on a corner ($n = 3$)

s_Cylinder: diffraction on a cylinder

Class	s_Cylinder
Math. model	Wave equation
Domain	$t > 0$, $\mathbf{r} \in \mathbb{R}^2$, $\lvert\mathbf{r}\rvert > R$
Problem setup	Initial conditions are $\rho'(0,\mathbf{r}) = p'(0,\mathbf{r}) = \exp\left(-\ln 2\,((\mathbf{r}-\mathbf{r}_0)/b)^2\right)$, $\mathbf{u}'(0,r) = 0$. It is assumed that at $\lvert\mathbf{r}\rvert$ = R this function is negligible. At the cylinder boundary Neumann conditions are imposed
Smoothness	Infinitely smooth
Behavior at infinity	Quickly vanishes as $\mathbf{r} \to \infty$
Computational complexity	The solution is calculated as a combination of eigenfunctions, as series by the azimuthal number and an integral by the radial wave number. The computation of the solution may take much time. Due to the inaccuracies of the high-index Bessel function calculation, one can rely on the 5 decimal digits of the solution
Note	This problem was suggested in [6], but the solution in this paper is in error

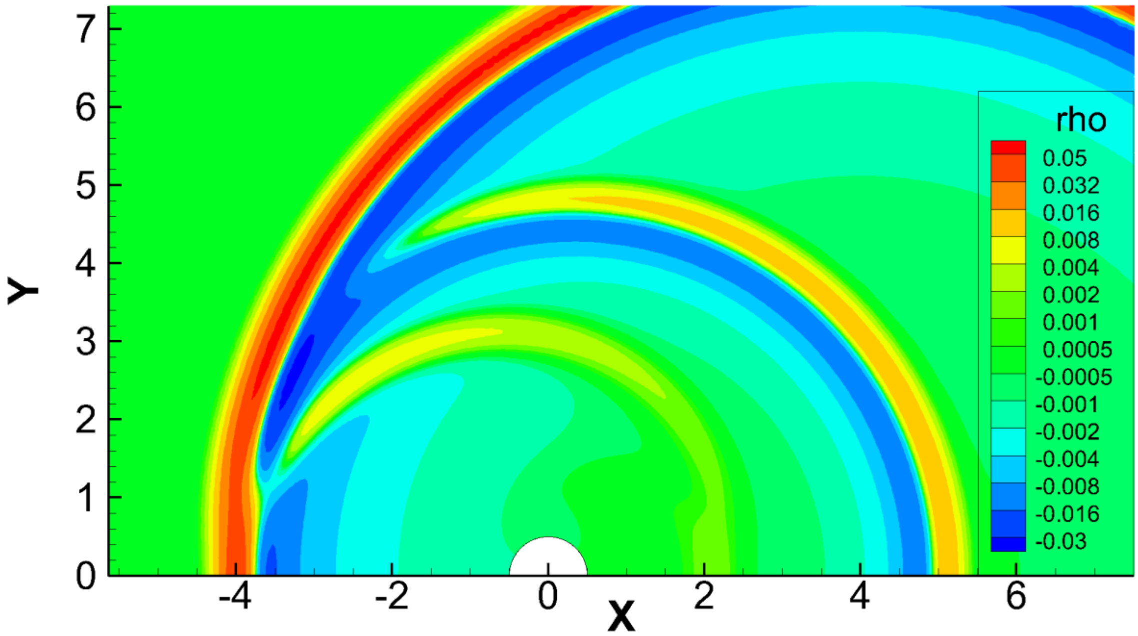


Fig. 16. Diffraction of the initial pulse by a cylinder

Linear perturbations with diffusion

s_SinusVisc: simple wave in free space

Class	s_SinusVisc
Math. model	Linearized Navier – Stokes equations
Domain	$t \in \mathbb{R}, \mathbf{r} \in \mathbb{R}^3$ (the solution depends on one coordinate only)
Problem setup	Single mode is considered, i. e. all the physical pulsations depend on the coordinates as exp(i**k·r**). a) Non-heat-conductive vase. The solution is the sum of a potential and vortex components, amplitudes of which are set by the user. Decrement of both components can be found from the equations. б) Heat-conductive case. Only potential component is considered.
Smoothness	Infinitely smooth
Behavior at infinity	The solution is periodical
Computational complexity	Composition of elementary functions
Purpose	Verification of the viscous and heat fluxes (simple solution of LNSE)
Note	The solution admits 1D and 2D setups; if neither viscosity nor heat conductivity is set. then the solution coincides with the one of s_PlanarSinus.

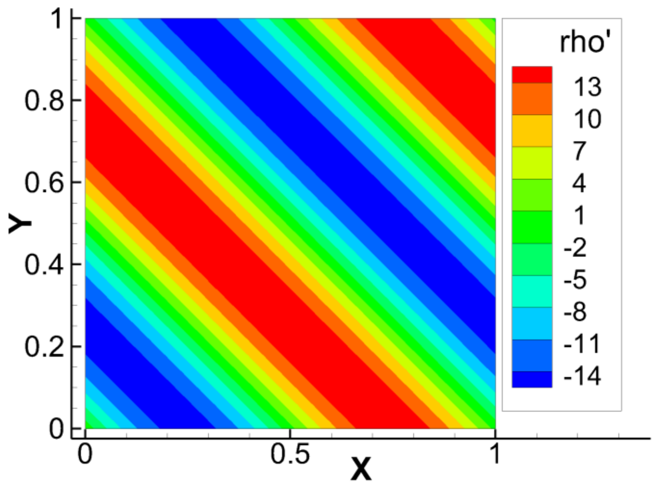


Fig. 17. Periodic cell for the simple wave

s_VortexInCylinder: vortex perturbation inside a circle

Class	s_VortexInCylinder
Math. model	Linearized Navier – Stokes equations with no heat conductivity
Domain	$t > 0, \mathbf{r} \in \mathbb{R}^2, \lvert \mathbf{r} \rvert < R$
Problem setup	No-slip conditions are specified on the circle boundary. Initial conditions are given by $u_\phi(r) = f(r/R)$, where f is a continuous function such that $f(0) = f(1) = 0$; pulsations of radial velocity, density and pressure are zero
Smoothness	Infinitely smooth in $t > \varepsilon > 0$ unless viscosity is zero
Computational complexity	Solution is given by the series of eigenfunctions
Note	If the viscosity is zero, then the initial conditions preserve forever

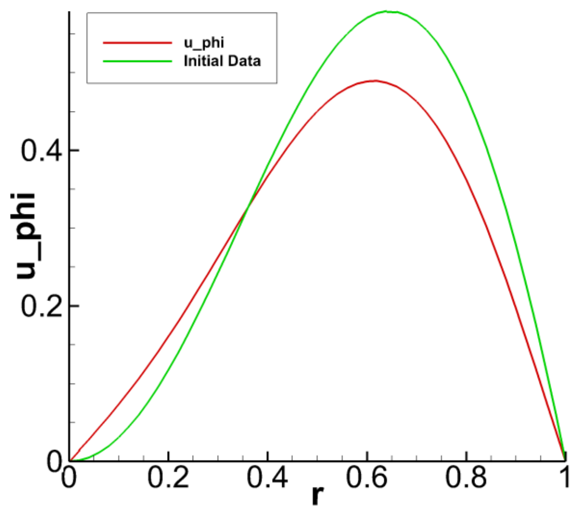


Fig. 18. Profile of the azimuthal velocity at $t = 1$ compared to the initial profile $f(x) = x\sin(\pi x)$

s_WaveInChannel: wave in a square or cylindrical channel

Class	s_WaveInChannel
Math. model	Linearized Navier – Stokes equations
Domain	a) $t \in \mathbb{R}, \mathbf{r} \in \mathbb{R} \times [y_{\min}, y_{\max}];$ b) $t \in \mathbb{R}, \mathbf{r} \in \mathbb{R}^3, x^2 + y^2 < R^2$
Problem setup	On the channel boundary applied are no-slip conditions and (if heat conductivity is nonzero) zero temperature pulsations. We are looking for the single-mode solution. a) Density, pressure, and velocity pulsations are of the form $f(y)\exp(ik_x x + i\omega t)$. Axial wave number ($k_x$) is set by the user. b) Density, pressure, and cylindrical components of the velocity pulsations have the form $f(r)\exp(ik_z z + i\nu\phi + i\omega t)$. Axial ($k_z$) and azimuthal wave numbers are set by the user
Smoothness	Infinitely smooth
Computational complexity	a) composition of elementary functions; б) composition of elementary and cylindrical functions. The primary complexity is to get $\omega \in \mathbb{C}$ by solving a nonlinear algebraic equation at initialization stage. If the viscosity is high, the solution found may not correspond to the mode the user asks for. If during the iteration process ω approaches the imaginary axis, solving the equations for ω usually fails. If the iteration process succeeds, then the solution satisfies (5)
Purpose	Verification of numerical methods for solving high-Reynolds number problems. If viscosity is small, then the solution has a boundary layer with steep gradients on velocity and/or temperature. The case (b) is especially useful for the methods that use curvilinear elements
Note	The following cases are possible: $k_x = 0$ (reducing to a 2D problem), inviscid and/or non-heat-conductive case. If the viscosity and heat conductivity coefficients are small enough, than the solution (b) is close to a steady field in the coordinate frame rotating with $\Omega = -\mathrm{Re}\,\omega / \nu$. This solution was first obtained by G. Kirchgoff [7]; details of implementation in ColESo are described in [8]

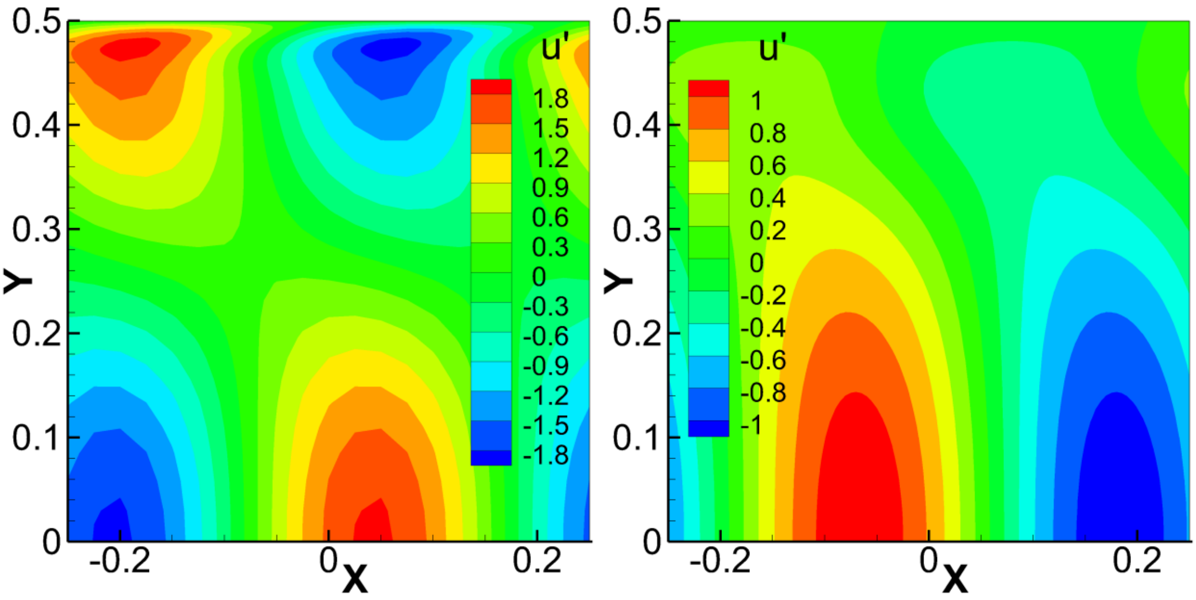
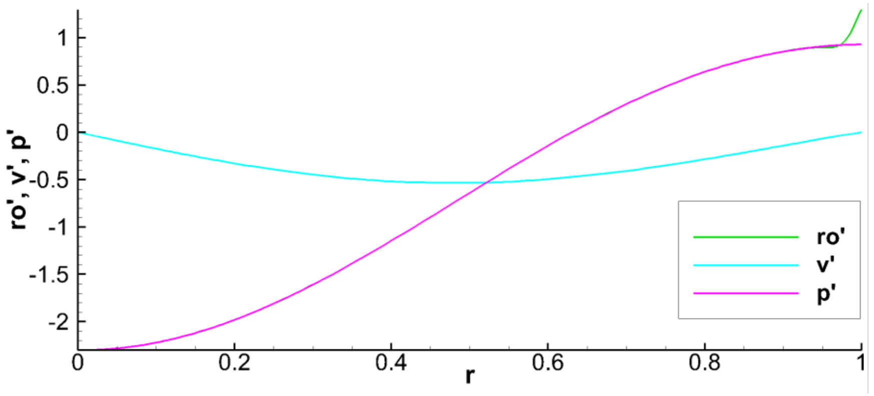
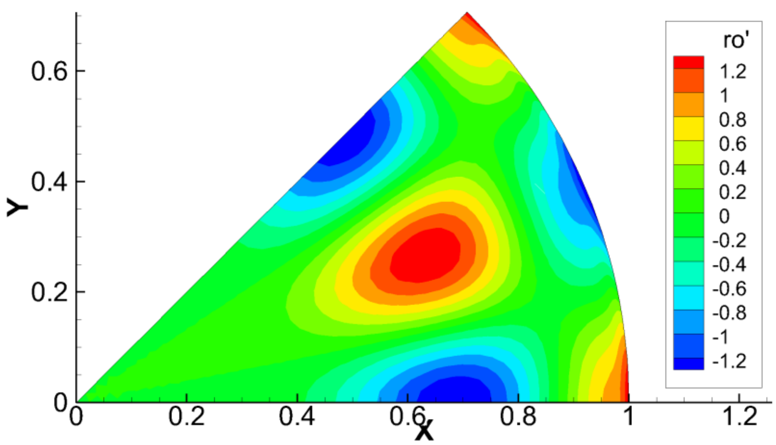


Fig. 19. Wave in a planar channel. Left: $\mu = 10^{-3}$, right: $\mu = 0.02$



a)



б)

Fig. 20. Acoustical mode in a cylindrical channel:
a) axisymmetrical 1st radial mode with $k_z = 0$ and $\mu = 4 \cdot 10^{-4}$, Pr = 1;
б) 8th angular 1st radial mode with $k_z = 0$ and $\mu = 10^{-3}$, Pr = 1.

Planar vortexes

Steady planar vortex is a steady solution of the 2D Euler equations such that density, pressure and azimuthal velocity component depends on r only and the radial component of the velocity is zero. In this case the continuity and energy equations are satisfied automatically, and the momentum equation takes the form

$$\frac{dp}{dr} = \rho \frac{u_\phi^2}{r} . \tag{8}$$

Below we consider isentropic solutions, i. e. assume that $\ln p - \gamma \ln \rho = const$. The profile of the azimuthal velocity can be different (for high velocities, solution may not exist). In all cases we denote by M the ratio of the maximal vortex velocity to the sound speed in the undisturbed medium.

s_FiniteVortex: vortex with the finite velocity profile

Class	s_FiniteVortex
Math. model	Euler equations
Domain	$\mathbf{r} \in \mathbb{R}^2$
Problem setup	Possible velocity profiles: $u_\phi^{(1)}(r) = M\left(\frac{r}{R}\left(2-\frac{r}{R}\right)\right)^n \Theta(2R-r)$ (n is set by the user) or $u_\phi^{(2)}(r) = M\,\frac{r(3R-r)^2}{4R^3}\Theta(3R-r)$. Pressure and density can be found from (8) with the constant entropy constraint
Smoothness	Finitely smooth
Behavior at infinity	Solution is constant outside a compact set
Computational complexity	Composition of elementary functions
Purpose	Verifications of methods for solving Euler equations
Note	Stability of these vortexes were not studied, but the numerical evidence shows that for a rather small M these vortexes are stable

s_GaussianVortex: vortex with the Gaussian circulation profile

Class	s_GaussianVortex
Math. model	Euler equations
Domain	$\mathbf{r} \in \mathbb{R}^2$
Problem setup	Velocity profile $u_\phi(r) = \frac{A_0 MR}{r}\left(1 - \exp\left(-\alpha_0 \frac{r^2}{R^2}\right)\right)$, where $A_0 \sim 1.398$ and $\alpha_0 \sim 1.256$ are the constants such that the velocity profile has the maximum at $r = R$ equal to Mc_∞. Pressure and density can be found from (8) with the constant entropy constraint
Smoothness	Infinitely smooth
Behavior at infinity	**Slowly** tends to a constant value as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions and exponential integral
Purpose	Verifications of the methods for solving Euler equations
Note	Stability analysis provided numerically in [9] shows that this vortex is stable at least for $M \leqslant 1.5$

s_Vortex_BG: vortex with the velocity profile u = r/(1+r^2)

Class	s_Vortex_BG
Math. model	Euler equations
Domain	$\mathbf{r} \in \mathbb{R}^2$
Problem setup	Velocity profile is $u_\phi(r) = \frac{\Gamma}{2\pi}\frac{r}{r^2 + R^2}$. Pressure and density can be found from (8) with the constant entropy constraint
Smoothness	Infinitely smooth
Behavior at infinity	**Slowly** tends to a constant value as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions
Purpose	Verifications of the methods for solving Euler equations

s_RankineVortex: Rankine vortex

Class	s_RankineVortex
Math. model	Euler equations
Domain	$\mathbf{r} \in \mathbb{R}^2$
Problem setup	Velocity profile $u_\phi(r) = Mr, r < 1;\ u_\phi(r) = M / r, r \geqslant 1$. Pressure and density can be found from (8) with the constant entropy constraint
Smoothness	Solution is continuous but not its derivatives
Behavior at infinity	**Slowly** tends to a constant value as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions
Note	This vortex is an unstable steady solution of the Euler equations
Purpose	Is not used for the verification directly

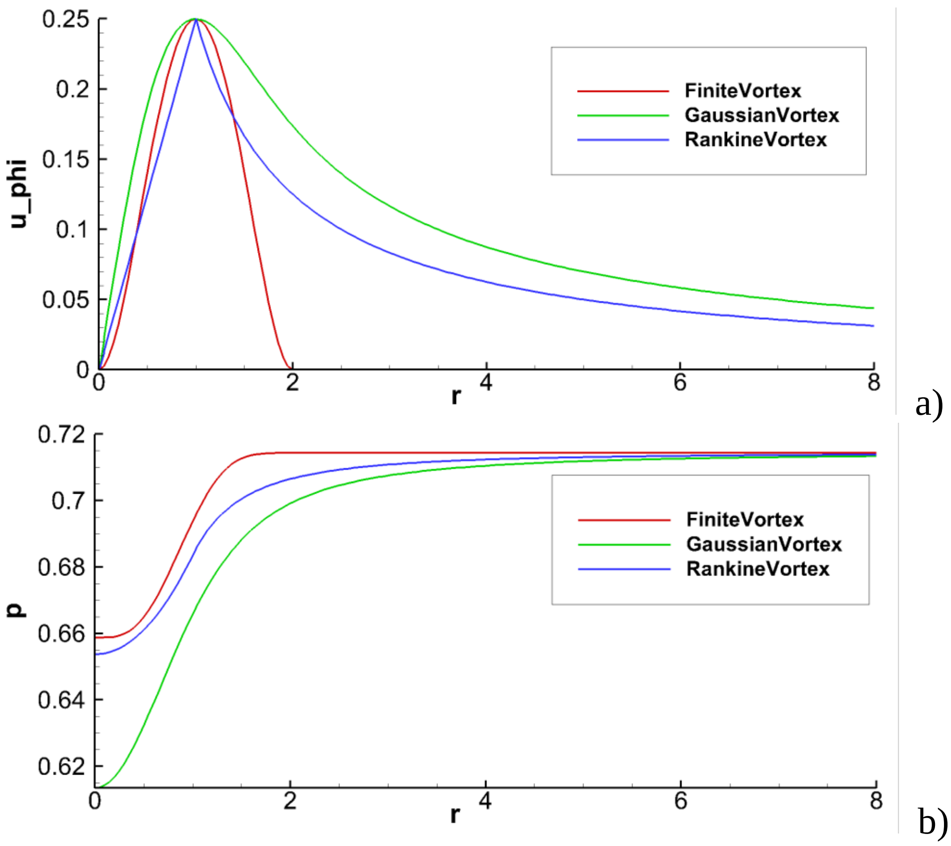


Fig. 21. Planar axisymmetric vortexes.
a) azimuthal velocity; b) pressure.

One-dimensional solutions of the Euler equations

s_Riemann: Riemann problem

Class	s_Riemann
Math. model	Euler equations
Domain	$t > 0,\ x \in \mathbb{R}$
Problem setup	Prescribed are the discontinuity position X_0, values ρ_L, u_L, p_L left to the discontinuity and ρ_R, u_R, p_R right to it
Smoothness	Discontinuous
Behavior at infinity	The solution is constant right to and left to a finite domain (which depends on time)
Computational complexity	In some cases, a quickly converging iterative process is needed
Purpose	Verification and accuracy analysis of the methods for the discontinuous solutions of the Euler equations
Note	Specific ratio is assumed constant. Solution is given in [2] and [10]

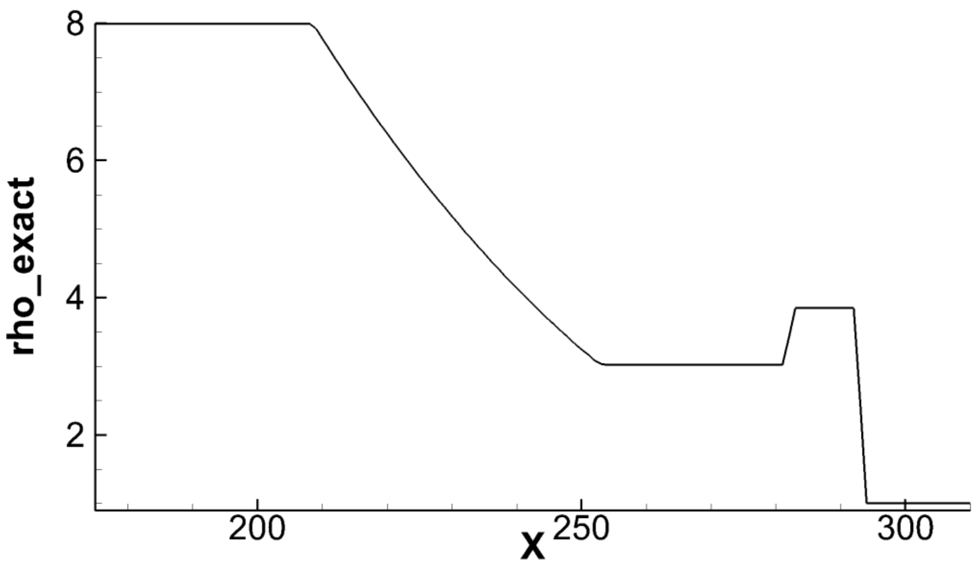


Fig. 22. A solution of the Riemann problem

s_Shock: shock wave

Class	s_Shock
Math. model	Euler equations
Domain	$t > 0,\ x \in \mathbb{R}$
Problem setup	The solution is a shock traveling with a constant speed. Shock is prescribed by the Mach number and parameters in front of the shock
Smoothness	Discontinuous
Behavior at infinity	The solution is constant left to the shock and right to the shock
Computational complexity	Composition of elementary functions
Purpose	Simple test for the verification. Can be also used to prescribe the background field for s_AcousticShock
Note	Particular case of s_Riemann

s_ShockRefl: shock wave reflection from a wall

Class	s_ShockRefl
Math. model	Euler equations
Domain	$t > 0$, $x < x_w$
Problem setup	Initial conditions prescribe a shock traveling to the right. At $x = x_w$ the slip condition $u = 0$ is imposed.
Solution	At each time moment the solution represents a shock. For a small time moment, the shock travels to the right, and for a large time moment, it travels to the left with another speed
Smoothness	Discontinuous
Behavior at infinity	The solution is constant left to the shock and right to the shock
Computational complexity	The solution is a composition of elementary functions.
Purpose	Verification of the slip conditions in the case of discontinuous solutions
Note	Solution is given in [11]

s_SimpleWave: a simple wave

Class	s_SimpleWave
Math. model	Euler equations
Domain	$t>0,\ \ x\in\mathbb{R}$
Solution	Simple wave that corresponds to the left-acoustic Riemann invariant. Thus, physical fields satisfy $$p(t,x)=c_0^2\rho(t,x)/\gamma,\quad u(t,x)=u_0-\frac{2c_0}{\gamma-1}(\rho(t,x)^{(\gamma-1)/2}-1).$$ Initial conditions are given by $$\rho(0,x)=\begin{cases}\left(1+a\exp(-2x^2/(l^2-x^2))\right)^{2/(\gamma-1)}, & \lvert x\rvert<l;\\ 1, & \lvert x\rvert\ge l.\end{cases}$$
Smoothness	Infinitely smooth up to a blow-up time; discontinuous after it
Behavior at infinity	Constant outside a finite set
Computational complexity	Quickly convergent iterative process is used
Purpose	Accuracy analysis of the methods for Euler equations on smooth but essentially nonlinear solutions
Note	Idea of problem setup is borrowed from [12]. **The solution is implemented only up to the shock formation time**

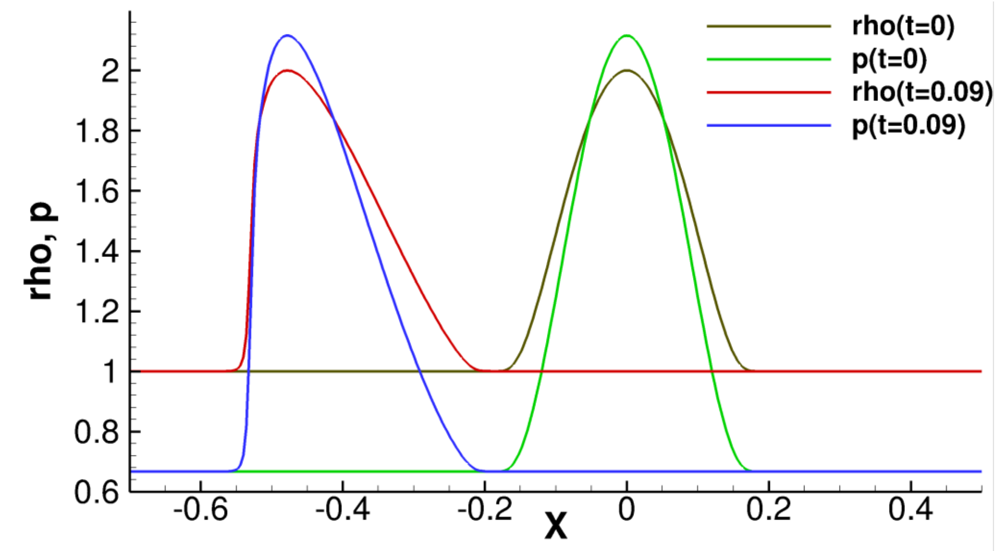


Fig. 23. Simple wave. Solutions for t = 0 and t = 0.09 for $a=\sqrt[3]{2}-1, l=0.2, \gamma=5/3, c_0=\sqrt{10}/3$

Diffusion-dominated solutions

s_ViscShock: profile of a shock wave

Class	s_ViscShock
Math. model	Navier – Stokes equations with or without heat conductivity
Domain	$x \in \mathbb{R}$ or $t > 0$, $x \in \mathbb{R}$
Problem setup	Physical variables (density, velocity, pressure) $\mathbf{Q}_-$ are set as $x \to -\infty$ and $\mathbf{Q}_+$ as $x \to +\infty$. They must prescribe a shock (they must satisfy the Hugoniot conditions, i. e. for some D there holds $\mathcal{F}_c(\mathbf{Q}_L) - D\mathbf{Q}_L = \mathcal{F}_c(\mathbf{Q}_R) - D\mathbf{Q}_R$, and the entropy condition). If D = 0, computations can be provided to find a steady solution. Otherwise the initial conditions should be given by the exact solution, and the time integration should be run for a finite time. Admissible Prandtl numbers: 1) Pr = ∞; 2) Pr = ¾; Admissible viscosity: 1) constant kinematic viscosity coefficient, i. e. $\mu = \rho\nu$ with $\nu = const$; 2) constant dynamic viscosity coefficient, i. e. $\mu = const$.
Smoothness	Infinitely smooth
Behavior at infinity	The solution quickly tends to constant values as $x \to +\infty$ and $x \to -\infty$
Computational complexity	Composition of elementary functions or quickly converging iterative process
Purpose	Verification and accuracy analysis of the numerical methods for solving the Navier – Stokes equations **with no solid walls**
Note	Exact solution is defined up to a shift by *x*. It is given: a) in [2, 13]; b) in [14, 15]; c) in [15]

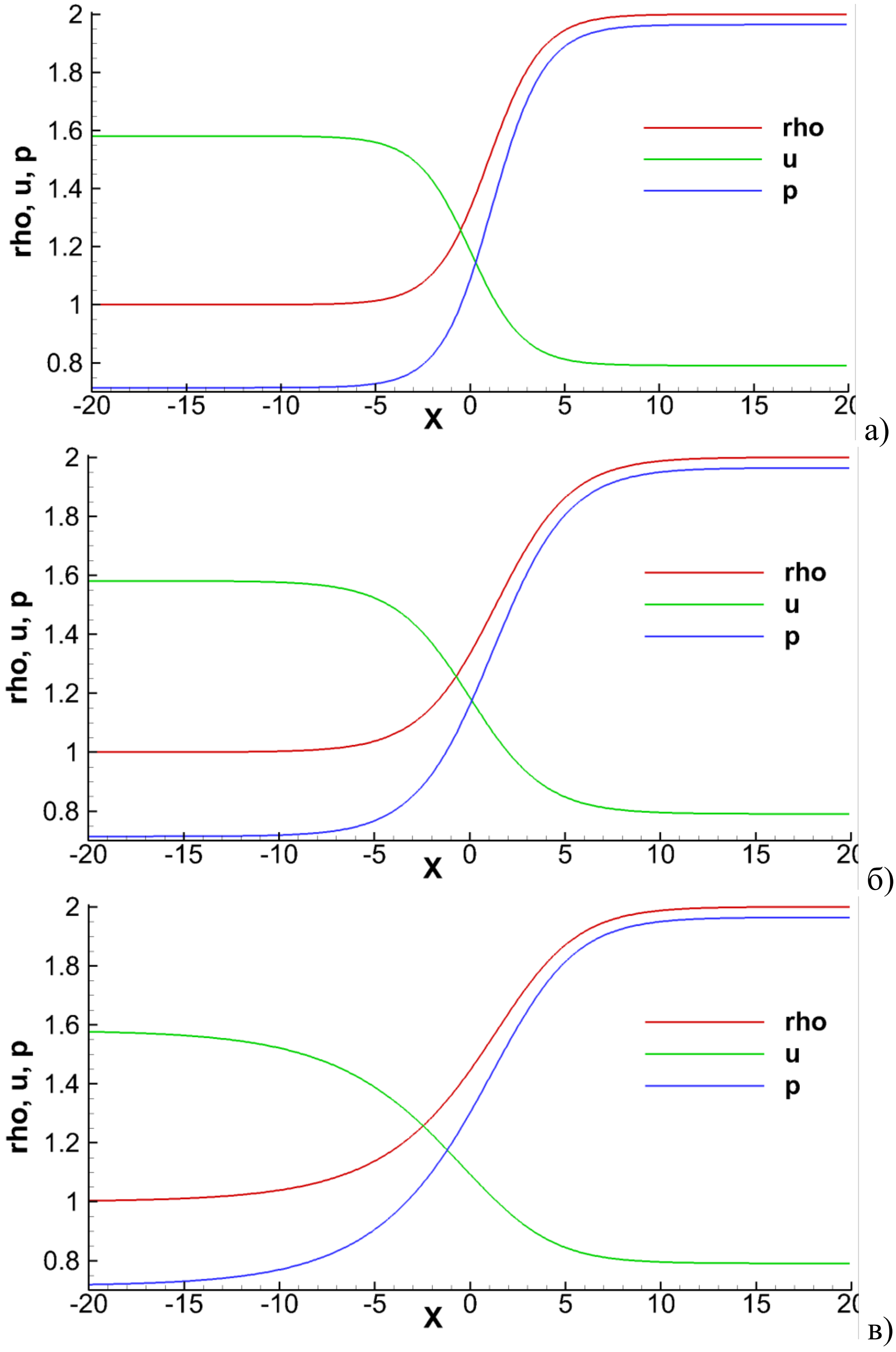


Fig. 24. Shock wave profiles: 1) $\nu = 1$, Pr = ∞; 2) $\nu = 1$, Pr = ¾; 3) $\mu = 1$, Pr=¾.

s_Couette: flow between parallel plates

Class	s_Couette
Math. model	Navier – Stokes equations
Domain	$x_L < x < x_R$, $y \in \mathbb{R}$
Problem setup	At $x = x_L$ and $x = x_R$ noslip conditions are set with, in general, nonzero vertical velocity: $v(x_L, y) = v_L$, $v(x_R, y) = v_R$. For the temperature, either isothermal or adiabatic conditions can be set, but not adiabatic conditions for both walls. The computation is run up to a steady state. Viscosity can depend on temperature as follows a) $\mu = \mu_0 T^\delta$ with δ = 0, ½, ½, 1; b) $\mu = \mu_0 T^{3/2} / (T + C)$ (the Sutherland law)
Smoothness	Infinitely smooth
Behavior at infinity	Domain is bounded in x; the solution does not depend on y
Computational complexity	Composition of elementary functions or quickly converging iterative process
Purpose	Verification and accuracy analysis of the numerical methods for solving the Navier – Stokes equations with noslip conditions
Note	The comparison of the numerical calculations with the exact solutions may be provided by velocity and temperature $T = p/\rho$; pressure is constant in space but can be determined uniquely

s_ConcCyl: flow between coaxial cylinders

Class	s_ConcCyl
Math. model	Navier – Stokes equations
Domain	$\mathbf{r} \in \mathbb{R}^2$, $r_L < \lvert\mathbf{r}\rvert < r_R$
Problem setup	Constant coefficient of the dynamic viscosity. At $\lvert\mathbf{r}\rvert = r_L$ the noslip conditions are set with the velocity $\mathbf{u} = \boldsymbol{\omega}_L \times \mathbf{r}$ and the adiabatic conditions; at $\lvert\mathbf{r}\rvert = r_R$ the velocity $\mathbf{u} = \boldsymbol{\omega}_R \times \mathbf{r}$ and the temperature T_R. are set. Azimuthal velocities $\boldsymbol{\omega}_L$, $\boldsymbol{\omega}_R$ and the temperature T_R are specified by the user
Smoothness	Infinitely smooth
Behavior at infinity	Computational domain is finite
Computational complexity	Composition of elementary functions
Purpose	Verification and accuracy analysis of the numerical methods for solving the Navier – Stokes equations with noslip conditions
Note	The solution is given in [2, 16]. The comparison of the numerical calculations with the exact solutions may be provided by velocity and temperature $T = p/\rho$; pressure and density alone are not determined uniquely

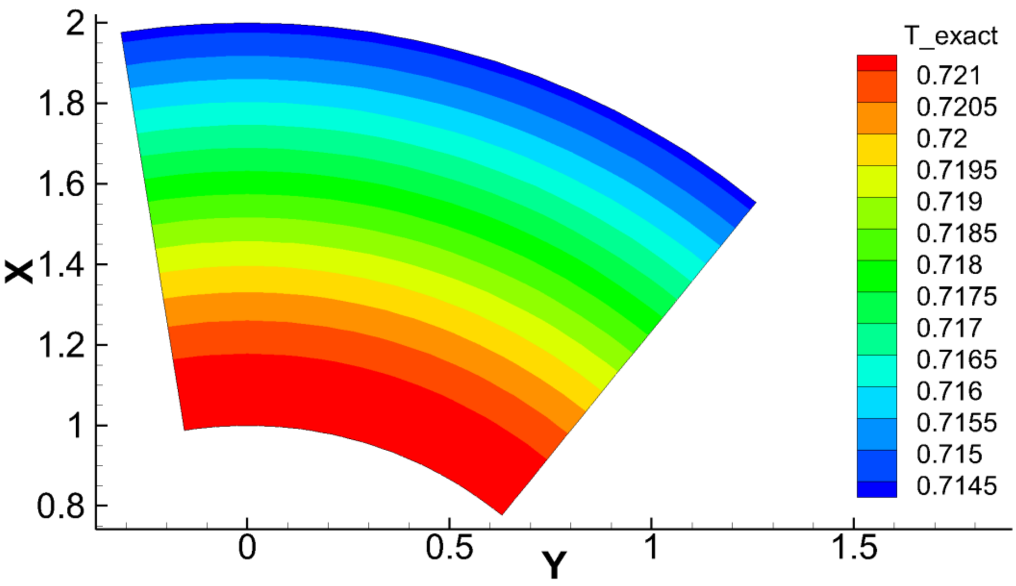


Fig. 25. Temperature distribution between the coaxial cylinders. Computation in the sector π/3 with the angular-periodic boundary conditions

Incompressible flows

s_CurlFreeCylinder: curl-free flow around a cylinder

Class	s_CurlFreeCylinder
Math. model	Euler equations for incompressible fluids
Domain	$\mathbf{r} \in \mathbb{R}^2$, $\lvert\mathbf{r}\rvert > R$
Problem setup	Uniform flow with M << 1 along X axis, noslip conditions on the cylinder surface
Smoothness	Infinitely smooth
Behavior at infinity	The solution **slowly** tends to a constant value as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions
Purpose	Verification of the method for Euler equations for low-speed flows
Note	Solution is given in [2, 17]. The error of this solution for compressible flows is O(M^2). The solution contains a free parameter, namely, the circulation around a cylinder. In numerical calculations, the value of the circulation is generally unpredictable

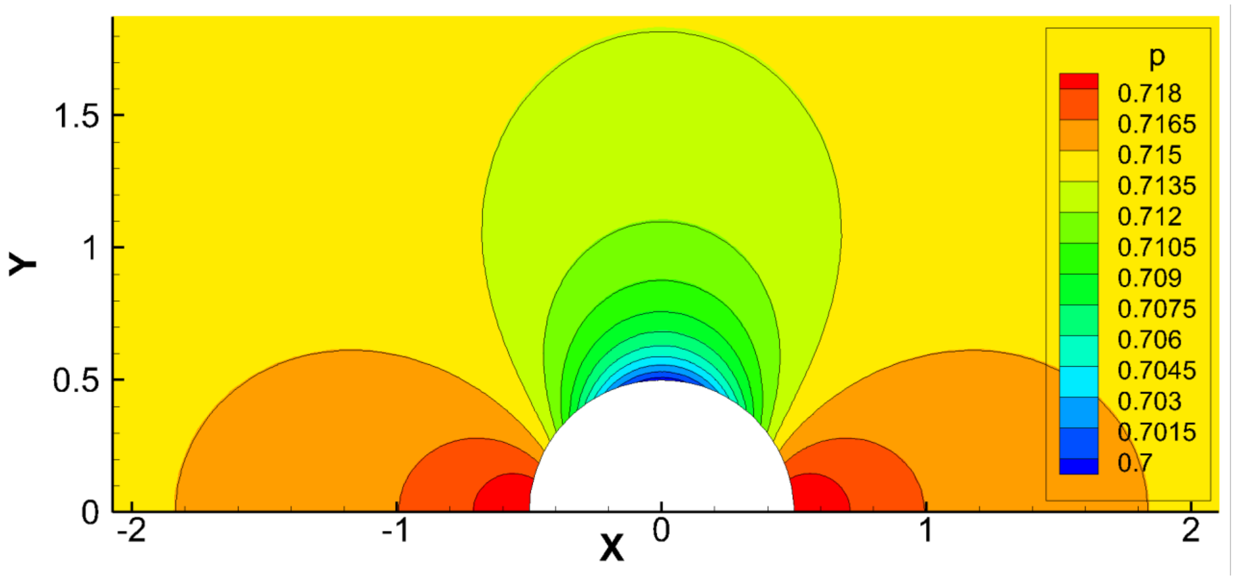


Fig. 26. Curl-free flow around a cylinder

s_PotentialSphere: potential flow around a sphere

Class	s_PotentialSphere
Math. model	Euler equations for incompressible fluids
Domain	$\mathbf{r} \in \mathbb{R}^3$, $\lvert\mathbf{r}\rvert > R$
Problem setup	Uniform flow with $M << 1$ along X axis, noslip conditions on the sphere surface
Smoothness	Infinitely smooth
Behavior at infinity	The solution **slowly** tends to a constant value as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions
Purpose	Verification of the method for Euler equations for low-speed flows
Note	Solution is given in [2]. The error of this solution for compressible flows is $O(M^2)$

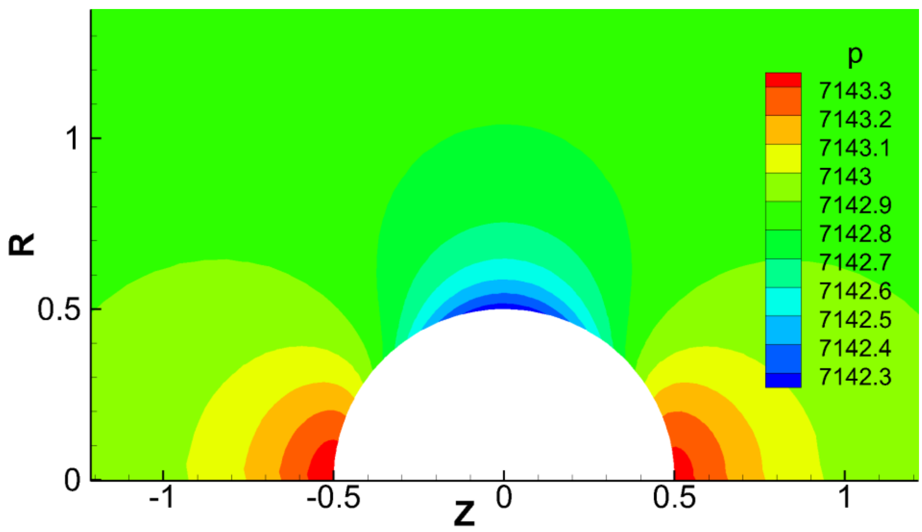


Fig. 27. Potential flow around a sphere

s_ViscSphere: viscous flow around a sphere

Class	s_ViscSphere
Math. model	Navier – Stokes equations for incompressible fluids
Domain	$\mathbf{r} \in \mathbb{R}^3$, $\lvert\mathbf{r}\rvert > R$
Problem setup	Viscosity-dominated flow (Re << 1). Uniform flow with M << 1 along X axis, noslip adiabatic conditions on the sphere surface
Smoothness	Infinitely smooth
Behavior at infinity	The solution **slowly** tends to a constant value as $\mathbf{r} \to \infty$
Computational complexity	Composition of elementary functions
Purpose	Correctness check of a method for the Navier – Stokes system in 3D. Accuracy estimates are hardly possible, because of unknown influence of the compressibility
Note	The solution is given in [2, 18]. It is valid for M << 1 and Re << 1

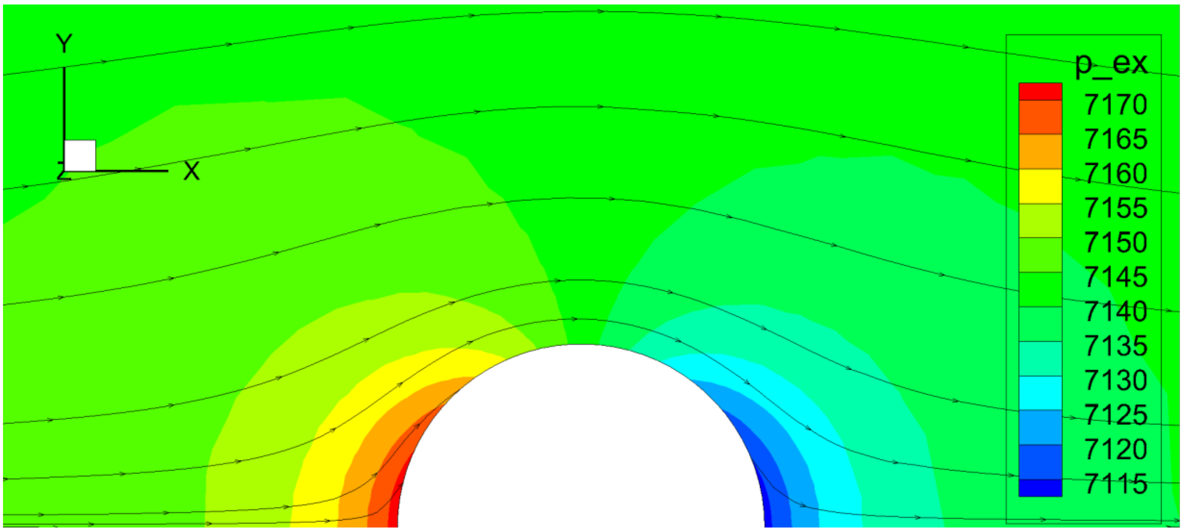


Fig. 28. Viscosity-dominated flow around a sphere

s_Blasius: flow around a semi-infinite plate

Class	s_Blasius
Math. model	Prandtl equation
Domain	$x > 0$, $y = 0$
Problem setup	Uniform flow with $M << 1$ along X axis, noslip adiabatic conditions on the plate ($y = 0$)
Smoothness	Infinitely smooth
Behavior at infinity	No limit as $\mathbf{r} \to \infty$
Computational complexity	At the initialization stage, it is obtained by solving an ODE, and at each call of PointValue( ) this precomputed solution is interpolated.
Note	The solution is exact for the Prandtl equation and valid for the Navier – Stokes equations for large x, see [17]
Purpose	Blasius problem is frequently used for the verification of numerical methods for aerodynamical problems. However, it is not very convenient, because the difference between the solutions of the Prandtl and Navier – Stokes equations may be significant

Summary of the solutions

Summary of the solutions is given in Table 1 (for the linearized equations) and in Table 2 (for the full equations). For each problem, the following information is given.

- model, i. e. the system of equation the solution satisfies. The following notation is used: transp. – transport equation, WE – wave equation, EE – Euler equations, NSE –Navier – Stokes equations, LEE – linearized Euler equations, LNSE – linearized Navier – Stokes equations, IEE – incompressible Euler equations, INSE – incompressible Navier – Stokes equations;
- variables the solution depends on. The following notation is used: $r_{xy} = (x^2 + y^2)^{1/2}$, $r_{xyz} = (x^2 + y^2 + z^2)^{1/2}$. If a variable is written in parentheses, then the dependency on it is optional;
- inequalities prescribing the domain where the solution is defined;
- smoothness of the solution: BV – solution with the bounded variation (generally discontinuous), sing – solution with a singularity. If the smoothness of the solution depends on the parameters, the several possible variants may be given. $C^?$ means a finitely smooth solution, the degree of smoothness depending on the parameters;
- is the solution nearly constant outside a compact set for a finite time moment. If the domain is bounded (including the case that the solution is periodic by each direction) “--" or “per” is written. If the solution is approaching a constant with an exponential rate, “yes” is written, otherwise “no”;
- computational complexity of the solution: ”--" means an explicit formula or a quickly converging iterative process, I –calculation of an integral, S – of a series, I^2 – of a 2D integral, IS – of an integral and a series.

Notes to the Table 1:

- s_PlanarAcousticShock: linearized Euler equation with a discontinuous background field is used. The linearized Hugoniot conditions are imposed taking into account the pulsation of the shock velocity;
- s_WaveInChannel and (in heat-conductive case only) s_SinusVisc: at the initialization stage, solving of a nonlinear algebraic system is required.

Table 1

List of linear solutions

Exact solution	Model	Depends on	Domain	Smooth-ness	Finit-ness	Comp-lexity
s_Polynom	transp.	t, x		C^∞	no	--
s_4peak	transp.	t, x		BV	per	--
s_PlanarSinus, s_PlanarGauss	WE	t, x, (y, z)		C^∞ / BV	yes / no / per	--
s_EntropyVortex	LEE	t, x, y		C^∞	yes	--
s_Source1D	WE	t, x		C^∞	yes	I
s_AcousticShock	LEE*	t, x		BV	yes	--
s_Gaussian2D	WE	t, r_{xy}		C^∞	yes	I
s_IVP2D	WE	t, r_{xy}		$C^?$	yes	I^2
s_Source2D	WE	t, r_{xy}		C^∞	no	I
s_Gaussian3D	WE	t, r_{xyz}		C^∞	yes	--
s_Source3D	WE	t, r_{xyz}		C^∞	yes	I
s_PointSource	WE	t, x, r_{yz}		$C^?$ /sing	yes	--
s_RotatingDipole	WE	t, x, y, z		sing	no	--
s_Coaxial	WE	t, x, y, (z)	$r_{xy} < r_2$ or $r_1 < r_{xy} < r_2$	C^∞	per	--
s_Corner	WE	t, x, y	sector	C^∞ / sing	yes	I^2
s_CornerPlanar	WE	t, x, y	sector	C^∞ /sing	no	I
s_Cylinder	WE	t, x, y	$r_{xy} > r_{min}$	C^∞	yes	IS
s_SinusVisc	LNSE	t, x, (y, z)		C^∞	per	-- *
s_VortexInCylinder	LNSE	t, r_{xy}	$r_{xy} < r_{max}$	C^∞	--	S
s_WaveInChannel	LNSE	t, x, (y, z)	$y_1<y<y_2$ or $r_{yz} < r_{max}$	C^∞	per	-- *

Table 2

List of nonlinear solutions

Exact solution	Model	Depends on	Domain	Smooth-ness	Finit-ness	Comp-lexity
s_RankineVortex	EE	r_{xy}		C	no	--
s_GaussianVortex	EE	r_{xy}		C^{∞}	no	--
s_FiniteVortex	EE	r_{xy}		$C^{?}$	yes	--
s_Riemann, s_Shock	EE	t, x		BV	yes	--
s_ShockRefl	EE	t, x		BV	yes	--
s_SimpleWave	EE	t, x		C^{∞} *	yes	--
s_ViscShock	NSE	x		C^{∞}	yes	--
s_Couette	NSE	x	$x_1<x<x_2$	C^{∞}	--	--
s_ConcCyl	NSE	r_{xy}	$r_1 < r_{xy} < r_2$	C^{∞}	--	--
s_CurlFreeCylinder	IEE	x, y	$r_{xy} > r_{min}$	C^{∞}	no	--
s_PotentialSphere	IEE	x, r_{yz}	$r_{xyz} > r_{min}$	C^{∞}	no	--
s_ViscSphere	INSE, Re << 1	x, r_{yz}	$r_{xyz} > r_{min}$	C^{∞}	no	--
s_Blasius	Prandtl	y/x	x, y > 0	C^{∞}/sing	no	*

Notes to the table 2:

- s_SimpleWave: the solution is smooth up to a finite time moment; after this moment, the exact solution is not implemented in ColESo;
- s_Blasius: the solution is exact for the Prandtl equation; at the initialization stage, it is obtained by solving an ODE, and at each call of PointValue( ) this precomputed solution is interpolated.

III. Usage of ColESo

Program implementation

ColESo library is written in C++ within the standard C++03.

Each exact solution is provided by a class inheriting the class tPointFunction. Abstract class tPointFunction has the following methods.

- `virtual const char* description() const`
  Returns the description of the solution (not more than 80symbols).
- `virtual tFuncType Type() const`
  Returns the type of the solution (see Table 3).
- `int NumVars() const`
  Returns the number of variables defining by the solution (uses Type()).
- `virtual ~tPointFunction(void)`
  Destructor: frees of the dynamic memory if allocated in Init().
- `virtual const char* filename() const`
  Returns the default name of the file to read parameters from.
- `virtual void ReadParams(struct tFileBuffer&)`
  Reading parameters from a file that was previously stored in memory.
- `virtual void ReadParamsFromFile(const char* fname)`
  Reading parameters from a file.
- `virtual void Init()`
  Initializing of the solution. Must me called after prescription or reading of the parameters before the first call of PointValue.
- `virtual void PointValue (double t, const double* coor, double* V) const`
  Returns the solution at time moment 't' and point 'coor'. The subroutine may address three coordinates even if the solution is 1D. Output array V should be of size NumVars( ). Order of variables in the output array is given in Table 3.

A particular solution must override the methods description( ), Type( ), filename( ), ReadParams( ) и PointValue( ). Methods Init( ) and destructor can be overridden if necessary.

Table 3

Types of the solutions and order of returning variables

Type of the solution (Type())	Number of variables (NumVars())	Order of variables in the array returning by PointValue(…)
FUNC_SCALAR	1	V[0] – solution
FUNC_PULSATIONS	5	V[0] – density pulsation, V[1..3] – velocity pulsations, V[4] – pressure pulsation
FUNC_PULSATIONS_ COMPLEX	10	V[0..4] – real components of the pulsations (ordered in analogy with FUNC_PULSATIONS), V[5..9] – imaginary components in the same order
FUNC_PHYSICAL	5	V[0] – density, V[1..3] – velocities, V[4] – pressure
FUNC_TEMPVEL	5	V[0] = 1, V[1..3] – velocities, V[4] – ratio of pressure and density
FUNC_CONSERVATIVE	5	not used in ColESo

ColESo library does not contain non-constant static variables (excluding the interface to use ColESo in C and FORTRAN programs. Hence, it can be used in the multi-thread mode. Correctness of the copy constructors and assignments is also assumed.

ColESo is partially compatible with the double-double and quad-double arithmetics package QD [19]. Some exact solutions are represented by template classes. Parameter of the templates is the floating-point type, which can be double, dd_real и qd_real. In this case, PointValue() takes and returns the values of this type.

Several exact solutions use special functions of the mathematical physics. Their program implementations are borrowed from Cephes and GNU LibQuadMath. To define the Gauss – Jacobi quadrature rules, ColESo uses the code due to J. Burkadrt [22].

Usage examples

To compute an exact solution using ColESo, one should proceed with the following operations.

- Create an object corresponding to the solution in need.
- Specify parameters of the solution or call the function reading them from file.
- Call the function initializing the solution.
- Call the function computing the solution at a point and a time moment given. This function can be called multiple times, at single- or multiple-thread mode

Detailed instruction about the usage of ColESo can be found in README_ENG file attached to the source code.

Example of the program on C++

```
#include "coleso.h"
int main(int, char**) {
// Creating an object corresponding to the problem
// of diffraction of a planar wave
// by a corner 2*Pi/n
s_CornerPlanar S;
// Prescribing problem parameters
const double Pi = 4.0*atan(1.0);
S.angle = 2.*Pi; // angle measure
S.phi0 = Pi / 4.; // wave direction
S.X0 = 0.2; // distance to the wave center at t=0
S.Bterm = 0.025; // half-width of the Gaussian
S.Aterm = 1.0; // Gaussian amplitude
// Initializing the solution
S.Init();
// Defining time and coordinates
double t = 0.4;
double c[3] = {0.005, 0.,0.};
double V[5];
// Calculating the solution and printing it
S.PointValue(t, c, V);
for(int i=0; i<5; i++)
    printf("V[%i] = %25.15e\n”, i, V[i]);
return 0;
}
```

In order to use ColESo in programs on FORTRAN, ColESo contains an interface based on the string variables.

Example of the program on FORTRAN.

```
INCLUDE "coleso_fortran.h"

program main
USE COLESO_FORTRAN
USE, INTRINSIC :: ISO_C_BINDING
CHARACTER(LEN=100, KIND=C_CHAR) :: NAME, VALUE

integer ID
real*8 t, c(3), v(5)

! Create an object corresponding to the problem:
! wave in free space from a 2D Gaussian pulse
NAME = 'Gaussian2D' // C_NULL_CHAR
call coleso_add_function(NAME, ID)

! Store parameters in the ColESo's string buffer
NAME = 'Bterm' // C_NULL_CHAR
write(VALUE,*) 6.0, C_NULL_CHAR
call coleso_set_parameter(NAME, VALUE)
NAME = 'NormalizeForm' // C_NULL_CHAR
write(VALUE,*) 1, C_NULL_CHAR
call coleso_set_parameter(NAME, VALUE)
! Reading parameters from the string buffer
call coleso_read_set(ID)

! Initializing of the solution
call coleso_init(ID)
! Calculating of the solution at time moment t
! and coordinates c
t = 1
c = 0d0
do i = 0, 1100
  c(1) = i * 0.02d0
  call coleso_pointvalue(ID, t, c, v)
  write(*,*) c(1),v
enddo
end
```

Building ColESo and running checks

Building is possible using Makefile. The description of targets is below. All libraries and executables will be created in ./bin.

qd-package	Build QD library
lib	Build ColESo library with double precision only
libqd	Build ColESo library with double, double-double and quad-double precision
examplecpp	Build a simple program on C++ that uses ColESo
examplec	Build a simple program on C that uses ColESo
examplef	Build a simple program on FORTRAN that uses ColESo
examplecppqd	Build a simple program on C++ that uses ColESo with extra precision
testcpp	Build a program on C++ that dumps data needed to plot figures
testcpprun	The same, and run it, and plot figures using gnuplot
testc	Build a program on C that dumps data needed to plot figures
testcrun	The same, and run it, and plot figures using gnuplot
testf	Build a program on FORTRAN that dumps data needed to plot figures
testfrun	The same, and run it, and plot figures using gnuplot
check	Run checks that the solutions satisfy the equations they should satisfy
checkqd	For the solution for wave in a circular channel, check the solution satisfies linearized Navier – Stokes equations with double, double-double, and quad-double precision
clean	Removes all output directories (bin, test/DATA1D, test/DATA2D)

For Miscrosoft Visual Studio users, we enclose a solution containing two simple programs on C++, one of them uses double precision and another one uses double-double precision. To build the second one, the user should define the corresponding flag (#define EXTRAPRECISION_COLESO) in src/personal.h and rebuild the ColESo library.

Bibliography

1. Katate Masatsuka. *I do like CFD*, Vol.1. 304 p.
2. M. A. Isakovich. *General Acoustics*. Nauka, Moscow, 1973. (in Russian)
3. Tam C. K. W., Hardin J. C. Second Computational Aeroacoustics (CAA) Workshop on Benchmark Problems: Proceedings of a workshop ... held in Tallahassee, Florida, November 4-5, 1996. Hampton, Va: National Aeronautics and Space Administration, Langley Research Center.
4. Kirchhoff G. *Uber der Einfluss der Warmeleitung in einem Gase auf die Schallbewegung* // Annalen der Physik und Chemie. 1868. Vol. 134, no. 6. P. 177–193.
5. P. A. Bakhvalov. *Sound wave in an infinite circular cylinder in the presence of viscosity and heat conductivity* // Keldysh Institute Preprints. 2017. № 135. 32 p. (in Russian)
6. Chan W. Shariff K. Pulliam T. *Instabilities of 2D inviscid compressible vortices* // J. Fluid Mech. 1993. Vol. 253. pp. 173–209.
7. S. K. Godunov, A. V. Zabrodin, M. Ya. Ivanov, A. N. Kraiko, G. P. Prokopov. *Numerical solution of multidimensional problems of gas dynamics*. Moscow: Nauka, 1976. (in Russian)
8. K. P. Stanyukovich, *Unsteady motion of Continuous Media*. XV + 745 S. London/Oxford/Paris/New York 1960. Pergamon Press.
9. Ladonkina M. E., Neklyudova O. A., Tishkin V. F. *The high order limiter for RKDG on triangular meshes* // Keldysh Institute Preprints. 2013. № 53. 26 p. (in Russian)
10. B. L. Rozdestvenskii, N. N. Janenko. *Systems of Quasilinear Equations and Their Applications to Gas Dynamics* (Translations of Mathematical Monographs). American Mathematical Society; 2nd Edition. 1983.
11. Becker R. *Stosswelle und Detonation* // Z. Physik. 1922. Vol. 8. P. 321–362.
12. Johnson B. M. *Closed-form shock solutions* // J. Fluid. Mech. 2014. Vol. 745. p. R1.
13. N. E. Kochin, I. A. Kibel, N. V. Roze. *Theoretical Hydromechanics*. John Wiley & Sons, Ltd. 1964
14. L. G. Loitsyanskii, *Mechanics of Liquids and Gases*. (International Series of Monographs in Aeronautics and Astronautics, Vol. 6) XII + 804 S. 1966. Pergamon Press.
15. L. D. Landau, E. M. Lifshitz. *Course of Theoretical Physics*, Volume 6: Fluid Mechanics. Elsevier, 2013.
16. Yozo Hida, Xiaoye S. Li, David H. Bailey et al. *QD – A C++/Fortran-90 double-double and quad-double package*. URL: https://github.com/aoki-t/QD
17. Gauss-Jacobi Quadrature Rules. URL: people.sc.fsu.edu/~jburkardt/cpp_src/jacobi_rule/jacobi_rule.html

Contents