Numerical Simulation of Aerodynamic Noise with DLR's aeroacoustic code PIANO[1]

Jan W. Delfs, Marcus Bauer[2], Roland Ewert[3], Herwig A. Grogger[4], Markus Lummer[5] and Thomas G. W. Lauke[6]

Deutsches Zentrum für Luft- und Raumfahrt e. V.
in der Helmholtz-Gemeinschaft

Institut für Aerodynamik und Strömungstechnik
Abteilung Technische Akustik
Lilienthalplatz 7
38108 Braunschweig

January 2008

[1]**P**erturbation **I**nvestigation of **A**erodynamic **No**ise
[2]parts concerning SNGR
[3]parts concerning APE and RPM
[4]initial version
[5]parts concerning interpolation
[6]corresponding author

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The code PIANO, developed at the Institute of Aerodynamics and Flow Technology (Division Technical Acoustics), is designed to simulate aeroacoustic noise generation and acoustic wave propagation in non-uniform flows. It is based on the equations governing the inviscid dynamics of perturbations to a given time-averaged mean flow field. PIANO simulates the noise generation process, when vorticity interacts with solid structures or gradients in the flow field. PIANO is not designed to calculate wave propagation over very large distances in uniform flows, like from aircraft to ground. In that case methods based on wave equations are more appropriate.
Currently, PIANO is based on structured, curvilinear multi-block grids. The coding structure is designed for high performance on workstations, vector machines, PCs or PC clusters as well.
PIANO is a research code and permanently under construction. Therefore, there is no guarantee that it comes without bugs.

Chapter 2

Governing Equations

2.1 Euler Equations

The unsteady Euler equations are able to support vorticity, entropy and pressure waves. Hence they are appropriate to describe the generation and propagation of aeroacoustic sound. The non-dimensional equations write as follows:

$$\begin{aligned} \frac{\partial \varrho}{\partial t} + \vec{v}\cdot\nabla\varrho + \varrho\,\nabla\cdot\vec{v} &= 0\,, \\ \frac{\partial \vec{v}}{\partial t} + \vec{v}\cdot\nabla\vec{v} &= -\frac{1}{\varrho}\,\nabla p\,, \\ \frac{\partial p}{\partial t} + \vec{v}\cdot\nabla p + \kappa\, p\,\nabla\cdot\vec{v} &= 0\,, \end{aligned} \tag{2.1a}$$

equivalent to

$$\begin{aligned} \frac{\partial \varrho}{\partial t} + v_i\frac{\partial \varrho}{\partial x_i} + \varrho\frac{\partial v_i}{\partial x_i} &= 0\,, \\ \frac{\partial v_j}{\partial t} + v_i\frac{\partial v_j}{\partial x_i} &= -\frac{1}{\varrho}\,\frac{\partial p}{\partial x_j}\,, \\ \frac{\partial p}{\partial t} + v_i\frac{\partial p}{\partial x_i} + \kappa\, p\,\frac{\partial v_i}{\partial x_i} &= 0\,. \end{aligned} \tag{2.1b}$$

The energy equation has been derived from the assumption of isentropic flow and a caloric perfect gas (with the isentropic exponent of ideal gas κ). The reference quantities used for the non-dimensionalization are the chord-length L, ambient speed of sound a_∞, ambient pressure p_∞ and density ϱ_∞ (* denotes quantity with dimension):

$$t = t^*\frac{a_\infty}{L}\,, \quad x_i = \frac{x_i^*}{L}\,, \quad \varrho = \frac{\varrho^*}{\varrho_\infty}\,, \quad \vec{v} = \frac{\vec{v}^*}{a_\infty}\,, \quad p = \frac{p^*}{\varrho_\infty a_\infty^2}\,.$$

Since most aeroacoustic problems are characterized by small amplitude fluctuations about a steady mean flow field, the simulation is limited to the dynamics of perturbations, including all linear processes like wave propagation, scattering and interaction issues. For further transformations the primitive variable ϕ will be splitted into steady mean-flow quantity $\bar{\phi}$ or ϕ_0 and fluctuation ϕ', i. e. $\phi = \bar{\phi} + \varepsilon\phi'$ equivalent to $\vec{\phi} = \vec{\phi}_0 + \varepsilon\vec{\phi}'$. Further, an appropriate function N will be defined for each equation, such that $N(\varepsilon) = 0$ represents the solution of (2.1). In

particular following definitions are obtained:

$$\begin{aligned}
N_\varrho(\varepsilon) &:= \frac{\partial(\bar\varrho+\varepsilon\varrho')}{\partial t} + (\bar v_i+\varepsilon v_i')\frac{\partial(\bar\varrho+\varepsilon\varrho')}{\partial x_i} + (\bar\varrho+\varepsilon\varrho')\frac{\partial(\bar v_i+\varepsilon v_i')}{\partial x_i}\,,\\
N_v(\varepsilon) &:= \frac{\partial(\bar v_j+\varepsilon v_j')}{\partial t} + (\bar v_i+\varepsilon v_i')\frac{\partial(\bar v_j+\varepsilon v_j')}{\partial x_i} + \frac{1}{(\bar\varrho+\varepsilon\varrho')}\,\frac{\partial(\bar p+\varepsilon p')}{\partial x_j}\,,\\
N_p(\varepsilon) &:= \frac{\partial(\bar p+\varepsilon p')}{\partial t} + (\bar v_i+\varepsilon v_i')\frac{\partial(\bar p+\varepsilon p')}{\partial x_i} + \kappa\,(\bar p+\varepsilon p')\,\frac{\partial(\bar v_i+\varepsilon v_i')}{\partial x_i}\,.
\end{aligned} \tag{2.2}$$

Expanding $N(\varepsilon)$ in a TAYLOR's series

$$N(\varepsilon) \approx N(0) + \varepsilon\frac{\partial N}{\partial\epsilon}|_{\epsilon=0} + \frac{\varepsilon^2}{2}\frac{\partial^2 N}{\partial\epsilon^2}|_{\epsilon=0} + \mathcal{O}(\varepsilon^3)$$

with

$$\begin{aligned}
\frac{\partial N_\varrho}{\partial\varepsilon} &= \frac{\partial\varrho'}{\partial t} + v_i'\frac{\partial(\bar\varrho+\varepsilon\varrho')}{\partial x_i} + (\bar v_i+\varepsilon v_i')\frac{\partial\varrho'}{\partial x_i} + \varrho'\frac{\partial(\bar v_i+\varepsilon v_i')}{\partial x_i} + (\bar\varrho+\varepsilon\varrho')\frac{\partial v_i'}{\partial x_i}\,,\\
\frac{\partial^2 N_\varrho}{\partial\varepsilon^2} &= 2\left(v_i'\frac{\partial\varrho'}{\partial x_i} + \varrho'\frac{\partial v_i'}{\partial x_i}\right)\,,\\
\frac{\partial N_v}{\partial\varepsilon} &= \frac{\partial v_j'}{\partial t} + v_i'\frac{\partial(\bar v_j+\varepsilon v_j')}{\partial x_i} + (\bar v_i+\varepsilon v_i')\frac{\partial v_j'}{\partial x_i} + \frac{1}{(\bar\varrho+\varepsilon\varrho')}\,\frac{\partial p'}{\partial x_j} - \frac{\varrho'}{(\bar\varrho+\varepsilon\varrho')^2}\,\frac{\partial(\bar p+\varepsilon p')}{\partial x_j}\,,\\
\frac{\partial^2 N_v}{\partial\varepsilon^2} &= 2\left(v_i'\frac{\partial v_j'}{\partial x_i}\right) - \frac{2\varrho'}{(\bar\varrho+\varepsilon\varrho')^2}\,\frac{\partial p'}{\partial x_j} + \frac{2\varrho'^2}{(\bar\varrho+\varepsilon\varrho')^3}\,\frac{\partial(\bar p+\varepsilon p')}{\partial x_j}\,,\\
\frac{\partial N_p}{\partial\varepsilon} &= \frac{\partial p'}{\partial t} + v_i'\frac{\partial(\bar p+\varepsilon p')}{\partial x_i} + (\bar v_i+\varepsilon v_i')\frac{\partial p'}{\partial x_i} + \kappa\,p'\,\frac{\partial(\bar v_i+\varepsilon v_i')}{\partial x_i} + \kappa(\bar p+\varepsilon p')\,\frac{\partial v_i'}{\partial x_i}\,,\\
\frac{\partial^2 N_p}{\partial\varepsilon^2} &= 2\left(v_i'\frac{\partial p'}{\partial x_i} + \kappa\,p'\,\frac{\partial v_i'}{\partial x_i}\right)
\end{aligned} \tag{2.3}$$

yields

$$\begin{aligned}
&\frac{\partial\varrho'}{\partial t} + v_i'\frac{\partial\bar\varrho}{\partial x_i} + \bar v_i\frac{\partial\varrho'}{\partial x_i} + \varrho'\frac{\partial\bar v_i}{\partial x_i} + \bar\varrho\frac{\partial v_i'}{\partial x_i} + \varepsilon\left(v_i'\frac{\partial\varrho'}{\partial x_i} + \varrho'\frac{\partial v_i'}{\partial x_i}\right) + \mathcal{O}(\varepsilon^2) &&= 0\,,\\
&\frac{\partial v_j'}{\partial t} + v_i'\frac{\partial\bar v_j}{\partial x_i} + \bar v_i\frac{\partial v_j'}{\partial x_i} + \frac{1}{\bar\varrho}\left(\frac{\partial p'}{\partial x_j} - \frac{\varrho'}{\bar\varrho}\,\frac{\partial\bar p}{\partial x_j}\right) + \varepsilon\left(v_i'\frac{\partial v_j'}{\partial x_i} - \frac{\varrho'}{\bar\varrho^2}\left[\frac{\partial p'}{\partial x_j} - \frac{\varrho'}{\bar\varrho}\,\frac{\partial\bar p}{\partial x_j}\right]\right) + \mathcal{O}(\varepsilon^2) &&= 0\,,\\
&\frac{\partial p'}{\partial t} + v_i'\frac{\partial\bar p}{\partial x_i} + \bar v_i\frac{\partial p'}{\partial x_i} + \kappa\,p'\,\frac{\partial\bar v_i}{\partial x_i} + \kappa\,\bar p\,\frac{\partial v_i'}{\partial x_i} + \varepsilon\left(v_i'\frac{\partial p'}{\partial x_i} + \kappa\,p'\,\frac{\partial v_i'}{\partial x_i}\right) + \mathcal{O}(\varepsilon^2) &&= 0
\end{aligned} \tag{2.4}$$

for appropriate $\bar\varrho, \bar v_j, \bar p$ (satisfying EULER's equation, i. e. $N(0) = 0$). In (2.4) the first and second order terms of the expansion are written explicitly: For $\varepsilon \longrightarrow 0$ the linear perturbation equations are obtained; for weakly nonlinear perturbations, the terms $\mathcal{O}(\varepsilon^2)$ are neglected and ε is specified as a small number (typically $0 < \varepsilon < 1$). From now on the terms $\mathcal{O}(\varepsilon^2)$ are neglected.
Since steady mean-flow is assumed, using

$$\frac{\partial\bar v_j}{\partial t} = -\frac{1}{\bar\varrho}\,\frac{\partial\bar p}{\partial x_j} - \bar v_i\frac{\partial\bar v_j}{\partial x_i} = 0$$

(2.4) on the preceding page may be written as

$$\begin{aligned}
\frac{\partial \varrho'}{\partial t} + v_i' \frac{\partial \bar{\varrho}}{\partial x_i} + \frac{\partial \varrho'}{\partial x_i}(\bar{v}_i + \varepsilon v_i') + \frac{\partial \bar{v}_i}{\partial x_i}\varrho' + \frac{\partial v_i'}{\partial x_i}(\bar{\varrho} + \varepsilon \varrho') &= 0 \,, \\
\frac{\partial v_j'}{\partial t} + v_i' \frac{\partial \bar{v}_j}{\partial x_i} + \frac{\partial v_j'}{\partial x_i}(\bar{v}_i + \varepsilon v_i') + \frac{1}{\bar{\varrho}}\left(1 - \varepsilon \frac{\varrho'}{\bar{\varrho}}\right)\left(\frac{\partial p'}{\partial x_j} + \varrho'\, \bar{v}_i \frac{\partial \bar{v}_j}{\partial x_i}\right) &= 0 \,, \\
\frac{\partial p'}{\partial t} + v_i' \frac{\partial \bar{p}}{\partial x_i} + \frac{\partial p'}{\partial x_i}(\bar{v}_i + \varepsilon v_i') + \kappa \left[\frac{\partial \bar{v}_i}{\partial x_i}\, p' + \frac{\partial v_i'}{\partial x_i}(\bar{p} + \varepsilon p')\right] &= 0 \,,
\end{aligned} \tag{2.5a}$$

being equivalent to

$$\begin{aligned}
\frac{\partial \varrho'}{\partial t} + \vec{v}' \cdot \nabla \varrho_0 + (\vec{v}_0 + \varepsilon \vec{v}') \cdot \nabla \varrho' + \nabla \cdot \vec{v}_0\, \varrho' + \nabla \cdot \vec{v}' (\varrho_0 + \varepsilon \varrho') &= 0 \,, \\
\frac{\partial \vec{v}'}{\partial t} + \vec{v}' \cdot \nabla \vec{v}_0 + (\vec{v}_0 + \varepsilon \vec{v}') \cdot \nabla \vec{v}' + \frac{1}{\varrho_0}\left(1 - \varepsilon \frac{\varrho'}{\varrho_0}\right)\left(\nabla p' + \varrho'\, \vec{v}_0 \cdot \nabla \vec{v}_0\right) &= 0 \,, \\
\frac{\partial p'}{\partial t} + \vec{v}' \cdot \nabla p_0 + (\vec{v}_0 + \varepsilon \vec{v}') \cdot \nabla p' + \kappa \left[\nabla \cdot \vec{v}_0\, p' + \nabla \cdot \vec{v}' (p_0 + \varepsilon p')\right] &= 0 \,.
\end{aligned} \tag{2.5b}$$

For a given mean-flow field $(\varrho_0, \vec{v}_0, p_0)$ eqs. (2.5) are solved for $(\varrho', \vec{v}', p')$, which contain the acoustic field.

2.2 Acoustic Perturbation Equations

2.2.1 Introduction

Acoustic Perturbation Equations (APE) are also available in PIANO. The APE serve as alternative governing acoustic equations and can be deemed to be a modification of the genuine linearized EULER equations (LEE). The system solved for the pressure and velocity perturbations (p', $\vec{v}'$) is

$$\begin{aligned}
\frac{\partial p'}{\partial t} + c_0^2 \nabla \cdot \left(\varrho_0\, \vec{v}' + \vec{v}_0 \frac{p'}{c_0^2}\right) &= c_0^2\, q_\mathrm{c} \,, \\
\frac{\partial \vec{v}'}{\partial t} + \nabla \left(\vec{v}_0 \cdot \vec{v}'\right) + \nabla \left(\frac{p'}{\varrho_0}\right) &= \vec{q}_\mathrm{m} \,.
\end{aligned} \tag{2.6}$$

Here the ϱ_0, p_0, and $\vec{v}_0$ denote the density, pressure and velocity of the time averaged flow, respectively. Furthermore, $c_0 = \sqrt{\kappa\, p_0/\varrho_0}$ is the local speed of sound. The right-hand side sources q_c and $\vec{q}_\mathrm{m}$ will be defined below. Main purpose of the APE system is to provide governing acoustic equations that can be forced by appropriate right-hand side sources (e. g. vortex or combustion sources) without triggering growing hydrodynamic instabilities. In the framework of hybrid methods, the unsteady sound sources are provided by an unsteady CFD method, e. g. through a Large Eddy Simulation (LES). A hybrid approach allows especially for low MACH number flows to separate the small length-scale effects of the flow from the large length-scales that are present in the acoustic field. Another approach to set up sources is based on the stochastic modelling of unsteady sound sources from a steady REYNOLDS averaged NAVIER-STOKES (RANS) simulation. Due to the reduced effort of a RANS simulation compared to that of an LES, an efficient stochastic method yields a broadband prediction method, which is applicable for aeroacoustic design purposes. The stability of the APE system is accomplished by removing

the vorticity convection mode from the governing equations. However, convecting vorticity can still be present in the solution of the perturbation velocity field, but is entirely prescribed by the right-hand side source term. This allows to simulate the highly relevant airframe noise source mechanism due to the interaction of vorticity with solid surfaces by modelling only the vorticity related contributions of the right-hand side source terms.

Note that although the APE can be proven to be stable for arbitrary mean-flows, the homogeneous system is not intended to provide a solution to the long-standing — if ever solvable — problem of formulating acoustic equations that resolve acoustic convection and refraction effects in shear flows with arbitrary strong gradients but simultaneously suppress any hydrodynamic instabilities. Rather the system with right-hand side sources serves as an equation system based extended acoustic analogy, whose right-hand side is non-zero just in the sound generating very near field. However, the homogeneous equation system can be shown to be equivalent to the wave-equation of irrotational flow, which for a wide class of technical problems is a very good model to describe wave propagation through flows with mean vorticity present. More details can be found in [ES03, ES04].

2.2.2 Homogeneous APE

In the subsequent section the properties of the homogeneous system with all right-hand side sources removed will be discussed. The system considered reads

$$\frac{\partial p'}{\partial t} + c_0^2 \nabla \cdot \left(\varrho_0 \, \vec{v}' + \vec{v}_0 \frac{p'}{c_0^2} \right) = 0 \,, \tag{2.7}$$

$$\frac{\partial \vec{v}'}{\partial t} + \nabla \left(\vec{v}_0 \cdot \vec{v}' \right) + \nabla \left(\frac{p'}{\varrho_0} \right) = \vec{0} \,. \tag{2.8}$$

The equation system (2.7) and (2.8) is equivalent to a wave-equation for an acoustic potential φ. To identify the related wave-operator, the perturbation velocity $\vec{v}'$ has to be split into an irrotational $\nabla\varphi$ plus a remaining part $\vec{v}_\mathrm{r}$ that contains all the vorticity

$$\vec{v}' = \nabla\varphi + \vec{v}_\mathrm{r} \,. \tag{2.9}$$

Since $\vec{v}_\mathrm{r}$ is not defined to be irrotational (solenoidal), the decomposition becomes uniquely defined after imposing the additional condition that the unsteady pressure is related to the unsteady potential φ by

$$p' = -\varrho_0 \frac{\mathrm{D}_0 \varphi}{\mathrm{D}\, t} \,, \tag{2.10}$$

with the substantial time derivative $\mathrm{D}_0/\mathrm{D}\, t = \partial/\partial t + \vec{v}_0 \cdot \nabla$. Introducing (2.9) and (2.10) into (2.7) yields

$$\mathcal{L}\,\varphi \;:=\; \nabla \cdot \left(\varrho_0 \nabla\varphi - \frac{\varrho_0}{c_0^2} \frac{\mathrm{D}_0 \varphi}{\mathrm{D}\, t} \vec{v}_0 \right) - \frac{\partial}{\partial t} \left(\frac{\varrho_0}{c_0^2} \frac{\mathrm{D}_0 \varphi}{\mathrm{D}\, t} \right) = -\nabla \cdot \left(\varrho_0 \, \vec{v}_\mathrm{r} \right) . \tag{2.11}$$

Using the mean flow relation $\nabla \cdot (\varrho_0 \, \vec{v}_0) = 0$ to simplify the wave equation (2.11) and inserting the eqs. (2.9) and (2.10) into eqs. (2.8), the APE system can be rewritten as the equivalent system

$$\mathcal{L}'\varphi \;:=\; \left[\frac{\mathrm{D}_0}{\mathrm{D}\, t} \left(\frac{1}{c_0^2} \frac{\mathrm{D}_0}{\mathrm{D}\, t} \right) - \frac{1}{\varrho_0} \nabla \cdot \left(\varrho_0 \nabla \right) \right] \varphi = \frac{1}{\varrho_0} \nabla \cdot \left(\varrho_0 \, \vec{v}_\mathrm{r} \right) \,, \tag{2.12}$$

$$\frac{\partial \vec{v}_\mathrm{r}}{\partial t} + \nabla \left(\vec{v}_0 \cdot \vec{v}_\mathrm{r} \right) = \vec{0} \,. \tag{2.13}$$

Hence, the eqs. (2.12) and (2.13) could be understood as an equivalent system that follows by changing the independent variables of eqs. (2.7) and (2.8) from $(\vec{v}', p')^{\mathrm{T}}$ to $(\varphi, \vec{v}_{\mathrm{r}})^{\mathrm{T}}$. Each component of the (2.12) as well as (2.13) describe the behaviour of one eigenmode of the APE system. The convected wave operator $\mathcal{L}'$ for the variable φ of (2.12) governs the acoustic mode. Equation (2.13) describes the behavior of the vortical perturbations in the APE system. Taking the curl of this equation yields the vorticity equation of the APE system

$$\frac{\partial \vec{\omega}'}{\partial t} = 0 \,.$$

If the vorticity $\vec{\omega}' := \nabla \times \vec{v}' \equiv \nabla \times \vec{v}_{\mathrm{r}}$ is initially zero, it will remain so subsequently and this also holds for the related velocity component $\vec{v}_{\mathrm{r}}$ such that the right-hand side term in (2.12) vanishes. In what follows is that the homogeneous APE system is fully equivalent to the homogeneous wave-equation $\mathcal{L}'\varphi = 0$.

A variable transform from $(\vec{v}', p')^{\mathrm{T}}$ to $(\varphi, \vec{v}_{\mathrm{r}})^{\mathrm{T}}$ could also be accomplished for the linearized Euler equations. If we suppress for simplicity the entropy mode of the LEE by demanding the perturbation pressure and density to describe homentropic fields, the coupled acoustic/vortical system of equations corresponds to that proposed by Goldstein [Gol78] and recently used by Golubev & Atassi [GA98] and by Cooper & Peake [CP01] to predict the propagation of acoustic disturbances in swirling flows. The inhomogeneous wave operator that governs the acoustic mode agrees with (2.12), but the equation (2.13) for $\vec{v}_{\mathrm{r}}$ changes to

$$\frac{\partial \vec{v}_{\mathrm{r}}}{\partial t} + \vec{v}_0 \cdot \nabla \vec{v}_{\mathrm{r}} + \vec{v}_{\mathrm{r}} \cdot \nabla \vec{v}_0 = -\vec{\omega}_0 \times \nabla \varphi \,. \tag{2.14}$$

Hence, the APE system differs from the linearized Euler equations in that it does not possess the convection property for the vorticity perturbations, whereas the vorticity equation that follows from (2.14) by taking the curl of it also describes vorticity convection. The vorticity equation of the LEE can become subject to growing hydrodynamic instabilities. Since the APE system removes the vorticity equation while maintaining the same acoustic wave equation, one can guess that the APE system excludes all hydrodynamic instabilities in general. This is an important feature if the equations are forced by additional right-hand side sources, since in unstable mean-flows the solution will otherwise diverge. It will be discussed below that the wave operator $\mathcal{L}'$ is indeed stable for arbitrary mean flow fields and thus this feature also holds for the equivalent APE system.

The wave operator $\mathcal{L}'$ on the left-hand side of equation (2.12) on the previous page is that of Pierce's approximate wave equation [Pie90]. As discussed by Howe [How98], the extended wave-equation that governs wave propagation through irrotational flow is

$$\left[\frac{\mathrm{D}}{\mathrm{D}\,t}\left(\frac{1}{c^2}\frac{\mathrm{D}}{\mathrm{D}\,t}\right) - \frac{1}{\varrho}\nabla \cdot (\varrho\,\nabla)\right] \dot{\varphi} = 0 \,. \tag{2.15}$$

Here $\mathrm{D}/\mathrm{D}\,t = \partial/\partial t + \vec{V} \cdot \nabla$ is the substantial time derivative, $\vec{V} = \nabla\varphi_0(\vec{x})$ is the irrotational mean flow field, and $\dot{\varphi} = \partial\varphi'/\partial t$ is the time derivative of an irrotational disturbance, such that the unsteady and irrotational velocity field is described through the potential

$$\varphi(\vec{x}, t) = \varphi_0(\vec{x}) + \varphi'(\vec{x}, t) \,.$$

Since $\dot{\varphi} = \dot{\varphi}'$, (2.15) agrees with the time differentiated approximate wave equation introduced by Pierce [Pie90]. In other words, the homogeneous APE system without right-hand side sources is a first order partial differential system to integrate the wave equation of irrotational flow.

2.2.3 Extended Acoustic Analogy

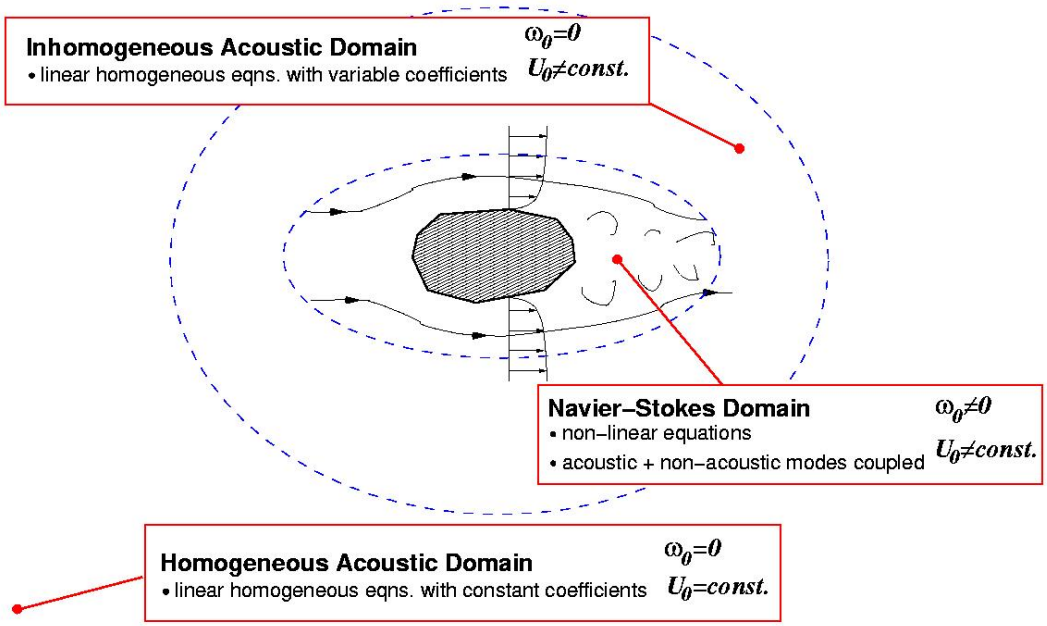


Figure 2.1: Domains of CAA

Figure 2.1 gives a principle classification of acoustic domains that occur for a body in external turbulent flow. In the near-field the flow is dominated by highly non-linear turbulent flow features that are accurately described by the full set of NAVIER-STOKES equations (referred to as the *Navier-Stokes domain*). In the NAVIER-STOKES domain acoustic and non-acoustic modes usually are coupled and cannot be separated into independent equations, e. g. a special wave-equation for the acoustic mode. This feature can be seen, e. g., in the LEE given in the form of (2.12) on page 11 or (2.14) on the facing page. On the right hand side of these equations the expressions $\varrho_0^{-1}\,\nabla\cdot(\varrho_0\,\vec{v}_\mathrm{r})$ and $-\vec{\omega}_0\times\nabla\varphi$ occur as terms that couple both equations. It is evident that the second term will vanish in irrotational mean-flow with $\vec{\omega}_0 = 0$. Furthermore, in irrotational flow and in case of initially vanishing velocity fluctuations $\vec{v}_\mathrm{r} = 0$ applies, such that the first term also disappears. Hence, in irrotational flow the acoustic and non-acoustic modes become decoupled.

In the far-field the time averaged mean-flow becomes irrotational and constant. Acoustic fluctuations that occur in this region are usually so small that their propagation is exactly described by the homogeneous wave equation for constant flow (referred to as *homogeneous acoustic domain*),

$$\left[\frac{1}{c_\infty^2}\frac{\mathrm{D}_\infty^2}{\mathrm{D}\,t^2} - \Delta\right]\varphi',\ \varrho',\ p' = 0\,. \tag{2.16}$$

Here the substantial time derivative based on the constant free stream velocity $\vec{v}_\infty$ is introduced, i. e. $\mathrm{D}_\infty/\mathrm{D}\,t = \partial/\partial t - \vec{v}_\infty\cdot\nabla$. The different variables in (2.16) indicate the wave-operator to be valid in conjunction with, respectively, an acoustic potential φ', a density $\varrho' = \varrho - \varrho_\infty$, and a pressure $p' = p - p_\infty$ as acoustic variables. LIGHTHILL's acoustic analogy [Lig52] follows from

the governing flow equations by recasting them into a simple wave equation on the left-hand side plus additional sources on the right-hand side. Considering the case of a constant mean-flow $\vec{v}_\infty$ via GALILEAN transformation of the acoustic analogy from a quiescent to a constantly moving medium acoustic analogy, it reads

$$\left[\frac{1}{c_\infty^2}\frac{\mathrm{D}_\infty^2}{\mathrm{D}\,t^2} - \Delta\right] p' = \nabla\nabla : \left(\varrho\,\vec{v}'\vec{v}' + (p' - c_\infty^2\varrho')\vec{\vec{I}} - \vec{\vec{\tau}}\right) \ ,$$

where $\vec{v}' = \vec{v} - \vec{v}_\infty$. Since the homogeneous left-hand side wave equation describes wave propagation exactly in the homogeneous domain (non-linearities neglected due to the smallness of acoustic fluctuations), the right-hand side sources of LIGHTHILL's analogy must vanish in the linear acoustic domain. Closer to the body the mean-flow, although still irrotational, becomes non-uniform. Since the physical correct wave equation in this domain is (2.15) that starts to deviate from the constant-flow wave equation (2.16), the sources of LIGHTHILL's acoustic analogy will be non-zero in this domain (referred to as *inhomogeneous acoustic domain*). Note, however, that the homogeneous wave equation (2.15) is still sufficient to describe the acoustics properly. Sound propagation phenomena in the inhomogeneous acoustic domain appear as sources of the LIGHTHILL analogy although they are obviously pure kinematic effects. LIGHTHILL's analogy cannot describe these effects, which have to be modelled through equivalent sources. Therefore, sources in the inhomogeneous acoustic domain cannot be regarded as 'true' acoustic sources.

LIGHTHILL's analogy applies the underlying wave equation of the homogeneous acoustic domain with virtual and acoustic sources on the right-hand side in all three acoustic domains. A straight forward extension of this concept would be to recast the governing flow equations such that the left-hand side realizes the wave equation of the inhomogeneous domain, which is (2.15), plus the remaining sources lumped together on the right-hand side. As such, wave propagation is computed in all three domains by the convective wave equation that governs the complete non-uniform and irrotational flow field (homogeneous and inhomogeneous domain). Such an extension would have the advantage that the region where the sources are non-zero reduces to the NAVIER-STOKES domain where vorticity is present since the homogenous wave-equation is valid in the inhomogeneous as well as the homogenous acoustic domain.

Attempting to extend LIGHTHILL's acoustic analogy [Lig52] to an acoustic analogy based on the wave-operator of irrotational flow, MÖHRING [Möh99, Möh79, How99] derived the equation

$$\mathcal{L}'B = \frac{\mathrm{D}}{\mathrm{D}\,t}\left(\frac{1}{c^2}\frac{\mathrm{D}\,B}{\mathrm{D}\,t}\right) - \frac{1}{\varrho}\nabla\cdot(\varrho\,\nabla B) = -\frac{q_{\mathrm{tot}}}{\varrho} \ , \tag{2.17}$$

where the source q_{tot} is determined by the governing fluid dynamic equations to be (viscous sources neglected)

$$q_{\mathrm{tot}} = \mathrm{div}\varrho\,\vec{v}\times\vec{\omega} + \left(\frac{\partial}{\partial t}\varrho_{\mathrm{s}}\frac{\partial}{\partial t} + \mathrm{div}\varrho_{\mathrm{s}}\vec{v}\frac{\partial}{\partial t} + \mathrm{div}\varrho T\nabla\right) s \ .$$

MÖHRING's acoustic analogy (2.17) uses the total enthalpy B as acoustic variable. In irrotational flow the total enthalpy is related to the velocity potential through BERNOULLI's equation, i. e. $\partial\varphi/\partial t + B = 0$. Furthermore B is constant in steady irrotational flow, and for external flow problems far from the acoustic source represents acoustic waves, since the energy equation of the EULER equation is

$$\frac{\mathrm{D}\,B}{\mathrm{D}\,t} = \frac{1}{\varrho}\frac{\partial p}{\partial t} \ .$$

2.2.4 APE Based Acoustic Analogy

Since the homogeneous APE system realizes an exact solution of the convective wave equation, recasting the governing flow equations such that the left hand side equals (2.6) on page 10, an extended analogy can be found, which is based on a set of dependent variables, namely p' and $\vec{v}'$, rather than on one scalar variable. Similar to the previous discussion, it can be concluded, that vortex sound sources on the right-hand side defined in such a way will vanish in and beyond the inhomogeneous acoustic domain, respectively. Note that an acoustic analogy concept, defined in the aforementioned way, must not necessarily be based on one scalar wave equation, nor is it necessary that it can be solved via integral methods.

Starting from the NAVIER-STOKES equations in non-linear disturbance formulation, the source terms of an APE based analogy become [ES03,ES04] (neglecting non-linear perturbation entropy terms)

$$\begin{aligned} q_\mathrm{c} &= -\nabla\cdot\left(\varrho'\,\vec{v}'\right)' + \frac{\varrho_0}{c_p}\frac{\mathrm{D}_0\,s'}{\mathrm{D}\,t}\ , \\ \vec{q}_\mathrm{m} &= -\left(\vec{\omega}\times\vec{v}\right)' + T'\,\nabla s_0 - s'\,\nabla T_0 - \left(\nabla\frac{(u')^2}{2}\right)' + \left(\frac{\nabla\cdot\vec{\vec{\tau}}}{\varrho}\right)'\ , \end{aligned} \tag{2.18}$$

where $(\ldots)' := (\ldots) - \overline{(\ldots)}$ denotes perturbations of terms obtained by subtracting from an actual term its time average, indicated by an overline. Major vortex source term is the fluctuating Lamb vector

$$\vec{q}_\mathrm{m} = -\left(\vec{\omega}\times\vec{v}\right)' = -\vec{\omega}_0\times\vec{v}' - \vec{\omega}'\times\vec{v}_0 - \left(\vec{\omega}'\times\vec{v}'\right)'\ . \tag{2.19}$$

A similar vortex source term appears in the acoustic analogies of POWELL [Pow64], HOWE [How75], and MÖHRING [Möh79]. Note that the linear dependence on vorticity in the vortex source (2.18) confirms the previously made assumption that the sources will vanish in the inhomogeneous acoustic domain, where the flow is irrotational.

2.2.5 Stability of the APE System

The wave operator $\mathcal{L}'$ of PIERCE's approximate wave equation, respectively the operator of MÖHRING's analogy has some unique features. As outlined by MÖHRING [Möh99, Möh79] the wave operator $\mathcal{L}'$ can be derived from a least action principle and can be shown to be formally self-adjoint, i. e., for a scalar product defined by $(f,\,g) = \int fg\,\mathrm{d}\,x^3\mathrm{d}\,t$ the relation

$$\left(\widetilde{B},\,\mathcal{L}'B\right) = \left(B,\,\mathcal{L}'\widetilde{B}\right)$$

holds for all arbitrary functions B and $\widetilde{B}$, which satisfy the constraint of vanishing sufficiently rapidly at infinity. This holds for the operator $\mathcal{L}'$ of (2.12) on page 11 in conjunction with arbitrary mean flow functions ϱ_0, c_0, and $\vec{v}_0$, which might be time-dependent, whereas ϱ_0 and $\vec{v}_0$ don't have to fulfill the continuity equation. From the self-adjointness a reciprocity relation for the GREEN's function associated with $\mathcal{L}'$ can be derived. Furthermore, from the self-adjointness one can conclude the existence of a variational principle from which the wave equation (2.11) on page 11 can be derived. One has

$$\delta L = 0 \quad \text{with} \quad L = \frac{1}{2}\left(B,\,\mathcal{L}'B\right) - \left(B,\,q_\mathrm{tot}\right)\ . \tag{2.20}$$

Variation of the LAGRANGian L yields

$$\delta L = \frac{1}{2}(\delta B,\, \mathcal{L}' B) + \frac{1}{2}(B,\, \mathcal{L}' \delta B) - (\delta B,\, q_{\text{tot}}) = (\delta B,\, \mathcal{L}' B - q_{\text{tot}}) \,.$$

With the above given definition of the scalar product $(f,\, g)$ and from the fundamental theorem of variational calculus it follows from the right-hand scalar product that the LAGRANGian L of (2.20) yields MÖHRING's acoustic analogy. As shown by MÖHRING, the LAGRANGian can be simplified further by integration by parts to obtain a form to be the simplest possible extension of the variational principle of the wave equation to non-uniform flow problems. From the action a conservation law for the acoustic energy can be deduced due to NOETHER's theorem.

MÖHRING concludes from the energy theorem that for initial value problems with vanishing right-hand side q_{tot} and for a vanishing solution at large distances from the source region the total energy in the sound field remains constant. Since the total energy is a sum of positive contributions, none of these can grow exponentially in time, i. e., instabilities cannot occur.

Since the wave operator $\mathcal{L}'$ can be proven to be stable, it follows that also the APE system is stable. This is a remarkable result since it is neither restricted to a particular class of mean flows, e. g. shear flows, nor limited to constant mean flow densities. Note that this stability property is related to convective or absolute hydrodynamic, i. e. physical, instabilities. It does not include numerical stability issues due to the chosen temporal or spatial discretization schemes. For example, CFL constraints still hold for the APE system.

2.2.6 Numerical APE Specifics

Note that the equation system (2.7), (2.8) on page 11 is fully decoupled from the linearized continuity equation

$$\frac{\partial \varrho'}{\partial t} + \vec{v} \cdot \nabla \varrho' + \vec{v}' \cdot \nabla \varrho + \varrho \nabla \cdot \vec{v}' + \varrho' \nabla \cdot \vec{v} = 0 \,,$$

which describes the time evolution of fluctuating density. However, in the current PIANO implementation the linearized continuity equation is integrated along with eqs. (2.7), (2.8) and as such the performance gain from omitting the equation is currently not completely exploited. In general, the continuity equation, although decoupled, yields a non-zero solution for the fluctuating density due to the non-zero term $\varrho \nabla \cdot \vec{v}'$.

The usual boundary conditions used for the LEE can also be applied for the APE. Note, however, that due to the suppressed vorticity mode in the homogeneous equations no vorticity will be present at a downstream boundary. Hence, at all boundaries a radiation boundary condition has to be applied to give a well posed problem. The application of an outflow boundary condition on the contrary would introduce an unnecessary additional degree of freedom, which eventually causes numerical instabilities. If right-hand side sources are present, they can induce velocity fluctuations that carry vorticity. Usually, however, the extension of the sources is restricted to the interior of the computational domain such that even in this case radiation conditions at all boundaries are appropriate.

Akin to the LEE, the truncation of the source terms of the APE will cause spurious sound sources. In order to prevent them, sources have to be smoothly introduced by spatially weighting the source term at up- and downstream boundaries of the source domain. As a rule of thumb, the streamwise width of the ramp function to avoid spurious sound sources should be of similar order as the characteristic length-scale of the vortical disturbances induced by the source term.

Furthermore, the source should also be turned on gradually at the beginning of the computation to avoid an initial acoustic bang due to the sudden appearance of vorticity in the interior source domain. Note that any fluctuating source on the right-hand side of the APE system in general will induce acoustic fluctuations. However, a physical source is in general a function of mean and fluctuating quantities that satisfy the NAVIER-STOKES equations. Hence, the flow quantities that underlie the sources have to satisfy the NAVIER-STOKES equations as an additional compatibility constraint that renders physical sources. This is similar to electromagnetic theory, where the electromagnetic field $\vec{E} = \nabla\varphi$ is prescribed by a scalar inhomogeneous wave-equation

$$\left[\frac{1}{c_0^2}\frac{\partial^2}{\partial t^2} - \Delta\right]\varphi = -\frac{\varrho}{\epsilon_0}\,, \tag{2.21}$$

where ϱ denotes the charge density. In principle the equation allows for longitudinal electromagnetic waves, which could be induced by a fluctuating right-hand side source. However, the charge density is conserved, i. e., it satisfies the continuity equation

$$\frac{\partial\varrho}{\partial t} + \vec{v}\cdot\nabla\varrho = 0\,.$$

Taking the conservation of density as additional compatibility equation into account, a fluctuating right-hand side always must be accompanied by an additional current such that a decoupled solution of (2.21) alone never occurs, yielding longitudinal electromagnetic waves non-physical.

Unphysical acoustic sources could occur in the APE if the sources are inappropriately modelled, i. e. by not resolving vorticity convection properly, or by using an inappropriately simplified source term by neglecting certain contributions.

2.3 Basic Equations in a Curvilinear Coordinate System

To be more flexible in treating curved boundaries the basic equations, e. g. (2.5) on page 10, are transformed into a curvilinear coordinate system (ξ, η, ζ). The Cartesian operator ∇_x can be written as a dyadic

$$\nabla_x = J\nabla_\xi\,, \tag{2.22}$$

where J denotes the JACOBian matrix. In expanded form, (2.22) writes

$$\begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\eta}{\partial x} & \frac{\partial\zeta}{\partial x} \\ \frac{\partial\xi}{\partial y} & \frac{\partial\eta}{\partial y} & \frac{\partial\zeta}{\partial y} \\ \frac{\partial\xi}{\partial z} & \frac{\partial\eta}{\partial z} & \frac{\partial\zeta}{\partial z} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial\xi} \\ \frac{\partial}{\partial\eta} \\ \frac{\partial}{\partial\zeta} \end{pmatrix}. \tag{2.23}$$

The complete set of the governing equations finally reads as follows:

$$\begin{aligned}
&\frac{\partial\varrho'}{\partial t} + \vec{v}'\cdot J\nabla_\xi\varrho_0 + (\vec{v}_0+\varepsilon\vec{v}')\cdot J\nabla_\xi\varrho' + (J\nabla_\xi)\cdot\vec{v}_0\,\varrho' + (J\nabla_\xi)\cdot\vec{v}'(\varrho_0+\varepsilon\varrho') &&= 0\,,\\
&\frac{\partial\vec{v}'}{\partial t} + \vec{v}'\cdot J\nabla_\xi\vec{v}_0 + (\vec{v}_0+\varepsilon\vec{v}')\cdot J\nabla_\xi\vec{v}' + \frac{1}{\varrho_0}\left(1-\varepsilon\frac{\varrho'}{\varrho_0}\right)\left(J\nabla_\xi p' + \varrho'\,\vec{v}_0\cdot J\nabla_\xi\vec{v}_0\right) &&= 0\,,\\
&\frac{\partial p'}{\partial t} + \vec{v}'\cdot J\nabla_\xi p_0 + (\vec{v}_0+\varepsilon\vec{v}')\cdot J\nabla_\xi p' + \kappa\left[(J\nabla_\xi)\cdot\vec{v}_0\,p' + (J\nabla_\xi)\cdot\vec{v}'(p_0+\varepsilon p')\right] &&= 0\,.
\end{aligned} \tag{2.24}$$

The aeroacoustic code PIANO solves the weakly nonlinear EULER equations in the form given by eqs. (2.24) subject to given ε. A full expansion (with indices also) of all terms can be found in Appendix A to A.9 on pages 106–108; handling details are given in Section 6.10 on page 76.

2.4 Sound Propagation Through Unsteady Base Flow

In primitive variable notation the NAVIER-STOKES equations are

$$\begin{aligned} \frac{\mathrm{D}\,\varrho}{\mathrm{D}\,t} + \varrho\,\nabla\cdot\vec{v} &= 0\,, \\ \frac{\mathrm{D}\,\vec{v}}{\mathrm{D}\,t} + \frac{\nabla p}{\varrho} &= \vec{q}_\mathrm{m}\,, \\ \frac{\mathrm{D}\,p}{\mathrm{D}\,t} + \kappa\,p\,\nabla\cdot\vec{v} &= q_\mathrm{e}\,, \end{aligned} \tag{2.25}$$

where $\mathrm{D}/\mathrm{D}\,t = \partial/\partial t + \vec{v}\cdot\nabla$ is the substantial time derivative. The primitive variables ϱ, $\vec{v}$, p denote density, velocities, and pressure, respectively. The terms on the right-hand side describe viscous and heat conduction effects. For example, the term on the right-hand side of the momentum equation is $\vec{q}_\mathrm{m} = \nabla\cdot\vec{\bar{\tau}}/\varrho$, where $\vec{\bar{\tau}}$ denotes the viscous stresses with pressure excluded. Acoustic wave propagation is governed by the NAVIER-STOKES equations, since they represent eigenmodes of the conservation equations. To simulate wave propagation in a prescribed unsteady turbulent flow field, in a first step the governing equations are expanded by considering small disturbances imposed on the flow field, i. e.

$$\begin{aligned} \widetilde{\varrho} &= \varrho + \varrho'\,, \\ \widetilde{\vec{v}} &= \vec{v} + \vec{v}'\,, \\ \widetilde{p} &= p + p'\,. \end{aligned}$$

Next, these expanded variables are substituted for the primitive variables in eqs. (2.25). No fluctuations are considered in the right-hand side terms, i. e. $\vec{q}'_\mathrm{m} \approx 0$ and $\vec{q}'_\mathrm{e} \approx 0$. In other words, as usual for the prediction of acoustic wave propagation, the effect of fluctuations in the viscous and heat conduction terms on the resolved perturbation variables is neglected. For ambient air the attenuation of acoustic waves due to dissipative effects yields a linear decay of the sound pressure levels with about $10^{-2}\,\mathrm{dB/m}$ at 1 kHz, so that it seems reasonable to neglect all dissipative effects on wave propagation over distances given by the typical length scales of aeronautical problems. By expanding the equations, rearrangement, and introduction of the mean-flow properties, which are still described by eqs. (2.25), and finally neglecting nonlinear terms of perturbation variables, the linearized equations that govern the evolution of the disturbances over the (unsteady) base flow field read

$$\begin{aligned} \frac{\partial\varrho'}{\partial t} + \vec{v}\cdot\nabla\varrho' + \vec{v}'\cdot\nabla\varrho + \varrho\,\nabla\cdot\vec{v}' + \varrho'\,\nabla\cdot\vec{v} &= 0\,, \\ \frac{\partial\vec{v}'}{\partial t} + \vec{v}\cdot\nabla\vec{v}' + \vec{v}'\cdot\nabla\vec{v} + \frac{\nabla p'}{\varrho} - \frac{\nabla p\varrho'}{\varrho^2} &= \vec{0}\,, \\ \frac{\partial p'}{\partial t} + \vec{v}\cdot\nabla p' + \vec{v}'\cdot\nabla p + \kappa\,p\,\nabla\cdot\vec{v}' + \kappa\,p'\,\nabla\cdot\vec{v} &= 0\,. \end{aligned} \tag{2.26}$$

Note that these homogeneous equations formally take on the form of the linearized EULER equations (LEE), however, the base flow is unsteady and acoustic wave propagation therefore is

simulated about this unsteady base flow. In a final step, the unsteady base flow is decomposed further in a time-averaged mean part (denoted with subscript 0) and a fluctuating part (denoted by subscript t). That is,

$$\begin{aligned} \varrho(\vec{x},t) &= \varrho_0(\vec{x}) + \varrho_\mathrm{t}(\vec{x},t)\ , \\ \vec{v}(\vec{x},t) &= \vec{v}_0(\vec{x}) + \vec{v}_\mathrm{t}(\vec{x},t)\ , \\ p(\vec{x},t) &= p_0(\vec{x}) + p_\mathrm{t}(\vec{x},t)\ . \end{aligned} \tag{2.27}$$

Since a time averaged quantity f_0 is defined by

$$f_0 = \overline{f} := \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} f \,\mathrm{d}\, t\ ,$$

the fluctuating part $f' = f - f_0$ satisfies $\overline{f'} = 0$. By introducing the decomposition (2.27) into the eqs. (2.26), the perturbation equations can be rewritten

$$\begin{aligned} \frac{\partial \varrho'}{\partial t} + \vec{v}_0\cdot\nabla\varrho' + \vec{v}'\cdot\nabla\varrho_0 + \varrho_0\,\nabla\cdot\vec{v}' + \varrho'\,\nabla\cdot\vec{v}_0 &= h_1\ , \\ \frac{\partial \vec{v}'}{\partial t} + \vec{v}_0\cdot\nabla\vec{v}' + \vec{v}'\cdot\nabla\vec{v}_0 + \frac{\nabla p'}{\varrho_0} - \frac{\nabla p_0\varrho'}{\varrho_0^2} &= \vec{h}_2\ , \\ \frac{\partial p'}{\partial t} + \vec{v}_0\cdot\nabla p' + \vec{v}'\cdot\nabla p_0 + \kappa\, p_0\,\nabla\cdot\vec{v}' + \kappa\, p'\,\nabla\cdot\vec{v}_0 &= h_3\ , \end{aligned} \tag{2.28}$$

where now interaction terms of acoustic and turbulent quantities appear, which are moved to the right-hand side and read up to second order in the perturbations

$$\begin{aligned} h_1 &= -\vec{v}_\mathrm{t}\cdot\nabla\varrho' - \vec{v}'\cdot\nabla\varrho_\mathrm{t} - \varrho_\mathrm{t}\,\nabla\cdot\vec{v}' - \varrho'\,\nabla\cdot\vec{v}_\mathrm{t}\ , \\ \vec{h}_2 &= -\vec{v}_\mathrm{t}\cdot\nabla\vec{v}' - \vec{v}'\cdot\nabla\vec{v}_\mathrm{t} - 2\frac{\nabla p_0}{\varrho_0^3}\varrho'\varrho_\mathrm{t} + \frac{\nabla p_\mathrm{t}\varrho'}{\varrho_0^2} + \frac{\nabla p'\cdot\varrho_\mathrm{t}}{\varrho_0^2}\ , \\ h_3 &= -\vec{v}_\mathrm{t}\cdot\nabla p' - \vec{v}'\cdot\nabla p_\mathrm{t} - \kappa\, p_\mathrm{t}\,\nabla\cdot\vec{v}' - \kappa\, p'\,\nabla\cdot\vec{v}_\mathrm{t}\ . \end{aligned} \tag{2.29}$$

The left-hand side equation system (2.28) is completely identical to the linearized EULER equations. Especially, a time averaged viscous mean-flow occurs, which can be predicted by a steady RANS solution of the time averaged flow problem. The interaction terms on the right-hand side, defined by products of unsteady fluctuations (ϱ', $\vec{v}'$, p') with fluctuations of the unsteady base flow (ϱ_t , $\vec{v}_\mathrm{t}$, p_t), can be esteemed to describe acoustic refraction effects of the unsteady base flow. Moreover, they do not introduce additional sound sources (which must be avoided if just scattering effects due to the base flow have to be resolved but not sound generation e. g. in a jet). The latter feature can be seen, considering a trivial initial solution $(\varrho',\ \vec{v}',\ p')^\mathrm{T} = \vec{0}$ for the resolved perturbations, which would cause vanishing source terms (2.29), eventually leaving the further progressed solution unchanged. Real flow induced sound sources, however, would generate responses in the perturbation variables, irrespective of their initial values. Note that these extended LEE, akin to the genuine LEE, could also become subject to hydrodynamic instabilities. It has to be validated, whether the CAA haystacking simulations for the actual used set of jet parameters trigger hydrodynamic instabilities, which could affect the simulation quality.

The turbulence velocity field is induced by the irregular motion of vorticity in the flow field. It was assumed that the fluctuating base flow is a solution of the NAVIER-STOKES equations (2.25) on page 18. Or, considering the velocity to be induced by vorticity, the latter has to satisfy the vorticity equation that follows from the momentum equation of (2.25) by taking the curl of it, i. e.

$$\frac{\mathrm{D}}{\mathrm{D}\,t}\left(\frac{\vec{\omega}}{\varrho}\right) = \frac{\vec{\omega}}{\varrho}\cdot\nabla\vec{v} - \frac{1}{\varrho^2}\nabla\varrho\times\frac{\mathrm{D}\,\vec{v}}{\mathrm{D}\,t}\ .$$

Consequently, the pure vortex induced fluctuating velocities should be divergence free (solenoidal), which is esteemed of importance in order to avoid the occurrence of spurious sound sources. Usually, turbulence realizations do not prescribe density and pressure fluctuations. Hence, the unsteady refraction terms that can be resolved through synthetic turbulence follow from (2.29) on the preceding page by neglecting all base flow pressure and density fluctuations and by considering $\nabla\cdot\vec{v}_\mathrm{t} = 0$, i. e.

$$\begin{aligned} h_1 &= -\vec{v}_\mathrm{t}\cdot\nabla\varrho'\ , \\ \vec{h}_2 &= -\vec{v}_\mathrm{t}\cdot\nabla\vec{v}' - \vec{v}'\cdot\nabla\vec{v}_\mathrm{t}\ , \\ h_3 &= -\vec{v}_\mathrm{t}\cdot\nabla p'\ . \end{aligned}$$

Finally, the system

$$\begin{aligned} &\frac{\partial\varrho'}{\partial t} + v_i'\frac{\partial\bar{\varrho}}{\partial x_i} + \frac{\partial\varrho'}{\partial x_i}(\bar{v}_i + v_i^\mathrm{t} + \varepsilon v_i') + \frac{\partial\bar{v}_i}{\partial x_i}\varrho' + \frac{\partial v_i'}{\partial x_i}(\bar{\varrho} + \varepsilon\varrho') &&= 0\ , \\ &\frac{\partial v_j'}{\partial t} + v_i'\frac{\partial(\bar{v}_j + v_j^\mathrm{t})}{\partial x_i} + \frac{\partial v_j'}{\partial x_i}(\bar{v}_i + v_i^\mathrm{t} + \varepsilon v_i') + \frac{1}{\bar{\varrho}}\left(1 - \varepsilon\frac{\varrho'}{\bar{\varrho}}\right)\left(\frac{\partial p'}{\partial x_j} + \varrho'\,\bar{v}_i\frac{\partial\bar{v}_j}{\partial x_i}\right) &&= 0\ , \\ &\frac{\partial p'}{\partial t} + v_i'\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial p'}{\partial x_i}(\bar{v}_i + v_i^\mathrm{t} + \varepsilon v_i') + \kappa\left[\frac{\partial\bar{v}_i}{\partial x_i}\,p' + \frac{\partial v_i'}{\partial x_i}(\bar{p} + \varepsilon p')\right] &&= 0\ , \end{aligned} \tag{2.30a}$$

being equivalent to

$$\begin{aligned} &\frac{\partial\varrho'}{\partial t} + \vec{v}'\cdot\nabla\varrho_0 + (\vec{v}_0 + \vec{v}_\mathrm{t} + \varepsilon\vec{v}')\cdot\nabla\varrho' + \nabla\cdot\vec{v}_0\,\varrho' + \nabla\cdot\vec{v}'(\varrho_0 + \varepsilon\varrho') &&= 0\ , \\ &\frac{\partial\vec{v}'}{\partial t} + \vec{v}'\cdot\nabla(\vec{v}_0 + \vec{v}_\mathrm{t}) + (\vec{v}_0 + \vec{v}_\mathrm{t} + \varepsilon\vec{v}')\cdot\nabla\vec{v}' + \frac{1}{\varrho_0}\left(1 - \varepsilon\frac{\varrho'}{\varrho_0}\right)\left(\nabla p' + \varrho'\,\vec{v}_0\cdot\nabla\vec{v}_0\right) &&= 0\ , \\ &\frac{\partial p'}{\partial t} + \vec{v}'\cdot\nabla p_0 + (\vec{v}_0 + \vec{v}_\mathrm{t} + \varepsilon\vec{v}')\cdot\nabla p' + \kappa\left[\nabla\cdot\vec{v}_0\,p' + \nabla\cdot\vec{v}'(p_0 + \varepsilon p')\right] &&= 0\ . \end{aligned} \tag{2.30b}$$

is solved by PIANO; handling details are given in Section 6.7 on page 71.

2.5 Equations for Source Modelling

2.5.1 Unsteady Sound Source from RPM

The Random Particle-Mesh (RPM) Modelling Approach

Introduction Two-point space-time correlations $\mathcal{R}(\vec{x},\vec{r},\tau) = \overline{\psi(\vec{x},t)\,\psi(\vec{x}+\vec{r},t+\tau)}$ based on GAUSSian and exponential functions for the spatial and temporal decay are widely used in statistical noise theories. Here the variable $\psi(\vec{x},t)$ shall indicate a scalar turbulent quantity, e. g.

an axial turbulent velocity component in a round jet. The correlation function can be written as

$$\mathcal{R}(\vec{x},\vec{r},\tau) = \hat{R}\ \exp\left[-\frac{|\tau|}{\tau_\mathrm{s}} - \frac{\pi(\vec{r}-\vec{v}_\mathrm{c}\tau)^2}{4l_\mathrm{s}^2}\right]. \tag{2.31}$$

The parameters τ_s and l_s define respectively the correlation time- and length scales and $\hat{R}$ denotes the mean-square (MS) value of the correlated quantity for vanishing separation space $\vec{r}$ and time τ. TAYLOR's hypothesis is taken into account by the convection velocity $\vec{v}_\mathrm{c}$. For inhomogeneous turbulence τ_s, $\vec{v}_\mathrm{c}$, l_s, and $\hat{R}$ depend on position $\vec{x}$.

Statistical noise prediction methods usually are applied in the frequency domain. They have been widely used to predict broadband jet noise. Some recent development is related to the extension of statistical time domain prediction methods to airframe noise problems [AM06] by using the exact GREEN's function of a complex airframe geometry instead of the free space GREEN's function applied for jet noise predictions. TAM & AURIAULT [TA99] extended the statistical jet noise prediction methodology based on LIGHTHILL's acoustic analogy by applying the linearized EULER equations as an extended wave-operator to describe wave propagation through the jet mean-flow.

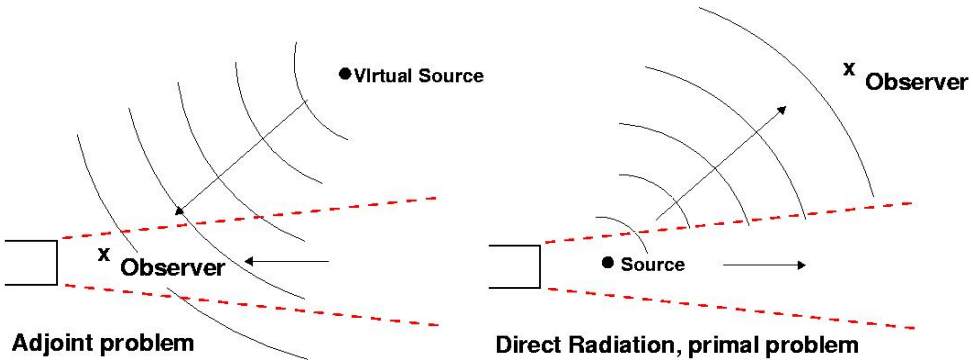


Figure 2.2: Adjoint and primal jet-noise problem

For example, Figure 2.2 shows a schematic of the approach proposed by TAM & AURIAULT [TA99] to model jet noise in the frequency domain. To determine the flow GREEN's functions that include jet refraction effects, TAM & AURIAULT utilize the adjoint linearized EULER equations (LEE) to compute its value in the jet source region with CAA techniques. From acoustic reciprocity the GREEN's value for the complete jet can be computed for one observer position (becoming the virtual source position in the adjoint problem) and one single frequency. This gives $n \times m$ computations for n observer positions and m frequency bands. A broadband CAA approach to solve the primal problem via LEE in the time-domain, on the contrary, allows to obtain the solution for all m frequency bands and n observers with one single computation. Note that a direct broadband approach in the time-domain yields equivalent results to a frequency domain method if the two-point space-time source statistics coincide.

In this framework the random particle-mesh method was introduced recently as a fast and efficient approach to set up fluctuating acoustic sound sources in the time-domain. The RPM method is capable of generating spatially and temporally fluctuating quantities whose statistics reproduce very accurately target two-point space-time correlations of the type described by (2.31), whereby local values of the parameters are realized. The fluctuating acoustic sources can

be used for a direct (primal) CAA approach. The applications of CAA techniques allows to consider complex mean-flow fields and geometries without any simplifications.

The RPM noise source model can be used for a prediction of broadband sound generation and radiation based on steady RANS, since it realizes very accurately target turbulence length-scales and kinetic energy distributions provided by RANS. The range of applications includes problems such as (fine scale turbulence) jet noise and airframe noise problems (e. g. trailing edge, slat, or flap side edge noise). Sources with the statistical features corresponding to that used by TAM & AURIAULT in statistical noise source models can be generated with the RPM method for a direct time-domain approach as well.

The relatively small overall computational time, which follows from the time necessary to conduct a steady RANS simulation and the time of a subsequent CAA-RPM simulation to solve the broadband sound radiation and propagation problem, allows the method to be applied for high REYNOLDS number aeroacoustic design purposes (e. g. slat noise at REYNOLDS numbers around $8 \cdot 10^6$).

Moreover, the efficient generation of turbulent velocity fields can be utilized for CAA simulations of spectral broadening effects due to the transmission of acoustic tones through unsteady turbulent shear layers. This effect is assumed to be responsible for the spectral broadening of jet engine turbine tones (haystacking effect).

Approach The fluctuating quantity $\psi(\vec{x},t)$ is generated by spatially filtering a white-noise field. The procedure can be expressed through a continuous convolution or spatial filtering integral, which reads for an n-dimensional problem

$$\psi(\vec{x},t) = \underbrace{\int \cdots \int_{A_S}}_{n\times} \hat{A}(\vec{x}')\, G^0\left(|\vec{x}-\vec{x}'|, l_s(\vec{x}')\right)\, \mathcal{U}(\vec{x}')\, \mathrm{d}\vec{x}' \,. \tag{2.32}$$

The integration area A_S corresponds to the source patch in which unsteady sources are realized. In (2.32) G^0 is a filter kernel, $\hat{A}$ is a local amplitude functions, $\mathcal{U}$ denotes the spatiotemporal white-noise field with properties defined below, and ψ is the realized fluctuating quantity . The filter kernel is normalized such that $\overline{\psi(\vec{x},t)\,\psi(\vec{x},t)} = 1$ for $\hat{A} = 1$. The argument of the filter kernel indicate that it is a function of the separation distance $|\vec{x}-\vec{x}'|$, and of the position-dependent kernel width l_s. The amplitude function in general depends on position as well.

The white-noise in (2.32) is defined to have special spatio-temporal properties. For the simulation of frozen turbulence, e. g., the spatiotemporal white-noise field is uniquely defined by the properties

$$\overline{\mathcal{U}(\vec{x},t)} = 0 \,, \tag{2.33}$$

$$\overline{\mathcal{U}(\vec{x},t)\,\mathcal{U}(\vec{x}+\vec{r},t)} = \delta(\vec{r}) \,, \tag{2.34}$$

$$\frac{\mathrm{D}_0}{\mathrm{D}\,t}\,\mathcal{U} = 0 \,, \tag{2.35}$$

where $\delta(\vec{r})$ denotes a multi-dimensional DIRAC δ-function, which e. g. reads in 2D $\delta(\vec{r}) = \delta(r_1)\,\delta(r_2)$. Equation (2.35) introduces the convection property into the fluctuation model through the passive convection of the white-noise field in a mean-flow $\vec{v}_0$. It is to be understood such that in a locally comoving frame of reference the spatiotemporal white-noise field

remains locally static. This condition can be satisfied even for an ideal white-noise realization (which is non-differentiable) although the substantial time derivative $\mathrm{D}_0/\mathrm{D}\,t = \partial/\partial t + \vec{v}_0\nabla$ involves spatial and temporal derivatives. As will be discussed subsequently in more detail, the spatial filtering in (2.32) causes the δ-correlated white-noise field to be smeared out over a surrounding area, such that the generated fluctuating quantity ψ shows a larger correlation length-scale. It is evident that the correlation length-scale must be a function of the filter kernel width. However, for a vanishing filter kernel width, i. e. if it changes into a δ-function, the filtered quantity ψ will preserve the properties of the white-noise field. In other words, for very small correlation length scales the modelled fluctuating quantity ψ will convect at the local mean-flow velocity. For larger turbulent structures the convection velocity will be an average over the surrounding mean-flow flow field. This feature is intended to model the convection of turbulent structures in a flow-field, which might agree for very small vortices with the local mean-flow, however can deviate from it if the turbulent vortices have a size comparable to mean-flow length scales. For example, it is well known that the convection velocity in turbulent boundary layers approaches approximately $v_\mathrm{c} \approx 0.6 \ldots 0.7 v_\infty$, where v_∞ denotes the flow velocity outside the boundary layer. Therefore, the convection velocity, which appears in the correlation function (2.31) on page 21 as a parameter that has to be appropriately modelled, is implicitly fixed in the RPM framework through the local turbulent length scale $l_\mathrm{s}(\vec{x})$ used in (2.32) on the facing page and the mean-flow field $\vec{v}_0(\vec{x})$ used in (2.34). The convection acoustic sources is deemed to be an essential feature in the modelling of jet-noise or airframe noise sources. For example, without convection the proper MACH number scaling laws cannot be achieved.

The definition (2.34) on the preceding page has to be satisfied for each point in the source domain where $\mathcal{U}$ is defined, whereas the convection equation (2.35) on the facing page determines the solution to be completely prescribed by the values on the inflow boundary of the source domain. It can be shown that the constraints (2.34) and (2.35) are satisfied simultaneously.

In [EE05, Ewe06, Ewe] it was shown that the spatial normalized correlation function $\mathcal{R}^0(\vec{r},\tau)$, which is defined by

$$\mathcal{R}^0(\vec{r},\tau) := \frac{\overline{\psi(\vec{x}+\vec{r},t+\tau)\,\psi(\vec{x},t)}}{\overline{\psi(\vec{x},t)\,\psi(\vec{x},t)}}\,, \tag{2.36}$$

and satisfies $\mathcal{R}^0(\vec{0},0) = 1$, is related for a n-dimensional problem to the filter-kernel function G^0 via

$$\mathcal{R}^0(\vec{r},0) = \underbrace{\int\cdots\int}_{n\times} G^0(\vec{r}-\vec{\xi})\,G^0(\vec{\xi})\,\mathrm{d}\vec{\xi}\,. \tag{2.37}$$

This relation is a consequence of (2.34). Using (2.37), it can be shown that a GAUSSian correlation function

$$\mathcal{R}^0(\vec{r},0) = \exp\left(-\frac{\pi}{4}\frac{|\vec{r}|^2}{l_\mathrm{s}^2}\right) \tag{2.38}$$

is generated through the GAUSSian filter kernel

$$G^0(\vec{r}) = \exp\left(-\frac{\pi}{2}\frac{|\vec{r}|^2}{l_\mathrm{s}^2}\right), \tag{2.39}$$

whose width is a factor $2^{-1/2}$ smaller in comparison with that of the correlation function. Although filter kernels of other correlation functions could be derived from (2.36) as well, a n-dimensional GAUSSian filter kernel is advantageous in such that he has the property to be

separable into a sequence of n one-dimensional filtering operations, which allows for a very efficient numerical discretization.

For a constant mean-flow $\vec{v}_0$, the convection equation (2.35) on page 22 is solved by $\mathcal{U}(\vec{x},t) = \mathcal{U}_0(\vec{x}-\vec{v}_0 t)$, where $\mathcal{U}_0(\vec{x})$ shall indicate the white-noise field at time-level $t=0$. Consequently, the relationship $\mathcal{U}(\vec{x}+\vec{r},t+\tau) = \mathcal{U}_0(\vec{x}+\vec{r}-\vec{v}_0\tau-\vec{v}_0 t) = \mathcal{U}(\vec{x}+\vec{r}-\vec{v}_0\tau,t)$ holds. Using this relation, the white-noise property (2.34) becomes for $\tau \neq 0$

$$\overline{\mathcal{U}(\vec{x},t)\,\mathcal{U}(\vec{x}+\vec{r},t+\tau)} = \overline{\mathcal{U}(\vec{x},t)\,\mathcal{U}(\vec{x}+\vec{r}-\vec{v}_0\tau,t)} = \delta(\vec{r}-\vec{v}_0\tau).$$

Using the last expression the extension of (2.37) to non-vanishing time separations $\tau \neq 0$ can be deduced. For the constant mean-flow case it follows by simply substituting $\vec{r}-\vec{v}_0\tau$ for $\vec{r}$ on the right-hand side of (2.37) on the preceding page, which becomes

$$\mathcal{R}^0(\vec{r},\tau) = \underbrace{\int\cdots\int}_{n\times} G^0(\vec{r}-\vec{v}_0\tau-\vec{\xi})\,G^0(\vec{\xi})\,\mathrm{d}\vec{\xi}\,. \tag{2.40}$$

Accordingly, the GAUSSian filter kernel (2.39) yields as an extension of (2.38) a normalized spatial correlation with non-vanishing time separation τ of the form

$$\mathcal{R}^0(\vec{r},\tau) = \exp\left(-\frac{\pi|\vec{r}-\vec{v}_0\tau|^2}{4l_\mathrm{s}^2}\right). \tag{2.41}$$

Note that (2.31) on page 21 takes on the form realized through (2.41) for $\vec{v}_\mathrm{c} = \vec{v}_0$ and $\tau_\mathrm{s} \to \infty$, i. e., for frozen turbulence. That is, in the constant mean-flow case the RPM convection velocity corresponds to $\vec{v}_0$, irrespective of the actual correlation length l_s. To introduce an additional exponential temporal correlation as in (2.31) the homogeneous convection equation (2.35) on page 22 has to be modified into a LANGEVIN equation. The numerical discretization of an exponential temporal correlation will be discussed in Section 2.5.1 on page 30.

Rigorously, the results presented so far are restricted to homogeneous filter kernels that realize constant correlations and length-scales throughout the source domain. Local kernels that realize inhomogeneous correlations and length-scales can be also deduced. However, the variation of these stationary quantities is usually small compared to the turbulent length-scale itself. Numerical test indicated the analytical findings for homogeneous filter kernels to be also valid with good accuracy for length scales $l_\mathrm{s}(\vec{x}')$ and kernel amplitudes $\hat{A}(\vec{x}')$ not locally varying too strong. The amplitude has to be chosen such that the local mean-square values $\hat{R}$ in (2.31) equals a target value. Appropriate values for $\hat{R}$ can be found utilizing the results from statistical broadband noise approaches.

Numerical Discretization of the RPM Method

In the two-dimensional discretization of the RPM method the continuous integral (2.32) on page 22 is approximated through the finite sum

$$\psi(\vec{x},t) = \sum_{i=1}^{i_\mathrm{max}} \sum_{j=1}^{j_\mathrm{max}} G_{ij}(\vec{x})\,r_{ij}(t)\,. \tag{2.42}$$

Equation (2.42) follows by splitting the source domain A_S in (2.32) into $M = i_\mathrm{max}\cdot j_\mathrm{max}$ non-overlapping control volumes ΔA_{ij} and by approximating the integral through the summation

over all control volumes. The indices i and j of a control volume ΔA_{ij} indicate its location with respect to the discretized source area. The amplitude $\hat{A}$ is absorbed in the filter kernel, i. e. $G_{ij}(\vec{x}) = G(\vec{x}, \vec{x}'_{ij}) = \hat{A}(\vec{x}'_{ij})\, G^0(\vec{x}, \vec{x}'_{ij})$. The spatial coordinate $\vec{x}'_{ij}$ denotes a point in the control volume related to control cell (i, j). It is advantageous to evaluate the filter kernel at the cell centre

$$\vec{x}^{\,\mathrm{s}}_{ij} = \frac{1}{\Delta A_{ij}} \int\limits_{\Delta A_{ij}} \vec{x}' \, \mathrm{d}\vec{x}' \tag{2.43}$$

of control volume ΔA_{ij}, i. e. $G_{ij}(\vec{x}) := \hat{A}_{ij}(\vec{x}^{\,\mathrm{s}}_{ij})\, G^0_{ij}(\vec{x}, \vec{x}^{\,\mathrm{s}}_{ij})$. Note that for simplicity in the following multi-dimensional integration is indicated through one single integral. The quantity r_{ij} in (2.42) is a random value defined through

$$r_{ij}(t) = \int\limits_{\Delta A_{ij}} \mathcal{U}(\vec{x}', t) \, \mathrm{d}\vec{x}' \; . \tag{2.44}$$

Let $\langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle := r_{ij}/\Delta A_{ij}$ be the average of the white noise field over ΔA_{ij}, then (2.42) is

$$\psi(\vec{x}, t) = \sum_{i=1}^{i_{\max}} \sum_{j=1}^{j_{\max}} G_{ij}(\vec{x}) \, \langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle \Delta A_{ij} \; . \tag{2.45}$$

Since $\langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle \to \mathcal{U}_{ij}$ holds in the limit of infinitely small subdomains $\Delta A_{ij} \to 0$, the (2.45) — and thus (2.42) — is a consistent approximation to (2.32). Two basic approximations are introduced in (2.45):

1. $\langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle$ is a filtered approximation to $\mathcal{U}_{ij}$,
2. the summation is a fourth order accurate approximation to the integral based on $\langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle$ as the underlying white-noise field,

please refer to Appendix B on page 109 for further details.

The local integration of the white-noise field over ΔA_{ij} in (2.44) can be deemed a low-pass filter applied to the field $\mathcal{U}$ at point (i, j), which causes a spectral cut-off of the spectrum corresponding to $\langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle$. As long as this cut-off wave number is larger than the highest wave-number to be resolved by the fluctuating streamfunction, it is obvious that such an approximation has only little effect on the resolved scales. All further properties of r_{ij} can be derived from the definitions (2.33)–(2.35) of the spatiotemporal white-noise field $\mathcal{U}$. Due to (2.33) and definition (2.44) the random value exhibits the property $\overline{r_{ij}} = 0$. Using definition (2.34), the correlation of the random values becomes

$$\overline{r_{ij}\, r_{kl}} = \int\limits_{\Delta A_{ij}} \int\limits_{\Delta A_{kl}} \delta(\vec{x} - \vec{x}') \, \mathrm{d}\vec{x} \, \mathrm{d}\vec{x}' = \begin{cases} 0 & \text{if} \quad i \neq k \vee j \neq l \\ \Delta A_{ij} & \text{if} \quad i = k \wedge j = l \end{cases} \tag{2.46}$$

That is, r_{ij} is a fluctuating quantity with zero mean and mean-square value $\overline{r_{ij}^2} = \Delta A_{ij}$.

According to (2.35) on page 22 the white-noise field remains locally static in a comoving frame of reference. That is, if $\Delta A'_{ij}$ describes a convecting control volume, whose boundary curve is drifting in the mean-flow and corresponds for $t = t_0$ with that of a fixed control volume ΔA_{ij}, the random value r_{ij} defined by the integral (2.44) applied to control volume $\Delta A'_{ij}$ is independent

of time in incompressible mean-flow. In a second order consistent approximation the cell centre of $\Delta A'_{ij}$ convects with its local mean-flow velocity

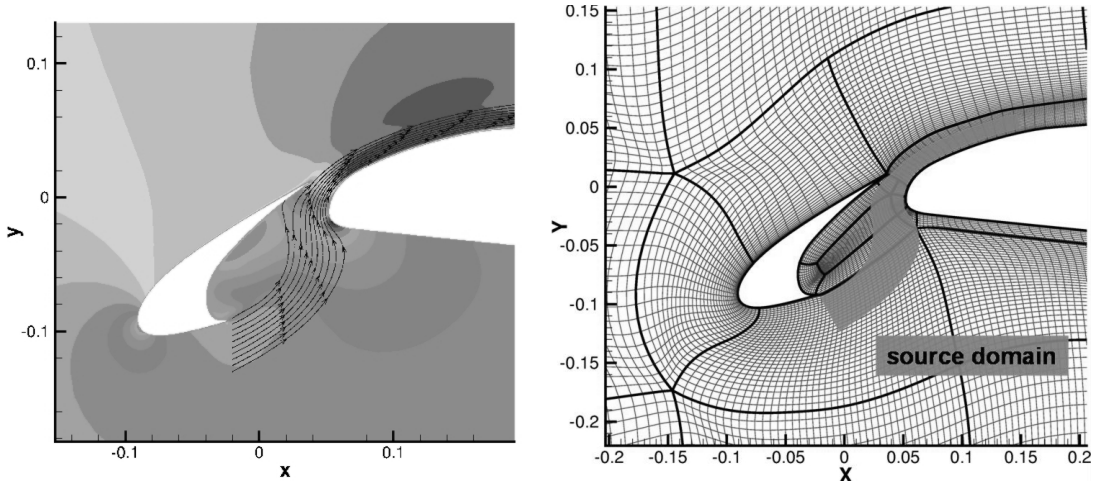
$$\dot{\vec{x}}^{\mathrm{s}}_{ij} = \vec{v}_0(\vec{x}^{\mathrm{s}}_{ij}) + \mathcal{O}(h^2) \;, \tag{2.47}$$

see Appendix C on page 110 for further discussion.

The discretization can be interpreted such that the source domain A_{S} is resolved by M discrete particles at moving locations $\vec{x}^{\mathrm{s}}_{ij}$. Each particle carries (in low MACH number flow) a frozen random value r_{ij} and drifts with the local mean-flow velocity through the source domain, Figure 2.3. In this picture the random values r_{ij} represents the white-noise field for the surrounding control volume $\Delta A'_{ij}$. To evaluate the fluctuating quantity ψ for discrete points $\vec{x}_{ij}$ (e. g. on a CAA mesh), it is computed with (2.42) by summing over all drifting particles, i. e.

$$\psi(\vec{x}_{ij}) = \sum_{k=1}^{k_{\max}} \sum_{l=1}^{l_{\max}} G_{kl}(\vec{x}_{ij})\, r_{kl} \;.$$

In the current RPM implementation the source patch is constructed by following the paths of the mean-flow streamlines, which start along an upstream seeding line, to a user-defined downstream position. See e. g. the left-hand side of Figure 2.3 that shows a bundle of streamlines to resolve the slat-cove shear layer of a high-lift airfoil, whose initial streamlines are equidistantly distributed along the seeding line. The complete source domain is depicted in Figure 2.3 (b).



(a) Streamlines in the slat-cove

(b) Resolved source domain and curvilinear multi-block CAA mesh

Figure 2.3: Resolution of the slat shear-layer in the two-dimensional test problem

Random particles are introduced along each streamline, whereby a constant drift time separation Δt between the particles is realized. Since the maximal time to reach the downstream border of a source patch depends on the considered streamline, the number of discrete particles also varies accordingly, see e. g. Figure 2.4 on the facing page. The sketch furthermore highlights the area ΔA_{ij} surrounding each discrete particle. The drift separation time and the number of streamlines determine the total number of random particles involved as well as the size of the subdomain ΔA_{ij}. The drift separation usually is chosen to be larger than the CAA time step.

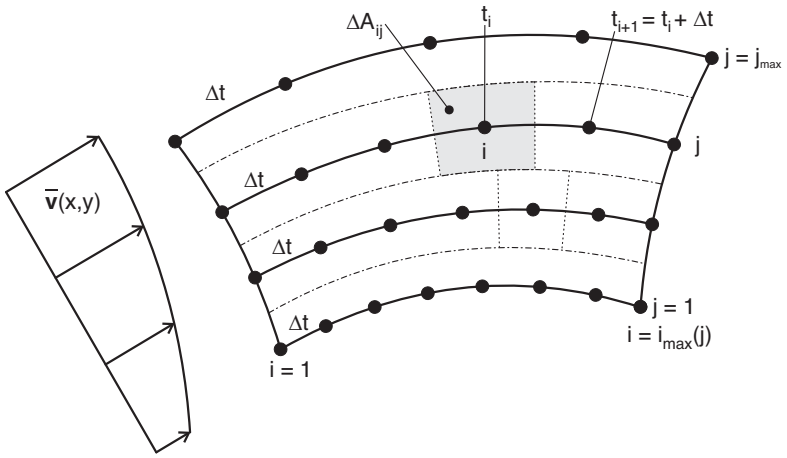
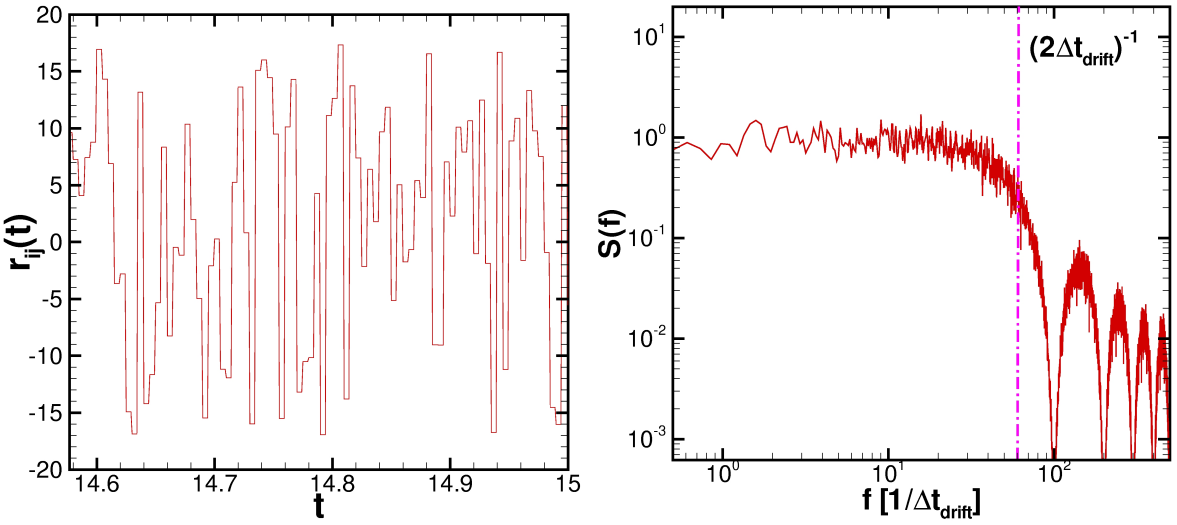


Figure 2.4: Sketch of streamlines and discrete particles in non-uniform mean-flow

An approximation to white-noise with a root-mean square (RMS) value normalized to one can be realized through a sequence of uniformly distributed random numbers in the range $\pm\sqrt{3}$, generated with a constant clock rate. The highest resolved frequency of this realization is linked to the seeding clock-rate Δt through the sampling theorem, i. e. $f_{\max} = 1/2\Delta t$. To achieve a mean-square value $\overline{r_{ij}^2} = \Delta A_{ij}$ through a sequence of uniformly distributed random numbers, each element ij has to take on a random value in the range $\pm\sqrt{3\Delta A_{ij}}$. As an example the left-hand side of Figure 2.5 shows the time history of the random values r_{ij} at a fixed location inside the source domain for a convection velocity normalized to one. The corresponding spectrum is presented in Figure 2.5 (b). It evidences a good realization of a unity spectrum up to approx. 50% of the NYQUIST frequency $1/2\Delta t$.



(a) Random data $r_{ij}(t)$ at a fixed probe location

(b) Corresponding spectrum

Figure 2.5: White noise representation at fixed position and corresponding spectrum; ratio of drift time and time step $\Delta t_{\text{drift}}/\Delta t_{\text{CAA}} = 10$, convection velocity $v_0 = 1.0$, 10^5 CAA time steps ($\Delta t_{\text{CAA}} = 10^{-3}$) sampled

The complete algorithm to compute the streamfunction becomes the following:

- For each CAA time increment convect the random particles downstream.
- If a particle crosses the downstream border delete it and update the first upstream position with a new random particle with a new random value uniformly distributed in the range $\pm\sqrt{3\Delta A_{ij}}$.
- Filter and simultaneously interpolate the random field onto the CAA grid.

The local value of the turbulence kinetic energy at the random particle location scales the local amplitude $\hat{A}$ of the filter kernel. The exact value of $\hat{A}$ based on the RANS mean-flow field is discussed in the next section. The filtered values are directly computed for the relevant CAA grid points.

The filter kernel is computed in a sequence of one-dimensional filter operations, see Figure 2.6. It takes typically 10% to 1% of the time the direct evaluation of the full filter kernel would need in 2D and 3D, respectively. First, the random field is filtered along the streamline for each discrete point on the streamline, using the length scale at each particle location for the kernel scaling. Next, the intermediate filtered values are distributed onto the CAA grid, Figure 2.6. Let us denote l_1 the length-scale at a given stochastic particle position **A** on the streamline, **P** the point on the streamline with smallest distance to the CAA grid point **B**, s the distance along the streamline to the base-point, l_2 the length-scale at the base-point **P**, and d the distance to the grid point **B**, Figure 2.6. Then the contribution of a random element in **A** to the grid point **B** due to the filter kernel reads

$$G^0_{AB} := G^0(\vec{x}_B, \vec{x}_A) = G^0(d)\, G^0(s) = \exp\left(-\frac{\pi}{2}\frac{s^2}{\Delta_1^2}\right) \exp\left(-\frac{\pi}{2}\frac{d^2}{\Delta_2^2}\right). \tag{2.48}$$

For uniform flow with constant length-scale this is identical with the GAUSSian kernel; for curved streamlines a small distortion of the kernel shape might occur.

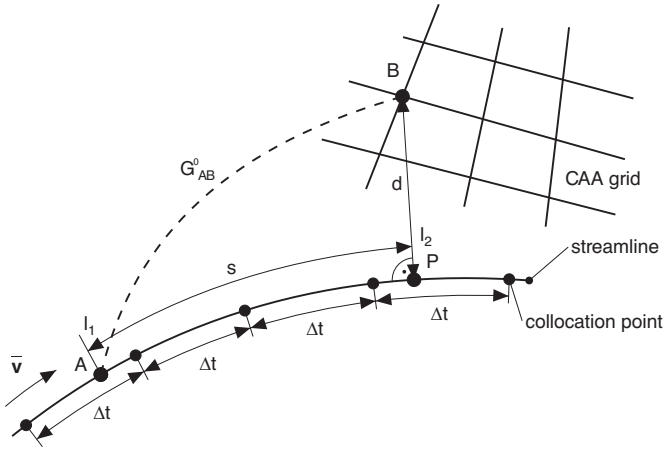


Figure 2.6: Interpolation onto CAA grid points

Sound Sources from Modelled Turbulent Velocities

One approach to realize acoustic sound sources is to model turbulent velocities as a basis to determine vortex sound sources. For example, the major vortex sound source of the acoustic

analogy presented in Section 2.2.4 on page 15 is based on the perturbed Lamb vector, whose major contribution is given by $q_i = -\varepsilon_{ijk}(\omega_0)_j v'_k - \varepsilon_{ijk}\omega'_j(v_0)_k$, where $(\omega_0)_i = \varepsilon_{ijk}\partial(v_0)_k/\partial x_j$ and $\omega'_i = \varepsilon_{ijk}\partial v'_k/\partial x_j$. Hence, to model this source, beside mean-flow velocities $\vec{v}_0$ also fluctuating velocities $\vec{v}'$ have to be measured.

To model turbulent related velocities with the RPM method the divergence-free conditions of vorticity related turbulence is taken into account by identifying the modelled fluctuating quantity with a streamfunction $\psi(\vec{y}, t)$. Strictly solenoidal 2D perturbation velocities are deduced from this fluctuating streamfunction at each time instant via

$$v_i = \varepsilon_{ij}\frac{\partial \psi}{\partial y_j} \,, \tag{2.49}$$

where ε_{ij} denotes the two-dimensional ε-tensor. In case of homogeneous isotropic turbulence the velocity correlations that follow from (2.49),

$$\mathcal{R}_{ij}(\vec{r}, \tau) = \overline{v_i(\vec{r}_1, t_1)\, v_j(\vec{r}_2, t_2)} \,, \tag{2.50}$$

match perfectly the complete velocity correlation tensor of isotropic turbulence for $\tau = 0$,

$$\mathcal{R}_{ij}(\vec{r}, 0) = \frac{2}{3}\,\overline{k}\left[\frac{f(r) - g(r)}{r^2} r_i r_j + g(r)\delta_{ij}\right] . \tag{2.51}$$

In the above expressions $\vec{r} = \vec{r}_1 - \vec{r}_2$ and $\tau = |t_1 - t_2|$ are the spatial and temporal separations between point 1 and 2. The separation distance is $r = |\vec{r}|$ and r_i is the ith component of vector $\vec{r}$. The turbulence kinetic energy is denoted by $\overline{k}$ and $f(r)$ and $g(r)$ denote the longitudinal and lateral correlation functions, respectively, which are connected for a two-dimensional problem through

$$g(r) = f(r) + r\frac{\mathrm{d}f(r)}{\mathrm{d}r} \,. \tag{2.52}$$

This feature to exactly realize homogeneous isotropic turbulence in the two-point correlations can be proven by considering the space-time correlations of the fluctuating streamfunction in this homogeneous case to be a pure function of the separation vector $\vec{r}$

$$\mathcal{R}_{\psi\psi}(\vec{r}, \tau) = \overline{\psi(\vec{r}_1, t_1)\, \psi(\vec{r}_2, t_2)} \,. \tag{2.53}$$

The velocity correlations (2.50) are connected via (2.49) with the correlations (2.53). By taking $\vec{r} = \vec{r}_1 - \vec{r}_2$ into account, the relationship is

$$\mathcal{R}_{ij} = -\varepsilon_{ik}\varepsilon_{jl}\frac{\partial^2 \mathcal{R}_{\psi\psi}}{\partial r_k \partial r_l} \,. \tag{2.54}$$

Since ψ is generated by a procedure that realizes correlations of the type described by (2.31) on page 21, the correlation $\mathcal{R}_{\psi\psi}$ in (2.54) can be expressed through the right-hand side of (2.31), where the parameters are constants for the homogeneous problem. Next, the explicit expressions found for $\mathcal{R}_{ij}$ in the case $\tau = 0$ can be verified to have the formal shape defined by (2.51), whereby the resulting longitudinal and lateral correlation functions furthermore satisfy (2.52). To match quantitatively (2.51) the amplitude $\hat{R}$ in (2.31) has to set to

$$\hat{R} = \frac{4 l_s^2 \overline{k}}{3\pi} \,. \tag{2.55}$$

Then, the resulting longitudinal correlation function that follows from (2.31) is a GAUSSian,

$$f(r) = \exp\left(-\frac{\pi}{4}\frac{r^2}{l_\mathrm{s}^2}\right), \tag{2.56}$$

with an integral length scale directly determined by the parameter l_s,

$$L = \int_0^\infty f(r)\,\mathrm{d}r = l_\mathrm{s}\ .$$

Hence, the parameters $\hat{R}$ and l_s in (2.31) on page 21 are directly linked via (2.55) and (2.56) to the turbulence kinetic energy and length scale as provided by RANS. One free empirical constant c_l has to be determined, which defines the relation between the RANS length scale and the integral length scale of turbulence. For a k-ϵ turbulence model the relation is

$$l_\mathrm{s} = c_\mathrm{l}\frac{\overline{k}^{3/2}}{\epsilon}\ . \tag{2.57}$$

For a k-ω model the corresponding relation becomes

$$l_\mathrm{s} = \frac{c_\mathrm{l}}{C_\mu}\frac{\overline{k}^{1/2}}{\omega}\ . \tag{2.58}$$

Following the discussion of BAILLY & JUVÉ [BJ99] the constant can be estimated to be $c_\mathrm{l} \approx 0.54$ for a modified VON KÁRMÁN spectrum. Hence, with $C_\mu = 0.09$, the constant can be estimated with $c_\mathrm{l}/C_\mu \approx 6.0$ for the k-ω model. A certain value of $\hat{R}$ in (2.31) on page 21 can be achieved, if $\hat{A}$ in (2.32) on page 22 is chosen appropriately. Explicit expressions for $\hat{A}$ can be found in [Ewe]. The approach (2.49) can be extended to 3D, details can be found in [Ewe] as well.

Time Domain Realization of Tam & Auriault's Scalar Jet-Noise Source

The statistical jet noise model of TAM & AURIAULT [TA99] is based on the modified linearized EULER equations rewritten in cylindrical coordinates as governing acoustic equations. The modifications include the neglect of jet spreading terms as well as mean-flow gradients. Through the mean-flow gradients the continuity equation is coupled with the momentum equations. In the TAM & AURIAULT model the continuity equation is not used since it decouples from the momentum equation if mean-flow gradients are not considered. Furthermore, an acoustic source term is introduced on the right-hand side of the momentum equation. In Cartesian coordinates and including the jet spreading terms (i. e. only with the mean-flow gradients omitted), the governing equations of TAM & AURIAULT read

$$\varrho_0\left[\frac{\partial \vec{v}'}{\partial t} + \vec{v}_0\nabla\vec{v}'\right] + \nabla p = -\nabla q_\mathrm{s}\ , \tag{2.59}$$

$$\frac{\partial p}{\partial t} + \vec{v}_0\nabla p + \gamma p_0 \nabla\cdot\vec{v}' = 0\ . \tag{2.60}$$

In a next step TAM & AURIAULT use the adjoint equations to the frequency domain FOURIER transform of (2.59), (2.60) to derive an expression for the free-space GREEN's functions of the governing equations (2.59), (2.60) in the frequency domain. The GREEN's functions include

mean-flow refraction effects and have been computed with CAA techniques. Finally, the numerically determined values of the GREEN's function are used to approximately solve the integral expression that defines the far-field acoustic spectrum in terms of the noise source space-time correlation

$$\left\langle \frac{\mathrm{D}_0\, q_\mathrm{s}(\vec{x}_1, t_1)}{\mathrm{D}\, t_1} \; \frac{\mathrm{D}_0\, q_\mathrm{s}(\vec{x}_2, t_2)}{\mathrm{D}\, t_2} \right\rangle . \tag{2.61}$$

In the above expression $^{\mathrm{D}_0}/_{\mathrm{D}\,t} := {}^{\partial}/_{\partial t} + \vec{v}_0 \cdot \nabla$ denotes a substantial time derivative and the brackets indicate an ensemble average. TAM & AURIAULT propose the correlation function (2.61) to be described by

$$\left\langle \frac{\mathrm{D}_0\, q_\mathrm{s}(\vec{x}_1, t_1)}{\mathrm{D}\, t_1} \; \frac{\mathrm{D}_0\, q_\mathrm{s}(\vec{x}_2, t_2)}{\mathrm{D}\, t_2} \right\rangle = \frac{\hat{q}_\mathrm{s}^2}{c^2 \tau_\mathrm{s}^2} \times \exp\left(-\frac{|\xi|}{v_0 \tau_\mathrm{s}} - \frac{\ln 2}{\hat{l}_\mathrm{s}^2} \left[(\xi - v_0 \tau)^2 + \eta^2 + \zeta^2 \right] \right) . \tag{2.62}$$

Note that the above correlation function considers a mean-flow in x-direction, i. e. only the velocity component v_0 is non-zero. This condition is approximately satisfied for a slightly spreading jet. Furthermore, the ratio $|\xi|/v_0$ of separation distance between point 1 and 2 and local convection velocity could be expressed through the time separation $|\tau|$ as well.

To model the noise source with the fast RPM method in the time-domain the initial equation system (2.59), (2.60) has to be reformulated. That is, the pressure is decomposed according to

$$p := p' - q_\mathrm{s} \; , \tag{2.63}$$

respectively $p' = p + q_\mathrm{s}$. Since q_s was introduced by TAM & AURIAULT as a turbulence related pressure fluctuation, the meaning of the decomposition (2.63) is that p' is a perturbation pressure with all turbulence related pressure fluctuations included, whereas p in (2.59), (2.60) is a fluctuating pressure, which excludes the turbulence related pressure fluctuations. Note that p' as well as p comprise all relevant acoustic pressure fluctuations. Next, by introducing the decomposition (2.63) into (2.59), (2.60) the modified equation system that follows reads

$$\varrho_0 \left[\frac{\partial \vec{v}'}{\partial t} + \vec{v}_0 \nabla \vec{v}' \right] + \nabla p' \;\; = \;\; 0 \; , \tag{2.64}$$

$$\frac{\partial p'}{\partial t} + \vec{v}_0 \nabla p' + \gamma p_0 \nabla \cdot \vec{v}' \;\; = \;\; \frac{\mathrm{D}_0\, q_\mathrm{s}}{\mathrm{D}\, t} \; . \tag{2.65}$$

In this reformulation of the governing perturbation equations the scalar source term $^{\mathrm{D}_0\, q_\mathrm{s}}/_{\mathrm{D}\, t}$ appears on the left-hand side of the pressure equation. The correlation of this scalar quantity can be modelled with the RPM method, which becomes evident by comparing the correlation (2.62) on this page with (2.31) on page 21 realized by the RPM method for the case of a constant mean-flow in x-direction with $\vec{v}_0 = (v_0, 0, 0)^\mathrm{T}$. It can be identified that the turbulent length-scales l_s in (2.31) and $\hat{l}_\mathrm{s}$ in (2.62) are related through

$$l_\mathrm{s} = \frac{1}{2} \sqrt{\frac{\pi}{\ln 2}} \, \hat{l}_\mathrm{s} \; . \tag{2.66}$$

TAM & AURIAULT propose to model the length scale as $\hat{l}_\mathrm{s} = \hat{c}_\mathrm{l} k^{3/2} / \epsilon$, with $\hat{c}_\mathrm{l} = 0.256$. This yields

$$l_\mathrm{s} = c_\mathrm{l} \frac{k^{3/2}}{\epsilon} \; , \tag{2.67}$$

with $c_\mathrm{l} = 0.273$. Note that the turbulent pressure, whose correlations are modelled here, is a function of the turbulent velocities squared. Interestingly, a value of $c_\mathrm{l} = 0.273$ is exactly half as large as the parameter found for the velocity correlations (2.57) on the facing page.

As an extension of the statistical source description, the restriction of having only a non-zero velocity component v_0 along the jet axis can be dropped in the framework of the RPM model in favor of a full use of the mean-flow from RANS. In the applications discussed so far the modelled fluctuating quantity ψ is used as a streamfuntion, from which solenoidal turbulent velocity components are derived subsequently. The turbulent velocities serve as a basis to compute velocity dependent sound sources. In contrast, the extension for a primal TAM & AURIAULT approach in the time-domain mainly means to generate the scalar source $\mathrm{D}_0\, q_\mathrm{s}/\mathrm{D}\, t$ by directly identifying it with the fluctuating quantity ψ provided by the RPM model. For this, the amplitude $\hat{A}$ in (2.32) on page 22 has to be chosen such that a value $\hat{R}$ in (2.31) on page 21 is realized, which corresponds to the prefactor in (2.62) on the previous page, i. e.,

$$\hat{R} = \frac{\hat{q}_\mathrm{s}^2}{c^2 \tau_\mathrm{s}^2} \,. \tag{2.68}$$

This relation uniquely fixes the amplitude $\hat{A}$. More details about scaling $\hat{A}$ can be found in [Ewe06, Ewe].

The generation of fluctuations with the RPM method was discussed so far for frozen turbulence, having GAUSSian spatial correlations and involving TAYLOR's hypothesis. The simulation of frozen turbulence is found to be sufficient for the modelling of airframe noise mechanisms. To simulate jet-noise, however, the realization of proper time correlations, using local time-scales, is deemed crucial, since it is the change in turbulence that is responsible for jet-noise generation. To include the exponential time decay of (2.31), (2.35) on page 22 has to be extended to a LANGEVIN equation. In the numerical realization this means that the random values carried by each particle are not constant but in principle change over time according to the discrete equation

$$r_i^{n+1} = \alpha r_i^n + \beta s_i^n \,. \tag{2.69}$$

Here r_i^{n+1} and r_i^n denote the random value of a particle at time-level $n+1$ and n, respectively. The quantity s_i^n is a new random value in the same range as r_i. This procedure realizes an exponential decay, cf. [BED03]. The constant α is related to the time-scale τ_s via

$$\alpha = \exp\left(-\frac{\Delta t}{\tau_\mathrm{s}}\right) \,, \tag{2.70}$$

where Δt denotes the time-increment between levels $n+1$ and n. To preserve the root-mean square value of r_i over time, β is related to α via $\beta = \sqrt{1-\alpha^2}$.

2.5.2 Modified Euler Equations with Source Terms

Although the implementation of the SNGR source terms is still under construction and currently *not* working, some details are given.

The SNGR[1] method is designed to calculate the turbulent sound generation by means of the solution of a modified form of the EULER equations with defined source terms on the right hand side of the equations of motion. One can choose between four possible source terms namely the so-called *shear source term* $\vec{v}_\mathrm{t} \cdot \nabla \vec{v}_0$, the so-called *time term* $\partial \vec{v}_\mathrm{t}/\partial t$, the so-called *perturbed Lamb vector* $(\mathrm{rot}\vec{v}_\mathrm{t}) \times \vec{v}_0 + (\mathrm{rot}\vec{v}_0) \times \vec{v}_\mathrm{t}$ and the *self source term* $\vec{v}_\mathrm{t} \cdot \nabla \vec{v}_\mathrm{t}$. The equations (2.71) are the equations of motion of the SNGR model in case the shear source term $v_{\mathrm{t}j} \partial \bar{v}_i/\partial x_j$ is used and the

[1] **S**tochastic **N**oise **G**eneration and **R**adiation

equations (2.72) in case the time term $\partial\vec{v}_\mathrm{t}/\partial t$, the perturbed Lamb vector $(\nabla\times\vec{v}_\mathrm{t})\times\vec{v}_0+(\nabla\times\vec{v}_0)\times\vec{v}_\mathrm{t}=\varepsilon_{jlk}\,\varepsilon_{lmn}\partial v_{\mathrm{t}n}/\partial x_m\,\bar{v}_k+\varepsilon_{jlk}\,\varepsilon_{lmn}\partial\bar{v}_n/\partial x_m\,v_{\mathrm{t}k}$ or the self source term $v_{\mathrm{t}j}\partial v_{\mathrm{t}i}/\partial x_j$ is used.

$$\frac{\partial v_j'}{\partial t}+\bar{v}_i\frac{\partial v_j'}{\partial x_i}+\frac{\varrho'}{\bar{\varrho}}\,\bar{v}_i\frac{\partial\bar{v}_j}{\partial x_i}+\frac{1}{\bar{\varrho}}\frac{\partial p'}{\partial x_j} = -v_{\mathrm{t}i}\frac{\partial\bar{v}_j}{\partial x_i} \tag{2.71}$$

$$\frac{\partial v_j'}{\partial t}+v_i'\frac{\partial\bar{v}_j}{\partial x_i}+\bar{v}_i\frac{\partial v_j'}{\partial x_i}+\frac{\varrho'}{\bar{\varrho}}\,\bar{v}_i\frac{\partial\bar{v}_j}{\partial x_i}+\frac{1}{\bar{\varrho}}\frac{\partial p'}{\partial x_j} = -v_{\mathrm{t}i}\frac{\partial v_{\mathrm{t}j}}{\partial x_i} \tag{2.72}$$

In [Bau03, Sections 4.4 and 3.2] there is a detailed derivation of these equations besides numerous comments. The equations can also be derived from the equation (2.5) on page 10 by taking a few steps. One will see that there are only linear terms ($\varepsilon=0$) in the equations of motion (2.71) and (2.72) except from the self source term. The source terms are calculated by means of the synthetic turbulence $\vec{v}_\mathrm{t}(\vec{x},t)$ and they are located only in certain parts of the calculation area, the so-called *patches*.

The continuity equation (2.73) and the energy equation (2.74) of the SNGR model correspond to the equations (2.5) on page 10 with ($\varepsilon=0$):

$$\frac{\partial\varrho'}{\partial t}+v_i'\frac{\partial\bar{\varrho}}{\partial x_i}+\bar{v}_i\frac{\partial\varrho'}{\partial x_i}+\varrho'\frac{\partial\bar{v}_i}{\partial x_i}+\bar{\varrho}\frac{\partial v_i'}{\partial x_i} = 0 \tag{2.73}$$

$$\frac{\partial p'}{\partial t}+v_i'\frac{\partial\bar{p}}{\partial x_i}+\bar{v}_i\frac{\partial p'}{\partial x_i}+\kappa\left(p'\frac{\partial\bar{v}_i}{\partial x_i}+\bar{p}\frac{\partial v_i'}{\partial x_i}\right) = 0 \tag{2.74}$$

2.5.3 Weighting Function

At present only a single rectangular patch can be integrated into the calculation area. The patch boundaries are defined by the parameters x_l, x_r, y_u and y_o, see Figure 2.7. The source terms

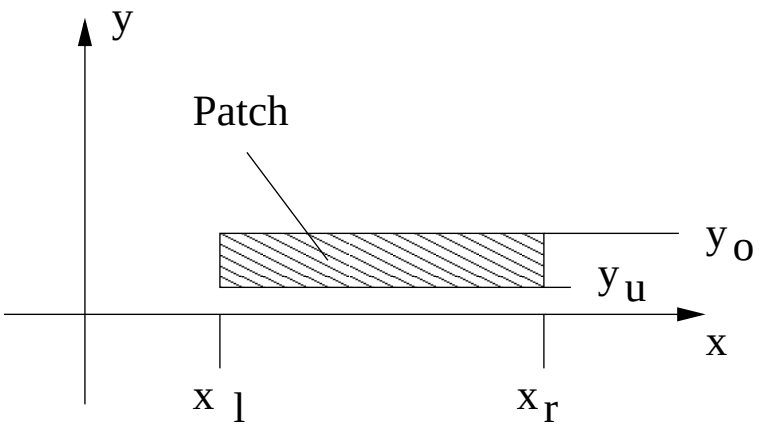


Figure 2.7: Description of the source term area (patch) with the parameters x_l, x_r, y_u and y_o

can be multiplied with a weighting function $W(x,y)$. In the two-dimensional case valid is

$$W(x,y)=\begin{cases} C\cdot W(x)W(y) & -x_l\le x\le x_r\wedge y_u\le y\le y_o \\ 0 & \text{else}\end{cases}. \tag{2.75}$$

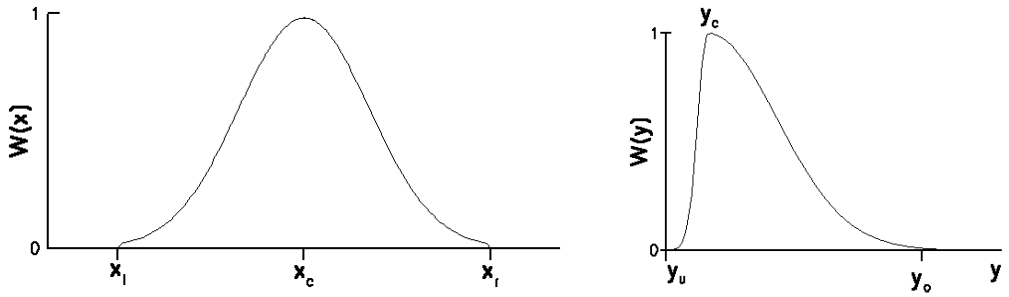
C is a constant with which the whole weighting function can be multiplied. In general, C is equal one. The implemented $W(x)$ and $W(y)$ are given in the following equations:

$$W(x) = \begin{cases} \exp\left[-C_x\left(\dfrac{x-x_c}{x_c-x_l}\right)^2\right] & \text{for } x_l \le x < x_c \\ \exp\left[-C_x\left(\dfrac{x-x_c}{x_c-x_r}\right)^2\right] & \text{for } x_c \le x \le x_r \end{cases} , \tag{2.76}$$

$$W(y) = \begin{cases} \exp\left[-C_y\left(\dfrac{y-y_c}{y_c-y_u}\right)^2\right] & \text{for } y_u \le y < y_c \\ \exp\left[-C_y\left(\dfrac{y-y_c}{y_c-y_o}\right)^2\right] & \text{for } y_c \le y \le y_o \end{cases} . \tag{2.77}$$

C_x and C_y are constants which can influence the form of the exponential functions $W(x)$ and $W(y)$ whose location of the maxima are indicated by x_c and y_c.

If the source term is moved through the patch into the x-direction because of the mean flow for example, $W(x)$ should have a certain minimum expansion in order to minimize unwanted sound waves or turbulence by fading in and out the source term with $W(x)$. Then the recommended length of the fading in area as well as fading out area (x_l, x_r) depends on the appearing x-components of the wave lengths of the source term, see therefore [KW00, EMS02, Bau03]. Figure 2.9a describes $W(x)$ qualitatively. The function $W(y)$ (depicted in Figure 2.9b) can be



(a) x-component (b) y-component

Figure 2.8: The weighting function $W(x, y)$

approximated to the history of the kinetic turbulence energy calculated with the RANS computation in a flow in x-direction for example.

2.5.4 Synthetic SNGR Turbulence

Following [BJ99, BLC95, Lon98] for the synthetic turbulent velocity vector generated by the SNGR model applies

$$\vec{v}_t(\vec{x}, t) = 2\sum_{n=1}^{N} \hat{v}_n \vec{\sigma}_n \cos(\vec{\alpha}_n(\vec{x} - t\vec{v}_{\text{SNGR}}) + \Psi_n + 2\pi f_n t) \ . \tag{2.78}$$

Thus $\vec{v}_t(\vec{x}, t)$ is given by a sum of N FOURIER modes. A set of N mode parameters ($\hat{v}_n$, $\vec{\alpha}_n$, Ψ_n, f_n, $\vec{\sigma}_n$) is called a *mode realisation*. Each mode has

- a velocity amplitude $\hat{v}_n$,
- a wave vector $\vec{\alpha}_n$,
- a phase angle Ψ_n,
- a fluctuation frequency f_n and
- a unit vector $\vec{\sigma}_n$.

The direction of each wave vector, the phase angles, the fluctuation frequency and the unit vector are defined with the help of certain probability distributions in order to simulate the stochastic character of real turbulences. The wanted convection velocity of the synthetic turbulence field is given by the parameter $\vec{v}_{\mathrm{SNGR}}$. The velocity amplitude of each mode is calculated from the appropriate wavenumber $|\vec{\alpha}_n|$ with the help of a turbulence spectrum and parameters of the previously calculated mean flow (kinetic turbulence energy etc.)

Important simplifications concerning $\vec{v}_{\mathrm{t}}(\vec{x}, t)$ are:

- The synthetic turbulence field is isotropic and homogenous. This is achieved by a rectangular distribution of the directions of the wave vectors $\vec{\alpha}_n$ among other things.
- One assumes that the synthetic turbulence is incompressible ($\varrho = \text{const.}$). From equation (2.1) on page 8 follows then with $\vec{v} = \vec{v}_{\mathrm{t}}$

$$\mathrm{div}(\vec{v}_{\mathrm{t}}) = 0 \ . \tag{2.79}$$

  If equation (2.79) is applied to (2.78) on the preceding page one will get an important relation between $\vec{\alpha}_n$ and $\vec{\sigma}_n$:

$$\vec{\alpha}_n \cdot \vec{\sigma}_n = 0 \ . \tag{2.80}$$

  In order to get an incompressible synthetic turbulence the vectors $\vec{\alpha}_n$ and $\vec{\sigma}_n$ have to be perpendicular at each mode.

More details about the synthetic turbulence can be found in [Bau03, KW00].

Chapter 3

Numerical Algorithm

The differential equation system (2.24) on page 17 is solved numerically subject to the given boundary and initial conditions.

3.1 Spatial Discretization

Spatial gradients are approximated using the dispersion relation preserving 7-point stencil finite difference scheme (DRP) of Tam & Webb [TW92] on curvilinear (block-) structured grids, see e. g. [GLL01]. The basic idea of that scheme is to minimize the numerical dispersion introduced by the discretization for a chosen wavenumber range, cf. Figure 3.1. The price to pay is a reduced order of accuracy of the spatial discretization (4^{th} order instead of 6^{th} order). The physical grid

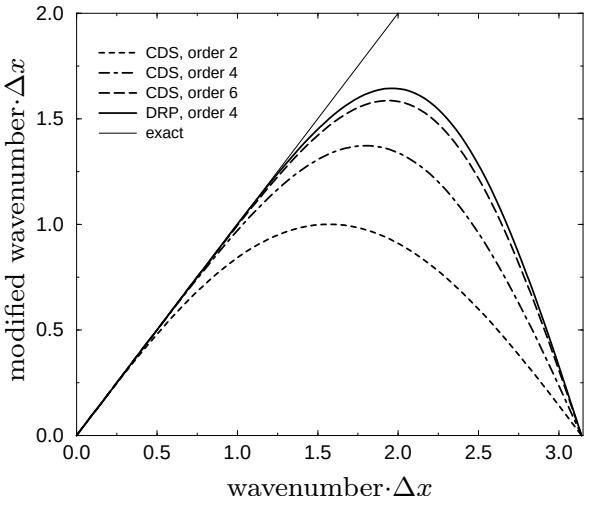


Figure 3.1: Spectral functions of several finite difference schemes

is given as node sequence in the indices i, j, k

$$\vec{x}_{ijk} = \vec{x}(\xi = i,\ \eta = j,\ \zeta = k) \tag{3.1}$$

where ξ, η and ζ represent a uniform cartesian system and assume integer values on the nodes. For instance, for fixed $\xi = I$ the variables η, ζ define a curved grid surface in physical space,

for fixed $\xi = I$, $\eta = J$ the variable ζ defines a curved grid line in physical space. Since the coefficients of the DRP scheme are defined for the uniform computation grid $\xi = i$, $\eta = j$ and $\zeta = k$ the perturbation equations (2.5) on page 10 need to be transformed from the physical domain to the computational domain ξ, η, ζ. This is done by replacing ∇_x by

$$\nabla_x = J\nabla_\xi \tag{3.2}$$

where $J_{mn} = {}^{\partial\xi_n}/_{\partial x_m}$, $(m, n) = 1, 2, 3$ is the metric of the transform and $(x_1, x_2, x_3) = (x, y, z)$ as well as $(\xi_1, \xi_2, \xi_3) = (\xi, \eta, \zeta)$. The metric is obtained by inverting $J^{-1} = \nabla_\xi \vec{x}(\xi, \eta, \zeta)$, which is available with high accuracy employing the DRP differencing scheme along the grid lines. The metric is needed accurately in order that the high resolution and accuracy properties of the DRP scheme would be transferred into the physical space.

In general, PIANO uses a symmetric[1] 7-point stencil to approximate the first derivative numerically. On a uniformly spaced one-dimensional grid with spacing Δx this reads

$$\left.\frac{\partial\phi}{\partial x}\right|_i = \frac{1}{\Delta x}\sum_{l=-3}^{+3} c_l \phi_{i+l}. \tag{3.3}$$

At boundaries it is not possible to stay with symmetric 7-point stencils. Here, unsymmetric 7-point calculation molecules are used. The coefficients c_l for symmetric and unsymmetric 7-point stencils can be found in Appendix D on page 111. Using the given DRP-coefficients guarantees 4^{th}-order accuracy of the spatial discretization.

3.2 Time Integration

The temporal discretization, selected by `NoRKS` (see Section 6.10 on page 76), is currently done with a 4^{th}-order Runge-Kutta scheme, either the classical 4- or 6-stage Runge-Kutta scheme or the well-known *l*ow-*d*issipation, low-*d*ispersion *R*unge-*K*utta (LDDRK) algorithm.

They are self-starting and relatively stable. A general 4-stage Runge Kutta scheme for an evolution equation

$$\frac{\partial\vec{U}}{\partial t} = \vec{F}(\vec{U}) \tag{3.4}$$

has the form

$$\vec{U}^{n+1} = \vec{U}^n + \Delta t\sum_{i=1}^{4} a_i \vec{K}_i \tag{3.5}$$

with

$$\vec{K}_i = \vec{F}\left(\vec{U}^n + \Delta t\sum_{j=1}^{i-1} b_{ij}\vec{K}_j\right), \tag{3.6}$$

where $\vec{U}^n := \vec{U}(t = n\,\Delta t)$; the time step is denoted Δt and n is the current time level.

[1] The reason for prefering symmetric stencils rather than unsymmetric ones comes from the fact, that in combination with anti-symmetric coefficients c_l the numerical derivative is non-dissipative.

The classical Runge-Kutta scheme has the following coefficients:

$$\begin{array}{lllll} a_i =: & a_1 = \frac{1}{6} & b_{ij} =: & b_{21} = \frac{1}{2} & \\ & a_2 = \frac{1}{3} & & b_{31} = 0 & b_{32} = \frac{1}{2} \\ & a_3 = \frac{1}{3} & & b_{41} = 0 & b_{42} = 0 \quad b_{43} = 1 \\ & a_4 = \frac{1}{6} & & & \end{array} \tag{3.7}$$

The time marching algorithm $\vec{U}^n \longrightarrow \vec{U}^{n+1} := \vec{U}_4^{n+1}$ consists of 4 stages for each time step:

$$\begin{array}{lllll} \text{stage 1:} & \vec{U}_1^{n+1} = \vec{U}^n + \frac{\Delta t}{6}\vec{K}_1 & \text{with} & \vec{K}_1 = \vec{F}\left(\vec{U}^n\right) \\ \text{stage 2:} & \vec{U}_2^{n+1} = \vec{U}_1^{n+1} + \frac{\Delta t}{3}\vec{K}_2 & \text{with} & \vec{K}_2 = \vec{F}\left(\vec{U}^n + \frac{1}{2}\Delta t\vec{K}_1\right) \\ \text{stage 3:} & \vec{U}_3^{n+1} = \vec{U}_2^{n+1} + \frac{\Delta t}{3}\vec{K}_3 & \text{with} & \vec{K}_3 = \vec{F}\left(\vec{U}^n + \frac{1}{2}\Delta t\vec{K}_2\right) \\ \text{stage 4:} & \vec{U}_4^{n+1} = \vec{U}_3^{n+1} + \frac{\Delta t}{6}\vec{K}_4 & \text{with} & \vec{K}_4 = \vec{F}\left(\vec{U}^n + \Delta t\vec{K}_3\right) \end{array} \tag{3.8}$$

3.2.1 Low-dissipation and Low-dispersion Runge-Kutta Scheme (LDDRK)

Moreover, an alternating two-step *l*ow-*d*issipation, low-*d*ispersion *R*unge-*K*utta scheme (LDDRK) proposed by Hu et al. [HHM96] is implemented in PIANO. The coefficients are chosen in such a way that the dissipation and dispersion errors are minimized without compromising the stability limits. Combining two alternating steps in the optimization, the dispersion errors are further reduced and a higher order of accuracy is maintained. The LDDRK implementation exploits the advantage of low storage requirements. In the following, the low storage version as it is implemented in PIANO is outlined for the 5–6 stage LDDRK scheme. For $i = 1 \ldots p$, and $p = 5, 6$ as well as $\overline{\beta}_1 = 0$ equation (3.5) on the previous page reads in discretized form

$$\vec{U}^{n+1} = \vec{U}^n + \vec{K}_p \tag{3.9}$$

with

$$\vec{K}_i = \Delta t\vec{F}\left(\vec{U}^n + \overline{\beta}_i\vec{K}_{i-1}\right) . \tag{3.10}$$

The coefficients $\overline{\beta}_i$ read

$$c_2 = \overline{\beta}_p , \tag{3.11}$$

$$c_3 = \overline{\beta}_p\overline{\beta}_{p-1} , \tag{3.12}$$

$$\vdots$$

$$c_p = \overline{\beta}_p\overline{\beta}_{p-1}\cdots\overline{\beta}_2 . \tag{3.13}$$

The coefficients c_p for the alternating 5-stage and 6-stage Runge-Kutta scheme are given in Table 3.1 on the facing page.

3.3 Numerical Damping

Very short wave length components of the signals which cannot be represented physically correctly on the given computation grid may be suppressed with artificial selective damping (ASD)

Stages	c1	c2	c3	c4	c5	c6
5	1	0.5	0.166558	0.0395041	0.00781071	
6	1	0.5	1/3!	1/4!	0.00781005	0.00132141

Table 3.1: Optimized coefficients for amplification factor

due to Tam, Webb & Dong [TWD93]. For each of the equations of the system (2.24) on page 17 the same symmetric linear, scalar damping operator $\tilde{D}$ is introduced:

$$\tilde{D}(\dots) = \left|\frac{\partial\vec{x}}{\partial\xi}\right|^{-1} D_\xi(\dots) + \left|\frac{\partial\vec{x}}{\partial\eta}\right|^{-1} D_\eta(\dots) + \left|\frac{\partial\vec{x}}{\partial\zeta}\right|^{-1} D_\zeta(\dots) \text{ with } \left|\frac{\partial\vec{x}}{\partial\xi_m}\right| = \sqrt{\left(\frac{\partial\vec{x}}{\partial\xi_m}\right)^2}. \quad (3.14)$$

The subscripts on D indicate the grid line direction along which the operator is to be applied, e. g.

$$(D_\xi(\phi))_{i,j,k} = \sum_{l=-N}^{+N} d_l \phi_{i+l,j,k} \; . \quad (3.15)$$

Throughout the calculation domain the damping operator employs symmetric 7-point stencils ($N = 3$). Near boundaries, where a symmetric 7-point molecule cannot be built up, 5-point and 3-point stencils are used. Directly at the boundary continuity and momentum equation do not allow for a damping operator at all. Due to the use of the ghost point concept at slip walls, a 3-point damping operator can still be built up for the pressure using the pressure value of the ghost point. Hence, damping is included in the energy equation also at slip wall boundaries.
Depending on the wavenumber range, which artificial selective damping should work on, different damping coefficients have been developed. In PIANO the following coefficients for $N = 3$, $N = 2$, $N = 1$ are used (handling details are given in Section 6.4 on page 67):

d_{-3}	-0.02385304819				
d_{-2}	0.10630357877	d_{-2}	0.0625		
d_{-1}	-0.22614695181	d_{-1}	-0.2500	d_{-1}	-0.25
d_0	0.28739284246 ,	d_0	0.3750 ,	d_0	0.50 ,
d_1	-0.22614695181	d_1	-0.2500	d_1	-0.25
d_2	0.10630357877	d_2	0.0625		
d_3	-0.02385304819				

Table 3.2: Coefficients for damping stencils

which correspond to $\sigma = 0.3$ in [TWD93].

The damping operator is supplemented as sink term to the right hand side of (2.24) on page 17 thus one finally gets

$$\begin{aligned}
\frac{\partial\varrho'}{\partial t} + \vec{v}'\cdot J\nabla_\xi\varrho_0 + (\vec{v}_0+\varepsilon\vec{v}')\cdot J\nabla_\xi\varrho' + (J\nabla_\xi)\cdot\vec{v}_0\,\varrho' + (J\nabla_\xi)\cdot\vec{v}'(\varrho_0+\varepsilon\varrho') &= -\nu_{\text{ASD}}\tilde{D}(\varrho')\,,\\
\frac{\partial\vec{v}'}{\partial t} + \vec{v}'\cdot J\nabla_\xi\vec{v}_0 + (\vec{v}_0+\varepsilon\vec{v}')\cdot J\nabla_\xi\vec{v}' + \frac{1}{\varrho_0}\left(1-\varepsilon\frac{\varrho'}{\varrho_0}\right)\left(J\nabla_\xi p' + \varrho'\,\vec{v}_0\cdot J\nabla_\xi\vec{v}_0\right) &= -\nu_{\text{ASD}}\tilde{D}(\vec{v}')\,,\\
\frac{\partial p'}{\partial t} + \vec{v}'\cdot J\nabla_\xi p_0 + (\vec{v}_0+\varepsilon\vec{v}')\cdot J\nabla_\xi p' + \kappa\left[(J\nabla_\xi)\cdot\vec{v}_0\,p' + (J\nabla_\xi)\cdot\vec{v}'(p_0+\varepsilon p')\right] &= -\nu_{\text{ASD}}\tilde{D}(p')\,.
\end{aligned} \quad (3.16)$$

The damping coefficient ν_{ASD}, adjusted by `damping`, must be chosen such that i) non-physical, i. e. purely numerically caused signal components are efficiently damped while affecting the

physical wave components is as little as possible, and ii) no numerical instability of the overall scheme is generated. For changing spatial resolution of the grid the artificial damping viscosity ν_{ASD} (the inverse of which is also called grid Reynolds number) remains constant.

For special situations it might be useful to limit the damping term to a distinct physical area:

- To damp only the solution on slip walls a so-called *wall-damping* is implemented.
- The spatial shape of the damping coefficient may be influenced by user-defined damping spots which are superposed with the constant global damping.
- As a special damping spot convected by the local mean velocity the so-called *sponge bath-tub* can be regarded: the gaussian shape centred with the localised vortex disappears temporal but erases the spurious noise generated by initialization appropriately. Thus the right hand side of (2.24) on page 17 is modified ($\ldots = -\sigma(x)\exp(-t/\tau)\cdot\phi$) by $\sigma(x) = \hat{\sigma}\left[1+\tanh\left(\frac{r}{b_{\mathrm{SBT}}}-1\right)-\tanh\left(\frac{r}{b_{\mathrm{SBT}}}+1\right)\right]$ with $r=\sqrt{(x_i-x_i^{\mathrm{SBT}})^2}$ where x_i denotes the physical coordinates and b_{SBT} the half-value radius of the tub while for the current tub centre $x_i^{\mathrm{SBT}} = x_i^{\mathrm{c}} + u_i^0 t$, moving in time t due to the local mean velocity (cf. Section 6.3.3 on page 64), applies.

The application of all these features is explained in detail in Section 6.4 on page 67, too.

In practical applications artificial selective damping has shown to enhance numerical instability, when high values of ν_{ASD} were used. On the other hand, too low ν_{ASD} may also end in numerical instabilities due to unsymmetric spatial discretization stencils near boundaries. Furthermore, evaluating damping terms every Runge-Kutta stage of each time step is somewhat time consuming. For these reasons the artificial selective damping option may be replaced by a filtering technique.

3.4 Filtering

Another approach to eliminate spurious oscillations is filtering [Sha99, VLM98]. Here the solution is filtered and the governing equations are not affected. The filtering procedure along e. g. direction ξ may be expressed as

$$F_\xi(\phi)_{i,j,k} = \sum_{l=-N}^{+N} f_l \phi_{i+l,j,k} \; . \tag{3.17}$$

The filtering operation on the complete field $\phi_{i,j,k}$ is successively applied along the ξ-, η- and ζ-direction. This procedure is very effective and faster than artificial selective damping. Currently, 3 different filters, selected by `Filter` (see Section 6.4 on page 67 for handling details), are implemented in PIANO: a 6^{th} order ($N=3$) and an 8^{th} order filter ($N=4$); the third one is described in the next section. Figure 3.2 on the facing page shows the spectral transfer function of both filters and of ASD. It points out that both filters have the same characteristic as ASD: The higher the frequency the lower is the transfered amplitude, i. e. due to the used grid spacing unresolvable waves with too high frequencies will be suppressed. The 8^{th} order filter does slightly the best job. Furthermore, filtering can be applied at distinctive time intervals, e. g. every 5^{th}, 10^{th} or 20^{th} time step, which diminishes the effect on the solution (and on the performance, of course).

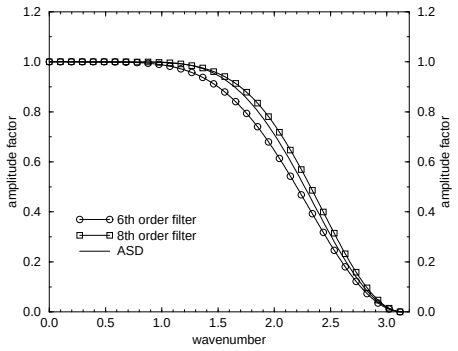


Figure 3.2: Spectral functions of filters and ASD (filtering procedure applied each time step)

Normal to boundaries no symmetric filter stencil can be constructed. Practical applications have shown that filtering at slip walls is very important for a numerically stable solution.[2] For this reason, as one option "diagonal smoothing" may be employed there, cf. Figure 3.3. The filter

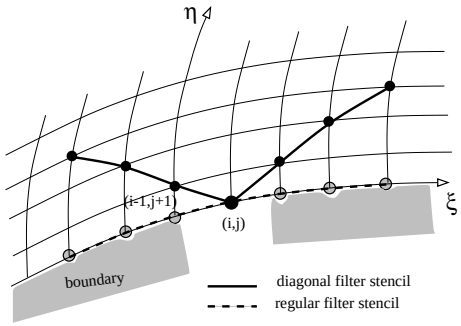


Figure 3.3: Filtering at a $j_{\min}$-boundary (slip wall)

for node (i, j, k) on a boundary has the following form, e. g. for a $\eta = j_{\min}$-face (slip wall):

$$\begin{aligned}
\xi\text{-direction:} \quad F_\xi(\phi)_{i,j,k} &= \sum_{l=-N}^{+N} f_l \phi_{i+l,j,k} \ , \\
\eta\text{-direction:} \quad F_\eta(F_\xi)_{i,j,k} &= \sum_{l=-N}^{+N} f_l ((F_\xi)_{i+l,j+|l|,k} + (F_\xi)_{i,j+|l|,k+l}) \ , \\
\zeta\text{-direction:} \quad F_\zeta(F_\eta)_{i,j,k} &= \sum_{l=-N}^{+N} f_l (F_\eta)_{i,j,k+l} \ .
\end{aligned} \tag{3.18}$$

Though this approach is heuristic, it has shown to be very effective for grids with very small spacing in the wall normal direction compared to the spacing along the wall at highly convex body boundaries. The table below gives the coefficients used in PIANO for both the 6^{th}- and

[2] Filtering at boundaries is currently under development. Up to now a final formulation has not yet been found. The current version may be regarded as preliminary.

8^{th}-order filter (note $f_{-i} = f_i$):	6^{th} order	8^{th} order
$f_0 :=$	0.68750	0.7265625
$f_1 :=$	0.46875	0.4375000
$f_2 :=$	−0.18750	−0.2187500
$f_3 :=$	0.03125	0.0625000
$f_4 :=$		−0.0078125

Table 3.3: Coefficients for filter stencils

As another option a filtering of the wall nodes and their neighbouring nodes may be achieved by artificially mirroring them about the boundary point into the domain adjacent to the wall (cf. Figure 3.4). Then symmetric filtering following (3.18) on the previous page or (3.19) on this page is applied to this artificially extended set of points.

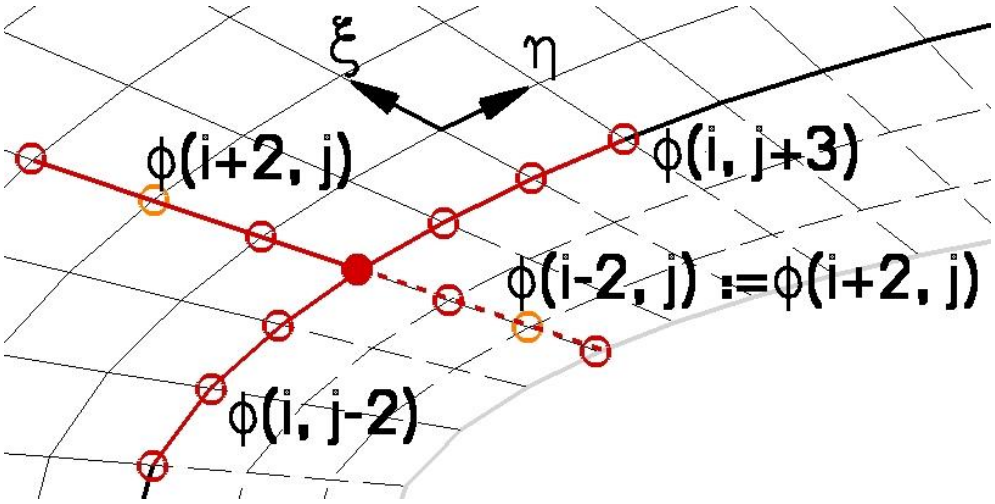


Figure 3.4: Stencil for a wall node $\phi(i, j)$ after artificially mirroring

3.4.1 Padé Filtering

Although more time consuming and difficult to adjust the most effective coefficients it is sometimes advisable to use the implicite padé filtering by `filter = 100`. For the tridiagonal equation system

$$\alpha \hat{f}_{i-1} + \alpha \hat{f}_i + \alpha \hat{f}_{i+1} = a f_i + \frac{c}{2}(f_{i-2} + f_{i+2}) + \frac{b}{2}(f_{i-1} + f_{i+1}) \ , \tag{3.19}$$

where $\hat{f}_i$ represents the filtered value at node x_i, three schemes following the Appendix C in [Lel92] are implemented: The fourth-order accurate scheme 1, the sixth-order accurate scheme 2 and a more 'smooth' scheme 3. All coefficients are subject to $\alpha =$ `PadeAlpha` in the range of about 0.3 up to 0.5 (the closer to 0.5 the less the effect!); for handling details see Section 6.4 on page 67.

	1	2	3	scheme
a	$(5+6\alpha)/8$	$(11+10\alpha)/16$	$(2+3\alpha)/4$	
b	$(1+2\alpha)/2$	$(15+34\alpha)/32$	$(9+16\alpha)/16$	
c	$(2\alpha-1)/8$	$(6\alpha-3)/16$	$\alpha/4$	
d	0	$(1-2\alpha)/32$	$-1/16$	
coeffis				

Chapter 4

Boundary Conditions

There are five types of physical boundary conditions implemented (handling is described in Section 6.3.5 on page 66):

- outflow,
- radiation,
- slip wall with and without adiabatic condition and
- sponge layer.

4.1 Outflow Boundary Condition

The outflow boundary condition, developed by Tam & Webb [TW92], is used on grid surfaces N which bound the computation domain in regions where the mean flow vector points outwards. The conditions are employed as well on the two curvilinear neighbouring grid surfaces $N-1$ and $N-2$. At an outflow boundary vorticity and entropy perturbations leave the computational domain due to mean-flow convection while pressure perturbations are radiated outwards. The equations to be satisfied are:

$$\frac{\partial \varrho'}{\partial t} + \vec{v}_0 \cdot \nabla \varrho' = \frac{1}{a^2}\left[\vec{v}_0 \cdot \nabla p' + \frac{\partial p'}{\partial t}\right] \tag{4.1}$$

$$\frac{\partial \vec{v}'}{\partial t} + \vec{v}_0 \cdot \nabla \vec{v}' = -\frac{1}{\varrho_0}\nabla p', \tag{4.2}$$

$$\frac{1}{V(\Theta)}\frac{\partial p'}{\partial t} + \frac{\partial p'}{\partial r} + \frac{p'}{r\, c_{2\mathrm{D}}} = 0, \tag{4.3}$$

where $V(\Theta) = |\vec{v}_0|\cos\Theta + \sqrt{a^2 - |\vec{v}_0|^2 \sin^2\Theta}$ and $c_{2\mathrm{D}} = 2$ for two-dimensional calculations (input parameter `RBD2D` set) or $c_{2\mathrm{D}} = 1$ for three-dimensional calculations (input parameter `RBD2D` unset); $a = \sqrt{\kappa\, p_0/\varrho_0}$ denotes the local speed of sound. Distance $r = |\vec{x} - \vec{x}_{\mathrm{ref}}|$ is measured from the centre of acoustic sources `Xref`, the angle $\Theta = \arccos\left(\vec{r}\cdot\vec{v}_0/|\vec{r}||\vec{v}_0|\right)$. The radial derivative $\partial/\partial r = \sin\vartheta(\cos\varphi\,\partial/\partial x + \sin\varphi\,\partial/\partial y) + \cos\vartheta\,\partial/\partial z$ (due to $x = r\sin\vartheta\cos\varphi$, $y = r\sin\vartheta\sin\varphi$, $z = r\cos\vartheta$) occurs according to a transformation from spherical to cartesian coordinates.

4.2 Radiation Boundary Condition

At bounding grid surfaces (and the respective two curvilinear neighbouring surfaces), where there are only outgoing acoustic waves, i. e. inflow and parallel flow boundaries, the radiation boundary condition (Tam & Webb [TW92], cf. (4.3) on the preceding page) is enforced.

$$\left(\frac{1}{V(\Theta)}\frac{\partial}{\partial t}+\frac{\partial}{\partial r}+\frac{1}{r\,c_{2\mathrm{D}}}\right)\begin{pmatrix}\varrho'\\u'\\v'\\w'\\p'\end{pmatrix}=0 \tag{4.4}$$

with $c_{2\mathrm{D}}=2$ for two-dimensional calculations (input parameter `RBD2D` set) or $c_{2\mathrm{D}}=1$ for three-dimensional calculations (input parameter `RBD2D` unset) using the same transformation from spherical to cartesian coordinates as above.

4.3 Wall Boundary Condition

At walls the ghost point concept of Tam & Dong [TD94] is used to fulfill the boundary condition. The idea is to introduce an additional (computational) node beyond the wall boundary, i. e. a node typically located inside the body. The pressure value at such a ghost point is evaluated so that the non-penetration condition at the boundary

$$\vec{n}\cdot\vec{v}=0$$

is guaranteed. Scalar multiplication of the momentum equation (cf. (2.1) on page 8) with the surface normal vector $\vec{n}$ yields an equation for the pressure at the ghost point:

$$\varrho\,v_i\frac{\partial v_j}{\partial x_i}n_j+n_j\frac{\partial p}{\partial x_j}=0.$$

Analogously as in Chapter 2 on page 8 the left hand side is expanded in a Taylor's series in the perturbation parameter ε:

$$N(\varepsilon):=(\bar\varrho+\varepsilon\varrho')(\bar v_i+\varepsilon v_i')\frac{\partial(\bar v_j+\varepsilon v_j')}{\partial x_i}\,n_j+n_j\frac{\partial(\bar p+\varepsilon p')}{\partial x_j}=0.$$

With

$$\begin{aligned}\frac{\partial N}{\partial\varepsilon}&=(\bar\varrho+\varepsilon\varrho')(\bar v_i+\varepsilon v_i')\frac{\partial v_j'}{\partial x_i}n_j+(\bar\varrho+\varepsilon\varrho')\frac{\partial(\bar v_j+\varepsilon v_j')}{\partial x_i}v_i'n_j+\varrho'(\bar v_i+\varepsilon v_i')\frac{\partial(\bar v_j+\varepsilon v_j')}{\partial x_i}n_j+n_j\frac{\partial p'}{\partial x_j}\\
\frac{\partial^2 N}{\partial\varepsilon^2}&=n_j\left\{\frac{\partial v_j'}{\partial x_i}\left[\varrho'(\bar v_i+\varepsilon v_i')+v_i'(\bar\varrho+\varepsilon\varrho')\right]+v_i'\left[\varrho'\frac{\partial(\bar v_j+\varepsilon v_j')}{\partial x_i}+\frac{\partial v_j'}{\partial x_i}(\bar\varrho+\varepsilon\varrho')\right]+\right.\\
&\qquad\left.\varrho'\left[v_i\frac{\partial(\bar v_j+\varepsilon v_j')}{\partial x_i}+\frac{\partial v_j'}{\partial x_i}(\bar v_i+\varepsilon v_i')\right]\right\}\end{aligned}$$

one finally obtains

$$n_j\left\{\frac{\partial p'}{\partial x_j}+\bar\varrho\,v_i'\frac{\partial\bar v_j}{\partial x_i}+\varepsilon\varrho'\bar v_i\frac{\partial v_j'}{\partial x_i}+(\bar v_i+\varepsilon v_i')\left(\bar\varrho\frac{\partial v_j'}{\partial x_i}+\varrho'\frac{\partial\bar v_j}{\partial x_i}\right)\right\}=0 \tag{4.5}$$

equivalent to

$$\frac{\partial p'}{\partial n} = n_j \frac{\partial p'}{\partial x_j} = -n_j \left\{ \frac{\partial \bar{v}_j}{\partial x_i}[\varrho' \bar{v}_i + (\bar{\varrho} + \varepsilon \varrho')v_i'] + \frac{\partial v_j'}{\partial x_i}[(\bar{\varrho} + \varepsilon \varrho')\bar{v}_i + \varepsilon \bar{\varrho}\, v_i'] \right\}. \tag{4.6}$$

The ghost point value of p' is evaluated such that (4.6) is satisfied. For a wall $\eta = j = 0$ this means for instance:

$$\left.\frac{\partial p'}{\partial \eta}\right|_0 = \left[\frac{\partial p'}{\partial n} - \sum_{l=1}^{3} n_l \left(J_{1l} \frac{\partial p'}{\partial \xi} + J_{3l} \frac{\partial p'}{\partial \zeta} \right) \right] \Bigg/ \sum_{l=1}^{3} n_l J_{2l} \tag{4.7}$$

or with

$$\left.\frac{\partial p'}{\partial \eta}\right|_0 \approx \sum_{m=-1}^{5} c_{m+1}\, p'_{i,m,k}$$

the ghost point value of the pressure is obtained as:

$$p'_{i,-1,k} = \frac{1}{c_0} \left\{ - \sum_{m=0}^{5} c_{m+1}\, p'_{i,m,k} + \left[\frac{\partial p'}{\partial n} - \sum_{l=1}^{3} n_l \left(J_{1l} \frac{\partial p'}{\partial \xi} + J_{3l} \frac{\partial p'}{\partial \zeta} \right) \right] \Bigg/ \sum_{l=1}^{3} n_l J_{2l} \right\}. \tag{4.8}$$

4.3.1 Adiabatic Condition on Walls

In addition to the non-penetration condition one can fix the density ϱ' to the pressure p' by the adiabatic condition $\varrho' = {}^{p'}/_{c_0^2}$.

4.4 Sponge Layer

The general sponge layer approach is capable to enforce a given function in the specified layer (in addition to the given boundary condition in this region) due to the supplement $\sigma(\xi)(\phi' - \phi_{\text{ref}})$ subtracted from the right hand side of the equations (2.24) on page 17 to be solved.
The default forcing function ϕ_{ref} is constantly equal to zero, but one can define its own sponge layer function by a user-defined sponge layer type. In this case the definition has to be implemented in file `Sponge.f`, otherwise just the segment's boundary condition has to be specified in `FilLog`.
The default fading function $\sigma(\xi)$ is subject to the distance to the wall.

Chapter 5

Interpolation of Arbitrary Mean-Flows

Input data from an arbitrary mean flow, e. g. calculated by a solver of the Reynolds averaged Navier-Stokes equations (RANS) like the DLR FLOWer code, are provided by means of interpolation by a separate program called interpol. This interpolation procedure is generally necessary because different grids have to be utilized in RANS and CAA[1] calculations. The reason for this are the different numerical requirements in both cases. E. g. the RANS calculations need very fine grids in the vicinity of the body, whereas the solution of the linearized Euler equations (LEE) require a sufficient fine resolution in the far field of the body.

Here, only a brief introduction into the utilized interpolation algorithm will be given. Information for using the interpol program are distributed with the source.

The interpolation procedure between the RANS grid and the CAA grid is complicated by two circumstances. First, the two grids can be located almost arbitrary in three-dimensional space; the only restriction is that the CAA grid has to be enclosed by the RANS grid. This results in an expensive search for appropriate initial conditions for the iterative determination of the parameters of a node of the CAA grid. The second one is the very fact that in complicated three-dimensional calculations, some million CAA nodes are to be searched for in some million RANS cells. This poses great demands upon the speed of the algorithm. Therefore an easy to evaluate, local polynomial approximation was chosen for the representation of the grid functions.

[1] **C**omputational **A**ero**a**coustics

5.1 The Interpolating Polynomial

The flow solver FLOWer utilizes a general curvilinear multi block structure. Every block of the grid consists of the $n_1 \times n_2 \times n_3$ nodes

$$\vec{x}_{ijk} = \begin{pmatrix} x_{ijk} \\ y_{ijk} \\ z_{ijk} \end{pmatrix}, \quad 0 < i < n_1 - 1\,, \quad 0 < j < n_2 - 1\,, \quad 0 < k < n_3 - 1\,. \tag{5.1}$$

Every flow variable in the FLOWer code is defined at these grid nodes. It is convenient to consider the grid coordinates and every other function defined on the block as a function of the grid indices, i. e. the parameters (ξ, η, ζ) are introduced in such a way that e. g.

$$\vec{x}_{ijk} = \vec{x}(\xi = i,\ \eta = j,\ \zeta = k). \tag{5.2}$$

In the following, f symbolises any variable that is defined at the grid nodes of the RANS grid. The interpolation problem can now be formulated as follows

1. Choose an appropriate representation of a function $f(\xi, \eta, \zeta)$ on the block.

2. Determine for a given node $\vec{y}$ inside the block the values (ξ, η, ζ) from the vector equation $\vec{y} = \vec{x}(\xi, \eta, \zeta)$.

Since one has to interpolate a vast amount of nodes (up to some millions in three-dimensional calculations), one needs a fast interpolation method with acceptable accuracy. This leads to a polynomial interpolation of the function f on every cell of the block. Therefore, now a cell $i \le \xi \le i + 1$, $j \le \eta \le j + 1$, $k \le \zeta \le k + 1$ of the block is considered and local coordinates $u = \xi - i$, $v = \eta - j$, $w = \zeta - k$ are introduced in this cell. The interpolation problem now boils down to a polynomial interpolation of a function on a unit cell $0 \le u \le 1$, $0 \le v \le 1$, $0 \le w \le 1$, where the function values (and perhaps appropriate derivates of the function) are given at the corners of this unit cell. The most simple interpolation is the linear one:

$$f(u, v, w) = a_{000} + a_{100}u + a_{010}v + a_{001}w + a_{110}uv + a_{101}uw + a_{011}vw + a_{111}uvw\,. \tag{5.3}$$

The eight coefficients $a_{000}, \ldots, a_{111}$ can be determined by the eight function values $f^{000} = f(0, 0, 0), \ldots, f^{111} = f(1, 1, 1)$ at the edges of the cell. Unfortunately, the accuracy of this interpolation is quite poor. However an improvement can be achieved using a higher order polynomial. In this case one needs additional information from the neighbouring cells or from derivatives at the corners of the cell in order to determine the larger number of coefficients. In order to construct a local approximation the second approach was chosen, using appropriate values for the 2^{nd} derivatives of f. It is convenient to consider the following polynomial up to third order

in u, v and w as interpolating function

$$
\begin{aligned}
f(u,v,w) = a_{000} + a_{111}uvw &+ \begin{pmatrix} a_{110} \\ a_{011} \\ a_{101} \end{pmatrix}^T \begin{pmatrix} uv \\ vw \\ uw \end{pmatrix} + \begin{pmatrix} a_{100} \\ a_{010} \\ a_{001} \end{pmatrix}^T \begin{pmatrix} u \\ v \\ w \end{pmatrix} \\
&+ \begin{pmatrix} a_{200} \\ a_{020} \\ a_{002} \end{pmatrix}^T \begin{pmatrix} u^2 \\ v^2 \\ w^2 \end{pmatrix} + \begin{pmatrix} a_{210} \\ a_{201} \\ a_{120} \\ a_{021} \\ a_{012} \\ a_{102} \end{pmatrix}^T \begin{pmatrix} u^2v \\ u^2w \\ uv^2 \\ v^2w \\ vw^2 \\ uw^2 \end{pmatrix} + \begin{pmatrix} a_{211} \\ a_{121} \\ a_{112} \end{pmatrix}^T \begin{pmatrix} u^2vw \\ uv^2w \\ uvw^2 \end{pmatrix} \\
&+ \begin{pmatrix} a_{300} \\ a_{030} \\ a_{003} \end{pmatrix}^T \begin{pmatrix} u^3 \\ v^3 \\ w^3 \end{pmatrix} + \begin{pmatrix} a_{310} \\ a_{301} \\ a_{130} \\ a_{031} \\ a_{013} \\ a_{103} \end{pmatrix}^T \begin{pmatrix} u^3v \\ u^3w \\ uv^3 \\ v^3w \\ vw^3 \\ uw^3 \end{pmatrix} + \begin{pmatrix} a_{311} \\ a_{131} \\ a_{113} \end{pmatrix}^T \begin{pmatrix} u^3vw \\ uv^3w \\ uvw^3 \end{pmatrix}
\end{aligned}
\tag{5.4}
$$

Now one has to determine the 32 coefficients $a_{000}, a_{100}, \ldots, a_{113}$. Eight equations are provided by the function values at the corners of the cell. Another 24 have to be determined from appropriate derivatives at the corners. Since only the function values are provided by the RANS calculation, one has to determine sufficiently accurate approximations of the derivatives at the grid nodes numerically. One way to do this is a (one-dimensional) cubic spline interpolation of the function along every grid line. A cubic spline has continuous derivatives up to second order and therefore values for the second derivatives f_{uu}, f_{vv} and f_{ww} at the grid nodes can easily be obtained. An inspection of the second derivatives of the interpolating polynomial

$$
\begin{aligned}
f_{uu} &= 2a_{200} + 6a_{300}u + 2a_{210}v + 2a_{201}w + 6a_{310}uv + 6a_{301}uw + 2a_{211}vw + 6a_{311}uvw \\
f_{vv} &= 2a_{020} + 2a_{120}u + 6a_{030}v + 2a_{021}w + 6a_{130}uv + 2a_{121}uw + 6a_{031}vw + 6a_{131}uvw \\
f_{ww} &= 2a_{002} + 2a_{102}u + 2a_{012}v + 6a_{003}w + 2a_{112}uv + 6a_{103}uw + 6a_{013}vw + 6a_{113}uvw
\end{aligned}
\tag{5.5}
$$

yields that the coefficients can be computed successively from the following recursively solvable linear systems (zero entries are marked with (.).).

$$\begin{pmatrix} 2 & . & . & . & . & . & . & . \\ 2 & 6 & . & . & . & . & . & . \\ 2 & . & 2 & . & . & . & . & . \\ 2 & . & . & 2 & . & . & . & . \\ 2 & 6 & 2 & . & 6 & . & . & . \\ 2 & 6 & . & 2 & . & 6 & . & . \\ 2 & . & 2 & 2 & . & . & 2 & . \\ 2 & 6 & 2 & 2 & 6 & 6 & 2 & 6 \end{pmatrix} \begin{pmatrix} a_{200} \\ a_{300} \\ a_{210} \\ a_{201} \\ a_{310} \\ a_{301} \\ a_{211} \\ a_{311} \end{pmatrix} = \begin{pmatrix} f_{uu}^{000} \\ f_{uu}^{100} \\ f_{uu}^{010} \\ f_{uu}^{001} \\ f_{uu}^{110} \\ f_{uu}^{101} \\ f_{uu}^{011} \\ f_{uu}^{111} \end{pmatrix} \tag{5.6}$$

$$\begin{pmatrix} 2 & . & . & . & . & . & . & . \\ 2 & 2 & . & . & . & . & . & . \\ 2 & . & 6 & . & . & . & . & . \\ 2 & . & . & 2 & . & . & . & . \\ 2 & 2 & 6 & . & 6 & . & . & . \\ 2 & 2 & . & 2 & . & 2 & . & . \\ 2 & . & 6 & 2 & . & . & 6 & . \\ 2 & 2 & 6 & 2 & 6 & 2 & 6 & 6 \end{pmatrix} \begin{pmatrix} a_{020} \\ a_{120} \\ a_{030} \\ a_{021} \\ a_{130} \\ a_{121} \\ a_{031} \\ a_{131} \end{pmatrix} = \begin{pmatrix} f_{vv}^{000} \\ f_{vv}^{100} \\ f_{vv}^{010} \\ f_{vv}^{001} \\ f_{vv}^{110} \\ f_{vv}^{101} \\ f_{vv}^{011} \\ f_{vv}^{111} \end{pmatrix} \tag{5.7}$$

$$\begin{pmatrix} 2 & . & . & . & . & . & . & . \\ 2 & 2 & . & . & . & . & . & . \\ 2 & . & 2 & . & . & . & . & . \\ 2 & . & . & 6 & . & . & . & . \\ 2 & 2 & 2 & . & 2 & . & . & . \\ 2 & 2 & . & 6 & . & 6 & . & . \\ 2 & . & 2 & 6 & . & . & 6 & . \\ 2 & 2 & 2 & 6 & 2 & 6 & 6 & 6 \end{pmatrix} \begin{pmatrix} a_{002} \\ a_{102} \\ a_{012} \\ a_{003} \\ a_{112} \\ a_{103} \\ a_{013} \\ a_{113} \end{pmatrix} = \begin{pmatrix} f_{ww}^{000} \\ f_{ww}^{100} \\ f_{ww}^{010} \\ f_{ww}^{001} \\ f_{ww}^{110} \\ f_{ww}^{101} \\ f_{ww}^{011} \\ f_{ww}^{111} \end{pmatrix} \tag{5.8}$$

$$\begin{pmatrix} 1 & . & . & . & . & . & . & . \\ 1 & 1 & . & . & . & . & . & . \\ 1 & . & 1 & . & . & . & . & . \\ 1 & . & . & 1 & . & . & . & . \\ 1 & 1 & 1 & . & 1 & . & . & . \\ 1 & 1 & . & 1 & . & 1 & . & . \\ 1 & . & 1 & 1 & . & . & 1 & . \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_{000} \\ a_{100} \\ a_{010} \\ a_{001} \\ a_{110} \\ a_{101} \\ a_{011} \\ a_{111} \end{pmatrix} = \begin{pmatrix} f^{000} \\ f^{100} - a_{200} - a_{300} \\ f^{010} - a_{020} - a_{030} \\ f^{001} - a_{002} - a_{003} \\ f^{110} - \dots \\ f^{101} - \dots \\ f^{011} - \dots \\ f^{111} - \dots \end{pmatrix} \tag{5.9}$$

5.2 Calculation of the Parameters

Once an interpolating polynomial can be constructed for any function $f(\xi, \eta, \zeta)$, the main problem of the grid to grid interpolation can be tackled, i. e. the determination of the parameter values (ξ, η, ζ) for an arbitrary point $\vec{y}$ inside the CFD grid. This can be split into two parts:

1. Find the appropriate block of the CFD grid and some initial conditions for the parameter values.
2. Solve the vector equation $\vec{y} = \vec{x}(\xi, \eta, \zeta)$ for (ξ, η, ζ) using the polynomial approximation above for every component function of $\vec{x}$.

The crucial point in this algorithm is to provide appropriate initial conditions for the parameter. First, $\vec{y}$ is checked against the bounding box of the block. If $\vec{y}$ is inside the bounding box, initial conditions are searched in every cell of the block. In order to do this, $\vec{x}(\xi, \eta, \zeta)$ is approximated on every cell by a linear function. Initial values for (ξ, η, ζ) can than be found by solution of a simple system of linear equations. This procedure works for most of the cells of the CFD grid. If, however, the faces of a cell have strong curvature and/or if the aspect ratio of the cell takes very large values, this procedure can fail. This can happen especially with cells in the boundary layer at the nose of the profile. Then a second pass is started which uses the parameter of successfully found neighbouring nodes as initial values. If initial conditions are found, a Newton iteration is performed in order to solve $\vec{y} = \vec{x}(\xi, \eta, \zeta)$, which uses the full polynomial approximation of $\vec{x}$.

5.3 Treatment of Wall Points

Points at walls can be treated differently from interior ones, since a wall point in the CAA grid has to be in the CFD grid representation on the wall, too. Therefore, the search for initial conditions can be performed only over the surfaces of the CFD blocks.

The difficulty which arises is that a wall point lies at the surface of a CFD block and the Newton iteration for the determination of the parameter values may run outside the block which deteriorates the convergence of the procedure. Now, however, one parameter value is known from the wall condition and consequently the search can be performed inside the face of the block (and not in the volume of the wall cell). In this case, the determining vector equation $\vec{y} = \vec{x}(\xi, \eta, \zeta)$ has to be replaced by a set of two equations, since one is searching two parameter values only. If, for example, $\zeta = \text{const.}$ denotes the coordinate of a wall, ξ and η are determined such that the difference vector $\vec{y} - \vec{x}(\xi, \eta)$ is perpendicular to both surface tangent vectors $\partial\vec{x}(\xi,\eta)/\partial\xi$ and $\partial\vec{x}(\xi,\eta)/\partial\eta$. This is equivalent to the condition that $|\vec{y} - \vec{x}(\xi, \eta)|$ is minimum. This approach also solves the problem encountered when small differences between the wall surfaces of both grids exist, originating from slightly different spline approximations in both cases.

Chapter 6

Practical Handling of PIANO

The given source code is tested on HP, SGI, Sun machines and personal computers operated by UNIX or Linux as well as on NEC's vector computers SX6 and SX8. Using the MPI library parallel computations are performed on PC clusters with distributed memory and local hard disc drives running the operating system Linux. Thus binary files (e. g. grid, mean-flow and record file) can be given in little- or big-endian format[1].

6.1 Installing the Code

The PIANO source is delivered as a packed archive containing a separate directory with all source files of the current version. Besides an exemplary input file three additional files are given:

`Makefile` is a typical make file, describing all dependencies and the way how to get an executable. It is controlled by several environment variables containing the specific commands for a special system, but should not be used separately! Sometimes it is useful to modify the setting of `DIRECTIVE`.

`MakeCall` is a typical shell script, determining the running system and calling the aforementioned `Makefile` with appropriately set environment variables. The following command line options given in any order control this script:

- `-h` gives a short overview about all command line options.
- `-o` *`ExecutableName`* determines the name of the executable, if the default PIANO is not sufficient, because the naming should indicate the specified preprocessor directives, e. g. `parallel` or `3D`.
- `run` selects these compiler options, which produce an optimised executable.

[1] The adjectives *big-endian* and *little-endian* refer to which bytes are most significant in multi-byte data types and describe the order in which a sequence of bytes is stored in a computer's memory: In a big-endian system, the most significant value in the sequence is stored at the lowest storage address (i. e. first). In a little-endian system, the least significant value in the sequence is stored first.

Many mainframe computers, particularly IBM and SGI, use a big-endian architecture. Most modern computers, including PCs, use the little-endian system.

The terms *big-endian* and *little-endian* are derived from the Lilliputians of Gulliver's Travels, whose major political issue was whether soft-boiled eggs should be opened on the big side or the little side.

- `parallel` (default is `sequential`) enforces a parallel code using the MPI[2] library.
- `2D` (default is `3D`) adds the preprocessor directive `twoD` to the other ones.
- `noplt` (default is `plt`) adds the preprocessor directive `noTecplotLib` to the other ones.
- If one of `g95`, `gfc`, `ifc` or `psc` is given, the compilation will be done with the chosen compiler: g95 (a Fortran 95 compiler recommended by AS/TA, which is based on the GNU Compiler Collection GCC, `http://www.g95.org`), gfortran (a Fortran 95 compiler that is part of GCC, `http://gfortran.info`), Intel's Fortran Compiler (`http://www.intel.com/cd/software/products/asmo-na/eng/compilers/flin/index.htm`) or PathScale's Fortran Compiler (`http://www.pathscale.com`).
- By `-D`*`PianoDirective`* one may set any valid preprocessor directive one likes (cf. Section 6.2 on the next page, a complete list is given in the `Makefile` starting at line 162); some directives will be added automagically in special cases. The directives given on the command line are added to the already predefined ones in the `Makefile` at the end of the command line, thus override them possibly!
- Any *`MakeFileOption`* may be given in the command line to control the `make` run, especially the following make file targets:

  `Run` is the first (and default) target, which starts a PIANO run in the background after the compilation; thus it does not have to be mentioned. The input will be taken from the default input file `Piano.in` and the output will be directed into `Piano.out`, too.

  `ExecutableName`, depending on `-o` *`ExecutableName`* (default: `Piano`), will generate an executable *`ExecutableName`* with the given preprocessor directives.

  `new` does the same as the aforementioned target, but starts from scratch, i. e. without any remnants from previous compilations. This target should be used, when different executables are compiled successively with different preprocessor directives, because the `Makefile` is not able to track the directives' modification.

  `archiv` bundles all files needed to generate a PIANO or RPMCHECKER executable in a compressed tarfile. The tarfile's name is determined by *`ExecutableName`* as well as the current date and version.

  `clean` removes all log files helpful during debugging which are produced by the compilers.

  `cleanup` deletes any product of the compilers, except the specified (default: `Piano`) executable.

  `cleanall` clears the directory, except the source code and all subdirectories.

  `print` concatenates all files needed to generate a PIANO or RPMCHECKER executable in an ASCII file for latter printout.

  `RPM` compiles the RPMCHECKER, a tool for checking RPM stuff or generating FilRPMRec file without PIANO.

Typing just `MakeCall` compiles a three-dimensional, sequential code for debug purpose and starts a run afterwards. To compile a parallel executable called MyPIANO the command line is `MakeCall -o MyPiano parallel MyPiano`.

[2] `http://www-unix.mcs.anl.gov/mpi`

CAVE: Because the make mechanism is not able to detect any change in the defined preprocessor directives *the user* has to take care about that! That is why the target `new` is sometimes useful ...

If the local system is not already set up, one has to extend or modify the file `MakeCall` appropriately; the given comments aim at helping the user to accomplish one's intention.

`run_Piano*` should be used to perform a PIANO run on a high-performance system using a queueing programme. Assuming a machine with local disc space for each node, this pbs script does everything necessary: pre- and postprocessing, i. e. move the files, and even restart of PIANO in case of a relay calculation. The given comments will help the user to configure this batch script for one's needs. One can use it also as a template for one's local cluster system.

6.2 Compiling the Source

The source code contains several versions, so that it is in some cases necessary to specify at compilation by preprocessor flags which parts should be used. Following cases (ordered alphabetically) are distinguished:

- `convergence`: special damping coefficients are used to allow an examination of convergence.
- `debug`: additional output is produced, e. g. the grid metrics is written into one file per block, detailed information about found circle points and/or singular nodes are given (includes `debugMetrik` and `debugSings`).
- `debugCircOut`: even for a positive `CircOut` the files are written in ASCII format keeping PIANO's binary file structure.
- `debugHistoryOut`: even for a positive `HistoryOut` the files are written in ASCII format keeping PIANO's binary file structure.
- `debugLogic`: additional logic information will be printed.
- `debugMetrik`: the grid metrics is written into one Tecplot© file per block.
- `debugSBT`: values of SBT's fade out function `sigma` will be saved into a file.
- `debugSings`: detailed information about singular nodes will be given.
- `debugSponge`: values of sponge layer's forcing function `Uref` will be printed.
- `debugTout`: even for a positive `Tout` the files are written in ASCII format keeping PIANO's binary file structure.
- `KeepOrder`: stick to 8th order of filtering even on cuts, i. e. on the cuts themselves no filtering is applied due to missing ghost points.
- `MirrorWall`: mirror all variables in vicinity of walls in order to use symmetric stencils; excludes `SmoothWall`.
- `noTecplotLib`: activate these code lines which save the results without the Tecplot© library in PIANO's native binary format.

- `parallel`: enable the use of machines with distributed memory and local hard disc drives engaging the MPI library.

- `rpm`: the **r**andom **p**article **m**esh source terms are considered on right hand side of the equations.

  CAVE: This feature is still under construction and currently restricted to two-dimensional cases! Any information about new and/or successful applications, but also curiosities are welcome.

  This flag may be accompanied by one of the following subflags (the definitions are deviated in Section 2.5 on page 20):

  - `haystacking`: models the source term as an instationary flow field.
  - `Langevin`: the sources will decay with convection as proposed in [Ewe07].
  - `tam`: calculates the source term following TAM [TA99] for jet noise, e. g.
  - `rpmmaster`: engages one processor, the so-called *rpm-master*, for RPM business exclusively.
  - `debugRPM`: additional information of RPM model will be printed (into files and on screen), used random seeding will be saved, `FilRPMRec` will be written and read formatted.
  - `RPMdebug`: print additional RPM information for debugging purpose.
  - `RPMoutput`: write additional information concerning RPM model into files.

- `silent`: reduce PIANO's talkativeness, i. e. less warnings occur.

- `SmoothCut`: extend diagonal smoothing at slip walls adjacent to inner cuts (cf. Section 3.4 on page 40); needs `SmoothWall` additionally.

- `SmoothOut`: use diagonal smoothing at outflow boundaries (cf. Section 3.4 on page 40).

- `SmoothWall`: use diagonal smoothing at slip walls (cf. Section 3.4 on page 40); excludes `MirrorWall`.

- `SNGR`: **s**tochastic **n**oise **g**eneration and **r**adiation source terms are calculated on right hand side of the equations.

  CAVE: This feature is currently under construction and will not be supported, because the rpm algorithm is an adequate replacement!

  This flag has to be accompanied by one of the following subflags (the definitions are deviated in Section 2.5.2 on page 32):

  - `onlyLambSNGR`: consider only terms describing the so called *perturbed Lamb vector*.
  - `onlySelfSNGR`: consider only terms describing the so called *self noise*.
  - `onlyShearSNGR`: consider only terms describing the so called *shear noise*.
  - `onlyTimeSNGR`: consider only terms describing the so called *time term*.
  - `SNGRdebug`: print additional information (into files and on screen) for debugging purpose.

- `twoD`: only equations for x- and y-direction are solved, a computational domain varying in i and j is assumed, but ghost points in k are allocated.
- `quadint`: utilise a quadratic interpolation in case of input of external sources on right hand side of the equations given in a set of files (cf. Section 6.7 on page 71).
- `vector`: easy vectorization is supported with long vector lengths.

Each arbitrary combination of the above explained flags is in general possible, except some exclusions: `SmoothWall` and `MirrorWall` as well as the `SNGR` flag family.

If no flag is given a three-dimensional code for sequential execution is generated which is linked with the Tecplot© library `tecio.a`.

Since PIANO determines itself the necessary memory and allocates it on the fly only one main directive is independent of the hard- and software environment: `twoD` which is selected by the configuration to be simulated.

6.3 General Remarks to Preparation of a Run

To start a calculation the following four files described in detail below have to be available:

Input file (optionally specified on command line)	contains control parameters, an explanation is given on pages 57–63
Mean-flow file (mandatorily specified by `FilMean`)	contains mean-flow on CAA grid, specification of the (un)formatted file is given in Section 6.11 on page 77
Grid file (mandatorily specified by `FilGrd`)	contains geometrical data of CAA grid, specification of the (un)formatted file is also given in Section 6.11 on page 77
Logic file (mandatorily specified by `FilLog`)	contains logical data of CAA grid, details of the formatted file are given in Section 6.3.4 on page 65

In special cases one needs additional files:

Indices file	(specified by `FilIJK`)	contains indices for monitoring nodes, details are given in Section 6.9.1 on page 75
Record file	(specified by `FilRec`)	contains last state of calculation
Load-balancing file	(specified by `FilProc`)	specifies manual modifications of the automatic load-balancing, format is given in Section 6.12.2 on page 79
SNGR file	(specified by `FilSNGR`)	contains input data for SNGR
RHS file	(specified by `FilRHS`)	contains source term data for right hand side of the equations

After checking all those files except the input file to be available in the input directory (given by `DirIn`), the programme may be started by running `MakeCall` in the source directory, for example. All output will be written to the output directory (given by `DirOut`), which will be generated if necessary. To reduce interprocessor communication in case of parallel computations each processor writes one file per block for contour plots as well as time histories on discrete nodes and circle points. Thus the output structure is *no* subject to the load-balancing. Additional sub-directories are used to separate the different time levels of the contour plots.

6.3.1 Controlling Code by Keywords

The input file contains the controlling specifications, i. e. parameters, filenames and initial conditions.

First some general remarks concerning the input file (default name is `Piano.in` which may be modified by specifying as the one and only parameter of PIANO the new name on command line):

- keywords are recognized (regardless the case) in any order, but only one per line as long as followed by an appropriate (number of) parameter(s) (care for correct type!);
- everything after `$$` as well as an empty line is ignored as comment;
- blanc space(s) and tab(s) are separators;
- everything in addition to keyword plus necessary parameter(s) will be neglected;
- only the last occurence of several times mentioned keywords is valid;
- boolean parameters are switched off (set `.FALSE.`) by default; they are set `.TRUE.` by simple mentioning the appropriate keyword;
- the scanning of the input file ends at the keyword `End`;
- Since the input file is read after each iteration it is possible to modify some parameters during the run. Only for defined `vector` the reading of the input file is restricted to periods of `Tupdate` (assuming on a supercomputer it would slow down the performance otherwise and it is unusual to need such feature).

  To avoid time consuming inspection of the complete input file in run cases where nothing is modified the repeated reading stops at keyword `QuickEnd`. Due to this feature it is for instance possible to enlarge or to shorten a simulation by modification of `Tend` during run time (CAVE: Due to binary Tecplot format reading error is caused for all history files!).
- most parameters have useful defaults (see `ReadPara.inc`) which will be effective if a parameter is not set; for safety used parameters are reported twice: first time at the beginning of calculation and at the end again (maybe modified by PIANO or the user in the meantime).
- concatenating the parameters of `Dir...` and `Fil...` the necessary `/` in-between will be added automatically on the fly.

What follows is the listing of a typical input file with short descriptions of the parameters, which will be explained in detail afterwards:

```
$$ everything after '$$' as well as empty lines are ignored as comments,
$$ blanc space and tab are separators,
$$ keywords are recognized (regardless the case) in any order
$$ only one per line as long as followed by appropriate number of parameter(s) :)
$$
$$ !!! length of all following lines <= 256 characters !!!
$$ ================================================================
```

```
$$ each parameter up to 'QuickEnd' will be read at least after Tupdate
$$ time steps thus modification of some parameters influence the current run
$$ ------------------------------------------------------------------
$$ Tend 0   activate to enforce immediately regular stop of current run

$$ indicates end of periodical update of input parameters after Tupdate time steps
QuickEnd

$$ root directory for all necessary input files: FilLog, FilGrd, FilMean
$$ and if necessary FilRec (i.e. new set), FilIJK, FilProc, FilSNGR or FilRHS
$$ ------------------------------------------------------------------
DirIn  ./Input

$$ mandatory formatted file of grid logic
$$ ------------------------------------------------------------------
FilLog  logic

$$ mandatory (un)formatted file of grid coordinates (excl. ghost points!)
$$ ------------------------------------------------------------------
FilGrd  GRID

$$ mandatory (un)formatted file of mean flow in same physical domain as in given grid
$$ ------------------------------------------------------------------
FilMean  Flow.5

$$ optional (un)formatted file with input data for Right-Hand-Side terms
$$ ------------------------------------------------------------------
$$ FilRHS  rhs/q_

$$ optional formatted file with input data for SNGR
$$ ------------------------------------------------------------------
$$ FilSNGR  SNGR

$$ optional formatted file with indices for time history of flow values (and vorticity)
$$ ------------------------------------------------------------------
$$ FilIJK  Indices.dat

$$ optional formatted file with CPU to block assignment (one line per CPU)
$$ ------------------------------------------------------------------
$$ FilProc  myBlock2CPU

$$ formatted file with coordinates for time history of flow values (and vorticity),
$$ unfortunately not implemented currently :(
$$ ------------------------------------------------------------------
$$ FilXYZ  Locations.dat

$$ formatted (ASCII Tecplot point format) file with RPM source patch data
$$ ------------------------------------------------------------------
```

```
FilRPM  patch.dat

$$ saves RPM filter coefficient data (ASCII format for -DdebugRPM else binary format)
$$ ------------------------------------------------------------------
FilRPMRec  RPMrecord.bin

$$ mandatory (un)formatted file for relay calculation
$$ ------------------------------------------------------------------
FilRec  Record

$$ directory for all output:
$$ FilRec, FilNois (especially for non-zero Tout), FilHis (for non-zero HistoryOut),
$$ FilRMS (for positive RMSstart) and FilCirc (for non-zero CircNoMic)
$$ ------------------------------------------------------------------
DirOut  ./Output

$$ (TecPlot) file name for field values at by Tout specified time
$$ ------------------------------------------------------------------
FilNois  Contour   ({'.bin','.plt'}/'.dat' will be added depending on sign of Tout)

$$ title string in FilNois overriding default
$$ ------------------------------------------------------------------
$$ FilTitle My personal title

$$ (TecPlot) file name for time history of sound (and vorticity) at specified points
$$ ------------------------------------------------------------------
FilHis  Time   ({'.bin','.plt'}/'.dat' will be added depending on sign of HistoryOut)

$$ (TecPlot) file name for RMS values (sampled within specified interval)
$$ ------------------------------------------------------------------
$$ FilRMS  RMS   ({'.bin','.plt'}/'.dat' will be added depending on sign of Tout)

$$ (TecPlot) file name for circle values for specific acoustic variables
$$ ------------------------------------------------------------------
FilCirc  Circle   ({'.bin','.plt'}/'.dat' will be added depending on sign of CircOut)

Tout  5   store field values after |Tout| steps in FilNois, at least first/last time step
$$ HistoryOut  3   store time history in FilHis after |HistoryOut| steps
VorOut   include vorticity into field values and time history
$$ RPMOut   include stochastic source information into field values and time history (exc

dt  5.D-3   time step size will be compared to global stability limit
Tend  1   number of time steps to be calculated
$$ Tupdate  1000   specifies periodic update of input parameters up to QuickEnd
$$ Tsave  100   record file is written in periods of Tsave time steps

Xref  .0D0  0.D0  0.D0   reference point (as normal as possible to reflecting boundary)
$$ RBD2D   only useful in 3D: just 2D or full 3D boundary conditions?
```

```
$$ kappa  1.4D0   = c_p/c_v = rho/p * (dp/drho)_s
$$ NoRKS  4   number of Runge-Kutta stages (= 4, 5 or 6)
$$ eps  1.D-1   adjusting with which ratio nonlinear terms are taken into account
$$ APE   solve Acoustic Perturbation Equations instead the Linearized Euler Equations

$$ specification of RPM (Random Particle Mesh) parameters:
$$ ------------------------------------------------------------------
RPMdt    1.5d-1  time step of white-noise field
RPMlimit 0.01D0  minimal used length scale
RPMup    0.D0    upstream source window
RPMdown  0.D0    downstream source window
RPMfac   6.D0    scaling factor for patch-data length scale
RPMalfa  1.D0    Langevin coefficient exp(-RPMdt/tscale)
RPMtau   500.D0  time decay constant for ramping source term: 1-exp(-Tstep/tau)

$$ damping  5.D-3   general damping coefficient
$$ WallDamping  2.D2   damping coefficient on slip walls
Filter  6   filter type out of {6,8,100}; 0 means NO filtering
FilterStep  20   filter period, i.e. after FilterStep RK steps filtering is done
$$ NoFilterRun  2   number of filter performances at once
PadeScheme  1   specifies used pade scheme out of {1,2,3} (ONLY valid for Filter=100)
PadeAlpha  .2D0   specifies used pade coefficient (ONLY valid for Filter=100)
$$ PadeVar  4   specifies variable to be filtered (=-1 filters all variables)

$$ specification of local damping spot(s) (general damping has to be set!):
$$ each additional occurence of {Xdamp, MagDamp, RadDamp}
$$ starts a new damping spot definition, last definition is used as default
$$ ------------------------------------------------------------------
$$ Xdamp  0.D0  0.D0  0.D0   centre of local damping spot
$$ MagDamp  .1D0   magnitude of local damping spot
$$ RadDamp  3.D0   half-value radius of local damping spot

$$ RMSstart  100   time step at which sampling for RMS value starts
$$ wavenumber  1.D0   vibration wavenumber of defined/assumed periodic/sponge term
$$ periodic   enforce a periodic pulse on RHS of equations to be solved

$$ specification of sponge layer parameters:
$$ ------------------------------------------------------------------
$$ Sponge 1  60  1.D0 1.D0 4 4   name, depth, sigma, beta, DimX, DimT

$$ specification of sponge bath-tub properties:
$$ ------------------------------------------------------------------
$$ BathTub 0.1D0 5.D0 3.D0   time decay constant, magnitude, half-value radius

$$ input of auxiliary source on RHS (only active for NoSrcFiles > 0):
$$ ------------------------------------------------------------------
SrcPeriod  40.D0   time period of periodical source data set in FilRHS
dtSource  1.D0   time increment between two files with RHS data
```

```
NoSrcFiles  0   number of source data files called FilRHS

$$ initialization with analytic distributions, possibly superposed:
$$ ------------------------------------------------------------------
new   start with given initialization (or restart with recorded state?)
$$ MagP  1.D0   magnitude of pressure pulse at initialization
Xp  -1.D0  0.D0  0.D0   centre of pressure pulse at initialization
RadP  1.D-1   half-value radius of initial pressure pulse

$$ MagS  1.D0   magnitude of entropy spot at initialization
Xs  0.D0  0.D0  0.D0   centre of entropy spot at initialization
RadS  1.D-1   half-value radius of initial entropy spot

$$ MagV  1.D0   magnitude for vorticity of vortex at initialization
Xv  -1.D0  0.D0  0.D0   centre (line) of vortex at initialization
AxisV  0.D0  0.D0  1.D0   orientation for vortex's axis of rotation
RadV  1.D-1   half-value radius of initial vortex

$$ specification of directivity circle(s):
$$ CircVar[01] and CircOut are valid for all circles; each additional occurence of
$$ {CircNoMic, CircNormVec, CircStartVec, CircCentre, CircRadius}
$$ starts a new circle definition, last definition is used as default
$$ ------------------------------------------------------------------
CircVar0  p   first recorded variable out of {rho, u, v, [w,] p}
$$ CircVar1  p   last recorded variable out of {rho, u, v, [w,] p}
CircOut  0   store circle values after CircOut steps in FilCirc
CircNoMic  360   number of microphones refering whole circumference
CircNormVec  0.D0  0.D0  1.D0   normal vector of directivity circle
CircStartVec  1.D0  0.D0  0.D0   start vector of directivity circle
CircCentre  0.D0  0.D0  0.D0   centre of directivity circle
CircRadius  1.D0   radius of directivity circle
$$ ==================================================================
End  indicates end of input, quod libet may follow :)
c------------------------------------------------------------------
c APE           boolean switch to solve Acoustic Perturbation Equations
c               instead the Linearized Euler Equations
c AxisV         axis of rotation of initial vortex
c BathTub       time decay constant, magnitude, half-value radius
c CircCentre    centre of directivity circle
c CircOut       absolute value sets output period for circle history, sign the format
c               < 0: output in ASCII format for TecPlot
c               > 0: output in Piano's native binary format
c CircNoMic     number of microphones refering whole circumference
c CircNormVec   normal vector of directivity circle
c CircRadius    radius of directivity circle
c CircStartVec  direction at which output and naming in positive direction starts,
c               have to be non-collinear to normal vector
c CircVar0      first recorded variable saved in FilCirc
```

```
      c CircVar1      last recorded variable saved in FilCirc
      c damping       general damping coefficient (local value depends on cell size!)
      c DirIn         name of directory containing all input files
      c DirOut        name of directory for all output files
205   c dt            time step size (will be compared to global stability limit)
      c dtSource      time increment between two files with RHS data given in FilRHS
      c End           indicates end of parameter input list
      c eps           ratio with which nonlinear terms are taken into account
      c FilCirc       name of data file for circle values of specific acoustic variables
210   c FilGrd        name of (un)formatted data file with coordinates of used grid
      c FilHis        name of data file for sound (and vorticity) history in specific nodes
      c FilIJK        name of optional formatted index file defining history nodes
      c FilLog        name of mandatory formatted logic file for used grid
      c FilMean       name of mandatory (un)formatted data file for mean flow
215   c FilNois       name of field data file for acoustic variables (+ vorticity)
      c FilProc       name of optional formatted file with CPU to block assignment
      c FilRec        name of mandatory (un)formatted data file for relay calculation
      c FilRHS        name of optional (un)formatted file with input data for RHS terms
      c FilRMS        name of optional (un)formatted file for RMS spatial distribution
220   c FilRPM        name of optional formatted file with RPM source patch data
      c FilRPMRec     name of optional (un)formatted file with  RPM filter coefficients,
      c               ASCII format for -DdebugRPM else binary format
      c FilSNGR       name of optional formatted file with input data for SNGR
      c FilXYZ        name of optional formatted coordinates file defining history locations
225   c Filter        filter type, i.e. order of used filter, out of {0, 6, 8, 100}
      c FilterStep    filter period, i.e. after FilterStep RK steps filtering is done
      c FilTitle      optional title string in FilNois, default depends on directive 'twoD'
      c HistoryOut    absolute value sets output period for history, sign the format
      c               < 0: output in ASCII format for TecPlot
230   c               > 0: output in binary format (for TecPlot)
      c kappa         c_p/c_v, i.e. isentropic exponent of ideal gas
      c MagDamp       magnitude of local damping spot
      c MagP          magnitude of initial pressure pulse
      c MagS          magnitude of initial entropy spot
235   c MagV          magnitude for initial vorticity of vortex
      c new           boolean switch to start from initialization or calculated results
      c               mentioned: new calculation
      c               not mentioned: relay calculation, FilRec have to be in DirIn!
      c NoFilterRun   filter performances at once
240   c NoRKS         number of Runge-Kutta stages (= 4, 5 or 6)
      c NoSrcFiles    number of source data files containing RHS terms called FilRHS,
      c               (dis)allows further settings
      c PadeAlpha     specifies used pade coefficient (ONLY valid for Filter=100)
      c PadeScheme    specifies used pade scheme (ONLY valid for Filter=100)
245   c PadeVar       specifies variable to be filtered (-1: filter all variables)
      c periodic      boolean switch to enforce a periodic pulse on RHS of equations
      c QuickEnd      indicates end of periodical input parameter update after Tupdate steps
      c RadDamp       half-value radius of local damping spot
```

```
      c RadP          half-value radius of initial pressure pulse
250   c RadS          half-value radius of initial entropy spot
      c RadV          half-value radius of initial vortex
      c RBD2D         boolean switch to enforce two-dimensional treatment of boundaries
      c               mentioned: full three-dimensional treatment of boundaries
      c               not mentioned: boundary conditions/initialization restricted to 2D
255   c RMSstart      time step at which sampling for RMS value starts, stop of sampling is
      c               calculated automagically out of given frequency
      c               for RMSstart=0 the start time step is calculated also:
      c               as many periods as possible, starting as late as feasible
      c RPMalfa       Langevin coefficient exp(-RPMdt/tscale)
260   c RPMdown       downstream source window
      c RPMdt         time step of white-noise field
      c RPMfac        scaling factor for patch-data length scale
      c RPMlimit      minimal used length scale
      c RPMtau        time decay constant for ramping source term: 1-exp(-Tstep/tau)
265   c RPMOut        boolean switch for output of RPM source field in FilNois and FilHis
      c               mentioned: output of vertical velocity component(DrpmDx(2)) as Omega3
      c               not mentioned: NO output of RPM source field property
      c RPMup         upstream source window
      c Sponge        user-defined name, depth in nodes, magnitude, exponent, number of space/t
270   c SrcPeriod     time period of periodical source data set given in FilRHS
      c Tend          number of time steps to be calculated during current run
      c               < 0: |Tend| will be made from given/initialized state
      c               > 0: start/continue until in total Tend iterations are completed
      c Tout          absolute value sets output period for field values, sign the format
275   c               < 0: output in ASCII format for TecPlot
      c               > 0: output in binary format (for TecPlot)
      c Tsave         restart files are written in periods of Tsave time steps
      c               < 0: current Tstep is used as suffix to distinguish the saved files
      c               > 0: names are alternating, only up to three states remain in general
280   c Tupdate       update of input parameters up to QuickEnd happens after Tupdate time step
      c               vector or parallel is defined, otherwise update happens after each time s
      c VorOut        boolean switch for output of vorticity in FilNois and FilHis
      c               mentioned: output of vorticity
      c               not mentioned: NO output of vorticity
285   c WallDamping   damping coefficient on slip walls
      c wavenumber    vibration wavenumber of defined/assumed periodic/sponge term
      c Xdamp         centre of local damping spot (general damping has to be set!)
      c Xp            centre coordinates of initial pressure pulse
      c Xref          coordinates of reference point to determine V for Out-/Rad-BC
290   c Xs            centre coordinates of initial entropy spot
      c Xv            centre coordinates of initial vortex
      c-------------------------------------------------------------------


295
```

6.3.2 Controlling Code by Source Parameters

In special cases it is necessary to modify the content of `parameter.h`:

`TECPRECISION` adjusts the precision of Tecplot's binary output: `0` enforces single precision, `1` causes double precision. The file size increases with higher precision, of course.

`TECPLOTDEBUG` set to `1` forces Tecplot© to log all activities; just useful for debugging!

`PIP...` fixes the I/O unit number for different channels, which are subject to the number of blocks and/or circles as well as the operating system due to the way of output in parallel mode (a separate channel for each circle and block is needed). This adjusting will be done by input parameters in the future!

6.3.3 Initial Conditions

The initial conditions may be composed of three types of perturbations:

- (acoustic) pressure pulse,
- entropy spot,
- localised vortex.

All the initial distributions of the variables are based on Gaussian functions, i. e.

$$p'(x_i,0) = p_{\max}\,\exp\left[-\ln 2\,\frac{(x_i-x_i^{\mathrm{c}})^2}{b^2}\right], \tag{6.1}$$

$$s'(x_i,0) = s_{\max}\,\exp\left[-\ln 2\,\frac{(x_i-x_i^{\mathrm{c}})^2}{b^2}\right], \tag{6.2}$$

$$\psi'(x_i,0) = v_{\max}\,b\,\sqrt{\frac{\mathrm{e}}{\ln 4}}\,\exp\left[-\ln 2\,\frac{(x_i-x_i^{\mathrm{c}})^2}{b^2}\right]\vec{\mathrm{e}}_{\psi}\,, \tag{6.3}$$

where

$p_{\max}$	magnitude of (acoustic) pressure pulse, specified by `MagP`
$s_{\max}$	magnitude of entropy spot, specified by `MagS`
$v_{\max}$	magnitude of speed in localised vortex, specified by `MagV`
$\vec{\mathrm{e}}_{\psi}$	vortex's axis of rotation, specified by `AxisV`
x_i^{c}	centre of Gaussian function, specified by `Xp`, `Xs`, `Xv`
b	half-value radius of Gaussian, i. e. $p'(b,0)=p_{\max}/2$, specified by `RadP`, `RadS`, `RadV`

In order to start a calculation with one of the aforementioned initial conditions the boolean `new` has to be set, otherwise the simulation is continued with the state stored in the by `FilRec` given record file. Remind, that all settings will be ignored as long as *no* magnitude `Mag...` is specified. This behaviour could be used to manage different initial conditions very easy.

Any combination of the implemented three basic settings (pressure pulse → `...P`, entropy spot → `...S`, vortex → `...V`) results in a linear superposition.

The initial vortex is defined by $\vec{v}'_{\mathsf{V}} = \nabla \times \vec{\psi}' = \mathrm{rot}(\vec{\psi}')$ to ensure $\nabla \cdot \vec{v}'_{\mathsf{V}} = \mathrm{div}(\vec{v}'_{\mathsf{V}}) = 0$. Explicitly

$$\vec{v}' = v_{\max}\, \vec{e}_\psi \times (\vec{x} - \vec{x}^c) \frac{(\mathrm{e} \ln 4)^{\frac{1}{2}}}{b}\ \exp\left[- \ln 2\ \frac{(\vec{x} - \vec{x}^c)^2}{b^2} \right] \tag{6.4}$$

is being set as initial condition. Hence for maximum rotational speed $\vec{v}'_{\mathsf{V}}(|\vec{r}| = \frac{b}{\sqrt{\ln 4}}) = v_{\max}\vec{e}_\psi \times \vec{e}_r$ with $\vec{r} = \vec{x} - \vec{x}^{\mathrm{c}} = r\vec{e}_r$ applies.

For a nonlinear calculation density and pressure have to be initialised, too:

$$p'(x_i, 0) = \varrho'(x_i, 0) = -v_{\max}^2\, \frac{\mathrm{e}}{2}\, \varepsilon\ \exp\left[- \ln 4\ \frac{(x_i - x_i^{\mathrm{c}})^2}{b^2} \right]\ . \tag{6.5}$$

6.3.4 Grid Logic

For easy handling the same logic as in FLOWer [BBE+00] is used.

As a consequence Logic [Zie96] can be employed to generate the logic file `FilLog`.

There are a few differences:

- The logic refers to the node numbering of FLOWer: physical node numbering starts with `2` instead of `0` (as used in PIANO, internally and in indices file).
  Tecplot© starts the numbering of physical nodes with `1` ...
- As long as one keeps the definitions of `parameter.h` one has to adapt the boundary flags already defined by Logic to PIANO's needs. Additionally one has to complete Logic's work if the right boundary condition could not be detected.
- Due to node oriented management of boundary nodes (in contrast to volume oriented treatment in FLOWer) some manual 'fine tuning' is necessary if a block face is segmented: For unmodified logic the nodes belonging to two adjacent segments would be treated twice, although only one physical boundary condition is realistic. Thus the start index (end index) of one of the segments has to be increased (decreased) by 1. In any case one has to keep inner cuts untouched and must modify any other neighbouring segment instead. Otherwise one would change indirectly the CAA grid, because PIANO uses the logic for the coordinates also. Due to twin occurence of the inner cut segments this method is less error prone, too.

Singular Nodes

Since PIANO is only able to exchange data on inner cuts with one neighbouring block it is useful to specify so-called *singular nodes* in case of multiple block (> 2) junctions to overcome this imperfection due to missing topological information. By addressing all adjacing blocks to one physical location PIANO will treat these nodes in a special way: The exact arithmetic mean value is calculated and propagated to all computational storages.

The format of such specification using PIANO's logic (in contrast to the rest of the file) at the very end of the logic file `FilLog` is explained in the following example:

```
$$ BlockNo  i  j
$$
$$ right up
1  0 0
$$ right down
2  30 0
$$ left up
3  40 0
$$ left down
4  0 0 $$ that's it
```

Everything beyond `$$` and empty lines are skipped as a comment. Each valid index line names the block number and the corresponding indices (anything afterwards will be skipped!). A new singular node is introduced by more than 1 comment line.

In contrast to the above two-dimensional listing a three-dimensional specification may also look like the following example:

```
$$ BlockNo  i  j  k   i  j  k
$$
$$ top right
1  0 30 30  10 30 30
$$ bottom right
2  0 30 0  10 30 0
$$ bottom left
3  0 10 40  0 0 40
$$ top left
4  30 0 0  30 10 0 $$ that's it
```

For a single singular node the one and only difference is the third index. But for a singular line the input is simplified: Instead of specifying each singular node separately it is possible to give the start and end index. Attention must be paid to the different directions and indices; fortunately PIANO will check the input ...!

6.3.5 Boundary Conditions

As mentioned in Chapter 4 currently six different types of boundary conditions are implemented. Note, that the flags for these boundary conditions in the logic file are not the same as in FLOWer, but this naming may be modified redefining (arbitrarily, but consistently) the parameters `...BC` in `parameter.h`.

In detail, following boundary conditions are available:

inner cut condition	:=	-1,
slip wall condition	:=	10,
adiabatic slip wall condition	:=	11,
outflow condition	:=	20,
radiation condition	:=	21,
sponge layer condition	:=	22.

The boundary flags have to be adjusted in the logic file `FilLog` by hand. The former naming is still accepted (and corrected internally), but a warning is issued.

For quasi two-dimensional calculations using a three-dimensional grid (boolean `RBD2D` set) the

lateral boundary conditions (on faces 5 and 6) have to be 'slip wall'.

One should be aware of the solution's sensitivity with respect to the location of the reference point given by `Xref`: In order to fulfill the ray concept, on which TAM's boundary conditions base, the reference point has to be located as close as possible to the (assumed) origin of the perturbations leaving the computational domain. Although this is no easy job for several noise sources, one should try to specify a reference point which encloses at least the same angle with the most important boundary as the supposed noise origin does.
Setting the boolean `RBD2D` causes a reference *line* (varying z) to be used.

6.4 Damping and Filtering

In addition to the global damping coefficient ν_{ASD} (specified by `damping`, see Section 3.3 on page 38) it is possible to increase the damping factor just on slip walls. The coefficient given by `WallDamping` is used only on the slip wall boundaries for the right hand side of (2.24) on page 17 ($\ldots = -\texttt{WallDamping} \cdot \phi$). Sometimes massive damping on the slip walls (values with 3 digits) is necessary to keep the solution stable.

For some configurations it might be useful to define a local damping spot: Once again the Gaussian function is employed to describe a useful shape with the local magnitude `MagDamp` and the half-value radius `RadDamp` centered at `Xdamp`:

$$\nu_{\mathrm{spot}}(x_i) = \texttt{MagDamp}\ \exp\left[-\ln 2\ \frac{(x_i - \texttt{Xdamp}_i)^2}{\texttt{RadDamp}^2}\right] \tag{6.6}$$

Each additional occurence of `Xdamp`, `MagDamp` or `RadDamp` starts a new damping spot definition (the previous definition defines the current initial setting). For example, in order to define some spots with the same shape just the location has to be redefined from the second one on!

One has to keep in mind, that the actual local damping is subject to the local cell size, i. e. for a given damping coefficient the coarser the mesh the lower the local damping, and requires the global damping coefficient `damping` to be set.

To get rid of the pressure pulse caused by initialization of a localised vortex a *sponge bath-tub* might be defined: With the keyword `BathTub` the three parameters τ, $\hat{\sigma}$ and b_{SBT} are specified in this order and complete the definition given in Section 3.2 on page 40. τ influences the temporal decay of the damping: the higher τ, the later the damping ends (after $t = 5\tau$ the exponential decay will be terminated completely!). Further temporal decay functions are under way. $b_{\mathrm{SBT}} = 3b_{\mathrm{v}}$ is a good first guess. The calculation of the convective velocity will be improved also in a coming release.

The order of the filter (valid values are 6 and 8) is at the same time the appropriate value for `Filter`, except for the Padé filter (`Filter` = 100); for details see 3.4 on page 40. If no value for `Filter` is given or `Filter` is set to 0, no filtering will be applied. The filter time interval is specified by `FilterStep`. By `NoFilterRun` the number of performed filter runs at once (one after another) may be defined.

For the Padé filter some more coefficients have to be set: `PadeScheme` out of the set {1, 2, 3} selects the employed scheme, `PadeAlpha` adjusts the accompanying coefficient and `PadeVar` out of {1, 2, 3, 4[, 5]} designates the variable to be filtered (a value of −1 causes a filtering of all variables!).

6.5 Periodic Sources

To enforce a harmonically oscillating source on the right hand side of the equations to be solved a source term may be introduced by mentioning the boolean `periodic`. The spatial shape of the source distribution is specified with the same parameters as an initial pressure pulse, the wave number is specified by `wavenumber`. Finally the right hand side source reads $q_p(x_i, t) = p(x_i)\ \cos(t \cdot \texttt{wavenumber})$.

6.6 Employment of the RPM Model

As described in Section 6.2 on page 54 the stochastic sound sources from RPM are activated in PIANO via compiler directives, but some additional keywords have to be used, too.

`FilRPM` selects the name of the formatted file with RPM source patch data.

`FilRPMRec` selects the name of the RPM filter coefficient data file.

`RPMalfa` defines the Langevin coefficient $\exp(-\texttt{RPMdt}/\texttt{tscale})$.

`RPMdown` specifies the lower fade in intervall in percent of the patch length.

`RPMdt` determines the time increment after a turbulent particle enters or leaves the patch area.

`RPMfac` is the scaling factor of the length scale.

`RPMlimit` is the lower limit for the length scale.

`RPMtau` determines the exponentially fading in of the RPM sources: for $t = 70\%$ of `RPMtau` the source magnitude has reached more than half of the final value, for $t > 4.6 \cdot$ `RPMtau` the final magnitude is achieved approximately. This method is just used to suppress the initial pressure pulse.

`RPMup` specifies the upper fade in intervall in percent of the patch length.

For a set preprocessor directive `debugRPM` the used random seeding will be saved, which will be read in if available! Thus it is possible, to reproduce even the statistical varying model identically.

6.6.1 Input Data Set

The particle mesh necessary for a RPM run has to be provided in an additional input file located in the PIANO input directory (specified by `FilRPM`, default is set to `patch.dat`).

The input file `FilRPM` has Tecplot© ASCII format (field data in ASCII point format) and defines a structured orthogonal single-block grid that resolves a patch of turbulent sound sources in the interior CAA domain. In two-dimensional cases the grid lines η = const. are defined by sequentially numbering a bundle of N streamtraces, i. e., each streamtrace has a running index $j = 1 \ldots N$. The streamtraces define particle paths through the steady RANS mean-flow field. In the left hand side of Figure 6.1 on the facing page some streamtraces in the slat-cove of a two-element high-lift airfoil are depicted. For an existing bundle of η = const. lines a set of orthogonal ξ = const. grid-lines can be generated with an orthogonal grid algorithm (cf. right

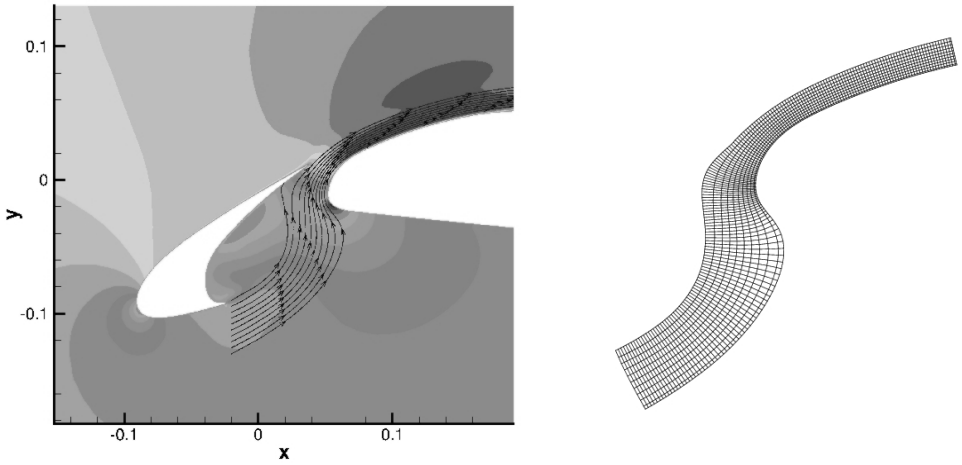


Figure 6.1: Streamtraces in the slat-cove and auxiliary grid; η = const. lines are identical with the streamtraces (particle paths in the steady RANS solution), ξ = const. lines are normal to the streamtraces [McN72, Fle97]

hand side of Figure 6.1). For this, a certain number of discrete points $i = 1 \ldots M$ is equidistantly distributed along either the smallest or the largest η = const. line, respectively. Next, starting from point i, the related ξ = const. line is constructed by marching locally orthogonal to the η = const. lines through the grid.

The variables provided in the collocation points of the auxiliary patch grid are:

`X`:	first spatial direction,
`Z`:	second spatial direction,
`x_velocity`:	velocity in first spatial direction,
`z_velocity`:	velocity in second spatial direction,
`turb_kinetic_energy`:	turbulent kinetic energy k,
`turb_omega`:	turbulent rate of dissipation ω,
`l`:	turbulent length scale $l = \sqrt{k}/\omega$,
`v_tangential`:	modul of velocity on the streamtrace,
`s_strmtrce`:	arc length along the streamtrace,
`t_strmtrce`:	convection time, first upstream points equals 0,
`xix`:	metric coefficient $\xi_x = y_\eta/J$,
`etx`:	metric coefficient $\eta_x = -y_\xi/J$,
`xiy`:	metric coefficient $\xi_y = -x_\eta/J$,
`ety`:	metric coefficient $\eta_y = x_\xi/J$, $J = x_\xi y_\eta - x_\eta y_\xi$.

Generation of a New Patch File

To generate a patch file `FilRPM` a two-step procedure is proposed. First, a streamtrace bundle is generated with the help of Tecplot©:

- Load the RANS mean-flow solution (including the turbulence model variables) into Tecplot©.
- With the *Streamtrace Details Dialog* or the *Add Streamtrace Tool* place an appropriate

rake position from which the streamtrace will originate (e. g. by defining the *Rake Start* and *Rake End* positions on the *position page*).

- Define an appropriate number of *Streams per Rake* (this number will later define N in the turbulence patch grid).
- Place the streamtraces.
- If these streamtraces do not cover the desired source region, delete all or last streamtrace(s) and specify an adjusted rake position.
- Extract streamtraces through the menue option *Data/Extract/Streamtraces/Extract*.
- Calculate the turbulent length scale, i. e. $l = \sqrt{k}/\omega$ for a k-ω-RANS calculation or $l = \frac{k^{3/2}}{\epsilon}$. Keep in mind the dimensionless quantities of PIANO!
- Export the final streamtraces using the option *File/Write Data File*:
  - Select the N streamtraces to be written.
  - Select the variables to be written (2D problem with X- and Z-coordinates):

`X`:	first spatial direction,
`Z`:	second spatial direction,
`x_velocity`:	velocity in first spatial direction,
`z_velocity`:	velocity in second spatial direction,
`turb_kinetic_energy`:	turbulent kinetic energy k,
`turb_omega`:	turbulent rate of dissipation ω,
`l`:	turbulent length scale $l = \sqrt{k}/\omega$.

  - Select the format options: *ASCII* and *point format*.
  - Select an appropriate file name.
  - Write data file *streamtrace data*.

In a second step employ the utility programme `RANDOM_PATCH` to generate from the raw Tecplot© streamtrace output the desired source patch grid `FilRPM`. The programme reads the previously generated raw streamtrace data output from Tecplot©. This programme reads in an optional input file freely named (default: `Random_Patch.lst`), with all parameters, like the input streamtrace file `cfpatch` and the output patch file `cfout2`. The programme generates a patch grid with an advancing front algorithm starting from the first or last streamline. This choice is done by the parameter `istrc`, which takes on the values `0` or `1`, respectively. Furthermore, the length of the generated source patch, starting from the initial upstream rake position, is defined by the parameter `lpatch`. If `lpatch` is greater than the actual length of the streamtraces, the programme will terminate with an error message. The number of points along the reduced streamtrace are set by the parameter `imax`. In some cases it might be helpful not to start at the first point of the streamtrace rather further downstream, therefore the parameter `ibegin` can be adopted.

6.6.2 Restart the RPM Model

Of course, also for a restart using the RPM model the source patch data file `FilRPM` is needed. Fortunately some additional information, generated the first time by PIANO, could be read in,

if the RPM filter coefficient data file `FilRPMRec` is found. The format of this file depends on the preprocessor directive `debugRPM`: if specified `FilRPMRec` will be written and read formatted, if not the unformatted version will be written and read!

6.7 Input of Auxiliary Sources

Another method to establish a source term on the right hand side of the employed equations is the external input: `|NoSrcFiles|` (default is 0) specifies the number of available files, whose format has to be the same as for a mean-flow file `FilMean`. The name consists of the base `FilRHS` and a trailing numbering (8 digits, possibly with leading zeros, starting with 1). The read in data is reported immediately in a Tecplot© file, unless `NoSrcFiles` is negative. The output files (format is controlled by sign of `Tout`) are placed in a separate sub-directory `RHS`, the file name is extended by the file number; all analogously to the output of `FilNois`. `SrcPeriod` adjusts the time period of the periodical source data set, `|dtSource|` gives the time increment between two `FilRHS` files. To avoid redundancy $\texttt{|NoSrcFiles|} \cdot \texttt{|dtSource|} \approx \texttt{SrcPeriod}$ is assumed.

If $\texttt{dtSource} < 0$ applies, the file data will be regarded as the turbulent velocity components, thus the number of variables reduces from 4 to 2 (two-dimensional) or 5 to 3 (three-dimensional). For $\texttt{NoSrcFiles} = 0$ all settings concerning input of source terms will be ignored.

Source Term Interpolation

Depending on the original sampling time step of the instantaneous source term data some intermediate states have to be interpolated for all Runge-Kutta sub-steps. By default this is done by a linear interpolation, which closes the `SrcPeriod` appropriately, but the accuracy can be increased to second order by the preprocessor directive `quadint`.

Then PIANO uses the Stirling interpolation formula [BS79], which reads for the quadratic interpolation with $t_l - \Delta t/2 \leq t < t_l + \Delta t/2$

$$\begin{aligned} \vec{U}_{\text{rhs}}(t) &= \vec{U}^l_{\text{rhs}} \left[1 - \left(\frac{t - t_l}{\Delta t} \right)^2 \right] \\ &\quad + \frac{1}{2} \vec{U}^{l+1}_{\text{rhs}} \left(\frac{t - t_l}{\Delta t} \right) \left[1 + \left(\frac{t - t_l}{\Delta t} \right) \right] \\ &\quad - \frac{1}{2} \vec{U}^{l-1}_{\text{rhs}} \left(\frac{t - t_l}{\Delta t} \right) \left[1 - \left(\frac{t - t_l}{\Delta t} \right) \right]. \end{aligned} \tag{6.7}$$

The quadratic interpolation allows a larger sampling time step for the input data at the same order of accuracy and, therefore, a smaller amount of disc space memory. However, the quadratic interpolation reduces the computation speed.

Another method for storage and input reduction is a restricted input: Only the bounding box with non-zero source term values has to be read in. Unfortunately this way is currently only implemented in a developer version of PIANO and will be integrated soon.

6.8 Employment of a Sponge Layer

As usual the typical segment definition (block, face, start and end indices) has to be given using the boundary condition type *sponge layer* (the current reference number is fixed by `SpongeBC` in `parameter.h`). The respective sponge layer type is specified by the seventh integer (used figure is arbitrary, default is `0`). One has to remind the fact, that the respective number of segments per face (given at the beginning of each block information) has to be increased also, because a sponge layer should be used additionally to another boundary condition!

All (maybe multiple times) employed sponge layer types are defined in detail in the input file of PIANO (e. g. `Piano.in`): After the keyword `Sponge`

- the sponge layer type ***name***,
- the sponge layer depth ***n***,
- the magnitude of the fading function ***sigma***,
- the exponent of the fading function ***beta***,
- the number ***dimX*** of variables per sponge layer node varying in space, but constant in time and vice versa
- the number ***dimT*** of variables per sponge layer node varying in time, but constant in space

have to be specified in this order. The predefined fading function

$$\sigma(\xi) = \mathit{sigma}\left[\frac{1-\cos(\pi\xi)}{2}\right]^{\mathit{beta}} \tag{6.8}$$

is subject to ξ varying normal to the respective boundary. $\xi_{\text{wall}} = 1$ and $\xi_{\text{n}} = 0$, thus $\sigma_{\text{wall}} =$ ***sigma*** and $\sigma_{\text{n}} = 0$ applies. Sometimes only big values of ***sigma*** (values with 3 digits) are able to ensure the forcing function in the given sponge layer!

The default sponge layer type defines a radiation boundary condition by

- ***name*** `= 0`,
- ***n*** `= SpongeDepth`,
- ***sigma*** `= SpongeSigma`,
- ***beta*** `= SpongeBeta`,
- ***dimX*** `= SpongeDimX`,
- ***dimT*** `= SpongeDimT`,

where the constants `SpongeDepth`, `SpongeSigma`, `SpongeBeta` are fixed in `parameter.h`.

The final sponge layer term $\sigma(\xi)$(`Up`$-$`Uref`), where `Up` are the perturbations and `Uref` the forcing function terms, is subtracted from the right hand side of the equations (2.24) on page 17 to be solved.

In order to use the approach implemented the forcing function definition has to be prepared appropriately by simple mathematics sometimes. Currently, sponge layer typ 1 defines as forcing function the annular duct mode $(0, 1)$ with specific values of a benchmark (ensure `dimT = 4` and `dimX = NoVar`); typ 2 generates sound waves of a monopole located at a specific source position beyond the boundaries, i. e. nearly plane waves enter the domain (ensure `dimT = 3` and `dimX = 4`).

For the implementation of new forcing functions the already existing ones may be used as templates.

The following list explains in detail file `Sponge.f` containing the three small subroutines identically structured, i. e. controlled by the user-defined sponge layer type `name` (also accessed as `typ%name`):

`CalcX` calculates, while the preparation is done by `PrepBound` once per run, all user-defined functions subject to the coordinates x, y and z for all sponge layer nodes, e. g.

$$X_1 = \cos\left(k_{01}^{+} R_2 x\right) \ ,$$

$$X_2 = \sin\left(k_{01}^{+} R_2 x\right) \ ,$$

$$X_3 = A\left[J_0\left(\sigma\sqrt{y^2+z^2}\right) - \frac{J_1(\sigma)}{Y_1(\sigma)}\, Y_0\left(\sigma\sqrt{y^2+z^2}\right)\right] \ ,$$

$$X_4 = A\left[J_1\left(\sigma\sqrt{y^2+z^2}\right) - \frac{J_1(\sigma)}{Y_1(\sigma)}\, Y_1\left(\sigma\sqrt{y^2+z^2}\right)\right] \frac{\varrho_2 c_2}{\varrho_j c_j} \frac{\sigma}{k_0 R_2\left(1 - M_j k_{01}^{+}/k_0\right)} \frac{y}{\sqrt{y^2+z^2}} \ ,$$

$$X_5 = A\left[J_1\left(\sigma\sqrt{y^2+z^2}\right) - \frac{J_1(\sigma)}{Y_1(\sigma)}\, Y_1\left(\sigma\sqrt{y^2+z^2}\right)\right] \frac{\varrho_2 c_2}{\varrho_j c_j} \frac{\sigma}{k_0 R_2\left(1 - M_j k_{01}^{+}/k_0\right)} \frac{z}{\sqrt{y^2+z^2}} \ .$$

`CalcT` initializes the constant factors accessible for all sponge layers of the same sponge layer type (controlled by the boolean string `inTimeLoop` set to `FALSE`) once at the beginning, e. g. $T_3 = \dfrac{\varrho_2 c_2}{\varrho_j c_j} \dfrac{k_{01}^{+}/k_0}{1 - M_j\, k_{01}^{+}/k_0}$ and $T_4 = \dfrac{c_2^2}{c_j^2}$.

During all Runge-Kutta sub-steps it is called once per sponge layer segment by subroutine `CalcSponge` with `inTimeLoop` set to `TRUE` and some other appropriate parameters. In this case it calculates all time dependent functions, e. g.

$$T_1 = \cos\left(\omega \frac{R_2}{c_2} t\right) \ ,$$

$$T_2 = \sin\left(\omega \frac{R_2}{c_2} t\right) \ .$$

`CalcUref` calculates the user-defined forcing function `Uref` subject to time `t` and space `X` (i. e. `typ%T(dimT)` and `X(dimX)`); for example

$$u_{\mathrm{ref}} = T_3 X_3 (X_1 T_1 + X_2 T_2) \ ,$$

$$v_{\mathrm{ref}} = X_4 (X_1 T_2 - X_2 T_1) \ ,$$

$$w_{\mathrm{ref}} = X_5 (X_1 T_2 - X_2 T_1) \ ,$$

$$p_{\mathrm{ref}} = X_3 (X_1 T_1 + X_2 T_2) \ ,$$

$$\varrho_{\mathrm{ref}} = T_4\, p_{\mathrm{ref}} \ .$$

This subroutine is called for all sponge layer nodes during all Runge-Kutta substeps by subroutine `CalcSponge` with the appropriate parameters. A modification of the array(s) `X` is possible, but unusual (contradicts the splitting into functions subject to time and space!).

6.9 Output Files of the Simulation

As a result PIANO generates several groups of files (listed below) containing all data of the complete physical domain, time histories on designated discrete nodes or points on user-defined circles in one of the following formats: Tecplot's ASCII (`ext` = `dat`) or binary format (latter one only if `tecio.a` is linked, i. e. `-UnoTecplotLib` is specified; `ext` = `plt`) as well as PIANO's native binary format (enables best performance without Tecplot's library, on parallel computers e. g.; `ext` = `bin`). Unfortunately Tecplot's library supports only up to 10 files simultaneously, thus the circle data cannot be written in Tecplot© binary format! The format is selected by the sign of the output interval, `Tout`, `HistoryOut`, `CircOut`, respectively.

At least the first three files will be generated:

*FilNois*_d8(n·`Tout`)`[.cont].`*ext*	contains the current perturbation field
FilNois`_mean[.cont].`*ext*	contains the used mean-flow field
FilRec`[.cont]`	record file after last time step
*FilRec*d8(n·`Tsave`)	record file after n·`Tsave` time steps for `Tsave` < 0
FilRec`[01]`	record file after n·`Tsave` time steps for `Tsave` > 0
FilRMS.ext	contour plot of root mean squared perturbation variables

For `HistoryOut`$\neq 0$ and a non-trivial content of `FilIJK` one will get the following files:

*FilHis*d8(`Tend`)`_R.`*ext*	time plot of density ϱ' in monitoring-nodes
*FilHis*d8(`Tend`)`_U.`*ext*	time plot of u'-velocity component in monitoring-nodes
*FilHis*d8(`Tend`)`_V.`*ext*	time plot of v'-velocity component in monitoring-nodes
*FilHis*d8(`Tend`)`_W.`*ext*	time plot of w'-velocity component in monitoring-nodes
*FilHis*d8(`Tend`)`_P.`*ext*	time plot of pressure p' in monitoring-nodes
*FilHis*d8(`Tend`)`_W1.`*ext*	time plot of Ω_1'-vorticity in monitoring-nodes
*FilHis*d8(`Tend`)`_W2.`*ext*	time plot of Ω_2'-vorticity in monitoring-nodes
*FilHis*d8(`Tend`)`_W3.`*ext*	time plot of Ω_3'-vorticity in monitoring-nodes

For `CircOut`$\neq 0$ and consistently set further parameters following files will be produced:

FilCirc`_XYZ.`*ext*	coordinates and ordinal number of circle points
*FilCirc*d2(*Circ*)*_CircVar0.ext*	time plot of perturbation variable *CircVar0* on circle points
⋮	⋮
*FilCirc*d2(*Circ*)*_CircVar1.ext*	time plot of perturbation variable *CircVar1* on circle points

d[238](`T`) is an integer function which expands `T` to 2, 3 or 8 digits by leading zeros. `.cont` is sometimes added to prevent overwriting of existing files in case of relay calculation.

By `FilTitle` the user-defined title for all `FilNois` files might be given, otherwise the default depending on directive `twoD` will be used.

*FilNois*_d8(n·`Tout`) contains the variables (x, y, [z,]) ϱ', u', v', [w',] p' (, [Ω_1', Ω_2',] Ω_3') after $n \cdot$ `Tout` time steps, where n is a non-negative integer. Even without an specified useful value for `Tout` at least the initial and last state will be saved. The vorticity $\vec{\Omega}$ will be calculated and saved only if the boolean `VorOut` is set.

In order to structure the sometimes numerous files in case of output in PIANO's native format all files of a specific state will be put in a newly made sub-directory. Nevertheless d3(*Blocknumber*) will be the distinguishing prefix for the usual file name.

For one variable per file each *FilHis* d8(Tend)_?.*ext* contains time t and corresponding variable, named according to the specified locations (generated out of the indices and block number) to watch the recorded signal at each virtual microphone.
Keep in mind that Tecplot© starts numbering with '1' not '0', but generated variables names obey PIANO's logic conventions . . . !

6.9.1 Specification of Virtual Data Sensors

If in file FilIJK nodes are specified one gets continous signals stored in time history files named FilHis containing the variable values in these discrete nodes. The output is controlled by HistoryOut: The absolute value specifies the sampling rate, i. e. the number of time steps after which the recording of the specified variables happens, the sign determines the output format. If Tecplot's routines are available (PIANO compiled with -UnoTecplotLib) a positive HistoryOut leads to binary Tecplot© files, a negative one produces an ASCII file readable by Tecplot©. Otherwise a positive HistoryOut enforces binary files in PIANO's format, a negative one generates an ASCII file readable by Tecplot©, too.
By setting the boolean VorOut it is possible to get the time history for the vorticity value(s) $[\Omega'_1, \Omega'_2,] \Omega'_3$, too. Otherwise only the values of the variables $(\varrho', \vec{v}', p')$ are written into the time history files. In both modes (parallel or sequential) each processor writes its own file per variable.

Following items are important for the syntax, which can be seen in the listed example file given as FilIJK on the current page.

- first and third line are read over, thus can be used for comments;
- the integer on second line specifies the number of sensor locations to be read (regardless the actual number of given indices . . . !);
- starting on fourth line the topologic locations are given line by line: first integer specifies the block number, following two or three (depending on -DtwoD) integers are interpreted as appropriate indices (will be checked for safety reasons) conforming to PIANO logic;
- the order of the locations is arbitrary and will be sorted by PIANO for output.

```
total number of nodes (i.e. indices)
4
[23]D: BlockNo  i  j  [k]
     1        39  00
     1        40  00
     2        30  00
     2        31  00
```

6.9.2 Circle(s) for Recording of Directivity

In preparation of determining the noise source's directivity an arbitrary number of user-defined circles may be specified: CircNoMic selects the number of microphones equally spaced along the

whole circumference. `CircCentre` fixes the centre of the directivity circle, `CircRadius` sets the respective radius. The vector `CircNormVec` normal to the plane spanned by the circle and the direction `CircStartVec` at which the output and naming in positive direction starts complete the definition (both vectors have to be non-collinear!). Each additional occurence of `CircNoMic`, `CircCentre`, `CircRadius`, `CircNormVec` or `CircStartVec` starts a new circle definition; the last circle characteristics will be used as initial definition for the now introduced one.

The absolute value of `CircOut` specifies the output interval, i. e. the sampling rate, the sign the format: negative values cause an ASCII file readable by Tecplot©, positive ones enforce binary files in PIANO's format. Also valid for all circles is the selection of the first (by `CircVar0`) and last (by `CircVar1`) variable to be recorded by `CircOut`: One element of the set {`rho`, `u`, `v`, [`w`,] `p`} is expected.
Setting `CircOut` = 0 or misordering `CircVar?` is like cancelling all circle definitions at once, no matter what is defined additionally!

6.9.3 Output of Contour Plot with RMS Distribution

Especially for computations with periodic sources it is helpful to get immediately the contour plot of the RMS distribution in Tecplot's format. This might be produced simultaneously with a PIANO run by an appropriate setting: Taking the given `wavenumber` into account the sampling starts after time step `RMSstart` and stops at the very far end, i. e. after the last completed period of the oscillation before the simulation ends. If `RMSstart` = 0 applies, the start time step is calculated as well: As many complete periods as possible are used for the calculation of the RMS distribution, but the sampling starts as late as feasible, i. e. the sampling is shifted to the end of the computation.

6.10 Remaining Parameters

`Tend` determines the number of time steps to be calculated during the current run: positive values are interpreted as the absolute number of iterations to be made, negative values will be added to the already made number to get the given state. Thus it is possible to specify the number of time steps of one job in a relay calculation.

To enforce intermediate saving of the reached state the parameter `Tsave` may be used: For negative values a usual record file will be saved in periods of `Tsave`, the number of time steps accomplished will be used for the file name suffix. Positive values cause an alternating naming, hence only two intermediate and the final record file will remain in general!

`dt` specifies the time step size which will be compared to the global stability limit:

$$\mathtt{t}_{\text{limit}} = \frac{2.83 l_{\min}}{\pi(1+Ma)} \text{ with } l_{\min} = \min\left(\left|\frac{\partial\vec{x}}{\partial\xi_m}\right| = \sqrt{\left(\frac{\partial\vec{x}}{\partial\xi_m}\right)^2}\right)_{i,j,k,m} . \tag{6.9}$$

For values of `dt` too large for the used grid will be rejected!

`kappa` adjusts the isentropic exponent of ideal gas $c_{\text{p}}/c_{\text{v}} = \varrho/p \cdot (\mathrm{d}\, p\varrho)_s$ (= 1.4 for an ideal gas, e. g. normal air).

`NoRKS` selects out of {4, 5, 6} the number of Runge-Kutta stages to be made for a physical time step. For `NoRKS` = 5 the number of stages alters from 5 to 6 and back, i. e. the well-known

*l*ow-*d*issipation, low-*d*ispersion *R*unge-*K*utta (LDDRK) algorithm (cf. Section 3.2 on page 37) is performed.

With `eps` the ratio ε of fluctuation to mean value, see (2.24) on page 17, is adjusted. If no value is given or `eps` $= 0$ applies, the *L*inear *E*uler *E*quations (LEE) will be solved, unless the boolean `APE` is mentioned: Then the *A*coustic *P*erturbation *E*quations (APE) (2.7), (2.8) on page 11 will be solved (numerical specifics are given in Section 2.2.6 on page 16).

6.11 Format of Grid, Mean-flow, Record and Output Files

Although generally some helpful tools (e. g. interpol, PreFlow, MegaCADs, PreGrid) generate these (binary) files, here are given the formats, because data can be given in ASCII format also. The letter `D` as separator of mantissa and exponent in ASCII format has to be replaced by letter `E` for correct input.

The different formats (big-/little-endian, ASCII) are distinguished automatically as long as all ASCII files start with `$` (may be used anywhere to insert comments as well).

Grid files have the POPINDA format used by FLOWer:

```
NumberOfBlocks, TmpInteger1, TmpInteger2
DO block = 1, NumberOfBlocks
   Imax, Jmax, Kmax, TmpInteger3
   (((x(i,j,k,1), x(i,j,k,2), x(i,j,k,3),
      i = 1, Imax),
      j = 1, Jmax),
      k = 1, Kmax)
END DO
```

`TmpInteger[123]` have no function in PIANO (as well as in FLOWer?), and are of `INTEGER` type, as well as `NumberOfBlocks`, `[IJK]max`. `x` are of `REAL` type.

Hence a grid generated by MegaCADs and exported in FLOWer format may be used. For two-dimensional grids `Kmax = 1` and `x(i,j,k,3) =` const. (will only be used for output) applies.

Mean-flow files are formatted analogously:

```
NumberOfBlocks
DO block = 1, NumberOfBlocks
   Imax, Jmax, Kmax
   ((((u0(i,j,k,var),
      i = 1, Imax),
      j = 1, Jmax),
      k = 1, Kmax),
      var = 1, NumberOfVariables)
END DO
```

`NumberOfBlocks` and `[IJK]max` are of `INTEGER` type, `u0` are of `DOUBLE PRECISION` type.
For two-dimensional calculations `NumberOfVariables = 4`, i. e. ϱ_0, u_0, v_0, p_0 applies.

Record files are formatted similarly to mean-flow files: After some additional information

given in 5 lines in the header the values of perturbation variables are saved in the same way for sequential and parallel mode.

```
NumberOfBlocks, nextTstep, NoRKS, NoFilterRun, Filter, FilterStep,
   PadeVar, PadeScheme, PadeAlpha
damping, WallDamping, eps, dt, t, (Xref(box), box = 1, DimCD), RBD2D
SizeOfRndseed, Rndseed
NoFilLog, FilLog
NoFilGrd, FilGrd
NoFilMean, FilMean
DO block = 1, NumberOfBlocks
   Imax, Jmax, Kmax
   ((((up(i,j,k,var),
      i = 1, Imax),
      j = 1, Jmax),
      k = 1, Kmax),
      var = 1, NumberOfVariables)
END DO
```

`nextTstep`, `NoRKS`, `NoFilterRun`, `Filter`, `FilterStep`, `PadeVar`, `PadeScheme`, `SizeOfRndseed` (terms the length of the following array `Rndseed`) and `Rndseed` as well as `NumberOfBlocks`, `NoFilLog`, `NoFilGrd`, `NoFilMean` (`NoFil...` denotes the length of the following string) and `[IJK]max` are of `INTEGER` type; `PadeAlpha`, `damping`, `WallDamping`, `eps`, `dt`, `t`, `Xref`, `RBD2D` and `up` are of `DOUBLE PRECISION` type; `FilLog`, `FilGrd` as well as `FilMean` are strings of `CHARACTER` type.
For two-dimensional calculations `NumberOfVariables = 4`, i. e. ϱ', u', v', p' applies again. Of course, `SizeOfRndseed` and `Rndseed` are written and read for RPM calculations only!

Output files of field values in PIANO's native format (also similar to mean-flow files) look like

```
FilTitle
BlockNumber
Imax, Jmax, Kmax
((((up(i,j,k,var), i = 1, Imax),
                   j = 1, Jmax),
                   k = 1, Kmax), var = 1, NumberOfVariables)
```

with the given `FilTitle` of `CHARACTER` type and the above explained data types. A negative `BlockNumber` limits the `NumberOfVariables` to the perturbation variables ϱ', u', v', [w',] p', otherwise the values of the vorticity [Ω_1', Ω_2',] Ω_3' are added at the end.

Output files of time histories in PIANO's native format have the following look:

```
StandardTitle
VariableNames
DO timelevel = 0, NumberOfLevels
   t
   (up(i(var),j(var),k(var),var), var = 1, NumberOfVariables)
END DO
```

with the predefined `StandardTitle` and `VariableNames` of `CHARACTER` (names separated by colon) type as well as `t` and `up` of `DOUBLE PRECISION` type. `NumberOfLevels` terms the number

of time levels `(Tend-Tbeg+1)/HistoryOut + 1`, `NumberOfVariables` differs from block to block in general, but is indirectly specified by `VariableNames`.
The format of the circle data is analogously the same.

For debug purpose the use of `debugTout`, `debugHistoryOut` or `debugCircOut` maybe make sense (cf. Section 6.2 on page 54).

6.12 Parallelization

The parallelization of the PIANO code is based on the **S**ingle **P**rogram **M**ultiple **D**ata (SPMD) model and uses the **M**essage **P**assing **I**nterface (MPI[3]) library for the communication of the data between the distributed blocks whereas the first CPU is the so-called *master*. Among other things this *primus inter pares* cares for single data file output: `FilRec` and `*.dat` files will be written by this CPU.

6.12.1 Parallelization Strategy

Each processor runs one or more blocks such that the number of blocks is larger or equal the number of processors. The data exchange procedure in parallel mode is the same as in the sequential mode: At boundaries marked by `CutBC` (defined in `parameter.h`) in `FilLog`, the flow variables on the inner three layers of the computational domain are copied to the ghost-point layer flow variables of the neighbouring block and vice versa for each Runge-Kutta sub-step, while an arithmetic mean value is calculated for the coincident nodes at the same time. For the data exchange between two blocks on the same processor the code uses the sequential routines only, whereas the code uses additionally MPI routines to perform the exchange between two blocks on different processors. The parallel code can run on a single processor.

6.12.2 General Recommendations concerning Parallel Runs

- To generate a parallel version of PIANO one uses the preprocessor directive `parallel` for compilation, e. g., one just types `MakeCall parallel Piano`.
- To minimize the elapsed CPU time one sets all parameters controling the output (`Tout`, `HistoryOut`, `CircOut`) to positive values, since in this mode the output data is written to one file per block by each processor, while negative values enforce the master process to collect all data and to write ASCII output into one single file. The latter convenient mode is intended for debugging and needs no post-processing, but it is very time consuming due to data communication and conversion.
- The load-balancing, i. e. the distribution of the blocks on the available processors, may be controlled by specifications in `FilProc`. If no file is given, currently PIANO will deal out the blocks like cards in a game. In file `FilProc` only those blocks, which are not to be distributed by the automatical load-balancing, are redistributed. Its format is convenient and shown in the following listing:

```
$$ CPU number followed by number of these blocks which have to be handled
```

[3] `http://www-unix.mcs.anl.gov/mpi`

```
1 2 4 $$6 8 10
2 1 3 $$ 5 7 9
3 11 12 13
4
```

Anything beyond `$$` is skipped as a comment. Each valid line lists only the blocks which are to be redistributed, but not existing CPU numbers as well as unknown blocks are skipped smartly.

To optimize the elapsed CPU time the following guide lines should be taken into account during the grid generation process for an optimized mapping of the blocks on the processors in decreasing order of importance:

1. The number of grid nodes per processor should be balanced to minimize waiting time of processors (especially avoid peak loads for single processors).
2. The volume to surface ratio should be maximized, that means, e. g., that a rectangular grid should be cut into two blocks with the same number of grid nodes and with a minimized number of grid nodes on the cut surface.
3. The size of the surfaces for the data exchange between different processors should be minimized. That means, e. g., that for a multiblock grid with equally distributed grid nodes and cut surfaces, the mapping between processors and blocks should be done such that the number of grid nodes on the cut surfaces between different processors is minimized. This can be done by appropriate block clustering, since the data exchange by MPI routines is in general slower than by sequential exchange routines on the same processor.

Parallel RPM Runs

There are two different strategies for the use of RPM in parallel mode:

The default method performs the RPM calculation with the first processor, which is the *master* CPU. This strategy implies some additional work subject to the number of base points, a mean to describe the effort due to the RPM business. Consequently the communication overhead increases, too.

The second method can be activated by the directive `rpmmaster`: Then the last processor, the so-called *rpm-master*, will calculate all the RPM stuff exclusively. For manual load-balancing one has to keep the last processor unused! That implies *no* parallel overhead, which means all working processors are waiting just for lagging CPUs, if the RPM calculations are sufficiently efficient. For checking this, it might be helpful to link your executable with `-mpilog` in order to get some parallel performance information of a PIANO test run (therefore switch off all output and run just one time step). In case of too much RPM business, i. e. all CPUs are waiting for the rpm-master, it might be helpful to increase the current quotient of `RPMdt` and `dt`, which indicates how often a turbulent particle leaves and a new particle enters the patch area. If this quotient is too small, too many particles have to be calculated and the effort is tremendous. As a rule of thumb, `RPMdt` should be at least equal to 5 times the stable PIANO `dt`.

6.12.3 Current Restrictions

- In the same simulation ASCII output for history files and binary output for contour plots, i. e. `HistoryOut` < 0 and `Tout` > 0 at the same time, is not available. PIANO will continue with a `HistoryOut` converted to a positive value. With a `HistoryOut` converted to a negative value PIANO will continue for the reversed case, i. e. `HistoryOut` > 0 and `Tout` $<$ 0, if `noTecplotLib` is unset. All other combinations are supported, i. e. ASCII, Tecplot© or binary PIANO format for history, circle and contour data.

- Currently a more sophisticated real automatic load-balancing is not implemented, but the blocks are, as explained above, distributed automatically.

6.13 A first Example

To demonstrate the capabilities of PIANO a simple case may be run. In this way the correct installation may be checked without time consuming number crunching.

6.13.1 Problem Description

As first example a linear two-dimensional aero-acoustic problem (`-DtwoD` has to be used) is simulated:
A localised vortex with Gaussian distribution (s. 6.3.3) rotating around the z-axis is initialised 0.5 chord-lengths upstream of a profile (see Figure 6.2). During its convection towards the Joukowski-type profile of 12 % thickness by analytic free stream of $Ma = 0.5$ it interacts with the airfoil.

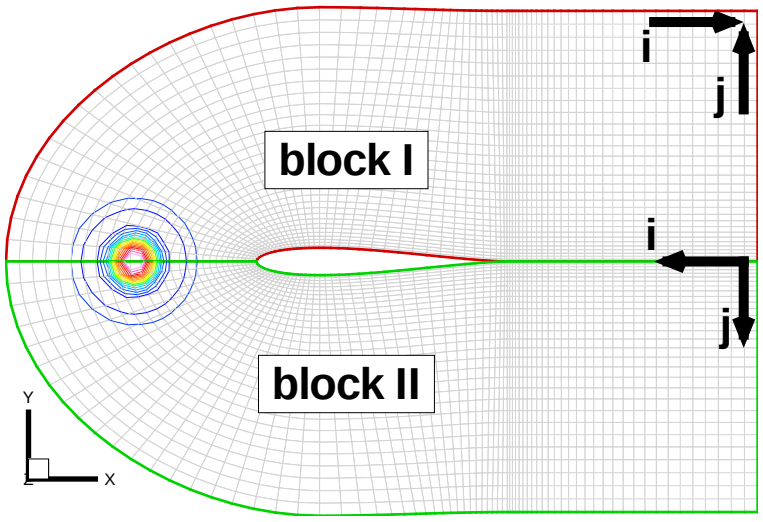


Figure 6.2: Initial vorticity distribution and used mesh of first example

The following listing shows a minimal input file:

```
QuickEnd
DirIn  ./Primer1
FilGrd  GRID
FilMean  Flow.5
FilIJK  Indices.dat
DirOut  ./Primer1/Output
Tout  50   store field values after |Tout| steps in FilNois
$$HistoryOut  3   store time history in FilHis after |HistoryOut| steps
dt  5.D-3   time step size will be compared to global stability limit
Tend  500   number of time steps to be calculated
Filter  6   filter type out of {6,8,100}; 0 means NO filtering
new   start with given initialization (or restart with recorded state)
MagV  1.D0   magnitude for vorticity of vortex at initialization
Xv  -1.D0  0.D0   centre (line) of vortex at initialization
RadV  1.D-1   half-value radius of initial vortex
End  indicates end of input, quod libet may follow :)
```

6.13.2 Mesh

The curvilinear mesh, shown in Figure 6.2 on the preceding page, consists of 2 two-dimensional blocks in the x,y-plane, 71×31 nodes each. The file name for the employed grid is given by keyword `FilGrd` (see line 3 in input file).

6.13.3 Mean-flow

Figures 6.3 and 6.4 show the analytical mean-flow around the Joukowski profile contained in a given file. The file name is specified by keyword `FilMean` (see line 4 in input file).

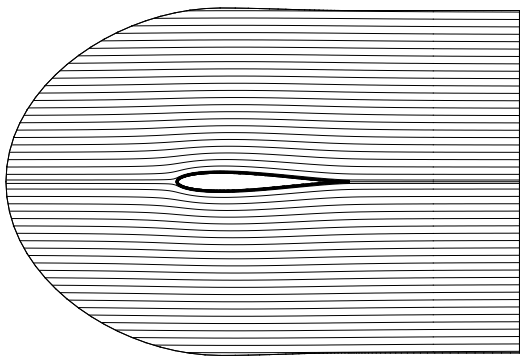


Figure 6.3: Streamlines of the mean-flow

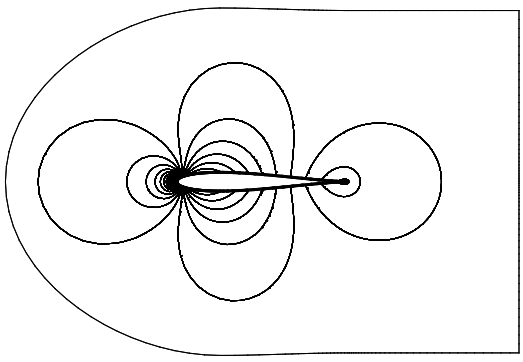


Figure 6.4: Pressure contours of the mean-flow

6.13.4 Boundary Conditions

In this case 'slip wall' for the surface of the profile, 'outflow' for the downstream boundary, and 'radiation' for the remaining edges has been chosen. Following listing shows the used logic file:

```
$$   format for integers in grid point file

$$   format for reals in grid point file

$$   nblock   imax     jmax    kmax   ijkmax    icoord
       2       71       31        1     7227        1
$$
$$   total nodes = 39886, biggest block = 19943, SegMax = 14
$$
$$   topology of block no.      1
$$   ----------------------------
$$   iblock   nseg(1)  nseg(2)  nseg(3)  nseg(4)   nseg(5)  nseg(6)  isolve
       1        1        1        2        1        1        1        1
$$    ibeg     iend     jbeg     jend     kbeg     kend   (physical boundaries)
       2       72        2       32        2        2
$$
$$   segments
$$
$$   ityp lb l1beg l1end l2beg l2end mbls lbs l1begs l1ends l2begs l2ends icomp
$$   cut to another block
    -1     1     2    32     2     2     2     2     2    32     2     2     0
$$   Outflow
    20     2     2    32     2     2     0     0     0     0     0     0     0
$$   slip wall
    10     3     2     2     2    41     0     0     0     0     0     0     0
$$   cut to another block
    -1     3     2     2    42    72     2     3     2     2    32     2     0
$$   Inflow
    21     4     2     2     2    72     0     0     0     0     0     0     0
$$   Symmetry in Z direction (W)
    23     5     2    72     2    32     0     0     0     0     0     0     0
$$   Symmetry in Z direction (W)
    23     6     2    72     2    32     0     0     0     0     0     0     0
$$
$$   topology of block no.      2
$$   ----------------------------
$$   iblock   nseg(1)  nseg(2)  nseg(3)  nseg(4)   nseg(5)  nseg(6)  isolve
       2        1        1        2        1        1        1        1
$$    ibeg     iend     jbeg     jend     kbeg     kend   (physical boundaries)
       2       72        2       32        2        2
$$
$$   segments
$$
```

```
      $$   ityp lb l1beg l1end l2beg l2end mbls lbs l1begs l1ends l2begs l2ends icomp
45    $$   Outflow
          20     1     2    32     2     2     0     0     0     0     0     0     0
      $$   cut to another block
          -1     2     2    32     2     2     1     1     2    32     2     2     0
      $$   cut to another block
50        -1     3     2     2     2    32     1     3     2     2    72    42     0
      $$   slip wall
          10     3     2     2    33    72     0     0     0     0     0     0     0
      $$   Inflow
          21     4     2     2     2    72     0     0     0     0     0     0     0
55    $$   Symmetry in Z direction (W)
          23     5     2    72     2    32     0     0     0     0     0     0     0
      $$   Symmetry in Z direction (W)
          23     6     2    72     2    32     0     0     0     0     0     0     0
```

The setting for the two boundaries normal to the 2D-wing (faces 5 and 6) are not considered! Note the modification of start (line 52) as well as end index (line 25) on faces 3 ('slip wall') and the splitting into two segments, thus an 'inner cut' occurs.

6.13.5 Initial Conditions

Assuming the coordinate system to be located at the centre-line of the profile 0.5 cordlengths downstream the nose, the initial conditions for a half-value radius of $b = 0.1$ write

$$\begin{aligned}
\text{density:} \quad & \varrho'(x,y,0) &&= 0, \\
\text{velocity:} \quad & \vec{u}'(x,y,0) &&= -v_{\max}\, y\, \frac{(\mathrm{e}\ln 4)^{\frac{1}{2}}}{0.1}\, \exp\left[-\ln 2\, \frac{(x+1.)^2+y^2}{0.1^2}\right], \\
& \vec{v}'(x,y,0) &&= v_{\max}\, (x+1.)\, \frac{(\mathrm{e}\ln 4)^{\frac{1}{2}}}{0.1}\, \exp\left[-\ln 2\, \frac{(x+1.)^2+y^2}{0.1^2}\right], \\
\text{pressure:} \quad & p'(x,y,0) &&= 0.
\end{aligned} \tag{6.10}$$

The initial conditions of (6.10) are given by parameters in the input file (see on page 82): The lines 13–15 set the details. Acoustic and entropy spot are deactivated by zero magnitude or commenting out.
The default setting for `AxisV` (for two-dimensional calculations even mandatory) is used.

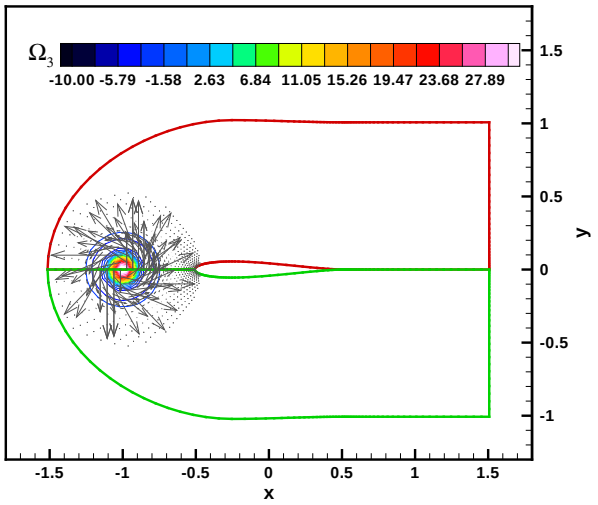
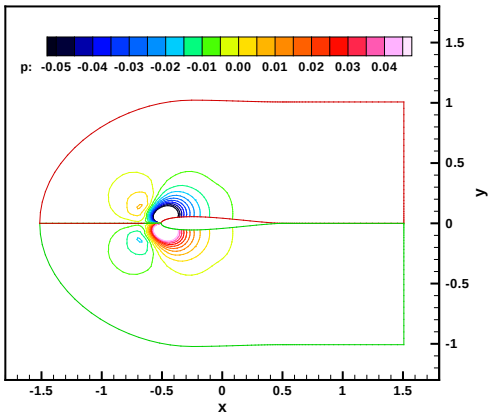
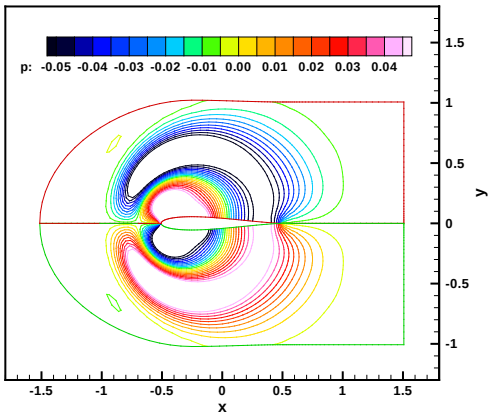


Figure 6.5: Initial vortex

6.13.6 Calculated Results

Figure 6.6 on the following page shows the contours of pressure at different (dimensionless) times, i. e. a plot made out of the data contained in file `Contour_00000150.plt`, e. g. Figure 6.7 on the next page shows for the same settings the vorticity distribution.





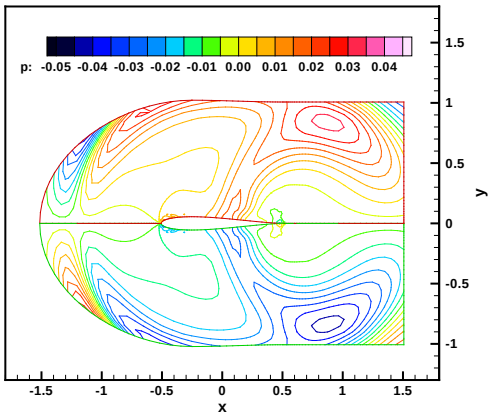
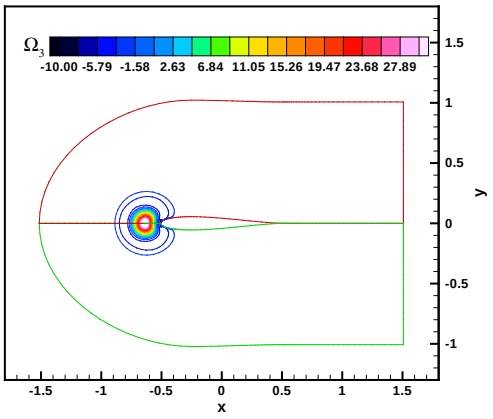
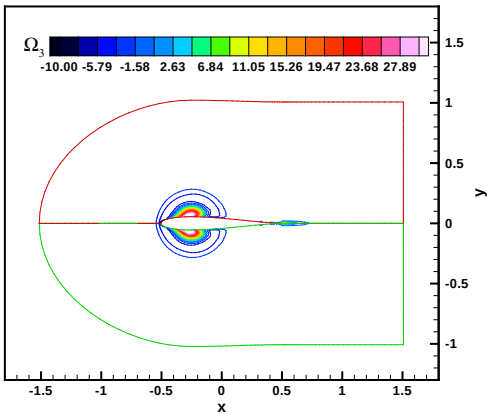


Figure 6.6: Pressure distribution at time $t = 0.75$, $t = 1.5$ and $t = 2.5$ (20 isolines)





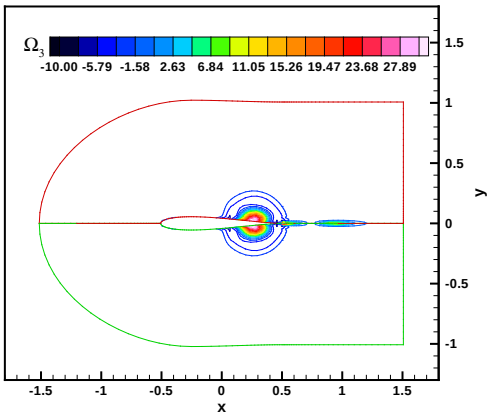


Figure 6.7: Vorticity distribution at time $t = 0.75$, $t = 1.5$ and $t = 2.5$ (20 isolines)

6.14 A second Example

6.14.1 Problem Description

As second example a linear quasi two-dimensional aero-acoustic problem is simulated: A cylindrical acoustic pulse with Gaussian pressure and density distribution (s. 6.3.3) is initialised 0.5 chord-lengths upstream of a profile (see Figure 6.8). During its propagation it is convected towards the Joukowski-type profile of 12 % thickness by analytical free stream of $Ma = 0.5$.

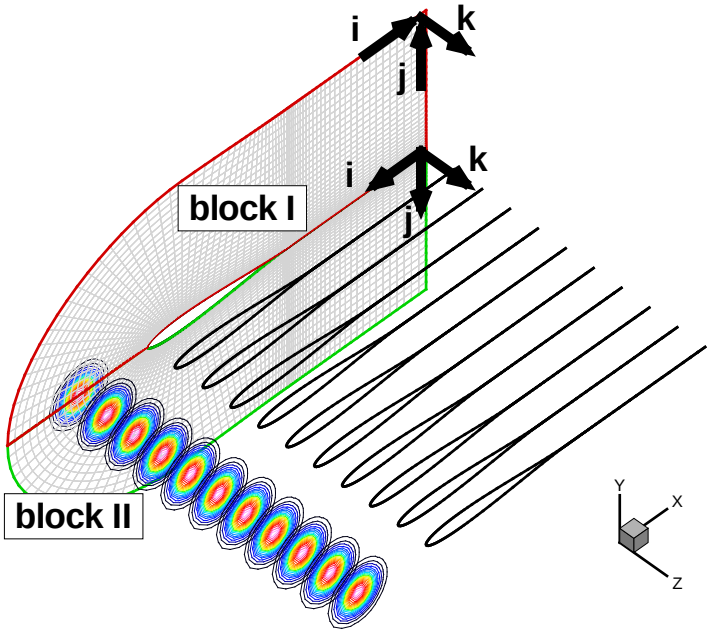


Figure 6.8: Initial pulse and used grid of second example

The following listing shows a minimal input file:

```
QuickEnd
DirIn  ./Primer2
FilGrd  GRID
FilMean  Flow.5
DirOut  ./Primer2/Output
Tout  50   store field values after |Tout| steps in FilNois
$$VorOut   include vorticity into field values and time history
dt  5.D-3   time step size will be compared to global stability limit
Tend  500   number of time steps to be calculated
Xref  .0D0  0.D0  1.D0   reference point
RBD2D   only useful in 3D: just 2D or full 3D boundary conditions?
Filter  6   filter type out of {6,8,100}; 0 means NO filtering
new   start with given initialization (or restart with recorded state)
MagP  1.D0   magnitude of pressure pulse at initialization
Xp  -1.D0  0.D0  0.D0   centre of pressure pulse at initialization
RadP  1.D-1   half-value radius of initial pressure pulse
```

6.14.2 Mesh

The curvilinear mesh, shown in Figure 6.8 on the previous page, consists of 2 three-dimensional blocks, $71 \times 31 \times 11$ nodes each. It is generated by duplicating the grid of the first example using equidistant spaces in z-direction with PreGrid. The file name for the employed grid has to be given by keyword `FilGrd`.

To calculate three-dimensional cases PIANO needs at least 7 nodes especially in z-direction to make up its differencing stencil. In this case the third direction is calculated at every point, despite of the fact that all layers have the same solution.

6.14.3 Mean-flow

Once again the analytical mean-flow shown in figures 6.3 and 6.4 on page 82 is generated with PreFlow and saved in a file specified by keyword `FilMean`.

6.14.4 Boundary Conditions

In this case the same setting as for the first example is used: 'slip wall' for the surface of the profile, 'outflow' for the downstream, and 'radiation' for the 'upstream' boundary. The two boundaries normal to the 2D-wing (faces 5 and 6) have to be defined in a special manner: 'slip wall' assures two-dimensional treatment of the lateral boundaries in addition to activated `RBD2D`.

Following listing shows the used logic file:

```
$$   format for integers in grid point file

$$   format for reals in grid point file

$$   nblock   imax     jmax    kmax   ijkmax    icoord
       2       71       31       11    31317        1
$$
$$   total nodes = 96866, biggest block = 48433, SegMax = 14
$$
$$   topology of block no.      1
$$   ----------------------------
$$   iblock   nseg(1)  nseg(2)  nseg(3)  nseg(4)   nseg(5)  nseg(6)  isolve
       1        1        1        2        1        1        1        1
$$    ibeg     iend     jbeg     jend     kbeg     kend   (physical boundaries)
       2       72        2       32        2       12
$$
$$   segments
$$
$$   ityp lb l1beg l1end l2beg l2end mbls lbs l1begs l1ends l2begs l2ends icomp
$$   cut to another block
      -1     1     2    32     2    12     2     2     2    32     2    12     0
$$    Outflow
      20     2     2    32     2    12     0     0     0     0     0     0     0
$$   slip wall (airfoil)
```

```
25      10     3     2    12     2    41     0     0     0     0     0     0     0
    $$   cut to another block
        -1     3     2    12    42    72     2     3     2    12    32     2     0
    $$   Inflow
        21     4     2    12     2    72     0     0     0     0     0     0     0
30  $$   slip wall
        10     5     2    72     2    32     0     0     0     0     0     0     0
    $$   slip wall
        10     6     2    72     2    32     0     0     0     0     0     0     0
    $$
35  $$   topology of block no.      2
    $$   ----------------------------
    $$   iblock   nseg(1)  nseg(2)  nseg(3)  nseg(4)   nseg(5)  nseg(6)  isolve
           2        1        1        2        1        1        1        1
    $$    ibeg     iend     jbeg     jend     kbeg     kend   (physical boundaries)
40         2       72        2       32        2       12
    $$
    $$   segments
    $$
    $$   ityp lb l1beg l1end l2beg l2end mbls lbs l1begs l1ends l2begs l2ends icomp
45  $$   Outflow
        20     1     2    32     2    12     0     0     0     0     0     0     0
    $$   cut to another block
        -1     2     2    32     2    12     1     1     2    32     2    12     0
    $$   cut to another block
50      -1     3     2    12     2    32     1     3     2    12    72    42     0
    $$   slip wall (airfoil)
        10     3     2    12    33    72     0     0     0     0     0     0     0
    $$   Inflow
        21     4     2    12     2    72     0     0     0     0     0     0     0
55  $$   slip wall
        10     5     2    72     2    32     0     0     0     0     0     0     0
    $$   slip wall
        10     6     2    72     2    32     0     0     0     0     0     0     0
```

Here the start (line 52) as well as the end index (line 25) on faces 3 ('slip wall') have to be modified once again; splitting into two segments, one as 'slip wall', the other as 'inner cut', occurs also.

6.14.5 Initial Conditions

Assuming the coordinate system to be located again at the centre-line of the profile 0.5 cordlengths downstream the nose, the initial conditions for a half-value radius of $b = 0.1$ write

$$\begin{aligned} \text{density:} \quad \varrho'(x,y,0) &= p'(x,y,0), \\ \text{velocity:} \quad \vec{v}'(x,y,0) &= \vec{0}, \\ \text{pressure:} \quad p'(x,y,0) &= p_{\max} \, \exp\left[-\ln 2 \, \frac{(x+1.0)^2 + y^2}{0.1^2} \right]. \end{aligned} \tag{6.11}$$

The initial conditions of (6.11) shown in Figure 6.8 on page 87 are adjusted by the parameters in line 14–16 of the input file.

6.14.6 Calculated Results

Figures 6.10 and 6.11 on the facing page show the distribution of the velocity components u' and v' at different (dimensionless) times for one of the 11 slices with the same solution. Figure 6.9 shows for the same settings the pressure contours, i. e. a plot made out of the data contained in file `Contour_00000150.plt`, e. g.

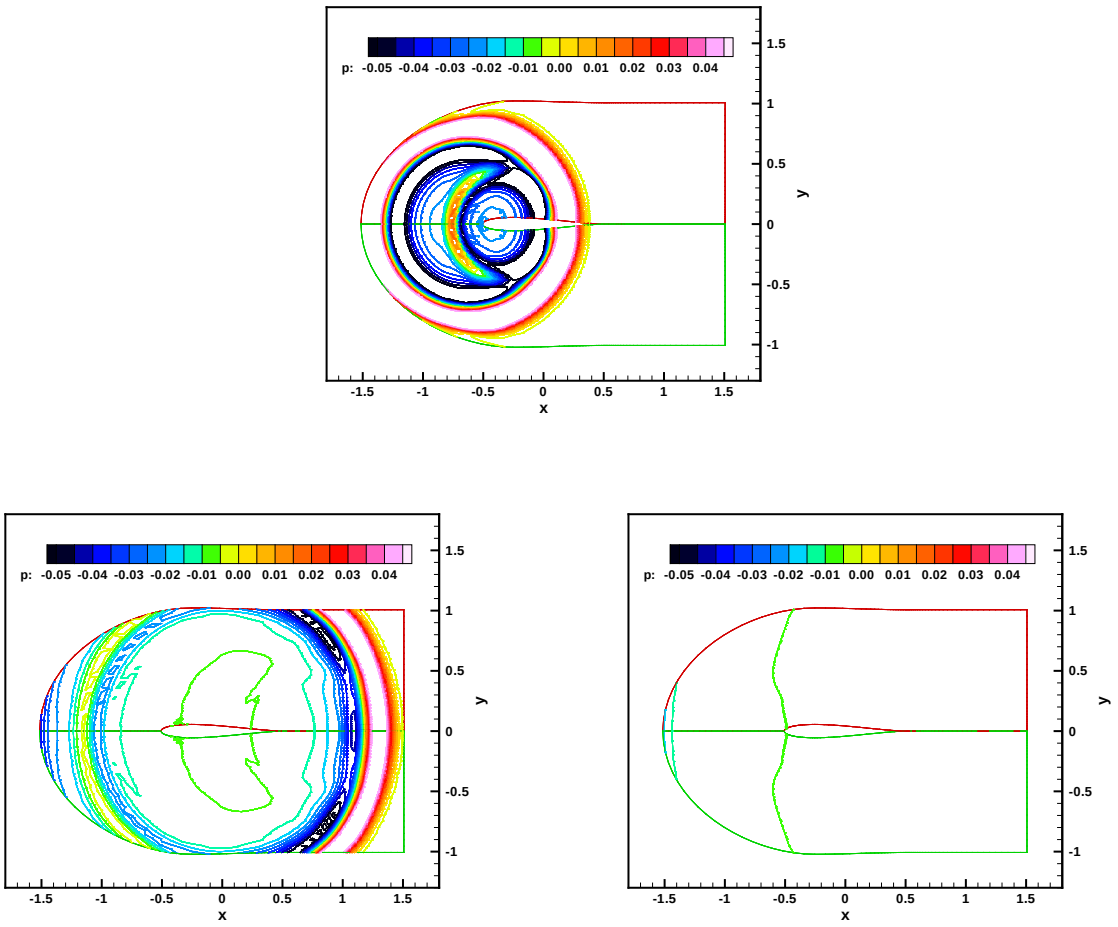
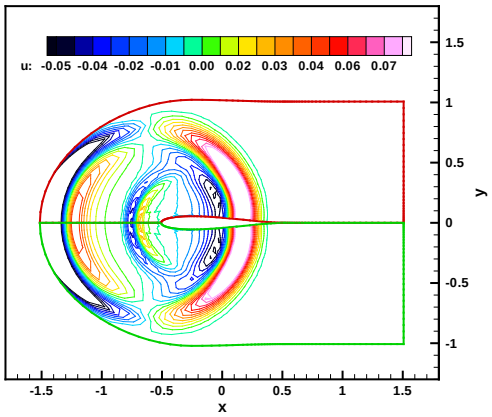
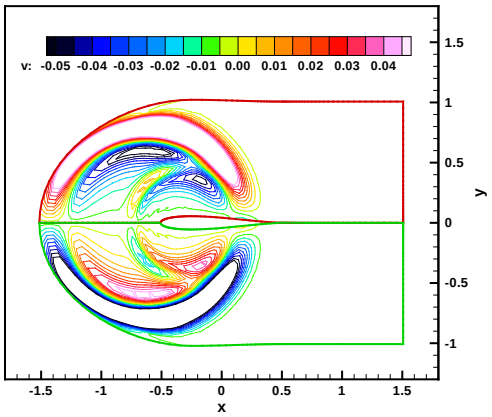
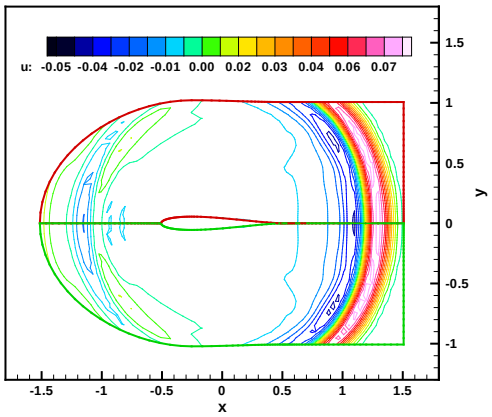
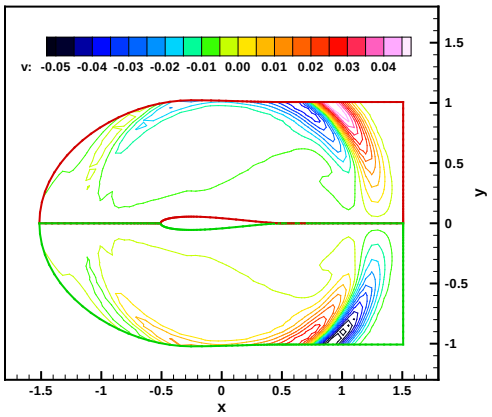


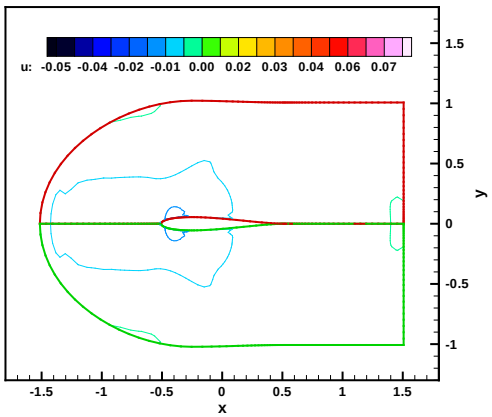
Figure 6.9: Pressure distribution at time $t = 0.75$, $t = 1.5$ and $t = 2.5$ (20 isolines)











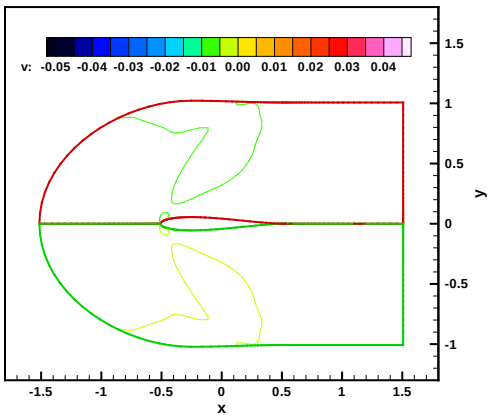


Figure 6.10: u'-velocity distribution at time $t = 0.75$, $t = 1.5$ and $t = 2.5$ (20 isolines)

Figure 6.11: v'-velocity distribution at time $t = 0.75$, $t = 1.5$ and $t = 2.5$ (20 isolines)

6.15 An Example for a SNGR Calculation

6.15.1 Problem Description

As an example the broadband trailing edge noise of a flat plate is going to be simulated with the SNGR method. In this case the source mechanism and the synthetic turbulence are used first of all to excite turbulent fluctuations in the modified Euler equations that will be solved by PIANO. The patch is located thereby above the trailing edge so that the turbulence developing in PIANO will interact with the trailing edge and thus the trailing edge noise is coming into existence. Therefore the sound waves are generated indirectly through the source mechanism. The length of the infinite thin plate L is 0.2 m, the angle of attack of the plate 0° and the two-dimensional flow uniform at $Ma = 0.11$.

6.15.2 Mesh, Mean Flow, and Boundary Conditions

The two-dimensional Cartesian grid is made up of two blocks with 434 × 111 calculation nodes each. It covers only the area around the trailing edge of the flat plate. In Figure 6.12 the plate is finitely thick for getting visible, the trailing edge is at $(x; y) = (0; 0)$.

There are no particularities concerning the mean flow (given in `FilMean`) calculated by FLOWer and interpolated on the PIANO grid as well as concerning the boundary conditions (specified in `FilLog`) used.

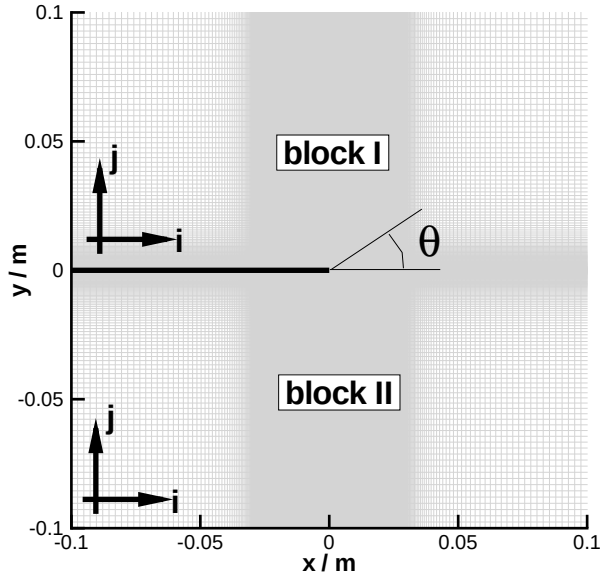


Figure 6.12: Grid for the SNGR example

6.15.3 SNGR Input

Since the SNGR method is used, the initialisation with a vortex or with an acoustic or entropic pulse is deactivated in the input file.

The shear source term has been used and according to equation (2.71) on page 33 the term $v'_j \mathrm{d}\,\bar{v}_i x_j$ slips on the left hand side of the equations of motion.

The SNGR specific input parameters are given in the file `SNGR.dat`: The mode realisation used implies $M = 300$ modes on the whole out of which only the modes up to $N = 97$ are considered because of the delimited resolution of the grid (see also `SNGR_Modes.dat`). The history of $W(y)$ has been adapted to the history of the kinetic turbulence energy k above the trailing edge. The parameters x_l and x_r were chosen such, that for a specific `alpha_1_grenz` given in `sngrreal.m` as less as possible spurious noise or vorticity is generated. For the maximum $W(x_c)$ applies $x_c = 0$.

6.15.4 Calculated Results

Figure 6.13 shows isolines of the non-dimensional pressure p at the non-dimensional time $t = 1.05$, i. e. after the last calculation step of the simulation. The zoomed area around the trailing edge of the flat plate is shown in Figure 6.14 on page 95. The history of p over the non-dimensional time t is depicted in Figure 6.15 on page 95. This time history was recorded by a virtual microphone that is located at $(x; y) = (0; 0.08\,\mathrm{m})$.

The power spectral density (PSD) of the dimensional sound pressure $p^*(t^*)$ calculated from this time history is shown in Figure 6.16 on page 96.

Figure 6.17 is a presentation of the directivity pattern calculated from the simulation

$$\Gamma = \frac{\tilde{p}^2(\theta)}{\tilde{p}^2(\theta)_{\max}}$$

and the theoretically expected directivity pattern in the form of the cardioid

$$\Gamma = \sin^2(\theta/2)\ .$$

The angle θ can be seen in Figure 6.12 on the preceding page. For recording the directivity pattern 24 virtual microphones at intervals of 15° were arranged in the calculation area on a circle around the trailing edge with the radius $R = 0.08\,\mathrm{m}$. The recording of $p(t)$ was started after the initial pressure pulse generated in the source area had passed the virtual microphones in order to get the physical signal only. The calculation of the directivity pattern and the PSD were made by means of the attached MATLAB routines `savedat.m` and `Richtspec.m`. The PIANO output file *`FilHis`*`d8(Tend)_P.dat` had been adapted to the input format needed by MATLAB by means of Tecplot© and an editor:

1. PIANO's output file *`FilHis`*`d8(Tend)_P.dat` was transformed into an ASCII file in the point format with a user-defined title (`Time_P.m` e. g.) by Tecplot©.
2. Further adaptation of `Time_P.m` on the MATLAB format (attaching MATLAB header, putting data files between brackets) done by an editor.
3. Execution of `savedat.m` (transformation of `Time_P.m` into the MATLAB binary format) by MATLAB.

4. Execution of `Richtspec.m` (calculation of directivity pattern and spectrum) by MATLAB.

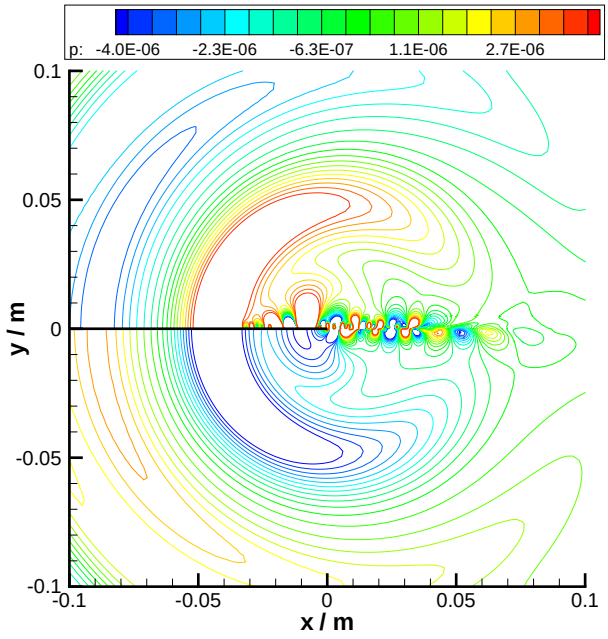


Figure 6.13: Non-dimensional pressure p at $t = 1.05$ (20 isolines)

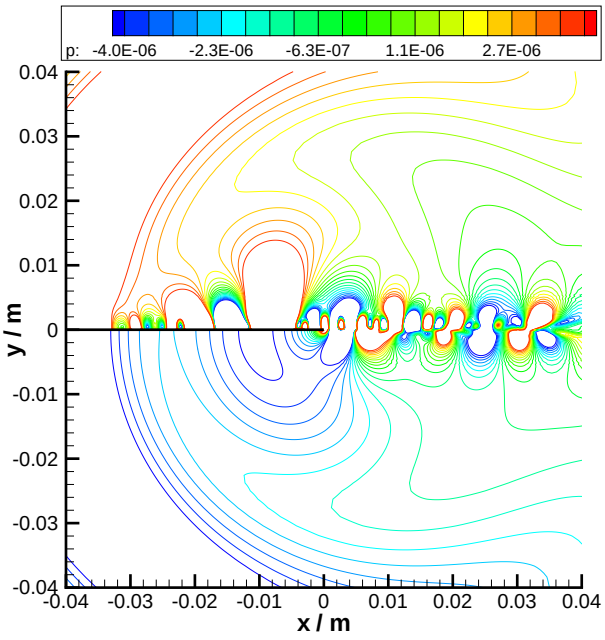


Figure 6.14: Zoomed representation of the non-dimensional pressure p at $t = 1.05$ at the trailing edge of the flat plate (20 isolines)

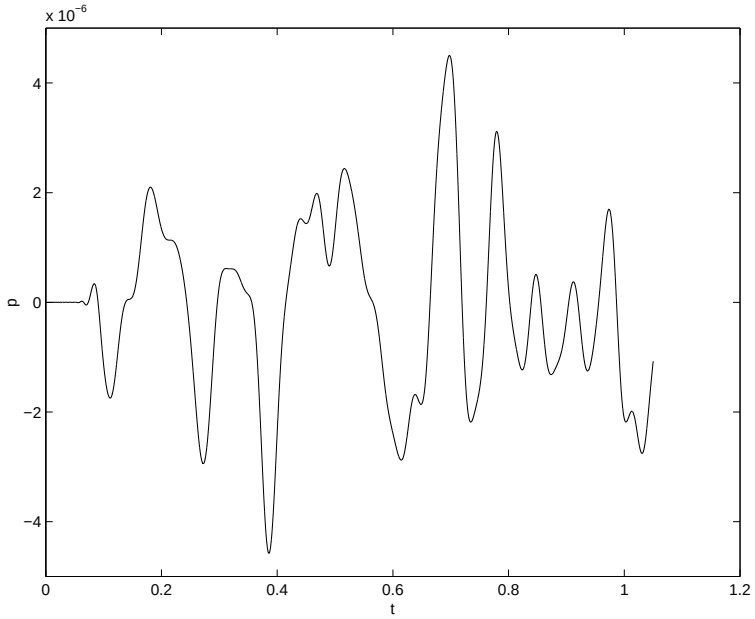


Figure 6.15: Time history $p(t)$ at $(x; y) = (0; 0.08\,\mathrm{m})$

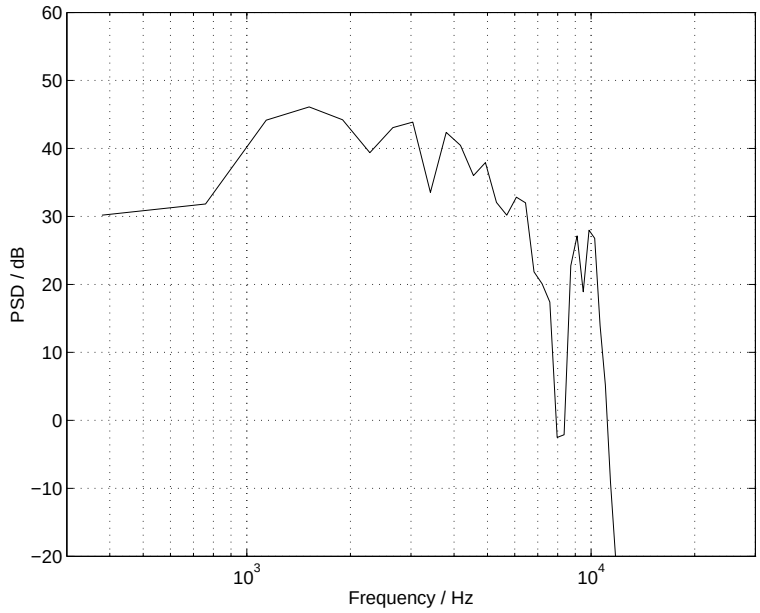


Figure 6.16: Power spectral density (PSD) of the sound pressure at $(x; y) = (0; 0.08\,\mathrm{m})$

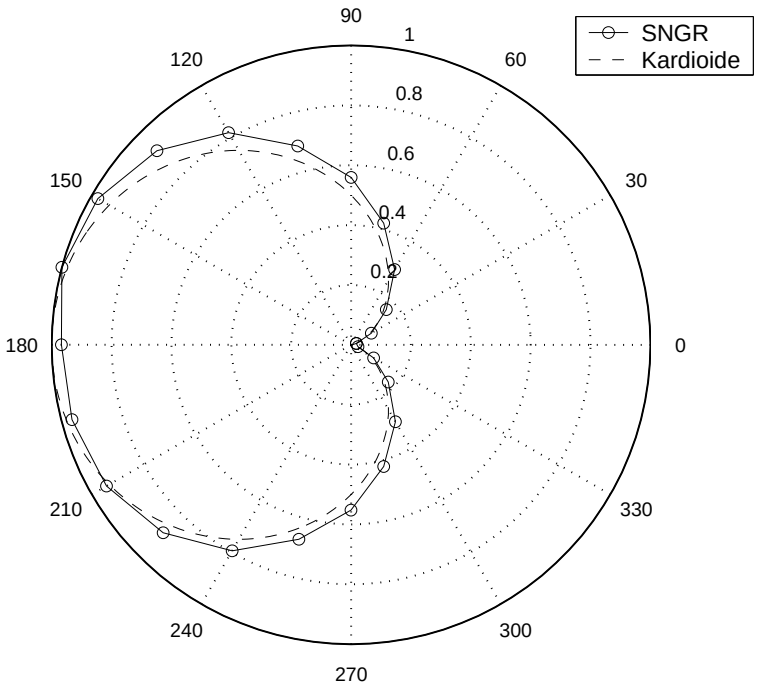


Figure 6.17: Directivity pattern and cardioid calculated with the SNGR method

6.16 An Example for a RPM Calculation

6.16.1 Problem Description

As an example the broadband trailing edge noise of a flat plate is going to be simulated with the RPM method. In this case the source mechanism and the synthetic turbulence are used first of all to excite turbulent fluctuations in the modified Euler equations that will be solved by PIANO. The patch is located thereby around the trailing edge so that the turbulence developing in PIANO will interact with the trailing edge and thus the trailing edge noise is coming into existence. Therefore the sound waves are generated indirectly through the source mechanism. The length of the infinite thin plate L is 0.2 m, the angle of attack of the plate equals to 0° and the velocity of the two-dimensional uniform flow amounts $Ma = 0.11$.

6.16.2 Mesh, Mean Flow, and Boundary Conditions

The two-dimensional Cartesian grid is made up of six blocks with at least 66810 nodes. The trailing edge is located at $(x; y) = (0; 0)$.

The homogeneous mean flow (given in `FilMean`) calculated by FLOWer and interpolated onto the PIANO grid enters the domain from the left.

The flate plate is realized by an adiabatic wall boundary condition, on all other boundaries the radiation condition is used, as specified in `FilLog`.

6.16.3 RPM Input

For the RPM method a patch file has to be created via the streamtrace concept, for details see Section 6.6.1 on page 69.

The following listing shows the additional parameters of the PIANO input file:

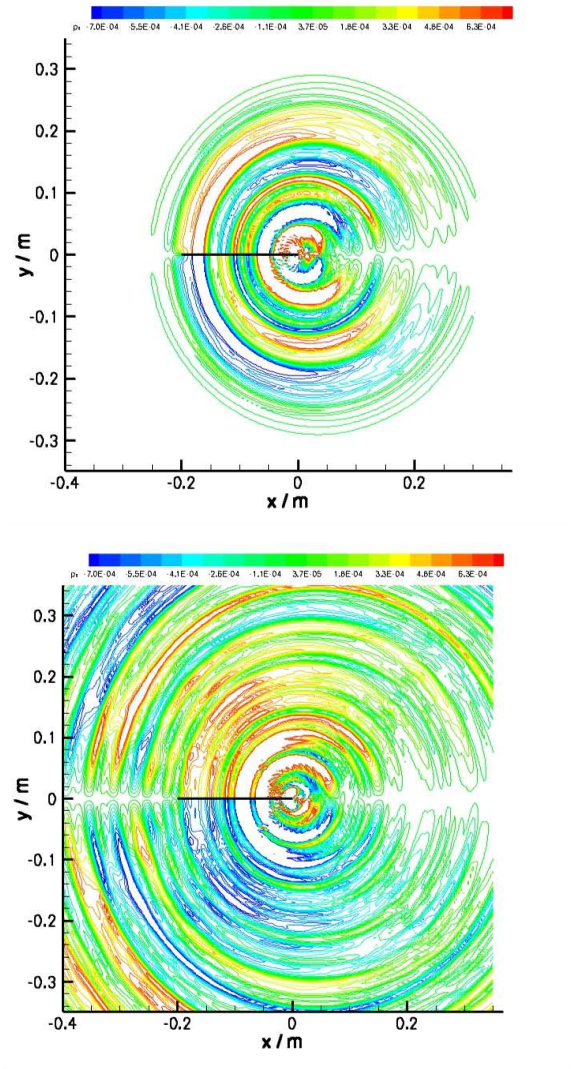
```
FilRPM  patch011.dat
FilRPMRec  RPMrecord.bin
$$ RPMOut  include stochastic source information into field values and time history
RPMdt    2.1D-3  time step of white-noise field
RPMlimit 0.001D0 minimal used length scale
RPMup    0.02D0  upstream source window
RPMdown  0.02D0  downstream source window
RPMfac   6.D0    scaling factor for patch-data length scale
RPMalfa  1.D0    Langevin coefficient exp(-RPMdt/tscale)
RPMtau   333.D0  time decay constant for ramping source term: 1-exp(-Tstep/tau)
```

In this case applies `RPMdt` $= 7 \cdot$ `dt`, that means after 7 PIANO time steps a new vortex particle enters the patch and the last particle layer leaves the source area.

6.16.4 Calculated Results

Figure 6.18 on the next page shows isolines of the non-dimensional pressure p at different non-dimensional times. The history of p over the non-dimensional time t is depicted in Figure 6.19

on the facing page. This time history was recorded by a virtual microphone located at $(x; y) = (0; 0.3\,\mathrm{m})$.



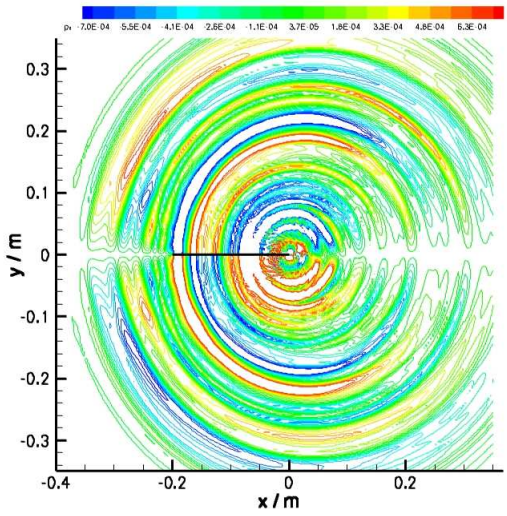


Figure 6.18: Pressure distribution at time $t = 0.3$, $t = 0.45$ and $t = 0.9$ (20 isolines)

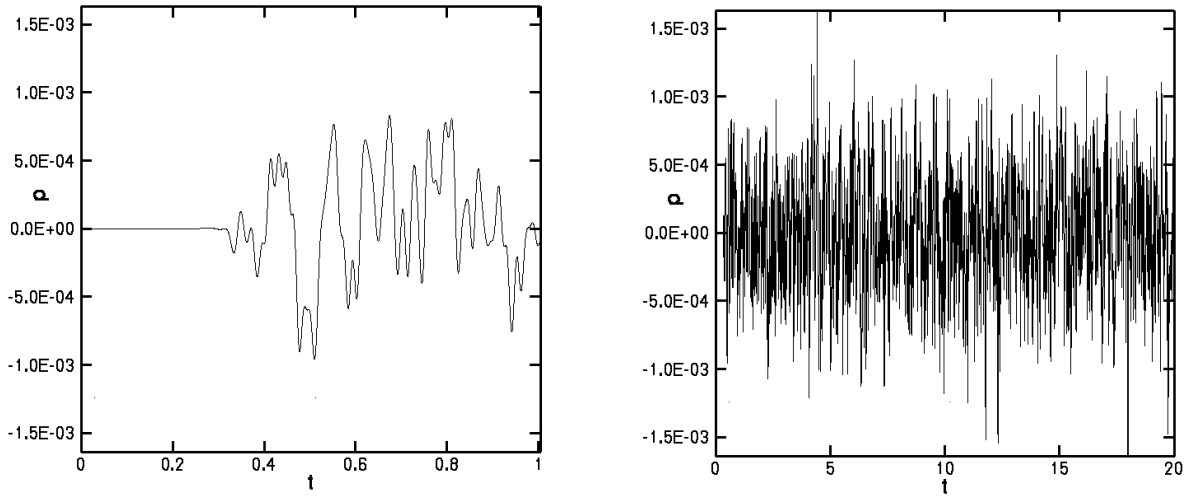


Figure 6.19: Time history of a virtual microphone located at $(x; y) = (0; 0.3\,\mathrm{m})$

Chapter 7

Current Limits of PIANO

Since PIANO is still under development, not all desireable features are yet implemented and there even might be some bugs. That is why we need the user's feedback: What is missing? Where do errors occur? What is nice? For which application was PIANO employed?

7.1 View to Future Developments

Following some known incapabilities which will be overcome in the future are given without ranking:

- Locations for history plots cannot be given in coordinates. Useful interpolation for arbitrary positions will be integrated as soon as possible. Until then arbitrary circles may be specified to generate directivities.
- A tool for postprocessing of history data will be added to the distribution, which also performs the data conversion of binary files.
- An interface to couple PIANO with APSIM will be supplemented.
- The implemented boundary conditions do not fulfill their task in each case properly; some new concepts, e. g. impedance or axial-symmetry, will be developed.
- The restricted input of RHS data and convenient SNGR developments will be integrated.
- The RPM concept will be extended in order to handle full three-dimensional cases.
- An automatic algorithm of load-balancing will be implemented when a detailed experience considering all influencing factors is available.

7.2 Useful Bug Reports

If unexpected results are produced by PIANO and no reasonable explanation (even with any announcement on PIANO's web site or in `KnownBugs`) is found, an useful bug report could help *all* users and increases the quality of PIANO.

For a quick response the few listed instructions should be followed:

- The input and configuration should be checked carefully in order to make sure that really no misuse or handling error causes the fact to be criticised.
- In- and output have to be reduced as far as possible: As many features as possible should be switched off, the most simple configuration and grid available are to be used.
- Tecplot's *binary* files are preferred as PIANO's output.
- In addition to some appropriate information concerning hardware configuration, compiler, libraries and error following files are needed at least: PIANO's protocol, useful output files.

  In order to reproduce the errorneous behaviour the input data and, if PIANO is tampered, the source code would be helpful.
- Data has to be packed and compressed for transfer. Attachments greater than 3 MB are unwillingly accepted. Access to ftp.dlr.de could be offered instead!

Bug reports may be send to one of the following addresses:

Prof. Dr.-Ing. Jan Delfs
Deutsches Zentrum für Luft- und Raumfahrt e. V.
in der Helmholtz-Gemeinschaft
Institut für Aerodynamik und Strömungstechnik
Technische Akustik
Lilienthalplatz 7
38108 Braunschweig
tel. ++49-531-295-2170
jan.delfs@dlr.de

Dipl.-Ing. Thomas Lauke
DLR
AS/TA
Lilienthalplatz 7
38108 Braunschweig
fon ++49-531-295-3317
fax ++49-531-295-2320
thomas.lauke@dlr.de

Bibliography

[AM06] A. Agarwal and P. J. Morris. Prediction method for broadband noise from unsteady flow in a slat cove. *AIAA Journal*, 44(2):301–310, 2006.

[Bat60] G. Batchelor. *The Theory of Homogeneous Turbulence*. Cambridge University Press, 1960.

[Bau03] Markus Bauer. Berechnung der Schallabstrahlung überströmter Hinterkanten (Hinterkantenlärm). Master's thesis, Institut für Akustik und Sprachkommunikation, Technische Universität Dresden, 2003.

[BBE+00] [Wolf] Bartelheimer, [Hans] Bleecke, [Bernhard] Eisfeld, [Jan] Lieser, [Ralf] Heinrich, [Norbert] Kroll, [Martin] Kuntz, [Erik] Monsen, and [Jochen] Raddatz. *Installation and User Handbook for the Project FLOWer*. DLR, Institut für Aerodynamik und Strömungstechnik, 38108 Braunschweig, Lilienthalplatz 7, Mai 2000.

[BED03] M. Billson, L.-E. Erikson, and L. Davidson. Jet noise prediction using stochastic turbulence modeling. *AIAA Paper*, (2003-3282), 2003.

[BJ99] C. Bailly and D. Juvé. A stochastic approach to compute subsonic noise using linearized Euler's equations. *AIAA Paper*, (99-1872), 1999.

[BLC95] C. Bailly, P. Lafon, and S. Candel. A stochastic approach to compute noise generation and radiation of free turbulent flows. In *1st AIAA/CEAS Aeroacoustics Conference*, number 95-092. AIAA/CEAS, 1995.

[BS79] I. N. Bronstein and K. A. Semendjajew. *Taschenbuch der Mathematik*. BSB B. G. Teubner Verlagsgesellschaft, Leipzig, 1979.

[CP01] A. J. Cooper and N. Peake. Propagation of unsteady disturbances in a slowly varying duct with mean swirling flow. *J. Fluid Mech.*, 445:207–234, 2001.

[CSS93] A. Careta, F. Sagués, and J. M. Sancho. Stochastic generation of homogeneous isotropic turbulence with well-defined spectra. *Physical Review E*, 48(3):2279–2287, 1993.

[EDL05] Roland Ewert, Jan-Werner Delfs, and Markus Lummer. The simulation of airframe noise applying Euler-perturbation and acoustic analogy approaches. *International Journal of Aeroacoustics*, 4(1+2):69–91, 2005.

[EE05] Roland Ewert and Rolf Emunds. CAA slat noise studies applying stochastic sound sources based on solenoidal digital filters. *AIAA Paper*, (2005-2862), 05 2005.

[EMS02] R. Ewert, M. Meinke, and W. Schröder. Computation of trailing edge noise via LES and acoustic perturbation equations. *AIAA Paper*, (2002-2467), 2002.

[ES03] R. Ewert and W. Schröder. Acoustic perturbation equations based on flow decomposition via source filtering. *Journal of Computational Physics*, 188:365–398, 2003.

[ES04] R. Ewert and W. Schröder. On the simulation of trailing edge noise with a hybrid LES/APE method. *Journal of Sound and Vibration*, 270(1):509–524, 2004.

[Ewe] R. Ewert. Broadband slat noise prediction based on CAA and stochastic sound sources from a random particle-mesh (RPM) method. *in press Comp. and Fluids*.

[Ewe06] R. Ewert. Slat noise trend predictions using CAA with stochastic sound sources from a random particle-mesh method (RPM). *AIAA Paper*, (2006-2667), 2006.

[Ewe07] R. Ewert. RPM — the fast random particle-mesh method to realize unsteady turbulent sound sources and velocity fields for CAA applications. *AIAA Paper*, (2007-3506), 2007.

[Fle97] C. A. J. Fletcher. *Near-Orthogonal Grids*, volume 2, chapter 13.2.5. Springer, computational techniques for fluid dynamics edition, 1997.

[GA98] V. V. Golubev and H. M. Atassi. Acoustic-vorticity waves in swirling flows. *Journal of Sound and Vibration*, 209(2):203–222, 1998.

[GLL01] H. A. Grogger, M. Lummer, and T. G. W. Lauke. Simulating the interaction of a three-dimensional vortex with airfoils using CAA. In *7th AIAA/CEAS Aeroacoustics Conference, Maastricht (nl), 28.–30.05.2001*, number 2001-2137. AIAA/CEAS, 2001.

[Gol78] M. E. Goldstein. Unsteady vortical and entropic disturbances of potential flows round arbitrary obstacles. *J. Fluid Mech.*, 891:433–468, 1978.

[HHM96] F. Q. Hu, M. Y. Hussaini, and J. L. Manthey. Low-dissipation and low-dispersion Runge-Kutta schemes for computational acoustics. *Journal of Computational Physics*, 124:177–191, 1996.

[How75] M. S. Howe. Contributions to the theory of aerodynamic sound, with application to excess jet noise and the theory of the flute. *J. Fluid Mech.*, 71(4):625–673, 1975.

[How98] M. S. Howe. *Acoustics of Fluid-Structure Interactions*. Cambridge University Press, 1998.

[How99] M. S. Howe. On the scattering of sound by a rectilinear vortex. *Journal of Sound and Vibration*, 227(5):1003–1017, 1999.

[KSJ03] M. Klein, A. Sadiki, and J. Janicka. A digital filter based generation of inflow data for spatially developing direct numerical or large eddy simulations. *Journal of Computational Physics*, 186:652–665, 2003.

[KW00] N. Kalitzin and A. Wilde. Entwicklung eines Verfahrens zur numerischen Modellierung des aeroakustischen Quellmechanismus' an den Klappenseitenkanten von Flugzeugtragflügeln. Technical report, Abschlussbericht zum DFG-Projekt Ko 1242/6-1 und /6-2, Institut für Akustik und Sprachkommunikation, Technische Universität Dresden, May 2000. 26.05.2000.

[LDL02] Markus Lummer, Jan W[erner] Delfs, and Thomas [G.W.] Lauke. Simulation of sound generation by vortices passing the trailing edge of airfoils. AIAA-paper 2002-2578, American Institute of Aeronautics and Astronautics, 2002.

[Lel92] Sanjiva K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992.

[LGD01] Markus Lummer, Herwig A. Grogger, and Jan W. Delfs. Using RANS mean flow fields in numerical aeroacoustics simulations (CAA). In *NATO RTO-AVT Symposium on Aging Mechanisms and Control, Part A — Development in Computational Aero- and Hydro-Acoustics, Manchester, UK*, October 8–11, 2001.

[Lig52] M. J. Lighthill. On sound generated aerodynamically: I. General theory. *Proc. R. Soc. London Ser. A*, 211:564–587, 1952.

[Lon98] E. Longatte. *Modélisation de la propagation et de la génération du bruit au sein des écoulements turbulents internes*. PhD thesis, École Centrale Paris, 1998.

[McN72] W. D. McNally. Fortran program for generating a two-dimensional orthogonal mesh between two arbitrary boundaries. *NASA TN-D6766*, 1972.

[Möh79] W. Möhring. Modelling low Mach number noise. In E.-A. Müller, editor, *Mechanics of Sound Generation in Flows*. Springer, 1979.

[Möh99] W. Möhring. A well posed acoustic analogy based on a moving acoustic medium. In P. Költzsch and N. Kalitzin, editors, *SWING Aeroacoustic Workshop, Dresden, Germany*, 1999.

[Pie90] A. D. Pierce. Wave equation for sound in fluids with unsteady inhomogeneous flow. *Journal of the Acoustical Society of America*, 87/6:2292–2299, 1990.

[Pow64] A. Powell. Theory of vortex sound. *Journal of the Acoustical Society of America*, 36/1:177–195, 1964.

[Sha99] J. S. Shang. High-order compact-difference schemes for time-dependent Maxwell equations. *Journal of Computational Physics*, 153:312–333, 1999.

[SSC01] A. Smirnov, S. Shi, and I. Celik. Random flow generation technique for large eddy simulations and particle dynamics modeling. *Journal of Fluids Engineering*, 123:359–371, 2001.

[TA99] C. K. W. Tam and L. Auriault. Jet mixing noise from fine-scale turbulence. *AIAA Journal*, 37(2):145–153, 1999.

[TD94] C. K. W. Tam and Z. Dong. Wall boundary conditions for high-order finite-difference schemes in computational aeroacoustics. *Theoret. Comput. Fluid Dynamics*, 6:303–322, 1994.

[TW92] C. K. W. Tam and J. C. Webb. Dispersion-relation-preserving finite difference schemes for computational acoustics. *Journal of Computational Physics*, 107:262–281, 1992.

[TWD93] C. K. W. Tam, J. C. Webb, and Z. Dong. A study of the short wave components in computational acoustics. *Journal of Computational Acoustics*, 1:1–30, 1993.

[VLM98] Oleg V. Vasilyev, Thomas S. Lund, and Parviz Moin. A general class of commutative filters for LES in complex geometries. *Journal of Computational Physics*, 146:82–104, 1998.

[Zie96] R[obin] Ziegler. LOGIC: A semi-automatic interface between MegaCads and FLOWer. Institutsbericht IB 129-96/34, DLR, Institut für Entwurfsaerodynamik, 38108 Braunschweig, Lilienthalplatz 7, August 1996.

Appendix A

Linearized Euler Equations in Expanded Form

The mean flow and perturbation velocity vectors, the Cartesian operator, and the Jacobian matrix (2.22) on page 17 are

$$\vec{v}_0 = \begin{pmatrix} \bar{u} \\ \bar{v} \\ \bar{w} \end{pmatrix}, \quad \vec{v}' = \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix}, \quad \frac{\partial}{\partial x_i} = \frac{\partial \xi_m}{\partial x_i}\frac{\partial}{\partial \xi_m}, \qquad J = \begin{pmatrix} \xi_x & \eta_x & \zeta_x \\ \xi_y & \eta_y & \zeta_y \\ \xi_z & \eta_z & \zeta_z \end{pmatrix} = (J^{-1})^{-1} = \begin{pmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ x_\zeta & y_\zeta & z_\zeta \end{pmatrix}^{-1}. \tag{A.1}$$

A.1 Continuity

$$\frac{\partial \varrho'}{\partial t} + \vec{v}' \cdot J\nabla_\xi \varrho_0 + (\vec{v}_0 + \varepsilon\vec{v}') \cdot J\nabla_\xi \varrho' + (J\nabla_\xi) \cdot \vec{v}_0\, \varrho' + (J\nabla_\xi) \cdot \vec{v}'(\varrho_0 + \varepsilon\varrho') = 0 \tag{A.2a}$$

equivalent to

$$\frac{\partial \varrho'}{\partial t} + v_i'\frac{\partial \xi_m}{\partial x_i}\frac{\partial \bar{\varrho}}{\partial \xi_m} + (\bar{v}_i + \varepsilon v_i')\frac{\partial \xi_m}{\partial x_i}\frac{\partial \varrho'}{\partial \xi_m} + \frac{\partial \xi_m}{\partial x_i}\frac{\partial \bar{v}_i}{\partial \xi_m}\,\varrho' + \frac{\partial \xi_m}{\partial x_i}\frac{\partial v_i'}{\partial \xi_m}\,(\bar{\varrho} + \varepsilon\varrho') = 0 \tag{A.2b}$$

expands to

$$\begin{aligned}
&\frac{\partial \varrho'}{\partial t} + u'\left(\frac{\partial \bar{\varrho}}{\partial \xi}\xi_x + \frac{\partial \bar{\varrho}}{\partial \eta}\eta_x + \frac{\partial \bar{\varrho}}{\partial \zeta}\zeta_x\right) + v'\left(\frac{\partial \bar{\varrho}}{\partial \xi}\xi_y + \frac{\partial \bar{\varrho}}{\partial \eta}\eta_y + \frac{\partial \bar{\varrho}}{\partial \zeta}\zeta_y\right) + \\
&w'\left(\frac{\partial \bar{\varrho}}{\partial \xi}\xi_z + \frac{\partial \bar{\varrho}}{\partial \eta}\eta_z + \frac{\partial \bar{\varrho}}{\partial \zeta}\zeta_z\right) + (\bar{u} + \varepsilon u')\left(\frac{\partial \varrho'}{\partial \xi}\xi_x + \frac{\partial \varrho'}{\partial \eta}\eta_x + \frac{\partial \varrho'}{\partial \zeta}\zeta_x\right) + \\
&(\bar{v} + \varepsilon v')\left(\frac{\partial \varrho'}{\partial \xi}\xi_y + \frac{\partial \varrho'}{\partial \eta}\eta_y + \frac{\partial \varrho'}{\partial \zeta}\zeta_y\right) + (\bar{w} + \varepsilon w')\left(\frac{\partial \varrho'}{\partial \xi}\xi_z + \frac{\partial \varrho'}{\partial \eta}\eta_z + \frac{\partial \varrho'}{\partial \zeta}\zeta_z\right) + \\
&\varrho'\left(\frac{\partial \bar{u}}{\partial \xi}\xi_x + \frac{\partial \bar{u}}{\partial \eta}\eta_x + \frac{\partial \bar{u}}{\partial \zeta}\zeta_x + \frac{\partial \bar{v}}{\partial \xi}\xi_y + \frac{\partial \bar{v}}{\partial \eta}\eta_y + \frac{\partial \bar{v}}{\partial \zeta}\zeta_y + \frac{\partial \bar{w}}{\partial \xi}\xi_z + \frac{\partial \bar{w}}{\partial \eta}\eta_z + \frac{\partial \bar{w}}{\partial \zeta}\zeta_z\right) + \\
&(\bar{\varrho} + \varepsilon\varrho')\left(\frac{\partial u'}{\partial \xi}\xi_x + \frac{\partial u'}{\partial \eta}\eta_x + \frac{\partial u'}{\partial \zeta}\zeta_x + \frac{\partial v'}{\partial \xi}\xi_y + \frac{\partial v'}{\partial \eta}\eta_y + \frac{\partial v'}{\partial \zeta}\zeta_y + \frac{\partial w'}{\partial \xi}\xi_z + \frac{\partial w'}{\partial \eta}\eta_z + \frac{\partial w'}{\partial \zeta}\zeta_z\right) = 0
\end{aligned} \tag{A.3}$$

A.2 Momentum

$$\frac{\partial \vec{v}'}{\partial t} + \vec{v}' \cdot J\nabla_\xi \vec{v}_0 + (\vec{v}_0 + \varepsilon\vec{v}') \cdot J\nabla_\xi \vec{v}' + \frac{1}{\bar{\varrho}} \left(1 - \varepsilon \frac{\varrho'}{\bar{\varrho}}\right) \left(J\nabla_\xi p' + \varrho'\, \vec{v}_0 \cdot J\nabla_\xi \vec{v}_0\right) = 0 \tag{A.4a}$$

equivalent to

$$\frac{\partial v_j'}{\partial t} + v_i' \frac{\partial \xi_m}{\partial x_i} \frac{\partial \bar{v}_j}{\partial \xi_m} + (\bar{v}_i + \varepsilon v_i') \frac{\partial \xi_m}{\partial x_i} \frac{\partial v_j'}{\partial \xi_m} + \frac{1}{\bar{\varrho}} \left(1 - \varepsilon \frac{\varrho'}{\bar{\varrho}}\right) \left(\frac{\partial \xi_m}{\partial x_j} \frac{\partial p'}{\partial \xi_m} + \varrho'\, \bar{v}_i \frac{\partial \xi_m}{\partial x_i} \frac{\partial \bar{v}_j}{\partial \xi_m}\right) = 0 \tag{A.4b}$$

expands analogously in each direction:

A.2.1 Momentum in x-Direction in Expanded Form

$$\begin{aligned}
&\frac{\partial u'}{\partial t} + u' \left(\frac{\partial \bar{u}}{\partial \xi}\xi_x + \frac{\partial \bar{u}}{\partial \eta}\eta_x + \frac{\partial \bar{u}}{\partial \zeta}\zeta_x\right) + v' \left(\frac{\partial \bar{u}}{\partial \xi}\xi_y + \frac{\partial \bar{u}}{\partial \eta}\eta_y + \frac{\partial \bar{u}}{\partial \zeta}\zeta_y\right) + \\
&w' \left(\frac{\partial \bar{u}}{\partial \xi}\xi_z + \frac{\partial \bar{u}}{\partial \eta}\eta_z + \frac{\partial \bar{u}}{\partial \zeta}\zeta_z\right) + (\bar{u} + \varepsilon u') \left(\frac{\partial u'}{\partial \xi}\xi_x + \frac{\partial u'}{\partial \eta}\eta_x + \frac{\partial u'}{\partial \zeta}\zeta_x\right) + \\
&(\bar{v} + \varepsilon v') \left(\frac{\partial u'}{\partial \xi}\xi_y + \frac{\partial u'}{\partial \eta}\eta_y + \frac{\partial u'}{\partial \zeta}\zeta_y\right) + (\bar{w} + \varepsilon w') \left(\frac{\partial u'}{\partial \xi}\xi_z + \frac{\partial u'}{\partial \eta}\eta_z + \frac{\partial u'}{\partial \zeta}\zeta_z\right) + \\
&\frac{1}{\bar{\varrho}} \left(1 - \varepsilon \frac{\varrho'}{\bar{\varrho}}\right) \left\{ \frac{\partial p'}{\partial \xi}\xi_x + \frac{\partial p'}{\partial \eta}\eta_x + \frac{\partial p'}{\partial \zeta}\zeta_x + \varrho' \left[\bar{u} \left(\frac{\partial \bar{u}}{\partial \xi}\xi_x + \frac{\partial \bar{u}}{\partial \eta}\eta_x + \frac{\partial \bar{u}}{\partial \zeta}\zeta_x\right) + \right.\right. \\
&\left.\left. \bar{v} \left(\frac{\partial \bar{u}}{\partial \xi}\xi_y + \frac{\partial \bar{u}}{\partial \eta}\eta_y + \frac{\partial \bar{u}}{\partial \zeta}\zeta_y\right) + \bar{w} \left(\frac{\partial \bar{u}}{\partial \xi}\xi_z + \frac{\partial \bar{u}}{\partial \eta}\eta_z + \frac{\partial \bar{u}}{\partial \zeta}\zeta_z\right) \right] \right\} = 0
\end{aligned} \tag{A.5}$$

A.2.2 Momentum in y-Direction in Expanded Form

$$\begin{aligned}
&\frac{\partial v'}{\partial t} + u' \left(\frac{\partial \bar{v}}{\partial \xi}\xi_x + \frac{\partial \bar{v}}{\partial \eta}\eta_x + \frac{\partial \bar{v}}{\partial \zeta}\zeta_x\right) + v' \left(\frac{\partial \bar{v}}{\partial \xi}\xi_y + \frac{\partial \bar{v}}{\partial \eta}\eta_y + \frac{\partial \bar{v}}{\partial \zeta}\zeta_y\right) + \\
&w' \left(\frac{\partial \bar{v}}{\partial \xi}\xi_z + \frac{\partial \bar{v}}{\partial \eta}\eta_z + \frac{\partial \bar{v}}{\partial \zeta}\zeta_z\right) + (\bar{u} + \varepsilon u') \left(\frac{\partial v'}{\partial \xi}\xi_x + \frac{\partial v'}{\partial \eta}\eta_x + \frac{\partial v'}{\partial \zeta}\zeta_x\right) + \\
&(\bar{v} + \varepsilon v') \left(\frac{\partial v'}{\partial \xi}\xi_y + \frac{\partial v'}{\partial \eta}\eta_y + \frac{\partial v'}{\partial \zeta}\zeta_y\right) + (\bar{w} + \varepsilon w') \left(\frac{\partial v'}{\partial \xi}\xi_z + \frac{\partial v'}{\partial \eta}\eta_z + \frac{\partial v'}{\partial \zeta}\zeta_z\right) + \\
&\frac{1}{\bar{\varrho}} \left(1 - \varepsilon \frac{\varrho'}{\bar{\varrho}}\right) \left\{ \frac{\partial p'}{\partial \xi}\xi_y + \frac{\partial p'}{\partial \eta}\eta_y + \frac{\partial p'}{\partial \zeta}\zeta_y + \varrho' \left[\bar{u} \left(\frac{\partial \bar{v}}{\partial \xi}\xi_x + \frac{\partial \bar{v}}{\partial \eta}\eta_x + \frac{\partial \bar{v}}{\partial \zeta}\zeta_x\right) + \right.\right. \\
&\left.\left. \bar{v} \left(\frac{\partial \bar{v}}{\partial \xi}\xi_y + \frac{\partial \bar{v}}{\partial \eta}\eta_y + \frac{\partial \bar{v}}{\partial \zeta}\zeta_y\right) + \bar{w} \left(\frac{\partial \bar{v}}{\partial \xi}\xi_z + \frac{\partial \bar{v}}{\partial \eta}\eta_z + \frac{\partial \bar{v}}{\partial \zeta}\zeta_z\right) \right] \right\} = 0
\end{aligned} \tag{A.6}$$

A.2.3 Momentum in z-Direction in Expanded Form

$$\begin{aligned}
&\frac{\partial w'}{\partial t} + u'\left(\frac{\partial \bar{w}}{\partial \xi}\xi_x + \frac{\partial \bar{w}}{\partial \eta}\eta_x + \frac{\partial \bar{w}}{\partial \zeta}\zeta_x\right) + v'\left(\frac{\partial \bar{w}}{\partial \xi}\xi_y + \frac{\partial \bar{w}}{\partial \eta}\eta_y + \frac{\partial \bar{w}}{\partial \zeta}\zeta_y\right) + \\
&w'\left(\frac{\partial \bar{w}}{\partial \xi}\xi_z + \frac{\partial \bar{w}}{\partial \eta}\eta_z + \frac{\partial \bar{w}}{\partial \zeta}\zeta_z\right) + (\bar{u} + \varepsilon u')\left(\frac{\partial w'}{\partial \xi}\xi_x + \frac{\partial w'}{\partial \eta}\eta_x + \frac{\partial w'}{\partial \zeta}\zeta_x\right) + \\
&(\bar{v} + \varepsilon v')\left(\frac{\partial w'}{\partial \xi}\xi_y + \frac{\partial w'}{\partial \eta}\eta_y + \frac{\partial w'}{\partial \zeta}\zeta_y\right) + (\bar{w} + \varepsilon w')\left(\frac{\partial w'}{\partial \xi}\xi_z + \frac{\partial w'}{\partial \eta}\eta_z + \frac{\partial w'}{\partial \zeta}\zeta_z\right) + \\
&\frac{1}{\bar{\varrho}}\left(1 - \varepsilon\frac{\varrho'}{\bar{\varrho}}\right)\left\{\frac{\partial p'}{\partial \xi}\xi_z + \frac{\partial p'}{\partial \eta}\eta_z + \frac{\partial p'}{\partial \zeta}\zeta_z + \varrho'\left[\bar{u}\left(\frac{\partial \bar{w}}{\partial \xi}\xi_x + \frac{\partial \bar{w}}{\partial \eta}\eta_x + \frac{\partial \bar{w}}{\partial \zeta}\zeta_x\right) + \right.\right. \\
&\left.\left.\bar{v}\left(\frac{\partial \bar{w}}{\partial \xi}\xi_y + \frac{\partial \bar{w}}{\partial \eta}\eta_y + \frac{\partial \bar{w}}{\partial \zeta}\zeta_y\right) + \bar{w}\left(\frac{\partial \bar{w}}{\partial \xi}\xi_z + \frac{\partial \bar{w}}{\partial \eta}\eta_z + \frac{\partial \bar{w}}{\partial \zeta}\zeta_z\right)\right]\right\} = 0
\end{aligned} \tag{A.7}$$

A.3 Energy

$$\frac{\partial p'}{\partial t} + \vec{v}' \cdot J\nabla_\xi p_0 + (\vec{v}_0 + \varepsilon\vec{v}') \cdot J\nabla_\xi p' + \kappa\left[(J\nabla_\xi) \cdot \vec{v}_0\, p' + (J\nabla_\xi) \cdot \vec{v}'(p_0 + \varepsilon p')\right] = 0 \tag{A.8a}$$

equivalent to

$$\frac{\partial p'}{\partial t} + v_i'\frac{\partial \xi_m}{\partial x_i}\frac{\partial \bar{p}}{\partial \xi_m} + (\bar{v}_i + \varepsilon v_i')\frac{\partial \xi_m}{\partial x_i}\frac{\partial p'}{\partial \xi_m} + \kappa\left[\frac{\partial \xi_m}{\partial x_i}\frac{\partial \bar{p}}{\partial \xi_m}\, p' + \frac{\partial \xi_m}{\partial x_i}\frac{\partial v_i'}{\partial \xi_m}(\bar{p} + \varepsilon p')\right] = 0 \tag{A.8b}$$

expands to

$$\begin{aligned}
&\frac{\partial p'}{\partial t} + u'\left(\frac{\partial \bar{p}}{\partial \xi}\xi_x + \frac{\partial \bar{p}}{\partial \eta}\eta_x + \frac{\partial \bar{p}}{\partial \zeta}\zeta_x\right) + v'\left(\frac{\partial \bar{p}}{\partial \xi}\xi_y + \frac{\partial \bar{p}}{\partial \eta}\eta_y + \frac{\partial \bar{p}}{\partial \zeta}\zeta_y\right) + \\
&w'\left(\frac{\partial \bar{p}}{\partial \xi}\xi_z + \frac{\partial \bar{p}}{\partial \eta}\eta_z + \frac{\partial \bar{p}}{\partial \zeta}\zeta_z\right) + (\bar{u} + \varepsilon u')\left(\frac{\partial p'}{\partial \xi}\xi_x + \frac{\partial p'}{\partial \eta}\eta_x + \frac{\partial p'}{\partial \zeta}\zeta_x\right) + \\
&(\bar{v} + \varepsilon v')\left(\frac{\partial p'}{\partial \xi}\xi_y + \frac{\partial p'}{\partial \eta}\eta_y + \frac{\partial p'}{\partial \zeta}\zeta_y\right) + (\bar{w} + \varepsilon w')\left(\frac{\partial p'}{\partial \xi}\xi_z + \frac{\partial p'}{\partial \eta}\eta_z + \frac{\partial p'}{\partial \zeta}\zeta_z\right) + \\
&\kappa\left[p'\left(\frac{\partial \bar{u}}{\partial \xi}\xi_x + \frac{\partial \bar{u}}{\partial \eta}\eta_x + \frac{\partial \bar{u}}{\partial \zeta}\zeta_x + \frac{\partial \bar{v}}{\partial \xi}\xi_y + \frac{\partial \bar{v}}{\partial \eta}\eta_y + \frac{\partial \bar{v}}{\partial \zeta}\zeta_y + \frac{\partial \bar{w}}{\partial \xi}\xi_z + \frac{\partial \bar{w}}{\partial \eta}\eta_z + \frac{\partial \bar{w}}{\partial \zeta}\zeta_z\right) + \right. \\
&\left.(\bar{p} + \varepsilon p')\left(\frac{\partial u'}{\partial \xi}\xi_x + \frac{\partial u'}{\partial \eta}\eta_x + \frac{\partial u'}{\partial \zeta}\zeta_x + \frac{\partial v'}{\partial \xi}\xi_y + \frac{\partial v'}{\partial \eta}\eta_y + \frac{\partial v'}{\partial \zeta}\zeta_y + \frac{\partial w'}{\partial \xi}\xi_z + \frac{\partial w'}{\partial \eta}\eta_z + \frac{\partial w'}{\partial \zeta}\zeta_z\right)\right] = 0.
\end{aligned} \tag{A.9}$$

Appendix B

Consistency of the Filter Integral Approximation

To show the integral approximation to be fourth order accurate, rewrite (2.32) on page 22 using an averaged white-noise field $\langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle$ instead of $\mathcal{U}_{ij}$

$$\psi(\vec{x},t) = \sum_{i=1}^{i_{\max}} \sum_{j=1}^{j_{\max}} \int\limits_{\Delta A_{ij}} G(\vec{x},\vec{x}')\, \langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle \, \mathrm{d}\vec{x}' = \sum_{i=1}^{i_{\max}} \sum_{j=1}^{j_{\max}} \langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle \int\limits_{\Delta A_{ij}} G(\vec{x},\vec{x}')\, \mathrm{d}\vec{x}' \,. \tag{B.1}$$

Now, inserting a TAYLOR expansion of the kernel around $\vec{x}^{\mathrm{s}}_{ij}$, i. e. $G' = G_{ij} + \nabla G_{ij} \cdot \left(\vec{x}' - \vec{x}^{\mathrm{s}}_{ij}\right) + \mathcal{O}\left(h^2\right)$ (with $G_{ij} := G(\vec{x},\vec{x}^{\mathrm{s}}_{ij})$, $G' := G(\vec{x},\vec{x}')$, and $h^2 \propto \Delta A_{ij}$), and using the definition (2.43) on page 25, it follows

$$\psi(\vec{x},t) = \sum_{i=1}^{i_{\max}} \sum_{j=1}^{j_{\max}} G_{ij}(\vec{x})\, \langle\!\langle \mathcal{U}_{ij} \rangle\!\rangle \Delta A_{ij} + \mathcal{O}\left(h^4\right).$$

Appendix C

Convection Velocity of Random Particles

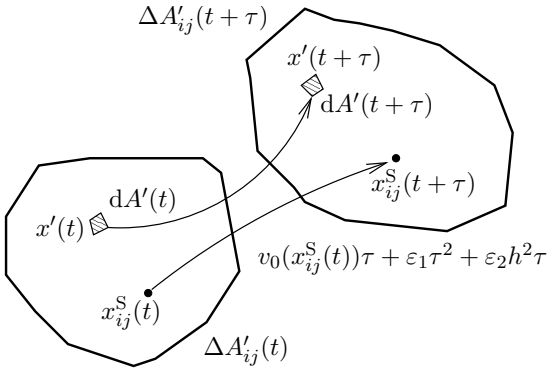


Figure C.1: Sketch of one drifting control volume $\Delta A'_{ij}$; ε_1 and ε_2 denote constants that limit the leading temporal and spatiotemporal error terms

Consider a small element $\mathrm{d}A'$ in subdomain $\Delta A'_{ij}$ at initial position $\vec{x}'(t)$ (cf. Figure C.1). Taylor expansion gives its position at later time $t+\tau$ with $\vec{x}'(t+\tau) = \vec{x}'(t) + {}^{\mathrm{d}\vec{x}'(t)}/_{\mathrm{d}t}\,\tau + \varepsilon_1\tau^2$. The element drifts with the mean-flow, i. e. ${}^{\mathrm{d}\vec{x}'}/_{\mathrm{d}t} = \vec{v}'_0$. The velocity $\vec{v}'_0$ at $\vec{x}'$ follows from a spatial Taylor expansion around the subdomain centre $\vec{x}^{\mathrm{s}}_{ij}$ at time level t: $\vec{v}'_0 = \vec{v}^{\mathrm{s}}_{ij} + \nabla\vec{v}^{\mathrm{s}}_{ij}\left(\vec{x}'(t) - \vec{x}^{\mathrm{s}}_{ij}(t)\right) + \varepsilon_2 h^2$ (with $\vec{v}^{\mathrm{s}}_{ij} := \vec{v}_0(\vec{x}^{\mathrm{s}}_{ij}(t))$. Then the cell centre location at $t+\tau$ follows by introducing

$$\vec{x}'(t+\tau) = \vec{x}'(t) + \vec{v}^{\mathrm{s}}_{ij}\tau + \nabla\vec{v}^{\mathrm{s}}_{ij}\left(\vec{x}'(t) - \vec{x}^{\mathrm{s}}_{ij}(t)\right)\tau + \varepsilon_1\tau^2 + \varepsilon_2 h^2\tau$$

into (2.43). Evaluation of the integral (in incompressible flow $\mathrm{d}A'$ and $\Delta A'_{ij}$ are invariants) and differentiation with respect to τ yields the convection velocity (2.47) on page 26 of the cell centre at time level t (i. e. at $\tau = 0$).

Appendix D

Coefficients for Spatial Discretization (DRP-Coefficients)

$c_{-3} = -0.020843142770$	$c_{-2} = 0.026369431$	$c_{-1} = -0.048230454$	$c_0 = 0.203876371$
$c_{-2} = 0.166705904415$	$c_{-1} = -0.166138533$	$c_0 = 0.281814650$	$c_1 = -1.128328861$
$c_{-1} = -0.770882380518$	$c_0 = 0.518484526$	$c_1 = -0.768949766$	$c_2 = 2.833498741$
$c_0 = 0.0$	$c_1 = -1.273274737$	$c_2 = 1.388928322$	$c_3 = -4.461567104$
$c_1 = 0.770882380518$	$c_2 = 0.474760914$	$c_3 = -2.147776050$	$c_4 = 5.108851915$
$c_2 = -0.166705904415$	$c_3 = 0.468840357$	$c_4 = 1.084875676$	$c_5 = -4.748611401$
$c_3 = 0.020843142770$	$c_4 = -0.049041958$	$c_5 = 0.209337622$	$c_6 = 2.192280339$

Table D.1: The coefficients for 7-point stencils used for spatial discretization

Index