Nektar++: Development of the Compressible Flow Solver for Large Scale Aeroacoustic Applications

Daniel Lindblad[a,*], João Isler[a], Margarida Moragues Ginard[b], Spencer J. Sherwin[a], Chris D. Cantwell[a]

[a] *Imperial College London, London, SW7 2AZ, United Kingdom*
[b] *Basque Center for Applied Mathematics, Bilbao, 48009, Spain*

Abstract

A recently developed computational framework for jet noise predictions is presented. The framework consists of two main components, focusing on source prediction and noise propagation. To compute the noise sources, the turbulent jet is simulated using the compressible flow solver implemented in the open-source spectral/hp element framework Nektar++, which solves the unfiltered Navier-Stokes equations on unstructured grids using the high-order discontinuous Galerkin method. This allows high-order accuracy to be achieved on unstructured grids, which in turn is important in order to accurately simulate industrially relevant geometries. For noise propagation, the Ffowcs Williams - Hawkings method is used to propagate the noise between the jet and the far-field. The paper provides a detailed description of the computational framework, including how the different components fit together and how to use them. To demonstrate the framework, two configurations of a single stream subsonic jet are considered. In the first configuration, the jet is

* Corresponding author. *E-mail address:* d.lindblad@imperial.ac.uk

treated in isolation, whereas in the second configuration, it is installed under a wing. The aerodynamic results for these two jets show strong agreement with experimental data, while some discrepancies are observed in the acoustic results, which are discussed. In addition to this, we demonstrate close to linear scaling beyond $100,000$ processors on the ARCHER2 supercomputer.
Keywords:
Jet Noise; Discontinuous Galerkin; Scaling; Large Eddy Simulation; Ffowcs Williams - Hawkings; Acoustics; Installation Noise.

1. Introduction

Many fluid flows of engineering interest generate noise, either directly, or indirectly through interactions with nearby solid surfaces. Examples of this include the flow around cooling fans in computers [1], the flow around A-pillars and rear view mirrors on cars and trucks [2], the flow over wind turbine blades [3], the flow around the undercarriages of high-speed trains [4], and the flows generated by different parts of an aircraft, including high-lift systems [5, 6], fan- and turbine blades [7, 8], landing gears [5, 6], and the propulsive jet [9, 10, 11]. This type of aerodynamically generated noise will often be perceived as unpleasant, and can in some cases even be harmful. Therefore, there is a widespread need to understand and minimize the underlying noise sources.

The study of aerodynamically generated noise, known as aeroacoustics, saw a great surge in the middle of the last century. In the beginning, the motivation for studying aerodynamically generated noise came mainly from the aircraft industry, where it was realized that jetliners equipped with the

newly invented turbojet engine would be too loud for widespread use. This led to an era of intense research both in the UK and the US, known as the first golden age of aeroacoustics [12]. An important result that came out of this era is that jet noise scales with the eight power of the jet velocity [13, 14], whereas the jet propulsive power only varies with the third power of the jet velocity. Accordingly, it is possible to reduce the jet noise by lowering the jet velocity and increasing the diameter of the jet. This led to the introduction of high-bypass ratio turbofan engines, which today are in widespread use in commercial aviation. For a comprehensive overview of the theory developed during the first golden age of aeroacoustics, the interested reader is referred to [15].

As computers became more and more powerful, numerical predictions of aerodynamically generated noise started to become possible. An early example of this is the propeller and rotorcraft prediction codes developed at NASA Langley [16, 17]. These codes did not explicitly resolve the flow field, but instead required input from correlations, simplified aerodynamic calculations, or experiments in order to estimate the loading on the blades, which in turn translates into noise sources in the Ffowcs Williams - Hawkings equation [18]. Later, in 1992, Sir Michael James Lighthill foresaw a second golden age of aeroacoustics, this time driven by direct computations of the aerodynamically generated noise sources using CFD [12]. This led to a new field of aeroacoustics, known as computational aeroacoustics (CAA). The progress of this field over the past three decades has been summarized in several review papers, see e.g. [19, 20]

In general, CAA shares a lot of similarities with CFD, including the

need for robust numerical methods and accurate turbulence models for high-Reynolds number flows. However, as pointed out by e.g. Tam [21], some of the challenges faced by CAA are often not encountered in CFD, including the need to accurately resolve the acoustic waves (whose amplitude often are orders of magnitude smaller than the amplitude of the hydrodynamic waves), the need to propagate the acoustic waves over very large distances without significant dissipation and dispersion errors, and the need to resolve unsteady phenomena over a wide frequency range. In addition to this, reflections of acoustic or hydrodynamic waves against an artificial boundary, e.g. an inlet or an outlet to the computational domain, can quickly contaminate the acoustic part of the solution, rendering the whole simulation useless [22]. Therefore, numerical methods that are tailored to the needs of CAA have been actively developed over the past decades. These include, but are not limited to, the dispersion relation preserving (DRP) finite difference schemes by Tam and Webb [23], the compact finite difference schemes by Lele [24], the prefactored compact finite difference schemes by Ashcroft and Zhang [25], the Navier-Stokes characteristic boundary conditions by Poinsot and Lele [26], the radiation boundary conditions by Tam and Dong [27], Dong [28], and the non-reflecting boundary conditions by Giles [29].

The spatial discretization schemes mentioned above were all developed for structured grids, and some of them rely on wide finite-difference stencils. Although highly successful, the limitation of these schemes to structured grids makes it hard to adopt them for more complex configurations. Unstructured finite-volume schemes, on the other hand, are typically limited to second-order accuracy in space. This raises the need for high-order unstructured

schemes in order to tackle complex configurations of industrial interest with optimal efficiency [20].

The development of high-order methods for unstructured grids has seen great progress over the past decades. Interestingly, much of this development has not been done with aeroacoustics in mind, but rather for the purpose of performing accurate direct numerical simulations (DNS) and large eddy simulations (LES) of turbulent flows. Among the many high-order schemes that have been developed, we find the discontinuous Galerkin (DG) method [30, 31], the flux reconstruction (FR) method [32], the spectral-difference (SD) method [33, 34, 35, 36], and the continuous Galerkin (CG) method [37].

In recent years, the aforementioned high-order methods have started to be applied for CAA, see, e.g., [38, 39, 40, 41, 42]. However, given the large number of possible applications for CAA mentioned earlier, the potential benefits of these methods remain largely unexplored. Therefore, there is a need to further develop and test these methods in the context of CAA.

The purpose of this paper is to document a recently developed computational framework for jet noise predictions based on the high-order DG method [40, 42]. This framework is based on a hybrid approach, in which the turbulent jet is first simulated using the open-source spectral/hp element framework Nektar++ [43, 44]. After this, the far-field noise is computed using the Antares library [45, 46, 47]. Antares solves Formulation 1A [48] of the Ffowcs Williams and Hawkings [18] equation in the time-domain using a source-time dominant algorithm.

The present paper focuses on describing the setup of the framework using

Nektar++ and Antares. In particular, in section 2, the pre-processing steps (section 2.1), numerical method (section 2.2), and the post-processing steps (section 2.3) are described in detail. In section 4, the framework is then applied to an isolated and installed jet (section 4.1), followed by a demonstration of the scaling properties of the compressible flow solver in Nektar++ (section 4.2). Finally, in section 5, some conclusions are drawn.

2. Methodology

2.1. Pre-Processing

In this section we will present the pre-processing steps that we use to set up the jet noise simulations. An overview of these steps, including how they are linked together, is provided in Fig. 1.

2.1.1. CAD Preparation

Similar to many other computer aided engineering (CAE) problems, the starting point for our simulations is a CAD description of the geometry that we want to analyze. Typically, this geometry contains some surfaces that are not relevant to the aeroacoustic analysis, such as small features that we do not wish to resolve and/or surfaces that are not in direct contact with the fluid. In addition to this, the outer boundaries of the computational domain are often missing from the CAD. Therefore, the first step in the pre-processing pipeline is to clean up the CAD and add additional surfaces that define the missing boundaries. To this end, we use the open-source software Gmsh [49], which provides access to the OpenCASCADE CAD kernel. The Gmsh software is in turn controlled through its Python API, which makes it very convenient to automate the CAD preparation steps.

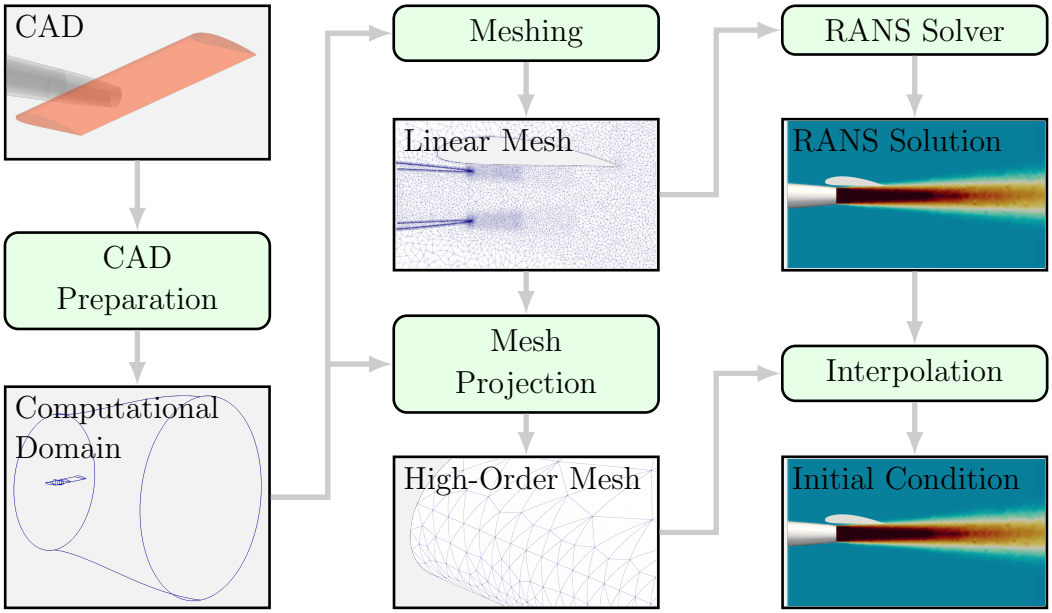


Figure 1: Flowchart illustrating the pre-processing setup.

2.1.2. Mesh Generation

The next step in the pre-processing pipeline is to generate a linear volume mesh. To this end, we use the commercial software STAR-CCM+ v2019.2. In particular, we use the *Advancing Layer Mesher* and the *Tetrahedral Mesher* in STAR-CCM+ to generate a single prism layer along solid walls, and tetrahedral elements in the rest of the domain, respectively.

The use of an unstructured grid has two important advantages. Firstly, it can be used to mesh complex geometries with relative ease, which is important in industrial applications. Secondly, it is straightforward to locally refine the mesh where it is needed, without having to refine other parts of the domain due to the constraints imposed by a structured grid. On the other

hand, when unstructured grids are used, it is very difficult to create high-quality anisotropic mesh elements, which could be beneficial in some areas, such as the early shear layer of the jet. However, recent work has found that it is beneficial to use isotropic mesh elements in the early shear layer for jet noise applications [50, 51]. Therefore, we believe that the current setup is well suited for our purposes. If, however, we would like to switch to a hybrid meshing strategy in the future, it is possible since Nektar++ also supports hexahedral and pyramid elements.

After the linear volume mesh has been generated, the `NekMesh` utility [44], which is part of the Nektar++ framework, is used to generate a high-order mesh with several prism elements in the wall-normal direction. To this end, the mesh is first projected onto the CAD by curving the mesh faces that are in contact with the boundary such that they conform to the CAD surface. After this, the curved prism elements are split in the wall-normal direction using an isoparametric approach [52].

If the mesh elements attached to the boundary are highly anisotropic and/or the underlying geometry is highly curved, it is possible that self-intersecting elements are created during the CAD-projection step. To minimize the risk of this happening, the aspect ratio of mesh elements adjacent to a boundary must be carefully controlled during the meshing process. In addition to this, the prism elements should be curved before they are split in the wall-normal direction. For complex geometries, however, it is often hard to completely eliminate the presence of invalid elements. To ensure that the simulations can still be run, the `NekMesh` utility therefore provides a module which can detect and remove the curvature from invalid elements. For

details on this, the interested reader is referred to the official documentation [53].

The output of the mesh projection step is a high-order mesh that conforms to the underlying CAD geometry. This mesh is written to an HDF5 file [54] using the layout described in [44]. The HDF5 format is used since this enables several processes to read parts of the mesh in parallel, which in turn is necessary to efficiently partition the mesh in massively parallel simulations.

We finally note that the `NekMesh` utility can read mesh files in both the `.ccm` format used by STAR-CCM+ and the `.msh` format used by Gmsh. Therefore, any meshing software that supports one of these file formats can be used in the pre-processing pipeline shown in Fig. 1.

2.1.3. Initial Conditions

In this work, we use the commercial software STAR-CCM+ v2019.2 to compute the initial condition for the LES simulations. More precisely, STAR-CCM+ is used to solve the Reynolds Averaged Navier Stokes (RANS) equations using the $k-\omega$ SST turbulence model. For the RANS simulations, we use the same mesh topology as for the LES simulations. However, since STAR-CCM+ uses a finite volume discretization, whereas Nektar++ uses a high-order discontinuous Galerkin discretization, we typically use a finer mesh for the RANS simulations.

In order to interpolate the RANS solution onto the high-order expansion used in the LES simulations, the RANS solution is first interpolated to the mesh vertices and written to a `.csv` file. After this, we use the `FieldConvert` utility, which is part of the Nektar++ framework, to interpolate the point cloud defined by the `.csv` file onto the polynomial expan-

sion used in the LES simulations. More precisely, the field values are first interpolated onto the quadrature points inside each element, followed by an elemental projection in which the point values are converted to the corresponding modal coefficients. Since the RANS solution is relatively smooth, and because it only represents the initial conditions for the LES simulations, we typically project it onto a linear polynomial basis ($p = 1$). More details about how to use the interpolation routines in `FieldConvert` can be found in the official documentation [53].

The interpolated solution is then written to a HDF5 file [54] using the layout described in [44]. The HDF5 format is used to export the high-order solution for the same reason that it is used to export the mesh, namely to enable parallel I/O where each process only reads a subset of the solution during initialization of the solver.

It is important to emphasize that there are other options for defining initial conditions to the simulation. In particular, if another RANS solver is available, it can easily be plugged into the pre-processing pipeline shown in Fig. 1 as long as it can export the solution in the `.csv` format. In addition to this, Nektar++ allows the user to define the initial conditions using analytical expressions. Note that, in this case, there is no need to write the initial conditions to file. Instead, the initial conditions can be added directly to the input files for Nektar++. For more details on this, the interested reader is referred to the official documentation [53].

2.2. Numerical Method

The flow is modeled by the compressible Navier-Stokes equations written in conservative form

$$\frac{\partial \boldsymbol{q}}{\partial t} + \frac{\partial}{\partial x_i}(\boldsymbol{f}_i^c(\boldsymbol{q}) - \boldsymbol{f}_i^v(\boldsymbol{q}, \boldsymbol{\nabla q})) = 0, \tag{1}$$

where $\boldsymbol{q} = (\rho, \rho u_1, \rho u_2, \rho u_3, E)^T$ is the vector of conservative variables, $\boldsymbol{f}_i^c(\boldsymbol{q})$ is the convective flux in the i^{th} direction and $\boldsymbol{f}_i^v(\boldsymbol{q}, \boldsymbol{\nabla q})$ is the diffusive flux in the i^{th} direction. In this work, we assume that the gas is calorically perfect and obeys the ideal law. In addition to this, we assume that the viscosity is constant and equal to the far-field value outside the jet. This assumption is valid since we only consider cold jets whose temperature is close to the ambient temperature.

2.2.1. Spatial Discretization

The DG method is used to discretize Eq. (1) on a fully unstructured mesh [55]. The diffusion terms are treated with the incomplete Interior Penalty (IP) method [56] and the advection terms are rewritten using a standard weak DG scheme, followed by the application of Roe's approximate Riemann solver to compute the convective flux across the element boundaries [57, 58]. A detailed explanation of how to define the spatial discretization in Nektar++ is provided in the official documentation [53].

In traditional LES, subgrid-scale models are used to model the coupling between the resolved and the unresolved turbulent scales. However, by using the Roe Riemann solver for coupling between adjacent elements, the solution is inherently diffused by the numerical discretization introduced by the upwinding. This dissipation tends to smooth out high-frequency features

and suppress small-scale turbulent fluctuations. As a result, the unresolved turbulent scales are implicitly handled through the dissipation introduced by the discretization scheme. Previous work has shown that the DG discretization in combination with Roe's approximate Riemann solver is well suited for implicit Large Eddy Simulations (iLES) [59, 60, 61, 62, 63, 64, 65].

In order to stabilize the solution in regions where the resolution is not high enough, an artificial viscosity approach is used. This stabilization strategy works by augmenting the dynamic viscosity and the thermal conductivity by an artificial viscosity and thermal conductivity in under-resolved regions. The artificial viscosity and thermal conductivity are respectively defined as

$$\mu_{av} = \mu_0 \rho \frac{h}{P}(c + \sqrt{u_k u_k})S, \tag{2}$$

$$\kappa_{av} = \mu_{av} \frac{C_p}{Pr}, \tag{3}$$

where μ_0 controls the magnitude of μ_{av}, S is the sensor, C_p is the specific heat and Pr is the Prandtl number.

A sensor is used to make sure that the artificial viscosity and thermal conductivity are only activated in under-resolved regions. The sensor used in this work is a modal resolution based sensor [66]. To compute this sensor, the following resolution indicator is first computed.

$$s_e = \log_{10}\left(\frac{\langle u - \hat{u}, u - \hat{u}\rangle}{\langle u, u\rangle}\right), \tag{4}$$

where $\langle \cdot, \cdot \rangle$ is the L_2 inner product, u is the full expansion of a state variable (density in our case) of order P. Lastly, $\hat{u}$ is the truncated expansion with terms up to $P-1$. Based on the resolution indicator, the sensor is computed

as

$$S = \begin{cases} 0 & s_e < s_0 - \kappa, \\ \frac{1}{2}\left(1 + \sin\left(\frac{\pi(s_e - s_0)}{2\kappa}\right)\right) & |s_e - s_0| \leq \kappa, \\ 1 & s_e > s_0 - \kappa, \end{cases} \tag{5}$$

where $s_0 = s_k - 4.25\log_{10}(P)$. Note that, the parameters s_κ and κ have to be chosen in the correct range in order to make the artificial viscosity effective.

2.2.2. Temporal Discretization

The set of ordinary differential equations (ODEs) obtained from the spatial discretization are integrated in time using a second-order singly diagonally implicit multi-stage Runge–Kutta (SDIRK2) method [67]. The nonlinear system from SDIRK2 is iteratively solved by the Jacobian-free Newton Krylov (JFNK) method [55]. Thus, the Newton method is used to solve the nonlinear system [68]. The restarted generalized minimal residual method (GMRES) [69] is adopted to solve the linear system. A Jacobian-free method is applied in order to avoid explicitly calculating and storing the Jacobian matrix [68].

In addition, a preconditioner in GMRES is employed which is important for solving stiff problems. The parameters used to set up the preconditioner are shown in listing 1, where `PreconMatFreezNumb` specifies number of iteration that the GMRES solver will perform before the preconditioner matrix be updated, `NonlinIterTolRelativeL2` is the convergence tolerance of the nonlinear system relative to the initial nonlinear system residual, `LinSysRelativeTolInNonlin` represents the convergence tolerance of the linear system solver in each nonlinear iteration. `PreconItsStep` de-

termines the number of preconditioning iterations to calculate the preconditioned vector. For further understanding of the mathematical formulation of these parameters, the reader may find useful the work of Yan et al. [55].

It is worth mentioning that the parameters shown in Listing 1 have a strong impact on the convergence and efficiency of the implicit solver, and consequently speed up the simulation. Special attention should be given to the `PreconMatFreezNumb` parameter, since if its value is too low the preconditioner matrix may be updated in every time-step, which reduces the efficiency. However, large values may lead to divergence since the preconditioner matrix is not updated as frequently as the numerical simulation requires, making the preconditioner less effective at every time step. The `PreconItsStep` parameter may significantly reduce the number of GMRES and nonlinear iterations when its value is increased; the time per time-step increases but it allows for larger time steps which speeds up the simulation overall. Therefore, the appropriate balance of these two parameters is essential for the efficient execution of the numerical simulation.

```
<SOLVERINFO>
    <!-- This part is put inside the <CONDITIONS> tag -->
    <P> PreconMatFreezNumb        = 1000    </P>
    <P> NonlinIterTolRelativeL2   = 1.0E-3  </P>
    <P> LinSysRelativeTolInNonlin = 5.0E-2  </P>
    <P> PreconItsStep             = 3      </P>
</SOLVERINFO>
```

Listing 1: Parameters for the preconditioner.

2.2.3. Boundary Conditions

Boundary conditions can have a large impact on the stability and accuracy of a simulation. In LES simulations of jet noise, three types of boundaries

are typically present; solid walls, far-field boundaries, and inlet(s) to the nozzle. In this section we will describe the implementation of these boundary conditions in detail.

The far-field boundaries are used to impose the ambient conditions of the flow, far away from the jet. The way the ambient conditions are imposed depends on whether the flow enters or exits the domain though the boundary. Along the parts of the far-field boundary where the flow enters the domain, the far-field state is imposed using the *Weak-Riemann* approach described in [57]. In this approach, the inviscid flux across boundary faces is computed from the solution of a Riemann problem, using the far-field state and the internal solution as the "left" and "right" state, respectively. This boundary condition is non-reflective for normally incident waves as long as the Riemann solver correctly differentiates between incoming and outgoing waves. The Riemann solver used in this work approximately satisfies this condition. Despite not being perfectly non-reflective, the amount of reflections against this boundary is deemed to be negligible since the far-field boundary is placed far away from the jet, and because the mesh is coarsened as it approaches the far-field boundary.

At the outlet part of the far-field boundary, the *Weak-Riemann* approach is also used. However, in contrast to the inlet part of the boundary, only the ambient pressure is imposed. This is done by setting the left state in the Riemann problem to be the internal solution, but with the pressure substituted for the ambient pressure [57]. This boundary condition is reflective. Therefore, it is important that all waves are attenuated before they reach this boundary. If only mesh coarsening is used for this purpose, an excessively

long computational domain would need to be used. To avoid this, we add a sponge zone upstream of the outlet [70]. In this zone, the right hand side of Eq. (1) is augmented with a damping term given by

$$... = -\sigma(\boldsymbol{x})\left(\boldsymbol{q} - \overline{\boldsymbol{q}}\right). \tag{6}$$

Here, $\sigma(\boldsymbol{x})$ is a function that controls the magnitude of the damping and $\overline{\boldsymbol{q}}$ is a steady reference solution. In this work, the RANS solution that we use to initialize the simulations is also used to define $\overline{\boldsymbol{q}}$.

As explained in [22], the best results are obtained when the function $\sigma(\boldsymbol{x})$ is increased slowly over a sufficiently long distance. To achieve this, we compute it as

$$\sigma(\boldsymbol{x}) = \begin{cases} 0 & x_1 < x_{s_1}, \\ \sigma_{\max} S\left(\frac{x_1 - x_{s_1}}{x_{s_2} - x_{s_1}}\right) & x_1 \in (x_{s_1}, x_{s_2}), \\ \sigma_{\max} & x_1 > x_{s_2}. \end{cases} \tag{7}$$

Here, (x_{s_1}, x_{s_2}) denotes the interval where the sponge is increased (the jet is aligned with the x_1-axis), $\sigma_{\max}$ is the maximum value of the damping coefficient, and $S(\xi)$ is the smooth-step function

$$S(\xi) = 6\xi^5 - 15\xi^4 + 10\xi^3, \quad \xi \in (0, 1). \tag{8}$$

The smooth-step function is shown in Fig. 2. As can be seen from this figure, it increases smoothly from 0 to 1 over the interval $(0, 1)$.

The sponge zone is activated in Nektar++ by adding the commands shown in Listing 2 and Listing 3 under the `<CONDITIONS>` tag and the

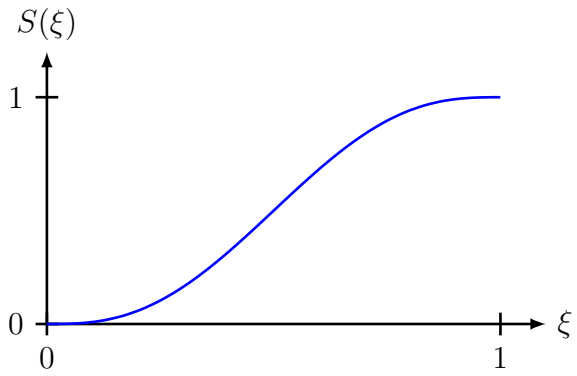


Figure 2: Smooth-step function.

<FORCING> tag in the input file, respectively. Note that these commands require that the parameters `x1`, `x2`, and `sigma_max` have been defined under the <PARAMETERS> tag in the input file.

```
<!-- This part is put inside the <CONDITIONS> tag -->
<FUNCTION NAME="AverageSolution">
    <F VAR="rho,rhou,rhov,rhow,E" FILE="avg.fld" />
</FUNCTION>

<FUNCTION NAME="DampingCoeff">
    <E VAR="rho,rhou,rhov,rhow,E" VALUE="-sigma_max*((x>x1)*(x<x2)*(6*((x-x1)/(x2
    -x1))^5 - 15*((x-x1)/(x2-x1))^4 + 10*((x-x1)/(x2-x1))^3) + (x>=x2))" />
</FUNCTION>
```

Listing 2: Definition of average solution and damping coefficient.

```
<!-- This part is put inside the <FORCING> tag -->
<FORCE TYPE="Absorption">
    <REFFLOW> AverageSolution </REFFLOW>
    <COEFF>   DampingCoeff    </COEFF>
</FORCE>
```

Listing 3: Definition of sponge zone.

The length of the sponge zone is set to $20D_j$, where D_j is the diameter of the jet. The damping coefficient, which has a unit equal to s^{-1}, is then selected such that it is proportional to the reciprocal of the residence time of a convected disturbance inside the sponge zone. For the simulations presented in this work, we found that a value of

$$\sigma_{\max} = 10\frac{U_j}{x_{s_2} - x_{s_1}}, \tag{9}$$

where U_j is the velocity of the jet at the nozzle exit, worked well.

At the inlet to the nozzle, we impose the stagnation pressure, stagnation temperature, and the flow direction. This boundary condition is chosen because in most experiments, including the ones used to validate the simulations performed in this work, the stagnation conditions are known upstream of the nozzle exit. The flow direction, on the other hand, is typically not known. Therefore, we set the flow direction equal to the boundary normal direction.

The inlet boundary condition is also imposed weakly through the Riemann solver. To this end, the 4 variables imposed by the boundary condition must be combined with one variable from the interior in order to form a complete left state for the Riemann solver. Depending on which interior variable is "extrapolated", the flow will develop in a different way from a given initial condition. For the simulations presented in this paper, it was found that extrapolating the magnitude of the velocity gave good stability. More precisely, the left state in the Riemann solver is computed as follows. First, the temperature at the boundary is computed from the imposed stagnation temperature and the magnitude of the velocity taken from the interior, following

the definition of the stagnation temperature

$$T_{\mathrm{BC}} = T_{0,\mathrm{BC}} - \frac{1}{2C_p} u_{\mathrm{R}}^2. \tag{10}$$

Here, $u_{\mathrm{R}} = \sqrt{u^2 + v^2 + w^2}$ is the magnitude of the velocity obtained from the interior solution and C_p is the specific heat capacity. After this, the density is computed using the well known isentropic flow relation

$$\rho_{\mathrm{BC}} = \rho_{0,\mathrm{BC}} \left(\frac{T_{\mathrm{BC}}}{T_{0,\mathrm{BC}}} \right)^{\frac{1}{\gamma - 1}}, \tag{11}$$

where R is the specific gas constant, γ the ratio of specific heats, and $\rho_{0,\mathrm{BC}} = \frac{p_{0,\mathrm{BC}}}{RT_{0,\mathrm{BC}}}$ is the stagnation density. Finally, the left state at the boundary is defined as

$$\boldsymbol{q}_{\mathrm{L}} = \begin{bmatrix} \rho_{\mathrm{BC}} \\ \rho_{\mathrm{BC}} u_{\mathrm{R}} \boldsymbol{n} \\ \frac{p_{\mathrm{BC}}}{\gamma - 1} + \frac{1}{2} \rho_{\mathrm{BC}} u_{\mathrm{R}}^2 \end{bmatrix}. \tag{12}$$

Here, $\boldsymbol{n} = (n_1, n_2, n_3)$ is the imposed flow direction and $p_{\mathrm{BC}} = \rho_{\mathrm{BC}} R T_{\mathrm{BC}}$ is the static pressure.

To use the inflow boundary condition described above, the commands shown in Listing 4 are added under the `<BOUNDARYCONDITIONS>` tag in the input file. In this listing, `rho0` is the stagnation density, (`n1`, `n2`, `n3`) define the flow direction, and `E0` $= p_{0,\mathrm{BC}}/(\gamma - 1)$ is the stagnation total energy. These parameters must be defined inside the `<PARAMETERS>` tag in the input file.

```
<!-- This part is put inside the <BOUNDARYCONDITIONS> tag -->
<REGION REF="ID">
    <D VAR="rho"  USERDEFINEDTYPE="StagnationInflow" VALUE="rho0" />
```

```
    <D VAR="rhou" USERDEFINEDTYPE="StagnationInflow" VALUE="n1" />
    <D VAR="rhov" USERDEFINEDTYPE="StagnationInflow" VALUE="n2" />
    <D VAR="rhow" USERDEFINEDTYPE="StagnationInflow" VALUE="n3" />
    <D VAR="E"    USERDEFINEDTYPE="StagnationInflow" VALUE="E0" />
</REGION>
```

Listing 4: Definition of stagnation inflow boundary condition.

To save computational resources, we only include a short part of the nozzle in our simulations. As a result, the boundary layer does not have enough time to develop between the inlet and the end of the nozzle. To overcome this issue, we impose non-uniform profiles of stagnation temperature and pressure at the inlet. This can be done by passing a file containing the boundary expansion of ρ_0 and E_0 to the boundary condition. Alternatively, analytical expressions describing the spatial variation of ρ_0 and E_0 can be used instead of the constant values shown in Listing 4.

To construct a file containing the boundary expansion of ρ_0 and E_0, we start by extracting the mesh elements defining the boundary using the `extract` module in `NekMesh` (see section 4 in the official documentation for details [53]). After this, the resulting mesh file is edited to ensure that the boundary expansion defined in the file uses the same polynomial order as the one that will be used in the simulations. An example of this is for a mesh containing quad and triangular elements is provided in Listing 5.

```
<!-- This part is put inside the <NEKTAR> tag -->
<EXPANSIONS>
    <E COMPOSITE="C[0]" TYPE="MODIFIED" NUMMODES="2" FIELDS="rho,E" />
    <E COMPOSITE="C[1]" TYPE="MODIFIED" NUMMODES="2" FIELDS="rho,E" />
</EXPANSIONS>
```

Listing 5: Expansion definition for an inlet containing quad and triangular elements.

After the expansion definition has been edited, the quadrature points are exported to a `.csv` file using `FieldConvert`. By default, `FieldConvert` will export data on a uniform set of points for visualization purposes. However, since we want to project data on the quadrature points back onto the polynomial expansion, we need to export the location of the quadrature points instead. To do this, the flag `--noequispaced` should be passed to `FieldConvert`. After this, we use Python to compute the values of ρ_0 and E_0 at the location of the quadrature points. Note that, if an analytical expression is used for this purpose, it is simpler to specify this expression directly in the input file. In this work, however, we specify the profiles of ρ_0 and E_0 using experimental data. More precisely, we use `SciPy` [71] to interpolate the experimental data to the location of the quadrature points. After this, we write the coordinates of the quadrature points together with the associated values of ρ_0 and E_0 to a new `.csv` file. A new field file containing the boundary expansions of ρ_0 and E_0 can then be obtained using the `pointdatatofld` module in `FieldConvert`.

The implementation of the solid wall-boundary condition used for the nozzle walls is described in detail in [57].

2.3. Post-Processing

In this section we describe the different tools that we use to post process the solution. In particular, the technique used to propagate the noise between the jet and the far-field observers is described in section 2.3.2. It does in turn depend on the data collection strategy described in section 2.3.1.

2.3.1. Data Collection

Nektar++ provides a set of filters that can be used to compute derived quantities from the solution. In this work, we make extensive use of the history points filter to probe the solution at a set of predefined points. By default, the history point filter writes the data to a `.csv` file every N^{th} time step. If a small number of points is used, this is usually not a problem. However, as more and more points are added, the I/O quickly starts to consume a significant portion of the overall execution time. In addition to this, the `.csv` format can not be used to store complex data structures, it requires a relatively large amount of memory per data point, and it does not allow the user to easily add additional metadata to the file. To overcome these limitations, the history point filter in Nektar++ was extended to store the points in the HDF5 format [54].

The HDF5 format is a flexible data format that allows the user to design the layout of the file based on the needs of a particular application. To this end, the HDF5 group provides the HDF5 library, which gives the user access to a rich API that can be used to create, structure, and read/write large amounts of data to the file.

The file structure chosen to store the history points is illustrated in Fig. 3. The folders shown in this figure represent HDF5 groups. These groups are synonymous to folders in a file system, in the sense that they can be used to organize data inside the file. Starting from the root group, which we call `NEKTAR`, two sub-groups are added. The first is called `COORDINATES` which, as the name suggests, contains the coordinates of history points stored as three HDF5 datasets. The second sub-group below the root group is called

TIME-DATA. Each time a new time step is written to the file, a new sub-group called TIME-STEP-N is added to the TIME-DATA group, where N is an integer which is incremented by one each time a new time step is added. Inside this sub group, five HDF5 datasets are created, one per solution variable. In addition to this, the current time is added as a HDF5 attribute.

The HDF5 format is well suited for post-processing since many popular high-level programming languages, such as Python, Julia, and MATLAB, can be used to read/write HDF5 files. Support for this functionality is usually provided either directly, or through high-quality third-party packages. In this work, we have used the h5py package to post-process the history point data in Python. One example of this is provided in the next section.

2.3.2. Far-Field Noise Predictions

The overall goal of this work is to predict the far-field noise of turbulent jets. Usually, the location where we want to measure the noise is located outside the computational domain. Therefore, a separate method is needed to propagate the noise between the jet and the far-field. In this work, we use Antares 1.17.0 [45, 46, 47] for this purpose.

Antares solves the Ffowcs Williams - Hawkings equation [18] in the time domain using a source-time dominant algorithm. The solution to the Ffowcs Williams - Hawkings equation expresses the pressure signal at an arbitrary observer location as an integral over a closed surface, plus a volume integral over the volume outside the surface. In this work, we don't include the volume integral in the solution to the Ffowcs Williams - Hawkings equation. This simplification is widely used in jet noise applications [10, 11, 72], and is valid as long as the integration surface is chosen such that it encloses

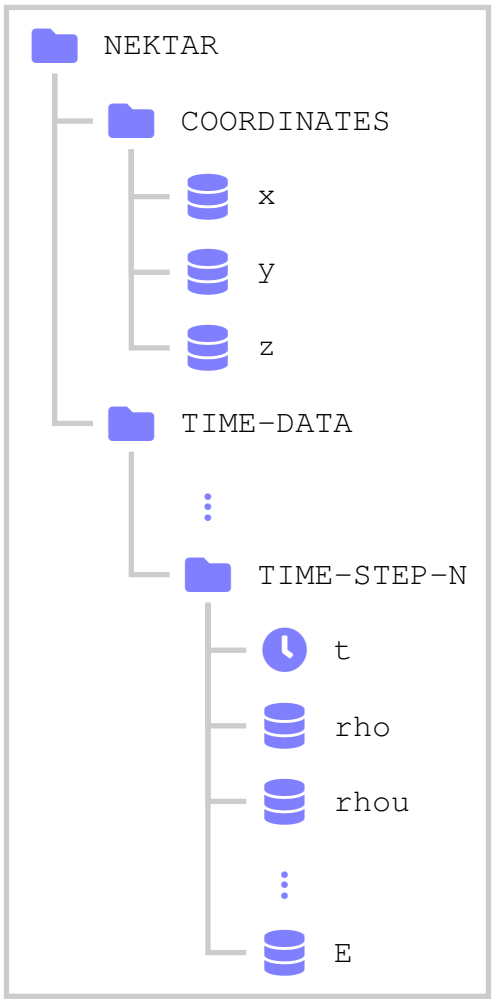


Figure 3: Layout of an HDF5 file storing history point data.

all relevant noise sources [73], and such that no entropy or vorticity waves cross the surface [74, 75]. To satisfy these conditions, a conical surface that follows the spreading rate of the jet is usually chosen [11]. Ideally, this surface should be placed as close to the jet as possible in order to minimize numerical dissipation and dispersion of the acoustic waves as they propagate between the source and the surface. However, if the surface is too narrow, the conditions stated above will be violated. Therefore, there is an optimal width of the integration surface [10, 11].

In addition to adjusting the width, the length of the surface will influence the accuracy of the results. Typically, a length of 30 nozzle diameters is considered enough to enclose all relevant noise sources [72]. Unfortunately, the entropy and vorticity waves generated in the jet do not decay over this distance. Therefore, if the surface is closed at the downstream end, the second condition stated above will be violated. As a result, spurious noise is generated by the "end-cap" of the surface, which degrades the quality of the results. One way to solve this problem is to extend the domain and the integration surface much further, but since this also increases the computational cost, it is not desirable. Instead, alternative strategies such as the method of end-caps [74], the pressure-formulation of the Ffowcs Williams - Hawkings equation [74, 75], and the additional surface terms developed by Rahier et al. [76], have been proposed. Unfortunately, the method of end-caps and the additional surface terms proposed by Rahier et al. [76] are not supported by Antares. In addition to this, the pressure formulation was found to not improve the results for the cold jets considered in this work [40]. Therefore, a fourth strategy, in which the integration surface is left open at

the downstream end, is adopted instead. The benefit of this method is that it eliminates spurious noise from the end-cap. However, this comes at the cost of less accurate noise predictions for the lowest frequencies, see [74] for details.

At present, Antares provides two solutions to the Ffowcs Williams - Hawkings equation, Formulation 1A by Farassat [48] and Formulation 1C by Najafi-Yazdi et al. [77]. The main difference between these two formulations is that the latter includes the effect of a flight stream on noise propagation. In this work, we only consider jets operating in a medium at rest. Therefore, Formulation 1A is used.

To compute the noise propagation, we need the time history of the solution on the integration surface. The process that we use to generate this time history is illustrated in Fig. 4. It is important to point out that this process is relatively generic, and will therefore work with any post-processing software that implements the Ffwocs Williams - Hawkings method, as long as this software can read one of the file formats that Nektar++ exports.

To begin with, the integration surface is constructed in Gmsh [49] using the OpenCASCADE CAD kernel. After this, Gmsh is also used to create a triangular surface mesh. The mesh resolution on the integration surface is typically higher than the volume mesh. The reason for this is that Antares does not support Gaussian quadrature on high-order elements. Instead, it stores the data on the mesh vertices, and then compute the element contribution as the average over all vertex values, scaled by the area of the element.

To sample data on the integration surface, we use the history point filter

described in section 2.3.1. We start by converting the mesh generated with Gmsh into the format used by Nektar++ using the `NekMesh` utility. By default, `NekMesh` adds an XML tag to the mesh file that defines which type of polynomial expansion, and which quadrature rule, to use for each composite (group of mesh elements). Since Antares does not support high-order elements, we edit the default expansion definition such that linear finite elements are used. An example of this is provided in listing 6.

```
<!-- This part is put inside the <NEKTAR> tag -->
<EXPANSIONS>
    <E COMPOSITE="C[0]" TYPE="MODIFIED" NUMMODES="2" FIELDS="u" />
</EXPANSIONS>
```

Listing 6: Expansion definition for the Ffowcs Williams - Hawkings integration surface.

After the expansion definition has been edited, the quadrature points are exported to a `.csv` file using the `FieldConvert` utility. As explained in section 2.2.3, `FieldConvert` exports data on a uniform set of points instead of the quadrature points by default. For a linear finite element expansion, these points are simply the vertices of the elements. Although this is the data that we eventually want to export to Antares, we do not want to sample the solution at these points for reasons that will become clear soon. Therefore, the flag `--noequispaced` is passed to `FieldConvert` when the points are exported to the `.csv` file. The points defined in the `.csv` file are then used by the history point filter to sample the solution at the integration surface.

Once the simulation is finished, we need to convert the history point data into a format that Antares can read. To this end, the Python script shown in listing 7 is used. This script works by looping through all time steps stored

in the HDF5 file using the `h5py` package. For each time step, the script converts the solution to primitive variables, and stores the result in a temporary `.csv` file. The `.csv` file is then used by the process module called `pointdatatofld`, which projects data on the Gaussian quadrature points onto an existing expansion. Here, the existing expansion is stored in the `field` variable in the script, which in turn is obtained by reading the mesh file described earlier. Once the projection is completed, the updated expansion, which now contains 5 fields (one for each primitive variable) is written to a new file in the TecPlot binary format. Since the flag `--noequispaced` is not passed to `FieldConvert` this time, the solution will be exported on the mesh vertices instead of the quadrature points. Note that all these operations are performed by directly calling the underlying Nektar++ functions through the `NekPy` module, which is provided together with Nektar++.

```
from NekPy.FieldUtils import Field, InputModule, ProcessModule, OutputModule
import h5py
import numpy as np

# Name of HDF5 file
hdf5_filename = "fwh-data.hdf5"

# Name of output file (in tecplot format)
plt_filename = os.path.splitext("fwh-data.plt")

# Name of mesh file (including expansion definition)
mesh_filename = "fwh-surface.xml"

# Open HDF5 file
with h5py.File(hdf5_filename, 'r') as hdf5_file:

    # Read coordinates
    cgroup = hdf5_file["NEKTAR/COORDINATES"]
    x = cgroup["x"][()]
```

```
    y = cgroup["y"][()]
    z = cgroup["z"][()]

    # Go through all time steps
    tgroup = hdf5_file["NEKTAR/TIME-DATA"]
    for indx, tstep in enumerate(tgroup.values()):

        # Read solution at this time step
        rho  = tstep['rho'][()]
        rhou = tstep['rhou'][()]
        rhov = tstep['rhov'][()]
        rhow = tstep['rhow'][()]
        E    = tstep['E'][()]

        # Convert to primitive variables
        u = rhou / rho
        v = rhov / rho
        w = rhow / rho
        p = (E - 0.5 * (rhou**2 + rhov**2 + rhow**2)/rho) * 0.4

        # Save data in .csv format
        np.savetxt("tmp.csv", np.column_stack((x,y,z,rho,u,v,w,p)), delimiter=","
    , header="x, y, z, rho, u, v, w, p")

        # Create a Nektar++ Field and read expansion from file
        field = Field([])
        InputModule.Create("xml", field, mesh_filename).Run()

        # Project points to expansion
        ProcessModule.Create("pointdatatofld", field, frompts="tmp.csv").Run()

        # Save data to .plt format
        output_filename = plt_filename[0] + "-%05i"%(index) + plt_filename[1]
        OutputModule.Create("plt", field, output_filename).Run()
```

Listing 7: Python script for converting history point data into TecPlot binary format.

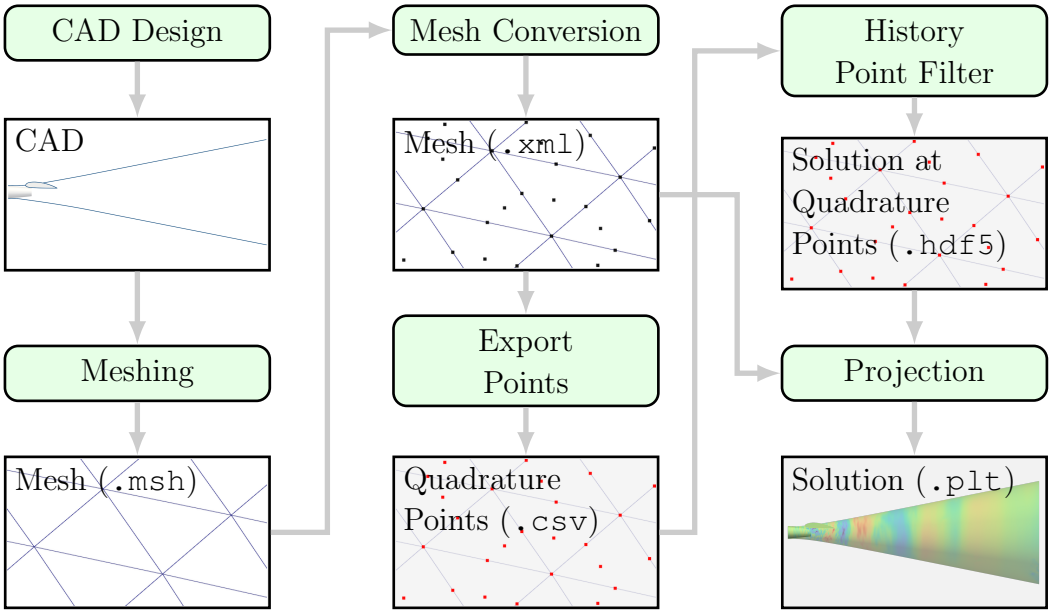


Figure 4: Flowchart illustrating the steps required to sample data on the integration surface used for the Ffowcs Williams - Hawkings method.

3. Performance considerations

Parallelization. Parallelization in Nektar++ is carried out using a domain decomposition strategy, where the computational mesh is divided into one subset per process. Mesh partitioning is achieved using either the Scotch or the METIS libraries at the beginning of the simulation. Note that each processor only reads the geometric information that is required for its own partition, that is, the partitioning of the mesh file is done in parallel. Parallel mesh partitioning, contrary to serial mesh partitioning, avoids one single thread reading the whole mesh and partitioning it alone which could lead to the node running out of memory. This is achieved through the

use of HDF5 routines when reading the mesh datasets and the use of the `--use-hdf5-node-comm` flag with Nektar++. Output solution files can also be written in HDF5 format which is enabled by the flag `--io-format Hdf5`. The different processes communicate through the standard MPI protocol, where each process is independently executed and there is no explicit use of shared memory protocols such as OpenMP. In this work we use the Scotch library to partition the mesh.

Vectorization. The limitation in performance of modern hardware concerns memory bandwidth speeds more than processor clock speeds. For this reason, the use of arithmetically intense schemes, that is, schemes performing a high number of floating point operations for each byte transferred from memory, continues to be of considerable interest. At high polynomial orders, high-order or spectral/hp element methods are arithmetically intense schemes since they involve dense, compact kernels for key finite element operators. This is a significant advantage of high-order methods compared to lower-order methods. Key aspects to exploit this potential for increased performance are matrix-free implementations of finite element operators combined with the use of sum-factorization and effective approaches for exploiting single-instruction multiple-data (SIMD) vectorization. Nektar++ is provided with the matrix-free evaluation of basic finite element operations which still utilize sum-factorization even for non-tensor-product elements (triangles, tetrahedra, and prisms) [78, 79, 80]. Nektar++ also allows explicit exploitation of SIMD vectorization, showing that although performance is naturally degraded when going from hexahedra to triangles, tetrahedra, and prismatic elements, efficient implementations are still obtained, achieving 50% to 70%

of the peak floating-point throughput of modern processors with both AVX2 and AVX512 instruction sets [81]. In addition to that, the SIMD vectorization of the diffusion operator and the block diagonal matrix obtained after decoupling the jacobian matrix to calculate the preconditioned vector has recently been implemented in Nektar++. Also, the $(\hat{\boldsymbol{L}} + \hat{\boldsymbol{U}})$ matrix, which is also obtained from the jacobian matrix, employed to calculate the preconditioned vector is now able to use the auto-tuning routines (e.g. matrix free operator) to optimize this operator. For further details about the preconditiner calculation the reader is referring to [55]

Automatic selection of the operations optimization. An auto-tuning capability, to automatically select the most efficient operations optimization for each execution, is also available in Nektar++ [80]. It can be turned on by including the lines, given in Listing 8, in the problem setting script.

```
<NEKTAR>
    <!-- This is how to enable auto tuning -->
    <COLLECTIONS DEFAULT="auto"/>
</NEKTAR>
```

Listing 8: Flag for enabling performance auto-tuning in Nektar++.

For more details on this, the interested reader is referred to the official documentation [53].

4. Results

4.1. Isolated and Installed Jet Noise

In this section we will present the application of the computational framework presented in this paper to investigating noise generated by isolated and installed jets.

4.1.1. Case Description

The nozzle considered in this work has an inner diameter of $D_j = 40$mm, a convergence half-angle of 2.44°, and a total length of $19D_j$. Both an isolated and an installed configuration are considered. In the installed configuration, an uniform-chord (2D) NACA4415 airfoil at 4° angle of attack is placed close to the nozzle. The axial and vertical distance between the nozzle exit and the airfoil trailing edge are $L = 3D_j$ and $H = 0.6D_j$, respectively. The chord and span of the airfoil are $3.75D_j$ and $15D_j$, respectively. A schematic view of the installed configuration is shown in Fig. 5.

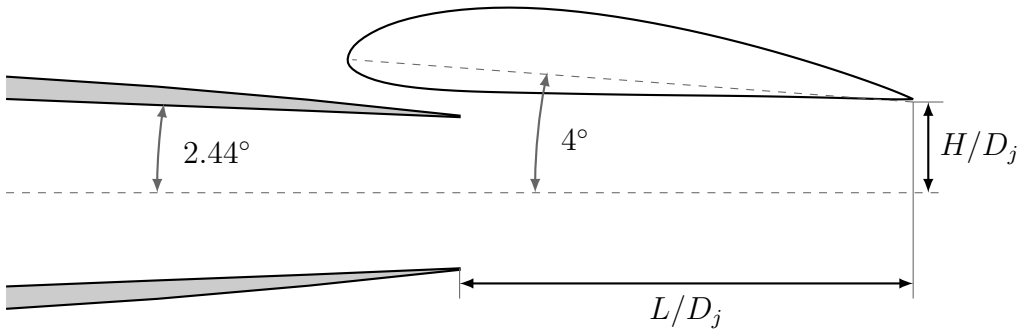


Figure 5: Schematic view of the nozzle installed under the wing.

In this work, we consider a single operating point of the nozzle, corresponding to an acoustic Mach number of $M_a = U_j/c_\infty = 0.6$, a Reynolds number of $Re_j = \rho_\infty U_j D_j/\mu_\infty = 5.5 \cdot 10^5$, and a static temperature ratio of $T_j/T_\infty = 0.9335$. In addition to this, no flight stream is considered, meaning that the ambient medium is at rest.

Both the isolated and installed configuration of the nozzle have recently been tested in the Doak Laboratory Flight Jet Rig, located at the University

of Southampton. During these tests, both aerodynamic data (velocity measurements in the jet plume) and far-field acoustic data was recorded. This data will be used in sections 4.1.3 and 4.1.4 to validate the simulations. More details about the experimental setup can be found in [40, 42, 82].

4.1.2. Computational Setup

A schematic view of the axi-symmetric, funnel-shaped computational domain used in this work is presented in Fig. 6. Along the left and top boundary shown in this figure, the far-field state is imposed using the *Weak-Riemann* approached described in section 2.2.3. As indicated by the arrows in the figure, a small co-flow corresponding to 2% of the jet exit velocity is also added to the (stagnant) far-field state. This is done to ensure that vortical structures generated in the jet are flushed out of the domain and to facilitate flow entrainment.

Along the right boundary in Fig. 6, the ambient pressure is imposed according to the method described in section 2.2.3. Since this boundary condition is reflective, a sponge zone of length $20D_j$ is also added upstream of the outlet boundary. In this region, the damping term described in section 2.2.3 is used to attenuate the jet before it reaches the outflow boundary.

The inlet to the nozzle is placed $1.5D_j$ upstream of the nozzle exit. At this boundary, non-uniform profiles of stagnation pressure and temperature are imposed to ensure that the mean velocity matches the experimental data as closely as possible at the nozzle exit, see section 2.2.3 for details about the implementation.

In addition to matching the mean velocity profile at the nozzle exit, recent work has demonstrated the importance of getting the turbulence levels

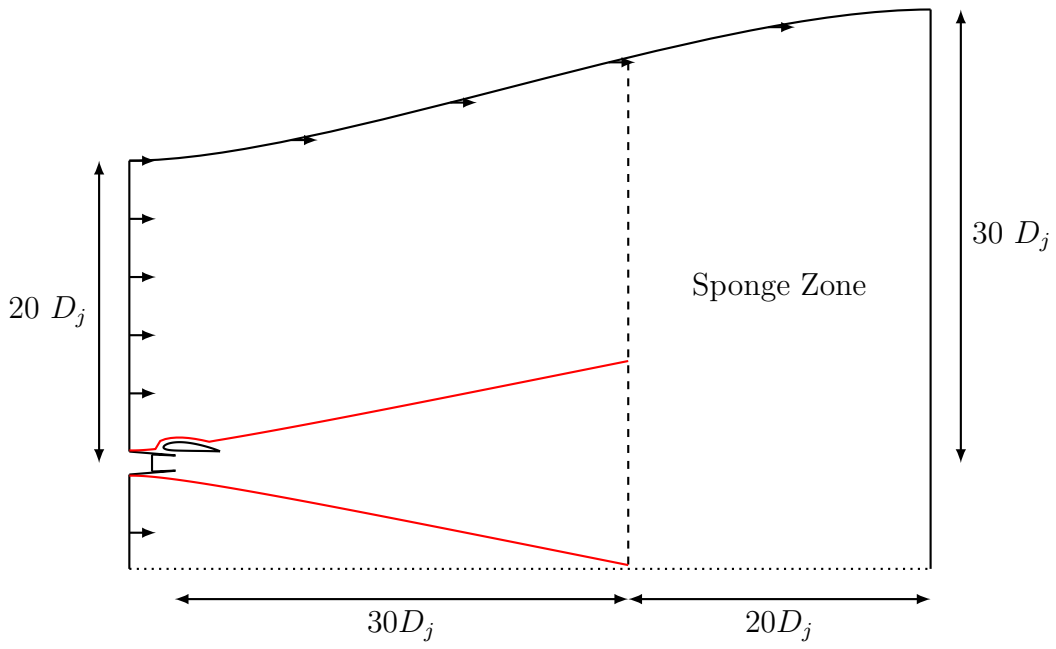
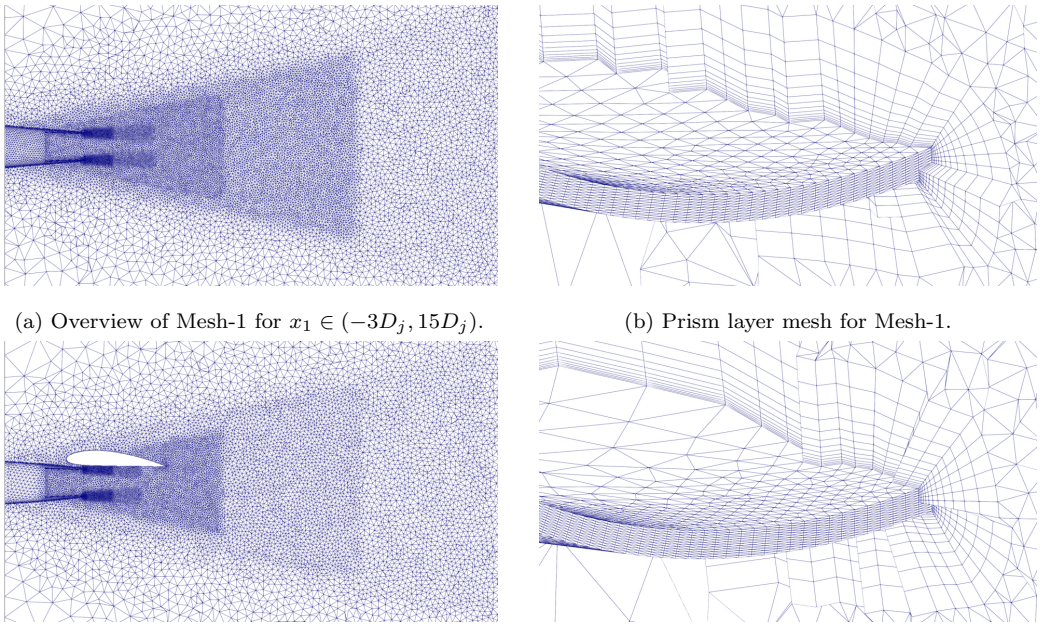


Figure 6: Schematic view of the computational domain and the Ffowcs Williams - Hawkings integration surface.

in the boundary layer right [50, 51, 83, 84]. In order to satisfy this condition, some type of artificial tripping mechanism and/or injection of synthetic turbulence at the inlet boundary may be necessary when only a short part of the nozzle is included in the simulations. In addition to this, for high Reynolds number applications, it is usually necessary to model the inner part of the boundary layer, either through the use of a wall model [50] or a hybrid RANS-LES model [51, 85]. Unfortunately, Nektar++ currently does not support any tripping mechanism, synthetic turbulence injection, or wall model. Therefore, a standard no-slip boundary condition is used to model the nozzle instead. In earlier work, we noted that the turbulence levels at the nozzle exit are not accurately predicted with this setup [40, 42]. It is well known that the state of the boundary at the nozzle exit can have a large influence on the far-field acoustic results [50, 83, 84]. Therefore, a better wall modeling strategy is expected to improve the results, but was unfortunately outside the scope of this work.

The methodology described in section 2.1.2 was used to generate the two unstructured meshes shown in Fig 7. As can be seen from this figure, the main difference between these two meshes is that the latter includes the wing. In addition to this, the resolution far downstream is slightly lower for the second mesh. The resulting mesh count for the two meshes are summarized in Table 1. More details about the two meshes are provided in [42].

Three simulations have been performed, one for the isolated jet, and two for the installed jet. The simulation of the isolated jet was performed with Mesh-1 using a polynomial degree of $P = 2$. The two simulations of the installed jet were performed with Mesh-2, using a polynomial degree of



(a) Overview of Mesh-1 for $x_1 \in (-3D_j, 15D_j)$.

(b) Prism layer mesh for Mesh-1.

(c) Overview of Mesh-2 for $x_1 \in (-3D_j, 15D_j)$.

(d) Prism layer mesh for Mesh-2.

Figure 7: Illustration of unstructured grids defined in Table 1.

Table 1: Mesh settings.

Name	Configuration	N_{prism}	N_{tet}
Mesh-1	Isolated	$4.36 \cdot 10^5$	$6.18 \cdot 10^6$
Mesh-2	Installed	$4.63 \cdot 10^5$	$4.54 \cdot 10^6$

Table 2: Simulation settings. Δt_{jet} and $\Delta t_{\text{FW-H}}$ denote the sampling rate in the jet plume and on the Ffowcs Williams - Hawkings integration surface, respectively. τ_{init} and τ_{sample} denote the time the simulation is run before sampling and during sampling, respectively.

Name	P	Mesh	DOF	Initialize from	$\frac{\Delta t c_\infty}{D_j}$	$\frac{\Delta t_{\text{jet}} c_\infty}{D_j}$	$\frac{\Delta t_{\text{FW-H}} c_\infty}{D_j}$	$\frac{\tau_{\text{init}} c_\infty}{D_j}$	$\frac{\tau_{\text{sample}} c_\infty}{D_j}$
Isolated-1	2	Mesh-1	$70 \cdot 10^6$	RANS	0.002	0.04	0.04	350	320
Installed-1	2	Mesh-2	$54 \cdot 10^6$	RANS	0.002	0.04	0.04	450	900
Installed-2	3	Mesh-2	$109 \cdot 10^6$	Installed-1	0.002	0.04	0.04	300	450

$P = 2$ and $P = 3$, respectively. A summary of the settings used for each simulation is presented in Table 2.

4.1.3. Aerodynamic Results

We start by considering the mean and RMS of the axial velocity in the jet plume. The simulation results obtained for the isolated jet are compared against experimental data in Figs. 8a and 8b. As can be seen from these figures, the agreement is quite satisfactory for all radial and axial locations. The most notable discrepancies are found close to the jet centerline and further downstream. In this region, the mean velocity and turbulence levels are over-predicted and under-predicted by the simulation, respectively. From earlier work, we know that the state of the boundary layer at the nozzle exit is not perfectly predicted in our simulations due to insufficient resolution and lack of a wall-model [42]. Since the state of the boundary layer affects the development of the shear layer, it is possible that the under-resolved boundary layer contributes to the discrepancies seen far downstream in Figs.. 8a and 8b. Another possible source of the discrepancies is that we only include a short part of the conical nozzle in the simulations, which in turn should

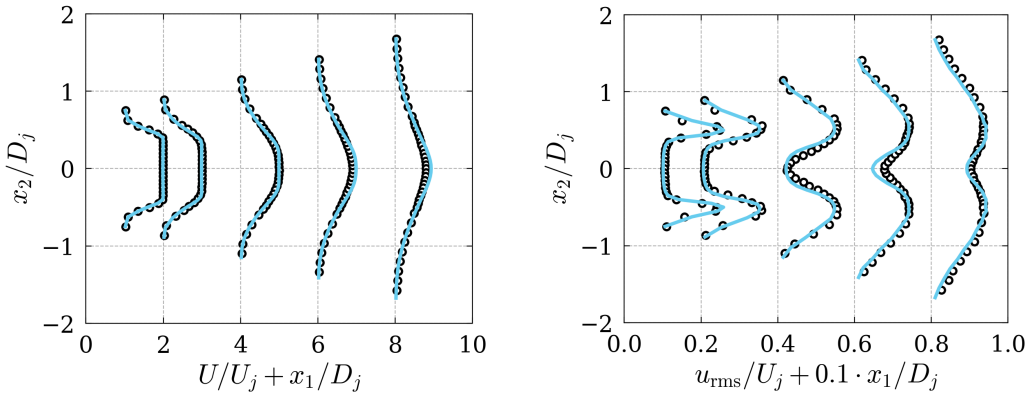
lead to slight discrepancies in the radial velocity component at the nozzle exit. This is explained in more detail in [42].

The simulation results obtained for the installed nozzle are compared against the corresponding experimental data in Fig. 8. In general, we can note that the agreement between simulations and experiments is quite good for the installed nozzle as well. It is also interesting to note that the higher polynomial degree used in the simulation called "Installed-2" only has a minor impact on the flow statistics. Finally, we note that the discrepancies between the simulations and the experiments is consistent with those observed for the isolated nozzle. This is expected, considering that the simulation setup is almost identical for the two configurations.

4.1.4. Far-Field Acoustic Results

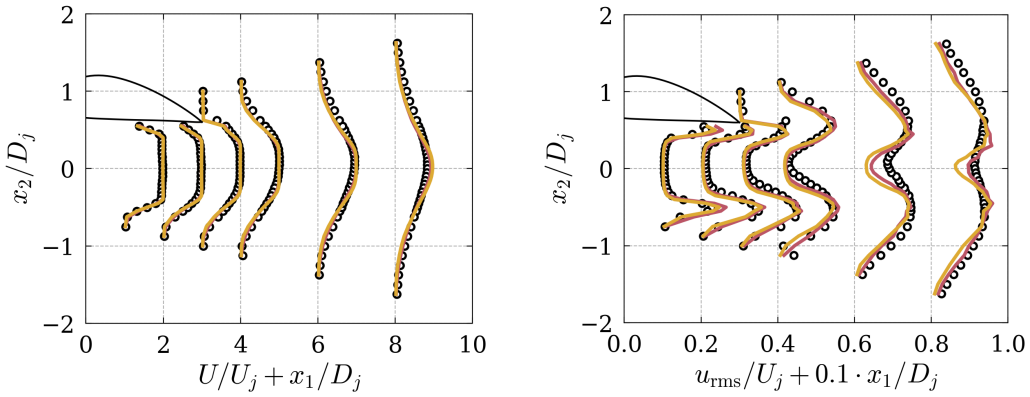
As explained in section 2.3.2, we use the Ffowcs Williams - Hawkings method implemented in Antares 1.17.0. to compute the far-field noise. The output of Antares is the pressure signal at the same microphone locations as were used in the experiments. The time-series signal is then converted into a power spectral density (PSD) in units [dB/St] using the implementation of Welch's method [86] in SciPy [71]. More details on the exact locations of the microphones and how the PSD is computed can be found in [40, 42].

The PSD at two representative microphone locations located at $\theta = 90°$ and $43°$ with regards to the downstream jet axis is presented in Fig. 9a and 9b, respectively. By comparing the results in these two figures, it is clear that the simulations agree better with the experiments for the $\theta = 43°$ angle. This is expected since jet noise peaks at shallower angles [87]. From Fig. 9b, we can also see that the effect of the wing is very small at $\theta = 43°$. In contrast,



(a) Mean axial velocity for the isolated jet.

(b) RMS of axial velocity for the isolated jet.



(c) Mean axial velocity for the installed jet.

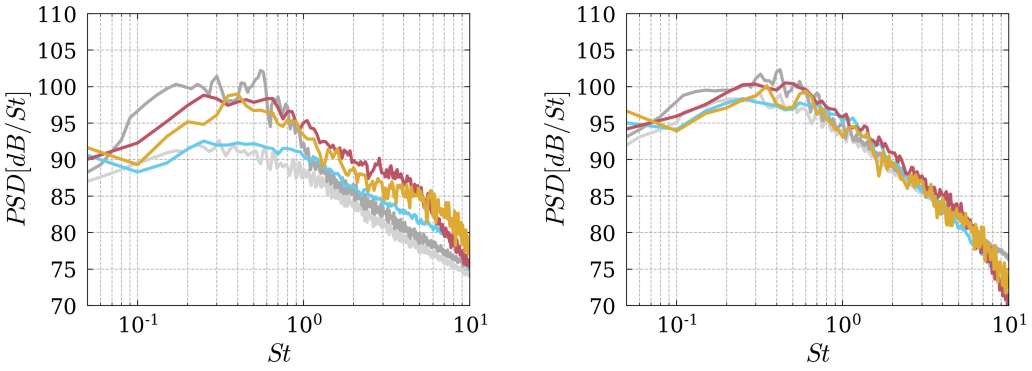
(d) RMS of axial velocity for the installed jet.

Figure 8: Velocity statistics in jet plume for the isolated and the installed jet. Isolated-1 (——), Installed-1 (——), Installed-2 (——), Experiments (○).

the far-field noise levels are considerably higher for the installed nozzle at the $\theta = 90°$ angle, especially for $St < 2$ (Fig. 9a). These results are explained by the fact that jet installation noise has a dipole directivity that peaks in the $\theta = 90°$ and $270°$ direction [88].

Figures 9a and 9b show that the simulation of the isolated nozzle tends to over-predict the corresponding experimental results, especially for the higher frequencies. We believe these discrepancies can be partly explained by the under-resolved boundary layer inside the nozzle. The resolution (mesh and polynomial degree) might also be too low for the isolated nozzle. In fact, if we compare the spectra obtained with the $P = 2$ and $P = 3$ simulation of the installed nozzle, we see that the simulation with the higher polynomial degree (Installed-2) gives lower noise levels for higher frequencies. Since jet installation noise mostly contributes to the lower frequencies, it is likely that the noise levels for the isolated jet would come down further if $P = 3$ was used. Unfortunately, this turned out to be outside the scope of this paper.

Despite the improvement seen when going from $P = 2$ to $P = 3$ for the installed nozzle, there are still some notable discrepancies in the far-field spectra for the highest and lowest frequencies. The discrepancies seen for the highest frequencies are likely caused by the same error sources as for the isolated nozzle, e.g., insufficient modeling of the boundary layer and resolution in the jet plume. Also, since the microphones are located below the wing, the errors at the highest frequencies are amplified due to reflections against the pressure side of the wing. With regards to the lowest frequencies, on the other hand, the effect of the resolution along the wing span and size of the Ffowcs Williams - Hawkings integration surface could be potential



(a) $\theta = 90°$.

(b) $\theta = 43°$.

Figure 9: Power Spectral Density in the far-field. Isolated-1 (——), Installed-1 (——), Installed-2 (——), Experiments for isolated jet (——), Experiments for installed jet (——).

sources of error. The effect of these should be investigated in future work.

4.2. Scaling and performance

The scalability of Nektar++ has been demonstrated on a number of parallel computers. In this case, strong scaling results of the compressible flow solver in Nektar++ on ARCHER2 and JUWELS supercomputers are presented. The scaling plots are generated based on the time-integration computational time (whole application except initialization and I/O). This approach is taken since the majority of time is spent on time-integration in a production run.

The DG method and 2nd order implicit time integration scheme introduced in sections 2.2.1 and 2.2.2, respectively, are considered for the scaling. The installed round nozzle case presented in sections 4.1.1 to 4.1.3 is used. This test has been run for 100 time steps on a mesh of approximately 5 mil-

lion elements. Two polynomial orders, $P = 2$ and $P = 3$, have been tested. In total, this case contains approximately 54 million DOF for $P = 2$ and 109 million DOF for $P = 3$.

The scaling tests on ARCHER2 and JUWELS are presented in Fig. 10. Tables including the timings of interest on ARCHER2 and JUWELS are also presented in Table 3 and Table 4, respectively. It can be seen from these figures that the scalings are done between 2,560 and 140,800 cores on ARCHER2 and between 2,400 and 28,800 cores on JUWELS. ARCHER2 has 128 cores (2x AMD Zen2 Rome EPYC 7742 CPUs and 64 cores per CPU) and 256 GB per node, while JUWELS has 48 cores (2x Intel Xeon Platinum 8168 Skylake CPUs and 24 cores per CPU) and 96 GB per node. We believe that the higher memory per node on ARCHER2 allowed us to use a higher number of nodes compared to JUWELS. In Fig. 10 we observe a pronounced super-linear scaling on ARCHER2 probably due to a larger number of cache misses for small job sizes. This probably means that Nektar++ is not cache optimized [89]. We can conclude from the scaling results that the optimal number of DOF per core is between 650 and 850. Below that, parallel efficiency decreases.

5. Conclusions

In this paper, we have presented a computational framework for accurately predicting jet noise. Our framework utilizes the spectral*hp* approach of Nektar++ to simulate turbulent jets. To compute the noise propagation from the jet vicinity to the far-field, we employed the Ffowcs Williams - Hawkings method, which is implemented in the Antares library. The primary

Table 3: Time-integration CPU time of the scaling runs done on ARCHER2.

Num. nodes	Num. cores	CPU time P=2	CPU time P=3
20	2560	1309.62	3083.04
50	6400	410.658	1066.50
100	12800	191.250	496.383
150	19200	123.885	308.756
200	25600	91.6463	221.697
250	32000	78.5870	176.408
300	38400	60.2471	140.109
350	44800	52.2270	120.575
400	51200	44.9938	105.134
450	57600	40.8790	95.1388
500	64000	37.6927	84.2975
550	70400	32.3210	-
600	76800	31.6789	70.1267
650	83200	29.4066	-
700	89600	30.5700	61.5184
800	102400	31.6817	52.5133
850	108800	-	49.4919
900	115200	-	47.9693
1000	128000	-	43.7558
1100	140800	-	53.4228

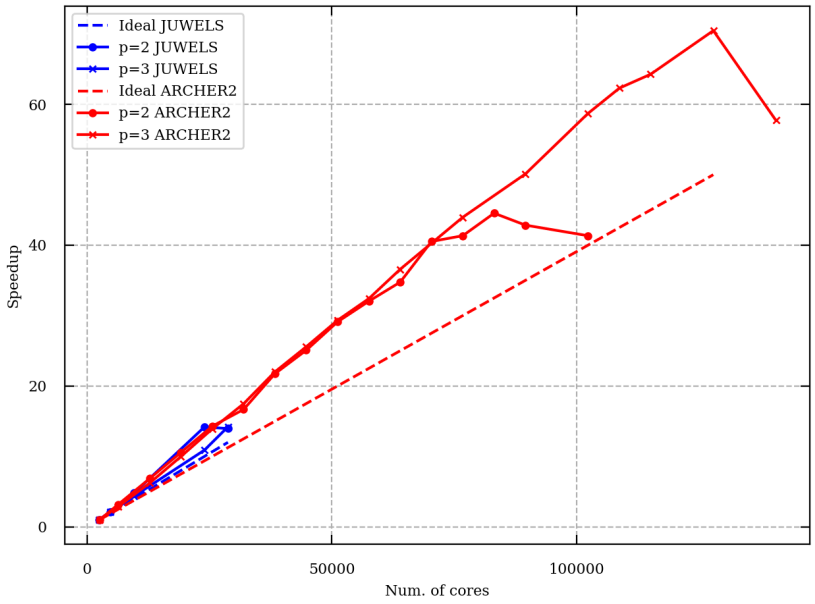


Figure 10: Strong scaling for the installed round nozzle case on ARCHER2, between 2,560 and 140,800 cores, and on JUWELS, between 2,400 and 28,800 cores, for polynomial orders $P = 2$ and $P = 3$.

objective of our work is to provide a comprehensive description of the computational framework developed, highlighting the integration of Nektar++ with Antares to enable turbulent jet simulations and reliable predictions of jet noise.

To enable high-order simulations for turbulent jets and noise predictions, several important methodological aspects were discussed. Firstly, the generation of high-order meshes that accurately conform to the underlying CAD geometry was described. Additionally, the importance of establishing compatible initial and boundary conditions for high-order simulations was highlighted. In terms of post-processing, we presented a data acquisition strategy focused on probing the solution. As part of this strategy, we extended the

Table 4: Time-integration CPU time of the scaling runs done on JUWELS.

Num. nodes	Num. cores	CPU time P=2	CPU time P=3
50	2400	919.817	1811.14
100	4800	435.883	870.604
200	9600	192.249	416.621
500	24000	64.8598	166.230
600	28800	65.9233	127.540

history point filter in Nektar++ to store points in HDF5 format. Finally, we addressed the process of generating the time history required for computing noise propagation using Antares.

To showcase the computational framework, two configurations of a single stream subsonic jet were computed. The first configuration focuses on an isolated jet nozzle, while the second configuration incorporates a NACA4415 airfoil positioned near the nozzle exit. The numerical simulations were compared with experimental data provided by the University of Southampton. The numerical aerodynamic results demonstrated very good agreement with the experimental data. Small discrepancies were observed in the centreline of the jet and further downstream. These discrepancies may be explained by the under-resolved boundary layer and the inclusion of only a short part of the conical nozzle. In terms of far-field results, some discrepancies were observed between the simulations and the experiments. In particular, the simulations tend to over-predict the corresponding experimental results for the isolated nozzle, mainly for the highest frequencies. For the installed nozzle, discrepancies in the far-field spectra were observed for the highest and

lowest frequencies. For the highest frequencies, the discrepancies were attributed to the insufficient modeling of the boundary layer and resolution in the jet plume in both cases. On the other hand, the discrepancies in the lowest frequencies for the installed nozzle could potentially be caused by a relatively low resolution along the wing span and size of the Ffowcs Williams - Hawkings integration surface around the wing. These source errors will be investigated in future work.

Recent developments in computational performance have resulted in significant speed up for highly computationally intensive simulations. One noteworthy development is the use of SIMD vectorization to calculate the diffusion operator and preconditioner within the compressible implicit solver in Nektar++. Furthermore, the scalability of the Nektar++ framework has been demonstrated on ARCHER2 and JUWELS supercomputers. The scaling results indicate linear scaling up to more than 100,000 cores for the ARCHER2 supercomputer.

Acknowledgements

This work was supported by the European Union's H2020 program under the DJINN (Decrease Jet INstallation Noise) project, Grant Agreement No. 861438, and the Marie Skłodowska-Curie individual fellowship, Grant Agreement No. 842536. Computational resources have been provided by the Partnership for Advanced Computing in Europe (PRACE) on the JUWELS cluster. This work also used the ARCHER2 UK National Supercomputing Service (https://www.archer2.ac.uk). The authors would like to acknowledge the Antares development team at CERFACS for providing the Antares library.

References

[1] L. Huang, Characterizing computer cooling fan noise, The Journal of the Acoustical Society of America 114 (2003) 3189–3200. doi:10.1121/1.1624074.

[2] N. Oettle, D. Sims-Williams, Automotive aeroacoustics: An overview, Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 231 (2017) 1177–1189. doi:10.1177/0954407017695147.

[3] H. H. Hubbard, K. P. Shepherd, Aeroacoustics of large wind turbines, The Journal of the Acoustical Society of America 89 (1991) 2495–2508. doi:10.1121/1.401021.

[4] G. Minelli, H.-D. Yao, N. Andersson, D. Lindblad, J. Forssén, P. Höstmad, S. Krajnović, Using horizontal sonic crystals to reduce the aeroacosutic signature of a simplified ice3 train model, Applied Acoustics 172 (2021). doi:10.1016/j.apacoust.2020.107597.

[5] N. Molin, Airframe noise modeling and prediction, CEAS Aeronautical Journal 10 (2019) 11–29. doi:10.1007/s13272-019-00375-4.

[6] W. Dobrzynski, Almost 40 years of airframe noise research: What did we achieve?, Journal of Aircraft 47 (2010) 353–367. doi:10.2514/1.44457.

[7] E. Envia, A. G. Wilson, D. L. Huff, Fan noise: A challenge to caa, International Journal of Computational Fluid Dynamics 18 (2004) 471–480. doi:10.1080/10618560410001673489.

[8] N. Peake, A. B. Parry, Modern challenges facing turbomachinery aeroacoustics, Annual Review of Fluid Mechanics 44 (2012) 227–248. doi:10.1146/annurev-fluid-120710-101231.

[9] D. J. Bodony, S. K. Lele, Current status of jet noise predictions using large-eddy simulation, AIAA Journal 46 (2008) 364–380. doi:10.2514/1.24475.

[10] G. A. Brès, S. K. Lele, Modelling of jet noise: a perspective from large-eddy simulations, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 377 (2019) 20190081. doi:10.1098/rsta.2019.0081.

[11] A. S. Lyrintzis, M. Coderoni, Overview of the use of large-eddy simulations in jet aeroacoustics, AIAA Journal 58 (2020) 1620–1638. doi:10.2514/1.J058498.

[12] M. J. Lighthill, Report on the Final Panel Discussion on Computational Aeroacoustics, ICASE Report No. 92-53, NASA, Hampton, VA, 1992.

[13] M. J. Lighthill, On sound generated aerodynamically. i. general theory, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 211 (1952) 564–587. doi:10.1098/rspa.1952.0060.

[14] M. J. Lighthill, On sound generated aerodynamically. ii. turbulence as a source of sound, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 222 (1954) 1–32. doi:10.1098/rspa.1954.0049.

[15] D. Crighton, Basic principles of aerodynamic noise generation, Progress in Aerospace Sciences 16 (1975) 31–96. doi:10.1016/0376-0421(75)90010-X.

[16] F. Farassat, G. Succi, A review of propeller discrete frequency noise prediction technology with emphasis on two current methods for time domain calculations, Journal of Sound and Vibration 71 (1980) 399–419. doi:10.1016/0022-460X(80)90422-8.

[17] K. Brentner, F. Farassat, Helicopter noise prediction: The current status and future direction, Journal of Sound and Vibration 170 (1994) 79–96. doi:10.1006/jsvi.1994.1047.

[18] J. E. Ffowcs Williams, D. L. Hawkings, Sound generation by turbulence and surfaces in arbitrary motion, Philosophical Transactions of the Royal Society of London. Series A, Mathematics and Physical Sciences 264 (1969) 321–342. doi:10.1098/rsta.1969.0031.

[19] S. K. Lele, J. W. Nichols, A second golden age of aeroacoustics?, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372 (2014). doi:10.1098/rsta.2013.0321.

[20] S. Moreau, The third golden age of aeroacoustics, Physics of Fluids 34 (2022) 031301–1–031301–14. doi:10.1063/5.0084060.

[21] C. K. Tam, Computational aeroacoustics: An overview of computational challenges and applications, International Journal of Computational Fluid Dynamics 18 (2004) 547–567. doi:10.1080/10618560410001673551.

[22] T. Colonius, Modeling artificial boundary conditions for compressible flow, Annual Review of Fluid Mechanics 36 (2004) 315–345. doi:10.1146/annurev.fluid.36.050802.121930.

[23] C. K. Tam, J. C. Webb, Dispersion-relation-preserving finite difference schemes for computational acoustics, Journal of Computational Physics 107 (1993) 262–281. doi:10.1006/jcph.1993.1142.

[24] S. K. Lele, Compact finite difference schemes with spectral-like resolution, Journal of Computational Physics 103 (1992) 16–42. doi:10.1016/0021-9991(92)90324-R.

[25] G. Ashcroft, X. Zhang, Optimized prefactored compact schemes, Journal of Computational Physics 190 (2003) 459–477. doi:10.1016/S0021-9991(03)00293-6.

[26] T. Poinsot, S. Lele, Boundary conditions for direct simulations of compressible viscous flows, Journal of Computational Physics 101 (1992) 104–129. doi:10.1016/0021-9991(92)90046-2.

[27] C. K. Tam, Z. Dong, Radiation and outflow boundary conditions for direct computation of acoustic and flow disturbances in a nonuniform mean flow, Journal of Computational Acoustics 4 (1996) 175–201. doi:10.1142/S0218396X96000040.

[28] T. Z. Dong, On boundary conditions for acoustic computations in nonuniform mean flows, Journal of Computational Acoustics 5 (1997) 297–315. doi:10.1142/S0218396X97000174.

[29] M. B. Giles, Nonreflecting boundary conditions for euler equation calculations, AIAA Journal 28 (1990) 2050–2058. doi:10.2514/3.10521.

[30] W. H. Reed, T. Hill, Triangular mesh methods for the neutron transport equation, Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[31] B. Cockburn, G. E. Karniadakis, C.-W. Shu, The development of discontinuous galerkin methods, in: Discontinuous Galerkin Methods, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 3–50. doi:10.1007/978-3-642-59721-3_1.

[32] H. T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous galerkin methods, in: 18th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2007-4079, Miami, FL, 2007. doi:10.2514/6.2007-4079.

[33] D. A. Kopriva, J. H. Kolias, A conservative staggered-grid chebyshev multidomain method for compressible flows, Journal of Computational Physics 125 (1996) 244–261. doi:10.1006/jcph.1996.0091.

[34] D. A. Kopriva, A conservative staggered-grid chebyshev multidomain method for compressible flows. ii. a semi-structured method, Journal of Computational Physics 128 (1996) 475–488. doi:10.1006/jcph.1996.0225.

[35] D. A. Kopriva, A staggered-grid multidomain spectral method for the compressible navier–stokes equations, Journal of Computational Physics 143 (1998) 125–158. doi:10.1006/jcph.1998.5956.

[36] Y. Liu, M. Vinokur, Z. Wang, Spectral difference method for unstructured grids i: Basic formulation, Journal of Computational Physics 216 (2006) 780–801. doi:10.1016/j.jcp.2006.01.024.

[37] G. Karniadakis, S. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, 2 ed., Oxford University Press, 2005. doi:10.1093/acprof:oso/9780198528692.001.0001.

[38] M. Lorteau, M. de la Llave Plata, V. Couaillier, Turbulent jet simulation using high-order dg methods for aeroacoustic analysis, International

Journal of Heat and Fluid Flow 70 (2018) 380 – 390. doi:10.1016/j.ijheatfluidflow.2018.01.012.

[39] M. A. Alhawwary, Z. J. Wang, Implementation of a fwh approach in a high-order les tool for aeroacoustic noise predictions, in: AIAA Scitech 2020 Forum, AIAA Paper 2020-1724, Orlando, FL, 2020. doi:10.2514/6.2020-1724.

[40] D. Lindblad, S. Sherwin, C. Cantwell, J. Lawrence, A. Proenca, M. Moragues Ginard, Aeroacoustic analysis of a subsonic jet using the discontinuous galerkin method, in: 28th AIAA/CEAS Aeroacoustics Conference, AIAA Paper 2022-2932, Southampton, UK, 2022. doi:10.2514/6.2022-2932.

[41] R. Modi, M. A. Alhawwary, A. Akbarzadeh, F. Witherden, A. Jameson, Aeroacoustics noise prediction for the airfoil-rod benchmark using high-order large eddy simulation on unstructured grids and the acoustic analogy approach in frequency-domain, in: AIAA SCITECH 2023 Forum, AIAA Paper 2023-0978, National Harbor, MD, 2023. doi:10.2514/6.2023-0978.

[42] D. Lindblad, S. Sherwin, C. Cantwell, J. Lawrence, A. Proenca, M. Moragues Ginard, Large eddy simulations of isolated and installed jet noise using the high-order discontinuous galerkin method, in: AIAA SCITECH 2023 Forum, AIAA Paper 2023-1546, National Harbour, MD, 2023. doi:10.2514/6.2023-1546.

[43] C. D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Men-

galdo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. Kirby, S. Sherwin, Nektar++: An open-source spectral/hp element framework, Computer Physics Communications 192 (2015) 205 – 219. doi:10.1016/j.cpc.2015.02.008.

[44] D. Moxey, C. D. Cantwell, Y. Bao, A. Cassinelli, G. Castiglioni, S. Chun, E. Juda, E. Kazemi, K. Lackhove, J. Marcon, G. Mengaldo, D. Serson, M. Turner, H. Xu, J. Peiró, R. M. Kirby, S. J. Sherwin, Nektar++: Enhancing the capability and application of high-fidelity spectral/hp element methods, Computer Physics Communications 249 (2020). doi:10.1016/j.cpc.2019.107110.

[45] Antares Development Team, Antares Documentation Release 1.17.0, 2023. URL: https://cerfacs.fr/antares/.

[46] D. Di Stefano, A. Rona, E. Hall, G. Puigt, Implementing the ffowcs williams and hawkings acoustic analogy in antares, in: The 22nd International Congress on Sound and Vibration (ICSV22), Florence, Italy, 2015.

[47] D. Di Stefano, A. Rona, E. Hall, C. L. Morfey, G. Puigt, Validating the ffowcs williams and hawkings acoustic analogy implementation in antares, in: 22nd AIAA/CEAS Aeroacoustics Conference, AIAA Paper 2016-3059, Lyon, France, 2016. doi:10.2514/6.2016-3059.

[48] F. Farassat, Derivation of Formulations 1 and 1A of Farassat, Technical Report NASA/TM-2007-214853, NASA, Hampton, VA, 2007.

[49] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (2009) 1309–1331. doi:10.1002/nme.2579.

[50] G. A. Brès, P. Jordan, V. Jaunet, M. Le Rallic, A. V. G. Cavalieri, A. Towne, S. K. Lele, T. Colonius, O. T. Schmidt, Importance of the nozzle-exit boundary-layer state in subsonic turbulent jets, Journal of Fluid Mechanics 851 (2018) 83–124. doi:10.1017/jfm.2018.476.

[51] F. Gand, M. Huet, T. Renaud, F. Sartor, Zdes of jets aeroacoustics: recent progress with unstructured grids and challenges, in: AIAA AVIATION 2023 FORUM, San Diego, CA, 2023.

[52] D. Moxey, M. D. Green, S. J. Sherwin, J. Peiró, An isoparametric approach to high-order curvilinear boundary-layer meshing, Computer Methods in Applied Mechanics and Engineering 283 (2015) 636–650. doi:10.1016/j.cma.2014.09.019.

[53] The Nektar++ Developer Team, Nektar++: Spectral/hp Element Framework: User Guide, 5.3.0 ed., 2023. URL: https://www.nektar.info.

[54] The HDF Group, Hierarchical data format, version 5, 2023. URL: https://www.hdfgroup.org/HDF5/.

[55] Z.-G. Yan, Y. Pan, G. Castiglioni, K. Hillewaert, J. Peiró, D. Moxey, S. J. Sherwin, Nektar++: Design and implementation of an implicit,

spectral/hp element, compressible flow solver using a jacobian-free newton krylov approach, Computers & Mathematics with Applications 81 (2021) 351–372. doi:10.1016/j.camwa.2020.03.009.

[56] R. Hartmann, P. Houston, Symmetric interior penalty dg methods for the compressible navier-stokes equations i: Method formulation, International Journal of Numerical Analysis and Modeling 3 (2006) 1–20.

[57] G. Mengaldo, D. De Grazia, F. Witherden, A. Farrington, P. Vincent, S. J. Sherwin, J. Peiro, A guide to the implementation of boundary conditions in compact high-order methods for compressible aerodynamics, in: 7th AIAA Theoretical Fluid Mechanics Conference, AIAA Paper 2014-2923, Atlanta, GA, 2014. doi:10.2514/6.2014-2923.

[58] P. L. Roe, Approximate riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (1981) 357–372. doi:10.1016/0021-9991(81)90128-5.

[59] A. Uranga, P.-O. Persson, M. Drela, J. Peraire, Implicit large eddy simulation of transition to turbulence at low reynolds numbers using a discontinuous galerkin method, International Journal for Numerical Methods in Engineering 87 (2011) 232–261. doi:10.1002/nme.3036.

[60] A. D. Beck, T. Bolemann, D. Flad, H. Frank, G. J. Gassner, F. Hindenlang, C.-D. Munz, High-order discontinuous galerkin spectral element methods for transitional and turbulent flow simulations, International Journal for Numerical Methods in Fluids 76 (2014) 522–548. doi:10.1002/fld.3943.

[61] C. Carton de Wiart, K. Hillewaert, L. Bricteux, G. Winckelmans, Implicit les of free and wall-bounded turbulent flows based on the discontinuous galerkin/symmetric interior penalty method, International Journal for Numerical Methods in Fluids 78 (2015) 335–354. doi:10.1002/fld.4021.

[62] M. Bergmann, C. Morsbach, M. Franke, Implicit LES of a Turbulent Channel Flow with High-Order Discontinuous Galerkin and Finite Volume Discretization, volume 25 of *ERCOFTAC Series*, Springer International Publishing, 2019, pp. 61–67. doi:10.1007/978-3-030-04915-7.

[63] R. C. Moura, S. J. Sherwin, J. Peiró, Linear dispersion–diffusion analysis and its application to under-resolved turbulence simulations using discontinuous galerkin spectral/hp methods, Journal of Computational Physics 298 (2015) 695–710. doi:10.1016/j.jcp.2015.06.020.

[64] R. C. Moura, G. Mengaldo, J. Peiró, S. J. Sherwin, On the eddy-resolving capability of high-order discontinuous galerkin approaches to implicit les / under-resolved dns of euler turbulence, Journal of Computational Physics 330 (2017) 615–623. doi:10.1016/j.jcp.2016.10.056.

[65] G. Mengaldo, R. C. Moura, B. Giralda, J. Peiró, S. J. Sherwin, Spatial eigensolution analysis of discontinuous galerkin schemes with practical insights for under-resolved computations and implicit les, Computers and Fluids 169 (2018) 349–364. doi:10.1016/j.compfluid.2017.09.016.

[66] P.-O. Persson, J. Peraire, Sub-cell shock capturing for discontinuous galerkin methods, in: 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 2006-112, Reno, Nevada, 2006. doi:10.2514/6.2006-112.

[67] C. Kennedy, M. Carpenter, Diagonally implicit Runge-Kutta methods for ordinary differential equations. A review, Technical Report NASA/TM-2016-219173, NASA, Hampton, VA, 2016.

[68] D. A. Knoll, D. E. Keyes, Jacobian-free newton–krylov methods: a survey of approaches and applications, Journal of Computational Physics 193 (2004) 357–397. doi:10.1016/j.jcp.2003.08.010.

[69] Y. Saad, M. H. Schultz, Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (1986) 856–869. doi:10.1137/0907058.

[70] M. Israeli, S. A. Orszag, Approximation of radiation boundary conditions, Journal of Computational Physics 41 (1981) 115–135. doi:10.1016/0021-9991(81)90082-6.

[71] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M.

Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[72] S. Mendez, M. Shoeybi, S. K. Lele, P. Moin, On the use of the ffowcs williams-hawkings equation to predict far-field jet noise from large-eddy simulations, International Journal of Aeroacoustics 12 (2013) 1–20. doi:10.1260/1475-472X.12.1-2.1.

[73] P. di Francescantonio, A new boundary integral formulation for the prediction of sound radiation, Journal of Sound and Vibration 202 (1997) 491–509. doi:10.1006/jsvi.1996.0843.

[74] M. L. Shur, P. R. Spalart, M. K. Strelets, Noise prediction for increasingly complex jets. part i: Methods and tests, International Journal of Aeroacoustics 4 (2005) 213–245. doi:10.1260/1475472054771376.

[75] P. R. Spalart, M. L. Shur, Variants of the ffowcs williams - hawkings equation and their coupling with simulations of hot jets, International Journal of Aeroacoustics 8 (2009) 477–491. doi:10.1260/147547209788549280.

[76] G. Rahier, M. Huet, J. Prieur, Additional terms for the use of ffowcs williams and hawkings surface integrals in turbulent flows, Computers and Fluids 120 (2015) 158–172. doi:10.1016/j.compfluid.2015.07.014.

[77] A. Najafi-Yazdi, G. A. Brès, L. Mongeau, An acoustic analogy formulation for moving sources in uniformly moving media, Proceedings of the Royal Society A, Mathematical, Physical and Engineering Sciences 467 (2011) 144–165. doi:10.1098/rspa.2010.0172.

[78] P. E. Vos, S. J. Sherwin, R. M. Kirby, From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations, Journal of Computational Physics 229 (2010) 5161–5181. URL: https://www.sciencedirect.com/science/article/pii/S0021999110001506. doi:https://doi.org/10.1016/j.jcp.2010.03.031.

[79] C. Cantwell, S. Sherwin, R. Kirby, P. Kelly, From h to p efficiently: Strategy selection for operator evaluation on hexahedral and tetrahedral elements, Computers and Fluids 43 (2011) 23–28. URL: https://www.sciencedirect.com/science/article/pii/S0045793010002057. doi:https://doi.org/10.1016/j.compfluid.2010.08.012, symposium on High Accuracy Flow Simulations. Special Issue Dedicated to Prof. Michel Deville.

[80] D. Moxey, C. Cantwell, R. Kirby, S. Sherwin, Optimising the performance of the spectral/hp element method with collective linear algebra operations, Computer Methods in Applied Mechanics and Engineering 310 (2016) 628–645. URL: https://www.sciencedirect.com/science/article/pii/S0045782516306739. doi:https://doi.org/10.1016/j.cma.2016.07.001.

[81] D. Moxey, R. Amici, M. Kirby, Efficient matrix-free high-order finite element evaluation for simplicial elements, SIAM Journal on Scientific Computing 42 (2020) C97–C123. URL: https://doi.org/10.1137/19M1246523. doi:10.1137/19M1246523. arXiv:https://doi.org/10.1137/19M1246523.

[82] A. Proença, J. Lawrence, R. Self, Experimental investigation into the turbulence flow field of in-flight round jets, AIAA Journal 58 (2020) 1–11. doi:10.2514/1.J059035.

[83] C. Bogey, O. Marsden, C. Bailly, Influence of initial turbulence level on the flow and sound fields of a subsonic jet at a diameter-based reynolds number of 10^5, Journal of Fluid Mechanics 701 (2012) 352–385. doi:10.1017/jfm.2012.162.

[84] M. Zhu, C. Pérez Arroyo, A. Fosso Pouangué, M. Sanjosé, S. Moreau, Isothermal and heated subsonic jet noise using large eddy simulations on unstructured grids, Computers and Fluids 171 (2018) 166–192. doi:10.1016/j.compfluid.2018.06.003.

[85] F. Gand, M. Huet, On the generation of turbulent inflow for hybrid rans/les jet flow simulations, Computers and Fluids 216 (2021). doi:10.1016/j.compfluid.2020.104816.

[86] P. Welch, The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms, IEEE Transactions on Audio and Electroacoustics 15 (1967) 70–73. doi:10.1109/TAU.1967.1161901.

[87] U. Michel, The role of source interference in jet noise, in: 15th AIAA/CEAS Aeroacoustics Conference (30th AIAA Aeroacoustics Conference), AIAA Paper 2009-3377, Miami, FL, 2009. doi:10.2514/6.2009-3377.

[88] R. Head, M. Fisher, Jet/surface interaction noise - analysis of farfield low frequency augmentations of jet noise due to the presence of a solid shield, in: 3rd Aeroacoustics Conference, AIAA Paper 76-502, Palo Alto, CA, 1976. doi:10.2514/6.1976-502.

[89] S. Ristov, R. Prodan, M. Gusev, K. Skala, Superlinear speedup in hpc systems: Why and when?, in: 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), 2016, pp. 889–898.