Proceedings of the ASME Turbo Expo 2023
GT 2023
June 26-30, 2023, Boston, MA, USA

GT2023-102647

DESKTOP-DNS: AN OPEN TOOLKIT FOR TURBOMACHINERY AERODYNAMICS

Andrew P.S. Wheeler
The Whittle Laboratory,
Engineering Department, University of Cambridge, United Kingdom,
aw329@cam.ac.uk

ABSTRACT

The prevailing view is that high fidelity simulation, particularly DNS (direct numerical simulation), is not something for the practical turbomachinery aerodynamicist - requiring too much computational and personal effort to make it worth it. The aim of the 'Desktop-DNS' toolkit described in this paper is to change this by greatly lowering the barrier to entry for running DNS. The paper shows how, using an efficient high-order Navier-Stokes computer code, it is becoming increasingly possible to solve testcases of industry relevance with high fidelity LES and DNS, making use of the latest advances in single compute node performance. This is achievable using both efficient algorithms and GPU acceleration. The paper will use a compressor blade testcase to illustrate how, in some cases, high-fidelity simulations can be performed at relatively low costs on a small number of computer nodes. This raises the possibility of a much more widespread use of DNS to inform early design choices, enhance or benchmark current models, and to potentially reveal new physical mechanisms. An increasing uptake of DNS into mainstream CFD use has profound consequences in terms of data-processing and the training needs for practitioners. The paper therefore provides guidance on some practical ways in which high fidelity data can be exploited for turbomachinery design, as well as a step-by-step guide and set of codes to enable a turbomachinery aerodynamicist to explore DNS.

NOMENCLATURE

c	Chord
c_{ax}	Axial chord
M	Mach number
p	Pressure
p_o	Total pressure
q	Heat flux
Re	Reynolds number (based on axial chord)
s	Entropy
t	Time
T	Temperature
Tu	Turbulence intensity
V	Velocity magnitude

GREEK LETTERS

α	Flow angle
ε	Turbulent dissipation
ρ	Density
ν	Kinematic viscosity

INTRODUCTION

Around 20 years ago, the first early direct numerical simulation (DNS) of low-pressure turbines were published [1, 2]. The simulations were run at a blade Re$\sim$50,000-70,000, taking around 1.5-3 months on the latest hardware at the time (using $\sim$20 million solution points). Today, this paper suggests that these same cases could be run within 4 hours on an Amazon Virtual Machine, costing around \$40USD (assuming a 'p4d.24xlarge' instance and current spot prices [3]).

DNS and high fidelity simulations have developed considerably over the past decades [4, 5]. Although the costs are still considerable, particularly for high Reynolds numbers, there are

Copyright © 2023 by ASME

many practical benefits; Figure 1 illustrates the challenges and opportunities for engineers wishing to exploit DNS in design. DNS offers unrivalled predictive accuracy compared to conventional RANS and wall-modelled LES. The data determined from DNS can be used to improve low-order methods and RANS models. For new designs, DNS can be used to discover new physical mechanisms or de-risk early design choices. Virtual experiments can be used to determine quantities which would otherwise be very challenging to measure in physical experiments (such as dissipation). Overcoming the 'DNS Mountain' requires access to suitable resources. However before investing in computer resources, it is important that there are means to run simulations efficiently through access to suitable software, an ability to store and make use of the data through appropriate post-processing tools, and the in-house expertise to ensure high quality simulations are conducted.

This paper, along with the open 'Desktop-DNS' toolkit is intended to help overcome some of these challenges. The term 'Desktop-DNS' is used here in a loose way to describe high fidelity simulations that can be run on local computer systems, rather than large multi-node machines. The point of this distinction is to demonstrate what individuals can in theory achieve without access to very large computer systems. The advantage of being able to run cases 'locally' (e.g. using a powerful single compute node, a small local cluster, or on a cloud virtual machine) is that a much wider group of users can benefit from the advancements in computing power and high fidelity codes. For instance, a small business or a dedicated team within a larger business can begin to explore the benefits of DNS without a great deal of risk and capital investment.

The paper is in three parts: First the *3DNS-gpu* solver is described; second, the 'Desktop-DNS' toolkit is demonstrated on a compressor blade profile; third the opportunity for using 'Desktop-DNS' in design is mapped-out over time.

The Desktop-DNS toolkit, including pre- and post-processing codes, test-case geometry and the 3DNS-gpu solver is provided to readers who wish to follow the examples used in this paper and adapt for personal/academic use at this link: `https://github.com/aw329/desktop-dns`

THE DNS SOLVER

The solver used here is a development of the high-order compressible Navier-Stokes solver 3*DNS* first described in [6]. To-date *3DNS* has been used in a range of studies of compressor cascades, rotor-stator interactions, non-equilibrium boundary layers, tip leakage and real-gas flows [7–12]. There are several branches of 3*DNS* with different parallelization options. The main branch used for large production jobs typically runs on $10^3 - 10^5$ processors on national level supercomputers (such as ARCHER2) for simulations with order of 1 billion solution points.

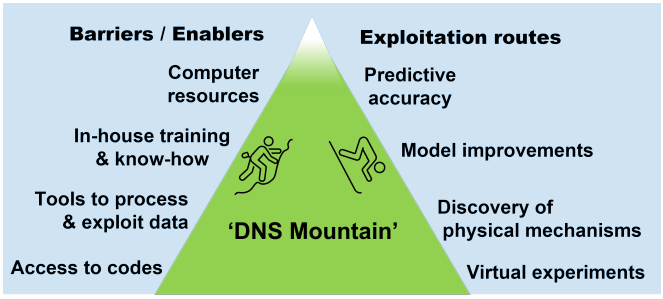


FIGURE 1. **The 'DNS Mountain' preventing widespread use of DNS.**

The aim of this paper is to described how users without access to very large computing systems can run DNS. In order to do this, a new version of the solver was developed called *3DNS-gpu* which is designed for running on systems with a small number of GPU cards, such as: an individual desktop machine or gaming PC; a small compute cluster; or a cloud computing virtual machine.

Numerical scheme

All branches of 3*DNS* solve the flow on curvi-linear structured grids in strong conservation form. The algorithm uses explicit differencing with a 4^{th} order Tam and Webb [13] 7-point stencil scheme and an 8^{th} order explicit filter. Skew-split differencing is applied to the inviscid fluxes using the method of Kennedy and Gruber [14] to reduce dispersion errors. Characteristic interfaces using the method of Kim and Lee [15, 16] are applied at inflow and outflow boundaries, as well as at multi-block boundaries (see Appendix for more details). Time integration is performed using a standard low-storage 4-step Runge-Kutta scheme.

Parallelization strategy

GPUs enable very high levels of parallelization on a single device. However the size of the domain that can be loaded onto a single device is limited by the GPU memory. Therefore, for large simulations it is normally necessary to use several GPUs which carries an overhead due to the need for communication between devices. This is normally what limits the parallel performance for large simulations.

In *3DNS-gpu*, parallelization is achieved with both MPI and OpenACC directives in order to enable the use of both CPUs and GPUs. The topology and geometry is assumed to be two-dimensional, with multi-block boundaries restricted to be normal to the homogeneous (spanwise) direction. Blocks can be split across processors to create sub-blocks. Any number of sub-blocks can also be grouped onto a GPU. Communication between devices is achieved with MPI.

Copyright © 2023 by ASME

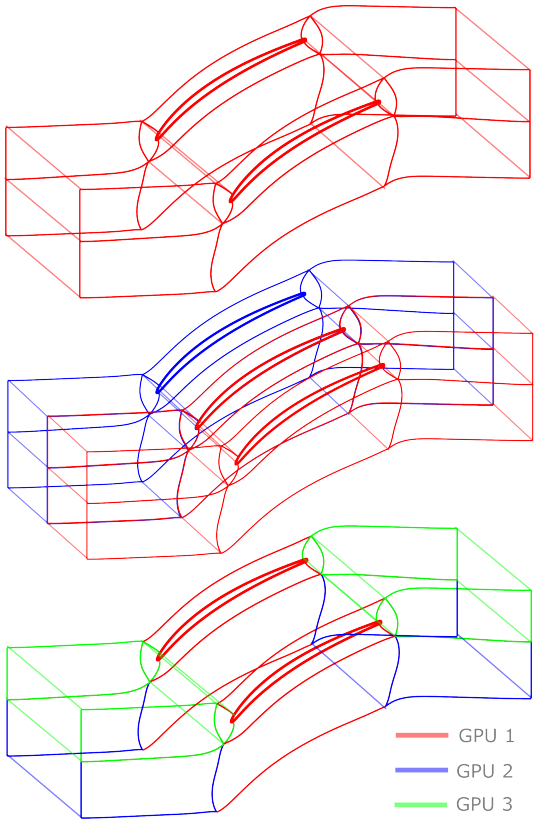


FIGURE 2. **Different parallelization options for a 9-block domain using either 1, 2 or 3 GPU cards**

Figure 2 illustrates three examples for a 9-block computational mesh of a compressor blade. If only one GPU is available, the whole domain can be loaded onto a single GPU. If users have access to multiple GPUs, the blocks can be split in the spanwise direction across multiple GPUs. Blocks or sub-blocks can also be grouped onto GPUs in order to parallelize in the blade-to-blade plane. In the latter, groups of blocks are not restricted to be contiguous. Ensuring GPUs are evenly load balanced is important for compute efficiency. Load balancing is achieved when GPUs and processors have a similar number of solution points - an ability to either split blocks across several GPUs and/or load several blocks onto a single GPU provides a good deal of versatility in this respect.

The method allows users a certain degree of flexibility depending on the hardware available to them. A key advantage of leveraging GPUs in this way is that it offers users who do not have access to large computer systems a possibility for running DNS.

Strong scaling performance of *3DNS-gpu* is shown in Table 1 for a 9-block mesh of 109 million points. The scaling study was performed using the the Cambridge Wilkes 3 computer. Each Wilkes 3 node contains 4 NVIDIA A100 SXM4 80GB GPU cards. The domain topology is as shown in Figure 2. Scaling performance is particularly good when parallelization is achieved in the blade-to-blade plane. In this case communication across GPUs is restricted to multi-block interfaces. When parallelization is performed in the spanwise direction scaling performance drops - this is because spanwise interfaces carry a greater overhead than multi-block boundaries. Spanwise interfaces are treated as interior surfaces. This means that, across every spanwise interface, communication is required twice every Runge-Kutta iteration (once for the flow quantities and once for fluxes); for a seven point stencil, the transferred array size is three times the interface points per flow variable. Multi-block interfaces use a 1-sided difference and therefore only require one inter-device communication per Runge-Kutta iteration to transfer the characteristic terms, using an array size equal to the number of interface points per characteristic term. Multi-block interfaces are therefore more efficient than interior interfaces.

Total GPUs	GPUs blade-to-blade	GPUs span	Time per step (s)	Scaling efficiency (%)
1	1	1	1.21	100
2	2	1	0.600	101
4	4	1	0.290	104
2	1	2	0.759	80
8	4	2	0.185	82

TABLE 1. Strong scaling for 109M pt 9-block mesh with different parrallelization strategies

GPUs	Points $\times 10^6$	Time per step (s)	Efficiency (%)
1	109	1.21	100
4	436	1.16	104

TABLE 2. Weak scaling on a single compute node on Wilkes3

Weak scaling on a single Wilkes3 node is shown in Table 2. Two cases are shown; a 109M point mesh on 1 card and a 436M point mesh on 4 cards - in this case the blocks are split in the blade-to-blade plane to achieve uniform load per GPU. There is virtually no difference in compute-time in these two cases, demonstrating good scaling performance when using a high number of solution points ($\sim$ 0.5 billion points).

EXPLORING A NEW COMPRESSOR PROFILE

The next part of the paper uses an industry-relevant testcase of a compressor blade section to show how we might practically

Copyright © 2023 by ASME

s/c_{ax}	c/c_{ax}	α_{in}	α_{exit}	Re_{in}	M_{in}	Tu_{in}
0.72	1.2	50.2^o	25.2^o	2.33×10^5	0.2	0.4%

TABLE 3. Compressor blade details - Tu_{in} calculated at $0.5c_{ax}$ upstream of the leading-edge.

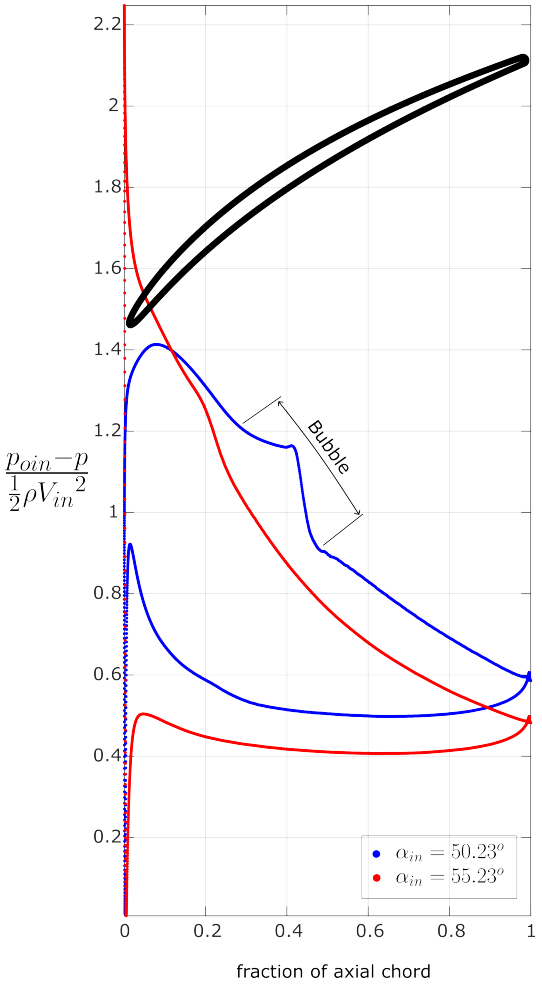


FIGURE 3. **Compressor blade section and loading distributions at two incidences.**

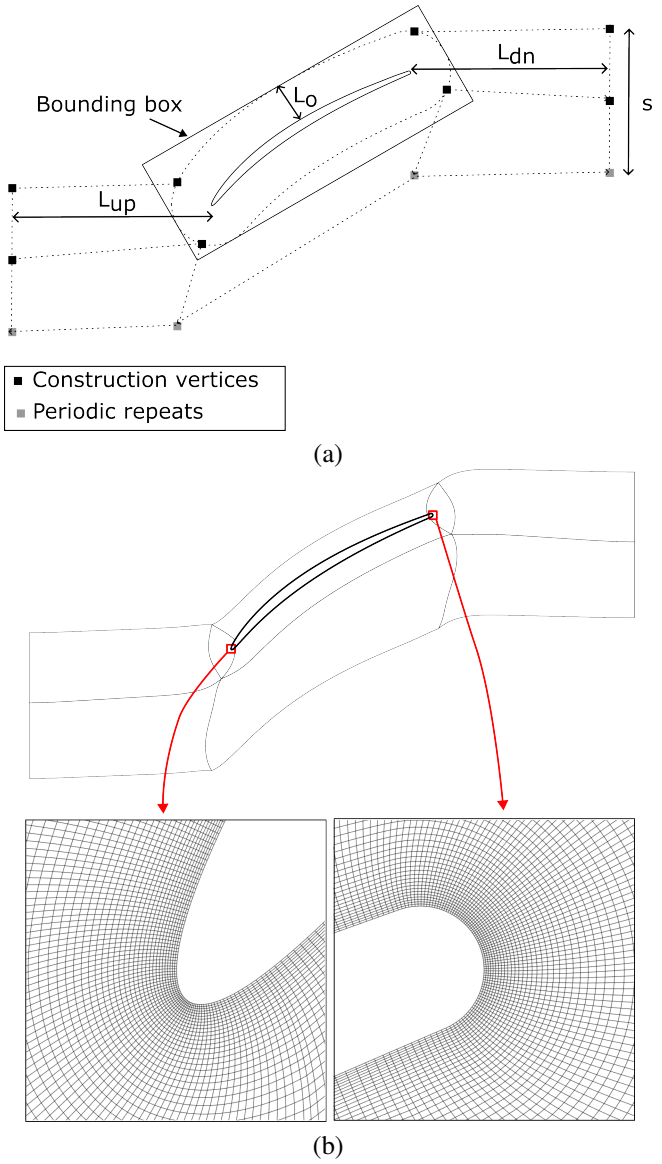


FIGURE 4. **Mesh generation routine: (a) Automatic construction of the block topology for a blade profile; (b) final topology and mesh in region of the leading-edge and trailing-edge (every 4^{th} grid line shown).**

use DNS to inform design. The profile design details are shown in Table 3 and the profile can be seen in Figure 3. The profile is entirely new and created for the purpose of this study.

Also shown in Figure 3 is the time-average loading at two incidences. As can be seen, the design is a controlled diffusion style and follows industry-standard design practise. A separation bubble forms downstream of peak-suction at the design condition, $\alpha_{in} = 50.23^o$. At 5 degrees incidence an over-speed is generated on the suction surface close to the leading-edge. These two phenomena drive the transition process and turbulent breakdown on the blade surface. These phenomena will be discussed later.

The next three sections describe: first how the computational domain is generated; secondly the steps needed to establish a high quality simulation; and thirdly how the results can be exploited to inform the design process.

Copyright © 2023 by ASME

Step 1 - setting-up the case

Readers who wish to follow this example can do so by using the Jupyter-notebook found here `https://github.com/aw329/desktop-dns`

In this section the steps for creating a DNS quality mesh and an initial flow are described.

Mesh generation An important part of DNS and wall-resolved LES is high quality meshing. Near-wall resolution is particularly important because non-uniformity in the near wall points can significantly increase compute costs. For compressible codes such as 3*DNS* explicit time-stepping is used because meshes are required to be relatively uniform throughout the domain - this means that dual-time-stepping and implicit time integration is not well suited in most cases. Thus, the minimum cell size (typically set by the near-wall cells) governs the global time step according to a CFL criterion, which directly controls the compute cost. Ensuring the wall-normal cell sizes are uniform around the blade surface therefore minimizes the compute cost for a given number of solution points.

The method for generating the block topology for a compressor blade is illustrated in Figure 4. The initial o-grid boundary is a uniform offset a distance L_o away from the blade surface. A bounding box is constructed around this boundary, and construction vertices are set at the minimum distance between the o-grid boundary and the bounding box corners. The inlet and outlet boundaries are positioned distances of L_{up} and L_{dn} from the leading and trailing edges. This provides a framework to determine the initial block boundaries. A Poisson solver (see Appendix) can then be applied to obtain the final block boundaries. Boundary conditions are imposed at walls to ensure both wall-normal orthogonality and a uniform near-wall cell height. The final refined mesh is obtained by successive refinement and application of the Poisson solver. An example mesh is shown in Figure 4; the mesh is downsampled to show every 4^{th} grid line in the leading and trailing-edge regions.

The choice of the upstream and downstream lengths is important. Setting L_{up} and L_{dn} too large can add significantly to the solution points and compute cost. On the other-hand, characteristic boundary conditions used at the inlet and exit make use of a local one-dimensional approximation and therefore work best where the flow is relatively uniform in the pitch-wise direction. Furthermore, a short upstream length may not give sufficient distance for inflow turbulence to develop to a natural state. Setting L_{up} and L_{dn} to the axial chord length c_{ax} tends to be a good compromise. Further detail regarding the implementation of the characteristic boundary conditions is described in the Appendix.

Once the computational domain has been selected, initial steady simulations can be run in order to check the set-up and to provide a starting solution for the unsteady simulations. Starting the unsteady simulations from a steady case is useful for reducing the computational time required overall; this avoids long time-scale transients that may be created by initiating the unsteady simulations from a poor starting solution.

Initial flow The Desktop-DNS toolkit provides options to initiate the flow using a compressible potential-flow solution, as shown in Figure 5. This uses an adapted version of the Poisson solver used for mesh generation (described in the Appendix). The inviscid solution is interpolated onto the mesh points to provide an initial flow field for the *3DNS-gpu* solver.

In order to establish the viscous field, *3DNS-gpu* can be run in 'RANS' mode. In this case, local time-stepping is used to time-march the flow to a steady-state. Two options are available: a simple mixing-length model can be used to compute the eddy viscosity, or an eddy viscosity can be prescribed. Prescribing an eddy viscosity is possible once we have established a high fidelity simulation - this will be discussed later in the paper. The simple mixing-length model is used in order to establish a new test-case.

Figure 5 shows an example of the flow determined using this method. This 'steady' flow field can then be used as an initial flow for the time-accurate simulations by re-starting the simulations with the RANS-mode switched off.

Step 2 - running a simulation

Readers who wish to follow this example can do so by using the Jupyter-notebook found here `https://github.com/aw329/desktop-dns`

In this section the steps taken to run a high fidelity simulation and assess accuracy are described.

Extruding the two-dimensional flow For this example, which is homogeneous in the spanwise direction, it is advantageous to run initial simulations which are strictly two-dimensional (i.e. with zero spanwise extent) - this solution is then extruded onto a three-dimensional mesh once the flow is well established. The number of spanwise points is then successively increased to achieve the required spanwise resolution. In this case a spanwise extent of $0.1c_{ax}$ is used. The spanwise boundaries are periodic (equivalent to a cascade with infinite span). Snapshots from simulations with progressively higher spanwise points are shown in Figure 6. The figure shows that the strictly two-dimensional case is highly unsteady, but because there is no possibility for three-dimensional breakdown, structures from the separation bubble and the trailing-edge retain a strong coherence. Successively increasing the spanwise points leads to the development of three-dimensional instabilities and turbulence.

Copyright © 2023 by ASME

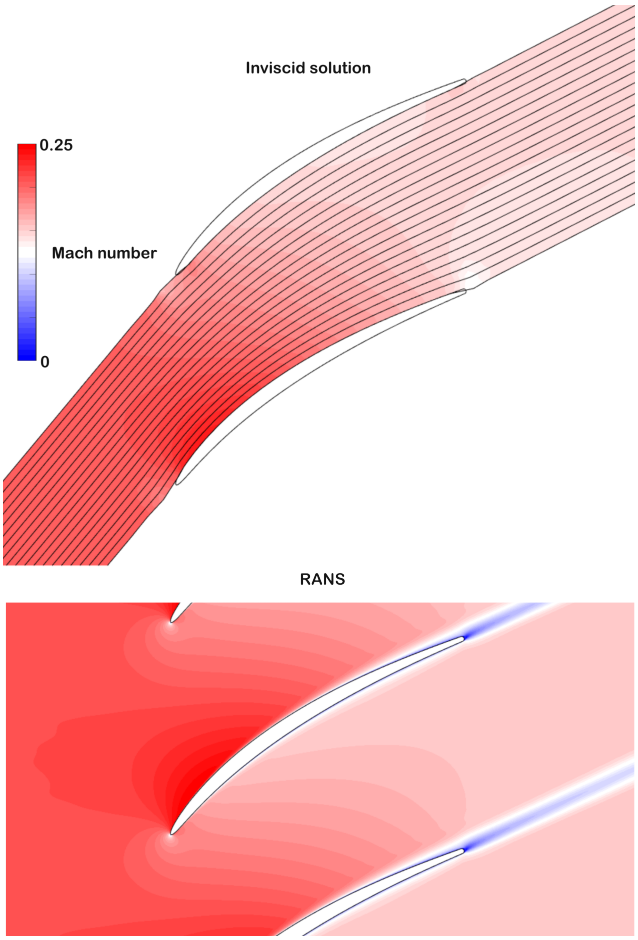


FIGURE 5. **Initial guess from potential-flow solver (top) and RANS (bottom).**

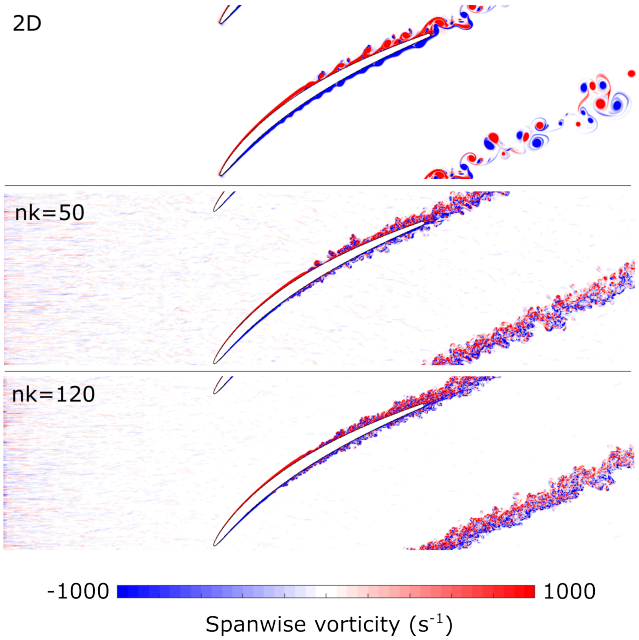


FIGURE 6. **Snapshots of spanwise vorticity from simulations with progressively higher spanwise resolution**

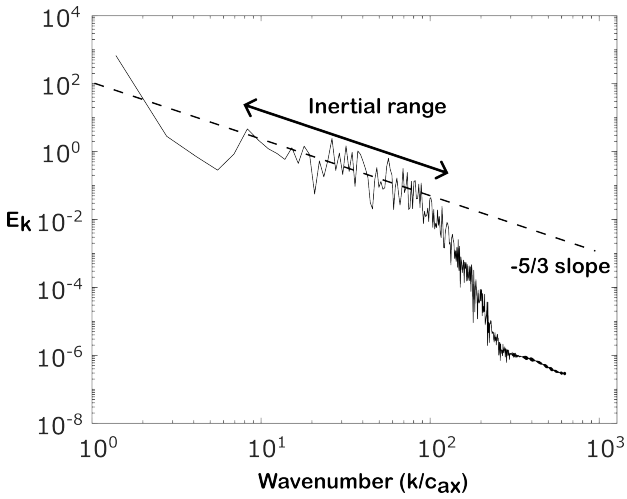


FIGURE 7. **Turbulence spectrum computed in the inlet blocks of the computational domain.**

Adding inflow turbulence Imposing inflow turbulence is very important in any wall-resolved simulation. This is because turbulent breakdown is normally driven by the development of instabilities initiated by turbulence in the mainstream flow. Without inflow turbulence, unsteadiness such as trailing-edge vortex shedding will tend to retain unnaturally strong two-dimensional coherence. On the other-hand there is no consistent view on what is 'realistic' inflow turbulence; the turbulence spectrum in a real machine is not well characterized and unlikely to be homogeneous and isotropic. In *3DNS-gpu* an automatic method for synthetic inflow turbulence is used; this method is described in the Appendix. In this case an intensity of $Tu = 0.4\%$ is imposed - the turbulence spectrum is shown in Figure 7 showing energy in the inertial sub-range over around two orders of magnitude.

Gathering statistics In *3DNS-gpu*, statistical data is determined from time integration of the conserved quantities $(\rho, \rho u, \rho v, \rho w, E_t)$, momentum flux terms $(\rho uu, \rho vv, \rho ww, \rho uv, \rho uw, \rho vw)$ and terms used in the entropy transport equation (discussed later). Because the geometry is homogeneous in the spanwise direction, these quantities are also integrated across the span during the simulation in order to give

Copyright © 2023 by ASME

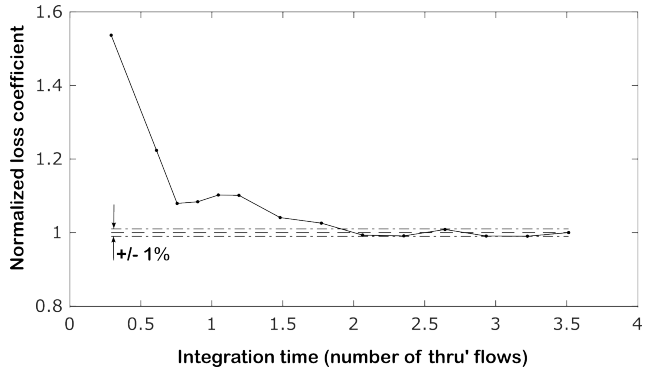


FIGURE 8. **Statistical convergence of loss coefficient Y_p at $0.5c_{ax}$ downstream of the trailing edge (651M mesh point case).**

a two-dimensional time-average flow field.

In order to compute total pressure loss coefficient Y_p, the time average pressure p is determined using Equation 1.

$$\frac{p}{\gamma-1} = \overline{Et} - \frac{1}{2}\left(\overline{\rho uu} + \overline{\rho vv} + \overline{\rho ww}\right) \tag{1}$$

It's important to note that Equation 1 uses the time-average momentum terms to determine time-average pressure. Equation 1 is equivalent to a simple time average of the instantaneous pressure. The Mach number M and total pressure p_o are then determined from the time-average velocity field.

$$M = \sqrt{\frac{(\overline{\rho u})^2 + (\overline{\rho v})^2 + (\overline{\rho w})^2}{\gamma R p \overline{\rho}}}; \tag{2}$$

$$p_o = p\left(1 + \frac{\gamma-1}{2}M^2\right)^{\gamma/(\gamma-1)}. \tag{3}$$

The total pressure loss coefficient Y_p is computed based on the inlet conditions $Y_p = (p_o - p_{oin})/(1/2\rho_{in}V_{in}^2)$. Figure 8 shows the convergence of the loss coefficient computed with increasing integration time. The integration time is normalized by the through-flow time, based on the axial chord and inlet axial velocity c_{ax}/u_{in}. The figure shows that around 3.5 through-flow times are required to achieve statistical convergence. It's important to note that this time would be increased if there were any large length scale unsteadiness. Longer times will also be necessary to wash through any transients, but this can be avoided by running simulations on a prior less refined mesh, as discussed next.

Mesh sensitivity In general, it is far better to establish a simulation on an under-resolved mesh first, to make sure the boundary conditions and set-up are as required, before running highly resolved simulations. This can avoid a great deal of compute time in washing-through initial transients which occur due to the initial conditions. The Desktop-DNS toolkit can be used to interpolate simulations onto successively finer meshes until mesh independence has been achieved.

Total points $\times 10^6$	174	418	651
Spanwise points	50	120	120
$\Delta_n{}^+$	0.63	0.65	0.65
$\Delta_t{}^+$	5.54	5.78	4.59
$\Delta_z{}^+$	19.0	8.08	8.06

TABLE 4. Mesh sizes and mean near-wall resolution for compressor section case study

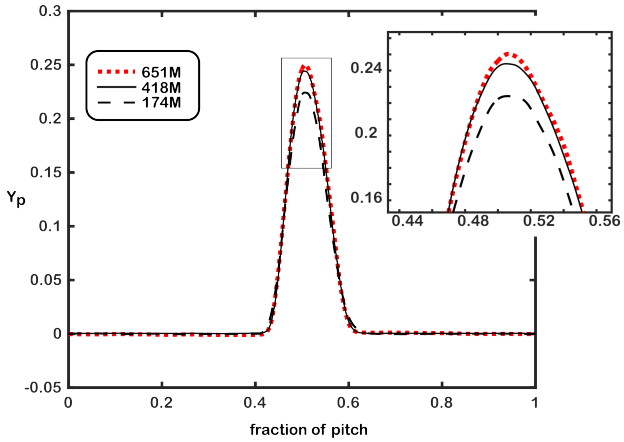


FIGURE 9. **Mesh sensitivity of wake profile at $0.5c_{ax}$ downstream of the trailing-edge.**

Table 4 shows details for 3 meshes used for this paper. The flow was first established on a mesh of 174 million points, then interpolated onto a mesh with double the spanwise resolution (418M pts). The third mesh (651M pts) was created by increasing the points in the blade-to-blade plane by a factor of 1.55 and the flow from the 418M pt mesh was interpolated onto this new mesh to create a third case.

Table 4 also shows the average near-wall cell sizes for these cases. It is important that the near-wall mesh is sufficiently resolved to capture both the regions of turbulent flow, and also the regions where instabilities develop and ultimately drive transition. A good rule of thumb is to ensure that the near-wall cell size in the wall normal direction in wall units is $\Delta_n{}^+ < 1$, the streamwise cell length $\Delta_t{}^+ < 15$ and the lateral (spanwise) cell length is $\Delta_z{}^+ < 10$. Table 4 shows that these criteria are met for the two largest meshes with 418 and 651 million points. The spanwise resolution of the 174 million point mesh is around a

Copyright © 2023 by ASME

factor of 2 too high.

Having run cases with different mesh sizes, it is then possible to determine mesh sensitivity. Figure 9 compares the wake loss coefficient for these 3 cases. The figure shows very little difference between the 418 and 651 million points. The mass-averaged Y_p values for the three cases are Y_p =0.0208 (651M pts), 0.0211 (418M pts), 0.0199 (174M pts) while the average exit flow angles are $\alpha_{out} = 25.16^o$ (651M pts), 25.11^o (418M pts), 25.08^o (174M pts). The results show that even with 174 million points we are able to capture the loss quite accurately (within 5 per cent for loss and within 0.1^o for deviation).

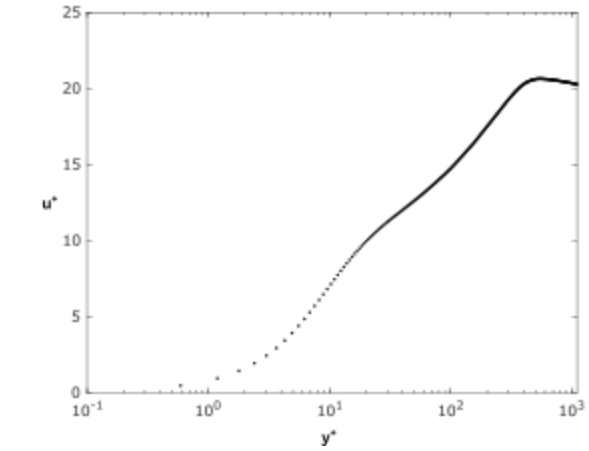
Given that the mesh count has increased by nearly a factor of 4 across these cases, the results show that further increases in mesh resolution are unlikely to yield significant improvements in predictive accuracy of the blade performance.

Assessing accuracy The next step is to assess the quality of the simulations. A rigorous method to assess accuracy is to check the balance of terms in the turbulent kinetic energy transport equation. An alternative is to make use of the entropy transport equation. The entropy transport equation is much simpler since it contains far fewer terms, and is normally more useful for assessing loss mechanisms. The latter is the author's preferred choice. The entropy transport equation is

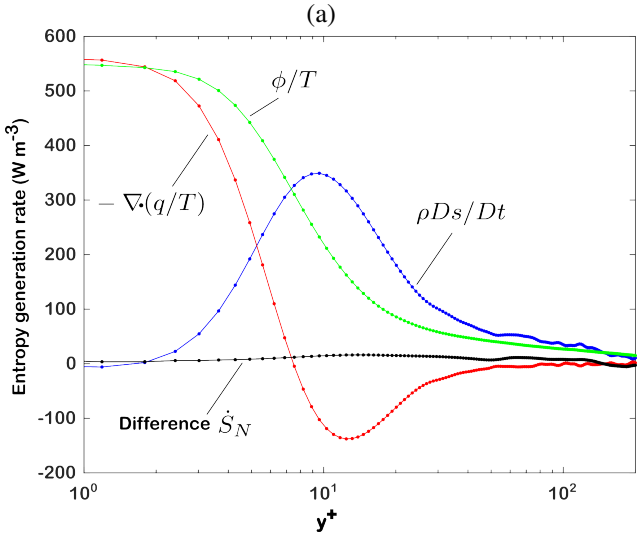
$$\rho \frac{Ds_{irr}}{Dt} = \rho \frac{Ds}{Dt} + \nabla \cdot \left(\frac{q}{T}\right) = \frac{\phi}{T} + \dot{S}_N + \theta. \qquad (4)$$

On the left-hand-side is the irreversible entropy generation rate, which is the sum of the entropy generation and contribution due to heat transfer rate. On the right is the contribution due to resolved dissipation ϕ/T and $\dot{S}_N$ which is the entropy generation rate per unit volume due to numerical dissipation within the simulation. The contribution due to irreversible heat transfer θ across finite temperature differences can be safely ignored in low speed adiabatic cases such as this ($\theta \approx 0$) [8].

Resolving the full dissipation spectrum requires capturing the very smallest-scales in the flow. Under-resolved calculations underestimate the dissipation ϕ because this is computed using the resolved gradients in the flow. The irreversible entropy generation rate on the other hand contains the effects of both the resolved dissipation and numerical dissipation required to stabilize the numerical scheme; in *3DNS-gpu* this is applied using the 8^{th} order explicit filter discussed earlier. When integrated over a volume, the irreversible entropy generation rate will be larger and less resolution dependent than determining dissipation from the resolved flow. The entropy 'budget' described in Equation 4 is therefore very useful because it allows us to quantify the unresolved dissipation. Therefore in *3DNS-gpu* the quantities $\rho us, \rho vs, q_x/T, q_y/T, \phi/T$ are included in the time-average quantities.



(a)



(b)

FIGURE 10. **Time-average suction-surface boundary-layer profiles at $x/c_{ax} = 0.9$ for the 651M mesh point case: (a) velocity; (b) entropy budget.**

Figure 10 shows the boundary-layer profile and entropy budget terms extracted on the suction-surface at $x/c_{ax} = 0.9$ for the 651M mesh point case. Readers may be interested to see that the entropy flux term $\rho Ds/Dt$ peaks at around $y^+ \simeq 10$ and tends to zero at the wall; this is because the flow speed at the wall is zero and therefore there is no entropy flux. While the entropy flux is zero, the irreversible entropy generation rate is not and this is because the heat-transfer term $\nabla \cdot (q/T)$ is equal and opposite to the dissipation term ϕ/T. At first it seems surprising that for an adiabatic case such as this, that the heat transfer term should be of similar magnitude to the dissipation; although the first derivative of temperature at the wall is zero (set by the adiabatic condition) the second derivative must balance the dissipation for a no-slip

Copyright © 2023 by ASME

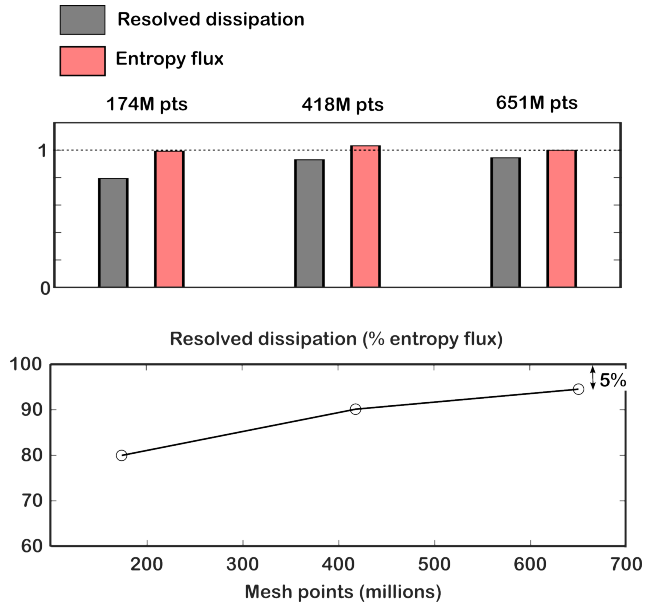


FIGURE 11. **Effect of mesh resolution on resolved dissipation Σ_{res}. The bar charts are normalized to Σ_{flux} of the 651M pt mesh.**

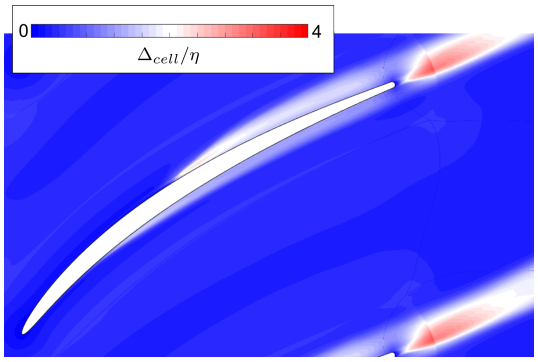


FIGURE 12. **Comparison of cell size against Kolmogorov length scale η for the 651M mesh point case.**

wall.

Figure 10 shows the numerical dissipation $\dot{S}_N$ is small compared to the other terms for this case, showing that the dissipation is being well resolved. We can determine the resolved loss generation (due to dissipation) and entropy flux by integration within a control volume Ω:

$$\Sigma_{res} = \int_{\Omega} \frac{\phi}{T} d\Omega; \tag{5}$$

$$\Sigma_{flux} = \int_{\Omega} \frac{Ds}{Dt} d\Omega. \tag{6}$$

When integrated over a volume with adiabatic boundaries, the heat transfer contribution to the entropy transport equation becomes zero and therefore the difference between Σ_{flux} and Σ_{res} is the effect of numerical dissipation. The flux term Σ_{flux} can either be determined from a volume integral of $\rho Ds/Dt$ or from the difference in entropy flux across the boundaries - these are equivalent according to Gauss's theorem.

Figure 11 shows the effect of mesh resolution on resolved dissipation. In this case Σ_{res} and Σ_{flux} are computed using a volume integral from the inlet of the computational domain to the axial plane $0.1c_{ax}$ downstream of the trailing-edge. The results show an increase in resolved dissipation as mesh resolution increases. The 651M point mesh resolves 95 percent of the dissipation.

The dissipation can be used to estimate the Kolmogorov length-scale and using this we can determine mesh resolution. Whereas the near-wall cell aspect ratios should be in the region of $10-15$, the free-stream cell aspect ratios ought to be around 1. It therefore follows that the free-stream cell sizes will be around 10 in $+$ units. This is quite different to most RANS calculations where coarsening the mesh in the freestream is a sensible approach to reducing compute costs. In DNS, this will lead to poor resolution of turbulence within the passage and lead to higher frequency (smaller wavelength) turbulence being dissipated prematurely or incurring dispersion errors. In regions where the turbulence is likely to be more isotropic (such as in the far wake of a blade), a rule of thumb is to set cell lengths to be within 10 Kolmogorov length-scales. The Kolmogorov length-scale η can be estimated either using turbulent correlations or from a prior RANS or LES calculation, to estimate turbulent dissipation ε. In the current study we can compute turbulent dissipation from the time-average dissipation:

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}; \tag{7}$$

$$\varepsilon = \overline{\phi} - \Phi; \tag{8}$$

where Φ is the dissipation due to the time-mean strain computed from the time-average flow field. As discussed above, poorly-resolved simulations will tend to underestimate the predicted dissipation and will thus overestimate η, giving an unreliably optimistic estimate of cell sizes. Therefore the estimation of the Kolmogorov length-scale should be treated with some caution.

Figure 12 shows the cell length Δ_{cell} for the 651M point case, compared to the Kolmogorov length scale η using the method described above. The cell length Δ_{cell} is taken as the square root of the cell in the blade-to-blade plane. This shows cell sizes $\Delta_{cell} < 4\eta$.

In this section it was shown that with a mesh of 651 million points, DNS quality results are achievable for the compressor section tested here ($Re = 2.33 \times 10^5$). Nonetheless, the mesh

Copyright © 2023 by ASME

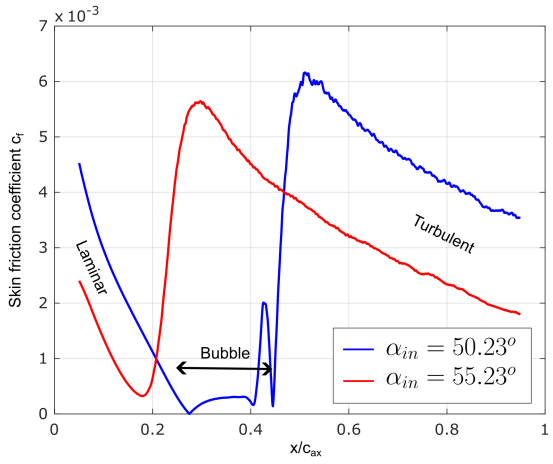


FIGURE 13. **Suction surface skin friction coefficient at two incidences.**

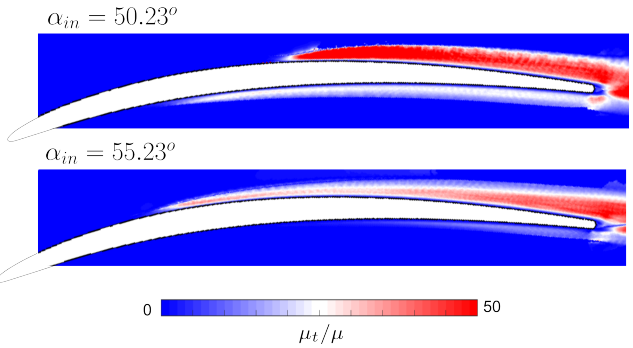


FIGURE 14. **Optimum eddy viscosity from simulations at two incidences.**

sensitivity analysis showed meaningful results were obtained using meshes with fewer solution points - compared to the 651M point mesh, the 174M point mesh gave a loss within 5%, while the 418M point mesh differed by only 1%. The results show that it is not necessary to capture the full dissipation spectrum in order to achieve accurate predictions of loss. This is because the entropy generation rate is less sensitive to mesh resolution than the resolved dissipation (see Figure 11). The reason for this is that the artificial dissipation of the numerical scheme (from the 8^{th} order filter) does a pretty good job of accounting for the unresolved dissipation. As the mesh is refined, the effect of the filter is reduced and more of the smaller-scale dissipation is captured. Beyond a certain point the filtering effect is very small and there is very little benefit in increasing the resolution further.

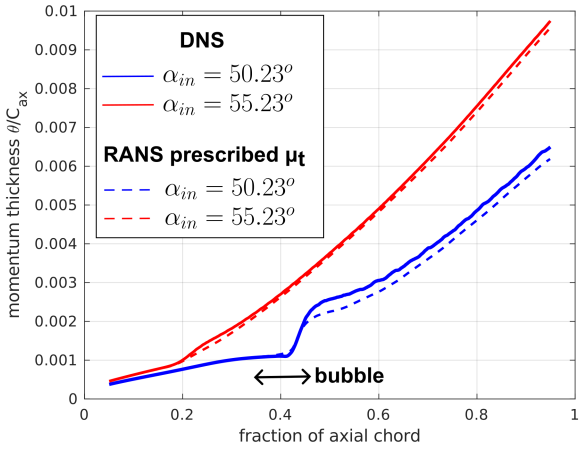


FIGURE 15. **Suction surface momentum thickness from the DNS and from RANS calculations with prescribed eddy viscosity μ_t**

Step 3 - data analysis

Readers who wish to follow this example can do so by using the Jupyter-notebook found here `https://github.com/aw329/desktop-dns`

In this next example we use DNS to investigate the compressor section performance at two different incidences - design ($\alpha = 50.23^o$) and 5 degrees positive incidence ($\alpha = 55.23^o$). The aim here is to show how DNS data can inform the design process. The data described next was obtained from simulations using the 418 million point mesh described above.

The effect of incidence on loading was shown earlier in Figure 3, which shows the separation bubble on the suction surface at design, and the leading-edge over-speed at 5 degrees incidence. The predicted skin friction on the suction surface is shown in Figure 13. The figure shows a clear shift in transition towards the leading edge as incidence increases. The transition process also switches from separated to attached transition at the higher incidence.

A particularly useful quantity to extract from DNS is an estimate of the eddy viscosity μ_t. An optimum eddy viscosity minimizes the error in the Reynolds stress tensor [17]

$$\frac{\mu_t}{\mu} = \frac{\left(\begin{array}{c}\tau_{xy}\overline{\rho u'v'} + \tau_{xz}\overline{\rho u'w'} + \tau_{yz}\overline{\rho v'w'} \\ +\tau_{xx}\overline{\rho u'u'} + \tau_{yy}\overline{\rho v'v'} + \tau_{zz}\overline{\rho w'w'}\end{array}\right)}{\tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2 + \tau_{xx}^2 + \tau_{yy}^2 + \tau_{zz}^2} \tag{9}$$

where τ is the anisotropic part of the stress tensor, determined from the traceless strain tensor.

Figure 14 shows the optimum eddy viscosity determined from the simulations. Interestingly, although the high incidence

Copyright © 2023 by ASME

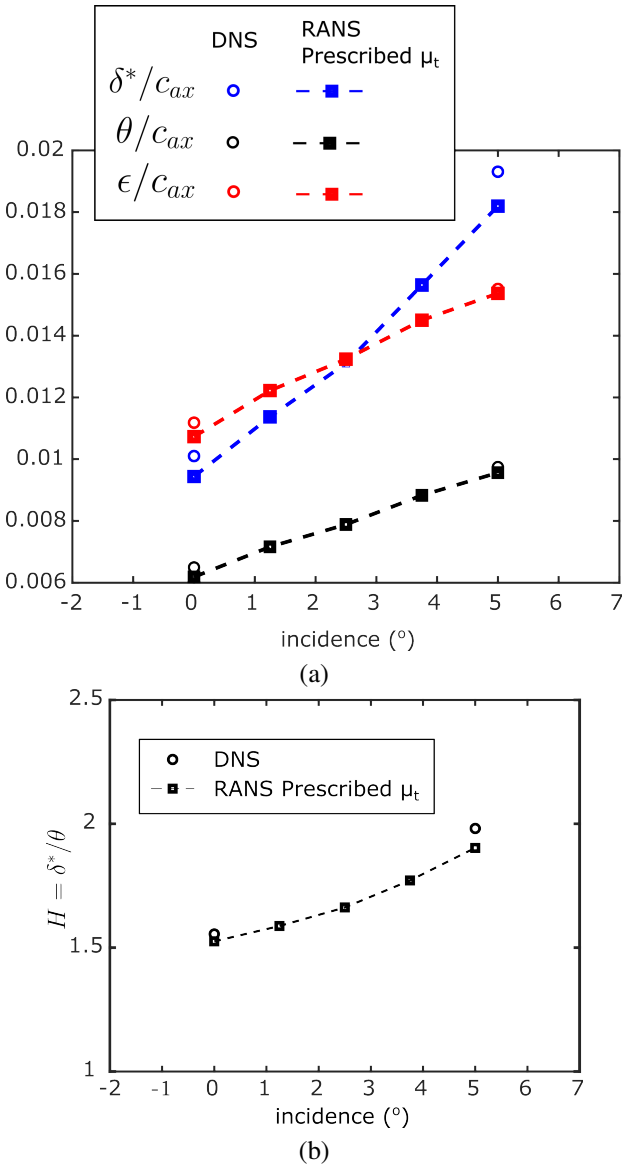


FIGURE 16. **Suction surface trailing-edge boundary-layer parameters from the DNS and from RANS calculations with prescribed eddy viscosity μ_t interpolated to each flow angle**

.

leads to earlier transition, the levels of eddy viscosity drop significantly at high incidence. The presence of a separation bubble clearly elevates the eddy viscosity over the aft suction surface at design incidence.

The eddy viscosity μ_t can be added to the molecular viscosity in order to obtain a RANS simulation with a prescribed μ_t. In general, this produces a steady flow and can be used to verify the accuracy of the Boussinesq approximation of the flow. This is done in *3DNS-gpu* by reading the prescribed μ_t at the start of the calculation and time-marching in the usual way.

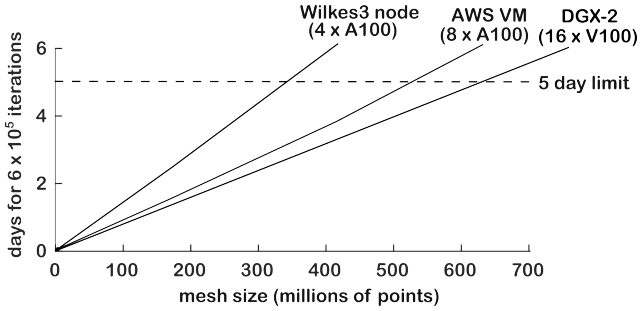


FIGURE 17. **Estimated time to compute 3.5 throughflows (6×10^5 time steps) with various mesh sizes on a single compute node on Wilkes3 and an AWS Virtual Machine ('p4d.24xlarge'). Note, it is assumed that the minimum cell size (and therefore time step) remains constant.**

For this investigation the eddy viscosity was interpolated onto a two-dimensional mesh of 0.9 million points. Figure 15 shows the suction-surface momentum thickness from the DNS and from the prescribed μ_t calculations. The agreement is very encouraging and is strong confirmation of the Boussinesq approximation. There is some difference incurred in the region of the bubble reattachment at the design incidence ($\alpha = 50.23^o$) and this is a likely result of the high level of unsteadiness which occurs in this region due to the development of shear layer instabilities.

Next is shown an example of how we might make further use of this prescribed eddy viscosity. Prescribed eddy viscosity simulations were run at a range of incidences between $0-5$ degrees. For each simulation, the eddy viscosity was determined by linear interpolation of the eddy viscosity with inflow angle α using the DNS results.

Figure 16 shows the variation in trailing-edge suction surface integral properties with incidence from the prescribed μ_t model and the DNS. The results show that the prescibed μ_t approach gives a useful way to extend the use of DNS data across the design space.

In this section it has been shown how data extracted from DNS has enabled a sweep of calibrated RANS simulations to be run using a mesh around 500 times smaller than the DNS. A similar approach can be adapted to explore a range of design parameters including pitch-to-chord and variations in loading style.

OPPORTUNITY FOR 'DESKTOP-DNS'

In order to demonstrate what is achievable on a single computer node, Figure 17 shows the compute time for the compressor blade case used in this paper on three systems: a single Wilkes3

Copyright © 2023 by ASME

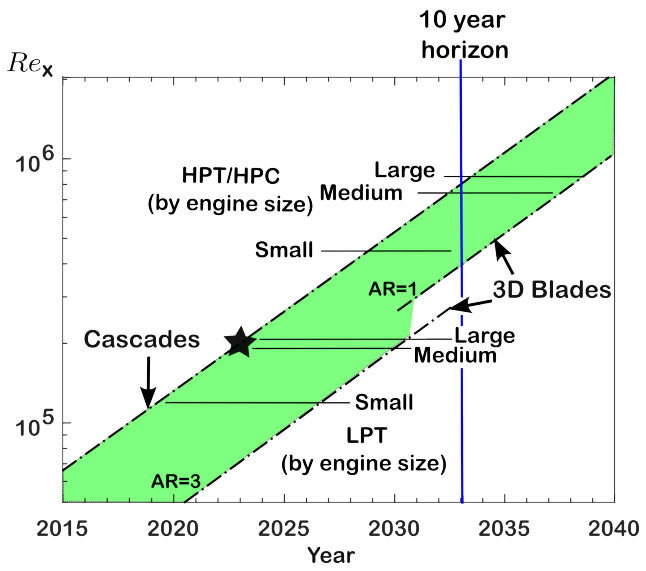


FIGURE 18. **Opportunity for 'Desktop DNS' assuming accessible Reynolds number doubles every 5 years. The ★ is an estimate of the achievable Reynolds number running on a single Wilkes3 node for 5 days. Engine Reynolds number are based on [18] for sea-level conditions.**

node [19]; an Amazon Virtual Machine instance [3]; and a DGX-2 system [20]. The figure shows the time required to simulate 6×10^5 iterations - for the case tested here this is sufficient to achieve statistical convergence (see Figure 8).

We can see that, using a single Wilkes3 node (which contains 4 A100 GPUs) we can run cases of up to 340M solution points within a working week (5 days). A single AWS Virtual Machine 'p4d.24xlarge instance' [3] contains 8 A100 GPUs, with this system simulations with around 540M points are achievable within the 5 day limit. Based on spot prices at the time of writing this paper, this would cost around $\$1,200$, or approximately $\$4,500$ based on on-demand pricing [3].

The third system tested was a DGX-2 which contains 16 V100 GPUs each with 32 GB memory and connected via NVLINK [20]. Although the V100 GPUs are less powerful than the A100s, Figure 17 shows that by leveraging the high speed connection of 16 GPUs on one machine, cases of around 630 million points can be run within the 5 day limit.

For DNS, compute cost scales approximately as Re^3. If we consider 5 days on 1 Wilkes3 node as a practical limit for 'Desktop-DNS' this suggests a feasible Reynolds number of $Re \simeq 190,000$ for cascade simulations on current hardware. This is assuming we wish to achieve similar accuracy to the case discussed in this paper.

Figure 18 illustrates the enormous potential of this. Using the results of this paper, the limits for running DNS on a single compute node are mapped-out with Reynolds number. Assuming computer performance doubles every 18 months (faster rates may be achieved with GPU developments) we would expect the accessible Reynolds number to double approximately every 5 years. Also shown is an indication of where different aero-engine/gas-turbine components operate by engine size based on [18] (although this is very dependent on type of machine and flight condition). The shaded region indicates the range of simulations from cascade simulations (of 10 percent chord in spanwise extent) to 3D blades with aspect ratio $AR = 3$ to indicate typical low-pressure turbine (LPT) blades and $AR = 1$, typical of high-pressure turbines (HPT) and core compressors (HPC). Full blade simulation will require larger solution points in order to resolve the flow across the whole span and endwall boundary layers; an assumption is made here that the number of solution points will be $10 \times AR$ larger than the equivalent cascade simulation at the same Reynolds number.

Figure 18 shows that over the next 10 years 'Desktop-DNS' could be used to resolve much of the engine, either as part-span cascade cases (for large HPTs and core compressors) or full blades for LPTs up to large engine sizes. The figure illustrates the potential for DNS to be a much more common-place tool to support aerodynamic modelling and design.

CONCLUSIONS

The paper shows that turbomachinery aerodynamicists now have access to unprecedented levels of predictive accuracy. There remain, however, significant 'barriers to entry' for businesses and wider groups of researchers to benefit from DNS, including - know how / training, access to the appropriate software/hardware, data-processing methods and storage. It is hoped that this paper and the associated 'Desktop-DNS' toolkit will help in some way to reduce this barrier.

ACKNOWLEDGEMENTS

The author is grateful to Alistair Senior (Whittle Lab) who kindly produced the compressor profile used in this paper. Many thanks to Filippo Spiga (NVIDIA) for help and advice with improvements to *3DNS-gpu* performance and also for providing data on the DGX-2 system. Thanks also to colleagues at the Whittle Lab in particular Jordi Gomez Alberti for computing support and Graham Pullan for feedback on the paper.

REFERENCES

[1] V. Michelassi, W. Rodi, J. Wissink. "Analysis of DNS and LES of Flow in a Low Pressure Turbine Cascade with Incoming Wakes and Comparison with Experiments." *Flow, Turbulence and Combustion* Vol. 69 No. 4 (2002): pp. 295–330. DOI 10.1023/A:1027334303200.

Copyright © 2023 by ASME

[2] Wissink, J. and Rodi, W. "Direct numerical simulation of flow and heat transfer in a turbine cascade with incoming wakes." *Journal of Fluid Mechanics* Vol. 569 (2006): p. 209–247. DOI 10.1017/S002211200600262X.

[3] "Amazon-Web-Services." Accessed 2023-09-04, URL `https://aws.amazon.com/ec2/spot/pricing/`.

[4] Sandberg, R. D. and Michelassi, V. "The Current State of High-Fidelity Simulations for Main Gas Path Turbomachinery Components and Their Industrial Impact." *Flow, Turbulence and Combustion* Vol. 102 No. 4 (2019): pp. 797–848. DOI 10.1007/s10494-019-00013-3.

[5] Sandberg, R. D. and Michelassi, V. "Fluid Dynamics of Axial Turbomachinery: Blade- and Stage-Level Simulations and Models." *Annual Review of Fluid Mechanics* Vol. 54 (2022): pp. 255–285. DOI 10.1146/annurev-fluid-031221-105530.

[6] Wheeler, A.P.S., Dickens, A.M.J. and Miller, R.J. "The effect of nonequilibrium boundary layers on compressor performance." *Journal of Turbomachinery* Vol. 140 (2018): pp. 101003–1–10.

[7] Przytarski, P.J. and Wheeler, A.P.S. "The Effect of Gapping on Compressor Performance." *Journal of Turbomachinery* Vol. 142 (2020): pp. 121006–1–10.

[8] Przytarski, P.J. and Wheeler, A.P.S. "Accurate Prediction of Loss Using High Fidelity Methods." *Journal of Turbomachinery* Vol. 143 (2021): pp. 031008–1–9.

[9] Spencer, R., Przytarski, P.J., Adami, P., Grothe, P. and Wheeler, A.P.S. "Importance of Non-Equilibrium Modelling for Compressors." *ASME Turbo Expo* Vol. 84928 (2021): pp. V02CT34A003–1–11.

[10] Liu, Q., Ager, W., Hall, C. and Wheeler, A.P.S. "Low Reynolds Number Effects on the Separation and Wake of a Compressor Blade." *ASME Turbo Expo* Vol. 84904 (2021): pp. V02AT31A031–1–10.

[11] J. M. Maynard, J. V. Taylor R. Wells, A. P. S. Wheeler. "Unsteady Structure of Compressor Tip Leakage Flows." *Journal of Turbomachinery* Vol. 145 No. 5. DOI 10.1115/1.4055769. 051005.

[12] F.Tosto, M. Pini, A.P.S. Wheeler. "High fidelity simulations and modelling of dissipation in boundary layers of non-ideal fluid flows." *4th International Seminar on Non-Ideal Compressible Fluid Dynamics City, University of London, 3-4th November 2022* .

[13] Tam, C. K.W. and Webb, J. C. "Dispersion-Relation-Preserving Finite Difference Schemes for Computational Acoustics." *Journal of Computational Physics* Vol. 107 No. 2 (1993): pp. 262–281. DOI https://doi.org/10.1006/jcph.1993.1142.

[14] Kennedy, C.A. and Gruber, A. "Reduced aliasing formulations of the convective terms within the Navier-Stokes equations for a compressible fluid." *J. Comput. Phys.* Vol. 227 (2008): pp. 1676–1700.

[15] Kim, J.W. and Lee, D.J. "Generalized Characteristic Boundary Conditions for Computational Aeroacoustics." *AIAA Journal* Vol. 38 No. 11 (2000): pp. 2040–2049. DOI 10.2514/2.891.

[16] Kim, J.W. and Lee, D.J. "Characteristic Interface Conditions for Multiblock High-Order Computation on Singular Structured Grid." *AIAA Journal* Vol. 41 No. 12 (2003): pp. 2341–2348. DOI 10.2514/2.6858.

[17] V. Michelassi, R. Pichler R. D. Sandberg, L. Chen. "Compressible Direct Numerical Simulation of Low-Pressure Turbines—Part II: Effect of Inflow Disturbances." *Journal of Turbomachinery* Vol. 137 No. 7 (2014): p. 071005. DOI 10.1115/1.4029126.

[18] Mayle, R. E. "The Role of Laminar-Turbulent Transition in Gas Turbine Engines." *Turbo Expo: Power for Land, Sea, and Air* Vol. 5. DOI 10.1115/91-GT-261.

[19] "Cambridge-Service-for-Data-Driven-Discovery." Accessed 2023-09-04, URL `https://www.hpc.cam.ac.uk/high-performance-computing`.

[20] "NVIDIA DGX-2." Accessed 2023-09-04, URL `https://www.servethehome.com/nvidia-nvswitch-details-at-hot-chips-30/nvidia-dgx-2-pcie-network-diagram/`.

[21] Visbal, M. R. and Gaitonde, D. V. "High-Order-Accurate Methods for Complex Unsteady Subsonic Flows." *AIAA Journal* Vol. 37 No. 10 (1999): pp. 1231–1239.

APPENDIX
Mesh generation

The use of high order finite differencing means that numerical schemes require a smooth distribution of solution points. This can be achieved by solving a form of the Poisson equation to determine the solution point positions x, y:

$$x = \frac{\left(\alpha\bar{x}^i - \frac{1}{4}\beta x_{ij} + \gamma\bar{x}^j\right)}{(\alpha+\gamma)}; \tag{10}$$

$$y = \frac{\left(\alpha\bar{y}^i - \frac{1}{4}\beta y_{ij} + \gamma\bar{y}^j\right)}{(\alpha+\gamma)}; \tag{11}$$

$$\alpha = x_j^2 + y_j^2; \tag{12}$$

$$\beta = x_i x_j + y_i y_j; \tag{13}$$

$$\gamma = x_i^2 + y_i^2; \tag{14}$$

x_i, y_i, x_j, y_j are finite difference approximations of the gradients of x and y in the grid directions i, j, and x_{ij}, y_{ij} are cross derivatives. The quantities shown with an overbar (such as $\bar{x}^i$) are an average in the grid direction indicated by the superscript. At a grid point within the block, a second-order central difference using surrounding points can be used to set the quantities α, β, γ

Copyright © 2023 by ASME

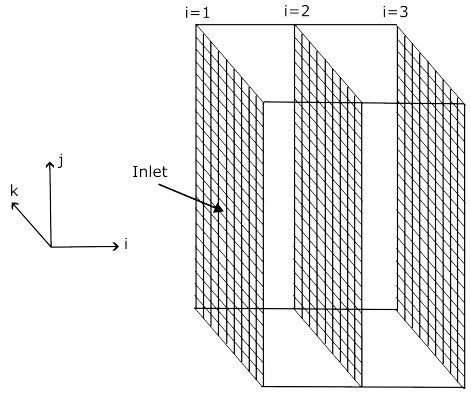


FIGURE 19. **Schematic of inlet domain used when determining inflow turbulence.**

and the average quantities $(\bar{x}^i, \bar{x}^j, \bar{y}^i, \bar{y}^j)$ can be determined from the sum of the values on either side of the grid point.

Across block boundaries, the values of x, y in the neighbouring block are communicated in order to solve for the values required on the boundaries. Where there is no neighbour, a boundary condition can be applied - at a wall this is orthogonality to the surface and a near-wall distance requirement. At inlets and exits it is normally sufficient to extrapolate grid lines from the interior grid. Regions near multi-block corners require extra care; in these regions it is normally necessary to distribute points more closely around the corner point. The grid smoothing routine is applied within an iterative loop, first on a coarse grid and then successively finer meshes. Each mesh refinement uses a bilinear interpolation to determine the new solution points.

Inflow turbulence

There are many methods for imposing inflow turbulence in DNS and LES. Synthetic turbulence methods typically add perturbations according to a pre-determined Fourier series, with coefficients derived empirically. Other methods impose filtered random numbers. It can sometimes be the case that the generation and filtering of random numbers adds significantly to compute costs. In order to mitigate this, turbulent fluctuations can be pre-computed and read in at run time. Without detailed knowledge of the true turbulence spectrum within a machine, it seems pragmatic to choose a simple method which allows some control to allow sensitivities to be analysed. An inexpensive method for implementing inflow turbulence is described here. This eliminates the need for intrisic random number generators entirely and makes use of a smoothing routine to provide spatial correlation. The method exploits the chaotic sequence $s_{n+1} = \sin(2\pi s_n)$ where s_0 is a seed value. The values of this sequence are bounded to be within the range $|s| < 1$. Any value (excluding 0) with magnitude below unity can be used as an initial seed value. This sequence of numbers is mapped onto a three-dimensional grid with grid dimensions matching the inlet of the computational domain in the j,k plane and with three points in the i direction (see Fig. 19). A Laplacian smoothing operation is then performed in the i, j, k directions. Averaging of the values over periodic interfaces is performed during this operation to ensure periodicity. The filtered values are given by

$$f^*_{ijk} = \alpha f_{ijk} + (1-\alpha)(f_{i+1,j,k} + f_{i+1,j,k} + f_{i,j+1,k} + f_{i,j-1,k} + f_{i,j,k+1} + f_{i,j,k+1})/8, \tag{15}$$

where the filter coefficient $\alpha = 0.5$. This essentially acts as a Gaussian filter.

This process is used to set v' and w'. The value of u' is then determined by setting the divergence to zero and using numerical integration of equation 16.

$$\Delta u' = \int -\left(v'_y + w'_z\right) dx \tag{16}$$

Central differencing is used to determine the gradients v'_y and w'_z, while a first-order numerical integration gives u', taking $u' = 0$ at the inlet plane ($i = 1$). An amplification factor is then used to scale the fluctuations, before adding them to the target velocity in the characteristic boundary condition within the inlet domain. The process is called typically every 500 iterations with the seed value being stored and updated each time. The frequency determines the streamwise length-scale of the perturbations, and can be varied to ensure approximately isotropic fluctuations.

The method naturally gives a range of scales which depends on the mesh resolution - finer meshes which are capable of supporting smaller-scales will naturally produce a greater range of scales.

Characteristic interfaces

In *3DNS-gpu* characteristic interfaces are imposed across multi-block interfaces to provide a high-order treatment at the boundary. A characteristic decomposition is also used at inflow and outflow regions. These are discussed next.

Multi-block boundaries Across multi-block boundaries, the characteristic interface method of Kim et al. [15, 16] is used in order to enable the use of curvi-linear grids. At a boundary, a one-sided stencil is used to compute the flux derivatives. A characteristic decomposition of the flux derivatives is then performed to determine the wave directions and magnitudes at the interface. The incoming characteristics are then used to reconstruct the flux derivatives in order to provide a higher order treatment at the boundary.

Copyright © 2023 by ASME

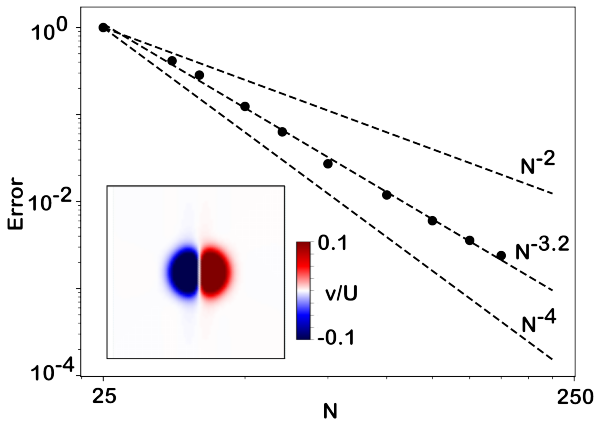


FIGURE 20. **Effect of multi-block boundary on solution accuracy for the case of a convecting vortex. In this case the error is determined as $e = \max|v_f - v_i|$; where v_f and v_i are the swirl velocities along the line $y = 0$ at the final and initial times.**

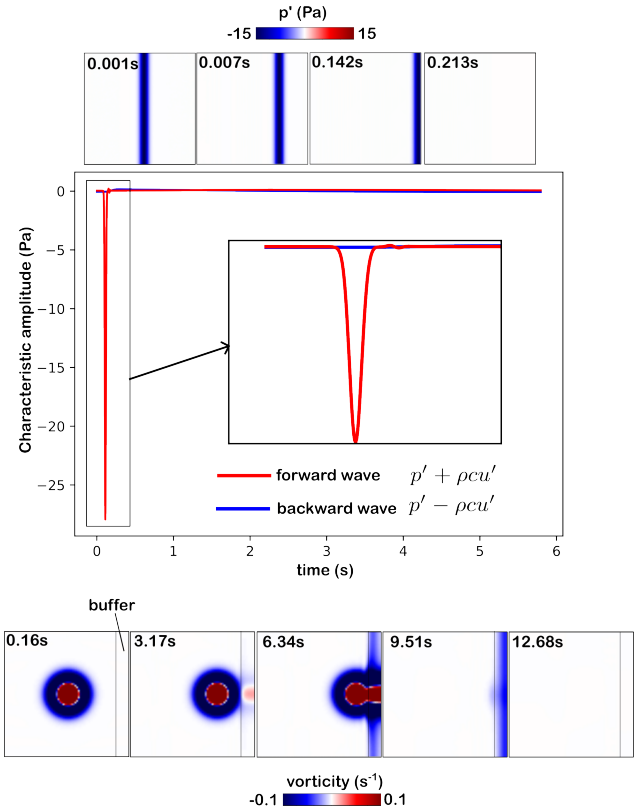


FIGURE 21. **Characteristic boundary condition tests: plane wave (top) ; convecting vortex (bottom).**

The accuracy of the implemented method was tested using a convecting vortex simulation (matching [21]). Figure 20 shows how the error varies with mesh points. A best-fit line through the points shows an order of accuracy higher than 3^{rd} order.

Inflow and outflow non-reflecting boundaries A characteristic decomposition is also used at inflow and outflow boundaries in order to control reflected waves. Incoming wave amplitudes are modified in accordance with the required boundary condition and then the flux derivative is reconstructed with the modified wave amplitudes. In *3DNS-gpu* a velocity vector and entropy are imposed at the inlet, and an exit pressure is imposed on the outlet. The wave amplitudes imposing the boundary condition are blended with the waves computed from the interior of the domain according to a blending function ϕ which is ramped-up linearly to the inlet: $L = \phi L_{set} + (1-\phi)L_i$ where L is any of the five characteristic terms, L_{set} is the imposed characteristic term and L_i is the term computed from the interior of the domain. The blending function ϕ across this buffer region varies linearly with the grid direction across a fixed number of points normal to the boundary (typically around 10-15) from 1 (at the boundary) to 0.

The size of the imposed characteristic term is controlled by a constant K; $L_{set} = K(f - f_{set})$, where f and f_{set} are a computed flow variable (such as pressure) and the value required by the boundary conditions. Reducing K reduces reflections at the inlet/exit, but can allow the conditions to drift unacceptably making accurate statistical analysis impossible. Larger values of K enforce the boundary conditions more stiffly but can lead to undesirable reflections which can excite instabilities in the flow. Previous work suggests values of $K \approx 0.25(1-M^2)c/L$, where c is the speed of sound, L is a characteristic length for the computational domain and M is a characteristic Mach number. However the author's experience is that the optimum choice of K is a compromise which is normally case dependent and can require some experimentation using coarse simulations and analysis of the statistical convergence of the flow.

A test of the outflow boundary was performed using a plane pressure wave set by an initial flow: $p = p_\infty - \varepsilon\rho_\infty {c_\infty}^2 \exp(-\ln 2(x^2)/9R^2)$; $u = U - \rho_\infty c_\infty (p - p_\infty)$; $v = 0$. $U = 1m/s$, $M_\infty = U_\infty/\sqrt{\gamma p_\infty/\rho_\infty} = 0.1$, $\rho = \rho_\infty = 1.225kg/m^3$, $p_\infty = 10^5 Pa$, $R = 1m$, $\varepsilon = 10^{-4}$. The domain used was a square uniform mesh of $L = 50m$ with 125×125 points. Figure 21 shows the forward and backward wave amplitudes, thus demonstrating the effectiveness of the method in removing reflections. Figure 21 also shows the response of the exit boundary to a vortex, using the same initial flow as in the multi-block interface test discussed above [21]. The effect of the buffer on dispersing the vortex before leaving the domain can be clearly seen. Again, the method effectively removes any significant reflection upstream of the buffer region.

Copyright © 2023 by ASME