3DNS-GPU
User Guide*

Andrew P S Wheeler
Professor of Aerothermal and Fluids Engineering
University of Cambridge, UK
e-mail: aw329@cam.ac.uk
May 2023

*this guide is written for the publicly released version of ***3DNS-GPU*** as part of the Desktop-DNS open toolkit

Contents

1. Introduction

3DNS (3-Dimensional Navier-Stokes solver) is a higher-order finite-difference compressible flow code. The flow is solved in curvi-linear coordinates, and metrics are used to convert to ordinary Cartesian coordinates. Skew-split differencing is used to reduce dispersion errors, which enables very low levels of filtering. A 4-stage Runge-Kutta method is used to progress the flow explicitly in time. A 9-point explicit filter is used to filter unresolved scales. Non-reflecting boundary conditions are used for inlet and exit boundary conditions. Buffer zones are applied within inlet and exit blocks respectively (these can be adjusted by the user). Multi-block interfaces use curvi-linear characteristic interfaces to provide a high order treatment across block interfaces. The code is well suited to turbomachinery flow problems and has been used in a range of studies of turbomachinery aerodynamics (for more details see [1-13])

Three versions of the ***3DNS*** code are now available:

- ***3DNS-GPU* branch** – This version is GPU accelerated using OpenACC. Communication across GPU cards is achieved using MPI. The geometry is restricted to be invariant in the z-direction.
- ***3DNS* main branch** – This is a pure MPI parallelization. The geometry is restricted to be invariant in the z-direction although options for tip gap and endwalls are available.
- ***4DNS* branch** – This is a pure MPI parallelization. The geometry can vary in all directions (i.e. fully curvilinear).

The information below is for the publicly released version of *3DNS-GPU* as part of the Desktop-DNS open toolkit. The publicly released version of *3DNS-GPU* has some features switched-off - please get in touch if you wish to use other versions of 3DNS.

2. Input data

The follow sections outline the structure for the input data.

2.1. Grid files

A grid file is required for each block. The format for each grid file 'grid_#.txt'

x(i,j) y(i,j)	x, y values in metres
Repeat for each i,j value (i leading)	

2.2. Input file

The input data is read from file 'input_gpu.txt'. All units are SI. The file has the following format:

Section	Quantities	Description
1	nblocks, kproc	Number of blocks – maximum of 20 for public release of 3DNS-GPU Number of GPU tasks in the spanwise (k) direction (see section 5)
2	*For each block input the following data*	
2.1	nib,njb,nkb	Grid dimensions in i,j,k. nkb must be either >=10 or 1 for 2D simulations
2.2	im_type, ip_type, jm_type, jp_type	Patch types for i=1,i=nib etc. (0=interface,1=inlet,2=static pressure exit,3=no-slip-wall).

2.3	*For each interface input the following data*	
	next_block, next_type	For each interface list the next block which this patch interfaces with and the type (1=im, 2=ip, 3=jm, 4=jp). New line for each interface.
	Repeat 2.3 for each block interface	
	Repeat 2 for all blocks	
3	Ncorner	Number of multi-block corner groups (usually 4 for a 9 block mesh)
4	*For each corner group input the following data*	
4.1	ncornerblocks, cor_type	Number of blocks interfacing at corner. cor_type (not used)
4.2	*For each block in this corner group, input the following data*	
	nbcorner, ic, jc	Block number, I,J coordinates
	Repeat 4.2 for each block in this corner group	
	Repeat 4 for all corner groups	
5	nblockgroups	Number of block groups – this determines the number of GPU tasks in the i-j plane (see section 5)
6	*For each block group input the following data*	
6.1	nb_block_group	Number of blocks in current block group
6.2	blocks	List blocks in current block group
	Repeat 6 for each block group	
7	niter, nwrite, ncut	Number of iterations, frequency to write flow files, frequency to write spanwise cut files
8	CFL, sigma	CFL (usually=1 but values of 1.5 can sometimes be achieved), 8th order filter coefficient (usually=0.03). The filter is used to cut-off unresolved scales which lead to dispersion errors. Lower values of filter coefficient will reduce artificial dissipation, but low values (<0.01) may lead to the growth in spurious oscillations.
9	Toin,poin,pexit,vinlet, alpha, gamma, aturb, ilength,radprof,dum	Initial guess of inlet To and Po (sets inlet entropy – see section 7.2) exit static P Inlet velocity vinlet Inlet yaw angle alpha (degrees) Inlet pitch angle gamma (degrees) Factor multiplying inflow turbulence aturb (values of 10-50 are typical – see section 6) Turbulence updated every ilength iterations (typically 500 – see section 6) radprof – not used for public release of 3DNS-gpu dum – not used for public release of 3DNS-gpu
10	gam,cp,mu_ref,Tref,mu_s, prd	ratio of specific heats, specific heat at constant pressure, coefficients for Sutherlands law, Prandtl number.
11	span, fexpan	spanwise extent in k-direction fexpan must = 1.0 for public release of 3DNS-gpu
12	irestart, istat	Read a restart file if equal to 1, restart a 3d solution from a 2d solutions if irestart=12. Set to 0 if starting from scratch. istat sets the statistics gathering (see section 8): istat = 0, no statistics istat = 1, 3D statistics (time-average 3D flow)

		istat = 2, 2D statistics (time and spanwise average flow, including entropy budgets)

2.3. Probe files

Monitor points are set using a file called ‘probe.txt’. The format of this file is as follows:

Nprobe, nskip — number of probes, data written every nskip time steps

now for each nprobe list the following

Nb_probe, i_probe, j_probe, k_probe — Block number, I,J,K value of probe within block

If the probe file is not present or empty, then no probes are written. The output probe files (probe_#) are appended to every time a simulation is run.

2.4. Characteristic inlet and outlet boundary condition files

The inlet and outlet characteristic boundary conditions can be modified using the files ‘in_buffer.txt’ and ‘out_buffer.txt’.

The format for each file is as follows:

nibuf, Lwave — Number of i-planes in buffer region

Lwave defines a length-scale (as a proportion of the total domain length) which sets the stiffness. Smaller values give stiffer inlets/exits.

If ‘in_buffer.txt’ and/or ‘out_buffer.txt’ are not present or empty, then the default values of nibuf = 10, and Lwave (inlet) = 0.01 and Lwave (outlet) = 1.0 are used. See notes below for more details.

2.5. Inlet and exit smoothing

The characteristic inlet/exit non-reflecting boundary conditions remove plane wave reflections from the inlet and exit boundaries, however oblique waves can sometimes persist. In order to mitigate this, within the inlet and exit buffer zones, the pressure is driven to be pitch-wise uniform using a factor inex_fac. The value of inex_fac is read from the file ‘inex.txt’. If ‘inex.txt’ is not present or empty, then the default value of inex_fac=0.001 is used. Values above 0.005 are not recommended.

2.6. RANS mode

3DNS-gpu has a simple RANS mode option for running steady simulations. The format for the ‘rans.txt’ file is:

If_rans — if_rans RANS mode (0=off,1=mixing-length, 2=prescribed eddy viscosity)

Speed_up — limit local time-step to be speed_up time larger than the minimum time-step. Larger values will accelerate

	convergence, smaller values will be more stable. Usually around 3-5.
Nrans_file	Number of prescribed rans files to read-in when if_rans=2. Must be either 1 or 2 when if_rans=2
Path1	Path to first case with rans_# files containing the prescribed eddy viscosity
Path2	Path to second case with rans_# files (when nrans_file=2)
Fac	factor used to interpolate between two cases when nrans_file=2. A linear interpolation is used to determine the prescribed eddy viscosity between the two cases defined above. For example, when fac=0, case 1 values are used, when fac=1, case 2 values are used, when fac=0.5, the average of case 1 and 2 is used. If nrans_file=1, fac is not used.

If the ‘rans.txt’ file is not present or empty, then the default is if_rans=0 (i.e. time-accurate direct simulation). See section 10 for more details.

3. Output files

3.1. Text files

monitor.txt – convergence history, written every 100 time steps. Values are written for the central i,j point for Block 1 (at k=1). The format is:

iteration, time, ρ, ρu, ρv, ρw, Et

mean_time.txt – this file stores the integration time for each mean flow file. The format is:

mean_count, time, mean_time

mean_count is a counter to identify the mean (statistics) file, time is the solution time when the mean files are written and mean_time is the integration time for the simulation over which the statistics are gathered.

kslice_time.txt – this file stores the time for the kcut files. The format is:

file_count, time, kslice

file_count is a counter to identify the kcut file, time is the solution time at which the slice is written and kslice is the value of k for the slice.

span_#.txt – the spanwise distribution of points for each block, the format is:

z, dzk

z is the spanwise location and dzk is the spanwise grid spacing (both in metres)

blockdims.txt – the block dimensions:

nib, njb, nkb

3.2. Binary files (float 64)

flow_# - stores the flow field for each block (order = nvar, i,j,k)

flow variables q = (ρ, ρu, ρv, ρw, Et)

rans_# - stores the eddy-to-laminar viscosity ratio μ_t/μ (order = i,j,k). Only written when if_rans>0.

probe_# - stores the flow at each # probe point

time, ρ, ρu, ρv, ρw, Et

kcut_#_## - stores the flow for each # block and ## counter (order = nvar, i,j)

flow variables q = (ρ, ρu, ρv, ρw, Et)

mean_#_## - stores the time-average flow for each # block and ## counter (order = nvar, i,j)

flow variables q = (ρ, ρu, ρv, ρw, Et)

mean2_#_## - stores the time and spanwise average flow for each # block and ## counter (order = nvar, i,j)

flow variables (nvar=1-11,i,j)

q = (ρ, ρu, ρv, ρw, Et, ρuu, ρvv, ρww, ρuv, ρuw, ρvw)

entropy budget (nvar=12-17,i,j)

q2 = (ρus, ρvs, ρws, dissT, qxT, qyT, qzT)

where ρus, ρvs, ρws are the entropy flux terms, dissT is the dissipation divided by temperature, and qxT, qyT, qzT are the heat fluxes divided by temperature.

4. Numerical scheme

All branches of 3DNS solve the flow on curvi-linear structured grids in strong conservation form. The algorithm uses explicit differencing with a 4th order Tam and Webb [14] 7-point stencil scheme and an 8th order explicit filter. The strength of the filter is determined by the filter coefficient set in in the 'input_gpu.txt' file (sigma).

Skew-split differencing is applied to the inviscid fluxes using the method of Kennedy and Gruber [15] to reduce dispersion errors. Characteristic interfaces using the method of Kim and Lee [16,17] are applied at inflow and outflow boundaries, as well as at multi-block boundaries). Time integration is performed using a standard low-storage 4-step Runge-Kutta scheme.

For the public release 3DNS-gpu, the geometry is restricted to be invariant in the z-direction. The multi-block interfaces must be contiguous or one pitch apart, with mesh points matching on either side of the boundary. A further restriction is that the start and end of the interface must have the same positions on either side of the interface (i.e., if side 1 interface is from y(1)=0 to y(njb)=1, side 2 must also be y(1)=0 to y(njb)=1.).

The flow is assumed periodic in the k-direction unless fexpan < 0 (not available for the publicly released version of 3DNS-gpu).

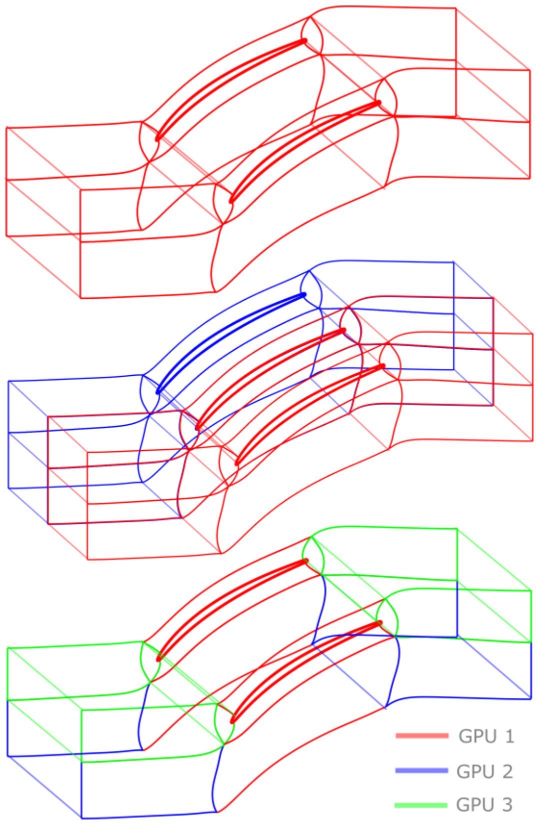

FIGURE 1. Different parallelization options for a 9-block domain using either 1, 2 or 3 GPU cards

5. Parallelization strategy

In 3DNS-gpu, parallelization is achieved with both MPI and OpenACC directives in order to enable the use of both CPUs and GPUs. For the public release 3DNS-gpu the topology and geometry is assumed to be two dimensional with multi-block boundaries restricted to be normal to the homogeneous (spanwise) direction (other versions can be made available on request). 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.

Figure 1 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 (by setting kproc>1 in the input_gpu.txt file).

Blocks or sub-blocks can also be grouped onto GPUs in order to parallelize in the blade-to-blade plane. The number of block groups and blocks in each block group are set in the 'input_gpu.txt' file (as described above).

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 total number of GPU tasks is given by the product of kproc and nblockgroups:

np = kproc x nblockgroups

Usually there is 1 task per GPU, however it is possible to have multiple tasks per GPU using NVIDIA MPS – this can sometimes yield performance improvements.

6. Inflow turbulence

The public release of 3DNS-gpu includes an inexpensive method for implementing inflow turbulence. The method avoids the use of random number routines which can add significant compute time. The method exploits the chaotic sequence

$\sigma_{n+1}= \sin(2\pi\ \sigma_n)$

where σ_0 is a seed value.

The values of this sequence are bounded to be within the range $|\sigma|< 1$. 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. 2). A 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

$f_{i,j,k}' = \alpha\ f_{i,j,k} + (1-\alpha)(f_{i+1,j,k} + fi_{+1,j,k} + f_{i,j+1,k} + f_{i,j-1,k} + f_{i,j,k+1} + f_{i,j,k+1})/8$

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.

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 (aturb) 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 (set by ilength) 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.

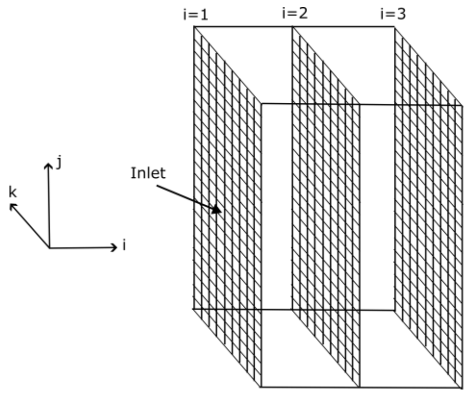

FIGURE 2. Schematic of inlet domain used when determining inflow turbulence.

7. Characteristic interfaces

In 3DNS 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.

7.1. 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.

7.2. 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

$L = \varphi L_{set} + (1-\varphi) L_i$

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, set by nibuf in 'in_buffer.txt' and 'out_buffer.txt') from 1 (at the boundary) to 0. The flow in these regions is to some degree non-physical and should not be included in any analysis of the flow.

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 $K \approx 0.25(1-M^2)c/L_{ref}$, where c is the speed of sound, L_{ref} is a characteristic length for the computational domain and M is a characteristic Mach number [15, 16]. 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.

For this reason the values of K at inlet and exit can be independently modified using the values of L_{wave} in the 'in_buffer.txt' and 'out_buffer.txt' files (see also section 2.4). When L_{wave}=1, K is equal to the default value $K_{def}=0.25(1-M^2)c/L_{ref}$, for other values of L_{wave}, the value of K is scaled according to $K = K_{def} / L_{wave}$.

Smaller values of L_{wave} give rise to a stiffer inlet (larger K).

It is important to note that the inlet To and po set in the input file sets the entropy of the inlet flow – not the stagnation conditions. The inlet flow is driven towards the inlet entropy, inlet velocity and angle. After the initial transient when starting the solution, the flow should converge to the desired values at the inlet and exit. However the user is strongly encouraged to monitor (using the probes) the convergence of the flow to verify that the desired conditions are achieved.

Inlets can only be on im boundaries, exits can only be on ip boundaries.

8. Gathering statistics

Statistics are gathered by setting istats to either 1 or 2. When istats=1, the conserved quantities are simply time-averaged and the three-dimensional flow is written to the mean_#_## files (see section 3.2). A counter is used to increment the file name every time the simulation is run – this means the flow can be summed over several simulations.

The three-dimensional time average flow is usually of limited use. Instead the option istats=2 is more commonly used. In this case flow quantities are both time and spanwise averaged – for cases where the geometry is homogenous in the spanwise direction this is the preferred choice. A far greater number of statistical quantities can be stored for the same file size compared to the 3D statistics files.

The mean2_#_## files contain both the conserved quantities and also terms which enable the Reynolds stresses and entropy budget to be computed (see section 3.2). A counter is used to increment the file name every time the simulation is run so that the quantities can be summed over several simulations.

The integration time is written to the 'mean_time.txt' file, which is appended to each time the simulation runs (provided istats > 0).

9. Restarting from a previous solution

Setting irestart=1 means that the flow_# files will be read in add the start of the simulation.

It is possible to start a 3D simulation from a 2D simulation by setting irestart=12. In this case the flow is extruded in the spanwise direction – governed by the number of spanwise points (nk) and spanwise extend (span) in the 'input_gpu.txt' file.

When irestart=12 an initial spanwise pulse is added to the flow variables when the flow is extruded – this is to excite any three-dimensional global instabilities in the flow in order to break-up the 2D flow.

$f = f\,(1 + \text{pulse})$

where f contains (ρ, ρu, ρv, ρw, Et) and pulse = -10^{-8} for spanwise positions from k=5 to 10, otherwise pulse = 10^{-8}. This introduces a small-amplitude rectangular pulse in the flow and therefore contains a wide range of frequencies.

10. Running in 'RANS' mode

Running in RANS mode provides a means of establishing an initial steady flow or for assessing the accuracy of the Boussinesq assumption for a particular case, using a prescribed eddy viscosity determined from a prior DNS or LES. In both cases, local time-stepping is used to accelerate convergence to a steady state (see section 2.6).

10.1. RANS mode 1

When if_rans=1, the classic mixing-length model is used – in which case the eddy viscosity is determined based on the wall distance and strain magnitude

A mixing length limit is set at 0.5% percent of the domain length L_{ref}

$l_{mix\_lim} = 0.005\, L_{ref}$.

This is used to set an outer eddy viscosity

$\mu_{to} = 0.1681\, \rho\, l_{mix\_lim}^2\, |S|$

where |S| is the strain magnitude.

The near-wall eddy viscosity is set to

$\mu_{ti} = 0.1681\, D^2\, \rho\, l_{mix}^2\, |S|$

where l_{mix} is the minimum of the wall distance and l_{mix_lim} and D is the van Driest Damping factor

$D = 1 - \exp(-y^+/A^+)$.

The value of A^+ is set to 26.

A simple method for accounting for laminar flow near leading-edges is used based on an estimate of the skin friction coefficient c_f, which amplifies the van Driest damping when $c_f >$ 0.0075,

$A^+ = A^+\, (c_f/0.0075)^4$

In addition, the eddy viscosity is set to zero near stagnation points when the wall normal velocity is larger than the near-wall tangential velocity.

A limit of 1/200 is applied to the laminar-to-turbulent eddy viscosity ratio.

10.2. RANS mode 2

When if_rans=2, a prescribed eddy-to-laminar viscosity ratio is read in from the rans_# files (1 per block). This mode is useful when the eddy viscosity has been determined from a prior unsteady simulation.

In this mode, users have two further options – the first is to set nrans_file=1, which assumes only 1 set of rans_# files are read in. Setting nrans_file=2 means that two sets of rans_# files are read in and the value of fac is used to interpolate between the two cases – see section 2.6.

11. Running jobs

Provided you have the correct software environment, running 3DNS requires a simple mpirun command:

- mpirun –n NP threedns

Where NP is the number of tasks. NP needs to be consistent with the 'input_gpu.txt' file (see section 5).

Check the tables below for different system options.

Amazon AWS

Executible name	**NVIDIA HPC SDK toolkit**	**Instances tested on**
threedns_aws_ccall	Version 23.3	p3.2xlarge p3.8xlarge

Cambridge Wilkes3

Executible name	**NVIDIA HPC SDK toolkit**
threedns_wilkes_ccall	21.7

Docker container

Executible name	**Container**	**NVIDIA HPC SDK toolkit**
threedns_ccall	nvcr.io/nvidia/nvhpc:23.1-devel-cuda_multi-centos7	23.1

12. References

12.1. Publications using 3DNS

[1] Wheeler A.P.S. (2023) 'Desktop-DNS: An open toolkit for turbomachinery aerodynamics', ASME Turbo Expo 2023, ASME Paper no. GT2023-1023647

[2] Przytarski P. J. et al (2023) ‘The Role of Turbulence Scales in Mechanical Energy Budgets on High Fidelity Simulations of Compressors’, ASME Turbo Expo 203, ASME Paper no. GT2023-102767

[3] Taylor J. V. et al. (2023) ‘Compressor Tip Leakage Mechanisms’, ASME Turbo Expo 2023, ASME Paper no. GT2023-103005

[5] Maynard et al. (2023) ‘Unsteady structure of compressor tip leakage flows’, J. Turbomachinery, vol. 145, no. 5. https://doi.org/10.1115%2F1.4055769

[6] Spencer R. A. et al. (2023) ‘Importance of Non-equilibrium modeling for compressors’, J. Turbomachinery, vol. 145, no. 4. https://doi.org/10.1115%2F1.4054813

[4] Tosto F. et al (2022) ‘High fidelity simulations and modelling of dissipation in boundary layers of non-ideal fluid flows’, 4th International Seminar on Non-Ideal Compressible Fluid Dynamics, 3-4 Nov. 2022.

[7] Liu Q. et al. (2022) ‘Low Reynolds Number Effects on the Separation and Wake of a Compressor Blade’. J. Turbomachinery, vol. 144, no. 10. https://doi.org/10.1115%2F1.4054148

[8] Przytarski P. J. et al (2021) ‘Accurate Prediction of Loss Using High Fidelity Methods’, J. Turbomachinery, vol. 143, no. 3. https://doi.org/10.1115%2F1.4050115

[9] Spencer R. A. (2021) ‘Improving turbomachinery loss prediction using high fidelity CFD’ PhD Thesis, University of Cambridge, https://doi.org/10.17863/CAM.76867

[8] Przytarski P. J. et al (2021) ‘Data-driven analysis of high fidelity simulation of multi-stage compressor’ Proceedings of XIVth European Conference on Turbomachinery Fluid dynamics & Thermodynamics

[10] Przytarski, P. J. (2021) ‘High Fidelity Simulation of Loss Mechanisms in Compressors’. CFD’ PhD Thesis, University of Cambridge, https://doi.org/10.17863/CAM.76086

[11] Przytarski P. J. et al (2020) ‘The Effect of Gapping on Compressor Performance’, J. Turbomachinery, vol. 142, no. 12. https://doi.org/10.1115%2F1.4047933

[12] Jardine L. J. (2020) ‘The Effect of Heat Transfer on Turbine Performance’ PhD Thesis, University of Cambridge, https://doi.org/10.17863/CAM.56517

[13] Wheeler A. P. S. et al (2018) ‘The Effect of Nonequilibrium Boundary Layers on Compressor Performance’, J. Turbomachinery, vol. 140, no. 10. https://doi.org/10.1115%2F1.4040094

12.2. Other References

[14] Tam, C. K.W. and Webb, J. C. “Dispersion-RelationPreserving 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.

[15] 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.

[16] 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.

[17] 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.