Problem C3.3 Report: Direct Numerical Simulation of the Taylor Green Vortex

Brian C. Vermeire, Freddie D. Witherden, and Peter E. Vincent

November 14, 2014

Department of Aeronautics, Imperial College, London, UK

1 Code Desctription

We are using PyFR[10], which is an open-source Python-based framework for solving advection-diffusion type problems on streaming architectures using the Flux Reconstruction scheme of Huynh[5]. PyFR is platform portable via the use of a domain specific language based on Mako templates. This means PyFR can run on AMD or NVIDIA GPUs as well as CPUs. A summary of the functionality of PyFR is given in Table 1

Table 1: Functionality of PyFR v0.2.3

Systems	Compressible Euler and Navier Stokes
Dimensionality	2D, 3D
Element Types	triangles, quadrilaterals, hexahedra, prisms, tetrahedra
Platforms	CPU clusters, Nvidia GPU clusters, AMD GPU clusters
Spatial Discretization	arbitrary order Flux Reconstruction
Temporal Discretization	explicit Runge-Kutta
Flux Differentiation	Lagrange polynomial
Precision	single, double
Riemann Solvers	Rusanov, HLL[4], HLLC[8], Roe[7]
Viscous Interface	LDG[2]
Parallel Structure	MPI
Post Processing	Paraview

2 Case Summary

2.1 Test Case

The initial flow field is given by:

$$u = V_0 \sin\left(\frac{x}{L}\right)\cos\left(\frac{y}{L}\right)\cos\left(\frac{z}{L}\right), \tag{1}$$

$$v = -V_0 \cos\left(\frac{x}{L}\right)\sin\left(\frac{y}{L}\right)\cos\left(\frac{z}{L}\right), \tag{2}$$

$$w = 0, \tag{3}$$

$$p = p_o + \frac{\rho_0 v_0^2}{16}\left(\cos\left(\frac{2x}{L}\right) + \cos\left(\frac{2y}{L}\right)\right)\left(\cos\left(\frac{2z}{L} + 2\right)\right). \tag{4}$$

The Reynolds number, defined as $Re = \frac{\rho_0 V_0 L}{\mu}$, is equal to 1600. A compressible flow solver is used and the fluid is specified with $\gamma = 1.4$ and $Pr = \frac{\mu c_p}{\kappa} = 0.71$. The initial Mach number is nearly incompressible at $Ma = \frac{V_0}{c_0} = 0.1$ given a constant initial temperature field of T_0. The simulation is run until a final time of $t_c = 20$, where one non-dimensional time unit is defined as $t_c = \frac{L}{V_0}$.

2.2 Mesh and Solver

For this abstract we use P1 to P8 schemes with structured hexahedral elements over $-L\pi \leq x, y, z \leq +L\pi$. Each of these are generated to provide $\approx 256^3$ degrees of freedom, as shown in Table 2. The interface fluxes are LDG and HLL and Gauss-Legendre solution points are used. A five-stage fourth-order adaptive Runge-Kutta scheme[1, 3, 6] is used with a maximum and relative error tolerances of 10^{-6}.

Table 2: Mesh and Degrees of Freedom

Degree	Elements	DOF
P1	128^3	256^3
P2	86^3	258^3
P3	64^3	256^3
P4	52^3	260^3
P5	43^3	258^3
P6	37^3	259^3
P7	32^3	256^3
P8	29^3	261^3

2.3 Hardware

The simulation is performed across three NVIDIA Tesla K20c GPUs for P2 to P8 schemes and six GPUs for the P1 scheme due to memory limitations. Communication is performed using OpenMPI and with Infiniband interconnects. We use the CUDA backend available in PyFR on a local group cluster at Imperial College London.

No GPU implementation of TauBench was available at the time of writing. Therefore, we scale our results from the TauBench results of a similarly priced 12 core Intel Xeon E5-2697 v2 CPU. An individual core of this CPU had a TauBench time of 5.71 s, or 0.476 s adjusted for 12 cores.

3 Results

3.1 Performance

Simulation wall time, number of cards, card hours, and normalized computational time are shown in Table 3. The normalized time is computed from the core count adjusted 0.476 s Taubench on the Intel Xeon E5-2697 v2 CPU. In general, the higher-order schemes are more expensive due to a reduction in the time-step size required to maintain stability with the adaptive Runge-Kutta scheme and an increase in the total number of operations per degree of freedom. This could be mitigated in the future by implementing bespoke sparse matrix multiplication packages, such as the GiMMiK package developed for PyFR[11].

The P1 simulation is slightly slower than P2, which may be associated with additional overhead from node to node communication. However, no analysis was performed to confirm if this was the case.

Table 3: Performance Statistics

Degree	Wall Time (hrs)	Cards	Card Hours	Normalized
P1	6.0	6	36.1	2.72×10^5
P2	10.7	3	32.2	2.43×10^5
P3	11.9	3	35.7	2.70×10^5
P4	18.4	3	55.3	4.18×10^5
P5	25.2	3	75.6	5.71×10^5
P6	38.9	3	116.9	8.84×10^5
P7	43.4	3	130.2	9.84×10^5
P8	69.7	3	209.1	1.58×10^6

3.2 Contours

Iso-contours of the dimensionless vorticity norm $\frac{L}{V_0}|\omega| = 1, 5, 10, 20, 30$ on the periodic face $\frac{x}{L} = -\pi$ at $\frac{t}{t_c} = 8$ are shown in Figure 31 to Figure 7. These are generated using $\nabla \mathbf{u}$, where $\mathbf{u}$ is the discontinuous representation of the solution in each element.

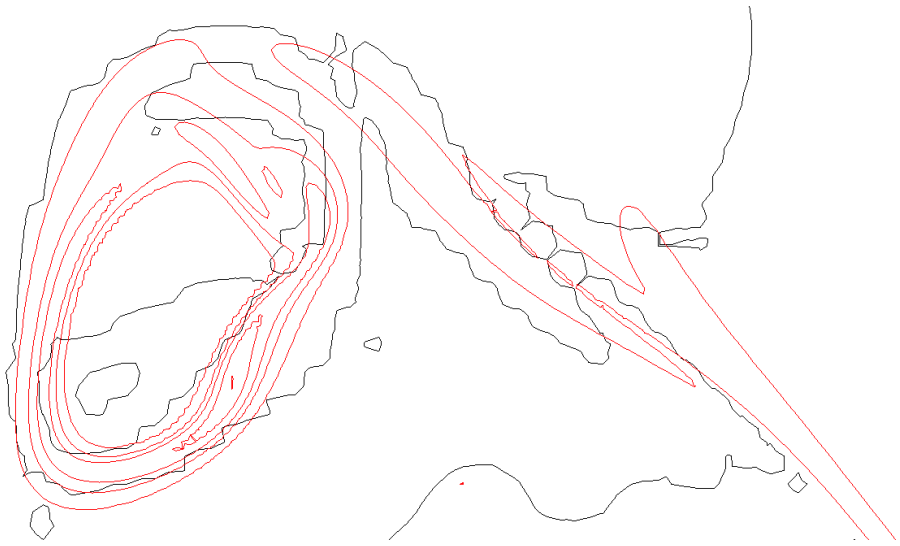

Figure 1: Iso-contours of the dimensionless vorticity norm for P1 (black) and reference spectral DNS (red) [9].

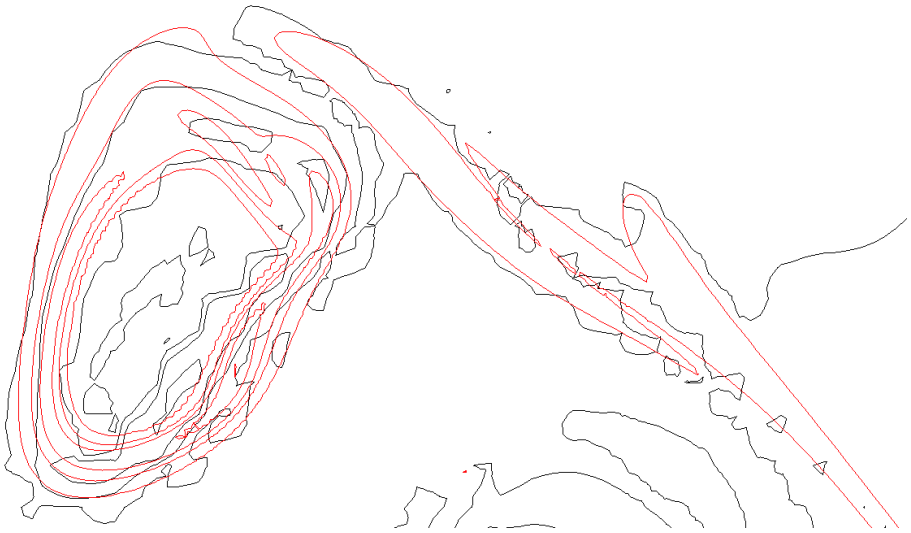

Figure 2: Iso-contours of the dimensionless vorticity norm for P2 (black) and reference spectral DNS (red) [9].

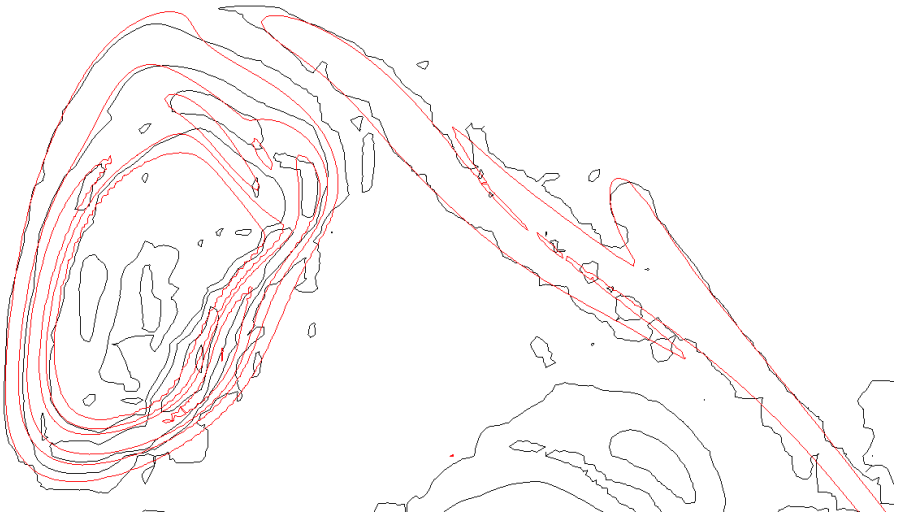

Figure 3: Iso-contours of the dimensionless vorticity norm for P3 (black) and reference spectral DNS (red) [9].

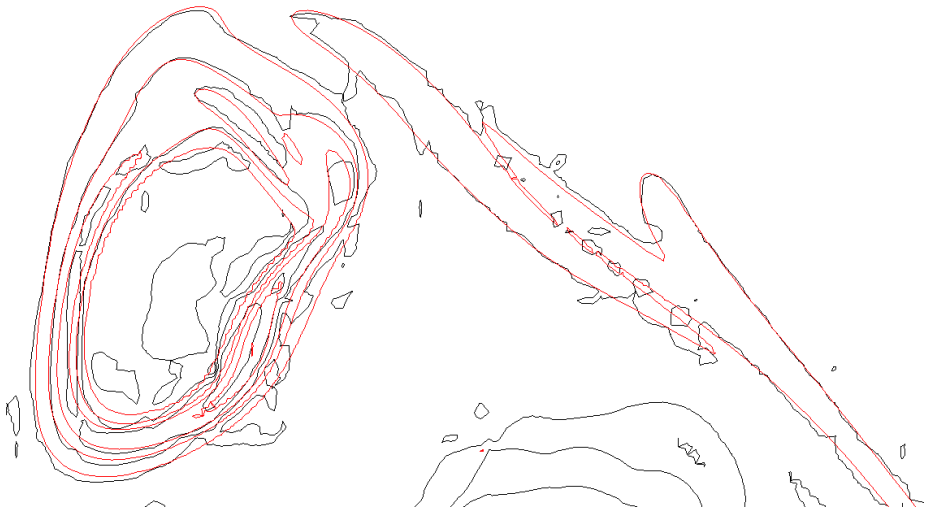

Figure 4: Iso-contours of the dimensionless vorticity norm for P4 (black) and reference spectral DNS (red) [9].

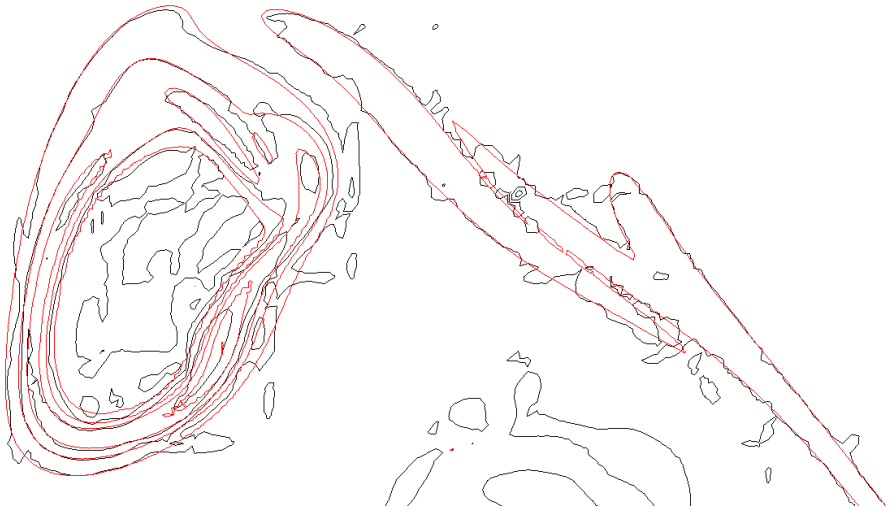

Figure 5: Iso-contours of the dimensionless vorticity norm for P5 (black) and reference spectral DNS (red) [9].

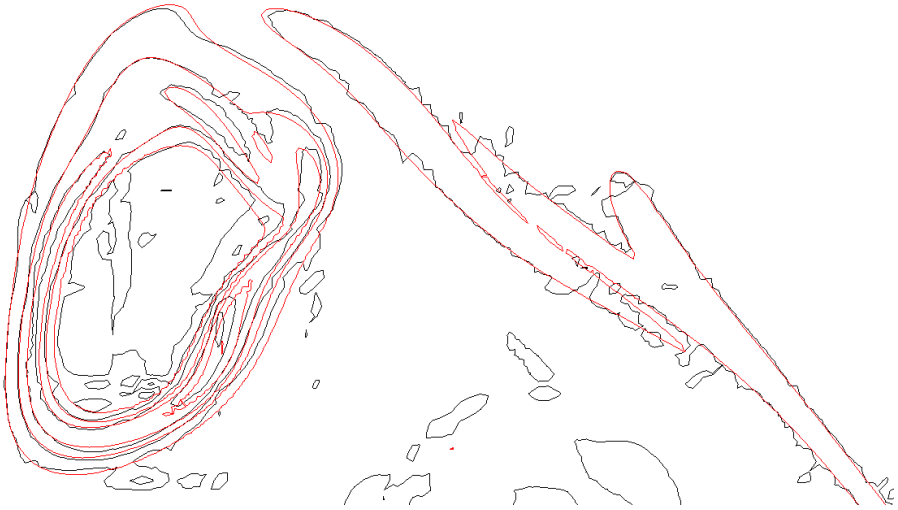

Figure 6: Iso-contours of the dimensionless vorticity norm for P6 (black) and reference spectral DNS (red) [9].

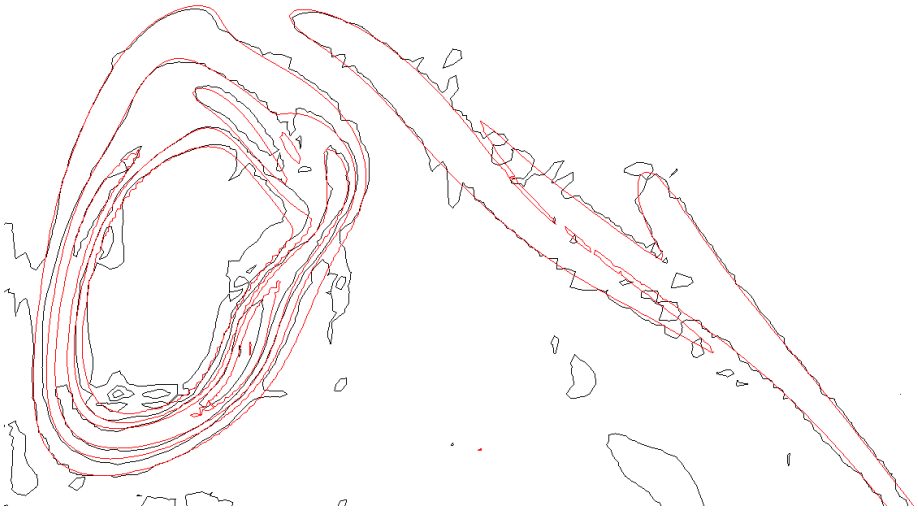

Figure 7: Iso-contours of the dimensionless vorticity norm for P7 (black) and reference spectral DNS (red) [9].

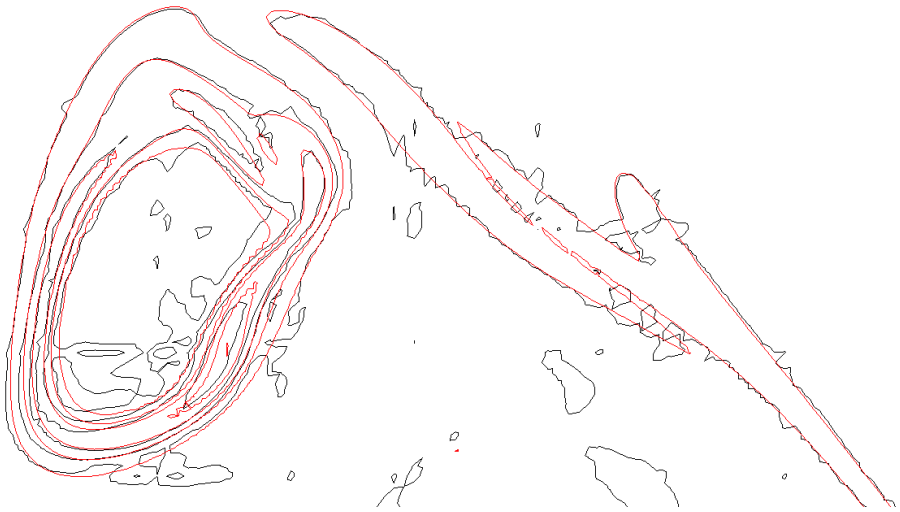

Figure 8: Iso-contours of the dimensionless vorticity norm for P8 (black) and reference spectral DNS (red) [9].

3.3 Energy

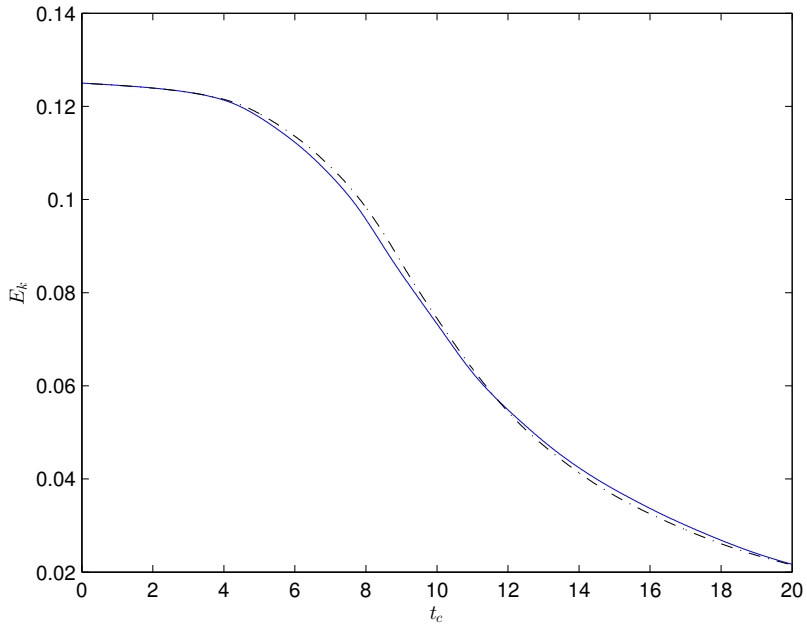

Figure 9: Temporal evolution of kinetic energy for P1 (blue) and reference spectral DNS (black) [9].

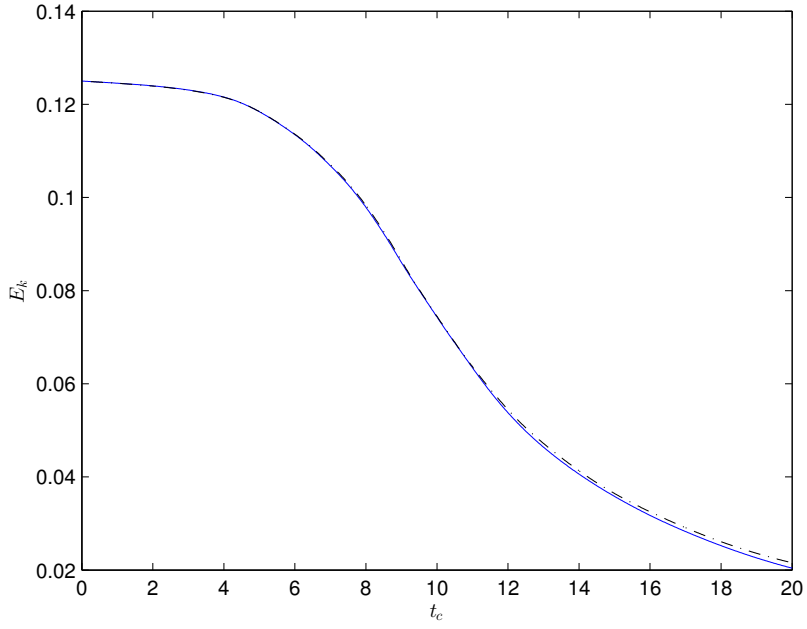

Figure 10: Temporal evolution of kinetic energy for P2 (blue) and reference spectral DNS (black) [9].

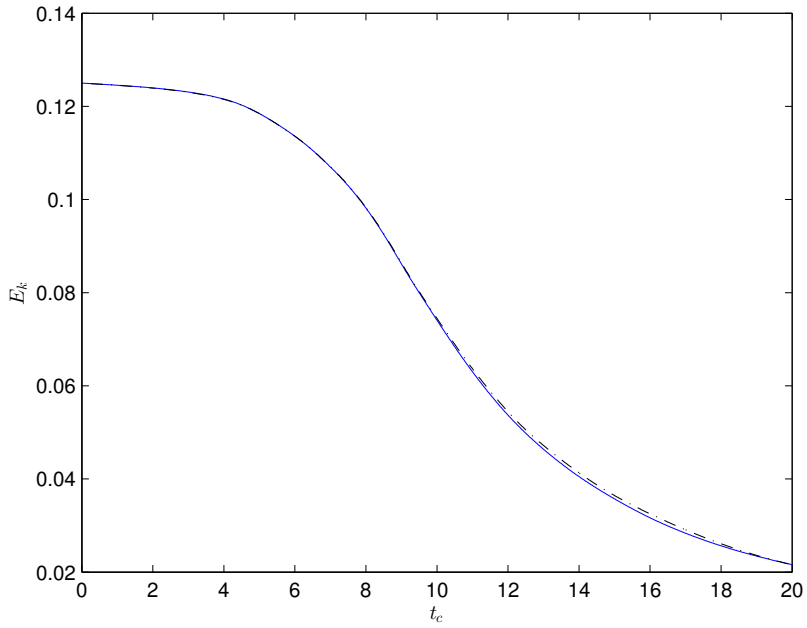

Figure 11: Temporal evolution of kinetic energy for P3 (blue) and reference spectral DNS (black) [9].

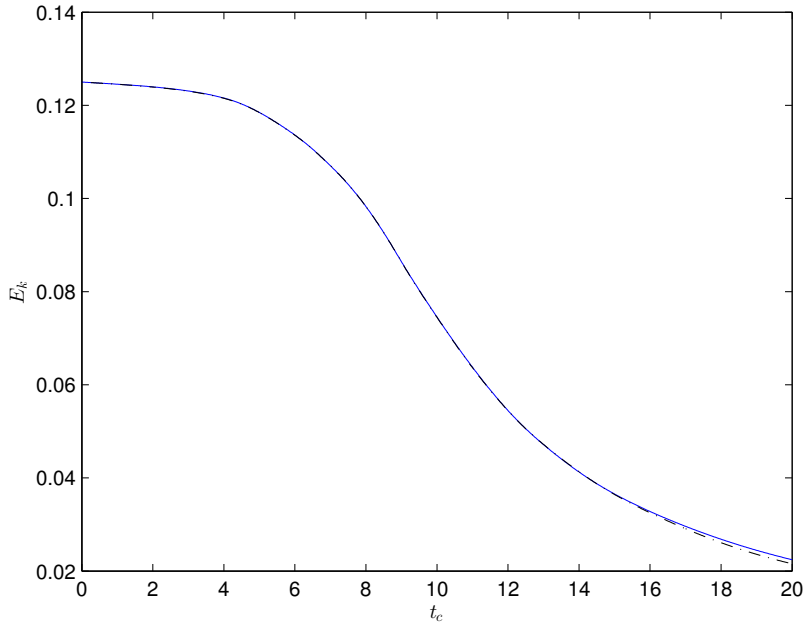

Figure 12: Temporal evolution of kinetic energy for P4 (blue) and reference spectral DNS (black) [9].

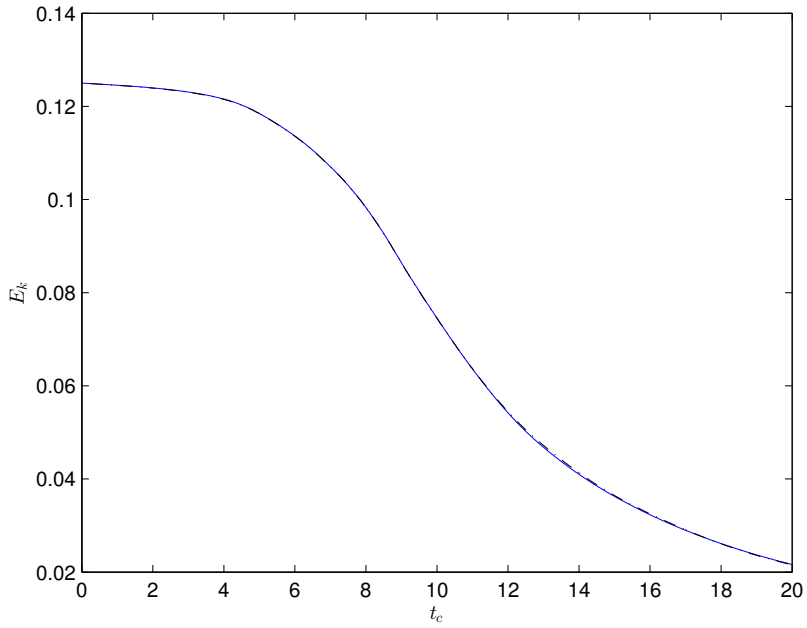

Figure 13: Temporal evolution of kinetic energy for P5 (blue) and reference spectral DNS (black) [9].

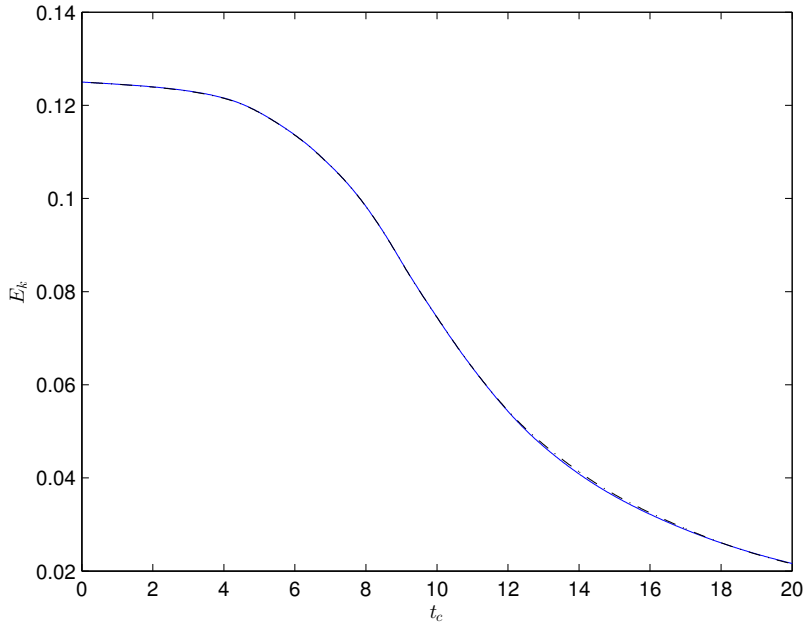

Figure 14: Temporal evolution of kinetic energy for P6 (blue) and reference spectral DNS (black) [9].

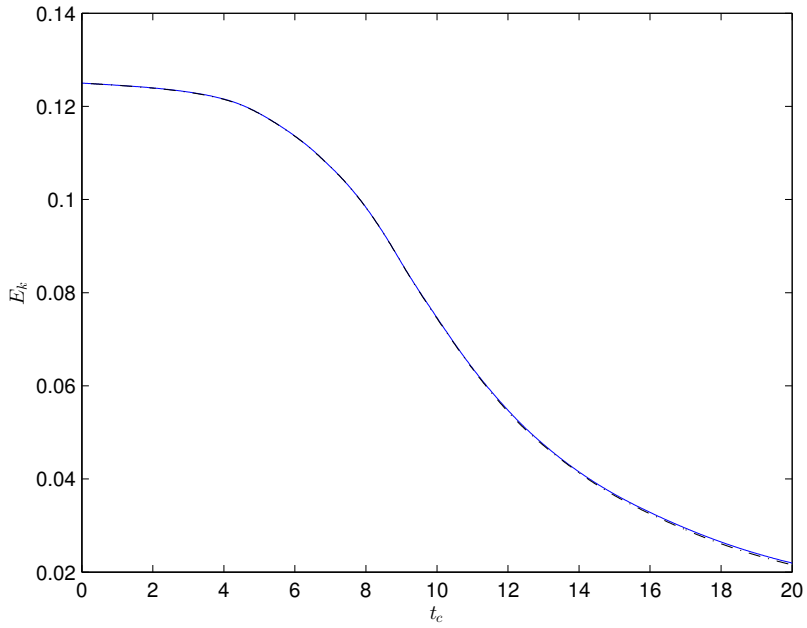

Figure 15: Temporal evolution of kinetic energy for P7 (blue) and reference spectral DNS (black) [9].

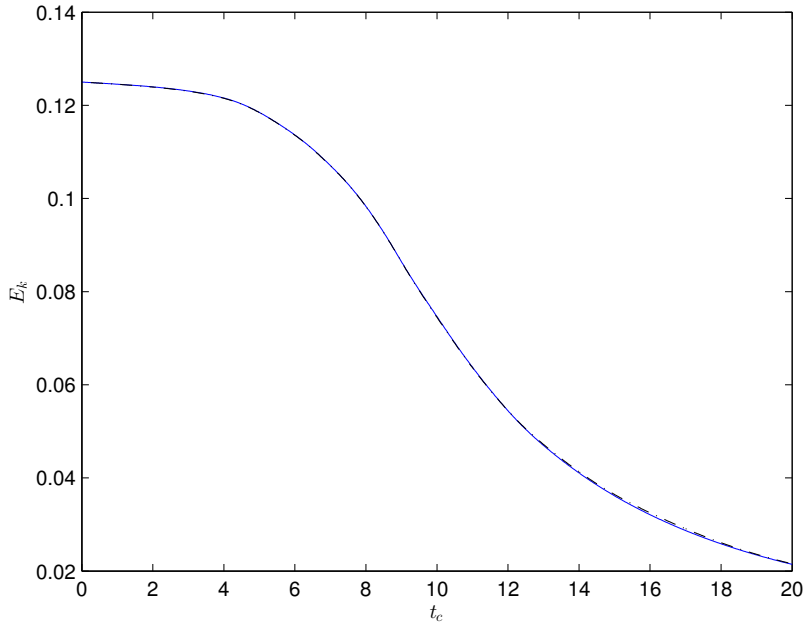

Figure 16: Temporal evolution of kinetic energy for P8 (blue) and reference spectral DNS (black) [9].

3.4 Dissipation

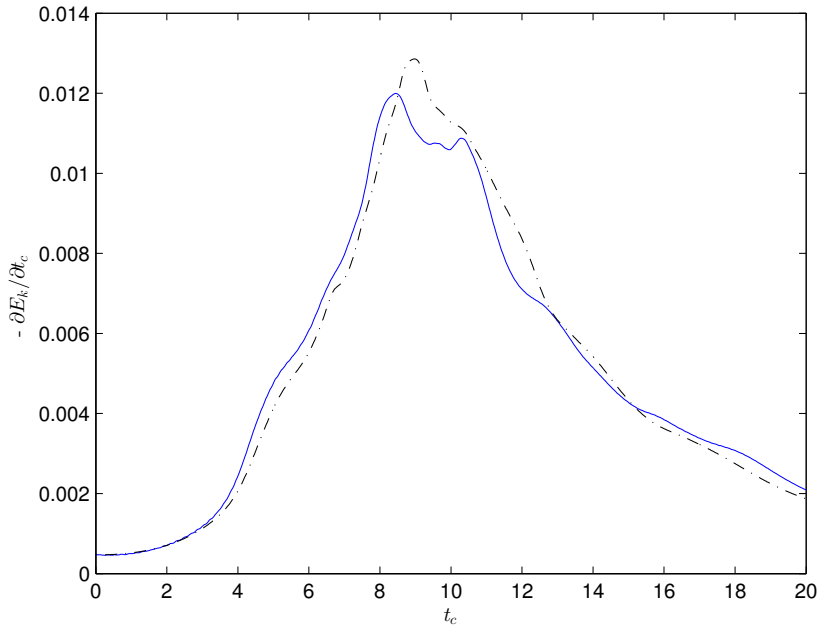

Figure 17: Temporal evolution of kinetic energy dissipation rate for P1 (blue) and reference spectral DNS (black) [9].

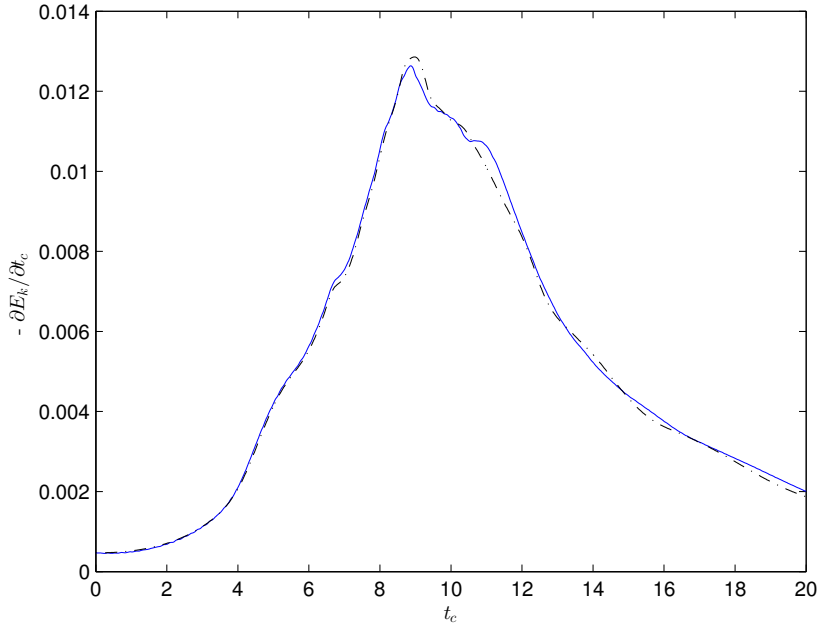

Figure 18: Temporal evolution of kinetic energy dissipation rate for P2 (blue) and reference spectral DNS (black) [9].

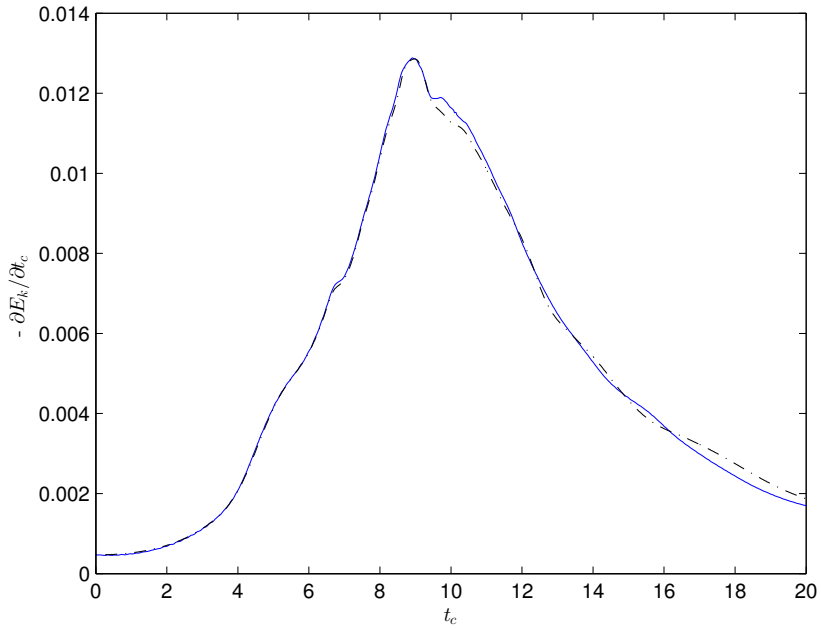

Figure 19: Temporal evolution of kinetic energy dissipation rate for P3 (blue) and reference spectral DNS (black) [9].

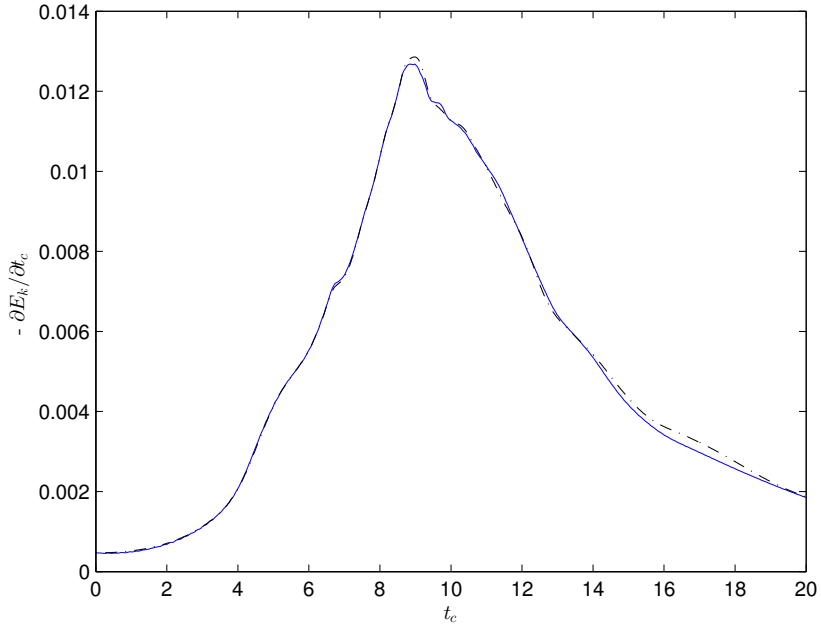

Figure 20: Temporal evolution of kinetic energy dissipation rate for P4 (blue) and reference spectral DNS (black) [9].

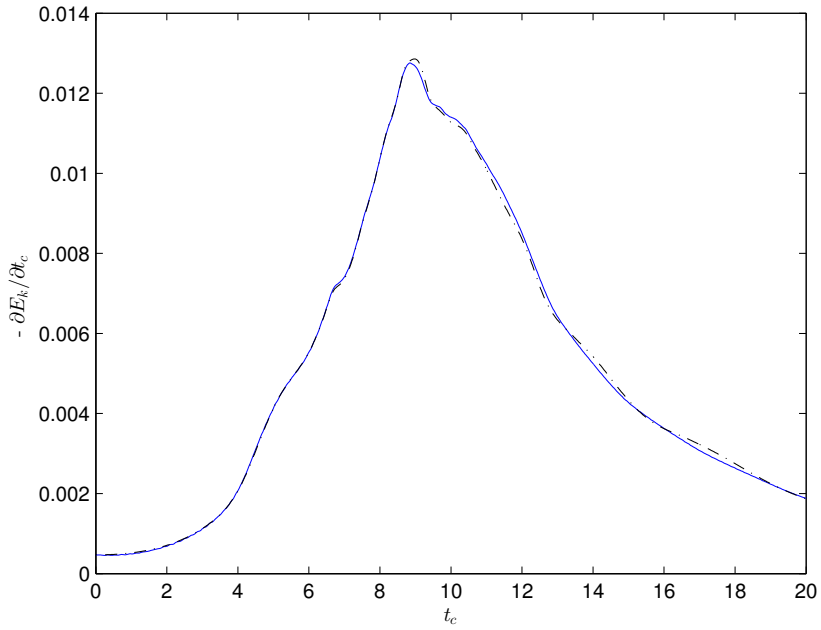

Figure 21: Temporal evolution of kinetic energy dissipation rate for P5 (blue) and reference spectral DNS (black) [9].

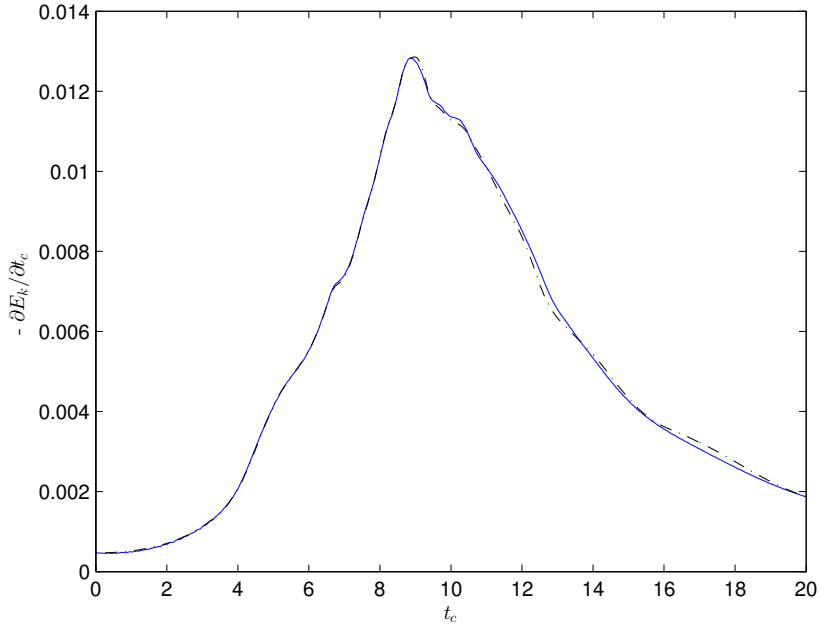

Figure 22: Temporal evolution of kinetic energy dissipation rate for P6 (blue) and reference spectral DNS (black) [9].

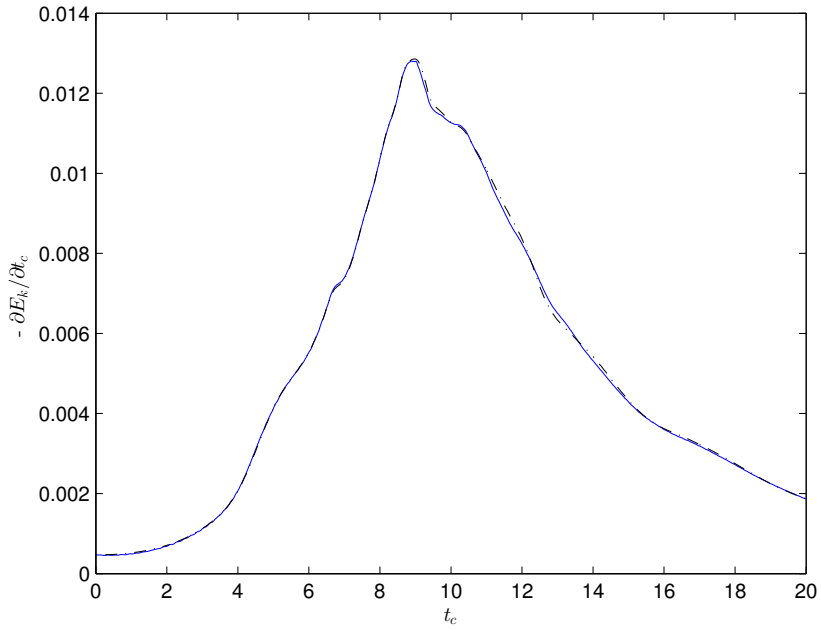

Figure 23: Temporal evolution of kinetic energy dissipation rate for P7 (blue) and reference spectral DNS (black) [9].

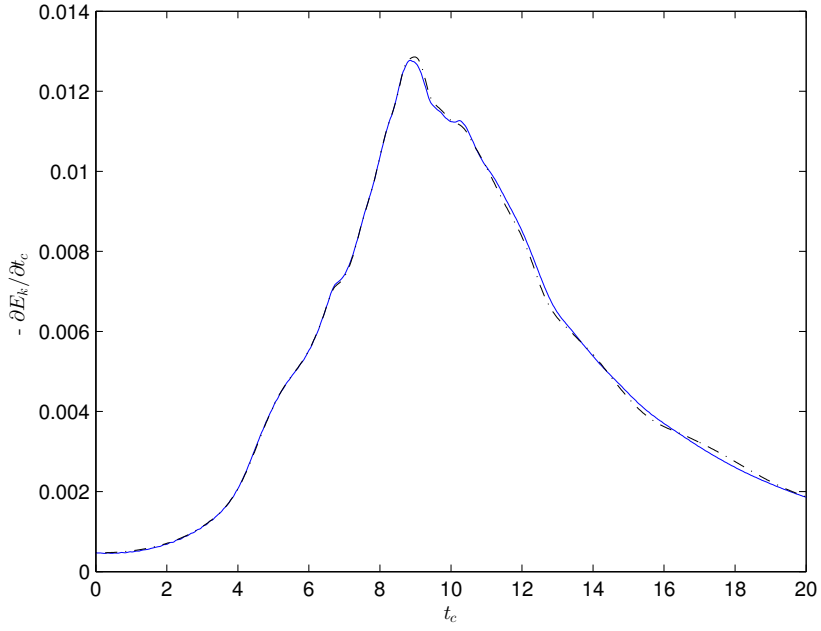

Figure 24: Temporal evolution of kinetic energy dissipation rate for P8 (blue) and reference spectral DNS (black) [9].

3.5 Enstrophy

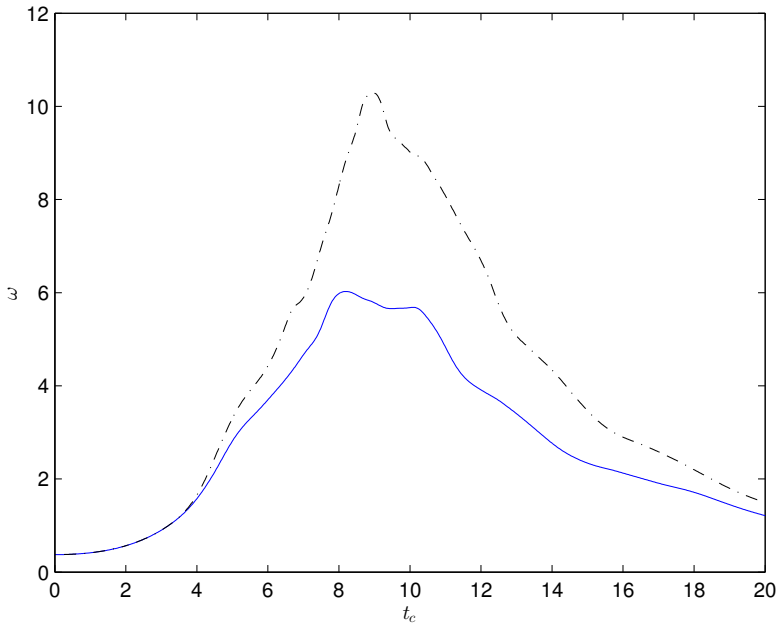

Figure 25: Temporal evolution of enstrophy for P1 (blue) and reference spectral DNS (black) [9].

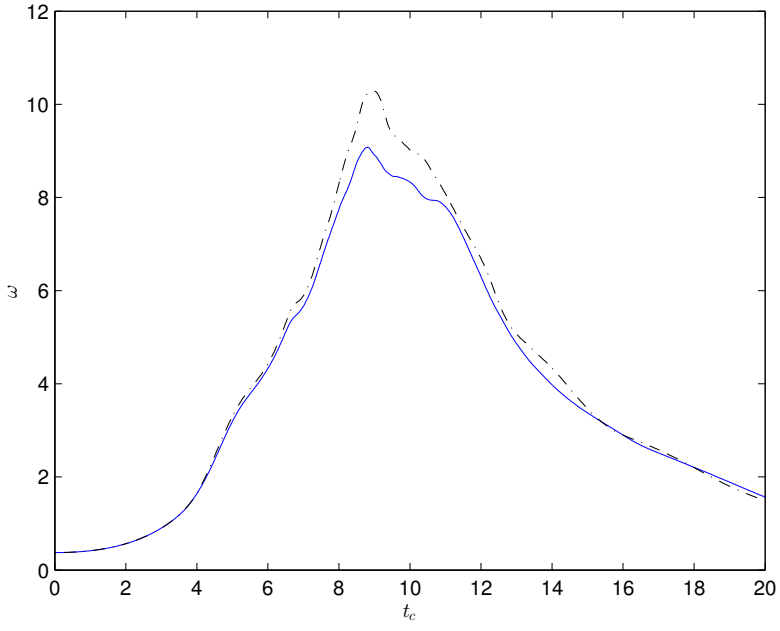

Figure 26: Temporal evolution of enstrophy for P2 (blue) and reference spectral DNS (black) [9].

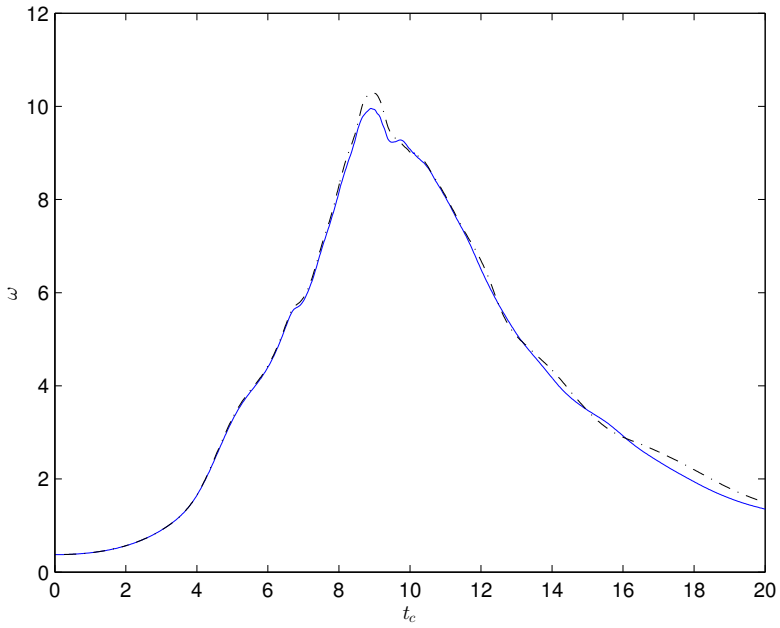

Figure 27: Temporal evolution of enstrophy for P3 (blue) and reference spectral DNS (black) [9].

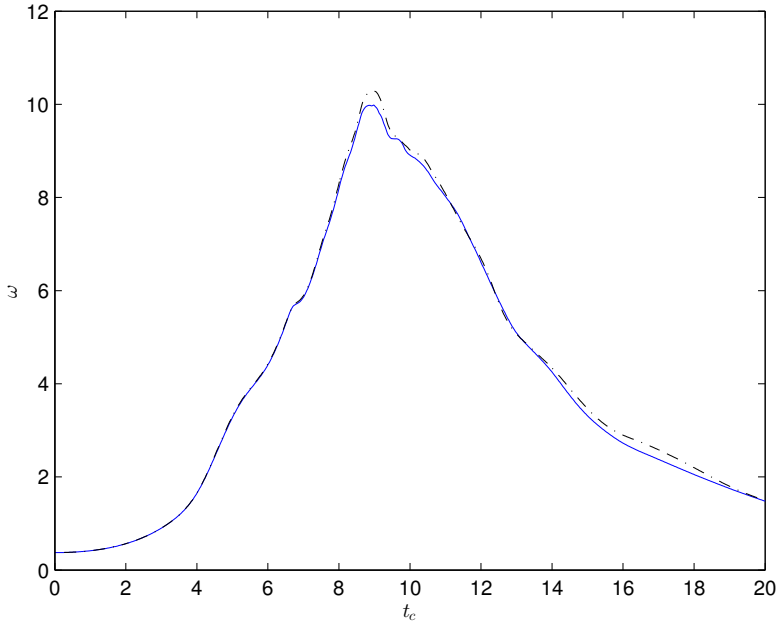

Figure 28: Temporal evolution of enstrophy for P4 (blue) and reference spectral DNS (black) [9].

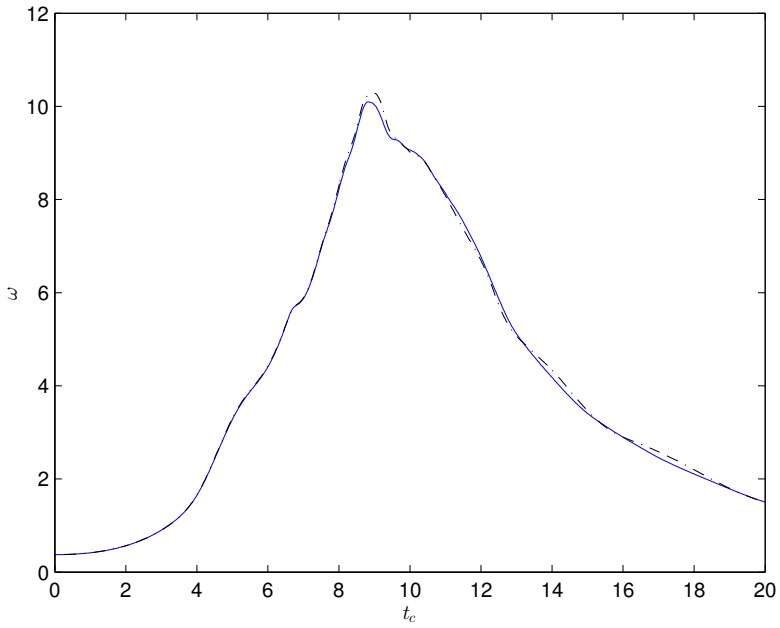

Figure 29: Temporal evolution of enstrophy for P5 (blue) and reference spectral DNS (black) [9].

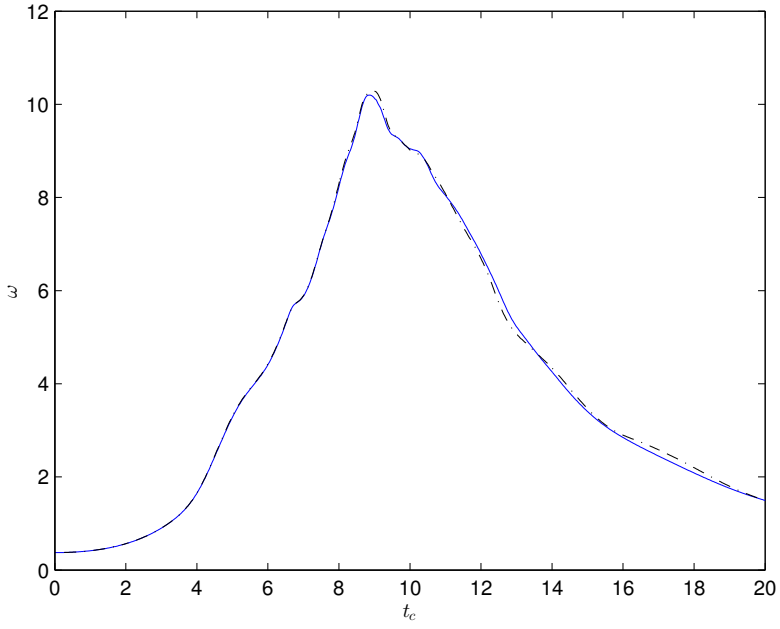

Figure 30: Temporal evolution of enstrophy for P6 (blue) and reference spectral DNS (black) [9].

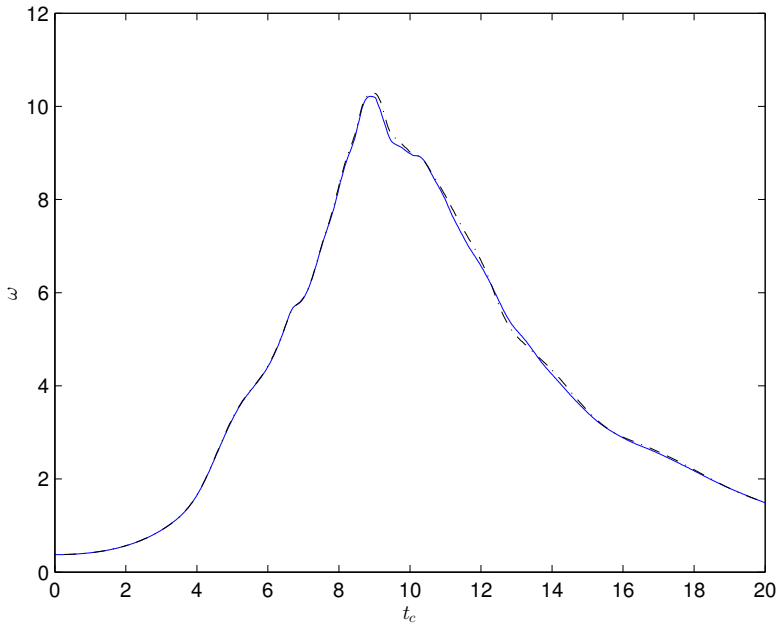

Figure 31: Temporal evolution of enstrophy for P7 (blue) and reference spectral DNS (black) [9].

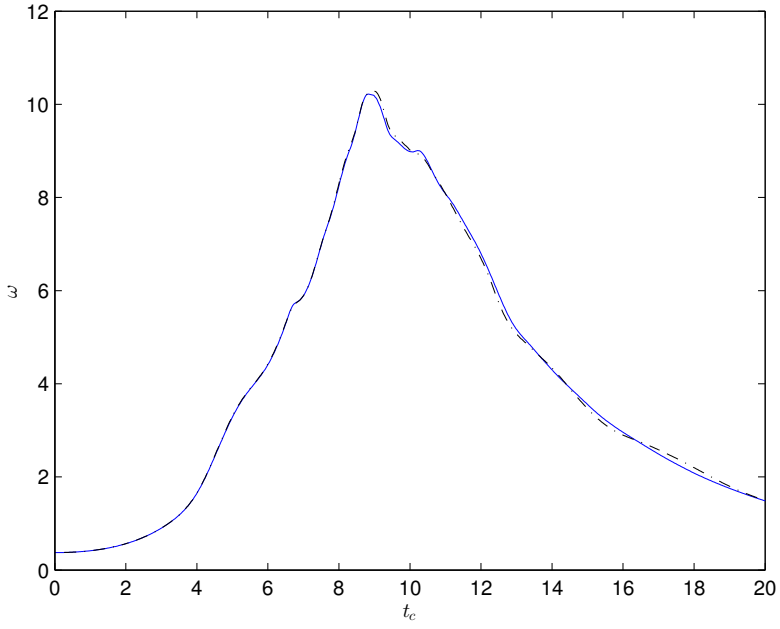

Figure 32: Temporal evolution of enstrophy for P8 (blue) and reference spectral DNS (black) [9].

References

[1] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2 edition, 2008.

[2] B. Cockburn and C. Shu. The local discontinuous galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, 1998.

[3] E Harier and H Wanner. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*. Springer, 2 edition, 1996.

[4] A. Harten, P. Lax, and B. Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM Review*, 25(1):35–61, 1983.

[5] H.T. Huynh. A flux reconstruction approach to high-order schemes including discontinuous galerkin methods. In *18th AIAA Computational Fluid Dynamics Conference*. AIAA 2007-4079, 2007.

[6] Christopher A. Kennedy, Mark H. Carpenter, and R.Michael Lewis. Low-storage, explicit rungekutta schemes for the compressible navierstokes equations. *Applied Numerical Mathematics*, 35(3):177 – 219, 2000.

[7] P.L Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357 – 372, 1981.

[8] E.F. Toro, M. Spruce, and W. Speares. Restoration of the contact surface in the hll-riemann solver. *Shock Waves*, 4(1):25–34, 1994.

[9] Wim M. van Rees, Anthony Leonard, D.I. Pullin, and Petros Koumoutsakos. A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high reynolds numbers. *Journal of Computational Physics*, 230(8):2794 – 2805, 2011.

[10] F.D. Witherden, A.M. Farrington, and P.E. Vincent. Pyfr: An open source framework for solving advectiondiffusion type problems on streaming architectures using the flux reconstruction approach. *Computer Physics Communications*, 185(11):3028 – 3040, 2014.

[11] B.D. Wozniak, F.D. Witherden, F.P. Russell, P.E. Vincent, and P.H.J. Kelly. Gimmik - generating bespoke matrix multiplication kernels for accelerators: Application to high-order computational fluid dynamics. *In Review*, 2014.