Supplementary Material

Video captions

This appendix contains captions for the online movies which accompany this article. All movies correspond to certain figures contained in the previous sections and are included to elucidate the behaviours under discussion. The files are found online as supplementary material and also hosted at `http://www.maths.bris.ac.uk/~maxdl/kolmo/movies/`.

0.1. *Movie 1*

This movie shows the vorticity field, ω, for a DNS of the chaotic kink-antikink pair at $Re = 70$ as discussed in section 4. Total integration is $T = 500$ with timestep $dt = 0.005$ and individual frames are separated by 5 time units. The dynamics are mostly limited to the central 'eyes' of the kink and antikink which oscillate aperiodically. This is in stark contrast to the non-localised chaos observed for $\alpha = 1$.

0.2. *Movie 2*

This movie shows the vorticity field, ω, for the solution P1 at $Re = 20$ and corresponds directly with figure 16. Total integration is $T = 20.7$ with timestep $dt = 0.05$ and individual frames are separated by 0.2 time units. This movie clearly shows the standing wave-like motion of the central vorticity distribution, while the outer kink (right) and antikink (left) remain steady.

0.3. *Movie 3*

This movie shows the vorticity field, ω, for the chaotic repeller version of P1 at $Re = 24.1$ as discussed in section 4.5 and corresponding directly with figures 21 and 22. Total integration is $T = 10^5$ with timestep $dt = 0.05$ and individual frames are separated by 200 time units. This movie indicates the chaotic motions of the central region, eventually the motions result in a catastrophic collision with the antikink.

0.4. *Movie 4*

This movie shows the vorticity field, ω, for the solution P2 at $Re = 20$ and corresponds directly with figure 22. Total integration is $T = 21.4$ with timestep $dt = 0.05$ and individual frames are separated by 0.2 time units. Here we see the largely the same behaviour as for P1, only now with two periodic regions.

0.5. *Movie 5*

This movie shows the vorticity field, ω, for the chaotic saddle at $Re = 20.75$ corresponding with figure 23. Total integration is $T = 10^5$ with timestep $dt = 0.05$ and individual frames are separated by 1000 time units. Striking in this movie is the uniform translation of the flow while two vortical patches oscillate chaotically between two kinks. Eventually these collide and in doing so one is annihilated leaving the solution P1.

0.6. *Movie 6*

This movie shows the vorticity field, ω, for $\alpha = \frac{1}{8}$ at $Re = 22$ as discussed in section 5 and corresponding directly with figure 24. Total integration is $T = 10^5$ with timestep $dt = 0.05$ and individual frames are separated by 500 time units. Given a randomised initial condition this movie demonstrates the emergence of stable propagating kink-antikink bound states. The flow rapidly localises from the initial conditions to an assortment of kinks and antikinks and an isolated, translating P1-like structure in the left had portion of the domain.

0.7. *Movie 7*

This movie shows the vorticity field, ω, for $\alpha = \frac{1}{8}$ at $Re = 19$ as discussed in section 5 and corresponding directly with figure 26. Total integration is $T = 10^5$ with timestep $dt = 0.05$ and individual frames are separated by 500 time units. Given an initial condition comprised of kink-antikink travelling waves, this movie demonstrates several of the collisional behaviours we encounter. Beginning with an elastic swapping collision, then two rebounding collisions and finally a merging collision which results in 2 kinks and 1 antikink forming a localised chaotic oscillatory structure.

Utilising GPUs

It was found to be a remarkably straightforward process to port the algorithm into CUDA. In the pseudospectral method, the only nonlocal computation is performed via the Fourier Transform and a GPU accelerated FFT library is freely available with CUDA called CUFFT. As the rest of the operations (timestepping, convolutions etc.) are pointwise arithmetic the GPU implementation simply amounts to unwrapping nested loops into the multithreaded architecture of the GPU.

To assess the added benefit of using GPUs compared to the Fortran CPU code of Chandler & Kerswell (2013), both codes were run at $Re = 40$ for $20,000$ timesteps ($T = 100$, $\Delta t = 0.005$) using the same randomised initial data, with $\alpha = 1$ ($N_x = N_y = N$). This is long enough to marginalise the overhead in copying to and from the GPU (less than 0.1% CPU time here). For the problem sizes in this study (equivalent to $N = 256$ since typically $\alpha = \frac{1}{4}$ and $N_x = 128$, $N_y = 512$), there is a moderate $\times 4$ speed up: see figure 1 while brief tests with larger problem sizes ($[4096]^2$) suggest a speed up of $\times 30$ is achievable. For our application it was found crucial to employ double precision arithmetic to allow sufficient accuracy to converge solutions with Newton-GMRES, however if such accuracy is not critical then an additional $\times 2$ speed up can be achieved using single precision (making the $[4096]^2$ case 70 times quicker than the CPU equivalent).

In general, the code spends around 80% of GPU time in the CUFFT components of the algorithm limiting the possibility for further optimisation. The code uses a single GPU card for the calculations; communication across multiple GPUs is likely to be a critical overhead for such a small problem size (see Lou & Yin (2012)). In this study, the state vector is copied across to the GPU and timestepped until a return to the CPU is required (either for Newton-GMRES or recurrence checking) thus minimising communication to and from the GPU. The majority of computations presented here are carried out on the EMERALD GPU cluster at Rutherford Appleton Laboratory which has Intel Xeon X5650 host compute nodes and 512-core NVIDIA Tesla M2090 GPUs. A small number of calculations are carried out on a local machine with an Intel Xeon X5670 host CPU and 448-core NVIDIA Tesla S2050 GPUs.

REFERENCES

Chandler, Gary J & Kerswell, Rich R 2013 Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow. *Journal of Fluid Mechanics* **722**, 554–595.

Lou, Kaiyuan & Yin, Zhaohua 2012 A CUDA Pseudo-spectral Solver for Two-dimensional Navier-Stokes Equation. *Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2012 11th International Symposium* pp. 62–66.

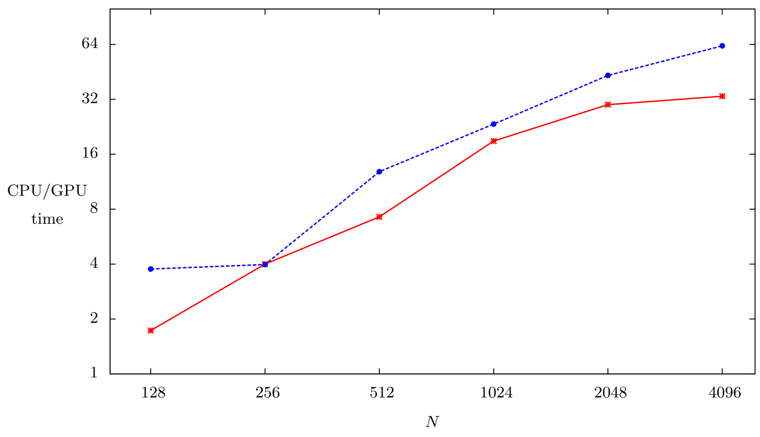

FIGURE 1. CPU/GPU computation speed up for the timestepping code with resolution $N \times N$ (e.g. a 256×256 run is ≈ 4 times faster in the GPU code than the CPU code). Blue dahsed curve represents single precision acceleration, red double precision.