nekRS and Nek5000 comparative report

Introduction:

The objective of this test is to compare the runtime performance of Nek5000 (CPU-based) and NekRS (GPU-based) solvers when running the same flow case at increasing simulation durations: 0.3, 3, and 30.

To compare runtimes, simulations were run for a canonical turbulent pipe flow where Re_tau=180 (viscosity = -2650 in turbPipe.par files) and of 8 GLL points. Each solver was run on the same problem setup for consistency.

As for the hardware, Nek5000 was executed on the University of Manchester's CSF3 cluster, using AMD EPYC 9454 CPUs and parallelized with MPI. nekRS, on the other hand, was run on the MaSC cluster, using a single NVIDIA GPU via CUDA.

The mesh and further details on the simulations can be found in the scripts of the following repository:
https://github.com/etiennejacquot/nekRSvs5000

This report presents the runtime results in a series of tables, followed by key observations, a discussion of limitations, and recommendations for future work. The raw runtime summaries for each simulation are provided in the appendix.

Results and Observations:

For each simulation, the data below was extracted. Here, "runtime" refers to the total wall-clock time (in seconds):

endtime	Nek5000 runtime	nekRS runtime	Speedup factor ($\frac{Nek5000\ runtime}{nekRS\ runtime}$)
0.3	803	42	19.3
3	4041	108	37.9
30	23762	738	32.2

Table 1: runtime comparisons for each run

These results clearly demonstrate the performance advantage of NekRS. The GPU-based solver is approximately 20 to 40 times faster than its CPU counterpart. For instance, a Nek5000 run that takes over 13 minutes can be completed in under 1 minute with NekRS. Likewise, a nearly 7-hour Nek5000 simulation is reduced to just over 12 minutes with NekRS.

This dramatic improvement in runtime is consistent with expectations for GPU-accelerated codes and highlights the efficiency of NekRS in compute-intensive flow problems.

In the table below, the time per step (in seconds) is extracted for each simulations:

endtime	Nek5000 time/step	nekRS time/step
0.3	0.642	0.782
3	0.590	0.212
30	0.390	0.148

Table 2: time per step comparison for each run

Before comparing the results obtained, it should be noted that Nek5000 uses variable timestepping where nekRS uses a fixed timestep. This distinction limits direct comparison and should be considered when drawing conclusions.

It can be seen that for nekRS, the time/step decreases considerably as the endtime increases, from 0.782 to 0.148, for endtime = 0.3 and 3 respectively. This decrease show an amortization of the startup overhead, such as memory initialization, over a larger number of steps.

In Nek5000, the adaptive timestep increases over time, starting small during the initial transient and growing as the flow stabilizes. This results in a more gradual and modest improvement in time per step compared to nekRS.

Limits and Recommended Work:

First, the difference in timestepping strategies, adaptive in Nek5000 and fixed in nekRS, means the simulations were not identical in time integration. While both used their default solvers (BDF3 for Nek5000 and TOMBO2 for nekRS), these differ in order (3rd vs. 2nd) and stability properties. This contributes to differences in both performance and accuracy.

Another consideration is that Nek5000 was run on 42 CPU cores, which is about 187 local elements per MPI rank. This is more than the previously tested 600 elements per MPI rank. The results are then run on a much faster setup with a lot of resources, and would me much slower if it was using less nodes. Performance scaling with fewer CPUs would be an insightful direction for further testing.

For future work, a more detailed study could first explore varying polynomial order. In fact, this study used 8 GLL points per element (polynomial order 7). Increasing this to 10 or 12 would highlight scalability differences.

Another valuable extension of this work would be to explore longer simulation times. While an end time of 30 already provides meaningful insights, pushing this further to about 200 could reveal additional performance trends and stability characteristics. However, due to limited time resources and queueing constraints on the CSF cluster, it was not feasible to run Nek5000 simulations beyond this point making this a promising direction for future work.

Conclusion:

The key takeaway from this study is the significant performance difference between Nek5000 and nekRS. With nekRS achieving 20 to 40 speedup factors, it proves to be a highly efficient alternative for time-dependent spectral element simulations, especially when GPU resources are available. Importantly, this performance boost does not appear to come at the cost of numerical stability.

However, several important limitations were identified. The two solvers use different timestepping strategies, adaptive in Nek5000 and fixed in nekRS, which limits the precision of direct comparisons. Additionally, the underlying time integration schemes (BDF3 vs. TOMBO2) differ in order and stability characteristics. These differences must be considered when interpreting the results.

Despite these limitations, the performance gap observed suggests that nekRS is a strong candidate for large-scale simulations and time-sensitive workflows, particularly when GPU resources are available. Future work could include extending simulations to longer end times, exploring increased polynomial orders through more GLL points and testing Nek5000 across varying core counts

In the appendix of this report, a list of the un-edited runtime summaries, from which further insights can be extracted.

Raw Output Data

Nek5000

endtime = 0.3

```
total elapsed time          :   8.02805E+02 sec
total solver time w/o IO     :   7.98464E+02 sec
time/timestep              :  6.42368E-01 sec
avg throughput per timestep   :   1.00699E+05 gridpts/CPUs
total max memory usage       :   1.37458E+01 GB
```

endtime = 3

```
total elapsed time          :   4.04060E+03 sec
total solver time w/o IO     :   4.03487E+03 sec
time/timestep              :  5.89893E-01 sec
avg throughput per timestep   :   1.09657E+05 gridpts/CPUs
total max memory usage       :   1.37750E+01 GB
```

endtime = 30

```
total elapsed time          :   2.37620E+04 sec
total solver time w/o IO     :   2.37571E+04 sec
time/timestep              :  3.90485E-01 sec
avg throughput per timestep   :   1.65655E+05 gridpts/CPUs
total max memory usage       :   1.41648E+01 GB
```

nekRS

endtime = 0.3

```
>>> runtime statistics (step= 53  totalElapsed= 41.4623s):
name                  time         abs%  rel%  calls
  solve              1.05609e+01s  100.0
    min             1.62905e-01s
    max              4.64347e-01s
    flops/rank        6.81238e+11
    checkpointing       1.15599e-01s   1.1        1
    udfExecuteStep      7.08160e-05s   0.0        53
    makef              4.01548e+00s  38.0        53
    udfProperties       9.41120e-05s   0.0        54
    velocitySolve       3.65029e+00s  34.6        55
      rhs              1.71016e-01s   1.6   4.7  55
      preconditioner    3.76005e-01s   3.6  10.3  831
    pressureSolve       2.18152e+00s  20.7        55
      rhs              2.86873e-01s   2.7  13.2  55
      preconditioner    1.33979e+00s  12.7  61.4  261
        pMG smoother    9.24921e-01s   8.8  69.0  522
        pMG smoother    1.23791e-01s   1.2   9.2  522
        coarse grid     1.31330e-01s   1.2   9.8  261
      initial guess     1.25538e-01s   1.2   5.8  55
```

endtime = 3

```
>>> runtime statistics (step= 503  totalElapsed= 106.706s):
name                  time         abs%  rel%  calls
  solve                7.62994e+01s  100.0
    min                1.38197e-01s
    max                4.61493e-01s
    flops/rank          7.21124e+11
    checkpointing       1.31174e-01s   0.2        1
    udfExecuteStep      6.95328e-04s   0.0        503
    makef              3.84249e+01s  50.4        503
    udfProperties       7.39328e-04s   0.0        504
    velocitySolve       2.37637e+01s  31.1        505
      rhs              1.59629e+00s   2.1   6.7  505
      preconditioner    2.29109e+00s   3.0   9.6  5055
    pressureSolve       9.89341e+00s  13.0        505
      rhs              2.70540e+00s   3.5  27.3  505
      preconditioner    4.34094e+00s   5.7  43.9  842
        pMG smoother    2.98748e+00s   3.9  68.8  1684
        pMG smoother    3.98613e-01s   0.5   9.2  1684
        coarse grid     4.33606e-01s   0.6  10.0  842
      initial guess     1.25737e+00s   1.6  12.7  505
```

endtime = 30

```
>>> runtime statistics (step= 5003  totalElapsed= 737.619s):
name                    time          abs%  rel%  calls
  solve                 7.07009e+02s  100.0
    min                 1.34464e-01s
    max                 4.59852e-01s
    flops/rank          7.32855e+11
    checkpointing       2.03415e-01s   0.0        2
    udfExecuteStep      6.34090e-03s   0.0        5003
    makef               3.82741e+02s  54.1        5003
    udfProperties       6.99341e-03s   0.0        5004
    velocitySolve       2.06151e+02s  29.2        5005
      rhs               1.58530e+01s   2.2   7.7  5005
      preconditioner    1.91849e+01s   2.7   9.3  42303
    pressureSolve       7.88866e+01s  11.2        5005
      rhs               2.68908e+01s   3.8  34.1  5005
      preconditioner    2.76513e+01s   3.9  35.1  5347
        pMG smoother    1.89899e+01s   2.7  68.7  10694
        pMG smoother    2.53881e+00s   0.4   9.2  10694
        coarse grid     2.78048e+00s   0.4  10.1  5347
      initial guess     1.26034e+01s   1.8  16.0  5005
```