ANL/ALCF/ESP-13/3

Direct Numerical Simulation of Autoignition in a Jet in a Cross-Flow Configuration

ALCF-2 Early Science Program Technical Report

Argonne Leadership Computing Facility

ANL/ALCF/ESP-13/3

Direct Numerical Simulation of Autoignition in a Jet in a Cross-Flow Configuration

ALCF-2 Early Science Program Technical Report

prepared by
Ammar Abdilghanie[1], Christos E. Frouzakis[2], Paul F. Fischer[3]
[1]Argonne Leadership Computing Facility, Argonne National Laboratory
[2]Federal Institute of Technology Zürich (ETH Zürich)
[3]Math and Computer Science, Argonne National Laboratory

May 7, 2013

Direct Numerical Simulation of Autoignition in a Jet in a Cross-Flow Configuration

Ammar Abdilghanie[1][1], Christos E. Frouzakis[2][2] and Paul F. Fischer[3][3]

[1]Leadership Computing Facility, Argonne National Laboratory, Argonne, IL 60439
[2]Aerothermochemistry and Combustion Systems Laboratory, Swiss Federal Institute of Technology Zurich (ETHZ), Zurich, Switzerland
[3]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

April 3, 2013

[1]aabdilghanie@alcf.anl.gov
[2]frouzakis@lav.mavt.ethz.ch
[3]fischer@mcs.anl.gov

Abstract

Autoignition in turbulent flows is a challenging fundamental problem due to the intricate coupling of different physical and chemical processes extending over multiple flow and chemistry scales. At the same time, the improved understanding and ability to predict autoignition in flows characterized by considerable fluctuations of velocity, composition, and temperature is essential for the development of novel low-emission concepts for power generation. The aim of this project is to study the fundamental aspects of autoignition in a fuel-air mixing device directly applicable to mixing ducts in gas turbines. The NEK5000-based code for low Mach number reactive flows is used to perform very large scale direct numerical simulations of autoignition of a diluted hydrogen jet ejected in a cross-flowing stream of hot turbulent air in a laboratory-scale configuration. We report on our experience running NEK5000 on the new BGQ system at ALCF **mira** during the early science period (ESP). First of all, the most efficient problem size per MPI-rank is obtained through core-level efficiency metric measured from the target simulation. Furthermore, the most efficient number of ranks is found through strong scaling experiments. Finally, low-level insight into the observed parallel efficiency is enabled through IBM's HPC Toolkit libraries.

Contents

Chapter 1

Introduction

Understanding the conditions under which autoignition of reacting mixtures occurs is of primary importance for the design and operation of modern lean premixed (LP) and lean premixed prevaporized (LPP) combustion devices such as low-NO_x stationary gas turbines (Dobbeling *et al.*, 2007) and propulsion devices such as subsonic ramjets and supersonic scramjets (Micka and Driscoll, 2012). In particular, the enhanced turbulent mixing between the fuel and oxidizer streams of the jet in cross-flow (JICF) configuration makes it an essential component in the design of premixing sections of modern high efficiency, low NO_x combustors.

The main objective of this study is to investigate the sensitivity of the JICF dynamics to the cross-flow temperature. In particular we ask the following questions: When and where do localized flame kernels form? Could the observed ignition time be correlated to ignition-delay time in the corresponding homogeneous system? Do these kernels develop into stable flame later on? How does the mixture get prepared for ignition and what is the role of the vorticity and pressure in mixing the jet (fuel) with the cross-flow (oxidizer) fluid? What is the local combustion mode associated with the observed ignition scenarios?

1.1 Overview of the Numerical Method

A weak formulation of the compressible reactive Navier-Stokes for ideal gas mixtures at the low Mach number limit were solved, using a parallel spectral-element code based on NEK5000 (Fischer *et al.*, 2008). The spectral element method is a high-order weighted residual technique that couples the rapid convergence of global spectral methods with the geometric flexibility of finite element methods (Deville *et al.*, 2002).

A high-order operator splitting technique is used to split the thermochemistry (species and energy equations) from the hydrodynamic subsystem (continuity and momentum equations) (Tomboulides *et al.*, 1997). The latter is integrated in time using a 3rd-order semi-implicit method, with explicit treatment of the nonlinear terms and implicit treatment of the viscous and pressure terms, while the thermo-

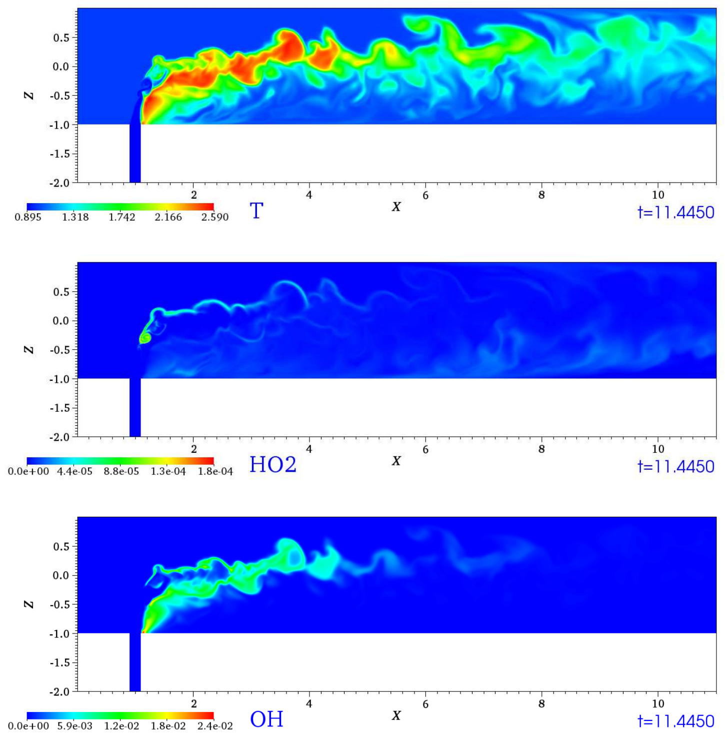

Figure 1.1: Snapshots of temperature and mass fractions of HO_2, OH for the T_{cf} = 950 K case on a cross section through the symmetry plane.

chemistry subsystem is solved fully implicitly by CVODE, a scalable BDF-based stiff ODE solver (Cohen and Hindmarsh, 1996). Detailed chemistry, thermodynamic properties and mixture-averaged transport properties are evaluated using Chemkin (Kee *et al.*, 1986).

Tensor product factorization is the core computational kernel in NEK5000. It is used to evaluate spectral element operators, interpolation, integration and differentiation. It is cast in terms of a sequence of dense and non-square matrix products (mxm). Optimizing the mxm kernel was performed on the BG/P using loop tiling and architecture-specific assembly code (Kerkemeier, 2010). The code developments resulted in improved core throughput and cache utilization (through data reuse) compared to highly optimized BLAS libraries. Finally, Kerkemeier (2010) developed a chemistry calculation kernel for the computation of the production rates, transport and thermodynamic property evaluation. All expensive computations were vectorized using the highly optimized math libraries MASS/MASSV which allowed best floating point, load and store performance.

1.1.1 Summary of Numerical Simulations

Exploratory numerical simulations on experimental computational grids as well as production runs were performed during the early science period on Mira. The computational grid is composed of $N_E = 1,591,752$ spectral elements within which the solution is interpolated using three-dimensional tensorial product of $p = 6$-th order Lagrange polynomials.

Post processing of the solution was performed using NEK5000 on the two datasets including computations of key species generation rates, mixture fraction, scalar dissipation rate, Takeno flame index and vorticity components. Analysis of the two large data sets was also mainly facilitated through visualizations with VisIt on the ALCF's GPU system (Eureka). Two crossflow stream temperatures (T_{cf} =930 and 950 K) were simulated at a Reynolds number, based on the friction velocity and the channel half-width, of $Re_\tau = 180$. Typical instantaneous snapshots of the fully ignited case are shown in fig. 1.1.

The simulations performed up till now employed close to 345 million grid points corresponding to 4.8 billion degrees of freedom (14 unknowns per grid point). We are currently developing the computational mesh for a planned high Reynolds number simulation ($Re_\tau = 590$). It is estimated that the total number of elements will be close to 3 millions and the polynomial order will be at least 8.

Chapter 2

Parallel Scaling and parallel efficiency metrics

2.1 Measurement of Parallel Efficiency

Parallel code scaling focuses on one of two forms: strong scaling or weak scaling. The goal of strong scaling is to reduce execution time for a fixed total problem size by adding processors (and hence reducing problem size per worker/MPI rank). On the other hand, ideal weak scaling behavior is to keep the execution time constant by adding processors in proportion to an increasingly larger problem size (and hence keeping problem size per worker fixed).

Parallel efficiency, η, for a problem run on N_1 processes/MPI ranks is defined relative to a reference run on N_2 ranks as

$$\eta_p = \frac{N_1\ t(N_1)}{N_2\ t(N_2)},$$

where $t(N)$ is the execution time on N ranks.

In order to ensure core utilization we measure the MPI-threading efficiency which we define as

$$\eta_t = \frac{N_1\ t(N_1)}{N_2\ t(N_2)},$$

where N_1 is the number of cores for the one MPI rank/thread per core configuration and $N_2 = 2N_1$ in the two threads/core and $N_2 = 4N_1$ in the four threads/core configurations. Note that a net speedup is obtained for threading efficiency of more than 50% in the two threads/core configuration and more than 25% in the four threads/core.

2.1.1 Core-level MPI Threading Efficiency

Instantiation of two and four MPI-ranks per core is enabled on Mira and hence full core compute power utilization is assured through the use of more than one rank per

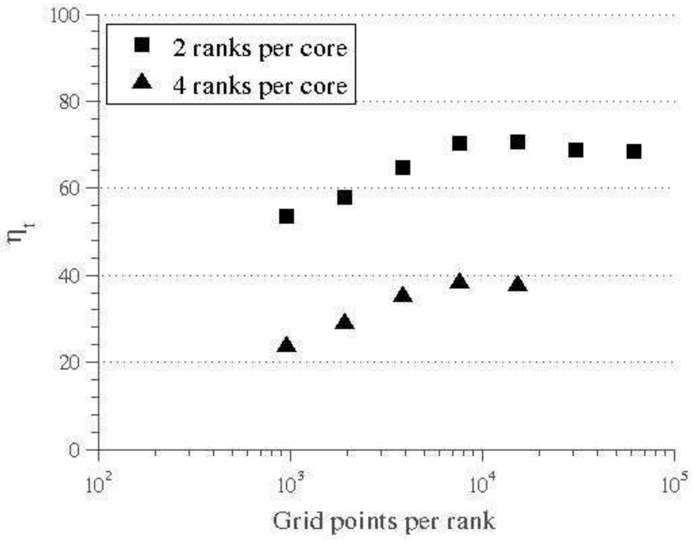

Figure 2.1: MPI threading efficiency for autoignition simulation using NEK5000. Note that the abscissa represents grid size per MPI rank.

core. However, this may lead to performance degradation for very large problem size because of the resulting on-chip resource contention. We have conducted a core-level parallel efficiency measurements on Mira for the target simulations using different number of cores and either two or four ranks per code.

Figure 2.1 compares the MPI threading efficiency for two and four ranks per core. It is clear that using two ranks per core is more efficient for same problem size (i.e. grid points) per rank and that the optimum grid points per rank is 7,000-10,000. This size ensures maximum utilization of the core's compute-power while minimizing on-chip resource contention among MPI threads.

2.1.2 Strong Scaling Experiment

A strong scaling is performed under the two MPI-ranks per core configuration for an autoignition simulation with a total of $\approx$ 345 million grid points.. Figure 2.2 shows that an ideal efficiency of a 100% is mainated up to approximately 130,000 MPI-ranks (65,000 cores) and that up to 60% efficiency is sustainable using nearly 500,0000 ranks.

2.2 Profiling and Hardware Performance Monitoring

Performance monitoring is enabled through a mechanism for obtaining information about the use of MPI routines (profiling) or wall clock time and hardware counters

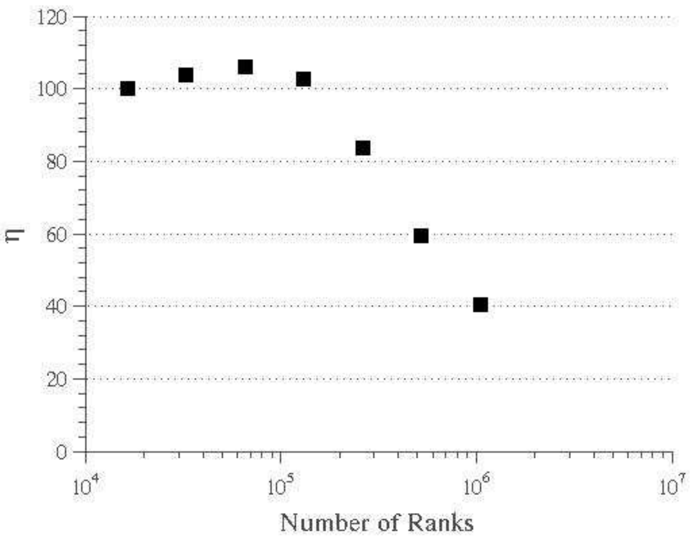

Figure 2.2: Parallel efficiency for autoignition simulation: strong scaling with $\approx$ 345 Million gridpoints and two MPI ranks per core.

for a code as a whole or a segment that is usually bracketed with calls to initiate and stop monitoring libraries.

2.2.1 IBM HPC Toolkit

IBMs' HPC Toolkit provides a mechanism for tracking the use of MPI routines during a programs execution. This is done through the use of a library which intercepts calls to MPI routines, records information about the call, and then continues with the MPI call. The primary types of information that are gathered are

- MPI Profile Data: A summary of MPI usage information typically consisting of the number of times each MPI routine was called, how much time was spent in each MPI routine, and the average size of a message for that routine.
- MPI Trace Data: A detailed time-history of every invocation of an MPI routine that the program made.
- Point-to-Point Communication Pattern: This information is derived from the collected trace information and indicates the number of bytes sent between ranks by point-to-point MPI routines.

IBM also provides a HPM Toolkit for hardware perfromance monitoring that has

- A utility *hpmcount*, which starts an application and provides at the end of execution the wall clock time, hardware performance counters inforamation, derived hardware metrics, and resource utilization statistics

- An instrumentation library *libhpm*, which provides instrumented programs with a summary output containing the above information for each instrumented region in a program (resource utilization statistics is provided only once, for the whole section of the program that is instrumented). This library supports serial and parallel (MPI, threaded, and mixed mode) applications, written in Fortran, C, and C++.

2.2.2 Nek5000 Instrumentation

In order to focus the analysis on the time stepper part of NEK5000, exclude restart-file reading and recurring IO, the time stepping loop was bracketed by calls to the respective libraries as follows:

```
call summary_start()
call hpm_start("nek_advance")
DO ISTEP=1,NSTEPS
  TIME STEPPING LOOP
ENDDO
call hpm_stop("nek_advance")
call summary_stop()
```

It should be noted that the code needs to be compiled with the debugging flag ”-g” enabled. Finally, the following link to IBM libraries needs to be appended to the list of libraries (`USER_FLAGS`) used by NEK5000:

```
USR_LFLAGS="${USR_LFLAGS} -L/soft/perftools/hpctw -lmpihpm
-L/bgsys/drivers/ppcfloor/bgpm/lib -lbgpm"
```

2.2.3 Sample MPI Profile

A small section of a sample profile dumped by rank 0 is shown below.

```
Data for MPI rank 0 of 65536
Times and statistics from summary_start() to summary_stop().
-----------------------------------------------------------------
MPI Routine  #calls avg. bytes  time(sec)
-----------------------------------------------------------------
MPI_Isend    508287 915.7 2.083
MPI_Irecv    508287 916.1 0.416
MPI_Waitall 162759 0.0 8.845
MPI_Bcast   1 4.0 0.000
MPI_Barrier 81 0.0 0.002
```

```
MPI_Allreduce  19296 2716.7 17.573
...etc.
-----------------------------------------------------------------
total communication time = 28.919 seconds.
total elapsed time = 195.156 seconds.
heap memory used = 103.090 MBytes.
```

Although the MPI-time is dominated by synchronization (`MPI_Allreduce` during global inner product evaluation for the linear system solve step for the velocity, pressure and thermo-chemistry), it is still representing only about 14% of the total execution time. The communication as well as total execution/elapsed time for all MPI-ranks is plotted in figure 2.3. It can be seen that the communication time as well the total elapsed time (and hence the computation time) is approximately uniform across all MPI-ranks, an indication of load balancing between the MPI ranks.

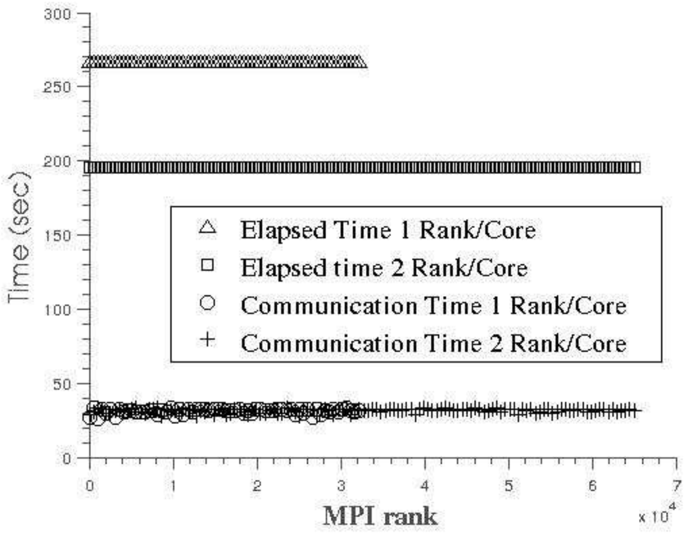

Figure 2.3: Communication and elapsed time for MPI ranks collected with IBM's HPC Toolkit

2.2.4 Hardware Performance

Hardware performance metrics are written to `hpm_summary.rank` files. The file lists information about the overall performance of the whole code as well as information about the bracketed segment of the code (the time stepper in this case).

Table 2.1 shows that with the use of two MPI-ranks per core the floating point operations per second per node has increased from 4.398 to 6.057 GFLOPS ensuring higher utilization of the core compute-power. Finally, it can be seen that there is a trend of increasing access to both the L1P, L2 Cache and main memory with increasing number of ranks per core. This is most likely the reason beyond performance degradation for the four ranks per core configuration. Measurements from a four ranks per core were not possible due to the overhead of the measurement libraries themselves, even with light-weight versions of the same libraries provided by IBM.

Table 2.1: Summary of Hardware Performance Metrics Measured by hpm Library

	1 Rank/Core	2 Ranks/Core
FPU% , FXU %	20.6-79.4	17.7-82.3
Inst/cycle/core	0.3586	0.5794
GFLOPS/node	4.398	6.057
% of max issue rate/core	35.9	47.7
L1 d-cache hit	96.4	95.5
L1P buffer hit	1.46	2.09
L2 Cache hit	1.85	2.0
DDR hit	0.32	0.4
DDR total traffic (Bytes/cycle)	2.264	3.588

Chapter 3

Discussion of ESP Experience

The highly-scalable low Mach number reactive flow solver based on NEK5000 is used to simulate turbulent autoigniting flows in a laboratory scale jet-in-cross-flow configuration. The code relies heavily on two highly optimized mxm and thermo-chemistry PTTP kernels that were previously optimized on the BGP architecture (Kerkemeier, 2010).

Scaling results on Mira showed that ideal parallel efficiency of 100% can be sustained up to 130000 MPI ranks and up to 60% efficiency on nearly 500,0000 ranks and hence no further tuning is necessary for the current problem size.

We are currently evaluation different design strategies for the computational grid that will be used for the high Reynolds number simulations. Further optimization of grid-generation routines for the Algebraic Multigrid solver is necessary for tackling the high Reynolds number grids and is the subject of ongoing investigation.

Bibliography

Cohen, S. and Hindmarsh, A. (1996) CVODE, a stiff/nonstiff ODE solver in C. *Comp. Phys.* **10**(2), 138–143.

Deville, M., Fischer, P. and Mund, E. (2002) *High-order methods for incompressible fluid flow*. Cambridge University Press.

Dobbeling, K., Hellat, J. and Koch, H. (2007) 25 years of BBC/ABB/Alstom lean premix combustion technologies. *Trans.-ASME J. Eng. Gas Turb. Power* **129**(1), 2.

Fischer, P., J.W., L. and Kerkemeier, S. (2008) nek5000 Web page. http://nek5000.mcs.anl.gov.

Kee, R., Dixon-Lewis, G., Warnatz, J., Coltrin, M. and Miller, J. (1986) Technical report sand86-8246. *Sandia National Laboratories* .

Kerkemeier, S. (2010) Direct numerical simulation of combustion on petascale platforms. Ph.D. thesis, Swiss Federal Institute of Techonolgy Zurich (ETHZ), Nr. 19162.

Micka, D. and Driscoll, J. (2012) Stratified jet flames in a heated air cross-flow with autoignition. *Combust. Flame* **159**(3), 1205–1214.

Tomboulides, A., Lee, J. and Orszag, S. (1997) Numerical simulation of low mach number reactive flows. *J. Sci. Comp.* **12**(2), 139–167.