Distributed Tabulation of Flamelet Lookup Tables

Nicholas Abel*†, Ricard Borrell Pol*, Daniel Mira Martinez*
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
E-mail: {nicholas.abel, ricard.borrell, daniel.mira}@bsc.es

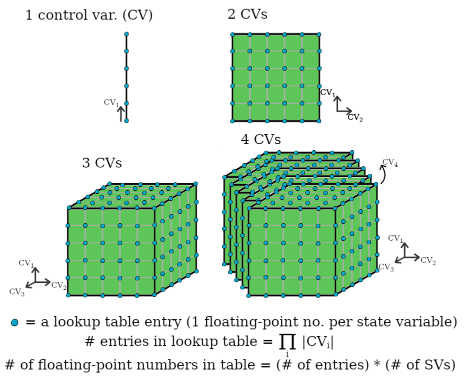
1 control var. (CV)
2 CVs
3 CVs
4 CVs
● = a lookup table entry (1 floating-point no. per state variable)
# entries in lookup table = ∏ |CVᵢ|
# of floating-point numbers in table = (# of entries) * (# of SVs)

Fig. 1. The discretization of the flamelet lookup table can be visualized on the p-dimensional unit hypercube, where p is the number of controlling variables. If there are N discretization points of each controlling variable, the total number of floating point numbers stored in the lookup table is SN^p where S is the number of state variables, or arguments to the function being retrieved.

***Keywords*—combustion, tabulated chemistry, mpi3-rma, High-Performance computing.**

I. Extended Abstract

One of the fundamental questions in combustion simulation is how to account for detailed chemistry effects, while controlling both the error of the chemical scheme and the computational cost. Combustion chemistry is important for resolving processes such as flame propagation and pollutant formation, which are non-linear processes that can be computationally expensive. The direct solution of the governing equations of turbulent reacting flows can be prohibitively expensive as the chemical integration is often stiff. Tabulated chemistry methods with flamelet modelling emerge as an alternative to perform direct integration of the chemical source terms and has been extended to a wide range of conditions [1].

In flamelet methods, the chemical time scale is assumed smaller than the time scales of the turbulence, so the flame structure is not affected by the turbulence. In flamelet methods, the thermochemical states of the flame are computed in a pre-processing step, and these values are retrieved from a lookup table loaded into memory at the beginning of the simulation. The flame structure can be recovered through the use of controlling variables, which represent dimensions along the multidimensional space of the flame manifold.

A restriction of the tabulated chemistry method is that it is bound by memory. Increasing the number of dimensions, the number of tabulated values per dimension, and the number of discretization points on the flamelet manifold as desired are not possible given the way that tabulated chemistry is currently implemented in combustion codes [2]. In particular, the entirety of the lookup table is stored on each core on distributed-memory machines. Modern supercomputers typically have about 2GB of memory on each core, and tables quickly outgrow this limit as they increase in dimension (see Figure 1).

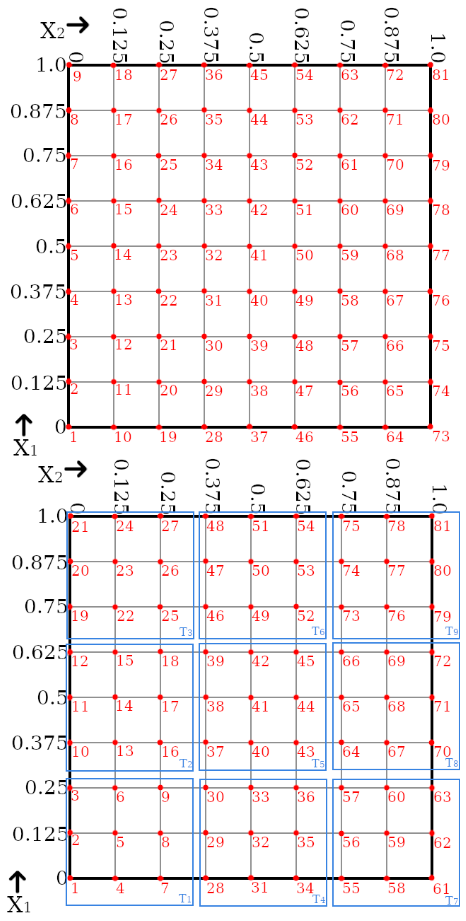

Fig. 2. An example of block structuring applied to a small 2D lookup table, with controlling variables labelled X_1 and X_2. Numbers 1-81 give the ordering of entries as they appear in memory before and after applying the block structure. Here, the table size is 9×9 and the block size is 3×3. The blocks are labelled T_1 to T_9.

Since combustion simulations are typically run in parallel across many compute cores, the memory-boundedness of the current implementation may be addressed by storing the lookup table in a distributed manner, as opposed to keeping a

D_T ($D_R = 50$)	1	2	5	10	20	25	50	100	250	500
Wall Time (s)	1.3858	1.2581	1.2532	1.2344	1.1940	1.2697	1.2209	1.1985	1.2733	1.2264
Speedup	1.0000	1.1014	1.1057	1.1227	1.1606	1.0914	1.1350	1.1563	1.0884	1.1299
D_T ($D_R = 100$)	1	2	5	10	20	25	50	100	250	500
Wall Time (s)	1.8338	1.6665	1.4573	1.9551	1.6779	1.4743	1.6022	1.4384	1.4908	1.6966
Speedup	1.0000	1.1004	1.2584	0.9380	1.0929	1.2439	1.1445	1.2749	1.2301	1.0808

TABLE I. WALL TIME (IN SECONDS) AND SPEEDUP FOR 10^8 RANDOM ACCESSES TO THE TABLE, FOR ACCESS RANGES $D_R = 50, 100$ AS BLOCK SIZE, GIVEN BY DT, VARIES.

copy of the entire table in memory at each core. We refer to this method as **distributed tabulation**. The goal of our project is to develop, implement, and test distributed tabulation in the multiphysics code Alya [3].

A. Methodology

The most naive approach to distributed tabulation is as follows. Consider a lookup table that contains N_E total entries, used in a simulation run on N_R total compute cores. Then, at the beginning of runtime, we load N_E/N_R table entries into memory onto the each core (assume for simplicity N_E is divisible by N_R). Then, using MPI 3's remote memory access capabilities [4], we open a memory window on each rank's table entries to permit one-sided access during the solve phase of the simulation. If a sub-domain needs an entry which is stored on another sub-domain, it will retrieve that entry using a MPI Get call. This technique alone will solve the problem of memory-boundedness given enough parallel resources, but in the worst case it will incur a significant cost in overhead due to communication.

To reduce communication, we exploit a pattern of locality which tends to occur in flamelet table lookups. When a sub-domain requires a lookup table entry, successive entries requested by that sub-domain will tend to be spatially near previous entries in the flamelet manifold. To exploit this, we organize the table in a block structure and keep recent remotely-retrieved entries in a small local pile.

The block structure is a partitioning over the unit hypercube. Table entries which are spatially collocated on the unit hypercube are rearranged to be contiguous in memory. An example of block-structure reorganization on a table with 2 controlling variables is illustrated in Figure 2.

Then, when a sub-domain requests a non-local entry, it will use MPI remote memory access to retrieve all entries on its block and store the block locally on a pile. When a sub-domain needs a non-local entry, it looks first in the local pile to see if the entry is available there. The local pile is restrained by a maximum number of stored blocks. When a block is added to the pile or used locally, it is moved to the front. As a result, blocks that are not used for some time are removed from local memory.

B. Test on locality optimization

Distributed tabulation reorganizes lookup tables such that entries of the tables which are spatially co-located on small subsets of the unit hypercube, called blocks, are stored contiguously in memory. Combustion simulations tend to successively request lookup table values which are close to one another in space, but current lookup table strategies do not necessarily store spatially close values near one another in memory. While block structuring of lookup tables was devised with message passing in mind, it may also increase CPU cache efficiency and decrease the total time needed to retrieve local lookup table values. To test this effect, we run the following test on a single MPI rank.

Using a three-dimensional lookup table of size 1000×1000×1000, we choose a range R on the table with dimensions $D_R \times D_R \times D_R$, for $D_R = \{50, 100\}$, in which we access 10^8 randomly chosen elements. For each size of D_R, the size of the blocks T on the cube are chosen as $D_T \times D_T \times D_T$, for $D_T = \{1, 2, 5, 10, 20, 25, 50, 100, 250, 500\}$. Table I shows the wall-time (in seconds) and speedup comparing 10^8 accesses on R for $D_R = 50$ and $D_R = 100$, as the block size T defined by D_T varies. This demonstrates that the block structure can yield modest sequential speedup for repeated access of spatially close entries on the table, a side-effect of a functionality intended for point-to-point communication.

REFERENCES

[1] J. Benajes, J. García-Oliver, J. Pastor, I. Olmeda, A. Both, and D. Mira, "Analysis of local extinction of a n-heptane spray flame using large-eddy simulation with tabulated chemistry," *Combustion and Flame*, vol. 235, p. 111730, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010218021004739

[2] E. Illana, D. Mira, and A. Mura, "An extended flame index partitioning for partially premixed combustion," *Combustion Theory and Modelling*, vol. 25, no. 1, pp. 121–157, 2021. [Online]. Available: https://doi.org/10.1080/13647830.2020.1841912

[3] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, E. D. Burness, J. M. Cela, and M. Valero, "Alya: Multiphysics engineering simulation toward exascale," *Journal of Computational Science*, vol. 14, pp. 15–27, 2016, the Route to Exascale: Novel Mathematical Methods, Scalable Algorithms and Computational Science Skills. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877750315300521

[4] T. Hoefler, J. Dinan, R. Thakur, B. Barrett, P. Balaji, W. Gropp, and K. Underwood, "Remote memory access programming in mpi-3," *ACM Trans. Parallel Comput.*, vol. 2, no. 2, jun 2015. [Online]. Available: https://doi.org/10.1145/2780584

Nicholas Abel received his BSc degree in Mathematics from The University of New Mexico (UNM) in 2017, and completed his MSc degree in Mathematics from UNM in 2020. He currently is a member of the Computer Applications in Science and Engineering (CASE) group at the Barcelona Supercomputing Center (BSC), as well as a PhD student in the School of Mathematics and Statistics at Universitat Politècnica de Catalunya (UPC).