Interactive Analysis of Large Combustion Simulation Datasets on Commodity Hardware

Category: Applications

Abstract— Recent advances in high performance computing have enabled large scale scientific simulations that produce massive amounts of data, often on the order of terabytes or even petabytes. Analysis of such data on desktop workstations is difficult and often limited to only small subsets of the data or coarse resolutions yet can still be extremely time-consuming. Alternatively, one must resort to big computing clusters which are not easily available, difficult to program and with software often not flexible enough to provide custom, scalable analyses. Here we present a use case from turbulent combustion where scientist are interested in studying, for example, the heat release rate of turbulent flames in a simulation totaling more than 5.5 TB of data. This requires a non-trivial processing pipeline of extracting iso-surfaces, tracing lines normal to the surface and integrating the heat release for all 660 time steps to accumulate histograms and time curves. Using existing tools, this analysis even at reduced resolution, takes days and is thus rarely performed. Instead, we present a novel application based on light-weight streaming processing and fast multi-resolution disk accesses that enables this analysis to be done interactively using commodity hardware. In particular, our system allows scientist to immediately explore all parameters involved in the process, i.e. iso-values, integration lengths, etc., providing unprecedented analysis capabilities. Our application has already led to a number of interesting insights. For example, our results suggest using an adaptive iso-value and integration length for each time steps would lead to more accurate results. The new approach replaces a traditional batch-processing work flow, and provides a new paradigm in scientific discovery through interactive analysis of large simulation data.

Index Terms—Large-Scale Data, Streaming Processing, Turbulent Combustion, Multi-Resolution Data Access.

1 Introduction

Today's cutting-edge scientific research increasingly relies on large scale computational modeling and simulation as a means to advance our understanding of fundamental physical and chemical processes. With the rapid increases in available computing power, scientists are able to perform simulations that are larger, more detailed, and more realistic from first principle descriptions. Aside from the generation of large-scale high-fidelity simulations, however, the combination of unprecedented data sizes and more intricate analysis algorithms creates challenges in extracting scientific knowledge from the data. Even with the state-of-the-art visualization and analysis tools [2, 4], common analysis tasks may require substantial computing resources and personal time, especially for any queries involving large time series. Consequently, these tasks can be performed only on a subset of data or at a lower resolution, limiting the ability to explore all useful information in simulation.

Alternatively, such an analysis could be conducted in situ as part of the simulation. However, in many cases the various parameters used are not known a priori. Sometimes, traditional defaults can be used or appropriate values could be extrapolated from similar use cases. In either case, this can easily lead to incorrect results and missed opportunities, thus wasting the resources and time. Furthermore, these limitations tend to encourage a conservative approach in which the same analysis pipelines with the same parameters are used since exploring and validating new techniques is too costly.

This paper presents a prime example of the challenges discussed above in the area of combustion research. Dynamic interactions between turbulent flows and premixed flame propagation have been a long-standing fundamental science question, with practical significance in identifying the engine operation conditions for higher efficiency, lower emissions, and combustion stability. In particular, how the net burning velocity varies as a function of the turbulence intensity is still not clearly understood. As such, direct numerical simulations (DNS) of premixed flames have been pursued at higher levels of turbulence intensity, requiring extremely large scale simulations with billions of grid points and millions of time steps. The massive datasets produced by such simulations often require a variety of statistical analysis associated with flame surfaces identified as iso-surfaces, such as local and surface-integrated front speed, heat release, and strain rates. Understanding how flames of different chemical mixtures respond to intense turbulence will help provide information about their stability under operation conditions. In addition, this information can be used to inform the development of new combustion systems that run at fuel-lean conditions, and are therefore less polluting and more energy efficient, which will result in the reduction of emissions of particulates and carbon dioxide from combustion devices.

The present study shows an example of such an analysis by computing the consumption speed of the flame, defined as the surface-normal integrated reaction rate of a chosen reactant species [3, 7]. Computing the consumption speed involves extracting an iso-surface, and integrating heat release rate along the normals on both sides of the surface. The results are summarized through histograms over time. Using the existing post-processing infrastructures, even a lower resolution approximation of some medium sized data set can take days. As a result, such an analysis is commonly performed with a fixed set of the parameters such as the iso-value, integration length, etc. The present study employs the recent advances in progressive, streaming data processing, pioneered in the visualization community, resulting in an interactive pipeline to compute the consumption speeds and related measures using commodity hardware.

The new analysis framework allows scientists to interactively explore not only the core parameters such as the iso-value and integration length but also different resolutions in both space and time. Furthermore, given the integration values, the surface can be colored according to the local heat release rate, providing a simple way to study the spatial distribution of the heat release. The analysis has produced a number of unexpected results, allowing, for the first time, the assessment of the stability of the results under different parameter choices in a way that was not previously possible. For example, the analysis clearly showed the sensitivity of the integration length to the surface's curvature, as well as that of the iso-value to the integration length. More specifically, the present study provides combustion scientists with the ability to:

- Interactively compute flame consumption speeds and related measures from large-scale time dependent data.
- Freely explore all parameters involved in the computation.
- Compare and contrast results for different parameter settings and time ranges.
- Interactively visualize time dependent iso-surfaces of massive data colored by the local flame consumption speed.

In summary, the present framework provides scientists unprecedented capabilities to explore and analyze the large and complex simulation data sets. Furthermore, this only requires a simple commodity workstation rather than the large-scale distributed resources that would likely be required for alternative approaches.

2 Related Work

In combustion research, a flame's displacement speed is defined through a one-dimensional flamelet approach [6], which describes the dynamics of the flame speed contributions of the diffusion of mass into the flame sheet and the reactivity of the combustion reactants. This approach can be extend to define displacement speeds for relevant species across a flame with finite thickness. Similarly, we can extract a non-dimensional consumption speed for the chemical energy of the flame. This can provide us with information about how the topology of the flame, which is affected by the turbulence of the fluid, changes the overall consumption of premixed fuel.

The definition of the non-dimensional consumption speed is defined as:

$$S_h = \frac{\int \omega_{hr} d\eta}{\int \omega_{hr,lam} d\eta} \qquad (1)$$

where ω_{hr} refers to the heat release rate of the gas and η refers to the integration length. Since the heat release rate can vary in intensity and has the potential of being non-monotonic, we choose a value of a scalar that varies monotonically across the flame front. Consistent with definition of displacement speed in previous work [7], we choose O_2 mass fraction that corresponds to the peak heat release rate at the one-dimensional laminar condition.

Previous work was able to extract integrated information for two-dimensional conditions, in particular for the study of extinction by water spray of diffusion flames[1]. Day et al. [5] utilized a Lagrangian approach to trace path lines across the flame.

In terms of computation, it is possible to use existing systems such as MATLAB [13], VisIt [4] or ParaView [2] to compute the integration of heat release across a chosen iso-surface. in particular, ParaView and VisIt use the VTK framework [18] and are able to express copmlex analysis pipelines. However both tools assume the data to be in memory, which in this case would require substantial compute resources. Depending on the location of the computing resources interacting with the solution, for example, to change parameters can be difficult. Furthermore, neither system provides an easy path to use coarser resolution or subsets of the data. Finally, VTK pipelines are not designed progressively and will only return a single final result often after considerable time for complex tasks. Instead, the system presented in this paper remains interactive by progressively processing the data and reporting intermediate results and only requires a personal computer.

More recently several libraries and frameworks have emerged to extend VTK to multi-core and heterogeneous architectures, such as EAVL [14], DAX [15], or PISTON [11]. These solutions could improve the responsiveness of a VTK based approach and add features such as multi-core parallel processing. However, these are not yet mature enough to be easily used especially by non-experts. More importantly, they do not address the in memory requirement thus requiring the same resources or suffering from a severe file I/O bottleneck.

Our system is based on ViSUS [17], a fast, progressive streaming data analysis and visualization framework. ViSUS uses the IDX data format [10], an open-source I/O format that supports very efficient parallel write [9], and multi-resolution read through the use of hierarchical Z indexing [16]. We leverage the I/O capabilities of the IDX format to build a fast, general stream processing pipeline and apply it to the analysis of combustion flame speed in this paper. The idea of stream processing of data is certainly not new. There have been streaming algorithms developed for specific problems such as [8] and [19]. Our progressive data streaming pipeline however is much more general and is integrated into a full-scale system.

3 Applications

Real combustion systems typically favor turbulence in order to allow for fast conversion of chemical energy, since turbulence can significantly wrinkle a flame area and speed up the consumption of fuel and oxidizer. On the other hand, intense turbulence can often lead to dynamics that result in flame extinction due to thermo-diffusive effects or flame annihilation. For example, gas turbines that operate at lean conditions are known to exhibit instabilities that result in catastrophic failures due to the coupling of thermodynamics and acoustics. In jet engines, flame blowouts can occur due to the lack of anchoring of the diffusion flame within the combustor, even though the chemical energy conversion is shown to remain quite vigorous. Thus, studying how flames tend to behave under strong turbulence interactions can provide us with insights about possible modes of instabilities that might adversely affect flame, as well as insights towards mitigating these unstable modes.

In the context of premixed flames, researchers interested in combustion physics look to understand the effects of turbulence on the structure of the flame surface. Real combustion systems involving premixed flames tend to produce flames with some sort of mean shear (gas turbines) or mean curvature (gasoline engines). This science study seeks to understand the effects of the localized flame surface's corrugation in the absence of these larger scale effects in order to extract universal scaling laws for turbulent flame speeds that engineering-based models can utilize.

Computing the histogram of a flame's consumption speed one way of extracting information about the flame dynamics and is useful in determining to what extent a flame deviates from laminar conditions when affected by turbulence. Extracting the consumption speed requires an integration of heat release rate across a path that is normal to a selected iso-surface. The histogram of the consumption speed can provide us with an idea of how the flame is affected by the action of turbulence; that is, to what degree the consumption of energy across the flame's normal is affected by the corrugation of a selected iso-surface that corresponds with the flame location. In addition, a visual inspection of the locations where peak heat release rate is enhanced or diminished can provide us with details about how the topology of the flame creates localized regions of flame speed enhancement. This is crucial in determining how the characteristics of turbulence can lead to either favorable or unfavorable outcomes in the combustion regime of interest.

4 System

To provide the analysis capabilities needed to solve the challenges described above and do so interactively, we have developed a highly flexible data processing framework. The system consists of two main components: a cache-oblivious, multi-resolution data access library and a progressive, streaming dataflow model for processing. Here we first describe these components in general before discussing the specific analysis pipelines used to compute flame speed and related measures.

4.1 Cache-oblivious, multi-resolution data access

The data access of our system is based on reordering the data according to a hierarchical Z-order. In particular, we are using a publicly available version of the IDX format [10] originally introduced in [17]. It provides highly efficient, cache-oblivious access to subsets of spatio-temporal data with variable resolution. In our system this is coupled to a distributed client-server architecture based on the notion of progressive queries. More specifically, the client requests data or some subset of data, at a given resolution and at a specified granularity. The server produces this data in several non-redundant pieces starting with a small low resolution approximation which is subsequently refined according to the requested granularity. The actual data can reside either locally, remotely, or even be distributed among several sites. This aspect of our system has two key properties that are crucial to deliver the capabilities discussed in Section 5. First, it restricts the file I/O as well as any potential network transfers to only a small amount beyond what is strictly required by a given query. This is distinctly different from most existing analysis frameworks [4, 2], which typically must load, for example, an entire field to access a spatial subset or a subsampled approximation. Second, the progressive nature of the query response delivers some coarse approximation (in both space and time) of the data very fast and subsequently refines it. As discussed below, we have developed a suit of algorithms that take advantage of this progressivity and start processing data as soon as it arrives. Pipelining these algorithms results in a highly responsive and typically interactive system in which each user request produces an almost immediate if approximate answer that is continuously refined depending on how fast data can be loaded and processed.

4.2 Progressive streaming data processing

The second component of our system is a highly flexible data flow framework specifically designed to support progressive, streaming processing.

At the center of our system is a dataflow consisting of nodes linked together through input and output ports which store messages. Nodes communicate by publishing and receiving messages, which can be as small as a boolean variable or as large as a whole volume of data. We use ubiquitously a generic 1-D C++ array type to store multi-dimensional data, as array elements are stored contiguously in memory, the data structure is very cache friendly. To avoid unnecessary data copying, which can be very costly, message passing is always handled through shared pointers.

Nodes and the links between them form a Directed Acyclic Graph. An example of a node is the iso-surface extraction module, which takes a scalar field and outputs a mesh. A rendering node then takes the extracted mesh and renders it to the screen. Every node is agnostic about which nodes connect to it and which it connects to; it simply receives an input, processes the input, and publishes an output. This modular design brings great power and flexibility: a complex analysis can be expressed by putting together a small set of data processing nodes, which can be re-used in other analysis or visualization tasks.

Data movement is often the main bottleneck in most data visualization and analysis software. Since our dataflow does not keep a global state (i.e., each node is restricted to work on its own inputs), and does not have any synchronization mechanism built in, data can be simply streamed through a dataflow in a push-based manner. This allows for very efficient data movement and is the key to our system's interactivity.

Our system is also interactive in that the user interface remains responsive during data processing. The main GUI thread is also responsible for managing and dispatching dataflow's messages. If a node does heavy processing, it runs on its own thread. Since message passing and data processing run on different threads, long computation can be preempted whenever new inputs come to replace old inputs.

A dataflow can also help limit the amount of data to process. For example, if the user only cares about a 2D slice in a big 3D volume, the server can extract only this slice and send it to the client, which can do further processing on the slice. Such a data extraction task is called a query. Users can write a script to query data from different sources, possibly even on different servers, and mix them together as one data source. Data is cached on the client at query level.

By default, data querying works in progressive mode. Low resolution data will be read and sent first, finer and finer resolution data will be streamed in as the user waits, until some desired resolution level is reached. For visualization, this maximum resolution level is determined by the size of the on-screen projection of visible data in pixels, and the data's native resolution. The user, however, can manually specify a desired resolution, as well as disable progressive streaming. Non-progressive mode might be more preferred for analysis tasks, where the user often wants full control over the resolution levels at which the analysis is done.

4.3 Progressive algorithms for combustion analysis

To calculate a combustion flame's speed, we extend the general set of data processing nodes in our system with a special node that calculates integration along the normals of a given surface. This node takes in an iso-surface of the field O_2, and for each point on this surface, integrate

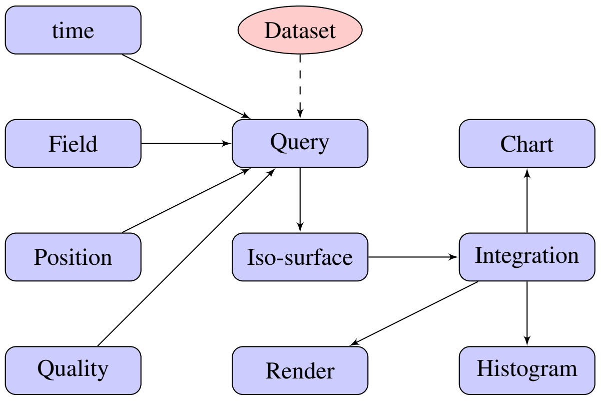


Fig. 1: Dataflow of the analysis. Data is pushed in the direction of the arrows, except for the dashed arrow, which means pull-based data movement.

values from the field HRR (Heat Release Rate) for a short distance along the surface normal at the point. Note that this node is completely general and one can substitute O_2 and HRR for any two other fields. We also observe that this type of integral computation along normals to a surface is relatively common in combustion analysis, justifying the inclusion of such a processing node in our system's library. The whole dataflow for this analysis is depicted in Figure 1.

The user specifies a time step using a Time node, and enters O_2 and HRR as two fields of interest into a Field node. A Position node allows the user to, for example, zoom into the data and thus limit the analysis to only the visible portion of the data. The Quality node controls the desired resolution level (discussed in Section 4.2). Using all these inputs, a Query node extracts two fields, 0_2 and HRR from the dataset, interleaves them, and pushes the result to the Iso-surface node. This node allows the user to enter an iso-value, extracts the corresponding iso-surface from the first field (O_2) and outputs it as a triangle mesh. This mesh and the second scalar field (HRR) are passed on to the next node, Integration, which computes an integral of HRR (IHRR) for each vertex on the mesh. The integration results are then passed on to a Histogram node, which draws a histogram of IHRR for the entire surface. The mean IHRR value for the entire surface is then sent to a Chart node. This allows the user to, for example, move the time slider and see this mean IHRR value evolves through time. Finally, the IHRR and the mesh are also sent to a Render node to be rendered onto the screen, each vertex colored by its corresponding IHRR value.

The user controls time, quality, iso-value, integration length (the flame thickness), and gets quick feedbacks from the histogram, the chart, and the rendered iso-surface. Note that each of the three output nodes (Render, Char, and Histogram) is optional and can be taken out without affecting the rest of the dataflow. Also, Quality is a dataflow node rather than just a property of Query because this way we can have another Quality node to control the quality of other data processing tasks, such as rendering (by connecting a Quality node to the Render node). The same idea applies to the other three input nodes (Time, Field, and Position).

For the current analysis of flame consumption speed, most of the algorithm is implemented in two nodes: Iso-surface and Integration. The Iso-surface node implements a fast variant of the popular marching cube method [12]. Once a iso-surface mesh is produced, we need to first compute a normal for each vertex. This is done by first identifying the all triangles adjacent to the vertex, computing a normal for each of them, then adding up all these normals, weighted by the areas of the corresponding triangles. We then step along this normal, both forward and backward, using the trapezoidal scheme to sum up tri-linearly interpolated samples from the field HRR. The step size is chosen so that the sampling frequency is never more than the native resolution of the data.

The integration results for the whole mesh is then fed into a Histogram node. This is a general histogram computing node that computes common statistics and draws a histogram for any given input array of values. The histogram of IHRR provides one of the main visual channels to observe the behavior of the flame's consumption speed as parameters are varied. Note that the Histogram node draws a normal, linear-scale histogram (in black color) and also a log-scale histogram (in light blue). In all our experiments and screenshots, the Histogram node is also configured to draw with a fixed scale in both axes, to make it easier to observe the behavior of the histogram as parameters change.

The mean IHHR value for each mesh is also output to an optional Chart node, which saves all values that it receives and plots them in a graph. With this node, the user can, for example, drags the time slider and see how the mean flame's consumption speed evolves over time.

5 Results

With our system, scientists can interactively explore the whole space of input parameters. In particular, here the user can vary the iso-value of O_2 to get a sense of where most reactivity happens (Section 5.2). The user can also vary the length of integration (or flame thickness) and watch the effects of this on both the histogram and the color mapping on the iso-surface, which can guide the user in choosing an appropriate length (Section 5.2). In fact, the color mapping of IHRR on the surface not only acts as a validation tool but can also give new scientific insights not easily obtained otherwise (Section 5.1). To further increase interactivity, quick analysis can be done with low-resolution data first to narrow down the range of parameters, followed by accurate but more expensive analysis of the parameters in the reduced ranges using high-resolution data (Section 5.4). We give details about the size of data and timing information for each step of the analysis in Section 5.5.

5.1 Color mapping of integrated heat release rate on surface

We normalize all IHRR values across the iso-surface and use a transfer function to map the computed IHRR values to colors. These colors are defined for the vertices and are interpolated across the whole surface. In all screenshots given in this paper, we use a "banded" transfer function that maps small absolute values to purple/blue colors, and large absolute values to orange/red colors. Intermediate values vary smoothly through different shades of cyan, green, and yellow. Note that because of the way we normalize the IHHR values, blue colors actually correspond to high absolute IHHR values, while red colors mean low IHHR values. The transfer function is shown in all screenshots, near bottom right corner. Users can easily specify their preferred transfer function. Iso-surfaces are rendered with standard Phong shading, with a gray diffuse color to not dilute the transfer function too much. See any screenshot of the system for an example of the coloring of iso-surface (e.g., Figure 5).

The colored iso-surface helps to describe the locations where reactivity occurs. We can also qualitatively describe how the topology of the surface affects the energy consumption speed. It is observed that values of high, absolute value of integrated reactivity correspond with positively curved areas. Low, absolute values of integrated reactivity occur at locations that are negatively curved. There are also "valley" areas where the surface curves negatively but with IHRR values that are not low. This is because as we integrate reactivity in the normal direction of points from these valleys, we end up picking up high heat release rate values produced by the two "sides" of the valley, where the surface has positive curvatures.

We can confirm that this phenomenon occurs as well when we look at the displacement speed correlation [6] with stretch as defined by the Karlovitz number Ka [3]:

$$Ka = Ka_{a_T} + Ka_c = \frac{\delta}{S_{L,0}}(a_T + S_d^* \nabla \cdot \mathbf{n}) \tag{2}$$

where Ka_{a_T} corresponds to the stretch component and Ka_c corresponds to the curvature component of total strain (Ka). Figure 2 shows

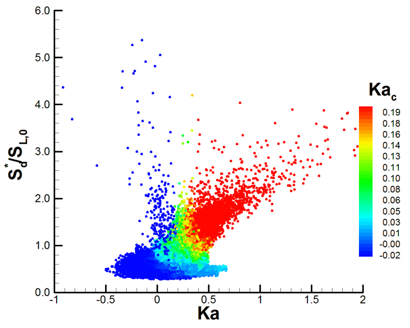


Fig. 2: Correlation of density weighted displacement speed with total strain (Ka), colored with the curvature component of total strain (Ka_c)

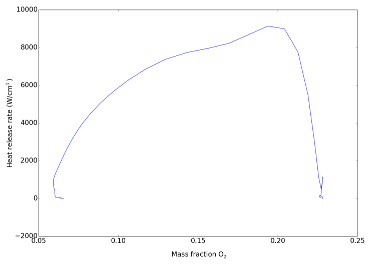


Fig. 3: Heat release rate versus mass fraction of O_2. The peak value of heat release rate corresponds approximately to a value of $Y_{O_2} = 0.19$.

that for positive curvature. The analysis provided by the consumption speed hints that the factor for this acceleration is likely due to enhanced reactivity at these positively curved areas, similar to what is observed by the color maps in subsequent figures. The speed by which we are able to extract these feature provides us with clues that inform our investigation of the causes for this acceleration, and is valuable towards understanding the physics we seek to investigate.

5.2 Effects of different iso-values and integration lengths

Choosing an appropriate iso-value of O_2 is crucial for the analysis. A value either too high or too low will not correspond to where most reactivity happens, and where most energy is released. The traditional way of determining this is to compute the value of the O_2 mass fraction that corresponds to the peak heat release point in the laminar flame condition, which in this premixed composition is $Y_{O_2} = 0.19$ as shown in Figure 3.

Using our system however, the user can just drag the iso-value slider and see the effects of it on the graph of mean IHRR drawn interactively (Figure 6). As the graph indicates, a value around 0.195 gives the most stable result, consistent with the laminar condition estimate. This can also be seen more clearly in Figure 6a, which is just a more complete version of the graph seen at the lower right corner of Figure 5d. If the iso-value is too large or too small, we run the risk of not integrating over the entire heat release rate profile. As a result, the color distribution on the surface becomes skewed and the histogram loses its Gaussian-like shape (Figure 5d).

If the user fixes a reasonably short integration length and drags the iso-value slider back and forth, the Chart node will show the graph seen in Figure 6a, indicating the iso-value that gives the most stable results (in this case $Y_{O_2} = 0.195$). Also, as Figure 6b shows, as we increase the integration length, the effect of the iso-value becomes less pronounced, as indicated by the mean IHRR curve becoming much flatter. Its peak value also drifts away from the laminar estimation of 0.19. This suggests that the ideal choice of a stable iso-value is very sensitive to the choice of integration length. This correlation between the iso-value and the integration length can be theoretically derived

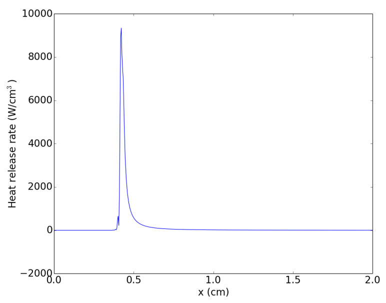


Fig. 4: Heat release rate versus distance for a laminar flame. The physical thickness of the laminar flame condition can be see to approximately span 1.0 mm.

once one knows of it, but without an environment that encourages interactive exploration and provide clear anecdotal evidences, subtle but important points like this can easily go unnoticed.

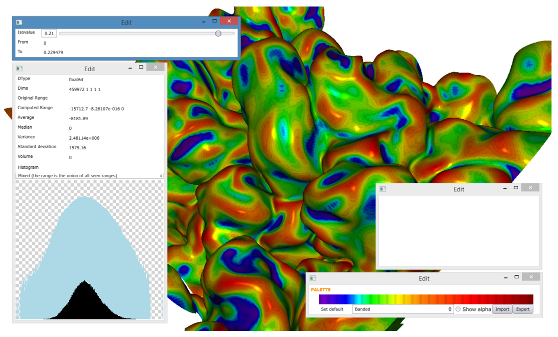
Choosing a right integration length is also important. A length too short does not capture the full range of the energy conversion profile. A long integration length on the other hand increases the risks of going through the surface at a different point in areas of high curvature, thereby overlapping the energy release associated with another section of the flame. Traditionally, scientists can arrive at a value for the integration length based on the physical thickness of the flame which can be observed in Figure 4. A flame thickness of 1 mm can be deduced from that graph. Our tool, however, shows clear evidences that a length of 0.3 mm is already too large, as seen in Figure 7, which shows integration length of 8 voxels (0.3 mm) skewing the IHHR distribution. This discrepancy can be explained in the context of highly corrugated flames, where choosing a shorter integration length ensures that we are not integrating through some of the highly folded flame surface locations. Again, Figure 7 shows this effect in both the histogram, which loses its Gaussian-like shape, and the surface colors, which turn significantly more yellow and red. A shorter length gives rise to a narrower and taller histogram, but the distribution remains relatively stable.

Given clear evidences showing that the integration length is sensitive to the flame surface's curvature, and the choice of iso-value is sensitive to the integration length, we could propose a new method of computing IHRR, where both the integration length and the iso-value are adaptive to each time step. Even with a fixed choice for these parameters, with our system, scientists can be more confident in their choice of parameters, found by experimenting and analyzing the system's feedbacks, compared to if they come up with some values just by purely rationalizing or guesswork.

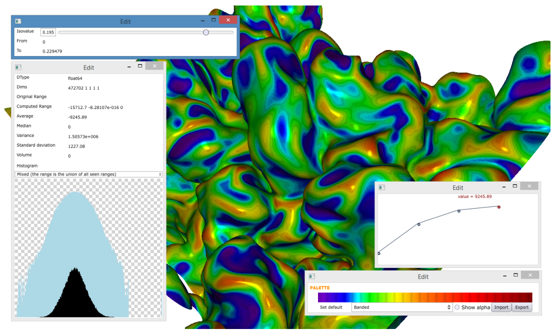
5.3 Integrated heat release rate as a function of time

A valuable tool to understand the overall flame behavior as a function of time is to compute the mean value of the consumption speed at the selected iso-surface. This can provide us with a metric that describes how the flame is affected by the intensity of the turbulence. Figure shows this behavior. The first response of the flame is simply a transient response associated with the initial flame profile. As turbulence is injected, the flame begins to increase its consumption of energy and is positively affected as it is folded and stretched. When the mean injection velocity of turbulence reaches a steady state, the overall consumption of energy exhibits a stable behavior throughout most of the lifetime of the simulation. At the last section, the flame exhibits an unstable mode that results in significant acceleration, leading to the flame propagating too close to the inlet boundary, and thus requiring the termination of the simulation.

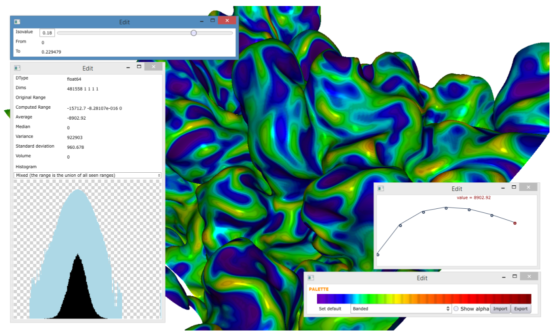
To aid users in understanding the flame behavior through time, our system can interactively plot the graph of mean IHRR as a function of time. We use low-resolution data (one-eighth the original) to maintain interactive dragging speed. The time slider advances 20 steps at a time, for a total of 33 time steps. It takes roughly two minutes and a



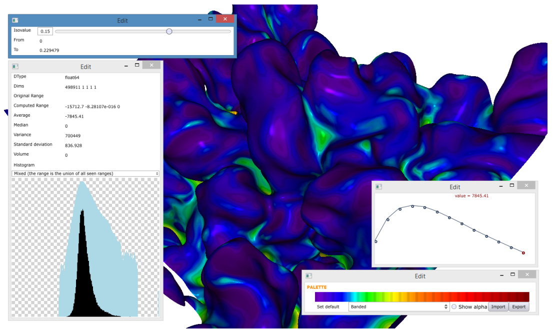
(a) Iso-value = 0.21



(b) Iso-value = 0.195

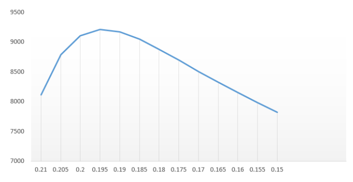


(c) Iso-value = 0.18

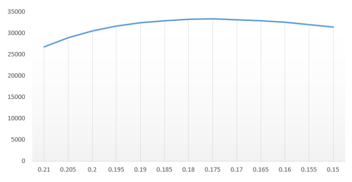


(d) Iso-value = 0.15

Fig. 5: Effects of different iso-values (at time step 200). We fix an integration length of 1 voxel (about 0.03 mm). From (a) to (e), the iso-value slider is dragged gradually from 0.21 to 0.15, with a step of 0.15.



(a) Integration length = 0.0375mm



(b) Integration length = 0.15mm

Fig. 6: Mean IHRR (vertical) as a function of iso-value (horizontal), for different integration lengths. These plots are the more complete version of the built-in, interactive graph.

half for the user to finish dragging the whole range of 660 time steps. If the Render node is removed (since rendering is not important here), the dragging time reduces to 48 seconds. In Figure 8, we compare our online graph with a graph generated offline by a Python script, using finer resolution data and more time steps. The two graphs match pretty well in their overall shapes. If the user only wants to get an overall sense of how the mean IHRR value behaves through time, the crude, interactive graph would suffice.

5.4 Effects of different resolutions

Figure 9 shows that when the quality level is reduced gradually from 0 to 3 (with 0 being to the finest resolution, and 3 corresponding to one-eighth of the data), the mean and standard deviation vary only slightly, while the overall shape of the histogram remains largely Gaussian-like, indicating it is most likely sufficient to do quick analysis at these resolution levels. When the quality is reduced to 4, however, the distribution of integration results change significantly and the histogram no longer resembles a Gaussian. This effect can also be seen from the color mapping on the surface. This large degradation of the results at quality level 4 suggests that level 3 is probably the lowest quality level for this dataset that we can do analysis on.

We also notice that the quality of the results depends a lot on the integration length. As we integrate a longer distance, errors add up quicker, and at some point, full-resolution data is needed to get an acceptable results. Fortunately, as Section 5.2 shows, a shorter integration length is often a more appropriate choice once we fix an optimal iso-value anyway. The results in Fig 9 are produced with an integration length of roughly 0.02 mm.

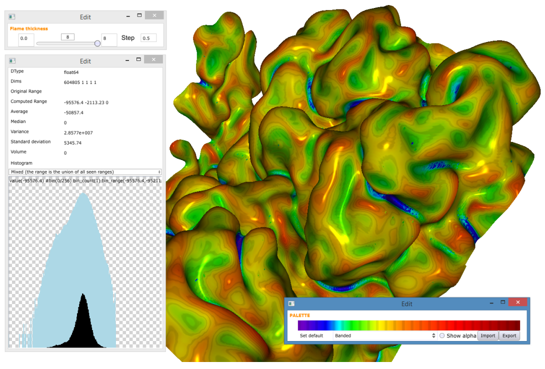
5.5 Timings

Here we provide timing information of each step in the analysis using our system. All benchmarks run on a laptop with an Intel Core i7-3610QM 2.3 GHz processor, 8 GB of DDR3 memory, and an NVIDIA GeForce GTX 660M graphics card. The laptop's hard drive has an average transfer rate of 100 MB/s. Both our client and server software are in the same LAN network, with an average bandwidth of 40 MB/s. The native resolution of our dataset is 512x256x256 voxels. For this analysis, we extract two scalar fields stored in 64-bit floating point format, for a total of 512 MB per time step. The whole dataset has 34 fields and 660 time steps in total, for a total size of 5.7 TB. Table 1 shows the detailed timings.

	Full (N)	Full (C)	Half (N)	Half (C)
Query	21	13	3.4	0.3
Iso-surface	0.6	0.6	0.08	0.08
Integration	0.9	0.9	0.2	0.2
Render	0.5	0.5	0.1	0.1

Table 1: Timings of each step. All numbers are in seconds. Full (N) means full resolution data from network. Half (C) means half-resolution data (one-eighth the full data in bytes) from cache.

As can be seen from Table 1, iso-surface extraction, integration, and rendering are relatively fast. Data transfer (Query) is the bottle neck of the system, either through network, or from cache to memory. This



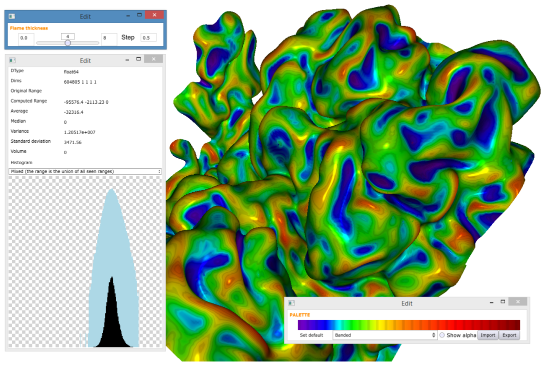
Flame thickness: 0.0 | 8 | 8 | Step 0.5

DType	float64
Dims	604805 1 1 1 1
Original Range	
Computed Range	-95576.4 -2113.23 0
Average	-50857.4
Median	0
Variance	2.8577e+007
Standard deviation	5345.74
Volume	0

Mixed (the range is the union of all seen ranges)

value(-95576.4) #bin(0/256) bin_count(1) bin_range(-95576.4,-95211

(a) Integration length = 8 voxels (≈ 0.3 mm)

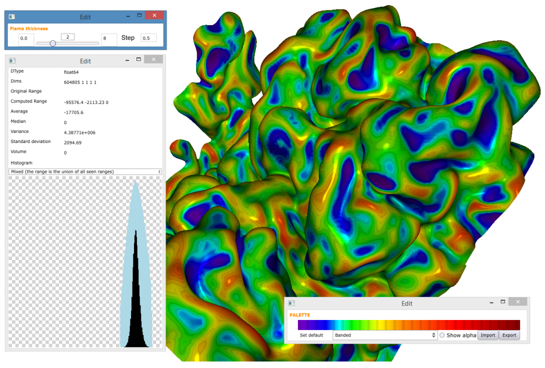


Flame thickness: 0.0 | 4 | 8 | Step 0.5

DType	float64
Dims	604805 1 1 1 1
Original Range	
Computed Range	-95576.4 -2113.23 0
Average	-32316.4
Median	0
Variance	1.20517e+007
Standard deviation	3471.56
Volume	0

Mixed (the range is the union of all seen ranges)

(b) Integration length = 4 voxels (≈ 0.15 mm)

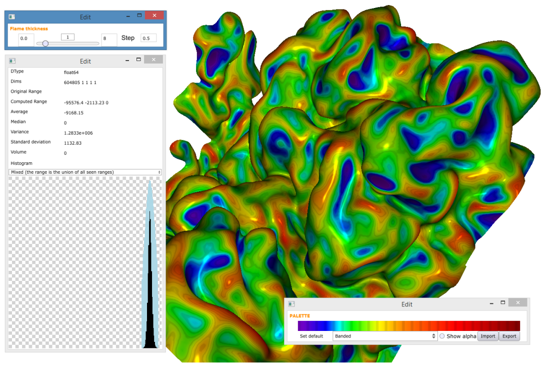


Flame thickness: 0.0 | 2 | 8 | Step 0.5

DType	float64
Dims	604805 1 1 1 1
Original Range	
Computed Range	-95576.4 -2113.23 0
Average	-17705.6
Median	0
Variance	4.38771e+006
Standard deviation	2094.69
Volume	0

Mixed (the range is the union of all seen ranges)

(c) Integration length = 2 voxels (≈ 0.08 mm)



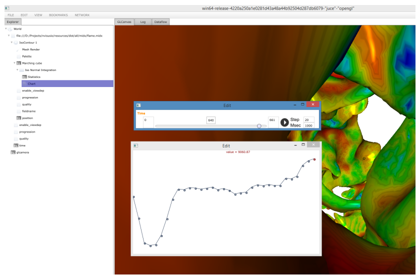
Flame thickness: 0.0 | 1 | 8 | Step 0.5

DType	float64
Dims	604805 1 1 1 1
Original Range	
Computed Range	-95576.4 -2113.23 0
Average	-9168.15
Median	0
Variance	1.2833e+006
Standard deviation	1132.83
Volume	0

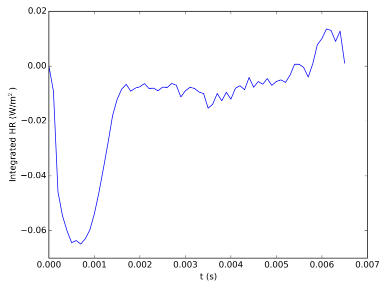
Mixed (the range is the union of all seen ranges)

(d) Integration length = 1 voxels (≈ 0.04 mm)

Fig. 7: Effects of different integration lengths (at time step 391) . We fix an iso-value of 0.19.



(a) A graph of mean IHRR values drawn interactively as the user drags the time slider at the rate of every 20 time steps, using one-eighth the resolution of the original data.



(b) The same graph drawn offline using a Python script, with full resolution data and at the rate of every 10 time steps.

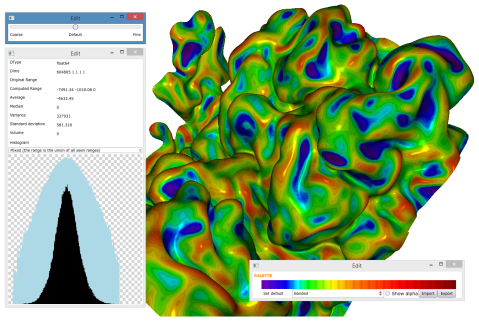
Fig. 8: The graph of mean IHRR values plotted through time. This mean energy consumption behavior is consistent with the physical flame response, particularly its stable positioning in the total domain volume.

implies that even with caching, using low-resolution data is required to achieve interactivity. However, note that the Query node only activates when there is new data to process, such as when the user drags the time slider. Once the data for a time step is transferred, for analysis that does not require changing the time step, the data querying time does not factor in anymore, and the system becomes fully interactive even with full-resolution data.

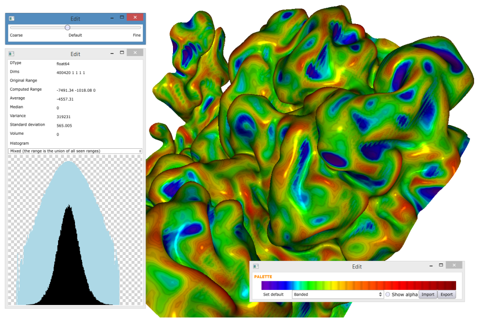
Assuming the user wants to do analysis over time (for example, plotting a graph like the one shown in Figure 8), and thus must use low-resolution data to achieve interactivity, the trade-off is greater errors in the integration results. But as we have shown in previous sections, these errors are quite tolerable up until one-eighth the original data, especially for quick exploration purposes. Scientists can use low resolution data to experiment with different input parameters and get feedback quickly. Once the optimal parameter values have been found, a more expensive analysis can be done with high-resolution data. Even then, our system only takes about half a minute to process each time step.

6 Conclusions

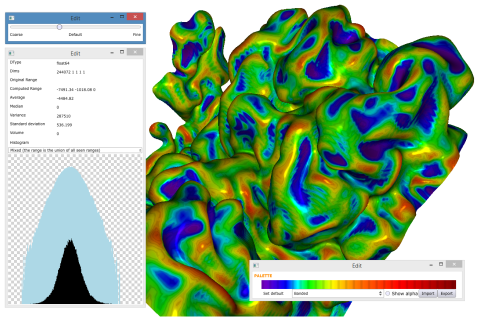
We present a new interactive analysis and visualization system to support large-scale combustion research. Through the coupling of highly efficient data accesses and a streaming, progressive dataflow our application provides unprecedented capabilities to our collaborators. In particular, we have demonstrated the analysis of a statistically premixed planar flame of lean hydrogen-air with well resolved spatial and temporal scales, allowing for detailed analysis of turbulence-chemistry interactions. The system provides both an on-the-fly analysis of the integrated heat release rate for all time steps as well as a local representation of the heat release rate through a pseudo-colored iso-surface which provides new insights about the connection between the local surface geometry and the level of reactivity. Our system enables our collaborators to quickly explore the space of input parameters, observe



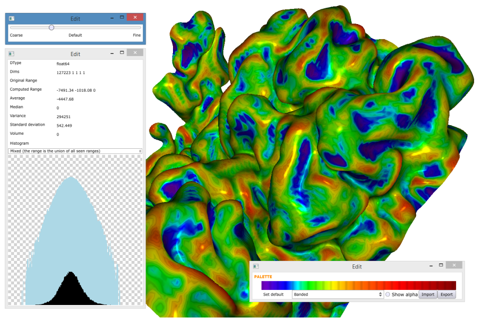
(a) Quality level 0 (finest resolution)



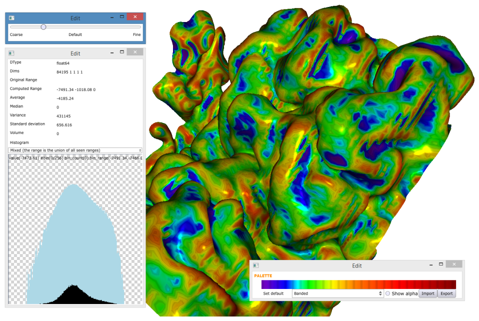
(b) Quality level 1 (half of the data)



(c) Quality level 2 (one-fourth of the data)



(d) Quality level 3 (one-eighth of the data)



(e) Quality level 4 (one-sixteenth of the data)

Fig. 9: Effects of different resolutions levels (at time step 391)

how the results evolve and refine their initial estimates of these parameters. This is a crucial step forward as it allows scientists to interactively explore previously inaccessible analysis techniques.

References

[1] P. Arias, H. Im, P. Narayanan, and A. Trouvé. A computational study of non-premixed flame extinction by water spray. *Proceedings of the Combustion Institute*, 33(2):2591 – 2597, 2011.

[2] U. Ayachit. The ParaView Guide: A Parallel Visualization Application. Kitware Inc., Clifton Park, NY, 2015.

[3] J. H. Chen and H. G. Im. Correlation of flame speed with stretch in turbulent premixed methane/air flames. *Proceedings of the Combustion Institute*, 27:819–826, 1998.

[4] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012.

[5] M. Day, S. Tachibana, J. Bell, M. Lijewski, V. Beckner, and R. K. Cheng. A combined computational and experimental characterization of lean premixed turbulent low swirl laboratory flames. II. Hydrogen flames. *Combustion and Flame*, 2015. In Press. Corrected Proof.

[6] T. Echekki and J. H. Chen. Unsteady strain rate and curvature effects in turbulent premixed methane-air flames. *Combustion and Flame*, 106:184 – 202, 1996.

[7] H. G. Im and J. H. Chen. Effects of flow transients on the burning velocity of laminar hydrogen/air premixed flames. *Proceedings of the Combustion Institute*, 28:1833–1840, 2000.

[8] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink. Large mesh simplification using processing sequences. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 61–, Washington, DC, USA, 2003. IEEE Computer Society.

[9] S. Kumar, V. Vishwanath, P. Carns, J. Levine, R. Latham, G. Scorzelli, H. Kolla, R. Grout, R. Ross, M. Papka, J. Chen, and V. Pascucci. Efficient data restructuring and aggregation for I/O acceleration in PIDX. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 50:1–50:11. IEEE Computer Society Press, 2012.

[10] S. Kumar, V. Vishwanath, P. Carns, B. Summa, G. Scorzelli, V. Pascucci, R. Ross, J. Chen, H. Kolla, and R. Grout. PIDX: Efficient parallel I/O for multi-resolution multi-dimensional scientific datasets. In *Proceedings of The IEEE International Conference on Cluster Computing*, pages 103–111, September 2011.

[11] L. Lo, C. Sewell, and J. P. Ahrens. PISTON: A portable cross-platform framework for data-parallel visualization operators. In *Eurographics Symposium on Parallel Graphics and Visualization, EGPGV 2012, Cagliari, Italy, May 13-14, 2012. Proceedings*, pages 11–20, 2012.

[12] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM.

[13] MATLAB. *version 8.5 (R2015a)*. The MathWorks Inc., Natick, Massachusetts, 2015.

[14] J. S. Meredith, S. Ahern, D. Pugmire, and R. Sisneros. EAVL: The Extreme-scale Analysis and Visualization Library. In H. Childs, T. Kuhlen, and F. Marton, editors, *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2012.

[15] K. Moreland, U. Ayachit, B. Geveci, and K.-L. Ma. Dax toolkit: A proposed framework for data analysis and visualization at extreme scale. In D. Rogers and C. T. Silva, editors, *LDAV*, pages 97–104. IEEE, 2011.

[16] V. Pascucci and R. J. Frank. Global static indexing for real-time exploration of very large regular grids. In *Proceedings of Supercomputing 2001*. ACM, 2001.

[17] V. Pascucci, G. Scorzelli, B. Summa, P.-T. Bremer, A. Gyulassy, C. Christensen, S. Philip, and S. Kumar. *The ViSUS Visualization Framework*, chapter 19, pages 401–414. Chapman \& Hall/CRC Computational Science, 2012.

[18] W. J. Schroeder, L. S. Avila, and W. Hoffman. Visualizing with vtk: A tutorial. *IEEE Comput. Graph. Appl.*, 20(5):20–27, Sept. 2000.

[19] H. T. Vo, D. K. Osmari, J. Comba, P. Lindstrom, and C. T. Silva. Hyperflow: A heterogeneous dataflow architecture. In *Eurographics Symposium on Parallel Graphics and Visualization, EGPGV 2012, Cagliari, Italy, May 13-14, 2012. Proceedings*, pages 1–10, 2012.