

Reduction of Steady-State CFD HVAC Simulations into a Fully Transient Lumped Parameter Network

Robert E Smith
US Army TARDEC

Edward Lumsdaine
Michigan Technological University

ABSTRACT

Since transient vehicle HVAC computational fluids (CFD) simulations take too long to solve in a production environment, the goal of this project is to automatically create a lumped-parameter flow network from a steady-state CFD that solves nearly instantaneously. The data mining algorithm k-means is implemented to automatically discover flow features and form the network (a reduced order model). The lumped-parameter network is implemented in the commercial thermal solver MuSES to then run as a fully transient simulation. Using this network a "localized heat transfer coefficient" is shown to be an improvement over existing techniques. Also, it was found that the use of the clustering created a new flow visualization technique. Finally, fixing clusters near equipment newly demonstrates a capability to track localized temperatures near specific objects (such as equipment in vehicles).

CITATION: Smith, R. and Lumsdaine, E., "Reduction of Steady-State CFD HVAC Simulations into a Fully Transient Lumped Parameter Network," *SAE Int. J. Passeng. Cars - Mech. Syst.* 7(1):2014, doi:10.4271/2014-01-9121.

INTRODUCTION

Vehicle cabin temperature CFD simulations are extremely important to Army ground systems to assure the functionality of sensitive electronics which are frequently added to vehicles. Sizing an HVAC system for operations in desert environments is entirely driven by the need to cool a heat-soaked vehicle to operational condition within 15 minutes. The definition of "operational condition" is the requirement to cool the air around equipment below its temperature threshold within a specified time limit - the ability for the equipment to function takes precedence over crew comfort. However, if the system has the cooling capacity, the objective requirement is to additionally keep the crew air temperature below 85°F when possible.

Full cabin-cooldown CFD simulations take great computational effort. Transient simulations run 15-30 minutes in simulated duration and must use no larger than 30 second time steps. Simulations using a commercial CFD solver with a 15-20 million volume cell mesh (polyhedral elements), require at least two weeks of physical time to solve on a large compute cluster. The interior forced flow and geometry details are complex and require turbulence modeling and occasionally buoyancy. Even when the number of processors available is unlimited, commercial CFD code scalability is limited by the communications speed between the processors.

The airflow in a transient HVAC simulation is generally stable and fairly constant and only the temperatures vary. Surface temperatures change very slowly. This is referred to as "quasi-steady-state" condition for the thermal solution, where the flow is at a steady state condition since it is on a different time scale than surface temperatures. The quasi-steady-state nature of these simulations may be exploited to solve a lumped-parameter flow network where only the air temperatures change to obtain a transient HVAC simulation.

Subvoluming Method

The current state-of-the-art technique to accelerate the solution of transient cool-down simulations of the crew area is subvoluming [1]. Subvoluming simplifies a steady-state CFD solution into multiple lumped control volumes that communicate by advection. The process simply subdivides a solved steady-state CFD solution into regular sized Cartesian partitions (Figure 1) and tracks the advection across these planes. The process is volume based and does not conserve mass.

The subvoluming process starts by manually coupling a steady-state CFD model to a MuSES model. The coupling process is an existing feature where convection coefficients and fluid film temperatures are exported from StarCCM+ and read into MuSES. MuSES solves the conjugate heat transfer

and then exports the wall temperatures back into StarCCM+. This process is usually repeated 5-6 times [2], but may take up to 20-30 times.

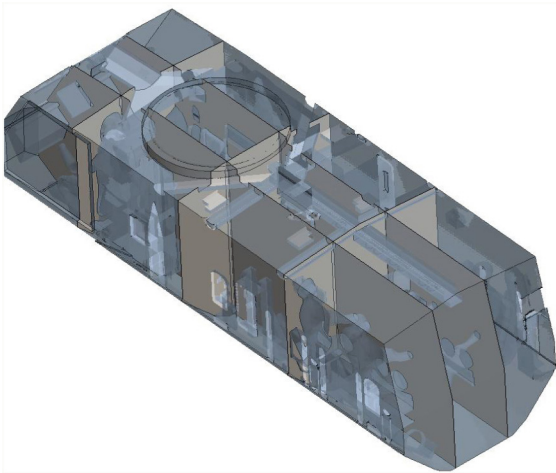


Figure 1. Example of decomposing an MRAP CFD domain using subvoluming. Each box represents a lumped control volume and flow is tracked at the interfaces. Note the Cartesian break-up of the domain.

MuSES is primarily a shell element finite-difference thermal solver, which allows for very fast conduction and radiation calculations, but MuSES lacks a method to directly solve convection. MuSES is an important part of vehicle cooldown simulations because it adds the capability of including detailed environmental modeling by coupling to a CFD model. MuSES is also sold as Radtherm commercially, where Radtherm does not incorporate an infrared signature prediction capability as does MuSES.

The next step in subvoluming, after obtaining a coupled steady-state MuSES/CFD solution is to calculate the fluid transport between subvolumes (advection). To do this, the fluid advection is calculated by summing the volume flux across dividing planes that form the new lumped volumes. The authors [1] then assume based on past experience that all convection from the walls to the subvolumes is $15 \text{ W/m}^2\text{-K}$. At this point a lumped-parameter flow model may be created which solves for all fluid parameters. The conjugate heat transfer simulation may now be run entirely within MuSES. This technique was also used in [3]. Additionally, subvoluming also appears in [4,5].

There are a number of deficiencies in using the subvoluming technique, but the main one is the fact that it discards available information from the CFD solution because of indiscriminate lumping into a Cartesian network. Additionally, the near-wall convection data from the CFD solution could be recalculated in reference to the average temperature of the nearest subvolume (preserving the wall heat flux), but the existing process simply assumes a convection of $15 \text{ W/m}^2\text{-K}$ which is a typical average value for interior HVAC simulations. Lastly, this method is based on volume flow and does not conserve mass.

NEW “CFDMINE” PROCESS

A new process, called “cfdMine” is shown in Figure 2. Like subvoluming, the process starts with a steady-state MuSES model (which predicts wall temperatures) that is manually coupled to a steady-state CFD model (which predicts convection). Once the manual coupling has converged, the clustering algorithm is run to group the flow domain into clusters that are similar in flow direction, geometric location, and temperature. The clustering result is visually inspected by exporting to a post processor via the Enight Gold format. Clustering parameters are iteratively adjusted until a satisfactory clustering has been achieved.

Once an acceptable clustering scheme has been achieved, an agglomeration algorithm is used to make sure all elements grouped by the clustering are touching each other (discussed in detail in subsequent sections). Next, the mass flux between clusters is computed by finding shared volume element faces which have different clusters on each side of the face. The fluxes are summed and tracked. Finally, the user may set up the simplified control volume network in the steady-state MuSES model which is done using “fluid nodes” and advection links. The association between the wall elements and the fluid nodes is created using “thermal links” and remaps the convection coefficients in MuSES from the fluid film based convection to a localized convection based on the fluid node temperature. Finally a fully transient MuSES model is run.

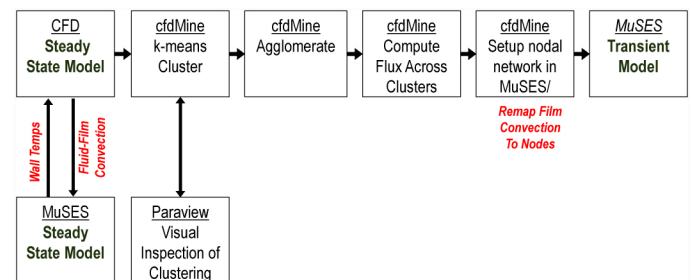


Figure 2. New cfdMine process to use a steady-state CFD solution to create a transient MuSES thermal simulation.

Simple Clustering Test Case

In order to explore the feasibility of using the k-means algorithm for clustering, a simple impinging jet test case was developed as shown Figure 3. Figure 4, Figure 5, Figure 6 show examples of visualizing the clustering results for “Case 1”. Figure 6 shows the flow between the clusters as lines.

An unintended discovery of the cfdMine method is the enhanced visualization of flows. The simplified nodal network shown in Figure 6 provides a useful new way to visualize the flow domain. Normally analysts use particle traces or planar cuts to cull through the CFD domain to understand flow patterns. The newly developed method represents each node as a sphere. The size of the sphere represents the volume of enclosed fluid and connecting lines between nodes represent fluid advection. The thickness or color of the lines would represent the magnitude of fluid flux. The lines and colors

could be also be colored by fluid temperature or velocity. Only the dominant flow features show up in Figure 6 and allow a quick understanding of dominant patterns.

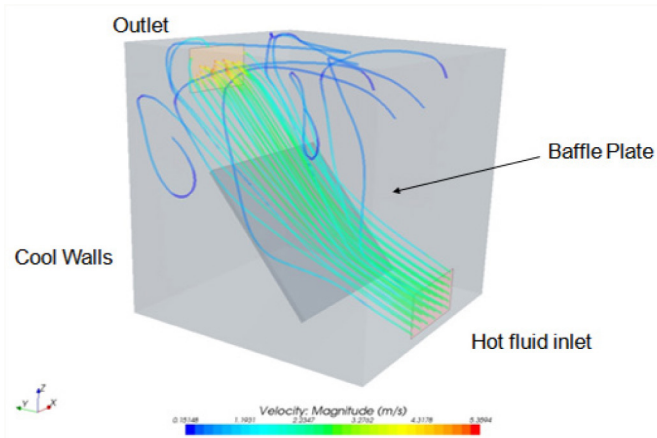


Figure 3. Test Case 1. This case consists of a hot fluid impinging on a baffle and exiting from an outlet slot. The walls of the cube are held at a constant temperature.

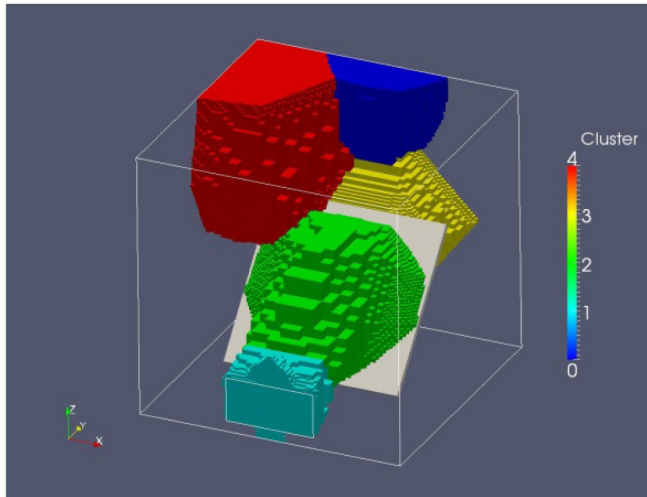


Figure 4. Clustering results for Case 1. Shown are clusters 0-4 of 15 total clusters.

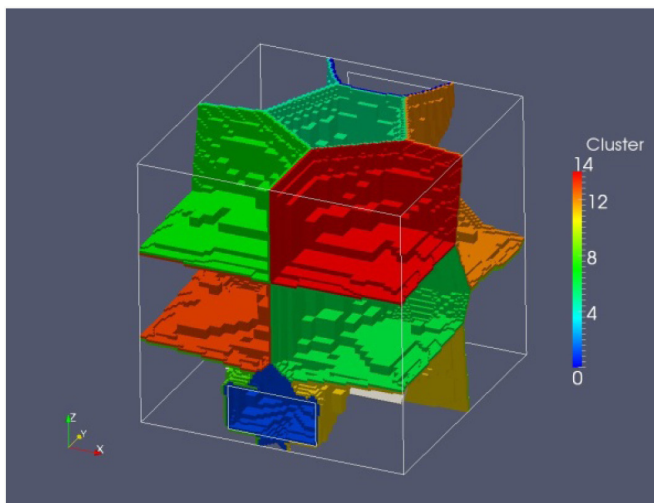


Figure 5. Clustering interfaces. This plot shows only connected elements between cluster borders. Elements are colored still by cluster numbers.

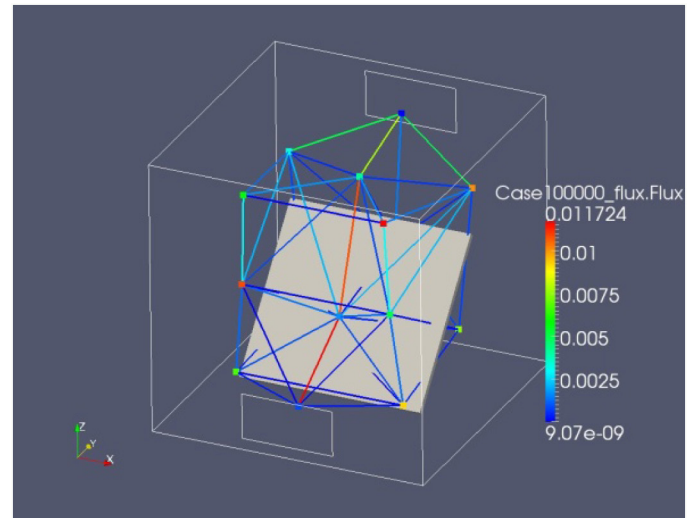


Figure 6. All 15 cluster centers are shown as square dots. The fluxes between the clusters are shown as lines colored by flux intensity.

K-Means Clustering

The cfdMine methodology uses k-means clustering to group the CFD domain automatically into regions of mostly homogeneous temperature, flow direction, and spatial proximity. Jain defines clustering as the unsupervised classification of patterns, observations, data items, or feature vectors into groups or clusters [6]. Cormack states clusters should be externally isolated and internally cohesive, implying homogeneity within clusters and heterogeneity between clusters [7]. The notion that clustering is unsupervised means that there is no predetermined answer to which the machine learning algorithm might train.

After reviewing the literature and some initial experimentation, it was decided the k-means clustering had the most potential to automatically generate a reduced order model in terms of performance, computational intensity, and also suitability to fixing nodes and for visualization. K-means was first suggested in [8] and independently in [9]. A detailed history and full discussion of various deviations may be found in "K-Means Clustering: A Half-Century Synthesis" [10].

K-means attempts to find the minimal sum of the squares of distances, as in Equation 1. The algorithm seeks to minimize this performance function by recursively assigning membership of points to the nearest cluster. Then a new average cluster value (a multidimensional centroid) is computed. This iteration is embodied in Equation 2:

K-Means Performance Function:

$$\text{Sum Squared Error} = \sum_{l=1}^K \sum_{x \in S_l} ||\vec{x} - m_l||^2 \quad (1)$$

K-Means Iterative Function:

$$\bar{m}_l = \frac{1}{n_l} \sum_{x \in S_l} \vec{x} \quad (2)$$

Similarity Measures

All clustering processes utilize some form of similarity measures, which provide a metric defining the similarities between data based on the features of interest. Similarity measures may be thought of as the sum of all distances between data points and cluster properties, or a global error, "E". Distance metrics are the pairwise measurements between points or a defining point (such as the centroid in k-means). Usually data is preprocessed via normalization prior to computing the similarity. By far, the most popular similarity measure is the Euclidean distance (Equation 3), which measures the hyperspherical distance in multidimensions. For higher order data sets, these are not physical distances. For example if temperature is a feature of interest, difference in temperature may be used as part of a Euclidean distance space. Similarity scores may be combined by weighting them and summing. CfdMine uses a weighted Euclidean similarity metric that measures spatial distance (x,y,z distance to the cluster centroid), temperatures distance, and the flow direction cosine distance.

Examples of the most common similarity metrics are:

Euclidean Distance

$$d_e = \left(\sum_{k=1}^d (x_{i,k} - x_{j,k})^2 \right)^{\frac{1}{2}} \quad (3)$$

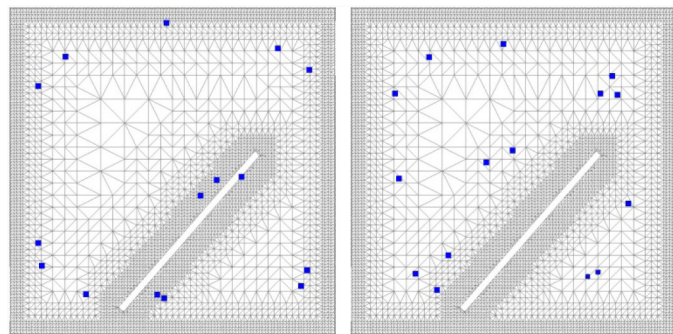


Figure 7. The image at the left shows a side view of test case 1 clusters without mass-weighting. Notice that the density of cells is high near boundaries and regions of high gradient. The image at the right shows clustering results when mass weighting is enabled.

Minkowski Distance:

$$d_m = \left(\sum_{k=1}^d (x_{i,k} - x_{j,k})^p \right)^{\frac{1}{p}} \quad (4)$$

Cosine Similarity (dot product):

$$d_e = \left(\sum_{k=1}^d x_{i,k} \cdot x_{j,k} \right) \quad (5)$$

Mass Weighting

Mass weighting is a necessity for clustering CFD data since the computational grid does not consist of a randomly distributed sample of points. Without weighting, cluster centers are attracted to areas where the mesh is more dense (walls for example). Mass or volume weighting allows the clusters to form with no mesh dependencies.

Changing the k-means recursive function to a mass-weighted one is quite easy:

K-Means Mass-Weighted Iterative Function

$$\bar{m}_l = \frac{\sum_{x \in S_l} m_x}{n_l} \sum_{x \in S_l} \vec{x} \cdot m_x \quad (6)$$

To demonstrate the need for mass weighting more explicitly, Test Case 1 was run weighted and unweighted (Figure 7). A cutting plane through the center of the domain was taken and it is easy to see from a side view that the cluster centers are attracted to regions of high density simply because there are more elements in those areas. While this paper assumes having more clusters where mesh density is highest to be a negative feature, further work should be performed to investigate the validity of this assumption.

Data Normalization

It is recommended that the data be normalized across the spatial and temperature domains. The two most popular normalization methods are z-scores and extents. Milligan and Cooper [11] found that it is better to standardize on extents rather than z-scores. The method of normalizing based on extents is very simple: After all the distances are computed for each point, each parameter is divided by the global maximum-global minimum (or the spread) for each parameter so that each has a range of zero and one. It was found that the x,y,z data should be divided by their averaged combined standard deviation so that distance in those directions would be equally penalized. If this is not done, a domain with a narrow z domain and long x and y domains will create clusters that look