2012

Computational fluids domain reduction to a simplified fluid network

Robert E. Smith
Michigan Technological University

COMPUTATIONAL FLUIDS DOMAIN REDUCTION TO A SIMPLIFIED FLUID NETWORK

By

Robert E Smith

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

(Mechanical Engineering-Engineering Mechanics)

MICHIGAN TECHNOLOGICAL UNIVERSITY

2012

THIS PAGE INTENTIONALLY LEFT BLANK

This dissertation, “Computational Fluids Domain Reduction to a Simplified Fluid Network,” is hereby approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING-ENGINEERING MECHANICS.

Department of Mechanical Engineering-Engineering Mechanics

Signatures:

Dissertation Advisor _____

Edward Lumsdaine

Department Chair _____

William W. Predebon

Date _____

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

List of Tables

Disclaimer

Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA, and shall not be used for advertising or product endorsement purposes.

Abstract

The primary goal of this project is to demonstrate the practical use of data mining algorithms to cluster a solved steady-state computational fluids simulation (CFD) flow domain into a simplified lumped-parameter network. A commercial-quality code, “cfdMine” was created using a volume-weighted k-means clustering that that can accomplish the clustering of a 20 million cell CFD domain on a single CPU in several hours or less. Additionally agglomeration and k-means Mahalanobis were added as optional post-processing steps to further enhance the separation of the clusters. The resultant nodal network is considered a reduced-order model and can be solved transiently at a very minimal computational cost. The reduced order network is then instantiated in the commercial thermal solver MuSES to perform transient conjugate heat transfer using convection predicted using a lumped network (based on steady-state CFD). When inserting the lumped nodal network into a MuSES model, the potential for developing 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 temperatures near specific objects (such as equipment in vehicles).

Introduction

The present research can be divided into two main thrust areas: The first thrust area is developing a new overall process for generating and solving a transient reduced-order lumped network from steady-state CFD simulation. The new network is then instantiated as a one-dimensional fluid network in the commercial solver MuSES. The second thrust area is to explore a number of clustering techniques that generate the lumped control volumes for the network efficiently and optimally. Like many things, the basic concepts are fairly straightforward, but a lot of interesting nuances surfaced in implementation of the concept.

Vehicle cabin temperature CFD simulations are extremely important for Army ground systems since there is a strong push to add extensive sensitive electronics devices to vehicles. Sizing an HVAC system for current 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 objective requirement is to additionally keep the crew air temperature below 85°F when possible. This contrasts to the commercial automotive industry where the focus is above all, passenger comfort. Passenger comfort must additionally include consideration of dry bulb temperature, relative humidity, and air velocity (2011a). The main objective of this research is to create a new equipment cooldown simulation capability for the US Army and the focus is not on human comfort or simulation of the HVAC system itself. However, this technique may be fully utilized for passenger comfort as well.

Full cabin-cooldown CFD simulations take great computational effort. These transient simulations run 15-30 minutes in time duration and must use no larger than 30 second time steps. Using a commercial CFD solver with a 15-20 million volume cell mesh, these types of simulations require at least two weeks to solve on a large compute cluster. Even when the number of processors available is unlimited, commercial CFD codes are limited by the communications speed between the processors. While running a transient CFD simulation accounts for changing solar and surface temperatures in the model, it is not intended to capture transient flow nuances. The airflow itself is generally stable and fairly constant and only the component surface temperatures vary. 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.

CFD domain sizes at TARDEC are generally on the order of 20 million volume cells for a typical problem. Solution time for a steady-state simulation of 20 million cells is usually overnight, but would run a week or two for a transient run which is clearly unacceptable. Historically, engineers

tend to size the detail of their models to allow overnight runs and the transient case simply is too slow to be practical. It is the goal of this research to propose a new method which will exploit the quasi-steady-state nature of the vehicle cooldown problem to create a simple fluid network which will solve very rapidly inside of the commercial solver MuSES.

MuSES Overview

MuSES software is a critical piece of the present proposal. MuSES originated as a cooperative research project (CRADA) between Ford, the US Army Tank Automotive Research and Development and Engineering Command (TARDEC), and Michigan Technological University. MuSES is maintained and sold by Thermoanalytics, Inc. and is a standard part of the thermal modeling process at TARDEC. MuSES stands for Multi-Service Electro-optic Signature, but Thermoanalytics sells versions of sans the infrared signature capability under the names Radtherm and Wintherm (2012a). MuSES is primarily a shell element finite-difference thermal solver that provides the following features (2011a):

- Automatic multibounce radiation and view factor calculation
- Multilayer shell-element conduction
- Wind convection model and weather inputs
- Solar loading, including consideration of transparent materials such as glass
- Lumped network solver
- Infrared signature solver

The reason MuSES is an important part of vehicle cooldown simulations is that it adds the capability of including detailed environmental modeling. Most commercial CFD solvers do not have the ability to incorporate weather conditions and a surrounding faceted terrain as a boundary condition. Additionally the fact MuSES is primarily a shell-element-based solver is a very good assumption for most military and automotive problems. Shell elements are much less computationally expensive than running full solid-elements simulations for walls. Shell element models tend to require a lower mesh density since surfaces lack the intricate details of a flow regime. This speeds up the solution time dramatically, but clearly accurate convection data is important which is lacking in MuSES as standalone software. MuSES is particularly useful for the present research since it also provides an internal lumped network solver which will be utilized to incorporate the results of data mining into the conjugate thermal solution.

Literature Review

There are essentially two major categories of topics addressed in this review. The first is methods that address the problems associated with running fully transient vehicle heat-up and cool-down simulations. The second and more extensive section is specific to data mining and clustering algorithms.

CFD Coupling Literature Review

An approach used quite commonly in industry for passenger compartment transient simulations is to utilize MuSES to perform the transient simulation by extrapolating from a steady-state CFD solution (Curran et al. 2010). This is typically done by importing convection coefficients and fluid film temperatures from a CFD model in a manually coupled manner. A steady-state CFD simulation is converged with assumed temperature boundaries and then the convection data is exported to MuSES. MuSES then computes wall temperatures from the imported convection and adds radiation and conduction. Then MuSES exports new wall boundaries which are imported into the CFD code. Generally it takes approximately six iterations to achieve a convergence between the codes (based on firsthand experience).

Most major CFD codes have been coupled to MuSES by passing data using packet 17 in a modified PATRAN Neutral file format. The mesh density of a Muses model is much lower than a CFD model since a CFD code needs to resolve phenomenon at the boundary. The data from the CFD model must be mapped to the lower resolution MuSES model. MuSES accomplishes this via a ray tracer which casts a ray from the MuSES mesh and assigns the convection data based on the first CFD mesh element encountered (2011a). The data passed is a convection coefficient and fluid film temperature. The temperature of the fluid element nearest the wall is considered the fluid film temperature, so the temperature of the fluid is in the nearfield of the convective exchange.

The coupling process is very loose and the solvers do not talk during computation. The commercial CFD code provides MuSES with the cell temperature nearest the wall and a convection coefficient; MuSES provides the CFD code with wall temperatures. (Haupt et al. 2010) provides an example of this process for building temperature prediction. The CFD code NPHASE-PSU is only updated at the following times over 24 hours: 00:00, 08:00, 10:00, 13:00, 16:00, and 18: 00. For each of those times the authors iterate between MuSES and the CFD code until wall temperatures change less than 1C at that timestep. The transient MuSES timesteps are much smaller, although the authors do not mention the specifics. That means convection parameters are assumed to be constant between, for example, 08:00 and 10:00.

The problem with this type of process is that it really doesn't work efficiently with a loose coupling. The convection coefficient passed from CFD to MuSES is based on a fluid film temperature within the boundary layer. Since the difference between the wall temperature and the fluid film is very small, a large convection coefficient is used to obtain the correct heat flux. Small changes in the wall temperature in MuSES tend to overpredict the change in heat flux and this is only corrected by extensively iterating back-and-forth with the loosely-coupled CFD code. Occasionally this causes observable problems where the two codes don't converge relative to each other.

A better method would be to assume a localized reference temperature which would result in smaller convection coefficients and decrease the amount of iterating between codes.

Lumped System Modeling

Full transient models are computationally too expensive to run directly. The simplest work-around is to treat the cabin area as one lumped control. Additionally, this is useful to simulate the refrigeration components as in the work performed by (Huang et al. 1999). The Huang model considers the capacitance and conductivity of surface elements representing a vehicle mass. Additionally the model tracks humidity, solar loading through the windows, passenger heat, and moisture loads on the interior air. The interior air is considered to be one homogeneously mixed control volume. This type of simulation is well suited to the design of the HVAC components, but is not optimal for the tracking of localized temperatures within the cabin area. It would not, for example, be able to predict the localized temperatures around crewmembers and 30 electronics boxes.

Subvoluming

The state-of-the-art technique to solve transient cool-down simulations of the crew area is subvoluming and was first published by (Curran et al. 2010). Subvoluming surpasses the aforementioned lumped technique by simplifying steady-state CFD solutions into multiple lumped control volumes that communicate by advection. The process simply subdivides a solved steady-state CFD solution into regular sized partitions. Figure 1 shows a typical ground vehicle cabin CFD and Figure 2 shows how cutting planes partition the domain.

The fluid transport between subvolumes (nodes) is calculated by summing the volume flux across dividing planes that form the new lumped volumes. The method described by (Curran et al. 2010) then uses commercial MuSES software to actually solve the subvolume network, solar loading, radiation, and surface temperatures. The authors then assume all convection from the walls to the subvolumes is 15 w/m2-k. This technique was also used in (Hepokoski et al. 2010). Additionally, subvoluming also appears in (Han et al. 2010) and (Shah et al. 2008).

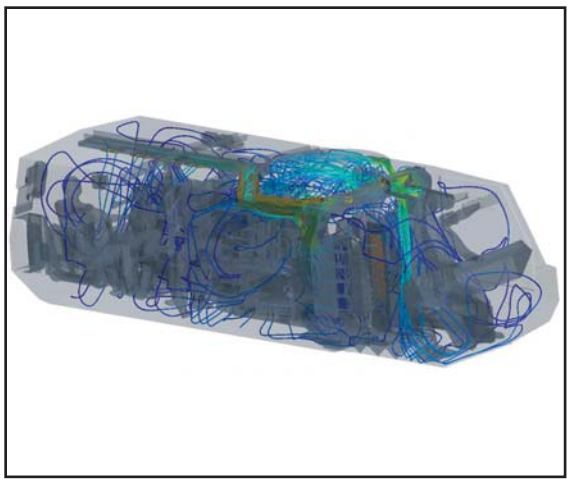

Figure 1: Example CFD of an MRAP.

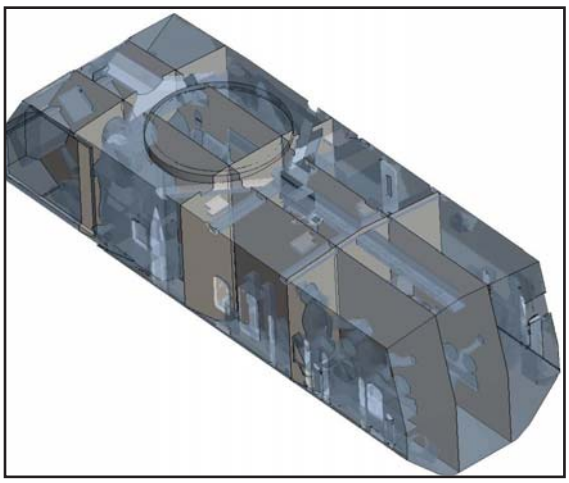

Figure 2: Example of decomposing a domain into a reduced-order network. Each box represents a lumped control volume and flow is tracked at the interfaces. Note the arbitrary break-up of the domain.

There are a number of deficiencies in using this subvoluming technique, but the main one is the fact that it discards available information from the CFD solution because of indiscriminate lumping. Additionally, the near-wall convection data from the CFD solution could be recalculated in reference to the average temperature of the nearest subvolume, but the existing process simply assumes a convection of 15 w/m2-k based on “experience”. The detailed CFD model has fluid film temperatures and a convection coefficient at the cell nearest the wall which is really a heat flux that could be remapped to the subvolume temperature and a new convection coefficient. There is no known validation study of the subvoluming approach.

Clustering Literature Review

Clustering is proposed as an alternative to subvoluming. It is proposed that clustering may be used to optimally group volume elements within a solved steady-state CFD solution into lumped homogeneous control volumes. Subvoluming does not find homogeneous control volumes but blindly subdivides the domain. It is hoped that the clusters will better preserve the flow structure within the CFD domain. Additionally, if those clusters are fixed around mission equipment in the vehicle, localized equipment temperatures may be predicted.

Jain defines clustering as the unsupervised classification of patterns observations data items or feature vectors into groups or clusters (Jain et al. 1999). Cormack states clusters should be externally isolated and internally cohesive, implying homogeneity within clusters and

heterogeneity between clusters (Cormack 1971). The notion that clustering is unsupervised means that there is no predetermined answer to which the machine learning algorithm might train. There is also a common term, partially supervised learning, where the user inputs some data, but mostly the classifications are unknown.

Clustering is widely used. Examples of places it is used are pattern recognition, image processing, market research, and even document classification for web searching and cataloging. Spatial data analysis is another less common use of clustering techniques. For example, terrains may be categorized from satellite imagery into types of soils. An insurance company might use data mining to identify demographics groups of insurance policy holders with high average claim costs.

The primary two categories for data clustering are hierarchical and partitional. Hierarchical clustering is based on the notion that objects are more related to nearby objects than to objects farther away. Hierarchical clustering can be computed bottom-up or top-down. Hierarchical agglomerative clustering is starting with single elements and aggregating them into clusters. Hierarchical divisive clustering is starting with the complete data set and dividing it into partitions. Partitional clustering operates on data globally and is exemplified by k-means where given an initial set of “k” means, points are assigned to the most similar “k” cluster. K-means then recomputes the cluster means and iterates. K-means separates data into equal-sized clusters or equal “inertia”, also known as Voronoi-cells (illustrated in Figure 3 and Figure 4). Centroid-based clustering optimization is ultimately considered an np-hard problem, but is possible to solve for most practical cases.

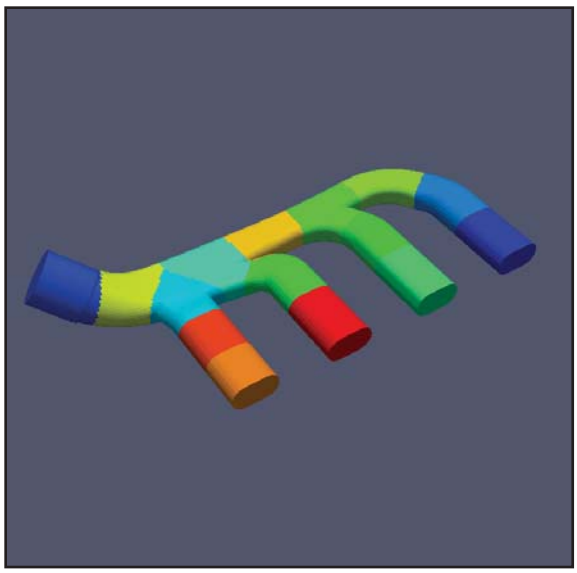

Figure 3: K-means run on a manifold using pure Euclidean distance divides the domain into equal centroid regions.

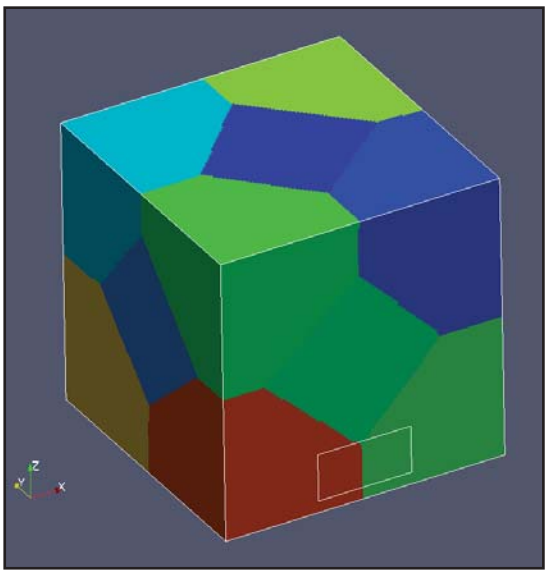

Figure 4: K-means run on a box using pure Euclidean distance divides the domain into equal centroid regions.

A hierarchical clustering algorithm yields a tree structure, or a dendrogram. The dendrogram represents a nested grouping of patterns where each branching represents a subdivision of the groupings. The dendogram can be formed from the “bottom up” or “top down”. The bottom up approach is called agglomeration. Agglomeration starts with each object in a separate group. It successively merges the groups according to some distance measure (usually between the centers of two groups). This is done until all groups are merged into one top level. The top down approach is called a divisive approach and starts with one cluster and splits successive cluster via some distance measure.

Partitional centroid-based clustering does not produce contiguous data and some elements assigned to a group may not be physically in contact with the rest of the group. Non-compact clusters which may have “islands” are not ideal for grouping a CFD domain into homogeneous control volumes. All points in a hierarchical cluster are contiguous since they subdivide the domain via optimal cutting planes.

Any clustering algorithm starts with preprocessing. Preprocessing generally applies filters to data, eliminates outliers, and normalizes the data set. A common practice is to statistically transform data into z-space where it is statistically normalized. Z normalization is accomplished by subtracting the data mean from all points and then dividing the data by its standard deviation. Another part of preprocessing is feature extraction where some type of heuristic is used to locate

“features” which are computed from the original data set. For example, a velocity vector may be decomposed into a unit normal and magnitude during preprocessing. It is typical to normalize the values of the data to span zero to a maximum value of one.

There is a vast collection of clustering algorithms in the literature and can easily confuse a user attempting to select the best suited algorithm for a specific task. There are no theoretical guidelines to assist on proper features selection for a specific situation. The user must gather facts and domain knowledge and make best assumptions (Jain et al. 1999). A well planned investigation of available features, filtering, and transformations can yield significant improvements in results. Features can be categorized as follows (Gowda and Diday 1992):

1. Quantitative features:
    a. Continuous values (e.g. weight).
    b. Discrete values (e.g. number of computers)
    c. Interval values (e.g. the duration of an event)
2. Qualitative features:
    a. Nominal or unordered (e.g. color)
    b. Ordinal (e.g. qualitative evaluations of temperature “cool” or “hot”)

A distinction in clustering methods is whether the method uses a hard membership assignment or soft. Hard membership assigns each data point to one cluster. Soft or “fuzzy” membership assigns a degree of membership for each data point across multiple clusters. Fuzzy clustering requires more memory and computational effort, but it also allows clustering to be more mobile and avoid local minima. Ultimately soft clustering’s final step is to assign a hard membership by picking the strongest cluster membership for a final assignment. The most common implementation of fuzzy clustering is “Fuzzy C-Means”, which is fuzzy k-means. FCM was developed by (Dunn 1973) and improved by (Bezdek 1981).

No clustering approach is globally suitable for all situations due to the varied and complex structures of data that exist, particularly in multidimensional data sets. Approaches to preprocessing, and the algorithms themselves usually contain many implicit assumptions about the data shape. For example, center-based clustering methods, such a k-means, usually assume a spherical cluster shape. The result is there is a lot of literature about clustering as new methodologies are matured which are optimal in specific situations.

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, or a global error, “E”. Distance

metrics are the pairwise measurements between points or a defining point (such as the centroid in k-means). By far, the most popular metric for feature space is the Euclidean distance (Equation 1), 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.

Examples of the most common similarity metrics are:

Equation 1: Euclidean Distance

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

Equation 2: Minkowski Distance

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

Equation 3: Cosine Similarity (dot product)

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

Equation 4: Manhattan Distance

$$d_m = \sum_{k=1}^{d} (x_{i,k} - x_{j,k})$$

The Manhattan Distance or “city-block distance” is simply the absolute difference between attribute values and is not squared. The Manhattan Distance may be thought of as the distance to walk through a city without cutting through any building. Euclidean distances are a special

case of the Minkowski Distance, where p=2. Euclidean distances are the straight line distance "as the crow flies". The problem with higher order "p" values for the Minkowski Distance in Equation 2 is the metric tends to place higher emphasis data points with larger distances. The value of p for a Minkowski Distance does not need to be an integer. Experientially and according to (Whitten et al. 2011), the Euclidean distance, p=2, represents a good compromise in most situations.

K-Means Algorithm and Improvements

K-Means was first suggested in (Sebestyen 1962) and independently in (MacQueen 1967). A detailed history and full pedantic discussion of various deviations may be found in "K-Means Clustering: A Half-Century Synthesis" (Steinly 2006).

K-means attempts to find the minimal sum of the squares of distances, as in

Equation 5. The algorithm seeks to minimize this performance function by recursively assigning membership of points to the nearest cluster. Then a new average cluster value (centroid) is computed. This is embodied in Equation 5 and Equation 6:

Equation 5: K-Means Performance Function

$$Sum\ Squared\ Error = \sum_{l=1}^{K} \sum_{x \in S_l} \left|\left|\vec{x} - m_l\right|\right|^2$$

Equation 6: K-Means Iterative Function

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

K-Means is considered to be generally successful in practice, but has the pitfall of converging frequently to local minimums and is extremely sensitive to initial conditions (B Zhang et al. 1999; Teboulle 2007). Also, it is based on the assumption the data is essentially spherical. In fact, Steinly determined that random restarts numbering in the thousands were required to find global minimums and suggested careful study of initial conditions might produce better converged solutions (Steinly 2006). A number of heuristic approached were created which attempt to find better initial conditions for K-Means.

K-Means++ (Vassilvitskii and Arthur 2007) is an example of choosing careful initial conditions. The algorithm performs as follows:

1. Choose a center at random from among the data points.
2. For each data point x, compute the distance, D(x), between x and the nearest already-chosen center.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Proceed with k-means clustering.

This method outperforms a blind k-means search for a global minimum, but still does not necessary find the global minimum. Kumar and Wasan compare several techniques similar to K-Means++ in (Parvesh Kumar and Wasan 2010). K-Means++ performs as well as X-Means, Efficient K-Means, and outperforms the typical K-Means approach.

Since k-means has a problem with initial conditions and converging to a local minima, several successful attempts have been made at using simulated annealing (Qiu et al. 1994; Rose 1998). Simulated annealing is a method that attempts to prevent convergence to a local minimum by adding a decreasing level of randomness about the mean points. The convergence is annealed which is an analog to the physical process of annealing. The physical process involves heating and cooling of a metal which allows atoms to become unstuck from their initial positions and wander randomly through states of higher energy. A slow cooling process allows them a chance to find states of lower global energy than the initial one. This is done for a minima search space by replacing the current solution by a randomized candidate solution set from samples near the current solution. The new solution may then be accepted with a probability based on a global "temperature" parameter *T* that is gradually decreased during the process.

The most promising improvement to k-means is a method entitled K-Harmonic Means (B Zhang et al. 1999; B Zhang 2000; B Zhang et al. 2003). K-Harmonic means improves on k-means by combining two different ideas. The first is to use a fuzzy membership, so each point is assigned a weighted membership in each cluster. Standard k-means has hard set membership which stiffens the search algorithm. Conversely a fuzzy membership allows the centroid locations to move around more easily without dropping into a global minimum. The second feature of k-harmonic means is averaging using a harmonic means, which causes dynamic weighting and allows centers near each other to avoid being trapped. This is done by increasing the weight on objects farthest from the centroid (Hamerly and Elkan 2002). The performance equation for k-harmonic means is below:

Equation 7: K-Harmonic Means

$$Performance\ Function = \sum_{i=1}^{N} \frac{K}{\sum_{l=1}^{k} \frac{1}{\left||x - m_l|\right|^2}}$$

The k-harmonic means function is insensitive to initial conditions. This means that it need only be performed one time. Zhang shows (B Zhang et al. 1999; B Zhang 2000; B Zhang et al. 2003) that the overall computational effort of finding a global minimum is greatly reduced even though the effort for the individual convergence is more expensive. There has been no volume-weighted formulation of k-harmonic means to date which is critical to clustering CFD cells which represent distinct volumes.

Bisecting k-means is also known to be less sensitive to initial conditions (Savaresi and Boley 2001). Bisecting k-means simple uses standard k-means to hierarchically partition data. It is extreme attractive for document clustering since it produces a hierarchy of division, so classes and subclasses of data emerge. In its simplest form, it works in the following manner (Savaresi and Boley 2001):

1. Find the centroid of the dataset M, which is w
2. Divide M=[x1,x2,…x_n] into two subclusters M_L and M_R according to the rule:
    a. $x_i \in M_L \quad if \quad (x_i - w) \leq 0$
    b. $x_i \in M_R \quad if \quad (x_i - w) > 0$
3. Compute the centroids of M_L and M_R , w_L and w_R
4. Repeat 2-3

Another potential improvement in the k-means algorithm is the use of principle components analysis (PCA) to find the axes on which the global k-means solution lies. PCA is typically used to reduce the dimension of a data set to find the dimensions with the largest variances. Ding and He (Ding and He 2004) have shown that PCA is actually the continuous solution of the cluster membership indicators in the k-means method. Using PCA, the authors were able to show for several demonstrated sets K-means objective are within 0.5-1.5% of the global maximums. This is related to the K-Means Mahalanobis Distance described in the next section when the clusters are essentially allowed to expand normal to the within-cluster PCA principle axes.

K-Means with a Mahalanobis Distance

A very notable deviation to the standard k-means algorithm is to use Mahalanobis distance, which is a statistical distance, instead of the conventional Euclidean distance. Since the basic k-means method proposed by (MacQueen 1967) is based on the Euclidean distance, the clusters created are generally spherical in shape. Standard k-means works well for the case of well-separated, approximately spherical clusters, but fails when the data is hyper-elliptical in shape. The Mahalanobis distance is a measure of the difference of the means between data, which considers also the standard deviations (for a single variable case) to factor the overlap of data. For multiple variables, the covariance between the variables is also considered which allows the clusters to assume an ellipsoid shape. Figure 5 and Figure 6 should the notional difference in clustering results that may be obtained. Better figures are provided in the literature (Younis 1996; Russell and Lines 2003; Kim et al. 2004) of actual clustering results using a Mahalanobis distance.

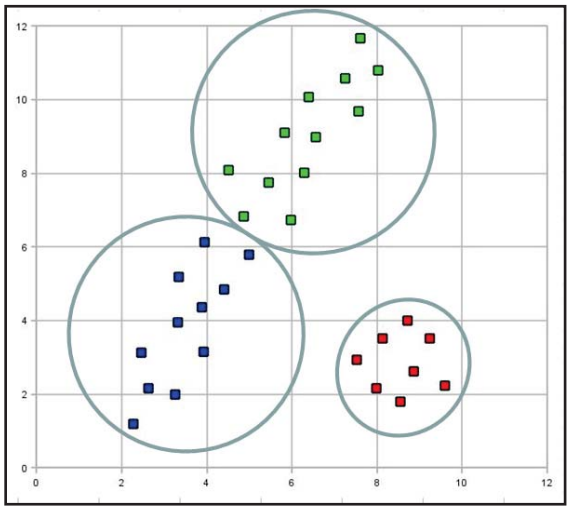

Figure 5: Illustration of clustering results when using a Euclidean distance metric.

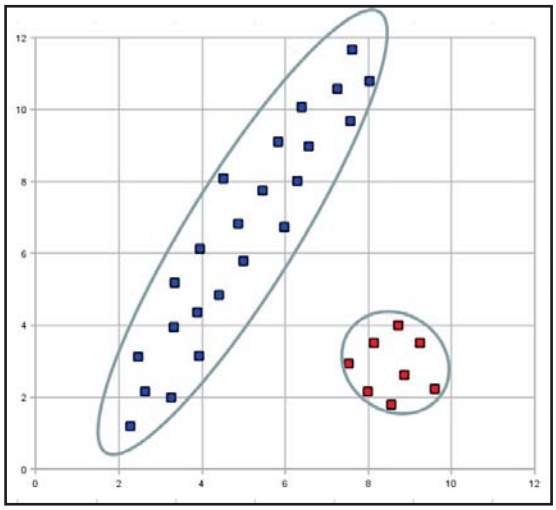

Figure 6: Illustration of clustering results when using a Mahalanobis distance metric.

The use of Mahalanobis for clustering was first suggested by (Johnson and Wichern 1998) but implemented first by (Russell and Lines 2003). Equation 8 is the mathematical equation for Mahalanobis distance. Mahalanobis distance may be substituted directly into the standard k-means algorithm although there are some drawbacks. A detailed explanation of the metric may be found in (McLachlan 1999).

Equation 8: Mahalanobis Distance

$\Delta^2 = (x_i - \mu_k)^T \sum^{-1} (x_i - \mu_k)$, where

$$\sum = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{1L} \\ \sigma_{21} & \sigma_{22} & \sigma_{2L} \\ \sigma_{L1} & \sigma_{L2} & \sigma_{LL} \end{bmatrix}$$

There are definite known problems with using the Mahalanobis metric. Krishnapuram (Krishnapuram 1999) pointed out that the Mahalanobis distance cannot generally be used directly for clustering. Shioda (Shioda and Tuncel 2005) points out there is a challenge in identifying the covariance matrix of the Mahalanobis clustering technique with no *a priori* knowledge. Shioda also points out that when metrics other than the Euclidean distance are used in k-means poor clusters usually result. Younis (Younis 1996) found two specific problems with k-means Mahalanobis clustering in addition to the lack of *a priori* covariance information. The first finding was that the risk that high dimensional data can easily produce a singular covariance matrix. Next, the k-means algorithm in combination with k-means typically produces unusually large or unusually small clusters. Younis also suggests that the partitioning portion of k-means should be converged prior to updating the covariance matrix.

Following the general trend of partitional clustering, Mahalanobis has been integrated into fuzzy clustering techniques as well. The progression to its use started with the Gustafson-Kessel (Gustafson and Kessel 1979) clustering algorithm and the Gath-Geva (Gath and Geva 1989) algorithm. The Gustafson and Kessel algorithm employs an adaptive norm to fuzzy c-means in order to detect clusters of different geometrical shape. According to Liu (Liu et al. 2009), the GK algorithm needs added constraint of fuzzy covariance matrix. The Gath-Geva (GG) fuzzy clustering algorithm expands on the Gustafson-Kessel (GK) fuzzy clustering algorithm, and also takes the size and density of clusters. The GG algorithm can only be used for the data with multivariate Gaussian distribution. Ultimately due to these problems, Liu implemented a fuzzy c-means algorithm based on the Mahalanobis distance (Liu et al. 2009).

Wolfel implemented a technique to weight various features in a Mahalanobis clustering algorithm (Wolfel and Ekenel 2005). Wolfel provides a methodology to weight the covariance matrix components. Essentially the technique is to normalize the covariance matrix and apply a weight matrix based on features.

One last note is there is a similar technique to Mahalanobis clustering called Minimum Volume Ellipsoids (MVE) proposed by several authors (Shioda and Tuncel 2005; Mahesh Kumar and Orlin 2006). Effectively this is very similar to using Mahalanobis distance with k-means when implemented for multiple dimensions. Functionally MVE depends on ellipse fitting algorithms as opposed to computing an inverse covariance matrix or defaulting to a spherical Euclidean distance. Using a fitted ellipse specifically allows a better rejection of outliers in the data but these algorithms suffer from long computation times. Jolion (Jolion 1991) proposed a solution to the fitting problem by employing a subsampling technique to alleviate the time fitting the ellipsoids.

Computational Effort of Clustering Algorithms

An important element of the implementation of any computer algorithm is the resources consumed in pursuit of a solution. Two common measures of the computational effort required as a function of the inputs are time complexity and space complexity. Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the size of the input. The space complexity quantifies the amount of storage space required by an algorithm varies with the size of the problem it is solving. Table 1 shows time complexity for the algorithms researched in this document.

Table 1

Computational Time Complexity of Various Clustering Algorithms. Note: n = number of data points ; K is number of clusters ; i is number of iterations ; d is number of attributes.

Method	Time Complexity
k-means	O(nKid)
k-harmonic means	O(nKid)
c-means	O(n K^2id)
agglomeration	O(n^3)

CFD Multigrid/ Load Balancing

There are a number of ways to subdivide a domain into better subvolumes than arbitrary subdivision. It is very typical for computer aided engineering applications to use hierarchical techniques to provide multiple simultaneous grid resolutions (multigrid) and also to perform domain decomposition for parallel processing. For this reason, it is valuable to consider the state-of-the-art in partitioning algorithms that may be useful for better subvoluming.

Multigrid and domain decomposition are complementary and are very commonly both used in commercial CFD applications. Multigrid reduces the amount of time it takes information to propagate across the domain. Groups of elements of nearby points are essentially compressed into a large "group" which allows the CFD solver to solve a coarse model and then interpolate the results back to the fine grid solution. Domain decomposition is used to assign groups of elements to processors on a high performance computing network. The idea domain decomposition will balance both computational loads and minimize communications requirements between processors.

The most common technique used for multigrid/ domain decomposition is the METIS library (Karypis and Kumar 2011). METIS is a graph partitioner which implements several algorithms including multilevel *k*-way, and multi-constraint partitioning schemes. Essentially the partitioning of the domain is done by forming the following graph representation:

1. Vertices represent computational tasks. Vertices are assigned a weight proportional to their task.
2. Edges represent data exchanges. Edges are assigned weights that reflect the amount of data that needs to be exchanged.

METIS finds partitions which attempt to balance computational loads while minimizing cuts across edges since those represent data exchanges. Successive subdivision is performed on the domain until elements have been grouped into larger groupings as necessary. METIS is very efficient for large domains.

Another type of grouping used for partitioning and multigrid is the agglomeration technique. The objective of agglomeration is to derive coarse grids from a given fine grid. Agglomeration works according to the following algorithm (Venkatakrishnan and Mavriplis 1995):

1. Pick a starting vertex on one of the surface elements.
2. Agglomerate the elements associated with neighboring vertices which are not already agglomerated.
3. Place the exposed edges in a queue.
4. Pick the new starting vertex as the unprocessed vertex incident to a new starting edge which is chosen from the following choices given by order of priority:
   a. An edge on the front that is on the solid wall.
   b. An edge on the solid wall.
5. Go to Step 2 until the control volumes for all vertices have been agglomerated.

Research Objective

There are five main objectives to this research:

1) Demonstrate a new process to accelerate heat-up/ cooldown simulations by using clustering to create a simplified lumped parameter thermal network from a steady-state CFD solution which is then used to solve a transient solution. This is a unique contribution to the literature.
2) Show that convection coefficients from the CFD boundary layer may be remapped to local region convection coefficients based on the clusters which enclose the wall areas.
3) Investigate the use of clustering to track temperatures around specific equipment or locations in the domain by fixing clusters around points or bounding boxes. This is a unique contribution to the literature.
4) Show the new clustering technique can be used to visualize flows within a CFD domain. This is a unique contribution to the literature.
5) Perform a validation study of the subvoluming approach and the newly proposed clustering approach to a fully transient CFD solution. This is a unique contribution to the literature.

Objective 1: New Clustering Process

The main goal of this work is to demonstrate an efficient clustering process to reduce an already-solved steady-state CFD model of a vehicle cab into a fully-transient reduced-order model that can be run in the commercial code MuSES. The goal is to extract as much physics information from the CFD domain as is possible and to use it in the reduced-order model. While it was found that the basic k-means algorithm is well suited to the posed problem, other clustering techniques and variations of k-means may also work. There is a very limited amount of data in the literature about data mining spatial information (Yang and Parthasarathy 2006), so this work will contribute to that body of knowledge.

Objective 2: Localized Convection Coefficients

Volume-weighted clustering can be utilized to find better convection coefficients “h” for thermal models by computing “h” relative to clusters. Keep in mind the two alternatives are subvoluming (which fixes h at 15 w/m^2) and coupling CFD and MuSES at each timestep. The two typical methods of defining a heat transfer coefficient reference temperatures are 1) bulk flow temperature 2) cell nearest the wall. These are illustrated pictorially in Figure 7. While the bulk temperature is the standard method for hand calculations, it is insufficient to capture the localized

temperature differences in a volume. Computing "h" based on a reference temperature of the cell nearest the wall is the default for most CFD codes. The near wall temperatures are too localized and do not react to changes in the bulk flow. Further, T_{FILM} temperatures are coupled tightly to the wall temperature. Since the conventional methods represent two extremes, a local reference temperature, T_{LOCAL}, that is some distance from the wall would be an optimal compromise. Equation 9 may be used to find a local convection equation, where the local temperature is obtained from the cluster containing the wall element.

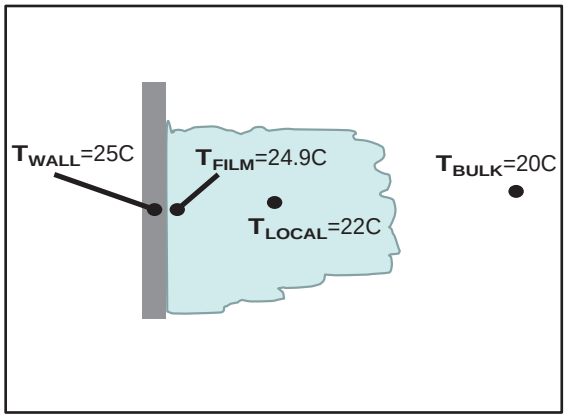

Figure 7: Illustration of the difference between convection concepts.

Equation 9: Local convection coefficient from T_{film} and h_{film}:

$$h_{local} = \frac{h_{film}(T_{film} - T_{wall})}{T_{local} - T_{wall}}$$

Another related item that may be explored related this this is that a clustered CFD domain could be better coupled to a MuSES model by using local convection coefficients. If the flow doesn't change except for the energy equation, this would allow convergence between the codes in a lesser number of iterations.

Objective 3: Enhanced Temperature Tracking

Another objective of the research is to show that clustering will allow for automatic discovery of temperatures around objects. A common problem in modeling vehicle interiors is that surrounding-air electronic box temperatures are difficult to assess. Specification sheets on

electronics usually call out a maximum operating temperature. It is an assumption that once the nearfield surround air is below that threshold, the equipment will operate properly. Ideally the equipment would be checked for operational reliability with various transient ramp rates, but to-date there is no established test procedure.

By breaking the CFD domain into optimal clusters, some of the clusters can be fixed at equipment centroid locations. This allows automatic decomposition of the CFD domain so that the new lumped network would output air temperature surrounding equipment. The equipment air temperature is highly useful for determining if the electronics will overheat. An example equipment rack is show in Figure 8 and Figure 9 below. There are approximately 40 electronics boxes which need to be tracked to assure they are within their operating specification. Additionally a typical cooldown simulation requires that the air around the boxes be cooled to the required temperature within a certain time period.

This clustering around equipment application is useful beyond coupling to MuSES simply to find reasonable temperatures around equipment in the CFD domain.

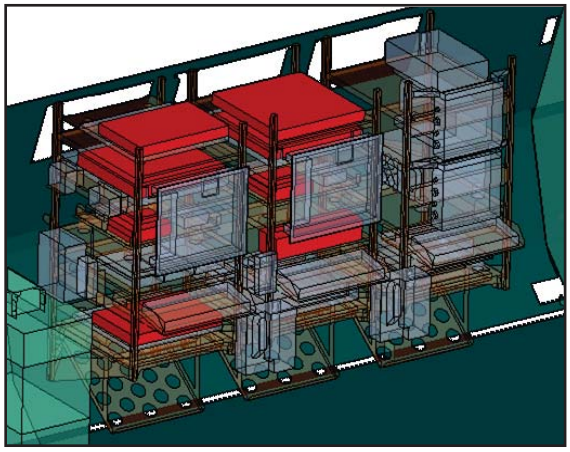

Figure 8: Example of a command and control mission equipment rack in a military vehicle.

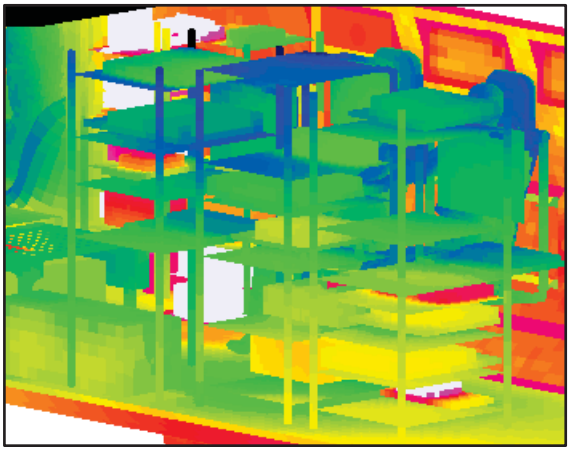

Figure 9: Example of equipment racks thermal solution in typical vehicle cab.

Objective 4: Enhanced Flow Visualization

Since the CFD domain is highly complex, it is very useful to use a simplified flow network for enhanced visualization. The simplified nodal network 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 fluid flux. The lines and colors could be also be colored by fluid temperature or velocity. Figure 10 and Figure 11 below show a notional example of the reduction of the complex flow to a network for visualization and illustrate the amount of information that may be captured in a simple representation.

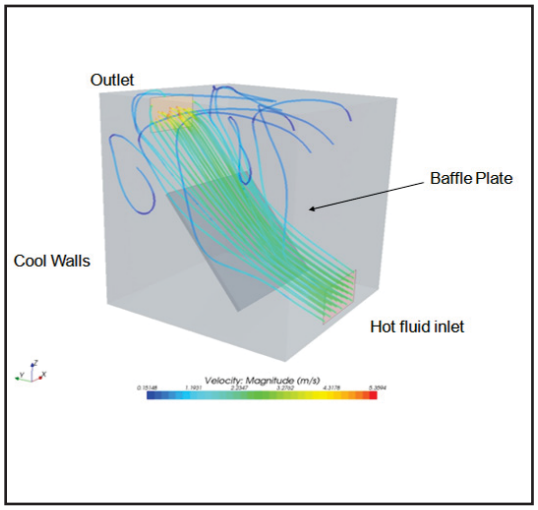

Figure 10: Sample domain visualized with particle traces.

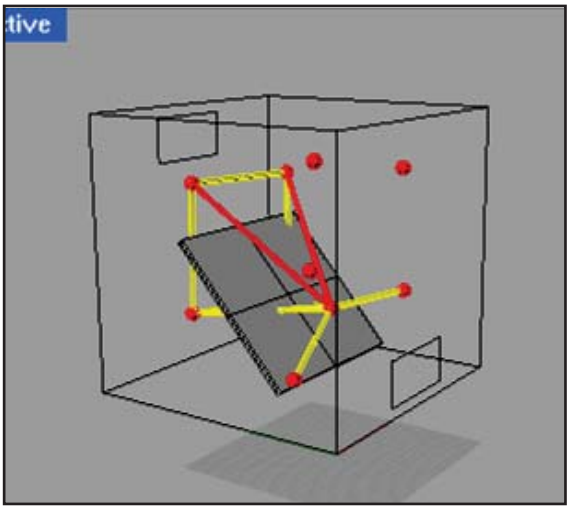

Figure 11: Example of simplified flow visualization. This is an example of what the interconnections between clusters might look like. High flow connections are shown as red, medium are shown as yellow and low flow connections are omitted.

Objective 5: General Expiration of Clustering Methods

There are a vast number of clustering algorithms that may be suitable for Objectives one through four. Part of the goal of this research is to try and find the best method for a reasonable computation time. K-means, agglomeration, k-means with a Mahalanobis distance, and k-harmonic means will all be investigated. The “knee-in-the-curve” best value algorithm will be implemented for the production version of the code.

Constraints

The constraint for the developed methodology is simply to perform all clustering and subsequent solution phases in no more than a few hours on a single CPU machine. This constraint is due to the fact that the solution must perform at least two orders of magnitudes faster than simply running the computational fluids simulation. Unless this magnitude of overall process enhancement is achievable, the additional human intervention in the solution process does not make any sense.

Additionally, the end product needs to be “commercial” quality software code with a graphical user interface (GUI) and relatively easy to use. The code needs to read polyhedral mesh which is notoriously difficult to contend with since the mesh may have “n” (any number) of faces on an element and each element may have “n” sides. Polyhedral mesh precludes the use of simple repetitive numeric schemes on faces since nested loops must be utilized.

Key Results and Metrics

Newly Developed Process

The new overall process developed is shown in Figure 12. The process starts with a steady-state MuSES model (predicts wall temperatures) that is manually coupled to a steady-state CFD model (predicts convection). Once the manual coupling has converged, the clustering algorithm is run. The clustering result is visually inspected by exporting to the Ensight Gold format and then adjusting clustering parameters until a satisfactory clustering has been achieved. No metric has been established aside from visual inspection for accepting a particular clustering scheme. 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. This will be discussed in detail in subsequent sections. Next, the 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.

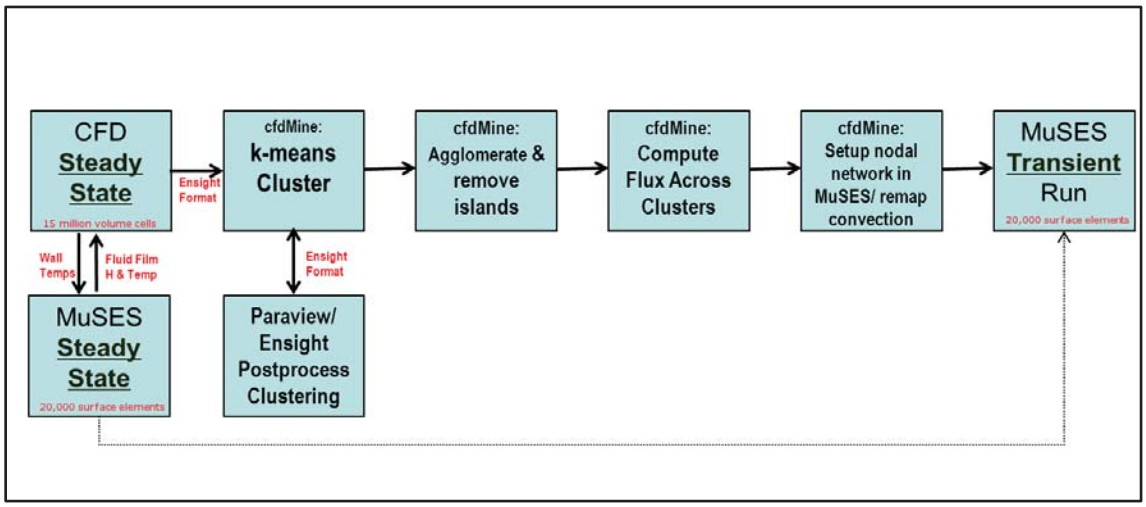

Figure 12: New process to use a steady-state CFD solution to create a transient MuSES thermal simulation.

Feasibility of Clustering CFD Data

The key goal for this dissertation is to find clusters which capture flow temperature and distributions. Visual inspection of the clusters along with the overall ability to minimize sum squared error for the domain are the primary techniques used to assess suitability of the overall method and to learn which weights deliver the best result. Much effort may be spent in trying to find an optimal value, and it may not be particularly significant to simply proving out a new process.

Clustering should be thought of as a lossy data (or information) compression method. Essentially the CFD domain will be compressed into a smaller number of clusters. Taking this into consideration, it is desirable to compress the domain into a simplified nodal network with a minimal loss of information. Since the CFD domain is governed by physics, there is information available that is not conventionally available to most clustering applications. The subvoluming technique (Hepokoski et al. 2010) takes very little advantage of available information. However, as more physics are incorporated into clustering the CFD domain, the computational cost goes up as well. A balance must be struck between the amount of physics captured and obtaining a reasonable solution time for production work. The obvious information available to cluster for the cooldown CFD domain is:

1) Physical proximity to other cells.
2) Physical proximity to walls.
3) Cells have volumes and are not just discrete sample points in space.
4) Temperature, velocity direction and velocity magnitude. Other parameters could also be clustered such as turbulent energy, vorticity, etc.
5) Boundary conditions are known states in the CFD data.
6) Upwind communications / path distances.

Some thought was given to which parameters in the CFD domain to cluster. There are a very large number of parameters available, but fundamentally a CFD solution tracks conservation of mass, momentum, and energy. Most of the other available data are derived quantities and will contain recombinations of the basic information. Principle independent data is generally preferred. However, some of the quantities such as turbulence parameters, which are models of more complex behavior, may be valuable. Incorporation of these was not investigated in the current research.

Of the identified information, it is relatively easy to capture the physical proximity since this is what is normally dealt with in clustering. Volume may be incorporated by volume weighting the

contribution of points to the clustered solution. The temperature, velocities, etc. may be captured by incorporating those into the distance metric. Boundary condition locations may be captured by specifying fixed cluster center locations. Upwinding (flow along streamlines) and path distances are not very easy to capture numerically due to computational intensity. It would be ideal to use particle traces to measure upwinding distances. The nearest fast reasonable approximation to an upwinding particle trace is to use an advancing front technique across the mesh (which has implicit connectivity) to assure clusters only contain cells reachable from the kernel. The advancing front technique is basically agglomeration.

Initially to explore the feasibility of this project, two test CFD cases were developed and solved as shown in Figure 13 and Figure 14. The cases are quite simple compared to a model of the inside of a vehicle, but to gain a low level understanding of the nature of this numeric scheme a simple problem is best. Also, the original test cases were based on structured grids which are somewhat rare in commercial CFD due to the complexity of geometries. The flows modeled are simply a hot or cold jet of air impinging on a baffle.

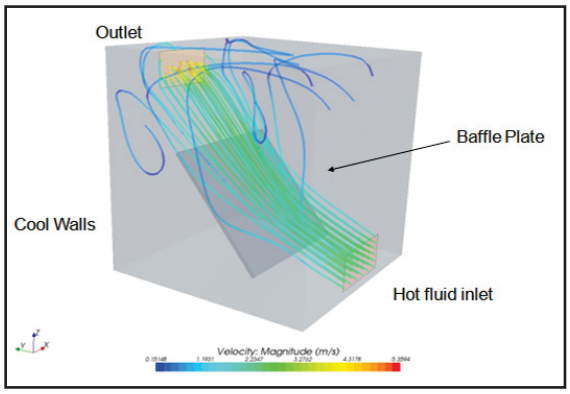

Figure 13: Test Case 1. This case consists of a hot fluid impinging on a baffle and exiting from an outlet slot. The walls of the cube are held at a constant temperature. The mesh is 488418 volume elements.

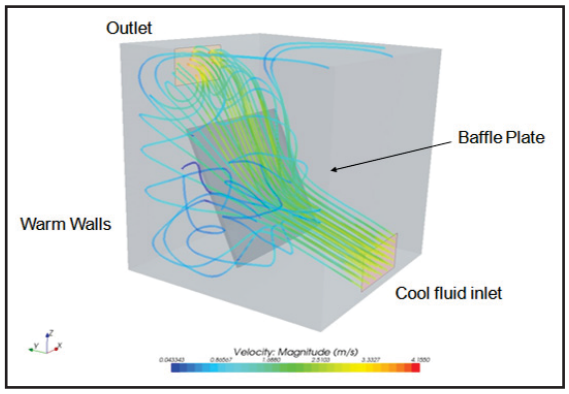

Figure 14: Test Case 2. This case consists of a cool fluid impinging on a baffle at a different angle and exiting from an outlet slot. The walls of the cube are held at a constant temperature. The mesh is 483162 volume elements.

After reviewing the literature, it was decided the k-means clustering had the most potential in terms of performance, computational intensity, and also suitability to fixing nodes and for visualization. A prototype MATLAB code was created (see Appendix A) to perform weighted k-means clustering of the test cases. The limitation of the MATLAB code was it could not actually read mesh connectivity data, so no flux computations could be made. The code only read the

cell-center discrete data: temperature, velocity, centroid, and volume. Also, the original MATLAB code did not perform a z-score preprocessing of the data and thus the feature weights are relative to the data preprocessing. The method used for preprocessing was to divide each point by its spread (maximum value less the minimum value).

The MATLAB prototype employed volume-weighted sum approach (Manhattan distance) as opposed to squaring each distance (Euclidean). The final code, called cfdMine uses the Euclidean distance. The squaring of the distance provides tighter clusters as it penalizes outliers very heavily. Figure 15, Figure 16, Figure 17, and Figure 18 show the result obtained by running volume weighted k-means to cluster the flow domain into ten clusters. After running a number of experiments, good distance weighting (as judged subjectively) was determined to be:0.9 Spatial; 0.1 Temperature; 0.05 Velocity Magnitude; 0.05 Velocity Direction. The k-means algorithm was run ten times with randomized starting points in an attempt to find a global best solution for the clusters.

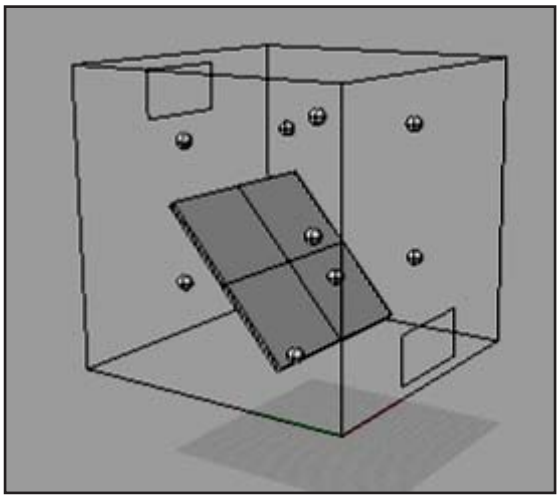

Figure 15: Test Case 1. This image shows the cluster nodes for ten clusters. The weighting was as follows:0.9 Spatial; 0.1 Temperature; 0.05 Velocity Magnitude; 0.05 Velocity Direction.

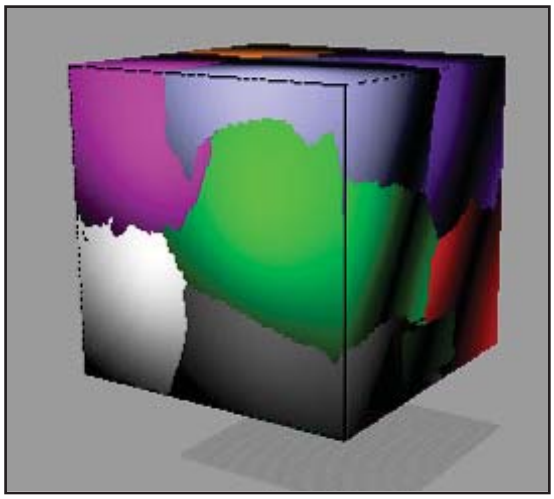

Figure 16: Test Case 1. This image shows the cluster points for the ten clusters. Each color indicates a different cluster. The weighting is the same as Figure 15

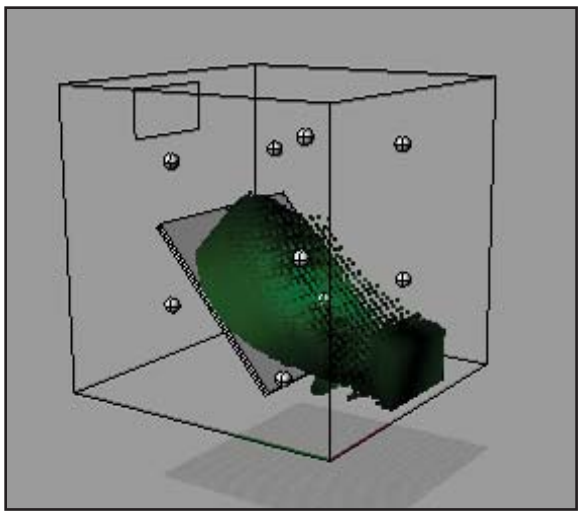

Figure 17: Test Case 1 This image shows only cluster 4. Note the symmetry and lack of "islands" of data. The weighting is the same as Figure 15.

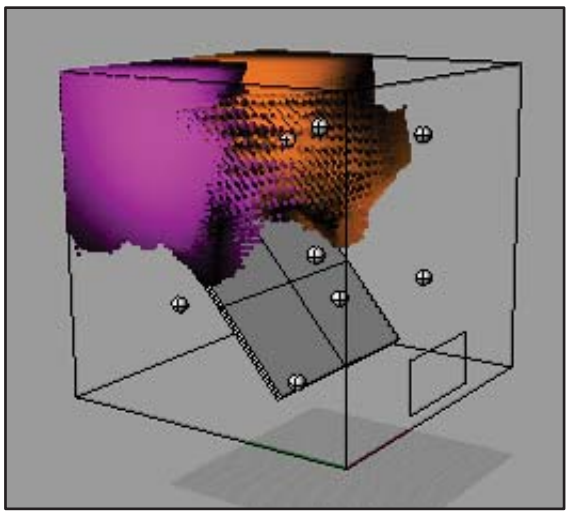

Figure 18: Test Case 1. This image shows clusters 6 and 8. Note the symmetry and lack of "islands" of data. The weighting is the same as Figure 15.

For Test Case 1, k-means is noted to break the domain into distinct clusters which appropriately track the hot jet. The interface edges between clusters are not especially jagged and there are very few "islands" of points that are not part of the main cluster. Both islands and jagged edges would be undesirable because they artificially create a large surface area for computing flow between clusters. Figure 19 shows the similar results for Test Case 2. The typical solution time on a 3.0 GHz processor for the prototype MATLAB code is only about 8 minutes (the case is 488418 elements) which is very reasonable.

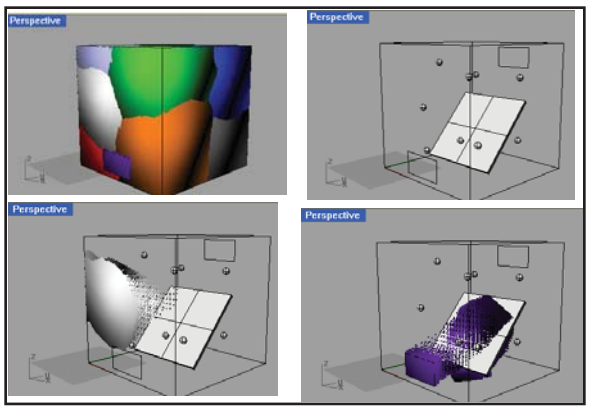

Figure 19: Test Case 2. This image shows the cluster results for ten clusters. The weighting was as follows:0.9 Spatial; 0.1 Temperature; 0.05 Velocity Magnitude; 0.05 Velocity Direction

As mentioned, ten clusters seems to be the minimum quantity for the test cases based on a comparison of Figure 15 to Figure 20 and Figure 21. The only difference in the figures is the number of clusters specified for Test Case 1. Seven clusters is insufficient as it fails to capture the hot, high-speed jet outlet. Fifteen clusters appears to work quite well, but one should consider the problem to decide if ten or fifteen nodes is the better choice. Note in Figure 20 that the hot jet is broken into two distinct sections.

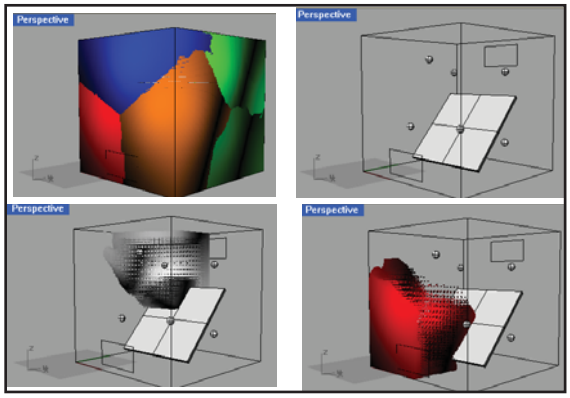

Figure 20: Test Case 1, 7 Clusters. The weighting was as follows: 0.9 Spatial; 0.1 Temperature; 0.05 Velocity Magnitude; 0.05 Velocity Direction.

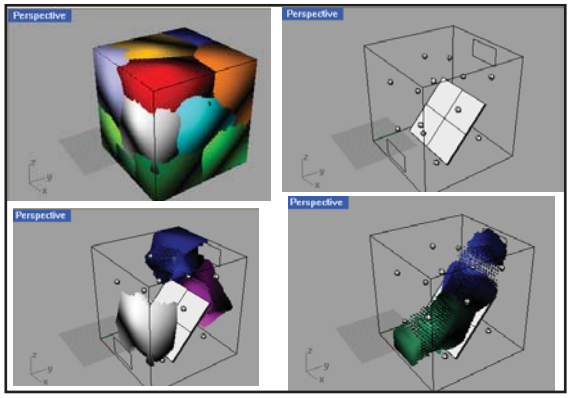

Figure 21: Test Case 2, 15 Clusters. The weighting was as follows: 0.9 Spatial; 0.1 Temperature; 0.05 Velocity Magnitude; 0.05 Velocity Direction.

Volume Weighting

Volume weighting is a necessity for clustering CFD data since the computational grid does not consist of a randomly distributed sample of points. It is common practice in CFD modeling to generate more mesh points near surfaces subject to the nature of the boundary layer treatment. Additionally CFD solutions usually require regions with a lot of flow details to be refined as well. When a typical CFD domain is run through a clustering algorithm, with no volume weighting, high density regions dominate the clustering centers which is not generally desirable. Figure 22 and Figure 23 show typical mesh densities for trimmed polyhedral meshes. Notice that the density of cells is high near boundaries and regions of high gradient.

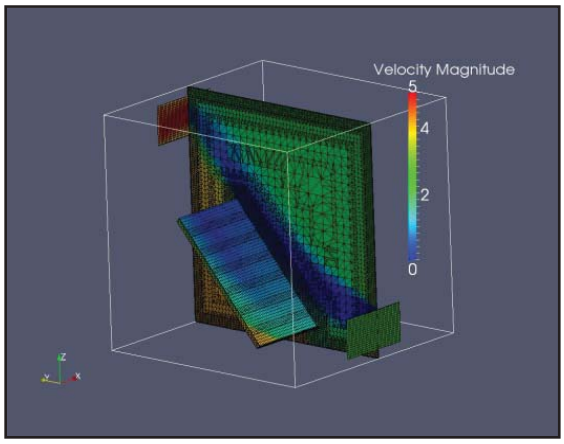

Figure 22: Test Case 1 Mesh Cutting Plane.

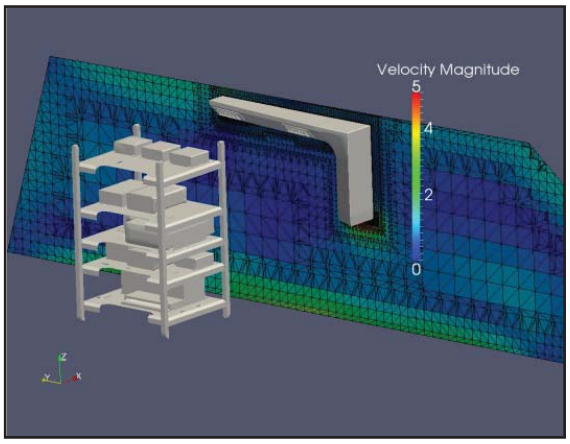

Figure 23: Typical vehicle HVAC mesh cutting plane.

Changing the k-means recursive function to a volume-weighted one is quite easy and the formulation may be found in

Equation 10.

Equation 10: K-Means Volume-Weighted Iterative Function

$$\bar{m}_l = \frac{\sum_{x \in S_l} v_x}{n_l} \sum_{x \in S_l} \vec{x} \cdot v_x$$

To demonstrate the need for volume weighting more explicitly, Test Case 1 was run weighted and unweighted. A cutting plane through the center of the domain was taken and it is easy to see

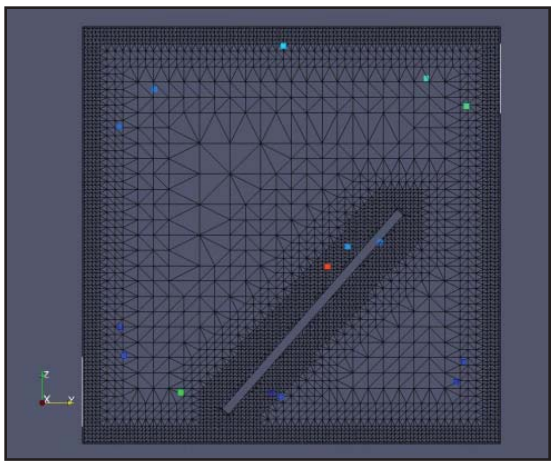

Figure 24: Side View of Test Case 1 Clusters, Not Volume-Weighted.

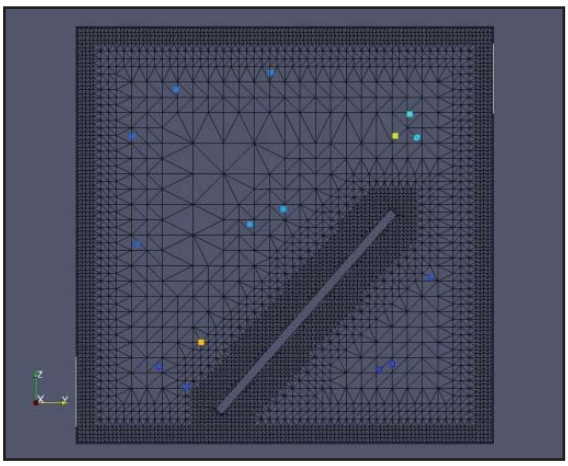

Figure 25: Side View of Test Case 1 Clusters, Volume-Weighted.

from a side view that the cluster centers are drawn to regions of high density (Figure 24 and Figure 25). Again, this shows that the boundary temperatures and velocities will dominate the cluster centroid averages. Although, uninvestigated, it may be beneficial to volume weight the temperature and velocity of the cluster, but leave the geographic position unweighted. That would draw clusters toward boundaries, which may be desirable as long as the average temperature is properly weighted for export into the MuSES subvolume. There is a potential that in some cases, it may be desirable to actually have boundaries and areas of high interest dominate the clustering. The best way to implement this would be to allow a user to select how much volume weighting to apply.

Another observation that can be made about Figure 24 and Figure 25 is that the clustering is able to find the symmetry of the flowfield across the center plane. Since the CFD domain is symmetric, the centroids that appear as “pairs” are really centroids from the right and left sides, which nearly lay on top of each other. This is typical and desirable if the flowfield is symmetric. This is observed as pairs of nodes shown from the side views in Figure 24 and Figure 25.

Data Normalization

It is recommended that the data be stabilized to the z-domain in the following manner. Subtract the mean from all data. Divide the data by its standard deviation. This allows for better weighting to be achieved. 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 compressed in the z direction. This is illustrated in Figure 26 and Figure 27. Figure 26 treats

x,y,z data independently and results in different penalties for distance in the three axes. Figure 27 shows the clusters that result from penalizing each geometric dimension equally by averaging the standard deviations.

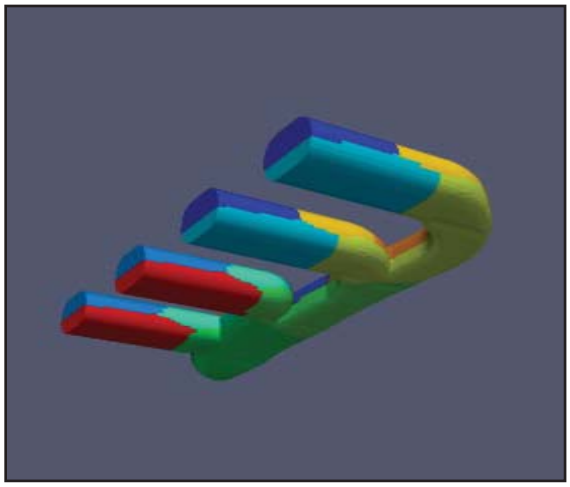

Figure 26: Clustering results when z normalization is not averaged for x,y,z coordinates.

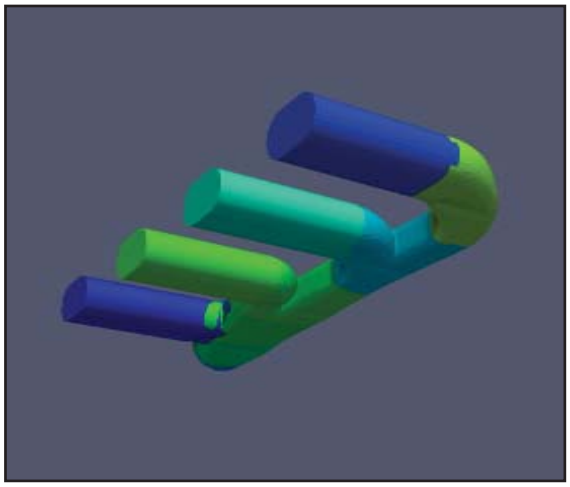

Figure 27: Clustering results when z normalization is averaged for x,y,z coordinates.

Software Tool Development

The developed clustering tool that is the output of the present research has been named "cfdMine" by the author. A copy of the code may be requested from the author, but due to the length of the code (multiple thousands of lines) it is not included. The entire code was tested and developed under Linux (SuSE and Mint mainly) since Linux allows for the mixing of release and debug code. Also, most of the Department of Defense's compute clusters are based on Linux. Most commercial software code is developed using C++. For this reason, there are a large number of development tools available for code development and cfdMine was developed entirely in C and C++. Since a fully portable code was desired, cfdMine uses all statically linked libraries so there is only one executable.

Much effort was put into finding the best code libraries that would maximize code readability while being cross-platform and easy-to-use. cfdMine uses the following library components:

1) QT - Cross-platform application and UI framework (2011b).

2) Eigen - a C++ template library for linear algebra: matrices, vectors, numerical solvers (2012b). Eigen is as fast as free versions of BLAS.

3) kdtree – a simple library for working with kd-trees (2009).

4) tdfIO – a software API for Thermoanalytics' MuSES code (2012c)

5) libEIO – out of date undocumented Ensight reader / writer that required heavily modification

The graphic user interface (GUI) as show below was developed using QT which is one of the major GUI codes along with Tcl/Tk. QT especially stands out as also emphasizing that it is a cross-platform development tool. This means that code written within the QT framework may be compiled on nearly any platform (Mac, Windows, or Linux).

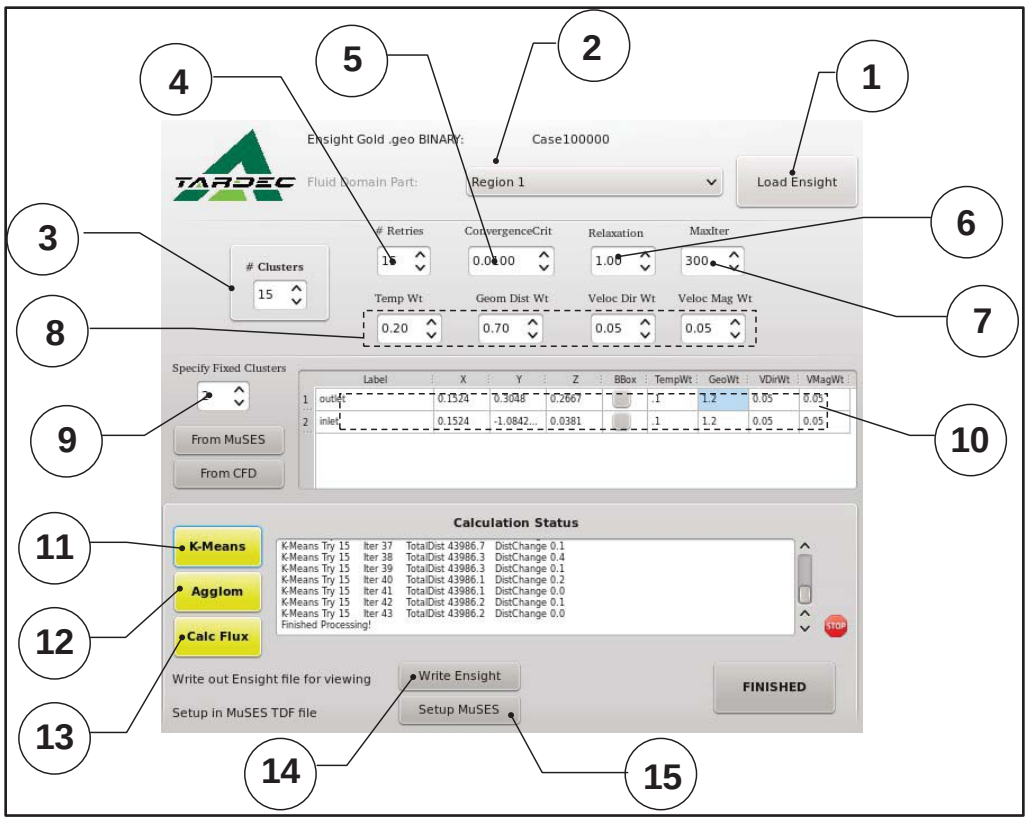

Figure 28: cfdMine Beta Version GUI.

The software code that TARDEC uses for computational fluids is StarCCM+. A large amount of effort was expended in attempting to establish how to extract polyhedral data from the CFD code and read it into useable data structures. Since the government does not show favoritism to any software or company, it was highly desirable to find a commonly used file format that many codes support. StarCCM+ (CFD code) only exports two formats which contain polyhedral data: Ensight Gold and Tecplot. Neither format had a readily available read/ write software library. Code components from the open source projects OpenFoam and Paraview were explored for their adaptability to the project. Both Paraview and OpenFoam read polyhedral mesh. OpenFoam does not read results data. Paraview actually allows for user “filters” and already has a k-means filter. However, to allow stand-alone use with a GUI, Paraview was not selected.

Ultimately an out-of-date library from CEI (Ensight) was adapted to handle reading and writing the latest Ensight Gold file format standard (2011c), including polyhedral mesh. An additional benefit of using the Ensight Gold format is that output from cfdMine may be written out into that format for

easy visualization as well. One simply appends cluster assignments to the existing Ensight dataset and reads everything into Ensight or Paraview.

A number of complications arise from the use of the Ensight Gold format. The first is that StarCCM+ does not export the mesh face data. This means every element type in the Ensight Gold format (2011c) must be decomposed into faces from the input data. Next, the faces must be checked to see which faces match up (i.e. indicated there is flow between faces). It is convention to use the right hand rule to make each face point to the interior of the volume element. However, the exported Ensight Gold data from StarCCM+ does not obey this convention. To assure the faces match, the number of vertices that match between faces is counted and must equal the total vertices on each face and then denotes a shared face. This is slightly slower than simply finding one matching node and checking the subsequent nodes (ordered by the right hand rule) for a match, which is preferred by convention.

Exhaustively checking every face against every other face in the mesh domain is too computationally expensive. To accelerate the search process, an index is made for the faces connected to each vertex. Then, when a face is checked for a match, possible matches can be found by seeing which faces share the vertices in an element. It is unknown if this algorithm is common practice, but it works very efficiently.

Once a matching face is located, a record is stored of the match. This record will be needed later to compute fluxes between clusters and also to perform agglomeration. Without deciphering the mesh connectivity, essentially the mesh is nothing but a jumble of unconnected data. Another important feature of the mesh is the area of the faces which connect the mesh. The polygon area is computed using an adaptation of the numerical algorithm described by O'Rourke (O'Rourke 1998). The O'Rourke formulation is for the area of a polygon on a plane. Therefore, the surface normal of the element must be computed. An averaged normal is computed for each element since there is no guarantee that elements actually form a plane. Actually it can be assumed they are close to a plane, but no meshing tool creates perfect geometry no matter how much smoothing is employed.

cfdMine GUI Features

Referencing Figure 28 which labels all the GUI components, descriptions are as follows:

1. Load Ensight Button: This button loads the Ensight data as exported from the CFD code. The data is stored in multiple binary files. At a minimum cfdMine requires the following

scalars: temperature, volume. Additionally the following vector data is required: velocity vectors, and element centroids. Finally the ".geo" file contains the actual mesh structure.

2. Region selection. This allows the user to pick which region (fluid region only) to perform the clustering on.

3. Number of clusters. The user may pick the total number of clusters the algorithm will utilize.

4. Number of retries. As discussed in other sections, pure k-means is prone to falling into local minima. To attempt to try and find the global minima, several random starts of the algorithm are recommended.

5. Convergence criteria. The convergence criterion as specified is the change from each iteration of the k-means algorithm in the global distance, or the total error "E".

6. Relaxation factor. This is the proportion of cluster center change that is updated at each iteration of the k-means algorithm. Mainly this is for research and to stabilize the algorithm, if necessary.

7. Maximum number of iterations. This is the limit of iterations (per retry) that cfdMine will perform. Sometimes, the algorithm will not converge completely and will oscillate slightly and this is a means to eliminate hanging up the algorithm.

8. Temperature, geometry, velocity magnitude, and velocity direction weights. The user is able to choose the importance of the various criteria during clustering. Each parameter is normalized during the solution to have equal influences which are then multiplied by the weights to properly influence the solution. Note the velocity direction takes the dot product of each element to the cluster center. The others are expressed as intuitive distances.

9. Number of fixed-position clusters. Since one goal of this research is to predict the temperature around specific components, those may be specific here. Each fixed cluster will be prevented from moving during the iterative solution. The temperature, velocity, and membership in the cluster will, however, change with each iteration. Note there are two ways to fix the cluster positions: 1) The cluster may be fixed around a point. 2) The cluster distance may be computed from the component bounding box and the point is represented at the centroid of the bounding box. The fixed cluster data may be loaded from a MuSES file or from the already loaded CFD geometry.

10. Fixed-position cluster weighting. Each fixed cluster may utilize a unique set of weights to achieve the desired size and wrapping of the surrounding elements. Generally higher geometric weighting will be assigned which makes the cluster smaller since it is harder for elements to achieve membership in the cluster.

11. K-Means button. This button starts the k-means iterative solution. The global error may be monitored in the “calculation status” window (as shown).

12. Agglomeration button. This button is provided to be used after the k-means solution has been executed. The basic idea is that the k-means clustering creates “islands” or elements detached from the bulk cluster at the fringes. To prevent this and assure all members of each cluster are contiguous neighbors, agglomeration is used. A seed is chosen for each cluster and connected elements are found. Elements which cannot be reached by their k-means assignments are thrown into a candidate pool where each cluster expands out and grabs the lowest distance elements. The end result is that no islands are possible and minimum contiguous distance is achieved.

13. Flux calculation. This button calculates the fluxes between the clusters. It is provided as a separate operation as it triggers the calculation to find the faces between elements. This is a time consuming process. Cluster may be achieved without pressing this button until it is required to export the data into MuSES.

14. Write Ensight. This button writes out the results of clustering into the Ensight Gold file format for visualization. If the fluxes have been computed, additional data is output to represent the flux linkages between the clusters.

15. Setup MUSES. This button opens a MuSES .tdf file and creates “fluid nodes” representing the final clusters. Additionally, it creates “advection links” which specify how much volume flux flows between the fluid nodes. This also remaps all the fluid-film convection to the cluster nodes.

General cfdMine Software Usage

The typical way the software is used (reference the process diagram in Figure 12) is to first create a steady-state CFD model in StarCCM+. The CFD model is coupled to MuSES and the two are manually iterated until a convergence is achieved between the two codes. 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 (2004), but may take up to 20-30 times.

Next the StarCCM+ solution is exported in Ensight Gold format. CfdMine is able to directly read this format and specifically reads: .geo mesh, cell temperature, cell volume, and cell velocity vectors. At this point the user selects appropriate weights for the distance vector. Additionally, fixed nodes may be selected from either the Ensight Gold file or from any MuSES TDF file. The fixed node distance may be computed from the centroid or from the bounding box based on a check box selection on the GUI. The fixed nodes may have their own weighting parameters which allow them to cluster more tightly to the component or boundary than the size of more general clusters.

The next step is to execute the k-means algorithm. A best practice is to specify zero for the convergence criteria and ~10-20 iterations for one trial, initially. This is a good idea so that some idea of the total distance, or total “big E” error may be understood. The global error is somewhat meaningless in a physical sense and a touch non-intuitive. Also, the weight parameters may be adjusted and a snapshot of what the structure looks like may be observed. Once the entire weight scheme looks promising, a reasonable convergence criterion is selected and the k-means algorithm is run to completion.

At this point, the user may export an Ensight file for visualization in Paraview, Ensight, or any other visualization code. Until fluxes and mesh connectivity is computed, only cluster visualization may be achieved as shown in Figure 29. Either running agglomeration or calculating fluxes causes cfdMine to find all the mesh connectivity, which can be a time consuming process. Once the flux computation is run, the mesh connectivity is cached and does not need to be rerun. If the user wishes to adjust the weights and rerun, the k-means algorithm may be run any number of times until the clustering is acceptable.

The next step is to optionally agglomerate or to calculate fluxes. The agglomeration is used to eliminate islands of data and is discussed later. The flux calculation is required to setup the simplified network in MuSES. The flux calculation also outputs the additional visualization data in Ensight format, once executed. See Figure 29 through Figure 32 for examples of visualizing the clustering results for “Case 1”.

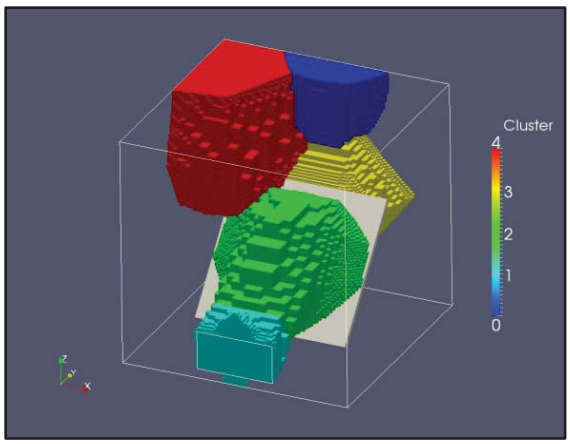

Figure 29: Clustering results for Case 1, clusters 0-4.

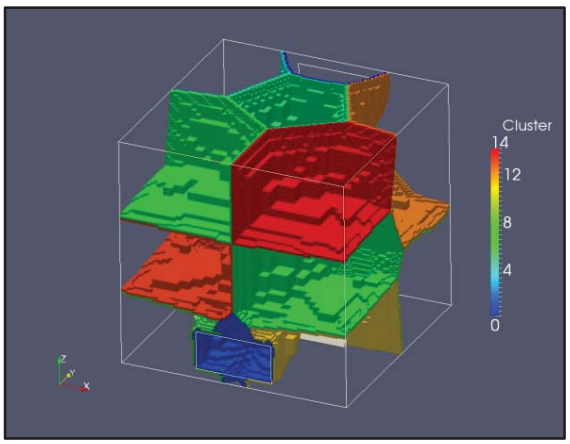

Figure 30: Clustering interfaces. This plot shows only connected elements between cluster borders. Elements are colored still by cluster numbers.

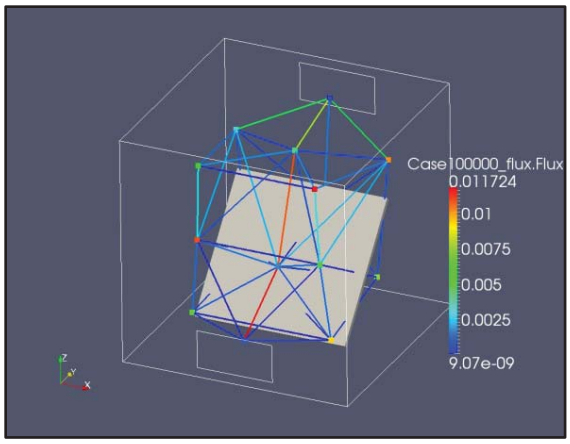

Figure 31: Only cluster centers are shown as square dots. The fluxes between the clusters are shown as lines colored by flux intensity.

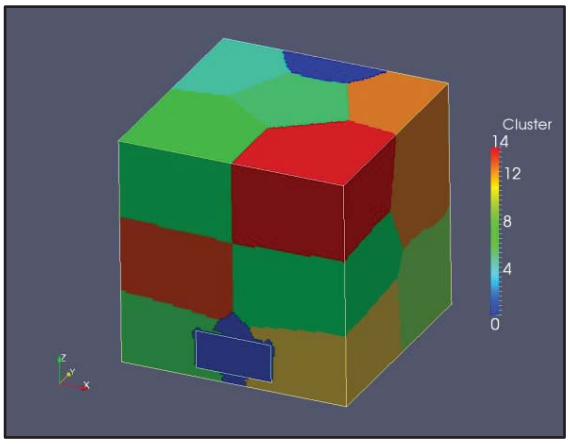

Figure 32: Clustering results for Case 1, clusters 0-14.

cfdMine K-Means Implementation

After much consideration of the problem, the k-means algorithm was selected to cluster the flow data into a simplified fluid network. The main distinguishing feature of k-means is that it is strongly an optimization algorithm. MATLAB was utilized to create the code since k-means calculations are easily vectorized and run quickly. A pseudo algorithm is provided below to outline the scheme utilized.

The MATLAB code developed operates generally in the following manner:

```
1. for i=1 to number of tries
      a. repeat
            i.  randomly select cluster nodes
           ii.  compute distance of every point in domain to nodes
          iii.  assign points to cluster based on minimum distance
           iv.  compute volume-weighted means to find new nodes
            v.  track the solution with minimum total error
      b. until no change
2. end for
```

Generally, the k-means algorithm will tend to converge to a local minimum. The simplest way to overcome this limitation is to seed the algorithm with random starting “nodes” for the cluster and repeatedly converge to a solution, keeping the solution with the lowest total error amongst those tried. This is the reason for the looping labeled “number of tries” above.

For this implementation, the distance metric is computed as the weighted sum of three normalized parameters: the spatial distance, the velocity magnitude, and the velocity direction A CFD solution solves conservation of mass, momentum, and energy, thus providing cell velocity, volume, and temperature. Each cell also has an associated centroid; the centroids are used to represent each CFD volume cell. The spatial distance is computed as the sum of squares of the distance of each centroid point’s i,j,k components to each cluster center. Next, the velocity distance is computed as a simple absolute difference. Then, the velocity direction is computed by evaluating the dot product between the cluster node and the i,j,k velocity components of the dataset points. After all the distances are computed for each point, each parameter is normalized based on global maximum for each parameter so that each has a range of zero and one. This preprocessing method was chosen based on the findings of (Milligan and Cooper 1988) that it is better to standardize on extents rather than z-scores. The extents-based normalization method was only used for the first version of the cfdMine code in this section. The later section on a Mahalanobis version of the code uses z-scores which is found for the present purpose to be much better. The distance metric is then assessed by summing each of the three parameters multiplied times a weight. The weighting allows the user to vary the importance of the parameters to the clustering algorithm.

Usually, a major drawback to the use of the k-means algorithm is its tendency of dense areas of points to bias the cluster. Since the CFD solution provides an associated volume for each point, this limitation is easily negated. The developed algorithm computes mean values and total error

by multiplying each point value times its volume and then dividing the sum by the total volume, yielding a volume-weighted mean and volume-weighted total error. Lack of implementing the volume weighting for this problem tends to pull clusters towards walls because of the fact that the point cloud is very dense along wall surfaces in CFD problems.

CfdMine initializes the clusters by using the nearest random points within the extents of the domain. Another common way of initializing the starting positions is to randomly assign cluster membership. The literature disagrees on which method is best and there is no research on which is best for a volume-weighted domain where areas of high gradients are likely to have denser mesh. The method of using random points within the domain was easiest for coding and gives good results. Since there is no evidence that several random restarts is inferior for the present work, no effort was spent on finding optimal initial cluster centers.

It is entirely possible for a cluster to reach zero membership during convergence. This would create a divide-by-zero situation during the volume weighting. To alleviate this problem, when a cluster has zero membership, a new random position is selected and the algorithm continues to iterate. This usually only happens during the initial few steps of the solution.

The question of how many clusters to use depends on the dataset. It is possible to start with a small number of clusters and work up on the quantity, tracking total squared distance criteria. Care must be taken when using this approach as the minimum total distance is to choose as many clusters as there are data points. Another approach is to consider the value of splitting each cluster and using a tree approach. Each cluster would be split and run successively through k-means (Whitten et al. 2011). There are methods which apply a penalty for each additional cluster created, but the easiest method is to simply employ a visual inspection of the data. Globally optimal solutions were found for the several test cases to help with visual inspection during the research. The global optimum can be found by running an excessive number of random restarts (1000+) which impractical for a production utilization of the code.

Fast Agglomeration

When trying to allow clusters to best match the flow domain, it is desirable to reduce the weighting of the Euclidean distance as much as possible. The extreme opposite is to only use Euclidean weighting which finds clusters of equal centroid about the mean point (i.e. equal volume) which doesn't at all follow the flow patterns. When the geometric distance weight is reduced greatly, islands may occur near the boundaries as shown below in Figure 33.

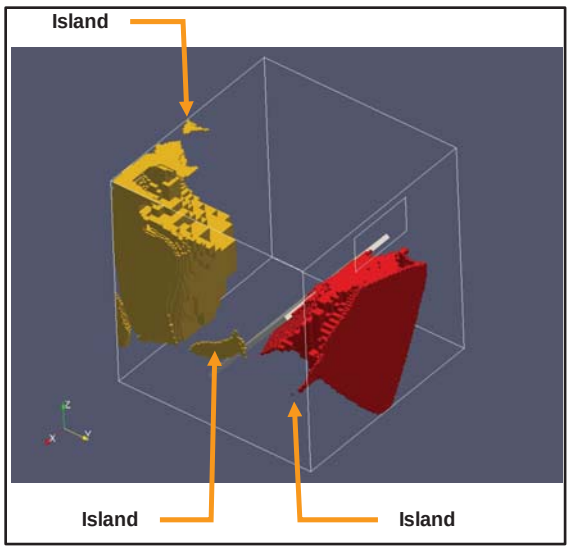

Figure 33: Shown is an example of the results of pure k-mean clustering. Note "islands" of non-contiguous data.

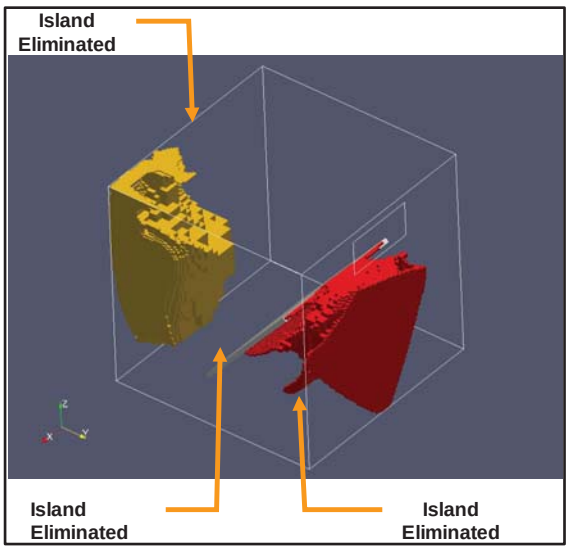

Figure 34: Shown is the previous figure after agglomeration is performed. Note the elimination of "islands".

Islands are an undesirable feature because they may cause flow from one cluster to another where is actually should not physically occur. Additionally, it artificially increases the mixing between clusters.

To eliminate the islands, an expanding-front agglomeration algorithm was implemented. This algorithm was bound by reasonable computation times. For the purpose of speed, a two stage agglomeration was designed. The first stage starts with the element closest to the center of the already computed k-means cluster. From there it expands out to all connected elements in the same cluster. What is notable about this is that no distance metrics need to be computed and a dramatic speed-up is achieved. This is done for each cluster. The second stage takes all the orphans which are essentially the said "islands" and expands out by the minimal distance for each cluster. The end result is that all elements are assigned to reachable clusters and there are no islands.

A key to implementing agglomeration was to use linked lists to hold the front data. Standard Template Library (STL) Vectors are notoriously slow at adding/ subtracting data as compared to a linked list. This made at least an order of magnitude of difference. Further performance enhancements are planned by always maintaining the front data in a list of increasing distances. Again a STL vector would only hinder the implementation of this type of enhancement.

Agglomeration is used in cfdMine such that it is a post-processing technique on the already-clustered data. Ideally agglomeration would be used at every clustering iteration instead of k-means, but this is far too computationally intensive for the current effort. Again, the reason agglomeration is desirable is that it incorporates connectivity information into the clustering and preserves more information than pure k-means.

Surface Tension Algorithm

An alternative to agglomeration worth mentioning that was tried unsuccessfully to eliminate islands is termed by this author “surface tension”. The algorithm is very simple: loop through all elements in the domain and assign each to the cluster which has the most shared faces around the cell. In other words, if the cell is a protrusion into a surrounding cluster, it will be absorbed into the cluster into which it protrudes. While this is slightly effective, it must be rerun a number of times and fails to eliminate large islands. It was abandoned in favor of agglomeration after implementation.

Flow Network Post Processing

Once the clusters have been developed by a combination of k-means and agglomeration, the quantity of flow between the clusters (the flux) must be found. This is done by simply looping through all the volume elements in the CFD solution. Each volume element has multiple faces which maps either to a boundary condition (example fluid inlet), a wall, or a neighboring volume element. Where a face links two volume elements in differing clusters, the fluid flow is summed at that boundary. When all the flows between clusters are summed, the total flow between clusters can be used to make the final simplified fluid nodal network since flow in/out of each note is tallied. Both Figure 31 and Figure 35 shows an example of what this network looks like for real domains.

Validation of Conservation of Mass

The flow network algorithm for subvoluming is based on volume flows. The present method expands on that technique and is also based on a volume flow network. The flow across interfaces is calculated by Equation 11 by summing each face that is on an interface. Since StarCCM+ is a cell based solver, there is no velocity data at the cell faces. Thus the average velocity between cell centers which share the face must be taken. This is done by taking the face normal dot product with each connected cell’s velocity vector as shown:

Equation 11: Interface Flux

$$interface\ flux = \sum_{1}^{N_{faces}} a \cdot \frac{\vec{n} \cdot \vec{v}_1 + \vec{n} \cdot \vec{v}_2}{2}$$

One validation exercise for cfdMine is to create a duct with multiple inlets/ outlets and then check to make sure the volume fluxes between the clusters match the original CFD run from StarCCM+. Two such cases were created of an exhaust manifold. The first case shown in Figure 35 is a converging manifold where there are multiple inlets and one outlet. The second case is not shown but is a diverging case where there is one inlet and multiple outlets. Both cases have similar results, but the results for Figure 35 are tabulated in Table 2. Note the flow exits at 0 and flowing from 8. The flux value .005798 m^3/s is maintained from 8 to 5. The flows from clusters 2 and 1 merge flowing into 8. This is all expected behavior showing the conservation of volume across interfaces. Any errors are simply numerical tolerance errors and are not cumulative along the flow path.

Table 2

Volume flux linkage between clusters from cfdMine versus actual values from StarCCM+ for a converging manifold test case. The flow was incompressible in StarCCM+.

From Cluster	To Cluster	cfdMine m^3/s	StarCCM+ m^3/s	Error
8	0	0.005785	0.005798	0.23%
1	5	0.004345	0.004353	0.19%
6	1	0.002899	0.002908	0.33%
7	1	0.001445	0.001447	0.12%
2	5	0.001442	0.001445	0.19%
4	3	0.001447	0.001448	0.05%
3	6	0.001447	0.001448	0.05%
5	8	0.005788	0.005798	0.17%
9	6	0.001446	0.001445	-0.08%

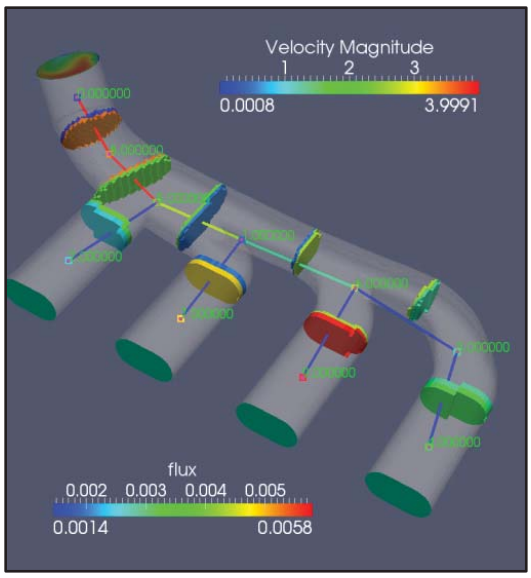

Figure 35: Converging manifold test case showing volume flux linkages and cluster interfaces.

While volume may be conserved, mass is not conserved unless the CFD solution and MuSES are both constant density. This was not known to be the case at the start of the investigation. A future exploration will be to move towards mass-weighted k-means and full conservation of mass which is a readily identifiable solution of error in the current clustering technique. This was found to be quite critical when validating the approach with a more complex problem where the cluster and flow interfaces tended to not be normal and larger numeric errors were seen. Also, up to 10% imbalances in inflows and outflows were found in both clustering and subvoluming approaches due to numerical errors at the interfaces.

Exporting Data Back into MuSES

Once the clusters and flow information have been processed, the data must be exported back into MuSES where it will be run in a transient condition. MuSES has a library called the TdfIO library which allows a programmer to setup most of the model parameters from a C++ code. This made it relatively easy to insert new clusters as MuSES “Fluid Nodes”. Fluid nodes are lumped mass parameters which can be any fluid, although for the current purpose we are assuming air. The nodes can advect fluid using “Advection Links.” The advection may be specified as either a volume or mass flow. CfdMine uses volume flow since this is intuitively the most physical parameter.

Fluid nodes are connected to surfaces in MuSES by CfdMine in an unusual way. There is an entity in the MuSES Solver called a “Thermal Link”. A thermal link is created from each cluster to the surface elements in MuSES which are contained in the cluster. There are several other ways to connect the fluid nodes, but this method is the lowest level hook into the actual solver. Note this is the first large-quantity use of thermal links in MuSES. The other methods considered were: 1) Write a user routine that imposes a flux on each element based on the difference between the wall temperature and the fluid node. 2) Use a new feature in MuSES which automatically assigns the closest fluid node to the wall. The latter option works well with an evenly spaced subvolume of points as typically implemented by Thermoanalytics. This does not work well when the cluster shapes are not spheres and vary in size.

The next problem encountered is what convection coefficient to use to link the fluid node and the wall elements. This dissertation proposes a new “localized” convection coefficient. A graphical depiction of this notion is provided in Figure 7. Typically engineers refer to a convection coefficient based on the bulk flow temperature. Computational fluids codes typically consider a convection coefficient to the element nearest the wall (in the log-log region of the boundary layer). The latter type is known as a fluid film temperature and convection coefficient. The technique used for clustering is to take the convection coefficient from the steady-state solution and multiply it by the area and the temperature difference. This yields a heat flux which can now be divided by the temperature difference from the cluster temperature to the wall. This is shown in Equation 9. Numerically, this unpins the wall temperature to the near-wall fluid film.

When remapping the convection data, Equation 9 produces negative convection coefficients in limited areas of the domain. This happens when the cluster temperature is higher than the fluid film temperature when the fluid film is cooling an element. There are two options to negate this: 1) Set that temperature to zero. 2) Take the absolute value. It will be assumed the best method is to take the absolute value, since the relationship will still be linear as the temperatures vary. Figure 36 shows on a test case where reversed convection may occur. The number of elements relative to the whole is very small, so the choice of how to handle these elements impacts the solution very minimally.

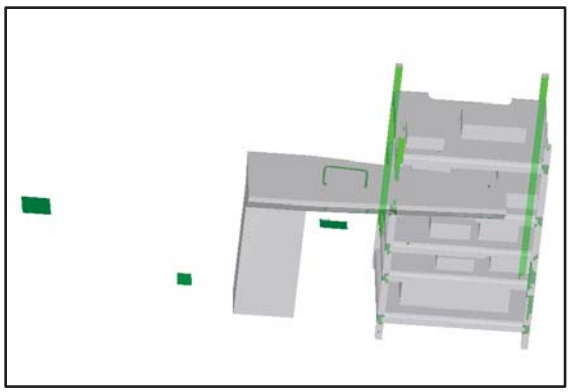

Figure 36: Example of where convection reversal occurs on a typical vehicle interior.

Additionally, one must keep in mind that intuitively, the convection phenomenon exchanges energy in both local and global areas. In moving to a reduced-order model, the major bulk flow is captured in the volume flux connections. The localized exchange of energy is captured by mapping temperatures to cluster temperatures.

The former state-of-the-art subvoluming technique which evenly divides the domain into subvolumes simply assumed a convection coefficient of 15 w/m-k at the walls (Shah et al. 2008; Curran et al. 2010; Han et al. 2010; Hepokoski et al. 2010). This is actually a fairly reasonable assumption for typical vehicle interiors since the wall temperatures really do not affect the air temperatures much. However, localized regions will have streams with higher values. Also, assuming a value of 15 w/m-k is not mechanistically based.

Another decision that was made is to only create advection links in one direction between fluid nodes. There may or may not be an advantage to tracking the mixing between clusters, but for proof of concept at this time, only unidirectional links are calculated, which is the net of the fluid flux between two clusters. Future work may look at the importance of bidirectional exchange between clusters.

Equipment Clusters and Fixed Clusters

A very important application of thermal analysis for the government is to assure equipment in the vehicle does not overheat. Modern military vehicles have dozens of electronics boxes. Additionally, the rapid fielding of MRAP (mine resistant ambush protected) vehicles for Iraq and Afghanistan resulted in a fleet with substandard air conditioning. Most of these vehicles have HVAC systems which can only cool the air at the vent outlet to 110°F on a 130°F day. It is

difficult to track the local temperatures around the equipment in the CFD domain. It is also difficult to physically measure the temperatures around equipment since a thermocouple temperature is a point reading of a distribution. The former subvoluming technique (Shah et al. 2008; Curran et al. 2010; Han et al. 2010; Hepokoski et al. 2010) is only able to split the domain at specific planes. This means that it cannot create subvolumes around each electronics box when there are numerous boxes.

It is purposed (and shown) that clustering about points fixed at the centroid of equipment will automatically find volumes around the equipment that track surrounding temperatures. Specifying fixed nodes is considered a partially supervised learning algorithms since the user supplies *a priori* information on where to fix cluster locations. Since the equipment is not spherical in shape, it may be difficult to fit a cluster around it. Additionally some of the equipment has very high aspect ratios which would make the distance from the centroid even more difficult to fit. The ability to compute the distance from a bounding box was incorporated into cfdMine which performs much better. The difference in bounding box and Euclidean distance from the centroid is shown in Figure 37 and Figure 38.

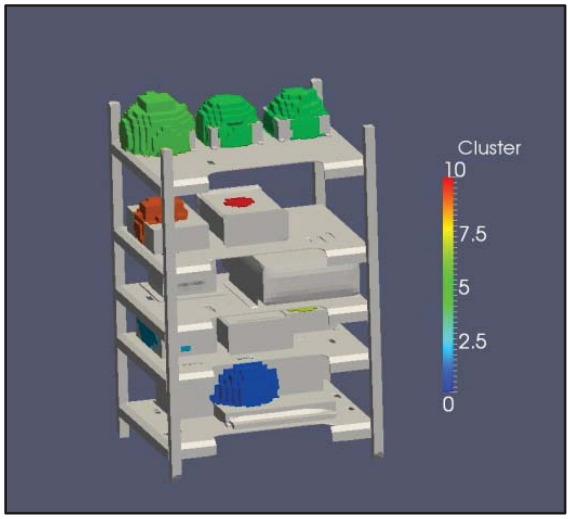

Figure 37: Clusters around equipment with the geometric distance calculated from their centroids.

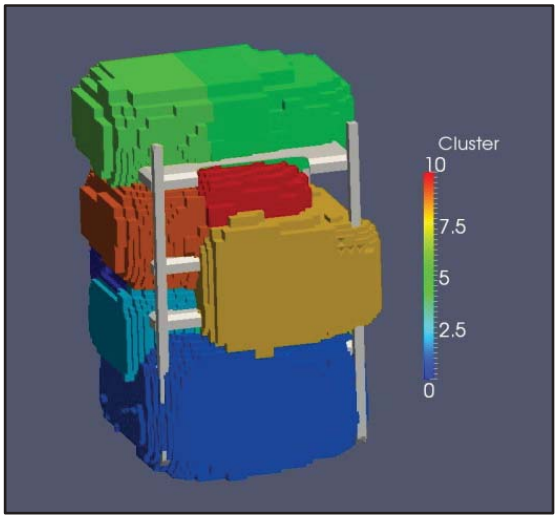

Figure 38: Clusters around equipment with the geometric distance calculated from their bounding boxes. The weights are the same as in the prior figure.

A user may load the bounding boxes from the equipment from either a MuSES file or from the Ensight CFD results. Section 10 of the cfdMine GUI in Figure 28 is used to assign heavier weights to the distance metric for clusters around equipment. Equation 12 is used to calculate the distance of each point from the bounding box.

Equation 12: Distance of a point in the domain to a bounding box.

$$x_{bbox\_min} = (x < x_{min})\,?\,x_{min} : (x > x_{max})\,?\,x_{max} : x$$
$$y_{bbox\_min} = (y < y_{min})\,?\,y_{min} : (y > y_{max})\,?\,y_{max} : y$$
$$z_{bbox\_min} = (z < z_{min})\,?\,z_{min} : (z > z_{max})\,?\,z_{max} : z$$

$$dist_{min} = \sqrt{(x - x_{bbox\_min})^2} + \sqrt{(y - y_{bbox\_min})^2} + \sqrt{(z - z_{bbox\_min})^2}$$

Figure 39, Figure 40, and Figure 41 show the end result of using this technique for clustering. The results are very favorable when using the bounding box distances.

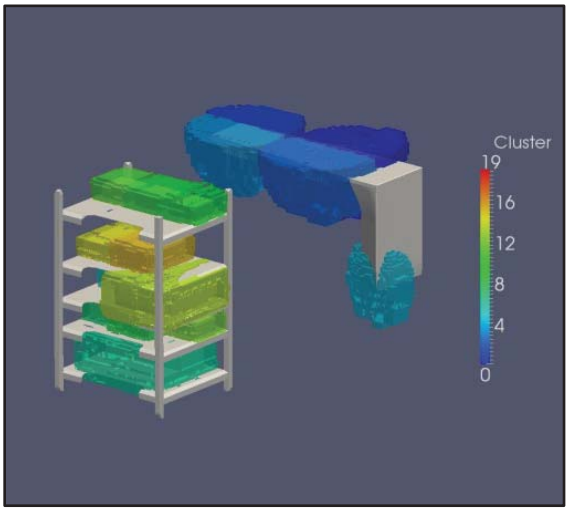

Figure 39: Example cluster results around a simple duct and equipment in a rack. Note additionally fixed clusters were placed at each inlet and outlet on the duct.

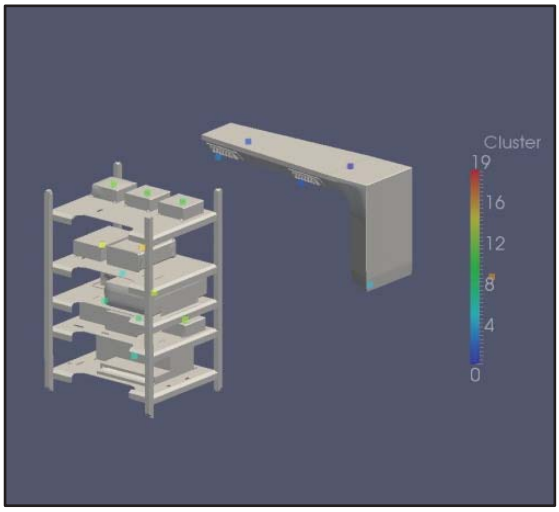

Figure 40: Shown are the cluster center locations.

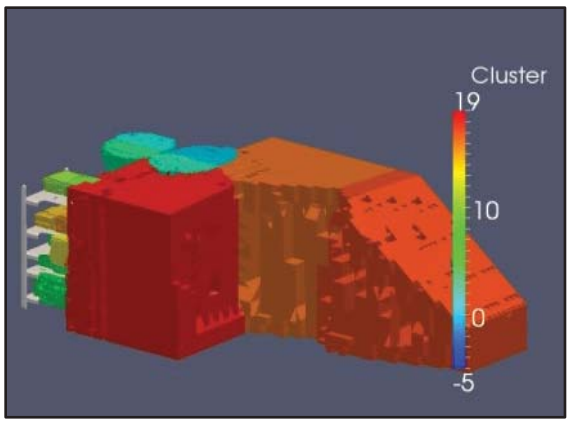

Figure 41: Shown is the same case as in the preceding figures is shown, with the addition of the larger clusters in the rest of the crew area.

Other Fixed Clusters

Inlet and outlets are an important feature that should be explicitly captured by the reduced order network. This can be done in cfdMine by specifying a fixed x,y,z coordinate node at the inlet/ outlet. This is important because these nodes in the flow domain represent sources and aren't always automatically captured. This also allows k-means to find a faster answer if some nodes are fixed and there is *a priori* knowledge of where those nodes should be placed. Sample images are shown in Figure 42 and Figure 43. The k-means solution may be sped up by caching the distances between fixed points (or bounding boxes) and domain points.

A problem that is experienced in fixing cluster centers at inlets and outlets will loosely be termed a "bifurcated cluster" as show in Figure 42. Since the distance vector used in cfdMine includes temperature, velocity magnitude, and velocity direction, the red cluster provides a minima point even though it is detached from the center. This may be alleviated by increasing the geometric weight of the cluster. However, k-means is known to exhibit this behavior in pure geometric instances as well (Bin Zhang 2001). Also, it is recommended that the distance not be taken from the bounding box (i.e. boundary plane), but rather from the center point. This is done by unchecking the bounding box button on the user interface of cfdMine. Since it is a major goal of using clustering as opposed to subvoluming to better capture the flow field, the end user should pay very careful attention to fixed clusters.

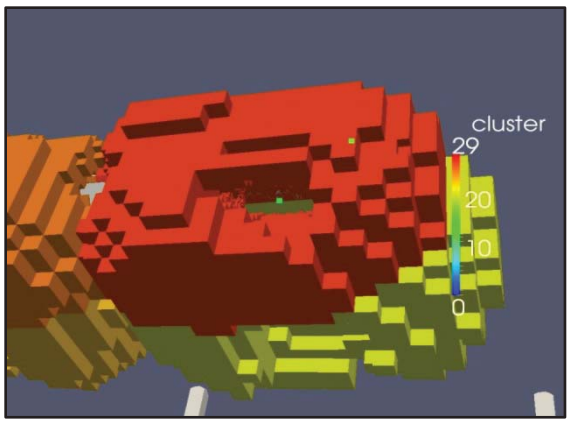

Figure 42: Bifurcated cluster. Note the center is hollow as the global error is at a minimum when the outlet stream which coincides with the cluster center is linked to a different cluster.

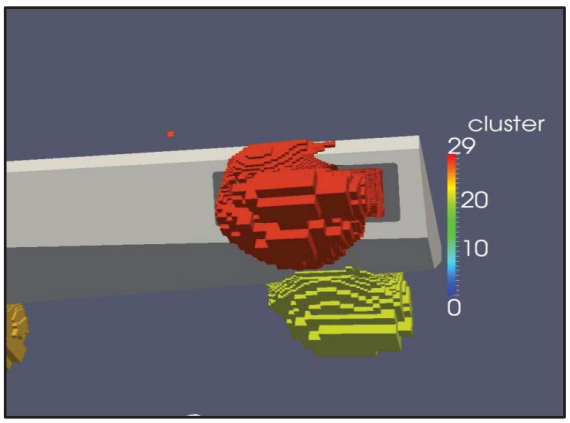

Figure 43: A careful weighting scheme better captures same outlet boundary as shown in the preceding figure.

Generally selection of the distance weights to use for inlets and outlets requires several attempts by the user. The user performs clustering, inspects the results, and as necessary, repetitively adjusts the weighting parameters. This can be a tedious process because basic k-means uses a spherical distance (Euclidean) and the resulting clusters tend to be spherical regardless of the actual flow shape. The best practice is to adjust the geometry weighting in isolation. If the geometry weight is one, increasing it to two will generate a cluster half the size of the original. It is recommended to find the geometry size that encloses the opening plus some margin and then apply a slight weighting to the velocity and temperature weights. This will elongate the inlet/ outlet cluster.

Another note about ducts and the domain in general is that areas of truly distinct flows should be partitioned off during the CFD solution and mined separately. It is difficult to converge the k-means solution where element associativity switches between the inside of a duct and the flow domain while iterating. This is a trivial extra step in the CFD code and again, helps embed the physicality of the domain into the clustered solution.

In order to better extract the jet shape and eliminate the “bifurcation” problem, the selectable ability to fix the variable parameters (not just the x,y,z) coordinates is also desirable. This was added to the second beta version of cfdMine and the results are shown in Figure 44. Additionally the image in Figure 44 portrays using z-normalization of the input data, where Figure 42 and Figure 43 normalized the data based on maximum and minimum values for each variable. Note

how much better the clustering algorithm is able to track the jet through the domain with fixed variable values.

Finding the correct fixed temperature and velocity values is somewhat dependent on trial and error. The best values are not the temperature of the fluid interface since the fluid mixes quickly with the surroundings. A mean value of sorts give much better performance. For example if the outlet temperature is 65 degrees and the surrounding air is 80 degrees, the average “jet” temperature for best clustering may be 70 degrees.

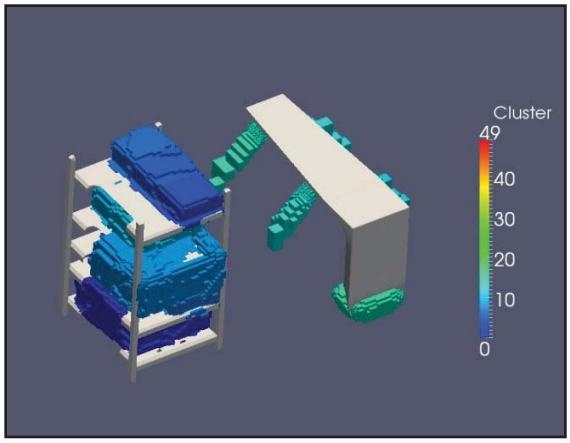

Figure 44: Example of clustering with fixed variable values at the inlet and outlets of the duct. Also the data was z-normalized for the clusters which provides better separation.

Numerical Nature of K-Means

A number of trials were run in cfdMine with different weighting parameters and relaxation factors. The algorithm always converges unless bad data is presented. Bad data is usually in the form of a CFD input file where there are no real thermal boundaries and thus the cells very only a small amount in temperature. Since the input vectors are all normalized, the minor temperature differences are magnified and the algorithm does not converge. No other situation has been encountered where the algorithm does not converge. This is easily negated by setting the temperature weight to zero. Typical convergence behavior is shown in Figure 45 and Figure 46. There is a very quick convergence initially and then the cluster centroids jostle around much slower. The convergence plot suggests a two stage search for global minima might be very beneficial. For example, the ten most promising trials after fifty iterations could be fully iterated to convergence. Next those could be converged completely and the best solution chosen. The premise is that after 50 iterations, most of the convergence has occurred and computational effort can be applied to just the most promising solutions.

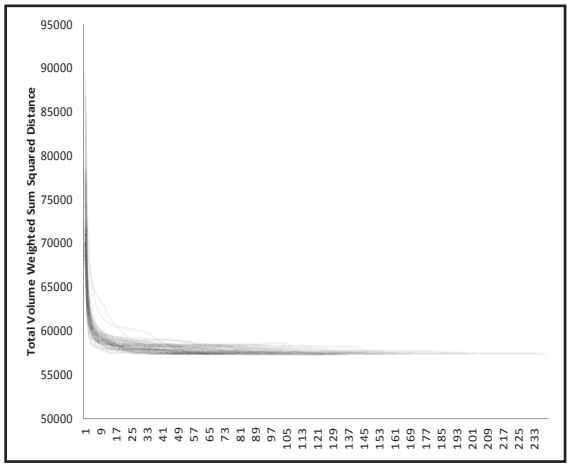

Figure 45: Typical k-means convergence plot for 50 random start trials on a vehicle duct geometry. The lines are translucent so the probability of quick convergence can be assessed.

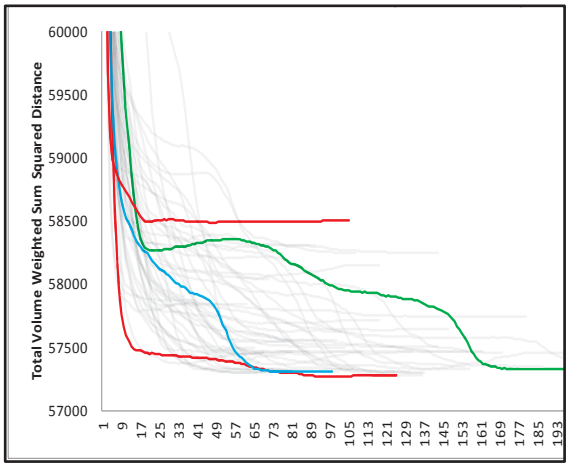

Figure 46: "Zoomed-in" plot of 50 random-start k-means trials. The lines are translucent so the probability of quick convergence can be assessed. The red lines are the best and worst solutions. The green line is a typical "average" convergence.

A relaxation factor was incorporated to cfdMine to assess the potential benefit of limiting how much each cluster might move per iteration. Typically relaxation factors are applied for numerical stability, but it was thought it might show benefit toward finding better global answers for the k-

means algorithm. Figure 47 below shows the result of running a typical internal vehicle HVAC simulation. There is little benefit to using a large relaxation factor and potentially a slight benefit to using a slight relaxation of .8 or .9. The relaxation factor greatly reduces convergence speed and increases computational effort required in seeking a global minima.

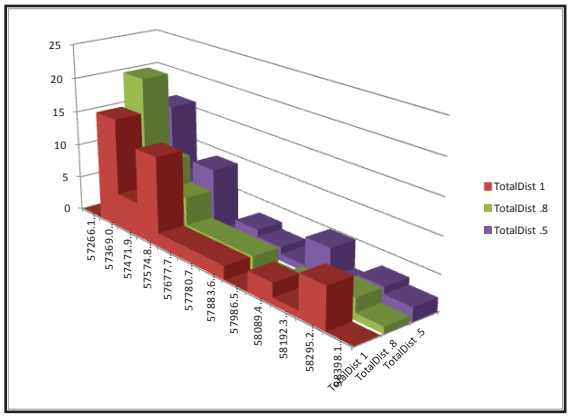

Figure 47: Histogram of final sum-squared distances. This is shown over 50 attempts for three relaxation factors. Columns to the left represent the best performance.

cfdMine Mahalanobis Exploration

As part of this research a number of alternative methods were implemented to determine feasibility and the best mode of implementation. Fuzzy clustering (Bezdek 1981) was implemented with volume weighting. K-Harmonic Means (B Zhang et al. 2003) was also implemented, although a formulation was not generated for volume weighting. Finally k-means with volume weighting and the Mahalanobis metric was implemented. The evaluation of these algorithms for their suitability was fairly subjective since comparison of a Euclidean metric with a Mahalanobis distance isn't directly possible. The test case shown in Figure 10 was used extensively since it is a fully 3d flow with a stream, but it is fairly symmetric.

Since k-harmonic means and fuzzy clustering only address the tendency of k-means to converge to local minima, they did not vastly improve the solution relative to the increase in computational effort required. In contrast, a fairly large performance gain was seen when the Mahalanobis distance metric was combined with k-means. As mentioned in the literature review, it is difficult to use the Mahalanobis metric directly as the covariance matrix isn't inherently stable.

A new algorithm was developed and named “cfdMine Mahalanobis” which works according to the following process and is based on stabilizing the distance metric in

Equation 8:

1) Pick random cluster centers from the domain and assign elements from the CFD domain randomly to the various clusters.
2) Z-normalize all the data in the domain. The x,y,z normalization is done by averaging the standard deviation.
3) Compute the covariance matrix and invert (OR fix at an identity matrix subject to n iterations which forces a Euclidean start).
4) Divide the x,y,z 3x3 inverse covariance matrix by the maximum value and multiply by the weight of importance for physical distance. For temperature, velocity magnitude, and velocity direction, divide the associated rows and columns by the diagonal value. This step weights the inverse covariance matrix.
5) Blend the inverse covariance matrix with an identity matrix with variable weights on the diagonal.
6) Compute the Mahalanobis Distance from each domain point to the covariance matrix.
7) Assign each point the closest cluster.
8) Go to step 3 until converged.

The algorithm above has several steps which were experimentally discovered to stabilize the Mahalanobis clustering. Without these steps, the clustering tends to produce very large and very small clusters and fall even more fatefully into local minima than standard k-means. It makes sense that K-Means Mahalanobis (KMM) would be more likely to fall into local minima because there are more available degrees of freedom (a six-by-six covariance matrix in addition to the eight values that define the cluster centers).

The specific stabilization techniques employed show up in steps 3 and 5 above. Both steps exploit the fact that the Mahalanobis Distance becomes the Euclidean Distance when the inverse covariance matrix is set to be an identity matrix. Step 3 uses this to avoid local minima and improve symmetry by converging the solution to some level or number of iterations using the Euclidean distance prior to allowing the Mahalanobis Distance to be computed. Step 5 blends the inverse covariance matrix with an identity matrix, so that the user may specify how “Euclidean” the shapes of the clusters may be. Recall an identity matrix for the Mahalanobis

distance is a special case and exactly equals the Euclidean distance. It was observed that a 75% Euclidean and 15% Mahalanobis blend worked best.

Experimentally it was found best to completely converge the solution (including random restarts) as purely Euclidean clustering. Then after that, a separate Mahalanobis clustering was performed based on the best Euclidean result. Further, the Mahalanobis clustering was found to produce better results if the inverse covariance matrix was only recomputed approximately every five or ten iterations instead of at every iteration. By allowing the partitioning step to converge completely before recomputing a new inverse covariance matrix, better "*a priori*" knowledge was available for finding the distribution within the cluster.

A comparison of typical Euclidean and Mahalanobis is shown in Figure 48 through Figure 53. Note in comparing Figure 48 to Figure 49 that the Mahalanobis distance is much better suited to finding jets of air and automatically found a cluster starting at the inlet and going all the way to the outlet. This was unachievable using only the Euclidean distance. Additionally the Mahalanobis distance was able to find a concentric cluster around the jet which is more accurate.

Another observation is that "non-jet" portions of the domain have a harder time finding a symmetric convergence since there are more degrees of freedom. This is illustrated by comparing Figure 50 to Figure 51. Finally, Figure 52 and Figure 53 show the effectiveness of using Mahalanobis on a more complex case. It is harder to separate the jet from the equipment clusters and pure Mahalanobis weighting is undesirable. While these images are point snapshots of convergence, they are very typical. Pure Mahalanobis clustering is definitely not effective and providing tight clustering.

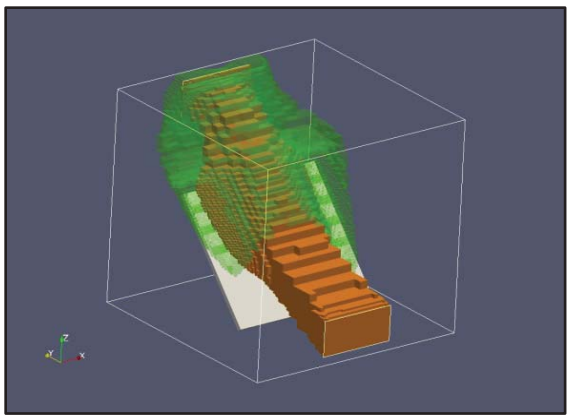

Figure 48: Test Case 1 Mahalanobis jet clusters. Shown are 15 clusters, 75% Euclidean and 15% Mahalanobis.

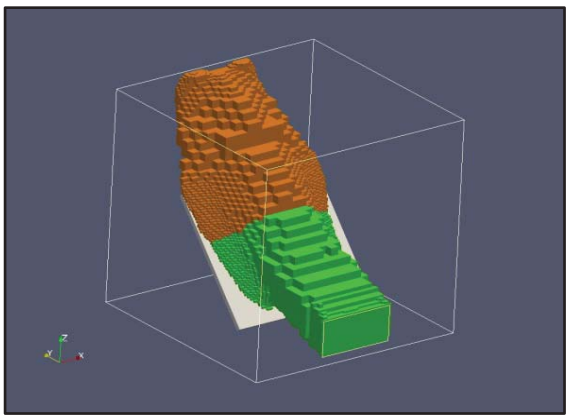

Figure 49: Test Case 1 pure Euclidean jet clusters. Shown are 15 clusters, 100% Euclidean and 0% Mahalanobis.

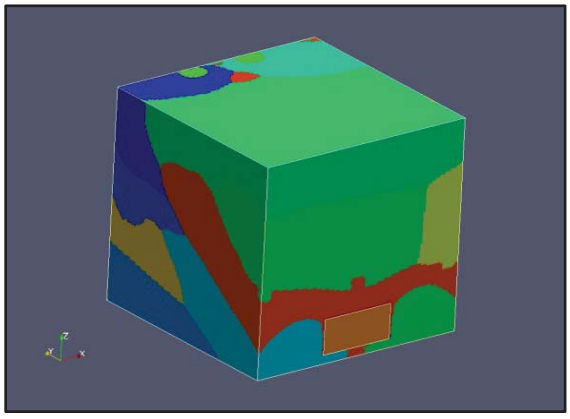

Figure 50: Test Case 1 Mahalanobis all clusters shown. Shown are 15 clusters, 75% Euclidean and 15% Mahalanobis.

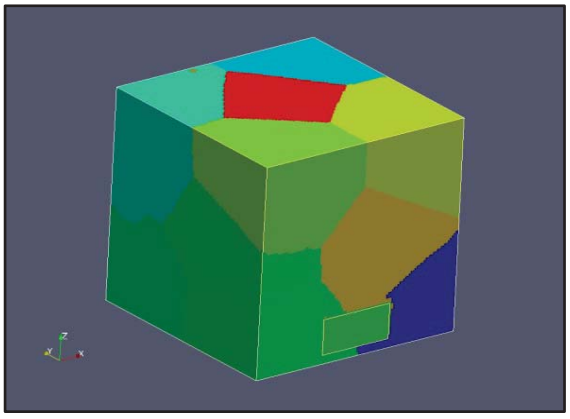

Figure 51: Test Case 1 pure Euclidean all clusters shown. Shown are 15 clusters, 100% Euclidean and 0% Mahalanobis.

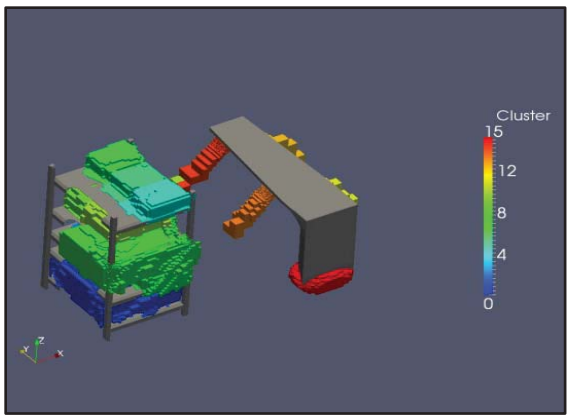

Figure 52: Duct electronics test case Mahalanobis Fixed clusters shown. Shown is 75% Euclidean and 15% Mahalanobis.

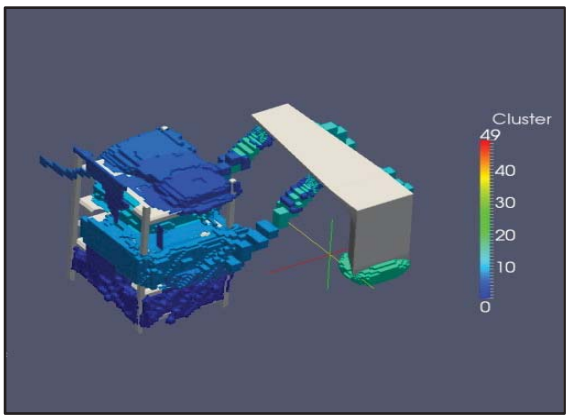

Figure 53: Duct electronics test case Mahalanobis fixed clusters Shown. Shown is 0% Euclidean and 100% Mahalanobis.

Numerical Nature of K-Means Mahalanobis

A number of trials were run with different weighting parameters. The KMM algorithm always converges and falls quickly into local minima unless stabilization efforts are undertaken. Typical convergence behavior is shown in Figure 45 and Figure 46 below. There is a very quick convergence initially and then the cluster centroids jostle around much slower. The convergence plot suggests a two stage search for a global minimum might be very beneficial and this was tried and verified that it works quite well. The 10 most promising trials after 50 iterations could be chosen out of 100. It was found better to specify the number of iterations to converge and then pick the best converged initial clustering to completely converge.

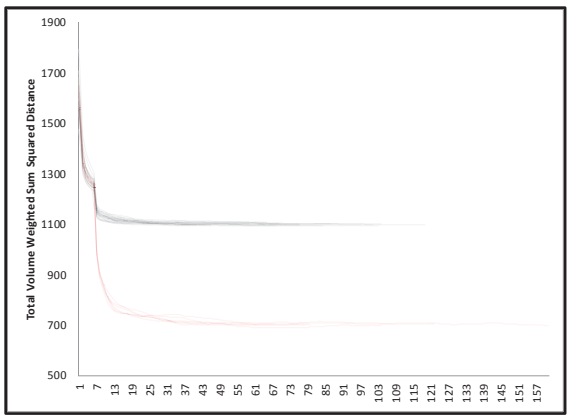

Figure 54: Typical k-means convergence plot for 33 random start trials on a vehicle duct geometry. The lines are translucent. The black lines show 25 runs with a 50% Mahalanobis/Euclidean blend versus the 8 red lines which show 75% Mahalanobis. Note the first 5 iterations are pure Euclidean.

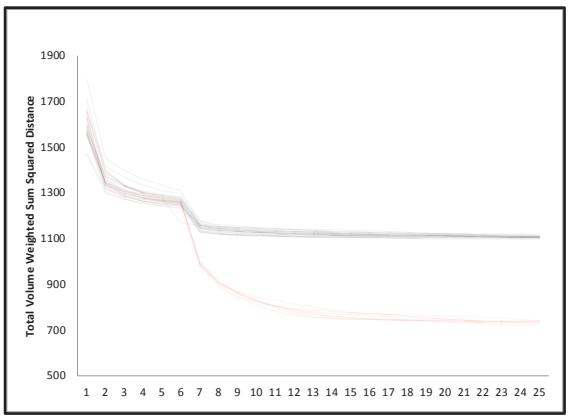

Figure 55: This is a "zoomed-in" plot of typical k-means convergence plot for 33 random start trials on a vehicle duct geometry. The lines are translucent. The black lines show 25 runs with a 50% Mahalanobis/Euclidean blend versus the 8 red lines which show 75% Mahalanobis. Note the first 5 iterations are pure Euclidean.

Validation: Comparison of Transient CFD, MuSES + Subvoluming, and MuSES + Clustering

A CFD test case representing a typical vehicle analysis was created to validate the new clustering approach and also to compare the subvoluming technique. The model is simply a hull structure with a rack of equipment and a cooling duct with jets of air blowing on the equipment. An image of the model is shown in Figure 56. A labeled close-up of the equipment rack is shown in Figure 57. The final validation model is 2 million cells. Flow through the duct is 550 CFM. Each piece of equipment has 100 watts of heat applied evenly to each component's rear. Finally, the evaporator maintains 9 degrees of temperature drop from the inlet temperature. The initial condition for all fluids and walls is they are set explicitly to 150°F. The surrounding bounding environment is maintained at a constant 130°F.

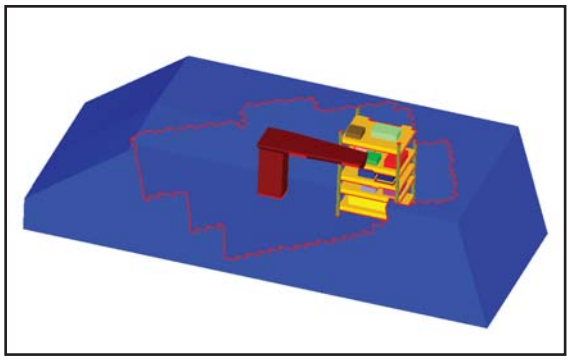

Figure 56: Final validation model. The red edge highlights a cutaway view of the duct and rack. There are four outlets on the duct and the air intake is on the bottom.

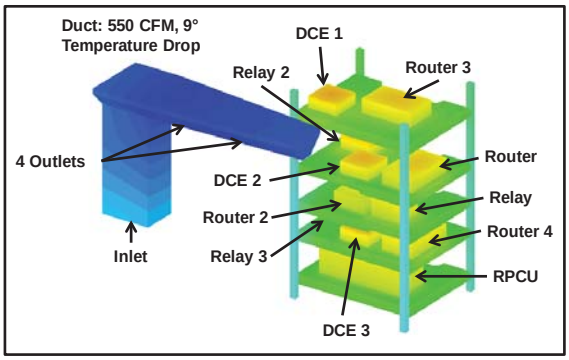

Figure 57: Final validation model equipment close-up.

The “known standard” model was run fully within StarCCM+ transiently. This was done on 32 cores and took approximately 5 days of runtime. A lot of that time is due to computing radiation which is not typically done in the CFD code, but the model is also 2 million cells compared to the 15 million cell models TARDEC often runs. The full conjugate heat transfer was run within StarCCM+, including computing the wall temperatures using solid volume mesh. Again, this is not how the CFD model is typically run in a production environment, but a transient validation model was run fully in CFD so that a direct comparison could be made to the subvolume and cluster-based transient MuSES models later.

In order to validate the MuSES-based methods, identical models were created using MuSES-with-subvoluming and MuSES-with-clustering. A couple steady-state CFD and MuSES model is the starting point for both subvoluming and the new clustering technique (reference the process diagram in Figure 12). The steady-state CFD flow domain was manually iterated with a matching MuSES model until no significant changes in either domain were noted. The coupling is done by exporting fluid film temperatures and convection coefficients from StarCCM+ into MuSES . MuSES then solves the conjugate heat transfer on the walls and exports the wall temperatures back into StarCCM+. The MuSES surface mesh is exactly matched to the CFD mesh for the validation case. The MuSES model boundary conditions were matched precisely to the fully transient StarCCM+ model. Even the material properties for aluminum were matched between the two models.

The MuSES subvolume model was created from the steady-state CFD solution discussed in the prior paragraph and is shown in Figure 54 and Figure 58. Seventy two fluid nodes were used and an attempt was made to match the node spacing to the shelves in the best manner possible. The nodes that appear to protrude past the volume of the domain actually have small corners that extend into the domain. A user routine was written that exactly enforces a 9 degree temperature delta for air going from the duct inlet into the outlets. Note the closest fluid nodes are assigned to this evaporator link manually. There is a new feature in MuSES which associates elements to the nearest fluid node and that is the option used for the subvolume model to associate convection to the walls at a fixed rate of 15 w/m²k. There is a concern about the non-physicality of having control volumes cut across the shelves in a MuSES model as illustrating in Figure 59. The clustering technique does a little better job, but does not entirely eliminate this either.

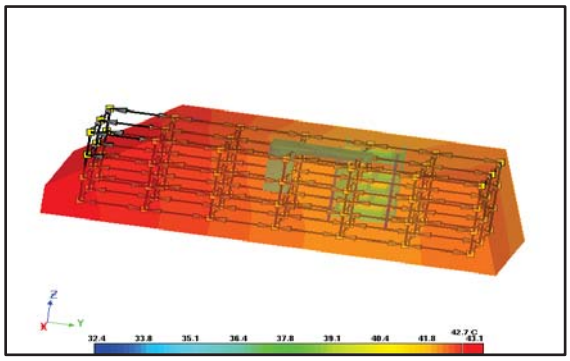

Figure 58: Subvolume nodal arrangement.

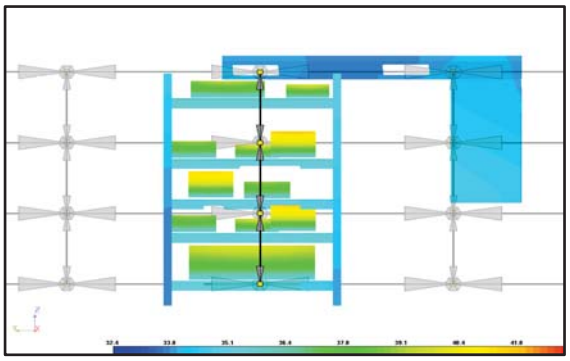

Figure 59: Close-up of the subvolume duct. Note the volumes cut complete across the shelves. Note there is multiple equipment associated with each node.

One minor problem identified in the subvoluming is that due to the way the macro cuts the planes, the edges of the subvolumes are jagged. It is suspected this causes a slight loss in the conservation of volume (and mass). This contributes slightly to the inaccuracy of the technique. The example shown in Figure 60 illustrates a typical imbalance. Note that the return into the duct is fully enclosed by the pictured subvolume and pulls in 550 CFM (CFD boundary condition). However, when the net in/out advection for the subvolume are calculated, a net in of 625 CFM is found. The imbalance caused by the edges may be minimized by refining the CFD domain, although this is a computationally expensive proposition.

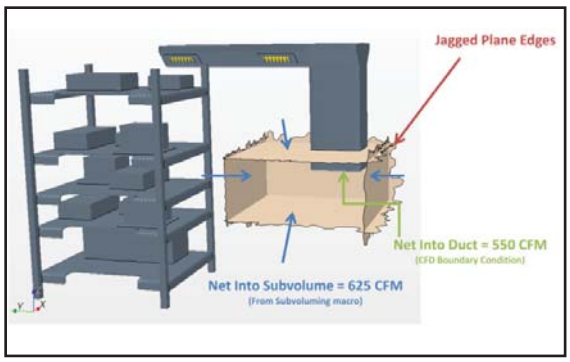

Figure 60: Subvoluming conservation of mass (volume) imbalance.

The exact clusters used for this validation are shown in Figure 61, Figure 62, and Figure 63. Figure 64 shows the clustering-based MuSES model and advection links from the MuSES GUI. The clustering MuSES model uses 35 nodes, 11 of which are fixed at the centroids of the equipment, 4 at each duct outlet, and one at the duct inlet. This is nearly half the nodes of the subvolume model. Note the clusters near the equipment better captures the flow around the shelves compared to subvoluming, but still allows non-physical communications of the flow across shelves.

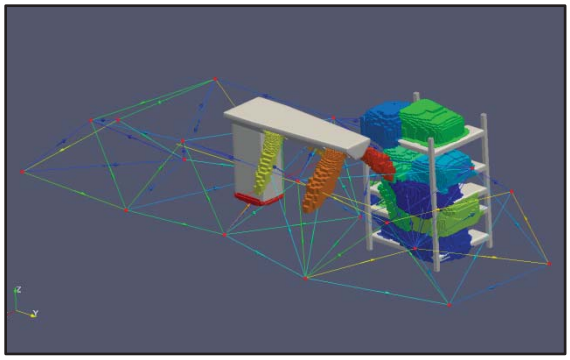

Figure 61: Clustering and nodal arrangement for validation case.

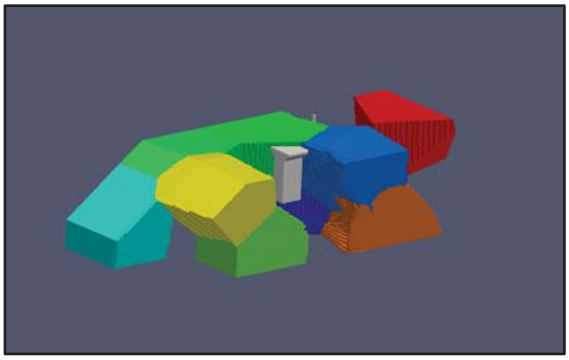

Figure 62: Validation case clusters 16-26.

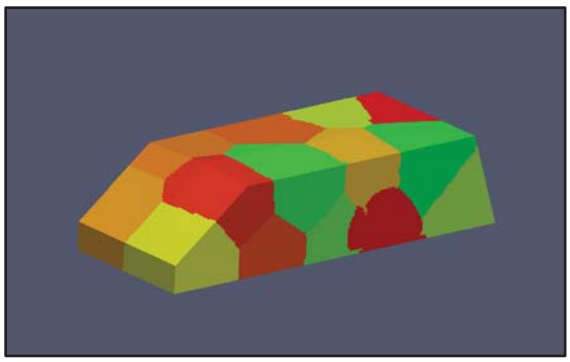

Figure 63: All validation case clusters visualized.

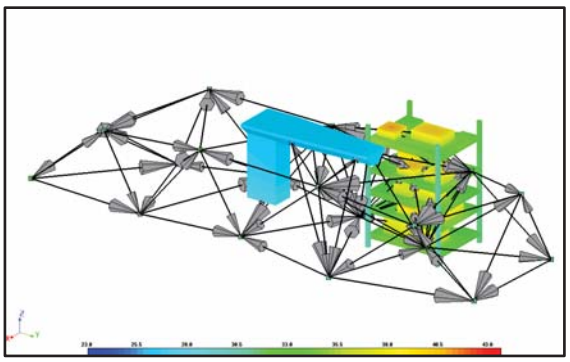

Figure 64: Clustering and advection links as visualized from MuSES.

A major unanticipated problem was encountered when running the validation MuSES models. The models showed odd behavior during convergence. For this reason, a steady-state clustering model was run to see how closely it aligned to the CFD steady-state. Instead of converging, the model slowly diverged at a constant rate. After much investigation it was determined that imbalance in the nodal network was the source of the problem. MuSES does not enforce a volume balance in its solver. Each node temperature is simply computed from the inflow streams and heat balance with no regard to downstream nodes which specify an advection link. A routine was added to force the nodal network for clustering to balance volume flows exactly. The following algorithm was implemented to fix the problem:

1) Start with the first cluster and assess all incoming and outgoing advection links. If there is a source outside the domain, enforce this in the calculation as well.
2) Divide the imbalance in half and apply half to the incoming and half to the outgoing flow to make them balance to zero (or reduce by a relaxation factor). Do this proportional to each advection link divided by the total incoming or outgoing flow.
3) Evaluate each cluster according to step 2 successively until each cluster exhibits no further imbalance.

Once the imbalance was fixed, the clustering algorithm had a much better convergence. The subvoluming MuSES model was also run in a steady-state condition and also diverged, but much more slowly. It was not felt the subvoluming model needed rebalancing for acceptable results and that was beyond this body of work to address. It is also noted that the air properties are temperature dependent and a volume flow based network will cause a slight imbalance due to

density changes (i.e. mass is not conserved even if volume is conserved). This should be addressed by moving to a mass-based clustering and resultant mass-based MuSES nodal network in the future. Figure 65 shows the density variation within the CFD domain and areas of high gradients will likely be energy sources or sinks in the flow network when a volume-based network is used.

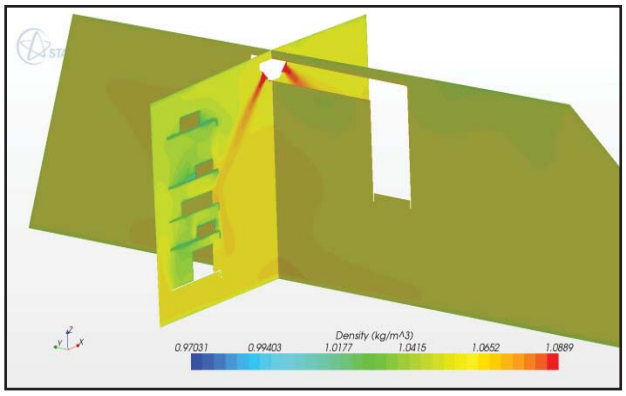

Figure 65: Plot of CFD domain densities. Areas of large gradients will be energy gain/loss areas.

The following figures are typical results from the validation study. The plots were randomly chosen to pick an equipment item from different rack shelves. The full set of data may be found in Appendix 2. Nearly all of the tracked fluid temperatures and surface temperatures more closely matched when clustering was used. It is interesting to note that the clustering method generally tracks the trend of the transient CFD with an offset. It is suspected this offset is the direct result of energy/mass loss where the jets of cool air leave the duct and mix with the surrounding air. The energy loss would be caused by mass imbalance since the code presently sets the network up as constant volume flows.

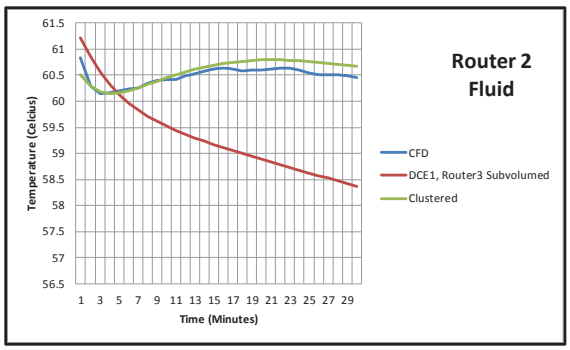

Figure 66: Validation results for the fluid temperature around "Router 2". Note in the plot legend that the subvoluming node encompasses several pieces of equipment.

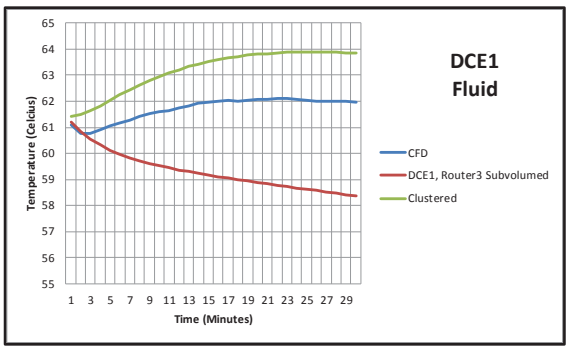

Figure 67: Validation results for the fluid temperature around "DCE 1".

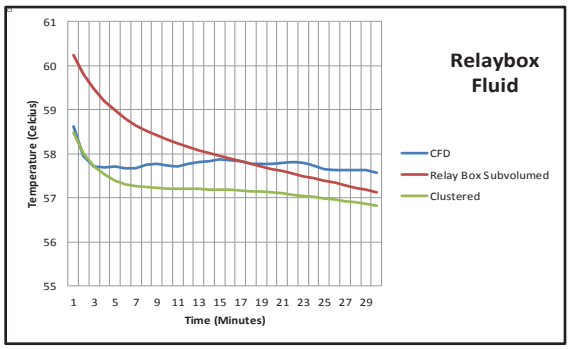

Figure 68: Validation results for the fluid temperature around "Relay Box".

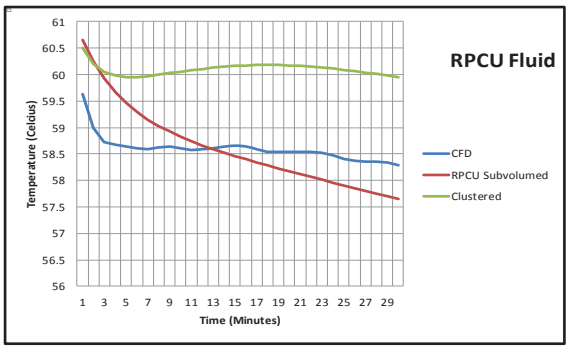

Figure 69: Validation results for the fluid temperature around "RPCU". This is the worst performance seen for clustering, but the trend matches for clustering with an offset.

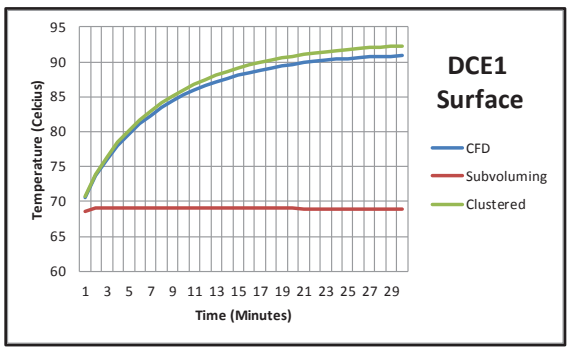

Figure 70: Validation results for the fluid temperature around "DCE1". This is very representative of all the surface temperature plots.

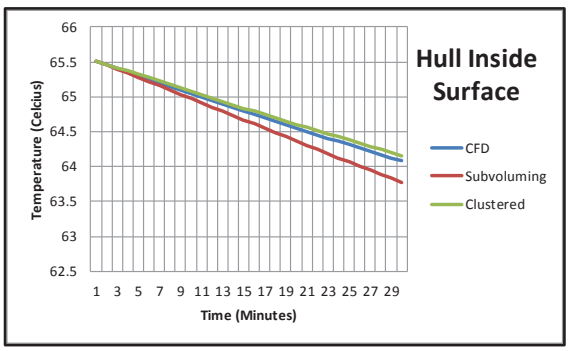

Figure 71: Validation results comparing the average inside temperature of the hull.

The clustering method is more accurate than the subvoluming method, except for in specific locations (i.e. the RPCU). The plot of the hull temperature in Figure 71 clearly demonstrates the problem with using a fixed convection value of 15 in all locations. This high convection causes too much coupling between the hull and the fluids. Another illustration of the problem with applying a constant convection coefficient is illustrated by comparing Figure 72 and Figure 74 to Figure 73 and Figure 75.

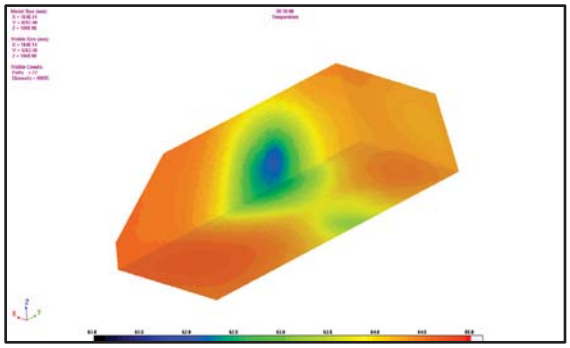

Figure 72: Clustering technique validation surface temperatures on the hull exterior at the end of the 30 minute simulation.

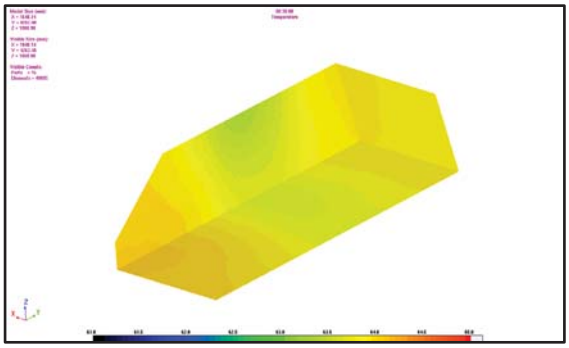

Figure 73: Subvoluming technique validation surface temperatures on the hull exterior at the end of the 30 minute simulation.

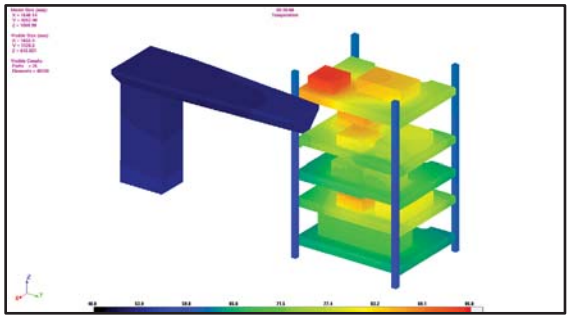

Figure 74: Clustering technique validation surface temperatures on the equipment rack at the end of the 30 minute simulation.

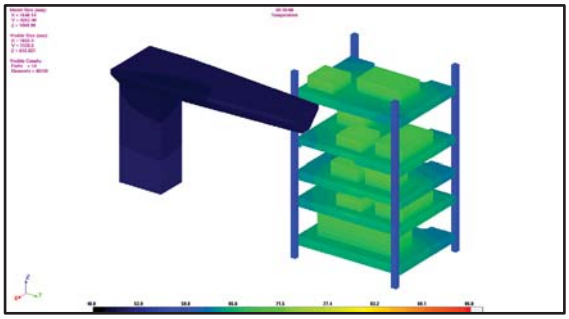

Figure 75: Subvoluming technique validation surface temperatures on the equipment rack at the end of the 30 minute simulation.

Conclusions

New Contributions to the Field

1) First to implement clustering to simplify a CFD network into a simplified network.
2) First known volume-weighted clustering of a CFD domain particularly to reduce it to a reduced-order network.
3) Showed value of using agglomeration as a post-processing step was used to make compact clusters.
4) Showed the first known remapping of fluid-film convection coefficients to localized nodes.
5) Found a practical way to measure the temperatures around equipment in a CFD cooldown simulation by fixing clusters around equipment.
6) Proposed new methods to stabilize the Mahalanobis distance and make it usable.
7) Demonstrated several new ways to visualize flow domains.
8) Proved that it is absolutely critical to conserve mass within fluid network.

There were a number of new contributions made to the field. The main contribution is being the first to demonstrate that clustering is very effective for creating a simplified nodal thermal network. As a subset of the overall contribution, a more minor contribution was made by using agglomeration as a post-processing step to generate compact clusters and eliminate “islands”. Further enhancing the overall technique, localized convection coefficients were remapped from the CFD to the fluid network and also helped to make the model more accurate. An additional contribution was to the idea of semi-supervised clustering technique to setup immobile clusters in regions of interest, such as around equipment to track equipment temperatures. The validation exercises demonstrated a decrease in solution time from a week running on 36 CPUS to four hours on one CPU with reasonable accuracy.

The second category of contributions was made on the side of data mining itself. There has been very little work done on clustering volumetric data. A unique way of stabilizing k-means clustering using a Mahalanobis volume-weighted distance metric has been proposed and implemented. This method starts with a Euclidean distance metric and switches to a weighted Mahalanobis metric that is then blended with the Euclidean in the inverse-covariance matrix. The blended inverse-covariance matrix is then updated only periodically. Visual inspection shows very high potential for separating data better, while avoiding dropping too quickly into local minima. This method is not recommended for use at this time as the clustering computational effort increases by an order of magnitude when measured in real-time. This algorithm may be more practical in the near future by parallelizing the algorithm.

Finally it has been showed that visualization of the structure of the flowfield is enhanced using clustering with connecting flux lines. The cluster interface plots are also very useful in looking at the structure of the domain.

The proposed objectives were all met: 1) A new technique based on clustering was shown to accurately cluster a steady-state CFD Domain to a transient simplified fluid network. 2) Localized convection coefficients were remapped successfully to the localized nodes in the fluid network. 3) It was shown to be possible to fix nodes about the bounding box of equipment at inflow/ outflow regions to enhance temperature tracking in those areas. 4) Several new flow visualization plots were developed. 5) K-means, agglomeration, k-means Mahalanobis, and k-harmonic means were all explored and implemented.

Overall Conclusions

Ultimately the new developed techniques implemented in cfdMine and cfdMine-Mahalanobis demonstrated a marked improvement over state-of-the-art (i.e. subvoluming). Most air temperatures were more accurate than subvoluming and overwhelming tracked the CFD-predicted temperature trends very accurately with a slight offset. It is suspected this offset may be fixed by moving toward a mass flow network.

The k-means algorithm when combined with volume weighting and parameter weighting is very well suited to automatically generating a simplified fluid nodal from a solved CFD solution. A moderate improvement could be made by moving toward Mahalanobis distance which does not generate spherical clusters, but Mahalanobis is also an order of magnitude more computationally intensive. Also it was found that initial data normalization using z-normalization is very valuable in helping extract quality clusters. It is also critical to provide weighting parameters for the variable of interest, which in this case were temperature, physical distance, velocity magnitude, and velocity direction.

Depending on the problem, there is a minimal number of nodes required to capture the relevant flow details. A higher cluster count allows the algorithm to more easily find the flow path, but increases the solution time significantly. Another surprising find was how nicely the volume weighting of the domain offset the biasing of the cluster centers toward high point density areas.

The biggest observable problem with using k-means is that it tends to converge to local minima. The simplest means of avoiding this is to simply rerun the entire algorithm multiple times from random initializations. It was found that simply starting with random initializations and clustering for a limited number of iterations and then finally completely converging the most promising solution was very practical and effective. Better solutions may be to 1) make very informed initial guesses (Vassilvitskii and Arthur 2007), or 2) use an enhanced version of k-mean such as k-harmonic means (B Zhang et al. 1999) or using a gradient descent method with simulated annealing (Qiu et al. 1994; Rose 1998).

Since there is more physical information available when clustering CFD domain data, it would be ideal to utilize a path distance instead of a “straight-line” Euclidean distance. The path distance, would be the distance a flow must travel around a blockage. The most common method for finding this type of distance is Dijkstra's algorithm (Dijkstra 1959). This algorithm is considered np-hard and to compute this multiple times for each cluster in a k-means context is immediately not possible. The next possible candidate for performing these computations, is the A* Algorithm (Hart et al. 1968). A* is an improvement on Dijkstra’s algorithm and is used extensively in

robotics path planning. A* uses heuristics to improve the calculation time for a distance computation and achieved $O(e^n)$ with a good heuristic. Again, from a practical standpoint, it is still too numerically intensive to implement A*, which operates very similar to an agglomeration method. There may be a post processing step which could exploit the A* algorithm.

The overall process of coupling to MuSES can achieve a “local convection coefficient” which is better than a bulk coefficient and also a coefficient tied to the fluid film temperature. A potential future investigation could easily look at how much of a difference the localized convection coefficients versus a fixed (such as the 15 used in subvoluming) may affect the solution.

Agglomeration is extremely useful as a post processing step after clustering with k-means in extracting quality clusters that do not have islands. The drawback to agglomeration is the numerical computation is fairly significant. It is possible to use the original k-means clusters and perform a very quick advancing front technique to find which elements are “islands” and then perform full agglomeration only on this small fraction of points (relative to the full CFD domain volume).

The use of fuzzy clustering, k-harmonic means, and k-means with a Mahalanobis distance are all beneficial in finding better clusters. All of these techniques, however, have the problem of being too computationally intensive. The basic k-means algorithm gives reasonable performance for the test cases investigated with a very low computational burden. For most situations it was also observed that finding a true optimal solution was less important than getting the correct weighting parameters. For these reasons the final version of cfdMine will be based on Euclidean distances and the k-means algorithm.

A very important result is that the use of a volume-flow based network does not conserve mass when used in the MuSES solver. Initially it was thought that the temperature fluxuations would be minimal and no problems would be encountered. MuSES uses temperature-dependent properties for air, which means that a volume network won’t conserve mass. Further, energy sources and sinks develop in the system and have thus-far prevented solving a MuSES model that actually converges in the steady-state case. These were majorly reduced by balancing the volume flow network but it is suspected these may be eliminated by moving toward actual conservation of mass.

Another finding is that the MuSES code should be modified to incorporate element based properties (such as the linking of an element to a fluid node) without the use of “thermal links”. Thermal links seems to work well in terms of hooking into the solver at a low level with no convergence or numerical problems. However, there is no way to visualize the association of

elements with the appropriate fluid node. Also there is no way to visualize the remapped convection coefficients.

Future Work

There are a lot of potential avenues for future work. The first planned modification to the existing algorithms is to move toward a mass-based fluid network. This should fix the offset of temperatures found in the validation study. It is intended to submit this body of work to the Journal of Heat Transfer after switching to a mass-based methodology.

A very promising method that should be investigated is to add a post processing step to the clustering which performs one iteration of clustering based on the A* algorithm to add real path length computations. This would prevent the clusters from allowing flow through walls and other non-physical phenomenon. This technique would supersede the use of agglomeration for the elimination of islands.

Another area for investigation is to investigate whether unidirectional flux linkages should be used or bidirectional. Presently unidirectional links were used, which should provide more numerical stability, but might miss the amount of mixing that occurs in the flow network. The subvoluming technique does use bidirectional links.

Another potentially promising investigation might be to enhance subvoluming by adding localized equipment control volumes. It is unknown how important capturing flow details may be versus having high resolutions in areas of interest. Additionally adding localized convection coefficients to the subvoluming technique would certainly enhance the accuracy.

Another question that has plagued the author and experts in the field is the nature of heat transfer when the solution is transient versus steady state. For example, an outlet stream which impinges on an object transiently moves around and does not maintain a fixed position. This is not necessarily captured when a steady state solution is run. The steady state solution may converge to a seemingly “averaged” stream position, but the average transient effect of the stream may be spread over a much wider area. Another place this occurs is in engine exhausts for infrared imagery. The image seen is not the average of all time, but an instant snapshot in time which moves around. Finding the best clusters from temporal data is very challenging and there is not much data in the literature on performing this task (Yang and Parthasarathy 2006). Additionally (Yang and Parthasarathy 2006) introduces a notion of tracking relationships between clusters (features) in space that may evolve in time using SOAPs (Spatial Object Association Patterns). The authors claim the ability to identify vortices and relationships in a flow down. This would again be a concept worth further research.

A better approach to the fundamental problem of finding the influence of convection on a vehicle or similar structure may be to mine the CFD domain for “convection view factors”. Expounding upon the idea that the more physics that can be compressed into a reduced-order technique is beneficial, the basic idea would be to do the following: 1) Solve a steady-state CFD solution. 2) Release particles from all wall elements and boundary elements (perform a particle trace). Next using turbulence parameters from the CFD code, account for the mixing and the fraction of volume of influence each wall-released particle has. Ultimately track the influence each wall element / boundary has on each other element/ boundary. The trick here is to determine a method to handle all the mixing appropriately. Essentially the CFD domain is held constant for flow conditions, but the energy equation is solved. There are many leaps in this description and the author has much work to do to determine a new technique to close the gaps.

Works Cited

Bezdek, J C. *Pattern Recognition with Fuzzy Objective Function Algoritms.* New York: Plenum Press, 1981.

Cormack, R M. "A Review of Cassification." *Journal of the Royal Statistical Society, Series A* 134, no. 3 (1971): 321-367.

Curran, Alan R, S D Peck, Tony J Schwenn, and Mark A Hepokoski. "Improving Cabin Thermal Comfort by Controlling Equivalent Temperature." *SAE Int. J. Aerosp.*, 2010: 263-267.

Dijkstra, E W. "A Note on Two Problems in Connexion with Graphs." *Numerische Mathematik*, 1959: 269-271.

Ding, Chris, and Xiaofeng He. "K-means Clustering via Principal Component Analysis." *Proceedings of the 21st International Conference on Machine Learning.* Banff, Canada, 2004.

Dunn, J C. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters." *Journal of Cybernetics* 3 (1973): 32-57.

Eigen Version 3 Documentation. Mar 9, 2012. http://eigen.tuxfamily.org/dox/index.html (accessed April 19, 2012).

Ensight Version 9 Users Manual. CEI Software, 2011.

Gath, and A B Geva. "Unsupervised Optimal Fuzzy Clustering." *IEEE Trans. Pattern Anal. Machine Intell.* 11 (1989): 773-781.

Gowda, K C, and E Diday. "Symbolic Clustering Using a New Dissimilarity Measure." *IEEE Trans. Systems, Man and Cybernetics* 22 (1992): 368-378.

Gustafson, D E, and W C Kessel. "Fuzzy Clustering with a Fuzzy Covariance Matrix." *Proc. IEEE Conf. Decision Contr.* San Diego, CA: IEEE Press, 1979. 761–766.

Hamerly, G, and C Elkan. "Alternatives to the K-Means Algorithm That Find Better Clusterings." *Proceeding of CIKM.* ACM, 2002. 600 - 607.

Han, T, K Chen, B Khalighi, A Curran, J Pryor, and M Hepokoski. "Assessment of Various Environmental Thermal Loads on Passenger Thermal Comfort." *SAE International Journal of Passenger Cars- Mechanical Systems*, 2010: 830-841.

Hart, P E, N J Nilsson, and B Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *IEEE Transactions on Systems Science and Cybernetics*, 1968: 100-107.

Haupt, S E, R Kunz, L Peltier, and J Dreyer. "Computational Fluid Dynamics Coupled with Thermal Impact Model for Building Design." *Journal of Computers* 5, no. 10 (Oct. 2010): 1552-1559.

Hepokoski, M A, A Curran, Mark Klein, Robert E Smith, and Vamshi Korivi. "Analysis of Soldier Effectiveness in a Mine Resistance Ambush Protected Vehicle." *NDIA Ground Vehicle Sytems Engineering and Technology Symposium.* Dearborn, MI, 2010.

Huang, D C, E Oker, S L Yang, and O Arici. "Computer Model for Automobile Climate Control System Simulation and Application." *International Journal of Thermodynamics*, 1999.

Jain, A K, M N Murty, and P J Flynn. "Data Clustering: A Review." *ACM Computing Surverys* 13, no. 3 (1999): 264-323.

Johnson, R A, and D W Wichern. *Applied Mu ltivariate Statistical Analysis, 4th Edition.* New Jersey: Prentice Hall, 1998.

Jolion, J M. "Robust Clustering with Applications in Computer Vision." *IEEE Trans. Pattern Anal. Machine Intell* 13, no. 8 (1991): 791-802.

Karypis, G, and V Kumar. *METIS-Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0.* 2011. ftp://ftp.cs.umn.edu/dept/users/kumar/metis/manual (accessed July 15, 2011).

kdtree - A Simple C Library for Working with KD-Trees. June 2009. http://code.google.com/p/kdtree/ (accessed January 12, 2012).

Kim, Dae-Won, Kwang H Lee, and Doheon Lee. "Detecting Clusters of Ddifferent Geometrical Shapes in Microarray Gene Expression Data." *Oxford Journals* 21, no. 9 (2004): 1927-1934.

Krishnapuram, R. "A Note on the Gustafson-Kessel and Adaptive Fuzzy Clustering Algorithm." *IEEE Transactions on Fuzzy Systems* 7, no. 4 (1999): 453 - 461.

Kumar, Mahesh , and James B Orlin. "A Scale Invariant Clustering using Minimum Volume Ellipsoids." MIT Sloan Research Paper No. 4586-06, 2006.

Kumar, Parvesh, and Siri Krishan Wasan. "Comparative Analysis of k-mean Based Algorithms." *IJCSNS International Journal of Computer Science and Network Security* 10, no. 4 (April 2010): 314-318.

Liu, Hsiang-Chuan, Bai-Cheng Jeng, Jeng-Ming Yih, and Yen-Kuei Yu. "Fuzzy C-Means Algorithm Based on Standard Mahalanobis Distances." *Proceedings of the 2009 International Symposium on Information Processing*, 2009: 422-427.

MacQueen, J. "Some Methods of Classification and Analysis of Multivariate Observations." *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability.* Berkeley, CA: University of California Press, 1967. 281-297.

McLachlan, G J. "Mahalanobis Distance." *RESONANCE*, June 1999: 20-26.

Milligan, G W, and M C Cooper. "A Study of Standardization of Variables in Cluster Analysis." *Journal of Classification* 5 (1988): 181-204.

MuSES 10.2.0 User Manual. Calumet, MI: Thermoanalytics, Inc., 2012.

MuSES Technical Manual Version 10.0. Calumet, MI: Thermoanalytics, Inc., 2011.

O'Rourke, J. "Area of a Polygon." In *Computational Geometry in C (2nd Edition)*, 16-21. New York, NY: Cambridge University Press, 1998.

Qiu, G, M R Varley, and T J Terrell. "Improved Clustering Using Deterministic Annealing." *Pattern Recognition Letters* 15 (1994): 607-610.

QT 4.7 Documentation. 2011. http://doc.qt.nokia.com/4.7/index.html (accessed January 12, 2012).

RadTherm's Convection Options. 2004. http://www.thermoanalytics.com/support/bulletins/bulletin340/index.htm (accessed January 15, 2012).

Rose, Kenneth. "Deterministic Annealing for Clustering,Compression, Classification, Regression,and Related Optimization Problems." *Proceeding of the IEEE* 86, no. 11 (1998): 2210-2239.

Russell, B H, and L R Lines. *Mahalanobis Clustering, with Applications to AVO Classification.* CREWES Research Report, 2003.

Savaresi, S M, and Daniel L Boley . "On the Performance of Bisecting K-Means and PDDP." *Proceedings of the First SIAM International Conference on Data Mining.* Chicago, IL, 2001.

Sebestyen, George. "Decision-making processes in pattern recognition (ACM monograph series)." Indianapolis, IN: Macmillan Publishing Co., Inc., 1962.

Shah, A, C Cless, and John Curlee. "Thermal Analysis and Simulations for Optimizing HVAC Load on Heavy Trucks." *SAE Technical Papers Series, 2008-01-2657*, 2008.

Shioda, Romy, and Levent Tuncel. "Clustering Via Minimum Volume Ellipsoids." *Computational Optimization and Applications* 37, no. 3 (2005): 247-295.

Steinly, Douglas. "K-Means Clustering: A Half-Century Synthesis." *British Journal of Mathematical and Statistic al Psycholog y* 59 (2006): 1-34.

Teboulle, Marc. "A Unified Continuous Optimization Framework forCenter-Based Clustering Methods." *Journal of Machine Learning Research* 65, no. 102 (2007).

Thermoanalytics CAE Software Products. Thermoanalytics, Inc. 2012. http://www.thermoanalytics.com/products (accessed February 3, 2012).

Vassilvitskii, S, and D Arthur. "k- means++:The advantages of careful seeding." *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.* Philadelphia, PA: Society for Industrial and Applied Mathematics, 2007. 1027 - 1035.

Venkatakrishnan, V, and D J Mavriplis. *Improving Agglomeration Multigrid for The Three-Dimension Euler Equations.* NASA Contractor Report 195073, Hampton, VA: Institute for Computer Applications in Science and Engineering, 1995, Accessed July, 2011.

Whitten, Ian H, Eibe Frank, and Mark A Hall. "Advanced Data Mining: Clustering." In *Data Mining Third Edition*, 273-293. Burlington, MA: Morgan Kaufmann Publishers, 2011.

Wolfel , M, and Hazim Kemal Ekenel. "Feature Weighted Mahalanobis Distance: Improved Robustness for Gaussian Classifiers." *13th European Signal Processing Conference.* 2005.

Yang, H, and Srinivasan Parthasarathy. "Mining Spatial and Spatio-temporal Patterns in Scientific Data." *ICDEW '06 Proceedings of the 22nd International Conference on Data Engineering Workshops .* Washington, DC, 2006.

Younis, Khaled S. "Thesis: Weighted Mahalanobis Distance for Hyper-Ellipsoidal Clustering." Master's Thesis, 1996.

Zhang, B. *Generalized K-Harmonic Means -- Boosting in Unsupervised Learning.* Research Report, Palo Alto, CA: Hewett Packard Laboratories, 2000.

Zhang, B, H Meichun , and Umeshwar Dayal. *K-Harmonic Means - A Data Clustering Algorithm.* Research Report, Palo Alto, CA: Hewlett-Packard Labs, 1999.

Zhang, B, Hsu Meichun, and Umeshwar Dyal. Harmonic Average Based Clustering Method and System. US Patent 6,584,433. June 24, 2003.

Zhang, Bin. *Dependence of Clustering Algorithm Performance on Clustered-ness of Data.* Research Report, Palo Alto, CA: HP Labs Technical Reports, 2001.

Appendix 1: Prototype MATLAB Code for K-Means Algorithm

This algorithm is extremely limited since it only performs Euclidean k-means clustering on structured grids. There is no capability to perform agglomeration or to compute volume fluxes between clusters.

```
%_____________________________
% Rob E Smith
% 12/1/2009
% Data Mining Project: CFD Domain Node Network Mining
%
% The general purpose of this program is to implement k-means
% to locate the best nodes for a fluid nodal network based on
% a computational fluids solution domain from StarCCM+.
%
%
% For reference, the following data is expected by column:
%
%Centroid[X] (m)
%Centroid[Y] (m)
%Centroid[Z] (m)
%Temperature (K)
%Velocity: Magnitude (m/s)
%Velocity[i] (m/s)
%Velocity[j] (m/s)
%Velocity[k] (m/s)
%Density (kg/m^3)
%Volume (m^3)
%_____________________________

%clear

%first read the ASCII data into memory
[data textData]=importdata('FN_Results1.csv',' ',1)
[flowLength flowWidth] =size(data.data);

%----------------------------
% Parameters
%----------------------------
numClust=10  %number of clusters to generate
changeCrit=0 %accept convergence once this point is reached
relaxation=1
retrys=10; %number of times to try in hopes of finding a global solution
volWtMode=true;  %when mean computed, volume weight it
eucDistWt=.9
tempDiffWt=0.1
velocDiffWt=0.05
dotProdWt=0.05

maxIter=300; % maximum allowable iterations per try
```

```
%---------------------------
% Computations
%---------------------------

lastDist=0;

retryCount=0;


while(retryCount<retrys)

   %pick some random starting points and load the data
   change=100;
   count=0;
   clustData=zeros(numClust,8);

   randPoints=zeros(numClust,1);
   for i = 1:numClust
      pt=ceil(rand*flowLength);
      clustData(i,:)=data.data(pt,1:8);  % only using thru column 8
      randPoints(i)=pt;
   end


   while(and(change>changeCrit,count<maxIter))
      %compute similarity of each data point to each cluster

      %first lets do distance for each point (euclidian)
      eucDist=zeros(flowLength,numClust);

      for i = 1:numClust

         eucDist(:,i)= (data.data(:,1)-clustData(i,1)).^2+...
            (data.data(:,2)-clustData(i,2)).^2+...
            (data.data(:,3)-clustData(i,3)).^2;

      end

      %then do temperature (simple difference measure)
      tempDiff=zeros(flowLength,numClust);
      for i = 1:numClust
         tempDiff(:,i)= abs(data.data(:,4)-clustData(i,4));
      end



      %next, let's look at the direction match (dot product/magnitude)
      dotProd=zeros(flowLength,numClust);
      for i = 1:numClust
         dotProd(:,i)= ((data.data(:,6).*clustData(i,6))+...
            (data.data(:,7).*clustData(i,7))+...
            (data.data(:,8).*clustData(i,8))./...
            ((data.data(:,5).*clustData(i,5)));
      end
```

```
%finally, compared the velocity magnitudes
%then do temperature (simple difference measure)
velocDiff=zeros(flowLength,numClust);
for i = 1:numClust
   velocDiff(:,i)= abs(data.data(:,5)-clustData(i,5));
end


%OK now normal all these vectors and compute a distance measure
%We'll do it so a high score is bad...
eucDist=eucDist./max(max(eucDist));  %for a maximum of 1
tempDiff=tempDiff./max(max(tempDiff));  %for a maximum of 1
velocDiff=velocDiff./max(max(velocDiff));  %for a maximum of 1
dotProd=1-(1+dotProd)./2; %now range is 0 to 1

distance=eucDist*eucDistWt+tempDiff*tempDiffWt+velocDiff*velocDiffWt+...
   dotProd*dotProdWt;


%Now assign each point to a cluster based on best match
[distVal clustAssign] = min(distance');  %finds the column with a min dist.
clustAssign=clustAssign';  %make it so each row matches the row in data


%finally, compute new cluster center points
totalDist=0;
for i = 1:numClust
   cluster=find(clustAssign==i);
   for j = 1:8

      %Now the distance function should be volume-weighted for each
      %cell and later we'll also volume weight the mean calculations

      if(volWtMode==true)
         %compute new data point
         newClustData(i,j)=sum(data.data(cluster,j).*data.data(cluster,10))/...
            sum(data.data(cluster,10));
      else
         %compute new data point
         newClustData(i,j)=mean(data.data(cluster,j));
      end
   end
   if(volWtMode==true)
      %compute the global error - i.e. big E
      totalDist=totalDist+sum(distance(cluster,i).*data.data(cluster,10));
   else
      %compute the global error - i.e. big E w/o weighting
      totalDist=totalDist+sum(distance(cluster,j));
   end
end

change=sum(sum(abs(clustData-newClustData)));
clustData=clustData*(1-relaxation)+newClustData*relaxation;
```

```
      count = count + 1;
      fprintf(1,'Count %d   TotalDist %d   Change %d\n',count, totalDist,change);
   end
   retryCount=retryCount+1;

   fprintf(1,'Try:%d   Total distance:%10.8f   Last best:%10.8f\n',...
      retryCount,totalDist,lastDist);

   if(or(totalDist<lastDist,retryCount==1))
      lastDist=totalDist;
      finalClustData=newClustData;
      finalClustAssign=clustAssign;
   end


end

clear dumpme
dumpme=finalClustData(:,1:3);
save -ascii -double 'output.txt' finalClustData;
save -ascii -double 'output_plot.txt' dumpme;

%now output the x,y,z of the points in each cluster
%finally, compute new cluster center points
clear dumpme
totalDist=0;
for i = 1:numClust
   cluster=find(finalClustAssign==i);
   dumpme=data.data(cluster,1:3);

   outName=strcat('cluster',int2str(i),'.txt');
   save(outName, 'dumpme','-ascii','-double')
end
```

Appendix 2: Full Set of Validation Plots

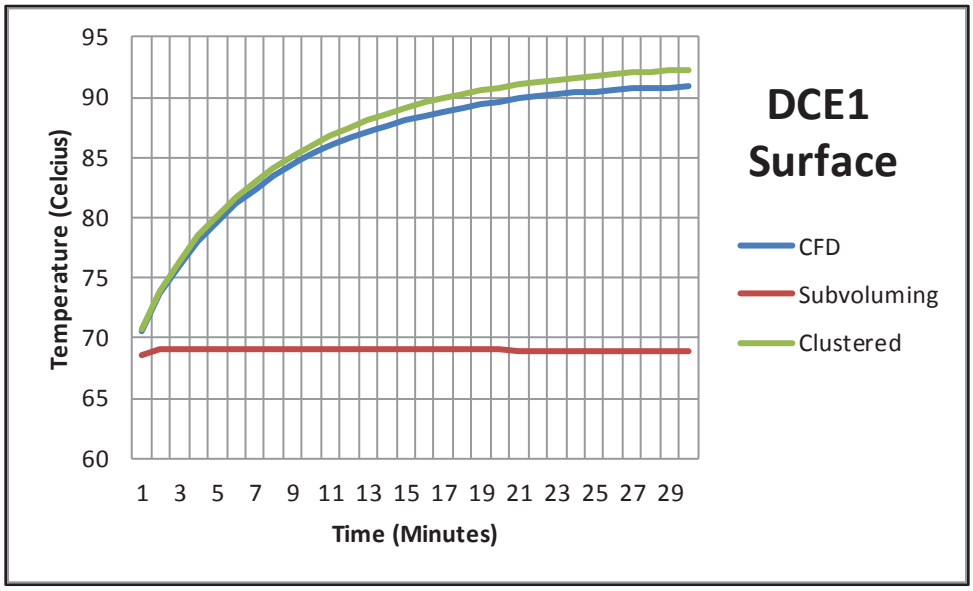

Figure 76: DCE 1 Surface Temperatures.

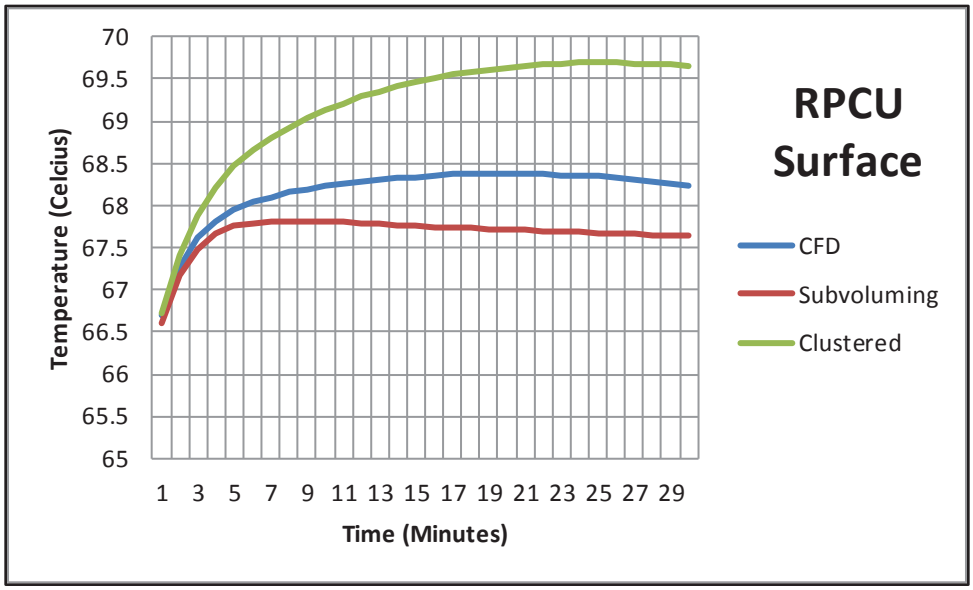

Figure 77: RPCU Surface Temperatures.

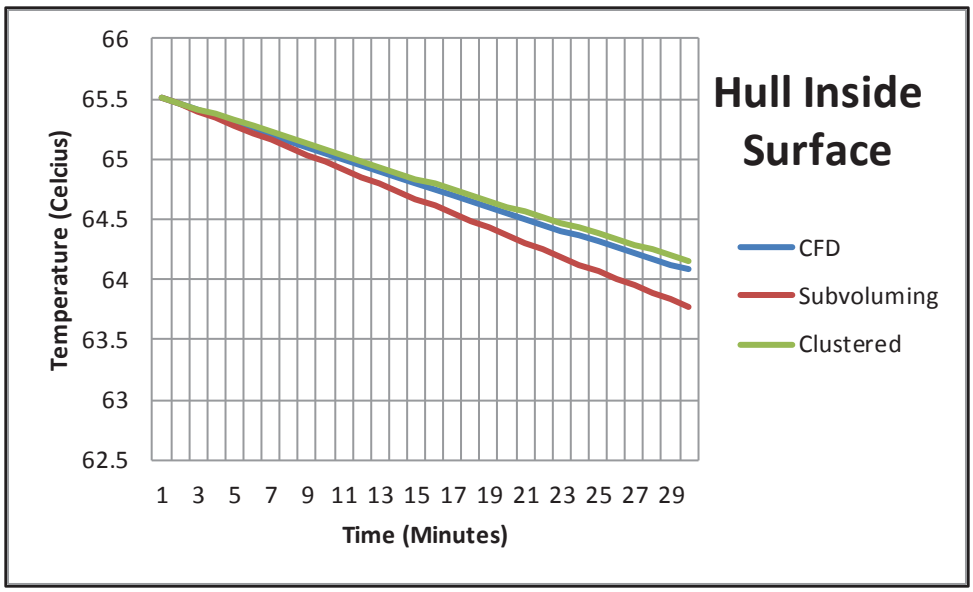

Figure 78: Hull Inside Surface Temperatures.

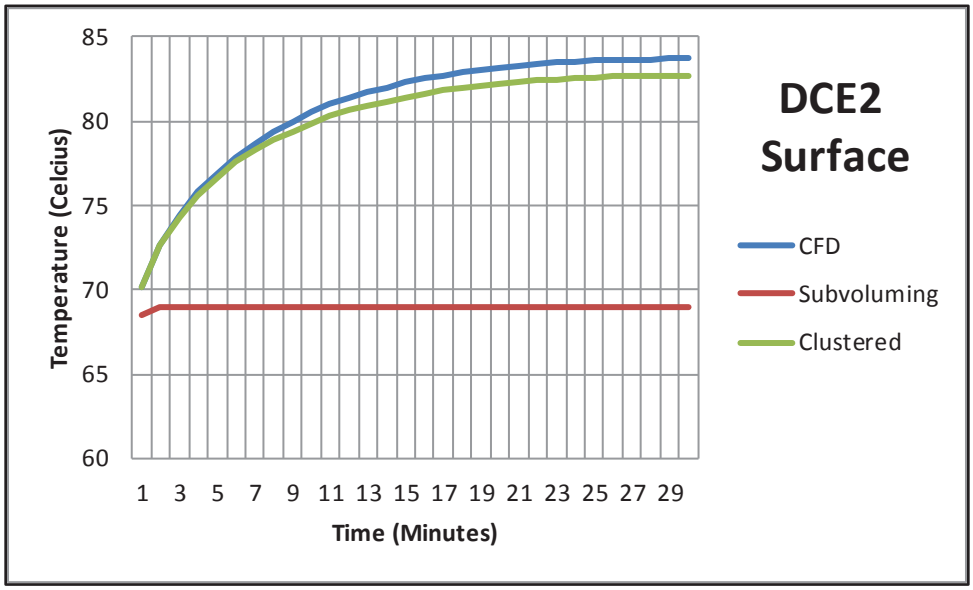

Figure 79: DCE 2 Surface Temperatures.

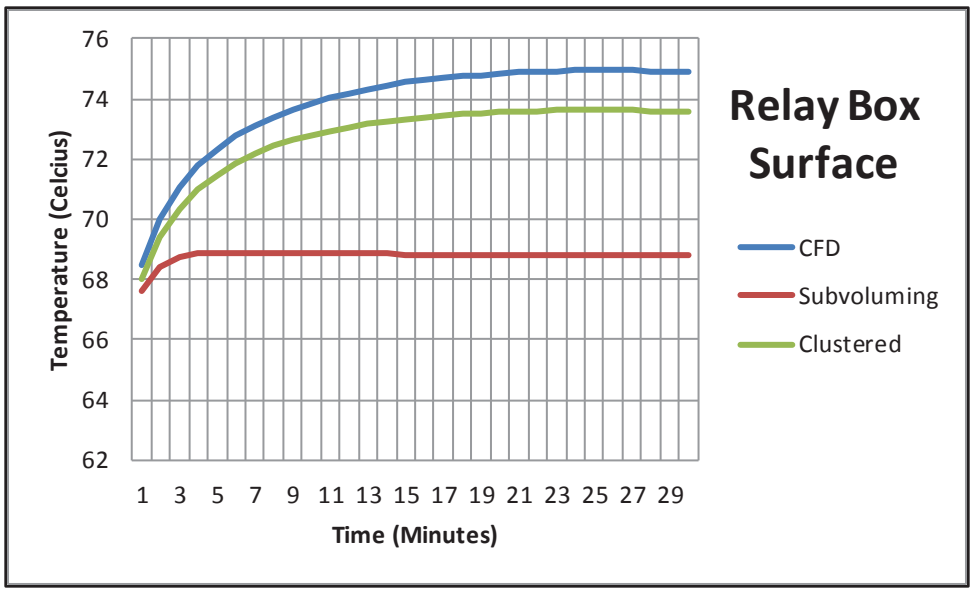

Figure 80: Relay Box Surface Temperatures.

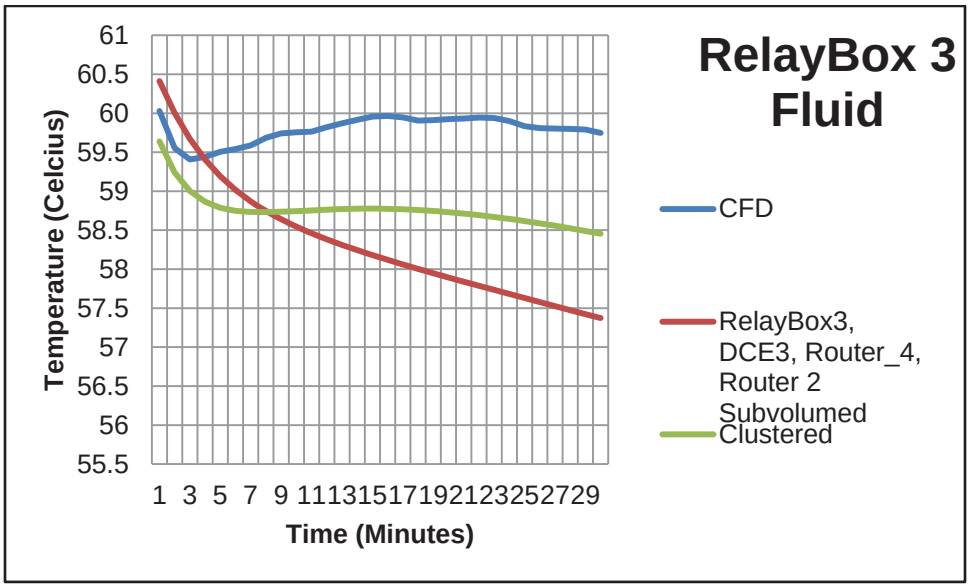

Figure 81: Relay Box 3 Fluid Temperature.

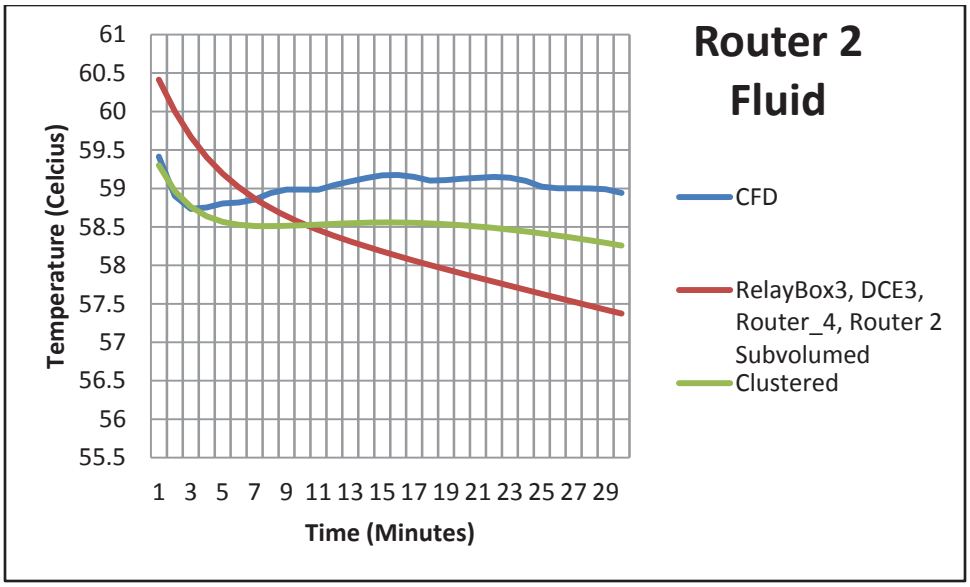

Figure 82: Router 2 Fluid Temperature.

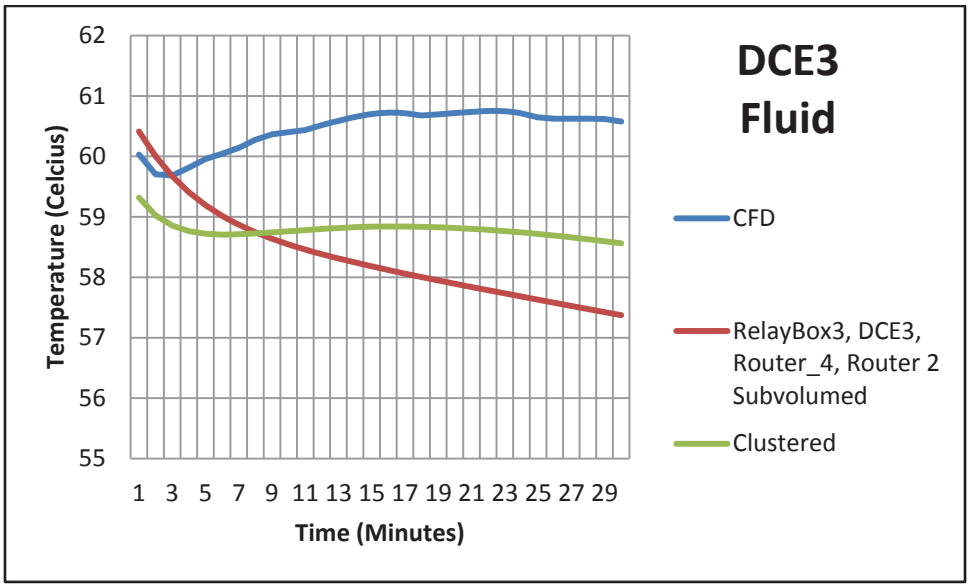

Figure 83: DCE 3 Fluid Temperature.

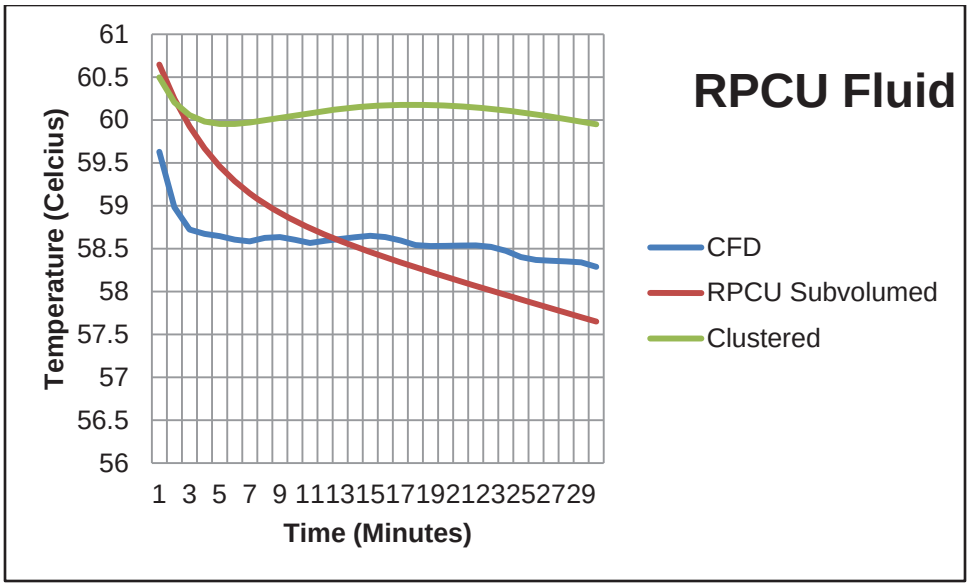

Figure 84: RPCU Fluid Temperature.

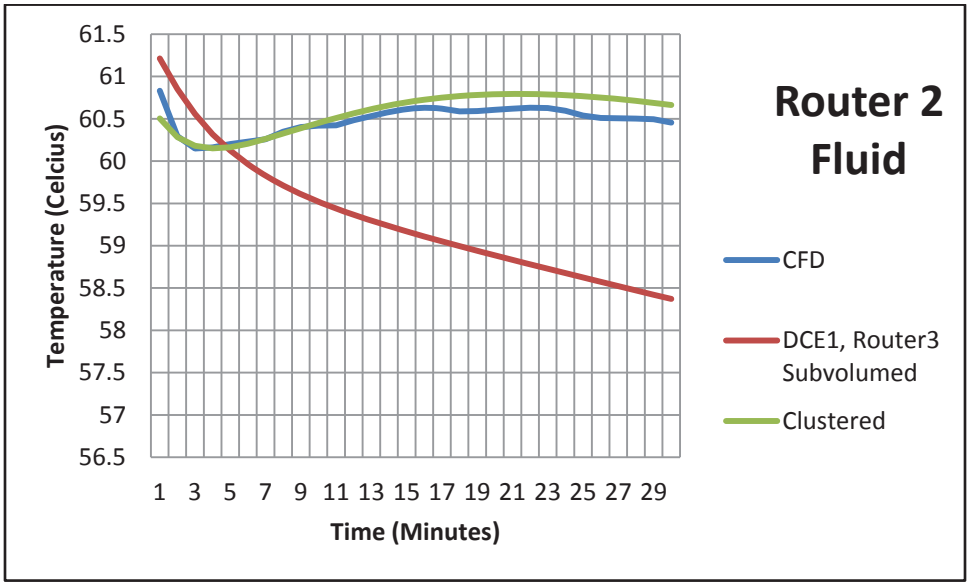

Figure 85: Router 2 Fluid Temperature.

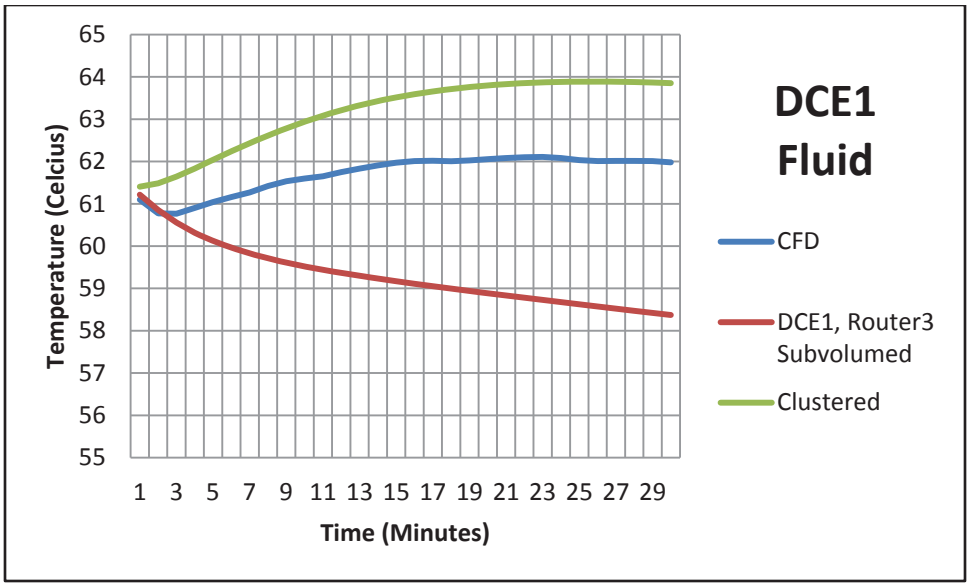

Figure 86: DCE 1 Fluid Temperature.

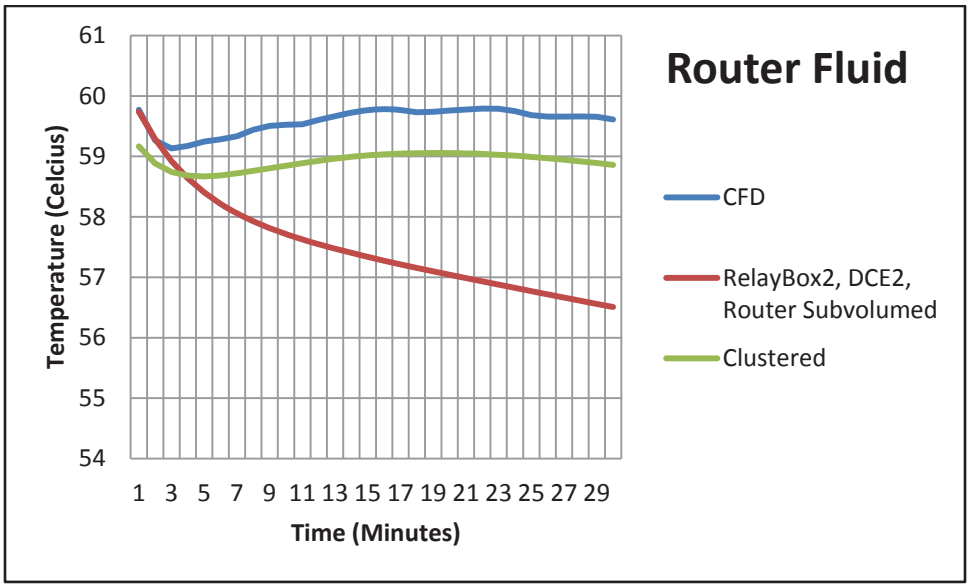

Figure 87: Router Fluid Temperature.

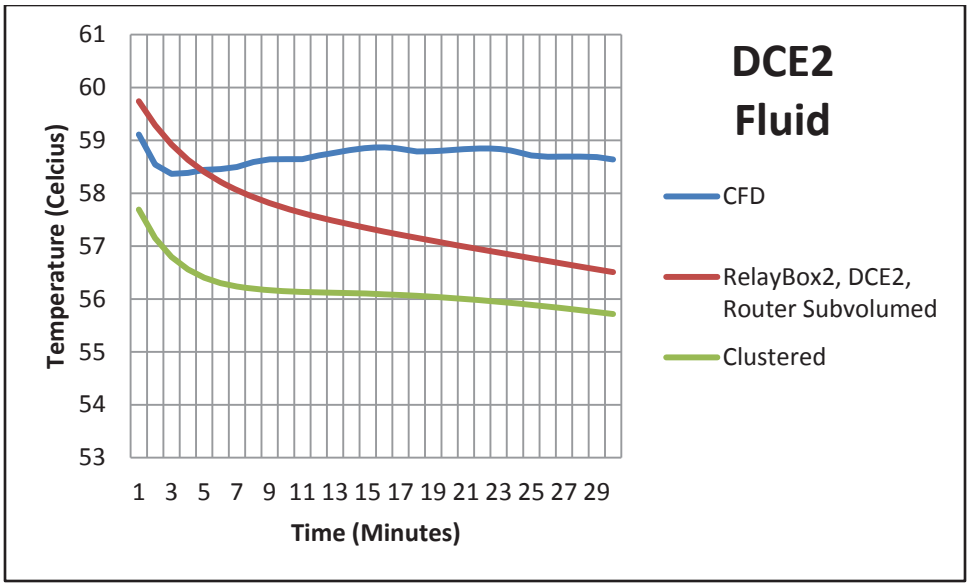

Figure 88: DCE 2 Fluid Temperature.

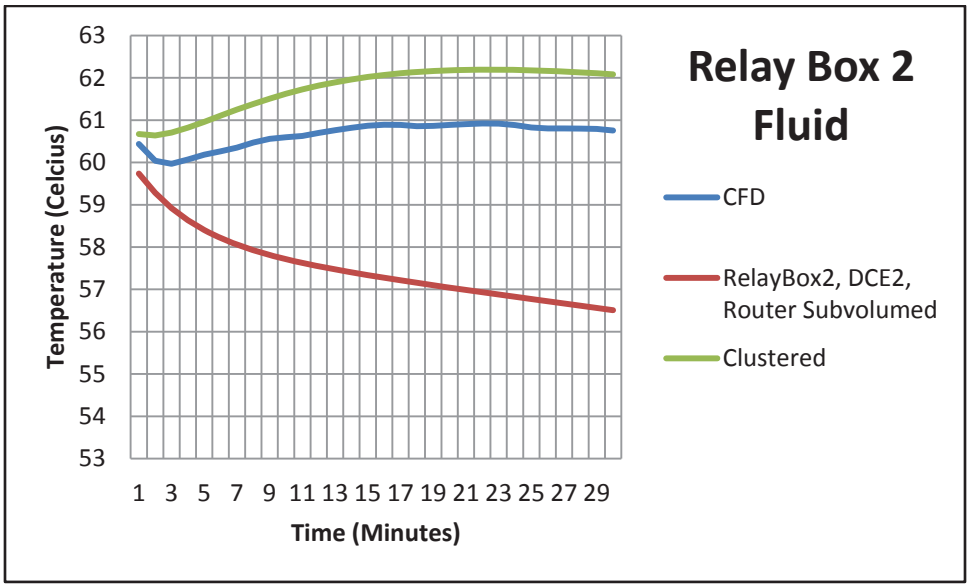

Figure 89: Relay Box 2 Fluid Temperature.

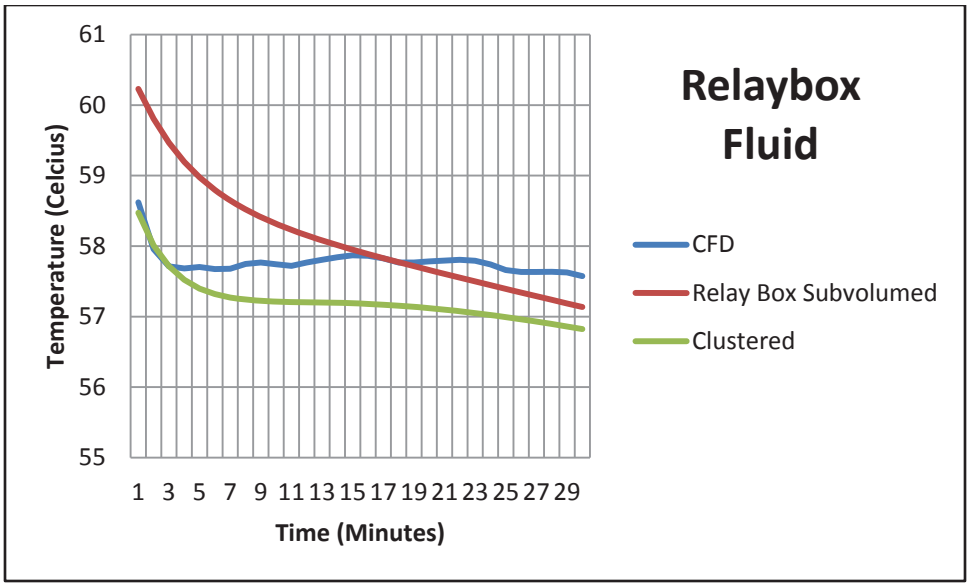

Figure 90: Relay Box Fluid Temperature.

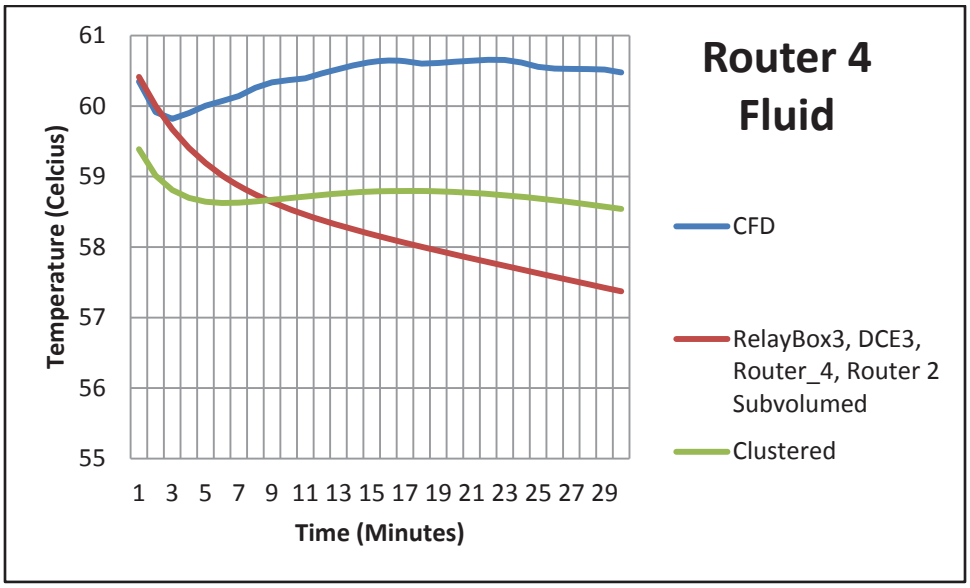

Figure 91: Router 4 Fluid Temperature.

‘

Appendix 3: cfdMine Mahalanobis GUI

cfdMine Mahalanobis GUI. Note the addition of “initial Euclidean iterations” , the covariance update frequency, and the slider bar allowing blending of Euclidean and Mahalanobis distances.

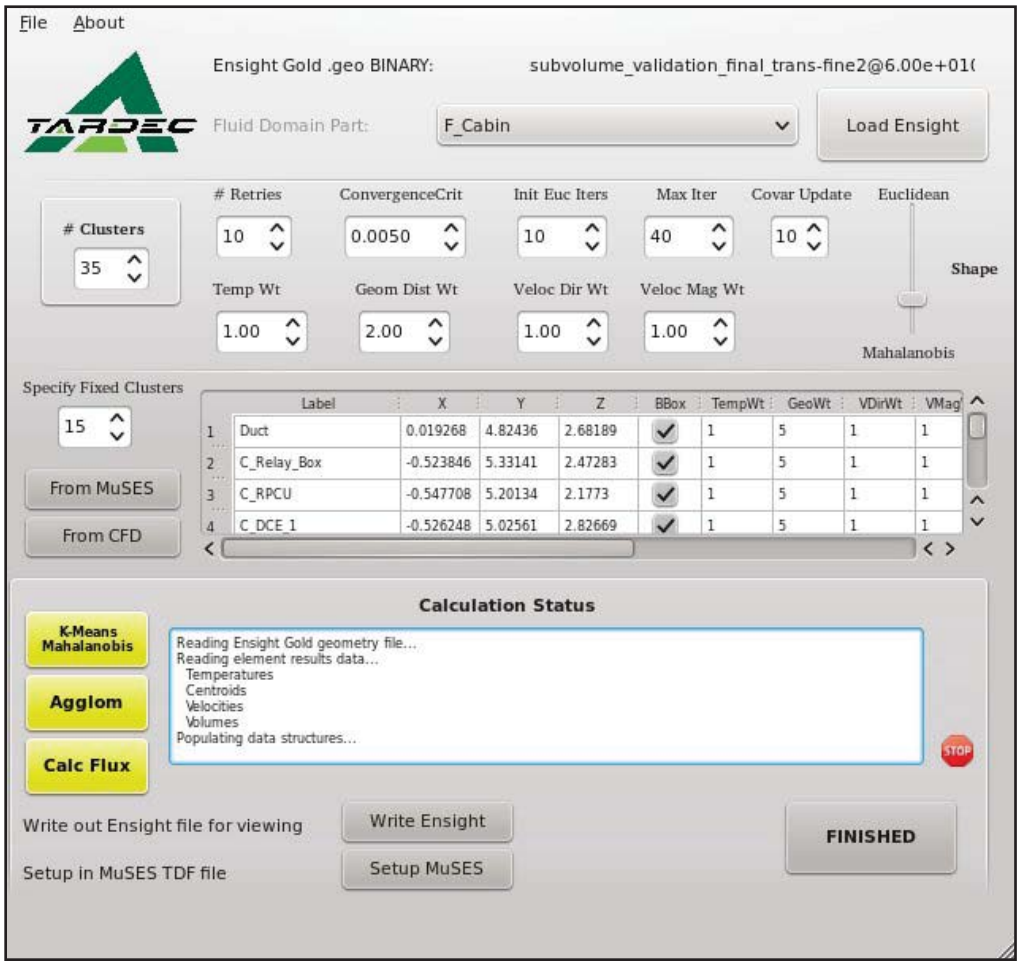

Figure 92: cfdMine-Mahalanobis GUI.