

The CVT Basis Generation Algorithm

John Burkardt

29 May 2003

Abstract

This document describes the CVT basis algorithm employed in the program `CVT_BASIS`, and used for computations in joint work with Max Gunzburger and H C Lee.

1 Introduction

The model reduction algorithm relies on the observation that a physical system having thousands of degrees of freedom may nevertheless often be well described by its extent in just a few (well-chosen) directions.

The CVT Basis Generation algorithm is the intermediate step of the model reduction algorithm. It follows the generation of sample data, creating a set of basis vectors, which the subsequent step will use to approximate new physical states.

The CVT Basis Generation algorithm begins by preprocessing the data to account for the boundary conditions. Once this is done, the physical state data will be treated *abstractly*, that is, the calculations will be made without any knowledge of the governing equations, constraints or parameter values associated with the original physical system.

Viewed in the abstract setting, the algorithm proceeds with a set X of n vectors, elements of \mathfrak{R}^m , called *data points*. The goal is to determine a set Z of k vectors, and a function $C(x) = z$ assigning each $x \in X$ to a $z \in Z$.

We call the collection (X, C, Z) a *cluster configuration*. Each point z is called a *cluster generator*; the function $C(x)$ is the *cluster assignment rule*, and the set of all data points $x \in X$ which are assigned to a particular z is called a *cluster*, and may be symbolized by $D(X, C, z)$.

The basis generation algorithm works by creating, modifying, and evaluating a number of cluster configurations, searching for a set of underlying generators with an optimal approximating property. An objective function will be defined to formalize and quantify the desired behavior of an optimal set of cluster generators Z , and the accompanying cluster assignment rule C . This function is called the *cluster energy*, defined as:

$$E(X, C, Z) \equiv \sum_{i=1}^n \|C(x_i) - x_i\|^2 \quad (1)$$

Here $\|*\|$ is a suitable norm. The notation $\|*\|_2$, in particular, will denote the usual Euclidean norm.

Given a data set X , and a distance function $\|*\|$, the goal of the basis generation algorithm can now be formulated as the determination of a set of cluster centers Z and a cluster assignment rule C which together minimize the cluster energy $E(X, C, Z)$.

2 Discrete Centroidal Clustering Algorithms

The discrete nature of the problem rules out many optimization approaches that would be suitable for a continuous problem. The number of possible clusterings of the data is enormous, meaning that an exhaustive search cannot be carried out.

A reasonable way to approach the problem, then, might first concentrate on reducing the number of possibilities to be considered. In particular, if we can identify a smaller abstract space in which the optimal clustering must lie, we can save much effort by working only in that space.

A decisive criterion is available for clustering problems that use the Euclidean norm in \mathfrak{R}^m . In this case, the minimal cluster energy will be obtained by a *nearest-neighbor centroidal-generator clustering* (“**NNCGC**”), which is a clustering (X, C, Z) that satisfies the following two conditions:

- *Nearest-neighbor*: the clustering rule C assigns each $x \in X$ to the nearest $z \in Z$:

$$C(x_i) = z_j \implies \|x_i - z_j\|_2 = \min_{z \in Z} \|x_i - z\|_2 \quad (2)$$

- *Centroidal-generator*: each of the cluster generators z is the centroid of the points assigned to it:

$$z = \sum_{x \in D(X, C, z)} x / |D(X, C, z)| \quad (3)$$

Please note that, for a given set of data X , there may be many NNCGC’s. So simply finding one doesn’t guarantee that we have minimized the energy. What we do know is that, if possible, we should confine our search to NNCGC’s, since that’s where the optimal solution is sure to lie.

There are effective algorithms for computing NNCGC’s. The algorithms to be used here are the H-means and K-means methods (Hartigan[3], Späeth[7], Sparks[8]). The H-means method is cheaper to use than K-means, but does not minimize the energy function directly, and so is not completely reliable. Since the two methods are compatible, an efficient technique is to apply H-means until it can achieve no further energy reduction, followed by (presumably few) steps of K-means to drive the energy down.

The H-means and K-means algorithms have the attractive property of being *descent methods* with respect to the cluster energy. Given any cluster configuration (X, C, Z) , both algorithms consider a series of modifications to the cluster

assignment function C . However, a given modification is only accepted if it is guaranteed to reduce the cluster energy $E(X, C, Z)$. Thus, an algorithm based on H-means or K-means steps can never diverge or produce a final cluster with higher energy than the initial cluster. The algorithms are guaranteed to terminate in finitely many steps. By virtue of the nearest neighbor and centroidal generator features, the resulting clusters will be discretely convex, and tend to correspond to natural clusterings of the data.

The single step approach has the disadvantage of making the methods vulnerable to “local minima”, a term which needs to be clarified for the discrete problem being considered. Given a nearest-neighbor centroidal cluster configuration (X, C, Z) , define a *one point change* as a change, for a single data point x_i , in the value of the cluster assignment function $C(x_i) = z_j$ to z_k , followed by the replacement of z_j and z_k by the centroids of their clusters.

A *local minimum* is then a cluster configuration (X, C, Z) , with the property that every one-point change results in an increase in the cluster energy. The K-means algorithm will never allow a one-point change that increases the cluster energy, and so it is guaranteed to halt if it reaches a local minimum. However, for a given problem, there are usually many local minima that do not minimize the cluster energy. To deal with the problem of local minima, then, the algorithm will simply be carried out a number of times, using different initial configurations, and the final configuration with lowest cluster energy shall be taken as the best estimate of the optimal configuration.

2.1 The H-Means Algorithm

The H-Means algorithm begins with an initial cluster configuration (X, C, Z) . A distance function $d(x, y)$ is also required, which is usually simply derived from the vector norm being used:

$$d(x, y) = \|x - y\|. \tag{4}$$

Then for each data point $x \in X$, the algorithm finds the nearest cluster generator $z \in Z$; if that is not the cluster to which the data point is currently assigned, the data point is “transferred” to the cluster associated with the nearest generator, that is, the assignment function C is modified so that $C(x) = z$. When all data points have been considered, the cluster centers z are discarded, and are replaced by the centroids of the clustered data points.

If at least one data point was moved during this process, a new sweep of all the data points is carried out, and this continues until a sweep has been made in which no data point was moved.

This may be formalized as:

Do

For each data point $x_i \in X$

Determine the generator $z_j \in Z$ which minimizes $d(x_i, z_j)$.

if $C(x_i) \neq z_j$, then

$C(x_i) = z_j$,

end if

end for

If no transfers occurred, exit the loop.

For each $j \in \{1, \dots, k\}$

$z_j = \sum_{x_i \in D(X, C, z_j)} x_i / |D(X, C, z_j)|$

end for

end do

2.2 The K-Means Algorithm

The K-Means algorithm is a sophisticated version of H-Means. The H-Means exchange test is based on the distance from a point to a cluster center. But if a point is moved to a new cluster, the cluster center will move closer to the point when it is recomputed, something that H-Means misses. The K-Means algorithm works with an exchange test that directly measures the change in energy that occurs when a point moves from one cluster to another, including effects caused by the subsequent adjustment of both cluster centers. This test is computationally more expensive, and also requires that the pair of cluster centers be updated immediately upon an exchange. This expense is the reason that the H-Means algorithm is employed first to do the “easy” exchanges.

Do

For each data point $x_i \in X$

Evaluate the cluster energy for all one-point changes involving x_i .

If moving x_i from cluster j to k will most reduce the cluster energy, then

$C(x_i) = z_k$,

replace z_j and z_k by the centroids of the modified clusters.

end if

end for

If no transfers occurred, exit the loop.

end do

3 Data Processing

With the understanding that the preprocessing has been carried out, and that a set of data points X is ready for processing, the basis generation algorithm may now be outlined as follows:

Preprocess the data, as requested.

Given data points $X \in \mathfrak{R}^m$ and a cluster size k :

$$E_{opt} = \infty$$

do n_{trial} times

 Initialize the cluster configuration (X, C, Z)
 by randomly setting the entries of Z ,
 and randomly assigning each data point to a cluster.

 Apply (many) H-Means steps to (X, C, Z) ;

 Apply (a few) K-Means steps to (X, C, Z) ;

 if $E(X, C, Z) < E_{opt}$
 $(X, C_{opt}, Z_{opt}) = (X, C, Z)$;
 $E_{opt} = E(X, C, Z)$
 end if

end do

output (X, C_{opt}, Z_{opt}) ;

The main lines of the algorithm should be clear from this sketch. Each trial of the loop randomly initializes the cluster configuration, and uses H-means and K-means to produce a locally optimal cluster configuration which is the next candidate. If the cluster energy of this candidate is the lowest seen so far, it is saved as the current estimate of the optimal configuration, and the best candidate encountered during all the trials is output at the end.

Note that this approach can easily be used in a parallel-processing environments with very little communication. Each processor receives a copy of the data points, and the initial value of the minimal clustering energy, which might as well be infinity. After any processor computes a new clustering, if the clustering energy has been reduced, it reports the new minimal energy to the master

processor, and then proceeds to its next computation. At the end of the computation, the processor that actually encountered the minimal clustering energy needs to report the corresponding cluster configuration to the master processor.

It is now appropriate to consider some details of the preprocessing, the initialization, the H-means and K-means algorithms, and certain difficulties that may occur unexpectedly during the computation.

3.1 Preprocessing

Before the PDE data points are processed by the algorithm, there are good reasons to consider preprocessing the data so that it more readily exposes the patterns of clustering being sought. In particular, the data points may be subjected to appropriate shifting, scaling and thinning.

The thinning operation is the simplest to describe. In some cases, we may wish to determine the robustness of our computation by thinning the data points, that is, by deleting some percentage of the data points, so that a comparison may be done to a computation with the full set of data.

Shifting is done to factor out the effect of boundary conditions. The PDE is assumed to have nonzero boundary conditions. It will usually be most convenient to “subtract off” the components of the data points associated with boundary conditions, typically by subtracting a steady state solution, or some other reference solution, denoted ϕ_0 .

If shifting is done, then the resulting basis set vectors will need to be augmented in some way to handle boundary conditions. It may be enough simply to append ϕ_0 to the basis set.

The data points can now be thought of as abstract points in a linear space of perturbations of the steady state solution. Because the goal is to produce a set of basis vectors, each data point in the linear space is primarily interesting because of its direction, not its magnitude. In this view, if one data point is simply a multiple of another, it carries no interesting information at all. To factor out the effect of differences in norm, an optional preprocessing step may be taken which gives each data point unit Euclidean norm. In this case, the preprocessed data points will now lie on a hypersphere. However, the clustering computation will still be carried out in Euclidean space.

It is possible to modify the algorithm to take this data normalization into account. The changes necessary to the original algorithm are small. The primary change is that the Euclidean distance between two data vectors is replaced by an angular measure:

$$d(x, y) \equiv \left| \sin\left(\arccos\left(\frac{x \cdot y}{\|x\| \|y\|}\right)\right) \right| \quad (5)$$

and the associated norm becomes

$$\|x\| \equiv d(x, 0) \quad (6)$$

The cluster generators Z will also be required to lie on the hypersphere. After new cluster generators are computed by the centroidal computation, they must

be normalized. This computation produces the analog of the usual centroid, and in turn, has the desired minimizing properties for the cluster energy, (which must also now be computed using angular distance).

3.2 Cluster Initialization

The initialization of the set of cluster generators Z might seem unimportant. However, experience showed that a poor initial choice of Z could retard or derail the algorithm, especially if some generators z attracted few or no cluster points. Smooth program execution occurred when the set Z was computed by setting each z to a convex combination of the values in X . Formally, this amounts to computing a $k \times n$ coefficient matrix R . The entries are chosen uniformly at random, and then normalized so that each row sums to 1. Z is then computed as

$$Z = R \times X \tag{7}$$

which means that each z is a convex combination of entries of X . A cluster generator can still have zero adherents, but this is unlikely. Further, the generators are forced by construction to lie in the convex hull of the data.

By contrast, initialization of C has almost no effect on the current algorithm. The H-means and K-means algorithms will adjust the assignment rule so that it always assigns a data point to the cluster with the nearest generator.

3.3 Repair of Empty Clusters

In the H-Means algorithm, the cluster generator stays fixed until the end of a full sweep of all the points. During that time, it is possible for a cluster to become empty, that is, for all the points in the cluster to be transferred to other clusters. The H-Means algorithm is susceptible to this occurrence in part because the generators are not immediately updated as points are transferred. And once the cluster is empty, the algorithm for updating the cluster generators will not produce a meaningful result.

The appropriate response to such a breakdown in the algorithm is simply to replace the generator for the empty cluster by a random point, generated by the same method used to initialize the cluster generators. The cluster populations need to be recomputed at this time, with the expectation that some points will fall into the new cluster. If the cluster is still empty, another random generator can be tried, until the problem is overcome.

4 An Example Computation

Flow solutions were computed in a channel with a deep rectangular well. The flow entered from the left and exited on the right. A grid of 4961 nodes was used. A parameter α controlled the strength of the inflow. The first 250 time steps were computed at a fixed parameter value $\alpha = \frac{5}{3}$; time steps 251 through

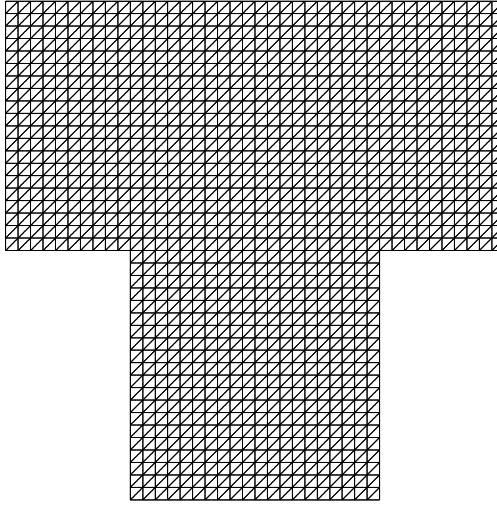


Figure 1: The finite element grid for the example computation.

Table 1: Energy and Iteration Statistics for a Single Case

Process	Final Energy	Iterations
Random Init	49,860.5	1
H-Means	3,283.4	22
K-Means	3,212.5	4

500 were computed with $\alpha = \frac{1}{3}$, causing sharp transient effects. At each time step, the velocity vector field was saved.

The basis generation algorithm preprocessed the data by subtracting an appropriate multiple of the steady state boundary condition, which had been computed with $\alpha = 1$. In the processing stage, 8 cluster generators were sought. The overall computation was repeated with 15 random initial configurations. The inner H-Means and K-Means algorithms were limited to a maximum of 30 iterations, but always concluded before the limit was reached. The initial cluster energy of the random configurations was generally about 50,000; the final cluster energy of the 15 cases varied from 2,879.5 to 3,212.5.

For a “typical” random initial configuration, Table (1) shows how the cluster energy dropped: Of course, the huge energy at the initial random configuration is slightly misleading. It drops tremendously after the first H-Means step, as each data point is assigned to its nearest cluster center.

There was an interesting feature of the final cluster generators. For each cluster, the cluster elements tended to comprise all the solutions in a time interval. In particular, for the case discussed here, the 8 cluster generators had

Table 2: Cluster Statistics for Computed Generators

Cluster	Cluster Energy	Population	Range
1	281.1	5	[1, 5]
2	247.5	10	[6, 15]
3	236.4	18	[16, 33]
4	215.3	41	[34, 74]
5	532.1	369	[75, 250] + [308,500]
6	523.3	7	[251, 257]
7	428.3	15	[258, 272]
8	420.1	35	[273, 307]
Σ	3,212.5	500	[1, 500]

components as listed in Table (2). In the interests of clarity, we’ve renumbered the cluster indices.

Note that, except for cluster 5, the clusters are formed exactly from a sequence of data points at neighboring times. The two smallest clusters, number 1 and number 6, correspond to the initial transient, and the “shock” to the system that occurs after step 250, when the system parameter is changed. Since the solution is changing rapidly at those times, it makes sense that the clusters are small, and yet still comprise a sizable portion of the total cluster energy. Also, the one exceptional cluster, number 5, corresponds to the relatively quiescent “tails” of the two evolution processes, since it contains the solutions for steps [75,250] and [308,500].

References

- [1] Qiang Du, Vance Faber, Max Gunzburger, *Centroidal Voronoi Tessellations: Applications and Algorithms*, SIAM Review, Volume 41, Number 4, pages 637-676, December 1999.
- [2] John Hartigan, *Clustering Algorithms*, Wiley, 1975.
- [3] John Hartigan, M A Wong, *Algorithm AS 136: A K-Means Clustering Algorithm*, Applied Statistics, Volume 28, Number 1, 1979, pages 100-108.
- [4] Wendy Martinez and Angel Martinez, *Computational Statistics Handbook with MATLAB*, Chapman and Hall/CRC, 2002.
- [5] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, Sung Nok Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, seconde edition, Wiley, 2000.
- [6] Helmut Späth, *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Ellis Horwood, 1980.

- [7] Helmut Späth, *Cluster Dissection and Analysis, Theory, FORTRAN Programs, Examples*, Ellis Horwood, 1985.
- [8] D N Sparks, *Algorithm AS 58: Euclidean Cluster Analysis*, Applied Statistics, Volume 22, Number 1, 1973, pages 126-130.