Dimensionality Reduction in the Parameter Space: Active Subspace and Nonlinear Level Set Learning Approaches

Malcolm Gaynor and Cade Stanley
Mentor: Zhu Wang

July 15, 2022

1 Introduction

Many problems in computational science and engineering involve complex models using functions of several variables. Unfortunately, performing studies using these models comes with significant computational costs. This problem is often referred to as the "curse of dimensionality," due to the fact that computational costs increase exponentially as the input dimension increases. An ideal solution to this problem would be reducing the dimension of the input parameter space without compromising the model's accuracy.

A simple way to do this would be to eliminate the unimportant input parameters, where "unimportant" means that changes in these inputs result in minuscule or inconsequential changes in the value of the output function. However, in many cases it is not this simple - all of the inputs may contribute significantly to the output. Instead of searching for significant (or insignificant) input parameters, then, we can search for significant *directions* in the parameter space. That is, we can try to find particular directions in which much of the function's variability is concentrated; in other directions, the value of the function will change very little.

For instance, consider the function $f(x_1, x_2) = e^{1.5x_1+0.8x_2}$. Although f is a function of two variables, it behaves like a function of one variable along a particular direction. We see this behavior in the figure below.

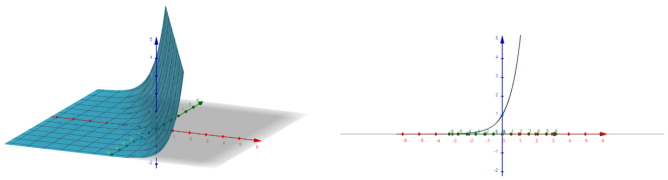


Figure 1: Reducing f from a function of two variables to a function of one variable. Here, rotating the function helps identify an important direction in the parameter space.

We explore two different approaches to finding important directions in the parameter space - the active subspace (AS) method, developed in [2], and nonlinear level set learning (NLL), developed in [8]. The AS method is a linear approach, with the goal of finding linear combinations of the input parameters in which almost all of the changes in the function's value can be observed. This method relies on eigendecomposition of a gradient covariance matrix. In many cases, finding an active subspace is a good first approach for dimensionality reduction. However, it may not work well for functions whose level sets have highly nonlinear geometry. In these cases, the NLL approach may be preferable. This method can produce nonlinear transformations in addition to linear transformations, but it requires training a reversible residual neural network (RevNet).

We will summarize the development of these two approaches, demonstrate some basic examples, and apply each method to the dimension reduction of three functions with applications in engineering. The implementation of these methods will rely on code based on the tutorials provided in [4] and [1].

2 The Active Subspace Method

2.1 Development

2.1.1 Finding the Active Subspace

Suppose f is a d-dimensional function, with domain $\Omega \subset \mathbb{R}^d$, whose dimension we would like to reduce. Using the AS method, the goal is to find a subspace of Ω in which changes in the input parameters explain most of f's variability. To do this, we focus on the matrix $\boldsymbol{C} = \mathbb{E}\left[(\nabla_{\boldsymbol{x}} f)(\nabla_{\boldsymbol{x}} f)^\top\right]$, as developed in [2]. This matrix, in a sense, provides information about the important directions in a function's parameter space. However, it is difficult to calculate exactly because it requires evaluating multiple integrals. Thus, in practice we approximate it using

$$\hat{\boldsymbol{C}} = \frac{1}{M}\sum_{i=1}^{M}(\nabla_x f_i)(\nabla_x f_i)^T.$$

Here, $\nabla_{\boldsymbol{x}} f_i = \nabla_{\boldsymbol{x}} f(x_i)$, where each x_i is one of M randomly sampled points from the domain Ω. Note that $\hat{\boldsymbol{C}}$ is symmetric, so it has a real eigendecomposition, $\hat{\boldsymbol{C}} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T$, where we list the eigenvalues in Λ in decreasing order. This eigendecomposition is key to the AS method. The eigenvector matrix $\boldsymbol{W}$ can be thought of as supplying all of the potentially important directions in the parameter space, and the eigenvalue matrix $\boldsymbol{\Lambda}$ supplies information about how important each eigenvector direction is. The larger an eigenvalue is, the more important the corresponding eigenvector direction is. Thus, when looking at the eigenvalues in decreasing order, the AS method requires looking for large gaps - where the eigenvalues go from very large to very small, and therefore the eigenvectors go from important to unimportant. When a large eigenvalue gap exists, we can partition $\boldsymbol{W}$ and $\boldsymbol{\Lambda}$ as follows:

$$\boldsymbol{W} = \begin{bmatrix}\boldsymbol{W}_1 & \boldsymbol{W}_2\end{bmatrix} \qquad\qquad \boldsymbol{\Lambda} = \begin{bmatrix}\boldsymbol{\Lambda}_1 & \\ & \boldsymbol{\Lambda}_2\end{bmatrix}$$

The eigenvalue gap separates $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$, and the corresponding eigenvectors are separated similarly. Thus, $\boldsymbol{W}_1$ holds the "important" directions for the function - those that make up the active subspace.

2.1.2 Visualizing the Active Subspace

In order to fully analyze the performance and behavior of the AS method for dimension reduction, we utilize several visualization tools:

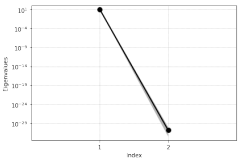
- **Eigenvalue plot**. A plot of the eigenvalues in $\boldsymbol{\Lambda}$, ordered from largest to smallest. This plot is used to identify a large gap in the eigenvalues, where $\boldsymbol{W}$ can be partitioned into active and inactive eigenvectors.
- **Sufficient summary plot**. A plot of the function's value against linear combinations of the active eigenvectors in $\boldsymbol{W}_1$, used to visualize the function's value along the active subspace directions in the parameter space. If the function's value can be explained well by only looking at the active subspace directions, a sufficient summary plot will demonstrate a strong trend, with limited variability. That is, a sufficient summary plot with a strong trend indicates that the AS method provides a good dimension reduction for the function.
- **Sensitivity chart**. A chart that compares the function's sensitivity to its original parameters with its sensitivity to the active and inactive dimensions created by the AS (or NLL) transformation. In practice, the function's sensitivity to a given dimension is calculated as the sample mean of the absolute values of the partial derivative. We obtain a relative sensitivity measure by dividing each sensitivity value by the sum of sensitivity values across every dimension. The goal is to have much of the sensitivity concentrated in the active dimensions. For example, an active subspace of dimension 2 should result in much of the sensitivity being concentrated in the first 2 dimensions, with very little in the remaining dimensions.

- **Activity score**. A sensitivity metric used to measure the importance of each input parameter to the function's value, calculated using information from the eigenvectors and eigenvalues yielded by the AS method. Unlike the relative sensitivity measures described above, which are used to evaluate the efficiency of the dimension reduction technique, activity scores simply describe the behavior of the original function. They do so by indicating how strongly each input parameter influence's the function's output.

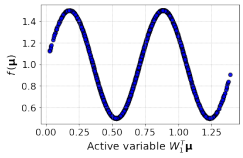
- **Grid transformation**. A method for visualizing the behavior and properties of the transformation produced by the AS or NLL methods. This method involves visualizing a grid representing the original parameter space, then applying the dimension reduction transformation to the original space in order to produce a new, transformed parameter space.

2.1.3 A Simple Example

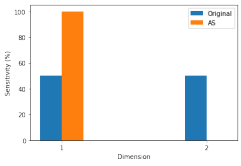
Consider $f(\boldsymbol{x}) = \frac{1}{2}\sin(2\pi(x_1 + x_2)) + 1$, a function of two variables. We apply the AS method in an attempt to reduce it to one active input dimension, with the results shown in the figures below.



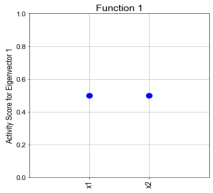
(a) Eigenvalue plot



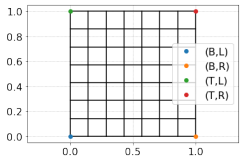
(b) Sufficient summary plot



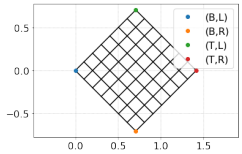
(c) Sensitivity chart



(d) Activity score



(e) Original parameter space



(f) Transformed parameter space

Figure 2: Output plots visualizing the performance of the AS transformation when applied to f.

The eigenvalue plot in (a) exhibits a very large gap (30 orders of magnitude) between the first and second eigenvalues produced by the active subspace eigendecomposition, indicating that an active subspace of dimension one would be appropriate for this function. Thus, we take the first eigenvector direction from the eigendecomposition as the active subspace, and plot the function's value against this subspace. This yields the sufficient summary plot in (b), which demonstrates a strong univariate trend, with minimal variability in the function's value at any given value of the active variable. Thus, we conclude that the value of f can be explained very well by only considering input parameters along the active eigenvector direction. That is, the active subspace allows us to reliably study the behavior of f using only one input dimension rather than two - the dimension reduction is effective.

Another way to judge the performance of the dimension reduction is to examine a sensitivity chart, as seen in (c). Here, we see that the original function is equally sensitive to changes in both input dimensions (x_1 and x_2), with relative sensitivities of 50%. Upon performing the AS transformation, though, the function is almost completely dependent on the first dimension (the active subspace dimension), with nearly 100% of its sensitivity concentrated there. Again, this demonstrates that we can study the function using only one input dimension, namely, the active dimension provided by the AS method. The activity score chart in (d) provides another measure of sensitivity for the original function. Using the eigenvectors and eigenvalues generated by the AS method, the activity score measures how important, or how "active", each input variable is. Here, the activity scores for x_1 and x_2 reflect the same information provided by their sensitivity measurements. That is, each variable is equally active in determining the function's output.

Finally, we visualize the AS transformation by illustrating the original parameter space in (e), followed by the transformed parameter space in (f). We see that the AS transformation is a simple 45 degree rotation - a linear transformation. Of course, any transformation produced by the AS method will be linear because it will use linear combinations of the active eigenvectors to form a subspace. This further highlights the need for the NLL method, which will be able to learn nonlinear as well as linear transformations, providing some valuable flexibility in our pursuit of dimension reduction.

2.2 Gradient Evaluation

In practice, it is not always possible or preferred to fully compute a large sample of gradients every time the AS method is used. Thus, because gradients are a crucial part of the $\hat{C}$ matrix that is used to find the contents of the active subspace, it is necessary to have methods of approximating or estimating gradients, along with methods to verify the accuracy of these estimations. We focus on two main approaches to gradient approximation: Gaussian regression and local linear models. For both methods, there are important principles that can be followed to limit error, such as increasing sample size and decreasing step size when estimating gradients. Using a sine function for an example, below we visualize the results of the AS method using exact gradients, followed by gradient approximation with Gaussian process regression and local linear models.

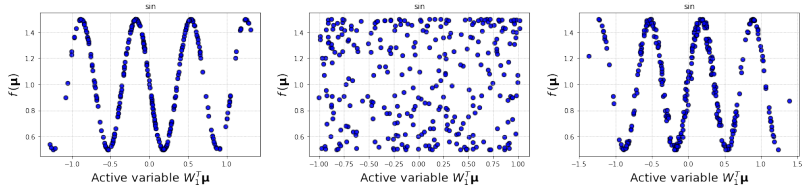


Figure 3: Sufficient summary plots using the AS method on a sine function with exact gradient calculation (left), approximation using Gaussian process regression (middle), and approximation using local linear models (right). All three approaches use 300 samples.

The sufficient summary plots indicate that dimension reduction using the AS method achieves good performance with exact gradients and with gradients approximated using local linear models. However, gradient approximation using Gaussian process regression demonstrates very poor performance. This is because, in the example given by the ATHENA code, the 300 samples taken are insufficient for the Gaussian process, and error analysis shown in the code demonstrates that the model is not accurate until at least 700 samples are taken. Also, it is important to note that while the local linear approximation appears good, it still would benefit in accuracy from an increased number of samples. Below, we see the sufficient summary plot produced when 700 samples are taken with Gaussian process regression.

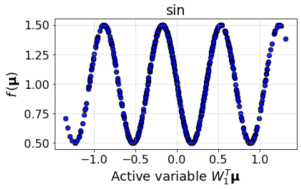


Figure 4: The sufficient summary plot using 700 samples, indicating much better performance in the dimension reduction.

With these examples, we have demonstrated that the AS method can still perform well when using approximated gradients, provided that a sufficiently large sample size is used.

2.3 Response Surfaces

Knowing that the AS method can be used to reliably reduce the dimension of a given function, an additional question we could ask is how to make use of an active subspace. One application, developed in [3], is in the training of a response surface - a process of estimating the relationship between several explanatory variables and one or more response variables. In particular, we focus on regression models, which provide a more smooth approximation than methods like exact interpolation.

Suppose we want to construct a response surface for a function f whose domain of interest is $\mathbb{X}$. Recall the partition of $\boldsymbol{W}$ into $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, and say the number of columns in $\boldsymbol{W}_1$ (the dimension of the active subspace) is n. Defining new variables $\boldsymbol{y} = \boldsymbol{W}_1^T\boldsymbol{x}$ and $\boldsymbol{z} = \boldsymbol{W}_2^T\boldsymbol{x}$, it can be shown that any $\boldsymbol{x}$ can be written in the form $\boldsymbol{x} = \boldsymbol{W}_1\boldsymbol{y} + \boldsymbol{W}_2\boldsymbol{z}$. Thus, $f(\boldsymbol{x}) = f(\boldsymbol{W}_1\boldsymbol{y} + \boldsymbol{W}_2\boldsymbol{z})$, and by the construction of the active subspace, changes in $\boldsymbol{z}$ have a relatively small effect on the value of f. With this, define $g(\boldsymbol{y}) = f(\boldsymbol{W}_1\boldsymbol{y})$. To approximate $f(\boldsymbol{x})$, we could use $g(\boldsymbol{W}_1^T\boldsymbol{x}) = f(\boldsymbol{W}_1\boldsymbol{W}_1^T\boldsymbol{x})$, the value of f at the projection of x onto the column space of $\boldsymbol{W}_1$, which is the active subspace.

For the purpose of constructing a response surface, the domain of g is defined as

$$\mathbb{Y} = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{y} = \boldsymbol{W}_1^T\boldsymbol{x}, \boldsymbol{x} \in \mathbb{X}\},$$

and g is defined as the conditional expectation of f given $\boldsymbol{y}$:

$$g(\boldsymbol{y}) = \int f(\boldsymbol{W}_1\boldsymbol{y} + \boldsymbol{W}_2\boldsymbol{z})\pi_{Z|Y}(\boldsymbol{z})d\boldsymbol{z}.$$

Here, $\pi_{Z|Y}$ is the conditional probability density of Z given Y. (When the density for the input parameters is Gaussian, this conditional density is also Gaussian). The conditional expectation provides the best possible guess for the value of f when given $\boldsymbol{y}$. In practice, though, it is approximated using Monte Carlo estimation:

$$\hat{g}(\boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{W}_1\boldsymbol{y} + \boldsymbol{W}_2\boldsymbol{z}_i),$$

where the $\boldsymbol{z}_i$ are randomly drawn from the conditional density $\pi_{Z|Y}$.

The response surface is fitted using the pairs $\{\boldsymbol{y}_j, \hat{g}(\boldsymbol{y}_j)\}$. That is, for $j = 1, 2, \ldots M$, we sample $\boldsymbol{y}_j$ (such that $\boldsymbol{y}_j \in \mathbb{Y}$) and approximate the corresponding conditional expectation of f. After fitting the response surface, we can approximate $f(\boldsymbol{x})$ by obtaining the value of the response surface at $\boldsymbol{W}_1^T\boldsymbol{x}$.

Note that the domain of $\hat{g}$, $\mathbb{Y}$, is of lower dimension than the domain of f, $\mathbb{X}$. Thus, this procedure allows us to fit a response surface on a lower-dimensional domain in order to approximate a function of higher dimensions. And the properties of the active subspace ensure that, under the appropriate conditions, this will be a reasonable approximation. Thus, the active subspace allows us to study the behavior of a complex function (in particular, the relationship between the explanatory variables and the response) using fewer computational resources.

3 The Nonlinear Level Set Learning Method

3.1 Introduction

First, suppose we have a d-dimensional function, $f(\boldsymbol{x})$, which we wish to approximate in reduced dimensions. And suppose that f has a bounded domain, $\Omega \subset \mathbb{R}^d$, with a probability density function, ρ, that can be used to sample elements from the domain. The goal is to create a bijective nonlinear transformation, $\boldsymbol{g} : \Omega \to \mathbb{R}^d$. For any $\boldsymbol{x} \in \Omega$, we want $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{x} = \boldsymbol{g}^{-1}(\boldsymbol{z})$ so that $\boldsymbol{z}$ has only a small number of "active" input components. That is, we want to be able to write $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2)$, where $f \circ \boldsymbol{g}^{-1}$ is sensitive only to changes in $\boldsymbol{z}_1$ and $dim(\boldsymbol{z}_1) << d$. Note that $f \circ \boldsymbol{g}^{-1}(\boldsymbol{z}) = f(\boldsymbol{x})$, so approximating $f \circ \boldsymbol{g}^{-1}$ using only its active components (held in $\boldsymbol{z}_1$) will allow us to approximate f.

Knowing how we want the nonlinear transformation, $\boldsymbol{g}$, to behave, we can now explore the method for producing it. As previously stated, the transformation is produced using a RevNet, because the reversible architecture allows for the production of a bijective (and therefore invertible) transformation. However, the

transformation should not only be invertible; it should also produce points that have a small number of active components, and ideally it should be volume-preserving as well. These requirements guide the design of a proper loss function for the neural network. Using such a loss function, the neural network can be trained with the sample data

$$\left\{\left(\boldsymbol{x}^{(s)}, f(\boldsymbol{x}^{(s)}), \nabla f(\boldsymbol{x}^{(s)})\right) : s = 1, \ldots, S\right\},$$

where $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(S)}$ are drawn from the density ρ.

3.2 The Neural Network

As mentioned above, the neural network architecture used for this approach is called a nonlinear RevNet, which allows for training an invertible function. The structure of this network is as follows.

$$\begin{cases} \boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\boldsymbol{K}_{n,1}^T \boldsymbol{\sigma}(\boldsymbol{K}_{n,1}\boldsymbol{v}_n + \boldsymbol{b}_{n,1}) \\ \boldsymbol{v}_{n+1} = \boldsymbol{v}_n - h\boldsymbol{K}_{n,2}^T \boldsymbol{\sigma}(\boldsymbol{K}_{n,2}\boldsymbol{u}_{n+1} + \boldsymbol{b}_{n,2}) \end{cases},$$

for $n = 0, 1, \ldots, N-1$. Here, $\boldsymbol{u}_n, \boldsymbol{v}_n$ form a partition of the inputs, $h \in \mathbb{R}$ is the time step, and $\boldsymbol{\sigma}$ is the activation function, usually chosen as a hyperbolic tangent function in practice. $\boldsymbol{K}_{n,1}, \boldsymbol{K}_{n,2}$ are the weight matrices, and $\boldsymbol{b}_{n,1}, \boldsymbol{b}_{n,2}$ are the bias vectors.

This network structure yields the transformation $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{z}$ if we define $\boldsymbol{x}$ and $\boldsymbol{z}$ as follows:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{v}_0 \end{bmatrix}, \boldsymbol{z} = \begin{bmatrix} \boldsymbol{u}_N \\ \boldsymbol{v}_N \end{bmatrix}.$$

Training the network, and thus obtaining the desired nonlinear transformation, is done by updating the weight matrices and bias vectors according to the appropriate loss function, which is outlined next.

3.3 The Loss Function

The loss function will have two components. The first is meant to reduce the number of active dimensions, and the second is meant to ensure that the transformation is invertible and volume-preserving.

The construction of the first component is guided by the key fact that a function's gradient is orthogonal to its level sets - that is, any tangent direction of a level set is perpendicular to the gradient direction. And for some $\boldsymbol{x} = \boldsymbol{g}^{-1}(\boldsymbol{z})$, if $\boldsymbol{x}$ moves along a tangent direction of the level set at $f(\boldsymbol{x})$ when the i^{th} component of $\boldsymbol{z}$, z_i, is changed, then the output of $f(\boldsymbol{x})$ does not change with a perturbation of z_i in the neighborhood of $\boldsymbol{z}$. Thus, if $\boldsymbol{x}$ moves in a perpendicular direction to the gradient when z_i is perturbed, the value of f does not change with perturbations of z_i in the neighborhood of $\boldsymbol{z}$. That is, z_i is an "inactive" component. And we want the transformation to produce a certain number of these inactive components.

To this end, consider the Jacobian matrix of the inverse transformation $\boldsymbol{g}^{-1} : \boldsymbol{z} \to \boldsymbol{x}$,

$$\boldsymbol{J}_{\boldsymbol{g}^{-1}}(\boldsymbol{z}) = [\boldsymbol{J}_1(\boldsymbol{z}), \boldsymbol{J}_2(\boldsymbol{z}), \ldots, \boldsymbol{J}_d(\boldsymbol{z})], \text{where } \boldsymbol{J}_i(\boldsymbol{z}) = \left(\frac{\partial x_1}{\partial z_i}(\boldsymbol{z}), \frac{\partial x_2}{\partial z_i}(\boldsymbol{z}), \ldots, \frac{\partial x_d}{\partial z_i}(\boldsymbol{z})\right)^T.$$

The movement of $\boldsymbol{x}$ with respect to z_i (in a neighborhood of $\boldsymbol{z}$) is merely $\boldsymbol{J}_i(\boldsymbol{z})$, so z_i is inactive if the inner product of $\boldsymbol{J}_i(\boldsymbol{z})$ and the gradient, $\langle \boldsymbol{J}_i(\boldsymbol{z}), \nabla f(\boldsymbol{x}) \rangle$, is zero. Thus, reducing the active dimension means finding many components z_i which satisfy this condition, which would suggest that a good approach is to minimize these inner products. Following from this idea, the first component of the loss function is

$$L_1 = \sum_{s=1}^{S} \sum_{i=1}^{d} \left[\omega_i \left\langle \frac{\boldsymbol{J}_i(\boldsymbol{z}^{(s)})}{\|\boldsymbol{J}_i(\boldsymbol{z}^{(s)})\|_2}, \nabla f(\boldsymbol{x}^{(s)}) \right\rangle\right],$$

where $\omega_1, \ldots, \omega_d$ are user-specified weights that determine how strongly we want to minimize the inner product for each dimension. For example, if $\boldsymbol{\omega} = (0, 0, 1, 1, ..., 1)$, then the goal is to produce a transformation $\boldsymbol{g}$ that transforms elements of the domain so that all but the first two components are inactive. That is, $\boldsymbol{g}$ would allow us to approximate f in two dimensions.

The second component of the loss function is included simply to control the properties of the transformation. Namely, we want $\boldsymbol{g}$ to be invertible and volume-preserving. A transformation is invertible near a point if the Jacobian determinant at that point is nonzero, and the transformation preserves volume near a point if the Jacobian determinant at that point is one. To coerce our transformation to have these properties, the goal would be to minimize

$$L_2 = (\det(\boldsymbol{J}_{\boldsymbol{g}^{-1}}) - 1)^2,$$

which becomes the second component of the loss function.

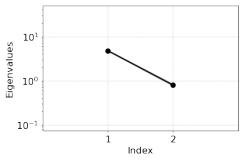
Thus, the loss function for the NLL approach to dimension reduction is

$$L = L_1 + \lambda L_2,$$

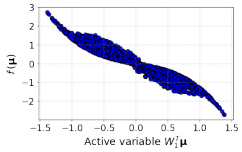
where λ is a user-specified constant used to balance the weight of the two terms (we usually choose $\lambda = 1$ in practice).

3.4 A Simple Example

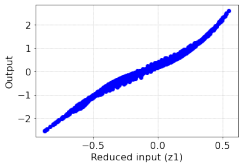
Consider $f(\boldsymbol{x}) = x_1^3 + x_2^3 + 0.2x_1 + 0.6x_2$, a polynomial function of two variables. We apply the AS and NLL methods in an attempt to reduce it to one active input dimension, with the results shown in the figures below.



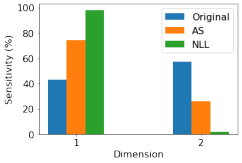
(a) AS eigenvalues



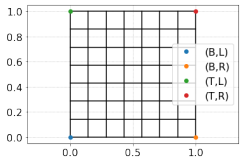
(b) AS sufficient summary



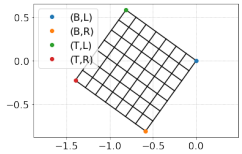
(c) NLL sufficient summary



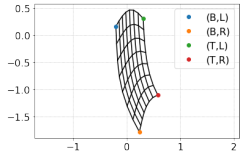
(d) Sensitivity chart



(e) Original parameter space



(f) AS transformation



(g) NLL transformation

Figure 5: Output plots visualizing the performance of the AS and NLL transformations when applied to f.

In this example, applying the AS method does not achieve a very reliable dimension reduction. First, the eigenvalue plot in (a) does not exhibit a very large gap between the first and second eigenvalues; the difference is smaller than one order of magnitude. And the sufficient summary plot in (b) shows considerable variability in the function's value along the active subspace direction. Thus, the function's value cannot be reliably observed by only looking at input parameters in the active subspace - the dimension reduction using the AS method is not effective.

Since the AS method performs poorly for this function, we also apply the NLL method with the hope that it can train a more apt transformation. Indeed, we see in the sufficient summary plot in (c) that the NLL method appears to perform better. This plot demonstrates a stronger trend than the plot for the AS method, with less variability in the function's value at any given point of the reduced input. That is, the active dimension produced by the NLL method does a better job at explaining the value of the function than

the active subspace does. This conclusion is also supported by the sensitivity chart in (d), which illustrates that the NLL transformation concentrates almost 100% of the function's sensitivity into one dimension (the active dimension), whereas the AS transformation is only able to concentrate about 75% sensitivity into the active subspace dimension. That is, the function can be well-approximated by only considering input parameters in the single active dimension specified by the NLL transformation. Meanwhile, trying to study the function using only the active subspace may cause problems, since the function is still fairly sensitive to the second, supposedly "inactive," dimension.

Finally, we can visualize the behavior of the AS and NLL transformations by illustrating their effect on the original parameter space, which is shown in (e). The AS transformation is given in (f); it appears to be a simple rotation of the parameter space. Again, the AS method can only produce linear transformations. The NLL transformation, on the other hand, has clear nonlinear behavior, as shown in (g). Comparing the two transformations shows the flexibility of the NLL method, which can learn both linear and nonlinear transformations. Unfortunately, this flexibility does come with some costs. The NLL method requires the training of a neural network, which is considerably more expensive than the the eigendecomposition required for the AS method. And while both methods require the evaluation of gradients, the NLL method also requires computation of Jacobians and Jacobian determinants. So, while the NLL method can be applied to a wider variety of functions than the AS method, it also requires more computational resources.

4 Real-World Applications

We now turn our attention to more practical applications of the AS and NLL methods. We consider three functions with applications in engineering: a wing weight function, a piston simulation function, and a circuit function. We will compare the efficacy of a dimension reduction provided by the AS method versus that provided by the NLL method, keeping in mind the benefits and drawbacks of each method.

4.1 Wing Weight Function

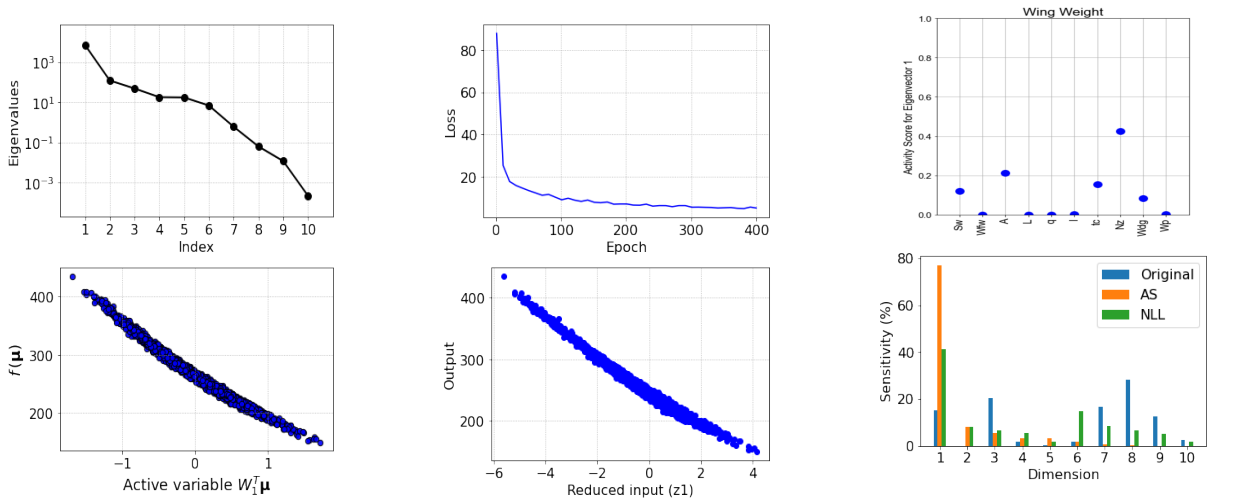
The wing weight function, retrieved from [5], is a function of 10 parameters that models the weight of an aircraft wing. The function is given by

$$f(x) = 0.036 S^{0.758} W_f^{0.0035} \left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100t}{\cos(\Lambda)}\right)^{-0.3} (NW_d)^{0.49} + SW_p,$$

with the parameters defined as follows.

Variable	Parameter	Lower bound	Upper bound
S	wing area (ft^2)	150	200
W_f	weight of fuel in the wing (lb)	220	300
A	aspect ratio	6	10
Λ	quarter-chord sweep (degrees)	-10	10
q	dynamic pressure at cruise (lb/ft^2)	16	45
λ	taper ratio	0.5	1
t	aerofoil thickness to chord ratio	0.08	0.18
N	ultimate load factor	2.5	6
W_d	flight design gross weight (lb)	1700	2500
W_p	paint weight (lb/ft^2)	0.025	0.08

The results of the AS method and the NLL method for reducing the dimension of this function are shown below.



(a) AS method

(b) NLL method

(c) Activity score and sensitivity comparison

Figure 6: Dimension reduction for the wing weight function using the AS and NLL methods.

The results of the AS method are depicted in (a), where a fairly large gap (close to two orders of magnitude) can be observed between the first and second eigenvalues in the eigenvalue plot. As a result, it is reasonable to search for an active subspace of dimension one that can successfully reduce the dimension of the wing function. Using the first eigenvector direction for the active subspace, the sufficient summary plot exhibits a strong univariate trend, with fairly low variability in the function's value at any given point along the active subspace direction. That is, it appears that the wing weight function can be well-approximated by only considering input parameters in the one-dimensional active subspace.

Even though the AS method performs well for this function, it is still worthwhile to apply the NLL method in the interest of comparing the performance of the two approaches. In (b), we see that the loss for the NLL method's neural network decreases rapidly, indicating that the network converges well. In addition, the sufficient summary plot for the NLL method, like the plot for the AS method, looks good. It shows a strong trend, with very little variability in the function's value at points along the active dimension. However, a sufficient summary plot is not the only tool that we can use to evaluate the effectiveness of a dimension reduction. Looking at the sensitivity chart in (c), we see that the AS technique does a good job at explaining the function's value in a one-dimensional parameter space (with close to 80% relative sensitivity in the active dimension), while the NLL performs considerably worse (only about 40% relative sensitivity in the active dimension). The sensitivity metrics for the NLL method actually go against the conclusion suggested by the sufficient summary plot, demonstrating that it is important to take multiple tools into account when analyzing the AS or NLL methods.

Overall, the AS method is the more reliable choice for the wing weight function, especially considering that it incurs fewer computational costs than the NLL method. With the AS method, it is possible to study the wing weight function, a function of 10 parameters, in only 1 input dimension, marking a significant improvement.

4.2 Piston Simulation Function

The piston simulation function, retrieved from [6], is a function of seven parameters that models the time it takes for a piston to complete one cycle. The function is given by

$$C(\boldsymbol{x}) = 2\pi\sqrt{\frac{M}{k + S^2\frac{P_0V_0}{T_0}\frac{T_a}{V^2}}}, \text{ where}$$

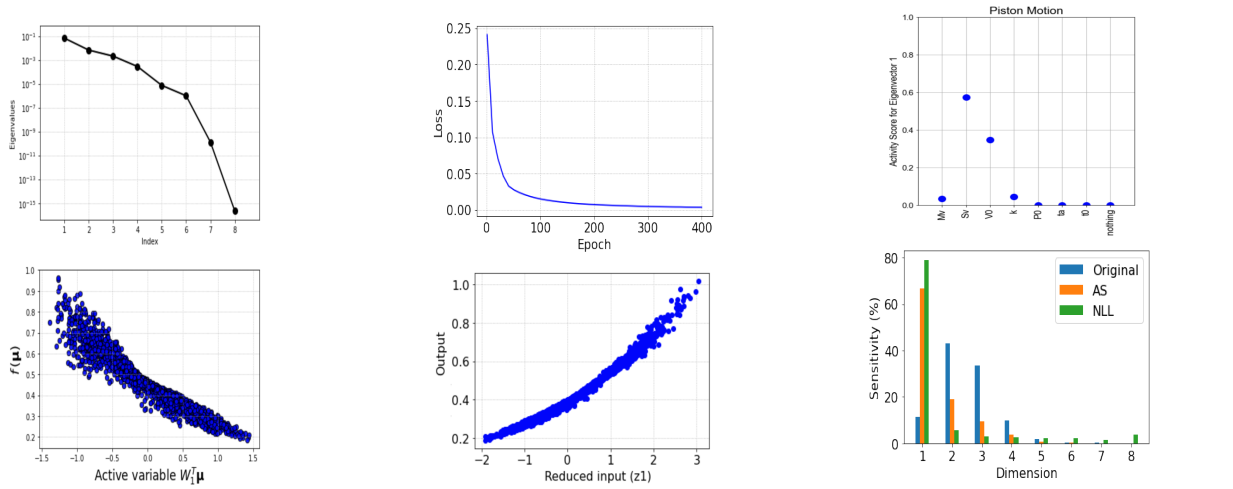
$$V = \frac{S}{2k}\left(\sqrt{A^2 + 4k\frac{P_0 V_0}{T_0}T_a} - A\right) \text{ and}$$

$$A = P_0 S + 19.62M - \frac{kV_0}{S}.$$

The parameters are defined as follows.

Variable	Parameter	Lower bound	Upper bound
M	piston weight (kg)	30	60
S	piston surface area (m^2)	0.005	0.020
V_0	initial gas volume (m^3)	0.002	0.010
k	spring coefficient (N/m)	1000	5000
P_0	atmospheric pressure (N/m^2)	90000	110000
T_a	ambient temperature (K)	290	296
T_0	filling gas temperature (K)	340	360

The results of the AS method and the NLL method for reducing the dimension of this function are shown below.



(a) AS method

(b) NLL method

(c) Activity score and sensitivity comparison

Figure 7: Dimension reduction for the piston function using the AS and NLL methods. *Note: An eighth variable was introduced as a dummy variable for the purpose of training the NLL neural network, which requires an even number of inputs.*

The results of the AS method for reducing the piston function to one active dimension are not very promising. First, the eigenvalue plot does not display a significant gap until the sixth and seventh eigenvalues. This indicates that we could find an active subspace of dimension six, but reducing the function from seven input dimensions to six would not be worthwhile. For the sake of easily visualizing the results, we attempt to use an active subspace of dimension one to study the piston function. The resulting sufficient summary plot illustrates the dangers of using a one-dimensional subspace, as the function value exhibits considerable variability at points along the active subspace direction. Perhaps a two- or three-dimensional active subspace could achieve better results, but visualizing such subspaces with sufficient summary plots becomes quite difficult.

Because an active subspace of dimension one does not work very well here, we attempt the NLL method with hopes of seeing an improvement. As in the wing function example, the loss for the NLL method's neural network decays rapidly, indicating the network converges well. And the sufficient summary plot is

also promising - it shows a stronger trend than the plot generated from the active subspace method, with considerably less variability in the function value at points in the active dimension. Looking only at the sufficient summary plots, it appears that the NLL method performs significantly better at reducing the piston function to one active dimension in the parameter space. To reinforce this conclusion, though, it is also important to examine the sensitivity chart in (c). In this chart, we see that close to 80% relative sensitivity is concentrated in one dimension upon applying the NLL transformation to the piston function, whereas the AS transformation only achieves a little over 60% sensitivity in one dimension. Thus, the piston function is a practical example of a function whose behavior lends itself better to nonlinear dimension reduction techniques than linear ones. Here, the NLL method does a good job at reducing the piston function into one active dimension in the parameter space, which could allow for the function to be studied using fewer computational resources.

4.3 OTL Circuit Function

The OTL circuit function, retrieved from [7], is a function of six parameters that models the midpoint voltage of an OTL circuit. The function is given by

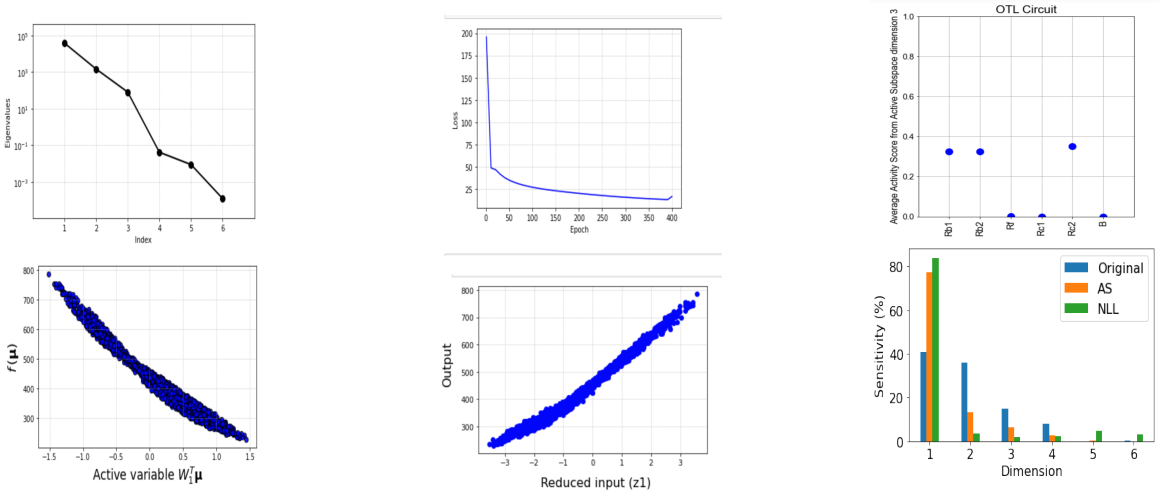
$$V_m(\boldsymbol{x}) = \frac{(V_{b1}+0.74)\beta(R_{c2}+9)}{\beta(R_{c2}+9)+R_f} + \frac{11.35R_f}{\beta(R_{c2}+9)+R_f} + \frac{0.74R_f\beta(R_{c2}+9)}{(\beta(R_{c2}+9)+R_f)R_{c1}}, \text{ where}$$

$$V_{b1} = \frac{12R_{b2}}{R_{b1}+R_{b2}}.$$

The parameters are defined as follows.

Variable	Parameter	Lower bound	Upper bound
R_{b1}	resistance b1 (K-Ohms)	50	150
R_{b2}	resistance b2 (K-Ohms)	25	70
R_f	resistance f (K-Ohms)	0.5	3
R_{c1}	resistance c1 (K-Ohms)	1.2	2.5
R_{c2}	resistance c2 (K-Ohms)	0.25	1.2
β	current gain (Amperes)	50	300

The results of the AS method and the NLL method for reducing the dimension of this function are shown below.



(a) AS method

(b) NLL method

(c) Activity score and sensitivity comparison

Figure 8: Dimension reduction for the circuit function using the AS and NLL methods.

Applying the AS method to the circuit function produces some interesting results. First, the eigenvalue plot shows a large gap between the third and fourth eigenvalues, indicating that a three-dimensional active subspace may be appropriate. Unfortunately, visualizing a sufficient summary plot using three active dimensions would be very difficult (if not impossible), so for the purpose of this analysis we again use a one-dimensional active subspace. If the setting were more realistic, and we were truly interested in studying the circuit function in lower dimensions, we would likely choose a three-dimensional active subspace. For now, though, we are satisfied simply with visualizing how the method works, which benefits from working in lower dimensions. With that being said, the sufficient summary plot using a one-dimensional active subspace still looks promising, with a fairly strong trend and a small amount of variability, which is noticeable but not excessive. Even though the eigenvalue plot suggests the presence of a three-dimensional active subspace, this sufficient summary plot indicates that a one-dimensional subspace performs decently as well.

Applying the NLL method, we again see that the loss of the neural network decays rapidly, as desired. The sufficient summary plot also behaves desirably, with a fairly strong univariate trend and only a small amount of variability (less variability than the active subspace plot). This plot suggests that the NLL method performs well at reducing the active parameter space of the piston function to a single dimension. To add more confidence to our conclusions, though, we again observe the sensitivity chart. Here, we see that both methods indeed perform well at explaining the function's value using only one input dimension. The NLL method is able to concentrate just over 80% relative sensitivity into one dimension, while the AS method concentrates just under 80% into one dimension. As was the case for the wing and piston functions, we have found an effective way to reduce the circuit function to one input dimension.

Returning to the idea of a three-dimensional active subspace, though, we can see that nearly all of the sensitivity for the AS method is concentrated in the first three dimensions, with the function exhibiting very little sensitivity to the remaining three dimensions. Not only does this further support the idea of using a three-dimensional active subspace for this specific piston function, but it also demonstrates the general importance of looking beyond one dimension when applying these techniques. Although sufficient summary plots can only be utilized in very low dimensions, tools like eigenvalue plots and sensitivity charts can be useful for exploring a higher number of active dimensions in the parameter space. Perhaps a dimension reduction does a poor job of concentrating a function's relative sensitivity all into one dimension, but it does a great job at concentrating the sensitivity into two or three dimensions. While it is not as straightforward to deal with extra dimensions, this would still be important information in a practical study of a high-dimensional function. It is very unlikely that a function with many parameters will be easily reduced to one active dimension; a more in-depth analysis involving higher dimensions will probably be necessary.

5 Conclusion

In this report, we have given general explanations of the AS and NLL methods for dimension reduction, with a focus both on their mathematical development as well as on the intuition that supports them. We pointed out the important differences between the two methods, including the benefits and drawbacks that come with each. To support our exploration of these techniques, we also supplied two simple examples that made it easy to visualize how the methods performed when attempting to reduce a function's active dimension.

Finally, we applied the AS and NLL methods to three high-dimensional functions with applications in engineering, demonstrating the methods' effectiveness when applied to functions with practical applications, not just simple test functions. In these examples, we included comparisons between the performance of the two methods in order to emphasize the fact that some functions can be easily reduced using linear methods, while others require a nonlinear approach.

In short, this report helps provide an introduction to the subject of dimensionality reduction in the parameter space by thoroughly exploring two prominent techniques in the field.

References

[1] Paul Constantine. "Active Subspaces". In: *Software Impacts* (2010). DOI: 10.5281/zenodo.158941.

[2] Paul Constantine. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*. SIAM Spotlights. Society for Industrial and Applied Mathematics, 2015. ISBN: 9781611973860. URL: https://books.google.com/books?id=AuJ9BwAAQBAJ.

[3] Paul Constantine, Eric Dow, and Qiqi Wang. "Active Subspace Methods in Theory and Practice: Applications to Kriging Surfaces." In: *SIAM Journal on Scientific Computing* 36 (2014).

[4] Francesco Romor, Marco Tezzele, and Gianluigi Rozza. "ATHENA: Advanced Techniques for High dimensional parameter spaces to Enhance Numerical Analysis". In: *Software Impacts* 10 (2021), p. 100133. DOI: 10.1016/j.simpa.2021.100133.

[5] Sonja Surjanovic and Derek Bingham. "Virtual Library of Simulation Experiments: Test Functions and Datasets". In: (2013). URL: http://www.sfu.ca/~ssurjano/wingweight.html.

[6] Sonja Surjanovic and Derek Bingham. "Virtual Library of Simulation Experiments: Test Functions and Datasets". In: (2013). URL: https://www.sfu.ca/~ssurjano/piston.html.

[7] Sonja Surjanovic and Derek Bingham. "Virtual Library of Simulation Experiments: Test Functions and Datasets". In: (2013). URL: https://www.sfu.ca/~ssurjano/otlcircuit.html.

[8] Guannan Zhang, Jiaxin Zhang, and Jacob Hinkle. *Learning nonlinear level sets for dimensionality reduction in function approximation*. 2019. DOI: 10.48550/ARXIV.1902.10652. URL: https://arxiv.org/abs/1902.10652.