2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

# Constrained Nonlinear Control Allocation based on Deep Auto-Encoder Neural Networks

Huang Huang[1], Wei Wang[2], Chunling Wei[1], and Yingzi He[1]

*Abstract*— **An intelligent control allocation method for modern reentry vehicles with a strong aerodynamic nonlinearity is presented in this paper. A specially designed deep auto-encoder (DAE) neural network is proposed that shares similar information flow with the control allocation process. This similarity is showed by setting the decoder to approximate the control effector function, and by setting the encoder to allocate the expected control moments. The decoder is trained independently based on the aerodynamic coefficients database, and with the help of the well-trained decoder, the encoder is then trained in an unsupervised way without labeled data. This proposed control allocation method could deal with strong nonlinearity of the control effectors at a high accuracy, thanks to the powerful modeling and regression ability of deep neural networks. Numerical examples are provided by the end that explain the training and implementation details, as well as the strong learning and modeling ability of the deep neural network.**

## I. Introduction

Aiming at stronger robustness, maneuverability, and reliability, modern reentry vehicles are equipped with a redundant set of effectors. A control allocation deals with this redundancy by allocating or distributing the desired control moments among these effectors. Over the past few decades, various control allocation algorithms have been proposed [1], some of which have already been testified by flight tests, such as the daisy chaining on the X-38 test vehicle [2]. Other allocation algorithms have also been well-studied in literature, including the direct control allocation, the redistributed pseudoinverse, the quadratic programming, the interior-point algorithm, et. al.. Some of the algorithms were developed towards efficient and simple onboard implementation, such as the linear programming technique [3], [4] and the fixed-point iteration algorithm [5]. Almost all of these allocation methods were discussed under one common assumption that the control effectiveness mapping was *linear*, or sometimes was linear with uncertainties [6].

It was recently discovered that the nonlinearity between the aerodynamic effectors and the resulted moments is phenomenal at some angle of attacks and effector deflections [7]. Meanwhile, with an increasingly high level of maneuverability requirement, reentry vehicles are expected to fly at large angle of attacks so as to attain sufficient overloads, or to perform a fast roll over maneuver to change their trajectories. In such cases, the control effectors reach their position limits or rate limits, and the nonlinear dynamics characteristic of effectors is so evident that the control effectiveness mapping is too nonlinear to be neglected [8]. If the nonlinearity of the control effectiveness mapping is linearized, then the allocation errors could be large. Even if they could be mitigated by the robustness of the inner controller, the closed-loop performance could be degraded. In some cases, large allocation errors may even jeopardize the overall stability.

The problem of nonlinear control allocation is challenging due to the possibility of local minimums and computational issues [9], and so far literature on this subject is limited. One way is to treat the nonlinearity as a time-varying affine model, as discussed in [10] and [11], where the bias was assumed known *a priori*. Another way, which is more intuitive, is to treat the nonlinear mapping directly through the sequential quadratic programming approach [12]. Although the computational complexity could be overwhelming, this was a meaningful attempt towards a high allocation accuracy. The nonlinearity could also be treated as a piecewise linear problem, as discussed in [11].

This paper addresses the nonlinear control allocation problem using the deep neural network. Early efforts in the 1990s [13] were limited by network scale and flexibility due to network convergence issues. With the development of deep learning in recent years, a neural network can grow much deeper and much larger than ever before with improved convergency, which opens a new window to modeling and optimization. In this work, a multi-layer perceptron (MLP) is built to model the nonlinearity of the control effectors. With this well-trained MLP in hand, a deep auto-encoder (DAE) is further built to allocate the expected moments to the available control effectors, i.e., to solve the allocation problem. The unique feature of the DAE that reconstructs the input at the output allows the network to be trained without any labeled data, which overcomes the main defects in the 1990s. Numerical examples are provided by the end that explain the training and implementation details, as well as the strong learning and modeling ability of the deep neural network.

## II. Background

### A. Control Allocation

In the attitude control of modern reentry vehicles, control allocation is designed to allocate the desired control moments

This work was supported by the National Natural Science Foundation of China (Grant No. 61203075, Grant No. 61333008), and by Beijing Natural Science Foundation (Grant No. 4172070)

[1] Science and Technology on Space Intelligent Control Laboratory, Beijing Institute of Control Engineering, P.O. Box 2729, Beijing 100190, China hhuang33@gmail.com

[2] Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100094, China wangwei@nlpr.ia.ac.cn

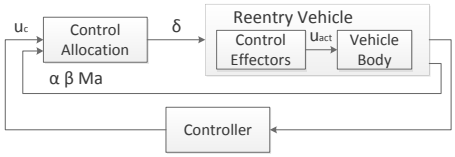



Fig. 1. Attitude control system of a reentry vehicle.

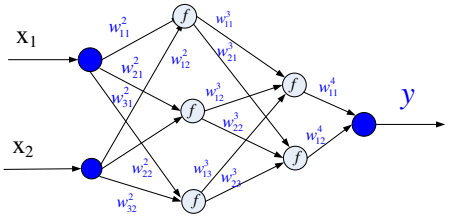



Fig. 2. An MLP of $K = 4$ layers. The number of hidden layers is 2.

to the control effectors of the vehicle, as shown in Fig. 1. By introducing an allocation modular, controller design is separated from aerodynamic control surfaces that could bring over-actuation and nonlinearity. Meanwhile, control allocation could strengthen the robustness of the controller with respect to vehicle effectors change and aerodynamic environment change.

The mapping from control effectors to actual control moments acting on the vehicle body at different flight conditions is described by

$$\mathbf{u}_{act} = \overline{q}SL\mathbf{C} \tag{1}$$

where $\mathbf{u}_{act} \in \mathbf{R}^{3\times 1}$ is a vector with three real entries representing the expected control moments on roll, pitch, and yaw of the vehicle. The dynamic pressure $\overline{q}$ is determined by the velocity and the altitude, and $S, L$ are the reference surface area and the reference aerodynamic length, respectively. $\mathbf{C} \in \mathbf{R}^{3\times 1}$ consists of the equivalent aerodynamic coefficients on the three axes and is determined by

$$\mathbf{C} = \mathbf{g}(\delta, \alpha, \beta, Ma) \tag{2}$$

where $\delta \in \mathbf{R}^{n\times 1}$ is the $n$ available aerodynamic control effectors including, for example, elevons, elevators, rudders, body flaps, et. al.. $\alpha$ and $\beta$ are the angle of attack and the side-slip angle, respectively, and $Ma$ is the Mach number. In a linear assumption, $\mathbf{C} = \mathbf{B}\delta$, where matrix $\mathbf{B}$, known as the control effectiveness matrix, is calculated at a specific flight condition. Without loss of generality, function $\mathbf{g}$ is referred to as a *control effector function*.

Given an expected control moment $\mathbf{u_c} \in \mathbf{R}^{3\times 1}$, the control allocation solves a constrained optimization problem over the variable $\delta$, that is,

$$L = \min_{\delta} \|\mathbf{u_c} - \overline{q}SL\mathbf{g}(\delta, \alpha, \beta, Ma)\| \tag{3}$$

with an amplitude constraint $|\delta| \le \bar{\delta}$ and sometimes a rate limit $|\dot{\delta}| \le \bar{\dot{\delta}}$.

Throughout the paper, it is assumed that

*Assumption 1:* The expected control moments $\mathbf{u_c}$ are attainable.

It is worth noting that although this separated design procedure is commonly accepted by scholars and engineers, the closed-loop performance might be quite different depending on the allocation methods [14], [15]. Sometimes allocation may result in zero dynamics [16], or even jeopardize the closed-loop stability. A closed-loop allocation method was proposed in [17] for spacecraft attitude stabilization. In our previous work [18], a sufficient criterion on the maximum allowable allocation errors was provided to guarantee the closed-loop stability.

B. Deep Auto-Encoder

The most fundamental form of neural networks is a feed-forward neural network with hidden layers. A feed-forward network with multiple hidden layers is also known as a multi-layer perception (MLP), as shown in Fig. 2. An MLP with $K$ layers could be denoted by

$$\Xi = \{[n_1, n_2, \ldots, n_K], f(x), o(x), J\} \tag{4}$$

where $n_i$ is the number of neurons of layer $i$, $f(x)$ is the activation function, $o(x)$ is the output function, and $J$ is the loss function that describes the performance of the network. In an MLP, information is reproduced at each neuron by the activation function $f(x)$. Even though the activation function is a simple one, the function represented by an MLP could be sufficiently complex. Neuron $j$ at layer $l$ passes information according to

$$a_j^l = f(\sum_k w_{jk}^l a_k^{l-1} + b_j^l)$$

with the vector form

$$\mathbf{a}^l = \mathbf{f}(\mathbf{w}^l \mathbf{a}^{l-1} + \mathbf{b}^l), \tag{5}$$

where $a_j^l$ is the activation of the neuron, $w_{jk}^l$ is the weight connecting neuron $k$ in layer $l-1$ to neuron $j$ in layer $l$, and $b_j^l$ is the bias being added to neuron $j$.

An MLP is trained in a supervised manner by labeled data. Paired input-output data has to be collected before training. The divergence of the training data is crucial to the generalization and modelling accuracy of the MLP.

If two MLPs are connected by overlapping one MLP's output to another MLP's input, an auto-encoder network is obtained. An auto-encoder is a kind of neural network that transcribes the unstructured data into some representative form. It abstracts interested information from the input data. An auto-encoder consists of an encoder and a decoder, as shown in Fig. 3. The input data $x$ is transcribed into *code* by the encoder, which is a high-level representation of the input data, and the decoder maps the *code* back to its origin. In other words, the input $x$ is reconstructed at the output $\hat{x}$ through an encoder and a decoder, that is,

$$\hat{\mathbf{x}} = \text{decoder}(code) = \text{decoder}(\text{encoder}(\mathbf{x})) \tag{6}$$

The goal is to train an encoder that extracts features of $x$ so that it can be decoded and reconstructed by $\hat{\mathbf{x}}$: $\hat{\mathbf{x}} = \mathbf{x}$. Therefore, the training of auto-encoder is implemented in an unsupervised fashion, which is the most outstanding feature of the auto-encoder. No labeled or paired input-output data is required but only the input data itself.

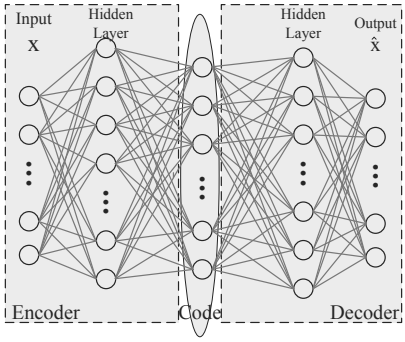



Fig. 3. The network structure of a DAE

Early auto-encoders with one hidden layer are limited in their learning and representation abilities. The development of deep learning[1] allows the neural network to grow deeper and larger with guaranteed convergence and accuracy. Therefore multiple hidden layers with hundreds of neurons are included to build a deep auto-encoder (DAE) that learns features of the input hierarchically. Increasing depths simplifies network training in terms of computational cost and required dataset size [19].

## III. Control Allocation Based On DAE

It is well-proved that a neural network with only one hidden layer can approximate any functions of arbitrary complexities provided that the network is given enough hidden units [20]. A network with two hidden layers is capable of providing an accurate model at a negligibly small error while using about half of the neurons used in a network with a single hidden layer [21].

Therefore, scholars introduced neural networks to solve the control allocation problem early in the 1990s [13]. However, neural networks were not so popular in the field in the early days. There are probably two reasons. One is that early neural networks were quite limited in their layers and neurons due to convergence issues. The neural network being trained contained only one or two hidden layers and a small number of neurons, and was always trapped in the local minimum. The modeling accuracy and generalization ability are far bellow expectation in reality. The other obstacle is that in the early days the only well-developed neural network was the MLP, which however, required supervised training. A large amount of paired data consisted of expected control moments and the allocated control effectors had to be calculated beforehand so as to train the MLP in a supervised way. This lead to a contradiction – We depend on an MLP to solve the allocation problem, while we have to solve the allocation problem itself beforehand in order to generate sufficient data for MLP training.

In recent years, deep learning proposed a few innovative techniques such as batch normalization and greedy layer-wise initialization such that a network with hundreds of hidden layers and thousands of neurons can still converge fast and accurately. New kinds of network structures specialize in different scenarios emerge every now and then. In this paper, the deep auto-encoder (DAE) is introduced to solve the control allocation problem. It will be shown that the information flow of a DAE is quite similar to the control allocation procedure, and by fitting into a specific DAE we proposed, the nonlinear control allocation problem is solved in an unsupervised training manner where no paired training data is needed.

Substituting the control effector function $\mathbf{g}$ in (2) into (1), the actual control moments acting on the vehicle body are determined by

$$\mathbf{u}_{act} = \overline{q}SL\mathbf{g}(\delta, \alpha, \beta, Ma). \tag{7}$$

Without loss of generality, let $\alpha$, $\beta$, and $Ma$ be constant, and denote

$$\mathbf{u}_{act} = \mathbf{g}'(\delta). \tag{8}$$

Further, denote the mapping from the expected moments to the allocated control effectors by $\pi : u_c \to \delta$, that is,

$$\delta = \pi(\mathbf{u}_c) \tag{9}$$

Then

$$\mathbf{u}_{act} = \mathbf{g}'(\pi(\mathbf{u}_c)) \tag{10}$$

By comparing Eq. (10) with Eq. (6), and by setting the encoder to model function $\pi$ and the decoder to model function $\mathbf{g}'$, connections between a DAE and the control allocation could be built, as being illustrated in Fig. 4. The encoder outputs the allocated control effectors $\delta$ according to the expected control moments $\mathbf{u}_c$ in the input layer. The decoder estimates $\hat{\mathbf{u}}_c$ produced by the allocated $\delta$.

With a well-trained DAE, the attitude control system is then structured according to Fig. 5, where the well-trained encoder from the DAE is embedded in the allocation modular. The controller[2] calculates $\mathbf{u}_c$, which is then feed into the encoder. The encoder allocates $\mathbf{u}_c$ to the vehicles' control effectors $\delta$, presuming that by deploying the effectors according to $\delta$, the vehicle will produce moments $\mathbf{u}_{act}$ such that $\mathbf{u}_{act} = \mathbf{u}_c$.

According to Fig. 5, when allocating the expected control moments, only the encoder is used without the assistance of the decoder. However, the decoder is a necessity for the training of encoder. The combination of encoder and decoder allows the encoder to be trained in an unsupervised way. Without the help of the decoder, a large amount of labeled training data consists of the expected control moments and the allocated control effectors is needed to train the encoder itself, which is overwhelming and contradict with the problem itself.

It may seem that the decoder can be replaced by the aerodynamic coefficient data set. This is true for simulation research. However, if provided sufficient real world flight data, a decoder could be trained exclusively on those flight data so as to build a high fidelity model of the control effectors. In such a case, no data table could be built, and

[1]For more comprehensive introduction to deep learning, please refer to, for example, the monograph [19].

[2]The design of controller is not the concern of this paper.

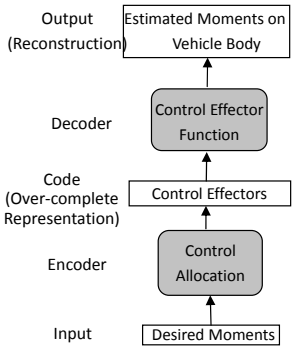



Fig. 4. The correspondence between a DAE and control allocation

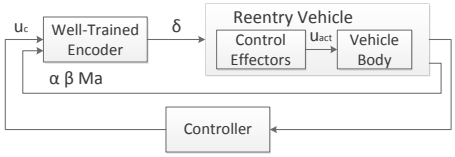



Fig. 5. The structure of the control system

mathematical expressions always fail to provide an explicit and accurate expression. On the other hand, compared to be built on a table, the control effector model built on a neural network could be adjusted online more efficiently.

Now we have setup the framework to solve the control allocation problem using a special kind of neural network, the DAE. The training of DAE is the major concern of the remaining work. Despite of the similarity, the training of DAE to solve the allocation problem is not straight forward. Traditional DAE is trained by updating the encoder and the decoder simultaneously. However, in the control allocation setting, when training the encoder to solve the allocation problem, the decoder should be constant because the control effector function $\mathbf{g}$ or $\mathbf{g}'$ modeled by the decoder is solely determined by the vehicle structure. Meanwhile, in deep learning, the DAE is related to image, text, or voice, while in the control allocation setting, information flow, the code in particular, has its physical attribute characterised by amplitudes and rate limits. Therefore, various efforts are expected to build and train the DAE for control allocation.

A. Network Structure

The DAE for control allocation is an over-complete one due to the redundancy of effectors. In other words, the dimension of the code is larger than the dimension of the input.

Effector amplitude constraints could be incorporated into the DAE by a specifically designed output function of the encoder

$$\phi(\delta) = max(\underline{\delta}, min(\overline{\delta}, \delta)), \tag{11}$$

which is a generalized form of the hard tanh activation function. $\underline{\delta}$ and $\overline{\delta}$ are the lower limit and the upper limit of the effectors, respectively.

B. Loss Function

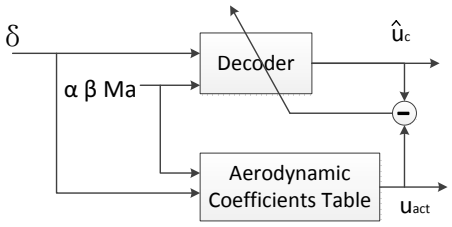



Fig. 6. The training of the decoder.

The loss function to be optimized is

$$J = \frac{1}{2N}\sum_{i=1}^{N} \|\mathbf{u}_c^i - \hat{\mathbf{u}}_c^i\|_2^2 + \frac{\lambda}{N}\sum_{i=1}^{N} \|P(\delta^i)\| \tag{12}$$

where $\mathbf{u}_c^i$ is the $i$-th training sample, $\hat{\mathbf{u}}_c^i$ is the reconstructed signal at the output. $\delta^i$ is the corresponding effector deflections, $N$ is the number of training data, and $\lambda$ is the weight that determines the importance of the regularizer $P(\delta^i)$.

In control allocation, the redundancy of effectors allows us to choose among different effectors so as to meet certain performance, for example the minimal energy consumption. This criterion can also be incorporated into the loss function by modifying the regularizer $P(\delta^i)$ of (12) into

$$J = \frac{1}{2N}\sum_{i=1}^{N} \|\mathbf{u}_c^i - \hat{\mathbf{u}}_c^i\|_2^2 + \frac{\lambda}{N}\sum_{i=1}^{N} \|\delta^i - \delta_{trim}^i\|_1 \triangleq J_1 + J_2 \tag{13}$$

where $\|\delta^i - \delta_{trim}^i\|_1 = \sum_{k=1}^{N} |\delta_k^i - \delta_{trim_k}^i|$ is the sum of position errors between the allocated effector $\delta_k^i$ and the preferred trim position $\delta_{trim_k}^i$. This objective indicates that when a vehicle is disturbed from its trim condition, the encoder should allocate the effectors close to the trims in addition to match the expected control moments.

C. Training and Tuning

According to Fig. 4, the decoder in the DAE is not any arbitrary ones but is expected to specifically approximate the control effector function. When training the encoder, the decoder should already be well-trained, and should be fixed. Therefore the DAE is trained in two subsequential steps. In the first step, the decoder is trained independently using paired data $(\delta, \mathbf{u}_{act})$. The training data set is generated by simple table-referring on the aerodynamic coefficients database that is built by either wind tunnel experiment or computational fluid dynamics (CFD). Sufficient and divergent paired data for decoder training could be generated upon request on this database. The decoder represents the control effector function $\mathbf{g}$. Although the decoder is trained in a supervised way, the paired data is from $\delta$ to $\mathbf{u}_{act}$, which involves no optimization. Fig. 6 illustrates the training of the decoder.

Once the decoder is well-trained, the encoder could be trained upon it, as shown in Fig. 7. The encoder is trained using back propagation in an unsupervised way, meaning no labeled data is needed, which is the most valuable feature to control allocation. The training data set consists only the input data, which is generated from the attainable control moment set.

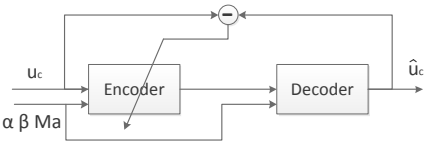



Fig. 7. The training of the DAE for control allocation.

When training the network, the parameters of the decoder are fixed, and the output activation of the encoder is specially designed as in Eq. (11). Therefore the ordinary gradient decent algorithm needs to be recalculated for this specific DAE.

The input layer of the DAE is labeled 1 and the first hidden layer of the encoder is labeled 2. The code layer is indexed by $\varepsilon$, and the output layer of DAE is indexed by $\tau$. Recall Eq. (5), the output $\hat{u}_c$ of the DAE is

$$\begin{aligned}\hat{\mathbf{u}}_c &= \mathbf{f}(\mathbf{w}^\tau \mathbf{a}^{\tau-1} + \mathbf{b}^\tau) \\ &= \mathbf{f}(\mathbf{w}^\tau \mathbf{f}(\mathbf{w}^{\tau-1}\mathbf{a}^{\tau-2} + \mathbf{b}^{\tau-1}) + \mathbf{b}^\tau \\ &= \cdots \\ &= \mathbf{f}(\mathbf{w}^\tau \mathbf{f}(\mathbf{w}^{\tau-1} \ldots \mathbf{f}(\mathbf{w}^{\varepsilon+1}\delta + \mathbf{b}^{\varepsilon+1})\ldots)) + \mathbf{b}^\tau, \quad (14)\end{aligned}$$

where $\mathbf{f}$ is a sigmoid activation function that is element-wise with

$$f(x) = \frac{1}{1+e^{-x}} \quad (15)$$

During the training of DAE, only weights $\mathbf{w}^i, i \le \varepsilon$ are updated while the weights in the decoder are fixed. Denote $\mathbf{z}^k = \mathbf{w}^k \mathbf{a}^{k-1} + \mathbf{b}^k$, the derivative of $\hat{\mathbf{u}}_c$ over $\delta$ is

$$\begin{aligned}\frac{\partial \hat{\mathbf{u}}_c}{\partial \delta} &= (\mathbf{w}^{\varepsilon+1})^T(((\mathbf{w}^{\varepsilon+2})^T \cdots (((\mathbf{w}^\tau)^T \mathbf{f}'(\mathbf{z}^\tau)) \odot \mathbf{f}'(\mathbf{z}^{\tau-1})) \\ &\quad \odot \cdots \odot \mathbf{f}'(\mathbf{z}^{\varepsilon+2})) \odot \mathbf{f}'(\mathbf{z}^{\varepsilon+1})) \\ &= (\mathbf{w}^{\varepsilon+1})^T \theta^{\varepsilon+1} \quad (16)\end{aligned}$$

where $\odot$ is the Hadamard product and $\theta^\varepsilon = ((\mathbf{w}^{\varepsilon+1})^T \theta^{\varepsilon+1}) \odot \mathbf{f}'(\mathbf{z}^\varepsilon)$ with $\theta^\tau = \mathbf{f}'(\mathbf{z}^\tau)$.

Further the partial difference of the loss function $J$ in Eq. (13) over weight $\mathbf{w}^\varepsilon$ is

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}^\varepsilon} &= \frac{\partial J}{\partial \delta}\frac{\partial \delta}{\partial \mathbf{z}^\varepsilon}\frac{\partial \mathbf{z}^\varepsilon}{\partial \mathbf{w}^\varepsilon} \\ &= (\frac{\partial J_1}{\partial \delta} + \frac{\partial J_2}{\partial \delta})\phi'(\mathbf{z}^k)\mathbf{a}^{\varepsilon-1} \\ &= (\frac{\partial \hat{\mathbf{u}}_c}{\partial \delta} + \frac{\partial J_2}{\partial \delta})\phi'(\mathbf{z}^k)\mathbf{a}^{\varepsilon-1} \quad (17)\end{aligned}$$

$\phi(\cdot)$ is the hard tanh activation function from (11) with derivative

$$\begin{aligned}\phi'(\mathbf{z}^k) &= \frac{\partial \max(\underline{\mathbf{z}}, \min(\overline{\mathbf{z}}, \mathbf{z}^k))}{\partial \mathbf{z}^k} \\ &= \begin{cases} 1, & \underline{\mathbf{z}} \le \mathbf{z}^k \le \overline{\mathbf{z}} \\ 0, & Others \end{cases} \quad (18)\end{aligned}$$

Substituting Eq. (16) into Eq. (17), the partial derivative of $J$ over $\mathbf{w}^\varepsilon$ is

$$\frac{\partial J}{\partial \mathbf{w}^\varepsilon} = [(\mathbf{w}^{\varepsilon+1})^T \theta^{\varepsilon+1} + \lambda sign(\delta)] \odot \phi'(\mathbf{z}^\varepsilon) \odot \mathbf{a}^{\varepsilon-1} \quad (19)$$

The gradient for the rest layers in the encoder is similar to the standard gradient in an DAE:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}^i} &= \mathbf{a}^{i-1}\frac{\partial J}{\partial \mathbf{z}^i} \\ &= \mathbf{a}^{i-1}((\mathbf{w}^{i+1})^T \frac{\partial J}{\partial \mathbf{z}^{i+1}} \odot \mathbf{f}'(\mathbf{z}^i)), \ i < \varepsilon \quad (20)\end{aligned}$$

## IV. Examples

The test reentry vehicle is a slender wing body with six aerodynamic control effectors. Namely, two flaperons $\delta_1$, $\delta_2$, two ruddervators $\delta_3$, $\delta_4$, one bodyflap $\delta_5$, and one speedbraker $\delta_6$. The flaperons are usually mounted close to the center of gravity, and thus the ruddervators and the bodyflap play the dominant roles for pitch control. Yaw moment is mainly provided by the ruddervators, whereas the flaprons may also result in a slight yaw moment. The roll control depends on the two flaperons when deflected differentially. However, the ruddervators can also result in an unexpected but small roll moment.

### A. Control Effector Modeling – The Decoder

Recall Eq. (1) and Eq. (2), the MLP for control effector modeling has nine input variables

$$\mathbf{x} = [\delta_1 \quad \delta_2 \quad \ldots \quad \delta_6 \quad \alpha \quad \beta \quad Ma]^T$$

The corresponding output is

$$\hat{\mathbf{u}}_c = [\hat{u}_{c_x} \quad \hat{u}_{c_y} \quad \hat{u}_{c_z}]^T$$

representing the estimated moments on roll, yaw, and pitch.

By exhaustively combining $\alpha = [0,2,4,6]$, $\beta = [-3,0,1,3]$,$Ma = [1,2]$, $\delta_5 = [0,5]$ and the rest effectors in the set $[-10,-5,2,10]$, a total of

$$4 \times 4 \times 2 \times 2 \times 4 \times 4 \times 4 \times 4 \times 4 = 65536$$

input samples are collected. A more refined quantization would improve the training performance but result in oversize dataset. According to our experiment, the relatively crude levels chosen above could be compensated by the generalization ability of the network. Another 44464 samples are generated by randomly select each variables in their feasible ranges. The expected output $\{\mathbf{u}^i_{act}\} = [u^i_{act_x} \ u^i_{act_y} \ u^i_{act_z}]^T$ of the input $\{\mathbf{x}^i\}, i \in [1, 110000]$ is obtained by referring to the aerodynamic coefficients database calculated by CFD. A total of 90000 samples are randomly selected from the 110000 data set as training data, and the rest 20000 samples are testing data.

These raw data should be normalized for better training performance and generalization ability. However, in this specific example, we do NOT normalize the input. Because this MLP is going to be the decoder in the later DAE, and this input corresponds to the code in the DAE, namely, the actual effector deflections we are seeking for. Normalization of the input could distort the results. Only the output data set is normalized by z-score normalization. Consider vector $\mathbf{u}_{act_x} = [u^i_{act_x}]$ of $N$ samples, its z-score normalization is

$$u^{i'}_{act_x} = \frac{u^i_{act_x} - \sum_1^N \frac{u^i_{act_x}}{N}}{\sigma(\mathbf{u}_{act_x})} \quad (21)$$

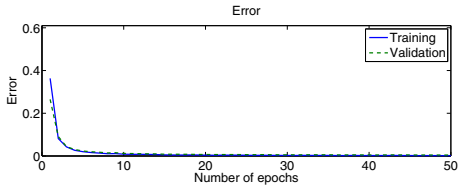



Fig. 8. Convergence of the training errors with $\theta = [9 \;\; 150 \;\; 50 \;\; 3]$

TABLE I

THE TESTING PERFORMANCE

| | Training | Testing | | | | | |
|---|---|---|---|---|---|---|---|
| | MSE | $MSE_x$ | $MSE_y$ | $MSE_z$ | $\Delta_x$ | $\Delta_y$ | $\Delta_z$ |
| $\Xi_1$ | 0.0019 | 0.0022 | 0.0028 | 0.0037 | 21% | 12% | 24% |
| $\Xi_2$ | 0.0027 | 0.0027 | 0.0036 | 0.0038 | 20% | 11% | 24% |
| $\Xi_3$ | 0.0233 | 0.0089 | 0.0117 | 0.0222 | 39% | 37% | 59% |
| $\Xi_4$ | 0.0130 | 0.0136 | 0.0120 | 0.0220 | 47% | 46% | 59% |

where $\sigma(\cdot)$ is the standard deviation.

In deep learning, the training data of size $s$ is processed in batches. Each batch contains $m$ samples that are randomly selected from the data set. At each epoch, $n = s/m$ batches are generated for training. Depending on the complexity of the problem and the convergence of training error, there are tens to hundreds of epochs. In this example, a total of 50 epochs are made with $m = 50$.

Four MLPs with different numbers of hidden layers and neurons are trained in parallel: $\Xi_i = \{\theta_i, f(x), o(x), J\}$ where $\theta_1 = [9, 150, 50, 3]$, $\theta_2 = [9, 50, 50, 3]$, $\theta_3 = [9, 50, 3]$, and $\theta_4 = [9, 200, 3]$. Function $f(x)$ is sigmoid, and $o(x) = x$. The loss function is the standard mean-square prediction error (MSE)

$$J = \lambda \|w\|^2 + \frac{2}{N} \sum_{t=1}^{N} \|\mathbf{u}_{act}^t - \hat{\mathbf{u}}_c^t\|_2^2 \tag{22}$$

with $w$ being the network weights.

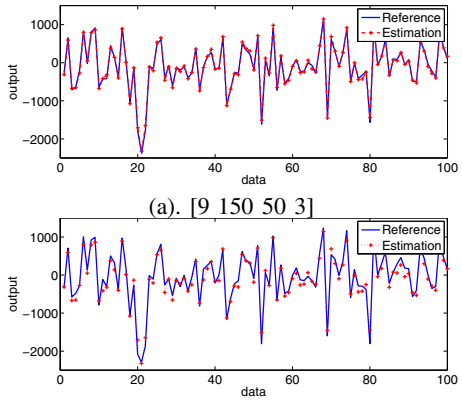
Fig. 8 shows the convergence of the network with $\Xi_1$. Those trained MLPs are than evaluated on the 11000 testing data set. Fig. 9 compares the estimated output with the reference output at 100 randomly selected testing samples on network $\Xi_1$ and $\Xi_4$.

The testing performance on the testing data set is measured by the MSE on $\hat{u}_{c_x}, \hat{u}_{c_y}$ and $\hat{u}_{c_z}$, individually, as been recorded in Table I. We also compare the four MLPs by their estimation untrusty rates. Given a testing sample $(\mathbf{x}^i, \mathbf{u}_{act}^i)$, the network output $\hat{u}_{c_x}^i$ is an trusty estimation if

$$\left| \frac{\hat{u}_{c_x}^i - u_{act_x}^i}{u_{act_x}^i} \right| < 0.2 \tag{23}$$

The percentages of the *untrusty* estimations on the three output variables, as denoted by $\Delta_x, \Delta_y, \Delta_z$, are counted in Table I as well. A smaller percentage indicates more estimations from the network are trusty.

According to Table I, the most important conclusion we can draw is that a deep network, as expected, requires less hidden neurons. $\Xi_1$ and $\Xi_2$ with two hidden layers achieve significantly higher accuracies in compared with $\Xi_3$ and $\Xi_4$ that have only one hidden layer. Even when the number of hidden neurons in $\Xi_4$ is twice of the one in $\Xi_2$ , the estimation error of $\Xi_4$ is still several times larger than that of $\Xi_2$ . This validates that a deeper MLP with less neurons outperforms a shallow one with a larger number of neurons. Moreover, according to Table I, there are smaller percentages of samples that exceeds the 20% error bound in $\Xi_1$ and $\Xi_2$, which outperform the MLPs with one hidden layer by a large margin.



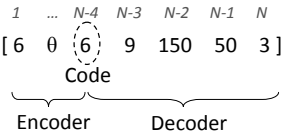

(a). [9 150 50 3]

(b). [9 200 3]

Fig. 9. The reference output and the estimated output by MLPs

## B. Control Allocation – The Encoder

The structure of the DAE is




where the $i$-th entry represents the number of neurons in layer $i$. The decoder is $\Xi_1$, the best one from Table I. The input to the encoder is

$$\mathbf{x} = [u_{cx} \; u_{cy} \; u_{cz} \; \alpha \; \beta \; Ma]$$

The output of the encoder are the six control effectors $\delta_i$. Recall the previously trained MLP, the input to the decoder consists not only the six allocated effectors, but also $\alpha$, $\beta$, and $Ma$. Therefore, the six outputs in layer $N-4$ are then copied to layer $N-3$, and $\alpha$, $\beta$, and $Ma$ from the input layer of the DAE are also copied to the rest three neurons. The network structure of the hidden layers in the encoder is denoted by $\theta$

Meanwhile, when effector saturations are considered, the output activation function of the encoder, i.e., the activation function for layer $N-4$, is altered to the saturation function $\phi(x)$ shown in Eq. (11) with $\underline{\delta} = [-10 \;\; -10 \;\; -10 \;\; -10 \;\; -10 \;\; 0]$ and $\overline{\delta} = [10 \;\; 10 \;\; 10 \;\; 10 \;\; 10 \;\; 5]$.

A total of 40 epochs are made with $m = 50$.

The decoder is initialized and fixed using the previousely well-trained $\Xi_1$.

The DAE is trained by updating only the weights connecting layer 1 to layer $N-4$. Due to the saturation function

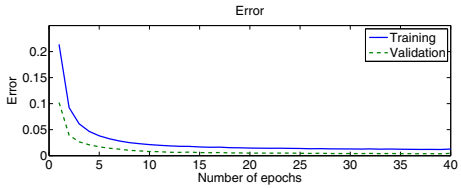
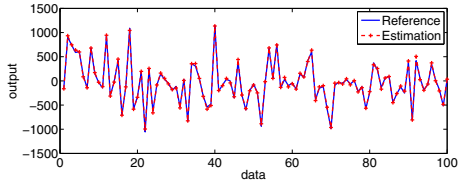
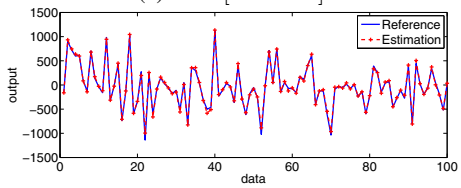



Fig. 10. Convergence of the training errors with $\theta = [50\ 30\ 10]$




(a). $\theta = [50\ 30\ 10]$




(b). $\theta = [100\ 50]$

Fig. 11. The reference output and the estimated output by DAEs

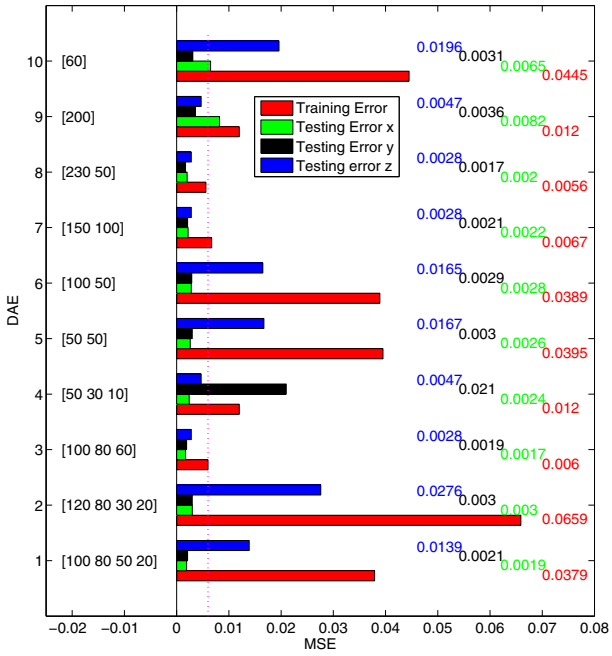



Fig. 12. The training and testing errors

in layer $N-4$ and the fixed decoder, weights are updated according to Eq. (19) and Eq. (20).

Ten encoders with different numbers of hidden layers and neurons are setup to show the performance of the DAE. The numbers of hidden layers range from 1 to 4. The training errors with $\theta = [50\ 30\ 10]$ are demonstrated in Fig. 10, and the estimated outputs between $\theta = [50\ 30\ 10]$ and $\theta = [100\ 50]$ on the testing data are compared in Fig. 11. The training and testing performance is also evaluated by MSE and the estimation untrusty rate $\Delta(\cdot)$ in Fig. 12 and Fig. 13, respectively.

According to Fig. 12 and Fig. 13, DAE 3 with three hidden layers shows the best performance in terms of MSE and estimation untrusty rate. The DAEs with only one hidden layer, no matter how many neurons are chosen, are left behind by a large margin. When being designed appropriately, an DAE with two hidden layers, say DAE 7 or DAE 8, could have comparative performance with a DAE (in this example DAE 3) with three hidden layers. However, DAE 8 catches DAE 3 in terms of testing errors while showing a great drop down at the estimation untrusty rate, and DAE 7 catches DAE 3 in terms of estimation untrusty rate while being a little bit worse at testing errors. In other words, the overall performance of DAE 3 with three hidden layers outperforms that of all the others with less hidden layers. Meanwhile, DAE 3 has the smallest number of neurons among these three DAEs, which proves the conclusion that when the network goes deeper, it requires less neurons to achieve an equal or usually better performance. However, an over complex network could degrade the training performance, as being indicated by DAE 1 and DAE 2.

The MLP and the DAE are trained off-line using the aerodynamic coefficients. When being implemented online, the allocated effectors are calculated almost instantly through feed-forward propagation over the encoder.

## V. Conclusions

This work is inspired by the challenge of constrained nonlinear control allocation for reentry vehicles that work at extreme conditions to perform fast maneuvers. An MLP with more than one hidden layers is built to model the nonlinear aerodynamics. A DAE with different training rules and activation functions from the conventional auto-encoder in deep learning is built and trained to solve the control allocation problem under effector amplitude constraints. With appropriate hidden layers and neurons in the encoder, the allocation error could be sufficiently small when the network is properly trained with techniques from deep learning. Different network structures correlate to different levels of effector nonlinearity. No optimization is evolved during the generalization of training data, and the encoder that solves the allocation problem is trained in an unsupervised manner.

The DAE for control allocation is trained off-line and fine-tuned online. Therefore the online tuning of the neural network uses a small amount of flight data and would be fast. Meanwhile, during the allocation, we only *use* the network instead of train the network, therefore the allocation is almost realtime.

At the current stage, only the position limits of effectors are considered. The dynamics of effectors such as rate limits has not been treated properly, which is the main focus of future work. Online adaption of the decoder should also be addressed in the future so as to further improve the fidelity of the proposed method.

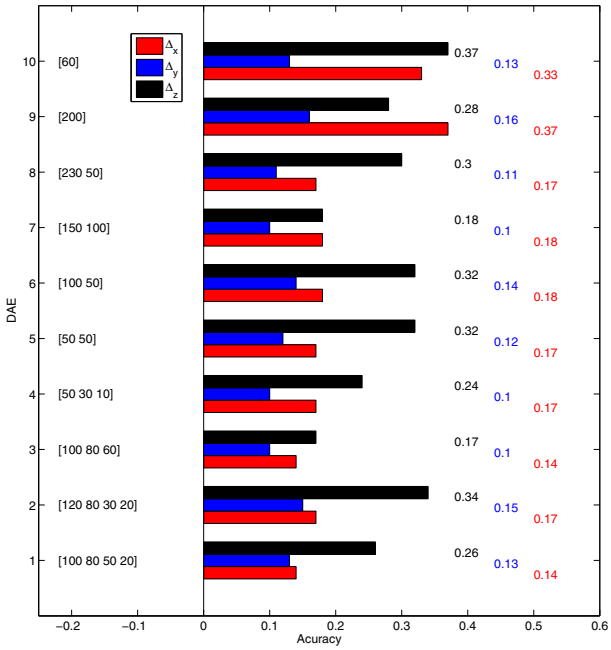



Fig. 13. The estimation untrusty rate

## REFERENCES


[1] Bodson, M., "Evaluation of Optimization Methods for Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 703–711.

[2] Wallner, E. M., and Well, K. H., "Attitude Control of a Reentry Vehicle with Internal Dynamics," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 6, 2003, pp. 846–854.

[3] Oppenheimer, M. W., Doman, D. B., and Bolender, M. A., "Control Allocation for Over-Actuated Systems," *Proceedings of the 14th Mediterranean Conference on Control and Automation*, IEEE, Ancona, Italy, June 2006, pp.1–6.

[4] Doman, D. B., Gamble, B. J., and Ngo, A. D., "Quantized Control Allocation of Reaction Control Jets and Aerodynamic Control Surfaces," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 13–24.

[5] Burken, J. J., Lu, P., Wu, Z., and Bahm, C., "Two Reconfigurable Flight-Control Design Methods: Robust Servomechanism and Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 3, 2001, pp. 482–493.

[6] Cui, Lei, and Yang, Ying, "Disturbance Rejection and Robust Least-Squares Control Allocation in Flight Control System," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 6, 2011, pp. 1632–1643.

[7] Huijts, C., and Voskuijl, M., "The Impact of Control Allocation on Trim Drag of Blended Wing Body Aircraft," *Aerospace Science and Technology*, Vol. 46, No. 2015, 2015, pp. 72–81.

[8] Luo, Y., Serrani, A., Yurkovich, S., Doman, D. B., and Oppenhiemer, M. W., "Model Predictive Dynamic Control Allocation with Actuator Dynamics," *Proceedings of the 2004 American Control Conference*, IEEE, Boston, Massachusetts, June 2004, pp. 1695–1700.

[9] Johansen, T. A., and Fossen, T. I., "Control Allocation - A Survey," *Automatica*, Vol. 49, No. 2013, 2013, pp. 1087–1103.

[10] Luo, Y., Serrani, A., Yurkovich, S., Oppenhiemer, M. W., and Doman, D. B., "Model-Predictive Dynamic Control Allocation Scheme for Reentry Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 1, 2007, pp. 100–113.

[11] Bolender, M. A., and Doman, D. B., "Non-Linear Control Allocation Using Piecewise Linear Functions: A Linear Programming Approach," *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA, Providence, RI, Aug. 2004.

[12] Poonamallee, V. L., Yurkovich, S., Serrani, A., Doman, D. B., and Oppenheimer, M. W., "A Nonlinear Programming Approach for Control Allocation," *Proceeding of the 2004 American Control Conference*, Boston, Massachusetts, June 2004, pp. 1689–1694.

[13] Grogan, R. L., Durham, W. C., and Krishnan, R., "On the Application of Neural Network Computing to the Constrained Flight Control Allocation Problem," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA, Washington, DC, Aug. 1994, pp. 911-921.

[14] Jung, D. W., Lowenberg, M. H., and Jones, C. D. C., "Integration of Control Allocation Methods in Bifurcation Analysis Framework," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA, Keystone, Colorado, Aug. 2006.

[15] Page, A. B., and Steinberg, M. L., "Effects of Control Allocation Algorithms on a Nonlinear Adaptive Design," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit,* AIAA, Reston, VA, Aug. 1999.

[16] Buffington, J. M., and Enns, D. F., "Lyapunov Stability Analysis of Daisy Chain Control Allocation," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 6, 1996, pp. 1226–1230.

[17] Hu, Q., Li, B., and Zhang, Y., "Nonlinear Proportional-Derivative Control Incorporating Closed-Loop Control Allocation for Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 799–812.

[18] Huang, H., "Control Allocation of Reaction Control Jets: Quantization And Stability," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 1, 2015, pp. 136–143.

[19] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*. MIT Press, Cambridge MA, 2016, pp. 1–25, 289–290.

[20] Sartori, M. A., and Antsaklis, P. J., "A Simple Method to Derive Bounds on the Size and to Train Multilayer Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 4, 1991, pp. 467–471.

[21] Tamura, S., and Tateishi, M., "Capabilities of a Four-Layered Feedforward Neural Network: Four Layers Versus Three," *IEEE Transactions on Neural Networks*, Vol. 8, No. 2, 1997, pp. 251–255.