Aerospace Science and Technology 86 (2019) 826–835

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

# Aerodynamic shape optimization using a novel optimizer based on machine learning techniques

Xinghui Yan [a,b,*], Jihong Zhu [a,b], Minchi Kuang [a,b], Xiangyang Wang [a,b]

[a] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People's Republic of China
[b] State Key Laboratory of Intelligent Technology and System, Tsinghua University, Beijing 100084, People's Republic of China

ARTICLE INFO

*Article history:*
Received 4 October 2018
Received in revised form 2 January 2019
Accepted 1 February 2019
Available online 7 February 2019

*Keywords:*
Aerodynamic optimization
Reinforcement learning
Transfer learning
Computational fluid dynamics

ABSTRACT

Aerodynamic shape optimization is usually a loop of an optimization model, an optimizer and an evaluation workflow. A new optimizer is proposed and tested for a typical aerodynamic shape optimization of missile control surfaces with computational fluid dynamics (CFD). The new optimizer emphasizes the use of machine learning techniques, reinforcement learning and transfer learning, to improve performance and efficiency. Reinforcement learning is applied to extract the optimization experience from the semi-empirical method DATCOM using deep neural networks. Transfer learning is implemented to reuse the experience as priori knowledge in the CFD-based optimization by sharing neural network parameters. For the considered aerodynamic shape optimization problem of missile control surfaces, a remarkable reduction in the computational time has been accomplished. The new approach significantly decreases the required CFD calls by over 62.5%. Meanwhile, the time spent in the experience extraction and parameter transfer process is negligible.



## 1. Introduction

With the development of computer science and computational fluid dynamics (CFD), aerodynamic design methods have been developed from cut-and-try to reverse design [1] and through to optimal design [2]. The related analysis methods have also evolved from wind tunnel tests to high fidelity CFD simulations [3]. Meanwhile, aircraft performance requirements have become much stricter in terms of complex design conditions, design constraints and optimization model. Consequently, aerodynamic design methods face formidable challenges, especially in the global optimization of high-dimensional design space [4]. The current mainstream design flow is described in Fig. 1. It is a combination of aerodynamic calculation and optimization theories, converting the optimization problem into an extremum solving problem with numerical simulations [5]. Firstly, an optimization model should be built including design variables, design space, design constraints and objective functions. Then, an evaluation workflow of geometry model parameterization [6], mesh generation/deformation [7] and numerical calculation is established. Finally, the optimization model and the evaluation workflow are connected by an optimizer which could be either gradient-based or evolutionary algorithms.

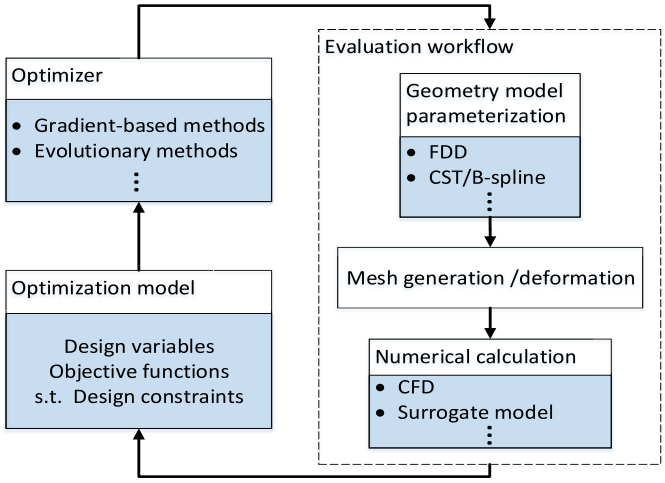



**Fig. 1.** Flowchart of aerodynamic shape optimization.

The existing optimization methods have limitations. The most widely used gradient-based methods, the finite difference method [8] and the adjoint method [9], address the problems of unacceptable computation amount and convergences on local optima respectively. The evolutionary methods, such as multi-objective genetic algorithm (MOGA) [10] or multi-objective particle swarm op-

* Corresponding author at: Department of Computer Science and Technology, Tsinghua University, Beijing 100084, People's Republic of China.
*E-mail address:* yanxh16@mails.tsinghua.edu.cn (X. Yan).

https://doi.org/10.1016/j.ast.2019.02.003
1270-9638/

timization (MOPSO) [11], are capable of handling multi-extreme and multi-objective optimization problems. However, their convergence speeds are generally slow enough that they're incapable of direct cooperation with time-consuming CFD. Here, some other alternative method with better computational effectiveness are available for CFD, such as the semi-empirical methods and the surrogate model based methods. The semi-empirical methods combine data sheets, linear aerodynamic theory and empirical equations to obtain aerodynamic characteristics, among which DATCOM [12,13] is a widely used one which covers methodologies for predicting aerodynamic performance and stability of a variety of aircraft configurations. However, semi-empirical methods might be limited when confronted with novel configurations [14]. Researches have also been concentrated on combining surrogate models [15,16] with evolutionary methods. But the curse of dimensionality [17] holds back the application of surrogate-based optimization with large design variable sets. The choice of which aerodynamic calculation method to use is determined by the balance between cost and fidelity.

The recent developments in the machine learning domain, especially in reinforcement learning (RL) [18] and transfer learning (TL) [19], offers a new way for the optimizer in aerodynamic optimization [20]. RL could be regarded as a new application of deep neural networks (DNN), which tunes the network parameters to perform better via interactions and feedbacks [21]. By combining RL with the previously mentioned evaluation workflow, the DNN will be able to learn from the experience, which accounts for aerodynamic improvement for any given design variables within the design space. Herein, the experience equals the increases or decreases of some design variables at a given condition. TL is a widely used machine learning technique that transfers knowledge from one trained DNN to other untrained DNNs through sharing network parameters. Thus, the training time could be greatly reduced.

In this paper, we propose a new optimizer based on the two machine learning techniques, namely reinforcement learning and transfer learning. Deep deterministic policy gradient (DDPG) [22], a state-of-the-art RL method, is introduced, which adopts a dual DNN structure called actor-critic to improve the problem of convergence on local optimum. We first combine DDPG with the evaluation workflow based on the fast aerodynamic prediction software DATCOM. Thus, much implicit optimization experience could be learned by DDPG networks through a large number of interactions with DATCOM. Subsequently, transfer learning is applied to reuse the optimization experience as priori knowledge with the CFD-based evaluation workflow to accelerate the convergence speed. Finally, a contrast experiment is conducted between the proposed optimizer and traditional optimizers based on evolutionary methods including MOGA and MOPSO. The results demonstrate that the proposed optimizer significantly outperforms the others in both optimized results and convergence speed.

This paper is organized as follows: Section 2 describes the optimization problem, including objective functions, design variables and constraints. Section 3 presents the optimization approach based on machine learning techniques. Section 4 gives the results and discussions of the numerical experiments. Finally, the paper is concluded in Section 5.

## 2. Statement of the problem

Taking the shape optimization of the aerodynamic fin of a missile as an example, the optimization goal is to develop the geometry with the highest possible lift–drag ratio at cruise conditions, which at the same time, satisfies the geometrical and aerodynamic constraints. And this goal needs to be accomplished through CFD simulations.

Considering the choice of the objective function, it is assumed that the lift–drag ratio (CL/CD) is a sensitive and reliable indicator of aerodynamic performance and thus, the maximization of the lift–drag ratio is employed as the objective function of the optimization problem. Additionally, the stability issue and drag coefficient (CD) should also be taken into consideration. The static margin is the distance between the center of pressure and the center of gravity, and the latter is assumed to be the midpoint of the missile body. Due to the mass of missile fins is relatively small compared with the main body, the variation of mass center of the missile during the optimization procedure is neglected for simplification. Thus, the static stability can be maintained with the constraint of pressure center position. The missile's drag coefficient should also be constrained to avoid flight performance degradation. The optimization model is described as follow:

Flow conditions: flight Mach number of 2.0
flight altitude of 5000 m
angle of attack of 4 degrees
Objective functions: maximize lift–drag ratio coefficient
Constraints: design parameters should obey the geometrical constraints
XCP should be maintained within [0.53,0.6]
CD should be no more than the baseline value

where XCP is the position of pressure center and measured from the nose tip and divided by the missile length.

The design variables are defined in Table 1 and shown in Fig. 2. All the length variables are in meters and the angle variables are in degrees. The baseline model consists of a cone nose, a cylindrical body and trapezoidal wings and tails. The wings and tails are arranged in "+" form and both have hexagonal airfoils. Due to the considered problem focuses on the missile fin geometry and placements, the optimization of missile body and airfoils are out of consideration in this paper. Hence, the parameters for determining the missile body and hexagonal airfoils are set to fixed values shown in Table 1, including LNOSE, LCENTER, DCENTER, ZUPPER, ZLOWER, LMAXU, LFLATU and LFLATL. The trailing edges of the fins are set perpendicular to the body axis. Thus, the geometries of the wings or tails are right trapezoids and can be determined by 6 variables: the lengths of span, tip chord and root chord of the wings and tails. The placements of the wings and tails can be determined by XLE1 and XLE2 respectively. Hence, there are 8 independent design variables for determining the wings and tails, and each is geometrically constrained with upper and lower bounds, which constitute the design space.

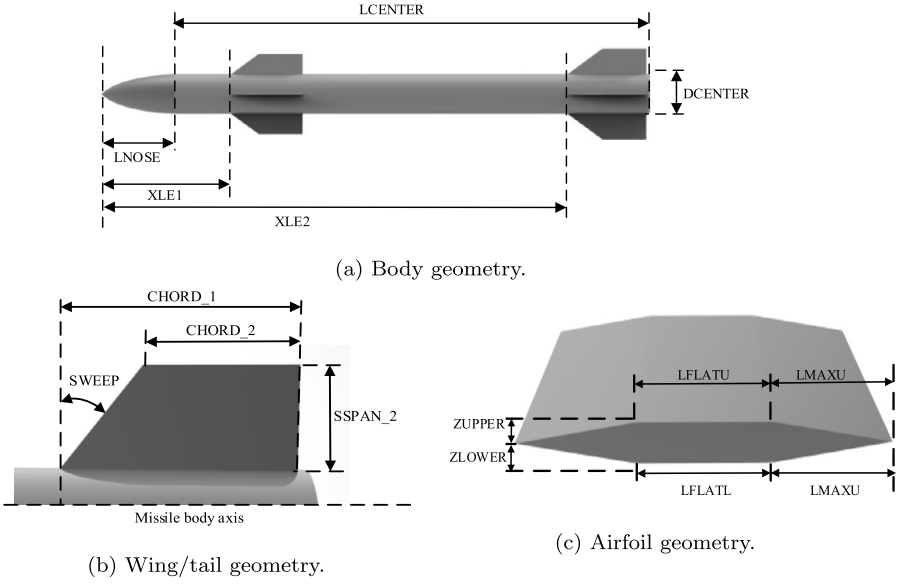
## 3. Optimization approach

### 3.1. Architecture

The two related tasks are defined as follows: solve the optimization problem with DATCOM-based and CFD-based evaluation workflow, respectively. The former task aims to extract optimization experience from DATCOM. The latter task aims to improve the design configurations through accurate CFD simulations. Deep reinforcement learning is introduced here to extract the optimization experience from DATCOM. Unlike the traditional applications of neural networks which train the networks with a prepared data base, deep reinforcement learning enables the deep neural networks to interact with DATCOM by themselves and learn self-directly: the DNNs choose a design configuration by the current learned optimization experience and feed the configuration to DATCOM or CFD software. Then the software outputs the calculation results back to the DNNs. Hence, the DNNs could learn the optimization experience from these input-output feedbacks by updating the network parameters. Moreover, due to the time-consuming

**Table 1**
Definition and value bounds of design variables.

| Design variable | Definition | Lower bound | Upper bound | Baseline |
|---|---|---|---|---|
| LNOSE | Nose length | – | – | 0.49 |
| LCENTER | Missile body length | – | – | 3.2 |
| DCENTER | Missile body diameter | – | – | 0.18 |
| XLE1 | Distance from nose tip to wing leading edge | 1.25 | 1.75 | 1.72 |
| SWEEP1 | Wing leading-edge sweep angle | – | – | – |
| CHORD1_1 | Wing root chord | 0.1 | 0.4 | 0.29 |
| CHORD1_2 | Wing tip chord | 0 | 0.09 | 0.06 |
| SSPAN1_2 | Exposed semispan of wing | 0.1 | 0.3 | 0.23 |
| ZUPPER1 | Wing thickness to chord ratio of upper surface | – | – | 0.025 |
| ZLOWER1 | Wing thickness to chord ratio of lower surface | – | – | 0.025 |
| LMAXU1 | Wing fraction of chord from section leading edge to maximum thickness of lower surface | – | 0.25 | |
| LFLATU1 | Wing constant thickness section length to chord ratio of upper surface | – | 0.25 | |
| LFLATL1 | Wing constant thickness section length to chord ratio of lower surface | – | 0.25 | |
| XLE2 | Distance from nose tip to tail leading edge | 3 | 3.2 | 3.2 |
| SWEEP2 | Wing trailing-edge sweep angle | – | – | – |
| CHORD2_1 | Tail root chord | 0.1 | 0.4 | 0.38 |
| CHORD2_2 | Tail tip chord | 0 | 0.25 | 0.19 |
| SSPAN2_2 | Exposed semispan of tail | 0.1 | 0.3 | 0.22 |
| ZUPPER2 | Tail thickness to chord ratio of upper surface | – | – | 0.025 |
| ZLOWER2 | Tail thickness to chord ratio of lower surface | – | – | 0.025 |
| LMAXU2 | Tail fraction of chord from section leading edge to maximum thickness of lower surface | – | 0.25 | |
| LFLATU2 | Tail constant thickness section length to chord ratio of upper surface | – | 0.25 | |
| LFLATL2 | Tail constant thickness section length to chord ratio of lower surface | – | 0.25 | |




(a) Body geometry.

(b) Wing/tail geometry.

(c) Airfoil geometry.

**Fig. 2.** Geometric sketch of the canard-controlled missile.

feature of CFD simulations, transfer learning is applied in the latter task to accelerate the convergence speed by reusing the extracted experience as priori knowledge. The architecture overview is described in Fig. 3.

### 3.2. Experience extraction via reinforcement learning

The main idea of this section is to use neural networks to extract optimization experience from semi-empirical method, which will be reused as priori knowledge in the latter CFD-based optimization. We choose DATCOM as the source of training data since it has been proved to be a reasonable tool [10,13]. It is used to predict the missile's aerodynamic coefficients and center of pressure position under the given design configuration and flight condition. Compared with the time-consuming CFD simulations, the time spent in DATCOM calculation can be neglected which costs less than half a second each time.

The aerodynamic optimization process is a Markov Decision Process (MDP) which meets RL's essential requirement. Taking the design variables [XLE1, XLE2, CHORD1_1, CHORD1_2, CHORD2_1, CHORD2_2, SSPAN1_2, SSPAN2_2] as the DNN input and the corresponding variations [$\Delta$XLE1, $\Delta$XLE2, $\Delta$CHORD1_1, $\Delta$CHORD1_2, $\Delta$CHORD2_1, $\Delta$CHORD2_2, $\Delta$SSPAN1_2, $\Delta$SSPAN2_2] as the output, the experience of choosing variations to make aerodynamic improvements can be extracted for any design configuration within

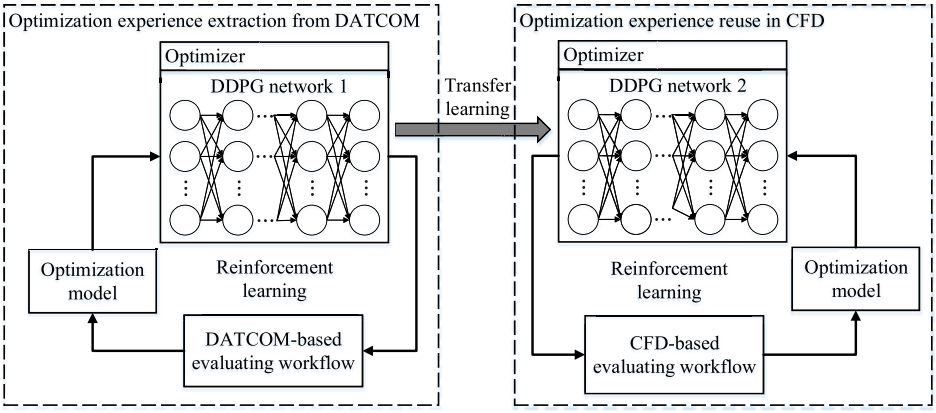



**Fig. 3.** Architecture of the machine learning approach for aerodynamic optimization.

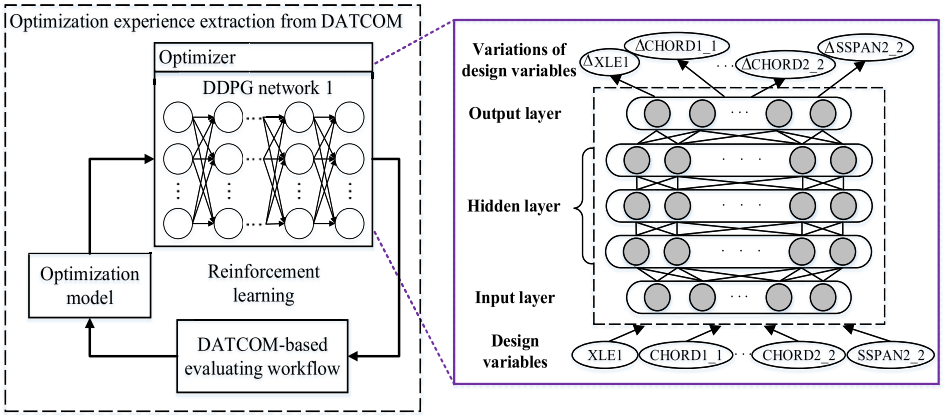



**Fig. 4.** Experience extraction from DATCOM via reinforcement learning.

the design space by training DNNs. This proposed approach is a novel way to utilize the empirical equations and experimental data involved in this semi-empirical software. Fig. 4 shows the experience extraction process.

In terms of the choice of reinforcement learning algorithm, deep deterministic policy gradient (DDPG) algorithm is employed after comparisons with other widely used reinforcement learning algorithms. Q-learning usually uses a data table [23] to store value functions, which is the expected improvement or degradation of the optimization objectives caused by the chosen variations. This algorithm is only applicable for the scenarios with very limited choices or possibilities, such as Atari game or fault detections [21]. Deep Q-learning network (DQN) [24] is a great improvement over Q-learning as it uses a DNN to store value functions and a buffer to replay the experience. However, DNQ still faces the problem of low resolution because it only targets problems with a discrete design space. Therefore, we introduce a state-of-the-art RL algorithm, deep deterministic policy gradient [22], to handle the continuous design space with high resolution.

### 3.3. Training methodology

The goal of DDPG network training is to extract the optimization experience $\pi : S \rightarrow A$, which maps design configurations to a probability distribution over the design variable variations as shown later in Algorithm 1. Herein, $S$ denotes the design configuration set. $A$ denotes the design variable variation set. With the network parameters $\theta$, the design variable variation can be determined as:

$$a = \pi(s|\theta) \tag{1}$$

where $s$ and $a$ are the current configuration and the corresponding design variables' variations respectively. Hence, the training goal is to obtain the optimal network parameters under a given reward function [25]. We use the same reward function $r_t$ for training with the DATCOM-based and CFD-based evaluators. The function rewards the lift–drag ratio improvement and penalizes the excessive XCP variation and CD increment immediately after each interaction between the network and the evaluator:

$$r_t = \begin{cases} f(s_t) - f(s_0), \ \text{XCP} \in [0.53, 0.6] \text{ and CD} < \text{baseline} \\ f(s_t) - f(s_0) - 10 \cdot \Delta\text{XCP} - 10 \cdot \Delta\text{CD}, \ \text{otherwise} \end{cases} \tag{2}$$

where $f$ denotes the lift–drag ratio calculation with DATCOM or Fluent, and $s_0$ and $s$ are the baseline and current design configurations, respectively. $\Delta$XCP denotes the XCP deviation from the baseline. Then, the long-term reward $G_t$ can be defined as follows, which is a means of avoiding the local optimum:

$$G_t = \sum_{n=0}^{N} \gamma^n r_{t+n} \tag{3}$$

where the reward discount factor $\gamma \in [0, 1]$ determines the weight between future rewards and immediate rewards. The value function of the design configuration is denoted as $V_\theta$ and defined as follows, which equals the expectation of the long-term reward in terms of the current design configuration and network parameters:

$$V_\theta(s) = E_\theta[G_t | S_t = s] \tag{4}$$

Then the value function of the design variable variation is denoted as $Q_\theta$ and defined as follows, which is an evaluation of the

current design variable variations under the current design configuration and network parameters:

$$Q_\theta(s) = E_\theta[G_t \,|S_t = s\,,\, A_t = a] \tag{5}$$

The sub-network "actor" functions as the optimization experience with network parameters $\theta^\mu$. The sub-network "critic" functions as the value function $Q$ with network parameters $\theta^Q$. They play the roles of choosing design variable variations and evaluating the choice, respectively, as their name imply. The network parameters are iteratively updated during maximizing the expectation of the long-term reward. For "actor", the parameter update is conducted according to the gradient as:

$$\begin{aligned}\frac{\partial G_N(\theta^\mu)}{\partial \theta^\mu} &= E\left[\frac{\partial Q\,(s, a\,|\theta^Q)}{\partial \theta^Q}\right] \\ &= E\left[\frac{\partial Q\,(s, a\,|\theta^Q)}{\partial a}\frac{\partial \pi\,(s\,|\theta^\mu)}{\partial \theta^\mu}\right] \\ &= E\left[\nabla_a Q\,(s, a\,|\theta^Q) \cdot \nabla_{\theta^\mu}\pi\,(s\,|\theta^\mu)\right]\end{aligned} \tag{6}$$

For "critic", the network parameters are updated by the loss function, as same as regular DQN, which is based on a batch of $N$ sampled transitions $(s_i, a_i, r_i, s_{i+1})$:

$$\begin{aligned}Loss = \frac{1}{N}\sum\nolimits_i [r_i + \gamma\, Q'(s_{i+1}, \pi'(s_{i+1}\,|\theta^{\mu'})\,|\theta^{Q'}) \\ - Q\,(s_i, a_i\,|\theta^Q)]^2\end{aligned} \tag{7}$$

Moreover, "actor" and "critic" both has its own corresponding target network, which are the replicas of "actor" and "critic" except for the update mechanism. To improve the stability and convergence of the training process, there is a lag between the updates of target networks and "actor" or "critic". The sampling strategy from all existing transitions is also important for escaping from localization. Besides, the exploration of $\pi$ is realized by adding noise to "actor", which has a similar effect:

$$\pi'(s) = \pi\,(s|\theta^\mu) + \omega, \tag{8}$$

where $\omega$ is a certain kind of noise distribution such as a Gaussian or uniform. By this means, all the design configurations in the value range could be explored via probabilities. Hence, a high resolution searching process in the continuous design space is achieved which leads to high-performance optimized results.

### 3.4. Accelerated optimization via transfer learning

We choose Fluent as the CFD software, which has been widely used in aerodynamic optimization problems [26]. In order to realize the automatic optimization with closed loop, RBF Morph software [7] is employed to avoid manual re-modeling and re-meshing. RBF Morph is a numerical tool for mesh deformation and is fully integrated with Fluent in the solving process. Fig. 5 shows the predefined missile geometry, which has a one-to-one correspondence with the design variables. The mesh deformation is basically realized by shifting, rotating and scaling the predefined missile geometry along with the mesh. Fig. 6 shows the comparison between the missile mesh models before and after deformation, in which all the fin tip chord lengths are shortened by 50%.

There are many similarities between the optimizations based on DATCOM and CFD. In most cases, the trends indicated by the calculation results of the two softwares are consistent, since both are built on tested theories and data [27]. However, in some cases

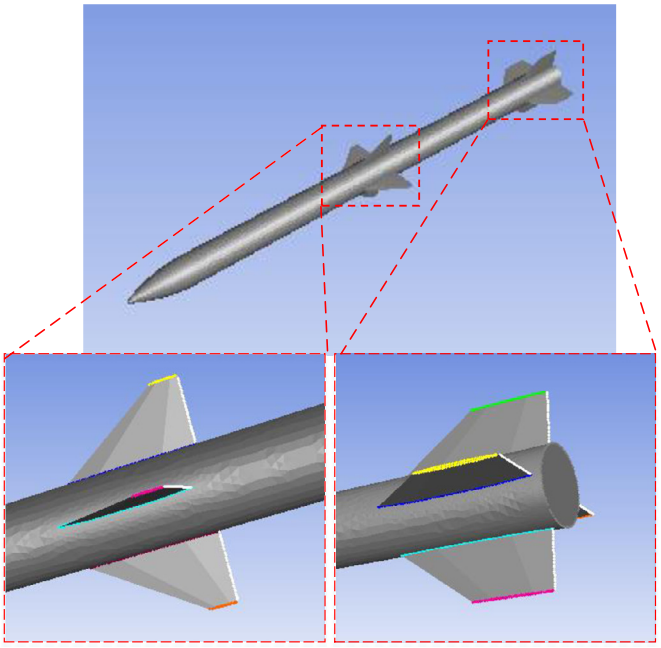
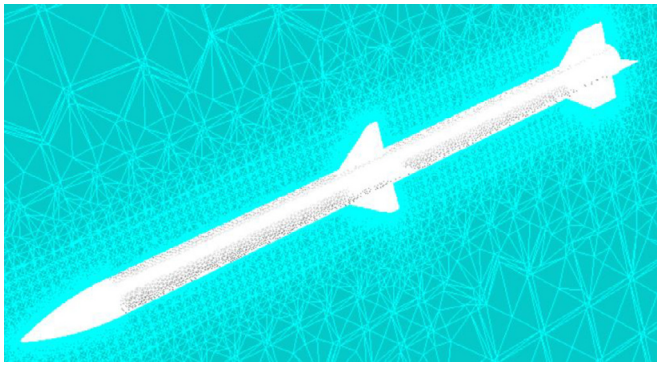
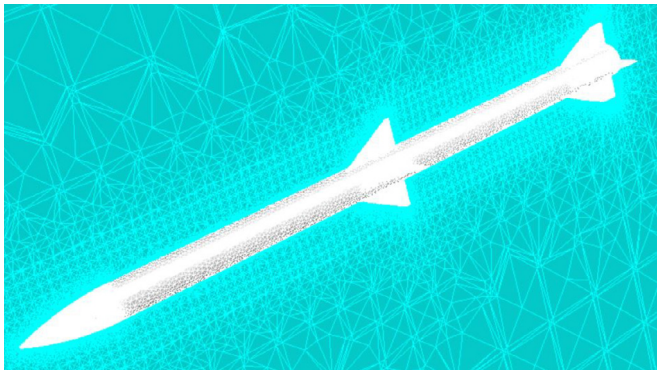


**Fig. 5.** Predefined missile geometry for mesh deformation.



(a) Baseline missile mesh model.



(b) Deformed missile mesh model.

**Fig. 6.** Mesh deformation example.

the calculation results of the two softwares may not be consistent in values or even in trends. Transfer learning is applicable to such related but distinct optimization problems [19].

We first train the DDPG networks in the DATCOM-based optimization problem and then use the trained networks as initialization of the DDPG networks in the CFD-based optimization problem. On the one hand, we share the parameters of the input and the hidden layers of the DDPG networks trained in DATCOM as initial-

izations of those in CFD, which could greatly shorten the training time. On the other hand, the parameters of the output layer are not shared because of the differences between the calculation methods of DATCOM and CFD. Additionally, all the layers are allowed to be tuned after parameter transfer as an adaptation of evaluating workflow switching. By this means, the CFD calls required in network training can be reduced while the excellent global search ability of DDPG can still be fully utilized. The overview of transfer learning is shown in Fig. 7.

The proposed method of combination of DDPG and transfer learning (TL-DDPG) is described as follows:

**Algorithm 1** Aerodynamic shape optimization using reinforcement learning and transfer learning.

**Use DDPG to extract the optimization experience from DATCOM**
Randomly initialize critic network $Q_1(s,a|\theta^{Q_1})$ and actor network $\pi_1(s|\theta^{\mu_1})$ along with corresponding target network $Q_1'$ and $\pi_1'$
**for** *episode* $= 1$ *to* $M$ **do**
  Initialize design configuration $s_1$ and variation exploration $N$
  **for** *stept* $= 1$ *to* $T$ **do**
    Select and execute action to maximize value function: $\pi(s) = \arg\max_a Q(s,a)$
    Observe reward $r_t$, new configuration $s_{t+1}$ and store transition $(s_t, a_t, r_t, s_{t+1})$ in replay buffer
    Sample a minibatch of $n$ transitions from replay buffer
    Update critic network by minimizing the loss function according to equation (7)
    Update actor network using the sampled policy gradient according to equation (6)
    Update the target networks: $\theta^{Q'_1} = \tau\theta^{Q_1} + (1-\tau)\theta^{Q'_1}$, $\theta^{\mu'_1} = \tau\theta^{\mu_1} + (1-\tau)\theta^{\mu'_1}$
  **end for**
**end for**
**Use shared layers to obtain the TL-DDPG networks for CFD-based optimization**
Initialize critic network $Q_2(s,a|\theta^{Q_2})$ and actor network $\pi_2(s|\theta^{\mu_2})$ along with respective target network $Q_2'$ and $\pi_2'$ with trained layers from $Q_1(s,a|\theta^{Q_1})$, $\pi_1(s|\theta^{\mu_1})$, $Q_1'$ and $\pi_1'$
**Use TL-DDPG to search high performance design configurations with CFD simulations**
Repeat the DDPG training procedures

## 4. Numerical experiments and results

### 4.1. Network training setup

In the numerical experiments, we used TensorFlow [28] platform and Adam [29] algorithm to train the DDPG networks. The learning rates of the actor and the critic sub-networks were 0.001 and 0.005, respectively. The reward discount $\gamma$ was 0.99. Every sub-network has three fully connected hidden layers with 200 units each. $\sigma(X_i) = \max(0, X_i)$ and $\psi(X_i) = \sinh(X_i)/\cosh(X_i)$ were the activation functions used in the hidden layers and the output layer respectively. The replay buffer size and the mini-batch were 1000 and 32 respectively. For the DATCOM-based optimization, the max episodes and max steps for each episode were 800 and 20 respectively. For the CFD-based optimization, the max episodes and max steps were 50 and 20 respectively. The variations of every design variable are constrained within $[-0.03, 0.03]$ at each step.

**Table 2**
Comparison of the calculated aerodynamic coefficients with different mesh elements.

| Number of mesh elements | Lift coefficient | Drag coefficient |
|---|---|---|
| 1517777 | 1.198 | 0.67 |
| 2462011 | 1.299 | 0.669 |
| 3038232 | 1.306 | 0.663 |
| 3985293 | 1.308 | 0.662 |
| 4805498 | 1.309 | 0.662 |

### 4.2. Simulation setup

An unstructured mesh of approximately three million elements created by the grid generation software ANSYS ICEM was used, as shown in Fig. 8. As the tetrahedral mesh existed in the majority area, the prism mesh was adopted near the boundary of missile fairing, body and control surfaces. Fig. 8 also highlights the presence of local refinement regions, which allowed an improvement of the volume mesh around the missile model, in particular near the nose and control surface region. Due to meshing constraints and computer capability, few areas of mesh model's $y^+$ value was greater than 1. On the one hand, the aerodynamic coefficients results would not be sensitive to the increase of mesh element number when this number had been up to a certain level. On the other hand, it would take much more time on convergence of the simulation when mesh element number became considerably large. Consequently, an investigation on the influence of the mesh element number on the calculated aerodynamic coefficients was conducted as shown in Table 2. It can be observed that when the mesh element number is up to three million, the aerodynamic coefficients almost remain unchanged with an increasing element number. Therefore, the mid-resolution mesh with about three million elements was adopted for the optimization.

A density based implicit solver was used for the steady state CFD simulation using ANSYS Fluent software. Roe flux-difference splitting (Roe-FDS) was selected as the convective flux type.

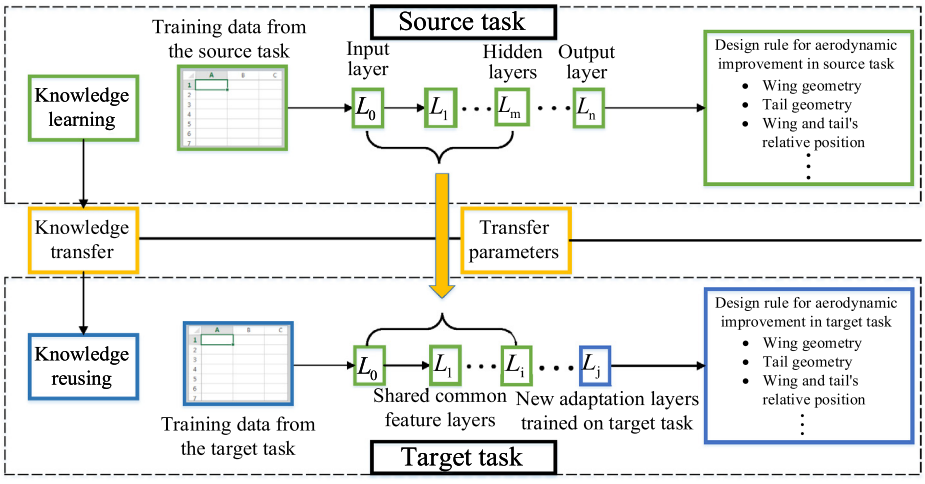
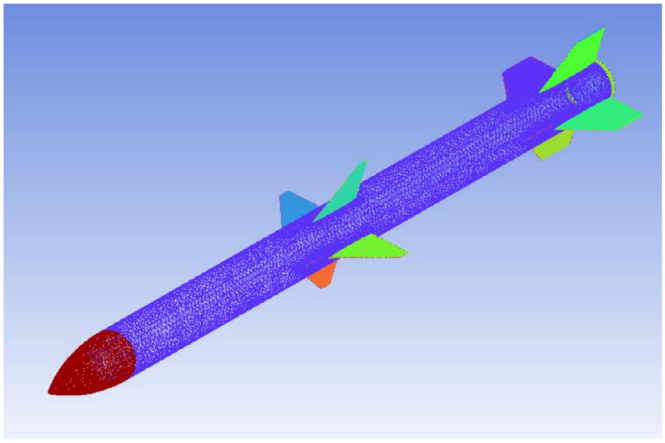
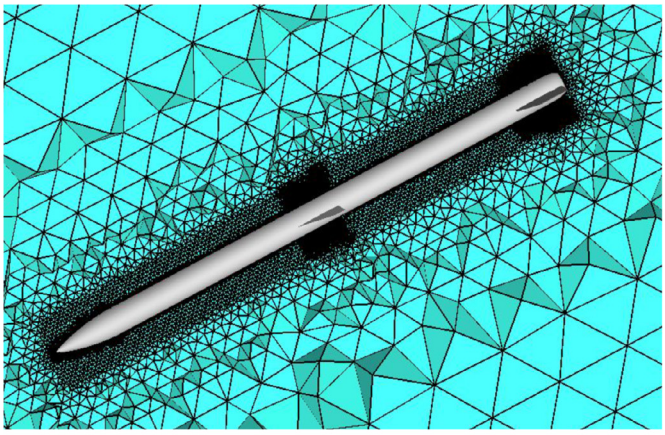


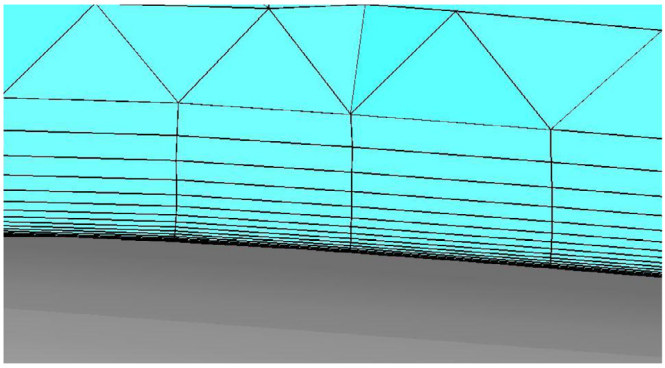

**Fig. 7.** Optimization experience transfer via transfer learning.



(a) Missile surface mesh.



(b) Side cut view of the volume mesh.



(c) Boundary layers over the control surface.

**Fig. 8.** Computational domain mesh of the baseline design configuration.

A Green–Gauss based discretization scheme was adopted and the momentum and turbulence equations are discretized using second-order upwind scheme. For the present engineering optimization problem, the Spalart–Allmaras (SA) one-equation eddy viscosity model was used to resolve RANS equations according to Ref. [30,31]. The full multi-grid (FMG) initialization was applied to accelerate the convergence speed. Convergence was achieved when the residuals reduced by at least three orders of magnitude. The CFD simulations of the numerical experiment were performed on 28 cores (two workstations with one Intel E5-2683V4 processor each).

**Table 3**
Optimized configurations of DDPG, NCGA, NSGA-II and MOPSO in DATCOM.

| Variables \ Optimizer | DDPG | NCGA | NSGA-II | MOPSO | Baseline |
|---|---|---|---|---|---|
| XLE1 | 1.25 | 1.251 | 1.313 | 1.305 | 1.72 |
| XLE2 | 3.081 | 3.121 | 3.053 | 3.16 | 3.2 |
| CHORD1_1 | 0.231 | 0.197 | 0.211 | 0.176 | 0.29 |
| CHORD1_2 | 0.031 | 0.026 | 0.036 | 0.073 | 0.06 |
| CHORD2_1 | 0.248 | 0.28 | 0.29 | 0.235 | 0.38 |
| CHORD2_2 | 0.249 | 0.081 | 0.075 | 0.236 | 0.19 |
| SSPAN1_2 | 0.239 | 0.281 | 0.232 | 0.27 | 0.23 |
| SSPAN2_2 | 0.227 | 0.285 | 0.3 | 0.219 | 0.22 |
| XCP | 0.587 | 0.574 | 0.597 | 0.581 | 0.587 |
| CL | 1.371 | 1.274 | 1.287 | 1.321 | 1.142 |
| CD | 0.44 | 0.44 | 0.44 | 0.433 | 0.44 |
| CL/CD | 3.125 | 2.894 | 2.922 | 3.048 | 2.597 |

### 4.3. Results and discussion

To validate the optimization performance, simulations were performed on the DATCOM-based evaluation workflow with DDPG and three other widely used evolutionary algorithms, namely neighborhood cultivation genetic algorithm (NCGA), improved non-dominated sorting genetic algorithm (NSGA-II) and multi-objective particle swarm optimization algorithm (MOPSO) [32–34], respectively. Herein, NSGA-II and NCGA both had 80 generations and a population size of 40. The crossover rate and mutation rate were set to 0.9 and 0.01, respectively. In MOPSO, the swarm size, number of iterations and maximum velocity were set to 40, 80 and three hundredth of the upper bound of each variable, respectively.

Table 3 shows a performance comparison of the design configurations optimized by DDPG, NCGA, NSGA-II and MOPSO in DATCOM. Compared with the baseline value, all the lift–drag ratios of the optimized results achieved significant increases under the given constraint condition. Among the optimization algorithms, DDPG performed the best by producing design configuration with lift–drag ratio 20.33% higher than the baseline. All the optimization algorithms learned to separate the wings and tails and increase the fin span length to improve the design configurations. It can be observed that the maximum drag coefficient 0.44 were achieved by DDPG, NCGA and NSGA-II, which implies the CD constraint, or the implicit design space, has reached its limit. Meanwhile, both DDPG and MOPSO obtained the approximately rectangular tails, which might be the cause of achieving lift–drag ratios higher than 3. By contrast, NCGA and NSGA-II might have fallen into local optimum due to the tails optimized by them were highly swept. From this point of view, only DDPG both reached the CD constraint upper bound and discovered the approximately rectangular tail configuration, which validates its global optimization ability and the effectiveness of the extracted optimization experience.

To fulfill the optimization task and validate the effectiveness of DDPG with transfer learning (TL-DDPG), simulations were performed on the CFD-based evaluation workflow with DDPG, TL-DDPG and previously mentioned contrast algorithms. Herein, the algorithm parameters of NCGA, NSGA-II and MOPSO were modified to keep their calculation amount close to that of DDPG. The generations and population size of NCGA and NSGA-II were reset to 50 and 20, respectively. The iterations and swarm size of MOPSO were reset to 50 and 20, respectively. Thus, the number of CFD calls in each episode or generation of all the optimization algorithms are the same.

Fig. 9 shows a comparison of the optimization process of DDPG, TL-DDPG, NCGA, NSGA-II and MOPSO. It is observed that TL-DDPG performed the best in both search speed and optimized results while MOPSO performed the second-best. The lift–drag ratio optimized by TL-DDPG is 3.18% higher than that of MOPSO and 18.67% higher than the baseline. TL-DDPG achieved a lift–drag ratio of 2.30 at the 15th episode, while it took MOPSO 40 generations

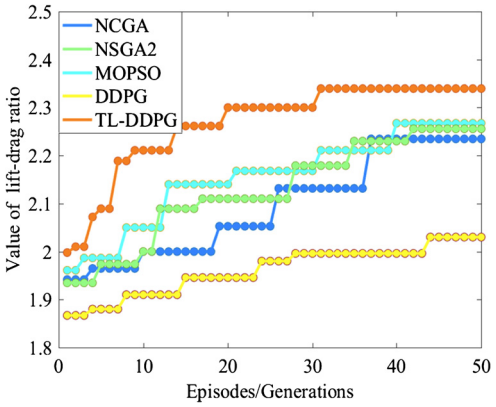



**Fig. 9.** Optimization process of NCGA, NSGA-II, MOPSO, DDPG and TL-DDPG in CFD. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

**Table 4**
Optimized configurations of TL-DDPG and MOPSO in Fluent along with the baseline.

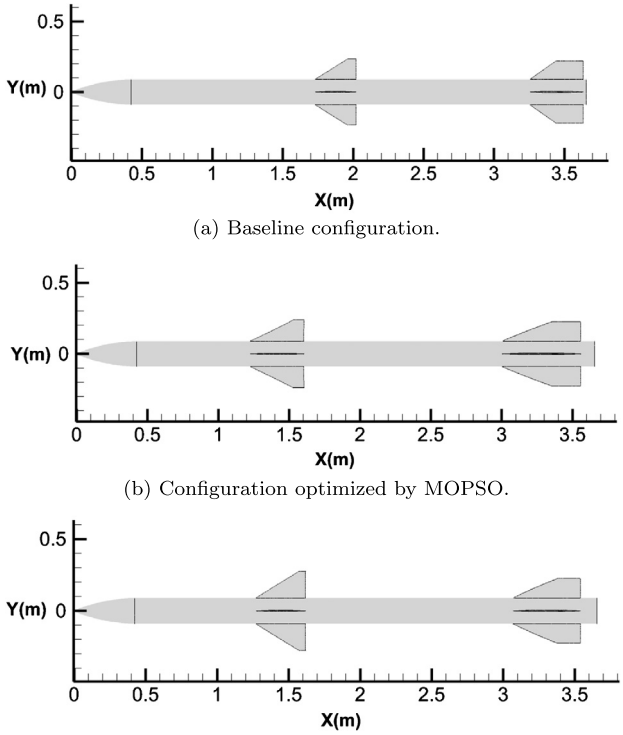
| Variables \ Optimizer | TL-DDPG | MOPSO | Baseline |
|---|---|---|---|
| XLE1 | 1.347 | 1.392 | 1.72 |
| XLE2 | 3.106 | 3.124 | 3.2 |
| CHORD1_1 | 0.354 | 0.398 | 0.29 |
| CHORD1_2 | 0.046 | 0.08 | 0.06 |
| CHORD2_1 | 0.476 | 0.565 | 0.39 |
| CHORD2_2 | 0.164 | 0.21 | 0.19 |
| SSPAN1_2 | 0.277 | 0.238 | 0.23 |
| SSPAN2_2 | 0.224 | 0.226 | 0.22 |
| XCP | 0.53 | 0.538 | 0.569 |
| CL | 1.551 | 1.512 | 1.306 |
| CD | 0.663 | 0.667 | 0.663 |
| CL/CD | 2.339 | 2.267 | 1.971 |

to achieve a similar result. By contrast, DDPG was still far from convergence with such limited iterations. From this point of view, TL-DDPG benefits greatly from the experience transferred from the pre-trained networks in DATCOM, which accounts for the approximately 62.5% faster search speed than NCGA, NSGA-II and MOPSO.

To further analyze the obtained design configurations, we choose the best configurations optimized by TL-DDPG and MOPSO and the baseline configuration as a contrast in Table 4. As the geometry and the pressure coefficient (Cp) are important indicators of the missile aerodynamic performance, Fig. 10 and 11 visualize the missile geometries and Cp distributions respectively.

Compare the optimizations in DATCOM and Fluent from Table 3 and 4, some differences and similarities can be found. In DATCOM, rectangular tails perform better than swept ones, while in Fluent the result is opposite. Besides, the quantitative impacts of design variables on the aerodynamic coefficients in DATCOM and Fluent are also different. These factors caused by different calculation methods mainly result in the differences between the optimized configurations in Table 3 and 4. However, there're still some features in common. Firstly, lengthening the fin spans can produce more lift, especially for the wings. Secondly, increasing the wing sweep angle may benefit the drag reduction. Finally, spreading the wings and tails apart helps improving CL/CD under the given flight condition. These common features are essential to accelerate the optimization process in CFD.

Fig. 10 graphically shows the optimized design configurations of TL-DDPG and MOPSO, along with the baseline configuration. It is apparent that there's no rectangular tail in the configurations optimized in Fluent as those optimized in DATCOM, which reflects the differences between the calculation methods of DATCOM and Fluent. These differences agree with the fact that DATCOM might be limited when confronted with novel configurations [14]. In contrast, CFD method has a more extensive adaptability and that's the reason why the optimization need to be accomplished through CFD simulations. It can be observed that both the optimization algorithms learned to make the wing and tail areas larger, which mainly accounts for the growth of the lift coefficient. The wing and tail area enlargement was achieved by increasing the lengths of root chord and span. Meanwhile, both TL-DDPG and MOPSO used highly swept wings and tails to maintain the drag coefficient. Apart from the wing/tail planform geometry, the wing/tail placement also has a significant impact on the center of pressure position. According to the qualitative representations of the design configurations, there must be some benefit to spatially spreading the wings and tails apart. On the one hand, moving the wings towards the nose may enlarge the pressure difference between the upper and lower surfaces, which contributes to the increase of lift coefficient. On the other hand, the wings need to be placed forward to maintain the position of pressure center by offsetting the influence of the tail area enlargement. According to Table 4, both TL-DDPG and MOPSO improved the design configuration with higher lift coefficient and nearly equal drag coefficient, while moving the center of pressure towards the center of gravity. It is consistent with the fact that relaxed static stability helps increasing the flight performance.



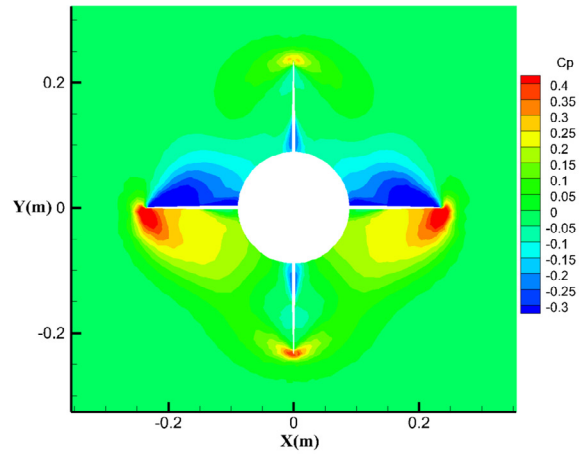

(a) Baseline configuration.

(b) Configuration optimized by MOPSO.

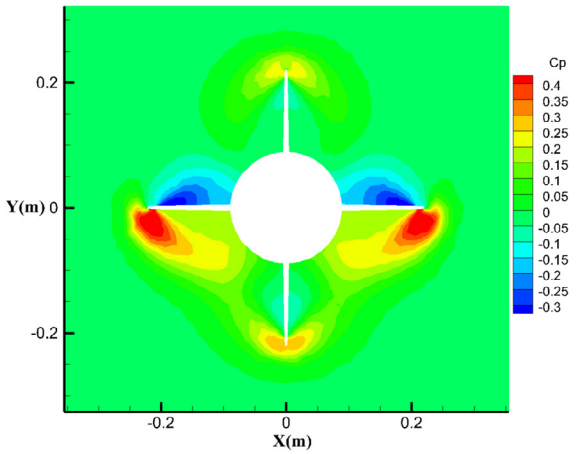
(c) Configuration optimized by TL-DDPG.

**Fig. 10.** Geometric shapes of the design configurations.

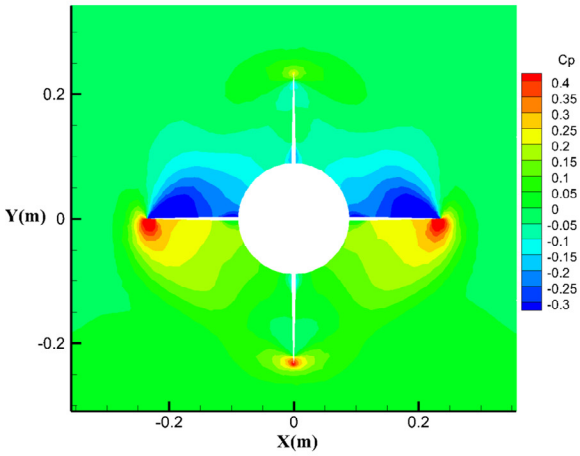
The Cp distributions around the wings and tails of the baseline and optimized design configurations are plotted in Fig. 11. It can be seen that the pressure coefficient difference between the upper and lower surfaces of the wings optimized by TL-DDPG is not only biggest in numeric but also largest in area. Additionally, owing to the increasement of fin root chord lengths and span lengths, the lift coefficients were effectively improved. Compared with the optimized wings, the optimized tails produced less lift, which implies the wings became the main lift producing surfaces after optimization. By contrast, the baseline configuration showed better balance in producing lift using the wings and tails, which might benefit the generation of controlling moment for a tail-controlled missile.



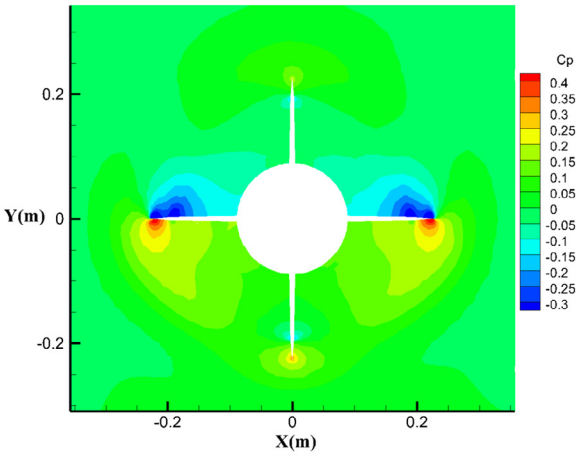

(a) Wings of baseline model.



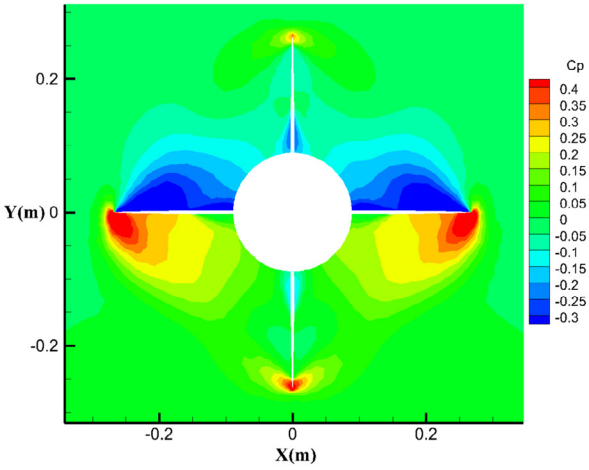

(b) Tails of baseline model.



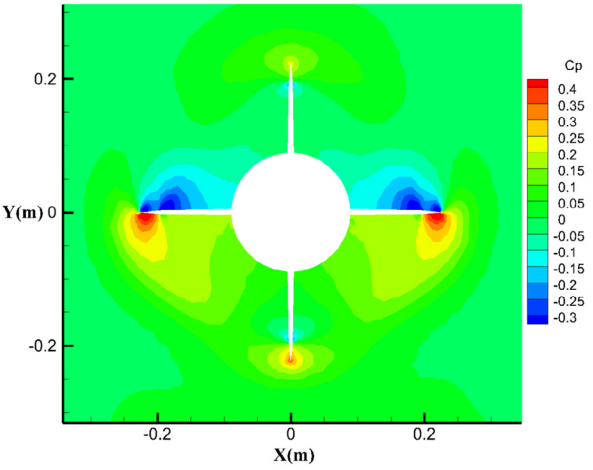

(c) Wings of the model optimized by MOPSO.




(d) Tails of the model optimized by MOPSO.




(e) Wings of the model optimized by TL-DDPG.




(f) Tails of the model optimized by TL-DDPG.

**Fig. 11.** Pressure coefficient distributions around the missile fin surfaces.

It can be concluded that the proposed optimizer with TL-DDPG successfully optimized the design configuration with lift–drag ratio 18.67% higher than the baseline through CFD simulations. With the experience extracted from DATCOM, TL-DDPG can produce CFD-based high lift–drag ratio designs much faster than traditional evolutionary algorithms, including NCGA, NSGA-II and MOPSO. In terms of obtaining similar results, TL-DDPG saved over 62.5% CFD calls for the present optimization problem. Meanwhile, compared with the time-consuming CFD simulations, the time spent in rule extraction from DATCOM is negligible.

## 5. Conclusions

This paper proposes a novel optimizer for the challenging aerodynamic shape optimization problem using two machine learning techniques, namely reinforcement learning and transfer learning. A state-of-the-art reinforcement learning algorithm, deep deterministic policy gradient (DDPG), is used to extract the optimization experience from the fast aerodynamic prediction software DATCOM. Then, the experience is reused for CFD-based aerodynamic optimization by transferring learning (TL).

The proposed optimizer is validated by an aerodynamic shape optimization problem of missile fins. By comparing the optimized configuration with three widely used evolutionary algorithms, i.e., NCGA, NSGA-II and MOPSO, through DATCOM calculations, the optimization performance of DDPG is verified. The CFD-based experimental results demonstrate that transfer learning accelerates the search speed of DDPG effectively. Moreover, compared with the three contrast algorithms, the proposed DDPG with transfer learning (TL-DDPG) increases the missile's lift–drag ratio most significantly by 18.67% higher than baseline, while successfully satisfying the constrains of drag coefficient, pressure center position and design variable values. In terms of obtaining similar results, TL-DDPG saves up to 62.5% CFD-calls for the present optimization problem.

Overall, TL-DDPG performs the best among evolutionary algorithms in both search speed and optimized results by using the optimization experience in DATCOM as priori knowledge, which is of great value for time-consuming aerodynamic optimization. Since deep neural networks can be easily extended to handle distinct but related tasks, TL-DDPG may provide guidance for more complex optimization problems.

## Conflict of interest statement

There is no conflict of interest.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No.61603210) and Aeronautical Science Foundation of China (Grant No.20160758001). The authors also thank the anonymous reviewers, whose critical and constructive comments significantly improved this paper.

## References

[1] M.J. Lighthill, A New Method of Two-Dimensional Aerodynamic Design, Aero. Res. Council, London, 1945, R&M 2112.
[2] R.M. Hicks, E.M. Murman, G.N. Vanderplaats, An assessment of airfoil design by numerical optimization, NASA TM X-3092.
[3] B. Epstein, S. Peigin, S. Tsach, A new efficient technology of aerodynamic design based on CFD driven optimization, Aerosp. Sci. Technol. 10 (2) (2006) 100–110.
[4] Y. Qiu, J. Bai, N. Liu, C. Wang, Global aerodynamic design optimization based on data dimensionality reduction, Chin. J. Aeronaut. 31 (4) (2018) 643–659.
[5] M. Secanell, P. Gamboa, A. Suleman, Design of a morphing airfoil using aerodynamic shape optimization, AIAA J. 44 (7) (2006) 1550–1562.
[6] J.E. Hicken, D.W. Zingg, Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement, AIAA J. 48 (2) (2010) 400–413.
[7] M.E. Biancolini, E. Costa, U. Cella, C. Groth, G. Veble, M. Andrejašič, Glider fuselage-wing junction optimization using CFD and RBF mesh morphing, Aircr. Eng. Aerosp. Technol. 88 (6) (2016) 740–752.
[8] R.M. Hicks, P.A. Henne, Wing design by numerical optimization, J. Aircr. 15 (7) (1977) 407–412.
[9] J.R.R.A. Martins, E. Weide, C.A. Mader, J.J. Alonso, Adjoint: an approach for rapid development of discrete adjoint solvers, AIAA J. 46 (4) (2008) 863–873.
[10] Y.R. Yang, S.K. Jung, T.H. Cho, R.S. Myong, Aerodynamic shape optimization system of a canard-controlled missile using trajectory-dependent aerodynamic coefficients, J. Spacecr. Rockets 49 (2) (2012) 243–249.
[11] J. Tao, G. Sun, J. Si, Z. Wang, A robust design for a winglet based on NURBS-FFD method and PSO algorithm, Aerosp. Sci. Technol. 70 (2017) 568–577.
[12] T.J. Sooy, R.Z. Schmidt, Aerodynamic predictions, comparisons, and validations using Missile DATCOM (97) and Aeroprediction 98 (ap98), J. Spacecr. Rockets 42 (2) (2005) 257–265.
[13] W. Zhang, Y. Wang, Y. Liu, Aerodynamic study of theater ballistic missile target, Aerosp. Sci. Technol. 24 (1) (2013) 221–225.
[14] A.D. Ronch, M. Ghoreyshi, K. Badcock, On the generation of flight dynamics aerodynamic tables by computational fluid dynamics, Prog. Aerosp. Sci. 47 (8) (2011) 597–620.
[15] R.P. Liem, C.A. Mader, J.R. Martins, Surrogate models and mixtures of experts in aerodynamic performance prediction for aircraft mission analysis, Aerosp. Sci. Technol. 43 (2015) 126–151.
[16] D. Giacché, L. Xu, J. Coupland, Optimization of bypass outlet guide vane for low interaction noise, AIAA J. 52 (6) (2014) 1145–1158.
[17] A.I.J. Forrester, A.J. Keane, Recent advances in surrogate-based optimization, Prog. Aerosp. Sci. 45 (1) (2009) 50–79.
[18] Y.-J. Liu, L. Tang, S. Tong, C.P. Chen, D.-J. Li, Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time mimo systems, IEEE Trans. Neural Netw. Learn. Syst. 26 (1) (2015) 165–176.
[19] M.E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: a survey, J. Mach. Learn. Res. 10 (Jul. 2009) 1633–1685.
[20] S. Sankaran, L. Grady, C.A. Taylor, Impact of geometric uncertainty on hemodynamic simulations using machine learning, Comput. Methods Appl. Mech. Eng. 297 (2015) 167–190.
[21] Y.C. Choi, J.H. Son, H.S. Ahn, Fault detection and isolation for a small CMG-based satellite: a fuzzy q-learning approach, Aerosp. Sci. Technol. 47 (1–4) (2015) 340–355.
[22] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint, arXiv:1509.02971.
[23] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, F. De Turck, Design and optimisation of a Q-learning-based http adaptive streaming client, Connect. Sci. 26 (1) (2014) 25–43.
[24] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[25] J. Peters, Reinforcement learning for humanoid robotics, Auton. Robots 12 (1) (2003) 1–20.
[26] E. Khalil, Computations of the flow field, temperature patterns, and thermal comfort in an air conditioned atrium, AIAA J.
[27] A. Ridluan, CFD investigation of compressible low angles of attack flow over the missile, J. Phys. Sci. Appl. 4 (6) (2014) 339–347.
[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: a system for large-scale machine learning, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, 2016, pp. 265–283.
[29] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv preprint, arXiv:1412.6980.
[30] A. Bueno-Orovio, C. Castro, F. Palacios, E. Zuazua, Continuous adjoint approach for the Spalart–Allmaras model in aerodynamic optimization, AIAA J. 50 (3) (2012) 631–646.
[31] I.C. Kampolis, X.S. Trompoukis, V.G. Asouti, K.C. Giannakoglou, CFD-based analysis and two-level aerodynamic optimization on graphics processing units, Comput. Methods Appl. Mech. Eng. 199 (9) (2010) 712–722.
[32] Z. Yan, W. Mao, X. Weng, Optimal design and experimental study of the active vibration isolation system based on NCGA algorithm, in: Proceedings of International Conference of Information Science and Management Engineering (ISME), vol. 2, IEEE, 2010, pp. 286–290.
[33] X.D. Wang, C. Hirsch, S. Kang, C. Lacor, Multi-objective optimization of turbomachinery using improved NSGA-II and approximation model, Comput. Methods Appl. Mech. Eng. 200 (9) (2011) 883–895.
[34] S. Ding, C. Chen, B. Xin, P.M. Pardalos, A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches, Appl. Soft Comput. 63 (2018) 249–267.