Research Reports of National Institute of Technology, Nagaoka College, Vol. 59, 2024

Paper

Shape optimization of an object in an incompressible viscous fluid for drag minimization (Considerations for introducing a shape selection process using deep reinforcement learning)

Yudai SUGIYAMA[1], Takahiko KURAHASHI[2] and Toshihiko ETO[3]

[1]Mechanical Engineering, Master's Program, Graduate School of Engineering, Nagaoka University of Technology

[2]Department of Mechanical Engineering, Institute of GIGAKU, Nagaoka University of Technology

[3]Department of Civil Engineering, National Institute of Technology, Nagaoka College

In this study, we introduce and consider a shape optimization process using deep reinforcement learning for the drag minimization of an object in an incompressible viscous fluid. Most previous studies on shape optimization have used the adjoint variable method. However, it has been demonstrated that shape optimization results are dependent on the initial shape and that the final shape is not necessarily the optimal shape[1)]. Deep reinforcement learning is a method that learns only from the interactions between an agent and its environment. We believe that this is appropriate for optimization where the correct answer is unknown. Furthermore, if the shape design variables increase, the behavior patterns that an agent can consider will significantly increase. Therefore, we believe that deep learning is more suitable for this purpose.

Key Words : *Deep Q network, Reinforcement Learning, Q-learning, FreeFEM++*

1. Introduction

Shape optimization is a method used in product shape design. Shape analysis with minimum drag and maximum lift is one of the most important design processes in shape determination. Shape optimization using the adjoint variable method has been applied to several optimization problems, such as drag minimization problems. However, it has been shown that shape optimization results depend on the initial shape and that the final shape is not necessarily the optimal shape[1)]. However, in recent years, with the development of machine learning and computer performance, called deep learning, the field of image recognition has shown significantly higher performance than existing methods in general object recognition contests[2)]. Moreover, a method called deep Q-learning network (DQN), which learns solely from trial and error, has been devised and shown to outperform skilled humans in a majority of games[3)]. Therefore, this study proposes shape optimization with a low dependence on the initial shape. A DQN that learns by searching for solutions to problems for which it does not know the answers is expected to perform global optimization.

2. DQN

2.1 What is DQN?

In reinforcement learning (RL), a specific environment is provided, and the behavior is learned such that the reward obtained is maximized. As shown in Fig. 1, RL consists of an agent that makes behavioral decisions for the model based on the policy and an environment that provides rewards. The environment outputs rewards and the current state as feedback for the agent's actions. This agent learns to perform actions that are expected to be maximally rewarding in the current state.

RL is commonly used for robot control, Shogi, and Go and is often used for dynamic problems that depreciate over time. However, other research[4),5)] applied it for optimization in static and time-invariant problems, such as the optimization of the hyperparameters of neural net-works. By receiving only the value of the first step as feedback in an inherently time-evolving finite element analysis, the problem addressed in this study can be considered static. Various methods have been proposed for action selection, and this study used a DQN, which is one such method. The DQN determines actions based on the action value function, represented by the following equation. This function was further modeled using a deep neural network.

$$\begin{aligned} Q(s_t,a_t) &\leftarrow Q(s_t,a_t) \\ &\quad + \alpha\{r_t + \gamma \max_{a\in A} Q(s_{t+1},a) - Q(s_t,a_t)\} \end{aligned} \tag{1}$$

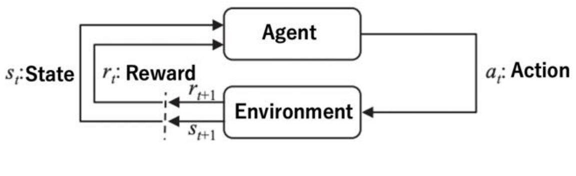

Fig. 1 Conceptual diagram of RL[4)]

where, learning rate, action-value function, and discount rate are denoted by α, Q and γ ($0 < \alpha, \gamma \leq 1$), respectively, and A represents the entire set of possible actions. The second term of Eq.(1) represents the difference between the expected action and the current values.

2.2 Reward

The reward is defined in Eq.(2) as:

$$r_t = F_{D(\mathrm{init})} - F_{D(t)} \tag{2}$$

Here, the drag forces on the initial shape and the state at the t epoch are denoted by $F_{D(\mathrm{init})}$ and $F_{D(t)}$. The flow field is an incompressible viscous flow, and the governing equations are the Navier-Stokes equations in Eq.(3) and the continuity equation in Eq.(4).

$$\dot{u_i} + u_j u_{i,j} + p_{,i} - \frac{1}{Re} u_{i,jj} = 0 \quad \text{on} \quad \Omega, \tag{3}$$

$$u_{i,i} = 0 \quad \text{on} \quad \Omega, \tag{4}$$

where, flow velocity, pressure, and Reynolds number are denoted by u_i[m/s], p[Pa], and Re, respectively. The computational domain is considered as a typical problem, as shown in Fig. 2[1)]. The boundary conditions are as follows:

$$u_i = \hat{u}_i \quad \text{on} \quad \Gamma_U, \tag{5}$$

$$T_1 = 0, \quad u_2 = 0 \quad \text{on} \quad \Gamma_S, \tag{6}$$

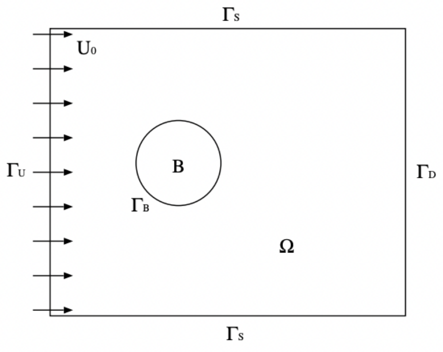

Fig. 2 Computational domain and boundary conditions[1]

$$u_i = 0 \quad \text{on} \quad \Gamma_B, \tag{7}$$

$$T_i = 0 \quad \text{on} \quad \Gamma_D, \tag{8}$$

$$T_i = \{-p\delta_{ij} + \mu u_{i,j}\} n_{j,} \tag{9}$$

where, the stress and unit normal vectors outward from the boundary are denoted as t_i , n_j , respectively. The fluid force acting on the object is denoted by F_i, where the drag and lift forces are denoted as when $i = 1, 2$, respectively. F_i is obtained using Eq.(10), the integral of T_i at the boundary Γ_B.

$$F_i = -\int_{\Gamma_B} T_i \, d\Gamma. \tag{10}$$

Therefore, the reward in Eq.(2) is the difference, that is, a positive reward if the drag force is smaller than the initial shape at the t epoch. Conversely, if it becomes larger than the initial shape at the t epoch. it represents a negative reward as a penalty.

2.3 Network update

RL defines the value of an action when s and a are performed according to π as follows:

$$Q^{\pi}(s, a) \coloneqq \mathbb{E}_{s,a,\pi}[\sum r_t] \tag{11}$$

where the policy for choosing action is denoted by π. The optimal values are as follows:

$$Q^{*}(s, a) \coloneqq \max_{\pi} Q^{\pi}(s, a) \tag{12}$$

Instead of using tabular encodings, neural networks are used to learn estimates by parameterized $Q(s, a; \theta)$ to scale up to large problems such as models with many design variables. The following formula is used to update the network parameters.

$$\theta_{t+1} \leftarrow \theta_t + \alpha \left(y_t^Q - Q(s_t, a_t; \theta_t) \right) \nabla_{\theta} Q(s_t, a_t; \theta_t) \tag{13}$$

where、y_t^Q is target value as follows:

$$y_t^Q = r_t + \gamma \max_{\text{a}} Q(s_{t+1}, a; \theta^-) \tag{14}$$

where θ^- are the parameters of the target network and $\theta^- = \theta_t$. Several important changes in network updates improve the learning stability of the DQN[3]. One is that the algorithm is not learned entirely online but rather by sampling from the experience buffer. The idea is to use a target network with parameters θ^- copied from the main network. The target network updates its parameters at each defined epoch $(\theta^- \leftarrow \theta_t)$. This method is called a Double DQN (DDQN), which modifies the target value as follows:

$$y_t^Q = r_t + \gamma \max_{\text{a}} Q(s_{t+1}, \arg\max_{a} Q(s_{t+1}, a; \theta_t)\,; \theta^-) \tag{15}$$

where the main network selects the action with the highest expected action value, as expressed in Eq.(15) but adopts the policy ϵ-greedy method and sometimes randomly selects another action.

3. Numerical study

Shape optimization using the DQN and adjoint variable method was performed based on a previous study's analytical conditions[1]. The flow field was analyzed using FreeFEM++ v4.9, and the free software application focused on solving partial differential equations using the finite element method. The programming language Python 3.8.0 was used to implement the DQN, and the machine learning framework PyTorch 1.13.1 was used.

3.1 Analysis conditions

Fig. 3 and 4 show the finite element mesh, and computational domain, respectively. The parameters used in the DQN analysis are listed in Tab. 1. In the finite element analysis, the time step width was 0.03[s], end time step was 2,000[s], and the Reynolds number was 100.

where, the design variable in the DQN analysis is given by the elliptical equation as in Eq.(16).

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1 \tag{16}$$

The action was to increase or decrease a. To update the shape with a constant area, b was obtained from

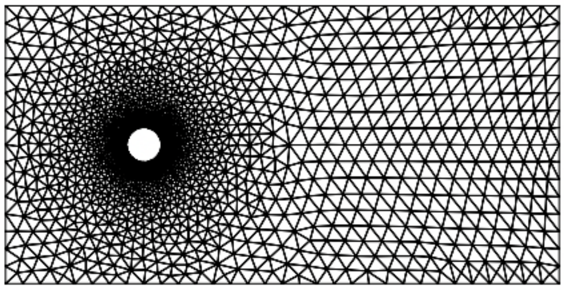

Fig. 3 Finite element mesh[1)]

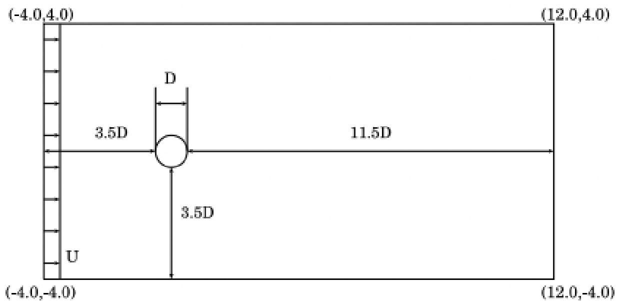

Fig. 4 Computational domain[1)]

Tab.1 DQN analysis conditions

Max epoch	-	200
Max Episode	-	100
Worm up epoch	-	10
Batch size	-	5
Learning rate	α	0.00025
Adam parameter 1	β_1	0.9
Adam parameter 2	β_2	0.999
Adam parameter 3	ϵ_1	1×10^{-8}
Epsilon	ϵ_2	0.1
Gamma	γ	0.9
Initial design variable	a, b	0.5[m]
Action width	-	0.01[m]
Newton threshold	ϵ_3	1×10^{-6}

a and the area of the initial shape was determined using Newton's method. The Adam optimization method, which is a sequential optimization method that uses first-order derivatives of the objective function with various mod-ifications to the steepest descent method, was used. Specifically, this method combines RMSprop and mom-entum SGD. Eq.(1) is represented by a neural network, in which the second term is the difference between the expected action value and the current expected value, as follows:

$$E=\frac{1}{2}\sum_{n=1}^{N}\left(Q^n(s_t,a;\theta)-y_t^Q\right)^2 \tag{17}$$

$$\nabla E := \frac{\partial E}{\partial w} \tag{18}$$

where, E and ∇E in Eqs.(17) and (18) denote the error function and gradient of the error function, respectively. By applying Adam to them we obtain:

$$m_i^{[t]}=\beta_1 m_i^{[t-1]}+(1-\beta_1)\nabla E \tag{19}$$

$$v_i^{[t]}=\beta_2 v_i^{[t-1]}+(1-\beta_2)\nabla(E^{[t]})^2 \tag{20}$$

$$\widehat{m}_i^{[t]}=\frac{m_i^{[t]}}{(1-\beta_1^t)} \tag{21}$$

$$\hat{v}_i^{[t]}=\frac{v_i^{[t]}}{(1-\beta_2^t)} \tag{22}$$

$$\theta_i^{[t]}=\theta_i^{[t-1]}-\alpha\frac{\widehat{m}_i^{[t]}}{(\sqrt{\hat{v}_i^{[t]}}+\epsilon_1)} \tag{23}$$

where, $m_i^{[t]}$ in Eq.(19) is the momentum for suppressing the ith parameter vibration at t, $m_0=0$. β_1 is the decay rate, the hyperparameter that determines how much of the past gradient is accumulated, and $v_i^{[t]}$ in Eq.(20) denotes the accumulation of past gradients for the ith learning parameter at time t. The initial value was zero. The ratio of the past gradient to the latest gradient accumulated was $\beta_2 : 1-\beta_2$. Therefore, past information accumulates exponentially, allowing more past information to be largely ignored as the epochs progressed, and eliminating the gradient loss problem.

3.2 Result

Fig. 5 shows the transition from the initial to convergent solutions during the verification of the initial shape, where a is the width from the center of the ellipse, indicating gradual widening and convergence to a width of 2.14 for an initial condition of width 0.5. The width of the movement in one action was 0.01[m]. Fig. 6 shows the final shapes of shape optimization using DQN and the associated variable method.

The adjoint variable method seeks a bullet shape with a pointed tip and the rounded rear. The pressure distributions after 2,000 steps for the two geometries are shown in Fig. 7. The drag force history for 2,000 steps is shown in Fig. 8. The final shape of the adjoining variable method appears to be a backward vortex, and the flow appears to peel off around the middle of the object. Pressure resistance is dominant in vortex-generating objects. Contrarily, there were no noticeable eddies in the DQN shape, suggesting that almost no exfoliation occurred. Therefore, we believe that the frictional drag was dominant, but the frictional drag was not that large because of the smooth shape.

Immediately after the inflow, the drag was large, with drags of 39.88 and 29.67 for the DQN and adjoint variable methods at the first step, respectively. Fig. 8 shows the drag after the second step. The cumulative values of the square of the drag for Steps 1 to 2,000 were 26.42 and 50.46 for the DQN and adjoint variable methods, respectively. It is difficult to represent the complexity of the geometry in this analysis because of the small number of design

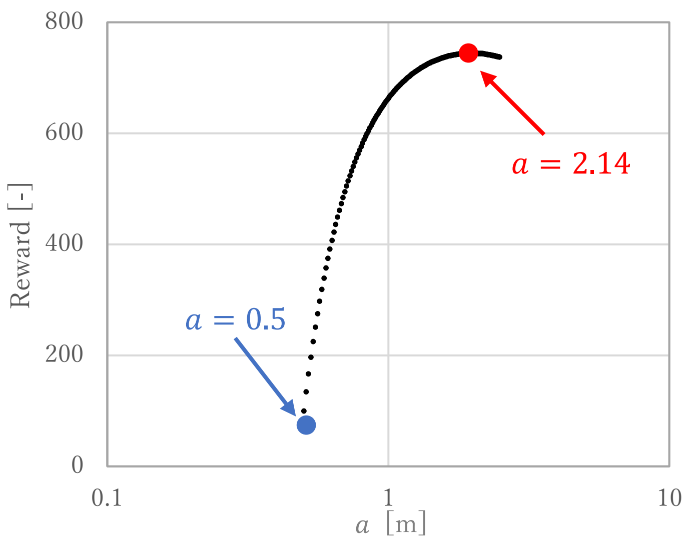

Fig. 5 Transition from initial to convergent

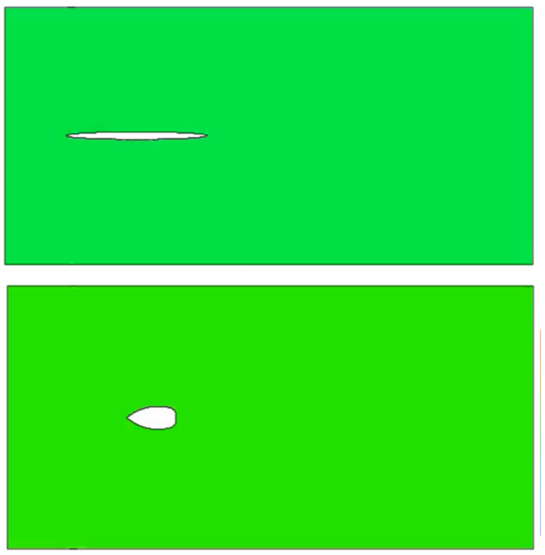

Fig. 6 Final shape of DQN (top) and adjoint variable method (bottom)

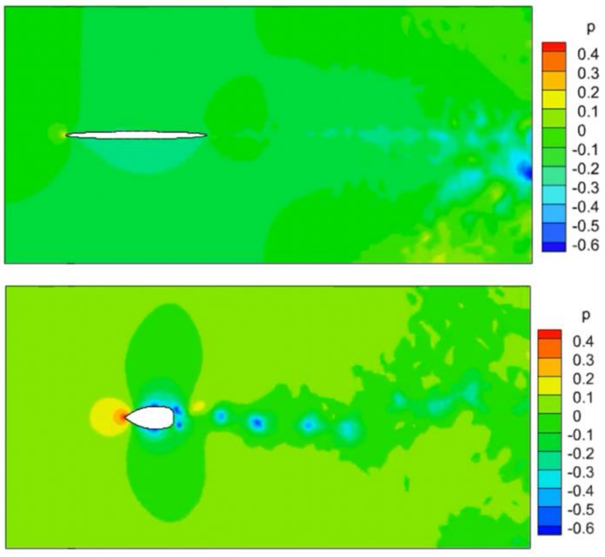

Fig. 7 Pressure distribution for DQN (top) and adjoint variable method (bottom)

variables in the DQN. However, we believe that resetting the shape and repeating the search for each episode is less likely to result in a locally optimal solution. We also believe that a larger step width and wider search compared with conventional analysis conditions contributed to the success of the search.

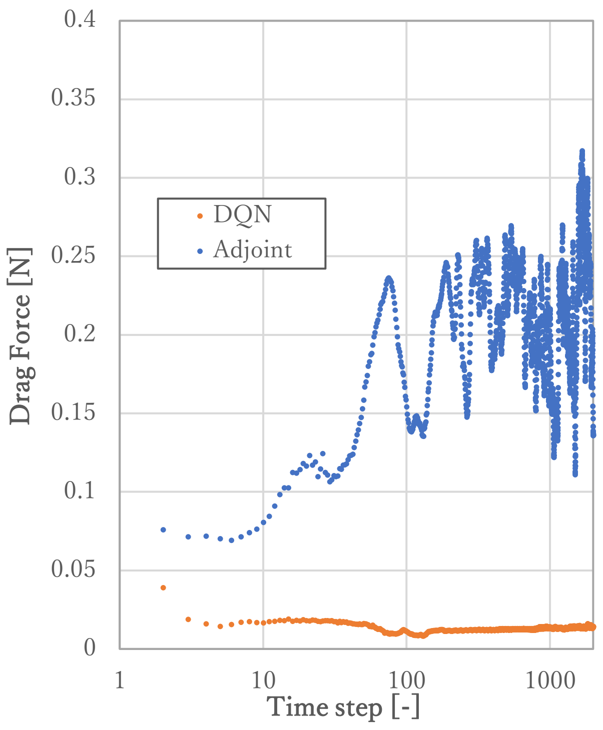

Fig. 8 Drag history

4. Conclusion

In this study, we proposed a shape optimization method using a DQN and compared it with a conventional shape optimization method that uses adjoint variable method. Although the number of design variables was different and therefore, not completely comparable, we believe that the DQN is useful for shape optimization of models with fe w design variables based on drag forces. In the future, we intend to verify this using additional design variables.

Acknowledgement: We would like to thank Editage (www.editage.com) for English language editing.

References

1) Hiroko Yagi and Mutsuto Kawahara: *Optimal shape determination of a body located in incompressible viscous fluid flow*, Computer Methods in Applied Mechanics and Engineering, 196, 5084-5091, 1 November 2007.

2) Simonyan, K. and Zisserman, A:Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556, 2014.

3) V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis: *Human-level control through deep reinforcement learning*, Nature., 518, 529–533, 2015.

4) S. S. Richard and G. B. Andrew: *Reinforcement Learning: An Introduction*, MIT Press., 2014, 2015.

5) K. Li, J. Malik: *Learning to optimize*. ICLR 2017 conference., 5 November 2016 (modified: 22 July 2022).

6) M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford and Nando de Freitas: *Learning to learn by gradient descent by gradient descent*, arXiv preprint arXiv, 1606, 044-74v2, 14 Jun 2016.

(Received July 19, 2023)