ICAS 2021 SHANGHAI
International Council of the Aeronautical Sciences
32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

APPLICATION OF REINFORCEMENT LEARNING IN 1-D AERODYNAMIC DESIGN OF AXIAL COMPRESSOR

Yi Liu[1], Hang Xiang[1], Jiang Chen[1]

[1]School of Energy and Power Engineering, Beihang University Beijing, China

Abstract

The traditional aerodynamic design of compressors depends on the experiences of the designer and many optimization calculations. As a result, the performance of the compressor depends on the design level of the designer. A new design method based on reinforcement learning is proposed for the 1-Dimensional aerodynamic design of axial compressors. Deep Deterministic Policy Gradient (DDPG) algorithm is used to extract design experience from the 1-D experiences design program HARIKA. After training, the efficiency was increased by 2.0% relative to the initial value, and the design margin was 23%, which met the design requirements. Under the same design time, the efficiency of new method optimization results was 0.2% lower than the traditional method's results following by better pressure ratio performance at the design point, which proved the effectiveness of the new design method. Furthermore, the approach gained the redesign ability of after training. Compared to the Differential Evolution (DE) optimization results (16000 steps), the efficiency of DDPG redesign results was 0.3% lower and the number of steps is 20, which verified the quick redesign ability of the new design method.

Keywords: reinforcement learning, axial compressor mean-line aerodynamic design, optimization method

1. Introduction

The main difficulties of the aerodynamic design of the compressor are reflected in two aspects: one is to pursue the high performance of the design conditions; the other is to consider the wide range of working conditions. Typically, the process of compressor aerodynamic design usually consists of one-dimensional design, blade modeling, and three-dimensional analysis, as well as other full three-dimensional aerodynamic design systems and optimization methods. To reach the design requirements, designers need to manually modify the parameters many times, and the modification of parameters needs to combine with expert experience. This method relies on expert experience and will take plenty of time. With the development of computer science, it is possible to use artificial intelligence technology to guide the aerodynamic design process of axial compressors.

In terms of using machine learning for aerodynamic design, it is mainly divided into three directions: First, it uses ANN to replace the iterative calculation part of traditional design [1]. In 2000, Michel [2] used ANN to replace the CFD calculation part in the design optimization process, which greatly reduced the optimization design time. Ghorbanian [3] used experimental data to train neural networks and studied the application of different types of ANNs such as general regression neural networks, multilayer perceptron networks, and radial basis function networks in the prediction of compressor characteristics. In the training of the layer perceptron network, the accuracy of predicting the characteristics of the compressor can reach 92%, which can improve the accuracy and speed of the calculation during the design process. In 2016, Jonathan [4] used a deep convolutional neural network to directly solve the full-field pressure distribution in the calculation of the three-dimensional flow field, which greatly improved the convergence speed of the Euler equation and reduced the iteration time. The second is to use artificial neural networks to improve traditional optimization algorithms. In 2014, Timoleon [5] used an artificial neural network to guide the multi-objective particle swarm optimization problem. After the optimization algorithm gives the target solution, the artificial neural network first determines the feasibility of the target solution and then submits it to the CFD program. This method can save more computing resources. The third direction is to use reinforcement learning to train the computer from scratch to perform the aerodynamic design of the axial compressor independently. Reinforcement learning algorithm was proposed by Sutton [6]. Due to the limitation of computing power, researchers did not pay much attention until the emergence of "Alpha Go". In the application of compressor aerodynamic design, Goel [7] used a reinforcement learning method based on a genetic algorithm to improve the aerodynamic design of turbine blades and successfully designed turbine blades using a classifier model, which saved 33% iteration time compared to the traditional methods. In 2013, Mumit [8] used the Q-learning algorithm to design the thermoacoustic heat engine. Compared with the traditional design method, the energy conversion efficiency was increased by 3.29%. The design problems of thermoacoustic heat engines are similar to the 1-D aerodynamic design of axial flow compressors, and they all have a large number of design parameters. Reinforcement learning methods can better solve the problem of too many optimized design parameters.

In this paper, we proposed a new design method based on Reinforcement learning, which uses reinforcement learning to train the machine how to design the compressor independently. In this process, human intervention is minimized to achieve the Design requirements. Reinforcement learning is a branch of machine learning, which is mainly composed of agents, environments, states, actions, and rewards. After the agent performs an action in one state, it enters the next state, and the environment gives a reward according to the new state based on a certain strategy. Subsequently, the agent will repeat the circle many times, train the neural network, and acquire design experience. This paper focuses on the mean-line aerodynamic design of axial compressors and uses the one-dimensional axial compressor characteristic calculation program (HARIKA) to explore the feasibility of the reinforcement learning algorithm in guiding the aerodynamic design of axial compressors.

2. Statement of the problem

The essence of this process is a kind of thermal calculation instead of flow analysis. In this process, many contradictory choices must be continuously judged and decided based on experience. A mass of data required for the calculation itself also comes from experiments. After this process, the diameter, speed, number of stages, and load distribution of the compressor can be determined. Although the calculations are simple, they are important as the most basic part of aerodynamic design. After the one-dimensional design is determined, subsequent design analysis can only be performed in its specified frame. We use HARIKA to help evaluate the compressor performance. HARIKA is a widely used one-dimensional compressor aerodynamic design program, with many empirical formulas built in. It can quickly give design results based on input conditions, which meet the requirements of the reinforcement learning for the evaluation program. Aiming at the one-dimensional aerodynamic design of the compressor, we choose efficiency as the design goal, and to ensure the stability of the compressor, we choose the margin as the constraints. Through a noticeably short time of calculation, HARIKA will give the values of efficiency and minimum margin. Following is a general description of the design model:

Pressure ratio: 16.0

Mass Flow: 20.5kg/s

Atmospheric pressure: 101325Pa

Total temperature: 288.15K

Rotation speed: 15350rpm

Surge margin: ≥15%

Stage number: 13 stages

To simplify the design process, the form of the flow channel was set as equal middle diameter. The optimization goal of the model is to maximize the efficiency at the design point. And the surge margin is not lower than 15%.

The definitions and ranges of the design variables are shown in Table 1. The length variables are in meters and the angle variables are in degrees. There are 11 parameters, including rotating speed, hub ratio, axial speed distribution and reaction force, etc. These are all the parameters HARIKA needs for design calculation. In this paper, we will train an agent to learn how to design the compressor. After being given the design requirements, the agent can determine all the rest design variables automatically.

Table 1 The definition and ranges of design variables

Design Variable	Definition	Ranges
DK	Hub/tip ratio	0.43-0.65
CA1(m/s)	Inlet air velocity	130-190
CA2(m/s)	Middle air velocity	120-190
CA3(m/s)	Outlet air velocity	110-150
ALF	Inlet airflow angle	53-85
TAO	Reaction	0.40-0.60
HZ1	Tip load factor	0.15-0.35
HZZ	Root load factor	0.15-0.35
HR1	Tip pitch ratio	1.0-1.6
HRZ	Root pitch ratio	0.6-1.4
HORDA	Minimum chord length	0.030-0.070

3. Method

3.1 The Choose of Specific Algorithm

Through training, experience is extracted from HARIKA through deep reinforcement learning (RL). The aerodynamic optimization process is a Markov Decision Process (MDP) which meets RL's essential requirement. Unlike other algorithms, reinforcement learning requires constant interaction with the environment, the HARIKA program, to train neural networks. As for optimization problem, the algorithm requires multi-dimensional and continuous inputs to meet the adjustment of multiple design variables. Further, we choose DDPG to test the specific algorithm of reinforcement learning. The Q-learning algorithm relies on table data storage, which is not applicable for continuous states; The Deep Q-learning network (DQN) uses a neural network to approximate the behavior value function so that it can be input as multi-dimensional, continuous state that meets the requirements of this design. However, the problem of DQN is that it can only provide discrete parameter changes, which may miss the point of the global optimal result. The DDPG, Deep Deterministic Policy Gradient, is a value-based deterministic strategic algorithm, and its action and state space are multi-dimensional continuous. Sufficient state information can be learned while making precise parameter adjustments to meet the design requirements.

3.2 Deep Deterministic Policy Gradient

To extract design experience from HARIKA and simulate the human design process, we need to train the agent's action strategy, that is, the next action that the agent takes after obtaining the current state to obtain a higher reward. The strategy is defined as follows:

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

μ_θ represents the strategy μ in the case of the current network parameter θ.

To avoid falling into a local optimum, we set a decay coefficient for the reward, the formula is as follows:

$$G_t = \sum_{n=0}^{N} \gamma^n r_{t+n+1} \tag{2}$$

DDPG uses deterministic strategy μ to select actions $a_t = \mu(s_t|\theta^\mu)$where θ^μ is a parameter of the strategy network that generates deterministic actions. DDPG is divided into two parts, actor network and critical network. The algorithm uses the policy network μ to act as an actor, and the value network to fit the (s, a) function to act as a critical role. In order to solve the optimal strategy, value function and action state value function are introduced to evaluate the pros and cons of a certain state and action. The value function is defined as follows:

$$J(\theta_\mu) = E_{\theta^\mu}\left[\sum_{n=0}^{N} \gamma^n r_{t+n+1} | s_t = s\right] = E_{\theta^\mu}[r_{t+1} + \gamma V(s_{t+1}) | s_t = s] \tag{3}$$

The action-states value function is defined as:

$$Q^\mu(s_t, a_t) = E[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))] \tag{4}$$

Silver proved that the gradient of the objective function using the μ strategy is equivalent to the expected gradient of the Q function using the μ strategy:

$$\frac{\partial J(\theta_\mu)}{\partial \theta_\mu} = E_s\left[\frac{\partial Q(s, a|\theta^\mu)}{\partial \theta_\mu}\right] \tag{5}$$

Cause the strategy is deterministic, the gradient of the Actor network is:

$$\frac{\partial J(\theta_\mu)}{\partial \theta_\mu} = E_s\left[\frac{\partial Q(s, a|\theta^\mu)}{\partial a}\frac{\partial \pi(s, \theta^\mu)}{\partial \theta_\mu}\right] \tag{6}$$

$$\nabla_\theta J_\beta(\mu_\theta) = \int_S \rho^\beta(s) \nabla_\theta \mu_\theta(s) Q^\mu(s, a)|_{a=\mu_\theta} ds = E_{s\sim\rho^\beta}\left[\nabla_\theta \mu_\theta(s) Q^\mu(s, a)|_{a=\mu_\theta}\right] \tag{7}$$

β is the random noise, The value gradient on the Critic network is:

$$\frac{\partial L(\theta^Q)}{\partial \theta^Q} = E_{s,a,r,s' \sim D}\left[(TargetQ - Q(s, a|\theta^Q)\frac{\partial Q(s, a|\theta^\mu)}{\partial \theta_\mu}\right] \tag{8}$$

$$TargetQ = r + \gamma Q'(s', \pi(s'|\theta^{\mu'})|\theta^{Q'}) \tag{9}$$

The actor network is used to output actions, and the critical network is used to evaluate the value of actor output actions. So, the updated formula of DDPG is:

$$\delta_t = r_t + \gamma Q^{\omega^-}\left(s_{t+1}, \mu_{\theta^-}(s_{t+1})\right) - Q^\omega(s_t, a_t) \tag{10}$$

$$\omega_{t+1} = \omega_t + \alpha_\omega \delta_t \nabla_\omega Q^\omega(s_t, a_t) \tag{11}$$

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^\omega(s_t, a_t)|_{a=\mu_\theta(s)} \tag{12}$$

$$\theta^- = \tau\theta + (1-\tau)\theta^- \tag{13}$$

$$\omega^- = \tau\omega + (1-\tau)\omega^- \tag{14}$$

3.3 Optimization Settings

Fig. 1 shows the flowchart of the new design method We set 12 independent design variables [ZS, DK, CA1, CA2, CA3, ALF, RT, HZ1, HZZ, HR1, HRZ, HORDA] as the input of the DDPG algorithm and the corresponding variations as the output. The corresponding variations are the followings: [ΔZS, ΔDK, ΔCA1, ΔCA2, ΔCA3, ΔALF, ΔRT, ΔHZ1, ΔHZZ, Δ HR1, Δ HRZ, Δ HORDA] to get enough non-design points, the characteristics of six relative rotational speeds are calculated by the HARIKA program. The design goal is to maximize the efficiency at design point while ensuring that all the non-design point margin is not lower than 15%.

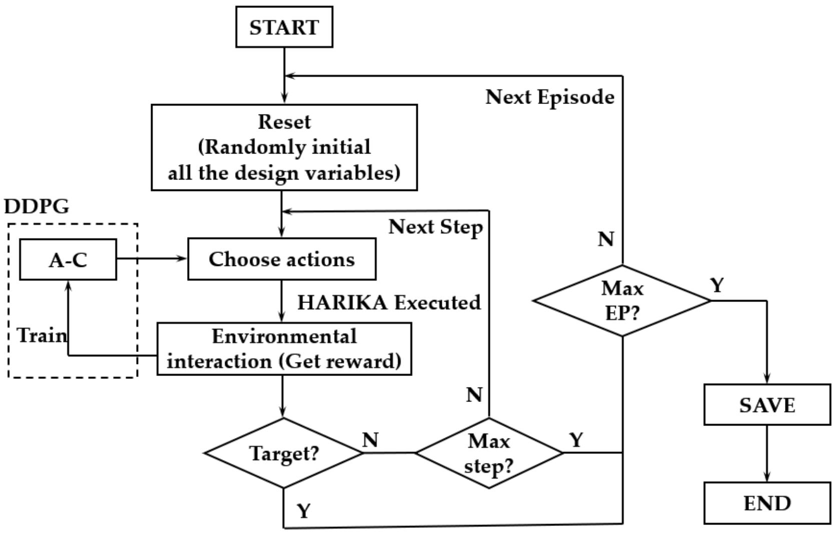

Figure 1-The flowchart of the new design method

In this case, the states of DDPG are set as an 11-dimensional array, and the values are equal to the design variables. The actions array are set as the same size as the states, but the actions' values are normalized before the next state since the boundaries of variables are different from each other.

For rewards, according to the optimization goals and limits, the rewards for efficiency (η) and margin (ky) are as follows:

$$reward_\eta = \begin{cases} -50 & \eta < 0 \\ -50 + 60\eta & 0 \le \eta \le 0.85 \\ 845\eta - 720 & 0.85 < \eta \le 0.92 \\ -50 & \eta > 0.92 \end{cases} \tag{15}$$

$$reward_{ky} = \begin{cases} -50 & ky < 0 \\ 200ky - 50 & 0 \le ky \le 0.15 \\ 400ky - 80 & 0.15 < ky \le 0.20 \\ 5 & 0.20 < ky \le 0.30 \\ 0 & 0.30 < ky \le 0.50 \\ -30 & ky > 0.50 \end{cases} \tag{16}$$

The total reward is the sum of the two rewards. By setting the reward, the multi-objective optimization can be achieved.

Algorithm 1：Use DDPG to extract the optimization experience from HARIKA

```
Randomly initialize actor network and critic network along with corresponding target network:
For episode = 1 to Ep do
    Initialize and randomly reset the design variables.
    For step = 1 to N 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.
        Save maximum efficiency values that meet margin requirements.
        Sample a minibatch of n transitions from replay buffer.
        Update critic network by minimizing the loss function according to equation 14.
        Update actor network using the sampled policy gradient according to equation 13.
        Update the target networks.
    End for
End for
```

4. Results

TensorFlow and Adam algorithm is applied to build a neural network. The Adams algorithm is used as a convergence criterion. The learning rate is set to 0.001 and 0.001. The reward discount is set to 0.99. The replay buffer size and the minibatch were 1500 and 32, respectively. To verify the effectiveness of the optimization method, we use the ISIGHT to calculate the same optimization problem using a multi-island genetic algorithm and particle swarm optimization algorithm.

Multi-island genetic algorithm and particle swarm algorithm set the total number of rounds to 20,000, which is the same as DDPG algorithm. The results of the three methods are shown in Table 2. Result of optimizations.

Table 2-Results of optimization

Run	DK	CA1	CA2	CA3	ALF	TAO	HZ1	HZZ	HR1	HRZ	HORDA	η	ky
Baseline	0.450	130	130	130	65	0.50	0.240	0.200	1.3	0.9	0.047	0.836	0.03
DDPG	0.505	134	156	116	77	0.46	0.247	0.259	1.3	1.05	0.053	0.856	0.23
MOGA	0.610	168	130	134	71	0.40	0.243	0.295	1.04	0.78	0.064	0.858	0.49
MOPS	0.625	163	129	140	66	0.40	0.214	0.289	1.6	0.80	0.070	0.854	0.53
EVOL	0.550	152	143	135	76	0.40	0.240	0.310	1.17	0.84	0.053	0.858	0.36

DDPG and the other two optimization algorithms show significant design optimization effects compared to the baseline value. After the optimization design, the efficiency reached 85.6%, which was 2.0% higher than the initial value, and the surge margin increased by 20%, meeting the optimization limits. Compared with the traditional optimization algorithm, the optimal value is the same as the multi-island genetic algorithm, and it is slightly smaller than the particle swarm algorithm under the condition of satisfying the margin constraint.

Fig. 2 and Fig. 3 show the flow efficiency characteristic line and flow pressure ratio characteristic line of the DDPG algorithm, genetic algorithm, and particle swarm algorithm.

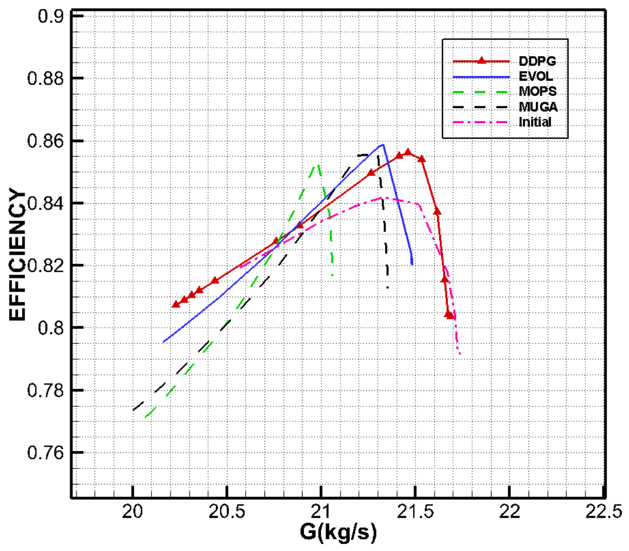

Figure 2-Efficiency characteristic line

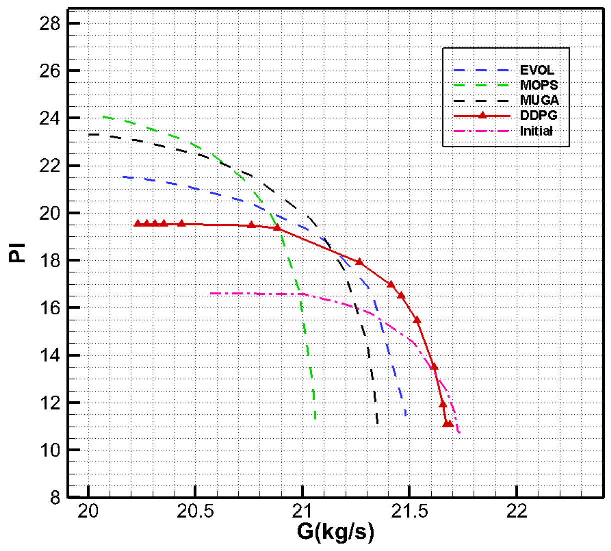

Figure 3-Pressure ratio characteristic line

At 100% speed at the design point, the efficiency value is equivalent, but the pressure ratio of the design result of DDPG is significantly higher than the other two algorithms. At the non-design point, the efficiency of DDPG is significantly lower than the other two methods. Because only the efficiency of the design point and the margin of the non-design point are considered when designing the algorithm. And the efficiency of the non-design point is not reflected in the reward, which may be added to the calculation in subsequent work.

The reward history of DDPG training process is shown in Figure 4. The reward had a significant increase during the training process, which reflects a learning process of DDPG.

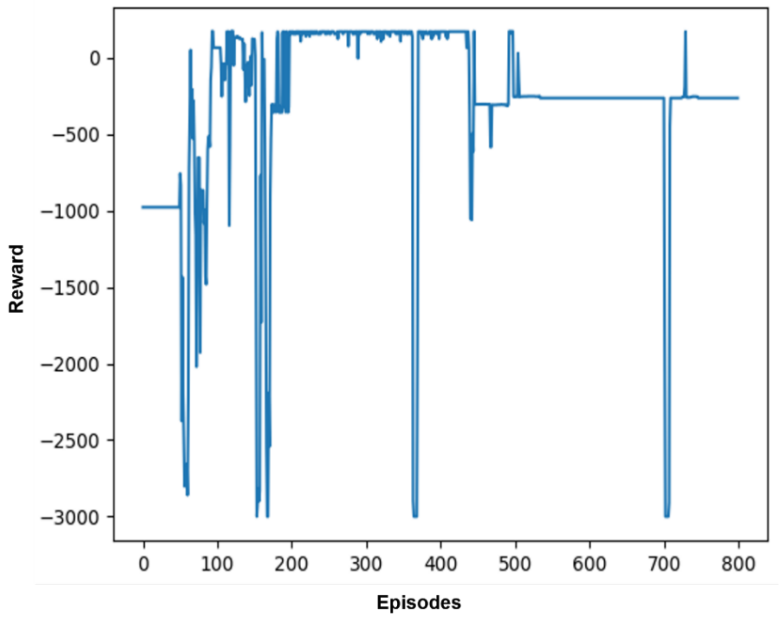

Figure 4-The reward history of DDPG

5. Discussions

5.1 The extending application of the DDPG

Chapter 4 provided the design results of the new design method based on DDPG and verified the effectiveness of the new approach. In the design process, the new method was mainly used as an optimization method. Reinforcement learning method is reproducibility. After training, the agent should gain the ability to give a new design result in a short time. The whole design process of axial compressor contains the 1-D mean-line design, 2-D throughflow design, blade modeling, and 3-D predicting process. The mean-line design provides the flow path and few virtual flow parameters which are the basis of the whole design process. Designer preliminary gives these parameters depends on the experiences, following with a deterministic process usually based on the optimization method. As the design process is iterative, it is necessary to obtain different design results which met the design requirements quickly. In the traditional optimization process, the samples will be useless once the optimal solution is found. Different from the traditional method, the DDPG can gain the experience from the optimization process and give new design results in few steps after training, which provides different results for the designer. Thus, this chapter verified the redesign ability of the new design method through the mean-line design process of a practical 15-stage axial compressor.

5.2 Design variables and requirements.

Pressure ratio: 14.82

Mass Flow: 356.3kg/s

Atmospheric pressure: 101325Pa

Total temperature: 288.15K

Rotation speed: 3000rpm

Stage number: 15 stages

Target efficiency: 0.87

To simplify the training process, the surge margin calculation will not be considered. Thus, the design variables reduced to 5 (DK, CA1, CA2, ALF, HZ1) and other variables are defined as certain values. The definitions and ranges or values of the design variables are shown in Table 3.

Table 3-The definitions and ranges/values of the design variables

Definition	Design Variable	Ranges/Values
Hub/tip ratio	**DK**	**0.40-0.85**
Inlet air velocity	**CA1(m/s)**	**130-240**
Middle air velocity	**CA2(m/s)**	**120-240**
Outlet air velocity	CA3(m/s)	130
Inlet airflow angle	**ALF**	**40-75**
Reaction	TAO	0.51
Tip load factor	**HZ1**	**0.15-0.35**
Root load factor	HZZ	0.20
Tip pitch ratio	HR1	2.3
Root pitch ratio	HRZ	1.4
Minimum chord length	HORDA	0.060

The evaluation process consists of thermodynamics calculation and flow path generating to get the efficiency and flow path. Thus, the DDPG method settings are shown in the following.

5.3 The results of the redesign process.

The neural network algorithm was Adam. The learning rate is set to 0.001 and 0.002. The replay buffer size and the minibatch were 10000 and 32, respectively. The reward discount is set to 0.95. The reward for efficiency is shown in Eq 17.

$$reward_\eta = \begin{cases} 0 & \eta < 0.875 \\ 860\eta - 735 & 0.875 < \eta \leq 0.92 \\ -50 & \eta > 0.92 \end{cases} \tag{17}$$

The basic algorithm is the same as Algorithm 1. The max episodes number is 400, and the max steps number is 400. The design goal of efficiency is set to 0.874. In each episode, once the design goal is satisfied, the design variables will be randomly initialized and turn to the next episode.

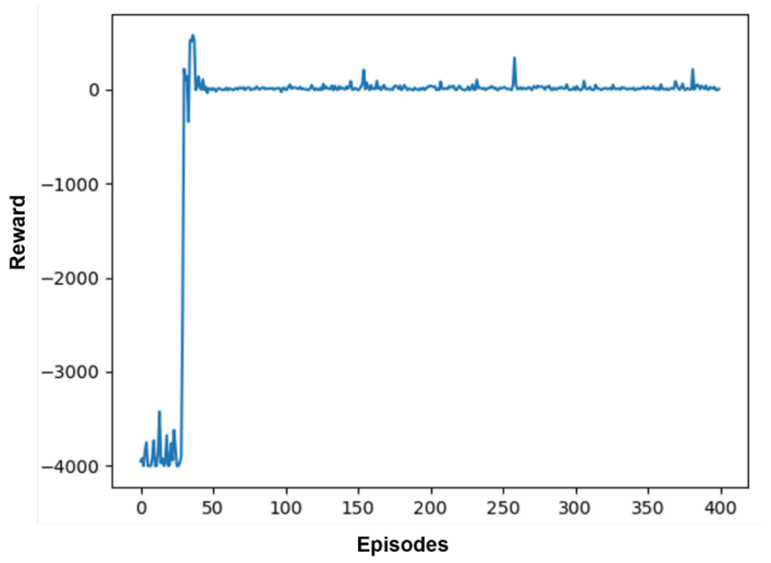

Figure 5-Reward history of the training process

Fig. 5 shows the reward history of the training process. At the start, the reward is negative and cannot reach the design goal in each episode. At episode 100, the reward turns to positive and stable, which means the effectiveness of the learning process. In each process, the agent reached the design goal in few steps. Fig 6 shows the redesign history of the trained DDPG. To verify the design result, the Differential Evolution (DE) optimization method was introduced to solve the same problem. The maximum predicting number was 16000 (the same as the DDPG max step number). Fig. 7 shows the optimization history of DE, each generation contains 100 steps.

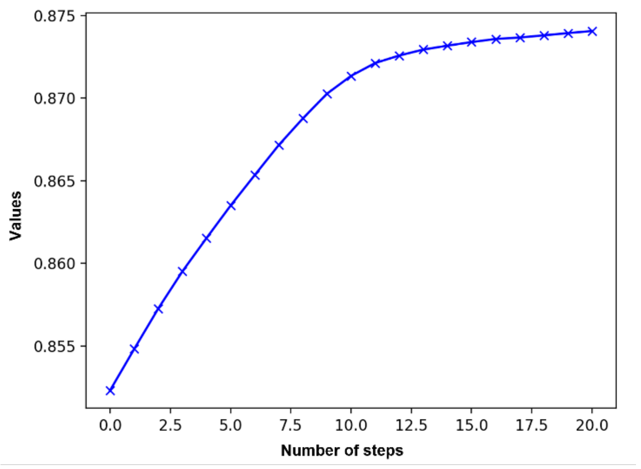

Figure 6-Redesign history of the trained DDPG

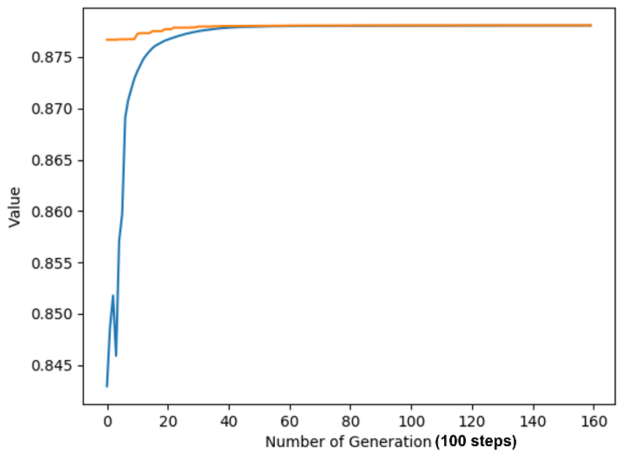

Figure 7-The optimization history of DE

The results of the DDPG and DE are shown in Table 4. The max efficiency of DE optimization results is 87.8%, which is separately 0.4% and 0.2% higher than the two redesign results DDPG1 and DDPG2. The DDPG 1 and DDPG 2 separately reached the design goal in 20 steps and 2 steps, which saved considerable time compared to the DE method with slightly low efficiency. Furthermore, the design results of DDPG 1 and DDPG 2 were obviously different, which can give the designer different references.

Table 4-The optimization results of DDPG and traditional methods

Run	Baseline	DDPG 1(trained)	DDPG 2(trained)	DE
DK	0.4315	0.635	0.6236	0.70
CA1 (m/s)	218.3	165.7	188.6	184
CA2 (m/s)	169	168.7	194.8	192
ALF	63.45	60.2	43.0	69.05
HZ1	0.317	0.264	0.238	0.232

KPD	85.6%	87.4%	87.6%	87.8%
Number of Steps	-	20	2	16000

6. Conclusions

Reinforcement learning can obtain experience in the continuous interaction with the environment to guide future actions. This paper discussed the feasibility of the reinforcement learning method for the aerodynamic design of axial compressors. The DDPG algorithm was used to optimize and accelerate the mean-line aerodynamic design process of axial compressors. After training, the adiabatic efficiency is improved by 2.0% with 23% surge margin, which verified the effectiveness of the design method. This approach obtained experiences from continuous interaction with the predicting program and gained redesign ability to provide different design references for the designer in few steps. The Differential Evolution (DE) optimization method was introduced to solve the same problem to verify the effectiveness of the redesign results. After training, the redesigned efficiency was 0.2% lower than the DE optimization results. The total number of steps of DE optimized and DDPG redesigned were separately 16000 and 20, which verified the quick redesign ability of the new design method.

References

[1] Kutler P,Mehta U. Computational aerodynamics and artificial intelligence[J]. *Mathematics & Its pplication*, 2013, 5(2):131-151.

[2] AIAA. Design of axial compressor airfoils with artificial neural networks and genetic algorithms[J]. *Aiaa Journal*, 2000.

[3] Ghorbanian K, Gholamrezaei M. An artificial neural network approach to compressor performance prediction[J]. *Applied Energy*, 2009, 86(7):1210-1221.

[4] Tompson J, Schlachter K, Sprechmann P, et al. Accelerating eulerian fluid simulation with convolutional networks[C]. *International Conference on Machine Learning*. PMLR, 2017: 3424-3433.

[5] Rawlins T, Lewis A, Hettenhausen J, et al. Enhancing MOPSO through the guidance of ANNs[C]*International Joint Conference on Neural Networks. IEEE*, 2014:4003-4010.

[6] Sutton, Richard S, Barto, et al. Introduction to Reinforcement Learning[J]. *IEEE Transactions on Neural Networks*, 2005, 16(1):285-286.

[7] Goel S, Hajela P . Turbine aerodynamic design using reinforcement learning based optimization[C]. *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* 1998.

[8] Mumith J A, Karayiannis T, Makatsoris C. Design and optimization of a thermoacoustic heat engine using reinforcement learning[J]. *International Journal of Low-Carbon Technologies*, 2016, 11(3):ctv023.

Copyright Statement