Supplementary Materials
Physics Driven Correction for Inverse Estimation

Anonymous Author(s)
Affiliation
Address
email

A Studied Inverse Problems

A.1 Problem Description

Problem 1, Turbofan Design: Turbofan is one of the most complex gas aero engine systems, and it is the dominant propulsion system favoured by commercial airliners [1]. The inverse problem for turbofan design is to find a group of design parameters modelled as the state to achieve the desired performance modelled as observation. The observation includes two performance parameters, including the thrust y_t and the thrust specific fuel consumption y_f. The state includes 11 design parameters that control the performance of the engine, including the bypass ratio r_{bp}, the fan pressure ratio π_{fan}, the fan efficiency η_{fan}, the low-pressure compressor pressure ratio π_{LC}, the low-pressure compressor efficiency η_{LC}, the high-pressure compressor pressure ratio π_{HC}, the high-pressure compressor efficiency η_{HC}, the combustor efficiency η_{LC}, the combustion temperature in the burner T_{B}, the efficiency of high-pressure turbine η_{HT}, and the efficiency of low-pressure turbine η_{LT}. Following the same setting as in [2], the goal is to estimate the design parameters that can achieve the performance of a CFM-56 turbofan engine, for which the thrust should be 121 KN and the thrust specific fuel consumption should be 10.63 g/(kN.s) [3]. This corresponds to the observation vector $\mathbf{y} = [y_t, y_f] = [121, 10.63]$. The 100 experiment cases tested on this problem differ from the state to correct, which is randomly sampled from the feasible region of the design parameter space provided by [2]. Table 1 reports the allowed range of each design parameter, which all together define the feasible region.

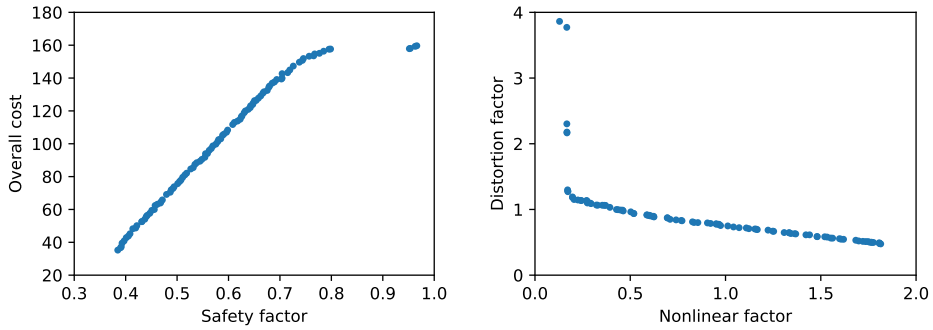
Table 1: Feasible region of the design parameter space for problem 1.

Range	r_{bp}	π_{fan}	π_{LC}	π_{HC}	T_{B}	η_{fan}	η_{HC}	η_{LC}	η_B	η_{HT}	η_{LT}
Min	5	1.3	1.2	8	1300 K	0.85	0.82	0.84	0.95	0.86	0.87
Max	6	2.5	2	15	1800 K	0.95	0.92	0.94	0.995	0.96	0.97

Problem 2, Electro-mechanical Actuator Design: An electro-mechanical actuator is a device that converts electrical energy into mechanical energy [4], by using a combination of an electric motor and mechanical components to convert an electrical signal into a mechanical movement. It is commonly used in industrial automation[4], medical devices[5], and aircraft control systems [6], etc. We consider the design of an electro-mechanical actuator with a three-stage spur gears. Its corresponding inverse problem is to find 20 design parameters modelled as the state, according to the requirements for the overall cost y_c and safety factor y_s modelled as the observation. The 100 experiment cases tested on this problem differ from the observation $\mathbf{y} = [y_c, y_s]$. We have randomly selected 100 combinations of the safety factor and overall cost from the known Pareto front [7], which is shown in Fig. 1a. For each observation, the state to correct is obtained by using an untrained ML model to provide a naturally failed design.

Problem 3, Pulse-width Modulation of 13-level Inverters: Pulse-width modulation (PWM) of n-level inverters is a technique for controlling the output voltage of an inverter that converts DC

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.



(a) (b)

Figure 1: Illustration of the 100 used test cases in the 2-dimensional observation space for problem 2 (subfigure a) and 3 (subfigure b).

power into AC power [8]. It modulates the duty cycle of the output waveform of the inverter, thereby affecting the effective value of the voltage applied to a load. Particularly, an PWM of 13-level inverter adjusts its output waveform using 13 power switching devices to achieve higher precision, which is widely used in renewable power generation systems [9], electric vehicles [10], and industrial automation equipment [11]. It results in a typical inverse problem of finding the suitable control parameters including 30 switch angles modelled as the state, according to the requirements of the distortion factor y_d (which measures the harmonic distortion of the output waveform) and the nonlinear factor y_n (which avoids the malfunctioning of the inverter and the connected load) modelled as the observation. As in problem 2, the 100 experiment cases tested on this problem also differ from the observation, i.e. $\mathbf{y} = [y_d, y_n]$. They correspond to 100 randomly selected combinations of the distortion and nonlinear factors from the known Pareto front in [12], which are shown in Fig. 1b. For each observation, the state to correct is obtained by using an untrained ML model to provide a naturally failed estimation.

A.2 Physical Evaluation

We describe in this section how the physical evaluations are conducted, more specifically, how the physical errors are assessed. Overall, it includes the *observation reconstruction error*, which is based on the difference between the given observation and the reconstructed observation from the estimated state. For different problems, different physical models are used to simulate the reconstruction. It also includes the *feasible domain error*, which examines whether the estimated state is within a feasible region of the state space, and this region is often known for a given engineering problem. Apart from these, there are also other problem-specific errors.

A.2.1 Problem 1

Observation Reconstruction Error: The gas turbine forward model [2] is used to simulate the performance of the turbofan engine. It is constructed through the aerodynamic and thermodynamic modelling of the components in a turbofan engine, where the modelled components include the inlet, fan, low-pressure and high-pressure compressors, combustor, high-pressure and low-pressure turbines, core and fan nozzles, as well as through considering the energy losses. This model can transform the input of state into physically reasonable output of observation, which is the thrust y_t and fuel flow y_f of the engine. Let $F(\mathbf{x})$ denote a forward model. In problem 1, the performance requirement is specifically $\mathbf{y} = [y_t, y_f] = [121, 10.63]$, thus, for a estimated state $\hat{\mathbf{x}}$, the reconstruction error is

$$e_r(\hat{\mathbf{x}}) = \sum_{i=1}^{2} \frac{||F_i(\hat{\mathbf{x}}) - \mathbf{y}_i||_1}{2\mathbf{y}_i}, \tag{1}$$

where, when i respectively equals to 1 or 2, $F_1(\hat{\mathbf{x}})$ and $F_2(\hat{\mathbf{x}})$ are the estimated thrust and fuel consumption in the engine case, respectively. Because the magnitude of the thrust and fuel consumption are different, we use the relative error to measure the reconstruction error of the two observation elements.

Feasible Domain Error: In aero engine design, the design parameters cannot exceed their feasible region and such a region has already been identified by existing work [2] as in Table 1. For the i-th dimension of an estimated state $\hat{x}_i$ (an estimated design parameter), and given its maximum and minimum allowed values $x_{\max}$ and $x_{\min}$, we define its feasible domain error by

$$e_i^{(f)} = \max\left(\frac{\hat{\mathbf{x}}_i - x_{\min}}{x_{\max} - x_{\min}} - 1, 0\right) + \max\left(-\frac{\hat{\mathbf{x}}_i - x_{\min}}{x_{\max} - x_{\min}}, 0\right). \tag{2}$$

After normalization, all the feasible values are within the range of $[0, 1]$, while the non-feasible ones outside. The above error simply examines how much the normalized state value exceeds 1 or below 0. We compute an accumulated feasible error for all the 11 design parameters, given by $e_f(\hat{\mathbf{x}}) = \frac{1}{11}\sum_{i=1}^{11} e_i^{(f)}$.

Design Balance Error: Another desired property by aero engine design is a low disparity among the values of the design parameters after normalizing them by their feasible ranges, which indicates a more balanced design, offering better cooperation between the design components and resulting in lower cost [2, 1]. Standard deviation is a suitable measure to assess this balance property, resulting in another physical error

$$e_\sigma(\hat{\mathbf{x}}) = \sigma\left(\left\{\frac{\hat{\mathbf{x}}_i - x_{\min}}{x_{\max} - x_{\min}}\right\}_{i=1}^{11}\right). \tag{3}$$

where $\sigma(\cdot)$ denotes the standard deviation of the elements from its input set.

Accumulated Physical Error: The above three types of errors are combined to form the following accumulated physical error:

$$\hat{e}(\hat{\mathbf{x}}) = e_r(\hat{\mathbf{x}}) + 0.1e_f(\hat{\mathbf{x}}) + 0.1e_\sigma(\hat{\mathbf{x}}). \tag{4}$$

The weights are given as 1, 0.1 and 0.1, respectively. This is because the reconstruction error determines whether the estimated state is feasible, while the other two errors are used to further improve the quality of the estimated state from the perspective of the design preference. Here $e_r(\hat{\mathbf{x}})$ is obtained using a forward simulation process thus is an implicit error, while $\mathbf{e}_f$ and e_σ have analytical expressions and simple gradient forms, and thus are explicit errors.

A.2.2 Problem 2

Observation Reconstruction Error: The used forward model for electro-mechanical actuator design is a performance simulation model, considering a stepper motor, three stages of spur gears and a housing to hold the components (i.e., stepper motor, and three stages of spur gears) [7]. It consists of a physical model that predicts its output speed and torque and component-specific constraints, a cost model and a geometric model that creates 3-D meshes for the components and the assembled system. The integrated model predicts the observation $\mathbf{y} = \{y_c, y_s\}$, and is named as the "CS1" model in [7]. After reconstructing by CS1 the safety factor y_s and total cost y_c from the estimated design parameters $\hat{\mathbf{x}}$, the reconstruction error is computed using Eq. 1.

Feasible Domain Error: The same feasible domain error $\mathbf{e}_f$ as in Eq. (2) is used for each design parameter of problem 2. The only difference is that the allowed parameter ranges for defining the feasible region have changed. We use the region identified by [7]. There are 20 design parameters, thus e_f is an average of 20 individual errors.

Inequality Constraint Error: We adopt another seven inequality constraints provided by the forward model [7] to examine how reasonable the estimated design parameters are. These constraints do not have analytical forms, and we express them as $c_i(\hat{\mathbf{x}}) \leq 0$ for $i = 1, 2, , \ldots, 7$. Based on these, we define the following inequality constraint error

$$e_c(\hat{\mathbf{x}}) = \frac{1}{7}\sum_{i=1}^{7} \max(c_i(\hat{\mathbf{x}}), 0). \tag{5}$$

Accumulated Physical Error: We then combine the above three types of errors, given as

$$\hat{e}(\hat{\mathbf{x}}) = e_r(\hat{\mathbf{x}}) + 0.1e_f(\hat{\mathbf{x}}) + e_c(\hat{\mathbf{x}}), \tag{6}$$

where both $e_r(\hat{\mathbf{x}})$ and $e_c(\hat{\mathbf{x}})$ are implicit errors computed using a black-box simulation model, while $\mathbf{e}_f$ is an explicit error. In this case, we increase the weight for inequality constraint error to be the

same as the reconstruction error, this is because we regard the implicit errors irrespective of their types as the same. Of course, one can also use different weights for different types of errors according to their expertise.

A.2.3 Problem 3

We use the forward model from [12] to reconstruct the observation for the 13-level inverter. It takes the control parameters as the input and returns the distortion factor y_d and the nonlinear factor y_n. Based the reconstructed y_d and y_n, the observation reconstruction error e_r is computed by Eq. (1) in the same way as in problems 1 and 2. Similarly, the same feasible boundary error e_f as in Eq. (2) is computed, but the feasible region is different where the range of $[0, \frac{\pi}{2}]$ is applied for all the 30 control parameters, which is defined in [12]. A similar inequality constraint error as in Eq. (5) is used, which contains 29 inequality constraints in the form of

$$c_i(\hat{\mathbf{x}}) = \hat{\mathbf{x}}_i - \hat{\mathbf{x}}_{i+1} < 0, \text{for } i = 1, 2, \dots 29. \tag{7}$$

Finally, the accumulated physical error is given by

$$\hat{e}(\hat{\mathbf{x}}) = e_r(\hat{\mathbf{x}}) + 0.1e_f(\hat{\mathbf{x}}) + 10e_c(\hat{\mathbf{x}}), \tag{8}$$

where a large weight is used for $e_c(\hat{\mathbf{x}})$ because the inequality constraints that it involves are very critical for the design. Among the three types of errors, $e_r(\hat{\mathbf{x}})$ is an implicit error, while $e_f(\hat{\mathbf{x}})$ and $e_c(\hat{\mathbf{x}})$ are explicit errors.

B Extra Implementation Information

In this section, we introduce extra implementation information for GEESE and the compared methods, in addition to what has been mentioned in the main text. In GEESE implementation, the latent vector $\mathbf{z}$ has the same dimension as the state $\mathbf{x}$ in problems 1 and 2, because the optimization is done directly on the latent vectors. In problem 3, the dimension of $\mathbf{z}$ is set be 1, and transformed into a 30-dimensional vector $\mathbf{x}$ by the state generator. The number of the latent vector $\mathbf{z}$ used for sampling distribution of generators is set increasingly as $d = 64, 128, 256$ for problems 1, 2, and 3, due to the increasing dimension of the state space of the three problems. Although, the budget query number equals to 1000, because GEESE may query two times per iteration, thus, the maximum iteration number is smaller than 1000, which is determined when the budget is used up.

For BOGP, its Bayesian optimization is implemented using the package [13]. The prior is set to be a Gaussian process, and its kernel is set as Matern 5/2. The acquisition function is set to be the upper confidence bound (UCB). The parameter kappa, which indicates how closed the next parameters are sampled, is set to be 2.5. The other hyperparameters are kept as default. Since Bayesian optimization only queries one state-error pair in each iteration, its maximum iteration number is equal to the maximum number of queries, i.e., 1000.

The other methods of GA, PSO, CMAES, ISRES, NSGA2, and UNSGA3 are implemented using the package pymoo [14]. For ISRES, we apply a $1/7$ success rule to generate seven times more candidates than that in the current population in order to perform a sufficient search. The other parameters are kept as default. Since these algorithms need to query the whole population in each iteration, their maximum iteration number is thus much smaller than the query budget 1000. In the experiments, these algorithms are terminated when the maximum query number 1000 is reached.

To implement SVPEN [2], we use the default setting for problem 1. As for problems 2 and 3, to construct the state estimator and the error estimator for SVPEN, the same structures of the base neural networks and the exploitation generator as used by GEESE are adopted, respectively. Also the same learning rate as used by GEESE is used for SVPEN, while the other settings are kept as default for problems 2 and 3. In each iteration, SVPEN queries three times the physical errors for simulating the exploitation, as well as the regional and global exploration. Thus, the maximum iteration number of SVPEN is set as 333 to match the query budget 1000.

All the methods are activated or initialized using the same set of N state-error pairs randomly sampled from a predefined feasible region in the state space. For GEESE and SVPEN, these samples are used to train their surrogate error models, i.e., the base neural networks in GEESE and the error estimator in SVPEN, thus their batch size for training is also set as N. For Bayesian optimization,

Table 2: Performance comparison for two different values of feasibility threshold ϵ, where the best is shown in **bold** while the second best is underlined for query times.

Threshold	Algorithm	Problem 1 State Dimension:11		Problem 2 State Dimension:20		Problem 3 State Dimension:30	
		Failure times	Query times	Failure times	Query times	Failure times	Query times
$\epsilon = 0.1$	BOGP	0	3.04 ±0.83	78	849.26 ±295.35	3	86 ±200.49
	GA	0	64 ±0	0	65.92 ±10.92	8	183.04 ±287.80
	PSO	0	64 ±0	0	64.00 ±0	8	199.92 ±284.94
	CMAES	0	12 ±0	0	73.84 ±25.81	3	127.29 ±233.71
	ISRES	0	65 ±0	0	108.52 ±41.36	10	203.30 ±297.13
	NSGA2	0	64 ±0	0	70.40 ±19.20	8	189.04 ±293.60
	UNSGA3	0	64 ±0	0	68.48 ±16.33	7	177.52 ±275.84
	SVPEN	82	932.51 ±176.38	100	1000 ±0	100	1000 ±0
	GEESE (ours)	0	**2.34 ±17.99**	0	**23.13 ±17.99**	0	**35.58 ±63.82**
$\epsilon = 0.05$	BOGP	0	**9.24 ±3.97**	100	1000 ±0	16	227.63 ±364.08
	GA	0	64.00 ±0	0	353.28 ±105.74	20	297.92 ±363.45
	PSO	0	64.00 ±0	1	**157.84 ±137.40**	18	290.96 ±373.65
	CMAES	0	77.56 ±4.38	1	302.59 ±156.24	22	344.54 ±363.18
	ISRES	0	193.00 ±0	3	391.54 ±241.22	19	313.69 ±368.78
	NSGA2	0	64.00 ±0	0	352.00 ±114.31	20	299.84 364.63
	UNSGA3	0	64.00 ±0	0	368.64 ±102.85	20	310.72 ±370.24
	SVPEN	100	1000 ±0	100	1000 ±0	100	1000 ±0
	GEESE (Ours)	0	20.20 ±16.37	0	189.90 ±164.96	2	**81.26 ±155.30**

these samples are used to construct the Gaussian process prior. For GA, PSO, ISRES, NSGA2, and UNSGA3, these samples are used as the initial population to start the search. The only special case is CMAES, as it does not need a set of samples to start the algorithm, but one sample. So we randomly select one state-error pair from the N pairs to activate its search.

For problem 3, we post-process the output of all the compared methods, in order to accommodate the element-wise inequality constraints in Eq. (7), by

$$\hat{\mathbf{x}}_i^{(p)} = \hat{\mathbf{x}}_1^{(p)} + \frac{1}{1 + e^{-\sum_{j=1}^{i} \hat{\mathbf{x}}_j^{(p)}}} \left(1 - \hat{\mathbf{x}}_1^{(p)}\right). \tag{9}$$

As a result, the magnitude of the element in $\hat{\mathbf{x}}^{(p)}$ is monotonically increasing, and the inequality constraints are naturally satisfied. But this can complicate the state search, as the elements are no longer independent. A balance between correlating the state elements and minimizing the accumulated physical error is needed. But in general, we have observed empirically that the above post-processing can accelerate the convergence for all the compared methods. One way to explain the effectiveness of this post-processing is that it forces the inequality constraints to hold, and this is hard for the optimization algorithms to achieve on their own.

C Extra Results

Varying Feasibility Threshold: In addition to the feasibility threshold of $\epsilon = 0.075$ as studied in the main text, we test two other threshold values, including $\epsilon = 0.05$ representing a more challenging case with lower error tolerance, and $\epsilon = 0.1$ representing a comparatively easier case with higher error tolerance. Results are reported in Table 2. In both cases, GEESE has failed the least times among all the compared methods and for all three problems studied. It is worth to mention that, in most cases, GEESE has achieved zero failure, and a very small $N_{\text{failure}} = 2$ out of 100 in only one experiment when all the other methods have failed more than fifteen times. Also, this one experiment is the most challenging, solving the most complex problem 3 with the highest state dimension $d = 30$ and having the lowest error tolerance $\epsilon = 0.05$. In terms of query times, GEESE has always ranked among the top 2 most efficient methods for all the tested cases and problems, while the ranking of the other methods vary quite a lot. For instance, when $\epsilon = 0.05$, BOGP performs the best for the easiest problem 1, but it performs the worst for the more difficult problem 2 where it has failed to find a feasible solution within the allowed query budget. In the most difficult experiment that studies problem 3 under $\epsilon = 0.05$, GEESE requires much less query times and is significantly more efficient than the second most efficient method.

Varying Initial Sample Size: In addition to the studied initial sample size $N = 64$ in the main text, we further compare to more cases of $N = 16$ and $N = 32$ under $\epsilon = 0.05$. The results are shown in Table 3. Still, GEESE has the least failure times in all experiments, which is important

Table 3: Performance comparison under for two different sizes of initial samples, where the best is shown in **bold** while the second best is underlined for query times.

Initial Size	Algorithm	Problem 1 State Dimension:11		Problem 2 State Dimension:20		Problem 3 State Dimension:30	
		Failure times	Query times	Failure times	Query times	Failure times	Query times
$N = 32$	BOGP	0	**9.60 ±3.89**	100	1000 ±0	15	239.12 ±367.12
	GA	0	32.00 ±0	0	241.60 ±71.75	21	270.80 ±382.51
	PSO	0	32.00 ±0	18	311.20 ±333.45	14	283.28 ±324.54
	CMAES	0	77.56 ±4.38	1	321.01 ±188.6	22	280.54 ±363.18
	ISRES	0	64.00 ±0	3	416.24 ±209.23	21	276.24 ±386.75
	NSGA2	0	32.00 ±0	1	239.44 ±150.26	22	262.88 ±394.99
	UNSGA3	0	32.00 ±0	2	**218.72 ±136.53**	22	260.64 ±396.51
	SVPEN	100	1000 ±0	100	1000 ±0	100	1000 ±0
	GEESE (Ours)	0	33.63 ±19.35	0	233.96 ±180.01	10	**167.77 ±284.31**
$N = 16$	BOGP	0	**10.62 ±5.53**	100	1000 ±0	17	249.88 ±372.99
	GA	0	16.00 ±0	43	657.04 ±352.42	23	364.40 ±373.30
	PSO	0	32.00 ±0	10	293.76 ±271.02	21	247.76 ±392.87
	CMAES	0	77.56 ±4.38	1	333.49 ±156.24	17	320.07 ±350.84
	ISRES	0	17.00 ±0	2	260.50 ±189.71	20	243.20 ±392.70
	NSGA2	0	32.00 ±0	33	590.96 ±355.93	25	377.20 ±385.78
	UNSGA3	0	32.00 ±0	28	487.04 ±360.22	28	408.80 ±397.77
	SVPEN	100	1000 ±0	100	1000 ±0	100	1000 ±0
	GEESE(Ours)	0	36.72 ±22.52	0	**248.26 ±176.64**	9	**163.26 ±279.34**

in remediating failed ML estimations. In terms of query times, GEESE still ranks among the top 2 most efficient methods for the two more complex problems 2 and 3, being the top 1 with significantly less query times for the most complex problem 3. However, GEESE does not show advantage in the simplest problem 1 with the lowest state dimension. It performs similarly to those top 2 methods under $N = 32$, e.g. 34 vs. 32 query times, while performs averagely when the initial sample size drops to $N = 16$. This is in a way not surprising, because BOGP, GA, PSO, NSGA2 and UNSGA3 can easily explore the error distribution of low state dimensions. BOGP uses Gaussian process to construct accurate distribution of errors, while GA, PSO, NSGA2, and UNSGA3 sample sufficient samples in each iteration to sense the distribution of error in each iteration, and there is a high chance for them to find a good candidate in early iterations when the search space has a low dimension. However, the valuable samples are sparsely distributed into the higher dimensional space, and it is challenging for them to explore the error distribution and find the feasible states in the early iterations.

D GEESE Sensitivity Analysis

We conduct extra experiments to assess the hyperparameter sensitivity of GEESE using problem 1 under $\epsilon = 0.05$. The studied hyperparameters include the number L of the base neural networks, the number N_{IT} of the candidate states generated for exploitation, the learning rate for training the exploitation generator η_{IT}, and the early stopping threshold ϵ_e for training the base neural networks. The results are reported in Table 4. It can be seen from the table that, although the performance varies versus different settings, the change is mild within an acceptable range. This makes it convenient to tune the hyperparameters for GEESE.

Below we further discuss separately the effects of different hyperparameters and analyze the reasons behind: (1) We experiment with three base network numbers $L = 2, 4, 8$. It can be seen from Table 4 that there is a performance improvement as L increases in terms of the required query times, but this is on the expense of consuming higher training cost. Thus, we choose the more balanced setting $L = 4$ as the default in our main experiments. (2) We test different candidate state numbers $N_{\text{IT}} = 1, 32, 64, 128$ used for exploitation. Results show a performance increase followed by a decrease as N_{IT} increases. Using a candidate set containing one single state is insufficient, while allowing a set with too many candidate states can also harm the efficiency. This can be caused by the approximation gap between the surrogate error model and the true physical evaluation. In our main experiments, we go with the setting of 64 for problem 1 as we mentioned in Appendix B, because it provides a proper balance between the exploitation performance and the overfitting risk. (3) We also examine different settings of the learning rate for training the exploitation generator, i.e., $\eta_{\text{IT}} = 1e^{-1}, 1e^{-2}, 1e^{-3}$. Similarly, there is a performance increase first but followed by a decrease, as in changing N_{IT}. A larger learning rate can accelerate the learning of the exploitation generator and subsequently enable a potentially faster search of the feasible state. But an overly high learning

rate can also cause fluctuation around the local optimum, and this then consumes more query times. Although a smaller learning rate can enable a more guaranteed convergence to the local optimum, it requires more iterations, thus more query times. (4) We experiment with three values of early stopping threshold, i.e., $\epsilon_e = 1e^{-3}, 1e^{-4}, 1e^{-5}$. It can be seen from Table 4 that a decreasing ϵ_e can first improve the efficiency but then reduce it, however without changing much the standard deviation. An inappropriate setting of the early stopping threshold can lead to base neural networks overfitting (or underfitting) to the actual data distribution, thus harm the performance.

Table 4: Results of sensitivity Analysis, where a better performance is highlighted in **bold**.

(1): Effect of Base Network Number

Base Network Number	**Query times**
$L = 2$	20.20 ±16.37
$L = 4$	15.44 ±13.86
$L = 8$	**15.09 ±13.01**

(2): Effect of Latent Vector Number

Latent vector number	**Query times**
$N_{\text{IT}} = 1$	72.56 ±36.13
$N_{\text{IT}} = 32$	28.21 ±17.05
$N_{\text{IT}} = 64$	**20.20 ±16.37**
$N_{\text{IT}} = 128$	26.95 ±14.12

(3): Effect of Learning rate for Exploration Generator

Learning Rate	**Query times**
$\eta_{\text{IT}} = 1e^{-1}$	27.56 ±9.28
$\eta_{\text{IT}} = 1e^{-2}$	**20.20±16.37**
$\eta_{\text{IT}} = 1e^{-3}$	64.36 ±44.50

(4): Effect of Early Stopping Threshold

Early stopping threshold	**Query times**
$\epsilon_e = 1e^{-3}$	37.55±17.28
$\epsilon_e = 1e^{-4}$	**20.20 ±16.37**
$\epsilon_e = 1e^{-5}$	26.20 ±15.45

References

[1] Joachim Kurzke and Ian Halliwell. *Propulsion and Power: An Exploration of Gas Turbine Performance Modeling*. Springer, 2018.

[2] Ruiyuan Kang, Dimitrios C. Kyritsis, and Panos Liatsis. Self-Validated Physics-Embedding Network: A General Framework for Inverse Modelling, October 2022.

[3] Hakan Aydin, Onder Turan, T. Hikmet Karakoc, and Adnan Midilli. Exergetic Sustainability Indicators as a Tool in Commercial Aircraft: A Case Study for a Turbofan Engine. *International Journal of Green Energy*, 12(1):28–40, 2015.

[4] Johan Åkerberg, Mikael Gidlund, and Mats Björkman. Future research challenges in wireless sensor and actuator networks targeting industrial automation. In *2011 9th IEEE International Conference on Industrial Informatics*, pages 410–415. IEEE, 2011.

[5] Patrick R Buckley, Gareth H McKinley, Thomas S Wilson, Ward Small, William J Benett, Jane P Bearinger, Michael W McElfresh, and Duncan J Maitland. Inductively heated shape memory polymer for the magnetic actuation of medical devices. *IEEE transactions on biomedical engineering*, 53(10):2075–2083, 2006.

[6] Xidong Tang, Gang Tao, and Suresh M Joshi. Adaptive actuator failure compensation for nonlinear mimo systems with an aircraft control application. *Automatica*, 43(11):1869–1883, 2007.

[7] Cyril Picard and Jurg Schiffmann. Realistic Constrained Multiobjective Optimization Benchmark Problems From Design. *IEEE Transactions on Evolutionary Computation*, 25(2):234–246, April 2021.

[8] Amarendra Edpuganti and Akshay Kumar Rathore. Optimal Pulsewidth Modulation for Common-Mode Voltage Elimination Scheme of Medium-Voltage Modular Multilevel Converter-Fed Open-End Stator Winding Induction Motor Drives. *IEEE Transactions on Industrial Electronics*, 64(1):848–856, January 2017.

[9] G Vasuki, M Vinothini, SB Rushmithaa, K Karthik Kumar, AS Kamaraja, and M Willjuice Iruthayarajan. 13-level inverter configuration with a reduced auxiliary circuit for renewable energy applications. In *2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 101–105. IEEE, 2021.

[10] Ashish Kumar, Ram Kumar, Abhishek Kumar, and PR Thakura. Design and development of 13-level multilevel inverter for hybrid electric vehicles. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 121–126. IEEE, 2017.

[11] Yuanmao Ye, Shikai Chen, Xiaolin Wang, and Ka-Wai Eric Cheng. Self-balanced 13-level inverter based on switched capacitor and hybrid pwm algorithm. *IEEE Transactions on Industrial Electronics*, 68(6):4827–4837, 2020.

[12] Abhishek Kumar, Guohua Wu, Mostafa Z. Ali, Qizhang Luo, Rammohan Mallipeddi, Ponnuthurai Nagaratnam Suganthan, and Swagatam Das. A Benchmark-Suite of real-World constrained multi-objective optimization problems and some baseline results. *Swarm and Evolutionary Computation*, 67:100961, December 2021.

[13] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014.

[14] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.