## A Additional details for compositional inverse design in time

This section provides additional details for Section 4.1 and Section 4.2. In both sections, we use the same dataset for training, and the model architecture and training specifics are the same for both sections.

**Dataset.** We use two Python packages Pymunk (Blomqvist, 2007) and Pygame (Shinners, 2000) to generate the trajectories for this N-body dataset. We use 4 walls and several bodies to define the simulation environment. The walls are shaped as a $200 \times 200$ rectangle, setting elasticity to 1.0 and friction to 0.0. A body is described as a ball (circle) with a radius of 20, which shares the same elasticity and friction coefficient as the wall it interacts with. The body is placed randomly within the boundaries and its initial velocity is determined using a uniform distribution $v \sim U(-100, 100)$. We performed 2000 simulations, for 2 balls, 4 balls, and 8 balls in each simulation. Each simulation has a time step of 1/60 seconds, consisting of 1000 steps in total. During these simulations, we record the positions and velocities of each particle in two dimensions at each time step to generate 3 datasets with a shape of $[N_s, N_t, N_b, N_f]$. $N_s$ means number of simulations, $N_t$ means number of time steps, $N_b$ is number of bodies, $N_f$ means number of features. The input of one piece of data shaped as $[B, 1, N_b \times N_f]$, $B$ is batch size, for example, $[32, 1, 8]$ for 2 bodies conditioning on only one step. Before training the model, the final data will be normalized by dividing it by 200 and setting the time resolution to four simulation time steps.

**Model structure.** The U-Net (Ronneberger et al., 2015) consists of three modules: the downsampling encoder, the middle module, and the upsampling decoder. The downsampling encoder comprises 4 layers, each including three residual modules and downsampling convolutions. The middle module contains 3 residual modules, while the upsampling decoder includes four layers, each with 3 residual modules and upsampling. We mainly utilize one-dimensional convolutions in each residual module and incorporate attention mechanisms. The input shape of our model is defined as $[batch\_size, n\_steps, n\_features]$, and the output shape follows the same structure. The GNS (Sanchez-Gonzalez et al., 2020) model consists of three main components. First, it builds an undirected graph based on the current state. Then, it encodes nodes and edges on the constructed graph, using message passing to propagate information. Finally, it decodes the predicted acceleration and utilizes semi-implicit Euler integration to update the next state. In our implementation of GNS, each body represents a node with three main attributes: current speed, distance from the wall, and particle type. We employ the standard k-d tree search algorithm to locate adjacent bodies within a connection radius, which is set as 0.2 twice the body radius. The attribute of an edge is the vector distance between the two connected bodies. More details are in Table 4.

**Training.** We utilize the MSE (mean squared error) as the loss function in our training process. Our model is trained for approximately 60 hours on a single Tesla V100 GPU, with a batch size of 32, employing the Adam optimizer for 1 million iterations. For the first 600,000 steps, the learning rate is set to 1e-4. After that, the learning rate is decayed by 0.5 every 40,000 steps for the remaining 400,000 iterations. More details are provided in Table 5.

To perform inverse design, we mainly trained the following models: U-Net, conditioned on 1 step and capable of rolling out 23 steps; U-Net (single step), conditioned on 1 step and limited to rolling out only 1 step; GNS, conditioned on 1 step and able to roll out 23 steps; GNS (single step), conditioned on 1 step and restricted to rolling out only 1 step; and the diffusion model. Simultaneously, we conducted a comparison to assess the efficacy of time compose by training a diffusion model with 44 steps directly for inverse design, eliminating the requirement for time compose. The results and analysis are shown in Appendix C. Throughout the training process, we maintained consistency in the selection of optimizers, datasets, and training steps for these models.

**Inverse design.** The center point is defined as the target point, and our objective is to minimize the mean squared error (MSE) between the position of the trajectory's last step and the target point. To compare our CinDM method, we utilize U-Net and GNS as forward models separately. We then use CEM (Rubinstein & Kroese, 2004) and Backprop (Allen et al., 2022) for inverse design with conditioned state $(x_0, y_0, v_{x0}, v_{y0})$ used as input, and multiple trajectories of different bodies as rolled out. While the CEM algorithm does not require gradient information, we define a parameterized Gaussian distribution and sample several conditions from it to input into the forward model for prediction. After the calculation of loss between the prediction and target, the best-performing samples are selected to update the parameterized Gaussian distribution. Through multiple iterations, we can

Table 4: **Hyperparameters of model architecture for N-body task**.

| Hyperparameter name | 23-steps | 1-step |
|---|---|---|
| Hyperparameters for U-Net architecture: | | |
| Channel Expansion Factor | $(1, 2, 4, 8)$ | $(1, 2, 1, 1)$ |
| Number of downsampling layers | 4 | 4 |
| Number of upsampling layers | 4 | 4 |
| Input channels | 8 | 8 |
| Number of residual blocks for each layer | 3 | 3 |
| Batch size | 32 | 32 |
| Input shape | $[32, 24, 8]$ | $[32, 2, 8]$ |
| Output shape | $[32, 24, 8]$ | $[32, 2, 8]$ |
| Hyperparameters for GNS architecture: | | |
| Input steps | 1 | 1 |
| Prediction steps | 23 | 1 |
| Number of particle types | 1 | 1 |
| Connection radius | 0.2 | 0.2 |
| Maximum number of edges per node | 6 | 6 |
| Number of node features | 8 | 8 |
| Number of edge features | 3 | 3 |
| Message propagation layers | 5 | 5 |
| Latent size | 64 | 64 |
| Output size | 46 | 2 |
| Hyperparameters for the U-Net in our CinDM: | | |
| Diffusion Noise Schedule | cosine | cosine |
| Diffusion Step | 1000 | 1000 |
| Channel Expansion Factor | $(1, 2, 4, 8)$ | $(1, 2, 1, 1)$ |
| Number of downsampling layers | 4 | 4 |
| Number of upsampling layers | 4 | 4 |
| Input channels | 8 | 8 |
| Number of residual blocks for each layer | 3 | 3 |
| Batch size | 32 | 32 |
| Input shape | $[32, 24, 8]$ | $[32, 2, 8]$ |
| Output shape | $[32, 24, 8]$ | $[32, 2, 8]$ |

sample favorable conditions from the optimized distribution to predict trajectories with low loss values. Backpropagation heavily relies on gradient information. It calculates the gradient of the loss concerning the conditions and updates the conditions using gradient descent, ultimately designing conditions that result in promising output.

During training, we can only predict a finite number of time steps based on conditional states, but the system evolves over an infinite number of time steps starting from an initial state in real-world physical processes. To address this, we need to combine time intervals while training a single model capable of predicting longer trajectories despite having a limited number of training steps. For the forward model, whether using U-Net or GNS, we rely on an intermediate time step derived from the last prediction as the condition for the subsequent prediction. We iteratively forecast additional time steps based on a single initial condition in this manner. As for the forward model (single step), we employ an autoregressive approach using the last step of the previous prediction to predict more steps.

Table 5: **Hyperparameters of training for N-body task**.

| Hyperparameter name | 23-steps | 1-step |
|---|---|---|
| Hyperparameters for U-Net training: | | |
| Loss function | MSE | MSE |
| Number of examples for training dataset | $3 \times 10^5$ | $3 \times 10^5$ |
| Total number of training steps | $1 \times 10^6$ | $1 \times 10^6$ |
| Batch size | 32 | 32 |
| Initial learning rate | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| Number of training steps with a fixed learning rate | $6 \times 10^5$ | $6 \times 10^5$ |
| Learning rate adjustment strategy | StepLR | StepLR |
| Optimizer | Adam | Adam |
| Number of steps for saving checkpoint | $1 \times 10^4$ | $1 \times 10^4$ |
| Exponential Moving Average decay rate | 0.95 | 0.95 |
| Hyperparameters for GNS training: | | |
| Loss function | MSE | MSE |
| Number of examples for training dataset | $3 \times 10^5$ | $3 \times 10^5$ |
| Total number of training steps | $1 \times 10^6$ | $1 \times 10^6$ |
| Batch size | 32 | 32 |
| Initial learning rate | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| Number of training steps with a fixed learning rate | $6 \times 10^5$ | $6 \times 10^5$ |
| Learning rate adjustment strategy | StepLR | StepLR |
| Optimizer | Adam | Adam |
| Number of steps for saving checkpoint | $1 \times 10^4$ | $1 \times 10^4$ |
| Exponential Moving Average decay rate | 0.95 | 0.95 |
| Hyperparameters for our CinDM training: | | |
| Loss function | MSE | MSE |
| Number of examples for training dataset | $3 \times 10^5$ | $3 \times 10^5$ |
| Total number of training steps | $1 \times 10^6$ | $1 \times 10^6$ |
| Batch size | 32 | 32 |
| Initial learning rate | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| Number of training steps with a fixed learning rate | $6 \times 10^5$ | $6 \times 10^5$ |
| Learning rate adjustment strategy | StepLR | StepLR |
| Optimizer | Adam | Adam |
| Number of steps for saving checkpoint | $1 \times 10^4$ | $1 \times 10^4$ |
| Exponential Moving Average decay rate | 0.95 | 0.95 |

## B FULL RESULTS FOR COMPOSITIONAL INVERSE DESIGN OF THE N-BODY TASK

Here we provide the full statistical results including the 95% confidence interval (for 500 instances) for N-body experiments, including compostional inverse design in time and more objects. Specifically, Table 6 shows detailed results for Table 1 in Section 4.1; and Table 7 extends Table 2 in Section 4.2.

Table 6: **Compositional Generalization Across Time**. The confidence interval information is provided in addition to Table 1.

| | **2-body 24 steps** | | **2-body 34 steps** | | **2-body 44 steps** | | **2-body 54 steps** | |
|---|---|---|---|---|---|---|---|---|
| Method | design obj | MAE | design obj | MAE | design obj | MAE | design obj | MAE |
| CEM, GNS (1-step) | 0.2622 ± 0.0090 | 0.13963 ± 0.00999 | 0.2204 ± 0.0072 | 0.15378 ± 0.01123 | 0.2701 ± 0.0079 | 0.21277 ± 0.01264 | 0.2773 ± 0.0070 | 0.21706 ± 0.01256 |
| CEM, GNS | 0.2699 ± 0.0081 | 0.12746 ± 0.00662 | 0.3142 ± 0.0064 | 0.14637 ± 0.00657 | 0.3056 ± 0.0060 | 0.18155 ± 0.00689 | 0.3124 ± 0.0062 | 0.20266 ± 0.00679 |
| CEM, U-Net (1-step) | 0.2364 ± 0.0068 | 0.07720 ± 0.00623 | 0.2391 ± 0.0081 | 0.09701 ± 0.00796 | 0.2744 ± 0.0073 | 0.11885 ± 0.00854 | 0.2729 ± 0.0074 | 0.12992 ± 0.00897 |
| CEM, U-Net | 0.1762 ± 0.0071 | 0.03597 ± 0.00395 | 0.1639 ± 0.0062 | 0.03094 ± 0.00342 | 0.1816 ± 0.0072 | 0.03900 ± 0.00451 | 0.1887 ± 0.0075 | 0.04350 ± 0.00487 |
| Backprop, GNS (1-step) | 0.1452 ± 0.0050 | 0.04339 ± 0.00285 | 0.1497 ± 0.0061 | 0.03806 ± 0.00304 | 0.1511 ± 0.0062 | 0.03621 ± 0.00322 | 0.1851 ± 0.0062 | 0.04104 ± 0.00285 |
| Backprop, GNS | 0.2407 ± 0.0067 | 0.09788 ± 0.00615 | 0.2678 ± 0.0072 | 0.11017 ± 0.00620 | 0.2762 ± 0.0071 | 0.12395 ± 0.00657 | 0.2952 ± 0.0073 | 0.13963 ± 0.00623 |
| Backprop, U-Net (1-step) | 0.2182 ± 0.0068 | 0.07554 ± 0.00466 | 0.2445 ± 0.0093 | 0.08278 ± 0.00613 | 0.2536 ± 0.0078 | 0.08487 ± 0.00611 | 0.2751 ± 0.0088 | 0.10599 ± 0.00709 |
| Backprop, U-Net | 0.1228 ± 0.0040 | 0.01974 ± 0.00223 | **0.1171** ± 0.0032 | 0.01236 ± 0.00104 | **0.1143** ± 0.0026 | 0.00970 ± 0.00076 | **0.1289** ± 0.0043 | 0.01067 ± 0.00090 |
| **CinDM (ours)** | **0.1160** ± 0.0019 | **0.01264** ± 0.00057 | 0.1288 ± 0.0030 | **0.00917** ± 0.00070 | 0.1447 ± 0.0040 | **0.00959** ± 0.00116 | 0.1650 ± 0.0045 | **0.01064** ± 0.00117 |

Table 7: **Compositional Generalizaion Across Objects.**. The confidence interval information is provided in addition to Table 2.

| Method | **4-body 24 steps** design obj | **4-body 24 steps** MAE | **4-body 44 steps** design obj | **4-body 44 steps** MAE | **8-body 24 steps** design obj | **8-body 24 steps** MAE | **8-body 44 steps** design obj | **8-body 44 steps** MAE |
|---|---|---|---|---|---|---|---|---|
| CEM, GNS (1-step) | 0.3173 ± 0.0040 | 0.23293 ± 0.01007 | 0.3307 ± 0.0022 | 0.53521 ± 0.00987 | 0.3323 ± 0.0023 | 0.38632 ± 0.00737 | 0.3306 ± 0.0023 | 0.53839 ± 0.01001 |
| CEM, GNS | 0.3314 ± 0.0023 | 0.25325 ± 0.00369 | 0.3313 ± 0.0023 | 0.28375 ± 0.00336 | 0.3314 ± 0.0023 | 0.25325 ± 0.00369 | 0.3313 ± 0.0023 | 0.28375 ± 0.00336 |
| Backprop, GNS (1-step) | 0.2947 ± 0.0044 | 0.06008 ± 0.00437 | 0.2933 ± 0.0041 | 0.30416 ± 0.03387 | 0.3280 ± 0.0026 | 0.46541 ± 0.02768 | 0.3317 ± 0.0023 | 0.72814 ± 0.01783 |
| Backprop, GNS | 0.3221 ± 0.0043 | 0.09871 ± 0.00499 | 0.3195 ± 0.0042 | 0.15745 ± 0.00561 | 0.3251 ± 0.0021 | 0.15917 ± 0.00261 | 0.3299 ± 0.0022 | 0.21489 ± 0.00238 |
| **CinDM (ours)** | **0.2034** ± 0.0032 | **0.03928** ± 0.00161 | **0.2254** ± 0.0044 | **0.03163** ± 0.00251 | **0.3062** ± 0.0021 | **0.09241** ± 0.00210 | **0.3212 ± 0.0023** | **0.09249** ± 0.00276 |

## C ADDITIONAL BASELINE FOR TIME COMPOSITION OF THE N-BODY TASK

We also make a comparison with a simple baseline that performs diffusion over 44 steps directly without time composition. We designed this baseline to verify the effectiveness of our time-compositional approach. This baseline takes the same architecture as CinDM but with 44 time steps instead of 24 time steps, thus has almost twice of number of parameters in CinDM. The results are displayed in Table 8, which indicates that this sample baseline is outperformed by our CinDM. Its reason may be the difficulty in capturing dynamics across 44 time steps simultaneously using a single model, due to the presence of long-range dependencies. In such cases, a 24-step diffusion model proves to be more suitable. Hence, when dealing with designs that involve a larger number of time steps, employing time composition is a more effective approach, with lower cost and better performance.

Table 8: **Compositional Generalization Across Time. Comparison to a baseline that directly diffuses 44 steps without time composition.**

| **Methods** | **#parameters(Million)** | **2-body 44 steps** design_obj | **2-body 44 steps** MAE | **4-body 44 steps** design_obj | **4-body 44 steps** MAE |
|---|---|---|---|---|---|
| **Our method** | 20.76M | 0.1326 ± 0.0087 | 0.00695 ± 0.00067 | 0.2281 ± 0.0145 | 0.03195 ± 0.00705 |
| **Directly diffuse 44 steps** | 44.92M | 0.2779 ± 0.0197 | 0.00810 ± 0.00200 | 0.2986 ± 0.01481 | 0.05166 ± 0.01218 |

## D Additional details for compositional inverse design of 2D airfoils

### D.1 Details for the main experiment

**Dataset.** We use Lily-Pad (Weymouth, 2015) as our data generator (Fig. 5). We generate 30,000 ellipse bodies and NACA airfoil boundary bodies and perform fluid simulations around each body. The bodies are sampled by randomizing location, thickness, and rotation between respective ranges. Each body is represented by 40 two-dimensional points composing its boundary. The spatial resolution is $64 \times 64$ and each cell is equipped with temporal pressure and velocities in both horizontal and vertical directions. Each trajectory consists of 100 times steps. To generate training trajectories, we use a sliding time window over the 100 time steps. Each time window contains state data of $T = 6$ time steps with a stride of 4. So each original trajectory amounts to 25 training trajectories, and we get 750,000 training samples in total.

**Model architecture.** We use U-Net (Ronneberger et al., 2015) as our backbone for denoising from a random state sampled from a prior distribution. Without considering the mini-batch size dimension, the input includes a tensor of shape $(3T + 3) \times 64 \times 64$, which concatenates flow states (pressure, velocity of horizontal and vertical directions) of $T$ time steps and the boundary mask and offsets of horizontal and vertical directions along the channel dimension, and additionally the current diffusion step $s$. The output tensor shares the same shape with the input except $s$. The model architecture is illustrated in Fig. 4. The hyperparameters in our model architecture are shown in Table 9.

**Training.** We utilize the MSE (mean squared error) between prediction and a Gaussian noise as the loss function during training. We take a batch size of 48 and run for 700,000 iterations. The learning rate is initialized as $1 \times 10^{-4}$. Training details are provided in Table 10.

**Evaluation of design results.** In inference, we set $\lambda$ in Eq. 3 as 0.0002. We find that this $\lambda$ could get the best design result. More discussion on the selection of $\lambda$ is presented in Appendix I. For each method and each airfoil design task (one airfoil or two airfoils), we conduct 10 batches of design, and each batch contains 20 examples. After we get the designed boundaries, we input them into Lily-Pad and run the simulation. To make the simulation more accurate and convincing, we use a $128\times128$ resolution of the flow field, instead of $64 \times 64$ as in the generation of training data. Then we use the calculated horizontal and vertical flow force to compute our two metrics: $-$lift $+$ drag and lift-to-drag ratio. In each batch, we choose the best-designed boundary (or pair of boundaries in two airfoil scenarios) and then we report average values regarding the two metrics over 10 batches.

Table 9: **Hyperparameters used in 2D diffusion model architecture**.

| | |
|---|---|
| Number of downsampling blocks | 4 |
| Number of upsampling blocks | 4 |
| Input channels | 21 |
| Number of residual blocks for each layer | 2 |
| Batch size | 48 |
| Input shape | $[48, 21, 64, 64]$ |
| Output shape | $[48, 21, 64, 64]$ |

### D.2 Surrogate Model for Force Prediction

**Model architecture.** In the 2D compositional inverse design of multiple airfoils, we propose a neural surrogate model $g_\varphi$ to approximate the mapping from the state $U_t$ and boundary $\gamma$ to the lift and drag forces, so that the design objective $\mathcal{J}$ is differentiable to the design variable $z = U_{[0,T]} \bigoplus \gamma$. The input of our model is a tensor comprising pressure, boundary mask, and offsets (both horizontal and vertical directions) of shape $4 \times 64 \times 64$ for a given time step. The output is the predicted drag and lift forces of dimension 2. Boundary masks indicate the inner part (+1) and outside part (0) of a closed boundary. Offsets measure the signed deviation of the center of each square on a $64 \times 64$ grid from the boundary in horizontal and vertical direction respectively, where the deviation of a given point is defined as its distance to the nearest point on a boundary. If two or more boundaries appear in a sample, the input mask (resp. offsets) is given by the summation

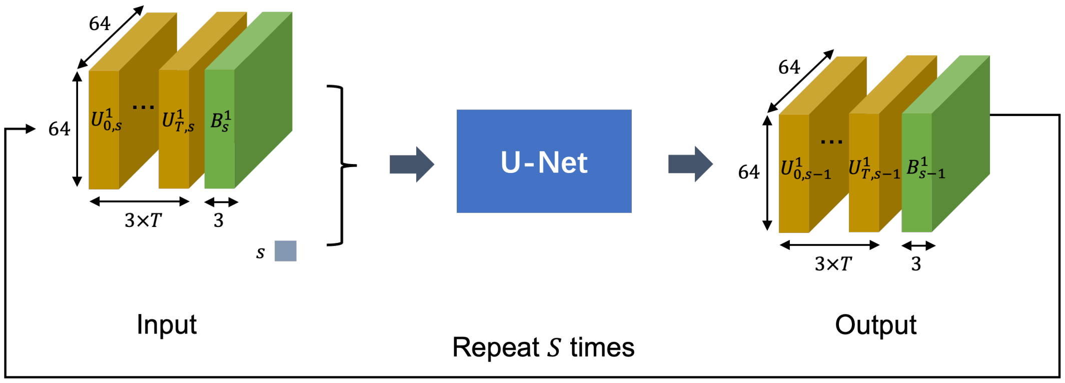


Figure 4: **Diffusion model architecture of 2D inverse design**.

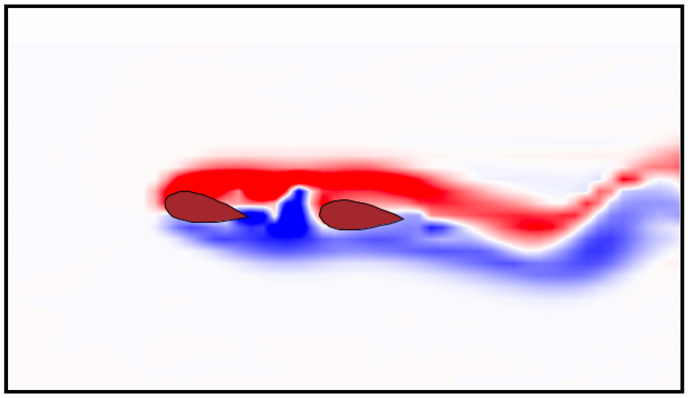


Figure 5: **Example of Lily-Pad simulation**.

of masks (resp. offsets) of all the boundaries. Notice that since the input boundaries are assumed not to be overlapped, the summed mask and offset are still valid. The model architecture is *half* of a U-Net Ronneberger et al. (2015) where we only take the down-sampling part to embed the input features to a 512-dimensional representation; then we use a linear transformation to output forces.

**Dataset.** We use Lily-Pad (Weymouth, 2015) to generate simulation data with 1, 2, or 3 airfoil boundaries to train and evaluate the surrogate model. Boundaries are a mixture of ellipses and NACA airfoils. We generate 10,000 trajectories for the training dataset and 1,000 trajectories for the test dataset. Each trajectory consists of 100 time steps. We use pressure as features and lift and drag forces as labels. Thus we have 3 million training samples and 300 thousand testing samples in total.

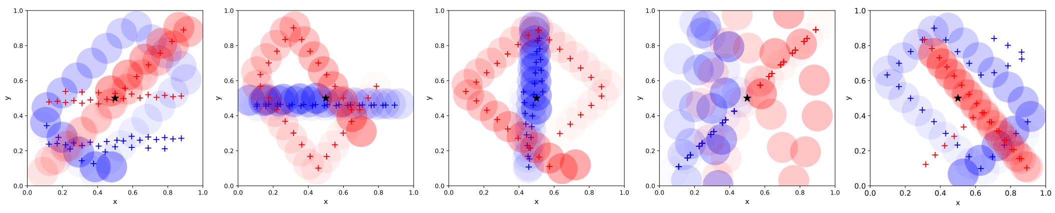
**Training.** We use MSE (mean squared error) loss between the ground truth and predicted forces to train the surrogate model. The optimizer is Adam (Kingma & Ba, 2014). The batch size is 128. The model is trained for 20 epochs. The learning rate starts from $1 \times 10^{-4}$ and multiplies a factor of 0.1 every five epochs. The test error is 0.04, smaller than 5% of the average force in the training dataset.

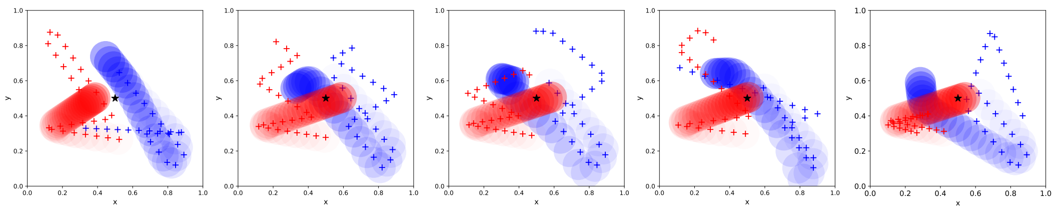
Table 10: **Hyperparameters used in 2D diffusion model training**.

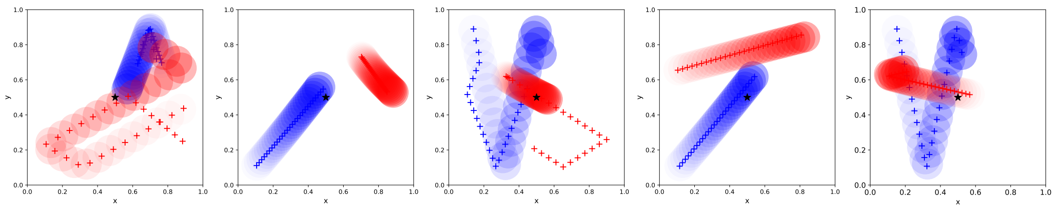
| | |
|---|---|
| Loss function | MSE |
| Number of examples for training dataset | $3 \times 10^6$ |
| Total number of training steps | $7 \times 10^5$ |
| Batch size | 48 |
| Initial learning rate | $1 \times 10^{-4}$ |
| Number of training steps with a fixed learning rate | $6 \times 10^5$ |
| Learning rate adjustment strategy | StepLR |
| Optimizer | Adam |
| Number of saving checkpoint | 700 |
| Exponential Moving Average decay rate | 0.995 |

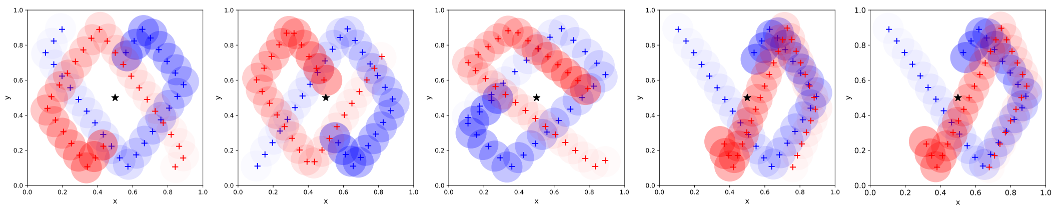
## E VISUALIZATION OF N-BODY INVERSE DESIGN.

Examples of N-body design results are provided in this section. Figure 6 shows the results of using the backpropagation algorithm and CinDM to design 2-body 54-time step trajectories. The results of designing 2-body 54-time steps trajectories using CEM and CinDM are provided in Figure 7. Figure 8 are the results of designing44-time 44 time steps trajectories using CEM, backpropagation, and CinDM.


(a)


(b)


(c)


(d)

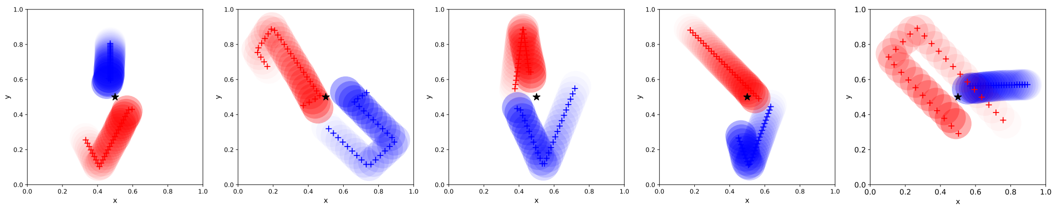
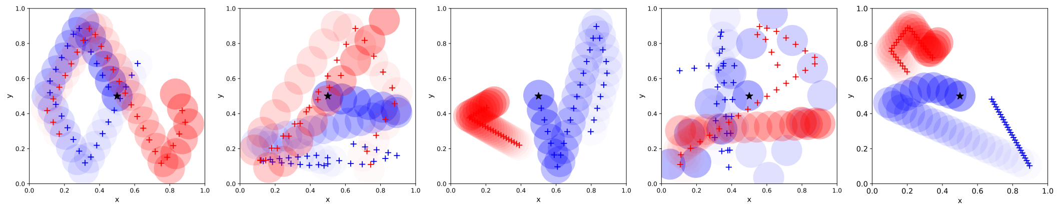
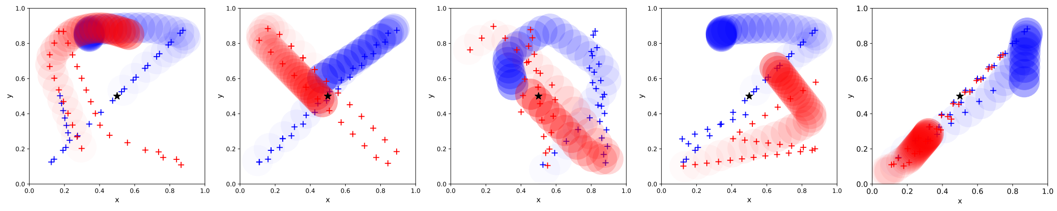

(e)

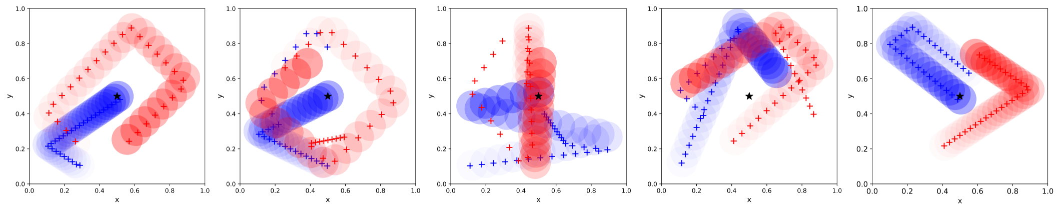
Figure 6: **54 time steps trajectories of 2 bodies after performing inverse design using the back-propagation algorithm**. Figures (a), (b), (c), and (d) represent the trajectory graphs obtained using GNS, GNS (single step), U-Net, and U-Net (single step) as the forward models, respectively. And (e) is the result of CinDM. The legend of this figure is consistent with Figure 2.



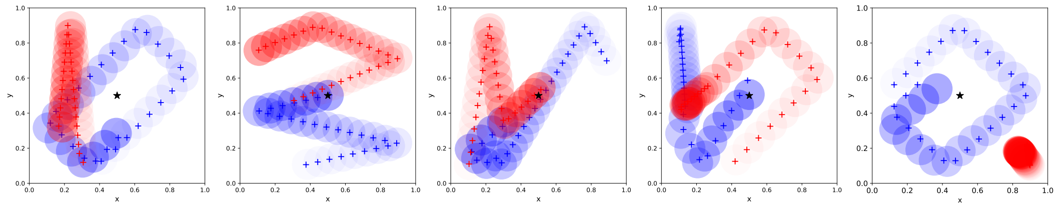
(a)



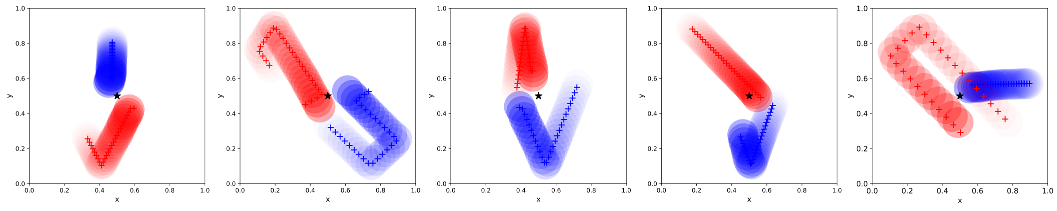
(b)



(c)

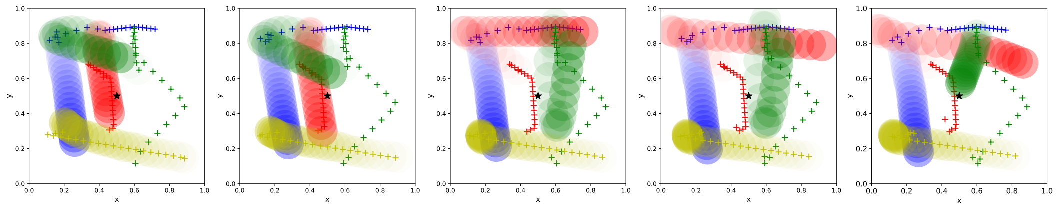


(d)

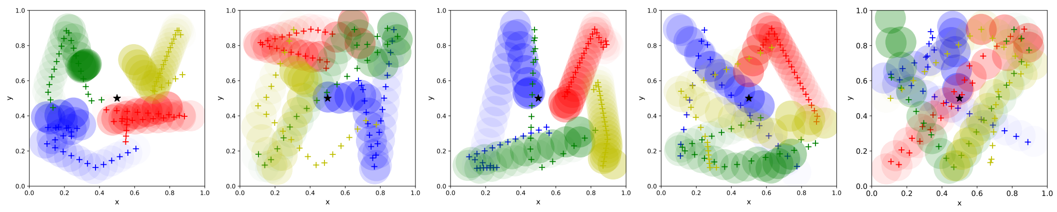


(e)

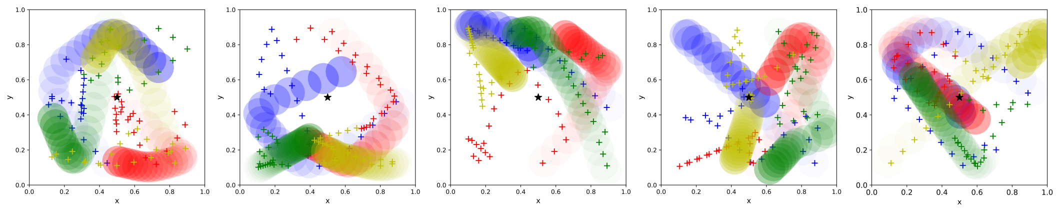
Figure 7: **54-step trajectories of 2 bodies after performing inverse design using the CEM algorithm**. The trajectory graphs (a), (b), (c), and (d) depict the outcomes using different forward models such as GNS, GNS (single step), U-Net, and U-Net (single step) respectively. Additionally, figure (e) demonstrates the result generated by CinDM. The legend of this figure is consistent with Figure 2.



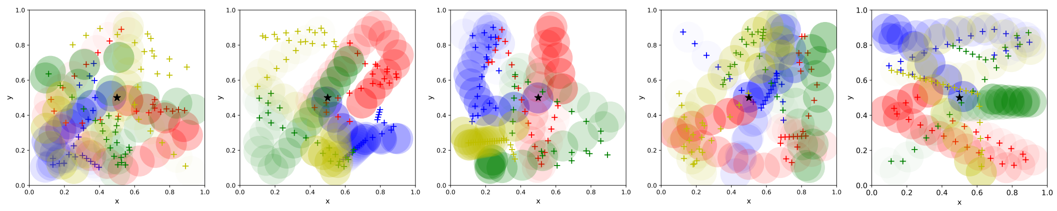
(a)



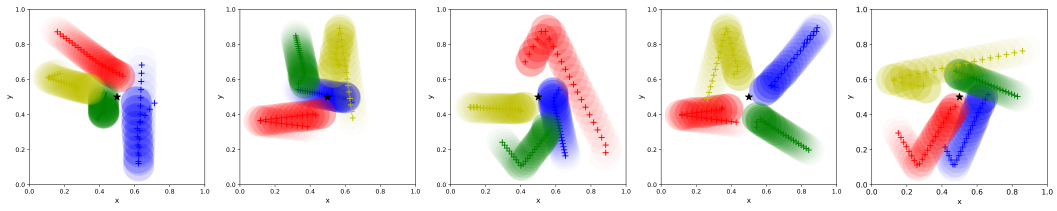
(b)



(c)



(d)



(e)

Figure 8: **44-time steps trajectories of 4 bodies after performing inverse design using CEM**. Figures (a), (b), (c), and (d) represent the trajectory graphs obtained using GNS, GNS (single step), U-Net, and U-Net (single step) as the forward models, respectively. And (e) is the result of CinDM. The legend of this figure is consistent with Figure 2.

## F VISUALIZATION RESULTS OF 2D INVERSE DESIGN BY OUR CINDM

We show the compositional design results of our method in 2D airfoil generation in Figure 9.

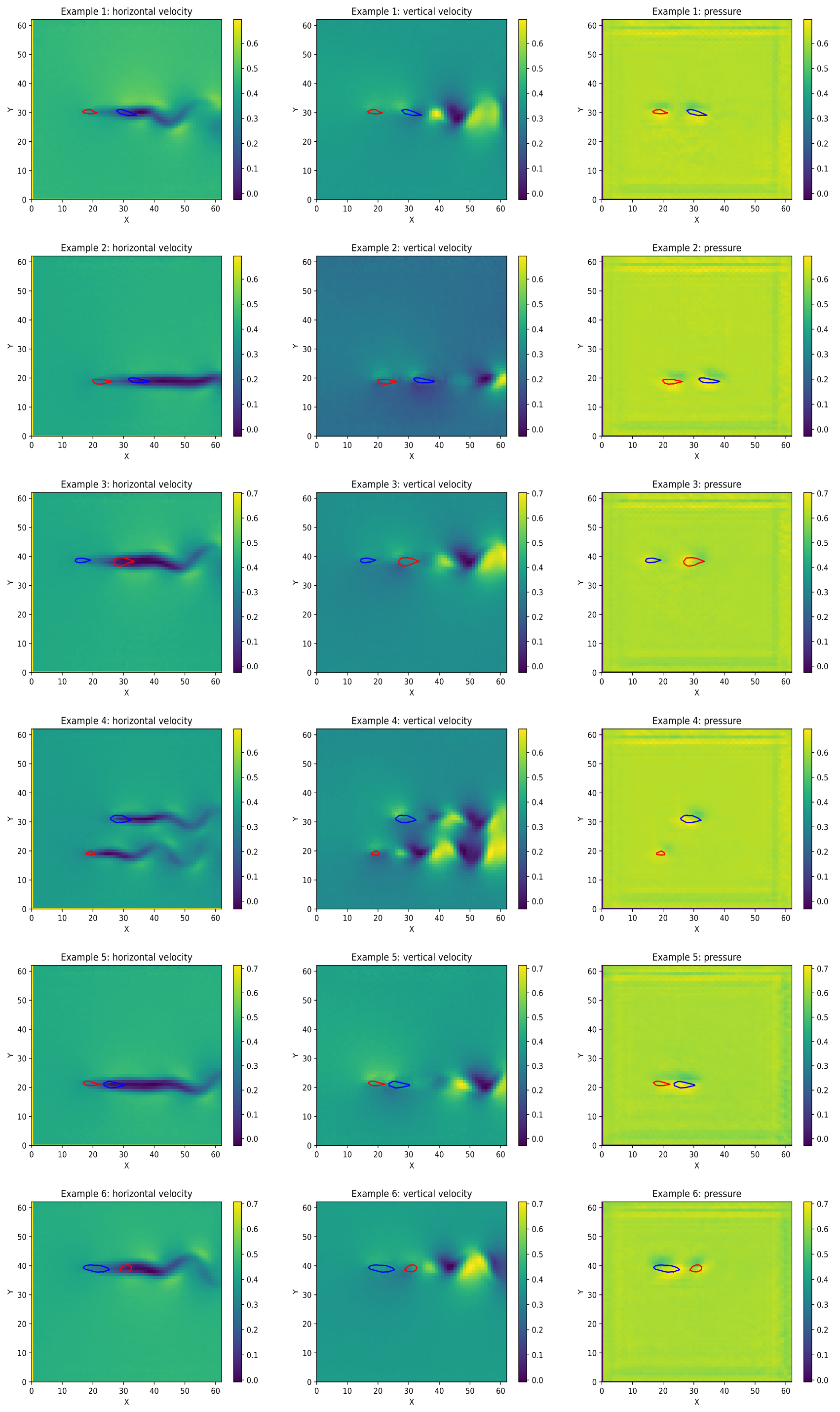
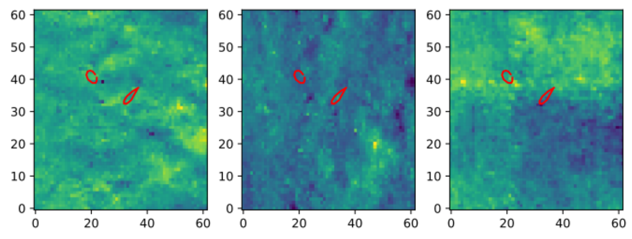
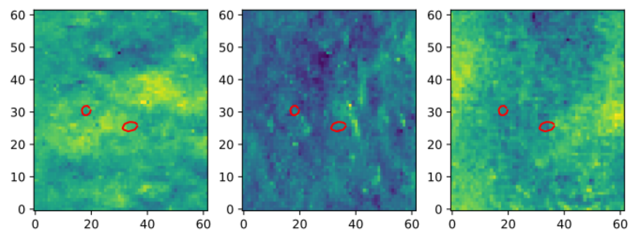


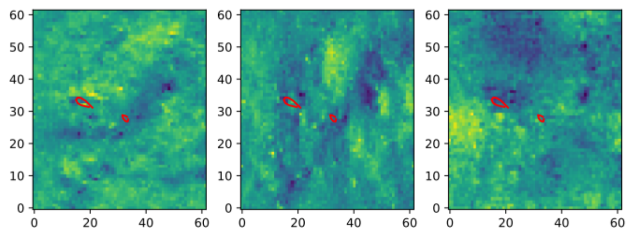
Figure 9: **Compositional design results of our method in 2D airfoil generation**. Each row represents an example. We show the heatmap of velocity in horizontal and vertical direction and pressure in the initial time step, inside which we plot the generated airfoil boundaries.

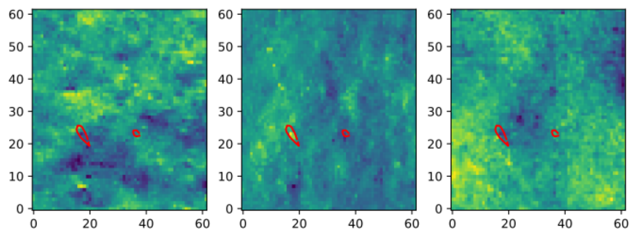
## G SOME VISUALIZATION RESULTS OF 2D INVERSE DESIGN BASELINE.

We show some 2D design results of our baseline model in Fig. 10.


(a)


(b)


(c)


(d)

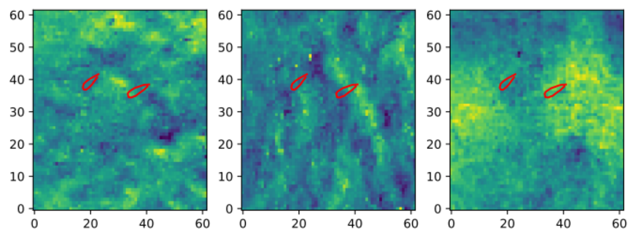

(e)

Figure 10: **Design results of FNO with CEM in 2D airfoil generation**. Each row is the heatmap of optimized velocities in the horizontal and vertical direction and optimized pressure in the initial time step, inside which we plot the generated airfoil boundaries.

Table 11: **Comparison to NABL and cINN for N-body time composition inverse design task.**

| Method | 2-body 24 steps design obj | 2-body 24 steps MAE | 2-body 34 steps design obj | 2-body 34 steps MAE | 2-body 44 steps design obj | 2-body 44 steps MAE | 2-body 54 steps design obj | 2-body 54 steps MAE |
|---|---|---|---|---|---|---|---|---|
| NABL, U-Net (1-step) | 0.1174 | 0.01650 | 0.1425 | 0.01511 | 0.1788 | 0.01185 | 0.2606 | 0.02042 |
| cINN | 0.3235 | 0.11704 | 0.3085 | 0.18015 | 0.3478 | 0.18322 | 0.3372 | 0.19296 |
| **CinDM (ours)** | **0.1143** | **0.01202** | **0.1251** | **0.00763** | **0.1326** | **0.00695** | **0.1533** | **0.00870** |

Table 12: **Comparison to NABL and cINN for 2D airfoils inverse design task.**

| Method | # parameters (Million) | 1 airfoil design obj ↓ | 1 airfoil lift-to-drag ratio ↑ | 2 airfoils design obj ↓ | 2 airfoils lift-to-drag ratio ↑ |
|---|---|---|---|---|---|
| NABL, FNO | 3.29 | 0.0323 | 1.3169 | 0.3071 | 0.9541 |
| NABL, LE-PDE | 3.13 | 0.1010 | 1.3104 | 0.0891 | 0.9860 |
| cINN | 3.07 | 1.1745 | 0.7556 | - | - |
| **CinDM (ours)** | 3.11 | 0.0797 | **2.177** | 0.1986 | **1.4216** |

## H Comparison to additional works

Besides comparison results of baselines shown in the main text, we further evaluated additional two baselines: neural adjoint method + boundary loss function (NABL) and conditional invertible neural network (cINN) method for both N-body and airfoils design experiments.

We implement NABL on top of baselines FNO and LE-PDE in the airfoil design task and U-net in tcompositionalostional taskamed as “NABL, FNO”, “NABL, LE-PDE” and “NABL, U-net” respectively. These new NABL baselines additionally use the boundary loss defined by the mean value and 95% significance radius of the training dataset. cINN does not apply to compositional design because the input scale for the invertible neural network function is fixed. Therefore, for the time composition task, we trained 4 cINN models, each for one of the time steps: 24, 34, 44, and 54. These models differ only in the input size. The input $x$ to cINN is a vector of size $2 \times 4 \times T$, where 2 is the number of objects, 4 is the number of features and $T$ is the number of time steps. The condition $y$ is set to 0, the minimal distance to the target point. For cINN for 2D airfoil design, we adopt 2D coordinates of 40 boundary pointsarewhich is spanned 80-dimensionalensional vector, as the input, since the invertible constraint on the cINN model hardly accepts image-like inputs adopted in the main experiments. Therefore we evaluate cINN only in the single airfoil design task. The condition $y$ is set as the minimal value of drag - lift drag in the training trajectories. In both tasks, the random variable $z$ has a dimension of dim($x$) - dim($y$). It is drawn from a Gaussian distribution and then input to the INN for inference. We also adjust the hyperparameters, such as hidden size and a number of reversible blocks, to make the number of parameters in cINN close to ours for fair comparison.

The results of NABL and cINN are shown in Table 11 and Table 12. We can see that CinDM significantly outperforms the new baselines in both experiments. Even compared to the original baselines (who contains contain “Backprop-”) without the boundary loss function, as shown in Table 1 and Table 3, the NABL baselines in both tasks do not show the improvement in the objective for out-of-distribution data. These results show that our method generalizes to out-of-distribution while the original and new baselines struggle to generalize the out-of-distribution. CinDM also outperforms cINN by a large margin in both the time composition and airfoil design tasks. Despite the quantities, we also find that airfoil boundaries generated by cINN have little variation in shape, and the orientation is not as desired, which could incur high drag force in simulation. These results may be caused by the limitation of the model architecture of cINN, which utilizes fully connected layers as building blocks, and thus has an obvious disadvantage in capturing inductive bias of spatial-temporal features. We think it is necessary to extend cINN to convolutional networks when cINN is applied to such high-resolution design problems. However, this appears challenging when the invertible requirement is imposed. In summary, our method outperforms both NABL and cINN in both tasks. Furthermore, our method could be used for flexible compositional design. We use only one trained model to generate samples lying in a much larger state space than in training during inference, which is a unique advantage of our method beyond these baselines.

## I PERFORMANCE SENSITIVITY TO HYPERPARAMETERS, INITIALIZATION AND SAMPLING STEPS.

This section evaluate the effects of different $\lambda$, initialization and sampling steps on performance of CinDM.

### I.1 INFLUENCE OF THE HYPERPARAMETER $\lambda$

Table 13: **Effect of $\lambda$ in N-body time composition inverse design.**

| $\lambda$ | 2-body 24 steps | | 2-body 34 steps | | 2-body 44 steps | | 2-body 54 steps | |
|---|---|---|---|---|---|---|---|---|
| | **design_obj** | **MAE** | **design_obj** | **MAE** | **design_obj** | **MAE** | **design_obj** | **MAE** |
| **0.0001** | 0.3032 ± 0.0243 | 0.00269 ± 0.00047 | 0.2954 ± 0.0212 | 0.00413 ± 0.00155 | 0.3091 ± 0.0223 | 0.00394 ± 0.00076 | 0.2996 ± 0.0201 | 0.01046 ± 0.00859 |
| **0.001** | 0.2531 ± 0.0185 | 0.00385 ± 0.00183 | 0.2937 ± 0.0213 | 0.00336 ± 0.00115 | 0.2797 ± 0.0190 | 0.00412 ± 0.00105 | 0.2927 ± 0.0219 | 0.00521 ± 0.00103 |
| **0.01** | 0.1200 ± 0.0069 | 0.00483 ± 0.00096 | 0.1535 ± 0.0135 | 0.00435 ± 0.00100 | 0.1624 ± 0.0137 | 0.00416 ± 0.00059 | 0.1734 ± 0.0154 | 0.00658 ± 0.00267 |
| **0.1** | 0.1201 ± 0.0046 | 0.01173 ± 0.00150 | 0.1340 ± 0.0107 | 0.00772 ± 0.00099 | 0.1379 ± 0.0088 | 0.00816 ± 0.00149 | 0.1662 ± 0.0180 | 0.01141 ± 0.00473 |
| **0.2** | 0.1283 ± 0.0141 | 0.01313 ± 0.00312 | 0.1392 ± 0.0119 | 0.00836 ± 0.00216 | 0.1529 ± 0.0130 | 0.01019 ± 0.00584 | 0.1513 ± 0.0131 | 0.00801 ± 0.00172 |
| **0.4** | 0.1172 ± 0.0084 | 0.01500 ± 0.00207 | 0.1385 ± 0.0145 | 0.00948 ± 0.00293 | 0.1402 ± 0.0113 | 0.00763 ± 0.00112 | 0.1663 ± 0.0126 | 0.00850 ± 0.00124 |
| **0.6** | 0.1259 ± 0.0100 | 0.01382 ± 0.00115 | 0.1326 ± 0.0126 | 0.01171 ± 0.00595 | 0.1592 ± 0.0151 | 0.01140 ± 0.00355 | 0.1670 ± 0.0177 | 0.00991 ± 0.00287 |
| **0.8** | 0.1217 ± 0.0073 | 0.01596 ± 0.00127 | 0.1385 ± 0.0120 | 0.01095 ± 0.00337 | 0.1573 ± 0.0116 | 0.00893 ± 0.00113 | 0.1715 ± 0.0181 | 0.01026 ± 0.00239 |
| **1** | 0.1330 ± 0.0063 | 0.01679 ± 0.00139 | 0.1428 ± 0.0112 | 0.01087 ± 0.00149 | 0.1634 ± 0.0119 | 0.00968 ± 0.00079 | 0.1789 ± 0.0164 | 0.01102 ± 0.00185 |
| **2** | 0.1513 ± 0.0079 | 0.02654 ± 0.00160 | 0.1795 ± 0.0129 | 0.01765 ± 0.00193 | 0.1779 ± 0.0121 | 0.01707 ± 0.00474 | 0.2113 ± 0.0161 | 0.01447 ± 0.00130 |
| **10** | 0.2821 ± 0.0197 | 0.21153 ± 0.01037 | 0.2210 ± 0.0149 | 0.09715 ± 0.00236 | 0.2273 ± 0.0133 | 0.07781 ± 0.00232 | 0.2269 ± 0.0175 | 0.06538 ± 0.00210 |

Table 14: **Effect of $\lambda$ in 2D inverse design.**

| $\lambda$ | **obj** | **lift/drag** |
|---|---|---|
| **0.05** | 0.7628±0.1892 | 1.015±0.2008 |
| **0.02** | 0.3849±0.0632 | 1.0794±0.1165 |
| **0.01** | 0.2292±0.0408 | 1.286±0.1402 |
| **0.005** | 0.2061±0.0388 | 1.2378±0.1414 |
| **0.002** | 0.217±0.0427 | 1.2429±0.1243 |
| **0.001** | 0.2277±0.0451 | 1.2608±0.1469 |
| **0.0005** | 0.2465±0.0473 | **1.4102±0.1771** |
| **0.0002** | **0.1986±0.0431** | 1.4216±0.1607 |
| **0.0001** | 0.271±0.0577 | 1.1962±0.1284 |

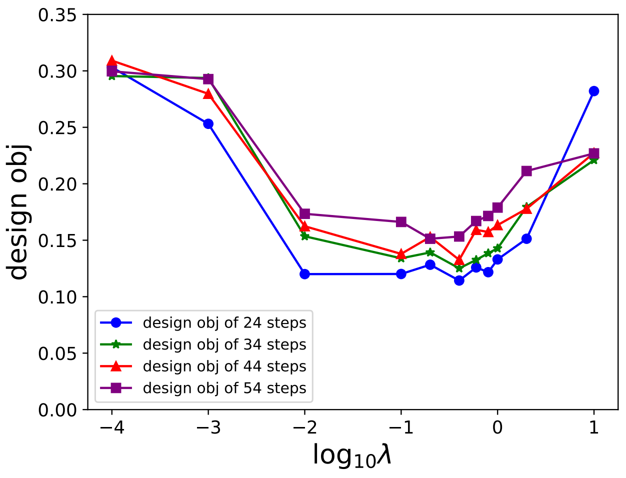


Figure 11: **Design objective of different $\lambda$ in N-body time composition inverse design.**

To evaluate influence of the hyperparameter $\lambda$ in Eq. 3, we perform inference in both N-body time composition and 2D airfoils design task for a wide range of $\lambda$. The results are shown in Table 13, Table 14, Fig 11, Fig 12, and Fig 13, where Table 13 corresponds to Fig 11 and Fig 12 while Table Table 14 corresponds to Fig 13. Our method demonstrates robustness and consistent performance across a wide range of lambda values. However, if $\lambda$ is set too small ($\leq$0.0001 in the 2D airfoil task, or $\leq$0.01 in the N-body task), the design results will be subpar because there is minimal objective

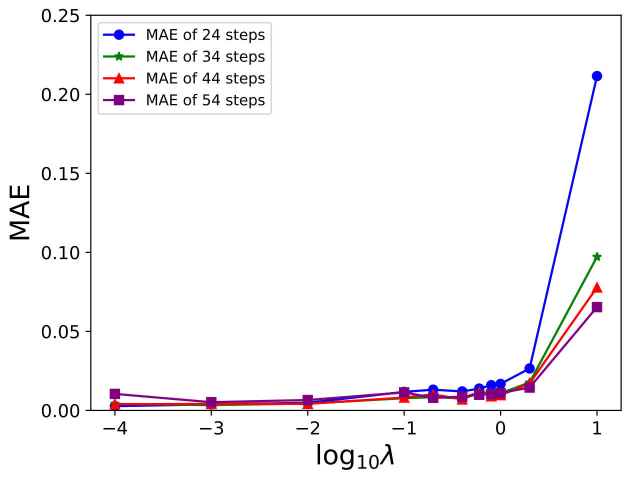


Figure 12: **MAE of different $\lambda$ in N-body time composition inverse design.**

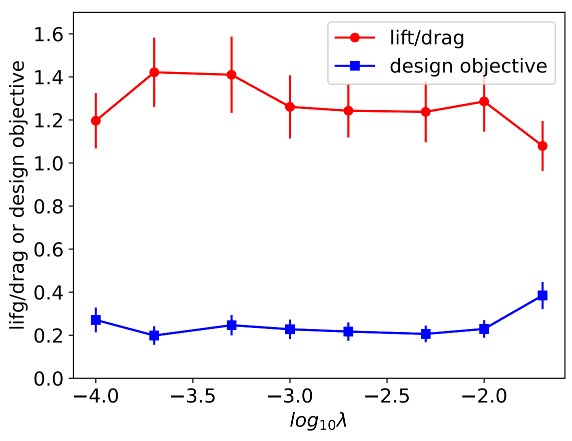


Figure 13: **Performance of different $\lambda$ in 2D airfoil inverse design.**

guidance incorporated. On the other hand, if $\lambda$ is set too large ($\geq$ 0.01 in the 2D airfoil task, or $\geq$ 1.0 in the N-body task), there is a higher likelihood of entering a poor likelihood region, and the preservation of physical consistency is compromised. In practical terms, $\lambda$ can be set between 0.01 and 1.0 for the N-body task, and between 0.0002 and 0.02 for the 2D airfoil task. In our paper, we choose based on the best evaluation performance, namely we set as 0.4 for the N-body task and 0.0002 for the 2D airfoils task.

### I.2 Influence of initialization

To analyze the sensitivity of initialization in our approach, we follow a similar methodology discussed in Ren et al. (2020). We consider the "re-simulation" error $r$ of a target objective $y$ as a function of the number of samplings $B$, where each sampling starts from a Gaussian initialization $z$. We use the simulator to obtain the output $\hat{y}$ for each design $x$ from the $B$ design results given the target $y$ and compute the "re-simulation" error $L(\hat{y}, y)$. We then calculate the least error among a batch of $B$ design results. This process is repeated for several batches, and the mean least error $r_B$ is obtained by averaging over these batches.

Table 15 and Fig 14 present the results for the N-body inverse design task. We consider values of $B$ ranging from 10 to 100, with $N = 10$ batches. The target $y$ is set to be 0, which represents the

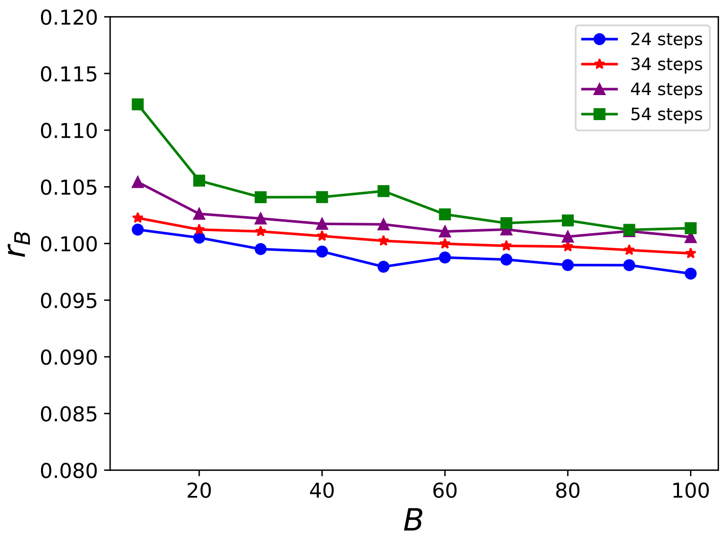


Figure 14: **"Re-simulation" error $r_B$ of different $B$ in N-body inverse design.**

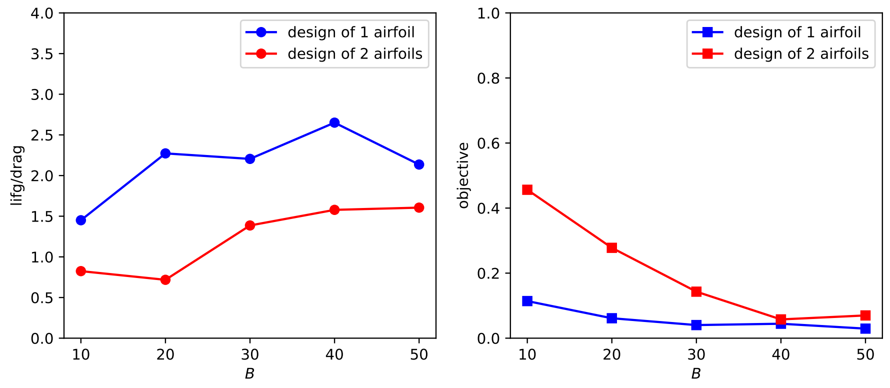


Figure 15: **Design performance (lift-to-drag) of different $B$ in 2D airfoil inverse design.**

distance to a fixed target point. The results show that $r_B$ gradually decreases as $B$ increases in the 24-step design, indicating that the design space is well explored and most solutions can be retrieved even with a small number of samplings. This demonstrates the efficiency of our method in generating designs. Moreover, similar observations can be made when time composition is performed in 34, 44, and 54 steps, indicating the effectiveness of our time composition approach in capturing long-time range physical dependencies and enabling efficient generation in a larger design space.

In the 2D inverse design task, the target $y$ is slightly different. Here, we aim to minimize the model output (drag - lift force). Hence, we adopt the "re-simulation" performance metric, which is the lift/drag ratio, as opposed to the "re-simulation" error used in the N-body task, to evaluate sensitivity to initialization. For each $B$, the lift/drag ratio is chosen as the highest value among the simulation results of a batch of $B$ designed boundaries (or boundary pairs for the 2 airfoils design). Any invalid design results, such as overlapping airfoil pairs in the 2-airfoil design, are removed from the $B$ results before computing the maximal lift/drag ratio. The reported numbers are obtained by averaging over $N = 10$ batches for each $B$.

Table 16 and Fig 15 present the results for the 2D airfoils design task. In the 1 airfoil design column, we observe that the lift/drag ratio is relatively low for $B = 10$, indicating that the design space is not

Table 15: **Influence of initialization**. $r_B$ with respect to $B$ for N-body inverse design task. Each number is an average over 10 batches.

| $B$ | **2-body 24 steps** | **2-body 34 steps** | **2-body 44 steps** | **2-body 54 steps** |
|---|---|---|---|---|
| **10** | 0.10122654 | 0.1022556 | 0.10542078 | 0.11227837 |
| **20** | 0.10051114 | 0.10122902 | 0.10261874 | 0.10554917 |
| **30** | 0.09950846 | 0.10106587 | 0.10220513 | 0.10408381 |
| **40** | 0.09928784 | 0.10066015 | 0.10173534 | 0.10409425 |
| **50** | 0.09794939 | 0.10023642 | 0.10168899 | 0.10462530 |
| **60** | 0.09876589 | 0.09997466 | 0.10105932 | 0.10257294 |
| **70** | 0.09858151 | 0.09979441 | 0.10124100 | 0.10179855 |
| **80** | 0.09809845 | 0.09972977 | 0.10060663 | 0.10203485 |
| **90** | 0.09808731 | 0.09941968 | 0.10108861 | 0.10120515 |
| **100** | 0.09734109 | 0.09912691 | 0.10056177 | 0.10135190 |

Table 16: **Influence of initialization**. Design performance (lift-to-drag) with respect to $B$ for 2D inverse design task. Each number is an average over 10 batches.

| $B$ | **1 airfoil** | **2 airfoils** |
|---|---|---|
| **10** | 1.4505 | 0.8246 |
| **20** | 2.2725 | 0.7178 |
| **30** | 2.2049 | 1.3862 |
| **40** | 2.6506 | 1.5781 |
| **50** | 2.1355 | 1.6055 |

sufficiently explored due to its high dimensionality ($64 \times 64 \times 3$ in our boundary mask and offsets representation). For $B \geq 20$, the lift/drag performance remains steady. In the 2 airfoils design column, the lift/drag ratio increases roughly with $B$. This is attributed to the higher dimensional and more complex design space compared to the single airfoil design task. The stringent constraints on boundary pairs, such as non-overlapping, lead to the presence of complex infeasible regions in the design space. Random initialization may lead to these infeasible regions, resulting in invalid design results. The rate of increase in lift/drag ratio becomes slower when $B \geq 30$, indicating that a majority of solutions have been explored. Despite the training data only containing a single airfoil boundary, which lies in a relatively lower dimensional and simpler design space, our model demonstrates a strong ability to generalize and efficiently generate designs for this challenging 2 body compositional design problem.

### I.3 Influence of the number of sampling steps in inference

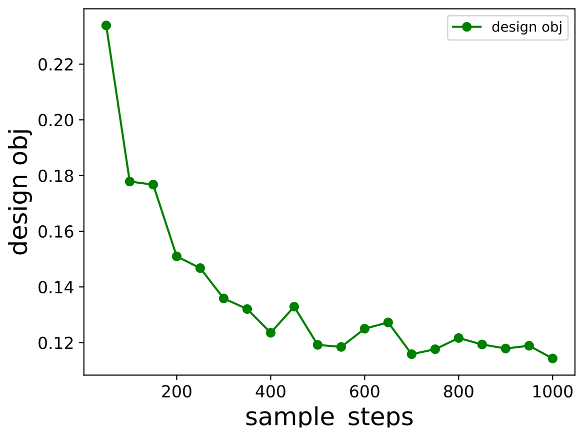


Figure 16: **Design objective of different sampling steps in N-body inverse design.**

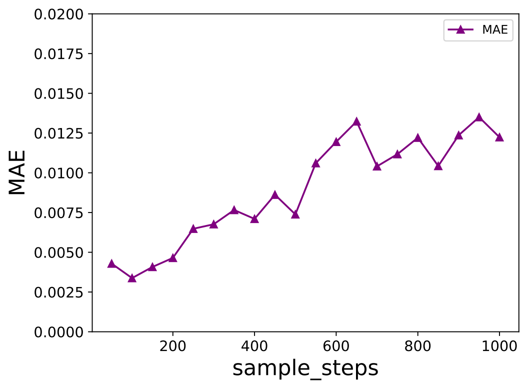


Figure 17: **MAE of different sampling steps in N-body inverse design.**

Fig 16 and Fig 17 illustrate the outcomes of inverse design carried out by CinDM. It is apparent that with an increase in the number of sampling time steps, the design objective gradually decreases. In contrast, the MAE fluctuates within a small range, occasionally rising. This phenomenon can be examined as follows: as the number of sampling steps increases, the participation of the design objective in the diffusion process intensifies. As a result, the designs improve and align more closely with the design objective, ultimately leading to a decrease in the design objective. However, when the number of sampling steps increases, the MAE also increases. This is because, with a small number of sampling steps, the initial velocities of some designed samples are very small, causing the diffusion of trajectories to be concentrated within a narrow range. Consequently, both the true trajectory and the diffused trajectory are highly concentrated, resulting in a small calculated MAE. By analyzing the sensitivity of the design objective and MAE to different sampling steps, we can conclude that CInDM can achieve desired design results that align with design objectives and physical constraints by appropriately selecting a sampling step size during the inverse design process.

## J Broader impacts and limitations

Our method, CinDM, extends the scope of design exploration and enables efficient design and control of complex systems. Its application across various scientific and engineering fields has profound implications. In materials science, utilizing the diffusion model for inverse design facilitates the customization of material microstructures and properties. In biomedicine, it enables the structural design of drug molecular systems and optimizes production processes. Furthermore, in the aerospace sector, integrating the diffusion model with inverse design can lead to the development of more diverse shapes and structures, thereby significantly enhancing design efficiency and quality.

CinDM combines the advantages of diffusion models, allowing us to generate more diverse and sophisticated design samples. However, some limitations need to be addressed at present. In terms of design quality and exploration space, we need to strike a balance between different objectives to avoid getting stuck in local optima, especially when dealing with complex, nonlinear systems in the real world. We also need to ensure that the designed samples adhere to complex multi-scale physical constraints. Furthermore, achieving interpretability in the samples designed by deep learning models is challenging for inverse design applications. From a cost perspective, training diffusion models requires large datasets and intensive computational resources. The complexity of calculations also hinders the speed of our model design.

Moving forward, we intend to incorporate more physical prior knowledge into the model, leverage multi-modal data for training, employ more efficient sampling methods to enhance training efficiency, improve interpretability, and generalize the model to multiple scales.