A DETAILS OF STATIC REGRESSION EXPERIMENTS

In this section, we provide the model architectures, training process, and additional experimental results for the polynomial regression task.

Model architecture We employ `Encoder`, `CAVIA`, and `CoDA` as the baselines. All models use the same multi-layer perceptron (MLP) for f_θ and are trained with the same data batches for fair comparisons.

For brevity, we refer to a multi-layer perceptron (MLP) with hidden dimensions n_1, n_2, ... n_l for each layer and hidden activation `act`, as `MLP`(n_1, n_2, ..., n_l; `act`). We refer to a cross multi-head attention block (Vaswani et al., 2017) with h heads and x hidden dimensions as `X.MHA`($h \times x$).

Table A.1: Polynomial regression model architectures

	Context encoder / parameter generator	f_θ	Inner step K	Inner step size α	τ
`Encoder`	`MLP(2,32)/MLP(1,32)-X.MHA(4×32)`	`MLP(1+32, 64, 32, 1;LeakyReLU)`	–	–	–
`CAVIA`	–	`MLP(1+32, 64, 32, 1;LeakyReLU)`	5	1	–
`CoDA`	$\boldsymbol{W} : \mathbb{R}^{\lvert c\rvert} \to \mathbb{R}^{\lvert\theta\rvert}$	`MLP(1, 64, 32, 1;LeakyReLU)`	500	0.001	–
`FOCA`	–	`MLP(1+32, 64, 32, 1;LeakyReLU)`	100	0.001	0.1

Table A.1 summarizes the network architectures. The green colored values indicate the dimension of context $\hat{c}$. For `CAVIA`, we also perform hyperparameter search to optimize K. We found that `CAVIA` with $K > 5$ underperforms, as compared to $K = 5$. For `CoDA`, we set the dimension of context as 2 (i.e., $|c| = 2$), following the default setting of Kirchmeyer et al. (2022).

Training details We train all models with mini-batches of 256 polynomials for 4,048 epochs using Adam (Kingma & Ba, 2015) with an initial learning rate of 0.001. The learning rate is scheduled by the cosine annealing method (Loshchilov & Hutter, 2017).

B DETAILS OF MASS SPRING (MS) EXPERIMENTS

In this section, we provide the model architectures, training process, and additional experimental results for the mass-spring systems.

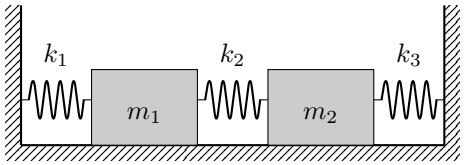

Figure A.1: Target mass-spring system. The models require to adapt to the change of spring constants (k_1, k_2, k_3) and masses (m_1, m_2).

We consider a frictionless three-spring two-mass system shown in Fig. A.1 with mass positions x_1 and x_2 governed by the following second-order ODE:

$$\begin{aligned} \frac{\mathrm{d}^2 x_1}{\mathrm{d}t^2} &= -\frac{k_1 + k_2}{m_1} x_1 + \frac{k_2}{m_1} x_2 \\ \frac{\mathrm{d}^2 x_2}{\mathrm{d}t^2} &= -\frac{k_2 + k_3}{m_2} x_2 + \frac{k_2}{m_2} x_1 \end{aligned} \tag{A.1}$$

where K is a coefficient matrix with spring constants (k_1, k_2, k_3) and masses (m_1, m_2).

Data generation For training data generation, we generate 128 mass-spring systems whose parameters $(k_1, k_2, k_3, m_1, m_2)$ are sampled from $\mathcal{U}(0.75, 1.25)^5$ and numerically solve the mass-spring systems with Runge–Kutta 45 for $T = 10$ seconds with $\Delta t = 0.15$ second time intervals.

Model architecture For f_θ, we employ the 1D-CNN model from Brandstetter et al. (2021), which stacks a MLP, 1D-CNN, and consistency decoder (Brandstetter et al., 2021) with the bundling parameter $N = 25$. For brevity, we refer to a 1D-CNN layer with x input channels, y output channels, and filter size w as `Conv`(x, y, w), a consistency decoder as `C.Dec`, and the bi-directional GRU (Cho et al., 2014) with hidden dimension x as `GRU`(x).

Table A.2: Mass-spring prediction model architectures

	Context encoder / parameter generator	f_θ
`Encoder`	`GRU(64)-MLP(64, 64)`	`MLP(4×25+64, 64, 32, 4×25;LeakyReLU)-Conv(4, 4, 8)-LeakyReLU-Conv(4,4,1)-C.Dec`
`CAVIA`	–	`MLP(4×25+64, 64, 32, 4×25;LeakyReLU)-Conv(4, 4, 8)-LeakyReLU-Conv(4,4,1)-C.Dec`
`CoDA`	$\boldsymbol{W} : \mathbb{R}^{\lvert c\rvert} \to \mathbb{R}^{\lvert\theta\rvert}$	`MLP(4×25, 64, 32, 4×25;LeakyReLU)-Conv(4, 4, 8)-LeakyReLU-Conv(4,4,1)-C.Dec`
`FOCA`	–	`MLP(4×25+64, 64, 32, 4×25;LeakyReLU)-Conv(4, 4, 8)-LeakyReLU-Conv(4,4,1)-C.Dec`

Table A.2 summarizes the network architectures. The green colored values indicate the dimension of context $\hat{c}$. We set the inner step K and step size α as 100/1 and 0.001/1 for `FOCA` and `CAVIA`, respectively, and τ as 0.1. For `CoDA`, we set the dimension of context as 2 (i.e., $|c| = 2$), following the default setting of Kirchmeyer et al. (2022).

Training details We train all models with mini-batches of size 512 for 1,000 epochs using Adam (Kingma & Ba, 2015) with the initial learning rate of 0.001 and pushforward regularization (Brandstetter et al., 2021). We use the past observations from the previous $2N$ to N steps as the input of the adaptation process.

Evaluation setting For in-training evaluations, we generate 2,048 mass-spring systems whose parameters $(k_1, k_2, k_3, m_1, m_2)$ are sampled from $\mathcal{U}(0.75, 1.25)^5$ and numerically solve mass-spring system with Runge–Kutta 45 with $T = 5.0$ and $\Delta t = 0.01$. For out-of-training evaluations, we generate 2,048 mass-spring systems whose parameters $(k_1, k_2, k_3, m_1, m_2)$ are sampled from $\mathcal{U}(0.60, 0.75)^5 \cup \mathcal{U}(1.25, 1.35)^5$ for $T = 5.0$ with $\Delta t = 0.01$. All models take the first and second 0.25 seconds of observation for the task adaptation and predict 4.5 seconds future states via the model rollout.

Additional results We provide the visualization of the model generalization errors to the change of m_1, m_2 on the mass-spring systems, as shown in Fig. A.2.

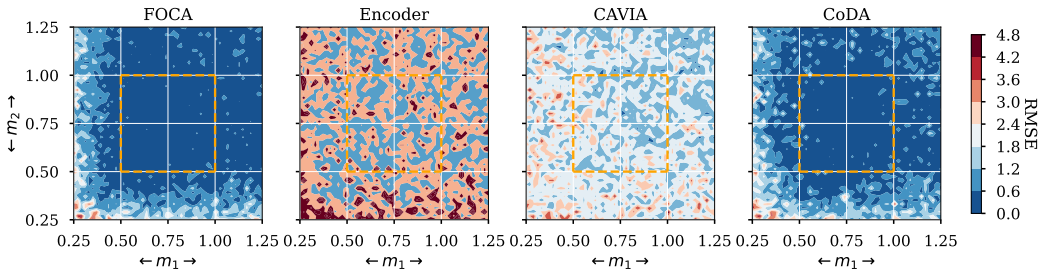

Figure A.2: In/out-of-distribution losses (in RMSE) on MS. Two parameters of MS (m_1, m_2) are generated from $[0.25, 1.25]^2$ while the other two parameters are fixed to 0.75. The orange box indicates the boundaries of the training parameter distribution.

C Details of Lotka-Volterra (LV) experiments

In this section, we provide the model architectures, training process, and additional experimental results for the Lokta-Volterra systems (Lotka, 1910).

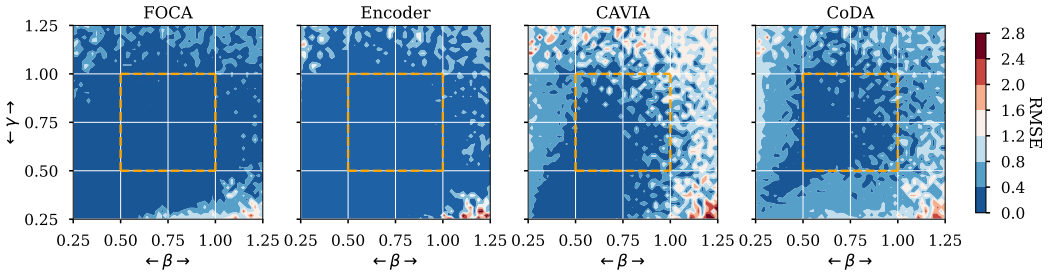

Figure A.3: In/out-of-distribution losses (in RMSE) on LV. Two parameters of LV (β, γ) are generated from $[0.25, 1.25]^2$ while the other two parameters are fixed to 0.5. The orange box indicates the boundaries of the training parameter distribution.

The Lokta-Volterra system describes the interaction between a prey-predator pair in an ecosystem, formalized into the following ODE:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \alpha x - \beta xy, \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= \delta xy - \gamma y, \end{aligned} \tag{A.2}$$

where x, y are respectively the quantity of the prey and the predator and $\alpha, \beta, \gamma, \delta$ define how two species interact.

Data generation For training data generation, we generate 128 Lokta-Volterra systems whose parameters $(\alpha, \delta) = (0.5, 0.5)$ and $(\beta, \gamma) \sim \mathcal{U}(0.5, 1.0)^2$ following Kirchmeyer et al. (2022). We then numerically solve the systems with Runge–Kutta 45 for $T = 50.0$ seconds with $\Delta t = 0.5$ second time intervals.

Model architecture We employ the same network architectures of the MS experiments except with a different input dimensions of 2 and bundling parameter $N = 20$.

Training details We train all models with mini-batches of size 128 for 5,000 epochs. Other details are kept the same as mass-spring experiments.

Evaluation setting For in-training evaluations, we generate 2,048 Lokta-Volterra systems similarly to the training data generation. For out-of-training evaluations, we generate 2,048 Lokta-Volterra systems whose parameters $(\alpha, \delta) = (0.5, 0.5)$ and $(\beta, \gamma) \sim \mathcal{U}(0.25, 0.5)^2 \cup \mathcal{U}(1.0, 1.25)^2$ for $T = 50.0$ with $\Delta t = 0.5$. All models take the first and second 10.0 seconds of observation for the task adaptation and predict 30.0 seconds future states via the model rollout.

D Details of Glycolytic oscillator (GO) experiments

In this section, we provide the model architectures, training process, and additional experimental results for the glycolytic oscillators (Daniels & Nemenman, 2015).

The glycolytic oscillators describe yeast glycolysis dynamics with the following ODE:

$$
\begin{aligned}
\frac{\mathrm{d}S_1}{\mathrm{d}t} &= J_0 - \frac{k_1 S_1 S_6}{1 + (1/K_1^q) S_6^q} \\
\frac{\mathrm{d}S_2}{\mathrm{d}t} &= 2\frac{k_1 S_1 S_6}{1 + (1/K_1^q) S_6^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5 \\
\frac{\mathrm{d}S_3}{\mathrm{d}t} &= k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6) \\
\frac{\mathrm{d}S_4}{\mathrm{d}t} &= k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7) \\
\frac{\mathrm{d}S_5}{\mathrm{d}t} &= k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5 \\
\frac{\mathrm{d}S_6}{\mathrm{d}t} &= -2\frac{k_1 S_1 S_6}{1 + (1/K_1^q) S_6^q} + 2k_3 S_3 (A - S_6) - k_5 S_6 \\
\frac{\mathrm{d}S_7}{\mathrm{d}t} &= \psi \kappa (S_4 - S_7) - k S_7,
\end{aligned}
\tag{A.3}
$$

where $S_1, S_2, S_3, S_4, S_5, S_6, S_7$ (states) represent the concentrations of 7 biochemical species and $J_0, k_1, k_2, k_3, k_4, k_5, k_6, K_1, q, N, A, \kappa, \psi$ and k are the parameters of the glycolytic oscillators.

Data generation For training data generation, we generate 128 glycolytic oscillators with the fixed parameters $J_0 = 2.5, k_2 = 6, k_3 = 16, k_4 = 100, k_5 = 1.28, k_6 = 12, q = 4, N = 1, A = 4, \kappa = 13, \psi = 0.1$ and $k = 1.8$ by sampling integer $k_1 \sim \mathcal{U}(80, 100)$ and $K_1 \sim \mathcal{U}(0.5, 1.0)$. We adopt the values or ranges of the parameters from Kirchmeyer et al. (2022). We then numerically solve the systems with Runge–Kutta 45 for $T = 5.0$ seconds with $\Delta t = 0.05$ second time intervals.

Model architecture We employ the same network architectures of the MS experiments except with a different input dimensions of 7 and bundling parameter $N = 10$. For `CoDA`, we set the context dimension as 3.

Training details We train all models with mini-batches of size 512 for 5,000 epochs. Other details are kept the same as mass-spring experiments.

Evaluation setting For in-training evaluations, we generate 2,048 glycolytic oscillators similarly to the training data generation. For out-of-training evaluations, we generate 2,048 glycolytic oscillators whose parameters $k_1 \sim \mathcal{U}(75, 80) \cup \mathcal{U}(100, 105)$ and $K_1 \sim \mathcal{U}(0.45, 0.5) \cup \mathcal{U}(1.00, 1.05)$ and $(J_0, k_2, k_3, k_4, k_5, k_5, k_6, q, N, A, \kappa, \psi, k) = (2.5, 6, 16, 100, 1.28, 12, 4, 1, 4, 13, 0.1, 1.8)$ for $T = 5.0$ with $\Delta t = 0.05$. All models take the first and second 0.5 seconds of observation for the task adaptation and predict 4.5 seconds future states via the model rollout.

Additional results We provide the visualization of the model generalization errors to the change of k_1, K_1 on the glycolytic oscillators, as shown in Fig. A.4.

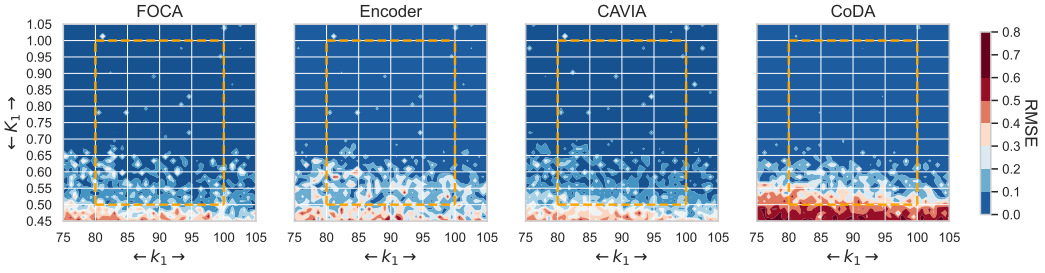

Figure A.4: In/out-of-distribution losses (in RMSE) on GO. The orange box indicates the boundaries of the training parameter distribution.

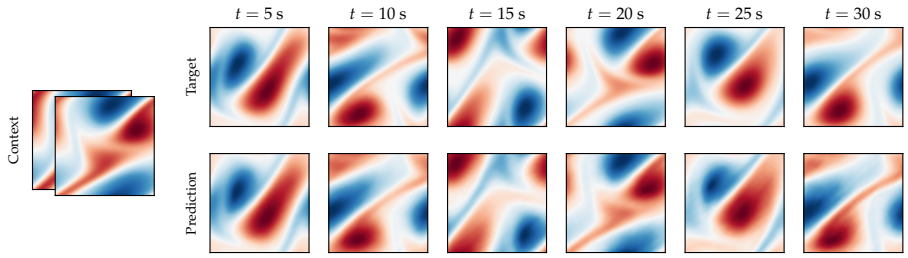

Figure A.5: Out-of-distribution generalization on the Navier-Stokes equations: FOCA is trained with viscosity $\nu \in [8 \times 10^{-4}, 1.2 \times 10^{-3}]$ and tested with $\nu = 6 \times 10^{-4}$. The model can reconstruct states even after long autoregressive rollouts in unseen conditions.

E Details of Navier-Stokes (NS) experiments

In this section, we provide the model architectures, training process, and additional experimental results for modeling the 2D Navier-Stokes equations (Stokes, 1851).

The Navier-Stokes equations describe the dynamics of incompressible flows with a two-dimensional PDE. In vorticity form they can be written as:

$$\begin{aligned} \frac{\partial w}{\partial t} &= -v\nabla w + \nu \Delta w + f \\ \nabla v &= 0 \\ w &= \nabla \times v \end{aligned} \tag{A.4}$$

where v is the velocity field and w is the vorticity, ν is the viscosity, and f is a forcing term. The domain is subject to periodic boundary conditions.

Data generation We generate trajectories with a temporal resolution of $\Delta t = 1$ and a time horizon of $t = 10$. The space is discretized on a 32×32 grid and we set $f(x, y) = 0.1(\sin(2\pi(x + y)) + \cos(2\pi(x + y)))$, where x, y are coordinates on the discretized domain (Yin et al., 2021). For training data, similarly to Kirchmeyer et al. (2022), we consider 5 training environments with $\nu \in \{8 \cdot 10^{-4}, 9 \cdot 10^{-4}, 1.0 \cdot 10^{-3}, 1.1 \cdot 10^{-3}, 1.2 \cdot 10^{-3}\}$ respectively. Each environment contains a total of 100 different initial conditions for a total of 500 training sequences.

Model architecture For this experiment, we employ the Fourier Neural Operator (FNO) (Li et al., 2021). Thanks to their layers in the spectral domain and frequency mode pruning, frequency domain models have been proven to be well suited to model complex dynamical systems characterized by a multitude of natural frequencies (Pathak et al., 2022; Poli et al., 2022). We employ an FNO model with 4 spectral convolution layers, hidden layers with width 10, and frequency mode pruning set to the 12 highest frequencies. Temporal bundling is set to $N = 1$. We set the context dimension of `CoDA` to 3. For models whose input includes the context, i.e. `Encoder`, `CAVIA` and `FOCA`, we keep the same context dimension $|\hat{c}| = 64$ as in the previous experiments; the inferred context passes through a linear layer and resized to the grid size of 32×32 as an additional input channel of the FNO.

Training details We train all models with mini-batches of size 16 for 100 epochs. Other details are kept the same as in the mass-spring experiments.

Evaluation setting For in-training adaptation, we consider 4 environments with viscosity $\nu \in \{8.5 \cdot 10^{-4}, 9.5 \cdot 10^{-4}, 1.05 \cdot 10^{-3}, 1.15 \cdot 10^{-3}\}$ while for out-of-distribution adaptation we generate data from 4 environments with unseen ranges of viscosity, i.e. $\nu \in \{7.0 \cdot 10^{-4}, 7.5 \cdot 10^{-4}, 1.25 \cdot 10^{-3}, 1.30 \cdot 10^{-3}\}$. Each adaptation environment is generated out of 30 different initial conditions. All the models take as input the first two steps (i.e. the first 2 seconds) and predict the rest of the sequence with autoregressive rollouts.

Additional results We provide in Fig. A.5 an additional qualitative visualization showing the out-of-distribution generalization on the Navier-Stokes equations. FOCA is trained with viscosity $\nu \in [8 \times 10^{-4}, 1.2 \times 10^{-3}]$ and tested with $\nu = 6 \times 10^{-4}$, demonstrating its adaptation capability in challenging and unseen settings.