Appendix for State Feedback Enhanced Graph Differential Equations for Multivariate Time Series Forecasting

Jiaxu Cui[1,2], **Qipeng Wang**[1,2], **Yiming Zhao**[1,2], **Bingyi Sun**[1,3], **Pengfei Wang**[4] and **Bo Yang**[1,2*]
[1]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, China
[2]College of Computer Science and Technology, Jilin University, China
[3]Public Computer Education and Research Center, Jilin University, China
[4]Computer Network Information Center, Chinese Academy of Sciences, China
cjx@jlu.edu.cn, {wangqp24,zhaoym24}@mails.jlu.edu.cn, bysun@jlu.edu.cn, pfwang@cnic.cn, ybo@jlu.edu.cn

A Observability guarantee of the system in Proposition 1

Before establishing the system's corresponding state feedback system in Proposition 1, we need to analyze its observability. Therefore, we provide the following proposition.

Proposition A1. *If the parameter matrix $\boldsymbol{W}_k$ is invertible, the system in Proposition 1 is completely observable.*

Proof. If a system can infer the state of all dimensions based on its output, then the system is completely observable. The output of the GDE in Proposition 1 is $\mathbf{H}_k^{out} = \mathbf{H}_k\mathbf{W}_k$, where $\mathbf{H}_k$ is the system state, $\mathbf{H}_k^{out}$ is its output, and $\mathbf{W}_k$ is the output matrix. We know that $\mathbf{W}_k$ is invertible. Therefore, we have $\mathbf{H}_k = \mathbf{H}_k^{out}\mathbf{W}_k^{-1}$. That is to say, the system state can be fully derived from the system output, indicating that the system in Proposition 1 is completely observable. □

The condition in Proposition A1 is relatively easy to guarantee in implementation empirically. We can ensure the invertibility by applying a shrinkage constraint to the parameter matrix $\mathbf{W}_k$, which is a commonly used technique in ridge regression [Kipruto and Sauerbrei, 2024]. Specifically, we constrain the $\mathbf{W}_k$ by using the l_2-norm during training, i.e., by setting weight decay to 1e-6.

B Derivation process

B.1 Derivation of the spatial GDE with state feedback

We define the nonlinear state feedback result after the *K*-th propagation as:

$$\boldsymbol{H}_K^S = \boldsymbol{H}_0^S + \int_0^K \hat{\mathbf{A}}\boldsymbol{H}_k^S - \boldsymbol{f}_{feedback}(\boldsymbol{H}_k^S)dk. \tag{B1}$$

Currently, commonly used numerical integration methods include the Euler methods, whose solution accuracy is usually first-order or second-order, the Runge-Kutta methods, whose solution accuracy are usually fourth-order, and so on. Since complex systems in the real world often do not have explicit governing equations and their sampling data often has noise and missing values, the use of the Runge-Kutta methods with high accuracy and complexity often leads to excessive computational overhead and low effectiveness. Since the Euler method has the advantages of simple implementation and fast calculation speed, by balancing the computational efficiency and approximate accuracy, we use the Euler trapezoidal integral formula with second-order accuracy to solve Eq. B1:

$$\boldsymbol{H}_{k+\Delta k}^S = \boldsymbol{H}_k^S + \tfrac{\Delta k}{2}[\boldsymbol{f}'(\boldsymbol{H}_k^S) + \boldsymbol{f}'(\boldsymbol{H}_{k+\Delta k}^S)]. \tag{B2}$$

Since Eq. B2 is the implicit Euler trapezoidal integration method, the term $\boldsymbol{H}_{k+\Delta k}^S$ cannot be obtained and therefore cannot be directly solved. Therefore, we estimate $\boldsymbol{H}_{k+\Delta k}^S$ by referring to the Predictor-Corrector method, which makes a preliminary estimate of the solution through the prediction step, and then further adjusts this estimate in the correction step to improve the accuracy of the solution. Specifically, we use the two-step Euler integral formula with high computational efficiency for prediction, and then use the Euler trapezoidal method for correction as:

$$\begin{aligned} &\textit{Predictor}: \\ &\quad \overline{\boldsymbol{H}}_{k+\Delta k}^S = \boldsymbol{H}_{k-\Delta k}^S + 2\Delta k\boldsymbol{f}'(\boldsymbol{H}_k^S) \\ &\textit{Corrector}: \\ &\quad \boldsymbol{H}_{k+\Delta k}^S = \boldsymbol{H}_k^S + \frac{\Delta k}{2}[\boldsymbol{f}'(\overline{\boldsymbol{H}}_{k+\Delta k}^S + \boldsymbol{f}'(\boldsymbol{H}_k^S)]. \end{aligned} \tag{B3}$$

Substituting *Predictor* into *Corrector*, we have the improved Euler integral formula as:

$$\boldsymbol{H}_{k+\Delta k}^S = \boldsymbol{H}_k^S + \frac{\Delta k}{2}[\boldsymbol{f}'(\boldsymbol{H}_k^S) + \boldsymbol{f}'(\boldsymbol{H}_{k-\Delta k}^S + 2\Delta k\boldsymbol{f}'(\boldsymbol{H}_k^S))]. \tag{B4}$$

Now, Eq. B1 can be solved iteratively through Eq. B4 to obtain the state after the *K*-th propagation:

$$\boldsymbol{H}_K^S = \boldsymbol{H}_{K-1}^S + \frac{1}{2}[\boldsymbol{f}'(\boldsymbol{H}_{K-1}^S) + \boldsymbol{f}'(\boldsymbol{H}_{K-2}^S + 2\boldsymbol{f}'(\boldsymbol{H}_{K-1}^S))]. \tag{B5}$$

B.2 Derivation of the temporal NDE

We revisit the temporal NDE as:

$$\boldsymbol{H}_t^{\mathrm{T}} = \boldsymbol{H}_0^{\mathrm{T}} + \int_0^t \boldsymbol{f}_T(\boldsymbol{H}_\tau^{\mathrm{T}})d\tau. \tag{B6}$$

*Corresponding Author

Similar to the derivation process of the spatial GDE, first use the implicit Euler trapezoidal method as:

$$\boldsymbol{H}_{t+\Delta t}^{T} = \boldsymbol{H}_{t}^{T} + \frac{\Delta t}{2}[\boldsymbol{f}_T(\boldsymbol{H}_{t}^{T}) + \boldsymbol{f}_T(\boldsymbol{H}_{t+\Delta t}^{T})]. \tag{B7}$$

And then the Predictor-Corrector method estimates the state $\boldsymbol{H}_{t+\Delta t}^{\mathrm{T}}$ and makes correction as:

$$\begin{aligned} &\textit{Predictor}: \\ &\quad \overline{\boldsymbol{H}}_{t+\Delta t}^{T} = \boldsymbol{H}_{t-\Delta t}^{T} + 2\Delta t \boldsymbol{f}_T(\boldsymbol{H}_{t}^{T}) \\ &\textit{Corrector}: \\ &\quad \boldsymbol{H}_{t+\Delta t}^{T} = \boldsymbol{H}_{t}^{T} + \frac{\Delta t}{2}[\boldsymbol{f}_T(\overline{\boldsymbol{H}}_{t+\Delta t}^{T} + \boldsymbol{f}_T(\boldsymbol{H}_{t}^{T})]. \end{aligned} \tag{B8}$$

Finally, Eq. B6 can be solved iteratively as:

$$\boldsymbol{H}_{t+\Delta t}^{\mathrm{T}} = \boldsymbol{H}_{t}^{\mathrm{T}} + \frac{\Delta t}{2}[\boldsymbol{f}_T(\boldsymbol{H}_{t}^{\mathrm{T}}) + \boldsymbol{f}_T(\boldsymbol{H}_{t-\Delta t}^{\mathrm{T}} + 2\Delta t \boldsymbol{f}_T(\boldsymbol{H}_{t}^{\mathrm{T}}))]. \tag{B9}$$

C Computational complexity analysis

We analyze the computational complexity of the SF-GDE in detail. The SF-GDE mainly consists of five steps, and we analyze the computational cost for each step. For Step 1, constructing a graph requires calculating the cosine similarity between each variable, so the computational workload needs to be $\mathcal{O}(d_e \times n^2)$, where d_e is the embedding dimension and n is the number of variables. For Step 2, the computational cost to obtain the initial spatial representations using historical time series is $\mathcal{O}(l \times d_h \times n + l \times n^2)$, where l is the length of the historical time series and d_h is the dimension of the hidden representation. Since n is usually greater than d_h, the main computational cost of this part is $\mathcal{O}(l \times n^2)$. For Step 3, we first analyze the cost for a single calculation of the differential equation in spatial GDE, and then analyze that of numerical integration approximation. As we know, in spatial GDE, regardless of linear or nonlinear implementation, the dominant term is $\hat{\mathbf{A}}\boldsymbol{H}_k$, so the computational complexity for one single calculation is $\mathcal{O}(d_h \times n^2)$. Thus, after K iterations, the consumption of approximate integration is $\mathcal{O}(K \times d_h \times n^2)$. In Step 4, the main cost for one single calculation of the differential equation in temporal NDE is mainly consumed on the long short-term memory (LSTM) layer, i.e., $\mathcal{O}(L \times d_h^2 \times n)$, where L is the sequence length. In order to obtain the hidden representations for the future m steps, an m-step iterative approximate integration is required, thus requiring a computational cost of $\mathcal{O}(m \times L \times d_h^2 \times n)$. In Step 5, the main cost during the decoding process is LSTM, i.e., $\mathcal{O}(m \times d_h^2 \times n)$. To sum up, we can obtain the overall time complexity of the SF-GDE is $\mathcal{O}((d_e + l + Kd_h)n^2 + Ld_h^2 mn)$.

For analysis of overfitting risk, we add the comparison of the number of parameters for different methods, as shown in Table C1. Overfitting occurs more often when the amount of data is insufficient and the number of trainable parameters is large. From Table C1, among all the 10 algorithms, our SF-GDE model has the fourth fewest number of parameters, so SF-GDE may face a lower risk of overfitting. At the same time, the overfitting problem is a thorny issue facing deep learning models. There are many techniques that can alleviate overfitting to a certain extent during training, such as weight decay and dropout. In fact, we have used the above techniques to reduce the risk of overfitting during training. In addition, the Lyapunov stability constraint we proposed, as a regularization term that limits the stability of the dynamic system, can also be regarded as an operation to avoid overfitting. The experimental results also show the effectiveness of using them.

Model	# Parameter	Model	# Parameter
LSTNet	6089	STDDE	102982
Graph WaveNet	2777702	Autoformer	1443699
STGCN	98902	Crossformer	44199940
STGODE	244210	iTransformer	4746
TG-ODE	3160	SF-GDE(ours)	10198

Table C1: The number of parameters.

D Detailed introduction of experimental settings, datasets, and baselines

D.1 Experimental settings

All experiments were conducted on the hardware environment with the same computational resources, including Intel(R) Core(TM) i7-12800HX CPU and NVIDIA GeForce RTX 3080Ti.

All parameters in the SF-GDE model are set as follows: the number of variables (n) is determined by the dataset, the embedding dimension d_e=10, the length of the input historical time series l=5, the dimension of the hidden representation d_h=16, the number of graph propagations K=10 and the predictive steps m=10.

The neural networks involved in the SF-GDE include: $\boldsymbol{f}_{feedback}$, $\boldsymbol{f}^T$, and $\boldsymbol{f}_{De}$. We provide their specific architectures below. $\boldsymbol{f}_{feedback}$ is a multi-layer perceptron with two hidden layers and a Tanh activation function. $\boldsymbol{f}^T$ is an LSTM layer and a Linear Layer with Tanh activation. $\boldsymbol{f}_{De}$ is still a LSTM layer and a linear layer. If not explicitly stated, the hidden layer dimensions of the above neural networks are set to 16.

During training, we set the batch size to 64 and chose the Adam optimizer with an initial learning rate of 0.001 to learn the weights in SF-GDE. The hyperparameters β and ξ in the loss function are set to 0.01 and 0.1, and the training epoch is 2000. All datasets are divided into training, validation, and testing sets according to the ratio of 6:2:2.

D.2 Datasets

There are four real-world datasets: the Seoul $PM_{2.5}$ dataset[1], the PEMS04 traffic flow dataset [Song *et al.*, 2020], Beijing temperature dataset[2], and Traffic dataset [Wu *et al.*, 2021]. Specifically, the Seoul $PM_{2.5}$ dataset contains $PM_{2.5}$ concentration data collected from 25 locations ($i.\ e$ the number of nodes is n=25) from 2017 to 2019. The data collection interval is 1 hour, so the data format is $[n, 24 \times 365 \times 2]$. The PEMS04 traffic flow dataset recorded traffic flow every

[1] www.kaggle.com/datasets/bappekim/air-pollution-in-seoul

[2] https://biendata.com/competition/kdd2018

5 minutes at 307 locations, so a total of 59 days were collected. The size of the dataset is $[307, 288 \times 59]$. For the Beijing temperature dataset, it collects the temperature data from 18 regions in Beijing, China, from January 30, 2017 to January 30, 2018, with a sampling frequency of once an hour. The dataset format is $[18, 24 \times 366]$. The Traffic dataset has 862 locations over 61 days, with a 5-minute interval between records, so the format of the data is $[862, 288 \times 61]$. Table D2 lists the information of datasets.

Attribute	Seoul $PM_{2.5}$	PEMS04	Beijing Temperature	Traffic
Field	Air quality	Traffic	Temperature	Traffic
#Variables	25	307	18	862
Length	24×365×2	288×59	24×366	288×61

Table D2: Datasets information.

D.3 Baselines

The comparison methods are described in detail as follows:
LSTNet [Lai *et al.*, 2018]: Using CNN to capture the local spatial information while using LSTM or GRU to capture long-term macro information.
Graph WaveNet [Wu *et al.*, 2019]: Spatio-temporal features are captured by stacking multiple GCN and Temporal Convolution Network (TCN) blocks with residual connection and nonlinear activation.
STGCN [Song *et al.*, 2020]: Combining the graph convolution and residual connection to model spatial information and using the improved gated linear unit (GLU) to model temporal dependence.
Autoformer [Wu *et al.*, 2021]: Decomposing predictable components through the deep decomposition architecture and capturing the temporal features using the auto-correlation mechanism.
Crossformer [Zhang and Yan, 2023]: Crossformer improves the attention mechanism so that it can pay attention to the interaction between different time series and achieve spatio-temporal prediction.
iTransformer [Liu *et al.*, 2024]: Feedforward network is used to capture the temporal features, and the attention mechanism captures the interactions between sequences to achieve spatio-temporal time series prediction.
STGODE [Fang *et al.*, 2021]: The graph ODE algorithm is combined with the temporal convolution algorithm to model the spatio-temporal characteristics in multivariate time series.
TG-ODE [Gravina *et al.*, 2024]: The multi-kernel convolution algorithm TAGCN is combined with multi-layer linear layers to model multivariate time series based on flexible receptive fields.
STDDE [Long *et al.*, 2024]: The delay effect is added to the neural ODE to model the dynamic evolution in traffic flow.
T-Mixer [Wang *et al.*, 2024]: High-precision time series forecasting is achieved by hybrid multi-scale time series decomposition and adaptive frequency domain analysis.
M-TCN [Luo and Wang, 2024]: Modern temporal convolutional networks based on deep dilated convolutions and residual connections for efficient modeling of long-term dependencies.
P-former [Chen *et al.*, 2024]: Combining the multi-path attention mechanism with the temporal pyramid structure, a multivariate prediction model that captures complex temporal patterns.

E More results

E.1 Full performance comparison

Figure D1 shows the error comparison of various methods for predicting the next 1 to 10 steps on three datasets, demonstrating the superiority of our SF-GDEs, especially for medium and long-term forecasting.

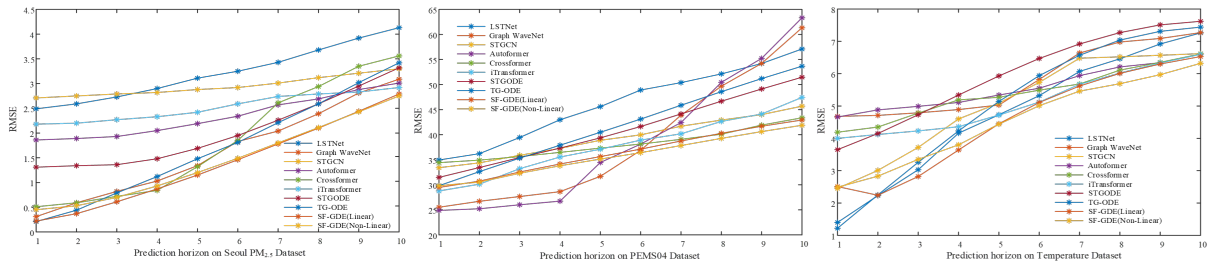
Note that, the linear control term (SF-GDE_l) and nonlinear control term (SF-GDE_{nl}) are applicable to different scenarios. Linear control term shows better prediction results in short-term predictions with less reliance on complex nonlinear characteristics, while nonlinear control term has more advantageous in medium and long-term predictions, that is, the longer the prediction time steps, the more prominent the effect of nonlinear control term. At the same time, as can be seen from the enlarged sub-figure in Figure 3 in the main text and Figure D1, the longer the prediction step, the more local prediction deviations appear in the linear control term. In other words, the trends of some predictions will be opposite to the actual situation.

At the same time, the prediction accuracy between the two control terms is also related to the dataset. If the dataset has more nonlinear features, prediction accuracy of the nonlinear control item will be better than that of the linear control item. For example, the Seoul $PM_{2.5}$ dataset is a smoothed time series, so the nonlinear characteristics contained in it are significantly weaker than those of the unprocessed PEMS04 and temperature datasets. Therefore, the nonlinear control term is better than the linear control item only in the 10-th step prediction, while for the other two datasets, it is significantly better than the linear control term in the 7-th step prediction.

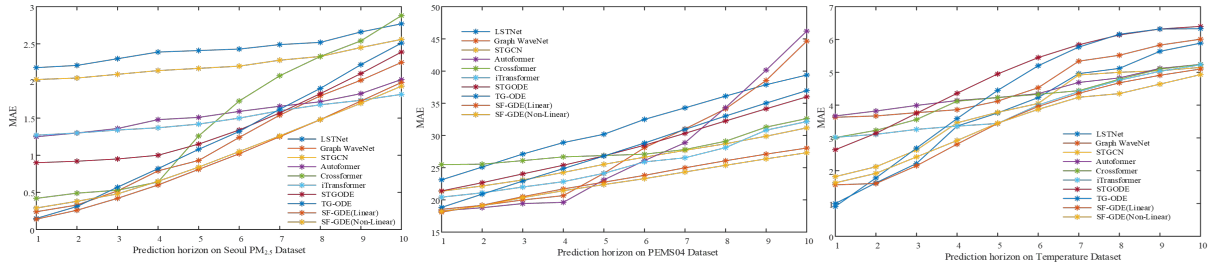
In summary, when making short-term predictions on datasets with weaker nonlinear characteristics, it is recommended to use linear control term with better theoretical support, while in long-term prediction tasks in real scenarios that are highly dependent on nonlinear characteristics, nonlinear control term with higher accuracy should be selected.

E.2 More comparative experimental results

Table E3 shows the results of using 5 historical steps to predict the next 1 step and 10 steps on the Seoul $PM_{2.5}$ dataset. We can obviously see that our models are still competitive, demonstrating the effectiveness of our mechanism. To verify that our method still has certain advantages in long-term prediction, we set the experiment to predict the next 96 steps. As shown in Table E4, the non-linear feedback mechanism demonstrates promising performance, particularly because long-term forecasting tasks often exhibit complex nonlinear relationships. This finding aligns with the conclusions drawn in the paper.



(a) RMSE



(b) MAE

Figure D1: Step-wise comparison results of predictive errors on Seoul $PM_{2.5}$ dataset, PEMS04 dataset and Temperature dataset.

Steps	T-Mixer	M-TCN	P-former	SF-GDE$_l$	SF-GDE$_{nl}$
1	0.50	0.28	1.38	**0.22**	0.45
10	3.48	3.16	4.09	2.79	**2.75**

Table E3: Root mean square error (RMSE) of 1,10-step ahead predictions on Seoul $PM_{2.5}$ Dataset. The best results are bolded.

Steps	T-Mixer	M-TCN	P-former	SF-GDE$_l$	SF-GDE$_{nl}$
96	12.65	13.72	14.07	14.25	**11.76**

Table E4: Comparison of root mean square error (RMSE) of 96-step ahead prediction on Seoul $PM_{2.5}$ Dataset. The best result is bolded.

E.3 The comparison of node feature similarity

To provide a clearer and more supportive proof that the proposed SF-GDE is more effective than traditional methods, we conduct a feature over-smoothing comparison experiment of traditional methods on the PEMS04 dataset. Specifically, we use different methods to perform 7 times of message passing. Secondly, the cosine similarity is calculated between the feature vectors of each node obtained after the 7-th passing to obtain a similarity matrix of shape $[N, N]$, where N is the number of nodes, where, $[i, j] = [j, i]$ represents the cosine similarity between the i-th node and the j-th node. The greater the similarity, the more similar the node features are, that is, the greater the risk of feature over-smoothing.

In order to more intuitively understand the effect of each method on the over-smoothing problem, we quantify the results and report the average similarity of all nodes by taking the average value of all elements in the matrix to compare the occurrence of over-smooth from different methods. Observing Table E5, the traditional methods, such as adding residual connections, regularization and normalization, have achieved

Model	Original feature	Residual connection	Regularization
Similarity	0.7191	0.9225	0.9225
Model	Normalization	SF-GDE$_l$(ours)	SF-GDE$_{nl}$(ours)
Similarity	0.9228	0.7432	0.7584

Table E5: The average similarity of node features after 7 message passing.

feature similarity of 0.92 or above, which means that the features of all nodes are 92% like each other, resulting in serious redundancy. By comparison, our methods do not produce such similarity, preserving the diversity of features among each node.

E.4 Validation on noise and missing data

Although real-world datasets we tested in this work may contain measurement noise and missing data, in order to demonstrate our method's tolerance for noise and missing data more clearly, we test the methods using data generated by the mathematical dynamical system model, i.e., Lorenz system [Han and Wang, 2024]. Lorenz 96 dataset is a lightweight dataset generated from an atmospheric model with 40 variables and 1,000 time steps per variable. Specifically, we add Gaussian noise to the data, with the level of noise varying, *i.e.*, data with noise level σ : $data + \sigma \times \xi$, where ξ is sampled from a standard normal distribution. For missing data, we randomly delete a portion of the data to simulate situations where data acquisition fails. Note that in order to demonstrate the effectiveness and superiority of the proposed SF-GDE method in modeling data with noise, incompleteness, and larger scale, we select one from each of the current mainstream prediction models: GNN-based, GDE-based, and transformer-based methods for comparison. Table E6 shows the comparison results. As shown in Table E6, with the increase of noise level, all methods show a trend of

Model	Complete and clean data	10% noise	20% noise	30% noise	10% miss	20% miss	30% miss
STGCN	0.2647	0.5030	0.8700	1.0092	0.2747	0.4334	0.5093
TG-ODE	1.1494	1.7595	1.9440	1.7195	1.1627	1.1704	1.1859
Autoformer	0.1911	0.5071	0.6522	1.0706	0.3648	0.7487	0.8228
SF-GDE$_l$(ours)	<u>0.0591</u>	<u>0.0648</u>	<u>0.0807</u>	<u>0.0834</u>	<u>0.0679</u>	<u>0.0905</u>	<u>0.0925</u>
SF-GDE$_{nl}$(ours)	**0.0581**	**0.0621**	**0.0775**	**0.0729**	**0.0633**	**0.0718**	**0.0725**

Table E6: Predictive error (MAE) on the Lorenz dataset with noise and missing data at 10 steps ahead. The best results are bolded, while the second-best results are underlined.

performance deterioration, but our methods significantly outperform others, especially SF-GDE with non-linear feedback performs the best, which demonstrates our method's stronger noise tolerance. For missing data, the methods also exhibit similar performance. This demonstrates the effectiveness of our method in generalizing scenarios with measurement noise and missing data.

E.5 Boosting more nonlinear GDEs

Table E7 lists the full comparison results of predictive errors from nonlinear GDEs boosted by state feedback for multivariate time series forecasting on the PEMS04 dataset. Figure E2 shows the cosine similarity matrix of node features after 1, 3, 7 message passing steps on the PEMS04 traffic dataset. Observations are that 1) the introduction of feedback mechanisms can indeed enhance the performance of nonlinear GDEs, including GCN and GIN; 2) the feedback mechanism can adaptively adjust the representations towards the desired performance direction, thereby fundamentally avoiding over-smoothing.

References

[Chen *et al.*, 2024] Peng Chen, Yingying ZHANG, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *ICLR*, 2024.

[Fang *et al.*, 2021] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ode networks for traffic flow forecasting. In *ACM SIGKDD*, pages 364–373, 2021.

[Gravina *et al.*, 2024] Alessio Gravina, Daniele Zambon, Davide Bacciu, and Cesare Alippi. Temporal graph odes for irregularly-sampled time series. In *IJCAI*, 2024.

[Han and Wang, 2024] Min Han and Qipeng Wang. Adaptive graph convolution neural differential equation for spatio-temporal time series prediction. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[Kipruto and Sauerbrei, 2024] Edwin Kipruto and Willi Sauerbrei. Post-estimation shrinkage in full and selected linear regression models in low-dimensional data revisited. *Biometrical Journal*, 66(7):e202300368, 2024.

[Lai *et al.*, 2018] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*, pages 95–104, 2018.

[Liu *et al.*, 2024] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *ICLR*, pages 1–12, 2024.

[Long *et al.*, 2024] Qingqing Long, Zheng Fang, Chen Fang, Chong Chen, Pengfei Wang, and Yuanchun Zhou. Unveiling delay effects in traffic forecasting: A perspective from spatial-temporal delay differential equations. In *WWW*, pages 1035–1044, 2024.

[Luo and Wang, 2024] Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. In *ICLR*, pages 1–43, 2024.

[Song *et al.*, 2020] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *AAAI*, volume 34, pages 914–921, 2020.

[Wang *et al.*, 2024] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. Timemixer: Decomposable multiscale mixing for time series forecasting. In *ICLR*, 2024.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, pages 1907–1913, 2019.

[Wu *et al.*, 2021] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, pages 22419–22430, 2021.

[Zhang and Yan, 2023] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *ICLR*, pages 1–12, 2023.

Models	Metrics	1	2	3	4	5	6	7	8	9	10
GCN	RMSE	159.03	159.03	159.03	159.03	159.02	159.02	159.01	159.00	158.99	158.98
	MAE	130.10	130.10	130.09	130.09	130.08	130.06	130.05	130.03	130.02	130.01
GDE_{GCN}	RMSE	30.20	32.44	34.53	36.49	38.51	40.69	43.06	45.37	47.48	49.41
	MAE	19.36	20.90	22.43	23.86	25.33	26.94	28.68	30.41	32.03	33.55
$\text{SF-GDE}_{\text{GCN}_l}$	RMSE	30.40	32.23	34.04	35.80	37.71	39.84	42.26	44.66	46.93	49.13
	MAE	19.82	20.87	22.06	23.27	24.60	26.14	27.87	29.66	31.41	33.15
$\text{SF-GDE}_{\text{GCN}_{nl}}$	RMSE	**29.57**	**31.71**	**33.76**	**35.63**	**37.57**	**39.63**	**41.88**	**44.08**	**46.06**	**47.84**
	MAE	**18.76**	**20.29**	**21.79**	**23.18**	**24.59**	**26.10**	**27.75**	**29.38**	**30.88**	**32.26**
GIN	RMSE	30.44	32.75	35.04	37.19	39.40	41.49	44.17	46.60	48.87	51.00
	MAE	19.60	21.22	22.89	24.48	26.13	27.85	29.75	31.63	33.44	35.21
GDE_{GIN}	RMSE	29.43	31.98	34.44	36.77	39.07	41.45	43.95	46.37	48.61	50.66
	MAE	18.54	20.46	22.36	24.16	25.88	27.71	29.59	31.50	33.25	34.94
$\text{SF-GDE}_{\text{GIN}_l}$	RMSE	29.51	32.04	34.48	36.73	39.01	**41.31**	43.88	**46.23**	**48.50**	**50.39**
	MAE	18.76	20.50	22.30	**23.92**	**25.59**	**27.27**	**29.15**	**30.94**	**32.63**	**34.17**
$\text{SF-GDE}_{\text{GIN}_{nl}}$	RMSE	**29.35**	**31.89**	**34.36**	**36.67**	**38.99**	41.36	**43.87**	46.30	48.52	50.55
	MAE	**18.45**	**20.32**	**22.19**	23.93	25.68	27.45	29.36	31.22	32.98	34.60

Table E7: Full comparison results of predictive errors from nonlinear GDEs boosted by state feedback for multivariate time series forecasting on the PEMS04 dataset. The best results are bolded, while the second-best results are underlined.

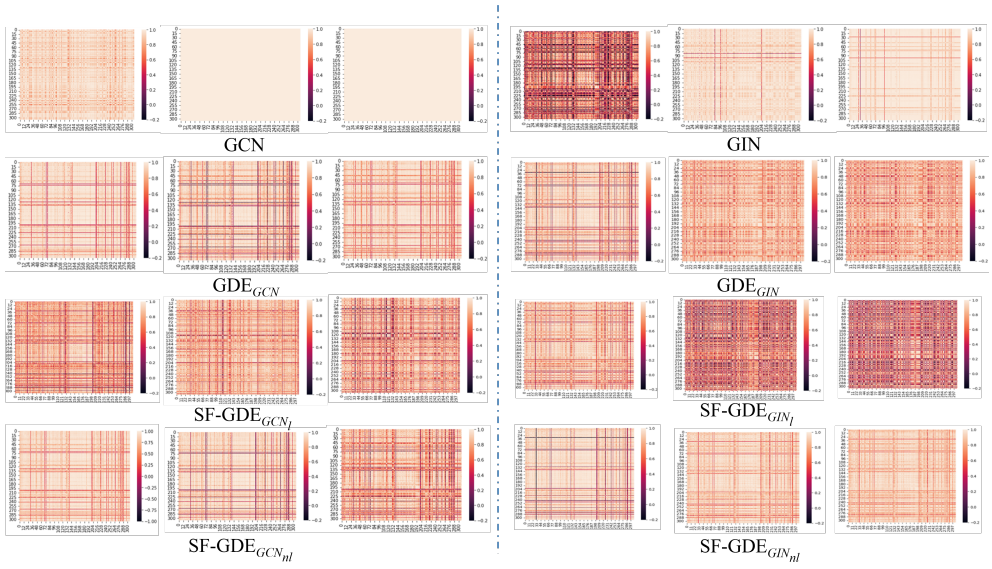


Figure E2: The cosine similarity matrix of node features after 1, 3, 7 message passing steps on the PEMS04 traffic dataset.