INTERPOLATION FOR IRREGULARLY SAMPLED SEQUENCE WITH SMOOTHING COMPENSATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Combining recurrent neural networks and neural ordinary differential networks is effective to model irregularly sampled sequence. However, such combination (ODE-RNN) has continuous hidden state among observations but a jump at the observation point, triggering the discontinuity for interpolation. To smooth the discontinuity, we propose the *cubic spline smoothing compensation*, which is a stand along module upon either the output or the hidden state of ODE-RNN and can be trained ent-to-end. We derive the optimal solution for such compensation and provide its theoretical interpolation error bound. Extensive experiments indicates its merits over both ODE-RNN and cubic spline interpolation.

1 INTRODUCTION

Recurrent neural networks (RNNs) are commonly used for modeling regularly sampled time series. However, irregularly sampled time series in the domain such as healthcare (Rajkomar et al., 2018) and business () fails to be modeled by standard RNN because it can only process discrete series without considering the unequal temporal interval between sample points. While some works adapt RNN to the irregular scenario, they often assumes a exponential decay (either at output or hidden state) between two observations (Che et al., 2018; Cao et al., 2018), which may not always hold.

Recently, the neural ordinary differential equation (ODE) (Chen et al., 2018) has been proposed to parametrize the ODE as a learnable neural network, with which the hidden state dynamics during the observation interval could be continuously modelled and thus is free from the exponential decay assumption. Since an ODE determines the trajectory given its initial state, a common design to leverage the observations is to update the hidden state at the observation using an RNN, and evolve the hidden state using an ODE during the time gap (Rubanova et al., 2019; De Brouwer et al., 2019). Such design combines the continuous (ODE) and discrete (RNN) modeling, triggering a jump at the observation point, and thus jeopardizing the modeling of continuous time series, especially the interpolation task.

We propose a Cubic Spline Smoothing Compensation (CSSC) technique that can address such jump problem and is especially suitable for continuous time series interpolation. CSSC employs the cubic spline function as a compensation to eliminate the jump either at hidden space or output space. While the latent ODE (Rubanova et al., 2019) with an encoder-decoder structure can also produce continuous interpolation, CSSC can even ensure the interpolated curve pass strictly through the observation points. Extensive experiments demonstrate the effect of CSSC.

2 RELATED WORKS

Spline interpolation is an engineer way to construct smooth curves using a number of points (De Boor et al., 1978), even for unequally spaced poins. Cubic spline interpolation leverage the piecewise third order polynomials to prevent from the Runge's phenomenon (Runge, 1901), and is applied as a classical way to missing data imputing (Che et al., 2018).

Recent literatures focus on RNNs to model the irregularly sampled time series, given its strong modeling ability. Since standard RNNs can only process discrete series without considering the unequal temporal interval between sample points, different improvements are proposed. One solution is to augment input with the observation mask or concatenate the time lag Δt to expect the network

would make use of Δt in an arbitrary manner (Lipton et al., 2016; Mozer et al., 2017). While such flexible structure can already achieve good performance under some circumstance (Mozer et al., 2017), more researches advocate certain prior knowledge for missing data imputation. GRU-D (Che et al., 2018) impute the missing value with the weighted sum of exponential decay of the previous observation and the empirical mean. Shukla & Marlin (2019) employ the radial basis function kernel to construct interpolation network. Cao et al. (2018); Tan et al. (2020) assume hidden state exponential decay for non-observed time points.

Another track is probabilistic generative model. Gaussian processes (GPs) are adopted for missing data imputing (Futoma et al., 2017; Tan et al., 2020; Moor et al., 2019) before RNN or CNN structure, but need to manually set covariance function. Neural processes (Garnelo et al., 2018) eliminate such constraint by introducing a global latent variable that represents the whole process. Generative adversarial networks is also adopted for imputing (Luo et al., 2018).

Most of the aforementioned works have the assumption of the exponential decay for the latent state during the time lag between observations. To eliminate this, neural ODEs (Chen et al., 2018) utilize a continuous state transfer function parameterized by a neural network to learn the dynamics of latent state. Rubanova et al. (2019) combine the RNN and ODE together to balance both the observation and latent state evolution between observations. De Brouwer et al. (2019) modify the ODE into a GRU structure on continuous time. However, ODEs model a continuous change of hidden state but RNN will introduce a jump of the hidden state, leading to discontinuous hidden state along the trajectory, which fail to model the time series with continuous change. Our method can address such discontinuous issue.

3 METHODS

3.1 PROBLEM DEFINITION

The task is interpolation. Given a set of n observations $\Omega = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ at the irregularly spaced time points $\Pi : a = t_0 < t_1 < ... < t_n = b$, the goal is to learn a function $F : [a, b] \to \mathbb{R}^d$ such that $F(t_i) = \mathbf{x}_i$.

3.2 BACKGROUND OF ODE-RNN

[Jing: The time series modeled by Neural ODE is decided given initial states. However, interpolation sporadical time series using neural ODE requires the ability to take in the inputs along the time.] ODE-RNN (Rubanova et al., 2019) achieve the interpolation using ODE and RNN alternatively through a time series. The function on scope $t \in [t_i, t_{i+1})$ is described using ODE with the initial state $\mathbf{h}(t_i)$:

$$\dot{\mathbf{h}}(t) = f(\mathbf{h}(t)); \tag{1}$$

$$\mathbf{o}(t) = g(\mathbf{h}(t)), \tag{2}$$

where the hidden state $\mathbf{h} \in \mathbb{R}^m$, $\dot{\mathbf{h}} = \frac{\mathrm{d}\mathbf{h}}{\mathrm{d}t}$, the output $\mathbf{o} \in \mathbb{R}^d$, the transfer function $f : \mathbb{R}^m \to \mathbb{R}^m$ and the output function $g : \mathbb{R}^m \to \mathbb{R}^d$. At time $t = t_i$, the hidden state will be updated by RNN as:

$$\mathbf{h}(t_i) = \mathrm{RNNCell}(\mathbf{h}(t_i^-), \mathbf{x}_i); \tag{3}$$

$$\mathbf{o}(t_i) = g(\mathbf{h}(t_i)), \tag{4}$$

where the input $\mathbf{x} \in \mathbb{R}^d$, t_i^- and t_i^+ are the left and right-hand limits of t_i. The above formulation has two downsides. The first is the discontinuity problem. While the function described by ODE is right continuous $\mathbf{o}(t_i) = \mathbf{o}(t_i^+)$, the RNN cell in Eq. (3) renders the hidden state discontinuity $\mathbf{h}(t_i^-) \neq \mathbf{h}(t_i^+)$ and therefore output discontinuity $\mathbf{o}(t_i^-) \neq \mathbf{o}(t_i^+)$. The second is that the model cannot guarantee $\mathbf{o}(t_i) = \mathbf{x}_i$ without explicit constraints.

3.3 CUBIC SPLINE SMOOTHING COMPENSATION

To remedy the two downsides, we propose a compensated output $\hat{\mathbf{o}}(t)$ as:

$$\hat{\mathbf{o}}(t) = \mathbf{c}(t) + \mathbf{o}(t), \tag{5}$$

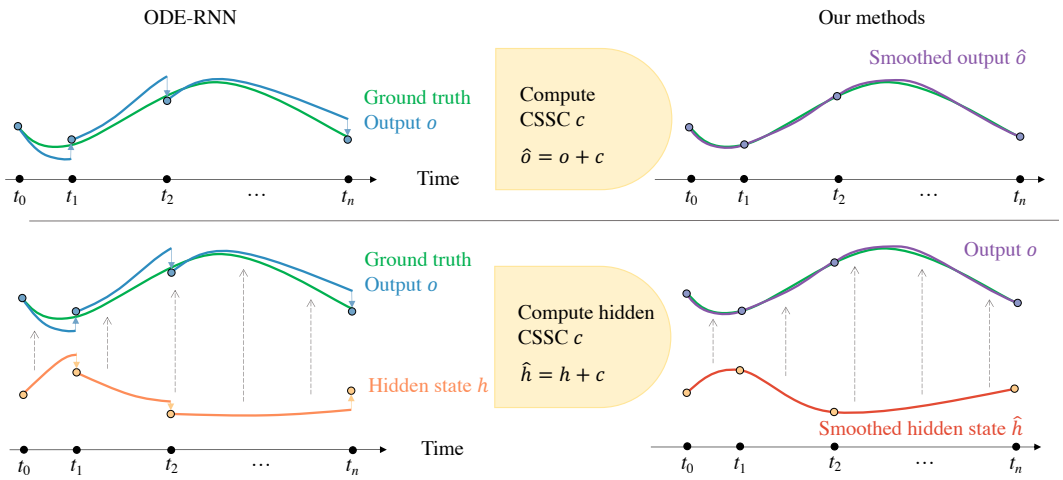


Figure 1: The difference between ODE-RNN and our methods. The left is ODE-RNN, where each

where $\mathbf{o}(t)$ is the ODE-RNN output, and the $\mathbf{c}(t)$ is a compensation composed of piecewise cubic functions. To simplify the notation, we represent the j th entry of $\mathbf{o}, \mathbf{c}, \hat{\mathbf{o}}, \mathbf{x}$ as $o, c, \hat{o}, x$, respectively. The $c(t)$ can be represented in pieces as $c_k(t)$ at domain $[t_k, t_{k+1})$, $k = 0, 1, ..., n-1$. [Jing: process annotation i and k.] Here are four constraints applied to $\hat{o}(t)$:

1. $\hat{o}(t_k^-) = \hat{o}(t_k^+) = x_k$ for $k = 1, ..., n-1$, $\hat{o}(t_0) = x_0$ and $\hat{o}(t_n) = x_n$.
2. $\dot{\hat{o}}(t_k^-) = \dot{\hat{o}}(t_k^+)$, $k = 1, ..., n-1$
3. $\ddot{\hat{o}}(t_k^-) = \ddot{\hat{o}}(t_k^+)$, $k = 1, ..., n-1$
4. $\ddot{\hat{o}}(t_0) = \ddot{\hat{o}}(t_n) = 0$.

The constraint 1 ensures the interpolation curve passing through the observation and the continuity at the observation points. Constraint 2 and 3 enforce the first and second order continuity at the observation points, which usually hold when the real curve is smooth. And constraint 4 specifies the natural boundary. [Jing: Why using natural boundary?] Given o and such four constraints, we can derive a unique solution of c.

Theorem 1. *Given the first order and second order jump difference of ODE-RNN as*

$$\dot{r}_k = \dot{o}(t_k^+) - \dot{o}(t_k^-); \tag{6}$$

$$\ddot{r}_k = \ddot{o}(t_k^+) - \ddot{o}(t_k^-). \tag{7}$$

where $\dot{o} = \frac{\partial g}{\partial \mathbf{h}} f$, $\ddot{o} = f^\intercal \frac{\partial^2 g}{\partial \mathbf{h}^2} f + \frac{\partial g}{\partial \mathbf{h}}^\intercal \frac{\partial f}{\partial \mathbf{h}} f$, *and the error defined as*

$$\epsilon_k^+ = x_k - o(t_k^+); \tag{8}$$

$$\epsilon_k^- = x_k - o(t_k^-), \tag{9}$$

the c_k *is uniquely determined as*

$$c_k(t) = \frac{M_{k+1} + \ddot{r}_{k+1} - M_k}{6\tau_k}(t - t_k)^3 + \frac{M_k}{2}(t - t_k)^2 + \left(\frac{\epsilon_{k+1}^- - \epsilon_k^+}{\tau_k} - \frac{\tau_k(M_{k+1} + \ddot{r}_{k+1} + 2M_k)}{6}\right)(t - t_k) + \epsilon_k^+. \tag{10}$$

M_k *is obtained as:*

$$\mathbf{M} = A^{-1}\mathbf{d}. \tag{11}$$

where

$$A = \begin{pmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{pmatrix}, \mathbf{M} = \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{pmatrix}, \mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix}, \tag{12}$$

$\tau_k = t_{k+1} - t_k$, $\mu_k = \frac{\tau_{k-1}}{\tau_{k-1}+\tau_k}$, $\lambda_k = \frac{\tau_k}{\tau_{k-1}+\tau_k}$, $d_k = 6\frac{\epsilon[t_k^+, t_{k+1}^-] - \epsilon[t_{k-1}^+, t_k^-]}{\tau_{k-1}+\tau_k} + \frac{6\dot{r}_k - 2\ddot{r}_k\tau_{k-1} - \ddot{r}_{k+1}\tau_k}{\tau_{k-1}+\tau_k}$, $\epsilon[t_k^+, t_{k+1}^-] = \frac{\epsilon_{k+1}^- - \epsilon_k^+}{\tau_k}$, $M_0 = M_n = 0$.

The proof for Thm. 1 is in Appx. A. Thm. 1 reveals that the computation of c is channel-wise independent, hence $\mathbf{c}$ can be obtained by computing c at every channel independently.

3.4 Inference and Training

For inference, firstly run ODE-RNN (Eq. (1-4)) for $\mathbf{o}$ and then compute cubic SSC $\mathbf{c}$ (Eq. (10)) thus yielding $\hat{\mathbf{o}}$ (Eq. 5). For training, the cubic SSC is a standalone nonparametric module on top of the ODE-RNN, keeping the end-to-end training fashion for ODE-RNN parameters. We employ Mean Squire Error (MSE) loss at the smoothed interpolated points $\hat{\mathbf{o}}$. In addition, we expect that the compensation term $\mathbf{c}$ is small, allowing the ODE-RNN output $\mathbf{o}$ take the leading role in order to fully utilize its model capacity. Therefore we consider a 2-norm of the compensation as penalty and construct the final loss as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(||\mathbf{x}(t_i) - \hat{\mathbf{o}}(t_i)||^2 + \alpha||\mathbf{c}(t_i)||^2) \tag{13}$$

We show in ablation study 4.6 that the balance weight α can allocate the contribution from ODE-RNN and cubic SSC.

Gradient flow. The gradient can flow from the compensation term $\mathbf{c}$ into the ODE-RNN, since $\mathbf{c}(t)$ depends on the left and right limit of $\mathbf{o}(t_k), \dot{\mathbf{o}}(t_k), \ddot{\mathbf{o}}(t_k)$, hence applying the penalty on $\mathbf{c}(t_i)$ is feasible.

3.5 Extend to interpolate hidden state

Cubic SSC have to calculate each dimension of output space, which might not scale well when the input dimension is high, e.g., video interpolation. More importantly, some time series may not have continuous property at each dimension, but have the continuity in its lower dimensional manifold. For example, each pixel of a video does not change continuously, but the spatial movement of the video content could be continuous. Thus we extend the cubic SSC to work the hidden states, which can have a fixed number of dimension regardless the output size, and allow the discontinuity of output space controlled by its continuous change in the hidden space. [Jing: Briefly Compare the spatial complexity of the output size]. Hence, the Eq. (5) is modified to $\hat{\mathbf{h}}(t) = \mathbf{c}(t) + \mathbf{h}(t)$, and the output (Eq. (2),(4)) to $\mathbf{o}(t) = g(\hat{\mathbf{h}}(t))$ However, since there is no groundtruth for the hidden state like the constraint 1, we assume the $\mathbf{h}(t_k)$ are the knots the compensated curve passing through, rendering the constraints 1 as $\hat{\mathbf{h}(t_k^-)} = \hat{\mathbf{h}(t_k^+)} = \mathbf{h}(t_k^+)$. The rationality for the knots is that $\mathbf{h}(t_k^+)$ is updated by $\mathbf{x}(t_k)$ and thus contain more information than $\mathbf{h}(t_k^-)$. And the solution for $\mathbf{c}$ is explained in Thm. 2.

Theorem 2. *Given the second order jump at hidden state as*

$$\dot{\mathbf{r}}_k = \dot{\mathbf{h}}(t_k^+) - \dot{\mathbf{h}}(t_k^-); \tag{14}$$

$$\ddot{\mathbf{r}}_k = \ddot{\mathbf{h}}(t_k^+) - \ddot{\mathbf{h}}(t_k^-), \tag{15}$$

where $\dot{\mathbf{h}} = f$, $\ddot{\mathbf{h}} = \frac{\partial f}{\partial \mathbf{h}} f$, *and the error defined as*

$$\boldsymbol{\epsilon}_k^+ = \mathbf{h}(t_k^+) - \mathbf{h}(t_k^+) = \mathbf{0}; \tag{16}$$

$$\boldsymbol{\epsilon}_k^- = \mathbf{h}(t_k^+) - \mathbf{h}(t_k^-), \tag{17}$$

the c_k *is uniquely determined as Eq. (10).*

Another advantage for interpolation for the hidden state is that it has easier computation of the Jacobian.

3.6 Analysis of Properties

3.6.1 Model Reduction to Cubic Spline Interpolation

Our model would reduce to cubic spline interpolation if setting $\mathbf{o}$ in Eq. (5) as zero. In light of this, we further analyze our model with techniques used for cubic spline interpolation and experimentally show our advantages against it in Sec. 4.3.

3.6.2 Error Bound

With CSSC, we can further analyze the interpolation error bound presented in Thm. 3, which is hard to obtain for ODE-RNN.

Theorem 3. *Given the CSSC at the output space as Eq. (5), if $x \in C^4(\Omega)$, then the error and the first order error are bounded as*

$$||(x-\hat{o})^{(r)}||_\infty \leq C_r ||(x-o)^{(4)}||_\infty \tau^{4-r}, \quad (r=0,1), \tag{18}$$

where $||\cdot||_\infty$ is uniform norm, $(\cdot)^r$ is r-th derivative, $C_0 = \frac{5}{384}$, $C_1 = \frac{1}{24}$, τ is the maximum interval over Π.

The proof is in Appx. B. Theorem 3 provides a strict guideline for better interpolation: (G1) reducing τ; (G2) decrease $||(x-o)^{(4)}||$. G1 means denser sampling. G2 indicates the ODE-RNN output should be as similar to the ground truth as possible.

Interestingly, Eq. (18) resembles the error bound for cubic spline interpolation (Hall & Meyer, 1976), and can reduce to it if o is set zero. Hence, we analyse the advantage of our methods against ODE-RNN and cubic spline interpolation. Compared with ODE-RNN, our model more effectively mitigates the error for densely sampled curve; compared with the cubic spline interpolation, our error bound has an adjustable o that can leads smaller $||(x-o)^{(4)}||$ than $||x^{(4)}||$.

An implicit assumption for this error bound is that the x should be 4-th order derivable, hence this model is not suitable for sharply changing signal.

4 Experiments

4.1 Baseline

We compare the baselines as (1) Cubic Spline Interpolation (Cubic Spline) (2) A classic RNN where Δt is concatenated to the input (RNN-Δt) (3) GRU-D (Che et al., 2018) (4) ODE-RNN (Rubanova et al., 2019) which is what our compensation adds upon (5) Latent-ODE (Rubanova et al., 2019) which is a VAE structure employing ODE-RNN as the encoder and ODE as decoder (6) GRU-ODE-Bayes (De Brouwer et al., 2019) which extends the ODE as a continuous GRU. Our proposed methods are denoted as CSSC for output compensation and hidden CSSC for hidden space compensation.

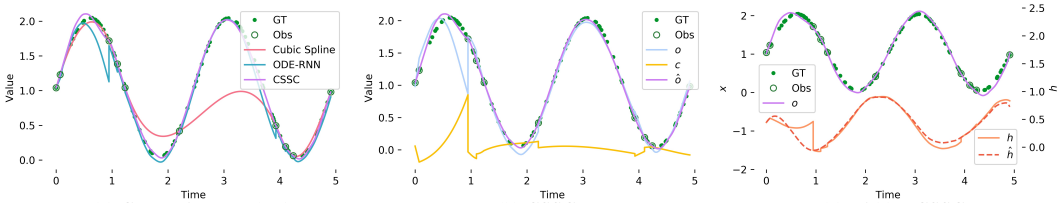
4.2 Implementation Details

The neural ODE state $\mathbf{h}$ has size 15. f is a 5-layer MLP with hidden state size 300. g is a 2-layer MLP with hidden state size 300. RNNCell has hidden state size 100. The α is selected as 1000 based on the ablation study of α in Sec. 4.6. The network is optimized by AdaMax (Kingma & Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and learning rate 0.02.

4.3 Toy Sinuous Wave Dataset

The toy dataset is composed of 1,000 periodic trajectories with variable frequency and the same amplitude. Following the setting of Rubanova et al. (2019), each trajectory has 100 irregularly-sampled time points with the initial point sampled from a standard Gaussian distribution. A fixed percentages of observations are randomly selected with the first and last time points included. The goal is to interpolate the full set of 100 points.

Table 1 shows the quantitative results compared with other baselines, and our method CSSC is the best. Figure 2 (a) illustrates the benefit of the CSSC over the ODE-RNN and cubic spline interpolation,



(a) Comparison methods (b) CSSC (c) Hidden CSSC

Figure 2: The visual result for Sinuous Wave dataset with 10 observations. (a) shows the comparison of CSSC against other methods; (b) demonstrates the effect of c to smooth the o; (c) shows the hidden CSSC output is smooth because its unsmooth hidden state h is smoothed into $\hat{h}$.

Figure 3: The visual result for MuJoCo.

where the performance of ODE-RNN is limited by the jump introduced at the new observation, CSSC can help eliminate such jump and guaranteed the smoothness at the observation time. Classic Cubic spline interpolation also fails in this case due to undersampling. Figure 2 (b) confirms the smoothing ability of CSSC at the output space by visualizing o c, and $\hat{o}$. Where o has a jump, c hedges against it with an opposite jump to keep the smoothness. Finally, Fig. 2 (c) presents the interpolation output and the first dimension of hidden states before and after smoothing.

4.4 MuJoCo Physics Simulation

We test our proposed method with the "hopper" model provided by DeepMind Control Suite (Tassa et al., 2018) based on MuJoCo physics engine. The hopper is thrown up, then rotates and freely falls down to the ground (Fig. 3). It is controlled by 14 dimension of state where 7 are position and the other 7 speed. Different trajectories are generated with random noise applied to the initial states. We interpolate the 7-dimensional position since speed is the first-order difference of position. Both our hidden CSSC and CSSC achieve improved performance from ODE-RNN, especially when observations are less, as Tab. 1 indicates. And the visual result is shown in Fig. 3.

4.5 Moving MNIST

Here we evaluate our method on high-dimensional data interpolation. Moving MNIST consists of 20-frame video sequences where two handwritten digits drawn from MNIST move with some velocity and direction within the 64×64 patch, with potential overlapping and bounce at the boundaries. As a matter of expediency, we use a subset of 10k videos and resize the frames into 32×32. 4 (20%) and 6 (30%) frames out of 20 are randomly observed, including the starting and ending frames. We encode the image with 2 ResBlock (He et al., 2016) into 32-d hidden vector and decode it to image with stack of transpose convolution layers. Since the pixels are not continuous, we only evaluate hidden CSSC with comparison methods and Tab. 1 manifests the hidden CSSC can further improves ODE-RNN's result, and the visual result (Fig. 4) shows that the performance gain comes from the smoother movement and the more clear overlapping.

Table 1: Interpolation MSE on toy, MuJoCo, Moving MNIST test sets with different percentages of observations.

	Toy			MuJoCo			Moving MNist	
	10%	30%	50%	10%	30%	50%	20%	30%
Cubic Spline	0.801249	0.003142	0.000428	0.016417	0.000813	0.000125	0.072738	0.252647
RNN-Δt	0.473954	0.247522	0.102043	0.028457	0.019682	0.008975	-	-
GRU-D	0.473954	0.247522	0.121482	0.056064	0.018285	0.008968	-	-
Latent-ODE	0.013768	0.002282	0.002031	0.010246	0.009601	0.009032	0.031287	0.031287
GRU-ODE-Bayes	0.117258	0.010716	0.000924	0.028457	0.006782	0.002352	-	-
ODE-RNN	**0.025336**	**0.001429**	0.000441	0.011321	0.001572	0.000388	0.022141	0.016998
hidden CSSC	0.027073	0.001503	**0.000244**	**0.004554**	**0.000378**	**0.000106**	**0.019475**	**0.015662**
CSSC	**0.024656**	**0.000457**	**0.000128**	**0.006097**	**0.000375**	**0.000087**	-	-

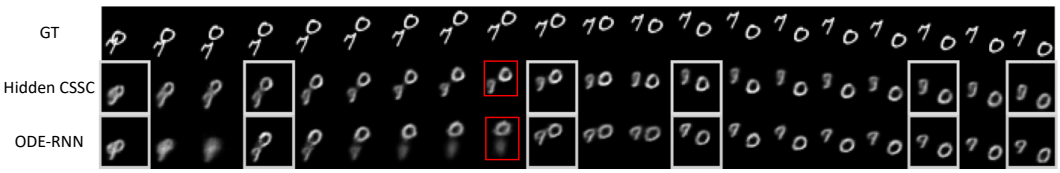


Figure 4: The visual result for Moving MNIST. The observation is indicated in white box. The comparison of discontinuity is highlighted in red box.

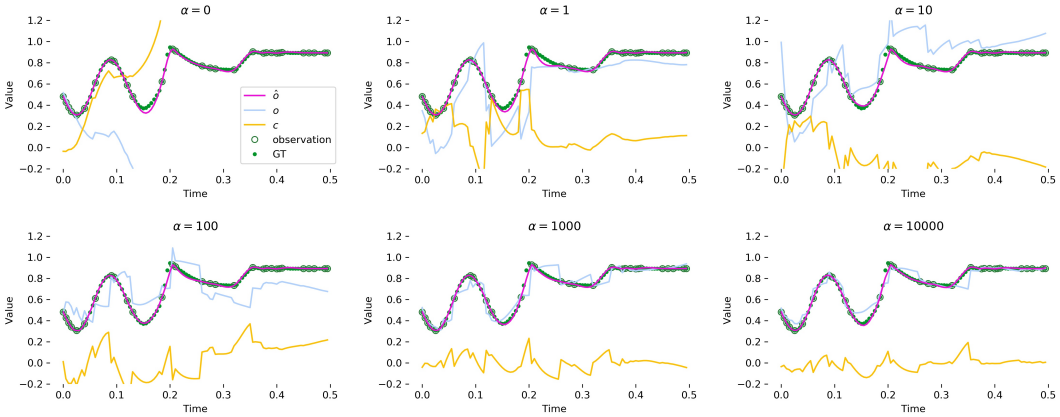


Figure 5: The prediction result with different α.

4.6 Ablation Study

The effect of end-to-end training. Apart from our standard end-to-end training of CSSC, two alternative training strategies are pre-hoc and post-hoc CSSN. Pre-hoc CSSC is to train a standard CSSC but only use the ODE-RNN part when inference. On the contrary, post-hoc CSSC is to train an ODE-RNN without CSSC, but apply CSSC upon the output of ODE-RNN when inference. We call this post-hoc CSSC. The comparison of pre-hoc, post-hoc and standard CSSC is presented in Tab. 2, where standard CSSC behaves the best. The pre-hoc CSSC is the worst because training with CSSC can tolerate the error of the ODE-RNN, thus inference without CSSC exposes the error of ODE-RNN, and it even perform worse than standard ODE-RNN. The post-hoc CSSC can increase the performance of ODE-RNN with a simple post-processing when inference. However, such performance gain is not guaranteed when the observation is sparse. For example, the post-hoc CSSC even decreases the performance of ODE-RNN in 10% observation setting. While standard CSSC can always increase the performance upon ODE-RNN, indicating the importance of end-to-end training.

The effect of α. We study the effect of α ranging from 0 to 10000 given different percentage of observation on MuJoCo dataset, shown in Tab. 3. The performance of CSSC is stable w.r.t. the choice

Table 2: The MSE for on MuJoCo test set.

	10%	30%	50%
ODE-RNN	0.011321	0.001572	0.000388
pre-hoc CSSC	0.053217	0.006131	0.002062
post-hoc CSSC	0.013574	0.000514	0.000110
CSSC	**0.006097**	**0.000375**	**0.000087**

Table 3: The MSE of CSSC for different α on MuJoCo test set.

α	0	1	10	100	1000	10000
10%	0.006551	0.011915	0.009691	**0.005421**	0.006097	0.005886
30%	0.000745	0.000463	0.000426	0.000400	**0.000375**	0.000422
50%	0.000126	0.000102	0.000074	0.000072	0.000087	**0.000057**

of α, especially the MSE only fluctuated from 0.000375 to 0.000463 as α ranges from 1 to 10000 in 30% observation setting. Interestingly, Fig. 5 visually compares the interpolation for o, c, and $\hat{o}$ under variant α and indicates higher α contributes to lower c, thus o is more dominant of smoothed output $\hat{o}$. On the other hand, the smaller α will makes o less correlated with the ground truth, but the CSSC c can always make $\hat{o}$ a well interpolated curve.

Different endpoint conditions

5 Conclusion, Limitation and Future Work

We introduce the Cubic Spline Smoothing Compensation (CSSC) that can address the discontinuity issue for ODE-RNN. We have derived the analytical solution for the CSSC and even proved its error bound for interpolation task, which is hard to obtained in pure neural network models. The CSSC combines the modeling ability of deep neural networks (ODE-RNN) and the smoothness advantage of cubic spline interpolation. Our experiments have shown the benefit of such combination. The hidden CSSC extends the smoothness from output space to the hidden semantic space, enabling a more general format of continuous signal, e.g., videos.

One limitation is that our model is not feasible for sharply changing signals, otherwise the fourth order derivative of the signal is large, leading to a large error bound (Eq. (18)). Therefore, firstly know the property of the data, then select the model.

Currently, our model is designed for interpolation. We can further extend the CSSC in an incremental computing manner; thus it can achieve sequence prediction.

REFERENCES

Garrett Birkhoff and Arthur Priver. Hermite interpolation errors for derivatives. *Journal of Mathematics and Physics*, 46(1-4):440–447, 1967.

Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, pp. 6775–6785, 2018.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems*, pp. 7377–7388, 2019.

Joseph Futoma, Sanjay Hariharan, and Katherine Heller. Learning to detect sepsis with a multitask gaussian process rnn classifier. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1174–1182. JMLR. org, 2017.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/garnelo18a.html`.

Charles A Hall and W Weston Meyer. Optimal error bounds for cubic spline interpolation. *Journal of Approximation Theory*, 16(2):105–122, 1976.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Zachary C Lipton, David Kale, and Randall Wetzel. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Machine Learning for Healthcare Conference*, pp. 253–270, 2016.

Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 1596–1607, 2018.

Michael Moor, Max Horn, Bastian Rieck, Damian Roqueiro, and Karsten Borgwardt. Early recognition of sepsis with gaussian process temporal convolutional networks and dynamic time warping. In Finale Doshi-Velez, Jim Fackler, Ken Jung, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens (eds.), *Proceedings of the 4th Machine Learning for Healthcare Conference*, volume 106 of *Proceedings of Machine Learning Research*, pp. 2–26, Ann Arbor, Michigan, 09–10 Aug 2019. PMLR. URL `http://proceedings.mlr.press/v106/moor19a.html`.

Michael C Mozer, Denis Kazakov, and Robert V Lindsey. Discrete event, continuous time rnns. *arXiv preprint arXiv:1710.04110*, 2017.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):18, 2018.

Yulia Rubanova, Tian Qi Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pp. 5321–5331, 2019.

Carl Runge. Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten. *Zeitschrift für Mathematik und Physik*, 46(224-243):20, 1901.

Satya Narayan Shukla and Benjamin Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2019.

Qingxiong Tan, Mang Ye, Baoyao Yang, Si-Qi Liu, and Andy Jinhua Ma. Data-gru: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. 2020.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

A PROOF OF INTERPOLATION

Substitute $\hat{o}$ with Eq. (5) we will have

$$c_k(t_k^+) + o(t_k^+) = x_k \tag{19}$$

$$c_k(t_{k+1}^-) + o(t_{k+1}^-) = x_{k+1} \tag{20}$$

$$\dot{c}_{k-1}(t_k^-) + \dot{o}(t_k^-) = \dot{c}_k(t_k^+) + \dot{o}(t_k^+) \tag{21}$$

$$\ddot{c}_{k-1}(t_k^-) + \ddot{o}(t_k^-) = \ddot{c}_k(t_k^+) + \ddot{o}(t_k^+) \tag{22}$$

$$\ddot{c}_0(t_0^+) + \ddot{o}(t_0^+) = 0 \tag{23}$$

$$\ddot{c}_{n-1}(t_n^-) + \ddot{o}(t_n^-) = 0 \tag{24}$$

And we let $\ddot{c}_k(t_k^+) = M_k, k = 0, 1, ..., n-1$, and $\ddot{c}_{n-1}(t_n^-) = M_n$. We define the first order and second order jump difference of ODE-RNN as

$$\dot{r}_k = \dot{o}(t_k^+) - \dot{o}(t_k^-); \tag{25}$$

$$\ddot{r}_k = \ddot{o}(t_k^+) - \ddot{o}(t_k^-). \tag{26}$$

With Eq. (22), we have

$$\hat{M}_{k+1} = \ddot{c}_k(t_{k+1}^-) = M_{k+1} + \ddot{r}(t_{k+1}). \tag{27}$$

Using constraint Eq. (19)(20), we denote

$$c_k(t_k) = x_k - o(t_k^+) = \epsilon_k^+ \tag{28}$$

$$c_k(t_{k+1}) = x_{k+1} - o(t_{t+1}^-) = \epsilon_{k+1}^-. \tag{29}$$

Also denote the step size $h_k = t_{k+1} - t_k$. Then applying constraint Eq. (19)(20)(22) we have the piece cubic function expressed as

$$c_k(t) = \frac{\hat{M}_{k+1} - M_k}{6h_k}(t - t_k)^3 + \frac{M_k}{2}(t - t_k)^2 + (\frac{\epsilon_{k+1}^- - \epsilon_k^+}{h_k} - \frac{h_k(\hat{M}_{k+1} + 2M_k)}{6})(t - t_k) + \epsilon_k^+. \tag{30}$$

Next, we try to solve all M_k. We firstly express $\dot{c}_{k-1}(t_k^-)$ and $\dot{c}_k(t_k^+)$ as

$$\dot{c}_{k-1}(t_k^-) = \frac{\hat{M}_k - M_{k-1}}{2}h_{k-1} + M_{k-1}h_{k-1} + \frac{\epsilon_k^- - \epsilon_{k-1}^+}{h_{k-1}} - \frac{\hat{M}_k + 2M_{k-1}}{6}h_{k-1}, \tag{31}$$

$$\dot{c}_k(t_k^+) = \frac{\epsilon_{k+1}^- - \epsilon_k^+}{h_k} - \frac{\hat{M}_{k+1} + 2M_k}{6}h_k \tag{32}$$

Applying Eq. (21) we have

$$2M_k + \frac{h_{k-1}}{h_{k-1} + h_k}M_{k-1} + \frac{h_k}{h_{k-1} + h_k}M_{k+1} = 6\frac{\epsilon[t_k^+, t_{k+1}^-] - \epsilon[t_{k-1}^+, t_k^-]}{h_{k-1} + h_k} + \frac{6\dot{r}_k - 2\ddot{r}_k h_{k-1} - \ddot{r}_{k+1}h_k}{h_{k-1} + h_k} \tag{33}$$

where $\epsilon[t_k^+, t_{k+1}^-] = \frac{\epsilon_{k+1}^- - \epsilon_k^+}{h_k}$.

$$\mu_k = \frac{h_{k-1}}{h_{k-1} + h_k} \tag{34}$$

$$\lambda_k = \frac{h_k}{h_{k-1} + h_k} \tag{35}$$

$$d_k = 6\frac{\epsilon[t_k^+, t_{k+1}^-] - \epsilon[t_{k-1}^+, t_k^-]}{h_{k-1} + h_k} + \frac{6\dot{r}_k - 2\ddot{r}_k h_{k-1} - \ddot{r}_{k+1} h_k}{h_{k-1} + h_k}. \tag{36}$$

Then M_k can be obtained by solving by system of linear equations:

$$A\mathbf{M} = \mathbf{d}, \tag{37}$$

where

$$A = \begin{pmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{pmatrix}, \mathbf{M} = \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{pmatrix}, \mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix} \tag{38}$$

And A is non-singular since it is strict diagonally dominant matrix, hence guarantee single solution for M_k. Hence

$$\mathbf{M} = A^{-1}\mathbf{d}. \tag{39}$$

Now we still need to calculate $\dot{o}(t)$ and $\ddot{o}(t)$.

$$\dot{o}(t) = \frac{\partial g}{\partial \mathbf{h}}^\intercal \frac{\partial \mathbf{h}}{\partial t} = \frac{\partial g}{\partial \mathbf{h}}^\intercal f \tag{40}$$

$$\ddot{\mathbf{h}}(t) = \frac{df(\mathbf{h}(t))}{dt} = \frac{\partial f}{\partial \mathbf{h}}^\intercal \frac{d\mathbf{h}}{dt} = \frac{\partial f}{\partial \mathbf{h}}^\intercal f, \tag{41}$$

$$\ddot{o}(t) = \frac{d\frac{\partial g}{\partial \mathbf{h}}^\intercal}{dt} \dot{\mathbf{h}} + \frac{\partial g}{\partial h}^\intercal \ddot{\mathbf{h}} = (\frac{\partial^2 g}{\partial \mathbf{h}^2} f)^\intercal f + \frac{\partial g}{\partial \mathbf{h}}^\intercal \frac{\partial f}{\partial \mathbf{h}} f \tag{42}$$

B Proof for Error Bound

The following proof are based on the default notation and setting for scalar time series interpolation: given a set of $n+1$ points $\Omega = \{x(t_i)\}_{i=0}^n$ irregularly spaced at time points $\Pi : a = t_0 < t_1 < ... < t_n = b$, the goal is to approximate the ground truth function $x(t)$, $t \in [a, b]$. Let $o(t)$ as the ODE-RNN prediction, and $\hat{o}(t) = c(t) + o(t)$ as our smoothed output where $c(t)$ is the CSSC defined by (10). To simplify the notation, we drop the argument t for a function, e.g. $x(t) \to x$.

Lemma 4. *(Hall & Meyer, 1976) Let e be any function in $C^2(\Omega) \bigcap_{k=1}^n C^4(x_{k-1}, x_k)$ with constraints $x(t_k) = 0$, $k = 0, ..., n$, and natural boundary $e''(a) = e''(b) = 0$, then*

$$|e^{(r)}(t_k)| \leq \rho_r ||e^{(4)}||_\infty \tau^{4-r} \qquad (k = 0, ..., n;\ r = 1, 2), \tag{43}$$

where $\rho_1 = \frac{1}{24}$, $\rho_2 = \frac{1}{4}$.

Lemma 5. *Let $e = x - \hat{o}$, $x \in C^4$, $g \in C^4$, then $e \in C^2(\Omega) \bigcap_{k=1}^n C^4(x_{k-1}, x_k)$*

Proof. Since $\hat{o} \in C^2(\Omega)$, and $c \in \bigcap_{k=1}^n C^4(x_{k-1}, x_k)$, hence we only have to prove that $o \in \bigcap_{k=1}^n C^4(x_{k-1}, x_k)$. According to Eq. (1) and (2), different orders of the derivative of o are as follows

$$\dot{o} = \frac{\partial g}{\partial \mathbf{h}}^\intercal f, \tag{44}$$

$$\ddot{o} = (\frac{\partial^2 g}{\partial \mathbf{h}^2} f)^\intercal f + \frac{\partial g}{\partial \mathbf{h}}^\intercal \frac{\partial f}{\partial \mathbf{h}} f, \tag{45}$$

$$o^{(3)} = f^{\intercal}(\frac{\partial f}{\partial \mathbf{h}}\frac{\partial^2 g}{\partial \mathbf{h}^2} + \frac{\partial^3 g}{\partial \mathbf{h}^3} \circ f + 2\frac{\partial^2 g}{\partial \mathbf{h}^2}\frac{\partial f}{\partial \mathbf{h}})f + \frac{\partial g}{\partial \mathbf{h}}^{\intercal}(\frac{\partial^2 f}{\partial \mathbf{h}^2} \circ f + \frac{\partial f}{\partial \mathbf{h}}\frac{\partial f}{\partial \mathbf{h}})f, \tag{46}$$

$$\begin{aligned} o^{(4)} =& f^{\intercal}\big(\frac{\partial^4 g}{\partial \mathbf{h}^4} \circ f \circ f + \frac{\partial^3 g}{\partial \mathbf{h}^3} \circ (\frac{\partial f}{\partial \mathbf{h}}\frac{\partial f}{\partial \mathbf{h}}) + 3\frac{\partial^3 g}{\partial \mathbf{h}^3} \circ f\frac{\partial f}{\partial \mathbf{h}} + 2\frac{\partial f}{\partial \mathbf{h}}\frac{\partial^3 g}{\partial \mathbf{h}^3} \circ f + 3\frac{\partial^2 g}{\partial \mathbf{h}^2}\frac{\partial^2 f}{\partial \mathbf{h}^2} \circ f \\ &+ 3\frac{\partial^2 g}{\partial \mathbf{h}^2}\frac{\partial f}{\partial \mathbf{h}}\frac{\partial f}{\partial \mathbf{h}} + \frac{\partial^2 f}{\partial \mathbf{h}^2} \circ f\frac{\partial^2 g}{\partial \mathbf{h}^2} + \frac{\partial f}{\partial \mathbf{h}}\frac{\partial f}{\partial \mathbf{h}}\frac{\partial^2 g}{\partial \mathbf{h}^2} + 3\frac{\partial f}{\partial \mathbf{h}}\frac{\partial^2 g}{\partial \mathbf{h}^2}\frac{\partial f}{\partial \mathbf{h}}\big)f \\ &+ \frac{\partial g}{\partial \mathbf{h}}^{\intercal}\big(\frac{\partial^3 f}{\partial \mathbf{h}^3} \circ f \circ f + \frac{\partial^2 f}{\partial \mathbf{h}^2} \circ (\frac{\partial f}{\partial \mathbf{h}}f) + 2\frac{\partial^2 f}{\partial \mathbf{h}^2} \circ f\frac{\partial f}{\partial \mathbf{h}} + \frac{\partial f}{\partial \mathbf{h}}\frac{\partial^2 f}{\partial \mathbf{h}^2} \circ f + \frac{\partial f}{\partial \mathbf{h}}\frac{\partial f}{\partial \mathbf{h}}\frac{\partial f}{\partial \mathbf{h}}\big)f, \end{aligned} \tag{47}$$

where $\frac{\partial f}{\partial \mathbf{h}}$ is the Jacobian matrix since f is a multi-valued function, $\frac{\partial^2 g}{\partial \mathbf{h}^2}$ is the Hessian matrix since o is a scalar and g is single-valued function. The higher order derivative than these is multi-dimensional matrix which we cannot write the matrix production as normal, so we indicate the product of a multi-dimensional matrix and a vector as $\circ$, which has higher computing priority then normal matrix production. From Eq. (47), $o^{(4)}$ depends on $\frac{\partial^4 g}{\partial \mathbf{h}^4}$ and $\frac{\partial^3 f}{\partial \mathbf{h}^3}$. Hence given $g \in C^4(x_{k-1}, x_k)$ and $f \in C^3(x_{k-1}, x_k)$, we obtain $o \in C^4(x_{k-1}, x_k)$.

□

Lemma 6. *(Birkhoff & Priver, 1967) Let u be the cubic Hermite interpolation of $x - o$ over Π, indicating u is piecewise cubic function in each $[x_i, x_{i+1})$, then*

$$|(x(t) - o(t) - u(t))^{(r)}| \leq A_r(t)||(x - o)^{(4)}||_\infty \tau^{4-r} \qquad (r = 0, 1), \tag{48}$$

with

$$A_r(t) = \frac{\tau_i^r[(t - x_i)(x_{i+1} - t)]^{2-r}}{r!(4 - 2r)!\tau^{4-r}} \qquad \text{if } t \in [x_i, x_j). \tag{49}$$

Let $\varepsilon = u - c$, then ε is cubic Hermite interpolation implying

$$\varepsilon(t_k) = x(t_k) - o(t_k) - c(t_k) = 0, \tag{50}$$

and

$$\varepsilon'(t_k) = x'(t_k) - o'(t_k) - c'(t_k) = e'(t_k). \tag{51}$$

Therefore, ε can be constructed for each interval $[t_k, t_{k+1})$ as

$$\varepsilon(t) = e'(t_k)H_1(t) + e'(t_{k+1})H_2(t), \tag{52}$$

where

$$H_1(t) = (t - t_k)(t - t_{k+1})^2/\tau_k^2, \tag{53}$$

$$H_2(t) = (t - t_{k+1})(t - t_k)^2/\tau_k^2. \tag{54}$$

Using Lemma 4,

$$|\varepsilon^{(r)}(t)| \leq \rho_1||e^{(4)}||_\infty\big(|H_1^{(r)}(t)| + |H_2^{(r)}(t)|\big)\tau^3 \qquad (r = 0, 1), \tag{55}$$

which is rewritten as

$$|(x(t) - \hat{o}(t))^{(r)}| \leq B_r(x)||e^{(4)}||_\infty \tau^{4-r} = B_r(x)||(x - o)^{(4)}||_\infty \tau^{4-r}, \tag{56}$$

with $B_r(t) = \rho_1\big(|H_1^{(r)}(t)| + |H_2^{(r)}(t)|\big)\tau^{r-1}$.

Using Triangle inequality, Lemma 6, and Eq. 56, it issues

$$\begin{aligned} |x^{(r)}(t) - \hat{o}^{(r)}(t)| &= |x^{(r)}(t) - o^{(r)}(t) - u^{(r)}(t) + u^{(r)}(t) - c^{(r)}(t)| \\ &\leq |x^{(r)}(t) - o^{(r)}(t) - u^{(r)}(t)| + |u^{(r)}(t) - c^{(r)}(t)| \\ &\leq (A_r(x) + B_r(x))||(x - o)^{(4)}||_\infty \tau^{4-r}. \end{aligned} \tag{57}$$

Let $C_r(x) = A_r(x) + B_r(x)$, and an analysis of the optimality (Hall & Meyer, 1976) yields

$$C_0(x) \leq \frac{5}{384}; \qquad C_1(x) \leq \frac{1}{12}. \tag{58}$$