Shallow Flow Matching
for Coarse-to-Fine Text-to-Speech Synthesis

Dong Yang, Yiyi Cai, Yuki Saito, Lixu Wang, Hiroshi Saruwatari

Presenter: Dong Yang
Ph.D. Student, Saruwatari & Saito Lab,
The University of Tokyo

Motivation

- Coarse-to-fine generative models with flow matching (FM)
  - Weak generator: coarse representations
  - FM module (refiner): fine representations

Motivation

- Coarse-to-fine generative models with flow matching (FM)
  - Weak generator: coarse representations
  - FM module (refiner): fine representations

- Coarse representations
  - Conventional: FM conditions
    - Training and inference starts from pure noise
  - Proposed: build intermediate states on the FM paths
    - Training and inference starts from these intermediate states

Motivation

- Coarse-to-fine generative models with flow matching (FM)
  - Weak generator: coarse representations
  - FM module (refiner): fine representations

- Coarse representations
  - Conventional: FM conditions
    - Training and inference start from pure noise
  - Proposed: build intermediate states on the FM paths
    - Training and inference start from these intermediate states

- Shallow flow matching (SFM)
  - More focused computation
    - Better quality
  - More stable sampling process
    - Faster inference when using adaptive-step ODE solvers

J. Liu et al. Diffsinger: Singing voice synthesis via shallow diffusion mechanism. In AAAI, pages 11020–11028, 2022.

Methods

- Theorem 1
  - How to map samples from a Gaussian distribution onto the FM paths

Theorem 1. *For any random variable $\boldsymbol{x}_m \sim \mathcal{N}(t_m \boldsymbol{x}_1, \sigma_m^2 \boldsymbol{I})$, where $t_m \in [0, \infty)$ and $\sigma_m \in (0, \infty)$, we define a transformation that maps $\boldsymbol{x}_m$ onto the conditional optimal transport (CondOT) paths. The output distribution varies continuously with respect to t_m and σ_m under the Wasserstein-2 metric.*

$$\Delta = (1 - \sigma_{\min}) t_m + \sigma_m,$$

$$\boldsymbol{x}_\tau = \begin{cases} \sqrt{(1 - (1 - \sigma_{\min}) t_m)^2 - \sigma_m^2}\, \boldsymbol{x}_0 + \boldsymbol{x}_m, & \text{if } \Delta < 1, \\ \frac{1}{\Delta} \boldsymbol{x}_m, & \text{if } \Delta \geq 1, \end{cases}$$

where $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, with corresponding $\tau = \min(t_m, \frac{t_m}{\Delta})$ on the path.

Methods

- Theorem 2
  - How to divide the FM paths into two segments

Theorem 2. *For arbitrary intermediate states on the CondOT paths:*

$$\boldsymbol{x}_{t_m} = (1 - t_m)\boldsymbol{x}_0 + t_m(\boldsymbol{x}_1 + \sigma_{\min}\boldsymbol{x}_0), \quad t_m \in (0, 1), \boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),$$

we can divide the paths into two segments at t_m and represent the flow and vector field using piecewise functions:

$$\boldsymbol{x}_t = \begin{cases} (1 - \frac{t}{t_m})\boldsymbol{x}_0 + \frac{t}{t_m}\boldsymbol{x}_{t_m}, & \text{if } t < t_m, \\ (1 - \frac{t-t_m}{1-t_m})\boldsymbol{x}_{t_m} + \frac{t-t_m}{1-t_m}(\boldsymbol{x}_1 + \sigma_{\min}\boldsymbol{x}_0), & \text{if } t \geq t_m, \end{cases}$$

$$\boldsymbol{u}_t = \begin{cases} \frac{1}{t_m}(\boldsymbol{x}_{t_m} - \boldsymbol{x}_0), & \text{if } t < t_m, \\ \frac{1}{1-t_m}(\boldsymbol{x}_1 + \sigma_{\min}\boldsymbol{x}_0 - \boldsymbol{x}_{t_m}), & \text{if } t \geq t_m. \end{cases}$$

Methods

- Validation on TTS models (training)
  - Orthogonal projection
    - Adaptively find the intermediate timestep

Algorithm 1 Training procedure of SFM

Input: the training set $(\mathcal{X}, \mathcal{C})$.

1: **repeat**

2: Sample $(\boldsymbol{X}_1, \boldsymbol{C})$ from $(\mathcal{X}, \mathcal{C})$;

3: Generate coarse representations

$$\boldsymbol{H_g}, \boldsymbol{X_g} \leftarrow \boldsymbol{g_\omega}(\boldsymbol{C})$$

$$\boldsymbol{X_h}, \hat{t}_{\boldsymbol{h}}, \log \hat{\sigma}^2_{\boldsymbol{h}} \leftarrow \boldsymbol{h_\psi}(\boldsymbol{H_g})$$

4: Compute coarse loss

$$\mathcal{L}_{\text{coarse}} = \mathbb{E}||\boldsymbol{X_g} - \boldsymbol{X}_1||^2$$

5: Project onto CondOT paths

$$t_{\boldsymbol{h}} \leftarrow \mathbb{E}\left[\frac{sg[\boldsymbol{X_h}] \cdot \boldsymbol{X}_1}{\boldsymbol{X}_1 \cdot \boldsymbol{X}_1}\right], \quad \sigma^2_{\boldsymbol{h}} \leftarrow \mathbb{E}||sg[\boldsymbol{X_h}] - t_{\boldsymbol{h}}\boldsymbol{X}_1||^2$$

6: Construct the intermediate state

$$\Delta \leftarrow \max((1-\sigma_{\min})t_{\boldsymbol{h}} + \sigma_{\boldsymbol{h}}, 1)$$

$$\boldsymbol{X_h} \leftarrow \frac{1}{\Delta}\boldsymbol{X_h}, \quad t_{\boldsymbol{h}} \leftarrow \frac{1}{\Delta}t_{\boldsymbol{h}}, \quad \sigma^2_{\boldsymbol{h}} \leftarrow \frac{1}{\Delta^2}\sigma^2_{\boldsymbol{h}}$$

$$\boldsymbol{X}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

$$\boldsymbol{X_h} \leftarrow \sqrt{\max((1-(1-\sigma_{\min})t_{\boldsymbol{h}})^2 - \sigma^2_{\boldsymbol{h}}, 0)}\boldsymbol{X}_0 + \boldsymbol{X_h}$$

$$\mathcal{L}_t = (\hat{t}_{\boldsymbol{h}} - t_{\boldsymbol{h}})^2, \quad \mathcal{L}_\sigma = (\log \hat{\sigma}^2_{\boldsymbol{h}} - \log \sigma^2_{\boldsymbol{h}})^2, \quad \mathcal{L}_\mu = \mathbb{E}||\boldsymbol{X_h} - t_{\boldsymbol{h}}\boldsymbol{X}_1||^2$$

7: Apply FM

$$t \sim \mathcal{U}[0,1], \quad t \leftarrow \mathcal{S}(t)$$

$$\boldsymbol{X}_t \leftarrow (1-t)\boldsymbol{X_h} + t(\boldsymbol{X}_1 + \sigma_{\min}\boldsymbol{X}_0)$$

$$\boldsymbol{U}_t \leftarrow \frac{1}{1-t_{\boldsymbol{h}}}(\boldsymbol{X}_1 + \sigma_{\min}\boldsymbol{X}_0 - \boldsymbol{X_h})$$

$$t \leftarrow (1-t_{\boldsymbol{h}})t + t_{\boldsymbol{h}}$$

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}||\boldsymbol{v_\theta}(\boldsymbol{X}_t, t) - \boldsymbol{U}_t||^2$$

8: Apply gradient descent to minimize $\mathcal{L}_{\text{SFM}}$

$$\mathcal{L}_{\text{SFM}} = \mathcal{L}_{\text{coarse}} + \mathcal{L}_t + \mathcal{L}_\sigma + \mathcal{L}_\mu + \mathcal{L}_{\text{CFM}}$$

9: **until** convergence

Methods

- Validation on TTS models (training)
  - Orthogonal projection
    - Adaptively find the intermediate timestep
  - Theorem 1
    - Intermediate state construction

Algorithm 1 Training procedure of SFM

Input: the training set $(\mathcal{X}, \mathcal{C})$.

repeat

Sample $(\boldsymbol{X}_1, \boldsymbol{C})$ from $(\mathcal{X}, \mathcal{C})$;

Generate coarse representations

$$\boldsymbol{H_g}, \boldsymbol{X_g} \leftarrow \boldsymbol{g_\omega}(\boldsymbol{C})$$

$$\boldsymbol{X_h}, \hat{t}_{\boldsymbol{h}}, \log \hat{\sigma}^2_{\boldsymbol{h}} \leftarrow \boldsymbol{h_\psi}(\boldsymbol{H_g})$$

Compute coarse loss

$$\mathcal{L}_{\text{coarse}} = \mathbb{E}||\boldsymbol{X_g} - \boldsymbol{X}_1||^2$$

Project onto CondOT paths

$$t_{\boldsymbol{h}} \leftarrow \mathbb{E}\left[\frac{sg[\boldsymbol{X_h}] \cdot \boldsymbol{X}_1}{\boldsymbol{X}_1 \cdot \boldsymbol{X}_1}\right], \quad \sigma^2_{\boldsymbol{h}} \leftarrow \mathbb{E}||sg[\boldsymbol{X_h}] - t_{\boldsymbol{h}}\boldsymbol{X}_1||^2$$

Construct the intermediate state

$$\Delta \leftarrow \max((1-\sigma_{\min})t_{\boldsymbol{h}} + \sigma_{\boldsymbol{h}}, 1)$$

$$\boldsymbol{X_h} \leftarrow \frac{1}{\Delta}\boldsymbol{X_h}, \quad t_{\boldsymbol{h}} \leftarrow \frac{1}{\Delta}t_{\boldsymbol{h}}, \quad \sigma^2_{\boldsymbol{h}} \leftarrow \frac{1}{\Delta^2}\sigma^2_{\boldsymbol{h}}$$

$$\boldsymbol{X}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

$$\boldsymbol{X_h} \leftarrow \sqrt{\max((1-(1-\sigma_{\min})t_{\boldsymbol{h}})^2 - \sigma^2_{\boldsymbol{h}}, 0)}\boldsymbol{X}_0 + \boldsymbol{X_h}$$

$$\mathcal{L}_t = (\hat{t}_{\boldsymbol{h}} - t_{\boldsymbol{h}})^2, \quad \mathcal{L}_\sigma = (\log \hat{\sigma}^2_{\boldsymbol{h}} - \log \sigma^2_{\boldsymbol{h}})^2, \quad \mathcal{L}_\mu = \mathbb{E}||\boldsymbol{X_h} - t_{\boldsymbol{h}}\boldsymbol{X}_1||^2$$

Apply FM

$$t \sim \mathcal{U}[0,1], \quad t \leftarrow \mathcal{S}(t)$$

$$\boldsymbol{X}_t \leftarrow (1-t)\boldsymbol{X_h} + t(\boldsymbol{X}_1 + \sigma_{\min}\boldsymbol{X}_0)$$

$$\boldsymbol{U}_t \leftarrow \frac{1}{1-t_{\boldsymbol{h}}}(\boldsymbol{X}_1 + \sigma_{\min}\boldsymbol{X}_0 - \boldsymbol{X_h})$$

$$t \leftarrow (1-t_{\boldsymbol{h}})t + t_{\boldsymbol{h}}$$

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}||\boldsymbol{v_\theta}(\boldsymbol{X}_t, t) - \boldsymbol{U}_t||^2$$

Apply gradient descent to minimize $\mathcal{L}_{\text{SFM}}$

$$\mathcal{L}_{\text{SFM}} = \mathcal{L}_{\text{coarse}} + \mathcal{L}_t + \mathcal{L}_\sigma + \mathcal{L}_\mu + \mathcal{L}_{\text{CFM}}$$

until convergence

Methods

- Validation on TTS models (training)
  - Orthogonal projection
    - Adaptively find the intermediate timestep
  - Theorem 1
    - Intermediate state construction
  - Theorem 2
    - Single-segment piecewise flow

Algorithm 1 Training procedure of SFM

Input: the training set $(\mathcal{X}, \mathcal{C})$.

repeat

Sample $(\boldsymbol{X}_1, \boldsymbol{C})$ from $(\mathcal{X}, \mathcal{C})$;

Generate coarse representations

$$\boldsymbol{H_g}, \boldsymbol{X_g} \leftarrow \boldsymbol{g_\omega}(\boldsymbol{C})$$

$$\boldsymbol{X_h}, \hat{t}_{\boldsymbol{h}}, \log \hat{\sigma}^2_{\boldsymbol{h}} \leftarrow \boldsymbol{h_\psi}(\boldsymbol{H_g})$$

Compute coarse loss

$$\mathcal{L}_{\text{coarse}} = \mathbb{E}||\boldsymbol{X_g} - \boldsymbol{X}_1||^2$$

Project onto CondOT paths

$$t_{\boldsymbol{h}} \leftarrow \mathbb{E}\left[\frac{sg[\boldsymbol{X_h}] \cdot \boldsymbol{X}_1}{\boldsymbol{X}_1 \cdot \boldsymbol{X}_1}\right], \quad \sigma^2_{\boldsymbol{h}} \leftarrow \mathbb{E}||sg[\boldsymbol{X_h}] - t_{\boldsymbol{h}}\boldsymbol{X}_1||^2$$

Construct the intermediate state

$$\Delta \leftarrow \max((1-\sigma_{\min})t_{\boldsymbol{h}} + \sigma_{\boldsymbol{h}}, 1)$$

$$\boldsymbol{X_h} \leftarrow \frac{1}{\Delta}\boldsymbol{X_h}, \quad t_{\boldsymbol{h}} \leftarrow \frac{1}{\Delta}t_{\boldsymbol{h}}, \quad \sigma^2_{\boldsymbol{h}} \leftarrow \frac{1}{\Delta^2}\sigma^2_{\boldsymbol{h}}$$

$$\boldsymbol{X}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

$$\boldsymbol{X_h} \leftarrow \sqrt{\max((1-(1-\sigma_{\min})t_{\boldsymbol{h}})^2 - \sigma^2_{\boldsymbol{h}}, 0)}\boldsymbol{X}_0 + \boldsymbol{X_h}$$

$$\mathcal{L}_t = (\hat{t}_{\boldsymbol{h}} - t_{\boldsymbol{h}})^2, \quad \mathcal{L}_\sigma = (\log \hat{\sigma}^2_{\boldsymbol{h}} - \log \sigma^2_{\boldsymbol{h}})^2, \quad \mathcal{L}_\mu = \mathbb{E}||\boldsymbol{X_h} - t_{\boldsymbol{h}}\boldsymbol{X}_1||^2$$

Apply FM

$$t \sim \mathcal{U}[0,1], \quad t \leftarrow \mathcal{S}(t)$$

$$\boldsymbol{X}_t \leftarrow (1-t)\boldsymbol{X_h} + t(\boldsymbol{X}_1 + \sigma_{\min}\boldsymbol{X}_0)$$

$$\boldsymbol{U}_t \leftarrow \frac{1}{1-t_{\boldsymbol{h}}}(\boldsymbol{X}_1 + \sigma_{\min}\boldsymbol{X}_0 - \boldsymbol{X_h})$$

$$t \leftarrow (1-t_{\boldsymbol{h}})t + t_{\boldsymbol{h}}$$

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}||\boldsymbol{v_\theta}(\boldsymbol{X}_t, t) - \boldsymbol{U}_t||^2$$

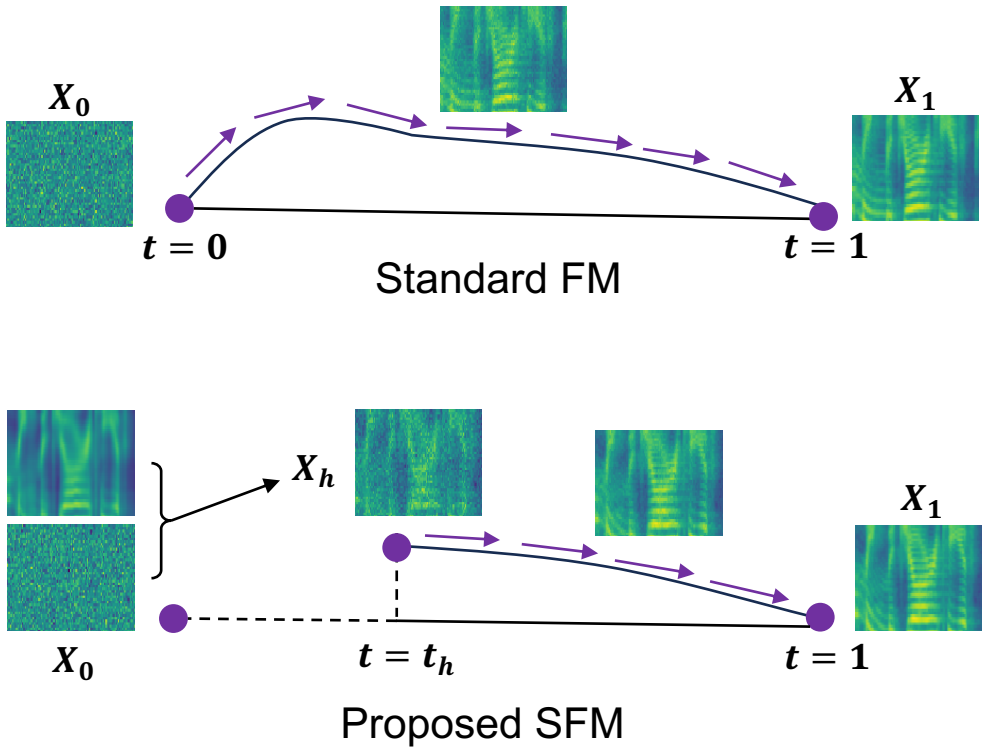
Apply gradient descent to minimize $\mathcal{L}_{\text{SFM}}$

$$\mathcal{L}_{\text{SFM}} = \mathcal{L}_{\text{coarse}} + \mathcal{L}_t + \mathcal{L}_\sigma + \mathcal{L}_\mu + \mathcal{L}_{\text{CFM}}$$

until convergence

Methods

- Validation on TTS models (inference)
  - SFM strength ($\alpha \geq 1$)
    - Encourage stronger guidance of the intermediate state



Algorithm 2 Inference procedure of SFM

Input: Condition C.

1: Generate coarse representations

$$\boldsymbol{H_g}, \boldsymbol{X_g} \leftarrow \boldsymbol{g_\omega}(\boldsymbol{C})$$
$$\boldsymbol{X_h}, t_{\boldsymbol{h}}, \log \sigma_{\boldsymbol{h}}^2 \leftarrow \boldsymbol{h_\psi}(\boldsymbol{H_g})$$
$$\sigma_{\boldsymbol{h}}^2 \leftarrow \exp(\log \sigma_{\boldsymbol{h}}^2)$$

2: Construct the intermediate state with SFM strength

$$\Delta \leftarrow \max(\alpha((1-\sigma_{\min})t_{\boldsymbol{h}} + \sigma_{\boldsymbol{h}}), 1)$$
$$\boldsymbol{X_h} \leftarrow \frac{\alpha}{\Delta}\boldsymbol{X_h}, \quad t_{\boldsymbol{h}} \leftarrow \frac{\alpha}{\Delta}t_{\boldsymbol{h}}, \quad \sigma_{\boldsymbol{h}}^2 \leftarrow \frac{\alpha^2}{\Delta^2}\sigma_{\boldsymbol{h}}^2$$
$$\boldsymbol{X}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$
$$\boldsymbol{X_h} \leftarrow \sqrt{\max((1-(1-\sigma_{\min})t_{\boldsymbol{h}})^2 - \sigma_{\boldsymbol{h}}^2, 0)}\boldsymbol{X}_0 + \boldsymbol{X_h},$$

3: Use an ODE solver to solve the integral

$$\boldsymbol{X}_{\text{pred}} = \boldsymbol{X_h} + \int_{t_{\boldsymbol{h}}}^{1} \boldsymbol{v_\theta}(\boldsymbol{X}_t, t)\, dt$$

Experiments

- Configurations

Table 1: **Training and inference configurations of TTS models.** β: CFG strength. †: 24,000 maximum frames.

Model	Vocoder	Dataset	Epochs	Batch Size	Learning Rate	Warmup	GPUs	β	Solver
Matcha-TTS [1]	Vocos [4]	LJ Speech [6]	800	128	4×10^{-4}	–	1	–	Euler
Matcha-TTS	Vocos	VCTK [7]	800	128	2×10^{-4}	–	2	–	Euler
StableTTS [2]	Vocos	VCTK	800	128	2×10^{-4}	10%	2	3	Dopri(5)
CosyVoice [3]	HiFi-GAN [5]	LibriTTS [8]	200	dynamic†	1×10^{-4}	10%	8	0.7	Euler
CosyVoice-DiT	Vocos	LibriTTS	200	256	1×10^{-4}	10%	4	3	Dopri(5)

[1] S. Mehta et al. Matcha-TTS: A fast TTS architecture with conditional flow matching. In ICASSP, pages 11341–11345, 2024.

[2] StableTTS. https://github.com/KdaiP/StableTTS, 2024.

[3] Z. Du et al. CosyVoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens. arXiv preprint arXiv:2407.05407, 2024

[4] H. Siuzdak. Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis. In ICLR, 2024.

[5] J. Kong et al. HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. In NeurIPS, 2020.

[6] K. Ito and L. Johnson. The LJ Speech dataset. https://keithito.com/ LJ-Speech-Dataset/, 2017.

[7] C. Veaux et al. CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit. The Centre for Speech Technology Research (CSTR), 2017.

[8] H. Zen et al. LibriTTS: A corpus derived from LibriSpeech for text-to-speech. In Interspeech, pages 1526–1530, 2019.

Experiments

- SFM strength selection
  - $\overline{\text{PMOS}}$: selection criterion
    - Increase initially and decrease as α grows

Table 3: **Partial objective evaluation results on validation sets for α selection.** The highest value for each metric is highlighted in bold. $\overline{\text{PMOS}}$: the average value of pseudo-MOS. WER: word error rate. SIM: speaker similarity.

α	$\tilde{t}_g$	$\tilde{\sigma}_g$	$\overline{\text{PMOS}}\uparrow$	UTMOS [1]↑	UTMOSv2 [2]↑	Distill-MOS [3]↑	WER(%)↓	SIM↑
Matcha-TTS (SFM) trained on LJ Speech								
1.0	0.099	0.092	4.036	4.194	3.721	4.192	4.641	0.972
2.0	0.198	0.183	4.158	**4.305**	3.834	4.337	3.496	**0.973**
2.5	0.248	0.229	**4.176**	4.276	**3.872**	4.381	3.556	0.972
3.0	0.297	0.275	4.168	4.260	3.842	4.402	3.496	0.970
3.5	0.347	0.320	4.132	4.190	3.802	4.403	3.496	0.969
4.0	0.397	0.366	4.107	4.137	3.763	**4.421**	3.556	0.966
5.0	0.496	0.458	4.025	3.977	3.694	4.403	3.376	0.960
6.0	0.520	0.480	3.997	3.958	3.648	4.386	**3.315**	0.958
7.0	0.520	0.480	3.990	3.960	3.625	4.386	**3.315**	0.958
8.0	0.520	0.480	3.990	3.959	3.625	4.386	**3.315**	0.958
9.0	0.520	0.480	3.993	3.956	3.638	4.386	**3.315**	0.958
10.0	0.520	0.480	3.987	3.955	3.620	4.386	**3.315**	0.958

[1] T. Saeki et al. UTMOS: utokyo-sarulab system for voicemos challenge 2022. In Interspeech, pages 4521–4525, 2022.

[2] K. Baba et al. The t05 system for the VoiceMOS Challenge 2024: Transfer learning from deep image classifier to naturalness MOS prediction of high-quality synthetic speech. In SLT, 2024.

[3] B. Stahl and H. Gamper. Distillation and pruning for scalable self-supervised representation-based speech quality assessment. In ICASSP, 2025.

Experiments

- Evaluation results
  - All SFM models perform better in pseudo-MOS, CMOS, and SMOS

Table 4: **Evaluation results on test sets.** * indicates statistically significant differences ($p < 0.05$) compared with SFM models in subjective evaluations. The highest value for each metric is bolded. WER: word error rate. SIM: speaker similarity. CMOS: comparative mean opinion score. SMOS: similarity mean opinion score.

System	UTMOS↑	UTMOSv2↑	Distill-MOS↑	WER↓	SIM↑	CMOS↑	SMOS↑
Matcha-TTS trained on LJ Speech							
Ground truth	4.380	3.964	4.241	3.566	1.000	+0.22	–
Reconstructed	4.085	3.739	4.208	3.472	0.993	+0.12	–
Baseline	4.186	3.692	4.282	**3.308**	0.971	−0.48	–
Ablated	4.217	3.763	4.311	3.355	0.972	−0.27	–
SFM (α=2.5)	**4.257**	**3.848**	**4.386**	3.413	**0.972**	**0.00**	–
Matcha-TTS trained on VCTK							
Ground truth	3.999	3.562	3.986	1.534	1.000	+0.16	–
Reconstructed	3.819	3.246	3.977	1.666	0.985	+0.08	–
Baseline	4.008	2.978	3.870	1.534	0.939	−0.31*	–
Ablated	4.026	2.997	3.872	1.613	**0.941**	−0.39*	–
SFM (α=3.5)	**4.106**	**3.105**	**3.898**	**0.952**	0.937	**0.00**	–
StableTTS trained on VCTK							
Ground truth	3.999	3.562	3.986	1.534	1.000	+0.48*	–
Reconstructed	3.360	2.908	3.855	1.719	0.972	+0.04	–
Ablated	3.328	2.958	3.929	1.798	0.932	−0.34*	–
SFM (α=3.0)	**3.516**	**3.020**	**3.953**	**1.745**	**0.933**	**0.00**	–
CosyVoice trained on LibriTTS							
Ground truth	4.136	3.262	4.345	3.180	1.000	+0.19	3.40
Reconstructed	3.942	3.126	4.336	3.146	0.993	−0.24	2.82*
Baseline	4.191	3.303	4.481	**3.513**	0.932	−0.21*	3.47
Ablated	4.183	3.369	4.487	3.578	**0.932**	−0.14	3.58
SFM (α=2.0)	**4.194**	**3.480**	**4.541**	3.810	0.931	**0.00**	**3.67**
CosyVoice-DiT trained on LibriTTS							
Ground truth	4.136	3.262	4.345	3.180	1.000	+0.23	3.31
Reconstructed	3.322	2.855	4.211	3.144	0.989	−0.12	2.86*
Ablated	3.499	3.086	4.316	3.614	**0.936**	−0.31*	3.15
SFM (α=2.5)	**3.751**	**3.171**	**4.502**	**3.598**	0.932	**0.00**	**3.21**

Experiments

- Acceleration of adaptive-step ODE solvers

Table 2: Adaptive-step ODE solvers used in our experiments. All of them are variants of the explicit Runge–Kutta family.

Abbreviation	Full Name	Order
Heun(2)	Adaptive Heun's Method	2
Fehlberg(2)	Runge–Kutta–Fehlberg Method	2
Bosh(3)	Bogacki–Shampine Method	3
Dopri(5)	Dormand–Prince Method	5

Table 5: **Partial RTF and NFE results for adaptive-step ODE solvers.** $\overline{\text{RTF}}$ denotes the mean real-time factor. Rate (%) denotes the relative speedup in terms of $\overline{\text{RTF}}$ compared to the ablated model. NFE is number of function evaluations.

System	Heun(2)			Fehlberg(2)			Bosh(3)			Dopri(5)		
	$\overline{\text{RTF}}\downarrow$	Rate↑	NFE↓	$\overline{\text{RTF}}\downarrow$	Rate↑	NFE↓	$\overline{\text{RTF}}\downarrow$	Rate↑	NFE↓	$\overline{\text{RTF}}\downarrow$	Rate↑	NFE↓
Matcha-TTS trained on LJ Speech												
Baseline	0.407	-1.496	312.36	0.054	3.571	44.82	0.271	-0.370	223.95	0.142	2.069	120.10
Ablated	0.401	0.000	306.72	0.056	0.000	45.28	0.270	0.000	221.81	0.145	0.000	121.46
SFM (α=1.0)	0.361	9.858	277.56	0.053	4.171	44.08	0.225	16.924	186.23	0.132	9.100	111.14
SFM (α=2.0)	0.299	25.541	229.43	0.048	13.306	39.16	0.187	30.931	153.08	0.115	20.484	96.02
SFM (α=3.0)	0.249	37.931	191.49	0.043	22.749	34.88	0.157	41.895	129.02	0.100	30.753	84.20
SFM (α=4.0)	0.207	48.486	156.78	0.037	34.246	29.74	0.131	51.576	107.90	0.086	40.559	72.56
SFM (α=5.0)	0.157	60.825	120.07	0.027	50.968	22.14	0.110	59.266	90.74	0.076	47.605	63.74

Conclusion

- Shallow flow matching
  - Scene
    - The FM module serves as a refiner
  - Method
    - Use coarse representations to build intermediate states on the FM paths
  - Effect
    - Better generation quality
    - Faster inference when using adaptive-step ODE solvers
  - Outlook
    - Applicable to other tasks and domains

Check our project page: https://github.com/ydqmkkx/ShallowFlowMatching-TTS