A. Appendix

A.1. Dataset Collection and Preprocessing

In this work we use the dataset gathered by Abbeel et al. (2010) and available at `http://heli.stanford.edu/`. A gas-powered helicopter was flown by a professional pilot to collect a large dataset of 6290s of flight. There are four controls: the longitudinal and lateral cyclic pitch, the tail rotor pitch and the collective pitch. The state is measured thanks to an accelerometer, a gyroscope, a magnetometer and vision cameras. Abbeel et al. (2010) provide the raw data, as well as states estimates in the Earth reference frame obtained with extended Kalman smoothing. Following Punjani & Abbeel (2015), we use the fused sensor data and downsample it from 100Hz to 50Hz.

From the Earth frame accelerations provided in the dataset, we compute body frame accelerations (minus gyroscopic terms) which are the prediction targets for our training. Using the notations from Punjani & Abbeel (2015), we can write the helicopter dynamics in the following form:

$$\dot{s} = F(s, \delta) = \begin{bmatrix} C_{12}v \\ \frac{1}{2}\hat{\omega}q \\ C_{12}^{\top}g - \omega \times v + f_v(s,\delta) \\ f_\omega(s,\delta) \end{bmatrix} \tag{18}$$

where $s \in \mathbb{R}^{13}$ is the helicopter state consisting of its position r, quaternion-attitude q, linear velocity v, angular velocity ω, and $\delta \in \mathbb{R}^4$ to be the control command. C_{12} is the rotation-matrix from the body to Earth reference frame, and f_v and f_ω are the linear and angular accelerations caused by aerodynamic forces, and are what we aim to predict.

The above notation is related to that used in Section 4.2 as follows:

- We define u as the concatenation of all inputs to the model, including the relevant state variables v and ω and control commands δ.
- We define y as the output predicted, which would correspond to a concatenation of f_v and f_ω.
- We define x as the vector of unobserved flow states to be estimated and is not present in their model.

The processed dataset used in our experiments can be found at `https://doi.org/10.5281/zenodo.3662987` (Menda et al., 2020).

A.2. Training Helicopter Models

Neural networks in the *naive* and *H25* models have eight hidden layers of size 32 each, and tanh nonlinearities. We optimize these models using an Adam optimizer (Kingma & Ba, 2015) with a harmonic learning rate decay, and mini-batch size of 512.

The neural network in the *NL* model has two hidden layers of size 32 each, and tanh nonlinearity. We train the *NL* model with CE-EM, using $\rho_x = \rho_\theta = 0.5$, $\sigma_w = \sigma_v = 1.0$, and use an Adam optimizer to optimize Equation (11) in the learning step. The learning rate for dynamics parameters in θ_{NL} is 5.0×10^{-4} and observation parameters in θ_{NL} is 1.0×10^{-3}. For its relative robustness, we optimize Equation (10) using a nonlinear least squares optimizer with a Trust-Region Reflective algorithm (Jones et al., 2001–) in the smoothing step. This step can be solved very efficiently by providing the solver with the block diagonal sparsity pattern of the Jacobian matrix.

To evaluate the test metric, running an EKF is required. The output of an EKF depends on several user-provided parameters:

- x_0: value of the initial state
- Σ_0: covariance of error on initial state
- Q: covariance of process noise
- R: covariance of observation noise

In this work, we assume that Q, R and Σ_0 are all set to the identity matrix. x_0 is assumed to be 0 on all dimensions.

A well-tuned EKF with an inaccurate initial state value converges to accurate estimations in only a few time steps of transient behavior. Since the *H25* model needs 25 past inputs to predict its first output prediction, we drop the first 25 predictions from the EKF when computing RMSE, thereby omitting some of the transient regime.

A.3. Figures

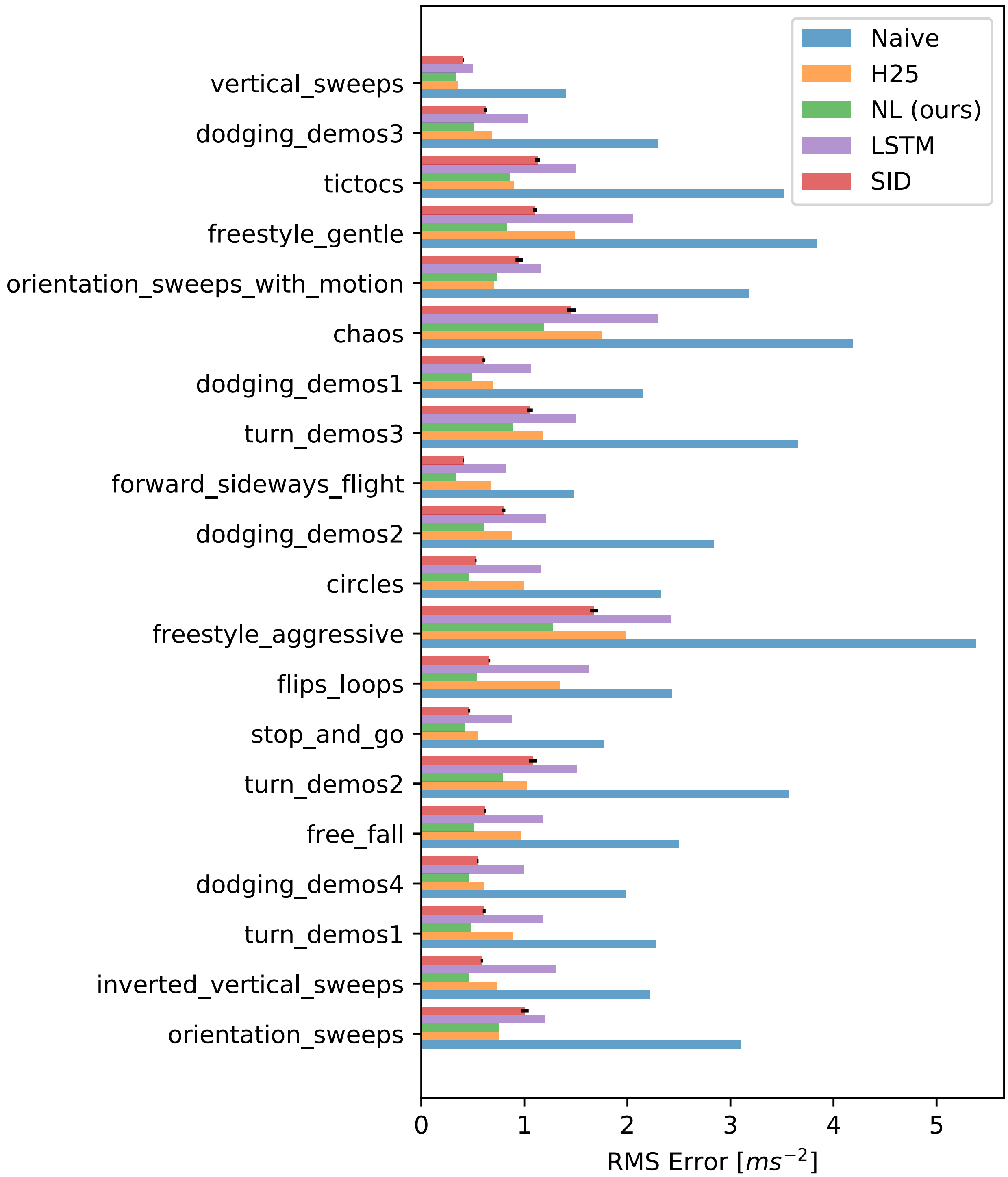

Figure 5. Test performance of optimized models on various trajectories

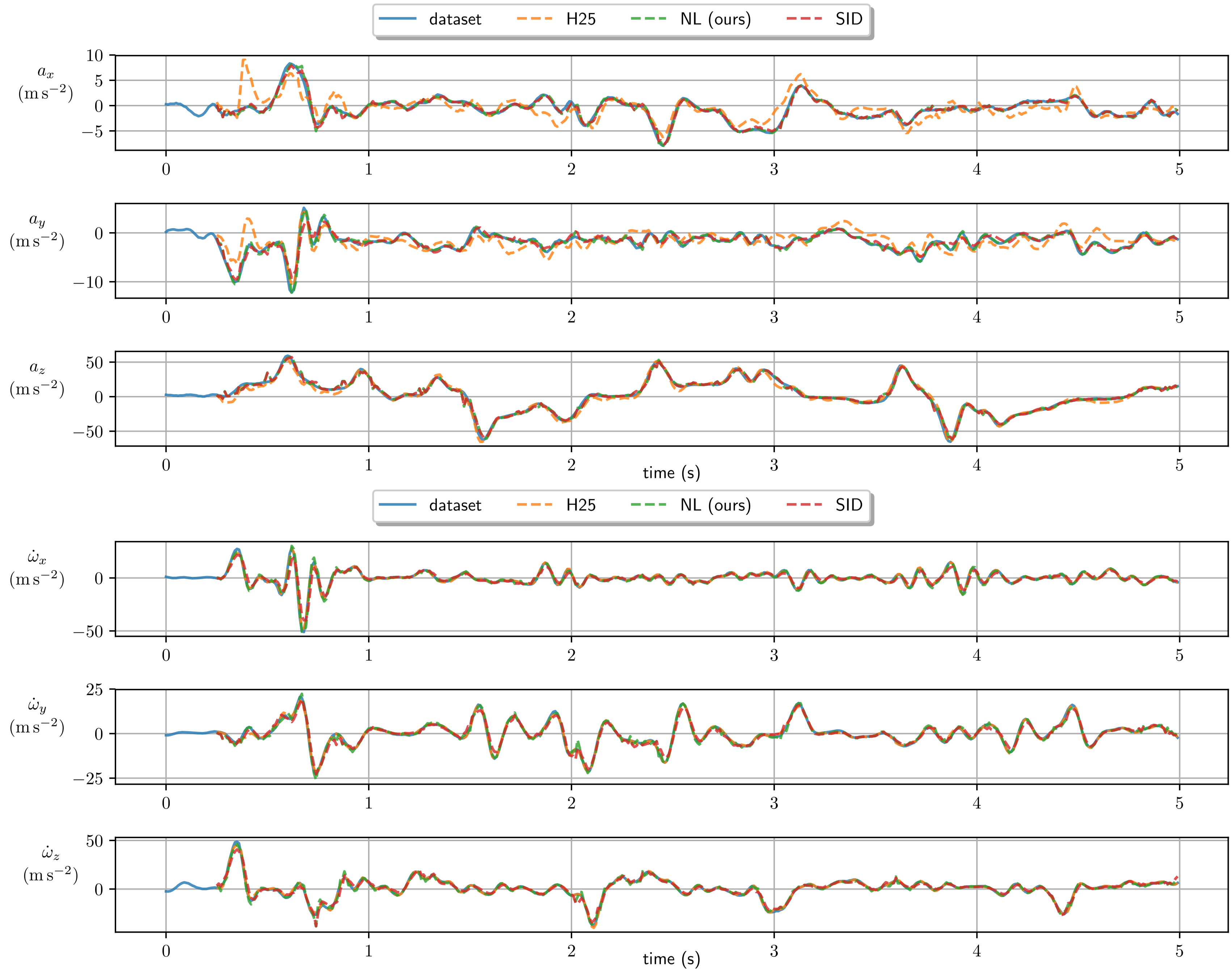

Figure 7. Predicted circular accelerations around axis x, y and z in the body frame. For subspace identification and models trained by CE-EM, this plot requires running an extended Kalman filter. These figures can be reproduced for any other trajectory with the included code.