Parity Calibration
(Supplementary Material)

Youngseog Chung[1] **Aaron Rumack**[1] **Chirag Gupta**[1]

[1]Machine Learning Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

A DETAILS ON EVALUATION: RELIABILITY DIAGRAMS AND METRICS

We provide details on how we assess a sequence of distributional forecasts $\{\hat{F}_t\}_{t=1}^T$ and parity probabilities $\{\hat{p}_t\}_{t=1}^T$, given a test dataset $\mathcal{D}_{\text{test}} = \{\mathbf{x}_t, y_t\}_{t=1}^T$. We assess distributional forecasts via Quantile Calibration, and the parity probabilities via Parity Calibration, Sharpness, and Accuracy metrics.

- **Quantile Calibration: reliability diagram and calibration error**
  To assess the quantile calibration of the distributional forecast $\hat{F}_t$, we produce the reliability diagram using the *Uncertainty Toolbox* [Chung et al., 2021]. This process works as follows. We take 100 equi-spaced quantile levels in $[0, 1]$: $p_i \in$ `np.linspace(0, 1, 100)`, and for each p_i, we compute the empirical coverage of the predictive quantile $\hat{F}_t^{-1}(p_i)$ with $\frac{1}{T}\sum_{t=1}^T \mathbb{1}\{y_t \leq \hat{F}_t^{-1}(p_i)\}$, and we denote this quantity as $p_{i,\text{obs}}$. Note that $p_{i,\text{obs}}$ is an empirical estimate of the term $\frac{1}{T}\sum_{t=1}^T F_t(\hat{F}_t^{-1}(p_i))$, from Eq. (3). The reliability diagram is produced by plotting $\{p_i\}$ on the x-axis against $\{p_{i,\text{obs}}\}$ on the y-axis. Quantile Calibration Error (QCE) is then computed as the average of the absolute difference between p_i and $p_{i,\text{obs}}$ over the 100 values of p_i: $\frac{1}{100}\sum_{i=1}^{100} \mid p_{i,\text{obs}} - p_i \mid$.
- **Parity Calibration: reliability diagram and calibration error**
  For parity calibration, we produce the reliability diagram following the standard method in binary classification [DeGroot and Fienberg, 1981, Niculescu-Mizil and Caruana, 2005]. Note that the parity probability $\hat{p}_t$ is a prediction for the parity outcome $\widetilde{y}_t := \mathbb{1}\{y_t \leq y_{t-1}\}$ (Eq. (5)). Specifically, we first take 30 fixed-width bins of the predicted parity probabilities: $\{B_m\}_{m=1}^{30}$, where $B_m = [\frac{m-1}{30}, \frac{m}{30})$ for $m < 30$ and $B_{30} = [\frac{29}{30}, 1]$. The average outcome in bin B_m is computed as $\text{obs}(B_m) = \frac{1}{|B_m|}\sum_{t:\hat{p}_t \in B_m} \mathbb{1}\{\widetilde{y}_t = 1\}$, and the average prediction of bin B_m is computed as $\text{pred}(B_m) = \frac{1}{|B_m|}\sum_{t:\hat{p}_t \in B_m} \hat{p}_t$. The reliability diagram is then produced by plotting $\text{pred}(B_m)$ on the x-axis against $\text{obs}(B_m)$ on the y-axis. The blue bars in the background of each parity calibration reliability diagram represents the size of the bin: $|B_m|$. Parity Calibration Error (PCE) is then computed with this reliability diagram following the standard definition of (ℓ_1-)expected calibration error (ECE): $\sum_{m=1}^{30} \frac{|B_m|}{T} \mid \text{obs}(B_m) - \text{pred}(B_m) \mid$.
- **Sharpness**
  Assuming the same notation as above, sharpness is computed as: $\sum_{m=1}^{M} \frac{|B_m|}{T} \cdot \text{obs}(B_m)^2$, where M is the total number of bins. As indicated above, we use $M = 30$ in all of our experiments. We provide some additional intuition on this metric. A perfectly knowledgeable forecaster which outputs $\hat{p}_t = \widetilde{y}_t$ will place all predictions in either B_1 or B_M and achieve sharpness $= \frac{|B_1|}{T} \cdot \text{obs}(B_1)^2 + \frac{|B_M|}{T} \cdot \text{obs}(B_M)^2 = \frac{|B_1|}{T} \cdot 0^2 + \frac{|B_M|}{T} \cdot 1^2 = \frac{|B_M|}{T} = \frac{\sum_{t=1}^T \widetilde{y}_t}{T}$. On the other hand, if the forecaster places all predictions into a single bin B_k, then its sharpness will be $\text{obs}(B_k)^2 = \left(\frac{\sum_{t=1}^T \widetilde{y}_t}{T}\right)^2$. It can be shown that sharpness is always within the closed interval $\left[\left(\frac{\sum_{t=1}^T \widetilde{y}_t}{T}\right)^2, \frac{\sum_{t=1}^T \widetilde{y}_t}{T}\right]$ [Bröcker, 2009]. Intuitively, sharpness measures the degree to which the forecaster attributes different valued predictions to events with different outcomes (i.e. labels). Hence, a sharper, or more precise, forecaster has more discriminative power, and this is reflected in a higher sharpness metric.

- **Accuracy metrics (Acc and AUROC)**

  Accuracy is measured in the binary classification sense, where the true labels are the observed parity outcomes: $\mathbb{1}\{y_t \leq y_{t-1}\}$ (Eq. (5)).

  - **Binary accuracy (Acc)** is computed by regarding $\hat{p}_t \geq 0.5$ as the positive class prediction, and the opposite case as the negative class prediction.
  - **Area under the ROC curve (AUROC)** is computed using the `scikit-learn` Python package, which implements the standard definition of the score. Specifically, we called the function `sklearn.metrics.roc_auc_score` with the predictions $\{\hat{p}_t\}$ and labels $\mathbb{1}\{y_t \leq y_{t-1}\}$.

B ADDITIONAL DETAILS ON CASE STUDIES

B.1 ADDITIONAL DETAILS ON COVID-19 CASE STUDY

B.1.1 Details on Interpolating Expert Forecasts for COVID-19 Case Study

The expert forecast provided by the COVID-19 Forecast Hub is represented as a set of quantiles. To derive the parity probabilities $\hat{p}_{s,t}$, we need to interpolate the expert forecast, as the forecast contains predicted quantiles at only 7 quantile levels : $\{0.025, 0.1, 0.25, 0.5, 0.75, 0.9, 0.975\}$. We interpolate under the assumption that the density between two adjacent quantiles τ_k and τ_{k+1} are defined by the normal distribution specified by those two quantiles. Specifically, for two quantiles τ_k and τ_{k+1} and forecast values $x_k^{(s,t)}$ and $x_{k+1}^{(s,t)}$, we compute

$$\sigma_k^{(s,t)} = \frac{x_{k+1}^{(s,t)} - x_k^{(s,t)}}{\Phi^{-1}(\tau_{k+1}) - \Phi^{-1}(\tau_k)},$$

$$\mu_k^{(s,t)} = x_k^{(s,t)} - \sigma_k^{(s,t)}\Phi^{-1}(\tau_k),$$

where Φ is the standard normal cdf. For each forecast, if $x_k^{(s,t)} \leq y_{s,t-1} < x_{k+1}^{(s,t)}$, then the parity probability

$$\hat{p}_{s,t} = \Phi\left(\frac{y_{s,t-1} - \mu_k^{(s,t)}}{\sigma_k^{(s,t)}}\right).$$

If $y_{s,t-1} < x_1^{(s,t)}$, we can extrapolate using $\mu_1^{(s,t)}$ and $\sigma_1^{(s,t)}$, and if $y_{s,t-1} >= x_7^{(s,t)}$, we can extrapolate using $\mu_6^{(s,t)}$ and $\sigma_6^{(s,t)}$. However, this never occurs with the forecasts and observations in this dataset. Figure 9 provides a visualization of this interpolation scheme.

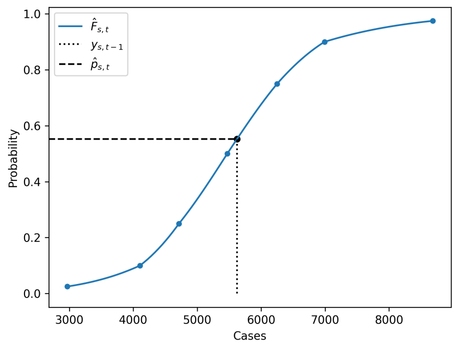

Figure 9: We use a piece-wise Gaussian interpolation of the expert forecast quantiles to estimate the predictive cdf, from which we then calculate the parity probabilities.

B.1.2 Details on Experiment Setup for COVID-19 Case Study

Section 3.1.1 compares the expert forecaster, its parity probabilities and posthoc calibration by OPS. We did not tune OPS hyperparameters in this experiment, so the full 119 weeks' worth of data was used for testing and reporting the results.

For Section 3.1.2, the first 20 weeks' worth of data was used for tuning hyperparameters, and the reported results are based on the remaining 99 weeks' worth of data as the test set.

For the decision-making experiment in Section 3.1.3, we used the parity probabilities produced from Section 3.1.2. Although the chosen loss function is just one example, we observe that similar results hold with any loss function that satisfies: $l_{2,3} \leq l_{2,2} \leq l_{1,1} \leq l_{2,1} \leq l_{1,2} \leq l_{1,3}$.

B.2 ADDITIONAL DETAILS ON WEATHER FORECASTING CASE STUDY

B.2.1 Details on Experiment Setup for Weather Forecasting Case Study

We used the modeling and training infrastructure provided by the Keras tutorial on *Timeseries Forecasting for Weather Prediction*[1] which models this same dataset with an LSTM network [Hochreiter and Schmidhuber, 1997]. We made one change to the model provided by the tutorial: since we are interested in probabilistic forecasts instead of point forecasts, we changed the head of the model and the loss function from a point output trained with mean squared error loss to a mean and variance output that parameterizes a Gaussian distribution and trained it with the Gaussian likelihood loss. Such a model is also referred to as a mean-variance network or a probabilistic neural network [Lakshminarayanan et al., 2017, Nix and Weigend, 1994], and it is one of the most popular methods currently used in probabilistic regression.

While the tutorial's setup takes as input the past 120 hours' window of 7 features to predict the value of one feature (Temperature) 12 hours into the future, we expand the setting to predict all 7 features: Pressure, Temperature, Saturation vapor pressure, Vapor pressure deficit, Specific humidity, Airtight, and Wind speed. We thus train 7 separate base regression models, one for each prediction target.

For the in-text experiment **Binary classifers as expert forecasts**, we trained binary classification base models with parity outcomes (Eq. (5)) as the labels and took this model as the expert forecaster. We adopted the same model architecture as the base regression model and changed the last layer to output a logit. We then trained the model with the cross entropy loss.

The full Jena dataset spans from the beginning of January 2009 to the end of December 2016, with $420,551$ datapoints in total. In chronological order, we set $272,638$ datapoints to train the base models (both the regression and classification model) and the subsequent $83,390$ datapoints for validation. Following the same model training procedure as the tutorial, training was stopped early if the validation loss did not increase for 20 training epochs.

Afterwards, in running the posthoc calibration methods (MW, IW, and OPS), we used the last $8,640$ datapoints of the validation set to tune the hyperparameters of each calibration method, and used subsequent windows of $8,640 \times 3 = 25,920$ datapoints for testing.

We run 50 test trials with a moving test timeframe to produce the mean and standard errors reported in Tables 3 and 4. Denoting the first test window as $[t+1, t+H]$ (i.e. H is set to $25,920$), we move this frame by a multiple of a fixed offset c into the future, and repeat this 50 times, to create a new set of 50 test sets. The resulting new test timeframes are $[t+1+(ck), t+H+(ck)]$, where $k = 0, 1, 2, \ldots 49$, and c was set to 336.

B.2.2 Additional Results on Weather Forecasting Case Study

We shows additional plots and tables from the experimental results in Section 3.2 of the main paper.

Figure 10 displays the full set of reliability diagrams for Figure 6, which corresponds to the in-text experiment **Binary classifiers as expert forecasts** in Section 3.2.

Table 6 displays the numerical results from the weather forecasting case study when averaged across all 7 prediction target settings. This corresponds to the in-text experiment **Results across all 7 timeseries** in Section 3.2. To produce these results, we fixed the test timeframe to be the first test timeframe $[t+1, t+H]$ for all prediction target settings, then computed the mean and standard errors across the 7 sets of metrics produced (one set for each prediction target).

[1] https://keras.io/examples/timeseries/timeseries_weather_forecasting/

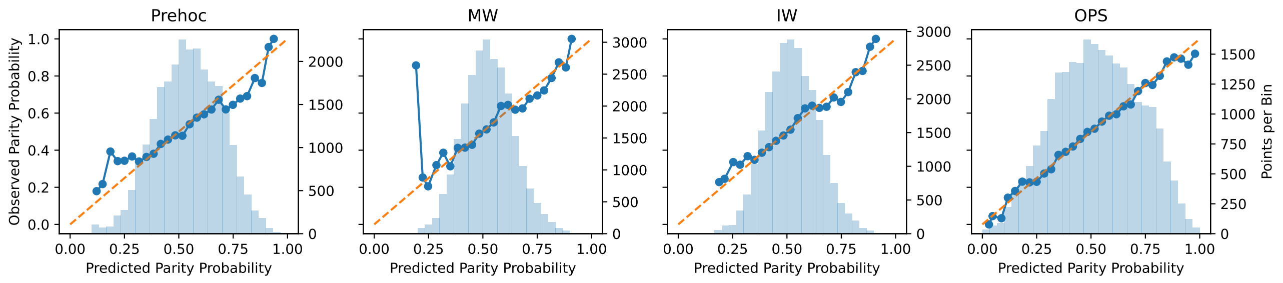

Figure 10: Reliability diagrams with a binary classification base model predicting Pressure. This is the full set of reliability diagrams for Figure 6 from Section 3.2. The left-most plot shows parity calibration of the base classification model (Prehoc), and the next three plots show the effects of MW, IW and OPS in calibrating the Prehoc parity probabilities. OPS produces the most calibrated and sharp parity probabilities.

	QCE $\downarrow$	PCE $\downarrow$	Sharp $\uparrow$	Acc $\uparrow$	AUROC $\uparrow$
Prehoc	$\mathbf{0.0266 \pm 0.0052}$	0.2794 ± 0.0161	0.2915 ± 0.0117	0.4902 ± 0.0159	0.4806 ± 0.0249
MW	N/A	0.0233 ± 0.0048	0.2913 ± 0.0117	0.5610 ± 0.0106	0.5419 ± 0.0195
IW	N/A	0.0188 ± 0.0047	0.2913 ± 0.0118	0.5630 ± 0.0099	0.5403 ± 0.0209
OPS	N/A	$\mathbf{0.0159 \pm 0.0009}$	$\mathbf{0.2961 \pm 0.0122}$	$\mathbf{0.5790 \pm 0.0122}$	$\mathbf{0.5830 \pm 0.0217}$

(a) Numerical results averaged across all 7 prediction settings where the base model is a Gaussian regression model. The base regression model (Prehoc) tends to be well quantile calibrated (QCE) but terribly parity calibrated (PCE). All methods (MW, IW, OPS) improve parity calibration, but OPS is the only method which improves all metrics simultaneously. Best value for each metric is in bold.

	PCE $\downarrow$	Sharp $\uparrow$	Acc $\uparrow$	AUROC $\uparrow$
Prehoc	0.0247 ± 0.0016	0.3049 ± 0.0074	0.6078 ± 0.0099	0.6348 ± 0.0136
MW	0.0170 ± 0.0018	0.3049 ± 0.0075	0.6061 ± 0.0102	0.6340 ± 0.0143
IW	0.0156 ± 0.0012	0.3047 ± 0.0074	0.6075 ± 0.0098	0.6340 ± 0.0136
OPS	$\mathbf{0.0135 \pm 0.0013}$	$\mathbf{0.3134 \pm 0.0075}$	$\mathbf{0.6278 \pm 0.0121}$	$\mathbf{0.6643 \pm 0.0183}$

(b) Numerical results averaged across all 7 prediction settings where the base model is a binary classification model trained with parity outcome labels. The base classification model (Prehoc) tends to be much better parity calibrated than when a regression base model is used (above Table 6a). All methods (MW, IW, OPS) improve parity calibration further, but OPS is the only method which improves all metrics simultaneously. Notably, MW and IW tends to decrease the accuracy of the parity probabilities. Best value for each metric is in bold.

Table 6: Numerical results from the weather forecasting case study (Section 3.2), averaged across all 7 forecasting targets. Table 6a displays results with the Gaussian regression base model, and Table 6b displays results with the binary classification base model. $\pm$ indicates mean $\pm$ 1 standard error, across the 7 prediction target settings.

B.3 ADDITIONAL DETAILS ON CONTROL IN NUCLEAR FUSION CASE STUDY

B.3.1 Details on Experiment Setup for Control in Nuclear Fusion Case Study

The expert forecaster for the nuclear fusion experiment in Section 3.3 is provided by a pretrained dynamics models that was used to optimize control policies for deployment on the DIII-D tokamak [Luxon, 2002], a nuclear fusion device in San Diego that is operated by General Atomics. The dynamics model was trained with logged data from past experiments (referred to as “shots”) on this device. Each shot consists of a trajectory of (state, action, next state) transitions, and one trajectory consists of $\sim$ 20 transitions (i.e. 20 timesteps).

As input, the model takes the current state of the plasma and the actuator settings (i.e. actions). The model outputs a multi-dimensional predictive distribution over the state variables in the next timestep. The state is represented by three signals: β_N (the ratio of plasma pressure over magnetic pressure), *density* (the line-averaged electron density), and *li* (internal inductance). For the actuators, the model takes in the amount of power and torque injected from the neutral beams, the current, the magnetic field, and four shape variables (*elongation*, a_{minor}, *triangularity-top*, and *triangularity-bottom*). This, along with the states, makes for an input dimension of 11 and output dimension of 3 for the states.

The model was implemented with a recurrent probabilistic neural network (RPNN), which features an encoding layer by an RNN with 64 hidden units followed by a fully connected layer with 256 units, and a decoding layer of fully connected layers with [128, 512, 128] units, which finally outputs a 3-dimensional isotropic Gaussian parameterized by the mean and a log-variance prediction.

The training dataset consisted of trajectories from 10294 shots, and the model was trained with the Gaussian likelihood loss, with a learning rate of 0.0003 and weight decay of 0.0001. In using dynamics models to sample trajectories and train policies, the key metric practitioners are concerned with is explained variance, hence explained variance on a held out validation set of 1000 shots was monitored during training. Training was stopped early if there was no improvement in explained variance over the validation set for more than 250 epochs. The test dataset consisted of another held-out set of 900 shots, with which we report all results presented in Section 3.3.

In all of our experiments, since β_N is the key signal of interest in our problem setting, we just examine the predictive distribution for β_N in the model outputs and ignore the other dimensions of the outputs.

In running the posthoc calibration methods (MW, IW, and OPS), we used the same validation set to tune the hyperparameters of each calibration method, and used windows of $15,000$ datapoints from the concatenated test shot data for testing.

We run 50 test trials with a moving test timeframe to produce the mean and standard errors reported in Table 5. Denoting the first test window as $[t+1, t+H]$ (i.e. H is set to $15,000$), we move this frame by a multiple of a fixed offset c into the future, and repeat this 50 times, to create a set of 50 test datasets. The resulting test timeframes are $[t+1+(ck), t+H+(ck)]$, where $k = 0, 1, 2, \dots 49$, and c was set to 100.

C DETAILS ON HYPERPARAMETERS

Each of the three calibration methods we consider in Section 2.2, which we use in our experiments in Section 3, requires a set of hyperparameters.

- **MW** requires `uf` and `ws`.
  - `uf` determines how often the PS parameters $(a^{\mathrm{MW}}, b^{\mathrm{MW}})$ are updated.
  - `ws` determines the size of the calibration set that is used to update the PS parameters
- **IW** requires `uf`.
  - `uf` determines how often the PS parameters $(a^{\mathrm{IW}}, b^{\mathrm{IW}})$ are updated.
    Note that IW always uses all of the data seen so far to update the PS parameters.
- **OPS** requires γ and `D`.
  - γ can be understood as step size for the OPS updates.
  - D can be understood as regularization for the OPS updates.

We provide details on how these hyperparameters were tuned for each of the three case studies.

C.1 HYPERPARAMETERS FOR COVID-19 CASE STUDY

We observed that OPS performed well with the default hyperparameters, so we did not tune hyperparameters for OPS for the COVID-19 case study. The default hyperparameter values used for OPS were $\gamma = 0.001$ and $\mathtt{D} = 10$.

For MW and IW, we tuned hyperparameters by optimizing parity calibration error (PCE, Section 3) on the first 20 weeks' worth of data as the validation set, over the following grids:

- $\mathtt{uf} \in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$, separately for MW and IW
- $\mathtt{ws} \in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$, for MW.

The COVID-19 dataset records data for each week, so the grid size of 1 represents 1 week.

The tuned hyperparameters we used for MW and IW are as follows:

- MW: $\mathtt{uf} = 1, \mathtt{ws} = 10$
- IW: $\mathtt{uf} = 5$

C.2 HYPERPARAMETERS FOR WEATHER FORECASTING CASE STUDY

For each calibration method, the hyperparameters were tuned by optimizing parity calibration error (PCE, Section 3) on the validation dataset over the following grids:

- $\texttt{uf} \in [1, 24, 168, 336, 720, 2160]$, separately for MW and IW
- $\texttt{ws} \in [24, 168, 336, 720, 2160, 4320, 8640]$, for MW
- $\gamma \in$ [1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2], for OPS
- $\texttt{D} \in [1, 10, 30, 50, 70, 100, 150, 200]$, for OPS.

The hyperparameters were tuned separately for each base model setting (regression and classification), for each method (MW, IW, and OPS), and for each base model predicting one of 7 targets (Pressure, Temperature, Saturation vapor pressure, Vapor pressure deficit, Specific humidity, Airtight, and Wind speed).

The tuned hyperparameters we used are as follows:

- **Base Regression Model**
  - Pressure Model
    - MW: $\texttt{uf} = 2160, \texttt{ws} = 8640$
    - IW: $\texttt{uf} = 2160$
    - OPS: $\gamma =$ 1e-5, $\texttt{D} = 50$
  - Temperature Model
    - MW: $\texttt{uf} = 336, \texttt{ws} = 8640$
    - IW: $\texttt{uf} = 168$
    - OPS: $\gamma =$ 1e-5, $\texttt{D} = 30$
  - Saturation Vapor Pressure Model
    - MW: $\texttt{uf} = 2160, \texttt{ws} = 2160$
    - IW: $\texttt{uf} = 336$
    - OPS: $\gamma =$ 1e-4, $\texttt{D} = 10$
  - Vapor Pressure Deficit Model
    - MW: $\texttt{uf} = 1, \texttt{ws} = 4320$
    - IW: $\texttt{uf} = 1$
    - OPS: $\gamma =$ 1e-3, $\texttt{D} = 1$
  - Specific Humidity Model
    - MW: $\texttt{uf} = 1, \texttt{ws} = 4320$
    - IW: $\texttt{uf} = 168$
    - OPS: $\gamma =$ 1e-5, $\texttt{D} = 30$
  - Airtight Model
    - MW: $\texttt{uf} = 2160, \texttt{ws} = 2160$
    - IW: $\texttt{uf} = 720$
    - OPS: $\gamma =$ 5e-5, $\texttt{D} = 10$
  - Wind Speed Model
    - MW: $\texttt{uf} = 1, \texttt{ws} = 168$
    - IW: $\texttt{uf} = 24$
    - OPS: $\gamma =$ 1e-4, $\texttt{D} = 10$
- **Base Classification Model**
  - Pressure Model
    - MW: $\texttt{uf} = 2160, \texttt{ws} = 8640$
    - IW: $\texttt{uf} = 720$
    - OPS: $\gamma =$ 5e-5, $\texttt{D} = 30$

- Temperature Model
  - MW: $\texttt{uf} = 1, \texttt{ws} = 4320$
  - IW: $\texttt{uf} = 168$
  - OPS: $\gamma =$ 1e-5, $\texttt{D} = 150$
- Saturation Vapor Pressure Model
  - MW: $\texttt{uf} = 336, \texttt{ws} = 4320$
  - IW: $\texttt{uf} = 720$
  - OPS: $\gamma =$ 1e-4, $\texttt{D} = 30$
- Vapor Pressure Deficit Model
  - MW: $\texttt{uf} = 1, \texttt{ws} = 168$
  - IW: $\texttt{uf} = 1$
  - OPS: $\gamma =$ 1e-5, $\texttt{D} = 70$
- Specific Humidity Model
  - MW: $\texttt{uf} = 1, \texttt{ws} = 2160$
  - IW: $\texttt{uf} = 2160$
  - OPS: $\gamma =$ 1e-5, $\texttt{D} = 50$
- Airtight Model
  - MW: $\texttt{uf} = 24, \texttt{ws} = 4320$
  - IW: $\texttt{uf} = 336$
  - OPS: $\gamma =$ 1e-3, $\texttt{D} = 10$
- Wind Speed Model
  - MW: $\texttt{uf} = 24, \texttt{ws} = 2160$
  - IW: $\texttt{uf} = 1$
  - OPS: $\gamma =$ 1e-5, $\texttt{D} = 10$.

C.3 HYPERPARAMETERS FOR CONTROL IN NUCLEAR FUSION CASE STUDY

The nuclear fusion dataset records measurements in 25 millisecond intervals. Therefore, in tuning hyperparameters, we design the search grid to represent lengths of time during which evolution of various plasma states are expected to be observable.

For each calibration method, the hyperparameters were tuned by optimizing parity calibration error (PCE, Section 3) on a validation dataset consisting of 1000 shot's worth of data, over the following grids:

- $\texttt{uf} \in [1, 2, 4, 8, 24]$, separately for MW and IW
- $\texttt{ws} \in [2, 8, 16, 24, 48, 60, 80, 100, 200]$, for MW
- $\gamma \in$ [1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2], for OPS
- $\texttt{D} \in [1, 10, 30, 50, 70, 100, 150, 200]$, for OPS

The tuned hyperparameters we used are as follows:

- MW: $\texttt{uf} = 1, \texttt{ws} = 60$
- IW: $\texttt{uf} = 8$
- OPS: $\gamma =$ 5e-3, $\texttt{D} = 150$.

D ONLINE PLATT SCALING ALGORITHM

Algorithm 1 Online Platt Scaling (based on Gupta and Ramdas [2023])

```
**Input:** $\mathcal{K} = \{(x, y) : \|(x, y)\|_2 \leq 100\}$, time horizon $H$, and initialization parameter $(a_1^{\text{OPS}}, b_1^{\text{OPS}}) = (1, 0) =: \theta_1 \in \mathcal{K}$
**Hyperparameters and default values:** $\gamma = 0.1$, $D = 1$, $A_0 = (1/\gamma D)^2\ \mathbf{I}_2$
**for** $t = 1$ **to** $H$ **do**
    Play $\theta_t$, observe log-loss $l(m^{\theta_t}(f(\mathbf{x}_t)), y_t)$ and its gradient $\nabla_t := \nabla_{\theta_t} l(m^{\theta_t}(f(\mathbf{x}_t)), y_t)$
    $A_t = A_{t-1} + \nabla_t \nabla_t^\intercal$
    Newton step: $\widetilde{\theta}_{t+1} = \theta_t - \frac{1}{\gamma} A_t^{-1} \nabla_t$
    Projection: $(a_{t+1}^{\text{OPS}}, b_{t+1}^{\text{OPS}}) = \theta_{t+1} = \arg\min_{\theta \in \mathcal{K}} (\widetilde{\theta}_{t+1} - \theta)^\intercal A_t (\widetilde{\theta}_{t+1} - \theta)$
**end for**
```

References

Jochen Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 135(643): 1512–1519, 2009.

Youngseog Chung, Ian Char, Han Guo, Jeff Schneider, and Willie Neiswanger. Uncertainty toolbox: an open-source library for assessing, visualizing, and improving uncertainty quantification. *arXiv preprint arXiv:2109.10254*, 2021.

Morris H DeGroot and Stephen E Fienberg. Assessing probability assessors: Calibration and refinement. Technical report, Carnegie Mellon University, 1981.

Chirag Gupta and Aaditya Ramdas. Online Platt scaling with calibeating. In *International Conference on Machine Learning*, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

James L Luxon. A design retrospective of the DIII-D tokamak. *Nuclear Fusion*, 42(5):614, 2002.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *International Conference on Machine Learning*, 2005.

David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *International Conference on Neural Networks*, 1994.