Transform Once
Supplementary Material

Table of Contents

Notation We report here a reference for notation used in main text and supplementary.

Symbol	Description
$\mathbb{R}$	Set of reals
$\mathbb{C}$	Set of complex numbers
$\mathbb{E}[x]$	Expected value of random variable x
$\mathbb{V}[x]$	Variance of random variable x
Σ_x	Covariance matrix of random variable x
tr	Trace operator for square matrices. $\mathrm{tr}(A) = \sum_n A_{nn}$
$\circ$	Composition of functions $f \circ g(x) = f(g(x))$
$*$	Conjugate transpose operator. $A^* = \bar{A}^\top$ where $\bar{A}$ has complex conjugated entries
$\wedge$	Outer product $u \wedge v = uv^*$ for $u, v \in \mathbb{C}^n$

A Proof of Theorem 3.2

A.1 Preliminary Lemmas

Lemma A.1 (Propagation of Uncertainty under DFT/DCT). *Let $X = Wx$ with $x \in \mathbb{R}^N$ and $W \in \mathbb{C}^{N\times N}$. Then*

$$\Sigma_X = W\Sigma_x W^*$$

Proof.

$$\begin{aligned}
\Sigma_X &= \mathbb{E}\left[(Wx - \mathbb{E}[Wx]) \wedge (Wx - \mathbb{E}[Wx])\right] \\
&= \mathbb{E}\left[W(x - \mathbb{E}[x]) \wedge W(x - \mathbb{E}[x])\right] \\
&= \mathbb{E}\left[W(x - \mathbb{E}[x])(x - \mathbb{E}[x])^\top W^*\right] \\
&= W\mathbb{E}\left[W(x - \mathbb{E}[x])(x - \mathbb{E}[x])^\top\right] W^* \\
&= W\Sigma_x W^*
\end{aligned}$$

□

Lemma A.2 (Propagation of Total Variance under DFT/DCT). *Let $X = Wx$ with $x \in \mathbb{R}^N$ and $W \in \mathbb{C}^{N\times N}$. Then*

$$\mathbb{V}[X] = \mathbb{V}[x]$$

Proof. Recalling that the total variance of a random variable is equal to the trace of its covariance matrix, i.e.

$$\mathbb{V}[x] = \text{tr}(\Sigma_x), \quad \mathbb{V}[X] = \text{tr}(\Sigma_X)$$

then

$$\text{tr}(\Sigma_x) = \text{tr}(\Sigma_X) \Leftrightarrow \mathbb{V}[X] = \mathbb{V}[x]$$

Recalling Lemma A.1 yields

$$\begin{aligned}
& \mathbb{V}[X] = \mathbb{V}[x] \\
\Leftrightarrow \quad & \text{tr}(\Sigma_x) = \text{tr}(W\Sigma_x W^*) \\
\Leftrightarrow \quad & \text{tr}(\Sigma_x) - \text{tr}(W\Sigma_x W^*) = 0 \\
\Leftrightarrow \quad & \text{tr}(\Sigma_x) - \text{tr}(\Sigma_x W^* W) = 0
\end{aligned}$$

Since the DCT/DFT matrix is orthonormal, i.e. $W^* = W^{-1}$ we have that

$$\text{tr}(\Sigma_x W^* W) = \text{tr}(\Sigma_x),$$

proving the result. □

Lemma A.3 (Gaussian initialization in rank–deficient linear layers). *Let $\hat{X} = S_m^\top A S_m X$ with $X \in \mathbb{R}^N$, $A \in \mathbb{C}^{m\times m}$ and $S_m \in \mathbb{C}^{m\times N}$,*

$$S_m = \dot{m}mN - m$$

If $\mathbb{E}[X_k] = 0$, $\mathbb{V}[X_k] = \sigma^2$ for all k the following hold:

i. for $k \geq m$

$$\mathbb{E}[\hat{X}_k] = 0, \quad \mathbb{V}[\hat{X}_k] = 0$$

ii. for $k < m$ and $\text{Re}(A_{ij}), \text{Im}(A_{ij}) \sim \mathcal{N}(0, \sigma_A^2)$

$$\mathbb{E}[\hat{X}_k] = 0, \quad \mathbb{V}[\hat{X}_k] = 2m\sigma^2\sigma_A^2$$

iii. for $k < m$ and $\text{Re}(A_{ij}) \sim \mathcal{N}(0, \sigma_A^2)$, $\text{Im}(A_{ij}) = 0$

$$\mathbb{E}[\hat{X}_k] = 0, \quad \mathbb{V}[\hat{X}_k] = m\sigma^2\sigma_A^2$$

Proof. Let $M = S_m^\top A S_m$. It holds,

$$M = \begin{bmatrix} A & \times \\ \times & \times \end{bmatrix} \in \mathbb{C}^{N\times N}$$

where"$\times$" are blocks of complex zeros. By expanding component–wise the layer computation, i.e.

$$\hat{X}_k = \sum_{j=0}^{N-1} M_{kj} X_j,$$

it holds that for $k < m$

$$\hat{X}_k = \sum_{j=0}^{m-1} A_{kj} X_j,$$

while $\hat{X}_k = 0$ for $k \geq m$. Hence $i.$ follows naturally from the latter and we focus on proving $ii.$ and $iii.$

Case $ii.$ The probability distribution of $\hat{X}_k$ is a sum of product distributions involving independent random variables A_{kj} and X_j. The first central moment is readily obtained

$$\mathbb{E}[\hat{X}_k] = \sum_{t=0}^{m-1} \mathbb{E}[A_{kj}]\mathbb{E}[X_j] = 0$$

since both $\mathbb{E}[X_k] = 0$ and $\forall\ k, j < m :\ \mathbb{E}[A_{kj}] = 0$. $\mathbb{V}[\hat{X}_k]$ can be then obtained by computing the variance of the product of two random variables, i.e.

$$\begin{aligned}
\mathbb{V}[\hat{X}_k] &= \sum_{j=0}^{m-1} \Big(\mathbb{V}[A_{kj}] + \cancel{\mathbb{E}[A_{kj}]^2})(\mathbb{V}[X_j] + \cancel{\mathbb{E}[X_j]^2}) - \cancel{\mathbb{E}[A_{kj}]^2\mathbb{E}[X_j]^2}\Big) \\
&= \sum_{j=0}^{m-1} \mathbb{V}[A_{kj}]\mathbb{V}[X_j] \\
&= \sum_{j=0}^{m-1} \sigma^2 \mathbb{V}[A_{kj}] \\
&= \sigma^2 \sum_{j=0}^{m-1} (\mathbb{V}[\mathrm{Re}(A_{kj})] + \mathbb{V}[\mathrm{Im}(A_{kj})]) \\
&= \sigma^2 \sum_{j=0}^{m-1} 2\sigma_A^2 = 2m\sigma^2\sigma_A^2
\end{aligned}$$

Case $iii.$ Similarly to the previous case we get

$$\begin{aligned}
\mathbb{V}[\hat{X}_k] &= \sigma^2 \sum_{j=0}^{m-1} \big(\mathbb{V}[\mathrm{Re}(A_{kj})] + \cancel{\mathbb{V}[\mathrm{Im}(A_{kj})]}\big) \\
&= \sigma^2 \sum_{j=0}^{m-1} \sigma_A^2 = m\sigma^2\sigma_A^2
\end{aligned}$$

□

A.2 Proof of Main Result

Theorem 3.2 (Variance Preserving (vp) Initialization)**.** *Let $\hat{x} = W^* S_m^\top A S_m W x$ be a k-space reduced-order layer and W is a normalized DCT-II transform. If $x \in \mathbb{R}^N$ is a random vector with*

$$\mathbb{E}[x] = \mathbb{0}, \quad \mathbb{V}[x] = \sigma^2 \mathbb{I}.$$

Then,

$$A_{ij} \sim \mathcal{N}\left(0, \frac{N}{m^2}\right) \Rightarrow \mathbb{V}[\hat{x}] = \mathbb{V}[x].$$

Proof. According to Lemma A.2, the total variance is preserved under the normalized DCT. Therefore, with $X = W\hat{x}$ and $\hat{X} = Wx$ we have

$$\mathbb{V}[X] = \mathbb{V}[x], \quad \mathbb{V}[\hat{X}] = \mathbb{V}[\hat{x}].$$

Using $\hat{X} = S_m^\top A S_m X$, we can find the condition under which the variance is preserved by the map $x \mapsto \hat{x}$:

$$\begin{aligned}
& \mathbb{V}[\hat{x}] = \mathbb{V}[x] \\
\Leftrightarrow \quad & \sum_{n=0}^{N-1} \mathbb{V}[\hat{x}_n] = \sum_{n=0}^{N-1} \mathbb{V}[x_n] \\
\Leftrightarrow \quad & \sum_{k=0}^{N-1} \mathbb{V}[\hat{X}_k] = \sum_{k=0}^{N-1} \mathbb{V}[X_k] \\
\Leftrightarrow \quad & \sum_{k=0}^{m-1} m\sigma^2\sigma_A^2 = \sum_{k=0}^{N-1} \sigma^2 \qquad \text{Lemma A.3} \\
\Leftrightarrow \quad & m^2\sigma^2\sigma_A^2 = N\sigma^2 \\
\Leftrightarrow \quad & \sigma_A^2 = \frac{N}{m^2}
\end{aligned}$$

Hence, initializing A by sampling its entries from a normal distribution with zero mean and variance N/m^2 is sufficient for preserving the variance under the reduced-order FDM layer, i.e.

$$A_{ij} \sim \mathcal{N}\left(0, \frac{N}{m^2}\right) \quad \Rightarrow \quad \mathbb{V}[\hat{x}] = \mathbb{V}[x],$$

proving the result. □

Corollary 3.2 (vp initialization for DFTs)**.** *Under the assumptions of Theorem 3.2, if W is a normalized DFT matrix we have* $\mathrm{Re}(A_{ij}), \mathrm{Im}(A_{ij}) \sim \mathcal{N}(0, \frac{N}{2m^2}) \Rightarrow \mathbb{V}[\hat{x}] = \mathbb{V}[x]$.

Proof. The proof follows directly from the one of Theorem 3.2 using the fact that since the DFT's k-space is complex ($\mathcal{D}_k \equiv \mathbb{C}^N$) as $W \in \mathbb{C}^{N\times N}$, the weights are typically chosen complex, i.e. $A \in \mathbb{C}^{m\times m}$. Therefore, in this case $\mathbb{V}[\hat{X}] = 2m\sigma^2\sigma_A^2$ according to Lemma A.3. □

Corollary A.1 ((vp) initialization with diagonal layers)**.** *Under the assumptions of Theorem 3.2, if A is diagonal s.t $\forall i \neq j : A_{ij} = 0$, we have $A_{ii} \sim \mathcal{N}\left(0, \frac{N}{m}\right) \quad \Rightarrow \quad \mathbb{V}[\hat{x}] = \mathbb{V}[x]$.*

Proof. The proof follows directly from Lemma A.3

$$\begin{aligned}
\mathbb{V}[\hat{X}_k] &= \sum_{j=0}^{m-1} \mathbb{V}[A_{kj}]\mathbb{V}[X_j] \\
&= \mathbb{V}[A_{kk}]\mathbb{V}[X_k] \\
&= \sigma^2 \left(\mathbb{V}[\mathrm{Re}(A_{kk})] + \cancel{\mathbb{V}[\mathrm{Im}(A_{kk})]}\right) \\
&= \sigma^2\sigma_A^2
\end{aligned}$$

leading to the condition

$$\begin{aligned}
& \mathbb{V}[\hat{x}] = \mathbb{V}[x] \\
\Leftrightarrow \quad & \sum_{k=0}^{m-1} \sigma^2\sigma_A^2 = \sum_{k=0}^{N-1} \sigma^2 \\
\Leftrightarrow \quad & m\sigma^2\sigma_A^2 = N\sigma^2 \\
\Leftrightarrow \quad & \sigma_A^2 = \frac{N}{m}
\end{aligned}$$

□

The layer structure treated by A.1 is common among many FDMs, e.g. FNOs in (Li et al., 2020).

B Additional Details

Broader impact FDMs are widely used in the context of learning to predict the evolution of dynamical systems. The model class presented in this work, `T1`, provides an accessible way to

train and evaluate large-scale FDMs, reducing memory overhead and overall training times. When predicting the solution of e.g. a *partial differential equation* (PDE), care should be taken especially when the prediction is used to inform downstream decision making, as many systems are optimally predictable only for a certain time scale (Strogatz, 2018, pp. 366). We anticipate a potential positive environmental impact from the adoption of `T1` as a replacement for the largest FDMs currently in use.

Experimental setup Experiments have been performed on an NVIDIA© DGX workstation equipped with a 128 threads AMD© EPYC 7742 CPU, 512GB of RAM and four NVIDIA© A100 GPUs. The main software implementation has been done within the `PyTorch` (Paszke et al., 2017) ecosystem building upon the `pytorch-lightning` (Falcon et al., 2019) framework.

Common experimental settings

B.1 Incompressible Navier–Stokes

Dataset We use data generated in (Li et al., 2020) in the form of pairs of initial conditions and solutions of the incompressible Navier-Stokes equations in vorticity form solved with a pseudospectral method. The dataset [6] is comprised rollouts of solutions as images of resolution 64.

Models and training The training configuration is shared by all models:

```yaml
datamodule:
    ntrain: 1000
    ntest: 200
    batch_size: 64
    history_size: 1
train:
    optimizer:
        type: AdamW
        learning_rate: 1e-3
        weight_decay: 1e-4
    scheduler:
        type: Step
        step_size: 100
        gamma: 0.5
        scheduler_interval: epoch
loss_fn: RelativeL2Loss
```

For the high viscosity ($1e^{-3}$) setting, the models are trained to predict the solution at time $T = 50$ seconds directly, without producing rollouts and supervising the model with solutions at times between 0 and 50. Crucially, this ensures that the task is much more challenging than that of (Li et al., 2020), where for a single training sample the entire rollout is used as supervision. For the low viscosity setting ($1e^{-4}$), target times are $T = 15$ seconds.

Model configurations are given below:

FNO

```yaml
modes: 24
nlayers: 6
width: 32
```

T1

```yaml
modes: 24
nlayers: 6
width: 48
```

FFNO

```yaml
modes: 32
nlayers: 10
width: 82
```

where each layer in a model shares the same structure. In FNOs and FFNOs, we employ a regular FDM layer following (Li et al., 2020; Tran et al., 2021) with k-space convolutions and residual connections given by n-space layers (pointwise convolutions for FNOs, dense for FFNOs). `T1` uses a similar layer without n-space residual paths. The differences in number of layers and width have

[6] Data can be downloaded here: Google Drive link. High viscosity: `NavierStokes_V1e-3_N5000_T50`, Low viscosity: `NavierStokes_V1e-4_N10000_T30`.

been introduced to keep parameter counts comparable. At a given channel width, FNOs require the largest number of parameters due to k-space convolutions on complex numbers given by the DFT coefficients. Although FFNOs (Tran et al., 2021) are most parameter efficient due to parameter sharing, we found them unable to tackle the task and produce high-quality predictions.

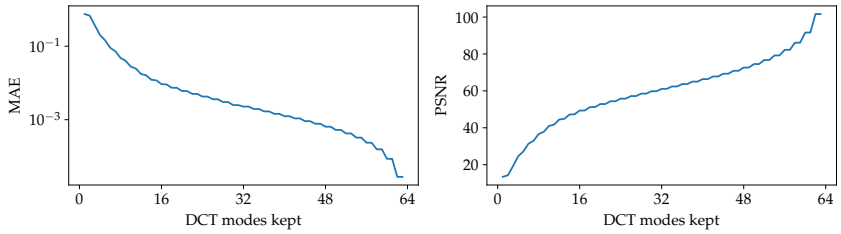

Figure B.1: Incompressible Navier-Stokes: metrics vs number of DCT modes (i.e. m elements) kept (i.e. not pruned).

`T1+` employs a UNet on the patch constructed by the elements of the k-space kept, and shares its structure with `T1` otherwise. The `vp` parameter initialization scheme in `T1` is applied only to the first layer performing the truncation in k-space, not to the following layers which use standard Kaiming initialization He et al. (2015). In FNOvp the scheme is applied to all layers.

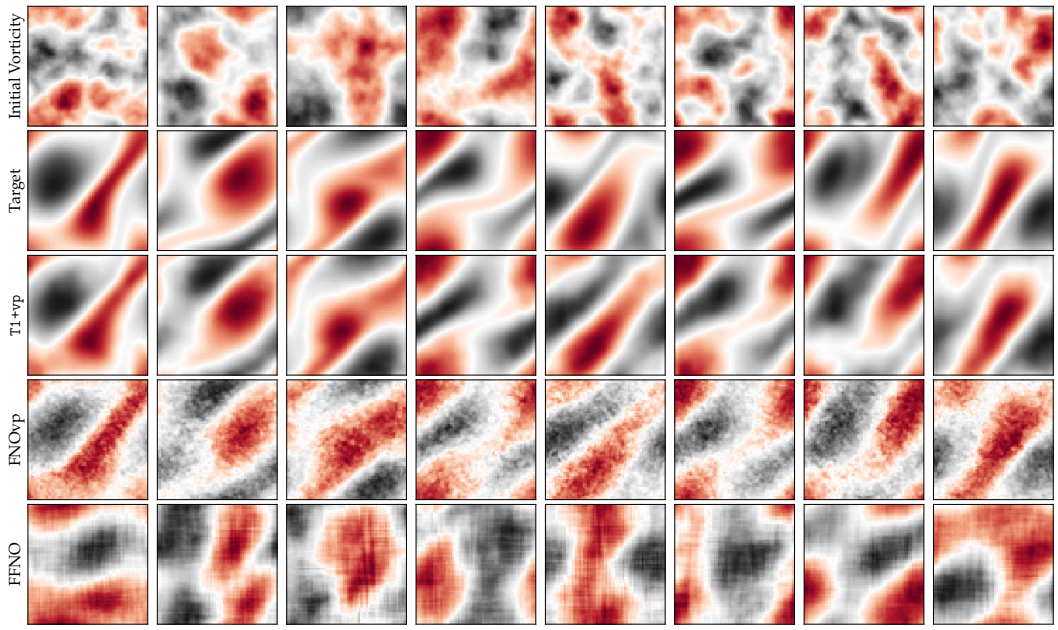

Figure B.2: Initial conditions, ground truth solutions at time $T = 50$ seconds, and models predictions for incompressible Navier-Stokes in vorticity form (high viscosity of $1e^{-3}$). `T1` reduces solution error w.r.t FNOs by over 20% and FFNOs by over 40%. A single forward pass of `T1` models is on average $2\times$ faster than FNO and $10\times$ than FFNOs.

Hyperparameter tuning We start with the basic model structure of FNOs as detailed (Li et al., 2020) and perform a basic hyperparameter search on a small slice of the training set, with the goal of ensuring proper convergence of a model. We did not find the number of layers to have a significant impact on convergence. Width plays an important role and is best kept above 24.

Scaling laws We use the same settings as the main experiment, repeating separate training runs for the low viscosity setting. In particular, we increase the dataset size for each set of runs by a factor of 2: $1024, 2048, 4096, 8192$. The total number of epochs is kept fixed, so that more iterations are performed for larger datasets. The same test set of size 200 is used in all cases.

Further comments Additional predictions are provided in Fig. B.2. Fig. B.1 shows the approximation error on the Navier-Stokes solutions due to truncation at different number of k-space elements m.

B.2 Flow Around Airfoils

Dataset We use a slice of the dataset introduced by Thuerey et al. (2020) in the form of 11000 training pairs of initial conditions and solutions. The solutions are obtained via `OpenFOAM` (Jasak et al., 2007) SIMPLE, a steady-state solver for incompressible and turbulent flows. In particular, the initial conditions are specified as freestream velocities over the domain (two-directional components), in addition to a specification of the airfoil in point cloud format. Delaunay triangulation is used for mesh generation.

After simulation, data is provided as initial condition and steady-state solution pairs. The initial condition is a three channel 128×128 image: two channels for freestream velocities and one for the airfoil mask. The solution is a three channel 128×128 image: a velocity field and a scalar pressure field. All data is normalized using training set statistics.

Models and training Training configuration is given as

```yaml
datamodule:
    ntrain: 8000
    nval: 2000
    ntest: 1000
    batch_size: 64
train:
    optimizer:
        type: AdamW
        learning_rate: 1e-3
        weight_decay: 1e-4
    scheduler:
        type: Step
        step_size: 100
        gamma: 0.6
        scheduler_interval: epoch
loss_fn: RelativeL2Loss
```

The baseline UNet matches the architecture of (Thuerey et al., 2020) (DFPNet). The FNO architecture is comprised of a standard stack of FDM layer as discussed in B.1. The k-space UNet in `T1+` has the same structure as a DFPNet.

FNO

```yaml
modes: 24
nlayers: 6
width: 48
```

DFPNET

```yaml
channel_exponent: 6
```

T1+

```yaml
modes: 100
channel_exponent: 5
```

Hyperparameter tuning This is an example of a dataset where the k-space is full due to discontinuity in the solution given by the airfoil mask.

We use the training and validation sets to inspect the k-space and set m to 100 for the irreducible loss term to be sufficiently small as shown in Fig. B.5. We swept over m for FNOs and found larger than 24 to perform worse, likely due to k-space convolution being sufficient to capture higher frequency components. We observe DFPNets with larger channel exponents perform worse due to overfitting.

Further comments A sample of predictions is given in Fig. B.3. Fig. B.4 shows the n-space and corresponding DCT k-space of a data point. As can be observed, the k-space is structured but full due to the discontinuity caused by the airfoil mask. Fig. B.5 shows the approximation error on solution fields due to truncation in k-space at different m. In this task, the DCT is more efficient, given a budget of modes to keep, as it yields lower errors. This error provides a theoretical lower

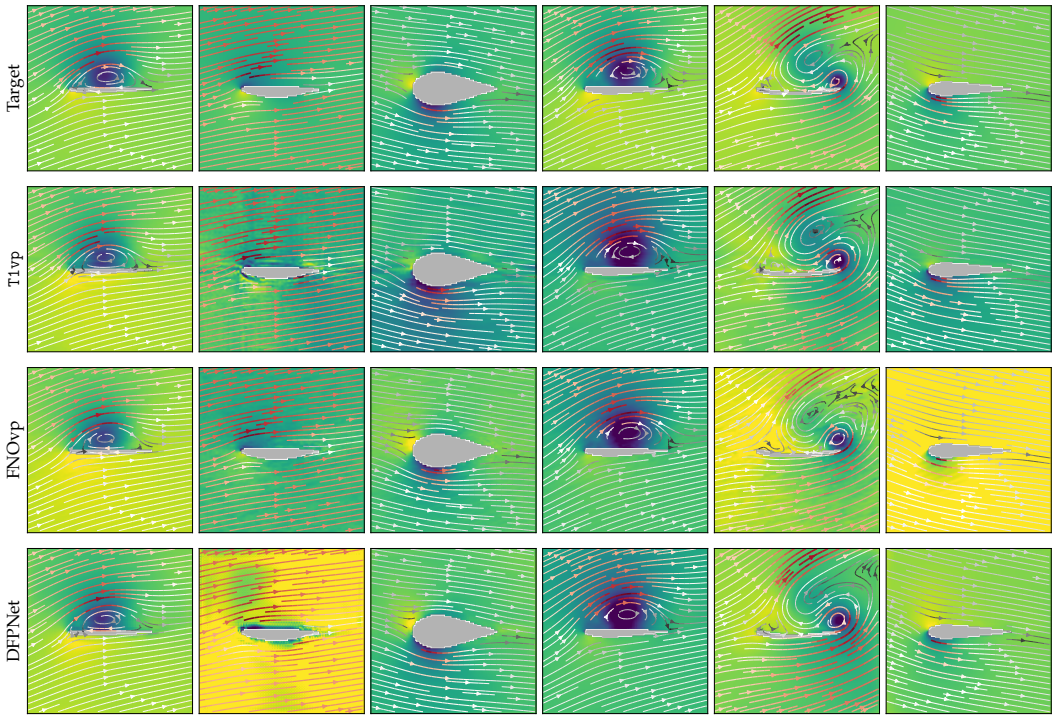

Figure B.3: Ground truth solutions and predictions with different airfoil designs and angles of attack of the flow. The background color is the scalar pressure value while the vector field represents the velocity field: arrow colors indicate its "strength" i.e. 2-norm.

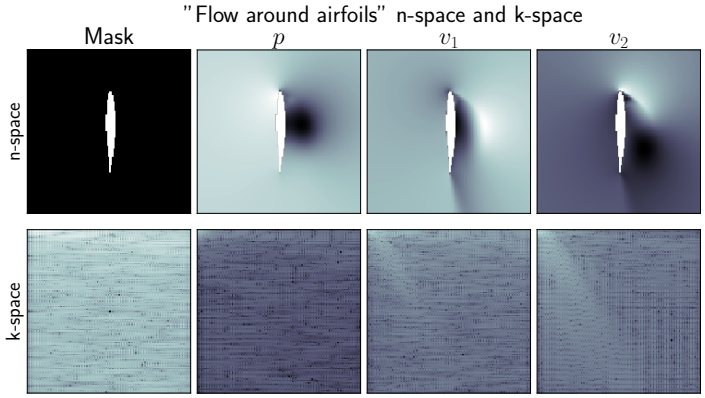

Figure B.4: Flow around airfoils: example of n-space: input mask, output pressure p and velocity field v_1, v_2. Below, the corresponding DCT k-space in abs-log i.e. $\log\left(|\mathcal{T}(x)|\right)$ to highlight its structure.

bound for the predictive error achievable by a `T1` model with a given budget, reachable only if the `T1` predicts the first m modes perfectly.

The vertical line indicates the budget used for the main text `T1` experiments ($m = 100$), and the horizontal one the test N-MSE achieved. Various segments of the vertical line indicate reducible and irreducible components of the loss as discussed in § 3.1. The theoretical limit at $m = 100$ is well below what has been empirically achieved by `T1` and other models. Indeed, the irreducible loss is an order of magnitude smaller than what the best model (including non-reduced-order variants) achieves on the task.

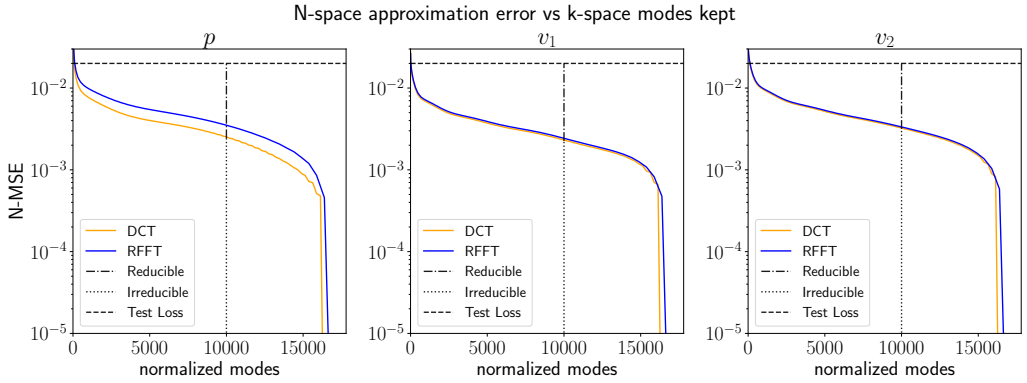

Figure B.5: Average approximation error (N-MSE) due to truncation in k-space at different number of elements m for the *flow around airfoils* dataset. In blue, the real FFT k-space, in orange the regular DCT k-space. On the x-axis, the normalized cost for a number of modes m: for DCTs, since the k-space is real, truncation at m modes requires m^2 floats, for real FFTs with complex k-space and conjugacy the cost in floats is $4m^2$. The vertical line indicates the budget used for `T1` used in this task ($m = 100$), while the horizontal line is the test N-MSE achieved.

B.3 Turbulent Smoke

Dataset We employ for this experiment the ScalarFlow dataset introduced in (Eckert et al., 2019) which is available online under the Creative Commons license CC-BY-NC-SA 4.0[7]. Eckert et al. (2019) created an environment for controlling the release of smoke plumes: a fog machine generated fog inside of a container; the fog was then heated up by a heating cable and a valve controlled its release. Data was captured via multiple calibrated cameras in high resolution at 60 fps (frames per second) for 150 frames.

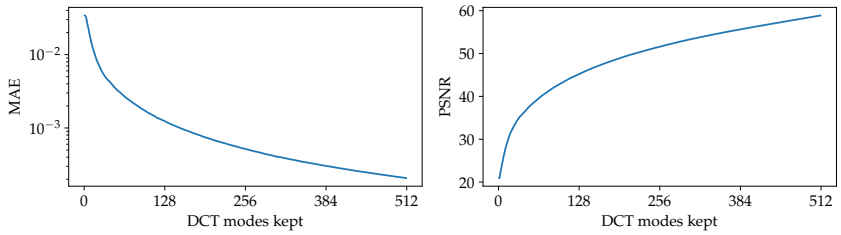

Figure B.6: ScalarFlow dataset: reconstruction error versus number of kept DCT modes.

The dataset contains 3D reconstructions of the smoke plumes and 2D input and rendered images: input images are used by Eckert et al. (2019) to solve an optimization problem in which the goal is to generate a 3D reconstruction that minimizes the difference between input and rendered images. 2D input images are obtained directly from raw data on which only post-processing is applied by (Eckert et al., 2019) in the form of gray scaling and denoising: these are saved in compressed `numpy` (Harris et al., 2020) arrays named `imgsTarget_000xxx.npz`. Each resulting frame comprises 5 different camera views 600×1062 in size. Since we want to use `T1` on high-resolution experimental data, we directly utilize the central camera view of these input images in our learning task without any further downsampling or data processing. Similarly to (Lienen and Günnemann, 2022), we divide the 104 recordings into the first 64 for training and use the remaining 20 for validation and 20 for testing.

Data is normalized to the $[0, 1]$ range based on training dataset statistics.

Hyperparameter selection and tuning We performed a search on the most representative hyperparameters. One of the most important hyperparameters to choose from is the number of DCT modes to keep, i.e. first m elements in k-space. We note that for simplicity as well as for compatibility with the UNet inside of `T1+`, we consider a *square* mode pruning, i.e. we keep the same

[7] ScalarFlow dataset download: https://ge.in.tum.de/publications/2019-scalarflow-eckert/

number of frequencies on both height and width of the image and refer to the modes kept in both dimensions as m. Fig. B.7 and Fig. B.6 show trends of DCT modes in terms of errors and visual quality: while the first modes m contribute the most to the quality of the representation in n-space, the last elements contribute only to high-frequency details whose effect is minor on the overall reconstruction. Thus, we set `T1+` to $m = 224$ and consequently `T1` to $m = 512$ to have comparable model sizes. We set $m = 48$ for FNO due to memory and model size limitations, noting that its residual connections effectively enlarge the training spectrum to all possible frequencies as shown in Fig. 4.3. Similarly to other experiments (B2), we observe raising m in FNO to not significantly improve predictive error, even when the additional k-space elements would include a larger portion of the dataset.

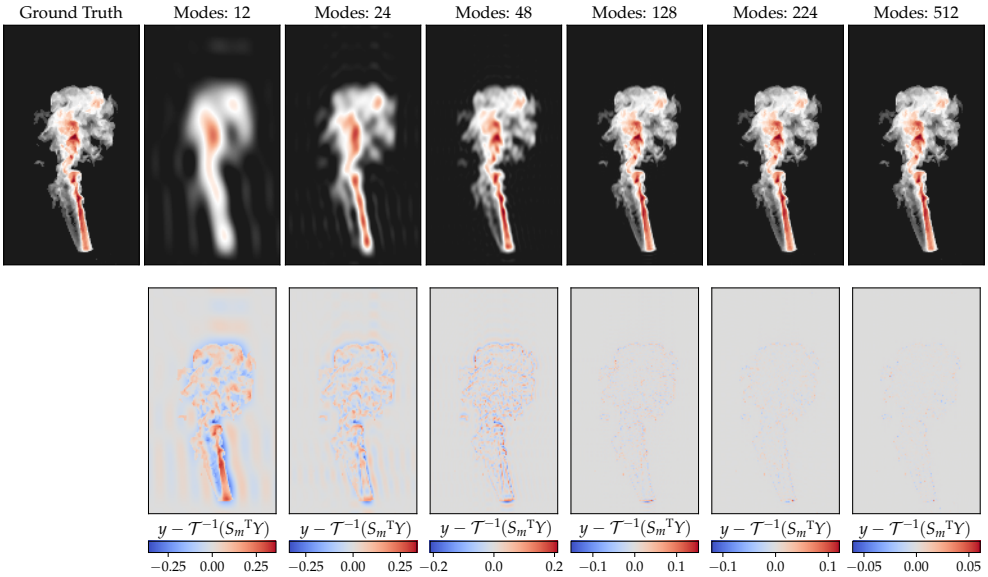

Figure B.7: **[Top]** Visual comparison of ScalarFlow frames with changing number of DCT modes kept (i.e. first m elements) . **[Bottom]** Error between the ground truth frame y and its inverse transformation after mode pruning from k-space back to n-space. As expected, the first few k-space elements are crucial to minimizing reconstruction errors, with higher frequency components contributing minimally.

We also experiment with different iterative rollout update strategies as in (Pfaff et al., 2020). We consider the time step Δt to be unitary, i.e. $\Delta t = 1$, given that the training frames are sampled consistently at 60 fps. We call 0-order integration an update of the type: $x_{t+1} = h_\theta(x_t; x_{t-1}, \ldots, x_{t-H})$ in which h_θ denotes a learned model which takes as inputs the current state x_t and optionally a history of size H of past states $x_{t-1}, \ldots, x_{t-H}$ and directly predicts the next state x_{t+1}. A 1-order integrator performs the following update: $x_{t+1} = x_t + h_\theta(x_t; \cdot)$, in which the model predicts the state update, i.e. the *velocity*, similarly to an Euler step. A 2-order integrator, also known as basic Störmer—Verlet (Verlet, 1967) can be written as following: $x_{t+1} = 2x_t - x_{t-1} + h_\theta(x_t; \cdot)$; the model h_θ predicts the *acceleration* of the system. We empirically found the zero-order integration to be more prone to generating artifacts with slower convergence, which may be because the model has to directly predict the next step with no "help" from the current step information. We found models trained with first-order integrators to have lower predictive errors than those trained with second-order ones, and we thus use it in all the experiments. As for the history size, we selected $H = 1$ since it provided noticeable benefits compared to $H = 0$, in which the model has no way of knowing previous states and thus inferring velocities. Larger history sizes did not seem to provide any improvements and only made the models larger as also noted in (Pfaff et al., 2020).

Mode selection We further show in Fig. B.8 the effect of simple low-pass filtering of lowest m frequency modes and $\texttt{top}_k(m)$ mode selection in pixel space reconstruction (as a fraction of total pixes, i.e., 600×1062). The latter achieves better reconstruction results with the same number of parameters.

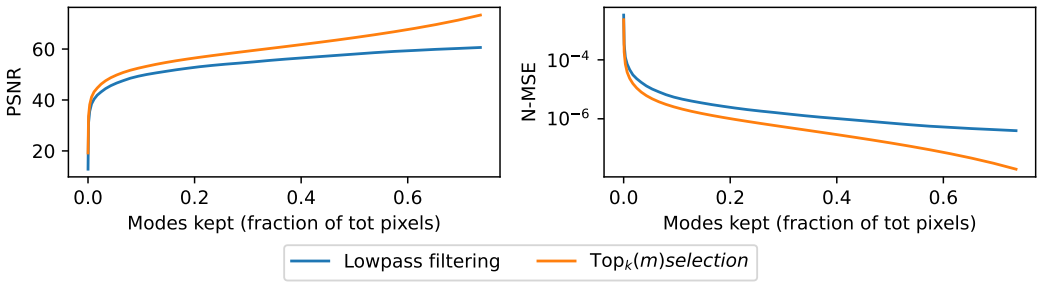

Figure B.8: Reconstruction errors in pixel space of low-pass filtering of the lowest m frequency modes vs $\texttt{top}_k(m)$ selection on a single frame of ScalarFlow.

Models and training All models share the configuration for training:

```yaml
datamodule:
    ntrain: 64
    nval: 20
    ntest: 20
    batch_size: 1
    history_size: 1
    target_steps_train: 3
    target_steps_val_test: 10
train:
    optimizer:
        type: AdamW
        learning_rate: 1e-3
        weight_decay: 1e-4
    scheduler:
        type: CosineAnnealingWarmRestarts
        T_0: 32
        step_size: 1
        scheduler_interval: step
loss_fn: RelativeL2Loss
```

Where we used the implementation in `PyTorch` of the cosine annealing schedule with warm restarts[8]. The FNO architecture comprises a standard stack of FDM layers as discussed in B.1. The k–space UNet in `T1+` (and in its vp variant) has the same structure as a DFPNet.

FNO

```yaml
modes: 48
nlayers: 4
width: 48
```

T1

```yaml
modes: 512
nlayers: 4
width: 8
```

T1+

```yaml
modes: 224
nlayers: 1
width: 4
channel_exponent: 7
```

where we note that all models employ `GeLU` (Hendrycks and Gimpel, 2016) activation functions between inner layers.

Analysis of results Table B.1 provides a larger version of the table in the main text, including 1-step mean absolute errors (MAE). We note that while FNO produces smaller errors in one-step predictions, it quickly accumulates larger errors in extrapolation. Fig. B.9 shows mean errors in k-space of FNO vs `T1` and `T1+`. `T1` models demonstrate smaller overall errors and lower maxima compared to the FNO.

[8]We used the scheduler `torch.optim.lr_scheduler.CosineAnnealingWarmRestarts` with the number of iterations for the first restart $T\_0 = 32$. All other hyperparameters are the same as in the reference implementation.

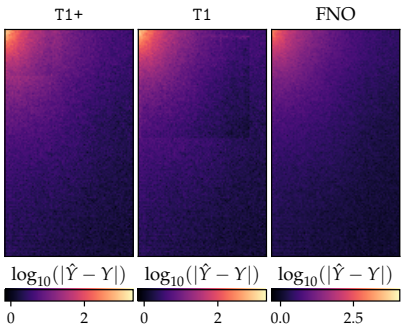

Figure B.9: Mean log-absolute values of predictions in k-space (DCT-II) of a 20-elements batch in the test dataset. Although `T1` is limited to $m = 512$ and `T1+` to $m = 224$ k-space elements (visible as square "shadows" in the error plots), its predictions are overall more physically accurate in n-space.

C Properties of Frequency Domain Models

C.1 Preliminary Results

Lemma C.1 (Finite cosine series convergence)**.** *Let $k \in \mathbb{N}^+$, $N \in \mathbb{N}^+$ with $N \geq 2$. The following holds*

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi kn}{N}\right) = 0. \tag{4}$$

Proof. Let us substitute $z = \frac{2\pi k}{N}$ for simplicity. We can rewrite the finite series as follows

$$y = \sum_{n=0}^{N-1} \cos(zn) = \cos(z \cdot 0) + \cos(z \cdot 1) + \cdots + \cos(z(N-1)). \tag{5}$$

By multiplying both sides of the equation by $2\sin(z)$ we obtain

$$2\sin(z)y = 2\cos(z \cdot 0)\sin(z) + 2\cos(z \cdot 1)\sin(z) + \cdots + 2\cos(z(N-1))\sin(z). \tag{6}$$

By applying the following trigonometric identity

$$2\cos(\alpha)\sin(\beta) = \sin(\alpha + \beta) - \sin(\alpha - \beta), \tag{7}$$

Table B.1: Full benchmark on the ScalarFlow dataset over 5 runs with different random seeds. N-MSE refers to 10-step test rollouts. `T1+vp` generates more stable rollouts while requiring a fraction of FNO's training time.

Method	Param (M)	Size (MB)	Time (hrs)	N-MSE ($\times 10^{-1}$)
FNO	84.9	339	32.4	2.32 ± 0.02
`T1`	83.9	335	8.1	2.39 ± 0.02
`T1+`	67.8	271	4.7	2.56 ± 0.16
`T1+vp`	67.8	271	4.7	2.28 ± 0.09

Equation (6) becomes

$$\begin{aligned} 2\sin(z)y = 2\sin(z) \\ &+ \sin(z+z) - \sin(z-z) \\ &+ \sin(2z+z) - \sin(2z-z) \\ &+ \sin(3z+z) - \sin(3z-z) \\ &+ \dots \\ &+ \sin((N-1)z+z) - \sin((N-1)z-z) \end{aligned} \tag{8}$$

where terms on the right-hand side cancel out pairwise[9]. After cleanup, we are left with the following

$$2\sin(z)y = \sin(z) + \sin((N-1)z) + \sin(Nz). \tag{9}$$

By substituting back $z = \frac{2\pi k}{N}$ we obtain

$$\begin{aligned} 2\sin\left(\frac{2\pi k}{N}\right)\cdot y &= \sin\left(\frac{2\pi k}{N}\right) + \sin\left((N-1)\frac{2\pi k}{N}\right) + \sin\left(N\frac{2\pi k}{N}\right) \\ &= \cancel{\sin\left(\frac{2\pi k}{N}\right)} - \cancel{\sin\left(\frac{2\pi k}{N}\right)} + \cancel{\sin(2\pi k)}^{0}, \end{aligned} \tag{10}$$

where we used the trigonometric identity $\sin(-\alpha) = -\sin(\alpha)$. After dividing by the factor $2\sin\left(\frac{2\pi k}{N}\right)$, we readily obtain the result $y = 0$.

□

Lemma C.2 (Finite squared cosine series convergence)**.** *Let $k \in \mathbb{N}^+$, $N \in \mathbb{N}^+$ with $N \geq 2$. The following holds*

$$\sum_{n=0}^{N-1} \cos^2\left(\frac{2\pi kn}{N}\right) = \frac{N}{2}. \tag{11}$$

Proof. We recall the following trigonometric identity

$$\cos^2(\alpha) = \frac{1+\cos(2\alpha)}{2}. \tag{12}$$

Let us substitute $z = \frac{2\pi k}{N}$ for simplicity. We can thus rewrite the finite series as follows

$$\begin{aligned} \sum_{n=0}^{N-1} \cos^2(zn) &= \sum_{n=0}^{N-1} \frac{1+\cos(2zn)}{2} \\ &= \frac{1+\cos(2z\cdot 0)}{2} + \frac{1+\cos(2z\cdot 1)}{2} + \dots + \frac{1+\cos(2z(N-1))}{2} \\ &= \frac{N}{2} + \frac{1}{2}\left[\cos(2z\cdot 0) + \cos(2z\cdot 1) + \dots + \cos(2z(N-1))\right] \\ &= \frac{N}{2} + \frac{1}{2}\cancel{\sum_{n=0}^{N-1}\cos(2zt)}^{0} \quad \text{(from Lemma C.1)} \\ &= \frac{N}{2}. \end{aligned} \tag{13}$$

□

[9] Alternatively, we could think about the finite cosine series itself as the summation of N cosine terms on a circle with terms from 0 up to $N-1$ – scaled by k, which does not affect the result. The cosine terms then cancel out in a pair–wise fashion (or in triplets, depending on even or odd N).

C.2 Statistics Under Fourier Transform

There are various ways to show how probability measures and moments propagated under frequency domain transforms. We showcase two additional proof methods based on change of variables or explicit computation for simple input distributions.

Lemma C.3 (Central moment preservation under unitary linear operators)**.** *Let $x \sim p_x(x)$, $x \in \mathbb{C}$ and let $\mathcal{T}$ be a unitary linear operator. With $X = \mathcal{T}(x)$, it holds*

$$p_X(X) = p_x(\mathcal{T}^{-1}(X))$$

Proof. The result follows immediately from the change of variables formula

$$\begin{aligned} p_X(X) &= p_x(\mathcal{T}^{-1}(X)) \det \left[\frac{\mathrm{d}}{\mathrm{d}X} \mathcal{T}^{-1}(X) \right] \\ &= p_x(x), \end{aligned}$$

being $\partial_X \mathcal{T}(X)$ the Jacobian of $\mathcal{T}$, since

$$\det \frac{\mathrm{d}}{\mathrm{d}X} \mathcal{T}^{-1}(X) = \det \frac{\mathrm{d}}{\mathrm{d}X} \mathcal{T}(X) = 1.$$

□

Lemma C.4 (Variance preservation under unitary linear operators)**.** *Let $x \in \mathbb{R}^N$ be a random vector with*

$$\mathbb{E}[x] = \mathbb{0}, \quad \mathbb{V}[x] = \sigma^2 \mathbb{I}.$$

with $\mathcal{T}$ a normalized DFT. If $X = \mathcal{T}(x)$, it holds

$$\forall k, n: \quad \mathbb{E}[X_k] = \mathbb{E}[x_n] = 0 \quad \textit{and} \quad \mathbb{V}[X_k] = \mathbb{V}[x_n] = \sigma^2$$

Proof. Let x be real-valued input and distributed according to

$$p_{\mathrm{Re}(x)} = \mathcal{N}(0, \sigma^2 \mathbb{I}) \quad p_{\mathrm{Im}(x)} = \delta(\mathbb{0}).$$

Consider a single element of X

$$X_k = \sum_{n=0}^{N-1} v_n$$

with

$$v_n = \frac{1}{\sqrt{N}} e^{\frac{2\pi j n k}{N}} x_n = \frac{1}{\sqrt{N}} \cos \frac{2\pi n k}{N} x_n + j \frac{1}{\sqrt{N}} \sin \frac{2\pi n k}{N} x_n.$$

For clarity, we will treat the real part $\mathrm{Re}(X_k)$ first.

$$\mathrm{Re}(v_n) = \frac{1}{\sqrt{N}} \cos \frac{2\pi n k}{N} \mathrm{Re}(x_n)$$

and

$$\begin{aligned} \mathbb{E}[v_n] &= \frac{1}{N} \cos^2 \frac{2\pi n k}{N} \mathbb{E}[x_n] = 0 \\ \mathbb{V}[v_n] &= \frac{1}{N} \cos^2 \frac{2\pi n k}{N} \mathbb{V}[x_n] = \frac{\sigma^2}{N} \cos^2 \frac{2\pi n k}{N} \end{aligned}$$

where we have used the fact that

$$\frac{1}{\sqrt{N}} \sin \frac{2\pi n k}{N} \mathrm{Im}(x_n) = 0.$$

Thus,

$$\begin{aligned} \mathbb{E}[\mathrm{Re}(X_k)] &= 0 \\ \mathbb{V}[\mathrm{Re}(X_k)] &= \sum_{n=0}^{N-1} \frac{\sigma^2}{N} \cos^2 \frac{2\pi n k}{N} \end{aligned}$$

We observe that (a) the first central moment is preserved and (b) the variance term can be simplified as

$$\begin{aligned}\mathbb{V}[\mathrm{Re}(X_k)] &= \sum_{n=0}^{N-1} \frac{\sigma^2}{N} \cos^2 \frac{2\pi nk}{N} \\ &= \frac{\sigma^2}{N} \sum_{n=0}^{N-1} \cos^2 \frac{2\pi nk}{N} \\ &= \frac{\sigma^2}{N} \frac{N}{2} \quad \text{(from Lemma C.2)} \\ &= \frac{\sigma^2}{2}\end{aligned}$$

We follow a similar procedure for $\mathrm{Im}(X_k)$, arriving at

$$\begin{aligned}\mathbb{E}[\mathrm{Im}(X_k)] &= 0 \\ \mathbb{V}[\mathrm{Im}(X_k)] &= \sum_{n=0}^{N-1} \frac{\sigma^2}{N} \sin^2 \frac{2\pi nk}{N}\end{aligned}$$

where the variance again simplifies to

$$\sum_{n=0}^{N-1} \frac{\sigma^2}{N} \sin^2 \frac{2\pi nk}{N} = \frac{\sigma^2}{2}$$

Since $X_k = \mathrm{Re}(X_k) + j\,\mathrm{Im}(X_k)$,

$$\begin{aligned}\mathbb{E}[X_k] &= \mathbb{E}[\mathrm{Re}(X_k)] + j\mathbb{E}[\mathrm{Im}(X_k)] = 0 + j0 = 0 \\ \mathbb{V}[X_k] &= \mathbb{V}[\mathrm{Re}(X_k)] + \mathbb{V}[\mathrm{Im}(X_k)] = \sigma^2\end{aligned}$$

□

A similar argument can be developed using basic properties of circular-symmetry of complex Normals.

It is critical that the normalization factor $\frac{1}{\sqrt{N}}$ be included in W in order to preserve the variance of $\mathbb{V}[X]$.

Indeed, normalization factors used in different conventions lead to different results

$$\begin{aligned}&\text{forward factor } \frac{1}{N} \implies \mathbb{V}[X_k] = \frac{\sigma^2}{N} \\ &\text{backward factor } 1 \implies \mathbb{V}[X_k] = N\sigma^2\end{aligned}$$

As N can easily be in the order of hundreds or thousands for generic signals, explosion of variance can be an issue if the orthogonalization factor $\frac{1}{\sqrt{N}}$ is not applied to W.