Learning to generate physical ocean states: Towards hybrid climate modeling

Etienne Meunier

Meunier, Kamm, Gachon, Lguensat, & Deshaye. Learning to generate physical ocean states: Towards hybrid climate modeling. ICLRw, 2025

Gorce, Ollier & Meunier Physically Consistent Sampling For Ocean Model Initialization. NEURIPSw, 2025

08-10-2025

Motivation: Spin-Up (Mostly)

Spin-up: the time taken to reach **statistical equilibrium**

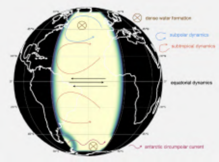
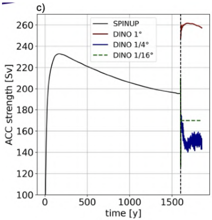
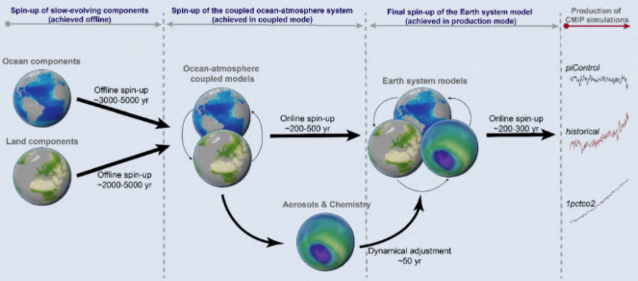


Figure: DINO [a]





0.5M cpu-hours / model 5 spin-up/year = **2.5M cpu-hours/year**

Spin-up cost at IPSL

[a] https://github.com/vopikamm/DINO

Method

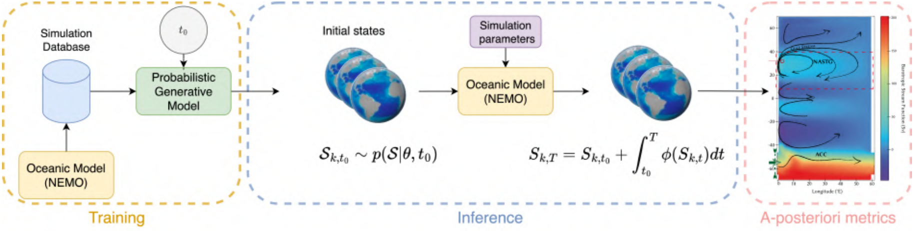
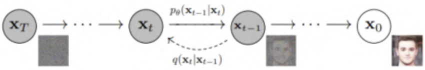


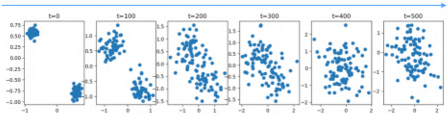
Figure: Pipeline of the training and evaluation protocol. From left to right: training of the diffusion model using a database of stable states produced by our oceanic model, generation of initialization states from our diffusion model and temporal integration using numerical simulation, then evaluation of physical consistency on simulated trajectories.

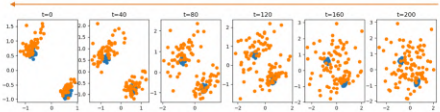
Introduction - Diffusion Model [HJA20]



$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t \mid \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I})$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1} \mid \mu_\theta(x_t, t), \beta_t \mathbf{I})$$





Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
$\approx \nabla_x \log p(x)$
5: **end for**
6: **return** $\mathbf{x}_0$

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33, 6840-6851.

Training

```
nnet =  NoiseNet()
opt = optim.SGD(nnet.parameters(), lr=0.001)
for epoch in range(epochs) :
    ts = torch.randint(0, T, (sp0.shape[0],1)) # randomly get time steps
    noisy, epsilon = forward(sp0,    1- alpha_bars[ts]) # forward process
    loss = ((nnet(noisy, ts/T) - epsilon)**2).sum() # predict noise
    opt.run(loss) # gradient descent
```

```
def forward(x, beta) :
    epsilon = torch.randn_like(x)
    noisy = x*torch.sqrt(1-beta) +  epsilon* torch.sqrt(beta)
    return noisy, epsilon

T = 500
betas =  torch.linspace(1e-6, 5e-3, T)
alpha_bars = torch.cumprod(1 - betas, dim=0)
```

Generation

```
gen = torch.randn((100, 2))
for itx in range(T-1, 0, -1):
    eps = nnet(gen, torch.ones((gen.shape[0],1)) * itx/T) # Predict noise for step itx
    mut = (gen - eps*(betas[itx] / torch.sqrt(1-alpha_bars[itx])))
    gen =  (1/torch.sqrt(1-betas[itx])) *mut + (torch.sqrt(betas[itx]) * torch.randn_like(gen)) *(T > 0)
```

Generative Model

Sampling (Langevin):

$$\mathrm{d}x(t) = [f(t)x(t) - g(t)^2 \nabla_{x(t)} \log p(x(t))]\mathrm{d}t + g(t)\mathrm{d}w(t) \tag{1}$$

Running backward (x_0: Data, x_T: Noise).

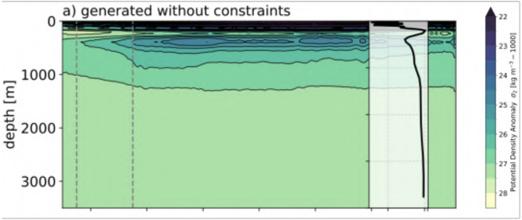
Training (Denoising):
Where $\epsilon_\theta(x(t), t) \approx \nabla_{x(t)} \log p(x(t))$ trained with denoising loss [HJA20]:

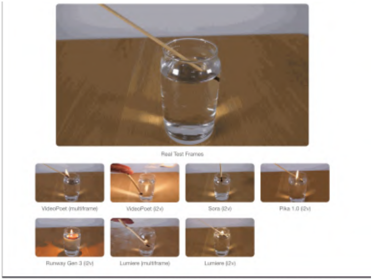
$$\mathcal{L}(\theta) = \mathbb{E}_{s,x_0,\epsilon}[||\epsilon_\theta(x_s, s) - \epsilon||^2] \tag{2}$$

Terms:

- $\epsilon_\theta(x_s, s)$: Trained denoiser
- x_0: Training sample
- $x_s = \sqrt{\bar{\alpha}_s}x_0 + \sqrt{1 - \bar{\alpha}_s}\epsilon$
- $\epsilon \sim \mathcal{N}(0, \mathbb{I})$

Initial (unconstrained) results





Motamed, Saman, et al. "Do generative video models learn physical principles from watching videos?."

In practice, unconstrained generation doesn't respect basic physical principles. In our work they don't respect stratification for example. In [Mot+25] state of the art models doesn't respect physical laws.

Integrate physical constraints (Stratification)

Guided sampling: [CKK24]

$$\min_{q \in \mathcal{P}_2(\mathbb{R}^d)} D(q\|p) \quad \text{s.t. } \mathbb{E}_{x\sim q}[C(x)] = 0 \rightarrow \mathcal{L} = D(q\|p) + \lambda \mathbb{E}_{x\sim q}[C(x)] \tag{3}$$

With p learned distribution and q constrained distribution.

Sampling Algorithm

$$\text{Primal-update} : x_{s+1} = x_s - \tau_s \nabla_x \mathcal{L}(x_s, \lambda_s) + \sqrt{2\tau_s}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0,1)$$

$$\text{Dual-update} : \lambda_{s+1} = \lambda_s + \eta \nabla_\lambda \mathcal{L}(x_s, \lambda_s)$$

Stratification constraint:

$$C_1(x) = \sum_k \left(\rho_k - \frac{1}{N}\sum_{i,j} \rho(x_{ijk})\right)^2 \qquad C_2(x) = \sum_k \left(\nabla\rho_k - \frac{1}{N}\sum_{i,j} \nabla\rho(x_{ijk})\right)^2$$

Mean density constraint

Density gradient constraint

Results - Stratification prior integration

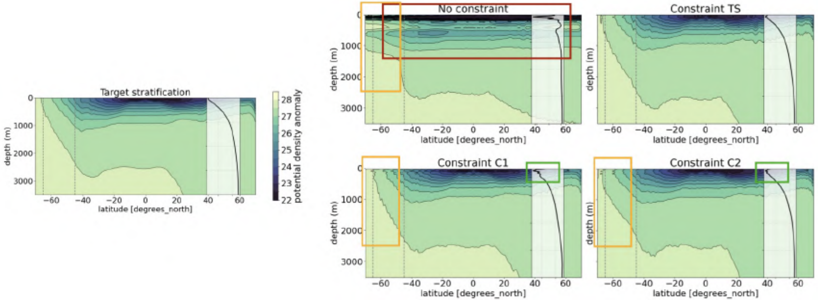


Figure: Stratification: zonal average representation of the density vs depth

Results - Stratification after integration

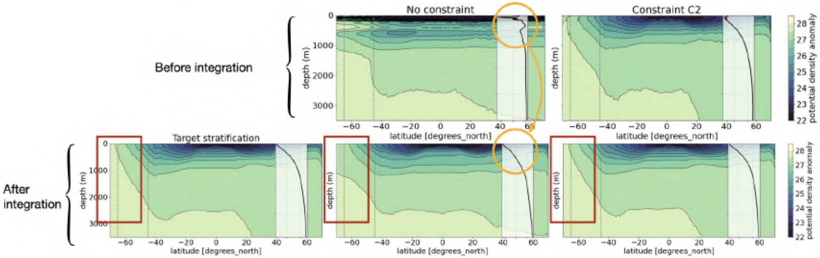


Figure: Stratification after 10 years integration in NEMO

Results - Visual inspection of states

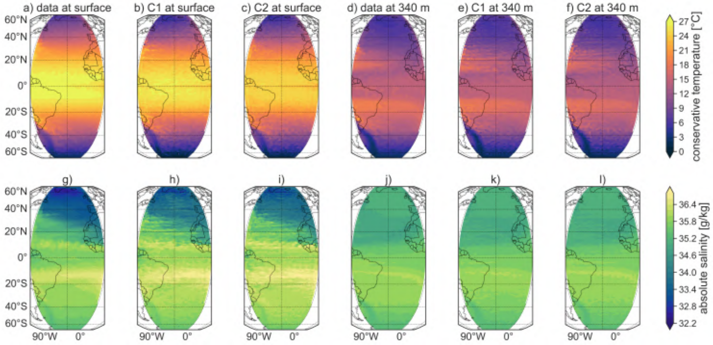
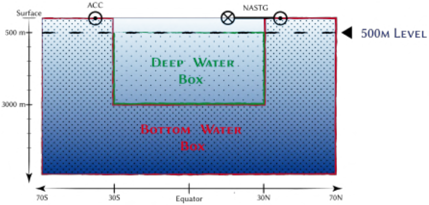


Figure: $\mathcal{T}$ and $\mathcal{S}$ fields training data and generated samples

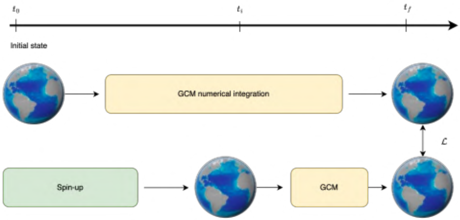
Quantitative results

Source	Bottom-Water $\mathcal{S}$	Bottom-Water $\mathcal{T}$	Deep-Water $\mathcal{S}$	Deep-Water $\mathcal{T}$	Density Errors
Data	35.2 ± 5.0e-6	4.8 ± 2.2e-3	35.3 ± 1.7e-4	2.6 ± 6.3e-3	0.3 ± 4.1e-2
No constraint	35.2 ± 6.4e-2	4.7 ± 0.5	35.3 ± 8.0e-2	2.9 ± 1.0	26.8 ± 9.6
Constraint TS	35.2 ± 1.5e-3	4.7 ± 9.9e-3	35.3 ± 1.6e-3	2.7 ± 1.2e-2	2.0 ± 0.2
Constraint C1	35.2 ± 8.0e-3	4.5 ± 8.0e-2	35.3 ± 1.7e-2	2.4 ± 0.2	5.8 ± 2.5
Constraint C2	35.2 ± 1.2e-2	4.5 ± 0.1	35.3 ± 1.5e-2	2.6 ± 0.2	4.9 ± 1.7

Table: Statistical analysis of water masses and density errors. mean ± std.



Temperature at 500m & in Deep & Bottom Water Boxes



Results - Impact physical constraint

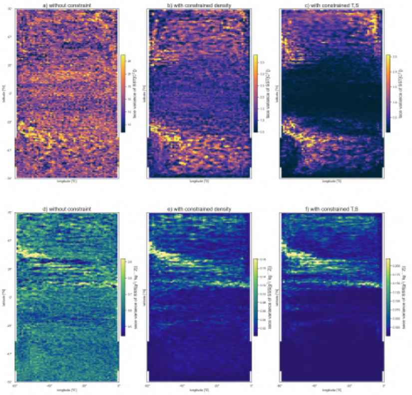


Figure: Spatial variability of temperature (top) and salinity (bottom) fields under three conditions: without constraint (left), density-constrained (center), and temperature-salinity constrained (right).

Current works - conditional sampling

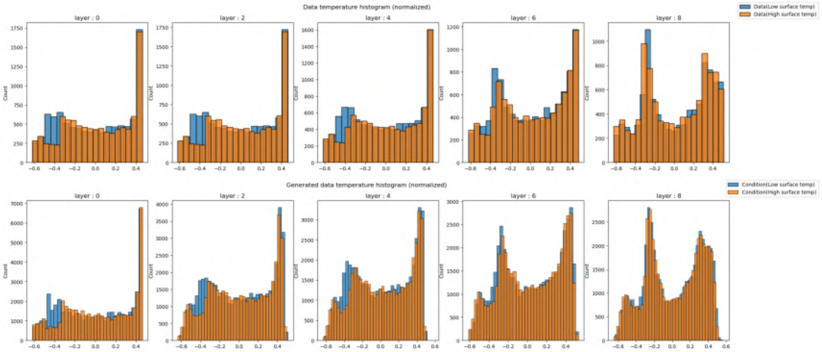


Figure: Condition sampling with surface temperature (high: orange, low: blue). We can note that the generated data follow the same tendency with lower layers. Data is normalised by layer here.

Afterword

Conclusion [1] :

- Generative models can generate non-physically valid states
- Physical evaluation is important and shouldn't be underestimated [2]
- We can impose some constraints on generation

Future work:

- Complex physical constraints formulated on the clean states ($C(x_0)$)
- Conditional models based on physical parameters
- Conditional sampling with variance-preserving schemes

[1] Meunier, Kamm, Gachon, Lguensat & Deshayes. Learning to generate physical ocean states: Towards hybrid climate modeling. ICLRw, 2025

[2] That's why we are building a tool for that github.com/m2lines/Spinup-NEMO