

Ensemble Smoothing for Joint State and Parameter Estimation in Turbulent Urban Flows

Challenges of using LES simulators for data assimilation in urban areas

Nikolaj T. Mücke, Femke Vossepoel

TU Delft · UrbanAIR project

EnKF Workshop 2026 | Rosendal, Norway | 15 -- 18 June



Funded by
the European Union

Overview

- LES forward models: uDALES, PALM, LBM
- The pyurbanair framework
- Parameter estimation with ESM DA
- State estimation
- Neural surrogate model

Digital twins for cities

UrbanAIR builds digital twins for urban air quality and heat (part of *Destination Earth*).

- Street-level forecasts of heat & pollution hotspots
- Decision support: zoning, greening, extreme-heat planning
- Uncertainty quantification, data assimilation

Horizon Europe

€14.3 M, 2025–2028

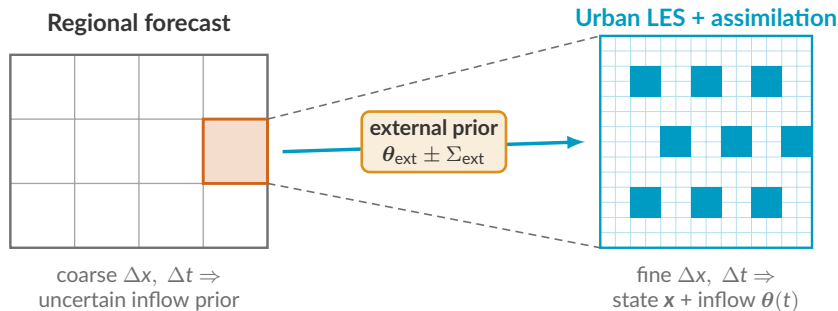
Consortium

18 partners, 11 countries

Cities

Antwerp, Barcelona;
Paris, Bristol, Rotterdam

Connecting regional model with local model

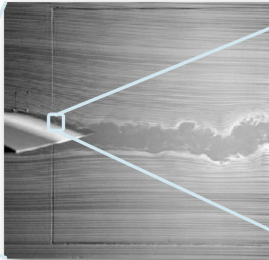


Correct the **time-varying inflow** (wind speed & direction) of an urban LES from **sparse in-domain sensors**.

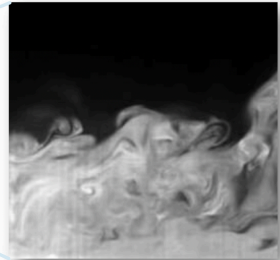
Urban turbulence is multiscale



~10 km



~1 m



~1 mm

Why this is hard

Strong nonlinearity

chaotic dynamics; inflow
changes cascade through
wakes

Non-Gaussianity

flow-building interactions
break Gaussian assumptions

High dimension

$\dim(\mathbf{x}) \sim 10^5 - 10^7$,
 $N_e \sim \mathcal{O}(10^2)$

Part 1

LES forward models

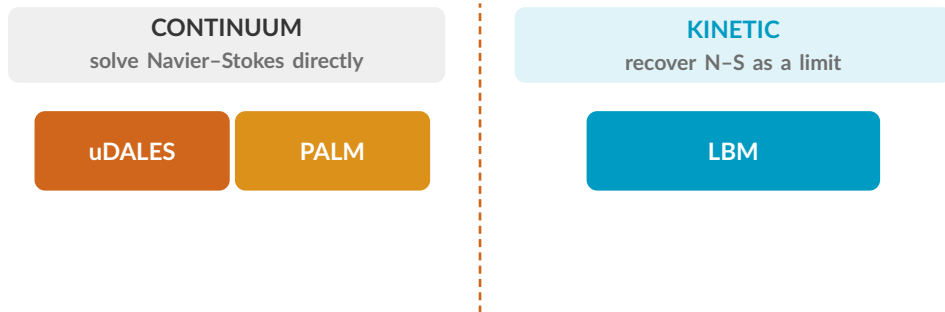
Large-Eddy Simulation in a nutshell

Resolve the large eddies, model the sub-grid scales. Filtered incompressible Navier–Stokes:

$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + (\bar{\mathbf{u}} \cdot \nabla) \bar{\mathbf{u}} = -\frac{1}{\rho} \nabla \bar{p} + \nu \nabla^2 \bar{\mathbf{u}} - \nabla \cdot \boldsymbol{\tau}^{\text{sgs}}, \quad \nabla \cdot \bar{\mathbf{u}} = 0.$$

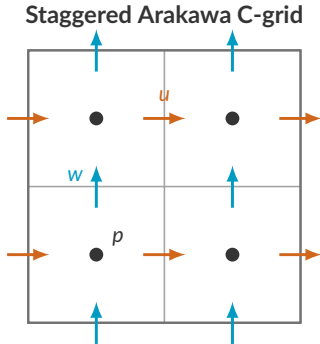
- **sub-grid stress** $\boldsymbol{\tau}^{\text{sgs}}$
- buildings as no-slip walls
- chaotic

Three LES solvers

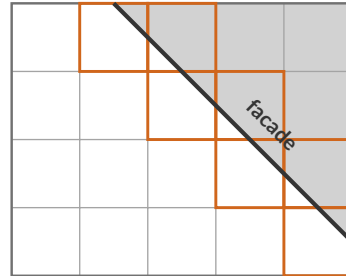


Different numerics *and* physics.

uDALES: staggered C-grid, cut-cell walls

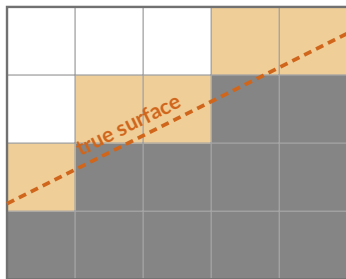


Conservative cut cells

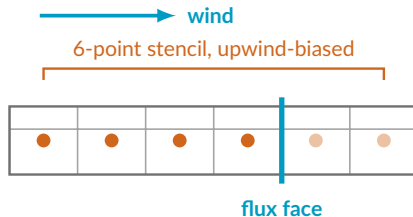


PALM: staircase walls, upwind advection

Mask method: staircase walls

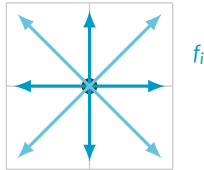


5th-order upwind advection



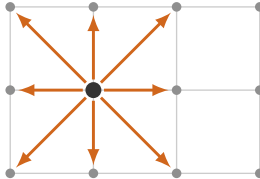
LBM: collide & stream on a uniform lattice

Populations f_i



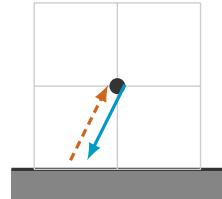
discrete velocity set
(D2Q9 sketch)

Collide & stream



local relaxation, then hop
to neighbours – no global solve

Bounce-back walls



populations reflect
at solid nodes

Part 2

The `pyurbanair` framework

Three Fortran solvers

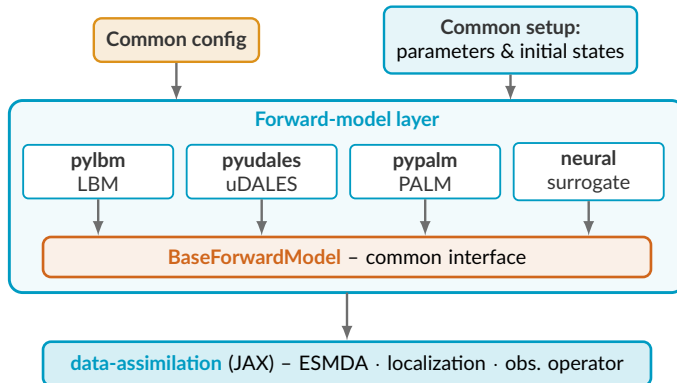
uDALES, PALM and the LBM code are all **Fortran**

- **Compilers & libraries** – each needs its own build (MPI, netCDF...)
- **Config & geometry** – a different format for each
- **Output** – different data types & layouts
- **Warm start** – handled differently by each

pyurbanair: one Python layer over all three

- Automatic download
- Environment and library handling
- Unified config & geometry
- Unified input/output
- Data assimilation

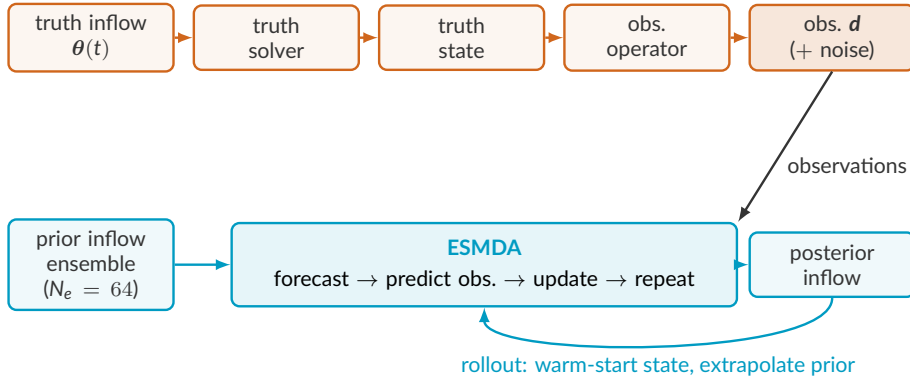
Architecture: one abstraction, many backends



Part 3

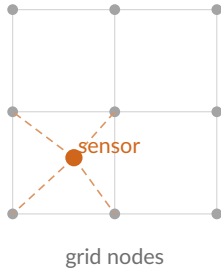
Data assimilation: parameter estimation

Setup

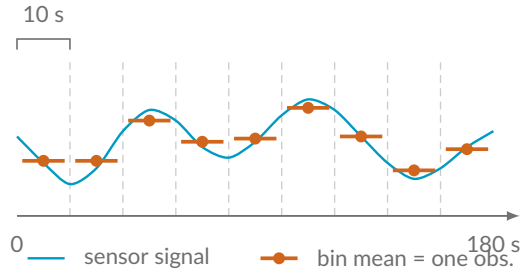


The observation operator

Spatial: N_{obs} sensors, **interpolated** from the grid



Temporal: mean over **intervals**



Ensemble Smoother with Multiple Data Assimilation (ESMDA)

$$\underbrace{\mathbf{m}_j^a}_{\text{posterior}} = \underbrace{\mathbf{m}_j^f}_{\text{prior}} + \overbrace{\mathbf{C}_{MD}(\mathbf{C}_{DD} + \alpha \mathbf{C}_D)^{-1}}^{\text{Kalman gain (from the ensemble)}} \left(\underbrace{\mathbf{d} + \sqrt{\alpha} \mathbf{C}_D^{1/2} \mathbf{z}_j}_{\text{perturbed obs.}} - \underbrace{g(\mathbf{m}_j^f)}_{\text{predicted obs.}} \right)$$

$$\sum_i \frac{1}{\alpha_i} = 1 - \text{the inflation schedule over } N_a \text{ steps.}$$

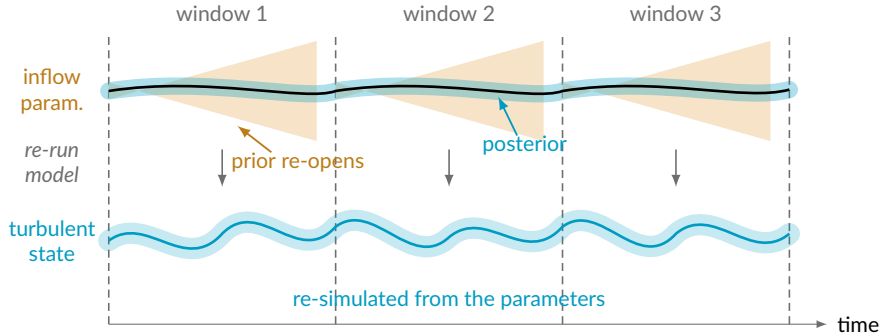
What ESMDA actually updates

The inflow is a **trajectory**, carried on a few time knots per window:

$$\mathbf{m} = \left[\underbrace{\alpha_0, \alpha_1, \dots}_{\text{inflow angle}}, \underbrace{|U|_0, |U|_1, \dots}_{\text{wind speed}} \right].$$

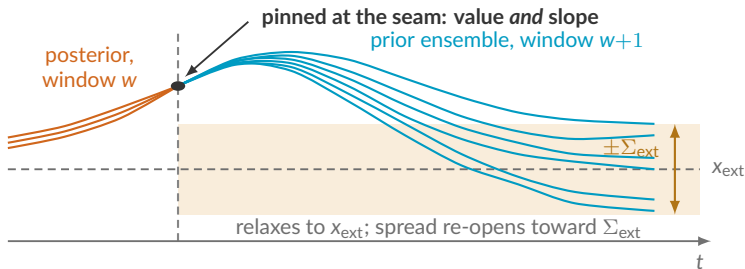
- Smooth, time-correlated between knots
- Assimilate whole window at once

Windowed rollout: re-open uncertainty, keep the flow

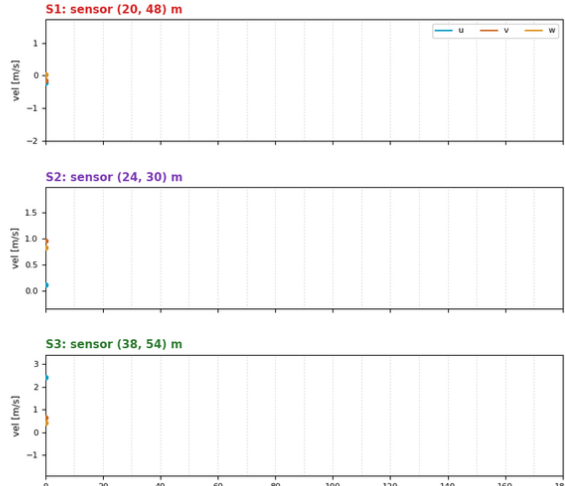
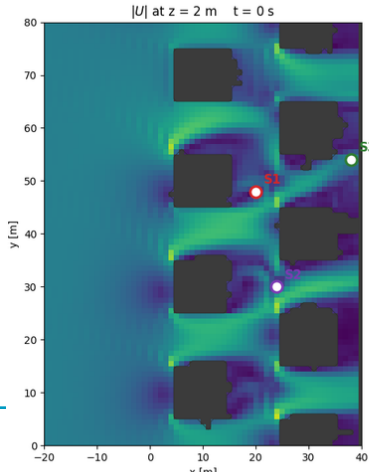


Every update **re-simulates** the state from the new parameters – only the IC carries over.

Dynamic parameters: AR(2) trajectories

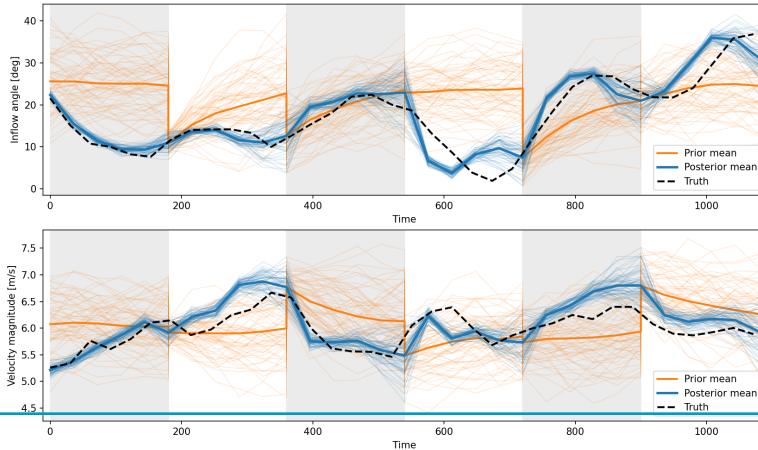


Sensor observations

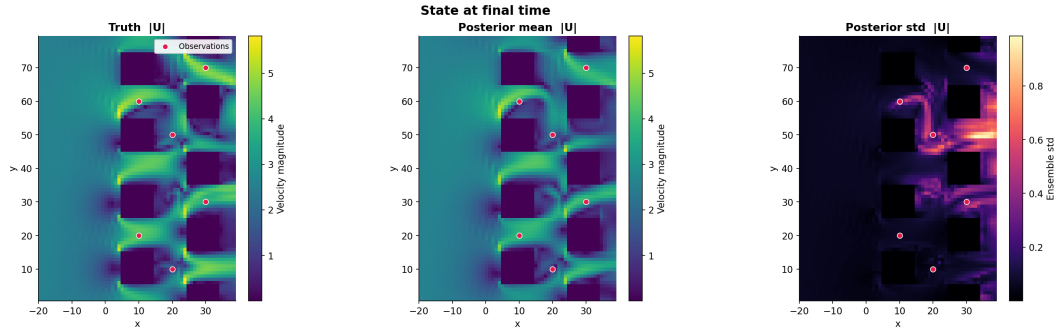


uDALES ($\Delta x = 1$ m): parameter trajectories

Parameter trajectories: truth vs prior vs posterior



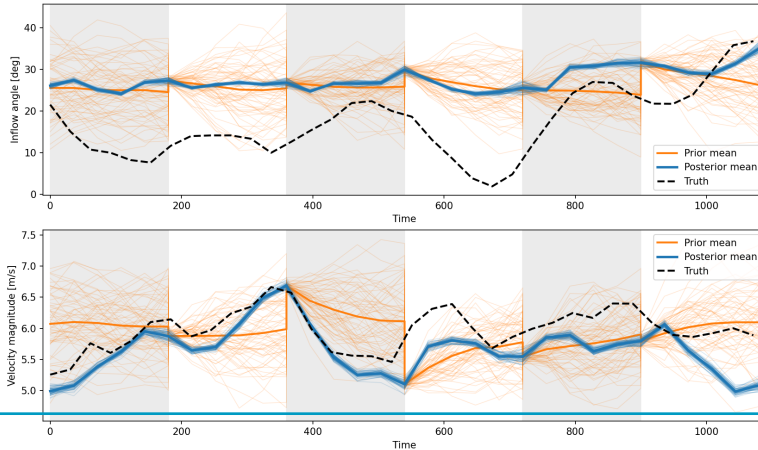
uDALES ($\Delta x = 1$ m): reconstructed final state



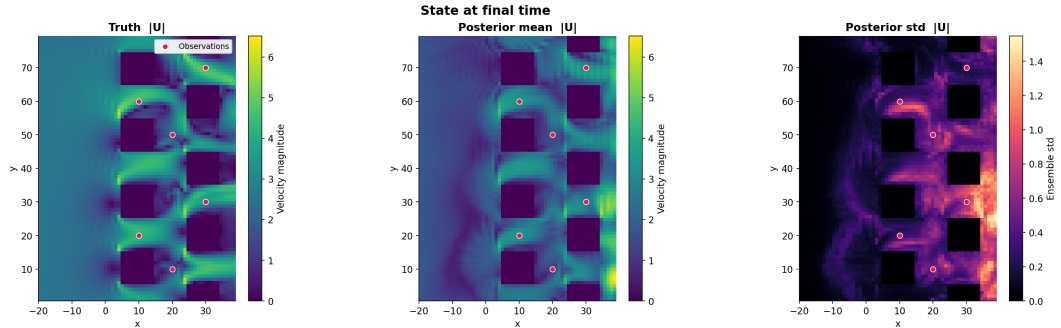
$|U|$ at the final time: **truth**, **posterior mean**, and ensemble std; dots = assimilation sensors.

PALM ($\Delta x = 1$ m): parameter trajectories

Parameter trajectories: truth vs prior vs posterior



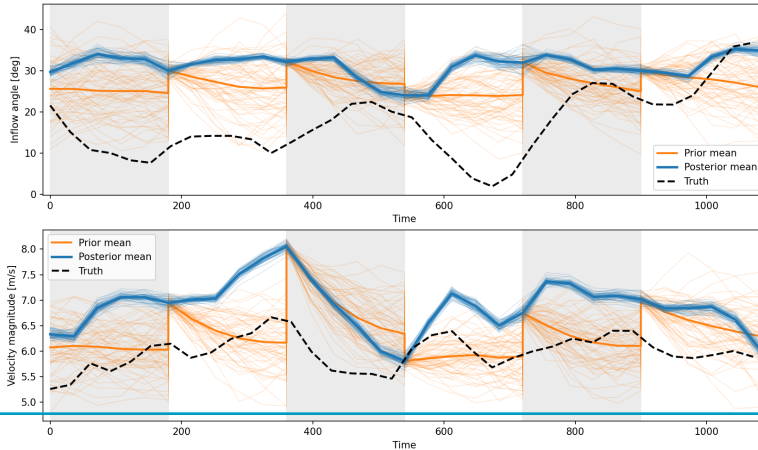
PALM ($\Delta x = 1$ m): reconstructed final state



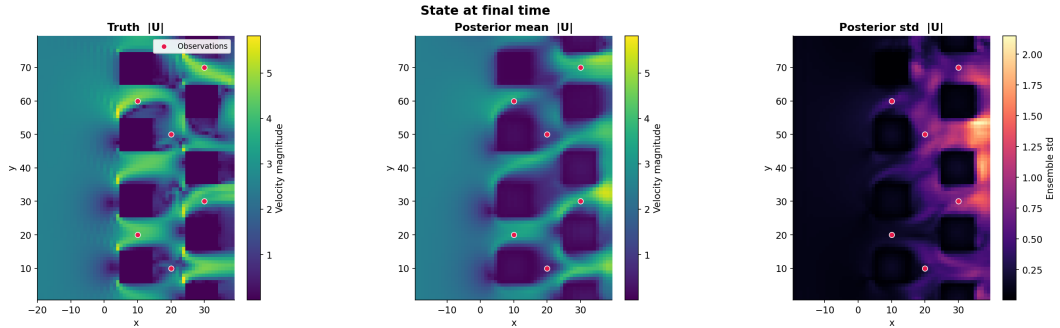
$|U|$ at the final time: **truth**, **posterior mean**, and ensemble std; dots = assimilation sensors.

LBM ($\Delta x = 1$ m): parameter trajectories

Parameter trajectories: truth vs prior vs posterior



LBM ($\Delta x = 1$ m): reconstructed final state



$|U|$ at the final time: **truth**, **posterior mean**, and ensemble std; dots = assimilation sensors.

Accuracy at a glance: parameters & state

Model	Δx [m]	α RMSE [deg]	$ U $ RMSE [m/s]	field RMSE [m/s]	val. RMSE [m/s]
LBM	2	27.0	2.23	2.34	1.54
LBM	1.3	11.2	0.61	0.78	0.35
LBM	1	14.2	0.90	1.00	0.78
PALM	2	11.8	0.68	0.96	0.84
PALM	1.3	15.5	0.62	0.78	0.63
PALM	1	11.1	0.35	0.76	1.00
uDALES	2	18.5	1.18	1.01	0.71
uDALES	1.3	7.49	0.45	0.37	0.31
uDALES	1	3.37	0.30	0.26	0.24

Part 4

Data assimilation: state & parameter estimation

Estimating the state, not just the inflow

ESMDA now acts on the **augmented state** $\mathbf{w} = [\mathbf{x}, \boldsymbol{\theta}]$.

- The state is **high-dimensional** ($10^5 - 10^7$)
- Two routes: **localize** the update, or **reduce** the state dimension
- Focus on initial condition in each window

Localizing the initial-condition update

Distance-based

- keep obs within a physical distance d_t
- must be **predefined**

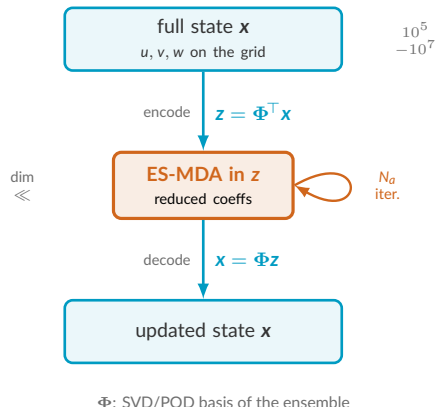
Correlation-based (adaptive)

- keep obs with ensemble correlation $|\rho| > \rho_t$
- **no coordinates**

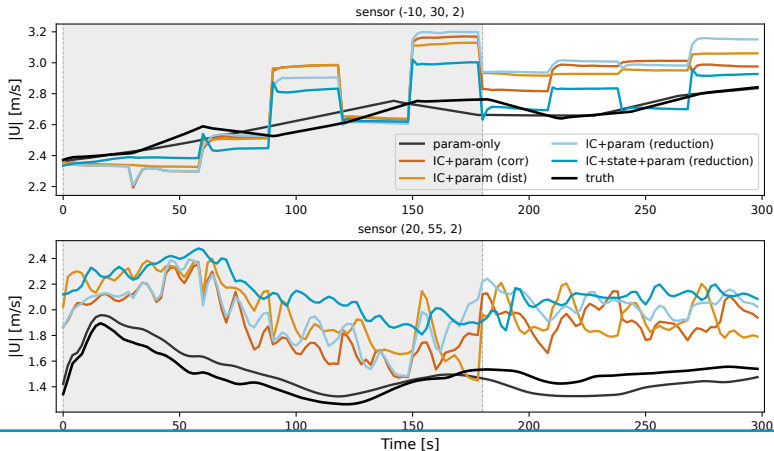
State reduction with an SVD subspace

Alternative to localization. Two variants:

- **reduced IC** – estimate the window's initial condition
- **reduced IC + state** – plus a **final full-state Kalman update**



Validation sensors by method (uDALES)



Method comparison (uDALES, 0--300 s)

Method	α RMSE [deg]	$ U $ RMSE [m/s]	field RMSE [m/s]	val. RMSE [m/s]	N_e
param-only	1.02	0.25	0.11	0.08	64
IC+param (corr)	6.34	0.75	0.52	0.35	128
IC+param (dist)	6.35	0.75	0.51	0.41	128
IC+param (reduction)	6.97	0.79	0.54	0.41	128
IC+state+param (reduction)	5.00	0.43	0.34	0.47	128

Part 5

Neural surrogates

Why a surrogate?

- LES forward models are very expensive to run!
- We need to run them many times
- Train a surrogate **offline**, then use it **online**

ESMDA inner loop

forecast N_e members



predict observations



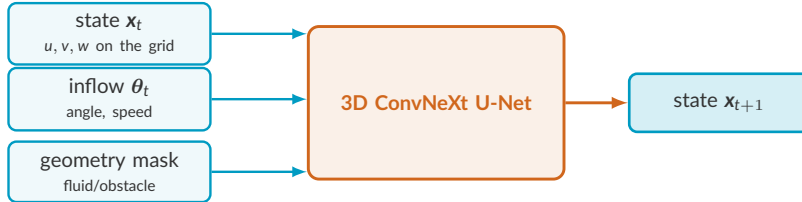
update parameters



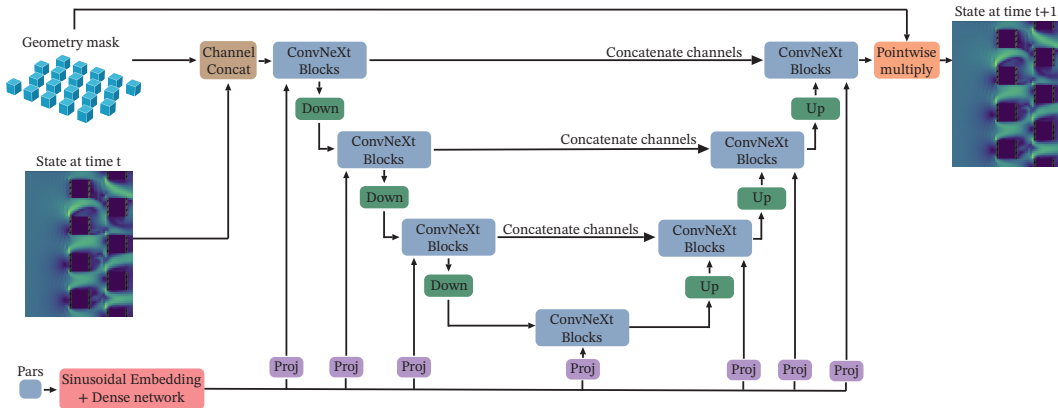
repeat

Surrogate replaces the forecast

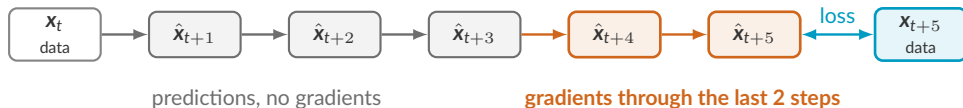
General setup: learn the one-step transition



Architecture: ConvNeXt U-Net



Training: pushforward unrolling + horizon curriculum



- **Rollout** predictions
- Only compute gradient of last 2 steps
- Ramp up rollout steps during training

Computational efficiency

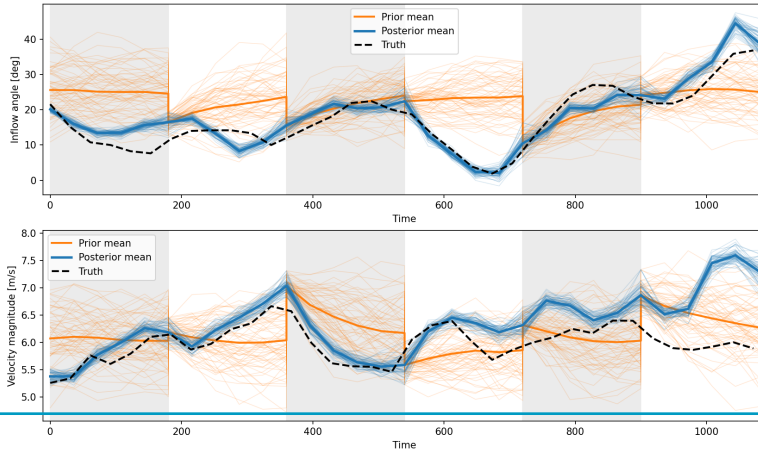
- Separable depthwise convs
- Compile model
- bf16 autocast
- Async dataloading
- Geometry de-duplication

Inside pyurbanair: a drop-in forward model

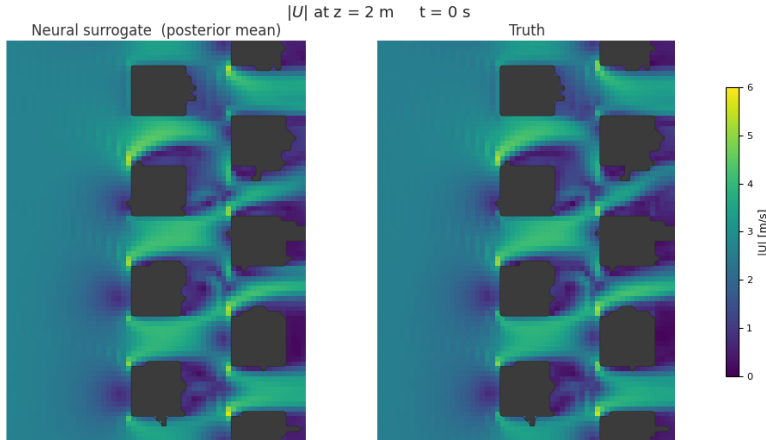
- Trained neural network is wrapped in the same forward model setup
- Cold start: **uDALES spins up** the flow
- Vectorized over **ESMDA ensemble members**

Neural surrogate: parameter trajectories

Parameter trajectories: truth vs prior vs posterior



Surrogate vs. truth: reconstructed state (0--600 s)



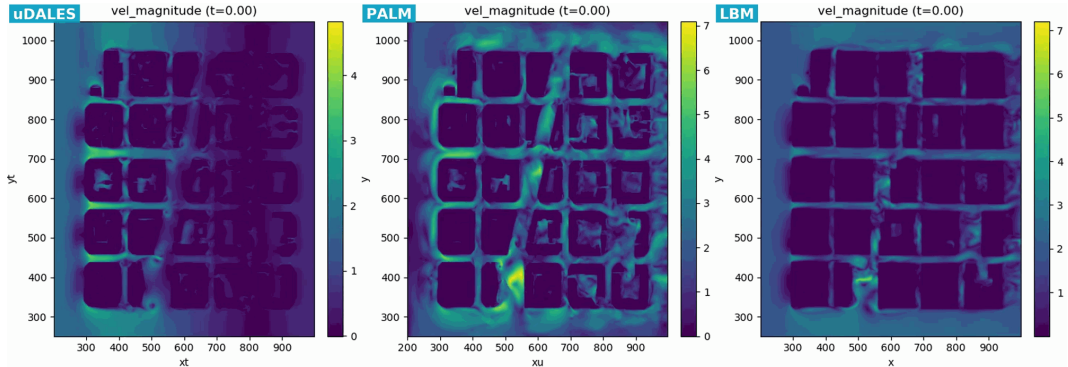
Part 7

Next steps

Where we go from here

- **Unify model settings further**
- **More tests on hyperparameters** – ensemble size, localization, window length, observation intervals
- **State-estimation methods** – improve state localization and reduction
- **Improve the neural surrogate** – accuracy and stability over longer rollouts
- **Larger cases** – scale up to bigger domains and real urban geometries

Geometry from Barcelona: uDALES, PALM, LBM



Estimated velocity magnitude $|U|$ over the building array — the three solvers side by side. (animation plays in Adobe Acrobat)

Conclusion

- Not easy to simply use a different model!
- State estimation is difficult!
- Neural surrogates show potential

Thank you!

Questions & discussion welcome.

Nikolaj T. Mücke · ntmucke@tudelft.nl · github.com/nmucke/pyurbanair