JEDI-LETKF Experiments

DARC/NCEO Data Assimilation Training Course 2025

This practical uses the Object-Oriented Prediction System (OOPS) repository within the Joint Effort for Data assimilation Integration (JEDI) framework. OOPS implements most data assimilation algorithms and includes two toy models for experimentation.

We use a two-level quasi-geostrophic (QG) model to explore the performance of the Local Ensemble Transform Kalman Filter (LETKF) under various configurations. We will examine how factors such as ensemble size, localization radius and inflation factor, as well as observation distributions and types affect the resulting analysis.

This practical will help to develop a better understanding of

- How background error statistics depend on ensemble size
- How observation networks affect analysis accuracy
- How to tune the LETKF system to reduce analysis error

Step-by-step instructions are provided below.

Step 1: Set up the experiment by running the following commands in terminal

```
cp /storage/research/nceo/DA-training-course/docs/setup_letkf_exps.sh .

bash setup_letkf_exps.sh

cd letkf_exps
```

Note that we need to first log in to RACC2 using the command: ssh -X USERNAME@racc.rdg.ac.uk

Step 2: Run the LETKF

```
bash run_letkf.sh
```

This produces three NetCDF files in the “output” folder. The runtime with an ensemble size of 100 and all observation types assimilated is about 50 seconds. For much faster execution (2-3 seconds) run

```
sbatch run_letkf.sh
```

In this case, the screen output is saved to “myout.txt”.

Step 3: Load the Anaconda environment for Python

```
module load anaconda/2023.09-0/met-env
```

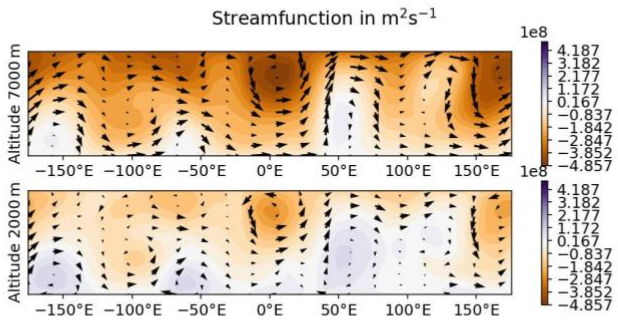
Step 4: Plot the results:

```
bash draw_letkf.sh
```

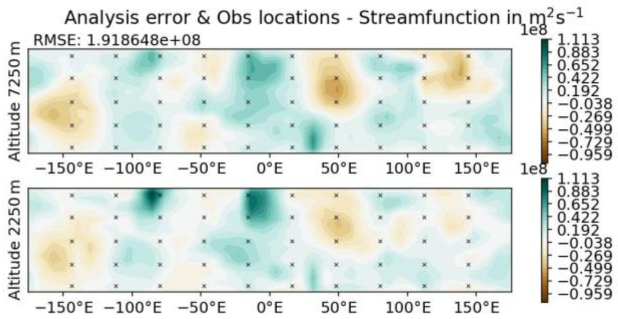
This generates eight JPG files:

- Four plots of the true fields: streamfunction (x), potential vorticity (q), eastward wind (u) and northward wind (v) of the two-level QG model.
- Four plots of the corresponding analysis error fields.

For example, the figure below shows the true streamfunction field at the upper and lower levels, with the arrows indicating the wind vectors derived from the streamfunction.



In addition, the figure below shows the difference between the analysis and the truth streamfunction fields. In this experiment, 50 observations of each observation type (streamfunction, eastward and northward wind components, and wind speed) were assimilated at the location marked by '×'. The Root Mean Square Error (RMSE) of the entire analysis field is indicated at the top-left of the upper panel.



The figures can be viewed with

```
display <figure name>
```

Alternatively, the reader is welcome to use an IDE (e.g., Visual Studio Code) for viewing if they are familiar with it.

Questions to Discuss:

1. *How do background error statistics vary with different ensemble sizes?*

2. *How does the spatial distribution of observations influence the analysis field?*
3. *How is the analysis variable (streamfunction) related to different observation types via the observation operator? Which type contributes most effectively to improving the analysis?*
4. *How does changing the localization radius affect the analysis error? Why is localization a critical component of ensemble Kalman filters?*
5. *How does the analysis respond to changes in the inflation factor?*

At the end of the practical, you are welcome to share your configuration that produced the best analysis.

Hints:

a) Explore background error covariances

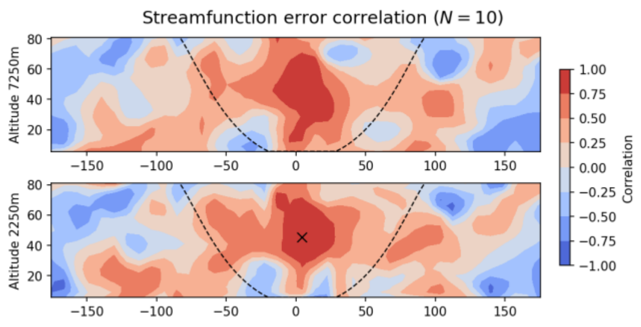
To visualize the background error correlation between streamfunction at a selected grid point and streamfunction at all grid points in physical space, run

```
python plot_ensemble_covariance.py x <ensemble_size> <radius_m>
```

Replace <ensemble_size> with an integer between 2 and 100.

Replace <radius_m> with the localization radius in metres, e.g., 5e6.

This generates “correlation_fields.png” (for an example see below). The selected point is marked with ‘×’. The dashed line outlines the localisation domain, helping the reader visualise the area within which observations influence the selected point.



b) Use different observation files

Three observation files with different observation locations are pre-generated:

- Regularly distributed observations over the entire globe
- Regularly distributed observations on the southern half of the domain
- Regularly distributed observations on the northern half of the domain

To change the observation file

1. Open “letkf.yaml”
2. Find the line: “obsfile: &obs_file Data/truth.obs4d_12h_global.nc”

3. Replace “truth.obs4d_12h_global.nc” with either “truth.obs4d_12h_north.nc” or “truth.obs4d_12h_south.nc”

To ensure the figures reflect the new observation locations, also update the plotting script:

1. Open “draw_letkf.sh”
2. Find the line: “OBS_FILE="truth.obs4d_12h_global.nc"
3. Update the filename to match the change made in “letkf.yaml”

Re-run Steps 2 and 4 to generate updated results and plots.

c) Select observation types

There are three observation types with the same locations:

- Streamfunction
- Eastward and northward winds
- Wind speed

To exclude one type from the assimilation, comment out its corresponding block in “letkf.yaml”. The beginning of an observation type block is marked by the line with “- obs operator:”. This means comment out this line and all lines that are more indented than it.

Tip: Use # at the beginning of each line to comment it out.

Re-run Steps 2 and 4.

d) Tune LETKF parameters

The reader can experiment with different LETKF configurations by editing the following parameters in “letkf.yaml”.

Paramter	**What to edit**	**Notes**
Ensemble size	Line with “nmembers:”	Range: 2-100
Localization radius	Line with “lengthscale:”	Try e.g., 5e6 → 3e6 (in metres)
Inflation factor	Line with “mult:”	Multiplicative inflation factor

Do not forget to re-run Steps 2 and 4 after modifying these parameters to apply the changes and update the outputs.