# SCALE-LETKF User's Guide

Data Assimilation Research Team, R-CCS

September 2025

# Contents

# Chapter 1

# Introduction

## 1.1 About SCALE-LETKF

SCALE-LETKF is a data assimilation and numerical weather prediction system utilizing the regional model SCALE-RM and the data assimilation method Local Ensemble Transform Kalman Filter (LETKF; Hunt et al. (2007)).

## 1.2 Version and history

The current version of SCALE-LETKF is **5.5.5-v1**.

SCALE-LETKF has been developed since 2014, when the prototype was made by Guo-Yuan Lien. The first research papers which used SCALE-LETKF were published in 2016 (Miyoshi et al., 2016a,b).

The development of SCALE-LETKF continues in the Data Assimilation Research Team of RIKEN Center for Computational Science (R-CCS).

## 1.3 License

The SCALE-LETKF code is licensed by MIT license.

# Chapter 2

# Getting started

## 2.1 Code structure

SCALE-LETKF is designed to be used with the library Scalable Computing for Advanced Library and Environment (SCALE) developed in RIKEN. The main code of SCALE-LETKF is almost entirely written in Fortran, as well as SCALE. The main compiled binaries include `letkf`, which is the main code for data assimilation with LETKF, and a group of ensemble wrappers of SCALE regional model (SCALE-RM), which are `scale-rm_ens`, `scale-rm_init_ens`, and `scale-rm_pp_ens`. The code also includes other tools such as ensemble forecast sensitivity to observation (EFSO), and observation data decoders.

## 2.2 Sources

### 2.2.1 SCALE

Obtain the source code from SCALE web page or github repository. The model description and user guide can be found in the documentation page.

The current SCALE-LETKF code only supports the latest public version of SCALE (version 5.5.5 as of September 2025).

### 2.2.2 SCALE-LETKF

The public version of SCALE-LETKF can be downloaded from github.

### 2.2.3 Database

The topography and landuse dataset for SCALE-RM can be obtained from the SCALE web page. The SCALE-LETKF observation and boundary condition data for the tutorial can be found in the following path.

```
Fugaku:/share/ra000007/u10335/scale_database
```

If you are a Fugaku user, copy the database to your own directory under `/vol0003` or `/vol0004`, as `/share` is not accessible directly from the compute

node. Set the environmental variable `SCALE_DB` to the `scale_database` path copied to your directory.

## 2.3 Compilation

### 2.3.1 Environmental variables

Before the compilation of SCALE and SCALE-LETKF, set environmental variables as follows.

```
export GROUP=<your group number> ### not Fugaku but hpXXXXXX or raXXXXXX
export SCALE_SYS="FUGAKU"
export SCALE_DB=<path to your directory>/scale_database ### your own directory
export SCALE_ENABLE_OPENMP=T
export SCALE_ENABLE_PNETCDF=F
```

The following environmental variables are not necessary but optional.

```
export SCALE_USE_SINGLEFP=T
export SCALE_QUICKDEBUG=T
export SCALE_DEBUG=T
```

On Fugaku, some libraries such as NetCDF need to be loaded and linked from spack. The sample can be found in the database directory.

```
source <your $SCALE_DB>/setup-scale-compile.sh
```

### 2.3.2 Compile SCALE-RM

```
git clone https://github.com/scale-met/scale.git
cd scale/scale-rm/src
make -j
```

For more information, please refer to the SCALE user's guide.

### 2.3.3 Compile SCALE-LETKF

```
git clone https://github.com/SCALE-LETKF-RIKEN/scale-letkf.git
cd scale-letkf/scale
make -j
```

The compile options are in `arch/configure.${SCALE_SYS}`.
After the compilation, check if all the binary files are created.

```
ensmodel/scale-rm_pp_ens
ensmodel/scale-rm_init_ens
ensmodel/scale-rm_ens
letkf/letkf
letkf/efso
```

## 2.4 Run a test script

The directory `scale-letkf/scale/run_light` provides a set of simple scripts to test the SCALE-LETKF code on Fugaku.

First, enter the directory and specify the path of the database in `prep.sh`.

prep.sh

```
LETKFDIR=${mydir}/..
SCALEDIR=${mydir}/../../..
### path to your directory
DATADIR="/data/$(id -ng)/$(id -nu)/scale_database/scale-letkf-test-suite"
```

Execute prep.sh and you will have symbolic links to the binary files `scale-rm_init_ens`, `scale-rm_ens`, and `letkf` in the directory. You can also find the input data files in the directories `0001-0005` and `mean`.

First, let's just run the script.

```
### Fugaku interactive job
pjsub --interact --sparam "wait-time=300" exec_pjsub.sh
```

In the batch job script `exec_pjsub.sh`, three binaries are executed.

```
echo "scale-rm_init_ens"
mpiexec -std-proc log/scale_init/NOUT -n 48 \
./scale-rm_init_ens config/scale-rm_init_ens_20220101000000.conf
echo "scale-rm_ens"
mpiexec -std-proc log/scale/NOUT -n 48 \
./scale-rm_ens config/scale-rm_ens_20220101000000.conf
echo "letkf"
mpiexec -std-proc log/letkf/NOUT -n 48 \
./letkf config/letkf_20220101060000.conf
echo "done."
```

When the job starts running, the log file of the job script `exec_pjsub.sh.<jobid>.out` appears. When the job is completed successfully, the following text appears in the log file.

exec_pjsub.sh.<jobid>.out

```
scale-rm_init_ens
scale-rm_ens
letkf
done.
```

You can also find a pair of analysis and first guess data of each ensemble member and ensemble mean at 2022/01/01 06:00:00.

```
$ ls */*/init*pe000000.nc
0001/anal/init_20220101-000000.000.pe000000.nc
0001/anal/init_20220101-060000.000.pe000000.nc
0001/gues/init_20220101-060000.000.pe000000.nc
0002/anal/init_20220101-000000.000.pe000000.nc
...
mean/anal/init_20220101-000000.000.pe000000.nc
mean/anal/init_20220101-060000.000.pe000000.nc
mean/gues/init_20220101-060000.000.pe000000.nc
```

## 2.5 Run a test script (step by step)

After you confirm that the script works, let's do it again step by step and see the necessary settings and input data at each step. First, run the script `clean.sh` to remove the output files and temporary log files in the directory `run_light`, then execute `prep.sh` again.

In this test case, the target domain is the area around Japan with a horizontal grid spacing of 18 km.

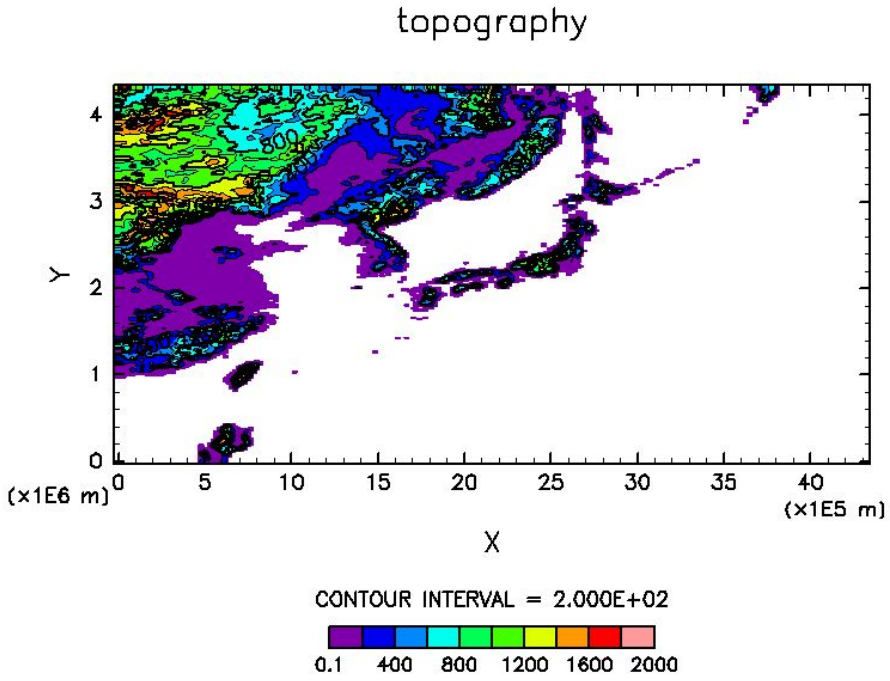


Figure 2.1: Topography

### 2.5.1 scale-rm_init_ens

A binary `scale-rm_init_ens` is a wrapper of `scale-rm_init` with multiple ensemble members. It calls the core program of `scale-rm_init` for each member inside, reading corresponding separate namelist files. For example, the first member uses the file `0001/init.d01_20220101000000.conf`.

The necessary input data includes

- namelist file for the entire program (`config/scale-rm_init_ens_20220101000000.conf`)
- namelist file for each member (`0001/init.d01_20220101000000.conf`)
- topography/landuse data (`const/topo`, `const/landuse`)
- parent model data for boundary condition (`mean/bdyorg*.grd`)
- namelist file for parent model input (`mean/gradsbdy.conf`)

- output directory for log file (`log/scale_init`)

The output data includes

- boundary data (`mean/boundary*.nc`)
- log files

In this testcase, the initial conditions are already prepared and copied from `DATADIR` for each member. The boundary conditions are created by `scale-rm_init_ens`. All the members share the same boundary condition generated from the parent model forecast. Therefore, `scale-rm_init_ens` only creates boundary conditions for the first member `0001`. This is set by the parameter `MEMBER_RUN` in the namelist `config/scale-rm_init_ens_20220101000000.conf`.

scale-rm_init_ens_20220101000000.conf

```
&PARAM_ENSEMBLE
MEMBER = 5,
MEMBER_RUN = 1,
CONF_FILES = "<member>/init.d<domain>_20220101000000.conf",
CONF_FILES_SEQNUM = .false.,
DET_RUN = .false.,
DET_RUN_CYCLED = .true.,
/
```

The initial condition for each member in this tutorial is already prepared. They are made by adding random perturbations to `RHOT` and `DENS` with a selected wavelength range to the background analysis of NCEP GDAS. The example of `RHOT` fields of the first two members are shown below.

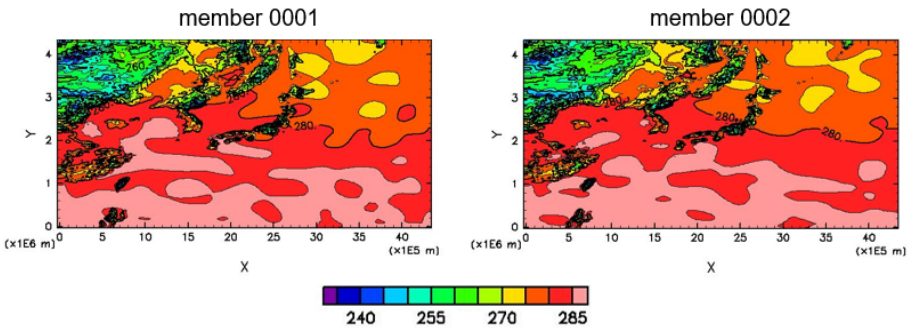


Figure 2.2: RHOT of member 1 and 2

### 2.5.2 scale-rm_ens

Next, a binary `scale-rm_ens` is a wrapper of `scale-rm` with multiple ensemble members. It calls the core program of `scale-rm` for each member inside, reading corresponding separate namelist files. The namelist for the first member is `0001/run.d01_20220101000000.conf`.

The necessary input data includes

- namelist file for the entire program (`config/scale-rm_ens_20220101000000.conf`)

- namelist file for each member (`0001/run.d01_20220101000000.conf`)
- topography/landuse data (`const/topo`, `const/landuse`)
- initial condition files (`<member>/anal/init_20220101-000000.000.*.nc`)
- boundary files (`mean/boundary*.nc`)
- output directory for log file (`log/scale`)

The output data includes

- restart files (`<member>/gues/init_20220101-060000.000.*.nc`)
- history files (`<member>/hist/history*.nc`)
- log files

Both restart and history files are necessary in this testcase, as it performs 4-D LETKF. Note that the ensemble forecast is performed from 2022-01-01 00:00:00 to 2022-01-01 09:00:00 to cover the 6-hour assimilation window from 03:00:00, whereas the restart files are created at 2022-01-01 06:00:00.

Also note that the original restart files `<member>/gues/init_20220101-060000.000.*.nc` are copied to `<member>/anal/init_20220101-060000.000.*.nc`. This is prerequisite for LETKF which does not create but overwrite NetCDF files.

### 2.5.3 letkf

The necessary input data includes

- namelist file for the program (`config/letkf_20220101060000.conf`)
- topography/landuse data (`const/topo`, `const/landuse`)
- restart files to overwrite (`<member>/anal/init_20220101-060000.000.*.nc`)
- history files (`<member>/hist/history*.nc`)
- observation files (`obs/obs_20220101060000.nc`)
- output directory for log file (`log/letkf`)

The output data includes

- restart files (overwrite) (`<member>/gues/init_20220101-060000.000.*.nc`)
- log files

The observation data in this case is the PREPBUFR which is used in the NCEP Global Data Assimilation System (GDAS). The assimilation time window is 6 hours from "2022-01-01 03:00:00" to "2022-01-01 09:00:00". The time series of background ensemble forecast data is loaded from history files and transformed to an observation space. The resultant analysis mean and each member fields are overwritten to `<member>/anal/init_20220101-060000.000.*.nc`.

To quickly check if the LETKF works, see the log file `log/letkf/NOUT.3.0`. The observation departure statistics before and after the LETKF indicates how the misfit between the background (forecast ensemble mean) and the observation is reduced by the data assimilation.

log/letkf/NOUT.3.0

```
OBSERVATIONAL DEPARTURE STATISTICS [GUESS] (GLOBAL):
=====================================================================
                U           V           T           Q           PS
---------------------------------------------------------------------
BIAS     2.317E-02   7.718E-01   1.208E-02   2.321E-05  -1.346E+02
RMSE     3.009E+00   2.762E+00   1.717E+00   4.158E-04   2.145E+02
NUMBER       14652       14654          93          36        1379
=====================================================================

OBSERVATIONAL DEPARTURE STATISTICS [ANALYSIS] (GLOBAL):
=====================================================================
                U           V           T           Q           PS
---------------------------------------------------------------------
BIAS    -2.040E-02   7.287E-01   1.227E-02   2.174E-05  -1.137E+02
RMSE     2.821E+00   2.636E+00   1.634E+00   4.204E-04   1.955E+02
NUMBER       14652       14654          93          36        1379
=====================================================================
```

# Chapter 3

# Run an example DA experiment using the scripts

This chapter explains how to run a data assimilation cycle and extended ensemble forecasts using the scripts in `scale/run`. It uses the example of the real case experiment of a 6-hourly data assimilation of NCEP PREPBUFR conventional observation set in a Japan domain with a 18-km grid. The basic configuration follows Lien et al. (2017) with some modifications.

## 3.1 Computational domain

The computational domain and grid settings are shown in Table 3.1.

| | |
|---|---|
| Target area | Japan |
| Map projection | Lambert conformal conic |
| dx | 18 km |
| Model top | 28.8 km |
| levels | 36 |
| dt | 40 sec |
| Number of processes | 8 |
| Number of subdomain in X | 4 |
| Number of subdomain in Y | 2 |

Table 3.1: Domain and grid settings

The computational domain of this experiment is shown in Figure 3.1.

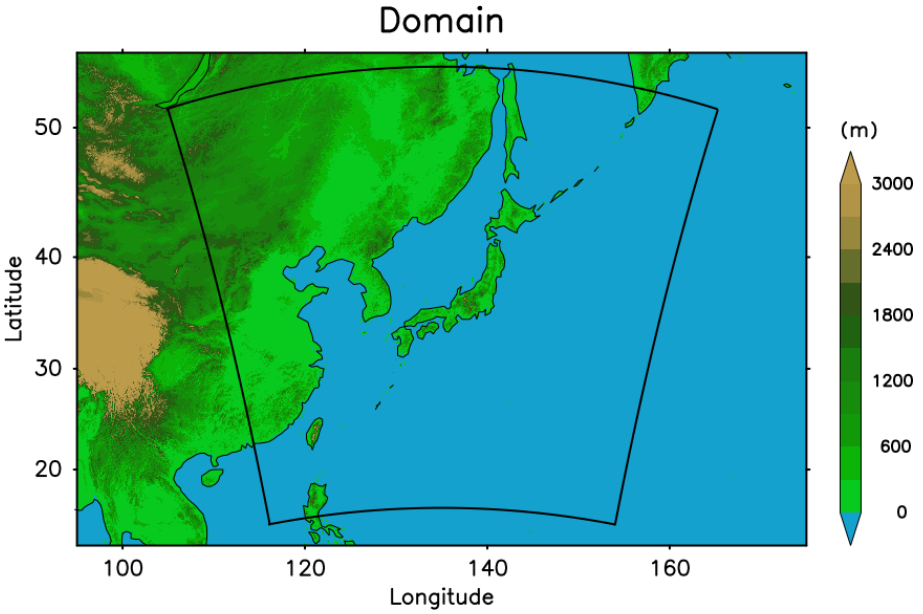


Figure 3.1: Computational domain

## 3.2 Data

### 3.2.1 PREPBUFR

The PREPBUFR observation data set is obtained from NCEP NOMAD. No user registration is required. The original data in wgrib2 format needs to be decoded and converted to LETKF observation file format before the experiment. Super-observation, observation selection or thinning treatment may be performed along with it. In this testcase, converted observation files in `$DATADIR/scale-letkf-test-suite/obs/prepbufr_Japan/` are used. The preparation of PREPBUFR data is explained in detail in Preparation of NCEP PREPBUFR data.

### 3.2.2 Ensemble initial condition

To initiate a data assimilation cycle, it is necessary to create an ensemble of initial model states in a form of restart files. The natural way to create initial conditions to start DA cycle is to add random perturbations to the true state. In this testcase, a set of restart files located in `$DATADIR/scale-letkf-test-suite/exp/18km_Japan/init` is used.

### 3.2.3 Boundary condition

The model needs 3-D atmospheric variable data for lateral boundary conditions. Boundary data files are needed in a special SCALE-RM format that are prepared by the executable `scale-rm_init_ens`. In this testcase experiment, NCEP GFS forecast data is used as the original data to create the boundary conditions. We use a prepared set of boundary files located in

`$DATADIR/scale-letkf-test-suite/exp/18km_Japan/bdy/mean` for all the ensemble members.

There is an option to use different boundary conditions for different members, by setting `BDY_ENS=1` in `config.main`. In that case, boundary data for each member needs to be prepared.

The preparation of NCPE GFS data is explained in detail in Preparation of NCEP GFS data.

## 3.3 Run a data assimilation cycle

Copy files in `run/config/18km_Japan` to `run`.

```
cd scale/run
cp -r config/18km_Japan/* .
ln -s config.main.FUGAKU config.main  ### Fugaku
```

The main settings are in `config.cycle`. Make sure that `MAKEINIT=0` as this experiment uses a pre-made initial ensemble. `STIME` is the initial time to start the first cycle, and `ETIME` is for the last cycle. If they are the same, only one cycle is performed.

config.main

```
STIME='20220101000000'
ETIME='20220101060000'
TIME_LIMIT='00:30:00'

ISTEP=
FSTEP=
CONF_MODE='static'

#-------------------------------------------------------------

FCSTOUT=

ADAPTINFL=0      # Adaptive inflation
                 #  0: OFF
                 #  1: ON

#-------------------------------------------------------------

MAKEINIT=0       # 0: No
                 # 1: Yes
```

The periods of the cycle and observation window are set in the following part of `config.main`. This experiment utilizes "4D-LETKF", in which observations within a time window with a finite period are assimilated. With the settings below, observations in the 6-hour window centered at the assimilation time are used. When the cycle starts at 00Z for example, observations from 03Z to 09Z are assimilated to create analysis at 06Z. For this purpose, the ensemble forecast is performed up to 9 hours from 00Z, as the LETKF needs ensemble first-guess field between 03-09Z.

config.cycle

```
#-------------------------------------------------------------
# Cycling settings

WINDOW_S=10800     # SCALE forecast time when the assimilation window starts (second)
WINDOW_E=32400     # SCALE forecast time when the assimilation window ends (second)
LCYCLE=21600       # Length of a DA cycle (second)
LTIMESLOT=3600     # Timeslot interval for 4D-LETKF (second)
```

The paths to input data are set in `config.main`. This experiment assumes that the initial ensemble restart files and topo/landuse files are already made and placed in `$INDIR`. For boundary files, NCEP GFS data in GrADS format is in `$DATA_BDY_GRADS`. Observation data is in `$OBS`.

config.main

```
DIR="$(cd "$(pwd)/.." && pwd)"    # Root directory of the SCALE-LETKF

DATADIR=${SCALE_DB}          # Directory of the SCALE database

OUTDIR=/data/${GROUP}/$(id -nu)/test_scale/result/18km_Japan ### EDIT HERE ###
INDIR="$DATADIR/scale-letkf-test-suite/exp/18km_Japan/init"

#===============================================================
# Location of model/data files

SCALEDIR="$DIR/../.."        # Directory of the SCALE model

DATA_TOPO="$DATADIR/scale-letkf-test-suite/exp/18km_Japan/init"   # Directory of the
↪  prepared topo files
DATA_TOPO_BDY_SCALE=
DATA_LANDUSE=$DATA_TOPO      # Directory of the prepared landuse files
DATA_BDY_SCALE=              # Directory of the boundary data in SCALE history format
↪  (parent domain)
DATA_BDY_SCALE_PRC_NUM_X=
DATA_BDY_SCALE_PRC_NUM_Y=
DATA_BDY_SCALE_PREP=         # Directory of the prepared SCALE boundary files
DATA_BDY_WRF=                # Directory of the boundary data in WRF format
DATA_BDY_NICAM=              # Directory of the boundary data in NICAM format (not finished)

DATA_BDY_GRADS="$DATADIR/scale-letkf-test-suite/exp/18km_Japan/bdy"

OBS="$DATADIR/scale-letkf-test-suite/obs/prepbufr_Japan"
```

Let's run the experiment by executing `cycle_run.sh`. It is recommendable to run the cycle in the background using the following command.

```
nohup ./cycle_run.sh &> log_cycle_run &
```

While it is running, the progress status can be monitored from the LETKF log files after each cycle created in `$OUTDIR/<atime>/log/letkf/` (note that `<atime>` is `$STIME` plus 6 hours). For a quick check, find the observation departure statistics in the log file.

log/letkf/0/NOUT_20220101060000.3.0

```
OBSERVATIONAL DEPARTURE STATISTICS [GUESS] (IN THIS SUBDOMAIN):
=================================================================
                  U           V           T           Q          PS
-----------------------------------------------------------------
BIAS     7.600E-01   4.386E-01         N/A         N/A  -1.932E+02
RMSE     2.076E+00   2.368E+00         N/A         N/A   2.295E+02
NUMBER        1205        1205           0           0         182
=================================================================
OBSERVATIONAL DEPARTURE STATISTICS [GUESS] (GLOBAL):
=================================================================
                  U           V           T           Q          PS
-----------------------------------------------------------------
BIAS     1.764E-02   7.334E-01   3.690E-03   2.431E-05  -1.326E+02
RMSE     2.863E+00   2.631E+00   1.634E+00   4.179E-04   2.104E+02
NUMBER       14654       14654          93          36        1376
=================================================================
```

log/letkf/0/NOUT_20220101060000.3.0

```
OBSERVATIONAL DEPARTURE STATISTICS [ANALYSIS] (IN THIS SUBDOMAIN):
================================================================
                U           V           T           Q           PS
----------------------------------------------------------------
BIAS     5.967E-01   4.044E-01         N/A         N/A  -1.262E+02
RMSE     1.960E+00   2.160E+00         N/A         N/A   1.723E+02
NUMBER        1205        1205           0           0         182
================================================================
OBSERVATIONAL DEPARTURE STATISTICS [ANALYSIS] (GLOBAL):
================================================================
                U           V           T           Q           PS
----------------------------------------------------------------
BIAS    -1.437E-02   5.357E-01  -1.075E-01   2.920E-06  -8.016E+01
RMSE     2.389E+00   2.387E+00   1.584E+00   3.764E-04   1.708E+02
NUMBER       14654       14654          93          36        1376
================================================================
```

After the job finishes successfully, first guess and analysis restart files at each analysis time step are stored in `$OUTDIR/<time>/gues/<member>` and `$OUTDIR/<time>/anal/<member>`. Output files are separated for each subdomain. You can use `sno` to combine separated files into one. You also have an option to perform regridding with sno plugins.

## 3.4 Run an ensemble forecast from analysis

Use the script `fcst_run.sh` to run an (extended) ensemble forecast from the anlaysis. The main settings are in `config.fcst`. Multiple forecasts can be performed simultaneously from a set of different initial times from `STIME` to `ETIME` every `LCYCLE`. If `STIME` and `ETIME` are the same, only an ensemble forecast from a single initial time is performed. Forecast length and output interval are specified by `FCSTLEN` and `FCSTOUT`. It is possible to perform an extended forecast longer than `LCYCLE`, as long as the original parent domain data for corresponding times are available in a directory specified by `$DATA_BDY_GRADS`. Ensemble members to run forecast are specified by `MEMBERS`, separated by a blank. When `MEMBERS="all"`, all members (`MEMBER` in `config.main`) and "mean" are chosen.

config.fcst

```
#===============================================================
#
#  Settings for fcst.sh
#
#===============================================================

STIME='20220101060000'
ETIME=$STIME
TIME_LIMIT='00:30:00'
MEMBERS='mean 0001 0002 0003 0004 0005'
CYCLE=
CYCLE_SKIP=1
ISTEP=
FSTEP=
CONF_MODE="static"

#===============================================================
# Forecast settings

FCSTLEN=86400      # SCALE forecast length in the forecast mode (second)
FCSTOUT=3600

#===============================================================

MAKEINIT=0       # 0: No
                 # 1: Yes
```

Now the analysis restart files to initialize the forecast are in the same location, as they are just made in the previous data assimilation cycle. Therefore `INDIR=$OUTDIR` in `config.main`.

config.main

```
DIR="$(cd "$(pwd)/.." && pwd)"    # Root directory of the SCALE-LETKF

DATADIR=${SCALE_DB}          # Directory of the SCALE database

OUTDIR=/data/${GROUP}/$(id -nu)/test_scale/result/18km_Japan ### EDIT HERE ###
INDIR=${OUTDIR}
```

After the forecast job is completed, the output files can be found in `$OUTDIR/<time>/fcst`, where `<time>` is the initial forecast time. Note that it is different from `$OUTDIR/<time>/hist` which is used in data assimilation. In the default setting, both history and restart files are created in fcst directory. The output setting is controlled by `OUT_OPT` in `config.fcst`.

# Chapter 4

# Customize your experiment

## 4.1 Structure of the script

The scripts in the directory `scale/run` help you execute data assimilation and ensemble forecast tasks for your experiment in a consistent and efficient manner.

The main shell scripts for executing the computation are `cycle_run.sh` and `fcst_run.sh`. Those scripts are used with the configuration files `config.xxxxx`, which you are supposed to prepare for your own experiment. The example of configuration files for various experiments can be found in `scale/run/config`. Table 4.1 shows the list of configuration files.

| File name | Description | File type | cycle | fcst |
|---|---|---|---|---|
| `config.rc` | Common configuration | shell | yes | yes |
| `config.main` | Main configuration file | shell | yes | yes |
| `config.cycle` | Configurations for cycle jobs | shell | yes | no |
| `config.fcst` | Configurations for fcst jobs | shell | no | yes |
| `config.nml.ensmodel` | Common namelist for `*_ens` | fortran | yes | yes |
| `config.nml.scale_pp` | Namelist for `scale-rm_pp` | fortran | (yes) | (yes) |
| `config.nml.scale_init` | Namelist for `scale-rm_init` | fortran | yes | yes |
| `config.nml.grads_boundary` | Namelist for `scale-rm_init` | fortran | (yes) | (yes) |
| `config.nml.scale` | Namelist for `scale-rm` | fortran | yes | yes |
| `config.nml.scale_user` | Namelist for `scale-rm` | fortran | (yes) | (yes) |
| `config.nml.obsmake` | Namelist for `obsmake` | fortran | (yes) | no |
| `config.nml.letkf` | Namelist for `letkf` | fortran | yes | no |

Table 4.1: Configuration files

The script `fcst_run.sh` does the following tasks.

- Create a runtime directory and prepare necessary executable files, input files, and configuration files.
- Create namelist files for each executable, using parameters specified in the configuration files.
- Create a batch job script for the supercomputer job management system and submit it.

- Call the executable files `scale-rm_pp_ens`, `scale-rm_init_ens`, and `scale-rm_ens` in the batch job script with the appropriate order.

The script for a data assimilation cycle `cycle_run.sh` works similarly, but it also calls `scale-rm_init_ens`, `scale-rm_ens`, and `letkf` repeatedly for the period of the experiment described in the configuration file.

## 4.2 Plan your experiment

Data assimilation experiments can be classified into real and ideal experiments. Real experiments use actual observation data to calculate analysis of the real world, whereas ideal experiments use synthetic observation, which is hypothetical observation data generated from some other simulation data. In both cases, you are supposed to design your experiment to meet your research purpose. Some of the basic features to consider are listed below.

- Available observation variable, temporal and spatial resolution
- Observation error standard deviation
- Model domain and spatial resolution
- Parent data for initial and boundary condition
- Data assimilation interval
- Ensemble size
- Ensemble perturbation initialization
- Spatial localization
- Ensemble spread control, such as covariance inflation and relaxation-to-prior-perturbation (RTPP)

## 4.3 Experiments with real-world observation

### 4.3.1 Observation data

First, prepare your observation data in the SCALE-LETKF observation format. The current public SCALE-LETKF code supports conventional observations included in NCEP PREPBUFR, and radar observations. If you use other observation types such as satellite, you need additional code modification.

If you assimilate PREPBUFR, you can use the fortran decoder `scale/obs/dec_prepbufr.f90` along with `bufrlib` to convert an original BUFR file to a SCALE-LETKF observation file.

Otherwise, you need to manually convert your observation data from the original format to the SCALE-LETKF observation format. You need to put observation element and type indices and specify observation error standard deviation and relative time difference from the target assimilation time for each observation record.

If you assimilate radar observation data, you need to regrid it to a longitude-latitude-z coordinate and specify observation error accordingly. Also, you need to be careful with the choice of radar observation operator (the parameter `METHOD_REF_CALC`), which is supposed to be consistent with your data.

### 4.3.2 SCALE-RM setting

Next, define your model. Setting up the SCALE-RM for your experiment includes the design of the computational domain, the horizontal and vertical grid configuration, domain decomposition for MPI parallelization, the methods and parameters of physics schemes, choice of parent model data for downscaling, and so on. The parent model data can be forecast or reanalysis data of a global or regional model with a larger domain.

If you are not familiar with these settings, it is recommendable to refer to "Part 4 : Various settings" of the SCALE user's guide.

In this document, it is assumed that these SCALE-RM settings and testing are already performed, including the generation of topography and landuse files by `scale-rm_pp` and the namelist parameters for `scale-rm_init` and `scale-rm`.

### 4.3.3 Prepare ensemble initial and boundary condition

In order to launch a data assimilation experiment, you need initial and boundary condition data. The boundary condition could be common for all ensemble members, whereas initial condition data must be an ensemble of members having different states. In a real-world experiment, both initial and boundary data need to be generated by downscaling its parent model data. When the parent model data is an ensemble, the ensemble generation is straightforward. However, when the parent model data is a deterministic forecast, or has less number of members than desired for the SCALE-LETKF experiment, you need to generate ensemble perturbation. Current SCALE-LETKF code offers a python script and its wrapper by shell script to create an ensemble of restart files with structured random perturbation added on a reference data.

Modify the header part of the python script `init_perturb/init_perturb.py` to set up random ensemble perturbation.

init_perturb.py

```
wavel1 =  8000.   ### short-wave cutoff wavelength
wavel2 = 32000.   ### long-wave  cutoff wavelength
dx = 2000.        ### horizontal grid spacing
zheight = 16000.  ### top height for perturbation
taper_width = 10  ### number of grid points to suppress perturbation near the lateral
↪  boundary
taper_mtop = 10   ### number of grid points to suppress perturbation near the top level

pert_std = 0.2    ### STANDARD DEVIATION (in Kelvin) of potential temperature perturbation
halo = 2          ### number of grid points for HALO
```

You can create the perturbed ensemble by running `init_perturb/init_perturb.sh` with initial time and reference data path. The script reads `config.main` and generate ensemble initial flies in `OUTPUT/<atime>/anal/<member>` for all members specified by `MEMBER`.

### 4.3.4 Set up DA cycle with LETKF

The basic settings of a DA cycle experiment are in `config.main` and `config.cycle`.

#### DA step interval and window

The interval and window of the data assimilation is defined in the following part of `config.main`. In this example, a 4-D LETKF experiment with an interval of 6 hours and a window from 3-h to 9-h forecast time is described.

config.main

```
# Cycling settings

WINDOW_S=10800     # SCALE forecast time when the assimilation window starts (second)
WINDOW_E=32400     # SCALE forecast time when the assimilation window ends (second)
LCYCLE=21600       # Length of a DA cycle (second)
LTIMESLOT=3600     # Timeslot interval for 4D-LETKF (second)
```

In the case of a 3-D LETKF experiment, they have all the same value.

config.main

```
# Cycling settings

LCYCLE=3600       # Length of a DA cycle (second)
WINDOW_S=$LCYCLE     # SCALE forecast time when the assimilation window starts (second)
WINDOW_E=$LCYCLE     # SCALE forecast time when the assimilation window ends (second)
LTIMESLOT=$LCYCLE     # Timeslot interval for 4D-LETKF (second)
```

#### Ensemble size and node usage

Ensemble size is set by `MEMBER` in config.main as follows. `PPN=4` and `THREADS=12` are the specific recommended settings for Fugaku. `SCALE_NP_X` and `SCALE_NP_Y` must be consistent with `NPROCS_X` and `NPROCS_Y` in config.nml.scale respectively.

config.main

```
MEMBER=50           # Ensemble size

PPN=4               # Number of processes per node

THREADS=12          # Number of threads per process

SCALE_NP_X=4          # (= PRC_NUM_X)
SCALE_NP_Y=2          # (= PRC_NUM_Y)
SCALE_NP=$(( SCALE_NP_X  * SCALE_NP_Y ))
```

#### DA cycle period

A DA cycle experiment consists of multiple DA steps of the interval defined by `$LCYCLE`. The time of the first and last DA steps are defined as `STIME` and `ETIME` in `config.cycle`. Note that those are the initial times of the corresponding DA steps of the length `$LCYCLE`. Therefore, in the following example with `LCYCLE=21600`, the two DA steps are performed and the final analysis time is `20220101120000`. `TIME_LIMIT` stands for the estimated elapse time of the DA cycle experiment.

config.cycle

```
  STIME='20220101000000'
  ETIME='20220101060000'
  TIME_LIMIT='00:30:00'
```

**Other DA settings**

Some settings of the DA experiment, such as analysis and first guess members to output, can be controlled by parameters in `config.cycle`. Others can be modified by directly editing `config.nml.letkf`.

## 4.4 Ideal experiments with synthetic observation

Ideal experiments with synthetic observation data can be done with either real or ideal meteorological conditions. In both cases, we first assume the time series of 'nature run', which is considered as the true evolution of the state variables. Then we generate synthetic observation from the nature run, which is used in the data assimilation experiment. We can set the details of the experiment as we like, such as observation coverage, variables, accuracy, as well as forecast model imperfections compared to the nature run.

### 4.4.1 SCALE-RM setting and nature run

First, you need to set up the SCALE-RM to create the nature run. The complexity of the model varies depending on the type of the experiment. We can use some real atmospheric data for the nature run, such as a downscaled forecast from the global forecast obtained by an operational center. Or we can also use an idealized setup without topography and land data. We can also use cyclic boundary conditions and run the model without external boundary data.

Note that the model setup for the calculation of first guess in data assimilation can be different from that of the nature run. For example, we can use coarser grid spacing or different version of physical parameterization scheme. We can also set different initial or boundary condition from the nature run, to investigate the impact of those imperfect information.

### 4.4.2 Synthetic observation

Once the nature run time series data is created, the synthetic observation can be generated from it using `obsmake` executable. It requires the input file, which has the same format with SCALE-LETKF observation data format, to specify observation type, location, and random error. The sample fortran files to create input files for dynamical and radar observation cases can be found in the directory `config/supercell/make_obsin/`. You can modify it or make your own script to generate the observation input file for your purpose.

The launcher of `obsmake` is `obsmake_run.sh`, which can be found in `config/supercell` or `config/barocwave`. You are supposed to edit `config.obsmake` and the part of `obsmake_run.sh` to make it accord with your experiment. The necessary input data along with an observation input file is the set of both restart and history files of the nature run.

obsmake_run.sh

```
conf_file=$TMP/config/obsmake_${time}.conf
cat $SCRP_DIR/config.nml.obsmake | sed \
    -e "/!--PPN--/a PPN=$PPN,"  \
    -e "/!--PRC_DOMAINS--/a PRC_DOMAINS=$SCALE_NP,"  \
    -e "/!--OBS_IN_NAME--/a OBS_IN_NAME=\"$TMP/obsin/obsin.dat\","  \
    -e "/!--OBS_IN_FORMAT--/a OBS_IN_FORMAT=\"${OBS_IN_FORMAT}\","  \
    -e "/!--LETKF_TOPOGRAPHY_IN_BASENAME--/a
    ↪  LETKF_TOPOGRAPHY_IN_BASENAME=\"$OUTDIR/const/topo/topo\"," \
    -e "/!--HISTORY_IN_BASENAME--/a HISTORY_IN_BASENAME=\"$OUTDIR/nature/hist/history\"," \
    -e "/!--GUES_IN_BASENAME--/a
    ↪  GUES_IN_BASENAME=\"$OUTDIR/nature/init/init_$(datetime_scale $time)\"," \
    -e "/!--SLOT_START--/a SLOT_START=$nslot,"  \
    -e "/!--SLOT_END--/a SLOT_END=$nslot,"  \
    -e "/!--SLOT_BASE--/a SLOT_BASE=$nslot,"  \
    -e "/!--SLOT_TINTERVAL--/a SLOT_TINTERVAL=$FCSTOUT,"  \
> $conf_file
```

### 4.4.3 Initial and boundary condition

To initialize the ensemble data assimilation with LETKF, you need the ensemble of initial data. The design of ensemble initial perturbation (and boundary data perturbation as well in some cases) is important for the performance of the DA cycle. The common way to generate an ensemble of initial condition is to add random perturbation to the nature run data at that time step. In the case of atmospheric state variables, the perturbation should have spatial structure, with a dominant spatial scale much larger than grid spacing. The SCALE-LETKF code includes a sample python program to generate structured random perturbation using Fourier transform and add it to the restart files. You can generate a set of ensemble initial restart files by using a shell script `init_perturb/init_perturb.sh`. The amplitude, location and spatial scale of the perturbation can be set in `init_perturb/init_perturb.py` as follows.

init_perturb.py

```
wavel1 =  500000.  ### short-wave cutoff wavelength
wavel2 = 2000000.  ### long-wave  cutoff wavelength
dx = 200000.       ### horizontal grid spacing
zheight = 16000.   ### top height for perturbation (can be below the model top)
taper_width = 10  ### number of grid points to suppress perturbation near the lateral
↪  boundary
taper_mtop = 10   ### number of grid points to suppress perturbation near the top level

pert_std = 1.0    ### STANDARD DEVIATION (in Kelvin) of temperature perturbation
halo = 2          ### number of grid points for HALO
```

# Chapter 5

# Performance monitoring of LETKF

## 5.1 Quicklook

The quick summary of observation data in the input file is shown in the log file of LETKF in various forms.

### 5.1.1 Observation counts

The summary of observation counts before and after QC for each observation type and variable is shown in tables as follows (partially omitted).

log/letkf/NOUT.3.0

```
OBSERVATION COUNTS BEFORE QC (GLOBAL):
================================================================~=========
TYPE          U       V       T       Tv       Q       RH       PS ~    TOTAL
----------------------------------------------------------------~---------
ADPUPA      614     614       0      100      36        0        0 ~     1364
AIRCAR        0       0       0        0       0        0        0 ~        0
AIRCFT        0       0       0        0       0        0        0 ~        0
SATWND    69485   69485       0        0       0        0        0 ~   138970
PROFLR       11      11       0        0       0        0        0 ~       22
VADWND        7       7       0        0       0        0        0 ~       14
SATEMP        0       0       0        0       0        0        0 ~        0
ADPSFC        0       0       0        0       0        0     5062 ~     5062
SFCSHP      515     515     275      213     213        0      938 ~     2669
SFCBOG        0       0       0        0       0        0        0 ~        0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
WDSATR        0       0       0        0       0        0        0 ~        0
ASCATW     2575    2575       0        0       0        0        0 ~     5150
TMPAPR        0       0       0        0       0        0        0 ~        0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
----------------------------------------------------------------~---------
TOTAL     73207   73207     275      313     249        0     6000 ~   153251
================================================================~=========

OBSERVATION COUNTS AFTER QC (GLOBAL):
================================================================~=========
TYPE          U       V       T       Tv       Q       RH       PS ~    TOTAL
----------------------------------------------------------------~---------
ADPUPA      146     147       0       89      32        0        0 ~      414
AIRCAR        0       0       0        0       0        0        0 ~        0
AIRCFT        0       0       0        0       0        0        0 ~        0
SATWND    14496   14496       0        0       0        0        0 ~    28992
PROFLR        0       0       0        0       0        0        0 ~        0
VADWND        7       7       0        0       0        0        0 ~       14
SATEMP        0       0       0        0       0        0        0 ~        0
ADPSFC        0       0       0        0       0        0     1631 ~     1631
SFCSHP        4       4       0        4       4        0      362 ~      378
SFCBOG        0       0       0        0       0        0        0 ~        0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
WDSATR        0       0       0        0       0        0        0 ~        0
ASCATW        0       0       0        0       0        0        0 ~        0
TMPAPR        0       0       0        0       0        0        0 ~        0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
----------------------------------------------------------------~---------
TOTAL     14653   14654       0       93      36        0     1993 ~    31429
================================================================~=========
```

Another table of the observation counts including numbers for a specific subdomain is provided in the following format.

log/letkf/NOUT.3.0

```
OBSERVATION COUNTS (GLOBAL AND IN THIS SUBDOMAIN #     0):
=====================================================================
TYPE   VAR      GLOBAL     GLOBAL  SUBDOMAIN  SUBDOMAIN EXT_SUBDOMAIN
             before QC   after QC  before QC   after QC      after QC
---------------------------------------------------------------------
ADPUPA   U         614        146        429          0           146
ADPUPA   V         614        147        429          0           147
ADPUPA  Tv         100         89          0          0            89
ADPUPA   Q          36         32          0          0            32
SATWND   U       69485      14496      56194       1205          6730
SATWND   V       69485      14496      56194       1205          6730
PROFLR   U          11          0         11          0             0
PROFLR   V          11          0         11          0             0
VADWND   U           7          7          0          0             7
VADWND   V           7          7          0          0             7
ADPSFC  PS        5062       1631       2944        179          1203
SFCSHP   U         515          4        329          0             2
SFCSHP   V         515          4        329          0             2
SFCSHP   T         275          0        145          0             0
SFCSHP  Tv         213          4        138          0             2
SFCSHP   Q         213          4        138          0             2
SFCSHP  PS         938        362        581         26           227
ASCATW   U        2575          0       2575          0             0
ASCATW   V        2575          0       2575          0             0
---------------------------------------------------------------------
TOTAL           153251      31429     123022       2615         15326
=====================================================================
```

These tables serve for quick check of the observation data input.

### 5.1.2 Observation departure

The following tables summarize the observation departure statistics. It only shows the tables for first guess, but the same output for analysis follows as well. It shows the number of observation and BIAS (mean of O-B/O-A) and RMSE (root mean of O-B/O-A squared) averaged over them for each variable. Note that they are calculated over all the observation regardless of location. Therefore, for example, the statistics of wind observation may take average including lower and upper atmosphere, and the statistics of radar reflectivity may take average over the areas of heavy rain and weak rain. More detailed analysis can be performed using the observation departure output file described in the following.

log/letkf/NOUT.3.0

```
OBSERVATIONAL DEPARTURE STATISTICS [GUESS] (IN THIS SUBDOMAIN):
=================================================================
                 U           V           T           Q          PS
-----------------------------------------------------------------
BIAS     8.085E-01   3.820E-01         N/A         N/A  -1.987E+02
RMSE     2.066E+00   2.409E+00         N/A         N/A   2.354E+02
NUMBER        1205        1205           0           0         182
=================================================================
OBSERVATIONAL DEPARTURE STATISTICS [GUESS] (GLOBAL):
=================================================================
                 U           V           T           Q          PS
-----------------------------------------------------------------
BIAS     7.833E-02   7.777E-01   3.025E-03   3.890E-05  -1.303E+02
RMSE     2.892E+00   2.725E+00   1.708E+00   4.731E-04   2.111E+02
NUMBER       14653       14654          93          36        1378
=================================================================
```

## 5.2 Observation statistics file

Detailed information of observation statistics can obtained from the additional output file in a binary format, by setting the namelist parameter `OBSDEP_OUT = 1` in `config.cycle`. By default, LETKF generates a file `$OUTDIR/<analysis time>/obsdep/obsdep.dat` for each data assimilation step. It contains the information of the differences between observation and analysis, the difference between observation and first guess, and ensemble spread of first guess, for each assimilated observation, along with other information such as location and observation error standard deviation. The file format is described in Observation departure file format section.

## 5.3 First guess and analysis ensemble statistics

LETKF output files include ensemble mean of analysis and first guess. The analysis increment can be obtained simply by taking the difference between the analysis and first guess mean. Note that an analysis file takes the SCALE restart file format, which has a specific set of variables such as `MOMX` (x-component of wind multiplied by density) and `RHOT` (potential temperature multiplied by density). The common variables such as U, V, and T must be manually derived from them for each member.

### 5.3.1 Ensemble spread file format

When `SPRD_OUT=1` is set in `config.cycle`, the ensemble spread files in `gues/sprd` and `anal/sprd` are generated by `letkf` by overwriting existing files. The spread file has the similar restart file format, but has a different list of variables. As the current LETKF program does not have a function to edit NetCDF metadata, note that the variable set **does not correspond** to the variable name list in a NetCDF file.

The actual variables in a spread file are as follows.

| variable name in NetCDF file | actual variable in sprd file |
|---|---|
| DENS | ensemble spread of U |
| MOMX | ensemble spread of V |
| MOMY | ensemble spread of W |
| MOMZ | ensemble spread of T |
| RHOT | ensemble spread of P |
| Q | ensemble spread of Q |

Table 5.1: Variables in spread NetCDF file

## 5.4 Elapse time monitoring

LETKF log file has the record of elapse time taken by each component of the program. You can easily see it by `grep` command. The first and second values correspond to the elapse time on a chosen process and the total elapse time for all the processes including waiting time, respectively.

```
$ grep -e "##### TIMER" NOUT.3.0
##### TIMER # INITIALIZE        0.277521      0.277727
##### TIMER # READ_OBS          0.155641      0.155646
##### TIMER # OBS_OPERATOR      5.124453      5.126911
##### TIMER # PROCESS_OBS       0.043350      0.043997
##### TIMER # SET_GRID          0.073217      0.077331
##### TIMER # READ_GUES         0.478075      0.490939
##### TIMER # GUES_MEAN         0.074179      2.824000
##### TIMER # DAS_LETKF         7.158692     15.069771
##### TIMER # ANAL_MEAN         0.000805      0.000818
##### TIMER # WRITE_ANAL        2.773918      3.054343
##### TIMER # FINALIZE          0.047013      0.047026
```

# Chapter 6

# Auxilirary

## 6.1 Observation file format

Observation data is processed in the format specific to the SCALE-LETKF. Currently, different formats are used for radar and other observation types. The parameter `OBS_IN_FORMAT` needs to be properly set in order to read the data.

### 6.1.1 Common format

The common observation format used in SCALE-LETKF consists of 8-record elements of 4-byte floating-point value for each observation. Note that the access type is *sequential* and not *direct*. The following is a part of the subroutine `write_obs` in `scale/common/common_obs_scale.f90` (simplified for brevity).

scale/common/common_obs_scale.f90

```
OPEN(iunit,FILE=cfile,FORM='unformatted',ACCESS='sequential')
DO n=1,obs%nobs
    wk(1) = REAL(obs%elm(n),r_sngl)
    wk(2) = REAL(obs%lon(n),r_sngl)
    wk(3) = REAL(obs%lat(n),r_sngl)
    wk(4) = REAL(obs%lev(n),r_sngl)
    wk(5) = REAL(obs%dat(n),r_sngl)
    wk(6) = REAL(obs%err(n),r_sngl)
    wk(7) = REAL(obs%typ(n),r_sngl)
    wk(8) = REAL(obs%dif(n),r_sngl)
    WRITE(iunit) wk
END DO
```

The 8 elements of the observation data array are as follows.

### 6.1.2 Radar observation format

For the radar observation, there is a slight modification in the data format. First, the location of the radar is recorded at the beginning of the file. Second, as it is common to use 3D-LETKF for radar data, the last record describing a time gap is omitted unless the parameter `RADAR_OBS_4D` is not set. See the part of the subroutine `write_obs_radar` (again simplified).

| Record # | Description |
|---|---|
| 1 | Observation variable (integer code defined in `common_obs_scale.f90`) |
| 2 | Longitude (degree) |
| 3 | Latitude (degree) |
| 4 | Vertical level (hPa or meter, see below) |
| 5 | Observed value |
| 6 | Observation error standard deviation |
| 7 | Observation type (integer code defined in `common_obs_scale.f90`) |
| 8 | Difference of observation time from the target time of data assimilation (second) |

Table 6.1: Observation data elements

scale/common/common_obs_scale.f90

```
IF(RADAR_OBS_4D) THEN
    nrec = 8
  ELSE
    nrec = 7
  END IF

  OPEN(iunit,FILE=cfile,FORM='unformatted',ACCESS='sequential')

  WRITE(iunit) REAL(obs%meta(1),r_sngl)
  WRITE(iunit) REAL(obs%meta(2),r_sngl)
  WRITE(iunit) REAL(obs%meta(3),r_sngl)

  DO n=1,obs%nobs
      wk(1) = REAL(obs%elm(n),r_sngl)
      wk(2) = REAL(obs%lon(n),r_sngl)
      wk(3) = REAL(obs%lat(n),r_sngl)
      wk(4) = REAL(obs%lev(n),r_sngl)
      wk(5) = REAL(obs%dat(n),r_sngl)
      wk(6) = REAL(obs%err(n),r_sngl)
      wk(7) = REAL(obs%typ(n),r_sngl)
      IF(RADAR_OBS_4D) THEN
        wk(8) = REAL(obs%dif(n),r_sngl)
      END IF
      WRITE(iunit) wk(1:nrec)
    end if
  END DO
```

At the beginning of the file, three 4-byte floating point real values are recorded. They are longtitude, latitude, elevation (m) of the radar location, respectively.

### 6.1.3 Note on vertical coordinate

Some settings in LETKF about vertical location or length implicitly assume a specific unit. It might be confusing and cause errors that are difficult to track down. This note serves to help users avoid errors by the inconsistency of vertical unit.

In the LETKF observation data format, vertical location of the observation is indicated in the 4-th element of each record. The coordinate unit depends on the observation data type and element as follows.

Vertical localization length scales are defined for each observation type in the parameter `VERT_LOCAL` of the namelist `PARAM_LETKF_OBS`. The unit is assumed depending on the observation type as follows.

| type # | type name | element # | element name | unit |
|---|---|---|---|---|
| 22 | 'PHARAD' | 4001-4004 | 'REF', 'Vr', 'PRH', 'RE0' | meter |
| 1 | 'ADPUPA' | 14593 | 'PS' | meter |
| else | - | else | - | hPa |

Table 6.2: Vertical coordinate of observation data

| type # | type name | unit |
|---|---|---|
| 22 | 'PHARAD' | meter |
| else | - | log-pressure |

Table 6.3: Vertical coordinate in localization

Therefore, in the array `vert_local`, the 22-th element corresponding to 'PHARAD' is interpreted as 'm' and the rest of the elements are as 'log-p'. Currently, the LETKF code supports only observation type 1 ('ADPUPA') and 22 ('PHARAD'). In the case if you modify the code to include other type of observation, be sure to set consistent vertical unit in the observation data file and the localization parameter.

## 6.2 Observation departure file format

Observation departure statistics output is enabled when the namelist parameter `OBSDEP_OUT` of `PARAM_LETKF_MONITOR` is set.

The common obsdep file format used in SCALE-LETKF consists of 12-record elements of 4-byte floating-point value for each observation. Note that the access type is *sequential* and not *direct*. The following is a part of the subroutine `write_obs_dep` in `scale/common/common_obs_scale.f90` (simplified for brevity).

scale/common/common_obs_scale.f90

```
OPEN(iunit,FILE=cfile,FORM='unformatted',ACCESS='sequential')
DO n=1,obs%nobs
  wk(1) = real(obs(set(n))%elm(idx(n)), r_sngl)
  wk(2) = real(obs(set(n))%lon(idx(n)), r_sngl)
  wk(3) = real(obs(set(n))%lat(idx(n)), r_sngl)
  wk(4) = real(obs(set(n))%lev(idx(n)), r_sngl)
  wk(5) = real(obs(set(n))%dat(idx(n)), r_sngl)
  wk(6) = real(obs(set(n))%err(idx(n)), r_sngl)
  wk(7) = real(obs(set(n))%typ(idx(n)), r_sngl)
  wk(8) = real(obs(set(n))%dif(idx(n)), r_sngl)
  wk(9) = real(qc(n), r_sngl)
  wk(10) = real(omb(n), r_sngl)
  wk(11) = real(oma(n), r_sngl)
  wk(12) = real(spr(n), r_sngl)
  WRITE(iunit) wk
END DO
```

The 12 elements of the observation departure data array are as follows.

| Record # | Description |
|---|---|
| 1 | Observation variable (integer code defined in `common_obs_scale.f90`) |
| 2 | Longitude (degree) |
| 3 | Latitude (degree) |
| 4 | Vertical level (hPa or meter, see Notes on vertical coordinate) |
| 5 | Observed value |
| 6 | Observation error standard deviation |
| 7 | Observation type (integer code defined in `common_obs_scale.f90`) |
| 8 | Difference of observation time from the target time of data assimilation (second) |
| 9 | QC flag (integer code defined in `common_obs_scale.f90`) |
| 10 | O-B |
| 11 | O-A |
| 12 | First guess ensemble spread in the observation space |

Table 6.4: Observation departure data elements

## 6.3 Preparation of NCEP GFS data

### 6.3.1 Data source

NCEP Global Forecast System (GFS) is a global numerical prediction model developed and used in NOAA NCEP, the US national organization. The GFS forecast data is distributed online and used worldwide in a real time manner.

NCEP provides various products based on the GFS model. In SCALE user's guide, the use of NCEP Final (FNL) operational analysis data for the boundary condition of SCALE real-world experiment is explained. Here, this section explains the use of the operational forecast data, which enables a real-time data assimilation cycle.

The example script uses the forecast data with a 0.5 degree grid spacing downloaded from the following addresses.

```
### archive data
 https://www.ncei.noaa.gov/data/global-forecast-system/access/grid-004-0.5-degree/forecast/
### realtime data
 https://nomads.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/
```

### 6.3.2 Download and convert

Use the script `scale-letkf-dev/scale/run/get_data_ext/ncepgfs/get_ncep_gfs.sh` with the initial time of the forecast.

```
./get_ncep_gfs.sh 20250801000000
```

Note that the script requires wgrib2 for data conversion from grib2 format to GrADS binary format. After you run the script, you will have the raw and converted data in the directories as follows.

```
### raw data
$ ls (your directory where scale is located)/external/ncepgfs/2025080100/
gfs.20250801000000  gfs.20250801060000  gfs.20250801120000  gfs.20250801180000
gfs.20250801030000  gfs.20250801090000  gfs.20250801150000
### GrADS binary data
$ ls (your directory where scale is located)/external/ncepgfs_grads/2025080100/
atm_20250801000000.grd   land_20250801090000.grd  reg_20250801180000.grd
atm_20250801030000.grd   land_20250801120000.grd  reg.ctl
atm_20250801060000.grd   land_20250801150000.grd  sfc_20250801000000.grd
atm_20250801090000.grd   land_20250801180000.grd  sfc_20250801030000.grd
atm_20250801120000.grd   land.ctl                 sfc_20250801060000.grd
atm_20250801150000.grd   reg_20250801000000.grd   sfc_20250801090000.grd
atm_20250801180000.grd   reg_20250801030000.grd   sfc_20250801120000.grd
atm.ctl                  reg_20250801060000.grd   sfc_20250801150000.grd
land_20250801000000.grd  reg_20250801090000.grd   sfc_20250801180000.grd
land_20250801030000.grd  reg_20250801120000.grd   sfc.ctl
land_20250801060000.grd  reg_20250801150000.grd
```

You can use GrADS to quickly look into binary format data, with control files such as `atm.ctl`.

### 6.3.3 Namelist file for scale-rm_init

When you use NCEP GFS binary data as parent model data for initial and boundary conditions, `scale-rm_init` (and its wrapper `scale-rm_init_ens`) needs a corresponding namelist parameters, which are provided by the file `config.nml.grads_boundary` in `scale/run`.

scale/common/common_obs_scale.f90

```
#
# Dimension
#
&GrADS_DIMS
 nx     = 720
 ny     = 361
 nz     = 41
/

#
# Variables
#
&GrADS_ITEM  name='lon',   dtype='linear',  swpoint=0.0d0,   dd=0.5d0 /
&GrADS_ITEM  name='lat',   dtype='linear',  swpoint=-90.0d0, dd=0.5d0 /
&GrADS_ITEM  name='plev',  dtype='levels',  lnum=41,
lvars=100000,97500,95000,92500,90000,85000,80000,75000,70000,65000,60000,
    55000,50000,45000,40000,35000,30000,25000,20000,15000,10000,7000,
    5000,4000,3000,2000,1500,1000,700,500,300,200,100,70,40,20,10,7,4,2,1, /
&GrADS_ITEM  name='MSLP',  dtype='map',   fname='--DIR--/bdysfc', startrec=1,  totalrec=7  /
&GrADS_ITEM  name='PSFC',  dtype='map',   fname='--DIR--/bdysfc', startrec=2,  totalrec=7  /
&GrADS_ITEM  name='U10',   dtype='map',   fname='--DIR--/bdysfc', startrec=3,  totalrec=7  /
&GrADS_ITEM  name='V10',   dtype='map',   fname='--DIR--/bdysfc', startrec=4,  totalrec=7  /
&GrADS_ITEM  name='T2',    dtype='map',   fname='--DIR--/bdysfc', startrec=5,  totalrec=7  /
&GrADS_ITEM  name='RH2',   dtype='map',   fname='--DIR--/bdysfc', startrec=6,  totalrec=7  /
&GrADS_ITEM  name='topo',  dtype='map',   fname='--DIR--/bdysfc', startrec=7,  totalrec=7  /
&GrADS_ITEM  name='HGT',   dtype='map',   fname='--DIR--/bdyatm', startrec=1,  totalrec=205/
&GrADS_ITEM  name='U',     dtype='map',   fname='--DIR--/bdyatm', startrec=42, totalrec=205/
&GrADS_ITEM  name='V',     dtype='map',   fname='--DIR--/bdyatm', startrec=83, totalrec=205/
&GrADS_ITEM  name='T',     dtype='map',   fname='--DIR--/bdyatm', startrec=124,totalrec=205/
&GrADS_ITEM  name='RH',    dtype='map',   fname='--DIR--/bdyatm', startrec=165,totalrec=205/
&GrADS_ITEM  name='llev',  dtype='levels',lnum=4, lvars=0.05,0.25,0.70,1.50, /
&GrADS_ITEM  name='lsmask',dtype='map',   fname='--DIR--/bdyland', startrec=1, totalrec=10 /
&GrADS_ITEM  name='SKINT', dtype='map',   fname='--DIR--/bdyland', startrec=2, totalrec=10 /
&GrADS_ITEM  name='STEMP', dtype='map',   fname='--DIR--/bdyland', nz=4, startrec=3,
↪  totalrec=10, missval=9.999e+20 /
&GrADS_ITEM  name='SMOISVC', dtype='map',   fname='--DIR--/bdyland', nz=4, startrec=7,
↪  totalrec=10, missval=9.999e+20 /
```

For the details of these parameter settings, see the part "Input from binary format data" in Section "4.1.2 How to prepare initial and boundary data" of the SCALE user's guide.

## 6.4 Preparation of NCEP PREPBUFR data

### 6.4.1 Data source

PREPBUFR is a final post-processed product of observation data used for NCEP global data assimilation system (GDAS). The data is provided in Binary Universal Form for the Representation of meteorological data (BUFR) format. The PREPBUFR is provided every 6 hours, which is the interval of GDAS. The data includes various observation types within 6-hour window centered at the assimilation time.

The near real-time PREPBUFR data can be downloaded from the following address.

```
https://nomads.ncep.noaa.gov/pub/data/nccf/com/obsproc/prod/
```

### 6.4.2 Download and convert

You need to convert PREPBUFR data from BUFR format to LETKF observation file format. You can use the script `scale-letkf-dev/scale/run/get_data_ext/ncepgfs/get_ncep_gfs.sh` to download and convert the data.

Before use it, you need to prepare the binary file of the BUFR decoder. First, install the library bufrlib provided by NCEP to your environment. Then specify the path as `BUFR_LIB` in `arch/configure.user.FUGAKU` and compile the program `dec_prepbufr`.

```
cd scale/obs
make dec_prepbufr
```

Once you have a binary file `scale/obs/dec_prepbufr`, run the script `scale-letkf-dev/scale/run/get_data_ext/ncepobs/get_ncep_obs.sh` with the assimilation time.

```
./get_ncep_obs.sh 20250801000000
```

and you have observation data in the SCALE-LETKF format.

```
### raw data
$ ls (your directory where scale is located)/external/ncepobs_gdas/2025080100/
prepbufr.2025080100
### LETKF format data
$ ls (your directory where scale is located)/external/ncepobs_gdas_letkf/2025080100/
obs_20250801000000.dat
```

## 6.5 Preparation of JMA MSM data

### 6.5.1 Data source

You have an option to use Japan Meteorological Agency (JMA) mesoscale model (MSM) data as a parent model data for your real-world experiment with the

SCALE-LETKF. The real-time data is provided by Japan Meteorological Busuiness Support Center (JMBSC) for contracted users. With several more hours of latency, the near-real-time and past data can be downloaded from RISH data server of Kyoto Univ.

You can find the documentation of JMA models from JMA web page.

### 6.5.2 Download and convert

Similarly as in the case of NCEP GFS, use the script to download and convert the JMA MSM data.

```
./get_msm.sh 20250801000000
```

Check the converted data in the directory specified in the script.

```
### GrADS binary data
$ ls (your directory where scale is located)/external/jma/msm/2025080100/
atm_20250801000000.grd  atm_20250801150000.grd  sfc_20250801090000.grd
atm_20250801030000.grd  atm.ctl                 sfc_20250801120000.grd
atm_20250801060000.grd  sfc_20250801000000.grd  sfc_20250801150000.grd
atm_20250801090000.grd  sfc_20250801030000.grd  sfc.ctl
atm_20250801120000.grd  sfc_20250801060000.grd
```

Importantly, note that the JMA MSM data obtained in this manner does not cover all the necessary variables for `scale-rm_init`. Specifically, at least surface topography and soil temperature and moisture data are missing. You need to consider using some additional data source for these variables or getting more complete JMA data in some other ways.

# Bibliography

Brian R. Hunt, Eric J. Kostelich, and Istvan Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter. *Physica D: Nonlinear Phenomena*, 230(1-2):112–126, jun 2007. ISSN 0167-2789. doi: 10.1016/J.PHYSD.2006.11.008.

Guo-Yuan Lien, Takemasa Miyoshi, Seiya Nishizawa, Ryuji Yoshida, Hisashi Yashiro, Sachiho A. Adachi, Tsuyoshi Yamaura, and Hirofumi Tomita. The Near-Real-Time SCALE-LETKF System: A Case of the September 2015 Kanto-Tohoku Heavy Rainfall. *Sola*, 13(0):1–6, 2017. ISSN 1349-6476. doi: 10.2151/sola.2017-001.

Takemasa Miyoshi, Masaru Kunii, Juan Ruiz, Guo-Yuan Lien, Shinsuke Satoh, Tomoo Ushio, Kotaro Bessho, Hiromu Seko, Hirofumi Tomita, and Yutaka Ishikawa. "Big Data Assimilation" Revolutionizing Severe Weather Prediction. *Bulletin of the American Meteorological Society*, 97(8):1347–1354, 2016a. ISSN 0003-0007. doi: 10.1175/BAMS-D-15-00144.1.

Takemasa Miyoshi, Guo-Yuan Lien, Shinsuke Satoh, Tomoo Ushio, Kotaro Bessho, Hirofumi Tomita, Seiya Nishizawa, Ryuji Yoshida, Sachiho A. Adachi, Jianwei Liao, Balazs Gerofi, Yutaka Ishikawa, Masaru Kunii, Juan Ruiz, Yasumitsu Maejima, Shigenori Otsuka, Michiko Otsuka, Kozo Okamoto, and Hiromu Seko. "Big Data Assimilation" Toward Post-Petascale Severe Weather Prediction: An Overview and Progress. *Proceedings of the IEEE*, 104(11): 2155–2179, 2016b. ISSN 0018-9219. doi: 10.1109/JPROC.2016.2602560.