UFS Offline Land DA User's Guide

Release v1.0.0

Mar 06, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

This User's Guide provides guidance for running the Unified Forecast System (*UFS*) offline Land Data Assimilation (DA) System. Land DA is an offline version of the Noah Multi-Physics (Noah-MP) land surface model (LSM) used in the UFS Weather Model (WM). Its data assimilation framework uses the Joint Effort for Data assimilation Integration (*JEDI*) software. The offline UFS Land Data Assimilation (Land DA) System currently only works with snow data. Thus, this User's Guide focuses primarily on the snow DA process.

This User's Guide is organized as follows:

- This chapter (Introduction) provides background information on the Unified Forecast System (*UFS*) and the NoahMP model.
- Chapter 2 (Technical Overview) outlines prerequisites, user support levels, and directory structure.
- Chapter 3 (Land DA Workflow [Hera & Orion]) explains how to build and run the Land DA System on *Level 1* systems (currently Hera and Orion).
- Chapter 4 (Land DA Workflow [in a Container]) explains how to build and run the containerized Land DA System on non-Level 1 systems.
- Chapter 5 (Model) provides information on input data and configuration parameters in the Noah-MP LSM and its Vector-to-Tile Converter.
- Chapter 6 (DA Framework) provides information on the DA system, required data, and configuration parameters.
- Chapter 7 (Glossary) lists important terms.

Users and developers may post questions and exchange information on the Land DA System's GitHub Discussions forum if their concerns are not addressed in this User's Guide.

The Land DA System citation is as follows and should be used when presenting results based on research conducted with the Land DA System:

UFS Development Team. (2023, March 6). Unified Forecast System (UFS) Land Data Assimilation (DA) System (Version v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.7675721

1.1 Background Information

1.1.1 Unified Forecast System (UFS)

The UFS is a community-based, coupled, comprehensive Earth modeling system. It includes multiple applications that support different forecast durations and spatial domains. NOAA's operational model suite for numerical weather prediction (*NWP*) is quickly transitioning to the UFS from many different modeling systems. For example, the UFS-based Global Forecast System (GFS) and the Global Ensemble Forecast System (GEFS) are currently in operational use. The UFS is designed to enable research, development, and contribution opportunities within the broader *Weather Enterprise* (including government, industry, and academia). For more information about the UFS, visit the UFS Portal.

1.1.2 Noah-MP

The offline Noah-MP LSM is a stand-alone, uncoupled model used to execute land surface simulations. In this traditional uncoupled mode, near-surface atmospheric forcing data is required as input forcing. This LSM simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snow depth, snow water equivalent (SWE), snow density, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance.

Noah-MP uses a big-leaf approach with a separated vegetation canopy accounting for vegetation effects on surface energy and water balances, a modified two-stream approximation scheme to include the effects of vegetation canopy gaps that vary with solar zenith angle and the canopy 3-D structure on radiation transfer, a 3-layer physically-based snow model, a more permeable frozen soil by separating a grid cell into a permeable fraction and impermeable fraction, a simple groundwater model with a TOPMODEL-based runoff scheme, and a short-term leaf phenology model. Noah-MP LSM enables a modular framework for diagnosing differences in process representation, facilitating ensemble forecasts and uncertainty quantification, and choosing process presentations appropriate for the application. Noah-MP developers designed multiple parameterization options for leaf dynamics, radiation transfer, stomatal resistance, soil moisture stress factor for stomatal resistance, aerodynamic resistance, runoff, snowfall, snow surface albedo, supercooled liquid water in frozen soil, and frozen soil permeability.

The Noah-MP LSM has evolved through community efforts to pursue and refine a modern-era LSM suitable for use in the National Centers for Environmental Prediction (NCEP) operational weather and climate prediction models. This collaborative effort continues with participation from entities such as NCAR, NCEP, NASA, and university groups.

Noah-MP has been implemented in the UFS via the *CCPP* physics package and is currently being tested for operational use in GFSv17 and RRFS v2. Noah-MP has also been used operationally in the NOAA National Water Model (NWM) since 2016. Details about the model's physical parameterizations can be found in Niu *et al.* [NYM+11] (2011).

1.2 Disclaimer

The United States Department of Commerce (DOC) GitHub project code is provided on an "as is" basis and the user assumes responsibility for its use. DOC has relinquished control of the information and no longer has a responsibility to protect the integrity, confidentiality, or availability of the information. Any claims against the Department of Commerce stemming from the use of its GitHub project will be governed by all applicable Federal laws. Any reference to specific commercial products, processes, or services by service mark, trademark, manufacturer, or otherwise, does not constitute or imply their endorsement, recommendation, or favoring by the Department of Commerce. The Department of Commerce seal and logo, or the seal and logo of a DOC bureau, shall not be used in any manner to imply endorsement of any commercial product or activity by DOC or the United States Government.

1.3 References

CHAPTER

TWO

TECHNICAL OVERVIEW

2.1 Prerequisites

2.1.1 Minimum System Requirements

UFS applications, models, and components require a UNIX-based operating system (i.e., Linux or MacOS).

Additionally, users will need:

- Disk space: ~23GB (11GB for Land DA System [or 6.5GB for Land DA container], 11GB for Land DA data, and ~1GB for staging and output)
- 6 CPU cores (or option to run with "oversubscribe")

2.1.2 Software Prerequisites

The Land DA System requires:

- An *MPI* implementation
- A Fortran compiler
- Python
- *NetCDF*
- Lmod
- spack-stack
- FV3-bundle

These software prerequisites are pre-installed in the Land DA *container* and on other Level 1 systems (see *below* for details). However, users on non-Level 1 systems will need to install them.

Before using the Land DA container, users will need to install Singularity and an **Intel** MPI (available free here).

2.2 Supported Systems for Running Land DA

Four levels of support have been defined for *UFS* applications, and the Land DA System operates under this paradigm:

- **Level 1** *(Pre-configured)*: Prerequisite software libraries are pre-built and available in a central location; code builds; full testing of model.
- **Level 2** *(Configurable)*: Prerequisite libraries are not available in a centralized location but are expected to install successfully; code builds; full testing of model.
- **Level 3** *(Limited-test platforms)*: Libraries and code build on these systems, but there is limited testing of the model.
- **Level 4** *(Build-only platforms)*: Libraries and code build, but running the model is not tested.

2.2.1 Level 1 Systems

Preconfigured (Level 1) systems for Land DA already have the required external libraries available in a central location via *spack-stack* and *JEDI*'s `fv3-bundle`. Land DA is expected to build and run out-of-the-box on these systems, and users can download the Land DA code without first installing prerequisite software. With the exception of the Land DA container, users must have access to these Level 1 systems in order to use them.

Platform	Compiler/MPI	spack-stack & fv3-bundle Installations
Hera	intel/2022.1.2 / impi/2022.1.2	/scratch1/NCEPDEV/nems/role.epic/spack-stack/envs/landda-release-1.0-intel /scratch1/NCEPDEV/nems/role.epic/contrib/fv3-bundle
Orion	intel/2022.1.2 / impi/2022.1.2	/work/noaa/epic-ps/role-epic-ps/spack-stack/envs/landda-release-1.0-intel /work/noaa/epic-ps/role-epic-ps/contrib/fv3-bundle
Container	intel-oneapi-compilers/2021.8.0 / intel-oneapi-mpi/2021.8.0	/opt/spack-stack/ (inside the container) /opt/fv3-bundle (inside the container)

2.2.2 Level 2-4 Systems

On non-Level 1 platforms, the Land DA System can be run within a container that includes the prerequisite software; otherwise, the required libraries will need to be installed as part of the Land DA build process. Once these prerequisite libraries are installed, applications and models should build and run successfully. However, users may need to perform additional troubleshooting on Level 3 or 4 systems since little or no pre-release testing has been conducted on these systems.

Note: Running on Jet, Cheyenne, and NOAA Cloud systems is supported via container.

2.3 Code Repositories and Directory Structure

2.3.1 Directory Structure

The main repository for the Land DA System is named `land-offline_workflow`; it is available on GitHub at https://github.com/NOAA-EPIC/land-offline_workflow/. A number of submodules are nested under the main `land-offline_workflow` directory. When the `release/public-v1.0.0` branch of the `land-offline_workflow` repository is cloned with the `--recursive` argument, the basic directory structure will be similar to the example below. Some files and directories have been removed for brevity.

```
land-offline_workflow
 ├── DA_update
 │      ├── add_jedi_incr
 │      ├── jedi/fv3-jedi
 │      └── do_LandDA.sh
 ├── cmake
 ├── configures
 ├── docs
 ├── modulefiles
 ├── test
 ├── ufs-land-driver
 │     └── ccpp-physics
 ├── vector2tile
 ├── CMakeLists.txt
 ├── README.md
 ├── LICENSE
 ├── do_submit_cycle.sh
 ├── release.environment
 ├── settings_DA_*
 ├── submit_cycle.sh
 └── template.*
```

2.3.2 Land DA Components

Table 2.1 describes the various subrepositories that form the UFS Land DA System.

Table 2.1: UFS Land DA System Components

Repository Name	Repository Description	Authoritative repository URL
land-DA_update	Contains scripts and components for performing data assimilation (DA) procedures.	https://github.com/NOAA-EPIC/land-DA_update
– *land-apply_jedi_incr*	Contains code that applies the JEDI-generated DA increment to UFS `sfc_data` restart	https://github.com/NOAA-PSL/land-apply_jedi_incr
ufs-land-driver	Repository for the UFS Land Driver	https://github.com/NOAA-EMC/ufs-land-driver
– *ccpp-physics*	Repository for the Common Community Physics Package (CCPP)	https://github.com/NCAR/ccpp-physics
land-vector2tile	Contains code to map between the vector format used by the Noah-MP offline driver, and the tile format used by the UFS atmospheric model.	https://github.com/NOAA-PSL/land-vector2tile

CHAPTER

THREE

LAND DA WORKFLOW (HERA & ORION)

This chapter provides instructions for building and running a basic Land DA case for the Unified Forecast System (*UFS*) Land DA System. This out-of-the-box Land DA case builds a weather forecast for January 1, 2016 at 18z to January 3, 2016 at 18z.

Attention: These steps are designed for use on *Level 1* systems (i.e., Hera and Orion) and may require significant changes on other systems. It is recommended that users on other systems run the containerized version of Land DA. Users may reference Chapter 4: Containerized Land DA Workflow for instructions.

3.1 Create a Working Directory

Create a directory for the Land DA experiment (`$LANDDAROOT`):

```
mkdir /path/to/landda
cd /path/to/landda
export LANDDAROOT=`pwd`
```

where `/path/to/landda` is the path to the directory where the user plans to run Land DA experiments.

3.2 Get Data

Table 3.1 shows the locations of pre-staged data on NOAA *RDHPCS* (i.e., Hera and Orion).

Table 3.1: Level 1 RDHPCS Data

Platform	Data Location
Hera	/scratch1/NCEPDEV/nems/role.epic/landda/inputs
Orion	/work/noaa/epic-ps/role-epic-ps/landda/inputs
Jet	/mnt/lfs4/HFIP/hfv3gfs/role.epic/landda/inputs
Cheyenne	/glade/work/epicufsrt/contrib/landda/inputs

Users can either set the `LANDDA_INPUTS` environment variable to the location of their system's pre-staged data or use a soft link to the data. For example, on Hera, users may set:

```
export LANDDA_INPUTS=/scratch1/NCEPDEV/nems/role.epic/landda/inputs
```

Alternatively, users can add a soft link to the data. For example, on Orion:

```
cd $LANDDAROOT
ln -s /work/noaa/epic-ps/role-epic-ps/landda/inputs .
```

Users who have difficulty accessing the data on Hera or Orion may download it according to the instructions in Section 4.3. Users with access to data for additional experiments may use the same process described above to point or link to that data by modifying the path to the data appropriately.

Users who are not using Land DA on Hera or Orion should view Chapter 4 for instructions on running the containerized version of Land DA. Section 4.3 explains options for downloading the sample data onto their system.

3.3 Get Code

Clone the Land DA repository.

```
git clone -b release/public-v1.0.0 --recursive https://github.com/NOAA-EPIC/land-offline_
↪workflow.git
```

3.4 Build the Land DA System

1. Navigate to the workflow directory, and source the modulefiles.

   ```
   cd $LANDDAROOT/land-offline_workflow
   module use modulefiles
   module load landda_<machine>.intel
   ```

   where <machine> is either hera or orion.

2. Create and navigate to a build directory.

   ```
   mkdir build
   cd build
   ```

3. Build the Land DA System.

   ```
   ecbuild ..
   make -j 8
   ```

   If the code successfully compiles, the console output should end with:

   ```
   [100%] Built target ufsLandDriver.exe
   ```

   Additionally, the build directory will contain several files and a bin subdirectory with three executables:

   - apply_incr.exe
   - ufsLandDriver.exe
   - vector2tile_converter.exe

3.5 Configure the Experiment

1. Navigate back to the `land-offline_workflow` directory and check that the account/partition is correct in `submit_cycle.sh`.

```
cd ..
vi submit_cycle.sh
```

If necessary, modify lines 3 and 4 to include the correct account and queue (qos) for the system. It may also be necessary to add the following line to the script to specify the partition:

```
#SBATCH --partition=my_partition
```

where `my_partition` is the name of the partition on the user's system.

2. Configure other elements of the experiment if desired. The v1.0.0 release includes four scripts with default experiment settings:

   - `settings_DA_cycle_gdas` for running the Jan. 1-3, 2016 sample case.
   - `settings_DA_cycle_era5` for running a Jan. 1-3, 2020 sample case.
   - `settings_DA_cycle_gdas_restart` for running the Jan. 3-4, 2016 sample case.
   - `settings_DA_cycle_era5_restart` for running a Jan. 3-4, 2020 sample case.

   These files contain reasonable default values for running a Land DA experiment. Users who wish to run a more complex experiment may change the values in these files and the files they reference using information in Chapters 5 & 6.

Note: The `*restart` settings files will only work after an experiment with the corresponding non-restart settings file has been run. These settings files are designed to use the restart files created by the first experiment cycle to pick up where it left off. For example, `settings_DA_cycle_gdas` runs from 2016-01-01 at 18z to 2016-01-03 at 18z. The `settings_DA_cycle_gdas_restart` will run from 2016-01-03 at 18z to 2016-01-04 at 18z.

3.6 Run an Experiment

The Land DA System uses a script-based workflow that is launched using the `do_submit_cycle.sh` script. This script requires an input file that details all the specifics of a given experiment.

```
./do_submit_cycle.sh settings_DA_cycle_gdas
```

The system will output a message such as `Submitted batch job ########`, indicating that the job was successfully submitted. If all goes well, two full cycles will run with data assimilation (DA) and a forecast.

3.7 Check Progress

Verify that the experiment ran successfully:

To check on the job status, users on a system with a Slurm job scheduler may run:

```
squeue -u $USER
```

To view progress, users can open the `log*` and `err*` files once they have been generated:

```
tail -f log* err*
```

The `log*` file for a successful experiment will end with an exit code of `0:0` and a message like:

```
Job 42442720 (not serial) finished for user User.Name in partition hera with exit code␣
↪0:0
```

The `err*` file for a successful experiment will end with something similar to:

```
+ THISDATE=2016010318
+ date_count=2
+ '[' 2 -lt 2 ']'
+ '[' 2016010318 -lt 2016010318 ']'
```

Users will need to hit `Ctrl+C` to exit the files.

Attention: If the log file contains a NetCDF error (e.g., `ModuleNotFoundError: No module named 'netCDF4'`), run:

```
python -m pip install netCDF4
```

Then, resubmit the job (`sbatch submit_cycle.sh`).

Next, check for the background and analysis files in the `cycle_land` directory.

```
ls -l ../cycle_land/DA_GHCN_test/mem000/restarts/vector/
```

CHAPTER

FOUR

CONTAINERIZED LAND DA WORKFLOW

These instructions will help users build and run a basic case for the Unified Forecast System (*UFS*) Land Data Assimilation (DA) System using a Singularity container. The Land DA *container* packages together the Land DA System with its dependencies (e.g., *spack-stack*, *JEDI*) and provides a uniform environment in which to build and run the Land DA System. Normally, the details of building and running Earth systems models will vary based on the computing platform because there are many possible combinations of operating systems, compilers, *MPIs*, and package versions available. Installation via Singularity container reduces this variability and allows for a smoother experience building and running Land DA. This approach is recommended for users not running Land DA on a supported *Level 1* system (i.e., Hera, Orion).

The out-of-the-box Land DA case described in this User's Guide builds a weather forecast for January 1, 2016 at 18z to January 3, 2016 at 18z.

Attention: This chapter of the User's Guide should **only** be used for container builds. For non-container builds, see Chapter 3, which describes the steps for building and running Land DA on a *Level 1 System* **without** a container.

4.1 Prerequisites

The containerized version of Land DA requires:

- Installation of Singularity
- At least 6 CPU cores
- An **Intel** compiler and *MPI* (available for free here)

4.1.1 Install Singularity

To build and run Land DA using a Singularity container, first install the Singularity package according to the Singularity Installation Guide. This will include the installation of dependencies and the installation of the Go programming language. SingularityCE Version 3.7 or above is recommended.

Note: Docker containers can only be run with root privileges, and users generally do not have root privileges on *HPCs*. However, a Singularity image may be built directly from a Docker image for use on the system.

4.2 Build the Container

4.2.1 Set Environment Variables

For users working on systems with limited disk space in their `/home` directory, it is important to set the `SINGULARITY_CACHEDIR` and `SINGULARITY_TMPDIR` environment variables to point to a location with adequate disk space. For example:

```
export SINGULARITY_CACHEDIR=/absolute/path/to/writable/directory/cache
export SINGULARITY_TMPDIR=/absolute/path/to/writable/directory/tmp
```

where `/absolute/path/to/writable/directory/` refers to a writable directory (usually a project or user directory within `/lustre`, `/work`, `/scratch`, or `/glade` on NOAA *RDHPCS* systems). If the `cache` and `tmp` directories do not exist already, they must be created with a `mkdir` command.

On NOAA Cloud systems, the `sudo su` command may also be required:

```
mkdir /lustre/cache
mkdir /lustre/tmp
sudo su
export SINGULARITY_CACHEDIR=/lustre/cache
export SINGULARITY_TMPDIR=/lustre/tmp
exit
```

Note: `/lustre` is a fast but non-persistent file system used on NOAA Cloud systems. To retain work completed in this directory, tar the files and move them to the `/contrib` directory, which is much slower but persistent.

4.2.2 Build the Container

Set a top-level directory location for Land DA work, and navigate to it. For example:

```
export LANDDAROOT=/path/to/landda
[[ -d $LANDDAROOT ]] || mkdir -p $LANDDAROOT
cd $LANDDAROOT
```

where `/path/to/landda` is the path to this top-level directory (e.g., `/Users/Joe.Schmoe/landda`). The second line will create the directory if it does not exist yet.

Hint: If a `singularity: command not found` error message appears in any of the following steps, try running: `module load singularity`.

NOAA RDHPCS Systems

On many NOAA *RDHPCS* systems, a container named `ubuntu20.04-intel-landda.img` has already been built, and users may access the container at the locations in Table 4.1.

Table 4.1: Locations of Pre-Built Containers

Machine	File location
Cheyenne	/glade/scratch/epicufsrt/containers
Hera	/scratch1/NCEPDEV/nems/role.epic/containers
Jet	/mnt/lfs4/HFIP/hfv3gfs/role.epic/containers
Orion	/work/noaa/epic-ps/role-epic-ps/containers

Note: Singularity is not available on Gaea, and therefore, container use is not supported on Gaea.

Users can simply set an environment variable to point to the container, as described in Section 4.4.1.

If users prefer, they may copy the container to their local working directory. For example, on Jet:

```
cp /mnt/lfs4/HFIP/hfv3gfs/role.epic/containers/ubuntu20.04-intel-landda.img .
```

Other Systems

On other systems, users can build the Singularity container from a public Docker *container* image or download the container from the Land DA Data Bucket. Downloading may be faster depending on the download speed on the user's system.

To download from the data bucket, users can run:

```
wget https://noaa-ufs-land-da-pds.s3.amazonaws.com/current_land_da_release_data/ubuntu20.
↪04-intel-landda.img
```

To build the container from a Docker image, users can run:

```
singularity build ubuntu20.04-intel-landda.img docker://noaaepic/ubuntu20.04-intel-
↪landda:release-public-v1.0.0
```

This process may take several hours depending on the system.

Note: Some users may need to issue the `singularity build` command with `sudo` (i.e., `sudo singularity build...`). Whether `sudo` is required is system-dependent. If `sudo` is required (or desired) for building the container, users should set the `SINGULARITY_CACHEDIR` and `SINGULARITY_TMPDIR` environment variables with `sudo su`, as in the NOAA Cloud example from Section 4.2.1 above.

4.3 Get Data

In order to run the Land DA System, users will need input data in the form of fix files, model forcing files, restart files, and observations for data assimilation. These files are already present on NOAA RDHPCS systems (see Section 3.1 for details).

Users on any system may download and untar the data from the Land DA Data Bucket into their `$LANDDAROOT` directory.

```
wget https://noaa-ufs-land-da-pds.s3.amazonaws.com/current_land_da_release_data/landda-
↪input-data-{YEAR}.tar.gz
tar xvfz landda-input-data-{YEAR}.tar.gz
```

replacing `{YEAR}` with either `2016` or `2020`. The default name for the untarred file is `inputs`.

4.4 Run the Container

To run the container, users must:

1. *Set up the container*
2. *Configure the experiment*
3. *Run the experiment*

4.4.1 Set Up the Container

Save the location of the container in an environment variable.

```
export img=path/to/ubuntu20.04-intel-landda.img
```

Set the `USE_SINGULARITY` environment variable to "yes".

```
export USE_SINGULARITY=yes
```

This variable tells the workflow to use the containerized version of all the executables (including python) when running a cycle.

Users may convert a container `.img` file to a writable sandbox. This step is required when running on Cheyenne but is optional on most other systems:

```
singularity build --sandbox ubuntu20.04-intel-landda $img
```

When making a writable sandbox on NOAA RDHPCS systems, the following warnings commonly appear and can be ignored:

```
INFO:    Starting build...
INFO:    Verifying bootstrap image ubuntu20.04-intel-landda.img
WARNING: integrity: signature not found for object group 1
WARNING: Bootstrap image could not be verified, but build will continue.
```

From within the `$LANDDAROOT` directory, copy the `land-offline_workflow` directory out of the container.

```
singularity exec -H $PWD $img cp -r /opt/land-offline_workflow .
```

There should now be a `land-offline_workflow` directory in the `$LANDDAROOT` directory. Navigate into the `land-offline_workflow` directory. If for some reason, this is unsuccessful, users may try a version of the following command instead:

```
singularity exec -B /<local_base_dir>:/<container_dir> $img cp -r /opt/land-offline_
↪workflow .
```

where `<local_base_dir>` and `<container_dir>` are replaced with a top-level directory on the local system and in the container, respectively. Additional directories can be bound by adding another `-B /<local_base_dir>:/<container_dir>` argument before the container location (`$img`).

Attention: Be sure to bind the directory that contains the experiment data!

Note: Sometimes binding directories with different names can cause problems. In general, it is recommended that the local base directory and the container directory have the same name. For example, if the host system's top-level directory is `/user1234`, the user may want to convert the `.img` file to a writable sandbox and create a `user1234` directory in the sandbox to bind to.

Navigate to the `land-offline_workflow` directory after it has been successfully copied into `$LANDDAROOT`.

```
cd land-offline_workflow
```

When using a Singularity container, Intel compilers and Intel *MPI* (preferably 2020 versions or newer) need to be available on the host system to properly launch MPI jobs. Generally, this is accomplished by loading a module with a recent Intel compiler and then loading the corresponding Intel MPI. For example, users can modify the following commands to load their system's compiler/MPI combination:

```
module load intel/2022.1.2 impi/2022.1.2
```

Note: *Spack-stack* uses lua modules, which require Lmod to be initialized for the `module load` command to work. If for some reason, Lmod is not initialized, users can source the `init/bash` file on their system before running the command above. For example, users can modify and run the following command:

```
source /path/to/init/bash
```

Then they should be able to load the appropriate modules.

4.4.2 Configure the Experiment

Users on a system with a Slurm job scheduler will need to make some minor changes to the `submit_cycle.sh` file. Open the file and change the account and queue (qos) to match the desired account and qos on the system. Users may also need to add the following line to the script to specify the partition. For example, on Jet, users should set:

```
#SBATCH --partition=xjet
```

Save and close the file.

4.4.3 Run the Experiment

The Land DA System uses a script-based workflow that is launched using the `do_submit_cycle.sh` script. That script requires an input file that details all the specifics of a given experiment. EPIC has provided four sample `settings_*` files as examples: `settings_DA_cycle_gdas`, `settings_DA_cycle_era5`, `settings_DA_cycle_gdas_restart`, and `settings_DA_cycle_era5_restart`. The `*restart` settings files will only work after an experiment with the corresponding non-restart settings file has been run. This is because they are designed to use the restart files created by the first experiment cycle to pick up where it left off. (e.g., `settings_DA_cycle_gdas` runs from 2016-01-01 at 18z to 2016-01-03 at 18z. The `settings_DA_cycle_gdas_restart` will run from 2016-01-03 at 18z to 2016-01-04 at 18z.)

To start the experiment, run:

```
./do_submit_cycle.sh settings_DA_cycle_gdas
```

The `do_submit_cycle.sh` script will read the `settings_DA_cycle_*` file and the `release.environment` file, which contain sensible experiment default values to simplify the process of running the workflow for the first time. Advanced users will wish to modify the parameters in `do_submit_cycle.sh` to fit their particular needs. After reading the defaults and other variables from the settings files, `do_submit_cycle.sh` creates a working directory (named `workdir` by default) and an output directory called `landda_expts` in the parent directory of `land-offline_workflow` and then submits a job (`submit_cycle.sh`) to the queue that will run through the workflow. If all succeeds, users will see `log` and `err` files created in `land-offline_workflow` along with a `cycle.log` file, which will show where the cycle has ended. The `landda_expts` directory will also be populated with data in the following directories:

```
landda_expts/DA_GHCN_test/DA/
landda_expts/DA_GHCN_test/mem000/restarts/vector/
```

Users can check experiment progress/success according to the instructions in Section 3.7, which apply to both containerized and non-containerized versions of the Land DA System.

CHAPTER

FIVE

NOAH-MP LAND SURFACE MODEL

This chapter provides practical information on input files and parameters for the Noah-MP Land Surface Model (LSM) and its Vector-to-Tile Converter component. For background information on the Noah-MP Land Surface Model (LSM), see Section 1.1.2 of the Introduction.

5.1 Input Files

The UFS land model requires multiple input files to run: static datasets (fix files containing climatological information, terrain, and land use data), initial conditions and forcing files, and model configuration files (such as namelists). Users may reference the Community Noah-MP User's Guide for a detailed technical description of certain elements of the Noah-MP model.

There are several important files used to specify model parameters: the static file (`ufs-land_C96_static_fields.nc`), the initial conditions file (`ufs-land_C96_init_*.nc`), and the model configuration file (`ufs-land.namelist.noahmp`). These files and their parameters are described in the following subsections. They are publicly available via the Land DA Data Bucket. Users can download the data and untar the file via the command line, replacing `{YEAR}` with the year for the desired data. Release data is currently available for 2016 and 2020:

```
wget https://noaa-ufs-land-da-pds.s3.amazonaws.com/current_land_da_release_data/landda-
↪input-data-{YEAR}.tar.gz
tar xvfz landda-input-data-{YEAR}.tar.gz
```

5.1.1 Static File (`ufs-land_C96_static_fields.nc`)

The static file includes specific information on location, time, soil layers, and fixed (invariant) experiment parameters that are required for Noah-MP to run. The data must be provided in *netCDF* format.

The static file is available in the `inputs` data directory (downloaded *above*) at the following path:

```
inputs/forcing/<source>/static/ufs-land_C96_static_fields.nc
```

where <source> is either `gdas` or `era5`.

Table 5.1: Configuration variables specified in the static file (ufs-land_C96_static_fields.nc)

Configuration Variables	Description
land_mask	land-sea mask (0-ocean, 1-land)
vegetation_category	vegetation type
soil_category	soil type
slope_category	slope type
albedo_monthly	monthly albedo
lai_monthly (leaf area index_monthly)	monthly leaf area index
emissivity	emissivity
z0_monthly	monthly ground roughness length
cube_tile	FV3 tile where the grid is located
cube_i	i-location in the FV3 tile where the grid is located
cube_j	j-location in the FV3 tile where the grid is located
latitude	latitude
longitude	longitude
elevation	elevation
deep_soil_temperature	lower boundary soil temperature
max_snow_albedo	maximum snow albedo
gvf_monthly	monthly green vegetation fraction (gvf)
visible_black_sky_albedo	visible black sky albedo
visible_white_sky_albedo	visible white sky albedo
near_IR_black_sky_albedo	near infrared black sky albedo
near_IR_white_sky_albedo	near infrared white sky albedo
soil_level_nodes	soil level nodes
soil_level_thickness	soil level thickness

5.1.2 Initial Conditions File (`ufs-land_C96_init_*.nc`)

The offline Land DA System currently only supports snow DA. The initial conditions file includes the initial state variables that are required for the UFS land snow DA to begin a cycling run. The data must be provided in *netCDF* format.

The initial conditions file is available in the `inputs` data directory (downloaded *above*) at the following path:

```
inputs/forcing/GDAS/init/ufs-land_C96_init_fields_1hr.nc
inputs/forcing/ERA5/init/ufs-land_C96_init_2010-12-31_23-00-00.nc
```

Table 5.2: Configuration variables specified in the initial forcing file (ufs-land_C96_init_fields_1hr.nc)

Configuration Variables	Units
time	seconds since 1970-01-01 00:00:00
date (date length)	UTC date
latitude	degrees north-south
longitude	degrees east-west
snow_water_equivalent	mm
snow_depth	m
canopy_water	mm
skin_temperature	K
soil_temperature	mm
soil_moisture	m^3/m^3
soil_liquid	m^3/m^3
soil_level_thickness	m
soil_level_nodes	m

5.1.3 Model Configuration File (`ufs-land.namelist.noahmp`)

The UFS land model uses a series of template files combined with user-selected settings to create required namelists and parameter files needed by the UFS Land DA workflow. This section describes the options in the `ufs-land.namelist.noahmp` file, which is generated from the `template.ufs-noahMP.namelist.*` file.

Note: Any default values indicated are the defaults set in the `template.ufs-noahMP.namelist.*` files.

Run Setup Parameters

`static_file`
Specifies the path to the UFS land static file.

`init_file`
Specifies the path to the UFS land initial condition file.

`forcing_dir`
Specifies the path to the UFS land forcing directory where atmospheric forcing files are located.

`separate_output`
Specifies whether to enable separate output files for each output time. Valid values: `.false.` | `.true.`

Value	Description
.false.	do not enable (should only be used for single point or short simulations)
.true.	enable

`output_dir`
Specifies the output directory where output files will be saved. If `separate_output=.true.`, but no `output_dir` is specified, it will default to the directory where the executable is run.

`restart_frequency_s`
Specifies the restart frequency (in seconds) for the UFS land model.

restart_simulation
Specifies whether to enable the restart simulation. Valid values: `.false.` | `.true.`

Value	Description
.false.	do not enable
.true.	enable

restart_date
Specifies the restart date. The form is `YYYY-MM-DD HH:MM:SS`, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, HH is a valid 2-digit hour, MM is a valid 2-digit minute, and SS is a valid 2-digit second.

restart_dir
Specifies the restart directory.

timestep_seconds
Specifies the land model timestep in seconds.

simulation_start
Specifies the simulation start time. The form is `YYYY-MM-DD HH:MM:SS`, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, HH is a valid 2-digit hour, MM is a valid 2-digit minute, and SS is a valid 2-digit second.

simulation_end
Specifies the simulation end time. The form is `YYYY-MM-DD HH:MM:SS`, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, HH is a valid 2-digit hour, MM is a valid 2-digit minute, and SS is a valid 2-digit second.

run_days
Specifies the number of days to run.

run_hours
Specifies the number of hours to run.

run_minutes
Specifies the number of minutes to run.

run_seconds
Specifies the number of seconds to run.

run_timesteps
Specifies the number of timesteps to run.

Land Model Options

land_model
Specifies which land surface model to use. Valid values: `1` | `2`

Value	Description
1	Noah
2	Noah-MP

Structure-Related Parameters

num_soil_levels
Specifies the number of soil levels.

forcing_height
Specifies the forcing height in meters.

Soil Setup Parameters

soil_level_thickness
Specifies the thickness (in meters) of each of the soil layers (top layer to bottom layer).

soil_level_nodes
Specifies the soil level centroids from the surface (in meters).

Noah-MP Options

dynamic_vegetation_option: (Default: 4)
Specifies the dynamic vegetation model option. Valid values: 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

Value	Description
1	off (use table LAI; use FVEG=SHDFAC from input)
2	on (dynamic vegetation; must use Ball-Berry canopy option)
3	off (use table LAI; calculate FVEG)
4	off (use table LAI; use maximum vegetation fraction)
5	on (use maximum vegetation fraction)
6	on (use FVEG = SHDFAC from input)
7	off (use input LAI; use FVEG = SHDFAC from input)
8	off (use input LAI; calculate FVEG)
9	off (use input LAI; use maximum vegetation fraction)
10	crop model on (use maximum vegetation fraction)

LAI
Routines for handling Leaf/Stem area index data products

FVEG
Green vegetation fraction [0.0-1.0]

SHDFAC
Greenness vegetation (shaded) fraction

canopy_stomatal_resistance_option: (Default: 2)
Specifies the canopy stomatal resistance option. Valid values: 1 | 2

Value	Description
1	Ball-Berry
2	Jarvis

soil_wetness_option: (Default: 1)
Specifies the soil moisture factor for the stomatal resistance option. Valid values: 1 | 2 | 3

Value	Description
1	Noah (soil moisture)
2	CLM (matric potential)
3	SSiB (matric potential)

runoff_option: (Default: 1)
Specifies the runoff option. Valid values: 1 | 2 | 3 | 4 | 5

Value	Description
1	SIMGM: TOPMODEL with groundwater (Niu *et al.* [NYD+07])
2	SIMTOP: TOPMODEL with an equilibrium water table (Niu *et al.* [NYREDG05])
3	Noah original surface and subsurface runoff (free drainage) (Schaake *et al.* [SKD+96])
4	BATS surface and subsurface runoff (free drainage)
5	Miguez-Macho&Fan groundwater scheme (Miguez-Macho *et al.* [MMFW+07]; Fan *et al.* [FMMW+07])

surface_exchange_option: (Default: 3)
Specifies the surface layer drag coefficient option. Valid values: 1 | 2

Value	Description
1	Monin-Obukhov
2	original Noah (Chen 1997)

supercooled_soilwater_option: (Default: 1)
Specifies the supercooled liquid water option. Valid values: 1 | 2

Value	Description
1	no iteration (Niu and Yang [NY06])
2	Koren's iteration (Koren *et al.* [KSM+99])

frozen_soil_adjust_option: (Default: 1)
Specifies the frozen soil permeability option. Valid values: 1 | 2

Value	Description
1	linear effects, more permeable (Niu and Yang [NY06])
2	nonlinear effects, less permeable (Koren *et al.* [KSM+99])

radiative_transfer_option: (Default: 3)
Specifies the radiation transfer option. Valid values: 1 | 2 | 3

Value	Description
1	modified two-stream (gap = F(solar angle, 3D structure…)<1-FVEG)
2	two-stream applied to grid-cell (gap = 0)
3	two-stream applied to a vegetated fraction (gap=1-FVEG)

snow_albedo_option: (Default: 2)
Specifies the snow surface albedo option. Valid values: 1 | 2

Value	Description
1	BATS
2	CLASS

precip_partition_option: (Default: 1)
Specifies the option for partitioning precipitation into rainfall and snowfall. Valid values: 1 | 2 | 3 | 4

Value	Description
1	Jordan [Jor91] (1991)
2	BATS: when SFCTMP<TFRZ+2.2
3	Noah: when SFCTMP<TFRZ
4	Use WRF microphysics output

SFCTMP
Surface air temperature

TFRZ
Freezing/melting point (K)

soil_temp_lower_bdy_option: (Default: 2)
Specifies the lower boundary condition of soil temperature option. Valid values: 1 | 2

Value	Description
1	zero heat flux from the bottom (ZBOT and TBOT not used)
2	TBOT at ZBOT (8m) read from a file (original Noah)

TBOT
Lower boundary soil temperature [K]

ZBOT
Depth[m] of lower boundary soil temperature (TBOT)

soil_temp_time_scheme_option: (Default: 3)
Specifies the snow and soil temperature time scheme. Valid values: 1 | 2 | 3

Value	Description
1	semi-implicit; flux top boundary condition
2	fully implicit (original Noah); temperature top boundary condition
3	same as 1, but FSNO for TS calculation (generally improves snow; v3.7)

FSNO
Fraction of surface covered with snow

TS
Surface temperature

thermal_roughness_scheme_option: (Default: 2)
Specifies the method/scheme used to calculate the thermal roughness length. Valid values: 1 | 2 | 3 | 4

Value	Description
1	z0h=z, thermal roughness length = momentum roughness length
2	czil, use canopy height method based on (Chen and Zhang [CZ09])
3	European Center method
4	kb inverse method

surface_evap_resistance_option: (Default: 1)
Specifies the surface evaporation resistance option. Valid values: 1 | 2 | 3 | 4

Value	Description
1	Sakaguchi and Zeng [SZ09]
2	Sellers *et al.* [SMDHH92]
3	adjusted Sellers to decrease RSURF for wet soil
4	option 1 for non-snow; rsurf = rsurf_snow for snow

rsurf
Ground surface resistance (s/m)

glacier_option: (Default: 1)
Specifies the glacier model option. Valid values: 1 | 2

Value	Description
1	include phase change of ice
2	simple (ice treatment more like original Noah)

Forcing Parameters

forcing_timestep_seconds: (Default: 3600)
Specifies the forcing timestep in seconds.

forcing_type
Specifies the forcing type option, which describes the frequency and length of forcing in each forcing file. Valid values: single-point | gswp3 | gdas

Value	Description
single-point	All forcing times are in one file
gswp3	three-hourly forcing stored in monthly files
gdas	hourly forcing stored in daily files

Note: There is no separate era5 format. It is the same as the gdas format, so users should select gdas for this parameter when using era5 forcing.

---

forcing_filename
Specifies the forcing file name prefix. A date will be appended to this prefix. For example: C96_ERA5_forcing_2020-10-01.nc. The prefix merely indicates which grid (C96) and source (i.e., GDAS, GEFS) will be used. Common values include: C96_GDAS_forcing_ | C96_ERA5_forcing_ | C96_GEFS_forcing_ | C96_GSWP3_forcing_

Value	Description
C96_GDAS_forcing_	GDAS forcing data for a C96 grid
C96_ERA5_forcing_	ERA5 forcing data for a C96 grid
C96_GEFS_forcing_	GEFS forcing data for a C96 grid
C96_GSWP3_forcing_	GSWP3 forcing data for a C96 grid

forcing_interp_solar
Specifies the interpolation option for solar radiation. Valid values: linear | gswp3_zenith

Value	Description
linear	Performs a linear interpolation between forcing times
gswp3_zenith	Performs a cosine zenith angle interpolation between forcing times

forcing_time_solar
Valid values include: "instantaneous" | "gswp3_average"

forcing_name_precipitation
Specifies the variable name of forcing precipitation. Valid values include: "precipitation_conserve" | "precipitation_bilinear"

forcing_name_temperature``(Default: ``"temperature")
Specifies the variable name of forcing temperature.

forcing_name_specific_humidity: (Default: "specific_humidity")
Specifies the variable name of forcing specific-humidity.

forcing_name_wind_speed: (Default: "wind_speed")
Specifies the variable name of forcing wind speed.

forcing_name_pressure: (Default: "surface_pressure")
Specifies the variable name of forcing surface pressure.

forcing_name_sw_radiation: (Default: "solar_radiation")
Specifies the variable name of forcing shortwave radiation.

forcing_name_lw_radiation: (Default: "longwave_radiation")
Specifies the variable name of forcing longwave radiation.

Example Namelist Entry

The `ufs-land.namelist.noahmp` file should be similar to the following example, which comes from the `template.ufs-noahMP.namelist.gdas` file.

```
&run_setup

   static_file      = "/LANDDA_INPUTS/forcing/gdas/static/ufs-land_C96_static_fields.nc"
   init_file        = "/LANDDA_INPUTS/forcing/gdas/init/ufs-land_C96_init_fields_1hr.nc"
   forcing_dir      = "/LANDDA_INPUTS/forcing/gdas/gdas/forcing"

   separate_output = .false.
   output_dir       = "./"

   restart_frequency_s = XXFREQ
   restart_simulation  = .true.
   restart_date        = "XXYYYY-XXMM-XXDD XXHH:00:00"
   restart_dir         = "./"

   timestep_seconds = 3600

! simulation_start is required
! either set simulation_end or run_* or run_timesteps, priority
!   1. simulation_end 2. run_[days/hours/minutes/seconds] 3. run_timesteps

   simulation_start = "2000-01-01 00:00:00"   ! start date [yyyy-mm-dd hh:mm:ss]
```

(continues on next page)

(continued from previous page)

```
   ! simulation_end   = "1999-01-01 06:00:00"   !   end date [yyyy-mm-dd hh:mm:ss]

   run_days         = XXRDD   ! number of days to run
   run_hours        = XXRHH   ! number of hours to run
   run_minutes      = 0       ! number of minutes to run
   run_seconds      = 0       ! number of seconds to run

   run_timesteps    = 0       ! number of timesteps to run

   location_start   = 1
   location_end     = 18360

/

&land_model_option
   land_model        = 2   ! choose land model: 1=noah, 2=noahmp
/

&structure
   num_soil_levels   = 4     ! number of soil levels
   forcing_height    = 6     ! forcing height [m]
/

&soil_setup
   soil_level_thickness   =  0.10,    0.30,    0.60,    1.00      ! soil level␣
↪thicknesses [m]
   soil_level_nodes       =  0.05,    0.25,    0.70,    1.50      ! soil level centroids␣
↪from surface [m]
/

&noahmp_options
   dynamic_vegetation_option         = 4
   canopy_stomatal_resistance_option = 2
   soil_wetness_option               = 1
   runoff_option                     = 1
   surface_exchange_option           = 3
   supercooled_soilwater_option      = 1
   frozen_soil_adjust_option         = 1
   radiative_transfer_option         = 3
   snow_albedo_option                = 2
   precip_partition_option           = 1
   soil_temp_lower_bdy_option        = 2
   soil_temp_time_scheme_option      = 3
   thermal_roughness_scheme_option   = 2
   surface_evap_resistance_option    = 1
   glacier_option                    = 1
/

&forcing
   forcing_timestep_seconds       = 3600
   forcing_type                   = "gdas"
   forcing_filename               = "C96_GDAS_forcing_"
```

(continues on next page)

(continued from previous page)

```
   forcing_interp_solar           = "linear"  ! gswp3_zenith or linear
   forcing_time_solar             = "instantaneous"  ! gswp3_average or instantaneous
   forcing_name_precipitation     = "precipitation_conserve"
   forcing_name_temperature       = "temperature"
   forcing_name_specific_humidity = "specific_humidity"
   forcing_name_wind_speed        = "wind_speed"
   forcing_name_pressure          = "surface_pressure"
   forcing_name_sw_radiation      = "solar_radiation"
   forcing_name_lw_radiation      = "longwave_radiation"
/
```

5.2 Vector-to-Tile Converter

The Vector-to-Tile Converter is used for mapping between the vector format used by the Noah-MP offline driver and the tile format used by the UFS atmospheric model. This converter is currently used to prepare input tile files for JEDI. Note that these files include only those fields required by JEDI, rather than the full restart.

5.2.1 Input File

The input files containing grid information are listed in Table 5.3:

Table 5.3: Input Files Containing Grid Information

Filename	Description
Cxx_grid.tile[1-6].nc	Cxx grid information for tiles 1-6, where `xx` is the grid resolution.
Cxx_oro_data.tile[1-6].nc / oro_Cxx.mx100.tile[1-6].nc	Orography files that contain grid and land mask information. Cxx refers to the atmospheric resolution, and mx100 refers to the ocean resolution (100=1º). Both file names refer to the same file; there are symbolic links between them.

5.2.2 Configuration File

This section describes the options in the `namelist.vector2tile` file.

Run Setup Parameters

`direction`
Specifies the conversion option. Valid values: `vector2tile` | `tile2vector` | `lndp2tile` | `lndp2vector`

Value	Description
vector2tile	vector-to-tile conversion for restart file
tile2vector	tile-to-vector conversion for restart file
lndp2tile	land perturbation to tile
lndp2vector	land perturbation to vector

FV3 Tile-Related Parameters for Restart/Perturbation Conversion

Parameters in this section include the FV3 resolution and path to orographic files for restart/perturbation conversion.

tile_size
Specifies the size (horizontal resolution) of the FV3 tile. Valid values: `96`.

Note:

- The `C96` grid files correspond to approximately 1º latitude/longitude.
- Additional resolutions (e.g., `192`, `384`, `768`) are under development.

---

tile_path
Specifies the path to the orographic tile files.

tile_fstub
Specifies the name (file stub) of orographic tile files. The file stub will be named `oro_C${RES}` for atmosphere-only and `oro_C{RES}.mx100` for atmosphere and ocean.

Parameters for Restart Conversion

These parameters apply *only* to restart conversion.

static_filename
Specifies the path for static file.

vector_restart_path
Specifies the location of vector restart file, vector-to-tile direction.

tile_restart_path
Specifies the location of tile restart file, tile-to-vector direction.

output_path
Specifies the path for converted files. If this is same as tile/vector path, the files may be overwritten.

Perturbation Mapping Parameters

These parameters are *only* relevant for perturbation mapping in ensembles. Support for ensembles is *not* provided for the Land DA v1.0.0 release.

lndp_layout
Specifies the layout options. Valid values: `1x4` | `4x1` | `2x2`

lndp_input_file
Specifies the path for the input file.

output files
Specifies the path for the output file.

lndp_var_list
Specifies the land perturbation variable options. Valid values: `vgf` | `smc`

Value	Description
vgf	Perturbs the vegetation green fraction
smc	Perturbs the soil moisture

Example of a `namelist.vector2tile` Entry

```
&run_setup

direction = "vector2tile"

&FV3 resolution and path to oro files for restart/perturbation
conversion

tile_size = 96
tile_path ="/ /"
tile_fstub = "oro_C96.mx100"

!------------------- only restart conversion -------------------

! Time stamp for conversion for restart conversion
restart_date = "2019-09-30 23:00:00"

! Path for static file
static_filename="/*/filename.nc "

! Location of vector restart file (vector2tile direction)
vector_restart_path ="/ /"

! Location of tile restart files (tile2vector direction)
tile_restart_path ="/ /"

output_path ="/ /"

!------------------- only perturbation mapping -------------------
lndp_layout = "1x4"

! input files
lndp_input_file ="/*/filename.nc "

! output files
lndp_output_file = "./output.nc"

! land perturbation variable list
lndp_var_list='vgf','smc'
```

CHAPTER

SIX

LAND DATA ASSIMILATION SYSTEM

This chapter describes the configuration of the offline Land *Data Assimilation* (DA) System, which utilizes the UFS Noah-MP components with the JEDI `fv3-bundle` to enable cycled model forecasts. The data assimilation framework applies the Local Ensemble Transform Kalman Filter-Optimal Interpolation (LETKF-OI) algorithm to combine the state-dependent background error derived from an ensemble forecast with the observations and their corresponding uncertainties to produce an analysis ensemble (Hunt *et al.* [HEJKS07], 2007).

6.1 Joint Effort for Data Assimilation Integration (JEDI)

The Joint Effort for Data assimilation Integration (*JEDI*) is a unified and versatile *data assimilation* (DA) system for Earth System Prediction that can be run on a variety of platforms. JEDI is developed by the Joint Center for Satellite Data Assimilation (JCSDA) and partner agencies, including NOAA. The core feature of JEDI is separation of concerns. The data assimilation update, observation selection and processing, and observation operators are all coded with no knowledge of or dependency on each other or on the forecast model.

The NOAH-MP offline Land DA System uses three JEDI components:

- The Object-Oriented Prediction System (OOPS) for the data assimilation algorithm
- The Interface for Observation Data Access (IODA) for the observation formatting and processing
- The Unified Forward Operator (UFO) for comparing model forecasts and observations

JEDI's Unified Forward Operator (UFO) links observation operators with the Object Oriented Prediction System (OOPS) to compute a simulated observation given a known model state. It does not restrict observation operators based on model-specific code structures or requirements. The UFO code structure provides generic classes for observation bias correction and quality control. Within this system, IODA converts the observation data into model-specific formats to be ingested by each model's data assimilation system. This involves model-specific data conversion efforts.

6.1.1 Object-Oriented Prediction System (OOPS)

A data assimilation experiment requires a `yaml` configuration file that specifies the details of the data assimilation and observation processing. OOPS provides the core set of data assimilation algorithms in JEDI by combining the generic building blocks required for the algorithms. The OOPS system does not require knowledge of any specific application model implementation structure or observation data information. In the Noah-MP offline Land DA System, OOPS reads the model forecast states from the restart files generated by the Noah-MP model. JEDI UFO contains generic quality control options and filters that can be applied to each observation system, without coding at certain model application levels. More information on the key concepts of the JEDI software design can be found in Trémolet and Auligné [TA20] (2020), Holdaway *et al.* [HVMWK20] (2020), and Honeyager *et al.* [HHZ+20] (2020).

JEDI Configuration Files & Parameters

To create the DA experiment, the user should create or modify an experiment-specific configuration `yaml` file. This `yaml` file should contain certain fundamental components: geometry, window begin, window length, background, driver, local ensemble DA, output increment, and observations. These components can be implemented differently for different models and observation types, so they frequently contain distinct parameters and variable names depending on the use case. Therefore, this section of the User's Guide focuses on assisting users with understanding and customizing these top-level configuration items in order to run Land DA experiments. Users may also reference the JEDI Documentation for additional information.

Users may find the following example `yaml` configuration file to be a helpful starting point. This file (with user-appropriate modifications) is required by JEDI for snow data assimilation. The following subsections will explain the variables within each top-level item of the `yaml` file.

```
geometry:
  fms initialization:
    namelist filename: Data/fv3files/fmsmpp.nml
    field table filename: Data/fv3files/field_table
  akbk: Data/fv3files/akbk127.nc4
  npx: 97
  npy: 97
  npz: 127
  field metadata override: Data/fieldmetadata/gfs-land.yaml
  time invariant fields:
    state fields:
      datetime: 2016-01-02T18:00:00Z
      filetype: fms restart
      skip coupler file: true
      state variables: [orog_filt]
      datapath: /mnt/lfs4/HFIP/hfv3gfs/role.epic/landda/inputs/forcing/gdas/orog_files
      filename_orog: oro_C96.mx100.nc

window begin: 2016-01-02T12:00:00Z
window length: PT6H

background:
  date: &date 2016-01-02T18:00:00Z
  members:
    - datetime: 2016-01-02T18:00:00Z
      filetype: fms restart
      state variables: [snwdph,vtype,slmsk]
      datapath: mem_pos/
      filename_sfcd: 20160102.180000.sfc_data.nc
      filename_cplr: 20160102.180000.coupler.res
    - datetime: 2016-01-02T18:00:00Z
      filetype: fms restart
      state variables: [snwdph,vtype,slmsk]
      datapath: mem_neg/
      filename_sfcd: 20160102.180000.sfc_data.nc
      filename_cplr: 20160102.180000.coupler.res

driver:
  save posterior mean: false
  save posterior mean increment: true
```

(continues on next page)

(continued from previous page)

```
  save posterior ensemble: false
  run as observer only: false

local ensemble DA:
  solver: LETKF
  inflation:
    rtps: 0.0
    rtpp: 0.0
    mult: 1.0

output increment:
  filetype: fms restart
  filename_sfcd: xainc.sfc_data.nc

observations:
  observers:
  - obs space:
      name: Simulate
      distribution:
        name: Halo
        halo size: 250e3
      simulated variables: [totalSnowDepth]
      obsdatain:
        engine:
          type: H5File
          obsfile: GHCN_2016010218.nc
      obsdataout:
        engine:
          type: H5File
          obsfile: output/DA/hofx/letkf_hofx_ghcn_2016010218.nc
    obs operator:
      name: Identity
    obs error:
      covariance model: diagonal
    obs localizations:
    - localization method: Horizontal SOAR
      lengthscale: 250e3
      soar horizontal decay: 0.000021
      max nobs: 50
    - localization method: Vertical Brasnett
      vertical lengthscale: 700
    obs filters:
    - filter: Bounds Check # negative / missing snow
      filter variables:
      - name: totalSnowDepth
      minvalue: 0.0
    - filter: Domain Check # missing station elevation (-999.9)
      where:
      - variable:
          name: height@MetaData
        minvalue: -999.0
    - filter: Domain Check # land only
```

(continues on next page)

(continued from previous page)

```
    where:
    - variable:
        name: slmsk@GeoVaLs
      minvalue: 0.5
      maxvalue: 1.5
  # GFSv17 only.
  #- filter: Domain Check # no sea ice
  #  where:
  #  - variable:
  #      name: fraction_of_ice@GeoVaLs
  #    maxvalue: 0.0
  - filter: RejectList  # no land-ice
    where:
    - variable:
        name: vtype@GeoVaLs
      minvalue: 14.5
      maxvalue: 15.5
  - filter: Background Check # gross error check
    filter variables:
    - name: totalSnowDepth
    threshold: 6.25
    action:
      name: reject
```

Note: Any default values indicated in the sections below are the defaults set in `letkfoi_snow.yaml` or `GHCN.yaml` (found within the `land-offline_workflow/DA_update/jedi/fv3-jedi/yaml_files/release-v1.0/` directory).

Geometry

The `geometry:` section is used in JEDI configuration files to specify the model grid's parallelization across compute nodes (horizontal and vertical).

`fms initialization`
: This section contains two parameters, `namelist filename` and `field table filename`.

  `namelist filename`
  : Specifies the path for the namelist filename.

  `field table filename`
  : Specifies the path for the field table filename.

`akbk`
: Specifies the path to a file containing the coefficients that define the hybrid sigma-pressure vertical coordinate used in FV3. Files are provided with the repository containing `ak` and `bk` for some common choices of vertical resolution for GEOS and GFS.

`npx`
: Specifies the number of grid cells in the east-west direction.

`npy`
: Specifies the number of grid cells in the north-south direction.

npz
Specifies the number of vertical layers.

field metadata override
Specifies the path for file metadata.

time invariant state fields
This parameter contains several subparameters listed below.

datetime
Specifies the time in YYYY-MM-DDTHH:00:00Z format, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour.

filetype
Specifies the type of file. Valid values include: `fms restart`

skip coupler file
Specifies whether to enable skipping coupler file. Valid values are: `true` | `false`

Value	Description
true	enable
false	do not enable

state variables
Specifies the list of state variables. Valid values include: `[orog_filt]`

datapath
Specifies the path for state variables data.

filename_orog
Specifies the name of orographic data file.

Window begin, Window length

These two items define the assimilation window for many applications, including Land DA.

window begin:
Specifies the beginning time window. The format is YYYY-MM-DDTHH:00:00Z, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour.

window length:
Specifies the time window length. The form is PTXXH, where XX is a 1- or 2-digit hour. For example: `PT6H`

Background

The `background:` section includes information on the analysis file(s) (also known as "members") generated by the previous cycle.

date
Specifies the background date. The format is `&date YYYY-MM-DDTHH:00:00Z`, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour. For example: `&date 2016-01-02T18:00:00Z`

members
Specifies information on analysis file(s) generated by a previous cycle.

datetime
Specifies the date and time. The format is YYYY-MM-DDTHH:00:00Z, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour.

filetype
Specifies the type of file. Valid values include: `fms restart`

state variables
Specifies a list of state variables. Valid values: `[snwdph,vtype,slmsk]`

datapath
Specifies the path for state variables data. Valid values: `mem_pos/` | `mem_neg/`. (With default experiment values, the full path will be `workdir/mem000/jedi/$datapath`.)

filename_sfcd
Specifies the name of the surface data file. This usually takes the form `YYYYMMDD.HHmmss.sfc_data.nc`, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour, mm is a valid 2-digit minute and ss is a valid 2-digit second. For example: `20160102.180000.sfc_data.nc`

filename_cprl
Specifies the name of file that contains metadata for the restart. This usually takes the form `YYYYMMDD.HHmmss.coupler.res`, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour, mm is a valid 2-digit minute and ss is a valid 2-digit second. For example: `20160102.180000.coupler.res`

Driver

The `driver:` section describes optional modifications to the behavior of the LocalEnsembleDA driver. For details, refer to Local Ensemble Data Assimilation in OOPS in the JEDI Documentation.

save posterior mean
Specifies whether to save the posterior mean. Valid values: `true` | `false`

Value	Description
true	save
false	do not save

save posterior mean increment
Specifies whether to save the posterior mean increment. Valid values: `true` | `false`

Value	Description
true	enable
false	do not enable

save posterior ensemble
Specifies whether to save the posterior ensemble. Valid values: `true` | `false`

Value	Description
true	enable
false	do not enable

run as observer only
Specifies whether to run as observer only. Valid values: `true` | `false`

Value	Description
true	enable
false	do not enable

Local Ensemble DA

The `local ensemble DA:` section configures the local ensemble DA solver package.

solver
Specifies the type of solver. Currently, `LETKF` is the only available option. See Hunt *et al.* [HEJKS07] (2007).

inflation
Describes ensemble inflation methods.

rtps: (Default: `0.0`)
Relaxation to prior spread (Whitaker and Hamill [WH12], 2012).

rtpp: (Default: `0.0`)
Relaxation to prior perturbation (Zhang *et al.* [ZCSS04], 2004).

mult: (Default: `1.0`)
Parameter of multiplicative inflation.

Output Increment

output increment:

filetype
Type of file provided for the output increment. Valid values include: `fms restart`

filename_sfcd
Name of the file provided for the output increment. For example: `xainc.sfc_data.nc`

Observations

The `observations:` item describes one or more types of observations, each of which is a multi-level YAML/JSON object in and of itself. Each of these observation types is read into JEDI as an `eckit::Configuration` object (see JEDI Documentation for more details).

obs space:

The `obs space:` section of the `yaml` comes under the `observations.observers:` section and describes the configuration of the observation space. An observation space handles observation data for a single observation type.

name
Specifies the name of observation space. The Land DA System uses `Simulate` for the default case.

distribution:

name
Specifies the name of distribution. Valid values include: `Halo`

halo size
Specifies the size of the halo distribution. Format is e-notation. For example: `250e3`

simulated variables
Specifies the list of variables that need to be simulated by observation operator. Valid values: `[totalSnowDepth]`

obsdatain
This section specifies information about the observation input data.

engine
This section specifies parameters required for the file matching engine.

type
Specifies the type of input observation data. Valid values: `H5File | OBS`

obsfile
Specifies the input filename.

obsdataout
This section contains information about the observation output data.

engine
This section specifies parameters required for the file matching engine.

type
Specifies the type of output observation data. Valid values: `H5File`

obsfile
Specifies the output file path.

obs operator:

The `obs operator:` section describes the observation operator and its options. An observation operator is used for computing H(x).

name
Specifies the name in the `ObsOperator` and `LinearObsOperator` factory, defined in the C++ code. Valid values include: `Identity`. See JEDI Documentation for more options.

obs error:

The `obs error:` section explains how to calculate the observation error covariance matrix and gives instructions (required for DA applications). The key covariance model, which describes how observation error covariances are created, is frequently the first item in this section. For diagonal observation error covariances, only the diagonal option is currently supported.

covariance model
Specifies the covariance model. Valid values include: `diagonal`

`obs localizations:`

`obs localizations:`

`localization method`
Specifies the observation localization method. Valid values include: `Horizontal SOAR`

Value	Description
Horizontal SOAR	Second Order Auto-Regressive localization in the horizontal direction.
Vertical Brasnett	Vertical component of the localization scheme defined in Brasnett [Bra99] (1999) and used in the snow DA.

`lengthscale`
Radius of influence (i.e., maximum distance of observations from the location being updated) in meters. Format is e-notation. For example: `250e3`

`soar horizontal decay`
Decay scale of SOAR localization function. Recommended value: `0.000021`. Users may adjust based on need/preference.

`max nobs`
Maximum number of observations used to update each location.

`obs filters:`

Observation filters are used to define Quality Control (QC) filters. They have access to observation values and metadata, model values at observation locations, simulated observation value, and their own private data. See Observation Filters in the JEDI Documentation for more detail. The `obs filters:` section contains the following fields:

`filter`
Describes the parameters of a given QC filter. Valid values include: `Bounds Check` | `Background Check` | `Domain Check` | `RejectList`. See descriptions in the JEDI's Generic QC Filters Documentation for more.

Filter Name	Description
Bounds Check	Rejects observations whose values lie outside specified limits:
Background Check	This filter checks for bias-corrected distance between the observation value and model-simulated value (*y - H(x)*) and rejects observations where the absolute difference is larger than the `absolute threshold` or `threshold` * *observation error* or `threshold` * *background error*.
Domain Check	This filter retains all observations selected by the `where` statement and rejects all others.
RejectList	This is an alternative name for the BlackList filter, which rejects all observations selected by the `where` statement. The status of all others remains the same. Opposite of Domain Check filter.

`filter variables`
Limit the action of a QC filter to a subset of variables or to specific channels.

name
Name of the filter variable. Users may indicate additional filter variables using the `name` field on consecutive lines (see code snippet below). Valid values include: `totalSnowDepth`

```
filter variables:
- name: variable_1
- name: variable_2
```

minvalue
Minimum value for variables in the filter.

maxvalue
Maximum value for variables in the filter.

threshold
This variable may function differently depending on the filter it is used in. In the Background Check Filter, an observation is rejected when the difference between the observation value (y) and model simulated value ($H(x)$) is larger than the `threshold` * *observation error*.

action
Indicates which action to take once an observation has been flagged by a filter. See Filter Actions in the JEDI documentation for a full explanation and list of valid values.

name
The name of the desired action. Valid values include: `accept` | `reject`

where
By default, filters are applied to all filter variables listed. The `where` keyword applies a filter only to observations meeting certain conditions. See the Where Statement section of the JEDI Documentation for a complete description of valid `where` conditions.

variable
A list of variables to check using the `where` statement.

name
Name of a variable to check using the `where` statement. Multiple variable names may be listed under `variable`. The conditions in the where statement will be applied to all of them. For example:

```
filter: Domain Check # land only
  where:
  - variable:
      name: variable_1
      name: variable_2
    minvalue: 0.5
    maxvalue: 1.5
```

minvalue
Minimum value for variables in the `where` statement.

maxvalue
Maximum value for variables in the `where` statement.

6.1.2 Interface for Observation Data Access (IODA)

This section references Honeyager, R., Herbener, S., Zhang, X., Shlyaeva, A., and Trémolet, Y., 2020: Observations in the Joint Effort for Data assimilation Integration (JEDI) - UFO and IODA. JCSDA Quarterly, 66, Winter 2020.

The Interface for Observation Data Access (IODA) is a subsystem of JEDI that can handle data processing for various models, including the Land DA System. Currently, observation data sets come in a variety of formats (e.g., netCDF, BUFR, GRIB) and may differ significantly in structure, quality, and spatiotemporal resolution/density. Such data must be pre-processed and converted into model-specific formats. This process often involves iterative, model-specific data conversion efforts and numerous cumbersome ad-hoc approaches to prepare observations. Requirements for observation files and I/O handling often result in decreased I/O and computational efficiency. IODA addresses this need to modernize observation data management and use in conjunction with the various components of the Unified Forecast System (*UFS*).

IODA provides a unified, model-agnostic method of sharing observation data and exchanging modeling and data assimilation results. The IODA effort centers on three core facets: (i) in-memory data access, (ii) definition of the IODA file format, and (iii) data store creation for long-term storage of observation data and diagnostics. The combination of these foci enables optimal isolation of the scientific code from the underlying data structures and data processing software while simultaneously promoting efficient I/O during the forecasting/DA process by providing a common file format and structured data storage.

The IODA file format represents observational field variables (e.g., temperature, salinity, humidity) and locations in two-dimensional tables, where the variables are represented by columns and the locations by rows. Metadata tables are associated with each axis of these data tables, and the location metadata hold the values describing each location (e.g., latitude, longitude). Actual data values are contained in a third dimension of the IODA data table; for instance: observation values, observation error, quality control flags, and simulated observation (H(x)) values.

Since the raw observational data come in various formats, a diverse set of "IODA converters" exists to transform the raw observation data files into IODA format. While many of these Python-based IODA converters have been developed to handle marine-based observations, users can utilize the "IODA converter engine" components to develop and implement their own IODA converters to prepare arbitrary observation types for data assimilation within JEDI. (See https://github.com/NOAA-PSL/land-DA_update/blob/develop/jedi/ioda/imsfv3_scf2ioda_obs40.py for the land DA IMS IODA converter.)

6.2 Input Files

The Land DA System requires grid description files, observation files, and restart files to perform snow DA.

6.2.1 Grid Description Files

The grid description files appear in Section 5.2.1 and are also used as input files to the Vector-to-Tile Converter. See Table 5.3 for a description of these files.

6.2.2 Observation Data

Observation data from 2016 and 2020 are provided in NetCDF format for the v1.0.0 release. Instructions for downloading the data are provided in Section 4.3, and instructions for accessing the data on *Level 1 Systems* are provided in Section 3.2. Currently, data is taken from the Global Historical Climatology Network (GHCN), but eventually, data from the U.S. National Ice Center (USNIC) Interactive Multisensor Snow and Ice Mapping System (IMS) will also be available for use.

Observation Types

GHCN Snow Depth Files

Snow depth observations are taken from the Global Historical Climatology Network, which provides daily climate summaries sourced from a global network of 100,000 stations. NOAA's NCEI provides access to these snow depth and snowfall measurements through daily-generated individual station ASCII files or GZipped tar files of full-network observations on the NCEI server or Climate Data Online. Alternatively, users may acquire yearly tarballs via `wget`:

```
wget https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/{YYYY}.csv.gz
```

where `${YYYY}` should be replaced with the year of interest. Note that these yearly tarballs contain all measurement types from the daily GHCN output, and thus, snow depth must be manually extracted from this broader data set.

These raw snow depth observations need to be converted into IODA-formatted netCDF files for ingestion into the JEDI LETKF system. However, this process was preemptively handled outside of the Land DA workflow, and the initial GHCN IODA files for 2016 and 2020 were provided by NOAA PSL (Clara Draper).

The IODA-formatted GHCN files are structured as follows (using 20160102 as an example):

```
netcdf ghcn_snwd_ioda_20160102 {
dimensions:
   nlocs = UNLIMITED ; // (9946 currently)
variables:
   int nlocs(nlocs) ;
                nlocs:suggested_chunk_dim = 9946LL ;

// global attributes:
      string :_ioda_layout = "ObsGroup" ;
      :_ioda_layout_version = 0 ;
      string :converter = "ghcn_snod2ioda_newV2.py" ;
      string :date_time_string = "2016-01-02T18:00:00Z" ;
      :nlocs = 9946 ;

group: MetaData {
   variables:
      string datetime(nlocs) ;
         string datetime:_FillValue = "" ;
      float height(nlocs) ;
         height:_FillValue = 9.96921e+36f ;
```

(continues on next page)

(continued from previous page)

```
         string height:units = "m" ;
      float latitude(nlocs) ;
         latitude:_FillValue = 9.96921e+36f ;
         string latitude:units = "degrees_north" ;
      float longitude(nlocs) ;
         longitude:_FillValue = 9.96921e+36f ;
         string longitude:units = "degrees_east" ;
      string stationIdentification(nlocs) ;
         string stationIdentification:_FillValue = "" ;
   } // group MetaData

group: ObsError {
   variables:
      float totalSnowDepth(nlocs) ;
         totalSnowDepth:_FillValue = 9.96921e+36f ;
         string totalSnowDepth:coordinates = "longitude latitude" ;
         string totalSnowDepth:units = "mm" ;
   } // group ObsError

group: ObsValue {
   variables:
      float totalSnowDepth(nlocs) ;
         totalSnowDepth:_FillValue = 9.96921e+36f ;
         string totalSnowDepth:coordinates = "longitude latitude" ;
         string totalSnowDepth:units = "mm" ;
   } // group ObsValue

group: PreQC {
   variables:
      int totalSnowDepth(nlocs) ;
         totalSnowDepth:_FillValue = -2147483647 ;
         string totalSnowDepth:coordinates = "longitude latitude" ;
   } // group PreQC
}
```

The primary observation variable is `totalSnowDepth`, which, along with the metadata fields of `datetime`, `latitude`, `longitude`, and `height` is defined along the `nlocs` dimension. Also present are `ObsError` and `PreQC` values corresponding to each `totalSnowDepth` measurement on `nlocs`. These values were attributed during the IODA conversion step (not supported for this release). The magnitude of `nlocs` varies between files; this is due to the fact that the number of stations reporting snow depth observations for a given day can vary in the GHCN.

Observation Location and Processing

GHCN

GHCN files for 2016 and 2020 are already provided in IODA format for the v1.0.0 release. Table 3.2 indicates where users can find data on NOAA *RDHPCS* platforms. Tar files containing the 2016 and 2020 data are located in the publicly-available Land DA Data Bucket. Once untarred, the snow depth files are located in `/inputs/DA/snow_depth/GHCN/data_proc/{YEAR}`. These GHCN IODA files were provided by Clara Draper (NOAA PSL). Each file follows the naming convention of `ghcn_snwd_ioda_${YYYY}${MM}${DD}.nc`, where `${YYYY}` is the four-digit cycle year, `${MM}` is the two-digit cycle month, and `${DD}` is the two-digit cycle day.

In each experiment, the DA_config file sets the name of the experiment configuration file. This configuration file is typically named settings_DA_test. Before assimilation, if "GHCN" was specified as the observation type in the DA_config file, the ghcn_snwd_ioda_${YYYY}${MM}${DD}.nc file corresponding to the specified cycle date is soft-linked to the JEDI working directory (${JEDIWORKDIR}) with a naming-convention change (i.e., GHCN_${YYYY}${MM}${DD}${HH}.nc). Here, the GHCN IODA file is appended with the cycle hour, ${HH} which is extracted from the ${STARTDATE} variable defined in the relevant DA_config file.

Prior to ingesting the GHCN IODA files via the LETKF at the DA analysis time, the observations are further quality controlled and checked using letkf_land.yaml (itself a concatenation of GHCN.yaml and letkfoi_snow.yaml; see the GitHub yaml files for more detail). The GHCN-specific observation filters, domain checks, and quality control parameters from GHCN.yaml ensure that only snow depth observations which meet specific criteria are assimilated (the rest are rejected). The contents of this YAML are listed below:

```
- obs space:
    name: Simulate
    distribution:
      name: Halo
      halo size: 250e3
    simulated variables: [totalSnowDepth]
    obsdatain:
      engine:
        type: H5File
        obsfile: GHCN_XXYYYYXXMMXXDDXXHH.nc
    obsdataout:
      engine:
        type: H5File
        obsfile: output/DA/hofx/letkf_hofx_ghcn_XXYYYYXXMMXXDDXXHH.nc
  obs operator:
    name: Identity
  obs error:
    covariance model: diagonal
  obs localizations:
  - localization method: Horizontal SOAR
    lengthscale: 250e3
    soar horizontal decay: 0.000021
    max nobs: 50
  - localization method: Vertical Brasnett
    vertical lengthscale: 700
  obs filters:
  - filter: Bounds Check # negative / missing snow
    filter variables:
    - name: totalSnowDepth
    minvalue: 0.0
  - filter: Domain Check # missing station elevation (-999.9)
    where:
    - variable:
        name: height@MetaData
      minvalue: -999.0
  - filter: Domain Check # land only
    where:
    - variable:
        name: slmsk@GeoVaLs
      minvalue: 0.5
      maxvalue: 1.5
```

(continues on next page)

(continued from previous page)

```
  # GFSv17 only.
  #- filter: Domain Check # no sea ice
  #  where:
  #  - variable:
  #      name: fraction_of_ice@GeoVaLs
  #    maxvalue: 0.0
  - filter: RejectList  # no land-ice
    where:
    - variable:
        name: vtype@GeoVaLs
      minvalue: 14.5
      maxvalue: 15.5
  - filter: Background Check # gross error check
    filter variables:
    - name: totalSnowDepth
    threshold: 6.25
    action:
      name: reject
```

Viewing NetCDF Files

Users can view file information and notes for NetCDF files using the `ncdump` module. First, load a compiler, MPI, and NetCDF modules:

```
# To see available modules:
module avail
# To load modules:
module load intel/2022.2.0 impi/2022.2.0 netcdf/4.7.0
```

Users may need to modify the module load command to reflect modules that are available on their system.

Then, run `ncdump -h path/to/file`. For example, on Hera, users can run:

```
ncdump -h /scratch1/NCEPDEV/nems/role.epic/landda/inputs/DA/snow_depth/GHCN/data_proc/
↪2016/ghcn_snwd_ioda_20160102.nc
```

to see the contents of the 2016-01-02 GHCN file.

6.2.3 Restart Files

To restart the `ufs-land-driver` successfully after land model execution, all parameters, states, and fluxes used for a subsequent time iteration are stored in a restart file. This restart file is named `ufs_land_restart.{FILEDATE}.nc` where `FILEDATE` is in YYYY-MM-DD_HH-mm-SS format (e.g., `ufs_land_restart.2016-01-02_18-00-00.nc`). The restart file contains all the model fields and their values at a specific point in time; this information can be used to restart the model immediately to run the next cycle. The Land DA System reads the states from the restart file and replaces them after the DA step with the updated analysis. Table 6.1 lists the fields in the Land DA restart file. Within the `ufs-land-driver`, read/write of the restart file is performed in `ufsLandNoahMPRestartModule.f90`.

Table 6.1: Files Included in ufs_land_restart.{FILEDATE}.nc

Variable	Long name	Unit
time	time	"seconds since 1970-01-01 00:00:00"
timestep	time step	"seconds"
vegetation_fraction	Vegetation fraction	"-"
emissivity_total	surface emissivity	"-"
albedo_direct_vis	surface albedo - direct visible	"-"
albedo_direct_nir	surface albedo - direct NIR	"-"
albedo_diffuse_vis	surface albedo - diffuse visible	"-"
albedo_diffuse_nir	surface albedo - diffuse NIR	"-"
temperature_soil_bot	deep soil temperature	"K"
cm_noahmp	surface exchange coefficient for momentum	"m/s"
ch_noahmp	surface exchange coefficient heat & moisture	"m/s"
forcing_height	height of forcing	"m"
max_vegetation_frac	maximum fractional coverage of vegetation	"fraction"
albedo_total	grid composite albedo	"fraction"
snow_water_equiv	snow water equivalent	"mm"
snow_depth	snow depth	"m"
temperature_radiative	surface radiative temperature	"K"
soil_moisture_vol	volumetric moisture content in soil level	"m3/m3"
temperature_soil	temperature in soil level	"K"
soil_liquid_vol	volumetric liquid content in soil level	"m3/m3"
canopy_water	canopy moisture content	"m"
transpiration_heat	plant transpiration	"W/m2"
friction_velocity	friction velocity	"m/s"
z0_total	surface roughness	"m"
snow_cover_fraction	snow cover fraction	"fraction"
spec_humidity_surface	diagnostic specific humidity at surface	"kg/kg"
ground_heat_total	soil heat flux	"W/m2"
runoff_baseflow	drainage runoff	"mm/s"
latent_heat_total	latent heat flux	"W/m2"
sensible_heat_flux	sensible heat flux	"W/m2"
evaporation_potential	potential evaporation	"mm/s"
runoff_surface	surface runoff	"mm/s"
latent_heat_ground	direct soil latent heat flux	"W/m2"
latent_heat_canopy	canopy water latent heat flux	"W/m2"
snow_sublimation	sublimation/deposit from snowpack	"mm/s"
soil_moisture_total	total soil column moisture content	"mm"
precip_adv_heat_total	precipitation advected heat - total	"W/m2"
cosine_zenith	cosine of zenith angle	"-"
snow_levels	active snow levels	"-"
temperature_leaf	leaf temperature	"K"
temperature_ground	ground temperature	"K"
canopy_ice	canopy ice	"mm"
canopy_liquid	canopy liquid	"mm"
vapor_pres_canopy_air	water vapor pressure in canopy air space	"Pa"
temperature_canopy_air	temperature in canopy air space	"K"
canopy_wet_fraction	fraction of canopy covered by water	"-"
snow_water_equiv_old	snow water equivalent - before integration	"mm"
snow_albedo_old	snow albedo - before integration	"-"
snowfall	snowfall	"mm/s"

continues on next page

Table 6.1 – continued from previous page

Variable	Long name	Unit
lake_water	depth of water in lake	"mm"
depth_water_table	depth to water table	"m"
aquifer_water	aquifer water content	"mm"
saturated_water	aquifer + saturated soil water content	"mm"
leaf_carbon	carbon in leaves	"g/m2"
root_carbon	carbon in roots	"g/m2"
stem_carbon	carbon in stems	"g/m2"
wood_carbon	carbon in wood	"g/m2"
soil_carbon_stable	stable carbon in soil	"g/m2"
soil_carbon_fast	fast carbon in soil	"g/m2"
leaf_area_index	leaf area index	"m2/m2"
stem_area_index	stem area index	"m2/m2"
snow_age	BATS non-dimensional snow age	"-"
soil_moisture_wtd	soil water content between bottom of the soil and water table	"m3/m3"
deep_recharge	deep recharge for runoff_option 5	"m"
recharge	recharge for runoff_option 5	"m"
temperature_2m	grid diagnostic temperature at 2 meters	"K"
spec_humidity_2m	grid diagnostic specific humidity at 2 meters	"kg/kg"
eq_soil_water_vol	equilibrium soil water content	"m3/m3"
temperature_snow	snow level temperature	"K"
interface_depth	layer-bottom depth from snow surface	"m"
snow_level_ice	ice content of snow levels	"mm"
snow_level_liquid	liquid content of snow levels	"mm"

The restart files also include one text file, `${FILEDATE}.coupler.res`, which contains metadata for the restart.

Example of `${FILEDATE}.coupler.res`:

```
2        (Calendar: no_calendar=0, thirty_day_months=1, julian=2, gregorian=3, noleap=4)
2016     1     2    18     0     0    Model start time:   year, month, day, hour, minute,
↪ second
2016     1     2    18     0     0    Current model time: year, month, day, hour, minute,
↪ second
```

6.3 DA Workflow Overview

The cycling Noah-MP offline DA run is initiated using `do_submit_cycle.sh` to call the `submit_cycle.sh` script. `submit_cycle.sh` calls a third script (`do_landDA.sh`) if DA has been activated in the experiment.

Note: The offline Noah-MP model runs in vector space, whereas a cycling Noah-MP offline DA job uses JEDI's tiled cubed-sphere (*FV3*) format. Section 5.2 describes the Vector-to-Tile Converter that maps between these two formats.

6.3.1 do_submit_cycle.sh

The `do_submit_cycle.sh` script sets up the cycling job based on the user's input settings. Figure 6.1 illustrates the steps in this process.

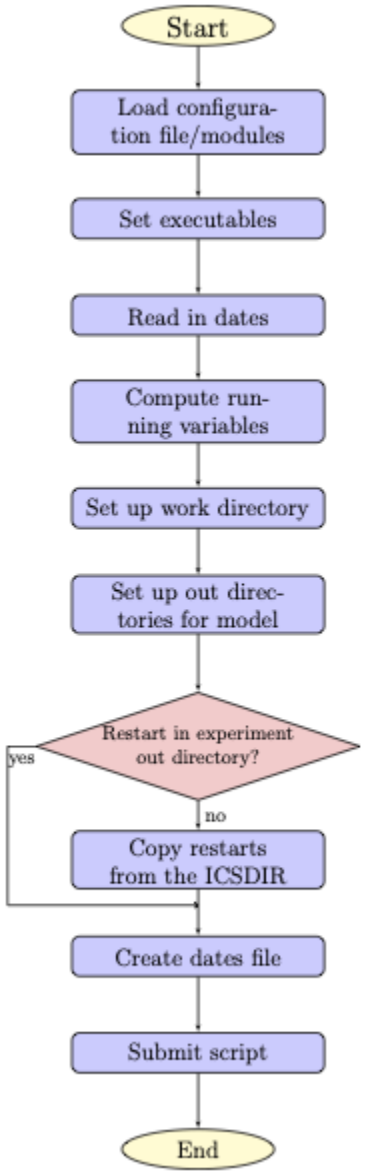

Fig. 6.1: *Flowchart of 'do_submit_cycle.sh'*

First, `do_submit_cycle.sh` reads in a configuration file for the cycle settings. This file contains the information required to run the cycle: the experiment name, start date, end date, the paths of the working directory (i.e., `workdir`) and output directories, the length of each forecast, atmospheric forcing data, the Finite-Volume Cubed-Sphere Dynamical Core (*FV3*) resolution and its paths, the number of cycles per job, the directory with initial conditions, a namelist file for running Land DA, and different DA options. Then, the required modules are loaded, and some executables are set for running the cycle. The restart frequency and running day/hours are computed from the inputs, and directories are created for running DA and saving the DA outputs. If restart files are not in the experiment output directory, the script

will try to copy the restart files from the ICSDIR directory, which should contain initial conditions files if restart files are not available. Finally, the script creates the dates file (analdates.sh) and submits the submit_cycle.sh script, which is described in detail in the next section.

6.3.2 submit_cycle.sh

The submit_cycle.sh script first exports the required paths and loads the required modules. Then, it reads the dates file and runs through all the steps for submitting a cycle if the count of dates is less than the number of cycles per job (see Figure 6.2).

As the script loops through the steps in the process for each cycle, it reads in the DA settings and selects a run type — either DA or openloop (which skips DA). Required DA settings include: DA algorithm choice, directory paths for JEDI, Land_DA (where the do_landDA.sh script is located), JEDI's input observation options, DA window length, options for constructing yaml files, etc.

Next, the system designates work and output directories and copies restart files into the working directory. If the DA option is selected, the script calls the vector2tile function and tries to convert the format of the Noah-MP model in vector space to the JEDI tile format used in *FV3* cubed-sphere space. After the vector2tile is done, the script calls the data assimilation job script (do_landDA.sh) and runs this script. Then, the tile2vector function is called and converts the JEDI output tiles back to vector format. The converted vector outputs are saved, and the forecast model is run. Then, the script checks the existing model outputs. Finally, if the current date is less than the end date, this same procedure will be processed for the next cycle.

Note: The v1.0.0 release of Land DA does not support ensemble runs. Thus, the first ensemble member (mem000) is the only ensemble member.

Here is an example of configuration settings file, settings_cycle, for the submit_cycle script:

```
export exp_name=DA_IMS_test
STARTDATE=2016010118
ENDDATE=2016010318

export WORKDIR=/*/*/
export OUTDIR=/*/*/

############################

# for LETKF,
export ensemble_size=1

export FCSTHR=24

export atmos_forc=gdas

#FV3 resolution
export RES=96
export TPATH="/*/*/"
export TSTUB="oro_C96.mx100"

# number of cycles
export cycles_per_job=1
```

(continues on next page)

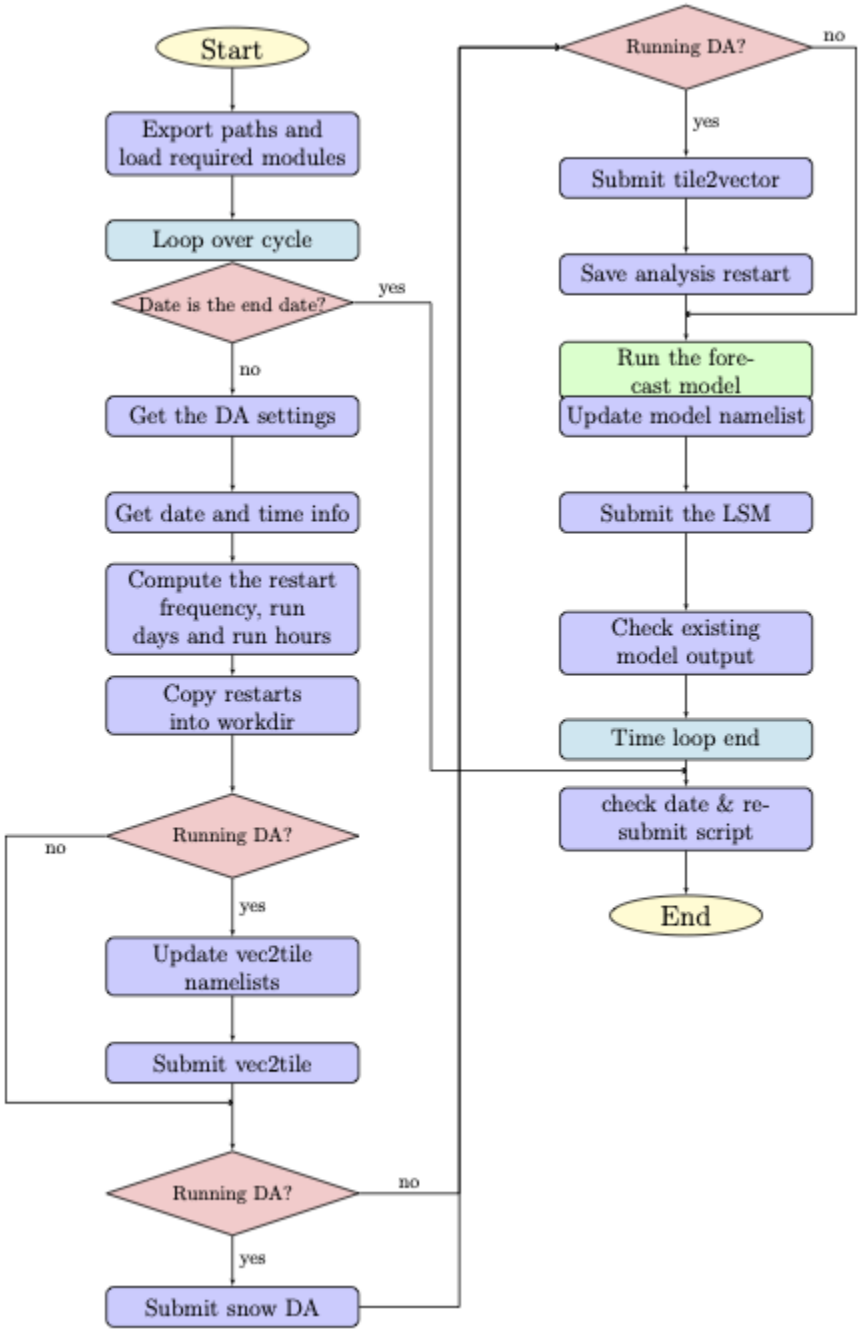

Fig. 6.2: *Flowchart of 'submit_cycle.sh'*

(continued from previous page)

```
# directory with initial conditions
export ICSDIR=/*/*/

# namelist for do_landDA.sh
export DA_config="settings_DA_test"

# if want different DA at different times, list here.
export DA_config00=${DA_config}
export DA_config06=${DA_config}
export DA_config12=${DA_config}
export DA_config18=${DA_config}
```

Parameters for `submit_cycle.sh`

exp_name
Specifies the name of experiment.

STARTDATE
Specifies the experiment start date. The form is YYYYMMDDHH, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour.

ENDDATE
Specifies the experiment end date. The form is YYYYMMDDHH, where YYYY is a 4-digit year, MM is a valid 2-digit month, DD is a valid 2-digit day, and HH is a valid 2-digit hour.

WORKDIR
Specifies the path to a temporary directory from which the experiment is run.

OUTDIR
Specifies the path to a directory where experiment output is saved.

ensemble_size
Specifies the size of the ensemble (i.e., number of ensemble members). Use 1 for non-ensemble runs.

FCSTHR
Specifies the length of each forecast in hours.

atmos_forc
Specifies the name of the atmospheric forcing data. Valid values include: `gdas` | `era5`

RES
Specifies the resolution of FV3. Valid values: `C96`

Note: Other resolutions are possible but not supported for this release.

TPATH
Specifies the path to the directory containing the orography files.

TSTUB
Specifies the file stub/name for orography files in `TPATH`. This file stub is named `oro_C${RES}` for atmosphere-only orography files and `oro_C{RES}.mx100` for atmosphere and ocean orography files.

cycles_per_job
Specifies the number of cycles to submit in a single job.

ICSDIR
Specifies the path to a directory containing initial conditions data.

DA_config
Configuration setting file for `do_landDA.sh`. Set `DA_config` to `openloop` to skip data assimilation (and prevent a call `do_landDA.sh`).

DA_configXX
Configuration setting file for `do_landDA.sh` at `XX` hr. If users want to perform DA experiment at different times, list these in the configuration setting file. Set to `openloop` to skip data assimilation (and prevent a call `do_landDA.sh`).

6.3.3 do_landDA.sh

The `do_landDA.sh` runs the data assimilation job inside the `sumbit_cycle.sh` script. Currently, the only DA option is the snow Local Ensemble Transform Kalman Filter-Optimal Interpolation (LETKF-OI, Frolov *et al.* [FJSWD22], 2022). Figure 6.3 describes the workflow of this script.

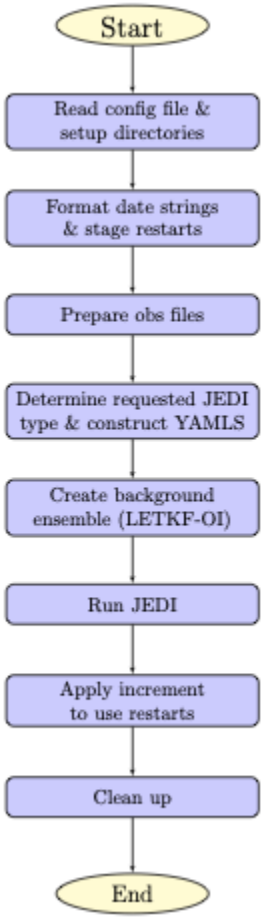

Fig. 6.3: *Flowchart of 'do_landDA.sh'*

First, to run the DA job, `do_landDA.sh` reads in the configuration file and sets up the directories. The date strings are formatted for the current date and previous date. For each tile, restarts are staged to apply the JEDI update. In this stage, all files will get directly updated. Then, the observation files are read and prepared for this job. Once the

JEDI type is determined, `yaml` files are constructed. Note that if the user specifies a `yaml` file, the script uses that one. Otherwise, the script builds the `yaml` files. For LETKF-OI, a pseudo-ensemble is created by running the python script (`letkf_create_ens.py`). Once the ensemble is created, the script runs JEDI and applies increment to UFS restarts.

Below, users can find an example of a configuration settings file, `settings_DA`, for the `do_landDA.sh` script:

```
LANDDADIR=${CYCLEDIR}/DA_update/

#############################
# DA options

OBS_TYPES=("GHCN")
JEDI_TYPES=("DA")

WINLEN=24

# YAMLS
YAML_DA=construct

# JEDI DIRECTORIES
JEDI_EXECDIR=
fv3bundle_vn=20230106_public
```

LANDDADIR
Specifies the path to the `do_landDA.sh` script.

OBS_TYPES
Specifies the observation type. Format is "Obs1" "Obs2". Currently, only GHCN observation data is available.

JEDI_TYPES
Specifies the JEDI call type for each observation type above. Format is "type1" "type2". Valid values: `DA` | `HOFX`

Value	Description
DA	Data assimilation
HOFX	A generic application for running the model forecast (or reading forecasts from file) and computing H(x)

WINLEN
Specifies the DA window length. It is generally the same as the `FCSTLEN`.

YAML_DA
Specifies whether to construct the `yaml` name based on requested observation types and their availabilities. Valid values: `construct` | *desired YAML name*

Value	Description
construct	Enable constructing the YAML
desired YAML name	Will not test for availability of observations

JEDI_EXECDIR
Specifies the JEDI FV3 build directory. If using different JEDI version, users will need to edit the `yaml` files with the desired directory path.

fv3bundle_vn
Specifies the date for JEDI `fv3-bundle` checkout (used to select correct `yaml`).

CHAPTER

SEVEN

GLOSSARY

CCPP
The Common Community Physics Package is a forecast-model agnostic, vetted collection of code containing atmospheric physical parameterizations and suites of parameterizations for use in Numerical Weather Prediction (NWP) along with a framework that connects the physics to the host forecast model.

container
Docker describes a container as "a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another."

cycle
An hour of the day on which a forecast is started. In the Land DA System, it usually follows YYYYMMDD-HHmmss format.

data assimilation
One of the major sources of error in weather and climate forecasts is uncertainty related to the initial conditions that are used to generate future predictions. Even the most precise instruments have a small range of unavoidable measurement error, which means that tiny measurement errors (e.g., related to atmospheric conditions and instrument location) can compound over time. These small differences result in very similar forecasts in the short term (i.e., minutes, hours), but they cause widely divergent forecasts in the long term. Errors in weather and climate forecasts can also arise because models are imperfect representations of reality. Data assimilation systems seek to mitigate these problems by combining the most timely observational data with a "first guess" of the atmospheric state (usually a previous forecast) and other sources of data to provide a "best guess" analysis of the atmospheric state to start a weather or climate simulation. When combined with an "ensemble" of model runs (many forecasts with slightly different conditions), data assimilation helps predict a range of possible atmospheric states, giving an overall measure of uncertainty in a given forecast.

forcing data
In coupled mode, data that are generated by one component of a model can be fed into another component to provide required input describing the state of the Earth system. When models are run in offline, or "uncoupled" mode, the model does not receive this input from another active component, so "forcing" files are provided. These files may consist of observational data or data gathered when running other components separately, and they contain values for the required input fields.

FV3
The Finite-Volume Cubed-Sphere dynamical core (dycore). Developed at NOAA's Geophysical Fluid Dynamics Laboratory (GFDL), it is a scalable and flexible dycore capable of both hydrostatic and non-hydrostatic atmospheric simulations. It is the dycore used in the UFS Weather Model.

JEDI
The Joint Effort for Data assimilation Integration (JEDI) is a unified and versatile data assimilation (DA) system for Earth System Prediction. It aims to enable efficient research and accelerated transition from research to operations by providing a framework that takes into account all components of the Earth system in a consistent manner. The JEDI software package can run on a variety of platforms and for a variety of purposes, and it is

designed to readily accommodate new atmospheric and oceanic models and new observation systems. The JEDI User's Guide contains extensive information on the software.

JEDI is developed and distributed by the Joint Center for Satellite Data Assimilation, a multi-agency research center hosted by the University Corporation for Atmospheric Research (UCAR). JCSDA is dedicated to improving and accelerating the quantitative use of research and operational satellite data in weather, ocean, climate, and environmental analysis and prediction systems.

HPC
High-Performance Computing.

MPI
MPI stands for Message Passing Interface. An MPI is a standardized communication system used in parallel programming. It establishes portable and efficient syntax for the exchange of messages and data between multiple processors that are used by a single computer program. An MPI is required for high-performance computing (HPC) systems.

netCDF
NetCDF (Network Common Data Form) is a file format and community standard for storing multidimensional scientific data. It includes a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

NWP
Numerical Weather Prediction (NWP) takes current observations of weather and processes them with computer models to forecast the future state of the weather.

RDHPCS
Research and Development High-Performance Computing Systems.

Spack
Spack is a package management tool designed to support multiple versions and configurations of software on a wide variety of platforms and environments. It was designed for large supercomputing centers, where many users and application teams share common installations of software on clusters with exotic architectures.

spack-stack
The spack-stack is a collaborative effort between the NOAA Environmental Modeling Center (EMC), the UCAR Joint Center for Satellite Data Assimilation (JCSDA), and the Earth Prediction Innovation Center (EPIC). *spack-stack* is a repository that provides a *Spack*-based method for building the software stack required for numerical weather prediction (NWP) tools such as the Unified Forecast System (UFS) and the Joint Effort for Data assimilation Integration (JEDI) framework. *spack-stack* uses the Spack package manager along with custom Spack configuration files and Python scripts to simplify installation of the libraries required to run various applications. The *spack-stack* can be installed on a range of platforms and comes pre-configured for many systems. Users can install the necessary packages for a particular application and later add the missing packages for another application without having to rebuild the entire stack.

UFS
The Unified Forecast System (UFS) is a community-based, coupled, comprehensive Earth modeling system consisting of several applications (apps). These apps span regional to global domains and sub-hourly to seasonal time scales. The UFS is designed to support the *Weather Enterprise* and to be the source system for NOAA's operational numerical weather prediction applications. For more information, visit https://ufscommunity.org/.

Weather Enterprise
Individuals and organizations from public, private, and academic sectors that contribute to the research, development, and production of weather forecast products; primary consumers of these weather forecast products.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Bra99] B. Brasnett. A global analysis of snow depth for numerical weather prediction. *J. Appl. Meteorol.*, 38(6):726–740, 1999. doi:10.1175/1520-0450(1999)038<0726:AGAOSD>2.0.CO;2.

[CZ09] F. Chen and Y. Zhang. On the coupling strength between the land surface and the atmosphere: from viewpoint of surface exchange coefficients. *Geophys. Res. Lett.*, 2009. doi:10.1029/2009GL037980.

[FMMW+07] Y. Fan, G. Miguez-Macho, C.P. Weaver, R. Walko, and A. Robock. Incorporating water table dynamics in climate modeling: 1. water table observations and equilibrium water table simulations. *Journal of Geophysical Research*, 112(D10):19569–19585, 2007. doi:10.1029/2006JD008111.

[FJSWD22] S. Frolov, J.S. Whitaker, and C. Draper. Including parameterized error covariance in local ensemble solvers: experiments in a 1d model with balance constraints. *Quarterly Journal of the Royal Meteorological Society*, 2022. doi:10.1002/qj.4289.

[HVMWK20] D. Holdaway, G. Vernieres, M. Wlasak, and S. King. Status of model interfacing in the joint effort for data assimilation integration (jedi). *JCSDA Quarterly Newsletter*, 66:15–24, 2020. doi:10.25923/RB19-0Q26.

[HHZ+20] R. Honeyager, S. Herbener, X. Zhang, A. Shlyaeva, and Y. Trémolet. Observations in the joint effort for data assimilation integration (jedi) — unified forward operator (ufo) and interface for observation data access (ioda). *JCSDA Quarterly Newsletter*, 66:24–33, 2020. doi:10.25923/RB19-0Q26.

[HEJKS07] B.R. Hunt, E.J. Kostelich, and I. Szunyogh. Efficient data assimilation for spatiotemporal chaos: a local ensemble transform kalman filter. *Physica D: Nonlinear Phenomena*, 230(1–2):112–126, 2007. doi:10.1016/j.physd.2006.11.008.

[Jor91] R. Jordan. A one-dimensional temperature model for a snow cover: technical documentation for sntherm.89. Technical Report Special Rep. 91-16, U.S. Army Cold Regions Research and Engineering Laboratory, Hanover, NH, October 1991.

[KSM+99] V. Koren, J. Schaake, K. Mitchell, Q.Y. Duan, F. Chen, and J.M. Baker. A parameterization of snowpack and frozen ground intended for ncep weather and climate models. *Journal of Geophysical Research*, 104(D16):19569–19585, 1999. doi:10.1029/1999JD900232.

[MMFW+07] G. Miguez-Macho, Y. Fan, C.P. Weaver, R. Walko, and A. Robock. Incorporating water table dynamics in climate modeling: 2. formulation, validation, and soil moisture simulation. *Journal of Geophysical Research*, 2007. doi:10.1029/2006JD008112.

[NY06] G.-Y. Niu and Z.-L. Yang. Effects of frozen soil on snowmelt runoff and soil water storage at a continental scale. *Journal of Hydrometeorology*, 7(5):937–952, 2006. doi:10.1175/JHM538.1.

[NYD+07] G.-Y. Niu, Z.-L. Yang, R.E. Dickinson, L.E. Gulden, and H. Su. Development of a simple groundwater model for use in climate models and evaluation with gravity recovery and climate experiment data. *Journal of Geophysical Research*, 2007. doi:10.1029/2006JD007522.

[NYM+11] G.-Y. Niu, Z.-L. Yang, K.E. Mitchell, F. Chen, M.B. Ek, M. Barlage, A. Kumar, K. Manning, D. Niyogi, E. Rosero, M. Tewari, and Y. Xia. Simple water balance model for estimating runoff at different spatial and temporal scales. *Journal of Geophysical Research*, 2011. doi:10.1029/2010JD015139.

[NYREDG05] G.-Y. Niu, Z.-L. Yang, R.E. Dickinson, and L.E. Gulden. A simple topmodel-based runoff parameterization (simtop) for use in gcms. *Journal of Geophysical Research*, 2005. doi:10.1029/2005JD006111.

[SZ09] K. Sakaguchi and X. Zeng. Effects of soil wetness, plant litter, and under-canopy atmospheric stability on ground evaporation in the community land model (clm3.5). *Journal of Geophysical Research*, 2009. doi:10.1029/2008JD010834.

[SKD+96] J.C. Schaake, V.I. Koren, Q.-Y. Duan, K.E. Mitchell, and F. Chen. Simple water balance model for estimating runoff at different spatial and temporal scales. *Journal of Geophysical Research*, 101(D3):7461–7475, 1996. doi:10.1029/95JD02892.

[SMDHH92] P.J. Sellers, M.D. Heiser, and F.G. Hall. Relations between surface conductance and spectral vegetation indices at intermediate (100 m2 to 15 km2) length scales. *Journal of Geophysical Research*, 97(D17):19033–19059, 1992. doi:10.1029/92JD01096.

[TA20] Y. Trémolet and T. Auligné. The joint effort for data assimilation integration (jedi). *JCSDA Quarterly Newsletter*, 66:1–5, 2020. doi:10.25923/RB19-0Q26.

[WH12] J.S. Whitaker and T.M. Hamill. Evaluating methods to account for system errors in ensemble data assimilation. *Monthly Weather Review*, 140(9):3078–3089, 2012. doi:10.1175/MWR-D-11-00276.1.

[ZCSS04] F. Zhang, C. Snyder, and J. Sun. Impacts of initial estimate and observation availability on convective-scale data assimilation with an ensemble kalman filter. *Monthly Weather Review*, 132(5):1238–1253, 2004. doi:10.1175/1520-0493(2004)132%3C1238:IOIEAO%3E2.0.CO;2.

INDEX