Overview on the MPAS-Workflow and graphics package

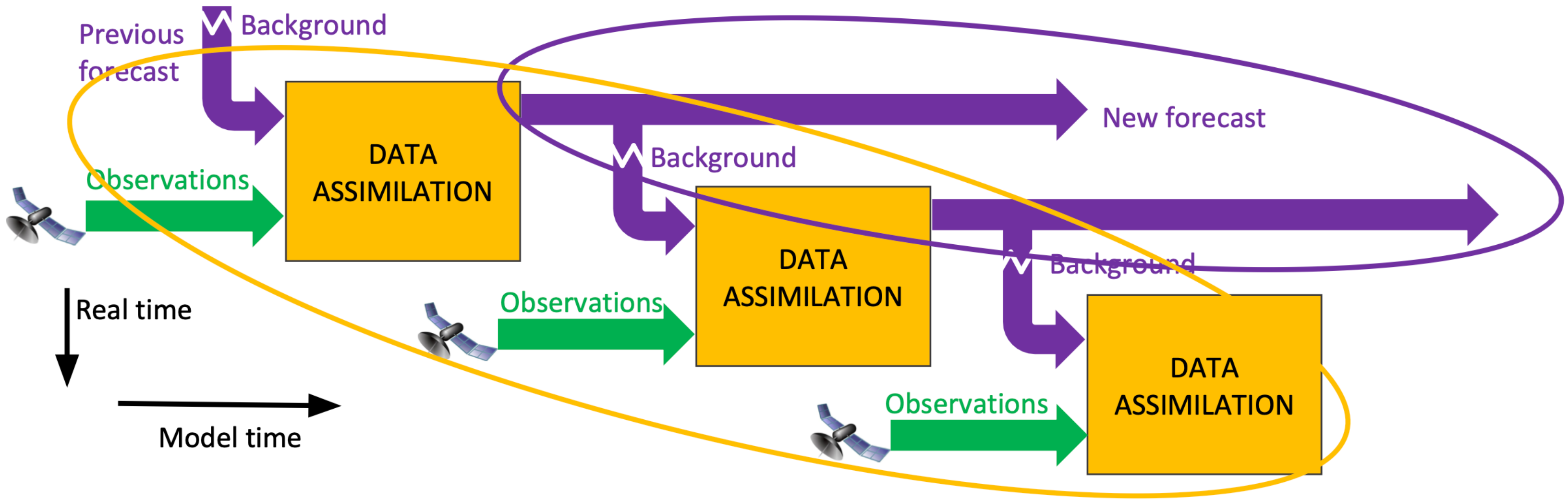
Presented by Zhiquan (Jake) Liu
based on materials prepared by Ivette Hernández Baños

Mesoscale & Microscale Meteorology Laboratory
National Center for Atmospheric Research

MPAS-JEDI Tutorial, NCU, 25-26 October, 2023

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Typical workflow for real-time or retrospective cycling DA and forecast



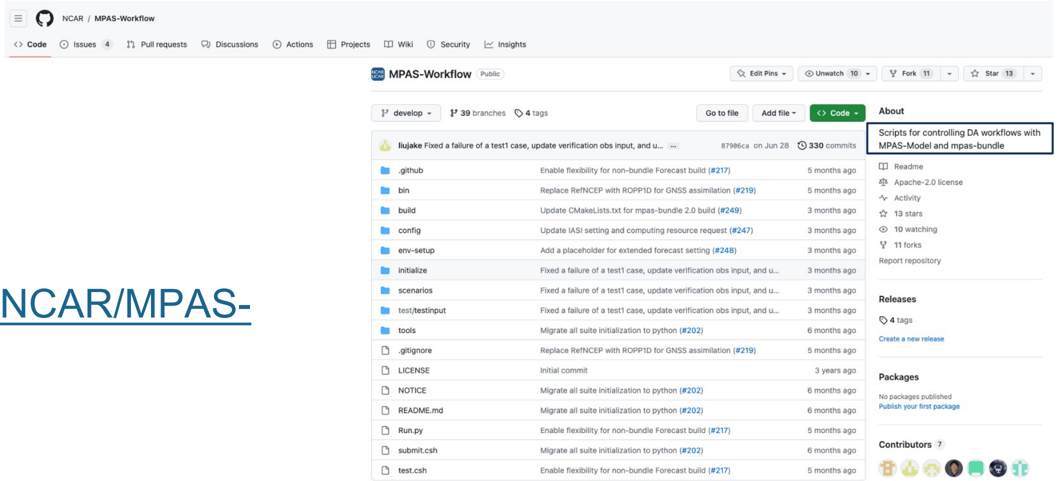
Source: Tom Auligné and Yannick Trémolet

Outline

- ❑ MPAS-Workflow
  - ❑ Applications
  - ❑ Data (Pre-processing)
  - ❑ Post-processing
  - ❑ Framework
  - ❑ Scenario YAMLS
  - ❑ Predefined tests
  - ❑ Suites
  - ❑ Tips
- ❑ Graphics package
  - ❑ Functionalities
  - ❑ Examples

MPAS-Workflow

- ➢ Developed at NCAR/MMM to aid cycling experiments with MPAS and MPAS-JEDI
  - ○ Tailored for the PANDAC specific use
  - ○ last version: 2.0.0
- ➢ CYLC-based workflow manager (v7.8.3) + Python + C-Shell scripts
- ➢ Currently, only operates on NCAR's Cheyenne HPC



- ➢ Open-source: https://github.com/NCAR/MPAS-Workflow

but **NOT** supported

MPAS-Workflow

- ❑ constructs each JEDI application YAML, with high flexibility for a number of configurations
  - ❑ e.g., do variational bias correction or not, SST and XICE update, number of outer loops, number of ensemble members, observers, etc.
- ❑ links all necessary input data
- ❑ can be used for cycling and no cycling experiments
  - ❑ e.g., generate observations, generate GFS analyses in MPAS ICs format, generate free forecast from GFS analyses
- ❑ can handle cold and warm start
- ❑ constructs and submit the CYLC suite for the cycling (and no cycling) experiment
- ❑ can be used to run real-time applications

MPAS-Workflow: applications

Data assimilation

- Variational
  - 3DVar
  - 3/4DEnVar
  - Hybrid
  - EDA
    - Application specific YAML templates:
      `config/jedi/applications/*.yaml`
    - Task script:
      `bin/Variational.csh`
      `bin/EnsembleOfVariational.csh`
- EnKF
  - LETKF
  - GETKF
    - Application YAML template:
      `config/jedi/applications/enkf.yaml`
    - Task script:
      `bin/EnKF.csh`
      `bin/EnKFObserver.csh`
- RTPP
  - Application YAML template:
    `config/jedi/applications/rtpp.yaml`
  - Task script:
    `bin/RTPP.csh`
- HofX
  - Application YAML template:
    `config/jedi/applications/hofx.yaml`
  - Task script:
    `bin/HofX.csh`

- Observation YAML stubs:

```
config/jedi/ObsPlugs/da
├── base/*.yaml
├── bias/*.yaml
├── filters/*.yaml
└── filtersWithBias/*.yaml
```

- YAMLS are further populated by `bin/PrepJEDI.csh` and/or
  `bin/InitVariationals.csh`
  `bin/InitEnKF.csh`

- Initialization scripts (`initialize/`) set up the suite:
  1. Read/parse scenario YAML
  2. Define c-shell environment variables for use by task scripts in the run directory
  3. Define inter-task and inter-cycle dependencies
  4. Submit Cylc suite

MPAS-Workflow: applications

Data assimilation:

- 3denvar.yaml

<u>Configurable options:</u>

\`InnerNamelistFile\`, \`InnerStreamsFile\`, \`thisISO8601Date\`, \`AnalysisVariables\`, \`VariationalMinimizer\`, \`VariationalIterations\`, \`StateVariables\`, \`EnsemblePbMembers\`, \`Observers\`, …


```yaml
_iteration: &iterationConfig
iteration: &iterationConfig
  geometry:
    nml_file: {{InnerNamelistFile}}
    streams_file: {{InnerStreamsFile}}{{StreamsFileMember}}
    deallocate non-da fields: true
    interpolation type: unstructured
  gradient norm reduction: 1e-3
_member: &memberConfig
  date: &analysisDate {{thisISO8601Date}}
  state variables: &incvars [{{AnalysisVariables}}]
  stream name: ensemble
output:
  filename: {{anStateDir}}{{MemberDir}}/{{anStatePrefix}}.$Y-$M-$D_$h.$m.$s.nc
  stream name: analysis
variational:
  minimizer:
{{VariationalMinimizer}}
  iterations:
{{VariationalIterations}}
final:
  diagnostics:
    departures: oman
cost function:
  cost type: 3D-Var
  window begin: {{windowBegin}}
  window length: {{windowLength}}
  jb evaluation: false
  geometry:
    nml_file: {{OuterNamelistFile}}
    streams_file: {{OuterStreamsFile}}{{StreamsFileMember}}
    deallocate non-da fields: true
    interpolation type: unstructured
  analysis variables: *incvars
  background:
    state variables: [{{StateVariables}}]
    filename: {{bgStateDir}}{{MemberDir}}/{{bgStatePrefix}}.{{thisMPASFileDate}}.nc
    date: *analysisDate
  background error:
    covariance model: ensemble
    localization:
      localization method: SABER
      saber central block:
        saber block name: BUMP_NICAS
        active variables: *incvars
        read:
          io:
            data directory: {{bumpLocDir}}
            files prefix: {{bumpLocPrefix}}
          drivers:
            multivariate strategy: duplicated
            read local nicas: true
          model:
            level for 2d variables: last
{{EnsemblePbMembers}}
{{EnsemblePbInflation}}
  observations:
    obs perturbations: {{ObsPerturbations}}
    observers:
{{Observers}}
```


NCAR | UCAR

MPAS-Workflow: applications

Data assimilation:

- enkf.yaml

<u>Configurable options:</u>

\`driver\`, \`thisISO8601Date\`, \`AnalysisVariables\`, \`EnKFNamelistFile\`, \`EnKFStreamsFile\`, \`StateVariables\`, \`EnsembleMembers\`, \`localEnsembleDASolver\`, \`verticalLocalizationLengthscale\`, …

```yaml
member: &memberConfig
  date: &analysisDate {{thisISO8601Date}}
  state variables: [{{StateVariables}}]
  stream name: background

_as observer: &asObserver
  run as observer only: true
  update obs config with geometry info: false

_as solver: &asSolver
  read HX from disk: true
  do posterior observer: false
  save posterior ensemble: true
  save posterior mean: true

_letkf geometry: &3DLETKFGeometry
  iterator dimension: 3

_letkf geometry: &2DLETKFGeometry
  iterator dimension: 2

_lgetkf geometry: &3DGETKFGeometry
  iterator dimension: 2

geometry:
  <<: *{{localizationDimension}}{{localEnsembleDASolver}}Geometry
  nml_file: {{EnKFNamelistFile}}
  streams_file: {{EnKFStreamsFile}}
  deallocate non-da fields: true

window begin: {{windowBegin}}
window length: {{windowLength}}

background:
{{EnsembleMembers}}

increment variables: [{{AnalysisVariables}}]

observations:
  observers:
{{Observers}}

driver: *{{driver}}

local ensemble DA:
  solver: {{localEnsembleDASolver}}
  vertical localization:
    fraction of retained variance: 0.95
    lengthscale: {{verticalLocalizationLengthscale}}
    lengthscale units: modellevel

output:
  filename: {{anStateDir}}/mem%{member}%/{{anStatePrefix}}.$Y-$M-$D_$h.$m.$s.nc
  stream name: analysis
```

MPAS-Workflow: applications

Data assimilation:

- Observers: e.g., amsua_n15

aircraft, sondes, sfc, satwind, satwnd, gnssro ⇒ base + filters
amsua, mhs ⇒ base + filters or base + bias + filtersWithBias

base → **bias** → **filtersWithBias**

```
- obs space:
    <<: *ObsSpace
    name: amsua_n15
    _obsdatain: &ObsDataIn
      engine:
        type: H5File
        obsfile: {{InDBDir}}/amsua_n15_obs_{{thisValidDate}}.h5
    _obsdataout: &ObsDataOut
      engine:
        type: H5File
        obsfile:
{{OutDBDir}}{{MemberDir}}/{{obsPrefix}}_amsua_n15{{ObsOut
Suffix}}.h5
    obsdatain: *{{ObsDataIn}}
    {{ObsDataOut}}
    simulated variables: [brightnessTemperature]
    channels: &amsua_n15_channels 1-15
  obs error: *ObsErrorDiagonal
  <<: *horizObsLoc
  obs operator:
    <<: *clearCRTMObsOperator
    obs options:
      <<: *CRTMObsOptions
      Sensor_ID: amsua_n15
  get values:
    <<: *GetValues
```

```
obs bias:
  input file: {{biasCorrectionDir}}/satbias_amsua_n15.h5
  output file: {{OutDBDir}}{{MemberDir}}/satbias_amsua_n15.h5
  variational bc:
    predictors: &predictors2
    - name: constant
    - name: lapse_rate
      order: 2
      tlapse: &amsua15tlap {{fixedTlapmeanCov}}/amsua_n15_tlapmean.txt
    - name: lapse_rate
      tlapse: *amsua15tlap
    - name: emissivity
    - name: scan_angle
      order: 4
    - name: scan_angle
      order: 3
    - name: scan_angle
      order: 2
    - name: scan_angle
  covariance:
    minimal required obs number: 20
    variance range: [1.0e-6, 10.]
    step size: 1.0e-4
    largest analysis variance: 10000.0
    prior:
      input file: {{biasCorrectionDir}}/satbias_cov_amsua_n15.h5
      inflation:
        ratio: 1.1
        ratio for small dataset: 2.0
    output file: {{OutDBDir}}{{MemberDir}}/satbias_cov_amsua_n15.h5
```

```
obs filters:
- filter: Domain Check
  where:
  - variable:
      name: MetaData/sensorZenithAngle
    maxvalue: 45.0
#  CLW Retrieval Check
- filter: Bounds Check
  filter variables:
  - name: brightnessTemperature
    channels: 1-6, 15
  test variables:
  - name: ObsFunction/CLWRetMW
    options:
      clwret_ch238: 1
      clwret_ch314: 2
      clwret_types: [ObsValue]
  maxvalue: 999.0
  action:
    name: reject
```

Functions in filters see: https://jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com/en/stable/index.html

MPAS-Workflow: applications

Forecast:

- `bin/Forecast.csh`: performs 6-hr forecast from DA analysis or extended forecast (longer than DA window)

Input:

- analysis (mpasin)
- static files
- lookup tables
- mesh graph info
- namelist and streams files
- atmosphere_model executable

Options:

- update SST and XICE from GFS/GEFS analysis valid at analysis time
- IAU (Incremental Analysis Update)

```
# Run the executable
# ==================
# load Forecast environment here to avoid conflict between multiple python versions
cd ${mainScriptDir}
source config/environmentForecast.csh
cd -

set log = log.${MPASCore}.0000.out
foreach f ($log $ForecastEXE)
  if ( -e $f ) rm -v $f
end
ln -sfv ${ForecastBuildDir}/${ForecastEXE} ./
${mpiCommand} ./${ForecastEXE}
```

MPAS-Workflow: applications

HofX:

- `bin/HofX.csh`: Carries out multiple observation operators ("h(x)") on 1 or more MPAS-Atmosphere forecasts

Input:

- state (single or ensemble members) ⇒ previously generated
- static files
- lookup tables
- mesh graph info
- namelist and streams files
- mpasjedi_hofx3d.x executable
- geovars.yaml
- observations in `/dbIn` folder (observers specified in `initialize/applications/HofX.py`)

Standalone application used to verify MPAS 6-hr forecasts on observation space
Facilitates verifying independent observations

MPAS-Workflow: applications

Generate external analysis:

- ➔ Link external analyses, pre-converted to MPAS data format
- ➔ Retrieve GFS analysis and convert it to MPAS format
    - ◆ Get/download grib analysis:
        - RDA: `GetGFSAnalysisFromRDA.csh`
        - FTP: `GetGFSAnalysisFromFTP.csh`
    - ◆ Ungrib:
        - `UngribExternalAnalysis.csh`
    - ◆ Convert to MPAS ICs format:
        - `ExternalAnalysisToMPAS.csh`
        - `mpas_init executable`

Specific suite: `initialize/suites/GenerateExternalAnalyses.py`

Scenario YAML: `scenarios/GenerateGFSAnalyses.yaml`

MPAS-Workflow: applications

Generate observations:

➔ Converts observations in prepBURF and BURF to IODA-V3 format

- GetObs.csh
- ObsToIODA.csh
  - obs2ioda converter

Specific suite: `initialize/suites/GenerateObs.py`

Scenario YAML: `scenarios/GenerateObs.yaml`

MPAS-Workflow: data

```
initialize/data
├── DataList.py
├── ExternalAnalyses.py
├── FirstBackground.py
├── Model.py
├── ObsEnsemble.py
├── Observations.py
├── StateEnsemble.py
└── StaticStream.py
```

```
benchmarkObservations = [
  # anchor
  'aircraft',
  'gnssrobndropp1d',
  'satwind',
  'satwnd',
  'sfc',
  'sondes',
  # MW satellite-based
  'amsua_aqua',
  'amsua_metop-a',
  'amsua_metop-b',
  'amsua_n15',
  'amsua_n18',
  'amsua_n19',
  'mhs_metop-a',
  'mhs_metop-b',
  'mhs_n18',
  'mhs_n19',
]
```

defaults = 'scenarios/defaults/observations.yaml'

- resources:
  - NCEPFTPOnline
  - GladeRDAOnline
  - PANDACArchive
  - PANDACArchiveForVarBC
  - GenerateObs

Other resources can be added as needed

outerMesh`, `innerMesh`, `ensembleMesh`, `GraphInfoDir`

MPAS-Workflow: Post-processing

- ❑ Verify vs. GFS analyses: VerifyModel
  - ➢ Inputs: MPAS forecast and GFS analyses on MPAS format
- ❑ Verify vs. observations: VerifyObs
  - ➢ Inputs: HofX or DA observation feedback files:
    - ■ DA: omb/oma obsout diagnostics (same assimilated observations)
    - ■ model on observations space: HofX obsout diagnostics + VerifyObs (instantiates it own HofX)

Observers

- **Sondes**, **aircraft**, **satellite-derived winds**, **GNSSRO**, **surface pressure**
- **AMSU-A** (NOAA-15, NOAA-18, NOAA-19, METOP-A, METOP-B)
- **MHS** (NOAA-18, NOAA-19, METOP-A, METOP-B)
- **IASI** (METOP-A, METOP-B, METOP-C)
- **ABI** (GOES-16) and **AHI** (Himawari-8)

MPAS-Workflow: framework

```
initialize/framework
├── Build.py
├── Experiment.py
├── HPC.py
├── Naming.py
└── Workflow.py
```

Specify:

- mpas-bundle build directory
- non-bundle applications
- executables names

Download and compile maps-bundle

HPC-specific resource:

- account number
- queue options
- default processor taks

MPAS-Workflow: scenarios

- Configuration for a particular instance of an MPAS-Workflow Cylc suite
- Nested key-value parameters that users can specify for their particular needs
- Include default YAMLS that describe options that users may select, such as the observations resource, the first background, etc…

  `scenarios/defaults/*.yaml`

source env-script/cheyenne.${YourShell}

Running:

./Run.py ./scenarios/{{scenario}}.yaml

OR

./Run.py ./test/testinput/{{scenario}}.yaml

MPAS-Workflow: predefined tests

`/test/testinput`

Pre-defined scenarios that exercise functionality in the workflow (WarmStart == offline 1st state; ColdStart == online 1st state)

`test1.yaml`

scenarios: [
- 3denvar_O30kmIE60km_WarmStart.yaml
- 3denvar_OIE120km_IAU_WarmStart.yaml
- 3dvar_O30kmIE60km_ColdStart.yaml
- 3dvar_OIE120km_ColdStart.yaml
- 3dvar_OIE120km_WarmStart_PostProcess.yaml
- 3dvar_OIE120km_WarmStart.yaml
- eda_OIE120km_WarmStart.yaml
- ForecastFromGFSAnalysesMPT.yaml
- getkf_OIE120km_WarmStart.yaml
- letkf_OIE120km_WarmStart.yaml]

Run:

./test.csh

./Run.py test/testinput/test1.yaml

or

./Run.py test/testinput/test2.yaml

MPAS-Workflow: scenarios

`3dvar_OIE120km_WarmStart.yaml`

Default is post-processing
To turn it off:

```
forecast
    post: []
variational:
        post: []
```

`execute` key can be added to allow for more flexibility, e.g.,

```
forecast:
        execute: True
```

Already generated/archived observations in IODA format

```yaml
experiment:
  name: '3dvar_OIE120km_WarmStart_TEST'
externalanalyses:
  resource: "GFS.PANDAC"
firstbackground:
  resource: "PANDAC.GFS"
forecast:
  # turn off post to reduce overhead
  post: []
hpc:
  CriticalQueue: economy
  NonCriticalQueue: economy
members:
  n: 1
model:
  outerMesh: 120km
  innerMesh: 120km
  ensembleMesh: 120km
observations:
  resource: PANDACArchive
variational:
  DAType: 3dvar
  nInnerIterations: [15,]
  # turn off post to reduce overhead
  post: []
workflow:
  first cycle point: 20180414T18
  final cycle point: 20180415T06
```

MPAS-Workflow: scenarios

YAML configuration for extended forecast:

```
extendedforecast:
  meanTimes: T00,T06,T12,T18
  lengthHR: 120
  outIntervalHR: 12
  post: [verifyobs, verifymodel]
forecast:
  execute: False
  post: [ ]
variational:
  execute: False
  post: [ ]
…
```

Add more configurations in the a scenario YAML:

```
build:
  mpas bundle: <path/to/build/code>
variational:
  observers: [
    aircraft,
    sfc,
    sondes,
forecast:
  job:
    60km:
      seconds: 600
      nodes: 3
      PEPerNode: 32
observations:
  resource: Archive
  resources:
    Archive:
      IODADirectory:
        da:
          aircraft: <path/to/data>
```

MPAS-Workflow: suites

```
initialize/suites
├── Cycle.py
├── ForecastFromExternalAnalyses.py
├── GenerateExternalAnalyses.py
├── GenerateObs.py
└── SuiteBase.py
```

When an experiment is launched, a local copy of the MPAS-Workflow resides in the experiment root folder with the generated CYLC suite

MPAS-Workflow/suite.rc

```
[meta]
  title = exp_name
[cylc]
  UTC mode = False
[scheduling]
  initial cycle point = 20180414T18
  final cycle point   = 20180420T00
max active cycle points = 4
  [[queues]]
    # externalanalyses
    [[[LinkExternalAnalyses]]]
      members = LinkExternalAnalyses
…
[[dependencies]]        <- first time
[[[R1]]]
  graph = """
  """
    [[[+PT6H/PT6H]]]                                   } cycling
      graph = """
        ObsReady__ => PreDA__
        InitVariationals_0 => InitVariationals_1
        ForecastFinished__[-PT6H] => PreDA__
        PreDA__ => InitDA
        InitDA:succeed-all => DAExec
                …
      """
[runtime]
…
```

MPAS-Workflow: suites

```
initialize/suites
├── Cycle.py
├── ForecastFromExternalAnalyses.py
├── GenerateExternalAnalyses.py
├── GenerateObs.py
└── SuiteBase.py
```

Processed suite (with actual variable value changed)

```
/glade/scratch/<username>/cylc-run/<experiment_name>
    ├── cylc-suite.db -> log/db
    ├── log
    ├── share
    ├── suite.rc.processed
    └── work
```

`log/job` ⇒ submitted jobs and log files for each cycle and task

Very helpful for debugging

MPAS-Workflow: suites

Experiments folders structure: ivette_3dvar_OIE120km_WarmStart

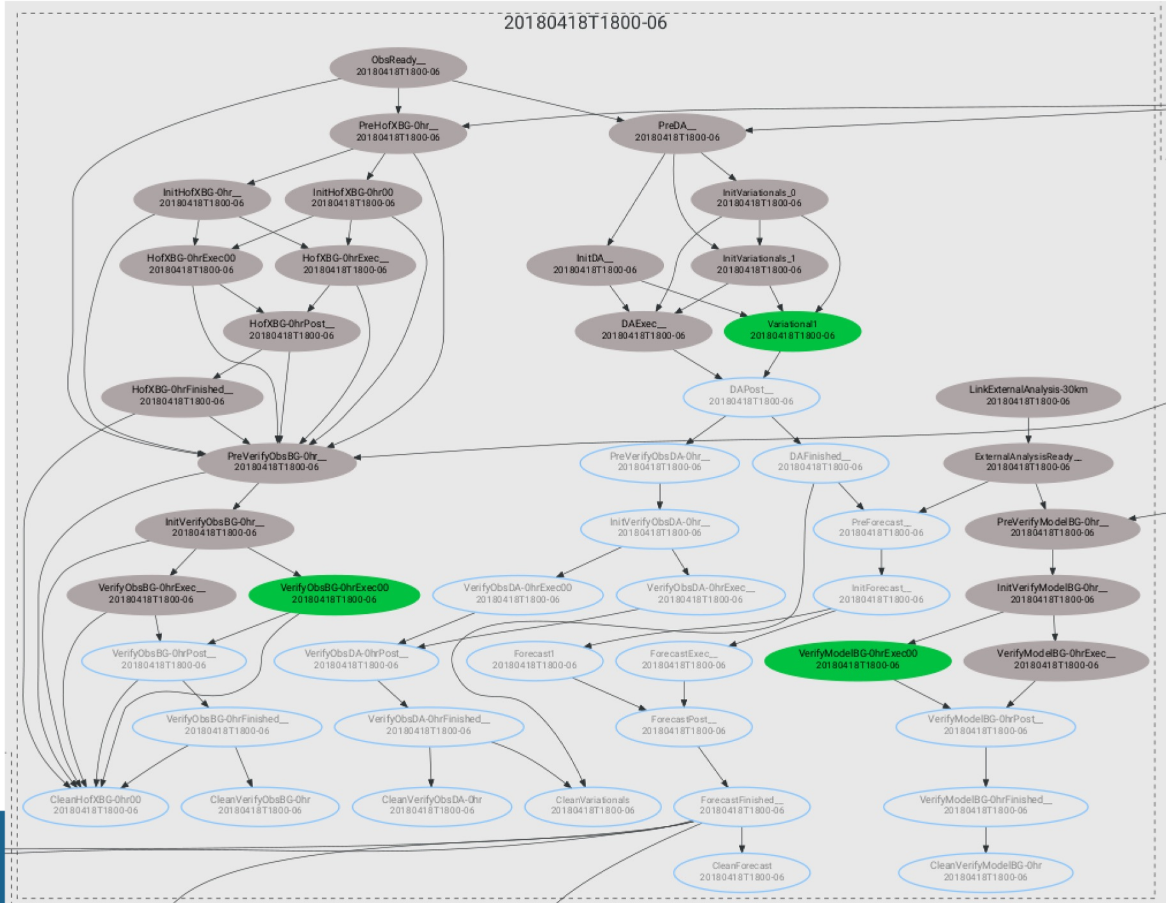
```
├── CyclingDA
│   ├── 2018041500
│   ├── …
│   └── 2018041600
├── CyclingFC
│   ├── 2018041418
│   ├── …
│   └── 2018041600
├── ExtendedFC
│   ├── 2018041500
│   ├── …
│   ├── …
│   └── 2018041600
├── ExternalAnalyses
│   ├── 120km
├── MPAS-Workflow
│   ├── bin
│   ├── config
│   ├── scenarios
│   ├── test
│   └── tools
└── Verification
    ├── bg
    ├── da
    └── fc
```

- CyclingDA, CyclingFC: for cycling DA (analysis + 6-h fcst)
  - CyclingDA → Executes mpasjedi_variational.x with corresponding yaml file
  - CyclingFC → Executes mpas_atmosphere for 6-h with corresponding namelist and stream files
- ExtendedFC: 168-h deterministic forecast → Executes mpas_atmosphere for 168-h with corresponding namelist and stream files
- ExternalAnalyses: For the SST and XICE update at each cycle, and verification of the MPAS 6-h fcst → Link of external GFS analyses converted to MPAS initial conditions format and grid
- MPAS-Workflow: Local copy of the workflow with corresponding changes for this experiment
- Verification: For the MPAS 6-h fcst verification against observation and GFS analyses, and extended fcst against GFS analyses → Runs HofX application and diagnostic scripts

MPAS-Workflow: suites

Example of the CYLC gui interface:

- cycle points
- tasks
- dependencies
- status of the jobs



MPAS-Workflow: tips

For debugging, you have a couple of ways to check what is happening:

1. the CYLC gui interface will tell you the status of each job
2. check if the job is actually submitted by issuing 'qstat -u $USER'
3. check the log file of the application that seems to be submitted/failed/etc
   a. e.g., HofX or DA: you can check the jedi.log/jedi.log.all files in the cycle date)
4. check the CYLC log file in the cylc-run directory (/glade/scratch/<username>/cylc-run)

Useful CYLC line commands:

cylc scan

cylc trigger suitename "*.*:failed"

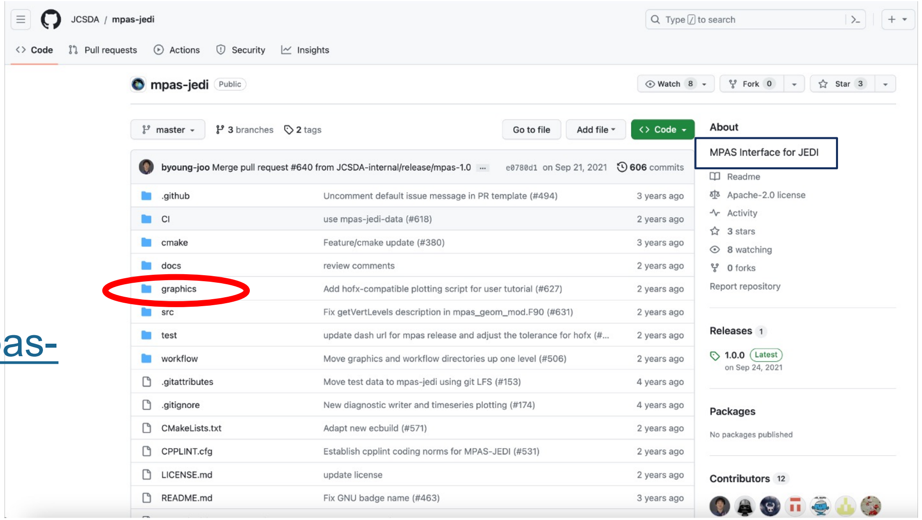
cylc restart --until=final_end_point suitename | add restart point in the scenario and run it

cylc reset -s succeeded suitename *:failed

Graphics package

- Developed at NCAR/MMM to aid in diagnosing results with MPAS and MPAS-JEDI
  - Observation space verification can be used for any JEDI model interface
- Python scripts

➢ Open-source: https://github.com/JCSDA/mpas-jedi/tree/release/2.0.0/graphics



but **NOT** supported

Graphics package: functionalities

- ❑ Produces statistics for selected diagnostics using the `DiagSpaces` selection.
- ❑ Distributed generation of information results in a database of processed statistics, stored in HDF5 files
- ❑ Distributed diagnostic files across multiple experiments, multiple cycle initial times, and multiple forecast lengths
- ❑ Enables portable reading of user-selected variables from multiples types of UFO feedback files (ObsSpace, GeoVaLs, ObsDiagnostics)
- ❑ Supports PBS script to submit verification jobs on Casper and Cheyenne
- ❑ IODA observation convention updates
- ❑ Updated QC flag numbers based on recent changes in UFO
- ❑ Users can select specific observation types, channels and variables to plot

Graphics package: functionalities

DiagSpaces:

Sondes, aircraft, AMV winds, GNSSRO, surface pressure

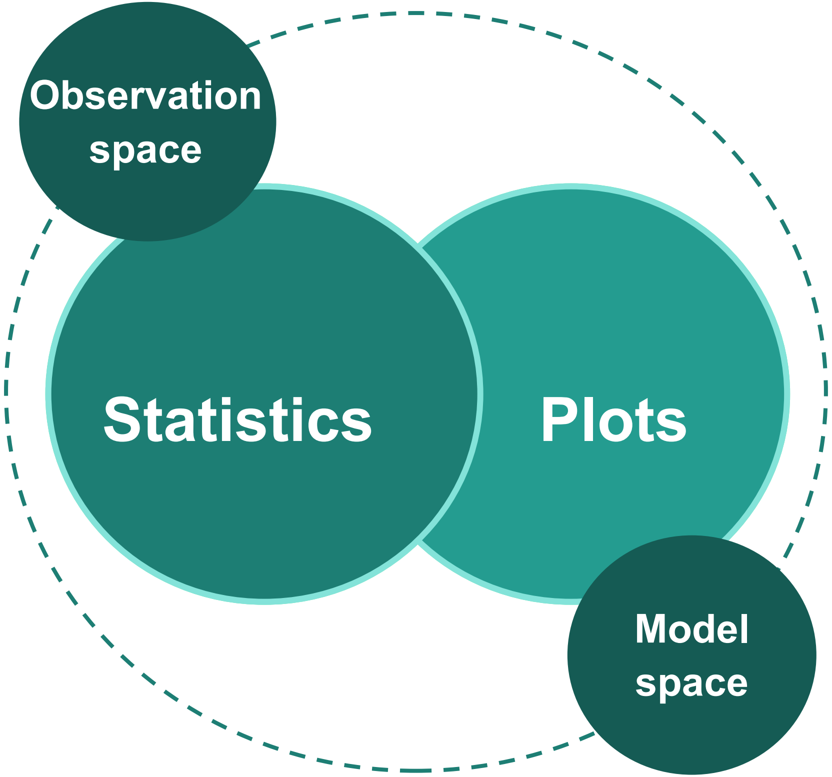
AMSU-A (NOAA-15, NOAA-18, NOAA-19, METOP-A, METOP-B)

MHS (NOAA-18, NOAA-19, METOP-A, METOP-B)

IASI (METOP-A, METOP-B, METOP-C)

ABI (GOES-16)

AHI (Himawari-8)



Analyzed variables:

2m T
2m Q
10m U and V
Ps
T
Theta
rho
W
Ps
U and V
Qv
Qv 1 to 10 model level
Qv 11 to 20 model level
Qv 21 to 30 model level
Qv 31 to 40 model level
QV 41 to 55 model level

Graphics package: functionalities

- **Binning methods:**
  - global
  - by latitude bands: Tro (-30.0, 30.0), NXTro (30.0, 90.0), SXTro (-90.0, -30.0), NMid (30.0, 60.0), SMid (-60.0, -30.0), NPol (60.0, 90.0), SPol (-90.0, -60.0)
  - by tropical latitude bands: ITZC (-5.0, 5.0), STro (-30.0, -5.0), NTro (5.0, 30.0))
  - by cloudiness: clear, mixed-pixels, cloudy, all-sky
  - Latitude vs Pressure 2D
  - Longitude vs Latitude 2D
  - Brightness temperature as a function of cloud fraction 2D

- **Types of plots:**
  - Time series plots with or without confidence intervals calculated using bootstrap resampling
  - profile plots of binned data (e.g., over pressure or latitude on the y-axis) with and without confidence intervals
  - maps of 2D-binned statistics
  - score-card
  - standalone: OmA/OmB diagnostics, observations locations, analysis increments, cost function

Count, Mean, STD, RMS, RMS relative difference

Graphics package: functionalities

Statistics

How to run it?

Observation space:

OmA/OmB

python **DiagnoseObsStatistics.py** -n 36 -p ./dbOut -o obsout -g geoval -d ydiags -app variational -nout 2

Forecast vs observations (HofX)

python **DiagnoseObsStatistics.py** -n 36-p ./dbOut -o obsout -g geoval -d ydiags -app hofx

Model space (vs GFS analysis):

Forecast vs model

python **DiagnoseModelStatistics.py** YYYYMMMDDDHHH -n 36 -r ./x1.655362.init

30km

Graphics package: functionalities

Plots

How to run it?

analyze_config.py: top-level script that controls cycle times and forecast length, verification configuration, experiments and statistics to analyze, and analysis types to apply to the statistics

Observation space:

Carry out analyses for all DiagSpaces that contain "amsua"

python **AnalyzeStats.py** -d amsua

Job-submission examples:

./**SpawnAnalyzeStats.py** -nout 2 -d amsua_,sonde,airc,sfc,gnssro,satw

./**SpawnAnalyzeStats.py** -app hofx -d mhs,amsua,abi_,ahi_,sonde,airc,sfc,gnssro,satw

Model space (vs GFS analysis):

./**SpawnAnalyzeStats.py** -d mpas

Graphics package: examples

Experiments folders structure: ivette_3dvar_OIE120km_WarmStart

```
└── Verification
    ├── bg
    │   ├── mean/YYYYMMMDDDHHH/diagnostic_stats
    ├── da
    │   ├── mean/YYYYMMMDDDHHH/diagnostic_stat
    ├── fc
    │   ├── mean/YYYYMMMDDDHHH/diagnostic_stats
```

```
└── model
    ├── stats_mpas.h5
```

```
└── obs
    ├── stats_da_aircraft.h5
    ├── stats_da_amsua_aqua.h5
    ├── stats_da_amsua_metop-a.h5
    ├── stats_da_amsua_metop-b.h5
    ├── stats_da_amsua_n15.h5
    ├── stats_da_amsua_n18.h5
    ├── stats_da_amsua_n19.h5
    ├── stats_da_gnssrobndropp1d.h5
    ├── stats_da_mhs_metop-a.h5
    ├── stats_da_mhs_metop-b.h5
    ├── stats_da_mhs_n18.h5
    ├── stats_da_mhs_n19.h5
    ├── stats_da_satwind.h5
    ├── stats_da_satwnd.h5
    ├── stats_da_sfc.h5
    ├── stats_da_sondes.h5
```

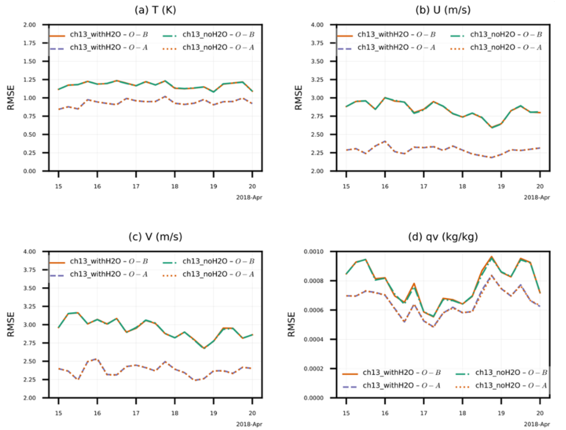
```
├── 0hr
├── 120hr
├── 24hr
├── 48hr
├── 72hr
└── 96hr
    ├── diag.2018-05-16_00.00.00.nc ->
    ├── model
    ├── obs
    └── restart.2018-05-16_00.00.00.nc ->
```

Graphics package: examples

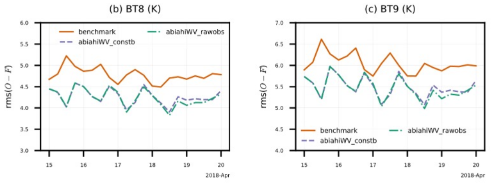
Observation space

DiagSpace_analyses
- BinValAxes2D
- BinValAxisProfileDiffCI
- CYandBinValAxes2D
- CYAxisExpLines

aircraft: OmA/OmB

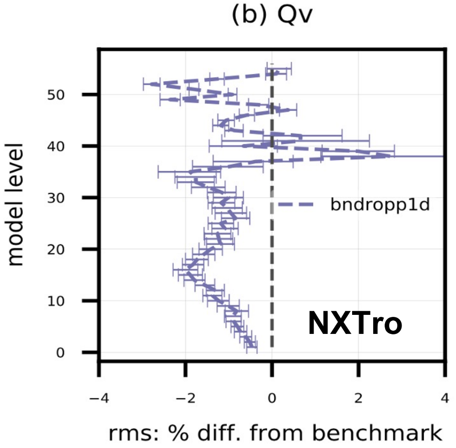


ABI: OmB (HofX)

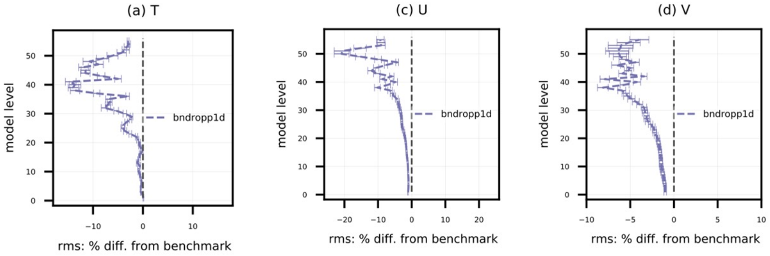


Model space

mpas_analyses
- BinValAxes2D
- BinValAxisProfileDiffCI
- CYandBinValAxes2D
- CYAxisExpLines



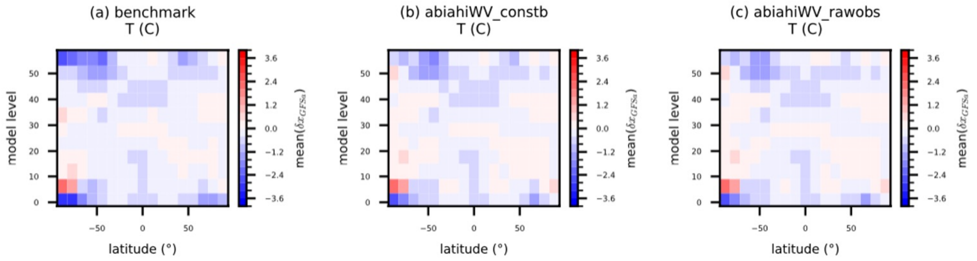
MPAS 6-h verification vs GFS analysis



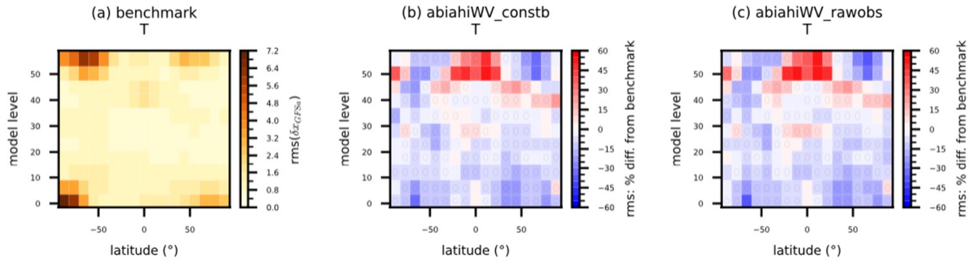
Graphics package: examples

MPAS 6-h verification vs GFS analysis

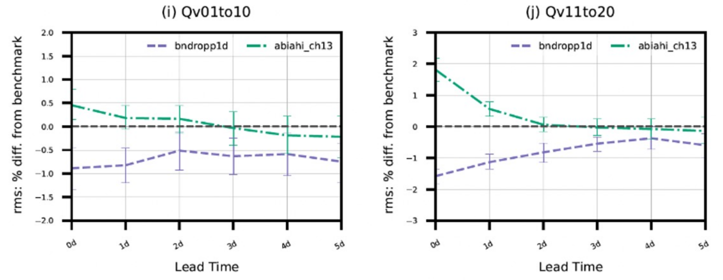
BIAS

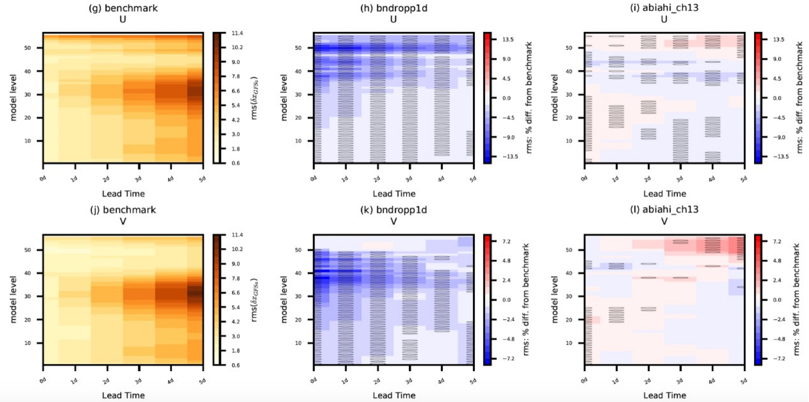


RMSE



MPAS 5-days verification vs GFS analysis





Thank you!

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.