NMRF/TR/04/2020

TECHNICAL REPORT

UMRider – A Parallel Post Processing Utility for Unified Model

Arulalan T.

June 2020

National Centre for Medium Range Weather Forecasting
Ministry of Earth Sciences, Government of India
A-50, Sector-62, NOIDA-201 309, INDIA

UMRider – A Parallel Post Processing Utility for Unified Model

Arulalan T.

June 2020

National Centre for Medium Range Weather Forecasting

Ministry of Earth Sciences

A-50, Sector 62, NOIDA-201309, INDIA

Ministry of Earth Sciences
National Centre for Medium Range Weather Forecasting
Document Control Data Sheet

1	Name of the Institute	National Centre for Medium Range Weather Forecasting
2	Document Number	NMRF/TR/04/2020
3	Date of publication	June 2020
4	Title of the document	UMRider – A Parallel Post Processing Utility for Unified Model
5	Type of Document	Technical Report
6	No. of pages & Figures	51 & 12
7	Number of References	8
8	Author (s)	Arulalan T.
9	Originating Unit	NCMRWF
10	Abstract	Under the Unified Model (UM) Partnership, NCMRWF has been continuously evolving towards a seamless Numerical Weather Prediction (NWP) modelling strategy and is routinely operating different configurations of the NCMRWF Unified Model (NCUM): Global Model (12 km), Global-Ensemble (12 km) for next ten days forecast, and Regional model (4 km, 1.5 km, and 330 m), Regional-Ensemble (4 km) for three days forecast along with ocean and atmospheric data assimilation systems. The UM dumps the forecast outputs in the proprietary file format PP-format (Post Processing Format) or FF-format (Field Files Format), developed by UKMO to increase the rate of writing speed of model data to disk - highly essential to maintain real-time forecast delivery time to the forecasters. However, it is required to convert the model outputs to GRIB, and NetCDF formats (which are standardized (open to the public) by the World Meteorological Organization (WMO) and the University Corporation for Atmospheric Research (UCAR), respectively) for easy visualisation and data manipulation. Keeping in mind the internal users who find it challenging to deal with the UM fields file format for various operational and research-oriented tasks, a parallel post-processing utility, namely UMRider has been developed at NCMRWF. UMRider has been created to post-process the NCUM forecast outputs to open standard file formats (GRIB2 or NetCDF4) at the required spatial and temporal resolution for distribution to research use and other applications. It is written in the Python programming language, aimed to post-process the Unified Model outputs in parallel computing to minimize the processing time which is essential for the NWP centres. UMRider will be beneficial to all the UM partners. This report is a user guide to the UMRider utility, its operational applications, modules usage, and its web graphical user interface named 'UMRiderGUI'. This utility enables the users to generate the UM output files in GRIB2 format (optionally GRIB1, or netCDF4) for use in other user applications and mesoscale models.
11	Security classification	Non-Secure
12	Distribution	Unrestricted Distribution
13	Key Words	UMRider, UMRiderGUI, PostProcess, NCUM-G, NEPS-G, NCUM-R, Grib2, Grib1, netCDF4

Contents

Topic	*Page No.*
Abstract	1
1. Introduction	2
1.1. NCMRWF's Unified Modelling Systems	2
1.1.1. Global Deterministic Unified Model (NCUM-G)	2
1.1.2. Global Ensemble Prediction System (NEPS-G)	2
1.1.3. Regional Unified Model (NCUM-R)	3
1.2. NWP File Formats	3
1.2.1. PP/FF Format	3
1.2.2. NETCDF Format	4
1.2.3. GRIB Format	5
1.3. High Performance Computing Systems (Bhaskara and Mihir)	5
2. UMRider – A Parallel Post Processing Utility	5
2.1. Introduction	5
2.2. Operational Applications	9
2.2.1. NCUM-G Global Deterministic Model Post Processed Products	9
2.2.2. NCUM-R Regional Deterministic Model Post Processed Products	12
2.2.3. NEPS-G Global Ensemble Model Post Processed Products	14
2.2.4. NCUM-G Global Hindcast Post Processed Products	15
2.2.5. IMDAA Regional Reanalysis Post Processed Products	15
2.3. Configurations and Options	15
2.3.1. Installation	15
2.3.2. UMRider Setup Configuration	16
2.3.3. UMRider Vars Configuration	24
2.4. Executions in HPCS	26
2.4.1. bsub jobs – IBM – Bhaskara	26
2.4.2. qsub jobs – CRAY – Mihir	27
2.5. UMRiderGUI – A Web Interface	29
3. UMRider Utilities Usage	37
4. Acknowledgments	40
5. References	41
Appendix	
A. UMRider – Source Code Structure, Access, Future Releases and License	42
B. NCUM STASH and Grib2 Param Codes	46
C. NCUM Grib2 Local Table	50

Abstract

Under the Unified Model (UM) Partnership, NCMRWF has been continuously evolving towards a seamless Numerical Weather Prediction (NWP) modelling strategy and is routinely operating different configurations of the NCMRWF Unified Model (NCUM): Global Model (12 km), Global-Ensemble (12 km) for next ten days forecasts, and Regional model (4 km, 1.5 km, and 330 m), Regional-Ensemble (4 km) for three days forecasts along with ocean and atmospheric data assimilation systems. The UM dumps the forecast outputs in the proprietary file format PP-format (Post Processing Format) or FF-format (Field Files Format), developed by UKMO to increase the rate of writing speed of model data to disk - highly essential to maintain real-time forecast delivery time to the forecasters. However, it is required to convert the model outputs to GRIB, and NetCDF formats (which are standardized (open to the public) by the World Meteorological Organization (WMO) and the University Corporation for Atmospheric Research (UCAR), respectively) for easy visualisation and data manipulation. Keeping in mind the internal users who find it challenging to deal with the UM fields file format for various operational and research-oriented tasks, a parallel post-processing utility, namely UMRider has been developed at NCMRWF. UMRider has been created to post-process the NCUM forecast outputs to open standard file formats (GRIB2 or NetCDF4) at the required spatial and temporal resolution for distribution to research use and other applications. It is written in the Python programming language, aimed to post-process the Unified Model outputs in parallel computing to minimize the processing time which is essential for the NWP centres. UMRider will be beneficial to all the UM partners. This report is a user guide to the UMRider utility, its operational applications, modules usage, and its web graphical user interface named 'UMRiderGUI'. This utility enables the users to generate the UM output files in GRIB2 format (optionally GRIB1, or netCDF4) for use in other user applications and mesoscale models.

1. Introduction

1.1 NCMRWF's Unified Modelling Systems

Modern day weather forecasting is carried out through Numerical Weather Predictions (NWP) by the application of computer models that describe the way the atmosphere evolves using a set of governing equations. NCMRWF uses state of the art global and regional NWP models to predict weather 10 days in advance.

The concept of a Unified Modelling system for seamless prediction of weather and climate has gained importance and acceptance during the last decade, after its first demonstration by the Met Office, UK. NCMRWF is using the latest version of the Unified Model (UM v10.8 deterministic as NCUM-G and ensemble as NEPS-G) at a global horizontal resolution of ~12 km and 80 levels in the vertical and the associated 4D-Var data assimilation system for real-time weather prediction. Additionally, the regional model (NCUM-R) over the Indian subcontinent at 4 km and sub-regional scale at 1.5 km and 330 m horizontal resolutions and 80 vertical levels have also been implemented at NCMRWF. An experimental Regional Ensemble Prediction System (NEPS-R) at 4 km resolution has been implemented for case studies. All four types of UM at NCMRWF (NCUM-G, NEPS-G, NCUM-R and NEPS-R) are being post-processed (convert the model output to grib2 file format) operationally by using UMRider utility, which is an in-house development. This utility's scripts are written in the parallel Python programming language, released under the open-source GNU v3 license.

1.1.1 Global Deterministic Unified Model (NCUM-G)

Under UM Partnership, NCMRWF is running the Unified Model (NCUM) operationally, the source codes of which are available on Met Office Shared Repository Service. This seamless prediction system is used for medium range numerical weather prediction at NCMRWF. The NCUM-G global system was upgraded in May 2018 with the latest UM (version 10.8), with an improved model horizontal resolution of 12 km and science settings of Global Atmosphere (GA6.1). The Observation Processing System, Hybrid 4D-Var data assimilation system, Surface data assimilation/preparation system, Unified Model, and in-house developed Observation Pre-Processing System are the major components of the NCUM system (Sumit Kumar et al., 2018). This end-to-end global numerical weather prediction system routinely produces 10-day forecasts based on 00 and 12 UTC initial conditions, and the model outputs are post-processed to 1-hourly, 3-hourly, 6-hourly, 24-hourly grib2 files using the UMRider utility.

1.1.2 Global Ensemble Prediction System (NEPS-G)

The NCMRWF global Ensemble Prediction System (NEPS-G) was upgraded to 12 km resolution and made operational from 1st June 2018 for the generation of 10-day forecasts at 00 and 12 UTC (Ashu Mamgain et

al., 2018). The NEPS-G global model configuration is based on the recent version UM10.8 of the UK Met Office Global and Regional Ensemble Prediction System (MOGREPS). The Ensemble Transform Kalman Filter (ETKF) method is used to create the initial condition perturbations. The forecast perturbations obtained from 6 hour short forecasts of 11 ensemble members at 00 and 12 UTC are updated by ETKF four times a day at 00, 06, 12, and 18 UTC. The model uncertainties are estimated by the Stochastic Kinetic Energy Backscatter (SKEB) and Random Parameters (RP) schemes. Surface parameters like sea surface temperature, soil moisture content, and soil temperature are also perturbed in the initial condition to remove the deficiency of lack of ensemble spread near the surface. 10-day probabilistic forecasts are issued daily using 23 ensemble members (1 control + 22 perturbed) as grib2 files, which is generated by UMRider post-process utility (Paromita Chakraborty et al., 2019). The NCUM-G deterministic forecasts at both 00 and 12 UTC are considered as control. One set of 11 perturbed members run from 00 UTC of the current day, and another set of 11 perturbed members run from 12 UTC of the previous day to form 22 perturbed ensemble members by merging with proper forecast valid time at 6-hourly grib2 files.

1.1.3 Regional Unified Model (NCUM-R)

NCMRWF high resolution regional convective scale Unified Model with latest version UM10.6, with Regional Atmospheric version 1 Tropical science settings (RA1-T), generates1-hourly forecasts up to next three days at 4 km spatial resolution, operationally. This model uses explicit convection and produces better diurnal cycle of rainfall (Jayakumar et al., 2016 and 2017). UMRider is used to post process of NCUM-R model outputs by converting the spatial grid from rotated to regular, along with time processing in the output grib2 files.

1.2 NWP File Formats

Several file formats (such as pp/ff, netcdf, grib, hdf, binary, bufr, ascii, etc.) are being used to store the meteorological data in Numerical Weather Prediction (NWP) model outputs. Among those formats, UMRider aimed to support reading pp/ff, netcdf, grib1, grib2 formats, and to write in grib2 and netcdf formats, and to convert grib2 to grib1 via cnvgrib external tool. These three file formats details are discussed in this section.

1.2.2 PP/FF Format

The PP-format (Post Processing Format) and FF-format (Fields File Format) are proprietary file formats for meteorological data developed by the Met Office, UK. Simulations of the weather are performed by the Met Office's Unified Model, which can be used for various weather and climate related applications. This data is usually meteorology-specific in nature and may include averaged data for parameters like global surface temperatures or accumulations of rainfall at latitude-longitude locations. However, the Unified Model is capable of generating many sophisticated user-specified diagnostics to FF-format. These files are binary

streams, structured in a proprietary file format that can then be processed and transformed into other, more portable formats like PP format. The main reason for using such a format is to increase the rate at which data can be written from the model to disk, a significant consideration when running a simulation that must be timely and efficient.

1.2.3 NetCDF Format

NetCDF (Network Common Data Form) is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. This NetCDF project is hosted as an open standard file format by the Unidata Program at the University Corporation for Atmospheric Research (UCAR). The NetCDF Classic and 64-bit Offset Format are an international standard of the Open Geospatial Consortium.

The NetCDF project started in 1989 and is still actively supported by UCAR. The original NetCDF binary format (released in 1997, now known as "NetCDF classic format") is still widely used across the world and continues to be fully supported in all NetCDF releases. Version 4.0 (released in 2008) allowed the use of the HDF5 data file format. Version 4.1 (2010) added support for C, FORTRAN, Python client access to specified subsets of remote data via OPeNDAP. Version 4.3.0 (2012) added a CMake build system for Windows builds. Version 4.7.0 (2019) added support for reading Amazon S3 objects. Further, UCAR has plans to release the next versions to improve performance, add features, and fix bugs.

1.2.4 GRIB Format

GRIB (GRIdded Binary or General Regularly-distributed Information in Binary form) is a concise data format (as an open standard) commonly used in meteorology to store weather and climate forecast data. The World Meteorological Organization's Commission standardizes it for Basic Systems, known under number GRIB FM 92-IX, described in WMO Manual on Codes No.306. Currently, there are three versions of GRIB. Version 0 was used to a limited extent by projects such as TOGA and is no longer in operational use. The first edition (grib1, released 1994) is used operationally worldwide by most meteorological centres to store Numerical Weather Prediction (NWP) outputs. A newer generation (grib2, published 2003) has been introduced, known as GRIB second edition, and many operational NWP centres (NCEP, ECMWF, NCMRWF, etc.,) have adopted to this format, for distribution of data to the user community.

1.3 High Performance Computing Systems (Bhaskara and Mihir)

There are two high-performance computing system (HPCS) available at NCMRWF, namely, Bhaskara and Mihir that are used for its operational NWP routine works.

Bhaskara (IBM)

NCMRWF high performance computing (HPC) system named, "Bhaskara" is a 350 TFLOPS IBM iDataPlex Cluster, configured with 16 cores of intel Sandybridge processors clocking at 2.6 GHz, with 64 GB DDR3 1600MH2 RAM per Compute node, was commissioned in June 2015. Bhaskara has 1052 compute nodes and each compute node contains a dual port FDR embedded Infiniband adapter for inter-processor Message Parsing Interface (MPI) Communication. Platform Load Sharing Facility (LSF) is a workload management platform, job scheduler for distributed HPC environments.

Mihir (Cray)

The latest supercomputer at NCMRWF, named "MIHIR," is a Cray-XC40 LC [Liquid Cooled] System with 2320 nodes running Intel Xeon Broadwell E5-26952695 v4 2.1GHz 18C CPU processors with a peak performance of 2,806 TFLOPS and a total system memory of 290TB. Besides, the system consists of 12 Intel KNL 7210 accelerator nodes with 96 GB DDR4, gives a peak performance of 31.92 TFLOPS and a total memory of 1.1TB. The entire system operates on Cray's customized Linux OS, called Cray Linux Environment. Also, it has many parallel libraries like OpenMP, MPI, libsci, Intel Cluster software, etc. The system uses Portable Batch System (PBS) Pro as Workload Manager. Total usable archival space is 16.86PB, configured by the Parallel File System Lustre. The eight utility nodes are dedicated to pre and post data process jobs of HPC users. This HPCS was commissioned in January 2018.

2. UMRider – A Parallel Post Processing Utility

2.1 Introduction

UMRider is a parallel post-processing utility written in python open-source scientific modern programming language; the utility is created as in-house development at NCMRWF for the conversion of the Unified Model (UM) with Global, Ensemble, Regional specific outputs from fields file (ff) or post-processed (pp) file formats to grib2 or grib1 or NetCDF file format, built based on 'Iris' python library (Iris is a powerful, format-agnostic, community-driven Python library for analysing and visualising Earth science data). Through Storage Handling and Diagnostic System (STASH- Spatial and Temporal Averaging and Storage Handling) code of Unified Model and climate forecast standard name (cf_standard_name) of meteorological variable, it extracts and converts to the WMO-NCEP standard grib2 by setting correct Parameter Code (Discipline, Category, Number, type Of First Fixed Surface) and followed by creation Grads Control File by using g2ctl.pl scripts. Additionally, it has the option to convert the model outputs to grib1 and NetCDF file

formats. This utility is named as "**UMRider**" to indicate that the scripts are aimed to post-process the unified model outputs in parallel computing to minimize the processing time.

A customised Unified Model (v7.7) with a 4D var data assimilation and forecast was implemented at NCMRWF (Rajagopal et al., 2012) during 2012 to predict the weather over global and regional (Indian subcontinent) scales. The model outputs were visualized by another utility called 'xconv' and sliced, or regridded by a script named 'subset.ctl'. However, for archival and data exchanges to the external users and agencies, there is an operational requirement to convert the model outputs from fields file into a standard data format like 'GRIB' or CF standard 'NetCDF'. Additionally there is a requirement of GRIB formatted data for feeding into other applications and mesoscale models. Met Office provided UM utilities to convert the model output in files format to pp format and for basic functionalities. Mohandas (2014) customised it as per the need of NCMRWF operational usage and developed a UM to grib1 conversion utility named as 'umfld2grib.sh'.

During 2015, the global UM at NCMRWF was upgraded to version 10.6 with 17 km horizontal resolution which required additional computing resources. At that same time Grib version 2 became popular among many atmospheric and oceanic model applications, which forced us to think for an alternate solution to convert the model outputs to grib2 standard with maximum speed by utilizing parallel processors of the available High Performance Computing System (HPCS). UK Met Office developed a Python library named 'Iris,' which handles the UM model fields file outputs for visualization, regridding, store into pp, grib2, netcdf formats and under 'SciTools' multi-project repositories which is a collaborative place to produce powerful Python-based open-source tools for Earth scientists. The utility 'umfld2grib.sh' (Mohandas, 2014) to post-process UM outputs was replaced by the UMRider after its initial development in 2016. Since then, it has been used for all significant post-processing operational applications (more than 20 different jobs, see section 2.2) at NCMRWF. The NCMRWF's Unified Model (NCUM) outputs are being post post-processed by UMRider, which handles customised Spatio-temporal resolution, interpolation, regridding, processing of rotated grid and hybrid level coordinates, and storing into grib2 format. UMRider produces the NCUM post-processed grib2 files operationally, which are being used by many other atmospheric-oceanic research and operational institutes (both national and international) such as IMD, INCOIS, SAC-ISRO, IITM, Navy, NIWE, GSI, UK Met Office etc., and the THORPEX Interactive Grand Global Ensemble (TIGGE) project hosted at ECMWF.

NCUM-G Global Deterministic model produce the following analysis/6-hours short forecasts files namely, qwqg00.pp0, umglca_pb, umglca_pd, umglca_pe, umglca_pf, umglca_pi and long forecast files up to forecast lead time of 240 hours, namely, umglaa_pb???, umglaa_pd???, umglaa_pe???, umglca_pf???, umglca_pi???, where '???' replaces corresponding 24 hourly time intervals like 024, 048, ..., 240. (These file names are specific to NCMRWF's operational runs and can be different for other UM partner

organizations). All the long forecast files contain either twenty-four 1-hourly data or eight 3-hourly data. UMRider supports to create either 1-hourly or 3-hourly or 6-hourly (either 6th hour instantaneous or average/accumulation of two 3-hourly data or six 1-hourly data) or 24-hourly (either 24th hour instantaneous or average/accumulation of eight 3-hourly data/twenty-four 1-hourly data) grib2 files. Similarly, UM regional and ensemble model has been configured to produce meteorological parameters in many different output units (file names) of their long forecast. Users will be able to create 1-hourly, 3-hourly, 6-hourly, 24-hourly,and at various spatial resolutions post-processed grib2 files of NCUM-G, NEPS-G, NCUM-R models by controlling UMRider utility via customizing two configuration text files (setup.cfg and vars.cfg).

For example, if UMRider has been configured for 6-hourly grib2 files, then

- It creates four analysis 6-hourly files (at 00, 06, 12, 18 UTC cycles)
- It creates 40 forecast 6-hourly files (000, 006, 012, ..., 234, 240 during 00 and 12 UTC cycles)
- Finally, it creates ctl, idx files for all 44 grib2 files by using g2ctl.pl
- Also, all 44 grib2 files variables are in the same order (defined by the user)

Users can install the UMRider in any Linux based operating system (see section 2.3.1), and they are allowed to control the UMRider by modifying two configuration text files namely setup.cfg, and vars.cfg. In the current version of UMRider (v3.1.1), users can manage the following 38 arguments of utility by creating and editing **setup.cfg** file and followed by export the setup.cfg file path into their shell environment variable called **UMRIDER_SETUP**.

The options available in **setup.cfg** are listed below:

1. startdate or UMRIDER_STARTDATE
2. enddate or UMRIDER_ENDDATE
3. UMtype
4. UMInAnlFiles
5. UMInShortFcstFiles
6. UMInLongFcstFiles
7. inPath
8. outPath
9. tmpPath
10. anl_step_hour
11. anl_aavars_reference_time
12. anl_aavars_time_bounds
13. fcst_step_hour
14. start_long_fcst_hour
15. end_long_fcst_hour_at_00z
16. end_long_fcst_hour_at_12z
17. latitude
18. longitude
19. targetGridResolution
20. targetGridFile
21. extraPolateMethod
22. pressureLevels
23. fillFullyMaskedVars

24. soilFirstSecondFixedSurfaceUnit
25. loadg2utils
26. overwriteFiles
27. anlOutGrib2FilesNameStructure
28. fcstOutGrib2FilesNameStructure
29. createCtlIdxFiles
30. convertGrib2FilestoGrib1Files
31. grib1FilesNameSuffix
32. removeGrib2FilesAfterGrib1FilesCreated
33. createGrib1CtlIdxFiles
34. debug
35. setGrib2TableParameters
36. wgrib2Arguments
37. wgrib2netcdf
38. callBackScript

The detailed description of the above 38 options are discussed in section 2.3.2. There is another environment variable named **UMRIDER_VARS,** which lets the user to control the variables of their choice by changing **vars.cfg** configuration text file, which is discussed in detail in section 2.3.3. This vars.cfg configure file is used for um2grib2 conversion of only needed NCUM (NCUM-G, NEPS-G, NCUM-R) models output variables. User-defined variables and order of variables will be retained in the output Grib2 files (but pressure level variables comes first followed by non-pressure level variables), same as vars.cfg configuration file. Furthermore, the users will be able to control the variables and frequency of output as 1-hourly, or 3-hourly, or 6-hourly, or 24-hourly forecast Grib2 or Grib1 or NetCDF files.

As mentioned in section 1.3, there are two HPCS named Bhaskara (IBM) and Mihir (Cray) machines at NCMRWF, which uses BSUB (see section 2.4.1) and QSUB (see section 2.4.2) method for submitting UMRider jobs into compute nodes. The NCMRWF's Grib2 files, all of which follow the WMO-NCEP standard Grib2 param codes to match with the STASH codes of variables (see Appendix – B), but few of the variables of NCUM do not have param codes in the existing WMO-NCEP standard table. So UMRider had to create an NCMRWF's local table (see Appendix – C) to assign Grib2 local param codes to those missing variables in the NCEP Grib2 table, which also assigned the centre code as 29 (belongs to Delhi, India also pointing to NCMRWF, India).

Due to the increasing number of customization in operational jobs, it became necessary to make 38 options in UMRider, which also encouraged to make platform independent web browser based graphical user interface named as UMRiderGUI (discussed detailed in section 2.5), to make an easy interface to create and edit the setup.cfg (section 2.3.2) and vars.cfg (section 2.3.3), BSUB bash scripts (section 2.4.1) and also submit the UMRider jobs to the compute nodes. Finally, section 3 explains the UMRider utility usage (python library for the developers) and also discusses the Source Code Structure, Access, Future Releases and License in Appendix – A.

2.2 Operational Applications

UMRider v1.0.0 was implemented as its first operational post process utility for Unified Model v10.6 (Global at 17 km, Ensemble at 33 km, Regional at 4 km) in Bhaskara HPCS during Dec 2015, During January 2016 to June 2018, UMRider was developed further as per clients need, bug fixes, improved parallel operational performance and released up to 13 versions. In early 2018, NCMRWF procured new Cray HPCS, named as Mihir and moved all operational numerical weather model analysis and forecast applications with upgraded and highest horizontal resolution (0.12°x0.18°) for global model, and other models. Subsequently, UMRider v3.1.1 was implemented as an operational post processing utility of analysis and forecast outputs of Unified Model v10.8 (Global at 12 km, Ensemble at 12 km, Regional at 4km, 1.5 km, and 330 m) in Mihir HPCS and Regional Reanalysis at 12 km in Bhaskara HPCS. The list of operational post processing jobs on Mihir HPCS and their purposes are given below.

2.2.1 NCUM-G Global Deterministic Model Post Processed Products

Using the global deterministic model outputs, there are several post processed jobs that are being run operationally. To run the several different products, we do not require to duplicate the source code individually. Instead we can keep UMRider script at common location path in server, and make individual setup files dedicated for the respective jobs.

Common Script Path: **/home/umfcst/smallapps/UMRider/UMRRun/**

2.2.1.1 Global Post Processed Products

The NCUM-G global deterministic model's post processed outputs produces at 0.12°x0.18° and 0.5°x0.5° resolutions -grib2 files at 6-hourly time interval forecasts from 6 to 240 hour based on both 00 and 12 UTC initial condition along with analysis files for every day. Post processed output contains 72 variables generated from model outputs.

1) Resolution: 0.12°x0.18°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_post

Output path: /home/umfcst/NCUM_output/post

2) Resolution: 0.5°x0.5°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_post_0p5

Output path: /scratch/umfcst/NCUM_output/post_0.5/

2.2.1.2 Indian Region Post Processed Products

The Indian region is extracted from NCUM-G global deterministic model and is used to produce post processed outputs at both 0.125° and 0.25° resolution grib2 files of 6-hourly forecast from 6 to 240 hour and analysis files based on both 00 and 12 UTC initial conditions for every day. This product is used by

India Meteorological Department (IMD) and SASE, Chandigarh to run their WRF models. It has 72 post processed variables generated from model outputs.

1) Resolution: 0.125°x 0.125°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_india_reg_from_global/0p125

Output path: /home/umfcst/ShortJobs/IndRegion/0p125

2) Resolution: 0.25°x0.25°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_india_reg_from_global/0p25

Output path: /home/umfcst/ShortJobs/IndRegion/0p25.

2.2.1.3 OSF Model Input Products

Ocean State Forecast (OSF) systems at INCOIS, Hyderabad and Indian Navy uses NCUM-G global deterministic model analysis and forecast outputs 6-hourly grib2 files as atmospheric initial conditions. There are a total of 15 post processed variables generated from model outputs.

1) Resolution: 0.125°x 0.125°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_osf_input/0p125

Output path: /home/umfcst/ShortJobs/OSF/0p125/

2) Resolution: 0.25°x 0.25°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_osf_input/0p25

Output path: /home/umfcst/ShortJobs/OSF/0p25/

3) Resolution: 0.5°x 0.5° (Grib1)

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_osf_input/0p5

Output path: /home/umfcst/ShortJobs/OSF/0p5/

2.2.1.4 HYCOM Model Input Products

The HYbrid Coordinate Ocean Model (HYCOM) operational at INCOIS, Hyderabad uses NCUM-G global deterministic model analysis and forecast outputs (3-hourly grib2 files) as atmospheric initial conditions. A total of 8 post processed variables are generated from model outputs.

Resolution: 0.125°x 0.125°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_hycom_input

Output path: /home/umfcst/ShortJobs/Hycom/0p25/

2.2.1.5 HYSPLIT Trajectory Model Input Products

The Hybrid Single Particle Lagrangian Integrated Trajectory Model (HYSPLIT) operational at NCMRWF uses NCUM-G global deterministic model analysis and forecast outputs (6-hourly grib1 files) as atmospheric input. A total of 13 post processed variables are generated from model outputs.

Resolution: 0.5°x 0.5° (Grib1)

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_hysplit_input
Output path: /home/umfcst/ShortJobs/NCUM_HYSPLIT/0p5/

2.2.1.6 IMD-MFI Model Input Products

The Indian Meteorological Department (IMD) uses the Synergy system from Toulouse-based French company Meteo France International (MFI), which uses NCUM-G global deterministic model analysis, and forecast outputs (6-hourly grib2 files) along with other model products. A total of 72 post processed variables are generated from model outputs.

Resolution: 0.5°x 0.5°
Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_imd_mfi_input
Output path: /home/umfcst/ShortJobs/IMD-MFI/0p5/

2.2.1.7 Solar Energy Input Products

The National Institute of Solar Energy uses NCUM-G global deterministic model analysis and forecast outputs (1-hourly grib2 files) as atmospheric initial conditions to forecast solar energy power production. There are 3 Post-processed generated from model outputs.

Resolution: 0.5°x 0.5°
Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_solar_energy_input
Output path:/home/umfcst/ShortJobs/NCUM_SOLAR_ENERGY/0.25

2.2.1.8 Solar and Wind Energy Input Products

The National Institute of Wind Energy (NIWE) uses NCUM-G global deterministic model analysis and forecast outputs (1-hourly grib2 files) as atmospheric initial conditions to forecast solar and wind renewable energy power production. 14 Post processed variables are extracted from model outputs.

Resolution: 0.125°x 0.125°
Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_solar_wind_energy_input
Output path: /home/umfcst/ShortJobs/NCUM_SOLAR_WIND_ENERGY/0.125

2.2.1.9 VSDB Input Products

At NCMRWF, the Verification Statistics Data Base (VSDB) is used to verify the multi global numerical weather models forecast against its own analysis. In which one of the model inputs feed is from NCUM-G global deterministic model analysis and forecast outputs (6-hourly grib1 files) at 3 different resolutions. Post processed 6 variables from model outputs.

1) Resolution: 1.5°x 1.5°
Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_vsdb_input/1p5x1p5
Output path: /home/umfcst/ShortJobs/VSDB_Input/1p5x1p5

2) Resolution: 1°x 1°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_vsdb_input/1x1

Output path: /home/umfcst/ShortJobs/VSDB_Input/1x1

3) Resolution: 2.5°x 2.5°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_vsdb_input/2p5

Output path: /home/umfcst/ShortJobs/VSDB_Input/2p5

2.2.1.10 Meteogram Input Products

A meteogram is a graphical presentation of one or more meteorological variables with respect to time, whether observed or forecast, for a particular location. NCMRWF produces meteogram charts over all major districts of India by using NCUM-G global deterministic model analysis and forecast outputs (1-hourly grib1 files). 8 Post processed variables are extracted for the district locations from model outputs.

Resolution: 0.125°x 0.125°

Setup path: /home/umfcst/ShortJobs/NCUM_Meteogram_Input/0.125/

Output path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_meteogram_input

2.2.1.11 Precipitation Forecast at 03-03 UTC Product

The Indian Meteorological Department (IMD) observes 24 hourly accumulated rainfall at many stations over India by using rain gauges, and NCMRWF produces merged rainfall observed product by merging the satellite and rain gauge observed rainfall. The NCUM-G global model provides forecast up to the next 10 days based on the initial conditions at 00 and 12Z. To verify the model forecast with observed rainfall, we must compute a 24-hourly accumulated rainfall forecast valid at every 03 UTC for the next 10 days. i.e., 03-27 forecast hours will be treated as Day-1 accumulated rainfall, and similarly, UMRider produces accumulated rainfall valid at 03 UTC for all other days (Day-2 to Day-10)

Resolution: 0.125°x 0.125°

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_india_rain03Z

Output path: /home/umfcst/ShortJobs/IndRegion/NativeResolution/

2.2.2 NCUM-R Regional Deterministic Model Post Processed Products

Regional model forecast outputs are produced at rotated grids and UMRider converts it into regular grids while doing post process. Under the regional deterministic model outputs there are several post processed jobs are being run operationally at NCMRWF. To execute the different products, we do not require to duplicate the source code individually. Instead we can keep UMRider script at common location path in server, and make individual setup files dedicated for the respective jobs.

Common Script Path: /home/umfcst/smallapps/UMRider/UMRRun/

2.2.2.1 Regional Post Processed Products

The NCUM-R regional deterministic model post processed outputs produced at 0.04°x0.04° (4km) resolution as grib2 files over Indian with 1-hourly time interval forecast from 1 to 75 hour based on 00 UTC initial condition along with analysis files for every day. Post processed 72 variables from model outputs.

1) Resolution: 0.04°x0.04° (4 km)

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_regional_post

Output path: /home/umreg/4km_output/post/0.04/

2) Resolution: 0.015°x0.015° (1.5 km)

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_regional_post/1p5km

3) Resolution: 0.0036°x0.0036° (330 m)

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_regional_post/330m

Both 1.5 km and 330 m model runs only over specific sub regions of India (say over Delhi or Mumbai) and only during severe weather conditions.

2.2.2.2 Regional Wind Energy Input Products

The National Institute of Wind Energy (NIWE) uses NCUM-R regional deterministic model analysis and forecast outputs (1-hourly grib2 files) as atmospheric initial conditions to forecast solar and wind renewable energy for electricity power production. Post processed 5 variables from model outputs.

Resolution: 0.04°x0.04° (4 km)

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_regional_wind_energy_input

Output path: /home/umreg/ShortJobs/NCUMReg_WIND_ENERGY/0.04/

2.2.2.3 Lightning Prediction Product

IMD uses 1-hourly lightning variable grib2 files of the NCUM-R (4 km) model.

Resolution: 0.04°x0.04° (4 km)

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_regional_lightning

Output path: /home/umreg/ShortJobs/NCUM_LIGHTNING

2.2.2.4 Regional Global Landslip Prediction Input Product

The Geological Survey of India (GSI) collaborated with UK Met Office, to do a project to predict the landslides which uses 1-hourly precipitation files of NCUM-R (4 km) and NCUM-G (12 km) models over two pilot regions (Darjeeling, Nilgiris) in India.

Setup path: /home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_regional_global_landslip

Output path: /home/umreg/ShortJobs/NCUM_LANDSLIP/0.04,

/home/umreg/ShortJobs/NCUM_LANDSLIP/0.125

2.2.2.5 Hybrid Model Levels to Height AGL Conversion Products

NCUM-R regional model (4 km) on hybrid vertical levels needs to be converted to height above ground level (metre unit). 7 variables are post processed from model outputs.

Setup path:

/home/umfcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_regional_model_level_2_heights_above_ground

Output path: /home/umreg/ShortJobs/NCUMReg_MDL2Heights_AGL/0.04

2.2.3 NEPS-G Global Ensemble Model Post Processed Products

Under the ensemble model outputs there are several post processing jobs which are being run operationally at NCMRWF. To execute the different products, we do not require to duplicate the source code individually. Instead we can keep UMRider script at common location path in server and make individual setup files dedicated for the respective jobs.

Common Script Path: /home/umeps/NEPS_PostProductions/ShortJobs/scripts/UMRider/UMRRun/

2.2.3.1 Global Ensemble Post Processed Products

The NEPS-G Global Ensemble model outputs are being converted to 6-hourly grib2 files with merged members (current day's 00 UTC 11 perturbed members and previous day's 12 UTC 11 perturbed members) along with deterministic model as control member. Finally, the grib2 files have 23 members at each 6-hourly forecast up to 240 hours.

Resolution: 0.12°x 0.18°

Setup path:

/home/umeps/NEPS_PostProductions/ShortJobs/scripts/UMRider/UMRRun/qsubScripts/ncumeps_global_post

Output path: /home/umeps/NEPS_PostProductions/long_fcst/post

2.2.3.2 Global Ensemble TIGGE Products

The European Centre for Medium Range Weather Forecasts (ECMWF) hosts the THORPEX International Grand Global Ensemble (TIGGE) project, which collects all the International NWP centres ensemble model outputs and archives it for free public research use. NCMRWF is also contributing to the TIGGE project since August 2017. The post-processing tool generates 1 control and 11 perturbed member grib2.gz files at both 00 and 12 UTC daily.

Resolution: 0.12°x 0.18°

Setup path:

/home/umeps/NEPS_PostProductions/ShortJobs/scripts/UMRider/UMRRun/qsubScripts/ncumeps_global_tigge

Output path: /home/umeps/NEPS_PostProductions/ShortJobs/outputs/NCUM_EPS_TIGGE/TarFiles

2.2.4 NCUM-G Global Hindcast Post Processed Products

NCMRWF produced hindcast outputs by re-running the highest resolution (12 km) global NCUM-G (v10.8) from 2015 to 2018. Using the post-processing tool, 1-hourly and 6-hourly reforecast grib2 files up to 240 hours are generated from model reforecast outputs. These hindcast data is used to generate model climatology to identify the extreme events and bias correction of the current forecast model outputs.

Resolution: 0.12°x 0.18°

Setup path: /home/umhcst/smallapps/UMRider/UMRRun/qsubScripts/ncum_global_post

Output path: /home/umhcst/NCUM_output/post_0.12x0.18/

2.2.5 IMDAA Regional Reanalysis Post Processed Products

NCMRWF released India's first high resolution (12km) regional reanalysis for the 40 year period (1979 to 2018). The 1-hourly grib2 files were generated from 00, 06, 12, and 18 UTC daily analysis. The IMDAA project model outputs and its post-processed grib2 files were generated in the Bhaskara HPCS.

Resolution: 0.12°x 0.12°

Script path: /gpfs2/home/umdas/UMRider

Setup path: /gpfs2/home/umdas/imdaa_post

Bhaskara Output path: /gpfs5/home/imdaa/imdaa_Grib

Mihir Output path: /home/moum/IMDAA/imdaa_Grib/

Data Public Access Link: https://www.ncmrwf.gov.in/data/

2.3 Configurations and Options

2.3.1 Installation

The following softwares are pre-requisites for the installation of UMRider:

- Linux OS
- Python 2.7.9 and following libraries to be installed python environment
    - iris 1.13.0
    - numpy 1.14.0
    - scipy 0.10
    - cartopy 0.11.0
    - biggus 0.14
    - gdal 1.9.1
    - grib-api 1.9.16
    - scipy 0.10
    - netcdf4-python 0.9.9
    - cf_units 1.0
    - udunits2 2.1.24
    - mo_pack 0.1.0dev0
    - shapely 1.2.14
    - multiprocessing 0.70a1
    - stratify 0.1
- wgrib2 v2.0.7 (to compress and tweak grib2 files)

- cnvgrib v1.4.1 (to convert grib2 to grib1)
- g2ctl.pl v0.1.1 (to make grib2 control file)
- grib2ctl.pl v0.9.13 (to make grib1 control file)
- gribmapv2.1.1.b0 (comes with grads software, to make grib2 index file)

The above softwares are essential to run UMRider utility. User can download the latest version of UMRider from https://github.com/NCMRWF/UMRider and extract it.

$ cd UMRider_version

$ **sudo python setup.py build install**

The above command will install the UMRider in global system Python environment. It requires sudo access, if user does not have sudo or root permission, then execute below command to install at user level.

$ **python setup.py build install --user**

The above command will install the UMRider in local user Python environment, and not requires sudo or root privilege. Also, user can execute the UMRider scripts without installing it in system python or user area python environment.

The development of UMRider started in late 2015 and was operationalized by January 2016 at NCMRWF. During that time Iris python library was supported by only python version 2.7, not 3 and above. So UMRider is available only in python version 2.7 only and there are no requirements from the end user to upgrade to python version 3.
For download and installation of Python-Iris and other required libraries visit the documentation link: https://scitools.org.uk/iris/docs/v1.13.0/installing.html

Also, before installing Iris, one needs to download the source code of Python-Iris library from the above link, also download modified Iris grib python scripts from this link, then copy into Python-Iris source path https://github.com/NCMRWF/UMRider/tree/master/others/ncmrwfIRIS, and finally install the updated iris library via "python setup.py build install" command.

2.3.2 UMRider Setup Configuration

Setup configure file: Used to setup indata path, outdata path, temporary path to run the um2grb2.py python parallel scripts which will create analysis and forecast files.

NCUM-G GLOBAL MODEL POST PROCESSING SETUP CONFIGURE FILE

Filename: **setup.cfg** Comment Line: Line begins with **#** symbol HPCS: **Mihir**-CRAY

Author: Arulalan <arulalan@ncmrwf.gov.in>Total Number of Options: 38

################UMRider -User Defined Frequent Change Arguments Begins #################

1. ***startdate*** - By default *startdate* takes argument as YYYYMMDD (which means it assume today's date). But user can specify the different *startdate* by following the same format.
   For e.g., if startdate = 20151209 then it will execute the scripts for 09 December 2015,
   If startdate = YYYYMMDD, then it will execute the scripts for today's date.

   Note: However, **UMRIDER_STARTDATE** environment variable will override this *startdate* option.

   startdate= YYYYMMDD

2. ***enddate*** - By default *enddate* is None. User can specify different enddate (but >*startdate*) i.e., for only specified date/one date, user needs to control only in the *startdate* and *enddate* must be None. If user want to execute the um2grb2.py conversion for the range of dates, then user need to set the lower *startdate* end higher *enddate*. An *enddate* could be even YYYYMMDD, but make sure that *startdate* is lower than *enddate*.

   For e.g., startdate = 20151209 and enddate = 20160114, then um2grb2.py conversion program executes from 09-Dec-2015 to 14-Jan-2016.

   Note: However, **UMRIDER_ENDDATE** environment variable will override this enddate option.

   enddate = None

################ UMRider - User Defined Frequent Change Arguments End ##################

############ UMRider - User Defined One Time Setup Configuration Options Begin ############

3. *UMtype* - UM model type takes either *'global'* or *'regional'* or *'ensemble'*. By default it takes *'global'* as argument.

   UMtype = global

4. *UMInAnlFiles* - takes list of absolute filenames which has correct analysis fieldsfile/pp file of UM model valid at 00UTC. None option will take hardcoded proper infile names.

   UMInAnlFiles = ['qwqg00.pp0']

5. *UMInShortFcstFiles* - takes list of partial filenames which has short forecast (kind of analysis) fieldsfile/pp files of UM model produced at 00, 06, 12 and 18 UTC. None option will take hardcoded proper infile names.

   UMInShortFcstFiles = ['umglc.pp0', 'umglca_pb', 'umglca_pd', 'umglca_pe', 'umglca_pf', 'umglca_pi']

6. *UMInLongFcstFiles* - takes list of partial filenames which has long forecast fieldsfile/pp file of UM model based on reference time at 00 &12 UTC. None option will take hardcoded proper infile names.

   UMInLongFcstFiles = ['umglaa_pb','umglaa_pd', 'umglaa_pe', 'umglaa_pf', 'umglaa_pi']

7. *inPath* - model pp filedsfiles/pp files directory absolute path.

YYYYMMDD - forecast reference date

ZZ - forecast reference UTC hour

i.e., *YYYYMMDD* will be replaced with the *startdate* (actual date) and similarly *ZZ* will be replaced with 00 (or 06, 12, 18) UTC.

inPath = /home/umfcst/NCUM_output/fcst/*YYYYMMDD*/*ZZ*/

8. *outPath* - model grib2 files path. *YYYYMMDD*- will be replaced with *startdate* (actual date) string and respective datestamp folder will be created.

   outPath = /home/umfcst/NCUM_output/post_0.12x0.18/*YYYYMMDD*/

9. *tmpPath* - working directory (used to create temporary log files)

   tmpPath = /home/umfcst/ShortJobs/logs/UMRiderLogs/post

10. *anl_step_hour* - analysis step/interval hours. By default it takes 6 hour which means um2grb2.py produces 6 hourly instantaneous or 6 hourly average or 6 hourly accumulation values analysis files. If user specified 3 then it will extract only 3 hourly instantaneous fields. By default model produced 3 hourly average/accumulation.

    anl_step_hour = 6

11. *anl_aavars_reference_time* - takes either *'analysis'* or *'shortforecast'*. When some variables are taken from previous cycle short-forecast (average/accumulation)variables, the reference time need to be set as either current *'analysis'* reference cycle (UTC) or previous cycle's *'shortforecast'* reference time. The *'shortforecast'* option gives exactly based on at which UTC that variable has been processed, whereas *'analysis'* shifts reference time UTC as actual analysis UTC time.

    Note: This option is applicable only to average/accumulation variables in the out analysis grib2 files. Let's keep default option (i.e., shortforecast)

    anl_aavars_reference_time = shortforecast

12. *anl_aavars_time_bounds* - takes either *'True'* or *'False'*. By default, *True* option keeps the analysis time bounds, reference time bounds and *False* option removes it (so that it becomes instantaneous instead of average/accumulation variables). *False* option will be applicable only if *anl_aavars_reference_time* set as *'analysis'*. (i.e., *anl_aavars_reference_time = analysis*)

    Note: This option is applicable only to average/accumulation variables in the out analysis grib2 files.

    anl_aavars_time_bounds = True

13. *fcst_step_hour* -long forecast step/interval hours. By default it takes 6 hour which means um2grb2.py produces 6 hourly instantaneous or 6 hourly average or 6 hourly accumulation values. If user specifies it as 3 or 24 then it will extract only 3 or 24 hourly instantaneous fields and for calculating the average/accumulation for 24 hour. By default model produced 3 hourly average/accumulation.

    Note1: The average and accumulation supports only for either 3 or 6 or 24 hours!!!

Note2: If *fcst_step_hour = 24*, then precipitation_amount variable has been chosen in vars.cfg (see section 2.3.3) and set *'umglaa_pe'* as one of input files in *UMInLongFcstFiles* option, then the final post processed grib2 outfile contains precipitation from 03-03Z (UTC), not from 00-00Z. This is useful to compare with Indian Gauged Observed Rainfall.

fcst_step_hour = 6

14. *start_long_fcst_hour* - long forecast start hour. By default it takes 6 hour which means um2grb2.py produces grib2 files from 06th hour forecasts. If user wants to start from a different hour, then they can specify it! It should be multiples of *'fcst_step_hour'* (see above option)!

    start_long_fcst_hour = 6

15. *end_long_fcst_hour_at_00z* - maximum long forecast hours at 00 UTC cycle produced by NCUM-G model for 10days forecast – i.e., up to 240 hours (by default 240 hours).

    end_long_fcst_hour_at_00z = 240

16. *end_long_fcst_hour_at_12z* – maximum long forecast hours at 12 UTC cycle produced by NCUM-G model for 10 days forecast – i.e. up to 240 hours (by default 240 hours). User can decide what could be the end long forecast hour for post processed out files.

    end_long_fcst_hour_at_12z = 240

17. *latitude* - required latitude which user wants to extract from the model global data. By default it takes *None* (i.e., extract model global latitudes). User can specify their required *latitude* in tuple. For e.g., *latitude= (-30, 30)* will extract only latitudes from 30 S to 30 N.

    latitude = None

18. *longitude* - required longitude which user wants to extract from the model global data. By default it takes *None* (i.e. extract model global longitudes). User can specify their required longitude in tuple. For e.g., *longitude= (60, 100)* will extract only longitude from 60 E to 100 E.

    Note: Model requires longitude to be specified based on (0 to 360) format, and not by (-180 to 180).

    longitude = None

19. *targetGridResolution* – output resolution in degree (1 degree = 100 km approx., 0.12 = 12 km) if *targetGridResolution* is set to *None*, then model resolution will be kept in the grib2 file. This must be a *number* or *None*.

    targetGridResolution = 0.125 or
    targetGridResolution = None

20. *targetGridFile* – absolute path of sample grib2/pp/ff/nc file, which contains at least one variable with latitude, longitude information. Model variables will be converted to this target grid file resolution including spatial regrid, start-end of latitude-and-longitude.

    targetGridFile =
    /home/umfcst/smallapps/UMRider/UMRRun/data/sample_global_0p12x0p18.grib2

21. *extraPolateMethod* - takes either *'auto'* or *'linear'* or *'mask'* or Iris supported interpolation method. *'linear'* means all variables will be linearly extrapolated over masked regions also. *'mask'* means all masked variables of model outputs, will not be extrapolated over masked regions. *'auto'* will take care properly the necessary variables will be extrapolated over masked regions and remaining variables will be masked over mask regions. The *'auto'* option is suggested one, because it will make sure precipitation variables are processed by nearest neighbourhood interpolation method if user passed different target grid file or resolution other than model resolution.

    extraPolateMethod = auto

22. *pressureLevels*– takes required pressure levels slice/extract only particular set of pressure levels from model pressure levels. User can specify either one or more levels. By default it takes *None*, i.e., it will extract all the model pressure levels.
    e.g. 1: pressureLevels = [850] -> extract 850 hPa only
    e.g. 2: pressureLevels = [850, 500, 200] -> extract only 850, 500 & 200 hPa levels only.
    Note 1: These pressure slice levels applicable to all the pressure level variables.
    Note 2: At the moment pressure levels interpolation is not supported!

    pressureLevels = None

23. *fillFullyMaskedVars* - If some variable has fully masked (for e.g., Incoming Shortwave flux during night time) then this option value will be set to that variable. By default it takes None, which will do nothing. If it has been set to 0 as value (or any other number), then those fully masked variables will be filled with this value wherever the mask is present in the grids.

    fillFullyMaskedVars = None

24. *soilFirstSecondFixedSurfaceUnit* - takes either *'cm'* or *'mm'*. By default it takes 'cm' argument (suggested for general purpose/WRF-Noah supported). For soil moisture/soil temperature variables depth below land surface units are initially set to either 'cm' (centimetre) or 'mm' (millimetre), and finally converted to 'm' (meter) in wgrib2. But anyhow if grib2 files will be read through by some other utility other than wgrib2 (say CDO, Grads, etc), then this first & second fixed surface units are important. So suggested unit is 'cm'.

    soilFirstSecondFixedSurfaceUnit = cm

25. *loadg2utils* - Load g2utils (UMRider Library) from *'system'* python which is installed through setup.py (OR) Load g2utils from *'local'* previous directory for the operational purpose, where normal user don't have write permission to change the g2utils! So *loadg2utils* argument should be either *system* (default) or *local*.

    loadg2utils = local

26. *overwriteFiles* – takes *'True'* or *'False'*. If *overwriteFiles* option set to *'True'* then existing output final files (if any) will be deleted from *outPath* and re-creating freshly. If *overwriteFiles* option is *'False'* and

all output final files are already existing in the *outPath*, then program will be exited without re-creating the output files. If partially created files exist (like few hours outfiles only exist or intermediate nc files) then by default make *overwriteFiles* option as *True* (though False as passed to *overwriteFiles* option) and re-create output fully.

overwriteFiles = True

27. *anlOutGrib2FilesNameStructure* and *fcstOutGrib2FilesNameStructure* takes list of string naming arguments to construct out file names (nomenclature). um2grb2.py will just concatenate the arguments, by replacing 3 predefined naming structure *('*HHH*', *YYYYMMDD*', '*ZZ*')* with its corresponding values/numbers in place of it.

*'*H*'* - forecast hours

*'*D*'* - forecast days (applicable only for multiples of 24 hours)

Note: Either '*H*' or '*D*' valid, not both!

*'*YYYYMMDD*'* - forecast reference date

'%d%m%y*'* - Alternate to above *'*YYYYMMDD*'* option. User can specify any acceptable time *strftime* format. um2grb2.py will figure it by finding *'%'* symbol.

*'*Z*'* - forecast reference UTC time (optional)

If user wants to 3 digit filled hours, then they need to specify as 3 times '*HHH*'. If they specify 2 digit filled hours (say '*HH*' only), but forecast hours have 3 digit, then by default it will assume as 3 digits but for single digit hour, it will fill 0 as prefix to make it as 2 digit. Same option is for UTC '*Z*'.

*'*pXp*'* - latitude x longitude grid resolution

Note: * will not be included in the name of the final out grib2 files.

e.g. 1: ('um_ana', '_', '*HHH*', 'hr', '_', '*YYYYMMDD*', '_', '*ZZ*', 'Z', '.grib2') this will produce grib2 files as 'um_ana_006hr_20160208_12Z.grib2' for the 6th hour forecast, 8th Feb 2016, 12 UTC.

e.g. 2: ('fcs', '_', '*HH*', 'h', '_z', '*YYYYMMDD*', '.grb2') this will produce grib2 files as 'fcs_06h_z20160208.grb2'

e.g. 3: ('prg', '*D*', '00z', '*%d%m%y*', '.grb2') will produce grib2 files as ‘prg100z080216.grb2'

e.g. 4: ('prg', '*D*', '00z', '*%d%m%y*', '_', '*pXp*' '.grb2') will produce grib2 files as 'prg100z080216_0p17x0p17.grb2' in case of *targetGridResolution = None* (i.e., modelResolution) or as 'prg100z080216_2p5x2p5.grb2' in case of t*argetGridResolution = 2.5* Defining analysis grib2 fileName structure. Must be in single line.

anlOutGrib2FilesNameStructure = ('um_ana', '_', '*HHH*', 'hr', '_', '*YYYYMMDD*', '_', '*ZZ*', 'Z', '_', '*pXp*', '.grib2')

28. *fcstOutGrib2FilesNameStructure* - Defining forecast grib2 file Name structure must be in single line.

fcstOutGrib2FilesNameStructure = ('um_prg', '_', '*HHH*', 'hr', '_', '*YYYYMMDD*', '_', '*ZZ*', 'Z', '_', '*pXp*', '.grib2')

29. *createCtlIdxFiles* - takes *True* or *False*. If it is set to *True* then um2grb2.py module will create grads control files and its index files for each and every grib2 files by using g2ctl.pl

    createGrib2CtlIdxFiles = True

30. *convertGrib2FilestoGrib1Files* - takes *True* or *False*. If it is set to *True* then using *'cnvgrib -g21'* command line um2grb2.py module will convert grib2 file to grib1 files. *CAUTION*: It may produce invalid variables names, grib1 param code for few variables which are produced by this um2grib2 conversion tool!

    convertGrib2FilestoGrib1Files = False

31. *grib1FilesNameSuffix* - takes grib1 extension. If *grib1FilesNameSuffix* is set as *'.grib1'*, then grib1 files will end with *'.grib1'* (default). Otherwise whatever string assigned will be added at the end of grib1 file names. None will add nothing to grib1 file names at the end of it.

    grib1FilesNameSuffix = '.grib1'

32. *removeGrib2FilesAfterGrib1FilesCreated* - takes *True* or *False*. If it is set to *True*, then grib2 files will be deleted and keep only grib1 files. By default *False*.

    removeGrib2FilesAfterGrib1FilesCreated = False

33. *createGrib1CtlIdxFiles*- takes *True* or *False*. If it is set to *True* then um2grb2.py module will create grads control files and its index files for each and every grib1 files by using grib2ctl.pl

    createGrib1CtlIdxFiles = False

34. This *debug* option should be either *True* or *False*. This will just print extra information like variables details, shape, execution process, etc.,

    debug = False

35. *setGrib2TableParameters* - option takes list of tuples which may contain WMO-Grib2 table parameters and its value. This means, the grib2 table parameter options will be overwritten as per user's setting in this option.

    e.g. 1: *setGrib2TableParameters* = [('centre', 28), ('subCentre', 0)]

    The above two options will be set to out grib2 files.

    e.g. 2: *setGrib2TableParameters* = [('shapeOfTheEarth', 0)]

    The above option will be set to out grib2 files.

    CAUTION: User must be aware on what are they setting in this option and its causes in out grib2 files! By default this option takes None.

    setGrib2TableParameters= None

36. *wgrib2Arguments* - After successfully creation of the grib2 (final ordered variables) file, wgrib2 command will be executed with the *'wgrib2Arguments'* options. pygrib/IRIS/UMRider is capable to write grib2 file with "*grid_simple*" packing algorithm, whereas wgrib2 is capable to convert packing from "*grid_simple*" to "*grid_complex_spatial_differencing*" by setting *-set_grib_typecomplex2* option

in it. This complex type packing reduces file size 1/3 compared to simple packing. And further can reduce the file size, by passing *-set_bin_prec 12* (compatible with ECMWF) which reduces the floating point precision (which further reduces the file size 1/5th of the original simple packing). By default *wgrib2Arguments* takes *"-set_grib_type complex2 -grib_out*" as argument. User can override this option by including extra wgrib2 arguments Or None (wgrib2 will not be executed).

-grib_out is important argument (to be compress, set precision, etc.). For more details on what are all options can be set in this *wgrib2Arguments*, see below links.

http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/speed.html

http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/set_bin_prec.html

wgrib2Arguments = -set_grib_type complex2 -grib_out

37. *wgrib2netcdf* - takes *True* or *False*. If wgrib2netcdf enabled then grib2 file will be converted to NetCDF via wgrib2 -netcdf command and removed the grib2 source files.

    wgrib2netcdf= False

38. This *callBackScript* option takes any user defined script (any script, not limited to Python)! User should provide absolute or relative path of their script and make sure that script is self-executable with shebang and executable permission! After successfully created out grib2 files, this *callBackScript* will be executed with possibly command line keyword arguments as follows

    Keyword args: (*date*, *outpath*, *oftype*, *utc*) where,

    'date' -> out files processed date,

    'outpath' -> out files path,

    'oftype' -> 'analysis' or 'forecast'

    'utc' -> UTC cycle value in string ('00' or '06' or '12' or '18')

    For e.g.: callBackScript = imd_mfi_rename_g2files_and_put_into_ftp.sh, where this shell script access the above four command line arguments and push output grib2 files into NCMRWF ftp server (ftp.ncmrwf.gov.in) using scp command. This kind of post jobs are executed after grib2 files have been generated by UMRider jobs. Similarly, user can pass their own script for making tar.gz files of grib2 outfiles (as per user need!).

    callBackScript = None

############ UMRider - User Defined One Time Setup Configuration Options Ends ##############

User can **export UMRIDER_SETUP** in bash or compute node shell with value of an absolute path of above setup.cfg file, which will be read by the UMRider scripts.

2.3.3 UMRiderVars Configuration

---

vars configure file: Used for the purpose of um2grb2.py conversion of only needed NCUM-G model out variables. um2grb2.py python parallel scripts will create analysis and forecast files, by converting to grib2 file only for the following cf_standard_name and varSTASH coded vars.

NCUM-G GLOBAL MODEL POST PROCESSING VARS CONFIGURE FILE

Filename: **vars.cfg** Comment Line : Line begins with **#** symbol HPCS : **Mihir** (Cray)

The following "variables configure" file will produce 72 fields in post processed out grib2 files.

Author: Arulalan <arulalan@ncmrwf.gov.in>Total number of variables: 65

############ UMRider - User Defined One Time Vars Configuration Options Begin ##############

Pressure Level Variable names & STASH codes
('geopotential_height', 'm01s16i202')
('x_wind', 'm01s15i201')
('y_wind', 'm01s15i202')
('upward_air_velocity', 'm01s15i242')
('air_temperature', 'm01s16i203')
('relative_humidity', 'm01s16i256')
('specific_humidity', 'm01s30i205')

Non Pressure Level Variable names & STASH codes
('tropopause_altitude', 'm01s30i453')
('tropopause_air_temperature', 'm01s30i452')
('tropopause_air_pressure', 'm01s30i451')
('surface_air_pressure', 'm01s00i409')
('air_pressure_at_sea_level', 'm01s16i222')
('surface_temperature', 'm01s00i024')
('relative_humidity', 'm01s03i245')
('specific_humidity', 'm01s03i237')
('air_temperature', 'm01s03i236')
('dew_point_temperature', 'm01s03i250')
('atmosphere_convective_available_potential_energy_wrt_surface', 'm01s05i233')
('atmosphere_convective_inhibition_wrt_surface', 'm01s05i234')
('high_type_cloud_area_fraction', 'm01s09i205')
('medium_type_cloud_area_fraction', 'm01s09i204')
('low_type_cloud_area_fraction', 'm01s09i203')
('cloud_area_fraction_assuming_random_overlap', 'm01s09i216')
('cloud_area_fraction_assuming_maximum_random_overlap', 'm01s09i217')
('atmosphere_cloud_liquid_water_content', 'm01s30i405')
('atmosphere_cloud_ice_content', 'm01s30i406')
('atmosphere_mass_content_of_water', 'm01s30i404')
('atmosphere_mass_content_of_dust_dry_aerosol_particles', 'm01s30i403')

if STASH is None, because atmosphere_precipitable_water_content will be calculated by using
#atmosphere_mass_content_of_water –
atmosphere_mass_content_of_dust_dry_aerosol_particles–
atmosphere_cloud_liquid_water_content–atmosphere_cloud_ice_content
('atmosphere_precipitable_water_content', 'None')
('x_wind', 'm01s03i225')
('y_wind', 'm01s03i226')
('visibility_in_air', 'm01s03i247')
('precipitation_amount', 'm01s05i226')
('stratiform_snowfall_amount', 'm01s04i202')
('convective_snowfall_amount', 'm01s05i202')
('stratiform_rainfall_amount', 'm01s04i201')

```
('convective_rainfall_amount', 'm01s05i201')
('rainfall_flux', 'm01s05i214')
('snowfall_flux', 'm01s05i215')
('precipitation_flux', 'm01s05i216')
('fog_area_fraction', 'm01s03i248')
('toa_incoming_shortwave_flux', 'm01s01i207')
('toa_outgoing_shortwave_flux', 'm01s01i205')
('toa_outgoing_shortwave_flux_assuming_clear_sky', 'm01s01i209')
('toa_outgoing_longwave_flux', 'm01s02i205')
('toa_outgoing_longwave_flux_assuming_clear_sky', 'm01s02i206')
('surface_upward_latent_heat_flux', 'm01s03i234')
('surface_upward_sensible_heat_flux', 'm01s03i217')
('surface_downwelling_shortwave_flux_in_air', 'm01s01i235')
('surface_downwelling_longwave_flux', 'm01s02i207')
('surface_net_downward_longwave_flux', 'm01s02i201')
('surface_net_downward_shortwave_flux', 'm01s01i202')

# for the following two vars STASH is None, because it will be calculated by using
# net and down shortwave/longwave flux
('surface_upwelling_longwave_flux_in_air', 'None')
('surface_upwelling_shortwave_flux_in_air', 'None')

('atmosphere_boundary_layer_thickness', 'm01s00i025')
('atmosphere_optical_thickness_due_to_dust_ambient_aerosol', 'm01s02i422')
('moisture_content_of_soil_layer', 'm01s08i223'),  # 4 layers

# single layer, this must be after 4 layers as in order
('soil_moisture_content', 'm01s08i208'),       # single layer

## though moisture_content_of_soil_layer and volumetric_moisture_of_soil_layer
## has same STASH code, but we must include seperate entry here.
('volumetric_moisture_of_soil_layer', 'm01s08i223'), # 4 layers

# single layer, this must be after 4 layers as in order
('volumetric_moisture_of_soil_layer', 'm01s08i208'), # single layer
('soil_temperature', 'm01s03i238')
('sea_ice_area_fraction', 'm01s00i031')
('sea_ice_thickness', 'm01s00i032')

# the snowfall_amount might be changed as liquid_water_content_of_surface_snow by convert # it
intowater equivalent of snow amount, before re-ordering itself.
('liquid_water_content_of_surface_snow', 'm01s00i023')

# the below one is for land-sea binary mask which should presents only in analysis files.
# so we must keep this as the last one in the ordered variables!
('land_binary_mask', 'm01s00i030')

# the below one is for orography which presents only in analysis 00 file.so we must keep this as #
last one in the ordered variables!
('surface_altitude', 'm01s00i033')
```

############ UMRider - User Defined One Time Vars Configuration Options Ends ###############

User can **export UMRIDER_VARS** in bash or compute node shell with value of an absolute path of above setup.cfg file, which will be read by the UMRider scripts.

2.4 Executions in HPCS

At NCMRWF, there are two HPCS (High Performance Computing Systems), namely Bhaskara (IBM) and Mihir (CRAY). To submit any jobs in compute node of HPCS, user has to write either bsub job script (for Bhaskara) or qsub job script (for Mihir). Listed below is a detailed example for submitting UMRider parallel job in bsub and qsub compute nodes of both HPCS.

2.4.1 bsub jobs – Bhaskara (IBM)

In bsub scripts, bsub commands starts with "#BSUB" followed by options to utilise compute node resources. To produce analysis grib2 files at 00UTC of NCUM-G global deterministic model, user has to run the UMRider module script named "um2grb2_anl_00Z.py" which will read UMRIDER_SETUP and UMRIDER_VARS environmental variable, followed by executing the post process of unified model output conversion to grib2 files by parallel python. Also exported is an environment GRIB2TABLE variable which is essential while writing grib2 files for the purpose to read custom variables, grib2 param codes and local table information.

2.4.1.1 ncum_global_post_anl_00Z.sh

The below bash bsub script will produce NCUM-G global deterministic analysis grib2 files at 00 UTC when it is submitted to Bhaskara HPCS compute node via "bsub< ncum_global_post_anl_00Z.sh"

```
################ UMRider - ncum_global_post_anl_00Z.sh bsub bash script Begins ################
#!/bin/bash
#
#BSUB -a poe               # set parallel operating environment
#BSUB -J umranl            # job name
#BSUB -W 06:00             # wall-clock time (hrs:mins)
#BSUB -n 6                 # number of tasks in job
#BSUB -q small             # queue
#BSUB -e um2grb2.anl.00hr.err.%J.hybrid     # error file name in which %J is replaced by the job ID
#BSUB -o um2grb2.anl.00hr.out.%J.hybrid     # output file name in which %J is replaced by the job ID

# find out the directory of this bash script after submitted to bsub
DIR="$( cd "$( dirname "${BASH_SOURCE[1]}" )" &&pwd )"

# set the absolute path of the local table
localTable=/gpfs2/home/arulalan/UMRiderUsr/UMRRun/tables/local/ncmr/v1/ncmr_grib2_local_table

# set the absolute path of the script for analysis 00utc
g2script=/gpfs2/home/arulalan/UMRiderUsr/UMRRun/g2scripts/um2grb2_anl_00Z.py

# export the configure paths to needed variables
export UMRIDER_SETUP=$DIR/umr_setup.cfg
export UMRIDER_VARS=$DIR/umr_vars.cfg
export GRIB2TABLE=$localTable

echo "export UMRIDER_SETUP="$UMRIDER_SETUP
echo "export UMRIDER_VARS="$UMRIDER_VARS
```

```
echo "export GRIB2TABLE="$GRIB2TABLE

# sourcing umtid_bashrc to load module python-uvcdat-iris!
source "$DIR/umrider_bashrc"
# execute the script
python $g2script

################ UMRider - ncum_global_post_anl_00Z.sh bsub bash script Ends #################
```

2.4.1.2 Other bsub scripts

By replacing the g2script environment variable in above bsub script with um2grb2_anl_06Z.py (rename script as ncum_global_post_anl_06Z.sh) will produce NCUM-G global deterministic analysis grib2 files at 06 UTC when it is submitted to Bhaskara HPCS compute node via bsub command as shown here "bsub< ncum_global_post_anl_06Z.sh" .

Similarly, UMRider g2scripts module have scripts to produce analysis (00, 06, 12, 18 UTC) and forecast (00, 12 UTC) grib2 files as follows - um2grb2_anl_12Z.py (analysis 12 UTC), um2grb2_anl_18Z.py (analysis 18 UTC), um2grb2_fcst_00Z.py (forecast 00 UTC) and um2grb2_fcst_12Z.py (forecast 12 UTC). The same scripts will work for all the model types.

2.4.2 qsub jobs – Mihir (Cray)

In qsub scripts, qsub commands starts with "#PSUB" followed by options to utilise compute node resources. To produce analysis grib2 files at 00 UTC of NCUM-G global deterministic model, user has to run the UMRider module script named "um2grb2_anl_00Z.py" which will read UMRIDER_SETUP and UMRIDER_VARS environmental variable, followed by executing the post process of unified model output conversion to grib2 files by parallel python. Also exported an environment GRIB2TABLE variable which is essential while writing grib2 files for the purpose to read custom variables grib2 param codes and local table information. Apart from #PSUB commands, in qsub python script need to be submitted by APRUN command, which will submit the python script to compute node and execute in it.

2.4.2.1 ncum_global_post_anl_00Z.sh

The below bash qsub script will produce NCUM-G global deterministic analysis grib2 files at 00 UTC when it is submitted in Mihir HPCS compute node via qsub command as shown here "qsub ncum_global_post_anl_00Z.sh"

```
################ UMRider - ncum_global_post_anl_00Z.sh qsubbash script Begins ################
#!/bin/bash
#
#PBS -N u2g2eps
#PBS -l walltime=06:00:00
#PBS -q NCMRWF
#PBS -l select=1:ncpus=36:vntype=cray_compute -l place=scatter
#PBS -V
```

```
#PBS -e um2grb2.anl.00z.err.hybrid
#PBS -o um2grb2.anl.00z.out.hybrid
export APRUN="aprun -j1 -n1 -N1 -d36"

# find out the directory of this bash script after submitted to qsub
DIR=${PBS_O_WORKDIR}
localTable=/home/umfcst/smallapps/UMRider/UMRRun/ncmr_grib2_local_table
g2scripts_absolute_dir=/home/umfcst/smallapps/UMRider/UMRRun/g2scripts/
g2script=$g2scripts_absolute_dir/um2grb2_anl_00Z.py

# export the configure paths to needed variables
export UMRIDER_SETUP=$DIR/ncum_global_post_um2grb2_setup.cfg
export UMRIDER_VARS=$DIR/ncum_global_post_um2grb2_vars.cfg
export GRIB2TABLE=$localTable

echo "export UMRIDER_SETUP="$UMRIDER_SETUP
echo "export UMRIDER_VARS="$UMRIDER_VARS
echo "export GRIB2TABLE="$GRIB2TABLE

# sourcing umtid_bashrc to load module python-uvcdat-iris!
source "$DIR/umtid_bashrc"
# execute the script
$APRUN python $g2script

################ UMRider - ncum_global_post_anl_00Z.sh qsub bash script Ends #################
```

2.4.2.2 Other qsub scripts

By replacing the g2script environment variable in above qsub script with um2grb2_anl_06Z.py (rename script as ncum_global_post_anl_06Z.sh) will produce NCUM-G global deterministic analysis grib2 files at 06UTC when it is submitted in Mihir HPCS compute node via “qsub ncum_global_post_anl_06Z.sh”. Similarly, UMRider g2scripts module have scripts to produce analysis (00, 06, 12, 18 UTC) and forecast (00, 12 UTC) grib2 files as follows - um2grb2_anl_12Z.py (analysis 12 UTC), um2grb2_anl_18Z.py (analysis 18 UTC), um2grb2_fcst_00Z.py (forecast 00 UTC) and um2grb2_fcst_12Z.py (forecast 12 UTC). The same scripts will work for all the model types.

2.5 UMRiderGUI – A Graphical User Web Interface

There are 38 options in setup configuration file of UMRider, which makes it difficult to design and use the command line interface or script. So the graphical user web interface (UMRiderGUI) for UMRider is developed, keeping in mind to the easy access of sections 2.3.2, 2.3.3 and 2.4.1. The UMRiderGUI was developed by a student Mr. Somjeet Dasgupta, VIT University, Chennai as final semester mini project and implemented at NCMRWF during June 2016. Users whoever having account in Bhaskara HPCS of NCMRWF, can access the UMRiderGUI at http://192.168.0.222:8083/ from their web browser (preferably in google chrome). Figure 1 shows the schematic flow chart of web interface design. Figures 2 to 12, show the screenshots taken on 21-May-2020. Figure 2 shows login screen, Figures 3-6 shows available options

(38 questions) to the user and it will generate the setup.cfg file based on user's choices in the temporary directory of UMRiderGUI server path. Figure 7 shows the interface to select the list of variables by the user, which will create the vars.cfg file in the temporary. Figures 8-10 show the choices for selecting bsub scripts and interface to edit them before submit.

Finally, based on user's choice the generated files such as setup.cfg, vars.cfg, bsub bash scripts which are stored in the temporary directory of UMRiderGUI server, will be copied to the user's home directory in Bhaskara HPCS via scp (secured copy) command. Now users are left with option to logout or submit bsub jobs which are generated by them. Figures 11-12 show the screenshots of last pages of UMRiderGUI, which displays the steps on how interface submits bsub bash script jobs into compute nodes of Bhaskara by ssh (secured shell) remote login.

The UMRiderGUI is designed and developed by using web technologies such as PHP, XML, HTML, CSS, and JavaScript. The interface for setup.cfg, vars.cfg, and bsub scripts are dynamically generated by reading the following xml files.

1. setup_description.xml
2. vars_description.xml
3. description_global_bsubs.xml
4. description_regional_bsubs.xml
5. description_ensemble_bsubs.xml

It is easy to insert new options or edit existing options in the xml file (aimed for the future releases of UMRider), which will be reflected in the web interface of UMRiderGUI immediately. In future, the same interface will be implemented for the qsub jobs in Mihir HPCS. This interface can be easily setup and configured in other institutions (Unified Model partners) HPCS environment.

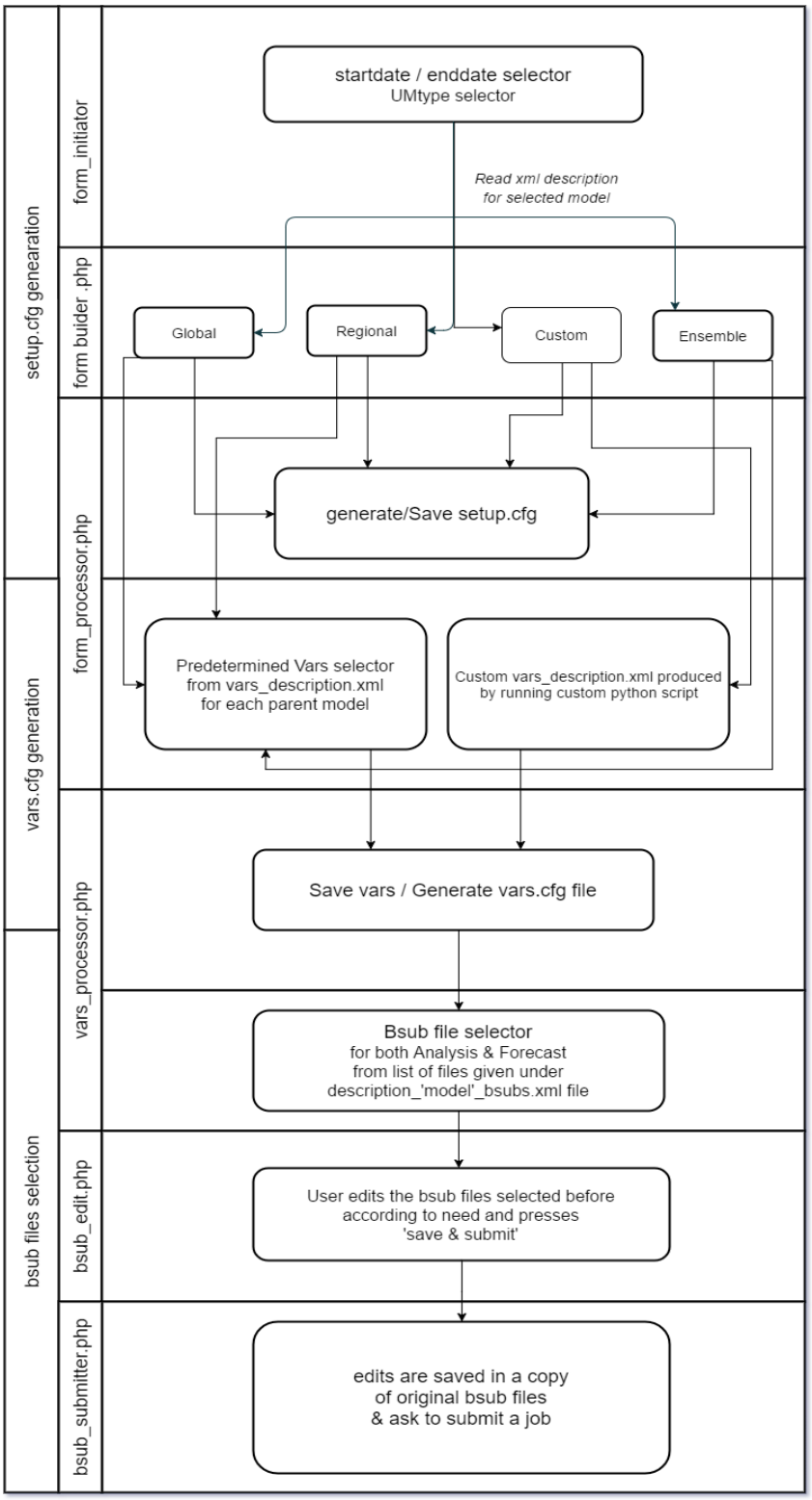

Figure 1: Schematic Work Flow Chart of the UMRiderGUI web interface state from beginning to end.

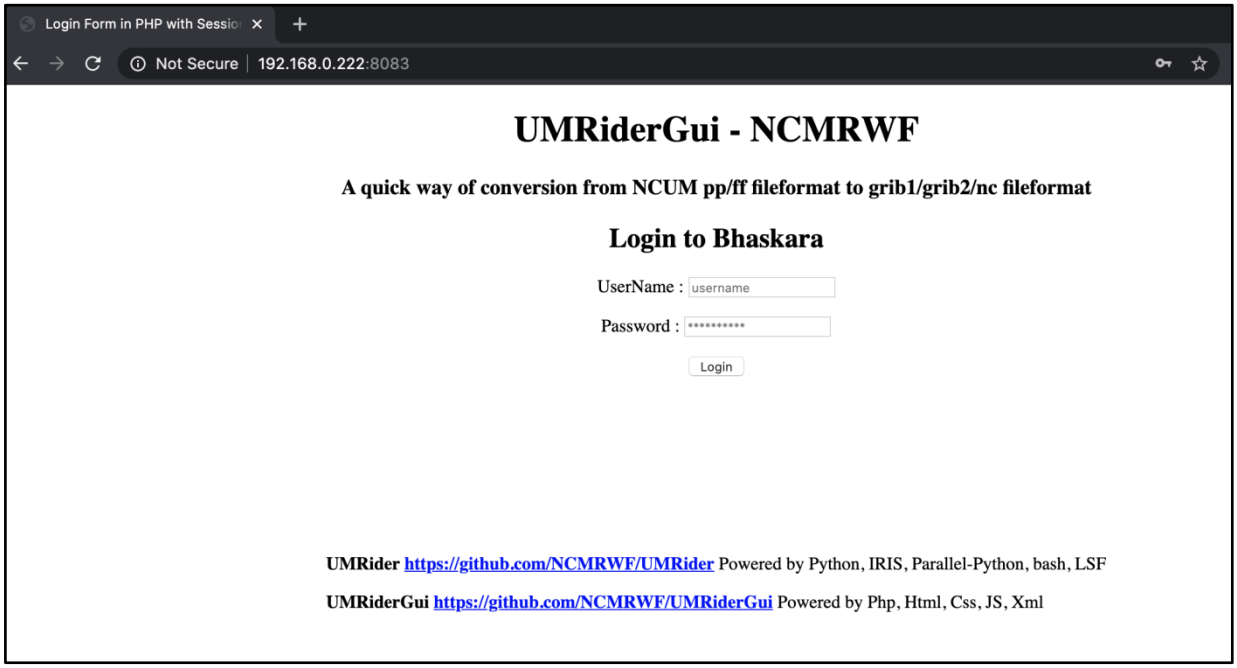

Figure 2: Clicking on http://192.168.0.222:8083/ in web browser will land into user login page, in which Bhakasra HPCS user can enter their user name and password.

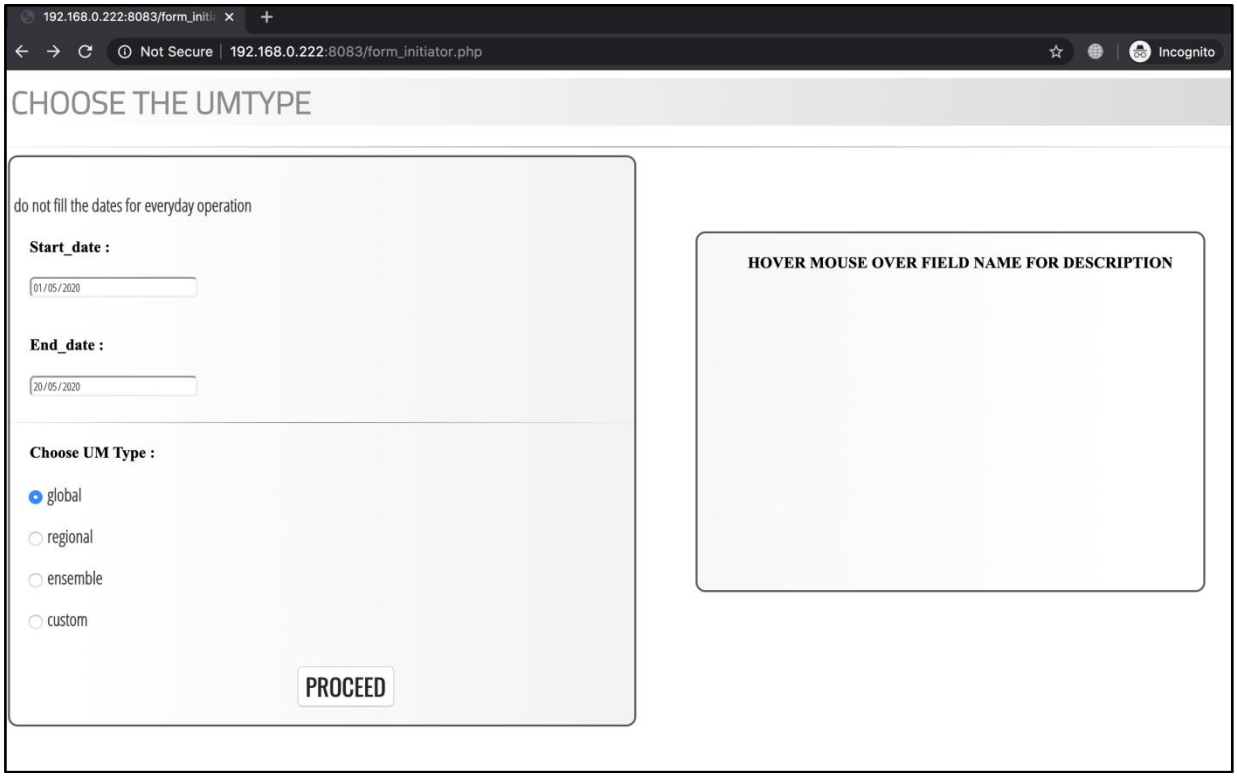

Figure 3: After successful login (Figure-2) the user will be asked 38 questions (to generate setup.cfg file – see section 2.3.2) starting from start date, end date and the UM type. User has to fill start date, end date (if needed), and select their choice of UMtype followed by click on proceed button. If user mouse is placed over on any of the field in left side, the proper description will be shown in right box of the page

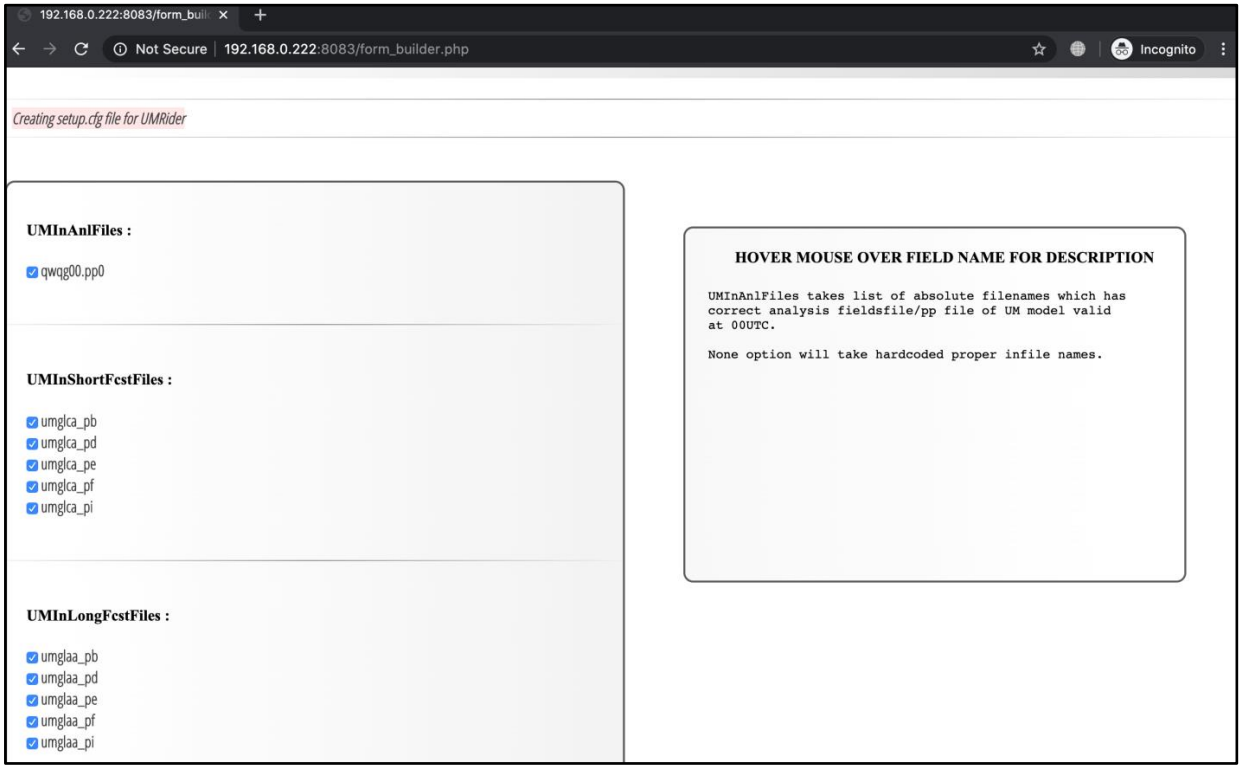

Figure 4: Based on the UM type (selected by the user – Figure 3), the following choices will be shown to (UMInAnlFiles, UMInShortFcstFiles, UMInLongFcstFiles) to user.

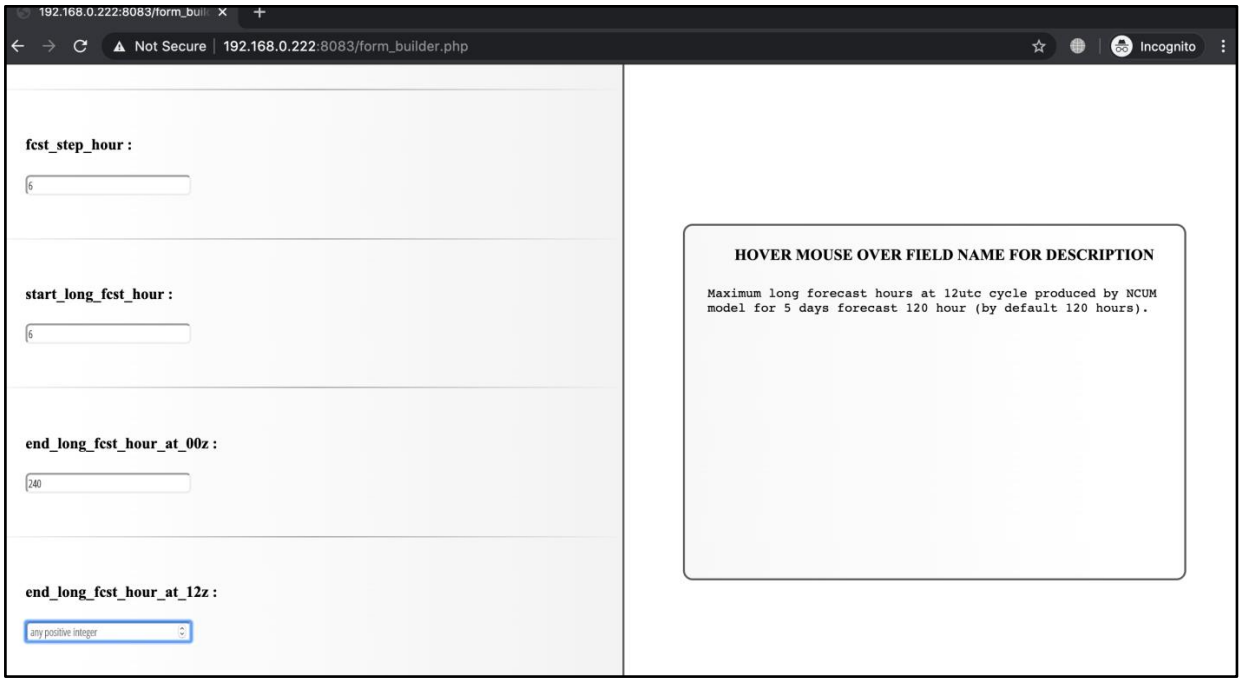

Figure 5: User has to scroll down to see rest of questions to be filled to generate the setup.cfg file. Next set of questions are shown here.

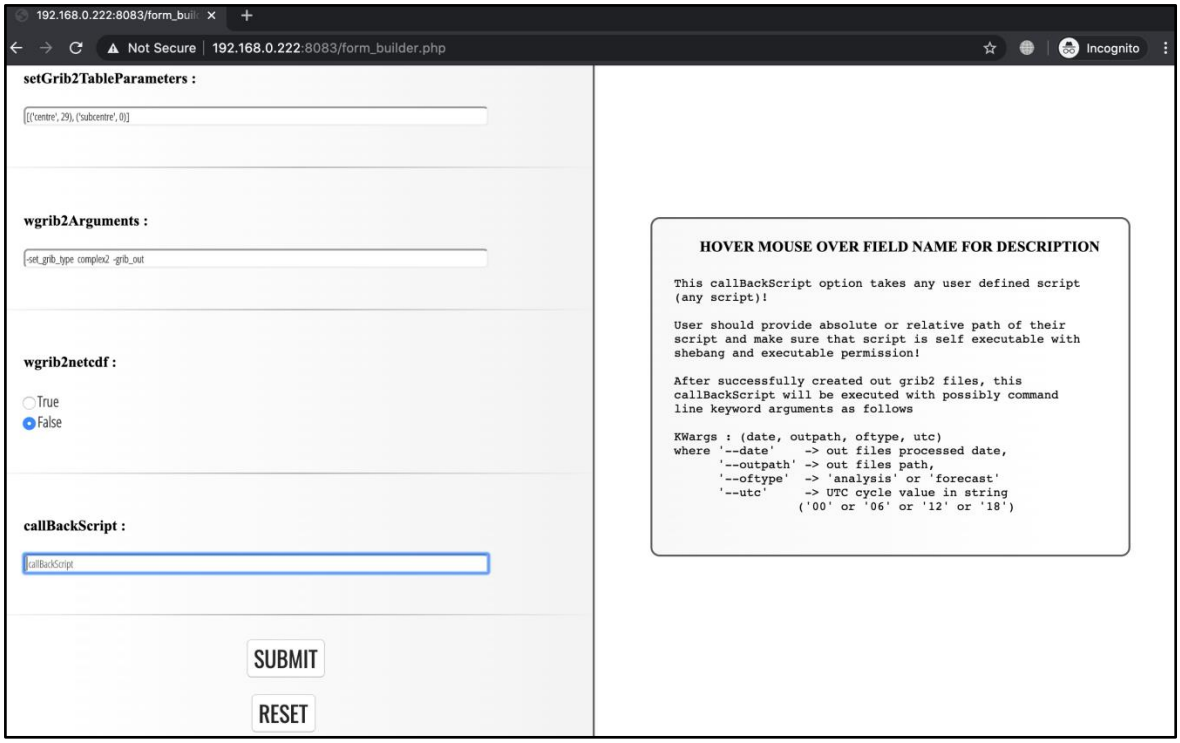

Figure 6: Finally it reaches last set of questions (wgrib2netcdf, callBackScript). User can either click on submit or reset button. Reset button will clear all the fields in the form. Submit button will create the "setup.cfg" file (in the server's temporary directory) from the user's choices and answers for the above UMRider's 38 options. Empty fields will be treated as None value for the options.

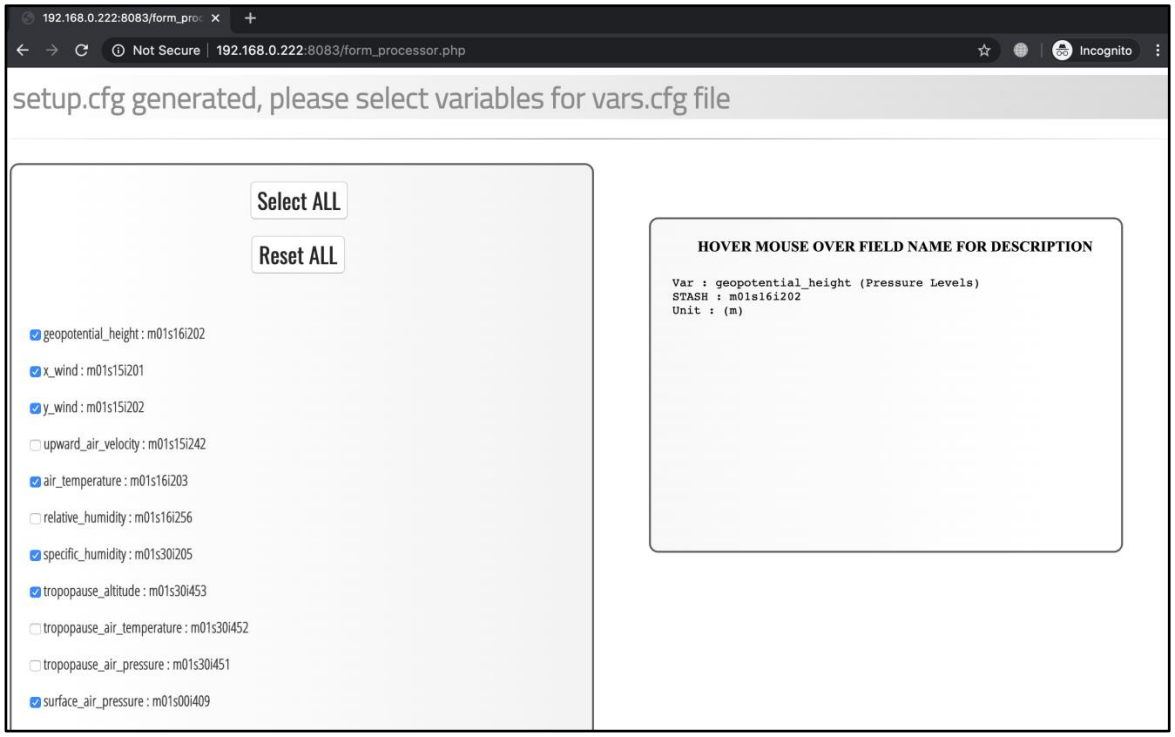

Figure 7: After setcup.cfg is generated, and next it asks users to select list of variables shown. User can select multiple variables by manually or by clicking on "Select ALL" button to mark all available variables as selected. User has to scroll till end of the page, and click on "submit" button, which will create "vars.cfg" file (in the server's temporary directory) based on user's choices of variables.

Figure 8: After vars.cfg is generated, the user selects the bsub files for analysis and forecast. User can select multiple bsub files for both. Then click on submit button.

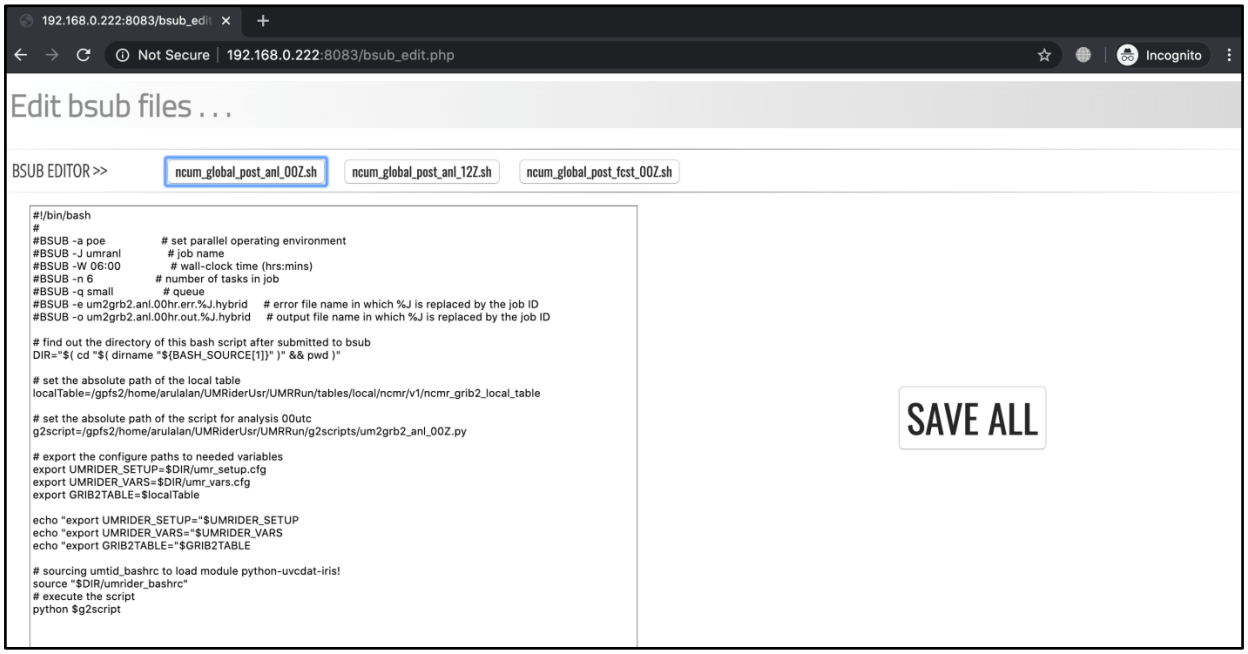

Figure 9: After submission with selection for bsub files need to be generated (See Figure 8), it will show the bsub file editing pages, where it can be edited. Here shown ncum_global_post_anl_00Z.sh bsubscript, which will be stored after submission of save all button (see section 2.4.1).

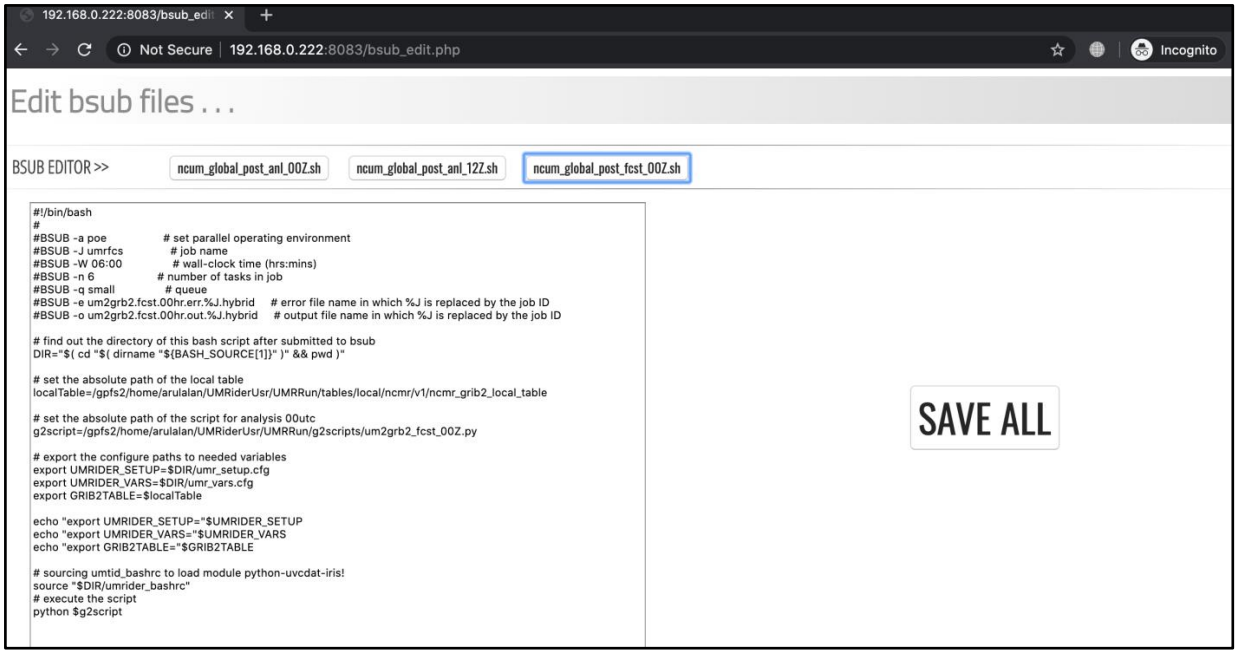

Figure 10: Here shown ncum_global_post_fcst_00Z.sh bsub script, which will be stored after submission of save all button (see section 2.4.1).

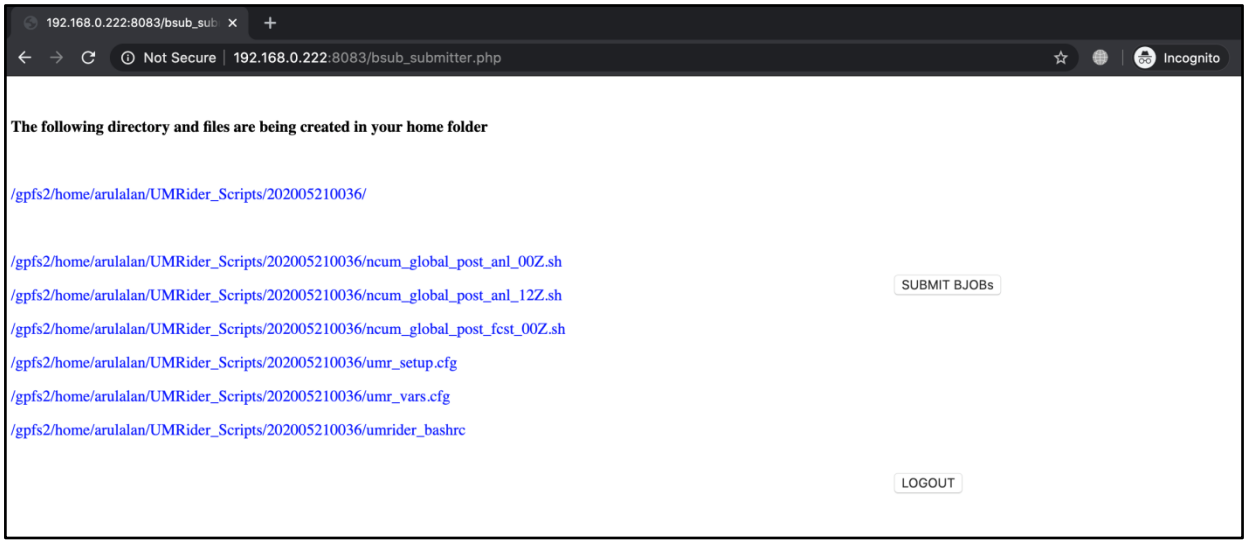

Figure 11: After click on “SAVE ALL” in previous window (see Figure 10), whatever generated files previously and stored in temporary directory of server, will be copied to user’s home directory in Bhaskara HPCS via scp. See below screenshot, which shows new directory path “UMRider_Scripts/202005210036” has been created and copied setup.cfg, vars.cfg, bsub bash scripts into that folder. Now user can click on “SUBMIT BJOBS” which will submit the all busb scripts to Bhaskara IBM HPCS compute node. Also, user can logout by click on the button instead of submit bjobs

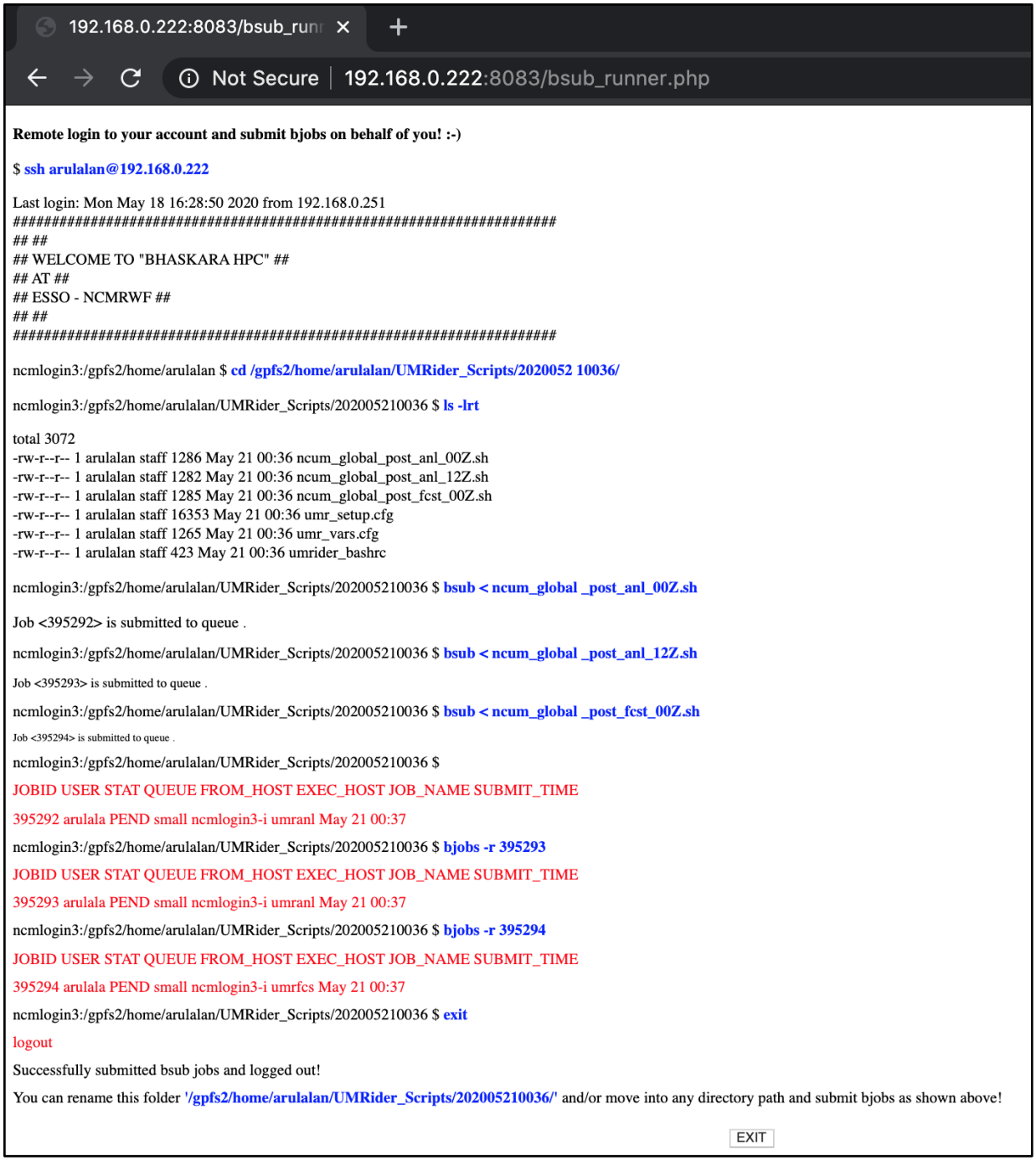

Figure 12: After submission of bsub job in previous stage (see Figure 11), the page will show how the UMRiderGUI logged into user's account in Bhaskara HPCS, followed by listing newly created directory "UMRider_Scripts/202005210036" and files. Followed by it submit all bsub bash scripts to compute node via bsub command and listing out job ids, finally logged out of HPCS. User can click on exit button to close the UMRiderGUI web interface.

3. UMRider Utilities Usage

UMRider source package contains a collection of modules named as “g2utils” which have the following python modules.

1. ncum_load_rules.py
2. cubeutils.py
3. mdlutils.py
4. um2grb2.py
5. um2grb2tigge.py
6. umeps2grb2.py
7. configpaths.py

In the above python modules, first three modules are coded with many functions which are useful far beyond the scope of UMRider. i.e., these module are programmed in such a way that one can use the available functions for generic post processes using python-iris, not limited to unified model outputs.

3.1 ncum_load_rules.py

This module has a function named “update_cf_standard_name” which can be passed as callback script while loading cubes in iris python library. This function is very useful in UMRider as it supports to override the meta information of cubes while reading unified model outputs pp/ff files and post processed grib2 files which are generated by UMRider itself. This call back function used to update the cubes standard_name if not exists (i.e. unknown) or to correct the cf_standard_name of particular cube (if wrongly assigned in um_cf_map.py) while loading iris cubes.

```
>>> import iris
>>> cubes = iris.load('umglaa_pf000', callback=update_cf_standard_name)
>>> cubes = iris.load('um_ana.grib2', callback=update_cf_standard_name)
```

3.2 cubeutils.py

This module have the following generic functions aimed to solve basic mathematical operator on the cubes without losing its meta information, which can be called by any python-iris based scripts.

3.2.1 cubeAverager

This function takes the following parameters as argument aimed to make either temporal average or summation of a given iris cube.

:tmpCube: The temporary cube data (in iris format) with non-singleton time dimension

:action: mean| sum (accumulated fields are summed and instantaneous are averaged).

:dt: A standard string representing forecast step duration/intervals.

:actionIntervals: A nonstandard string to add inside cell_methods comments section.

:tpoint: cbound | lbound | rbound time point

:fpoint: cbound | lbound | rbound forecast period

:tbounds: True | False (False removes time bounds)

:fbounds: True | False (False removes fcst bounds)

:return: meanCube: An iris formatted cube of the resultant data either averaged or summed.

3.2.2 cubeAddSubtractor

The inbuilt function of iris.analysis.maths.subtract does not serve out the purpose (it loses meta information after processed). So this cubeAddSubtractor do the addition or subtraction of two given cubes and set all necessary meta information to the resultant cube.

: cube: primary cube data (in iris format)

: otherCube: secondary cube data (in iris format)

: action: add | sub

: standard_name: resultant cube's standard name

: long_name: resultant cube's long name

: removeSTASH: True | False. If it is True, then resultant cube will have no stash meta information

: return : resCube: An iris formatted cube of the resultant data either added or subtracted.

3.2.3cubeCummulator

This function is useful to make cumulative of cube data over time dimension where cumulative happens over each and every time step. i.e., resultant cube will have same time dimension as input cube's time dimension.

: cube: primary cube data (in iris format)

: standard_name: resultant cube's standard name

: long_name: resultant cube's long name

: unit : resultant cube's unit

: standard_name: resultant cube's standard name

: removeSTASH : True | False. If it is True, then resultant cube will have no stash meta information

: addZerosFirstCube: True | False. It does add zeros cube at beginning of the cumulated cubes, which is useful to the TIGGE grib2 data generation.

:return : Generators | Yields the cumulated cubes (not the list, which is useful for memory management)

3.2.4 cubeRealizationAverager

This function is used to product realization wise average or summation at single time point.

: tmpCube : primary cube data (in iris format)

: action : mean | sum over realization dimension (i.e. over ensemble dimension of each variable)

: dr : A standard string representing realization step duration/intervals.

: return : meanCube: resCube: An iris formatted cube of the resultant data either averaged or summed over ensemble dimension.

3.3 mdlutils.py

This module named as mdlutils.py (i.e., model level utils) and it have the following generic functions aimed to convert hybrid model level dimension to user desired height (in meters), which can be called by any python iris based scripts.

3.3.1 cubeModelLevel2HeightConverter

This function is useful to convert a cube from model level to height above mean sea level or height above ground level (i.e., orography). This function requires to import “stratify” library to interpolate vertical height levels of cube.

: cube: primary cube data (in iris format) must have hybrid model level dimension data and can have multiple time coordinate dimensions as well along with latitude, longitude dimensions.

: orography: either user can pass orography cube as argument otherwise function will retrieve the surface_altitude from cube auxiliary coordinate system.

: target_altitudes: User can pass their required altitudes in meter unit as python list. If None was passed, then it will convert model level to the following heights in meter unit [10,50,80,100,120,150,180,200,250,300,350,400,500,600,700,800,900,1000,1500,2000,2500,3000,3500,4000,4500,5000]

: AGL: True | False (i.e. Above Ground Level True mean orography will be subtracted from hybrid altitude coordinate for computing base height. If it is True, then the resultant cube have dimension as height above ground level (AGL), otherwise above sea level (ASL).

: return: cube converted from model level to height ASL or AGL.

3.3.2 cubeModelLevel2CloudTopHeightConverter

This function is aimed to convert the hybrid model level dimensioned cube data into cloud top height by parallel python processes and it can be invoked by any python iris based scripts.

: data: primary cube data (in iris format) must be hybrid model level coordinated variable, but make sure only one time point is passed here.

: target_height: it must be shape of (latitude x longitude). User has to pass the cloud top height parameter value for single time point and its unit must be in meters.

: orography: either user can pass orography cube as argument otherwise function will retrieve the surface_altitude from cube auxiliary coordinate system.

: AGL: True | False (i.e. Above Ground Level True mean orography will be subtracted from hybrid altitude coordinate for computing base height. If it is True, then the resultant cube have dimension as height above ground level (AGL), otherwise above sea level (ASL).

: pblock_size: default value is 4. In parallel python-numpy block size (i.e. if pblock_size=4, then 4x4 numpy block will be taken at a time and by single processor.

: nprocesses: default value is 16. i.e. 16 parallel processors job will be performed by sharing same numpy array across processors.

: return: meanCube: cube converted from model level to cloud top height (in meters). Resultant cube will have single height dimension as its equivalent to cloud top height coordinate.

Acknowledgements

The developer of UMRider, the author of this technical report would like to thank Dr. E.N. Rajagopal, Head, NCMRWF who believed and encouraged the development of UMRider from the beginning, also who showed keen interest to make this utility as an operational post processing tool for all unified models operated at the centre. Author acknowledges Mr. Gopal Iyengar, MoES/NCMRWF, who assigned this utility development as project work, initially. Special thanks are also due to Mr. M.N. Raghavendra Sreevathsa who sparked the idea of this utility and the need of it. Most importantly, author acknowledge Mr. Somjeet Dasgupta who was a student (Register Number: 13BCE1151 of VIT University Chennai Campus) and completed his final semester Project work titled "UMRider Dynamic GUI" which was turned into "UMRiderGUI", under the guidance of this author jointly with Mr. M.N. Raghavendra Sreevathsa, and submitted to NCMRWF during June 2016.

A special thanks are due to Dr. Saji Mohandas who inspired the author, by sharing the basic knowledge about STASH, grib1 param codes, and encouraged from the beginning of this utility development for NCUM-G deterministic model post process. A warm acknowledgments are due to Drs. Jayakumar and Abhijit Sarkar for their support to post process of NCUM-R and NEPS-G model, respectively. Author would also like to thank Drs. John P. George and S. Indira Rani for their demand for using UMRider to post process IMDAA regional reanalysis. Author also acknowledges Mrs. Paromita Chakraborty and Dr. Ashu Mamgain who supported for making UMRider post processor for NEPS-G.

Without good evaluation, development of a good utility is not possible! Author wishes to thank Dr. Raghavendra Ashrit and his team members (Dr. Kuldeep Sharma, Mr. Sushant Kumar), who tested UMRider generated grib2 files and used it for their research activities. Author acknowledges all the national users (IMD, INCOIS, IITM, Indian Navy, SAC/ISRO, SASE, NIWE) who demanded and tested the products generated by UMRider. Many thanks are due to Dr. Richard Mladek, Analyst at ECMWF who tested the NEPS-G grib2 files (generated by UMRider) multiple times and accepted the contribution of NCMRWF global ensemble forecasts to the TIGGE project. Thanks are also due to Dr. João Teixeira, UK Met Office who tested and confirmed that UMRider works for their Unified Model outputs with minimum modifications in the configuration files.

Finally the author would also like to thank Dr. Saji Mohandas and Dr. E.N. Rajagopal who reviewed and edited this technical report to improve its content for better readability.

References

Ashu Mamgain, Abhijit Sarkar, Anumeha Dube, Arulalan T., Paromita Chakraborty, John. P. George and E.N. Rajagopal (2018), "Implementation of Very High Resolution (12 km) Global Ensemble Prediction System at NCMRWF and its Initial Validation", August 2018, NCMRWF Technical Report, NMRF/TR/02/2018.

Jayakumar, A., A. Mamgain, A. S. Jisesh, S. Mohandas, R. Rakhi, and E. N. Rajagopal (2016), "Evaluation of NCMRWF Unified Model vertical cloud structure with CloudSat over the Indian summer monsoon region", Proc. SPIE 9882, Remote Sens. Model. of the Atmos., Oceans, and Interactions VI, 988207; doi:10.1117/12.2223622.

Jayakumar, A., J. Sethunadh, R. Rakhi, T. Arulalan, S. Mohandas, G. R. Iyengar, and E. N. Rajagopal (2017), "Behavior of predicted convective clouds and precipitation in the high-resolution Unified Model over the Indian summer monsoon region", Earth and Space Science, 4, 303–313, doi:10.1002/2016EA000242.

Paromita Chakraborty, Anumeha Dube, Harvir Singh, Arulalan T., S. Kiran Prasad, Ashu Mamgain, Aditi, Sushant Kumar, Abhijit Sarkar, Raghavendra Ashrit and E. N. Rajagopal (2019), "Generation of Probabilistic Forecast Products from NCMRWF Ensemble Prediction System (NEPS)", April 2019, NCMRWF Technical Report, NMRF/TR/03/2019.

Raghavendra Ashrit, Kuldeep Sharma, Sushant Kumar, Anumeha Dube, S. Karunasagar, T. Arulalan, Ashu Mamgain, Paromita Chakraborty, Sumit Kumar, Abhishek Lodh, Devajyoti Dutta, Imranali Momin, M. T. Bushair, Buddhi J. Prakash, A. Jayakumar and E. N. Rajagopal (2019), "Prediction of the August 2018 heavy rainfall events over Kerala with high- resolution NWP models", Meteorol Appl.2020; 27:e1906, doi:10.1002/met.1906.

E.N. Rajagopal, G.R. Iyengar, John P. George, Munmun Das Gupta, Saji Mohandas, Renu Siddharth, Anjari Gupta, Manjusha Chourasia, V.S. Prasad, Aditi, Kuldeep Sharma and Amit Ashish (2012), "Implementation of Unified Model based Analysis-Forecast System at NCMRWF", May 2012, NCMRWF Technical Report, NMRF/TR/2/2012.

Saji Mohandas (2014), "Utility to convert UM fields file output to NCEP GRIB1 format: A user guide", April 2014, NCMRWF Technical Report, NMRF/TR/1/2014.

Sumit Kumar, A. Jayakumar, M. T. Bushair, Buddhi Prakash J., Gibies George, Abhishek Lodh, S. Indira Rani, Saji Mohandas, John P. George and E. N. Rajagopal (2018), "Implementation of New High Resolution NCUM Analysis-Forecast System in Mihir HPCS", August 2018, NCMRWF Technical Report, NMRF/TR/01/2018.

Appendix - A

A. UMRider – Source Code Structure, Access, Future Releases and License

A. 1. Source Code Structure

Listed out below are UMRider source code directories and file structures and their purposes.

UMRider *(top level directory)*

data–*directory contains few sample grib2 files used as targetGrid to do regrid of model outputs*

- sample_global_0p125x0p125.grib2– *sample global region* 0.125°x0.125°*grid*
- sample_global_0p12x0p18.grib2– *sample global region* 0.12°x0.18° *grid*
- sample_global_2p5X2p5_73X144.grib2– *sample global region* 2.5°x2.5° *grid*
- sample_ind_7-38N_67-98E_0p04X0p04.grib2– *sample Indian region* 0.04°x0.04° *grid*
- sample_ind_Eq-40N_62-106E_0p04X0p04.grib2– *sample Indian region* 0.04°x0.04° *grid*

g2utils– *directory contains all utilities (python modules, functions) of UMRider*

- configpaths.py– *Used to set all global variables (wgrib2, g2ctl.pl, etc.,) absolute paths*
- ncum_load_rules.py – *Manipulate iris cubes while loading model outputs (see section 3.1)*
- cubeutils.py– *General python maths utilities to handle iris cubes (see section 3.3)*
- mdlutils.py– *General python utilities to convert model hybrid level (see section 3.3)*
- um2grb2.py – *Entire UMRider is constructed over on this module Post ProcessGrib2 files*
- umctl2grb2 – *Convert global deterministic model to control member of NEPS-G*
- umeps2grb2.py – *Script to convert global all perturbed members to Grib2 files*
- um2grb2tigge.py–*Script to convert NEPS-G model outputs to TIGGE standard Grib2 files*
- umREGeps2grb2.py – *Script to post process of NCUM-R regional ensemble model outputs*

g2scripts– *contains all python scripts to execute model conversions by using g2utils modules*

- loadconfigure.py– *read configuration files (setup.cfg, vars.cfg)*
- um2grb2_anl_00Z.py– *NCUM-G Global/Regional Model Analysis Grib2 Creation at 00 UTC*
- um2grb2_anl_06Z.py– *NCUM-G Global/Regional Model Analysis Grib2 Creation at 06UTC*
- um2grb2_anl_12Z.py– *NCUM-G Global/Regional Model Analysis Grib2 Creation at 12 UTC*
- um2grb2_anl_18Z.py– *NCUM-G Global/Regional Model Analysis Grib2 Creation at 18 UTC*
- um2grb2_fcst_00Z.py– *NCUM-G Global/Regional Model Forecast Grib2 Creation at 00 UTC*
- um2grb2_fcst_12Z.py– *NCUM-G Global/Regional Model Forecast Grib2 Creation at 12 UTC*
- umctl2grb2_anl.py– *NCUM-G Global Analysis as control member of NEPS-G in Grib2 at 00 & 12 UTC*
- umctl2grb2_fcst.py– *NCUM-G Global Forecast as control member of NEPS-G in Grib2 at 00 & 12 UTC*
- umeps2grb2_fcst_00Z.py– *NEPS-G Global Forecast of 11 perturbed members in Grib2 at 00 UTC*
- umeps2grb2_fcst_12Z.py– *NEPS-G Global Forecast of 11 perturbed members in Grib2 at 12 UTC*
- tigge_umctl2grb2_anl_00Z.py– *TIGGE NEPS-G Analysis control member in Grib2 at 00 UTC*
- tigge_umctl2grb2_anl_12Z.py– *TIGGE NEPS-G Analysis control member in Grib2 at 12 UTC*
- tigge_umctl2grb2_fcst_00Z.py– *TIGGE NEPS-G Forecast as control member in Grib2 at 00 UTC*

- tigge_umctl2grb2_fcst_12Z.py– *TIGGE NEPS-G Forecast as control member in Grib2 at 12 UTC*
- tigge_umeps2grb2_fcst_00Z.py– *TIGGE NEPS-G Forecast of 11 perturbed members in Grib2 at 00 UTC*
- tigge_umeps2grb2_fcst_12Z.py– *TIGGE NEPS-G Forecast of 11 perturbed members in Grib2 at 12 UTC*

grib2PostScripts – *directory contains scripts to read and further post process grib2 files*

- extractRainEPSIndia_03-03Z.py– *extracts NEPS-G daily precipitation at 03-03 UTC*
- extractRainEPSIndia_individual_6hourly.py –*extracts NEPS-G 6-hourly precipitation at 00 UTC*
- ncumeps_extractRainEPSIndia_03-03Z.sh – *submits NEPS-G 03 UTC precipitation extraction*
- ncumeps_extractRainEPSIndia_individual_6hourly.sh – *submitd NEPS-G 6-hourly rainfall*

bsubScripts

This directory contains many sub directories and files (setup.cfg, vars.cfg, bsub bash shell scripts) used to post process many individual operational jobs in Bhaskara HPCS (see Sections 2.3.2, 2.3.3, 2.4.1, and 2.5)

qsubScripts

This directory contains many sub directories and files (setup.cfg, vars.cfg, qsub bash shell scripts) used to post process many individual operational jobs in Mihir HPCS (see sections 2.3.2, 2.3.3, and 2.4.2)

others– *contains other software modified source code as per need of UMRider at NCMRWF*

- cnvgrib– *software used to convert grib2 to grib1 file format*
  - params.f– *customised grib2 param codes which support for UMRider grib2 files*
  - README.txt– *explains the procedure to install source of cnvgrib after update*
- ncmrwfIRIS– *customised IRIS-Python library grib2 module to support of UMRider*
  - __init__.py – *updated init script(see Appendix – B)*
  - _grib_cf_map.py – *updated mapping of grib params (NCUM) to cf standard names*
  - _grib_cf_map_tigge.py – *updated mapping of grib params (TIGGE) to cf standard name*
  - _load_convert.py – *updated script while loading grib2 files in iris*
  - load_rules.py – *updated script while loading grib2 files in iris*
  - _save_rules.py– *updated script to write grib2 files (NCUM-G, NEPS-G, NCUM-R) in iris*
  - README.md – *instruction on how to copy these updated scripts before install of iris*

tables– *directory contains two table of UMRider*

- local/ncmr/v1– *contains local table of grib2 (see Appendix – C)*
  - ncmr_grib2_local_table– *contains customised local table as per NCUM model grib2*
- WRF-Noah– *contains tables of WRF model to read NCUM-G model grib2 files*
  - METGRID.TBL_NCUM – *metgrid grib2 table param to match with NCUM grib2*
  - Vtable.NCUM– *contains variables table to read NCUM fields correctly*

setup.py– *used to install UMRider Utility Library into system or user python path*

README.md– *tells about UMRider and other necessary information for the end users*

CHANGELOG.md – *contains versions releases and updates generates by git*

LICENSE– *contains GNU3 open source license agreement rules to use, modify and redistribute*

A. 2. Source Code Access

Users and readers of this technical report can download the source code of UMRider from NCMRWF code repository at GitHub online platform indicated below.

UMRider: https://github.com/NCMRWF/UMRideror https://github.com/arulalant/UMRider

UMRiderGUI: https://github.com/NCMRWF/UMRiderGui

Iris-python: https://github.com/SciTools/iris

A. 3. Future Releases

Though UMRider version 1.0 was released during Dec 2015, this technical report was written on the basis of latest version 3.1.1 of UMRider during May 2020. In the upcoming releases, we will try to fulfil the following requirements and upgrades.

- UMRscan – which should scan all pp/ff files of UM and helps users to understand the fields meta information such as variable CF Standard Name, STASH, etc.,
- NCUM-REPS – which should support for NCUM Regional Ensemble model outputs to convert into to grib2 files
- Universal – UMRider should support for all partners of UK Met Office –Unified Model, not limited to only NCMRWF – Unified Model. There are many challenges to achieve this target, as all NWP centre partners are producing their own customised model parameters (model version, variables, time stamp, time intervals, variable units, output file units, etc.). To overcome this problem, a collaborative work on UMRider development is required.
- Python3 upgradation – From the beginning, UMRider was written in Python 2.7.1 and official support for this python version had stopped on 1st January 2020. So it is mandatory to upgrade to python 3 version during next releases of UMRider.

A. 4. License

The UMRider (https://github.com/arulalant/UMRider & https://github.com/NCMRWF/UMRider) and UMRiderGUI (https://github.com/NCMRWF/UMRiderGui) are released under **GNU General Public License v3.0** – which states the following:

Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.

GNU General Public License v3.0 comes with the following terms and conditions:

Permissions

- Commercial use

- Modification
- Distribution
- Patent use
- Private use

Limitations

- No Liability
- No Warranty

Conditions

- License and copyright notice
- State changes
- Disclose source
- Same license

For further details about License, visit the link given below:

https://github.com/NCMRWF/UMRider/blob/master/LICENSE.

Appendix - B

NCUM STASH and Grib2 Param Codes

NCUM STASH

NCUM model output STASH code and its equivalent variable cf_standard_name is exactly same as given at link http://puma.nerc.ac.uk/STASH_to_CF/STASH_to_CF.html

Note: For a few STASH codes, there may be available multiple variables cf_standard_name.

Cf_Standard_Name

All variables of cf_standard_name table version V30 can be seen from below link,

http://cfconventions.org/Data/cf-standard-names/30/build/cf-standard-name-table.html .

NCEP-WMO Grib2

NCEP-WMO Standard Grib2 Table Parameters link,

http://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc.shtml .

By using all the above 3 links, one can get the **STASH** Vs**cf_standard_name**Vs **Grib2 parameter** codes.

Table B1. The list of Unified Model field name, STASH code, unit, short variable name in control file produced by g2ctl.pl script, and IRIS-Grib2 param codes (discipline, category, number, and type of first fixed surface) are shown. The STASH code mentioned in section 2.3.3 is as per iris python standard and here listed out the same but in simple stash code (without prefix of model 'm', item 'i', sub's' code).

Sl. No	STASH Code	Fields Name	IRIS - Grib2 Param Code				Control File Short Variable Name	Unit
			Discipline	Category	Number	Type of First Fixed Surface		
1	33	OROGRAPHY	2	0	7	1	MTERHsfc	m
2	409	SURFACE PRESSURE	0	3	0	1	PRESsfc	Pa
3	15242	W COMPNT (OF WIND) ON PRESSURE LEVS	0	2	9	100	DZDTprs	m s-1
4	15243	U WIND ON PRESSURE LEVELS	0	2	2	100	UGRDprs	m s-1
5	15244	V WIND ON PRESSURE LEVELS	0	2	3	100	VGRDprs	m s-1
6	16202	GEOPOTENTIAL HEIGHT ON P LEV	0	3	5	100	HGTprs	m
7	16203	TEMPERATURE ON P LEV	0	0	0	100	TMPprs	K
8	16222	PRESSURE AT MEAN SEA LEVEL	0	3	1	101	PRMSLsfc	Pa
9	16256	RH WRT WATER ON P LEV	0	1	1	100	RHprs	%

Table B1 – continuation

Sl. No	STASH Code	Fields Name	IRIS - Grib2 Param Code				Control File Short Variable Name	Unit
			Discipline	Category	Number	Type of First Fixed Surface		
10	23	SNOW AMOUNT OVER LAND	0	1	13	1	WEASDsfc	Kg m-2
11	24	SURFACE TEMPERATURE	0	0	17	1	SKINTsfc	K
12	25	BOUNDARY LAYER DEPTH	0	3	18	1	HPBLsfc	m
13	30	LAND MASK(No halo)	2	0	0	1	LANDsfc	1 (or) Proportion
14	31	FRAC OF SEA ICE IN SEA	10	2	0	1	ICECsfc	1 (or) Proportion
15	32	SEA ICE DEPTH (MEAN OVER ICE)	10	2	1	1	ICETKsfc	m
16	3245	RELATIVE HUMIDITY AT 1.5M	0	1	1	1	RH2m	%
17	3247	VISIBILITY AT 1.5M	0	19	0	1	VIS2m	m
18	3248	FOG FRACTION AT 1.5 M	0	1	192	1	FOGsfc	%
19	3250	DEWPOINT AT 1.5M	0	0	6	1	DPT2m	K
20	30451	PRESSURE AT TROPOPAUSE LVEL	0	3	0	7	PREStrop	Pa
21	30452	TEMPERATURE AT TROPOPAUSE LVEL	0	0	0	7	TMPtrop	K
22	30453	HEIGHT AT TROPOPAUSE LVEL	0	3	6	7	DISTtrop	m
23	15242	W COMPNT (OF WIND) ON PRESSURE LEVS	0	2	9	100	DZDTprs	m s-1
24	15243	U WIND ON PRESSURE LEVELS	0	2	2	100	UGRDprs	m s-1
25	15244	V WIND ON PRESSURE LEVELS	0	2	3	100	VGRDprs	m s-1
26	16202	GEOPOTENTIAL HEIGHT ON P LEV	0	3	5	100	HGTprs	m
27	16203	TEMPERATURE ON P LEV	0	0	0	100	TMPprs	K
28	16256	RH WRT WATER ON P LEV	0	1	1	100	RHprs	%
29	30205	SPECIFIC HUMIDITY ON P LEV/UV GRID	0	1	0	100	SPFHprs	kg/kg
30	409	SURFACE PRESSURE	0	3	0	1	PRESsfc	Pa
31	3209	10 METRE WIND U-COMP	0	2	2	103	UGRD10m	m s-1
32	3210	10 METRE WIND V-COMP	0	2	3	103	VGRD10m	m s-1
33	3236	TEMPERATURE AT 1.5M	0	0	0	103	TMP2m	K

Table B1 – continuation

Sl. No	STASH Code	Fields Name	IRIS - Grib2 Param Code				Control File Short Variable Name	Unit
			Discipline	Category	Number	Type of First Fixed Surface		
34	3237	SPECIFIC HUMIDITY AT 1.5M	0	1	0	103	SPFH2m	kg/kg
35	4201	LARGE SCALE RAIN AMOUNT	0	1	47	1	LSWPsfc	Kg m-2
36	4202	LARGE SCALE SNOW AMOUNT	0	1	15	1	SNOLsfc	Kg m-2
37	5226	TOTAL PRECIPITATION AMOUNT	0	1	8	1	APCPsfc	kg m-2
38	5201	CONVECTIVE RAIN AMOUNT	0	1	48	1	CWPsfc	Kg m-2
39	5202	CONVECTIVE SNOW AMOUNT	0	1	14	1	SNOCsfc	Kg m-2
40	5234	UNDILUTE PARCEL CIN	0	7	7	1	CINsfc	J/kg
41	9203	LOW CLOUD AMOUNT	0	6	3	1	LCDCsfc	%
42	9204	MEDIUM CLOUD AMOUNT	0	6	4	1	MCDCsfc	%
43	9205	HIGH CLOUD AMOUNT	0	6	5	1	HCDCsfc	%
44	16222	PRESSURE AT MEAN SEA LEVEL	0	3	1	101	PRMSLsfc	Pa
45	1202	NET DOWN SURFACE SW FLUX	0	4	9	1	NSWRFsfc	W m-2
46	1205	OUTGOING SW RAD FLUX (TOA)	0	4	8	8	USWRFtoa	W m-2
47	1207	INCOMING SW RAD FLUX (TOA)	0	4	7	8	DSWRFtoa	W m-2
48	1209	CLEAR-SKY (II) UPWARD SW FLUX (TOA)	0	4	198	8	CSUSFtoa	W m-2
49	1235	DOWNWARD SURFACE SW FLUX	0	4	7	1	DSWRFsfc	W m-2
50	2201	NET DOWN SURFACE LW RAD FLUX	0	5	5	1	NLWRFsfc	W m-2
51	2205	OUTGOING LW RAD FLUX (TOA)	0	5	4	8	ULWRFtoa	W m-2
52	2206	CLEAR-SKY (II) UPWARD LW FLUX (TOA)	0	5	195	8	CSULFtoa	W m-2
53	2207	DOWNWARD LW RAD FLUX: SURFACE	0	5	3	1	DLWRFsfc	W m-2
54	3217	SURFACE SENSIBLE HEAT FLUX	0	0	11	1	SHTFLsfc	W m-2
55	3234	SURFACE LATENT HEAT FLUX	0	0	10	1	LHTFLsfc	W m-2

Table B1– continuation

Sl. No	STASH Code	Fields Name	IRIS - Grib2 Param Code				Control File Short Variable Name	Unit
			Discipline	Category	Number	Type of First Fixed Surface		
56	5214	TOTAL RAINFALL RATE: LS+CONV	0	1	65	1	RPRATEsfc	Kg m-2 s-1
57	5215	TOTAL SNOWFALL RATE: LS+CONV	0	1	53	1	TSRWEsfc	Kg m-2 s-1
58	5216	TOTAL PRECIPITATION RATE	0	1	7	1	PRATEsfc	Kg m-2 s-1
59	Calculated	UPWARD SURFACE SW FLUX	0	4	8	1	USWRFsfc	W m-2
60	Calculated	UPWARD LW RAD FLUX: SURFACE	0	5	4	1	ULWRFsfc	W m-2
61	2422	DUST OPTICAL DEPTH	3	1	192, 193, 194, 195, 196, 197	1	DAOT038, DAOT044, DAOT055, DAOT067, DAOT087, DAOT102	unitless
62	3238	DEEP SOIL TEMPERATURE AFTER B.LAYER	2	0	25	106	VSOILM0_10cm , VSOILM10_35cm, VSOILM35_100cm, VSOILM100_200cm	K
63	8223	SOIL MOISTURE CONTENT IN A LAYER	2	0	3	106	TSOIL0_10cm, TSOIL10_35cm, TSOIL35_100cm, TSOIL100_200cm	m3 m-3

In UMRider, SOIL MOISTURE CONTENT IN A LAYER variable has been converted to Volumetric by dividing each layer by its depth in mm (0 to 10 cm layer depth is 100 mm, 10 to 35 cm layer depth is 250 mm, 35 to 100 cm layer depth is 650 mm and 100 to 200 cm layer depth is 1000 mm), so as to change unit from kg/m^2 to m^3/m^3. Also whose grid values of Volumetric Soil Moisture less than 0.005 are reset to 0.0051, since NOAH WRF model requires the soil moisture volumetric values to be less not than 0.005.

Appendix - C

NCUM Grib2 Local Table

Users of UMRider and readers of this technical report, can download local table from this link https://github.com/NCMRWF/UMRider/blob/master/tables/local/ncmr/v1/ncmr_grib2_local_table. Store the below lines as text file and name it as "name ncmr_grib2_local_table". This text file absolute path can be exported to an environment variable called "GRIB2TABLE" which is used by many software including wgrib2.

```
/* asterix (*) Indicates comment lines
*  Name : ncmr grib2 local table
*  Filename : ncmr_grib2_local_table
*  Institute : National Centre for Medium Range Weather Forecasting, India
*  Centre Code : 29, New Delhi
*  Local Table Version No : 1
*  Local Table Created By : Arulalan.T, Project Scientist - C
*  Contact : arulalan@ncmrwf.gov.in
*  Release Version : 0.1, No of Entries : 9, Date : 21-Jan-2016
*  Release Version : 0.2, No of New Entries : 1, Date : 12-Apr-2016
*  Release Version : 0.3, No of New Entries : 2, Date : 26-Apr-2016
*  Release Version : 0.4, No of New Entries : 10, Date : 10-Jan-2017
*  Release Version : 0.5, No of New Entries : 1, Date : 27-Apr-2018
*  Release Version : 0.6, No of New Entries : 2, Date : 19-Mar-2019
*  Release Version : 0.7, No of New Entries : 1, Date : 27-Sep-2019
*  Total No of Entries : 26
*  Export Command : export GRIB2TABLE=/path/to/localdir/ncmr_grib2_local_table
*  Once we exported, then wgrib2 & g2ctl.pl will be able to read these
*             local variables properly
*  Reference Link : http://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/user_grib2tables.html
*
struct gribtable_s {
   int disc;      // Section 0 Discipline
   int mtab_set;  // Section 1 Master Tables Version Number used by set_var
   int mtab_low;  // Section 1 Master Tables Version Number low range of tables
   int mtab_high; // Section 1 Master Tables Version Number high range of tables
   int cntr;      // Section 1 originating centre, used for local tables
   int ltab;      // Section 1 Local Tables Version Number
   int pcat;      // Section 4 Template 4.0 Parameter category
   int pnum;      // Section 4 Template 4.0 Parameter number
const char *name;
const char *desc;
const char *unit;
  };

* ParameterDiscipline : MasterTableVersionSet : MasterTableVersionStart : MasterTableVersionEnd :
Centre Code : LocalTablesVersion: ParameterCategory : ParameterNumber: VariableShortName :
VariableDescription : VariableUnit
*
* Comment lines end
*/
```

* ncmr grib2 local table entries begin
0:1:0:10:29:1:1:192:FOG:Fog Area Cover:%
0:1:0:10:29:1:0:205:WETBPT:Wet Bulb Potential Temperature: K
0:1:0:10:29:1:1:193:EVARSS:Evaporation Rate From Soil Surface: kg m^{-2} s^{-1}
0:1:0:10:29:1:1:194:EVARCA:Evaporation Rate From Canopy: kg m^{-2} s^{-1}
0:1:0:10:29:1:1:195:EVAROS:Evaporation Rate From Open Sea: kg m^{-2} s^{-1}
0:1:0:10:29:1:1:196:DENRR:Density * R * R:
0:1:0:10:29:1:4:192:NSWRFC:Net Short Wave Radiation Flux Corrected: W m^{-2}
0:1:0:10:29:1:4:194:DUVB:Direct UV-B Solar Flux: W m^{-2}
0:1:0:10:29:1:4:196:CSUSFS:Clear Sky Downward Solar Flux at Surface: W m^{-2}
0:1:0:10:29:1:4:197:CSULFS:Clear Sky Upward Solar Flux at Surface: W m^{-2}
0:1:0:10:29:1:4:198:CSUSFT:Clear Sky Upward Solar Flux at TOA: W m^{-2}
0:1:0:10:29:1:5:195:CSULFT:Clear Sky Upward Long Wave Flux at TOA:W m^{-2}
0:1:0:10:29:1:5:192:CSDLFS:Clear Sky Downward Long Wave Flux at Surface: W m^{-2}
0:1:0:10:29:1:15:192:RAREF:Radar Reflectivity: dBZ
0:1:0:10:29:1:15:193:ROHLENABL:Roughness Length After Boundary Layer: m
3:1:0:10:29:1:1:192:DAOT038:Dust Aerosol Optical Thickness at 0.38 µm: unit less
3:1:0:10:29:1:1:193:DAOT044:Dust Aerosol Optical Thickness at 0.44 µm: unit less
3:1:0:10:29:1:1:194:DAOT055:Dust Aerosol Optical Thickness at 0.55 µm: unit less
3:1:0:10:29:1:1:195:DAOT067:Dust Aerosol Optical Thickness at 0.67 µm: unit less
3:1:0:10:29:1:1:196:DAOT087:Dust Aerosol Optical Thickness at 0.87 µm: unit less
3:1:0:10:29:1:1:197:DAOT102:Dust Aerosol Optical Thickness at 1.02 µm: unit less
3:1:0:10:29:1:1:198:TCDAM:Total Column Dry Aerosols Mass: kg m^{-2}
0:1:0:10:29:1:6:201:VLCDC:Very Low Cloud Cover:%
0:1:0:10:29:1:6:202:TCDCRO:Total Cloud Cover Assuming Random Overlap:%
0:1:0:10:29:1:6:203:TCDCMRO:Total Cloud Cover Assuming Maximum Random Overlap:%
0:1:0:10:29:1:6:204:TCDVFA:Total Cloud Volume Fraction in Atmosphere Layer:%
0:1:0:10:29:1:6:205:LCDVFA:Liquid Cloud Volume Fraction in Atmosphere Layer:%
0:1:0:10:29:1:6:206:ICDVFA:Ice Cloud Volume Fraction in Atmosphere Layer:%
0:1:0:10:29:1:6:207:LIGHTFC:Ligtning Flash Count: unit less
2:1:0:10:29:1:0:231:SSFCWROR:Sub Surface Water Runoff Rate :kg m^{-2} s^{-1}
2:1:0:10:29:1:0:232:SUFCWR:Surface Upward Water Rate :kg m^{-2} s^{-1}
2:1:0:10:29:1:0:193:DHFS:Downward Heat Flux in Soil: W m^{-2}
* ncmr grib2 local table entries end