NMRF/TR/07/2024

सत्यमेव जयते

TECHNICAL REPORT

A utility for the vertical interpolation of UM sigma level fields to height above ground

Mansi Bhowmick, A. Jayakumar, Saji Mohandas and T.J. Anurose

August 2024

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

A utility for the vertical interpolation of UM sigma level fields to height above ground

Mansi Bhowmick, A. Jayakumar, Saji Mohandas and T.J. Anurose

August, 2024

National Centre for Medium Range Weather Forecasting
Ministry of Earth Sciences
A-50, Sector 62, Noida-201 309, 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/07/2024
3	Date of Publication	August 2024
4	Title of the document	A utility for the vertical interpolation of UM sigma level fields to height above ground.
5	Type of the document	Technical Report
6	Number of pages, figures, and Tables	13 Pages, 6 Figures
7	Authors	Mansi Bhowmick, A. Jayakumar, Saji Mohandas and T.J. Anurose
8	Originating Unit	National Centre for Medium Range Weather Forecasting (NCMRWF)
9	Abstract (brief)	NCMRWF Unified Model (NCUM) forecast products can be directly output with user-defined spatial or temporal specifications. However, vertical interpolation from the native hybrid levels is needed to get the values at required special height levels. There is a large demand from the various end-users for many model parameters at non-conventional or multiple near-surface and boundary levels. The python based utility used so far for vertical interpolation using the generic height levels is having many limitations and is found to be less accurate. This report describes a new post-processing utility which involves interpolation of meteorological parameter values from model levels to regular height levels above ground level as required by various stakeholders and user community. It is also suitable for the direct comparison of model output profiles with Radar and Radio-sonde data. The utility uses interpolation based on exner height and is found to be well matching with the native model levels, as demonstrated in this document.
10	References	2
11	Security classification	Unrestricted
12	Distribution	General

Table of contents

ABSTRACT

NCMRWF Unified Model (NCUM) forecast products can be directly output with user-defined spatial or temporal specifications. However, vertical interpolation from the native hybrid levels is needed to get the values at required special height levels. There is a large demand from the various end-users for many model parameters at non-conventional or multiple near-surface and boundary levels. The python based utility used so far for vertical interpolation using the generic height levels is having many limitations and is found to be less accurate. This report describes a new post-processing utility which involves interpolation of meteorological parameter values from model levels to regular height levels above ground level as required by various stakeholders and user community. It is also suitable for the direct comparison of model output profiles with Radar and Radio-sonde data. The utility uses interpolation based on exner height and is found to be well matching with the native model levels, as demonstrated in this document.

सारांश

राष्ट्रीय मध्यम अवधि मौसम पूर्वानुमान केंद्र यूनिफाइड मॉडल (एनसीयूएम) पूर्वानुमान उत्पादों को उपयोगकर्ता द्वारा परिभाषित स्थानिक अथवा अस्थायी विशिष्टताओं के साथ सीधे आउटपुट किया जा सकता है। हालाँकि, आवश्यक विशेष ऊंचाई स्तरों पर मान प्राप्त करने के लिए मूल संकर स्तरों से ऊर्ध्वाधर प्रक्षेप की आवश्यकता होती है। गैर-पारंपरिक या कई निकट-सतह और सीमा स्तरों पर कई मॉडल मापदंडों के लिए विभिन्न अंतिम-उपयोगकर्ताओं से बड़ी मांग है। सामान्य ऊँचाई स्तरों का उपयोग करके ऊर्ध्वाधर प्रक्षेप के लिए अब तक उपयोग की जाने वाली पायथन आधारित उपयोगिता में कई सीमाएँ हैं और यह कम सटीक पाई गई है। यह रिपोर्ट एक नई प्रसंस्करण के बाद की उपयोगिता का वर्णन करती है। इस रिपोर्ट में विभिन्न हितधारकों और उपयोगकर्ता समुदाय द्वारा आवश्यक मॉडल स्तरों से लेकर जमीनी स्तर से ऊपर नियमित ऊंचाई स्तरों तक मौसम संबंधी पैरामीटर मूल्यों का अंतर्वेशन शामिल है। यह रडार और रेडियो-सोंडे डेटा के साथ मॉडल आउटपुट प्रोफाइल की सीधी तुलना के लिए भी उपयुक्त है। उपयोगिता एक्सनर ऊंचाई के आधार पर इंटरपोलेशन का उपयोग करती है तथा इसे मूल मॉडल स्तरों के साथ अच्छी तरह से मेल खाते हुए पाया गया है (जैसा कि यहां दिखाया गया है)।

1. INTRODUCTION

The operational NWP products in NCMRWF are generated either at model levels that is sigma/rho levels or standard pressure levels. The model levels are terrain following height levels but have irregular intervals and non-standard height values above ground/sea level like 2.5m, 11.07m, 26.79m, 49.64m, 79.64m, 116.8m, 161.1m and so on. Thus, for application purposes data at model levels are not useful as such. Noting the request from the wind energy sector and utilisation of NWP products by the Army for artillery firing system, a need arose to provide products at well-resolved regular height levels in the boundary layer. Here regular height levels are referred to as levels like 5m, 10m, 30m, 50m, 70m, 100m, 150m, 200m, 250m, 300m, 350m, 400m and so on.

The existing operational version of the vertical interpolation method (Height-Based Interpolation termed as HBInterp from here onwards) interpolates model variables to regular height levels using a Python-based *stratify* function from the SciKit tools. The HBInterp method has successfully interpolated various variables to regular height levels till it is tested for wind speed in the boundary layer (Figure 1) where it is unable to produce the expected results. Therefore, the current work documents the development of a new vertical interpolation method based on exner height (termed as EBInterp).

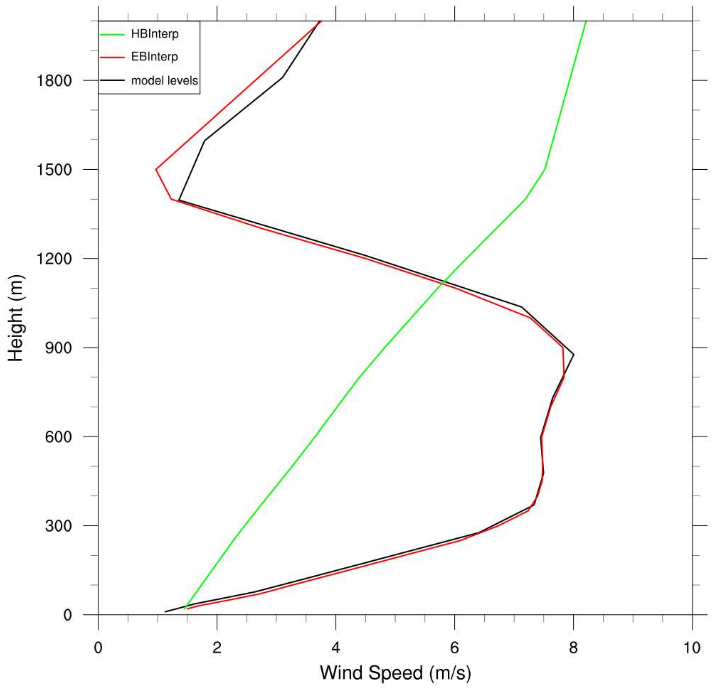

Figure 1: Comparison of wind speed profile in the boundary layer over Devlali using height-based and exner height-based interpolation.

2. INTERPOLATION METHOD

The formula used (eqn 1) for the exner height-based vertical interpolation method (EBInterp), is described below. The formula is extracted from the UKMO model code used to compute 50m wind components. The subroutine (vert_interp_mdi) in the model code can be found at path fcm_make_um/extract/um/src/control/grids/vert_interp_mdi.F90. The call location of the subroutine is fcm_make_um/ extract/ um/ src/ atmosphere/ dynamics_diagnostic/ dyn_diag.F90. This method uses cubic interpolation but instead of height, it uses exner height (*x1* and *x2*). A schematic of model height levels, exner height and regular height levels is shown in Figure 2. In the schematic, the red line indicates the desired regular height level and the dashed black lines denote the model height levels. Here, it is worth mentioning that interpolation is not possible below the first model level (h_0).

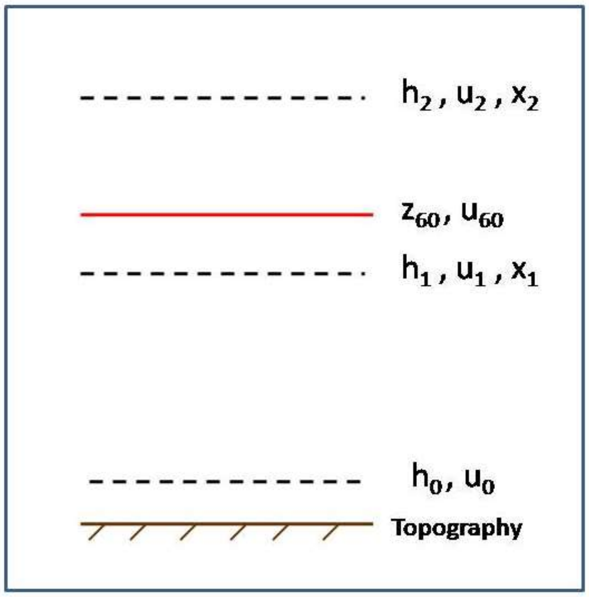

Figure 2: Schematic of model height levels (black) and required regular height levels (red).

The wind at required height above ground (U60) is derived as,

$$u_{60} = \frac{[(z_{60}-x_1)u_2]-[(z_{60}-x_2)u_1]}{[x_2-x_1]} \qquad \text{(eqn. 1)}$$

Where,

$z_{60\,=}\ height\ of\ desired\ level\ (e.g.\ 60m)$

$u_{60} = U\ component\ of\ wind\ at\ desired\ level$

$h_{0=} height\ of\ first\ model\ level$

$h_{1=} height\ of\ model\ level, below\ desired\ level$

$h_{2=}\ height\ of\ model\ level, above\ desired\ level$

$x_1 = exner\ height = h_1 - h_0$

$x_2 = h_2 - h_0$

$u_n = U\ component\ of\ wind\ at\ model\ level$

Appendix 1 describes the function call, arguments and return value. Appendix 2 lists the excerpts from the main python code.

3. VERIFICATION

The results of the EBInterp interpolation code are tested for location-specific, area-averaged and spatially for different types of orographic conditions such as inland, coast and over oceanic regions for both global (NCUM-G) and regional (NCUM-R) operational model simulations. The randomly chosen test date for all the evaluations is 22 Nov 2023 00UTC.
The observed vertical interpolation results for temperature and winds are captured reasonably well for location-specific (Figure 3) and small-area average cases (Figure 4). To capture inversions better using the EBInterp method, the resolution of the vertical coordinate should be taken high. The spatial plot of the 10m and 50m wind speed and direction are shown in Figure 5.

3.1 Location-specific global and regional profiles

The station selected for the comparison of vertical interpolation of temperature and U and V components of the wind field from global and regional UM is Bhopal. Figure 3 shows quite good similarity between the global and regional fields and temperature curves from model and

EBInterp methodology are exactly overlying with each other, while wind fields show very rare appearances of dashed curves with marginal differences with the solid curves.

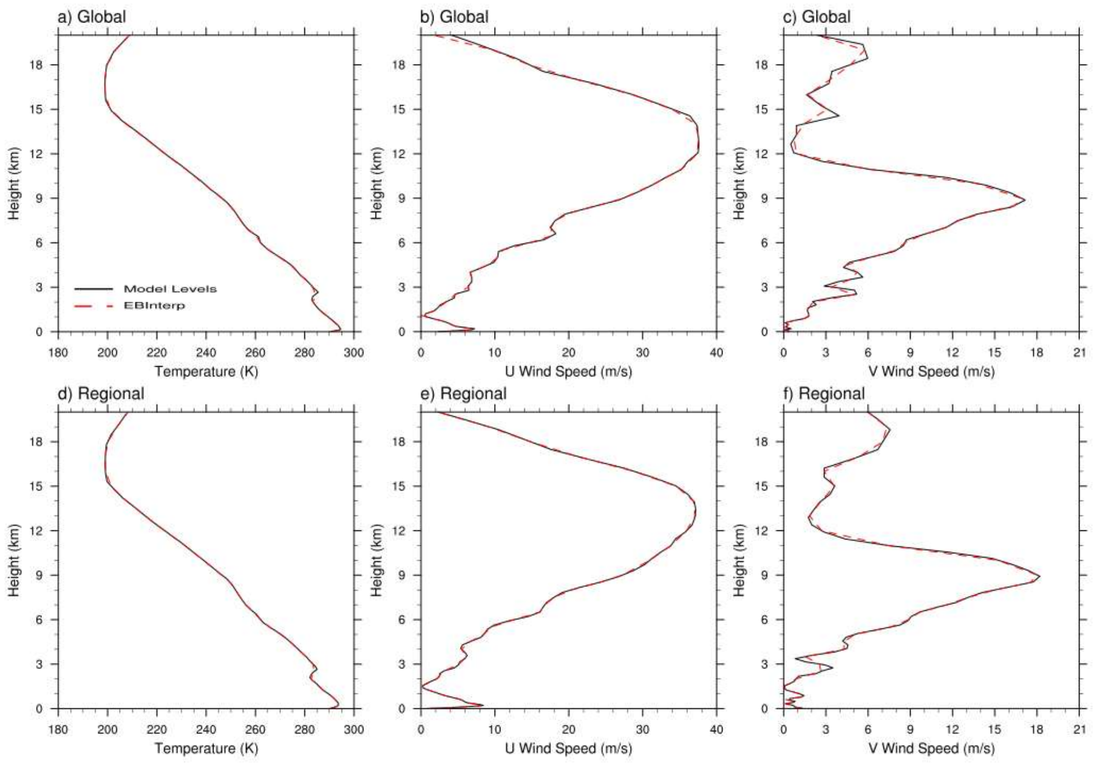

Figure 3: Model level and regular height level interpolated tropospheric profiles of temperature, u-wind speed and v-wind speed over Bhopal (23.26 °N/77.416 °E) from NCUM-G (Global) and NCUM-R (Regional).

3.2 Area average regional profiles for different topographic conditions

Area average (1° X 1°) locations located near Bhopal (23.26 °N/77.416 °E), Devlali (19.806 °N /73.776 °E) and Bay of Bengal (13 °N/ 88 °E) point are selected for deriving vertical profiles from the regional UM. The vertical profiles in Figure 5 show fairly useful similarities in temperature and wind speed.

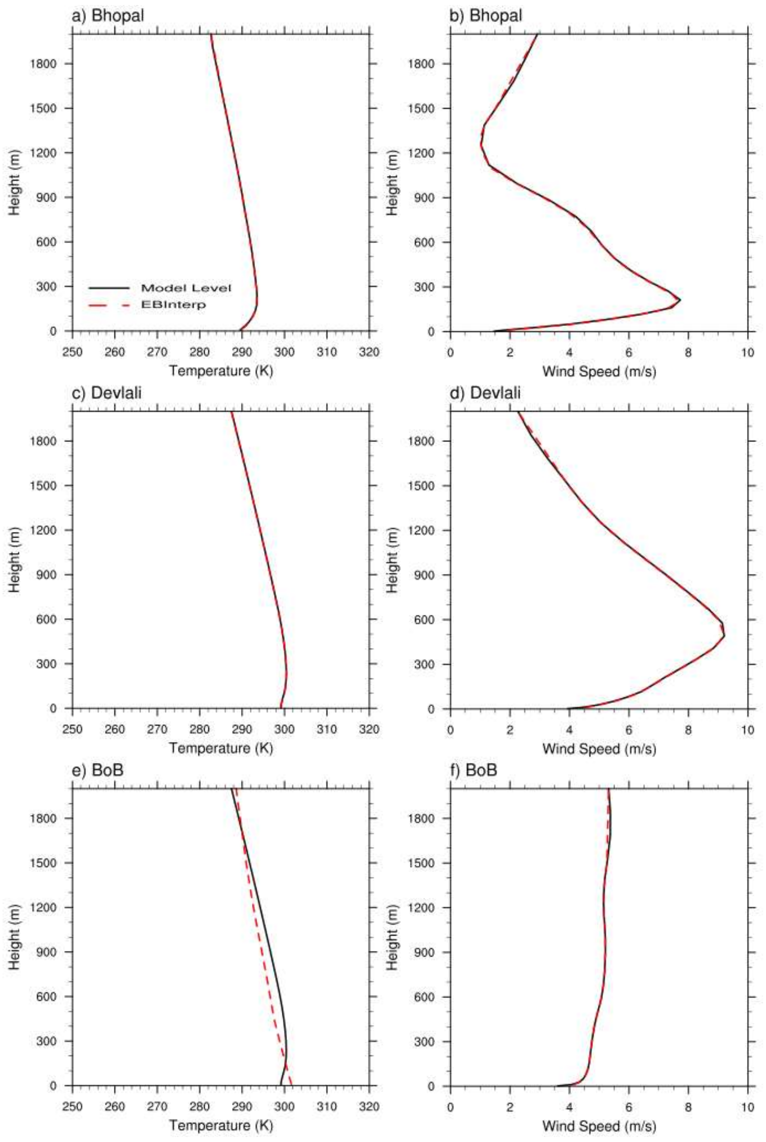

Figure 4: Area averaged (1ºX1º) EB interpolated and model height level boundary layer profiles of temperature, and wind speed over (a,b) Bhopal (Inland), (c,d) Devlali (Near west coast) and (e,f) Bay of Bengal (BoB, Ocean) using NCUM-R.

3.3 Spatial verification of speed and direction for 10m and 50m wind fields

To verify the spatial distribution of 10m and 50m wind derived from the utility against the model direct diagnostic output, Figure 5 and 6 displays the comparison for wind speed and direction from global and regional models. The output is well matching in general characteristics and patterns with each other. There are very few isolated pixels featuring slightly high peaks in EBInterp output compared with model output and can be ignored.

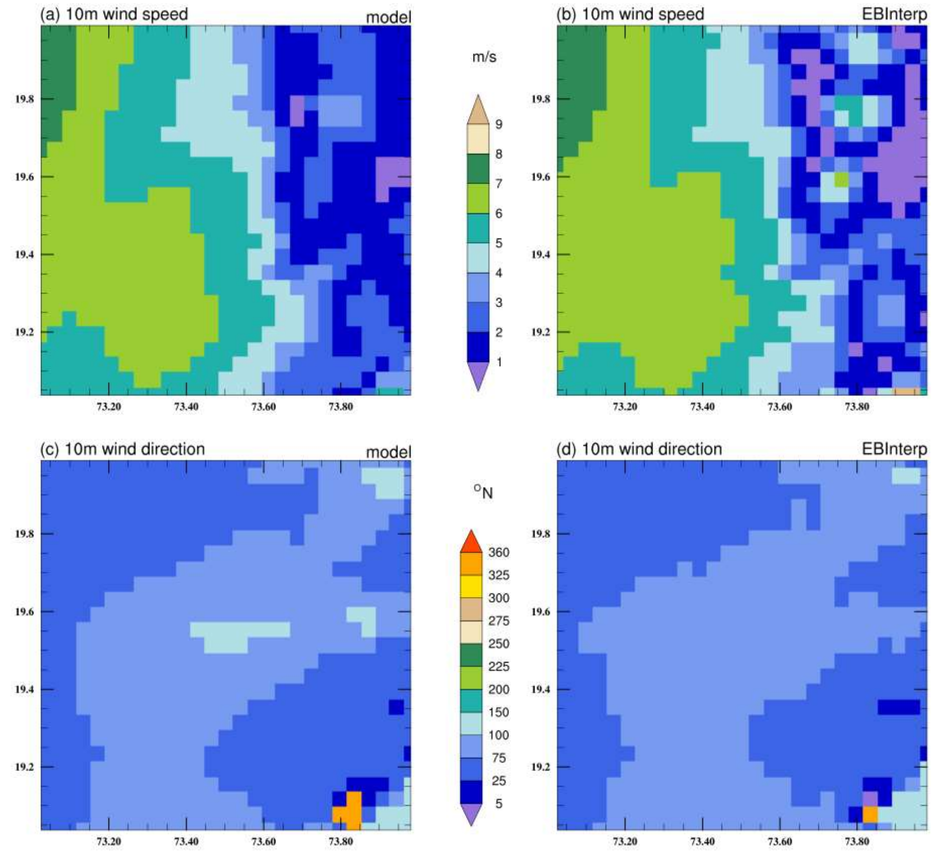

Figure 5: Comparison of 10-meter wind speed and direction from NCUM-R model and EBInterp interpolation near the location west coast, Devlali (19.806 °N/ 73.776 °E).

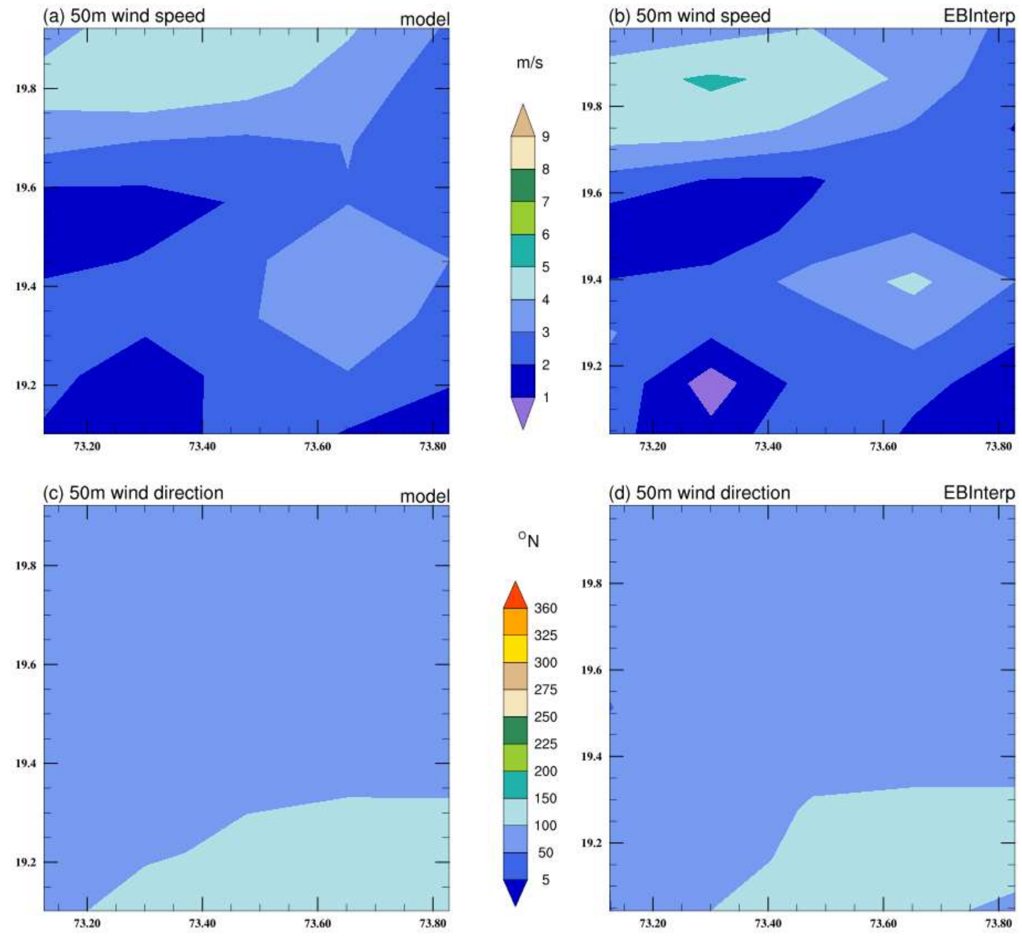

Figure 6: Comparison of 50-meter wind speed and direction from NCUM-G model and EBInterp interpolation near the location west coast, Devlali (19.806 °N/ 73.776 °E).

4. CONCLUDING REMARKS

An exner height-based interpolation method is developed as a post-processing utility using the 50m interpolation method and tested for various topographic conditions (Figure 4). Satisfactory results are observed related to point (Figure 3) and area average vertical profiles (Figure 4); however, some trivial discrepancies are noted in the spatial map of wind speed and direction in interpolated values when compared with model output wind fields at 10m and 50m (Figure 5 and Figure 6). This utility is soon implemented operationally within the software designed for the stakeholder's applications.

5. REFERENCES

The NCAR command Language (Version6.6.2)[Software],2019. Boulder, Colorado: UCAR/NCAR/CISL/TDD. http://dx.doi.org/10.5065/D6WD3XH5

Terry Davies (2023) Dynamical and Physical Diagnostic Calculations. Unified Model Documentation Paper 080,MetOffice, UK .

6. ACKNOWLEDGEMENTS

We sincerely acknowledge NCAR Command Language software for visualisation and Python for data analysis. We sincerely thank T. Arulalan for scripting the previous operational version of the present utility, part of which is used here.

APPENDIX 1

Description

Interpolation from model levels to desired height levels above ground level. The code is parallelized in time coordinate so requires ***mpi*** submission of job.

Load modules

```
module load gnu/pythonpackages/2.7.9 gnu/packagesuite/1
module load gnu/user-specific/rmedtoolbox_modules/1
module load gnu/dask/1.1.4
module load gnu/toolz/0.9.0
```

Prototype

sys.path.append('path to ml2hl_interp.py')

from *ml2hl_interp* import *modelLevel2RegularHeight*

newCube=modelLevel2RegularHeight(cube,targetHeight)

Arguments

cube: It should be a 4-d python datacube with all its attributes including level_height.

cube Dimensions (time,model_levels,latitude,longitude)

target Height: 1-d array with desired height levels in unit, meter e.g.

[20,30,50,70,100,150,200,300,400,800,1000,2500,5000,8000,10000]

Return value

Return value is a 4-d cube with dimensions (time,height,latitude,longitude)

APPENDIX 2

```
##Interpolating from model levels to regular height levels *** ****
## Script prepared by Mansi Bhowmick with the guidance from  Dr.T.J.
Anurose and  Dr.A. Jayakumar
####
##For more details please contact mansibhowmick@gmail.com or
jkumar@ncmrwf.gov.in
#Dec, 2023

#Import modules
import numpy, iris
from cf_units import Unit
import os, sys
import iris
import numpy as np
from iris.coords import DimCoord
from iris.cube import Cube
from netCDF4 import Dataset
import netCDF4 as nc
import dask

#input 3D cube (without time coordinate)
#output 3D cube (data only)

@dask.delayed
def ht_converter(field_data,ht_desired):

 z=np.array(ht_desired)
 bl_N=z.shape[0]
 nlat=field_data.shape[1]
 nlon=field_data.shape[2]
 u_desired = np.zeros(shape=(bl_N,nlat,nlon),dtype=float)
 z_desired = np.zeros(shape=(bl_N,nlat,nlon),dtype=float)

 #Creating 3D array from 1D array
 for i in range(0,bl_N):
  for m in range(0,nlat):
   for n in range(0,nlon):
    z_desired[i,m,n]=z[i]

 #compute exner height x
 ht=field_data.coord('level_height').points
 #print 'Model heights'
 #print ht
 x=ht-ht[0]
 M=ht.shape[0]

 #Creating 3D array from 1D array
 x_3d = np.zeros(shape=(M,nlat,nlon),dtype=float)
 for i in range(0,M):
  for m in range(0,nlat):
   for n in range(0,nlon):
    x_3d[i,m,n]=x[i]

 field_data=field_data.data

 #cubic interpolation in 3D using exner height
 for i in range(0,bl_N):
  for k in range(0,M):
   if (ht[k] > z[i]):
    lev1=k-1
    lev2=k
    if (lev1<0):
     lev1=lev2
     print "Interpolation Error: First regular height level is less 
than first model height level"
     print "First model height level =",ht[lev1]
     sys.exit()
    break
  u_desired[i,:,:]=(((z_desired[i,:,:]-
x_3d[lev1,:,:])*field_data[lev2,:,:])-((z_desired[i,:,:]-
x_3d[lev2,:,:])*field_data[lev1,:,:]))/(x_3d[lev2,:,:]-
x_3d[lev1,:,:])

 return u_desired




def modelLevel2RegularHeight(cube,targetHeight):
 #Parallelise the time coordinate using dask
 task_list=[]
 nrecord=cube.shape[0]     #time coordinate

 for i in range(nrecord):
  u_desired=ht_converter(cube[i,:,:,:],targetHeight)
  task_list.append(u_desired)

 task=dask.delayed()(task_list)
 compute_u=task.compute()

 #newCube is 4D data
 newCube=np.array(compute_u)

# appending coordinates---------
 co =  cube.dim_coords
 axidx = 0
 for i in range(len(co)):
  if co[i].standard_name == 'model_level_number':
   axidx = i
   break
 newCube = numpy.ma.masked_invalid(newCube)
 print "converted to", newCube.data.shape
 attr = cube.attributes
 lname = cube.long_name
 sname = cube.standard_name
 cm = cube.cell_methods[0] if cube.cell_methods else None
 unit = cube.units
 # new coordinate
 height = iris.coords.DimCoord(targetHeight, standard_name='height',
units=Unit('m'), attributes={'positive': 'up', 'comments': 'height
above orography'})
 numpy.ma.set_fill_value(newCube, 9.999e+20)
 # create new cube
 newCube = iris.cube.Cube(data=newCube, units=unit,
standard_name=sname,
                        long_name=lname, attributes=attr)#,
cell_methods=(cm,))
 # add dimension coords
 newCube.add_dim_coord(cube.coord('time'), 0)
 newCube.add_dim_coord(height, 1)
 newCube.add_dim_coord(co[-2], 2) # latitude
 newCube.add_dim_coord(co[-1], 3) # longitude
 # add aux coords
 newCube.add_aux_coord(cube.coord('forecast_period'), 0)
 newCube.add_aux_coord(cube.coord('forecast_reference_time'))

 #print newCube
 print 'Successful execution of ml2hl_interp.py'
 #return 4D cube
 return newCube
```