merra Documentation

Felix Zaussinger

Mar 04, 2019

Contents

The package provides readers and converters for the Land Surface Diagnostics within the Modern-Era Retrospective analysis for Research and Applications version 2 (MERRA-2). MERRA-2 is a NASA atmospheric reanalysis integrating satellite data assimilation and aims at historical climate analyses. MERRA-2 covers Land Surface Diagnostics for the period 1980-present at 0.5 ° x 0.625 ° spatial- and 1-hourly temporal-resolution.

The structure of the package is as follows:

- grid.py : implements the asymmetrical GMAO 0.5 x 0.625 grid
- interface.py : classes for reading a single image, image stacks and time series
- reshuffle.py : provides a command line utility for reshuffling a stack of 1-hourly sampled native images to time series format with an arbitraty temporal sampling between 1-hour and daily
- download.py : command line utility for downloading MERRA-2 data from the NASA GES DISC datapool

CHAPTER 1

Installation

For developers, it is recommended to first clone the repository and then use the provided environment.yml file to install all needed conda and pip dependencies:

```
git clone https://github.com/TUW-GEO/merra.git --recursive
cd merra
conda create -n merra python=3.7 # or any supported python version
source activate merra
conda update -f environment.yml
python setup.py develop
```

CHAPTER 2

Supported Products

- M2T1NXLND: MERRA-2 tavg1_2d_lnd_Nx: MERRA-2 2d, 1-Hourly, Time-Averaged, Single-Level, Assimilation, Land Surface Diagnostics V5.12.4

CHAPTER 3

Contribute

We are happy if you want to contribute. Please raise an issue explaining what is missing or if you find a bug. We will also gladly accept pull requests against our master branch for new features or bug fixes.

3.1 Guidelines

If you want to contribute please follow these steps:

- Fork the merra repository to your account
- make a new feature branch from the merra master branch
- Add your feature
- please include tests for your contributions in one of the test directories We use py.test so a simple function called test_my_feature is enough
- submit a pull request to our master branch

3.2 Reading MERRA-2 images

Reading of the MERRA-2 netcdf files can be done in two ways:

3.2.1 1) Reading by file name

```
import os
from datetime import datetime
from merra.interface import MerraImage

# parameters to read
param_list = ['SFMC', 'RZMC', 'PRECTOTLAND', 'TSOIL1']
```

(continues on next page)

(continued from previous page)

```
# timestamp (needed because there are 24 hourly simulations within
# each file (3D-stack), so you need to choose one to return a 2D image)
timestamp = datetime(2018, 10, 1, 0, 30)

# the class is initialized with the exact filename.
img = MerraImage(os.path.join(os.path.dirname(__file__),
                              'merra-test-data',
                              'M2T1NXLND.5.12.4',
                              '2018',
                              '10',
                              'MERRA2_400.tavg1_2d_lnd_Nx.20181001.nc4'),
                         parameter=param_list)

# reading returns an image object which contains a data dictionary
# with one array per parameter. The returned data is a global image/array
# with shape (361, 576)
image = img.read(timestamp=timestamp)
data = image.data
```

3.2.2 2) Reading by date

All the MERRA-2 data in a directory structure can be accessed by date. The filename is automatically built from the given date.

```
from merra.interface import MerraImageStack

# parameters to read
param_list = ['SFMC', 'RZMC', 'PRECTOTLAND', 'TSOIL1']

# initializes an image stack class given the path to the data directory
# the class knows about the default folder structure down the line
img_stack = MerraImageStack(data_path=os.path.join(os.path.dirname(__file__),
                                                   'merra-test-data',
                                                   'M2T1NXLND.5.12.4'),
                            parameter=param_list)

# timestamp
timestamp = datetime(2018, 10, 1, 0, 30)

# read one image out of the stack at specific timestamp
image = img_stack.read(timestamp=timestamp)
```

For reading all image between two dates the `merra.interface.MerraImageStack.iter_images()` iterator can be used.

CHAPTER 4

Variables of interest for MERRA-2

A full list of variable names can be found in the MERRA-2 README provided by NASA.

Short name	Long name	Parameter	Resolution	Depth [m]	Units
SFMC	water_surface_layer	Soil moisture	0.5° x 0.625°	0.00 - 0.05	[m-3 m-3]
RZMC	water_root_zone	Soil moisture	0.5° x 0.625°	0.10 - 1.00	[m-3 m-3]
GWETTOP	surface_soil_wetness	Soil moisture	0.5° x 0.625°	0.00 - 0.05	[]
GWETROOT	root_zone_soil_wetness	Soil moisture	0.5° x 0.625°	0.10 - 1.00	[]
GWETPROF	ave_prof_soil_moisture	Soil moisture	0.5° x 0.625°	1.34 - 8.53	[]
PRECTOTLAND	Total_precipitation_land	Land precipitation rate	0.5° x 0.625°	0	[kg m-2 s-1]
PRECSNOLAND	snowfall_land	Snow precipitation rate	0.5° x 0.625°	0	[kg m-2 s-1]
SNOMAS	Total_snow_storage_land	Land snow storage land	0.5° x 0.625°	0	[kg m-2]
TSOIL1	soil_temperatures_layer_1	Soil temperatures layer 1	0.5° x 0.625°	0. • 0.0988	[K]
TSOIL2	soil_temperatures_layer_2	Soil temperatures layer 2	0.5° x 0.625°	0. • 0.1952	[K]
TSOIL3	soil_temperatures_layer_3	Soil temperatures layer 3	0.5° x 0.625°	0. • 0.3859	[K]
TSOIL4	soil_temperatures_layer_4	Soil temperatures layer 4	0.5° x 0.625°	0. • 0.7626	[K]
TSOIL5	soil_temperatures_layer_5	Soil temperatures layer 5	0.5° x 0.625°	0. • 1.5071	[K]
TSOIL6	soil_temperatures_layer_6	Soil temperatures layer 6	0.5° x 0.625°	0. • 10.000	[K]
TSURF	surface_temperature_of_land_under_snow	Surface temperature	0.5° x 0.625°	0	[K]

CHAPTER 5

Conversion to time series format

For a lot of applications it is favorable to convert the image based format into a format which is optimized for fast time series retrieval. This is what we often need for e.g. validation studies. This can be done by stacking the images into a netCDF file and choosing the correct chunk sizes or a lot of other methods. We have chosen to do it in the following way:

- Store only the reduced gaußian grid points since that saves space.
- Further reduction the amount of stored data by saving only land points if selected.
- Store the time series in netCDF4 in the Climate and Forecast convention Orthogonal multidimensional array representation
- Store the time series in 5x5 degree cells. This means there will be 2566 cell files (without reduction to land points) and a file called `grid.nc` which contains the information about which grid point is stored in which file. This allows us to read a whole 5x5 degree area into memory and iterate over the time series quickly.

This conversion can be performed using the `merra_repurpose` command line program. An example would be:

```
merra_repurpose /merra2_data /timeseries/data -s 2000-01-01 -e 2018-11-30 --
↪parameters SFMC RZMC --temporal_sampling 6
```

Which would take MERRA-2 data stored in `/merra2_data` from January 1st 2000 to November 30th 2018 and store the parameters for 6-hourly sampled surface (SFMC) and root zone soil moisture (RZMC) as time series in the folder `/timeseries/data`.

Conversion to time series is performed by the repurpose package in the background. For custom settings or other options see the repurpose documentation and the code in `merra.reshuffle`.

Note: If a `RuntimeError: NetCDF: Bad chunk sizes.` appears during reshuffling, consider downgrading the netcdf4 library via:

```
conda install -c conda-forge netcdf4=1.2.2
```

if you are on Python 2.* and

```
conda install -c conda-forge netcdf4=1.2.8
```

if you are using Python 3.*.

5.1 Reading converted time series data

For reading the data the `merra_repurpose` command produces the class `MerraTs`:

```
from merra.interface import MerraTs

# specify path to data folder
path = '../timeseries/data'

# specify location lon and lat
lon, lat = (16.375, 48.125)

# initialize the time series class
merra_reader = MerraTs(ts_path=path,
                       ioclass_kws={'read_bulk':True},
                       parameters=['SFMC'])

# read SFMC time series at the location
ts = merra_reader.read(lon, lat)
```

CHAPTER 6

Contents

6.1 Reading MERRA-2 images

Reading of the MERRA-2 netcdf files can be done in two ways:

6.1.1 1) Reading by file name

```
import os
from datetime import datetime
from merra.interface import MerraImage

# parameters to read
param_list = ['SFMC', 'RZMC', 'PRECTOTLAND', 'TSOIL1']

# timestamp (needed because there are 24 hourly simulations within
# each file (3D-stack), so you need to choose one to return a 2D image)
timestamp = datetime(2018, 10, 1, 0, 30)

# the class is initialized with the exact filename.
img = MerraImage(os.path.join(os.path.dirname(__file__),
                              'merra-test-data',
                              'M2T1NXLND.5.12.4',
                              '2018',
                              '10',
                              'MERRA2_400.tavg1_2d_lnd_Nx.20181001.nc4'),
                         parameter=param_list)

# reading returns an image object which contains a data dictionary
# with one array per parameter. The returned data is a global image/array
# with shape (361, 576)
image = img.read(timestamp=timestamp)
data = image.data
```

6.1.2 2) Reading by date

All the MERRA-2 data in a directory structure can be accessed by date. The filename is automatically built from the given date.

```
from merra.interface import MerraImageStack

# parameters to read
param_list = ['SFMC', 'RZMC', 'PRECTOTLAND', 'TSOIL1']

# initializes an image stack class given the path to the data directory
# the class knows about the default folder structure down the line
img_stack = MerraImageStack(data_path=os.path.join(os.path.dirname(__file__),
                                                   'merra-test-data',
                                                   'M2T1NXLND.5.12.4'),
                            parameter=param_list)

# timestamp
timestamp = datetime(2018, 10, 1, 0, 30)

# read one image out of the stack at specific timestamp
image = img_stack.read(timestamp=timestamp)
```

For reading all image between two dates the `merra.interface.MerraImageStack.iter_images()` iterator can be used.

6.2 Variables of interest for MERRA-2

A full list of variable names can be found in the MERRA-2 README provided by NASA.

Short name	Long name	Parameter	Resolution	Depth [m]	Units
SFMC	water_surface_layer	Soil moisture	0.5° x 0.625°	0.00 - 0.05	[m-3 m-3]
RZMC	water_root_zone	Soil moisture	0.5° x 0.625°	0.10 - 1.00	[m-3 m-3]
GWETTOP	surface_soil_wetness	Soil moisture	0.5° x 0.625°	0.00 - 0.05	[]
GWETROOT	root_zone_soil_wetness	Soil moisture	0.5° x 0.625°	0.10 - 1.00	[]
GWETPROF	ave_prof_soil_moisture	Soil moisture	0.5° x 0.625°	1.34 - 8.53	[]
PRECTOTLAND	Total_precipitation_land	Land precipitation rate	0.5° x 0.625°	0	[kg m-2 s-1]
PRECSNOLAND	snowfall_land	Snow precipitation rate	0.5° x 0.625°	0	[kg m-2 s-1]
SNOMAS	Total_snow_storage_land	Land snow storage land	0.5° x 0.625°	0	[kg m-2]
TSOIL1	soil_temperatures_layer_1	Soil temperatures layer 1	0.5° x 0.625°	0. • 0.0988	[K]
TSOIL2	soil_temperatures_layer_2	Soil temperatures layer 2	0.5° x 0.625°	0. • 0.1952	[K]
TSOIL3	soil_temperatures_layer_3	Soil temperatures layer 3	0.5° x 0.625°	0. • 0.3859	[K]
TSOIL4	soil_temperatures_layer_4	Soil temperatures layer 4	0.5° x 0.625°	0. • 0.7626	[K]
TSOIL5	soil_temperatures_layer_5	Soil temperatures layer 5	0.5° x 0.625°	0. • 1.5071	[K]
TSOIL6	soil_temperatures_layer_6	Soil temperatures layer 6	0.5° x 0.625°	0. • 10.000	[K]
TSURF	surface_temperature_of_land_incl_snow	Surface temperature	0.5° x 0.625°	0	[K]

6.3 Conversion to time series format

For a lot of applications it is favorable to convert the image based format into a format which is optimized for fast time series retrieval. This is what we often need for e.g. validation studies. This can be done by stacking the images into a netCDF file and choosing the correct chunk sizes or a lot of other methods. We have chosen to do it in the following way:

- Store only the reduced gaußian grid points since that saves space.
- Further reduction the amount of stored data by saving only land points if selected.
- Store the time series in netCDF4 in the Climate and Forecast convention Orthogonal multidimensional array representation
- Store the time series in 5x5 degree cells. This means there will be 2566 cell files (without reduction to land points) and a file called `grid.nc` which contains the information about which grid point is stored in which file.

This allows us to read a whole 5x5 degree area into memory and iterate over the time series quickly.

This conversion can be performed using the `merra_repurpose` command line program. An example would be:

```
merra_repurpose /merra2_data /timeseries/data -s 2000-01-01 -e 2018-11-30 --
↪parameters SFMC RZMC --temporal_sampling 6
```

Which would take MERRA-2 data stored in `/merra2_data` from January 1st 2000 to November 30th 2018 and store the parameters for 6-hourly sampled surface (SFMC) and root zone soil moisture (RZMC) as time series in the folder `/timeseries/data`.

Conversion to time series is performed by the repurpose package in the background. For custom settings or other options see the repurpose documentation and the code in `merra.reshuffle`.

Note: If a `RuntimeError: NetCDF: Bad chunk sizes.` appears during reshuffling, consider downgrading the netcdf4 library via:

```
conda install -c conda-forge netcdf4=1.2.2
```

if you are on Python 2.* and

```
conda install -c conda-forge netcdf4=1.2.8
```

if you are using Python 3.*.

6.3.1 Reading converted time series data

For reading the data the `merra_repurpose` command produces the class `MerraTs`:

```
from merra.interface import MerraTs

# specify path to data folder
path = '../timeseries/data'

# specify location lon and lat
lon, lat = (16.375, 48.125)

# initialize the time series class
merra_reader = MerraTs(ts_path=path,
                       ioclass_kws={'read_bulk':True},
                       parameters=['SFMC'])

# read SFMC time series at the location
ts = merra_reader.read(lon, lat)
```

6.4 License

```
Copyright (c) 2019, TU Wien
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
```

(continues on next page)

(continued from previous page)

```
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of ascat nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

6.5 Developers

- Felix Zaussinger <felix.zaussinger@geo.tuwien.ac.at>

6.6 Changelog

6.6.1 Version 0.1

- First release. Support for downloading, reading and writing MERRA-2 M2T1NXLND.5.12.4 data.

6.7 merra

6.7.1 merra package

Submodules

merra.download module

The download module implements a command line script for downloading MERRA2 reanalysis data from the NASA GESDISC repository.

merra.download.**folder_get_version_first_last**(*root*, *fmt='MERRA2_{stream}.tavg1_2d_lnd_Nx.{time:%Y%m%d}.*
subpaths=['{time:%Y}', '{time:%m}'])

Get product version and first and last product which exists under the root folder.

Parameters

- **root** (*string*) – Root folder on local filesystem

- **fmt** (*string, optional*) – formatting string
- **subpaths** (*list, optional*) – format of the subdirectories under root.

Returns

- **version** (*string*) – Found product version
- **start** (*datetime.datetime*) – First found product datetime
- **end** (*datetime.datetime*) – Last found product datetime

merra.download.**get_first_folder**(*root*, *subpaths*)

...

Parameters

- **root** (*string*) – ...
- **subpaths** (*list of strings*) – ...

Returns **directory** – ...

Return type list of strings

merra.download.**get_first_formatted_dir_in_dir**(*folder*, *fmt*)

Get the (alphabetically) first directory in a directory which can be formatted according to fmt.

Parameters

- **folder** (*string*) – path to folder
- **fmt** – formatting rule

Returns **first_elem** – path to last directory

Return type string

merra.download.**get_last_folder**(*root*, *subpaths*)

...

Parameters

- **root** (*string*) – ...
- **subpaths** (*list of strings*) – ...

Returns **directory** – ...

Return type list of strings

merra.download.**get_last_formatted_dir_in_dir**(*folder*, *fmt*)

Get the (alphabetically) last directory in a directory which can be formatted according to fmt.

Parameters

- **folder** (*string*) – path to folder
- **fmt** – formatting rule

Returns **last_elem** – path to last directory

Return type string

merra.download.**get_start_date**(*product*)

Define start date of product version.

Parameters **product** (*string*) – product specification

Returns **datetime** – timestamp of start date

Return type datetime.datetime

merra.download.**main**(*args*)

merra.download.**parse_args**(*args*)
Parse command line parameters for recursive download

Parameters **args** (*list of strings*) – command line parameters

Returns **args** – command line parameters

Return type argparse.Namespace object

merra.download.**run**()

merra.grid module

The grid module implements the asymmetrical GMAO 0.5 x 0.625 grid used in MERRA2 as a pygeogrids BasicGrid instance.

merra.grid.**create_merra_cell_grid**()
Function creates the asymmetrical GMAO 0.5 x 0.625 grid as a BasicGrid instance.

Returns

Return type BasicGrid instance

merra.interface module

merra.reshuffle module

Module contents

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index