



Numerical Weather Prediction (NWP)

Theories



Numerical Weather Prediction

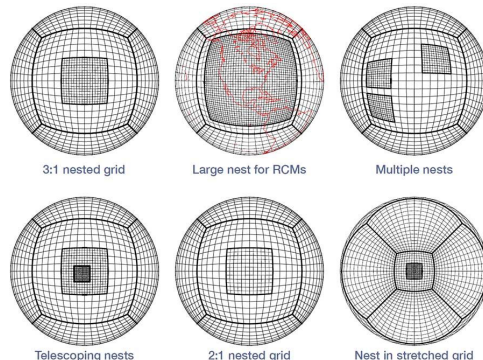
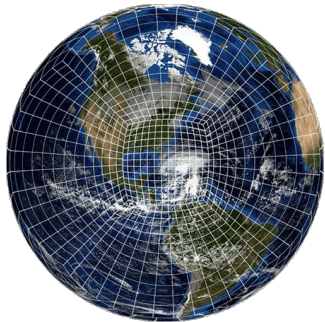
Skamarock 2012
MPAS-A seminal paper

Governing Partial Differential Equations,

- Fluid dynamics
- Thermodynamics
- Moisture, source & sink terms

=> next slides

Many models and gridings for solving PDEs:
Just pick an example:



GFS-FV3 model
<https://www.gfdl.noaa.gov/fv3/>

$$\begin{aligned} \frac{\partial \mathbf{V}_H}{\partial t} = & -\frac{\rho_d}{\rho_m} \left[\nabla_\zeta \left(\frac{p}{\xi_z} \right) - \frac{\partial \mathbf{z}_H p}{\partial \zeta} \right] - \boldsymbol{\eta} \mathbf{k} \times \mathbf{V}_H \\ & - \mathbf{v}_H \nabla_\zeta \cdot \mathbf{V} - \frac{\partial \Omega \mathbf{v}_H}{\partial \zeta} - \rho_d \nabla_\zeta K \\ & - eW \cos \alpha_r - \frac{\mathbf{v}_H W}{r_e} + \mathbf{F}_{\mathbf{V}_H}, \end{aligned} \quad (3)$$

$$\begin{aligned} \frac{\partial W}{\partial t} = & -\frac{\rho_d}{\rho_m} \left[\frac{\partial p}{\partial \zeta} + g \tilde{\rho}_m \right] - (\mathbf{V} \cdot \mathbf{v} W)_\zeta + \frac{uU + vV}{r_e} \\ & + e(U \cos \alpha_r - V \sin \alpha_r) + F_W, \end{aligned} \quad (4)$$

$$\frac{\partial \Theta_m}{\partial t} = -(\mathbf{V} \cdot \mathbf{V} \theta_m)_\zeta + F_{\Theta_m}, \quad (5)$$

$$\frac{\partial \tilde{\rho}_d}{\partial t} = -(\mathbf{V} \cdot \mathbf{V})_\zeta, \quad \text{and} \quad (6)$$

$$\frac{\partial Q_j}{\partial t} = -(\mathbf{V} \cdot \mathbf{V} q_j)_\zeta + F_{Q_j}. \quad (7)$$

Governing Equations

<u>Rate of change of something</u>	=	<u>terms related to itself and some other things in neighbouring positions</u>
<u>Derivatives with respect to time</u>	has a relation to	<u>Spatial operators</u> Divergence Gradient Curl

Systems of equations - variables are "coupled".

Dry atmosphere: V , W , θ , ρ

- Horizontal wind
- Vertical wind
- Potential temperature
- Density

Wet atmosphere: add q_v , q_c , q_i , q_r , ...

- Mixing ratio of moisture in different states

Relations:

Newton's classical mechanics:

- Conservation of momentum
- Conservation of mass (of dry air, moisture of different states)

Thermodynamics:

- Ideal gas law
- Conservation of energy (between heat and kinetic energy)

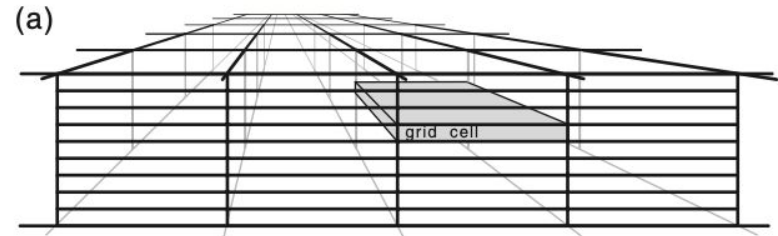
... and external forcing

Rectangular Grid - Dynamics

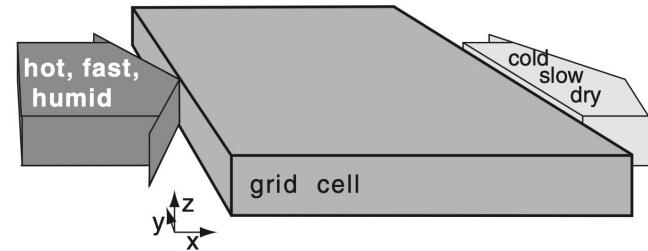
Figure 20.3

(a) The forecast **domain** (the portion of atmosphere we wish to forecast) is split into discrete grid cells, such as the shaded one. The 3-D grid cells are relatively thin, with sizes on the order of 10s m in the vertical, and 10s km in the horizontal.

(b) Enlargement of the shaded grid cell, illustrating one **dynamics** process (advection in the x-direction). Namely, the resolved U wind is blowing in hot, fast, humid air from the up-wind neighboring grid cell, and is blowing out colder, slower, drier air into the downwind neighboring cell. Simultaneously, advection could be occurring by the V and W components of wind (not shown). Not shown are other resolved forcings, such as Coriolis and pressure-gradient forces.



(b) Dynamics



Stull 2017 Practical Meteorology: An Algebra-based Survey of Atmospheric Science

https://www.eoas.ubc.ca/books/Practical_Meteorology/

Chapter 20 Numerical Weather Prediction (NWP)

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

By the way, a great textbook

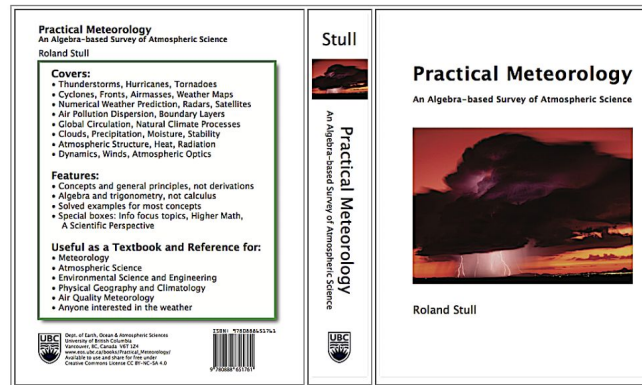
https://www.eoas.ubc.ca/books/Practical_Meteorology/

Practical Meteorology:

An Algebra-based Survey of Atmospheric Science

Copyright © 2017, 2018 by Roland Stull.

Available to use and share for [free](#) under a Creative Commons License.
(Click on chapter links below to get pdfs of individual chapters.
See bottom of this web page for license details.)



Cover and spine photo copyright © by Warren Faidley / weatherstock.com
Used with permission.

Stull, R., 2017: "Practical Meteorology: An Algebra-based Survey of Atmospheric Science" -version 1.02b. Univ. of British Columbia. 940 pages. isbn 978-0-88865-283-6 .

	Pages
Front cover , back cover-v1.02b & spine	I-XIV
Title-v1.02b, Contents-v1.02b, Preface-v1.02	
Chapters (as pdf files designed for printing or viewing)	
1. Atmospheric Basics -v1.02b	1-26
2. Solar & Infrared Radiation - v1.02b	27-52
3. Thermodynamics -v1.02b	53-86
4. Water Vapor - v1.02b	87-118
5. Atmos. Stability -v1.02b (& thermo diagrams)	119-158
6. Clouds - v1.02b	159-184
7. Precipitation Processes - v 1.02b	185-218
8. Satellites & Radar - v1.02b	219-266
9. Weather Reports & Map Analysis - v1.02b	267-288
10. Atmospheric Forces & Winds - v1.02b	289-328
11. General Circulation - v1.02b	329-388
12. Fronts & Airmasses - v1.02b	389-424
13. Extratropical Cyclones - v1.02b	425-480
14. Thunderstorm Fundamentals - v1.02b	481-544
15. Thunderstorm Hazards - v1.02b	545-602
16. Tropical Cyclones - v1.02b	603-644
17. Regional Winds - v1.02b	645-686
18. Atmospheric Boundary Layer - v1.02b	687-722
19. Pollutant Dispersion - v1.02b	723-744
20. Numerical Weather Prediction (NWP) - v1.02b	745-792
21. Natural Climate Processes - v1.02c	793-832
22. Atmospheric Optics - v1.02b	833-868
Appendices	
A. Scientific Tools - v1.02b	869-878
B. Constants & Conversion Factors - v1.02b	879-880
C. Notation - v1.03 (Draft version of a New Appendix)	C881-C888
Index & Errata	
Index- v1.02b	881-926
Errata & Enhancements for PrMet - v1.02	

Acknowledgement -
Credit: Roland Stull.

License: Creative Commons
Attribution-NonCommercial-Share
Alike 4.0 International License.

Creative Commons License:
 Creative Commons License

"Practical Meteorology: An Algebra-based Survey of Atmospheric Science" and "Meteorology for Scientists and Engineers, 3rd Edition" by Roland Stull are licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. Under this license, you are free to:

Share — copy, print and redistribute the material in any medium or format
Adapt — translate into another language, remix, transform, and build upon the material

Under the following terms:

Attribution — You must give appropriate credit to Roland Stull, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests that Stull endorses you or your use.

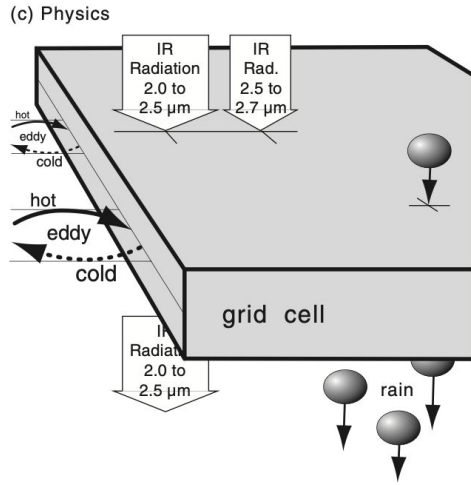
NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you translate into another language, remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

See exclusions on the back of the title page.

https://www.eoas.ubc.ca/books/Practical_Meteorology/
Last updated Nov 2018 by R. Stull

Physics Parameterization for unresolved physical processes



Source & Sink terms,
or called
Forcing terms,
in the
governing PDEs

Example:

Table 1: Physics Parameterization Schemes used in the experiments

Physics	Scheme
Convection	Scale-aware New Tiedtke (MPAS-A v6.x-openacc branch)
Microphysics	WSM6 (MPAS-A 7.0 = WRF 4.1)
Land surface	Noah (MPAS-A 7.0 = WRF 4.0.3)
Boundary layer	YSU (MPAS-A 7.0 = WRF 4.0.3)
Surface layer	Monin-Obukhov (MPAS-A 7.0 = WRF 4.0.3)
Radiation	RRTMG (MPAS-A 7.0 = WRF 3.8.1)
Cloud fraction	Xu and Randall (MPAS-A 7.0 = WRF 3.8.1)
Gravity wave drag	YSU (MPAS-A 7.0 = WRF 4.0.3)

(c) Further enlargement, illustrating **physics** such as turbulence, radiation, and precipitation. Turbulence is causing a net heat flux into the left side of the grid cell in this example, even though the turbulence has no net wind (i.e., the wind-gust arrows moving air into the grid cell are balanced by gusts moving the same amount of air out of the grid cell). Two of the many radiation bands are shown, where infrared (IR) wavelengths in the 2.0 to 2.5 μm “window” band shine through the grid cell, while wavelengths in the 2.5 to 2.7 μm band are absorbed by water vapor and carbon dioxide (see the Satellites & Radar chapter), causing warming in the grid cell. Some liquid water is falling into the top of the grid cell from the cell above, but even more is falling out the bottom into the grid cell below, suggesting a removal of water and net latent heating due to condensation.

We will revisit physics parameterization later.

Introducing the MPAS-A model

← → ↻ https://mpas-dev.github.io



MPAS Atmosphere

MPAS Home

Overview

- [MPAS-Atmosphere](#)
- [MPAS-Albany Land Ice](#)
- [MPAS-Ocean](#)
- [MPAS-Seaice](#)
- [Data Assimilation](#)
- [Publications](#)
- [Presentations](#)

Download

- [MPAS-Atmosphere download](#)
- [MPAS-Albany Land Ice download](#)
- [MPAS-Ocean download](#)
- [MPAS-Seaice download](#)

Resources

- [License Information](#)
- [Wiki](#)
- [Bug Tracker](#)
- [Mailing Lists](#)
- [MPAS Developers Guide](#)
- [MPAS Mesh Specification Document](#)

Overview

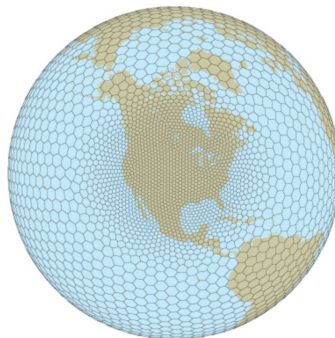
The atmospheric component of MPAS, as with all MPAS components, uses an unstructured centroidal Voronoi mesh (grid, or tessellation) and C-grid staggering of the state variables as the basis for the horizontal discretization in the fluid-flow solver. The unstructured variable resolution meshes can be generated having smoothly-varying mesh transitions (see the figure to the right); we believe that this capability will ameliorate many issues associated with the traditional mesh refinement strategy of one-way and two-way grid nesting where the transitions are abrupt. Using the flexibility of the MPAS meshes, we are working towards applications in high-resolution numerical weather prediction (NWP) and regional climate, in addition to global uniform-resolution NWP and climate applications.

The MPAS atmosphere consists of an atmospheric fluid-flow solver (the *dynamical core*) and a subset of the [Advanced Research WRF](#) (ARW) model atmospheric physics. Work is underway to port the MPAS atmospheric dynamical core to the Community Atmosphere Model (CAM) in the [Community Earth Systems Model](#) (CESM), which will provide coupling between MPAS Ocean and MPAS Atmosphere and coupling to the CAM physics and other components of the CESM system. Work is also progressing on porting the National Centers for Environmental Prediction (NCEP) Global Forecast System (GFS) atmospheric physics to MPAS.

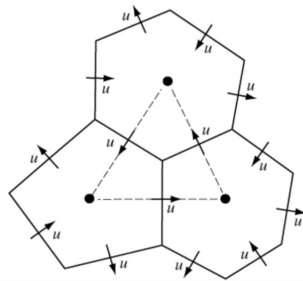
Dynamical Core

The MPAS atmospheric dynamical core solves the fully compressible nonhydrostatic equations of motion. The horizontal Voronoi mesh, depicted to the right, uses a C-grid staggering of the state variables; the horizontal velocity u is defined as the normal velocity on Voronoi cell faces while the other state variables are defined at the cell centers. The dual of the Voronoi mesh is the triangular mesh shown in dashed lines in the figure. The variable resolution meshes are predominantly comprised of hexagons, but pentagons and septagons are occasionally present. The primary advances associated with the C-grid-staggered Voronoi mesh can be found in [Thuburn et al JCP \(2009\)](#), and [Ringler et al JCP \(2010\)](#).

A description of the compressible nonhydrostatic atmospheric solver can be found in [Skamarock et al MWR \(2012\)](#). The fully compressible nonhydrostatic equations are cast in terms of a geometric-height vertical coordinate, and the solver makes use of a split-explicit time integration scheme that is described in [Klemp et al MWR \(2007\)](#). The time-integration scheme employs a 3rd-order Runge-Kutta method, and large time step, for

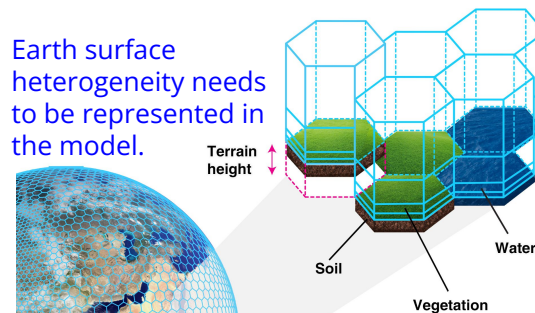


A variable resolution MPAS Voronoi mesh

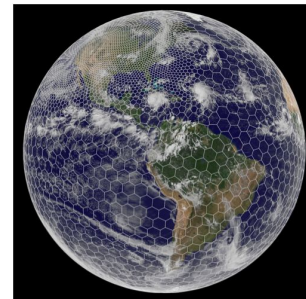


Vertical layers

Earth surface heterogeneity needs to be represented in the model.



Variable-resolution

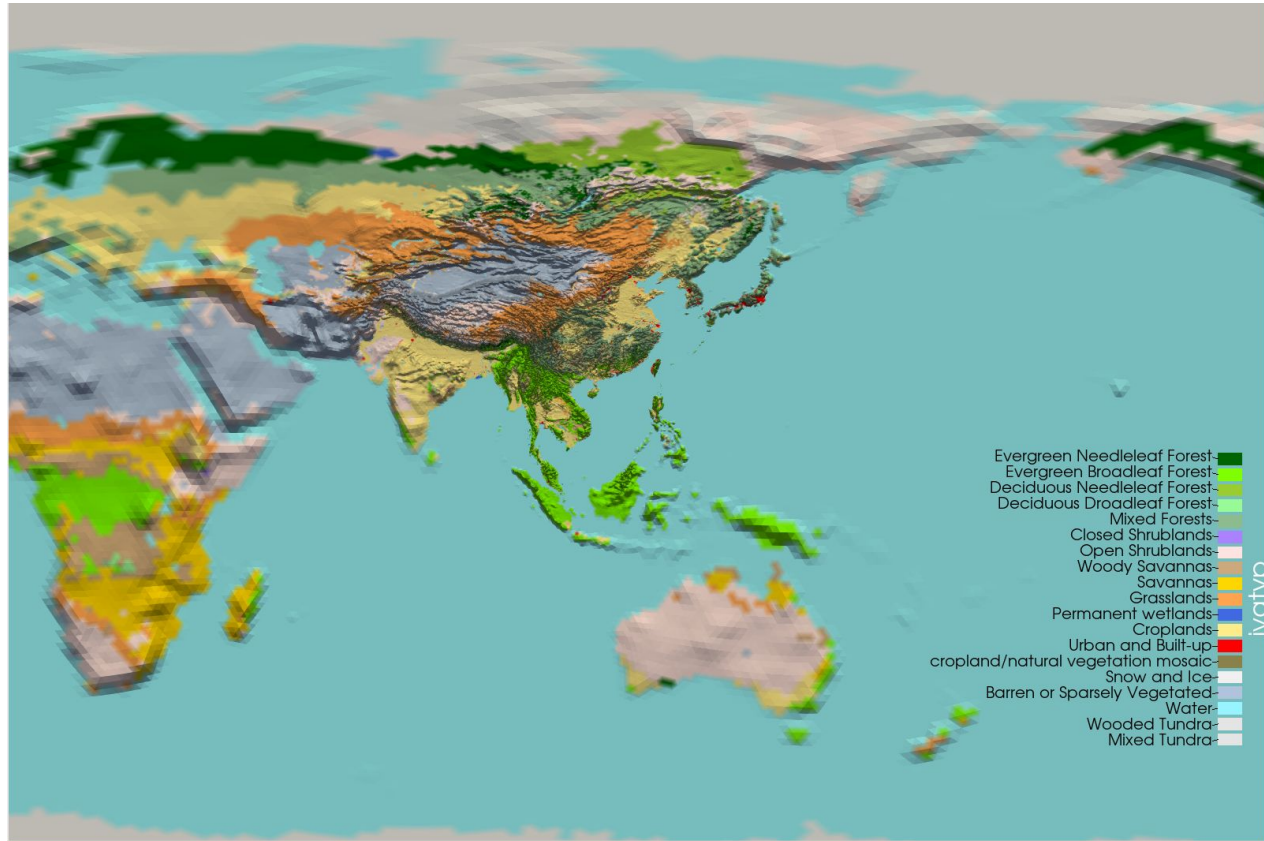


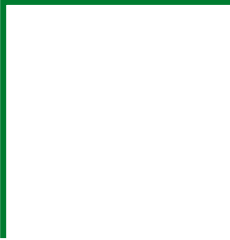
NCAR MPAS-Atmosphere model
<http://mpas-dev.github.io/>

High resolution for cities / provinces


E.g. What fine geographical features needs to be represented for 5-day ahead prediction for Hong Kong?

- Terrain in south China?
- Land-use in south China?
- Terrain of Indochinese Peninsula?
- Terrain of Taiwan / N. Philippines (Tropical cyclone passage)?
- Farther away less important?





Benefit of using
variable-resolution
unstructured grid for
atmospheric simulation



Regional vs global model

Regional model:
The grid covers a region only

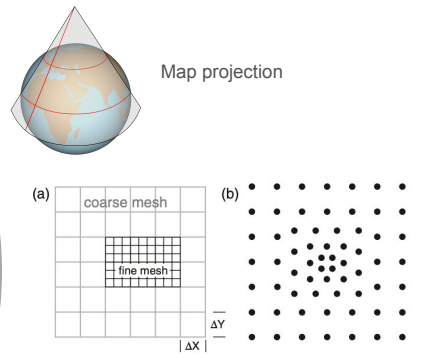
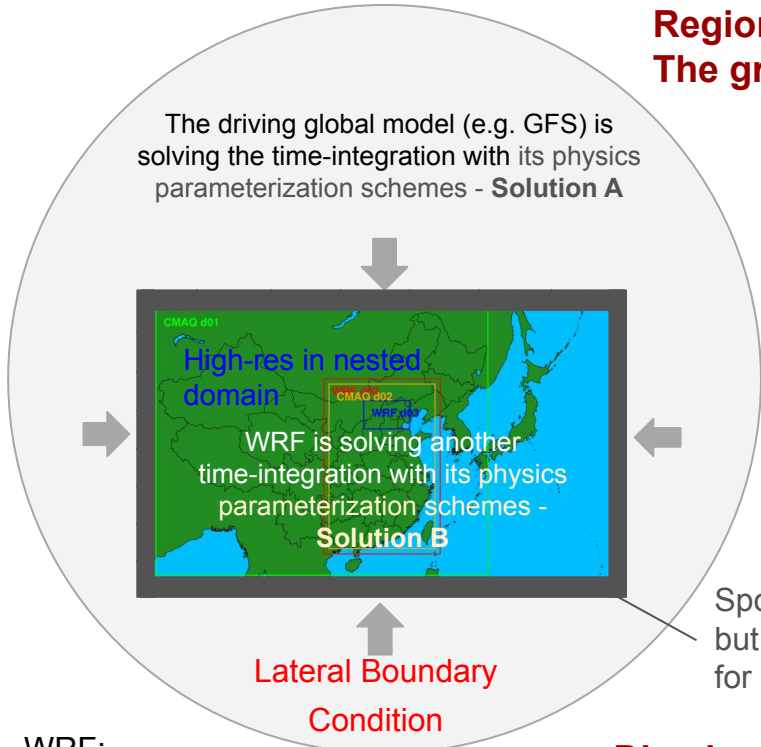


Figure 20.4
Horizontally nested grids. (a) Discrete meshes (shown as grid cells). (b) Variable mesh (shown as grid points).



Sponge layers help, but simulation valid for ~72 hours only

Disadvantage:
Limited forecast hours

MPAS-A:

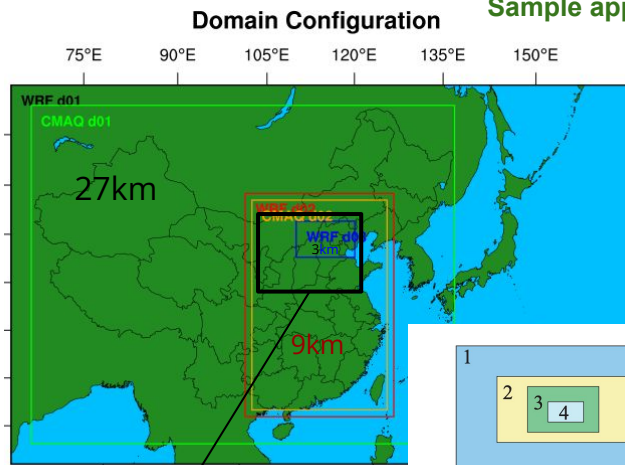
- A single **global model** with refinement regions
- A set of **consistent, scale-aware (improving)** physics parameterization schemes.
- **Longer period with valid forecast**

WRF:

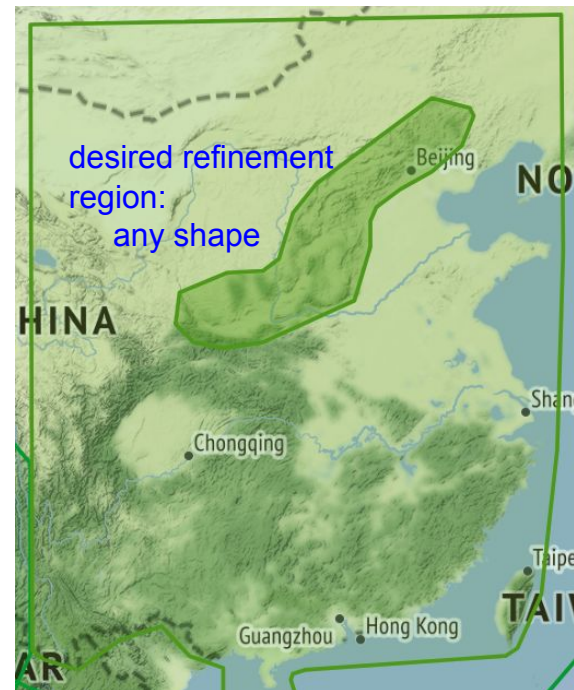
- A popular regional model

Inflexibility of domain nesting

Sample application: NWP driving Air Quality model CMAQ



E.g. to resolve mountains (not aligned with N-S, W-E direction) to predict a better timing of cold front arrival



Constrained refinement domain:
 - rectangular,
 - aligned with N-S, W-E direction
 Costly for enlarged refinement.

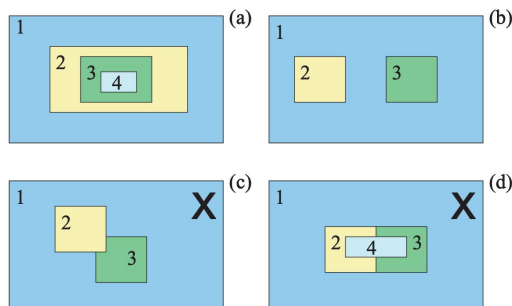


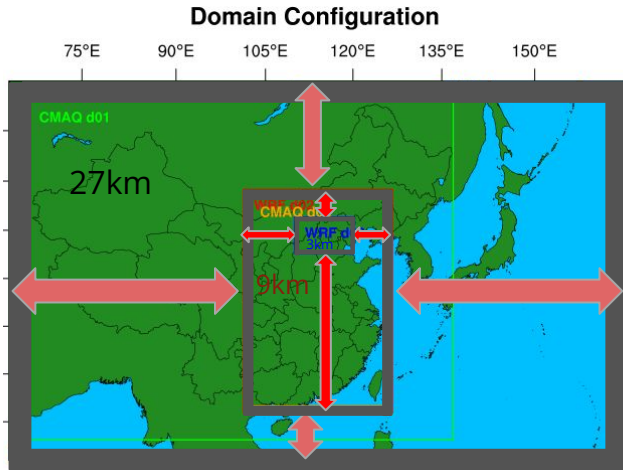
Figure 7.2: Various nest configurations for multiple grids. (a) Telescoping nests. (b) Nests at the same level with respect to a parent grid. (c) Overlapping grids: not allowed with feedback activated. (d) Inner-most grid has more than one parent grid: not allowed

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Liu, Z., Berner, J., ... Huang, X. -yu. (2021). A Description of the Advanced Research WRF Model Version 4.3 (No. NCAR/TN-556+STR). doi:10.5065/1dfh-6p97

WRF Nesting

MPAS variable-resolution
 unstructured mesh -> later slide

Domain nesting vs resolution transition



WRF nesting's Lateral Boundary

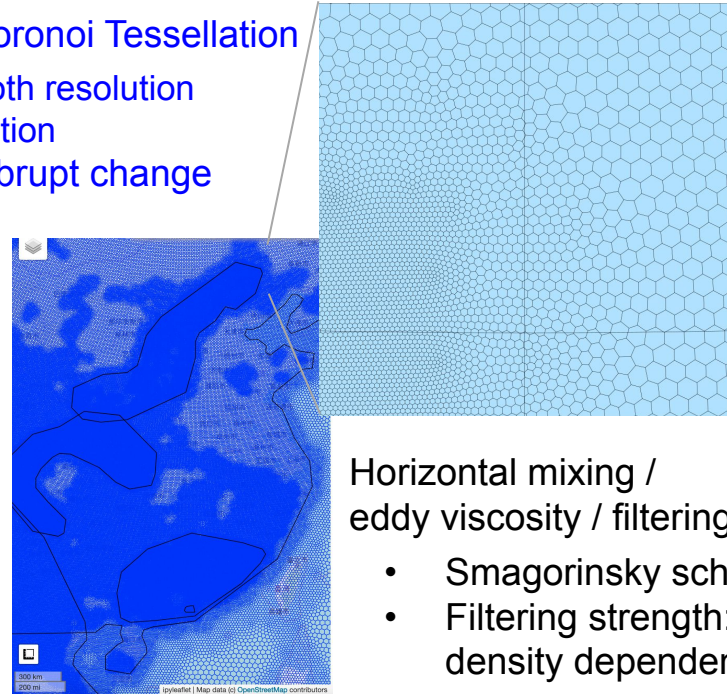
- Abrupt change of resolution
- Reflection of wave handed by sponge layers
- Best practice: far away from region of interest.

Sample application:

Nesting for densely populated area in China

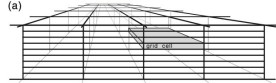
MPAS-A Voronoi Tessellation

- Smooth resolution transition
- No abrupt change



Hello Finite Volume Method

My favorite Finite Difference Method?



- Unstructured grid
 - No more Cartesian (x,y,z)
 - How do I do finite difference?
- Coordinates for sphere
 - Zonal (W-E direction)
 - Meridional (S-N direction)

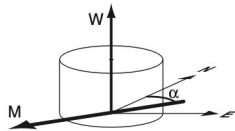


Figure 1.3
Notation used in cylindrical coordinates for velocity.

https://www2.mmm.ucar.edu/projects/mpas/tutorial/Boulder2019/slides/07.MPAS_solver.pdf
Slide 2

MPAS
Model for Prediction Across Scales

Nonhydrostatic formulation

Equations

- Prognostic equations for coupled variables.
- **Generalized height coordinate.**
- Horizontally vector invariant eqn set.
- Continuity equation for dry air mass.
- Thermodynamic equation for coupled potential temperature.

Time integration scheme

As in Advanced Research WRF - Split-explicit Runge-Kutta (3rd order), with additional splitting options.

MPAS Nonhydrostatic Atmospheric Solver

Variables:

$$(U, V, \Omega, \Theta, Q_j) = \bar{p}_d \cdot (u, v, \eta, \theta, q_j)$$

Vertical coordinate:

$$z = \zeta + A(\zeta) h_s(x, y, \zeta)$$

Prognostic equations:

$$\frac{\partial \mathbf{V}_H}{\partial t} = - \frac{\rho_d}{\rho_m} \left[\nabla_\zeta \left(\frac{p}{\zeta_z} \right) - \frac{\partial \mathbf{z}_{HP}}{\partial \zeta} \right] - \eta \mathbf{k} \times \mathbf{V}_H - \mathbf{v}_H \nabla_\zeta \cdot \mathbf{V} - \frac{\partial \Omega \mathbf{V}_H}{\partial \zeta} - \rho_d \nabla_\zeta K + \mathbf{F}_{V_H}$$

$$\frac{\partial W}{\partial t} = - \frac{\rho_d}{\rho_m} \left[\frac{\partial p}{\partial \zeta} + g \bar{\rho}_m \right] - (\nabla \cdot \mathbf{v} W)_\zeta + F_W$$

$$\frac{\partial \Theta_m}{\partial t} = - (\nabla \cdot \mathbf{V} \theta_m)_\zeta + F_{\Theta_m}$$

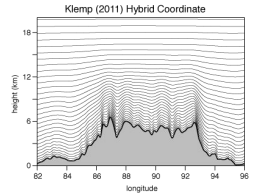
$$\frac{\partial \rho_d}{\partial t} = - (\nabla \cdot \mathbf{V})_\zeta$$

$$\frac{\partial Q_j}{\partial t} = - (\nabla \cdot \mathbf{V} q_j)_\zeta + F_{Q_j}$$

Diagnostics and definitions:

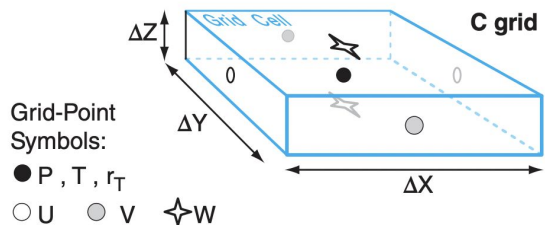
$$\theta_m = \theta [1 + (R_v/R_d) q_v] \quad p = p_0 \left(\frac{R_d \zeta_c \Theta_m}{p_0} \right)^\gamma$$

$$\frac{\rho_m}{\rho_d} = 1 + q_v + q_c + q_r + \dots$$

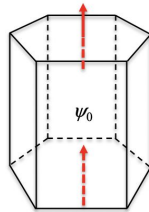
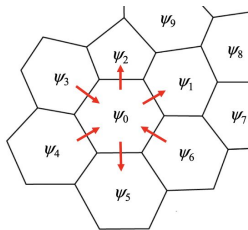


Scalar at cell center, wind on cell face!

On rectangular grid, called "C-grid staggering"



On unstructured grid, this is natural!



MPAS
 Model for Prediction Across Scales

MPAS Nonhydrostatic
 Atmospheric Solver

Prognostic equations:

$$\frac{\partial \mathbf{V}_H}{\partial t} = -\frac{\rho_d}{\rho_m} \left[\nabla_\zeta \left(\frac{p}{\zeta_z} \right) - \frac{\partial \mathbf{z}_H p}{\partial \zeta} \right] - \eta \mathbf{k} \times \mathbf{V}_H$$

$$- \mathbf{v}_H \nabla_\zeta \cdot \mathbf{V} - \frac{\partial \Omega \mathbf{v}_H}{\partial \zeta} - \rho_d \nabla_\zeta K + \mathbf{F}_{V_H}$$

$$\frac{\partial W}{\partial t} = -\frac{\rho_d}{\rho_m} \left[\frac{\partial p}{\partial \zeta} + g \tilde{\rho}_m \right] - (\nabla \cdot \mathbf{v} W)_c + F_W$$

$$\frac{\partial \Theta_m}{\partial t} = -(\nabla \cdot \mathbf{V} \theta_m)_c + F_{\Theta_m}$$

$$\frac{\partial \rho_d}{\partial t} = -(\nabla \cdot \mathbf{V})_c$$

$$\frac{\partial Q_j}{\partial t} = -(\nabla \cdot \mathbf{V} q_j)_c + F_{Q_j}$$

- (1) Gradient operators
- (2) Flux divergence operators
- (3) Nonlinear Coriolis term

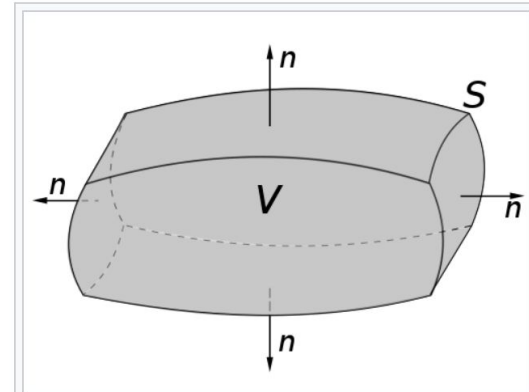
Divergence theorem

Mathematical statement [\[edit \]](#)

Suppose V is a subset of \mathbb{R}^n (in the case of $n = 3$, V represents a volume in [three-dimensional space](#)) which is [compact](#) and has a [piecewise smooth boundary](#) S (also indicated with $\partial V = S$). If \mathbf{F} is a continuously differentiable vector field defined on a [neighborhood](#) of V , then:^{[4][5]}

$$\iiint_V (\nabla \cdot \mathbf{F}) \, dV = \oiint_S (\mathbf{F} \cdot \hat{\mathbf{n}}) \, dS.$$

The left side is a [volume integral](#) over the volume V , the right side is the [surface integral](#) over the boundary of the volume V . The closed manifold ∂V is oriented by outward-pointing [normals](#), and $\hat{\mathbf{n}}$ is the outward pointing unit normal at each point on the boundary ∂V . ($d\mathbf{S}$ may be used as a shorthand for $\mathbf{n}dS$.) In terms of the intuitive description above, the left-hand side of the equation represents the total of the sources in the volume V , and the right-hand side represents the total flow across the boundary S .



A region V bounded by the surface $S = \partial V$ with the surface normal n

https://en.wikipedia.org/wiki/Divergence_theorem

Mass continuity

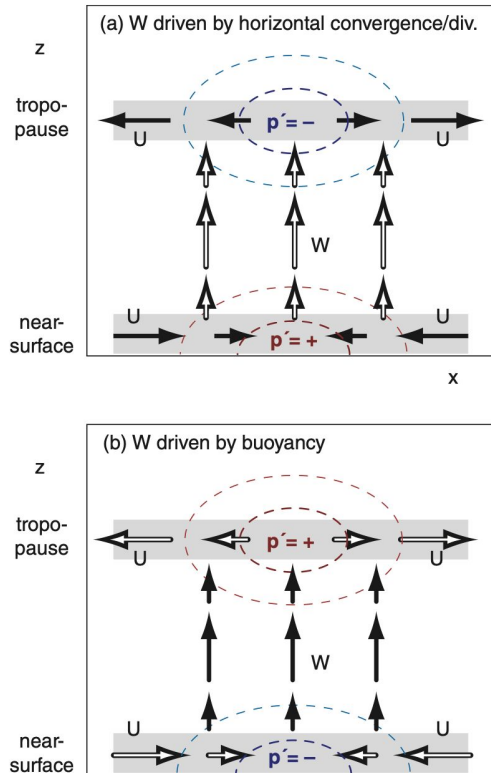


Figure 11.16

<https://cpas.earth/>

11.4.1.1. Horizontal Convergence/Divergence

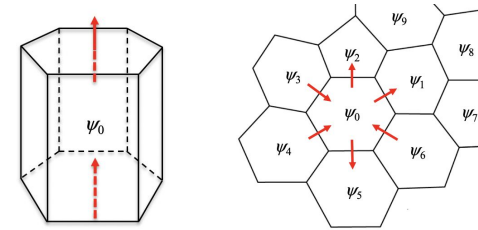
If external forcings cause air near the ground to converge horizontally, then air molecules accumulate. As density ρ increases according to eq. (10.60), the ideal gas law tells us that p' will also become positive (Fig. 11.16a). Non-zero p' implies that a pressure gradient exists, which will drive winds.

Positive p' does two things: it (1) decelerates the air that was converging horizontally, and (2) accelerates air vertically in the column. Thus, the pressure perturbation causes **mass continuity** (horizontal inflow near the ground balances vertical outflow).

11.4.1.2. Buoyant Forcings

For a different scenario, suppose air in a column is positively buoyant, such as in a thunderstorm where water-vapor condensation releases lots of latent heat. This vertical buoyant force creates upward motion (i.e., warm air rises, as in Fig. 11.16b).

As air in the thunderstorm column moves away from the ground, it removes air molecules and lowers the density and the pressure; hence, p' is negative near the ground. This suction under the updraft causes air near the ground to horizontally converge, thereby conserving mass.



Stull 2017 Practical Meteorology

https://www.eoas.ubc.ca/books/Practical_Meteorology/

Chapter 11 General Circulation

For conservative physical quantities


Partial Differential Equations:

- Derivative w.r.t time \sim divergence term

Finite Volume Method:

- Flux: How much quantity goes out (and in) to the 3D grid cell
- \Rightarrow net flux
- \Rightarrow rate of increase of the quantity inside the 3D grid cell

https://www2.mmm.ucar.edu/projects/mpas/tutorial/Boulder2019/slides/07.MPAS_solver.pdf
Slide 18



MPAS
Model for Prediction Across Scales

Operators on the Voronoi Mesh

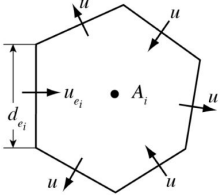
Flux divergence and transport

Transport equation, conservative form:
$$\frac{\partial(\rho\psi)}{\partial t} = -\nabla \cdot \mathbf{V}(\rho\psi)$$

Finite-Volume formulation, Integrate over cell:
$$\int_D \left[\frac{\partial}{\partial t}(\rho\psi) = -\nabla \cdot \mathbf{V}(\rho\psi) \right] dV$$

Apply divergence theorem:
$$\frac{\partial(\overline{\rho\psi})}{\partial t} = -\frac{1}{V} \int_{\Sigma} (\rho\psi) \mathbf{V} \cdot \mathbf{n} d\sigma$$

Discretize in time and space:
$$(\rho\psi)_i^{t+\Delta t} = (\rho\psi)_i^t - \Delta t \frac{1}{A_i} \sum_{n_{e_i}} d_{e_i} \overline{(\rho\mathbf{V} \cdot \mathbf{n}_{e_i})\psi}$$



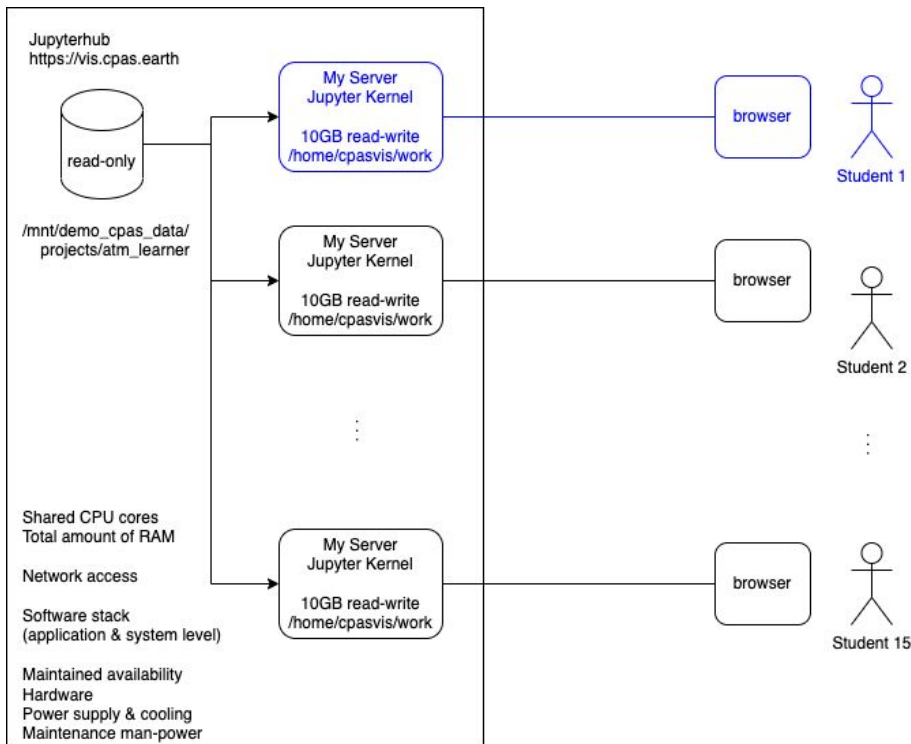
Velocity divergence operator is 2nd-order accurate for edge-centered velocities.



Online Programming Framework



Your learning infrastructure - CPAS Visualization system



The screenshot shows a Jupyter Notebook interface for **Tutorial1** (unsaved changes). The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Navigate, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar showing **Memory: 149 MB / 1 GB** and **CPU: 0% / 1 vCPU**.

The **Contents** panel on the left lists the following sections:

- ESSC4602 - Selected Topics in Earth System Science
- Tutorial 1 - Basic data manipulation and 2D visualization
- 1. Basics of Numpy
 - 1.1. Creating numpy arrays
 - 1.2. Indexing and slicing
 - 1.3. Operations on numpy array
 - 1.4. Broadcasting
- 2. Data manipulation with xarray
 - 2.1. Data structure
 - 2.2. Data manipulation
 - 2.2.1. Selecting data with `.isel` and `.isel`
 - 2.2.2. Functions for higher level computation
- 3. Creating 2D static plots with matplotlib
 - 3.1. Coding styles
 - 3.2. Figure and axes
 - 3.3. Subplots and plot types
 - 3.4. Features and Customization
 - 3.5. 2D spatial plots
 - 3.5.1. Visualization of scalars - filled contour
 - 3.5.2. Visualization of scalars - contour plot
 - 3.5.3. Visualization of vectors - vector field
 - 3.6. Overlay
 - 3.7. Saving plots
- 4. Creating 2D interactive plots with hvplot
 - 4.1. Line plots
 - 4.2. Filled contour plot
 - 4.3. Contour plot
 - 4.4. Vector field plot
 - 4.5. Overlay

The main content area displays the title **ESSC4602 - Selected Topics in Earth System Science** and **Tutorial 1 - Basic data manipulation and 2D visualization**. The **Content:** section lists:

- Basics of numpy
- Data manipulation with xarray
- Create 2D static plots and interactive plots with matplotlib and hvplot

Below the content, a code cell is shown with the following code:

```
In [ ]: # import necessary libraries
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt

import hvplot.xarray
```

The code was executed in 5.46s, finished 15:54:49 2022-03-30.

The **1. Basics of Numpy** section explains that **Numpy** is a powerful package that provides a multidimensional array object (the `ndarray` object), and supports various operations on arrays. It is important to learn **numpy** as to proceed further in the Python world.

A link is provided for a quickstart and learn important functions for Numpy: <https://numpy.org/doc/stable/user/quickstart.html>

Plain Python?

Imperative programming style

```
my_list = [1, 2, 3, 4, 5]
```

```
sum = 0
```

```
for x in my_list:
```

```
    sum += x
```

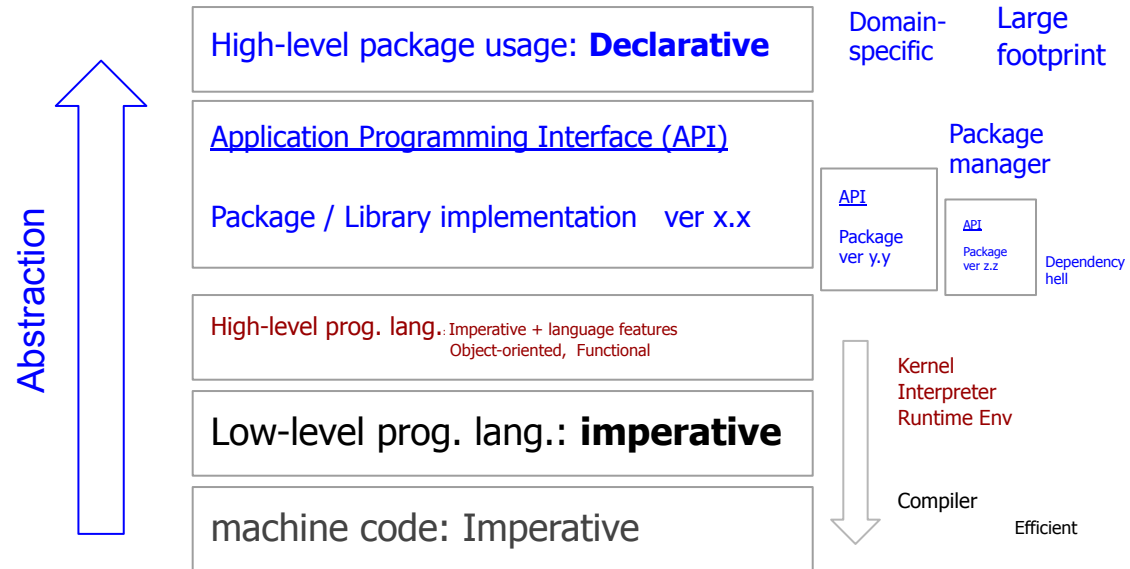
```
print(sum)
```

- Code *how* to achieve the goal
- Number of lines of code: high
- Tedious

Functional programming style

```
import functools
```

```
sum = functools.reduce(  
    (lambda a, b: a+b), my_list)  
print(sum)
```



Pyviz - high level data visualization

This course:

Matplotlib

Cartopy

Holoviews

Bokeh

Plotly

...

Giving up

NCL

PyNGL

The screenshot shows the Pyviz website at <https://pyviz.org/overviews/index.html>. The page features the Pyviz logo (a stylized bar chart) and the heading "Overviews". Below the heading, a paragraph states: "The Python visualization landscape can seem daunting at first. These overviews attempt to shine light on common patterns and use cases, comparing or discussing multiple plotting libraries. Note that some of the projects discussed in the overviews are no longer maintained, so be sure to check the list of [dormant projects](#) before choosing that library."

Below the text is a large, colorful hub-and-spoke diagram of the Python visualization ecosystem. The central node is "Matplotlib" (blue). Other prominent nodes include "bokeh" (orange), "d3.js" (pink), "OpenGL" (green), "holoviews" (yellow), and "javascript" (orange). Smaller nodes radiate from these, such as "plotly", "cufflinks", "toyplot", "dashader", "Vaex", "mpld3", "altair", "vega-lite", "vega", "vincent", "d3po", "pythreejs", "vispy", "glumpy", "visvis", "galry", "mayavi", "ipyvolume", "ipyleaflet", "ipython", "lighting", "cufflinks", "toyplot", "dashader", "Vaex", "mpld3", "altair", "vega-lite", "vega", "vincent", "d3po", "pythreejs", "vispy", "glumpy", "visvis", "galry", "mayavi", "ipython", "lighting", "cufflinks", "toyplot", "dashader", "Vaex", "mpld3", "altair", "vega-lite", "vega", "vincent", "d3po", "pythreejs", "vispy", "glumpy", "visvis", "galry", "mayavi".

At the bottom of the diagram are three buttons: "pygal", "chaco", and "PyQTGraph".

Below the diagram is a search result from Google for the query "ncl and pyngl have been put into 'maintenance mode'". The search results show a link to <https://www.ncl.ucar.edu> with the title "Important letter regarding the future of NCL" and a snippet: "NCL's core language and file I/O will be placed into maintenance mode. NCL's graphics will have continued development through PyNGL*." Another result is from <https://mailman.ucar.edu> with the title "[ncl-talk] November update from NCAR's GeoCAT team" and a snippet: "31 May 2021 — ... geosciences analysis tools that were based on the Python Scientific Ecosystem. Furthermore, we announced that NCL would be put into 'maintenance mode', ...". A third result is from <https://github.com> with the title "Python 3.9 Support on Conda · Issue #45 · NCAR/pyngl - GitHub" and a snippet: "3 May 2021 — We have put PyNGL into maintenance mode a long while ago and announced this in several community updates. For example, please see November 2020 ...".

Code *what* you want

https://dev.to/ruizb/declarative-vs-imperative-4a7l

Log in Create account

Making a chocolate cake

Let's take an example from the real world: we would like to make a chocolate cake. How would that look like with these 2 paradigms?

The imperative way

1. First, turn on the oven to preheat it at 180°C.
2. Next, add flour, sugar, cocoa powder, baking soda and salt to a large bowl, then stir the mixture with a paddle.
3. Then, add milk, vegetable oil, eggs and vanilla extract to the mixture, and mix together on medium speed until well combined.
4. Distribute the cake batter evenly in a large cake pan, then bake it for approx. 30 minutes.
5. Remove the pan from the oven with a pot holder, let it cool for 10 minutes.
6. Finally, remove the cake from the pan with the tapping method, and frost it evenly with chocolate frosting.

<https://dev.to/ruizb/declarative-vs-imperative-4a7l>

Benoit Ruiz
Follow

The declarative way

1. You have to preheat the oven to 180 °C.
2. You have to mix dry ingredients in a bowl.
3. Once dry ingredients are mixed, you have to add wet ingredients to the mixture, and mix together to form the cake batter.
4. Once the oven and batter are ready, you have to put the batter in a pan, then bake it for 30 minutes.
5. Once baked, you have to remove the pan from the oven and let it cool for 10 minutes.
6. Finally, you have to remove the cake from the pan, and frost it.
7. Ready? Go!



Typical code you'll write ...

1

```
# import necessary libraries
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt

import hvplot.xarray
```

2

```
# Open .nc file with open_dataset as xarray.Dataset
ds = xr.open_dataset("data/ERA5_sample.nc")
ds # show the data
```

executed in 157ms, finished 08:48:27 2022-04-13

xarray.Dataset

Dimensions: (longitude: 1440, latitude: 721, time: 12)

Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359.2 359.5
latitude	(latitude)	float32	90.0 89.75 89.5 ... -89.75 -89.5
time	(time)	datetime64[ns]	2021-01-01 ... 2021-12-31

Data variables:

u10	(time, latitude, longitude)	float32	...
v10	(time, latitude, longitude)	float32	...
t2m	(time, latitude, longitude)	float32	...
msl	(time, latitude, longitude)	float32	...

Attributes:

Conventions: CF-1.6

history: 2022-03-01 04:10:10 GMT by orih to netcdf-2.24.2 /opt/ecmw/m

3

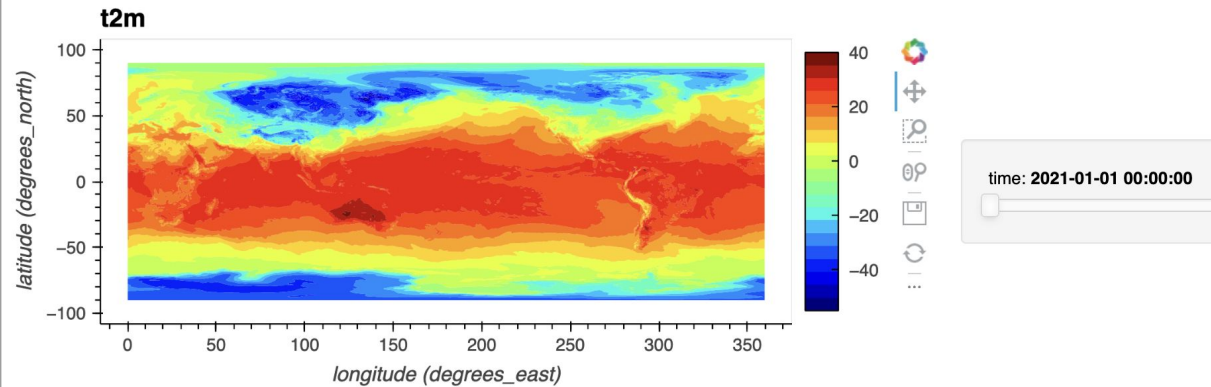
```
# reassign the variables from 'data' dataset
display(ds)

# Store the available data arrays from the data to the variables
# this time we retain all the dimension of the data arrays
t2m = ds.t2m-273.15 # convert the unit from K to deg C
msl = ds.msl/100 # convert the unit from Pa to hPa
u10 = ds.u10
v10 = ds.v10
```

4

```
# Customize the contour plot by adding different parameters
t2m.hvplot.contourf(x='longitude', y='latitude',
                  cmap='jet',
                  levels=20, # set the number of contour levels
                  colorbar=True,
                  title="t2m", # set the title of the plot
                  fontsize=1.2 # scale up the font size of all texts by 1.2
                  )
```

executed in 1.92s, finished 08:49:42 2022-04-13



Memory: 143.6 MB / 1 GB CPU: 0% / 1 vCPU

CPU: 0% / 1 vCPU

Memory: 928.3 MB / 1 GB

▲ Reduce memory usage by terminating other notebooks or restart kernel, or buy more resources.

Memory: 469.6 MB / 1 GB

CPU: 118.1% / 1 vCPU

Typical error you'll encounter ...

... how students become reliable professionals

Memory: 143.6 MB / 1 GB CPU: 0% / 1 vCPU

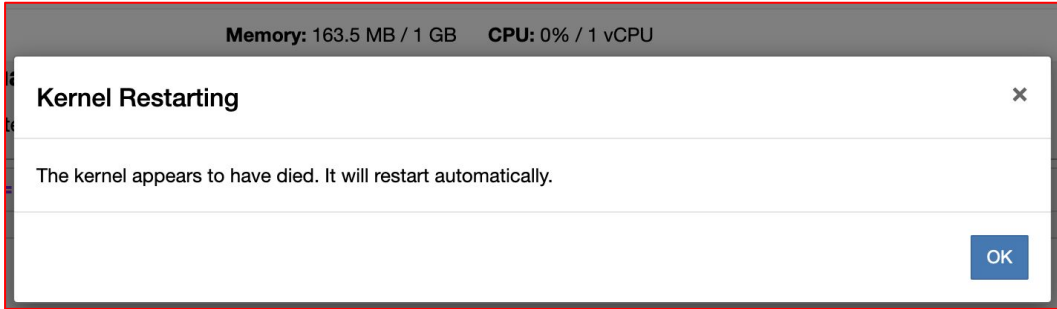
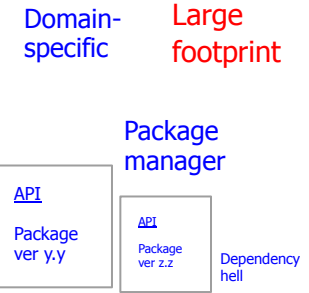
Memory: 469.6 MB / 1 GB CPU: 118.1% / 1 vCPU

Memory: 928.3 MB / 1 GB
⚠ Reduce memory usage by terminating other notebooks or restart kernel, or buy more resources.

High-level package usage: **Declarative**

Application Programming Interface (API)
Package / Library implementation ver x.x

High-level prog. lang.: Imperative + language features
Object-oriented, Functional



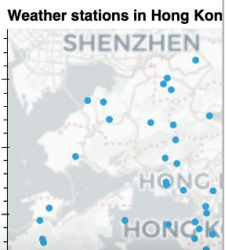
Expect to encounter error ...

... Keep calm and read the error message ... understand ... decide ...

```
In [34]: # create an interactive map
geo_hk_station.hvplot(geo=True,tiles='CartoLight', # add map tiles
                    hover_cols='name_tc', # show the names of stations when hovering over the point
                    size=40, # change the size of points
                    title="Weather stations in Hong Kong"
                    )
```

```
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:825: ShapelyDeprecationWarning: __len__ for multi-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` pr
operty instead to get the number of parts of a multi-part geometry.
  if len(multi_line_string) > 1:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:877: ShapelyDeprecationWarning: Iteration over multi-part geometries is deprecated and will be removed in Shapely 2.0. Use the `geoms` property to acc
ess the constituent parts of a multi-part geometry.
  for line in multi_line_string:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:944: ShapelyDeprecationWarning: __len__ for multi-part geometries is deprecated and will be removed in Shapely 2.0. Check the length of the `geoms` pr
operty instead to get the number of parts of a multi-part geometry.
  if len(p_mline) > 0:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:825: ShapelyDeprecationWarning: __len__ for multi-p
operty instead to get the number of parts of a multi-part geometry.
  if len(multi_line_string) > 1:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:877: ShapelyDeprecationWarning: Iteration over mult
ess the constituent parts of a multi-part geometry.
  for line in multi_line_string:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:944: ShapelyDeprecationWarning: __len__ for multi-p
operty instead to get the number of parts of a multi-part geometry.
  if len(p_mline) > 0:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:825: ShapelyDeprecationWarning: __len__ for multi-p
operty instead to get the number of parts of a multi-part geometry.
  if len(multi_line_string) > 1:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:877: ShapelyDeprecationWarning: Iteration over mult
ess the constituent parts of a multi-part geometry.
  for line in multi_line_string:
/opt/conda/lib/python3.8/site-packages/cartopy/crs.py:944: ShapelyDeprecationWarning: __len__ for multi-p
operty instead to get the number of parts of a multi-part geometry.
  if len(p_mline) > 0:
```

Out [34]:



```
from shapely.errors import ShapelyDeprecationWarning
import warnings
warnings.filterwarnings(action="ignore", category=ShapelyDeprecationWarning)
```

High-level package usage: **Declarative**

Domain-specific Large footprint

Application Programming Interface (API)

Package manager

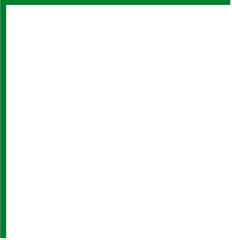
Package / Library implementation ver x.x

API
Package
ver y.y

API
Package
ver z.z

https://en.wikipedia.org/wiki/Dependency_hell

Dependency hell



Lab 1

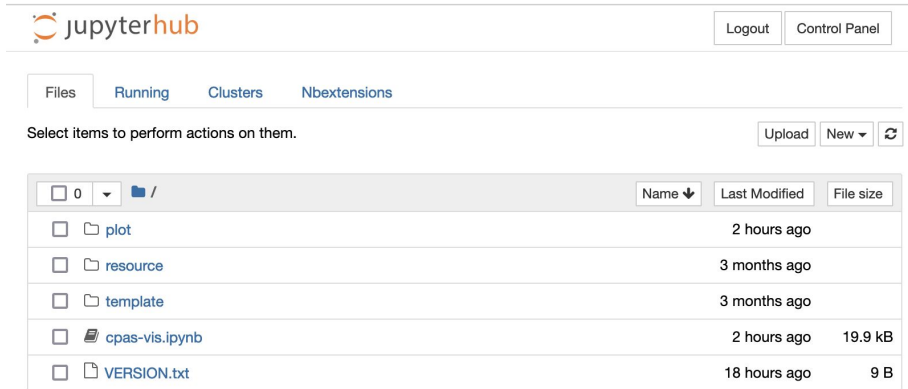
Hello My Jupyter Server

5 minutes



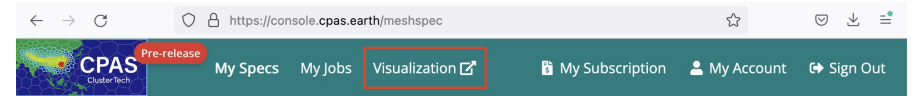
Let there be My Server

1. Click "Visualization"
2. (For new user / system update)
See the Update message.
Click "Continue using ..."
3. Click "Start My Server".
4. See the Tree view of your files.

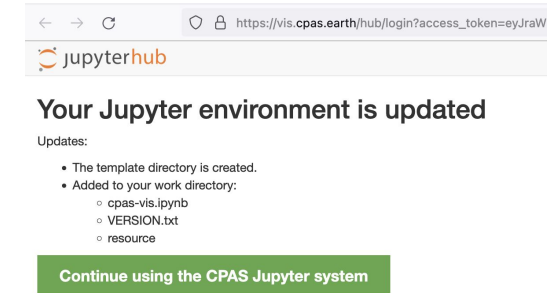


The screenshot shows the JupyterHub interface with the 'Files' tab selected. The top navigation bar includes 'Logout' and 'Control Panel'. Below the navigation bar, there are tabs for 'Files', 'Running', 'Clusters', and 'Nbextensions'. A message says 'Select items to perform actions on them.' with 'Upload', 'New', and a refresh icon. The file tree shows a directory structure with the following items:

	Name	Last Modified	File size
<input type="checkbox"/>	plot	2 hours ago	
<input type="checkbox"/>	resource	3 months ago	
<input type="checkbox"/>	template	3 months ago	
<input type="checkbox"/>	cpas-vis.ipynb	2 hours ago	19.9 kB
<input type="checkbox"/>	VERSION.txt	18 hours ago	9 B



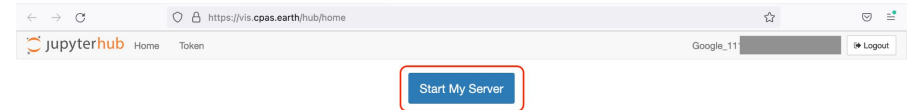
The screenshot shows the CPAS console dashboard. The URL is <https://console.cpas.earth/meshspec>. The navigation bar includes 'My Specs', 'My Jobs', 'Visualization' (highlighted with a red box), 'My Subscription', 'My Account', and 'Sign Out'.



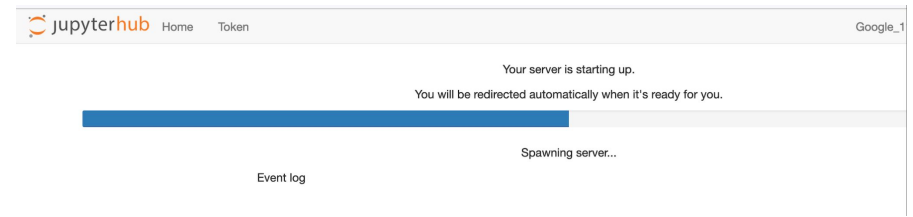
The screenshot shows the JupyterHub update message. The URL is https://vis.cpas.earth/hub/login?access_token=eyJraWw. The message says 'Your Jupyter environment is updated' and lists the updates:

- The template directory is created.
- Added to your work directory:
 - cpas-vis.ipynb
 - VERSION.txt
 - resource

A green button labeled 'Continue using the CPAS Jupyter system' is visible at the bottom.



The screenshot shows the JupyterHub home page. The URL is <https://vis.cpas.earth/hub/home>. The navigation bar includes 'Home', 'Token', and 'Logout'. A blue button labeled 'Start My Server' is highlighted with a red box.



The screenshot shows the JupyterHub server startup screen. The message says 'Your server is starting up. You will be redirected automatically when it's ready for you.' Below the message is a blue progress bar and the text 'Spawning server...'. An 'Event log' section is visible at the bottom.

Hello Terminal

1. Click "New", "Terminal"
2. See the **Linux** terminal.
3. Type Linux command "ls"
4.

```
$ cd work
```

```
$ ls -l
```

```
$ pwd
```

```
$ whoami
```



5. Note: Only the directory (and its sub-directory) </home/cpasvis/work> is permanent storage (during subscription)

You should not write files to other location.
They may be lost (e.g. "Stop Server")

If the Terminal becomes not responsive, close the tab, start a new Terminal

The screenshot shows the JupyterHub interface. At the top right, there are "Logout" and "Control Panel" buttons. Below the header, there are tabs for "Files", "Running", "Clusters", and "Nbextensions". A message says "Select items to perform actions on them." Below this is a file browser showing a directory structure with folders like "plot", "resource", and "template", and files like "cpas-vis.ipynb" and "VERSION.txt". A "New" button is highlighted with a red box, and a dropdown menu is open, showing options like "Notebook: Python 3", "Text File", "Folder", and "Terminal" (which is also highlighted with a red box). Below the file browser, there is another JupyterHub header with "Logout" and "Control Panel" buttons, and a terminal window showing the following commands and output:

```
(base) cpasvis@14bc0546e506:~$ ls
work
(base) cpasvis@14bc0546e506:~$ cd work
(base) cpasvis@14bc0546e506:~/work$ ls -l
total 24
-rw-r----- 1 cpasvis cpasvis 19913 Apr 13 02:23 cpas-vis.ipynb
drwxr-xr-x  2 cpasvis cpasvis   6 Apr 13 02:18 plot
drwxr-x---  2 cpasvis cpasvis  27 Jan 10 04:51 resource
dr-xr-x---  3 cpasvis cpasvis  63 Jan 10 04:51 template
-r--r----- 1 cpasvis cpasvis   9 Apr 12 10:13 VERSION.txt
(base) cpasvis@14bc0546e506:~/work$ pwd
/home/cpasvis/work
(base) cpasvis@14bc0546e506:~/work$ whoami
cpasvis
(base) cpasvis@14bc0546e506:~/work$
```

Soft-link data; copy your own .ipynb

Course materials are in

`/mnt/demo_cpas_data/projects/atm_learner`

Use soft-link to memorize the long path

```
$ ln -s /mnt/demo_cpas_data/projects/atm_learner ./
```

```
$ ls -l atm_learner
```

Your storage space is precious;

Avoid making copies of common data;
but you need your own writable .ipynb

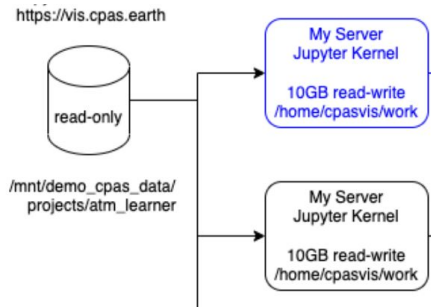
```
$ mkdir my_atm_learner
```

```
$ cd my[Tab autocompletion]_atm_learner/
```

```
$ ln -s ../atm[Tab]/data_for_tutorials/ ./
```

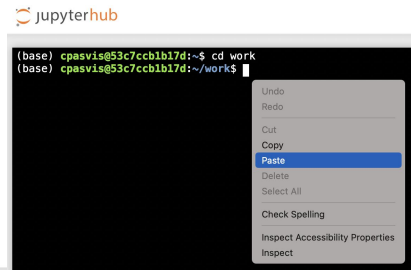
```
$ cp -r ../atm[Tab]/<which lesson>/ ./
```

```
$ ls -l
```



Reminder:

Send Slide URL in Zoom chat
Copy command from slide
Paste to the Terminal

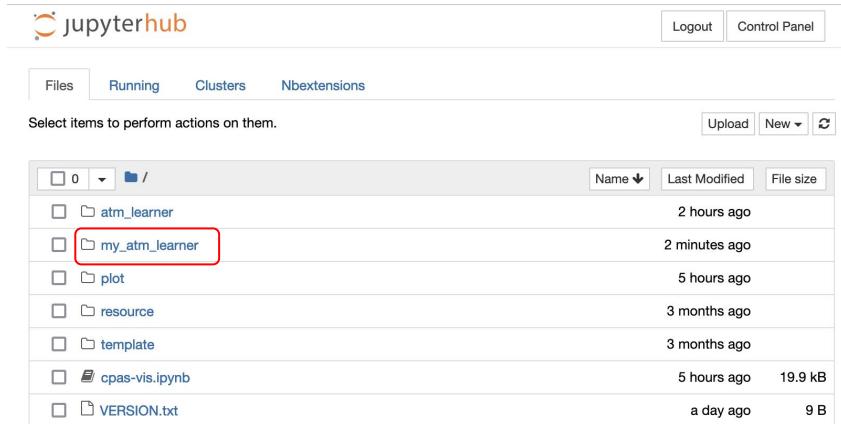


jupyterhub

```
(base) cpasvis@66b8f85f2441:~$ cd work
(base) cpasvis@66b8f85f2441:~/work$ ls
cpas-vis.ipynb  resource  template  VERSION.txt
(base) cpasvis@66b8f85f2441:~/work$ ln -s /mnt/demo_cpas_data/projects/atm_learner ./
(base) cpasvis@66b8f85f2441:~/work$ ls -l atm_learner
lrwxrwxrwx 1 cpasvis cpasvis 40 Apr 12 11:25 atm_learner -> /mnt/demo_cpas_data/projects/atm_learner
(base) cpasvis@66b8f85f2441:~/work$
```

```
(base) cpasvis@d524a698811f:~/work$ mkdir my_atm_learner
(base) cpasvis@d524a698811f:~/work$ cd my_atm_learner/
(base) cpasvis@d524a698811f:~/work/my_atm_learner$ ln -s ../atm_learner/data_for_tutorials/ ./
(base) cpasvis@d524a698811f:~/work/my_atm_learner$ cp -r ../atm_learner/tutorial1/ ./
(base) cpasvis@d524a698811f:~/work/my_atm_learner$ ls -l
total 0
lrwxrwxrwx 1 cpasvis cpasvis 34 May  6 06:03 data_for_tutorials -> ../atm_learner/data_for_tutorials/
drwxr-xr-x 3 cpasvis cpasvis 55 May  6 06:08 tutorial1
(base) cpasvis@d524a698811f:~/work/my_atm_learner$
```

Hello Tutor's Notebook

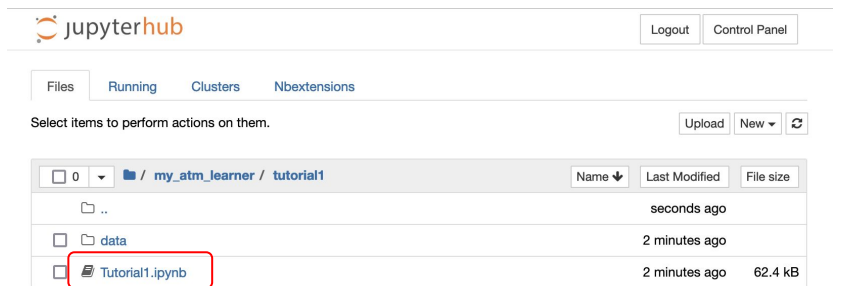


jupyterhub Logout Control Panel

Files Running Clusters Nbextensions

Select items to perform actions on them. Upload New ↻

	Name	Last Modified	File size
<input type="checkbox"/>	/		
<input type="checkbox"/>	atm_learner	2 hours ago	
<input type="checkbox"/>	my_atm_learner	2 minutes ago	
<input type="checkbox"/>	plot	5 hours ago	
<input type="checkbox"/>	resource	3 months ago	
<input type="checkbox"/>	template	3 months ago	
<input type="checkbox"/>	cpas-vis.ipynb	5 hours ago	19.9 kB
<input type="checkbox"/>	VERSION.txt	a day ago	9 B

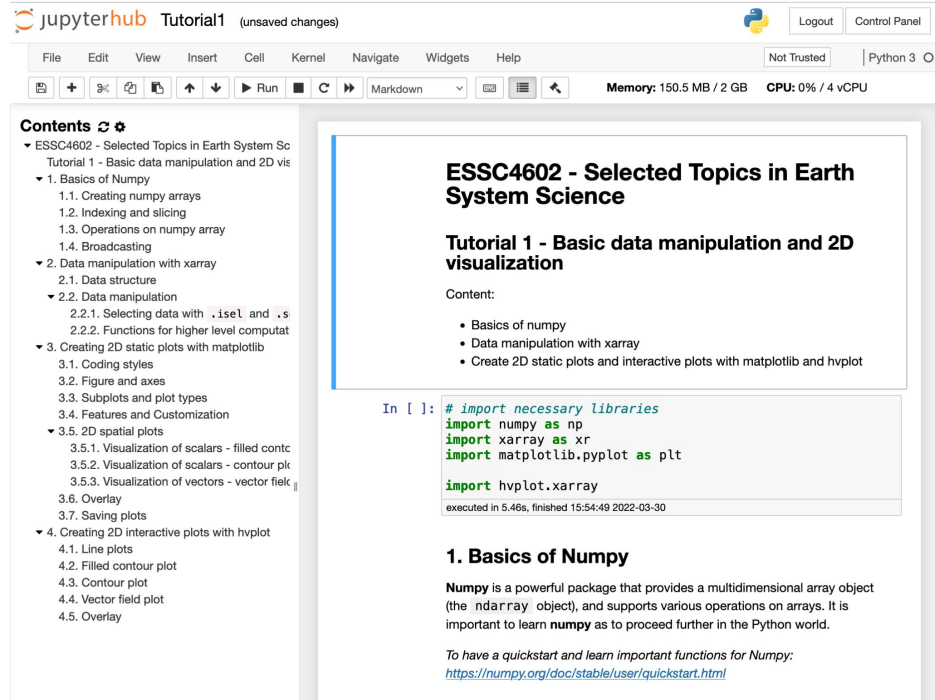


jupyterhub Logout Control Panel

Files Running Clusters Nbextensions

Select items to perform actions on them. Upload New ↻

	Name	Last Modified	File size
<input type="checkbox"/>	..	seconds ago	
<input type="checkbox"/>	data	2 minutes ago	
<input type="checkbox"/>	Tutorial1.ipynb	2 minutes ago	62.4 kB



jupyterhub Tutorial1 (unsaved changes) Logout Control Panel

File Edit View Insert Cell Kernel Navigate Widgets Help Not Trusted Python 3

Memory: 150.5 MB / 2 GB CPU: 0% / 4 vCPU

Contents

- ESSC4602 - Selected Topics in Earth System Science
 - Tutorial 1 - Basic data manipulation and 2D visualization
 - 1. Basics of Numpy
 - 1.1. Creating numpy arrays
 - 1.2. Indexing and slicing
 - 1.3. Operations on numpy array
 - 1.4. Broadcasting
 - 2. Data manipulation with xarray
 - 2.1. Data structure
 - 2.2. Data manipulation
 - 2.2.1. Selecting data with .isel and .isel
 - 2.2.2. Functions for higher level computation
 - 3. Creating 2D static plots with matplotlib
 - 3.1. Coding styles
 - 3.2. Figure and axes
 - 3.3. Subplots and plot types
 - 3.4. Features and Customization
 - 3.5. 2D spatial plots
 - 3.5.1. Visualization of scalars - filled contour
 - 3.5.2. Visualization of scalars - contour plot
 - 3.5.3. Visualization of vectors - vector field
 - 3.6. Overlay
 - 3.7. Saving plots
 - 4. Creating 2D interactive plots with hvplot
 - 4.1. Line plots
 - 4.2. Filled contour plot
 - 4.3. Contour plot
 - 4.4. Vector field plot
 - 4.5. Overlay

ESSC4602 - Selected Topics in Earth System Science

Tutorial 1 - Basic data manipulation and 2D visualization

Content:

- Basics of numpy
- Data manipulation with xarray
- Create 2D static plots and interactive plots with matplotlib and hvplot

```
In [ ]: # import necessary libraries
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt

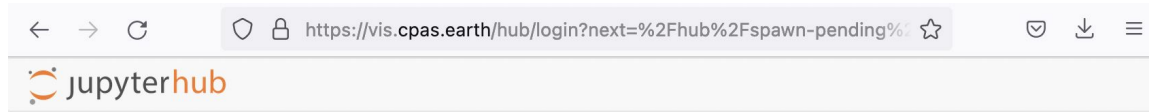
import hvplot.xarray
executed in 5.46s, finished 15:54:49 2022-03-30
```

1. Basics of Numpy

Numpy is a powerful package that provides a multidimensional array object (the `ndarray` object), and supports various operations on arrays. It is important to learn **numpy** as to proceed further in the Python world.

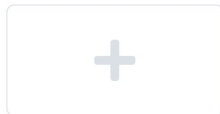
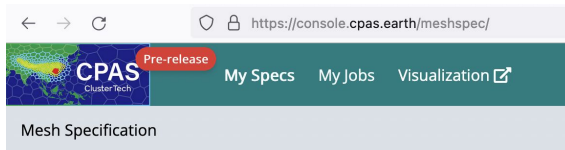
To have a quickstart and learn important functions for Numpy:
<https://numpy.org/doc/stable/user/quickstart.html>

If idle too long / lost connection

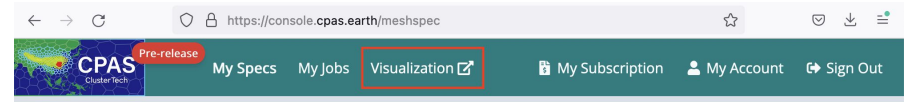


401 : Unauthorized

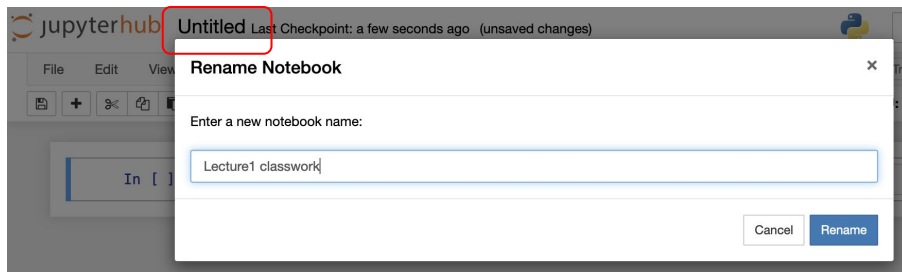
1 Refresh this tab:



2 Click "Visualization" link again.

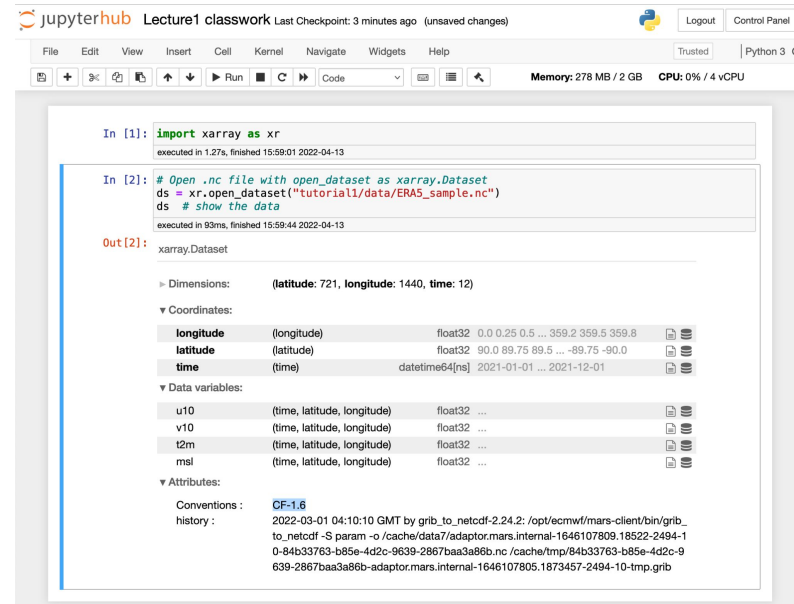


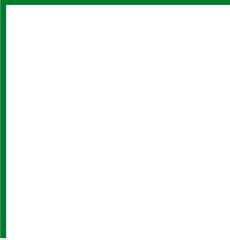
Create my own classwork Notebook



Task: Open a .nc data file

Hint: xarray provides [open_dataset\(\)](#)






Lab 1

Hello My Jupyter Server

All students should be done - Voice out if you haven't.
You cannot do later Labs if you have not done this step



netCDF file data structure

Data structure:

Data variables

- n-D array with name and attributes

Dimensions

- Name of dimensions and length

jupyterhub Lecture1 classwork Last Checkpoint: 3 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Navigate Widgets Help Trusted Python 3

Memory: 278 MB / 2 GB CPU: 0% / 4 vCPU

```
In [1]: import xarray as xr
executed in 1.27s, finished 15:59:01 2022-04-13
```

```
In [2]: # Open .nc file with open_dataset as xarray.Dataset
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
ds # show the data
executed in 93ms, finished 15:59:44 2022-04-13
```

Out [2]: xarray.Dataset

Dimensions: (latitude: 721, longitude: 1440, time: 12)

Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359.2 359.5 359.8
latitude	(latitude)	float32	90.0 89.75 89.5 ... -89.75 -90.0
time	(time)	datetime64[ns]	2021-01-01 ... 2021-12-01

Data variables:

u10	(time, latitude, longitude)	float32	...
v10	(time, latitude, longitude)	float32	...
t2m	(time, latitude, longitude)	float32	...
msl	(time, latitude, longitude)	float32	...

Attributes:

Conventions : CF-1.6

history : 2022-03-01 04:10:10 GMT by grib_to_netcdf-2.24.2: /opt/ecmwf/mars-client/bin/grib_to_netcdf -S param -o /cache/data7/adaptor.mars.internal-1646107809.18522-2494-1 0-84b33763-b85e-4d2c-9639-2867baa3a86b.nc /cache/tmp/84b33763-b85e-4d2c-9 639-2867baa3a86b-adaptor.mars.internal-1646107805.1873457-2494-10-tmp.grib

https://www.unidata.ucar.edu/software/netcdf/workshops/2011/datamodels/

2011 Unidata NetCDF Workshop > The Two NetCDF Data Models

4.11 A Convention for Coordinates: Coordinate Variables

Coordinate Variables contain the coordinate values for a dimension.



A variable with the same name as a dimension is called a *coordinate variable*. Many programs that read netCDF files recognize and use any coordinate values they find.

<https://www.unidata.ucar.edu/software/netcdf/workshops/2011/datamodels/NcCVars.html>



Lab 2

What is “CF” convention?

5 minutes



What you can do with a ds object?

Pre-condition:

```
import xarray as xr
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
```

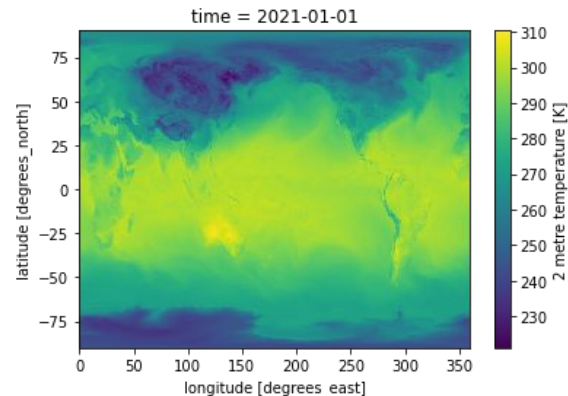
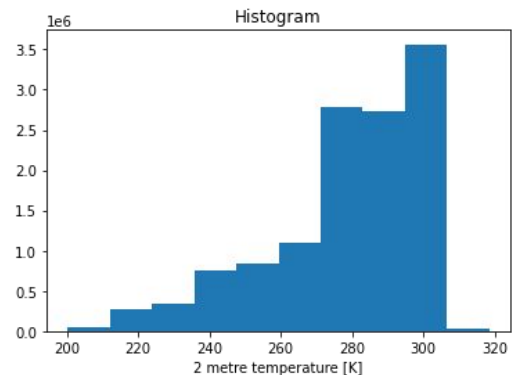
Explore the following:

1. `ds.t2m.plot()`
2. `ds.t2m.isel(time=0).plot()`

Advice:

Get used to typing code! => Proficiency

If you copy-and-paste, you don't know what you are coding.



How does the software know ??

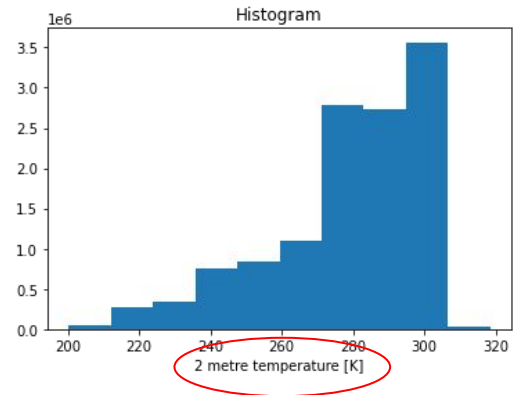
I didn't put in code

"2 metre temperature [K]"

"latitude [degrees_north]"

"Longitude [degrees_east]"

1.ds.t2m.plot()



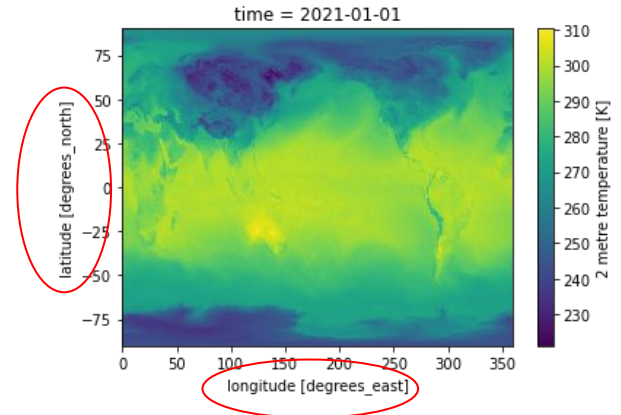
Explore here:

```
In [2]: # Open .nc file with open_dataset as xarray.Dataset
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
ds # show the data
executed in 78ms, finished 17:08:20 2022-04-13

Out[2]: xarray.Dataset

Dimensions: (latitude: 721, longitude: 1440, time: 12)
Coordinates:
  longitude (longitude) float32 0.0 0.25 0.5 ... 359.0
  latitude (latitude) float32 90.0 89.75 89.5 ... 88.25
  time (time) datetime64[ns] 2021-01-01 ... 2021-01-12
Data variables:
  u10 (time, latitude, longitude) float32 ...
  v10 (time, latitude, longitude) float32 ...
  t2m (time, latitude, longitude) float32 ...
  msl (time, latitude, longitude) float32 ...
Attributes:
  Conventions: CF-1.6
  history: 2022-03-01 04:10:10 GMT by grib_to_netcdf-2.24.2: /opt/ecmwf/mars-client/bin/grib_to_netcdf -S param -o /cache/data7/adaptor.mars.internal-1646107809.18522-2494-10-84b33763-b85e-4d2c-9639-2867baa3a86b.nc /cache/tmp/84b33763-b85e-4d2c-9639-2867baa3a86b-adaptor.mars.internal-1646107805.1873457-2494-10-tmp.grib
```

ds.t2m.plot()



Nice data producer

The data producer already put meta-data to netCDF attributes, following the CF convention.

```
# Open .nc file with open_dataset as xarray.Dataset
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
ds # show the data
```

executed in 78ms, finished 17:08:20 2022-04-13

xarray.Dataset

Dimensions: (latitude: 721, longitude: 1440, time: 12)

Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359...
units :	degrees_east		
long_name :	longitude		
latitude	(latitude)	float32	90.0 89.75 89.5 ...
units :	degrees_north		
long_name :	latitude		
time	(time)	datetime64[ns]	2021-01-01 ... 202...

Data variables:

u10	(time, latitude, longitude)	float32	...
v10	(time, latitude, longitude)	float32	...
t2m	(time, latitude, longitude)	float32	...
msl	(time, latitude, longitude)	float32	...

Attributes:

Conventions : CF-1.6
history : 2022-03-01 04:10:10 GMT by grib_to_netcdf-2.24.2: /opt/ecmwf/mars-client/bin/grib_to_netcdf -S param -o /cache/data7/adaptor.mars.internal-1646107809.18522-2494-10-84b33763-b85e-4d2c-9639-2867baa3a86b.nc /cache/tmp/84b33763-b85e-4d2c-9639-2867baa3a86b-adaptor.mars.internal-1646107805.1873457-2494-10-tmp.grib

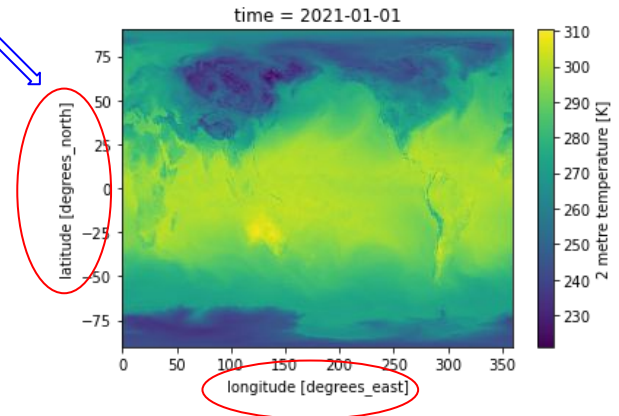
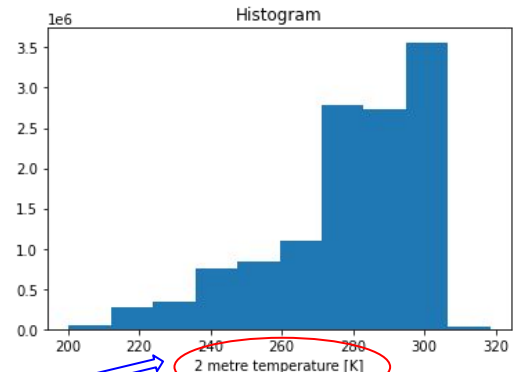
1.ds.t2m.plot()

Domain-specific:
The package knows CF

Labour
replaced by
software packages



ds.t2m.plot()



Meet hvplot(), and its (default) bokeh backend

Pre-condition:

```
import xarray as xr
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc")
```

Explore the following:

1. `ds.t2m.plot()`
2. `ds.t2m.isel(time=0).plot()`
3. `import hvplot.xarray`
`ds.t2m.isel(time=0).hvplot()`

Reference:

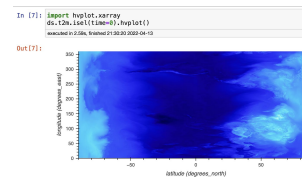
https://hvplot.holoviz.org/user_guide/Geographic_Data.html

Question:

- 1) Is `plot()` or `hvplot()` smarter?
Which one missed what desirable feature?

Hint: Use your curiosity and mouse to explore.

Problem: Longitude as y-axis
Latitude as x-axis



- 2) How to make `hvplot()` do exactly what I want?

Hint:

- i) Give more specific message to `hvplot()` via input arguments.
- ii) Try the arguments: `x`, `y`, `geo`
- iii) Can't recognize the place in 2D horizontal plot? Try argument: `coastline`

Putting Coordinates to the correct axis

Another example put Latitude and Longitude to the correct axis

https://hvplot.holoviz.org/user_guide/Gridded_Data.html

How does it do that?

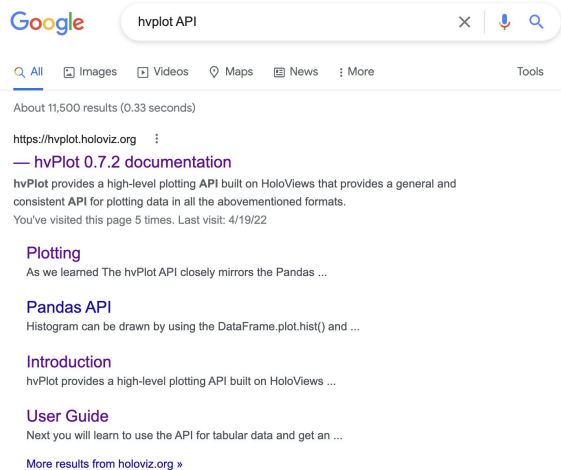
- Attributes 'axis' in Coordinates

» Dimensions: (lat: 25, lon: 53, time: 2920)

▼ Coordinates:

lat	(lat)	float32	75.0 72.5 70.0 ... 20.0 17.5 15.0	 
standard_name :	latitude			
long_name :	Latitude			
units :	degrees_north			
axis :	Y			
lon	(lon)	float32	200.0 202.5 205.0 ... 327.5 330.0	 
standard_name :	longitude			
long_name :	Longitude			
units :	degrees_east			
axis :	X			
time	(time)	datetime64[ns]	2013-01-01 ... 2014-12-31T18:00:00	 

Any other way? Google "hvplot API"



Google hvplot API

About 11,500 results (0.33 seconds)

<https://hvplot.holoviz.org>

— hvPlot 0.7.2 documentation

hvPlot provides a high-level plotting API built on HoloViews that provides a general and consistent API for plotting data in all the abovementioned formats.

You've visited this page 5 times. Last visit: 4/19/22

Plotting
As we learned The hvPlot API closely mirrors the Pandas ...

Pandas API
Histogram can be drawn by using the DataFrame.plot.hist() and ...

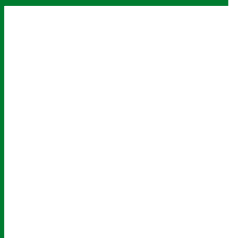
Introduction
hvPlot provides a high-level plotting API built on HoloViews ...

User Guide
Next you will learn to use the API for tabular data and get an ...

[More results from holoviz.org »](#)

https://hvplot.holoviz.org/user_guide/Plotting.html

`<data obj>.hvplot(x="Latitude", y="Longitude")`




Lab 2

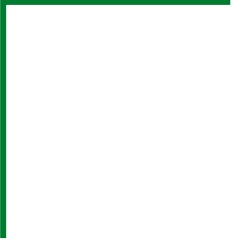
What is “CF” convention?

Time's up


One solution to get something plotted out by hvplot() in a way you may like:

```
import hvplot.xarray
ds.t2m.isel(time=5).hvplot(
    x='longitude', y='latitude',
    geo=True, coastline='110m')
```





Use of multiple
CPU core
/ under RAM size limitation



xarray chunks and parallel processing

Xarray is integrated with Dask

<https://docs.xarray.dev/en/latest/user-guide/dask.html>

([Dask documentation](#) itself talks more about Pandas.)

← → ↻ <https://docs.dask.org/en/stable/> 🔒 ☆ ⚙️ □ C ☰



What is a Dask array?

Dask divides arrays into many small pieces, called *chunks*, each of which is presumed to be small enough to fit into memory.

	s	s	s
5	(x, 0, 0)	(x, 0, 1)	(x, 0, 2)
5	(x, 1, 0)	(x, 1, 1)	(x, 1, 2)
5	(x, 2, 0)	(x, 2, 1)	(x, 2, 2)
5	(x, 3, 0)	(x, 3, 1)	(x, 3, 2)

Unlike NumPy, which has eager evaluation, operations on Dask



Dask

Dask is a flexible library for parallel computing in Python.

Dask is composed of two parts:

1. **Dynamic task scheduling** optimized for computation. This is similar to *Airflow*, *Luigi*, *Celery*, or *Make*, but optimized for interactive computational workloads.
2. **"Big Data" collections** like parallel arrays, dataframes, and lists that extend common interfaces like *NumPy*, *Pandas*, or *Python iterators* to larger-than-memory or distributed environments. These parallel collections run on top of dynamic task schedulers.

Tell xarray to cut the n-D array data into chunks

- Add argument
 - `chunks={...}`
- to
 - `xr.open_dataset("/the/data.nc", ...)`

Collections

(create task graphs)

Dask Array

Dask DataFrame

Dask Bag

Dask Delayed

Futures

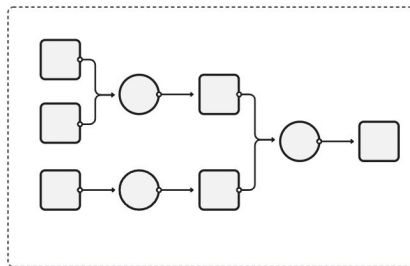


Task Graph



Schedulers

(execute task graphs)



Single-machine
(threads, processes,
synchronous)

Distributed

High level collections are used to generate task graphs which can be executed by schedulers on a single machine or a cluster.



xarray chunks and parallel processing

```
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc", chunks={})  
ds # show the data
```

executed in 132ms, finished 11:21:58 2022-04-20

xarray.Dataset

► Dimensions: (latitude: 721, longitude: 1440, time: 12)

▼ Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359...	📄	☰
latitude	(latitude)	float32	90.0 89.75 89.5	📄	☰
time	(time)	datetime64[ns]	2021-01-01 ... 202...	📄	☰

▼ Data variables:

u10	(time, latitude, longitude)	float32	dask.array<chunks...	📄	☰
v10	(time, latitude, longitude)	float32	dask.array<chunks...	📄	☰
t2m	(time, latitude, longitude)	float32	dask.array<chunks...	📄	☰
msl	(time, latitude, longitude)	float32	dask.array<chunks...	📄	☰

Choose a dimension to decompose

```
ds = xr.open_dataset("tutorial1/data/ERA5_sample.nc", chunks={'longitude': 100})  
ds # show the data
```

executed in 68ms, finished 11:27:13 2022-04-20

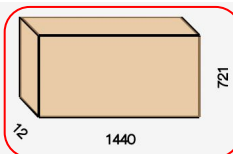
ds.msl

executed in 42ms, finished 11:22:19 2022-04-20

xarray.DataArray 'msl' (time: 12, latitude: 721, longitude: 1440)



	Array	Chunk
Bytes	47.53 MiB	47.53 MiB
Shape	(12, 721, 1440)	(12, 721, 1440)
Count	2 Tasks	1 Chunks
Type	float32	numpy.ndarray



▼ Coordinates:

longitude	(longitude)	float32	0.0 0.25 0.5 ... 359.2 359.5 359.8	📄	☰
latitude	(latitude)	float32	90.0 89.75 89.5 ... -89.75 -90.0	📄	☰
time	(time)	datetime64[ns]	2021-01-01 ... 2021-12-01	📄	☰

CPU: 0% / 4 vCPU

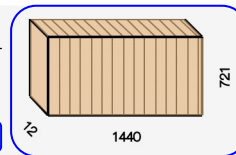


ds.msl

executed in 34ms, finished 11:29:09 2022-04-20

xarray.DataArray 'msl' (time: 12, latitude: 721, longitude: 1440)

	Array	Chunk
Bytes	47.53 MiB	3.30 MiB
Shape	(12, 721, 1440)	(12, 721, 100)
Count	16 Tasks	15 Chunks
Type	float32	numpy.ndarray



Lazy evaluation

The object memorize what needs to be done.

=> task graph

Execute the computation only when triggered

`.compute()`

`.load()`

Triggers computation.

```
In [9]: lazy = ds.msl.min()  
lazy
```

executed in 16ms, finished 11:37:29 2022-04-20

```
Out[9]: xarray.DataArray 'msl'
```



	Array	Chunk
Bytes	4 B	4.0 B
Shape	0	0
Count	46 Tasks	1 Chunks
Type	float32	numpy.ndarray

► Coordinates: (0)

► Attributes: (0)

```
In [10]: lazy.compute()
```

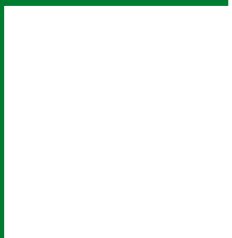
executed in 321ms, finished 11:37:50 2022-04-20

```
Out[10]: xarray.DataArray 'msl'
```

array(92538.625, dtype=float32)

► Coordinates: (0)

► Attributes: (0)



Lab 3

Make multi-cores busy

5 minutes



Try massive data... and monitor the system

- Open a terminal, run
 - (base) cpasvis@3cxxxxxxxxfb:~\$ `htop`
- Separate the browser tab as a window
- View both Notebook and htop windows
 - side by side

```
import xarray as xr

ds = xr.open_mfdataset(

"/home/cpasvis/work/atm_learner/demo_mangkhut/
rsim/2018091300/atm/diag.2018-09-13_*.nc",

    concat_dim='Time', combine="nested",

    chunks={"nCells": 1000}

)
```

```
import numpy as np

lazy = (np.exp(np.log(ds.qc)) - ds.qc).max()

lazy
```

```
%%time

lazy.compute()
```

- Try also

```
xr.open_mfdataset(

"/multiple/files/something*.nc",

    parallel=True)
```


https://vis.cpas.earth/user/Google_111125757376876243367/notebooks/my

jupyterhub Dask multi-core computing (autosaved)

CPU: 115.7% / 4 vCPU
 Memory: 1.6 GB / 2 GB

```

Type      float32  numpy.ndarray  55
  
```

Coordinates: (0)

Attributes:

```

long_name :      Cloud water mixing ratio
units      :      kg kg-1
  
```

In [46]: `import numpy as np`
`lazy = (np.exp(np.log(ds.qc)) - ds.qc).max()`
`lazy`
 executed in 60ms, finished 10:53:21 2022-04-20

Out[46]: `xarray.DataArray 'qc'`

	Array	Chunk
Bytes	4 B	4.0 B
Shape	0	0
Count	23449 Tasks	1 Chunks
Type	float32	numpy.ndarray

Coordinates: (0)

Attributes: (0)

In [*]: `%time`
`lazy.compute()`
 execution queued 10:55:35 2022-04-20

In []:

https://vis.cpas.earth/user/Google_111125757376876243367/terminals/2

jupyterhub

```

1 [||||| 35.1%] 6 [ 0.0%] 11 [ 0.0%] 16 [ 0.0%]
2 [ 0.0%] 7 [||||| 29.1%] 12 [ 0.0%] 17 [||||| 19.2%]
3 [ 0.0%] 8 [ 1.3%] 13 [ 0.0%] 18 [ 0.0%]
4 [ 0.0%] 9 [ 0.0%] 14 [ 0.0%] 19 [||||| 28.7%]
5 [ 0.0%] 10 [ 0.0%] 15 [||||| 17.9%] 20 [||||| 24.0%]
Mem| 3.28G/55.0G Tasks: 9.74 thr 2 running
Swp| 3.87M/557M Load average: 1.10 0.72 0.45
Uptime: 504 days(!), 01:36:09
  
```

PID	USER	PRI	NI	VR	RES	SHR	S	CPU%	MEM%	TIME+	Command
74	cpasvis	20	0	4689M	1357M	42924	R	107.2	2.4	0:48.99	/opt/conda/bin/python
108	cpasvis	20	0	4689M	1357M	42924	S	21.9	2.4	0:07.59	/opt/conda/bin/python
105	cpasvis	20	0	4689M	1357M	42924	S	20.6	2.4	0:07.40	/opt/conda/bin/python
107	cpasvis	20	0	4689M	1357M	42924	S	20.6	2.4	0:07.46	/opt/conda/bin/python
106	cpasvis	20	0	4689M	1357M	42924	S	16.6	2.4	0:07.42	/opt/conda/bin/python
7	cpasvis	20	0	476M	110M	16984	S	1.3	0.2	0:47.00	/opt/conda/bin/python
531	cpasvis	20	0	8072	3860	3120	S	0.7	0.0	0:05.48	htop
957	cpasvis	20	0	8072	4008	3272	R	0.0	0.0	0:00.22	htop
26	cpasvis	20	0	476M	110M	16984	S	0.0	0.2	0:00.77	/opt/conda/bin/python
535	cpasvis	20	0	476M	110M	16984	S	0.0	0.2	0:00.19	/opt/conda/bin/python
54	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.15	/opt/conda/bin/python
42	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.32	/opt/conda/bin/python
80	cpasvis	20	0	4689M	1357M	42924	S	0.0	2.4	0:00.21	/opt/conda/bin/python
1	cpasvis	20	0	2500	596	528	S	0.0	0.0	0:00.28	tmux -- start-no
25	cpasvis	20	0	476M	110M	16984	S	0.0	0.2	0:00.01	/opt/conda/bin/python
30	cpasvis	20	0	476M	110M	16984	S	0.0	0.2	0:00.00	/opt/conda/bin/python
31	cpasvis	20	0	476M	110M	16984	S	0.0	0.2	0:00.04	/opt/conda/bin/python
35	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.00	/opt/conda/bin/python
36	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.01	/opt/conda/bin/python
37	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.02	/opt/conda/bin/python
38	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.00	/opt/conda/bin/python
39	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.00	/opt/conda/bin/python
40	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.00	/opt/conda/bin/python
41	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:00.00	/opt/conda/bin/python
29	cpasvis	20	0	503M	48920	14248	S	0.0	0.1	0:01.04	/opt/conda/bin/python
47	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.00	/opt/conda/bin/python
48	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.00	/opt/conda/bin/python
49	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.01	/opt/conda/bin/python
50	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.00	/opt/conda/bin/python
51	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.00	/opt/conda/bin/python
52	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.00	/opt/conda/bin/python
53	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.00	/opt/conda/bin/python
55	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.03	/opt/conda/bin/python
56	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.04	/opt/conda/bin/python
57	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.04	/opt/conda/bin/python
58	cpasvis	20	0	3217M	126M	41280	S	0.0	0.2	0:00.03	/opt/conda/bin/python

F1 Help F2 Setup F3 Search F4 Alter F5 Free F6 Sortby F7 Nice F8 Nice F9 Kill F10 Quit

Execution time can be subtle ...

```
chunks = {}
```

Care about it only when the job is really slow.

- just let you know such support is right there.
- more other parallel computing skill can be explored ...
if repeated execution time > your learning + development time,
or you think it is interesting itself (hello computer scientist)

```
In [33]: %%time  
         lazy.compute()
```

```
executed in 7.22s, finished 10:42:01 2022-04-20
```

```
CPU times: user 462 ms, sys: 334 ms, total: 796 ms  
Wall time: 7.2 s
```

```
chunks={"nCells": 1000}
```

```
In [37]: %%time  
         lazy.compute()
```

```
executed in 2.88s, finished 10:44:00 2022-04-20
```

```
CPU times: user 2.53 s, sys: 629 ms, total: 3.16 s  
Wall time: 2.86 s
```



Lab 3

Make multi-cores busy

Time's up



More about Dask and effective data processing

Out of syllabus, only for those interested in computer science:

<https://docs.xarray.dev/en/latest/user-guide/dask.html#chunking-and-performance>

<https://docs.dask.org/en/stable/array.html>

<https://stepanhoyer.com/2015/06/11/xray-dask-out-of-core-labeled-arrays/>

(An old article, but stated a case Dask can handle data with size bigger than RAM size and chunks)

<https://examples.dask.org/delayed.html#Visualize-computation>

<https://www.youtube.com/watch?v=mDriGxaXQT4>
(highlight: 6:00 - 8:54, 11:00 - 12:20)

Chunking and performance

The `chunks` parameter has critical performance implications when using Dask arrays. If your chunks are too small, queuing up operations will be extremely slow, because Dask will translate each operation into a huge number of operations mapped across chunks. Computation on Dask arrays with small chunks can also be slow, because each operation on a chunk has some fixed overhead from the Python interpreter and the Dask task executor.

Conversely, if your chunks are too big, some of your computation may be wasted, because Dask only computes results one chunk at a time.

A good rule of thumb is to create arrays with a minimum chunksize of at least one million elements (e.g., a 1000x1000 matrix). With large arrays (10+ GB), the cost of queuing up Dask operations can be noticeable, and you may need even larger chunksizes.

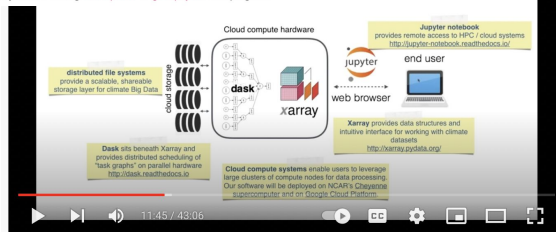
Tip

Check out the dask documentation on [chunks](#).

Optimization Tips

With analysis pipelines involving both spatial subsetting and temporal resampling, Dask performance can become very slow in certain cases. Here are some optimization tips we have found through experience:

1. Do your spatial and temporal indexing (e.g. `.sel()` or `.isel()`) early in the pipeline, especially before calling `resample()` or `groupby()`. Grouping and



2018-01-17 ESIP Tech Dive: "Pangeo: A big data science platform", Ryan Abernathy and Matthew Rocklin