Identification d'objets

Brève présentation de l'outil

Najda Villefranque

CNRM

Kézaco ?

- Un outil d'identification d'objets
- Pensé pour être générique
- ...en contrepartie, assez basique

À développer, compléter, améliorer !

A quoi ça sert ?

- Identifier des structures cohérentes
- Dans des champs 2D, 3D ou 4D
- À partir d'un masque binaire défini par l'utilisateur
- Les cellules du masque = 1 si $\in$ objet, 0 sinon

Structures cohérentes

Amas de cellules "1" contiguës

Comment ça marche ?

- Un module Python : identification_methods.py
- Readme généré automatiquement :
  `python identification_methods.py`

```
FUNCTIONS
    identify(ncdfFile, listVarNames, funcCalcMask, argsCalcMask=(), dimMask=(), name='objects', diagonals=True, cyclic=(-2, -1),
 delete=0, rename=False, criteria=None, write=True, rmbounds=False, overwrite=False, unique=False)
        identify and labels user defined objects in field, with several mandatory and optional arugments
        mandatory arguments :
            - ncdfFile : name of netCDF file containing user data. Object fields will be written in the same file.
            - listVarNames : a list of strings of any size, containing the names of the fields use to describe objects
            - funcCalcMask : a user defined function to compute a mask (0 / 1) from physical user fields. The arguments are pass
ed in the same order as defined in listVarNames.
        optional arguments :
            - argsCalcMask : a tuple of arguments to provide to the funcCalcMask function. Default is an empty tuple
            - dimMask : a tuple of int to select the dimensions of the first extracted variable in listVarNames, that will be th
e dimensions of the objects field
              -> by default, the objects field is defined along all the dimensions of the first extracted variable in listVarNam
es
            - name : "objects" (default) or a string to name the object field in the netCDF file
            - diagonals : True (default) or False, to consider diagonal cells as adjacent
            - cyclic : None or a tuple to identify the cyclic dimensions (default are the last 2 dimensions)
            - delete : 0 (default) or integer nb>0 to delete objects containing less than nb cells
            - rename : True or False (default) to rename objects with contiguous labels
            - criteria : a user function that should return an integer giving the category of the object. The arguments are pass
ed in the same order as defined in listVarNames.
            - write : True (default) or False, to write the computed fields in the netCDF file
            - rmbounds : Apply filter to remove bounds (if original data)
```

Arguments obligatoires

- **ncdfFile** : un fichier netCDF qui contient les champs à analyser
- **listVarNames** : la liste des noms des variables, dans le fichier ncdfFile, dont vous avez besoin pour définir vos objets
- **funcCalcMask** : une fonction Python qui reçoit en entrée les variables demandées dans listVarNames, et qui doit retourner un ”masque”

```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT'] # list of relevant variables.

def clouds(dictVars) :
    rc = dictVars['RCT']
    mask=rc*0.   # same shape as RCT
    mask[rc>0]=1 # cell is in cloud if liquid water mixing ratio is positive
    return mask

# To read more info on the function identify, uncomment the following line
#help(identify)
objects, types = identify(ncfile, listVarsNames, clouds)
```

Comportement par défaut

- La fonction **funcCalcMask** ne prend que le dictionnaire des variables en argument
- Le champs objets est écrit dans le fichier **ncdfFile**, sous le nom ”objects”
- Le champs objets a la même dimension que la première variable demandée dans **listVarNames**
- Les deux dernières dimensions du masque sont considérées comme cycliques : les objets voisins aux bords sont regroupés

Passer des arguments à la fonction masque

Les cellules sont nuageuses si la quantité d'eau liquide est supérieure à un seuil passé en argument à la fonction masque

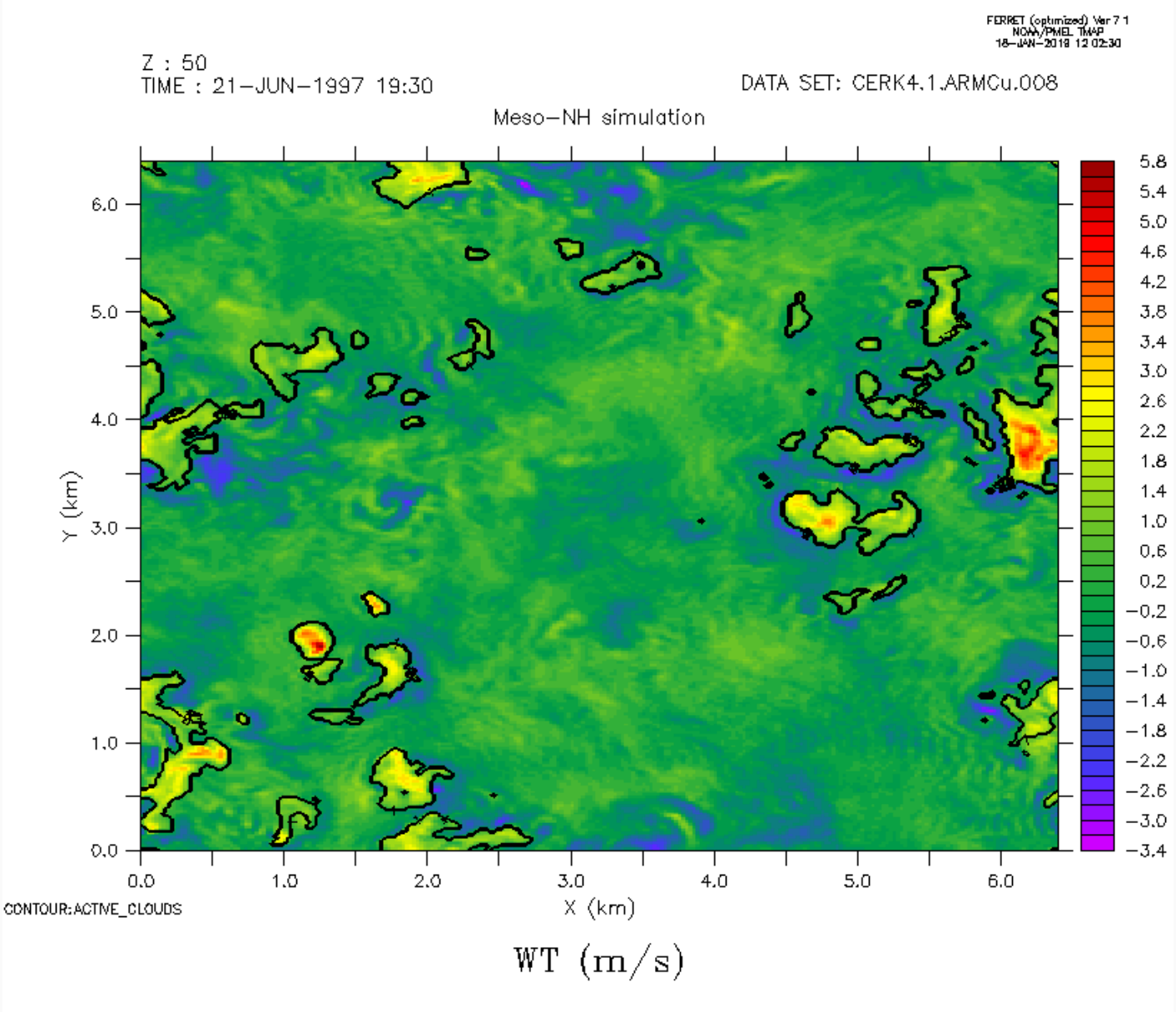
```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT'] # list of relevant variables.

def clouds(dictVars,threshold) :
    rc = dictVars['RCT']
    mask=rc*0.   # same shape as RCT
    mask[rc>threshold]=1 # cell is in cloud if liquid water mixing ratio
                         # is greater than a threshold
    return mask

# To read more info on the function identify, uncomment the following line
#help(identify)
objects, types = identify(ncfile, listVarsNames, clouds, argsCalcMask=1.e-6)
```

Passer des arguments à la fonction masque (2)

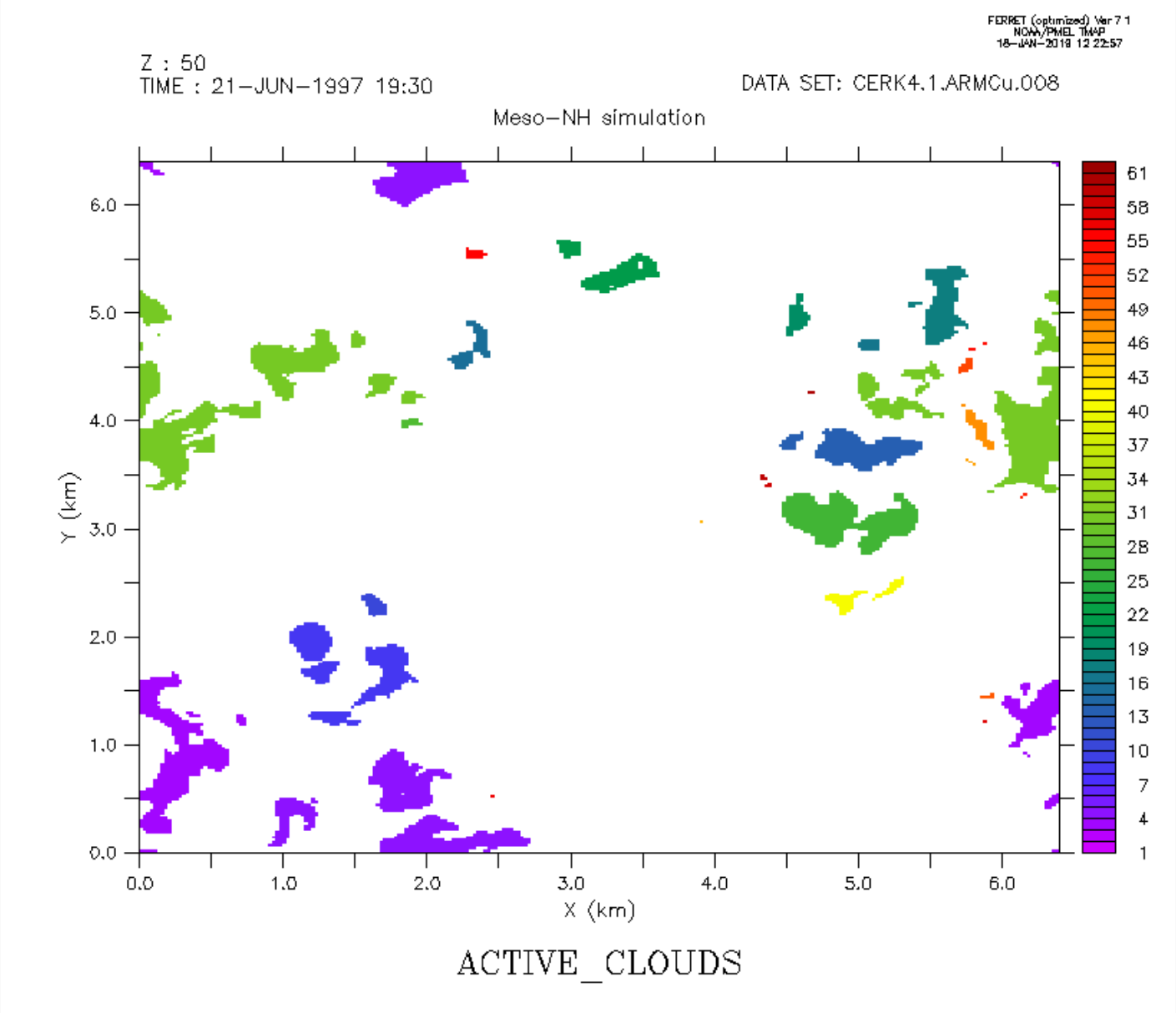
On ajoute la condition ”vitesse verticale supérieure à un deuxième seuil passé en argument de la fonction masque”

```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT','WT'] # list of relevant variables.

def clouds(dictVars,threshold1,threshold2) :
    rc = dictVars['RCT']
    wt = dictVars['WT']
    mask=rc*0.   # same shape as RCT
    mask[(rc>threshold1) & (wt>threshold2)]=1
    # cell is in cloud if liquid water mixing ratio
    # is greater than a threshold1 and if vertical
    # wind is greater than a threshold2
    return mask

# To read more info on the function identify, uncomment the following line
#help(identify)
objects, types = identify(ncfile, listVarsNames, clouds, argsCalcMask=(1.e-6,0.),
                          name="active_clouds")
```

Vent vertical (shade) et nuages actifs (contour) à 1.25km



Nuages actifs identifiés à 1.25km

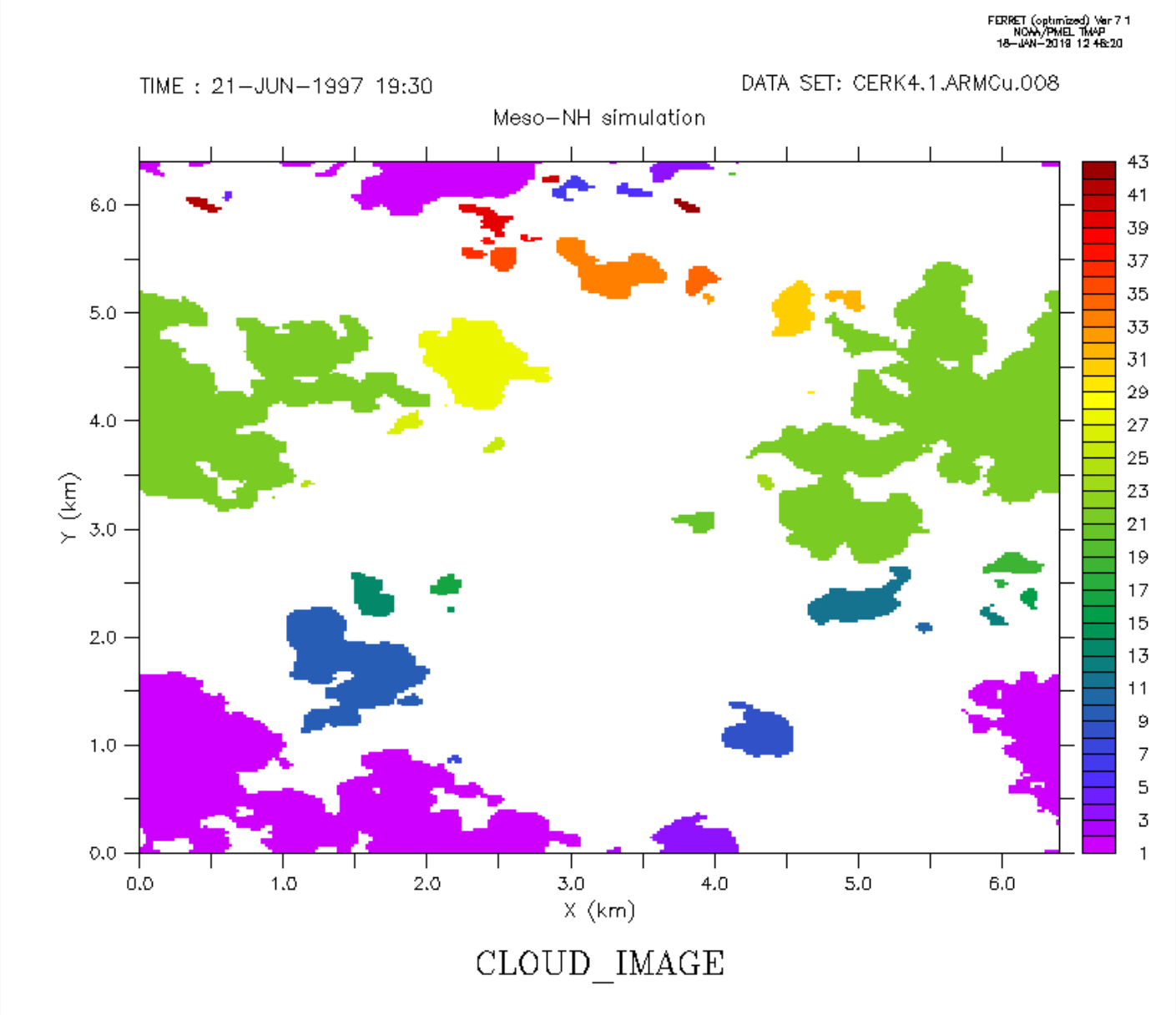


Changer les dimensions du masque

Le champ objet est 2D, construit à partir d'un champ netCDF 3D

```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT'] # list of relevant variables.

def clouds(dictVars) :
    rc = dictVars['RCT']        # original shape is t,z,y,x
    newshape = (rc.shape[0],rc.shape[2],rc.shape[3])
    mask=np.zeros(newshape)     # new shape is t,y,x
    mask[np.sum(rc,axis=1)>0]=1 # cell is in cloud if liquid water path is positive
    return mask

# To read more info on the function identify, uncomment the following line
#help(identify)
objects, types = identify(ncfile, listVarsNames, clouds,dimMask=(0,2,3),
                 rename=True, name="cloud_image")
```

Nuages 2D identifiés

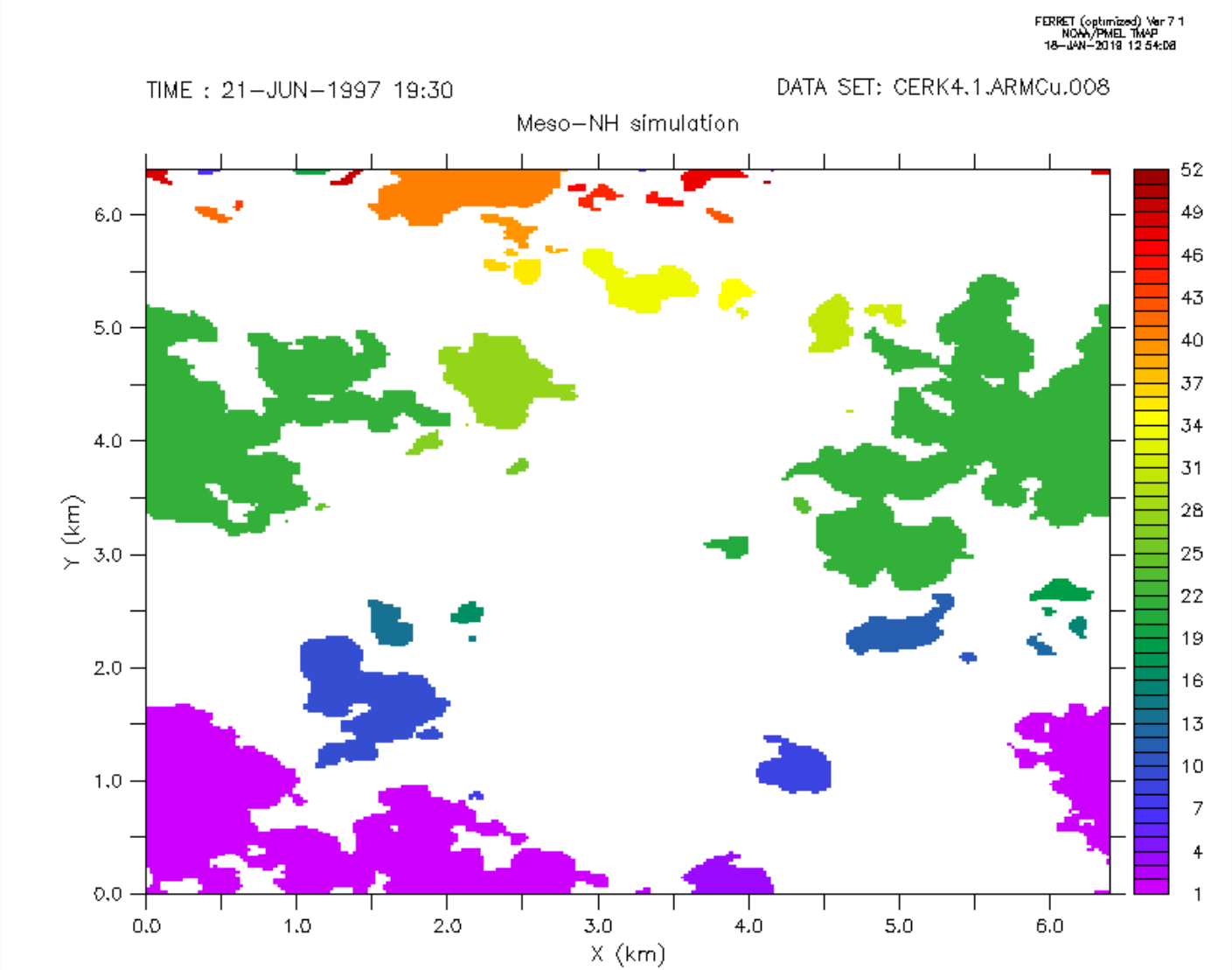


Changer les conditions cycliques

Les bords sont ouverts en y=0 et y=Ly, et cycliques en x=0 et x=Lx

```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT'] # list of relevant variables.

def clouds(dictVars) :
    rc = dictVars['RCT']        # original shape is t,z,y,x
    newshape = (rc.shape[0],rc.shape[2],rc.shape[3])
    mask=np.zeros(newshape)     # new shape is t,y,x
    mask[np.sum(rc,axis=1)>0]=1 # cell is in cloud if liquid water path is positive
    return mask

# To read more info on the function identify, uncomment the following line
#help(identify)
objects, types = identify(ncfile, listVarsNames, clouds, cyclic=(-1,),
                 dimMask=(0,2,3), rename=True, name="cloud_image_openY")
```

Nuages 2D identifiés



Autres arguments optionnels

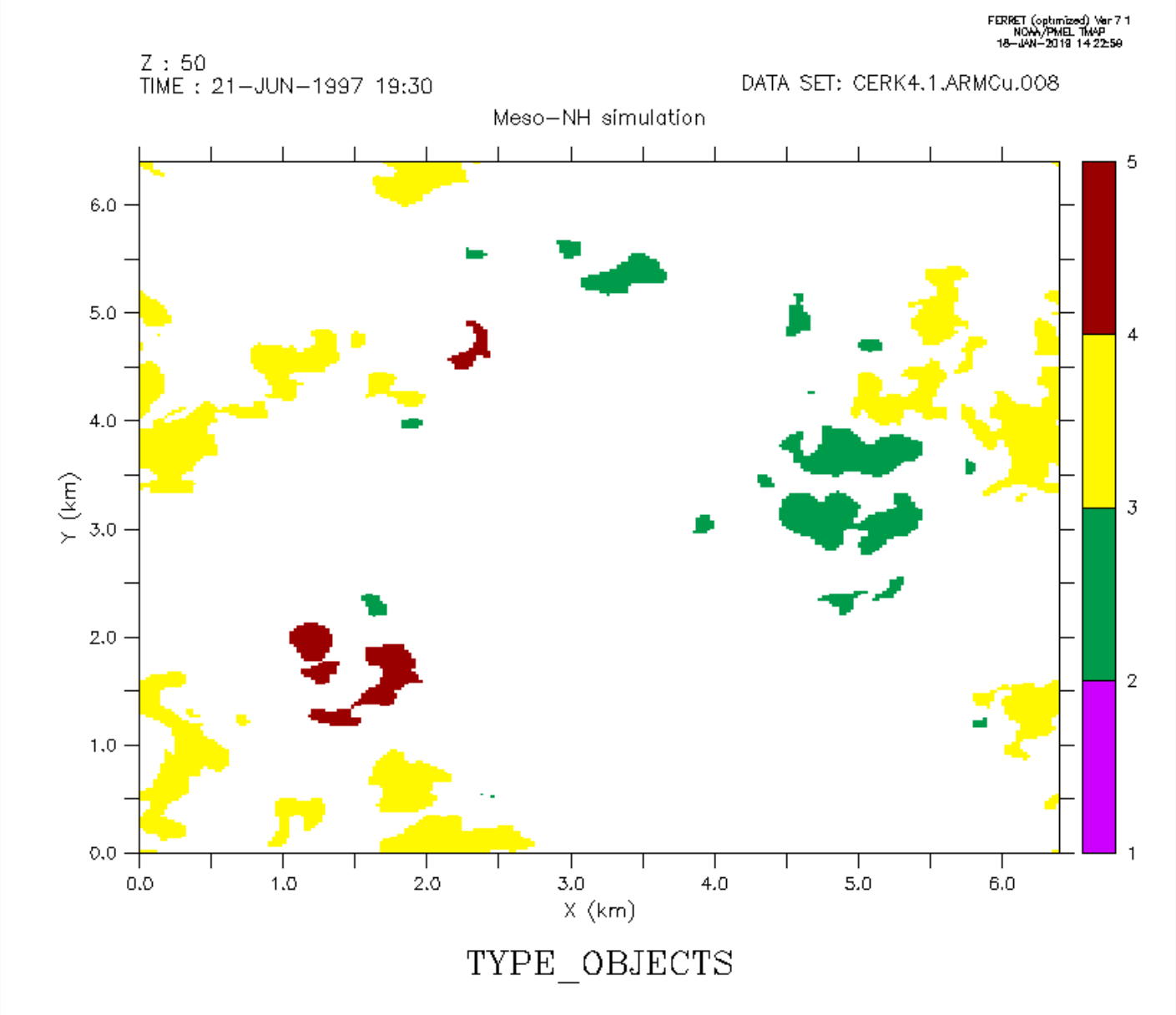
- **diagonals** (boolean) : la définition de contiguité inclut les diagonales ?
- **delete** (int) : seuil pour supprimer les objets composés d'un petit nombre de cellules
- **rename** (boolean) : renommer les objets pour leur donner des étiquettes contiguës ?
- **write** (boolean) : écrire le champ objet dans le netCDF ?
- **overwrite** (boolean) : écraser les champs objets de même nom dans le netCDF sans demander ?

Trier les objets

Post-traitement : on créé un deuxième champ où les objets sont classés en différentes catégories

```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT','VLEV'] # list of relevant variables.

def clouds(dictVars) :
    rc = dictVars['RCT']
    mask=rc*0.   # same shape as RCT
    mask[rc>0]=1 # cell is in cloud if liquid water mixing ratio is positive
    return mask

def criteria(dictVars, indsObj) :
    t,z,y,x=(indsObj)
    zz = dictVars['VLEV']
    if max(zz[z,y,x])>2. : num=1 # top height > 2 km
    else : num=2
    if len(np.unique(z))>20 : num+=2 # depth > 500m
    # 1 : top >  2km and depth <= 500m
    # 2 : top <= 2km and depth <= 500m
    # 3 : top >  2km and depth >  500m
    # 4 : top <= 2km and depth >  500m
    return num

# To read more info on the function identify, uncomment the following line
#help(identify)
objects, types = identify(ncfile, listVarsNames, clouds,
                 criteria=criteria)
```

Nuages 3D triés, à z=1.25km



Tracking temporel

Toujours un amas de cellules contiguës...

Deux objets distincts à un pas de temps qui se réunissent au pas de temps suivant = le même objet

Une partie d'un objet qui se détache reste le même objet

Importer la fonction identify et le module numpy

```
import os,sys
sys.path.append('path/to/scripts_objects/src/')
from identification_methods import identify
import numpy as np
```