Angular Momentum for Eddy Detection and tracking Algorithm

User manual

by Briac Le Vu, Alexandre Stegner and Evangelos Moschos,

June 2021

Versions history

V1 – published version (Le Vu *et al*. 2018)

V2.1 – parallelised version (with Romain Pennel, 2018)

V2.2 (current version) – still in development for analysing bigger size of models outputs.

Acknowledgment

I would like to acknowledege the financial support during the developpement of the algorithm: ANR-DYNED, CNES-Odatis and the Ecole Polytechnique at Palaiseau which employ me. Most of the algorithm was settled thanks to the AVISO products processed by SSALTO/DUACS and distributed by AVISO+ (https://www.aviso.altimetry.fr) with support from CNES.

AMEDA would not be what it is without the original code from Nencioli *et al.* (2010) and the adaptation of the code by Mkhinini *et al*. (2014) who include the LNAM computation. I particularly thanks Alexandre Stegner to trust me to take care of the code all along the years. Alexandre is associated as well as Evangelos Moschos to the writing of this manual. I should also associate to the developpement of AMEDA my others colleagues of the Laboratoire de Meteorologie Dynamique (LMD): Romain Pennel, Thomas Arsouze, Artemis Ioanou, Remi Laxenaire, Alexandre Barboni, as well as the external collabrators: Franck Dumas (SHOM), Antoine Delpoulle (CLS) and Cori Pegliasco (LEGOS).

I also have to warmly thank the first users, as far as I know, for their interest for AMEDA and the different questions they had during the use of AMEDA. They use different input files on different area of the ocean: Charly de Marez, Adam Ayouche and Mathieu Morvan (from the LOPS) for their study on HYCOM outputs and AVISO data on the Arabian sea; Jiaxun Lee (HFR data on the south of Taiwan); Léo Costa Aroucha (AVISO data on the North Brazil offshore ocean); Ozge Yelekci (NEMO ouputs on Mediterranean deep water); Quentin-Boris Barral (HYCOM ouputs on Baleares front area); Amaru Marquez (AVISO data on Bay of California and offshore waters); Wentao Ma (AVISO data on the south China Sea); Aviv Solodoch (model ouputs and satelite data); Furqon Azis Ismail (on Indonesian Sea); Yingli Zhu; Fehmi Dilmahamod and Arne Bedinger (model ouputs on Madagascar strait and Labrador Sea); Yingjun Zhang (HFR data on California coast); Amirhossein Barzandeh (AVISO on Persian Gulf). Their remarcks and questions helped me a lot to write and improve this user manual.

Content

What is AMEDA?

AMEDA, for Angular Momentum Eddy Detection and tracking Algorithm, is an algorithm that follows Nencioli algorithm in Matlab (Nencioli *et al.* 2010) to detect and to track moving eddy structures in the ocean with a strong geostrophic signature. The method records time series of center positions of the eddies, along with their size and intensity; it gives also an history of the merging and splitting events. It is a hybrid detection method based on the computation of Local Normalised Angular Momentum developed at the LMD by Nadia Mkhinini and Alexander Stegner (Mkhinini *et al.* 2014) which allows to locate a center of an eddy of any intensity and propose a contour based on the maximum radial velocity. The advantage of the algorithm resides in its capacity to analyse various gridded velocity fields of oceanic currents (velocimetry imagery, high-frequency radar, satellite, numerical model – see figure 1) without fine tuning of the parameters. It has been calibrated on SSALTO/DUACS products, velocities from PIV laboratory imagery and ROMS model output (Le Vu *et al.* 2018). The code is freely available and regularly updated on github to improve its efficiency and to correct bugs. Feel free to test it on your own field and ask for any advice if necessary.

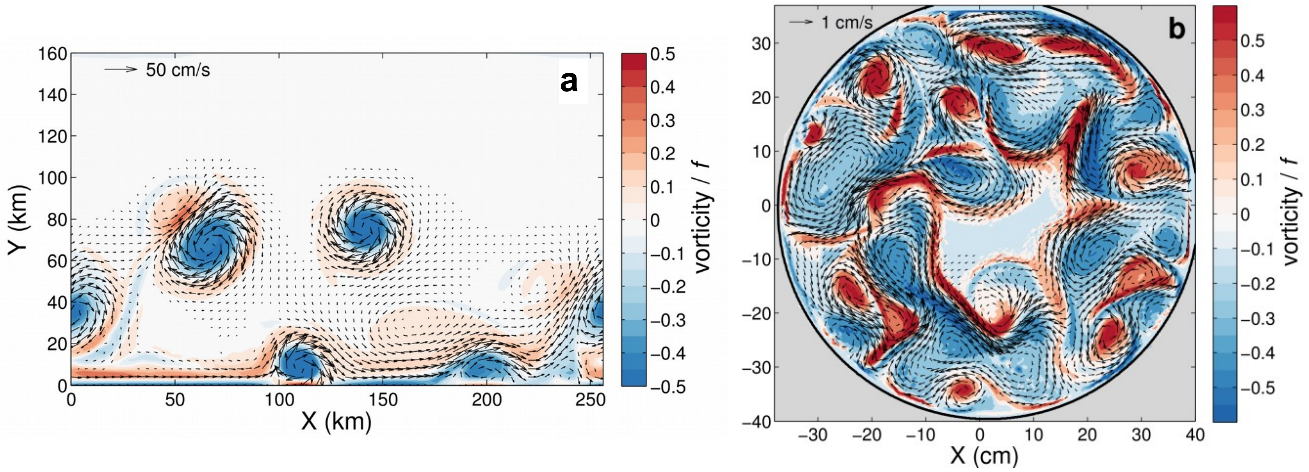

Figure 1: Examples of velocity fields analysed in Le Vu *et al.* 2018: a- Velocity and vorticity field from ROMS simulation in idealized channel (Cimoli *et al.* 2017) ; b- Vorticity from PIV in a tank experiment.

A nice example of the AMEDA's application is the ANR project DYNED-Atlas (https://www.lmd.polytechnique.fr/dyned/). The goal of this project was to build a 17 years database of surface intensified eddies in the Mediterranean sea and the Arabian sea. You can watch this movie to have an idea of the result: https://vimeo.com/327967941.

For any questions or remarks about the code, its adaptation to your input files or help concerning the exploitation of the output files, please feel free to contact the authors: briac.le-vu@lmd.polytechnique.fr or astegner@lmd.polytechnique.fr.

First steps with AMEDA

1. What do you require?

To run AMEDA at is best performance you need a Matlab License with at least version 7.3 and some of Matlab's toolboxes:

- Matlab tested versions (R009a, R2014a and 2017a).

- Toolboxes:

- **Curve Fitting Toolbox** to specifically compute the shape coefficient of the V-R profile. You must disactivate the *streamlines* key if you don't want to use this toolbox.
- **Parallel Computing Toolbox** to run the parallelised version of the code on a single node (tested up to 32 CPUs for CROCO outputs fields on datarmor cluster).
- **Distributed Computing Server** should be necessary to run the parallelised version of the code on the un cluster. This has not been tested yet, but here is the documentation: https://fr.mathworks.com/support/product/DM/installation/ver_current.html.
- **Matlab Compiler.** This tool allows to compile the AMEDA code and produce an executable which does not need a Matlab licence to run. This toolbox helps particularly in analyzing a long simulation when there are a limited number of available licences.
- **m_map** (Pawlowicz, 2020) or any tools for projecting your results on a map.

2. Where to find AMEDA?

You can download the stable version of the code from the repository AMEDA/version_vX. The up-to-date version with the last development still in testing phase are located on AMEDA/master branch. The AMEDA tutorial, along with a sample that can be analysed on a standalone verison producing pre- and post-processing tools, can be found on AMEDA-tutorial.

You can clone with the git command the AMEDA code https://github.com/briaclevu/AMEDA or go to the repository and click the download button as illustrated on the figure 2 below.

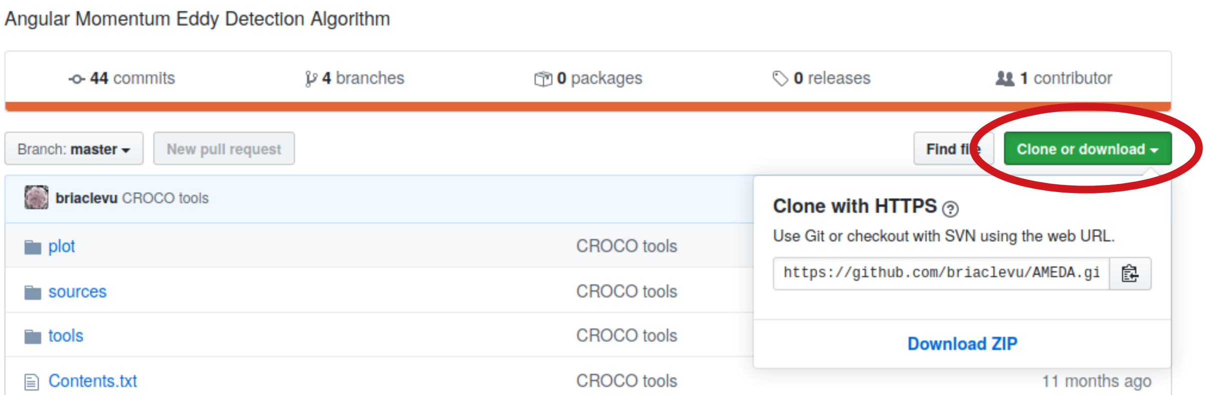

Figure 2: the way to download the code with the *clone* button on the master branch

3. How to deploy AMEDA?

In the main directory called **AMEDA** you can find the main routines which have to be modified by user to run the code. You may place this directory and all its content on your usual Matlab directory. From the downloaded files you also have to untar **Rossby_radius.tar** which contains matrixes of the Rossby radius, an essential part of the parameters computation automatically computed by AMEDA during the early steps of the algorithm. Figure 3 shows the detailed deployment of the code.

Routines of the code developed for AMEDA are located in the *sources* subdirectory and routines taken and adapted from https://fr.mathworks.com/ or from model post-processing tools are located in the *tools* subdirectory, alongwith a list of associated licences in the *licences.txt* file. To increase AMEDA speed performance, we suggest you to build MEX files according to your Matlab environnement from the few files provided in C or Fortran language.

AMEDA

Contents.txt

MAIN_AMEDA_MED_nopool.m	MAIN_AMEDA_MED_multi.m
MAIN_AMEDA_CROCO_test.m	MAIN_AMEDA_DYNED_MED.m
MAIN_AMEDA_NEMO_test.m	MAIN_AMEDA_NEMO_multi.m
keys_sources_AVISO.m	keys_sources_AVISO_DYNED_MED_adt.m
keys_sources_NEMO_test.m	keys_sources_CROCO_test.m

Rossby_radius

global_KNTN.mat	read_global_Rossby_radius.m	Rossby_radius.mat
global_Rossby_radius.mat	Rossby_Radius_WOA_Barocl1.mat	rossrad.dat

sources

load_fields_HYCOM.m	load_fields_AVISO.m	load_fields_CROCO.m
load_fields_NEMO.m	load_fields_PIV.m	load_fields_HFR.m
load_fields_ROMS.m	concat_eddy.m	compute_best_fit.m
compute_curve.m	compute_ellip.m	compute_psi.m
eddy_dim.m	integrate_vel.m	max_curve.m
make_netcdf_from_tracks.m	mean_radius.m	min_dist_shapes.m
scan_lines.m	mod_init.m	mod_eddy_centers.m
mod_fields.m	mod_eddy_params.m	mod_eddy_shapes.m
mod_eddy_tracks_nopool.m	mod_eddy_tracks_pool.m	mod_merging_splitting.m

tools

license.txt	assignmentoptimal.m	assignmentoptimal.c	
InPolygon.m	InPolygon.c	fitellipse.m	csf.m
nanstd.m	nansum.m	nanvar.m	nanmean.m
get_Dx_from_ll.m	get_missing_val_2d.m	get_z_croco.m	sw_dist.m
sw_dist2.m	sw_dist2.F	TaubinNTN.m	vinterp.m

plot

plot_shapes_example1.m	plot_shapes_example2.m	plot_tracking_example3.m
plot_tracking_example4.m		

Figure 3: Routines and components of the downloaded code of AMEDA

AMEDA algorithm workflow in details

1. User routines

In the AMEDA folder itself, you will find 2 kinds of matlab files and also the file **Contents.txt** which describes the workflow followed by the keys routines during a full analysis by AMEDA. The 2 others kind of routines that user needs to modify for running AMEDA are:

- **MAIN_AMEDA*.m** is the Matlab routine you launch to run AMEDA. You need a good understanding of this routine to be able to modify *MAIN_AMEDA.m* files for your own usage. The postfix *multi* is used when there is a need to time split a very long input file into more than one analysis in order to prevent a possible crash during a long computation time. For a multi years run for example, AMEDA_MAIN_*_muli.m will run, for each year, the detection analysis and will record the results in files with the year as a postname accordingly. Then AMEDA_MAIN_*_multi.m will run the tracking by loading the concatenated output files containing the yearly results.
  The postfixes *nopool* and *pool* define two versions of the same routine in which a '*parfor loop'* or a simple '*for loop'* is used, respectively.
- **keys_sources*.m** contain definitions of paths and fields names, as well as setting of keys and options depending on the simulation you want to execute. These distinct routines have been created at different moments of the code development. Thus, some of them may have missing keys or parameters that you will have to add if you want to use these AMEDA sources.

Details on modifying and using these routines are described extensively in the next section, **running AMEDA**.

2. Parameters computation

Running one of the *MAIN_AMEDA*.m* with its associated *keys_sources*.m* begins by the parameters computation (**mod_eddy_params.m**). Many parameters like the Coriolis parameter or deformation radius are computed in this routine. You can consult the different parameters imposed or computed in Table 1. It is possible to modify these parameters by changing the value directly in the routine or you can also remove some from the routine and define them in your user routine *keys_sources*.m*. We will describe in detail what user parameters need to be defined in the section **running AMEDA**.

All the 2D-fields and parameters necessary for running AMEDA as well as the different paths of data and results defined in *keys_sources*m* are recorded in the file **param_eddy_tracking.mat** at the end of this stage. This file is recorded in the folder defined by **path_out** in *keys_sources*.m*. **Warning**: Defining correctly the *path_out* variable is very important because it will be added to the Matlab path as a 'global variable' and scanned by the code when AMEDA loads a file. For this reason, if the value of *path_out* must change due to changes in the configuration or the simulation, please start a new Matlab session.

3. Main routines

After that the parameter have been computed as described above, and that the Matlab structures pre-allocation or update (**mod_init.m**) is successfully completed, the computational part of AMEDA algorithm will be executed. The execution of the AMEDA algorithm utilizes the four following routines:

1. **mod_fields.m**: Computes, through a finite spatial elements method, the 2D fields of the following variables: kinetic energy, divergence, vorticity, Okubo-Weiss, Local Okubo-Weiss (LOW) and Local Normalised Angular Momentum (LNAM). Among them, the two latter (LOW and LNAM) will be used to detect eddy centers by the next script. LOW and LNAM and the others fields are included in a structure array called {**detection_fields(t)**} and save in the **fields.mat** file located in *path_out*.

2. **mod_eddy_centers.m**: Detects the potential eddies centers present in the domain using the LNAM and LOW fields. In fact, when the precision of the grid is lower than the deformation radius like for AVISO fields, AMEDA interpolates the LNAM and LOW on a thinner grid by a factor *resol* (cf. Table 1) . This interpolation is made to locate centers at a precision higher than the size of the typical deformation radius of the domain. Firstly, the position of the maxima max(|LNAM(LOW<0)>*K*|) are resolved, with *K* refering to the LNAM threshold defined in *mod_eddy_params.m*. Then, the potential centers saved are these max LNAM surrounded by at least two closed contours of the streamfunction (*psi*) or *ssh*. Potential eddy centers are saved/updated as the structure array {**center(t)**} in the file **eddy_centers.mat** in *path_out*.

3. **mod_eddy_shapes.m**: Computes the shapes (if any) of eddies identified by their potential centers in the previous stage. Characteristic shapes are closed contours defined by the maximum mean tangential velocity along the closed streamlines around the center, hereby called as *speed radius*. These contours have a maximum (i.e. *nR_lim*, cf. Table 1) and a minimum (i.e. *nRmin*, cf. Table 2b) size. The corresponding identified eddy centers are saved as {**centers2(t)**} in the file **eddy_centers.mat** in *path_out* and *speed radius* together with other features as {**shapes1(t)**} in **eddy_shapes.mat** in *path_out*. The routine also records the outermost contours in {**shapes1(t)**} and defines contours which include two different eddy centers, called *dual eddies*, as {**shapes2(t)**} in the same **eddy_shapes.mat** file.

4. **mod_eddy_tracks.m**: Performs eddy tracking through the eddy centers positions and shapes features computed as described above. Eddy tracks are inter-connected by comparing the detected eddy fields at successive time steps. An eddy at time '*t*' is assumed to be the new position of an eddy of the same type detected at time '*t-dt*', only if the distance *d* between the two centers is smaller than the distance D:

$$D = V_{eddy}*(1+dt)/2 + r_{max}n + r_{max}m,$$

where $r_{max}n$ is the mean radius of each eddy time-averaged on the last *D_stp* tracked time steps (cf. Table 1) and $r_{max}m$ the new eddy radius to be tested at '*t*'. If an eddy does not connect any new detection after a number of time steps '*Dt*' (cf. Table 1), it is considered dissipated and the track *n* ends. In case two or more eddies are found within the same area defined by D, the track is connected to the centers which minimize the cost funtion of an assignment matrix considering all the past centers N at ‘*t-dt*’ which are not already dissipated and the new ones M at ‘*t*’. The cost function is a NxM matrix of the C_{nm} elements:

$$C_{nm} = \mathrm{sqrt}\left(d/D\ ^2 + \delta R/R_{max}\ ^2 + \delta Ro/Ro\ ^2 + dt/Dt/2\ ^2 \right),$$

Unfiltered tracked eddies are saved/updated as the structure array {**traks(n)**} in **eddy_tracks.mat** in *path_out* where 'n' is the index number of the recorded track.

On the first three stages, the routines are not time step dependant, and thus the parallelization is applied in the *MAIN_AMEDA*.m* routine to compute many time steps at the same time thanks to the ‘*parfor loop*’. On the last stage, tracking is dependent to the previous steps and thus the ‘*parfor loop’* cannot be applied with the time step. A ‘*parfor loop’* is included inside **mod_eddy_tracks_pool.m** at the cost function calculation. Unfortunately, the gain in terms of computation time is not important when the number of eddies tracked increases too much (i.e. in a big domain or a long run) due to the increase of the allocation time to the pools with the size of the *tracks* structure. While we have not modified the code to limit the size of this matrix we advice you to only use **mod_eddy_tracks_nopool.m**.

4. Flags and filtering

At the end of the AMEDA process, the unfiltered *tracks* structure is analysed to resolve merging and splitting events from eddy tracks. A time filtering is also applied, as two times the turnover time if *cut_off* is 0 (by default) or the value of *cut_off* otherwise. In order to avoid time filtering and to save every eddy tracks whatever their life length you must set *cut_off* to the value *dps* which is settled in your *keys_sources*.m* file (cf. Table 2b). The result of this flagging (merging or splitting) and filtering process is recorded as {**tracks2(n)**} in **eddy_tracks2.mat** in *path_out*.

Table 1: default parameters fixed or calculated in *mod_eddy_params.m*

Parameter	Definition	Value or *Dim*	Units	Range	From*
Dx	Grid horizontal spacing	*I x J*	km	-	*Grid*
f	Coriolis parameter	*I x J*	s^{-1}	-	*Grid*
g	Gravitation constant	9.8	$m.s^{-2}$	-	-
DH	*psi/ssh* 2D-field spacing	0.002	m	0.001-0.002	*Test*
nH_lim	Number of max scans	200	#	100-400	*Test*
n_min	Min contour vertices	6	#	4-6	*Test*
epsil	Min diff increase to follow scans	1	%Vmax %Δeta	1-2	*Test*
k_vel_decay	Min decrease to detect Vmax	97	%Vmax	95-99	*Test*
nR_lim	Max radius limit	100 (~no limit)	Rd	4-7	*Test*
Np	Grid point for curvature calculation	3	#	2-5	*Test*
nrho_lim	Max length with negative curvature	0.2	2πRmax	0.2-0.5	*Test*
lat_min	Minimum latitude of potential centers	5	Latitude (°)	1-15	*Test*
dc_max	Max distance between 2 eddy centers	3.5	Rmax	-	*Calcul*
V_eddy	Empirical eddy center speed	6.5	$km.day^{-1}$	0/6.5	*Ref1*
Dt	Max delay after last detection	2 (model) 10 (AVISO)	days	1-15	*Test*
cut_off	Time-filtered eddy life time	0 (2 turnovers)	days	0/1-100	*Test*
D_stp	Time period of averaged features	2 (model) 4 (AVISO)	steps	2-5	*Test*
N_can	Number of tested tracks association	30	#	-	-
Rd	First Baroclinic Rossby Radius of Deformation	*I x J*	km	0.1-1.5	*Ref2*
gama	Rd resolution	*I x J*	pixels/Rd	>0.1	*Rd, Dx*
resol	Grid interpolation coefficient	1 (model) 3 (AVISO)	#	1-3	*Rd, Dx*
K	|LNAM| threshold of potential centers	0.7	LNAM	0.4-0.8	*Test*
b	Half box size for LNAM calculation	*I x J*	pixels	>1	*gama*
Rb	LNAM box check	*I x J*	gama	0.5-1.5	*b, gama*
bx	Half box size of streamline scan	*I x J*	pixels	>1	*gama*
P_i	2D-field parameter *P* in the interpolated grid by *resol*	*I*resol x J*resol*	-	-	*P, resol*

*: *Grid* comes from the x and y grid from input file; *Calcul* comes from an idealized Gaussian eddy velocity field; *Ref1* from Mkhinini et al. 2014; *Ref2* from Chelton et al. 1998; *P* is a 2D-field parameter (ie. *Dx, f, Rd, b, bx, gama*). *Test* comes from the optimisation of the number of detected centers in the tested range.

Running the AMEDA walkthrough

1. Build input files

Apart from the two user routines (*MAIN_AMEDA*.m* and *keys_sources*.m*) that you need to adapt to your purpose, you first need to prepare your input files in agreement with the AMEDA specifications. The input interface is taken in charge by the routines **load_fields*.m**. Many specifications are allowed, but only some are absolutely necessary:

- Every processed file must be of reasonable size for your computer buff memory (typically few Go <10 Go). Keep this in mind when producing your input files.
- Time is given in terms of unitless and equally spaced steps as the time step duration is fixed in the **keys_sources*.m** script.
- The dimensions of the *fields* output of **load_fields*.m** needs to be [field] = (x | longitude, y | latitude).
- The velocity units as output of **load_fields*.m** must be in $m.s^{-1}$.

You could change these specifications. In that case you will need to produce your own **load_fields*.m** with your own requirement. For example, create a **load_fields_*model*.m** to read all the different fields from the same file like a 3D output model containing all the fields.

You might refer to the proper **load_fields_*source*.m** routine belonging to your ***source*** (AVISO, NEMO,…) and you can even modify it to properly build your input files. This routine is called by AMEDA at different steps of the process. Some sources like AVISO needs a lot of pre-processing of the input and others like CROCO files can be directly read through the model output files. Because the input files are manifold, the code does not provide any pre-processing tools to prepare the input netcdf files, but in the tutorial you will find some tools to process the AVISO input files for AMEDA.

2. Choose or prepare the *rossby_radius* file

The first assumption when using AMEDA is that you want to detect eddies of size close to the first baroclinic Rossby radius of deformation (**Rd**). Indeed, Rd is the first needed parameter from which the other parameters are calculated in *mod_eddy_params.m*. AMEDA code includes a global Rd 2D-field in the file **Rossby_radius/global_Rossby_radius.mat**, built from Chelton et al. 1998, that you can use for any area in the ocean (see Figure 4). This global field is somewhat coarse (1°x1°). If you work on the Mediterranean sea, you should use **Rossby_radius/Rossby_radius.mat** (see Figure 4) with a thinner resolution (1/8°x1/8°). You might also create your own Rd *.mat* file by following the script *Rossby_radius/read_global_Rossby_radius.m* which creates a *.mat* file from a *.dat* file. AMEDA must find in your *.mat* file a 2D matrix for Rd and also 2D matrixes of the same size as Rd for the longitude and the latitude grid, named **lon_Rd** and **lat_Rd**, respectively. Then you must specify the name of your Rd matrix in your *keys_sources*.m* (as 'name_Rd'). **Warning**: You should pay attention to use a continuous longitude grid in your area of interest. This is to apply a proper interpolation in *mod_eddy_params.m* routine. For instance, avoid longitude ranges of 180°E to -180°W or 359°E to 0°.

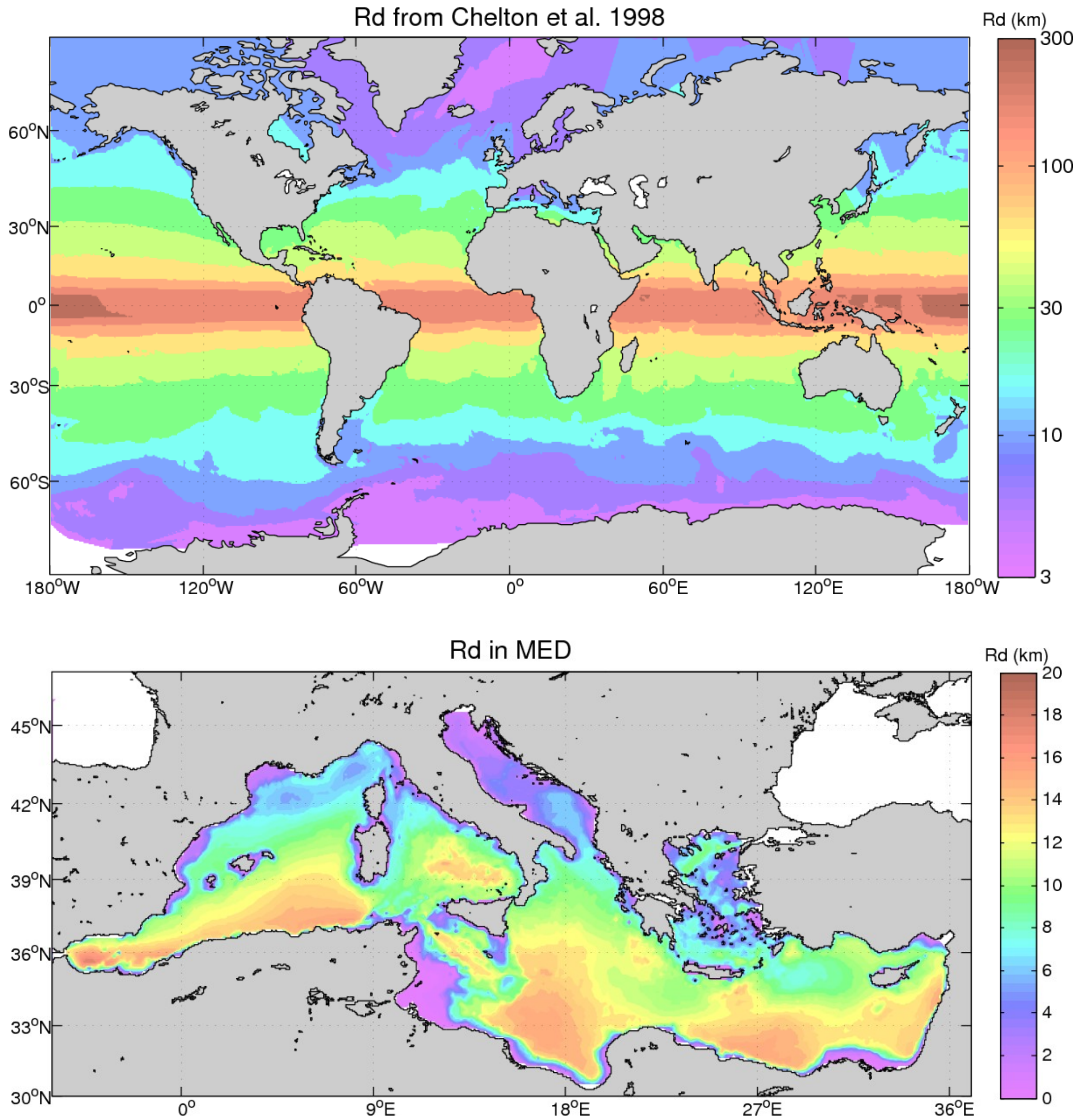

Figure 4: First baroclinic Rossby radius of deformation for the global ocean 1°x1° from Chelton et al. 1998 (up panel) and for the Mediterranean sea at 1/8° x 1/8° from MEDATLAS (bottom panel)

3. Choose and set *MAIN_AMEDA*

MAIN_AMEDA.m* are the master routines executing all the stages described in the previous section, **AMEDA workflow**. Each *MAIN_AMEDA*.m* depicts relevant schema depending on the pre-processed input files available and the specific configuration you want to apply (i.e. Do you have more than 1 CPU available? Do you need to produce more than one output file or not?…). You could also create a MAIN_AMEDA_*one_step*.m routine to run only one step if you need, for example... Once you have your input netcdf files in the right format and units you can run MAIN_AMEDA. Of course you need to add the paths AMEDA/, sources/, tools/, Rossby_radius/ and m_map/ in the Matlab path with the command *addpath*. This can be done through the **start.m** script, as it is indicated in the *MAIN_AMEDA*.m* routines, where you add your own paths or commands that you usually use with Matlab.

In pratical, to run AMEDA, you must choose and copy a *MAIN_AMEDA*.m* and a *keys_sources*.m* file, based on your problem setup and hardware availability (see questions below) and rename them based on the sources of your fields and the configuration you use. For example, copy and create MAIN_AMEDA_AVISO_test.m and keys_sources_AVISO_test.m from MAIN_AMEDA_MED.m and from keys_sources_AVISO_MED.m. These two files have user sections that can be modified to fit your input files specification, your options and also the active keys you want to use.

The choice of the *MAIN_AMEDA*.m* script you need to copy depends on:

- **is the source you want to analyse already known by AMEDA?** If not, copy a *MAIN_AMEDA*.m* not far from your sources. For example NEMO source for HYCOM.
- **Does hardware setup allows for using the parpool function?** If not copy a *MAIN_AMEDA*nopool.m* or simply fix the *cpus* variable to 1 and change the '*parfor loop*' into '*for loop*' in the MAIN_AMEDA routine.
- **do you have more than 1000 steps or a big input file more than 10Go?** If yes copy a *MAIN_AMEDA_*_multi.m* where you will find an explicit loop on the time step embedding the entire stages *mod_fields*, *mod_eddy_centers* and *mod_eddy_shapes* with a recording of intermediate files. For example, from an input file containing 10 years of velocities, you could save 10 intermediate files, each file conaining yearly analysis.

Once you have chosen, copied and renamed accordingly your *MAIN_AMEDA*.m*, modify the following parameters:

- **source**: a string which drives the load_fields_*source*.m used to read fields from the netcdf input files including their interpolation, degradation or regridding of the native grid to fit the Matlab format used by the mains routines. Ex: 'AVISO', 'NEMO', 'HFR', 'CROCO',…
- **keys**: a string which specifies the configuration or domain but generally refers to the keys_sources_*source_keys*.m routine with the same name run at many stages of the algorithm. Ex: 'MED', 'test',…
- **update**: an integer variable which offers the possibility to complete or to follow an existing AMEDA analysis. *update* takes the value 0 if you start an analysis from the beginning. Else, *update* takes the value of the missing or new steps. Ex: *update*=1, if you have 1 new step of new AVISO input data; *update*=365, if you have an entire year of daily input. You can also use update in the routine *mod_eddy_tracking.m*. **Warning**: Note that the starting step (*step0*) has to be exactly the same in the initial input file than in the updated input file, when you use *update*>0!
- **stepF**: an optional integer variable which serves to make a test of few steps. It forces the analysis to stop atfter *stepF* time steps. Leave by default commented for a full analysis of the input file.
- **deg**: an optional integer variable to degradate the input field by subsampling it, one grid point by every *deg* grid spacing. The degraded field becomes the native grid during the **mod_eddy_params.m** computation. This variable was originally used for testing the sensibility of AMEDA to resolution. It can be utilized by the user in order to increase the computation speed. In other case, it should be commented or removed, as *deg* will be set to 1 by default.
- **cpus**: an integer variable which defines the number of CPUs used by the Matlab **Parallel Computing Toolbox**.

- **list**: an array of the starting steps in increasing order in case of *MAIN_AMEDA_*_multi.m* to properly read from the input file the consecutive time steps and to record the analysis in intermediate output files. **Warning**: pay attention to specifically name the output results saved, in order to properly refer to that **list** and thus prevent *MAIN_AMEDA_multi*.m* from saving outputs always in the same file! For example, in case of yearly saved files, names them with a postfix including the year of the analysis.

- **name**: a string used as an input for *mod_eddy_tracks.m* which refers to the potential postname of *eddy_centers*.mat* and *eddy_shapes*.mat* because *mod_eddy_tracks.m* is applied to the final output files of the first three stages of AMEDA. This is relevant for a *multi* analysis which produces many output files. Thus *name* should refer to the result of the concatenation of all the output files built during the analysis of every step of the *list*. Ex: *name*='2001_2010' after the concatenation of yearly output files produced during a multi years analysis.

You can run some part of your *MAIN_AMEDA*.m* in a Matlab prompt, stage after stage, especially when you are setting it. To do that, use a small *stepF* and check the result in your *path_out*. You can also launch the full script all at once using a detached shell job if you are sure about your *MAIN_AMEDA*.m* setup and parametrization.

4. Set keys_sources

keys_sources.m* is the other user routine interface where you set your paths and your option keys belonging to the configuration. It is called at the beginning of the algorithm to describe the configuration and to add the results directory, *path_out*, to the Matlab path. Thus, this routine is also called once every time you want to load the results of AMEDA on the configuration, for plotting for example. Make sure to name **keys_sources_*source_keys*.m** with exactly the same names as the variables *source* and *keys* set in your *MAIN_AMEDA*.m*.

It would be unnecessarily complicated to explain in detail every line of the AMEDA routine. Instead please refer to the the list of parameters below (Tables 2) to setup your parameters according to the desired configuration.

Table 2a: Paths, files and names definitions in *keys_sources*.m*

Paths	Definition	Usage or value	Remarcks
source	Source of the input	Used to call proper load_fields_source.m file when reading input files	This name is the same that the one in MAIN_AMEDA*.m
config	Configuration	Used to define the path and the input file name	This name is different that *keys* or *domain* variable in MAIN_AMEDA*.m
sshtype	Type of ssh (adt, sla, model,...)	Used to distinguish between different source of ssh	These names have been settled during various tests but are optional and can be squeezed or modify. The idea is to make your *keys_sources* the more fexible to rename your own paths and input files
runname	Sensibility test name	Used to launch test runs at the same time	
postname	Specific analysis	Used to read a part in space or in time of a cropped common input file	
path_in	Input files directory	Used to open input files	Use previous names to build it
path_result	Ouput files common directory	Used to define the path directory including many path_out directories	Optional path convenient duriing test analysis. Keep it or not, belonging to your usage
path_out	Output files specific directory	Used to save output files and the parameters file param_eddy_tracking.mat	Important path which is added to the Matlab path to allow for a global call of param_eddy_tracking during AMEDA stages
path_tracks	Satelite tracks directory	Used to open satellite tracks	Useful in particular to plot results with satellite tracks
path_data	In situ data directory	Used to open other data	Useful to eventually plot results with drifter or ARGO positions
path_rossby	Rossby radius directory	Used to open the First Baroclinic Rossby Radius of Deformation file of your domain	The Rd file is an input file existing in the ocde but can be precised by your own computation
nc_dim	Input x,y,mask field file absolute name	Netcdf input file with grid data I x J	I is dimension along x J is dimension along y L is dimension along the time
nc_u, nc_v	Input u and v fields files absolute name	Netcdf input file with *u* and *v* data I x J x L	
nc_ssh	Input ssh field file absolute name	Netcdf input file with *ssh* data I x J x L	
x, y, m_name, u, v, s_name	Fields name in *nc_u, _grid, _v, _ssh*	Corresponding field names in netcdf input files	Can be adapted to the standard name in your files
mat_Rd	Rossby radius file absolute name	global_Rossby_radius.mat (global) / Rossby_radius.mat (Med sea)	You might create your own Rd matrix by modifying *read_global_Rossby_radius.m*
name_Rd	Rossby radius name in *mat_Rd*	Rd_Chelton (global) / Rd_baroc1_extra (Med sea)	longitude and latitude 2D matrixes must be named *lon_Rd* and *lat_Rd*

Table 2b: Parameters definition and keys activation in *keys_sources*.m*

parameters	Definition	Value	Usage	Remarcks
Rd_typ	Rossby radius typical value by default	12 km (Med sea)	Used to limit *Rd* to 1/3 of *Rd_typ*	Default valur for AVISO Med sea
nRmin	Minimal size for *Rmax*	0.5 Dx	Minimal limit radius of the recorded eddies	Default value for AVISO
T	Period for the fluid turnover	24*3600 seconds	Used to compute the Coriolis parameter for bench experiments	Dafault value for earth ocean
dps	Turnover period per step	1 period /step	Used during the tracking and the time filtering to convert steps to time	Daily time step by default dps = 1/8 gives 3h time step
level	Vertical level from 3D field	1	Used only when the input file contains more than 1 level	If >1, be care of the exact level in your input netcdf file
grid_ll	Grid type	1	0: cartesian coordinates in km (x,y) 1: earth coordinates in deg (lon,lat)	If 0, use boundaries extend as in *load_fields_PIV.m*
grid_reg	Regular grid or not	1	0: grid is not regular 1: regular grid	If 0, use regridding as in *load_fields_NEMO.m*
type_detection	Type of streamlines for *Vmax*	1	1: from velocity fields 2: from *ssh* field 3: use both	If 3, compute both detection and keep the higher *Vmax*
extended_diags	Compute xtra diagnostics	1	0: not saved 1: save Vort, Ke,...	You can compute these extra diags in post-processing
streamlines	Compute *V-R* profiles	1	0: not saved 1: save {profil2(n)} in *shapes.mat*	Need *Curve Fitting Toolbox*
daystreamfunction	Steps of *V-R*	1:stepF	Time steps recorded in code parts with *streamlines* key	The code accepts only all the steps of the serie
periodic	Periodic fields along x	0	0: boundary exists along x 1: no boundary	If 1, repeat field as in *load_fields_ROMS.m*
nrt	Near Real Time mode	1	0: not activated 1: no filter od the first and last eddies	If 1, no time filtration of eddies starting at the first step and finishing at the last step

5. Launch AMEDA

You can run the MAIN_AMEDA*.m routine all at once or partially by means of a Matlab interface. Otherwise, you could launch the script by detaching it from your connection while printing the output log in a specific file, if necessary. To run your *MAIN_AMEDA*.m* in a such detached job manner, use for example the following command in a terminal prompt or in a shell script. You need to run the command from the directory where your routine *start.m* setting your AMEDA paths is placed:

```
~/MATLAB$nohup matlab -nodesktop -nodisplay < ./AMEDA/MAIN_AMEDA_*.m > job_out.txt &
```

Browse AMEDA results

1. Variables names description

Once AMEDA has finished running all the stages (computing fields, detecting centers and shapes, then tracking eddies and merging and splitting events), it is possible to open the output files written in *path_out*. In *path_out* you will find the following files:

- *param_eddy_tracking.mat* contains all the parameters and paths of the configuration (cf. Tables 1 and 2). Load it if you want to check the value of any parameter used during the run. If you need only the paths (*path_in*, *path_out* or *path_data*), you can just run the *keys_sources*.m.*

- *fields.mat* contains the Matlab structure **{detection_fields(t)}** in *stepF* dimensions which record the various 2D fields computed for every step in *mod_fields.m* on the native grid or the degraded grid. These 2D fields are the LNAM, the kinetic energy, the divergence, the vorticity, the Okubo-Weiss parameter and the LOW (ie. Local averaged Okubo-Weiss parameter using the same number of grid points than for LNAM).

- *fields_inter.mat* contains the Matlab structure **{detection_fields(t)}** at *stepF* dimensions which records the same fields as in *fields.mat* but computed on a interpolated grid at a higher resolution by the factor *resol* if *resol* >1, otherwise they are the same than in *fields.mat*.

- *eddy_centers.mat* contains the structure **{centers2(t)}** at *stepF* dimensions which records the 'Nt' eddy center positions validated by their contours at every step *t* in *mod_eddy_shapes.m* (see table 3 for the detailed list of recorded variables). It contains also the structures **{centers(t)}** recording the potential eddies center positions computed in *mod_eddy_centres.m* and the intermediate **{centers0(t)}** which records the LNAM extrema. **Warning:** {centers0} has a higher number of recorded centers than {centers}; and this last one has also a higher number of recorded centres than {centers2}. This is due to the fact that the filtering criteria increases during the detection process.

- *eddy_shapes.mat* contains the structure **{shapes1(t)}** at *stepF* dimensions which records the N_t eddies contours detected and their associated feature variables for every step *t* (cf. Table 3a for the detailed list of recorded variables). It also contains **{shapes2(t)}** recording the N_t shapes of dual eddies and their associated feature variables at every step *t* (cf. Table 3b and also figure 5 to distinguish between the different contours). If the key *streamlines* is activated, *eddy_shapes.mat* may contain **{profil2(t)}** which records the features (mean velocity and equivalent radius) of the entire series of the streamlines scanned and the curve fitting parameters per eddy at every step *t* (cf. Table 3c). *eddy_shapes.mat* contains also **{warn_shapes(t)}** and **{warn_shapes2(t)}** which records flags and other parameters. {warn_shapes2(t)} is used to qualify the contours during the *mod_eddy_shapes.m* process (cf. Table 3d). **Warning**: Every feature variable has the same dimension (ie. N_t) as {shapes1(t)} variable at every *t*, but it can be filled with some NaN values if no dual eddy has been detected for a given detection n_t at *t*.

- *log_eddy_tracks.txt* keeps tracks of the log during the print out of the tracking process.
- *eddy_tracks.mat* contains the structure **{tracks(n)}** at N dimensions (the number of total tracked eddies) which records results of *mod_eddy_tracking.m* for all the time steps for every eddy track *n*. This structure reorganises the results of (cf. Tables 3 for the detailed list of recorded variables). *eddy_tracks.mat* contains also **{warn_tracks(t)}** used to debug the code but are not detailed here. **Warning**: *tracks(n)* has different variable names than *centers* and *shapes* which actually depict the same variable but are organised differently. You can use the tables 3 to found the connection.
- *log_eddy_merging_splitting.txt* keeps tracks of the log during the print out of the merging and splitting flagging and time filtering process.
- *eddy_tracks2.mat* contains the structure **{tracks2(n)}** at N' dimensions (the number of total tracked eddies after the time filtering includes in *mod_merging_splitting.m*) which records eddy variables at all the time steps for every eddy track *n'*. **{tracks2(n)}** records the same variables as **{tracks(n)}** with additional flags depicting splitting and merging events (cf. Table 3d). *eddy_tracks2.mat* contains also **{warn_tracks2(t)}** used to debug the code but are not detailled here. **Warning**: The index of the eddy *n'* in *tracks2* is not the same as *n* in *tracks* due to the filtering.

To read the output files you first need to run the routine keys_sources_*source_keys*.m associated with your configuration by setting the names 'keys' and 'sources' used in the name file routine.

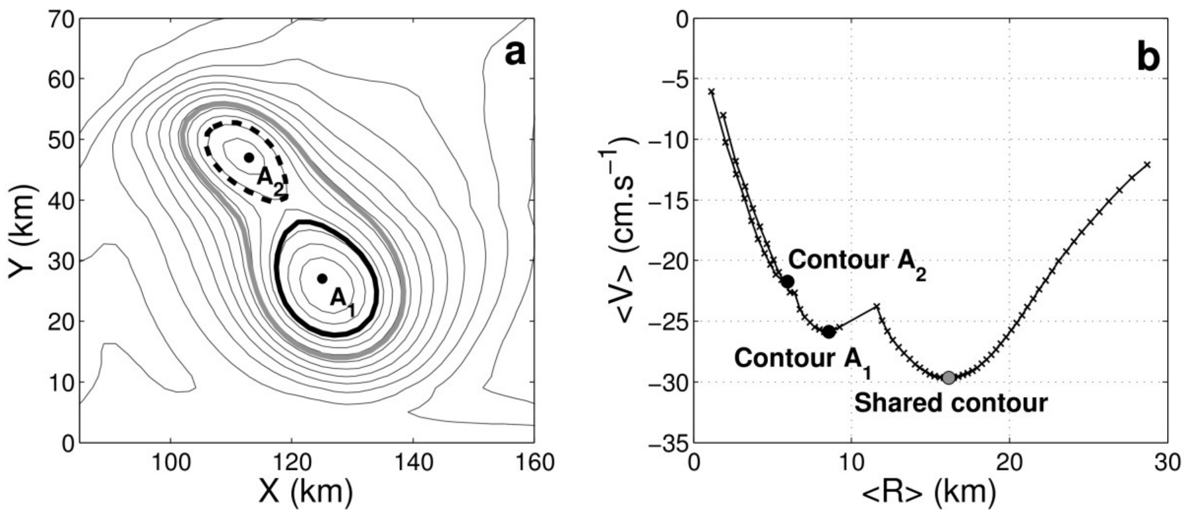

Figure 5: a- An illustration of the contours called characteristic contours (black lines) around the main eddy center (A_1) and the second center (A_2) of a dual eddy. The main eddy corresponds to a true maximum (continuous line), while the second eddy to a largest contour (dashed line). These two eddies also share a common contour (grey line). b- The <V>-<R> profile illustrates that local maxima of the curve corresponds to equivalent *radii* (rmax) and the velocities (vmax) of the characteristic contours. It also validates that the shared contour indeed represents a dual eddy because of its highest velocity.

Table 3a: Variables equivalence between {centers}, {shapes} and {tracks} for a main center

Variables definition associated to a main eddy *i* or an eddy track *n* at step *t*	**Units**	**Variables names**		
		Structures in eddy_centers.mat and eddy_shapes.mat		**Structures in eddy_tracks.mat or eddy_tracks2.mat**
		centers2(t)	**shapes1(t)**	**tracks(n) or tracks2(n)**
Time step	-	step	step	step
Eddy type (Anticyclone/Cyclone)	-1/1	type	-	type
LNAM extrema position	-	x1, y1	-	x1, y1
Coordinates serie of the characteristic contour*	-	-	[xy]	shapes1
Barycenter position of the characteristic contour	-	-	xbary ybary	xbary1 ybary1
Mean azimuthal velocity averaged along the characteristic contours	m.s^{-1}	-	velmax	velmax1
Equivalent radius of a circle with the same surface as the characteristic contour	km	-	rmax	rmax1
Variation of the *ssh* or *psi* between the characteristic contour and the extremum inside this contour	m	-	deta	deta1
Surface area delilmited by the characteristic contour	km^2	-	aire	aire1
Shortest turnover time around the main center	days	-	taumin	tau1
Part of the characteristic contour with negative curvature	%	-	nrho	nrho1
Ellipticity of an ellipse fitted on the characteristic contour	|1-b/a|	-	ellip	ellip1
Angle of the semi-major axis of the ellipse with the x-axis	radians	-	theta	theta1
Coordinate series of the last contour**		-	xy_end	shapes3
Mean azimuthal velocity averaged along the last contour	m.s^{-1}	-	vel_end	velmax3
Equivalent radius of a circle with the same surface as the last contour	km	-	r_end	rmax3
Variation of the *ssh* between the last contour and the extremum inside this contour	m	-	deta_end	deta3
Surface area delimited by the last contour	km^2	-	aire_end	aire3

*: The characteristic contour corresponds to the closed streamline including only the main center with the highest value of mean azimuthal velocity.

**: The largest contour corresponds to the most outward closed streamline including only the main eddy center.

Table 3b: Variables equivalence between {centers}, {shapes} and {tracks} for a dual eddy

Variables definition associated to an eventual dual eddy *i* or an eventual eddy track *n* at step *t*	Units	Variables names		
		Structures in eddy_centers.mat and eddy_shapes.mat		Structures in eddy_tracks.mat or eddy_tracks2.mat
		centers2(t)	shapes2(t)	tracks(n) or tracks2(n)
Time step	-	step	step	step
LNAM extrema position of the second center	-	x2, y2	-	x2, y2
Indice of the second LNAM center in {centers2(t)}	-	ind2	-	ind2
Distance from the main LNAM center	-	dc	-	dc
Coordinate series of the shared contour*		-	[xy]	shapes2
Barycenter of the shared contour	-	-	xbary ybary	xbary2 ybary2
Mean azimuthal velocity averaged along the shared contour	$m.s^{-1}$	-	velmax	velmax2
Equivalent radius of a circle with the same surface as the shared contour	km	-	rmax	rmax2
Variation of the *ssh* or *psi* between the shared contour and the extremum inside this contour	m	-	deta	deta2
Surface area delimited by the shared contour	km^2	-	aire	aire2
Shortest turnover time around the dual eddy	days	-	taumin	tau2
Part of the last contour with negative curvature	%	-	nrho	nrho2

*: The shared contour corresponds to the contour including two centers with the highest value of mean azimuthal velocity. This velocity must be higher than at least one characteristic contour to be considered as a dual eddy.

Table 3c: Variables equivalence between {shapes}, {profil} and {tracks} with *streamlines* key

Additional variables associated to all contours around a main eddy *I* or an eddy track *n* at step *t* The *streamlines* key is activated	Units	Variables names		
		Structures in n eddy_shapes.mat		Structures in eddy_tracks.mat or eddy_tracks2.mat
		shapes1(t)	profil2(t)	tracks(n) or tracks2(n)
Series of *ssh* or *psi* values	m	-	eta	-
Series of equivalent *radii serie*	km	-	rmoy	-
Series of mean azimuthal velocities	$m.s^{-1}$	-	vel	-
Series of one turnover times	days	-	tau	-
Series of part of negative curvature	%	-	nrhoi	-
Series of the steepness parameter*	-	*alpha*	myfit.curve.a	*alpha*
Series of the coefficient of determination**		rsquare	myfit.err.rsquare	rsquare
Series of the root mean square error**		rmse	myfit.err.rmse	rmse

*: steepness parameter associated to the generic function <V> = <R>.exp((1-(<R>)^{alpha}) / *alpha*) used to fit the vel-rmoy profile, with <V> = vel / velmax and <R> = rmoy / rmax.
**: R^2 and RMSE associated to the fitting of the generic fonction above with the vel-rmoy profile.

Table 3d: Variables equivalence between {shapes}, {warn_shapes} and {tracks}

<table>
<tr><th colspan="2" rowspan="3">Diagnostics associated to a main eddy i or an eddy tracks n at step t</th><th rowspan="3">Units</th><th colspan="3">Variables names</th></tr>
<tr><th colspan="2">Structures in n eddy_shapes.mat</th><th>Structures in eddy_tracks.mat or eddy_tracks2.mat</th></tr>
<tr><th>shapes1(t)</th><th>warn_shapes2(t)</th><th>tracks(n) or tracks2(n)</th></tr>
<tr><td colspan="2">Coriolis parameter at the main center</td><td>s^{-1}</td><td>-</td><td>f</td><td>f</td></tr>
<tr><td colspan="2">Deformation radius at the main center</td><td>km</td><td>-</td><td>Rd</td><td>Rd</td></tr>
<tr><td colspan="2">Resolution of Rd</td><td>-</td><td>-</td><td>gama</td><td>gama</td></tr>
<tr><td colspan="2">Is psi used instead of ssh?</td><td>0/1</td><td>-</td><td>calcul_curve</td><td>calcul</td></tr>
<tr><td colspan="2">Is the main eddy the largest contour?</td><td>0/1</td><td>-</td><td>large_curve1</td><td>large1</td></tr>
<tr><td colspan="2">Is the second eddy the largest contour?</td><td>0/1</td><td>-</td><td>large_curve2</td><td>large2</td></tr>
<tr><td colspan="2">Is the shared contour velocity too weak?</td><td>0/1</td><td>-</td><td>too_weak2</td><td>weak</td></tr>
<tr><td colspan="2">Main eddy index i in shapes1(t)</td><td>-</td><td>-</td><td>ind</td><td>ind</td></tr>
<tr><td colspan="2">Second eddy tracks indices n' interacting with eddy tracks n</td><td>-</td><td>-</td><td>interaction, interaction2</td><td>interaction, interaction2*</td></tr>
<tr><td colspan="2">Is the eddy track n coming from a splitting with eddy tracks n'?</td><td>0/1</td><td>-</td><td>-</td><td>split, split2**</td></tr>
<tr><td colspan="2">Is the eddy track n' ending due to splitting with eddy tracks n'?</td><td>0/1</td><td>-</td><td>-</td><td>merge, merge2**</td></tr>
<tr><td rowspan="5">extended_diags key activated</td><td>Sum of the kinetic energy inside the characteristic contour</td><td>$m^2.s^{-2}$</td><td>ke</td><td>-</td><td>ke1</td></tr>
<tr><td>Mean value of vorticity inside the characteristic contour</td><td>s^{-1}</td><td>vort</td><td>-</td><td>vort1</td></tr>
<tr><td>Maximal value of vorticity inside the characteristic contour</td><td>s^{-1}</td><td>vortM</td><td>-</td><td>vortM1</td></tr>
<tr><td>Mean value of the Okubo Weiss parameter inside the characteristic contour</td><td>s^{-2}</td><td>OW</td><td>-</td><td>OW1</td></tr>
<tr><td>Mean value of LNAM inside the characteristic contour</td><td>-</td><td>LNAM</td><td>-</td><td>LNAM1</td></tr>
</table>

*: *interaction2* records very unusual second interactions happening the same time as *interaction.*
**: *split* and *merge* are flags existing only in *tracks2(n) and come* from *mod_merging_splitting.m;* split2 and merge2 record splitting and merging from *interaction2,* if any.

2. Plotting examples

It is possible to use plotting routines provided with the code **but you need ouput files from AMEDA**. There are two kinds of such routines:

- Routines used to plot the outputs from the file *eddy_shapes.mat* when the goal is to visualize all the eddies detected in a particular day. These routines are called *plot_shapes*.m* (cf. Example 1).

- To plot the <V>-<R> profiles of a specific eddy, please refer to the example 2 script which uses the Matlab structure *profil2* recorded in *eddy_shapes.mat* (cf. Example 2)
- In order to visualize the tracking of a particular eddy, it is possible to use the routines utilized to plot the outputs from the file *eddy_tracks.mat* or *eddy_tracks2.mat.* These routines are called *plot_tracking*.m* (cf. Example 3).
- *tracks.mat* are also handly for plotting time series of eddy features (cf. Example 4).

Example 1: Eddies and dual eddies visualisation on a velocities field for a given time step *day* (see result on figure 6)

```
% define configuration source and keys as well as the time
start, clear; clc;
source = 'AVISO';
config = 'MED_adt';
day = 164; % time step
dayi=[2019 05 01 12 0 0];% from data file
dr = datevec(datenum(dayi)+day-1);

% load parameters, centers and shapes results
run(['keys_sources_',source,'_',config])
load('param_eddy_tracking');
load([path_out,'eddy_centers']); load([path_out,'eddy_shapes'])
CD21 = [centers2(day).x1;centers2(day).y1]; % LNAM main centers

% set boundaries and plot
minlat=36.5; maxlat=40.5; minlon=3.5; maxlon=9;
m_proj('Mercator','lat',[minlat maxlat],'lon',[minlon maxlon]);

% define a plot window and the map
close all
hfig=figure('visible','off');
set(hfig,'Position',[0 0 800 800])

% load grid and fields at the step day
eval(['[x,y,mask,ssu,ssv,~] = load_fields_',source,'(day,1,deg);'])

% plot quiver velocities
xnan=x; ynan=y; vel=sqrt(ssu.^2+ssv.^2);
xnan(isnan(vel) | vel<0.05) = nan; ynan(isnan(vel) | vel<0.05) = nan;
m_quiver(xnan,ynan,ssu,ssv,2,'color',[0.3 0.3 0.3])

% grid and fancy the map
m_coast('patch',[.9 .9 .9]); m_grid('tickdir','in','linewidth',1,'linestyle','none');
set(gcf,'color','w'); % set figure background to white

hold on
% plot shapes of the contours
for i=1:length(shapes1(day).xy)
  if CD21(1,i)<maxlon && CD21(1,i)>minlon && CD21(2,i)>minlat && CD21(2,i)<maxlat
    lonlat1=shapes1(day).xy{i}; lonlat2=shapes2(day).xy{i}; lonlat3=shapes1(day).xy_end{i};
    type=centers2(day).type(i);
    if type==-1,
      col=[.1 .1 .9]; % anticyclone
    else
```

```
      col=[.9 .1 .1]; % cyclone
    end
    if ~isempty(lonlat3)
      hl(3) = m_plot(lonlat3(1,:),lonlat3(2,:),'--k','linewidth',1.5); % last contour
    end
    if ~isnan(lonlat1)
        if type==-1
          hl(1) = m_plot(lonlat1(1,:),lonlat1(2,:),'color',col,'linewidth',2); % characteristic contour
        else
          hl(2) = m_plot(lonlat1(1,:),lonlat1(2,:),'color',col,'linewidth',2); % characteristic contour
        end
    end
    if ~isnan(lonlat2)
      hl(4) = m_plot(lonlat2(1,:),lonlat2(2,:),'color',[.1 .9 .1],'linewidth',2); % shared contour
    end
  end
end

% plot centers and indicate eddy indices
ind=find(CD21(1,:)<maxlon & CD21(1,:)>minlon & CD21(2,:)>minlat & CD21(2,:)<maxlat);
m_plot(CD21(1,ind),CD21(2,ind),'ok','MarkerFaceColor','k','MarkerSize',4)
for i=1:length(CD21)
  if CD21(1,i)<maxlon && CD21(1,i)>minlon && CD21(2,i)>minlat && CD21(2,i)<maxlat
    m_text(CD21(1,i),CD21(2,i),['  ',num2str(i)],'color',[0 0 0],'FontSize',10,'fontWeight','bold')
  end
end
hold off

% add legend on the map and information
title('AMEDA on geostrophic velocities from AVISO','Fontsize',14)
m_text(minlon+0.2,minlat+0.2,[datestr(dr,1)],'FontSize',10,'fontWeight','bold')
legend(hl,{'characteristic contour (AE)','characteristic contour (CE)','last contour','shared contour'})
```

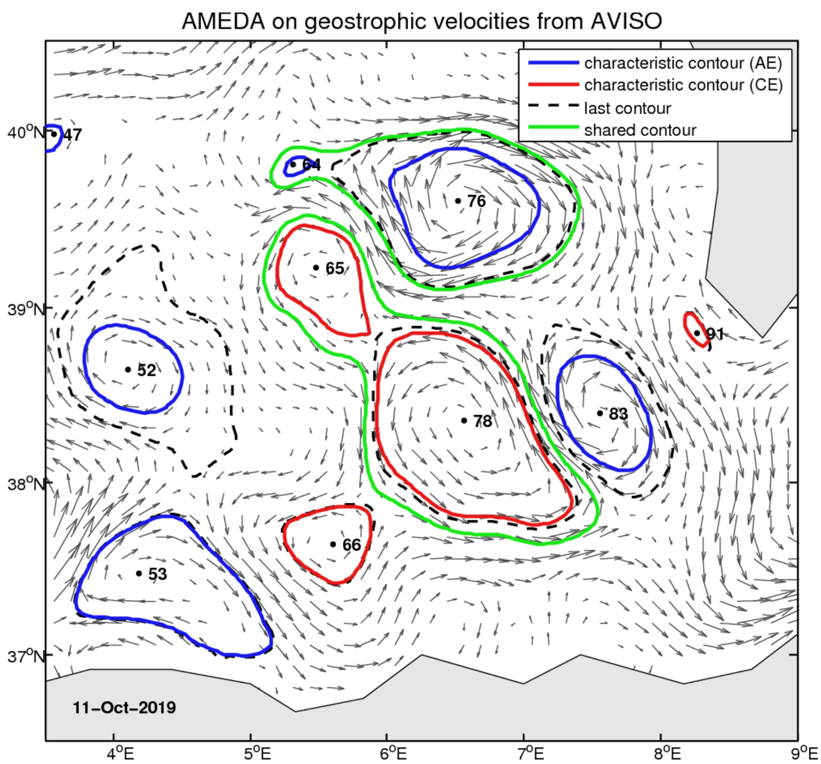

Figure 6: Example of AMEDA contours from AVISO (Ssalto/Duacs satelite imagery) using results in *eddy_centers.mat* and *eddy_shapes.mat*; note their index *i* numbers close to centers.

<u>Example 2</u>: Plot <V>-<R> profile of a dual eddy (see result on figure 7). We used result from the example 1 to plot the <V>-<R> profile of streamline around the dual eddy 78-65 (cf. Figure 6).

```
% define configuration and the time
start, clear; clc;
source = 'AVISO'; config = 'MED_adt';
day = 164; % time step
dayi=[2019 05 01 12 0 0];% from data file
dr = datevec(datenum(dayi)+day-1);
i=78; j=65; % eddy indices

% load parameters and shapes results
run(['keys_sources_',source,'_',config])
load('param_eddy_tracking'); load([path_out,'eddy_shapes'])

% open a plot window
clear hl, close all
hfig=figure('visible','off');
set(hfig,'Position',[0 0 600 400])
hold on

% plot the V-R profil of the dual eddy center 78 with 65
hl(1) = plot(profil2(day).rmoy{i},profil2(day).vel{i},'.');
hl(2) = plot(profil2(day).rmoy{j},profil2(day).vel{j},'.k');
hl(3) = plot(shapes1(day).rmax(i),shapes1(day).velmax(i),'or','markerfacecolor','r');
plot(shapes1(day).rmax(j),shapes1(day).velmax(j),'or','markerfacecolor','r');
hl(4) = plot(shapes2(day).rmax(j),shapes2(day).velmax(j),'og','markerfacecolor','g');

% add legend on the map and information
grid on, box on
axis([0 100 0 .4])
ylabel('<V> (m.s^-^1)','Fontsize',12), xlabel('<R> (km)','Fontsize',12)
title(['<V>-<R> profiles from streamlines around eddies ',num2str(i),' and ',num2str(j)],'Fontsize',14)
legend(hl,{['Profile from eddy ',num2str(i)],['Profile from eddy ',num2str(j)],…
          ['V_m_a_x and R_m_a_x for main centers'],…
          'V_m_a_x and R_m_a_x for shared contour'},'Location','Southeast')
```

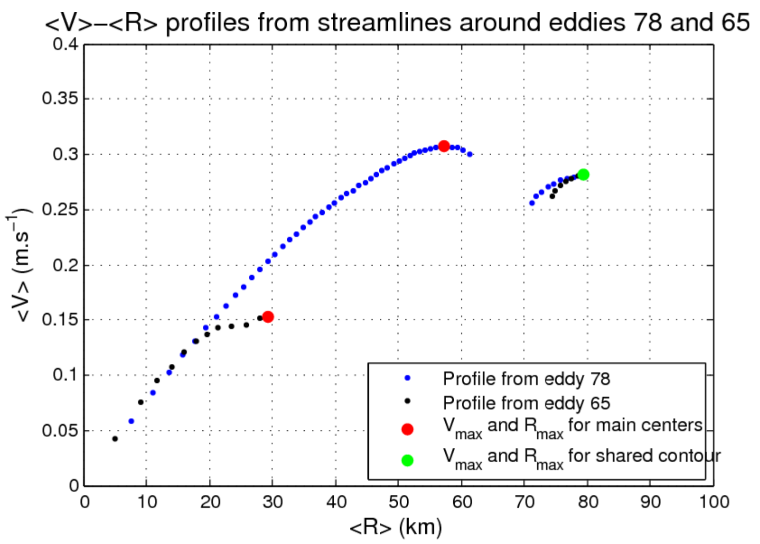

Figure 7: Example of AMEDA <V>-<R> profiles of the streamlines surrounding eddies 78 and 65 the 11th october 2019 from AVISO using results in *eddy_shapes.mat*.

Example 3: Eddy tracking visualisation at a given time step *day* (see result on figure 8).

```
% define configuration and the time
start, clear; clc;
source = 'AVISO'; config = 'MED_adt';
day = 164; % time step
dayi=[2019 05 01 12 0 0];% from data file
dr = datevec(datenum(dayi)+day-1);

% load parameters and tracks results
run(['keys_sources_',source,'_',config])
load('param_eddy_tracking'); load([path_out,'eddy_tracks2'])

% set boundaries and plot
minlat=36.5; maxlat=40.5; minlon=3.5; maxlon=9;
m_proj('Mercator','lat',[minlat maxlat],'lon',[minlon maxlon]);

% define a plot window and the map
close all, clear hl
hfig=figure('visible','off');
set(hfig,'Position',[0 0 800 800])

% load grid and fields at the step day
eval(['[x,y,mask,ssu,ssv,~] = load_fields_',source,'(day,1,deg);'])

% plot quiver velocities
xnan=x; ynan=y; vel=sqrt(ssu.^2+ssv.^2);
xnan(isnan(vel) | vel<0.05) = nan; ynan(isnan(vel) | vel<0.05) = nan;
m_quiver(xnan,ynan,ssu,ssv,2,'color',[0.3 0.3 0.3])

% grid and fancy the map
m_coast('patch',[.9 .9 .9]); m_grid('tickdir','in','linewidth',1,'linestyle','none');
set(gcf,'color','w'); % set figure background to white

hold on
% add eddy tracking
tracks=tracks2;
for i=1:length(tracks)
   ind=[];
   ind=find(tracks(i).step==day,1);
   if ~isempty(ind)
      CD = [(tracks(i).xbary1)';(tracks(i).ybary1)'];
      dura = tracks(i).step(ind)-tracks(i).step(1)+1;
      hl(5) = m_plot(CD(1,1:ind),CD(2,1:ind),'-','color',[0.4 0.4 0.4],'linewidth',2);
      m_plot(CD(1,1),CD(2,1),'sk','MarkerFaceColor','k','MarkerSize',4)
      m_plot(CD(1,ind),CD(2,ind),'ok','MarkerFaceColor','k',...
         'MarkerSize',round(dura/60+4))
      % plot last contour
      lonlat3=tracks(i).shapes3{ind};
      hl(3) = m_plot(lonlat3(1,:),lonlat3(2,:),'--k','linewidth',1.5);
      % plot characteristic contour
      lonlat1=tracks(i).shapes1{ind};
      if ~isnan(lonlat1)
         if tracks(i).type(1)==-1
```

```
        col=[.1 .1 .9]; % anticyclone
        hl(1) = m_plot(lonlat1(1,:),lonlat1(2,:),'color',col,'linewidth',2);
      else
        col=[.9 .1 .1]; % cyclone
        hl(2) = m_plot(lonlat1(1,:),lonlat1(2,:),'color',col,'linewidth',2);
      end
    end
    % plot shared contour
     lonlat2=tracks(i).shapes2{ind};
     if ~isnan(lonlat2)
       hl(4) = m_plot(lonlat2(1,:),lonlat2(2,:),'-','color',[.1 .9 .1],'linewidth',2); % dual eddy
     end
     % indicate splitting or merging events and track indices
     if CD(1,ind)<maxlon && CD(1,ind)>minlon && CD(2,ind)>minlat && CD(2,ind)<maxlat
       if tracks(i).split(ind)==1
         m_text(CD(1,ind)+0.2,CD(2,ind),'split',...
           'color',[0 0 0],'FontSize',8,'fontWeight','bold')
       end
       if tracks(i).merge(ind)==1
         m_text(CD(1,ind)+0.2,CD(2,ind),'merge',...
           'color',[0 0 0],'FontSize',8,'fontWeight','bold')
       end
       m_text(CD(1,ind),CD(2,ind)+.1,[num2str(i)],...
         'color',[0 0 0],'FontSize',10,'fontWeight','bold')
      end
  end
end

% add legend on the map and information
title('AMEDA tracks on geostrophic velocities from AVISO','Fontsize',14)
m_text(minlon+0.2,minlat+0.2,[datestr(dr,1)],'FontSize',10,'fontWeight','bold')
legend(hl,{'Anticyclone','Cyclone','last contour','Interaction contour','Barycenter tracks'})
```

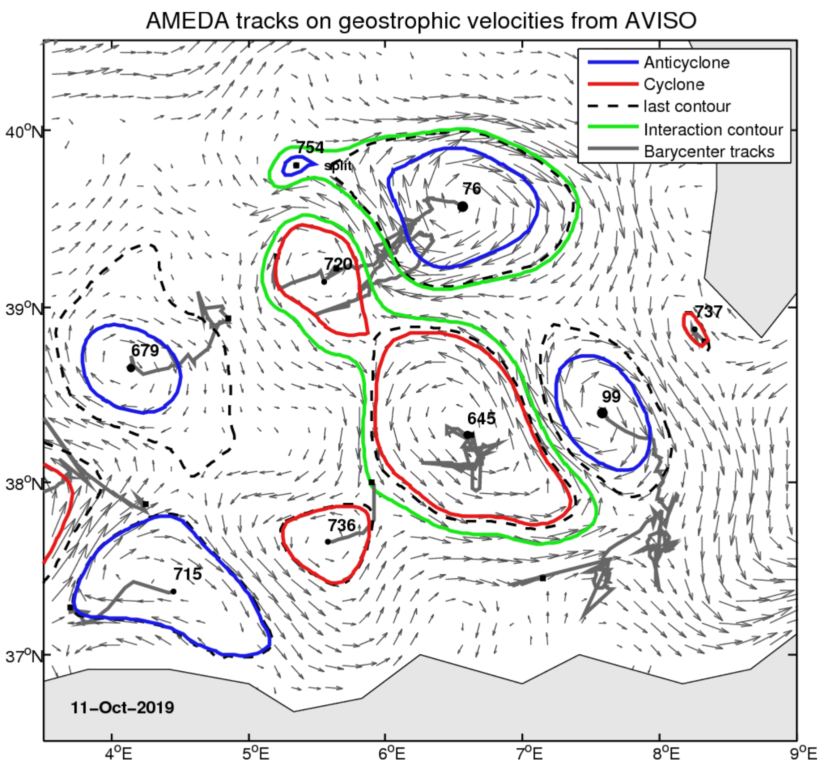

Figure 8: Example of AMEDA contours from AVISO (Ssalto/Duacs satelite imagery) using results in *eddy_centers.mat* and *eddy_shapes.mat*; note their index *i* numbers close to centers.

<u>Example 4</u>: Eddy time series visualisation (see figure 9). We used result from the example 3 to plot features time series of the eddies tracks 76 and 645 (cf. Figure 8). Note that eddy 76 is the only eddy with an indice in *tracks* equals to its indice in *shapes*.

```
% define configuration
start, clear; clc;
source = 'AVISO'; config = 'MED_adt';
dayi=[2019 05 01 12 0 0];% from data file
janp=246+datenum(dayi)-1;

% load parameters and tracks results
run(['keys_sources_',source,'_',config])
load('param_eddy_tracking'); load([path_out,'eddy_tracks2']), tracks=tracks2;

% longest eddies
n=76; m=645; % track indices
stepn = tracks(n).step+datenum(dayi)-1;
stepm = tracks(m).step+datenum(dayi)-1;

% plot features eddy tracks n and m
close all
hfig=figure('visible','off');
set(hfig,'Position',[0 0 1000 800])
set(gcf,'color','w'); % set figure background to white

% Rmax
subplot(2,2,1)
plot(stepn,tracks(n).rmax1,'color',[0 .2 .8],'linewidth',2)
ylim([0 80])
hold on, plot([janp janp],[0 80],'k','linewidth',1.5)
text(janp-5,5,'2019','HorizontalAlignment', 'right','Fontsize',12), text(janp+5,5,'2020','Fontsize',12)
datetick('x','dd/mm','keepticks')
box on, grid on
xlabel('date','FontSize',14), ylabel('R_m_a_x (km)','FontSize',12)
title(['Size Anticyclone Eddy ',num2str(n)],'FontSize',14)
subplot(2,2,2)
plot(stepm,tracks(m).rmax1,'color',[.8 .2 0],'linewidth',2)
ylim([0 80])
text(stepm(end)-5,5,'2019','HorizontalAlignment', 'right','Fontsize',12)
datetick('x','dd/mm','keepticks')
box on, grid on
xlabel('date','FontSize',14), ylabel('R_m_a_x (km)','FontSize',12)
title(['Size Cyclone Eddy ',num2str(m)],'FontSize',14)

% Velmax
subplot(2,2,3)
plot(stepn,tracks(n).velmax1,'color',[0 .2 .8],'linewidth',2)
ylim([0 .6])
hold on, plot([janp janp],[0 .6],'k','linewidth',1.5)
text(janp-5,.05,'2019','HorizontalAlignment', 'right','Fontsize',12), text(janp+5,.05,'2020','Fontsize',12)
datetick('x','dd/mm','keepticks')
box on, grid on
xlabel('date','FontSize',14), ylabel('V_m_a_x (m.s^-^1)','FontSize',12)
```

```
title(['Intensity Anticyclone Eddy ',num2str(n)],'FontSize',14)
subplot(2,2,4)
plot(stepm,tracks(m).velmax1,'color',[.8 .2 0],'linewidth',2)
ylim([0 .6])
text(stepm(end)-5,.05,'2019','HorizontalAlignment', 'right','Fontsize',12)
datetick('x','dd/mm','keepticks')
box on, grid on
xlabel('date','FontSize',14), ylabel('V_m_a_x (m.s^-^1)','FontSize',12)
title(['Intensity Cyclone Eddy ',num2str(m)],'FontSize',14)
```

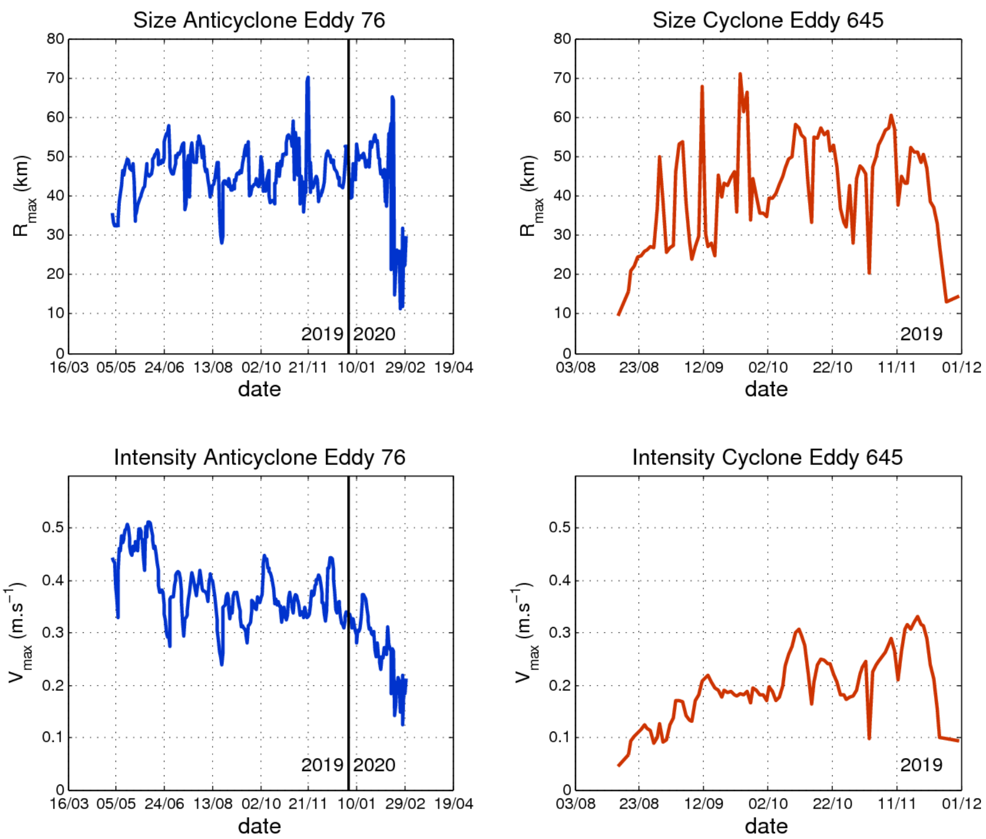

Figure 9: Time series of eddy tracking features computed by AMEDA and recorded in *eddy_tracks2.mat*. Examples are taken for AE tracks 76 and CE tracks 645 (see figure 8)

References

L. Cimoli, G. Roullet, A. Stegner « Meanders and eddies generation from a buoyant coastal current above a bathymetric slope. » *Ocean Science*, 13, 1–19, (2017). doi:10.5194/os-13-1-2017

Le Vu, B., A. Stegner, and T. Arsouze, 2018: *Angular Momentum Eddy Detection and Tracking Algorithm (AMEDA) and Its Application to Coastal Eddy Formation*. J. Atmos. Oceanic Technol., 35, 739–762, https://doi.org/10.1175/JTECH-D-17-0010.1

Mkhinini, N., A. L. S. Coimbra, A. Stegner, T. Arsouze, I. Taupier-Letage, and K. Béranger, 2014: *Long-lived mesoscale eddies in the eastern mediterranean sea: Analysis of 20 years of aviso geostrophic velocities*. Journal of Geophysical Research: Oceans, 119 (12), 8603–8626, doi: 10.1002/2014JC010176.

Nencioli, F., C. Dong, T. Dickey, L. Washburn, and J. C. McWilliams, 2010: *A vector geometry-based eddy detection algorithm and its application to a high-resolution numerical model product and high-frequency radar surface velocities in the southern california bight*. Journal of Atmospheric and Oceanic Technology, 27 (3), 564–579.

Pawlowicz, R., 2020: “*M_Map: A mapping package for MATLAB*", version 1.4m, [Computer software], available online at www.eoas.ubc.ca/~rich/map.html.