





## DOCUMENTATION

VERSION **-VO.1**

---

# Eddy Detection And Tracking

---

*Author:* 2015/08/03  
NIKOLAUS KOOPMANN



# *Contents*

<b>1</b>	<b>The Algorithm</b>	<b>5</b>
1.1	Data Preparation . . . . .	6
1.2	Rossby Radii and Phase Speeds . . . . .	6
1.3	Finding the Contours . . . . .	7
1.4	Filtering the Contours . . . . .	8
1.5	Tracking . . . . .	16
1.6	Post Processing 1 . . . . .	18
1.7	Post Processing 2 . . . . .	18
1.8	Running the Code . . . . .	18

<b>Bibliography</b>	<b>21</b>
---------------------	-----------



# 1

## *The Algorithm*

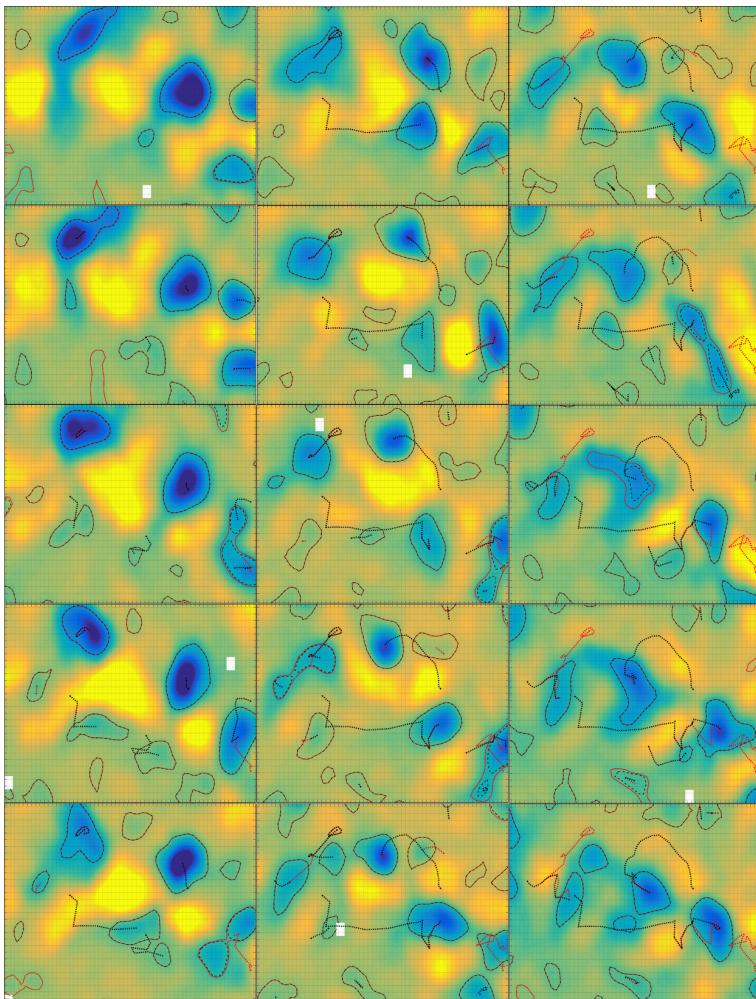


Figure 1.1: Fortnightly Aviso-MII snapshots (cyclones only) showing the area from  $47.6^{\circ}\text{S}$  to  $30.1^{\circ}\text{S}$  and  $40.1^{\circ}\text{E}$  to  $64.9^{\circ}\text{E}$ . Threshold age for saving is  $8 \times 7\text{days}$ . Red [black] color represents MI [MII]. Dashed lines are the contours and dotted lines are the tracks. Only *active* tracks are drawn. The general impression from animations of this sort is that the MI-method is good at tracking coherent, west-ward propagating, less-circular SSH anomalies while the MII-method seems superior at successfully tracking higher-amplitude vortices that get advected by mean currents (e.g. the strong cyclone in appr. the middle of the picture describing an anti-clockwise circular track due to advection by the ACC.)

THIS documentation walks through the algorithm step by step, so as to explain which methods are used and how they are implemented. The idea is that the code from step `s01..` on can only accept one particular structure of data. In earlier versions the approach was to write code that would adapt to different types of data automatically. All of this extra adaptivity turned

out to visually and structurally clog the code more than it offered much of a benefit. The concept was therefore reversed. Input SSH-data needs to be altered to required format (Steps S00...). Yet, there should be no need to adapt any of the later steps in any way. All input parameters are to be set in `INPUT.m` and `INPUTX.m`.

### 1.1 Step S00: Prepare Data

```
function S00a_prep_raw_data
```

BEFORE the actual eddy detection and tracking is performed, SSH-, latitude- and longitude grids are extracted from the given data at desired geo-coordinate bounds and saved as structures in the form needed by the next steps. This step also builds the file `window.mat` via `constructGeoFile.m` which saves geometric information about the input and output data as well as a cross-referencing index-matrix which is used to reshape all *cuts* to the user-defined geo-coordinate geometry. The code can handle coordinate input that crosses the longitudinal seam of the input data. E.g. say the input data came in matrices that start and end on some (not necessarily strictly meridional) line straight across the Pacific and it is the Pacific only that is to be analyzed for eddies, the output maps are stitched accordingly. In the zonally continuous case *i.e.* the full-longitude case, an *overlap* in x-direction across the *seam-meridian* of the chosen map is included so that contours across the seam can be detected and tracked across it. One effect is that eddies in proximity to the seam can get detected twice at both zonal ends of the maps. The surplus double-eddies get filtered out in `S04_track_eddies`.

### 1.2 Step S00b: Find Mean Rossby Radii and Phase Speeds

```
function S00b_rossbyStuff
```

THIS function...

- – ...calculates the pressure  $P(z, \phi)$  in order to...
- ...calculate the Brunt-Väisälä-Frequency according to  $N^2(S, T, P, \phi) = -\frac{g(\phi)}{P} \frac{\partial \rho(S, T, P)}{\partial z}$  in order to...
- – ...integrate the Rossby-Radius  $L_R^1 = \frac{1}{\pi f} \int_H N dz$  and ...
- apply the long-Rossby-Wave dispersion relation to found  $L_R^1$  to estimate Rossby-Wave phase-speeds  $c = -\frac{\beta}{k^2 + (1/L_r)^2} \approx -\beta L_R^{1/2}$

The 3-dimensional matrices ( $S$  and  $T$ ) are cut zonally into slices which then get distributed to the threads. This allows

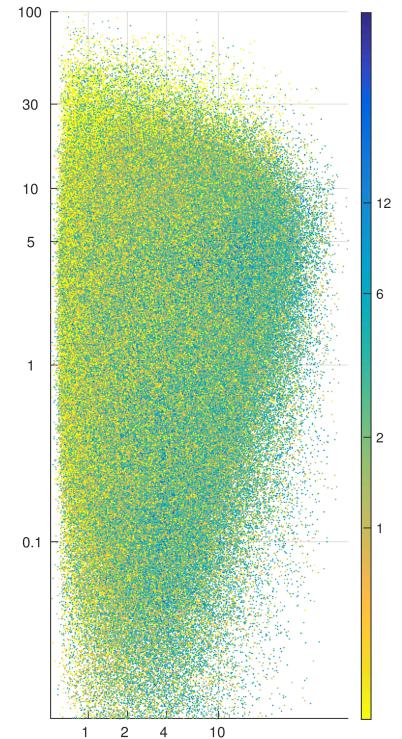


Figure 1.2: POP-7day-MII : Small amplitude correlates with a short life and a broad drift speed spectrum. y-axis: zonal drift speed [cm/s], x-axis: amplitude [cm], color: age [months].

for direct matrix operations for all calculations which would otherwise cause memory problems due to the immense sizes of the 3d-data<sup>1</sup>.

<sup>1</sup> E.g. the POP data has dimensions  $42 \times 3600 \times 1800$ .

### Step Sooc and Sood: Build SSH anomaly

```
function S00c_buildMeanSsh
function S00d_buildSshAnomaly
```

THIS step reads the cut SSH data from `s00_prep_data` to perform 2 steps:

1. Calculate a mean over time of  $SSH(y, x)$ .
2. Subtract the mean from each  $SSH(t)$  to filter out persistent SSH-gradients e.g. across the Gulf-Stream.

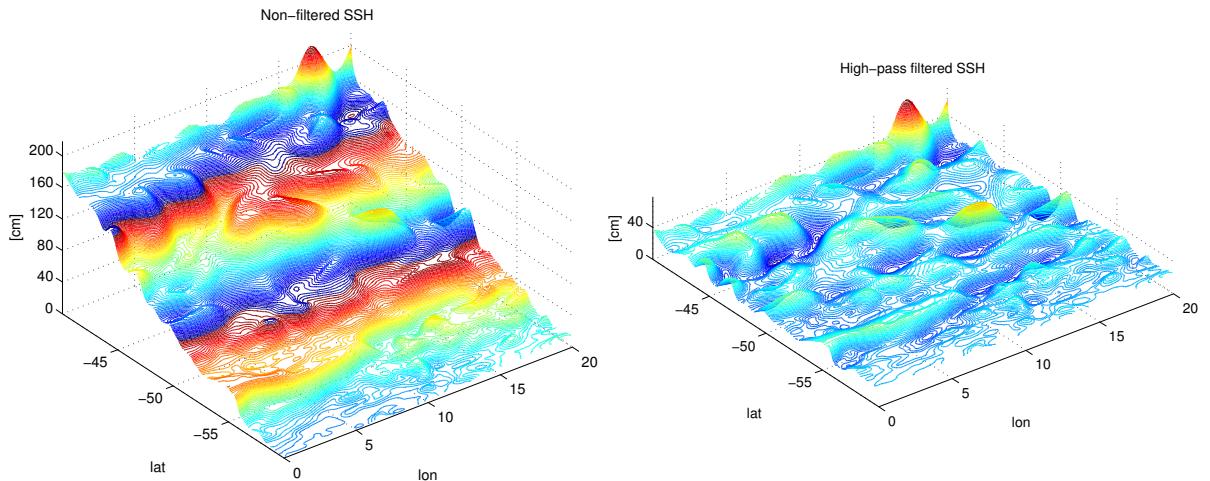


Figure 1.3: SSH with mean over time subtracted.

### 1.3 Step S02: Find Contours

```
function S02_contours
```

THE sole purpose of this step is to apply MATLAB's `contourc.m` function to the SSH data. It simply saves one file per time-step with all contour indices appended into one vector<sup>2</sup>. The contour intervals are determined by the user-defined increment and range from the minimum- to the maximum of given SSH data.

<sup>2</sup> see the MATLAB documentation.

The function `initialise.m`, which is called at the very beginning of every step, here has the purpose of rechecking the *cuts* for consistency and correcting the time-steps accordingly (i.e. when files are missing).

## 1.4 Step S03: Filter Contours

```
function S03_filterContours
```

SINCE indices of all possible contour lines at chosen levels are available at this point, it is now time to subject each and every contour to a myriad of tests to decide whether it qualifies for the outline of an eddy as defined by the user input threshold parameters.

### 1.4.1 Reshape for Filtering and Correct out of Bounds Values

```
function eddies2struct
function CleanEddies
```

In the first step the potential eddies are transformed to a more sensible format, that is, a structure `Eddies` of size `EddyCount`. The struct has fields for level, number of vertices, exact *i.e.* interpolated coordinates and rounded integer coordinates.

The interpolation of `contourc.m` sometimes creates indices that are either smaller than 0.5 or larger than<sup>3</sup>  $N + 0.5$  for contours that lie along a boundary. After rounding, this seldomly leads to indices of either 0 or  $N + 1$ . These values get set to 1 and  $N$  respectively in this step.

<sup>3</sup> where  $N$  is the domain size.

### 1.4.2 Descend/Ascend Water Column and Apply Tests

```
function walkThroughContsVertically
```

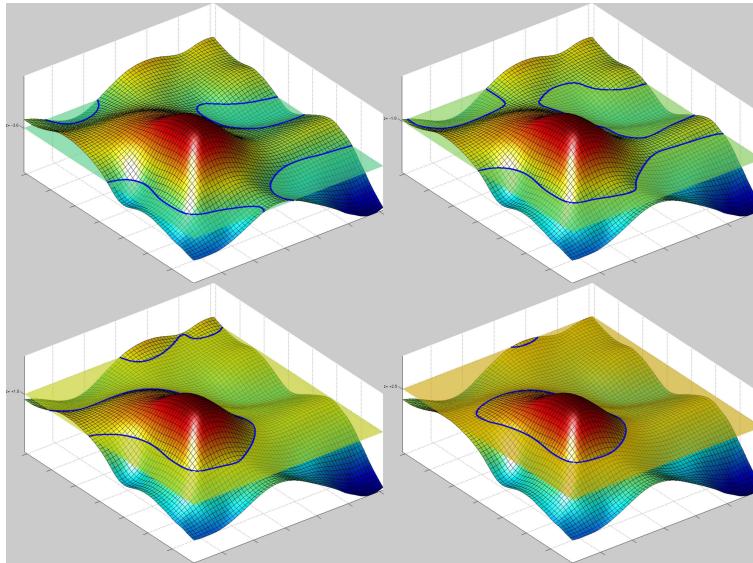


Figure 1.4: The algorithm approaches the appropriate vertical level incrementally.

The concept of this step is a direct adaptation of the algorithm described by Chelton *et al.* (2011). It is split into two steps, one for anti-cyclones and one for cyclones. Consider

e.g. the anti-cyclone situation: Since all geostrophic anti-cyclones are regions of relative high pressure, all ACs effect an elevated SSH *i.e.* a *hill*. The algorithm ascends the full range of SSH levels where contours were found. Consider next an approximately Gaussian shaped AC that has a peak SSH of say 5 increments larger than the average surrounding waters. As the algorithm approaches the sea surface from below, it will eventually run into contours that are closed onto themselves and that encompass the AC. At first these contours might be very large and describe not only one but several ACs and likely also cyclones, but as the algorithm continues upwards, found contour will get increasingly circular, describing some outer edge of the AC. Once the contour and its interior pass all of the tests, the algorithm will decide that an AC was found and write it and all its parameters to disk. The AC's region *i.e.* the interior of the contour will be flagged from here on. Hence any inner contour further up the water column will not pass the tests. Once all ACs are found for a given time-step, the SSH flags get reset and the entire procedure is repeated, only this time *descending* the SSH-range to find cyclones. The tests for cyclones and anti-cyclones are therefor identical except for a factor  $-1$  where applicable. In the following the most important steps of the analysis are outlined.

#### Contour filter 1 NaN-Check Contour

```
function CR_RimNan
```

The first and most efficient test is to check whether indices of the contour are already flagged. Contours within an already found eddy get thereby rejected immediately.

#### Contour filter 2 Closed Ring

```
function CR_ClosedRing
```

Contours that do not close onto themselves are obviously not eligible for further testing.

#### Contour filter 3 Sub-Window

```
function get_window_limits, EddyCut_init (see cutMaskOperation)
```

For further analysis a sub-domain around the eddy is cut out of the SSH data. These functions determine the indices of that window and subtract the resultant offset for the contour indices.

#### Contour filter 4 Logical Mask of Eddy Interior

```
function EddyCut_mask (see cutMaskOperation)
```

Basically this function creates a **flood-fill** logical mask of the eddy-interior. This is by far the most calculation-intensive part of the whole filtering procedure. A lot more time was wasted on attempting to solve this problem more efficiently than time could have been saved would said attempts have been successful. The current solution is basically just MATLAB's

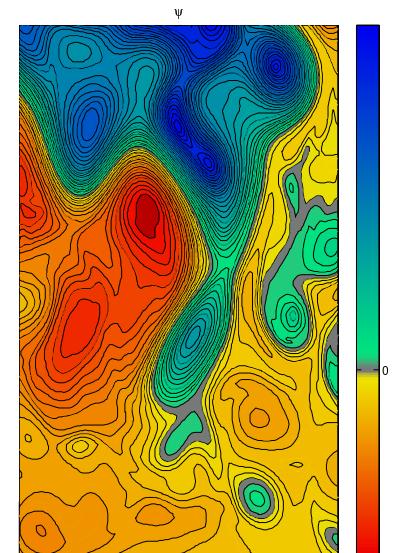


Figure 1.5: Stream function of a meandering jet shedding off a vortex. The line of strongest gradient *i.e.* fastest geostrophic speed later becomes the zero-vorticity-line at a theoretical distance  $\sigma$  from the center (Offset of  $\Psi$  is chosen arbitrarily).

`imfill.m`, which was also used in the very first version of 09/2013. EDIT: `imfill.m` was replaced by using `inpolygon.m` to determine which indices lie within the contour-polygon. This method seems to be more exact at determining whether the inside-part of one grid cell (with respect to the smooth, spline-interpolated contour) is larger than the outside part or not.

### Contour filter 5 Sense

```
function CR_sense
```

All of the interior SSH values must lie either above or below current contour level, depending on whether anti-cyclones or cyclones are sought.

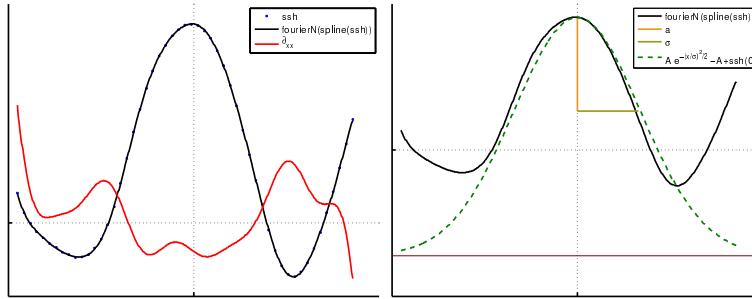


Figure 1.6: Left: Fourier-fit of an eddy from POP SSH-data and the 2nd differential thereof. Right: Theoretical Gauss shape built from the resulting *standard-deviation* i.e.  $\sigma$  and amplitude.

### Contour filter 6 Circumference

```
function EddyCircumference
```

Circumference e.g. line-length described by the contour. This is the other parameter needed for contour-filter 8. This is however neither related to the actual eddy scale determined in contour-filter 11.

### Contour filter 7 Area

```
function getArea
```

The main goal here is to determine the area encompassed by the interpolated coordinates of the contour. It does so via MATLAB's `polyarea` function. This area is not related to the scale  $\sigma$  that is determined in contour-filter 11. It is however the relevant scale for the determination of the isoperimetric quotient in contour-filter 8.

If the respective switch is turned on, this function also checks that the area of found contour does not surpass a given threshold which in turn is a function of  $L_R^1$ . Since  $L_R^1$  gets very small in high latitudes a lower bound on the  $L_R^1$  used here should be set as well. This is especially important for the southern ocean where  $L_R^1$  gets very small while the strong mesoscale turbulence of the Antarctic circumpolar current results in an abundance of relatively large eddies as far south as 60°S and beyond.

### Contour filter 8 Shape

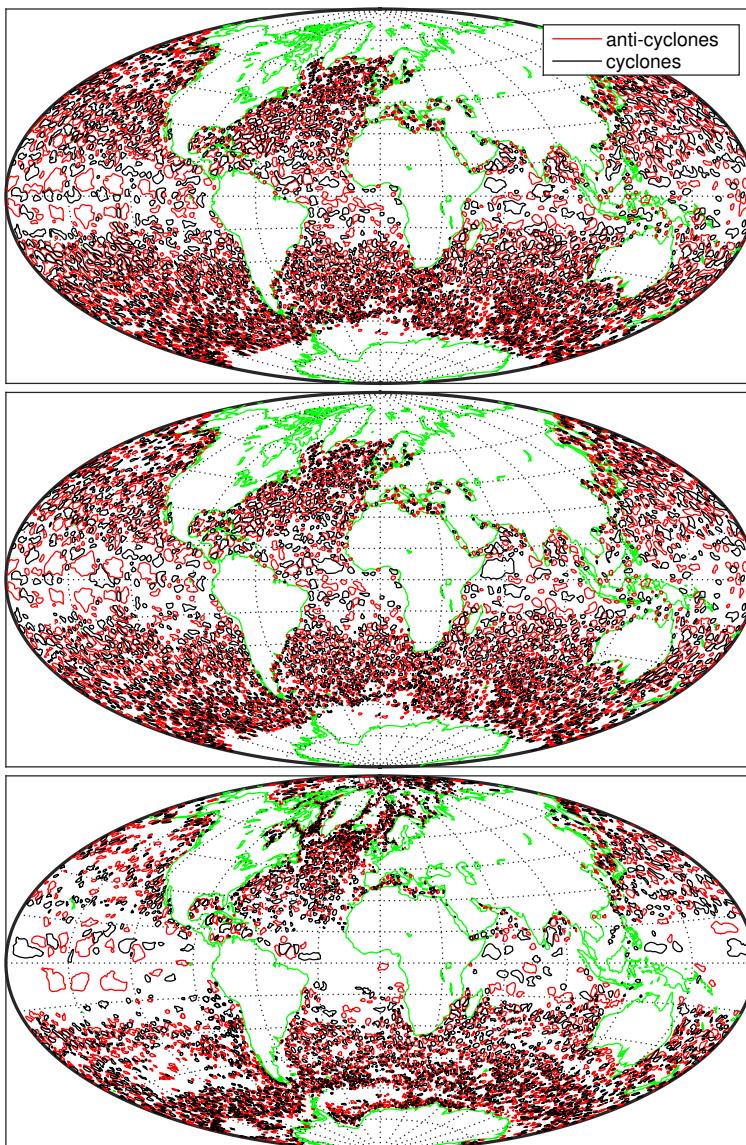


Figure 1.7: All contours that passed the filtering procedure for one exemplary time-step. Top: Aviso-MI . Mid: Aviso-MII . Bottom: POP-7day-MII .

```
function CR_Shape
```

This is the crucial part of deciding whether the object is *round enough*. A perfect vortex with  $\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$  is necessarily a circle. The problem is that eddies get formed, die, merge, run into obstacles, get asymmetrically advected etc. To successfully track them, it is therefore necessary to allow less circle-like shapes whilst still avoiding to e.g. count 2 semi-merged eddies as one.

This is achieved by calculating the isoperimetric quotient, defined as the ratio of a ring's area to the area of a circle with the same circumference. [Chelton et al. \(2011\)](#) use a similar method. They require:

*The distance between any pair of points within the connected region must be less than a specified maximum (Chelton et al., 2011).*

While this method clearly avoids overly elongated shapes it allows for stronger deformation within its distance bounds.

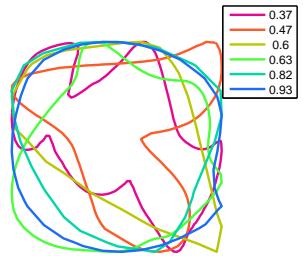


Figure 1.8: Different values of the isoperimetric quotient.

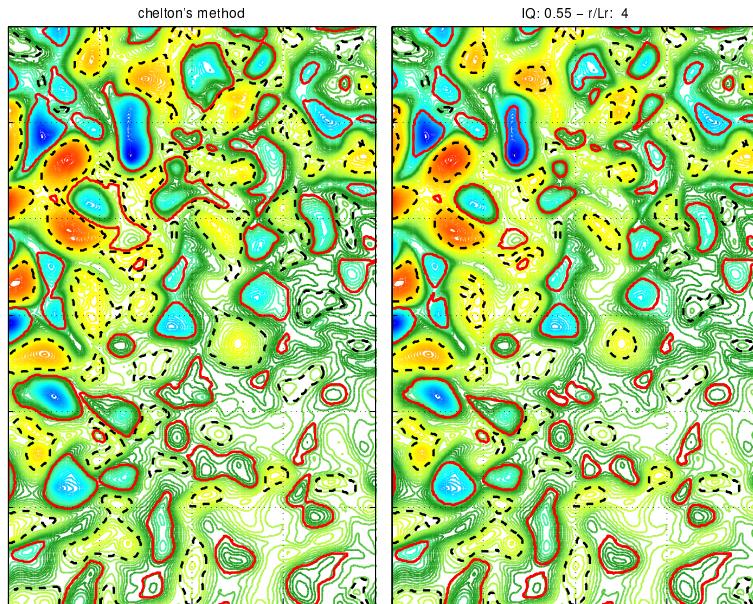


Figure 1.9: Left: The Chelton-method expects to detect eddies at their base and is rather tolerant with respect to the shape of found contour. The IQ-method aims more at detecting single round vortices without expecting found contour to be necessarily related to any howsoever-defined outer edge of the eddy.

### Contour filter 9 Amplitude

```
function CR_AmpPeak
```

This function determines the amplitude *i.e.* the maximum of the absolute difference between SSH and current contour level and the position thereof as well as the amplitude relative to the mean SSH value of the eddy interior as done by [Chelton et al. \(2011\)](#). The amplitude is then tested against the user-given threshold. The function also creates a matrix with current contour level shifted to zero and all values outside of the eddy set to zero as well.

### Contour filter 10 Profiles

```
function EddyProfiles
```

This step

- saves the meridional and zonal profiles of SSH, U and V through the Eddy's peak, spanning the entire sub-domain as described in contour-filter 4.
- creates spline functions from the ssh-profiles and uses them to interpolate the profiles onto 1000-piece equi-distant coordinate vectors to build smooth interpolated versions of ssh-profiles in both directions.
- in turn uses the splined data to create smooth 4-term Fourier series functions for the profiles.

### Contour filter 11 Dynamic Scale ( $\sigma$ )

`function EddyRadiusFromUV`

The contour line that is being used to detect the eddy is not necessarily a good measure of the eddy's *scale* *i.e.* it doesn't necessarily represent the eddy's outline very well. This becomes obvious when the area, as inferred by contour-filter 7, is plotted over time for an already successfully tracked eddy. The result is not a smooth curve at all. This is so because at different time steps the eddy usually gets detected at different contour levels. Since its surrounding changes continuously and since the eddy complies with the testing-criteria the better the closer the algorithm gets to the eddy's peak value, the determined area of the contour jumps considerably between time steps. This is especially so for large flat eddies with amplitudes on the order of 1cm. If the contour increment is on that scale as well, the difference in contour-area between two time steps easily surpasses 100% and more. Since there is no definition for the *edge* of an eddy, it is defined here as the ellipse resulting from the meridional [zonal] diameters that are the distances between the first local extrema of orbital velocity (one negative, one positive) away from the eddy's peak in y- [x-] direction<sup>4</sup>. In the case of a meandering jet with a maximum flow speed at its center, that is shedding off an eddy, this scale corresponds to half the distance between two opposing center-points of the meander. It is also the distance at which a change in vorticity-polarity occurs and is thus assumed to be the most plausible dividing line between vortices. Trying to determine the location where this sign change in vorticity occurs in the profiles turns out to be very tricky. What we seek are local extrema of the geostrophic speeds *i.e.* of the SSH-gradients  $h_x$ . In a perfect Gaussian-shaped eddy, these would simply correspond to the first local extrema of  $h_x$  away from the peak. In *reality* the eddies can be very wobbly with numerous local maxima and minima in the gradients of their flanks. One could argue, that it must be the largest extrema, as it is the highest geostrophic speeds that are sought. In practice<sup>5</sup>, multiple superimposed

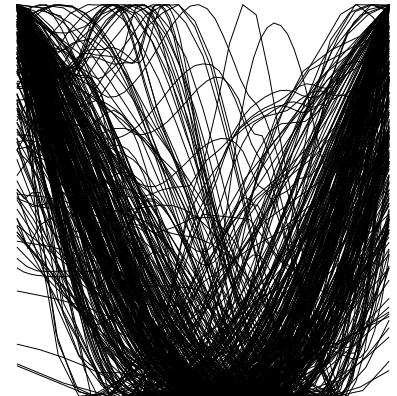


Figure 1.10: Zonal x- and z-normalized cyclone-profiles (early data  $\sim '13/12$ ).

<sup>4</sup> The velocities are calculated from the gradients of 4th-order Fourier fits to the SSH profile in respective direction (see contour-filter 10).

<sup>5</sup> Especially for the high-resolution model data.

signals of different scales often create very strong gradients locally. But the main issue here is that one weak eddy adjacent to one strong eddy also has the stronger gradients of the stronger one within its domain so that simply looking for the fastest flow speeds along the profiles is insufficient. It is also not possible to restrict the cut domain to the extent of a single eddy only, because at the time when the domain is selected, we do not know yet whether the detection algorithm *took bait* at the eddy's base or later close to the tip.

The best method thus far seems to be to use the Fourier-series functions from contour-filter 10 to determine the first extrema away from the eddy's peak (see fig. 1.11). The Fourier order was chosen to be 4 by trial and error. The effect is that small-scale low-amplitude noise is avoided, allowing for more reliable determinations of  $\nabla^2 h_{\text{fourier}} = 0$ .

Once the zero crossings in all 4 directions are found, their mean is taken as the eddy's scale ( $\sigma$ ).

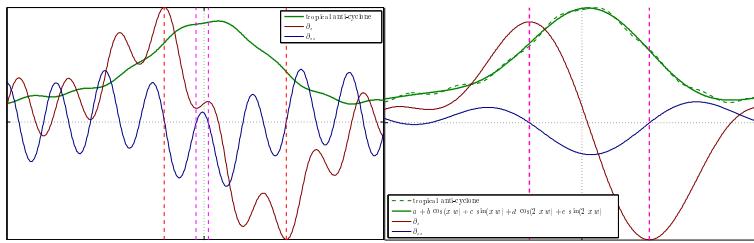


Figure 1.11: A flat wobbly low-latitude eddy resulting in multiple zero-crossings of its  $\nabla^2$ . The problem is addressed by differentiating the profile's Fourier-Series fit instead.

### Contour filter 12 Dynamic Amplitude

```
function EddyAmp2Ellipse
```

As mentioned above, the contour that helps to detect the eddy is not representative of its extent. This is also true for the  $z$ -direction, for the same reasons. This function therefore takes a SSH-mean at indices of the ellipse spun by the determined zonal and meridional *dynamical* diameters, and uses this as the basal value to determine a *dynamic* amplitude.

### Contour filter 13 Center of Volume (CoV)

```
function CenterOfVolume
```

Instead of using the geo-position of the eddy's peak in the tracking procedure, it was decided to instead use the center of the volume created by the basal shifted matrix from contour-filter 9 *i.e. the center of volume of the dome (resp. valley) created by capping off the eddy at the contour level*. This method was chosen because from looking at animations of the tracking procedure it became apparent that, still using peaks as reference points, the eddy sometimes jumped considerably from one time step to the next if two local maxima existed within the eddy. E.g. in one time-step local maximum  $A$  might be just a little bit larger than local maximum  $B$  and one time-step later a slight shift

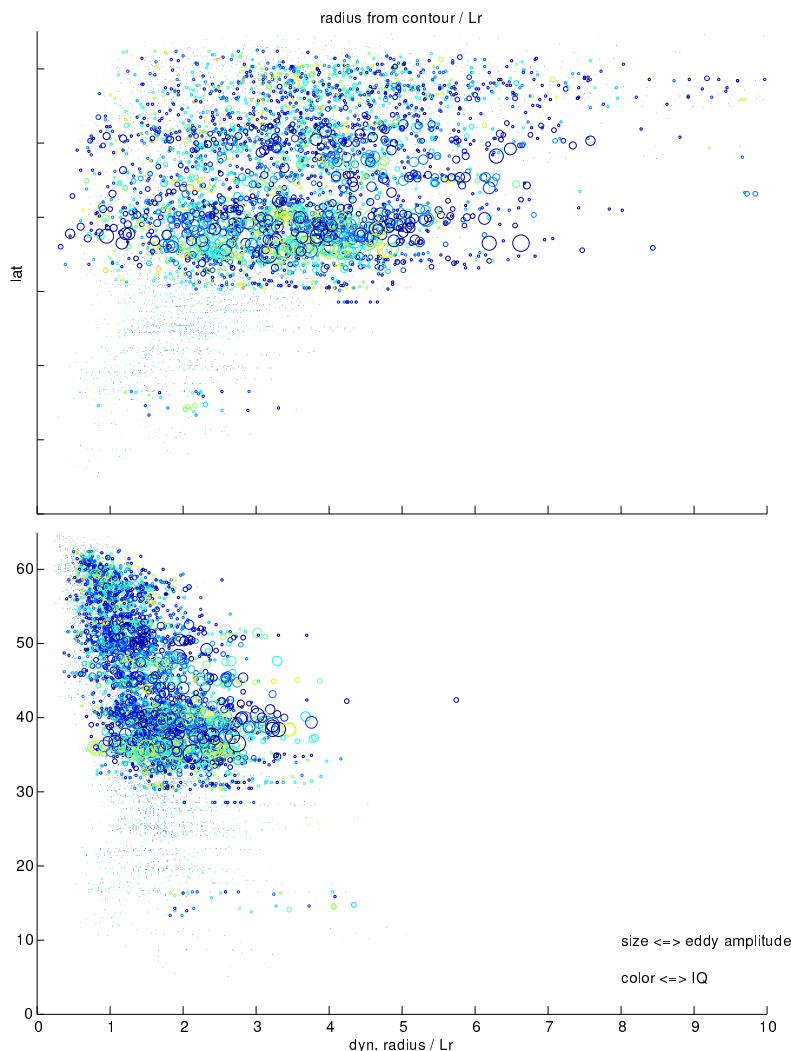


Figure 1.12: Eddies in the North-Atlantic. Y-axis: latitude. X-axis top: ratio of radius of circle with equal area to that of found contour to local Rossby-radius. X-axis bottom: ratio of  $\sigma$  to local Rossby-radius. Color-axis: Isoperimetric Quotient (from blue to red). Size: amplitude. The bottom plot suggests that a ratio of say 4 for  $\sigma/L_R^1$  should be a reasonable threshold.

of mass pushes local maximum  $B$  in pole position, creating a substantial jump in the eddy-identifying geo-position hence complicating the tracking procedure.

### Contour filter 14 Geo Projection

```
function ProjectedLocations
```

A threshold on the distance an eddy is allowed to travel over one time-step is implemented in the tracking algorithm in section 1.4. This is a direct adaptation of the ellipse-based constraint described by [Chelton et al. \(2011\)](#). The maximum distance in western direction traveled by the eddy within one time-step is limited according to  $x_{west} = \alpha c \delta t$  with  $c$  as the local long-Rossby-wave phase-speed and

e.g.  $\alpha = 1.75$ . In eastern direction the maximum is fixed to a value of e.g.  $x_{east} = 150\text{km}$ . This value is also used to put a lower bound on  $x_{west}$  and for half the minor axis ( $y$ -direction) of the resultant ellipse.

This function builds a mask of eligible geo-coordinates with respect to the next time-step.

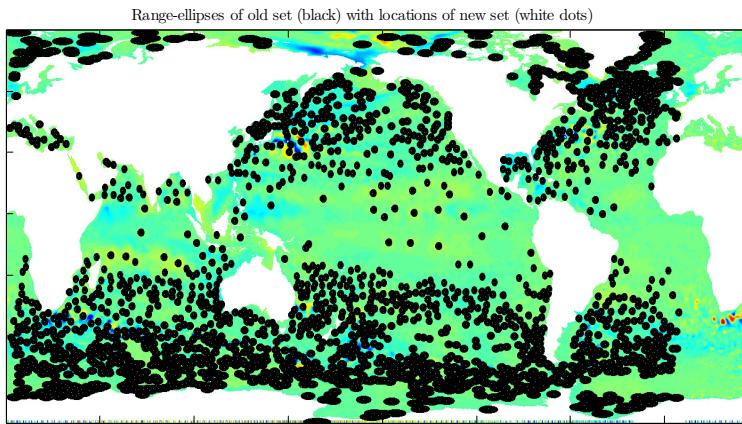


Figure 1.13: Among the saved meta-information for each eddy are also the indices describing the ellipse that defines the eddy's allowed locations for the next time-step.

## 1.5 Step S04: Track Eddies

```
S04_trackEddies
```

### 1.5.1 Main Tracking Procedure

THE algorithm builds an arc-length distance-matrix between all old and all new eddies and then determines the minima of that matrix in both directions *i.e.* one array for the new with respect to the old, and one for the old with respect to the new set. This leads to the following possible situations:

- Old and new agree on a pair. I.e. old eddy  $O_a$  has a closest neighbor  $N_a$  in the new set and  $N_a$  agrees that  $O_a$  is the

closest eddy from the old set. Hence the eddy is tracked.  $N_a$  is  $O_a$  at a later time.

- $N_a$  claims  $O_a$  to be the closest, but  $N_b$  makes the same claim. I.e. two eddies from the new set claim one eddy from the old set to be the closest. In this situation the closer one is decided to be the old one at a later time-step and the other one must be a newly formed eddy.
- At this point all new eddies are either allocated to their respective old eddies or assumed to be *newly born*. The only eddies that have not been taken care of are those from the old set, that *lost ambiguity* claims to another old eddy, that was closer to the same claimed new eddy. I.e. there is no respective new eddy available which must mean that the eddy just *died*. In this case the entire track with all the information for each time step is archived as long as the track-length meets the threshold criterion. If it doesn't, the track is abandoned.

### 1.5.2 Improvements

The former is the core of the tracking algorithm. It is almost sufficient by itself as long as the temporal resolution is fine enough. The larger the time-step, the more ambiguities arise, which are attempted to be mitigated by flagging elements of the distance matrix not meeting certain thresholds:

- `function` `checkDynamicIdentity`

Consider the ambiguous case when there are two new eddies  $N_a$  and  $N_b$  in sufficient proximity to old eddy  $O_a$ . Let's assume  $O_a$  is a relatively solid eddy of rel. large scale with a steep slope *i.e.* large amplitude and that  $N_a$  is merely a subtle blob of an eddy whilst  $N_b$  is somewhat similar to  $O_a$  but with only half the amplitude. The situation then is clear:  $N_b$  is the, apparently slowly dying,  $O_a$  at a later time, while  $N_a$  could either be a newly formed eddy, an old eddy with its respective representation in the old set something other than  $O_a$ , or even just temporary coincidental noise not representative of any significant mesoscale vortex at all. This interpretation should hold even when  $O_a$  sat right between the other two, thereby being much closer to  $O_a$  than  $N_b$  was. The purpose of this step is to make such decisions. It does so by comparing the *dynamic* versions of amplitude and scale (*ampToEllipse* and  $\sigma$ ) between the time-steps. If either ratio from new to old<sup>6</sup> surpasses a given threshold, the pair is flagged as non-eligible. It is important to use the *dynamic* parameters rather than those stemming from the contour line, because as mentioned in contour-filter 11, the contour line itself and the eddy's geometric *character* are hardly

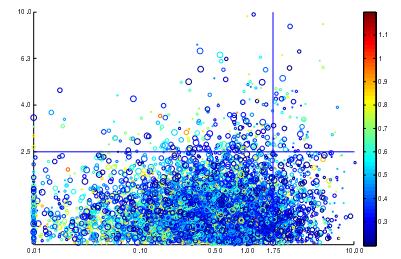


Figure 1.14: Each circle represents one eddy in the new time step. Y-axis: Maximum ratio to closest eddy in old set of either amplitude or  $\sigma$ , where 1 means *identical* and 2 means factor 2 difference. The threshold used for the final runs was 2. X-axis: Ratio of distance to closest eddy from old set divided by  $\delta t$  to local long-Rossby-wave phase-speed. Color-axis: Isoperimetric Quotient. Radius of circles: ratio of  $\sigma$  to local Rossby-radius. All eddies with said ratio larger than 10 are omitted. Note the obvious inverse correlation of scale to IQ, suggesting that all large *eddies* likely represent more than one vortex.

<sup>6</sup> In order to compare in both directions equally:  $\exp(|\log(v_n/v_o)|)$  where  $v$  is either amplitude or scale.

correlated at all. One eddy can get detected at different  $z$ -levels from one time-step to the next, resulting in completely different amplitudes, scales and shapes with respect to the contour.

The initial idea was, by assuming Gaussian shapes, to construct a single dimensionless number representing an eddy's geometrical character built upon the contour-related amplitude- and scale values only. Since we have no information about the vertical position of a given contour with respect to assumed Gauss bell, this problem turned out to be intrinsically under-determined and hence useless. The method eventually used, which checks amplitude and scale separately is again very similar to that described by [Chelton et al.](#).

- `function` `nanOutOfBounds`

This is the second half of the prognostic procedure described in section 1.5. It simply flags all pairs of the distance matrix for which the index representing the *new* eddy's geographic location is not among the set of indices describing the ellipse<sup>7</sup> around respective *old* eddy.

<sup>7</sup> see figure fig. 1.13.

- 

## 1.6 Step P01: analyze tracks

`P01_analyzeTracks` **TODO:**

## 1.7 Step P02: build mean-maps

`P02_analyzedTracks2maps` **TODO:**

## 1.8 Running the Code

THE separate steps can be run all at once (`Sall.m`) or one by one, as long as they are started consecutively in the order indicated by their name (`S00...`, then `S01...` etc.). Each step either creates its own output files or extends old ones, which are then read by the next step. All output data is saved in the user given root-path. This concept uses quite a lot of disk space and is also slowed substantially by all the reading and writing procedures. The benefit is that debugging becomes much easier. If the code fails at some step, at least all the calculations up to that step are dealt with and do not need to be re-run. The concept also makes it easier to extend the code by further add-ons and guarantees scalability.

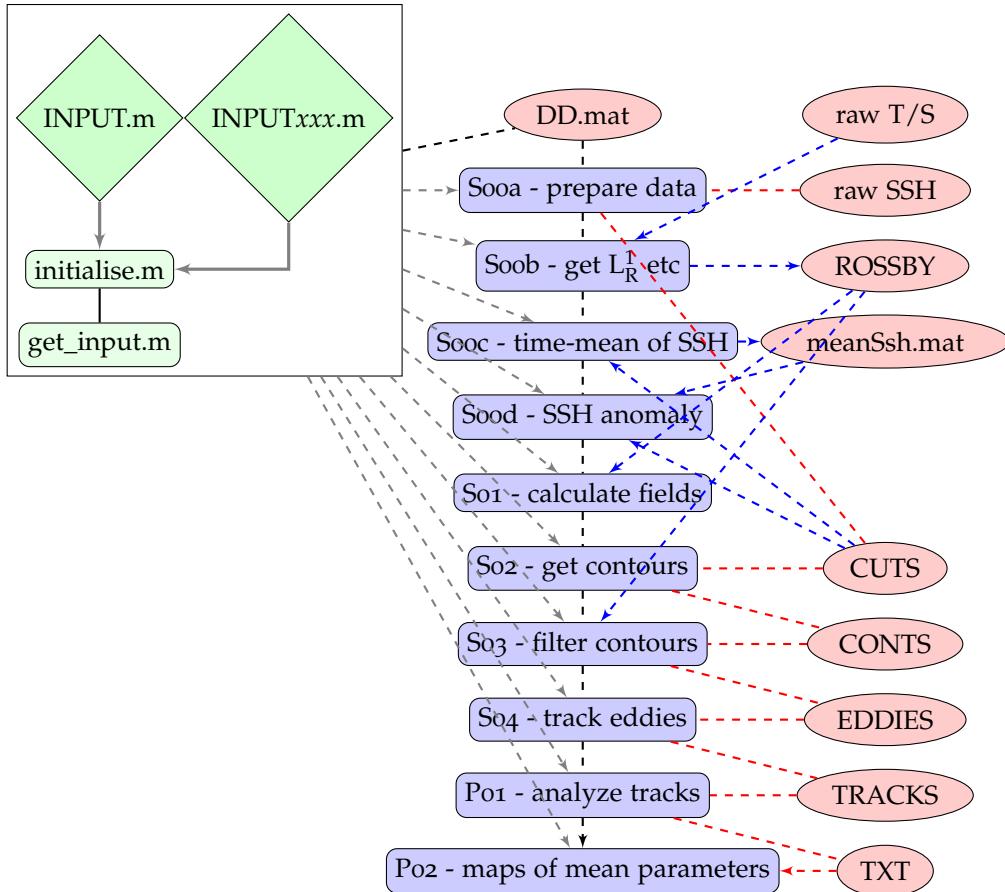


Figure 1.15: Basic code structure. The only files that are to be edited are the INPUT files. INPUT.m is independent of the origin of data, whilst the files INPUTTaviso.m, INPUTpop.m etc set are samples of more source-specific parameter settings. Each of the SXX-steps initially calls initialise.m, which in turn scans all available data, reads in the INPUT data via get\_input.m, corrects for missing data etc and creates DD.mat. The latter is the main meta-data file, which gets updated throughout all steps. All data is built step-by-step along the consecutive SXX-steps (red line). Note that meanSsh.mat should be recalculated if the time span is changed!



## *Bibliography*

Chelton, Dudley B., Schlax, Michael G., Samelson, Roger M., & de Szoeke, Roland a. 2007. Global observations of large oceanic eddies. *Geophys. Res. Lett.*, **34**(15), L15606.

Chelton, Dudley B., Schlax, Michael G., & Samelson, Roger M. 2011. Global observations of nonlinear mesoscale eddies. *Prog. Oceanogr.*, **91**(2), 167–216.