Contents

Definition 1: Reynolds Number Re [ ]

Compares advection of momentum to frictional acceleration.
$\mathrm{Re} = \frac{UL}{\nu}$

Definition 2: Rossby Number Ro [ ]

Compares advection of momentum to Coriolis acceleration.
$\mathrm{Ro} = \frac{U}{fL}$

Definition 3: Rhines Number R_β [ ]

Ratio of Rhines scale to horizontal scale.
$\mathrm{R}_\beta = \frac{U}{\beta L^2} = \frac{a}{L}\mathrm{Ro}$

Definition 4: Burger Number Bu [ ]

Ratio of relative vorticity to *stretching* vorticity.
$\sqrt{\mathrm{Bu}} = \frac{NH}{fL} = \frac{\mathrm{L_R}}{L}$

Definition 5: mass $m\,[kg]$

Definition 6: gravitational acceleration g $[m/s^2]$

Value of surface normal component of all body forces.

Definition 7: vorticity $\omega\,[1/s]$

Definition 8: Buoyancy Vector B $[1/s^2]$

$\mathbf{B} = -\frac{\nabla\rho\times\nabla p}{\rho^2}$

Definition 9: Kinetic Energy per mass E_k $[m^2/s^2]$

Definition 10: Mechanical Energy per mass E_k $[m^2/s^2]$

Sum of kinetic and potential Energy.

Definition 11: Rossby Radius $\mathrm{L_R}\,[m]$

The *geostrophic wavelength*. $\mathrm{L_R} = c/f$

Definition 12: Steering Level z_S

The critical depth where the real part of the Doppler shifted phase speed $c_S(z_S) = c(z) - u(z) = 0$ vanishes. I.e. the depth where Doppler shift creates a standing wave, causing the disturbances to grow in place instead of spreading space, analogous to a *supersonic bang*.

Definition 13: Rhines Scale $\mathrm{L}_\beta\,[m]$

scale at which earth's sphericity becomes important.
$\mathrm{L}_\beta{}^2 = \frac{U}{\beta}$

Definition 14: Gravity Wave Phase Speed c $[m/s]$

$\mathrm{c} = \sqrt{g'H}$

Definition 15: Reduced Gravity $g'(x,y,z)$ $[m/s^2]$

In the layered model $g' = g\frac{\delta\rho}{\rho_0} = N^2h$

Definition 16: Surface/interface Displacement $\eta(x,y)\,[m]$

Definition 17: Brunt Väisälä frequency $N\,[1/s]$

$N^2 = g/\rho_0\frac{\partial\rho}{\partial z}$

Definition 18: Mean Layer thickness $H\,[m]$

Definition 19: Layer Thickness/physical height of an isopycnal surface $h(x,y,t)\,[m]/h(x,y,\rho,t)\,[m]$

$h = H + \eta$ (in the layered model)

Definition 20: Planetary Vorticity $\Omega\,[1/s]$

$\Omega = 4\pi/\mathrm{day}_{fix\star}$

Definition 21: Latitude $\phi\,[\mathrm{rad}]$

Definition 22: Earth's Radius $a\,[m]$

Definition 23: Surface-Normal Planetary Vorticity Component f $[1/s]$

$\mathbf{f} = f\,\mathbf{z} = \Omega\sin\phi\mathbf{z}$

Definition 24: Change of Planetary Vorticity with Latitude $\beta\,[1/ms]$

$\beta = \frac{\partial f}{\partial y} = \Omega/a\cos\phi$

Definition 25: Okubo-Weiss Parameter $\mathrm{O_w}$ $[1/s^2]$

$\mathrm{O_w}$ = *divergence*^{2} + *stretching*^{2} + *shear*^{2} − *vorticity*^{2}.
A negative value indicates vorticity dominated motion, whereas a positive value indicates deformation.

Definition 26: Sea Surface Height SSH $[m]$

Definition 27: Isoperimetric Quotient IQ [ ]

$\mathrm{IQ} = A/A_c = \frac{A}{\pi r_c^2} = \frac{4\pi A}{U^2} \leq 1.$
The ratio of a ring's area to the area of a circle with equal circumference.

Chapter 1

The Algorithm

This section 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 well defined 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 did offer much of a benefit. The concept was therefor reversed. `S00_prep_data` can be altered to produce required output. 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_vars.m`.

1.1 Step S00: Prepare Data

`function S00_prep_data`
Before the actual eddy detection and tracking is performed, SSH-, latitude- and longitude-data is extracted from the given data at desired geo-coordinate bounds and saved as structures in the form needed by the next step (S01).

1.1.1 S00: Set Up

`function [DD]=set_up`
The main purpose of this step is to Find the boundary-indices in the data describing a window that includes the user-chosen geo-coordinate window.

1.1.2 S00: Parallel Part

`function spmd_body(DD)`
This function distributes chunks of data (i.e. bins of files) to the threads, which then perform the cutting and save their outputs to `'../DATA/CUTS/'`. The code can handle geo-coordinate input that crosses the longitudinal seam of the input data. E.g. say the input data comes 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. If the desired range is zonally continuous e.g. the entire southern ocean, the first tenth (zonally) of the input data is appended to the end of the output. This ensures that eddies right on the seam can be found and that eddies can be tracked across the seam. This results in eddies being identified twice in `function S03_filter_eddies.m` . Such *phantom* eddies are filtered out in `S04_track_eddies`.

ref

1.2 Step S01: Find Contours

`function S01_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 [1]. The contour intervals are determined by the user defined increment and range from the minimum- to the maximum of given SSH data.
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). `initialise.m` also distributes the files to the threads i.e. parallelization is in time dimension.

[1] see the MATLAB documentation.

1.3 Step S01b: Find Mean Rossby Radii and Phase Speeds

`function S01b_BruntVaisRossby`

TODO!

1.4 Step S02: Calculate Geostrophic Parameters

`function S02_infer_fields`
This step reads the cut SSH data from `S00_prep_data` to

- use one of the files' geo-information to determine f , β , g and the ratio g/f .
- calculate geostrophic fields from SSH gradients.
- calculate deformation fields (vorticity, divergence, stretch and shear) via the fields supplied by the last step.
- calculate $\mathrm{O_w}$.

1.5 Step S03: Filter Eddies

`function S03_filter_eddies`
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 as the outline of an eddy as defined by the user input threshold parameters.

1.5.1 Reshape for Filtering

`function eddies2struct`
In the first step the potential eddies are transformed to a more sensible format, that is, a structure `Eddies(EddyCount)` where `EddyCount` is the number of all contours. The struct has fields for level, number of vertices, exact i.e. interpolated coordinates and rounded integer coordinates.

1.5.2 Correct out of Bounds Values

`function CleanEddies`
The interpolation of `contourc.m` sometimes creates indices that are either smaller than 0.5 or larger than $N + 0.5$ [2] 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.

1.5.3 Descent/Ascend Water Column and Apply Checks

The concept of this step is a direct adaption 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 anti-cyclones [3] effect an elevated SSH i.e. a *hill*. The algorithm ascends the full range SSH levels where contours were found. Consider 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 encompass 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 AC's 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 identical except for a factor -1 where applicable. In the following the most important steps of the analysis are outlined.

[2] where N as the domain size
[3] henceforth abbreviated AC

1.5.3.1 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.

1.5.3.2 Closed Ring

`function CR_ClosedRing`

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

1.5.3.3 Sub-Window

`function get_window_limits`
`function EddyCut_init`

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.

1.5.3.4 Logical Mask of Eddy Interiour

`function EddyCut_mask`

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 `imfill.m`, which was also used in the very first version of 09/2013.

1.5.3.5 Islands

`function CR_Nan`

No flags within the eddy are allowed. This check also avoids contours around islands as all land is flagged *a priori*.

1.5.3.6 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.

1.5.3.7 Area

`function EddyCircumference`

Area described by the contour. This is needed for 1.5.3.9. This is not however related to the actual eddy scale determined in 1.5.3.12.

1.5.3.8 Cirumference

`function EddyCircumference`

Line-length (sum of Euclidean norm of all nodes) of the contour.

1.5.3.9 Shape

`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 therefor 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 equal 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.

compare both, show pictures

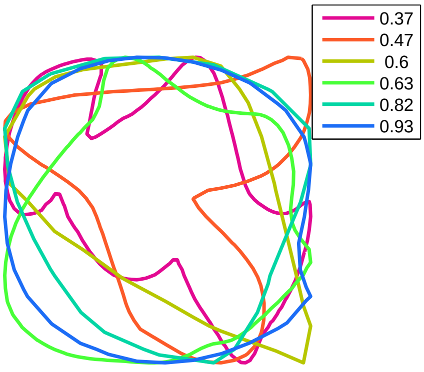

Figure 1.1: Different values of the isoperimetric quotient.

1.5.3.10 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.

1.5.3.11 Profiles

`function EddyProfiles`

This step saves the meridional and zonal profiles of SSH, U and V.

1.5.3.12 *Dynamic* Scale

`function EddyRadiusFromUV`

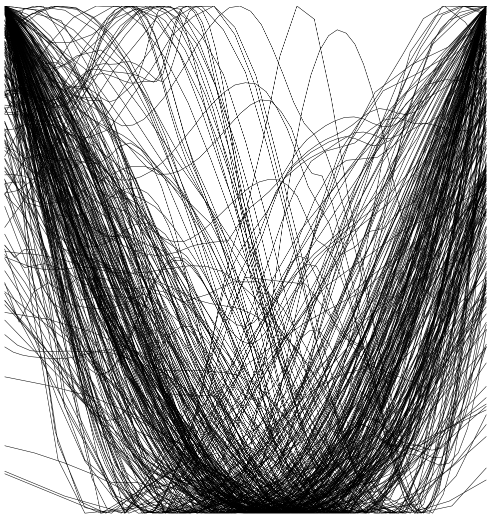

Figure 1.2: Zonal x- and z-normalized cyclone-profiles.

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 very obvious when the area, as inferred by 1.5.3.7, 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 continuously changes 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 here defined as the ellipse resulting from the meridional and zonal diameters that are the distances between first minimum and maximum orbital velocity, away from the eddy's peak in positive and negative y and x directions respectively. 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. In practice the velocity-gradient profiles need to be smoothed to successfully

theres probably a better way via okubo weiss!

determine their first adequate zero-crossing. Once the zero crossings in all 4 directions are found, their mean is taken as the eddy's scale. Note that this is again similar to what Chelton *et al.* (2011) did.

1.5.3.13 *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 therefor takes an SSH-mean at indices of the ellipse created by the determined zonal and meridional *dynamical* diameters, and uses this as the basal value to determine a *dynamic* amplitude.

1.5.3.14 Center of Volume

`function EddyArea2Ellipse`
`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 1.5.3.10 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 of mass pushes local maximum B in pole position, creating a substantial jump in the eddy-identifying geo-position hence complicating the tracking procedure.

1.6 Step S04: Track Eddies

`function function S04_track_eddies`
Due to the the relatively fine temporal resolution (daily) of the model data, the tracking procedure turns out to be much simpler than the one described by Chelton *et al.* (2007). There is really no need to project the new position of an eddy, as it generally does not travel further than its own scale in one day. This means that one eddy can be unambigiously tracked from one time step to the next as long both time-steps agree on which eddy from the other time-step is located the least distance away. The algorithm therefor simply 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 neighbour in the new set N_a 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* ambigious 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 saved as long as the track-length meets the threshold criterium. If it doesn't, the track is deleted.

1.7 Step S05: Cross Reference Old to New Indices

`function function S05_init_output_maps`
The main purpose of this step is to allocate all grid nodes of the input data to the correct node of the output map. Since the output map is usually much coarser than the input data there is no need for interpolation.

1.8 Step S06: Make Maps of Mean Parameters

`function function S06_analyze_tracks`

see github for progress

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.