Thank you very much for your positive comments and constructive feedback, you addressed some important points. Your clarifications helped to make the manuscript clearer for the reader. Our responses are provided in green (changes made in the manuscript are written in **bold**) together with your original comments in black. We really appreciate your time and insight in reviewing our manuscript!

Kind regards,
Susanna (on behalf of all co-authors)

Given the important roles of ocean circulation in shaping the climate system, correctly diagnosing the volume/salinity/heat transport in the ocean models is always important but complicated by different model grid configurations. Winkelbauer et al. present two new methods in calculating ocean transport, with one sticking to the original model grid lines and the other following strict vector projection and interpolation. Both methods work well with different Arakawa grids, and the results given by these two methods are very similar to assumed analytical results. And I would say the code is versatile in that users can define strait sections in different ways (to my understanding, 3 ways in defining the straits). What makes me more satisfied is that the code is organized into a Python package and maintained in GitHub, which facilitates its future upgrade and public use. Both the paper and the code are well written and I think the paper should be published after considering/answering my following suggestions/questions.

Major concerns:

I know that this is a scientific rather than a technical documentation, but some technical details are still needed to help the readers understand the calculation process.

1. Line123-125: I have difficulty in understanding the processes given here. Are the **transports or projection vectors** "interpolated bilinearly onto the closest points on the reference line? Put it another way, after the projection vectors of all touched cells are calculated, what is the next? Do we (1) calculate transports using these projection vectors of the touched cells (which might need the length of the reference line that falls into each grid cell), then interpolate the transport value onto the closest points on the reference line, or (2) interpolate the projection vectors onto the reference line. Also, the transports are "divided by the respective cell thicknesses on the reference line". What do you mean "thickness" here? Vertical thickness or the length of the horizontally overlapped part? I suggest the authors to rewrite these 3-4 sentences and give it in a more clear way.

   (1) is the correct assumption, however the term "transport" was probably a bit misleading. We use the projection vectors and multiply the projected u and v components with the respective vertical thickness (different before and after the interpolation), but not with the respective length of the reference line (not

needed as we would need to divide by exactly the same length after the interpolation to obtain velocities again). This is done for every grid cell that touches the reference line and it's neighboring grid cells (those are needed for the interpolation in the next step, see minor comment 5), the "transports" are then interpolated onto the closest points on the reference line (T_proj) and to obtain velocities again we divide by the vertical thicknesses/extents at the T_proj points.

We changed the sentences to the following:
Using the magnitudes of the projection vectors we calculate the u and v components pointing orthogonally onto the strait at all grid cells touching the strait and their neighboring cells (needed for the bilinear interpolation). Then, we multiply them with the respective vertical cell thicknesses at the u/v points and interpolate those orthogonal "transports" bilinearly onto the closest points on the reference line (black crosses in Fig. 2b, called T_proj henceforth). In a final step we divide by the vertical thickness of the cells on the reference line to obtain velocities again.

2. Line190: Section2.2.3 talks about the halo grids (some people call these halos or halo grids. E.g., https://github.com/pangeo-data/xESMF/issues/109). This section deserves more sentences because the authors only said "these conditions have to be handled with care" but how? A detailed discussion on the halos might go beyond the scope of the current manuscript, but I would like to see more discussion on how to overcome the halos in order to a smooth use of StraitFlux. An example on dealing with the halo grid problem would be even better.
In the newest version of StraitFlux we implemented an automatic check where the algorithm checks whether any overlapping points exist at the cyclic or the northern boundary points, and if so, those are removed. While quite strait forward with the cyclic points, it's a bit trickier to account for all the different handlings of the northern boundary points. We tested the indices selection for a strait going "over" the northern boundary of the tripolar grids for multiple models with different boundary definitions (CanESM5, CMCC-CM2-SR5, ACCESS-CM2, CAMS-CSM1-0, IPSL-CM6A-LR, EC-Earth3) and in all cases the new version of StraitFlux selected the correct indices to avoid duplicates and/or gaps. Below a figure of the indices selection for the CMCC-CM2-SR5 model, which pivots the 2 top points at the northern boundary. The colorful cells show "duplicate points which are pivoted" onto the "other" side (see jump in x indices). The blue and red filled dots show the selected u and v indices. The blue "empty" dot in the right panel corresponds to the top blue filled dot in the left panel and is **not** used. Even for a complicated case as this no cells are counted twice and also no gaps are present (we have a u index between the green and red cell (left, index 80) and between the neighboring blue and yellow cell (right, index 279)). While the selection worked perfectly fine for the tested models, we still can't guarantee that it will work for all

possible conditions, therefore the code still outputs the warning: “Attention: Strait crossing the northern boundary – make sure correct indices are chosen!”. We added the figure to the appendix and adapted the text as follows:

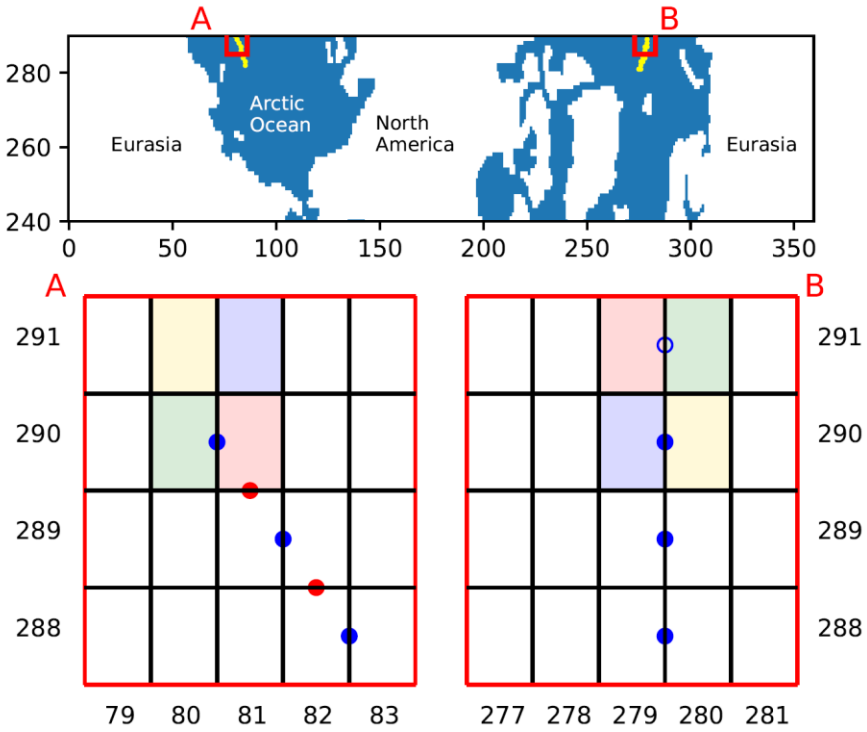


Figure A1. Indices selection across the northern boundary for the CMCC-CM2-SR5 model. Colorful cells show duplicate cells which are pivoted at the top boundary. Filled blue dots show selected u indices, filled red dots show selected v indices. Empty blue dot shows overlapping point which is not selected.

These conditions have to be handled with care, as especially the volume transport calculation is very sensitive and can yield useless results when there is a gap in the integration line or if any grid-cells are counted twice. ***StraitFlux automatically checks for overlapping cyclic boundary points and drops any duplicates, this should ensure correct transport calculations across the zonal boundaries independent of how the models deal with periodicity. Similarly concerning the north boundary conditions StraitFlux should automatically select the correct indices and avoid gaps and/or duplicates. We tested this successfully for an arbitrary line going over the top boundary of the model grids for various CMIP6 models with different boundary conditions (CMCC-CM2-Sr5, EC-Earth3, CanESM5, ACCESS-CM2, CAMS-CSM1-0, IPSL-CM6A-LR). Fig. A1 in the appendix shows an example for the CMCC-CM2-SR5 model. The top two rows of grid cells are rotated along the northern boundary, colorful cells show duplicate cells which are pivoted at the top boundary (= same cells but upside down). StraitFlux correctly chooses the indices so that a continuous line without overlaps is formed. While the indices selection worked for the tested models, the generated indices should still be checked to ensure a***

continuous line also for more complicated boundary conditions. Therefore, the code automatically outputs the warning "Attention: Strait crossing the northern boundary – make sure correct indices are chosen!" when moving across the boundary of the grid.

Minor concerns:

Dear

1. Line5: transports computed from those are not mass/**volume**
   We changed it to:
   *Use of data **interpolated to** standard latitude/longitude grids is not an option since transports computed from **interpolated velocities** are not mass consistent.*
2. Line45: it's not just the artificial meridional velocity. The artificial zonal velocity does not point to the true east either.
   we adapted the sentence to the following:
   *While solving the numerical problem of a singularity over the ocean, those curvilinear grids complicate the calculation of oceanic transports, especially in the proximity of the poles, as velocities in the direction of the artificial poles do not point in the direction of the true north **and artificial zonal velocities do not point to the true east**.*
3. Line46: "angle of the grid-lines" is always 90 degree since we are using general orthogonal curvilinear coordinates. I think you mean the angle between gridlines and regular lon-lat lines.
   Yes, we adapted it:
   *The exact position of the poles, **the angle between the native gridlines and regular longitude-latitude lines, as well as the horizontal and vertical resolution** varies between different models, forming a vast amount of different grid types that complicate inter-comparison between different models and to observations.*
4. Line81: Do xe and xw have to be land points in the code? Is possible to calculate transports between water points using the current version?
   It is possible to calculate transports between water points, however results should be viewed with caution as water will also pass to the left/right of the defined strait and the exact position of currents in the models is not known. Therefore, the code produces the requested transports, but it also gives a warning saying '!!!ATTENTION!!!: first/last point water, recheck indices line!'
   We added the following:
   *The boundaries z_b, x_1, x_2 should be chosen such that no water can "escape" the desired coast-to-coast section. This can be ensured if x_e and x_w are land points and the auxiliary fields describing model ocean depths are used appropriately. **It is also possible to calculate transports between two water points, however***

results should be viewed with caution and their correctness is left to the discretion of the user as water might bypass the strait.

5. In Figure 2b, four grid cells with blue, green and yellow arrows are shown. The one on the upper-left corner is misleading because the red reference line does not touch it.
   We added a cell not touching the reference line, as it is also needed for the bilinear interpolation of the orthogonal transports onto the strait.

   We clarified it in the text at L116-118:
   *For every grid cell touching the strait **and their neighboring cells (needed for the interpolation onto the strait)** we calculate direction vectors of the u and v components (blue and green arrows), and normal vectors pointing from the tracer grid cell in the direction of the strait (yellow arrows).*

   And at L123-125:
   *Using the magnitudes of the projection vectors we calculate the u and v components pointing orthogonally onto the strait at all grid cells touching the strait **and their neighboring cells (needed for the bilinear interpolation).** Then, we multiply them with the respective vertical cell thicknesses at the u/v points and interpolate those orthogonal "transports" bilinearly onto the closest points on the reference line (black crosses in Fig. 2b, called T_proj henceforth). In a final step we divide by the vertical thickness of the cells on the reference line to obtain velocities again.*

6. Line155: The authors said "the section can be kinked". Here does "kinked" mean a zigzag line? If it does, then this is good because we do need zigzag sections from time to time. According to the definition of *def_indicies()* function, this is only possible if *set_latlon* = *True* and *lon_p* and *lat_p* are provided. Maybe you can put it more clear in the paper or in the code documentation.
   Yes that's true, at the moment that's the only way to define zigzag sections.
   We added the following:
   The determination of section positions for transport calculations is accomplished in the def_indices function. Users can specify the start and end points of a section using the 'coords' parameter in the 'transports' function, the section will then follow the shortest distance along the sphere. Alternatively, users may pass specific coordinates by setting the 'set_latlon=True' parameter and providing a list of latitude ('lat_p') and longitude ('lon_p') points. The latter option also allows the calculation of "kinked" sections.

7. Line157: a reference line "consisting of equally spaced latitude-longitude pairs". What is the interval between points on the reference line? (about 0.1deg according to *def_indices()*, but why?)
   We chose 0.1 deg to ensure that also the higher resolution (0.25°x0.25°) models don't skip any points. This works great for models with a resolution of

up to approx. 0.25°x0.25° (lower resolution models produce duplicate indices, however those are removed automatically), for even higher resolution models we advise to use an even denser spacing. **We adapted the code for the newest version of StraitFlux, so that the resolution is checked automatically, and the interval is adapted accordingly to 0.4*resolution** (so about 0.4 for 1° models and 0.1 for 0.25° models and even higher for higher resolution models). We added the following to the manuscript:
Using the 'coords' option *the function generates a reference line (ref_line) consisting of equally spaced latitude-longitude pairs* ***whereat the interval between points on the reference line is set automatically to suite the resolution of the model. When passing coordinates via the 'set_latlon=True' option we advise the user to use intervals not larger than 0.4 times the resolution of the model (e.g., intervals of 0.1° for models with a resolution of about 0.25°) as coarser intervals might lead to the skipping of grid points and generate broken lines. This might create duplicates in the indices found, however those will be removed automatically.***

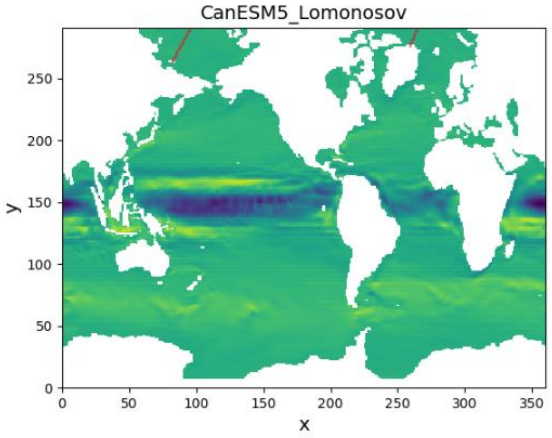
8. Line165: when I read the manuscript, I thought that if the interval between two neighboring points on the reference line is very small, then there should be duplicates in the indices found by *select_points()*. And when I read the code of *check_availability_indices()*, I realized that duplicates will be removed by the code automatically. But I still think it will be better if the authors explicitly write out that "**there might exist duplicates in the indices found by *select_points()*, but the code will later on remove the duplicates automatically**". This helps reader like me to release the puzzles.
See answer above

9. Line170: the first and last point should be land grid points. Again, is it possible to calculate transports between two ocean grid points?
Yes, it is possible, although we would advise to do it with caution. We adapted the sentence to the following:
To prevent water from "escaping", ***we advise the user to place the first and last point of the defined section over land. Transports may also be calculated for sections between two ocean points, however a warning will be given to the user as those should be treated with caution.***

10. Line175-176: left/right/above/below

It is straightforward to use these words to depict the directions of the grid line, but cautions need to be taken where grid lines are distorted greatly in the Arctic Ocean. For example, in a tripolar grid, if we draw a section following a meridional grid line in the Arctic Ocean (e.g., along the Lomonosov ridge), is the cell coming from left/right or above/below to its next one?
Note that the orientation of left/right/above/below is defined concerning the native grid indices (x,y), therefore the local direction of x is not necessarily west-east. E.g. coming from left in this sense means coming from x-1 to x. This should distinctly assign each direction (coming from above/below/left/right) to one move along the

native gridlines (coming from $y_{i+1}/y_{i-1}/x_{i-1}/x_{i+1}$), even for straits in the far north (see indices of Lomonosov on tripolar grid below).



We adapted Fig. 3 to show the indices selection process in respect to the native grid lines (x and y instead of lat/lon on the figure axes). Latitude and Longitude lines are shown in grey. Further, we changed the grid cell indices in Fig. 3c) to j to avoid confusion with the indices along the strait i in 3a+b).

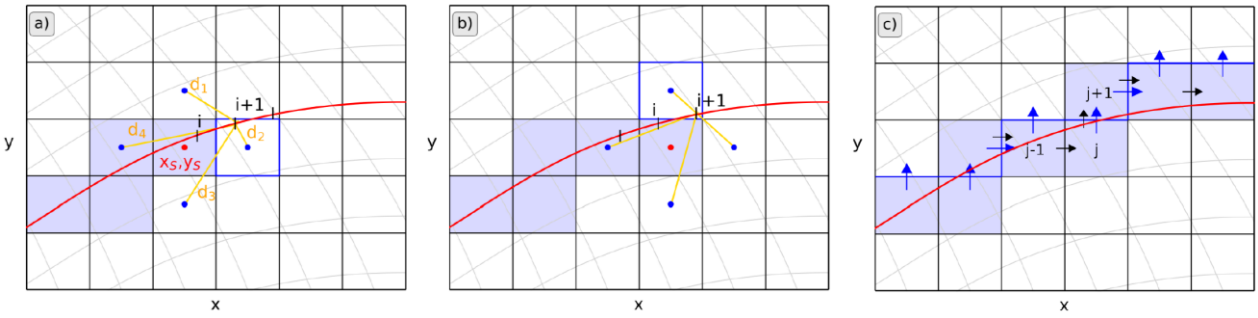


Figure 3. Illustration of the indices selection process using select_points. ***Lines of constant latitude/longitude are shown in grey.*** *a) and b) Determination of consecutive grid-cells* ***on the native grid*** *by comparing distances d (orange lines) of 4 surrounding grid-cells* ***for all equally spaced points i along the reference line****. c) Specification whether a u or v component should be taken.*

We adapted the indexing for Fig3c) in the text (L176-178):
For instance, to get to cell ***j*** *in Fig. 3c, we came from left, hence the v component of cell* ***j*** *is taken. In order to get to cell* ***j*** *+ 1 we come from below, therefore the u component of cell* ***j*** *+ 1 is taken.*

And we added the following footnote:
Note that we follow the native grid points (x,y) and the local direction of x is not necessarily west-east and the local direction of y not south-north. For instance, coming from left means coming from point [x_{i-1},y_i]to point [x_i,y_i].

1. Line227: similar to Line165: Are duplicates found by the code removed automatically?
   Yes, they are. We adapted the section to the following:
   As for the LM, the first step is to find the closest points on the native grids to the reference line. ***The selection of the indices proceeds similar as for LM, however herein additionally to the closest points to the reference line also the four immediate neighboring cells of the closest points are used. Those are needed for the interpolation of the transports onto the reference line. Again, any duplicate indices are removed automatically.***

2. Line313: typo “VPN”.
   We corrected it.
3. Line339: ITF needs to be clarified.
   We clarified it as Indonesian Throughflow Region.
4. Line373: what is “i.a.”?
   We changes it to “for instance”.
5. Line392: I can see that on github, there is an instruction on how to install StraitFlux. My own experience, however, is that I can shoot myself in the foot if I mix the use of conda and pip. In consideration of the future update of StraitFlux (e.g., Line128-129), I strongly encourage the authors to upload this tool to conda-forge.
   Thank you for the idea, we will consider uploading StraitFlux to conda-forge for a future version.

6. Line397: typo “downlaoded”.
   We corrected the typo.

Code bugs:

I test the code by myself and I do encounter some errors which turn out to be caused by bug in the code. When I ran the Examples.ipynb script, I got the following error.

TypeError: check_Arakawa() takes 4 positional arguments but 5 were given.

Then I found that *check_Arakawa()* only accepts 4 rather than 5 positional arguments. So in line113 and line131 of *mastersciprt_line.py*, there should be no the ***product*** parameter.
Thank you for checking the code. We corrected the error in the newest version of StraitFlux.