OVWST folks,

Let me apologize in advance for this software bug, which is going to force you to redo all of your divergence calculations if you made use of ScatTools. Just know that for my penance I will be suffering right alongside you, as my Philippines winds paper was slated to be submitted to JGR this week. There was something just persistently off about my Philippines divergence results, but I couldn’t lay my finger on it. So I tried an independent line-integral approach and could not reproduce my work. Then exploration of idealized wind fields revealed the finite difference approach in ScatTools to be transposing results in a weird way – it wasn’t just a simple sign error. But how?

Here is the culprit:

In scatttools.\_Scatterometer.\_calc\_div\_finite\_diff():

        dqu\_dx, dqu\_dy = np.gradient(smu[:, :], dxx, dyy)

        dqv\_dx, dqv\_dy = np.gradient(smv[:, :], dxx, dyy)

should instead be:

**dqu\_dy, dqu\_dx = np.gradient(smu[:, :], dyy, dxx)**

**dqv\_dy, dqv\_dx = np.gradient(smv[:, :], dyy, dxx)**

The reason for this can be gleaned from reading the numpy.gradient documentation (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.gradient.html)>. This function works on rows and then columns. In swath coordinates, rows (y & v’) are along the swath, while columns (x & u’) are across the swath. So y needs to be treated before x. In ScatTools, dyy & dxx are set to be equal and constant, so that swap didn’t matter in practice (though it is bad form to leave it technically incorrect). But the order of return matters crucially. In other words, the return matrices were getting swapped, and we were all calculating du/dy + dv/dx. This will often provide results that look plausible, but just a bit off if you stare a while. Like a vector-based uncanny valley!

At any rate, the two-line fix above solves the reproducibility error. Attached is a five-panel plot showing different divergence methods. (a) is the original winds. (b) is finite difference with no smoothing. (c) is finite difference with 5x5 smoothing of u’/v’ before the derivatives are taken (ScatTools approach). (d) is finite difference with 5x5 smoothing after the derivatives (mathematically, this turns out to be the same as smoothing before the derivative, hence why (c) and (d) look equivalent, but I wanted to check this anyway). (e) is the line-integral approach. The latter is a bit noisier than the standard ScatTools approach (this may be related to some simplifications in my line-integral approach – see below), but it is commonly seen in the literature (e.g., Kilpatrick and Xie 2015). At any rate, everything looks very similar, with really just degree of smoothing being the main difference. This is as it should be.

Also attached is the updated ScatTools. Please update your existing local repo. BTW, I plan to start the open-source process for this package soon.

If you are curious about the line-integral approach, here is a slow and hacky way to do it, with 12.5-km resolution assumed and the 50x50 km2 box being hardcoded in:

rs = scattools.RapidScat(files[0][:-3], pydap=False)

au, av = rs.\_get\_uprime\_vprime(rs.data['u'], rs.data['v'],

                               rs.data['latitude'], rs.data['longitude'])

ad = np.zeros\_like(au)

for i in xrange(np.shape(au)[0]):

    if i > 1 and i < np.shape(au)[0] - 2:

        if i % 100 == 0:

            print('i =', i, 'of', np.shape(au)[0] - 1)

        for j in xrange(np.shape(au)[1]):

            if j > 1 and j < np.shape(au)[1] - 2:

                # Average the three non-corner normal vectors along each box side.

                l1 = 5e4 \* np.mean([av[i+2, j-1], av[i+2, j], av[i+2, j+1]])

                l2 = 5e4 \* np.mean([au[i+1, j+2], au[i, j+2], au[i-1, j+2]])

                l3 = -5e4 \* np.mean([av[i-2, j-1], av[i-2, j], av[i-2, j+1]])

                l4 = -5e4 \* np.mean([au[i+1, j-2], au[i, j-2], au[i-1, j-2]])

                ad[i, j] = (l1 + l2 + l3 + l4) / (5e4)\*\*2

And **ad** will be your divergence. Note how in all methods, working in the swath grid (u’ & v’) is crucial. It especially makes line integrals straightforward. I plan to spruce up the line-integral approach in the future (maybe via Cython) and put it into ScatTools.

Again, my sincerest apologies for this bug.

Thanks,

Timothy Lang

Research AST, Atmospheric Measurements

NASA Marshall Space Flight Center (ST11)

Huntsville, AL 35812

(256) 961-7861

[timothy.j.lang@nasa.gov](mailto:timothy.j.lang@nasa.gov)