ADflow Zipper Mesh Debugging

Nick Bons and Anil Yildirim

March 13, 2019

1 Problem statement

Previous experiments by Ping and Sandy have shown that the zipper mesh derivatives are inaccurate. We want to quantify this inaccuracy and determine its source.

2 History

We have realized that the gradients obtained when we use zipper meshes to integrate forces on the surfaces are not accurate, whereas cases with no zipper meshes yield accurate gradients. Ping and Sandy tried to fix this bug. Ping found one bug where the order of the calls in the reverse mode master routine was wrong for the surface integrations. The commit with this fix can be found at Anil's fork of ADflow on Bitbucket, commit c4b1e38. Ping reported that while this improved the gradients' accuracy, the results still did not match the complex step method exactly.

After this, Ping also tried to comment out all the zipper mesh integration calls, to validate that the bug is caused by these routines. The commit with this modification can be found at Anil's fork of ADflow on Bitbucket, commit 11ca464. This seemed like it fixed the issue, and the gradients obtained with the adjoint method matched the ones obtained with the complex step method.

After this point, Ping and Sandy stopped working on this bug. No actual commits were pushed to the main ADflow branch. The fix in the order of surface integration calls is also not pushed to the main repo, all these modifications existed on Ping's fork, and now, on Anil's fork on Bitbucket.

Ping and Sandy concluded that the issue appeared when we have multiple zipper meshes in a configuration. This means that we should expect the gradients to match if the case has only one zipper mesh. A simple overset wing-fuse with a collar in between will have 2 zipper meshes, one on the wing and one on the fuse. To avoid the complexity, Ping created a 3d wing case with two overset patches on it. This case had only a single zipper mesh at around half span (well, half span of the half wing).

Finally, we started working on this bug before ADflow was moved to Github, therefore the commits we will reference in this document will be on Bitbucket. In the future, if this involves more coding, we will move these changes to Github, therefore look for similar commits on Anil's fork on Github. The commit messages will be identical.

3 Test case description

To rapidly analyze the effect of having no, 1 and 2 zipper meshes on a case, we created our own test cases. The goal with these cases is to quantify the effect of the number of zipper meshes.

The test case is a rectangular wing with NACA 0012 cross-sections and aspect ratio 6 (same as the ADODG wing). We generated three different grids for this geometry:

- Wing 0 - a continuous multi-block wing mesh
- Wing 1 - two meshes that overlap at half-span (one zipper)
- Wing 2 - two non-overlapping meshes with an overlapping collar mesh in between them (two zippers)

All three grids are extruded using pyHyp to 3 m off-wall and combined with a background mesh that extends to 100 m off-wall. The initial off-wall spacing for the pyHyp extrusions is 1 mm, intended for a solution of the Euler equations. The zippered surface meshes are shown in Figure 1. All solutions were obtained for Mach= 0.8 at sea level with 1.8 degrees of angle-of-attack.

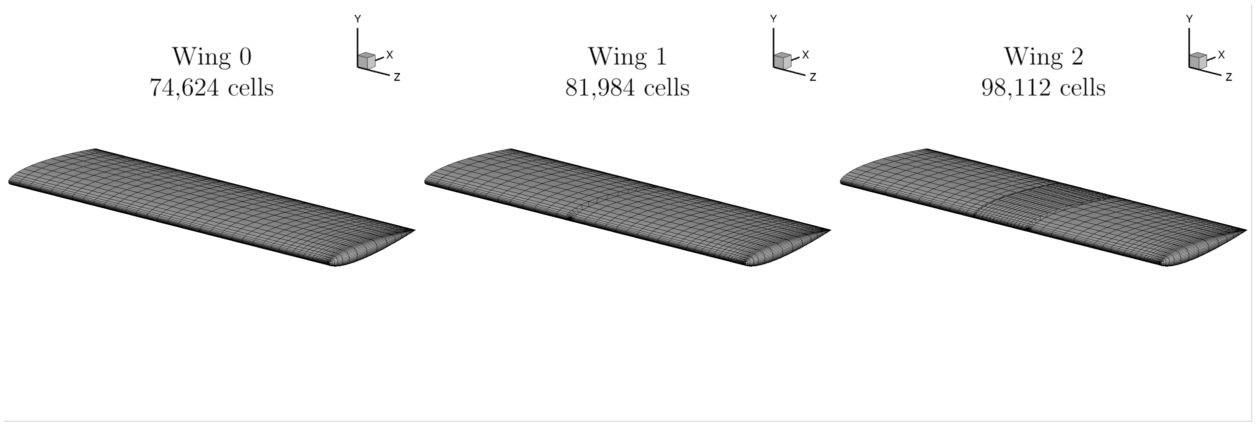

Figure 1: Surface meshes for debugging zipper code.

For each case, we computed 4 derivatives in total for two objective functions, CL and CD, and two design variables, angle of attack and span. For each grid, we computed adjoint and complex-step derivatives with three different versions of the code.

1. Default - main branch of ADflow (no changes)
2. Zipper order correction - commit c4b1e38
3. Zipper code commented out - commit 11ca464

In all but one case, the flow solutions and adjoint solutions were converged with `L2convergence=1e-12` and `adjointL2convergence=1e-12`. The exception is noted in the spreadsheet (link given below). We obtained solutions for both the Euler and RANS equations. Admittedly, the grids are not suitable for accurate RANS calculations, but we were able to obtain consistent converged results, which are sufficient for the stated purpose.

4 Results

We ran the analyses on Flux using 4 processors. The flow and adjoint solutions with real arithmetic require 30–60 seconds apiece (using ANK and NK solvers). The flow solutions with complex arithmetic require 10-40 minutes with D3ADI (depending on mesh and solution type). The results are tabulated in a google spreadsheet which can be found here. For reference, we included the spreadsheet in the last page of this document, however, it might become outdated as we run more cases.

We explain a few important trends in the results. First of all, we assume that the rest of ADflow is differentiated correctly. This means that the case with no zipper mesh, i.e. wing0, is expected to yield exact gradients. Furthermore, we expect the changes in the code to not affect these results. The results for wing0 (rows 10–12 for Euler and 20–22 for RANS) show these expected trends. The flow and adjoint systems were converged to 1e-12, and we are getting a relative error of 1e-10 for the lift and 1e-9 on the drag gradients.

We think that the error in the drag gradients are higher than the lift gradients because the actual drag value is lower in terms of absolute magnitude, and therefore the error is due to roundoff errors. Checking the cases at higher alpha values can validate this hypothesis. Bottom line for the no-zipper case is that the rest of the code and the test case setup seems to be correct. For the cases that contain zippers, *correct* gradients are expected to yield a relative error to these levels, i.e. 1e-10 for the lift and 1e-9 for the drag.

The Euler results (rows 9–18) show that the call order fix of the surface integration codes fixed the bug and the gradients match to the expected levels. The original code definitely has bugs, and the error caused by these seem to go away with the call-order fix. Furthermore, commenting out the zipper calls also yield similar error levels. Therefore, the Euler results suggest that the bug is fixed with the call-order fix, and rest of the code seems to be okay.

RANS cases show similar trends, except for a few outliers. In particular, the case with a single zipper (wing1) with RANS, seem to still have a relatively high error in the gradients, despite the fixes. Nick noted that the case with the call-order fix had a partially converged adjoint solution, which would explain the relative errors for this case. However, the next case where we commented out all the zipper calls had a fully converged adjoint solution, but the relative error still seems to be large in the drag gradients. Even though there is an improvement with the fixes, this case seem to have some other problem affecting the gradients.

5 Discussion

Based purely on the Euler results, we initially concluded that the call-order fix solved the problem, and the gradients match to the expected levels of accuracy. However, the few outlier cases in the RANS result shows that there might be other sources of errors that are still not fixed. There are a few possible explanations for these outliers.

First explanation is that the call-order modification fixed the bug, and the outlier errors are caused by small lift and drag values. We know that this is not a satisfying explanation, however Anil also saw similar results when comparing cases that have very small lift and drag values.

Another explanation is that the call-order modification fixed the bug completely for Euler cases, but RANS cases still contain bugs that introduce errors. We don't know why this may be, but the Euler results seem to yield correct gradients after the call-order fix, while some RANS cases have errors in the gradients. We think that this scenario is unlikely since some RANS results yield correct gradients with the call-order fix.

In conclusion, we believe that the outlier RANS cases have the relatively high errors due to partially converged adjoint solutions, and the call-order modification fixed the zipper mesh bug. In this case, we need to further validate that the modification actually fixed the bug by re-running the outlier cases and possibly by using other test cases.

Aside from the accuracy issue, it seems clear that there is another problem with the adjoint stalling in some cases. Ping noted that the adjoint solver stalls when he used zipper meshes. Nick also saw a similar behavior on the previously noted wing1 case with the zipper call-order fix. Therefore, we think that there is another bug that is affecting the adjoint solver performance, and causing it to stall. Perhaps the two problems are related in some way.

6 Possible Next Steps

There are a number of possibilities on what to do next. We can:

1. re-run the outlier RANS cases and make sure the adjoint solutions converge,
2. increase the angle of attack so that we get larger lift and drag values,
3. create RANS meshes with correct off-wall spacings and run the same tests,
4. investigate the effects of parallelization and having different number of processors for these tests,
5. refine the meshes and see how resolution effects these, and
6. build a wing-body series of meshes where we would have one multiblock version with no zippers, one wing/collar and fuse version with a single zipper, and a wing+collar+fuse version with two zippers.

It is not clear to us which step (or steps) would be the best. What are your suggestions? What results do we need to say that we fixed this bug and close the case? If the result is not fixed, where should we look next?

Geometry Descriptions

All wing geometries are based on the NACA 0012 wing (chord=1, AR=6). The wing is extruded to 3 m off-wall and a background mesh is used.

wing0 is a continuous multi-block wing mesh.

wing1 consists of two meshes that overlap at half-span.

wing2 consists of two non-overlapping quarter-span meshes with an overlapping collar mesh in between them.

					Alpha Derivatives						**Span Derivatives**						
			Actual function values		**CL**			**CD**			**CL**			**CD**			
Case	**Version**	**Geometry**	**CL**	**CD**	**AD**	**CS**	**Rel diff**	**AD**	**CS**	**Rel diff**	**AD**	**CS**	**Rel diff**	**AD**	**CS**	**Rel diff**	**Notes**
Euler																	
1	Default	wing0	1.42599762E-02	9.09509413E-04	7.76399893E-03	7.76399893E-03	1.34E-10	5.44148117E-04	5.44148117E-04	1.04E-09	5.25528718E-02	5.25528718E-02	1.78E-10	3.75366909E-03	3.75366909E-03	1.31E-09	
2	Zipper order	wing0	1.42599762E-02	9.09509413E-04	7.76399893E-03	7.76399893E-03	1.34E-10	5.44148117E-04	5.44148117E-04	1.04E-09	5.25528718E-02	5.25528718E-02	1.78E-10	3.75366909E-03	3.75366909E-03	1.31E-09	
3	No zipper	wing0	1.42599762E-02	9.09509413E-04	7.76399893E-03	7.76399893E-03	1.34E-10	5.44148117E-04	5.44148117E-04	1.04E-09	5.25528718E-02	5.25528718E-02	1.78E-10	3.75366909E-03	3.75366909E-03	1.31E-09	
4	Default	wing1	1.39052421E-02	9.06756703E-04	7.38075174E-03	7.58086019E-03	2.71E-02	5.13573847E-04	5.20994110E-04	1.44E-02	4.91765790E-02	4.98559174E-02	1.38E-02	3.59684355E-03	3.64938949E-03	1.46E-02	
5	Zipper order	wing1	1.39052421E-02	9.06756703E-04	7.58086019E-03	7.58086019E-03	3.10E-11	5.20994110E-04	5.20994110E-04	3.44E-11	4.98559174E-02	4.98559174E-02	4.99E-11	3.64938949E-03	3.64938949E-03	1.54E-10	
6	No zipper	wing1	1.35378774E-02	8.85768995E-04	7.38111804E-03	7.38111804E-03	3.13E-11	5.07162122E-04	5.07162122E-04	2.58E-11	4.91765790E-02	4.91765790E-02	4.94E-11	3.59684355E-03	3.59684355E-03	1.48E-10	
7	Default	wing2	1.34604007E-02	8.86642099E-04	7.17997860E-03	7.36455917E-03	2.57E-02	4.87227002E-04	4.93156380E-04	1.22E-02	4.70831430E-02	4.78178548E-02	1.56E-02	3.36233766E-03	3.41564400E-03	1.59E-02	
8	Zipper order	wing2	1.34604007E-02	8.86642099E-04	7.36455917E-03	7.36455917E-03	3.01E-10	4.93156379E-04	4.93156380E-04	2.25E-09	4.78178548E-02	4.78178548E-02	1.51E-11	3.41564400E-03	3.41564400E-03	1.49E-11	
9	No zipper	wing2	1.31230493E-02	8.64941771E-04	7.18035734E-03	7.18035734E-03	3.01E-10	4.81339110E-04	4.81339111E-04	2.25E-09	4.70831430E-02	4.70831430E-02	1.47E-11	3.36233766E-03	3.36233766E-03	1.72E-11	
RANS																	
1	Default	wing0	1.41934548E-02	8.53064225E-04	7.63505549E-03	7.63505549E-03	3.79E-10	5.40558659E-04	5.40558660E-04	1.92E-09	5.19796265E-02	5.19796265E-02	1.41E-10	3.51885001E-03	3.51885001E-03	6.56E-10	
2	Zipper order	wing0	1.41934548E-02	8.53064225E-04	7.63505549E-03	7.63505549E-03	3.79E-10	5.40558659E-04	5.40558660E-04	1.92E-09	5.19796265E-02	5.19796265E-02	1.41E-10	3.51885001E-03	3.51885001E-03	6.56E-10	
3	No zipper	wing0	1.41934548E-02	8.53064225E-04	7.63505549E-03	7.63505549E-03	3.79E-10	5.40558659E-04	5.40558660E-04	1.92E-09	5.19796265E-02	5.19796265E-02	1.41E-10	3.51885001E-03	3.51885001E-03	6.56E-10	
4	Default	wing1	1.38569133E-02	8.50661103E-04	7.28487198E-03	7.48174840E-03	2.70E-02	5.10567790E-04	5.17992677E-04	1.45E-02	4.88652170E-02	4.95344431E-02	1.37E-02	3.37461975E-03	3.42684786E-03	1.55E-02	
5	Zipper order	wing1	1.38569133E-02	8.50661103E-04	7.48174833E-03	7.48174840E-03	8.92E-09	5.17993219E-04	5.17992677E-04	1.05E-06	4.95344428E-02	4.95344431E-02	5.31E-09	3.42684955E-03	3.42684786E-03	4.92E-07	Adjoint only converged to 1e-8 and then stalled for some reason.
6	No zipper	wing1	1.34910024E-02	8.30723966E-04	7.28521995E-03	7.28522001E-03	7.41E-09	5.04181440E-04	5.04181016E-04	8.40E-07	4.88652170E-02	4.88652172E-02	4.05E-09	3.37461975E-03	3.37461867E-03	3.20E-07	
7	Default	wing2	1.34178756E-02	8.27804729E-04	7.10463836E-03	7.28720203E-03	2.57E-02	4.84171272E-04	4.89982273E-04	1.20E-02	4.67934667E-02	4.75193271E-02	1.55E-02	3.13279836E-03	3.18381639E-03	1.63E-02	
8	Zipper order	wing2	1.34178756E-02	8.27804729E-04	7.28720202E-03	7.28720203E-03	7.82E-10	4.89982307E-04	4.89982273E-04	6.76E-08	4.75193271E-02	4.75193271E-02	2.21E-11	3.18381590E-03	3.18381639E-03	1.54E-07	
9	No zipper	wing2	1.30815921E-02	8.07727952E-04	7.10498877E-03	7.10498877E-03	1.71E-10	4.78302019E-04	4.78302018E-04	1.97E-09	4.67934667E-02	4.67934667E-02	1.62E-11	3.13279836E-03	3.13279836E-03	2.59E-11	

Legend

🟥 Poor accuracy (>1e-2)

🟨 Low accuraccy (>1e-7)