Date: 8th June 2020

AP-266 Homework 03

Reynaldo Lima

Contents

1 Complex Step Test

Similarly to finite difference test, we can write:

$$\vec{y}(\vec{x}+\vec{\Delta x}) = \vec{y}(\vec{x}) + grad^T(\vec{y}).\vec{\Delta x}.$$

We can extend this definition by doing $\vec{\Delta x} \leftarrow i\vec{\Delta x}$. Therefore, with $J[y]$ defined as the jacobian of the function:

$$\text{Im}\Big\{\vec{y}(\vec{x}+i\vec{\Delta x})\Big\} = J[y]\vec{\Delta x} + O(h^3),$$

now, choosing Δx to be our seed with a h step in this direction, we have:

$$\frac{\text{Im}\Big\{\vec{y}(\vec{x}+ih\dot{\vec{x}})\Big\}}{h} = J[y]\dot{\vec{x}} = \dot{\vec{y}}(x)_{AD},$$

making possible to use this last equality as a complex step test to algorithmic differentiation. It must be clear that we approximated the term $O(h^3)$ to zero.

There are advantages regarding this test rather then finite difference test, *e.g.* we can choose such a small h as we want. From previous classes and homeworks, we already know that the complex step method does not depend on an "optimum" value of h. Furthermore, this method provides a lesser error and needs half as much function evaluations. Nonetheless, is important to notice that finite differences may be an easier to implement test.

2 Optimization with the lifting-line theory

2.1 Python Interface

The outputs for this part are obtained with the file *hm3_ test1.py.* The outputs for the requested values:

$$\bar{X} = \begin{bmatrix} -4.04950217 & 2.08471498 & 2.54897051 & 0.22408489 & -0.8082682 \\ -14.7594357 & 14.27401977 & -5.04278857 & 0.65624499 & 4.87195951 \\ 0.59555547 & -0.83782006 & 0.03601486 & -0.38566094 & 0.59191068 \end{bmatrix},$$

$$\bar{\Gamma} = \begin{bmatrix} 54.67416404 & -7.09550463 & 5.45562213 & 15.81322855 \end{bmatrix}^T,$$

$$\bar{\alpha_0} = \begin{bmatrix} -98.64728041 & 16.85547477 & 0. & -19.72945608 \end{bmatrix}^T,$$

$$\bar{c} = \begin{bmatrix} -8.3096734 & -2.40264652 & -1.53445338 & -2.88949739 \end{bmatrix}^T.$$

2.2 Solving the LLT problem

The outputs for this part are obtained with the file *hm3_ test2.py.*For the given inputs, we have:

$$\Gamma = \begin{bmatrix} 0.23818756 & 0.26183574 & 0.26183574 & 0.23818756 \end{bmatrix}^T,$$

and for this vale of Γ:

$$\vec{r} = \begin{bmatrix} 1.02271885e-19 & 5.96842124e-20 & 5.96842124e-20 & 1.02271635e-19 \end{bmatrix}^T,$$

$$S_{ref} = 8.0,$$

$$C_L = 0.5000232948751935,$$

$$C_D = 0.00810494164814668,$$

$$L = 2.000093179500774,$$

$$D = 0.03241976659258672.$$

2.3 Solving the adjoint problem

Observing that S_{ref}, as the area of reference, does not depends on the adjoint equation, we can skip this step. The outputs for this part are obtained with the file *hm3_ test3.py.* We have:

$$\vec{\psi_{C_D}} = \begin{bmatrix}3.08850024e+08 & -4.443870e+08 & -4.443870e+08 & 3.08850401e+08\end{bmatrix}^T,$$

$$\vec{\psi_{C_L}} = \begin{bmatrix}11843843.5849307 & -12008765.00909114\end{bmatrix} \ldots$$
$$\begin{bmatrix}-12008765.00909114 & 11843858.03199021\end{bmatrix}^T,$$

$$\vec{\psi_{S_{ref}}} = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T.$$

With those values, we can calculate the derivatives. In Figure 1 is explicated the respective values.

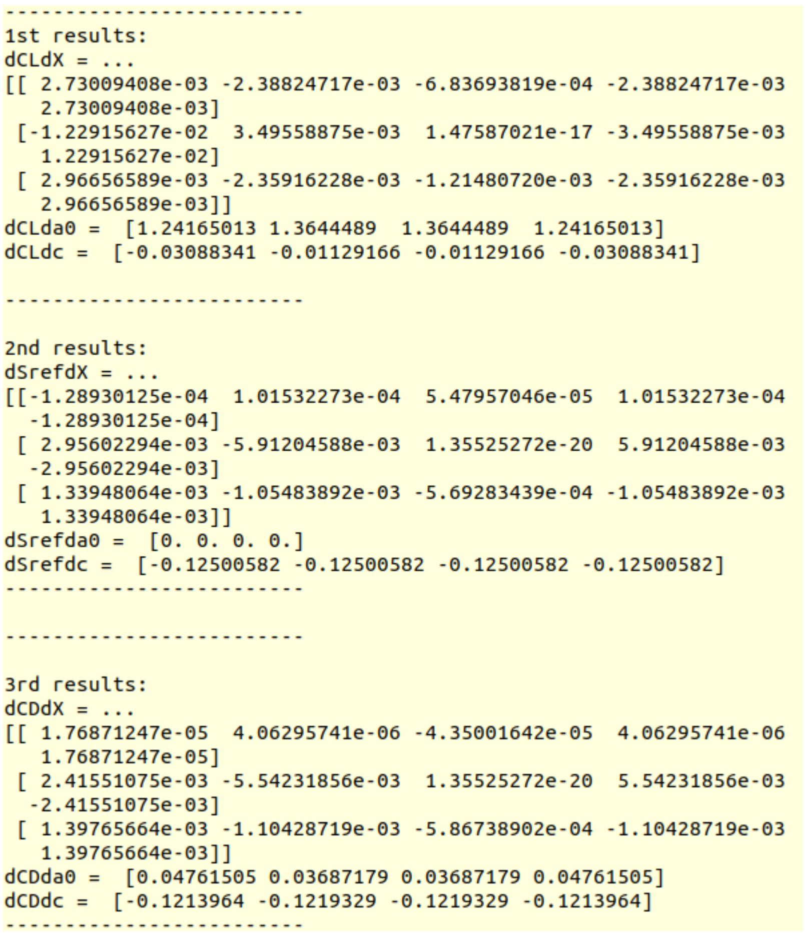
2.4 Twist optimization

The outputs for this part are obtained with the file *hm3_ test4.py.*

$$\bar{\alpha}_0^{(4)} = \begin{bmatrix}-0.01042016 & 0.00947173 & 0.00947173 & -0.01042016\end{bmatrix}^T,$$
$$C_D^{(4)} = 0.00795776014182388,$$
$$C_L^{(4)} = 0.5000004116952401.$$

2.5 Increasing the problem complexity

The outputs for this part are obtained with the file *hm3_ test5.py.* For 8 as number of vorteces:



```
------------------------
1st results:
dCLdX = ...
[[ 2.73009408e-03 -2.38824717e-03 -6.83693819e-04 -2.38824717e-03
   2.73009408e-03]
 [-1.22915627e-02  3.49558875e-03  1.47587021e-17 -3.49558875e-03
   1.22915627e-02]
 [ 2.96656589e-03 -2.35916228e-03 -1.21480720e-03 -2.35916228e-03
   2.96656589e-03]]
dCLda0 =  [1.24165013 1.3644489  1.3644489  1.24165013]
dCLdc =  [-0.03088341 -0.01129166 -0.01129166 -0.03088341]

------------------------

2nd results:
dSrefdX = ...
[[-1.28930125e-04  1.01532273e-04  5.47957046e-05  1.01532273e-04
  -1.28930125e-04]
 [ 2.95602294e-03 -5.91204588e-03  1.35525272e-20  5.91204588e-03
  -2.95602294e-03]
 [ 1.33948064e-03 -1.05483892e-03 -5.69283439e-04 -1.05483892e-03
   1.33948064e-03]]
dSrefda0 =  [0. 0. 0. 0.]
dSrefdc =  [-0.12500582 -0.12500582 -0.12500582 -0.12500582]
------------------------

------------------------

3rd results:
dCDdX = ...
[[ 1.76871247e-05  4.06295741e-06 -4.35001642e-05  4.06295741e-06
   1.76871247e-05]
 [ 2.41551075e-03 -5.54231856e-03  1.35525272e-20  5.54231856e-03
  -2.41551075e-03]
 [ 1.39765664e-03 -1.10428719e-03 -5.86738902e-04 -1.10428719e-03
   1.39765664e-03]]
dCDda0 =  [0.04761505 0.03687179 0.03687179 0.04761505]
dCDdc =  [-0.1213964 -0.1219329 -0.1219329 -0.1213964]
------------------------
```

Figure 1: Output of derivatives

$$\bar{\alpha_0^{(8)}} = \begin{bmatrix} -0.02273044 \\ -0.00048588 \\ 0.01152216 \\ 0.01684966 \\ 0.01684966 \\ 0.01152216 \\ -0.00048588 \\ -0.02273044 \end{bmatrix},$$

$$C_D^{(8)} = 0.008841957059376057,$$

$$C_L^{(8)} = 0.5000003787847209,$$

while, for 16:

$$\bar{\alpha_0^{(16)}} = \begin{bmatrix} -0.03436206 \\ -0.01741789 \\ -0.0041952 \\ 0.00529895 \\ 0.01205055 \\ 0.01669879 \\ 0.01962316 \\ 0.02103707 \\ 0.02103707 \\ 0.01962316 \\ 0.01669879 \\ 0.01205055 \\ 0.00529895 \\ -0.0041952 \\ -0.01741789 \\ -0.03436206 \end{bmatrix},$$

$$C_D^{(16)} = 0.009362418901281054,$$

$$C_L^{(16)} = 0.5000000012870989.$$

2.6 Comparing with analytical solutions

The outputs for this part are obtained with the file *hm3_test6.py*. The comparison between the elliptical expected value and the numerical solution for different number of vorteces is shown in Figure 2. We can see that there is a tendency for approximating the expected curve. This is better illustrated in Figure 3, when we put the C_D coefficient in

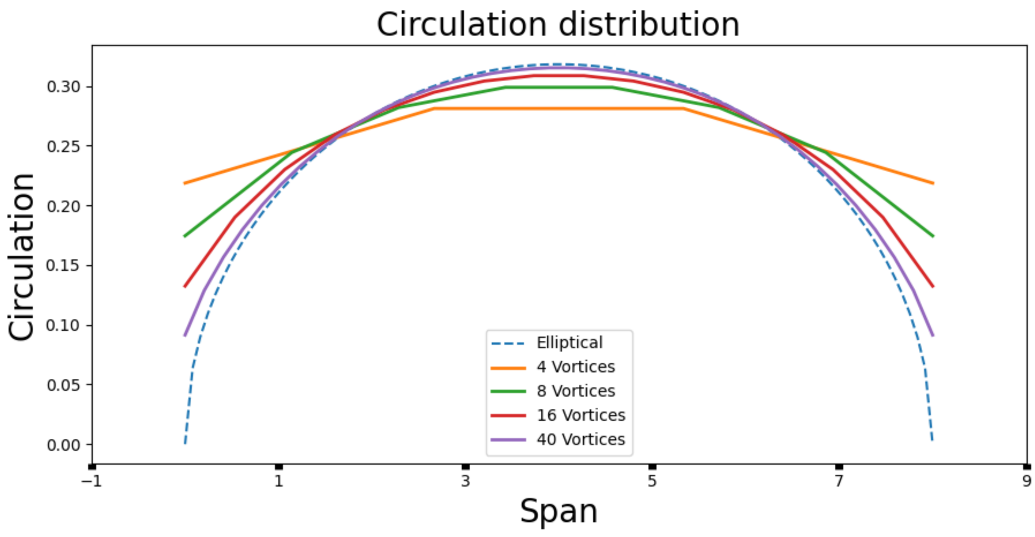


Figure 2: Comparison between different spans.

comparison. We can see that increasing the number of vortices there is a tendency to the expected value (approximately 0.0100). As for the requested number of function evaluations, it can be seen in Figure 4, that differently from other differentiation algorithms, the number of function evaluations changes really slowly relatively to the number of entries. This provides a really good option for programs that requires a large number of input variables with few output variables.

In Figure 5 we can see that for great values of N_{vortex}, the nummerical result is almost indistinguishble from the expected function. As for the coefficients:

$$C_D^{(100)} = 0.009860770691569896,$$

$$C_D^{(200)} = 0.009913062175066975,$$

$$C_D^{(Ideal)} \approx 0.009947183943,$$

where, for the ideal coefficient, $S_{ref} = 8.0$, $C_L = 0.5$, $b = 8.0$.

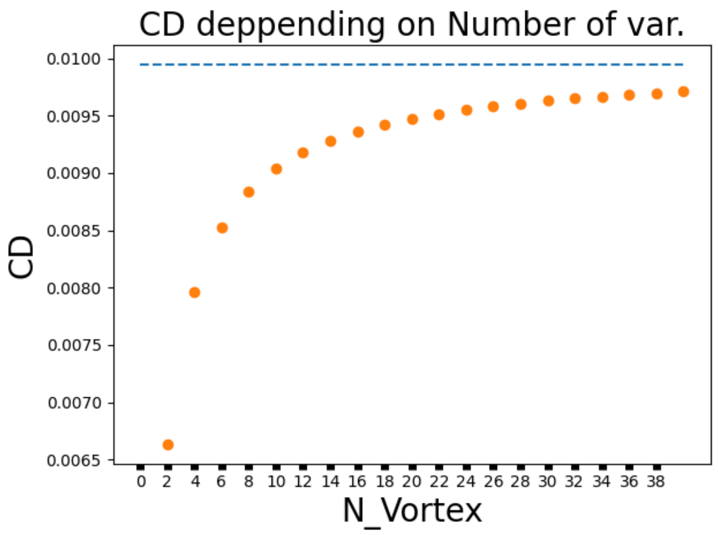


Figure 3: Comparison of CD for different number of vorteces.

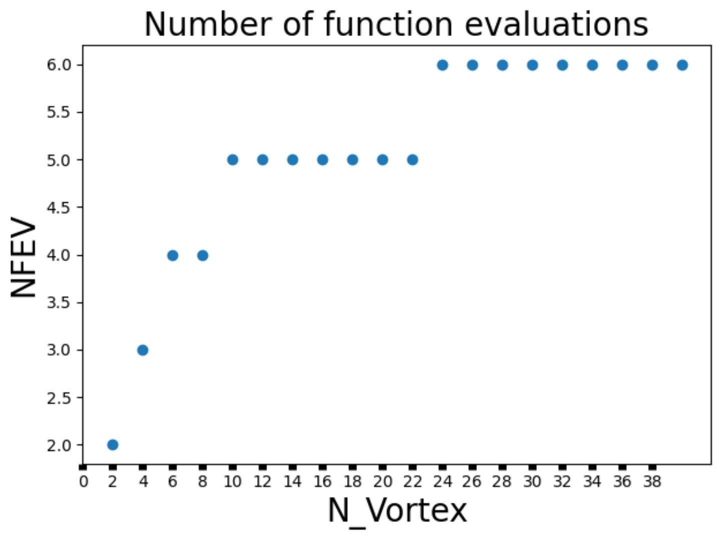


Figure 4: Comparison of NFEV for different number of vorteces.

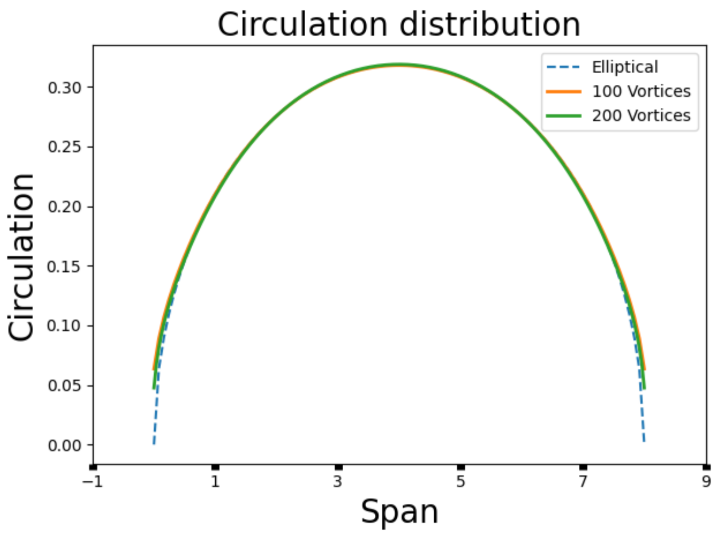


Figure 5: Convergence to expected value for 100 and 200 vortices.