Lober -light : Computation of lobes between two closed curves that can be locally tangent

Francois, Shane

July 25, 2003

1 Hypothesis

We consider two *closed* curves C_1 and C_2, both oriented in a counter-clockwise direction. We want to determine the area of the lobes (i.e. the area of $A\backslash B$ and $B\backslash A$) defined by these two curves.

In contrast with our previous approach, we do *not* require the two curves to have only transverse intersections. We allow non-tranverse intersection points and we also allow segments of curves that are common to C_1 and C_2.

2 Previous approaches

During previous numerical runs, we used an algorithm based on primary intersection points and regular intersection points to identify the boundary of each lobe and then compute the area of these lobes. These algorithms cannot be adapted to match the new problem. The tangency of the two curves creates both a confusion in the ordering of the interesection points and the postiion of the lobes as well as an enourmeous computational difficulty to extract the actual position of the intersection (determinant of a linear system $\to 0$).

The most reasonable patch used is to "inflate" or "deflate" the curves before computation in order to avoid the tangency. We transformed each point $\bar{x}$ of one of the curve to $\bar{x} + \epsilon(\bar{x} - \bar{x}_g)/\left\|(\bar{x} - \bar{x}_g)\right\|$, where $\bar{x}_g$ is the center of gravity of the curve. This can remove the presence of non-tranverse intersections for $\epsilon > \epsilon_0$, where the limit ϵ_0 is carefully chosen. However, the curve used in our problem contains many segment very close to each other in such a way that increasing ϵ above the minimum distance between the segments ϵ_1 creates new non-transverse intersections. In one example ϵ_1 was smaller than ϵ_0 forbidding the use of this trick to compute the lobe area. Moreover, the error on the area increases quickly with ϵ, so we developped the following exact method that is completely robust to non-transverse intersection of the curves.

3 One-dimensional Integrals for Area

Let us define A_1 and A_2 as the area enclosed by respectively C_1 and C_2. The area of each region can be computed as:

$$A_i = \iint_{A_i} dA = \frac{1}{2} \int_{C_i} ydx - xdy. \tag{1}$$

We proved this result before and it allows us to reduce the computation of the area of a complicated region to a one-dimensional integral over its boundary. Notice that the sign of the integral is to be reversed if the curves are oriented clockwise. Our hypothesis can be rewritten

$$\forall i : \int_{C_i} ydx - xdy \geq 0. \tag{2}$$

Let us define the complex function

$$f_{z_0}(z) = \frac{1}{x - x_0 + i(y - y_0)}. \tag{3}$$

The integral of $f(z)$ over a closed curve is equal to $2i\pi$ times the number of turns that the closed curve makes around the point $x_0 + iy_0$ (to prove that, use the fact that ($f(z)$ is analytic everywhere in the complex plane except at $x_0 + iy_0$ and use the residue theorem).

We define

$$J_i(x_0, y_0) = Im \left\{ \int_{C_i} \frac{dx + idy}{x - x_0 + i(y - y_0)} \right\}, \tag{4}$$

and

$$I_i(x_0, y_0) = \begin{cases} 1 & \text{if } J_i(x_0, y_0) < \pi \\ -1 & \text{if } J_i(x_0, y_0) \geq \pi \end{cases}, \tag{5}$$

and we notice that $I_i(x_0, y_0)$ is positive when the point (x_0, y_0) is contained in A_i and negative otherwise.

4 Area of Intersections

In order to extract $[A \backslash B]$ and $[B \backslash A]$ from the shape of the curves, we define the following quantities

$$Q_1 = \int_{C_1} I_2(x, y) \left(ydx - xdy \right) \int_{C_2} I_1(x, y) \left(ydx - xdy \right), \tag{6}$$

$$Q_2 = \int_{C_1} \frac{I_2(x, y) + 1}{2} \left(ydx - xdy \right) + \int_{C_2} \frac{I_1(x, y) + 1}{2} \left(ydx - xdy \right), \tag{7}$$

$$Q_3 = \int_{C_1} \frac{I_2(x, y) - 1}{-2} \left(ydx - xdy \right) + \int_{C_2} \frac{I_1(x, y) - 1}{-2} \left(ydx - xdy \right). \tag{8}$$

A quick look at the different paths reveals that

$$Q_1 = [A_1 \cup A_2] - [A_1 \cap A_2] = [A_1 \backslash A_2] + [A_2 \backslash A_1] \tag{9}$$

$$Q_2 = [A_1 \cup A_2], \tag{10}$$

$$Q_3 = [A_1 \cap A_2]. \tag{11}$$

Notice that the three equation above are not linearly independent and *any* of them can give us the expected results. However, we compute Q_1, Q_2 and Q_3 and use the redondency to minimize the error on the integrals and provide an approximation of the computational error. Since

$$[A_1 \backslash A_2] = [A_1 \cup A_2] - [A_2] = [A_1] - [A_1 \cap A_2], \tag{12}$$

and

$$[A_2 \backslash A_1] = [A_1 \cup A_2] - [A_1] = [A_2] - [A_1 \cap A_2], \tag{13}$$

we have

$$[A_1 \backslash A_2] = \frac{1}{2}(Q_2 - Q_3) + \frac{1}{2}([A_1] - [A_2]), \tag{14}$$

and

$$[A_2 \backslash A_1] = \frac{1}{2}(Q_2 - Q_3) + \frac{1}{2}([A_2] - [A_1]). \tag{15}$$

And using

$$[A_1] + [A_2] = [A_1 \cup A_2] + [A_1 \cap A_2], \tag{16}$$

we find

$$[A_1 \backslash A_2] = \frac{1}{2}Q_1 + \frac{1}{2}([A_1] - [A_2]), \tag{17}$$

and

$$[A_2 \backslash A_1] = \frac{1}{2}Q_2 + \frac{1}{2}([A_2] - [A_1]). \tag{18}$$

Our algorithm combines these two results and provides the following final answer

$$[A_1 \backslash A_2] = \frac{1}{2}(A_1 - A_2) + \frac{1}{4}(Q_1 + Q_2 - Q_3), \tag{19}$$

$$[A_2 \backslash A_1] = \frac{1}{2}(A_2 - A_1) + \frac{1}{4}(Q_1 + Q_2 - Q_3), \tag{20}$$

$$\delta[A_1 \backslash A_2] = \delta[A_2 \backslash A_1] = \frac{1}{4}\left|Q_2 - Q_3 - Q_1\right|. \tag{21}$$

5 Implementation

This algorithm is the *-light* mod of lober. The syntax is

```
lober -light <c1> <c2> <rslt> [ -DENS <nPass> <nDens> ]
```

where `<c1>` and `<c2>` are the names of the files containing respectively the curves C_1 and C_2 and `<rslt>` is the file to be created by lober to output the results. The optional arguments of the built-in densifier (`-DENS <nPass> <nDens>`) are described in the next section. The curve are stored in the files in a *Tecplot* ASCII format, i.e.

```
VARIABLES=''x''''y''
ZONE T=''the curve C1''
0.2    0.4
0.23   0.45
0.35   0.35
...
```

The output file will contain one line with 4 numbers: the area of the lobes inside, the area of the lobes outside and the relative error on these two values. The output file is usefull to automate the use of lober -light, for example with a `sh` script:

```
#!/bin/sh

FILENAME_A="separ."
FILENAME_B=""
FILENAME_I=0
FILENAME_F=5

OPT="-DENS 2 10"
RSLTF="lobes.out"

I=$FILENAME_I

STOP="no"
if [ -f $RSLTF ]; then
    rm -rf $RSLTF
fi
touch lobes.out

while [ "$STOP" = "no" ]; do
    STOP2="no"
    J=‘expr $I + 1‘
    while [ "$STOP2" = "no" ]; do
################################
# RUN LOBER $I, $J ############
```

```
##############################
F1=$FILENAME_A$I$FILENAME_B
F2=$FILENAME_A$J$FILENAME_B
CMD="lober -light "$F1" "$F2" lobe.out "$OPT
echo $CMD
$CMD
mv $RSLTF lobes.tmp
( cat lobes.tmp ; echo -n $I"   "$J ; cat lobe.out ) > $RSLTF
rm lobes.tmp
##############################
J=‘expr $J + 1‘
if [ $J -gt $FILENAME_F ]; then
    STOP2="yes"
fi
    done

    I=‘expr $I + 1‘
    if [ $I -ge $FILENAME_F ]; then
STOP="yes"
    fi
done
```

6 Built-In Densifier

There is a built-in densifier in the light version of lober. It adds points on the curves close to their intersections. To activate the densifier, add the paramters `-DENS <nPass> <nDens>` at the end of the command line. The arguments `<nPass>` and `<nDens>` give respectively the number of passes to be performed and the number of points to add near each intersection at each pass.

The extra precision is always $i_r = {n_{dens}}^{n_{pass}}$. In other words, the precision is the same with $(n_{pass} = 1, n_{dens} = 1000)$ than with $(n_{pass} = 3, n_{dens} = 10)$. For a constant i_r, the value of the two parameters n_{dens} and n_{pass} should minimize the computational time. Small n_{pass} means that fewer steps are necessary to densify the curve and can reduce the comptational time. However, small n_{pass} usually implies large n_{dens} to maintain a constant i_r. Since the extra length of the curve is $n_{dens}n_{pass}$, the number of points increases rapidly if n_{pass} is too small and lengthens the computation. So there is an optimal n_{pass} that minimizes computation time.

Note that I usually run `-DENS 1 100` or `-DENS 1 10` and check afterwards to see that `-DENS 1 1000` does not change the final result too much. In most example $n_{pass} = 1$ is close enough to the minimum. $n_{pass} = 0$ speeds up the computation but is a risk. For simple curves, $n_{pass} = 0$ is suitable.