Team project: Incompressible potential flow around an airfoil

UE 322.061 Num1

SS 2021

The task of this project is to compute numerically the steady two-dimensional irrotational incompressible (Ma$<<$ 1) flow over a half-cylindrical bump, visualize its streamlines and compute the total pressure force acting on the bump.

1 Problem formulation

The flow is governed by the Cauchy–Riemann equations, namely the irrotationality

$$\nabla \times \vec{u} \equiv \partial_x v - \partial_y u = 0 \tag{1a}$$

and incompressibility

$$\nabla \cdot \vec{u} \equiv \partial_x u + \partial_y v = 0, \tag{1b}$$

together with the no-penetration boundary condition

$$\begin{aligned} \vec{u} \cdot \vec{n} = 0 \qquad & \text{on } \Gamma_\mathrm{C} \cup \Gamma_\mathrm{F} \\ \Gamma_\mathrm{C} : \sqrt{x^2 + y^2} = r_\mathrm{c}, \quad & \Gamma_\mathrm{F} : y = 0 \end{aligned} \tag{1c}$$

and the far-field condition

$$\vec{u} \to (1, 0) \;\; \text{as } x^2 + y^2 \to \infty.$$

One option to impose boundary conditions at infinity is to transform coordinates using the tangent function. We will however restrict ourselves to a finite domain Ω and approximate the far-field condition by imposing a Dirichlet BC on the outer boundary

$$\vec{u} = (1, 0) \;\; \text{at } \Gamma_o : \sqrt{x^2 + y^2} = r_\mathrm{o}. \tag{1d}$$

1.a) Determine the type of the system (1). Comment if and where we have to impose initial and boundary conditions.
[1 Point]

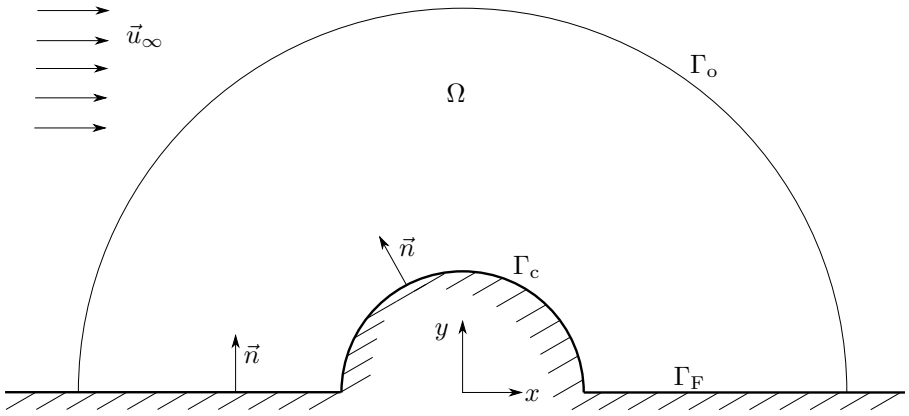

Figure 1: Sketch of the computational domain

1.b) The velocity field satisfying (1) is potential, which means there exists a velocity potential ϕ such that

$$\vec{u} = \nabla\phi \tag{2}$$

Show that using (2) the system (1a,1b) reduces to an equivalent Laplace equation

$$\nabla^2\phi = 0 \tag{3}$$

[1 Point]

1.c) Confirm that the type of (3) is the same as the type of (1)
[1 Point]

1.d) Confirm that the boundary conditions

$$\frac{\partial\phi}{\partial r} = 0 \text{ at } \Gamma_\mathrm{c} \qquad \frac{\partial\phi}{\partial\theta} = 0 \text{ at } \Gamma_\mathrm{F} \tag{4a}$$

$$\phi = x \text{ at } \Gamma_\mathrm{o} \tag{4b}$$

are equivalent to (1c,1d).
[1 Point]

2 Transformation of coordinates

We can simplify the mesh generation and implementation of the no-penetration boundary condition by solving (1) in polar coordinates

$$\theta = \arctan\frac{y}{x} \qquad r = \sqrt{x^2+y^2} \tag{5a}$$

$$x = r\cos\theta \qquad y = r\sin\theta. \tag{5b}$$

The transformed computational domain is then given by the rectangle $\Omega = \langle 0, \pi\rangle \times \langle r_\mathrm{c}, r_\mathrm{o}\rangle$.

2.a) Find the Jacobian of the coordinate transformation

$$\mathbf{J} = \begin{pmatrix} \frac{\partial x}{\partial\theta} & \frac{\partial y}{\partial\theta} \\ \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} \end{pmatrix} \tag{6}$$

[1 Point]

2.b) Express the inverse transformation of partial derivatives

$$\begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} = \mathbf{J}^{-1} \begin{pmatrix} \frac{\partial}{\partial\theta} \\ \frac{\partial}{\partial r} \end{pmatrix} \tag{7}$$

(*i.e.* find $\mathbf{J}^{-1}$).
[1 Point]

2.c) Check that the inverse transformation (7) satisfies the chain rule

$$\frac{\partial}{\partial x} = \frac{\partial\theta}{\partial x}\frac{\partial}{\partial\theta} + \frac{\partial r}{\partial x}\frac{\partial}{\partial r} \tag{8a}$$

$$\frac{\partial}{\partial y} = \frac{\partial\theta}{\partial y}\frac{\partial}{\partial\theta} + \frac{\partial r}{\partial y}\frac{\partial}{\partial r} \tag{8b}$$

[1 Point]

2.d) Find the expressions to compute the Cartesian second partial derivatives $\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\right)$ and $\frac{\partial}{\partial y}\left(\frac{\partial}{\partial y}\right)$ from the polar partial derivatives $\frac{\partial^2}{\partial\theta^2}$, $\frac{\partial^2}{\partial r^2}$, $\frac{\partial^2}{\partial\theta\partial r}$, $\frac{\partial}{\partial\theta}$, $\frac{\partial}{\partial r}$.
[2 Points]

2.e) Let us introduce a position vector

$$\vec{x} := x\vec{e}_x + y\vec{e}_y \equiv r\cos\theta\vec{e}_x + r\sin\theta\vec{e}_y, \tag{9}$$

where the Cartesian basis vectors $\vec{e}_x$ and $\vec{e}_y$ are unit vectors aligned with the Cartesian coordinate axes. Express the polar basis vectors

$$\vec{e}_r = \frac{\partial\vec{x}}{\partial r}\Big/\left\|\frac{\partial\vec{x}}{\partial r}\right\|_2 \qquad\qquad \vec{e}_\theta = \frac{\partial\vec{x}}{\partial\theta}\Big/\left\|\frac{\partial\vec{x}}{\partial\theta}\right\|_2 \tag{10}$$

in terms of the Cartesian ones.
[1 Point]

2.f) Invert the system (10) and express $\vec{e}_x$ and $\vec{e}_y$ in terms of $\vec{e}_r$ and $\vec{e}_\theta$.
[2 Points]

2.g) Using the result of the previous question, show that the transformation of vectors from Cartesian to polar basis reads

$$\vec{F} \equiv F_x\vec{e}_x + F_y\vec{e}_y = (F_x\cos\theta + F_y\sin\theta)\vec{e}_r + (-F_x\sin\theta + F_y\cos\theta)\vec{e}_\theta \tag{11}$$

[1 Point]

2.h) Using the results of the previous questions, show that the gradient in polar coordinates is given by

$$\nabla = \vec{e}_r\partial_r + \frac{\vec{e}_\theta}{r}\partial_\theta \tag{12}$$

[2 Points]

2.i) Show that the Laplace operator in polar coordinates reads

$$\nabla^2 \equiv \frac{1}{r}\frac{\partial}{\partial r} + \frac{\partial^2}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2}{\partial\theta^2} \tag{13}$$

[1 Point]

2.j) Write a MATLAB function `[u_c,v_c,x,y] = VecPol2Cart(u_p,v_p,theta,r)` to convert a discrete polar vector field to Cartesian coordinates

$$u_p(r,\theta)_i\vec{e}_r + v_p(r,\theta)_i\vec{e}_\theta = u_c(x,y)_i\vec{e}_x + v_c(x,y)_i\vec{e}_y.$$

[1 Point]

3 Discretization

Let us discretize (3) on a two-dimensional polar grid with I and J number of points in r and θ direction respectively, such that $\phi_{i,j} = \phi(\theta_{i,j}, r_{i,j})$. The coordinates of the grid points are given by

$$\boldsymbol{\theta} = \begin{pmatrix} 0 & \cdots & \pi \\ \vdots & & \vdots \\ 0 & \cdots & \pi \end{pmatrix} \qquad\qquad \mathbf{r} = \begin{pmatrix} r_\mathrm{c} & \cdots & r_\mathrm{c} \\ \vdots & & \vdots \\ r_\mathrm{o} & \cdots & r_\mathrm{o} \end{pmatrix}.$$

We will approximate partial derivatives with the standard second-order centered finite differences

$$\left.\frac{\partial\phi}{\partial\theta}\right|_{i,j} \approx \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta\theta} \qquad\qquad \left.\frac{\partial\phi}{\partial r}\right|_{i,j} \approx \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta r} \tag{14a}$$

$$\left.\frac{\partial^2\phi}{\partial\theta^2}\right|_{i,j} \approx \frac{\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}}{\Delta\theta^2} \qquad\qquad \left.\frac{\partial^2\phi}{\partial r^2}\right|_{i,j} \approx \frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{\Delta r^2} \tag{14b}$$

3.a) Provide a MATLAB code `[r,theta,dr,dtheta] = GenerateMesh(rc,ro,I,J)` to generate the matrices of grid point coordinates $\mathbf{r}, \boldsymbol{\theta}$ and the grid spacings $\Delta r, \Delta\theta$.
[1 Point]

3.b) Provide a MATLAB code `[ii,io,il,ir] = BoundaryIndices(r,theta)` to find the indices i_i, i_o, i_l and i_r of grid points belonging to the boundaries $r = r_\mathrm{c}, r = r_\mathrm{o}, \theta = 0$ and $\theta = \pi$ respectively.
[1 Point]

3.c) As usual, we will flatten $\phi_{i,j}$ into a column vector ϕ_m where $m = i + (j-1)I$ enumerates all the grid points. Re-write (14) using the linear indexing m and produce a MATLAB code `[Dr,Dth,Dr2,Dth2] = PartialDerivatives(r,th)` to create the matrices for discretization of partial derivatives with the second-order centered finite-differences.
[1 Point]

3.d) Write a MATLAB function `A = PolarLaplacian(r,Dr,Dr2,Dth2)` to compute the discretization matrix for the Laplace operator in polar coordinates
[1 Point]

3.e) Write a MATLAB function `[A,b] = DirichletBC(A,b,alpha,iD)` to impose a non-uniform Dirichlet boundary condition `phi(iD)=alpha` in a given linear system

$$\mathbf{A}\vec{\phi} = \vec{b}$$

at the grid points with indices `iD`. The suggested inputs are:

- `A` – the matrix $\mathbf{A}$ corresponding to the finite-difference approximation of the homogeneous part of the PDE to be solved.
- `b` – the vector of the discrete right-hand side $\vec{b}$ of the PDE to be solved (zero vector for a homogeneous PDE).
- `alpha` – vector of the Dirichlet values corresponding to the indices `iD`
- `iD` – indices of those grid points where the Dirichlet BC is to be imposed

[1 Point]

3.f) Write a MATLAB function `[A,b] = NeumannBC(A,b,beta,iN,dn,ig)` to impose a Neumann boundary condition

$$\frac{\partial\phi}{\partial n} = \beta, \qquad \text{where } \frac{\partial}{\partial n} = n_r\partial_r + n_\theta\partial_\theta,$$

at the boundary grid points with indices `iN`. Use the ghost-point method

$$\frac{\phi_{m+\mathtt{ig}} - \phi_{m-\mathtt{ig}}}{\mathrm{dn}} = \beta, \qquad \text{where } \mathtt{dn} = n_r\Delta r + n_\theta\Delta\theta$$

and `ig` is the (theoretical) index of the ghost points relative to the boundary points `iN`.
Hint: Don't forget to erase those entries of $\mathbf{A}$ which correspond to the opposite boundary.
[2 Points]

4 Computation of the velocity field

4.a) Compute the velocity potential ϕ for $r_\mathrm{c} = 0.5, r_\mathrm{o} = 3, I = J = 31$ using the template function `[phi,x,y] = ComputePotential(rc,ro,I,J)`. Compare the contours of ϕ to the analytical solution ϕ_a returned by `phi_a = AnalyticalPotential(rc,x,y)`.
[1 Point]

4.b) The centered finite-difference matrices for first derivatives `Dr` and `Dth` are not valid on some boundaries. Write a MATLAB function `[Dr,Dth] = BoundaryDerivatives(r,th,Dr,Dth)` to modify `Dr` and `Dth` such that on those boundaries where the derivatives cannot be computed with centered finite differences they will be computed with first-order one-sided finite differences. The rows corresponding to interior grid points should remain unaffected.
[1 Point]

4.c) Write a MATLAB function `[uc,vc,up,vp] = ComputeVelocity(phi,r,th)` to compute the polar velocity components from the velocity potential

$$\begin{pmatrix}\mathtt{up}\\ \mathtt{vp}\end{pmatrix} = \nabla\phi \equiv \begin{pmatrix}\partial_r\\ \frac{1}{r}\partial_\theta\end{pmatrix}\phi$$

and convert them to Cartesian velocity components using `VecPol2Cart`. Compare the Cartesian velocity field to the analytical solution returned by `[ua,va] = AnalyticalVelocity(rc,x,y)`. Plot both vector fields using `quiver`.
Hint: Use the matrices `Dr` and `Dth`.
[1 Point]

5 Pressure force

5.a) Write a MATLAB function `[p] = ComputePressure(u,v)` to compute the pressure filed from the velocity components, using the Bernouli equation

$$p = \frac{1}{2} - \frac{1}{2}\sqrt{u^2 + v^2} \tag{15}$$

Note: The reference pressure $p_0 = 1/2$ can be selected arbitrarily. The value selected here is such that the far field static pressure $p_\infty = 0$.
[1 Point]

5.b) Plot the contours of the pressure field.
[1 Point]

5.c) Write a function `[Fx,Fy] = ComputeForce(p,r,th)` to integrate the pressure along the surface of the half-cylindrical bump in order to obtain the total pressure force acting on the bump per unit spanwise length

$$\vec{F} = \int_0^\pi p(\theta, r = r_\mathrm{c}) r_\mathrm{c} \mathrm{d}\theta \tag{16}$$

Use any numerical integration scheme that you want (*e.g.* left Riemann sum, right Riemann sum, midpoint rule or trapezoidal rule).
[2 Points]

5.d) Is the total pressure force in agreement with the theory of potential flows? Explain.
[2 Points]

6 Streamlines

Streamlines of a steady flow are in polar coordinates governed by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\theta\\ r\end{pmatrix} = \begin{pmatrix}u(\theta, r)\\ v(\theta, r)/r\end{pmatrix} \tag{17}$$

We want to compute a set of streamlines initiated from a set of N seeding points $[\theta_1, r_1], \ldots, [\theta_N, r_N]$. Thus we have to integrate an explicit system of $2N$ first-order ODEs.

Although several ODE solvers are available in MATLAB, we are so badass that we will implement our own version of the Dormand-Prince algorithm RK5(4)7M (Dormand and Prince, 1980, Table 2) known as `ode45`. It is an adaptive time step algorithm based on the 5^{th} order Runge-Kutta method with the Butcher tableau given in Table 1. Recall that a general Butcher tabelau as in Table 2 defines the Runge-Kutta method for $\dot{\vec{y}} = \vec{f}(t, \vec{y})$ as

$$\begin{aligned}
\vec{k}_1 &= \vec{f}(t_n, \vec{y}_n)\\
\vec{k}_2 &= \vec{f}(t_n + a_2\Delta t, \vec{y}_n + \Delta t(b_{11}\vec{k}_1))\\
&\vdots\\
\vec{k}_s &= \vec{f}(t_n + a_s\Delta t, \vec{y}_n + \Delta t(b_{s-1,1}\vec{k}_1 + \cdots + b_{s-1,s-1}\vec{k}_{s-1}))\\
\vec{y}_{n+1} &= \vec{y}_n + \Delta t \sum_{i=1}^{s} c_i \vec{k}_i.
\end{aligned} \tag{18}$$

The advantage of the Dormand-Prince pairs is that a one order lower result can be obtained just by changing the coefficients c_i without any additional evaluations of the function $\vec{f}$

$$\hat{\vec{y}}_{n+1} = \vec{y}_n + \Delta t \sum_{i=1}^{s-1} \hat{c}_i \vec{k}_i. \tag{19}$$

The difference between the lower and higher order result is a good and yet computationally cheap estimate of the discretization error at each time step. This estimate is used by `ode45` to adjust the time-step length Δt such that the discretization error is of the same order of magnitude at all steps. The difference can also be expressed directly without explicitly computing the lower order result

$$\vec{y}_{n+1} - \hat{\vec{y}}_{n+1} = \sum_{i=1}^{s} e_i \vec{k}_i. \tag{20}$$

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	-56/15	32/9				
8/9	19372/6561	-25360/2187	64448/6561	-212/729			
1	9017/3168	-355/33	46732/5247	49/176	-5103/18656		
1	35/384	0	500/1113	125/192	-2187/6784	11/84	
	5179/57600	0	7571/16695	393/640	-92097/339200	187/210	1/40

Table 1: Butcher tableau for Runge-Kutta 5 7M from Dormand and Prince (1980)

0					
a_2	b_{11}				
a_3	b_{21}	b_{22}			
$\vdots$			$\ddots$		
a_s	$b_{s-1,1}$	$b_{s-1,2}$	$\dots$	$b_{s-1,s-1}$	
	c_1	c_2	$\dots$	c_{s-1}	c_s

Table 2: General Butcher tableau in the notation of MATLAB's `ode45`

The coefficients $\hat{c}_i$ and e_i for RK5(4)7M are given in Table 3.

The standard ODE solvers are designed for general systems of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} f_1(t, y_1, y_2, \dots, y_N) \\ f_2(t, y_1, y_2, \dots, y_N) \\ \vdots \\ f_N(t, y_1, y_2, \dots, y_N) \end{pmatrix}. \tag{21}$$

We will be a bit more specific and write a solver aimed at computation of multiple trajectories in a steady vector field

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} r_1 \\ \theta_1 \\ \vdots \\ r_N \\ \theta_N \end{pmatrix} = \begin{pmatrix} f_u(\theta_1, r_1) \\ f_v(\theta_1, r_1) \\ \vdots \\ f_u(\theta_N, r_N) \\ f_v(\theta_N, r_N) \end{pmatrix} \tag{22}$$

where for polar coordinates we have $f_u = u$, $f_v = v/r$. We therefore want to construct the Runge-Kutta method as

$$\begin{aligned} \vec{k}_{1r} &= \vec{f}_u(\theta, r) \\ \vec{k}_{1\theta} &= \vec{f}_v(\theta, r) \\ &\vdots \\ \vec{k}_{s\theta} &= \vec{f}_v(\theta + \Delta t \sum_{i=1}^{s-1} b_{s-1,i}\vec{k}_{i\theta}, r + \Delta t \sum_{i=1}^{s-1} b_{s-1,i}\vec{k}_{ir}) \\ \vec{r}_{n+1} &= \vec{r}_n + \Delta t \sum_{i=1}^{s} c_i \vec{k}_{ir} \\ \vec{\theta}_{n+1} &= \vec{\theta}_n + \Delta t \sum_{i=1}^{s} c_i \vec{k}_{i\theta} \end{aligned} \tag{23}$$

6.a) Implement the 5th order 7M Runge-Kutta method in the template function `[R,TH] = myode45(fu,fv,r0,th0)`. Note that the coefficients from the Butcher tableau are already

i	1	2	3	4	5	6	7
$\hat{c}_i$	35/384	0	500/1113	125/192	-2187/6784	11/84	
e_i	71/57600	0	-71/16695	71/1920	-17253/339200	22/525	-1/40

Table 3: Coefficients for RK4 7M and for the error estimate RK5-RK4 7M of Dormand and Prince (1980)

defined.
[3 Points]

6.b) Compute streamlines initiated from $N = 10$ seeding points uniformly distributed along the line segment $x = -2, y \in \langle 0.1, 1.9 \rangle$. Use the template function `[strx,stry] = ComputeStreamlines(up,vp,r,th,x0,y0)`.
[1 Point]

6.c) Plot the streamlines using a solid line with markers. How does the time-step length Δt depend on the local velocity and the curvature of the streamlines? Can the computation of streamlines become a stiff problem? How would you avoid the problem from being stiff?
[1 Point]

7 Other graphs

7.a) Plot the distribution of pressure along the cylinder surface
[1 Point]

7.b) Plot the velocity profile along the line segment $x = 0, y \in \langle 0.5, 3 \rangle$
[1 Point]

References

J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *J. Comput. Appl. Math.*, 6(1): 19–26, 1980.