P. Koumoutsakos, W. Petersen, P. Hadjidoukas
ETH Zentrum, CLT E 13
CH-8092 Zürich

Fall semester 2017

Set 11 - Particle Methods and MPI

Issued: December 8, 2017
Hand in (optional): December 15, 2017 8:00am

Question 1: Roll-up of 1D Vortex Line

In this exercise you will look at the evolution of a vortex line in 2D, which corresponds to a solution of the Euler equations describing inviscid flows. The vortex line is represented with N point vortices so that the total vorticity is expressed as

$$\omega(\mathbf{x},t) = \sum_{n=0}^{N-1} \Gamma_n \delta(\mathbf{x}-\mathbf{x}_n), \tag{1}$$

where $\Gamma_n(t)$ and $\mathbf{x}_n(t)$ are the circulation and location of vortex n. The corresponding velocity field $\mathbf{u}=(u,v)$ is given by

$$\mathbf{u}(\mathbf{x},t) = \sum_{n=0}^{N-1} \mathbf{u}_n(\mathbf{x},t) \tag{2}$$

where

$$u_n(\mathbf{x},t) = \frac{1}{2\pi}\Gamma_n \frac{-(y-y_n)}{|\mathbf{x}-\mathbf{x}_n|^2}, \tag{3}$$

$$v_n(\mathbf{x},t) = \frac{1}{2\pi}\Gamma_n \frac{x-x_n}{|\mathbf{x}-\mathbf{x}_n|^2}. \tag{4}$$

The evolution of this system of vortices is then governed by

$$\dot{\Gamma}_n = 0, \tag{5}$$

$$\dot{\mathbf{x}}_n = \sum_{k\neq n} \mathbf{u}_k(\mathbf{x}_n,t), \tag{6}$$

i.e. the vortices of constant circulations move with velocity $\mathbf{u}(\mathbf{x},t)$ (excluding self-interaction).

a) (**3 points**) Consider two vortices of circulations Γ_1 and Γ_2 initially located at

$$\mathbf{x}_1(0) = (\Delta/2, 0), \tag{7}$$

$$\mathbf{x}_2(0) = (-\Delta/2, 0). \tag{8}$$

Solve equation (6) to find velocities $\dot{\mathbf{x}}_1(t)$ and $\dot{\mathbf{x}}_2(t)$ and describe the motion of vortices for two cases: (i) $\Gamma_1 = \Gamma_2 = \Gamma$, (ii) $\Gamma_1 = -\Gamma_2 = \Gamma$.

Figure 1: Wake of an airplane visualized by condensation behind the engines.

The remaining part of the exercise is to implement the model for an arbitrary number of vortices and solve the problem described below. The solution should resemble the wake of an airplane (see Figure 1) in one cross-section.

Initially, N particles (vortices) are arranged uniformly on the x-axis

$$x_n = -0.5 + (n + 0.5)h, \quad n = 0, ..., N - 1 \tag{9}$$

and have circulation

$$\Gamma_n = \gamma(x_n)h, \tag{10}$$

where $h = 1/N$ and

$$\gamma(x) = -\frac{\mathrm{d}}{\mathrm{d}x}\left[\sqrt{1 - \left(\frac{x}{0.5}\right)^2}\right]. \tag{11}$$

b) (**2 points**) Set up the MPI framework following TODO 1 in the provided skeleton code. The particles should be distributed equally among the processors.

c) (**3 points**) Initialize the position and circulation of each particle based on (9) and (10). Compute the total circulation

$$\Gamma_{\mathrm{tot}} = \sum_{n=0}^{N-1} \Gamma_n. \tag{12}$$

Follow TODO 2 in the provided skeleton code.

d) (**2 points**) Implement the forward Euler scheme to advance in time (TODO 3).

e) (**5 points**) Evaluate the velocity of particles according to (6), (3) and (4) using the multi-pass approach (TODO 4): P processors exchange particles P times. Let D_p be the particles assigned to processor p. At every pass, each processor sends particles D_p (`dstParticles` in the skeleton code) to one rank, receives from another (to `srcParticles`) and computes the interactions between them. Table 1 illustrates the data arrangement.

f) (**5 points**) Implement output routines that write the position, velocity and circulation of all particles to a csv-file with columns: `x, y, vx, vy, g`. Try two approaches: (i) master receives from others and writes all data, (ii) each processor writes only its own particles using MPI I/O. See TODO 5 in the skeleton code.

Run your code with $10\,000$ particles and plot the particle positions at times $t = 0.5$, $t = 1.0$ and $t = 2.0$. Set the time step for integration to $\Delta t = 10^{-3}$.

		process p				
		0	1	$\cdots$	P-2	P-1
	0	D_0	D_1	$\cdots$	D_{P-2}	D_{P-1}
	1	D_{P-1}	D_0	$\cdots$	D_{P-3}	D_{P-2}
pass q	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	P-2	D_2	D_3	$\cdots$	D_0	D_1
	P-1	D_1	D_2	$\cdots$	D_{P-1}	D_0

Table 1: Multi-pass approach to compute N^2 interactions.

For visualization, use the provided Python-script `plot` (run from folder containing csv-files) or another tool (e.g. Paraview). On Euler, you may need to load module `python`.

g) (Optional, **4 points**) Make weak and strong scaling plots. For the weak scaling, remember that the algorithm has complexity $\mathcal{O}(N^2)$ at one time step.