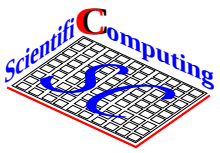
Institute of Scientific Computing
Technical University Braunschweig
Prof. Hermann G. Matthies, Ph.D.



We are now in the position to make larger and more complex simulations over a longer period of time, so it is about time, to deal in depth with visualization.

1 Visualization of Flow With Particle Trace and Streaklines

At the flows in our examples it is by interest, to analyze flows using particle traces. For that purpose we consider two methods:

1. **Particle trace:** The positions of *one* particle at *different*, successive time-steps are visualized.
2. **Streakline:** In regular short time intervals particles are injected in certain places of the flow and the positions of these *different* particles at *one* time-step are visualized.

The term steaklines comes from the observation of particles, which were brought into the flow by a *stroke of the brush*. A possibility to accomplish this in a physical experiment is to bring in small air blisters along a thin wire (which does not influence the flow).
For the calculation of such particle movements it is fundamental to be able to conclude from the position of a particle at time-step t_n to the particle position at time-step $t_{n+1} = t_n + \delta t$. This happens in two steps:

1. Find for a position $(x^{(n)}, y^{(n)})$ in the flow field the appropriate velocities $u^{(n)}$, $v^{(n)}$.
2. Determine the new position of the particle at time-step t_{n+1} with an Euler-step: $x^{(n+1)} = x^{(n)} + \delta t \cdot u^{(n)}, \quad y^{(n+1)} = y^{(n)} + \delta t \cdot v^{(n)}$.

Here the second step is elementary, but the first requires however some assertion.

1.1 Interpolation of Velocities in the „Staggered Grid“

One must approximate the velocities in the grid, because the velocities u and v are just given at fixed places in the „staggered grid“.
The position (x, y) of the particle and the velocities u_{ij}, v_{ij} in the „staggered grid“ are given:

$$u_{i,j} = u(x_i, y_j) \quad \text{mit} \quad x_i = i \cdot \delta x, \; y_j = (j - \frac{1}{2}) \cdot \delta y,$$

for $i = 0, \ldots, imax, \; j = 0, \ldots, jmax + 1$,

$$v_{i,j} = v(x_i, y_j) \quad \text{mit} \quad x_i = (i - \frac{1}{2}) \cdot \delta x, \; y_j = j \cdot \delta y,$$

for $i = 0, \ldots, imax+1,\ j = 0, \ldots, jmax$, and we want to calculate the velocites $u(x, y)$, $v(x, y)$ at the position (x, y).
At first we consider the calculation of the velocity $u(x, y)$. Therefore we had to determine the specific u-cell for a given position (x, y) (see figure 1 and 2).

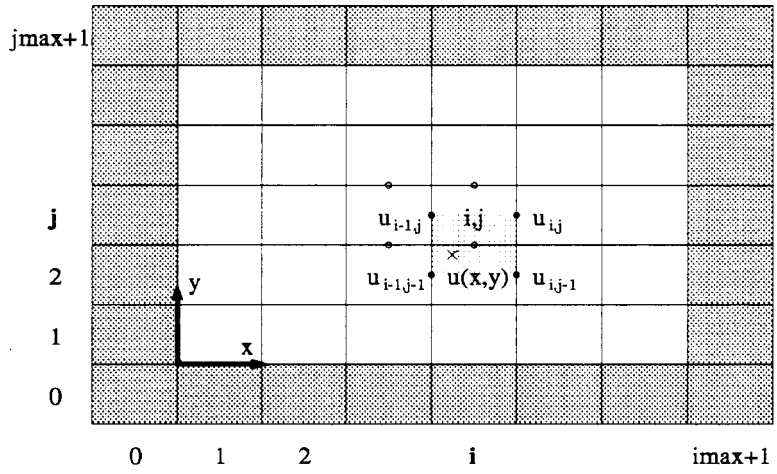


Figure 1: *u-cell (i, j) in the „staggered grid“*

This occurs with the help of the integral operations

$$i := (int)\left(\frac{x}{\delta x}\right) + 1, \quad j := (int)\left(\frac{y + \frac{\delta y}{2}}{\delta y}\right) + 1,$$

which determine the numbering of the u-cells, and the values

$$x_1 := (i-1) \cdot \delta x, \quad x_2 := i \cdot \delta x,$$

$$y_1 := ((j-1) - \frac{1}{2}) \cdot \delta y, \quad y_2 := (j - \frac{1}{2}) \cdot \delta y,$$

which determine the positions of the vertices of the u-cell, on which the four „staggered grid“-velocities

$$u_1 := u_{i-1,j-1},\ u_2 := u_{i,j-1},\ u_3 := u_{i-1,j},\ u_4 := u_{i,j}$$

are given.
Now one can aproximate the velocity $u(x, y)$ at position (x, y) by means of bilinear interpolation:

$$\begin{aligned} u(x,y) \quad := \quad & \frac{1}{\delta x \delta y}[(x_2 - x)(y_2 - y)u_1 + (x - x_1)(y_2 - y)u_2 \\ & +(x_2 - x)(y - y_1)u_3 + (x - x_1)(y - y_1)u_4]. \end{aligned}$$

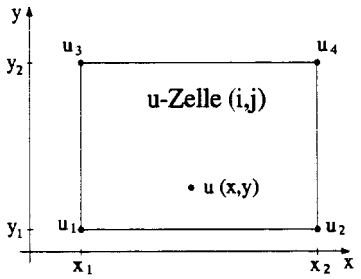


Figure 2: *u-cell* (i, j)

Similar one can calculate the velocity $v(x, y)$: The numbering of the v-cell which encloses the position (x, y) is

$$i := (int)\left(\frac{x + \frac{\delta x}{2}}{\delta x}\right) + 1, \quad j := (int)\left(\frac{y}{\delta y}\right) + 1,$$

the positions of the vertices with the v velocities of the „staggered grid“are

$$x_1 := ((i-1) - \frac{1}{2}) \cdot \delta x, \quad x_2 := (i - \frac{1}{2}) \cdot \delta x,$$
$$y_1 := (j-1) \cdot \delta y, \quad y_2 := j \cdot \delta y,$$

and the interpolation rule is

$$\begin{aligned} v(x,y) \quad := \quad & \frac{1}{\delta x \delta y}[(x_2 - x)(y_2 - y)v_1 + (x - x_1)(y_2 - y)v_2 \\ & +(x_2 - x)(y - y_1)v_3 + (x - x_1)(y - y_1)v_4], \end{aligned}$$

with the four „staggered grid“-velocites $v_1 := v_{i-1,j-1},\ v_2 := v_{i,j-1},\ v_3 := v_{i-1,j},\ v_4 := v_{i,j}$.

2 Tasks

Implement the two functions `particletrace` and `streaklines` which must be called regular in the time-loop of the main program and calculate the streamlines and the partice trace. The results are to be stored in each time-step t_n into two vectors, one for x-component and one for the y-component:

$$\texttt{part\_x} = (x_1,\, x_2 \ldots, x_N) \quad \text{und} \quad \texttt{part\_y} = (y_1,\, y_2 \ldots, y_N) \in I\!R^N,$$

N is the number of particles.
These functions should do the following:

1. `particletrace:` At the beginning of the simulation N particles are injected into the flow. The movement of these N particles is to be traced during the period of the simulation.

2. `streaklines:` All `delt_n` time steps N new particles are injected into the flow. At time-step t_n the positions of all particles in the flow are stored and traced.

If a particle leaves the domain $\omega = [0, imax\delta x] \times [0\cdot, jmax \cdot \delta y]$ (or colide with an obstacle), then this particle is deleted from the particle field. (such look-up and replacing operations can be executed in Matlab efficiently by using the `find` instruction.) For particle trace the following subroutines are to be implemented:

1. `[u, v] = interpolation(U, V, imax, jmax, delx, dely, x, y):`
   The velocities u and v are to be interpolated in the point (x, y) from the velocity fields U and V according to the calculated formulas.

2. `[part_x, part_y] = new_position(U, V, imax, jmax,`
   `delx, dely, delt, part_x, part_y):`
   In particle positions help in `part_x` and `part_y` of the time step t_n are moved by the Euler-step into the particle position of the time step t_{n+1}. This function is called in `particletrace` and `streaklines`.

3. `[part_x, part_y] = set_particles(N, ug, og, delx):`
   The initial positions of the particles are injected uniformly at the lefthand side of the flow (To perform an interpolation at the start correctly, the particles should not directly injected at the boundary, but somewhat shifted to the right into the flow). The input parameters mean `N` the number of particles, `ug` the lower boundary and `og` the upper boundary of the domain, where particles are injected. The particles are to be distributed equidistantly in the area at the lefthand side of the flow between `ug` to `og`. The parameters `N, ug, og` as well as `delt_n` (necessary in `streaklines`) to be set in the initialization function `progpar`.

4. `[part_x, part_y] = particletrace(U, V, imax, jmax,`
   `delx, dely, delt, part_x, part_y, problem):`
   The injected particles at time-step t_0 are to be calculated at each further time-step t_n. Particle, which leave the domain $\omega = [0, imax\delta x] \times [0\cdot, jmax \cdot \delta y]$ are removed.

5. `[part_x, part_y] = streaklines(U, V, imax, jmax, delx, dely, delt,`
   `part_x, part_y, N, ug, og, delt_n, problem):`
   At every time step `delt_n` new particles are injected into the flow by means of `set_particles N`, whose position is to be calculated for each time step t_n by the function `new_position`.

6. `particlevis(part_x, part_y, problem):`
   The calculated particle positions are to be visualized after the time loop. This can take place either as an animation or as a suitable plot. You should draw also the obstacle areas.

Test your particle simulation with the Karmann' vortex street and the following parameters:

```
imax = 64      jmax = 16      delx = 0.2      dely = 0.2
delt = 0.02    T_end = 20.0
epsi = 0.01    omg = 0.7      alpha = 0.9     itermax = 150
GX = 0.0       GY = 0.0       nu = 1.3414e-5
U_I = 1.0      V_I = 0.0      P_I = 0.0
wl = 3         wr = 3         wt = 1          wb = 1
N = 16         ug = 2 dely    og = 14 dely    delt_n = 4
```