E. A Numerical Solver for Hydrodynamic Flow

E.1. Introduction

This section presents a more involved example of parallel simulation of incompressible hydrodynamic flow. Although the model itself is much more complicated than that for diffusion we can use many of the techniques introduced above to discretize the model and parallelize the resulting algorithms.

The flow of a time dependent incompressible fluid, a fluid with constant density as for example water under most conditions, can be described by the two basic equations from hydrodynamics:

$$\nabla \cdot V = 0\,, \qquad [21]$$

$$\frac{\partial V}{\partial t} = -(V \cdot \nabla)V - \nabla P + \nu \nabla^2 V\,. \qquad [22]$$

where V is the velocity, t time, P the pressure, and ν the kinematic viscosity which will be assumed to be constant. In the first equation the conservation of mass is expressed; it states that the density at a point in space can only change by in- or out- flow of matter. The second equation is the standard Navier-Stokes equation which expresses the conservation of momentum. This equation describes the velocity changes in time, due to convection $(V\cdot\nabla)V$, spatial variations in pressure ∇P, and viscous forces $\nu\nabla^2 V$. Both equations can be solved analytically only in a very few, simple, cases. In most cases these equations can only be solved by simulation. There is a wide variety of numerical methods available for solving these equations, which already indicates that there is still no perfect method within reach. A good overview of the numerical methods for solving the hydrodynamic equations can be found in [7].

E.2. The numerical solver: finite differencing

In this section it will be demonstrated how the hydrodynamic equations can be solved using the Marker-and-Cell technique [8]. This method is based on a finite difference approximation of the hydrodynamic equations. This method is chosen as an example for several reasons. It is conceptually a simple method, the original equations [21,22] are converted straightforward into finite difference equations, parallellization of the algorithm is relatively easy, extension to three dimensions is trivial, and most types of boundary conditions can relatively easy be specified. Eq. [22] can be written as a set of partial differential equations:

$$\frac{\partial u}{\partial t} = -\frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} - \frac{\partial P}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \qquad [23]$$

$$\frac{\partial v}{\partial t} = -\frac{\partial uv}{\partial x} - \frac{\partial v^2}{\partial y} - \frac{\partial P}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right). \qquad [24]$$

Eq. [21] can be written as:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0\,, \qquad [25]$$

where V of Eqs. [21] and [22] is represented in two dimensions by the vector (u, v). The extension to 3D, where V is represented by a vector (u, v, w), is done analogous to the two-dimensional equations by adding a z-coordinate and an equation for $\partial w/\partial t$ in Eqs [23-25]. An additional equation for the pressure P, a Poisson equation, can be obtained by differentiation and addition of Eqs. [23,24]:

$$\nabla^2 P = -\frac{\partial^2 u^2}{\partial x^2} - 2\frac{\partial^2 uv}{\partial x \partial y} - \frac{\partial^2 v^2}{\partial y^2} - \frac{\partial D}{\partial t} + \nu\left(\frac{\partial^2 D}{\partial x^2} + \frac{\partial^2 D}{\partial y^2}\right), \qquad [26]$$

where D is the divergence:

$$D = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}. \qquad [27]$$

A feature of this equation is that in the mass conservation Eq. [25] it is stated that D has the value zero.

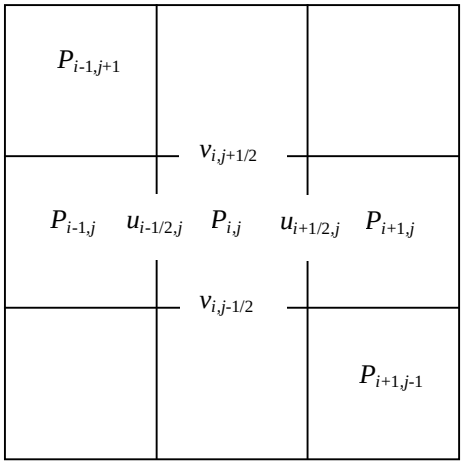

*Figure 20: Diagram of the Marker-and-Cell structure, showing cells in the neighborhood of lattice position (*i,j*) (after[7]).*

In the Marker-and-Cell technique the finite difference approximation is applied to solve the Eqs. [23-25]. In this method space is subdivided into cells with length Δx, Δy. In Figure 20 a diagram is shown of this structure. The velocities are located at the cell faces, while the pressure is located at the cell center. The cells are labeled with an index (i, j), $P_{i,j}$ is the pressure at the cell center, while $u_{i+1/2}$ denotes the velocity in the x-direction between the cells (i, j) and (i+1, j) etc. The finite difference expressions of Eqs. [23, 24] are:

$$\begin{aligned} u_{i+1/2,j}^{n+1} = u_{i+1/2,j} + \Delta t\Bigg[& -\frac{1}{\Delta x}\left(u_{i+1,j}^2 - u_{i,j}^2\right) \\ & -\frac{1}{\Delta y}\left((uv)_{i+1/2,j+1/2} - (uv)_{i+1/2,j-1/2}\right) \\ & -\frac{1}{\Delta x}\left(P_{i+1,j} - P_{i,j}\right) - \nu\Bigg(\frac{1}{\Delta x^2}(u_{i+3/2,j} + 2u_{i+1/2,j} + u_{i-1/2,j}) \\ & -\frac{1}{\Delta y^2}(u_{i+1/2,j+1} - 2u_{i+1/2,j} + u_{i+1/2,j-1})\Bigg)\Bigg] \end{aligned} \qquad [28]$$

$$
\begin{aligned}
v_{i,j+1/2}^{n+1} = v_{i,j+1/2} + \Delta t\Bigg[& -\frac{1}{\Delta y}\left(v_{i,j+1}^2 - v_{i,j}^2\right) \\
& -\frac{1}{\Delta x}\left((uv)_{i+1/2,j+1/2} - (uv)_{i-1/2,j+1/2}\right) \\
& -\frac{1}{\Delta y}\left(P_{i+1,j} - P_{i,j}\right) + \nu\Bigg(\frac{1}{\Delta x^2}(v_{i,j+3/2} + 2v_{i,j+1/2} + v_{i,j-1/2}) \\
& -\frac{1}{\Delta y^2}(v_{i+1,j+1/2} - 2v_{i,j+1/2} + v_{i-1,j+1/2})\Bigg)\Bigg]
\end{aligned}
\qquad [29]
$$

where $u_{i+1/2,j}^{n+1}$ and $v_{i,j+1/2}^{n+1}$ are the velocities obtained by advancing the previous velocities $u_{i+1/2,j}$ and $v_{i,j+1/2}$ one time step. Values like $u_{i+1,j}$ are obtained by using the average: $u_{i+1,j}$ =½($u_{i+3/2,j}$ + $u_{i+1/2,j}$) and product terms are evaluated as the product of averages:

$$
(uv)_{i+1/2,j+1/2} = \frac{1}{2}(u_{i+1/2,j} + u_{i+1/2,j+1}) \cdot \frac{1}{2}(v_{i+1,j+1/2} + v_{i,j+1/2}) . \qquad [30]
$$

The results $u_{i+1/2,j}^{n+1}$ and $v_{i,j+1/2}^{n+1}$ do not necessarily satisfy the mass conservation Eq. [25]. In the Marker and Cell method an iterative process is used in which the cell pressures are modified to obtain a value $D = 0$. For this purpose in each cell (i, j) the value of $D_{i,j}$, the finite difference form of Eq. [27] is determined:

$$
D_{i,j} = \frac{1}{\Delta x}\left(u_{i+1/2,j} - u_{i-1/2,j}\right) + \frac{1}{\Delta y}\left(v_{i,j+1/2} - v_{i,j-1/2}\right). \qquad [31]
$$

If the value $D_{i,j}$ is below a certain level *tolerance*, the flow is locally incompressible and it is not necessary to change the velocities at the cell face. If $D_{i,j}$ is above the level *tolerance*, the pressure is changed with a small value:

$$
\Delta P_{i,j} = -\beta D_{i,j} \qquad [32]
$$

where β is related to a relaxation factor β_0:

$$
\beta = \beta_0 / 2\Delta t(1/\Delta x^2 + 1/\Delta y^2) . \qquad [33]
$$

It is necessary to use a value $\beta_0 < 2$, otherwise the iteration process is not stable (compare this to the *Courant stability* derived in the previous sections). More details about the finite difference iterations using relaxation factors and its stability can be found in [3] and [9]. A plausible value for β_0 is 1.7. Once $\Delta P_{i,j}$ is determined for each cell (i,j) it is necessary to add it to $P_{i,j}$ and to adjust the velocity components at the faces of the cell (i, j):

$$
\begin{aligned}
& u_{i+1/2,j} \rightarrow u_{i+1/2,j} + (\Delta t / \Delta x)\Delta P_{i,j} \\
& u_{i-1/2,j} \rightarrow u_{i-1/2,j} - (\Delta t / \Delta x)\Delta P_{i,j} \\
& v_{i,j+1/2} \rightarrow v_{i,j+1/2} + (\Delta t / \Delta x)\Delta P_{i,j} \\
& v_{i,j-1/2} \rightarrow v_{i,j-1/2} - (\Delta t / \Delta x)\Delta P_{i,j}
\end{aligned} .
\qquad [34]
$$

The process is repeated successively in all cells until no cell has a divergence D greater than *tolerance*. The complete iteration process is summarized in Algorithm 6.

When the iteration process converges and the mass conservation equation is satisfied, a next time step can be done. It can be demonstrated [10] that adjusting P and V in this iterative process is equivalent with solving the Poisson Equation [26].

```
Advance the finite difference Eqs. 28 and 29 one time step Δt
do
   determine D_{i,j} (Eq. 21) for each cell
   if (D_{i,j} > tolerance) then
      determine ΔP_{i,j} (Eq. 22)
      add ΔP_{i,j} to P_{i,j}
   adjust velocity components (Eq. 24)
   end if
while (there are cells present with D > tolerance)
goto next time step
```

Algorithm 6: Pseudo code of the Marker and Cell method.

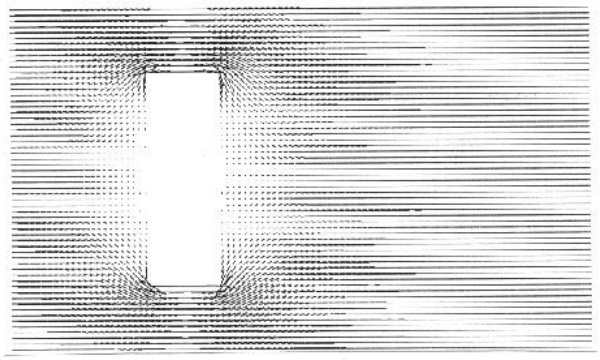

Figure 21: A 2D slice of a flow pattern about an obstacle for a large value of $\nu = 1.0$ *in a 3D lattice consisting of* 100^3 *cells.*

Before Eqs. [23-25] can be solved it is necessary to specify the boundary conditions, otherwise the problem is ill posed. Five common types of boundary conditions are: rigid free-slip walls, rigid no-slip walls, inflow, outflow boundaries, periodic boundaries, moving boundaries, free boundaries (for example a moving water surface). The specification of these boundary conditions, in general, is a difficult problem especially in the case of complex geometry (see [7]). If the value of ν of the kinematic viscosity is chosen large enough the equations are stable and it is possible to compute the flow pattern about an obstacle. In Figure 21 the flow pattern about a rectangular obstacle is determined. The flow, in this example, is directed in the positive x-direction. In this example four types of boundary conditions are used. At the cells situated at the obstacle the "rigid no-slip wall" condition is used by setting the u and v component to the value zero. There is an inflow boundary where the velocity is set a certain input value. Finally, there is an outflow boundary where the velocity V^{n+1} in cell (i, j) is set to the V^{n+1} velocity of the neighboring upstream cell $(i\text{-}1, j)$, while for the two other borders of the lattice periodic boundary conditions are used. If ν is chosen too low, the convective term $(V\cdot\nabla)V$ in Eq. 22 starts to dominate. For low values of ν the iteration process becomes unstable, furthermore truncation errors, which are unavoidable in finite difference approximations, may lead to instabilities. More details about the stability analysis, in which the lower limit of ν is determined, of the iteration process shown can be found in [8]. An example of a flow pattern generated with a lower value of ν is shown in Figure 22. In this example the same type of boundary conditions are used as in the previous example.

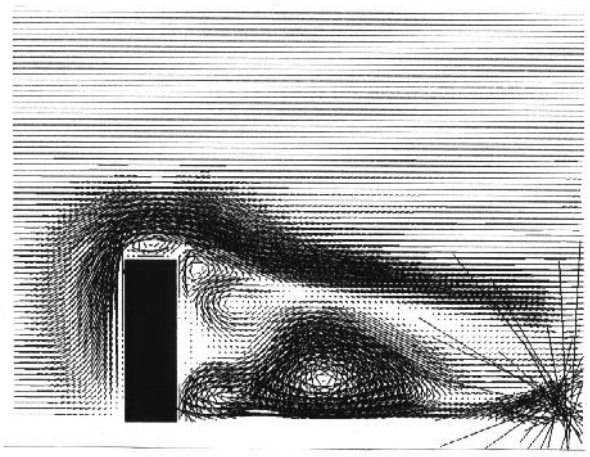

Figure 22: Example of a flow pattern about an obstacle for a low value of $\nu = 0.20$ *in a lattice consisting of* 100^3 *cells.*

In this example it can be seen that eddies start to develop behind the obstacle and furthermore an instability develops (which can be recognized by the strange size of the velocity vectors), due to truncation errors, at the very right of the picture.

In many papers on hydrodynamics the range of ν that can be simulated with the model, before instabilities occur, is expressed in the range of possible Reynolds numbers. The relation between ν and the Reynolds number Re is given by the equation:

$$\mathrm{Re} = \frac{V_0 L}{\nu}, \qquad [35]$$

where V_0 is the typical velocity of the system (for example the input velocity) and L the typical dimension resolved by the system. In the experiments shown in Figure 21 and Figure 22 a plausible choice of L is the height of the obstacle (number of cells necessary to represent the height times Δy. The choice of L depends on the geometry of the obstacle, more details can be found in the references [7]. Especially for an irregular geometry a plausible choice of L can become problematic.

E.3. The parallel implementation

From the finite difference Eqs.[28, 29, 31] used in the algorithm of the Marker and Cell method it can be seen that for the computation of the values of u^{n+1}, v^{n+1}, and D^{n+1} only the u, v, P values of the 8 nearest neighbors are required (see Figure 20). This locality can be used in a parallel implementation of the Marker and Cell algorithm. In this algorithm a successive updating scheme is used, where the P-values (and also the u and v values) of the cells neighboring cell (i, j) are in the states of Eq. [36],

$$\begin{matrix} P^{n}_{i-1,j+1} & P^{n}_{i,j+1} & P^{n}_{i+1,j+1} \\ P^{n+1}_{i-1,j} & P^{n}_{i,j} & P^{n}_{i+1,j} \\ P^{n+1}_{i-1,j-1} & P^{n+1}_{i,j-1} & P^{n+1}_{i+1,j-1} \end{matrix} . \qquad [36]$$

In a parallel implementation this locality can be used and the lattice of cells can be subdivided into a, square grid where each grid site is located on a different processor. In Figure 23 the situation is depicted for a lattice which is subdivided into four parts and where the lattice is mapped onto a grid of 2×2 processors. Each processor is bordered by boundary strips and corners which contain copies of the values of the adjacent cells. In the parallel version of Algorithm 6 a four-colored checkerboard domain partitioning [11] is used which accounts for the correct state (n or $n + 1$) of the adjacent cells as shown in Eq.

36, when cell (*i*, *j*) is updated. In the parallel version of Algorithm 6 the lattice of cells is updated in four successive phases. This is shown in Algorithm 7.

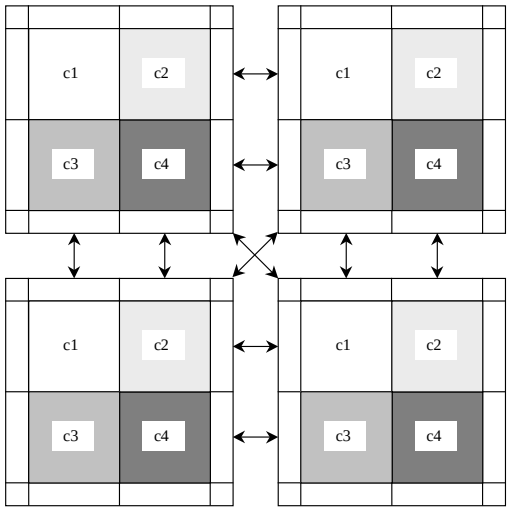

Figure 23: Mapping of the lattice of cells onto a grid of 2×2 processors using a four-colored checkerboard domain partitioning. The arrows indicate the exchange of boundary strips and boundary corners.

```
for each time step do
   exchange boundary strips between regions c1 and c2
   exchange boundary strips between regions c1 and c3
   exchange boundary corners between regions c1 and c4
   update of region c1 using algorithm 1
   exchange boundary strips between regions c2 and c1
   exchange boundary strips between regions c2 and c4
   exchange boundary corners between regions c2 and c3
   update of region c2 using algorithm 1
   exchange boundary strips between regions c3 and c1
   exchange boundary strips between regions c3 and c4
   exchange boundary corners between regions c3 and c2
   update of region c3 using algorithm 1
   exchange boundary strips between regions c4 and c2
   exchange boundary strips between regions c4 and c3
   exchange boundary corners between regions c4 and c1
   update of region c4 using algorithm 1
endfor
```

Algorithm 7: Pseudo code for a parallel numerical solver of the hydrodynamical equations

References

1. Hoekstra, A.G.: Syllabus APR Part 1 : Introduction to Parallel Computing. University of Amsterdam, (1999)
2. Stoer, J.,Bulirsch, R.: Introduction to Numerical Analysis.
3. Press, W.H., Flannery, B.P., Teukolsky, S.A.,Verrerling, W.T.: Numerical Recipes in C, the art of scientific computing.
4. Fox, G.C., Williams, R.D.,Messina, P.: Parallel Computing Works! (1994)

5. Dongarra, J.J.,Walker, D.W.: Constructing Numerical Software Libraries for High Performance Computer Environments. In: A. Zomaya (eds.): Parallel & Distributed Computing Handbook. (1998)
6. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C.,Van der Vorst, H.v.d.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition. Philadelphia, PA (1994)
7. Roache, P.J.: Computational Fluid Dynamics. Hermosa Publishers, Albequerque (1976)
8. Hirt, C.W.,Cook, J.L.: Calculating three-dimensional flow around structures and over rough terrain. J. Comput. Phys. **10** (1972) 324-340
9. Ames, W.F.: Numerical Methods for Partial Differential Equations. Academic Press, New York (1977)
10. Viecelli, J.A.: A computing method for incompressible flows bounded by moving walls. J. Comput. Phys. **8** (1971) 119-143
11. Fox, G.C., Johnson, M., Lyzenga, G., Otto, S., Salmon, J.,Walker, D.: Solving Problems on Concurrent Processors. Pretince Hall, (1988)
12. Wilkinson, J.H.: Error analysis of direct methods of matrix inversion. J. Assoc. Comp. Mach. **8** (1961) 281 - 330
13. Lawson, C., Hanson, R., Kincaid, D.,Krogh, F.: Basic Linear Algebra Subprograms for FORTRAN usage. ACM TOMS (Transactions On Math. Software) **5** (1979) 308-325
14. Golub, G.H.,Loan, C.F.v.: Matrix Computations. John Hopkins Univ. Press, (1989)
15. Freeman, T.L.,Philips, C.: Parallel Numerical Algorithms. Prentice Hall, (1992)