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

Now we want to proceed with the implementation of the algorithm described on the last assignment. As first we assort the required problem sizes and the used data structures. Furthermore we give the structures of the Matlab-functions, which represent the individual components of the program, and treat a first example, the so-called „Driven-Cavity problem“.

1 Problem sizes and data structures

The algorithm described on the second assignment requires the following quantities, which should be made available in an input file at the programm start:

- Geometry sizes:

`imax`	number of inner cells in x-direction,
`jmax`	number of inner cells y-direction,
`delx`	δx, size of one cell in x-direction,
`dely`	δy, size of one cell in y-direction.

- Sizes for the time-iteration:

`T_end`	final time T_{end},
`delt`	δt, time-step size.

- Parameter for the pressure-iteration:

`itermax`	max. number of pressure-iterations per time-step,
`epsi`	ϵ, precision criterion for the pressure iteration (`res < epsi`),
`omg`	ω, relaxion parameter for SOR iteration,
`alpha`	α, discretisation parameter (see assignment 2).

- Problem dependend sizes:

`U_I, V_I, P_I`	initial values for velocities and the pressure,
`GX, GY`	g_x, g_y, forces, e.g. gravitation,
`nu`	ν, viscosity $(\frac{1}{Reynoldsnumber})$.

You should use the following fields (matrices) of the dimension `(imax + 2)` $\times$ `(jmax + 2)` as data structures:

- `U`: velocity in x-direction,
- `V`: velocity in y-direction,
- `P`: pressure,
- `RHS`: RightHand Side f for the pressure-iteration,
- `F, G`: dummy variables F and G.

Within Matlab fields like `U, V, P, ...` can only be indexed by `1:n+2` and NOT by `0:n+1`. *So an index shift must be done*!

2 The programm and its components

The structure of the program:

1. Input of the parameters.
2. Initialization.
3. `T := 0`.
4. While `T` $\leq$ `T_end`:
   (a) Calculate the boundary values of `U` and `V`.
   (b) Calculate `F, G` and `RHS`.
   (c) Solve the linear equation system for `P` with the SOR method.
   (d) Calculate `U` and `V`.
   (e) `T` := `T` + `delt`.
5. Graphical output of the results (visualization).

The individual components of the program are to be implemented thereby as Matlab routines:

- `progpar()`:
  The sizes `imax, jmax, delx, dely, delt, T_end, itermax, epsi, omg, alpha, nu, GX, GY, U_I, V_I, P_I` are set in this script.
- `[U,V,P] = initgrid(imax,jmax,U_I,V_I,P_I)`:
  The fields `U, V, P` are initialized on the hole domain with constant values `U_I, V_I` and `P_I` as start values.

- `[U,V] = boundary_values(U,V,imax,jmax)`:
  The boundary values of the fields `U` and `V` are set according to the formulas of section 4.4 of assignment 2.

- `[F,G] = calc_FG(U,V,imax,jmax,delt,delx,dely,GX,GY,alpha,nu)`:
  Calculation of `F` and `G`. At the boundaries the formulas of section 4.4 must be used.

- `RHS = calc_RHS(F,G,imax,jmax,delt,delx,dely)`:
  Calculation of the righthand side of the pressure equation.

- `[P,it,res] =`
  `SOR(P,RHS,U,V,GX,GY,imax,jmax,delx,dely,epsi,itermax,omg,nu)`:
  SOR iteration for the pressure (Poisson) equation. The iteration is aborted, if the residue `res` falls below the tolerance limit `epsi` or the max. iteration number `itermax` is achieved. The return values of this function are the number of iterations , the residue `res` and the new pressure matrix `P`.

- `calc_UV(F,G,P,imax,jmax,delt,delx,dely)`:
  The new velocities are calculated.

- `visual(U,V,P)`:
  Use the Matlab instruction `mesh` to visualize the velocity components `U, V` and the pressure matrix `P` from the last time-step. Consider that the boundary values were artificially set and can falsify the result. Therefore these values should be omitted within the visualization. One can take a look at the velocity field with the instruction `quiver`. The several pictures should be separated by the instruction `pause`. Alternatively they can be put into a single plot using `subplot`.

In the main program `incompvis` the algorithm indicated above should be implemented. The program `continue_simulation` should continue an already started simulation till a new time step `T_end` is reached. There are two possibilities for the SOR method. **Both should be implemented:**

- As first the SOR method can be accomplished grid-oriented, i.e. one goes point by point over the grid and calculates the new pressure at each grid point. This can be realized by the use of several `for`-loops. To guarantee the stability of the variant, the boundary values for `P` must be set in each iteration step, e.g.:

$$p_{0,j}^{it+1} = p_{1,j}^{it},\ p_{imax+1,j}^{it+1} = p_{imax,j}^{it} \quad \text{for } j = 1,\dots,jmax,$$

$$p_{i,0}^{it+1} = p_{i,1}^{it},\ p_{i,jmax+1}^{it+1} = p_{i,jmax}^{it} \quad \text{our } i = 1,\dots,imax.$$

- The second (in Matlab faster) variant is, to set up an equation system $Ax = b$, whereby the vector x contains the components of the pressure matrix `P` and the vector b is the righthand side of the pressure-Poisson-equation. After that you can use the implemented SOR method (implemented in assignment 1) or as well the PCG method (also implemented in assignment 1) for solving the equation system.

3 The first example: „Driven Cavity “

As a first example we want to simulate a typical CFD problem, the so-called „Driven-Cavity problem“. It deals with a pot filled with a fluid, and a band which is pulled along with constant rate of speed. Thereby no-slip conditions are used at all four boundaries. To simulate the pulled band, the velocity at the upper boundary in x-direction u is set to 1. In the program this can be realized as follows:

```
U(2:imax+1,jmax+2) = 2 - U(2:imax+1,jmax+1)
```

All other parameters from the input file should be used:

```
imax = 16      jmax = 16      delx = 0.2      dely = 0.2
delt = 0.02    T_end = 0.2
epsi = 0.01    omg = 1.7      alpha = 0.12    itermax = 150
GX = 0         GY = 0         nu = 0.4
U_I = 0        V_I = 0        P_I = 0
```

4 Furher tasks

Task 1 *Use for the solution of the linear equation $Ax = b$ both the SOR method and the method of the conjugated gradients (PCG). Both are already implemented in assignment 1! Make a statistics for the number of iterations and the required calculation time per time-step for each method and submit these statistic in a suitable graphical form.*

Task 2 *Within Matlab one can prepare small movies with the instruction* `movie`*. To produce a movie, which shows the temporal development of the fluid, one must include the visualization routine into the time loop of the algorithm and store the picture in each time-step. In addition one must scale the axes suitable, to have no cracks in between the individual pictures.*

Task 3 *To get to know the influence of the parameters more closely, change some of the given parameters, and play with the example!*