ME-474 Numerical Flow Simulation

Comments to the exercise: 1D steady convection-diffusion

Fall 2021

After discretization with one of the four proposed schemes, we are left to solve a linear system $A\phi = b$. Regardless of the chosen scheme, we have

$$[A]_{i,i} = a_P, \quad [A]_{i,i-1} = -a_W, \quad [A]_{i,i+1} = -a_E \qquad \text{for} \quad 3 \le i \le N-1$$

where the indices 1 and N denote the boundary points. In addition, imposition of Dirichlet boundary conditions requires $[b]_1 = \phi_0$ and $[b]_N = \phi_L$. Accordingly, $[A]_{11} = 1$ and $[A]_{NN} = 1$ such that

$$\begin{aligned} [A]_{11}[\phi]_1 = [b]_1 \quad &\Rightarrow \quad 1 \cdot [\phi]_1 = \phi_0, \\ [A]_{NN}[\phi]_N = [b]_N \quad &\Rightarrow \quad 1 \cdot [\phi]_N = \phi_L. \end{aligned}$$

For the second line ($i = 2$) of the matrix A, and the expression of the right-hand side, we need to distinguish several cases depending on the discretization scheme:

CD and UD

Knowledge of the solution at WW is never required in the determination of the one at P, thus the lines 2 and $N-1$ of the matrix A are no different from the other "inner" lines 3 to $N-2$: $[A]_{i,i} = a_P$, $[A]_{i,i-1} = -a_W$, $[A]_{i,i+1} = -a_E$.

As neither natural nor deferred source is present, the right-hand side b is null everywhere except at boundary nodes. As b does not depend on ϕ, we can directly solve the linear system $A\phi = b$ with the command "\" of Matlab without iterating on ϕ. Remember also that solving the linear system does not imply to construct explicitly the inverse of A.

QUICK + dc and TVD + dc

For the next two schemes QUICK + dc and TVD + dc, we must iterate on ϕ until convergence because of the deferred correction. Thus, $b = b(\phi^*)$, where the asterisk denotes the values at the previous iteration. Who says iterations says initial guess. Of course, in general the closer the initial guess to the final solution, the quicker the convergence. For instance, it is good if this initial guess satisfies the boundary conditions and has a physically plausible shape. For both schemes, we write for the right-hand side terms

$$[b]_i = S([\phi^*]_{i-2}, [\phi^*]_{i-1}, [\phi^*]_i, [\phi^*]_{i+1}), \qquad \text{for } 3 \le i \le N-1.$$

Because this requires the knowledge of the solution at node $i-2$ (node WW), the construction of the second line ($i = 2$) of A and b needs a special treatment (since the node $i = 2-2 = 0$ does not exist). It is detailed as follows:

QUICK + dc: details

The general governing equation is:

$$F\phi_e - F\phi_w = D(\phi_E - \phi_P) - D(\phi_P - \phi_W).$$

At node $i = 2$, we propose $\phi_{WW} = 2\phi_W - \phi_P$, thus ϕ_w expresses in particular:

$$\phi_w = \frac{-\phi_{WW} + 6\phi_W + 3\phi_P}{8} = \frac{-2\phi_W + 3\phi_P + 6\phi_W + \phi_P}{8} = \frac{4\phi_W + 4\phi_P}{8} = \frac{\phi_W}{2} + \frac{\phi_P}{2}.$$

However, ϕ_e remains as usual:

$$\phi_e = \phi_P + \frac{-\phi_W^* - 2\phi_P^* + 3\phi_E^*}{8}.$$

As a consequence, at node $i = 2$ we have:

$$\begin{aligned} F\phi_e - F\phi_w &= D(\phi_E - \phi_P) - D(\phi_P - \phi_W) \Leftrightarrow \\ F\left(\phi_P + \frac{-\phi_W^* - 2\phi_P^* + 3\phi_E^*}{8}\right) - F\left(\frac{\phi_W}{2} + \frac{\phi_P}{2}\right) &= D(\phi_E - \phi_P) - D(\phi_P - \phi_W) \Leftrightarrow \\ \left(F + 2D - \frac{F}{2}\right)\phi_P &= (D)\,\phi_E + \left(\frac{F}{2} + D\right)\phi_W + S_{dc} \end{aligned}$$

where

$$S_{dc} = \frac{F}{8}\left(\phi_W^* + 2\phi_P^* - 3\phi_E^*\right).$$

As said, these equations only hold at $i = 2$, and we write accordingly

$$\left(F + 2D - \frac{F}{2}\right)[\phi]_2 = (D)\,[\phi]_3 + \left(\frac{F}{2} + D\right)[\phi]_1 + S_{dc},$$

where

$$S_{dc} = \frac{F}{8}\left([\phi^*]_1 + 2[\phi^*]_2 - 3[\phi^*]_3\right).$$

In the code, this corresponds to:

```
%--- Boundary condition in CV 2
   A(2,1) = -(D+F/2);      % West
   A(2,3) = -(D);          % East
   A(2,2) = 2*D + F - F/2; % P
   b(2) = F/8 *(Phi(1) +2*Phi(2) -3*Phi(3)); % deferred correction
```

TVD + dc: details

For the TVD scheme, the second line of the matrix is similar to others between 3 and $N - 1$, namely $[A]_{i,i} = a_P$, $[A]_{i,i-1} = -a_W$, $[A]_{i,i+1} = -a_E$ also holds for $i = 2$. However, at node $i = 2$, $\phi_{WW} = 2\phi_W - \phi_P$ thus $r_w = 1$.

About this scheme, one should also be careful with the quantities $\phi_E - \phi_P$ and $\phi_P - \phi_W$, that appear at the denominators of r_e and r_w but can be null. Dealing with infinity is numerically delicate and requires a special treatment. In the code, we propose to test systematically the nullity of $\phi_E - \phi_P$ and $\phi_P - \phi_W$. If they are null indeed, and if in addition the numerator is not null, we set the corresponding r to a very high value, for instance 10^{30}, still treatable by Matlab. On the other hand, if the numerator is null too, the corresponding r is set to 0 to avoid the undetermined form "NaN".

In the code, this corresponds to:

```
%--- Define r (ratio of gradients) with test to avoid dividing by 0
    if (Phi(i+1)-Phi(i))==0
        if abs(Phi(i)-Phi(i-1))>0,
            r_e = 1e30 * sign(Phi(i)-Phi(i-1));
        else
            r_e = 0;
        end
    else
        r_e = (Phi(i)-Phi(i-1)) / (Phi(i+1)-Phi(i));
    end
    if (Phi(i)-Phi(i-1))==0
        if abs(Phi(i-1)-Phi(i-2))>0
            r_w = 1e30 * sign(Phi(i-1)-Phi(i-2));
        else
            r_w = 0;
        end
    else
        r_w = (Phi(i-1)-Phi(i-2)) / (Phi(i)-Phi(i-1));
    end
```